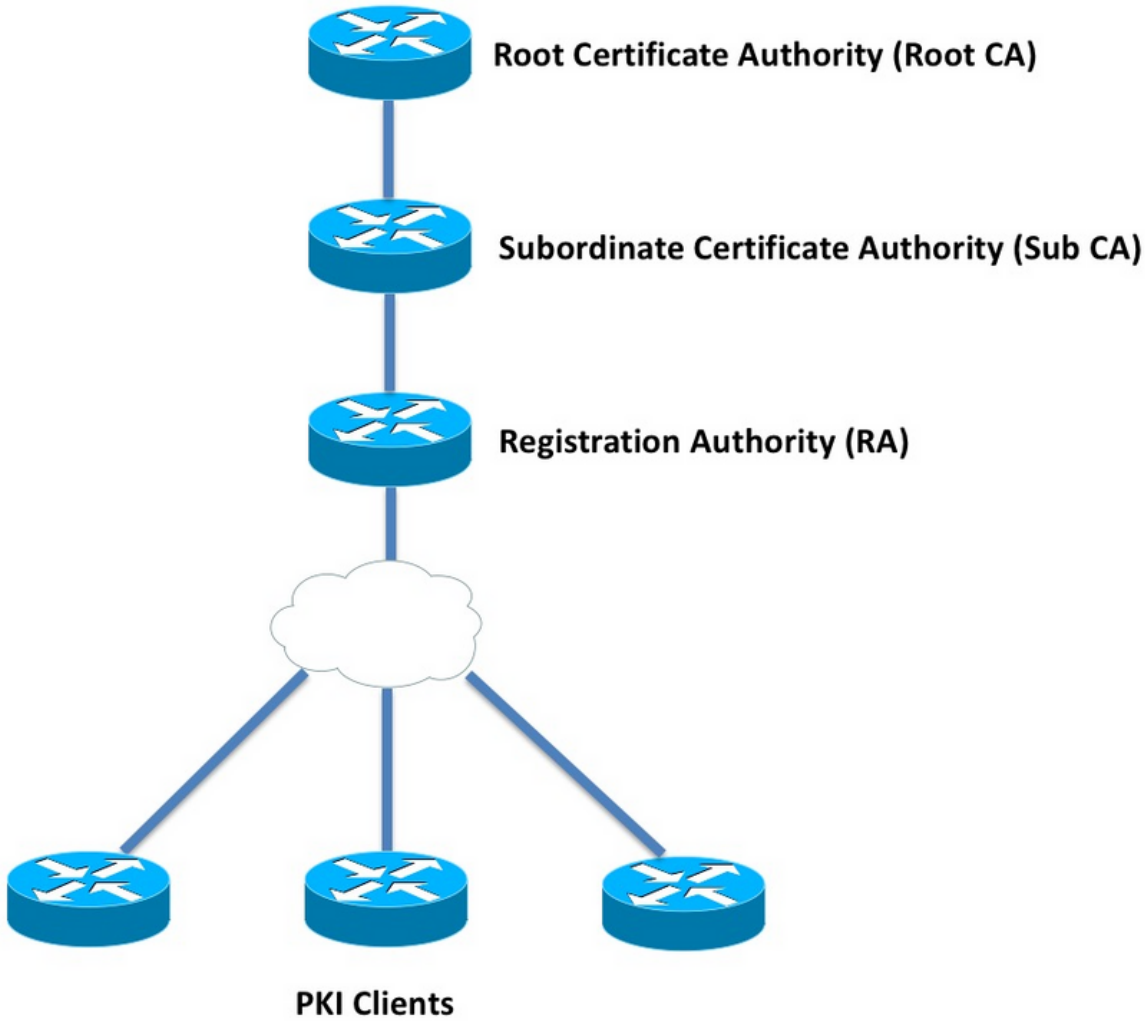
IOS PKI Deployment Guide: Initial Design and Deployment

Contents

Introduction

This document describes IOS PKI Server and client functionalities in detail. It addresses IOS PKI initial design and deployment considerations.

PKI Infrastructure



Certificate Authority

Certificate Authority (CA), also referred to as PKI Server throughout the document, is a trusted entity that issues certificates. PKI is based on trust, and trust-hierarchy starts at Root Certificate

Authority (Root-CA). Because the Root-CA is at the top of the hierarchy, it has a self-signed certificate.

Subordinate Certificate Authority

In PKI Trust-hierarchy all the certificate authorities below Root are known as Subordinate Certificate Authorities (Sub-CA). Evidently, a Sub-CA certificate is issued by the CA, which is one level above.

PKI imposes no limit on the number of Sub-CAs in a given hierarchy. However, in an enterprise deployment with more than 3 levels of certificate authorities may become difficult to manage.

Registration Authority

PKI defines a special certificate authority known as Registration Authority (RA), which is responsible for authorizing the PKI clients from enrolling to a given Sub-CA or Root-CA. RA does not issue certificates to PKI clients, instead it decides which PKI-Client can or cannot be issued a certificate by the Sub-CA or the Root-CA.

The main role of an RA is to offload basic client certificate request validation from the CA, and protect the CA from direct exposure to the clients. This way, RA stands between the PKI clients and the CA, thus protecting the CA from any kind of denial of service attacks.

PKI Client

Any device requesting for a certificate based on a resident public-private key-pair to prove its identity to other devices is known as a PKI client.

A PKI client must be capable of generating or storing a a public-private key-pair such as RSA or DSA or ECDSA.

A certificate is a proof of identity and validity of a given public-key, provided the corresponding private-key exists on the device.

IOS PKI Server

Table 1. IOS PKI Server Feature evolution

Feature	IOS [ISR-G1, ISR-G2]	IOS-XE [ASR1K, ISR4K]
IOS CA/PKI Server	12.3(4)T	XE 3.14.0 / 15.5(1)S
IOS PKI Server Certificate Rollover	12.4(1)T	XE 3.14.0 / 15.5(1)S
IOS PKI HA	15.0(1)M	NA [Implicit Inter-RP Redundancy is available]
IOS RA for 3rd Party CA	15.1(3)T	XE 3.14.0 / 15.5(1)S

Before getting into PKI Server configuration, the administrator must understand these core concepts.

Authoritative Source of Time

One of the foundations of PKI infrastructure is Time. The system clock defines whether a certificate is valid or not. Hence, in IOS, the clock must be made authoritative or trustworthy. Without an authoritative source of time, PKI server may not function as expected, and it is highly recommended to make the clock on IOS authoritative using these methods:

NTP (Network Time Protocol)

Synchronising the system clock with a Time Server is the only true way of making the system clock trustworthy. An IOS Router can be configured as an NTP client to a well-known and stable NTP server in the network:

```
configure terminal
ntp server <NTP Server IP address>
ntp source <source interface name>
ntp update-calendar

!! optional, if the NTP Server requires the clients to authenticate themselves
ntp authenticate
ntp authentication-key 1 md5 <key>

!! optionally an access-list can be configured to restrict time-updates from a specifc NTP
server
access-list 1 permit <NTP Server IP address>
ntp access-group peer 1
```

IOS can also be configured as an NTP server, which will mark the local system clock as authoritative. In small-scale PKI deployment, the PKI server can be configured as an NTP server for its PKI clients:

```
configure terminal
ntp master <stratum-number>

!! optionally, NTP authentication can be enforced
ntp authenticate
ntp authentication-key 1 md5 <key-1>
ntp authentication-key 2 md5 <key-2>
ntp authentication-key 2 md5 <key-2>
ntp trusted-key 1 - 3

!! optinally, an access-list can be configured to restrict NTP clients
!! first allow the local router to synchronize with the local time-server
access-list 1 permit 127.127.7.1
ntp access-group peer 1

!! define an access-list to which the local time-server will serve time-synchronization services
access-list 2 permit <NTP-Client-IP>
ntp access-group serve-only 2
```

Marking Hardware clock as trusted

In IOS, the hardware clock can be marked as authoritative using:

```
config terminal
clock calendar-valid
```

This can be configured along with NTP, and the key reason for doing this is to keep the system clock authoritative when a router reloads, for example due to a power outage, and the NTP servers are not reachable. At this stage, PKI timers will stop functioning, which in turn leads to Certificate Renewal/Rollover failures. **clock calendar-valid** acts as a safeguard in such situations.

While configuring this, it is key to understand that the system clock will go out of sync if the system battery dies, and PKI will start trusting an out-of-sync clock. However, it is relatively safer to configure this, than not having an authoritative source of time at all.

> **Note**: **clock calendar-valid** command was added in IOS-XE version XE 3.10.0 / 15.3(3)S onward.

Hostname and Domain Name

It is recommended to configure a hostname and a domain-name on Cisco IOS as one of the first steps before configuring any PKI related services. The Router hostname and domain-name are used in the following scenarios:

- Default RSA key-pair name is derived by combining the hostname and the domain-name
- When enrolling for a certificate, default subject-name consists of hostname attribute and an unstructured-name, which is hostname and domain-name put together.

As for PKI Server, hostname and domain-name are not used:

- Default key-pair name will be the same as that of the PKI server name
- Default Subject-name consists of CN, which is the same as that of the PKI server name.

General recommendation is to configure an appropriate hostname and a domain-name.

```
config terminal
hostname <string>
ip domain name <domain>
```

HTTP Server

IOS PKI Server is enabled only if HTTP Server is enabled. It is important to note that, if the PKI server is disabled due to HTTP server being disabled, it can continue to grant offline requests [via terminal]. HTTP Server capability is required to process SCEP requests, and send out SCEP responses.

IOS HTTP Server is enabled using:

```
ip http server
```

And the default HTTP server port can be changed from 80 to any valid port number using:

```
ip http port 8080
```

HTTP Max-connection

One of the bottlenecks, while deploying IOS as PKI server using SCEP is Maximum concurrent HTTP connections and average HTTP connections per minute.
Currently, the maximum concurrent connections on an IOS HTTP Server is limited to 5 by default and can be increased to 16, which is highly recommended in a medium scale deployment:

```
ip http max-connections 16
```

This IOS installations allow maximum concurrent HTTP connections up to 1000:

- UniversalK9 IOS with uck9 license-set

The CLI is automatically changed to accept a numerical argument between 1 to 1000

```
ip http max-connections 1000
```

IOS HTTP server allows 80 connections per minute [580 in the case of IOS releases where Max HTTP concurrent sessions can be increased to 1000] and when this limit is reached within a minute, IOS HTTP listener starts throttling the incoming HTTP connections by shutting down the listener for 15 seconds. This leads to client connection requests being dropped due to TCP **connection queue limit reached**. More information on this can be found here

RSA Key-pair

RSA key-pair for PKI Server functionality on IOS can be auto-generated or manually generated. While configuring a PKI Server, IOS automatically creates a Trustpoint by the same name as the PKI Server in order to store the PKI Server certificate.

Manually generating PKI Server RSA Key-pair:

Step 1. Create an RSA Key-pair with the same name as that of the PKI Server:

```
crypto key generate rsa general-keys label <LABEL> modulus 2048
```

Step 2. Before enabling the PKI Server, modify the PKI Server Trustpoint:

```
crypto pki trustpoint <PKI-SERVER-Name>
  rsakeypair <LABEL>
```

> **Note**: RSA Key-pair modulus value mentioned under the PKI Server trustpoint is not taken into consideration until IOS ver 15.4(3)M4, and this is a known caveat. The default key modulus is 1024 bits.

Auto-generating PKI Server RSA Key-pair:

When enabling the PKI Server, IOS automatically generates an RSA key-pair with the same name as that of the PKI Server, and the key modulus size is 1024 bits.

Starting IOS ver 15.4(3)M5, this configuration creates an RSA key-pair with <LABEL> as the name and the key-strength will be as per the defined <MOD> modulus.

```
crypto pki trustpoint <PKI-SERVER-Name>
rsakeypair <LABEL> <MOD>
```

Spoiler

CSCuu73408 IOS PKI server should allow for non-default key size for rollover cert.

CSCuu73408 IOS PKI server should allow for non-default key size for rollover cert.

Current industry standard is to use a minimum of 2048 bits RSA key-pair.

Auto-rollover timer consideration

Currently, IOS PKI Server does not generate a rollover certificate by default, and it has to be explicitly enabled under the PKI server using **auto-rollover <days-before-expiry>** command. More on Certificate rollover is explained in

This command specifies how many days before the PKI Server/CA certificate expiry should the IOS create a rollover CA certificate. Note that the rollover CA certificate is activated once the current active CA certificate expires. The default value currently is 30 days. This value should be set to a reasonable value depending on the CA certificate lifetime, and this in turn influences the auto-enroll timer configuration on the PKI client.

> **Note**: Auto-rollover timer should always trigger prior to auto-enroll timer on the client during CA and Client certificate rollover [known as]

CRL considerations

IOS PKI infrastructure supports two ways of distributing CRL:

Publish the CRL to an HTTP Server

IOS PKI Server can be configured to publish the CRL file to a specific location on an HTTP Server using this command under the PKI Server:

```
crypto pki server <PKI-SERVER-Name>
 database crl publish <URL>
```

And the PKI server can be configured to embed this CRL location into all the PKI client certificates using this command under the PKI Server:

```
crypto pki server <PKI-SERVER-Name>
cdp-url <CRL file location>
```

SCEP GetCRL Method

IOS PKI Server automatically stores the CRL file at the specific database location, which by default is nvram, and is highly recommended to keep a copy on an SCP/FTP/TFTP server using

this command under the PKI Server:

```
crypto pki server <PKI-SERVER-Name>
database url <URL>
or
database crl <URL>
```

By default, IOS PKI Server does not embed the CDP location into the PKI client certificates. If the IOS PKI clients are configured to perform revocation check, but the certificate being validated does not have a CDP embedded in it, and the validating CA trustpoint is configured with the CA location (using http://<CA-Server-IP or FQDN>), IOS falls back to SCEP based GetCRL method by default.
SCEP GetCRL performs CRL retrieval by executing HTTP GET on this URL:

```
http:://<CA-Server-IP/FQDN>/cgi-bin/pkiclient.exe?operation=GetCRL
```

> **Note**: In IOS CLI, prior to entering **?**, press **Ctrl + V** key-sequence.

IOS PKI Server can also embed this URL as the CDP location. The advantage of doing this is two-fold:

- It ensures that all non-IOS SCEP based PKI clients can perform CRL retrieval.
- Without an embedded CDP, IOS SCEP GetCRL request messages are signed (using a temporary self-signed certificate) as defined in the SCEP draft. However, CRL retrieval requests need not be signed, and by embedding the CDP URL for GetCRL method, signing the CRL requests can be avoided.

Lifetime of CRL

IOS PKI Server's CRL lifetime can be controlled using this command under the PKI Server:

```
crypto pki server <PKI-SERVER-Name>
 lifetime crl <0 - 360>
```

The value is in hours. By default the lifetime of the CRL is set to 6 hours. Depending on how frequently the certificates are revoked, tuning CRL lifetime to an optimum value increases the CRL retrieval performance in the network.

Database considerations

IOS PKI Server uses nvram as the default database location, and it is highly recommended to use an FTP or TFTP or SCP server as the database location. By default, IOS PKI Server creates two files:

- <Server-Name>.ser – This contains the last serial number issued by the CA in hex. The file is in plain-text format, and it contains this information:
  db_version = 1
  last_serial = 0x4

- <Server-Name>.crl – This is the DER encoded CRL file published by the CA

IOS PKI Server stores information in the database at 3 configurable levels:

- Minimum – This is the default level, and at this level no file is created in the database, and hence no information is available on the CA server regarding the client certificates granted in the past.
- Names – At this level IOS PKI server creates a file named <Serial-Number>.cnm for each client certificate issued, where the name <Serial-Number> refers to the serial number of the issued client certificate And this cnm file contains subject-name and the expiry date of the client certificate.
- Complete – At this level, IOS PKI Server creates two files for each client certificate issued:
- <Serial-Number>.cnm
- <Serial-Number>.crt

here, the crt file is the client certificate file, which is DER encoded.

These points are important:

- Before issuing a client certificate, IOS PKI Server refers to <Server-Name>.ser to determine and derive the Serial number of the certificate.
- With Database level set to Names or Complete, <Serial-Number>.cnm and <Serial-Number>.crt need to be written to the database before sending the granted/issued certificate to the client
- With database url set to Names or Complete, the database URL must have enough space to save the files. Hence the recommendation is to configure an external file server [FTP or TFTP or SCP] as the database URL.
- With External Database URL configured, it is absolutely necessary to make sure that the file server is reachable during certificate grant process, which otherwise would mark the CA Server as disabled. And manual intervention is required to bring the CA server back online.

Database archive

While deploying a PKI Server, it is important to consider the failure scenarios and be prepared, should there be a hadrware failure. There are two ways to achieve this:

1. Redundancy
   In this case, two devices or processing units act as Active-Standby to provide redundancy. IOS PKI Server highavailability can be achieved using two HSRP enabled ISR Routers [ISR G1 and ISR G2] as explained in

   IOS XE based systems [ISR4K and ASR1k] do not have device-redundancy option available. However, in ASR1k Inter-RP redundancy is available by default.
2. Archiving CA Server key-pair and files
   IOS provides a facility to archive the PKI Server Key-pair and the certificate. Archiving can be done using two types of files:
   PEM - IOS creates PEM formatted files to store RSA Public Key, Encrypted RSA Private

Key, CA Server certificate. Rollover Key-pair and certificates are automatically archivedPKCS12 - IOS creates a single PKCS12 file containing the CA Server certificate and the corresponding RSA Private Key encrypted using a password.

Database archiving can be enabled using this command under the PKI Server:

```
crypto pki server <PKI-SERVER-Name>
 database archive {pkcs12 | pem} password <password>
```

It is also possible to store the archived files to a separate server, possibly using a secure protocol (SCP) using the following command under the PKI Server:

```
crypto pki server <PKI-SERVER-Name>
 database url {p12 | pem} <URL>
```

Of all the files in the database except for the archived files and the .Ser file, all other files are in clear text and pose no real threat if lost, and hence can be stored on a seperate server without incurring much overhead while writing the files, for example a TFTP Server.

IOS as Sub-CA

IOS PKI Server by default takes up the role of a Root CA. To configure a subordinate PKI server (Sub-CA), first enable this command under the PKI Server configuration section (before enabling the PKI Server):

```
crypto pki server <Sub-PKI-SERVER-Name>
 mode sub-cs
```

Using this configure the Root-CA's URL under the PKI Server's trustpoint:

```
crypto pki trustpoint <Sub-PKI-SERVER-Name>
 enrollment url <Root-CA URL>
```

Enabling this PKI Server now triggers these events:

- PKI Server trustpoint is authenticated in order to install the Root-CA certificate.
- After the Root-CA is authenticated, IOS generates a CSR for the Subordinate-CA [x509 basic constraint containing CA:TRUE flag] and sends it to the Root-CA

Irrespective of the grant mode configured on the Root-CA, IOS puts the CA (or RA) certificate requests into pending queue. An administrator has to manually grant the CA certificates.
To view the pending certificate request and the request-id:

```
show crypto pki server <Server-Name> requests
```

To grant the request:

```
crypto pki server <Server-Name> grant <request-id>
```

- Using this, subsequent SCEP POLL (GetCertInitial) operation downloads the Sub-CA certificate and installs it on the router, which enables the Subordinate PKI Server

IOS as RA

IOs PKI Server can be configured as a Registration Authority to a given Subordinate or Root CA. To configure the PKI Server as a registration authority, first enable this command under the PKI Server configuration section (before enabling the PKI Server):

```
crypto pki server <RA-SERVER-Name>
```

```
 mode ra
```

Following this, configure the CA's URL under the PKI Server's trustpoint. This indicates which CA is protected by the RA:

```
crypto pki trustpoint <RA-SERVER-Name>
 enrollment url <CA URL>
 subject-name CN=<Common Name>, OU=ioscs RA, OU=TAC, O=Cisco
```

A Registration Authority does not issue certificates, hence the **issuer-name** configuration under the RA is not required, and is not effective even if it is configured. The subject-name of an RA is configured under the RA trustpoint using **subject-name** command. It is important to configure **OU = ioscs RA** as part of the subject-name in order for the IOS CA to identify the IOS RA i.e. to identify the certificate requests authorized by the IOS RA.

IOS can act as a Registration Authority to 3rd Party CAs such as Microsoft CA, and in order to stay compatible the IOS RA has to be enabled using this command under the PKI Server configuration section (before enabling the PKI Server):

```
mode ra transparent
```

In default RA mode, IOS signs the client requests [PKCS#10] using the RA certificate. This operation indicates the IOS PKI Server that the certificate request has been authorized by an RA.

With transparent RA mode, IOS forwards the client requests in their original format without introducing the RA certificate, and this is compatible with Microsoft CA as a well known example.

IOS PKI Client

One of the most important configuration entity in IOS PKI Client is a Trustpoint. The trustpoint configuration parameters are explained in detail in this section.

Authoritative Source of Time

As pointed out earlier, authoritative source of time is a requirement on the PKI client as well. IOS PKI client can be configured as an NTP client using these configuration:

```
configure terminal
ntp server <NTP Server IP address>
ntp source <source interface name>
ntp update-calendar

!! optional, if the NTP Server requires the clients to authenticate themselves
ntp authenticate
ntp authentication-key 1 md5 <key>

!! Optionally an access-list can be configured to restrict time-updates from a specifc NTP
server
access-list 1 permit <NTP Server IP address>
ntp access-group peer 1
```

Hostname and Domain Name

A general recommendation is to configure a hostname and a domain-name on the Router:

```
configure terminal
hostname <string>
ip domain name <domain>
```

RSA Key-Pair

In IOS PKI Client, RSA Key-pair for a given trustpoint enrollment can either be automatically generated or manually generated.

Automatic RSA key generation process involves the following:

- IOS by default creates 512 bit RSA Key-pair
- The automatically generated key-pair name is hostname.domain-name, which is the device hostname combined with the device domain-name
- The auto-generated key-pair is not marked as exportable.

Automatic RSA key generation process involves the following:

- Optionally, a general purpose RSA Key-pair of a suitable strength can be manually generated using:
- `crypto key generate rsa general-keys label <LABEL> modulus < MOD> [exportable]` Here, LABEL - the RSA key-pair name
  MOD - RSA key modulus or strength in bits between 360 till 4096, which is traditionally 512, 1024, 2048 or 4096.
  The advantage of manually generating the RSA key-pair is the ability to mark the key-pair as exportable, which in turn allows for the identity certificate to be completely exported, which can then be restored on another device. However, one should understand the security implications of this action.
- An RSA key-pair is linked to a trustpoint before enrollment using this command

  `crypto pki trustpoint MGMT`
  `rsakeypair <LABEL> [<MOD> <MOD>]` Here, if an RSA Key-pair named <LABEL> already exists, then it is picked up during trustpoint enrollment.
  If an RSA Key-pair named <LABEL> does not exist, then one of the following action is executed during the enrollment:
  - If no <MOD> argument is passed, then a 512 bits key-pair named <LABEL> is generated.
  - if one <MOD> argument is passed, then a <MOD> bits general purpose key-pair named <LABEL> is generated
  - if two <MOD> arguments are passed, then one <MOD> bits signature key-pair and one <MOD> bits encryption key-pair, both named <LABEL> are generated

Trustpoint

A trustpoint is an abstract container to hold a certificate in IOS. A single trustpoint is capable of storing two active certificates at any given time:

- A CA certificate - Loading a CA certificate into a given trustpoint is known as trustpoint authentication process.
- An ID certificate issued by the CA - Loading or Importing an ID certificate into a given trustpoint is known as trustpoint enrollment process.

A trustpoint configuration is known as a trust policy, and this defines that:

- Which CA certificate is loaded in the trustpoint?
- Which CA does the trustpoint enroll to?
- How does the IOS enroll the trustpoint?
- How a certificate issued by the given CA [loaded in the trustpoint] is validated?

Main components of a trustpoint are explained here.

Enrollment Mode

A trustpoint enrollment mode, which also defines the trustpoint authentication mode, can be performed via 3 main methods:

1. Terminal Enrollment - manual method of performing trustpoint authentication and certificate enrolment using copy-paste in the CLI terminal.
2. SCEP Enrollment - Trustpoint authentication and enrollment using SCEP over HTTP.
3. Enrollment Profile - Here, authentication and enrollment methods are defined separately. Along with terminal and SCEP enrollment methods, enrollment profiles provide an option to specify HTTP/TFTP commands to perform file retrieval from the Server, which is defined using an authentication or enrollment url under the profile.

Source Interface and VRF

Trustpoint authentication and enrolment over HTTP (SCEP) or TFTP (Enrollment Profile) uses the IOS file-system to perform file i/o operations. These packet exchanges can be sourced from a specific source interface and a VRF.

In case of classic trustpoint configuration, this functionality is enabled using **source interface** and **vrf** sub-commands under the trustpoint.

In case of enrollment profiles, **source interface** and **enrollment | authentication url <http/tftp://Server-location> vrf <vrf-name>** commands provide the same functionality.

Example configuration:

```
vrf definition MGMT
 rd 1:1
 address-family ipv4
 exit-address-family

crypto pki trustpoint MGMT
 source interface Ethernet0/0
 vrf MGMT
```

or

```
crypto pki profile enrollment MGMT-Prof
 enrollment url http://10.1.1.1:80 vrf MGMT
 source-interface Ethernet0/0
crypto pki trustpoint MGMT
 enrollment profile MGMT-Prof
```

Automatic Certificate Enrollment and Renewal

IOS PKI Client can be configured to perform automatic enrollment and renewal using this command under the PKI trustpoint section:

```
crypto pki trustpoint MGMT
auto-enroll <percentage> <regenerate>
```

Here, **auto-enroll <percentage> [regenerate]** command states that IOS should perform certificate renewal at exactly 80% of the current certificate's lifetime.

The keyword **regenerate** states that IOS should regenerate the RSA key-pair known as shadow key-pair during every certificate renewal operation.

This is the automatic enrollment behavior:

- The moment **auto-enroll** is configured, if the trustpoint is authenticated, IOS will perform an automatic enrollment to the server located at the URL mentioned as part of **enrollment url** command under the PKI trustpoint section or under the enrollment profile.
- The moment a trustpoint is enrolled with a PKI Server or a CA, a RENEW or a SHADOW timer is initialised on the PKI client based on the **auto-enroll** percentage of the current identity certificate installed under the trustpoint. This timer is visible under **show crypto pki timer** command. More on the timer fuctions *refer to*
- Renewal capability support comes from the PKI Server. More on this in
  IOS PKI Client performs two types of renewal:
  Implicit Renewal: If the PKI server does not send "Renewal" as a supported capability, IOS performs an initial enrollment at the defined auto-enroll percentage. i.e. IOS uses a self-signed certificate to sign the renewal request.Explicit Renewal: When the PKI Server supports PKI client certificate renewal feature, it advertises "Renewal" as a supported capability. IOS takes this capability into consideration during the certificate renewal i.e. IOS uses the current active identity certificate to sign the renewal certificate request.

Care should be taken while configuring auto-enroll percentage. On any given PKI client in the deployment, if a condition arises where the identity certificate expires at the same time as the issuing CA certificate, then the auto-enroll value should always trigger the [shadow] renewal operation after the CA has created the rollover certificate. *Refer to* **PKI Timer dependencies** section in

Certificate Revocation-check

An authenticated PKI trustpoint i.e. a PKI trustpoint containing a CA certificate is capable of performing certificate validation during an IKE or SSL negotiation, where the Peer certificate is subjected to a thorough certificate validation. One of the validation methods is to check the peer certificate revocation status using one of the following two methods:

- Certificate Revocation List (CRL) - This is a file containing the Serial numbers of the certificates revoked by a given CA. This file is signed using the issuing CA certificate. CRL method involves downloading the CRL file using HTTP or LDAP.

- Online Certificate Status Protocol (OCSP) - IOS establishes communication channel with an entity called as OCSP Responder, which is a designated Server by the Issuing CA. A Client such as IOS sends a request containing the serial number of the certificate is being validated. The OCSP responder responds with the revocation status of the given serial number. The communication channel could be established using any supported application/transport protocol, which usually is HTTP.

Revocation check can be defined using these command under the PKI trustpoint section:

```
crypto pki trustpoint MGMT
 revocation-check crl ocsp none
```

By default, a trustpoint is configured to perform revocation check using crl.

The methods can be re-ordered, and the revocation status check is performed in the defined order. The method "none" bypasses the revocation-check.

CRL cache

With CRL based revocation-check, every certificate validation may trigger a fresh CRL file download. And as the CRL file gets bigger or if the CRL distribution point (CDP) is farther away, downloading the file during every validation process hampers the performance of the protocol dependent on certificate validation. Hence, CRL caching is performed to improve the performance, and caching the CRL takes the CRL validity into consideration.

CRL validity is defined using two time parameters: **LastUpdate**, which is the last time the CRL was published by the issuing CA, and **NextUpdate**, which is the time is the future when a new version of CRL file is published by the issuing CA.

IOS caches every downloaded CRL for as long as the CRL is valid. However, under certain circumstances such as the CDP not being reachable temporarily, it may be necessary to retain the CRL in the cache for an extended period of time. In IOS a cached CRL can retained for as long as 24 hours after the CRL validity expires, and this can be configured using this command under the PKI trustpoint section:

```
crypto pki trustpoint MGMT
 crl cache extend <0 - 1440>
!! here the value is in minutes
```

Under certain circumstances such as an issuing CA revoking certificates within the CRL validity period, IOS can confiugured to delete the cache more frequently. By deleting the CRL prematurely, IOS is forced to download the CRL more frequently to keep the CRL cache up-to-date. This configuration option is available under the PKI trustpoint section:

```
crypto pki trustpoint MGMT
 crl cache delete-after <1-43200>
!! here the value is in minutes
```

And finally, IOS can be configured to not cache the CRL file using this command under the PKI trustpoint section:

```
crypto pki trustpoint MGMT
 crl cache none
```

Recommended Configuration

A typical CA deployment with Root-CA and a Sub-CA configuration is as below. The example also includes a Sub-CA configuration protected by an RA.

With 2048 bits RSA Key-pair across the board, this example recommends a setup where:
Root-CA has a lifetime of 8 years
Sub-CA has a lifetime of 3 years
Client certificates are issued for a year, which are configred to request for a certificate renewal,

automatically.

ROOT CA - Configuration

```
crypto pki server ROOTCA
database level complete
database archive pkcs12 password p12password
issuer-name CN=RootCA,OU=TAC,O=Cisco
lifetime crl 120
lifetime certificate 1095
lifetime ca-certificate 2920
grant auto rollover ca-cert
auto-rollover 85
database url ftp:://10.1.1.1/CA/ROOT/
database url crl ftp:://10.1.1.1/CA/ROOT/
database url crl publish ftp:://10.1.1.1/WWW/CRL/ROOT/
cdp-url http://10.1.1.1/WWW/CRL/ROOT/ROOTCA.crl
```

SUBCA without RA - Configuration

```
crypto pki server SUBCA
database level complete
database archive pkcs12 password p12password
issuer-name CN=SubCA,OU=TAC,O=Cisco
lifetime crl 12
lifetime certificate 365
grant auto SUBCA
auto-rollover 85
database url ftp:://10.1.1.1/CA/SUB/
database url crl ftp:://10.1.1.1/CA/SUB/
database url crl publish ftp:://10.1.1.1/WWW/CRL/SUB/
cdp-url http://10.1.1.1/WWW/CRL/SUB/SUBCA.crl
mode sub-cs


crypto pki trustpoint SUBCA
revocation-check crl
rsakeypair SUBCA 2048
enrollment url http://172.16.1.1
```

SUBCA with RA - Configuration

```
crypto pki server SUBCA
database level complete
database archive pkcs12 password p12password
issuer-name CN=SubCA,OU=TAC,O=Cisco
lifetime crl 12
lifetime certificate 365
grant ra-auto
grant auto rollover ra-cert
auto-rollover 85
 database url ftp:://10.1.1.1/CA/SUB/
database url crl ftp:://10.1.1.1/CA/SUB/
database url crl publish ftp:://10.1.1.1/WWW/CRL/SUB/
cdp-url http://10.1.1.1/WWW/CRL/SUB/SUBCA.crl
mode sub-cs

crypto pki trustpoint SUBCA
revocation-check crl
rsakeypair SUBCA 2048
enrollment url http://172.16.1.1
```

RA for SUBCA - Configuration

```
crypto pki server RA-FOR-SUBCA
database level complete
database archive pkcs12 password p12password
mode ra
grant auto RA-FOR-SUBCA
auto-rollover 85
database url ftp:://10.1.1.1/CA/RA4SUB/


crypto pki trustpoint RA-FOR-SUBCA
enrollment url http://172.16.1.2:80
password ChallengePW123
subject-name CN=RA,OU=ioscs RA,OU=TAC,O=Cisco
revocation-check crl
rsakeypair RA 2048
```

Certificate Enrollment

Manual Enrollment

Manual enrollment involves offline CSR generation on the PKI client, which is manually copied over to the CA. Administrator manually signs the request, which is then imported into the client.

PKI Client

PKI Client configuration:

```
crypto pki trustpoint MGMT
enrollment terminal
serial-number
ip-address none
password ChallengePW123
subject-name CN=PKI-Client,OU=MGMT,OU=TAC,O=Cisco
revocation-check crl
rsakeypair PKI-Key
```

Step 1. First authenticate the Trustpoint (this can also be performed after step 2).

```
crypto pki authenticate MGMT
!! paste the CA, in this case the SUBCA, certificate in pem format and enter “quit” at the end
in a line by itself] PKI-Client-1(config)# crypto pki authenticate MGMT

Enter the base 64 encoded CA certificate.
End with a blank line or the word "quit" on a line by itself

-----BEGIN CERTIFICATE-----
MIIDODCCAiCgAwIBAgIBAjANBgkqhkiG9w0BAQUFADAvMQ4wDAYDVQQKEwVDaXNj
bzEMMAoGA1UECxMDVEFDMQ8wDQYDVQQDEwZSb290Q0EwHhcNMTUxMDE4MjA0MjI3
WhcNMTgxMDE3MjA0MjI3WjAuMQ4wDAYDVQQKEwVDaXNjbzEMMAoGA1UECxMDVEFD
MQ4wDAYDVQQDEwVTdWJDQTCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEB
AJ7hKmBfDo/GOQAEYY/1ptpg28DejUE0ZlDorDkADP2vKfRI0kalSnOs2PIe01ip
7pHFurFVUx/p8teMCkmvnbrSBfyUrWo9YfQeGOELb4d3dSW4jGakm6M8lNRkO7HP
s+IVVTuJSeUZxov6DPa92Y/6HLayX15Iq8ZL+KwmA9oS5NeTi1tBbrcc3Hq8W2Ay
879nDDOqDOsQMQqKtc7E/IA7SBjowImra6FUxzgJ5ye5MymRfRYAH+c4qZJxwHTc
/tSmjiOJlM7X5dtehU/XPEEEbs78peXO9FyzAbhOtCRBVTnhc8WWijq84xu8Oej7
```

```
LbXGBKIHSP0uDe32CV0noEUCAwEAAaNgMF4wDwYDVR0TAQH/BAUwAwEB/zALBgNV
HQ8EBAMCAYYwHwYDVR0jBBgwFoAU+oNBdIj9mjpieQ2Z7v79JhKnL68wHQYDVR0O
BBYEFFOv8xtHRojMdJ65oQ2PFBeD5oHiMA0GCSqGSIb3DQEBBQUAA4IBAQAZ/W3P
Wqs4vuQ2jCnVE0v1PVQe/VNS54P/fprQRelceawiBCHA3D0SRgHqUWJUIqBLv4sD
QBegmyTmS76C8YC/jN7VbI30hf6R4qP7CWu8Ef9sWPRC/+Oy6e8AinrK+sVd2dp/
LLDMVoBhS2bQFLWiyRvC9FgyczXRdF+rhKTKeEVXGs7C/Yk/9z+/00rVmSGZAS+v
aPpZWjoC3459t51t8Y3iE6GtjBvmyxBwWt01/5gCu6Mszi7X/kXdmqgNfT5bBBnv
yjWE2ZS8NsH4hwDZpmDJqx4qhrH6bw3iUm+pK9fceZ/HTYasxtcr4NUvvxwXc60y
Wrtlpq3g2XfG+qFB
-----END CERTIFICATE-----
quit
Trustpoint 'MGMT' is a subordinate CA and holds a non self signed cert
Certificate has the following attributes:
      Fingerprint MD5: DBE6AFAC 9E1C3697 01C5466B 78E0DFE3
     Fingerprint SHA1: EAD41B32 BB37BC11 6E0FBC13 41701BFE 200DC46E

% Do you accept this certificate? [yes/no]: yes
Trustpoint CA certificate accepted.
% Certificate successfully imported
```

Step 2. Generate Certificate Signing request and Take the CSR to the CA and get the granted certificat:

```
PKI-Client-1(config)# crypto pki enroll MGMT
% Start certificate enrollment ..

% The subject name in the certificate will include: CN=PKI-Client,OU=MGMT,OU=TAC,O=Cisco
% The subject name in the certificate will include: PKI-Client-1.cisco.com
% The serial number in the certificate will be: 104Certificate Request follows:

MIIC2zCCAcMCAQAwdTEOMAwGA1UEChMFQ2lzY28xDDAKBgNVBAsTA1RBQzENMAsG
A1UECxMETUdNVDETMBEGA1UEAxMKUEtJLUNsaWVudDExMAoGA1UEBRMDMTA0MCMG
CSqGSIb3DQEJAhYWUEtJLUNsaWVudC0xLmNpc2NvLmNvbTCCASIwDQYJKoZIhvcN
AQEBBQADggEPADCCAQoCggEBANwa7g+DJxG57sMg020w1Fdv9+mIZ6R4livbt7vo
AbW8jpzQlMv4lV3r6ulTJumhBvV7xI+1ZijXP0EqqQZLNboYv37UTJgm83DGO57I
8RTn9DfDQpHiqvhtNuC5S3SCC/hvCxFXnfNXqC3dkfuVkVWoJiLZY87R6j44jUq0
tTL5d8t6lz2L0BeekzKJlOs73gONx0VgQyI/WjDiEwL0xF4DNHURaYyOxBWJc7/B
psDCf7376mb7XXz0LB++E8SvvM/Li6+yQzYv1Lagr0b8C4uE+tCDxG5OniNDiS82
JXsVd43vKRFW85W2ssrElgkuWAvS017XlwK+UDX21dtFdfUCAwEAAaAhMB8GCSqG
SIb3DQEJDjESMBAwDgYDVR0PAQH/BAQDAgWgMA0GCSqGSIb3DQEBBQUAA4IBAQA+
UqkqUZZar9TdmB8I7AHku5m79l42o8cuhwOccehxE6jmzh9P+Ttb9Me7l7L8Y2iR
yYyJHsL7m6tjK2+Gllg7RJdoxG8l8aMZS1ruXOBqFBrmo7OSz1nfXpiTyh88jyca
Hw/8G8uaYuQbZIj53BWmQGRpm7J//ktn0D4W3Euh9HttMuYYX7BOct05BLqqiCCw
n+kKHZxzGXy7JSZpUlDtvPPnnuqWK7iVoy3vtV6GoFOrxRoo05QVFehS0/m4NFQI
mXA0eTEgujSaQi4iWte/UxruO/3p/eHr67MtZXLRl0YDFgaQd7vD7fCsDx5pquKV
jNEUT6FNHdsnqrAKqodO

---End - This line not part of the certificate request---

Redisplay enrollment request? [yes/no]: no
```

Step 3. Now import the granted certificate via terminal:

```
PKI-Client-1(config)# crypto pki import MGMT certificate

Enter the base 64 encoded certificate.
End with a blank line or the word "quit" on a line by itself

-----BEGIN CERTIFICATE-----
MIIDcDCCAligAwIBAgIBAzANBgkqhkiG9w0BAQQFADAuMQ4wDAYDVQQKEwVDaXNj
```

```
bzEMMAoGA1UECxMDVEFDMQ4wDAYDVQQDEwVTdWJDQTAeFw0xNTEwMTkyMDM1MDZa
Fw0xNjEwMTgyMDM1MDZaMHUxDjAMBgNVBAoTBUNpc2NvMQwwCgYDVQQLEwNUQUMx
DTALBgNVBAsTBE1HTVQxEzARBgNVBAMTClBLSS1DbGllbnQxMTAKBgNVBAUTAzEw
NDAjBgkqhkiG9w0BCQIWFlBLSS1DbGllbnQtMS5jaXNjby5jb20wggEiMA0GCSqG
SIb3DQEBAQUAA4IBDwAwggEKAoIBAQDcGu4PgycRue7DINNtMNRXb/fpiGekeJYr
27e76AG1vI6c0JTL+JVd6+rpUybpoQb1e8SPtWYo1z9BKqkGSzW6GL9+1EyYJvNw
xjueyPEU5/Q3w0KR4qr4bTbguUt0ggv4bwsRV53zV6gt3ZH7lZFVqCYi2WPO0eo+
OI1KtLUy+XfLepc9i9AXnpMyiZTrO94DjcdFYEMiP1ow4hMC9MReAzR1EWmMjsQV
iXO/wabAwn+9++pm+1189CwfvhPEr7zPy4uvskM2L9S2oK9G/AuLhPrQg8RuTp4j
Q4kvNiV7FXeN7ykRVvOVtrLKxJYJLlgL0tNe15cCvlA19tXbRXX1AgMBAAGjUjBQ
MA4GA1UdDwEB/wQEAwIFoDAfBgNVHSMEGDAWgBRTr/MbR0aIzHSeuaENjxQXg+aB
4jAdBgNVHQ4EFgQUK+9/lrlL+TyYxvsgxzPwwrhmS5UwDQYJKoZIhvcNAQEEBQAD
ggEBAIrLrzFLnm9z7ula1uRh03r6dSCFy9XkOk6ZaHfksbENoDmkcgIwKoAsSF9E
rQmA9W5qXVU7PEsqOmcu8zEv7uuiqM4D4nDP69HsyToPjxVcoG7PSyKJYnXRgkVa
IYyMaSaRKWlhb2uWj3XPLzS0/ZBOGAG9rMBVzaqLfLAZgnQUVJvwsNofe+ASojk9
mCRsEHD8WVuAzcnwYKXx3j3x/T7jbB3ibPfbYKQqlS12XFHhJoK+HfSA2fyZBFLF
syN/B2Ow0bvc71YlYOQuYwz3XOMIHD6vARTO4f0ZIQti2dy1kHc+5lIdhLsn/bA5
yUo7WxnAE8LOoYIf9iU9q0mqkMU=
-----END CERTIFICATE-----
quit
% Router Certificate successfully imported
```

PKI Server

Step 1. First export the Issuing CA certificate from the CA, which in this case is SUBCA certificate. This is imported during step 1 above on the PKI client, i.e. Trustpoint authentication.

```
SUBCA(config)# crypto pki export SUBCA pem terminal
% CA certificate: !! Root-CA certificate
-----BEGIN CERTIFICATE-----
MIIDPDCCAiSgAwIBAgIBATANBgkqhkiG9w0BAQQFADAvMQ4wDAYDVQQKEwVDaXNj
bzEMMAoGA1UECxMDVEFDMQ8wDQYDVQQDEwZSb290Q0EwHhcNMTUxMDE4MjAwOTIx
WhcNMjMxMDE2MjAwOTIxWjAvMQ4wDAYDVQQKEwVDaXNjbzEMMAoGA1UECxMDVEFD
MQ8wDQYDVQQDEwZSb290Q0EwggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIB
AQCajfMy8gU3ZXQfKgP/wYKLB0cuywzYcDaSoNVlEvUZOWgUltCGP4CiCXyw0U0U
Zmy0rusibMV7mtkTX5muaPC0XfT98rswPiZV0qvEYpHF2YodPOUoqR3FeKj/tDbI
IikcLrfj87aeMJjCrWD888wfTN9Hw9x2QVDoSxLbzTLticXdXxwS5wxlM16GspmT
WL4fg1JRWgjRqMmOcpf716Or88XJ2N2HeWxxVFIwYQf3thHR6DgTdCgJ1uqjVE6q
1LQ1g8k81mvuCZX0uLZiTMJ69xo+Ot/RpeeE2RShxK5rh56ObQq4MT4lbIPKqIxU
lbKzWdh10NiYwjgTNwTs9GGvAgMBAAGjYzBhMA8GA1UdEwEB/wQFMAMBAf8wDgYD
VR0PAQH/BAQDAgGGMB8GA1UdIwQYMBaAFPqDQXSI/Zo6YnkNme7+/SYSpy+vMB0G
A1UdDgQWBBT6g0F0iP2aOmJ5DZnu/v0mEqcvrzANBgkqhkiG9w0BAQQFAAOCAQEA
VKwqI9vpmoRh9QoOJGtOA3qEgV4eCfXdMuYxmmo0sdaBYBfQm2RhZeQ1X90vVBso
G4Wx6cJVSXCtkqZTm1IoMtya+gdhLbKqZmxc+I5/js88SrbrBIm4zj+sOoySV9kW
THEEmZjdTCWXo2wNCr23gGdnb4RqZ0FTOfoZO/2Xnpcbvhz2/K7wlDRJ5k1wrsRW
RRwsQEh4LYMFIg0aBs4gmRLZ8ytwrvvrhQTVrAA/MeomUEPhcIYESg1AlWxoCYZU
0iqKfDa9+4wEJ+PMGDhM2UV0fuP0rWitKWxecSVbo54z3VHYwwCbz2jCs8XGE61S
+XlxCZKFVdlVaMmuaZTdFg==
-----END CERTIFICATE-----

% General Purpose Certificate: !! SUBCA certificate
-----BEGIN CERTIFICATE-----
MIIDODCCAiCgAwIBAgIBAjANBgkqhkiG9w0BAQUFADAvMQ4wDAYDVQQKEwVDaXNj
bzEMMAoGA1UECxMDVEFDMQ8wDQYDVQQDEwZSb290Q0EwHhcNMTUxMDE4MjA0MjI3
WhcNMTgxMDE3MjA0MjI3WjAuMQ4wDAYDVQQKEwVDaXNjbzEMMAoGA1UECxMDVEFD
MQ4wDAYDVQQDEwVTdWJDQTCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEB
AJ7hKmBfDo/GOQAEYY/1ptpg28DejUE0ZlDorDkADP2vKfRI0kalSnOs2PIe01ip
7pHFurFVUx/p8teMCkmvnbrSBfyUrWo9YfQeGOELb4d3dSW4jGakm6M8lNRkO7HP
s+IVVTuJSeUZxov6DPa92Y/6HLayX15Iq8ZL+KwmA9oS5NeTi1tBbrcc3Hq8W2Ay
879nDDOqDOsQMQqKtc7E/IA7SBjowImra6FUxzgJ5ye5MymRfRYAH+c4qZJxwHTc
/tSmjiOJlM7X5dtehU/XPEEEbs78peXO9FyzAbhOtCRBVTnhc8WWijq84xu8Oej7
LbXGBKIHSP0uDe32CV0noEUCAwEAAaNgMF4wDwYDVR0TAQH/BAUwAwEB/zALBgNV
```

```
HQ8EBAMCAYYwHwYDVR0jBBgwFoAU+oNBdIj9mjpieQ2Z7v79JhKnL68wHQYDVR0O
BBYEFFOv8xtHRojMdJ65oQ2PFBeD5oHiMA0GCSqGSIb3DQEBBQUAA4IBAQAZ/W3P
Wqs4vuQ2jCnVE0v1PVQe/VNS54P/fprQRelceawiBCHA3D0SRgHqUWJUIqBLv4sD
QBegmyTmS76C8YC/jN7VbI30hf6R4qP7CWu8Ef9sWPRC/+Oy6e8AinrK+sVd2dp/
LLDMVoBhS2bQFLWiyRvC9FgyczXRdF+rhKTKeEVXGs7C/Yk/9z+/00rVmSGZAS+v
aPpZWjoC3459t51t8Y3iE6GtjBvmyxBwWt01/5gCu6Mszi7X/kXdmqgNfT5bBBnv
yjWE2ZS8NsH4hwDZpmDJqx4qhrH6bw3iUm+pK9fceZ/HTYasxtcr4NUvvxwXc60y
Wrtlpq3g2XfG+qFB
-----END CERTIFICATE-----
```

Step 2. After Step-2 on the PKI-Client, take the CSR from the client and provide it for signing on the SUBCA using this command:

```
crypto pki server SUBCA request pkcs10 terminal pem
```

This command suggests that the SUBCA accepts a certificate signing request from the terminal, and once granted, the certificate data is printed in PEM format.

```
SUBCA# crypto pki server SUBCA request pkcs10 terminal pem
PKCS10 request in base64 or pem

% Enter Base64 encoded or PEM formatted PKCS10 enrollment request.
% End with a blank line or "quit" on a line by itself.
MIIC2zCCAcMCAQAwdTEOMAwGA1UEChMFQ2lzY28xDDAKBgNVBAsTA1RBQzENMAsG
A1UECxMETUdNVDETMBEGA1UEAxMKUEtJLUNsaWVudDExMAoGA1UEBRMDMTA0MCMG
CSqGSIb3DQEJAhYWUEtJLUNsaWVudC0xLmNpc2NvLmNvbTCCASIwDQYJKoZIhvcN
AQEBBQADggEPADCCAQoCggEBANwa7g+DJxG57sMg020w1Fdv9+mIZ6R4livbt7vo
AbW8jpzQlMv4lV3r6ulTJumhBvV7xI+1ZijXP0EqqQZLNboYv37UTJgm83DGO57I
8RTn9DfDQpHiqvhtNuC5S3SCC/hvCxFXnfNXqC3dkfuVkVWoJiLZY87R6j44jUq0
tTL5d8t6lz2L0BeekzKJlOs73gONx0VgQyI/WjDiEwL0xF4DNHURaYyOxBWJc7/B
psDCf7376mb7XXz0LB++E8SvvM/Li6+yQzYv1Lagr0b8C4uE+tCDxG5OniNDiS82
JXsVd43vKRFW85W2ssrElgkuWAvS017XlwK+UDX21dtFdfUCAwEAAaAhMB8GCSqG
SIb3DQEJDjESMBAwDgYDVR0PAQH/BAQDAgWgMA0GCSqGSIb3DQEBBQUAA4IBAQA+
UqkqUZZar9TdmB8I7AHku5m79l42o8cuhwOccehxE6jmzh9P+Ttb9Me7l7L8Y2iR
yYyJHsL7m6tjK2+Gllg7RJdoxG8l8aMZS1ruXOBqFBrmo7OSz1nfXpiTyh88jyca
Hw/8G8uaYuQbZIj53BWmQGRpm7J//ktn0D4W3Euh9HttMuYYX7BOct05BLqqiCCw
n+kKHZxzGXy7JSZpUlDtvPPnnuqWK7iVoy3vtV6GoFOrxRoo05QVFehS0/m4NFQI
mXA0eTEgujSaQi4iWte/UxruO/3p/eHr67MtZXLRl0YDFgaQd7vD7fCsDx5pquKV
jNEUT6FNHdsnqrAKqodO
quit
% Enrollment request pending, reqId=1
```

If the CA is in auto-grant mode, the granted certificate is displayed in PEM format above. When the CA is in manual granting mode, the certificate request is marked as **pending**, is assigned an id value and queued into enrollment request queue.

```
SUBCA#show crypto pki server SUBCA requests
Enrollment Request Database:

Router certificates requests:
ReqID  State      Fingerprint                      SubjectName
--------------------------------------------------------------------
1      pending    7710276982EA176324393D863C9E350E serialNumber=104+hostname=PKI-Client-
1.cisco.com,cn=PKI-Client,ou=MGMT,ou=TAC,o=Cisco
```

Step 3. Manually grant this request using this command:

```
SUBCA# crypto pki server SUBCA grant 1
% Granted certificate:
-----BEGIN CERTIFICATE-----
MIIDcDCCAligAwIBAgIBAzANBgkqhkiG9w0BAQQFADAuMQ4wDAYDVQQKEwVDaXNj
bzEMMAoGA1UECxMDVEFDMQ4wDAYDVQQDEwVTdWJDQTAeFw0xNTEwMTkyMDM1MDZa
```

```
Fw0xNjEwMTgyMDM1MDZaMHUxDjAMBgNVBAoTBUNpc2NvMQwwCgYDVQQLEwNUQUMx
DTALBgNVBAsTBE1HTVQxEzARBgNVBAMTClBLSS1DbGllbnQxMTAKBgNVBAUTAzEw
NDAjBgkqhkiG9w0BCQIWFlBLSS1DbGllbnQtMS5jaXNjby5jb20wggEiMA0GCSqG
SIb3DQEBAQUAA4IBDwAwggEKAoIBAQDcGu4PgycRue7DINNtMNRXb/fpiGekeJYr
27e76AG1vI6c0JTL+JVd6+rpUybpoQb1e8SPtWYo1z9BKqkGSzW6GL9+1EyYJvNw
xjueyPEU5/Q3w0KR4qr4bTbguUt0ggv4bwsRV53zV6gt3ZH7lZFVqCYi2WPO0eo+
OI1KtLUy+XfLepc9i9AXnpMyiZTrO94DjcdFYEMiP1ow4hMC9MReAzR1EWmMjsQV
iXO/wabAwn+9++pm+1189CwfvhPEr7zPy4uvskM2L9S2oK9G/AuLhPrQg8RuTp4j
Q4kvNiV7FXeN7ykRVvOVtrLKxJYJLlgL0tNe15cCvlA19tXbRXX1AgMBAAGjUjBQ
MA4GA1UdDwEB/wQEAwIFoDAfBgNVHSMEGDAWgBRTr/MbR0aIzHSeuaENjxQXg+aB
4jAdBgNVHQ4EFgQUK+9/lrlL+TyYxvsgxzPwwrhmS5UwDQYJKoZIhvcNAQEEBQAD
ggEBAIrLrzFLnm9z7ula1uRh03r6dSCFy9XkOk6ZaHfksbENoDmkcgIwKoAsSF9E
rQmA9W5qXVU7PEsqOmcu8zEv7uuiqM4D4nDP69HsyToPjxVcoG7PSyKJYnXRgkVa
IYyMaSaRKWlhb2uWj3XPLzS0/ZBOGAG9rMBVzaqLfLAZgnQUVJvwsNofe+ASojk9
mCRsEHD8WVuAzcnwYKXx3j3x/T7jbB3ibPfbYKQqlS12XFHhJoK+HfSA2fyZBFLF
syN/B2Ow0bvc71YlYOQuYwz3XOMIHD6vARTO4f0ZIQti2dy1kHc+5lIdhLsn/bA5
yUo7WxnAE8LOoYIf9iU9q0mqkMU=
-----END CERTIFICATE-----
```

Note: Manual enrollment of a Sub-CA to a Root-CA is not possible.

Note: A CA in a disabled state due to disabled HTTP server can manually grant the certificate requests.

Enrollment using SCEP

PKI Client configuration is:

```
crypto pki trustpoint MGMT
enrollment url http://172.16.1.2:80
 serial-number
 ip-address none
 password 7 110A1016141D5A5E57
 subject-name CN=PKI-Client,OU=MGMT,OU=TAC,O=Cisco
 revocation-check crl
 rsakeypair PKI-Key 2048
```

PKI Server configuration is:

```
SUBCA# show run all | section pki server
crypto pki server SUBCA
 database level complete
 database archive pkcs12 password 7 01100F175804575D72
 issuer-name CN=SubCA,OU=TAC,O=Cisco
 lifetime crl 12
 lifetime certificate 365
 lifetime ca-certificate 1095
 lifetime enrollment-request 168
 mode sub-cs
 auto-rollover 85
 database url ftp:://10.1.1.1/CA/SUB/
database url crl ftp:://10.1.1.1/CA/SUB/
database url crl publish ftp:://10.1.1.1/WWW/CRL/SUB/
```

Default mode of certificate request granting is manual:

```
SUBCA# show crypto pki server
Certificate Server SUBCA:
```

```
Status: enabled
State: enabled
Server's configuration is locked  (enter "shut" to unlock it)
Issuer name: CN=SubCA,OU=TAC,O=Cisco
CA cert fingerprint: DBE6AFAC 9E1C3697 01C5466B 78E0DFE3
Server configured in subordinate server mode
Upper CA cert fingerprint: CD0DE4C7 955EFD60 296B7204 41FB6EF6
 Granting mode is: manual
Last certificate issued serial number (hex): 4
CA certificate expiration timer: 21:42:27 CET Oct 17 2018
CRL NextUpdate timer: 09:42:37 CET Oct 20 2015
Current primary storage dir: unix:/SUB/
Current storage dir for .crl files: unix:/SUB/
Database Level: Complete - all issued certs written as <serialnum>.cer
Auto-Rollover configured, overlap period 85 days
Autorollover timer: 21:42:27 CET Jul 24 2018
```

Manual grant

Step 1. PKI Client: As a first step, which is mandatory, authenticate the trustpoint on the PKI client:

```
PKI-Client-1(config)# crypto pki authenticate MGMT
Trustpoint 'MGMT' is a subordinate CA and holds a non self signed cert
Certificate has the following attributes:
      Fingerprint MD5: DBE6AFAC 9E1C3697 01C5466B 78E0DFE3
     Fingerprint SHA1: EAD41B32 BB37BC11 6E0FBC13 41701BFE 200DC46E

% Do you accept this certificate? [yes/no]: yes
Trustpoint CA certificate accepted.
```

Step 2. PKI-Client: Following the trustpoint authentication, the PKI client can be enrolled for a certificate.

> **Note**: If auto-enroll is configured, the client will automatically perform enrollment.

```
config terminal
crypto pki enroll MGMT
```

Behind the scenes, these events take place:

- IOS looks for an RSA key-pair named PKI-Key. If it exists, it is picked up for requesting an identity certificate. If not, IOS creates a 2048 bit key-pair named PKI-Key, and then use it for requesting an identity certificate.
- IOS creates a certificate signing request in PKCS10 format.
- IOS then encrypts this CSR using a random symmetric key. The random symmetric key is encrypted using the public key of the recipient, which is the SUBCA (SUBCA's public key is available due to trustpoint authentication). The encrypted CSR, the encrypted random symmetric key and the recipient information is put together in PKCS#7 enveloped data.
- This PKCS#7 enveloped data is signed using a temporary self-signed certificate during the initial enrollment. The PKCS#7 enveloped data, the signing certificate used by the client and the client's signature are put together in a PKCS#7 signed data packet. This is base64 encoded, and then URL encoded. The resulting blob of data is sent as "message" argument in HTTP URI sent to the CA:

```
GET /cgi-bin/pkiclient.exe?operation=PKIOperation&message=MII…. HTTP/1.0
```

Step 3. PKI-Server:

When the IOS PKI Server receives the request, it checks these:

1. Checks if the enrollment request database contains a certificate request with the same transaction id associated with the new request.

> **Note**: A transaction id is an MD5 hash of the public-key, for which an identity certificate is being requested by the client.

2. Checks if the enrollment request database contains a certificate request with the same challenge-password as the one sent by the client.

> **Note**: If (1) returns true or both (1) and (2) together return true, then a CA server is capable of rejecting the request on the grounds of duplicate identity request. However, in such a case IOS PKI Server replaces the older request with the newer request.

Step 4. PKI-Server:

Manually grant the requests on the PKI server:

To view the request:

```
show crypto pki server SUBCA requests
```

To grant a specific request or all of the requests:

```
crypto pki server SUBCA grant <id|all>
```

Step 5. PKI-Client:

Meanwhile, a PKI client starts a POLL timer. Here, IOS performs GetCertInitial at regular intervals till SCEP CertRep = GRANTED along with the granted certificate is received by the client.

Once the granted certificate is received, IOS automatically installs it.