HTTP Services Configuration Guide, Cisco IOS XE Release 3SE (Catalyst 3650 Switches)

First Published: October 10, 2012

Americas Headquarters
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706
USA
http://www.cisco.com
Tel: 408 526-4000
800 553-NETS (6387)
Fax: 408 527-0883

CONTENTS

CHAPTER 1 **HTTP 1.1 Web Server and Client** 1

Finding Feature Information 1

Information About the HTTP 1.1 Web Server and Client 1

About HTTP Server General Access Policies 2

How to Configure HTTP 1.1 Web Server and Client 3

Configuring the HTTP 1.1 Web Server 3

Configuring the HTTP Client 5

Verifying HTTP Connectivity 7

Configuration Examples for HTTP 1.1 Web Server 8

Configuring the HTTP 1.1 Web Server Example 8

Where to Go Next 8

Additional References 8

Feature Information for the HTTP 1.1 Web Server and Client 10

CHAPTER 2 **HTTPS--HTTP Server and Client with SSL 3.0** 13

Finding Feature Information 13

Information About HTTPS--HTTP Server and Client with SSL 3.0 13

Secure HTTP Server and Secure HTTP Client 14

Certificate Authority Trustpoints 14

CipherSuites 14

How to Configure the HTTPS--HTTP Server and Client with SSL 3.0 15

Declaring a Certificate Authority Trustpoint 15

Configuring the HTTPS Server with SSL 3.0 18

Verifying the Configuration of the HTTPS Server 21

Providing Additional Security and Efficiency 21

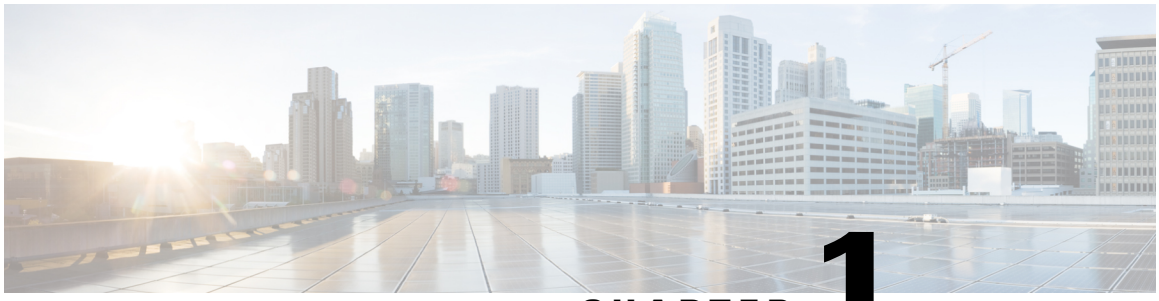
Configuring the HTTPS Client with SSL 3.0 23

Configuration Examples for the HTTPS--HTTP Server and Client with SSL 3.0 feature 25

Additional References 26

Feature Information for HTTPS--HTTP Server and Client with SSL 3.0 **27**

Glossary **29**



CHAPTER 1

HTTP 1.1 Web Server and Client

The HTTP 1.1 Web Server and Client feature provides a consistent interface for users and applications by implementing support for HTTP 1.1 in Cisco IOS XE software-based devices. When combined with the HTTPS feature, the HTTP 1.1 Web Server and Client feature provides a complete, secure solution for HTTP services between Cisco devices.

This module describes the concepts and the tasks related to configuring the HTTP 1.1 Web Server and Client feature.

- Finding Feature Information, page 1
- Information About the HTTP 1.1 Web Server and Client, page 1
- How to Configure HTTP 1.1 Web Server and Client, page 3
- Configuration Examples for HTTP 1.1 Web Server, page 8
- Where to Go Next, page 8
- Additional References, page 8
- Feature Information for the HTTP 1.1 Web Server and Client, page 10

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest caveats and feature information, see Bug Search Tool and 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 table at the end of this module.

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

Information About the HTTP 1.1 Web Server and Client

This feature updates the Cisco implementation of the Hypertext Transfer Protocol (HTTP) from 1.0 to 1.1. The HTTP server allows features and applications, such as the Cisco web browser user interface, to be run on your routing device.

The Cisco implementation of HTTP 1.1 is backward-compatible with previous Cisco IOS XE releases. If you are currently using configurations that enable the HTTP server, no configuration changes are needed, as all defaults remain the same.

The process of enabling and configuring the HTTP server also remains the same as in previous releases. Support for Server Side Includes (SSIs) and HTML forms has not changed. Additional configuration options, in the form of the **ip http timeout-policy** command and the **ip http max-connections**command, have been added. These options allow configurable resource limits for the HTTP server. If you do not use these optional commands, the default policies are used.

Remote applications may require that you enable the HTTP server before using them. Applications that use the HTTP server include:

- Cisco web browser user interface, which uses the Cisco IOS XE Homepage Server, HTTP-based EXEC Server, and HTTP IOS File System (IFS) Server
- VPN Device Manager (VDM) application, which uses the VDM Server and the XML Session Manager (XSM)
- QoS Device Manager (QDM) application, which uses the QDM Server
- IP Phone and Cisco IOS XE Telephony Service applications, which use the ITS Local Directory Search and IOS Telephony Server (ITS)

About HTTP Server General Access Policies

The **ip http timeout-policy** command allows you to specify general access characteristics for the server by configuring a value for idle time, connection life, and request maximum. By adjusting these values you can configure a general policy; for example, if you want to maximize throughput for HTTP connections, you should configure a policy that minimizes connection overhead. You can configure this type of policy by specifying large values for the **life** and **request** options so that each connection stays open longer and more requests are processed for each connection.

Another example would be to configure a policy that minimizes the response time for new connections. You can configure this type of policy by specifying small values for the **life** and **request** options so that the connections are quickly released to serve new clients.

A throughput policy would be better for HTTP sessions with dedicated management applications, as it would allow the application to send more requests before the connection is closed, while a response time policy would be better for interactive HTTP sessions, as it would allow more people to connect to the server at the same time without having to wait for connections to become available.

In general, you should configure these options as appropriate for your environment. The value for the **idle** option should be balanced so that it is large enough not to cause an unwanted request or response timeout on the connection, but small enough that it does not hold a connection open longer than necessary.

Access security policies for the HTTP server are configured using the **ip http authentication** command, which allows only selective users to access the server, the **ip http access-class** command, which allows only selective IP hosts to access the server, and the **ip http accounting commands** command, which specifies a particular command accounting method for HTTP server users.

How to Configure HTTP 1.1 Web Server and Client

Configuring the HTTP 1.1 Web Server

Perform this task to enable the HTTP server and configure optional server characteristics. The HTTP server is disabled by default.

Note If you want to configure authentication (step 4), you must configure the authentication type before you begin configuring the HTTP 1.1 web server.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **ip http server**
4. **ip http authentication** {**aaa** | **enable** | **local** | **tacacs**}
5. **ip http accounting commands** *level* {**default** | *named-accounting-method-list*}
6. **ip http port** *port-number*
7. **ip http path** *url*
8. **ip http access-class** *access-list-number*
9. **ip http max-connections** *value*
10. **ip http timeout-policy** **idle** *seconds* **life** *seconds* **requests** *value*

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	**ip http server** **Example:** `Router(config)# ip http server`	Enables the HTTP 1.1 server, including the Cisco web browser user interface. **Note** If you are enabling the HTTP over Secure Socket Layer (HTTPS) server using the **ip http secure-server** command, you should disable the standard HTTP server using the **no ip http server** command. This command is required to ensure only secure connections to the server.

	Command or Action	Purpose
Step 4	**ip http authentication** {**aaa** | **enable** | **local** | **tacacs**} **Example:** `Router(config)# ip http authentication local`	(Optional) Specifies the authentication method to be used for login when a client connects to the HTTP server. The methods for authentication are: • **aaa** --Indicates that the authentication method used for the AAA login service (specified by the **aaa authentication login default** command) should be used for authentication. • **enable** --Indicates that the "enable" password should be used for authentication. (This is the default method.) • **local** --Indicates that the login user name, password and privilege level access combination specified in the local system configuration (by the **username** global configuration command) should be used for authentication and authorization. • **tacacs** --Indicates that the TACACS (or XTACACS) server should be used for authentication.
Step 5	**ip http accounting commands** *level* {**default** | *named-accounting-method-list*} **Example:** `Router(config)# ip http accounting commands 15 default`	(Optional) Specifies a particular command accounting method for HTTP server users. Command accounting for HTTP and HTTPS is automatically enabled when authentication, authorization, and accounting (AAA) is configured on the device. It is not possible to disable accounting for HTTP and HTTPS. HTTP and HTTPS will default to using the global AAA default method list for accounting. The CLI can be used to configure HTTP and HTTPS to use any predefined AAA method list. • *level* --Valid privilege level entries are integers from 0 to 15. • **default** --Indicates the **default** accounting method list configured by the aaa accounting commands CLI. • *named-accounting-method-list* --Indicates the name of the predefined command accounting method list.
Step 6	**ip http port** *port-number* **Example:** `Router(config)# ip http port 8080`	(Optional) Specifies the server port that should be used for HTTP communication (for example, for the Cisco web browser user interface).
Step 7	**ip http path** *url* **Example:** `Router(config)# ip http path slot1:`	(Optional) Sets the base HTTP path for HTML files. The base path is used to specify the location of the HTTP server files (HTML files) on the local system. Generally, the HTML files are located in system flash memory.

	Command or Action	Purpose
Step 8	**ip http access-class** *access-list-number* **Example:** `Router(config)# ip http access-class 20`	(Optional) Specifies the access list that should be used to allow access to the HTTP server.
Step 9	**ip http max-connections** *value* **Example:** `Router(config)# ip http max-connections 10`	(Optional) Sets the maximum number of concurrent connections to the HTTP sever that will be allowed. The default value is 5.
Step 10	**ip http timeout-policy idle** *seconds* **life** *seconds* **requests** *value* **Example:** `Router(config)# ip http timeout-policy idle 30 life 120 requests 100`	(Optional) Sets the characteristics that determine how long a connection to the HTTP server should remain open. The characteristics are: • **idle** --The maximum number of seconds the connection will be kept open if no data is received or response data cannot be sent out on the connection. Note that a new value may not take effect on any already existing connections. If the server is too busy or the limit on the **life** time or the number of **requests** is reached, the connection may be closed sooner. The default value is 180 seconds (3 minutes). • **life** --The maximum number of seconds the connection will be kept open, from the time the connection is established. Note that the new value may not take effect on any already existing connections. If the server is too busy or the limit on the idle time or the number of requests is reached, it may close the connection sooner. Also, since the server will not close the connection while actively processing a request, the connection may remain open longer than the specified **life** time if processing is occurring when the life maximum is reached. In this case, the connection will be closed when processing finishes. The default value is 180 seconds (3 minutes). The maximum value is 86400 seconds (24 hours). • **requests** --The maximum limit on the number of requests processed on a persistent connection before it is closed. Note that the new value may not take effect on already existing connections. If the server is too busy or the limit on the idle time or the life time is reached, the connection may be closed before the maximum number of requests are processed. The default value is 1. The maximum value is 86400.

Configuring the HTTP Client

Perform this task to enable the HTTP client and configure optional client characteristics.

The standard HTTP 1.1 client and the secure HTTP client are always enabled. No commands exist to disable the HTTP client. For information about configuring optional characteristics for the HTTPS client, see the HTTPS--HTTP Server and Client with SSL 3.0 feature module.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **ip http client cache** {**ager interval** *minutes* | **memory** {**file** *file-size-limit* | **pool** *pool-size-limit*}
4. **ip http client connection** {**forceclose** | **idle timeout** *seconds* | **retry** *count* | **timeout** *seconds*}
5. **ip http client password** *password*
6. **ip http client proxy-server** *proxy-name* **proxy-port** *port-number*
7. **ip http client response timeout** *seconds*
8. **ip http client source-interface** *type number*
9. **ip http client username** *username*

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	**ip http client cache** {**ager interval** *minutes* | **memory** {**file** *file-size-limit* | **pool** *pool-size-limit*} **Example:** `Router(config)# ip http client cache memory file 5`	Configures HTTP client cache.
Step 4	**ip http client connection** {**forceclose** | **idle timeout** *seconds* | **retry** *count* | **timeout** *seconds*} **Example:** `Router(config)# ip http client connection timeout 10`	Configures an HTTP client connection.
Step 5	**ip http client password** *password* **Example:** `Router(config)# ip http client password pswd1`	Configures the default password used for connections to remote HTTP servers.

	Command or Action	Purpose
Step 6	**ip http client proxy-server** *proxy-name* **proxy-port** *port-number* **Example:** `Router(config)# ip http client proxy-server server1 proxy-port 52`	Configures an HTTP proxy server.
Step 7	**ip http client response timeout** *seconds* **Example:** `Router(config)# ip http client response timeout 60`	Specifies the timeout value, in seconds, that the HTTP client waits for a response from the server.
Step 8	**ip http client source-interface** *type number* **Example:** `Router(config)# ip http client source-interface ethernet1/0`	Configures a source interface for the HTTP client.
Step 9	**ip http client username** *username* **Example:** `Router(config)# ip http client user1`	Configures the default username used for connections to remote HTTP servers.

Verifying HTTP Connectivity

To verify remote connectivity to the HTTP server, enter the system IP address in a web browser, followed by a colon and the appropriate port number (80 is the default port number).

For example, if the system IP address is 209.165.202.129 and the port number is 8080, enter http://209.165.202.129:8080 as the URL in a web browser.

If HTTP authentication is configured, a login dialog box will appear. Enter the appropriate username and password. If the default login authentication method of “enable” is configured, you may leave the username field blank, and use the “enable” password to log in.

The system home page should appear in your browser.

Configuration Examples for HTTP 1.1 Web Server

Configuring the HTTP 1.1 Web Server Example

The following example shows a typical configuration that enables the server and sets some of the characteristics:

```
ip http server
ip http authentication aaa
ip http accounting commands 15 default
ip http path flash:
ip access-list standard 20
 permit 209.165.202.130 0.0.0.255
 permit 209.165.201.1 0.0.255.255
 permit 209.165.200.225 0.255.255.255
! (Note: all other access implicitly denied)
 end
ip http access-class 10
ip http max-connections 10
ip http accounting commands 1 oneacct
```

In the following example, a Throughput timeout policy is applied. This configuration would allow each connection to be idle a maximum of 30 seconds (approximately). Each connection will remain open (be “alive”) until either the HTTP server has been busy processing requests for approximately 2 minutes (120 seconds) or until approximately100 requests have been processed.

```
ip http timeout-policy idle 30 life 120 requests 100
```

In the following example, a Response Time timeout policy is applied. This configuration would allow each connection to be idle a maximum of 30 seconds (approximately). Each connection will be closed as soon as the first request has been processed.

```
ip http timeout-policy idle 30 life 30 requests 1
```

Where to Go Next

For information about secure HTTP connections using Secure Sockets Layer (SSL) 3.0, refer to the HTTPS--HTTP with SSL 3.0 feature module at:
http://www.cisco.com/en/US/docs/ios/12_2t/12_2t15/feature/guide/ftsslsht.html

Additional References

Related Documents

Related Topic	Document Title
Cisco IOS commands	Cisco IOS Master Commands List, All Releases
HTTP commands: complete command syntax, command mode, command history, defaults, usage guidelines, and examples	Cisco IOS HTTP Services Command Reference

Related Topic	Document Title
HTTPS	• HTTPS--HTTP with SSL 3.0 feature module • Firewall Support of HTTPS Authentication Proxy feature module

Standards and RFCs

Standard/RFC	Title
No specific standards are supported by this feature. Note that HTTP 1.1, as defined in RFC 2616, is currently classified as a “Standards Track” document by the IETF.	—
RFC 2616	*Hypertext Transfer Protocol -- HTTP/1.1*

The Cisco implementation of the HTTP Version 1.1 supports a subset of elements defined in RFC 2616. Following is a list of supported RFC 2616 headers:

- Allow (Only GET, HEAD, and POST methods are supported)
- Authorization, WWW-Authenticate - Basic authentication only
- Cache-control
- Chunked Transfer Encoding
- Connection close
- Content-Encoding
- Content-Language
- Content-Length
- Content-Type
- Date, Expires
- Location

MIBs

MIB	MIBs Link
• No specific MIBs are supported for this feature.	To locate and download MIBs for selected platforms, Cisco software releases, and feature sets, use Cisco MIB Locator found at the following URL: http://www.cisco.com/go/mibs

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 the HTTP 1.1 Web Server and Client

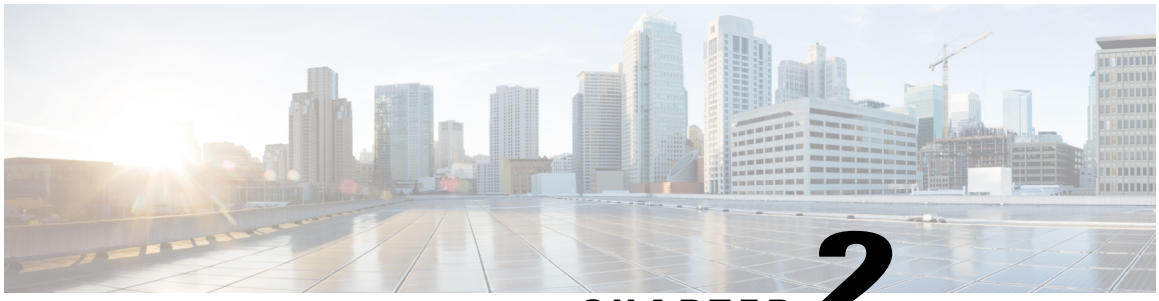
The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

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

Table 1: Feature Information for HTTP 1.1 Web Server and Client

Feature Name	Releases	Feature Information
HTTP 1.1 Web Server and Client	Cisco IOS XE Release 2.1	The HTTP 1.1 Web Server and Client feature provides a consistent interface for users and applications by implementing support for HTTP 1.1 in Cisco IOS XE software-based devices. When combined with the HTTPS feature, the HTTP 1.1 Web Server and Client feature provides a complete, secure solution for HTTP services between Cisco devices. The following commands were introduced or modified by this feature: **debug ip http all**, **debug ip http client**, **ip http access-class**, **ip http authentication**, **ip http client cache**, **ip http client connection**, **ip http client password**, **ip http client proxy-server**, **ip http client response timeout**, **ip http client source-interface**, **ip http client username**, **ip http max-connections**, **ip http path**, **ip http port**, **ip http server**, **ip http timeout-policy**, **show ip http client**, **show ip http client connection**, **show ip http client history**, **show ip http client session-module**, **show ip http server**, s**how ip http server secure status**.

Feature Name	Releases	Feature Information
HTTP TACAC+ Accounting Support	Cisco IOS XE Release 2.1	The HTTP TACAC+ Accounting Support feature introduces the **ip http accounting commands** command. This command is used to specify a particular command accounting method for HTTP server users. Command accounting provides information about the commands for a specified privilege level that are being executed on a device. Each command accounting record corresponds to one IOS XE command executed at its respective privilege level, as well as the date and time the command was executed, and the user who executed it. The following sections provide information about this feature: The following commands were introduced or modified by this feature: **ip http accounting commands**.
HTTP Security	Cisco IOS XE Release 2.1	This feature was introduced on Cisco ASR 1000 Series Routers.



CHAPTER 2

HTTPS--HTTP Server and Client with SSL 3.0

The HTTPS--HTTP Server and Client with SSL 3.0 feature provides Secure Socket Layer (SSL) version 3.0 support for the HTTP 1.1 server and HTTP 1.1 client within Cisco IOS XE software. SSL provides server authentication, encryption, and message integrity to allow secure HTTP communications. SSL also provides HTTP client authentication. HTTP over SSL is abbreviated as HTTPS.

- Finding Feature Information, page 13
- Information About HTTPS--HTTP Server and Client with SSL 3.0, page 13
- How to Configure the HTTPS--HTTP Server and Client with SSL 3.0, page 15
- Configuration Examples for the HTTPS--HTTP Server and Client with SSL 3.0 feature, page 25
- Additional References, page 26
- Feature Information for HTTPS--HTTP Server and Client with SSL 3.0, page 27
- Glossary, page 29

Finding Feature Information

Your software release may not support all the features documented in this module. For the latest caveats and feature information, see Bug Search Tool and 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 table at the end of this module.

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

Information About HTTPS--HTTP Server and Client with SSL 3.0

To configure the HTTP with SSL 3.0 (HTTPS) feature, you should understand the following concepts:

Secure HTTP Server and Secure HTTP Client

A secure HTTP connection means that data sent to and received from an HTTP server are encrypted before being sent out over the Internet. HTTP with SSL encryption provides a secure connection to allow such functions as configuring a router from a web browser. Cisco's implementation of the secure HTTP server and secure HTTP client uses an implementation of the SSL version 3.0. Application layer encryption provides an alternative to older methods such as having to set up a tunnel to the HTTP server for remote management. HTTP over SSL is abbreviated as HTTPS; the URL of a secure connection will begin with https:// instead of http://.

The Cisco IOS XE HTTP secure server's primary role is to listen for HTTPS requests on a designated port (the default HTTPS port is 443) and to pass the request to the HTTP 1.1 web server. The HTTP 1.1 server processes requests and passes responses (served pages) back to the HTTP secure server, which, in turn, responds to the original request.

The Cisco IOS XE HTTP secure client's primary role is to respond to Cisco IOS XE application requests for HTTPS User Agent services, perform HTTPS User Agent services on the application's behalf, and pass the response back to the application.

Certificate Authority Trustpoints

Certificate authorities (CAs) are responsible for managing certificate requests and issuing certificates to participating IPSec network devices. These services provide centralized security key and certificate management for the participating devices. Specific CA servers are referred to as "trustpoints."

The HTTPS server provides a secure connection by providing a certified X.509v3 certificate to the client when a connection attempt is made. The certified X.509v3 certificate is obtained from a specified CA trustpoint. The client (usually a web browser), in turn, has a public key that allows it to authenticate the certificate.

Configuring a CA trustpoint is highly recommended for secure HTTP connections. However, if a CA trustpoint is not configured for the routing device running the HTTPS server, the server will certify itself and generate the needed RSA key pair. Because a self-certified (self-signed) certificate does not provide adequate security, the connecting client will generate a notification that the certificate is self-certified, and the user will have the opportunity to accept or reject the connection. This option is available for internal network topologies (such as testing).

The HTTPS--HTTP Server and Client with SSL 3.0 feature also provides an optional command (**ip http secure-client-auth**) that, when enabled, has the HTTPS server request an X.509v3 certificate from the client. Authenticating the client provides more security than server authentication by itself.

For additional information on certificate authorities, see the "Configuring Certification Authority Interoperability" chapter in the Cisco IOS XE Security Configuration Guide .

CipherSuites

A CipherSuite specifies the encryption algorithm and digest algorithm to use on an SSL connection. Web browsers offer a list of supported CipherSuites when connecting to the HTTPS server, and the client and server will negotiate the best encryption algorithm to use from those that are supported by both. For example, Netscape Communicator 4.76 supports U.S. security with RSA Public Key Cryptography, MD2, MD5, RC2-CBC, RC4, DES-CBC, and DES-EDE3-CBC.

For the best possible encryption, you should use a browser that supports 128-bit encryption, such as Microsoft Internet Explorer version 5.5 (or later), or Netscape Communicator version 4.76 (or later). The SSL_RSA_WITH_DES_CBC_SHA CipherSuite provides less security than the other CipherSuites, because it does not offer 128-bit encryption.

In terms of router processing load (speed), the following list ranks the CipherSuites from fastest to slowest (slightly more processing time is required for the more secure and more complex CipherSuites):

1. SSL_RSA_WITH_DES_CBC_SHA
2. SSL_RSA_WITH_RC4_128_MD5
3. SSL_RSA_WITH_RC4_128_SHA
4. SSL_RSA_WITH_3DES_EDE_CBC_SHA

How to Configure the HTTPS--HTTP Server and Client with SSL 3.0

Declaring a Certificate Authority Trustpoint

Configuring a CA trustpoint is highly recommended for secure HTTP connections. The certified X.509v3 certificate for the secure HTTP server (or client) is obtained from the specified CA trustpoint. If you do not declare a CA trustpoint, then a self-signed certificate will be used for secure HTTP connections. The self-signed certificate is generated automatically.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **hostname** *name*
4. **ip domain-name** *name*
5. **crypto key generate rsa usage-keys**
6. **crypto ca trustpoint** *name*
7. **enrollment url** *url*
8. **enrollment http-proxy** *host-name port-number*
9. **crl** {**query** *url* | **optional** | **best-effort**}
10. **primary**
11. **exit**
12. **crypto ca authenticate** *name*
13. **crypto ca enrollment** *name*
14. Do one of the following:
    - **copy running-config startup-config**
    -
    - **copy system:running-config nvram:startup-config**

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	**hostname** *name* **Example:** `Router(config)# hostname Router`	Specifies the hostname of the router. • This step is needed only if you have not previously configured a hostname for your router. The hostname is required because a fully qualified domain name is needed for security keys and certificates.
Step 4	**ip domain-name** *name*	Specifies the IP domain name of the router.

	Command or Action	Purpose
	Example: `Router(config)# ip domain-name example.com`	• This step is needed only if you have not previously configured an IP domain name for your router. The domain name is required because a fully qualified domain name is needed for security keys and certificates.
Step 5	**crypto key generate rsa usage-keys** **Example:** `Router(config)# crypto key generate rsa usage-keys`	(Optional) Generates an RSA key pair. • The **usage-keys** keyword specifies that two RSA special-usage key pairs should be generated (that is, one encryption pair and one signature pair) instead of one general-purpose key pair. • RSA key pairs are used to sign and encrypt Internet key exchange (IKE) key management messages and are required before you can obtain a certificate for your router. • RSA key pairs are generated automatically. This command can be used to regenerate the keys, if needed. **Note** There are other keywords and arguments for this command, but they do not pertain to this feature.
Step 6	**crypto ca trustpoint** *name* **Example:** `Router(config)# crypto ca trustpoint TP1`	Specifies a local configuration name for the CA trustpoint and enters CA trustpoint configuration mode. **Note** The **crypto ca identity** command was replaced by the **crypto ca trustpoint** command.
Step 7	**enrollment url** *url* **Example:** `Router(ca-trustpoint)# enrollment url http://example.com`	Specifies a URL of the CA where your router should send certificate requests. • If you are using Simple Certificate Enrollment Protocol (SCEP) for enrollment, the URL argument must be in the form **http://***CA-name* , where *CA-name* is the host Domain Name System (DNS) name or IP address of the CA trustpoint.
Step 8	**enrollment http-proxy** *host-name port-number* **Example:** `Router(ca-trustpoint)# enrollment http-proxy example.com 8080`	(Optional) Configures the router to obtain certificates from the CA through an HTTP proxy server.
Step 9	**crl** {**query** *url* | **optional** | **best-effort**} **Example:** `Router(ca-trustpoint)# crl query ldap://example.com`	Configures the router to request a certificate revocation list (CRL), make CRL checking optional, or perform CRL checking on a "best-effort" basis. • CRLs ensure that the certificate of the peer has not been revoked. • The **crl optional** command configures the router to accept certificates even if the appropriate CRL cannot be downloaded.

	Command or Action	Purpose
		• Use the **crl query** *url* command to specify the Lightweight Directory Access Protocol (LDAP) URL of the CA server; for example, ldap://another-server.
Step 10	**primary** **Example:** `Router(ca-trustpoint)# primary`	(Optional) Specifies that this trustpoint should be used as the primary (default) trustpoint for CA requests. • Use this command if more than one CA trustpoint will be configured on this router.
Step 11	**exit** **Example:** `Router(ca-trustpoint)# exit`	Exits CA trustpoint configuration mode and returns to global configuration mode.
Step 12	**crypto ca authenticate** *name* **Example:** `Router(config)# crypto ca authenticate TP1`	Authenticates the CA by getting the public key of the CA. • Use the same name that you used when declaring the CA in the **crypto ca trustpoint** command.
Step 13	**crypto ca enrollment** *name* **Example:** `Router(config)# crypto ca enrollment TP1`	Obtains the certificate from the specified CA trustpoint. • This command requests a signed certificate from the CA for each RSA key pair.
Step 14	Do one of the following: • **copy running-config startup-config** • • **copy system:running-config nvram:startup-config** **Example:** `Router(config)# copy running-config startup-config`	Saves the configuration to NVRAM. • This command is required to save the certificates into NVRAM. If not used, the certificates would be lost at router reload. **Note** To execute EXEC mode commands in global configuration mode, you can add the **do** keyword before the command. For example, instead of **copy running-config startup-config**, you could enter **do copy running-config startup-config**.

Configuring the HTTPS Server with SSL 3.0

To disable the standard HTTP server and configure the HTTPS server with SSL 3.0, complete the procedure in this section.

Before You Begin

If a certificate authority is to be used for certification, you should declare the CA trustpoint on the routing device before enabling the secure HTTP server.

SUMMARY STEPS

1. **enable**
2. Router# **show ip http server status**
3. **configure terminal**
4. **no ip http server**
5. **ip http secure-server**
6. **ip http secure-port** *port-number*
7. **ip http secure-ciphersuite** [**3des-ede-cbc-sha**] [**rc4-128-sha**] [**rc4-128-md5**] [**des-cbc-sha**]
8. **ip http secure-client-auth**
9. **ip http secure-trustpoint** *name*
10. **end**
11. **show ip http server secure status**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Router> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	Router# **show ip http server status** **Example:** `Router# show ip http server status` **Example:**	(Optional) Displays the status of the HTTP server. • If you are unsure whether the secure HTTP server is supported in the software image you are running, enter this command and look for the line "HTTP secure server capability: {Present | Not present}". • This command displays the status of the standard HTTP server (enabled or disabled).
Step 3	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 4	**no ip http server** **Example:** `Router(config)# no ip http server`	Disables the standard HTTP server. **Note** When enabling the HTTPS server you should always disable the standard HTTP server to prevent insecure connections to the same services. This is a precautionary step (typically, the HTTP server is disabled by default).

	Command or Action	Purpose
Step 5	**ip http secure-server** **Example:** `Router(config)# ip http secure-server`	Enables the HTTPS server.
Step 6	**ip http secure-port** *port-number* **Example:** `Router(config)# ip http secure-port 1025`	(Optional) Specifies the port number that should be used for the HTTPS server. The default port number is 443. Valid options are 443 or any number in the range 1025 to 65535.
Step 7	**ip http secure-ciphersuite** [**3des-ede-cbc-sha**] [**rc4-128-sha**] [**rc4-128-md5**] [**des-cbc-sha**] **Example:** `Router(config)# ip http secure-ciphersuite rc4-128-sha rc4-128-md5`	(Optional) Specifies the CipherSuites (encryption algorithms) that should be used for encryption over the HTTPS connection. • This command allows you to restrict the list of CipherSuites that the server offers the connecting clients. For example, you may want to allow only the most secure CipherSuite to be used. • Unless you have a reason to specify the CipherSuites that should be used, or you are unfamiliar with the details of these CipherSuites, you should leave this command unconfigured and let the server and client negotiate the CipherSuite that they both support (this is the default).
Step 8	**ip http secure-client-auth** **Example:** `Router(config)# ip http secure-client-auth`	(Optional) Configures the HTTP server to request an X.509v3 certificate from the client in order to authenticate the client during the connection process. • In the default connection and authentication process, the client requests a certificate from the HTTP server, but the server does not attempt to authenticate the client. Authenticating the client provides more security than server authentication by itself, but not all clients may be configured for CA authentication.
Step 9	**ip http secure-trustpoint** *name* **Example:** `Router(config)# ip http secure-trustpoint trustpoint-01`	Specifies the CA trustpoint that should be used to obtain an X.509v3 security certificate and to authenticate the connecting client's certificate. • Use of this command assumes you have already declared a CA trustpoint using the **crypto ca trustpoint** command and associated submode commands. • Use the same trustpoint name that you used in the associated **crypto ca trustpoint** command.
Step 10	**end** **Example:** `Router(config)# end`	Ends the current configuration session and returns you to privileged EXEC mode.

	Command or Action	Purpose
Step 11	**show ip http server secure status** **Example:** `Router# show ip http server secure status`	Displays the status of the HTTP secure server configuration.

Verifying the Configuration of the HTTPS Server

To verify the configuration of the HTTPS server, connect to the router running the HTTPS server with a web browser by entering **https://***url* , where *url* is the IP address or hostname of the router. Successful connection using the **https** prefix (instead of the standard **http**) indicates that the HTTPS server is configured properly. If a port other than the default port is configured (using the **ip http secure-port** command), you must also specify the port number after the URL. For example:

```
https://209.165.202.129:1026
```
or

```
https://host.domain.com:1026
```
Generally, you can verify that the HTTPS server is configured and that you have a secure connection by locating an image of a padlock at the bottom of your browser window. Also note that secure HTTP connections have a URL that starts with “https:” instead of “http:”.

Providing Additional Security and Efficiency

The configuration of the standard HTTP server applies to the secure HTTP server as well. To provide additional security and efficiency to both the standard HTTP server and the HTTPS server, complete the procedure in this section.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **ip http path** *path-name*
4. **ip http access-class** *access-list-number*
5. **ip http max-connections** *value*
6. **ip http timeout-policy idle** *seconds* **life** *seconds* **requests** *value*

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable**	Enables privileged EXEC mode.

	Command or Action	Purpose
	Example: `Router> enable`	• Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**ip http path** *path-name* **Example:** `Router(config)# ip http path slot1:`	(Optional) Sets the base HTTP path for HTML files. • The base path is used to specify the location of the HTTP server files (HTML files) on the local system. Generally, the HTML files are located in system flash memory.
Step 4	**ip http access-class** *access-list-number* **Example:** `Router(config)# ip http access-class 20`	(Optional) Specifies the access list that should be used to allow access to the HTTP server.
Step 5	**ip http max-connections** *value* **Example:** `Router(config)# ip http max-connections 10`	(Optional) Sets the maximum number of concurrent connections to the HTTP server that will be allowed. The default value is 5.
Step 6	**ip http timeout-policy idle** *seconds* **life** *seconds* **requests** *value* **Example:** `Router(config)# ip http timeout-policy idle 30 life 120 requests 100`	(Optional) Sets the characteristics that determine how long a connection to the HTTP server should remain open. The characteristics are: • **idle** --The maximum number of seconds the connection will be kept open if no data is received or response data cannot be sent out on the connection. Note that a new value may not take effect on any already existing connections. If the server is too busy or the limit on the **life** time or the number of **requests** is reached, the connection may be closed sooner. The default value is 180 seconds (3 minutes). • **life** --The maximum number of seconds the connection will be kept open, from the time the connection is established. Note that the new value may not take effect on any already existing connections. If the server is too busy or the limit on the idle time or the number of requests is reached, it may close the connection sooner. Also, because the server will not close the connection while actively processing a request, the connection may remain open longer than the specified **life** time if processing is occurring when the life maximum is reached. In this case, the connection will be closed when processing finishes.

	Command or Action	Purpose
		The default value is 180 seconds (3 minutes). The maximum value is 86,400 seconds (24 hours). • **requests** --The maximum limit on the number of requests processed on a persistent connection before it is closed. Note that the new value may not take effect on any already existing connections. If the server is too busy or the limit on the idle time or the life time is reached, the connection may be closed before the maximum number of requests are processed. The default value is 1. The maximum value is 86,400.

Configuring the HTTPS Client with SSL 3.0

To configure the HTTPS client with SSL 3.0, complete the procedure in this section.

Before You Begin

The standard HTTP client and the secure HTTP client are always enabled.

A certificate authority is required for secure HTTP client certification; the following steps assume that you have previously declared a CA trustpoint on the routing device. If a CA trustpoint is not configured, and the remote HTTPS server requires client authentication, connections to the secure HTTP client will fail.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **ip http client secure-trustpoint** *trustpoint-name*
4. **ip http client secure-ciphersuite** [**3des-ede-cbc-sha**] [**rc4-128-sha**] [**rc4-128-md5**] [**des-cbc-sha**]
5. **end**
6. **show ip http client secure status**

DETAILED STEPS

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

	Command or Action	Purpose
Step 2	**configure terminal** **Example:** `Router# configure terminal`	Enters global configuration mode.
Step 3	**ip http client secure-trustpoint** *trustpoint-name* **Example:** `Router(config)# ip http client secure-trustpoint trustpoint01`	(Optional) Specifies the CA trustpoint that should be used if the remote HTTP server requests client authentication. • Use of this command assumes you have already declared a CA trustpoint using the **crypto ca trustpoint** command and associated submode commands. • Use the same trustpoint name that you used in the associated **crypto ca trustpoint** command. • This command is optional if client authentication is not needed, or if a primary trustpoint has been configured. If the **ip http client secure-trustpoint** command is not used, the router will use the primary trustpoint, as specified by the **primary**CA trustpoint configuration mode command.
Step 4	**ip http client secure-ciphersuite** [**3des-ede-cbc-sha**] [**rc4-128-sha**] [**rc4-128-md5**] [**des-cbc-sha**] **Example:** `Router(config)# ip http client secure-ciphersuite rc4-128-sha rc4-128-md5`	(Optional) Specifies the CipherSuites (encryption algorithms) that should be used for encryption over the HTTPS connection. • This command allows you to restrict the list of CipherSuites that the client offers when connecting to a secure HTTP server. For example, you may want to allow only the most secure CipherSuites to be used. • Unless you have a reason to specify the CipherSuites that should be used, or you are unfamiliar with the details of these CipherSuites, you should leave this command unconfigured and let the server and client negotiate the CipherSuite that they both support (this is the default).
Step 5	**end** **Example:** `Router(config)# end`	Ends the current configuration session and returns to privileged EXEC mode.
Step 6	**show ip http client secure status** **Example:** `Router# show ip http client secure status`	Displays the status of the HTTP secure server configuration.

Configuration Examples for the HTTPS--HTTP Server and Client with SSL 3.0 feature

The following example shows a configuration session in which the secure HTTP server is enabled, the port for the secure HTTP server is configured as 1025, and the remote CA trustpoint server "CA-trust-local" is used for certification.

```
Router# show ip http server status

HTTP server status: Disabled
HTTP server port: 80
HTTP server authentication method: enable
HTTP server access class: 0
HTTP server base path:
Maximum number of concurrent server connections allowed: 5
Server idle time-out: 600 seconds
Server life time-out: 600 seconds
Maximum number of requests allowed on a connection: 1
HTTP secure server capability: Present
HTTP secure server status: Disabled
HTTP secure server port: 443
HTTP secure server ciphersuite: 3des-ede-cbc-sha des-cbc-sha rc4-128-md5 rc4-12a
HTTP secure server client authentication: Disabled
HTTP secure server trustpoint:
Router# configure terminal

Enter configuration commands, one per line.  End with CNTL/Z.
Router(config)# ip http secure-server

Router(config)# ip http client secure-trustpoint CA-trust-local

Router(config)# ip http secure-port 1024

Invalid secure port value.
Router(config)# ip http secure-port 1025

Router(config)# ip http secure-ciphersuite rc4-128-sha rc4-128-md5
Router(config)# end

Router# show ip http server secure status

HTTP secure server status: Enabled

HTTP secure server port: 1025

HTTP secure server ciphersuite: rc4-128-md5 rc4-128-sha

HTTP secure server client authentication: Disabled

HTTP secure server trustpoint: CA-trust-local
```

In the following example, the CA trustpoint CA-trust-local is specified, and the HTTPS client is configured to use this trustpoint for client authentication requests:

```
Router# config terminal
Enter configuration commands, one per line.  End with CNTL/Z.
Router(config)# crypto ca trustpoint CA-trust-local

Router(ca-trustpoint)# enrollment url http://example.com

Router(ca-trustpoint)# crl query ldap://example.com

Router(ca-trustpoint)# primary

Router(ca-trustpoint)# exit
```

```
Router(config)# ip http client secure-trustpoint CA-trust-local
Router(config)# end
Router# copy running-config startup-config
```

Additional References

The following sections provide references related to the HTTPS--HTTP Server and Client with SSL 3.0 feature.

Related Documents

Related Topic	Document Title
SSL 3.0	The SSL Protocol Version 3.0 *This document is available from various sources online.*
Standard Cisco Web Client	HTTP 1.1 Web Client
Standard Cisco Web Server	HTTP 1.1 Web Server
Certification Authority Interoperability	• Configuring Certification Authority Interoperability • Certificate Autoenrollment • Certificate Enrollment Enhancements • Trustpoint CLI • Source Interface Selection for Outgoing Traffic with Certificate Authority

Standards

Standard	Title
No new or modified standards are supported by this feature.	--

Related MIBs

MIBs	MIBs Link
• No new or modified MIBs are supported by this feature and support for existing MIBs has not been modified by this feature.	To locate and download MIBs for selected platforms, Cisco IOS XE releases, and feature sets, use Cisco MIB Locator found at the following URL: http://www.cisco.com/go/mibs

Related RFCs

RFCs	Description
RFC 2616	Cisco's implementation of HTTP is based on RFC 2616: Hypertext Transfer Protocol -- HTTP/1.1 .

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 HTTPS--HTTP Server and Client with SSL 3.0

The following table provides release information about the feature or features described in this module. This table 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.

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

Table 2: Feature Information for HTTPS--HTTP Server and Client with SSL 3.0

Feature Name	Releases	Feature Information
HTTPS--HTTP Server and Client with SSL 3.0	Cisco IOS XE Release 2.1	This feature provides Secure Socket Layer (SSL) version 3.0 support for the HTTP 1.1 server and HTTP 1.1 client within Cisco IOS XE software. SSL provides server authentication, encryption, and message integrity to allow secure HTTP communications. SSL also provides HTTP client authentication. This feature is supported only in Cisco software images that support SSL. Specifically, SSL is supported in "IPSec 56" and "IPSec 3DES" images (contains "k8" or "k9" in the image name). The following commands are introduced or modified in the feature or features documented in this module. • **debug ip http ssl error** • **ip http client secure-ciphersuite** • **ip http client secure-trustpoint** • **ip http secure-ciphersuite** • **ip http secure-client-auth** • **ip http secure-port** • **ip http secure-server** • **ip http secure-trustpoint** • **show ip http client secure status** • **show ip http server secure status**

Glossary

RSA--RSA is a widely used Internet encryption and authentication system that uses public and private keys for encryption and decryption. The RSA algorithm was invented in 1978 by Ron Rivest, Adi Shamir, and Leonard Adleman. The abbreviation RSA comes from the first letter of the last names of the three original developers. The RSA algorithm is included in many applications, such as the web browsers from Microsoft and Netscape. The RSA encryption system is owned by RSA Security.

SHA --The Secure Hash Algorithm. SHA was developed by NIST and is specified in the Secure Hash Standard (SHS, FIPS 180). Often used as an alternative to Digest 5 algorithm.

signatures, digital--In the context of SSL, "signing" means to encrypt with a private key. In digital signing, one-way hash functions are used as input for a signing algorithm. In RSA signing, a 36-byte structure of two hashes (one SHA and one MD5) is signed (encrypted with the private key).

SSL 3.0--Secure Socket Layer version 3.0. SSL is a security protocol that provides communications privacy over the Internet. The protocol allows client and server applications to communicate in a way that is designed to prevent eavesdropping, tampering, or message forgery. SSL uses a program layer located between the Internet's HTTP and TCP layers. SSL is included as part of most web server products and as part of most Internet browsers.