Setting Up Secure Device Provisioning (SDP) for Enrollment in a PKI

First Published: May 2, 2005
Last Updated: March 20, 2008

RSF Note - Update 12.5(1st)T Release number - global

This module describes how to use Secure Device Provisioning (SDP) in a public key infrastructure (PKI). SDP is a web-based certificate enrollment interface that can be used to easily deploy PKI between two end devices, such as a Cisco IOS client and a Cisco IOS certificate server. The end devices may or may not be directly connected to the network at the time of deployment or provisioning. SDP provides a solution for users deploying a large number of peer devices (including certificates and configurations).

Finding Feature Information in This Module

Your Cisco IOS software release may not support all of the features documented in this module. To reach links to specific feature documentation in this module and to see a list of the releases in which each feature is supported, use the "Feature Information for SDP in a PKI" section on page 46.

Finding Support Information for Platforms and Cisco IOS and Catalyst OS Software Images

Use Cisco Feature Navigator to find information about platform support and Cisco IOS and Catalyst OS software image support. To access Cisco Feature Navigator, go to http://www.cisco.com/go/cfn. An account on Cisco.com is not required.

Contents

Americas Headquarters:
Cisco Systems, Inc., 170 West Tasman Drive, San Jose, CA 95134-1706 USA

Prerequisites for Setting Up SDP

Setting Up SDP for Enrollment in a PKI

Before you set up SDP, your environment should meet the following requirements:

- The petitioner device and the server must have IP connectivity between each other.
- The introducer must have a web browser that supports JavaScript.
- The introducer must have enable privileges on the client device.
- A Cisco IOS Release 12.3(8)T PKI-enabled image or a later image.

Setting Up SDP for Enrollment in a PKI Using USB Tokens

To leverage USB tokens to provision devices with SDP, your environment should meet the following requirements:

- Both the petitioner device and the server must have IP connectivity between each other.
- The introducer must have a web browser that supports JavaScript.
- The introducer must have enable privileges on the client device.
- The introducer must have access to a petitioner device.
- The introducer must have access to the USB token and PIN, if configured.
- A Cisco IOS Release 12.4(15)T PKI-enabled image or a later image.

Note Cisco IOS Release 12.4(15)T or a later release provides the flexibility to move credentials stored on the USB token. However, the device used to configure the USB token may run any Cisco IOS Release 12.3(14)T PKI-enabled image or a later image.

Using SDP To Configure a Device for an Internet Connection Through a Service Provider

To leverage SDP to configure a device that is not connected to the Internet, your environment should meet the following requirements:

- The introducer must have a web browser that supports JavaScript.
- The introducer must have enable privileges on the client device.
- A Cisco router that supports a DHCP client and a PPPoE client and has a configured LAN or WAN interface.
- A Cisco IOS Release 12.5(**1st**)T PKI-enabled image or a later image. If a previous Cisco IOS release is used on one of the devices, the SDP functionality will default to the older Cisco IOS version.

Information About Setting Up SDP for Enrollment in a PKI

Before using SDP for certificate enrollment, you should understand the following concepts:

- SDP Overview, page 3
- How SDP Works, page 4
- SDP Leveraging USB Tokens, page 10

SDP Overview

SDP (also referred to as Trusted Transitive Introduction [TTI]) is a communication protocol that provides a bidirectional introduction between two end entities, such as a new network device and a Virtual Private Network (VPN). SDP involves the following three entities (see Figure 1):

- Introducer—A mutually trusted device that introduces the petitioner to the registrar. The introducer can be a device user, such as a system administrator.
  - An introducer can be configured as an administrative introducer, which allows an administrator performing the introduction to supply the name for the device being introduced. The supplied device name is used as if it were the name of an introducer in the normal SDP mechanisms, preserving the existing functionality of the SDP configuration. For more information on function of the administrative introducer, see the section "Authentication and Authorization Lists for an Administrative Introducer."
- Petitioner—A client, or new device, to be introduced to the secure network.
- Registrar—A server that authorizes the petitioner. The registrar can be a certificate server.

Figure 1 Post-Introduction Secure Communication

As of Cisco IOS Release 12.5(**1st**)T or a later release, the introducer can start the SDP process without the petitioner having prior Internet connectivity established. The use of the prep-connect phase and the connect phase provides the ability to configure a petitioner for Internet connectivity through a service provider. For more information on the prep-connect phase and the connect phase, see the section "How SDP Works."

The registrar communicates directly with an external authentication, authorization, and accounting (AAA) server to verify petitioner credentials, permit or deny enrollment, and retrieve specific petitioner configuration information. The petitioner and registrar server web pages to the introducer, the end user. The petitioner receives the bootstrap configuration from a remote management system via the introducer's web browser.

SDP is implemented over a web browser with six possible phases—prep-connect (optional), connect, start (optional), welcome, introduction, and completion. Each phase is shown to the user via a web page. For more information on each phase, see the section "How SDP Works."

How SDP Works

The following sections describe how SDP deploys PKI between two devices:

- SDP Prep-Connect Phase
- SDP Connect Phase
- SDP Start Phase
- SDP Welcome Phase
- SDP Introduction Phase
- SDP Completion Phase

The SDP process starts with one of three entry pages being loaded into the web browser by the introducer: the SDP prep-connect phase received from the administrator; the start phase loaded from the registrar; or the welcome phase loaded from the petitioner.

The sample figures show how to introduce the local device (the petitioner) to the secure domain of the registrar. The "introducer" is referred to as the end user.

SDP Prep-Connect Phase

The prep-connect page is optional. Without the prep-connect page, the petitioner must have IP connectivity established.

The administrator must configure the prep-connect template and send the prep-connect page to the introducer. For more information on configuring the prep-connect template, see the section "Default Prep-Connect Template."

The administrator must also obtain and communicate the username and password for the secure network to the introducer by a telephone call, an email, a secure email, a CD, or a USB token. The registrar may be configured to authenticate the introducer using an existing AAA infrastructure for example, an existing username and password database that is part of the existing corporate domain. The SDP prep-connect phase supports a one time password mechanism as is used by common AAA infrastructures. For more information on SDP and AAA, see the section "How SDP Uses an External AAA Database."

After receiving the prep-connect page, the introducer must load the page onto the computer where the HTTP browser will run. The introducer then loads the prep-connect page into the HTTP browser as a local file and then the prep-connect page is displayed (see Figure 2).

Figure 2 Sample SDP Prep-Connect Page

After the introducer selects the "Log onto Cisco Device" button, the login dialog box is displayed (see Figure 3). The introducer enters the factory default username (cisco) and password (cisco) of the Cisco device.

Figure 3 Sample Petitioner Login Dialog Box

The introducer authenticates with the petitioner and then Internet connectivity is tested by attempting to access a known URL. Access to www.cisco.com (198.133.219.25) is tested by default. The administrator can modify the URL to be used for testing connectivity by modifying the default prep-connect template. For more information about modifying the default test URL and other fields that the administrator may configure for the prep-connect page, see the section "Default Prep-Connect Template."

Note To mitigate the possibility that the prep-connect page could be modified to contain an IP address of an untrusted registrar or that a prep-connect page might be emailed from an untrusted source, use a secure method to send the prep-connect page, such as secure email.

If Internet connectivity is established the start page or welcome page is displayed, depending on the prep-connect template setting as defined by the administrator. If Internet connectivity is not established, the connect page is displayed.

SDP Connect Phase

The connect page is displayed only if the prep-connect page is used and there is no IP connectivity for the petitioner at the completion of the prep-connect phase. The connect page has three IP address assignment methods to allow flexibility for your Cisco IOS platform: Dynamic Host Configuration Protocol (DHCP), Point to Point Protocol over Ethernet (PPPoE), or static IP address assignment.

Note SDP functionality is not used with the Cisco IOS configuration to establish Internet connectivity. SDP functionality includes a signature on the Cisco IOS configuration guaranteeing the values have not changed in transit.

DHCP IP Address Assignment Method

If the introducer selects DHCP, the default method, for the IP address assignment method option (see Figure 4), pressing the Connect button causes petitioner to be configured for Internet connectivity.

Figure 4 Sample Connect Page for DHCP IP Address Assignment Method

PPPoE IP Address Assignment Method

If the introducer selects PPPoE, input fields for PPPoE username and password are displayed (seeFigure 5). The introducer must enter a the username and password as supplied by the Internet service provider then press the Connect button, which causes petitioner to be configured for Internet connectivity.

Figure 5 Sample Connect Page for PPPoE IP Address Assignment Method

Static IP Address Assignment Method

If the introducer selects static, input fields for the IP address, netmask, and the default gateway are displayed (see Figure 6). The introducer must enter the configuration values as supplied by the Internet service provider then press the Connect button, which causes petitioner to be configured for Internet connectivity.

Figure 6 Connect Page for Static IP Address Assignment Method

Connect Page IP Address Configuration

After IP address configuration, Internet connectivity is tested again by attempting to access a known URL configured by the administrator in the prep-connect template (www.cisco.com by default). If Internet connectivity is now established the start page or welcome page is displayed, depending on the prep-connect template setting as defined by the administrator. If Internet connectivity is not established, the introducer should verify the settings entered or contact their administrator.

SDP Start Phase

The start page is optional. Without the start page, during the SDP exchange, the user clicks the Next button on the welcome page and is sent to the registrar's introduction page. Because the user has not previously connected to the registrar, the user is required to log in to the registrar using available credentials (per the registrar configuration). Some browsers fail to reconnect to the registrar after the user credentials are requested (after the user has entered the login data). As of Cisco IOS Release 12.4(4)T, users may configure their browsers to begin the SDP exchange by contacting the registrar's introduction URL via a start page. Thereafter, the registrar can direct the user to the welcome page, which is on the petitioner device. The SDP transaction will then continue through the welcome, introduction, and completion phases as described in this document.

To begin the SDP transaction from the registrar, the user must configure the browser via the **template http start** command; otherwise, the SDP transaction must begin from the welcome page on the petitioner. For more information on how to configure a custom template, see the section "How Custom Templates Work with SDP."

Before the welcome page is displayed, the user must direct his or her browser to the start page via the URL "http://registrar/ezsdd/intro". A login dialog box is then displayed and the end user can log into the registrar via a username and password supplied by the administrator to access the secure network (see Figure 7).

Figure 7 Registrar Remote Login Dialog Box

After entering a valid user name and password, the start page is displayed (see Figure 8).

Figure 8 Sample SDP Start Page

After entering the URL of the petitioner's welcome page (for example, http://10.10.10.1/ezsdd/welcome) and clicking the Next button on the start web page, the end user enters the SDP welcome phase and logs into his or her petitioner as shown in Figure 9.

SDP Welcome Phase

The welcome phase begins when the user clicks the Next button on the start page. Before the welcome page is displayed, the user must log into the petitioner via the URL “http://10.10.10.1/ezsdd/welcome.” The local login dialog box is then displayed (see Figure 9). The end user can then log into the local device via the factory default username (cisco) and password (cisco).

Figure 9 Petitioner Local Login Dialog Box

After the password is successfully entered, the welcome web page is displayed (see Figure 10), which is served by the petitioner.

Figure 10 Sample SDP Welcome Page

After entering the URL of the registrar (for example, http://192.0.2.155/ezsdd/intro) and clicking the Next button on the welcome web page, the SDP introduction phase begins and the introduction page is displayed, which is served by the registrar.

SDP Introduction Phase

Before the introduction page is displayed, the end user must log into the registrar if the user has not already done so from the start page (see "SDP Start Phase"), which utilizes the external AAA database.

With an external AAA database, the introducer can use an account on the database to perform the introduction without requiring knowledge of the enable password of the registrar. Without an external AAA database, the introducer may use the enable password of the registrar for authentication.

Note Using the enable password of the registrar exposes the password to end users; therefore, it is recommended that the enable password be used for administrative testing only.

The administrative introducer is identified by the HTTP authentication for the introduction page (or the start page), with the AAA database query returning administrative privilege for the user. If the introducer has administrator privilege, the device name is that which was entered in the administrative introduction page. If the introducer does not have administrative privileges, the device name is the introducer name. The existing device certificate is the current certificate on the petitioner, which may be the

manufacturing identification certificate (MIC). This certificate may or may not exist. For more information on the function of the external AAA database, see the section "How SDP Uses an External AAA Database."

After the end user successfully enters his or her password, the introduction web page is displayed (see Figure 11).

Figure 11 Sample SDP Introduction Page

At this point, the registrar passes device information to the external management system to obtain a bootstrap configuration file. For more information on options available to identify a customized bootstrap configuration file, see the section "Custom HTML Template Expansion Rules."

After the end user clicks the Next button on the introduction page, the end user enters the completion phase and automatically returns to his or her local device.

SDP Completion Phase

Now that the end user has enrolled the petitioner with the registrar, the petitioner will serve the completion page (see Figure 12).

Figure 12 Sample SDP Completion Page

The SDP exchange is now complete. The petitioner has received configuration information from the registrar and should received a certificate from the registrar shortly.

SDP Leveraging USB Tokens

SDP provides for highly scalable deployments and streamlines the deployment of an individual device or multiple devices. USB tokens provide for secure storage and configuration distribution.

As of Cisco IOS Release 12.4(15)T or a later release, USB tokens may be utilized to transfer PKI credentials using SDP to a remote device, and SDP may be used to configure the USB token. The USB token may then be used to provision a device at the same location, or the USB token may be transported to another location where it may be used to provision a remote device. For more information about configuring and using USB tokens, see the "Related Documents" section.

An example SDP deployment using a USB token to transfer PKI credentials is shown in Figure 13. The required devices include the USB token and the SDP entities required to provision a device. These SDP entities are the introducer, the registrar, a petitioner at the local location, Petitioner A, and a petitioner at the remote location, Petitioner B. Optionally, a management server may be used.

Note An optional configuration would be to configure one device as both the registrar and a petitioner, which may be beneficial when the USB token is transported to a remote location. The remote location would not require a separate petitioner device.

Figure 13 Example SDP Environment Using USB Tokens to Transfer Credentials

Use of SDP to Configure the USB Token

Prior to initiating an SDP introduction a USB token is inserted into the petitioner device. In the example configuration shown in Figure 13, the USB token would be inserted into Petitioner A. The petitioner may be configured to ignore any existing information on the USB token. As in regular SDP operations, for a scalable configuration of USB tokens, an initial template configuration has to be prepared and placed onto each SDP device with appropriate target configuration information.

Files used to provision a device are moved in the following sequence, shown by the numbered arrows in Figure 13.

1. One petitioner, Petitioner A, is at the local location. petitioner A engages directly with the SDP exchange to perform the initial configuration of the USB token. Files used to configure the USB token, binary files and template files, are retrieved from the registrar and moved to Petitioner A.

   The URL for the binary file location is expanded on the registrar. Binary files are not processed through the template expansion functions. The template expansion occurs on the registrar for both the source URL and destination URL.

   By default, binary files and template files will be retrieved from and stored to NVRAM on the registrar and petitioner respectively. The binary file location on the registrar and the destination binary file location on Petitioner A may be specified with the **binary file** command. The template file location on the registrar and the destination template file location on Petitioner A may be specified with the **template file** command.
2. The RSA keys and certificate chain information are moved from Petitioner A to the USB token.
3. The USB token is transported to the remote location where it is inserted into Petitioner B.
4. The configuration files on the USB token are used to provision the local device. Files from the USB token may be moved to a storage location on Petitioner B with the **crypto key move rsa** command.

SDP Phases with a USB Token

The same SDP phase concepts introduced in the "SDP Overview" section are used, with the following distinctions in the SDP welcome phase, the SDP introduction phase, and the SDP completion phase.

SDP Welcome Phase with a USB Token

The SDP welcome phase begins as usual, when an introduction is initiated by connecting to the welcome user interface. If there is an existing certificate on the USB token, it will be used for signing the SDP exchange. Instead of a local RSA key pair, a new RSA key pair on the token is used.

Note The RSA key pair generation may take a substantial length of time, anywhere from 5 to 10 minutes if the key is generated on the token. The length of time is dependent on hardware key generation routines available on the USB token. An informative web page will be presented to the introducer, indicating that RSA key pair generation is occurring.

The new key pair generated by Petitioner A is added to the USB token without removing any existing RSA key pairs. SDP AV pairs indicate both that a token is being used and if there is any token secondary configuration information. If an optional management server is in use, the AV pair information is used to determine if any special configuration commands are needed.

SDP Introduction Phase with a USB Token

The SDP Introduction phase begins with AV pairs being transferred to the registrar. When the registrar detects USB token related AV pairs, the registrar, if previously configured, may prepare configuration information destined for the USB token. Currently configuration commands are sent as a specific configuration files that are subsequently merged with the running configuration.

The administrator can leverage normal SDP configuration commands to configure the USB token. USB token information that should be configured includes the certificate, the bootstrap configuration, and the PIN number configuration.

SDP Completion Phase with a USB Token

At the beginning of the completion phase, the introduction proceeds with AV pairs being transferred to the petitioner (in Figure 13, this would be Petitioner A). The various files are stored in the specified file system locations and then the existing configuration file processing proceeds. This ordering allows the configuration to take advantage of the new files that have been transferred.

Use of the Configured USB Token

After the USB token is configured by Petitioner A, it is transported from its current location to the remote location, where the second petitioner, Petitioner B is located. The USB token is inserted into the target device, Petitioner B, which then inherits the USB token configuration and cryptographic material from the USB token. The end user at the remote location must have the PIN number on the USB token. The PIN number is either the default factory PIN or the PIN number the administrator configured during the introduction phase.

How SDP Uses an External AAA Database

The external AAA database is accessed twice during the SDP exchange. The first time the AAA database is accessed the introducer is authenticated; that is, when the registrar receives an introduction request via the secure HTTP (HTTPS) server, the registrar does an AAA lookup based on the introducer's username and password to authorize the request. The second time the AAA database is accessed authorization information is obtained and applied to the configuration and certificates that are issued to the petitioner device; that is, the registrar checks the integrity of the request by verifying the request signature using the petitioner-signing certificate. The certificate subject name may be specified in the AAA database, and up to nine configuration template variables may be specified and expanded into the template configuration.

Use of a Self-Signed Certificate Versus a Certificate Issued by Another CA Server

By default, the SDP exchange results in only one certificate being issued to the petitioner device. Although just one certificate is issued, the introducer is not restricted from introducing multiple devices and thus obtaining multiple certificates. By specifying the subject name in the certificate that is issued, you can be assured that all certificates that are issued in this way are associated with the introducer. You can use PKI AAA integration to further restrict the use of these certificates. Additionally, the AAA database can be configured to accept only one authentication and authorization request per user.

Because the petitioner certificate is self-signed, it is just used to convey the public key of the petitioner. No verification or authorization check is performed on the certificate; thus, authorization is per-user based and no per-device information is used.

There are some scenarios when per-device authorization is preferred. Therefore, if the petitioner is able to use certificates issued by other certification authority (CA) servers for SDP transactions, the existing PKI can be used and authorization can be achieved over the certificate attributes.

Configuring the petitioner and the registrar for certificate-based authorization provides authorization of the specific device being deployed. Previously, introducer-to-petitioner device communication was secured only using physical security between the introducer and the petitioner device. SDP certificate-based authorization gives the registrar an opportunity to validate the current device identity before accepting the introduction.

Authentication and Authorization Lists for SDP

When you are configuring your SDP registrar, if you specify an authentication list and an authorization list, the registrar uses the specified lists for all introducer requests. The authentication list is used when authenticating the introducer (the AAA server checks for a valid account by looking at the username and password). The authorization list is used to receive the appropriate authorized fields for the certificate subject name and a list of template variables to be expanded into the Cisco IOS command-line interface (CLI) snippet that is sent back to the petitioner. The authentication and authorization lists will usually point to the same AAA server list, but it is possible to use a different database for authentication and authorization. (Storing files on different databases is not recommended.)

When a petitioner makes an introduction request, multiple queries are sent to the AAA list database on the RADIUS or TACACS+ server. The queries search for entries of the following form:

```
user Password <userpassword>
   cisco-avpair="ttti:subjectname=<<DN subjectname>>"
   cisco-avpair="tti:iosconfig#<<value>>"
   cisco-avpair="tti:iosconfig#<<value>>"
   cisco-avpair="tti:iosconfig#=<<value>>"
```

Note The existence of a valid AAA username record is enough to pass the authentication check. The "cisco-avpair=tti" information is necessary only for the authorization check.

If a subject name was received in the authorization response, the SDP registrar stores it in the enrollment database, and that "subjectname" overrides the subject name that is supplied in the subsequent certificate request (PKCS10) from the petitioner device.

The numbered "tti:iosconfig" values are expanded into the SDP Cisco IOS snippet that is sent to the petitioner. The configurations replace any numbered ($1 through $9) template variable. Because the default Cisco IOS snippet template does not include the variables $1 through $9, these variables are ignored unless you configure an external Cisco IOS snippet template. To specify an external configuration, use the **template config** command. For more information on external configurations, see the section "Custom Configuration and File Template Variable Expansion Rules at the Petitioner."

Note The template configuration location may include a variable "$n," which is expanded to the name with which the user is logged in.

Authentication and Authorization Lists for an Administrative Introducer

The SDP mechanisms assume a permanent relationship between the introducer and the device. As a result, the introducer username is used to define the device name.

In some SDP deployment scenarios, the introducer is an administrator doing the introduction for many devices. However, using the introducer (the administrator) name to define the device name results in multiple devices being incorrectly deployed with the same device name. Instead, an administrative introducer allows the administrator to specify the correct device name during the introduction.

More generally stated, the introducer username is used as the database record locator to determine all other information about the device including the Cisco IOS configuration template, various template variables (pulled from an AAA database and expanded into the template), and the appropriate subject name for PKI certificates issued to the device. For simplicity, this database record locator is called the user/device name.

The administrative introducer provides a device name. In that way, an administrator can provide the appropriate record locator when doing an introduction. For example, if an administrator is trying to introduce a device for username "user1," the administrator introduces the device into the PKI network and provides user1 as the record locator after logging into the registrar using the administrator's own credentials. The record locator, user1, becomes the device name. All other template and PKI certificate subject name information specific to the introduction is then provided by the user1 username records instead of by the administrator's record.

The registrar device uses the supplied username information with a user introducer name. The username allows the existing mechanisms for determining a user's authorization, template, and PKI certificate information to be supported without modification.

How Custom Templates Work with SDP

You may use custom templates to streamline the SDP process.

- Custom templates allow you to complete the web pages with the required start information, so the introducer is no longer required to contact the registrar and can immediately begin the SDP transaction.
- Custom templates allow customized deployment information to be displayed on the web pages, thereby tailoring the user experience.

An easy way to define a custom template is to modify the default template. Without custom templates, the introducer must contact the registrar for information to begin the SDP transaction. For a list of the default templates, see the section "Default Templates for SDP Transaction Web Pages."

Note It is recommended that only advanced SDP users configure custom templates because problems can result from modifying templates incorrectly before the templates are displayed in the introducer's browser.

Custom Template Variable Expansion

There are expansion variables in the templates that are replaced by the Cisco IOS SDP registrar or petitioner. These variables are expanded as follows:

- $$—"$"
- $a—attribute-value (AV) pairs
- $c—Trusted certificate
- $d—Dump AV pairs in browser
- $h—Hostname
- $k—Keylabel or "tti"
- $l—Trustpoint label = "tti"
- $n—HTTP client's username
- $s—Default TTI key size
- $t—Trustpoint configuration
- $u—Completion URL
- $1 to $9—Variables retrieved from AAA server during user authentication.

Custom Template Variable Expansion Rules

Configuration and templates are used during an SDP exchange. Prior to use and after distribution, these templates are expanded using the following rules based in the SDP communication stage.

Custom HTML Template Expansion Rules

HTML templates are expanded immediately before being served to the HTTP client. The HTTP templates are expanded as follows:

- $u—Completion url, which is be populated with the SDP completion URL (for example: http://10.10.10.1/ezsdd/completion). This variable is used internally by SDP as the internal "wizard" state. It is expected that the SDP introduction page include something similar to the following text: "<FORM action=\"$u\"method=\"post\">" for normal wizard processing.
- $n—introducer name or the device name entered by the administrative introducer.
- $$—$
- $h—Hostname
- $a—All AV pairs with or without a specified template character will be written out in the following HTML form format. (Because these AV pairs are not "INPUT type=hidden," they are directly displayed on the web page for debugging templates or the SDP process.)

  <INPUT type=hidden NAME="attribute string here"

  value="variable string here">

  all HTML templates should have this!

  $d = dump all av pairs in: attribute = value

URL Template Expansion Rules

There are URLs for the configuration template source, the file template source, and the file destination. These variables are expanded when the registrar prepares the URL, just before retrieving the configuration or file. For the file destination, these variables are expanded just before the petitioner copies the file to the file destination.

- $$—$
- $h—Hostname

Custom Configuration and File Template Variable Expansion Rules

Custom configuration and file template variables are expanded both when the registrar prepares the configuration or file template and when the petitioner receives the configuration or file template.

Custom Configuration and File Template Variable Expansion Rules at the Registrar

When the registrar expands the configuration or file template, the following variables are used by the Cisco IOS CA. These variables are expanded before being sent through the SDP wizard.

- $$—$
- $h—Hostname
- $t—A simple default trustpoint configuration that includes $l, $k, and $s to be expanded at the client
- $1 to $9—Variables retrieved from AAA server during user authentication (not applicable to the file template)

Custom Configuration and File Template Variable Expansion Rules at the Petitioner

When the petitioner expands the configuration or file template, the following variables are expanded:

- $$—$
- $h—Hostname
- $k—Keylabel
- $l—Trustpoint label
- $s—Key size
- $c—Expanded to certificate chain
- $n—Expanded to username (not applicable to the file template)

Custom Configuration HTTP Template Variable Expansion Rules

Custom configuration HTTP templates provide flexibility for backend Common Gateway Interface (CGI) scripts and integration with external management systems. Template URLs run through the HTTP template expansions before registrar retrieves the bootstrap configuration from the external management system. The device name ($n) is expanded into the URL and passed to the external management system so that a specific bootstrap configuration file can be located based on the device information.

Note You should only modify the HTML text that is displayed. The existing expansion variables, Javascript, and forms in the default templates should not be removed when customizing the templates. They are required for SDP to function properly.

The HTTP template expansion and **template config** command allow you to specify either of the following file types to obtain a customized bootstrap configuration file:

- A configuration file based on the device name (for example, template config http://myserver/$n-config-file.conf*)*
- A CGI script based on the device name (for example, template config http://myserver/cgi-bin/mysdpcgi post)

As of Cisco IOS Release 12.4(6)T, the CGI support has been expanded so that the bootstrap configuration can be identified by not only the device name, but also the type, current Cisco IOS version information, and current configuration. This functionality expands the **template config** command with the **post** keyword, which tells the registrar to send this additional device information to the external management system via a CGI script with the HTTP or HTTPS protocol only.

The registrar passes the device information via AV pairs ($a) to the external management system. Using the AV pair information, the management system identifies the appropriate bootstrap configuration files and sends it back to the registrar. The additional AV pairs that are sent with the expanded CGI support for identification of customized bootstrap configuration file are shown in .

Table 1 AV Pairs Sent During HTTP Post to External Management System

AV Pair	Description
TTIFixSubjectName	AAA_AT_TTI_SUBJECTNAME (sent only if the realm authentication user is not the root user on the registrar)
TTIIosRunningConfig	Output of **show running-config brief**
TTIKeyHash	Digest calculated over the device public key

Table 1 AV Pairs Sent During HTTP Post to External Management System (continued)

AV Pair	Description
TTIPrivilege	AAA_AT_TTI_PRIVILEGE—"admin" is sent if the user is an administrator, "user" is sent if the user is not an administrator (sent only if the realm authentication user is an administrator and the information is available from the AAA server)
TTISignature	Digest calculated over all AV pairs except UserDeviceName and TTISignCert
TTISignCert	Device current certificate (sent only if the device currently has a certificate)
TTITemplateVar	AAA_AT_TTI_IOSCONFIG(1-9) (sent only if the realm authentication user is not the root user on the registrar)
TTIUserName	Device name
TTIVersion	TTI version of the registrar
UserDeviceName	Device name as entered by the administrative introducer (sent only if the realm authentication user is an administrator)

Note The registrar must be running Cisco IOS Release 12.4(6)T, the **template config** command must be issued with the **post** keyword, and the *url* argument must include either HTTP or HTTPS. No other protocol is supported for the expanded CGI template functionality (for example, FTP).

Default Templates for SDP Transaction Web Pages

The following default templates exist for each SDP transaction web page:

- Default Prep-Connect Template
- Default Start Page Template
- Default Welcome Page Template
- Default Introduction Page Template
- Default Admin-Introduction Page Template
- Default Completion Page Template

Default Prep-Connect Template

The prep-connect template may be modified by the administrator to contain values that are appropriate for their environment. The format of the prep-connect page may also be modified by the settings contained in the template.

Except for the registrar IP address, which the administrator must customize, the prep-connect template may be used as shown below.

```
<html><head><title>
SDP: Test Internet Connection</title></head>
<noscript><b>
If you see this message, your browser is not running JavaScript,<br>
which is required by Cisco Secure Device Provisioning.<br>
If you cannot enable JavaScript, please contact your system administrator.
<br><br></b></noscript>
<body style="background-color: rgb(204, 255, 255);">
```

```
<div style="text-align: center;"><big><big>
Secure Device Provisioning</big><br>
Test Internet Connection</big><br><br>

<form action="http://10.10.10.1/ezsdd/connect" method="post">
<input type="submit" value="Log onto Cisco Device"><br><br>
Default username/password is cisco/cisco.
<input type="hidden" name="TTIAfterConnectURL"
value="http://10.10.10.1/ezsdd/welcome">
<!-- Note, that for the below, 198.133.219.25 = www.cisco.com. -->
<input type="hidden" name="TTIConnectTestURL" value="http://198.133.219.25">
<input type="hidden" name="TTIInsideAddr" value="10.10.10.1">
<input type="hidden" name="TTIlanport" value="Vlan1">
<input type="hidden" name="TTIwanport" value="FastEthernet4">
</form></div></body></html>
```

Hidden HTML Form Fields

The hidden HTML form fields communicate initial configuration information to the browser as set by the administrator and are not signed.

Note The term “hidden” refers to the fact that these HTML form fields are not displayed on the prep-connect page to reduce potential confusion to the introducer.

The administrator can set the following hidden HTML form fields in the prep-connect template.

Table 2 Administrator Defined AV Pairs Sent During Prep-Connect Phase

AV Pair	Description
TTIAfterConnectURL	The administrator may set the TTIAfterConnectURL field to either the welcome page URL or the start page URL. The welcome page URL is specified with the factory default petitioner IP address. The connect after URL may be any valid URL if SDP is not going to be used after establishing Internet connectivity.
TTIConnectTestURL	The administrator may set the TTIConnectTestURL field to a valid URL that should be accessible when Internet connectivity is established. The default prep-connect template value is www.cisco.com (198.133.219.25).
TTIInsideAddr	The administrator may set the TTIInsideAddr field to the factory default IP address of the petitioner. For the Cisco 871 ISR, the IP Address is 10.10.10.1.
TTIlanportx	The administrator may set the TTIlanportx field to the LAN interface name of the petitioner platform. This field is used to apply the Cisco IOS connect configuration. For the Cisco 871, the field value is “Vlan1.”
TTIwanport	The administrator may set the TTIwanport field to the WAN interface name of the petitioner. This field is used to apply the Cisco IOS connect configuration. For the Cisco 871, the field value is “FastEthernet4.”

Note The connect template cannot be customized.

Default Start Page Template

```
<html><head><title>EZ-Secure Device Deployment Start page on $h</title></head>
<NOSCRIPT><B>
If you see this message, your browser is not running JavaScript.<BR>
Cisco Secure Device Deployment requires JavaScript.<BR> Please contact
your system adminstrator.<BR><BR></B></NOSCRIPT>
<SCRIPT LANGUAGE="JavaScript">
function submit_to_url(form){
form.action=form.TTIWelcomeURL.value;return true;}</SCRIPT>
<B>Welcome to Cisco Secure Device Deployment Server $h</B> <FORM
action="" method="post" onSubmit="return submit_to_url(this)"> Your
device:<BR> <INPUT type="text" name="TTIWelcomeURL" size=80
value=\"\"><BR><BR> <INPUT type="submit" value="Next>"><BR>
$a</FORM></html>
```

Default Welcome Page Template

```
<html><head><title>EZ-Secure Device Deployment WELCOME to $h</title></head>
<NOSCRIPT><B>
If you see this message, your browser is not running JavaScript.<BR>
Cisco Secure Device Deployment requires JavaScript.<BR> Please contact
your system adminstrator.<BR><BR></B></NOSCRIPT>
<SCRIPT LANGUAGE=\"JavaScript\">
function submit_to_url(form){
natURL=location.href.split(\"/\");
localURL=form.TTICompletionURL.value.split(\"/\");
if(natURL[2]!=localURL[2]){
form.TTICompletionURL.value=localURL[0]+\"//\"+natURL[2]+\"/
\"+localURL[3]+
\"/\"+localURL[4];}
form.action=form.vpnserviceurl.value;
return true;}</SCRIPT>
<B>Welcome to Cisco Secure Device Deployment for $h</B> <FORM
action=\"\" method=\"post\" onSubmit=\"return submit_to_url (this)\">
To join a Virtual Private Network (VPN) enter the web<BR> site URL
provided by your network administrator:<BR> <INPUT type=\"text\"
name=\"vpnserviceurl\" size=80 value=\"\"><BR><BR><INPUT
type=\"submit\" value=\"Next>\"><BR> $a</FORM></html>
```

Default Introduction Page Template

```
<html><head><title>EZ-Secure Device Deployment INTRODUCTION to $h</title>
</head><B>Welcome to the VPN network gateway on $h</B> <FORM
action=\"$u\" method=\"post\"> Your 'username' and 'password' entered
have been accepted.<BR> Your device will now be allowed to
automatically join the VPN network.<BR> <BR>Press Next to complete
automatic configuration of your VPN Device.<BR> <BR><INPUT
type=\"submit\" value=\"Next>\"><BR> $a</P></FORM></html>
```

Default Admin-Introduction Page Template

```
<html><head><title>EZ-Secure Device Deployment ADMINISTRATIVE
INTRODUCTION to $h</title></head> <NOSCRIPT><B> If you see this
message, your browser is not running JavaScript.<BR> Cisco Secure
Device Deployment requires JavaScript.<BR> Please contact your system
adminstrator.<BR><BR></B></NOSCRIPT>
<SCRIPT LANGUAGE=\"JavaScript\">
function submit_to_url(form){
form.introadminurl.value=location.href+\"/admin\";
form.action=form.introadminurl.value;
return true;}</SCRIPT>
```

```
<B>Welcome to the VPN network gateway on $h</B> <FORM action=\"\"
method=\"post\" onSubmit=\"return submit_to_url (this)\"> Your
administrator 'username' and 'password' entered have been
accepted.<BR> Please provide the name to be associated with this
device:<BR> <INPUT type=\"text\" name=\"userdevicename\" size=64
value=\"\"><BR><BR> <INPUT type=\"submit\" value=\"Next>\"><BR> <INPUT
type=\"hidden\" name=\"introadminurl\" value=\"\"><BR>
$a</FORM></html>
```

Default Completion Page Template

```
<html><head><title>EZ-Secure Device Deployment COMPLETE on $h</title></head>
<B>Now enrolling $h with the VPN network...</B><BR> Full network VPN
access should be available in a moment.<BR><BR> $d<BR></html>
```

Default Template for the Configuration File

The default configuration template is shown below. This default configuration file is used if a configuration template is not specified or if the **template config** command is issued *without* the **post** keyword. For more information on using the default configuration template, see the section "Using a Configuration Template File: Example."

```
$t
!
$c

!
end
```

How to Set Up SDP for a PKI

This section contains the following procedures that should be followed when setting up SDP for your PKI. You can configure the registrar according to only one of the registrar configuration tasks.

- Enabling the SDP Petitioner, page 21
- Enabling the SDP Registrar and Adding AAA Lists to the Server, page 23
- Enabling the SDP Registrar for Certificate-Based Authorization, page 27
- Configuring an Administrative Introducer, page 29
- Configuring Custom Templates, page 32

Enabling the SDP Petitioner

Perform this task to enable or disable the petitioner and associate a trustpoint with the SDP exchange.

You can also use this task to configure the petitioner to use a certificate and the Rivest, Shamir, and Adelman (RSA) keys associated with a specific trustpoint.

Note The petitioner is enabled by default on a Cisco device that contains a crypto image; thus, you have only to issue the **crypto provisioning petitioner** command if you have previously disabled the petitioner or if you want to use an existing trustpoint instead of the automatically generated trustpoint.

Note By default, the SDP petitioner device uses an existing certificate. If multiple certificates and one specific certificate exist, use this task to make a choice. However, this task is not necessary to enable the default behavior.

Prerequisites

- The HTTP server must be enabled via the **ip http server** command. (The HTTP server is typically enabled by default on many default Cisco IOS configurations.)
- If you are configuring the petitioner to use a certificate and RSA keys, your SDP petitioner device must have an existing manufacturer's or a third-party certificate.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **crypto provisioning petitioner**
4. **trustpoint** *trustpoint-label*

   or

   trustpoint signing *trustpoint-label*
5. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**crypto provisioning petitioner** **Example:** `Router(config)# crypto provisioning petitioner`	Allows SDP petitioner device behavior to be modified and enters tti-petitioner configuration mode. **Note** Effective with Cisco IOS Release 12.3(14)T, the **crypto provisioning petitioner** command replaced the **crypto wui tti petitioner** command.

	Command or Action	Purpose

Step 4	**trustpoint** *trustpoint-label*
	
	Example:
	`Router(tti-petitioner)# trustpoint mytrust`
	
	or
	
	trustpoint signing *trustpoint-label*
	
	Example:
	`Router(tti-petitioner)# trustpoint signing mytrust`
		(Optional) Specifies the trustpoint that is to be associated with the SDP exchange between the petitioner and the registrar.
		
		Note If this command is not issued, the *trustpoint-label* argument is automatically labeled "tti."
		
		(Optional) Specifies the trustpoint and associated certificate that are used when signing all introduction data during the SDP exchange.

Step 5	**end** **Example:** `Router(tti-petitioner)# end`	(Optional) Exits tti-petitioner configuration mode.

Troubleshooting Tips

After the SDP exchange is complete, a new trustpoint-label named "tti" will exist. The trustpoint will be automatically enrolled with the certificate server (the registrar). To verify that the trustpoint is really there, use the **show running-config** command.

What to Do Next

If you set up the petitioner to use a certificate and the RSA keys associated with the specified trustpoint, you should configure the registrar as shown in the task "Enabling the SDP Registrar for Certificate-Based Authorization."

Enabling the SDP Registrar and Adding AAA Lists to the Server

Perform this task to enable the registrar and associate a certificate server with the SDP exchange.

You can also use this task if you want to add an authentication list and an authorization list to the RADIUS or TACACS+ server.

Prerequisites

Before configuring an registrar, ensure the following tasks are complete:

- Enable the HTTP server or the HTTPS server.

Note It is recommended that you issue the **ip http secure-server** command to enable the HTTPS web server. If you enable a secure server, you should issue the **ip http secure-trustpoint** command. You must disable the standard HTTP server via the **no ip http server** command (if the standard server is enabled). The specified trustpoint is a registrar local trustpoint appropriate for HTTPS communication between the registrar and the user's browser.

- Configure the Cisco IOS certificate server (via the **crypto pki server** command).

If you are configuring AAA lists, you should complete the prerequisites required for the registrar in addition to completing the following tasks:

- Add user information to the AAA server database. To configure a RADIUS or TACACS+ AAA server, see the "Configuring RADIUS" and "Configuring TACACS+" chapters of the *Cisco IOS Security Configuration Guide*.
- Configure new AAA lists. To configure AAA lists, see the following chapters in the *Cisco IOS Security Configuration Guide*: "Configuring RADIUS," "Configuring TACACS+," "Configuring Authentication," and "Configuring Authorization."

Restrictions

Cisco IOS CA Device Requirement

During the SDP process, a Cisco IOS CA certificate is automatically issued to the peer device. If an SDP registrar is configured on a third-party vendor's CA device, the SDP process will not work.

The template config Command

There are nine Cisco IOS configuration variables. If you require more configuration flexibility, the **template config** command can be used to reference a configuration template that is specific to the introducer. For more information on configuration flexibility, see the "Custom Configuration and File Template Variable Expansion Rules" section.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **crypto provisioning registrar**
4. **pki-server** *label*
5. **authentication list** *list-name*
6. **authorization list** *list-name*
7. **template username** *name* [**password** *password*]
8. **template config** *url* [**post**]
9. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**crypto provisioning registrar** **Example:** `Router(config)# crypto provisioning registrar`	Configures a device to become a registrar for the SDP exchange and enters tti-registrar configuration mode. **Note** Effective with Cisco IOS Release 12.3(14)T, the **crypto provisioning registrar** command replaced the **crypto wui tti registrar** command.
Step 4	**pki-server** *label* **Example:** `Router(tti-registrar)# pki-server mycs`	Specifies the certificate server that is to be associated with the SDP exchange between the petitioner and the registrar.
Step 5	**authentication list** *list-name* **Example:** `Router (tti-registrar)# authentication list authen-tac`	(Optional) Authenticates the introducer in an SDP exchange.
Step 6	**authorization list** *list-name* **Example:** `Router (tti-registrar)# authorization list author-rad`	(Optional) Receives the appropriate authorized fields for the certificate subject name and list of template variables to be expanded into the Cisco IOS CLI snippet that is sent back to the petitioner.
Step 7	**template username** *name* [**password** *password*] **Example:** `Router(tti-registrar)# template username ftpuser password ftppwd`	(Optional) Establishes a username and password in which to access the configuration template on the file system.

	Command or Action	Purpose
Step 8	**template config** *url* [**post**] **Example:** `Router(tti-registrar)# template config http://myserver/cgi-bin/mycgi post`	(Optional) Specifies a remote URL for the Cisco IOS CLI configuration template. The *url* can reference a configuration file allows you to specify the device name ($n) to identify a bootstrap configuration. CGI support allows you to reference a CGI script via HTTP or HTTPS and identify the bootstrap configuration by not only the device name, but also by the type, current Cisco IOS version information, and current configuration. The **post** keyword must be used for CGI support. **Note** The registrar must be running Cisco IOS Release 12.4(6)T or later to utilize expanded CGI support. If the registrar is running an earlier version of Cisco IOS, the additional device identification information will be ignored.
Step 9	**end** **Example:** `Router(tti-registrar)# end`	(Optional) Exits tti-registrar configuration mode.

Examples

To help troubleshoot the SDP transaction, you can issue the **debug crypto provisioning** command, which displays output from the petitioner and registrar devices.

The following is output for the **debug crypto provisioning** command. The output from the petitioner and registrar devices are shown below.

```
Petitioner device
! The user starts the Welcome phase.
Nov  7 03:15:48.171: CRYPTO_PROVISIONING: received welcome get request.
! The router generates a Rivest, Shamir, and Adelman (RSA) keypair for future enrollment.
Nov  7 03:15:48.279: CRYPTO_PROVISIONING: keyhash 'A506BE3B83C6F4B4A6EFCEB3D584AACA'
! The TTI transaction is completed.
Nov  7 03:16:10.607: CRYPTO_PROVISIONING: received completion post request.

Registrar device
!. During the introduction phase, the browser prompts for login information.
06:39:18: CRYPTO_PROVISIONING: received introduction post request.
06:39:18: CRYPTO_PROVISIONING: checking AAA authentication (ipsecca_script_aaalist,
ttiuser)
! This happens if the user types in the wrong username or password.
06:39:19: CRYPTO_PROVISIONING: authentication declined by AAA, or AAA server not found -
0x3
06:39:19: CRYPTO_PROVISIONING: aaa query fails!
! The user re-enters login information.
06:39:19: CRYPTO_PROVISIONING: received introduction post request.
06:39:19: CRYPTO_PROVISIONING: checking AAA authentication (ipsecca_script_aaalist,
ttiuser)
06:39:20: CRYPTO_PROVISIONING: checking AAA authorization (ipsecca_script_aaalist,
ttiuser)
! The login attempt succeeds and authorization information is retrieved from the AAA
database.
06:39:21: CRYPTO_PROVISIONING: aaa query ok!
```

```
! These attributes are inserted into the configuration template.
06:39:21: CRYPTO_PROVISIONING: building TTI av pairs from AAA attributes
06:39:21: CRYPTO_PROVISIONING: "subjectname" = "CN=user1, O=company, C=US"
06:39:21: CRYPTO_PROVISIONING: "$1" = "ntp server 10.3.0.1"
06:39:21: CRYPTO_PROVISIONING: "$2" = "hostname user1-vpn"
! The registrar stores this subject name and overrides the subject name in the subsequent
enrollment request.
06:39:21: CRYPTO_PROVISIONING: subjectname=CN=user1, O=company, C=US
! The registrar stores this key information so that it may be used to automatically grant
the subsequent enrollment request.
06:39:21: CRYPTO_PROVISIONING: key_hash=A506BE3B83C6F4B4A6EFCEB3D584AACA
```

Enabling the SDP Registrar for Certificate-Based Authorization

Perform this task to enable the SDP registrar to perform the following functions:

- Verify the petitioner-signing certificate using a specified trustpoint or any configured trustpoint.
- Initiate authorization lookups using the introducer username and the certificate name field.

Prerequisites

You must also configure the SDP petitioner to use a certificate and RSA keys associated with a specific trustpoint. To complete this task, use the trustpoint signing command as shown in the task "Enabling the SDP Petitioner."

Restrictions

Because RADIUS does not differentiate between authentication and authorization, you need to use the default password, cisco, for certificate authorization.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **crypto provisioning registrar**
4. **template file** *sourceURL destinationURL*
5. **binary file** *sourceURL destinationURL*
6. **authentication trustpoint** {*trustpoint-label* | **use-any**}
7. **authorization {login | certificate | login certificate}**
8. **authorization username** {**subjectname** *subjectname*}
9. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**crypto provisioning registrar** **Example:** `Router(config)# crypto provisioning registrar`	Configures a device to become an SDP registrar and enters tti-registrar configuration mode.
Step 4	**template file** *sourceURL destinationURL* **Example:** `Router(tti-registrar)# template file http://myserver/registrar_file_r1 http://myserver/petitioner_file_p1`	(Optional) Specifies the source template file location on the registrar and the destination template file location on the petitioner. **Note** This command is useful when using a USB token to provision a device. The template expansion occurs on the registrar for both the source URL and file content. The destination URL is expanded on the petitioner.
Step 5	**binary file** *sourceURL destinationURL* **Example:** `Router(tti-registrar)# binary file http://myserver/registrar_file_a1 http://myserver/petitioner_file_b1`	(Optional) Specifies the binary file location on the registrar and the destination binary file location on the petitioner. **Note** This command is useful when using a USB token to provision a device. Both the source and destination URL are expanded on the registrar. Also, the destination URL and file content are expanded on the petitioner. Binary files are not processed through the template expansion functions.
Step 6	**authentication trustpoint {*trustpoint-label*| use-any}** **Example:** `Router(tti-registrar)# authentication trustpoint mytrust`	(Optional) Specifies the trustpoint used to authenticate the SDP petitioner device's existing certificate. • *trustpoint-label*—Specifies a specific trustpoint. • **use-any**—Specifies any configured trustpoint. **Note** If you do not use this command to specify a trustpoint, the existing petitioner certificate is not validated. (This functionality provides compatibility with self-signed petitioner certificates.)

	Command or Action	Purpose
Step 7	**authorization {login | certificate | login certificate}** **Example:** `Router(tti-registrar)# authorization login certificate`	(Optional) Enables AAA authorization for an introducer or a certificate. • Use the **login** keyword for authorization based on the introducer's username. • Use the **certificate** keyword for authorization based on the petitioner's certificate. • Use the **login certificate** keyword for authorization based on the introducer's username and the petitioner's certificate.
Step 8	**authorization username {subjectname** *subjectname}* **Example:** `Router(tti-registrar)# authorization username subjectname all`	Sets parameters for the different certificate fields that are used to build the AAA username. • The **all** keyword specifies that the entire subject name if the certificate is used as the authorization username.
Step 9	**end** **Example:** `Router(tti-registrar)# end`	(Optional) Exits tti-registrar configuration mode.

Configuring an Administrative Introducer

Perform the following task to configure an administrative introducer using administrator authentication and authorization lists.

Prerequisites

The administrative introducer must have enable privileges on the client device and administrator privileges on the server.

Restrictions

When using RADIUS, a user/device that needs to be introduced by the administrative introducer must always use cisco as its own password. TACACS+ does not have this limitation; a user/device can have any password and be introduced by the administrative introducer.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **crypto provisioning registrar**
4. **administrator authentication list** *list-name*
5. **administrator authorization list** *list-name*
6. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**crypto provisioning registrar** **Example:** `Router(config)# crypto provisioning registrar`	Configures a device to become an SDP registrar and enters tti-registrar configuration mode.
Step 4	**administrator authentication list** *list-name* **Example:** `Router(tti-registrar)# administrator authentication list authen-tac`	Configures the AAA list used to authenticate an administrator during an introduction.
Step 5	**administrator authorization list** *list-name* **Example:** `Router(tti-registrar)# administrator authorization list author-tac`	Configures the AAA list used to obtain authorization information for an administrator during an introduction. Information that can be obtained includes the certificate subject name and/or the list of template variables to be expanded into the Cisco IOS CLI snippet that is sent back to the petitioner.
Step 6	**end** **Example:** `Router(tti-registrar)# end`	(Optional) Exits tti-registrar configuration mode.

Examples

The following example from the **show running-config** command allows you to verify that an administrative introducer using administrator authentication and authorization lists have been created:

```
Router# show running-config

Building configuration...

Current configuration : 2700 bytes
!
! Last configuration change at 01:22:26 GMT Fri Feb 4 2005
!
version 12.4
no service pad
service timestamps debug datetime msec
service timestamps log datetime msec
no service password-encryption
!
hostname router
!
```

```
boot-start-marker
boot-end-marker
!
memory-size iomem 5
enable secret 5 $1$tpBS$PXnBDTIDXfX5pWa//1JX20
enable password lab
!
aaa new-model
!
!
!
aaa session-id common
!
resource manager
!
clock timezone GMT 0
ip subnet-zero
no ip routing
!
!
no ip dhcp use vrf connected
!
!
no ip cef
no ip domain lookup
ip domain name company.com
ip host router 10.3.0.6
ip host router.company.com 10.3.0.6
no ip ips deny-action ips-interface
!
no ftp-server write-enable
!
crypto pki server mycs
!
crypto pki trustpoint mycs
 revocation-check crl
 rsakeypair mycs
!
crypto pki trustpoint tti
 revocation-check crl
 rsakeypair tti
!
crypto pki trustpoint mic
 enrollment url http://router:80
 revocation-check crl
!
crypto pki trustpoint cat
 revocation-check crl
!
!
!
crypto pki certificate map cat 10
!
crypto pki certificate chain mycs
 certificate ca 01
crypto pki certificate chain tti
crypto pki certificate chain mic
 certificate 02
 certificate ca 01
crypto pki certificate chain cat
!
crypto provisioning registrar <---------- !SDP registrar device parameters!
 administrator authentication list authen-tac
 administrator authorization list author-tac
```

```
!
no crypto engine onboard 0
username qa privilege 15 password 0 lab
```

Configuring Custom Templates

Perform this task to create and configure custom templates.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **crypto provisioning registrar**
4. **template http start** *URL*
5. **template http welcome** *URL*
6. **template http introduction** *URL*
7. **template http admin-introduction** *URL*
8. **template http completion** *URL*
9. **template http error** *URL*
10. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**crypto provisioning registrar** **Example:** `Router(config)# crypto provisioning registrar`	Configures a device to become an SDP registrar and enters tti-registrar configuration mode.
Step 4	**template http start** *URL* **Example:** `Router(tti-registrar)# template http start tftp://registrar.company.com/start.html`	Directs the TTI registrar to use the custom start page template. **Note** This command is required to use the start page functionality. If this command is not issued, the welcome page will be the initial communication between the introducer and the petitioner.

	Command or Action	Purpose
Step 5	**template http welcome** *URL* **Example:** `Router(tti-registrar)# template http welcome tftp://registrar.company.com/welcome.html`	(Optional) Uses a custom welcome template rather than the default template.
Step 6	**template http introduction** *URL* **Example:** `Router(tti-registrar)# template http introduction tftp://registrar.company.com/intro.html`	(Optional) Uses a custom introduction template rather than the default template.
Step 7	**template http admin-introduction** *URL* **Example:** `Router(tti-registrar)# template http admin-introduction tftp://registrar.company.com/admin-intro.html`	(Optional) Uses a custom admin-introduction template rather than the default template.
Step 8	**template http completion** *URL* **Example:** `Router(tti-registrar)# template http completion tftp://registrar.company.com/completion.html`	(Optional) Uses a custom completion template rather than the default template.
Step 9	**template http error** *URL* **Example:** `Router(tti-registrar)# template http error tftp://registrar.company.com/error.html`	(Optional) Uses a custom error template rather than the default template.
Step 10	**end** **Example:** `Router(tti-registrar)# end`	(Optional) Exits tti-registrar configuration mode.

Examples

The following example shows the use of custom start, introduction, and completion templates:

```
template http start tftp://registrar.company.com/start.html
template http introduction tftp://registrar.company.com/intro.html
template http completion tftp://registrar.company.com/completion.html
```

Configuration Examples for Setting Up a PKI via SDP

This section contains the following configuration examples:

Verifying the SDP Registrar: Example

The following sample output from the **show running-config** command verifies that the certificate server "cs1" was configured and associated with the SDP exchange between the registrar and petitioner:

```
Router# show running-config

Building configuration...

Current configuration : 5902 bytes
!
! Last configuration change at 09:34:44 GMT Sat Jan 31 2004
!
version 12.3
service timestamps debug datetime msec
service timestamps log datetime msec
no service password-encryption
!
hostname pki-36a
!
boot-start-marker
boot-end-marker
!
logging buffered 32768 debugging
no logging console
enable secret 5 $1$b3jz$CKquLGjFIE3AdXA2/Rl9./
enable password lab
!
clock timezone GMT 0
no aaa new-model
ip subnet-zero
!
!
ip cef
ip domain name company.com
ip host msca-root
ip host yni-u10
ip host pki-36a 10.23.2.131
ip host pki-36a.company.com 10.23.2.131
!
!
crypto pki server cs1
 issuer-name CN=company,L=city,C=US
 lifetime crl 336
 lifetime certificate 730
!
crypto pki trustpoint pki-36a
 enrollment url http://pki-36a:80
 ip-address FastEthernet0/0
 revocation-check none
!
crypto pki trustpoint cs1
 revocation-check crl
```

```
 rsakeypair cs1
!
!
crypto pki certificate chain pki-36a
 certificate 03
  308201D0 30820139 A0030201 02020103 300D0609 2A864886 F70D0101 04050030
  34310B30 09060355 04061302 55533114 30120603 55040713 0B205361 6E746120
  4372757A 310F300D 06035504 03130620 696F7363 73301E17 0D303430 31333130
  39333334 345A170D 30363031 33303039 33333434 5A303A31 38301606 092A8648
  86F70D01 09081309 31302E32 332E322E 32301E06 092A8648 86F70D01 09021611
  706B692D 3336612E 63697363 6F2E636F 6D305C30 0D06092A 864886F7 0D010101
  0500034B 00304802 4100AFFA 8F429618 112FAB9D 01F3352E 59DD3D2D AE67E31D
  370AC4DA 619735DF 9CF4EA13 64E4B563 C239C5F0 1578B773 07BED641 A18CA629
  191884B5 61B66ECF 4D110203 010001A3 30302E30 0B060355 1D0F0404 030205A0
  301F0603 551D2304 18301680 141DA8B1 71652961 3F7D69F0 02903AC3 2BADB137
  C6300D06 092A8648 86F70D01 01040500 03818100 67BAE186 327CED31 D642CB39
  AD585731 95868683 B950DF14 3BCB155A 2B63CFAD B34B579C 79128AD9 296922E9
  4DEDFCAF A7B5A412 AB1FC081 09951CE3 08BFFDD9 9FB1B9DA E9AA42C8 D1049268
  C524E58F 11C6BA7F C750320C 03DFB6D4 CBB3E739 C8C76359 CE939A97 B51B3F7F
  3FF;A9D82 9CFDB6CF E2503A14 36D0A236 A1CCFEAE
  quit
 certificate ca 01
  30820241 308201AA A0030201 02020101 300D0609 2A864886 F70D0101 04050030
  34310B30 09060355 04061302 55533114 30120603 55040713 0B205361 6E746120
  4372757A 310F300D 06035504 03130620 696F7363 73301E17 0D303430 31333130
  39333132 315A170D 30373031 33303039 33313231 5A303431 0B300906 03550406
  13025553 31143012 06035504 07130B20 53616E74 61204372 757A310F 300D0603
  55040313 0620696F 73637330 819F300D 06092A86 4886F70D 01010105 0003818D
  00308189 02818100 FC0695AF 181CE90A 1B34B348 BA957178 680C8B51 07802AC3
  BF77B9C6 CB45092E 3C22292D C7D5FFC1 899185A1 FD8F37D5 C44FC206 6D1FA581
  E2264C83 1CC7453E 548C89C6 F3CD25BC 9BFFE7C5 E6653A06 62133950 78BED51B
  49128428 AB237F80 83A530EA 6F896193 F2134B54 D181F059 348AA84B 21EE6D80
  727BF668 EB004341 02030100 01A36330 61300F06 03551D13 0101FF04 05300301
  01FF300E 0603551D 0F0101FF 04040302 0186301D 0603551D 0E041604 141DA8B1
  71652961 3F7D69F0 02903AC3 2BADB137 C6301F06 03551D23 04183016 80141DA8
  B1716529 613F7D69 F002903A C32BADB1 37C6300D 06092A86 4886F70D 01010405
  00038181 00885895 A0141169 3D754EB2 E6FEC293 5BF0A80B E424AA2F A3F59765
  3463AAD1 55E71F0F B5D1A35B 9EA79DAC DDB40721 1344C01E 015BAB73 1E148E03
  9DD01431 A5E2887B 4AEC8EF4 48ACDB66 A6F9401E 8F7CA588 8A4199BB F8A437A0
  F25064E7 112805D3 074A154F 650D09B9 8FA19347 ED359EAD 4181D9ED 0C667C10
  8A7BCFB0 FB
  quit
crypto pki certificate chain cs1
 certificate ca 01
  30820241 308201AA A0030201 02020101 300D0609 2A864886 F70D0101 04050030
  34310B30 09060355 04061302 55533114 30120603 55040713 0B205361 6E746120
  4372757A 310F300D 06035504 03130620 696F7363 73301E17 0D303430 31333130
  39333132 315A170D 30373031 33303039 33313231 5A303431 0B300906 03550406
  13025553 31143012 06035504 07130B20 53616E74 61204372 757A310F 300D0603
  55040313 0620696F 73637330 819F300D 06092A86 4886F70D 01010105 0003818D
  00308189 02818100 FC0695AF 181CE90A 1B34B348 BA957178 680C8B51 07802AC3
  BF77B9C6 CB45092E 3C22292D C7D5FFC1 899185A1 FD8F37D5 C44FC206 6D1FA581
  E2264C83 1CC7453E 548C89C6 F3CD25BC 9BFFE7C5 E6653A06 62133950 78BED51B
  49128428 AB237F80 83A530EA 6F896193 F2134B54 D181F059 348AA84B 21EE6D80
  727BF668 EB004341 02030100 01A36330 61300F06 03551D13 0101FF04 05300301
  01FF300E 0603551D 0F0101FF 04040302 0186301D 0603551D 0E041604 141DA8B1
  71652961 3F7D69F0 02903AC3 2BADB137 C6301F06 03551D23 04183016 80141DA8
  B1716529 613F7D69 F002903A C32BADB1 37C6300D 06092A86 4886F70D 01010405
  00038181 00885895 A0141169 3D754EB2 E6FEC293 5BF0A80B E424AA2F A3F59765
  3463AAD1 55E71F0F B5D1A35B 9EA79DAC DDB40721 1344C01E 015BAB73 1E148E03
  9DD01431 A5E2887B 4AEC8EF4 48ACDB66 A6F9401E 8F7CA588 8A4199BB F8A437A02;
  F25064E7 112805D3 074A154F 650D09B9 8FA19347 ED359EAD 4181D9ED 0C667C10
  8A7BCFB0 FB
  quit
```

```
!
crypto provisioning registrar
 pki-server cs1
!
!
!
crypto isakmp policy 1
 hash md5
!
!
crypto ipsec transform-set test_transformset esp-3des
!
crypto map test_cryptomap 10 ipsec-isakmp
 set peer 10.23.1.10
 set security-association lifetime seconds 1800
 set transform-set test_transformset
 match address 170
!
!
interface Loopback0
 ip address 10.23.2.131 255.255.255.255
 no ip route-cache cef
 no ip route-cache
 no ip mroute-cache
!
interface FastEthernet0/0
 ip address 10.23.2.2 255.255.255.192
 no ip route-cache cef
 no ip route-cache
 no ip mroute-cache
 duplex auto
 speed auto
 crypto map test_cryptomap
!
interface FastEthernet1/0
 no ip address
 shutdown
 duplex auto
 speed auto
!
ip default-gateway 10.23.2.62
ip http server
no ip http secure-server
ip classless
ip route 0.0.0.0 0.0.0.0 10.23.2.62
!
!
access-list 170 permit ip host 10.23.2.2 host 10.23.1.10
dialer-list 1 protocol ip permit
!
!
control-plane
!
!
line con 0
 exec-timeout 0 0
 speed 115200
line aux 0
line vty 0 4
 password lab
 login
!
!
end
```

Verifying the SDP Petitioner: Example

After the SDP exchange is complete, the petitioner will automatically enroll with the registrar and obtain a certificate. The following sample output via the **show running-config** command shows the automatically generated configuration, which verifies that the trustpoint is really there:

```
Router# show running-config

Building configuration...

Current configuration : 4650 bytes
!
! Last configuration change at 09:34:53 GMT Sat Jan 31 2004
!
version 12.3
service timestamps debug datetime msec
service timestamps log datetime msec
no service password-encryption
!
hostname pki-36b
!
boot-start-marker
boot-end-marker
!
logging buffered 32768 debugging
no logging console
enable secret 5 $1$JYgw$060JKXgl6dERLZpU9J3gb.
enable password lab
!
clock timezone GMT 0
no aaa new-model
ip subnet-zero
!
!
ip cef
ip domain name company.com
ip host msca-root
ip host yni-u10
ip host pki-36a 10.23.2.131
ip host pki-36a.company.com 10.23.2.131
!
!
crypto pki trustpoint tti
 enrollment url http://pki-36a.company.com:80
 revocation-check crl
 rsakeypair tti 1024
 auto-enroll 70
!
!
crypto pki certificate chain tti
 certificate 02
  308201FC 30820165 A00302012;02020102 300D0609 2A864886 F70D0101 04050030
  34310B30 09060355 04061302 55533114 30120603 55040713 0B205361 6E746120
  4372757A 310F300D 06035504 03130620 696F7363 73301E17 0D303430 31333130
  39333333 385A170D 30363031 33303039 33333338 5A302231 20301E06 092A8648
  86F70D01 09021611 706B692D 3336622E 63697363 6F2E636F 6D30819F 300D0609
  2A864886 F70D0101 01050003 818D0030 81890281 8100E383 35584B6C 24751E2C
  F4088F06 C00BFECE 84CFF8EB 50D52044 03D14A2B 91E5A260 7D07ED24 DB599D27
  432065D9 0E459248 D7CDC15D 654E2AF6 BA27D79C 23850306 3E96C508 F311D333
  76FDDC9C A810F75C FCD10F1B 9A142F0C 338B6DB3 346D3F24 97A4B15D 0A9504E7
  1F6CB769 85E9F52B FE907AAF 63D54D66 1A715A20 D7DB0203 010001A3 30302E30
  0B060355 1D0F0404 03&#048;205A0 301F0603 551D2304 18301680 141DA8B1 71652961
  3F7D69F0 02903AC3 2BADB137 C6300D06 092A8648 86F70D01 01040500 03818100
```

```
  C5E2DA0E 4312BCF8 0396014F E18B3EE9 6C970BB7 B8FAFC61 EF849568 D546F73F
  67D2A73C 156202DC 7404A394 D6124DAF 6BACB8CF 96C3141D 109C5B0E 46F4F827
  022474ED 8B59D654 F04E31A2 C9AA1152 75A0C455 FD7EEEF5 A505A648 863EE9E6
  C361D9BD E12BBB36 16B729DF 823AD5CC 404CCE48 A4379CDC 67FF6362 0601B950
  quit
 certificate ca 01
  30820241 308201AA A0030201 02020101 300D0609 2A864886 F70D0101 04050030
  34310B30 09060355 04061302 55533114 30120603 55040713 0B205361 6E746120
  4372757A 310F300D 06035504 03130620 696F7363 73301E17 0D303430 31333130
  39333132 315A170D 30373031 33303039 33313231 5A303431 0B300906 03550406
  13025553 31143012 06035504 07130B20 53616E74 61204372 757A310F 300D0603
  55040313 0620696F 73637330 819F300D 06092A86 4886F70D 01010105 0003818D
  00308189 02818100 FC0695AF 181CE90A 1B34B348 BA957178 680C8B51 07802AC3
  BF77B9C6 CB45092E 3C22292D C7D5FFC1 899185A1 FD8F37D5 C44FC206 6D1FA581
  E2264C83 1CC7453E 548C89C6 F3CD25BC 9BFFE7C5 E6653A06 62133950 78BED51B
  49128428 AB237F80 83A530EA 6F896193 F2134B54 D181F059 348AA84B 21EE6D80
  727BF668 EB004341 02030100 01A36330 61300F06 03551D13 0101FF04 05300301
  01FF300E 0603551D 0F0101FF 04040302 0186301D 0603551D 0E041604 141DA8B1
  71652961 3F7D69F0 02903AC3 2BADB137 C6301F06 03551D23 04183016 80141DA8
  B1716529 613F7D69 F002903A C32BADB1 37C6300D 06092A86 4886F70D 01010405
  00038181 00885895 A0141169 3D754EB2 E6FEC293 5BF0A80B E424AA2F A3F59765
  3463AAD1 55E71F0F B5D1A35B 9EA79DAC DDB40721 1344C01E 015BAB73 1E148E03
  9DD01431 A5E2887B 4AEC8EF4 48ACDB66 A6F9401E 8F7CA588 8A4199BB F8A437A0
  F25064E7 112805D3 074A154F 650D09B9 8FA19347 ED359EAD 4181D9ED 0C667C10
  8A7BCFB0 FB
  quit
!
no crypto engine accelerator
!
!
crypto isakmp policy 1
 hash md5
!
!
crypto ipsec transform-set test_transformset esp-3des
!
crypto map test_cryptomap 10 ipsec-isakmp
 set peer 10.23.2.2
 set security-association lifetime seconds 1800
 set transform-set test_transformset
 match address 170
!
!
interface Ethernet0/0
 ip address 10.23.1.10 255.255.255.192
 no ip route-cache cef
 no ip route-cache
 no ip mroute-cache
 half-duplex
 crypto map test_cryptomap
!
interface Ethernet0/1
 no ip address
 shutdown
 half-duplex
!
interface Ethernet0/2
 no ip address
 shutdown
 half-duplex
!
interface Ethernet0/3
 no ip address
 shutdown
```

```
 half-duplex
!
interface Serial1/0
 no ip address
 shutdown
 serial restart-delay 0
!
interface Serial1/1
 no ip address
 shutdown
 serial restart-delay 0
!
interface Serial1/2
 no ip address
 shutdown
 serial restart-delay 0
!
interface Serial1/3
 no ip address
 shutdown
 serial restart-delay 0
!
ip default-gateway 10.23.1.62
ip http server
no ip http secure-server
ip classless
ip route 0.0.0.0 0.0.0.0 10.23.1.62
!
!
access-list 170 permit ip host 10.23.1.10 host 10.23.2.2
dialer-list 1 protocol ip permit
!
!
control-plane
!
!
line con 0
 exec-timeout 0 0
 speed 115200
line aux 0
line vty 0 4
 password lab
 login
!
!
end
```

Adding AAA Lists to a RADIUS or TACACS+ Server: Examples

This section contains the following configuration examples:

TACACS+ AAA Server Database: Example

In the following example, user information has been added to a TACACS+ AAA database. The username is “user1.” The password is “cisco.” Two Cisco IOS configuration template variables are configured for “user1”: iosconfig1 and iosconfig2. The variables will replace $1 and $2 in the configuration template file. The subject name “CN=user1, O=company, C=US” is also configured. This subject name will replace the subject name field in the subsequent enrollment request (PKCS10) that is received from the petitioner device.

```
user = user1
    password = clear "pswd"

    service=tti
        ! The certificate server inserts the following subject name to the certificate.
        set subjectname="CN=user1, O=company, C=US"

        ! Up to nine template variables may be added.
        set iosconfig1="ntp server 10.3.0.1"
        set iosconfig2="hostname user1-vpn"
```

RADIUS AAA Server Database: Example

User information has been added to the RADIUS AAA server database in the following example. The username is “user1.” The password is “cisco.” Two Cisco IOS configuration template variables are configured for “user1”: iosconfig1 and iosconfig2. The variables will replace $1 and $2 in the configuration template file. The subject name “CN=user1, O=company, C=US” is also configured. This subject name will replace the subject name field in the subsequent enrollment request (PKCS10) that is received from the petitioner device.

```
user = user1
    password = clear "pswd"
    radius=company
    reply_attributes=9,1="tti:subjectname=CN=user1, O=company, C=US"
    ! Up to nine template variables may be added.
    9,1="tti:iosconfig1=ntp server 10.3.0.5"
    9,1="tti:iosconfig2=hostname user1-vpn"
```

AAA List on a TACACS+ and a RADIUS AAA Server: Example

The following is a configuration example showing that AAA authentication has been configured on a TACACS+ server and that AAA authorization has been configured on a RADIUS server.

Note Authentication and authorization usually point to the same server.

```
Router(config)# tacacs-server host 10.0.0.48 key cisco
Router(config)# aaa authentication login authen-tac group tacacs+

Router(config)# radius-server host 10.0.1.49 key cisco
Router(config)# aaa authorization network author-rad group radius
```

Using a Configuration Template File: Example

You can use a different configuration template file on the basis of the introducer name. For example, if you have multiple template files for different users, each with the username in the filename, configure the following under the registrar:

```
Router(config)# crypto provisioning registrar
Router (tti-registrar)# pki-server cs1
Router (tti-registrar)# template config tftp://server/config-$n.txt
```

In this example, the default configuration file shown in the section "Default Template for the Configuration File" will be used because the **template config** command does not reference a CGI script.

CGI Script: Example

The following example would execute a CGI script named "mysdpcgi":

```
Router(config)# crypto provisioning registrar
Router (tti-registrar)# pki-server cs1
Router (tti-registrar)# template config tftp://server/cgi-bin/mysdpcgi post
```

The following is an example CGI script, named "mysdpcgi", that would be executed with the example **template config** command above:

```
#!/usr/bin/perl -w

# for debugging use the -debug form
# use CGI (-debug);
use CGI;

# base64 decoding is being used.
use MIME::Base64;

# The following has been commented out, but left for your information.
#
# Reading everything that has been received from stdin and writing it to the debug log to
#see what has been sent from the registrar.
#
# Remember to reset the STDIN pointer so that the normal CGI processing can get the input.
#
# print STDERR "mysdpcgi.cgi dump of stdin:\n";
# if($ENV{'REQUEST_METHOD'} eq "GET"){
#    $input_data = $ENV{'QUERY_STRING'};
# }
# else {
#    $data_length = $ENV{'CONTENT_LENGTH'};
#    $bytes_read = read(STDIN, $input_data, $data_length);
# }
# print STDERR $input_data,"\n";
# exit;

$query = new CGI;
my %av_table;

# A basic configuration file is being sent back, therefore it is being indicated as plain
# text in the command below.
```

```
print $query->header ("text/plain");
print "\n";

# For testing, parameters can be passed in so that the test applications can
# see what has been received.
#
# print STDERR "The following are the raw AV pairs mysdpcgi.cgi received:\n";
# for each $key ($query->param) {
#    print STDERR "! $key is: \n";
#    $value = $query->param($key);
#    print STDERR "! ",$value;
#    print STDERR "! \n";
#}

# The post process AV pairs are identical to those in Cisco IOS and may be used to produce
# AV pair specific configurations as needed.

%av_table = &postprocessavpairs($query->param);

# Decoded values may be written out.
# WARNING: Some error_logs cannot handle the amount of data and will freeze.
# print STDERR "The following are the decoded AV pairs mysdpcgi.cgi received:\n";
# now write the values out
# while ( ($a, $v) = each(%av_table) ) {
#    print STDERR "$a = $v\n";
# }

# Identifying the AV pairs and specifying them in the config.

while ( ($a, $v) = each(%av_table) ) {
    if ($a eq "TTIIosRunningConfig") {
        $search = "hostname ";
        $begin = index($v, $search) + length($search);
        $end = index($v, "\n", $begin);
        $hostname = substr($v, $begin, $end - $begin);
    }
    if ($a eq "TTIIosVersion") {
        $search = "Version ";
        $begin = index($v, $search) + length($search);
        $end = index($v, "(", $begin);
        $version = substr($v, $begin, $end - $begin);
    }
}

print <<END_CONFIG;
!
! Config auto-generated by sdp.cgi
! This is for SDP testing only and is not a real config
!
!
\$t
!
\$c
!
cry pki trust Version-$version-$hostname

! NOTE: The last line of the config must be 'end' with a blank line after the end
# statement.

END_CONFIG
;

# Emulate IOS tti_postprocessavpairs functionality
sub postprocessavpairs {
```

```
    @attributes = @_;

    # Combine any AV pairs that were split apart
    $n = 0; #element index counter
    while ($attributes[$n]) {
    # see if we are at the start of a set
    if ($attributes[$n] =~ m/_0/) {
        # determine base attribute name
        $a = (split /_0/, $attributes[$n])[0];
        # set initial (partial) value
        $v = $query->param($attributes[$n]);

        # loop and pull the rest of the matching
        # attributes's values into v (would be
        # faster if we stop at first non-match)
        $c = $n+1;
        while ($attributes[$c]) {
        if ($attributes[$c] =~ m/$a/) {
            $v = $v.$query->param($attributes[$c]);
        }
        $c++;
        }

        # store in the av hash table
        $av_table{$a} = $v;
    } else {
        # store in hash table if not part of a set
        if ($attributes[$n] !~ m/_\d/) {
        $av_table{$attributes[$n]} = $query->param($attributes[$n]);
        }
    }
    $n++;
    }

    # de-base64 decode all AV pairs except userdevicename
    while ( ($a, $v) = each(%av_table) ) {
        if ($a ne "userdevicename") {
            $av_table{$a} = decode_base64($av_table{$a});
        }
    }

    return %av_table;
}
```

Note A CGI script cannot be executed without using the **post** keyword with the **template config** command in Cisco IOS Release 12.4(6)T or a later release.

Configuring the Petitioner and Registrar for Certificate-Based Authentication: Example

The following examples shows how to configure a petitioner to use the certificate issued by the trustpoint named mytrust:

```
Router# configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.

Router(config)# crypto provisioning petitioner
Router(tti-petitioner)# trustpoint signing mytrust
Router(tti-petitioner)# end
```

The following example shows how to configure a registrar to verify the petitioner-signing certificate and to perform authorization lookups:

```
Router# configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
Router(config)# crypto provisioning registrar
Router(tti-registrar)# authentication trustpoint mytrust
Router(tti-registrar)# authorization login certificate
Router(tti-registrar)# authorization username subjectname all
Router(tti-registrar)# end
```

Configuring an Administrative Introducer Using Authentication and Authorization Lists: Example

The following example shows how to configure an administrative introducer with the authentication list “authen-tac” and the authorization list “author-tac”:

```
Router# configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.

Router(config)# crypto provisioning registrar
Router(tti-registrar)# administrator authentication list authen-tac
Router(tti-registrar)# administrator authorization list author-tac
Router(tti-registrar)# end
```

Additional References

The following sections provide references related to SDP.

Related Documents

Related Topic	Document Title
Certificate enrollment	*"Configuring Certificate Enrollment for a PKI" module*
Certificate server configuration	"Configuring and Managing a Cisco IOS Certificate Server for PKI Deployment" module
PKI AAA integration concepts and configuration tasks	"Configuration Revocation and Authorization of Certificates in a PKI" module
PKI commands: complete command syntax, command mode, defaults, usage guidelines, and examples	*Cisco IOS Security Command Reference*, Release 12.4T
USB token configuration	"Storing PKI Credentials" chapter in the *Cisco IOS Security Configuration Guide*, Release 12.4T. For other 12.4T features about using SDP and USB tokens to deploy PKI credentials, see the Feature Information Table.

Technical Assistance

Description	Link
The Cisco Support website provides extensive online resources, including documentation and tools for troubleshooting and resolving technical issues with Cisco products and technologies. To receive security and technical information about your products, you can subscribe to various services, such as the Product Alert Tool (accessed from Field Notices), the Cisco Technical Services Newsletter, and Really Simple Syndication (RSS) Feeds. Access to most tools on the Cisco Support website requires a Cisco.com user ID and password.	http://www.cisco.com/techsupport

Feature Information for SDP in a PKI

Table 3 lists the features in this module and provides links to specific configuration information. Only features that were introduced or modified in Cisco IOS Release 12.2(1) or a later release appear in the table.

For information on a feature in this technology that is not documented here, see the "Implementing and Managing PKI Features Roadmap."

Not all commands may be available in your Cisco IOS software release. For release information about a specific command, see the command reference documentation.

Use Cisco Feature Navigator to find information about platform support and software image support. Cisco Feature Navigator enables you to determine which Cisco IOS and Catalyst OS software images support a specific software release, feature set, or platform. To access Cisco Feature Navigator, go to http://www.cisco.com/go/cfn. An account on Cisco.com is not required.

Note Table 3 lists only the Cisco IOS software release that introduced support for a given feature in a given Cisco IOS software release train. Unless noted otherwise, subsequent releases of that Cisco IOS software release train also support that feature.

Table 3 Feature Information for SDP in a PKI

Feature Name	Releases	Feature Information
Secure Device Provisioning (SDP) Connect Template	12.5(**1st**)T	This feature provides the ability to configure a device for Internet connectivity through a service provider. The following sections provide information about this feature: • Prerequisites for Setting Up SDP • SDP Overview • How SDP Works • Default Templates for SDP Transaction Web Pages
USB Token and Secure Device Provisioning (SDP) Integration	12.4(15)T	This feature provides the ability to provision remote devices using a USB token as a mechanism to transfer credentials from one network device to a remote device via SDP. The following sections provide information about this feature: • Prerequisites for Setting Up SDP • SDP Leveraging USB Tokens • Enabling the SDP Registrar for Certificate-Based Authorization The following commands were introduced by this feature: **binary file**, **crypto key move rsa**, **template file**. **Note** For other documentation on this topic, see the "Related Documents" section.

Table 3 Feature Information for SDP in a PKI (continued)

Feature Name	Releases	Feature Information
SDP Expanded Template CGI Support	12.4(6)T	This feature allows users to configure the SDP registrar to send a bootstrap configuration to the SDP petitioner based on not only the device name, but also its current Cisco IOS version and current configuration. The following sections provide information about this feature: • SDP Introduction Phase • Custom Configuration and File Template Variable Expansion Rules • Default Template for the Configuration File • Enabling the SDP Registrar and Adding AAA Lists to the Server • CGI Script: Example The following command was modified by this feature: **template config**
Secure Device Provisioning (SDP) Start Page	12.4(4)T	This feature allows users to configure their browsers to begin the TTI transaction by contacting the registrar's introduction URL via a start page. Thus, users no longer have to begin the TTI transaction from the welcome page on the petitioner. The following sections provide information about this feature: • How Custom Templates Work with SDP • Configuring Custom Templates The following commands were introduced by this feature: **template http admin-introduction**, **template http completion**, **template http error**, **template http introduction**, **template http start**, **template http welcome**
Administrative Secure Device Provisioning Introducer	12.3(14)T	This feature allows you to act as an administrative introducer to introduce a device into a PKI network and then provide a username as the device name for the record locator in the AAA database. The following sections provide information about this feature: • Authentication and Authorization Lists for an Administrative Introducer • Configuring an Administrative Introducer The following commands were introduced by this feature: **administrator authentication list**, **administrator authorization list**

Table 3 Feature Information for SDP in a PKI (continued)

Feature Name	Releases	Feature Information
Easy Secure Device Deployment	12.3(8)T	This feature introduces support for SDP, which offers a web-based enrollment interface that enables network administrators to deploy new devices in large networks. The following sections provide information about this feature: • Information About Setting Up SDP for Enrollment in a PKI • Enabling the SDP Registrar and Adding AAA Lists to the Server The following commands were introduced or modified by this feature: **crypto wui tti petitioner**, **crypto wui tti registrar**, **pki-server**, **template config**, **template username**, **trustpoint (tti-petitioner)**
Easy Secure Device Deployment AAA Integration	12.3(8)T	This feature integrates an external AAA database, allowing the SDP introducer to be authenticated against a AAA database instead of having to use the enable password of the local Cisco certificate server. The following sections provide information about this feature: • How SDP Uses an External AAA Database • Enabling the SDP Registrar and Adding AAA Lists to the Server The following commands were introduced or modified by this feature: **authentication list (tti-registrar)**, **authorization list (tti-registrar)**, **debug crypto wui template config**, **template username**
Secure Device Provisioning (SDP) Certificate-Based Authorization	12.3(14)T	This feature allows certificates issued by other authority (CA) servers to be used for SDP introductions. The following sections provide information about this feature: • Use of a Self-Signed Certificate Versus a Certificate Issued by Another CA Server • Enabling the SDP Registrar for Certificate-Based Authorization The following commands were introduced by this feature: **administrator authentication list**, **administrator authorization list**

Any Internet Protocol (IP) addresses used in this document are not intended to be actual addresses. Any examples, command display output, and figures included in the document are shown for illustrative purposes only. Any use of actual IP addresses in illustrative content is unintentional and coincidental.