Storing PKI Credentials

First Published: May 2, 2005
Last Updated: March 11, 2011

This module explains how to store public key infrastructure (PKI) credentials, such as Rivest, Shamir, and Adelman (RSA) keys and certificates in a specific location.

An example of a certificate storage location includes NVRAM, which is the default location, and other local storage locations, such as flash, as supported by your platform.

An example of an RSA key and certificate storage location includes a USB token. Selected Cisco platforms support smart card technology in a USB key form factor (such as an Aladdin USB eToken key). USB tokens provide secure configuration distribution, provide RSA operations such as on-token key generation, signing, and authentication, and allow users to store Virtual Private Network (VPN) credentials for deployment.

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest feature information and caveats, see the release notes for your platform and software release. To find information about the features documented in this module, and to see a list of the releases in which each feature is supported, see the "Feature Information for Storing PKI Credentials" section on page 26.

Use Cisco Feature Navigator to find information about platform support and Cisco 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 Storing PKI Credentials

Prerequisites for Specifying a Local Certificate Storage Location

Before you can specify the local certificate storage location, your system should meet the following requirements:

- A Cisco IOS Release 12.4(2)T PKI-enabled image or a later image
- A platform that supports storing PKI credentials as separate files
- A configuration that contains at least one certificate
- An accessible local file system

Prerequisites for Specifying USB Token Storage for PKI Credentials

Before you can use a USB token, your system should meet the following requirements:

- A Cisco 871 router, Cisco 1800 series, Cisco 2800 series, a Cisco 3800 series router, or a Cisco 7200VXR NPE-G2 platform
- At least a Cisco IOS Release 12.3(14)T image running on any of the supported platforms
- A Cisco supported USB token
- A k9 image

Restrictions for Storing PKI Credentials

Restrictions for Specifying a Local Certificate Storage Location

When storing certificates to a local storage location, the following restrictions are applicable:

- Only local file systems may be used. An error message will be displayed if a remote file system is selected, and the command will not take effect.
- A subdirectory may be specified if supported by the local file system. NVRAM does not support subdirectories.

Restrictions for Specifying USB Token Storage

When using a USB token to store PKI data, the following restrictions are applicable:

- USB token support requires a 3DES (k9) Cisco IOS software image, which provides secure file storage.
- You cannot boot an image from a USB token. (However, you can boot a configuration from a USB token.)
- USB hubs are currently not supported. Thus, the number of supported devices is limited to the number of available USB ports.

Information About Storing PKI Credentials

To determine where to store PKI credentials, you should understand the following concepts:

Storing Certificates to a Local Storage Location

Certificates are stored to NVRAM by default, however some routers do not have the required amount of NVRAM to successfully store certificates. Introduced in Cisco IOS Release 12.4(2)T is the ability to specify where certificates are stored on a local file system.

All Cisco platforms support NVRAM and flash local storage. Depending on your platform, you may have other supported local storage options including bootflash, slot, disk, USB flash, or USB token.

During run time, you can specify what active local storage device you would like to use to store certificates.

PKI Credentials and USB Tokens

To use a secure USB token on your router, you should understand the following concepts:

How a USB Token Works

A smart card is a small plastic card, containing a microprocessor and memory that allows you to store and process data. A USB token is a smart card with a USB interface. The token can securely store any type of file within its available storage space (32 KB). Configuration files that are stored on the USB token can be encrypted and accessed only via a user PIN. The router will not load the configuration file unless the proper PIN has been configured for secure deployment of router configuration files.

After you plug the USB token into the router, you must log into the USB token; thereafter, you can change default settings, such as the user PIN (default: 1234567890) and the allowed number of failed login attempts (default: 15 attempts) before future logins are refused. For more information on accessing and configuring the USB token, see the section "Logging Into and Setting Up the USB Token."

After you have successfully logged into the USB token, you can copy files from the router on to the USB token via the **copy** command. USB token RSA keys and associated IPsec tunnels remain available until the router is reloaded. To specify the length of time before the keys are removed and the IPsec tunnels are torn down, issue the **crypto pki token removal timeout** command. The default timeout is zero, which causes the RSA keys to be removed automatically after the eToken is removed from the router. The default appears in the running configuration as:

```
crypto pki token default removal timeout 0
```

Table 1 highlights the capabilities of the USB token.

Table 1 Functionality Highlights for USB Tokens

Function	USB Token
Accessibility	Used to securely store and transfer digital certificates, preshared keys, and router configurations from the USB token to the router.
Storage Size	32 KB
File Types	• Typically used to store digital certificates, preshared keys, and router configurations for IPsec VPNs. • USB tokens cannot store Cisco IOS images.
Security	• Files can be encrypted and accessed only with a user PIN. • Files can also be stored in a nonsecure format.
Boot Configurations	• The router can use the configuration stored in the USB token during boot time. • The router can use the secondary configuration stored in the USB token during boot time. (A secondary configuration allows users to load their IPsec configuration.)

Benefits of USB Tokens

USB token support on a Cisco router provides the following application benefits:

Removable Credentials: Provide or Store VPN Credentials on an External Device for Deployment

A USB token can use smart card technology to store a digital certificate and configuration for IPsec VPN deployment. This ability enhances the capability of the router to generate RSA public keys to authenticate at least one IPsec tunnel. (Because a router can initiate multiple IPsec tunnels, the USB token can contain several certificates, as appropriate.)

Storing VPN credentials on an external device reduces the threat of compromising secure data.

PIN Configuration for Secure File Deployment

A USB token can store a configuration file that can be used for enabling encryption on the router via a user-configured PIN. (That is, no digital certificates, preshared keys, or VPNs are used.)

Touchless or Low Touch Configuration

The USB token can provide remote software configuration and provisioning with little or no human interaction. Configuration is set up as an automated process. That is, the USB token can store a bootstrap configuration that the router can use to boot from after the USB token has been inserted into the router. The bootstrap configuration connects the router to a TFTP server, which contains a configuration that completely configures the router.

RSA Operations

As of Cisco IOS Release 12.4(11)T and later releases, a USB token may be used as a cryptographic device in addition to a storage device. Using a USB token as a cryptographic device allows RSA operations such as key generation, signing, and authentication to be performed on the token.

General-purpose, special-usage, encryption, or signature RSA key pairs with a modulus of 2048 bits or less may be generated from credentials located on your token storage device. Private keys are not distributed and remain on the token by default, however you may configure the private key storage location.

Keys that reside on a USB token are saved to persistent token storage when they are generated. Key deletion will remove the keys stored on the token from persistent storage immediately. (Keys that do not reside on a token are saved to or deleted from nontoken storage locations when the **write memory** or a similar command is issued.)

Remote Device Configuration and Provisioning in a Secure Device Provisioning (SDP) Environment

As of Cisco IOS Release 12.4(15)T and later releases, SDP may be used to configure a USB token. The configured USB token may be transported to provision a device at a remote location. That is, a USB token may be used to transfer cryptographic information from one network device to another remote network device providing a solution for a staged USB token deployment.

For information about using USB tokens with SDP, see document titles in the "Related Documents" section.

How to Configure PKI Storage

Specifying a Local Storage Location for Certificates

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **crypto pki certificate storage** *location-name*
4. **exit**
5. **copy** *source-url destination-url*
6. **show crypto pki certificates storage**

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 pki certificate storage** *location-name* **Example:** `Router(config)# crypto pki certificate storage flash:/certs`	Specifies the local storage location for certificates.
Step 4	**exit** **Example:** `Router(config)# exit`	Exits global configuration mode.
Step 5	**copy** *source-url destination-url* **Example:** `Router# copy system:running-config nvram:startup-config`	(Optional) Saves the running configuration to the startup configuration. **Note** Settings will only take effect when the running configuration is saved to the startup configuration.
Step 6	**show crypto pki certificates storage** **Example:** `Router# show crypto pki certificates storage`	(Optional) Displays the current setting for the PKI certificate storage location.

Examples

The following is sample output for the **show crypto pki certificates storage** command where the certificates are stored in the certs subdirectory of disk0:

```
Router# show crypto pki certificates storage

Certificates will be stored in disk0:/certs/
```

Setting Up and Using USB Tokens on Cisco Routers

This section contains the following procedures that allow you to configure a router to support USB tokens:

Storing the Configuration on a USB Token

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. boot config usbtoken[0-9]:filename

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	**boot config usbtoken**[**0**-**9**]**:***filename* **Example:** `Router(config)# boot config usbtoken0:file`	Specifies that the startup configuration file is stored in a secure USB token.

Logging Into and Setting Up the USB Token

Perform this task to log into and to perform the initial set up of a USB token.

Use of RSA Keys with a USB Token

- RSA keys are loaded after the USB token is successfully logged into the router.
- By default, newly generated RSA keys are stored on the most recently inserted USB token. Regenerated keys should be stored in the same location where the original RSA key was generated.

Automatic Login

Automatic login allows the router to completely come back up without any user or operator intervention. The PIN is stored in the private NVRAM, so it is not visible in the startup or running configuration.

Note A hand-generated startup configuration can contain the automatic login command for deployment purposes, but the **copy system:running-config nvram: startup-config** command must be issued to put the hand-generated configuration in the private configuration.

Manual Login

Unlike automatic login, manual login requires that the user know the actual USB token PIN.

Manual login can be used when storing a PIN on the router is not desirable. Manual login may also be suitable for some initial deployment or hardware replacement scenarios for which the router is obtained from the local supplier or drop-shipped to the remote site. Manual login can be executed with or without privileges, and it will make files and RSA keys on the USB token available to the Cisco IOS software. If a secondary configuration file is configured, it will be executed only with the privileges of the user who is performing the login. Thus, if you want to use manual login and set up the secondary configuration on the USB token to perform anything useful, you need to enable privileges.

Manual login can also be used in recovery scenarios for which the router configuration has been lost. If the scenario contains a remote site that normally connects to the core network with a VPN, the loss of the configuration and RSA keys requires out-of-band services that the USB token can provide. The USB token can contain a boot configuration, a secondary configuration, or both, and RSA keys to authenticate the connection.

SUMMARY STEPS

1. **enable**
2. **crypto pki token** *token-name* [**admin**] **login** [*pin*]

   or

   configure terminal
3. **crypto pki token** *token-name* **user-pin** [*pin*]
4. exit
5. **show usbtoken**[**0-9**]**:***filename*

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.

Step 2	**crypto pki token** *token-name* [**admin**] **login** [*pin*]
	
	Example:
	`Router# crypto pki token usbtoken0 admin login 5678`
	
	or
	
	configure terminal
	
	Example:
	`Router# configure terminal`
		Manually logs into the USB token.
		You must specify the **admin** keyword if later you want to change the user PIN.
		
		or
		
		Puts the router in global configuration mode, which allows you to configure automatic USB token login.

Step 3	**crypto pki token** *token-name* **user-pin** [*pin*] **Example:** `Router(config)# crypto pki token usbtoken0 user-pin 1234`	(Optional) Configures the router to log into the token automatically, using the specified PIN at router startup or when the USB token is inserted into a USB slot. The PIN is encrypted and stored in NVRAM. **Note** You will be asked to enter your passphrase.
Step 4	**exit** **Example:** `Router(config)# exit`	Exits global configuration mode.
Step 5	**show usbtoken**[**0-9**]**:***filename* **Example:** `Router# show usbtoken0:usbfile`	(Optional) Verifies whether the USB token has been logged onto the router.

What to Do Next

After you have logged into the USB token, it is available for use.

- To further configure the USB token, see the "Configuring the USB Token" section.
- To perform USB token administrative tasks, such as changing the user PIN, copying files from the router to the USB token set key storage location, and changing USB tokens, see the "Setting Administrative Functions on the USB Token" section.
- To utilize the USB token as a cryptographic device to perform RSA operations, see the document titles in the "Related Documents" section.
- To specify that the USB token be used for RSA operations during initial autoenrollment, see the document titles in the "Related Documents" section.

Configuring the USB Token

After you have set up automatic login, you may perform this task to further configure the USB token.

PINs and Passphrases

For additional PIN security with automatic login, you may encrypt your PIN stored in NVRAM and set up a passphrase for your USB token. Establishing a passphrase allows you to keep your PIN secure; another user needs only to know the passphrase, not the PIN.

When the USB token is inserted into the router, the passphrase is needed to decrypt the PIN. Once the PIN is decrypted, the router can then use the PIN to login the USB token.

Note The user has only the access they would normally have and needs only privilege level 1 to log in.

Unlocking and Locking the USB Token

The USB token itself can be locked (encrypted) or unlocked (decrypted).

Unlocking the USB token allows it to be used. Once unlocked, Cisco IOS treats the token as if it were automatically logged in. Any keys on the USB token are loaded, and if a secondary configuration file is on the token, it is executed with full user privileges (privilege level 15) independent of the privilege level of the logged-in user.

Locking the token, unlike logging out of the token, deletes any RSA keys loaded from the token and runs the secondary unconfiguration file, if configured.

Secondary Configuration and Unconfiguration Files

Configuration files that exist on a USB token are called secondary configuration files. If you create and configure a secondary configuration file, it is executed after the token is logged in. The existence of a secondary configuration file is determined by the presence of a secondary configuration file option in the Cisco IOS configuration stored in NVRAM. When the token is removed or logged out and the removal timer expires, a separate secondary unconfiguration file is processed to remove all secondary configuration elements from the running configuration. Secondary configuration and secondary unconfiguration files are executed at privilege level 15 and are not dependent on the level of the user logged in.

SUMMARY STEPS

1. **enable**
2. **crypto pki token** *token-name* **unlock** [*pin*]
3. **configure terminal**
4. **crypto pki token** *token-name* **encrypted-user-pin** [**write**]
5. **crypto pki token** *token-name* **secondary unconfig** *file*
6. **exit**
7. **crypto pki token** *token-name* **lock** [*pin*]

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**crypto pki token** *token-name* **unlock** [*pin*] **Example:** `Router# crypto pki token mytoken unlock mypin`	(Optional) Allows the token to be used if the USB token has been locked. Once unlocked, Cisco IOS treats the token as if it has been automatically logged in. Any keys on the token are loaded and if a secondary configuration file exists, it is executed.
Step 3	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 4	**crypto pki token** *token-name* **encrypted-user-pin** [**write**] **Example:** `Router(config)# crypto pki token mytoken encrypted-user-pin write`	(Optional) Encrypts the stored PIN in NVRAM.
Step 5	**crypto pki token** *token-name* **secondary unconfig** *file* **Example:** `Router(config)# crypto pki token mytoken secondary unconfig configs/myunconfigfile.cfg`	(Optional) Specifies the secondary configuration file and its location.
Step 6	**exit** **Example:** `Router(config)# exit`	Enters privileged EXEC mode.
Step 7	**crypto pki token** *token-name* **lock** [*pin*] **Example:** `Router# crypto pki token mytoken lock mypin`	(Optional) Deletes any RSA keys loaded from the token and runs the secondary unconfiguration file, if it exists.

Examples

The following example shows both the configuration and encryption of a user PIN and then the router reloading and the user PIN being unlocked:

```
! Configuring the user PIN

Enter configuration commands, one per line.  End with CNTL/Z.
Router(config)# crypto pki token usbtoken0: user-pin
     Enter password:

! Encrypt the user PIN
```

```
Router (config)# crypto pki token usbtoken0: encrypted-user-pin
     Enter passphrase:
Router(config)# exit
Router#
Sep 20 21:51:38.076: %SYS-5-CONFIG_I: Configured from console by console
!

Router# show running config
.
.
.
crypto pki token usbtoken0 user-pin *encrypted*
.
.
.

! Reloading the router.
!
Router> enable
 Password:
!
! Decrypting the user pin.
!
Router# crypto pki token usbtoken0: unlock
 Token eToken is usbtoken0
!
Enter passphrase:
Token login to usbtoken0(eToken) successful
Router#
Sep 20 22:31:13.128: %CRYPTO-6-TOKENLOGIN: Cryptographic Token eToken
Login Successful
```

The following example shows a how a secondary unconfiguration file might be used to remove secondary configuration elements from the running configuration. For example, a secondary configuration file might be used to set up a PKI trustpoint. A corresponding unconfiguration file, named mysecondaryunconfigfile.cfg, might contain this command line:

```
no crypto pki trustpoint token-tp
```

If the token were removed and the following commands executed, the trustpoint and associated certificates would be removed from the router's running configuration:

```
Router# configure terminal
Router(config)# no crypto pki token mytoken secondary unconfig mysecondaryunconfigfile.cfg
```

What to Do Next

After you have logged into and configured the USB token, it is available for use.

- To perform USB token administrative tasks, such as changing the user PIN, copying files from the router to the USB token set key storage location, and changing USB tokens, see the "Setting Administrative Functions on the USB Token" section.
- To utilize the USB token as a cryptographic device to perform RSA operations, see the document titles in the "Related Documents" section.

- To specify that the USB token be used for RSA operations during initial autoenrollment, see the document titles in the "Related Documents" section.

Setting Administrative Functions on the USB Token

Perform this task to change default settings, such as the user PIN, the maximum number of failed attempts on the USB token, or the credential storage location.

SUMMARY STEPS

1. **enable**
2. **crypto pki token** *token-name* [**admin**] **change-pin** [*pin*]
3. **crypto pki token** *token-name device:* **label** *token-label*
4. **configure terminal**
5. **crypto key storage** *device*:
6. **crypto key generate rsa** [**general-keys** | **usage-keys** | **signature** | **encryption**] [**label** *key-label*] [**exportable**] [**modulus** *modulus-size*] [**storage** *devicename:*] [**on** *devicename:*]
7. **crypto key move rsa** *keylabel* [**non-exportable**] [**on** | **storage**] *location*
8. **crypto pki token** {*token-name* | **default**} **removal timeout** [*seconds*]
9. **crypto pki token** {*token-name* | **default**} **max-retries** [*number*]
10. **exit**
11. **copy usbflash**[**0-9**]**:***filename destination-url*
12. **show usbtoken**[**0-9**]**:***filename*
13. **crypto pki token** *token-name* **logout**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**crypto pki token** *token-name* [**admin**] **change-pin** [*pin*] **Example:** `Router# crypto pki token usbtoken0 admin change-pin`	(Optional) Changes the user PIN number on the USB token. • If the PIN is not changed, the default PIN—1234567890—will be used. **Note** After the PIN has been changed, you must reset the login failure count to zero (via the **crypto pki token max-retries** command). The maximum number of allowable login failures is set (by default) to 15.
Step 3	**crypto pki token** *token-name device:* **label** *token-label* **Example:** `Router# crypto pki token my token usb0: label newlabel`	(Optional) Sets or changes the name of the USB token. • The value of the *token-label* argument may be up to 31 alphanumeric characters in length including dashes and underscores. **Tip** This command is useful when configuring multiple USB tokens for automatic login, secondary configuration files, or other token specific settings.
Step 4	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 5	**crypto key storage** *device*: **Example:** `Router(config)# crypto key storage usbtoken0:`	(Optional) Sets the default RSA key storage location for newly created keys. **Note** Regardless of configuration settings, existing keys are stored on the device from where they were originally loaded.

	Command or Action	Purpose
Step 6	**crypto key generate rsa** [**general-keys** | **usage-keys** | **signature** | **encryption**] [**label** *key-label*] [**exportable**] [**modulus** *modulus-size*] [**storage** *devicename:*] [**on** *devicename:*] **Example:** `Router(config)# crypto key generate rsa label tokenkey1 storage usbtoken0:`	(Optional) Generates the RSA key pair for the certificate server. • The **storage** keyword specifies the key storage location. • When specifying a label name by specifying the *key-label* argument, you must use the same name for the label that you plan to use for the certificate server (through the **crypto pki server** *cs-label* command). If a *key-label* argument is not specified, the default value, which is the fully qualified domain name (FQDN) of the router, is used. If the exportable RSA key pair is manually generated after the CA certificate has been generated, and before issuing the **no shutdown** command, then use the **crypto ca export pkcs12** command to export a PKCS12 file that contains the certificate server certificate and the private key. • By default, the modulus size of a CA key is 1024 bits. The recommended modulus for a CA key is 2048 bits. The range for a modulus size of a CA key is from 350 to 4096 bits. • The **on** keyword specifies that the RSA key pair is created on the specified device, including a Universal Serial Bus (USB) token, local disk, or NVRAM. The name of the device is followed by a colon (:). **Note** Keys created on a USB token must be 2048 bits or less.
Step 7	**crypto key move rsa** *keylabel* [**non-exportable**] [**on** | **storage**] *location* **Example:** `Router(config)# crypto key move rsa keypairname non-exportable on token`	(Optional) Moves existing Cisco IOS credentials from the current storage location to the specified storage location. By default, the RSA key pair remains stored on the current device. Generating the key on the router and moving it to the token takes less than a minute. Generating a key on the token, using the **on** keyword could take five to ten minutes, and is dependent on hardware key generation routines available on the USB token. When an existing RSA key pair is generated in Cisco IOS, stored on a USB token, and used for an enrollment, it may be necessary to move those existing RSA key pairs to an alternate location for permanent storage. This command is useful when using SDP with USB tokens to deploy credentials.

	Command or Action	Purpose
Step 8	**crypto pki token** {*token-name* | **default**} **removal timeout** [*seconds*] **Example:** `Router(config)# crypto pki token usbtoken0 removal timeout 60`	(Optional) Sets the time interval, in seconds, that the router will wait before removing the RSA keys that are stored in the USB token after the USB token has been removed from the router. **Note** If this command is not issued, all RSA keys and IPsec tunnels associated with the USB token are torn down immediately after the USB token is removed from the router.
Step 9	**crypto pki token** {*token-name* | **default**} **max-retries** [*number*] **Example:** `Router(config)# crypto pki token usbtoken0 max-retries 20`	(Optional) Sets the maximum number of consecutive failed login attempts allowed before access to the USB token is denied. • By default, the value is set at 15.
Step 10	**exit** **Example:** `Router(config)# exit`	Exits global configuration mode.
Step 11	**copy** usbflash[**0-9**]:*filename destination-url* **Example:** `Router# copy usbflash0:file1 nvram:`	Copies files from USB token to the router. • *destination-url*—See the **copy** command page documentation for a list of supported options.
Step 12	**show usbtoken**[**0-9**]:*filename* **Example:** `Router# show usbtoken:usbfile`	(Optional) Displays information about the USB token. You can use this command to verify whether the USB token has been logged onto the router.

Step 13	`crypto pki token token-name logout`
	
	Example:
	`Router# crypto pki token usbtoken0 logout`
		Logs the router out of the USB token.
		
		Note If you want to save any data to the USB token, you must log back into the token.

Troubleshooting USB Tokens

This section contains descriptions of the following Cisco IOS commands that can be used to help troubleshoot possible problems that may arise while using a USB token:

The show file systems Command

Use the **show file systems** command to determine whether the router recognizes that there is a USB module plugged into a USB port. The USB module should appear on the list of file systems. If the module does not appear on the list, it can indicate any of the following problems:

- A connection problem with the USB module.
- The Cisco IOS image running on the router does not support a USB module.
- A hardware problem with the USB module itself.

SUMMARY STEPS

1. **show file systems**

DETAILED STEPS

Step 1 Sample output from the **show file systems** command showing a USB token appears below. The USB module listing appears in the last line of the examples.

```
Router# show file systems

File Systems:

     Size(b)     Free(b)      Type  Flags  Prefixes
           -           -    opaque     rw   archive:
           -           -    opaque     rw   system:
           -           -    opaque     rw   null:
           -           -   network     rw   tftp:
*  129880064    69414912      disk     rw   flash:#
      491512      486395     nvram     rw   nvram:
           -           -    opaque     wo   syslog:
           -           -    opaque     rw   xmodem:
           -           -    opaque     rw   ymodem:
           -           -   network     rw   rcp:
           -           -   network     rw   pram:
           -           -   network     rw   ftp:
           -           -   network     rw   http:
           -           -   network     rw   scp:
           -           -   network     rw   https:
           -           -    opaque     ro   cns:
    63158272    33037312  usbflash     rw   usbflash0:
       32768         858  usbtoken     rw   usbtoken1:
```

The show usb device Command

Use the **show usb device** command to determine if a USB token is supported by Cisco.

SUMMARY STEPS

1. **show usb device**

DETAILED STEPS

Step 1 The following sample output for the **show usb device** command indicates whether or not the module is supported is bold in the sample output below:

```
Router# show usb device

Host Controller:1
Address:0x11
Device Configured:YES
Device Supported:YES
Description:eToken Pro 4254
Manufacturer:AKS
Version:1.0
Serial Number:
Device Handle:0x1010000
USB Version Compliance:1.0
Class Code:0xFF
Subclass Code:0x0
Protocol:0x0
Vendor ID:0x529
Product ID:0x514
Max. Packet Size of Endpoint Zero:8
Number of Configurations:1
Speed:Low
Selected Configuration:1
Selected Interface:0

Configuration:
    Number:1
    Number of Interfaces:1
    Description:
    Attributes:None
    Max Power:60 mA

    Interface:
        Number:0
        Description:
        Class Code:255
        Subclass:0
        Protocol:0
        Number of Endpoints:0
```

The show usb controllers Command

Use the **show usb controllers** command to determine if there is a hardware problem with a USB flash module. If the **show usb controllers** command displays an error, the error indicates a hardware problem in the USB module.

You can also use the **show usb controllers** command to verify that copy operations onto a USB flash module are occurring successfully. Issuing the **show usb controllers** command after performing a file copy should display successful data transfers.

SUMMARY STEPS

1. **show usb controllers**

DETAILED STEPS

Step 1 The following sample output for the **show usb controllers** command displays a working USB flash module:

```
Router# show usb controllers

Name:1362HCD
Controller ID:1
Controller Specific Information:
    Revision:0x11
    Control:0x80
    Command Status:0x0
    Hardware Interrupt Status:0x24
    Hardware Interrupt Enable:0x80000040
    Hardware Interrupt Disable:0x80000040
    Frame Interval:0x27782EDF
    Frame Remaining:0x13C1
    Frame Number:0xDA4C
    LSThreshold:0x628
    RhDescriptorA:0x19000202
    RhDescriptorB:0x0
    RhStatus:0x0
    RhPort1Status:0x100103
    RhPort2Status:0x100303
    Hardware Configuration:0x3029
    DMA Configuration:0x0
    Transfer Counter:0x1
    Interrupt:0x9
    Interrupt Enable:0x196
    Chip ID:0x3630
    Buffer Status:0x0
    Direct Address Length:0x80A00
    ATL Buffer Size:0x600
    ATL Buffer Port:0x0
    ATL Block Size:0x100
    ATL PTD Skip Map:0xFFFFFFFF
    ATL PTD Last:0x20
    ATL Current Active PTD:0x0
    ATL Threshold Count:0x1
    ATL Threshold Timeout:0xFF

Int Level:1
Transfer Completion Codes:
         Success              :920             CRC              :0
         Bit Stuff            :0               Stall            :0
         No Response          :0               Overrun          :0
         Underrun             :0               Other            :0
         Buffer Overrun       :0               Buffer Underrun :0
Transfer Errors:
         Canceled Transfers   :2               Control Timeout :0
Transfer Failures:
         Interrupt Transfer   :0               Bulk Transfer   :0
         Isochronous Transfer :0               Control Transfer:0
Transfer Successes:
         Interrupt Transfer   :0               Bulk Transfer   :26
         Isochronous Transfer :0               Control Transfer:894

USBD Failures:
         Enumeration Failures :0               No Class Driver Found:0
         Power Budget Exceeded:0

USB MSCD SCSI Class Driver Counters:
```

```
          Good Status Failures :3                Command Fail    :0
          Good Status Timed out:0                Device not Found:0
          Device Never Opened  :0                Drive Init Fail :0
          Illegal App Handle   :0                Bad API Command :0
          Invalid Unit Number  :0                Invalid Argument:0
          Application Overflow :0                Device in use   :0
          Control Pipe Stall   :0                Malloc Error    :0
          Device Stalled       :0                Bad Command Code:0
          Device Detached      :0                Unknown Error   :0
          Invalid Logic Unit Num:0

USB Aladdin Token Driver Counters:
          Token Inserted       :1                Token Removed   :0
          Send Insert Msg Fail :0                Response Txns   :434
          Dev Entry Add Fail   :0                Request Txns    :434
          Dev Entry Remove Fail:0                Request Txn Fail:0
          Response Txn Fail    :0                Command Txn Fail:0
          Txn Invalid Dev Handle:0

USB Flash File System Counters:
          Flash Disconnected   :0                Flash Connected :1
          Flash Device Fail    :0                Flash Ok        :1
          Flash startstop Fail :0                Flash FS Fail   :0

USB Secure Token File System Counters:
          Token Inserted       :1                Token Detached  :0
          Token FS success     :1                Token FS Fail   :0
          Token Max Inserted   :0                Create Talker Failures:0
          Token Event          :0                Destroy Talker Failures:0
          Watched Boolean Create Failures:0
```

The dir Command

Use the **dir** command with the **filesystem** keyword option **usbtoken**[**0**-**9**]**:** to display all files, directories, and their permission strings on the USB token.

SUMMARY STEPS

1. **dir** [**filesystem**:]

DETAILED STEPS

Step 1 The following sample output displays directory information for the USB token:

```
Router# dir usbtoken1:

Directory of usbtoken1:/

    2  d---          64  Dec 22 2032 05:23:40 +00:00  1000
    5  d---        4096  Dec 22 2032 05:23:40 +00:00  1001
    8  d---           0  Dec 22 2032 05:23:40 +00:00  1002
   10  d---         512  Dec 22 2032 05:23:42 +00:00  1003
   12  d---           0  Dec 22 2032 05:23:42 +00:00  5000
   13  d---           0  Dec 22 2032 05:23:42 +00:00  6000
   14  d---           0  Dec 22 2032 05:23:42 +00:00  7000
   15  ----         940  Jun 27 1992 12:50:42 +00:00  mystartup-config
   16  ----        1423  Jun 27 1992 12:51:14 +00:00  myrunning-config

32768 bytes total (858 bytes free)
```

The following sample output displays directory information for all devices the router is aware of:

```
Router# dir all-filesystems

Directory of archive:/

No files in directory

No space information available
Directory of system:/

    2  drwx           0                    <no date>  its
  115  dr-x           0                    <no date>  lib
  144  dr-x           0                    <no date>  memory
    1  -rw-        1906                    <no date>  running-config
  114  dr-x           0                    <no date>  vfiles

No space information available
Directory of flash:/

    1  -rw-    30125020  Dec 22 2032 03:06:04 +00:00  c3825-entservicesk9-mz.123-14.T

129880064 bytes total (99753984 bytes free)
Directory of nvram:/

  476  -rw-        1947                    <no date>  startup-config
  477  ----          46                    <no date>  private-config
  478  -rw-        1947                    <no date>  underlying-config
    1  -rw-           0                    <no date>  ifIndex-table
    2  ----           4                    <no date>  rf_cold_starts
    3  ----          14                    <no date>  persistent-data

491512 bytes total (486395 bytes free)
Directory of usbflash0:/

    1  -rw-    30125020  Dec 22 2032 05:31:32 +00:00  c3825-entservicesk9-mz.123-14.T

63158272 bytes total (33033216 bytes free)
Directory of usbtoken1:/

    2  d---          64  Dec 22 2032 05:23:40 +00:00  1000
    5  d---        4096  Dec 22 2032 05:23:40 +00:00  1001
    8  d---           0  Dec 22 2032 05:23:40 +00:00  1002
```

```
  10  d---         512  Dec 22 2032 05:23:42 +00:00  1003
  12  d---           0  Dec 22 2032 05:23:42 +00:00  5000
  13  d---           0  Dec 22 2032 05:23:42 +00:00  6000
  14  d---           0  Dec 22 2032 05:23:42 +00:00  7000
  15  ----         940  Jun 27 1992 12:50:42 +00:00  mystartup-config
  16  ----        1423  Jun 27 1992 12:51:14 +00:00  myrunning-config

32768 bytes total (858 bytes free)
```

Configuration Examples for PKI Storage

This section contains the following configuration examples:

Storing Certificates to a Specific Local Storage Location: Example

The following configuration example shows how to store certificates to the certs subdirectory. The certs subdirectory does not exist and is automatically created.

```
Router# dir nvram:

 114  -rw-        4687                    <no date>  startup-config
 115  ----        5545                    <no date>  private-config
 116  -rw-        4687                    <no date>  underlying-config
   1  ----          34                    <no date>  persistent-data
   3  -rw-         707                    <no date>  ioscaroot#7401CA.cer
   9  -rw-         863                    <no date>  msca-root#826E.cer
  10  -rw-         759                    <no date>  msca-root#1BA8CA.cer
  11  -rw-         863                    <no date>  msca-root#75B8.cer
  24  -rw-        1149                    <no date>  storagename#6500CA.cer
  26  -rw-         863                    <no date>  msca-root#83EE.cer

129016 bytes total (92108 bytes free)

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

Router(config)# crypto pki certificate storage disk0:/certs
Requested directory does not exist -- created
Certificates will be stored in disk0:/certs/

Router(config)# end
Router# write
*May 27 02:09:00:%SYS-5-CONFIG_I:Configured from console by consolemem
Building configuration...
[OK]

Router# directory disk0:/certs

Directory of disk0:/certs/

  14  -rw-         707  May 27 2005 02:09:02 +00:00  ioscaroot#7401CA.cer
  15  -rw-         863  May 27 2005 02:09:02 +00:00  msca-root#826E.cer
  16  -rw-         759  May 27 2005 02:09:02 +00:00  msca-root#1BA8CA.cer
  17  -rw-         863  May 27 2005 02:09:02 +00:00  msca-root#75B8.cer
  18  -rw-        1149  May 27 2005 02:09:02 +00:00  storagename#6500CA.cer
```

```
  19  -rw-         863  May 27 2005 02:09:02 +00:00  msca-root#83EE.cer

47894528 bytes total (20934656 bytes free)


! The certificate files are now on disk0/certs:
```

Logging Into a USB Token and Saving RSA Keys to the USB Token: Example

The following configuration example shows to how log into the USB token, generate RSA keys, and store the RSA keys onto the USB token:

```
! Configure the router to automatically log into the eToken
configure terminal
 crypto pki token default user-pin 0 1234567890
! Generate RSA keys and enroll certificates with the CA.
crypto pki trustpoint IOSCA
 enrollment url http://10.23.2.2
 exit
crypto ca authenticate IOSCA
Certificate has the following attributes:
       Fingerprint MD5:23272BD4 37E3D9A4 236F7E1A F534444E
      Fingerprint SHA1:D1B4D9F8 D603249A 793B3CAF 8342E1FE 3934EB7A

% Do you accept this certificate? [yes/no]:yes
Trustpoint CA certificate accepted.
crypto pki enroll
crypto pki enroll IOSCA
%
% Start certificate enrollment ..
% Create a challenge password. You will need to verbally provide this
   password to the CA Administrator in order to revoke your certificate.
   For security reasons your password will not be saved in the configuration.
   Please make a note of it.

Password:
Re-enter password:

% The subject name in the certificate will include:c2851-27.cisco.com
% Include the router serial number in the subject name? [yes/no]:no
% Include an IP address in the subject name? [no]:no
Request certificate from CA? [yes/no]:yes
% Certificate request sent to Certificate Authority
% The 'show crypto ca certificate IOSCA verbose' command will show the fingerprint.


*Jan 13 06:47:19.413:CRYPTO_PKI: Certificate Request Fingerprint MD5:E6DDAB1B
 0E30EFE6 54529D8A DA787DBA
*Jan 13 06:47:19.413:CRYPTO_PKI: Certificate Request Fingerprint SHA1:3B0F33B
7 57C02A10 3935042B C4B6CD3D 61039251
*Jan 13 06:47:21.021:%PKI-6-CERTRET:Certificate received from Certificate Authority
! Issue the write memory command, which will automatically save the RSA keys to the eToken
! instead of private NVRAM.
Router# write memory
Building configuration...
[OK]

*Jan 13 06:47:29.481:%CRYPTO-6-TOKENSTOREKEY:Key c2851-27.cisco.com stored on
Cryptographic Token eToken Successfully
```

The following sample output from the **show crypto key mypubkey rsa** command displays stored credentials after they are successfully loaded from the USB token. Credentials that are stored on the USB token are in the protected area. When storing the credentials on the USB token, the files are stored in a directory called /keystore. However, the key files are hidden from the command-line interface (CLI).

```
Router# show crypto key mypubkey rsa

% Key pair was generated at:06:37:26 UTC Jan 13 2005
Key name:c2851-27.cisco.com
 Usage:General Purpose Key
 Key is not exportable.
 Key Data:
  305C300D 06092A86 4886F70D 01010105 00034B00 30480241 00E3C644 43AA7DDD
  732E0F4E 3CA0CDAB 387ABF05 EB8F22F2 2431F1AE 5D51FEE3 FCDEA934 7FBD3603
  7C977854 B8E999BF 7FC93021 7F46ABF8 A4BA2ED6 172D3D09 B5020301 0001
% Key pair was generated at:06:37:27 UTC Jan 13 2005
Key name:c2851-27.cisco.com.server
 Usage:Encryption Key
 Key is not exportable.
 Key Data:
  307C300D 06092A86 4886F70D 01010105 00036B00 30680261 00DD96AE 4BF912EB
  2C261922 4784EF98 2E70E837 774B3778 7F7AEB2D 87F5669B BF5DDFBC F0D521A5
  56AB8FDC 9911968E DE347FB0 A514A856 B30EAFF4 D1F453E1 003CFE65 0CCC6DC7
  21FBE3AC 2F8DEA16 126754BC 1433DEF9 53266D33 E7338C95 BB020301 0001
```

Additional References

Related Documents

Related Topic	Document Title
Connecting the USB modules to the router	*Cisco Access Router USB Flash Module and USB eToken Hardware Installation Guide*
eToken and USB flash data sheet	*USB eToken and USB Flash Features Support*
RSA keys	*Deploying RSA Keys Within a PKI*
File management (loading, copying, and rebooting files)	*Cisco IOS Configuration Fundamentals Configuration Guide* on Cisco.com
USB Token RSA Operations: Certificate server configuration	"Configuring and Managing a Cisco IOS Certificate Server for PKI Deployment" chapter in the *Cisco IOS Security Configuration Guide: Secure Connectivity* See the "Generating a Certificate Server RSA Key Pair" section, the "Configuring a Certificate Server Trustpoint" section, and related examples.

Related Topic	Document Title
USB Token RSA Operations: Using USB tokens for RSA operations upon initial autoenrollment	"Configuring Certificate Enrollment for a PKI" chapter in the *Cisco IOS Security Configuration Guide: Secure Connectivity.* See the "Configuring Certificate Enrollment or Autoenrollment" section.
SDP setup, configuration and use with USB tokens	"Setting Up Secure Device Provisioning (SDP) for Enrollment in a PKI" chapter in the *Cisco IOS Security Configuration Guide: Secure Connectivity.* See the feature information section for the feature names on using SDP and USB tokens to deploy PKI credentials.

Technical Assistance

Description	Link
The Cisco Support and Documentation website provides online resources to download documentation, software, and tools. Use these resources to install and configure the software and to troubleshoot and resolve technical issues with Cisco products and technologies. Access to most tools on the Cisco Support and Documentation website requires a Cisco.com user ID and password.	http://www.cisco.com/cisco/web/support/index.html

Feature Information for Storing PKI Credentials

Table 2 lists the release history for this feature.

Use Cisco Feature Navigator to find information about platform support and software image support. Cisco Feature Navigator enables you to determine which 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 2 lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Table 2 Feature Information for Storing PKI Credentials

Feature Name	Releases	Feature Information
USB Token and Secure Device Provisioning (SDP) Integration	12.4(15)T	This feature provides the ability to provision remote devices with USB tokens using SDP. The following sections in this document provide information about this feature: • Benefits of USB Tokens • Setting Administrative Functions on the USB Token The following commands were introduced by this feature: **binary file**, **crypto key move rsa**, **template file**. **Note** This document introduces the benefits of using USB tokens and SDP for a deployment solution. For other documentation on this topic, see the "Related Documents" section.
Cisco IOS USB Token PKI Enhancements — Phase 2	12.4(11)T	This feature enhances USB token functionality by using the USB token as a cryptographic device. USB tokens may be used for RSA operations such as key generation, signing, and authentication. The following sections in this document provide information about this feature: • Benefits of USB Tokens • Logging Into and Setting Up the USB Token • Setting Administrative Functions on the USB Token **Note** This document introduces the benefits of using USB tokens and the keys on the token for RSA operations. For other documentation on this topic, see the "Related Documents" section.

Table 2 Feature Information for Storing PKI Credentials (continued)

Feature Name	Releases	Feature Information
USB Storage PKI Enhancements	12.4(4)T 12.4(11)T	This feature enhances the USB token PIN security for automatic login and increases the flexibility of USB token configuration and the RSA key storage. Cisco IOS Release 12.4(11)T introduced support for USB Storage on NPE-G2. The following sections provide information about this feature: • Configuring the USB Token • Setting Administrative Functions on the USB Token The following commands were introduced or modified by this feature: **crypto key storage**, **crypto pki generate rsa, crypto pki token encrypted-user-pin**, **crypto pki token label**, **crypto pki token lock**, **crypto pki token secondary unconfig**, **crypto pki token unlock**
Certificate — Storage Location Specification	12.2(33)SXH 12.2(33)SRA 12.4(2)T	This feature allows you to specify the storage location of local certificates for platforms that support storing certificates as separate files. All Cisco platforms support NVRAM, which is the default location, and flash local storage. Depending on your platform, you may have other supported local storage options including bootflash, slot, disk, USB flash, or USB token. The following sections provide information about this feature: • Storing Certificates to a Local Storage Location • Specifying a Local Storage Location for Certificates • Storing Certificates to a Specific Local Storage Location: Example The following commands were introduced by this feature: **crypto pki certificate storage**, **show crypto pki certificates storage**

Table 2 Feature Information for Storing PKI Credentials (continued)

Feature Name	Releases	Feature Information
USB Storage	12.3(14)T 12.4(11)T	This feature enables certain models of Cisco routers to support USB tokens. USB tokens provide secure configuration distribution and allow users to VPN credentials for deployment. Cisco IOS Release 12.4(11)T introduced support for USB Storage on NPE-G2. The following sections provide information about this feature: • PKI Credentials and USB Tokens • Setting Up and Using USB Tokens on Cisco Routers • Troubleshooting USB Tokens • Logging Into a USB Token and Saving RSA Keys to the USB Token: Example The following commands were introduced or modified by this feature: **copy**, **crypto pki token change-pin**, **crypto pki token login**, **crypto pki token logout**, **crypto pki token max-retries**, **crypto pki token removal timeout**, **crypto pki token secondary config**, **crypto pki token user-pin**, **debug usb driver**, **dir**, **show usb controllers**, **show usb device**, **show usb driver**, **show usbtoken**
RSA 4096-bit Key Generation in Software Crypto Engine Support	15.1(1)T	The range value for the **modulus** keyword value for the **crypto key generate rsa** command is extended from 360 to 2048 bits to 360 to 4096 bits.