Chapter 46

Public Key Infrastructure (PKI)

Introduction

This chapter describes how to configure the router for interaction with a Public Key Infrastructure (PKI). The PKI is part of the router's suite of security modules, and consists of a set of tools for managing and using certificates.

IPsec requires an encryption PAC card and may be subject to government export controls.

The router acts as an End Entity (EE) in an X.509 certificate-based PKI. More specifically, the router can communicate with Certification Authorities (CAs) and Certificate Repositories to request, retrieve and verify X.509 certificates, and allow protocols running on the router, such as ISAKMP, access to these certificates.

The following sections of this chapter summarise these concepts and describe the router's implementation of them. All the commands described in this chapter, except the SHOW commands, require Security Officer status.

In the unlikely event of the router's flash memory becoming corrupted, you may lose current feature licences, including those enabled by default. If this happens, contact your authorised distributor or reseller.

Overview of PKI

The tools that make up the Public Key Infrastructure allow the router to securely exchange public keys, while being sure of the identity of the keyholder.

Public Keys

Public key encryption involves the generation of two keys for each user: private and public. Material encrypted with a private key can be decrypted with the corresponding public key and vice versa. An individual's private key must be kept secret, but the public key may be known widely because it is impossible to calculate the private key from the public key. The advantage of public key encryption is that the private key need never be exchanged and so can be kept secure more easily than a shared secret key.

Message Encryption

One of the two main services provided by public key encryption is the exchange of encrypted messages. User 1 can send a secure message to user 2 by encrypting it with user 2's public key. Only user 2 is able to decrypt it because only user 2 has access to the corresponding private key.

Digital Signatures

The second main service provided by public key encryption is digital signing. Digital signatures both confirm the identity of the message's supposed sender and protect the message from tampering. Therefore they provide message authentication and non-repudiation. It is very difficult for the signer of a message to claim that the message was corrupted, or to deny that it was sent.

The process for creating a digital signature is:

1. A "hash" algorithm is used to generate a one-way hash from the message.
2. The hash is encrypted with the sender's private key.
3. The message, the encrypted hash, and information about the hash algorithm are transmitted to the receiver.
4. The receiver uses the sender's public key to decrypt the hash.
5. The receiver applies the same hash algorithm to the message to generate a hash.
6. The receiver compares the decrypted hash with the generated hash. If they are identical, the message was sent by the owner of that public key.

More information about public key encryption and message authentication can be found in Chapter 25, Compression and Encryption Services.

The exchange of encrypted messages and digital signatures are secure when the public key used for encryption or decryption belongs to the message's expected recipient. If a public key is insecurely distributed, it is possible that a malicious agent could intercept it and replace it with the malicious agent's public key (the man-in-the-middle attack). To prevent this and other attacks, PKI provides a means for secure transfer of public keys by linking an identity and that identity's public key in a secure certificate.

While a certificate binds a public key to a subject to ensure the public key's security, it does not guarantee that the security of the associated private key has not been breached. A secure system is dependent upon private keys being kept secret, by protecting them from malicious physical and virtual access.

Certificates

A certificate is an electronic identity document. To create a certificate for a subject, a trusted third party (known as the Certification Authority) verifies the subject's identity, binds a public key to that identity, and digitally signs the certificate. A person receiving a copy of the certificate can verify the Certification Authority's digital signature and be sure that the public key is owned by the identity in it.

The router can generate a self-signed certificate but this should be used with an SSL enabled HTTP server only or where third party trust is not required.

X.509 Certificates

X.509 specifies a format for certificates. Almost all certificates use the X.509 version 3 format, described in RFC 2459, "*Internet X.509 Public Key Infrastructure Certificate and CRL Profile*". This is the format supported by the router.

An X.509 v3 certificate consists of:

- A serial number that distinguishes the certificate from others issued by the same issuer. This serial number identifies the certificate in a Certificate Revocation List if necessary.
- Details about the owner's identity such as name, company, and address.
- The owner's public key, and information about the algorithm with which it was produced.

- Details about the identity of the organisation that issued the certificate.
- The issuer's digital signature and the algorithm that produced it.
- The period when the certificate is valid.
- Optional information such as the type of application for which the certificate is intended.

The issuing organisation's digital signature is included in order to authenticate the certificate. If it is tampered with during transmission, it will be detected.

Elements of a Public Key Infrastructure

A Public Key Infrastructure is a set of applications that manage the creation, retrieval, validation, and storage of certificates. A PKI consists of the following key elements:

- At least one Certification Authority (CA), which issues and revokes certificates.
- At least one publicly accessible repository, which stores certificates and Certificate Revocation Lists (CRL).
- At least one End Entity (EE), which retrieves certificates from the repository, validates them, and uses them.

End Entities (EE)

End Entities own public keys and may use them for encryption and digital signing. An entity that uses its private key to digitally sign certificates is not considered to be an End Entity, but is a Certification Authority.

The router acts as an End Entity.

Certification Authorities

A Certification Authority is an entity that issues, updates, revokes, and otherwise manages public keys and their certificates. A CA receives requests for certification, validates the requester's identity according to the CA's requirements, and issues the certificate, signed with one of the CA's keys. CAs may also perform the functions of End Entities in that they may use other CAs' certificates for message encryption and verification of digital signatures.

An organisation may own a Certification Authority and issue certificates for use within its own networks. Its certificates may also be accepted by another network after an exchange of certificates has validated it from the other network's viewpoint. Alternatively, an outside CA may be used. The router can interact with the CA by sending it requests for certification, regardless of whether that CA is part of the organisation.

The usefulness of certificates depends on trust. Users must be able to trust the issuing CA to verify identities reliably. The level of verification required in a given situation depends on the organisation's security needs.

Repositories

Certificates are stored in repositories and can be accessed by:

- File Transfer Protocol (FTP)
- Trivial File Transfer Protocol (TFTP)
- HyperText Transfer Protocol (HTTP)
- Lightweight Directory Access Protocol (LDAP)

An LDAP repository can be accessed via LDAP. LDAP is a lightweight network-layer protocol for accessing X.500-like directories. It runs over TCP and uses a client/server model.

For more information on TFTP and LDAP, see "Loading Files onto the Router" on page 1-48 of Chapter 1, Operation.

Certificate Validation

To validate a certificate, the End Entity verifies the signature in the certificate by using the public key of the CA who issued the certificate. If the End Entity does not hold the issuer's certificate (which contains the issuer's public key), it downloads the certificate from a repository and tries to validate it. This process of accessing CA certificates and attempting to validate them can be followed as far as necessary up a certificate chain.

CA Hierarchies and Certificate Chains

It may not be practical for every individual certificate in an organisation to be signed by one Certification Authority. A certification hierarchy can be formed in which one CA (for example, national headquarters) is declared the root CA. This CA issues certificates to the next level down in the hierarchy (for example, regional headquarters), who become subordinate CAs and issue certificates to the next level down, and so on. A hierarchy can have as many levels as necessary.

Certificate hierarchies allow validation of certificates through certificate chains and cross-certification. If a router X, which holds a certificate signed by CA X, wants to communicate securely with a router Y, which holds a certificate signed by CA Y, there are two ways that the routers can validate each other's certificates. Cross-certification occurs when router X validates router Y's CA (CA Y) by obtaining a certificate for router Y's CA that has been issued by its own CA (CA X) . A certificate chain is formed when both CA X and CA Y hold a certificate signed by a root CA Z, which the routers have verified out of band. Router X can validate router Y's certificate (and vice versa) by following the chain up to CA Z.

Root CA Certificates

A root CA must sign its own certificate. The root CA is the most critical link in the certification chain because the validity of all certificates issued by a CA in the hierarchy depends on the validity of the root CA. Therefore, every device that uses a certificate from the root CA must verify it out of band.

Out of band verification involves both the owner of a certificate and the user who wants to verify it generating a one-way hash (a fingerprint) of the certificate. These two hashes must then be compared using at least one non-network-based communication method. Examples of suitable communication methods are post, telephone, fax, or transfer by hand from a storage device such as a smartcard or floppy disk. If the two hashes are the same, the certificate can be considered valid.

Certificate Revocation Lists (CRLs)

Every CA must keep a publicly accessible list of its certificates that have been revoked. A certificate may become invalid because:

- some of the details in it change (for example, the address)
- the relationship between the Certification Authority and the subject changes (for example, an employee leaves a company)
- the associated private key is compromised

PKI on the Router

The following standards are supported by the router:

- draft-ietf-pkix-roadmap-05, "*PKIX Roadmap*"
- RFC 1779, "*A String Representation of Distinguished Names*"
- RFC 2459, "*PKIX Certificate and CRL Profile*"
- RFC 2511, "*PKIX Certificate Request Message Format*"
- PKCS #10 v1.7, "*Certification Request Syntax Standard*"
- RFC 2559, "*Operational Protocols: LDAPv2*"
- RFC 2587, "*LDAPv2 Schema*"
- draft-ietf-pkix-cmp-transport-protocols-01, "*Transport Protocols for CMP*"
- RFC 2585, "*Operational Protocols: FTP and HTTP*"
  (HTTP only)
- RFC 2510, "*PKIX Certificate Management Protocols*"
  (a subset of messages and functionality)

Certificate Retrieval and Storage

Certificates are stored by CAs in publicly accessible repositories for retrieval by end entities. Repositories used in PKI are commonly accessed with the following protocols:

- Hypertext Transfer Protocol (HTTP)
- File Transfer Protocol (FTP)
- Lightweight Directory Access Protocol (LDAP)

The router supports retrieval of certificates from LDAP and HTTP repositories and via TFTP (*Trivial File Transfer Protocol*).

Before the router can use a certificate, it must be retrieved and added to the router's Certificate Database, which is stored in RAM memory. The router tries to validate the certificate and if successful, the certificate's public key is available for use.

To add a certificate to the certificate database, use the **add pki certificate** command on page 46-15. This command retrieves the certificate from one of the following locations:

- a file in the router's filesystem
- an LDAP repository, using the LDAP protocol
- an HTTP repository, using the HTTP protocol

To load a file containing a certificate into the router's flash file system using HTTP, LDAP or TFTP, use the **load** command, as described in "Loading Files onto the Router" on page 1-48 of Chapter 1, Operation. A certificate file has a .CER file extension. Storing certificates in flash memory may be particularly useful for administrators of small networks who may not want to operate a repository.

Certificates in the router's flash memory cannot be accessed by the ISAKMP module. The certificates must first be placed in the Certificate Database with the **add pki certificate**. This command can be included in the start-up configuration script to add certificates to the database when the router is restarted.

Certificates are retrieved from an LDAP repository and added to the Certificate Database without a user entering a command in the following cases:

- as required by ISAKMP to authenticate a message.
- as required by the certificate validation process.

These processes require that the router be configured with information about one or more LDAP repositories (see "Automatic Retrieval from LDAP repositories" on page 46-8).

To create a self-signed certificate for router management, use the **create pki certificate** command on page 46-18. This command creates a certificate that is suitable only for secure router management via the GUI. A pop-up message appears in the browser window warning that the certificate is not issued by a trusted authority.

To delete one or all existing certificates from the Certificate Database, use the **delete pki certificate** command on page 46-21.

To see details about one or all certificates in the Certificate Database, use the **show pki certificate** command on page 46-36. The router assigns names to certificates that it automatically retrieves as part of the certificate validation or message authentication process. The subject of the certificate is also displayed for these certificates.

Once a certificate has been downloaded and added to the Certificate Database, its type and the degree to which it is trusted can be changed with the **set pki certificate** command on page 46-27. A certificate should not be set to "trusted" unless it is a root CA certificate that has been verified out of band (see "Root CA Certificates" on page 46-5).

Certificate Validation

Before the public key contained in a certificate can be used, the certificate must be trusted. To gain trust, the router checks, among other things:

- that the time period that the certificate is valid includes the current time
- that the certificate has not been revoked (see "Certificate Revocation Lists" on page 46-8)
- that the certificate was signed by a trusted CA.

In the simplest case, the router already has a valid certificate from its CA that contains the public key used to sign the certificate. In this case, the certificate is easily verified.

Root CA Certificates

Root CA certificates are verified out of band by comparing the certificate's *fingerprint* (the encrypted one-way hash with which the issuing CA signs the certificate) with the fingerprint which the CA has supplied by a non-network-based method. To view a certificate's fingerprint, use the **show pki certificate** command on page 46-36.

To manually set a verified root certificate to "trusted", use the **trusted** parameter in the **set pki certificate** command on page 46-27. This indicates that it is unnecessary for PKI to validate this certificate and stops the validation process up the certificate chain. All certificates **below** this one in the chain are validated.

Automatic Retrieval from LDAP repositories

If the router does not have all the certificates it needs to validate a certificate in a certificate chain, it automatically tries to retrieve missing ones from LDAP repositories that the router is configured to use. To add information about an LDAP repository, use the **add pki ldaprepository** command on page 46-17.

To change the access details for an LDAP repository, use the **set pki ldaprepository** command on page 46-28.

To delete information about one or all existing LDAP repositories, use the **delete pki ldaprepository** command on page 46-22.

To see information about the configured LDAP repositories, use the **show pki ldaprepository** command on page 46-45.

Certificate Revocation Lists

CAs maintain publicly accessible Certificate Revocation Lists (CRLs), which are regularly downloaded by the router and checked as part of the process of validating a certificate.

The CRL Database

The CRL database contains CRLs that have been received by the router and information on how to update them.

CRLs are retrieved regularly by the PKI module from HTTP or LDAP repositories or from a local file. To add a CRL to the CRL database, use the **add pki crl** command on page 46-16.

Users can manually download a CRL into the router's flash memory by using the **load** command on page 1-96 of Chapter 1, Operation. A CRL file has a .CRL file extension. CRLs change often and need to be retrieved regularly from the repository.

To change the access details for a CRL, use the **set pki crl** command on page 46-27.

To delete one or all of the CRLs in the database, use the **delete pki crl** command on page 46-22.

To see details of one or all of the CRLs in the database, use the **show pki crl** command on page 46-39.

Requesting a Certificate

A new key pair can be created using the **create enco key** command on page 25-20 of Chapter 25, Compression and Encryption Services. The user can request a certificate from a CA for the new (or an existing) key pair. This request is called an enrollment request.

Enrollment requests can be sent to the CA by one of the following methods:

- manually
- using the Certificate Management Protocol (CMP)

To create a manual enrollment request, set **protocol=manua**l in the **create pki enrollmentrequest** command on page 46-19.

Manual enrollment requests may be in PKCS #10 format, defined in "*PKCS #10 v1.7: Certification Request Syntax Standard*", or the PKIX format, defined in RFC 2511, "*PKIX Certificate Request Message Format*". A Certificate Signing Request file is created and stored in the routers file system, named `<name-of-enrollment-request>.csr`. This file can be manually retrieved from the router and sent to a CA for enrollment.

The Certificate Management Protocol is described in RFC 2510 "*PKIX Certificate Management Protocols (CMP)*". CMP enrollment requests are possible with CAs that support CMP and must be of type PKIX. The enrollment request is transported directly to the CA over TCP.

To create a CMP enrollment request, set **protocol=cmp** in the **create pki enrollmentrequest** command on page 46-19 command:

To remove one or all current enrollment requests, use the **destroy pki enrollmentrequest** command on page 46-23.

To view details of one or all current enrollment requests, use the **show pki enrollmentrequest** command on page 46-42.

Updating a Key Pair

The public key for a certificate can be updated without re-enrollment, if the current certificate has not been revoked. The update request must use the Certificate Management Protocol (CMP) rather than manual enrollment, and therefore must be in PKIX format.

To create a key update request, use the **create pki keyupdaterequest** command on page 46-20.

To remove one or all of the current key update requests, use the **destroy pki keyupdaterequest** command on page 46-23.

To view details of one or all current enrollment requests, use the **show pki keyupdaterequest** command on page 46-43.

Global PKI parameters

Global PKI parameters include:

- the number of certificates that can be stored on the router, after which adding a new certificate removes the oldest dynamically-added certificate
- the number of times the router tries to resend an unsuccessful CMP request and the period between retries
- the frequency with which the router updates CRLs
- an alternative name for the router’s certificates’ subject

Use the **set pki** command on page 46-26 to modify these parameters.

To return parameters to their factory defaults, use the **purge pki** command on page 46-25.

To see the current settings for parameters, use the **show pki** command on page 46-29.

To see counters for certificates, CRLs, enrollment requests, and key update requests, use **counters** option in the **show pki** command on page 46-29.

For debugging purposes, some or all of the PKI data can be displayed on the terminal as it is received. To enable debugging, use the **enable pki debug** command on page 46-24.

To disable debugging, use the **disable pki debug** command on page 46-24. See Table 46-1 on page 46-24 for an explanation of the debugging options.

Configuration Examples

The examples in this section show how to configure PKI on a router.

- **Manual Enrollment**

  If your CA does not support CMP for certificate enrollment, you must manually create a Certificate Signing Request (CSR) for the router's public/private key-pair and use an out-of-band method to send it to the CA. The CA typically requires the CSR be attached to an email, pasted into a web page, or couriered.

- **Automatic Enrollment with CMP**

  If your CA supports CMP, you can configure the router to automatically send an enrollment request to the CA, and to retrieve certificates and CRLs from the CA's LDAP repository on the fly.

Manual Enrollment

This example shows how to:

- Load the CA's certificate on to the router and manually verify it.
- Create a public/private key-pair for the router.
- Generate a CSR and submit it to the CA.
- Load the newly generated router's certificate onto the router.
- Load the CA's Certificate Revocation List (CRL) onto the router.

Obtaining a certificate for the router

1. **Obtain preliminary details from your CA.**

   Determine the following:

   - how to retrieve and verify the CA's certificate.
   - the router's distinguished name, if the CA requires a particular name.
   - the CSR type (PCKS10 or PKIX).
   - the CSR format (DER, PEM or BASE64).
   - how to send the CSR to the CA.
   - how to retrieve the router's certificate once it has been generated by the CA.
   - how to retrieve the CA's CRL (if it is available).

2. **Load the CA's certificate on to the router and add it to the certificate database.**

   To load the CA's certificate named `ca_certificate.cer` from the TFTP server at 192.168.0.100, use the command:

   ```
   load file=ca_certificate.cer server=192.168.0.100
   ```

   Note that the default METHOD is TFTP.

   To add the certificate to the Certificate Database, use the command:

   ```
   add pki certificate=ca_certificate
       location=ca_certificate.cer type=ca
   ```

3. **Verify the CA's certificate and set it to "trusted".**

   The most common way of verifying the CA's certificate is to verbally compare the certificate's fingerprint using an out-of-band method with the CA administrator (for example, over the phone).

   To read the certificate's fingerprint, use the command:

   ```
   show pki certificate=ca_certificate
   ```

   To set the verified certificate to trusted, use the command:

   ```
   set pki certificate=ca_certificate trusted=true
   ```

4. **Set the router's distinguished name.**

   To set the routers distinguished name to "cn=router1,o=my_company,c=us", use the command:

   ```
   set sys distinguishedname="cn=router1,o=my_company,c=us"
   ```

5. **Create the router's encryption key-pair.**

   To create a 1024-bit RSA key-pair for the router, use the command:

   ```
   create enco key=1 type=rsa length=1024
   ```

6. **Create a Certificate Signing Request (CSR).**

   To create a PEM-encoded Certificate Signing Request of type PKSC#10, use the command:

   ```
   create pki enrollmentrequest=my_request keypair=1
      protocol=manual type=pkcs10 format=pem
   ```

7. **Upload the CSR and send to your CA.**

   To upload the CSR file to the TFTP server at 192.168.0.100, use the command:

   ```
   upload file=my_request.csr server=192.168.0.100
   ```

   Send `my_request.csr` to your CA using the method required by the CA, as determined in Step 1.

8. **Load the router's certificate on to the router and add it to the certificate database.**

   To load the certificate into flash memory on the router from the TFTP server at 192.168.0.100, use the command:

   ```
   load file=my_certificate.cer server=192.168.0.100
   ```

   To add the certificate to the Certificate Database, use the command:

   ```
   add pki certificate=my_certificate
      location=my_certificate.cer type=self
   ```

Maintaining the router's CRL database

1. **Load the CA's CRL on to the router and add it to the CRL Database.**

   To load the CA's CRL from the TFTP server at 192.168.0.100, use the command:

   ```
   load file=ca_crl.crl dest=flash server=192.168.0.100
   ```

   To add the CRL to the CRL Database, use the command:

   ```
   add pki crl=ca_crl location=ca_crl.crl
   ```

   Every time the CA updates its CRL, you must manually retrieve and load the new CRL.

Automatic Enrollment with CMP

This example shows how to:

- load the CA's certificate onto the router and manually verify it.
- create a public/private key-pair for the router.
- enrol with the CA using CMP.
- load the router's newly-generated certificate on to the router.
- set up the router to periodically download the CA's Certificate Revocation List (CRL).
- set up the router to automatically retrieve certificates as required from an LDAP repository.

Obtaining a certificate for the router

1. **Obtain preliminary details from your CA.**

   Determine the following:

   - IP address or domain name for the CA's LDAP repository.
   - Distinguished Name of the CA.
   - router's distinguished name, if the CA requires a particular name.
   - IP address or domain name for the CA's CMP server.
   - secret value and reference number from the CA administrator.

2. **Add the CA's certificate to the router's certificate database.**

   To load the certificate of a CA with the distinguished name "o=myCA,c=us" from the LDAP repository at 192.168.0.100, use the command:

   ```
   load method=ldap server=192.168.0.100
      baseobject="o=myca,c=us" dstfile=ca_certificate.cer
      attribute=cacertifcate
   ```

   To add the certificate to the Certificate Database, use the command:

   ```
   add pki certificate=ca_certificate
      location=ca_certificate.cer type=ca
   ```

3. **Verify the CA's certificate and set it to "trusted".**

   The most common way of verifying the CA's certificate is to verbally compare the certificate's fingerprint using an out-of-band method with the CA administrator (for example, over the phone).

   To read the certificate's fingerprint, use the command:

   ```
   show pki certificate=ca_certificate
   ```

   To set the verified certificate to trusted, use the command:

   ```
   set pki certificate=ca_certificate trusted=true
   ```

4. **Set the router's distinguished name.**

   To set the router's distinguished name to "cn=router1,o=myCA,c=us", use the command:

   ```
   set sys distinguishedname="cn=router1,o=myca,c=us"
   ```

5. **Create the router's encryption key-pair.**

   To create a 1024-bit RSA key-pair for the router, use the command:

```
create enco key=1 type=rsa length=1024
```

6. **Enrol with the CA using CMP.**

   To enrol with the CA using CMP, use the command:

```
create pki enrollmentrequest=my_ca location=192.168.0.101
   protocol=cmp keyid=1 secr=secretvalue ref=12345678
```

   where:

   - 192.168.0.101 is the IP Address of the CA's CMP server, as determined in Step 1.
   - *secretvalue* is the shared secret value obtained out-of-band from the CA operator, as determined in Step 1.
   - 12345678 is the reference number that identifies a new user to the CA, and is obtained with the shared secret from the CA operator, as determined in Step 1.

   The new certificate is published to the CA's LDAP repository. It must be loaded into the router's Certificate Database before it can be used.

7. **Load the router's certificate on to the router and add it to the certificate database.**

   To load the router's certificate from the LDAP repository at 192.168.0.100, use the command:

```
load method=ldap server=192.168.0.100
   baseobject="cn=router1,o=myca,c=us"
   dstfile=my_certificate.cer
```

   To add the certificate to the certificate database, use the command:

```
add pki certificate=my_certificate
   location=my_certificate.cer type=self
```

Maintaining the Router's Certificate and CRL Database

1. **Set up the router to periodically download the CA's Certificate Revocation List (CRL).**

   To add the CA's CRL to the CRL Database, use the command:

```
add pki crl=ca_crl
   location="ldap://192.168.0.100/o=myca,c=us"
```

2. **Set up the router to automatically retrieve certificates as required from an LDAP repository.**

   To add the LDAP repository at 192.168.0.100 to the router, use the command:

```
add pki ldaprepository=ca_rep
   location="ldap://192.168.0.100/"
```

Command Reference

This section describes the commands available to configure and manage PKI functionality. All the commands in this section except the SHOW commands require Security Officer status.

The shortest valid command is denoted by capital letters in the Syntax section. See "Conventions" on page xcv of Preface in the front of this manual for details of the conventions used to describe command syntax. See Appendix A, Messages for a complete list of messages and their meanings.

add pki certificate

Syntax

```
ADD PKI CERtificate=name LOCation={url|localfile}
    [PASSword=password] [PROXYAddress={hostname|ipadd}]
    [PROXYPort=port] [TRUsted={TRue|FALse|YES|NO|ON|OFF}]
    [TYpe={CA|SElf|ENDentity|EE}] [USERname=username]
```

where:

- *name* is a character string 1 to 24 characters long. Valid characters are any printable characters. If the name contains spaces, it must be in double quotes. Wildcards are not allowed.
- *url* is either an LDAP URL, as defined in "LDAP URLs" on page 1-51 of Chapter 1, Operation, or an HTTP URL.
- *localfile* is a filename with the file extension `.cer`.
- *password* is a character string 1 to 64 characters long. Valid characters are any printable characters. If the password contains spaces, it must be in double quotes. Wildcards are not allowed.
- *hostname* is a character string 1 to 40 characters long. Valid characters are any printable characters.
- *ipadd* is an IP address in dotted decimal notation.
- *port* is an integer between 1 and 65535.
- *username* is a character string 1 to 64 characters long. Valid characters are any printable characters. If the username contains spaces, it must be in double quotes. Wildcards are not allowed.

Description

This command adds a certificate to the router's Certificate Database from either a file in the router's file system or a remote repository.

The LOCATION parameter specifies from where the certificate is retrieved. If a URL is specified, the certificate is retrieved from a remote repository using the appropriate method for the URL (LDAP or HTTP). If a local file is specified, the certificate is retrieved from the router's file system. The local file must be a valid filename with a .CER file extension.

The PASSWORD parameter specifies the password for the remote repository.

The PROXYADDRESS parameter specifies the address of the optional HTTP proxy to use. The PROXYADDRESS parameter is valid when the LOCATION is an HTTP URL.

The PROXYPORT parameter specifies the port to use on the optional HTTP proxy.

The TRUSTED parameter specifies whether the certificate is automatically trusted. If TRUE, YES, or ON, the certificate is trusted unless it is badly formatted, out-of-date, or revoked. If FALSE, NO, or OFF, the certificate is not automatically trusted, only if another trusted certificate is found whose public key has signed this certificate. The default is FALSE. Only self-signed root CA certificates are typically set to be automatically trusted, and only after the user has checked the certificate’s fingerprint and other details by using the **show pki certificate** command on page 46-36.

The TYPE parameter specifies what type of certificate is being added. If CA is specified, the router tags this certificate as a CA certificate. If SELF, the router tags the certificate as its own. If ENDENTITY or EE, the router tags the certificate to indicate that it belongs to another end entity. The default is ENDENTITY.

The USERNAME parameter specifies the username for the remote repository.

Examples

To load a certificate called "bobscertificate" for an entity with the distinguished name "cn=router_city1, o=company1, c=us", from a remote LDAP repository, use the command:

```
add pki cer=bobscertificate
    loc="ldap://ca.trustworthy.com/cn=router_city1,
    o=company1, c=us"
```

Related Commands

delete pki certificate
set pki certificate
show pki certificate

add pki crl

Syntax

```
ADD PKI CRL=name LOCation={url|localfile}
    [PASSword=password] [PROXYAddress={hostname|ipadd}]
    [PROXYPort=1..65535] [USERname=username]
```

where:

- *name* is a character string 1 to 24 characters long. Valid characters are any printable characters. If the name contains spaces, it must be in double quotes. Wildcards are not allowed.
- *url* is either an LDAP URL, as defined in “LDAP URLs” on page 1-51 of Chapter 1, Operation, or an HTTP URL.
- *localfile* is a filename with the file extension `.crl`.
- *password* is a character string 1 to 64 characters long. Valid characters are any printable characters. If the password contains spaces, it must be in double quotes. Wildcards are not allowed.
- *hostname* is a character string 1 to 40 characters long.

- *ipadd* is an IP address in dotted decimal notation.
- *port* is an integer between 1 and 65535.
- *username* is a character string 1 to 64 characters long. Valid characters are any printable characters. If the username contains spaces, it must be in double quotes. Wildcards are not allowed.

Description This command adds a Certificate Revocation List request. A CRL is retrieved periodically from the specified location and added to the CRL Database. The frequency of download can be set with the **set pki** command on page 46-26, and defaults to 24 hours.

The LOCATION parameter specifies from where the certificate is retrieved. If a URL is specified, the certificate is retrieved from a remote repository using the appropriate method for the URL (LDAP or HTTP). If a local file is specified, the certificate is retrieved from the router's file system. The local file must be a valid filename with a .CRL file extension.

The PASSWORD parameter specifies the password for the remote repository.

The PROXYADDRESS parameter specifies the address of the optional HTTP proxy server to use. Either the IP address or the fully qualified domain name of the proxy server may be specified. If a domain name is specified the router performs a DNS lookup to resolve the name.

The PROXYPORT parameter specifies the port to use on the optional HTTP proxy. The PROXYPORT parameter is valid if PROXYADDRESS is specified. The default is 80.

The USERNAME parameter specifies the username for the remote repository.

Examples To add a CRL from a remote LDAP repository, use the command:

```
add pki crl=trustworthyscrl
    loc="ldap://ca.trustworthy.com/cn=trustworthycrl,
    o=trustworthy, c=us"
```

Related Commands **delete pki crl**
show pki crl

add pki ldaprepository

Syntax

```
ADD PKI LDaprepository=name LOCation=ldap-url
    [PASSword=password] [USERname=username]
```

where:

- *name* is a character string 1 to 24 characters long. Valid characters are any printable characters. If the name contains spaces, it must be in double quotes. Wildcards are not allowed.
- *ldap-url* is a valid LDAP URL, as defined in "LDAP URLs" on page 1-51 of Chapter 1, Operation, except that the URL must not contain a distinguished name.

- *password* is a character string 1 to 64 characters long. Valid characters are any printable characters. If the password contains spaces, it must be in double quotes. Wildcards are not allowed.
- *username* is a character string 1 to 64 characters long. Valid characters are any printable characters. If the username contains spaces, it must be in double quotes. Wildcards are not allowed.

Description This command adds information about a remote LDAP repository. The repository can then be used by the certificate validation process of the PKI operational protocols to dynamically retrieve certificates.

The LOCATION parameter specifies the address of the LDAP repository.

The PASSWORD parameter specifies the password for the remote repository.

The USERNAME parameter specifies the username for the remote repository.

Examples To add an LDAP repository, use the command:

```
add pki ld=trustworthy_repository
    loc=ldap://repository.trustworthy.com
```

Related Commands **delete pki ldaprepository**
set pki ldaprepository
show pki ldaprepository

create pki certificate

Syntax

```
CREate PKI CERtificate=name KEYpair=id SERialnumber=number
    [FORMat=DER|PEM|BASE64] [SUbject=dist-name]
```

where:

- *name* is a character string 1 to 8 characters long. Valid characters are uppercase and lowercase letters, digits (0-9), the underscore character ("_"), the hyphen (-), any printable characters, any alphanumeric characters. If name contains spaces, it must be in double quotes. Wildcard characters are not allowed.
- *id* is a decimal number from 0 to 65535.
- *number* is a decimal number from 0 to 4294967295.
- *dist-name* is an X.500 distinguished name, as described in *"Distinguished Names (DN)" on page 1-50 of Chapter 1, Operation*.

Description This command creates a self-signed certificate using an ENCO private RSA key and the router's distinguished name. The router's distinguished name, set with the **set system distinguishedname** command on page 1-123 of Chapter 1, Operation, is inserted in the issuer field of the certificate. This certificate is suitable for an SSL enabled HTTP server or where third party trust is not required.

⚠ To ensure router security, use an external CA to generate certificates for VPNs or when the certificate will be used for more than router management.

The KEYPAIR parameter specifies the ENCO keyId of the private RSA key used to sign the certificate.

The SERIALNUMBER parameter specifies the number to be inserted into the serial number field of the certificate.

The SUBJECT parameter specifies the distinguished name to be inserted in the subject field of the certificate. If this parameter is not specified, the system distinguished name is used.

The FORMAT parameter specifies the type of encoding the certificate uses. Both PEM and BASE64 are ASCII encoded and let the certificate be displayed once it has been generated. DER encoding is binary and the certificate cannot be displayed. The default is DER.

Examples To create a self-signed certificate, use the command:

```
cre pki cer=mycert key=1 ser=1234
```

Related Commands **add pki certificate**
show pki certificate

create pki enrollmentrequest

Syntax

```
CREate PKI ENRollmentrequest=name KEYpair=key-id
   [FORMat={DER|PEM|BASE64}] [LOCation={domainname|ipadd}]
   [PROTocol={CMP|MANual}] [REFerencenumber=refnum]
   [SECretvalue=secretvalue] [TYpe={PKCS10|PKIX}]
```

where:

- *name* is a character string 1 to 8 characters long. Valid characters are any printable characters. If the name contains spaces, it must be in double quotes. Wildcards are not allowed.
- *key-id* is a decimal number from 0 to 65535.
- *domainname* is a character string 1-64 characters long.
- *ipadd* is an IP address in dotted decimal notation.
- *refnum* is a character string 1-24 characters long.
- *secretvalue* is a character string 1-24 characters long.

Description This command creates a certificate enrollment request. This operation is required when the user first creates a new key pair and wishes to get it signed by a CA. The enrollment request can be transmitted to the CA automatically using CMP, or manually uploaded using the **upload** command on page 1-186 of Chapter 1, Operation.

The NAME parameter specifies a name to distinguish the enrollment request from any other current requests.

The KEYPAIR parameter specifies the ENCO key identification number for the new certificate.

The FORMAT parameter specifies the type of encoding format for the request. DER specifies that the enrollment request should be written straight to the binary file. PEM specifies that the enrollment request should be encoded using the "Privacy Enhanced Mail" format. BASE64 specifies that the enrollment request should be BASE64 encoded before it is written to the file. The default is DER. This parameter is valid for manual enrollment only.

The LOCATION parameter specifies the address of the CA. This parameter is only valid for the CMP protocol.

The PROTOCOL parameter specifies the PKI management protocol to use for the enrollment request. If CMP, the enrollment request is automatically sent using the PKI Certificate Management Protocol. If MANUAL, the enrollment request is written to a file in the router's file system. The file is named `<name-of-enrollment-request>.csr`. This file can be retrieved from the router later and sent to the CA to process. The default is CMP.

The REFERENCENUMBER parameter allows the user to enter a reference number if one has been provided by the CA. This parameter is valid for the CMP protocol.

The SECRETVALUE parameter specifies the shared secret for Proof of Possession (of private key) transactions that the CA may require. This parameter is valid for the CMP protocol.

The TYPE parameter specifies the type of request. If PKIX is specified, the request is encoded to RFC 2511, "*PKIX Certificate Request Message Format*" format. If PKCS10 is specified, the request is formatted according to PKCS #10. Note that for CMP, only PKIX can be used. The default is PKIX.

Examples To create an enrollment request, use the command:

```
cre pki enr=mycert key=1 prot=cmp loc=10.1.1.12
```

Related Commands

destroy pki enrollmentrequest
show pki enrollmentrequest

create pki keyupdaterequest

Syntax

```
CREate PKI KEYUpdaterequest=name CERtificate=name
   KEYpair=key-id [LOCation={domainname|ipadd}]
```

where:

- *name* is a character string 1 to 8 characters long. Valid characters are any printable characters. If the name contains spaces, it must be in double quotes. Wildcards are not allowed.
- *key-id* is a decimal number from 0 to 65535.
- *domainname* is a character string 1-64 characters long.
- *ipadd* is an IP address in dotted decimal notation.

Description This command creates a certificate key update request. This operation is required when a user wants to update the key pair for the router's own certificate. The user must have previously enrolled and obtained a certificate from the CA. This function is available when the CA supports CMP and when the current certificate is still valid.

The CERTIFICATE parameter specifies the certificate for which the keys are to be updated.

The KEYPAIR parameter specifies the ENCO key identification number of the new key pair.

The LOCATION parameter specifies the address of the CA.

Examples To create a key update request, use the command:

```
cre pki keyu=mycert cer=mycert key=2 loc=10.1.1.12
```

Related Commands **destroy pki keyupdaterequest**
show pki keyupdaterequest

delete pki certificate

Syntax DELete PKI CERtificate={*name*|ALL}

where *name* is a character string from 1 to 24 characters long. Valid characters are any printable characters. If the name contains spaces, it must be in double quotes. Wildcards are not allowed.

Description This command deletes one or all of the certificates stored in the router's certificate database.

The CERTIFICATE parameter specifies the name of the certificate to be deleted. If ALL is specified every certificate in the router's certificate database is deleted.

Examples To delete a certificate, use the command:

```
del pki cer=bobs_old_certificate
```

Related Commands **add pki certificate**
set pki certificate
show pki certificate

delete pki crl

Syntax

```
DELete PKI CRL={name|ALL}
```

where *name* is a character string from 1 to 24 characters long. Valid characters are any printable characters. If the name contains spaces, it must be in double quotes. Wildcards are not allowed.

Description

This command deletes one or all of the CRLs stored in the router's CRL database.

The CRL parameter specifies the name of the CRL to be deleted. If ALL is specified every CRL in the router's CRL database is deleted.

Examples

To delete all CRLs in the CRL database, use the command:

```
del pki crl=all
```

Related Commands

add pki crl
set pki crl
show pki crl

delete pki ldaprepository

Syntax

```
DELete PKI LDaprepository={name|ALL}
```

where *name* is a character string from 1 to 24 characters long. Valid characters are any printable characters. If the name contains spaces, it must be in double quotes. Wildcards are not allowed.

Description

This command deletes the information about one or all of the LDAP repositories configured for use by the router.

The LDAPREPOSITORY parameter specifies the name of the LDAP repository information to be deleted. If ALL is specified every repository is deleted.

Examples

To delete all LDAP repositories, use the command:

```
del pki ld=all
```

Related Commands

add pki ldaprepository
set pki ldaprepository
show pki ldaprepository

destroy pki enrollmentrequest

Syntax

```
DESTroy PKI ENRollmentrequest={name|ALL}
```

where *name* is a character string from 1 to 24 characters long. Valid characters are any printable characters. If the name contains spaces, it must be in double quotes. Wildcards are not allowed.

Description

This command destroys one or all of the current enrollment requests.

The ENROLLMENTREQUEST parameter specifies the name of the enrollment request to be destroyed. If ALL is specified every request is destroyed.

Examples

To destroy all current enrollment requests, use the command:

```
dest pki enr=all
```

Related Commands

create pki enrollmentrequest
show pki enrollmentrequest

destroy pki keyupdaterequest

Syntax

```
DESTroy PKI KEYUpdaterequest={name|ALL}
```

where *name* is a character string from 1 to 24 characters long. Valid characters are any printable characters. If the name contains spaces, it must be in double quotes. Wildcards are not allowed.

Description

This command destroys one or all of the current key update requests.

The KEYUPDATEREQUEST parameter specifies the name of the enrollment request to be destroyed. If ALL is specified every request is destroyed.

Examples

To destroy all current key update requests, use the command:

```
dest pki keyu=all
```

Related Commands

create pki keyupdaterequest
show pki keyupdaterequest

disable pki debug

Syntax

```
DISable PKI DEBug={ALL|CMDTRACE|CRTDBTRACE|CRTRUTRACE|
    CRTVGSTATE|CRTVGTRACE|CRTVUSTATE|CRTVUTRACE|MPPACKET|
    MPPKT|MPSTATE|MPTRACE|OPHTTPTRACE|OPLDAPTRACE|OPPACKET|
    OPPKT|PACKET|PKT|STATE|TRACE}[,...]
```

Description

This command disables PKI debugging.

The DEBUG parameter specifies which debugging options are to be disabled. The value of this parameter is a single item or a comma-separated list. The items allowed and the debugging that results from specifying each item are shown in Table 46-1 on page 46-24.

Examples

To turn all PKI module debugging off, use the command:

```
dis pki deb=all
```

Related Commands

enable pki debug

enable pki debug

Syntax

```
ENAble PKI DEBug={ALL|CERTTRACE|CMDTRACE|CRTDBTRACE|
    CRTRUTRACE|CRTVGSTATE|CRTVGTRACE|CRTVUSTATE|CRTVUTRACE|
    MPPACKET|MPPKT|MPSTATE|MPTRACE|OPHTTPTRACE|OPLDAPTRACE|
    OPPACKET|OPPKT|PACKET|PKT|STATE|TRACE}[,...]
```

Description

This command enables PKI debugging.

The DEBUG parameter specifies which debugging options are to be enabled. The value of this parameter is a single item or a comma-separated list. The items allowed and the debugging that results from specifying each item are shown in Table 46-1 on page 46-24.

Table 46-1: Public Key Infrastructure (PKI) debugging options

Option	Description
ALL	All debug options.
CERTTRACE	Certificate unit trace debug.
CMDTRACE	Command handler trace debug.
CRTDBTRACE	Certificate database trace debug.
CRTRUTRACE	Certificate retrieval unit trace debug.
CRTVGSTATE	Certification validation state debug, for the candidate issuer certificate retrieval process.
CRTVGTRACE	Certification validation trace debug, for the candidate issuer certificate retrieval process.
CRTVUSTATE	Certificate validation unit state debug.
CRTVUTRACE	Certificate validation unit trace debug.
MPPACKET or MPPKT	Management protocol packet debug.

Table 46-1: Public Key Infrastructure (PKI) debugging options

Option	Description
MPSTATE	Management protocol state debug.
MPTRACE	Management protocol trace debug.
OPHTTPTRACE	HTTP operation protocol trace debug.
OPLDAPTRACE	LDAP operation protocol trace debug.
OPPACKET or OPPKT	Operational protocol packet debug.
PACKET or PKT	All PKI packet debug options.
STATE	All PKI state debug options.
TRACE	All PKI trace debug options.

Examples To turn on all PKI debugging features, use the command:

```
ena pki deb=all
```

Related Commands **disable pki debug**

purge pki

Syntax `PURge PKI`

Description This command resets the PKI module to the default configuration. Certificates, LDAP repositories, CRLs, enrollment requests, and key update requests are destroyed. Debugging is turned off and parameters modified by the SET PKI command are returned to defaults.

Examples To reset the PKI module, use the command:

```
pur pki
```

Related Commands

show pki
show pki certificate
show pki crl
show pki enrollmentrequest
show pki keyupdaterequest

set pki

Syntax

```
SET PKI [CERtstorelimit=max-certificates]
    [CMPRETRYMax=retries] [CMPRETRYPeriod=seconds]
    [CRLUPdateperiod=hours] [SUbjectaltname={ipadd|name}]
```

where:

- *max-certificates* is a decimal number from 12 to 256.
- *retries* is a decimal number from 0 to 10.
- *seconds* is a decimal number from 10 to 600.
- *hours* is a decimal number from 1 to 168.
- *ipadd* is an IP address in dotted decimal notation.
- *name* is a character string 1 to 64 characters long. Valid characters are any printable characters. If the name contains spaces, it must be in double quotes. Wildcards are not allowed.

Description

This command sets global PKI parameters required for PKI operations.

The CERTSTORELIMIT parameter specifies the maximum number of certificates that can be stored in the router's certificate database. The default is 24. When the maximum number is reached, the router removes the oldest dynamically-added certificates first.

The CMPRETRYMAX parameter specifies the number of times a CMP message is resent to a CA. This parameter is intended to resend the message when the CA is transiently unavailable. The default is 1 retry.

The CMPRETRYPERIOD parameter specifies the length of time between resending CMP messages to a CA when the CA is busy. The default is 30 seconds.

The CRLUPDATEPERIOD parameter specifies the length of time between reloading CRLs in the CRL database. The default is 24 hours.

The SUBJECTALTNAME parameter specifies the name to be added into certificate requests in the subject alternative name extension. The alternative name can be an IP address or a descriptive name such as foo.bar.com.

Example

To specify that the IP address 192.168.1.2 is to be added into certificate requests as the subject alternative name extension, use the command:

```
set pki su=192.168.1.2
```

Related Commands

show pki

set pki certificate

Syntax

```
SET PKI CERtificate=name [TRUsted={TRue|FALse|YES|NO|ON|
    OFF}] [TYpe={CA|SElf|ENDentity|EE}]
```

where *name* is a character string from 1 to 24 characters long. Valid characters are any printable characters. If the name contains spaces, it must be in double quotes. Wildcards are not allowed.

Description

This command sets the level of trust for a certificate loaded into the router's certificate database.

The CERTIFICATE parameter specifies the name of the certificate.

The TRUSTED parameter specifies whether the certificate is automatically trusted. If TRUE, YES, or ON, the certificate is always trusted unless it is badly formatted, out-of-date, or revoked. If FALSE, NO, or OFF, the certificate is not automatically trusted, only if another trusted certificate is found whose public key has signed this certificate. The default is FALSE.

Only self-signed root CA certificates are typically set to be automatically trusted, and only after the user has checked the certificate's fingerprint and other details by using the **show pki certificate** command on page 46-36.

The TYPE parameter specifies what type of certificate is being added. If CA, the router tags this certificate as a CA certificate. If SELF, the router tags the certificate as its own. If ENDENTITY or EE, the router tags the certificate to indicate that it belongs to another end entity. The default is ENDENTITY.

Examples

To set a certificate to "trusted" after manually validating it, use the command.

```
set pki cer=bobscertificate tru=true
```

Related Commands

add pki certificate
delete pki certificate
show pki certificate

set pki crl

Syntax

```
SET PKI CRL=name [LOCation={url|localfile}]
    [PASSword=password] [USERname=username]
```

where:

- *name* is a character string from 1 to 24 characters long. Valid characters are any printable characters. If the name contains spaces, it must be in double quotes. Wildcards are not allowed.
- *url* is either an LDAP URL, as defined in "LDAP URLs" on page 1-51 of Chapter 1, Operation, or an HTTP URL.
- *localfile* is a filename with the file extension `.crl`.

- *password* is a character string from 1 to 64 characters long. Valid characters are any printable characters. If the password contains spaces, it must be in double quotes. Wildcards are not allowed.
- *username* is a character string from 1 to 64 characters long. Valid characters are any printable characters. If the username contains spaces, it must be in double quotes. Wildcards are not allowed.

Description This command sets the parameters for a certificate revocation list request.

The LOCATION parameter specifies where the CRL is loaded from. If a URL is specified, the certificate is loaded from a remote repository using the appropriate method for the URL (LDAP or HTTP). If a local file is specified, the certificate is loaded from the router’s flash filing system.

The USERNAME parameter specifies the username for the remote repository.

The PASSWORD parameter specifies the password for the remote repository.

Examples To set a CRL’s location, use the command:

```
set pki crl=trustworthyscrl
    loc=http://ca.trustworthy.com/mycrl.crl
```

Related Commands **add pki crl**
delete pki crl
show pki crl

set pki ldaprepository

Syntax

```
SET PKI LDaprepository=name [LOCation=ldap-url]
    [PASSword=password] [USERname=username]
```

where:

- *name* is a character string from 1 to 24 characters long. Valid characters are any printable characters. If the name contains spaces, it must be in double quotes. Wildcards are not allowed.
- *ldap-url* is a valid LDAP URL, as defined in “LDAP URLs” on page 1-51 of Chapter 1, Operation, except that the URL must not contain a distinguished name.
- *password* is a character string from 1 to 64 characters long. Valid characters are any printable characters. If the password contains spaces, it must be in double quotes. Wildcards are not allowed.
- *username* is a character string from 1 to 64 characters long. Valid characters are any printable characters. If the username contains spaces, it must be in double quotes. Wildcards are not allowed.

Description This command modifies information about a remote LDAP repository. The certificate validation process of the PKI operational protocols uses the repository to dynamically retrieve certificates.

The LOCATION parameter specifies the address of the LDAP repository.

The PASSWORD parameter specifies the password for the remote repository.

The USERNAME parameter specifies the username for the remote repository.

Examples To set parameters for a current LDAP repository, use the command:

```
set pki ld=trustworthy_repository loc=ldap://
   repository.trustworthy.com
```

Related Commands **add pki ldaprepository**
delete pki ldaprepository
show pki ldaprepository

show pki

Syntax

```
SHow PKI [COUnters]
```

Description This command displays information about the PKI module (Figure 46-1 on page 46-29, Table 46-2 on page 46-29).

If the COUNTERS parameter is specified, counters for the PKI module are displayed (Figure 46-2 on page 46-30, Table 46-3 on page 46-31). All of these counters apply since the router was last restarted.

Figure 46-1: Example output from the **show pki** command

```
PKI Module general information:
  subjectAltName ..........
  CRL update period ....... 24 hours
  CMP retry period ........ 5 seconds
  CMP maximum retries ..... 1
  Max. # of certificates .. 24
  Debug device ............ 16
  Debug types enabled: none
```

Table 46-2: Parameters in the output of the **show pki** command

Parameter	Meaning
subjectAltName	Alternative subject name for this router's certificates.
CRL update period	Period in hours between automatic updates of any Certificate Revocation Lists used by the router.
CMP retry period	Period in seconds between the initial transmission of a CMP message and the first retransmission of the message when no response is received.
CMP maximum retries	Maximum number of times a CMP is retransmitted when no response is received.
Max. # of certificates	Maximum number of certificates that may be stored in the router's certificate database.
Debug device	Device number where PKI debug output is sent.
Debug types enabled	The types of PKI debugging that are enabled. A list of available types and their meanings are in Table 46-1 on page 46-24.

Figure 46-2: Example output from the **show pki counters** command

```
PKI CERTIFICATE UNIT COUNTERS:
 GENERAL COUNTERS:
  checkExtractParseFail             0  checkExtractFieldGetFail          0
  createFingerprintFail             0

 VALIDATION UNIT COUNTERS:
  eventRequestNotFound              0
  startRequest                     13  startRequestEquivReqFnd           0
  stopRequestFail                   0  stopRequestGood                   0
  gncicResultNoCIC                  7
  actionIllegal                     0
  startInitChecksTooDeep            0  startInitChecksBadTime            0
  startInitChecksIssNotCA           0  startInitChcksBadKeyUsge          0
  startInitChcksSSignNotCA          0  startInitChcksSSgnUntrst          0
  startInitChecksRevoked            0
  startValCICCertNotFound           0  startValCICAlreadyTrustd          6
  startValCICAlrdyVldating          5  startValCICStartVURqFail          0
  startVerifyCertNotFound           0  startVerifyCertParseFail          0
  startVerifyCrtFldGetFail          0
  startVerifyCICParseFail           0  startVerifyCICFldGetFail          0
  startVerifyFailImm                0  startVerifyStarted                7
  verifyCbackReqNotFound            0  verifyCallbackVerifyGood          6
  verifyCallbackVerifyFail          1
  createRequest                    13  destroyRequest                   13
  dbAdd                            13  dbRemoveFail                      0
  dbFindFail                        0  dbFindByRequestorFail            13
  dbFindByCertFail                  2

 VALIDATION UNIT GET NEXT CANDIDATE ISSUER CERTIFICATE COUNTERS:
  eventRequestNotFound              0  getNextCICNotFound                0
  remoteRetrievdReqNotFnd           0  actionIllegal                     0
  startGetRemoteFail                7

 RETRIEVAL UNIT COUNTERS:
  startOPStartFail                  7  retrievedRequestNotExist          0
  getFileNotExist                   0  getFileOpenFail                   0
  getFileReadFail                   0  getFileCloseFail                  0

PKI CRL UNIT COUNTERS:
 GENERAL COUNTERS:
  retrievedNotFound                 0  retrievedParseFail                0
  retrievedGetFieldsFail            0  retrievedRUStartFail              0
  timeoutRUStartFail                0
  addAlreadyExists                  0  addRUStartFail                    0
  addGood                           0
  setNotFound                       0  setRUStartFail                    0
  setGood                           0  setNothingSet                     0
  deleteNotFound                    0  deleteGood                        0
  purgeGood                         0  showNotFound                      0
  showListParseFail                 0

 RETRIEVAL UNIT COUNTERS:
  startOPStartFail                  0  retrievedRequestNotExist          0
  getFileNotExist                   0  getFileOpenFail                   0
  getFileReadFail                   0  getFileCloseFail                  0
```

Figure 46-2: Example output from the **show pki counters** command (continued)

```
PKI MANAGEMENT UNIT COUNTERS:
 CMP COUNTERS:
  startedEnrollment                0  startedKeyUpdate                 1
  receivedPKIMessage               1  receivedPKIFIN                   1
  receivedTCPError                 0  receivedMsgVerifyOK              1
  receivedMsgVerifyFail            0  illegalEvent                     0
  requestFailed                    0  requestCompleted                 1
  receivedMessageDiscarded         0  retryEvent                       0
  timeoutEvent                     0  requestQueueError                0

 MANUAL ENROLLMENT COUNTERS:
  started                          0  fileOpenFailed                   0
  failed                           0  completed                        0
  fileOpenOK                       0  fileClosed                       0
  fileWriteFailed                  0  CSRCreationError                 0
```

Table 46-3: Parameters in the output of the **show pki counters** command

Parameter	Meaning
PKI CERTIFICATE UNIT COUNTERS – for events in the PKI certificate unit	
GENERAL COUNTERS	Counters about general events in the PKI certificate unit.
checkExtractParseFail	The number of times a bad certificate format prevented the certificate from being parsed successfully.
checkExtractFieldGetFail	The number of times a bad certificate format prevented a field from being read successfully.
createFingerprintFail	The number of times a certificate fingerprint was not created successfully.
VALIDATION UNIT COUNTERS – for events in the PKI certificate validation unit.	
eventRequestNotFound	The number of times an event was received for a certificate validation request that had been deleted.
startRequest	The number of times an certificate validation request was started successfully.
startRequestEquivReqFnd	The number of times an certificate validation request was not started because a request for the same certificate already existed.
stopRequestFail	The number of times a request to stop a certificate validation request failed because the request had been deleted.
stopRequestGood	The number of times an certificate validation request was stopped successfully.
gncicResultNoCIC	The number of times a "no candidate issuer certificate found" event was received.
actionIllegal	The number of times an illegal action was produced by the certificate validation state machine.
startInitChecksTooDeep	The number of times a certificate validation request failed because the validation chain was too long.
startInitChecksBadTime	The number of times a certificate validation request failed because the certificate was not valid at the current time.

Table 46-3: Parameters in the output of the **show pki counters** command

Parameter	Meaning
startInitChecksIssNotCA	The number of times a certificate validation request failed because the certificate was being used to sign another certificate and was not a CA certificate.
startInitChcksBadKeyUsge	The number of times a certificate validation request failed because the certificate was being used to sign another certificate and had a critical key usage extension that did not allow key signing.
startInitChcksSSignNotCA	The number of times a certificate validation request failed because the certificate was self-signed and not a CA certificate.
startInitChcksSSgnUntrst	The number of times a certificate validation request failed because the certificate was self-signed and not manually trusted.
startInitChecksRevoked	The number of times a certificate validation request failed because the certificate was revoked.
startValCICCertNotFound	The number of times a certificate validation request failed because no issuer certificate was found.
startValCICAlreadyTrustd	The number of times a trusted issuer certificate was found for a certificate validation request.
startValCICAlrdyVldating	The number of times a certificate validation request failed because the issuer certificate was in the process of being validated.
startValCICStartVURqFail	The number of times a certificate validation request failed because the issuer certificate validation failed.
startVerifyCertNotFound	The number of times a certificate validation request failed because the certificate signature could not verified because the certificate had been deleted.
startVerifyCertParseFail	The number of times a certificate validation request failed because the certificate signature could not verified because a bad certificate format prevented the certificate from being parsed successfully.
startVerifyCrtFldGetFail	The number of times a certificate validation request failed because the certificate signature could not verified because a bad certificate format prevented a field from being read successfully.
startVerifyCICParseFail	The number of times a certificate validation request failed because the certificate signature could not verified because a bad issuer certificate format prevented the issuer certificate from being parsed successfully.
startVerifyCICFldGetFail	The number of times a certificate validation request failed because the certificate signature could not verified because a bad issuer certificate format prevented a field from being read successfully.
startVerifyFailImm	The number of times a certificate validation request failed because the certificate signature verification failed immediately.
startVerifyStarted	The number of times a certificate validation request certificate signature verification was started successfully.

Table 46-3: Parameters in the output of the **show pki counters** command

Parameter	**Meaning**
verifyCbackReqNotFound	The number of times a certificate signature verification finished and the certificate validation request had been deleted.
verifyCallbackVerifyGood	The number of times a certificate validation request certificate signature was verified successfully.
verifyCallbackVerifyFail	The number of times a certificate validation request certificate signature was different to the calculated signature.
createRequest	The number of times a certificate validation request was created.
destroyRequest	The number of times a certificate validation request was destroyed.
dbAdd	The number of times a certificate validation request was added to the certificate validation request database.
dbRemoveFail	The number of times a specified certificate validation request could not be found and removed from the certificate validation request database.
dbFindFail	The number of times a specified certificate validation request could not be found in the certificate validation request database.
dbFindByRequestorFail	The number of times a certificate validation request with a specified requester could not be found in the certificate validation request database.
dbFindByCertFail	The number of times a certificate validation request with a specified certificate could not be found in the certificate validation request database.
VALIDATION UNIT GET NEXT CANDIDATE ISSUER CERTIFICATE COUNTERS for events in the PKI certificate "get next candidate issuer certificate" unit	
eventRequestNotFound	The number of times an event was received for a get next candidate issuer certificate request that had been deleted.
getNextCICNotFound	The number of times the get next candidate issuer certificate unit failed to find another candidate issuer certificate.
remoteRetrievdReqNotFnd	The number of times a certificate was retrieved for a get next candidate issuer certificate request that had been deleted.
actionIllegal	The number of times an illegal action was produced by the get next candidate issuer certificate state machine.
startGetRemoteFail	The number of times the retrieval of a remote certificate for a get next candidate issuer certificate request failed.
RETRIEVAL UNIT COUNTERS – for events in the PKI certificate retrieval unit	
startOPStartFail	The number of times a certificate retrieval using a PKI Operational Protocol failed to start.
retrievedRequestNotExist	The number of times a certificate retrieval finished but the retrieval request had been deleted.
getFileNotExist	The number of times the retrieval of a certificate from a router file failed because the file did not exist.

Table 46-3: Parameters in the output of the **show pki counters** command

Parameter	Meaning
getFileOpenFail	The number of times the retrieval of a certificate from a router file failed because the file could not be opened.
getFileReadFail	The number of times the retrieval of a certificate from a router file failed because the file could not be read.
getFileCloseFail	The number of times the retrieval of a certificate from a router file failed because the file could not be closed.
PKI CRL UNIT COUNTERS – for events in the PKI CRL unit	
GENERAL COUNTERS – for general events in the PKI CRL unit	
retrievedNotFound	The number of times a CRL was retrieved for a CRL request that had been deleted.
retrievedParseFail	The number of times a CRL with an invalid format was retrieved that prevented the CRL from being parsed successfully.
retrievedGetFieldsFail	The number of times a CRL with an invalid format was retrieved that prevented a field from being read successfully.
retrievedRUStartFail	The number of times a CRL retrieval failed to start when the CRL request had been changed.
timeoutRUStartFail	The number of times a CRL retrieval failed to start when doing a periodic update.
addAlreadyExists	The number of times a CRL could not be added to the CRL database because a CRL with the same name already existed.
addRUStartFail	The number of times a CRL retrieval failed to start when initially added.
addGood	The number of times a CRL was successfully added to the CRL database.
setNotFound	The number of times a request to change some information in a CRL failed because the CRL was not found in the CRL database.
setRUStartFail	The number of times a CRL retrieval failed to start after information about it was changed.
setGood	The number of times information about a CRL was successfully changed to the CRL database.
setNothingSet	The number of times a request to change some information in a CRL failed because the requested settings were no different.
deleteNotFound	The number of times a request to delete a CRL failed because the CRL was not found in the CRL database.
deleteGood	The number of times a CRL was successfully deleted from the CRL database.
purgeGood	The number of times the CRL database was successfully purged of all entries.
showNotFound	The number of times a request to show information about a CRL failed because the CRL was not found in the CRL database.

Table 46-3: Parameters in the output of the **show pki counters** command

Parameter	Meaning
showListParseFail	The number of times a request to show information about a CRL in the CRL database failed because the CRL had an invalid format that prevented the CRL from being parsed successfully.
RETRIEVAL UNIT COUNTERS – for events in the PKI CRL retrieval unit	
startOPStartFail	The number of times a CRL retrieval using a PKI Operational Protocol failed to start.
retrievedRequestNotExist	The number of times a CRL retrieval finished but the retrieval request had been deleted.
getFileNotExist	The number of times the retrieval of a CRL from a router file failed because the file did not exist.
getFileOpenFail	The number of times the retrieval of a CRL from a router file failed because the file could not be opened.
getFileReadFail	The number of times the retrieval of a CRL from a router file failed because the file could not be read.
getFileCloseFail	The number of times the retrieval of a CRL from a router file failed because the file could not be closed.
PKI MANAGEMENT UNIT COUNTERS – for events in the PKI management unit	
CMP COUNTERS – for events in the PKI CMP enrollment unit.	
startedEnrollment	The number of Enrollment Requests that have been started.
startedKeyUpdate	The number of Key Update Requests that have been started.
receivedPKIMessage	The number of PKI CMP Messages that have been received.
receivedPKIFIN	The number of PKI FIN messages that have been received from CMP Transport.
receivedTCPError	The number of TCP errors that have been indicated by CMP TCP Transport.
receivedMsgVerifyOK	The number of received messages that have valid content and protection.
receivedMsgVerifyFail	The number of received messages that have invalid content or protection.
illegalEvent	The number of internal illegal events.
requestFailed	The number of Enrollment or Key update requests that have failed.
requestCompleted	The number of Enrollment or Key update requests that have been completed.
receivedMessageDiscarded	The number of unsolicited messages or out of sequence messages discarded.
retryEvent	The number of timeout and retry events.
timeoutEvent	The number of timeout and fail events.
requestQueueError	The number of CMP request queue errors.
MANUAL ENROLLMENT COUNTERS – for events in the PKI manual enrollment unit	
started	The number of manual enrollments started.
fileOpenFailed	The number of times an attempt to open a file has failed.

Table 46-3: Parameters in the output of the **show pki counters** command

Parameter	Meaning
failed	The number of manual enrollment requests that have failed.
completed	The number of manual enrollment requests that have been completed.
fileOpenOK	The number of files that have been opened successfully.
fileClosed	The number of files that have been closed successfully.
fileWriteFailed	The number of times a file write attempt has failed.
CSRCreationError	The number of times an error has occurred generating a Certificate Signing Request.

Examples To show general information of the PKI module, use the command:

```
sh pki
```

To show counters for the PKI module, use the command:

```
sh pki cou
```

Related Commands set pki

show pki certificate

Syntax `SHow PKI CERtificate[=`*`name`*`]`

where *name* is a character string 1 to 24 characters long. Valid characters are any printable characters. If the name contains spaces, it must be in double quotes. Wildcards are not allowed.

Description This command displays information about a certificate or all certificates in the router's certificate database (Figure 46-3 on page 46-36, Table 46-4 on page 46-37).

If a name is specified, details for the certificate are displayed (Figure 46-4 on page 46-37, Table 46-5 on page 46-38).

Figure 46-3: Example output from the **show pki certificate** command

```
Certificate Database:
 Name                   State      MTrust   Type   Source
------------------------------------------------------------
 d0 (dc=yogi, dc=com)   TRUSTED    TRUE     CA     VLD_UNIT
 bart                   TRUSTED    FALSE    EE     COMMAND
------------------------------------------------------------
```

Table 46-4: Parameters in the output of the **show pki certificate** command

Parameter	Meaning
Name	Name given to the certificate by the user or by the router’s dynamic retrieval facility. In the latter case the subject of the certificate is displayed for the user’s information.
State	Whether the certificate is TRUSTED, UNTRUSTED, or VALIDATING.
MTrust	Whether the certificate has been manually trusted by a user command.
Type	Whether the type of certificate is SELF, CA, or EE (end entity).
Source	Source of the certificate: COMMAND VLD_UNIT (dynamic retrieval by certificate validation unit) UM_ISAKMP (user module, ISAKMP) MGMT_PROT (management protocol, CMP)

Figure 46-4: Example output from the **show pki certificate=*name*** command

```
Certificate:
  name ................ router1
  state ............... TRUSTED
  manually trusted .... FALSE
  type ................ EE
  source .............. COMMAND

  version ............. V3
  serial number ....... 3bf1 c141 [1005699393]
  signature alg ....... SHA1 with RSA
  public key alg ...... RSA
  not valid before .... 03:55:03 - 14-Nov-2001  (GMT)
  not valid after ..... 04:25:03 - 14-Nov-2002  (GMT)
  subject ............. cn=router1, dc=foo, dc=bar, dc=com
  issuer .............. dc=foo, dc=bar, dc=com

  MD5 fingerprint ..... e81e bb17 deb3 664d 91e3 5c58 c890 aae1
  SHA1 fingerprint .... d662 ba63 ecb9 be83 0962 9ca1 5888 1bee d96b 67d6
  key fingerprint ..... 49d4 4919 106f ea71 21c7 7bef ab69 48c1 0ca8 99d2

  key usage ........... Digital Signature
  subject key ID ...... e70d3c808b6d747f2a415ccf7efc8e16a94c9f8d
  authority key ID .... dcc16049a4e158dcda046cecb90b91c9a94c6800

  validation path ..... <- foobar[ manually trusted, self-signed ]

  Source Location:
    type .............. LDAP
    IP address ........ 192.168.100.200
    distinguished name  cn=router1, dc=foo, dc=bar, dc=com
```

Table 46-5: Parameters in the output of the **show pki certificate=*name*** command

Parameter	Meaning
Certificate	Information about the certificate.
name	Name given to the certificate by the user or by the router's dynamic retrieval facility. In the latter case the subject of the certificate is displayed for the user's information.
state	Whether the certificate is TRUSTED, UNTRUSTED, or VALIDATING. When the state is UNTRUSTED, a reason is given.
manually trusted	Whether the certificate has been manually trusted by a user command.
type	Whether the type of certificate is SELF, CA, or EE (end entity).
source	Source of the certificate: COMMAND VLD_UNIT (dynamic retrieval by certificate validation unit) UM_ISAKMP (user module, ISAKMP)
version	Version of X.509 with which the certificate complies.
serial number	Serial number of the certificate.
sha1 fingerprint	SHA1 fingerprint of the certificate.
signature alg	Algorithm used to sign the certificate.
public key alg	Algorithm of the public key certified by the certificate.
not valid before	Date before which this certificate is not valid.
not valid after	Date after which this certificate is not valid.
subject	Distinguished name of the subject of the certificate.
issuer	Distinguished name of the issuer of the certificate.
MD5 fingerprint	MD5 fingerprint of the certificate.
SHA1 fingerprint	SHA1 fingerprint of the certificate.
key fingerprint	Key fingerprint of the certificate.
key usage	Usages for which the public key certified by the certificate is valid.
subject key ID	ID that distinguishes this key from other keys owned by the same user.
authority key ID	ID that determines the CA keys used to sign this certificate.
subject alt name	Alternative names of the subject of the certificate.
validation path	List of the certificates in the validation path of the certificate.
Source Location	Information about the source location of the certificate.
file	Name of a file in the router file system.
type	Type of address of the location of the certificate; either LDAP or HTTP.
IP address	IP address of the location of the certificate.
domain name	Domain name of the location of the certificate.
distinguished name	Distinguished name of the location of the certificate.
port	Port of the location of the certificate.
HTTP file name	HTTP file name of the location of the certificate.

Table 46-5: Parameters in the output of the **show pki certificate=*name*** command

Parameter	Meaning
username	Username used to access the location of the certificate.
password	Password used to access the location of the certificate.

Examples To show the contents of a certificate named 'router1', use the command:

```
sh pki cer=router1
```

Related Commands **add pki certificate**
delete pki certificate
set pki certificate

show pki crl

Syntax `SHow PKI CRL[=name]`

where *name* is a character string from 1 to 24 characters long. Valid characters are any printable characters. If the name contains spaces, it must be in double quotes. Wildcards are not allowed.

Description This command displays information about a CRL or all CRLs in the router's CRL database (Figure 46-5 on page 46-39).

If a name is specified, details for the named CRL are displayed (Figure 46-6 on page 46-40).

This command may produce a large amount of output that can be interrupted when paging is turned on (see the **set asyn** command on page 7-37 of Chapter 7, Interfaces).

Figure 46-5: Example output from the **show pki crl** command

```
 PKI CRL list:
  Name         State            Minutes-to-next-update
-------------------------------------------------------------
  ca1          UPTODATE         1439
-------------------------------------------------------------
```

Table 46-6: Parameters in the output of the **show pki crl** command

Parameter	Meaning	
PKI CRL list shows a summary of the CRLs about which the router has information		
Name	Name assigned to this CRL.	
State	State of the CRL:	
	GETTINGFIRST	router is attempting to retrieve the CRL for the first time
	GETTING	router is attempting to retrieve the CRL for an update
	NOTFOUNDFIRST	router failed to retrieve the CRL for the first time

Table 46-6: Parameters in the output of the **show pki crl** command

Parameter	Meaning	
	NOTFOUND	router failed to retrieve the CRL for an update
	UPTODATE	CRL is up to date and being used
	NOTVALIDATED	the most recent copy of the CRL retrieved could not be validated
	OUTOFDATE	the most recent copy of the CRL retrieved was invalid at the current date
	UNRECFORMAT	the most recent copy of the CRL retrieved was corrupt
Minutes-to-next-update	Number of minutes until the CRL is updated.	

Figure 46-6: Example output from the **show pki crl=*name*** command

```
PKI CRL: ca1
  State .................... UPTODATE
  Minutes to next update ... 1434
  Number of updates ........ 1

  Type ..................... CRL
  Version .................. V1
  Issuer ................... cn=Test CA 1, ou=Web test, o=SSH Communications
                             Security, c=FI
  Signature algorithm ...... SHA1 with RSA
  Number of entries ........ 21
  This update .............. 04:54:01 - 14-Mar-2001 (GMT)
  Next update .............. 06:00:00 - 14-Mar-2001 (GMT)

  Source Location:
    file .............. ca1.crl

  Certificate List:
    Certificate Serial Number         Revocation Date         Revocation Reason
---------------------------------------------------------------------------
    380f 893a [940542266]         13:01:51 - 19-Oct-1999  unused
    3817 4a1f [941050399]         18:53:19 - 27-Oct-1999  unused
    3818 9c41 [941136961]         18:56:01 - 28-Oct-1999  unused
    3869 5da3 [946429347]         01:02:28 - 29-Dec-1999  unused
    389f 59b0 [949967280]         23:48:00 - 07-Feb-2000  unused
    38a0 5ca4 [950033572]         18:12:53 - 08-Feb-2000  unused
    38b8 cdc8 [951635400]         07:10:01 - 27-Feb-2000  unused
    38ba 124d [951718477]         06:14:37 - 28-Feb-2000  unused
    38ff 3486 [956249222]         16:47:02 - 20-Apr-2000  unused
    3922 d125 [958583077]         17:04:37 - 17-May-2000  unused
    3934 2329 [959718185]         20:23:06 - 30-May-2000  unused
    3950 2941 [961554753]         02:32:33 - 21-Jun-2000  unused
    3950 656f [961570159]         06:49:19 - 21-Jun-2000  unused
---------------------------------------------------------------------------
```

Table 46-7: Parameters in the output of the **show pki crl=*name*** command

Parameter	Meaning	
PKI CRL	Name of the CRL.	
State	State of the CRL:	
	GETTINGFIRST	router is attempting to retrieve the CRL for the first time

Table 46-7: Parameters in the output of the **show pki crl=*name*** command (continued)

Parameter	Meaning	
	GETTING	router is attempting to retrieve the CRL for an update
	NOTFOUNDFIRST	router failed to retrieve the CRL for the first time
	NOTFOUND	router failed to retrieve the CRL for an update
	UPTODATE	the CRL is up to date and being used
	NOTVALIDATED	the most recent copy of the CRL retrieved could not be validated
	OUTOFDATE	the most recent copy of the CRL retrieved was invalid at the current date
	UNRECFORMAT	the most recent copy of the CRL retrieved was corrupt
Minutes to next update	Number of minutes until the CRL is updated.	
Number of updates	Number of times the router has retrieved this CRL.	
Type	Type of CRL. Currently always CRL (basic CRL).	
Version	x.509 CRL version with which the CRL complies.	
Issuer	Distinguished name of the CA issued the CRL.	
Signature algorithm	Algorithm used by the issuing CA to sign the CRL.	
Number of entries	Number of revoked certificates described by the CRL.	
This update	Date when this update of the CRL was issued.	
Next update	Date when the next update of the CRL will be issued.	
Source Location	Location from which the CRL is retrieved.	
Certificate List	List of certificates revoked by this CRL.	
Certificate Serial Number	Serial number of a revoked certificate.	
Revocation Date	Date the certificate was revoked.	
Revocation Reason	Reason for the revocation.	

Examples

To display the list of CRLs loaded into the router's CRL database, use the command:

```
sh pki crl
```

To display details about a particular CRL in the CRL database, use the command:

```
sh pki crl=mycrl
```

Related Commands

add pki crl
delete pki crl
set pki crl

show pki enrollmentrequest

Syntax `SHow PKI ENRollmentrequest[=name]`

where *name* is a character string from 1 to 24 characters long. Valid characters are any printable characters. If the name contains spaces, it must be in double quotes. Wildcards are not allowed.

Description This command displays information about current enrollment requests (Figure 46-7 on page 46-42). Note that enrollment requests are processed rapidly and it may not be possible to capture the output.

If a name is specified, details for the named request are displayed (Figure 46-8 on page 46-42).

Figure 46-7: Example output from the **show pki enrollmentrequest** command

```
Enrollment Requests
  Name           KeyId   Protocol      State
-----------------------------------------------------
  bob                  1  CMP           WAIT_FOR_IP
-----------------------------------------------------
```

Table 46-8: Parameters in the output of the **show pki enrollmentrequest** command

Parameter	Meaning
Enrollment Requests	Information about current enrollment requests.
Name	Name of the enrollment request.
KeyID	ENCO key identification number for the new certificate.
Protocol	Whether the PKI management protocol used for the enrollment request is CMP or MANUAL.
State	State of the request, which can be used for debugging purposes; one of WAIT_FOR_IP (the router is waiting for an Initialisation Response Message from the CA), WAIT_FOR_CERT_CHECK (the router is validating the new certificate) or WAIT_FOR_CONFOK (the router is waiting for an Confirmation OK Indication from the CA)

Figure 46-8: Example output from the **show pki enrollmentrequest=*name*** command

```
Enrollment Request:

  Name ................ bob
  KeyID ............... 1
  Protocol ............ CMP
  State ............... WAIT_FOR_IP
  Secret Value ........ ASDFGHJKLZ
  Reference Number .... 80303003
  Location:
    IP address ........ 192.168.1.100
```

Table 46-9: Parameters in the output of the **show pki enrollmentrequest=*name*** command

Parameter	Meaning
Enrollment Request	Information about the named enrollment request.
Name	Name of the enrollment request.
KeyID	ENCO key identification number for the new certificate.
Protocol	Whether the PKI management protocol used for the enrollment request is CMP or Manual.
State	State of the request, which can be used for debugging purposes; one of WAIT_FOR_IP (the router is waiting for an Initialisation Response Message from the CA), WAIT_FOR_CERT_CHECK (the router is validating the new certificate) or WAIT_FOR_CONFOK (the router is waiting for an Confirmation OK Indication from the CA)
Secret Value	Shared secret for Proof of Possession (of private key) transactions. A shared secret is required by some CAs.
Reference Number	Reference number for this request from the End Entity. A reference number is provided by some CAs.
Location	Address of the CA; either an IP address or a domain name. These parameters are relevant for CMP requests.
IP address	IP address of the CA.
domain name	Fully-qualified domain name of the CA of the type foo.bar.com.

Examples To show a summary of all current enrollment requests, use the command:

```
sh pki enr
```

To display details about a particular enrollment request, use the command:

```
sh pki enr=myreq
```

Related Commands **create pki enrollmentrequest**
destroy pki enrollmentrequest

show pki keyupdaterequest

Syntax `SHow PKI KEYUpdaterequest[=`*`name`*`]`

where *name* is a character string from 1 to 24 characters long. Valid characters are any printable characters. If the name contains spaces, it must be in double quotes. Wildcards are not allowed.

Description This command displays information about current key update requests. If a name is specified, details for the name are displayed.

Figure 46-9: Example output from the **show pki keyupdaterequest** command

```
Key Update Requests
  Name                KeyID   Certificate   State
------------------------------------------------------------
  upbob                   2   bob           WAIT_FOR_KUP
------------------------------------------------------------
```

Table 46-10: Parameters in the output of the **show pki keyupdaterequest** command

Parameter	Meaning
Key Update Requests	Information about all current key update requests.
Name	Name of the key update request.
KeyID	ENCO key identification number for the new certificate.
Certificate	Name of the certificate for which the keys are to be updated.
State	State of the request, which can be used for debugging purposes; one of "WAIT_FOR_IP" (the router is waiting for an Initialisation Response Message from the CA), "WAIT_FOR_KUP" (the router is waiting for an Key Update Response Message from the CA), "WAIT_FOR_CERT_CHECK" (the router is validating the new certificate) or "WAIT_FOR_CONFOK" (the router is waiting for an Confirmation OK Indication from the CA).

Figure 46-10: Example output from the **show pki keyupdaterequest=*name*** command

```
Key Update Request:

  Name ................ upbob
  KeyID ............... 2
  Certificate Name .... bob
  Location:
    IP address ........ 192.168.1.100
```

Table 46-11: Parameters in the output of the **show pki keyupdaterequest=*name*** command

Parameter	Meaning
Key Update Request	Information about the named current key update request.
Name	Name of the key update request.
KeyID	ENCO key identification number for the new certificate.
Certificate Name	Name of the certificate for which the keys are to be updated.
Location	Address of the CA; either an IP address or a domain name.
IP address	IP address of the CA.
domain name	Fully-qualified domain name of the CA of the type foo.bar.com.

Examples To show a summary of all current key update requests, use the command:

```
sh pki keyu
```

To display details about a particular key update request, use the command:

```
sh pki keyu=myreq
```

Related Commands create pki keyupdaterequest
destroy pki keyupdaterequest

show pki ldaprepository

Syntax `SHOW PKI LDaprepository`

Description This command displays information about the LDAP repositories configured on the router (Figure 46-11 on page 46-45, Table 46-12 on page 46-45).

Figure 46-11: Example output from the **show pki ldaprepository** command

```
LDAP Repository Information:

  Index       Name                           Address
------------------------------------------------------------
  0           foobar                         192.168.100.200
------------------------------------------------------------
```

Table 46-12: Parameters in the output of the **show pki ldaprepository** command

Parameter	Meaning
Index	The local index of the LDAP repository.
Name	The name given to the LDAP repository by the user.
Address	The IP address or domain name of the LDAP repository.

Examples To show summary of all LDAP repositories, use the command:

```
sh pki ld
```

Related Commands add pki ldaprepository
delete pki ldaprepository
set pki ldaprepository