JUNOS OS IPSEC VPN WITH PKI CERTIFICATES PRIMER
Configuration and Troubleshooting
# Table of Contents

- Introduction ................................................................. 4
- Scope .................................................................................. 4
- Targeted Audiences ............................................................ 4
- Glossary of Terms .............................................................. 4
- Overview ............................................................................ 6
- Fundamentals of PKI in Junos OS ......................................... 6
  - Components ..................................................................... 6
- Certificate Life Cycle Management/Administration ............... 7
  - Generation of Keys and Cert Request ................................. 8
  - Cert Enrollment ............................................................... 8
  - Certificate Validation and Revocation Checking ................. 8
  - Certificate Renewal .......................................................... 8
- General Usage ................................................................. 8
- SSL Usage .......................................................................... 9
- IPsec/IKE Usage ............................................................... 9
- Administering PKI in Junos OS ........................................... 11
- Network Diagram ............................................................. 11
- Configuration Overview ..................................................... 12
- Setting the Device's Basic Configuration for PKI ................. 12
  - Configuring Interface IP Addresses .................................. 12
  - Configuring Default Route .............................................. 12
  - Setting System Time ...................................................... 13
  - Setting DNS Configuration ............................................. 13
- Generating the Certificate Request ....................................... 14
  - Creating a Trusted CA Profile ......................................... 14
  - Generating the Cert Request ........................................... 15
- Submit Cert Request to the CA and Retrieve Certs ............... 17
- Loading the Local Cert, CA Cert, and CRL ......................... 17
  - Uploading the Files to Local Storage ................................. 17
  - Loading the Local Cert .................................................... 18
  - Loading the CA Cert ....................................................... 18
  - Loading the CRL ............................................................. 18
  - Verify All Certs Loaded ................................................... 18
- Using the Cert in an IPsec VPN ........................................... 21
  - Steps to Configure ........................................................ 22
- IPsec VPN Configuration Example for Junos OS ................. 22
  - Configuring Security Zones and Assigning Interfaces to the Zones ......................................................... 22
  - Configuring Host-Inbound Services for Each Zone .......... 23
  - Configuring Address Book Entries for Each Zone ............ 23
  - Configuring IKE Phase 1 Proposal ................................... 23
APPLICATION NOTE - Junos OS IPsec VPN with PKI Certificates Primer

Table of Figures

Figure 1: Network diagram ................................................................. 11
Introduction

Juniper Networks® Junos® operating system runs on Juniper Networks J Series Services Routers and SRX Series Services Gateways and provides not only a powerful OS, but also a rich IP services tool kit. Junos OS has enhanced security and VPN capabilities via Juniper’s firewall/IPsec VPN platforms, which include the Juniper Networks SSG Series Secure Services Gateways.

This application note documents many of the public key infrastructure (PKI) concepts, specifications, usage scenarios/decisions, and debugging in the Junos OS. It is meant to be more than just a step-by-step guide to configuring and using PKI. Additional details regarding digital certificates can be found in the Junos OS System Basics Guide.

This document includes an overview of PKI in the Junos OS, and includes details on defining the architecture for VPNs based on PKI authentication. Sample configuration scenarios and details on PKI administration are included as well as troubleshooting information. This document also contains Frequently Asked Questions (FAQs) on the Junos OS PKI, appropriate for use with RFP responses. There are also appendices on CA server usage and DoD PKI usage, strictly for reference.

This document is not a primer on the general concepts of PKI. There are many websites that can provide documentation on basic PKI concepts. For more information on crypto, RSA, and PKI, visit www.rsasecurity.com/rsalabs.

Scope

This document applies to the following devices:

- J Series devices running:
  - Junos OS 9.4 and later
  - Junos OS 8.5 through 9.3
- SRX Series devices

Targeted Audiences

This document is intended for network design and security engineers, and implementation partners supporting customers who require secure connectivity over public networks.

Glossary of Terms

This section of the document defines some terms that are repeatedly used throughout this application note. It does not encompass all PKI terminology, but does address most topics related to PKI usage in IKE for the Junos OS.

CA—Certificate authority is the server (or set of servers) that signs certificates for VPN gateways and user systems (for client RAS VPN). CAs also generate CRLs, which are lists of revoked certificates. A CA acts as the trusted third party between two VPN gateways that are authenticating each other using certificates.

Certificates (certs)—These are the binding of an entity’s identity and public key into a file. The CA digitally signs this file so third parties can validate the entity’s ID and public key as authentic, and so others can validate that entity’s ID and public key if they trust that CA. Certificates have a finite lifetime. Each certificate has a start and end time that defines the lifetime. At such time that the lifetime expires, the certificate is considered invalid, and a certificate renewal or a new certificate request is required.

CRL—The certificate revocation list is a list of certificates that are signed and now prematurely invalid. The CA periodically publishes this list.

CDP—The CRL Distribution Point (CRL-DP) is the location to retrieve a CA's latest CRL. This is usually an LDAP server or HTTP (Web) server. The CDP is normally expressed as an “ldap://host/dir” or “http://host/path” URL.

DN—Distinguished name is the set of fields and values that uniquely defines a certificate and VPN gateway or RAS VPN
client identity. This is sometimes called the “Subject” of the certificate. This DN identity can be used as the IKE ID. The DN usually has the following form:

- **CN**= user name, server DNS name, or most any uniquely identifying string
- **OU**= organizational unit (“Sales”)
- **O**= organization (“Juniper Networks”)
- **L**= locality (“San Francisco”)
- **S**= state (“CA”)
- **C**= country (“US”)

DNS resolver—This refers to the Domain Name System settings that should be set on the Juniper VPN device to help resolve FQDN names into IP addresses. Many CAs and certs use hostnames that must be resolved into IP addresses.

FQDN—A fully qualified domain name is usually the name given to devices on the Internet, including the DNS-based zone in which they are located. Examples include www.juniper.net, ocsp.chemistry.nwu.edu, nsgw1.dklein.org, ftp1.whitehouse.gov, and ca2.nit.disa.mil.

IKE—Internet Key Exchange is the ISAKMP/Oakley-based process used by two VPN gateways to identify and authenticate each other. In addition, key generation for packet-level authentication (integrity) and encryption (privacy) is handled during IKE. The IKE RFC defines two basic gateway authentication mechanisms: 1) preshared key (such as a password) and 2) digital certificates (certs) based on RSA private/public key pairs. This document focuses on the second authentication mechanism, which is the use of certs.

IKE ID—The IKE identity is how two VPN peers identify each other. The IKE ID has a type and a value. The most common types and example values are the following:

1. IP Address (21.62.2.252)
2. FQDN—(vpn1.juniper.net)
3. U-FQDN or e-mail address (johndoe@juniper.net)
4. DN—(CN=John Doe, OU=eng, O=Juniper, C=US)

IPsec—This is the protocol used to authenticate, encrypt, and encapsulate IP packets between two VPN/IKE peers and create a tunnel.

NSR—NetScreen-Remote Client is the software for Windows-based PCs or laptops that allows clients to set up a personal VPN to a Junos OS-based device or other IPsec gateway. This is different than a site-to-site VPN, where two VPN devices set up a VPN tunnel between two sites each containing many hosts.

PKI—Public key infrastructure is the set of objects that allows the use of digital certificates to be used between two entities (such as VPN gateways). This usually includes CA, RA, certs, CRLs, CDPs, and OCSP.

PKCS—This is the set of documentation defined by RSA Laboratories on various PKI standards:

- **PKCS7**—Cryptographic Message Syntax Standard defines how messages are encoded and digitally signed. This includes a certificate. This is sometimes referred to as a p7 file.
- **PKCS10**—Certificate Request Syntax Standard defines how a VPN gateway can form a request for a cert that can be sent to a CA. This request usually contains the VPN gateway’s identity and public key, which is referred to as a p10 file. The CA digitally signs it with its own private key and returns a p7 (cert) file.
- **PKCS11**—Cryptographic Token Interface Standard defines how to store certs and private keys on a token card. This is not relevant to Junos OS-based devices but can be relevant to NSR.
- **PKCS12**—Personal Information Exchange Syntax Standard defines how to bundle up an entity’s certificate and public/private key pair into a password protected file, which is referred to as a p12 file. This facilitates moving a user from one machine to another for client VPNs. This standard is not used by Junos OS-based devices but can be relevant to NSR.
OCSP—The online certificate status protocol is the protocol used for a VPN device to contact a validation authority (VA) to check on the validity of a cert. This is a more scalable alternative to the use of CRLs and CDPs.

SCEP—Simple Certificate Enrollment Protocol allows a device to generate a certificate request and automatically submit the request to a CA. Using this protocol requires that both the device and the CA support it. This makes certificate enrollment and re-enrollment easier than manually collecting a PKCS10 from the device and submitting it to a CA. The Junos OS supports SCEP, beginning with version 9.0. Refer to Appendix D for more details.

U-FQDN—The user fully qualified domain name is usually the e-mail address given to users on the Internet or corporate network. Examples include johndoe@juniper.net, user1@org.corp.com, and john.smith@jscorp.com.

Overview

This section of the document addresses the fundamentals of PKI in the Junos OS, such as the various components (private/public keys, certs, CAs, and revocation checking options), certificate life cycle management, and how to use the certs in SSL (informational purposes for future support) and IPsec/IKE. Juniper uses public/private keys within four areas of the Junos OS:

- SSH/SCP (for secure CLI-based administration)
- SSL (for secure web-based administration and HTTPS-based WebAuth for user authentication)
- IKE (for IPsec VPN tunnels)

Currently for the Junos OS, only IKE uses PKI certificates for public key validation and identity binding, with SSL support possible in a future release. SSH/SCP are used exclusively for system administration and rely on the use of out-of-band “fingerprints” for public key identity binding and validation. Details on SSL are out of the scope of this document. The bulk of this document focuses on the administration of PKI and its use of IKE. A brief section on SSL is included for reference.

Fundamentals of PKI in Junos OS

This section describes the basic elements of PKI in the Junos OS, including components of the PKI, certificate life cycle management, and usage within IKE.

Components

There are three main components to administer PKI within the Junos OS:

- CA certificates and authority configuration
- Local certificates, including the device's identity (for example, IKE ID type and value) and private/public keys
- Certificate validation via a CRL

The Junos OS also has three specific types of PKI objects:

- Private/public key pair
- Certificate—three different kinds:
  1. Local cert
  2. CA cert
  3. Pending cert
- CRL (certificate revocation lists)

The local certificate contains the Juniper device’s public key and identity information. A certificate is considered a local cert if the Juniper device possesses the associated private key and that cert was generated based on a cert request from that Juniper device. A pending cert is when a key pair and identity information have been generated into a PKCS10 cert request and manually sent to a CA. Automatic sending of cert requests via SCEP is not supported in the Junos OS until version 9.0R1 (See Appendix D for more details on SCEP). While the Juniper device waits for the cert from the CA, the existing object (key pair plus cert request) is tagged as a certificate request or pending cert. Once the cert is issued by the CA and loaded into the Juniper device, the pending cert is replaced by the newly generated local cert. All other certs loaded into the device are considered CA certs.
The Junos OS can support multiple local certs. This is dependent on the device size. The use of multiple local certs generally occurs where a Juniper device might have VPNs in more than one administrative domain.

All PKI objects are stored in a separate part of persistent memory than the Junos OS image and the system's general configuration. Each PKI object has a unique name or certificate-id given to it from the beginning and maintains that ID until deletion. The certificate-id can be seen with the “show security pki local-certificate” command.

In general, a certificate cannot be copied off of a device. The private key on a device must be generated on that device. And the private key can never be viewed or saved from that device. So PKCS12 files (which contain a cert with the public key and the associated private key) are not supported on Junos OS-based devices. CA certs are generally used to validate certificates received by the IKE peer. If the cert is valid, it is checked against the CRL to determine if the cert has been revoked. In addition to the CA certs themselves, there is a CA profile configuration for each CA cert. This configuration object stores information related to this particular CA and includes the following:

- CA Identity, which is typically the domain name of the CA
- E-mail address to allow for cert requests to be sent directly to the CA
- Revocation settings, which include:
  1. Revocation check enable/disable
  2. Disable revocation check if failed to download CRL
  3. Location of CDP (manual URL setting)
  4. CRL refresh interval

The possible PKI objects and their average sizes are as follows:

- Private/public key pair (1 KB)
- Local Certificate (2 KB)
- CA Certificate (2 KB)
- CA authority configuration (500 Bytes)
- CRL (average size is highly variable depending on how many certs have been revoked by that particular CA—300 bytes up to 2 MB or more)

If there is an average CRL of 10 KB, a Junos OS-based device with one local cert, one CA cert and auth config, and the CRL from that cert has the following flash memory requirements:

- 15.5 KB = 2 KB (local cert) + 1 (key pair) + 2 (CA cert) + 0.5 (CA auth conf) + 10 (CRL)

For certificate chains you would need to include additional CA certs, CA profile configs, and CRLs for each CA in the hierarchy or cross-certified chain. In general, the higher-end Junos OS-based devices—with more persistent memory—can accommodate several local certs and CA chains. However, the smaller devices—with more limited storage capacity—can quickly have their portion of persistent memory allocated for PKI objects fill up. It’s typically recommended that the lower-end devices use only one local cert/key pair, one CA (or one chain of CAs), and one CRL.

**Certificate Life Cycle Management/Administration**

The general life cycle for certificates is as follows:

1. Generation of public/private keys, identity information, and cert request
2. Enrollment (request and retrieval)
3. Usage within IKE
4. Certificate validation and revocation checks
5. Renewal

Additional details regarding the aforementioned life cycle can be found in the following section.
**Generation of Keys and Cert Request**

As mentioned previously, the private key on a device must be generated on that device. And the private key can never be viewed or saved from that device. The user identity and private key form the basis of the certificate request and continue to be used with the local cert after enrollment. Therefore, it is important to match the certificate-id of the keys generated with the proper cert request and eventually the local cert. Also as mentioned earlier, the cert request is in the PKCS10 format and must be sent to the CA either via e-mail or some sort of web-based front-end site.

**Cert Enrollment**

Junos OS version 8.5 only supports the manual certificate request and retrieval process, which includes generation of a PKCS10 request, submittal to the CA, retrieval of the signed cert, and manually loading the cert into the Junos OS-based device as the local cert. SCEP, which can be administratively easier, is supported beginning with 9.0 (refer to Appendix D for SCEP details).

**Usage Within IKE (Identity Usage)**

During IKE phase 1 setup using certificates, a cert is used to identify the peer. The identity includes any of the following:

- IP address
- FQDN (domain name)
- U-FQDN (email address)
- DN (distinguished name)

Most VPN administrators are familiar with using IP address or FQDN for a VPN gateway’s identity type and an e-mail address (U-FQDN) for an NSR client VPN laptop/user’s identity type. These IKE IDs, if used, must be put into the SubjectAlternativeName (a v3 extension) field of the cert. IP addresses should generally be avoided because if the IP address of the VPN peer changes, you must issue a new certificate with the new IP address and revoke the old one. By using a non-IP IKE ID type, it is not necessary to change the certificate if the device’s IP address changes. If the CA doesn’t support the signing of certificates with a SubjectAlternativeName field, you must use the DN as the IKE ID on the Junos OS-based device or NSR configurations. Note that during the certificate request process, an IP address, domain name, or e-mail address must be specified as the IKE ID.

**Certificate Validation and Revocation Checking**

Revocation checking uses CRL lookup method in the Junos OS. The CRL can be either manually loaded in the Junos OS-based device or via automatic retrieval of CRL online from the CDP, on demand via LDAP or HTTP. Both DER and PEM formats for CRL are supported. OCSP is typically a much more scalable system than the use of CRLs and CDP. However, OCSP is not currently supported in the Junos OS, and many CAs also do not support an OCSP interface.

**Certificate Renewal**

The renewal of certificates is much the same as initial certificate enrollment except you are just replacing an old certificate (about to expire) on the VPN device with a new certificate. As with the initial certificate request, only manual renewal is supported. SCEP can be used to re-enroll local certificates automatically before they expire. Refer to Appendix D for more details.

**General Usage**

Currently IPsec/IKE is the only Junos OS feature that uses certificates for public key validation and identity binding. However, support for SSL might be possible in a later Junos OS release. Cert usage related to SSL is very straightforward. SSL support is discussed in this application note for informational purposes only.
SSL Usage

Certificates for SSL are much simpler to deal with than with IKE. IKE implementation is typically bidirectional authentication using certificates. Each VPN device sends its cert to the other. Each IPsec device must validate the other device’s cert using the CA cert (or chain of CA certs if a hierarchy of CAs is used) and monitor the cert via a CRL.

In SSL, the whole negotiation is a lot more one-sided and far simpler. For SSL, the security device is acting as an SSL server whereas the administrator’s Web browser (or for WebAuth, the user’s Web browser) is the SSL client. In these cases, the security device is only sending its cert to the Web browser for identification so the Web browser can begin a secured SSL session. The security device does not require a user-based SSL cert from the Web browser. It simply authenticates the user or administrator via a login and password secured with the SSL encryption. For SSL, a security device does not need CA CRLs loaded.

IPsec/IKE Usage

IKE is bidirectionally authenticated using certificates. Each VPN device sends its cert to the other. Each VPN device must validate the other device cert using the CA cert (or chain of CA certs if a hierarchy of CAs is used) and check the other cert via a CRL to detect if it has been revoked.

Process to Set up PKI Elements

The minimum PKI elements required to use certificate-based authentication in the Junos OS are a local cert, CA cert, and the CA’s CRL. These can be manually loaded. The manual process is the following:

1. Set basic device configuration items required by PKI:
   a) Set the clock, date, time zone, and daylight savings settings to be accurate.
   b) Set DNS to be able to resolve host names that might be used in certs and for CDPs.
   c) Set NTP to maintain accurate clock because certs have lifetimes based on specific date and time.
2. Create a ca-profile to be used for the certificate request and enrollment process.
3. Generate the cert request (p10 file) and save it to a local file system (or send via email).
4. Submit the p10 file to the CA. This is usually done via a Web server that front-ends the CA, although the OpenSSL CA is all command-line driven.
5. Retrieve the CA’s cert (via the CA’s Web server front-end interface).
6. Retrieve the Junos OS-based device’s new local cert after the CA has vetted it (usually via the CA’s Web server front-end interface).
7. Retrieve the CA’s CRL (usually via a pre-specified URL to the CA’s Web server).
8. Load the CA cert, local cert, and CRL onto the Junos OS-based device.
9. Define the IKE policy and gateway to use RSA-Signature authentication method (as opposed to preshared keys) and the local and CA certs.

The drawback to using manually loaded certs and CRLs is that it can be staff intensive, especially when it comes to maintaining an up-to-date CRL on all the boxes. To ease CRL administration, define a CDP that the Junos OS-based device can contact to automatically retrieve the latest CRL. Most CAs have the CDP defined in the CA cert, which the Junos OS-based device tries to use automatically. You can also define the CDP in the ca-profile configuration.

The step-by-step detailed procedures for all these processes can be found later in this document and in the Junos OS Configuration Guides: www.juniper.net/techpubs/software/Junos/
Choosing the IKE Identity to Use in the VPN and the Cert

With the Junos OS there are four possible IKE ID types that can be used for one VPN gateway to identify the other. (When doing preshared keys, you can only do 1, 2, or 3):

1. IP Address (for example, 2.2.2.2)
2. FQDN or fully qualified domain name (for example, vpn1.juniper.net)
3. U-FQDN or email address (for example, johndoe@juniper.net)
4. DN or distinguished name (for example, CN=John Doe, OU=sales, O=Juniper Networks, C=US)

If you use any of the first three (IP, domain name, or e-mail), this must be defined in the SubjectAlternativeName extension field of the cert. These must match the Junos OS peer configuration in the “security ike gateway <gateway-name>” hierarchy. It is mandatory that you use 1, 2, or 3 when generating a certificate request.

If your certificates don’t have SubjectAlternativeName fields, you must use DN for the IKE ID. You do this by defining distinguished-name in dynamic IKE configuration. For distinguished-name you can specify container (DN must match exactly) or wildcard (only a portion of the DN needs to match). Additional details regarding the difference between container and wildcard methods include:

- If the “container” keyword is used, all the DN identity fields must exactly match the values in the cert. In addition, the ordering of the values in both the remote cert received and the gateway definition must match.
- If the “wildcard” keyword is used, the DN values specified must match the DN in the cert received. However, any unspecified fields can contain anything in the cert. The ordering of the identity fields in the remote cert received and in the gateway definition can be different. The Junos OS-based device only parses out and compares the DN fields defined.

Also, if the certificate uses multiple fields of the same type (for example, OU=sales, OU=engr, OU=central), you must use the “container” method, and the DN definition must be defined exactly as it is received by the remote peer.

Cert Validation within the IKE Phase 1 Setup

During the IKE exchange, when the two VPN peers are establishing a tunnel, each VPN device receives a cert from the other IKE peer. The Junos OS-based device does the following:

1. It pulls the IKE identity offered in IKE phase 1 from the peer and does a search through the configuration to find the matching IKE gateway definition.
2. It validates the cert it has received, making sure there has been no tampering. It does this by validating the digital signature on the remote cert, using the public key in the CAs cert.
3. It validates the current time is after the cert’s “Valid from” but before its “Valid to” fields.
4. It validates that the cert contains the identity for that remote peer.
5. It does the revocation check (unless revocation check is disabled) on the cert by checking to make sure the cert’s serial number is not in the CA’s CRL. This can cause the Junos OS-based device to automatically retrieve a new CRL from the CDP.
6. IKE continues with the tunnel setup of the entire cert checks pass.

Unless explicitly disabled in configuration, CRLs are checked whenever the Junos OS-based device receives a certificate for a remote VPN gateway. If the CRL for the signing CA is not manually loaded, the Junos OS-based device attempts to do the following:

1. It loads the CRL from the CDP defined in the cert. Junos OS-based devices support HTTP and LDAP for CDPs.
2. It checks for a CDP setting in the CA profile if the CDP is not available in the cert.
3. It uses the globally (or default) CDP setting if one has been configured.

When verifying a certificate, a certificate chain is built. It is built from the remote’s local cert, the optional certificate chain sent from the IKE peer, and CA certificates stored locally. Any CA certificates loaded from local store during boot are considered “trusted.” The Junos OS supports certificate path validation upward through seven levels of CA authorities in the hierarchy.

The Junos OS currently supports only “partial” cert path validation. Unlike “full” cert path validation, “partial” does not require that the last certificate in the certificate chain be a root CA cert (a self-signed CA). With “partial,” the last cert might be a non-root CA cert. However, the last certificate in the validation chain must come from local storage.
What PKI Protocols Are Not Supported in the Junos OS

There are a couple of protocols used in PKI that the Junos OS does not support. These protocols are primarily alternative forms of management protocols for communication between CAs and entities that use certs. These include:

1. CMC—Certificate Management Messages over CMS defined in RFC 2797
2. CMP—Certificate Management Protocol defined in RFC 2510
3. XKMS—XML Key Management Specification defined at www.w3.org/TR/xkms/

Microsoft and VeriSign endorse CMC but there are limited products shipping that support it. CMP is endorsed by Entrust, Baltimore Technologies, SSH, RSA, and OpenSSL. It generally works, has limited deployment, and some interoperability has been demonstrated between vendors.

XKMS is relatively early on in its development and seems promising. However, Juniper does not support it at this time. The battle for certificate management seems to be between CMC and CMP. The existence of the two standards to accomplish the same goal has created a “roadblock” in the PKI industry. A VPN vendor would have to implement both to work with all major PKI vendors, which is not feasible given the percentage of VPNs currently using PKI. And the VPNs currently using PKI are compatible with SCEP and CDP/OCSP.

Administering PKI in Junos OS

This section of the document discusses more details related to the administration and use of PKI in the Junos OS. Detailed procedures for administration of PKI can be found in the Junos OS Configuration Guides for VPNs.

www.juniper.net/techpubs/software/Junos/

The reader should consult those documents when trying to set up PKI and use it with IPsec/IKE VPNs. This section tries to augment the Configuration Guide with other details using examples. More details regarding the use of a CA from PKI vendors such as Microsoft and OpenSSL can be found in Appendix B at the end of this document. The main subsections in this section of the document are the following:

- IPsec VPN—specifics of PKI related to IKE and configuration
- Verifying PKI configurations and operation
- Troubleshooting PKI and IKE in the Junos OS

Network Diagram

Refer to Figure 1 for network topology used for this configuration example. For the purposes of this example, a policy-based VPN is configured. However, there is no difference in PKI administration with either policy-based or route-based VPNs.

---

**Figure 1: Network diagram**
APPLICATION NOTE - Junos OS IPsec VPN with PKI Certificates Primer

Configuration Overview

For the purposes of this application note, the focus is on configuration and troubleshooting of Junos OS version 8.5. The remote VPN peer for this example is a Juniper Networks SSG5 Secure Services Gateway, which is very commonly used for branch offices. More details regarding PKI configuration with SSG5 and other platforms based on Juniper Networks ScreenOS® Software can be found in the following application note:

Juniper Networks PKI (Public Key Infrastructure) Primer & FAQ, Using X.509 Certificates in ScreenOS.

This example assumes the following:

- Internal LAN interface of the Junos OS-based device is ge-0/0/0 in zone “trust” and has a private IP subnet.
- Internet interface of the Junos OS-based device is ge-0/0/3 in zone “untrust” and has a public IP.
- All traffic between the local and remote LANs is to be permitted, and traffic might be initiated from either side.
- The SSG5 has already been preconfigured with the correct information from this example—and local-cert, CA-cert, and CRL are already loaded and ready to use.
- The SSG5 is configured to use FQDN as ssg5.juniper.net as the IKE ID.
- PKI certificates with 1024 bit keys are used for the IKE negotiations on both sides.
- The CA is a standalone CA at domain labdomain.com for both VPN peers.

Setting the Device's Basic Configuration for PKI

Configuring Interface IP Addresses

Referring to the previous network diagram (Figure 1), configure the IP addresses for ge-0/0/0.0 and ge-0/0/3.0. In this example, unit 0 and family inet (IPv4) are used.

Syntax (configure mode):

```
set interfaces <interface-name> unit <unit-value> family inet address <ip-prefix>
```

Example:

```
root@CORPORATE> configure
Entering configuration mode
[edit]
root@CORPORATE# set interfaces ge-0/0/0 unit 0 family inet address 10.10.10.1/24
[edit]
root@CORPORATE# set interfaces ge-0/0/3 unit 0 family inet address 1.1.1.2/30
```

Configuring Default Route

Configure the default route to the Internet next hop. Optionally, you can use a dynamic routing protocol such as OSPF instead but that is beyond the scope of this application note. When processing the first packet of a new session, the Junos OS-based device first performs a route lookup. The static route shown in the following example happens to be the default route that dictates which zone the VPN traffic must egress. In this example, the VPN traffic will ingress on interface ge-0/0/0.0 with the next hop of 1.1.1.1. The traffic will egress out interface ge-0/0/3.0 interface. Any tunnel policy must take into account ingress and egress interfaces.

Syntax (configure mode):

```
set routing-options static route <ip-prefix> next-hop <next-hop-ip>
```

Example:

```
[edit]
root@CORPORATE# set routing-options static route 0.0.0.0/0 next-hop 1.1.1.1
```
Setting System Time

An accurate clock is very important for all devices using certs. The best method is to use NTP for keeping clocks accurate. The following example enables NTP, sets the NTP server at 130.126.24.24, and sets the time zone to U.S. Pacific Daylight time-zone (for example, Los Angeles).

Syntax (configure mode):
set system time-zone (GMThour-offset | time-zone)
Syntax (operational mode):
set date ntp (ntp-server-ip | ntp-server-domainname) source-address <ip-prefix>
Example:
[edit]
root@CORPORATE# set system time-zone PST8PDT
[edit]
root@CORPORATE# commit and-quit
commit complete
Exiting configuration mode
root@CORPORATE> set date ntp 130.126.24.24
1 Nov 17:52:52 ntpdate[5204]: step time server 130.126.24.24 offset -0.220645 sec
Once the configuration is committed, verify clock settings with the following command (system current time highlighted in red):
root@CORPORATE> show system uptime
Current time: 2007-11-01 17:57:09 PDT
Last configured: 2007-11-01 17:52:32 PDT (00:04:37 ago) by root
5:57PM up 3:21, 4 users, load averages: 0.00, 0.00, 0.00

Setting DNS Configuration

Many CAs use hostnames (that is, FQDN) to specify various elements of the PKI. For example, usually the CDP is specified using a URL containing a FQDN. For this reason, DNS resolver on the Junos OS-based device should be configured. The following is an example of DNS resolver configuration.

Syntax (configure mode):
set system name-server <dns-server-ip>
Example:
root@CORPORATE> configure
Entering configuration mode
[edit]
root@CORPORATE# set system name-server 4.2.2.1
[edit]
root@CORPORATE# set system name-server 4.2.2.2
[edit]
root@CORPORATE# commit and-quit
commit complete
Exiting configuration mode
Generating the Certificate Request

There are two basic steps for the certificate request process. The first step is to create a CA profile to specify the CA settings. The second step is to generate the PKCS10 cert request. The PKCS10 cert request process involves first generating a public/private key pair and then generating the cert request using the key pair.

Creating a Trusted CA Profile

The CA profile configuration specifies the information specific to a certificate authority. There can be multiple such profiles present in the system. For example, you might have one such profile for VeriSign and one for Entrust. Each profile is associated with a CA certificate. If a new or renewed CA certificate needs to be loaded without removing the older CA certificate, a new profile is required. This profile can also be used for online fetching of the CRL.

Syntax (configure mode):

```
set security pki ca-profile <ca-profile-name> ca-identity <ca-identity>
set security pki ca-profile <ca-profile-name> revocation-check crl
  refresh <number-hours> url <url-string>;
```

Example:

```
root@CORPORATE> configure
Entering configuration mode
[edit]
root@CORPORATE# set security pki ca-profile ms-ca ca-identity labdomain.com
[edit]
root@CORPORATE# set security pki ca-profile ms-ca revocation-check crl refresh-interval 48
[edit]
root@CORPORATE# set security pki ca-profile ms-ca revocation-check crl url http://labsrv1.labdomain.com/CertEnroll/LABDOMAIN.crl
[edit]
root@CORPORATE# set security pki ca-profile ms-ca administrator email-address certadmin@labdomain.com
[edit]
root@CORPORATE# commit and-quit
Commit complete
Exiting configuration mode
```

The mandatory configuration includes the CA profile name (ms-ca for this example) and CA identity (labdomain.com). The CA profile name can be any value, while the CA identity is typically the CA domain name. All other CA profile settings are not mandatory.

Revocation-check specifies how certificate revocation is checked. If the disable flag is configured, the revocation check would be disabled. Within the revocation-check hierarchy, the CRL section defines how CRL is handled:

- Refresh interval—allows the user to specify frequency to update CRL
  Default: next-update time in CRL or 1 week if no next-update
- URL—location to retrieve CRL, could be HTTP or LDAP
  Default: empty (use CDP information embedded in CA cert)

The refresh interval is configured with units of one hour. Refresh-interval 48 means the CRL is refreshed every 48 hours or 2 days. The URL configuration overrides the CDP information embedded within the CA cert. The URL could also contain just the server-name/port information, (for example, “ldap://<ip-or-fqdn>:<port>”). If the port number is missing, HTTP uses port 80, or LDAP uses port 443. Currently, one URL is supported. There is no provision to configure a backup URL.
The certificate request can be sent to the CA through an out-of-band method. Or it can be sent to a CA administrator directly via an e-mail address. This is the administrator e-mail address configuration in the ca-profile-name hierarchy. In this latter case, the Junos OS-based device does the following:

- Composes the e-mail from the certificate request file
- Forwards the e-mail to the address configured
- Reports e-mail status to the device admin

A PKCS10 cert request would be generated and stored on the Junos OS-based device as a pending cert or certificate request. An e-mail is generated and sent to the administrator of the CA (in the previous example that would be certadmin@labdomain.com).

A special default (fallback) profile can also be created for intermediate CAs not pre-installed in device. The values in this default profile are used when there is no specific CA profile. In case of CDP, the first CDP found is used in the following order:

- per CA profile
- CDP embedded in CA certificate
- default CA profile

However, specific CA profiles are recommended. A default profile is not required.

**Generating the Cert Request**

**Generating Public/Private Key Pair**

Once the CA profile is configured, the next step is to generate a key-pair on the Junos OS-based device. The following command is used to generate the private and public key-pair.

**Syntax (operational mode):**

```
request security pki generate-key-pair certificate-id <id-name> size <key-size>
```

Currently, the Junos OS supports only the RSA algorithm. DSA is not supported. A unique ID called certificate-id is used to name the generated key-pair. This ID is used in certificate enrollment and request commands to get the right key-pair. The generated key-pair is saved in the certificate-store in a file with the same name as the certificate-id. The size can be 512, 1024, or 2048 bits.

**Example:**

```
root@CORPORATE> request security pki generate-key-pair certificate-id ms-cert size 1024
Generated key pair ms-cert, key size 1024 bits
```

**Generating Cert Request from the Key Pair**

The next step is to generate the PKCS10 cert request to be sent to the CA. The following command is used to generate the certificate request and save it in a file location as specified. Also, it keeps a local copy of the certificate request in the local certificate store. If the administrator reissues this command, the certificate request is generated all over again.

**Syntax (operational mode):**

```
request security pki generate-certificate-request certificate-id <id-name> subject "<subject-name>" (domain-name <domain-name> | ip-address <device-ip> | email <email-id>) filename <filename>
```

Example:
The certificate-id should match the same id-name used during the generation of the key pair. This ensures that the proper key pair is used for the cert request and ultimately the local cert. The subject name is specified in the distinguished name format. An administrator should enter the components—which include common name, department, company name, locality (usually city), state, country name, phone, and domain component—in the format:

- common name, CN=
- department, OU=
- company name, O=
- locality, L=
- state, ST=
- country, C=
- phone, CN=
- domain component, DC=

Not all subject-name components are required, and multiples of each type are allowed. It is mandatory that you use domain-name, ip-address, or e-mail. This defines the IKE ID type and must be configured to match in the IKE gateway profile described later in this document.

The filename defines the name of the file that contains the PKCS10 cert request, which can be off-loaded from the Junos OS-based device and sent to the CA for enrollment. However, the PKCS10 cert request also prints to the CLI window and can be copied and pasted to a Web front end for the CA server or to an e-mail. The filename is not mandatory as the certificate request could still be displayed with the following command: “show security pki certificate-request certificate-id <id-name>”

The following example is a certificate request and the command output.

Example:

```bash
root@CORPORATE> request security pki generate-certificate-request certificate-id ms-cert subject "CN=John Doe,CN=1.1.1.2,OU=sales,O=Juniper Networks, L=Sunnyvale,ST=CA, C=US" email user@juniper.net filename ms-cert-req
```

Generated certificate request

```
-----BEGIN CERTIFICATE REQUEST-----
MIIB3DCCAUUCAQAwbDERMA8GA1UEAxMIam9obiBkb2UxDjAMBgNVBAsTBXNhbGVz
MRkwFwYDVQQKEw9TaWV0aG9kMTIwEAYDVQQKExJUaXNhbiBEb24xEDAOBgNV
BAoTCUVSBGdvdGgEswgYIKwYForZmFsc2VudoHVybGlzaW9uc3RtcjB0
MDgAMC4wHjCCAtAwHQYDVR0OBBYwFIEQdMYwEwDAYBQAwGgYDVR0PAwID
BQAoAwIBAQCgYEA9CWcIw1mX2ib67E1gC5hKoEJajUg9Kh4ZQ9V
-------END CERTIFICATE REQUEST-----
```

The PKCS10 certificate request (highlighted in red) starts with and includes the “BEGIN CERTIFICATE REQUEST” line and ends with and includes the “END CERTIFICATE REQUEST” line. This portion can be copied and pasted to your CA for enrollment. Optionally, you can also off-load the “ms-cert-req” file and send that to your CA.
Submit Cert Request to the CA and Retrieve Certs

At this point, Junos OS-based device administrators need to submit the cert request to their CA administrators. Once the CA administrators have vetted the cert request, thereby generating a new cert for the Junos OS-based device, the Junos OS-based device administrators must retrieve it along with the CA cert and CRL.

The process of retrieving the CA cert, the Junos OS-based device's new local cert, and CRL from the CA is dependent on the CA configuration and software vendor in use. Later in this application note are two sections showing how to do this with a Microsoft CA available on the Windows 2000 Advanced Server and with OpenSSL.

The Junos OS supports these CA vendors:

- Entrust
- VeriSign
- Microsoft

Other CA software and/or services (such as OpenSSL) should work, but these have not been verified by Juniper. However, the Junos OS is likely to support other vendors as long as they conform to X.509 certificate standards.

Generating the Cert Request

Loading the Local Cert, CA Cert, and CRL

Once the local cert, CA cert, and CRL are retrieved from the CA, they can be loaded into the Junos OS-based device via the CLI. You can upload the individual files onto the Junos OS local storage via FTP.

Uploading the Files to Local Storage

If the new local cert is named certnew.cer, the CA cert is named CA-certnew.cer, and the CRL is named certcrl.crl, make sure to place the individual files on a reachable FTP server. Upload the individual files onto the Junos OS-based device local storage by using the following command.

```plaintext
Syntax (operational mode):
file copy <source-file-path/filename> <local-file-name>
Example:

root@CORPORATE> file copy ftp://10.10.10.10/certnew.cer certnew.cer
/var/tmp//...transferring.file.........crYdEC/100% of 1459 B 5864 kBps

root@CORPORATE> file copy ftp://10.10.10.10/CA-certnew.cer CA-certnew.cer
/var/tmp//...transferring.file.........UKXUWu/100% of 1049 B 3607 kBps

root@CORPORATE> file copy ftp://10.10.10.10/certcrl.crl certcrl.crl /var/
tmp//...transferring.file.........wpqnpA/100% of 401 B 1611 kBps
```

You can verify that all files have been uploaded with the following command: “file list”
Loading the Local Cert

This command is used to load the certificate into local store from the specified external file. It needs to specify the certificate-id to keep the proper linkage with the private/public key-pair. This also loads the certificate into RAM cache storage of the PKI module. The associated private key is also checked and the signing operation verified.

Syntax (operational mode):

```
request security pki local-certificate load certificate-id <certificate-id> filename <path/filename>
```

Example:

```
root@CORPORATE> request security pki local-certificate load certificate-id ms-cert filename certnew.cer
Local certificate loaded successfully
```

Loading the CA Cert

Load the CA certificate from the specified external file with the following command. The CA profile must be specified to link the CA cert to the profile configured.

Syntax (operational mode):

```
request security pki ca-certificate load ca-profile <ca-profile-name> filename <path/filename>
```

Example:

```
root@CORPORATE> request security pki ca-certificate load ca-profile ms-ca filename CA-certnew.cer
Fingerprint:
Do you want to load this CA certificate? [yes,no] (no) yes

CA certificate for profile ms-ca loaded successfully
```

Loading the CRL

This command loads the CRL into the local store. Maximum size is 5 MB. As with the CA cert, the CA profile must be specified.

Syntax (operational mode):

```
request security pki crl load ca-profile <ca-profile-name> filename <path/filename>
```

Example:

```
root@CORPORATE> request security pki crl load ca-profile ms-ca filename certcrl.crl
CRL for CA profile ms-ca loaded successfully
```

Verify All Certs Loaded

The commands to show the certs are listed in the following section. These commands show the certificates present in the local store. There is also an option to show the certificate request generated in the PKCS10 format.
**Viewing Local Certs**

The following command can show all local certs, or an individual one can be specified based on the certificate-id. Furthermore, the output can be either in brief (short) or detail (longer) format.

Syntax (operational mode):

```
show security pki local-certificate certificate-id <certificate-id> [brief | detail]
```

Example:

```
root@CORPORATE> show security pki local-certificate certificate-id ms-cert detail
Certificate identifier: ms-cert
Certificate version: 3
Serial number: 3a01c5a0000000000001
Issuer:
  Organization: JuniperNetworks, Organizational unit: JTAC, Country: US, State: CA, Locality: Sunnyvale,
  Common name: TACLAB
Subject:
  Organization: Juniper Networks, Organizational unit: sales, Country: US, State: CA, Locality: Sunnyvale,
  Common name: john doe
Alternate subject: “user@juniper.net”, fqdn empty, ip empty
Validity:
  Not before: 11-2-2007 22:54
  Not after: 11-2-2008 23:04
Public key algorithm: rsaEncryption(1024 bits)
Signature algorithm: sha1WithRSAEncryption
Distribution CRL:
  ldap:///CN=TACLAB,CN=TACLABSRV1,CN=CDP,CN=Public%20Key%20Services,CN=Services,
  CN=Configuration,DC=tacdomain,DC=com?certificateRevocationList?base?
objectclass=cRDLDistributionPoint
  http://taclabsrv1.tacdomain.com/CertEnroll/TACLAB.crl
Fingerprint:
Viewing CA Certs

The following command can show all CA certs, or an individual one can be specified based on the ca-profile. Furthermore, the output can be either in brief (short) or detail (longer) format.

Syntax (operational mode):
show security pki ca-certificate ca-profile <ca-profile> [brief | detail]

Example:
root@CORPORATE> show security pki ca-certificate ca-profile ms-ca detail
Certificate identifier: ms-ca
Certificate version: 3
Serial number: 44b033d1e5e158b44597d143bbfa8a13
Issuer:
Organization: JuniperNetworks, Organizational unit: JTAC, Country: US, State: CA, Locality: Sunnyvale,
Common name: TACLAB
Subject:
Organization: JuniperNetworks, Organizational unit: JTAC, Country: US, State: CA, Locality: Sunnyvale,
Common name: TACLAB
Validity:
Not before: 09-25-2007 20:32
Not after: 09-25-2012 20:41
Public key algorithm: rsaEncryption(1024 bits)
ad:de:db:55:ff:bb:6a:0e:36:81:e3:e9:3b:e5:c9:02:03:01:00:01
Signature algorithm: sha1WithRSAEncryption
Distribution CRL:
ldap:///CN=TACLAB,CN=TACLABSRV1,CN=CDP,CN=Public%20Key%20Services/CN=Services,CN=Configuration,DC=tacdomain,DC=com?certificateRevocationList?base?
objectclass=cRLDistributionPoint
http://taclabsrv1.tacdomain.com/CertEnroll/TACLAB.crl
Use for key: CRL signing, Certificate signing, Non repudiation
Fingerprint:
Viewing the CRL

The following command can show all CRLs loaded, or an individual one can be specified based on the ca-profile. For CRL viewing, either brief (short) or detail (longer) format can be specified but currently show the same output.

Syntax (operational mode):

```
show security pki crl ca-profile <ca-profile> [brief | detail]
```

Example:

```
root@CORPORATE> show security pki crl ca-profile ms-ca detail
CA profile: ms-ca
CRL version: V00000001
CRL issuer: emailAddress = certadmin@juniper.net, C = US, ST = CA, L = Sunnyvale, O = JuniperNetworks, OU = JTAC, CN = TACLAB
Effective date: 10-30-2007 20:32
Next update: 11- 7-2007 08:52
```

Verifying Cert Path

Finally, the certificate path for the local cert and any CA cert can be verified with the following command.

Syntax (operational mode):

```
request security pki local-certificate verify certificate-id <certificate-id>
request security pki ca-certificate verify ca-profile <ca-profile>
```

Example:

```
root@CORPORATE> request security pki local-certificate verify certificate-id ms-cert
Local certificate ms-cert verification success

root@CORPORATE> request security pki ca-certificate verify ca-profile ms-ca
CA certificate ms-ca verified successfully
```

Using the Cert in an IPsec VPN

The steps to configure a VPN using a certificate are much the same as with a VPN using preshared keys. The difference is in the authentication method used for the IKE (phase 1) policy. There is no change required for IPsec (phase 2) configuration because the use of certs is part of phase 1 negotiations. For this example, a policy-based VPN is configured because this is most commonly used for dial-up VPNs.

More details regarding VPN configuration can be found in the Junos OS Configuration Guides, which are downloadable from [www.juniper.net/techpubs/](http://www.juniper.net/techpubs/).

Furthermore, additional Junos OS Application Notes can be viewed from Juniper Networks Knowledge Base article KB10182 ([http://kb.juniper.net/KB10182](http://kb.juniper.net/KB10182)).
**Steps to Configure**

Using the network diagram (Figure 1), the following steps explain how to configure the IPsec VPN with the certificate.

1. Configure security zones and bind the interfaces to the appropriate zones. Also, ensure that necessary host-inbound services are enabled on the interfaces or the zone. For this example, you must enable IKE service on either ge-0/0/3 interface or the “untrust” zone.

2. Configure address book entries for each zone. This is used in the tunnel policies.

3. Configure IKE (phase 1) proposals to use RSA encryption.

4. Configure IKE policy by specifying the RSA proposal from step 3, local cert, CA cert, and x.509 type peer certificate.

5. Configure IKE gateway settings by specifying the IKE policy from step 4 and dynamic peer identified by hostname. This step depends on how the certificate request was first generated. For example, for this application note, “CN=ssg5.juniper.net” was specified during the certificate request by the SSG5, which means the IKE ID type is hostname.

6. Configure IPsec (phase 2) VPN settings. Optionally, you can also configure VPN monitor settings if desired. Note that this example uses “Standard” proposal set and PFS group 2. However, you can create a different proposal if necessary.

7. Configure tunnel policies to permit remote office traffic into the corporate LAN and vice versa. Also, configure outgoing “trust” to “untrust” permit-all policy with source NAT for Internet traffic. Ensure that the tunnel policy is placed before the permit-all policy. Otherwise the policy lookup never reaches the tunnel policy.

8. Configure tcp-mss for IPsec traffic to eliminate the possibility of fragmented TCP traffic. This lessens the resource utilization on the device.

**IPsec VPN Configuration Example for Junos OS**

**Configuring Security Zones and Assigning Interfaces to the Zones**

The ingress and egress zones are determined by the ingress and egress interfaces involved in the route lookup. As shown in the previous example, packets ingressing on ge-0/0/0 also means that the ingress zone is a “trust” zone. Following the route lookup, the egress interface is ge-0/0/3, and the egress zone is an “untrust” zone. The tunnel policy must be “from-zone trust to-zone untrust” and vice versa.

```
Syntax (configure mode):
set security zones security-zone <zone-name> interfaces <interface-name>
Example:
root@CORPORATE> configure
Entering configuration mode
[edit]
root@CORPORATE# set security zones security-zone trust interfaces ge-0/0/0.0
[edit]
root@CORPORATE# set security zones security-zone untrust interfaces ge-0/0/3.0
```
Configuring Host-Inbound Services for Each Zone

Host-inbound services are for traffic destined for the Junos OS-based device. This includes but is not limited to FTP, HTTP, HTTPS, IKE, PING, rlogin, rsh, SNMP, SSH, telnet, TFTP, and traceroute. For this example, assume that all such services are to be allowed from zone “trust.” For security reasons only IKE is allowed on the Internet-facing zone “untrust,” which is required for IKE negotiations to occur. However, other services such as for management and/or troubleshooting can also be individually enabled if required.

Syntax (configure mode):

set security zones security-zone <zone-name> host-inbound-traffic system-services
(ike | all | ping | ...)

Example:

[edit]
root@CORPORATE# set security zones security-zone trust host-inbound-traffic
system-services all
[edit]
root@CORPORATE# set security zones security-zone untrust host-inbound-traffic
system-services ike

Configuring Address Book Entries for Each Zone

This example uses address-book object names “local-net” and “remote-net.” There are some limitations with regards to which characters are supported for address-book names. Please refer to complete Junos OS documentation for more details.

Syntax (configure mode):

set security zones security-zone <zone-name> address-book address <address-name>
<ip-prefix>

Example:

[edit]
root@CORPORATE# set security zones security-zone trust address-book address
local-net 10.10.10.0/24
[edit]
root@CORPORATE# set security zones security-zone untrust address-book address
remote-net 192.168.168.0/24

Configuring IKE Phase 1 Proposal

Configure IKE (phase 1) proposal to use RSA encryption. This example uses 3DES encryption and SHA1 authentication algorithm and Diffie-Hellman Group2 keys.

Syntax (configure mode):

set security ike proposal <proposal-name> authentication-method (pre-shared-keys
| rsa-signatures) encryption-algorithm (3des-cbc | aes-128-cbc | aes-192-cbc |
aes-256-cbc | des-cbc) authentication-algorithm (md5 | sha-256 | sha1) dh-group
(group1 | group2 | group5) lifetime-seconds <lifetime>

Example:

[edit]
root@CORPORATE# set security ike proposal rsa-propl authentication-method rsa-
signatures
[edit]
root@CORPORATE# set security ike proposal rsa-propl encryption-algorithm 3des-cbc
[edit]
root@CORPORATE# set security ike proposal rsa-propl authentication-algorithm shal
[edit]
root@CORPORATE# set security ike proposal rsa-propl dh-group group2
Configuring IKE Policy
Main mode is typically used for site-to-site VPNs with static IP peers. Dynamic IP and dial-up peers typically use aggressive mode. However, for the purposes of this application note, main mode is used because both sides have static IPs even though hostname (typically used for dynamic tunnels) is used for the IKE ID.

Syntax (configure mode):
```
set security ike policy <policy-name> mode (aggressive | main) proposals <proposal-name> certificate local-certificate <certificate-id> peer-certificate-type (pkcs7 | x509-signature) trusted-ca (<ca-index> | use-all)
```
Example:
```
[edit]
root@CORPORATE# set security ike policy ike-policy1 mode main
[edit]
root@CORPORATE# set security ike policy ike-policy1 proposals rsa-propl
[edit]
root@CORPORATE# set security ike policy ike-policy1 certificate local-certificate ms-cert
[edit]
root@CORPORATE# set security ike policy ike-policy1 certificate peer-certificate-type x509-signature
[edit]
root@CORPORATE# set security ike policy ike-policy1 certificate trusted-ca use-all
```

Configuring IKE Gateway
A remote IKE peer can be identified by IP address, FQDN/u-FQDN, or ASN1-DN (PKI certificates). This example identifies the peer by FQDN (hostname). Therefore, the gateway IKE ID should be the remote peer’s domain name. It is also important to specify the correct external interface. If either the peer ID or external interface specified is incorrect, the IKE gateway is not properly identified during phase 1 setup.

Syntax (configure mode):
```
set security ike gateway <gateway-name> external-interface <interface-name> ike-policy <policy-name> dynamic (hostname | inet | user-at-hostname) <ike-user-id>
```
Example:
```
[edit]
root@CORPORATE# set security ike gateway ike-gate external-interface ge-0/0/3.0 ike-policy ike-policy1 dynamic hostname ssg5.juniper.net
```

Configuring IPsec Policy
For the purposes of this application note, "Standard" proposal set is being used, which includes esp-group2-3des-sha1 and esp-group2-aes128-sha1 proposals. However, a unique proposal can be created and specified in the IPsec policy if required.

Syntax (configure mode):
```
set security ipsec policy <policy-name> proposal-set (basic | compatible | standard)perfect-forward-secrecy keys (group1 | group2 | group5)
```
Example:
```
[edit]
root@CORPORATE# set security ipsec policy vpn-policy1 proposal-set standard perfect-forward-secrecy keys group2
```

Configuring IPsec VPN with IKE Gateway and IPsec Policy

For this example, the VPN name “ike-vpn” must be referenced in the tunnel policy to be able to create a security association. Additionally, if required, idle time can be specified as a proxy ID if it is different from the tunnel policy addresses.

Syntax (configure mode):

```
set security ipsec vpn <vpn-name> ike gateway <gateway-name> ipsec-policy <policy-name> [idle-time <value> proxy-identity local <local-prefix> remote <remote-prefix> service <application-name>]
```

Example:

```
[edit]
root@CORPORATE# set security ipsec vpn ike-vpn ike gateway ike-gate
[edit]
root@CORPORATE# set security ipsec vpn ike-vpn ike ipsec-policy vpn-policy1
```

Configuring Bidirectional Tunnel Policies for VPN Traffic

For this example, traffic from the corporate LAN to the remote office LAN requires a “from-zone trust to-zone untrust” tunnel policy. However, if a session needs to originate from the remote LAN to the corporate LAN, a tunnel policy in the opposite direction—“from-zone untrust to-zone trust”—is required. By specifying the policy in the opposite direction as the pair-policy, the VPN becomes bidirectional. Note also that in addition to action permit, the IPsec profile to be used also needs to be specified. Furthermore, source NAT can be enabled on the policy if desired but that is beyond the scope of this application note. Note that for tunnel policies the action is always permit. In fact, if configuring a policy with action of deny, you do not see an option for specifying the tunnel.

Syntax (configure mode):

```
edit security policies from-zone <source-zone> to-zone <dest-zone>
set policy <policy-name> match source-address <source-address> destination-address <dest-address> application <application-name>
set policy <policy-name> then permit tunnel ipsec-vpn <vpn-name> pair-policy <pair-policy-name>
```

Example:

```
[edit]
root@CORPORATE# edit security policies from-zone trust to-zone untrust
[edit security policies from-zone trust to-zone untrust]
root@CORPORATE# set policy tunnel-policy-out match source-address local-net
[edit security policies from-zone trust to-zone untrust]
root@CORPORATE# set policy tunnel-policy-out match destination-address remote-net
[edit security policies from-zone trust to-zone untrust]
root@CORPORATE# set policy tunnel-policy-out match application any
[edit security policies from-zone trust to-zone untrust]
root@CORPORATE# set policy tunnel-policy-out then permit tunnel ipsec-vpn ike-vpn pair-policy tunnel-policy-in
[edit security policies from-zone trust to-zone untrust]
root@CORPORATE# top edit security policies from-zone untrust to-zone trust
[edit security policies from-zone untrust to-zone trust]
root@CORPORATE# set policy tunnel-policy-in match source-address remote-net
[edit security policies from-zone untrust to-zone trust]
root@CORPORATE# set policy tunnel-policy-in match destination-address local-net
[edit security policies from-zone untrust to-zone trust]
root@CORPORATE# set policy tunnel-policy-in match application any
[edit security policies from-zone untrust to-zone trust]
root@CORPORATE# set policy tunnel-policy-in then permit tunnel ipsec-vpn ike-vpn pair-policy tunnel-policy-out
[edit security policies from-zone untrust to-zone trust]
root@CORPORATE# exit
```
Configuring Security Policy for Internet Traffic

This policy permits all traffic from zone “trust” to zone “untrust.” By specifying “source-nat interface” the device translates the source IP and port for outgoing traffic by using the IP address of the egress interface as the source IP and random higher port for the source port.

Syntax (configure mode):

```
edit security policies from-zone <source-zone> to-zone <dest-zone>
set policy <policy-name> match source-address <source-address> destination-
    address <dest-address> application <application-name>
set policy <policy-name> then permit source-nat (interface | pool <pool-name> |
    pool-set <pool-set-name>)
```

Example:

```
[edit]
root@CORPORATE# edit security policies from-zone trust to-zone untrust
[edit security policies from-zone trust to-zone untrust]
root@CORPORATE# set policy any-permit match source-address any
[edit security policies from-zone trust to-zone untrust]
root@CORPORATE# set policy any-permit match destination-address any
[edit security policies from-zone trust to-zone untrust]
root@CORPORATE# set policy any-permit match application any
[edit security policies from-zone trust to-zone untrust]
root@CORPORATE# set policy any-permit then permit source-nat interface
[edit security policies from-zone trust to-zone untrust]
root@CORPORATE# exit
```

Note that this policy must be below the tunnel policy because the policy list is read from top to bottom. If this policy were placed before the tunnel policy, the traffic would always match this policy and would not continue to the next policy. No user traffic is encrypted. To move the tunnel policy before the any-permit policy, use the following insert command.

```
[edit]
root@CORPORATE# edit security policies from-zone trust to-zone untrust
[edit security policies from-zone trust to-zone untrust]
root@CORPORATE# insert policy tunnel-policy-out before policy any-permit
[edit security policies from-zone trust to-zone untrust]
root@CORPORATE# exit
```
Configuring tcp-mss for TCP Traffic Across the Tunnel

Tcp-mss is negotiated as part of the TCP three-way handshake. It limits the maximum size of a TCP segment to better fit the MTU limits on a network. This is especially important for VPN traffic as the IPsec encapsulation overhead, along with the IP and frame overhead, can cause the resulting ESP packet to exceed the MTU of the physical interface—causing fragmentation. Fragmentation increases bandwidth and device resources and is always best avoided. Note the value of 1350 is a recommended starting point for most Ethernet-based networks with MTU of 1500 or greater. This value must be altered if any device in the path has lower MTU and/or if there is any added overhead such as PPP, Frame Relay, etc. As a general rule, experimenting with different tcp-mss values to obtain optimal performance might be required.

Syntax (configure mode):

```
set security flow tcp-mss ipsec-vpn mss <mss-value>
```

Example:

```
[edit]
root@CORPORATE# set security flow tcp-mss ipsec-vpn mss 1350
[edit]
root@CORPORATE# commit and-quit
commit complete
Exiting configuration mode
```

SSG Configuration Example

The focus of this application note is on Junos OS configuration and troubleshooting. For the purpose of completing the previous diagram, a sample of relevant configurations is provided from an SSG5 device strictly for reference. However, the concepts with regard to configuration of policy-based VPNs for Juniper firewall/VPN products are well documented in the Concepts & Examples guides. The SSG5 configuration is not discussed in this application note. For reference, the SSG5 Concepts & Examples guides can be found at [www.juniper.net/techpubs/software/screenos/](http://www.juniper.net/techpubs/software/screenos/).

Configuration Example for SSG5

```
set interface ethernet0/6 ip 192.168.168.1/24
set interface ethernet0/6 route
set interface ethernet0/0 ip 2.2.2.2/30
set interface ethernet0/0 route
set flow tcp-mss 1350
set domain juniper.net
set hostname ssg5
set pki x509 default cert-path partial
set pki x509 dn country-name “US”
set pki x509 dn state-name “CA”
set pki x509 dn local-name “Sunnyvale”
set pki x509 dn org-name “Juniper Networks”
set pki x509 dn org-unit-name “Sales”
set pki x509 dn ip 2.2.2.2
set dns host dns1 4.2.2.1
set dns host dns2 4.2.2.2
set address “Trust” “local-net” 192.168.168.0 255.255.255.0
set address “Untrust” “corp-net” 10.10.10.0 255.255.255.0
set ike gateway “corp-ike” address 1.1.1.2 Main outgoing-interface ethernet0/0
proposal “rsa-g2-3des-sha”
set vpn “corp-vpn” gateway “corp-ike” replay tunnel idletime 0 sec-level standard
set policy id 11 from “Trust” to “Untrust” “local-net” “corp-net” “ANY” tunnel
vpn “corp-vpn” pair-policy 10
set policy id 10 from “Untrust” to “Trust” “corp-net” “local-net” “ANY” tunnel
vpn “corp-vpn” pair-policy 11
set policy id 1 from “Trust” to “Untrust” “ANY” “ANY” “ANY” nat src permit
set ntp server “130.126.24.24”
set route 0.0.0.0/0 interface ethernet0/0 gateway 2.2.2.1
```
Verifying Configuration in Junos OS

Verification and troubleshooting IKE and IPsec is much the same as with site-to-site VPNs using preshared keys. The only difference is the use of the certificate for IKE identification, authentication, and encryption. Additional specific application notes for the Junos OS can be found on Juniper Networks Knowledge Base at http://kb.juniper.net. In particular, article KB10182 lists several application notes related to VPN configuration and troubleshooting. Also, more details can be found in Configuration Guides, downloadable at www.juniper.net/techpubs/.

Confirming IKE (Phase 1) Status

The first step to confirm VPN status is to check the status of any IKE phase 1 security associations. PKI, in relation to IPsec tunnels, happens during phase 1 setup. If phase 1 is complete, this indicates that PKI was also successful. The following is the CLI command to verify the state of IKE phase 1.

```
root@CORPORATE> show security ike security-associations
```

You can see that the remote peer is 2.2.2.2. The State shows UP. If the State shows DOWN, or if there is no IKE security associations present, there is a problem with phase 1 establishment. Confirm that the remote peer IKE ID, IKE policy, and external interfaces are all correct. Common errors include incorrect IKE policy parameters such as wrong Mode type (Aggr or Main); PKI issues (these are covered later in the troubleshooting section); or phase 1 proposals (all must match on both peers). Incorrect external interface is another common misconfiguration. This interface must be the correct interface that would receive the IKE packets. If configurations have been checked, and there are no PKI-related issues, check kmd log for any errors or run traceoptions (see troubleshooting section later in this application note).

Also, make note of Index number 20. This value is unique for each IKE security association and allows you to get more details from that particular security association as shown in the following.

```
root@CORPORATE> show security ike security-associations index 20 detail
IKE peer 2.2.2.2, Index 20,
  Role: Responder, State: UP
  Initiator cookie: af4f78bc135e4365, Responder cookie: 48a35f853ee95d21
  Exchange type: Main, Authentication method: RSA-signatures
  Local: 1.1.1.2:500, Remote: 2.2.2.2:500
  Lifetime: Expires in 23282 seconds
  Algorithms:
    Authentication : sha1
    Encryption : 3des-cbc
    Pseudo random function: hmac-sha1
  Traffic statistics:
    Input bytes : 10249
    Output bytes : 4249
    Input packets: 10
    Output packets: 9
  Flags: Caller notification sent
  IPsec security associations: 2 created, 1 deleted
  Phase 2 negotiations in progress: 0
```

The detail command gives much more information, which includes the Role (Initiator or Responder). This is useful to know because troubleshooting is usually always best done on the peer that has the Responder role. Also shown are details regarding the authentication and encryption algorithms used, the phase 1 lifetime, and traffic statistics. Traffic statistics can be used to verify that traffic is flowing properly in both directions. Finally, note also the number of IPsec security associations created or in progress. This can help to determine the existence of any completed phase 2 negotiations.
Confirming IPsec (Phase 2) Status

As mentioned previously, PKI is relevant for phase 1 setup. Phase 2 happens the same as with non-certificate-based VPNs. Once IKE phase 1 is confirmed, run the following command to view IPsec (phase 2) security associations.

```
root@CORPORATE> show security ipsec security-associations
total configured sa: 2
ID    Gateway        Port  Algorithm       SPI      Life:sec/kb  Mon vsys
<2    2.2.2.2        500   ESP:3des/sha1   bce1c6e0 1676/ unlim   -   0
>2    2.2.2.2        500   ESP:3des/sha1   1a24eab9 1676/ unlim   -   0
```

In the previous example, there is one IPsec SA pair, and the Port used is 500, which means no nat-traversal (nat-traversal would show port 4500 or random high port). Also, you can see the SPI used for both directions as the lifetime (in seconds) and usage limits or life size (in kilobytes). In the example shown previously, you see “1676/ unlim,” which means phase 2 lifetime is set to expire in 1676 seconds. There is no life size specified, and it shows unlimited. Phase 2 lifetime can differ from phase 1 lifetime because phase 2 is not dependent on phase 1 once the VPN is up. The “Mon” column refers to VPN monitoring status. If VPN monitoring were enabled, this would show U (up) or D (down). A hyphen (-) means VPN monitoring is not enabled for this SA. For more details regarding VPN monitoring, refer to the complete documentation for the Junos OS. Note that Vsys always shows 0.

Note also that the ID number 2 is shown previously. This is the Index value and is unique for each IPsec security association. You can view more details for a particular security association as shown in the following.

```
root@CORPORATE> show security ipsec security-associations index 2 detail
Virtual-system: Root
Local Gateway: 1.1.1.2, Remote Gateway: 2.2.2.2
Local Identity: ipv4_subnet(any:0,[0..7]=10.10.10.0/24)
Remote Identity: ipv4_subnet(any:0,[0..7]=192.168.168.0/24)
DF-bit: clear
Policy-name: tunnel-policy-out
  Direction: inbound, SPI: bce1c6e0, AUX-SPI: 0
  Hard lifetime: Expires in 1667 seconds
  Lifesize Remaining: Unlimited
  Soft lifetime: Expires in 1093 seconds
  Mode: tunnel, Type: dynamic, State: installed, VPN Monitoring: -
  Anti-replay service: enabled, Replay window size: 32
  Direction: outbound, SPI: 1a24eab9, AUX-SPI: 0
  Hard lifetime: Expires in 1667 seconds
  Lifesize Remaining: Unlimited
  Soft lifetime: Expires in 1093 seconds
  Mode: tunnel, Type: dynamic, State: installed, VPN Monitoring: -
  Anti-replay service: enabled, Replay window size: 32
```

The previous example shows Local Identity and Remote Identity. These elements comprise the proxy ID for this SA. Proxy ID mismatch is a very common reason for phase 2 failing to complete. For policy-based VPNs the proxy ID is derived from the tunnel policy. From the tunnel policy the local address and remote address are derived from the address book entries, and the service is derived from the application configured for the policy. If phase 2 fails due to a proxy ID mismatch, confirm from the policy which address book entries are configured and double-check the addresses to confirm they match what is being sent. Also, double-check the service to ensure that the ports match what is being sent.

Note that if multiple objects were configured in a tunnel policy for source address, destination address, or application, the resulting proxy ID for that parameter would be changed to zeroes. For example, assume the tunnel policy has multiple local addresses of 10.10.0.24 and 10.10.20.0/24, remote address 192.168.168.0/24, and application Junos-http. The resulting proxy-ID would be local 0.0.0.0/0, remote 192.168.168.0/24, service 80.

This can affect interoperability if the remote peer is not also configured for the second subnet. Also, for some third-party vendors you might need to manually enter the proxy ID to match. If IPsec cannot complete, check the kmd log or set traceoptions as detailed in the troubleshooting section of this application note.
Checking Statistics and Errors for an IPsec SA
The following command is used to check ESP and AH counters and for any errors with particular IPsec security associations.

```
root@CORPORATE> show security ipsec statistics index 2
ESP Statistics:
  Encrypted bytes:           674784
  Decrypted bytes:           309276
  Encrypted packets:           7029
  Decrypted packets:           7029
AH Statistics:
  Input bytes:                    0
  Output bytes:                   0
  Input packets:                  0
  Output packets:                 0
Errors:
  AH authentication failures: 0, Replay errors: 0
  ESP authentication failures: 0, ESP decryption failures: 0
  Bad headers: 0, Bad trailers: 0
```

You normally do not want to see error values other than zero. However, if you are experiencing packet loss issues across a VPN, one approach is to run the previous command multiple times and confirm that the Encrypted and Decrypted packet counters are incrementing. Also, see if any of the error counters increment while you are experiencing the issue. It might also be necessary to enable security flow traceoptions (see troubleshooting section) to see which ESP packets are experiencing errors and why.

Testing Traffic Flow Across the VPN
Once you have confirmed status of phase 1 and phase 2, the next step is to test traffic flow across the VPN. You can ping from local host PC to remote host PC. You can also initiate pings from the Junos OS-based device. The following is an example of ping testing from the Junos OS-based device to the remote PC host.

```
root@CORPORATE> ping 192.168.168.10 interface ge-0/0/0 count 5
PING 192.168.168.10 (192.168.168.10): 56 data bytes
64 bytes from 192.168.168.10: icmp_seq=0 ttl=127 time=8.287 ms
64 bytes from 192.168.168.10: icmp_seq=1 ttl=127 time=4.119 ms
64 bytes from 192.168.168.10: icmp_seq=2 ttl=127 time=5.399 ms
64 bytes from 192.168.168.10: icmp_seq=3 ttl=127 time=4.361 ms
64 bytes from 192.168.168.10: icmp_seq=4 ttl=127 time=5.137 ms
--- 192.168.168.10 ping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max/stddev = 4.119/5.461/8.287/1.490 ms
```

Note that when initiating pings from the Junos OS-based device, the source interface needs to be specified to ensure that route lookup is correct and the appropriate zones can be referenced in policy lookup. In this case because ge-0/0/0.0 resides in the same security zone as the local host PC, ge-0/0/0 must be specified in pings so that the policy lookup can be from zone “trust” to zone “untrust.” Additionally, a ping can be initiated from the remote host to the local host, or from the SSG5 as shown in the following example.

```
ssg5-> ping 10.10.10.10 from ethernet0/6
Type escape sequence to abort

Sending 5, 100-byte ICMP Echos to 10.10.10.10, timeout is 1 seconds from ethernet0/6
!!!
Success Rate is 100 percent (5/5), round-trip time min/avg/max=4/4/5 ms
```
If pings fail from either direction, this might indicate an issue with routing, policy, end host, or the encryption/decryption of the ESP packets. One way to check is to view IPsec statistics to see if any errors are reported. Also, you can confirm end host connectivity by pinging from a host on the same subnet as the end host. If the end host is reachable by other hosts, the issue is probably not with the end host. For routing and policy issues, enable security flow traceoptions. This topic is documented in other VPN-related Junos OS application notes.

**Troubleshooting Basics**

Basic troubleshooting begins by first isolating the issue and focusing the debugging efforts on the area where the problem is occurring. One common approach is to start with the lowest layer of the OSI model and work your way up the OSI stack to confirm at which layer the failure occurs.

Following this methodology, the first step to troubleshooting is to confirm the physical connectivity of the Internet link at the physical and data link levels. Next, using ping, confirm that the Junos OS-based device has connectivity to the Internet next hop, followed by confirming connectivity to the remote IKE peer. If all has been confirmed, verify that IKE phase 1 can complete by running the verification commands as shown previously. Once phase 1 is confirmed, verify phase 2. Finally, confirm traffic is flowing across the VPN. If the VPN is not in UP state, there is little reason to test any transit traffic across the VPN. Likewise, if phase 1 was not successful, checking for phase 2 issues is pointless.

To troubleshoot issues further at the different levels, configure traceoptions. Traceoptions are enabled in configuration mode and are a part of a Junos OS configuration. This means that a configuration commit is necessary before a traceoption can take effect. Likewise, removing traceoptions requires deleting or deactivating the configuration, followed by a commit. By enabling a traceoption flag, the data from the traceoption is written to a log file that might be predetermined or manually configured and stored in flash memory. This means that any trace logs are retained even after a system reboot. Keep in mind the available storage on flash before implementing traceoptions. You can check your available storage as shown in the following.

```
root@CORPORATE> show system storage
Filesystem              Size       Used      Avail  Capacity   Mounted on
/dev/ad0s1a             213M        74M       137M       35%  /
devfs                   1.0K       1.0K         0B      100%  /dev
devfs                   1.0K       1.0K         0B      100%  /dev/
/dev/md0                180M       180M         0B      100%  /Junos
/cf                     213M        74M       137M       35%  /Junos/cf
devfs                   1.0K       1.0K         0B      100%  /Junos/dev/
procfs                  4.0K       4.0K         0B      100%  /proc
/dev/bo0s1e              24M        13K        24M        0%  /config
/dev/md1                168M       7.6M       147M        5%  /mfs
/cf/var/jail            213M        74M       137M       35%  /jail/var
```

As shown in the previous example, /dev/ad0s1a represents the onboard flash memory and is currently at 35 percent capacity. You can also view available storage on Juniper Networks J-Web Software homepage under System Storage. The output of all traceoptions writes to logs stored in directory /var/log. To view a list of all logs in /var/log directory, run operational mode command: “show log.”
Checking traceoption Logs

As noted earlier, enabling traceoptions begins the logging of the output to the filenames specified or to the default log file for the traceoption. View the appropriate log to see the trace output. The following are the commands to view the appropriate logs.

```
root@CORPORATE> show log kmd
root@CORPORATE> show log pkid
root@CORPORATE> show log security-trace
root@CORPORATE> show log messages
```

Logs can also be uploaded to an FTP server with the “file copy” command.

Syntax (operational mode):

```
file copy <path/filename> <dest-path/filename>
```

Example:

```
root@CORPORATE> file copy /var/log/kmd ftp://10.10.10.10/kmd.log
ftp://10.10.10.10/kmd.log 100% of 35 kB 12 MBps
```

Troubleshooting IKE, PKI, and IPsec Issues

To view success or failure messages in IKE or IPsec, view the kmd log with the following command: “show log kmd.” Although the kmd log can show some general messages, it might be necessary to obtain additional details. For details, enable IKE and PKI traceoptions. Note that as a general rule, it is always best to troubleshoot on the peer that is the Responder. Obtaining trace output from the Initiator can also be useful in conjunction. However, obtaining trace output from only the Initiator alone might not reveal the cause of a failure.

Enabling IKE traceoptions for Phase 1 and Phase 2 Negotiation Issues

The following is an example of all IKE traceoptions.

```
root@CORPORATE> configure
Entering configuration mode
[edit]
root@CORPORATE# edit security ike traceoptions
[edit security ike traceoptions]
root@CORPORATE# set file ?
Possible completions:
<filename> Name of file in which to write trace information
files Maximum number of trace files (2..1000)
match Regular expression for lines to be logged
no-world-readable Don’t allow any user to read the log file
size Maximum trace file size (10240..1073741824)
world-readable Allow any user to read the log file
[edit security ike traceoptions]
root@CORPORATE# set flag ?
Possible completions:
all Trace everything
certificates Trace certificate events
database Trace security associations database events
general Trace general events
ike Trace IKE module processing
parse Trace configuration processing
policy-manager Trace policy manager processing
routing-socket Trace routing socket messages
timer Trace internal timer events
```
By default, if no file name is specified, all IKE traceoptions write to the kmd log. However, you can specify a different filename if desired. To write trace data to the log you must specify at least one flag option. Option “file size” determines the maximum size of a log file in bytes. For example, 1m or 1000000 generates a maximum file size of 1 MB. Option “file files” determines the maximum number of log files that are generated and stored in flash. Remember to commit the changes to start the trace.

In addition to IKE traceoptions, enabling PKI traceoptions is also a good idea to isolate whether an IKE failure is related to the certificate or due to a non-PKI issue. The following is an example of all PKI traceoptions.

```plaintext
root@CORPORATE> configure
Entering configuration mode
[edit]
root@CORPORATE# edit security pki traceoptions
[edit security pki traceoptions]
root@CORPORATE# set file ?
Possible completions:
<filename>           Name of file in which to write trace information
files                Maximum number of trace files (2..1000)
match                Regular expression for lines to be logged
no-world-readable    Don’t allow any user to read the log file
size                 Maximum trace file size (10240..1073741824)
world-readable       Allow any user to read the log file
[edit security pki traceoptions]
root@CORPORATE# set flag ?
Possible completions:
    all                  Trace with all flags enabled
    certificate-verification PKI certificate verification tracing
    online-crl-check     PKI online crl tracing
[edit security pki traceoptions]
root@CORPORATE# commit and-quit
commit complete
Exiting configuration mode
```

The same parameters as with IKE traceoptions also apply to the PKI traceoptions except that the default filename for all PKI-related traces can be found in the pkid log. The following is an example of recommended IKE and PKI traceoptions for troubleshooting most IKE setup issues with certs.

```plaintext
root@CORPORATE> configure
Entering configuration mode
[edit]
root@CORPORATE# edit security ike traceoptions
[edit security ike traceoptions]
root@CORPORATE# set file size 1m
[edit security ike traceoptions]
root@CORPORATE# set flag ike
[edit security ike traceoptions]
root@CORPORATE# set flag policy-manager
[edit security ike traceoptions]
root@CORPORATE# set flag routing-socket
[edit security ike traceoptions]
root@CORPORATE# set flag certificates
[edit security ike traceoptions]
root@CORPORATE# top edit security pki traceoptions
[edit security pki traceoptions]
root@CORPORATE# set file size 1m
[edit security pki traceoptions]
root@CORPORATE# set flag all
[edit security pki traceoptions]
root@CORPORATE# commit and-quit
commit complete
Exiting configuration mode
```
Reviewing Logs for Success/Failure Messages

The following are some excerpts of successful and common failing IKE phase 1 and phase 2 conditions. The output is from the following command: “show log kmd.”

Phase 1 and Phase 2 Successful

```
```

As shown in the previous example, the local address is 1.1.1.2, and the remote peer is IKE ID type hostname with FQDN of ssg5.juniper.net. Also, udp: 500 indicates that no nat-traversal was negotiated. You should see a phase 1 done message along with the role (initiator or responder). Next, you should also see a phase 2 done message with proxy ID information. At this point you can confirm that the IPsec SA is up using the verification commands mentioned in the previous section.

Phase 1 Failing to Complete, Example 1

```
Nov 7 11:52:14 Phase-1 [responder] failed with error(No proposal chosen) for local=unknown(any:0,[0..0]=) remote=fqdn(udp:500,[0..15]=ssg5.juniper.net)
Nov 7 11:52:14 1.1.1.2:500 (Responder) <-> 2.2.2.2:500 { 011359c9 ddef501d - 2216ed2a bfc50f5f [-1] / 0x00000000 } IP; Error = No proposal chosen (14)
```

In the previous example, the local address is 1.1.1.2, and the remote peer is IKE ID type hostname with FQDN of ssg5.juniper.net. The role is responder. The reason for failing is due to No proposal chosen. This is likely due to mismatched phase 1 proposals. To resolve this issue, confirm that phase 1 proposals match on both peers. Also, confirm that a tunnel policy exists for the VPN.

Phase 1 Failing to Complete, Example 2

```
Nov 7 12:06:36 Unable to find phase-1 policy as remote peer:2.2.2.2 is not recognized.
Nov 7 12:06:36 Phase-1 [responder] failed with error(Authentication failed) for local=ipv4(udp:500,[0..3]=1.1.1.2) remote=ipv4(any:0,[0..3]=2.2.2.2)
Nov 7 12:06:36 1.1.1.2:500 (Responder) <-> 2.2.2.2:500 { f725ca38 dad47583 - dab1ba4c ae26674b [-1] / 0x00000000 } IP; Error = Authentication failed (24)
```

In the previous example, the local address is 1.1.1.2, and the remote peer is 2.2.2.2. The role is responder. The reason for failing indicates that the peer is not recognized and Authentication failed. In the case of IKE with PKI certs, peer not recognized typically indicates incorrect IKE ID type was specified or IKE ID was not entered correctly. This needs to be checked first before the phase 1 proposal is checked. To resolve this issue, confirm that the local peer has the correct peer IKE ID type specified based on how the remote peer cert was generated. Also, confirm that the local peer is configured with the correct ID information based on the SubjectAlternativeName or DN information in the received remote peer certificate.
**Phase 1 Failing to Complete, Example 3**

Nov 7 13:52:39 Phase-1 [responder] failed with error(Timeout) for local=unknown(any:0,[0..0]=) remote=ipv4(any:0,[0..3]=2.2.2.2)

In the previous example, the remote peer is 2.2.2.2. The role is responder. This error means that IKE phase 1 was waiting for a response, but did not receive one in time. This could indicate that either the IKE packet might have been lost en route to the remote peer or perhaps the remote peer response was not received. This message might also indicate an issue with PKI failing. The timeout could be a result of waiting on a response from the PKI daemon. If that is the case, review PKI trace option output to see if there is a problem with PKI.

**Phase 1 Successful, Phase 2 Failing to Complete, Example 1**


Nov 7 11:52:14 1.1.1.2:500 (Responder) <-> 2.2.2.2:500 { cd9dff36 4888d398-6b0d3933 f0bc8e26 [0] / 0x1747248b } QM; Error = No proposal chosen (14)

In the previous example shown, the local address is 1.1.1.2, and the remote peer is IKE ID type hostname with FQDN of ssg5.juniper.net. The role is responder. You can clearly see that phase 1 was successful based on the “Phase-1 [responder] done” message. The reason for failing is due to No proposal chosen during phase 2. The issue is likely a phase 2 proposal mismatch between the two peers. To resolve this issue, confirm that phase 2 proposals match on both peers.

**Phase 1 Successful, Phase 2 Failing to Complete, Example 2**


Nov 7 11:52:14 Failed to match the peer proxy ids p2_remote=ipv4_subnet(any:0,[0..7]=192.168.168.0/24) p2_local=ipv4_subnet(any:0,[0..7]=10.10.20.0/24) for the remote peer:ipv4(udp:500,[0..3]=2.2.2.2)

Nov 7 11:52:14 KMD_PM_P2_POLICY_LOOKUP_FAILURE: Policy lookup for Phase-2 [responder] failed for pl_local=ipv4(udp:500,[0..3]=1.1.1.2) pl_remote=ipv4(udp:500,[0..3]=2.2.2.2) p2_local=ipv4_subnet(any:0,[0..7]=10.10.20.0/24) p2_remote=ipv4_subnet(any:0,[0..7]=192.168.168.0/24)

Nov 7 11:52:14 1.1.1.2:500 (Responder) <-> 2.2.2.2:500 { 41f638eb cc22bbfe-43fd0e85 b4f619d5 [0] / 0xc77faefc } QM; Error = No proposal chosen (14)

In the previous example, the local address is 1.1.1.2, and the remote peer is IKE ID type hostname with FQDN of ssg5.juniper.net. The role is responder. You can clearly see that phase 1 was successful based on the “Phase-1 [responder] done” message. The reason for failing might seem to indicate that No proposal was chosen. However, in this case you also see the message “Failed to match the peer proxy ids,” which means that the proxy ID received did not match what was expected. You can see that what was received was phase 2 proxy ID of local=192.168.168.0/24, service=any. Based on the configuration example, local address was expected to be 10.10.10.0/24. This address does not match the configurations on the local peer, and the proxy ID match fails. This results in the error and phase 2 failing to complete. To resolve this issue, correct the address book entry or configure the proxy ID on either peer so that it matches the other peer.
Common Problems Related to IKE and PKI

The following list is not comprehensive. To get a better understanding of the reason for any IKE or PKI failures, enabling traceoptions obtains much more detail than the normal log entries. By reviewing traceoption logs, one can usually get a better idea of the cause of failure. This application note does not go into details of all IKE and PKI problems and details of the trace outputs. That is best handled by qualified and experienced persons. Therefore, if more detailed analysis is required, contact Juniper Networks JTAC support or visit the Juniper Networks Technical Support website at www.juniper.net/support for further assistance. Some of the problems related to IKE and PKI are as follows:

- Clock, date, time zone, or daylight savings settings are incorrect. Use NTP to keep the clock accurate.
- Make sure to use a two-letter country code in the “C=” (country) field of the DN. For example, use “US” and not “USA” or “United States.” Some CAs require that the Country field of the DN be populated and only with a two-letter code.
- If the peer cert uses multiple “OU=” or “CN=” fields, use distinguished name with container method (the sequence must be maintained and is case sensitive).
- If certificate is not valid yet, check the system clock. Otherwise adjust the system time zone or just add a day in clock for quick tests.
- Make sure that the configured IKE ID type and value match.
- PKI might fail due to Revocation Check failure. One way to see if that is the case is to temporarily disable revocation check and see if IKE phase 1 is able to complete. To disable revocation checking, use the following command in configure mode:  “set security pki ca-profile <ca-profile> revocation-check disable” followed by “commit.”

PKI FAQ

Q. Does Juniper provide a CA with its products?
A. No, a customer who wants to use a PKI needs to obtain third-party CA software to implement the PKI or use a service such as VeriSign.

Q. What version of X.509 certs are supported (v1 or v3)?
A. Juniper does not support either. However, you need to use v3 if you want to use the SubjectAlternativeName extension field for a non-DN IKE ID type (for example, IP address, e-mail address, or FQDN).

Q. Does the Junos OS-based device support multiple certificates?
A. Yes, the Junos OS-based device can generate multiple key pairs, multiple cert requests, and have multiple local certificates loaded. The specific quantity depends on the particular platform.

Q. Can the Junos OS-based device use the same DN for different local certs?
A. In a particular Juniper device, Juniper does not support multiple certs with the same Subject (or DN) name. Therefore, Juniper recommends you use a separate subject name for every key pair to avoid confusion. Some CAs also have limitations on supporting multiple key pairs for the same subject name.

Q. Can the Junos OS-based device auto-generate CN values, such as FQDN and serial number in the DN?
A. The Junos OS does not auto-generate these CN values. The FQDN or any other CN values must be specified during the certificate request procedure.

Q. Does the Junos OS-based device support a hierarchical CA chain?
A. Yes, the Junos OS-based device can validate certs up through a chain of CA certs.

Q. How many levels of a CA chain can the Junos OS-based device validate?
A. Seven.

Q. I’ve got many levels in my chain, and my CDP servers are slow. How do I keep IKE from timing out?
A. Try adjusting the refresh interval for the CRL such that the CRL is not checked as frequently. This is at the expense of potentially allowing a cert that might have been revoked by the CA.

Q. Does Juniper support PKCS10 for certificate requests?
A. Yes, P10 cert requests can be generated by the Junos OS-based device and copied from the CLI, sent via e-mail, or uploaded onto an FTP server.
Q. Does Juniper support PKCS12 cert packages?
A. No, a Junos OS-based device does not accept a PKCS12 file. The Junos OS-based device must generate its own
private key. Also, a Juniper device does not generate a PKCS12 file for exporting its private/public keys and certificate.
This philosophy helps minimize the possibility that someone could steal a device’s keys and thereby impersonate
that device.

Q. Does the private key ever leave the Junos OS-based device?
A. No, but in future Junos OS releases, the private key might be copied from the active to the backup unit of a high
availability (HA) or JSRP pair as a runtime object (RTO).

Q. What special characters should I be leery of?
A. Juniper supports printable Strings, minus reserved characters it uses as delimiters, such as the comma. Names
with '_' or underscore can also potentially cause problems.

Q. What RFC does Juniper support for PKI?
A. RFC 3280 is followed. Required security features of RFC 2459 (predecessor of RFC 3280) are included.

Q. What are the possible PKI objects stored in flash and runtime memory?
A. CA cert, CA CRL, CA-profile config, local key-pair, and local cert or pending-cert.

Q. How do these relate?
A. Each CA cert typically uses three objects (CA cert, CRL, and CA-profile config). Each local cert uses two objects (cert
and key-pair). A pending cert is just a PKCS10 file that has been generated and sent to a CA. Once the signed cert
from the CA is installed, the “pending cert” object is replaced with the “local cert.”

Q. What are average sizes for these various items?
A. Average size of items:
- CRL, varies depending on how many certs a particular CA has revoked:
  minimum 300 bytes to maximum 5 MB
- Cert, average 2 K bytes each
- Key pair, average 1 KB each
- CA-profile configuration, average 500 bytes each.

Q. What is the maximum size of a CRL?
A. The maximum size supported in Junos OS version 8.5 is 5 MB.

Q. How do you disable CRL checking?
A. CRL checking is configurable per CA-profile. The command syntax to disable CRL checking is “set security pki ca-
profile <ca-profile> revocation-check disable” followed by “commit.”

Q. Why doesn’t the Junos OS-based device use or support two sets of keys for VPN?
A. Most people set up a PKI for e-mail and file encryption and signing. You should use two sets of keys. While you
certainly want two sets of keys when encrypting e-mails and files (one set for signing and one set for encryption),
you don’t need that for VPN. RSA keys are only used for authentication in IPsec and you do not need the second set
of keys for long-term storage of encrypted material.

Q. Does Juniper support CA cross-certiﬁcation? In other words, if one Juniper device uses a cert from one root CA, and
another Juniper device uses a cert from a different root CA, and the two root CAs are cross-certiﬁed, do the two Juniper
devices properly validate each other’s certs and form the VPN tunnel?
A. Yes, it can be done by using PKCS7 cert type. Via the cross-certificate, a full certificate path can be formed to the
root certificate that is stored locally.

Q. Which certificate formats does the Junos OS support?
A. Juniper follows the PKI profile described in RFC 3280. We support installation of end-entity (EE) or certificate
authority (CA) certificates. Juniper supports encodings, including X509 or PKCS7, DER or PEM. The Junos OS is
compatible with X509 v3 and can handle extensions defined in RFC 3280.
Q. Does the Junos OS support HA for PKI certificates?
A. Junos OS 8.5 does not currently support HA or JSRP with PKI. Future releases might support the transferring of a device key-pair and local certs between two HA peers. Check release notes for upcoming releases to see if this is supported in releases later than 8.5.

Q. How does the public key of a key pair get bound to a cert request?
A. When generating a new key pair, a certificate-id must be specified. This certificate-id is also used for the cert request and again when the local cert is loaded. To completely delete a certificate request and key pair, use the "clear security pki" operational mode command. Two clear operations are required. One operation clears the cert request. The other operation clears the key pair.

Q. Why not delete both the cert and the key pair at the same time?
A. Some administrators prefer the ability to keep the same key pair but use a new certificate with them. This allows for the deletion of the old certificate without destroying the old key pair.

Q. Does the Junos OS support DSA keys?
A. No, currently only RSA keys are supported. DSA keys might be supported in future releases.

Q. Is the Junos OS ICSA certified?
A. Not yet, although many of the security features in the Junos OS were sourced from Juniper Networks ScreenOS Software products, which are certified for version 1.2. For more information regarding ICSA certification, refer to the ICSA Labs website: http://www.icsalabs.com/.

Q. Is OCSP supported for revocation checking?
A. Not currently, but this might be supported in a future release.

Q. Are there special characters to consider when doing PKI?
A. Yes, the comma "," is a special character in ASN.1 DN and requires an escape character to use, which is the backslash "\". The UTF-8 encoded string should not have any of the following characters:
- a space or "#" character occurring at the beginning of the string
- a space character occurring at the end of the string
- one of the characters ":,,+" plus, "" double quote, \" backslash, "<" less than or left triangle bracket, ">" greater than or right triangle bracket, or ";" semicolon
  If the comma "," character needs to be escaped, it should be prefixed by a backslash (\ ASCII 92).

Q. I want my CDP function to communicate through a VPN tunnel. How do I set that up so the Junos OS-based device sources the IP from an internal interface that matches a tunnel definition and does not source the packet from the egress which doesn't match a tunnel policy (even though that interface is the tunnel endpoint/gateway IP)?
A. This is not currently supported in the Junos OS.
Appendix A: Show Configuration

The output of the show configuration is shown in the following section. For reference, traceoption configurations are highlighted for troubleshooting purposes. Always remember to delete or deactivate the traceoptions once troubleshooting is complete.

```
root@CORPORATE> show configuration | no-more
system {
    host-name CORPORATE;
    time-zone PST8PDT;
    root-authentication {
        encrypted-password "$1$wUchK29B$IACQWVtsyF2PBKtllAir."; ## SECRET-DATA
    }
    name-server {
        4.2.2.1;
        4.2.2.2;
    }
    services {
        ssh;
        telnet;
        web-management {
            http {
                interface ge-0/0/0.0;
            }
        }
    }
    syslog {
        user * {
            any emergency;
        }
        file messages {
            any any;
            authorization info;
        }
        file interactive-commands {
            interactive-commands any;
        }
    }
}
interfaces {
    ge-0/0/0 {
        unit 0 {
            family inet {
                address 10.10.10.1/24;
            }
        }
    }
    ge-0/0/3 {
        unit 0 {
            family inet {
                address 1.1.1.2/30;
            }
        }
    }
}
routing-options {
    static {
        route 0.0.0.0/0 next-hop 1.1.1.1;
    }
```
```xml
} security {
  ike {
    traceoptions {
      flag ike;
      flag policy-manager;
      flag routing-socket;
      flag certificates;
    }
    proposal rsa-prop1 {
      authentication-method rsa-signatures;
      dh-group group2;
      authentication-algorithm sha1;
      encryption-algorithm 3des-cbc;
    }
    policy ike-policy1 {
      mode main;
      proposals rsa-prop1;
      certificate {
        local-certificate ms-cert;
        trusted-ca use-all;
        peer-certificate-type x509-signature;
      }
    }
    gateway ike-gate {
      ike-policy ike-policy1;
      dynamic hostname ssg5.juniper.net;
      external-interface ge-0/0/3;
    }
  }
} ipsec {
  policy vpn-policy1 {
    perfect-forward-secrecy {
      keys group2;
    }
    proposal-set standard;
  }
  vpn ike-vpn {
    ike {
      gateway ike-gate;
      ipsec-policy vpn-policy1;
    }
  }
} zones {
  security-zone untrust {
    address-book {
      address remote-net 192.168.168.0/24;
    }
    host-inbound-traffic {
      system-services {
        ike;
      }
    }
    interfaces {
      ge-0/0/3.0;
    }
  }
}
security-zone trust {
    address-book {
        address local-net 10.10.10.0/24;
    }
    host-inbound-traffic {
        system-services {
            all;
        }
    }
    interfaces {
        ge-0/0/0.0;
    }
}
policies {
    from-zone trust to-zone untrust {
        policy tunnel-policy-out {
            match {
                source-address local-net;
                destination-address remote-net;
                application any;
            }
            then {
                permit {
                    tunnel {
                        ipsec-vpn ike-vpn;
                        pair-policy tunnel-policy-in;
                    }
                }
            }
        }
        policy any-permit {
            match {
                source-address any;
                destination-address any;
                application any;
            }
            then {
                permit {
                    source-nat {
                        interface;
                    }
                }
            }
        }
    }
    from-zone untrust to-zone trust {
        policy tunnel-policy-in {
            match {
                source-address remote-net;
                destination-address local-net;
                application any;
            }
            then {
                permit {
                    tunnel {
                        ipsec-vpn ike-vpn;
                        pair-policy tunnel-policy-out;
                    }
                }
            }
        }
    }
}
Appendix B: Administering Common CAs (Certificate Authorities)

There are several vendors of certificate authorities available. Junos OS-based devices work with the following:

- VeriSign
- Entrust
- Microsoft Windows 2000 Advanced Server

Open source code from OpenSSL, though not officially supported by Juniper, works with the Junos OS if set up properly. The choice of CA might be dependent on whether you want a standalone CA solution or you are relying on a third party such as VeriSign. The information in this appendix assumes that you want a standalone server for which you are the CA administrator.

It is beyond the scope of this document to address detailed procedures for interoperating with all of these CAs. However, given the basic concepts detailed in the previous section, a Junos OS-based device administrator should be able to work with the respective CA administrator to enroll and use certs on the Junos OS-based device.

The following two sections are example administrative procedures using a Microsoft CA and an open source CA from OpenSSL.
Microsoft Windows 2000 CA

The Microsoft CA—provided on Windows 2000 Advanced Server—offers CA services through a Web interface, including the support of a CDP. Microsoft also has a patch available that activates SCEP. Microsoft does not support OCSP.

The Microsoft CA is usually located at the http://host.domain/certsrv. If the Microsoft CA server has been set up and enabled, as you browse to the server URL, you should see the following screen.

![Microsoft Certificate Services - TACLAB](image)

This is where you can retrieve the CA cert and CRL. Select “Retrieve the CA certificate or certificate revocation list” and click “Next>.”

![Retrieve The CA Certificate Or Certificate Revocation List](image)

Select the CA and click “Download CA certificate.” You should get a pop-up window asking where you want to save this cert. Choose a location on your local file system and save the cert as some file with a .cer extension (for example, certnew.cer).

Now retrieve the CRL.

![Retrieve The CA Certificate Or Certificate Revocation List](image)
Select the “Download latest certificate revocation list” option. You should get a pop-up window asking where you want to save the CRL. Pick a location on your local file system and save the CRL as some file with a .crl extension (for example, certcrl.crl).

By returning to the homepage, you can select the option to request a certificate.

Select “Request a certificate” and click “Next>.” Select “Advanced request.”

Next, select the PKCS #10 option.
At this point, you can paste a copy of the cert request into the window as shown. Click “Submit>.”

If the CA is set up to auto-vet, the next screen automatically displays.

Click on “Download CA certificate” to download your new local certificate. The nomenclature used on this screen by Microsoft is misleading. At this point it is not really a CA certificate you are downloading but your local Junos OS-based device cert. If the MS CA is not set up to auto-vet, the CA administrator must manually vet the cert request and generate the cert. When this is done, you can go back to the Microsoft CA homepage and select “Check on a pending certificate” to retrieve the newly generated local cert.
Microsoft systems support an application to view certificates. Double-click the .cer file and select the “Details” tab to see all the cert fields and their values. For example, the following shows the Issuer (CA) of the certificate:

You can also validate the SubjectAlternativeName field for a certificate. This field needs to contain the IKE ID types and values used in the Junos OS-based device IKE gateway definition.
Check for the existence of a CDP. The following shows the CDP field and value. Make sure the hostname can be resolved by the Junos OS-based device and that it is reachable by the Junos OS-based device.

In addition to viewing the cert, a Microsoft Windows system can decode and display a CRL.

For more information on administering the Microsoft CA, see the Microsoft support site.
OpenSSL CA

The OpenSSL code is free and simple code available from the http://www.openssl.org/ website. It uses a command line only and has no GUI or web interfaces. All input (for example, p10 cert requests) and all output (signed certs and CRLs) are usually PEM-encoded files.

To use the OpenSSL CA, download and install the openssl executable. Now perform the one-time CA setup. The following example uses a Windows system.

Initializing the CA

Create a working directory and “cd” to that directory. Once there, copy the “openssl.exe” and “openssl.cfg” files to there. A sample copy of the openssl.cfg file is included at the end of this appendix.

Once in that working directory, create some additional subdirectories as shown in the following.

- mkdir demoCA
- mkdir demoCA\private

Create the CA's own key pair and CA certificate.

- openssl req -x509 -newkey rsa:1024 -keyout demoCA\private\key.pem -out demoCA\ca-cert.pem -config openssl.cfg

NOTE: The private key for the CA is not encrypted with this command.

The “ca-cert.pem” file can be loaded into the Junos OS-based device as the CA cert.

Now you need to do the following to set up a “database” for the certs that are generated by this CA.

- mkdir demoCA\certstore
- echo 01 > demoCA\ca-cert.srl

Create a new but empty file called index.txt in the demoCA directory.

- edit demoCA\index.txt

Now “save” and “exit,” leaving it empty. The CA is now initialized.

Generating a Local Cert

For each Junos OS-based device that needs a cert, set the basic configuration items and certificate request settings as described in this application note.

Once you’ve generated a PKCS10 file, save that cert request into a file called “jsNAME.pkcs10.” To sign the cert request (PKCS10 file) generated by the Junos OS-based device, go to the OpenSSL CA working directory (the parent directory of the “demoCA” subdirectory created earlier).

Although the SubjectAlternativeName field information is in the Junos OS-based device’s PKCS10 cert request, the OpenSSL CA cannot sign it as is. The OpenSSL server attempts to strip that part out of the cert request. To have the cert populated with a SubjectAlternativeName field, you must edit a setting in the openssl.cfg file. However, that file must be modified for every cert you sign. The following examples provide steps to edit the openssl.cfg file.

edit openssl.cfg

Search for the SubjectAltName field. Reset the SubjectAltName field to the correct value for this particular Junos OS-based device cert that you are about to sign. For example:

subjectAltName=DNS:ssg5.juniper.net

To create and sign the cert, if the cert request from the Junos OS-based device is in a file called “jsNAME.pkcs10,” and the cert is created in a file called “jsNAME.cer,” issue the following command:

openssl ca -config openssl.cfg -in jsNAME.pkcs10 -out jsNAME.cer
The Junos OS-based device's local cert is now the "jsNAME.cer" file and can be loaded into the Junos OS-based device. A copy of this certificate is also created in the demoCA\certstore subdirectory with a name of NN.pem, where NN is the serial number of this cert.

This cert is in PEM format. To view the cert with the Microsoft cert viewer, the cert must be converted to the DER encoding format. To do this, edit the jsNAME.cer file. Delete everything except the ----BEGIN/END certificate--- lines and all the data between those lines. This allows Microsoft Windows to decode the file properly to display its contents. The OpenSSL CLI also has the ability to convert the PEM encoded cert to DER encoding. See the OpenSSL documentation for details.

**Revoking a Cert and Generating a New CRL**

To revoke a particular cert, find the serial number of the cert. For example, to revoke certificate with serial number "01," use the following commands:

```
openssl ca -config openssl.cfg -revoke demoCA\certstore\01.pem
```

You might see an error here. If you see an error, manually move the file using the following command:

```
mv demoCA\index.txt.new demoCA\index.txt
```

Next, generate the new CRL using the following command:

```
openssl ca -config openssl.cfg -gencrl -out crl.crl
```

The crl.crl file can now be loaded onto the Junos OS-based device. Load the CA cert, CRL, and local certificate, following the same steps as described in this application note.

**OpenSSL.cfg File Sample**

```
=============  
# OpenSSL example configuration file.
# This is mostly being used for generation of certificate requests.
# This definition stops the following lines choking if HOME isn’t defined.
HOME = .
RANDFILE = $ENV::HOME\.rnd

# Extra OBJECT IDENTIFIER info:
#oid_file = $ENV::HOME\.oid
oid_section = new_oids

# To use this configuration file with the “-extfile” option of the
# “openssl x509” utility, name here the section containing the
# X.509v3 extensions to use:
# extensions =
# (Alternatively, use a configuration file that has only
# X.509v3 extensions in its main [= default] section.)
# [ new_oids ]

# We can add new OIDs in here for use by ‘ca’ and ‘req’.
# Add a simple OID like this:
# testoid1=1.2.3.4
```
# Or use config file substitution like this:
# testoid2=${testoid1}.5.6

#################################################################
[ ca ]
default_ca = CA_default # The default ca section

[ CA_default ]
dir = .\\demoCA # Where everything is kept
certs = $dir\\certs # Where the issued certs are kept
crl_dir = $dir\\crl # Where the issued crl are kept
database = $dir\\index.txt # database index file.
new_certs_dir = $dir\\certstore # default place for new certs.
certificate = $dir\\ca-cert.pem # The CA certificate
serial = $dir\\ca-cert.srl # The current serial number
crl = $dir\\crl.pem # The current CRL
private_key = $dir\\private\\key.pem # The private key
RANDFILE = $dir\\private\\.rand # private random number file
x509_extensions = usr_cert # The extentions to add to the cert

# Extensions to add to a CRL. Note: Netscape communicator chokes
# on V2 CRLs so this is commented out by default to leave a V1 CRL.
# crl_extensions = crl_ext

default_days = 365 # how long to certify for
default_crl_days= 30 # how long before next CRL
default_md = md5 # which md to use.
preserve = no # keep passed DN ordering

# A few difference way of specifying how similar the request should
# look. For type CA, the listed attributes must be the same, and
# the optional and supplied fields are just that :-) 
policy = policy_match

# For the CA policy
[ policy_match ]
countryName = optional
stateOrProvinceName = optional
organizationName = optional
organizationalUnitName = optional
commonName = supplied
emailAddress = optional

# For the ‘anything’ policy
# At this point in time, you must list all acceptable ‘object’
# types.
[ policy_anything ]
countryName = optional
stateOrProvinceName = optional
localityName = optional
organizationName = optional
organizationalUnitName = optional
commonName = supplied
emailAddress = optional

#################################################################
[ req ]
default_bits = 1024
default_keyfile = privkey.pem
distinguished_name = req_distinguished_name
attributes = req_attributes
x509_extensions = v3_ca # extensions to add to the self signed cert
# Passwords for private keys if not present they will be prompted
# for input_password = secret
# output_password = secret
# This sets a mask for permitted string types of which there are
# several options.
#
# default: PrintableString, T61String, BMPString.
# pkix : PrintableString, BMPString.
# utf8only: only UTF8Strings.
# nombstr : PrintableString, T61String (no BMPStrings or
# UTF8Strings).
#
# Mask: XXXX a literal mask value.
# WARNING: current versions of Netscape crash on BMPStrings or
# UTF8Strings
# so use this option with caution!
string_mask = nombstr

# req_extensions = v3_req # extensions to add to a cert request
[ req_distinguished_name ]
countryName = Country Name (2 letter code)
countryName_default = US
countryName_min = 2
countryName_max = 2
stateOrProvinceName = State or Province Name (full name)
stateOrProvinceName_default = Illinois
localityName = City or local name
localityName_default = Chicago
0.organizationName = demo-company.com
0.organizationName_default = netscreen.com

# we can do this but it is not typically required
#1.organizationName = Second Organization Name (eg, company)
#1.organizationName_default = Sales
organizationalUnitName = Org Unit
organizationalUnitName_default = CSE
commonName = Common Name
commonName_default = test-CA
commonName_max = 64
emailAddress = Email Address
emailAddress_default = admin@juniper.net
emailAddress_max = 40

# SET-ex3 = SET extension number 3
[ req_attributes ]
challengePassword = secretkey
challengePassword_min = 4
challengePassword_max = 20
unstructuredName = juniper.net
[ usr_cert ]

# These extensions are added when ‘ca’ signs a request.
# This goes against PKIX guidelines. Some CAs do this and some software requires this to avoid interpreting an end user certificate as a CA.

```
basicConstraints=CA:FALSE
```

# Here are some examples of the usage of nsCertType. If it is omitted the certificate can be used for anything *except* object signing. This is OK for an SSL server.

```
# nsCertType = server
# For an object signing certificate this would be used.
# nsCertType = objsign
# For normal client use this is typical
#
# nsCertType = client, email
#
# and for everything including object signing:
# nsCertType = client, email, objsign
#
# This is typical in keyUsage for a client certificate.
# keyUsage = nonRepudiation, digitalSignature, keyEncipherment
# This will be displayed in Netscape’s comment listbox.
nsComment = “OpenSSL Generated Certificate”
```

# PKIX recommendations harmless if included in all certificates.

```
subjectKeyIdentifier=hash
authorityKeyIdentifier=keyid,issuer:always
```

# This stuff is for subjectAltName and issuerAltName.
# Import the email address.
# subjectAltName=email:copy

```
subjectAltName=DNS:ssg5.juniper.net
```

# Copy subject details
# issuerAltName=issuer:copy
# nsCaRevocationUrl = http://www.domain.dom/ca-crl.pem
# nsBaseUrl
# nsRevocationUrl
# nsRenewalUrl
# nsCaPolicyUrl
# nsSslServerName

[ v3_req ]

# Extensions to add to a certificate request
basicConstraints = CA:FALSE
keyUsage = nonRepudiation, digitalSignature, keyEncipherment

[ v3_ca ]

# Extensions for a typical CA
# PKIX recommendation.
subjectKeyIdentifier=hash
authorityKeyIdentifier=keyid:always,issuer:always

# This is what PKIX recommends but some broken software chokes on critical extensions.
# basicConstraints = critical,CA:true
# Do this instead.
basicConstraints = CA:true

# Key usage: this is typical for a CA cert. However because it will
# prevent it being used as a test self-signed cert it is best
# left out by default.
# keyUsage = cRLSign, keyCertSign
# Some might want this also
# nsCertType = sslCA, emailCA
# Include email address in subject alt name: a PKIX recommendation
# subjectAltName=email:copy
# Copy issuer details
# issuerAltName=issuer:copy
# DER hex encoding of an extension: beware experts only!
# obj=DER:02:03
# Where ‘obj’ is a standard or added object
# You can even override a supported extension:
# basicConstraints= critical, DER:30:03:01:01:FF
[ crl_ext ]

# CRL extensions.
# Only issuerAltName and authorityKeyIdentifier make any sense in a
# CRL.
# issuerAltName=issuer:copy
# authorityKeyIdentifier=keyid:always,issuer:always

Summary

Juniper Networks® Junos® operating system runs on Juniper Networks J Series Services Routers and SRX Series Services Gateways and provides not only a powerful OS, but also a rich IP services tool kit. Junos OS has enhanced security and VPN capabilities via Juniper’s firewall/IPsec VPN platforms, which include the Juniper Networks SSG Series Secure Services Gateways.

This application note demonstrates the public key infrastructure (PKI) concepts, specifications, usage scenarios/decisions, and debugging in the Junos OS. It is meant to be more than just a step-by-step guide to configuring and using PKI. Additional details regarding digital certificates can be found in the Junos OS System Basics Guide: www.juniper.net/techpubs/software/Junos/.
Appendix C: DOD PKI Usage

The U.S. Federal Government (Department of Defense) maintains a PKI used by many entities, including the military.

DoD PKI Introduction

DoD PKI uses a custom PKI solution based on Netscape iPlanet CA server. Although the Junos OS does not officially support Netscape iPlanet CA, it does support much of what is required for DoD support. This custom PKI solution has defined its own certificate profiles and security policies that might differ from other CAs.

We need to examine the profile of DoD PKI and determine what is required to interoperate with DoD PKI. Here is the summary of the features of DoD PKI:

The DoD PKI comprises a three-layer certificate hierarchy. The root CA is “JITC DoD PKI Class 3 Root CA,” and the subordinate CA is “JITC DoD PKI Class 3 ID CA.” The subordinate CA issues all end-entity certificates.

A server certificate (which the Juniper device acquires) doesn’t have a SubjectAlternativeName extension field. An example is an e-mail certificate, which the NS Remote client acquires, and which normally has a SubjectAlternativeName field containing an IKE ID type of e-mail.

The DN of all DoD PKI certificates has multiple OU fields. A server certificate issued by DoD PKI has the DN form of the following:

- CN=<server DNS name or IP>,
- OU=<military/government organization>,
- OU=PKI,
- OU=DoD,
- O=U.S. Government,
- C=US

A user certificate has the DN form of the following:

1. Last Name
2. First Name
3. Middle Initial or Name (optional)
4. Generation [Jr., Sr., II, III, etc] (optional)
5. E-mail address
6. Organization (military/government or contractor)
7. City
8. State
9. Country

CA certificates can be downloaded from https://idca.nit.disa.mil/GetCACert.html. The certificates can be downloaded as two separate CA certificates (the root CA and the subordinate CA), or as a single PKCS7 envelope containing both.

The Root CRL is updated approximately every 30 days, while the ID CRL is automatically updated every 24 hours. The CRL distribution point is in the CA certificate with attributes of “certificaterevocationlist;binary.” There is no scope or filter defined in the LDAP URL.

To be able to interoperate with DoD PKI, the Junos OS needs to support several things. Details on what is required are shown in the following example.

Support multiple OU fields to comply with the DN convention of the DoD PKI. The Junos OS supports multiple OU entries. This can be specified when generating a PKCS10 cert request by adding multiple OU objects in the subject.
Example:

```
request security pki generate-certificate-request certificate-id test hostname user@idca.nit.disa.mil subject "CN=idca.nit.disa.mil,OU=DISA,OU=PKI,OU=DoD, O=U.S. Government,C=US"
```

This enhancement is not limited to just the OU or O fields of the DN—it applies to all fields, including S, L, and Country.

Support a CRL LDAP search with default attributes and filters. The LDAP URL of the DoD PKI doesn’t provide filters or a scope.

Support Certificate Chaining and multi-layer CRL verification. The DoD PKI is a two-layer CA hierarchy that is composed of a root CA and subordinate CA.

Support DN as peer gateway IKE ID type. The Junos OS supports distinguished-name as the IKE ID of a static or dynamic peer gateway.

Allow disabling of CRL-checking for easier debugging view. The Junos OS supports this in ca-profile settings.

**DoD PKI Setup**

Instructions for Juniper device IKE configuration based on DoD PKI authentication are shown in the following example. This is not a comprehensive list of instructions but more of an addendum to what is already outlined in the body of this application note. Refer to the detailed steps in previous sections.

**Generate PKCS10 and Retrieve Cert**

1. The DoD PKI uses a two-tiered hierarchy of CAs with the device certs sometimes considered the third or bottom tier. There is a root CA and a couple of subordinate CAs. You should retrieve and load the certs for all of these CAs in the Juniper device.

2. DoD PKI can only support one CN field in the DN.

3. The DoD PKI requires multiple OU fields. The Juniper device can generate multiple OU fields.

4. You can download the DoD CRL files, or you can automatically use LDAP to retrieve the CRL. If you do use LDAP, make sure that you have DNS set on the Juniper device. It needs this to resolve the name of the LDAP server. An easy way to test this is to ping the LDAP server from the Juniper device by name.

5. The CRL file can be larger than 20 KB. The Junos OS-based devices support up to 5 MB for the CRL.

**Set up IKE Using the Certs**

1. Set up the IKE gateway as usual by choosing a proposal that uses RSA algorithm.

2. You must specify distinguished-name as the IKE ID. Because the DoD PKI certs don’t support the SubjectAlternativeName v3 extensions, the default FQDN (hostname + domain-name) does not work.

3. In the IKE Gateway configuration, select the appropriate preferred local cert and peer CA cert. The Peer Type can be X509 or PKCS7. Try X509 first. If the tunnels don’t work, try the other format.

These steps should cover all required changes to allow the Juniper device to support IKE tunnels based on DoD-PKI authentication.
Appendix D: Simple Certificate Enrollment Protocol (SCEP)

During the normal life cycle of a PKI certificate, certificate expiration can become a challenge from an administration point of view. When a local-certificate expires, the administrator has to first delete the existing certificate, certificate request, and key pair; generate a new key pair; generate a new certificate request; and finally load the newly issued local certificate onto the device.

The SCEP protocol can significantly ease the administrative burden of managing expiration of local certificates by automatically re-enrolling and retrieving new certificates. Furthermore, SCEP can ease the process of the initial certificate request and retrieval process by automatically retrieving the certificate from the CA server (if the server supports SCEP).

SCEP is supported beginning with Junos OS version 9.0. The steps to configure SCEP are shown in the following example. Troubleshooting of SCEP issues can be performed by enabling PKI traceoptions within the security pki hierarchy.

Currently, only Microsoft SCEP is supported. VeriSign and Entrust are not currently supported but are expected to be in a future release.

Steps to Configure

1. Generate the key-pair.

   Syntax:
   
   request security pki generate-key-pair certificate-id <cert-id> size <512|1024|2048>
   
   Example:
   
   root@host> request security pki generate-key-pair certificate-id mscert1 size 1024
   Generated key pair mscert1, key size 1024 bits

2. Configure SCEP server under security pki hierarchy.

   Syntax:
   
   set security pki ca-profile <ca-profile-name> ca-identity <ca-id> enrollment url <scep-server-url> [retry <count>] [retry-interval <seconds>]
   
   Example:
   
   root@host> configure
   Entering configuration mode
   
   [edit]
   root@host# edit security pki
   [edit security pki]
   root@host# set ca-profile mscaprofile ca-identity msca2000 enrollment url http://172.19.50.129/certsrv/mscep/mscep.dll retry 3 retry-interval 3
   
   [edit security pki]
   root@host# show
   ca-profile mscaprofile {
     ca-identity msca2000;
     enrollment {
       url http://172.19.50.129/certsrv/mscep/mscep.dll;
       retry 3;
       retry-interval 3;
     }
   }
   
   [edit security pki]
   root@host# commit and-quit
3. Generate your local certificate by enrolling via SCEP.

Syntax:

```
request security pki local-certificate enroll ca-profile <ca-profile-name>
certificate-id <cert-id> challenge-password <challenge-string>
[ip-address <ip-address>|email <email>|domain-name <domain>]
subject “DC=<Domain component>,CN=<Common-Name>,OU=<Organizational-Unit-name>,O=<Organization-name>,L=<Locality>,ST=<state>,C=<Country>”
```

Example:

```
root@host> request security pki local-certificate enroll ca-profile msca-profile
certificate-id mscert1 challenge-password “” domain-name tacdomain.com subject “CN=testuser,OU=Support,O=Juniper Networks,L=Sunnyvale,ST=CA,C=US”
```

4. If necessary, also enroll your CA certificate.

Syntax:

```
request security pki ca-certificate enroll ca-profile <ca-profile-name>
```

Example:

```
root@host> request security pki ca-certificate enroll ca-profile msca-profile
Fingerprint:
Do you want to load the above CA certificate ? [yes,no] (no) yes
```

```
CA certificate for profile msca-profile loaded successfully
```

5. Once you have a certificate loaded, configure auto-re-enrollment.

Syntax:

```
set auto-re-enrollment certificate-id <cert-id> ca-profile-name <ca-profile-name> challenge-password <challenge-string> [re-enroll-trigger-time-percentage <percent>] [re-generate-keypair]
```

Example:

```
root@host> configure
Entering configuration mode
[edit]
root@host# edit security pki

[edit security pki]
root@host# set auto-re-enrollment certificate-id mscert1 ca-profile-name msca-profile challenge-password “” re-enroll-trigger-time-percentage 5 re-generate-keypair

[edit security pki]
root@host# show
ca-profile msca-profile {
  ca-identity msca-id;
  enrollment {
    url http://172.19.50.129/certsrv/mscep/mscep.dll;
    retry 3;
    retry-interval 3;
  }
}
```
auto-re-enrollment {
  certificate-id mscert1 {
    ca-profile-name msca-profile;
    challenge-password "$9$jx"; ## SECRET-DATA
    re-enroll-trigger-time-percentage 5;
    re-generate-keypair;
  };
}

[edit security pki]
root@host# commit and-quit

About Juniper Networks

Juniper Networks, Inc. is the leader in high-performance networking. Juniper offers a high-performance network infrastructure that creates a responsive and trusted environment for accelerating the deployment of services and applications over a single network. This fuels high-performance businesses. Additional information can be found at www.juniper.net.