



Cisco Content Services Switch SSL Configuration Guide

Software Version 7.40
August 2004

Corporate 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: 408 526-4100

Text Part Number: OL-5655-01



THE SPECIFICATIONS AND INFORMATION REGARDING THE PRODUCTS IN THIS MANUAL ARE SUBJECT TO CHANGE WITHOUT NOTICE. ALL STATEMENTS, INFORMATION, AND RECOMMENDATIONS IN THIS MANUAL ARE BELIEVED TO BE ACCURATE BUT ARE PRESENTED WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED. USERS MUST TAKE FULL RESPONSIBILITY FOR THEIR APPLICATION OF ANY PRODUCTS.

THE SOFTWARE LICENSE AND LIMITED WARRANTY FOR THE ACCOMPANYING PRODUCT ARE SET FORTH IN THE INFORMATION PACKET THAT SHIPPED WITH THE PRODUCT AND ARE INCORPORATED HEREIN BY THIS REFERENCE. IF YOU ARE UNABLE TO LOCATE THE SOFTWARE LICENSE OR LIMITED WARRANTY, CONTACT YOUR CISCO REPRESENTATIVE FOR A COPY.

The Cisco implementation of TCP header compression is an adaptation of a program developed by the University of California, Berkeley (UCB) as part of UCB's public domain version of the UNIX operating system. All rights reserved. Copyright © 1981, Regents of the University of California.

NOTWITHSTANDING ANY OTHER WARRANTY HEREIN, ALL DOCUMENT FILES AND SOFTWARE OF THESE SUPPLIERS ARE PROVIDED "AS IS" WITH ALL FAULTS. CISCO AND THE ABOVE-NAMED SUPPLIERS DISCLAIM ALL WARRANTIES, EXPRESSED OR IMPLIED, INCLUDING, WITHOUT LIMITATION, THOSE OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT OR ARISING FROM A COURSE OF DEALING, USAGE, OR TRADE PRACTICE.

IN NO EVENT SHALL CISCO OR ITS SUPPLIERS BE LIABLE FOR ANY INDIRECT, SPECIAL, CONSEQUENTIAL, OR INCIDENTAL DAMAGES, INCLUDING, WITHOUT LIMITATION, LOST PROFITS OR LOSS OR DAMAGE TO DATA ARISING OUT OF THE USE OR INABILITY TO USE THIS MANUAL, EVEN IF CISCO OR ITS SUPPLIERS HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

CCSP, the Cisco Square Bridge logo, Cisco Unity, Follow Me Browsing, FormShare, and StackWise are trademarks of Cisco Systems, Inc.; Changing the Way We Work, Live, Play, and Learn, and iQuick Study are service marks of Cisco Systems, Inc.; and Aironet, ASIST, BPX, Catalyst, CCDA, CCDP, CCIE, CCIP, CCNA, CCNP, Cisco, the Cisco Certified Internetwork Expert logo, Cisco IOS, Cisco Press, Cisco Systems, Cisco Systems Capital, the Cisco Systems logo, Empowering the Internet Generation, Enterprise/Solver, EtherChannel, EtherFast, EtherSwitch, Fast Step, GigaDrive, GigaStack, HomeLink, Internet Quotient, IOS, IP/TV, iQ Expertise, the iQ logo, iQ Net Readiness Scorecard, LightStream, Linksys, MeetingPlace, MGX, the Networkers logo, Networking Academy, Network Registrar, *Packet*, PIX, Post-Routing, Pre-Routing, ProConnect, RateMUX, Registrar, ScriptShare, SlideCast, SMARTnet, StrataView Plus, SwitchProbe, TeleRouter, The Fastest Way to Increase Your Internet Quotient, TransPath, and VCO are registered trademarks of Cisco Systems, Inc. and/or its affiliates in the United States and certain other countries.

All other trademarks mentioned in this document or Website are the property of their respective owners. The use of the word partner does not imply a partnership relationship between Cisco and any other company. (0406R)

Cisco Content Services Switch SSL Configuration Guide
Copyright © 2004, Cisco Systems, Inc. All rights reserved.



Preface xv

Audience xvi

How to Use This Guide xvi

Related Documentation xvii

Symbols and Conventions xxi

Obtaining Documentation xxii

 Cisco.com xxii

 Ordering Documentation xxii

Documentation Feedback xxiii

Obtaining Technical Assistance xxiii

 Cisco Technical Support Website xxiii

 Submitting a Service Request xxiv

 Definitions of Service Request Severity xxiv

Obtaining Additional Publications and Information xxv

CHAPTER 1

Overview of CSS SSL 1-1

SSL Cryptography Overview 1-1

 SSL Public Key Infrastructure Overview 1-2

 Confidentiality 1-3

 Authentication 1-4

 Message Integrity 1-4

 SSL Module Cryptography Capabilities 1-6

Overview of the SSL Module Functions in the CSS 1-7

SSL Termination 1-8

Client Authentication 1-9

Back-End SSL 1-11

SSL Initiation 1-12

CHAPTER 2

SSL Configuration Quick Starts 2-1

RSA Certificate and Key Generation Quick Start 2-2

RSA Certificate and Key Import Quick Start 2-5

SSL Proxy List Quick Start 2-6

SSL Termination Proxy List Quick Start 2-6

Back-End SSL Proxy List Quick Start 2-8

SSL Initiation Proxy List Quick Start 2-10

SSL Service and Content Rule Quick Start 2-13

SSL Termination Service and Content Rule Quick Start 2-13

Back-End SSL Service and Content Rule Quick Start 2-15

SSL Initiation Service Quick Start 2-18

SSL Initiation Content Rule Quick Start 2-20

CHAPTER 3

Configuring SSL Certificates and Keys 3-1

Overview of SSL Certificates and Keys 3-1

Generating Certificates and Private Keys in the CSS 3-4

Generating an RSA Key Pair 3-5

Generating a DSA Key Pair 3-6

Generating Diffie-Hellman Key Parameters 3-7

Using an RSA Key to Generate a Certificate Signing Request 3-8

Generating a Self-Signed Certificate 3-10

Preparing a Global Site Certificate 3-11

Importing or Exporting Certificates and Private Keys	3-12
Configuring the Default SFTP or FTP Server to Import Certificates and Private Keys	3-13
Transferring Certificates and Private Keys to the CSS	3-14
Associating Certificate and Private Key Files with Names	3-16
Associating a Certificate with a File	3-17
Associating an RSA Key Pair with a File	3-17
Associating a DSA Key Pair with a File	3-18
Associating Diffie-Hellman Parameters with a File	3-19
Verifying a Certificate Against a Key Pair	3-20
Removing Certificates and Private Keys from the CSS	3-20

CHAPTER 4

Configuring SSL Termination	4-1
Overview of SSL Termination	4-1
Creating an SSL Proxy List	4-2
Adding a Description to an SSL Proxy List	4-3
Configuring Virtual SSL Servers for an SSL Proxy List	4-4
Creating an SSL Server Index	4-6
Specifying a Virtual IP Address	4-6
Specifying a Virtual Port	4-7
Assigning Certificate, Key, and Cipher Suites for Server Authentication	4-8
Specifying the RSA Certificate Name	4-8
Specifying the RSA Key Pair Name	4-9
Specifying the DSA Certificate Name	4-9
Specifying the DSA Key Pair Name	4-10
Specifying the Diffie-Hellman Parameter Filename	4-10
Specifying Cipher Suites	4-11
Configuring Client Authentication	4-15
Enabling Client Authentication	4-16
Specifying CA Certificates for Client Certificate Verification	4-16

- Configuring a CRL Record **4-17**
- Assigning a CRL Record to the Virtual SSL Server **4-19**
- Handling Client Authentication Failures **4-19**
- Configuring HTTP Header Insertion **4-20**
 - Inserting Client Certificate Information **4-21**
 - Inserting Server Certificate Information **4-25**
 - Inserting Session Information **4-30**
 - Adding a Prefix to the Fields Inserted in the HTTP Header **4-32**
 - Inserting a Static Text String **4-32**
- Specifying SSL or TLS Version **4-33**
- Terminating a Client Connection with a TCP FIN Message Only **4-34**
- Specifying Secure URL Rewrite **4-34**
- Specifying SSL Session Cache Timeout **4-37**
- Specifying SSL Session Handshake Renegotiation **4-38**
- Configuring the Delay Time for SSL Queued Data **4-40**
- Specifying SSL TCP Client-Side Connection Timeout Values **4-41**
 - Specifying a TCP SYN Timeout Value (Client-Side Connection) **4-41**
 - Specifying a TCP Inactivity Timeout Value (Client-Side Connection) **4-42**
- Specifying SSL TCP Server-Side Connection Timeout Values **4-42**
 - Specifying a TCP SYN Timeout Value (Server-Side Connection) **4-43**
 - Specifying a TCP Inactivity Timeout Value (Server-Side Connection) **4-43**
- Specifying the Nagle Algorithm for SSL TCP Connections **4-44**
- Specifying the TCP Buffering for SSL TCP Connections **4-45**
- Activating and Suspending an SSL Proxy List **4-46**
- Configuring a Service for SSL Termination **4-47**
 - Creating an SSL Service **4-48**
 - Specifying the SSL Acceleration Service Type **4-48**
 - Adding an SSL Proxy List to an SSL Termination Service **4-49**
 - Specifying the SSL Module Slot **4-49**
 - Disabling Keepalive Messages for the SSL Module **4-50**

Specifying the SSL Session ID Cache Size	4-50
Activating the SSL Service	4-51
Suspending the SSL Service	4-52
Configuring a Content Rule for SSL Termination	4-52

CHAPTER 5

Configuring Back-End SSL	5-1
Overview of Back-End SSL	5-1
Creating an SSL Proxy List	5-2
Adding a Description to an SSL Proxy List	5-3
Configuring Back-End SSL Servers in an SSL Proxy List	5-3
Creating a Back-End SSL Server in an SSL Proxy List	5-5
Configuring a Back-End SSL Server Type	5-5
Configuring the VIP Address for an SSL Back-End Server	5-6
Configuring the Virtual Port	5-6
Configuring the Server IP Address	5-7
Configuring the Server Port	5-7
Configuring SSL Version	5-8
Configuring the Available Cipher Suites	5-8
Configuring SSL Session Cache Timeout	5-9
Configuring SSL Session Handshake Renegotiation	5-10
Configuring TCP Virtual Client Connections Timeout Values	5-11
Specifying a TCP SYN Timeout Value for the Virtual Client Connection	5-11
Specifying a TCP Inactivity Timeout for a Virtual Client Connection	5-12
Configuring TCP Server-Side Connection Timeout Values on the SSL Module	5-13
Specifying a TCP SYN Timeout Value for a Server-Side Connection	5-13
Specifying a TCP Inactivity Timeout for a Server-Side Connection	5-14
Specifying the Nagle Algorithm for SSL TCP Connections	5-15
Specifying the TCP buffering for SSL TCP Connections	5-16

- Activating and Suspending an SSL Proxy List **5-17**
- Configuring a Service for Back-End SSL **5-18**
 - Creating an SSL Service **5-19**
 - Configuring the Back-End SSL Service Type **5-19**
 - Adding an SSL Proxy List for a Back-End SSL Server **5-19**
 - Configuring an IP Address for a Back-End SSL Service **5-20**
 - Configuring the Port Number for a Back-End SSL Service **5-20**
 - Activating the SSL Service **5-21**
 - Suspending the SSL Service **5-22**
- Configuring a Content Rule for Back-End SSL **5-22**

CHAPTER 6

Configuring SSL Initiation 6-1

- Overview of SSL Initiation **6-1**
- Creating an SSL Initiation Proxy List **6-3**
- Adding a Description to an SSL Initiation Proxy List **6-4**
- Configuring Back-End SSL Servers in an SSL Initiation Proxy List **6-4**
 - Creating a Back-End Server in an SSL Initiation Proxy List **6-6**
 - Configuring the Back-End Server as an SSL Initiation Server **6-6**
 - Configuring an IP Address for the SSL Initiation Server **6-7**
 - Configuring a Port for the SSL Initiation Server **6-7**
 - Configuring the SSL Server IP Address **6-8**
 - Configuring the SSL Server Port **6-8**
 - Configuring SSL Version **6-9**
 - Configuring the Available Cipher Suites **6-9**
 - Configuring SSL Session Cache Timeout **6-11**
 - Configuring SSL Session Handshake Renegotiation **6-11**

Configuring TCP Virtual Client Connections Timeout Values	6-13
Specifying a TCP SYN Timeout Value for the Virtual Client Connection	6-13
Specifying a TCP Inactivity Timeout for a Virtual Client Connection	6-14
Specifying the Nagle Algorithm for Client-Side Connections	6-15
Configuring TCP Server-Side Connection Timeout Values on the SSL Module	6-15
Specifying a TCP SYN Timeout Value for a Server-Side Connection	6-16
Specifying a TCP Inactivity Timeout for a Server-Side Connection	6-17
Specifying the Nagle Algorithm for Server-Side Connections	6-17
Specifying the TCP Buffering for SSL TCP Connections	6-18
Configuring Client Certificates and Keys	6-19
Configuring the RSA Certificate Name	6-20
Configuring the RSA Key Name	6-20
Configuring Diffie Hellman Parameters	6-20
Configuring the DSA Certificate Name	6-21
Configuring the DSA Key Filename	6-21
Configuring CA Certificates for Server Authentication	6-21
Activating and Suspending an SSL Proxy List	6-23
Configuring a Service for SSL Initiation	6-24
Creating an SSL Service	6-25
Configuring the SSL Service Type	6-25
Configuring an IP Address for an SSL Initiation Service	6-25
Adding an SSL Proxy List to an SSL Initiation Service	6-26
Specifying the SSL Module Slot	6-26
Configuring the SSL Initiation Service Keepalive Type	6-27
SSL Session ID Cache Size	6-28
Activating the SSL Service	6-28
Suspending the SSL Service	6-29

Configuring a Content Rule for SSL Initiation 6-29

Troubleshooting SSL Initiation 6-30

CHAPTER 7

Displaying SSL Configuration Information and Statistics 7-1

Showing Certificate and Key Pair Information 7-1

Showing SSL Certificates 7-2

Showing SSL RSA Private Keys 7-4

Showing SSL DSA Private Keys 7-5

Showing SSL Diffie-Hellman Parameters 7-6

Showing SSL Associations 7-7

Showing SSL Certificates, Key Pairs, and Diffie-Hellman Parameter Files 7-8

Showing SSL Proxy Configuration Information 7-9

Showing CRL Record Configuration 7-14

Showing SSL URL Rewrite Statistics 7-15

Showing SSL Module Statistics 7-16

Clearing SSL Statistics 7-24

Showing SSL Flows 7-24

CHAPTER 8

Examples of CSS SSL Configurations 8-1

Processing of SSL Flows by the SSL Module 8-1

SSL Transparent Proxy Configuration — One SSL Module 8-5

SSL Transparent Proxy Configuration — Two SSL Modules 8-8

SSL Transparent Proxy Configuration — HTTP and Back-End SSL Servers 8-12

SSL Full Proxy Configuration — One SSL Module 8-17

SSL Initiation Configurations 8-21

SSL Tunnel to Four Data Centers 8-21

SSL Tunnel to One Data Center with Server Authentication 8-25

INDEX



<i>Figure 1-1</i>	SSL Handshake Without Client Authentication	1-9
<i>Figure 1-2</i>	SSL Handshake With Client Authentication	1-10
<i>Figure 3-1</i>	SSL Key and Server Certificate Configuration Overview	3-3
<i>Figure 4-1</i>	SSL Termination	4-2
<i>Figure 4-2</i>	Cipher Suite Algorithms	4-11
<i>Figure 5-1</i>	Back-End SSL with SSL Termination	5-2
<i>Figure 6-1</i>	SSL Initiation with an SSL Server	6-2
<i>Figure 6-2</i>	SSL Initiation with a Second CSS Running SSL Termination	6-2
<i>Figure 8-1</i>	CSS Configuration with Multiple SSL Modules	8-2
<i>Figure 8-2</i>	CSS Configuration with a Back-End SSL Server	8-4
<i>Figure 8-3</i>	Transparent Proxy Configuration with a Single SSL Module	8-6
<i>Figure 8-4</i>	Transparent Proxy Configuration with Two SSL Modules	8-9
<i>Figure 8-5</i>	SSL Transparent Proxy Configuration - HTTP and Back-End SSL Servers	8-13
<i>Figure 8-6</i>	Full Proxy Configuration Using a Single SSL Module	8-18
<i>Figure 8-7</i>	SSL Initiation Between a CSS and Four Data Centers	8-22
<i>Figure 8-8</i>	SSL Initiation Between a CSS and One Data Center	8-26



<i>Table 1-1</i>	SSL Module SSL Cryptography Capabilities	1-6
<i>Table 2-1</i>	RSA Certificate and Key Generation Quick Start	2-2
<i>Table 2-2</i>	RSA Certificate and Key Import Quick Start	2-5
<i>Table 2-3</i>	SSL Termination Proxy List Quick Start	2-7
<i>Table 2-4</i>	Back-End SSL Proxy List Quick Start	2-9
<i>Table 2-5</i>	SSL Initiation Proxy List Quick Start	2-11
<i>Table 2-6</i>	SSL Server Service and Content Rule Quick Start	2-13
<i>Table 2-7</i>	Back-End SSL Service and Content Rule Quick Start	2-15
<i>Table 2-8</i>	SSL Initiation Service Quick Start	2-18
<i>Table 2-9</i>	SSL Initiation Content Rule Quick Start	2-20
<i>Table 4-1</i>	SSL Cipher Suites Supported by the CSS	4-13
<i>Table 4-2</i>	Client Certificate Fields Inserted in the HTTP Header	4-22
<i>Table 4-3</i>	Server Certificate Fields Inserted In the HTTP Header	4-26
<i>Table 4-4</i>	SSL Session Fields Inserted In the HTTP Header	4-31
<i>Table 7-1</i>	Field Descriptions for the show ssl associate cert Command	7-2
<i>Table 7-2</i>	Field Descriptions for the show ssl associate cert certname Command	7-3
<i>Table 7-3</i>	Field Descriptions for the show ssl associate rsakey Command	7-5
<i>Table 7-4</i>	Field Descriptions for the show ssl associate dsakey Command	7-6
<i>Table 7-5</i>	Field Descriptions for the show ssl associate dhparam Command	7-6
<i>Table 7-6</i>	Field Descriptions for the show ssl files Command	7-8
<i>Table 7-7</i>	Field Descriptions for the show ssl-proxy-list Command	7-10
<i>Table 7-8</i>	Field Descriptions for the show ssl-proxy-list Command	7-10
<i>Table 7-9</i>	Field Descriptions for the show ssl crl-record Command	7-14

<i>Table 7-10</i>	Field Descriptions for the show ssl urlrewrite Command	7-15
<i>Table 7-11</i>	Field Descriptions for the show ssl statistics Command	7-17
<i>Table 7-12</i>	Field Descriptions for the show ssl flows Command	7-25



Preface

This guide provides instructions for configuring the SSL features of the Cisco 11500 Series Content Services Switches (CSS). Information in this guide applies to all CSS models except where noted.

The CSS software is available in a Standard or optional Enhanced feature set. Proximity Database and Secure Management, which includes Secure Shell Host and SSL strong encryption for the Device Management User Interface software, are optional features.

This preface contains the following major sections:

- [Audience](#)
- [How to Use This Guide](#)
- [Related Documentation](#)
- [Symbols and Conventions](#)
- [Obtaining Documentation](#)
- [Documentation Feedback](#)
- [Obtaining Technical Assistance](#)
- [Obtaining Additional Publications and Information](#)

Audience

This guide is intended for the following trained and qualified service personnel who are responsible for configuring the CSS:

- Web master
- System administrator
- System operator

How to Use This Guide

This guide is organized as follows:

Chapter	Description
Chapter 1, Overview of CSS SSL	Overview of SSL cryptography and the CSS SSL features.
Chapter 2, SSL Configuration Quick Starts	Configure the CSS SSL features quickly and easily using procedural tables with command examples followed by running-config examples for each table.
Chapter 3, Configuring SSL Certificates and Keys	Import, create, and associate certificates and key pairs.
Chapter 4, Configuring SSL Termination	Configure the CSS and the SSL Acceleration Module to perform Secure Sockets Layer (SSL) termination between the client and the Web servers.
Chapter 5, Configuring Back-End SSL	Configure the CSS and the SSL Acceleration Module to accept SSL encrypted data from a client, decrypt the data to make a load-balancing decision, then reencrypt the data and send it to a back-end SSL server.

Chapter	Description
Chapter 6, Configuring SSL Initiation	Configure the CSS and the SSL Acceleration Module to accept clear text from a client and perform SSL encryption between the CSS and the Web servers.
Chapter 7, Displaying SSL Configuration Information and Statistics	Display data and statistics related to your CSS SSL configuration.
Chapter 8, Examples of CSS SSL Configurations	Description of SSL flow processing in the CSS and running-config examples of the various SSL configurations available on a CSS.

Related Documentation

In addition to this guide, the Content Services Switch documentation includes the following publications.

Document Title	Description
<i>Release Note for the Cisco 11500 Series Content Services Switch</i>	This release note provides information on operating considerations, caveats, and command line interface (CLI) commands for the Cisco 11500 series CSS.
<i>Cisco 11500 Series Content Services Switch Hardware Installation Guide</i>	This guide provides information for installing, cabling, and powering the Cisco 11500 series CSS. In addition, this guide provides information about CSS specifications, cable pinouts, and hardware troubleshooting.

Document Title	Description
<i>Cisco Content Services Switch Getting Started Guide</i>	<p>This guide describes how to perform initial administration and configuration tasks on the CSS, including:</p> <ul style="list-style-type: none"> • Booting the CSS for the first time and a routine basis, and logging in to the CSS • Configuring the username and password, Ethernet management port, static IP routes, and the date and time • Configuring DNS server for hostname resolution • Configuring sticky cookies with a sticky overview and advanced load-balancing method using cookies • A task list to help you find information in the CSS documentation • Troubleshooting the boot process
<i>Cisco Content Services Switch Administration Guide</i>	<p>This guide describes how to perform administrative tasks on the CSS, including upgrading your CSS software and configuring the following:</p> <ul style="list-style-type: none"> • Logging, including displaying log messages and interpreting sys.log messages • User profile and CSS parameters • SNMP • RMON • XML documents to configure the CSS • CSS scripting language • Offline Diagnostic Monitor (Offline DM) menu

Document Title	Description
<i>Cisco Content Services Switch Routing and Bridging Configuration Guide</i>	<p>This guide describes how to perform routing and bridging configuration tasks on the CSS, including:</p> <ul style="list-style-type: none">• Management ports, interfaces, and circuits• Spanning-tree bridging• Address Resolution Protocol (ARP)• Routing Information Protocol (RIP)• Internet Protocol (IP)• Open Shortest Path First (OSPF) protocol• Cisco Discovery Protocol (CDP)• Dynamic Host Configuration Protocol (DHCP) relay agent
<i>Cisco Content Services Switch Content Load-Balancing Configuration Guide</i>	<p>This guide describes how to perform CSS content load-balancing configuration tasks, including:</p> <ul style="list-style-type: none">• Flow and port mapping• Services• Service, global, and script keepalives• Source groups• Loads for services• Dynamic Feedback Protocol (DFP)• Owners• Content rules• Sticky parameters• HTTP header load balancing• Content caching• Content replication

Document Title	Description
<i>Cisco Content Services Switch Global Server Load-Balancing Configuration Guide</i>	<p>This guide describes how to perform CSS global load-balancing configuration tasks, including:</p> <ul style="list-style-type: none"> • Domain Name System (DNS) • DNS Sticky • Content Routing Agent • Client-Side Accelerator • Network proximity
<i>Cisco Content Services Switch Redundancy Configuration Guide</i>	<p>This guide describes how to perform CSS redundancy configuration tasks, including:</p> <ul style="list-style-type: none"> • VIP and virtual interface redundancy • Adaptive session redundancy • Box-to-box redundancy
<i>Cisco Content Services Switch Security Configuration Guide</i>	<p>This guide describes how to perform CSS security configuration tasks, including:</p> <ul style="list-style-type: none"> • Controlling access to the CSS • Secure Shell Daemon protocol • Radius • TACACS+ • Firewall load balancing
<i>Cisco Content Services Switch Command Reference</i>	<p>This reference provides an alphabetical list of all CLI commands including syntax, options, and related commands.</p>
<i>Cisco Content Services Switch Device Management User's Guide</i>	<p>This guide describes how to use the Device Management user interface, an HTML-based Web-based application that you use to configure and manage your CSS.</p>

Symbols and Conventions

This guide uses the following symbols and conventions to identify different types of information.



Caution

A caution means that a specific action you take could cause a loss of data or adversely impact use of the equipment.



Warning

A warning describes an action that could cause you physical harm or damage the equipment.



Note

A note provides important related information, reminders, and recommendations.

Bold text indicates a command in a paragraph.

Courier text indicates text that appears on a command line, including the CLI prompt.

Courier bold text indicates commands and text you enter in a command line.

Italics text indicates the first occurrence of a new term, book title, emphasized text, and variables for which you supply values.

1. A numbered list indicates that the order of the list items is important.
 - a. An alphabetical list indicates that the order of the secondary list items is important.
- A bulleted list indicates that the order of the list topics is unimportant.
 - An indented list indicates that the order of the list subtopics is unimportant.

Obtaining Documentation

Cisco documentation and additional literature are available on Cisco.com. Cisco also provides several ways to obtain technical assistance and other technical resources. These sections explain how to obtain technical information from Cisco Systems.

Cisco.com

You can access the most current Cisco documentation at this URL:

<http://www.cisco.com/univercd/home/home.htm>

You can access the Cisco website at this URL:

<http://www.cisco.com>

You can access international Cisco websites at this URL:

http://www.cisco.com/public/countries_languages.shtml

Ordering Documentation

You can find instructions for ordering documentation at this URL:

http://www.cisco.com/univercd/cc/td/doc/es_inpck/pdi.htm

You can order Cisco documentation in these ways:

- Registered Cisco.com users (Cisco direct customers) can order Cisco product documentation from the Ordering tool:

<http://www.cisco.com/en/US/partner/ordering/index.shtml>

- Nonregistered Cisco.com users can order documentation through a local account representative by calling Cisco Systems Corporate Headquarters (California, USA) at 408 526-7208 or, elsewhere in North America, by calling 800 553-NETS (6387).

Documentation Feedback

You can send comments about technical documentation to bug-doc@cisco.com.

You can submit comments by using the response card (if present) behind the front cover of your document or by writing to the following address:

Cisco Systems
Attn: Customer Document Ordering
170 West Tasman Drive
San Jose, CA 95134-9883

We appreciate your comments.

Obtaining Technical Assistance

For all customers, partners, resellers, and distributors who hold valid Cisco service contracts, Cisco Technical Support provides 24-hour-a-day, award-winning technical assistance. The Cisco Technical Support Website on Cisco.com features extensive online support resources. In addition, Cisco Technical Assistance Center (TAC) engineers provide telephone support. If you do not hold a valid Cisco service contract, contact your reseller.

Cisco Technical Support Website

The Cisco Technical Support Website provides online documents and tools for troubleshooting and resolving technical issues with Cisco products and technologies. The website is available 24 hours a day, 365 days a year at this URL:

<http://www.cisco.com/techsupport>

Access to all tools on the Cisco Technical Support Website requires a Cisco.com user ID and password. If you have a valid service contract but do not have a user ID or password, you can register at this URL:

<http://tools.cisco.com/RPF/register/register.do>

Submitting a Service Request

Using the online TAC Service Request Tool is the fastest way to open S3 and S4 service requests. (S3 and S4 service requests are those in which your network is minimally impaired or for which you require product information.) After you describe your situation, the TAC Service Request Tool automatically provides recommended solutions. If your issue is not resolved using the recommended resources, your service request will be assigned to a Cisco TAC engineer. The TAC Service Request Tool is located at this URL:

<http://www.cisco.com/techsupport/servicerequest>

For S1 or S2 service requests or if you do not have Internet access, contact the Cisco TAC by telephone. (S1 or S2 service requests are those in which your production network is down or severely degraded.) Cisco TAC engineers are assigned immediately to S1 and S2 service requests to help keep your business operations running smoothly.

To open a service request by telephone, use one of the following numbers:

Asia-Pacific: +61 2 8446 7411 (Australia: 1 800 805 227)

EMEA: +32 2 704 55 55

USA: 1 800 553 2447

For a complete list of Cisco TAC contacts, go to this URL:

<http://www.cisco.com/techsupport/contacts>

Definitions of Service Request Severity

To ensure that all service requests are reported in a standard format, Cisco has established severity definitions.

Severity 1 (S1)—Your network is “down,” or there is a critical impact to your business operations. You and Cisco will commit all necessary resources around the clock to resolve the situation.

Severity 2 (S2)—Operation of an existing network is severely degraded, or significant aspects of your business operation are negatively affected by inadequate performance of Cisco products. You and Cisco will commit full-time resources during normal business hours to resolve the situation.

Severity 3 (S3)—Operational performance of your network is impaired, but most business operations remain functional. You and Cisco will commit resources during normal business hours to restore service to satisfactory levels.

Severity 4 (S4)—You require information or assistance with Cisco product capabilities, installation, or configuration. There is little or no effect on your business operations.

Obtaining Additional Publications and Information

Information about Cisco products, technologies, and network solutions is available from various online and printed sources.

- Cisco Marketplace provides a variety of Cisco books, reference guides, and logo merchandise. Visit Cisco Marketplace, the company store, at this URL:
<http://www.cisco.com/go/marketplace/>
- The Cisco *Product Catalog* describes the networking products offered by Cisco Systems, as well as ordering and customer support services. Access the Cisco Product Catalog at this URL:
<http://cisco.com/univercd/cc/td/doc/pcat/>
- *Cisco Press* publishes a wide range of general networking, training and certification titles. Both new and experienced users will benefit from these publications. For current Cisco Press titles and other information, go to Cisco Press at this URL:
<http://www.ciscopress.com>
- *Packet* magazine is the Cisco Systems technical user magazine for maximizing Internet and networking investments. Each quarter, Packet delivers coverage of the latest industry trends, technology breakthroughs, and Cisco products and solutions, as well as network deployment and troubleshooting tips, configuration examples, customer case studies, certification and training information, and links to scores of in-depth online resources. You can access Packet magazine at this URL:
<http://www.cisco.com/packet>

- *iQ Magazine* is the quarterly publication from Cisco Systems designed to help growing companies learn how they can use technology to increase revenue, streamline their business, and expand services. The publication identifies the challenges facing these companies and the technologies to help solve them, using real-world case studies and business strategies to help readers make sound technology investment decisions. You can access iQ Magazine at this URL:

<http://www.cisco.com/go/iqmagazine>

- *Internet Protocol Journal* is a quarterly journal published by Cisco Systems for engineering professionals involved in designing, developing, and operating public and private internets and intranets. You can access the Internet Protocol Journal at this URL:

<http://www.cisco.com/ipj>

- World-class networking training is available from Cisco. You can view current offerings at this URL:

<http://www.cisco.com/en/US/learning/index.html>



Overview of CSS SSL

Secure Sockets Layer (SSL) is an application-level protocol that provides encryption technology for the Internet, ensuring secure transactions such as the transmission of credit card numbers for e-commerce Web sites. SSL provides the secure transaction of data between a client and a server through a combination of privacy, authentication, and data integrity. SSL relies upon certificates, private-public key exchange pairs, and Diffie-Hellman key agreement parameters for this level of security.

This chapter contains the following major sections:

- [SSL Cryptography Overview](#)
- [Overview of the SSL Module Functions in the CSS](#)

SSL Cryptography Overview

The CSS uses the SSL Acceleration Module and a special set of SSL commands to perform the SSL cryptographic functions between a client and a server. The SSL functions include client and server authentication, private-key and public-key generation, certificate management, and data packet encryption and decryption.

The SSL module supports SSL version 3.0 and Transport Layer Security (TLS) version 1.0. The module understands and accepts an SSL version 2.0 ClientHello message to allow dual version clients to communicate with the CSS through the SSL module. In this case, the client indicates an SSL version of 3.0 in the version 2.0 ClientHello, which informs the SSL module that the client can support SSL version 3.0. The SSL module returns a version 3.0 ServerHello message.

**Note**

Although there are very few clients on the market today that support only SSL version 2.0, the SSL module will be unable to pass network traffic if the client supports only version 2.0.

A typical SSL session with the SSL module requires encryption ciphers to establish and maintain the secure connection. Cipher suites provide the cryptographic algorithms required by the SSL module to perform key exchange, authentication, and Message Authentication Code (MAC). See the “[Specifying Cipher Suites](#)” section in [Chapter 3, Configuring SSL Certificates and Keys](#) for details about the supported cipher suites.

This section provides an overview on SSL cryptography as implemented through the SSL module in the CSS. It covers:

- [SSL Public Key Infrastructure Overview](#)
- [SSL Module Cryptography Capabilities](#)

SSL Public Key Infrastructure Overview

SSL provides authentication, encryption, and data integrity in a Public Key Infrastructure (PKI). PKI is a set of policies and procedures to establish a secure information exchange between devices. Three fundamental elements characterize the PKIs used in asymmetric cryptography. These three elements provide a secure system for deploying e-commerce and a reliable environment for building virtually any type of electronic transactions, from corporate intranets to Internet-based e-business applications.

These elements include:

- Confidentiality
- Authentication
- Message integrity

Confidentiality

Confidentiality means that unintended users cannot view the data. In PKIs, confidentiality is achieved by encrypting the data through a variety of methods. In SSL, specifically, large amounts of data are encrypted using one or more symmetric keys that are known only by the two endpoints. Because the symmetric key is usually generated by one of the endpoints, it must be transmitted securely to the other endpoint. Secure transmittal of a symmetric key is generally achieved by two mechanisms, *key exchange* or *key agreement*.

Key exchange is the most common of these two secure transmittal mechanisms. In key exchange, one device generates the symmetric key and then encrypts it using an asymmetric encryption scheme before transmitting it to the other side. Asymmetric encryption requires that both devices have a public key and a private key. The two keys are mathematically related; data that can be encrypted by the public key can be decrypted by the private key, and vice versa. The most commonly used key exchange algorithm is the Rivest Shamir Adelman (RSA) algorithm.

For SSL, the sender encrypts the symmetric keys with the public key of the receiver. This ensures that the private key of the receiver is the only key that can decrypt the transmission. The security of asymmetric encryption depends entirely on the fact that the private key is known only by the owner and not by any other party. If this key were compromised for any reason, a fraudulent Web user (or Web site) could decrypt the stream containing the symmetric key and the entire data transfer.

In *key agreement*, the two sides involved in a data exchange cooperate to generate a symmetric (shared) key. The most common key agreement algorithm is the Diffie-Hellman algorithm. Diffie-Hellman depends on certain parameters to generate the shared key that is calculated and exchanged between the client and the server.

Authentication

Authentication is necessary for one or more devices in the exchange to verify that the party to whom they are talking is really who they claim to be. For example, assume a client is connecting to an e-commerce website. Before sending sensitive information such as a credit card number, the client verifies that the server is an e-commerce website. In certain instances, it may be necessary for both the client and the server to authenticate themselves to each other before beginning the transaction. In a financial transaction between two banks, both the client and the server need to be confident that the other is who they say they are. SSL facilitates this authentication through the use of digital certificates.

Digital certificates are a form of digital identification to prove the identity of the client to the server. A Certificate Authority (CA) issues digital certificates in the context of a PKI, which uses public-key and private-key encryption to ensure security. CAs are trusted authorities who “sign” certificates to verify their authenticity. Clients or servers connected to the CSS must have trusted certificates from the same CA, or from different CAs in a hierarchy of trusted relationships (for example, “A” trusts “B,” and “B” trusts “C,” therefore “A” trusts “C”).

A certificate ensures that the identification information is correct, and that the public key actually belongs to that client or server. Digital certificates contain information such as details about the owner, details about the certificate issuer, the owner’s public key, validity and expiration dates, and associated privileges.

Upon receiving a certificate, a client can connect to the certificate issuer and verify the validity of the certificate using the issuer’s public key. This ensures that the certificate is actually issued and signed by an authorized entity. A certificate remains valid until it expires or is terminated.

Message Integrity

Message integrity is a means of assuring the recipient of a message that the contents of the message have not been tampered with during transit. SSL achieves this by applying a message digest to the data before transmitting it. A message digest is a function that takes an arbitrary length message and outputs a fixed-length string that is characteristic of the message.

An important property of the message digest is that it is extremely difficult to reverse. Simply appending a digest of the message to itself before sending it is not enough to guarantee integrity. An attacker can change the message and then change the digest accordingly. Encoding the message digest with the sender's private key creates a Message Authentication Code (MAC), the message integrity algorithm, which the recipient can then decode using the sender's public key. SSL supports two different algorithms for a MAC: Message Digest 5 (MD5) and Secure Hash Algorithm (SHA).

This integrity scheme, however, does not work if the sender's private key is compromised. The attacker can now forge the sender's MACs. Message integrity also depends heavily on the protection of private keys. This process is known as digital signing.

RSA key pairs are effective for signing the MAC. However, it may be advantageous to separate the functions of key exchange and signing. The Digital Signature Algorithm (DSA) is an SSL algorithm that is used strictly for digital signatures but not for key exchange.

DSA was standardized as FIPS-186, which is the Digital Signature Standard (DSS). DSA and DSS can be used interchangeably. DSS uses the same crypto-math as Diffie-Hellman and requires parameters similar to Diffie-Hellman to generate keys. Additionally, DSS is restricted for use only with the Secure Hash Algorithm 1 (SHA-1) message digest.

SSL Module Cryptography Capabilities

[Table 1-1](#) provides information on the SSL cryptography capabilities of the SSL module.

Table 1-1 *SSL Module SSL Cryptography Capabilities*

SSL Cryptography Function	Functions Supported by the SSL Module
SSL versions	SSL version 3.0 and Transport Layer Security (TLS) version 1.0
Public key exchange and key agreement algorithms	<ul style="list-style-type: none"> • RSA - 512-bit, 768-bit, 1024-bit, and 2048-bit (key exchange and key agreement algorithm) • DSA - 512-bit, 768-bit, and 1024-bit (certificate signing algorithm) • Diffie-Hellman - 512-bit, 768-bit, 1024-bit, and 2048-bit (key agreement algorithm)
Encryption types	<ul style="list-style-type: none"> • Data Encryption Standard (DES) • Triple-Strength Data Encryption Standard (3DES) • RC4 <p>See Table 4-1 in Chapter 4, Configuring SSL Termination for a list of supported cipher suites and key encryption types.</p>
Hash types	<ul style="list-style-type: none"> • SSL MAC-MD5 • SSL MAC-SHA1 <p>See Table 4-1 in Chapter 4, Configuring SSL Termination for a list of supported cipher suites and hash types.</p>

Table 1-1 SSL Module SSL Cryptography Capabilities (continued)

SSL Cryptography Function	Functions Supported by the SSL Module
Digital certificates	<p>The SSL module supports all major digital certificates from Certificate Authorities (CAs), including those listed below:</p> <ul style="list-style-type: none">• VeriSign• Entrust• Netscape iPlanet• Windows 2000 Certificate Server• Thawte• Equifax• Genuity

Overview of the SSL Module Functions in the CSS

The CSS 11503 and CSS 11506 support multiple SSL modules; a maximum of two in a CSS 11503 and a maximum of four in a CSS 11506. The CSS 11501 supports a single integrated SSL module.

The SSL module is responsible for all user authentication, public/private key generation, certificate management, and packet encryption and decryption functions between the client and the server. It is dependent on the Switch Module to provide the interface for processing network traffic and the Switch Control Module (SCM) to send and receive configuration information.

The CSS stores all certificates and keys on the SCM disk. The CSS supports a maximum of 256 certificates and 256 key-pairs per SSL module, which equals approximately 3 MB of storage space on the disk. The CSS stores all certificate- and key-related files in a secure location on the disk. When processing connections, the CSS loads the certificates and keys into volatile memory on the SSL module for faster access.

No network traffic is sent to an SSL module from the SCM until an SSL content rule is activated to:

- Define where the content physically resides
- Specify where to direct the request for content (which service)
- Specify which load-balancing method to use

An SSL proxy list determines the flow of information to and from an SSL module. An entry in the proxy list defines the flow from a client to an SSL module. An entry also defines a flow from an SSL module to a back-end SSL server. To define how an SSL module processes SSL requests for content, add an SSL proxy list to an SSL service. For more detailed information on the SSL module functions, see the “[Processing of SSL Flows by the SSL Module](#)” section in [Chapter 8, Examples of CSS SSL Configurations](#).

The SSL module provides the following major SSL features:

- [SSL Termination](#)
- [Client Authentication](#)
- [Back-End SSL](#)
- [SSL Initiation](#)

SSL Termination

When you create an entry in a proxy list to define the flow between an SSL module and a client, the module operates as a virtual SSL server by adding security services between a web browser (the client) and the HTTP connection (the server). All inbound SSL flows from a client terminate at an SSL module in the CSS.

Once the connection is terminated, the SSL module decrypts the data and sends the data as clear text to the CSS for a decision on load balancing. The CSS transmits the data as clear text to an HTTP server. For more information about SSL termination in the CSS, see [Chapter 4, Configuring SSL Termination](#).

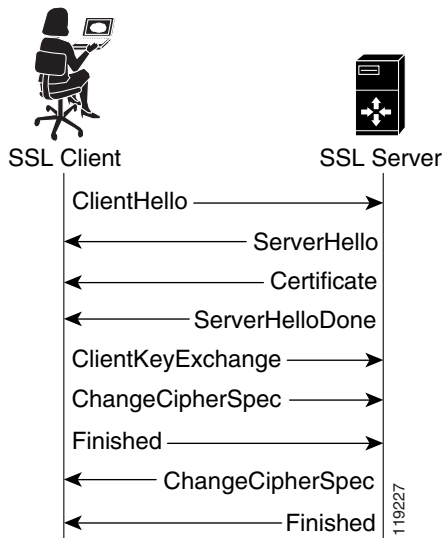
Client Authentication

When client authentication occurs on the CSS, the CSS verifies that the:

- Client sending the certificate has a corresponding private key
- Client certificate is signed by a known CA
- Certificate has not expired
- Signature is valid
- Issuing CA has not revoked the certificate if a Certificate Revocation List (CRL) is configured on the CSS

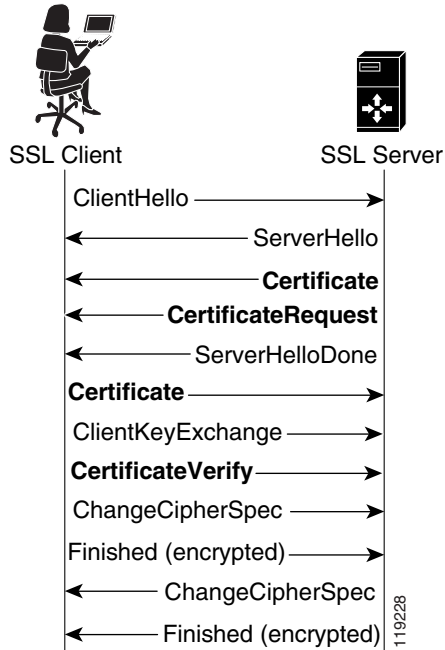
During a typical SSL handshake between a client and a server, the client does not send a certificate as shown in [Figure 1-1](#).

Figure 1-1 SSL Handshake Without Client Authentication



For a client to send a certificate, the server must include a certificate request (CertificateRequest) message in the handshake as shown in [Figure 1-2](#). The request message includes which types of certificates the server accepts. However, this message does not identify certificate authorities.

Figure 1-2 SSL Handshake With Client Authentication



After the server sends the ServerHelloDone message, the client responds with its certificate (Certificate) and key exchange. Then the client sends a CertificateVerify message that contains a digest of all the handshake messages from the server and was signed using the client public key. The server decrypts the message using the client public key ensuring that the client possesses the correct private key.

The CertificateVerify message does not check the authenticity of the certificate. However, it does check that the public portion of the client private key matches what is embedded in the certificate. This ensures that the client possesses the keypair that used to generate the certificate, and is not passing someone else's certificate. However, the CSS can check whether the issuer signature is authentic.

An X.509 certificate includes a signature that is generated by signing a message digest of the entire certificate object using the private key of the CA. A CA certificate contains the CA public key that verifies the digital signature of the client certificate. If the server has a CA certificate and thus the public key of the CA, it can verify that the client certificate was signed by the CA. The CSS allows you to configure up to four CA certificates per virtual SSL server.

When a CA revokes a client certificate, the CA adds the certificate to a published list called the Certificate Revocation List (CRL). The CA publicizes this list and updates it periodically. Clients and servers can access this list through HTTP to validate a certificate. The CSS allows you to configure a CRL record that defines how and when to retrieve a CRL onto the CSS. After the CSS retrieves the CRL, the virtual SSL server can use the downloaded CRL to check the validity of all client certificates.

For information on configuring client authentication on a CSS virtual SSL server, including enabling client authentication, verifying CA certificate authenticity, configuring a CRL record, and assigning it to a virtual SSL server, see [Chapter 4, Configuring SSL Termination](#).

Back-End SSL

A back-end SSL server entry in an SSL proxy list defines the flow from the SSL module to the back-end SSL server. After receiving encrypted data from a client, the SSL module, acting as a virtual client by preserving the originating client's IP address, encrypts the clear text data used for load balancing the flow and initiates the SSL connection to the back-end server.

On the outbound flow from the CSS, the SSL module responds in the reverse direction and sends the encrypted data from the server back to the client. For more information about back-end SSL in the CSS, see [Chapter 5, Configuring Back-End SSL](#).

SSL Initiation

SSL initiation enables the CSS to receive clear text from a client and then to originate an SSL session with an SSL server and join the client connection with the SSL back-end connection. The SSL server can either be a CSS configured for SSL termination (virtual SSL server) or a real back-end SSL server (Web server).

On the outbound flow, the CSS decrypts the SSL data from the server and sends clear text back to the client. For more information about SSL initiation in the CSS, see [Chapter 6, Configuring SSL Initiation](#).

For more detailed information on the SSL module functions, see the “[Processing of SSL Flows by the SSL Module](#)” section in [Chapter 8, Examples of CSS SSL Configurations](#).



SSL Configuration Quick Starts

This chapter provides a quick overview on how to manage SSL certificates in the CSS, create an SSL proxy list for virtual and back-end SSL servers, and add an SSL proxy list to an SSL service. Each step includes the CLI command required to complete the task. RSA has been chosen for the quick start procedures in this section because it is a popular public-key algorithm for encryption and authentication.

To configure SSL termination on a CSS, perform the steps in the following quick start procedures:

1. [RSA Certificate and Key Generation Quick Start, Table 2-1](#)
2. [RSA Certificate and Key Import Quick Start, Table 2-2](#)
3. [SSL Termination Proxy List Quick Start, Table 2-3](#)
4. [SSL Termination Service and Content Rule Quick Start, Table 2-6](#)

If your configuration includes back-end SSL, also perform the following quick start procedures:

1. [Back-End SSL Proxy List Quick Start, Table 2-4](#)
2. [Back-End SSL Service and Content Rule Quick Start, Table 2-7](#)

To configure SSL initiation, perform the following quick start procedures:

1. [SSL Initiation Proxy List Quick Start, Table 2-5](#)
2. [SSL Initiation Service Quick Start, Table 2-8](#)
3. [SSL Initiation Content Rule Quick Start, Table 2-9](#)

RSA Certificate and Key Generation Quick Start

Table 2-1 provides an overview of the steps required to generate and associate an RSA key pair and certificate in the CSS. Key and certificate generation may be necessary in instances where you do not have preexisting keys or certificates for the CSS. You may want to initially generate RSA keys and temporary certificates on the CSS for internal SSL testing. A generated certificate is temporary and expires in 30 days.

Table 2-1 RSA Certificate and Key Generation Quick Start

Task and Command Example

1. Enter global configuration mode.

```
# config
(config) #
```

2. Generate the RSA key pair used in the exchange.

```
(config) # ssl genrsa CSSrsakey1 1024 "passwd123"
Please be patient this could take a few minutes
```

3. Associate the generated RSA key pair with a file.

```
(config) # ssl associate rsakey myrsakey1 CSSrsakey1
```

Table 2-1 RSA Certificate and Key Generation Quick Start (continued)**Task and Command Example**

4. After generating the RSA key pair, generate the Certificate Signing Request (CSR) file for the RSA key pair file. For example, enter:

```
(config) # ssl gencsr myrsakey1
You are about to be asked to enter information
that will be incorporated into your certificate
request. What you are about to enter is what is
called a Distinguished Name or a DN.
For some fields there will be a default value,
If you enter '.', the field will be left blank.
Country Name (2 letter code) [US]US
State or Province (full name) [SomeState]New York
Locality Name (city) [SomeCity]Albany
Organization Name (company name) [Acme Inc]Cisco Systems, Inc.
Organizational Unit Name (section) [Web Administration]Web Admin
Common Name (your domain name) [www.acme.com]www.cisco.com
Email address [webadmin@acme.com]webadmin@cisco.com

-----BEGIN CERTIFICATE REQUEST-----
MIIBWDCCAQICAQAwgZwxCzAJBgNVBAYTA1VTMQswCQYDVQQIEwJNQTFETMBEGA1UE
BxMKQm94Ym9yY3VnaDEcMBoGA1UEChMTQ21zY28uG31zdGVtcywgSW5jLjESMBAG
A1UECxMJV2ViIEFkbWluMRYwFAYDVQQDEw13d3cuY21zY28uY29tMSEwHwYJKoZI
hvcNAQkBFhJra3JvZWJlckBjaXNjby5jb20wXDANBgkqhkiG9w0BAQEFAANLADBI
AkeEAqHxjtQUVXvmo6tAWPiMpe6oYhZbJUDgTxbW4VMCYgzGzn2wUJTgLfDB6N3
v+1tKFndE686BhKqfyOidml3wQIDAQABoAAwDQYJKoZIhvcNAQEEBQADQQA94yC3
4SUJJ4UQEnO2OqRGLOZpAE1c4+IV9aTWK6NmiZsM9Gt0vPhIkLx5jjhVRLlB27Ak
H6D5omXa0SPJan5x
-----END CERTIFICATE REQUEST-----

CSS11503 (config) #
```

The **ssl gencsr** command generates the CSR in PKCS10 encoded in Privacy Enhanced Mail (PEM) format and outputs it to the screen. Note that the CSR is not saved in the CSS.

5. Transfer the certificate request to the Certificate Authority (CA). Most major Certificate Authorities have Web-based applications that require you to cut and paste the certificate request to the screen.

If you require a global site certificate that allows 128-bit encryption for export-restricted browsers, apply for a StepUp/SGC or chained certificate from the CA.

You will receive your certificate in one to seven days.

Table 2-1 RSA Certificate and Key Generation Quick Start (continued)

Task and Command Example
<p>6. (Optional) While you are waiting to receive your signed certificate, you can test your CSR file by creating a temporary certificate by generating a CSR and signing it with your own private key. While this produces a valid certificate, most browsers flag the certificate as signed by an unrecognized signing authority. To generate a temporary certificate, see the “Generating a Self-Signed Certificate” section.</p>
<p>7. After you receive your certificate in one to seven days, save it as a file onto a secure FTP server.</p> <ul style="list-style-type: none">• If you received a server certificate, go to Step 11.• If you received a global site certificate, you must create a chained certificate. Go to Step 8.
<p>8. Obtain the intermediate certificate for the global site certificate from the following link: http://www.verisign.com/support/install/intermediate.htm. Save the certificate as a file on the secure FTP server.</p>
<p>9. Create a file, and copy the global site certificate and the intermediate certificate into it. The global site certificate must be first, followed by the intermediate certificate. Make sure that there is a single new line between the server and intermediate certificates.</p>
<p>10. Save the file.</p>
<p>11. Import the certificate into the CSS using the steps in the “RSA Certificate and Key Import Quick Start” section.</p>

RSA Certificate and Key Import Quick Start

Table 2-2 provides an overview of the steps required to import and associate an RSA certificate and key pair to the CSS from a remote server.

Table 2-2 RSA Certificate and Key Import Quick Start

Task and Command Example

1. Define a secure File Transfer Protocol (FTP) record file to import certificates and private keys into the CSS from an SFTP server.

```
# ftp-record ssl_record 192.168.19.21 johndoe "abc123"
/home/johndoe
```

2. Use secure FTP to transfer the imported certificates and private keys to the CSS.

```
# copy ssl sftp ssl_record import rsacert.pem PEM "passwd123"
Connecting
Completed successfully
```

```
# copy ssl sftp ssl_record import rsakey.pem PEM "passwd123"
Connecting
Completed successfully
```

3. Enter configuration mode.

```
# config
(config) #
```

4. To use RSA public key exchange and authentication:

- a. Associate the imported RSA certificate with a file.

```
(config) # ssl associate cert myrsacert1 rsacert.pem
```

- b. Associate the imported RSA key pair with a file.

```
(config) # ssl associate rsakey myrsakey1 rsakey.pem
```

5. Compare the public key in the associated certificate with the public key stored with the associated private key and verify that they are identical.

```
(config) # ssl verify myrsacert1 myrsakey1
Certificate mycert1 matches key mykey1
```

The following running-configuration example shows the results of entering the commands in [Table 2-2](#).

```
!***** GLOBAL *****
  ftp-record ssl-record 192.168.19.21 johndoe des-password
  1frapbyg4f1dce4d /home/johndoe

  ssl associate cert myrsacert1 rsacert.pem
  ssl associate rsakey myrsakey1 rsakey.pem
```

SSL Proxy List Quick Start

An SSL proxy list determines the flow of data to and from an SSL module. The following sections describe how to create a proxy list for:

- SSL termination
- Back-end SSL
- SSL initiation

SSL Termination Proxy List Quick Start

You must define a virtual SSL server in an SSL proxy list for an SSL module to properly process and terminate SSL communications from the client and initiate an HTTP connection to the server.

[Table 2-3](#) provides an overview of the steps required to create a virtual SSL server entry in an SSL proxy list for an RSA certificate and key pair. For information on configuring client authentication, see [“Configuring Client Authentication”](#) in [Chapter 4, Configuring SSL Termination](#).

Table 2-3 SSL Termination Proxy List Quick Start**Task and Command Example**

1. Create the SSL proxy list.

```
(config)# ssl-proxy-list ssl_list1  
Create ssl-list <ssl_list1>, [y/n]: y
```

Once you create an SSL proxy list, the CLI enters into ssl-proxy-list configuration mode for the newly created SSL proxy list.

```
(config-ssl-proxy-list[ssl_list1])#
```

2. Specify a number to identify a virtual SSL server in the SSL proxy list.

```
(config-ssl-proxy-list[ssl_list1])# ssl-server 20
```

3. Specify a virtual IP (VIP) address. Enter a VIP address that corresponds to an SSL content rule.

```
(config-ssl-proxy-list[ssl_list1])# ssl-server 20 vip address  
192.168.3.6
```

4. (Optional) Specify the virtual TCP port number if you need to change it to correspond with the content rule. By default, the virtual TCP port number is 443.

```
(config-ssl-proxy-list[ssl_list1])# ssl-server 20 port 444
```

5. Specify the name of an existing RSA certificate association and RSA key pair association for the SSL proxy list virtual SSL server.

```
(config-ssl-proxy-list[ssl_list1])# ssl-server 20 rsacert  
mysacert1  
(config-ssl-proxy-list[ssl_list1])# ssl-server 20 rsakey  
mysakey1
```

6. Assign the appropriate cipher suite for the RSA certificates and keys in use, the IP address of the back-end content rule used for the cipher suite, and the TCP port of the back-end content rule.

```
(config-ssl-proxy-list[ssl_list1])# ssl-server 20 cipher  
rsa-export-with-rc4-40-md5 192.168.3.6 8080 weight 5
```

7. (Optional) Specify the URL rewrite option for the domain name of the URL to be redirected to avoid nonsecure HTTP 300-series redirects.

```
(config-ssl-proxy-list[ssl_list1])# ssl-server 20 urlrewrite 22  
www.mydomain.com
```

Table 2-3 SSL Termination Proxy List Quick Start (continued)**Task and Command Example**

8. Continue to [Table 2-4](#) if the flow requires encryption to a back-end SSL server. If not, continue to step 9.
9. Activate the completed SSL proxy list.

```
(config-ssl-proxy-list[ssl_list1])# active
```

The following running-configuration example shows the results of entering the commands in [Table 2-3](#).

```
!***** SSL PROXY LIST *****
ssl-proxy-list ssl_list1
  ssl-server 20
  ssl-server 20 vip address 192.168.3.6
  ssl-server 20 port 444
  ssl-server 20 rsacert myrsacert1
  ssl-server 20 rsakey myrsakey1
  ssl-server 20 cipher rsa-export-with-rc4-40-md5 192.168.3.6 8080
weight 5
  ssl-server 20 urlrewrite 22 www.mydomain.com
active
```

Back-End SSL Proxy List Quick Start

If you require that a CSS send encrypted data to an SSL server, configure a back-end server entry in the SSL proxy list to allow the SSL module to encrypt the data and initiate an SSL connection to the server. You must configure back-end SSL with SSL termination. For the SSL termination quick start procedure, see the [“SSL Termination Proxy List Quick Start”](#) section.

Table 2-4 provides an overview of steps required to create a back-end SSL proxy list.

Table 2-4 Back-End SSL Proxy List Quick Start

Task and Command Example

1. Specify a number to identify a back-end SSL server in an existing SSL termination proxy list.

```
(config-ssl-proxy-list[ssl_list1])# backend-server 1
```

2. Specify an IP address. Enter an IP address that corresponds to the address of the service for the back-end SSL server.

```
(config-ssl-proxy-list[ssl_list1])# backend-server 1 ip address 192.168.4.4
```

3. (Optional) By default, the virtual TCP port number for the back-end server is 80. Assign the virtual TCP port number if you need to change it.

```
(config-ssl-proxy-list[ssl_list1])# backend-server 1 port 8080
```

4. Specify server IP address for the back-end server. Enter a valid IP address for the server.

```
(config-ssl-proxy-list[ssl_list1])# backend-server 1 server-ip 192.168.4.4
```

5. (Optional) By default, the server port number for the back-end server is 443. Assign the server port number if you need to change it.

```
(config-ssl-proxy-list[ssl_list1])# backend-server 1 server-port 113
```

Note If you configure the **backend-server number ip address** and **server-ip** commands with the same address, configure the **backend-server number port** and **server-port** commands with different port numbers.

6. (Optional) By default, the back-end server supports all available CSS cipher suites. If necessary, assign a specific cipher suite to be used by the back-end SSL server, for example the RSA certificates and keys:

```
(config-ssl-proxy-list[ssl_list1])# backend-server 1 cipher rsa-export-with-rc4-40-md5
```

7. Activate the completed SSL proxy list.

```
(config-ssl-proxy-list[ssl_list1])# active
```

The following running-configuration example shows the results of entering the commands in [Table 2-4](#) in bold and the commands associated with the virtual SSL server in [Table 2-3](#).

```
!***** SSL PROXY LIST *****
ssl-proxy-list ssl_list1
  ssl-server 20
  ssl-server 20 vip address 192.168.3.6
  ssl-server 20 port 444
  ssl-server 20 rsacert myrsacert1
  ssl-server 20 rsakey myrsakey1
  ssl-server 20 cipher rsa-export-with-rc4-40-md5 192.168.3.6 8080
weight 5
  ssl-server 20 urlrewrite 22 www.mydomain.com
  active

backend-server 1
backend-server 1 ip address 192.168.4.4
backend-server 1 port 8080
backend-server 1 server-ip 192.168.4.4
backend-server 1 server-port 113
backend-server 1 cipher rsa-export-with-rc4-40-md5
active
```

SSL Initiation Proxy List Quick Start

When you require that a CSS receive clear text from a client and then send encrypted data to an SSL server, configure an SSL initiation back-end server entry in the SSL proxy list to allow the SSL module to encrypt the data and initiate an SSL connection with the server.

[Table 2-5](#) provides an overview of steps required to create an SSL initiation proxy list.

Table 2-5 SSL Initiation Proxy List Quick Start**Task and Command Example**

1. Create the SSL proxy list.

```
(config)# ssl-proxy-list ssl_list1
Create ssl-list <ssl_list1>, [y/n]: y
```

Once you create an SSL proxy list, the CLI enters ssl-proxy-list configuration mode for the newly created SSL proxy list.

```
(config-ssl-proxy-list[ssl_list1])#
```

2. Specify a number to identify a back-end SSL server in an existing SSL termination proxy list.

```
(config-ssl-proxy-list[ssl_list1])# backend-server 1
```

3. Define the back-end server as an SSL initiation server.

```
(config-ssl-proxy-list[ssl_list1])# backend-server 1 type
initiation
```

4. Specify an IP address. Enter an IP address that corresponds to the IP address of the service for the back-end SSL server.

```
(config-ssl-proxy-list[ssl_list1])# backend-server 1 ip address
192.168.2.3
```

5. (Optional) By default, the virtual TCP port number for the back-end server is 80. Assign the virtual TCP port number if you need to change it.

```
(config-ssl-proxy-list[ssl_list1])# backend-server 1 port 8080
```

6. Specify a valid IP address for the back-end server.

```
(config-ssl-proxy-list[ssl_list1])# backend-server 1 server-ip
192.168.2.3
```

7. (Optional) By default, the server port number for the back-end server is 443. Assign the server port number if you need to change it.

```
(config-ssl-proxy-list[ssl_list1])# backend-server 1 server-port
40443
```

Note If you configure the **backend-server number ip address** and **server-ip** commands with the same address, configure the **backend-server number port** and **server-port** commands with different port numbers.

Table 2-5 SSL Initiation Proxy List Quick Start (continued)

Task and Command Example
<p>8. (Optional) By default, the back-end server supports all available CSS cipher suites. If necessary, assign a specific cipher suite to be used by the back-end SSL server.</p> <pre data-bbox="400 402 1157 451">(config-ssl-proxy-list[ssl_list1])# backend-server 1 cipher rsa-with-rc4-128-md5 weight 10</pre>
<p>9. (If Required) Configure client certificates and keys in the proxy list for SSL servers that request them. The certificates and keys must have already been imported and associated with a filename on the CSS. For example, to configure an existing RSA client certificate and key, enter:</p> <pre data-bbox="400 607 1170 711">(config-ssl-proxy-list[ssl_list1])# backend-server 1 rsacert myrsacert (config-ssl-proxy-list[ssl_list1])# backend-server 1 rsakey myrsakey</pre>
<p>10. (Optional) Configure CA certificates in the proxy list for server authentication by the SSL module (the client). The CA certificate must already have been imported and associated with a filename on the CSS.</p> <pre data-bbox="400 834 1157 883">(config-ssl-proxy-list[ssl_list1])# backend-server 1 cacert mycert1</pre>
<p>11. Activate the completed SSL proxy list.</p> <pre data-bbox="400 943 938 971">(config-ssl-proxy-list[ssl_list1])# active</pre>

The following running-configuration example shows the results of entering the commands in [Table 2-5](#).

```
!***** SSL PROXY LIST *****
ssl-proxy-list ssl-list1
  backend-server 1
  backend-server 1 initiation
  backend-server 1 ip address 192.168.2.3
  backend-server 1 port 8080
  backend-server 1 server-ip 192.168.2.3
  backend-server 1 server-port 40443
  backend-server 1 cipher rsa-with-rc4-128-md5 weight 10
  backend-server 1 rsacert myrsacert
  backend-server 1 rsakey myrsakey
  backend-server 1 cacert mycert1
  active
```

SSL Service and Content Rule Quick Start

Before the CSS can use an SSL proxy list, you must add the proxy to an SSL service and add the service to an SSL content rule. The following sections describe how to:

- Create an SSL service
- Create an SSL content rule
- Add the SSL service to the SSL content rule

SSL Termination Service and Content Rule Quick Start

[Table 2-6](#) provides an overview of the steps required to create an SSL service for SSL termination, including adding the SSL proxy list to the service and creating an SSL content rule.

Table 2-6 SSL Server Service and Content Rule Quick Start

Task and Command Example
<p>1. Create an SSL service.</p> <pre>(config)# service ssl_serv1 Create service <ssl_serv1>, [y/n]: y</pre>
<p>2. Specify ssl-accel as the service type.</p> <pre>(config-service[ssl_serv1])# type ssl-accel</pre>
<p>3. Specify the slot of the SSL module in the CSS chassis.</p> <pre>(config-service[ssl_serv1])# slot 3</pre>
<p>4. Disable the CSS from sending keepalive messages to the service.</p> <pre>(config-service[ssl_serv1])# keepalive type none</pre>
<p>5. Add the SSL proxy list to the SSL service.</p> <pre>(config-service[ssl_serv1])# add ssl-proxy-list ssl_list1</pre>
<p>6. Activate the SSL service.</p> <pre>(config-service[ssl_serv1])# active</pre>

Table 2-6 SSL Server Service and Content Rule Quick Start (continued)

Task and Command Example
<p>7. Create an SSL content rule.</p> <pre>(config)# owner ssl_owner Create owner <ssl_owner>, [y/n]: y (config-owner[ssl_owner])# content ssl_rule1 Create content <ssl_rule1>, [y/n]: y</pre>
<p>8. Configure a VIP address or domain name for the content rule. Ensure that the VIP address is the same as the address specified in the SSL proxy list.</p> <pre>(config-owner-content[ssl-rule1])# vip address 192.168.3.6</pre>
<p>9. Specify a TCP port number for the content rule. Ensure the port number is the same as the port specified in the SSL proxy list.</p> <pre>(config-owner-content[ssl-rule1])# port 444</pre>
<p>10. If you are using two or more SSL modules and want to use stickiness based on SSL version 3 session ID for a Layer 5 content rule, specify the following parameters in the content rule to take advantage of the SSL session ID reuse:</p> <ul style="list-style-type: none"> • Enter the application ssl command to specify the SSL application type. <pre>(config-owner-content[ssl-rule1])# application ssl</pre> • Enter the advanced-balance ssl command to enable the content rule to be sticky based on SSL. <pre>(config-owner-content[ssl-rule1])# advanced-balance ssl</pre>
<p>11. Add the SSL service to the content rule.</p> <pre>(config-owner-content[ssl_rule1])# add service ssl_serv1</pre>
<p>12. Activate the content rule.</p> <pre>(config-owner-content[ssl_rule1])# active</pre>
<p>13. Save your configuration changes to the running configuration.</p> <pre># copy running-config startup-config</pre>
<p>14. Continue to Table 2-7 if your configuration includes back-end SSL or Table 2-8 if your configuration includes SSL initiation.</p>

The following running-configuration example shows the results of entering the commands in [Table 2-6](#).

```
!***** SERVICE *****
service ssl-serv1
  type ssl-accel
  slot 3
  keepalive type none
  add ssl-proxy-list ssl_list1
  active

!***** OWNER *****
owner ssl_owner

content ssl_rule1
  protocol tcp
  vip address 192.168.3.6
  port 444
  application ssl
  advanced-balance ssl
  add service ssl-serv1
  active
```

Back-End SSL Service and Content Rule Quick Start

If you configured a back-end SSL server entry in an SSL proxy list, [Table 2-7](#) provides an overview of the steps required to create an SSL service for a back-end SSL server, including adding the SSL proxy list to the service and creating an SSL content rule.

Table 2-7 Back-End SSL Service and Content Rule Quick Start

Task and Command Example

1. Create an SSL service.

```
(config)# service ssl_serv2
Create service <ssl_serv2>, [y/n]: y
```

2. Specify **ssl-accel-backend** as the service type.

```
(config-service[ssl_serv2])# type ssl-accel-backend
```

3. Configure a virtual IP (VIP) address for the back-end server. The IP address must match the IP address configured for the back-end server.

```
(config-service[ssl_serv2])# vip address 192.168.4.4
```

Table 2-7 Back-End SSL Service and Content Rule Quick Start (continued)**Task and Command Example**

4. (Optional) Configure a virtual port number for the back-end server. The port number must match the virtual TCP port number configured for the back-end server. By default, the port number is 80. In this example, the port number is 8080.

```
(config-service[ssl_serv2])# port 8080
```

5. (Optional) By default, the service keepalive type is ICMP. You can also configure the keepalive type for a back-end service to be none, TCP, or SSL. If you configure a TCP or SSL keepalive type, you must configure the keepalive port correctly for the service to work.

For example, to configure a keepalive type of SSL, enter.

```
(config-service[ssl_serv2])# keepalive type ssl
```

Then configure the port for the back-end SSL server. For example, enter:

```
(config-service[ssl_serv2])# keepalive port 443
```

6. Add the SSL proxy list to the SSL service.

```
(config-service[ssl_serv2])# add ssl-proxy-list ssl_list1
```

7. Activate the SSL service.

```
(config-service[ssl_serv2])# active
```

8. Add the back-end server to an SSL content rule.

```
(config)# owner ssl_owner
(config-owner[ssl_owner])# content ssl_backend_rule1
Create content <ssl_backend_rule1>, [y/n]: y
```

9. Configure a virtual IP (VIP) address or domain name for the content rule. Ensure that the VIP address for the content rule is the same as the address specified for the virtual SSL server.

```
(config-owner-content[ssl_backend_rule1])# vip address 192.168.3.6
```

10. Specify a TCP port number for the content rule. Ensure the port number is the same as the virtual TCP port specified for the back-end SSL entry in the SSL proxy list.

```
(config-owner-content[ssl_backend_rule1])# port 8080
```

Table 2-7 Back-End SSL Service and Content Rule Quick Start (continued)**Task and Command Example**

11. Enter the **advanced-balance arrowpoint-cookie** command to enable the content rule to be sticky based on an arrowpoint cookie.

```
(config-owner-content[ssl_backend_rule1])# advanced-balance  
arrowpoint-cookie
```
12. (Optional) Enter the **url** command set to **/*** to use stickiness based on the cookie.

```
(config-owner-content[ssl_backend_rule1])# url "/*"
```
13. Add the SSL service to the content rule.

```
(config-owner-content[ssl_backend_rule1])# add service ssl_serv2
```
14. Activate the content rule.

```
(config-owner-content[ssl_backend_rule1])# active
```
15. Save your configuration changes to the running configuration.

```
# copy running-config startup-config
```

The following running-configuration example shows the results of entering the commands in [Table 2-7](#) in bold and the commands associated with the virtual SSL server in [Table 2-6](#).

```
!***** SERVICE *****
service ssl-serv1
  type ssl-accel
  slot 3
  keepalive type none
  add ssl-proxy-list ssl_list1
  active

service ssl_serv2
  type ssl-accel-backend
  ip address 192.168.4.4
  port 8080
  keepalive type ssl
  keepalive port 443
  add ssl-proxy-list ssl_list1
  active

!***** OWNER *****
owner ssl_owner

content ssl_backend_rule1
```

```

vip address 192.168.3.6
advanced-balance arrowpoint-cookie
protocol tcp
port 8080
url "/"
add service ssl_serv2
active

content ssl_rule1
protocol tcp
vip address 192.168.3.6
port 444
application ssl
advanced-balance ssl
add service ssl-serv1
active

```

SSL Initiation Service Quick Start

If you configured an SSL initiation server entry in an SSL proxy list, [Table 2-8](#) provides an overview of the steps required to create an SSL service for an SSL initiation server.

Table 2-8 SSL Initiation Service Quick Start

Task and Command Example

1. Create an SSL service.

```
(config)# service ssl_serv1
Create service <ssl_serv1>, [y/n]: y
```

2. Specify **ssl-init** as the service type.

```
(config-service[ssl_serv1])# type ssl-init
```

3. Configure the IP address for the service. The service IP address must be the same as the IP address specified in the SSL initiation proxy list using the **backend-server number ip address** command. See the “[SSL Initiation Proxy List Quick Start](#)” section.

```
(config-service[ssl_serv1])# ip address 192.168.2.3
```

4. Configure the service port. The service port must match the SSL initiation back-end server port.

```
(config-service[ssl_serv1])# port 8080
```

Table 2-8 SSL Initiation Service Quick Start (continued)**Task and Command Example**

5. By default, the service keepalive type is ICMP. For SSL initiation, the keepalive type can be ICMP, none, SSL, or TCP. If you specify either the SSL or TCP keepalive, you must configure the port that the keepalive uses. The keepalive port must match the SSL initiation back-end server port.

For example, to configure a keepalive type of SSL, enter:

```
(config-service[ssl_serv1])# keepalive type ssl
(config-service[ssl_serv1])# keepalive port 40443
```

6. Specify the slot in the CSS chassis where the SSL module designated for SSL initiation resides.

```
(config-service[ssl_serv1])# slot 5
```

7. Add the SSL proxy list to the SSL service.

```
(config-service[ssl_serv1])# add ssl-proxy-list ssl_list1
```

8. Activate the SSL service.

```
(config-service[ssl_serv1])# active
```

The following running-configuration example shows the results of entering the commands in [Table 2-8](#).

```
! ***** SERVICE *****
service ssl-serv2
  type ssl-init
  ip address 192.168.2.3
  port 8080
  slot 5
  keepalive type ssl
  keepalive port 40443
  add ssl-proxy-list ssl_list1
  active
```

SSL Initiation Content Rule Quick Start

If you configured an SSL initiation server entry in an SSL proxy list, [Table 2-9](#) provides an overview of the steps required to create an SSL content rule for an SSL initiation server.

Table 2-9 SSL Initiation Content Rule Quick Start

1. If necessary, create an owner.

```
(config)# owner ssl_owner
Create owner <ssl_owner>, [y/n]: y
```

2. Add the SSL initiation back-end server to an SSL content rule.

```
(config)# owner ssl_owner
(config-owner[ssl_owner])# content ssl_init_rule1
Create content <ssl_init_rule1>, [y/n]: y
```

3. Configure a virtual IP (VIP) address or domain name for the content rule.

```
(config-owner-content[ssl_backend_rule1])# vip address 192.168.2.3
```

4. Specify a TCP port number for the content rule.

```
(config-owner-content[ssl_backend_rule1])# port 80
```

5. (Optional) Enter the **url** command set to /* to use stickiness based on the cookie.

```
(config-owner-content[ssl_backend_rule1])# url "/*"
```

6. (Optional) Enter the **advanced-balance arrowpoint-cookie** command to enable the content rule to be sticky based on an arrowpoint cookie.

```
(config-owner-content[ssl_backend_rule1])# advanced-balance
arrowpoint-cookie
```

7. Add the SSL service to the content rule.

```
(config-owner-content[ssl_backend_rule1])# add service ssl_serv2
```

8. Activate the content rule.

```
(config-owner-content[ssl_backend_rule1])# active
```

9. Save your configuration changes to the running configuration.

```
# copy running-config startup-config
```

The following running-configuration example shows the results of entering the commands in [Table 2-9](#).

```
!***** OWNER *****
owner ssl_owner

content ssl_init_rule1
vip address 192.168.2.3
port 80
url "/"
advanced-balance arrowpoint-cookie
add service ssl_serv1
active
```




Configuring SSL Certificates and Keys

This chapter describes how to generate and import SSL certificates and keys, and how to associate them with files for use in the CSS. It contains the following major sections:

- [Overview of SSL Certificates and Keys](#)
- [Generating Certificates and Private Keys in the CSS](#)
- [Preparing a Global Site Certificate](#)
- [Importing or Exporting Certificates and Private Keys](#)
- [Associating Certificate and Private Key Files with Names](#)
- [Removing Certificates and Private Keys from the CSS](#)

Overview of SSL Certificates and Keys

Digital certificates and key pairs are a form of digital identification for user authentication. Certificate Authorities (CAs), such as VeriSign and Thawte, issue certificates. A client or server certificate includes the name of the issuing authority and digital signature, the serial number, the name of the client or server that the certificate was issued for, the public key, and the time stamps that indicate the certificate's expiration date.

A CA also provides a trusted CA certificate to verify that a client or server certificate originated from the CA. This certificate also can verify that a certificate revocation list (CRL) originated from the CA. This CA certificate includes the CA distinguished name, public key, and digital signature.

The CSS require certificates and keys for:

- SSL termination - You must obtain a server certificate and key.
- SSL initiation - You must obtain a client certificate and key.
- Client authentication - You must obtain a trusted CA certificate from the CA to verify that the client certificate and certificate revocation list (CRL) were issued by the CA.

Before configuring SSL termination, client authentication, or SSL initiation, you must load a digital certificate on the CSS disk (flash disk or hard disk). For SSL termination or SSL initiation, you must also load a public/private key pair on the CSS. The CSS stores digital certificates and key pairs in encrypted files in a secure area on the CSS.

For server and client certificates, you can use files received from a CA, import the certificate and keys from an existing secure server, or generate your own certificate and keys on the CSS. The CSS supports the generation of certificates and keys directly within the CSS for purposes of testing. Your requirement to use generated certificates and keys instead of certificates and keys from a trusted authority depends on your environment. For example, the use of the CSS and SSL for a company's internal website may not require the use of certificates from a trusted CA. A certificate and key pair generated within the CSS may be sufficient to satisfy the intranet SSL requirement.

After you import certificates or key pairs on the CSS, you must associate them to filenames. You will use these filenames when you configure SSL termination, client authentication, or SSL initiation.

**Caution**

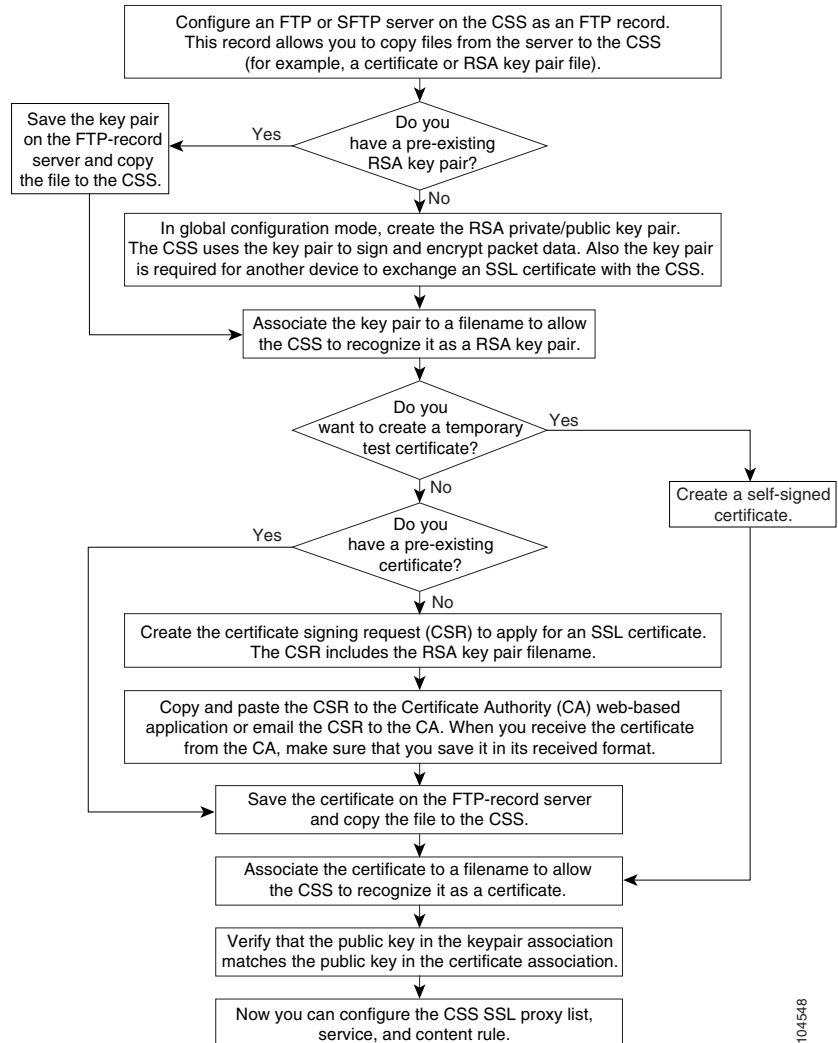
When importing or exporting certificates and keys with the CSS, ensure that the CSS is not configured to perform a network boot from a network-mounted file system on a remote system (operating the CSS in a diskless environment). The network-mounted method of CSS booting is not supported with SSL termination; the certificates and keys must be local to the CSS and SSL module.

**Note**

To implement good security policies when importing or generating SSL certificates and key pairs, administrators should understand the user modes of the CSS and have strong password policies to protect those user modes. For more information, refer to the *Cisco Content Services Switch Command Reference*, Chapter 2, CLI Commands, the “(config) username-technician” section.

Figure 3-1 provides an overview of how to configure an RSA key pair and SSL server certificate on the CSS.

Figure 3-1 SSL Key and Server Certificate Configuration Overview



Generating Certificates and Private Keys in the CSS

If you have preexisting certificates and private keys, you can import them to the CSS disk. For information on importing preexisting certificates and private keys, see the “[Importing or Exporting Certificates and Private Keys](#)” section.

If you do not have preexisting keys, Diffie-Hellman parameters, and certificates for the CSS, the CSS includes a series of certificate and private key management utilities to generate them. These utilities simplify the process of generating an RSA private key, a DSA private key, a Diffie-Hellman parameter file, a certificate signing request (CSR), and a self-signed temporary certificate.

**Note**

The `ssl genrsa`, `gencsr`, `gensda`, and `gencert` commands all produce a valid certificate or key pair. Be aware, however, that most Web browsers will flag the certificate as signed by an unrecognized signing authority. A generated certificate is temporary and expires in one year. The `ssl gencsr` command generates a certificate request in PKCS10 encoded in Privacy Enhanced Mail (PEM) format.

This section covers:

- [Generating an RSA Key Pair](#)
- [Generating a DSA Key Pair](#)
- [Generating Diffie-Hellman Key Parameters](#)
- [Using an RSA Key to Generate a Certificate Signing Request](#)
- [Generating a Self-Signed Certificate](#)

Generating an RSA Key Pair

RSA key pairs are used to sign and encrypt packet data, and they are required before another device (client or server) can exchange an SSL certificate with the CSS. The key pair refers to a public key and its corresponding private (secret) key. The CSS stores the generated RSA key pair as a file on the CSS.

Use the **ssl genrsa** command to generate an RSA private/public key pair for asymmetric encryption. The syntax for this command is:

```
ssl genrsa filename numbits "password"
```

The variables are:

- *filename* - The name of generated RSA key pair file. Enter an unquoted text string with a maximum of 31 characters. The key pair filename is used only for identification in the CSS.
- *numbits* - The key pair strength. The number of bits in the key pair file defines the size of the RSA key pair used to secure Web transactions. Longer keys produce a more secure implementation by increasing the strength of the RSA security policy. Available entries (in bits) are 512 (least security), 768 (normal security), 1024 (high security), and 2048 (highest security).
- *"password"* - The password used to encode the RSA private key using DES (Data Encryption Standard) before it is stored as a file on the CSS. Encoding the file prevents unauthorized access to the imported certificate and private key on the CSS. Enter the password as a quoted string with a maximum of 35 characters. The password appears in the CSS running configuration as a DES-encoded string.

For example, to generate the RSA key pair *myrsafile1*, enter:

```
(config) # ssl genrsa myrsafile1 1024 "passwd123"  
Please be patient this could take a few minutes
```

After you generate an RSA key pair, you can generate a Certificate Signing Request (CSR) file for the RSA key pair file and transfer the certificate request to the Certificate Authority (CA). This provides an added layer of security because the RSA private key originates directly within the CSS and does not have to be transported externally. You can then create a temporary certificate for internal testing until the CA responds to the certificate request and returns the authentic certificate. Each generated key pair must be accompanied by a certificate to work.

You must also associate an RSA key pair name with the generated RSA key pair, as discussed in the [“Associating Certificate and Private Key Files with Names”](#) section of this chapter.

Generating a DSA Key Pair

DSA is the public key exchange cryptographic system developed by the National Institutes of Science and Technology. DSA can only be used for digital signatures (signings); it cannot be used for key private/public exchange. The CSS stores the generated DSA key pair as a file on the CSS.

Use the **ssl gendsa** command to generate a DSA private/public key pair for asymmetric encryption. The syntax for this command is:

```
ssl gendsa filename numbits “password”
```

The variables are:

- *filename* - The name of the generated DSA key pair file. Enter an unquoted text string with a maximum of 31 characters. The key pair filename is used only for identification in the CSS.
- *numbits* - The key pair strength. The number of bits in the key pair file defines the size of the DSA key pair used to secure Web transactions. Longer keys produce a more secure implementation by increasing the strength of the DSA security policy. Available entries (in bits) are 512 (least security), 768 (normal security), and 1024 (highest security).
- *“password”* - The password used to encode the DSA private key using DES (Data Encryption Standard) before it is stored as a file on the CSS. Encoding the file prevents unauthorized access to the imported certificate and private key on the CSS. Enter the password as a quoted string with a maximum of 35 characters. The password appears in the CSS running configuration as a DES-encoded string.

For example, to generate the DSA key pair *mysakeyfile2*, enter:

```
(config) # ssl gendsa mysakeyfile2 512 “passwd123”  
Please be patient this could take a few minutes
```

You must also associate a DSA key pair name with the generated DSA key pair as discussed in the [“Associating Certificate and Private Key Files with Names”](#) section of this chapter.

Generating Diffie-Hellman Key Parameters

Diffie-Hellman is a shared key agreement algorithm. Diffie-Hellman key exchange uses a complex algorithm and public/private keys to encrypt and then decrypt packet data. The CSS stores the generated Diffie-Hellman key parameter file. Use the **ssl gendh** command to generate a Diffie-Hellman key agreement parameter file.



Note

Generation of a Diffie-Hellman key agreement parameter file can sometimes take a lengthy period of time (perhaps up to 20 minutes) and is a CPU-intensive utility. If you are running the **ssl gendh** utility, ensure that the CSS is not actively passing traffic at the same time to avoid impacting CSS performance.

The syntax for this command is:

```
ssl gendh filename numbits "password"
```

The variables are:

- *filename* - The name of the file to store the Diffie-Hellman key parameters. Enter an unquoted text string with a maximum of 31 characters. The filename is used only for identification in the CSS.
- *numbits* - The key strength. The number of bits in the file defines the size of the Diffie-Hellman key used to secure Web transactions. Longer keys produce a more secure implementation by increasing the strength of the Diffie-Hellman security policy. Available entries (in bits) are 512 (least security), 768 (normal security), 1024 (high security), and 2048 (highest security).
- "*password*" - The password used to encode the Diffie-Hellman key using DES (Data Encryption Standard) before it is stored as a file on the CSS. Encoding the file prevents unauthorized access to the imported certificate and private key on the CSS. Enter the password as a quoted string with a maximum of 35 characters. The password appears in the CSS running configuration as a DES-encoded string.

For example, to generate the Diffie-Hellman key parameter list *dhparamfile2*, enter:

```
(config) # ssl gendh dhparamfile2 512 "passwd123"  
Please be patient this could take a few minutes
```

You must also associate a Diffie-Hellman parameter filename with the generated Diffie-Hellman parameter file, as discussed in the “[Associating Certificate and Private Key Files with Names](#)” section of this chapter.

Using an RSA Key to Generate a Certificate Signing Request

To generate a Certificate Signing Request (CSR) file for an RSA key pair file and to transfer the certificate request to the Certificate Authority (CA), use the **ssl gencsr *rsa*key** command. This command generates a CSR in PKCS10 encoded in PEM format.

You must generate a CSR file if you are requesting a new certificate or renewing a certificate. When the CA signs the CSR using its RSA private key, the CSR becomes the certificate.

The *rsa*key variable specifies the key on which the RSA certificate is built. It is the public key that is embedded in the certificate.

To use the RSA key pair to generate a CSR, ensure the RSA key pair file is loaded on the CSS. Associate an RSA key pair name to the generated RSA keypair (see the “[Associating Certificate and Private Key Files with Names](#)” section). If the appropriate key pair does not exist, the CSS logs an error message.

For example, to generate a CSR based on the RSA key pair *myrsa*key1, enter:

```
CSS11503(config)# ssl gencsr myrsakey1
You are about to be asked to enter information
that will be incorporated into your certificate
request. What you are about to enter is what is
called a Distinguished Name or a DN.
For some fields there will be a default value,
If you enter '.', the field will be left blank.
Country Name (2 letter code) [US]US
State or Province (full name) [SomeState]New York
Locality Name (city) [SomeCity]Albany
Organization Name (company name) [Acme Inc]Cisco Systems, Inc.
Organizational Unit Name (section) [Web Administration]Web Admin
Common Name (your domain name) [www.acme.com]www.cisco.com
Email address [webadmin@acme.com]webadmin@cisco.com

-----BEGIN CERTIFICATE REQUEST-----
MIIBWDCCAQICAQAwgZwx CzAJBgNVBAYTA1VTMQswCQYDVQQIEwJNQTEtBEGAlUE
BxMKQm94Ym9yb3VnaDEcMBoGAlUEChMTQ21zY28gU31zdGVtcywgSW5jLjESMBAG
AlUECxMjV2ViIEFkbWluMRYwFAYDVQQDEw13d3cuY21zY28uY29tMSEwHwYJKoZI
hvcNAQkBFhJra3JvZWJ1ckBjaXNjby5jb20wXDANBgkqhkiG9w0BAQEFAANLADBI
```

```
AkEAqHXjtQUVXvmo6tAWP1Mpe6oYhZbJUDgTxbW4VMCygzGZn2wUJTgLrifDB6N3
v+1tKFndE686BhKqfyOidm13wQIDAQABoAAwDQYJKoZIhvcNAQEEBQADQQA94yC3
4SUJJ4UQEnO2OqRGL0ZpAE1c4+IV9aTWK6NmiZsM9Gt0vPhIkLx5jjhVRL1b27Ak
H6D5omXa0SPJan5x
-----END CERTIFICATE REQUEST-----
```

```
CSS11503(config)#
```

The **ssl gencsr** command generates the CSR in PKCS10 encoded in PEM format and outputs it to the screen. Most major Certificate Authorities have web-based applications that require you to cut and paste the certificate request to the screen. If necessary, you can also cut and paste the certificate to a file. Note that the CSR is not saved in the CSS.

**Note**

If you require a global site certificate that allows 128-bit encryption for export-restricted browsers, apply for a StepUp/SGC or chained certificate from the Certificate Authority. After you receive the certificate, you must prepare it for use with the CSS. For more information, see the [“Preparing a Global Site Certificate”](#) section.

After submitting your CSR to the Certificate Authority (CA), you will receive your signed certificate between one to seven business days. When you receive your CSR, import the CSR to the CSS and then associate it. For information on importing the CSR, see the [“Importing or Exporting Certificates and Private Keys”](#) section. For information on associating it, see the [“Associating Certificate and Private Key Files with Names”](#) section.

While you are waiting to receive your signed certificate, you can test your CSR file by creating a temporary certificate by generating a CSR and signing it with your own private key. While this produces a valid certificate, most browsers flag the certificate as signed by an unrecognized signing authority. To generate a temporary certificate, see the [“Generating a Self-Signed Certificate”](#) section.

Generating a Self-Signed Certificate

For purposes of SSL testing, you can generate a temporary certificate by generating a CSR and signing it with your own private key. A generated certificate is temporary and expires in 30 days. Use the **ssl gencert** command to generate and save a temporary certificate to a file on disk in the CSS.



Note

The **ssl gencert** command produces a valid certificate. However, most Web browsers flag this certificate as signed by an unrecognized signing authority.

Before you generate the certificate, consider:

- The key pair that the certificate is based on (RSA or DSA).
- The key used to sign the certificate.

The **ssl gencert** command can sign RSA or DSA certificates with either an RSA key pair or a DSA key pair.



Note

Although the CSS allows signing an RSA certificate with a DSA key (and a DSA certificate with an RSA key) it is a more standard practice that an RSA certificate is signed with RSA keys (and DSA certificate is signed with a DSA key).

The syntax for this command is:

```
ssl gencert certkey certkey signkey signkey certfile "password"
```

The variables are:

- **certkey certkey** - The name of the RSA or DSA key pair on which the certificate is based. Enter an unquoted text string with a maximum of 31 characters.
- **signkey signkey** - The RSA or DSA key pair to be used to sign the certificate. Enter an unquoted text string with a maximum of 31 characters.
- *certfile* - The name of the file used to store the certificate as a file on the CSS. Enter an unquoted text string with a maximum of 31 characters.
- *"password"* - The password used to encode the certificate file using DES (Data Encryption Standard) before it is stored as a file on the CSS. Encoding the file prevents unauthorized access to the imported certificate and private

key on the CSS. Enter the password as a quoted string with a maximum of 35 characters. The password appears in the CSS running configuration as a DES-encoded string.

For example, to interactively generate the *mycertfile2* certificate, enter:

```
CSS11503(config)# ssl gencert certkey myrsakey signkey myrsasignkey  
myrsacertfile "passwd123"  
You are about to be asked to enter information  
that will be incorporated into your certificate  
request. What you are about to enter is what is  
called a Distinguished Name or a DN.  
For some fields there will be a default value,  
If you enter '.', the field will be left blank.  
Country Name (2 letter code) [US]US  
State or Province (full name) [SomeState]New York  
Locality Name (city) [SomeCity]Albany  
Organization Name (company name) [Acme Inc]Cisco Systems, Inc.  
Organizational Unit Name (section) [Web Administration]Web Admin  
Common Name (your domain name) [www.acme.com]www.cisco.com  
Email address [webadmin@acme.com]webadm@cisco.com  
  
CSS11503(config)#
```

You must also associate the contents of this temporary certificate to a filename, as discussed in the “[Associating Certificate and Private Key Files with Names](#)” section of this chapter.

Preparing a Global Site Certificate

Export browsers may use 40-bit encryption to initiate connections to SSL servers. With a conventional server certificate, a browser and server complete the SSL handshake and use a 40-bit key to encrypt application data.

A global site certificate is an extended server certificate that allows 128-bit encryption for export-restricted browsers. When the server responds to a browser with a global certificate, the client automatically renegotiates the connection to use 128-bit encryption.

If you applied for a global site certificate from the CA, you must obtain both the global certificate and its intermediate CA certificate. The intermediate CA certificate validates the global certificate. You can obtain a VeriSign Intermediate certificate from the following link:

<http://www.verisign.com/support/install/intermediate.html>

Then you must chain both certificates together in a single file, creating a chained certificate. As one file, the CSS returns the entire certificate chain to the client upon the initial SSL handshake.

Copy the server global and intermediate certificates to an FTP server. When creating a chained certificate for the CSS, make sure that the global and intermediate certificate are in their proper order. In one file, paste the server global site certificate first, followed by the intermediate certificate. You must insert a single new line between the certificates.

Save the file and import it to the CSS, as described in the [“Importing or Exporting Certificates and Private Keys”](#) section.

Importing or Exporting Certificates and Private Keys

You can import preexisting or new certificates and private keys to the CSS disk from a file, or a series of files, that are stored on a remote secure server. For information on generating certificates, see the [“Generating Certificates and Private Keys in the CSS”](#) section.

To transfer these files, Cisco Systems recommends that you use a secure encrypted transport mechanism between the CSS and the remote server. The CSS supports the Secure Shell protocol (SSHv2), which provides secure encryption communications between two hosts over an insecure network. The CSS supports file transport between network devices using the Secure File Transfer Protocol (SFTP) and the File Transfer Protocol (FTP). Of the two file transport protocols, Cisco Systems recommends SFTP as the transport mechanism of choice. It is similar to FTP except that it uses a secure and encrypted connection.

Before you import certificates or keys to the CSS:

- On the CSS, ensure that SSH access to the CSS is enabled to accept connections from SSH clients and that the Secure Management license key is installed prior to transferring certificates and keys. By default, SSH access is enabled through the **no restrict ssh** global command. If SSH access is restricted, or if the license key is not installed, SSH will not accept connections from SSH clients and the **copy ssl sftp** command will fail, resulting in generation of an error message.



Note For details about configuring Secure Shell Daemon on the CSS, refer to the *Cisco Content Services Switch Security Configuration Guide*.

- On the SFTP server, verify that the server is properly configured so that the user directory points to the directory where the certificates and keys reside. This path is required to ensure certificates and keys are properly copied from or to the SFTP server.



Caution

When using SSH, ensure that the CSS is not configured to perform a network boot from a network-mounted file system on a remote system (a diskless environment). If SSH is enabled and the CSS has been booted using a network boot from a network-mounted file system, the CSS logs an error message by SSH as the protocol attempts to initialize and then exits from operation, which impacts importing and exporting certificates and keys.

Configuring the Default SFTP or FTP Server to Import Certificates and Private Keys

Before you begin, use the **ftp-record** command to define the SFTP or FTP server that you intend to use to download imported certificates and private keys to the CSS disk. For details about using the **ftp-record** command to create an SFTP or FTP record file to use when accessing the server from the CSS, refer to the *Cisco Content Services Switch Administration Guide*.



Note

When defining the FTP record for the **copy ssl** command, ensure that the base directory, if used, is relative to the SSH directory where the SSH server resides. For example, if the username is *sshlogin* and the SSH server is installed in `d:\Program Files\Network`, the default directory for the files would be `d:\Program Files\Network\ssh`. This path is required to ensure certificates and keys are properly copied to or from the SFTP server.

For example, to define the *ssl_record*, enter:

```
# ftp-record ssl_record 192.168.19.21 johndoe "abc123" /home/johndoe
```

Transferring Certificates and Private Keys to the CSS

To facilitate the import or export of certificates and private keys from or to the CSS, use the **copy ssl** command. The CSS stores all imported files in a secure location on the CSS. This command is available only in SuperUser mode.

The syntax for this command is:

```
copy ssl [protocol] ftp_record [import filename [format] "password"  
  {"passphrase"} | export filename2 "password"
```

The variables are:

- *protocol* - The type of protocol used to transfer the certificate and private key file. The valid entries are **sftp** or **ftp**. Cisco Systems recommends the SFTP protocol for the transport mechanism because it provides the most security.
- *ftp_record* - The name of the previously-created FTP record containing the remote host information.
- **import** - Imports the file from the remote server.
- *filename* - The name of the file you want to import from the server. Include the full path to the file. You can enter a maximum of 128 characters.
- *format* - The file format of the certificate to be imported. Once the certificate file is converted to PEM format and DES encoded, it is stored on the CSS SCM in a special (and secure) directory. The valid import file formats are:
 - **DER** - Binary format encoding of the certificate file in ASN.1 using the Distinguished Encoding Rules (DER-encoded X509 certificate). For example, an imported certificate from a Microsoft Windows NT IIS 4.0 server.
 - **PEM** - Privacy Enhanced Mail, a base64 encoding of the certificate file (PEM-encoded X509 certificate). For example, an imported certificate from an Apache/SSL UNIX server.
 - **PKCS12** - Standard from RSA Data Security, Inc. for storing certificates and private keys. For example, an imported certificate from a Microsoft Windows 2000 IIS 5.0 server.

- “*password*” - The password used to DES (Data Encryption Standard) encode the imported certificate or private key. Encoding the imported file prevents unauthorized access to the certificate or private key on the CSS. Enter the password as a quoted string with a maximum of 35 characters. The password appears in the CSS running configuration as a DES-encoded string.
- “*passphrase*” - (Optional for PEM files) The passphrase used to encrypt the certificate or key being imported into the CSS. Enter the passphrase as a quoted text string.

**Note**

You must enter a passphrase for a PKCS12 file (.pfx). The CSS uses the passphrase to decrypt the file.

- **export** - Export the file to the remote server.
- *filename2* - The name you want to assign to the file on the server. Include the full path to the file. Enter an unquoted text string with no spaces and a maximum length of 32 characters.

**Note**

An imported file can contain certificates, RSA or DSA key pairs, or Diffie-Hellman parameters. You must distinguish whether the files contain certificates, private keys, or Diffie-Hellman parameters by associating the specific contents to a filename. See the [“Associating Certificate and Private Key Files with Names”](#) section.

For example, to import the *rsacert.pem* certificate from a remote server to the CSS, enter:

```
# copy ssl sftp ssl_record import rsacert.pem PEM "passwd123"  
Connecting  
Completed successfully
```

For example, to import the *rsakey.pem* certificate from a remote server to the CSS, enter:

```
# copy ssl sftp ssl_record import rsakey.pem PEM "passwd123"  
Connecting  
Completed successfully
```

To export the *rsacert.pem* certificate from the CSS to a remote server, enter:

```
# copy ssl sftp ssl_record export rsacert.pem "passwd123"
```

If the **copy ssl** command fails to import certificates or keys, verify the following areas:

- The user account and password in the ftp record are correct
- The base directory is ssh or ssh/path
- The SSH server is reachable
- The SSH server IP address is correct in the ftp-record

Associating Certificate and Private Key Files with Names

After you import or generate certificate and key pair files, you must indicate to the CSS whether these files contain certificates, private keys, or Diffie-Hellman parameters. You do this by associating certificate names, private/public key pair names, or Diffie-Hellman parameter names with the particular imported files.

When you associate the entries specified in the various certificate and private key commands with files, the CSS stores the bindings in the running configuration. Before you log out or reboot the CSS, you must copy the contents of the running-config file to the startup-config file to save the configuration changes and to enable the CSS to use this configuration on subsequent reboots. When you reboot the CSS, the certificate and key associations are loaded automatically.

This section covers:

- [Associating a Certificate with a File](#)
- [Associating an RSA Key Pair with a File](#)
- [Associating a DSA Key Pair with a File](#)
- [Associating Diffie-Hellman Parameters with a File](#)
- [Verifying a Certificate Against a Key Pair](#)

Associating a Certificate with a File

To associate a certificate name with an imported or generated certificate, use the **ssl associate cert** command. Use the **no** form of the command to remove the association with the file.

The syntax for this command is:

```
ssl associate cert certname filename
```

The variables are:

- *certname* - The name of the certificate association. Enter an unquoted text string with a maximum of 31 characters.
- *filename* - The name of the file containing the certificate. Enter a maximum of 128 characters. To see a list of imported or generated certificates, use the **ssl associate cert certname ?** command.

For example, to associate the certificate name *myrsacert1* with the imported certificate file *rsacert.pem*, enter:

```
(config) # ssl associate cert myrsacert1 rsacert.pem
```

To remove the association with the file, enter:

```
(config) # no ssl associate ssl cert myrsacert1
```



Note

The **no** form of the command does not function if the associated certificate is in use by an active SSL proxy list.

Associating an RSA Key Pair with a File

To associate an RSA key pair name with an imported or generated RSA key pair, use the **ssl associate rsakey** command. Use the **no** form of the command to remove the association with the file.

The syntax for this command is:

```
ssl associate rsakey keyname filename
```

The variables are:

- *keyname* - The name of the RSA key pair association. Enter an unquoted text string with a maximum of 31 characters.
- *filename* - The name of the file containing the RSA key pair. Enter a maximum of 128 characters. To see a list of imported or generated RSA keys, use the **ssl associate rsakey *keyname* ?** command.

For example, to associate the RSA key name *myrsakey1* with the imported *rsakey.pem*, enter:

```
(config) # ssl associate rsakey myrsakey1 rsakey.pem
```

To remove the association with the file, enter:

```
(config) # no ssl associate rsakey myrsakey1
```



Note

The **no** form of the command will not function if the associated RSA key pair is in use by an active SSL proxy list.

Associating a DSA Key Pair with a File

To associate a DSA key pair name with an imported or generated DSA key pair, use the **ssl associate dsakey** command. Use the **no** form of the command to remove the association with the file.

The syntax for this command is:

```
ssl associate dsakey keyname filename
```

The variables are:

- *keyname* - The name of the DSA key pair association. Enter an unquoted text string with a maximum of 31 characters.
- *filename* - The name of the file containing the DSA key pair. Enter a maximum of 128 characters. To see a list of imported or generated DSA keys, use the **ssl associate dsakey *keyname* ?** command.

For example, to associate the DSA key name *mydsakey1* with the imported *dsakey.pem*, enter:

```
(config) # ssl associate dsakey mydsakey1 dsakey.pem
```

To remove the association with the file, enter:

```
(config) # no ssl associate dsakey mydsakey1
```

**Note**

The **no** form of the command will not function if the associated DSA key pair is in use by an active SSL proxy list.

Associating Diffie-Hellman Parameters with a File

To associate a Diffie-Hellman name with an imported or generated Diffie-Hellman parameter file, use the **ssl associate dhparam** command. Use the **no** form of the command to remove the association to the file.

The syntax for this command is:

```
ssl associate dhparam paramname filename
```

The variables are:

- *paramname* - The name of the Diffie-Hellman parameter association. Enter an unquoted text string with a maximum of 31 characters.
- *filename* - The name of the file containing the Diffie-Hellman parameters. Enter a maximum of 128 characters. To see a list of imported or generated Diffie-Hellman files, use the **ssl associate dhparam filename ?** command.

For example, to associate the Diffie-Hellman filename *mydhparam1* with the imported *dhparams.pem*, enter:

```
(config) # ssl associate dhparam mydhparam1 dhparams.pem
```

To remove the association with the file, enter:

```
(config) # no ssl associate dhparam mydhparam1
```

**Note**

The **no** form of the command will not function if the associated Diffie-Hellman parameter list is in use by an active SSL proxy list.

Verifying a Certificate Against a Key Pair

A digital certificate is built around a public key and can only be used with one key pair. Use the **ssl verify** command to compare the public key in the associated certificate with the public key stored with the associated private key, and verify that they are identical. To see a list of certificate and key pair associations, use the **ssl verify ?** command.

**Note**

If the certificate does not match the public/private key pair, the CSS logs an error message.

The syntax for this command is:

```
ssl verify certname keyname
```

The variables are:

- *certname* - The association name of the certificate used to verify against the specified key pair.
- *keyname* - The association name of the key pair used to verify against the specified certificate.

For example, to verify the *myrsacert1* digital certificate against the *myrsakey1* key pair, enter:

```
(config)# ssl verify myrsacert1 myrsakey1  
Certificate and key match
```

Removing Certificates and Private Keys from the CSS

To remove certificates and private keys from the CSS that are no longer valid, use the **clear ssl file** command. Note that the **clear ssl file** command does not function if the file currently has an association with it. First remove the association to the file by specifying the **no ssl associate** command (see the [“Associating Certificate and Private Key Files with Names”](#) section).

The syntax for this global configuration mode command is:

```
clear ssl file filename password
```

The variables are:

- *filename* - The name of the certificate, key pair, or Diffie-Hellman parameter file that you want to remove from the CSS.
- *password* - The password used to encode the file using DES when it was originally imported or generated by the CSS. This password must be an exact match or the file cannot be cleared.

For example, to remove *dsacert.pem* from the CSS, enter:

```
# clear ssl file dsacert.pem "passwd123"
```

■ Associating Certificate and Private Key Files with Names



Configuring SSL Termination

This chapter describes the steps required to configure a CSS as a virtual SSL server for SSL termination. It contains the following major sections:

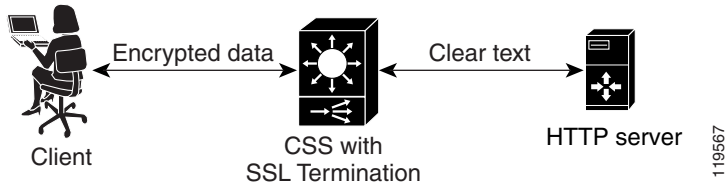
- [Overview of SSL Termination](#)
- [Creating an SSL Proxy List](#)
- [Adding a Description to an SSL Proxy List](#)
- [Configuring Virtual SSL Servers for an SSL Proxy List](#)
- [Activating and Suspending an SSL Proxy List](#)
- [Configuring a Service for SSL Termination](#)
- [Configuring a Content Rule for SSL Termination](#)

Overview of SSL Termination

SSL termination in a CSS occurs when an SSL module, acting as a proxy server, terminates an SSL connection from a client, and then establishes a TCP connection to a server. When the module terminates the SSL connection, it decrypts the data and sends the data as clear text to the CSS for a decision on load balancing. The CSS transmits the data as clear text either to an HTTP server or back to the SSL module for encryption to a configured back-end SSL server.

Figure 4-1 illustrates an SSL connection between a client and a CSS configured with an SSL module acting as an SSL server.

Figure 4-1 SSL Termination



An SSL proxy list determines the flow of SSL information between the SSL module, the client, and the server. An SSL proxy list comprises one or more virtual SSL servers (related by index entry). An SSL module in the CSS uses the virtual SSL servers to properly process and terminate SSL communications between the client and the server. You can define a maximum of 256 virtual SSL servers for a single SSL proxy list.

After you create and configure the entries in a proxy list, you must activate the list, and then add the SSL proxy list to a service to initiate the transfer of SSL configuration data to the SSL module. When you activate the service, the CSS transfers the data to the module. Then you can add each SSL service to an SSL content rule.

Creating an SSL Proxy List

An SSL proxy list is a group of related virtual SSL servers that are associated with an SSL service. To create an SSL proxy list, use the **ssl-proxy-list** command.

You can access the `ssl-proxy-list` configuration mode from most configuration modes except for `ACL`, `boot`, `group`, `rmon`, or `owner` configuration modes. You can also use this command from the `ssl-proxy-list` configuration mode to access another SSL proxy list. Enter the SSL proxy list name as an unquoted text string from 1 to 31 characters.

For example, to create the SSL proxy list, `ssl_list1`, enter:

```
(config)# ssl-proxy-list ssl_list1
Create ssl-list <ssl_list1>, [y/n]: y
```

Once you create an SSL proxy list, the CLI enters into the `ssl-proxy-list` configuration mode.

```
(config-ssl-proxy-list[ssl_list1])#
```

To delete an existing SSL proxy list, enter:

```
(config)# no ssl-proxy-list ssl_list1  
Delete ssl-list <ssl_list1>, [y/n]: y
```

**Note**

You cannot delete an SSL proxy list if an SSL service is in use and contains the active SSL proxy list. You must first suspend the SSL service to delete a specific SSL proxy list.

Adding a Description to an SSL Proxy List

To specify a description for an SSL proxy list, use the **description** command. Enter the description as a quoted text string with a maximum of 64 characters, including spaces.

For example, to add a description to the `ssl_list1` SSL proxy list, enter:

```
(config-ssl-proxy-list[ssl_list1])# description "This is the SSL list  
for www.brandnewproducts.com"
```

To remove the description from a specific SSL proxy list, enter:

```
(config-ssl-proxy-list[ssl_list1])# no description
```

Configuring Virtual SSL Servers for an SSL Proxy List

This section discusses creating one or more virtual SSL servers for an SSL proxy list. Use the **ssl-server** command to define an index entry in the SSL proxy list that you then use to configure specific SSL parameters associated with the SSL proxy list. An SSL module in the CSS uses the virtual SSL servers to properly process and terminate SSL communications between the client and the server. You must define an **ssl-server** index number before configuring SSL proxy list parameters. You can define a maximum of 256 virtual SSL servers for a single SSL proxy list.

For example, suppose the e-commerce vendor Brand New Products, Inc. wants to configure the CSS to perform SSL termination. They need to divert all traffic intended for <https://www.brandnewproducts.com> to the SSL module in the CSS. To do this, they must identify a VIP address for a virtual SSL server in the SSL proxy list and link the list to the same VIP address as a content rule. The VIP address requires the following additional SSL configuration parameters:

- Identification of a virtual TCP port number that corresponds with a content rule
- An existing RSA or DSA certificate for identification purposes
- An appropriate SSL key pair to perform encryption and signing (assuming you are using an RSA key pair)
- Diffie-Hellman parameters if your CSS SSL security requires the Diffie-Hellman key exchange algorithm
- Assignment of a cipher suite

**Note**

You cannot modify the virtual SSL servers in an active SSL proxy list. You must first suspend the SSL proxy list to make modifications to any of the virtual SSL servers in a specific SSL proxy list. Once you have modified the SSL proxy list, suspend the SSL service, activate the SSL proxy list, and then activate the SSL service.

The following sections describe how to create a virtual SSL server for an SSL proxy list:

- [Creating an SSL Server Index](#)
- [Specifying a Virtual IP Address](#)
- [Specifying a Virtual Port](#)
- [Assigning Certificate, Key, and Cipher Suites for Server Authentication](#)
- [Configuring Client Authentication](#)
- [Configuring HTTP Header Insertion](#)
- [Specifying SSL or TLS Version](#)
- [Terminating a Client Connection with a TCP FIN Message Only](#)
- [Specifying Secure URL Rewrite](#)
- [Specifying SSL Session Cache Timeout](#)
- [Specifying SSL Session Handshake Renegotiation](#)
- [Configuring the Delay Time for SSL Queued Data](#)
- [Specifying SSL TCP Client-Side Connection Timeout Values](#)
- [Specifying SSL TCP Server-Side Connection Timeout Values](#)
- [Specifying the Nagle Algorithm for SSL TCP Connections](#)
- [Specifying the TCP Buffering for SSL TCP Connections](#)

To view configuration information on an SSL proxy list, see [Chapter 7, Displaying SSL Configuration Information and Statistics](#).

Creating an SSL Server Index

You must create a virtual SSL server before you can configure SSL proxy list parameters. To identify SSL-specific parameters for the SSL proxy list, use the **ssl-server number** command. This command creates a number (index entry) in the SSL proxy list that you use to configure specific SSL parameters associated with the virtual SSL server (for example, VIP address, certificate name, and key pair). Enter an integer from 1 to 256.

For example, to specify virtual SSL server 20, enter:

```
(config-ssl-proxy-list[ssl_list1])# ssl-server 20
```

To remove the virtual SSL server from the SSL proxy list, enter:

```
(config-ssl-proxy-list[ssl_list1])# no ssl-server 20
```

Specifying a Virtual IP Address

The SSL module uses this VIP address as the means to know which traffic it should accept. Ensure that the VIP address matches a VIP address configured in a content rule. Use the **ssl-server number vip address ip_or_host** command to specify a virtual IP (VIP) address. Enter a VIP address for the virtual SSL server that corresponds to an SSL content rule. See the [“Configuring a Content Rule for SSL Termination”](#) section.

Enter a valid VIP address in either dotted-decimal IP notation (for example, 192.168.11.1) or mnemonic host-name format (for example, myhost.mydomain.com).



Note

When you use the mnemonic host name format for the VIP address, the CSS uses its Domain Name Service (DNS) facility to translate host names such as myhost.mydomain.com to IP addresses such as 192.168.11.1. If the host name cannot be resolved, the VIP address setting is not accepted and an error message appears indicating host resolution failure. For details on configuring a Domain Name Service, refer to the *Cisco Content Services Switch Global Server Load-Balancing Configuration Guide*.

If the VIP address has not been defined for the virtual SSL sever when you activate the SSL proxy list (see the “[Specifying the Nagle Algorithm for SSL TCP Connections](#)” section), the CSS logs an error message and does not activate the SSL proxy list. When you activate a content rule with a configured SSL service, the CSS verifies that each VIP address configured in the content rule matches at least one VIP address configured in the SSL proxy list in each of the added services. If a match is not found, the CSS logs an error message and does not activate the content rule.

For example, to specify a VIP address for the virtual SSL server that corresponds to a VIP address configured in a content rule, enter:

```
(config-ssl-proxy-list[ssl_list1])# ssl-server 20 vip address  
192.168.3.6
```

To remove a VIP address from a specific virtual SSL server, enter:

```
(config-ssl-proxy-list[ssl_list1])# no ssl-server 20 vip address
```

Specifying a Virtual Port

The SSL module uses the virtual port to know which traffic it should accept. Use the **ssl-server number port number** command to specify a virtual TCP port number for the virtual SSL server. Enter a TCP port number that corresponds with an SSL content rule, which uses the specified TCP port number.

Specify a port number from 1 to 65535. The default port is 443. Ensure that the specified port number matches the port configured in a content rule (see the “[Configuring a Content Rule for SSL Termination](#)” section).

If the virtual port has not been defined for the virtual SSL server when you activate the SSL proxy list (see the “[Specifying the Nagle Algorithm for SSL TCP Connections](#)” section), the CSS logs an error message and does not activate the SSL proxy list. When you activate a content rule with a configured SSL service, the CSS verifies that each virtual port configured in the content rule matches at least one port configured in the SSL proxy list in each of the added services. If a match is not found, the CSS logs an error message and does not activate the content rule.

For example, to specify a virtual port of 444, enter:

```
(config-ssl-proxy-list[ssl_list1])# ssl-server 20 port 444
```

To reset the virtual port to the default of 443, enter:

```
(config-ssl-proxy-list[ssl_list1])# no ssl-server 20 port
```

Assigning Certificate, Key, and Cipher Suites for Server Authentication

The CSS supports server certificates that it sends to all clients for authentication. To identify a certificate with a virtual SSL server, you must assign the certificates and key that you have either imported to or generated on the CSS described in [Chapter 3, Configuring SSL Certificates and Keys](#). You must also assign the cipher suite that correlates to the certificates and keys.

The following sections provide information for configuring server authentication:

- [Specifying the RSA Certificate Name](#)
- [Specifying the RSA Key Pair Name](#)
- [Specifying the DSA Certificate Name](#)
- [Specifying the DSA Key Pair Name](#)
- [Specifying the Diffie-Hellman Parameter Filename](#)
- [Specifying Cipher Suites](#)

Specifying the RSA Certificate Name

To identify the name of an RSA certificate association to be used in the exchange of a public/private key pair for authentication and packet encryption, use the **ssl-server number rsacert name** command. To see a list of existing RSA certificate associations, use the **ssl-server number rsacert ?** command.

The specified RSA certificate must already be loaded on the CSS and an association made (see [Chapter 3, Configuring SSL Certificates and Keys](#)). If there is not a proper RSA certificate association, when you activate the SSL proxy list, the CSS logs an error message and does not activate the list.

For example, to specify a previously defined RSA certificate association named *rsacert*, enter:

```
(config-ssl-proxy-list[ssl_list1])# ssl-server 20 rsacert myrsacert1
```

To remove an RSA certificate association from a specific virtual SSL server, enter:

```
(config-ssl-proxy-list[ssl_list1])# no ssl-server 20 rsacert
```

Specifying the RSA Key Pair Name

To identify the name of an RSA key pair association. RSA key pairs are required before another device (client or server) can exchange an SSL certificate with the CSS, use the **ssl-server *number* *rsa*key *name*** command. To see a list of existing RSA key pair associations, use the **ssl-server *number* *rsa*key ?** command.

The RSA key pair must already be loaded on the CSS and an association made (see [Chapter 3, Configuring SSL Certificates and Keys](#)). If there is not a proper RSA key pair association, when you activate the SSL proxy list, the CSS logs an error message and does not activate the list.

For example, to specify a previously defined RSA key pair association named *rsa*key, enter:

```
(config-ssl-proxy-list[ssl_list1])# ssl-server 20 rsakey myrsakey1
```

To remove an RSA key pair association from a specific virtual SSL server, enter:

```
(config-ssl-proxy-list[ssl_list1])# no ssl-server 20 rsakey
```

Specifying the DSA Certificate Name

To identify the name of a DSA certificate association that is to be used in the exchange of digital signatures, use the **ssl-server *number* *ds*acert *name*** command. To see a list of existing DSA certificate associations, use the **ssl-server *number* *ds*acert ?** command.

The specified DSA certificate must already be loaded on the CSS and an association made (see [Chapter 3, Configuring SSL Certificates and Keys](#)). If there is not a proper RSA certificate association, when you activate the SSL proxy list, the CSS logs an error message and does not activate the list.

For example, to specify a previously defined DSA certificate association named *ds*acert, enter:

```
(config-ssl-proxy-list[ssl_list1])# ssl-server 20 dsacert mydsacert1
```

To remove a DSA certificate association from a specific virtual SSL server, enter:

```
(config-ssl-proxy-list[ssl_list1])# no ssl-server 20 dsacert
```

Specifying the DSA Key Pair Name

DSA key pairs are used to sign packet data, and they are required before another device (client or server) can exchange an SSL certificate with the CSS. Use the **ssl-server number dsakey name** command to identify the name of a DSA key pair association. To see a list of existing DSA key pair associations, use the **ssl-server number dsakey ?** command.

The DSA key pair must already be loaded on the CSS and an association made (see [Chapter 3, Configuring SSL Certificates and Keys](#)). If there is not a proper DSA key pair association, when you activate the SSL proxy list, the CSS logs an error message and does not activate the list.

For example, to specify a previously defined DSA key pair association named *dsakey*, enter:

```
(config-ssl-proxy-list[ssl_list1])# ssl-server 20 dsakey mydsakey1
```

To remove a DSA key pair association from a specific virtual SSL server, enter:

```
(config-ssl-proxy-list[ssl_list1])# no ssl-server 20 dsakey
```

Specifying the Diffie-Hellman Parameter Filename

The Diffie-Hellman key exchange parameter file ensures that the two devices in a data exchange cooperate to generate a shared key for packet encryption and authentication. Use the **ssl-server number dhparam name** command to identify the name of a Diffie-Hellman key exchange parameter file association. To see a list of existing Diffie-Hellman key exchange parameter files, use the **ssl-server number dhparam ?** command.

The Diffie-Hellman parameter file must already be loaded on the CSS and an association made (see [Chapter 3, Configuring SSL Certificates and Keys](#)). If there is not a proper Diffie-Hellman parameter file association, when you activate the SSL proxy list, the CSS logs an error message and does not activate the list.

To specify a previously defined Diffie-Hellman parameter file association, enter:

```
(config-ssl-proxy-list[ssl_list1])# ssl-server 20 dhparam mydhparams1
```

To remove a Diffie-Hellman parameter file association from a specific virtual SSL server, enter:

```
(config-ssl-proxy-list[ssl_list1])# no ssl-server 20 dhparam
```

Specifying Cipher Suites

The SSL protocol supports a variety of different cryptographic algorithms, or ciphers, for use in operations such as authenticating the server and client to each other, transmitting certificates, and establishing session keys. Clients and servers may support different cipher suites, or sets of ciphers, depending on various factors such as the version of SSL they support, company policies regarding acceptable encryption strength, and government restrictions on export of SSL-enabled software. Among its other functions, the SSL handshake protocol determines how the server and client negotiate which cipher suites they will use to authenticate each other to transmit certificates and to establish session keys.

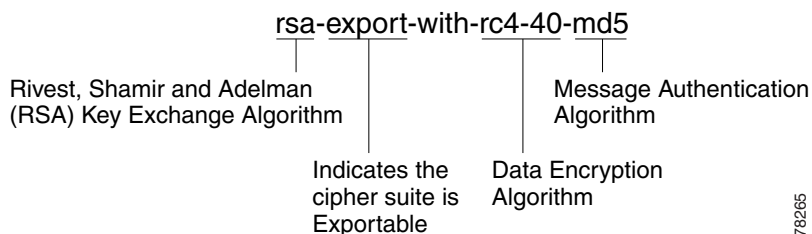


Note

Exportable cipher suites are those cipher suites that are considered not to be as strong as some of the other cipher suites (for example, 3DES or RC4 with 128-bit encryption) as defined by U.S. export restrictions on software products. Exportable cipher suites may be exported to most countries from the United States, and provide the strongest encryption available for exportable products.

Each cipher suite specifies a set of key exchange algorithms. [Figure 4-2](#) summarizes the algorithms associated with the `rsa-export-with-rc4-40-md5` cipher suite.

Figure 4-2 Cipher Suite Algorithms



Use the `ssl-server number cipher` command to assign a cipher suite for the SSL proxy list. The cipher suite that you choose must correlate to the certificates and keys that you have either imported to or generated on the CSS. For example, if you choose **all-cipher-suites**, you must have an RSA certificate and key, a DSA certificate and key, and a Diffie-Hellman parameter file prior to activating the SSL proxy list.

For each available SSL version, there is a distinct list of supported cipher suites representing a selection of cryptographic algorithms and parameters. Your choice depends on your environment, certificates and keys in use, and security requirements. By default, no supported cipher suites are enabled.

The syntax for this command is:

```
ssl-server number cipher name ip_address or hostname port {weight number}
```

The options and variables are:

- **ssl-server** *number* - The number used to identify the virtual SSL server in the SSL proxy list.
- **cipher** *name* - The name of a specific cipher suite (as listed in [Table 4-1](#)).
- *ip_address* or *hostname* - The IP address to assign to the back-end content rule used with the cipher suite. Specify the IP address in either dotted-decimal IP notation (for example, 192.168.11.1) or mnemonic host-name format (for example, myhost.mydomain.com).
- *port* - The TCP port of the back-end content rule through which the back-end HTTP connections are sent.
- **weight** *number* - Optional parameter. Assigns a priority to the cipher suite, with 10 being the highest weight. By default, all configured cipher suites have a weight of 1. When negotiating which cipher suite to use, the SSL module selects from the client list based on the cipher suite configured with the highest weight. A higher weight will bias towards the specified cipher suite. To set the weight for a cipher suite, enter a number from 1 to 10. The default is 1.

For example, to select the *dhe-rsa-with-3des-ede-cbc-sha* cipher suite with an assigned weight of 5, enter:

```
(config-ssl-proxy-list[ssl_list1])# ssl-server 20 cipher  
dhe-rsa-with-3des-ede-cbc-sha 192.168.11.1 80 weight 5
```

To remove a specific cipher suite from a specific virtual SSL server, enter:

```
(config-ssl-proxy-list[ssl_list1])# no ssl-server 20 cipher  
dhe-rsa-with-3des-ede-cbc-sha
```

[Table 4-1](#) lists all supported cipher suites and values for the specific SSL server (and corresponding SSL proxy list). [Table 4-1](#) also lists whether those cipher suites are exportable from the CSS, along with the authentication certificate and encryption key required by the cipher suite.

If you use the default setting or select the **all-cipher-suite** option, the CSS sends the suites in the same order as they appear in [Table 4-1](#), starting with `rsa-with-rc4-128-md5`.

**Note**

The **all-cipher-suites** setting works only when no specifically-defined ciphers are configured. To return to using the **all-cipher-suites** setting, you must remove all specifically-defined ciphers.

**Caution**

The `dh-anon` series of cipher suites are intended for completely anonymous Diffie-Hellman communications in which neither party is authenticated. Note that this cipher suite is vulnerable to attacks.

Cipher suites with “export” in the title indicate that they are intended for use outside of the domestic United States and that they have encryption algorithms with limited key sizes.

Table 4-1 SSL Cipher Suites Supported by the CSS

Cipher Suite	Exportable	Authentication Certificate Used	Key Exchange Algorithm Used
all-cipher-suites	No	RSA certificate, DSA certificate	RSA key exchange, Diffie-Hellman
rsa-with-rc4-128-md5	No	RSA certificate	RSA key exchange
rsa-with-rc4-128-sha	No	RSA certificate	RSA key exchange
rsa-with-des-cbc-sha	No	RSA certificate	RSA key exchange
rsa-with-3des-ede-cbc-sha	No	RSA certificate	RSA key exchange
dhe-dss-with-des-cbc-sha	No	DSA (DSS) certificate	Ephemeral Diffie-Hellman
dhe-dss-with-3des-ede-cbc-sha	No	DSA (DSS) certificate	Ephemeral Diffie-Hellman
dhe-rsa-with-des-cbc-sha	No	RSA certificate	Ephemeral Diffie-Hellman key exchange

Table 4-1 SSL Cipher Suites Supported by the CSS (continued)

Cipher Suite	Exportable	Authentication Certificate Used	Key Exchange Algorithm Used
dhe-rsa-with-3des-ede-cbc-sha	No	RSA certificate	Ephemeral Diffie-Hellman key exchange
dh-anon-with-rc4-128-md5	No	Neither party is authenticated	Diffie-Hellman
dh-anon-with-des-cbc-sha	No	Neither party is authenticated	Diffie-Hellman
dh-anon-with-3des-ede-cbc-sha	No	Neither party is authenticated	Diffie-Hellman
dhe-dss-with-rc4-128-sha	No	DSA (DSS) certificate	Ephemeral Diffie-Hellman
rsa-export-with-rc4-40-md5	Yes	RSA certificate	RSA key exchange
rsa-export-with-des40-cbc-sha	Yes	RSA certificate	RSA key exchange
dhe-dss-export-with-des40-cbc-sha	Yes	DSA (DSS) certificate	Ephemeral Diffie-Hellman key exchange
dhe-rsa-export-with-des40-cbc-sha	Yes	RSA certificate	Ephemeral Diffie-Hellman
dh-anon-export-with-rc4-40-md5	Yes	Neither party is authenticated	Diffie-Hellman
dh-anon-export-with-des40-cbc-sha	Yes	Neither party is authenticated	Diffie-Hellman
rsa-export1024-with-des-cbc-sha	Yes	RSA certificate	RSA key exchange
dhe-dss-export1024-with-des-cbc-sha	Yes	DSA (DSS) certificate	Ephemeral Diffie-Hellman
rsa-export1024-with-rc4-56-sha	Yes	RSA certificate	RSA key exchange
dhe-dss-export1024-with-rc4-56-sha	Yes	DSA (DSS) certificate	Ephemeral Diffie-Hellman

Configuring Client Authentication

For additional security, you can configure the SSL proxy server to request certificates from clients. By default, client certificate authentication is disabled. When you enable client authentication, the CSS requires the client to exchange a certificate during the SSL handshake. The CSS verifies that the:

- Client sending the certificate has a corresponding key
- Certificate has not expired
- Signature is valid
- Issuing CA has not revoked the certificate

You can configure how the CSS handles a certificate that has expired, is invalid, or has been revoked.

The following sections provide information on configuring client authentication:

- [Enabling Client Authentication](#)
- [Specifying CA Certificates for Client Certificate Verification](#)
- [Configuring a CRL Record](#)
- [Assigning a CRL Record to the Virtual SSL Server](#)
- [Handling Client Authentication Failures](#)

To view client authentication configuration information, use the **show ssl-proxy-list ssl-server** command. To view SSL counters for client authentication-related activities, use the **show ssl statistics** command. See [Chapter 7, Displaying SSL Configuration Information and Statistics](#) for more information.

Enabling Client Authentication

By default, client authentication is disabled on the CSS. The **authentication** option of the **ssl-server** command allows you to enable or disable client authentication. For example, to enable client authentication, enter:

```
(config-ssl-proxy-list[ssl_list1])# ssl-server 20 authentication enable
```

To reset the default setting of disabling client authentication, enter:

```
(config-ssl-proxy-list[ssl_list1])# no ssl-server 20 authentication
```

You can also reset the default setting of disabling client authentication by using the **disable** option. For example, enter:

```
(config-ssl-proxy-list[ssl_list1])# ssl-server 20 authentication disable
```

After you enable client authentication on the CSS, you must specify a CA certificate that the CSS uses to verify client certificates.

Specifying CA Certificates for Client Certificate Verification

CA certificates contain the public key of the CA. If a server has the CA public key, it can verify that a client certificate was signed by the CA. If you assign a CA certificate to a virtual SSL server, the CSS uses the key in the certificate to verify the digital signature in the client certificate.



Note

You must configure a CA certificate before you activate the SSL proxy list.

Before you configure the certificate on a virtual SSL server, you must import a CA certificate on the CSS and then associate it with a filename. For information on importing a CA certificate, see the “[Importing or Exporting Certificates and Private Keys](#)” section in [Chapter 3, Configuring SSL Certificates and Keys](#). For information on associating a certificate with a filename, see the “[Associating a Certificate with a File](#)” also in [Chapter 3, Configuring SSL Certificates and Keys](#).

You must configure at least one certificate; however, you can configure a maximum of four certificates. If you try to configure more than four certificates, the CSS displays an error message.

After you enable client authentication, you can assign the CA certificates to the virtual SSL server through the **ssl-server number cacert** command. For example, to specify the mycert1 CA certificate association to the virtual SSL server, enter:

```
(config-ssl-proxy-list[ssl_list1])# ssl-server 20 cacert mycert1
```

To remove a certificate association from the virtual SSL server, use the **no** form of the **ssl-server number cacert** command. For example, to remove the mycert1 CA certificate association, enter:

```
(config-ssl-proxy-list[ssl_list1])# no ssl-server 20 cacert mycert1
```

Configuring a CRL Record

When a CA revokes a certificate, the CA places the certificate on a certificate revocation list (CRL) and publishes it for public availability. For the CSS to use a CRL, it must obtain the CRL from its current location and download it via HTTP. To do so, you must create a CRL record on the CSS. The CRL record contains the complete URL information for the CSS to obtain the CRL from its current location and import it periodically. After you configure the CRL record, you can assign it to the virtual SSL server.



Note

The HTTP request to retrieve the CRL has a source IP address that is the VIP address of the virtual SSL server.

You can assign one to a virtual SSL server, however you can configure the CSS to store up to 10 CRL records. To configure the CRL record, use the **ssl crl-record** command in global configuration mode. The syntax for the command is:

```
ssl crl-record crl_name url sign_cert hours
```

The variables are:

- *crl_name* - The name for the CRL record. Enter a string with a maximum of 31 characters and no spaces.
- *url* - The URL where the CRL is located. Enter a string with a maximum of 168 characters and no spaces (for example, <http://www.example.com/crl/clientcert.crl>).

- *sign_cert* - The name of the CA certificate that signed the CRL. The CA certificate verifies that the CRL is authentic. You must import this certificate on the CSS before configuring the CRL. For information on importing a CA certificate, see the “[Importing or Exporting Certificates and Private Keys](#)” section in [Chapter 3, Configuring SSL Certificates and Keys](#). For information on associating a certificate with a filename, see the “[Associating a Certificate with a File](#)” also in [Chapter 3, Configuring SSL Certificates and Keys](#).
- *hours* - The number of hours to wait before retrieving an updated CRL. Enter a value from 0 to 2000. A value of 0 disables the retrieval of the CRL, which means that the CRL is not updated.

The CSS SSL module keeps a list of all configured CRLs. The module only attempts to retrieve a CRL when:

- The SSL proxy list containing CRL records is activated
- The service or content rule associated with the SSL proxy list is activated
- The CRL was previously retrieved and the time defined in the CRL record has now passed

The following example shows how to configure a CRL record named `mycrl`. The URL location of the CRL is `crl.verisign.com`. The CA certificate name on the CSS that authenticates the CRL is `verisign_cacert`. The CSS updates the CRL every 24 hours. Enter:

```
(config)# ssl crl-record mycrl http://crl.verisign.com/class1.crl  
verisign_cacert 24
```

To remove the CRL record, enter:

```
(config)# no ssl crl-record mycrl
```

To view configuration information on a CRL, use the **show ssl crl-record** command. For more information on this command, see [Chapter 7, Displaying SSL Configuration Information and Statistics](#).

Assigning a CRL Record to the Virtual SSL Server

After you configure the CRL record, you can assign it to the virtual SSL server. To assign the CRL record to the virtual SSL server, use the **ssl-server number crl** command. You can assign only one CRL record to a virtual SSL server. For example, to assign the **mycrl** CRL record, enter:

```
(config-ssl-proxy-list[ssl_list1])# ssl-server 20 crl mycrl
```

To remove the **mycrl** CRL record from a virtual SSL server, enter:

```
(config-ssl-proxy-list[ssl_list1])# no ssl-server 20 crl mycrl
```

Handling Client Authentication Failures

A client certificate can fail if it is invalid, expired, or revoked by a CA. By default, when authentication of a client certificate fails on the CSS, the CSS rejects the client connection.



Note

If a CSS cannot download the CRL, client connections will fail using a Revoked SSL alert. To verify that the CRL has successfully loaded, use the **show ssl statistics ssl** command.

You can configure how the CSS handles a failed client certificate through the **ssl-server number failure** command and the following options:

- **ignore** - The CSS ignores client authentication failures and allows both invalid and valid certificates to connect. For example, to configure the CSS to ignore client authentication failures, enter:

```
(config-ssl-proxy-list[ssl_list1])# ssl-server 20 failure ignore
```



Note

If you configure the **ignore** option, it may create a security risk.

- **reject** - Resets the CSS default behavior of rejecting the client connection when client authentication fails. For example, enter:

```
(config-ssl-proxy-list[ssl_list1])# ssl-server 20 failure reject
```

- **redirect** - The CSS sends connections of failed authentications to a configured URL.

```
(config-ssl-proxy-list[ssl_list1])# ssl-server 20 failure redirect
```

To configure the URL where the CSS redirects the client connection, use the **ssl-server *number* failure-url** command. Enter a URL with a maximum of 168 characters and no spaces. For example, to redirect the client connection to URL `www.service_css.com` when client authentication fails, enter:

```
(config-ssl-proxy-list[ssl_list1])# ssl-server 20 failure-url  
http://www.service_css.com
```

If you want to change an existing redirect URL, you must use the **no ssl-server *number* failure-url** command to remove it, and then reissue the **ssl-server *number* failure-url** command to configure the new URL. Note that you must suspend an activated virtual SSL server before modifying it.

For example, to remove the URL, enter:

```
(config-ssl-proxy-list[ssl_list1])# no ssl-server 20 failure-url
```

**Note**

Regardless of the failure settings, the CSS logs a client authentication failure as an error message in syslog.

Configuring HTTP Header Insertion

During an SSL connection, a client may need to pass specific information to a back-end server. HTTP header insertion allows the embedding of information into an HTTP header during a client connection. For example, when a client connects to the virtual SSL server and the CSS decrypts the data, the CSS can insert information about the SSL session, and insert the client and server certificate into the HTTP request header, and then pass the header to the back-end server.

**Note**

HTTP header insertion only occurs on the first HTTP request for a persistent HTTP 1.1 connection. Subsequent requests within the same TCP connection are sent unmodified. For HTTP 1.0, in which persistence is not implemented, all HTTP requests contain the inserted header.

The CSS can insert one or more of the following information into the HTTP header after it decrypts the client data:

- Client certificate fields and associated information
- Server certificate fields and associated information
- SSL session fields and associated information
- Static text string

You can also configure the CSS to add a prefix to the client certificate, server certificate, or SSL session fields inserted in the HTTP header. However, a prefix has no effect on insertion of a static text string.

The following sections provide information on HTTP header insertion:

- [Inserting Client Certificate Information](#)
- [Inserting Server Certificate Information](#)
- [Inserting Session Information](#)
- [Adding a Prefix to the Fields Inserted in the HTTP Header](#)
- [Inserting a Static Text String](#)

To view configuration information on an HTTP header insertion, use the **show ssl-proxy-list ssl-server** command. For more information on this command, see [Chapter 7, Displaying SSL Configuration Information and Statistics](#).

Inserting Client Certificate Information

When you need to send client certificate information to the back-end server, you can configure the CSS to insert client certificate fields and associated information into the HTTP header. To add a prefix to the fields, see the [“Adding a Prefix to the Fields Inserted in the HTTP Header”](#) section.



Note

If the SSL proxy list and its service are active, suspend the service and then the proxy list before configuring or disabling HTTP header insertion. Afterward, reactivate the SSL proxy list and activate its service.

To configure the CSS to insert client certificate fields in the HTTP header, use the **ssl-server number http-header client-cert** command. For example:

```
(config-ssl-proxy-list[ssl_list1])# ssl-server 20 http-header client-cert
```

To disable the insertion of client certificate fields and information in the HTTP header, enter:

```
(config-ssl-proxy-list[ssl_list1])# no ssl-server 20 http-header client-cert
```

Table 4-2 lists the inserted client certificate fields, description, format, and an example. Depending on how the certificate was generated and what key algorithm was used, all of these fields may not be present for the certificate.

Table 4-2 Client Certificate Fields Inserted in the HTTP Header

Client Certificate Field	Description, Format, and Example
ClientCert-Fingerprint	Description: Hash Output Format: ASCII string of hexadecimal bytes separated by colons Example: ClientCert-Fingerprint: 64:75:CE:AD:9B:71:AC:25:ED:FE:DB:C7:4B:D4:1A:BA
ClientCert-Subject-CN	Description: X.509 subject's common name Format: String of characters representing the common name of the subject to whom the certificate has been issued Example: ClientCert-Subject-CN: www.cisco.com
ClientCert-Issuer-CN	Description: X.509 Certificate Issuer's Common Name Format: String of characters representing the common name for the certificate issuer Example: ClientCert-Issuer-CN: www.exampleca.com
ClientCert-Certificate-Version	Description: X.509 Certificate Version Format: Numerical X.509 version (3, 2, or 1), followed by the ASN.1 defined value for X.509 version (2, 1, or 0) in parentheses Example: ClientCert-Certificate-Version: 3 (0x2)

Table 4-2 Client Certificate Fields Inserted in the HTTP Header (continued)

Client Certificate Field	Description, Format, and Example
ClientCert-Serial-Number	<p>Description: Certificate serial number</p> <p>Format: A whole integer value assigned by the certificate authority; this can be any arbitrary integer value</p> <p>Example: ClientCert-Serial-Number: 2</p>
ClientCert-Data-Signature-Algorithm	<p>Description: X.509 Hashing and Encryption Method</p> <p>Format: The md5WithRSAEncryption, sha1WithRSAEncryption, or dsaWithSHA1 algorithm used to sign the certificate and algorithm parameters</p> <p>Example: ClientCert-Signature-Algorithm: md5WithRSAEncryption</p>
ClientCert-DSA-Public-Key-Size	<p>Description: The size in bits of the DSA public key, if the certificate contains a DSA key.</p> <p>Format: Number of bits as a whole integer followed by the word bit</p> <p>Example: ClientCert-DSA-Public-Key-Size: 1024 bit</p>
ClientCert-DSA-Public-Key	<p>Description: The DSA algorithm public key. Only used if the certificate contains a DSA key.</p> <p>Format: The key is printed in big-endian format hexadecimal, without leading 0x, and lowercase alphanumeric characters separated by a colon (:) character</p> <p>Example: ClientCert-DSA-Public-Key: 00:d8:1b:94:de:52:a1:20:51:b1:77</p>
ClientCert-Subject	<p>Description: X.509 subject's distinguished name</p> <p>Format: String of characters representing the subject that owns the private key being certified</p> <p>Example: ClientCert-Subject: CN=Example, ST=Virginia, C=US/Email=ca@example.com, 0=Root</p>

Table 4-2 Client Certificate Fields Inserted in the HTTP Header (continued)

Client Certificate Field	Description, Format, and Example
ClientCert-Issuer	<p>Description: X.509 Certificate Issuer's Distinguished Name</p> <p>Format: String of characters representing the certificate authority that issued this certificate</p> <p>Example: ClientCert-Issuer: CN=Example CA, ST=Virginia, C=US/Email=ca@exampleca.com, O=Root</p>
ClientCert-Not-After	<p>Description: Certificate is not valid after this date</p> <p>Format: A universal time string or generalized time string in the Not After date of the Validity field</p> <p>Example: ClientCert-Not-After: 2003-1-27 23:59.59 UTC</p>
ClientCert-Not-Before	<p>Description: Certificate is not valid before this date</p> <p>Format: A universal time string or generalized time string in the Not Before date of the Validity field</p> <p>Example: ClientCert-Not-Before: 2002-1-27 00:00:00.00 UTC</p>
ClientCert-Public-Key-Algorithm	<p>Description: The algorithm used for the public key</p> <p>Format: The rsaEncryption, rsa, or dsaEncryption public key algorithm used to create the public key in the certificate</p> <p>Example: ClientCert-Public-Key-Algorithm: rsaEncryption</p>
ClientCert-RSA-Modulus-Size	<p>Description: Size of the RSA public key</p> <p>Format: Number of bits as a whole integer of the RSA modulus (typically 512, 1024, or 2048), followed by the word bit</p> <p>Example: ClientCert-RSA-Modulus-Size: 1024 bit</p>
ClientCert-RSA-Modulus	<p>Description: RSA modulus</p> <p>Format: The RSA algorithm modulus (n) printed in big-endian format hexadecimal, without leading 0x, and lowercase alphanumeric characters separated by a colon (:) character. Together with the exponent (e), this modulus forms the public key portion in the RSA certificate</p> <p>Example: ClientCert-RSA-Modulus: +00:d8:1b:94:de:52:a1:20:51:b1:77</p>

Table 4-2 Client Certificate Fields Inserted in the HTTP Header (continued)

Client Certificate Field	Description, Format, and Example
ClientCert-RSA-Exponent	<p>Description: The public RSA exponent</p> <p>Format: Printed as a whole integer for the RSA algorithm exponent (e)</p> <p>Example: ClientCert-RSA-Exponent: 65537</p>
ClientCert-Subject-Key-Identifier	<p>Description: X.509 subject key identifier</p> <p>Format: ASCII string of hexadecimal bytes separated by colons for the X.509 version 3 subject key identifier</p> <p>Example: ClientCert-Subject-Key-Identifier: 16:13:15:97:FD:8E:16:B9:D2:99</p>
ClientCert-Authority-Key-Identifier	<p>Description: X.509 authority key identifier</p> <p>Format: ASCII string of hexadecimal bytes separated by colons for the X.509 version 3 Authority Key Identifier</p> <p>Example: ClientCert-Authority-Key-Identifier: 16:13:15:97:FD:8E:16:B9:D2:99</p>
ClientCert-Basic-Constraints	<p>Description: X.509 basic constraints</p> <p>Format: String listing whether the certificate subject can act as a certificate authority. Possible values are CA=TRUE or CA=FALSE basic constraints</p> <p>Example: ClientCert-Basic-Constraints: CA=TRUE</p>
ClientCert-Signature-Algorithm	<p>Description: Certificate signature algorithm</p> <p>Format: The md5WithRSAEncryption, sha1WithRSAEncryption, or dsaWithSHA1 for the secure hash algorithm</p> <p>Example: ClientCert-Signature-Algorithm: md5WithRSAEncryption</p>
ClientCert-Signature	<p>Description: Certificate signature</p> <p>Format: Secure hash of the other fields in the certificate and a digital signature of the hash printed in big-endian format hexadecimal, without leading 0x, and lowercase alphanumeric characters separated by a colon (:) character</p> <p>Example: ClientCert-Signature: 33:75:8e:a4:05:92:65</p>

Inserting Server Certificate Information

When you need to send server certificate information to the back-end server, you can configure the CSS to insert server certificate fields and associated information. The server certificate resides on the CSS and is configured with the **ssl-server** *number* **rsacert** or **dsacert** command. To add a prefix to the fields, see the [“Adding a Prefix to the Fields Inserted in the HTTP Header”](#) section.



Note

If the SSL proxy list and its service are active, suspend the service and then the proxy list before configuring or disabling HTTP header insertion. After, reactivate the SSL proxy list and then activate its service.

To configure the insertion server certificate information, use the **ssl-server** *number* **http-header server-cert** command. For example:

```
(config-ssl-proxy-list[ssl_list1])# ssl-server 20 http-header  
server-cert
```

To disable the insertion of server certificate fields and information in the HTTP header, enter:

```
(config-ssl-proxy-list[ssl_list1])# no ssl-server 20 http-header  
server-cert
```

[Table 4-3](#) lists the inserted server certificate fields and their descriptions. Depending on how the certificate was generated and what key algorithm was used, all of these fields may not be present for the certificate.

Table 4-3 Server Certificate Fields Inserted In the HTTP Header

Field	Description
ServerCert-Fingerprint	<p>Description: Hash Output</p> <p>Format: ASCII string of hexadecimal bytes separated by colons</p> <p>Example: ServerCert-Fingerprint: 64:75:CE:AD:9B:71:AC:25:ED:FE:DB:C7:4B:D4:1A:BA</p>
ServerCert-Subject-CN	<p>Description: X.509 subject's common name</p> <p>Format: String of characters representing the common name of the subject to whom the certificate has been issued</p> <p>Example: ServerCert-Subject-CN: www.cisco.com</p>
ServerCert-Issuer-CN	<p>Description: X.509 Certificate Issuer's Common Name</p> <p>Format: String of characters representing the common name for the certificate issuer</p> <p>Example: ServerCert-Issuer-CN: www.exampleca.com</p>
ServerCert-Certificate-Version	<p>Description: X.509 Certificate Version</p> <p>Format: Numerical X.509 version (3, 2, or 1), followed by the ASN.1 defined value for X.509 version (2, 1, or 0) in parentheses</p> <p>Example: ServerCert-Certificate-Version: 3 (0x2)</p>
ServerCert-Serial-Number	<p>Description: Certificate serial number</p> <p>Format: A whole integer value assigned by the certificate authority; this can be any arbitrary integer value</p> <p>Example: ServerCert-Serial-Number: 2</p>
ServerCert-Data-Signature-Algorithm	<p>Description: X.509 Hashing and Encryption Method</p> <p>Format: The md5WithRSAEncryption, sha1WithRSAEncryption, or dsaWithSHA1 algorithm used to sign the certificate and algorithm parameters</p> <p>Example: ServerCert-Signature-Algorithm: md5WithRSAEncryption</p>

Table 4-3 Server Certificate Fields Inserted In the HTTP Header (continued)

Field	Description
ServerCert-DSA-Public-Key-Size	<p>Description: The size in bits of the DSA public key, if the certificate contains a DSA key.</p> <p>Format: Number of bits as a whole integer followed by the word bit</p> <p>Example: ServerCert-DSA-Public-Key-Size: 1024 bit</p>
ServerCert-DSA-Public-Key	<p>Description: The DSA algorithm public key. Only used if the certificate contains a DSA key.</p> <p>Format: The key is printed in big-endian format hexadecimal, without leading 0x, and lowercase alphanumeric characters separated by a colon (:) character</p> <p>Example: ServerCert-DSA-Public-Key: 00:d8:1b:94:de:52:a1:20:51:b1:77</p>
ServerCert-Subject	<p>Description: X.509 subject's distinguished name</p> <p>Format: String of characters representing the subject that owns the private key being certified</p> <p>Example: ServerCert-Subject: CN=Example, ST=Virginia, C=US/Email=ca@example.com, O=Root</p>
ServerCert-Issuer	<p>Description: X.509 Certificate Issuer's Distinguished Name</p> <p>Format: String of characters representing the certificate authority that issued this certificate</p> <p>Example: ServerCert-Issuer: CN=Example CA, ST=Virginia, C=US/Email=ca@exampleca.com, O=Root</p>
ServerCert-Not-After	<p>Description: Certificate is not valid after this date</p> <p>Format: A universal time string or generalized time string in the Not After date of the Validity field</p> <p>Example: ServerCert-Not-After: 2003-1-27 23:59.59 UTC</p>

Table 4-3 Server Certificate Fields Inserted In the HTTP Header (continued)

Field	Description
ServerCert-Not-Before	<p>Description: Certificate is not valid before this date</p> <p>Format: A universal time string or generalized time string in the Not Before date of the Validity field</p> <p>Example: ServerCert-Not-Before: 2002-1-27 00:00:00.00 UTC</p>
ServerCert-Public-Key-Algorithm	<p>Description: The algorithm used for the public key</p> <p>Format: The rsaEncryption, rsa, or dsaEncryption public key algorithm used to create the public key in a certificate</p> <p>Example: ServerCert-Public-Key-Algorithm: rsaEncryption</p>
ServerCert-RSA-Modulus-Size	<p>Description: Size of the RSA public key</p> <p>Format: Number of bits as a whole integer of the RSA modulus (typically 512, 1024, or 2048), followed by the word bit</p> <p>Example: ServerCert-RSA-Modulus-Size: 1024 bit</p>
ServerCert-RSA-Modulus	<p>Description: RSA modulus</p> <p>Format: The RSA algorithm modulus (n) printed in big-endian format hexadecimal, without leading 0x, and lowercase alphanumeric characters separated by a colon (:) character. Together with the exponent (e), this modulus forms the public key portion in the RSA certificate.</p> <p>Example: ServerCert-RSA-Modulus: +00:d8:1b:94:de:52:a1:20:51:b1:77</p>
ServerCert-RSA-Exponent	<p>Description: The public RSA exponent</p> <p>Format: Printed as a whole integer for the RSA algorithm exponent (e)</p> <p>Example: ServerCert-RSA-Exponent: 65537</p>

Table 4-3 Server Certificate Fields Inserted In the HTTP Header (continued)

Field	Description
ServerCert-Subject-Key-Identifier	<p>Description: X.509 subject key identifier</p> <p>Format: ASCII string of hexadecimal bytes separated by colons for the X.509 version 3 subject key identifier</p> <p>Example: ServerCert-Subject-Key-Identifier: 16:13:15:97:FD:8E:16:B9:D2:99</p>
ServerCert-Authority-Key-Identifier	<p>Description: X.509 authority key identifier</p> <p>Format: ASCII string of hex bytes separated by colons for the X.509 version 3 Authority Key Identifier</p> <p>Example: ServerCert-Authority-Key-Identifier: 16:13:15:97:FD:8E:16:B9:D2:99</p>
ServerCert-Basic-Constraints	<p>Description: X.509 basic constraints</p> <p>Format: String listing whether the certificate subject can act as a certificate authority. Possible values are CA=TRUE or CA=FALSE</p> <p>Example: ServerCert-Basic-Constraints: CA=TRUE</p>
ServerCert-Signature-Algorithm	<p>Description: Certificate signature algorithm</p> <p>Format: The md5WithRSAEncryption, sha1WithRSAEncryption, or dsaWithSHA1 for the secure hash algorithm</p> <p>Example: ServerCert-Signature-Algorithm: md5WithRSAEncryption</p>
ServerCert-Signature	<p>Description: Certificate signature</p> <p>Format: Secure hash of the other fields in the certificate and a digital signature of the hash printed in big-endian format hexadecimal, without leading 0x, and lowercase alphanumeric characters and separated by a colon (:) character</p> <p>Example: ServerCert-Signature: 33:75:8e:a4:05:92:65</p>

Inserting Session Information

When you want to send SSL session information to the back-end server, you can configure the CSS to insert SSL session fields and associated information. To add a prefix to the fields, see the [“Adding a Prefix to the Fields Inserted in the HTTP Header”](#) section.



Note

If the SSL proxy list and its service are active, suspend the service and then the proxy list before configuring or disabling HTTP header insertion. Afterward, reactivate the SSL proxy list and activate its service.

To configure the insertion of session information, use the **ssl-server *number* http-header session** command. For example:

```
(config-ssl-proxy-list[ssl_list1])# ssl-server 20 http-header session
```

To disable the insertion of SSL session fields and information in the HTTP header, enter:

```
(config-ssl-proxy-list[ssl_list1])# no ssl-server 20 http-header session
```

[Table 4-4](#) lists the inserted SSL session fields and their descriptions.

Table 4-4 SSL Session Fields Inserted In the HTTP Header

Field	Description
Session-Cipher-Name	Description: Symmetric cipher suite Format: The OpenSSL version name of the cipher suite negotiated during this session Example: Session-Cipher-Name: EXP1024-RC4-SHA
Session-Cipher-Key-Size	Description: Symmetric cipher key size Format: Whole integer representing the length in bytes of the public key Example: Session-Cipher-Key-Size: 128

Table 4-4 SSL Session Fields Inserted In the HTTP Header (continued)

Field	Description
Session-Cipher-Use-Size	Description: Symmetric cipher use Format: Whole integer representing the length in bytes of the key used for symmetric encryption during this session Example: Session-Cipher-Use-Size: 56
Session-Protocol-Version	Description: Version of SSL or TLS Format: SSL or TLS protocol followed by version number Example: Session-Protocol-Version: TLSv1
Session-Id	Description: SSL Session ID Format: The 32-byte session ID negotiated during this session if a session ID is or has been negotiated, printed in big-endian format; hexadecimal without leading 0x and lowercase alphanumeric characters separated by a colon (:). Example: Session-Id: 75:45:62:cf:ee:71:de:ad:be:ef:00:33:ee:23:89:25:75:45:62:cf:ee:71:de:ad:be:ef:00:33:ee:23:89:25
Session-Verify-Result	Description: SSL session verify result Format: Numeric value indicating the SSL session verify result Example: Session-Verify-Result: 0

Adding a Prefix to the Fields Inserted in the HTTP Header

To add a prefix to the client certificate, server certificate, or session fields inserted in the HTTP header, use the `ssl-server number http-header prefix` command. Enter a quoted text string with a maximum of 16 characters.



Note

If the SSL proxy list and its service are active, suspend the service and then the proxy list before configuring or disabling HTTP header insertion. Afterward, reactivate the SSL proxy list and then activate its service.

For example, to add the Acme-SSL prefix to all inserted fields, enter:

```
(config-ssl-proxy-list[ssl_list1])# ssl-server 20 http-header prefix  
"Acme-SSL"
```

The ClientCert-Certificate-Version field would now appear as Acme-SSL-ClientCert-Certificate-Version.

To reset the default of not including a prefix, enter:

```
(config-ssl-proxy-list[ssl_list1])# no ssl-server 20 http-header  
prefix
```

**Note**

The **ssl-server** *number* **http-header prefix** command affects only SSL fields that you configure to be inserted in the HTTP header. This command has no effect on the insertion of a static text string.

Inserting a Static Text String

The primary purpose of text string insertion in the HTTP header to a back-end server is to support Microsoft Outlook Web Access (OWA) applications; however, you may have other reasons to insert static text. To configure the insertion of a static text string, use the **ssl-server** *number* **http-header static** command. Enter a quoted text string with a maximum of 199 characters including spaces. For OWA support, enter the text string "FRONT-END-HTTPS: on".

**Note**

If the SSL proxy list and its service are active, suspend the service and then the proxy list before configuring or disabling HTTP header insertion. Afterward, reactivate the SSL proxy list and then active its service.

For example:

```
(config-ssl-proxy-list[ssl_list1])# ssl-server 20 http-header static  
"FRONT-END-HTTPS: on"
```

You can also insert multiple strings on different lines by using the `\r\n` characters in between each line. The `\r\n` characters that terminate the lines use 4 of the 199 characters. The following example shows the insertion of three strings, "FRONT-END-HTTPS: on", "session cache: on", and "vip address: www.acme.com".

```
(config-ssl-proxy-list[ssl_list1])# ssl-server 20 http-header static  
"FRONT-END-HTTPS: on\r\nsession cache: on\r\nvipaddress: www.acme.com"
```

To disable the insertion of the static string in the HTTP header and delete the string, enter:

```
(config-ssl-proxy-list[ssl_list1])# no ssl-server 20 http-header
static
```

Specifying SSL or TLS Version

By default, the SSL version is SSL version 3 and TLS version 1. The SSL module sends a ClientHello that has an SSL version 3 header with the ClientHello message set to TLS version 1.

Use the **ssl-server *number* *version* *protocol*** command to specify the SSL or Transport Layer Security (TLS) protocol version. The options include:

- **ssl-tls** - SSL protocol version 3.0 and TLS protocol version 1.0 (default)
- **ssl** - SSL protocol version 3.0
- **tls** - TLS protocol version 1.0

For example, to specify SSL version 3.0, enter:

```
(config-ssl-proxy-list[ssl_list1])# ssl-server 20 version ssl
```

To reset the SSL version to the default of SSL version 3.0 and TLS version 1.0, enter:

```
(config-ssl-proxy-list[ssl_list1])# no ssl-server 20 version
```

Terminating a Client Connection with a TCP FIN Message Only

Normally, the SSL Close-Notify alert terminates a client connection without an error. However, some versions of MSIE browsers do not close the connection upon receiving the Close-Notify alert. The browser may attempt to reuse the connection even though it appears to be closed to the CSS. Because the CSS cannot reply to a new request on this connection, the browser may display an error.

The **unclean-shutdown** option for the **ssl-server** command instructs the CSS to send only a TCP FIN message to terminate a client connection. The CSS does not send a Close-Notify alert to close a client connection.

For example, to configure the CSS to send only a TCP FIN message to terminate a client connection, enter:

```
(config-ssl-proxy-list[ssl_list1])# ssl-server 20 unclean-shutdown
```

The **no** version of this command resets the CSS default behavior of sending both a Close-Notify alert and TCP FIN message to close the client connection. For example, enter:

```
(config-ssl-proxy-list[ssl_list1])# no ssl-server 20 unclean-shutdown
```

Specifying Secure URL Rewrite

Client HTTPS connections can become HTTP connections when sent to a back-end server through a virtual SSL server in the SSL proxy list. The back-end server receives data as clear text from the client in the HTTP connection. If the server performs an HTTP 300-series redirect to another HTTP URL, the redirect causes the client to perform an HTTP request even though the client originally had been performing an HTTPS request. Because the client's connection changes to HTTP, the requested data may not be available from the server using a clear text connection.



Note

Do not specify secure URL rewrite as a configuration parameter for the virtual SSL server if you plan to include one or more back-end SSL servers in the SSL proxy list (as described in the [“Specifying the Nagle Algorithm for SSL TCP Connections”](#) section).

You can avoid problems with nonsecure HTTP redirects from the back-end server by configuring one or more URL rewrite rules. Each rewrite rule is associated with a virtual SSL server in the SSL proxy list. URL rewrite rules resolve the problem of a web site redirecting the user to a nonsecure HTTP URL by rewriting the domain from `http://` to `https://`. By using URL rewrite, all client connections to the Web server will be SSL, ensuring the secure delivery of HTTPS content back to the client.

Use the **ssl-server *number* urlrewrite** command to add a URL rewrite rule to the virtual SSL server to avoid nonsecure HTTP 300-series redirects. This command instructs the CSS, through the SSL module, to examine every HTTP header field received from the server for a 300-series redirection response (such as 302 Found or 304 Not Modified). If the CSS finds a 300-series return code, it searches the Location Response-Header field in the HTTP header to determine if the field matches the hostname defined in a URL rewrite rule. If there is a match, the CSS rewrites the Location field to contain an HTTPS location and the SSL port for the response.

For example, to define the following URL rewrite rule, keeping the default of port 443 for the SSL port and port 80 for the clear text port, enter:

```
(config-ssl-proxy-list[ssl_list1])# ssl-server 20 urlrewrite 22  
www.website.com
```

In this case, all HTTP redirects to `http://www.website.com/` are rewritten in the SSL module as `https://www.website.com/` and forwarded to the client.

The CSS supports the use of wildcards in domain hostnames as part of the matching criteria for a URL redirect rule. Include an asterisk (*) wildcard character in the domain name to identify more than one host in a single domain. You can specify a wildcard-only hostname (for example, *), a prefix wildcard (for example, *.mydomain.com), or a suffix wildcard (for example, www.mydomain.*). When using a wildcard-only hostname, the entire domain name is the * (asterisk) character and all HTTP redirects that come through this VIP address from the server are rewritten to HTTPS. In this case, there is no need to have additional URL rewrite rules for the SSL server.



Note

Use care when specifying wildcards to avoid unwanted rewriting of all URL references by the SSL module. Review your redirects and ensure that every URL that matches a specified wildcard rule needs to be rewritten.

The syntax for the **ssl-server *number* urlrewrite** command is:

```
ssl-server number urlrewrite number hostname [sslport port {clearport  
port}]
```

The options and variables are:

- **ssl-server *number*** - The number used to identify the virtual SSL server in the SSL proxy list.

- **urlrewrite *number*** - The number of the URL rewrite rule to be added to the virtual SSL server. Enter a value from 1 to 32 corresponding to the URL rewrite rule. You can add a maximum of 32 URL rewrite rules to each SSL server for handling HTTP to HTTPS redirects.
- **hostname** - The domain name of the URL to be redirected (for example, `www.mydomain.com`). Enter an unquoted text string with a maximum length of 240 characters that corresponds to the domain name of the URL rewrite host. Do not include the directory path as part of the hostname. If you intend to use wildcards in domain names to identify and match on more than one host in a single domain, insert an asterisk (*) wild card character in the domain name.
- **sslport *port*** - (Optional) Specifies the port used for SSL network traffic. Enter a TCP port number that corresponds with an SSL content rule, which uses the specified TCP port number. The SSL module rewrites an HTTP redirect matching the URL redirect rule with the specified SSL port (or default port 443 if no port number is specified). Enter a port value from 1 to 65535. The default value is 443.
- **clearport *port*** - (Optional) Specifies the port used for clear text network traffic. The SSL module matches redirects in the Location Response-Header field with the specified clear text port (or default port 80 if no port number is specified). Enter a port value from 1 to 65535. The default value is 80.

For example, to specify URL rewrite 22 for `www.mydomain.com` using port 444 for SSL traffic and port 81 for clear text, enter:

```
(config-ssl-proxy-list[ssl_list1])# ssl-server 20 urlrewrite 22  
www.mydomain.com sslport 444 clearport 81
```

To remove URL rewrite rule 22, enter:

```
(config-ssl-proxy-list[ssl_list1])# no ssl-server 20 urlrewrite 22
```

For example, for the HTTP URLs `www.sales.acme.com` and `www.services.acme.com`, you could include the wildcard asterisk (*) character as follows to match on the two URLs (keeping the default of port 443 for the SSL port and port 80 for the clear text port):

```
(config-ssl-proxy-list[ssl_list1])# ssl-server 20 urlrewrite 1  
*.acme.com  
(config-ssl-proxy-list[ssl_list1])# ssl-server 20 urlrewrite 2  
*.acme.com
```

Or, you could include the wildcard asterisk (*) character for the HTTP URLs `www.acmesales.com` and `www.acmeservices.com` as follows:

```
(config-ssl-proxy-list[ssl_list1])# ssl-server 20 urlrewrite 1
www.acme*
(config-ssl-proxy-list[ssl_list1])# ssl-server 20 urlrewrite 2
www.acme*
```

To view statistical information on SSL URL rewrite, see [Chapter 7, Displaying SSL Configuration Information and Statistics](#).

Specifying SSL Session Cache Timeout

In SSL, a new session ID is created every time the client and the CSS SSL module go through a full key exchange and establish a new master secret key. Specifying an SSL session cache timeout allows the SSL module to reuse the master key on subsequent connections with the client, which can speed up the SSL negotiation process. You can specify a timeout value to set the total amount of time an SSL session ID remains valid before the SSL module requires the full SSL handshake to establish a new SSL session.

The selection of an SSL session cache timeout value is important when using the **advanced-balance ssl** load-balancing method for a Layer 5 content rule to help fine-tune the SSL session ID that is used to stick the client to the server.

Use the **ssl-server number session-cache seconds** command to configure the SSL module to resume connection with a client using a previously established secret key. Enter an SSL session cache timeout value in seconds, from 0 (SSL session ID reuse disabled) to 72000 (20 hours). The default is 300 seconds (5 minutes). By disabling this option (entering a value of 0), the full SSL handshake occurs for each new connection between the client and the SSL module.



Note

Cisco Systems does not recommend specifying a zero value for the **ssl-server number session-cache seconds** command. A non-zero value ensures that the SSL session ID is reused to improve CSS performance.

For example, to configure the reuse of an SSL session ID with a client using a timeout value of 10 hours, enter:

```
(config-ssl-proxy-list[ssl_list1])# ssl-server 20 session-cache 36000
```

To reset the SSL session reuse timeout to the default of 300 seconds, enter:

```
(config-ssl-proxy-list[ssl_list1])# no ssl-server 20 session-cache
```

Specifying SSL Session Handshake Renegotiation

The SSL session handshake commands send the SSL HelloRequest message to a client to restart SSL handshake negotiation. SSL rehandshake is useful when a connection has been established for a lengthy period of time and you want to ensure security by reestablishing the SSL session.

Use the **ssl-server *number* handshake data *kbytes*** command to specify the maximum amount of data to be exchanged between the CSS and the client, after which the CSS transmits the SSL handshake message and reestablishes the SSL session. By setting the data value, you force the SSL session to renegotiate a new session key after a session has transferred the specified amount of data. Specify an SSL handshake data value in Kbytes, from 0 (handshake disabled) to 512000. The default is 0.

For example, to configure an SSL rehandshake message for the SSL proxy list after a data exchange of 125000 Kbytes is reached with the client, enter:

```
(config-ssl-proxy-list[ssl_list1])# ssl-server 20 handshake data 125000
```

To disable the rehandshake data option, enter:

```
(config-ssl-proxy-list[ssl_list1])# no ssl-server 20 handshake data
```

Use the **ssl-server *number* handshake timeout *seconds*** command to specify a maximum timeout value, after which the CSS transmits the SSL handshake message and reestablishes the SSL session. Setting a timeout value forces the SSL session to renegotiate a new session key after a session has lasted the defined number of seconds. The selection of an SSL rehandshake timeout value is important when using the **advanced-balance ssl** load-balancing method for a Layer 5 content rule to fine-tune the SSL session ID used to stick the client to the server. Specify an SSL handshake timeout value in seconds, from 0 (handshake disabled) to 72000 (20 hours). The default is 0.

For example, to configure an SSL rehandshake message after a timeout value of 10 hours has elapsed, enter:

```
(config-ssl-proxy-list[ssl_list1])# ssl-server 20 handshake timeout 36000
```

To disable the rehandshake timeout option, enter:

```
(config-ssl-proxy-list[ssl_list1])# no ssl-server 20 handshake timeout
```

**Note**

If a connection is stuck using SSL sticky, be aware that the connection loses SSL sticky persistence each time that the CSS performs handshake renegotiation because the SSL session ID regenerates within an existing TCP flow. Because of this situation, the CSS is not aware of the new SSL session ID. When the next TCP connection comes in for this SSL flow, the CSS considers it as a new SSL session and load balances the connections to an SSL service. If there is more than one service and multiple SSL modules, the CSS may send the connection to a different SSL module. The connection will be a new SSL session to that SSL module, which causes the connection to be renegotiated for a second time. After the second renegotiation, the CSS is aware of the SSL session ID and the SSL session sticks to the other SSL module.

In this case, turning on SSL rehandshaking can cause SSL sessions to require additional resources to perform handshake renegotiation. If you are operating in a high traffic environment, this may impact overall SSL performance.

Configuring the Delay Time for SSL Queued Data

SSL on the CSS queues packet data from the server and encrypts it for transmission to the client. SSL empties the data from the queue and encrypts it for transmission to the client when:

- The queue fills to 16,400 bytes (the maximum SSL record size)
- The server sends a TCP FIN packet
- When the delay time on the CSS has passed, even though the queue has less than 16,400 bytes

For efficiency, SSL encrypts data into SSL records with a maximum size of 16,400 bytes. In an attempt to fully queue 16,400 bytes for encryption, SSL delays the emptying of the queue data for encryption.

You can use the **ssl-server *number* ssl-queue-delay *ms*** command to set the amount of time for the CSS virtual SSL server to wait before emptying the queued data for encryption. The default delay is 200 milliseconds. Enter a delay time value in milliseconds from 0 (disabled) to 10000.

Setting the delay value to 0 disables the queuing of data. The virtual SSL server on the CSS encrypts the data as soon as it arrives from the server and then sends the data to the client.

For example, to configure a delay time value of 400 milliseconds, enter:

```
(config-ssl-proxy-list[ssl_list1])# ssl-server 20 ssl-queue-delay 400
```

To reset the delay time to the default of 200 milliseconds, enter:

```
(config-ssl-proxy-list[ssl_list1])# no ssl-server 20 ssl-queue-delay
```

Specifying SSL TCP Client-Side Connection Timeout Values

The TCP connection between the CSS and a client is terminated when the specified time interval elapses. The TCP timeout functions enable you to have more control over the TCP connection between the CSS SSL module and a client.

To configure an SSL proxy list virtual SSL server for termination of a TCP connection with the client, see the following sections:

- [Specifying a TCP SYN Timeout Value \(Client-Side Connection\)](#)
- [Specifying a TCP Inactivity Timeout Value \(Client-Side Connection\)](#)

Specifying a TCP SYN Timeout Value (Client-Side Connection)

The CSS SYN timer counts the delta between the CSS sending the SYN/ACK and the client replying with an ACK as the means to terminate the TCP three-way handshake. Use the `ssl-server number tcp virtual syn-timeout seconds` command to specify a timeout value that the CSS uses to terminate a TCP connection with a client that has not successfully completed the TCP three-way handshake prior to transferring data.

Enter a TCP SYN inactivity timeout value in seconds, from 0 (TCP SYN timeout disabled) to 3600 (1 hour). The default is 30 seconds. When you set the command to 0, the timer becomes inactive and the retransmit timer eventually terminates a broken TCP connection.



Note

The connection timer should always be less than the retransmit termination time for new SSL and TCP connections.

For example, to configure a TCP SYN timeout of 30 minutes (1800 seconds), enter:

```
(config-ssl-proxy-list[ssl_list1])# ssl-server 20 tcp virtual
syn-timeout 1800
```

To reset the TCP SYN timeout to the default of 30 seconds, enter:

```
(config-ssl-proxy-list[ssl_list1])# no ssl-server 20 tcp virtual
syn-timeout
```

Specifying a TCP Inactivity Timeout Value (Client-Side Connection)

The TCP inactivity timeout begins once the CSS receives an ACK from the client to terminate the TCP three-way handshake. The inactivity timer resumes immediately following where the SYN timer stops, with regard to traffic flow. Use the `ssl-server number tcp virtual inactivity-timeout seconds` command to specify a timeout value that the CSS uses to terminate a TCP connection with the client when there is little or no activity occurring on the connection.

Enter a TCP inactivity timeout value in seconds, from 0 (TCP inactivity timeout disabled) to 3600 (1 hour). The default is 240 seconds.

For example, to configure a TCP inactivity time of 30 minutes (1800 seconds), enter:

```
(config-ssl-proxy-list[ssl_list1])# ssl-server 20 tcp virtual  
inactivity-timeout 1800
```

To reset the TCP inactivity timer to the default of 240 seconds, enter:

```
(config-ssl-proxy-list[ssl_list1])# no ssl-server 20 tcp virtual  
inactivity-timeout
```

Specifying SSL TCP Server-Side Connection Timeout Values

The TCP connection between the CSS and a server is terminated when the specified time interval elapses. The TCP timeout functions enable you to have more control over TCP connections between the CSS SSL module and a server.

To configure an SSL proxy list virtual SSL server for termination of a TCP connection with the server, see the following sections:

- [Specifying a TCP SYN Timeout Value \(Server-Side Connection\)](#)
- [Specifying a TCP Inactivity Timeout Value \(Server-Side Connection\)](#)

Specifying a TCP SYN Timeout Value (Server-Side Connection)

The TCP SYN timer counts the delta between the CSS initiating the back-end TCP connection by transmitting a SYN and the server replying with a SYN/ACK. Use the **ssl-server number tcp server syn-timeout seconds** command to specify a timeout value that the CSS uses to end a TCP connection with a server that has not successfully completed the TCP three-way handshake prior to transferring data.

Enter a TCP SYN timeout value in seconds, from 0 (TCP SYN timeout disabled) to 3600 (1 hour). The default is 30 seconds. When you set the command to 0, the timer becomes inactive and the retransmit timer eventually terminates a broken TCP connection.



Note

The connection timer should always be less than the retransmit termination time for new SSL and TCP connections.

For example, to configure a TCP SYN timeout of 30 minutes (1800 seconds), enter:

```
(config-ssl-proxy-list[ssl_list1])# ssl-server 20 tcp server  
syn-timeout 1800
```

To reset the TCP SYN timeout to the default of 30 seconds, enter:

```
(config-ssl-proxy-list[ssl_list1])# no ssl-server 20 tcp server  
syn-timeout
```

Specifying a TCP Inactivity Timeout Value (Server-Side Connection)

The TCP inactivity timeout begins once the CSS receives a SYN/ACK from the server. The inactivity timer resumes immediately following where the SYN timer stops, with regard to traffic flow. Use the **ssl-server number tcp server inactivity-timeout seconds** command to specify a timeout value that the CSS uses to terminate a TCP connection with a server when there is little or no activity occurring on the connection.

Enter a TCP inactivity timeout value in seconds, from 0 (TCP inactivity timeout disabled) to 3600 (1 hour). The default is 240 seconds.

For example, to configure a TCP inactivity time of 30 minutes (1800 seconds), enter:

```
(config-ssl-proxy-list[ssl_list1])# ssl-server 20 tcp server  
inactivity-timeout 1800
```

To reset the TCP inactivity timer to the default of 240 seconds, enter:

```
(config-ssl-proxy-list[ssl_list1])# no ssl-server 20 tcp server  
inactivity-timeout
```

Specifying the Nagle Algorithm for SSL TCP Connections

The TCP Nagle algorithm automatically concatenates a number of small buffer messages transmitted over the TCP connection between a client and the SSL module or between a server and the SSL module. This process increases the throughput of your CSS by decreasing the number of packets sent over each TCP connection. However, the interaction between the Nagle algorithm and the TCP delay acknowledgment may increase latency in your TCP connection. Disable the Nagle algorithm when you observe an unacceptable delay in a TCP connection (clear-text or SSL).

- Use the **ssl-server *number* tcp virtual nagle** command to disable or reenble the Nagle algorithm for the TCP connection between the client and the SSL module. The syntax for this command is:

ssl-server *number* tcp virtual nagle enable|disable

To disable the Nagle algorithm for the TCP connection between the client and the SSL module, enter:

```
(config-ssl-proxy-list[ssl_list1])# ssl-server 20 tcp virtual  
nagle disable
```

To reenble the Nagle algorithm for the TCP connection between the client and the SSL module, enter:

```
(config-ssl-proxy-list[ssl_list1])# ssl-server 20 tcp virtual  
nagle enable
```

- Use the **ssl-server *number* tcp server nagle** command to disable or reenble the Nagle algorithm for the TCP connection between the server and the SSL module. The syntax for this command is:

ssl-server *number* tcp server nagle enable|disable

To disable the Nagle algorithm for the TCP connection between the server and the SSL module, enter:

```
(config-ssl-proxy-list[ssl_list1])# ssl-server 20 tcp server nagle disable
```

To reenable the Nagle algorithm for the TCP connection between the server and the SSL module, enter:

```
(config-ssl-proxy-list[ssl_list1])# ssl-server 20 tcp server nagle enable
```

Specifying the TCP Buffering for SSL TCP Connections

If the network is slow and congested, you can increase the buffer size that the CSS buffers for a given TCP connection before shutting down the TCP window to 0. Increasing the buffer size decreases latency on slow connections to a client but increases buffering in the CSS. Use this feature with caution, because if there are many slow clients, the CSS memory may run low.

Use the **ssl-server number tcp buffer-share** command to set the TCP buffering from the client or server on a given connection. The syntax for this command is:

```
ssl-server number tcp buffer-share rx number1/tx number2
```

- To set the amount of data in bytes that a given connection can buffer from the client traffic, use the **rx number1** keyword and variable. By default, the buffer size is 32768. The buffer size can range from 16400 to 262144. For example, to set the value to 65536, enter:

```
(config-ssl-proxy-list[ssl_list1])# ssl-server 20 tcp buffer-share rx 65536
```

To reset the reset the buffer size to the default of 32768, enter:

```
(config-ssl-proxy-list[ssl_list1])# no ssl-server 20 tcp buffer-share rx
```

- To set the amount of data in bytes that a given connection can buffer from the server to the client, use the **tx number2** keyword and variable. By default, the buffer size is 65536. The buffer size can range from 16400 to 262144. For example, to set the value to 131072, enter:

```
(config-ssl-proxy-list[ssl_list1])# ssl-server 20 tcp buffer-share  
tx 131072
```

To reset the reset the buffer size to the default of 65536, enter:

```
(config-ssl-proxy-list[ssl_list1])# no ssl-server 20 tcp  
buffer-share tx
```

Activating and Suspending an SSL Proxy List

Before you can activate an SSL proxy list, ensure that you have created at least one virtual or back-end SSL server in the list (see the “[Configuring Virtual SSL Servers for an SSL Proxy List](#)” section or the “[Specifying the Nagle Algorithm for SSL TCP Connections](#)” section earlier in this chapter).

The CSS checks the SSL proxy list to verify that all of the necessary components are configured, including verification of the certificate and key pair against each other. If the verification fails, the certificate name is not accepted and the CSS logs the error message `Certificate and key pair do not match` and does not activate the SSL proxy list. You must either remove the configured key pair or configure an appropriate certificate.

Use the **active** command to activate the new or modified SSL proxy list. For example, enter:

```
(config-ssl-proxy-list[ssl_list1])# active
```

After you activate an SSL proxy list, you can add it to a service. See the “[Configuring a Service for SSL Termination](#)” section later in this chapter.



Note

No modifications to an SSL proxy list are permitted on an active list. Suspend the list prior to making changes, and then reactivate the SSL proxy list once the changes are complete. Once you have modified the SSL proxy list, suspend the SSL service, reactivate the SSL proxy list, and then reactivate the SSL service.

To view the virtual or back-end SSL servers in a list, use the **show ssl-proxy-list** (see [Chapter 7, Displaying SSL Configuration Information and Statistics](#)).

Use the **suspend** command to suspend an active SSL proxy list.

To suspend an active SSL proxy list, enter:

```
(config-ssl-proxy-list[ssl_list1])# suspend
```

Configuring a Service for SSL Termination

An SSL proxy list may belong to multiple SSL services (one SSL proxy list per service), and an SSL service may belong to multiple content rules. You can apply the services to content rules that allow the CSS to direct SSL requests for content.



Note

The CSS supports one active SSL service for each SSL module in the CSS (one SSL service per slot). You can configure more than one SSL service for a slot but only a single SSL service can be active at a time.

The following sections cover:

- [Creating an SSL Service](#)
- [Specifying the SSL Acceleration Service Type](#)
- [Adding an SSL Proxy List to an SSL Termination Service](#)
- [Specifying the SSL Module Slot](#)
- [Disabling Keepalive Messages for the SSL Module](#)
- [Specifying the SSL Session ID Cache Size](#)
- [Activating the SSL Service](#)
- [Suspending the SSL Service](#)

Creating an SSL Service

When creating a service for use with an SSL module, you must identify it as an SSL service for the CSS to recognize it. For additional details on creating a service, refer to the *Cisco Content Services Switch Content Load-Balancing Configuration Guide*.

Enter the SSL service name, from 1 to 31 characters.

To create service `ssl_serv1`, enter:

```
(config)# service ssl_serv1  
Create service <ssl_serv1>, [y/n]: y
```

The CSS transitions into the newly created service mode.

```
(config-service[ssl_serv1])#
```

Specifying the SSL Acceleration Service Type

After you create the SSL service and the CSS enters into service mode, you must specify **ssl-accel** as the service type to:

- Configure the service as an SSL acceleration service.
- Add the SSL proxy list to an SSL service.

Use the **type** command to specify the SSL acceleration service type. For details on specifying an SSL service type, refer to the *Cisco Content Services Switch Content Load-Balancing Configuration Guide*.

To specify the SSL acceleration service type, enter:

```
(config-service[ssl_serv1])# type ssl-accel
```

Adding an SSL Proxy List to an SSL Termination Service

After you configure a virtual SSL server on an SSL proxy list for an SSL module, add the active list to an SSL service. The active list explains how the CSS processes SSL requests for content through the specific SSL module. To include an SSL proxy list as part of an SSL service, use the **add ssl-proxy-list** command in service mode. Enter the name of the previously created SSL proxy list (see the “[Creating an SSL Proxy List](#)” section in this chapter) that you want to add to the service.

To add SSL proxy list *ssl_list1* to service *ssl_serv1*, enter:

```
(config-service[ssl_serv1])# add ssl-proxy-list ssl_list1
```

To remove the SSL proxy list from the service, enter:

```
(config-service[ssl_serv1])# remove ssl-proxy-list ssl_list1
```

Specifying the SSL Module Slot

The CSS 11501 supports a single integrated SSL module. The CSS 11503 and CSS 11506 support multiple SSL modules; a maximum of two in a CSS 11503 and a maximum of four in a CSS 11506. The SSL service requires the SSL module slot number to correlate the SSL proxy list and virtual SSL server(s) to a specific SSL module. Use the **slot** command to specify the slot in the CSS chassis where the SSL module is located.

The valid slot entries are:

- CSS 11501 - 2
- CSS 11503 - 2 and 3
- CSS 11506 - 2 to 6

Slot 1 is reserved for the SCM.



Note

The CSS supports one active SSL service for each SSL module in the CSS (one SSL service per slot). You can configure more than one SSL service for a slot but only a single SSL service can be active at a time.

For example, to identify an SSL module in slot 3 of the CSS chassis, enter:

```
(config-service[ssl_serv1])# slot 3
```

Disabling Keepalive Messages for the SSL Module

The SSL module is an integrated device within the CSS chassis and, therefore, does not require the use of keepalive messages for the service. Use the **keepalive type none** command to instruct the CSS not to send keepalive messages to a service. For details on specifying a keepalive type, refer to the *Cisco Content Services Switch Content Load-Balancing Configuration Guide*.

To disable sending keepalive messages for an SSL service, enter:

```
(config-service[ssl_serv1])# keepalive type none
```

Specifying the SSL Session ID Cache Size

The cache size is the maximum number of SSL session IDs that can be stored in a dedicated session cache on an SSL module. By default, the SSL session cache can hold 10000 sessions. If necessary for your SSL service, you can increase the SSL session cache size to 100000. Use the **session-cache-size** command to reconfigure the size of the SSL session ID cache for a service. Valid entries are 0 (SSL session cache disabled) to 100000 sessions.



Note

Cisco Systems does not recommend specifying a zero value for the **session-cache-size** command to ensure that the SSL session ID is reused. Specifying an SSL session cache and cache timeout allows the reuse of the master key on subsequent connections between the client and the CSS SSL module, which can speed up the SSL negotiation process and improve CSS performance.

The back-end session ID cache is 4096 entries and is not configurable.

If you specify 0 as the SSL session cache size, the SSL module associated with the SSL service does not cache any SSL session IDs. If you choose to disable the SSL session cache, ensure the following parameters are properly configured to disable the use of SSL session ID:

- Set the **ssl-server number session-cache timeout** setting in the SSL proxy list to 0 (disabled) for a virtual SSL server.
- Disable the **advanced-balance ssl** command in the content rule to disable SSL sticky.

For example, to specify an SSL session cache size of 20000 sessions, enter:

```
(config-service[ssl_serv1])# session-cache-size 20000
```

To reset the SSL session cache size to the default of 10000 sessions, enter:

```
(config-service[ssl_serv1])# no session-cache-size
```

Activating the SSL Service

Once you configure an SSL proxy list service, use the **active** command to activate the service. Activating a service puts it into the resource pool for load-balancing SSL content requests between the client and the server.

Before activating an SSL service:

- For a virtual SSL server, you must add an SSL proxy list to an **ssl-accel** type service before you can activate the service. If no list is configured when you enter the **active** command, the CSS logs the following error message and does not activate the service.

```
Must add at least one ssl-proxy-list to an ssl-accel type service
```

- For a back-end SSL server, you must add an SSL proxy list to an **ssl-accel-backend** type service before you can activate the service. If no list is configured when you enter the **active** command, the CSS logs the following error message and does not activate the service.

```
Must add at least one ssl-proxy-list to an ssl-accel type service
```

- The SSL proxy list added to the service must be active before you can activate the service. If the list is suspended, the CSS logs the following error message and does not activate the service.

```
No ssl-lists on service, service not activated
```

Once the service is ready to activate, the CSS initiates the transfer of appropriate SSL configuration data for each SSL proxy list to a specific SSL module and activates the service. If there is an error in transfer, the CSS logs the appropriate error and does not activate the service.

No modifications may be made to an active SSL proxy list. If modifications are necessary, first suspend the ssl service to make changes to the SSL proxy list entries.

To activate service *ssl_serv1*, enter:

```
(config-service[ssl_serv1])# active
```

Suspending the SSL Service

To suspend an SSL service and remove it from the pool for future load-balancing SSL content requests, use the **suspend** command. Suspending an SSL service does not affect existing content flows, but it prevents additional connections from accessing the service for its content.

You must suspend a service prior to modifying an SSL proxy list.

To suspend service *ssl_serv1*, enter:

```
(config-service[ssl_serv1])# suspend
```

Configuring a Content Rule for SSL Termination

For the CSS to direct SSL requests for content, apply the virtual services to content rules. No network traffic is sent to an SSL module until you activate an SSL content rule to define where the content physically resides, where to direct the request for content (which SSL service), and which load-balancing method to use.

For a virtual SSL server content rule, ensure that the VIP address and port number configured for the rule match the VIP address and port number for the server entry in the SSL proxy list.

When you activate a content rule with a configured SSL service, the CSS verifies that there is a VIP address and port match. If a match is not found, the CSS logs the following error message and does not activate the content rule.

```
Not all content VIP:Port combinations are configured in an
ssl-proxy-list for sslAccel type of service
```

Verify the configured VIP addresses used in the content rule and SSL proxy list, and modify as necessary.

When a CSS uses two or more SSL modules, Cisco Systems recommends that you use stickiness based on SSL version 3 session ID for a Layer 5 content rule. For a virtual SSL server rule, specify the following:

- Enable the content rule to be sticky based on SSL using the **advanced-balance ssl** command.
- Specify the SSL application type using the **application ssl** command.

The Layer 5 SSL sticky content rule ensures SSL session ID reuse to eliminate the rehandshake process (which speeds up the SSL negotiation process) and to increase overall performance.

**Note**

If the 32K sticky table becomes full (which means that 32K simultaneous users are on the site) the table wraps and the first users in the table become “unstuck.” This may be due to a combination of number of flows and the duration of the sticky period, which can quickly use up the available space in the sticky table. This problem can typically occur in a CSS that contains multiple SSL modules. An SCM with 288M memory module can support a 128K sticky table.

**Note**

If you specify the **sticky-inact-timeout** command for a Layer 5 content rule using SSL sticky, the SSL sessions continue even if the sticky table is full. However, the CSS does not maintain stickiness on the new sessions.



Configuring Back-End SSL

This chapter describes the steps required to configure back-end SSL on a CSS. It contains the following major sections:

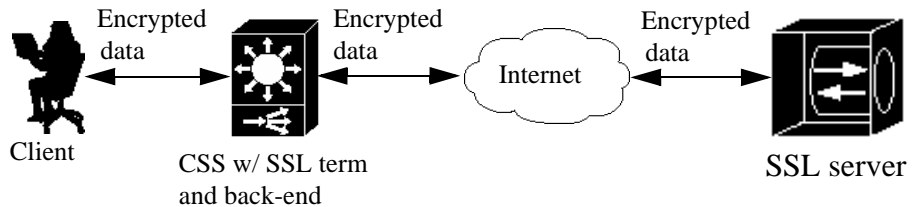
- [Overview of Back-End SSL](#)
- [Creating an SSL Proxy List](#)
- [Adding a Description to an SSL Proxy List](#)
- [Configuring Back-End SSL Servers in an SSL Proxy List](#)
- [Activating and Suspending an SSL Proxy List](#)
- [Configuring a Service for Back-End SSL](#)
- [Configuring a Content Rule for Back-End SSL](#)

Overview of Back-End SSL

Back-end SSL allows a CSS to initiate a connection with an SSL server. When used with SSL termination, back-end SSL provides a secure end-to-end connection between a client and an SSL server.

Figure 5-1 illustrates back-end SSL with SSL termination.

Figure 5-1 Back-End SSL with SSL Termination



An SSL proxy list determines the flow of SSL information among the SSL module, the client, and the server. An SSL proxy list comprises one or more back-end SSL servers (related by index entry). The back-end SSL server entry initiates the connection to an SSL server. You can define a maximum of 256 virtual or back-end SSL servers for a single SSL proxy list.

After you create and configure the entries in a proxy list, you must activate the list, and then add the SSL proxy list to a service to initiate the transfer of SSL configuration data to the SSL module. When you activate the service, the CSS transfers the data to the module. Then add each SSL service to an SSL content rule.

Creating an SSL Proxy List

An SSL proxy list is a group of related back-end SSL servers that are associated with an SSL service. To create an SSL proxy list, use the **ssl-proxy-list** command.

You can access the `ssl-proxy-list` configuration mode from most configuration modes except for `ACL`, `boot`, `group`, `rmon`, or `owner` configuration modes. You can also use this command from the `ssl-proxy-list` configuration mode to access another SSL proxy list. Enter the SSL proxy list name as an unquoted text string from 1 to 31 characters.

For example, to create the SSL proxy list, `ssl_list1`, enter:

```
(config)# ssl-proxy-list ssl_list1
Create ssl-list <ssl_list1>, [y/n]: y
```

Once you create an SSL proxy list, the CLI enters into the `ssl-proxy-list` configuration mode.

```
(config-ssl-proxy-list[ssl_list1])#
```

To delete an existing SSL proxy list, enter:

```
(config)# no ssl-proxy-list ssl_list1  
Delete ssl-list <ssl_list1>, [y/n]: y
```

**Note**

You cannot delete an SSL proxy list if an SSL service is in use and contains the active SSL proxy list. You must first suspend the SSL service to delete a specific SSL proxy list.

Adding a Description to an SSL Proxy List

To specify a description for an SSL proxy list, use the **description** command. Enter the description as a quoted text string with a maximum of 64 characters, including spaces.

For example, to add a description to the `ssl_list1` SSL proxy list, enter:

```
(config-ssl-proxy-list[ssl_list1])# description "This is the SSL list  
for www.brandnewproducts.com"
```

To remove the description from a specific SSL proxy list, enter:

```
(config-ssl-proxy-list[ssl_list1])# no description
```

Configuring Back-End SSL Servers in an SSL Proxy List

This section discusses creating one or more back-end SSL servers for an SSL proxy list. Use the **backend-server** command to define an index entry in the SSL proxy list that you then use to configure specific SSL parameters associated with the SSL proxy list. An SSL module in the CSS uses the SSL proxy list to initiate a connection to a back-end SSL server. You must define a back-end server index number before configuring SSL proxy list parameters. You can define a maximum of 256 back-end SSL servers for a single SSL proxy list.

**Note**

You cannot modify the back-end SSL servers in an active SSL proxy list. You must first suspend the SSL proxy list to make modifications to any of the back-end-servers in a specific SSL proxy list. Once you have modified the SSL proxy list, suspend the SSL service, activate the SSL proxy list, and then activate the SSL service.

To configure a back-end server for use by the SSL module, you must create and configure a back-end server entry in an SSL proxy list. Configure an IP address that corresponds to the address of the service and the server IP address. Then activate the SSL proxy list.

After you configure and activate the SSL proxy list, add the list to a back-end SSL service; assign a service type of **ssl-accel-backend**. When you activate the service, the CSS sends the configuration data to the SSL module.

The following sections describe:

- [Creating a Back-End SSL Server in an SSL Proxy List](#)
- [Configuring the VIP Address for an SSL Back-End Server](#)
- [Configuring the Virtual Port](#)
- [Configuring the Server IP Address](#)
- [Configuring the Server Port](#)
- [Configuring SSL Version](#)
- [Configuring the Available Cipher Suites](#)
- [Configuring SSL Session Cache Timeout](#)
- [Configuring SSL Session Handshake Renegotiation](#)
- [Configuring TCP Virtual Client Connections Timeout Values](#)
- [Configuring TCP Server-Side Connection Timeout Values on the SSL Module](#)
- [Specifying the Nagle Algorithm for SSL TCP Connections](#)
- [Specifying the TCP buffering for SSL TCP Connections](#)

Creating a Back-End SSL Server in an SSL Proxy List

You must create a back-end SSL server before you can configure back-end SSL proxy-list parameters. To create a back-end server in the SSL proxy list, use the **backend-server** *number* command. This command assigns it a number (index entry) in the SSL proxy list that you use to configure specific SSL parameters associated with the back-end SSL server (for example, VIP address, certificate name, and key pair). Enter a value from 1 to 256.

For example, to create back-end server 1 in the proxy list, enter:

```
(config-ssl-proxy-list[ssl_list3])# backend-server 1
```

To remove back-end server 1 from the SSL proxy list, enter:

```
(config-ssl-proxy-list[ssl_list3])# no backend-server 1
```

Configuring a Back-End SSL Server Type

By default, a back-end SSL server has a type of **backend-ssl** that allows a CSS to:

- Receive encrypted data from a client
- Decrypt the data for load balancing
- Re-encrypt the data and send it to an SSL server over an SSL connection

If you configure an SSL initiation server but want to reconfigure it as a back-end SSL server in the same proxy list, use the **backend-server** *number* **type backend-ssl** command.

For example, to reconfigure SSL initiation server 1 as a back-end SSL server in SSL proxy list `ssl_list3`, enter:

```
(config-ssl-proxy-list[ssl_list3])# backend-server 1 type backend-ssl
```

For information about SSL initiation, see [Chapter 5, Configuring SSL Initiation](#).

Configuring the VIP Address for an SSL Back-End Server

To configure the VIP address for the back-end server, use the **backend-server number ip address** command. The VIP address corresponds to the address of the service.

For example, to configure the VIP address 192.168.2.3 for back-end server 1, enter:

```
(config-ssl-proxy-list[ssl_list1])# backend-server 1 ip address 192.168.2.3
```

To remove the VIP address from the back-end server, enter:

```
(config-ssl-proxy-list[ssl_list1])# no backend-server 1 ip address
```



Note

If you have not configured the VIP address when you issue the **active** command, the following error message appears and the CSS does not activate the list.

```
SSL-server/Backend-server must have valid IP Address
```

Configuring the Virtual Port

By default, the virtual port for the back-end server is port 80. The virtual port directs the clear text data traffic from the SSL module to the CSS. To configure a different virtual port for the SSL back-end server, use the **backend-server number port** command. Enter a port number from 1 to 65535.



Note

If you configure the **backend-server number ip address** and **server-ip** commands with the same address, configure the **backend-server number port** and **server-port** commands with different port numbers.

For example, to configure a port number of 1200, enter:

```
(config-ssl-proxy-list[ssl_list1])# backend-server 1 port 1200
```

To reset the port to the default value of 80, enter:

```
(config-ssl-proxy-list[ssl_list1])# no backend-server 1 port
```

Configuring the Server IP Address

The server IP address is the IP address for the back-end SSL server. To configure the server IP address for the back-end server, use the **backend-server number server-ip** command.

For example, to configure the server IP address 192.168.2.3, enter:

```
(config-ssl-proxy-list[ssl_list1])# backend-server 1 server-ip 192.168.2.3
```

To remove the server IP address from the back-end server, enter:

```
(config-ssl-proxy-list[ssl_list1])# no backend-server 1 server-ip
```

**Note**

If you do not configure the server IP address when you issue the **active** command, the following error message appears and the CSS does not activate the list.

```
SSL-server/Backend-server must have valid IP address
```

Configuring the Server Port

By default the server port for the back-end SSL server is port 443. To configure a different server port for the SSL back-end server, use the **backend-server number server-port** command. Enter a port number from 1 to 65535.

**Note**

If you configure the **backend-server number ip address** and **server-ip** commands with the same address, configure the **backend-server number port** and **server-port** commands with different port numbers.

For example, to configure the server port number 155, enter:

```
(config-ssl-proxy-list[ssl_list1])# backend-server 1 server-port 155
```

To reset the port to the default value of 443, enter:

```
(config-ssl-proxy-list[ssl_list1])# no backend-server 1 server-port
```

Configuring SSL Version

For a back-end server, the SSL module initiates the SSL connection. The version in the ClientHello message sent to the server indicates the highest supported version.

By default, the SSL version is SSL version 3 and TLS version 1. The SSL module sends a ClientHello that has an SSL version 3 header with the ClientHello message set to TLS version 1.

Use the **backend-server *number* version** command to specify which version of SSL the back-end server supports:

- **ssl3** - SSL version 3.
- **tls1** - TLS version 1.
- **ssl-tls** - SSL version 3 and TLS version 1. The SSL module sends a ClientHello that has an SSL version 3 header with the ClientHello message set to TLS version 1.

For example, to configure the SSL version 3, enter:

```
(config-ssl-proxy-list[ssl_list1])# backend-server 1 version ssl3
```

To reset the default SSL version, enter:

```
(config-ssl-proxy-list[ssl_list1])# no backend-server 1 version
```

Configuring the Available Cipher Suites

To configure one or more specific cipher suites to be used by the back-end server, use the **backend-server *number* cipher** command. By default, all supported hardware accelerated cipher suites are enabled.

[Table 4-1](#) earlier in this chapter lists all supported cipher suites for the SSL module and the corresponding cipher suite value. These values match those defined for SSL version 3.0 and TLS version 1.0. The table also lists those Cipher suites that are exportable in any version of the software.

If you use the default setting or select the **all-cipher-suite** option, the CSS sends the suites in the same order as they appear in [Table 4-1](#), starting with `rsa-with-rc4-128-md5`.

**Note**

The **all-cipher-suites** option reenables all cipher suites for the back-end server. This option works only when you do not configure specifically-defined ciphers. To return to using the **all-cipher-suites** option, you must remove all specifically-defined ciphers.

For example, to configure a cipher of `rsa-with-rc4-128-md5`, enter:

```
(config-ssl-proxy-list[ssl_list1])# backend-server 1 cipher  
rsa-with-rc4-128-md5
```

When negotiating which cipher suite to use, the SSL module sends the ciphers in weighted order to the server with the highest weighted cipher first in the list.

By default, all configured cipher suites have a weight of 1. Optionally, you can assign a priority weight to the cipher suite, with 10 being the highest.

For example, to set a weight of 10 to a cipher suite, enter:

```
(config-ssl-proxy-list[ssl_list1])# backend-server 1 cipher  
rsa-with-rc4-128-md5 weight 10
```

To remove one or more of the configured cipher suites for the back-end server, enter:

```
(config-ssl-proxy-list[ssl_list1])# no backend-server 1 cipher  
rsa-with-rc4-128-md5
```

Configuring SSL Session Cache Timeout

In SSL, every time a client and server go through a full key exchange and establish a new master secret key, a new session is created. Enabling a session cache timeout allows the reuse of the master key on subsequent connections by the client. When you disable the cache timeout, the full SSL handshake must occur on each new connection to the SSL module (the virtual client). Use the **backend-server *number* session-cache** command to configure the SSL module to resume connection with a back-end SSL server using a previously established secret key.

By default, the cache timeout is enabled with a timeout of 300 seconds (5 minutes). The timeout value can range from 0 to 72000 (0 seconds to 20 hours). A timeout value of 0 disables the session cache reuse.

For example, to configure the SSL session cache timeout of 500 seconds, enter:

```
(config-ssl-proxy-list[ssl_list1])# backend-server 1 session-cache 500
```

To reset the session cache ID reuse to the default of enabled with a timeout of 300 seconds, enter:

```
(config-ssl-proxy-list[ssl_list1])# no backend-server 1 session-cache
```

To disable session cache ID reuse, enter a timeout value of 0 seconds:

```
(config-ssl-proxy-list[ssl_list1])# backend-server 1 session-cache 0
```

Configuring SSL Session Handshake Renegotiation

The SSL session handshake commands send the SSL HelloRequest message to a client to restart SSL handshake negotiation. SSL rehandshake is useful when a connection has been established for a lengthy period of time and you want to ensure security by reestablishing the SSL session between the CSS and the back-end SSL server.

Use the **backend-server number handshake data kbytes** command to force an SSL rehandshake after the exchange of a certain amount of data between the CSS and the back-end SSL server, after which the CSS transmits the SSL handshake message and reestablishes the SSL session.

By default, the SSL rehandshake is disabled (set to 0) for a back-end SSL server after the exchange of data. The data value is in kilobytes and is from 0 to 512000 kilobytes.

For example, to configure the SSL session rehandshake data value of 500 Kbytes, enter:

```
(config-ssl-proxy-list[ssl_list1])# backend-server 1 handshake data 500
```

To reset the rehandshake data value to 0, disable the rehandshake after the exchange of data. For example, enter:

```
(config-ssl-proxy-list[ssl_list1])# no backend-server 1 handshake data
```

Use the **backend-server number handshake timeout seconds** command to specify a maximum timeout value, after which the CSS transmits the SSL handshake message and reestablishes the SSL session. Setting a timeout value

forces the SSL session to renegotiate a new session key after a session has lasted the defined number of seconds. The selection of an SSL rehandshake timeout value is important when using the **advanced-balance** ssl load-balancing method for a Layer 5 content rule to fine-tune the SSL session ID used to stick the client to the server.

By default, the SSL rehandshake timeout is disabled (set to 0) for the back-end SSL server. The timeout value is from 0 to 72000 (0 seconds to 20 hours).

For example, to configure a 30-second timeout of an SSL session rehandshake, enter:

```
(config-ssl-proxy-list[ssl_list1])# back-end-server 1 handshake  
timeout 30
```

To reset the timeout to 0, disable the rehandshake timeout period for the back-end server by entering:

```
(config-ssl-proxy-list[ssl_list1])# no backend-server 1 handshake  
timeout
```

Configuring TCP Virtual Client Connections Timeout Values

The TCP connection between the client and the SSL module is terminated when the specified time interval elapses. The TCP timeout functions enable you to have more control over the TCP connection between the client and the SSL module.

To configure the TCP connection with the client, see the following sections:

- [Specifying a TCP SYN Timeout Value for the Virtual Client Connection](#)
- [Specifying a TCP Inactivity Timeout for a Virtual Client Connection](#)

Specifying a TCP SYN Timeout Value for the Virtual Client Connection

The CSS SYN timer counts the delta between the CSS sending the SYN/ACK and the client replying with an ACK as the means to terminate the TCP three-way handshake. Use the **backend-server number tcp virtual syn-timeout seconds** command to specify a timeout value that the CSS uses to terminate a TCP connection with a client and the SSL module that has not successfully completed the TCP three-way handshake prior to transferring data.

Enter a TCP SYN inactivity timeout value in seconds, from 0 (TCP SYN timeout disabled) to 3600 (1 hour). The default is 30 seconds. When you set the command to 0, the timer becomes inactive and the retransmit timer eventually terminates a broken TCP connection.

**Note**

The connection timer should always be less than the retransmit termination time for new SSL and TCP connections.

To configure the TCP SYN timeout of 100 seconds, enter:

```
(config-ssl-proxy-list[ssl_list1])# backend-server 1 tcp virtual
syn-timeout 100
```

To disable the timeout, set the value to 0:

```
(config-ssl-proxy-list[ssl_list1])# backend-server 1 tcp virtual
syn-timeout 0
```

To reset the timeout to the default value of 30 seconds, enter:

```
(config-ssl-proxy-list[ssl_list1])# no backend-server 1 tcp virtual
syn-timeout
```

Specifying a TCP Inactivity Timeout for a Virtual Client Connection

The TCP inactivity timeout begins once the CSS receives an ACK from the client to terminate the TCP three-way handshake. The inactivity timer resumes immediately following where the SYN timer stops, with regard to traffic flow. Use the **backend-server *number* tcp virtual inactivity-timeout *seconds*** command to specify a timeout value that the CSS uses to terminate a TCP connection with the client and the SSL module when there is little or no activity occurring on the connection.

Enter a TCP inactivity timeout value in seconds, from 0 (TCP inactivity timeout disabled) to 3600 (1 hour). The default is 240 seconds.

Based on the default parameters for retransmission, the timer value should be larger than 60 seconds (1 minute).

For example, to configure the TCP inactivity timeout period of 100 seconds for the virtual client connection, enter:

```
(config-ssl-proxy-list[ssl_list1])# backend-server 1 tcp virtual  
inactivity-timeout 100
```

To disable the timeout, set the value to 0:

```
(config-ssl-proxy-list[ssl_list1])# backend-server 1 tcp virtual  
inactivity-timeout 0
```

To reset the timeout to the default value of 240 seconds, enter:

```
(config-ssl-proxy-list[ssl_list1])# no backend-server 1 tcp virtual  
inactivity-timeout
```

Configuring TCP Server-Side Connection Timeout Values on the SSL Module

The TCP connection between the SSL module and a server is terminated when the specified time interval elapses. The TCP timeout functions enable you to have more control over TCP connections between the CSS SSL module and a server.

To configure the timeout values of a TCP connection with the server, see the following sections:

- [Specifying a TCP SYN Timeout Value for a Server-Side Connection](#)
- [Specifying a TCP Inactivity Timeout for a Server-Side Connection](#)

Specifying a TCP SYN Timeout Value for a Server-Side Connection

The TCP SYN timer counts the delta between the CSS initiating the back-end TCP connection by transmitting a SYN and the server replying with a SYN/ACK. Use the **backend-server number tcp server syn-timeout seconds** command to specify a timeout value that the CSS uses to end a TCP connection with a server that has not successfully completed the TCP three-way handshake prior to transferring data.

Enter a TCP SYN timeout value in seconds, from 0 (TCP SYN timeout disabled) to 3600 (1 hour). The default is 30 seconds. When you set the command to 0, the timer becomes inactive and the retransmit timer eventually terminates a broken TCP connection.

**Note**

The connection timer should always be less than the retransmit termination time for new SSL and TCP connections.

For example, to configure the TCP SYN timeout of 100 seconds for the server-side connection, enter:

```
(config-ssl-proxy-list[ssl_list1])# backend-server 1 tcp server  
syn-timeout 100
```

To disable the timeout, set the value to 0:

```
(config-ssl-proxy-list[ssl_list1])# backend-server 1 tcp server  
syn-timeout 0
```

To reset the timeout to the default value of 30 seconds, enter:

```
(config-ssl-proxy-list[ssl_list1])# no backend-server 1 tcp server  
syn-timeout
```

Specifying a TCP Inactivity Timeout for a Server-Side Connection

The TCP inactivity timeout begins once the CSS receives a SYN/ACK from the server. The inactivity timer resumes immediately following where the SYN timer stops, with regard to traffic flow. Use the **backend-server number tcp server inactivity-timeout seconds** command to specify a timeout value that the CSS uses to terminate a TCP connection with a server when there is little or no activity occurring on the connection.

Enter a TCP inactivity timeout value in seconds, from 0 (TCP inactivity timeout disabled) to 3600 (1 hour). The default is 240 seconds.

For example, to configure the TCP inactivity timeout period of 100 seconds for the server-side connection, enter:

```
(config-ssl-proxy-list[ssl_list1])# backend-server 1 tcp server  
inactivity-timeout 100
```

To disable the timeout, set the value to 0:

```
(config-ssl-proxy-list[ssl_list1])# backend-server 1 tcp server  
inactivity-timeout 0
```

To reset the timeout to the default value of 240 seconds, enter:

```
(config-ssl-proxy-list[ssl_list1])# no backend-server 1 tcp server  
inactivity-timeout
```

Specifying the Nagle Algorithm for SSL TCP Connections

The TCP Nagle algorithm automatically concatenates a number of small buffer messages transmitted over the TCP connection between a client and the SSL module or between a back-end server and the SSL module. This process increases the throughput of your CSS by decreasing the number of packets sent over each TCP connection. However, the interaction between the Nagle algorithm and the TCP delay acknowledgment may increase latency in your TCP connection. Disable the Nagle algorithm when you observe an unacceptable delay in a TCP connection (clear-text or SSL).

Use the **backend-server *number* tcp virtual nagle** command to disable or reenble the Nagle algorithm for the TCP connection between the client and the SSL module. The syntax for this command is:

backend-server *number* tcp virtual nagle enable|disable

To disable the Nagle algorithm for the TCP connection between the client and the SSL module, enter:

```
(config-ssl-proxy-list[ssl_list1])# backend-server 1 tcp virtual nagle  
disable
```

To reenble the Nagle algorithm for the TCP connection between the client and the SSL module, enter:

```
(config-ssl-proxy-list[ssl_list1])# backend-server 1 tcp virtual nagle enable
```

Use the **backend-server *number* tcp server nagle** command to disable or reenble the Nagle algorithm for the TCP connection between the server and the SSL module. The syntax for this command is:

backend-server *number* tcp server nagle enable|disable

To disable the Nagle algorithm for the TCP connection between the server and the SSL module, enter:

```
(config-ssl-proxy-list[ssl_list1])# backend-server 1 tcp server nagle
disable
```

To reenable the Nagle algorithm for the TCP connection between the server and the SSL module, enter:

```
(config-ssl-proxy-list[ssl_list1])# backend-server 1 tcp server nagle
enable
```

Specifying the TCP buffering for SSL TCP Connections

If the network is slow and congested, you can increase the buffer size that the CSS buffers for a given TCP connection before shutting down the TCP window to 0. Use the **backend-server number tcp buffer-share** command to set the TCP buffering from the client or server on a given connection. The syntax for this command is:

backend-server number tcp buffer-share rx number1 tx number2

To set the amount of data in bytes that a given connection can buffer from the client traffic, use the **rx number1** keyword and variable. By default, the buffer size is 32768. The buffer size can range from 16400 to 262144. For example, to set the value to 65536, enter:

```
(config-ssl-proxy-list[ssl_list1])# backend-server 20 tcp buffer-share
rx 65536
```

To reset the reset the buffer size to the default of 32768, enter:

```
(config-ssl-proxy-list[ssl_list1])# no backend-server 20 tcp
buffer-share rx
```

To set the amount of data in bytes that a given connection can buffer from the server to the client, use the **tx number2** keyword and variable. By default, the buffer size is 65536. The buffer size can range from 16400 to 262144. For example, to set the value to 131072, enter:

```
(config-ssl-proxy-list[ssl_list1])# backend-server 20 tcp buffer-share
tx 131072
```

To reset the reset the buffer size to the default of 65536, enter:

```
(config-ssl-proxy-list[ssl_list1])# no backend-server 20 tcp
buffer-share tx
```

Activating and Suspending an SSL Proxy List

Before you can activate an SSL proxy list, ensure that you have created at least one virtual or back-end SSL server in the list (see the “[Configuring Virtual SSL Servers for an SSL Proxy List](#)” section or the “[Specifying the Nagle Algorithm for SSL TCP Connections](#)” section earlier in this chapter).

The CSS checks the SSL proxy list to verify that all of the necessary components are configured, including verification of the certificate and key pair against each other. If the verification fails, the certificate name is not accepted and the CSS logs the error message `Certificate and key pair do not match` and does not activate the SSL proxy list. You must either remove the configured key pair or configure an appropriate certificate.

Use the **active** command to activate the new or modified SSL proxy list. For example, enter:

```
(config-ssl-proxy-list[ssl_list1])# active
```

After you activate an SSL proxy list, you can add it to a service. See the “[Configuring a Service for Back-End SSL](#)” section later in this chapter.



Note

No modifications to an SSL proxy list are permitted on an active list. Suspend the list prior to making changes, and then reactivate the SSL proxy list once the changes are complete. Once you have modified the SSL proxy list, suspend the SSL service, reactivate the SSL proxy list, and then reactivate the SSL service.

To view the virtual or back-end SSL servers in a list, use the **show ssl-proxy-list** (see [Chapter 7, Displaying SSL Configuration Information and Statistics](#)).

Use the **suspend** command to suspend an active SSL proxy list.

To suspend an active SSL proxy list, enter:

```
(config-ssl-proxy-list[ssl_list1])# suspend
```

Configuring a Service for Back-End SSL

An SSL proxy list may belong to multiple SSL services (one SSL proxy list per service), and an SSL service may belong to multiple content rules. You can apply the services to content rules that allow the CSS to direct SSL requests for content.

**Note**

The CSS supports one active SSL service for each SSL module in the CSS, one SSL service per slot. You can configure more than one SSL service for a slot but only a single SSL service can be active at a time.

The requirements for the type of service to be added to the back-end content rule is as follows:

- The service must have a configured IP address
- The keepalive type for a back-end service can be none, ICMP, TCP, or SSL. If you configure a TCP or SSL keepalive type, you must configure the keepalive port correctly for the service to work.
- You must configure an SSL proxy list that contains back-end-server configuration for this type of service.

**Note**

If you do not configure a service port, the CSS uses the same port number as the back-end content rule.

This section covers:

- [Creating an SSL Service](#)
- [Configuring the Back-End SSL Service Type](#)
- [Adding an SSL Proxy List for a Back-End SSL Server](#)
- [Configuring an IP Address for a Back-End SSL Service](#)
- [Configuring the Port Number for a Back-End SSL Service](#)
- [Activating the SSL Service](#)
- [Suspending the SSL Service](#)

Creating an SSL Service

When creating a service for use with an SSL module, you must identify it as an SSL service for the CSS to recognize it. For additional details on creating a service, refer to the *Cisco Content Services Switch Content Load-Balancing Configuration Guide*.

Enter the SSL service name, from 1 to 31 characters.

To create service `ssl_serv1`, enter:

```
(config)# service ssl_serv1  
Create service <ssl_serv1>, [y/n]: y
```

The CSS transitions into the newly created service mode.

```
(config-service[ssl_serv1])#
```

Configuring the Back-End SSL Service Type

You must configure the `ssl-accel-backend` service type for a back-end SSL service. To configure a service type for a back-end SSL service, enter:

```
(config-service[server1])# type ssl-accel-backend
```

Adding an SSL Proxy List for a Back-End SSL Server

After you configure an SSL proxy list for a back-end SSL server, add the active list to an SSL service to define how the CSS processes SSL requests for content from a back-end SSL server. Configuring the back-end SSL service is similar to configuring a local service except you must set the service type to `ssl-accel-backend`. Also, this type of service requires an SSL proxy list with a back-end server entry.

An SSL proxy list contains the parameters for the back-end SSL service. To add the proxy list to the service, use the `add ssl-proxy-list` command. For more information on configuring an SSL proxy list for a back-end server, see the [“Configuring the Back-End SSL Service Type”](#) section earlier in this chapter.

Enter the name of the previously created SSL proxy list (see the [“Creating an SSL Proxy List”](#) section in this chapter) that you want to add to the service.

For example, to add the SSL proxy list *ssl list3* for a back-end SSL service, enter:

```
(config-service[server1])# add ssl-proxy-list ssllist3
```

To remove an SSL proxy list for the back-end service, enter:

```
(config-service[server1])# remove ssl-proxy-list ssllist3
```

Configuring an IP Address for a Back-End SSL Service

The IP address for a back-end SSL service must match the IP address configured in the SSL proxy list for the back-end server.

For example, to configure the IP address 10.11.21.13 for the back-end SSL service, enter:

```
(config-service[server1])# ip address 10.11.21.13
```

To remove the IP address for the back-end SSL service, enter:

```
(config-service[server1])# no ip address
```

Configuring the Port Number for a Back-End SSL Service

The CSS uses the port number to send clear text data back to the SSL module for reencryption. By default, the CSS uses the port number of the back-end content rule associated with the service, port 80. If the port number is different from the the back-end HTTP-SSL content rule, use the **port** command to configure it.

Enter the port number as a integer from 1 to 65535. If you configure a port number, it must match the virtual port number configured in the SSL proxy list for the back-end server.

For example, to configure a port number of 55, enter:

```
(config-service[server1])# port 55
```

To reset the port number of the back-end content rule, enter:

```
(config-service[server1])# no port
```

Activating the SSL Service

Once you configure an SSL proxy list service, use the **active** command to activate the service. Activating a service puts it into the resource pool for load-balancing SSL content requests between the client and the server.

Before activating an SSL service:

- For a virtual SSL server, you must add an SSL proxy list to an **ssl-accel** type service before you can activate the service. If no list is configured when you enter the **active** command, the CSS logs the following error message and does not activate the service.

```
Must add at least one ssl-proxy-list to an ssl-accel type service
```

- For a back-end SSL server, you must add an SSL proxy list to an **ssl-accel-backend** type service before you can activate the service. If no list is configured when you enter the **active** command, the CSS logs the following error message and does not activate the service.

```
Must add at least one ssl-proxy-list to an ssl-accel type service
```

- The SSL proxy list added to the service must be active before you can activate the service. If the list is suspended, the CSS logs the following error message and does not activate the service.

```
No ssl-lists on service, service not activated
```

Once the service is ready to activate, the CSS initiates the transfer of appropriate SSL configuration data for each SSL proxy list to a specific SSL module and activates the service. If there is an error in transfer, the CSS logs the appropriate error and does not activate the service.

No modifications may be made to an active SSL proxy list. If modifications are necessary, first suspend the ssl service to make changes to the SSL proxy list entries.

To activate service *ssl_serv1*, enter:

```
(config-service[ssl_serv1])# active
```

Suspending the SSL Service

To suspend an SSL service and remove it from the pool for future load-balancing SSL content requests, use the **suspend** command. Suspending an SSL service does not affect existing content flows, but it prevents additional connections from accessing the service for its content.

You must suspend a service prior to modifying an SSL proxy list.

To suspend service `ssl_serv1`, enter:

```
(config-service[ssl_serv1])# suspend
```

Configuring a Content Rule for Back-End SSL

For the CSS to direct SSL requests for content, apply the back-end services to content rules. No network traffic is sent to an SSL module until you activate an SSL content rule to define where the content physically resides, where to direct the request for content (which SSL service), and which load-balancing method to use.

For an HTTP server or back-end SSL server content rule, ensure that each VIP address and port configured in the rule matches a VIP address and port configured in the cipher suite parameter for a virtual SSL server entry in the SSL proxy list (see the [“Specifying Cipher Suites”](#) section).

For a back-end server, you can specify a Layer 5 cookie or URL rule. The information in the rule finds a sticky server to use or load balances a new server for a new client request.

For more information on Layer 5 sticky and content rules, refer to the *Cisco Content Services Switch Content Load-Balancing Configuration Guide*.



Configuring SSL Initiation

This chapter describes the steps required to configure a CSS to initiate an SSL connection with an SSL server after receiving clear text from a client. It contains the following major sections:

- [Overview of SSL Initiation](#)
- [Creating an SSL Initiation Proxy List](#)
- [Adding a Description to an SSL Initiation Proxy List](#)
- [Configuring Back-End SSL Servers in an SSL Initiation Proxy List](#)
- [Activating and Suspending an SSL Proxy List](#)
- [Configuring a Service for SSL Initiation](#)
- [Configuring a Content Rule for SSL Initiation](#)
- [Troubleshooting SSL Initiation](#)

Overview of SSL Initiation

SSL initiation allows a CSS configured with an SSL module to:

- Receive clear text from a client
- Load balance the content
- Encrypt the clear text
- Originate an SSL connection with either an SSL server or another CSS configured with SSL termination (see [Chapter 4, Configuring SSL Termination](#)).

Use this feature for secure site-to-site data transfers. SSL initiation allows you to send clear text within a site for maximum speed, while sending encrypted text through the Internet between sites or to an SSL server for maximum security. For each SSL server or CSS to which you want to establish an SSL connection from a clear-text connection, you must configure an SSL initiation service on the CSS that maps to that SSL server or CSS. This service uses an SSL proxy list to properly direct the flows within the CSS.

Figure 6-1 illustrates a single SSL initiation flow with an SSL server.

Figure 6-1 SSL Initiation with an SSL Server

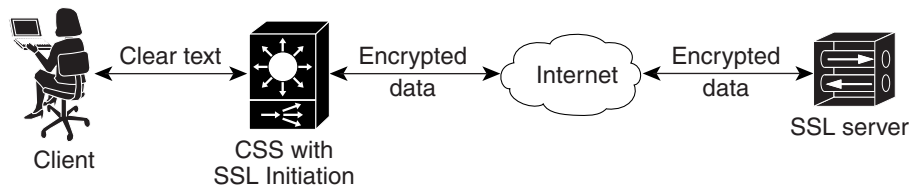
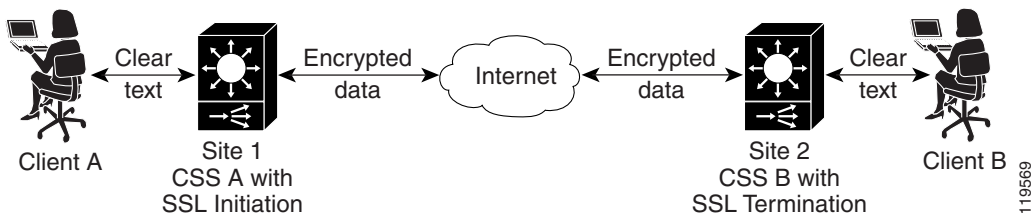


Figure 6-2 illustrates an SSL initiation flow with another CSS configured with SSL termination. In this case, the second CSS acts as a virtual front-end SSL server.

Figure 6-2 SSL Initiation with a Second CSS Running SSL Termination



An SSL proxy list determines the flow of SSL information among the client, SSL module, and the SSL server. An SSL proxy list comprises one or more back-end SSL servers (virtual servers that you create on the CSS SSL module) related by index entry. An SSL module in the CSS uses the back-end SSL server to initiate the connection to an SSL server. You can define a maximum of 256 back-end SSL servers in a single SSL proxy list.

After you create and configure the entries in a proxy list, you must activate the list, and then add the SSL proxy list to an initiation service to enable the transfer of SSL configuration data to the SSL module. When you activate the service, the CSS transfers the configuration data to the module. You can then add each SSL initiation service to an SSL content rule.

Creating an SSL Initiation Proxy List

An SSL initiation proxy list is a group of related back-end SSL servers that are associated with an SSL initiation service. To create an SSL proxy list, use the **ssl-proxy-list** command.

You can access the `ssl-proxy-list` configuration mode from most configuration modes except for ACL, boot, group, rmon, or owner configuration modes. You can also use this command from the `ssl-proxy-list` configuration mode to access another SSL proxy list. Enter the SSL proxy list name as an unquoted text string from 1 to 31 characters.

For example, to create the SSL proxy list, `ssl_list1`, enter:

```
(config)# ssl-proxy-list ssl_list1  
Create ssl-list <ssl_list1>, [y/n]: y
```

Once you create an SSL proxy list, the CLI enters into the `ssl-proxy-list` configuration mode.

```
(config-ssl-proxy-list[ssl_list1])#
```

To delete an existing SSL proxy list, enter:

```
(config)# no ssl-proxy-list ssl_list1  
Delete ssl-list <ssl_list1>, [y/n]: y
```



Note

You cannot delete a given SSL proxy list if any SSL service using that specific SSL proxy list is active. You must first suspend the SSL service to delete the specific SSL proxy list.

Adding a Description to an SSL Initiation Proxy List

To specify a description for an SSL initiation proxy list, use the **description** command. Enter the description as a quoted text string with a maximum of 64 characters, including spaces.

For example, to add a description to the SSL proxy list, `ssl_list1`, enter:

```
(config-ssl-proxy-list[ssl_list1])# description "This is the SSL list  
for www.brandnewproducts.com"
```

To remove the description from a specific SSL proxy list, enter:

```
(config-ssl-proxy-list[ssl_list1])# no description
```

Configuring Back-End SSL Servers in an SSL Initiation Proxy List

This section discusses the creation of one or more back-end SSL servers in an SSL proxy list for use with the SSL initiation feature. In SSL proxy-list mode, use the **backend-server** command to create an SSL initiation back-end server on the CSS and define an index entry in the SSL proxy list. You then use this index to configure specific back-end SSL parameters associated with the SSL proxy list. An SSL module in the CSS uses the back-end SSL server defined in the SSL proxy list to initiate a connection to an SSL server. You must define a back-end SSL server index number before configuring SSL proxy-list parameters. You can define a maximum of 256 back-end SSL initiation servers in a single SSL proxy list.



Note

You cannot modify the back-end SSL servers in an active SSL proxy list. You must first suspend the SSL proxy list to make modifications to any of the back-end servers in a specific SSL proxy list. Once you have modified the SSL proxy list, suspend the SSL service, activate the SSL proxy list, and then activate the SSL service to apply the changes.

Once you create a back-end server in an SSL proxy list, configure a IP address that corresponds to the address of the service and a server IP address that corresponds to the IP address of the SSL initiation server. Configure the other optional proxy-list parameters if desired, and then activate the SSL proxy list. To make the back-end server work for SSL initiation, you must configure the back-end server type as **initiation**.

After you configure and activate the SSL proxy list, add the list to an SSL initiation service. When you activate the service, the CSS sends the configuration data to the SSL module.

The following sections describe:

- [Creating a Back-End Server in an SSL Initiation Proxy List](#)
- [Configuring the Back-End Server as an SSL Initiation Server](#)
- [Configuring an IP Address for the SSL Initiation Server](#)
- [Configuring a Port for the SSL Initiation Server](#)
- [Configuring the SSL Server IP Address](#)
- [Configuring the SSL Server Port](#)
- [Configuring SSL Version](#)
- [Configuring the Available Cipher Suites](#)
- [Configuring SSL Session Cache Timeout](#)
- [Configuring SSL Session Handshake Renegotiation](#)
- [Configuring TCP Virtual Client Connections Timeout Values](#)
- [Configuring TCP Server-Side Connection Timeout Values on the SSL Module](#)
- [Specifying the Nagle Algorithm for Client-Side Connections](#)
- [Specifying the TCP Buffering for SSL TCP Connections](#)
- [Configuring Client Certificates and Keys](#)
- [Configuring CA Certificates for Server Authentication](#)

Creating a Back-End Server in an SSL Initiation Proxy List

Before you can configure SSL initiation proxy list parameters, you must create a back-end server in an SSL proxy list. To create a back-end server in the SSL proxy list, use the **backend-server** *number* command. By default, this command creates a back-end server of type **backend-ssl** and assigns it a number (index entry you enter) in the SSL proxy list that you use to configure specific SSL parameters associated with the back-end SSL server (for example, IP address, certificate name, and key pair). Enter a value from 1 to 256.

For example, to create back-end server 1 in the proxy list, enter:

```
(config-ssl-proxy-list[ssl_list1])# backend-server 1
```

To remove back-end server 1 and all its configured parameters from proxy list `ssl_list1`, enter:

```
(config-ssl-proxy-list[ssl_list1])# no backend-server 1
```

Configuring the Back-End Server as an SSL Initiation Server

To use the back-end server for SSL initiation, you must configure it as an initiation server for use with services of type **ssl-init**. By default, the back-end server is a server of type **backend-ssl** for use with services of type **ssl-accel-backend**.

To configure the back-end server as an initiation server (accepts clear traffic from a client and initiates SSL traffic with the SSL server) for use with services of type **ssl-init**, use the **backend-server** *number* **type** command. The syntax for this command is:

```
backend-server number type { backend-sslinitiation }
```

For example, to configure back-end server 1 as an SSL initiation server in proxy list `ssl_list1`, enter:

```
(config-ssl-proxy-list[ssl_list1])# backend-server 1 type initiation
```

To reconfigure the SSL initiation server as a back-end SSL server without having to configure all the back-end server parameters, enter:

```
(config-ssl-proxy-list[ssl_list1])# backend-server 1 type backend-ssl
```

Configuring an IP Address for the SSL Initiation Server

To configure an IP address for the SSL initiation server, use the **backend-server number ip address** command. The IP address corresponds to the IP address of the SSL initiation service.

For example, to configure the IP address 192.168.2.3 for SSL initiation back-end server 1, enter:

```
(config-ssl-proxy-list[ssl_list1])# backend-server 1 ip address 192.168.2.3
```

To remove the IP address from SSL initiation server 1, enter:

```
(config-ssl-proxy-list[ssl_list1])# no backend-server 1 ip address
```



Note

If you have not configured the SSL initiation back-end server IP address when you issue the **active** command, the following error message appears and the CSS does not activate the list.

```
%% Error in backend-server 1: SSL-server/Backend-server must have valid IP address
```

Configuring a Port for the SSL Initiation Server

By default, the port for the SSL initiation back-end server is port 80. This port accepts the clear text data traffic from the SSL initiation service and sends it to the SSL module. To configure a different port for the SSL initiation server, use the **backend-server number port** command. Enter a port number from 1 to 65535.



Note

If you configure the **backend-server number ip address** and **server-ip** commands with the same address, configure the **backend-server number port** and **server-port** commands with different port numbers.

For example, to configure a port number of 1200, enter:

```
(config-ssl-proxy-list[ssl_list1])# backend-server 1 port 1200
```

To reset the port to the default value of 80, enter:

```
(config-ssl-proxy-list[ssl_list1])# no backend-server 1 port
```

Configuring the SSL Server IP Address

The server IP address is the IP address for the real SSL server. To configure the IP address for the real SSL server, use the **backend-server *number* server-ip** command.

For example, to configure the server IP address 192.168.2.3, enter:

```
(config-ssl-proxy-list[ssl_list1])# backend-server 1 server-ip
192.168.2.3
```

To remove the real server IP address from the proxy list, enter:

```
(config-ssl-proxy-list[ssl_list1])# no backend-server 1 server-ip
```



Note

If you have not configured the real server IP address when you issue the **active** command, the following error message appears and the CSS does not activate the list.

```
%% Error in backend-server 1: SSL-server/Backend-server must have valid
IP address
```

Configuring the SSL Server Port

By default the port number for the real SSL server is port 443. To configure a different server port for the SSL server, use the **backend-server *number* server-port** command. Enter a port number from 1 to 65535.



Note

If you configure the **backend-server *number* ip address** and **server-ip** commands with the same address, configure the **backend-server *number* port** and **server-port** commands with different port numbers.

For example, to configure the server port number 155, enter:

```
(config-ssl-proxy-list[ssl_list1])# backend-server 1 server-port 155
```

To reset the port to the default value of 443, enter:

```
(config-ssl-proxy-list[ssl_list1])# no backend-server 1 server-port
```

Configuring SSL Version

The SSL module initiates the connection to the real SSL server. The version in the ClientHello message sent to the server indicates the highest supported version.

By default, the SSL version is SSL version 3 and TLS version 1. The SSL module sends a ClientHello that has an SSL version 3 header with the ClientHello message set to TLS version 1.

Use the **backend-server number version** command to specify which version of SSL the back-end server supports:

- **ssl3** - SSL version 3.
- **tls1** - TLS version 1.
- **ssl-tls** - SSL version 3 and TLS version 1. The SSL module sends a ClientHello that has an SSL version 3 header with the ClientHello message set to TLS version 1.

For example, to configure the SSL version 3, enter:

```
(config-ssl-proxy-list[ssl_list1])# backend-server 1 version ssl3
```

To reset the default SSL version, enter:

```
(config-ssl-proxy-list[ssl_list1])# no backend-server 1 version
```

Configuring the Available Cipher Suites

To configure one or more specific cipher suites to be used by the back-end SSL initiation server, use the **backend-server number cipher** command. By default, all supported hardware accelerated cipher suites are enabled.

For a list of all cipher suites that the SSL module supports and the corresponding cipher suite values, see [Table 4-1](#) in the “[Specifying Cipher Suites](#)” section in [Chapter 4, Configuring SSL Termination](#). These values match those defined for SSL version 3.0 and TLS version 1.0. [Table 4-1](#) also lists those Cipher suites that are exportable in any version of the software.

If you use the default setting or select the **all-cipher-suite** option, the CSS sends the suites in the same order as they appear in [Table 4-1](#), starting with `rsa-with-rc4-128-md5`.

**Note**

The **all-cipher-suites** option reenables all cipher suites for the back-end server. This option works only when you do not configure specifically-defined ciphers. To return to using the **all-cipher-suites** option, you must remove all specifically-defined ciphers.

For example, to configure a cipher of `rsa-with-rc4-128-md5`, enter:

```
(config-ssl-proxy-list[ssl_list1])# backend-server 1 cipher
rsa-with-rc4-128-md5
```

When negotiating which cipher suite to use, the SSL module sends the ciphers in weighted order to the server with the highest weighted cipher first in the list.

By default, all configured cipher suites have a weight of 1. Optionally, you can assign a priority weight to the cipher suite, with 10 being the highest.

**Note**

If two or more ciphers have the same weight (no weight has a value of 1), the ciphers appear in the Client Hello in the same order as they appear in the running-configuration file.

For example, to set a weight of 10 to a cipher suite, enter:

```
(config-ssl-proxy-list[ssl_list1])# backend-server 1 cipher
rsa-with-rc4-128-md5 weight 10
```

To remove one or more of the configured cipher suites for the SSL initiation back-end server, enter:

```
(config-ssl-proxy-list[ssl_list1])# no backend-server 1 cipher
rsa-with-rc4-128-md5
```

Configuring SSL Session Cache Timeout

In SSL, every time a client and server go through a full key exchange and establish a new master secret key, a new session is created. Enabling a session cache timeout allows the reuse of the master key on subsequent connections by the client. When you disable the cache timeout, the full SSL handshake must occur on each new connection to the SSL module (the virtual client). Use the **backend-server *number* session-cache** command to configure the SSL module to resume connection with a back-end SSL server using a previously established secret key.

By default, the cache timeout is enabled with a timeout of 300 seconds (5 minutes). The timeout value can range from 0 to 72000 (0 seconds to 20 hours). A timeout value of 0 disables the session cache reuse.

For example, to configure the SSL session cache timeout of 500 seconds, enter:

```
(config-ssl-proxy-list[ssl_list1])# backend-server 1 session-cache 500
```

To reset the session cache ID reuse to the default of enabled with a timeout of 300 seconds, enter:

```
(config-ssl-proxy-list[ssl_list1])# no backend-server 1 session-cache
```

To disable session cache ID reuse, enter a timeout value of 0 seconds:

```
(config-ssl-proxy-list[ssl_list1])# backend-server 1 session-cache 0
```

Configuring SSL Session Handshake Renegotiation

The SSL session handshake commands send the SSL HelloRequest message to a client to restart SSL handshake negotiation. SSL rehandshake is useful when a connection has been established for a lengthy period of time and you want to ensure security by reestablishing the SSL session between the CSS and the back-end SSL server.

Use the **backend-server *number* handshake data *kbytes*** command to force an SSL rehandshake after the exchange of a certain amount of data between the CSS and the back-end SSL server, after which the CSS transmits the SSL handshake message and reestablishes the SSL session.

By default, the SSL rehandshake based on data (flow) is disabled (set to 0) for a back-end SSL server after the exchange of data. The data value is in kilobytes and is from 0 to 512000 kilobytes.

For example, to configure the SSL session rehandshake data value of 500 Kbytes, enter:

```
(config-ssl-proxy-list[ssl_list1])# backend-server 1 handshake data 500
```

To reset the rehandshake data value to 0, disable the rehandshake based on the exchange of data. For example, enter:

```
(config-ssl-proxy-list[ssl_list1])# no backend-server 1 handshake data
```

Use the **backend-server number handshake timeout seconds** command to specify a maximum timeout value, after which the CSS transmits the SSL handshake message and reestablishes the SSL session. Setting a timeout value forces the SSL session to renegotiate a new session key after a session has lasted the defined number of seconds. The selection of an SSL rehandshake timeout value is important when using the **advanced-balance ssl** load-balancing method for a Layer 5 content rule to fine-tune the SSL session ID used to stick the client to the server.

By default, the SSL rehandshake timeout is disabled (set to 0) for the back-end SSL server. The timeout value is from 0 to 72000 (0 seconds to 20 hours).

For example, to configure a 30-second timeout of an SSL session rehandshake, enter:

```
(config-ssl-proxy-list[ssl_list1])# backend-server 1 handshake timeout 30
```

To reset the timeout to 0, disable the rehandshake timeout period for the back-end server by entering:

```
(config-ssl-proxy-list[ssl_list1])# no backend-server 1 handshake timeout
```

Configuring TCP Virtual Client Connections Timeout Values

The TCP connection between the client and the SSL module is terminated when the specified time interval elapses. The TCP timeout functions enable you to have more control over the TCP connection between the client and the SSL module.

To configure the parameters for the TCP connection with the client, see the following sections:

- [Specifying a TCP SYN Timeout Value for the Virtual Client Connection](#)
- [Specifying a TCP Inactivity Timeout for a Virtual Client Connection](#)
- [Specifying the Nagle Algorithm for Client-Side Connections](#)

Specifying a TCP SYN Timeout Value for the Virtual Client Connection

The CSS SYN timer counts the time difference between the CSS sending the SYN/ACK and the client replying with an ACK as the means to terminate the TCP three-way handshake. Use the **backend-server *number* tcp virtual syn-timeout *seconds*** command to specify a timeout value that the CSS uses to terminate a TCP connection with a client and the SSL module that has not successfully completed the TCP three-way handshake prior to transferring data.

Enter a TCP SYN inactivity timeout value in seconds, from 0 (TCP SYN timeout disabled) to 3600 (1 hour). The default is 30 seconds. When you set the command to 0, the timer becomes inactive and the retransmit timer eventually terminates a broken TCP connection.



Note

The connection timer should always be less than the retransmit termination time for new TCP connections.

To configure the TCP SYN timeout of 100 seconds, enter:

```
(config-ssl-proxy-list[ssl_list1])# backend-server 1 tcp virtual  
syn-timeout 100
```

To disable the timeout, set the value to 0:

```
(config-ssl-proxy-list[ssl_list1])# backend-server 1 tcp virtual  
syn-timeout 0
```

To reset the timeout to the default value of 30 seconds, enter:

```
(config-ssl-proxy-list[ssl_list1])# no backend-server 1 tcp virtual  
syn-timeout
```

Specifying a TCP Inactivity Timeout for a Virtual Client Connection

The TCP inactivity timeout begins once the CSS receives an ACK from the client to terminate the TCP three-way handshake. The inactivity timer resumes immediately following where the SYN timer stops, with regard to traffic flow. Use the **backend-server number tcp virtual inactivity-timeout seconds** command to specify a timeout value that the CSS uses to terminate a TCP connection with the client and the SSL module when there is little or no activity occurring on the connection.

Enter a TCP inactivity timeout value in seconds from 0 (TCP inactivity timeout disabled) to 3600 (1 hour). The default is 240 seconds.

Based on the default parameters for retransmission, the timer value should be larger than 60 seconds (1 minute).

For example, to configure the TCP inactivity timeout period of 100 seconds for the virtual client connection, enter:

```
(config-ssl-proxy-list[ssl_list1])# backend-server 1 tcp virtual  
inactivity-timeout 100
```

To disable the timeout, set the value to 0:

```
(config-ssl-proxy-list[ssl_list1])# backend-server 1 tcp virtual  
inactivity-timeout 0
```

To reset the timeout to the default value of 240 seconds, enter:

```
(config-ssl-proxy-list[ssl_list1])# no backend-server 1 tcp virtual  
inactivity-timeout
```

Specifying the Nagle Algorithm for Client-Side Connections

The TCP Nagle algorithm automatically concatenates a number of small buffer messages transmitted over the TCP connection between a client and the SSL module. This process increases the throughput of your CSS by decreasing the number of packets sent over each TCP connection. However, the interaction between the Nagle algorithm and the TCP delay acknowledgment may increase latency in your TCP connection. Disable the Nagle algorithm when you observe an unacceptable delay in a TCP connection (clear-text or SSL).

Use the **backend-server *number* tcp virtual nagle** command to disable or reenble the Nagle algorithm for the TCP connection between the client and the SSL module. The syntax for this command is:

backend-server *number* tcp virtual nagle enable/disable

To disable the Nagle algorithm for the TCP connection between the client and the SSL module, enter:

```
(config-ssl-proxy-list[ssl_list1])# backend-server 1 tcp virtual nagle disable
```

To reenble the Nagle algorithm for the TCP connection between the client and the SSL module, enter:

```
(config-ssl-proxy-list[ssl_list1])# backend-server 1 tcp virtual nagle enable
```

Configuring TCP Server-Side Connection Timeout Values on the SSL Module

The TCP connection between the SSL module and a server is terminated when the specified time interval elapses. The TCP timeout functions enable you to have more control over TCP connections between the CSS SSL module and a server.

To configure the timeout values of a TCP connection with the server, see the following sections:

- [Specifying a TCP SYN Timeout Value for a Server-Side Connection](#)
- [Specifying a TCP Inactivity Timeout for a Server-Side Connection](#)
- [Specifying the Nagle Algorithm for Server-Side Connections](#)

Specifying a TCP SYN Timeout Value for a Server-Side Connection

The TCP SYN timer counts the time difference between the CSS initiating the back-end TCP connection by transmitting a SYN and the server replying with a SYN/ACK. Use the **backend-server number tcp server syn-timeout seconds** command to specify a timeout value that the CSS uses to end a TCP connection with a server that has not successfully completed the TCP three-way handshake prior to transferring data.

Enter a TCP SYN timeout value in seconds from 0 (TCP SYN timeout disabled) to 3600 (1 hour). The default is 30 seconds. When you set the command to 0, the timer becomes inactive and the retransmit timer eventually terminates a broken TCP connection.



Note

The connection timer should always be less than the retransmit termination time for new TCP connections.

For example, to configure the TCP SYN timeout of 100 seconds for the server-side connection, enter:

```
(config-ssl-proxy-list[ssl_list1])# backend-server 1 tcp server  
syn-timeout 100
```

To disable the timeout, set the value to 0:

```
(config-ssl-proxy-list[ssl_list1])# backend-server 1 tcp server  
syn-timeout 0
```

To reset the timeout to the default value of 30 seconds, enter:

```
(config-ssl-proxy-list[ssl_list1])# no backend-server 1 tcp server  
syn-timeout
```

Specifying a TCP Inactivity Timeout for a Server-Side Connection

The TCP inactivity timeout begins once the CSS receives a SYN/ACK from the server. The inactivity timer resumes immediately following where the SYN timer stops, with regard to traffic flow. Use the **backend-server *number* tcp server inactivity-timeout *seconds*** command to specify a timeout value that the CSS uses to terminate a TCP connection with a server when there is little or no activity occurring on the connection.

Enter a TCP inactivity timeout value in seconds from 0 (TCP inactivity timeout disabled) to 3600 (1 hour). The default is 240 seconds.

For example, to configure the TCP inactivity timeout period of 100 seconds for the server-side connection, enter:

```
(config-ssl-proxy-list[ssl_list1])# backend-server 1 tcp server  
inactivity-timeout 100
```

To disable the timeout, set the value to 0:

```
(config-ssl-proxy-list[ssl_list1])# backend-server 1 tcp server  
inactivity-timeout 0
```

To reset the timeout to the default value of 240 seconds, enter:

```
(config-ssl-proxy-list[ssl_list1])# no backend-server 1 tcp server  
inactivity-timeout
```

Specifying the Nagle Algorithm for Server-Side Connections

The TCP Nagle algorithm automatically concatenates a number of small buffer messages transmitted over the TCP connection between a back-end server and the SSL module. This process increases the throughput of your CSS by decreasing the number of packets sent over each TCP connection. However, the interaction between the Nagle algorithm and the TCP delay acknowledgment may increase latency in your TCP connection. Disable the Nagle algorithm when you observe an unacceptable delay in a TCP connection (clear-text or SSL).

Use the **backend-server *number* tcp server nagle enable|disable** command to disable or reenble the Nagle algorithm for the TCP connection between the server and the SSL module. The syntax for this command is:

```
backend-server number tcp server nagle enable|disable
```

To disable the Nagle algorithm for the TCP connection between the server and the SSL module, enter:

```
(config-ssl-proxy-list[ssl_list1])# backend-server 1 tcp server nagle
disable
```

To reenable the Nagle algorithm for the TCP connection between the server and the SSL module, enter:

```
(config-ssl-proxy-list[ssl_list1])# backend-server 1 tcp server nagle
enable
```

Specifying the TCP Buffering for SSL TCP Connections

If you experience unacceptable latency in your network due to congestion, you can increase the buffer size that the CSS buffers for a given TCP connection before shutting down the TCP window to 0. Use the **backend-server number tcp buffer-share** command to set the TCP buffering from the client or server on a given connection. The syntax for this command is:

```
backend-server number tcp buffer-share rx number1 tx number2
```

To set the amount of data in bytes that a given connection can buffer from the client traffic, use the **rx number1** keyword and variable. By default, the buffer size is 32768. The buffer size can range from 16400 to 262144. For example, to set the value to 65536, enter:

```
(config-ssl-proxy-list[ssl_list1])# backend-server 1 tcp buffer-share
rx 65536
```

To reset the buffer size to the default of 32768, enter:

```
(config-ssl-proxy-list[ssl_list1])# no backend-server 1 tcp
buffer-share rx
```

To set the amount of data in bytes that a given connection can buffer from the server to the client, use the **tx number2** keyword and variable. By default, the buffer size is 65536. The buffer size can range from 16400 to 262144. For example, to set the value to 131072, enter:

```
(config-ssl-proxy-list[ssl_list1])# backend-server 1 tcp buffer-share
tx 131072
```

To reset the buffer size to the default of 65536, enter:

```
(config-ssl-proxy-list[ssl_list1])# no backend-server 1 tcp  
buffer-share tx
```

Configuring Client Certificates and Keys

SSL servers frequently require that a client authenticate itself before a data transfer can occur. To allow the client (in this case, the SSL module) to authenticate itself to such a server, you must configure client certificates and keys on the CSS.

To obtain a client certificate and key pair, contact your authorized certificate authority (CA). Once the CA has prepared your client certificate and key pair, you must import them into the CSS. For information about importing a certificate and key pair, see the [“Importing or Exporting Certificates and Private Keys”](#) section in [Chapter 3, Configuring SSL Certificates and Keys](#). Once you have imported the certificate and key pair, you must associate them with a filename. For information about associating a certificate and key pair with filenames, see the [“Associating Certificate and Private Key Files with Names”](#) section in [Chapter 3, Configuring SSL Certificates and Keys](#).

If the SSL module originates a connection to an SSL server that requests a client certificate and no client certificate and key are configured on the CSS, the CSS increments the Requested Client Certificate Not Sent counter.



Note

When the SSL server does not receive the requested client certificate, it may close the connection.

The following sections describe how to configure client certificates and keys.

Configuring the RSA Certificate Name

To configure the back-end server RSA certificate, use the **backend-server number rsacert name** command. The certificate must already be loaded on the SCM. If the certificate name does not exist, the CSS logs an error message. Enter a name for the RSA certificate as an unquoted text string from 1 to 31 characters.

For example, to configure an RSA certificate named myrsacert, enter:

```
(config-ssl-proxy-list[ssl_list1])# backend-server 1 rsacert myrsacert
```

To remove an RSA cert from the SSL proxy list, enter:

```
(config-ssl-proxy-list[ssl_list1])# no backend-server 1 rsacert
```

Configuring the RSA Key Name

To configure the back-end server RSA key name, use the **backend-server number rsakey name** command. The key pair must already be loaded on the SCM. If the key pair name does not exist, the CSS logs an error message. Enter a name for the RSA key pair as an unquoted text string from 1 to 31 characters.

For example, to configure an RSA key pair named myrsakey, enter:

```
(config-ssl-proxy-list[ssl_list1])# backend-server 1 rsakey myrsakey
```

To remove an RSA key pair from the SSL proxy list, enter:

```
(config-ssl-proxy-list[ssl_list1])# no backend-server 1 rsakey
```

Configuring Diffie Hellman Parameters

To configure the back-end server Diffie-Hellman (DH) parameter file, use the **backend-server number dhparam name** command. The DH parameters file must already be loaded on the SCM. If the parameter file does not exist, the CSS logs an error message. Enter a name for the DH parameter files as an unquoted text string from 1 to 31 characters.

For example, to configure a DH parameter file named dhparamfile2, enter:

```
(config-ssl-proxy-list[ssl_list1])# backend-server 1 dhparam  
dhparamfile2
```

To remove the configured DH parameter file from the SSL proxy list, enter:

```
(config-ssl-proxy-list[ssl_list1])# no backend-server 1 dhparam
```

Configuring the DSA Certificate Name

To configure the back-end server DSA certificate, use the **backend-server number dsacert name** command. The certificate must already be loaded on the SCM. If the certificate name does not exist, the CSS logs an error message. Enter a name for the DSA certificate as an unquoted text string from 1 to 31 characters.

For example, to configure a DSA certificate named mydsacert, enter:

```
(config-ssl-proxy-list[ssl_list1])# backend-server 1 dsacert mydsacert
```

To remove a DSA cert from the SSL proxy list, enter:

```
(config-ssl-proxy-list[ssl_list1])# no backend-server 1 dsacert
```

Configuring the DSA Key Filename

To configure the back-end server DSA key name, use the **backend-server number dsakey name** command. The key pair must already be loaded on the SCM. If the key pair name does not exist, the CSS logs an error message. Enter a name for the DSA key pair as an unquoted text string from 1 to 31 characters.

For example, to configure a DSA key pair named mydsakey, enter:

```
(config-ssl-proxy-list[ssl_list1])# backend-server 1 dsakey mydsakey
```

To remove an DSA key pair from the SSL proxy list, enter:

```
(config-ssl-proxy-list[ssl_list1])# no backend-server 1 dsakey
```

Configuring CA Certificates for Server Authentication

If the it has the public key of a particular certificate authority (CA), the CSS can verify that the server certificate was signed by that CA. The CSS obtains the public key of the CA from the CA certificate. If you configure a CA certificate name in an SSL initiation proxy list, the CSS can use the public key in the certificate to verify the digital signature of the CA in the server certificate. Defining a CA certificate in the SSL initiation proxy list indicates to the CSS that you want to verify the server certificate.



Note

By default, SSL servers are not authenticated.

Before you configure the CA certificate in an SSL initiation proxy list, you must import the certificate on the CSS and then associate the certificate with a filename. For information about importing a CA certificate, see the [“Importing or Exporting Certificates and Private Keys”](#) section in [Chapter 3, Configuring SSL Certificates and Keys](#). For information about associating a certificate with a filename, see the [“Associating Certificate and Private Key Files with Names”](#) section in [Chapter 3, Configuring SSL Certificates and Keys](#).

To enable the SSL module (the client) to authenticate the SSL server, you must configure at least one, with a maximum of four, CA certificates in the SSL initiation proxy list. If you attempt to configure more than four CA certificates, the CSS displays the following error message:

```
%% Max number of CA Certificates configured on server.
```

Use the **cacert** command to configure the CA certificates in the proxy list. The syntax for this command is:

```
backend-server number cacert {name}
```

The *name* variable specifies the filename with which you have previously associated the CA certificate. Enter a filename from 1 to 31 characters. The CA certificate must already be loaded on the SCM. You can define a maximum of four CA certificates for each SSL initiation proxy list. The CSS uses the CA certificates to verify the server certificate in the order in which you configure the CA certificates.

For example, to configure the mycert1 CA certificate in proxy list ssl_list1 for SSL initiation server 1, enter:

```
(config-ssl-proxy-list[ssl_list1])# backend-server 1 cacert mycert1
```

To remove a CA certificate from an SSL proxy list, use the **no** form of the command. For example, to remove the mycert1 CA certificate from the ssl_list1 proxy list for SSL initiation back-end server 1, enter:

```
(config-ssl-proxy-list[ssl_list1])# no backend-server 1 cacert mycert1
```

Activating and Suspending an SSL Proxy List

Before you can activate an SSL proxy list, ensure that you have created at least one back-end SSL server configured as type **initiation** in the list. See the [“Configuring Back-End SSL Servers in an SSL Initiation Proxy List”](#) section.

The CSS checks the SSL proxy list to verify that all of the necessary components are configured, including verification of the certificate and key pair against each other. If the verification fails, the certificate name is not accepted and the CSS logs the error message `Certificate and key pair do not match` and does not activate the SSL proxy list. You must either remove the configured key pair or configure a valid certificate.

Use the **active** command to activate the new or modified SSL proxy list. For example, enter:

```
(config-ssl-proxy-list[ssl_list1])# active
```

After you activate an SSL proxy list, you can add it to a service. See the [“Configuring a Service for SSL Initiation”](#) section.



Note

No modifications to an SSL proxy list are permitted on an active list. Suspend the list prior to making changes, and then reactivate the SSL proxy list once the changes are complete. Once you have modified the SSL proxy list, suspend the SSL service, reactivate the SSL proxy list, and then reactivate the SSL service to apply the changes.

To display the back-end SSL servers configured in a proxy list, use the **show ssl-proxy-list** command (see [Chapter 7, Displaying SSL Configuration Information and Statistics](#)).

Use the **suspend** command to suspend an active SSL proxy list.

To suspend an active SSL proxy list, enter:

```
(config-ssl-proxy-list[ssl_list1])# suspend
```

Configuring a Service for SSL Initiation

SA n SSL proxy list may belong to multiple SSL services (one SSL proxy list per service), and an SSL service may belong to multiple content rules. You can apply the services to content rules, which allow the CSS to direct clear content requests to the SSL module for encryption.

The requirements for the type of service to be added to the SSL initiation content rule is as follows:

- The service must have a configured IP address.
- The keepalive type for an SSL initiation service can be none, ICMP, TCP, or SSL. If you configure a TCP or SSL keepalive type, you must configure the keepalive port correctly for the service to work.
- You must configure an SSL proxy list that contains an SSL initiation back-end server configuration for a service of type **ssl-init**.

**Note**

The CSS supports multiple active SSL services of type **ssl-init** for each SSL module in the CSS.

This section covers:

- [Creating an SSL Service](#)
- [Configuring the SSL Service Type](#)
- [Configuring an IP Address for an SSL Initiation Service](#)
- [Adding an SSL Proxy List to an SSL Initiation Service](#)
- [Specifying the SSL Module Slot](#)
- [Configuring the SSL Initiation Service Keepalive Type](#)
- [SSL Session ID Cache Size](#)
- [Activating the SSL Service](#)
- [Suspending the SSL Service](#)

**Note**

If you do not configure a service port, the CSS uses the same port number as the content rule.

Creating an SSL Service

When creating a service for use with an SSL module, you must identify it as an SSL service for the CSS to recognize it. You can create multiple SSL services for use with an SSL initiation content rule. For additional details on creating a service, refer to the *Cisco Content Services Switch Content Load-Balancing Configuration Guide*.

Enter the SSL service name from 1 to 31 characters.

To create service *ssl_serv1*, enter:

```
(config)# service ssl_serv1  
Create service <ssl_serv1>, [y/n]: y
```

The CSS transitions to the service mode.

```
(config-service[ssl_serv1])#
```

Configuring the SSL Service Type

You must configure the **ssl-init** service type for an SSL initiation service. To configure a service type for an SSL initiation service, enter:

```
(config-service[server1])# type ssl-init
```

Configuring an IP Address for an SSL Initiation Service

The IP address for an SSL initiation service must match the IP address configured in the SSL proxy list for the SSL initiation back-end server (the **backend server-ip** address).

For example, to configure the IP address 192.168.21.7 for the SSL initiation service, enter:

```
(config-service[server1])# ip address 192.168.21.7
```

To remove the IP address for the SSL initiation service, enter:

```
(config-service[server1])# no ip address
```

Adding an SSL Proxy List to an SSL Initiation Service

After you configure an SSL proxy list for an SSL initiation server, add the active list to one or more SSL initiation services to define how the CSS processes SSL requests for content from an SSL initiation back-end server. Configuring the SSL initiation service is similar to configuring a local service except that you must set the service type to **ssl-init**. Also, an SSL initiation service requires an SSL proxy list with a back-end server entry.

An SSL proxy list contains the parameters for the SSL initiation service. To add the proxy list to the service, use the **add ssl-proxy-list** command. For more information on configuring an SSL proxy list for SSL initiation, see the [“Creating an SSL Initiation Proxy List”](#) section.

Enter the name of the previously created SSL proxy list (see the [“Creating an SSL Initiation Proxy List”](#) section) that you want to add to the service.

For example, to add the SSL proxy list *ssl_list1* for an SSL initiation service, enter:

```
(config-service[ssl_serv1])# add ssl-proxy-list ssl_list1
```

To remove an SSL proxy list for the SSL initiation service, enter:

```
(config-service[ssl_serv1])# remove ssl-proxy-list ssl_list1
```

Specifying the SSL Module Slot

The CSS 11501 supports a single integrated SSL module. The CSS 11503 and CSS 11506 support multiple SSL modules; a maximum of two in a CSS 11503 and a maximum of four in a CSS 11506. The SSL service requires the SSL module slot number to correlate the SSL proxy list and virtual SSL server(s) to a specific SSL module. Use the **slot** command to specify the slot in the CSS chassis where the SSL module is located.

The valid slot entries are:

- CSS 11501 - 2
- CSS 11503 - 2 and 3
- CSS 11506 - 2 to 6

Slot 1 is reserved for the SCM.

**Note**

The CSS supports multiple active SSL services of type **ssl-init** for each SSL module in the CSS.

For example, to identify an SSL module in slot 3 of the CSS chassis, enter:

```
(config-service[ssl_serv1])# slot 3
```

Configuring the SSL Initiation Service Keepalive Type

A service of type **ssl-init** supports the use of keepalives to periodically check the health of the SSL server. The CSS sends the keepalives to the IP address configured on the service. To configure a keepalive, use the **keepalive type** command in service configuration mode. The syntax for this service configuration mode command is:

```
(config-service[server1])# keepalive type type
```

For the *type* variable, enter one of the following keepalive types:

- **icmp** - An ICMP echo message (ping). This is the default keepalive type.
- **none** - Do not send keepalive messages to a service.
- **ssl** - SSL HELLO keepalives for this service. Use this keepalive for all back-end services supporting SSL. The CSS sends a client HELLO to connect the SSL server. After the CSS receives a HELLO from the server, the CSS closes the connection with a TCP RST.
- **tcp** - A TCP session that determines service viability through a 3-way handshake and reset; SYN, SYN-ACK, ACK, RST-ACK.

**Note**

If you configure either the SSL or TCP keepalive type, you need to configure the port used by the keepalive.

For more information about these and other CSS keepalives, refer to the *Cisco Content Services Switch Content Load-Balancing Configuration Guide*.

SSL Session ID Cache Size

The cache size is the maximum number of SSL session IDs that can be stored in a dedicated session cache on an SSL module. For services of type **ssl-init**, the SSL session cache size is fixed at 4096 entries and is not configurable.

Activating the SSL Service

Once you configure an SSL proxy list service, use the **active** command to activate the service. Activating a service puts it into the resource pool for load-balancing SSL content requests between the client and the server.

Before activating an SSL service:

- For an initiation SSL server, you must add an SSL proxy list to an **ssl-init** type service before you can activate the service. If no list is configured when you enter the **active** command, the CSS logs the following error message and does not activate the service.

```
Must add at least one ssl-proxy-list to an ssl-init type service
```

- The SSL proxy list added to the service must be active before you can activate the service. If the list is suspended, the CSS logs the following error message and does not activate the service.

```
No ssl-lists on service, service not activated
```

Once the service is ready to activate, the CSS initiates the transfer of appropriate SSL configuration data for each SSL proxy list to a specific SSL module and activates the service. If there is an error in transfer, the CSS logs the appropriate error and does not activate the service.

No modifications may be made to an active SSL proxy list. If modifications are necessary, first suspend the SSL service to make changes to the SSL proxy list entries.

To activate service *ssl_serv1*, enter:

```
(config-service[ssl_serv1])# active
```

Suspending the SSL Service

To suspend an SSL service and remove it from the pool for future load-balancing SSL content requests, use the **suspend** command. Suspending an SSL service does not affect existing content flows, but it prevents additional connections from accessing the service for its content.

You must suspend a service before you can reactivate its SSL proxy list.

To suspend service `ssl_serv1`, enter:

```
(config-service[ssl_serv1])# suspend
```

Configuring a Content Rule for SSL Initiation

For the CSS to encrypt clear client requests for content, apply the SSL initiation services to content rules. A content rule defines:

- Where the content physically resides
- Where to direct the request for content (which SSL initiation services)
- Which load-balancing method to use

For an HTTP server or back-end SSL server content rule, ensure that each configured service IP address matches an IP address configured for an SSL initiation server in the SSL proxy list (see the [“Configuring an IP Address for the SSL Initiation Server”](#) section).

For an SSL initiation content rule, you can specify a Layer 5 cookie or URL rule. The information in the rule enables the CSS to locate a sticky server to use or to load balance a new server for a new client request.

For more information on Layer 5 sticky and content rules, refer to the *Cisco Content Services Switch Content Load-Balancing Configuration Guide*.

Troubleshooting SSL Initiation

The following information is designed to assist you in troubleshooting issues that you may encounter when configuring SSL initiation.

For issues with the SSL proxy list:

- Verify that you have configured the back-end server as type initiation. See the [“Configuring the Back-End Server as an SSL Initiation Server”](#) section.
- Verify that you have added the SSL proxy list to a service of type **ssl-init** and you have activated the service. See the [“Configuring a Service for SSL Initiation”](#) section.
- Verify that you have added the SSL service to a content rule and you have activated the content rule. See the [“Configuring a Content Rule for SSL Initiation”](#) section.

For issues with client certificates:

- Verify that you have configured the client certificate and key on the appropriate back-end server in the SSL proxy list. See the [“Configuring Client Certificates and Keys”](#) section.
- Verify that you have added the SSL proxy list to a service of the type for which the back-end server will be used. Use the **type ssl-init** command for SSL initiation and the **type ssl-accel-backend** command for back-end SSL. See the [“Configuring a Service for SSL Initiation”](#) section.
- Verify that you have added the SSL service to a content rule and that the content rule is active. See the [“Configuring a Content Rule for SSL Initiation”](#) section.
- Ensure that the SSL server is configured to request a client certificate.
- Use a sniffer on the back-end connection to verify that the server is requesting a client certificate and that the CSS is sending the certificate.



Displaying SSL Configuration Information and Statistics

This chapter describes the **show** commands available for displaying CSS SSL configuration information and statistics and an explanation of the fields displayed in the **show** command output. It contains the following major sections:

- [Showing Certificate and Key Pair Information](#)
- [Showing SSL Proxy Configuration Information](#)
- [Showing CRL Record Configuration](#)
- [Showing SSL URL Rewrite Statistics](#)
- [Showing SSL Module Statistics](#)
- [Clearing SSL Statistics](#)
- [Showing SSL Flows](#)

Showing Certificate and Key Pair Information

A number of **show** commands in the CSS enable you to display information about SSL certificates and key pairs stored on the CSS. Enter the following **show** commands from any mode:

- **show ssl associate cert** - Displays certificate associations
- **show ssl associate rsakey** - Displays RSA key pair associations
- **show ssl associate dsakey** - Displays DSA key pair associations

- **show ssl associate dhparam** - Displays information about Diffie-Hellman parameter associations
- **show ssl associate** - Displays all file associations for the CSS
- **show ssl files** - Displays all certificate, key pair, and Diffie-Hellman parameter files loaded on the CSS

Showing SSL Certificates

Use the **show ssl associate cert *certname*** command to display summary data for certificate associations in the CSS. You can optionally specify a certificate name to view detailed information about the certificate, corresponding to the certificate association. If you do not specify a certificate name, all certificate associations appear in the **show ssl associate cert** output.

To display information about all certificate associations, enter:

```
show ssl associate cert
```

[Table 7-1](#) describes the fields in the **show ssl associate cert** output.

Table 7-1 *Field Descriptions for the show ssl associate cert Command*

Field	Description
Certificate Name	The name of the certificate association
File Name	The name of the file containing the certificate
Used By List	Indicates if the certificate association is used by the SSL proxy list containing the VIP address of the virtual server

To display information about a specific certificate association, enter:

```
show ssl associate cert myrsacert1
```

Table 7-2 describes the fields in the `show ssl associate cert certname` output.

Table 7-2 *Field Descriptions for the show ssl associate cert certname Command*

Field	Description
Certificate	The name of the Certificate Association (CA) that issued the certificate.
Version	The version of the certificate.
Serial Number	The serial number associated with the certificate.
Signature Algorithm	The digital signature algorithm (such as RSA) used for the encryption of information with a public/private key pair.
Issuer	The organization that generated the certificate and will vouch for it. An issuer is also the Certificate Authority (CA).
Validity	
Not Before	The starting time period, before which the certificate is not considered valid.
Not After	The ending time period, after which the certificate is not considered valid.
Subject	The certified party that possesses the private key.
Subject Public Key Info	
Public Key Algorithm	The name of the key exchange algorithm used to generate the public key (for example, RSA).
RSA Public Key	The number of bits in the key to define the size of the RSA key pair used to secure Web transactions.
Modulus	The actual public key on which the certificate was built.
Exponent	One of the base numbers used to generate the key.
X509v3 Extensions	An array of X509v3 extensions added to the certificate.

Table 7-2 *Field Descriptions for the show ssl associate cert certname Command (continued)*

Field	Description
X509v3 Basic Constraints	Indicates if the subject may act as a CA, with the certified public key being used to verify certificate signatures. If so, a certification path length constraint may also be specified.
Netscape Comment	A comment that may be displayed when the certificate is viewed.
X509v3 Subject Key Identifier	Identifies the public key being certified. It enables distinct keys used by the same subject to be differentiated (for example, as key updating occurs).
X509v3 Authority Key Identifier	Identifies the public key to be used to verify the signature on this certificate or CRL. It enables distinct keys used by the same CA to be distinguished (for example, as key updating occurs).
Signature Algorithm	The name of the algorithm used for digital signatures (but not for key exchanges).
Hex Numbers	The actual signature of the certificate. The client can regenerate this signature using the specified algorithm to make sure that the certificate data has not been changed.

Showing SSL RSA Private Keys

Use the **show ssl associate rsakey** *keyname* command to obtain information about RSA private key associations in the CSS. You can optionally specify an RSA key name to view information about a specific RSA key association (key size and type). If you do not specify an RSA keyname, you see a list of all RSA key associations.



Note

When you view the contents of a specific key only, specifics on the key size and key type appears. This restriction occurs because the key contents are secure and should not be viewed.

To display information about all RSA private key associations:

```
(config) # show ssl associate rsakey
```

Table 7-3 describes the fields in the `show ssl associate rsakey` output.

Table 7-3 Field Descriptions for the `show ssl associate rsakey` Command

Field	Description
Key Name	The name of the RSA key association
File Name	The name of the file containing the RSA key pair
Used By List	Indicates if the RSA key association is used by the SSL proxy list containing the VIP address of the virtual server

To display information about a specific RSA key pair association, enter:

```
(config) # show ssl associate rsakey myrsakey1
1024-bit RSA keypair
```

Showing SSL DSA Private Keys

Use the `show ssl associate dsakey keyname` command to obtain information about DSA private key associations in the CSS. You can optionally specify a DSA key name to view information about a specific DSA key association (key size and type). If you do not specify a DSA keyname, you see a list of all DSA key associations.



Note

When you view the contents of a specific key only, specifics on the key size and key type appears. This restriction occurs because the key contents are secure and should not be viewed.

To display information about all DSA key associations, enter:

```
(config) # show ssl associate dsakey
```

Table 7-4 describes the fields in the `show ssl associate dsakey` output.

Table 7-4 Field Descriptions for the `show ssl associate dsakey` Command

Field	Description
Key Name	The name of the DSA key association
File Name	The name of the file containing the DSA key pair
Used By List	Indicates if the DSA key association is used by the SSL proxy list containing the VIP address of the virtual server

To display information about a specific DSA key pair association, enter:

```
(config) # show ssl associate dsakey mydsakey1
1024-bit DSA keypair
```

Showing SSL Diffie-Hellman Parameters

Use the `show ssl associate dhparam paramname` to obtain information about Diffie-Hellman parameters. You can optionally specify a parameter filename to view information about a specific Diffie-Hellman parameter file association. If you do not specify a Diffie-Hellman parameter filename, you see a list of all Diffie-Hellman parameter file associations.

To display information about all Diffie-Hellman associations:

```
(config) # show ssl associate dhparam
```

Table 7-5 describes the fields in the `show ssl associate dhparam` output.

Table 7-5 Field Descriptions for the `show ssl associate dhparam` Command

Field	Description
Parameter Name	The name of the Diffie-Hellman parameter association
File Name	The name of the file containing the Diffie-Hellman parameters

Table 7-5 *Field Descriptions for the show ssl associate dhparam Command (continued)*

Field	Description
Used By List	Indicates if the Diffie-Hellman file association is used by the SSL proxy list containing the VIP address of the virtual server

To display information about a specific Diffie-Hellman parameter file association, enter:

```
(config) # show ssl associate dhparam mydhparam1
512-bit DH parameters
```

Showing SSL Associations

Use the **show ssl associate** to display a summary of all certificate and key associations stored on the CSS.

To display a summary of SSL associations for the CSS, enter:

```
CSS11506(config)# show ssl associate

Certificate Name      File Name            Used by List
-----
rsacert              rsacert.pem         yes

RSA Key Name         File Name            Used by List
-----
rsakey              rsakey.pem          yes

DH Param Name       File Name            Used by List
-----
dhparams            dhparams.pem        no

DSA Key Name         File Name            Used by List
-----
dsakey              dsakey.pem          no
```

Showing SSL Certificates, Key Pairs, and Diffie-Hellman Parameter Files

Use the **show ssl files** to display a list of certificates, key pairs, and Diffie-Hellman parameter files loaded on the CSS.

For example, enter:

```
(config) # show ssl files
```

Table 7-6 describes the fields in the **show ssl files** output.

Table 7-6 *Field Descriptions for the show ssl files Command*

Field	Description
File Name	The name of the imported or manually-generated certificate, RSA key pair, DSA key pair, or Diffie-Hellman parameter file.
File Type	The format of the imported or manually-generated certificate, RSA key pair, DSA key pair, or Diffie-Hellman parameter file. File types can include DES-encoded, PEM-encoded, or PKCS#12-encoded.
File Size	The total size (in Kbytes) of the certificate, RSA key pair, DSA key pair, or Diffie-Hellman parameter file.

Showing SSL Proxy Configuration Information

Use the **show ssl-proxy-list** command to display information about SSL proxy lists. You can display general information about all SSL proxy lists or detailed information about a specific SSL proxy list.

Enter the **show ssl-proxy-list** commands from the specified command modes to display configuration information for an SSL proxy list:

- **show ssl-proxy-list:**
 - In `ssl-proxy-list` mode, this command displays detailed configuration information for the specified SSL proxy list.
 - In `global`, `content`, `owner`, `service`, `SuperUser`, and `User` modes, this command displays general configuration information for all existing SSL proxy lists.
- **show ssl-proxy-list [ssl-server|backend-server] {number}** - Displays detailed configuration information for the SSL proxy list and the virtual SSL servers or back-end servers in the list. Optionally, you can specify an SSL or back-end server number to display its configuration information. This command is available in `ssl-proxy-list` mode.
- **show ssl-proxy-list list_name** - Displays detailed configuration information for the specified SSL proxy list and all virtual SSL servers associated with the list. This command is available in `global`, `content`, `owner`, `service`, `SuperUser`, and `User` modes.
- **show ssl-proxy-list list_name [ssl-server|backend-server] {number}** - Displays detailed configuration information for the SSL proxy list and all virtual SSL servers or back-end servers in the list. Optionally, you can specify an SSL or back-end server number to display its configuration information. This command is available in `global`, `content`, `owner`, `service`, `SuperUser`, and `User` modes.

To view general information about all configured SSL proxy lists, enter:

```
# show ssl-proxy-list
```

Table 7-7 describes the fields in the **show ssl-proxy-list** output.

Table 7-7 Field Descriptions for the show ssl-proxy-list Command

Field	Description
Name	The name of the SSL proxy list
Description	The description for the SSL proxy list
State	The state of the SSL proxy list (active or suspended)
Services Associated	The number of services associated with the SSL proxy list
Rules Associated	The number of content rules associated with the SSL proxy list

For example, to display detailed configuration information about *ssl_list1* from the `ssl-proxy-list` mode, enter:

```
(config-ssl-proxy-list[ssl_list1])# show ssl-proxy-list
```

To display detailed configuration information about *ssl_list1* from global configuration mode, enter:

```
(config)# show ssl-proxy-list ssl_list1
```

Table 7-8 describes the fields in the **show ssl-proxy-list list_name** output.

Table 7-8 Field Descriptions for the show ssl-proxy-list Command

Field	Description
Description	The description for the SSL proxy list.
SSL-Server	
Number of SSL-Servers	The total number of virtual SSL servers specified for the SSL proxy list.
SSL-Server	A unique number for the virtual SSL server.
Number of Backend-Servers	The total number of back-end servers specified for the SSL proxy list.
Backend-server	A unique number for the back-end server.

Table 7-8 Field Descriptions for the `show ssl-proxy-list` Command (continued)

Field	Description
VIP Address	The VIP address for the virtual SSL or back-end server (corresponding to an SSL proxy list).
VIP Port	The virtual TCP port for the virtual SSL or back-end server (corresponding to an SSL proxy list).
Server Address	The circuit IP address of the back-end SSL server.
Server Port	The back-end SSL server port used for the SSL initiation connection.
Type	The type of SSL.
RSA Certificate	The name of the RSA certificate.
RSA Keypair	The name of the RSA key.
DSA Certificate	The name of the DSA certificate.
DSA Keypair	The name of the DSA key pair.
DH Param	The name of the Diffie-Hellman parameter association.
Client Authentication	State of client authentication on the virtual SSL server: enabled or disabled.
Client Authentication Failure	Configured method by which the CSS responds to a client certificate failure; ignore, redirect, or reject (default).
Authentication Redirect URL	URL used by the CSS to redirect a client connection when the client authentication failure method is configured to redirect.
CA Certificate	Name of the CA certificate imported on the CSS for client authentication.
CRL	CRL record name.
Session Cache Timeout	The period of time an SSL session ID remains valid before the CSS requires the full SSL handshake to establish a new SSL connection.
SSL Version	The specified SSL (version 3.0), TLS (version 1.0), or SSL and TLS protocol in use.

Table 7-8 Field Descriptions for the `show ssl-proxy-list` Command (continued)

Field	Description
Re-handshake Timeout	The period of time the CSS waits before initiating an SSL rehandshake message.
Re-handshake Data	The maximum amount of data to be exchanged between the CSS and the client, after which the CSS transmits the SSL handshake message and reestablishes the SSL session.
Virtual TCP Inactivity Timeout	The time period that the CSS waits before terminating a TCP connection with a client when there is little or no activity occurring on the connection.
Virtual TCP Syn Timeout	The time period that the CSS waits before terminating a TCP connection with a client that has not successfully completed the TCP three-way handshake with the CSS prior to transferring data.
Server TCP Inactivity Timeout	The time period that the CSS waits before terminating a TCP connection with a server when there is little or no activity occurring on the connection.
Server TCP Syn Timeout	The time period that the CSS waits before terminating a TCP connection with a server that has not successfully completed the TCP three-way handshake with the CSS prior to transferring data.
Cipher Suite(s)	The name of the cipher suite(s) assigned to the SSL content rule (see Table 4-1 for a list of all supported cipher suites and values for the specific SSL server).
Weight	The priority assigned to the cipher suite.
Port	The TCP port of the back-end content rule through which the back-end HTTP connections are sent.
Server	The VIP address of the back-end content rule through which the back-end HTTP connections are sent.
URL Rewrite Rule(s)	

Table 7-8 Field Descriptions for the `show ssl-proxy-list` Command (continued)

Field	Description
Number	The number of the URL rewrite rule in the SSL server.
Rule	The domain name of the URL to be redirected.
SSL Port	The port used for rewriting the HTTP Header Location field to contain an HTTPS location when the URL rewrite rule matches.
Clear Port	The port used for performing the URL rewrite rule match.
Server	The IP address assigned to the back-end content rule used with the cipher suite.
HTTP Header Insert Prefix	Configured prefix text string inserted in front of each client certificate, server certificate, and session field.
HTTP Header Insert	Type of field information inserted in the HTTP request header; Client Cert for client certificate, Server Cert for server certificate, and Session Data for SSL connection information. For information on the fields inserted in the header, see Chapter 4, Configuring SSL Termination .
HTTP Header Insert Static	Configured static text string inserted in the HTTP request header.

Showing CRL Record Configuration

Use the **show ssl crl-record** command to display the configuration for all Certificate Revocation List (CRL) records. Use the **show ssl crl-record name** command to display the configuration for a specific CRL record.



Note

To verify that a CRL downloaded successfully, view the output of the **show ssl statistics ssl** command and the CSS syslog messages. For information on the **show ssl statistics** command, see the [“Showing SSL Module Statistics”](#) section.

For example, to display the configuration for all CRL records, enter:

```
(config) # show ssl crl-record
```

[Table 7-9](#) describes the fields in the **show ssl crl-record** output.

Table 7-9 Field Descriptions for the *show ssl crl-record* Command

Field	Description
CRL Record	Configured name of the CRL record.
Signer Cert	Name of the CA certificate imported on the CSS. This certificate verifies that the CRL is from the CA.
Update Delay	How long the CSS waits before updating the CRL on the CSS.
CRL URL	URL where the CSS downloads the latest CRL.

Showing SSL URL Rewrite Statistics

Use the **show ssl urlrewrite** command to view the URL rewrite rule statistics for one or more SSL modules. This command displays statistics related to the number of flows received and evaluated by the SSL module, and the number of HTTP 300-series redirects found and then rewritten.

The syntax for this command is:

```
show ssl urlrewrite {slot number}
```

The **slot number** option displays URL rewrite statistics for a specific SSL module in the CSS 11503 or CSS 11506 chassis (assuming more than one module is installed). The valid slot entries are 2 and 3 (CSS 11503) or 2 to 6 (CSS 11506). If no slot number is specified, the **show ssl urlrewrite** command displays URL rewrite statistics for all SSL modules in the chassis.

For example, to view URL rewrite statistics for all SSL modules, enter:

```
# show ssl urlrewrite
```

For example, to view URL rewrite statistics for the SSL module in slot 5 of the CSS 11506, enter:

```
# show ssl urlrewrite slot 5
```

[Table 7-10](#) describes the fields in the **show ssl urlrewrite** output.

Table 7-10 Field Descriptions for the show ssl urlrewrite Command

Field	Description
Virtual	The VIP address for the virtual SSL server.
Port	The virtual TCP port for the virtual SSL server.
Searches	The total number of flows received from the back-end server and evaluated by the SSL module to search for the presence of HTTP 300-series redirects.

Table 7-10 Field Descriptions for the `show ssl urlrewrite` Command (continued)

Field	Description
Redirects Found	The total number of flows examined by the SSL module for which an HTTP 300-series redirect was detected.
Redirects Rewritten	The total number of flows examined by the SSL module for which an HTTP 300-series redirect was found matching one of the configured URL rewrite rules. This number represents the total number of redirects that have been rewritten for this VIP address.

Showing SSL Module Statistics

Use the `show ssl statistics` command to view the statistics for the cryptography components and client authentication on one or more SSL modules. If you do not specify any options for this command, SSL statistics appear for all SSL modules in the CSS chassis.

The syntax for this command is:

```
show ssl statistics {component} {slot number}
```

The options and variables are:

- *component* - Selects a specific component in the SSL module to display statistics. The components include:
 - **backend-session-cache** - Displays counter statistics for back-end SSL or SSL initiation, where the CSS is acting as a client.
 - **crypto** - Displays counter statistics for the cryptography chip
 - **session-cache** - Displays counter statistics for SSL termination, where the CSS is acting as an SSL server.
 - **ssl** - Displays counter statistics for the SSL server counter
 - **ssl-proxy-server** - Displays counter statistics for the SSL proxy list component that provides SSL termination in the SSL module

- **slot number** - Displays statistics for a component in a specific SSL module in the CSS chassis (assuming more than one module is installed). Specify **slot number** after each **show ssl statistics** command. The valid slot entries are 2 and 3 (CSS 11503) or 2 to 6 (CSS 11506). If no slot number is specified, the **show ssl statistics** command displays statistics for all installed SSL modules.

For example, to view all SSL statistics for the SSL module in slot 5 of the CSS chassis, enter:

```
# show ssl statistics slot 5
```

Table 7-11 describes the fields in the **show ssl statistics** output.

Table 7-11 Field Descriptions for the show ssl statistics Command

Field	Description
Component	Indicates the specific component in the SSL module for which statistics are displayed. The SSL statistic functions include: <ul style="list-style-type: none"> • ssl-proxy-server - Displays counter statistics for the SSL proxy list component that provides SSL termination in the SSL module • crypto - Displays counter statistics for the cryptography chip in the SSL module • ssl - Displays counter statistics for the SSL server counter
Slot	Indicates the slot number of the SSL module for which statistics are displayed. Valid slots are 2 (CSS 11501), 2 and 3 (CSS 11503), or 2 to 6 (CSS 11506).
SSL Proxy List Statistics	
Handshake started for incoming SSL connections	Number of times the handshake process was initiated for incoming SSL connections from a client to the SSL module.
Handshake completed for incoming SSL connections	Number of times the handshake process was completed for incoming SSL connections from a client to the SSL module.

Table 7-11 Field Descriptions for the show ssl statistics Command (continued)

Field	Description
Handshake started for outgoing SSL connections	Number of times the handshake process was initiated for outgoing SSL connections from the SSL module to a client.
Handshake completed for outgoing SSL connections	Number of times the handshake process was completed for outgoing SSL connections from a client to the SSL module.
HTTP header insert of session data	Number of times that the CSS inserted SSL connection data information in the HTTP request header to a back-end server.
HTTP header insert of client certificate data	Number of times that the CSS inserted client certificate information in the HTTP request header to a back-end server.
HTTP header insert of server certificate data	Number of times that the CSS inserted server certificate information in the HTTP request header to a back-end server.
HTTP header insert of user defined prefix	Number of times that the CSS inserted the prefix field in the HTTP request header to a back-end server.
HTTP header insert of static phrase	Number of times that the CSS inserted the configured static text in the HTTP request header to a back-end server.
Active SSL flows high water mark	Maximum number of active SSL flows on the CSS.
Crypto Statistics	
RSA Private	Number of RSA private key calculations requested.
RSA Public	Number of RSA public key calculations requested.
DH Shared	Number of Diffie-Hellman shared secret key calculations requested.
DH Public	Number of Diffie-Hellman public key calculations requested.
DSA Sign	Number of DSA signings requested.
DSA Verify	Number of DSA verifications requested.

Table 7-11 Field Descriptions for the `show ssl statistics` Command (continued)

Field	Description
SSL MAC	Number of SSL MAC calculations requested.
TLS HMAC	Number of TLS HMAC calculations requested.
3DES	Number of 3 DES calculations requested.
ARC4	Number of ARC4 calculations requested.
HASH	Number of pure hash calculations requested.
RSA Private Failed	Number of RSA private key calculations that failed.
RSA Public Failed	Number of RSA public key calculations that failed.
DH Shared Failed	Number of Diffie-Hellman shared secret key calculations that failed.
DH Public Failed	Number of Diffie-Hellman public key calculations that failed.
DSA Sign Failed	Number of DSA signings that failed.
DSA Verify Failed	Number of DSA verifications that failed.
SSL MAC Failed	Number of SSL MAC calculations that failed.
TLS HMAC Failed	Number of TLS HMAC calculations that failed.
3DES Failed	Number of 3 DES calculations that failed.
ARC4 Failed	Number of ARC4 calculations that failed.
HASH Failed	Number of pure hash calculations that failed.
Hardware Device Not Found	Number of times that a call was made to the cryptography hardware and no hardware acceleration device was available.
Hardware Device Timed Out	Number of times the cryptography hardware did not complete an acceleration request within the specified time. This function is not currently implemented. This counter should always be 0.

Table 7-11 Field Descriptions for the show ssl statistics Command (continued)

Field	Description
Invalid Crypto Parameter	Number of times a hardware acceleration function was requested with an invalid parameter from the CSS. Invalid parameters include an invalid bit length for the operation, a buffer that is not a multiple of 4 bytes in length, a buffer that does not begin on an even 4-byte boundary, requesting an operation on a buffer with too many fragments or too few fragments (such as with no input), or requesting an illegal (nonsense) function.
Hardware Device Failed	Number of times the hardware acceleration device failed. This counter only increments on a DMA error.
Hardware Device Busy	Number of times the hardware acceleration device was busy and could not accept an acceleration request.
Out Of Resources	Number of times no hardware buffers were available and the cryptography hardware could not accept an acceleration request.
Cancelled -- Device Reset	Number of cancelled status returns due to a CSS reboot.
SSL Statistics	
RSA Private Decrypt calls	Number of RSA private decryption calls.
RSA Public Decrypt calls	Number of RSA public encryption calls.
DH Compute key calls	Number of Diffie-Hellman Compute key calls.
DH Generate key calls	Number of Diffie-Hellman Generate key calls.
DSA Verify calls	Number of DSA Verifications calls.
DSA Sign calls	Number of DSA Signing calls.
MD5 raw hash calls	Number of MD5 pure hash calls.
SHA1 raw hash calls	Number of SHA1 pure hash calls.
3-DES calls	Number of 3-DES calls.

Table 7-11 Field Descriptions for the `show ssl statistics` Command (continued)

Field	Description
RC4 calls	Number of RC4 calls.
SSL MAC (MD5) calls	Number of SSL Message Authentication Code (MAC) computations using MD5 algorithm.
SSL MAC (SHA1) calls	Number of SSL MAC computations using SHA algorithm.
TLS MAC (MD5) calls	Number of TLS MAC computations using MD5 algorithm.
TLS MAC (SHA1) calls	Number of TLS MAC computations using SHA algorithm.
Level 1 Alerts Received	Number of Level 1 alerts received.
Level 2 Alerts Received	Number of Level 2 alerts received.
Level 1 Alerts Sent	Number of Level 1 alerts transmitted.
Level 2 Alerts Sent	Number of Level 2 alerts transmitted.
SSL received bytes from TCP	Number of bytes SSL received from TCP.
SSL transmitted bytes to TCP	Number of bytes SSL transmitted to TCP.
SSL received Application Data bytes	Number of Application Data bytes received by the SSL module.
SSL transmitted Application Data bytes	Number of Application Data bytes transmitted by the SSL module.
SSL received non-application data bytes	Number of non-application data (handshake, alert, and change cipher) bytes received by the SSL module.
SSL transmitted non-application data bytes	Number of non-application data (handshake, alert, and change cipher) bytes transmitted by the SSL module.
RSA Private Decrypt failures	Number of RSA Private Decrypt calls that failed.
MAC failures for packets received	Number of times the MAC could not be verified for the incoming SSL messages.

Table 7-11 Field Descriptions for the show ssl statistics Command (continued)

Field	Description
Rehandshake TimerAlloc failed	Number of times the SSL module was unable to allocate the Rehandshake Timer.
Successful client authentications	Number of times that the CSS successfully authenticated a client certificate.
Client authentication failures	Number of times that the CSS could not authenticate a client certificate.
Unknown issuer certificates	Number of times that the CSS could not identify the issuer of a client certificate.
Signature unable to decrypt	Number of times that the CSS could not decrypt the signature on a client certificate.
Invalid issuer keys	Number of times that the CSS identified an invalid key of a client certificate.
Not yet valid certificate	Number of times that the CSS received a certificate that had not been validated by a CA at that time.
Expired certificates	Number of times that the CSS received a certificate with an expired time stamp.
Revoked certificate	Number of times that the CSS received a client certificate revoked by the issuer.
CRLs not obtained from host	A timeout occurred when the CSS tried to obtain a CRL from a host.
CRLs obtained but failed to load	The CSS successfully obtained the CRL but the CRL failed to load.
CRLs with invalid signatures	Number of times that the CSS could not validate the signer of the CRL with the signer certificate on the CSS.
CRL out of memory error	Number of times that the SSL module was out of memory and could not store the CRL. When a CRL cannot be stored in memory, all incoming client authentications will fail.
Session Cache Statistics	
Handshakes Accepted from Client	Number of handshakes that the SSL module accepted from clients.

Table 7-11 Field Descriptions for the show ssl statistics Command (continued)

Field	Description
Handshakes Renegotiated	Number of handshakes that the SSL module had to renegotiate.
Handshakes Completed	Number of successful handshakes that the SSL module completed with clients.
Session ID Misses	Number of session IDs offered by peers and looked up in the cache, but not found.
Session ID Timeouts	Number of cached sessions that reached their timeout limit and expired.
Session Cache Full	Number of times the cache was full.
Session ID Hits	Number of session IDs offered by peers that the SSL module found in its cache.
Total Number of Items Cached	Total number of sessions in the cache.
Backend Session Cache Statistics	
Handshakes Sourced to Server	Number of handshakes that the SSL module offered to servers.
Handshakes Renegotiated	Number of handshakes that the SSL module had to renegotiate.
Handshakes Completed	Number of successful handshakes that the SSL module completed with servers.
Session ID Misses	Number of times that there was not an existing valid session ID to send to the server.
Session ID Timeouts	Number of cached sessions that reached their timeout limit and expired.
Session Cache Full	Number of times that the cache was full.
Session ID Hits	Number of times that there was a valid session ID to offer to the server.
Total Number of Items Cached	Total number of sessions in the cache.

Clearing SSL Statistics

Use the **clear ssl statistics** command to clear the SSL statistics counters for all SSL modules in the CSS chassis. The reset statistics appear as 0 in the **show ssl statistics** display.

To clear SSL statistics counters for a specific module, use the **clear ssl statistics** command and specify the **slot number** following the command. The valid slot entries are 2 and 3 (CSS 11503) or 2 to 6 (CSS 11506).

To clear the SSL statistics counter, enter:

```
# clear ssl statistics
```

Showing SSL Flows

Use the **show ssl flows** command to display information about the active flows for each VIP address, port, and SSL module. The output displays TCP proxy flows, active SSL flows (a subset of TCP proxy flows), and SSL flows occurring during the handshake phase of the protocol (a subset of active SSL flows).

The syntax for this command is:

```
show ssl flows {slot number}
```

The **slot number** option displays information about the active flows for a specific SSL module in the CSS chassis (assuming more than one module is installed). The valid slot entries are 2 and 3 (CSS 11503) or 2 to 6 (CSS 11506). If no slot number is specified, the **show ssl flows** command displays statistics for all installed SSL modules.

To view SSL flows for all SSL modules in the CSS, enter:

```
# show ssl flows
```

To view SSL flows for a specific SSL module in the CSS chassis (for example, installed in slot 5), enter:

```
# show ssl flows slot 5
```

Table 7-12 describes the fields in the **show ssl flows** output.

Table 7-12 Field Descriptions for the show ssl flows Command

Field	Description
SSL Acceleration Flows for Slot	The slot number of the SSL module for which flows are displayed. Valid slots are 2 (CSS 11501), 2 and 3 (CSS 11503), or 2 to 6 (CSS 11506).
Virtual	Virtual address of the ssl-server.
Port	Virtual TCP port of the ssl-server.
TCP Proxy Flows	Number of TCP connections that are currently being proxied through the SSL virtual IP address. These connections could either be in: <ul style="list-style-type: none"> • The TCP handshake or teardown phase and, therefore, not carrying any SSL traffic • The Established TCP phase and carrying SSL traffic
Active SSL Flows	Current number of TCP Proxy Flows that are carrying active SSL connections. These flows are the Established TCP connections in which an SSL Client Hello message has been received by the CSS. The SSL flows remain in this active state until the teardown process is initiated, either by sending or receiving an SSL Alert message. The Active SSL Flows number is a subset of the TCP Proxy Flows column.
SSL Flows in Handshake	The current number of Active SSL Flows that are in the handshake phase of the SSL protocol but are not yet sending data. This means that an SSL Client Hello message has been received by the CSS but the final finished message still has not been sent. The SSL Flows in Handshake number is a subset of the Active SSL Flows column.

■ Showing SSL Flows



Examples of CSS SSL Configurations

This chapter describes the SSL flow process with the SSL module and includes example proxy configurations. Each configuration section includes a running-configuration example and an accompanying illustration.

This section covers:

- [Processing of SSL Flows by the SSL Module](#)
- [SSL Transparent Proxy Configuration — One SSL Module](#)
- [SSL Transparent Proxy Configuration — Two SSL Modules](#)
- [SSL Transparent Proxy Configuration — HTTP and Back-End SSL Servers](#)
- [SSL Full Proxy Configuration — One SSL Module](#)
- [SSL Initiation Configurations](#)

Processing of SSL Flows by the SSL Module

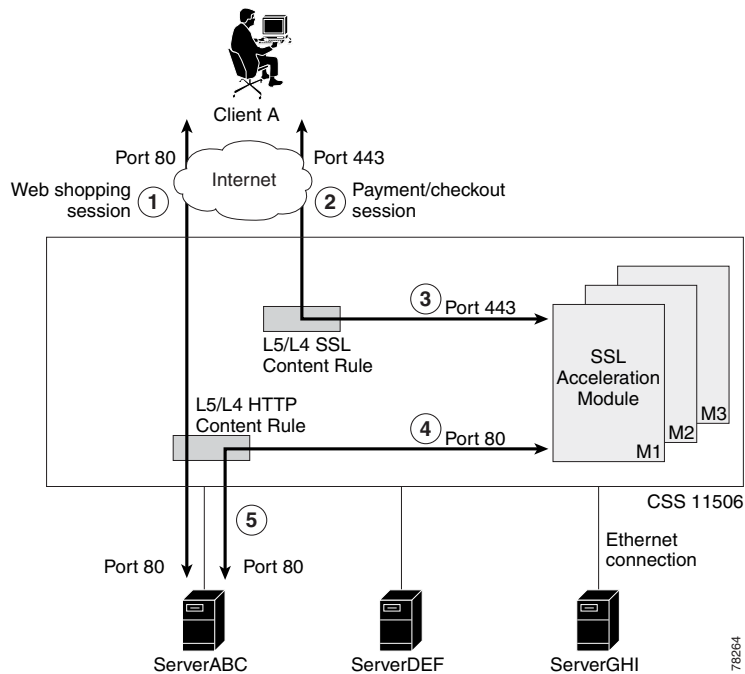
To terminate SSL flows, the SSL module functions as a proxy server, which means that it is the TCP endpoint for inbound SSL traffic. The SSL module maintains a separate TCP connection for each side of the communications, the client side and the server side. The proxy server can perform both TCP and SSL handshakes.

The following example is intended as an overview on the flow process; how the CSS and SSL module translate flows from HTTPS-to-HTTP for inbound packets and from HTTP-to-HTTPS for outbound packets.

Figure 8-1 illustrates a CSS with three SSL modules (M1, M2, and M3) configured to off load the SSL traffic from the back-end servers (ServerABC, ServerDEF, and ServerGHI). Figure 8-1 also shows the CSS maintaining a consistent stickiness between HTTP and SSL connections from the same client.

1. In a normal Web shopping-cart application, a transaction consists of multiple HTTP connections for shopping or browsing, and a few SSL connections for the final order placement and payment checkout sequence. The client must remain stuck to the same server that holds the customer's database information during the entire transaction. During the initial HTTP connections from a client to a server, the client is stuck to a server by using Layer 5 HTTP cookies or a URL content rule. At checkout, the client transitions to SSL connections.

Figure 8-1 CSS Configuration with Multiple SSL Modules



78264

2. The client transmits the encrypted payment or order information through an SSL connection (TCP SYN received through destination port 443). In this example, when the client connection reaches the CSS, the CSS uses a Layer 5 SSL Session ID sticky content rule to load balance the SSL connection among the three SSL modules (M1, M2, and M3). When the inbound TCP SYN connection reaches the SSL module (the SSL server), it terminates the TCP connections from the client.
3. Once an SSL module is selected (for example, M1), the CSS forwards the SSL packet to that module. The Session ID is saved in the sticky table for subsequent SSL connections from the same client. Once this SSL flow is mapped, the CSS forwards all subsequent packets for this connection to SSL module M1. If there are additional SSL connections associated with this transaction (as determined by the SSL Session ID), the CSS also forwards and maps the packets to SSL module M1.
4. The SSL module terminates the SSL connection and decrypts the packet data. The SSL module then initiates an HTTP connection to a content rule configured on the CSS. The data in this HTTP connection is clear text.
5. The HTTP content rule uses the Layer 5 HTTP cookies or URL sticky content rule on this HTTP request. The cookie or URL string in this clear text HTTP request is used to locate the same server (ServerABC) as the one initially used by the non-SSL HTTP connection in the transactions (for example, online shopping). The CSS forwards the request to ServerABC and maps this flow. Once the flow is mapped, the return HTTP response from the server is sent to the same SSL module (M1) that sent the original request. The SSL module encrypts the response as an SSL packet (it translates flows from HTTP-to-HTTPS for outbound packets) and sends the packets back to the client through the correct SSL connection.

When the TCP connection is finished, the four flows (the two flows between the client and SSL module, and the two flows between the SSL module and the Server) are torn down.

An entire SSL session can comprise Multiple TCP connections. For each of those connections, the same process takes place among the client, SSL module, and server. The SSL Session ID maintains the stickiness between the client and the SSL module and the cookie maintains the stickiness between the SSL module and the servers. In this way, stickiness can be maintained consistently through the entire web transaction.

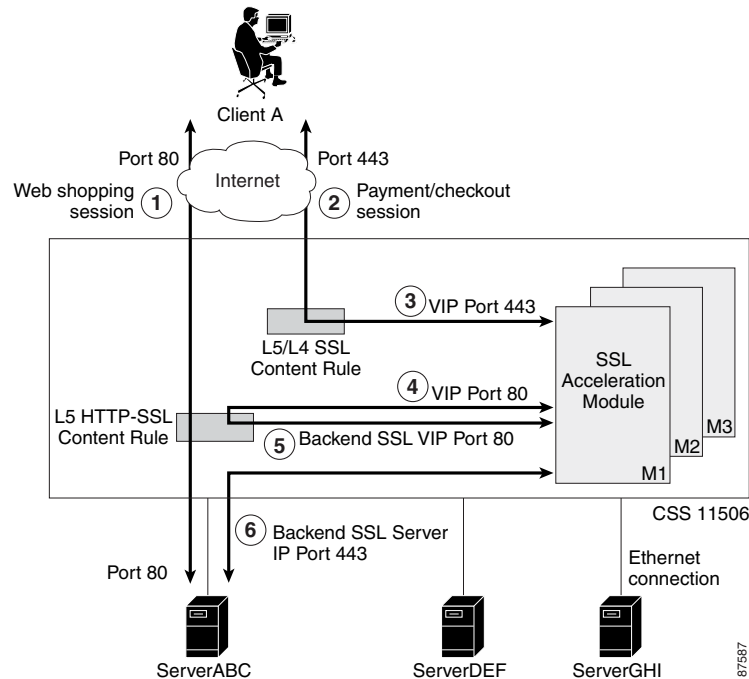
**Note**

By default, the SSL session cache for the SSL module can hold 10000 sessions. The cache size is the maximum number of SSL session IDs that can be stored in a dedicated session cache on an SSL module. If necessary for your SSL service, use the **session-cache-size** command to reconfigure the size of the SSL session ID cache for the SSL service.

The back-end session ID cache is 4096 entries and is not configurable.

When you configure a back-end SSL server on the CSS (Figure 8-2), flow processing from the client to CSS is the same as steps 1 through 4 in the previous example. However in step 4 shown in Figure 8-2, the SSL module initiates an HTTP connection to an HTTP-SSL content rule with services to a back-end SSL server.

Figure 8-2 CSS Configuration with a Back-End SSL Server



In step 5 shown in [Figure 8-2](#), the CSS directs the clear text traffic back to the SSL module through an IP address that maps directly to a back-end SSL server. The SSL module terminates the clear text connection.

In step 6 of [Figure 8-2](#), the SSL module re-encrypts the traffic and establishes an SSL connection to the back-end SSL server. The SSL module sends the traffic through the CSS to the selected back-end SSL server.

SSL Transparent Proxy Configuration – One SSL Module

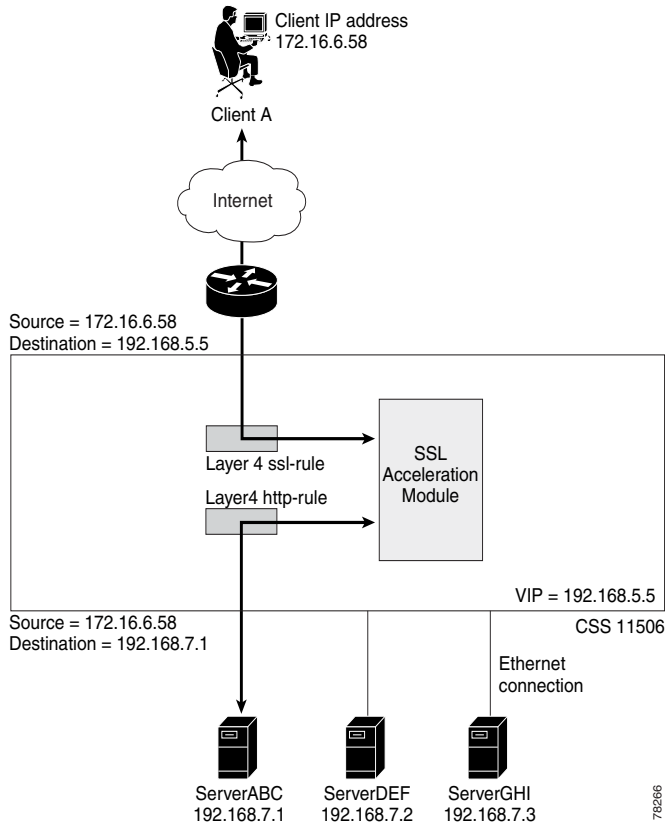
An SSL transparent proxy server is a proxy server that preserves the client's IP address as the source IP address for the back-end connection to the server. When you configure an SSL transparent proxy on the CSS, the CSS intercepts and redirects outbound client requests to an HTTP server on the network without changing the source IP address.

This section provides a simple configuration of an SSL transparent proxy between a client, a CSS with a single SSL module, and three HTTP servers (ServerABC, ServerDEF, and ServerGHI). Two content rules are used in this configuration, an SSL content rule and a HTTP content rule. The SSL content rule is for Layer 4 because there is only a single SSL module and there is no need to maintain client-to-server (SSL) stickiness. The use of a Layer 4 content rule in this configuration may improve CSS performance.

[Figure 8-3](#) illustrates this transparent proxy configuration.

For purposes of illustration, the configuration example in [Figure 8-3](#) shows the VIP address for the SSL content rule (ssl-rule) to be the same as the VIP address for the HTTP content rule (http-rule). These two VIP addresses do not have to be identical. Depending on the method that you choose to allow access to secure content on your HTTP servers, you may require specification of a different VIP address for the clear-text content rule to place it in non-routable address space. In this example, instead of specifying a VIP address of 192.168.5.5 for the http-rule content rule, you could specify a VIP address of 10.1.1.5. The clear-text http-rule will be unreachable from the Internet, which can offer you more flexibility and granularity while allowing the CSS to be seamlessly integrated for secure transactions.

Figure 8-3 Transparent Proxy Configuration with a Single SSL Module



```
! ***** GLOBAL *****
logging commands enable

ssl associate dsakey dsakey dsakey.pem
ssl associate dhparam dhparams dhparams.pem
ssl associate rsakey rsakey rsakey.pem
ssl associate cert rsacert rsacert.pem

ftp-record ssl_record 161.44.174.127 anonymous des-password
deye2gtcld1b6feeebabfcbfagyezc5f /
```

```
!***** CIRCUIT *****
circuit VLAN1

    ip address 192.168.8.254 255.255.255.0

circuit VLAN2

    ip address 192.168.7.254 255.255.255.0

!***** SSL PROXY LIST *****
ssl-proxy-list test
    ssl-server 111
    ssl-server 111 vip address 192.168.5.5
    ssl-server 111 port 443
    ssl-server 111 rsacert rsacert
    ssl-server 111 rsakey rsakey
    ssl-server 111 cipher rsa-with-rc4-128-md5 192.168.5.5 80
    active

!***** SERVICE *****
service ssl_module1
    type ssl-accel
    keepalive type none
    slot 5
    add ssl-proxy-list test
    active

service serverABC
    ip address 192.168.7.1
    protocol tcp
    port 80
    keepalive type http
    active

service serverDEF
    ip address 192.168.7.2
    protocol tcp
    port 80
    keepalive type http
    active

service serverGHI
    ip address 192.168.7.3
    protocol tcp
    port 80
    keepalive type http
    active
```

```

!***** OWNER *****
owner ap.com
content ssl-rule
  vip address 192.168.5.5
  protocol tcp
  port 443
  add service ssl_module1
  active

content http-rule
  vip address 192.168.5.5
  protocol tcp
  port 80
  add service serverABC
  add service serverDEF
  add service serverGHI
  advanced-balance cookies
  active

```

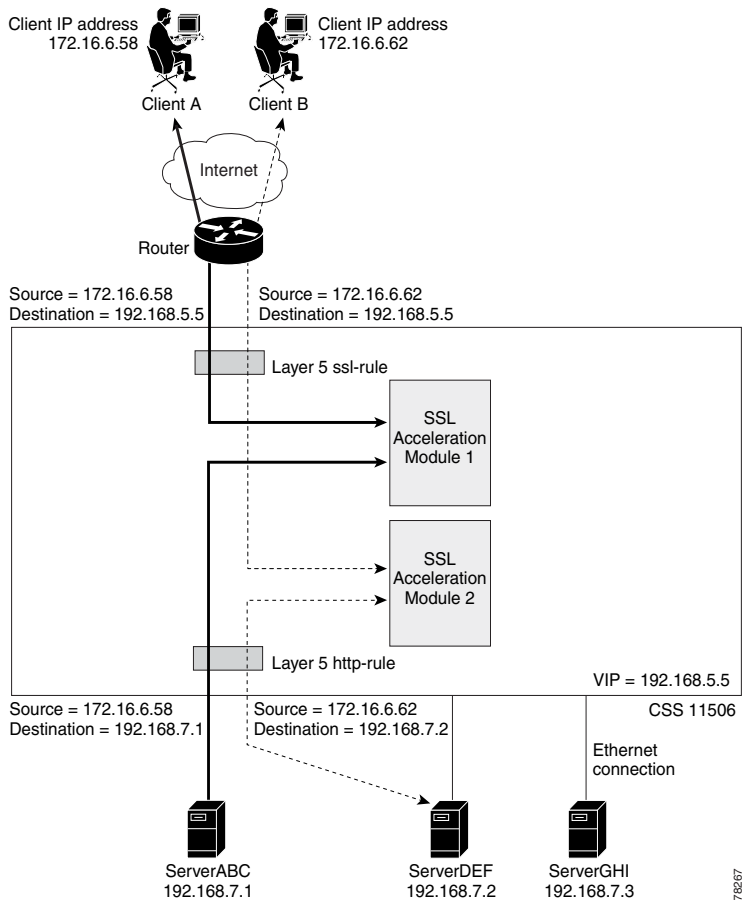
SSL Transparent Proxy Configuration — Two SSL Modules

This section provides an example configuration for an SSL transparent proxy between a client, a CSS with two SSL modules, and three HTTP servers (ServerABC, ServerDEF, and ServerGHI). A Layer 5 SSL sticky content rule is used in the configuration to maintain stickiness of the client to a particular SSL module. The Layer 5 SSL sticky content rule ensures SSL session ID reuse to eliminate the rehandshake process (which speeds up the SSL negotiation process) and to increase overall performance.

[Figure 8-4](#) illustrates this transparent proxy configuration.

For purposes of illustration, the configuration example in [Figure 8-4](#) shows the VIP address for the SSL content rule (ssl-rule) to be the same as the VIP address for the HTTP content rule (http-rule). These two VIP addresses do not have to be identical. Depending on the method that you choose to allow access to secure content on your HTTP servers, you may require specification of a different VIP address for the clear-text content rule to place it in nonroutable address space. In this example, instead of specifying a VIP address of 192.168.5.5 for the http-rule content rule, you could specify a VIP address of 10.1.1.5. The clear-text http-rule will be unreachable from the Internet, which can offer you more flexibility and granularity while allowing the CSS to be seamlessly integrated for secure transactions.

Figure 8-4 Transparent Proxy Configuration with Two SSL Modules



```

! ***** GLOBAL *****
logging commands enable

ssl associate dsakey dsakey dsakey.pem
ssl associate rsakey rsakey rsakey.pem
ssl associate cert rsacert rsacert.pem
ssl associate dhparam dhparams dhparams.pem

ftp-record ssl_record 161.44.174.127 anonymous des-password
deye2gtcld1b6feeebabfbcfagyezc5f /

```

```

!***** CIRCUIT *****
circuit VLAN1

    ip address 192.168.8.254 255.255.255.0

circuit VLAN2

    ip address 192.168.7.254 255.255.255.0

!***** SSL PROXY LIST *****
ssl-proxy-list test
    ssl-server 111
    ssl-server 111 vip address 192.168.5.5
    ssl-server 111 rsacert rsacert
    ssl-server 111 rsakey rsakey
    ssl-server 111 cipher rsa-with-rc4-128-md5 192.168.5.5 80
    ssl-server 111 port 443
    active

!***** SERVICE *****
service ssl_module1
    type ssl-accel
    keepalive type none
    slot 5
    add ssl-proxy-list test
    active

service ssl_module2
    type ssl-accel
    keepalive type none
    slot 6
    add ssl-proxy-list test
    active

service serverABC
    ip address 192.168.7.1
    protocol tcp
    port 80
    keepalive type http
    active

service serverDEF
    ip address 192.168.7.2
    protocol tcp
    port 80
    keepalive type http
    active

```

```
service serverGHI
  ip address 192.14.7.3
  protocol tcp
  port 80
  keepalive type http
  active

!***** OWNER *****
owner ap.com

content ssl-rule
  vip address 192.168.5.5
  protocol tcp
  port 443
  add service ssl_module1
  add service ssl_module2
  application ssl
  advanced-balance ssl
  active

content http-rule
  vip address 192.168.5.5
  protocol tcp
  port 80
  url "/"
  add service serverABC
  add service serverDEF
  add service serverGHI
  advanced-balance cookies
  active
```

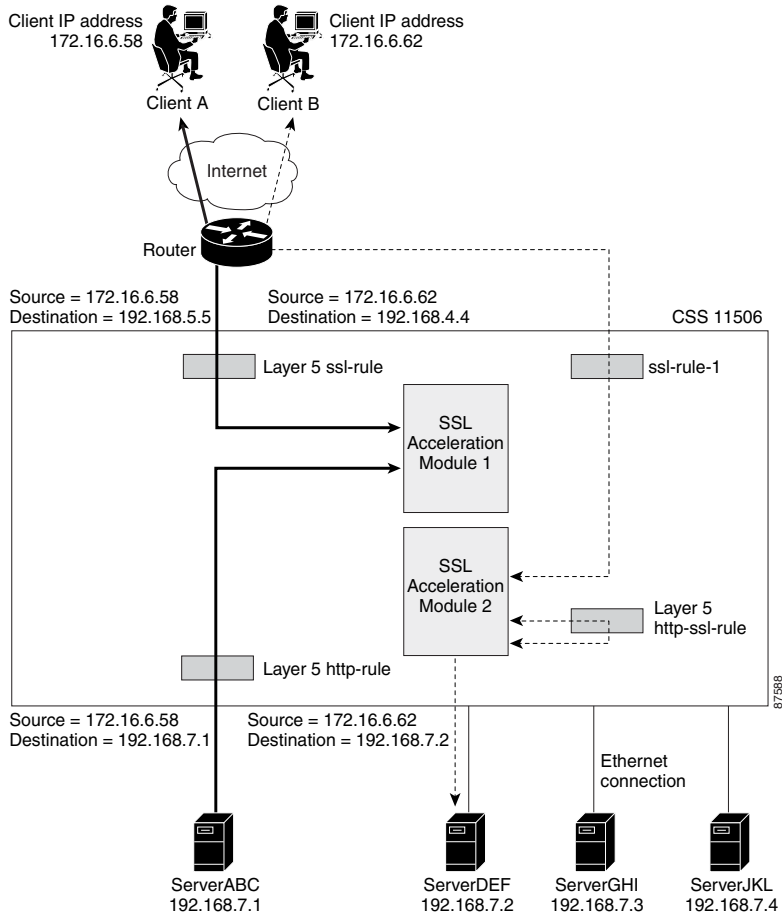
SSL Transparent Proxy Configuration — HTTP and Back-End SSL Servers

This section provides an example configuration for an SSL transparent proxy for two clients, a CSS with two SSL modules, two HTTP servers (ServerABC and ServerGHI), and two back-end SSL servers (ServerDEF and ServerJKL). This configuration is similar to the previous configuration. (See the [“SSL Transparent Proxy Configuration — Two SSL Modules”](#) section.) However, this example includes the configuration for a back-end SSL server.

In [Figure 8-5](#), Client A’s SSL connection has a destination address 192.168.5.5 that matches content rule `ssl-rule`. The CSS load balances the SSL connection to SSL module 1. The module terminates the connection, decrypts the data to clear text and initiates an HTTP connection to content rule `http-rule`. The CSS forwards the request to HTTP server ServerABC.

Client B’s SSL connection has a destination address 192.28.4.4 that matches content rule `ssl-rule-1`. The CSS load balances the SSL connection to SSL module 2. The module terminates the connection, decrypts the data to clear text and initiates an HTTP connection to content rule `http-ssl-rule`. The CSS directs the clear text data back to SSL module 2. The module terminates the connection, re-encrypts the traffic, and establishes an SSL connection to SSL server ServerDEF.

Figure 8-5 SSL Transparent Proxy Configuration - HTTP and Back-End SSL Servers



87598

The following configuration includes commands containing default values that do not appear in the running configuration. To identify these commands, they appear in *italic*.

```

!***** GLOBAL *****
 logging commands enable

 ssl associate dsakey dsakey dsakey.pem
 ssl associate rsakey rsakey rsakey.pem
 ssl associate cert rsacert rsacert.pem
 ssl associate dhparam dhparams dhparams.pem

 ftp-record ssl_record 161.44.174.127 anonymous des-password
 deye2gtcldlb6feeebabfcfagyezc5f /
!***** CIRCUIT *****
circuit VLAN1

 ip address 192.168.8.254 255.255.255.0

circuit VLAN2

 ip address 192.168.7.254 255.255.255.0

!***** SSL PROXY LIST *****
ssl-proxy-list test
  ssl-server 111
  ssl-server 111 vip address 192.168.5.5
  ssl-server 111 port 443
  ssl-server 111 rsacert rsacert
  ssl-server 111 rsakey rsakey
  ssl-server 111 cipher rsa-with-rc4-128-md5 192.168.5.5 80
  active

  ssl-server 2
  ssl-server 2 vip address 192.28.4.4
  ssl-server 2 port 443
  ssl-server 2 rsacert rsacert
  ssl-server 2 rsakey rsakey
  ssl-server 2 cipher rsa-with-rc4-128-md5 192.28.4.4 8080
  active

  backend-server 3
  backend-server 3 ip address 192.168.7.2
  backend-server 3 port 8080
  backend-server 3 server-ip 192.168.7.2
  backend-server 3 rsacert rsacert
  active

```

```
backend-server 4
backend-server 4 ip address 192.168.7.4
backend-server 4 port 8080
backend-server 4 server-ip 192.168.7.4
backend-server 4 rsacert rsacert
active

!***** SERVICE *****
service ssl_module1
  type ssl-accel
  keepalive type none
  slot 5
  add ssl-proxy-list test
  active

service ssl_module2
  type ssl-accel
  keepalive type none
  slot 6
  add ssl-proxy-list test
  active

service serverABC
  ip address 192.168.7.1
  protocol tcp
  port 80
  keepalive type http
  active

service serverDEF
  type ssl-accel-backend
  ip address 192.168.7.2
  protocol tcp
  keepalive type ssl
  keepalive port 443
  add ssl-proxy-list test
  active

service serverGHI
  ip address 192.14.7.3
  protocol tcp
  port 80
  keepalive type http
  active
```

```
service serverJKL
  type ssl-accel-backend
  ip address 192.168.7.4
  protocol tcp
  keepalive type ssl
  keepalive port 443
  add ssl-proxy-list test
  active

!***** OWNER *****
owner ap.com

content ssl-rule
  vip address 192.168.5.5
  protocol tcp
  port 443
  add service ssl_module1
  add service ssl_module2
  application ssl
  advanced-balance ssl
  active

content http-rule
  vip address 192.168.5.5
  protocol tcp
  port 80
  url "/*"
  add service serverABC
  add service serverGHI
  advanced-balance cookies
  active

content ssl-rule-1
  vip address 192.28.4.4
  protocol tcp
  port 443
  add service ssl_module1
  add service ssl_module2
  application ssl
  advanced-balance ssl
  active
```

```
content http-ssl-rule
  vip address 192.28.4.4
  protocol tcp
  port 8080
  url "/*"
  add service serverDEF
  add service serverJKL
  advanced-balance arrowpoint-cookie
  active
```

SSL Full Proxy Configuration — One SSL Module

An SSL full proxy server is a proxy server that terminates the client's SSL connections and initiates the back-end connection to the HTTP server using a different source IP address than that of the client. This configuration does not preserve the client's IP address for the back-end connection to the HTTP server.

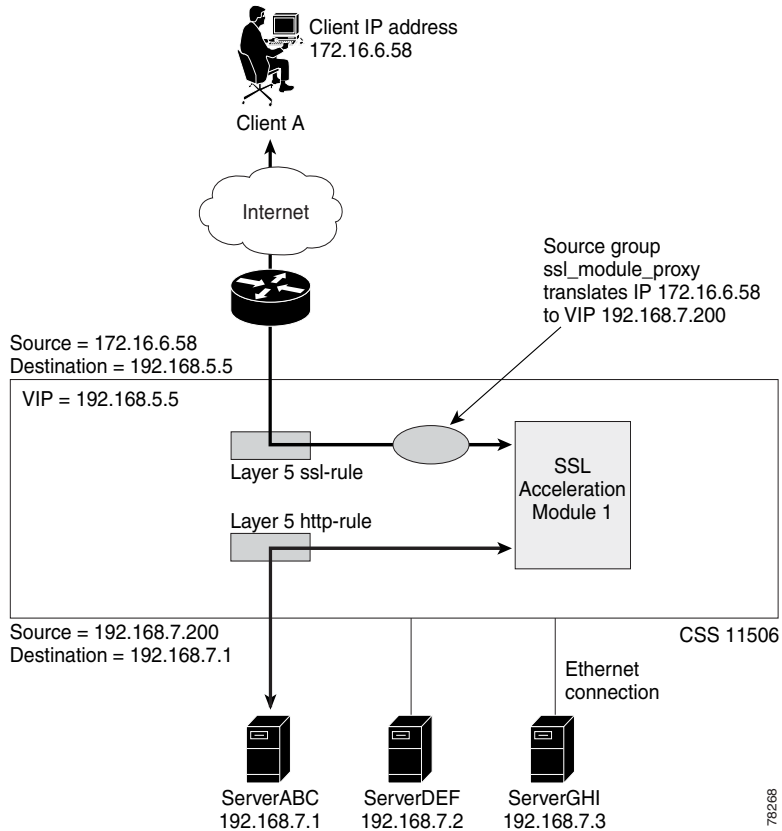
This section provides an example configuration for an SSL full proxy between a client, a CSS with a single SSL module, and three HTTP servers (ServerABC, ServerDEF, and ServerGHI). A Layer 5 sticky content rule is used in the configuration. For the CSS to implement a full proxy configuration with an SSL module, the configuration includes a source group that is used to isolate the SSL module traffic and to NAT its source address.

[Figure 8-6](#) illustrates this full proxy configuration.

For purposes of illustration, the configuration example in [Figure 8-6](#) shows the VIP address for the SSL content rule (ssl-rule) to be the same as the VIP address for the HTTP content rule (http-rule). These two VIP addresses do not have to be identical. Depending on the method that you choose to allow access to secure content on your HTTP servers, you may require specification of a different VIP address for the clear-text content rule to place it in nonroutable address space.

In this example, instead of specifying a VIP address of 192.168.5.5 for the http-rule content rule, you could specify a VIP address of 10.1.1.5. The clear-text http-rule will be unreachable from the Internet, which can offer you more flexibility and granularity while allowing the CSS to be seamlessly integrated for secure transactions.

Figure 8-6 Full Proxy Configuration Using a Single SSL Module



```

! ***** GLOBAL *****
logging commands enable

ssl associate dsakey dsakey dsakey.pem
ssl associate dhparams dhparams dhparams.pem
ssl associate rsakey rsakey rsakey.pem
ssl associate cert rsacert rsacert.pem

ftp-record ssl_record 161.44.174.127 anonymous des-password
deye2gtcld1b6feeebabfcbfagyezc5f /

```

```
!***** CIRCUIT *****
circuit VLAN1

    ip address 192.168.8.254 255.255.255.0

circuit VLAN2

    ip address 192.168.7.254 255.255.255.0

!***** SSL PROXY LIST *****
ssl-proxy-list test
    ssl-server 111
    ssl-server 111 vip address 192.168.5.5
    ssl-server 111 port 443
    ssl-server 111 rsacert rsacert
    ssl-server 111 rsakey rsakey
    ssl-server 111 cipher rsa-with-rc4-128-md5 192.168.5.5 80
    active

!***** SERVICE *****
service ssl_module1
    type ssl-accel
    keepalive type none
    slot 5
    add ssl-proxy-list test
    active

service ssl_module2
    type ssl-accel
    keepalive type none
    slot 6
    add ssl-proxy-list test
    active

service serverABC
    ip address 192.168.7.1
    protocol tcp
    port 80
    keepalive type http
    active

service serverDEF
    ip address 192.168.7.2
    protocol tcp
    port 80
    keepalive type http
    active
```

```
service serverGHI
  ip address 192.168.7.3
  protocol tcp
  port 80
  keepalive type http
  active

!***** OWNER *****
owner ap.com

content ssl-rule
  vip address 192.168.5.5
  protocol tcp
  port 443
  add service ssl_module1
  add service ssl_module2
  application ssl
  advanced-balance ssl
  active

content http-rule
  vip address 192.168.5.5
  protocol tcp
  port 80
  url "/"
  add service serverABC
  add service serverDEF
  add service serverGHI
  advanced-balance cookies
  active

!***** GROUP *****
group ssl_module_proxy
  add destination service serverABC
  add destination service serverDEF
  add destination service serverGHI
  vip address 192.168.7.200
  active
```

SSL Initiation Configurations

SSL initiation is the process whereby a properly configured CSS with an SSL module receives clear text from a client and connects that flow with an SSL flow that is originated by a back-end server configured on the SSL module. Use this configuration for secure site-to-site data transfers.

This section provides two SSL initiation example:

- [SSL Tunnel to Four Data Centers](#)
- [SSL Tunnel to One Data Center with Server Authentication](#)

SSL Tunnel to Four Data Centers

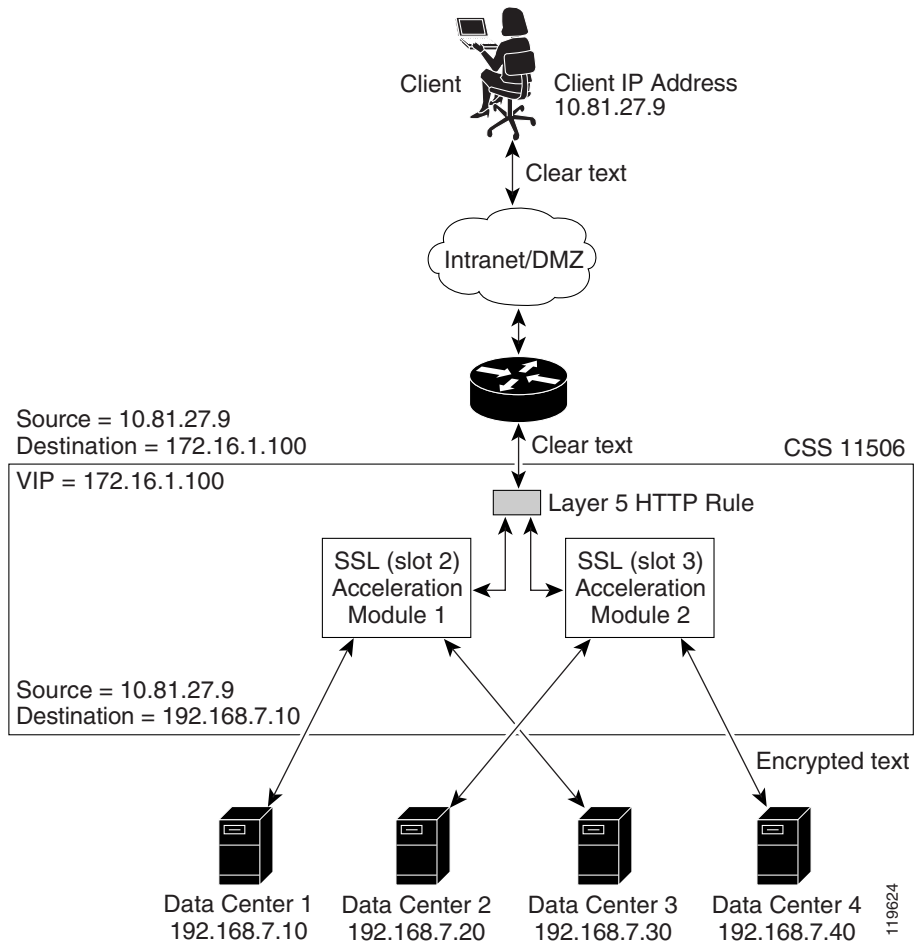
In [Figure 8-7](#), an office contains a CSS 11506 with two SSL modules. Clients connect to a CSS VIP using clear text. The CSS load balances (by applying one of the **advanced-balance** sticky commands), NATs, and sends the connection to an SSL initiation service.

The service of type **ssl-init** tells the CSS to send the connection to the SSL module defined by the **slot** command. The service also defines the IP address of the destination (remote site).

When the connection leaves the service and hits the appropriate SSL module, the SSL proxy list must contain the destination IP address (the **ssl-init** service IP address). The SSL module encrypts the traffic and sends it to the configured destination.

- To optimally load balance flows, you must balance the SSL initiation VIPs and the SSL modules when multiple SSL modules exist (as in this example).
- The SSL initiation feature requires that the proxy list be applied to the SSL module via a service of type **ssl-init**.

Figure 8-7 SSL Initiation Between a CSS and Four Data Centers



```
!!***** GLOBAL *****
ssl associate rsakey rsakey_association rsakey.pem
ssl associate cert rsacert_association rsacert.pem

ftp-record acct-ftp 192.168.7.241 root des-password
ig5haaufqbnfuarb/tmp

!***** INTERFACE *****
interface 1/1
  bridge vlan 10
```

```
interface 1/2
  bridge vlan 20

!***** CIRCUIT *****
circuit VLAN10

  ip address 172.16.1.1 255.255.255.0

circuit VLAN20

  ip address 192.168.7.1 255.255.255.0

!***** SSL PROXY LIST *****
ssl-proxy-list SSLInit_list
  backend-server 1
  backend-server 1 ip address 192.168.7.10
  backend-server 1 server-ip 192.168.7.10
  backend-server 1 type initiation
  backend-server 2
  backend-server 2 ip address 192.168.7.20
  backend-server 2 server-ip 192.168.7.20
  backend-server 2 type initiation
  backend-server 3
  backend-server 3 ip address 192.168.7.30
  backend-server 3 server-ip 192.168.7.30
  backend-server 3 type initiation
  backend-server 4
  backend-server 4 ip address 192.168.7.40
  backend-server 4 server-ip 192.168.7.40
  backend-server 4 type initiation
  active

!***** SERVICE *****

service DC1
  type ssl-init
  ip address 192.168.7.10
  protocol tcp
  port 80
  slot 2
  keepalive type ssl
  keepalive port 443
  add ssl-proxy-list SSLInit_list
  active

service DC2
  type ssl-init
  ip address 192.168.7.20
```

```

protocol tcp
port 80
slot 3
keepalive type ssl
keepalive port 443
add ssl-proxy-list SSLInit_list
active

service DC3
type ssl-init
ip address 192.168.7.30
protocol tcp
port 80
slot 2
keepalive type ssl
keepalive port 443
add ssl-proxy-list SSLInit_list
active

service DC4
type ssl-init
ip address 192.168.7.40
protocol tcp
port 80
slot 3
keepalive type ssl
keepalive port 443
add ssl-proxy-list SSLInit_list
active

!***** OWNER *****
owner Example

content ssl-init
protocol tcp
vip address 172.16.1.100
port 80
add service DC1
add service DC2
add service DC3
add service DC4
advanced-balance arrowpoint-cookie
active

```

SSL Tunnel to One Data Center with Server Authentication

In [Figure 8-8](#), an office contains a CSS 11506 with two SSL modules. Clients connect to the CSS VIP 192.168.7.101 using clear text. The CSS load balances (by applying the **advanced-balance arrowpoint-cookie** sticky commands), NATs, and sends the connection to an SSL initiation service.

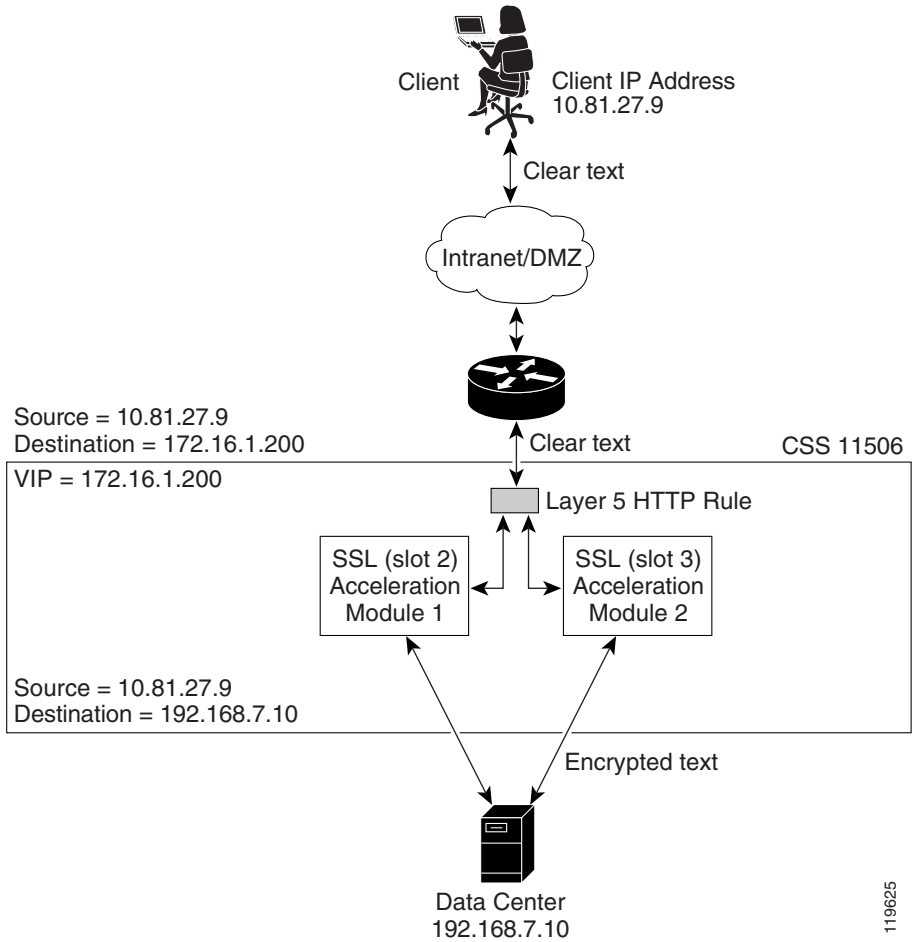
The service of type **ssl-init** tells the CSS to send the connection to the SSL module defined by the **slot** command. The service also defines the IP address of the destination (data center).

When the traffic leaves the service and enters the appropriate SSL module (in this case, slot 2), the SSL proxy list must contain the destination IP address (the **ssl-init** service IP address). The SSL module encrypts the traffic and sends it to the configured destination. By adding the certificate of the CA that signed the SSL server certificate, the CSS can authenticate the server during the SSL handshake.

Be aware of the following configuration requirements:

- To optimally utilize multiple SSL modules, you must balance the SSL initiation VIPs and the SSL modules in your configuration.
- You must apply the SSL initiation proxy list to the SSL module using a service of type **ssl-init**.
- You must obtain the certificate of the CA that issued the SSL server certificate. After you import it and associate it, define the CA certificate as a **cacert** within the SSL proxy list.

Figure 8-8 SSL Initiation Between a CSS and One Data Center



119625

```
! ***** GLOBAL *****
ssl associate rsakey rsakey_association rsakey.pem
ssl associate cert rsacert_association rsacert.pem

ftp-record acct-ftp 192.168.7.241 root des-password
ig5haaufqbnfuarb/tmp
ftp-record config 192.168.1.241 root des-password 4f1bxangrgehjgka
/users/rclement/ssl-init
```

```
!***** INTERFACE *****
interface 1/1
    bridge vlan 10

interface 1/2
    bridge vlan 20

!***** CIRCUIT *****
circuit VLAN10

    ip address 172.16.1.1 255.255.255.0

circuit VLAN20

    ip address 192.168.7.1 255.255.255.0

!***** SSL PROXY LIST *****
ssl-proxy-list SSLInit_list
    backend-server 1
    backend-server 1 ip address 192.168.7.10
    backend-server 1 server-ip 192.168.7.10
    backend-server 1 type initiation
    active

!***** SERVICE *****

service DC-SSL1
    type ssl-init
    ip address 192.168.7.10
    protocol tcp
    port 80
    slot 2
    keepalive type ssl
    keepalive port 443
    add ssl-proxy-list SSLInit_list
    active

service DC-SSL2
    type ssl-init
    ip address 192.168.7.10
    protocol tcp
    port 80
    slot 3
    keepalive type ssl
    keepalive port 443
    add ssl-proxy-list SSLInit_list
    active
```

```
!***** OWNER *****
owner Example

content ssl-init
  protocol tcp
  vip address 192.168.7.200
  port 80
  add service DC-SSL1
  add service DC-SSL2
  advanced-balance arrowpoint-cookie
  active
```



A

- assigning CRL record [4-19](#)
- associating (SSL)
 - Diffie-Hellman parameter file [3-19](#)
 - DSA key pair [3-18](#)
 - RSA key pair [3-17](#)
 - SSL certificates [3-17](#)
- audience [xvi](#)
- authentication, client [4-15](#)

B

- back-end server
 - configuring for SSL initiation [6-4](#)
 - SSL initiation [6-4](#)
 - SSL TCP client-side connection options [6-17](#)
- back-end SSL server
 - acceleration service type [5-19](#)
 - activating service [4-51, 5-21](#)
 - cipher suites [5-8](#)
 - configuration quick start [2-9](#)
 - configuring [5-3](#)
 - configuring service IP address [5-20](#)

- configuring service port number [5-20](#)
- configuring to a service [5-19](#)
- content rule [5-22](#)
- handshake negotiation [5-10](#)
- IP address [5-6](#)
- running-config example [2-10](#)
- server IP address [5-7](#)
- server port number [5-7](#)
- server-side TCP SYN timeout [5-14](#)
- session cache timeout [5-9](#)
- SSL TCP client-side connection options [5-15](#)
- SSL version [5-8](#)
- TCP buffering [5-16](#)
- TCP nagle algorithm, client-side connection [5-15](#)
- TCP nagle algorithm, server-side connection [5-15](#)
- virtual client TCP inactivity timeout [5-12](#)
- virtual client TCP SYN timeout [5-12](#)
- virtual port [5-6](#)

C

- CA certificate
 - client authentication [4-16](#)

- certificates (SSL)
 - associating [3-17](#)
 - associations, viewing [7-2, 7-8](#)
 - CA [6-21](#)
 - certificate signing request, generating [3-8](#)
 - DSA certificate association, SSL proxy list [4-9](#)
 - file formats [3-14](#)
 - global site certificate [3-9](#)
 - importing/exporting [3-12, 3-14](#)
 - overview [1-2, 1-6](#)
 - preparing global site [3-11](#)
 - removing [3-20](#)
 - RSA certificate association, SSL proxy list [4-8](#)
 - self-signed certificate, generating [3-10](#)
 - storage [1-7](#)
 - verifying [3-20](#)
- cipher suites (SSL) [4-11](#)
- client authentication
 - CA certificate [4-16](#)
 - certificates and keys [6-19](#)
 - configuring [4-15](#)
 - CRL record [4-17](#)
 - display fields [7-11](#)
 - enabling [4-16](#)
 - handling failures [4-19](#)
 - overview [1-9](#)
 - statistics [7-22](#)
 - client certificate information
 - HTTP header insertion [4-21](#)
 - Close-Notify alert [4-34](#)
 - configuration example
 - SSL proxy configurations [8-1](#)
 - configuration quick start
 - RSA certificate and key generation [2-2](#)
 - RSA certificate and key import [2-5](#)
 - SSL proxy list, back-end SSL server [2-9](#)
 - SSL proxy list, SSL initiation server [2-10](#)
 - SSL proxy list, virtual server [2-6](#)
 - SSL service [2-13](#)
 - configuring
 - CA certificate for client authentication [4-16](#)
 - client authentication [4-15](#)
 - configuring CRL record [4-17](#)
 - content rule
 - back-end SSL service [5-22](#)
 - running-config example for back-end SSL server [2-17, 2-19, 2-21](#)
 - running-config example for virtual SSL server [2-15](#)
 - SSL initiation [6-29](#)
 - SSL rule quick start [2-13](#)
 - virtual SSL service [4-52](#)
 - CRL record
 - assigning [4-19](#)
 - configuring [4-17](#)
 - displaying [7-14](#)

D

Diffie-Hellman

- associating key exchange file [3-19](#)
- cipher suites [4-11](#)
- generating key agreement file [3-7](#)
- key exchange parameter file association, SSL proxy list [4-10](#)
- overview [1-3](#)
- parameter associations, viewing [7-6](#)

displaying

- active flows [7-24](#)
- all certificate and key associations [7-7](#)
- certificate associations [7-2](#)
- certificates, key pairs, and Diffie-Hellman parameter files loaded on the CSS [7-8](#)
- client authentication information [7-16](#)
- CRL record [7-14](#)
- Diffie-Hellman parameters [7-6](#)
- DSA private key associations [7-5](#)
- RSA private key associations [7-4](#)
- SSL certificates and key pairs [7-1](#)
- SSL proxy list [7-9](#)
- SSL statistics [7-16](#)
- URL rewrite rule statistics [7-15](#)

documentation

- audience [xvi](#)
- chapter contents [xvi](#)
- set [xvii](#)
- symbols and conventions [xxi](#)

DSA

- associating key pair [3-18](#)
- certificate association, SSL proxy list [4-9](#)
- cipher suites [4-11](#)
- generating key pair [3-6](#)
- key pair association, SSL proxy list [4-10](#)
- key pair associations, viewing [7-5, 7-7, 7-8](#)
- overview [1-5](#)

E

example

- SSL proxy configurations [8-1](#)
- exporting SSL keys and certificates [3-14](#)

HHTTP header insertion [4-20](#)

- client certificate information [4-21](#)
- display fields [7-13](#)
- prefix [4-32](#)
- server certificate information [4-25](#)
- session information [4-30](#)
- static text string [4-32](#)

I

- importing SSL keys and certificates [3-14](#)
- initiation, SSL [6-1](#)

K

- keepalive
 - disabling for SSL Acceleration Module [4-50](#)
- keepalive, configuring for SSL initiation [6-27](#)
- keys (SSL)
 - associating [3-17](#), [3-18](#), [3-19](#)
 - Diffie-Hellman key agreement file [3-7](#)
 - Diffie-Hellman key exchange parameter file association, SSL proxy list [4-10](#)
 - Diffie-Hellman parameter associations, viewing [7-6](#)
 - DSA key pair association, SSL proxy list [4-10](#)
 - DSA key pair associations, viewing [7-5](#), [7-7](#), [7-8](#)
 - DSA key pairs [3-6](#)
 - importing/exporting [3-12](#), [3-14](#)
 - overview [1-2](#), [1-6](#)
 - removing [3-20](#)
 - RSA certificate association, SSL proxy list [4-9](#)
 - RSA key pair, generating [3-5](#)
 - RSA key pair associations, viewing [7-4](#), [7-8](#)
 - storage [1-7](#)

N

- nagle algorithm
 - client-side connection [6-15](#)
 - server-side connection [6-17](#)

P

- password for imported certificates/keys [3-15](#)

Q

- quick start
 - RSA certificate and key generation [2-2](#)
 - RSA certificate and key import [2-5](#)
 - SSL proxy list for back-end SSL server [2-9](#)
 - SSL proxy list for SSL initiation server [2-10](#)
 - SSL proxy list for virtual server [2-6](#)
 - SSL service [2-13](#)

R

- RSA
 - associating key pair [3-17](#)
 - certificate association, SSL proxy list [4-8](#)
 - certificate association in SSL proxy list [4-9](#)
 - cipher suites [4-11](#)
 - generating key pair [3-5](#)
 - key pair associations, viewing [7-4](#)
 - overview [1-3](#)
 - quick start [2-2](#), [2-5](#)
 - running-config example [2-6](#)
- running-config example
 - back-end SSL server [2-10](#)
 - back-end SSL server service and content rule [2-17](#), [2-19](#), [2-21](#)

RSA certificate [2-6](#)
 SSL initiation server [2-12](#)
 SSL proxy configurations [8-5, 8-8, 8-12](#)
 virtual SSL server [2-8](#)
 virtual SSL server service and content
 rule [2-15](#)

S

server certificate information
 HTTP header insertion [4-25](#)

service
 activating [4-51, 5-21, 6-28](#)
 configuring back-end SSL server IP
 address [5-20](#)
 configuring back-end SSL server port
 number [5-20](#)
 configuring SSL initiation server IP
 address [6-25](#)
 keepalive messages, disabling for SSL
 Acceleration Module [4-50](#)
 running-config example for back-end SSL
 server [2-17, 2-19, 2-21](#)
 running-config example for virtual SSL
 server [2-15](#)
 SSL Acceleration Module slot,
 specifying [4-49](#)
 SSL acceleration type [4-48, 5-19](#)
 SSL initiation type [6-25](#)
 SSL module slot, specifying [6-26](#)
 SSL proxy lists, adding [4-47, 4-49, 5-18, 5-19,](#)
[6-26](#)
 SSL service, creating [4-48, 5-19, 6-25](#)
 SSL service quick start [2-13](#)
 SSL session ID cache size [4-50, 6-28](#)
 suspending [4-52, 5-22, 6-29](#)

service type
 ssl-accel [4-48](#)
 ssl-accel-backend [5-19](#)
 ssl-init [6-25](#)

session information
 HTTP header insertion [4-30](#)

SSL
 certificate associations, viewing [7-2, 7-8](#)
 certificates [1-4, 3-10, 3-12, 3-14, 3-17, 3-20](#)
 certificate signing request, generating [3-8](#)
 certificate signing request, global site [3-9](#)
 cipher suites, specifying [4-11](#)
 configuration information, viewing [7-9](#)
 cryptography capabilities [1-6](#)
 Diffie-Hellman key agreement file [1-3, 3-7,](#)
[3-19, 7-6](#)
 DSA digital signatures [1-5](#)
 DSA key pairs [3-6, 3-18](#)
 generating keys and certificates [3-4](#)
 global site certificate, preparing [3-11](#)
 handshake negotiation [4-38](#)
 HTTP 300-series redirects [4-34](#)
 importing/exporting certificates and
 keys [3-14](#)
 initiation [6-1](#)
 key pairs [3-20, 7-4, 7-5, 7-7, 7-8](#)

- nagle algorithm, client-side connection [4-44](#), [5-15](#), [6-15](#)
- nagle algorithm, server-side connection [4-44](#), [5-15](#), [6-17](#)
- overview [1-1](#)
- processing of flows [8-2](#)
- public key infrastructure [1-2](#)
- queue data delay [4-40](#)
- quick start procedures [2-1](#)
- RSA key pairs [1-3](#), [3-5](#), [3-17](#)
- session cache [4-37](#), [4-50](#), [6-28](#)
- SSL Acceleration Module [1-7](#)
- SSL flows, viewing [7-24](#)
- SSL proxy configurations examples [8-1](#)
- SSL proxy list, creating [4-2](#), [5-2](#), [6-3](#)
- statistics [7-15](#), [7-16](#), [7-24](#)
- TCP client-side connection options [4-41](#), [4-44](#), [5-15](#), [6-15](#), [6-17](#)
- TCP connection buffering [4-45](#), [5-16](#), [6-18](#)
- TCP inactivity timeout [4-43](#)
- TCP server-side connection options [4-42](#), [6-17](#)
- TCP SYN timeout [4-43](#)
- URL rewrite [4-34](#)
- URL rewrite statistics, viewing [7-15](#)
- SSL Acceleration Module
 - creating SSL service [4-48](#), [5-19](#)
 - overview [1-1](#), [1-7](#)
 - specifying in SSL service [4-49](#)
 - statistics, viewing [7-15](#), [7-16](#)
- SSL back-end server, see back-end SSL server
- SSL initiation
 - adding a proxy list to services [6-26](#)
 - back-end server IP address, configuring [6-7](#)
 - back-end server virtual port, configuring [6-7](#)
 - CA certificates, configuring [6-21](#)
 - cipher suites, configuring [6-9](#)
 - client certificates and keys, configuring [6-19](#)
 - client-side TCP connection options [6-15](#)
 - configuring a back-end server [6-4](#)
 - content rule, configuring [6-29](#)
 - creating a proxy list [6-3](#)
 - initiation service type [6-25](#)
 - keepalive, configuring [6-27](#)
 - overview [6-1](#)
 - proxy list, activating and suspending [6-23](#)
 - real SSL server IP address, configuring [6-8](#)
 - real SSL server port number, configuring [6-8](#)
 - server, configuring [6-6](#)
 - server-side TCP inactivity timeout, specifying [6-17](#)
 - server-side TCP SYN timeout, specifying [6-16](#)
 - service, activating [6-28](#)
 - service, configuring [6-24](#)
 - service, creating [6-25](#)
 - service, suspending [6-29](#)
 - service IP address, configuring [6-25](#)
 - session cache timeout, configuring [6-11](#)
 - session ID cache size [6-28](#)
 - SSL module slot, specifying [6-26](#)

- SSL session handshake renegotiation, configuring [6-11](#)
 - SSL version, configuring [6-9](#)
 - TCP buffering [6-18](#)
 - TCP client-side connection options [6-15](#)
 - TCP nagle algorithm, client-side connection [6-15](#)
 - TCP nagle algorithm, server-side connection [6-17](#)
 - TCP server-side connection options [6-17](#)
 - troubleshooting [6-30](#)
 - virtual client TCP inactivity timeout, specifying [6-14](#)
 - virtual client TCP SYN timeout, specifying [6-13](#)
 - SSL initiation server
 - configuration quick start [2-10](#)
 - running-config example [2-12](#)
 - SSL module
 - specifying in SSL service [6-26](#)
 - SSL proxy configurations
 - full proxy example [8-17](#)
 - transparent example - HTTP and back-end SSL servers [8-12](#)
 - transparent example - one module [8-5](#)
 - transparent example - two SSL modules [8-8](#)
 - SSL proxy list
 - activating [4-46, 5-17, 6-23](#)
 - adding to service [4-49, 5-19, 6-26](#)
 - adding to SSL services [4-47, 5-18](#)
 - back-end SSL server, configuring [5-3](#)
 - creating [4-2, 5-2, 6-3](#)
 - initiation [6-3](#)
 - mode [4-2, 5-2, 6-3](#)
 - overview [4-2, 5-2](#)
 - quick start for back-end SSL server [2-9](#)
 - quick start for SSL initiation server [2-10](#)
 - quick start for virtual server [2-6](#)
 - SSL initiation back-end server, configuring [6-4](#)
 - suspending [4-47, 5-17, 6-23](#)
 - viewing [7-9](#)
 - virtual server, configuring [4-4](#)
 - SSL termination
 - configuring [4-1](#)
 - example [8-1](#)
 - overview [1-8](#)
 - static text string
 - HTTP header insertion [4-32](#)
-
- ## T
- TCP FIN message
 - terminating client connection [4-34](#)
 - TCP nagle algorithm
 - client-side connection [6-15](#)
 - server-side connection [6-17](#)
 - terminating client connection [4-34](#)
 - troubleshooting SSL initiation [6-30](#)

V

- virtual SSL server
 - acceleration service type [4-48](#)
 - activating service [4-51, 5-21](#)
 - cipher suites [4-11](#)
 - configuration quick start [2-6](#)
 - configuring content rule [4-52](#)
 - configuring to a service [4-49](#)
 - Diffie-Hellman parameter file association [4-10](#)
 - DSA certificate association [4-9](#)
 - DSA key pair association, specifying [4-10](#)
 - HTTP 300-series redirects [4-34](#)
 - queue data delay [4-40](#)
 - RSA certificate association [4-8](#)
 - RSA key pair association [4-9](#)
 - running-config example [2-8](#)
 - SSL session cache timeout [4-37](#)
 - SSL session handshake renegotiation [4-38](#)
 - SSL TCP client-side connection options [4-41, 4-44](#)
 - SSL TCP inactivity timeout [4-43](#)
 - SSL TCP server-side connection options [4-42](#)
 - SSL TCP SYN timeout [4-43](#)
 - TCP buffering [4-45](#)
 - TCP nagle algorithm, client-side connection [4-44](#)
 - TCP nagle algorithm, server-side connection [4-44](#)
 - terminating client connection (Close-Notify alert) [4-34](#)
 - URL rewrite [4-34](#)
 - version [4-33](#)
 - VIP address [4-6](#)
 - virtual TCP port [4-7](#)