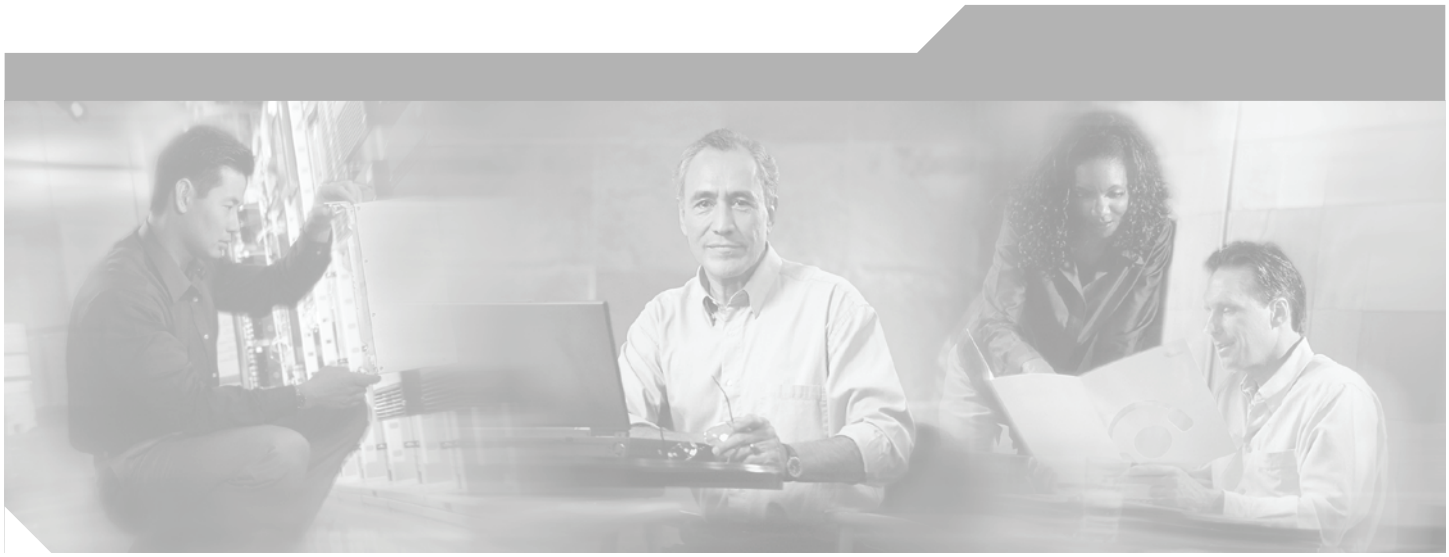


Catalyst 6500 Series Switch SSL Services Module Configuration Note

Software Release 3.1
January, 2006

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-9104-01

Catalyst 6500 Series Switch SSL Services Module Configuration Note

Copyright Notices

Third-party software used under license accompanies the Cisco SSL Services Module Software release 1.1(1). One or more of the following notices may apply in connection with the license and use of such third-party software.

GNU General Public License

The Catalyst 6500 Series SSL Services Module contains software covered under the GNU Public License (listed below). If you would like to obtain the source for the modified GPL code in the SSL Services Module, please send a request to ssl_sw_req@Cisco.com.

License Text

59 Temple Place - Suite 330, Boston, MA 02111-1307, USA

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software—to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program," below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you."

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and “any later version”, you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. END OF TERMS AND CONDITIONS.

CONTENTS

Preface

This preface describes who should read the *Catalyst 6500 Series Switch SSL Services Module Configuration Note*, how it is organized, and its document conventions.

This publication does not contain the instructions to install the Catalyst 6500 series switch chassis. For information on installing the switch chassis, refer to the *Catalyst 6500 Series Switch Installation Guide*.

Audience

This publication is for experienced network administrators who are responsible for configuring and maintaining Catalyst 6500 series switches.

Organization

This publication is organized as follows:

Chapter	Title	Description
Chapter 1	Overview	Presents an overview of the Catalyst 6500 series switch SSL Services Module.
Chapter 2	Initial Configurations	Describes the initial configuration for the Catalyst 6500 series switch and the SSL Services Module.
Chapter 3	Configuring the SSL Services Module	Describes how to configure the SSL Services Module.
Chapter 4	Advanced Configurations for the SSL Services Module	Describes how to configure advanced features on the SSL Services Module.
Chapter 5	Configuring Different Modes of Operation	Describes how to configure the SSL Services Module in either a standalone configuration or with a Content Switching Module (CSM).

Chapter	Title	Description
Appendix A	Example SSL Configurations	Contains example configurations.
Appendix B	Upgrading the Images	Contains information for upgrading the application and maintenance partitions.
Appendix C	Testing SSL Proxy Services	Contains information for testing or troubleshooting SSL proxy services.

Conventions

This publication uses the following conventions:

Convention	Description
boldface font	Commands, command options, and keywords are in **boldface**.
italic font	Arguments for which you supply values are in *italics*.
[ ]	Elements in square brackets are optional.
{ x | y | z }	Alternative keywords are grouped in braces and separated by vertical bars.
[x | y | z]	Optional alternative keywords are grouped in brackets and separated by vertical bars.
string	A nonquoted set of characters. Do not use quotation marks around the string or the string will include the quotation marks.
`screen` font	Terminal sessions and information the system displays are in `screen` font.
`boldface screen` font	Information you must enter is in **`boldface screen`** font.
italic screen font	Arguments for which you supply values are in *italic screen* font.
^	The symbol ^ represents the key labeled Control—for example, the key combination ^D in a screen display means hold down the Control key while you press the D key.
< >	Nonprinting characters, such as passwords are in angle brackets.

Notes use the following conventions:

Note Means *reader take note*. Notes contain helpful suggestions or references to material not covered in the publication.

Tips use the following conventions:

Tip Means *the following information will help you solve a problem*. The tips information might not be troubleshooting or even an action, but could be useful information, similar to a Timesaver.

Cautions use the following conventions:

Caution Means *reader be careful*. In this situation, you might do something that could result in equipment damage or loss of data.

Warnings use the following conventions:

Warning **IMPORTANT SAFETY INSTRUCTIONS**

This warning symbol means danger. You are in a situation that could cause bodily injury. Before you work on any equipment, be aware of the hazards involved with electrical circuitry and be familiar with standard practices for preventing accidents. Use the statement number provided at the end of each warning to locate its translation in the translated safety warnings that accompanied this device. Statement 1071

SAVE THESE INSTRUCTIONS

Related Documentation

For more detailed installation and configuration information, refer to the following publications:

- *Release Notes for Catalyst 6500 Series SSL Services Module Software Release 3.x*
- *Catalyst 6500 Series SSL Services Module Installation and Verification Note*
- *Catalyst 6500 Series SSL Services Module Command Reference*
- *Catalyst 6500 Series SSL Services Module System Messages*
- *Catalyst 6500 Series Switch Installation Guide*
- *Catalyst 6500 Series Switch Module Installation Guide*
- *Catalyst 6500 Series Switch Software Configuration Guide*
- *Catalyst 6500 Series Switch Command Reference*
- *Catalyst 6500 Series Switch Cisco IOS Software Configuration Guide*
- *Catalyst 6500 Series Switch Cisco IOS Command Reference*
- *Regulatory Compliance and Safety Information for the Catalyst 6500 Series Switches*

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/techsupport

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

Product Documentation DVD

Cisco documentation and additional literature are available in the Product Documentation DVD package, which may have shipped with your product. The Product Documentation DVD is updated regularly and may be more current than printed documentation.

The Product Documentation DVD is a comprehensive library of technical product documentation on portable media. The DVD enables you to access multiple versions of hardware and software installation, configuration, and command guides for Cisco products and to view technical documentation in HTML. With the DVD, you have access to the same documentation that is found on the Cisco website without being connected to the Internet. Certain products also have .pdf versions of the documentation available.

The Product Documentation DVD is available as a single unit or as a subscription. Registered Cisco.com users (Cisco direct customers) can order a Product Documentation DVD (product number DOC-DOCDVD=) from Cisco Marketplace at this URL:

http://www.cisco.com/go/marketplace/

Ordering Documentation

Beginning June 30, 2005, registered Cisco.com users may order Cisco documentation at the Product Documentation Store in the Cisco Marketplace at this URL:

http://www.cisco.com/go/marketplace/

Nonregistered Cisco.com users can order technical documentation from 8:00 a.m. to 5:00 p.m. (0800 to 1700) PDT by calling 1 866 463-3487 in the United States and Canada, or elsewhere by calling 011 408 519-5055. You can also order documentation by e-mail at tech-doc-store-mkpl@external.cisco.com or by fax at 1 408 519-5001 in the United States and Canada, or elsewhere at 011 408 519-5001.

Documentation Feedback

You can rate and provide feedback about Cisco technical documents by completing the online feedback form that appears with the technical documents on Cisco.com.

You can send comments about Cisco 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.

Cisco Product Security Overview

Cisco provides a free online Security Vulnerability Policy portal at this URL:

http://www.cisco.com/en/US/products/products_security_vulnerability_policy.html

From this site, you can perform these tasks:

- Report security vulnerabilities in Cisco products.
- Obtain assistance with security incidents that involve Cisco products.
- Register to receive security information from Cisco.

A current list of security advisories and notices for Cisco products is available at this URL:

http://www.cisco.com/go/psirt

If you prefer to see advisories and notices as they are updated in real time, you can access a Product Security Incident Response Team Really Simple Syndication (PSIRT RSS) feed from this URL:

http://www.cisco.com/en/US/products/products_psirt_rss_feed.html

Reporting Security Problems in Cisco Products

Cisco is committed to delivering secure products. We test our products internally before we release them, and we strive to correct all vulnerabilities quickly. If you think that you might have identified a vulnerability in a Cisco product, contact PSIRT:

- Emergencies—security-alert@cisco.com

  An emergency is either a condition in which a system is under active attack or a condition for which a severe and urgent security vulnerability should be reported. All other conditions are considered nonemergencies.

- Nonemergencies—psirt@cisco.com

In an emergency, you can also reach PSIRT by telephone:

- 1 877 228-7302
- 1 408 525-6532

Tip We encourage you to use Pretty Good Privacy (PGP) or a compatible product to encrypt any sensitive information that you send to Cisco. PSIRT can work from encrypted information that is compatible with PGP versions 2.*x* through 8.*x*.

Never use a revoked or an expired encryption key. The correct public key to use in your correspondence with PSIRT is the one linked in the Contact Summary section of the Security Vulnerability Policy page at this URL:

http://www.cisco.com/en/US/products/products_security_vulnerability_policy.html

The link on this page has the current PGP key ID in use.

Obtaining Technical Assistance

Cisco Technical Support provides 24-hour-a-day award-winning technical assistance. The Cisco Technical Support & Documentation website on Cisco.com features extensive online support resources. In addition, if you have a valid Cisco service contract, Cisco Technical Assistance Center (TAC) engineers provide telephone support. If you do not have a valid Cisco service contract, contact your reseller.

Cisco Technical Support & Documentation Website

The Cisco Technical Support & Documentation 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, at this URL:

http://www.cisco.com/techsupport

Access to all tools on the Cisco Technical Support & Documentation 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

Note Use the Cisco Product Identification (CPI) tool to locate your product serial number before submitting a web or phone request for service. You can access the CPI tool from the Cisco Technical Support & Documentation website by clicking the **Tools & Resources** link under Documentation & Tools. Choose **Cisco Product Identification Tool** from the Alphabetical Index drop-down list, or click the **Cisco Product Identification Tool** link under Alerts & RMAs. The CPI tool offers three search options: by product ID or model name; by tree view; or for certain products, by copying and pasting **show** command output. Search results show an illustration of your product with the serial number label location highlighted. Locate the serial number label on your product and record the information before placing a service call.

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 provides recommended solutions. If your issue is not resolved using the recommended resources, your service request is assigned to a Cisco 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 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, documentation, and logo merchandise. Visit Cisco Marketplace, the company store, at this URL:

  http://www.cisco.com/go/marketplace/

- *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

  or view the digital edition at this URL:

  http://ciscoiq.texterity.com/ciscoiq/sample/

- *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

- Networking products offered by Cisco Systems, as well as customer support services, can be obtained at this URL:

  http://www.cisco.com/en/US/products/index.html

- Networking Professionals Connection is an interactive website for networking professionals to share questions, suggestions, and information about networking products and technologies with Cisco experts and other networking professionals. Join a discussion at this URL:

  http://www.cisco.com/discuss/networking

- 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

The SSL Services Module is a Layer 4-through-Layer 7 service module that you can install into the Catalyst 6500 series switch. The module terminates secure sockets layer (SSL) transactions and accelerates the encryption and decryption of data used in SSL sessions.

The module operates either in a standalone configuration or with the Content Switching Module (CSM). In a standalone configuration, secure traffic is directed to the module using policy-based routing (PBR). When used with the CSM, only encrypted client traffic is forwarded to the module, while clear text traffic is forwarded to the real servers.

The SSL Services Module uses the SSL protocol to enable secure transactions of data through privacy, authentication, and data integrity; the protocol relies upon certificates, public keys, and private keys.

The certificates, which are similar to digital ID cards, verify the identity of the server to the clients. The certificates, which are issued by certificate authorities, include the name of the entity to which the certificate was issued, the public key of the entity, and the time stamps that indicate the certificate expiration date.

The public and private keys are the ciphers that are used to encrypt and decrypt information. The public key is shared without any restrictions, but the private key is never shared. Each public-private key pair works together; data that is encrypted with the public key can only be decrypted with the corresponding private key.

Initial Configurations

This chapter describes how to initially configure the SSL Services Module and has these sections:

Using the CLI

The software interface for the SSL Services Module is the Cisco IOS CLI. To understand the Cisco IOS CLI and Cisco IOS command modes, refer to Chapter 2, "Command-Line Interfaces," in the *Catalyst 6500 Series Switch Cisco IOS Software Configuration Guide*.

Unless your switch is located in a fully trusted environment, we recommend that you configure the SSL Services Module through a direct connection to the module's console port or through an encrypted session using Secure Shell (SSH). See the "Configuring SSH" section on page 2-4 for information on configuring SSH on the module.

Note The initial SSL Services Module configuration must be made through a direct connection to the console port on the module.

Initial SSL Services Module Configuration

Note You are required to make the following initial SSL Services Module configurations through a direct connection to the SSL Services Module console port. After the initial configurations, you can make an SSH or Telnet connection to the module to further configure the module.

The initial SSL Services Module configuration consists of the following tasks:

Configuring Interfaces on the SSL Services Module

Note The ssl-proxy0 interface is enabled by default and should not be shut down or otherwise configured.

To configure the SSL proxy interface, perform this task:

	Command	Purpose
Step 1	`ssl-proxy(config)# interface ssl-proxy 0.`*subinterface-number*	Selects a subinterface to configure.
Step 2	`ssl-proxy(config-subif)# encap dot1q` *vlan_id*	Uses 802.1Q to send the Ethernet frames from the subinterface to the assigned *vlan-id* without any encapsulation.
Step 3	`ssl-proxy(config-subif)# ip address` *ip-address ip-address-mask*	Configures an IP address on the subinterface.
Step 4	`ssl-proxy(config-subif)# no shutdown`	Enables SSL proxy access on the subinterface.

This example shows how to configure the SSL proxy interface:

```
ssl-proxy(config)# interface ssl-proxy 0.1
ssl-proxy(config-subif)# encap dot1q 100
ssl-proxy(config-subif)# ip address 10.1.0.20 255.255.255.0
ssl-proxy(config-subif)# no shutdown
ssl-proxy(config-subif)# exit
ssl-proxy(config)#
```

Configuring the Default Route

To configure the default route, perform this task:

Command	Purpose
`ssl-proxy(config)# ip route` *prefix mask ip-address*	Configures a default route.

This example shows how to configure the default route:

```
ssl-proxy(config)# ip route 0.0.0.0 0.0.0.0 10.10.10.100
ssl-proxy(config)#
```

Configuring Telnet Remote Access

To configure the SSL Services Module for Telnet remote access, perform this task:

	Command	Purpose
Step 1	`ssl-proxy(config)# enable password` *password*	Specifies a local enable password.
Step 2	`ssl-proxy(config)# line vty` *starting-line-number ending-line-number*	Identifies a range of lines for configuration and enters line configuration mode.
Step 3	`ssl-proxy(config-line)# login`	Enables password checking at login.
Step 4	`ssl-proxy(config-line)# password` *password*	Specifies a password on the line.

This example shows how to configure the SSL Services Module for remote access:

```
ssl-proxy(config)# enable password cisco
ssl-proxy(config) line vty 0 4
ssl-proxy(config-line)# login
ssl-proxy(config-line)# password cisco
ssl-proxy(config-line)# end
ssl-proxy#
```

Note In addition to the standard Telnet TCP port 23, other legacy Telnet variant ports will appear as open, but are only used for VTS debugging. These ports are TCP/2001-2003 (VTY virtual terminal), TCP/4001-4003 (raw TCP), and TCP/6001-6003 (binary mode Telnet).

Configuring the Fully Qualified Domain Name

If you are using the SSL Services Module to enroll for certificates from a certificate authority, you must configure the Fully Qualified Domain Name (FQFDN) on the module. The FQDN is the hostname and domain name of the module.

To configure the FQDN, perform this task:

	Command	Purpose
Step 1	`ssl-proxy(config)# hostname name`	Configures the hostname.
Step 2	`ssl-proxy(config)# ip domain-name name`	Configures the domain name.

This example shows how to configure the FQDN on the SSL Services Module:

```
ssl-proxy(config)# hostname ssl-proxy2
ssl-proxy2(config)# ip domain-name example.com
ssl-proxy2(config)# end
ssl-proxy2(config)#
```

Configuring SSH

After you complete the initial configuration for the module, enable SSH on the module, and then configure the user name and password for the SSH connection using either a simple user name and password or using an authentication, authorization, and accounting (AAA) server.

These sections describe how to enable and configure SSH:

Enabling SSH on the Module

SSH uses the first key pair generated on the module. In the following task, you generate a key pair used specifically for SSH.

Note If you generate a general-purpose key pair (as described in the "Generating RSA Key Pairs" section on page 3-5) without specifying the SSH key pair first, SSH is enabled and uses the general-purpose key pair. If this key pair is later removed, SSH is disabled. To reenable SSH, generate a new SSH key pair.

To generate an SSH key pair and enable SSH, perform this task:

	Command	Purpose
Step 1	`ssl-proxy# configure terminal`	Enters configuration mode, selecting the terminal option.
Step 2	`ssl-proxy(config)# ip ssh [version {1|2}] rsa keypair-name ssh_key_name`	Assigns the key pair name to SSH. **Note** Both SSHv1 and SSHv2 are enabled by default. You can specify to enable only version 1 or version 2.
Step 3	`ssl-proxy(config)# crypto key generate rsa general-keys label ssh_key_name`	Generates the SSH key pair. SSH is now enabled.
Step 4	`ssl-proxy(config)# end`	Exits configuration mode.
Step 5	`ssl-proxy# show ip ssh`	Shows the current state of SSH.

This example shows how to enable SSH on the module, and how to verify that SSH is enabled:

```
ssl-proxy(config)# ip ssh rsa keypair-name ssh-key
Please create RSA keys to enable SSH.
ssl-proxy(config)# crypto key generate rsa general-keys label ssh-key
The name for the keys will be: ssh-key
Choose the size of the key modulus in the range of 360 to 2048 for your
  General Purpose Keys. Choosing a key modulus greater than 512 may take
  a few minutes.

How many bits in the modulus [512]: 1024
% Generating 1024 bit RSA keys ...[OK]

ssl-proxy(config)#
*Aug 28 11:07:54.051: %SSH-5-ENABLED: SSH 1.5 has been enabled
ssl-proxy(config)# end

ssl-proxy# show ip ssh
SSH Enabled - version 1.99
Authentication timeout: 120 secs; Authentication retries: 3
ssl-proxy#
```

Configuring the User Name and Password for SSH

To configure the user name and password for the SSH connection, perform this task:

	Command	Purpose
Step 1	`ssl-proxy# configure terminal`	Enters configuration mode, selecting the terminal option.
Step 2	`ssl-proxy(config)# enable password` *password*	Specifies a local enable password, if not already specified.
Step 3	`ssl-proxy(config)# username` *username* {**password** | **secret**} *password*	Specifies the user name and password.
Step 4	`ssl-proxy(config)# line vty` *line-number ending-line-number*	Identifies a range of lines for configuration and enters line configuration mode.
Step 5	`ssl-proxy(config-line)# login local`	Enables local username authentication.

This example shows how to configure the user name and password for the SSH connection to the SSL Services Module:

```
ssl-proxy# configure terminal
ssl-proxy(config)# enable password cisco
ssl-proxy(config)# username admin password admin-pass
ssl-proxy(config)# line vty 0 4
ssl-proxy(config-line)# login local
ssl-proxy(config-line)# end
```

After you configure the user name and password, see the "Initial Catalyst 6500 Series Switch Configuration" section on page 2-6 to configure the switch.

Configuring Authentication, Authorization, and Accounting for SSH

To configure authentication, authorization, and accounting (AAA) for SSH, perform this task:

	Command	Purpose
Step 1	`ssl-proxy# configure terminal`	Enters configuration mode, selecting the terminal option.
Step 2	`ssl-proxy(config)# username` *username* `secret {0|5}` *password*	Enables enhanced password security for the specified, unretrievable username.
Step 3	`ssl-proxy(config)# enable password` *password*	Specifies a local enable password, if not already specified.
Step 4	`ssl-proxy(config)# aaa new-model`	Enables authentication, authorization, and accounting (AAA).
Step 5	`ssl-proxy(config)# aaa authentication login default local`	Specifies the module to use the local username database for authentication.
Step 6	`ssl-proxy(config)# line vty` *line-number ending-line-number*	Identifies a range of lines for configuration and enters line configuration mode.
Step 7	`ssl-proxy(config-line)# transport input ssh`	Configures SSH as the only protocol used on a specific line (to prevent non-SSH connections).

This example shows how to configure AAA for the SSH connection to the SSL Services Module:

```
ssl-proxy# configure terminal
ssl-proxy(config)# username admin secret admin-pass
ssl-proxy(config)# enable password enable-pass
ssl-proxy(config)# aaa new-model
ssl-proxy(config)# aaa authentication login default local
ssl-proxy(config)# line vty 0 4
ssl-proxy(config-line)# transport input ssh
ssl-proxy(config-line)# end
ssl-proxy#
```

After you configure AAA, see the "Initial Catalyst 6500 Series Switch Configuration" section on page 2-6 to configure the switch.

Initial Catalyst 6500 Series Switch Configuration

How you configure the Catalyst 6500 series switch depends on whether you are using Cisco IOS software or the Catalyst operating system software.

The following sections describe how to configure the switch from the CLI for each switch operating system:

Cisco IOS Software

The initial Catalyst 6500 series switch configuration consists of the following:

Configuring VLANs on the Switch

Note VLAN IDs must be the same for the switch and the module. Refer to the "Configuring VLANs" chapter in the *Catalyst 6500 Series Switch Software Configuration Guide* for details.

Note The SSL software supports only the normal-range VLANs (2 through 1005). Limit the SSL Services Module configuration to the normal-range VLANs.

To configure VLANs on the switch, perform this task:

	Command	Purpose
Step 1	`Router# configure terminal`	Enters configuration mode, selecting the terminal option.
Step 2	`Router(config)# vlan vlan_ID`	Enters VLAN configuration mode and adds a VLAN. The valid range is 2 through 1001. **Note** Do not add an external VLAN.
Step 3	`Router(config-vlan)# end`	Updates the VLAN database and returns to privileged EXEC mode.

This example shows how to configure VLANs on the switch:

```
Router> enable
Router# configure terminal
Router(config)# vlan 100
VLAN 100 added:
     Name: VLAN100

Router(config-vlan)# end
```

Configuring Layer 3 Interfaces

To configure the corresponding Layer 3 VLAN interface, perform this task:

	Command	Purpose
Step 1	`Router(config)# interface vlan vlan_ID`	Selects an interface to configure.
Step 2	`Router(config-if)# ip address ip_address subnet_mask`	Configures the IP address and IP subnet.
Step 3	`Router(config-if)# no shutdown`	Enables the interface.
Step 4	`Router(config-if)# exit`	Exits configuration mode.

This example shows how to configure the Layer 3 VLAN interface:

```
Router# configure terminal
Router(config)# interface vlan 100
Router(config-if)# ip address 10.10.1.10 255.255.255.0
Router(config-if)# no shutdown
Router(config-if)# exit
```

Configuring a LAN Port for Layer 2 Switching

To place physical interfaces that connect to the servers or the clients in the corresponding VLAN, perform this task:

	Command	Purpose
Step 1	`Router(config)# interface` *type*[1] *mod/port*	Selects the LAN port to configure.
Step 2	`Router(config-if)# switchport`	Configures the LAN port for Layer 2 switching. **Note** You must enter the **switchport** command once without any keywords to configure the LAN port as a Layer 2 port before you can enter additional **switchport** commands with keywords.
Step 3	`Router(config-if)# switchport mode access`	Puts the LAN port into permanent nontrunking mode and negotiates to convert the link into a nontrunk link. The LAN port becomes a nontrunk port even if the neighboring LAN port does not agree to the change.
Step 4	`Router(config-if)# switchport access vlan` *vlan_ID*	Configures the default VLAN, which is used if the interface stops trunking.
Step 5	`Router(config-if)# no shutdown`	Activates the interface.

1. *type* = ethernet, fastethernet, gigabitethernet, or tengigabitethernet

This example shows how to configure a physical interface as a Layer 2 interface and assign it to a VLAN:

```
Router(config)# interface gigabitethernet 1/1
Router(config-if)# switchport
Router(config-if)# switchport mode access
Router(config-if)# switchport access vlan 100
Router(config-if)# no shutdown
Router(config-if)# exit
```

Adding the SSL Services Module to the Corresponding VLAN

Note By default, the SSL Services Module is in trunking mode with native VLAN 1.

To add the SSL Services Module to the corresponding VLAN, enter this command:

Command	Purpose
`Router (config)# ssl-proxy module mod allowed-vlan vlan_ID`	Configures the VLANs allowed over the trunk to the SSL Services Module. **Note** One of the allowed VLANs must be the admin VLAN.

This example shows how to add an SSL Services Module installed in slot 6 to a specific VLAN:

```
Router>
Router> enable
Router# configure terminal
Router (config)# ssl-proxy module 6 allowed-vlan 100
Router (config)# end
```

Verifying the Initial Configuration

To verify the configuration, enter these commands:

Command	Purpose
`Router# show spanning-tree vlan vlan_ID`	Displays the spanning tree state for the specified VLAN.
`Router# show ssl-proxy mod mod state`	Displays the trunk configuration.

Note In the following examples, the SSL Services Module is installed in slot 4 (Gi4/1).

This example shows how to verify that the module is in forwarding (FWD) state:

```
Router# show spanning-tree vlan 100

VLAN0100
  Spanning tree enabled protocol ieee
  Root ID    Priority    32768
             Address     0009.e9b2.b864
             This bridge is the root
             Hello Time   2 sec  Max Age 20 sec  Forward Delay 15 sec

  Bridge ID  Priority    32768
             Address     0009.e9b2.b864
             Hello Time   2 sec  Max Age 20 sec  Forward Delay 15 sec
             Aging Time 15

Interface        Role Sts Cost      Prio.Nbr Type
---------------- ---- --- --------- -------- --------------------------------
Gi3/1            Desg FWD 4         128.129  P2p
Gi4/1            Desg FWD 4         128.193  P2p
Po261            Desg FWD 3         128.833  P2p
Router
```

This example shows how to verify that the VLAN information displayed matches the VLAN configuration:

```
Router# show ssl-proxy mod 6 state
SSL-services module 6 data-port:
 Switchport:Enabled
Administrative Mode:trunk
Operational Mode:trunk
Administrative Trunking Encapsulation:dot1q
Operational Trunking Encapsulation:dot1q
Negotiation of Trunking:Off
Access Mode VLAN:1 (default)
Trunking Native Mode VLAN:1 (default)
Trunking VLANs Enabled:100
Pruning VLANs Enabled:2-1001
Vlans allowed on trunk:100
Vlans allowed and active in management domain:100
Vlans in spanning tree forwarding state and not pruned:
100
Allowed-vlan :100
```

Catalyst Operating System Software

The initial Catalyst 6500 series switch configuration consists of the following:

- Configuring VLANs on the Switch, page 2-10
- Configuring Layer 3 Interfaces on the MSFC, page 2-11
- Adding the SSL Services Module to the Corresponding VLAN, page 2-11
- Verifying the Initial Configuration, page 2-12

Configuring VLANs on the Switch

Note VLAN IDs must be the same for the switch and the module. Refer to the "Configuring VLANs" chapter in the *Catalyst 6500 Switch Series Software Configuration Guide* for details.

Note The SSL software supports only the normal-range VLANs (2 through 1005). Limit the SSL Services Module configuration to the normal-range VLANs.

To configure VLANs on the switch, perform this task:

	Command	Purpose
Step 1	`Console> enable`	Enters privileged mode.
Step 2	`Console> (enable) set vlan vlan_id`	Adds a VLAN. The valid range is 2 through 1001. **Note** Do not add an external VLAN.

This example shows how to configure VLANs on the switch:

```
Console> enable
Enter Password: <password>
Console> (enable) set vlan 100
Vlan 100 configuration successful
Console> (enable)
```

Configuring Layer 3 Interfaces on the MSFC

To configure the corresponding Layer 3 VLAN interface on the multilayer switch feature card (MSFC), perform this task:

	Command	Purpose
Step 1	`Console> (enable) session [mod]`[1]	Accesses the MSFC from the switch CLI using a Telnet session[2].
Step 2	`Router> enable`	Enters enable mode.
Step 3	`Router# configure terminal`	Enters global configuration mode.
Step 4	`Router(config)# interface vlan vlan_id`	Specifies a VLAN interface on the MSFC.
Step 5	`Router(config-if)# ip address ip_address subnet_mask`	Assigns an IP address to the VLAN.
Step 6	`Router(config-if)# no shutdown`	Enables the interface.
Step 7	`Router(config-if)# exit`	Exits the MSFC CLI and returns to the switch CLI.

1. The *mod* keyword specifies the module number of the MSFC; either 15 (if the MSFC is installed on the supervisor engine in slot 1) or 16 (if the MSFC is installed on the supervisor engine in slot 2). If no module number is specified, the console will switch to the MSFC on the active supervisor engine.
2. To access the MSFC from the switch CLI directly connected to the supervisor engine console port, enter the **switch console** *mod* command. To exit from the MSFC CLI and return to the switch CLI, press **Ctrl-C** three times at the Router> prompt.

This example shows how to configure the Layer 3 VLAN interface on the MSFC:

```
Console> (enable) session 15
Trying Router-15...
Connected to Router-15.
Type ^C^C^C to switch back...
Router> config t
Router(config)# interface vlan 100
Router(config-if)# ip address 10.10.1.10 255.255.255.0
Router(config-if)# no shutdown
Router(config-if)# exit
Console> (enable)
```

Adding the SSL Services Module to the Corresponding VLAN

Note By default, the SSL Services Module is in trunking mode with native VLAN 1.

To add the SSL Services Module to the corresponding VLAN, enter this command:

Command	Purpose
`Console> (enable) set trunk mod/port vlan_id`	Configures the VLANs allowed over the trunk to the SSL Services Module. **Note** One of the allowed VLANs must be the admin VLAN.

This example shows how to add an SSL Services Module installed in slot 6 to a specific VLAN:

```
Console> (enable) set trunk 6/1 100
Adding vlans 100 to allowed list.
Console> (enable)
```

Verifying the Initial Configuration

To verify the configuration, enter one of these commands:

Command	Purpose
Console> **show spanntree** *vlan_ID*	Displays the spanning tree state for the specified VLAN.
Console> **show trunk** *mod/port*	Displays the trunk configuration.

Note In the following examples, the SSL Services Module is installed in slot 6.

This example shows how to verify that the module is in forwarding (FWD) state:

```
Console> show spantree 100
VLAN 100
Spanning tree mode          PVST+
Spanning tree type          ieee
Spanning tree enabled

Designated Root             00-06-2a-db-a5-01
Designated Root Priority    32768
Designated Root Cost        0
Designated Root Port        1/0
Root Max Age   20 sec   Hello Time 2  sec   Forward Delay 15 sec

Bridge ID MAC ADDR          00-06-2a-db-a5-01
Bridge ID Priority          32768
Bridge Max Age 20 sec   Hello Time 2  sec   Forward Delay 15 sec

Port                     Vlan Port-State    Cost      Prio Portfast Channel_id
------------------------ ---- ------------- --------- ---- -------- ----------
 6/1                     100  forwarding          100   32 enabled  033
Console>
```

This example shows how to verify that the VLAN information displayed matches the VLAN configuration:

```
Console> show trunk 6/1
* - indicates vtp domain mismatch
# - indicates dot1q-all-tagged enabled on the port
Port      Mode         Encapsulation  Status        Native vlan
--------  -----------  -------------  ------------  -----------
 6/1      nonegotiate  dot1q          trunking      1

Port      Vlans allowed on trunk
--------  ---------------------------------------------------------------------
 6/1      100
```

```
Port      Vlans allowed and active in management domain
--------  --------------------------------------------------------------------
 6/1      100

Port      Vlans in spanning tree forwarding state and not pruned
--------  --------------------------------------------------------------------
 6/1      100
```

Setting the Timezone

The supervisor engine sends the configured time and timezone informaion to the SSL Services Module. In some cases, the timezone information might be incorrect on the SSL Services Module.

To set the time and timezone on the SSL Services Module, perform this task:

Command	Purpose
webvpn(config)# **clock timezone** *zone hours-offset* [*minutes-offset*]	Sets the time zone for display purposes.

This example shows how to set the time zone to Pacific Standard Time (PST), which is 8 hours behind UTC:

```
ssl-proxy(config)# clock timezone PST -8
```

Recovering a Lost Password

Note You can download the password recovery script from the Cisco.com software center.

Note You must have access to the supervisor engine to perform the SSL Services Module password recovery procedures. To recover the enable password on the supervisor engine, refer to the software configuration guide for your software platform.

Note To run the password recovery script, the SSL Services Module must be in the application partition (AP).

Note The password recovery script is not compatible with SSL software release 1.x.

Caution For security reasons, all private keys are unusable after password recovery.

To recover a lost password on the SSL Services Module, perform this task:

	Command	Purpose
Step 1	`Console>(enable) session mod`	Sessions to the MSFC. This step is required if you are running Catalyst operating system software.
Step 2	`Router> enable`	Initiates enable mode enable.
Step 3	`Router# copy tftp: pclc#mod-fs:`	Downloads the script to the specified module.
Step 4	`ssl-proxy# copy startup-config running-config`	Saves the startup configuration into the running configuration.
Step 5	`ssl-proxy(config)# enable password password`	Specifies a local enable password.
Step 6	`ssl-proxy(config)# line vty starting-line-number ending-line-number`	Identifies a range of lines for configuration and enters line configuration mode.
Step 7	`ssl-proxy(config-line)# login`	Enables password checking at login.
Step 8	`ssl-proxy(config-line)# password password`	Specifies a password on the line.
Step 9	`ssl-proxy(config-line)# end`	Exits line configuration mode.
Step 10	`ssl-proxy# copy system:running-config nvram:startup-config`	Saves the configuration to NVRAM.
Step 11	`Router# hw-module module mod reset`	Resets the module.

The following example shows how to recover a lost password on the SSL module installed in slot 4:

- From the supervisor engine:

```
Console> (enable) session 15
Trying Router-15...
Connected to Router-15.
Type ^C^C^C to switch back...
Router>
Router> enable
Password:
Router# copy tftp: pclc#4-fs:
Address or name of remote host []? 10.1.1.100
Source filename []? images/c6svc-ssl-pwr.3-1-1.bin
Destination filename [images/c6svc-ssl-pwr.3-1-1.bin]?
Accessing tftp://10.1.1.100/images/c6svc-ssl-pwr.3-1-1.bin...
Loading images/c6svc-ssl-pwr.3-1-1.bin from 10.1.1.100(via Vlan999): !
[OK - 435 bytes]

435 bytes copied in 0.092 secs (4728 bytes/sec)
2003 Nov 10 21:53:25 %SYS-3-SUP_ERRMSGFROMPC:MP upgrade/Password Recovery started.
2003 Nov 10 21:53:25 %SYS-3-SUP_ERRMSGFROMPC:Uncompress of the file succeeded.
Continuing upgrade/recovery.
2003 Nov 10 21:53:25 %SYS-3-SUP_ERRMSGFROMPC:This file appears to be a
PasswordRecovery image. Continuing.
2003 Nov 10 21:53:25 %SYS-3-SUP_ERRMSGFROMPC:Extraction of password recovery image
succeeded.
2003 Nov 10 21:53:25 %SYS-3-SUP_ERRMSGFROMPC:Continuing with password recovery.

2003 Nov 10 21:55:03 %SYS-3-SUP_ERRMSGFROMPC:System in password recovery mode.
2003 Nov 10 21:55:03 %SYS-3-SUP_ERRMSGFROMPC:Please recover configuration and reset
board.

Router#
```

- From the SSL module console port:

```
ssl-proxy# copy system:startup-config nvram:running-config

ssl-proxy(config)# enable password cisco
ssl-proxy(config)# line vty 0 4
ssl-proxy(config-line)# login
ssl-proxy(config-line)# password cisco
ssl-proxy(config-line)# end
ssl-proxy# copy system:running-config nvram:startup-config
```

- From the supervisor engine:

```
Router# hw-module module 4 reset
```

- From the SSL module console port, import the keys from backup or regenerate the keys.

  See the "Configuring Keys and Certificates" section on page 3-3 for information about generating keys and importing keys.

Configuring the SSL Services Module

This chapter describes how to configure the SSL Services Module from the Command Line Interface (CLI) of the module:

Configuring Public Key Infrastructure

The SSL Services Module uses the SSL protocol to enable secure transactions of data through privacy, authentication, and data integrity. The SSL protocol relies upon a system of certificates, public keys, and private keys, known as a Public Key Infrastructure (PKI).

Tip For a detailed description of the elements and functions of a Public Key Infrastructure, see the Cisco IOS PKI Overview section of the *Cisco IOS Security Configuration Guide, Release 12.4* at this URL:

http://www.cisco.com/en/US/products/ps6350/products_configuration_guide_chapter09186a00804450cc.html

Certificates, which are similar to digital ID cards, verify the identity of the server to the clients and the clients to the server. Certificates are issued by certificate authorities (CAs, also known as trustpoints), and include the name of the entity to which the certificate was issued, the entity's public key, and the time stamps that indicate the certificate's expiration date. Your CA or trustpoint can be a third-party CA vendor or an internal CA, such as the Cisco IOS Certificate Server feature..

Public and private keys are the ciphers that are used to encrypt and decrypt information. The public key is shared without any restrictions, but the private key is never shared. Each public-private key pair works together; data that is encrypted with the public key can only be decrypted with the corresponding private key.

Each SSL Services Module acts as an SSL proxy for up to 256 SSL clients and servers. You must configure a pair of keys for each client or server in order to apply for a certificate for authentication.

We recommend that the certificates be stored in NVRAM so the module does not need to query the certificate authority at startup to obtain the certificates or to automatically enroll. See the "Saving Your Configuration" section on page 3-28 for more information.

The SSL Services Module authenticates certificates that it receives from external devices when you configure the SSL Services Module as an SSL server and you configure the server proxy to authenticate the client certificate, or when you configure the SSL Services Module as an SSL client. The SSL Services Module validates the start time, end time, and the signature on the certificate received.

A valid certificate may have been revoked if the key pair has been compromised. See the "Checking the Certificate Status" section on page 3-58 for information on configuring Online Certificate Status Protocol (OCSP) and certificate revocation lists (CRLs).

The certificate can also be filtered by matching certain certificate attribute values with access control list (ACL) maps. Only authenticated certificates that are issued by trusted certificate authorities are accepted. See the "Certificate Security Attribute-Based Access Control" section on page 3-65 for information on ACLs.

Note Only the certificate is authenticated, not the sender of the certificate. As part of the SSL handshake, the certificate sender is challenged for ownership of the private key that corresponds to the public key published in the certificate. If the challenge fails, the SSL handshake is aborted by the SSL Services Module.

However, the SSL Services Module cannot verify that the sender of the certificate is the expected end user or host of the communication session. To authenticate the end user or host, additional validation is necessary during the data phase, using a user name and password, bank account number, credit card number, or mother's maiden name.

If the certificate sender is an SSL client, the SSL Services Module can extract attributes from the client certificate and insert these attributes into the HTTP header during the data phase. The server system that receives these headers can further examine the subject name of the certificate and other attributes and then determine the authenticity of the end user or host. See the "HTTP Header Insertion" section on page 4-7 for information on configuring HTTP header insertion. See the "Client Certificate Authentication" section on page 3-51 for information on configuring client certificate authentication.

These sections describe how to configure the public key infrastructure (PKI):

Configuring Keys and Certificates

You can configure keys and certificates using one of the following methods:

- If you are using Simple Certificate Enrollment Protocol (SCEP), configure the keys and certificates by doing the following:
  - Generate a key pair.
  - Declare the trustpoint.
  - Get the certificate authority certificate.
  - Send an enrollment request to a certificate authority on behalf of the SSL server.

  See the "Configuring the Trustpoint Using SCEP" section on page 3-5 for details.
- If you are not using SCEP, configure the keys and certificates using the manual certificate enrollment (TFTP and cut-and-paste) feature by doing the following:
  - Generate or import a key pair.
  - Declare the trustpoint.
  - Get the certificate authority certificate and enroll the trustpoint using TFTP or cut-and-paste to create a PKCS10 file.
  - Request the SSL server certificate offline using the PKCS10 package.
  - Import the SSL server certificate using TFTP or cut-and-paste.

  See the "Manual Certificate Enrollment" section on page 3-11 for details.
- If you are using an external PKI system, do the following:
  - Generate PKCS12 or PEM files.
  - Import this file to the module.

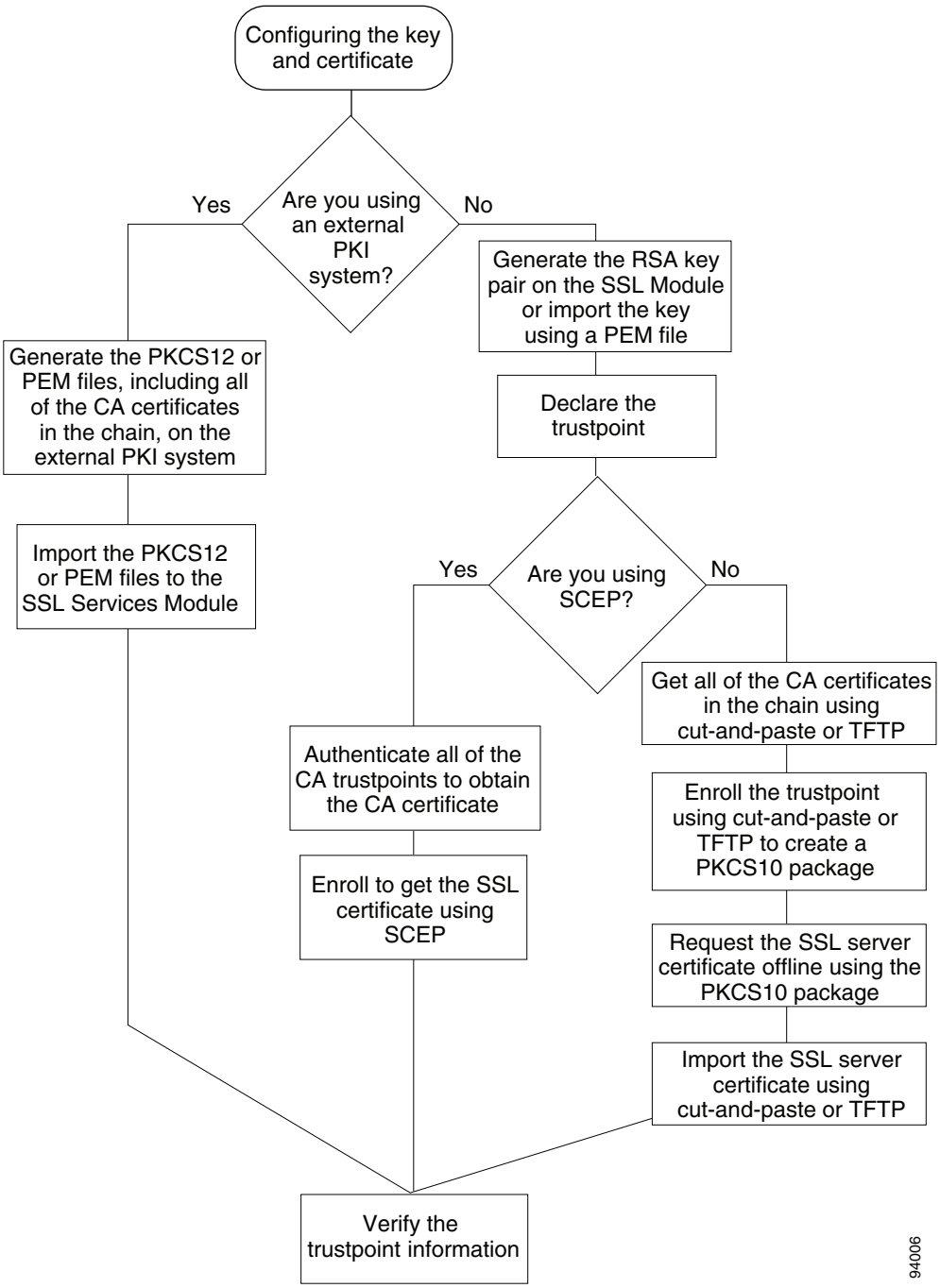
  See the "Importing and Exporting Key Pairs and Certificates" section on page 3-19 for details.

An external PKI system is a server or a PKI administration system that generates key pairs and enrolls for certificates from a certificate authority or a key and certificate archival system. The Public-Key Cryptography Standards (PKCS) specifies the transfer syntax for personal identity information, including the private keys and certificates. This information is packaged into an encrypted file. To open the encrypted file, you must know a pass phrase. The encryption key is derived from the pass phrase.

Note You do not need to configure a trustpoint before importing the PKCS12 or PEM files. If you import keys and certificates from PKCS12 or PEM files, the trustpoint is created automatically, if it does not already exist.

See Figure 3-1 for an overview on configuring keys and certificates.

Figure 3-1 Key and Certificate Configuration Overview



Configuring the Trustpoint Using SCEP

To configure a trustpoint using SCEP, complete the following tasks:

- Generating RSA Key Pairs, page 3-5
- Declaring the Trustpoint, page 3-7
- Obtaining the Certificate Authority Certificate, page 3-8
- Requesting a Certificate, page 3-9

Generating RSA Key Pairs

Note The first key pair generated enables SSH on the module. If you are using SSH, configure a key pair for SSH. See the "Configuring SSH" section on page 2-4.

RSA is the public key cryptographic system developed by Ron Rivest, Adi Shamir, and Leonard Aldeman. RSA algorithm is widely used by certificate authorities and SSL servers to generate key pairs. Each certificate authority and each SSL server has its own RSA key pair. The SSL server sends its public key to the certificate authority when enrolling for a certificate. The SSL server uses the certificate to prove its identity to clients when setting up the SSL session.

The SSL server keeps the private key in a secure storage, and sends only the public key to the certificate authority, which uses its private key to sign the certificate that contains the server's public key and other identifying information about the server.

Each certificate authority keeps the private key secret and uses the private key to sign certificates for its subordinate certificate authorities and SSL servers. The certificate authority has a certificate that contains its public key.

The certificate authorities form a hierarchy of one or more levels. The top-level certificate authority is called the root certificate authority. The lower level certificate authorities are called intermediate or subordinate certificate authorities. The root certificate authority has a self-signed certificate, and it signs the certificate for the next level subordinate certificate authority, which in turn signs the certificate for the next lower level certificate authority, and so on. The lowest level certificate authority signs the certificate for the SSL server.

Note The SSL Services Module supports up to eight levels of certificate authority (one root certificate authority and up to seven subordinate certificate authorities). For an example of a three-level (3-tier) enrollment, see the "Example of Three-Tier Certificate Authority Enrollment" section on page 3-9.

These certificates form a chain with the server certificate at the bottom and the root certificate authority's self-signed certificate at the top. Each signature is formed by using the private key of the issuing certificate authority to encrypt a hash digest of the certificate body. The signature is attached to the end of the certificate body to form the complete certificate.

When setting up an SSL session, the SSL server sends its certificate chain to the client. The client verifies the signature of each certificate up the chain by retrieving the public key from the next higher-level certificate to decrypt the signature attached to the certificate body. The decryption result is compared with the hash digest of the certificate body. Verification terminates when one of the certificate authority certificates in the chain matches one of the trusted certificate authority certificates stored in the client's own database.

If the top-level certificate authority certificate is reached in the chain, and there is no match of trusted self-signed certificates, the client may terminate the session or prompt the user to view the certificates and determine if they can be trusted.

After the SSL client authenticates the server, it uses the public key from the server certificate to encrypt a secret and send it over to the server. The SSL server uses its private key to decrypt the secret. Both sides use the secret and two random numbers they exchanged to generate the key material required for the rest of the SSL session for data encryption, decryption, and integrity checking.

Note The SSL Services Module supports only general-purpose keys.

When you generate general-purpose keys, only one pair of RSA keys is generated. Named key pairs allow you to have multiple RSA key pairs, enabling the Cisco IOS software to maintain a different key pair for each identity certificate. We recommend that you specify a name for the key pairs.

Note The generated key pair resides in system memory (RAM). Key pairs will be lost on power failure or module reset. You must enter the **copy system:running-config nvram:startup-config** command to save the running configuration, as well as save the key pairs to the private configuration file in the module NVRAM.

To generate RSA key pairs, perform this task:

Command	Purpose
`ssl-proxy(config)# crypto key generate rsa general-keys label` *key-label* [**exportable**[1]] [**modulus** *size*]	Generates RSA key pairs.

1. The **exportable** keyword specifies that the key is allowed to be exported. You can specify that a key is exportable during key generation. Once the key is generated as either exportable or not exportable, it cannot be modified for the life of the key.

Note When you generate RSA keys, you are prompted to enter a modulus length in bits. The SSL Services Module supports modulus lengths of 512, 768, 1024, 1536, and 2048 bits. Although you can specify 512 or 768, we recommend a minimum modulus length of 1024. A longer modulus takes longer to generate and takes longer to use, but it offers stronger security.

This example shows how to generate general-purpose RSA keys:

```
ssl-proxy(config)# crypto key generate rsa general-keys label kp1 exportable

The name for the keys will be: kp1

Choose the size of the key modulus in the range of 360 to 2048 for your
  General Purpose Keys. Choosing a key modulus greater than 512 may take
  a few minutes.

How many bits in the modulus [512]: 1024

Generating RSA keys.... [OK].
```

Note After you generate a key pair, you can test the SSL service by generating a self-signed certificate. To generate a self-signed certificate for testing, see the "Generating a Self-Signed Certificate" section on page C-1.

Declaring the Trustpoint

You should declare one trustpoint to be used by the module for each certificate.

To declare the trustpoint that your module uses and specify characteristics for the trustpoint, perform this task beginning in global configuration mode:

	Command	Purpose
Step 1	`ssl-proxy(config)# crypto pki trustpoint` *trustpoint-label*[1]	Declares the trustpoint that your module should use and enters ca-trustpoint configuration mode.
Step 2	`ssl-proxy(ca-trustpoint)# rsakeypair` *key-label*	Specifies which key pair to associate with the certificate.
Step 3	`ssl-proxy(ca-trustpoint)# enrollment [mode ra] [retry [period` *minutes*`] [count` *count*`]] url` *url*	Specifies the enrollment parameters for your certificate authority.
Step 4	`ssl-proxy(ca-trustpoint)# ip-address` *server_ip_addr*	(Optional) Specifies the IP address of the proxy service which will use this certificate[2].
Step 5	`ssl-proxy(ca-trustpoint)# revocation-check` *method1* `[`*method2*`[`*method3*`]]`	Checks the revocation status of a certificate. • **crl**—Certificate checking is performed by a CRL. This is the default option. • **none**—Certificate checking is ignored. • **ocsp**—Certificate checking is performed by an OCSP server. See the "Checking the Certificate Status" section on page 3-58 for information.
Step 6	`ssl-proxy(ca-trustpoint)# subject-name` *line*[3, 4]	(Optional) Configures the host name of the proxy service[5].
Step 7	`ssl-proxy(ca-trustpoint)# password` *password*	(Optional) Configures a challenge password.
Step 8	`ssl-proxy(ca-trustpoint)# exit`	Exits ca-trustpoint configuration mode.

1. The *trustpoint-label* should match the *key-label* of the keys; however, this is not a requirement.
2. Some web browsers compare the IP address in the SSL server certificate with the IP address that might appear in the URL. If the IP addresses do not match, the browser may display a dialog box and ask the client to accept or reject this certificate.
3. For example, **subject-name CN=***server1.domain2.com*, where *server1* is the name of the SSL server that appears in the URL. The **subject-name** command uses the Lightweight Directory Access Protocol (LDAP) format.
4. Arguments specified in the subject name must be enclosed in quotation marks if they contain a comma. For example, **O="Cisco, Inc."**
5. Some browsers compare the CN field of the subject name in the SSL server certificate with the hostname that might appear in the URL. If the names do not match, the browser may display a dialog box and ask the client to accept or reject the certificate. Also, some browsers will reject the SSL session setup and silently close the session if the CN field is not defined in the certificate.

This example shows how to declare the trustpoint PROXY1 and verify connectivity:

```
ssl-proxy(config)# crypto pki trustpoint PROXY1
ssl-proxy(ca-trustpoint)# rsakeypair PROXY1
ssl-proxy(ca-trustpoint)# enrollment url http://exampleCA.cisco.com
ssl-proxy(ca-trustpoint)# ip-address 10.0.0.1
ssl-proxy(ca-trustpoint)# password password
ssl-proxy(ca-trustpoint)# ocsp url http://myocspserver:81
ssl-proxy(ca-trustpoint)# revocation-check ocsp none
ssl-proxy(ca-trustpoint)# serial-number
ssl-proxy(ca-trustpoint)# subject-name C=US, ST=California, L=San Jose, O=Cisco, OU=Lab,
CN=host1.cisco.com
ssl-proxy(ca-trustpoint)# end
ssl-proxy# ping example.cisco.com
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 20.0.0.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/1/4 ms
ssl-proxy#
```

Obtaining the Certificate Authority Certificate

For each trustpoint, you must obtain a certificate that contains the public key of the certificate authority; multiple trustpoints can use the same certificate authority.

Note Contact the certificate authority to obtain the correct fingerprint of the certificate and verify the fingerprint displayed on the console.

To obtain the certificate that contains the public key of the certificate authority, perform this task in global configuration mode:

Command	Purpose
`ssl-proxy(config)# crypto pki authenticate` *trustpoint-label*	Obtains the certificate that contains the public key of the certificate authority. Enter the same *trustpoint_label* that you entered when declaring the trustpoint.

This example shows how to obtain the certificate of the certificate authority:

```
ssl-proxy(config)# crypto pki authenticate PROXY1
Certificate has the following attributes:
Fingerprint: A8D09689 74FB6587 02BFE0DC 2200B38A
% Do you accept this certificate? [yes/no]: y
Trustpoint CA certificate accepted.
ssl-proxy(config)# end
ssl-proxy#
```

Requesting a Certificate

You must obtain a signed certificate from the certificate authority for each trustpoint.

To request signed certificates from the certificate authority, perform this task in global configuration mode:

Command	Purpose
ssl-proxy(config)# **crypto pki enroll** *trustpoint-label*[1]	Requests a certificate for the trustpoint.

1. You have the option to create a challenge password that is not saved with the configuration. This password is required in the event that your certificate needs to be revoked, so you must remember this password.

Note If your module or switch reboots after you have entered the **crypto pki enroll** command but before you have received the certificates, you must reenter the command and notify the certificate authority administrator.

This example shows how to request a certificate:

```
ssl-proxy(config)# crypto pki enroll PROXY1
%
% Start certificate enrollment..

% The subject name in the certificate will be: C=US, ST=California, L=San Jose, O=Cisco,
OU=Lab, CN=host1.cisco.com
% The subject name in the certificate will be: host.cisco.com
% The serial number in the certificate will be: 00000000
% The IP address in the certificate is 10.0.0.1

% Certificate request sent to Certificate Authority
% The certificate request fingerprint will be displayed.
% The 'show crypto pki certificate' command will also show the fingerprint.
Fingerprint:  470DE382 65D8156B 0F84C2AF 4538B913

ssl-proxy(config)# end
```

After you configure the trustpoint, see the "Verifying Certificates and Trustpoints" section on page 3-28 to verify the certificate and trustpoint information.

Example of Three-Tier Certificate Authority Enrollment

The SSL Services Module supports up to eight levels of certificate authority (one root certificate authority and up to seven subordinate certificate authorities).

The following example shows how to configure three levels of certificate authority:

Generating the Keys

```
ssl-proxy(onfig)# crypto key generate rsa general-keys label key1 exportable
The name for the keys will be:key1
Choose the size of the key modulus in the range of 360 to 2048 for your
  General Purpose Keys. Choosing a key modulus greater than 512 may take
  a few minutes.

How many bits in the modulus [512]:1024
% Generating 1024 bit RSA keys ...[OK]
```

Defining the Trustpoints

```
ssl-proxy(config)# crypto pki trustpoint 3tier-root
ssl-proxy(ca-trustpoint)# enrollment url tftp://10.1.1.1
ssl-proxy(ca-trustpoint)#
ssl-proxy(ca-trustpoint)# exit
ssl-proxy(config)# crypto pki trustpoint 3tier-sub1
ssl-proxy(ca-trustpoint)# enrollment url tftp://10.1.1.2
ssl-proxy(ca-trustpoint)#
ssl-proxy(ca-trustpoint)# exit
ssl-proxy(config)# crypto pki trustpoint tp-proxy1
ssl-proxy(ca-trustpoint)# enrollment url tftp://10.1.1.3
ssl-proxy(ca-trustpoint)# serial-number
ssl-proxy(ca-trustpoint)# password cisco
ssl-proxy(ca-trustpoint)# subject CN=ste.cisco.com
ssl-proxy(ca-trustpoint)# rsakeypair key1
ssl-proxy(ca-trustpoint)# show
 enrollment url tftp://10.1.1.3
 serial-number
 password 7 02050D480809
 subject-name CN=ste.cisco.com
 rsakeypair key1
end

ssl-proxy(ca-trustpoint)# exit
```

Authenticating the Three Certificate Authorities (One Root And Two Subordinate Certificate Authorities)

```
ssl-proxy(config)# crypto pki authenticate 3tier-root
Certificate has the following attributes:
Fingerprint:84E470A2 38176CB1 AA0476B9 C0B4F478
% Do you accept this certificate? [yes/no]:yes
Trustpoint CA certificate accepted.
ssl-proxy(config)#
ssl-proxy(config)# crypto pki authenticate 3tier-sub1
Certificate has the following attributes:
Fingerprint:FE89FB0D BF8450D7 9934C926 6C66708D
Certificate validated - Signed by existing trustpoint CA certificate.
Trustpoint CA certificate accepted.
ssl-proxy(config)#
ssl-proxy(config)# crypto pki authenticate tp-proxy1
Certificate has the following attributes:
Fingerprint:6E53911B E29AE44C ACE773E7 26A098C3
Certificate validated - Signed by existing trustpoint CA certificate.
Trustpoint CA certificate accepted.
```

Enrolling with the Third Level Certificate Authority

```
ssl-proxy(config)# crypto pki enroll tp-proxy1
%
% Start certificate enrollment ..

% The fully-qualified domain name in the certificate will be:ste.
% The subject name in the certificate will be:ste.
% The serial number in the certificate will be:B0FFF0C2
% Include an IP address in the subject name? [no]:
Request certificate from CA? [yes/no]:yes
% Certificate request sent to Certificate Authority
% The certificate request fingerprint will be displayed.
% The 'show crypto pki certificate' command will also show the fingerprint.

ssl-proxy(config)#    Fingerprint: 74390E57 26F89436 6FC52ABE 24E23CD9

ssl-proxy(config)#
*Apr 18 05:10:20.963:%CRYPTO-6-CERTRET:Certificate received from Certificate Authority
```

Manual Certificate Enrollment

The Manual Certificate Enrollment (TFTP and Cut-and-Paste) feature allows you to generate a certificate request and accept certificate authority certificates as well as router certificates. These tasks are accomplished with a TFTP server or manual cut-and-paste operations. You may want to use TFTP or manual cut-and-paste enrollment in the following situations:

- Your certificate authority does not support Simple Certificate Enrollment Protocol (SCEP) (which is the most commonly used method for sending and receiving requests and certificates).
- A network connection between the router and certificate authority is not possible (which is how a router running Cisco IOS software obtains its certificate).

Configure the Manual Certificate Enrollment (TFTP and cut-and-paste) feature as described at this URL:

http://www.cisco.com/univercd/cc/td/doc/product/software/ios122/122newft/122t/122t13/ftmancrt.htm

Note If you have configured the **revocation-check crl** command, and if the certificate revocation list (CRL) fails to download because the CRL server is unreachable or the CRL download path does not exist, the certificate might fail to import. You should make sure all trustpoints that are linked to the import process are able to download the CRL. If the CRL path does not exist, or if the CRL server is unreachable, then you should enter the **revocation-check none** command for all trustpoints that are linked to the import process. Enter the **show crypto pki certificates** command to display information for all certificates, and obtain a list of associated trustpoints from the display of the certificate authority certificate. Enter the **revocation-check none** command for all these trustpoints.

For example, in a three-tier certificate authority hierarchy (root CA, subordinate CA1, and subordinate CA2), when you import the subordinate CA1 certificate, enter the **revocation-check none** command for all the trustpoints associated with root CA. Similarly, when you import the subordinate CA2 certificate, enter the **revocation-check none** command for all the trustpoints associated with root CA and subordinate CA1.

After you successfully import the certificate, you can restore the original CRL options on the trustpoints.

Example 1: Configuring Certificate Enrollment Using TFTP (One-Tier Certificate Authority)

1. Configure the trustpoint:

```
ssl-proxy# configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
ssl-proxy(config)# crypto pki trustpoint tftp_example
ssl-proxy(ca-trustpoint)# enrollment url tftp://10.1.1.2/win2k
ssl-proxy(ca-trustpoint)# rsakeypair pair3
ssl-proxy(ca-trustpoint)# exit
```

2. Request a certificate for the trustpoint:

```
ssl-proxy(config)# crypto pki enroll tftp_example
% Start certificate enrollment ..

% The fully-qualified domain name in the certificate will be: ssl-proxy.cisco.com
% The subject name in the certificate will be: ssl-proxy.cisco.com
% Include the router serial number in the subject name? [yes/no]: yes
% The serial number in the certificate will be: 00000000
% Include an IP address in the subject name? [no]:
```

```
Send Certificate Request to tftp server? [yes/no]: yes
% Certificate request sent to TFTP Server
% The certificate request fingerprint will be displayed.
% The 'show crypto pki certificate' command will also show the fingerprint.
ssl-proxy(config)#    Fingerprint:  D012D925 96F4B5C9 661FEC1E 207786B7
!!
```

3. Obtain the certificate that contains the public key of the certificate authority:

```
ssl-proxy(config)# crypto pki auth tftp_example
Loading win2k.ca from 10.1.1.2 (via Ethernet0/0.168): !
[OK - 1436 bytes]

Certificate has the following attributes:
Fingerprint: 2732ED87 965F8FEB F89788D4 914B877D
% Do you accept this certificate? [yes/no]: yes
Trustpoint CA certificate accepted.
ssl-proxy(config)#
```

4. Import the server certificate:

```
ssl-proxy(config)# crypto pki import tftp_example cert
% The fully-qualified domain name in the certificate will be: ssl-proxy.cisco.com
Retrieve Certificate from tftp server? [yes/no]: yes
% Request to retrieve Certificate queued

ssl-proxy(config)#
Loading win2k.crt from 10.1.1.2 (via Ethernet0/0.168): !
[OK - 2112 bytes]

ssl-proxy(config)#
*Apr 15 12:02:33.535: %CRYPTO-6-CERTRET: Certificate received from Certificate
Authority
ssl-proxy(config)#
```

Example 2: Configuring Certificate Enrollment Using Cut-and-Paste (One-Tier Certificate Authority)

1. Generate the RSA key pair:

```
ssl-proxy(config)# crypto key generate rsa general-keys label CSR-key exportable
The name for the keys will be:CSR-key
Choose the size of the key modulus in the range of 360 to 2048 for your
  General Purpose Keys. Choosing a key modulus greater than 512 may take
  a few minutes.

How many bits in the modulus [512]:1024
% Generating 1024 bit RSA keys ...[OK]
```

2. Configure the trustpoints:

```
ssl-proxy(config)# crypto pki trustpoint CSR-TP
ssl-proxy(ca-trustpoint)# rsakeypair CSR-key
ssl-proxy(ca-trustpoint)# serial
ssl-proxy(ca-trustpoint)# subject-name CN=abc, OU=hss, O=cisco
ssl-proxy(ca-trustpoint)# enrollment terminal
ssl-proxy(ca-trustpoint)# exit
```

3. Request a certificate for the trustpoint:

```
ssl-proxy(config)# crypto pki enroll CSR-TP
% Start certificate enrollment ..

% The subject name in the certificate will be:CN=abc, OU=hss, O=cisco
% The fully-qualified domain name in the certificate will be:ssl-proxy.cisco.com
% The subject name in the certificate will be:ssl-proxy.cisco.com
% The serial number in the certificate will be:B0FFF22E
% Include an IP address in the subject name? [no]:no
Display Certificate Request to terminal? [yes/no]:yes
Certificate Request follows:

MIIBwjCCASsCAQAwYTEOMAwGA1UEChMFY2lzY28xDDAKBgNVBAsTA2hzczEMMAoG
A1UEAxMDYWJjMTMwDwYDVQQFEwhCMEZGRjIyRTAgBgkqhkiG9w0BCQIWE3NzbC1w
cm94eS5jaXNjby5jb20wgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBALt7O6tt
30lBVVK1qAE/agsuzIaa15YZft3bDb9t3pPncKh0ivBTgVKpJiLPWGZPjdbtejxQ
tYSF77R1pmhK0WSKPuu7fJPYr/Cbo80OUzkRAgMBAAGgITAfBgkqhkiG9w0BCQ4x
EjAQMA4GA1UdDwEB/wQEAwIFoDANBgkqhkiG9w0BAQQFAAOBgQC2GIX06/hihXHA
DA5sOpxgLsO1rMP8PF4bZDdlpWLVBSOrp4S1L7hH9P2NY9rgZAJhDTRfGGm179JY
GOtUuCyPYPkpb0S5VGTUrHvvUWekleKq2d91kfgbkRmJmHBaB2Ev5DNBcV11SIMX
RULG7oUafU6sxnDWqbMseToF4WrLPg==

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

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

4. Import the certificate authority certificate:

```
ssl-proxy(config)# crypto pki authenticate CSR-TP

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

-----BEGIN CERTIFICATE-----
MIICxzCCAjCgAwIBAgIBADANBgkqhkiG9w0BAQQFADBSMQswCQYDVQQGEwJBVTET
MBEGA1UECBMKU29tZS1TdGF0ZTEhMB8GA1UEChMYSW50ZXJuZXQgV2lkZ2l0cyBQ
dHkgTHRkMQswCQYDVQQDEwJjYTAeFw0wMzA2MjYyMjM4MDlaFw0wODEyMTYyMjM4
MDlaMFIxCzAJBgNVBAYTAkFVMRMwEQYDVQQIEwpTb211LVN0YXRlMSEwHwYDVQQK
ExhJbnRlcm5ldCBXaWRnaXRzIFB0eSBMdGQxCzAJBgNVBAMTAmNhMIGfMA0GCSqG
SIb3DQEBAQUAA4GNADCBiQKBgQCcG9ObqOLmf0cASkF48jz8X7ZQxT1H68OQKNC3
ks95vkGbOAa/1/R4ACQ3s9iPkcGQVqi4Dv8/iNG/1mQo8HBwtR9VgG0l8IGBbuiZ
dlarYnQHUz6Bm/HzE1RXVOY/VmyPOVevYy8/cYhwx/xOE9BYQOyP15Chi8nhIS5F
+WWoHQIDAQABo4GsMIGpMB0GA1UdDgQWBBS4Y+/lSXKDrw5N5m/tgCzu/W81PDB6
BgNVHSMEczBxgBS4Y+/lSXKDrw5N5m/tgCzu/W81PKFWpFQwUjELMAkGA1UEBhMC
QVUxEzARBgNVBAgTClNvbWUtU3RhdGUxITAfBgNVBAoTGEludGVybmV0IFdpZGdp
dHMgUHR5IEx0ZDELMAkGA1UEAxMCY2GCAQAwDAYDVR0TBAUwAwEB/zANBgkqhkiG
9w0BAQQFAAOBgQB/rPdLFVuycbaJQucdFQG7kl/XBNI7aY3IL3Lkeumt/nXD+eCn
RpYE5WWY8X1Aizqnj4bqFdqPqYdD7Lg8viwqm2tQmU6zCsdaKhL1J7FCWbfs2+Z5
oNV2Vsqx0Ftnf8en/+HtyS2AdXHreThfgkXz3euXD0ISMFVKRy81o4EdzA==
-----END CERTIFICATE-----

Certificate has the following attributes:
Fingerprint:B8B35B00 095573D0 D3B8FA03 B6CA8934
% Do you accept this certificate? [yes/no]:yes
Trustpoint CA certificate accepted.
% Certificate successfully imported

ssl-proxy(config)#
```

5. Import the server certificate (the server certificate is issued by the certificate authority whose certificate is imported in Step 4):

```
ssl-proxy(config)# crypto pki import CSR-TP certificate
% The fully-qualified domain name in the certificate will be:ssl-proxy.cisco.com

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

-----BEGIN CERTIFICATE-----
MIIB7TCCAVYCAQQwDQYJKoZIhvcNAQEEBQAwUjELMAkGA1UEBhMCQVUxEzARBgNV
BAgTClNvbWUtU3RhdGUxITAfBgNVBAoTGEludGVybmV0IFdpZGdpdHMgUHR5IEx0
ZDELMAkGA1UEAxMCY2EwHhcNMDMxMTIwMDAxMzE2WhcNMDQxMTE5MDAxMzE2WjAs
MQ4wDAYDVQQKEwVjaXNjbzEMMAoGA1UECxMDaHNzMQwwCgYDVQQDEwNhYmMwgZ8w
DQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBALt7O6tt30lBVVK1qAE/agsuzIaa15YZ
ft3bDb9t3pPncKh0ivBTgVKpJiLPWGZPjdbtejxQksuSY589V+GMDrO9B4Sxn+5N
p2bQmd745NvI4gorNRvXcdjmE+/SzE+bBSBcKAwNtYSF77R1pmhK0WSKPuu7fJPY
r/Cbo80OUzkRAgMBAAEwDQYJKoZIhvcNAQEEBQADgYEAjqJ9378P6Gz69Ykplw06
Powp+2rbe2iFBrE1xE09BL6G6vzcBQgb5W4uwqxe7SIHrHsS0/7Be3zeJnlOseWx
/KVj7I02iPgrwUa9DLavwrTyaa0KtTpti/i5nIwTNh5xkp2bBJQikD4TEK7HAvXf
HQ9SyB3YZJk/Bjp6/eFHEfU=
-----END CERTIFICATE-----

% Router Certificate successfully imported

ssl-proxy(config)#^Z
```

Example 3: Configuring Certificate Enrollment Using TFTP (Three-Tier Certificate Authority)

1. Generate the RSA key pair:

```
ssl-proxy(config)# crypto key generate rsa general-keys label test-3tier exportable
The name for the keys will be:test-3tier
Choose the size of the key modulus in the range of 360 to 2048 for your
  General Purpose Keys. Choosing a key modulus greater than 512 may take
  a few minutes.

How many bits in the modulus [512]:1024
% Generating 1024 bit RSA keys ...[OK]
```

2. Configure the trustpoint:

```
ssl-proxy(config)# crypto pki trustpoint test-3tier
ssl-proxy(ca-trustpoint)# serial-number
ssl-proxy(ca-trustpoint)# password cisco
ssl-proxy(ca-trustpoint)# subject CN=test-3tier, OU=hss, O=Cisco
ssl-proxy(ca-trustpoint)# rsakeypair test-3tier
ssl-proxy(ca-trustpoint)# enrollment url tftp://10.1.1.3/test-3tier
ssl-proxy(ca-trustpoint)# exit
```

3. Generate the certificate signing request (CSR) and send it to the TFTP server:

```
ssl-proxy(config)# crypto pki enroll test-3tier
%
% Start certificate enrollment ..

% The subject name in the certificate will be:CN=test-3tier, OU=hss, O=Cisco
% The fully-qualified domain name in the certificate will be:ssl-proxy.cisco.com
% The subject name in the certificate will be:ssl-proxy.cisco.com
% The serial number in the certificate will be:B0FFF22E
% Include an IP address in the subject name? [no]:
Send Certificate Request to tftp server? [yes/no]:yes
% Certificate request sent to TFTP Server
% The certificate request fingerprint will be displayed.
```

```
% The 'show crypto pki certificate' command will also show the fingerprint.

ssl-proxy(config)# Fingerprint: 19B07392 319B2ACF F8FABE5C 52798971

ssl-proxy(config)#
!!
```

4. Use the CSR to acquire the SSL certificate offline from the third-level certificate authority.
5. Authenticate the three certificate authorities (one root and two subordinate certificate authorities):

```
ssl-proxy(config)# crypto pki trustpoint test-1tier
ssl-proxy(ca-trustpoint)# enrollment url tftp://10.1.1.3/test-1tier
ssl-proxy(ca-trustpoint)# revocation-check none
ssl-proxy(ca-trustpoint)# exit
ssl-proxy(config)# crypto pki authenticate test-1tier
Loading test-1tier.ca from 10.1.1.3 (via Ethernet0/0.172):!
[OK - 1046 bytes]

Certificate has the following attributes:
Fingerprint:AC6FC55E CC29E891 0DC3FAAA B4747C10
% Do you accept this certificate? [yes/no]:yes
Trustpoint CA certificate accepted.

ssl-proxy(config)# crypto pki trustpoint test-2tier
ssl-proxy(ca-trustpoint)# enrollment url tftp://10.1.1.3/test-2tier
ssl-proxy(ca-trustpoint)# revocation-check none
ssl-proxy(ca-trustpoint)# exit
ssl-proxy(config)# crypto pki authenticate test-2tier
Loading test-2tier.ca from 10.1.1.3 (via Ethernet0/0.172):!
[OK - 1554 bytes]

Certificate has the following attributes:
Fingerprint:50A986F6 B471B82D E11B71FE 436A9BE6
Certificate validated - Signed by existing trustpoint CA certificate.
Trustpoint CA certificate accepted.

ssl-proxy(config)# crypto pki authenticate test-3tier
Loading test-3tier.ca from 10.1.1.3 (via Ethernet0/0.172):!
[OK - 1545 bytes]

Certificate has the following attributes:
Fingerprint:2F2E44AC 609644FA 5B4B6B26 FDBFE569
Certificate validated - Signed by existing trustpoint CA certificate.
Trustpoint CA certificate accepted.
```

6. Import the server certificate:

```
ssl-proxy(config)# crypto pki import test-3tier certificate
% The fully-qualified domain name in the certificate will be:ssl-proxy.cisco.com
Retrieve Certificate from tftp server? [yes/no]:yes
% Request to retrieve Certificate queued

ssl-proxy(config)#
Loading test-3tier.crt from 10.1.1.3 (via Ethernet0/0.172):!
[OK - 1608 bytes]

ssl-proxy(config)#
*Nov 25 21:52:36.299:%CRYPTO-6-CERTRET:Certificate received from Certificate Authority
ssl-proxy(config)# ^Z
```

Example 4: Configuring Certificate Enrollment Using Cut-and-Paste (Three-Tier Certificate Authority)

1. Generate the RSA key pair:

```
ssl-proxy(config)# crypto key generate rsa general-keys label tp-proxy1 exportable
The name for the keys will be:tp-proxy1
Choose the size of the key modulus in the range of 360 to 2048 for your
  General Purpose Keys. Choosing a key modulus greater than 512 may take
  a few minutes.

How many bits in the modulus [512]:1024
% Generating 1024 bit RSA keys ...[OK]
```

2. Configure the trustpoint:

```
ssl-proxy(config)# crypto pki trustpoint tp-proxy1
ssl-proxy(ca-trustpoint)# enrollment ter
ssl-proxy(ca-trustpoint)# rsakeypair tp-proxy1
ssl-proxy(ca-trustpoint)# serial
ssl-proxy(ca-trustpoint)# subject-name CN=test
ssl-proxy(ca-trustpoint)# exit
```

3. Request a certificate for the trustpoint:

```
ssl-proxy(config)# crypto pki enroll tp-proxy1
% Start certificate enrollment ..

% The subject name in the certificate will be:CN=test
% The fully-qualified domain name in the certificate will be:ssl-proxy.
% The subject name in the certificate will be:ssl-proxy.
% The serial number in the certificate will be:B0FFF14D
% Include an IP address in the subject name? [no]:no
Display Certificate Request to terminal? [yes/no]:yes
Certificate Request follows:

MIIBnDCCAQUCAQAwOzENMAsGA1UEAxMEdGVzdDEqMA8GA1UEBRMIQjBGRkYxNEQw
FwYJKoZIhvcNAQkCFgpzc2wtcHJveHkuMIGfMA0GCSqGSIb3DQEBAQUAA4GNADCB
iQKBgQDFx1ol9IXoAx4fyUhaXH6s4p5t9soIZ1gvLtVX6Fp6zfuX47os5TGJH/IX
zV9B4e5Kv+wlMD0AvTh+/tvyAP3TMpCdpHYosd2VaTIgExpHf4M5Ruh8IebVKV25
rraIpNiS0PvPLFcrw4UfJVNpsc2XBxBhpT+FS9y67LqlhfSN4wIDAQABoCEwHwYJ
KoZIhvcNAQkOMRIwEDAOBgNVHQ8BAf8EBAMCBaAwDQYJKoZIhvcNAQEEBQADgYEA
kOIjd1KNJdKLMf33YELRd3MW/ujJIuiT1J8RYVbw1eE8JQf68TTdKiYqzQcoMgsp
ez3vSPxXFZ/c6naXdVyrTikTX3GZ1mu+UOvV6/Jaf5QcXa9tAi3fgyguV7jQMPjk
Qj2GrwhXjcqZGOMBh6Kq6s5UPsIDgrL036I42B6B3EQ=

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

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

4. Get the certificate request from Step 3 signed by a third-level certificate authority.

5. Define and import all certificate authorities (one root and two subordinate certificate authorities).
   a. Define two truspoints for root certificate authority and subordinate 1 certificate authority.

Note We will use **tp-proxy1** to import the subordinate 2 certificate authority certificate.

```
ssl-proxy(config)# crypto pki trustpoint 3tier-root
ssl-proxy(ca-trustpoint)# enrollment terminal
ssl-proxy(ca-trustpoint)# revocation-check none
ssl-proxy(ca-trustpoint)# exit
ssl-proxy(config)# crypto pki trustpoint 3tier-sub1
ssl-proxy(ca-trustpoint)# enrollment terminal
ssl-proxy(ca-trustpoint)# revocation-check none
ssl-proxy(ca-trustpoint)# exit
```

   b. Import the root certificate authority certificate:

```
ssl-proxy(config)# crypto pki authenticate 3tier-root

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

-----BEGIN CERTIFICATE-----
MIIC1zCCAoGgAwIBAgIQadUxzU/i97hDmZRYJ1bBcDANBgkqhkiG9w0BAQUFADB1
MQswCQYDVQQGEwJVUzETMBEGA1UECBMKY2FsaWZvcm5pYTERMA8GA1UEBxMIc2Fu
IGpvc2UxDjAMBgNVBAoTBWNpc2NvMQwwCgYDVQQLEwNoc3MxIDAeBgNVBAMTF3Np
bXBzb24tZGV2dGVzdC1yb290LUNBMB4XDTAzMTExMTIxNDgwMloXDTEzMTExMTIx
NTczOVowdTELMAkGA1UEBhMCVVMxEzARBgNVBAgTCmNhbGlmb3JuaWExETAPBgNV
BAcTCHNhbiBqb3NlMQ4wDAYDVQQKEwVjaXNjbzEMMAoGA1UECxMDaHNzMSAwHgYD
VQQDExdzaW1wc29uLWRldnRlc3Qtcm9vdC1DQTBcMA0GCSqGSIb3DQEBAQUAA0sA
MEgCQQCWEibAnUlVqQNUn0Wb94qnHi8FKjmVhibLHGRl6J+V7gHgzmF2MTz5WP5l
VQ2/1NVu0HjUORRdeCm1/raKJ/7ZAgMBAAGjgewwgekwCwYDVR0PBAQDAgHGMA8G
A1UdEwEB/wQFMAMBAf8wHQYDVR0OBBYEFCYGLUBTKNd9EgUonHnoSvbHg0axMIGX
BgNVHR8EgY8wgYwwQ6BBoD+GPWh0dHA6Ly9jaXNjby1sOGo2b2hwbnIvQ2VydEVu
cm9sbC9zaW1wc29uLWRldnRlc3Qtcm9vdC1DQS5jcmwwRaBDoEGGP2ZpbGU6Ly9c
XGNpc2NvLWw4ajZvaHBuclxDZXJ0RW5yb2xsXHNpbXBzb24tZGV2dGVzdC1yb290
LUNBLmNybDAQBgkrBgEEAYI3FQEEAwIBADANBgkqhkiG9w0BAQUFAANBACBqe1wy
YjalelGZqLVu4bDVMFo6ELCV2AMBgi41K3ix+Z/03PJd7ct2BIAF4lktv9pCe6IO
EoBcmZteA+TQcKg=
-----END CERTIFICATE-----

Certificate has the following attributes:
Fingerprint:AC6FC55E CC29E891 0DC3FAAA B4747C10
% Do you accept this certificate? [yes/no]:yes
Trustpoint CA certificate accepted.
% Certificate successfully imported
```

   c. Import subordinate 1 certificate authority certificate:

```
ssl-proxy(config)# crypto pki authenticate 3tier-sub1

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

-----BEGIN CERTIFICATE-----
MIIETzCCA/mgAwIBAgIKGj0cBwAAAAADjANBgkqhkiG9w0BAQUFADB1MQswCQYD
VQQGEwJVUzETMBEGA1UECBMKY2FsaWZvcm5pYTERMA8GA1UEBxMIc2FuIGpvc2Ux
DjAMBgNVBAoTBWNpc2NvMQwwCgYDVQQLEwNoc3MxIDAeBgNVBAMTF3NpbXBzb24t
ZGV2dGVzdC1yb290LUNBMB4XDTAzMTExMzIyMDQyMVoXDTA0MTExMzIyMTQyMVow
dTELMAkGA1UEBhMCVVMxEzARBgNVBAgTCmNhbGlmb3JuaWExETAPBgNVBAcTCHNh
biBqb3NlMQ4wDAYDVQQKEwVjaXNjbzEMMAoGA1UECxMDaHNzMSAwHgYDVQQDExdz
aW1wc29uLWRldnRlc3Qtc3ViMS1jYTBcMA0GCSqGSIb3DQEBAQUAA0sAMEgCQQDc
```

```
vV48nC2uukoSyGJ/GymCIEXZzMSzpbkYS7eWPaZYyiJDhCIKuUsMgFDRNfMQmUSA
rcWmPizFZc9PFumDa03vAgMBAAGjggJpMIICZTAQBgkrBgEEAYI3FQEEAwIBADAd
BgNVHQ4EFgQUWaaNN2U14BaBoU9mY+ncuHpP920wCwYDVR0PBAQDAgHGMA8GA1Ud
EwEB/wQFMAMBAf8wga4GA1UdIwSBpjCBo4AUJgYtQFMo130SBSiceehK9seDRrGh
eaR3MHUxCzAJBgNVBAYTAlVTMRMwEQYDVQQIEwpjYWxpZm9ybmlhMREwDwYDVQQH
EwhzYW4gam9zZTEOMAwGA1UEChMFY2lzY28xDDAKBgNVBAsTA2hzczEgMB4GA1UE
AxMXc2ltcHNvbi1kZXZ0ZXN0LXJvb3QtQ0GCEGnVMc1P4ve4Q5mUWCdWwXAwgZcG
A1UdHwSBjzCBjDBDoEGgP4Y9aHR0cDovL2Npc2NvLWw4ajZvaHBuci9DZXJ0RW5y
b2xsL3NpbXBzb24tZGV2dGVzdC1yb290LUNBLmNybDBFoEOgQYY/ZmlsZTovL1xc
Y2lzY28tbDhqNm9ocG5yXENlcnRFbnJvbGxcc2ltcHNvbi1kZXZ0ZXN0LXJvb3Qt
Q0EuY3JsMIHIBggrBgEFBQcBAQSBuzCBuDBZBggrBgEFBQcwAoZNaHR0cDovL2Np
c2NvLWw4ajZvaHBuci9DZXJ0RW5yb2xsL2Npc2NvLWw4ajZvaHBucl9zaW1wc29u
LWRldnRlc3Qtcm9vdC1DQS5jcnQwWwYIKwYBBQUHMAKGT2ZpbGU6Ly9cXGNpc2Nv
LWw4ajZvaHBuclxDZXJ0RW5yb2xsXGNpc2NvLWw4ajZvaHBucl9zaW1wc29uLWRl
dnRlc3Qtcm9vdC1DQS5jcnQwDQYJKoZIhvcNAQEFBQADQQA6kAV3Jx/BOr2hlSp9
ER36ZkDJNIW93gNt2MkpcA07RmcrHln6q5RJ9WbvTxFnONdgpsag1EcOwn97XErH
Z2ow
-----END CERTIFICATE-----

Certificate has the following attributes:
Fingerprint:50A986F6 B471B82D E11B71FE 436A9BE6
Certificate validated - Signed by existing trustpoint CA certificate.
Trustpoint CA certificate accepted.
% Certificate successfully imported
```

d. Import the subordinate 2 certificate authority certificate:

```
ssl-proxy(config)# crypto pki authenticate tp-proxy1

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

-----BEGIN CERTIFICATE-----
MIIESTCCA/OgAwIBAgIKHyiFxAAAAAAABjANBgkqhkiG9w0BAQUFADB1MQswCQYD
VQQGEwJVUzETMBEGA1UECBMKY2FsaWZvcm5pYTERMA8GA1UEBxMIc2FuIGpvc2Ux
DjAMBgNVBAoTBWNpc2NvMQwwCgYDVQQLEwNoc3MxIDAeBgNVBAMTF3NpbXBzb24t
ZGV2dGVzdC1zdWIxLWNhMB4XDTAzMTExMzIyMjI1MloXDTA0MTExMzIyMTQyMVow
dTELMAkGA1UEBhMCVVMxEzARBgNVBAgTCmNhbGlmb3JuaWExETAPBgNVBAcTCHNh
biBqb3NlMQ4wDAYDVQQKEwVjaXNjbzEMMAoGA1UECxMDaHNzMSAwHgYDVQQDExdz
aW1wc29uLWRldnRlc3Qtc3ViMi1jYTBcMA0GCSqGSIb3DQEBAQUAA0sAMEgCQQC7
ChZc0NYLBHf1sr/3Z4y6w5WoeioIpCOCSydhnbd5wnwuethoyStVt9lr6i61jWKl
d68Z8EoTg71daiV/WR/HAgMBAAGjggJjMIICXzAQBgkrBgEEAYI3FQEEAwIBADAd
BgNVHQ4EFgQU6FmJopqqzpbFMj6TaB2/wjlWlqEwCwYDVR0PBAQDAgHGMA8GA1Ud
EwEB/wQFMAMBAf8wgagGA1UdIwSBoDCBnYAUWaaNN2U14BaBoU9mY+ncuHpP922h
eaR3MHUxCzAJBgNVBAYTAlVTMRMwEQYDVQQIEwpjYWxpZm9ybmlhMREwDwYDVQQH
EwhzYW4gam9zZTEOMAwGA1UEChMFY2lzY28xDDAKBgNVBAsTA2hzczEgMB4GA1UE
AxMXc2ltcHNvbi1kZXZ0ZXN0LXJvb3QtQ0GCCho9HAcAAAAAAA4wgZcGA1UdHwSB
jzCBjDBDoEGgP4Y9aHR0cDovL2Npc2NvLWcyNXVhNm80ZS9DZXJ0RW5yb2xsL3Np
bXBzb24tZGV2dGVzdC1zdWIxLWNhLmNybDBFoEOgQYY/ZmlsZTovL1xcY2lzY28t
ZzI1dWE2bzRlXENlcnRFbnJvbGxcc2ltcHNvbi1kZXZ0ZXN0LXN1YjEtY2EuY3Js
MIHIBggrBgEFBQcBAQSBuzCBuDBZBggrBgEFBQcwAoZNaHR0cDovL2Npc2NvLWcy
NXVhNm80ZS9DZXJ0RW5yb2xsL2Npc2NvLWcyNXVhNm80ZV9zaW1wc29uLWRldnRl
c3Qtc3ViMS1jYS5jcnQwWwYIKwYBBQUHMAKGT2ZpbGU6Ly9cXGNpc2NvLWcyNXVh
Nm80ZVxDZXJ0RW5yb2xsXGNpc2NvLWcyNXVhNm80ZV9zaW1wc29uLWRldnRlc3Qt
c3ViMS1jYS5jcnQwDQYJKoZIhvcNAQEFBQADQQCieB8rvVCqVF2cFw9/v51jGn7L
Q6pUGT3bMRbOrgQKytTz/Yx09156nYZHrvVuLzmzz5CriI2saVx+q1Tarwil
-----END CERTIFICATE-----

Certificate has the following attributes:
Fingerprint:2F2E44AC 609644FA 5B4B6B26 FDBFE569
Certificate validated - Signed by existing trustpoint CA certificate.
Trustpoint CA certificate accepted.
% Certificate successfully imported
```

e. Import the server certificate:

```
ssl-proxy(config)# crypto pki import tp-proxy1 certificate
% The fully-qualified domain name in the certificate will be:ssl-proxy.

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

-----BEGIN CERTIFICATE-----
MIIENTCCA9+gAwIBAgIKLmibDwAAAAAACDANBgkqhkiG9w0BAQUFADB1MQswCQYD
VQQGEwJVUzETMBEGA1UECBMKY2FsaWZvcm5pYTERMA8GA1UEBxMIc2FuIGpvc2Ux
DjAMBgNVBAoTBWNpc2NvMQwwCgYDVQQLEwNoc3MxIDAeBgNVBAMTF3NpbXBzb24t
ZGV2dGVzdC1zdWIyLWNhMB4XDTAzMTExOTIzNDUzNVoXDTA0MTExMzIyMTQyMVow
PTERMA8GA1UEBRMIQjBGRkYxNEQxGTAXBgkqhkiG9w0BCQITCnNzbC1wcm94eS4x
DTALBgNVBAMTBHRlc3QwgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAMXHWiX0
hegDHh/JSFpcfqzinm32yghnWC8u1VfoWnrN+5fjuizlMYkf8hfNX0Hh7kq/7CUw
Af8EBAMCBaAwHQYDVR0OBBYEFCXlzcYHyo1PNbfubnivi8d2VO22MIGoBgNVHSME
gaAwgZ2AFOhZiaKaqs6WxTI+k2gdv8I5VpahoXmkdzB1MQswCQYDVQQGEwJVUzET
MBEGA1UECBMKY2FsaWZvcm5pYTERMA8GA1UEBxMIc2FuIGpvc2UxDjAMBgNVBAoT
BWNpc2NvMQwwCgYDVQQLEwNoc3MxIDAeBgNVBAMTF3NpbXBzb24tZGV2dGVzdC1z
dWIxLWNhggofKIXEAAAAAAAGMIGXBgNVHR8EgY8wgYwwQ6BBoD+GPWh0dHA6Ly9j
aXNjby1vam14Y25jenYvQ2VydEVucm9sbC9zaW1wc29uLWRldnRlc3Qtc3ViMi1j
YS5jcmwwRaBDoEGGP2ZpbGU6Ly9cXGNpc2NvLW9qbXhjbmN6dlxDZXJ0RW5yb2xs
XHNpbXBzb24tZGV2dGVzdC1zdWIyLWNhLmNybDCByAYIKwYBBQUHAQEEgbswgbgw
WQYIKwYBBQUHMAKGTWh0dHA6Ly9jaXNjby1vam14Y25jenYvQ2VydEVucm9sbC9j
aXNjby1vam14Y25jenZfc2ltcHNvbi1kZXZ0ZXN0LXN1YjItY2EuY3J0MFsGCCsG
AQUFBzAChk9maWxlOi8vXFxjaXNjby1vam14Y25jenZcQ2VydEVucm9sbFxjaXNj
by1vam14Y25jenZfc2ltcHNvbi1kZXZ0ZXN0LXN1YjItY2EuY3J0MA0GCSqGSIb3
DQEBBQUAA0EAtbxmUBOxZ/hcrCc3hY7pa6q/LmLonXSL8cjAbV2I7A5QGYaNi5k9
8FlEz1WOxW0J2C3/YsvIf4dYpsQWdKRJbQ==
-----END CERTIFICATE-----

% Router Certificate successfully imported

ssl-proxy(config)#^Z
```

Importing and Exporting Key Pairs and Certificates

You can import and export key pairs and certificates using either the PKCS12 file format or privacy-enhanced mail (PEM) file format.

To import or export key pairs and certificates, see one of these sections:

- Importing and Exporting a PKCS12 File, page 3-20
- Importing and Exporting PEM Files, page 3-21

Note A test PKCS12 file (test/testssl.p12) is embedded in the SSL software on the module. You can install the file into NVRAM for testing purposes and for proof of concept. After the PKCS12 file is installed, you can import it to a trustpoint and then assign it to a proxy service configured for testing. For information on installing the test PKCS12 file, see the "Importing the Embedded Test Certificate" section on page C-4.

Note If you have configured the **revocation-check crl** command, and if the certificate revocation list (CRL) fails to download because the CRL server is unreachable or the CRL download path does not exist, the certificate might fail to import. You should make sure all trustpoints that are linked to the import process are able to download the CRL. If the CRL path does not exist, or if the CRL server is unreachable, then you should enter the **revocation-check none** command for all trustpoints that are linked to the import process. Enter the **show crypto pki certificates** command to display information for all certificates, and obtain a list of associated trustpoints from the display of the certificate authority certificate. Enter the **revocation-check none** command for all these trustpoints.

For example, in a three-tier certificate authority hierarchy (root CA, subordinate CA1, and subordinate CA2), when you import the subordinate CA1 certificate, enter the **revocation-check none** command for all the trustpoints associated with root CA. Similarly, when you import the subordinate CA2 certificate, enter the **revocation-check none** command for all the trustpoints associated with root CA and subordinate CA1.

After you successfully import the certificate, you can restore the original CRL options on the trustpoints.

Importing and Exporting a PKCS12 File

You can use an external PKI system to generate a PKCS12 file and then import this file to the module.

Note When creating a PKCS12 file, include the entire certificate chain, from server certificate to root certificate, and public and private keys. You can also generate a PKCS12 file from the module and export it.

Note Imported key pairs cannot be exported.

Note If you are using SSH, we recommend using SCP (secure file transfer) when importing or exporting a PKCS12 file. SCP authenticates the host and encrypts the transfer session.

To import or export a PKCS12 file, perform this task:

Command	Purpose
`ssl-proxy(config)# crypto pki {import | export} trustpoint_label pkcs12 {scp:| ftp:| nvram:| rcp:| tftp:} [pkcs12_filename[1]] pass_phrase[2]`	Imports or exports a PKCS12 file. **Note** You do not need to configure a trustpoint before importing the PKCS12 file. Importing keys and certificates from a PKCS12 file creates the trustpoint automatically, if it does not already exist.

1. If you do not specify the *pkcs12_filename* value, you will be prompted to accept the default filename (the default filename is the *trustpoint_label* value) or enter the filename. For **ftp:** or **tftp:**, include the full path in the *pkcs12_filename* value.
2. You will receive an error if you enter the pass phrase incorrectly.

This example shows how to import a PKCS12 file using SCP:

```
ssl-proxy(config)# crypto pki import TP2 pkcs12 scp: sky is blue
Address or name of remote host []? 10.1.1.1
Source username [ssl-proxy]? admin-1
Source filename [TP2]? /users/admin-1/pkcs12/TP2.p12

Password:password
Sending file modes:C0644 4379 TP2.p12
!
ssl-proxy(config)#
*Aug 22 12:30:00.531:%CRYPTO-6-PKCS12IMPORT_SUCCESS:PKCS #12 Successfully Imported.
ssl-proxy(config)#
```

This example shows how to export a PKCS12 file using SCP:

```
ssl-proxy(config)# crypto pki export TP1 pkcs12 scp: sky is blue
Address or name of remote host []? 10.1.1.1
Destination username [ssl-proxy]? admin-1
Destination filename [TP1]? TP1.p12

Password:

Writing TP1.p12 Writing pkcs12 file to scp://admin-1@10.1.1.1/TP1.p12

Password:
!
CRYPTO_PKI:Exported PKCS12 file successfully.
ssl-proxy(config)#
```

This example shows how to import a PKCS12 file using FTP:

```
ssl-proxy(config)# crypto pki import TP2 pkcs12 ftp: sky is blue
Address or name of remote host []? 10.1.1.1
Source filename [TP2]? /admin-1/pkcs12/PK-1024
Loading /admin-1/pkcs12/PK-1024 !
[OK - 4339/4096 bytes]
ssl-proxy(config)#
```

This example shows how to export a PKCS12 file using FTP:

```
ssl-proxy(config)# crypto pki export TP1 pkcs12 ftp: sky is blue
Address or name of remote host []? 10.1.1.1
Destination filename [TP1]? /admin-1/pkcs12/PK-1024
Writing pkcs12 file to ftp://10.1.1.1//admin-1/pkcs12/PK-1024

Writing /admin-1/pkcs12/PK-1024 !!
CRYPTO_PKI:Exported PKCS12 file successfully.
ssl-proxy(config)#
```

After you import the PKCS12 file, see the "Verifying Certificates and Trustpoints" section on page 3-28 to verify the certificate and trustpoint information.

Importing and Exporting PEM Files

Note The **crypto pki import pem** command imports only the private key (.prv), the server certificate (.crt), and the issuer certificate authority certificate (.ca). If you have more than one level of certificate authority in the certificate chain, you need to import the root and subordinate certificate authority certificates before this command is issued for authentication. Use cut-and-paste or TFTP to import the root and subordinate certificate authority certificates.

Note Imported key pairs cannot be exported.

Note If you are using SSH, we recommend using SCP (secure file transfer) when importing or exporting PEM files. SCP authenticates the host and encrypts the transfer session.

To import or export PEM files, perform one of these tasks:

Command	Purpose
`ssl-proxy(config)# crypto pki import` *trustpoint_label* **pem** [**exportable**] {**terminal** | **url** {**scp:**| **ftp:**| **nvram:**| **rcp:**| **tftp:**} | **usage-keys**} *pass_phrase*[1,2]	Imports PEM files. **Note** You do not need to configure a trustpoint before importing the PEM files. Importing keys and certificates from PEM files creates the trustpoint automatically, if it does not already exist.
`ssl-proxy(config)# crypto pki export` *trustpoint_label* **pem** {**terminal** | **url** {**scp:**| **ftp:**| **nvram:**| **rcp:**| **tftp:**} [**des** | **3des**] *pass_phrase*[1,2]	Exports PEM files. **Note** Only the key, the server certificate, and the issuer certificate authority of the server certificate are exported. All higher level certificate authorities need to be exported using cut-and-paste of TFTP.

1. You will receive an error if you enter the pass phrase incorrectly.
2. A pass phrase protects a PEM file that contains a private key. The PEM file is encrypted by DES or 3DES. The encryption key is derived from the pass phrase. A PEM file containing a certificate is not encrypted and is not protected by a pass phrase.

This example shows how to import PEM files using TFTP:

Note The TP5.ca, TP5.prv, and TP5.crt files should be present on the server.

```
ssl-proxy(config)# crypto pki import TP5 pem url tftp://10.1.1.1/TP5 password
% Importing CA certificate...
Address or name of remote host [10.1.1.1]?
Destination filename [TP5.ca]?
Reading file from tftp://10.1.1.1/TP5.ca
Loading TP5.ca from 10.1.1.1 (via Ethernet0/0.168): !
[OK - 1976 bytes]

% Importing private key PEM file...
Address or name of remote host [10.1.1.1]?
Destination filename [TP5.prv]?
Reading file from tftp://10.1.1.1/TP5.prv
Loading TP5.prv from 10.1.1.1 (via Ethernet0/0.168): !
[OK - 963 bytes]
```

```
% Importing certificate PEM file...
Address or name of remote host [10.1.1.1]?
Destination filename [TP5.crt]?
Reading file from tftp://10.1.1.1/TP5.crt
Loading TP5.crt from 10.1.1.1 (via Ethernet0/0.168): !
[OK - 1692 bytes]
% PEM files import succeeded.
ssl-proxy(config)#end
ssl-proxy#
*Apr 11 15:11:29.901: %SYS-5-CONFIG_I: Configured from console by console
```

This example shows how to export PEM files using TFTP:

```
ssl-proxy(config)# crypto pki export TP5 pem url tftp://10.1.1.1/tp99 3des password
% Exporting CA certificate...
Address or name of remote host [10.1.1.1]?
Destination filename [tp99.ca]?
% File 'tp99.ca' already exists.
% Do you really want to overwrite it? [yes/no]: yes
!Writing file to tftp://10.1.1.1/tp99.ca!
% Key name: key1
    Usage: General Purpose Key
% Exporting private key...
Address or name of remote host [10.1.1.1]?
Destination filename [tp99.prv]?
% File 'tp99.prv' already exists.
% Do you really want to overwrite it? [yes/no]: yes
!Writing file to tftp://10.1.1.1/tp99.prv!
% Exporting router certificate...
Address or name of remote host [10.1.1.1]?
Destination filename [tp99.crt]?
% File 'tp99.crt' already exists.
% Do you really want to overwrite it? [yes/no]: yes
!Writing file to tftp://10.1.1.1/tp99.crt!
ssl-proxy(config)#
```

After you import the PEM files, see the "Verifying Certificates and Trustpoints" section on page 3-28 to verify the certificate and trustpoint information.

Example of Importing PEM Files for Three Levels of Certificate Authority

In this section, the root certificate authority certificate (Tier 1) and intermediate certificate authority certificate (Tier 2) are obtained using the cut-and-paste option of offline enrollment. The intermediate certificate authority certificate (Tier 3), private keys, and router certificate are obtained by importing PEM files.

1. Use cut-and-paste to obtain the root certificate authority-tier 1 certificate:

```
ssl-proxy(config)# crypto pki trustpoint 3tier-root
ssl-proxy(ca-trustpoint)# enrollment terminal
ssl-proxy(ca-trustpoint)# revocation-check none
ssl-proxy(ca-trustpoint)# exit
ssl-proxy(config)# crypto pki authenticate 3tier-root

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

```
-----BEGIN CERTIFICATE-----
MIIC1zCCAoGgAwIBAgIQadUxzU/i97hDmZRYJ1bBcDANBgkqhkiG9w0BAQUFADB1
MQswCQYDVQQGEwJVUzETMBEGA1UECBMKY2FsaWZvcm5pYTERMA8GA1UEBxMIc2Fu
IGpvc2UxDjAMBgNVBAoTBWNpc2NvMQwwCgYDVQQLEwNoc3MxIDAeBgNVBAMTF3Np
bXBzb24tZGV2dGVzdC1yb290LUNBMB4XDTAzMTExMTIxNDgwMloXDTEzMTExMTIx
NTczOVowdTELMAkGA1UEBhMCVVMxEzARBgNVBAgTCmNhbGlmb3JuaWExETAPBgNV
BAcTCHNhbiBqb3NlMQ4wDAYDVQQKEwVjaXNjbzEMMAoGA1UECxMDaHNzMSAwHgYD
VQQDExdzaW1wc29uLWRldnRlc3Qtcm9vdC1DQTBcMA0GCSqGSIb3DQEBAQUAA0sA
MEgCQQCWEibAnUlVqQNUn0Wb94qnHi8FKjmVhibLHGRl6J+V7gHgzmF2MTz5WP5l
VQ2/1NVu0HjUORRdeCm1/raKJ/7ZAgMBAAGjgewwgekwCwYDVR0PBAQDAgHGMA8G
A1UdEwEB/wQFMAMBAf8wHQYDVR0OBBYEFCYGLUBTKNd9EgUonHnoSvbHg0axMIGX
BgNVHR8EgY8wgYwwQ6BBoD+GPWh0dHA6Ly9jaXNjby1sOGo2b2hwbnIvQ2VydEVu
cm9sbC9zaW1wc29uLWRldnRlc3Qtcm9vdC1DQS5jcmwwRaBDoEGGP2ZpbGU6Ly9c
XGNpc2NvLWw4ajZvaHBuclxDZXJ0RW5yb2xsXHNpbXBzb24tZGV2dGVzdC1yb290
LUNBLmNybDAQBgkrBgEEAYI3FQEEAwIBADANBgkqhkiG9w0BAQUFAANBACBqe1wy
YjalelGZqLVu4bDVMFo6ELCV2AMBgi41K3ix+Z/03PJd7ct2BIAF4lktv9pCe6IO
EoBcmZteA+TQcKg=
-----END CERTIFICATE-----

Certificate has the following attributes:
Fingerprint:AC6FC55E CC29E891 0DC3FAAA B4747C10
% Do you accept this certificate? [yes/no]:yes
Trustpoint CA certificate accepted.
% Certificate successfully imported
```

2. Use cut-and-paste to obtain the subordinate certificate authority 1 certificate:

```
ssl-proxy(config)# crypto pki trustpoint 3tier-subca1
ssl-proxy(ca-trustpoint)# enroll terminal
ssl-proxy(ca-trustpoint)# revocation-check none
ssl-proxy(ca-trustpoint)# exit
ssl-proxy(config)# crypto pki authenticate 3tier-subca1

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

-----BEGIN CERTIFICATE-----
MIIETzCCA/mgAwIBAgIKGj0cBwAAAAADjANBgkqhkiG9w0BAQUFADB1MQswCQYD
VQQGEwJVUzETMBEGA1UECBMKY2FsaWZvcm5pYTERMA8GA1UEBxMIc2FuIGpvc2Ux
DjAMBgNVBAoTBWNpc2NvMQwwCgYDVQQLEwNoc3MxIDAeBgNVBAMTF3NpbXBzb24t
ZGV2dGVzdC1yb290LUNBMB4XDTAzMTExMzIyMDQyMVoXDTA0MTExMzIyMTQyMVow
dTELMAkGA1UEBhMCVVMxEzARBgNVBAgTCmNhbGlmb3JuaWExETAPBgNVBAcTCHNh
biBqb3NlMQ4wDAYDVQQKEwVjaXNjbzEMMAoGA1UECxMDaHNzMSAwHgYDVQQDExdz
aW1wc29uLWRldnRlc3Qtc3ViMS1jYTBcMA0GCSqGSIb3DQEBAQUAA0sAMEgCQQDc
vV48nC2uukoSyGJ/GymCIEXZzMSzpbkYS7eWPaZYyiJDhCIKuUsMgFDRNfMQmUSA
rcWmPizFZc9PFumDa03vAgMBAAGjggJpMIICZTAQBgkrBgEEAYI3FQEEAwIBADAd
BgNVHQ4EFgQUWaaNN2U14BaBoU9mY+ncuHpP920wCwYDVR0PBAQDAgHGMA8GA1Ud
EwEB/wQFMAMBAf8wga4GA1UdIwSBpjCBo4AUJgYtQFMo130SBSiceehK9seDRrGh
eaR3MHUxCzAJBgNVBAYTAlVTMRMwEQYDVQQIEwpjYWxpZm9ybmlhMREwDwYDVQQH
EwhzYW4gam9zZTEOMAwGA1UEChMFY2lzY28xDDAKBgNVBAsTA2hzczEgMB4GA1UE
AxMXc2ltcHNvbi1kZXZ0ZXN0LXJvb3QtQ0GCEGnVMc1P4ve4Q5mUWCdWwXAwgZcG
A1UdHwSBjzCBjDBDoEGgP4Y9aHR0cDovL2Npc2NvLWw4ajZvaHBuci9DZXJ0RW5y
b2xsL3NpbXBzb24tZGV2dGVzdC1yb290LUNBLmNybDBFoEOgQYY/ZmlsZTovL1xc
Y2lzY28tbDhqNm9ocG5yXENlcnRFbnJvbGxcc2ltcHNvbi1kZXZ0ZXN0LXJvb3Qt
Q0EuY3JsMIHIBggrBgEFBQcBAQSBuzCBuDBZBggrBgEFBQcwAoZNaHR0cDovL2Np
c2NvLWw4ajZvaHBuci9DZXJ0RW5yb2xsL2Npc2NvLWw4ajZvaHBucl9zaW1wc29u
LWRldnRlc3Qtcm9vdC1DQS5jcnQwWwYIKwYBBQUHMAKGT2ZpbGU6Ly9cXGNpc2Nv
LWw4ajZvaHBuclxDZXJ0RW5yb2xsXGNpc2NvLWw4ajZvaHBucl9zaW1wc29uLWRl
dnRlc3Qtcm9vdC1DQS5jcnQwDQYJKoZIhvcNAQEFBQADQQA6kAV3Jx/BOr2hlSp9
ER36ZkDJNIW93gNt2MkpcA07RmcrHln6q5RJ9WbvTxFnONdgpsag1EcOwn97XErH
Z2ow
-----END CERTIFICATE-----
```

```
Certificate has the following attributes:
Fingerprint:50A986F6 B471B82D E11B71FE 436A9BE6
Certificate validated - Signed by existing trustpoint CA certificate.
Trustpoint CA certificate accepted.
% Certificate successfully imported
```

3. Import the subordinate certificate authority 2 certificate, the RSA key pair, and router certificate. The router certificate should be signed by subordinate certificate authority 2.

```
ssl-proxy(config)# crypto pki import tp-proxy1 pem terminal cisco
% Enter PEM-formatted CA certificate.
% End with a blank line or "quit" on a line by itself.
-----BEGIN CERTIFICATE-----
MIIESTCCA/OgAwIBAgIKHyiFxAAAAAAABjANBgkqhkiG9w0BAQUFADB1MQswCQYD
VQQGEwJVUzETMBEGA1UECBMKY2FsaWZvcm5pYTERMA8GA1UEBxMIc2FuIGpvc2Ux
DjAMBgNVBAoTBWNpc2NvMQwwCgYDVQQLEwNoc3MxIDAeBgNVBAMTF3NpbXBzb24t
ZGV2dGVzdC1zdWIxLWNhMB4XDTAzMTExMzIyMjI1MloXDTA0MTExMzIyMTQyMVow
dTELMAkGA1UEBhMCVVMxEzARBgNVBAgTCmNhbGlmb3JuaWExETAPBgNVBAcTCHNh
biBqb3NlMQ4wDAYDVQQKEwVjaXNjbzEMMAoGA1UECxMDaHNzMSAwHgYDVQQDExdz
aW1wc29uLWRldnRlc3Qtc3ViMi1jYTBcMA0GCSqGSIb3DQEBAQUAA0sAMEgCQQC7
ChZc0NYLBHf1sr/3Z4y6w5WoeioIpCOCSydhnbd5wnwuethoyStVt9lr6i61jWKl
d68Z8EoTg71daiV/WR/HAgMBAAGjggJjMIICXzAQBgkrBgEEAYI3FQEEAwIBADAd
BgNVHQ4EFgQU6FmJopqqzpbFMj6TaB2/wjlWlqEwCwYDVR0PBAQDAgHGMA8GA1Ud
EwEB/wQFMAMBAf8wgagGA1UdIwSBoDCBnYAUWaaNN2U14BaBoU9mY+ncuHpP922h
eaR3MHUxCzAJBgNVBAYTAlVTMRMwEQYDVQQIEwpjYWxpZm9ybmlhMREwDwYDVQQH
EwhzYW4gam9zZTEOMAwGA1UEChMFY2lzY28xDDAKBgNVBAsTA2hzczEgMB4GA1UE
AxMXc2ltcHNvbi1kZXZ0ZXN0LXJvb3QtQ0GCCho9HAcAAAAAAA4wgZcGA1UdHwSB
jzCBjDBDoEGgP4Y9aHR0cDovL2Npc2NvLWcyNXVhNm80ZS9DZXJ0RW5yb2xsL3Np
bXBzb24tZGV2dGVzdC1zdWIxLWNhLmNybDBFoEOgQYY/ZmlsZTovL1xcY2lzY28t
ZzI1dWE2bzRlXENlcnRFbnJvbGxcc2ltcHNvbi1kZXZ0ZXN0LXN1YjEtY2EuY3Js
MIHIBggrBgEFBQcBAQSBuzCBuDBZBggrBgEFBQcwAoZNaHR0cDovL2Npc2NvLWcy
NXVhNm80ZS9DZXJ0RW5yb2xsL2Npc2NvLWcyNXVhNm80ZV9zaW1wc29uLWRldnRl
c3Qtc3ViMS1jYS5jcnQwWwYIKwYBBQUHMAKGT2ZpbGU6Ly9cXGNpc2NvLWcyNXVh
Nm80ZVxDZXJ0RW5yb2xsXGNpc2NvLWcyNXVhNm80ZV9zaW1wc29uLWRldnRlc3Qt
c3ViMS1jYS5jcnQwDQYJKoZIhvcNAQEFBQADQQCieB8rvVCqVF2cFw9/v51jGn7L
Q6pUGT3bMRbOrgQKytTz/Yx09156nYZHrvVuLzmzz5CriI2saVx+q1Tarwil
-----END CERTIFICATE-----

% Enter PEM-formatted encrypted private key.
% End with "quit" on a line by itself.
-----BEGIN RSA PRIVATE KEY-----
Proc-Type:4,ENCRYPTED
DEK-Info:DES-EDE3-CBC,F0D3269840071CF8

gQb9JMplIE5AEdhumLuBFWT53k+L/EGLhFfQn/roPlEOiIGEB6y3DeYNN/xZSiy3
JOHN0kh8Wjw3pshrdNVcoQj2X7BPI+YOipok40WOk5J/+dnRLwMjv+rl0tr+LcCk
nBdR8zIOkOJObULLUOXFBM7oB3Dsk4Y3FBv8EAR3AdQiZjevau4FIyQn+JfVZy+J
wctmvZnX0c0fevPsgID4dCPkeY6+I0DkxMyRiuyn+wIrJw1xVA2VIOrRJojBNlRu
6/APef8JwpfnNcgpcLYt/4Q+3Yjl9EfRLjgiL6eSRki/6K5lrV3eKbwOTyjvXq5h
G0Q6dtNEoIvOg1Vad0CXeL+TxJ4ySq4E63OxIHkclDBsusGoUGLoZ+OtaxApAZ+5
WbKqR+ND1LlPmS8/ZL9LMPhUh9eOqZJjJTe6NbxY7jeNHjAmpP7/WpB2f2kV/LZg
n2AV4GALBZtqXtreGiayZzXpEA5J00lbzRZWf9JHA1diz/unW00/GH9LvCqA9O15
YJGCrRMI9US7MWm8kIkiJqNgLtbPad5cOaieQe+Kncgcm18Hc7pfhDwXGG4RS40x
TSV/kIR4Gi7h8Lu71wZKTaWYHBPTUyTIpNsFUEdvItHXOSBw2LWNWzdYgpGoMT/t
ryuu0lAC9YdBalAxY0DaqqpuXKzxfiw5QDbqZWVq3qAxXfLAtTgu/gFCuFQvbBGl
87H1C+nOQUq2nkpMpHZLsl3V0w/2yqg+q6rUydANFF+a5vRaLgX/PGms92ZkZUdP
Z5qeKJmoURSlMYxDuhQDl93RYxXJxOYIYrCrI/QaBpIH6QvUH60wWA==
-----END RSA PRIVATE KEY-----
quit
% Enter PEM-formatted certificate.
% End with a blank line or "quit" on a line by itself.
```

```
-----BEGIN CERTIFICATE-----
MIIEXTCCBAegAwIBAgIKTJOcWgAAAAAACTANBgkqhkiG9w0BAQUFADB1MQswCQYD
VQQGEwJVUzETMBEGA1UECBMKY2FsaWZvcm5pYTERMA8GA1UEBxMIc2FuIGpvc2Ux
DjAMBgNVBAoTBWNpc2NvMQwwCgYDVQQLEwNoc3MxIDAeBgNVBAMTF3NpbXBzb24t
ZGV2dGVzdC1zdWIyLWNhMB4XDTAzMTEyNTIwMjExMFoXDTA0MTExMzIyMTQyMVow
PjERMA8GA1UEBRMIQjBGRkYyMkUxKTAnBgkqhkiG9w0BCQITGnNpbXBzb24tNjUw
OS1zdGUuY2lzY28uY29tMIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCkhRKm
38hSF7l0WXlYm8ixs2Hz/yjNw7tchtRPIp0qCTJKW00gzZpp8dqaNi3s2GVVWb+t
Cgsol0MZLIkyoj/9vT9MC7Zo3LOxYy9kD+6M9peUWMT4JLSD4Exzxsd87JpP1bo0
o8WhYjvMor/bL30sW8ly2RH2vppEMn9eLEN0vwIDAQABo4ICajCCAmYwCwYDVR0P
BAQDAgWgMB0GA1UdDgQWBBSx6uQ2sARlcjzhBSiMu7xeu1n6AjCBqAYDVR0jBIGg
MIGdgBToWYmimqrOlsUyPpNoHb/COVaWoaF5pHcwdTELMAkGA1UEBhMCVVMxEzAR
BgNVBAgTCmNhbGlmb3JuaWExETAPBgNVBAcTCHNhbiBqb3NlMQ4wDAYDVQQKEwVj
aXNjbzEMMAoGA1UECxMDaHNzMSAwHgYDVQQDExdzaW1wc29uLWRldnRlc3Qtc3Vi
MS1jYYIKHyiFxAAAAAAABjAoBgNVHREBAf8EHjAcghpzaW1wc29uLTY1MDktc3Rl
LmNpc2NvLmNvbTCBlwYDVR0fBIGPMIGMMEOgQaA/hj1odHRwOi8vY2lzY28tb2pt
eGNuY3p2L0NlcnRFbnJvbGwvc2ltcHNvbi1kZXZ0ZXN0LXN1YjItY2EuY3JsMEWg
Q6BBhj9maWxlOi8vXFxjaXNjby1vam14Y25jenZcQ2VydEVucm9sbFxzaW1wc29u
LWRldnRlc3Qtc3ViMi1jYS5jcmwwgcgGCCsGAQUFBwEBBIG7MIG4MFkGCCsGAQUF
BzAChk1odHRwOi8vY2lzY28tb2pteGNuY3p2L0NlcnRFbnJvbGwvY2lzY28tb2pt
eGNuY3p2X3NpbXBzb24tZGV2dGVzdC1zdWIyLWNhLmNydDBbBggrBgEFBQcwAoZP
ZmlsZTovL1xcY2lzY28tb2pteGNuY3p2XENlcnRFbnJvbGxcY2lzY28tb2pteGNu
Y3p2X3NpbXBzb24tZGV2dGVzdC1zdWIyLWNhLmNydDANBgkqhkiG9w0BAQUFAANB
ABFh7XeLwvfBtjAR+e5OaUH5KTGJDbeJppOmMFXnFakpgWop9Qg4cHRCQq7V0pAW
iA6VtJOmpYgEIVNTAzAAHR4=
-----END CERTIFICATE-----

% PEM files import succeeded.
ssl-proxy(config)# ^Z
ssl-proxy#
*Dec  4 18:11:49.850:%SYS-5-CONFIG_I:Configured from console by console
ssl-proxy#
```

4. Display the certificate information (optional):

```
ssl-proxy# show crypto pki certificates tp-proxy1
Certificate
  Status:Available
  Certificate Serial Number:04A0147B00000000010E
  Certificate Usage:General Purpose
  Issuer:
    CN = sub3ca
     C = US
Subject:
    Name:ssl-proxy.
    Serial Number:B0FFF0C2
    OID.1.2.840.113549.1.9.2 = ssl-proxy.
     OID.2.5.4.5 = B0FFF0C2
  CRL Distribution Point:
    http://sample.cisco.com/sub3ca.crl
  Validity Date:
    start date:18:04:09 UTC Jan 23 2003
    end   date:21:05:17 UTC Dec 12 2003
    renew date:00:00:00 UTC Apr 1 2003
  Associated Trustpoints:tp-proxy1
```

```
CA Certificate
  Status:Available
  Certificate Serial Number:6D1E6B0F000000000007
  Certificate Usage:Signature
  Issuer:
    CN = subtest
     C = US
Subject:
    CN = sub3ca
     C = US
CRL Distribution Point:
    http://sample.cisco.com/subtest.crl
  Validity Date:
    start date:22:22:52 UTC Mar 28 2003
    end   date:21:05:17 UTC Dec 12 2003
  Associated Trustpoints:tp-proxy1

ssl-proxy# show crypto pki certificates 3tier-subca1
CA Certificate
  Status:Available
  Certificate Serial Number:29A47DEF0000000004E9
  Certificate Usage:Signature
  Issuer:
    CN = 6ebf9b3e-9a6d-4400-893c-dd85dcfe911b
     C = US
Subject:
    CN = subtest
     C = US
CRL Distribution Point:
    http://sample.cisco.com/6ebf9b3e-9a6d-4400-893c-dd85dcfe911b.crl
  Validity Date:
    start date:20:55:17 UTC Dec 12 2002
    end   date:21:05:17 UTC Dec 12 2003
  Associated Trustpoints:3tier-sub1

ssl-proxy# show crypto pki certificates 3tier-root
CA Certificate
  Status:Available
  Certificate Serial Number:7FD5B209B5C2448C47F77F140625D265
  Certificate Usage:Signature
  Issuer:
    CN = 6ebf9b3e-9a6d-4400-893c-dd85dcfe911b
     C = US
Subject:
    CN = 6ebf9b3e-9a6d-4400-893c-dd85dcfe911b
     C = US
CRL Distribution Point:
    http://sample.cisco.com/6ebf9b3e-9a6d-4400-893c-dd85dcfe911b.crl
  Validity Date:
    start date:00:05:32 UTC Jun 13 2002
    end   date:00:11:58 UTC Jun 13 2004
  Associated Trustpoints:3tier-root
```

Verifying Certificates and Trustpoints

To verify information about your certificates and trustpoints, perform this task:

	Command	Purpose
Step 1	`ssl-proxy(ca-trustpoint)# show crypto pki certificates` [*trustpoint_label*]	Displays information about the certificates associated with the specified trustpoint, or all of your certificates, the certificates of the certificate authority, and registration authority certificates.
Step 2	`ssl-proxy(ca-trustpoint)# show crypto pki trustpoints` [*trustpoint_label*]	Displays information about all trustpoints or the specified trustpoint.

Sharing Keys and Certificates

The SSL Services Module supports the sharing of the same key pair by multiple certificates. However, this is not a good practice because if one key pair is compromised, all the certificates must be revoked and replaced.

Because proxy services are added and removed at different times, the certificates also expire at different times. Some certificate authorities require you to refresh the key pair at the time of renewal. If certificates share one key pair, you need to renew the certificates at the same time. In general, it is easier to manage certificates if each certificate has its own key pair.

The SSL Services Module does not impose any restrictions on sharing certificates among multiple proxy services and multiple SSL Services Modules. The same trustpoint can be assigned to multiple proxy services.

From a business point of view, the certificate authority may impose restrictions (for example, on the number of servers in a server farm that can use the same certificate). There may be contractual or licensing agreements regarding certificate sharing. Consult with the certificate authority or the legal staff regarding business contractual aspects.

In practice, some web browsers compare the subject name of the server certificate with the hostname or the IP address that appears on the URL. In case the subject name does not match the hostname or IP address, a dialog box appears, prompting the user to verify and accept the certificate. To avoid this step, limit the sharing of certificates based on the hostname or IP address.

Saving Your Configuration

Caution RSA key pairs are saved only to NVRAM. RSA keys are *not* saved with your configuration when you specify any other file system with the **copy system:running-config** *file_system***:** command.

Always remember to save your work when you make configuration changes.

To save your configuration to NVRAM, perform this task:

Command	Purpose
`ssl-proxy# copy [/erase] system:running-config nvram:startup-config`	Saves the configuration, key pairs, and certificate to NVRAM. The key pairs are stored in the private configuration file, and each certificate is stored as a binary file in NVRAM. On bootup, the module will not need to query the certificate authority to obtain the certificates or to auto-enroll. **Note** For security reasons, we recommend that you enter the **/erase** option to erase the public and the private configuration files before updating the NVRAM. If you do not enter the **/erase** option, the key pairs from the old private configuration file may remain in the NVRAM. **Caution** When you enter the **/erase** option, both the current and the backup buffers in NVRAM are erased before the running configuration is saved into NVRAM. If a power failure or reboot occurs after the buffers are erased, but before the running configuration is saved, both configurations might be lost.

Note If you have a large number of files in NVRAM, this task may take up to 2 minutes to finish.

The automatic backup of the configuration to NVRAM feature automatically backs up the last saved configuration. If the current write process fails, the configuration is restored to the previous configuration automatically.

Oversized Configuration

If you save an oversized configuration with more than 256 proxy services and 356 certificates, you might encounter a situation where you could corrupt the contents in the NVRAM.

We recommend that you always copy to running-config before saving to NVRAM. When you save the running-config file to a remote server, each certificate is saved as a hex dump in the file. If you copy the running-config file back to running-config and then save it to NVRAM, the certificates are saved again, but as binary files. However, if you copy the running-config file directly from the remote server to startup-config, the certificates saved as hex dumps are also saved, resulting in two copies of the same certificate: one in hex dump and one as a binary file. This action is unnecessary, and if the remote file is very large, it may overwrite part of the contents in the NVRAM, which could corrupt the contents.

Verifying the Saved Configuration

To verify the saved configuration, perform this task:

	Command	Purpose
Step 1	`ssl-proxy# show startup-config`	Displays the startup configuration.
Step 2	`ssl-proxy# directory nvram:`	Displays the names and sizes of the files in NVRAM.

Note With the maximum number or proxy services (256) and certificates (356) configured, the output takes up to seven minutes to display.

Erasing the Saved Configuration

To erase a saved configuration, perform this task:

Command	Purpose
`ssl-proxy# erase nvram:`	Erases the startup configuration and the key pairs.
`ssl-proxy# erase /all nvram:`	Erases the startup configuration, the key pairs, the certificates, and all other files from the NVRAM.

Note If you have a large number of files in NVRAM, this task may take up to 2 minutes to finish.

Caution If you erase the saved configuration, the automatic backup configuration in NVRAM is also erased.

Backing Up Keys and Certificates

If an event occurs that interrupts the process of saving the keys and certificates to NVRAM (for example, a power failure), you could lose the keys and certificates that are being saved. You can obtain public keys and certificates from the certificate authority. However, you cannot recover private keys.

If a secure server is available, back up key pairs and the associated certificate chain by exporting each trustpoint to a PKCS12 file. You can then import the PKCS12 files to recover the keys and certificates.

See Importing and Exporting a PKCS12 File, page 3-20.

Security Guidelines

When backing up keys and certificates, observe the following guidelines:

- For each PKCS12, you must select a pass phrase that cannot be easily guessed and keep the pass phrase well protected. Do not store the PKCS12 file in clear form.
- The backup server must be secure. Allow only authorized personnel to access the backup server.
- When importing or exporting the PKCS12 file (in which you are required to enter a pass phrase), connect directly to the module console or use an SSH session.
- Use SCP for file transfer.

Monitoring and Maintaining Keys and Certificates

The following tasks in this section are optional:

- Deleting RSA Keys from the Module, page 3-31
- Viewing Keys and Certificates, page 3-32
- Deleting Certificates from the Configuration, page 3-32

Deleting RSA Keys from the Module

Caution Deleting the SSH key will disable SSH on the module. If you delete the SSH key, generate a new key. See the "Configuring SSH" section on page 2-4.

Under certain circumstances you might want to delete the RSA keys from a module. For example, if you believe the RSA keys were compromised in some way and should no longer be used, you should delete the keys.

Note When a private key is deleted while applied to an active SSL proxy service, the service will automatically enter a graceful rollover state in which it will continue to accept connections. The deleted key will stay in memory until a new key has been uploaded.

To delete all RSA keys from the module, perform this task in global configuration mode:

Command	Purpose
`ssl-proxy(config)# crypto key zeroize rsa [key-label]`	Deletes all RSA key pairs, or the specified key pair. **Caution** If a key is deleted, all certificates that are associated with the key are deleted.

After you delete the RSA keys from a module, complete these two additional tasks:

- Ask the certificate authority administrator to revoke the certificates for your module at the certificate authority; you must supply the challenge password that you created for that module with the **crypto pki enroll** command when you originally obtained the certificates.

- Manually remove the trustpoint from the configuration, as described in the "Deleting Certificates from the Configuration" section on page 3-32.

Viewing Keys and Certificates

To view keys and certificates, enter these commands in EXEC mode:

Command	Purpose
`ssl-proxy# show crypto key mypubkey rsa`	Displays RSA public keys for the module.
`ssl-proxy# show crypto pki certificates` [*trustpoint_label*]	Displays information about the certificate, the certificate authority certificate, and any registration authority certificates.
`ssl-proxy# show running-config` [**brief**]	Displays the public keys and the certificate chains. If the *brief* option is specified, the hex dump of each certificate is not displayed.
`ssl-proxy# show ssl-proxy service` *proxy-name*	Displays the key pair and the serial number of the certificate chain used for a specified proxy service. **Note** The *proxy-name* value is case-sensitive.

Deleting Certificates from the Configuration

The module saves its own certificates and the certificate of the certificate authority (trustpoint). You can delete certificates that are saved on the module.

Note When a certificate or trustpoint is deleted while applied to an active SSL proxy service, the service will automatically enter a graceful rollover state in which it will continue to accept connections. The deleted certificate or trustpoint will stay in memory until a new certificate or trustpoint has been uploaded.

To delete the certificate from the module configuration, removing the trustpoint, perform this task in global configuration mode:

Command	Purpose
`ssl-proxy(config)# no crypto pki trustpoint` *trustpoint-label*	Deletes the certificate and trustpoint.

Assigning a Certificate to a Proxy Service

When you enter the **certificate rsa general-purpose trustpoint** *trustpoint_label* subcommand (under the **ssl-proxy service** *proxy_service* command), you assign a certificate to the specified proxy service. You can enter the **certificate rsa general-purpose trustpoint** subcommand multiple times for the proxy service.

If the trustpoint label is modified, the proxy service is momentarily taken out of service during the transition. Existing connections continue to use the old certificate until the connections are closed or cleared. New connections use the certificate from the new trustpoint, and the service is available again.

However, if the new trustpoint does not have a certificate yet, the operational status of the service remains down. New connections are not established until the new certificate is available. If the certificate is deleted by entering the **no certificate rsa general-purpose trustpoint** subcommand, the existing connections continue to use the certificate until the connections are closed or cleared. Although the certificate is obsolete, it is not removed from the proxy service until all connections are closed or cleared.

Note You can assign a generated self-signed certificate to a proxy service, but you cannot assign an imported self-signed certificate to a proxy service, because you cannot import the key pair of the certificate authority that signed the imported certificate. See the "Generating a Self-Signed Certificate" section on page C-1 for more information on self-signed certificates.

The following example shows how to assign a trustpoint to a proxy service:

```
ssl-proxy# configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
ssl-proxy(config)# ssl-proxy context c1
ssl-proxy(config-context)# service s2
ssl-proxy(config-ctx-ssl-proxy)# virtual ip 10.1.1.2 protocol tcp port 443
ssl-proxy(config-ctx-ssl-proxy)# server ip 20.0.0.3 p tcp p 80
ssl-proxy(config-ctx-ssl-proxy)# inservice
ssl-proxy(config-ctx-ssl-proxy)# certificate rsa general trustpoint tp-1
ssl-proxy(config-ctx-ssl-proxy)# end
ssl-proxy#
ssl-proxy# show ssl-proxy service s2
Service id:6, bound_service_id:262
Virtual IP:10.1.1.2, port:443
Server IP:20.0.0.3, port:80
rsa-general-purpose certificate trustpoint:tp-1
  Certificate chain in use for new connections:
    Server Certificate:
       Key Label:tp-1
       Serial Number:3C2CD2330001000000DB
    Root CA Certificate:
       Serial Number:313AD6510D25ABAE4626E96305511AC4
  Certificate chain complete
Admin Status:up
Operation Status:up
ssl-proxy#
```

The following example shows how to change a trustpoint for a proxy service:

Note The existing connections continue to use the old certificate until the connections are closed. The operational status of the service changes from up to down, and then up again. New connections use the new certificate.

```
ssl-proxy# configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
ssl-proxy(config)# ssl-proxy context c1
ssl-proxy(config-context)# service s2
ssl-proxy(config-ctx-ssl-proxy)# certificate rsa general trustpoint tp-2
ssl-proxy(config-ctx-ssl-proxy)# end
ssl-proxy#
```

```
ssl-proxy# show ssl-proxy service s2
Service id:6, bound_service_id:262
Virtual IP:10.1.1.2, port:443
Server IP:20.0.0.3, port:80
rsa-general-purpose certificate trustpoint:tp-2
  Certificate chain in use for new connections:
    Server Certificate:
       Key Label:k2
       Serial Number:70FCBFEC000100000D65
    Root CA Certificate:
       Serial Number:313AD6510D25ABAE4626E96305511AC4
  Obsolete certificate chain in use for old connections:
    Server Certificate:
       Key Label:tp-1
       Serial Number:3C2CD2330001000000DB
    Root CA Certificate:
       Serial Number:313AD6510D25ABAE4626E96305511AC4
  Certificate chain complete
Admin Status:up
Operation Status:up
ssl-proxy#
```

Renewing a Certificate

Some certificate authorities require you to generate a new key pair to renew a certificate, while other certificate authorities allow you to use the key pair of the expiring certificate to renew a certificate. Both cases are supported on the SSL Services Module.

The SSL server certificates usually expire in one or two years. Graceful rollover of certificates avoids sudden loss of services.

In the following example, proxy service s2 is assigned trustpoint t2:

```
ssl-proxy# configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
ssl-proxy(config)# ssl-proxy context c1
ssl-proxy(config-context)# service s2
ssl-proxy(config-ctx-ssl-proxy)# certificate rsa general-purpose trustpoint t2
ssl-proxy(config-ssl-proxy)# end
ssl-proxy#

ssl-proxy# show ssl-proxy service s2
Service id:0, bound_service_id:256
Virtual IP:10.1.1.1, port:443
Server IP:10.1.1.10, port:80
Nat pool:pool2
rsa-general-purpose certificate trustpoint:t2
  Certificate chain in use for new connections:
    Server Certificate:
       Key Label:k2
       Serial Number:1DFBB1FD000100000D48
    Root CA Certificate:
      Serial Number:313AD6510D25ABAE4626E96305511AC4
  Certificate chain complete
Admin Status:up
Operation Status:up
```

In the following example, the key pair for trustpoint t2 is refreshed, and the old certificate is deleted from the Cisco IOS database. Graceful rollover starts automatically for proxy service s2.

```
ssl-proxy# configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
ssl-proxy(config)# crypto key generate rsa general-key k2 exportable
% You already have RSA keys defined named k2.
% Do you really want to replace them? [yes/no]:yes
Choose the size of the key modulus in the range of 360 to 2048 for your
  General Purpose Keys. Choosing a key modulus greater than 512 may take
  a few minutes.

How many bits in the modulus [512]:1024
% Generating 1024 bit RSA keys ...[OK]
ssl-proxy(config)#end
ssl-proxy# show ssl-proxy service s2
Service id:0, bound_service_id:256
Virtual IP:10.1.1.1, port:443
Server IP:10.1.1.10, port:80
Nat pool:pool2
rsa-general-purpose certificate trustpoint:t2
  Certificate chain in graceful rollover, being renewed:
    Server Certificate:
       Key Label:k2
       Serial Number:1DFBB1FD000100000D48
    Root CA Certificate:
      Serial Number:313AD6510D25ABAE4626E96305511AC4
  Server certificate in graceful rollover
Admin Status:up
Operation Status:up
```

In the following example, existing and new connections use the old certificate until trustpoint t2 reenrolls. After trustpoint t2 reenrolls, new connections use the new certificate; existing connections continue to use the old certificate until the connections are closed.

```
ssl-proxy# configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
ssl-proxy(config)# crypto pki enroll t2
%
% Start certificate enrollment ..

% The subject name in the certificate will be:CN=host1.cisco.com
% The subject name in the certificate will be:ssl-proxy.cisco.com
% The serial number in the certificate will be:00000000
% The IP address in the certificate is 10.1.1.1

% Certificate request sent to Certificate Authority
% The certificate request fingerprint will be displayed.
% The 'show crypto pki certificate' command will also show the fingerprint.

    Fingerprint: 6518C579 A0498063 C5795057 A6170 075

ssl-proxy(config)# end
*Sep 24 15:19:34.339:%CRYPTO-6-CERTRET:Certificate received from Certificate Authority
```

```
ssl-proxy# show ssl-proxy service s2
Service id:0, bound_service_id:256
Virtual IP:10.1.1.1, port:443
Server IP:10.1.1.10, port:80
Nat pool:pool2
rsa-general-purpose certificate trustpoint:t2
  Certificate chain in use for new connections:
    Server Certificate:
       Key Label:k2
       Serial Number:2475A2FC000100000D4D
    Root CA Certificate:
      Serial Number:313AD6510D25ABAE4626E96305511AC4
  Obsolete certificate chain in use for old connections:
    Server Certificate:
       Key Label:k2
       Serial Number:1DFBB1FD000100000D48
    Root CA Certificate:
      Serial Number:313AD6510D25ABAE4626E96305511AC4
  Certificate chain complete
Admin Status:up
Operation Status:up
```

In the following example, the obsolete certificate is removed after all of the existing connections are closed.

```
ssl-proxy# show ssl-proxy service s2
Service id:0, bound_service_id:256
Virtual IP:10.1.1.1, port:443
Server IP:10.1.1.10, port:80
Nat pool:pool2
rsa-general-purpose certificate trustpoint:t2
  Certificate chain in use for new connections:
    Server Certificate:
       Key Label:k2
       Serial Number:2475A2FC000100000D4D
    Root CA Certificate:
      Serial Number:313AD6510D25ABAE4626E96305511AC4
  Certificate chain complete
Admin Status:up
Operation Status:up
```

Automatic Certificate Renewal and Enrollment

When you configure automatic enrollment, the module automatically requests a certificate from the certificate authority that is using the parameters in the configuration.

You can configure the certificate to automatically renew after a specified percentage of the validity time has passed. For example, if the certificate is valid for 300 days, and you specify *renewal_percent* as 80, the certificate automatically renews after 240 days have passed since the start validity time of the certificate.

Note The certificate authority certificate needs to be in the database prior to auto enrollment or renewal. Authenticate the trustpoint prior to configuring automatic enrollment. Also, configure a SCEP enrollment URL for the trustpoint.

To enable automatic enrollment and renewal and to display timer information, perform this task:

	Command	Purpose
Step 1	`ssl-proxy(config)# crypto pki trustpoint` *trustpoint-label*	Declares the trustpoint.
Step 2	`ssl-proxy(ca-trustpoint)# auto-enroll` {renewal*_percent*|**regenerate**}	Enables automatic renewal and enrollment for the specified trustpoint. **Note** Valid values for *renewal_percent* are 0 (enroll within 1 minute) through 100. **Note** The **regenerate** keyword generates a new key for the certificate even if a named key already exists.
Step 3	`ssl-proxy# show crypto pki timers`	Displays the time remaining before each timer expires.

This example shows how to enable auto enrollment and auto renewal:

```
ssl-proxy# configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
ssl-proxy(config)# crypto pki trustpoint tk21
ssl-proxy(ca-trustpoint)# auto-enroll 90
ssl-proxy(ca-trustpoint)# end
ssl-proxy# show crypto pki timers
PKI Timers
|          44.306
 |          44.306  RENEW tp-new
 |255d 5:28:32.348  RENEW tk21
ssl-proxy#
```

Enabling Key and Certificate History

When you enter the **ssl-proxy pki history** command, you enable the SSL proxy services key and certificate history. This history creates a record for each addition or deletion of the key pair and certificate chain for a proxy service.

When you enter the show **ssl-proxy certificate-history** command, the records are displayed. Each record logs the service name, key pair name, time of generation or import, trustpoint name, certificate subject name and issuer name, serial number, and date.

You can store up to 512 records in memory. For each record, a syslog message is generated. The oldest records are deleted after the limit of 512 records is reached.

To enable key and certificate history and display the records, perform this task:

	Command	Purpose
Step 1	`ssl-proxy(config)# ssl-proxy pki history`	Enables key and certificate history.
Step 2	`ssl-proxy# show ssl-proxy certificate-history` [**service** *proxy_service*]	Displays key and certificate history records for all services or the specified service.

This example shows how to enable the key and certificate history and display the records for a specified proxy service:

```
ssl-proxy# configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
ssl-proxy(config)# ssl-proxy pki history
ssl-proxy(config)# end
ssl-proxy# show ssl-proxy certificate-history service s2
  Record 1, Timestamp:00:00:22, 17:44:18 UTC Sep 29 2002
    Installed Server Certificate, Index 0
    Proxy Service:s2,  Trust Point:t2
    Key Pair Name:k2,  Key Usage:RSA General Purpose, Not Exportable
    Time of Key Generation:06:29:08 UTC Sep 28 2002
    Subject Name:CN = host1.cisco.com, OID.1.2.840.113549.1.9.2 = ssl-proxy.cisco.com,
OID.1.2.840.113549.1.9.8 = 10.1.1.1
    Issuer Name:CN = TestCA, OU = Lab, O = Cisco Systems, L = San Jose, ST = CA, C = US,
EA =<16> simpson-pki@cisco.com
    Serial Number:3728ADCD000100000D4F
    Validity Start Time:15:56:55 UTC Sep 28 2002
    End Time:16:06:55 UTC Sep 28 2003
    Renew Time:00:00:00 UTC Jan 1 1970
  End of Certificate Record
Total number of certificate history records displayed = 1
```

Caching Peer Certificates

You can configure the SSL Services Module to cache authenticated server and client (peer) certificates. Caching authenticated peer certificates saves time by not requiring the SSL Services Module to authenticate the same certificate again. The SSL Services Module uses the cached certificate information when it receives the same peer certificate within a specified timeout interval and the verify option (signature-only or all) matches.

Note When specifying **verifying all**, the CRL lookup or an ACL filtering result could change within the specified timeout interval, causing the SSL Services Module to incorrectly accept a certificate that should be denied. For instance, the SSL Services Module could cache a peer certificate, then, within the specified timeout, download an updated CRL in which the certificate is listed.

In an environment where the SSL Services Module repeatedly receives a number of certificates and the risk is acceptable, caching authenticated peer certificates can reduce the overhead.

For example, in a site-to-site VPN environment, where two SSL Services Modules authenticate each other's certificates during full handshakes, caching is applicable. A combination of verifying signature-only and caching authenticated peer certificates gives the best performance.

Peer certificates that expire within the specified time interval are not cached. The SSL Services Module uses separate cache entries for signature-only and verify-all options. Matching the verify options is one of the criteria for a cache hit.

Since the same peer certificate can be received on different proxy services at different time, and each proxy service has its own certificate authority pool, the issuer of the peer certificate may be in the certificate authority pool of the previous proxy service and not in the certificate authority pool of the current proxy service. This is considered a cache miss, and the peer certificate is verified.

To cache authenticated peer certificates, perform this task:

Command	Purpose
`ssl-proxy(config)# ssl-proxy pki cache size` *size* `timeout` *minutes*	Configures the certificate cache parameters. The default value for *size* is 0 (disabled); valid values are 0 through 5000 entries. The default value for *minutes* is 15 minutes; valid values are 1 through 600 minutes. To clear the cache, change the cache size or the timeout value.

Configuring Certificate Expiration Warning

You can configure the SSL Services Module to log warning messages when certificates have expired or will expire within a specified amount of time. When you enable certificate expiration warnings, the SSL Services Module checks every 30 minutes for the expiration information for the following:

- all the proxy services
- the certificate authority certificates associated with the proxy services
- all of the certificate authority trustpoints that are assigned to the trusted certificate authority pools

You can specify a time interval between 1 and 720 hours.

Note The SSL Services Module stores information about which certificates have been logged. Specifying 0 disables the warnings and clears the internal memory of any previously logged warning message. Specifying a time interval between 1 and 720 hours restarts the logging process. Log messages may not be displayed immediately after you restart logging. You may have to wait up to 30 minutes to see the first log message.

If a certificate will expire within the specified interval, or has already expired, the SSL Services Module logs a single warning message for the certificate.

In addition, you can enable the CISCO-SSL-PROXY-MIB certificate expiration trap to issue a trap each time a proxy service certificate expiration warning is logged. For more information on configuring SNMP traps, see the "Configuring SNMP Traps" section on page 4-18.

To enable the certificate expiration warning and configure the warning interval, perform this task:

Command	Purpose
`ssl-proxy(config)# ssl-proxy pki certificate check-expiring interval` *interval*	Enables certificate expiration warning and configures the warning interval. The default value for *interval* is 0 (disabled); valid values are 1 though 720 hours. **Note** Entering **0** disables warnings and clears the internal memory of any previously logged warning messages. No SNMP traps are sent.

This example shows how to enable the certificate expiration warning and configure the warning interval:

```
ssl-proxy# configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
ssl-proxy(config)# ssl-proxy pki certificate check-expiring interval 36
*Nov 27 03:44:05.207:%STE-6-PKI_CERT_EXP_WARN_ENABLED:Proxy service certificate expiration
warning has been enabled. Time interval is set to 36 hours.
ssl-proxy(config)# end
```

This example shows how to clear the internal memory of any previously logged warning messages and restart the logging process:

```
ssl-proxy# configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
ssl-proxy(config)# ssl-proxy pki certificate check-expiring interval 0
*Nov 27 03:44:15.207:%STE-6-PKI_CERT_EXP_WARN_DISABLED:Checking of certificate expiration
has been disabled.
ssl-proxy(config)# ssl-proxy pki certificate check-expiring interval 1
*Nov 27 03:44:16.207:%STE-6-PKI_CERT_EXP_WARN_ENABLED:Proxy service certificate expiration
warning has been enabled. Time interval is set to 36 hours.
ssl-proxy(config)# end
```

This example shows a syslog message for a proxy service certificate that will expire or has expired:

```
Jan  1 00:00:18.971:%STE-4-PKI_PROXY_SERVICE_CERT_EXPIRING:A proxy service certificate is
going to expire or has expired at this time:20:16:26 UTC Sep 5 2004, proxy service:s7,
trustpoint:tk21.
```

This example shows a syslog message for a certificate authority certificate that is associated with one or more proxy services that will expire or has expired:

```
Jan  1 00:00:18.971:%STE-4-PKI_PROXY_SERVICE_CA_CERT_EXPIRING:A CA certificate is going to
expire or has expired at this time:22:19:38 UTC Mar 4 2004, subject name:CN = ExampleCA,
OU = Example Lab, O = Cisco Systems, L = San Jose, ST = CA, C = US, EA =
example@cisco.com, serial number:313AD6510D25ABAE4626E96305511AC4.
```

This example shows a syslog message for a certificate authority certificate assigned to a trusted certificate authority pool that will expire or has expired:

```
Jan  1 00:00:18.971:%STE-4-PKI_CA_POOL_CERT_EXPIRING:A CA certificate in a CApool is going
to expire or has expired at this time:22:19:38 UTC Mar 4 2004, CA pool:pool2,
trustpoint:tp1-root.
```

Protected Private Key Storage

Digital signatures are a common method used to authenticate one device to another device. Certificates provide a key distribution mechanism that is required by digital signatures and public key cryptography. To use digital signatures, private information (the private key) must be stored on the device that is providing the signature. The stored private information may aid an attacker who steals the hardware device that contains the private key; for example, an attack may be able to cause the router to initiate a secure connection to another site by using the RSA private keys stored in the router.

The Protected Private Key Storage feature allows you to encrypt the private key that is stored in NVRAM by using a passphrase. In addition, you can also “lock” the private key, which blocks new connection attempts from a running router and protects the key in the router if the router is stolen by an attempted attacker.

Prerequisites

Before you can encrypt or lock a private key, you should perform the following tasks:

- Generate the RSA keys. (We recommend that you do not generate the RSA keys as exportable; however, doing so does not have an affect on this feature.)
- Optionally, you can authenticate and enroll each router with the certification authority (CA) server.

Note The RSA keys must be unlocked while enrolling the CA. The keys can be locked while authenticating the router with the CA because the private key of the router is not used during authentication.

Passphrase Limitations

If a passphrase is lost, you must regenerate the key, enroll with the CA server again, and obtain a new certificate. A lost passphrase cannot be recovered.

If you want to change a passphrase, you must decrypt the key with the current passphrase using the **crypto key decrypt rsa** command and encrypt the key once more to specify the new passphrase.

Encrypting and Locking Private Keys

To encrypt and lock the private keys, perform this task:

	Command	Purpose
Step 1	`ssl-proxy# configure terminal`	Enters global configuration mode.
Step 2	`ssl-proxy(config)# crypto key encrypt [write] rsa [name key-name] passphrase passphrase`	Encrypts the RSA keys. After this command is issued, the router can continue to use the key; the key remains unlocked. **Note** If the **write** keyword is not entered, the configuration must be manually written to NVRAM; otherwise, the encrypted key will be lost next time the router is reloaded.
Step 3	`ssl-proxy(config)# exit`	Exits global configuration mode.
Step 4	`ssl-proxy# show crypto key mypubkey rsa`	(Optional) Shows that the private key is encrypted (protected) and unlocked.
Step 5	`ssl-proxy# crypto key lock rsa [name key-name] passphrase passphrase`	(Optional) Locks the encrypted private key on a running router. **Note** After the key is locked, it cannot be used to authenticate the router to a peer device. This behavior disables any IPSec or SSL connections that use the locked key. Any existing IPSec tunnels created on the basis of the locked key will be closed. If all RSA keys are locked, SSH will automatically be disabled.

	Command	Purpose
Step 6	`ssl-proxy# show crypto key mypubkey rsa`	(Optional) Shows that the private key is protected and locked. The output will also show failed connection attempts via applications such as IKE, SSH, and SSL.
Step 7	`ssl-proxy# crypto key unlock rsa [name key-name] passphrase passphrase`	(Optional) Unlocks the private key.
Step 8	`ssl-proxy# configure terminal`	Enters global configuration mode
Step 9	`ssl-proxy# crypto key decrypt [write] rsa [name key-name] passphrase passphrase`	(Optional) Deletes the encrypted key and leaves only the unencrypted key. **Note** The **write** keyword immediately saves the unencrypted key to NVRAM. If the **write** keyword is not issued, the configuration must be manually written to NVRAM; otherwise, the key will remain encrypted the next time the router is reloaded.

The following example shows how to encrypt the RSA key "pki1-72a.cisco.com." Thereafter, the **show crypto key mypubkey rsa** command is issued to verify that the RSA key is encrypted (protected) and unlocked.

```
ssl-proxy(config)# crypto key encrypt rsa name pki1-72a.cisco.com passphrase cisco1234
ssl-proxy(config)# exit
ssl-proxy# show crypto key mypubkey rsa
% Key pair was generated at:00:15:32 GMT Jun 25 2003
Key name:pki1-72a.cisco.com
Usage:General Purpose Key
*** The key is protected and UNLOCKED. ***
Key is not exportable.
Key Data:
305C300D 06092A86 4886F70D 01010105 00034B00 30480241 00E0CC9A 1D23B52C
CD00910C ABD392AE BA6D0E3F FC47A0EF 8AFEE340 0EC1E62B D40E7DCC
23C4D09E
03018B98 E0C07B42 3CFD1A32 2A3A13C0 1FF919C5 8DE9565F 1F020301 0001
% Key pair was generated at:00:15:33 GMT Jun 25 2003
Key name:pki1-72a.cisco.com.server
Usage:Encryption Key
Key is exportable.
Key Data:
307C300D 06092A86 4886F70D 01010105 00036B00 30680261 00D3491E 2A21D383
854D7DA8 58AFBDAC 4E11A7DD E6C40AC6 66473A9F 0C845120 7C0C6EC8 1FFF5757
3A41CE04 FDCB40A4 B9C68B4F BC7D624B 470339A3 DE739D3E F7DDB549 91CD4DA4
DF190D26 7033958C 8A61787B D40D28B8 29BCD0ED 4E6275C0 6D020301 0001
ssl-proxy#
```

The following example shows how to lock the key "pki1-72a.cisco.com." Thereafter, the **show crypto key mypubkey rsa** command is issued to verify that the key is protected (encrypted) and locked.

```
ssl-proxy# crypto key lock rsa name pki1-72a.cisco.com passphrase cisco1234
ssl-proxy# show crypto key mypubkey rsa
% Key pair was generated at:20:29:41 GMT Jun 20 2003
Key name:pki1-72a.cisco.com
Usage:General Purpose Key
*** The key is protected and LOCKED. ***
Key is exportable.
Key Data:
305C300D 06092A86 4886F70D 01010105 00034B00 30480241 00D7808D C5FF14AC
```

```
0D2B55AC 5D199F2F 7CB4B355 C555E07B 6D0DECBE 4519B1F0 75B12D6F 902D6E9F
B6FDAD8D 654EF851 5701D5D7 EDA047ED 9A2A619D 5639DF18 EB020301 0001
ssl-proxy#
```

The following examples show how to verify that a key that is associated with a service is locked:

```
ssl-proxy# show ssl-proxy service
No context name provided, assuming context 'Default'...


Proxy Service Name  Context Name         Admin  Operation
                                         status status
service1            Default              up     down (No cert)

ssl-proxy# show ssl-proxy service service1
No context name provided, assuming context 'Default'...

Service id: 1, bound_service_id: 257
Virtual IP: 208.122.0.12, port: 443
Server IP: 209.0.207.203, port: 80
Virtual SSL Policy: test
Nat pool: np
rsa-general-purpose certificate trustpoint: test1
  Obsolete certificate chain for old connections:
    Certificate:
       Key Label: test1, 1024-bit, exportable
       Key Timestamp: 01:42:14 UTC Jul 12 2005
       Serial Number: 66A94DAC0000000000F7
    Root CA Certificate:
       Serial Number: 05A8229
  Keypair of trustpoint is locked
Context name: Default
Context Id  : 0
Admin Status: up
Operation Status: down
Proxy status: No certificate
```

Automatically Unlocking the Key Pair

In the event of a reload, you are able to unlock the key pair automatically.

You can store a passphrase in the startup configuration. The passphrase is encrypted using Advanced Encryption Standard (AES) encryption before it is saved to the configuration. When reading the passphrase from the configuration, the passphrase is treated as AES encrypted and is decrypted and used to unlock the key pair.

To unlock the keys automatically after a reload, perform this task in global configuration mode:

	Command	Purpose
Step 1	`ssl-proxy(config)# ssl-proxy crypto key unlock rsa` *key-name* `passphrase` *passphrase*	Unlocks the key automatically after a reload.
Step 2	`ssl-proxy(config)# exit`	Exits configuration mode.
Step 3	`ssl-proxy# copy system:running-config nvram:startup-config`	Saves the configuration.

Configuring Virtualization

When you configure virtualization, you are able to divide the SSL Services Module into multiple instances called contexts. Observe the following guidelines regarding virtualization:

- When you upgrade a previous SSL software release to SSL software release 3.x, your configuration is converted to a default context (Default) that is visible after the image is loaded on the SSL Services Module. After the conversion, you can configure additional contexts and assign them to different enterprises.
- You can configure multiple services per context.

Note You can configure up to 256 contexts; however, you can only have a maximum of 256 services configured per module, regardless of how many contexts you have configured.

- Services and policies are locally unique to a context.
- Statistics can be collected either per service or per context (by aggregating the statistics for all the services that are configured for that context).
- You can limit the maximum number of connections per context.

To configure the virtual context. perform this task:

	Command	Purpose
Step 1	`ssl-proxy(config)# ssl-proxy context name`	Enters the SSL context subcommand mode. The optional SSL context name is used to specify an SSL virtualization instance. The context name is case-sensitive.
Step 2	`ssl-proxy(config-context)# policy {health-probe tcp name | http-header | ssl policy-name | tcp policy-name | url-rewrite}`	Configures the various policies. See the "Configuring Policies" section on page 4-1 for information on configuring policies.
Step 3	`ssl-proxy(config-context)# maxconns connections`	(Optional) Configures the maximum number of connections for this context. The default is 192000. Valid values are 1 to 192000.
Step 4	`ssl-proxy(config-context)# natpool name start_ipaddr end_ipaddr netmask ipaddr`	Configures the NAT pool settings.
Step 5	`ssl-proxy(config-context)# pool ca name`	Configures a pool of resources.
Step 6	`ssl-proxy(config-context)# service service_name`	Enters SSL proxy service subcommand mode and lets you configure the SSL client or server proxy service. See the "Configuring SSL Proxy Services" section on page 3-45 for information on SSL proxy services.
Step 7	`ssl-proxy(config-context)# vrf-name name`	Configures the VRF associated with this context.
Step 8	`ssl-proxy(config-context)# description description`	(Optional) Allows you to enter a short description for this context.
Step 9	`ssl-proxy(config-context)# exit`	Exit from SSL context mode.

Configuring SSL Proxy Services

You define SSL proxy services using the **service** *ssl_proxy_name* command. You can configure the virtual IP address and port associated with the proxy service, and the associated target IP address and port. You define TCP and SSL policies for both client (**virtual**) and server (**server**) sides of the proxy.

The following sections describe how to configure proxy services:

SSL Server Proxy Services

To configure SSL server proxy services, perform this task:

	Command	Purpose
Step 1	`ssl-proxy(config)# ssl-proxy context` *name*	Enters the SSL context subcommand mode. The optional SSL context name is used to specify an SSL virtualization instance. The context name is case-sensitive.
Step 2	`ssl-proxy(config-context)# service` *service_name*	Defines the name of the SSL proxy service. **Note** You cannot use the same *service_name* for both the server proxy service and the client proxy service. **Note** The *service_name* value is case-sensitive.
Step 3	`ssl-proxy(config-ctx-ssl-proxy)# virtual ipaddr` *ip_addr* [*mask_addr*][1] `protocol tcp port` *port* [**secondary**[2,3,4]]	Defines the virtual server IP address, transport protocol (TCP), and port number for which the SSL Services Module is the proxy. **Note** The **secondary** keyword is required if the virtual IP address is not on a network with a direct connection.
Step 4	`ssl-proxy(config-ctx-ssl-proxy)# server ipaddr` *ip_addr* `protocol tcp port` *port*	Defines the IP address, port number, and the transport protocol of the target server for the proxy. **Note** The target server IP address can be a virtual IP address of an SLB device or a real IP address of a web server.
Step 5	`ssl-proxy(config-ctx-ssl-proxy)# virtual policy ssl` *ssl_policy_name*[5]	(Optional) Applies an SSL policy to the client side of the proxy server. See the "Configuring SSL Policy" section on page 4-2 for SSL policy parameters.
Step 6	`ssl-proxy(config-ctx-ssl-proxy)# virtual policy tcp` *tcp_policy_name*[5]	(Optional) Applies a TCP policy to the client side of the proxy server. See the "Configuring TCP Policy" section on page 4-5 for TCP policy parameters.
Step 7	`ssl-proxy(config-ctx-ssl-proxy)# server policy tcp` *tcp_policy_name*	(Optional) Applies a TCP policy to the server side of the proxy server. See the "Configuring TCP Policy" section on page 4-5 to configure the TCP policy.

	Command	Purpose
Step 8	`ssl-proxy(config-ctx-ssl-proxy)#` **`policy http-header`** *`http_header_policy_name`*	(Optional) Applies the HTTP header policy to proxy server. See the "HTTP Header Insertion" section on page 4-7 to configure the HTTP header insertion policy.
Step 9	`ssl-proxy(config-ctx-ssl-proxy)#` **`policy url-rewrite`** *`url_rewrite_policy_name`*	(Optional) Applies the URL rewrite policy. See the "Configuring URL Rewrite" section on page 4-11 to configure the URL rewrite policy.
Step 10	`ssl-proxy(config-ctx-ssl-proxy)#` **`policy health-probe tcp`** *`name`*	Applies the health probe policy. See the "Health Probe" section on page 4-13 to configure the health probe policy.
Step 11	`ssl-proxy(config-ctx-ssl-proxy)#` **`trusted-ca`** *`ca_pool_name`*	(Optional) Associates the trusted certificate authority pool with the proxy service. See the "Server Certificate Authentication" section on page 3-55 for information on certificate authority pools.
Step 12	`ssl-proxy(config-ctx-ssl-proxy)#` **`authenticate verify`** {**`signature-only`**[6] | **`all`**[7]}[**`optional`**]	(Optional) Enables server certificate authentication and specifies the form of verification. See the "Server Certificate Authentication" section on page 3-55 for information on server certificate authentication.
Step 13	`ssl-proxy(config-ctx-ssl-proxy)#` **`nat`** {**`server`** | **`client`** *`natpool_name`*}	(Optional) Specifies the usage of either server NAT (network address translation) or client NAT for the server-side connection opened by the SSL Services Module. See the "Configuring NAT" section on page 4-15.
Step 14	`ssl-proxy(config-ctx-ssl-proxy)#` **`certificate rsa general-purpose trustpoint`** *`trustpoint_label`*	Applies a trustpoint configuration to the proxy server[8]. **Note** The trustpoint defines the certificate authority server, the key parameters and key-generation methods, and the certificate enrollment methods for the proxy server. See the "Declaring the Trustpoint" section on page 3-7 for information on configuring the trustpoint.
Step 15	`ssl-proxy(config-ctx-ssl-proxy)#` **`inservice`**	Sets the proxy server as administratively Up. **Note** An SSL proxy service can be configured prior to configuring the associated trustpoint or downloading the corresponding certificate, but the service will not enter the administratively Up state until a certificate is obtained.

1. Configure the mask address to specify a wildcard proxy service. You must enter the **secondary** keyword to configure a wildcard proxy service.
2. When you enter the **secondary** keyword, the SSL Services Module does not respond to ARP requests of the virtual IP address.
3. You can enter the **secondary** keyword when the SSL Services Module is used in a standalone configuration or when the SSL Services Module is used as a real server on a load balancer (like the CSM) configured in dispatch mode (MAC address rewrite). See Chapter 5, "Configuring Different Modes of Operation," for information on configuring the SSL Services Module with the CSM.
4. You can enter the **secondary** keyword if you configure multiple devices using the same virtual IP address. The virtual IP address can be any legal IP address, and does not have to be in the VLAN (subnet) connected to the SSL Services Module.
5. If you create a policy without specifying any parameters, the policy is created using the default values.
6. When you verify signature-only, authentication stops at the level corresponding to one of the trusted certificate authority trustpoints in the trusted certificate authority pool.
7. When you verify all, the highest level issuer in the certificate chain must be configured as a trusted certificate authority trustpoint. The SSL Services Module authenticates all the certificates in the peer certificate chain and stops only at the highest level certificate authority. There must be a certificate authority trustpoint for the highest level certificate authority, and this trustpoint should be authenticated.

8. If the key (modulus) size is other than 512, 768, 1024, 1536, or 2048, you will receive an error and the trustpoint configuration is not applied. Replace the key by generating a key (using the same *key_label*) and specifying a supported modulus size, then repeat Step 14.

This example shows how to configure SSL proxy services:

```
ssl-proxy(config)# ssl-proxy context c1
ssl-proxy(config-context)# service proxy1
ssl-proxy(config-ctx-ssl-proxy)# virtual ipaddr 10.1.1.100 protocol tcp port 443
ssl-proxy(config-ctx-ssl-proxy)# server ipaddr 10.1.1.1 protocol tcp port 80
ssl-proxy(config-ctx-ssl-proxy)# virtual policy tcp tcp2
ssl-proxy(config-ctx-ssl-proxy)# server policy tcp tcp2
ssl-proxy(config-ctx-ssl-proxy)# virtual policy ssl ssl1
ssl-proxy(config-ctx-ssl-proxy)# nat client t2
ssl-proxy(config-ctx-ssl-proxy)# certificate rsa general-purpose trustpoint tp1
ssl-proxy(config-ctx-ssl-proxy)# inservice
ssl-proxy(config-ctx-ssl-proxy)# end
```

If you have many virtual and server IP addresses to manage and configure, you can configure a wildcard proxy service.

This example shows how to configure a wildcard SSL proxy service, so that **proxy1** accepts virtual IP addresses 10.0.0.1 through 10.25.255.254:

```
ssl-proxy(config)# ssl-proxy context c1
ssl-proxy(config-context)# service proxy1
ssl-proxy(config-ssl-proxy)# virtual ipaddr 10.0.0.0 255.0.0.0 protocol tcp port 443
secondary
ssl-proxy(config-ctx-ssl-proxy)# server ipaddr 20.1.2.3 protocol tcp port 80
ssl-proxy(config-ctx-ssl-proxy)# virtual policy tcp tcp2
ssl-proxy(config-ctx-ssl-proxy)# server policy tcp tcp2
ssl-proxy(config-ctx-ssl-proxy)# virtual policy ssl ssl1
ssl-proxy(config-ctx-ssl-proxy)# inservice
ssl-proxy(config-ctx-ssl-proxy)# end
```

SSL Version 2.0 Forwarding

The SSL Services Module is not able to terminate SSL version 2.0 (SSLv2) connections. However, you can configure the SSL Services Module to forward SSLv2 connections to another server by entering the **sslv2** keyword at the **server** command. When you configure the SSLv2 server IP address, the SSL Services Module transparently forwards all SSLv2 connections to that server. If you require SSLv2 forwarding, you need to configure the SSLv2 server IP address in addition to the IP address of the server that is used for offloading SSL version 3.0 or Transport Layer Security (TLS) version 1.0 connections.

To configure SSLv2 forwarding, enter this command:

`ssl-proxy(config-ctx-ssl-proxy)#` **`server ipaddr`** *`ip_addr`* **`protocol tcp port`** *`port`* **`sslv2`**[1]	Defines the IP address, port number, and the transport protocol of the target server for the proxy. **Note** The target server IP address can be a virtual IP address of an SLB device or a real IP address of a web server.

1. Enter the **sslv2** keyword to forward SSL version 2.0 client connections to a SSL v2.0 server. When you enter **sslv2**, configure another server IP address to offload SSL version 3.0 or Transport Layer Security (TLS) version 1.0 connections.

This example shows how to configure SSL proxy services to forward SSL v2.0 connections:

```
ssl-proxy(config)# ssl-proxy context c1
ssl-proxy(config-context)# service frontend
ssl-proxy(config-ctx-ssl-proxy)# virtual ipaddr 35.200.200.102 protocol tcp port 443
ssl-proxy(config-ctx-ssl-proxy)# server ipaddr 26.51.51.1 protocol tcp port 80
```

```
ssl-proxy(config-ctx-ssl-proxy)# server ipaddr 26.51.51.2 protocol tcp port 443 sslv2
ssl-proxy(config-ctx-ssl-proxy)# certificate rsa general-purpose trustpoint test-cert
ssl-proxy(config-ctx-ssl-proxy)# inservice
ssl-proxy(config-ctx-ssl-proxy)# end
```

SSL Client Proxy Services

You configure SSL client proxy services to specify that the proxy service accepts clear text traffic, encrypts the traffic into SSL traffic, and forwards the traffic to the backend SSL server.

While you are required to configure a certificate for the SSL server proxy, you are not required to configure a certificate for SSL client proxy. If you configure the certificate for the SSL client proxy, that certificate is sent in response to the CertificateRequest message that is sent by the server during the client authentication phase of the handshake protocol.

Note SSL policies are configured at the **server** subcommand for SSL client proxy services; SSL policies are configured at the **virtual** subcommand for SSL server proxy services.

To configure SSL client proxy services, perform this task:

	Command	Purpose
Step 1	`ssl-proxy(config)# ssl-proxy context` *name*	Enters the SSL context subcommand mode. The optional SSL context name is used to specify an SSL virtualization instance.
Step 2	`ssl-proxy(config-context)# service` *proxy_name* `client`	Defines the name of the SSL proxy service. The **client** keyword configures the SSL client proxy service. **Note** You cannot use the same *service_name* for both the server proxy service and the client proxy service. **Note** The *proxy-name* value is case-sensitive.
Step 3	`ssl-proxy(config-ctx-ssl-proxy)# virtual ipaddr` *ip_addr* `[`*mask_addr*`]`[1] `protocol tcp port` *port* `[secondary`[2,3,4]`]`	Defines the virtual server IP address, transport protocol (TCP), and port number for which the SSL Services Module is the proxy. **Note** The **secondary** keyword is required if the virtual IP address is not on a network with a direct connection.
Step 4	`ssl-proxy(config-ctx-ssl-proxy)# server ipaddr` *ip_addr* `protocol tcp port` *port*	Defines the IP address, port number, and the transport protocol of the target server for the proxy. **Note** The target server IP address can be a virtual IP address of an SLB device or a real IP address of a web server.
Step 5	`ssl-proxy(config-ctx-ssl-proxy)# virtual policy tcp` *tcp_policy_name*[5]	(Optional) Applies a TCP policy to the client side of the proxy server. See the "Configuring TCP Policy" section on page 4-5 for TCP policy parameters.
Step 6	`ssl-proxy(config-ctx-ssl-proxy)# server policy ssl` *ssl_policy_name*[5]	(Optional) Applies an SSL policy to the server side of the proxy server. See the "Configuring SSL Policy" section on page 4-2 for SSL policy parameters.

	Command	Purpose
Step 7	`ssl-proxy(config-ctx-ssl-proxy)#` **`server policy tcp`** *`tcp_policy_name`*	(Optional) Applies a TCP policy to the server side of the proxy server. See the "Configuring TCP Policy" section on page 4-5 for TCP policy parameters.
Step 8	`ssl-proxy(config-ctx-ssl-proxy)#` **`policy http-header`** *`http_header_policy_name`*	(Optional) Applies the HTTP header policy to proxy server. See the "HTTP Header Insertion" section on page 4-7 to configure the HTTP header insertion policy.
Step 9	`ssl-proxy(config-ctx-ssl-proxy)#` **`policy url-rewrite`** *`url_rewrite_policy_name`*	(Optional) Applies the URL rewrite policy. See the "Configuring URL Rewrite" section on page 4-11 to configure the URL rewrite policy.
Step 10	`ssl-proxy(config-ctx-ssl-proxy)#` **`policy health-probe tcp`** *`name`*	Configures the health probe policy. See the "Health Probe" section on page 4-13 to configure the health probe policy.
Step 11	`ssl-proxy(config-ctx-ssl-proxy)#` **`trusted-ca`** *`ca_pool_name`*	Associates the trusted certificate authority pool with the proxy service. See the "Client Certificate Authentication" section on page 3-51 for information on certificate authority pools.
Step 12	`ssl-proxy(config-ctx-ssl-proxy)#` **`authenticate verify`** {**`signature-only`**[6] | **`all`**[7]}	Enables client certificate authentication and specifies the form of verification. See the "Client Certificate Authentication" section on page 3-51 for information on client certificate authentication.
Step 13	`ssl-proxy(config-ctx-ssl-proxy)#` **`nat`** {**`server`** | **`client`** *`natpool_name`*}	(Optional) Specifies the usage of either server NAT (network address translation) or client NAT for the server-side connection opened by the SSL Services Module. See the "Configuring NAT" section on page 4-11.
Step 14	`ssl-proxy(config-ctx-ssl-proxy)#` **`certificate rsa general-purpose trustpoint`** *`trustpoint_label`*	(Optional) Applies a trustpoint configuration to the client proxy. **Note** The trustpoint defines the certificate authority server, the key parameters and key-generation methods, and the certificate enrollment methods for the proxy server. See the "Declaring the Trustpoint" section on page 3-7 for information on configuring the trust point.
Step 15	`ssl-proxy(config-ctx-ssl-proxy)#` **`inservice`**	Sets the proxy server as administratively Up.

1. Configure the mask address to specify a wildcard proxy service. You must enter the **secondary** keyword to configure a wildcard proxy service.
2. When you enter the **secondary** keyword, the SSL Services Module does not respond to ARP requests of the virtual IP address.
3. You can enter the **secondary** keyword when the SSL Services Module is used in a standalone configuration or when the SSL Services Module is used as a real server on a load balancer (like the CSM) configured in dispatch mode (MAC address rewrite). See Chapter 5, "Configuring Different Modes of Operation" for information on configuring the SSL Services Module with the CSM.
4. You can enter the **secondary** keyword if you configure multiple devices using the same virtual IP address. The virtual IP address can be any legal IP address, and does not have to be in the VLAN (subnet) connected to the SSL Services Module.
5. If you create a policy without specifying any parameters, the policy is created using the default values.
6. When you verify signature-only, authentication stops at the level corresponding to one of the trusted certificate authority trustpoints in the trusted certificate authority pool.
7. When you verify all, the highest level issuer in the certificate chain must be configured as a trusted certificate authority trustpoint. The SSL Services Module authenticates all the certificates in the peer certificate chain and stops only at the highest level certificate authority. There must be a certificate authority trustpoint for the highest level certificate authority, and this trustpoint should be authenticated.

This example shows how to configure SSL client proxy services:

```
ssl-proxy(config)# ssl-proxy context c1
ssl-proxy(config-context)# service proxy1 client
ssl-proxy(config-ctx-ssl-proxy)# virtual ipaddr 10.1.1.100 protocol tcp port 80
ssl-proxy(config-ctx-ssl-proxy)# virtual policy tcp tcp2
ssl-proxy(config-ctx-ssl-proxy)# server ipaddr 10.1.1.1 protocol tcp port 443
ssl-proxy(config-ctx-ssl-proxy)# server policy tcp tcp2
ssl-proxy(config-ctx-ssl-proxy)# server policy ssl ssl1
ssl-proxy(config-ctx-ssl-proxy)# inservice
ssl-proxy(config-ctx-ssl-proxy)# end
```

IP Fragment Reassembly

The SSL Services Module receives IP fragments if an intermediate router (between the client and the SSL Services Module) cannot transmit the IP segment over its physical interface. In this case, the router fragments the packet. Each of these fragments might take a different path. The fragments are reasembled at the SSL Services Module.

The individual fragments might also be duplicated and take different paths which may further fragment the packet, which leads to packets that contain the same data bytes. Fragments may overlap if successive retransmissions use different packetizing but the same reassembly ID.

IP fragmentation is enabled by default. To adjust the IP fragment reassembly timer, perform this task:

Command	Purpose
`ssl-proxy(config)# ssl-proxy ip-frag-ttl` [*time*]	Configures the IP reassembly timeout. The default is 6 seconds. Valid values are from 3 to 120 seconds.
`ssl-proxy# show ssl-proxy ip-frag-ttl`	Displays the IP reassembly timeout.

Configuring Certificate Authentication

This section describes how to configure client and server authentication:

When you configure client or server certificate authentication, you need to specify the form of verification as **signature-only** or **all**. Both options check the validity start time and validity end time of each certificate being authenticated. If the start time is in the future or the end time has passed, the SSL Services Module does not accept the certificate.

When you enter the **verify signature-only** command, the SSL Services Module verifies the certificate chain from the peer certificate to the next certificate (which should be the issuer of the previous certificate), then to the next certificate, and so on until one of the following conditions is met:

- The certificate is issued by a trusted certificate authority, or the certificate itself matches a trusted certificate authority certificate, and the trusted certificate authority is in the certificate authority pool assigned to the proxy service. In this case, the chain is accepted, and the rest of the chain does not need to be verified.
- The end of the chain is reached, and the last certificate in the chain is not issued by a trusted certificate authority. In this case, the chain is rejected.

When you enter the **verify all** command, the SSL Services Module sorts the certificate chain in order, ignoring any unrelated or redundant certificates. The SSL Services Module determines if the top-most certificate in the sorted chain is issued by a trusted certificate authority or if it matches a trusted certificate authority certificate.

If the SSL Services Module cannot trust the top-most certificate, the chain is rejected.

If the SSL Services Module trusts the top-most certificate, then the SSL Services Module performs the following for each certificate in the chain:

- Verifies the signature of each certificate.
- If the certificate is associated with one or more trustpoints, the SSL Services Module selects one of these trustpoints. Depending on the certificate checking and ACL map configuration for this trustpoint, the SSL Services Module performs revocation and certificate attribute filtering. If the certificate checking denies the certificate, the SSL Services Module rejects the chain.
- If the certificate is a X509 version 3 certificate authority certificate, the SSL Services Module verifies that the Basic Constraints extension is present and valid. If the Basic Constraints extension is not present or valid, the chain is rejected.

If you verify only the signature, that verification process checks only the validity and signature of a minimum number of certificates in the chain. Verifying all performs more checking and validates all the certificates received, but takes longer and uses more CPU time.

See the "Checking the Certificate Status" section on page 3-58 for information about on how to check certificate status using OCSP and CRLs.

Client Certificate Authentication

When you configure the SSL Services Module as an SSL server, you can configure the SSL Services Module to authenticate the SSL client. In this case, the SSL Services Module requests a certificate from the SSL client for authentication.

To authenticate the SSL client, the SSL Services Module verifies the following:

- The certificate at one level is properly signed by the issuer at the next level.
- At least one of the issuer certificates in the certificate chain is trusted by the SSL proxy service.
- None of the certificates in the certificate chain is in the certificate revocation list (CRL) and rejected by any access control list (ACL).

For verifying the SSL client certificates, the SSL Services Module is configured with a list of trusted certificate authorities (certificate authority pool). A trusted certificate authority pool is a subset of the trusted certificate authorities in the database. The SSL Services Module trusts only the certificates issued by the certificate authorities that you configure in the certificate authority pool.

When you configure the SSL Services Module as an SSL server, and it needs to authenticate the client's certificate,

Note For a proxy service to be operationally up, the certificate authority pool must have at least one trustpoint that has a certificate. If none of the trustpoints in the certificate authority pool has a certificate, the proxy service will go down automatically.

Note Authentication may fail if a particular level of certificate authority in the hierarchy is not included in the certificate authority pool. To avoid this type of failure and to improve efficiency when verifying signature-only for authentication, add all levels of subordinate certificate authorities together with the root certificate authority into a certificate authority pool.

Note If a certificate authority trustpoint is deleted, you should remove the corresponding trustpoint from the trusted certificate authority pool.

To configure client authentication, perform this task:

	Command	Purpose
Step 1	`ssl-proxy(config)# ssl-proxy context` *name*	Enters the SSL context subcommand mode. The optional SSL context name is used to specify an SSL virtualization instance.
Step 2	`ssl-proxy(config-context)# pool ca` *ca_pool_name*	Creates the certificate authority pool. **Note** You can create up to eight pools.
Step 3	`ssl-proxy(config-ctx-ca-pool)# ca trustpoint` *ca_trustpoint_label*[1]	Adds a trusted certificate authority to the pool. **Note** You can add up to 16 trusted certificate authorities per pool.
Step 4	`ssl-proxy(config-ctx-ca-pool)# exit`	Returns to config mode.
Step 5	`ssl-proxy(config-context)# service` *proxy_name*	Defines the name of the SSL server proxy service.
Step 6	`ssl-proxy(config-ctx-ssl-proxy)# trusted-ca` *ca_pool_name*	Associates the trusted certificate authority pool with the proxy service.

	Command	Purpose
Step 7	`ssl-proxy(config-ctx-ssl-proxy)#` **`authenticate verify`** {**`signature-only`**[2] | **`all`**[3]}	Enables client certificate authentication and specifies the form of verification. **Note** The **signature-only** keyword verifies the signature only, without looking up the CRL or matching ACL. The default is **all**.
Step 8	`ssl-proxy(config-ctx-ssl-proxy)#` **`exit`**	Exits from configuration mode.

1. The SSL Services Module supports up to eight levels of certificate authority. We recommend that you add all levels of certificate authority, or at least the root certificate authority, to the certificate authority pool.
2. When you verify signature-only, authentication stops at the level corresponding to one of the trusted certificate authority trustpoints in the trusted certificate authority pool.
3. When you verify all, the highest level issuer in the certificate chain must be configured as a trusted certificate authority trustpoint. The SSL Services Module authenticates all the certificates in the peer certificate chain and stops only at the highest level certificate authority. There must be a certificate authority trustpoint for the highest level certificate authority, and this trustpoint should be authenticated.

This example shows how to configure client certificate authentication verifying signature only:

```
ssl-proxy(config)# crypto pki trustpoint rootca
ssl-proxy(ca-trustpoint)# enrollment terminal
ssl-proxy(ca-trustpoint)# exit
ssl-proxy(config)#
ssl-proxy(config)# crypto pki authenticate rootca

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

-----BEGIN CERTIFICATE-----
MIIC1zCCAoGgAwIBAgIQadUxzU/i97hDmZRYJ1bBcDANBgkqhkiG9w0BAQUFADB1
MQswCQYDVQQGEwJVUzETMBEGA1UECBMKY2FsaWZvcm5pYTERMA8GA1UEBxMIc2Fu
IGpvc2UxDjAMBgNVBAoTBWNpc2NvMQwwCgYDVQQLEwNoc3MxIDAeBgNVBAMTF3Np
bXBzb24tZGV2dGVzdC1yb290LUNBMB4XDTAzMTExMTIxNDgwMloXDTEzMTExMTIx
NTczOVowdTELMAkGA1UEBhMCVVMxEzARBgNVBAgTCmNhbGlmb3JuaWExETAPBgNV
BAcTCHNhbiBqb3NlMQ4wDAYDVQQKEwVjaXNjbzEMMAoGA1UECxMDaHNzMSAwHgYD
VQQDExdzaW1wc29uLWRldnRlc3Qtcm9vdC1DQTBcMA0GCSqGSIb3DQEBAQUAA0sA
MEgCQQCWEibAnUlVqQNUn0Wb94qnHi8FKjmVhibLHGRl6J+V7gHgzmF2MTz5WP5l
VQ2/1NVu0HjUORRdeCm1/raKJ/7ZAgMBAAGjgewwgekwCwYDVR0PBAQDAgHGMA8G
A1UdEwEB/wQFMAMBAf8wHQYDVR0OBBYEFCYGLUBTKNd9EgUonHnoSvbHg0axMIGX
BgNVHR8EgY8wgYwwQ6BBoD+GPWh0dHA6Ly9jaXNjby1sOGo2b2hwbnIvQ2VydEVu
cm9sbC9zaW1wc29uLWRldnRlc3Qtcm9vdC1DQS5jcmwwRaBDoEGGP2ZpbGU6Ly9c
XGNpc2NvLWw4ajZvaHBuclxDZXJ0RW5yb2xsXHNpbXBzb24tZGV2dGVzdC1yb290
LUNBLmNybDAQBgkrBgEEAYI3FQEEAwIBADANBgkqhkiG9w0BAQUFAANBACBqe1wy
YjalelGZqLVu4bDVMFo6ELCV2AMBgi41K3ix+Z/03PJd7ct2BIAF4lktv9pCe6IO
EoBcmZteA+TQcKg=
-----END CERTIFICATE-----

Certificate has the following attributes:
Fingerprint:AC6FC55E CC29E891 0DC3FAAA B4747C10
% Do you accept this certificate? [yes/no]:yes
Trustpoint CA certificate accepted.
% Certificate successfully imported

ssl-proxy(config)# ssl-proxy context s1
ssl-proxy(config-context)# pool ca rootca
ssl-proxy(config-ctx-ca-pool)# ca trustpoint rootca
ssl-proxy(config-ctx-ca-pool)# exit
ssl-proxy(config-context)# service client-auth-sig-only
ssl-proxy(config-ctx-ssl-proxy)# virtual ipaddr 14.0.0.1 protocol tcp port 443
```

```
ssl-proxy(config-ctx-ssl-proxy)# server ipaddr 24.0.0.1 protocol tcp port 80
ssl-proxy(config-ctx-ssl-proxy)# certificate rsa general-purpose trustpoint test-cert
ssl-proxy(config-ctx-ssl-proxy)# trusted-ca rootca
ssl-proxy(config-ctx-ssl-proxy)# authenticate verify signature-only
ssl-proxy(config-ctx-ssl-proxy)# inservice
ssl-proxy(config-ctx-ssl-proxy)# exit
```

This example shows how to configure client certificate authentication verifying all:

```
ssl-proxy(config)# crypto pki trustpoint rootca
ssl-proxy(ca-trustpoint)# enrollment terminal
ssl-proxy(ca-trustpoint)# exit
ssl-proxy(config)#
ssl-proxy(config)# crypto pki authenticate rootca

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

-----BEGIN CERTIFICATE-----
MIIC1zCCAoGgAwIBAgIQadUxzU/i97hDmZRYJ1bBcDANBgkqhkiG9w0BAQUFADB1
MQswCQYDVQQGEwJVUzETMBEGA1UECBMKY2FsaWZvcm5pYTERMA8GA1UEBxMIc2Fu
IGpvc2UxDjAMBgNVBAoTBWNpc2NvMQwwCgYDVQQLEwNoc3MxIDAeBgNVBAMTF3Np
bXBzb24tZGV2dGVzdC1yb290LUNBMB4XDTAzMTExMTIxNDgwMloXDTEzMTExMTIx
NTczOVowdTELMAkGA1UEBhMCVVMxEzARBgNVBAgTCmNhbGlmb3JuaWExETAPBgNV
BAcTCHNhbiBqb3NlMQ4wDAYDVQQKEwVjaXNjbzEMMAoGA1UECxMDaHNzMSAwHgYD
VQQDExdzaW1wc29uLWRldnRlc3Qtcm9vdC1DQTBcMA0GCSqGSIb3DQEBAQUAA0sA
MEgCQQCWEibAnUlVqQNUn0Wb94qnHi8FKjmVhibLHGRl6J+V7gHgzmF2MTz5WP5l
VQ2/1NVu0HjUORRdeCm1/raKJ/7ZAgMBAAGjgewwgekwCwYDVR0PBAQDAgHGMA8G
A1UdEwEB/wQFMAMBAf8wHQYDVR0OBBYEFCYGLUBTKNd9EgUonHnoSvbHg0axMIGX
BgNVHR8EgY8wgYwwQ6BBoD+GPWh0dHA6Ly9jaXNjby1sOGo2b2hwbnIvQ2VydEVu
cm9sbC9zaW1wc29uLWRldnRlc3Qtcm9vdC1DQS5jcmwwRaBDoEGGP2ZpbGU6Ly9c
XGNpc2NvLWw4ajZvaHBuclxDZXJ0RW5yb2xsXHNpbXBzb24tZGV2dGVzdC1yb290
LUNBLmNybDAQBgkrBgEEAYI3FQEEAwIBADANBgkqhkiG9w0BAQUFAANBACBqe1wy
YjalelGZqLVu4bDVMFo6ELCV2AMBgi41K3ix+Z/03PJd7ct2BIAF4lktv9pCe6IO
EoBcmZteA+TQcKg=
-----END CERTIFICATE-----

Certificate has the following attributes:
Fingerprint:AC6FC55E CC29E891 0DC3FAAA B4747C10
% Do you accept this certificate? [yes/no]:yes
Trustpoint CA certificate accepted.
% Certificate successfully imported

ssl-proxy(config)# ssl-proxy context s1
ssl-proxy(config-context)# pool ca rootca
ssl-proxy(config-ctx-ca-pool)# ca trustpoint rootca
ssl-proxy(config-ctx-ca-pool)# exit
ssl-proxy(config-context)# service client-auth-verify-all
ssl-proxy(config-ctx-ssl-proxy)# virtual ipaddr 14.0.0.2 protocol tcp port 443
ssl-proxy(config-ctx-ssl-proxy)# server ipaddr 24.0.0.1 protocol tcp port 80
ssl-proxy(config-ctx-ssl-proxy)# certificate rsa general-purpose trustpoint test-cert
ssl-proxy(config-ctx-ssl-proxy)# trusted-ca rootca
ssl-proxy(config-ctx-ssl-proxy)# authenticate verify all
ssl-proxy(config-ctx-ssl-proxy)# inservice
ssl-proxy(config-ctx-ssl-proxy)# exit
```

Server Certificate Authentication

When you configure the SSL Services Module as an SSL client (for example, for backend encryption), the SSL Services Module always authenticates the SSL server.

To authenticate the SSL server, the SSL Services Module verifies the following:

- The certificate at one level is properly signed by the issuer at the next level.
- At least one of the issuer certificates in the certificate chain is trusted by the SSL proxy service.
- None of the certificates in the certificate chain is in the certificate revocation list (CRL) and rejected by any access control list (ACL).

By default, the SSL Services Module accepts any certificate issued by any certificate authority trustpoint that has a certificate, is not listed in the CRL, and is not rejected by an ACL.

Optionally, you can create a trusted certificate authority pool and associate it with the proxy service. A In this case, the SSL Services Module accepts only certificates issued by the certificate authorities in the pool.

You can also select to verify only the signature by entering the **authenticate verify signature-only** command. Verifying the signature skips CRL and ACL checking. You must configure a trusted certificate authorities pool in order to specify the signature-only option.

To configure server certificate authentication, perform this task:

	Command	Purpose
Step 1	`ssl-proxy(config)# ssl-proxy context` *name*	Enters the SSL context subcommand mode. The optional SSL context name is used to specify an SSL virtualization instance.
Step 2	`ssl-proxy(config-context)# pool ca` *ca_pool_name*	Creates the certificate authority pool. **Note** You can create up to eight pools.
Step 3	`ssl-proxy(config-ctx-ca-pool)# ca trustpoint` *ca_trustpoint_label*[1]	Adds a trusted certificate authority to the pool. **Note** You can add up to 16 trusted certificate authorities per pool.
Step 4	`ssl-proxy(config-ctx-ca-pool)# exit`	Returns to config mode.
Step 5	`ssl-proxy(config-context)# service` *proxy_name* `client`	Defines the name of the SSL server proxy service.
Step 6	`ssl-proxy(config-ctx-ssl-proxy)# trusted-ca` *ca_pool_name*	Associates the trusted certificate authority pool with the proxy service. **Note** If you specify **signature-only** in Step 7, you must configure a certificate authority pool.

	Command	Purpose
Step 7	`ssl-proxy(config-ctx-ssl-proxy)#` **`authenticate verify`** {**`signature-only`**[2] | **`all`**[3]}	Enables client certificate authentication and specifies the form of verification. **Note** The **signature-only** keyword verifies the signature only, without looking up the CRL or matching ACL. The default is **all**.
Step 8	`ssl-proxy(config-ctx-ssl-proxy)#` **`exit`**	Exits from configuration mode.

1. The SSL Services Module supports up to eight levels of certificate authority. We recommend that you add all levels of certificate authority, or at least the root certificate authority, to the certificate authority pool.
2. When you verify signature-only, authentication stops at the level corresponding to one of the trusted certificate authority trustpoints in the trusted certificate authority pool.
3. When you verify all, the highest level issuer in the certificate chain must be configured as a trusted certificate authority trustpoint. The SSL Services Module authenticates all the certificates in the peer certificate chain and stops only at the highest level certificate authority. There must be a certificate authority trustpoint for the highest level certificate authority, and this trustpoint should be authenticated.

This example shows how to configure server certificate authentication verifying signature only:

```
ssl-proxy(config)# crypto pki trustpoint rootca
ssl-proxy(ca-trustpoint)# enrollment terminal
ssl-proxy(ca-trustpoint)# exit
ssl-proxy(config)#
ssl-proxy(config)# crypto pki authenticate rootca

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

-----BEGIN CERTIFICATE-----
MIIC1zCCAoGgAwIBAgIQadUxzU/i97hDmZRYJ1bBcDANBgkqhkiG9w0BAQUFADB1
MQswCQYDVQQGEwJVUzETMBEGA1UECBMKY2FsaWZvcm5pYTERMA8GA1UEBxMIc2Fu
IGpvc2UxDjAMBgNVBAoTBWNpc2NvMQwwCgYDVQQLEwNoc3MxIDAeBgNVBAMTF3Np
bXBzb24tZGV2dGVzdC1yb290LUNBMB4XDTAzMTExMTIxNDgwMloXDTEzMTExMTIx
NTczOVowdTELMAkGA1UEBhMCVVMxEzARBgNVBAgTCmNhbGlmb3JuaWExETAPBgNV
BAcTCHNhbiBqb3NlMQ4wDAYDVQQKEwVjaXNjbzEMMAoGA1UECxMDaHNzMSAwHgYD
VQQDExdzaW1wc29uLWRldnRlc3Qtcm9vdC1DQTBcMA0GCSqGSIb3DQEBAQUAA0sA
MEgCQQCWEibAnUlVqQNUn0Wb94qnHi8FKjmVhibLHGRl6J+V7gHgzmF2MTz5WP5l
VQ2/1NVu0HjUORRdeCm1/raKJ/7ZAgMBAAGjgewwgekwCwYDVR0PBAQDAgHGMA8G
A1UdEwEB/wQFMAMBAf8wHQYDVR0OBBYEFCYGLUBTKNd9EgUonHnoSvbHg0axMIGX
BgNVHR8EgY8wgYwwQ6BBoD+GPWh0dHA6Ly9jaXNjby1sOGo2b2hwbnIvQ2VydEVu
cm9sbC9zaW1wc29uLWRldnRlc3Qtcm9vdC1DQS5jcmwwRaBDoEGGP2ZpbGU6Ly9c
XGNpc2NvLWw4ajZvaHBuclxDZXJ0RW5yb2xsXHNpbXBzb24tZGV2dGVzdC1yb290
LUNBLmNybDAQBgkrBgEEAYI3FQEEAwIBADANBgkqhkiG9w0BAQUFAANBACBqe1wy
YjalelGZqLVu4bDVMFo6ELCV2AMBgi41K3ix+Z/03PJd7ct2BIAF4lktv9pCe6IO
EoBcmZteA+TQcKg=
-----END CERTIFICATE-----

Certificate has the following attributes:
Fingerprint:AC6FC55E CC29E891 0DC3FAAA B4747C10
% Do you accept this certificate? [yes/no]:yes
Trustpoint CA certificate accepted.
% Certificate successfully imported

ssl-proxy(config)# ssl-proxy context s1
ssl-proxy(config-context)# pool ca rootca
ssl-proxy(config-ctx-ca-pool)# ca trustpoint rootca
ssl-proxy(config-ctx-ca-pool)# exit
ssl-proxy(config-context)# service client-proxy-sig-only client
ssl-proxy(config-ctx-ssl-proxy)# virtual ipaddr 14.0.0.3 protocol tcp port 81
```

```
ssl-proxy(config-ctx-ssl-proxy)# server ipaddr 24.0.0.2 protocol tcp port 443
ssl-proxy(config-ctx-ssl-proxy)# trusted-ca rootca
ssl-proxy(config-ctx-ssl-proxy)# authenticate verify signature-only
ssl-proxy(config-ctx-ssl-proxy)# inservice
ssl-proxy(config-ctx-ssl-proxy)# exit
ssl-proxy(config-context)#
```

This example shows how to configure server certificate authentication verifying all:

```
ssl-proxy(config)# crypto pki trustpoint rootca
ssl-proxy(ca-trustpoint)# enrollment terminal
ssl-proxy(ca-trustpoint)# exit
ssl-proxy(config)#
ssl-proxy(config)# crypto pki authenticate rootca

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

-----BEGIN CERTIFICATE-----
MIIC1zCCAoGgAwIBAgIQadUxzU/i97hDmZRYJ1bBcDANBgkqhkiG9w0BAQUFADB1
MQswCQYDVQQGEwJVUzETMBEGA1UECBMKY2FsaWZvcm5pYTERMA8GA1UEBxMIc2Fu
IGpvc2UxDjAMBgNVBAoTBWNpc2NvMQwwCgYDVQQLEwNoc3MxIDAeBgNVBAMTF3Np
bXBzb24tZGV2dGVzdC1yb290LUNBMB4XDTAzMTExMTIxNDgwMloXDTEzMTExMTIx
NTczOVowdTELMAkGA1UEBhMCVVMxEzARBgNVBAgTCmNhbGlmb3JuaWExETAPBgNV
BAcTCHNhbiBqb3NlMQ4wDAYDVQQKEwVjaXNjbzEMMAoGA1UECxMDaHNzMSAwHgYD
VQQDExdzaW1wc29uLWRldnRlc3Qtcm9vdC1DQTBcMA0GCSqGSIb3DQEBAQUAA0sA
MEgCQQCWEibAnUlVqQNUn0Wb94qnHi8FKjmVhibLHGRl6J+V7gHgzmF2MTz5WP5l
VQ2/1NVu0HjUORRdeCm1/raKJ/7ZAgMBAAGjgewwgekwCwYDVR0PBAQDAgHGMA8G
A1UdEwEB/wQFMAMBAf8wHQYDVR0OBBYEFCYGLUBTKNd9EgUonHnoSvbHg0axMIGX
BgNVHR8EgY8wgYwwQ6BBoD+GPWh0dHA6Ly9jaXNjby1sOGo2b2hwbnIvQ2VydEVu
cm9sbC9zaW1wc29uLWRldnRlc3Qtcm9vdC1DQS5jcmwwRaBDoEGGP2ZpbGU6Ly9c
XGNpc2NvLWw4ajZvaHBuclxDZXJ0RW5yb2xsXHNpbXBzb24tZGV2dGVzdC1yb290
LUNBLmNybDAQBgkrBgEEAYI3FQEEAwIBADANBgkqhkiG9w0BAQUFAANBACBqe1wy
YjalelGZqLVu4bDVMFo6ELCV2AMBgi41K3ix+Z/03PJd7ct2BIAF4lktv9pCe6IO
EoBcmZteA+TQcKg=
-----END CERTIFICATE-----

Certificate has the following attributes:
Fingerprint:AC6FC55E CC29E891 0DC3FAAA B4747C10
% Do you accept this certificate? [yes/no]:yes
Trustpoint CA certificate accepted.
% Certificate successfully imported

ssl-proxy(config)# ssl-proxy context s1
ssl-proxy(config-context)# pool ca rootca
ssl-proxy(config-ctx-ca-pool)# ca trustpoint rootca
ssl-proxy(config-ctx-ca-pool)# exit
ssl-proxy(config-context)# service client-proxy-verify-all client
ssl-proxy(config-ctx-ssl-proxy)# virtual ipaddr 14.0.0.4 protocol tcp port 81
ssl-proxy(config-ctx-ssl-proxy)# server ipaddr 24.0.0.2 protocol tcp port 443
ssl-proxy(config-ctx-ssl-proxy)# trusted-ca rootca
ssl-proxy(config-ctx-ssl-proxy)# authenticate verify all
ssl-proxy(config-ctx-ssl-proxy)# inservice
ssl-proxy(config-ctx-ssl-proxy)# exit
ssl-proxy(config-context)#
```

This example shows how to configure server certificate authentication verifying all; the SSL Services Module is configured as a client and sends its certificate to the SSL server if it is requested.

```
ssl-proxy(config)# crypto pki trustpoint rootca
ssl-proxy(ca-trustpoint)# enrollment terminal
ssl-proxy(ca-trustpoint)# revocation-check none
ssl-proxy(ca-trustpoint)# exit
ssl-proxy(config)#
```

```
ssl-proxy(config)# crypto pki authenticate rootca

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

-----BEGIN CERTIFICATE-----
MIIC1zCCAoGgAwIBAgIQadUxzU/i97hDmZRYJ1bBcDANBgkqhkiG9w0BAQUFADB1
MQswCQYDVQQGEwJVUzETMBEGA1UECBMKY2FsaWZvcm5pYTERMA8GA1UEBxMIc2Fu
IGpvc2UxDjAMBgNVBAoTBWNpc2NvMQwwCgYDVQQLEwNoc3MxIDAeBgNVBAMTF3Np
bXBzb24tZGV2dGVzdC1yb290LUNBMB4XDTAzMTExMTIxNDgwMloXDTEzMTExMTIx
NTczOVowdTELMAkGA1UEBhMCVVMxEzARBgNVBAgTCmNhbGlmb3JuaWExETAPBgNV
BAcTCHNhbiBqb3NlMQ4wDAYDVQQKEwVjaXNjbzEMMAoGA1UECxMDaHNzMSAwHgYD
VQQDExdzaW1wc29uLWRldnRlc3Qtcm9vdC1DQTBcMA0GCSqGSIb3DQEBAQUAA0sA
MEgCQQCWEibAnUlVqQNUn0Wb94qnHi8FKjmVhibLHGR16J+V7gHgzmF2MTz5WP5l
VQ2/1NVu0HjUORRdeCm1/raKJ/7ZAgMBAAGjgewwgekwCwYDVR0PBAQDAgHGMA8G
A1UdEwEB/wQFMAMBAf8wHQYDVR0OBBYEFCYGLUBTKNd9EgUonHnoSvbHg0axMIGX
BgNVHR8EgY8wgYwwQ6BBoD+GPWh0dHA6Ly9jaXNjby1sOGo2b2hwbnIvQ2VydEVu
cm9sbC9zaW1wc29uLWRldnRlc3Qtcm9vdC1DQS5jcmwwRaBDoEGGP2ZpbGU6Ly9c
XGNpc2NvLWw4ajZvaHBuclxDZXJ0RW5yb2xsXHNpbXBzb24tZGV2dGVzdC1yb290
LUNBLmNybDAQBgkrBgEEAYI3FQEEAwIBADANBgkqhkiG9w0BAQUFAANBACBqe1wy
YjalelGZqLVu4bDVMFo6ELCV2AMBgi41K3ix+Z/03PJd7ct2BIAF4lktv9pCe6IO
EoBcmZteA+TQcKg=
-----END CERTIFICATE-----

Certificate has the following attributes:
Fingerprint:AC6FC55E CC29E891 0DC3FAAA B4747C10
% Do you accept this certificate? [yes/no]:yes
Trustpoint CA certificate accepted.
% Certificate successfully imported

ssl-proxy(config)# ssl-proxy context s1
ssl-proxy(config-context)# pool ca rootca
ssl-proxy(config-ctx-ca-pool)# ca trustpoint rootca
ssl-proxy(config-ctx-ca-pool)# exit
ssl-proxy(config-context)# service client-sending-client-cert client
ssl-proxy(config-ctx-ssl-proxy)# virtual ipaddr 14.0.0.5 protocol tcp port 81
ssl-proxy(config-ctx-ssl-proxy)# server ipaddr 24.0.0.3 protocol tcp port 443
ssl-proxy(config-ctx-ssl-proxy)# trusted-ca rootca
ssl-proxy(config-ctx-ssl-proxy)# authenticate verify all
ssl-proxy(config-ctx-ssl-proxy)# inservice
ssl-proxy(config-ctx-ssl-proxy)# exit
ssl-proxy(config-context)#
```

Checking the Certificate Status

There are two methods to check certificate status: Online Certificate Status Protocol (OCSP) and certificate revocation list (CRL).

The following sections give overview and configuration tasks for both methods.

- Certificate Revocation List, page 3-59
- Online Certificate Status Protocol, page 3-60

Certificate Revocation List

A certificate revocation list (CRL) is a time-stamped list that identifies certificates that should no longer be trusted. Each revoked certificate is identified in a CRL by its certificate serial number. When a participating peer device uses a certificate, that device not only checks the certificate signature and validity but also checks that the certificate serial number is not on that CRL.

Note Downloading and using CRLs are time-consuming and CPU-intensive operations.

If the certificate being validated contains a CRL distribution point (CDP) extension field, the module uses the CDP as the download path. The SSL Services Module supports three types of CDPs:

- HTTP URL

  For example, http://hostname/file.crl

- X.500 distinguished name (DN)

  For example, CN=CRL,O=cisco,C=us

- Lightweight Directory Access Protocol (LDAP) URL

  For example, ldap://hostname/CN=CRL,O=cisco,C=us

If the certificate does not have a CDP, the SSL Services Module looks for a trustpoint that is associated with the certificate. If the module finds one or more associated trustpoints, the module uses the first configured trustpoint to determine the download path and protocol using the SCEP enrollment URL. If there is no SCEP enrollment URL, the validation fails.

Note When the configuration contains multiple trustpoints using one common certificate chain, the CRL download information of only the first listed trustpoint is used. If this first trustpoint is not configured with a SCEP enrollment URL, the validation fails, regardless of the configuration of the other trustpoints.

Note When using SCEP to request for a CRL, you need to associate a key pair with the trustpoint. You can assign any existing key pair for this purpose. The key pair is used for signing the CRL download request.

The SSL Services Module does not perform a CRL lookup on the root certificate authority certificates because of the following reasons:

- Many root certificate authority certificates do not contain a CDP extension. If the certificate authority does not support SCEP for CRL request, the CRL download fails.
- CRLs are signed by the root certificate authority. If the root certificate authority has been revoked, its CRL will probably become invalid. All trustpoints associated with a revoked root certificate authority should be deleted from the database as soon as the revocation is made known.

If the download path is not known, or if the download operation fails, the peer certificate chain is rejected.

After the module downloads the CRL, the module checks to see if the serial number of the certificate appears on the CRL. If the serial number of the certificate being validated appears on the CRL, the peer certificate chain is rejected.

Note One or more certificates in the peer certificate chain can fail the CRL lookup, even though some of the certificate authority certificates are trusted (for example, a certificate authority certificate was revoked after it was imported, or the CRL that was downloaded for this certificate authority certificate has expired and the attempt to download a updated CRL has failed).

Online Certificate Status Protocol

The Online Certificate Status Protocol (OCSP) feature allows you to enable OCSP instead of certificate revocation lists (CRLs) to check certificate status. Unlike CRLs, which provide only periodic certificate status, OCSP can provide timely information regarding the status of a certificate.

OCSP allows a network administrator to configure a central OCSP server to collect and update CRLs from different certification authority (CA) servers; therefore, the devices within the network can rely on the OCSP server to check the certificate status without retrieving and caching each CRL for every device.

An OCSP server usually operates in either push or poll mode. You can configure a CA server to push revocation information to an OCSP server or configure an OCSP server to periodically download (poll) a CRL from the CA server. To ensure that timely certificate revocation status is obtained, you should carefully consider the "push and poll" interval.

When configuring an OCSP server to return the revocation status for a CA server, the OCSP server must be configured with an OCSP response signing certificate that is issued by that CA server. Ensure that the signing certificate is in the correct format, or the router will not accept the OCSP response. Refer to your OCSP manual for additional information.

The following is a sample OCSP response signing certificate (the extensions are in bold):

```
Certificate:
        Data:
            Version: v3
            Serial Number:0x14
            Signature Algorithm:MD5withRSA - 1.2.840.113549.1.1.4
            Issuer:CN=CA server,OU=PKI,O=Cisco Systems
            Validity:
                Not Before:Thursday, August 8, 2002 4:38:05 PM PST
                Not  After:Tuesday, August 7, 2003 4:38:05 PM PST
            Subject:CN=OCSP server,OU=PKI,O=Cisco Systems
            Subject Public Key Info:
                Algorithm:RSA - 1.2.840.113549.1.1.1
                Public Key:
                    Exponent:65537
                    Public Key Modulus:(1024 bits) :
                        <snip>

            Extensions:
                Identifier:Subject Key Identifier - 2.5.29.14
                    Critical:no
                    Key Identifier:
                        <snip>

                Identifier:Authority Key Identifier - 2.5.29.35
                    Critical:no
                    Key Identifier:
                        <snip>

                Identifier:OCSP NoCheck:- 1.3.6.1.5.5.7.48.1.5
                     Critical:no
                Identifier:Extended Key Usage:- 2.5.29.37
                     Critical:no
```

```
                    Extended Key Usage:
                    OCSPSigning
                Identifier:CRL Distribution Points - 2.5.29.31
                    Critical:no
                    Number of Points:1
                    Point 0
                        Distribution Point:
[URIName:ldap://CA-server/CN=CA server,OU=PKI,O=Cisco Systems]
        Signature:
            Algorithm:MD5withRSA - 1.2.840.113549.1.1.4
            Signature:
            <snip>
```

The following is a sample certificate that has OCSP URL defined AIA extension (the extensions are in bold):

```
Certificate:
        Data:
            Version:  v3
            Serial Number: 0xBB1
            Signature Algorithm: SHA1withRSA - 1.2.840.113549.1.1.5
            Issuer: CN=CA server,OU=PKI,O=Cisco Systems
            Validity:
                Not Before: Friday, February 21, 2003 6:41:07 PM PST America/Los_Angeles
                Not  After: Sunday, March 23, 2003 6:41:07 PM PST America/Los_Angeles
            Subject: CN=Client,OU=PKI,O=Cisco Systems
            Subject Public Key Info:
                Algorithm: RSA - 1.2.840.113549.1.1.1
                Public Key:
                    Exponent: 65537
                    Public Key Modulus: (512 bits) :
                    <snip>
            Extensions:
                Identifier: Key Usage: - 2.5.29.15
                    Critical: no
                    Key Usage:
                        Key Encipherment
                Identifier: Authority Key Identifier - 2.5.29.35
                    Critical: no
                    Key Identifier:
                    <snip>
                Identifier: CRL Distribution Points - 2.5.29.31
                    Critical: no
                    Number of Points: 1
                    Point 0
                        Distribution Point: [URIName: ldap://crl.cisco.com/CN=CA server,
                         OU=PKI,O=Cisco Systems]
                Identifier: Authority Info Access: - 1.3.6.1.5.5.7.1.1
                    Critical: no
                    Access Description:
                        Method #0: ocsp
                        Location #0: URIName: http://ocsp.cisco.com:81
        Signature:
            Algorithm: SHA1withRSA - 1.2.840.113549.1.1.5
            Signature:
            <snip>
```

Observe the following guidelines for OCSP:

- OCSP transports messages over HTTP, so there may be a time delay when you access the OCSP server. If the OCSP server is unavailable, certificate verification will fail.

- The increased certificate size may cause a problem for low-end routers when certificates are stored on NVRAM. Therefore, before you add the Authority Info Access (AIA) extension to a certificate, make sure that the increased size will not cause deployment problems.

Configuring the Revocation Check Method

To configure the SSL Services Module to check certificate status, perform this task:

	Command	Purpose
Step 1	`ssl-proxy(config)# crypto pki trustpoint` *trustpoint_label*	Declares the trustpoint that your module should use. Enabling this command puts you in ca-trustpoint configuration mode.
Step 2	`ssl-proxy(ca-trustpoint)#` **revocation-check** *method1* [*method2*[*method3*]]	Checks the revocation status of a certificate. • **crl**—Certificate checking is performed by a CRL. This is the default option. • **none**—Certificate checking is ignored. • **ocsp**—Certificate checking is performed by an OCSP server. **Note** If a second and third method are specified, each method is used only if the previous method returns an error, such as a server being down. **Note** Since certificate checking is ignored when you enter the **none option**, you can only configure it at the end of the list.
Step 3	`ssl-proxy(ca-trustpoint)# ocsp url` *url*	(Optional) Specifies the URL of an OCSP server so that the trustpoint can check the certificate status. This URL overrides the URL of the OCSP server (if one exists) in Authority Info Access (AIA) extension of the certificate.

The following example shows how to configure the SSL Services Module to use the OCSP server at the HTTP URL "http://myocspserver:81." If the server is down, revocation check is ignored.

```
ssl-proxy(config)# crypto pki trustpoint mytp
ssl-proxy(ca-trustpoint)# ocsp url http://myocspserver:81
ssl-proxy(ca-trustpoint)# revocation-check ocsp none
```

This following sections describe how to download and configure CRLs:

- Updating a CRL, page 3-63
- Entering X.500 CDP Information, page 3-63
- Entering a CRL Manually, page 3-64
- Displaying CRL Information, page 3-65
- Deleting a CRL, page 3-65

Updating a CRL

A CRL can be reused with subsequent certificates until the CRL expires.

If the specified NextUpdate time of the CRL is reached, the CRL is deleted. Enter the **show crypto pki timers** command to display the time remaining for the CRL.

If a CRL has not expired yet, but you suspect that the contents of the CRL are out of date, you can download the latest CRL immediately to replace the old CRL.

To request immediate download of the latest CRL, enter the following command:

Command	Purpose
`ssl-proxy(config)# crypto pki crl request` *trustpoint_label*	Requests an updated CRL. This command replaces the currently stored CRL with the newest version of the CRL.

Note Downloading a new CRL overwrites any existing version.

The following example shows how to download the CRL associated with the trustpoint "tp-root:"

```
ssl-proxy(config)# crypto pki crl request tp-root
```

Entering X.500 CDP Information

You can enter the LDAP URL if the CDP is in X.500 DN format. The query takes the information in the following form: **ldap://***hostname*:[*port*]

For example, if a certficate being validated has the following:

- the X.500 DN is configured with **CN=CRL,O=Cisco,C=US**
- the associated trustpoint is configured with **crl query ldap://10.1.1.1**

then the two parts are combined to form the complete URL as follows:

ldap://10.1.1.1/CN=CRL,O=Cisco,C=US

Note Note that the trustpoint should be associated with the issuer certificate authority certificate of the certificate being validated. If there is no such trustpoint in the database, the complete URL cannot be formed, and CRL download cannot be performed.

Note The CRL query must be sent on the administrative VLAN. To configure the SSL VLAN, refer to Adding the SSL Services Module to the Corresponding VLAN, page 2-8.

To query the CRL with the X.500 URL, enter this command:

Command	Purpose
`ssl-proxy(ca-trustpoint)# crl query` *urk*	Allows you to specify the LDAP URL of the CA server that is used to query the CRL.

Entering a CRL Manually

If the certificate authority server does not publish the CRL online (through HTTP, LDAP, or SCEP), you can get a hexdump of the CRL offline and enter it manually.

To manually enter the CRL, perform this task:

	Command	Purpose
Step 1	`ssl-proxy(config)# crypto pki certificate chain` *name*	Enters certificate chain configuration mode; *name* specifies the name of the certificate authority.
Step 2	`ssl-proxy(config-cert-chain)# crl`	Allows you to enter the revocation list issued by the certificate authority manually.
Step 3	`ssl-proxy(config-pubkey)# quit`	Exits data entry mode.
Step 4	`ssl-proxy(config-cert-chain)# end`	Exits certificate chain configuration mode.

The following example shows how to manually enter a CRL:

```
ssl-proxy(config)# crypto pki certificate chain tp
ssl-proxy(config-cert-chain)# crl
Enter the CRL in hexidecimal representation....
ssl-proxy(config-pubkey)#     30 82 01 7E 30 81 E8 30 0D 06 09 2A 86 48 86 F7
ssl-proxy(config-pubkey)#     0D 01 01 05 05 00 30 16 31 14 30 12 06 03 55 04
ssl-proxy(config-pubkey)#     03 13 0B 69 6F 73 2D 72 6F 6F 74 20 43 41 17 0D
ssl-proxy(config-pubkey)#     30 33 31 32 31 31 32 33 33 37 35 34 5A 17 0D 30
ssl-proxy(config-pubkey)#     33 31 32 31 38 32 33 33 37 35 34 5A 30 81 A0 30
ssl-proxy(config-pubkey)#     12 02 01 04 17 0D 30 33 31 30 31 34 32 32 32 35
ssl-proxy(config-pubkey)#     30 34 5A 30 12 02 01 03 17 0D 30 33 31 30 31 34
ssl-proxy(config-pubkey)#     32 32 32 35 33 30 5A 30 12 02 01 05 17 0D 30 33
ssl-proxy(config-pubkey)#     31 31 31 33 31 39 30 39 33 36 5A 30 12 02 01 06
ssl-proxy(config-pubkey)#     17 0D 30 33 31 31 31 33 31 39 32 30 34 32 5A 30
ssl-proxy(config-pubkey)#     12 02 01 07 17 0D 30 33 31 31 31 33 32 32 30 35
ssl-proxy(config-pubkey)#     35 32 5A 30 12 02 01 08 17 0D 30 33 31 31 31 33
ssl-proxy(config-pubkey)#     32 32 34 34 32 30 5A 30 12 02 01 2B 17 0D 30 33
ssl-proxy(config-pubkey)#     31 31 31 33 32 33 33 36 35 37 5A 30 12 02 01 09
ssl-proxy(config-pubkey)#     17 0D 30 33 31 31 31 33 32 33 33 37 34 38 5A 30
ssl-proxy(config-pubkey)#     0D 06 09 2A 86 48 86 F7 0D 01 01 05 05 00 03 81
ssl-proxy(config-pubkey)#     81 00 67 DE 12 99 9F C5 DF 4A F8 24 76 CE 98 4F
ssl-proxy(config-pubkey)#     7C 5C 72 1C E0 00 A9 CE 08 6E 46 8F 4D 1B FA 8E
ssl-proxy(config-pubkey)#     C9 DE CF AC 13 7D 2F BF D4 A6 C2 7B E2 31 B1 EC
ssl-proxy(config-pubkey)#     99 83 54 B3 11 24 6F C3 C3 93 C4 53 38 B6 72 86
ssl-proxy(config-pubkey)#     0A 30 F2 95 71 AE 15 66 87 3E C1 7F 8B 46 6F A9
ssl-proxy(config-pubkey)#     77 D0 FF D4 FC 73 83 79 98 BD 40 DB C1 72 9D 95
ssl-proxy(config-pubkey)#     9B 57 D1 3C 2F EF B6 63 6B 5B E4 35 40 52 2D 3A
ssl-proxy(config-pubkey)#     19 1A 4E CA 70 C6 ED 49 7A 7C 01 88 B9 CA 14 7B
ssl-proxy(config-pubkey)#     0E 1F
ssl-proxy(config-pubkey)# quit
ssl-proxy(config-cert-chain)# end
```

Displaying CRL Information

To display information about the CRLs, enter this command:

Command	Purpose
`ssl-proxy# show crypto pki crls`	Displays the expiration date and time of each CRL in the database.

This example shows the expiration date and time of each CRL in the database:

```
ssl-proxy# show crypto pki crls
CRL Issuer Name:
    CN = test-root-CA, OU = lab, O = cisco, L = san jose, ST = california, C = US
    LastUpdate:19:08:45 UTC Dec 3 2003
    NextUpdate:20:13:45 UTC Dec 3 2003
    Retrieved from CRL Distribution Point:
      http://test-ca/CertEnroll/test-root-CA.crl
```

Deleting a CRL

Note CRLs are deleted globally and not per trustpoint.

To delete a CRL, enter the **no crl** command for any certificate chain of a trustpoint.

This example shows how to delete the CRL:

```
ssl-proxy(config)# crypto pki certificate chain tp1
ssl-proxy(config-cert-chain)# no crl
ssl-proxy(config-cert-chain)# end
```

Certificate Security Attribute-Based Access Control

The Certificate Security Attribute-Based Access Control feature adds fields to the certificate that allow specifying an access control list (ACL), to create a certificate-based ACL.

For information on configuring certificate security attribute-based access control, refer to *Certificate Security Attribute-Based Access Control* at this URL:

http://www.cisco.com/univercd/cc/td/doc/product/software/ios122/122newft/122t/122t15/ftcrtacl.htm

Advanced Configurations for the SSL Services Module

This chapter describes the following advanced configurations:

Configuring Policies

See the "Configuring SSL Proxy Services" section on page 3-45 for procedures for applying policies to a proxy service.

This section describes how to configure SSL and TCP policies:

Configuring SSL Policy

Note The SSL commands for the SSL Services Module apply either globally or to a particular proxy server.

The SSL policy template allows you to define parameters associated with the SSL stack.

One of the parameters you can configure is the SSL close-protocol behavior. The SSL close-protocol specifies that each of the SSL peers (client and server) should send a close-notify alert and receive a close-notify alert before closing the connection properly. If the SSL connection is not closed properly, the session is removed so that the peers cannot use same SSL session ID in future SSL connections.

However, many SSL implementations do not follow the SSL close-protocol strictly (for example, an SSL peer sends a close-notify alert but does not wait for the close-notify alert from the remote SSL peer before closing the connection).

When an SSL peer initiates the close connection sequence, the SSL Services Module strictly expects a close-notify alert message. If an SSL peer does not send a close-notify alert, SSL Services Module removes the session from the session cache so that the same session-id cannot be used for future ssl connections.

When the SSL Services Module initiates the close connection sequence, you can configure the following close-protocol options:

- **strict—**The SSL Services Module sends a close-notify alert message to the SSL peer, and the SSL Services Module expects a close-notify alert message from the SSL peer. If the SSL Services Module does not receive a close-notify alert, SSL resumption is not allowed for that session.
- **none**—The SSL Services Module does not send a close-notify alert message to the SSL peer, nor does the SSL Services Module expect a close-notify alert message from the SSL peer. The SSL Services Module preserves the session information so that SSL resumption can be used for future SSL connections.
- disabled (default)—The SSL Services Module sends a close-notify alert message to the SSL peer; however, the SSL peer does not expect a close-notify alert before removing the session. Whether SSL peer sends close-notify or not, the session information is preserved allowing session resumption for future SSL connections.

If you do not associate an SSL policy with a particular proxy server, the proxy server enables all the supported cipher suites and protocol versions by default.

To define an SSL policy template and associate an SSL policy with a particular proxy server, perform this task:

	Command	Purpose
Step 1	`ssl-proxy(config)# ssl-proxy context` *name*	Enters the SSL context subcommand mode. The optional SSL context name name is used to specify an SSL virtualization instance.
Step 2	`ssl-proxy(config-context)# policy ssl` *ssl_policy_name*	Defines SSL policy templates.

	Command	Purpose
Step 3	`ssl-proxy(config-ctx-ssl-policy)#` **cipher** *ciphersuite*	Configures a list of cipher-suite names acceptable to the proxy server. The cipher-suite names follow the same convention as that of existing SSL stacks. The default setting is **all-strong**. Valid values are as follows: • **all** • **all-export** • **all-strong** (default) • **rsa-exp-with-des40-cbc-sha** • **rsa-exp-with-rc4-40-md5** • **rsa-exp1024-with-des-cbc-sha** • **rsa-exp1024-with-rc4-56-md5** • **rsa-exp1024-with-rc4-56-sha** • **rsa-with-3des-ede-cbc-sha** • **rsa-with-des-cbc-sha** • **rsa-with-null-md5** • **rsa-with-rc4-128-md5** • **rsa-with-rc4-128-sha**
Step 4	`ssl-proxy(config-ctx-ssl-policy)#` **version** {**ssl3** | **tls1** | **all**}	Defines the various protocol versions supported by the proxy server.
Step 5	`ssl-proxy(config-ctx-ssl-policy)#` **close-protocol** {**strict** | **none**}	Configures the SSL close-protocol behavior. Close-protocol is disabled by default.
Step 6	`ssl-proxy(config-ctx-ssl-policy)#` **session-cache**	Enables the session-caching feature. Session caching is enabled by default.
Step 7	`ssl-proxy(config-ctx-ssl-policy)#` **timeout handshake** *time*	Configures how long the module keeps the connection in handshake phase. The valid range is 0 to 65535 seconds.
Step 8	`ssl-proxy(config-ctx-ssl-policy)#` **timeout session** *timeout* [**absolute**[1]]	Configures the amount of time that an entry is kept in the session cache. The valid range is 1 to 72000 seconds. **Note** The **absolute** keyword is required in order to configure session-cache size. **Note** The **absolute** keyword specifies that the session entry is kept in the session cache for the specified *timeout*. When the **absolute** keyword is specified, new incoming connections are rejected if there are no free entries available in the session cache.
Step 9	`ssl-proxy(config-ctx-ssl-policy)#` **session-cache size** *size*	(Optional) Specifies the size of the session cache[1]. The valid range is 1 to 262143 entries. **Note** Specify the session cache size when you enter the **absolute** keyword with the **timeout session** command. If this command is not entered or if no *size* is specified, the session cache size is the maximum size (262,144).

	Command	Purpose
Step 10	`ssl-proxy(config-ctx-ssl-policy)#` **`cert-req empty`**	Specifies that the SSL Services Module does not look for a CA-name match before returning a certificate. Enter this command if the backend SSL servers do not include CA name list in the certificate request during server authentication. **Note** By default, the SSL Services Module always looks for the CA name match before returning the certificate. If the SSL server does not include a CA-name list in the certificate request during client authentication, handshake will fail.
Step 11	`ssl-proxy(config-ctx-ssl-policy)#` **`tls-rollback`** [**`current`** | **`any`**]	Specifies the version of SSL protocol (SSL2.0, SSL3.0, TLS1.0) in the ClientHello message. TLS-rollback is disabled by default. When you configure the **current** keyword, the SSL protocol version can be either the maximum supported version or the negotiated version. When you configure the **any** keyword, the SSL protocol version is not checked at all. **Note** By default, the SSL Services Module uses the maximum supported version. Enter this command if the SSL client uses the negotiated version instead of the maximum supported version (as specified in the ClientHello message).
Step 12	`ssl-proxy(config-ctx-ssl-policy)#` **`renegotiation volume`** *`size`*	Enables autorenegotiation and specifies the data volume size (in kilobytes). When the encrypted or decrypted data amount exceeds this size, the SSL Services Module sends a renegotiation request. This setting is disabled by default. The valid range is from 1024 to 1073741824 kilobytes.
Step 13	`ssl-proxy(config-ctx-ssl-policy)#` **`renegotiation interval`** *`time`*	Enables autorenegotiation and specifies the interval (in seconds). After the set interval, the SSL Services Module sends an renegotiation request. This setting is disabled by default. The valid range is from 60 to 86400 seconds.
Step 14	`ssl-proxy(config-ctx-ssl-policy)#` **`renegotiation wait-time`** *`time`*	(Optional) When autorenegotiation is enabled, this command specifies the amount of time (in seconds) that the SSL Services Module waits for the peer to respond to the renegotiation request. The default is 100 seconds. The valid range is from 10 to 300 seconds.
Step 15	`ssl-proxy(config-ctx-ssl-policy)#` **`renegotiation optional`**	(Optional) When autorenegotiation is enabled, the SSL Services Module allows the session to continue if the peer does not respond to the renegotiation request after timeout. This setting is disabled by default and the session is disconnected after timeout.

	Command	Purpose
Step 16	`ssl-proxy(config-ctx-ssl-policy)# exit`	Returns to config mode.
Step 17	`ssl-proxy(config-context)# service service_name`	Defines the name of the SSL proxy service. See the "Configuring SSL Proxy Services" section on page 3-45 for more information about configuring SSL proxy services. **Note** The *service_name* value is case-sensitive.
Step 18	`ssl-proxy(config-ctx-ssl-proxy)# virtual policy ssl ssl_policy_name`	Applies the SSL policy to the client side of the proxy server.

1. When the **absolute** keyword is configured, the session entry is not reused until the configured session timeout expires. When **absolute** is configured, the number of session entries required is equal to (new_connection_rate * absolute_timeout). Depending on the timeout configuration and the new connection rate, the number of session entries might be very large. In this case, you can limit the number of session entries used by configuring the session-cache size.

Configuring TCP Policy

Note The TCP commands for the SSL Services Module apply either globally or to a particular proxy server.

The TCP policy template allows you to define parameters associated with the TCP stack.

To define an TCP policy template and associate an TCP policy with a particular proxy server, perform this task:

	Command	Purpose
Step 1	`ssl-proxy(config)# ssl-proxy context name`	Enters the SSL context subcommand mode. The optional SSL context name name is used to specify an SSL virtualization instance.
Step 2	`ssl-proxy(config-context)# policy tcp tcp_policy_name`	Defines TCP policy templates. All defaults are assumed unless otherwise specified.
Step 3	`ssl-proxy(config-ctx-tcp-policy)# mss max_segment_size`	Configures the maximum segment size (MSS), in bytes, that the connection will identify in the SYN packet that it generates. **Note** This command allows you to configure a different MSS for the client side and server side of the proxy server. The default is 1460 bytes. The valid range is from 256 to 2460 bytes[1].
Step 4	`ssl-proxy(config-ctx-tcp-policy)# timeout syn time`	Configures the connection establishment timeout. The default is 75 seconds. The valid range is from 5 to 75 seconds.
Step 5	`ssl-proxy(config-ctx-tcp-policy)# timeout reassembly time`	Configures the amount of time, in seconds, before the reassembly queue is cleared. If the transaction is not complete within the specified time, the reassembly queue is cleared and the connection is dropped. The default is 60 seconds. The valid range is 0 (disabled) to 960 seconds.

	Command	Purpose
Step 6	`ssl-proxy(config-ctx-tcp-policy)#` **`timeout inactivity`** *time*	Configures the amount of time, in seconds, that an established connection can be inactive. The default is 600 seconds. The valid range is 0 (disabled) to 7200 seconds (2 hours).
Step 7	`ssl-proxy(config-ctx-tcp-policy)#` **`timeout fin-wait`** *time*	Configures the FIN wait timeout in seconds. The default value is 600 seconds. The valid range is from 75 to 600 seconds.
Step 8	`ssl-proxy(config-ctx-tcp-policy)#` **`timeout persist`** *time*	Configures the persist wait timeout in seconds. The default value is 0 seconds. The valid range is from 5 to 7200 seconds.
Step 9	`ssl-proxy(config-ctx-tcp-policy)#` **`buffer-share rx`** *buffer_limit*	Configures the maximum receive buffer share per connection in bytes. The default value is 32768 bytes. The valid range is from 8192 to 262144 bytes.
Step 10	`ssl-proxy(config-ctx-tcp-policy)#` **`buffer-share tx`** *buffer_limit*	Configures the maximum transmit buffer share per connection in bytes. The default value is 32768 bytes. The valid range is from 8192 to 262144 bytes.
Step 11	`ssl-proxy(config-ctx-tcp-policy)#` **`forced-ack`**	Enable the forced-ACK algorithm.
Step 12	`ssl-proxy(config-ctx-tcp-policy)#` **`delayed-ack-threshold`** *delay*	Configures the delayed ACK threshold. The default is 2. The valid range is from 1 to 10.
Step 13	`ssl-proxy(config-ctx-tcp-policy)#` **`delayed-ack-timeout`** *timer*	Configures the delayed ACK timeout. The default is 200 seconds. The valid range is from 50 to 500 seconds.
Step 14	`ssl-proxy(config-ctx-tcp-policy)#` **`tos carryover`**	Forwards the type of service (ToS) value to all packets within a flow. **Note** If the policy is configured as a server TCP policy, the ToS value is sent from the server to the client. If the policy is configured as a virtual policy, the ToS value is sent from the client to the server. **Note** The ToS value needs to be learned before it can be propagated. For example, when a ToS value is configured to be propagated from the server to client connection, the server connection must be established before the value is learned and propagated. Therefore, some of the initial packets will not carry the ToS value.
Step 15	`ssl-proxy(config-ctx-tcp-policy)#` [**`no`**] **`nagle`**	Enables or disables the Nagle algorithm. Nagle is enabled by default.
Step 16	`ssl-proxy(config-ctx-tcp-policy)#` **`exit`**	Returns to config mode.
Step 17	`ssl-proxy(config-context)#` **`service`** *service_name*	Defines the name of the SSL proxy service. See the "Configuring SSL Proxy Services" section on page 3-45 for more information about configuring SSL proxy services. **Note** The *service_name* value is case-sensitive.
Step 18	`ssl-proxy(config-ctx-ssl-proxy)#` {**`virtual`**|**`server`**} **`policy tcp`** *tcp_policy_name*	Applies the TCP policy to the client (virtual) side or the server side of the proxy server.

1. If fragmentation occurs, decrease the MSS value until there is no fragmentation.

Note When large encrypted files are transferred by the module, the transmit and receive buffer sizes must be at least the maximum SSL record size of 16384 bytes for reassembly of the SSL record. For the **buffer-share rx** and **buffer-share tx** commands, we recommend transmit and receive share sizes of at least 20000 bytes for optimal performance.

Examples of situations requiring this setting are:
- When a service is configured for backend server encryption and the server sends large files.
- When the client-side is encrypted and a client posts large files to the server.

HTTP Header Insertion

In a typical SSL offloading environment, an SSL offloader terminates secure client HTTP (HTTPS) connections, decrypts the SSL traffic into cleartext, and forwards the cleartext to a Web server through an HTTP connection. The HTTPS connections become non-secure HTTP connections at the backend server. The backend server does not know that the client connection came in as a secure connection.

Here are a few reasons to configure HTTP header insertion:

- HTTP header insertion allows the SSL Services Module to embed information into an HTTP header during a client connection. When the backend server recognizes this header, the server returns all the URLs as HTTPS.
- You can have a backend application that logs information per connection by configuring an SSL offloader to insert the client certificate information into the HTTP header received from the client.
- When you use the SSL Services Module in a site-to-site configuration to send traffic over a secured channel, the server end of the connection may need to know the client IP address and port information, which gets removed during NAT.

HTTP header insertion is performed for the following methods: GET, HEAD, PUT, TRACE, POST, DELETE. HTTP header insertion is not performed for the CONNECT method.

Custom headers and client IP and port headers are inserted in every HTTP request packet. Full session headers and decoded client certificate fields and the full certificate in PEM format are inserted in the first HTTP request packets; only the session ID is inserted in subsequent HTTP requests that use the same session ID. Servers are expected to cache the session or client certificate headers and the full certificate in PEM format based on the session ID and use the session ID in subsequent requests to get the session and client certificate headers.

You can configure up to 100 HTTP header insertion policies, each policy consisting of 1 prefix and 16 custom headers. Prefix and custom headers can include up to 239 characters.

The information that can be inserted in the HTTP header is described in the following sections:

- Prefix, page 4-8
- Client Certificate Headers, page 4-8
- Client IP and Port Address Headers, page 4-9
- Custom Headers, page 4-9
- Header Alias, page 4-9
- SSL Session Headers, page 4-9

Prefix

When you specify **prefix** *prefix_string*, the SSL Services Module adds the specified prefix to every inserted HTTP header. Adding a prefix enables the server to identify connections as coming from the SSL Services Module, and not from other appliances. A prefix is not added to standard HTTP headers from the client. The *prefix_string* can be up to 239 characters.

Client Certificate Headers

Client certificate header insertion allows the backend server to see the attributes of the client certificate that the SSL Services Module has authenticated and approved. Client certificate headers are sent only once per session. The server is expected to cache these values using the session ID, which is also inserted with the headers. In subsequent requests, the server uses the session ID to look up the cached client certificate headers on the server itself.

Note If the client does not send a certificate, the SSL handshake fails. There is no data phase or header insertion.

Note You can insert the headers listed below by entering the **client-cert** command, or you can send the entire client certificate in PEM format by entering the **client-cert pem** command.

Note The client certificate headers, or the client certificate in PEM format, are inserted only if the policy's service is configured for client authentication. The root CA and intermediate CA certificates will not be inserted when client certificate is inserted in the HTTP header.

When you specify **client-cert**, the SSL Services Module passes the following headers to the backend server.

Field To Insert	Description
ClientCert-Valid	Certificate validity state
ClientCert-Error	Error conditions
ClientCert-Fingerprint	Hash output
ClientCert-Subject-CN	X.509 subject's common name
ClientCert-Issuer-CN	X.509 certificate issuer's common name
ClientCert-Certificate-Version	X.509 certificate version
ClientCert-Serial-Number	Certificate serial number
ClientCert-Data-Signature-Algorithm	X.509 hashing and encryption method
ClientCert-Subject	X.509 subject's distinguished name
ClientCert-Issuer	X.509 certificate issuer's distinguished name
ClientCert-Not-Before	Certificate is not valid before this date
ClientCert-Not-After	Certificate is not valid after this date
ClientCert-Public-Key-Algorithm	The algorithm used for the public key

Field To Insert	Description
ClientCert-RSA-Public-Key-Size	Size of the RSA public key
ClientCert-RSA-Modulus-Size	Size of the RSA private key
ClientCert-RSA-Modulus	RSA modulus
ClientCert-RSA-Exponent	The public RSA exponent
ClientCert-X509v3-Authority-Key-Identifier	X.509 authority key identifier
ClientCert-X509v3-Basic-Constraints	X.509 basic constraints
ClientCert-X509v3-Key-Usage	X.509 key usage
ClientCert-X509v3-Subject-Alternative-Name	X.509 subject alternative name
ClientCert-X509v3-CRL-Distribution-Points	X.509 CRL distribution points
ClientCert-X509v3-Authority-Information-Access	X.509 authority information access
ClientCert-Signature-Algorithm	Certificate signature algorithm
ClientCert-Signature	Certificate signature

Client IP and Port Address Headers

Network address translation (NAT) changes the client IP address and destination TCP port number information. When you specify **client-ip-port,** the SSL Services Module inserts the client IP address and TCP destination port information in the HTTP header, allowing the server to see the client IP address and destination port number.

Custom Headers

When you specify **custom** *custom_string*, the SSL Services Module inserts the user-defined header verbatim in the HTTP header. You can configure up to 16 custom headers per HTTP header policy. The *custom_string* can include up to 239 characters. If the string includes spaces, you must enclose it in quotes (“”).

Note The syntax for *custom_string* is in the form *name***:***value*. For example:

```
ssl-proxy(config-ctx-http-header-policy)# custom "SOFTWARE VERSION:3.1(1)"
```

Header Alias

Some applications use different names for the standard certificate header. You can create an alias for the standard name of the header so that the same value is passed using the aliased name instead of the standard name that the SSL Services Module sends. If you have specified a prefix for header insertion, the prefix is also applied to the aliased name.

SSL Session Headers

Session headers, including the session ID, are used to cache client certificates based on the session ID. Session headers are also cached based on the session ID if the server wants to track connections based on a particular cipher suite. The SSL Services Module inserts the full session headers in the HTTP request during full SSL handshake, but inserts only the session ID when the session resumes.

When you configure the SSL Services Module as a client, the module inserts the session ID of the connection between the module and the backend SSL server.

When you specify **session**, the SSL Services Module passes information specific to an SSL connection to the backend server in the form of the following session headers.

Field to insert	Description
Session-Id	The SSL session ID
Session-Cipher-Name	The symmetric cipher suite
Session-Cipher-Key-Size	The symmetric cipher key size
Session-Cipher-Use-Size	The symmetric cipher use size
Session-Step-Up	TRUE if the server presented a stepup certificate and the client renegotiated the cipher; otherwise FALSE
Session-Initial-Cipher-Name	If Session-Step-Up is TRUE, the initially negotiated cipher name
Session-Initial-Cipher-Key-Size	If Session-Step-Up is TRUE, the initially negotiated cipher's key size
Session-Initial-Cipher-Use-Size	If Session-Step-Up is TRUE, the initially negotiated cipher's use size

Configuring HTTP Header Insertion

To configure HTTP header insertion, perform this task:

	Command	Purpose
Step 1	`ssl-proxy(config)# ssl-proxy context` *name*	Enters the SSL context subcommand mode. The optional SSL context name name is used to specify an SSL virtualization instance
Step 2	`ssl-proxy(config-context)# policy http-header` *policy_name*	Configures HTTP header insertion.
Step 3	`ssl-proxy(config-ctx-http-header-policy)# {prefix` *prefix_string* `| client-cert [pem]| client-ip-port | custom` *custom_string*`} | session | alias` *user-defined-name standard-name*`]`	Specifies the prefix, type of header, or alias name of header. **Note** You can configure only one alias per standard name. You cannot configure the same alias name for multiple standard names.
Step 4	`ssl-proxy(config-ctx-http-header-policy)# exit`	Returns to context subcommand mode.
Step 5	`ssl-proxy(config-context)# service` *service_name*	Defines the name of the SSL proxy service. **Note** The *service_name* value is case-sensitive.
Step 6	`ssl-proxy(config-ctx-ssl-proxy)# policy http-header` *http_header_policy_name*	Applies the HTTP header policy to the proxy server, for request.

The following example shows how to configure the SSL Services Module to insert a prefix, session headers, custom headers, and a header alias:

```
ssl-proxy(config)# ssl-proxy context s1
ssl-proxy(config-context)# policy http-header ssl-offload
ssl-proxy(config-ctx-http-header-policy)# prefix SSL-OFFLOAD
ssl-proxy(config-ctx-http-header-policy)# session
ssl-proxy(config-ctx-http-header-policy)# custom "SOFTWARE VERSION:3.1(1)"
ssl-proxy(config-ctx-http-header-policy)# custom "module:SSL MODULE - CATALYST 6500"
ssl-proxy(config-ctx-http-header-policy)# custom
type-of-proxy:server_proxy_1024_bit_key_size
ssl-proxy(config-ctx-http-header-policy)# alias My-Session-Cipher session-cipher-name
ssl-proxy(config-ctx-http-header-policy)# exit
ssl-proxy(config-context)# ssl-proxy service ssl-offload
ssl-proxy(config-ctx-ssl-proxy)# policy http-header ssl-offload
```

In addition to the standard HTTP headers, the following header information is inserted:

Note The alias name (My-Session-Cipher) is used instead of the standard name (session-cipher-name).

```
SSL-OFFLOAD-SOFTWARE VERSION:3.1(1)
SSL-OFFLOAD-module:SSL MODULE - CATALYST 6500
SSL-OFFLOAD-type-of-proxy:server_proxy_1024_bit_key_size
SSL-OFFLOAD-Session-Id:33:FF:2C:2D:25:15:3C:50:56:AB:FA:5A:81:0A:EC:E9:00:00:0A:03:00:60:
 2F:30:9C:2F:CD:56:2B:91:F2:FF
SSL-OFFLOAD-My-Session-Cipher:RC4-SHA
SSL-OFFLOAD-Session-Cipher-Key-Size:128
SSL-OFFLOAD-Session-Cipher-Use-Size:128
SSL-OFFLOAD-Session-Step-Up:FALSE
SSL-OFFLOAD-Session-Initial-Cipher-Key-Size:
SSL-OFFLOAD-Session-Initial-Cipher-Name:
SSL-OFFLOAD-Session-Initial-Cipher-Use-Size:
```

Configuring URL Rewrite

In a typical SSL offloading environment, an SSL offloader terminates secure client HTTP (HTTPS) connections, decrypts the SSL traffic into cleartext, and forwards the cleartext to a Web server through an HTTP connection. The HTTPS connections become non-secure HTTP connections at the backend server. The backend server doesn't know that the client connection came in as a secure connection.

If data returned to the client contains an HTTP redirection link, and the client follows this link, the client leaves the secure domain and no longer has a secure connection. The redirected link may not be available from the server using a clear text connection.

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

Note The URL rewrite feature supports the rewriting of redirection links. The system scans only the "Location:" HTTP header field in the response from the server and rewrites the rules accordingly. The URL rewrite feature does not support embedded links.

The URL rewrite feature rewrites the protocol and the nondefault port (default ports are port 80 for cleartext and port 443 for SSL).

You can configure up to 100 URL rewrite policies, each policy consisting of up to 32 rewrite rules per SSL proxy service, up to 200 characters per rule.

Follow these guidelines for URL rewrite:

- An exact URL match takes precedence over a wildcard rule. A suffix wildcard rule takes precedence over a prefix wildcard rule.

  For example, **www.cisco.com** takes precedence, then **www.cisco.***, then ***.cisco.com**.
- Enter only one suffix or prefix wildcard rule at one time. For example, do not enter **www.cisco.*** and **www.cisco.c*** in the same policy. Similarly, do not enter ***w.cisco.com** and ***.cisco.com** in the same policy.
- Do not enter two exact URL match rules in the same policy. For example, do not enter **www.cisco.com clearport 80 sslport 443** and **www.cisco.com clearport 81 sslport 444** in the same policy. In this case, the second rule entered overwrites the first rule.
- URL rewrite is performed for both offload and backend (HTTP-to-HTTPS, and HTTPS-to-HTTP). This includes port rewrites.

To configure URL rewrite, perform this task:

	Command	Purpose
Step 1	`ssl-proxy(config)# ssl-proxy context` *name*	Enters the SSL context subcommand mode. The optional SSL context name name is used to specify an SSL virtualization instance.
Step 2	`ssl-proxy(config-context)# policy url-rewrite` *policy_name*	Configures the URL rewrite policy.
Step 3	`ssl-proxy(config-ctx-url-rewrite-policy)# url` *url* `[clearport` *port_number*[1]`] [sslport` *port_number*[2]`]`	Specifies the URL rewrite rules. You can configure up to 32 rewrite rules per SSL proxy service, up to 240 characters per rule. **Note** You should enter only one suffix or prefix wildcard character (*) only once per rewrite rule.
Step 4	`ssl-proxy(config-ctx-url-rewrite-policy)# exit`	Returns to config mode.
Step 5	`ssl-proxy(config-context)# service` *service_name*	Defines the name of the SSL proxy service. **Note** The *service_name* value is case-sensitive.
Step 6	`ssl-proxy(config-ctx-ssl-proxy)# policy url-rewrite` *policy_name*	Applies the URL rewrite policy.

1. The **clearport** *port_number* specifies the port portion of the URL to be rewritten. Specify the **cleartext** *port_number* if it is not the default cleartext port 80.
2. The **sslport** *port_number* specifies the port portion of the URL that should be rewritten. Specify the **ssltext** *port_number* if it is not the default SSL port 443.

Note When a server includes the default HTTP port number 80 in a URL redirect (for example, www.example.com:80), then the **url** command must be configured in the same manner (for example, **url www.example.com:80**). Non-standard port numbers need not be configured as part of the URL, but may instead by configured using the **clearport** keyword.

This example shows how to configure URL rewrite policy and apply the policy to a proxy service:

```
ssl-proxy(config)# ssl-proxy context s1
ssl-proxy(config-context)# policy url-rewrite cisco_url
ssl-proxy(config-ctx-url-rewrite-policy)# url www.cisco.*
ssl-proxy(config-ctx-url-rewrite-policy)# url www.cisco.com clearport 81 sslport 444
ssl-proxy(config-ctx-url-rewrite-policy)# url wwwin.cisco.com clearport 81 sslport 440
ssl-proxy(config-ctx-url-rewrite-policy)# url 10.1.1.10 clearport 81 sslport 444
ssl-proxy(config-ctx-url-rewrite-policy)# exit
ssl-proxy(config-context)# service cisco_service
ssl-proxy(config-ctx-ssl-proxy)# policy url-rewrite cisco_url
```

See Table 4-1 for examples that show URL rewrite.

Table 4-1 Rules and Outcome for Server Proxy

URL Rewrite Rule	URLs that Match	URL Rewrite:
url www.cisco.com	http://www.cisco.com/	https://www.cisco.com/
url www.cisco.com clearport 81	http://www.cisco.com:81/	https://www.cisco.com/
url www.cisco.com sslport 444	http://www.cisco.com/	https://www.cisco.com:444/
url www.cisco.com clearport 81 sslport 444	http://www.cisco.com:81/	https://www.cisco.com:444/

Health Probe

You can configure the SSL Services Module to probe a server to detect a server failure. When a server failure is detected, the SSL service relays the failure condition to the SLB device.

To configure the TCP health probe, perform this task:

	Command	Purpose
Step 1	`ssl-proxy(config-context)# policy health-probe tcp` *name*	Configures the health probe policy.
Step 2	`ssl-proxy(config-ctx-tcp-probe)# interval` *seconds*	(Optional) Sets the interval between probes in seconds (from the end of the previous probe to the beginning of the next probe) when the server is healthy. The default is 30 seconds. The valid range is from 30 to 300 seconds.
Step 3	`ssl-proxy(config-ctx-tcp-probe)# failed-interval` *seconds*	(Optional) Sets the time between health checks after the service has been marked as failed. The default is 60 seconds. The valid range is from 30 to 3600 seconds.
Step 4	`ssl-proxy(config-ctx-tcp-probe)# maximum-retry` *retries*	(Optional) Sets the number of failed probes that are allowed before marking the service as failed. The default is 0 retries. The valid range is from 1 to 5 retries.

	Command	Purpose
Step 5	`ssl-proxy(config-ctx-tcp-probe)#` **`open-timeout`** *seconds*	(Optional) Sets the maximum time to wait to establish a TCP connection. The default is 80 seconds. The valid range is from 70 to 120 seconds.
Step 6	`ssl-proxy(config-ctx-tcp-probe)#` **`port`** *port_number*	(Optional) Configures an optional port for the health probe. Valid values are from 1 to 65535. By default, the TCP health probe uses the server IP address and port for the SSL server proxy service. Enter the **port** command to specify a different port for the health probe. If you configured the SSL server proxy service with **no nat server**, the TCP health probe uses the virtual IP address that you configured on the SSL server proxy service instead of the server IP address. **Note** TCP health probe is not supported when you configure a wildcard proxy and **no nat server** on the SSL server proxy service. See the "SSL Server Proxy Services" section on page 3-45 for information about configuring the SSL server proxy service.

The following example shows how to configure TCP health probe to check whether service at port 81 (server port) is up and running on server IP address 19.0.0.1

```
ssl-proxy(config)# ssl-proxy context ssl
ssl-proxy(config-context)# service ssl-1
ssl-proxy(config-ctx-ssl-proxy)# virtual ipddr 7.100.100.180 protocol tcp port 443
ssl-proxy(config-ctx-ssl-proxy)# server ipaddr 19.0.0.1 protocol tcp port 81
ssl-proxy(config-ctx-ssl-proxy)# certificate rsa general-purpose trustpoint cert1024
ssl-proxy(config-ctx-ssl-proxy)# policy health-probe tcp probe1
ssl-proxy(config-ctx-ssl-proxy)# inservice
ssl-proxy(config-ctx-ssl-proxy)# exit
ssl-proxy(config-context)# policy health-probe tcp probe1
ssl-proxy(config-ctx-tcp-probe)# end
ssl-proxy#
```

The following example shows the state of the SSL proxy service when the health probe has failed:

Note The proxy service is down until service at port 81 is up and running again.

```
ssl-proxy# show ssl-proxy service ssl-1 context ssl
Service id: 0, bound_service_id: 256
Virtual IP: 7.100.100.180, port: 443
Server IP: 19.0.0.1, port: 81
TCP Health Probe Policy: probe1
rsa-general-purpose certificate trustpoint: cert1024
  Certificate chain for new connections:
    Certificate:
       Key Label: cert1024.key, 1024-bit, exportable
       Key Timestamp: 05:18:23 UTC Dec 30 2005
       Serial Number: 12F332E200000000000D
    Root CA Certificate:
       Serial Number: 6522F512C30E078447D8AFC35567B101
```

```
  Certificate chain complete
Context name: ssl
Context Id  : 1
Admin Status: up
Operation Status: down
Proxy status: Health Probe Failed
```

Configuring NAT

Client connections originate from the client and are terminated on the SSL Services Module. Server connections originate from the SSL Services Module.

You can configure client NAT, server NAT, or both, on the server connection.

Server NAT

The server IP address configured with the **ssl-proxy service** command specifies the IP address and port for the destination device, either the CSM or the real server for which the SSL Services Module acts as a proxy. If you configure server NAT, the server IP address is used as the destination IP address for the server connection. If the server NAT is not configured, the destination IP address for the server connection is the same as the **virtual ipaddress** for which SSL Services Module is a proxy. The server IP address points to a next hop router IP address or a virtual IP address on the CSM. The SSL Services Module always performs the port translation by using the port number entered in the **server ipaddress** subcommand.

To configure server NAT, enter the **nat server** subcommand under the **ssl-proxy service** command:

	Command	Purpose
Step 1	`ssl-proxy(config)# ssl-proxy context` *name*	Enters the SSL context subcommand mode. The optional SSL context name name is used to specify an SSL virtualization instance.
Step 2	`ssl-proxy(config-context)# service` *service_name*	Defines the name of the SSL proxy service. **Note** The *service_name* value is case-sensitive.
Step 3	`ssl-proxy(config-ctx-ssl-proxy)# nat server` *natpool_name*	Enables a NAT server address for the server connection of the specified service SSL offload.

Client NAT

If you configure client NAT, the server connection source IP address and port are derived from a NAT pool. If client NAT is not configured, the server connection source IP address and port are derived from the source IP address and source port of the client connection.

Allocate enough IP addresses to satisfy the total number of connections supported by the SSL Services Module (256,000 connections). Assuming you have 32,000 ports per IP address, configure 8 IP addresses in the NAT pool. If you try to configure fewer IP addresses than required by the total connections supported by the SSL Services Module, the command is rejected.

Note The addresses specified in the **natpool** command must match the address of one of the connected networks configured on the SSL Services Module subinterfaces.

To configure a NAT pool and assign the NAT pool to the proxy service, perform this task:

	Command	Purpose
Step 1	`ssl-proxy(config)# ssl-proxy context` *name*	Enters the SSL context subcommand mode. The optional SSL context name name is used to specify an SSL virtualization instance.
Step 2	`ssl-proxy(config-context)# natpool` *natpool_name start_ip_addr end_ip_addr* `netmask` *netmask*	Defines a pool of IP addresses that the SSL Services Module uses for implementing the client NAT.
Step 3	`ssl-proxy(config-context)# service` *service_name*	Defines the name of the SSL proxy service. **Note** The *service_name* value is case-sensitive.
Step 4	`ssl-proxy(config-ctx-ssl-proxy)# nat client` *natpool_name*	Configures a NAT pool for the client address used in the server connection of the specified service SSL offload.

Configuring Redundancy

In systems with an SSL Services Module and a Content Switching Module (CSM), the failover functionality on the CSM provides stateless redundancy on the SSL Services Module. When the SSL Services Module is used in a standalone configuration (using policy-based routing), you can configure HSRP to provide redundancy.

When configuring HSRP, note the following restrictions:

- You can configure up to 16 HSRP groups per SSL Services Module.
- You can configure HSRP for an SSL Services Module or for individual proxy services.
- Configuration changes in one module do not automatically propagate to the redundant module. You must manually apply configuration changes to all SSL Services Modules in the redundant system.
- The SSL Services Module software provides only stateless redundancy. When a standby module takes over the functionality of the active module, the existing connections are lost. New connections are established on the standby (now active) module using the same configuration available on the active module.

Before you configure HSRP on the SSL Services Module, do the following:

1. Configure proxy services. Enter the **secondary** keyword when you specify the virtual IP address.

   See the "Configuring SSL Proxy Services" section on page 3-45 for information on configuring proxy services.

2. Configure client NAT on the proxy service.

   See the "Client NAT" section on page 4-15 for information on configuring client NAT.

3. Configure policy-based routing. The next hop IP address is the standby IP address specified below.

   See the "Configuring Policy-Based Routing" section on page 5-2 or information on configuring policy-based routing.

To configure HSRP on the SSL Services Module, perform this task:

	Command	Purpose
Step 1	`ssl-proxy(config)# interface ssl-proxy 0.`*subinterface-number*	Selects a subinterface to configure.
Step 1	`ssl-proxy(config-subif)# standby` [*group_number*] **ip** *ip_address*	Enable HSRP and specify the HSRP IP address. If you do not specify a *group_number*, group 0 is used.
Step 2	`ssl-proxy(config-subif)# standby` [*group_number*] **priority** *priority*	(Optional) Specify the priority for the HSRP group on this module. The default *group_number* is 100. The module with the highest priority becomes active for that HSRP group.
Step 3	`ssl-proxy(config-subif)# standby` [*group_number*] [**preempt**] [**delay** *delay*]	(Optional) Configure the group to preempt the current active HSRP group and become active if the group priority is higher than the priority of the current active interface.
Step 4	`ssl-proxy(config-subif)# standby` [*group_number*] **timers** *hellotime holdtime*	(Optional) Set the HSRP hello timer and holdtime timer for the group. The default values are 3 (hello) and 10 (holdtime). All modules in the HSRP group should use the same timer values.
Step 5	`ssl-proxy(config-subif)# standby` [*group_number*] **authentication** *string*	(Optional) Specify a clear-text HSRP authentication string for the group. All modules in the HSRP group should use the same authentication string.

This example shows how to configure the HSRP on the SSL Services Module:

```
ssl-proxy(config)# interface ssl-proxy 0.100
ssl-proxy(config-subif)# encapsulation dot1q 100
ssl-proxy(config-subif)# ip address 10.1.0.20 255.255.255.0
ssl-proxy(config-subif)# standby 1 ip 10.1.0.21
ssl-proxy(config-subif)# standby 1 priority 110
ssl-proxy(config-subif)# standby 1 preempt
ssl-proxy(config-subif)# standby 2 ip 10.1.0.22
ssl-proxy(config-subif)# standby 2 priority 100
ssl-proxy(config-subif)# standby 2 preempt
ssl-proxy(config-subif)# exit
ssl-proxy(config)# ip route 223.255.254.0 255.255.255.0 10.1.0.254
```

Configuring TACACS, TACACS+, and RADIUS

For information on configuring TACACS, TACACS+, and RADIUS, refer to following URLs:

- "Configuring RADIUS" chapter in the *Cisco IOS Security Configuration Guide, Release 12.2*:
  http://www.cisco.com/univercd/cc/td/doc/product/software/ios122/122cgcr/fsecur_c/fsecsp/scfrad.htm
- "Configuring TACACS+" chapter in the *Cisco IOS Security Configuration Guide, Release 12.2*:
  http://www.cisco.com/univercd/cc/td/doc/product/software/ios122/122cgcr/fsecur_c/fsecsp/scftplus.htm

Configuring SNMP Traps

Note For more information on MIBs, refer to this URL:
http://www.cisco.com/public/sw-center/netmgmt/cmtk/mibs.shtml

Supported MIBs are as follows:

- CISCO-SSL-PROXY-MIB (all objects are read-only)
  - cspGlobalConfigGroup

    Version string

    Supported cipher suites

    Trap configuration setting
  - cspProxyServiceConfigGroup

    Type of proxy service

    IP addresses and TCP ports

    Policy names

    Keys and certificates
  - cspProxyServiceNotificationGroup

    Proxy service operational status change

    Proxy service certificate expiration warning
  - cspSslGroup

    Protocol counters

    Error counters

    Cumulative total values are reported in get responses, even if counters are cleared using CLI commands.
  - cspSsl3Group
  - cspTls1Group
  - cspSslErrorGroup
  - cspCpuStatusGroup

    Utilization of each CPU

    If counters have been cleared using CLI commands, the current values are reported in get responses, and the time of the last clear command are reported.
- CISCO-SSL-PROXY-CAPABILITY

To enable SNMP traps, perform this task:

	Command	Purpose
Step 1	`ssl-proxy(config)# snmp-server host addr traps version version ssl-proxy`	Specifies the IP address of an external network management device to which traps are sent.
Step 2	`ssl-proxy(config)# snmp-server enable traps ssl-proxy cert-expiring`	(Optional) Enable the SSL proxy certificate expiration notification trap. **Note** If you have set the certificate check-expiring interval to **0**, expiration notification traps are not sent. See the "Configuring Certificate Expiration Warning" section on page 3-39 for information on enabling certificate expiration warnings. **Note** Expiration notification traps are sent only for proxy service certificates that are currently configured.
Step 3	`ssl-proxy(config)# snmp-server enable traps ssl-proxy oper-status`	(Optional) Enable the SSL proxy operation status notification trap.
Step 4	`ssl-proxy(config)# snmp-server queue-length length`	(Optional) Specifies the number of trap events that are held before the queue must be emptied. The default *length* is 10; valid values are 1 through 1000.
Step 5	`ssl-proxy# show snmp`	Displays the SNMP information.

This example shows how to enable SNMP traps:

```
ssl-proxy# configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
ssl-proxy(config)# snmp-server host 10.1.1.1 traps version 2c ssl-proxy
ssl-proxy(config)# snmp-server enable traps ssl-proxy cert-expiring
*Nov 27 03:47:10.739:%STE-6-PROXY_CERT_EXPIRING_TRAP_ENABLED:SNMP trap for proxy service
certificate expiration warning has been enabled.
ssl-proxy(config)# snmp-server enable traps ssl-proxy oper-status
*Nov 27 03:46:59.607:%STE-6-PROXY_OPER_STATUS_TRAP_ENABLED:SNMP trap for proxy service
operational status change has been enabled.
ssl-proxy(config)# snmp-server queue-length 256
ssl-proxy(config)# end

ssl-proxy# show snmp
0 SNMP packets input
    0 Bad SNMP version errors
    0 Unknown community name
    0 Illegal operation for community name supplied
    0 Encoding errors
    0 Number of requested variables
    0 Number of altered variables
    0 Get-request PDUs
    0 Get-next PDUs
    0 Set-request PDUs
```

```
8 SNMP packets output
    0 Too big errors (Maximum packet size 1500)
    0 No such name errors
    0 Bad values errors
    0 General errors
    0 Response PDUs
    8 Trap PDUs

SNMP logging:enabled
    Logging to 10.1.1.1.162, 0/256, 0 sent, 0 dropped.
ssl-proxy#
```

Enabling the Cryptographic Self-Test

Note The power-on crypto chip self-test and key test are run only once at bootup.

Note Use the self-test for troubleshooting only. Running this test will impact run-time performance.

To run the self-test, perform this task:

Command	Purpose
`ssl-proxy(config)#` **`ssl-proxy crypto self-test time-interval`** *`time`*	Enables the crytographic self-test. The default value for *time* is 3 seconds; valid values are 1 though 8.

This example shows how to enable the cryptographic self-test and display cryptographic information:

```
ssl-proxy(config)# ssl-proxy crypto self-test time-interval 1
ssl-proxy(config)# end
```

Displaying Statistics Information

To display statistics information, perform this task:

Command	Purpose
`ssl-proxy(config)#` **`show ssl-proxy stats`** {**`crypto`** | **`hdr`** | **`ipc`** | **`pki`** [**`auth`** | **`cache`** | **`cert-header`** | **`database`** | **`expiring`** | **`history`** | **`ipc`** | **`memory`**] | **`service`** | **`ssl`** | **`tcp`** | **`url`**}	Displays specified statistics information.

This example shows how to display header insertion information:

```
ssl-proxy# show ssl-proxy stats hdr
Header Insert Statistics:
    Session Headers Inserted : 1          Custom Headers Inserted : 2
    Session Id's Inserted    : 1          Client Cert. Inserted   : 0
    Client IP/Port Inserted  : 0          PEM Cert. Inserted      : 0
    Aliased Hdrs Inserted    : 1
    No End of Hdr Detected   : 0          Payload no HTTP header  : 0
    Desc Alloc Failed        : 0          Buffer Alloc Failed     : 0
    Client Cert Errors       : 0          Malloc failed           : 0
    Service Errors           : 0          Conn Entry Invalid      : 0
    Buffers allocated        : 0          Buffers Scanned         : 1
    Insertion Points Found   : 1          Hdrs Spanning Records   : 0
    End of Header Found      : 1          Buffers Accumulated     : 1
    Multi-buffer IP Port     : 0          Multi-buffer Session Id : 0
    Multi-buffer Session Hdr : 0          Multi-buffer Custom Hdr : 0
    Scan Internal Error      : 0          Database Not Initialized: 0
```

See Table 4-2 for descriptions of the header insertion statistics:

Note The remaining statistics are for internal debugging.

Table 4-2 Header Insertion Statistics

Statistic	Description
Session Headers Inserted	The number of times session headers were inserted.
Custom Headers Inserted	The number of custom headers inserted.
Session Id's Inserted	The number of times session id was inserted.
Client Cert. Inserted	The number of times headers from a client certificate were inserted.
Client IP/Port Inserted	The number of times the client's IP address and port number were inserted.
PEM Cert. Inserted	The number of times a client certificate was inserted in PEM format.
Aliased Hdrs Inserted	The number of aliased headers inserted.

This example shows how to display crypto information:

```
ssl-proxy# show ssl-proxy stats crypto
Crypto Statistics from SSL Module:1
  Self-test is running
  Current device index is 1
  Time interval between tests is 1 seconds
  Device 0 statistics:
    Total Number of runs:50
    Runs all passed:50
    Number of timer error:0
    ------------------------------------------------------------
    Test Name                     Passed  Failed  Did-not-run
    ------------------------------------------------------------
      0  Power-on Crypto chip sel    1       0       0
      1  Power-on Crypto chip key    1       0       0
      2  Hash Test Case 1           50       0       0
      3  Hash Test Case 2           50       0       0
      4  Hash Test Case 3           50       0       0
      5  Hash Test Case 4           50       0       0
      6  SSL3 MAC Test Case 1       50       0       0
      7  SSL3 MAC Test Case 2       50       0       0
      8  TLS1 MAC Test Case 1       50       0       0
      9  TLS1 MAC Test Case 2       50       0       0
     10  DES Server Test            50       0       0
     11  DES Encrypt Test 1         50       0       0
     12  DES Decrypt Test 1         50       0       0
     13  DES Encrypt Test 2         50       0       0
     14  DES Decrypt Test 2         50       0       0
     15  ARC4 Test Case 1           50       0       0
     16  ARC4 Test Case 2           50       0       0
     17  ARC4 Test Case 3           50       0       0
     18  ARC4 State Test Case 1     50       0       0
     19  ARC4 State Test Case 2     50       0       0
     20  ARC4 State Test Case 3     50       0       0
     21  ARC4 State Test Case 4     50       0       0
     22  HMAC Test Case 1           50       0       0
     23  HMAC Test Case 2           50       0       0
     24  Random Bytes Generation    50       0       0
     25  RSA Encrypt/Decrypt Test   50       0       0
     26  Master Secret Generation   50       0       0
     27  Key Material Generation    50       0       0
     28  SSL3 Handshake Hash Test   50       0       0
     29  TLS1 Handshake Hash Test   50       0       0

  Device 1 statistics:
    Total Number of runs:49
    Runs all passed:49
    Number of timer error:0
    ------------------------------------------------------------
    Test Name                     Passed  Failed  Did-not-run
    ------------------------------------------------------------
      0  Power-on Crypto chip sel    1       0       0
      1  Power-on Crypto chip key    1       0       0
      2  Hash Test Case 1           50       0       0
      3  Hash Test Case 2           50       0       0
      4  Hash Test Case 3           50       0       0
      5  Hash Test Case 4           50       0       0
```

```
 6  SSL3 MAC Test Case 1       50      0       0
 7  SSL3 MAC Test Case 2       50      0       0
 8  TLS1 MAC Test Case 1       50      0       0
 9  TLS1 MAC Test Case 2       50      0       0
10  DES Server Test            50      0       0
11  DES Encrypt Test 1         50      0       0
12  DES Decrypt Test 1         50      0       0
13  DES Encrypt Test 2         50      0       0
14  DES Decrypt Test 2         50      0       0
15  ARC4 Test Case 1           50      0       0
16  ARC4 Test Case 2           50      0       0
17  ARC4 Test Case 3           50      0       0
18  ARC4 State Test Case 1     49      0       0
19  ARC4 State Test Case 2     49      0       0
20  ARC4 State Test Case 3     49      0       0
21  ARC4 State Test Case 4     49      0       0
22  HMAC Test Case 1           49      0       0
23  HMAC Test Case 2           49      0       0
24  Random Bytes Generation    49      0       0
25  RSA Encrypt/Decrypt Test   49      0       0
26  Master Secret Generation   49      0       0
27  Key Material Generation    49      0       0
28  SSL3 Handshake Hash Test   49      0       0
29  TLS1 Handshake Hash Test   49      0       0
```

This example shows how to display PKI certificate authentication and authorization statistics:

```
ssl-proxy# show ssl-proxy stats pki auth
Authentication request timeout:240 seconds
Max in process:100 (requests)
Max queued before dropping:0 (requests)
Certificate Authentication & Authorization Statistics:
  Requests started:2
  Requests finished:2
  Requests pending to be processed:0
  Requests waiting for CRL:0
  Signature only requests:0
  Valid signature:0
  Invalid signature:0
  Total number of invalid certificates:0
  Approved with warning (no crl check):2
  Number of times polling CRL:0
  No certificates present:0
  Failed to get CRL:0
  Not authorized (e.g. denied by ACL):0
  Root certificates not self-signed:0
  Verify requests failed (e.g. expired or CRL operation failed):0
  Unknown failure:0
  Empty certificate chain:0
  No memory to process requests:0
  DER encoded certificates missing:0
  Bad DER certificate length:0
  Failed to get key from certificate:0
  Issuer CA not in trusted CA pool:0
  Issuer CA certificates not valid yet:0
  Expired issuer CA certificates:0
  Peer certificates not valid yet:0
  Expired peer certificates:0
```

This example shows how to display PKI peer certificate cache statistics:

```
ssl-proxy# show ssl-proxy stats pki cache
Peer certificate cache size:0 (entries), aging timeout:30 (minutes)
Peer certificate cache statistics:
  In use:0 (entries)
  Cache hit:0
  Cache miss:0
  Cache allocated:0
  Cache freed:0
  Cache entries expired:0
  Cache error:0
  Cache full (wrapped around):0
  No memory for caching:0
```

Collecting Crash Information

The crash-info feature collects information necessary for developers to fix software-forced resets. Enter the **show ssl-proxy crash-info** command to collect software-forced reset information. You can retrieve only the latest crash-info in case of multiple software-forced resets. The **show ssl-proxy crash-info** command takes 1 to 6 minutes to complete the information collection process.

Note The **show stack** command is not a supported command to collect software-forced reset information on the SSL Service Module.

The following example shows how to collect software-forced reset information:

```
ssl-proxy# show ssl-proxy crash-info

===== SSL SERVICE MODULE - START OF CRASHINFO COLLECTION =====


------------- COMPLEX 0 [FDU_IOS] ----------------------

NVRAM CHKSUM:0xEB28
NVRAM MAGIC:0xC8A514F0
NVRAM VERSION:1

++++++++++ CORE 0 (FDU) ++++++++++++++++++++++

   CID:0
   APPLICATION VERSION:2003.04.15 14:50:20 built for cantuc
   APPROXIMATE TIME WHEN CRASH HAPPENED:14:06:04 UTC Apr 16 2003
   THIS CORE DIDN'T CRASH
   TRACEBACK:222D48 216894
   CPU CONTEXT  ----------------------------

$0 :00000000, AT :00240008, v0 :5A27E637, v1 :000F2BB1
a0 :00000001, a1 :0000003C, a2 :002331B0, a3 :00000000
t0 :00247834, t1 :02BFAAA0, t2 :02BF8BB0, t3 :02BF8BA0
t4 :02BF8BB0, t5 :00247834, t6 :00000000, t7 :00000001
s0 :00000000, s1 :0024783C, s2 :00000000, s3 :00000000
s4 :00000001, s5 :0000003C, s6 :00000019, s7 :0000000F
t8 :00000001, t9 :00000001, k0 :00400001, k1 :00000000
gp :0023AE80, sp :031FFF58, s8 :00000019, ra :00216894
LO :00000000, HI :0000000A, BADVADDR :828D641C
EPC :00222D48, ErrorEPC :BFC02308, SREG :34007E03
Cause 0000C000 (Code 0x0):Interrupt exception
```

```
CACHE ERROR registers  -------------------

CacheErrI:00000000, CacheErrD:00000000
ErrCtl:00000000, CacheErrDPA:0000000000000000

   PROCESS STACK -----------------------------
      stack top:0x3200000

   Process stack in use:

   sp is close to stack top;

   printing 1024 bytes from stack top:

031FFC00:06405DE0 002706E0 0000002D 00000001  .@]`.'.`...-....
031FFC10:06405DE0 002706E0 00000001 0020B800  .@]`.'.`..... 8.
031FFC20:031FFC30 8FBF005C 14620010 24020004  ..|0.?.\.b..$...
...........
...........
...........
FFFFFFD0:00000000 00000000 00000000 00000000 ................
FFFFFFE0:00627E34 00000000 00000000 00000000 .b~4............
FFFFFFF0:00000000 00000000 00000000 00000006 ................


===== SSL SERVICE MODULE - END OF CRASHINFO COLLECTION =======
```

Enabling Debugging

This sections describes how to debug PKI transactions and different module processors:

- Debugging PKI, page 4-25
- Debugging FDU, PKI, SSL, and TCP Processors, page 4-27

Debugging PKI

To display debug information about public key infrastructure (PKI) transactions, perform this task:

Command	Purpose
`ssl-proxy# [no] debug crypto pki {messages | transactions]`	Displays PKI debug messages. The **messages** keyword displays messages about the actual data being sent and received during public key infrastructure (PKI) transactions. The **transactions** keyword displays debug messages pertaining to PKI certificates. The messages show status information during certificate enrollment and verification.

The following example, which authenticates and enrolls a CA, contains sample output for the **debug crypto pki transactions** command:

```
ssl-proxy(config)# crypto ca authenticate msca

Certificate has the following attributes:
Fingerprint:A5DE3C51 AD8B0207 B60BED6D 9356FB00
% Do you accept this certificate? [yes/no]:y

ssl-proxy# debug crypto pki transactions

00:44:00:CRYPTO_PKI:Sending CA Certificate Request:
GET /certsrv/mscep/mscep.dll/pkiclient.exe?operation=GetCACert&message=msca HTTP/1.0

00:44:00:CRYPTO_PKI:http connection opened
00:44:01:CRYPTO_PKI:HTTP response header:
 HTTP/1.1 200 OK
Server:Microsoft-IIS/5.0
Date:Fri, 17 Nov 2000 18:50:59 GMT
Content-Length:2693
Content-Type:application/x-x509-ca-ra-cert

Content-Type indicates we have received CA and RA certificates.

00:44:01:CRYPTO_PKI:WARNING:A certificate chain could not be constructed while selecting
certificate status

00:44:01:CRYPTO_PKI:WARNING:A certificate chain could not be constructed while selecting
certificate status

00:44:01:CRYPTO_PKI:Name:CN = msca-rootRA, O = Cisco System, C = US
00:44:01:CRYPTO_PKI:Name:CN = msca-rootRA, O = Cisco System, C = US
00:44:01:CRYPTO_PKI:transaction GetCACert completed
00:44:01:CRYPTO_PKI:CA certificate received.
00:44:01:CRYPTO_PKI:CA certificate received.

ssl-proxy(config)# crypto ca enroll msca
%
% Start certificate enrollment ..
% Create a challenge password. You will need to verbally provide this
   password to the CA Administrator in order to revoke your certificate.
   or security reasons your password will not be saved in the configuration.
   Please make a note of it.

Password:
Re-enter password:

% The subject name in the certificate will be:Router.cisco.com
% Include the router serial number in the subject name? [yes/no]:n
% Include an IP address in the subject name? [yes/no]:n
Request certificate from CA? [yes/no]:y
% Certificate request sent to Certificate Authority
% The certificate request fingerprint will be displayed.
% The 'show crypto ca certificate' command will also show the fingerprint.
Router(config)#

00:44:29:CRYPTO_PKI:transaction PKCSReq completed
00:44:29:CRYPTO_PKI:status:
00:44:29:CRYPTO_PKI:http connection opened
00:44:29:CRYPTO_PKI: received msg of 1924 bytes
00:44:29:CRYPTO_PKI:HTTP response header:
 HTTP/1.1 200 OK
Server:Microsoft-IIS/5.0
Date:Fri, 17 Nov 2000 18:51:28 GMT
```

```
Content-Length:1778
Content-Type:application/x-pki-message

00:44:29:CRYPTO_PKI:signed attr:pki-message-type:
00:44:29:13 01 33
00:44:29:CRYPTO_PKI:signed attr:pki-status:
00:44:29:13 01 30
00:44:29:CRYPTO_PKI:signed attr:pki-recipient-nonce:
00:44:29:04 10 B4 C8 2A 12 9C 8A 2A 4A E1 E5 15 DE 22 C2 B4 FD
00:44:29:CRYPTO_PKI:signed attr:pki-transaction-id:
00:44:29:13 20 34 45 45 41 44 42 36 33 38 43 33 42 42 45 44 45 39 46
00:44:29:34 38 44 33 45 36 39 33 45 33 43 37 45 39
00:44:29:CRYPTO_PKI:status = 100:certificate is granted
00:44:29:CRYPTO__PKI:All enrollment requests completed.
00:44:29:%CRYPTO-6-CERTRET:Certificate received from Certificate Authority
```

Debugging FDU, PKI, SSL, and TCP Processors

A virtual terminal server (VTS) is built into the SSL Service Module for debugging different processors (FDU, PKI, SSL, TCP) on the module.

Note Use the TCP debug commands only to troubleshoot basic connectivity issues under little or no load conditions (for instance, when no connection is being established to the virtual server or real server).

If you use TCP debug commands, the TCP module displays large amounts of debug information on the console, which can significantly slow down module performance. Slow module performance can lead to delayed processing of TCP connection timers, packets, and state transitions.

From a workstation or PC, make a Telnet connection to one of the VLAN IP addresses on the module to reach the FDU (port 2001), TCP (port 2002), and SSL (port 2003) processor on the SSL Services Module.

To display debugging information for the different processors, perform this task:

Command	Purpose
ssl-proxy# [**no**] **debug ssl-proxy** {**fdu** | **pki** [**auth** | **ca-pool**]| **ssl** | **tcp**} [*type*]	Turns on or off the debug flags for the specified system component.

After you make the Telnet connection, enter the **debug ssl-proxy** {**fdu** | pki | **ssl** | **tcp**} command from the SSL Services Module console. One connection is sent from a client and displays the logs found in TCP console.

The following example shows how to display the log for TCP states for a connection and verify the debugging state:

```
ssl-proxy# debug ssl-proxy tcp state
ssl-proxy# show debugging
STE Mgr:
  STE TCP states debugging is on
```

The following example shows the output from the workstation or PC:

```
Conn 65066 state CLOSED --> state SYN_RECEIVED
Conn 65066 state SYN_RECEIVED --> state ESTABLISHED
Conn 14711 state CLOSED --> state SYN_SENT
Conn 14711 state SYN_SENT --> state ESTABLISHED
Conn 14711 state ESTABLISHED --> state CLOSE_WAIT
Conn 65066 state ESTABLISHED --> state FIN_WAIT_1
Conn 65066 state FIN_WAIT_1 --> state FIN_WAIT_2
Conn 65066 state FIN_WAIT_2 --> state TIME_WAIT
Conn 14711 state CLOSE_WAIT --> state LAST_ACK
Conn 14711 state LAST_ACK --> state CLOSED
##############Conn 65066 state TIME_WAIT --> state CLOSED
```

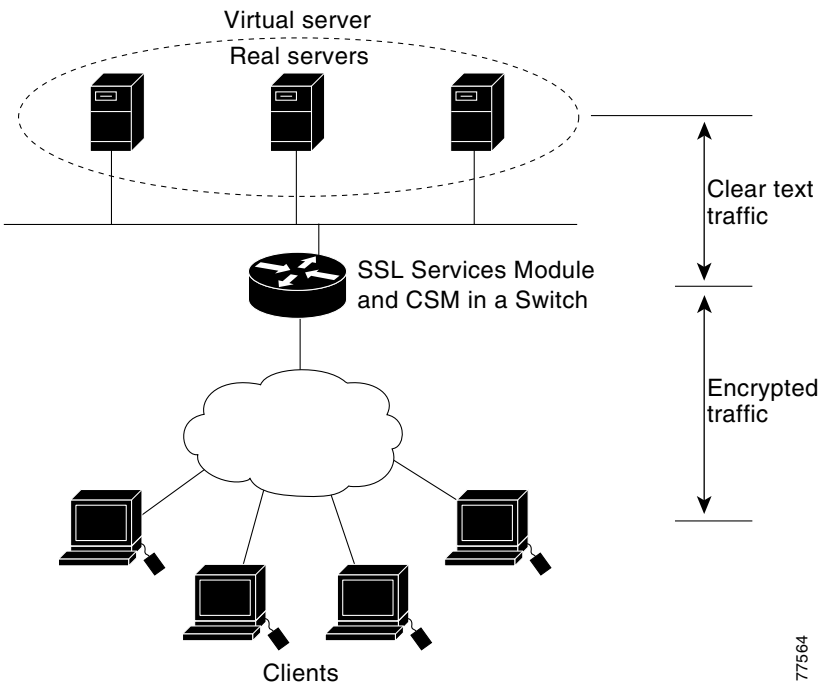
Configuring Different Modes of Operation

The SSL Services Module operates either in a standalone configuration or with a Content Switching Module (CSM). In a standalone configuration, secure traffic is directed to the SSL Services Module using policy-based routing. When used with a CSM, only encrypted client traffic is forwarded to the SSL Services Module, while clear text traffic is forwarded to real servers.

The following sections describe how to configure the SSL Services Module in a standalone configuration or with a CSM:

Figure 5-1 shows a sample network topology with an SSL Services Module and a CSM in a single Catalyst 6500 series switch.

Figure 5-1 Sample Network Layout—SSL Services Module with CSM



Configuring Policy-Based Routing

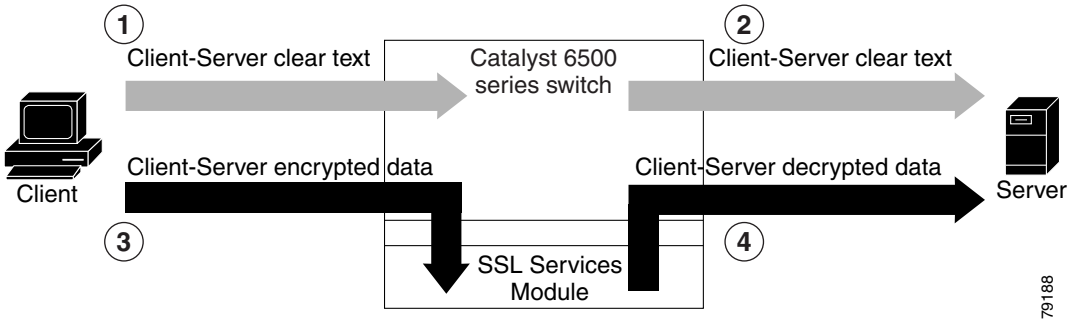
In a standalone configuration, encrypted SSL traffic is directed to the SSL Services Module using policy-based routing.

When you configure policy-based routing on the SSL Services Module, use the following guidelines:

- Configure clients and servers on separate subnets.
- Configure two VLANs (one for each subnet) on the switch.
- Configure IP interfaces on each VLAN.
- Configure an IP interface on the server-side VLAN of the SSL Services Module.

Two flows exist for each direction of traffic. In the client-to-server direction, traffic flow originates from the client as either clear text or as encrypted data. (See Figure 5-2.) In the server-to-client direction, all traffic originates from the server as clear text. However, depending on the source port, the traffic in the server-to-client direction may or may not be encrypted by the SSL Services Module before being forwarded to the client.

Figure 5-2 Client-to-Server Traffic Flow—Standalone Configuration



In Figure 5-2, the client sends clear text traffic to the server (as shown in flow 1). The switch then forwards clear text traffic to the server (flow 2).

The client sends encrypted traffic to the server (port 443); policy-based routing intercepts the traffic and forwards it to the SSL Services Module (flow 3). The SSL Services Module decrypts the traffic and forwards the stream to a well-known port (a port that has been configured on the server to expect decrypted traffic) (flow 4).

To enable policy-based routing, perform this task:

	Command	Purpose
Step 1	`Router(config)# ip access-list extended` *name*	Defines an IP extended access list.
Step 2	`Router(config-ext-nacl)# permit tcp` *source source-wildcard operator port destination destination-wildcard operator port*	Specifies conditions for the named access list. **Note** Use the **any** keyword as an abbreviation for a *source* and *source-wildcard* or *destination* and *destination-wildcard* of 0.0.0.0 255.255.255.255.
Step 3	`Router(config-ext-nacl)# route-map` *map-tag* [**permit** | **deny**] [*sequence-number*]	Defines a route map to control where packets are output. **Note** This command puts the switch into route map configuration mode.

	Command	Purpose
Step 4	`Router(config-route-map)# match ip address` *name*	Specifies the match criteria. Matches the source and destination IP address that is permitted by one or more standard or extended access lists.
Step 5	`Router(config-route-map)# set ip next-hop` *ip-address*	Sets the next hop to which to route the packet. (The next hop must be adjacent.)
Step 6	`Router(config-route-map)# interface` *interface-type interface-number*	Specifies the interface. **Note** This command puts the switch into interface configuration mode.
Step 7	`Router(config-if)# ip policy route-map` *map-tag*	Identifies the route map to use for policy-based routing. **Note** One interface can only have one **route-map** tag, but you can have multiple route map entries with different sequence numbers. These entries are evaluated in sequence number order until the first match. If there is no match, packets will be routed as usual.

Configuring the Content Switching Module

Note For detailed information on configuring the CSM, refer to the *Catalyst 6500 Series Switch Content Switching Module Installation and Configuration Note,* Release 3.1, at this URL:

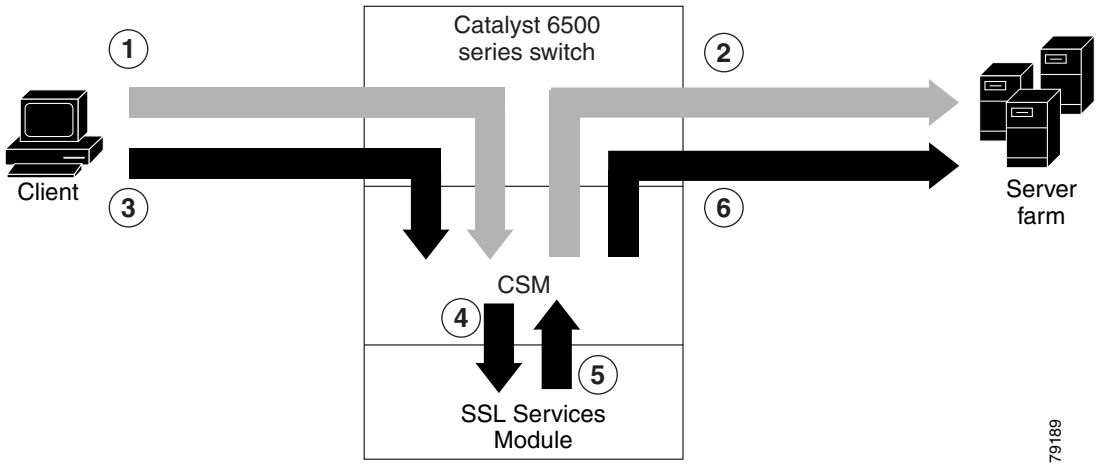
http://www.cisco.com/univercd/cc/td/doc/product/lan/cat6000/cfgnotes/csm_3_1/index.htm

The Content Switching Module (CSM) provides high-performance server load balancing (SLB) between network devices and server farms based on Layer 4 through Layer 7 packet information.

When you use the SSL Services Module with the CSM, only encrypted client traffic is forwarded to the SSL Services Module, while clear text traffic is forwarded to real servers.

The CSM parses for traffic destined to the server farm virtual IP address, port 443. The CSM forwards this traffic to the SSL Services Module without modifying the destination IP address. If there are multiple SSL Services Modules in the configuration, the CSM load balances the traffic across the SSL Services Modules. The SSL Services Module decrypts the traffic and forwards the new stream back to the CSM. The SSL Services Module does not change the destination IP address (the original server farm virtual IP address), but it does perform a port translation. With this new virtual IP address and port combination, the CSM balances the data across the servers in the server farm. (See Figure 5-3.)

Figure 5-3 Client-to-Server Traffic Flow—SSL Services Module and CSM



In Figure 5-3, clear text traffic is sent from the client to a virtual IP address, non-SSL port (for example, 80) (shown in flow 1). The CSM balances the clear text traffic across the servers in the server farm (flow 2).

Encrypted traffic is sent from the client to a virtual IP address, SSL port (443) (flow 3). The CSM forwards the encrypted traffic to the SSL Services Module (flow 4); if there is more than one SSL Services Module, the CSM balances the encrypted traffic across SSL Services Modules.

The SSL Services Module decrypts the traffic and forwards it to a virtual IP address and port on the CSM (flow 5).

The CSM balances the decrypted traffic across the servers in the server farm (flow 6).

On the return path, the CSM must monitor the port from which the server transmits data. If it is the standard clear text port (for example, 80), the data is forwarded back to the client unaltered, with the exception of the source address. If server NAT is configured on the clear text flow, the virtual IP address replaces the source IP address.

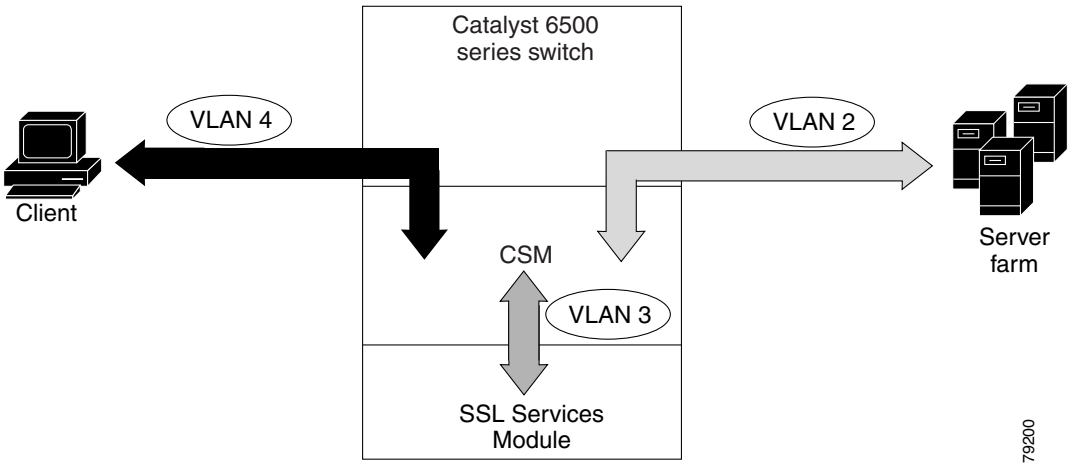
If traffic is destined to the virtual IP address and port 443, the CSM forwards this flow to the SSL Services Module. The SSL Services Module encrypts the traffic and performs port translation on the packet header. The SSL Services Module directs the traffic to the CSM with source port 443 (the SSL port to which the client originally directed encrypted traffic) so that the CSM can handle the reverse path traffic.

VLANs

As with normal CSM operation, you must configure separate client and server VLANs. If the CSM client and server VLANs are not on the same subnet, the CSM acts as a switch between the client and server VLANs.

To allow traffic to pass between the CSM and the SSL Services Module, you must configure a single VLAN between them. (See Figure 5-4.) All flows between the CSM and the SSL Services Module are on that VLAN.

Figure 5-4 SSL Services Module with CSM—3-VLAN Configuration



In Figure 5-4, VLAN 4 involves clear text and encrypted traffic between the client and the CSM virtual IP address.

VLAN 2 involves the following types of traffic between the server and the client:

- Clear text traffic between the client and the server
- Traffic sent by the client that was decrypted by the SSL Services Module
- Traffic sent by the server that needs to be encrypted by the SSL Services Module

VLAN 3 involves the following types of traffic between the CSM and the SSL Services Module:

- Encrypted client traffic that needs to be decrypted
- Decrypted client traffic that needs to be forwarded to the server farm
- Unencrypted server traffic that needs to be encrypted
- Encrypted server traffic that needs to be forwarded back to the client

To configure VLANs on the CSM, perform this task:

	Command	Purpose
Step 1	`Router(config)# mod csm slot`	Specifies the slot of the CSM.
Step 2	`Router(config-module-csm)# vlan vlan {client|server}`	Configures the VLAN as either a client or a server on the CSM.
Step 3	`Router(config-slb-vlan-client)# ip address ip_addr netmask`	Configures the IP address and netmask of the interface on the VLAN.
Step 4	`Router(config-slb-vlan-client)# gateway ip_addr`	Configures the gateway IP address.

Server Farms

When you use the SSL Services Module with a CSM, the CSM sees two types of server farms. The first server farm is the traditional farm that consists of a group of real servers and is mapped to one or more virtual server IP addresses. You may or may not choose to allow server or client NAT to act on traffic that goes to these servers.

The second type of server farm consists of the SSL Services Modules that are present in the chassis. The CSM views these SSL Services Modules as real servers and balances SSL traffic across the modules.

To configure a server farm on the CSM, perform this task:

	Command	Purpose
Step 1	`Router(config)# mod csm slot`	Specifies the slot of the CSM.
Step 2	`Router(config-module-csm)# serverfarm server_farm`	Configures the name of the server farm.
Step 3	`Router(config-slb-sfarm)# no nat server`	(Optional) Disables server NAT.
Step 4	`Router(config-slb-sfarm)# nat client natpool_name`	(Optional) Enables client NAT.
Step 5	`Router(config-slb-sfarm)# real ip_addr`	Configures the real IP address of the server.
Step 6	`Router(config-slb-real)# inservice`	Puts the server farm in service.

Virtual Servers

Three types of virtual servers are required for every real server farm supported in a CSM and SSL Services Module configuration. The main distinction between the three types of virtual servers is the port number. The clear text virtual server and the SSL virtual server have the same virtual IP address. The decryption virtual server may or may not have the same virtual IP address. The three types of virtual servers are as follows:

- Clear text virtual server—The clear text virtual server is the destination for any clear text traffic sent by the client. Typically, this traffic is destined to port 80. The CSM balances traffic sent to this virtual server directly to a real server in the server farm. The SSL Services Module is uninvolved.
- SSL virtual server—The SSL virtual server should be the destination for any SSL-encrypted traffic from the client to the server. This traffic is destined to port 443. The CSM forwards this type of traffic to the SSL Services Module for decryption.
- Decryption virtual server—After the SSL Services Module decrypts SSL traffic from the client, it forwards it back to the CSM, destined for the decryption virtual server. The CSM balances the traffic to a real server in the server farm, similar to the action it took for traffic destined to the clear text virtual server. The port associated with this decryption virtual server should match the port from which the real server has been configured to expect traffic decrypted by the SSL Services Module.

To configure a virtual server on the CSM, perform this task:

	Command	Purpose
Step 1	`Router(config)# mod csm slot`	Specifies the slot of the CSM.
Step 2	`Router(config-module-csm)# vserver vserver`	Configures the name of the virtual server.
Step 3	`Router(config-slb-vserver)# virtual ip_address tcp port`	Configures the IP address, protocol, and port of the virtual server.
Step 4	`Router(config-slb-vserver)# serverfarm server_farm`	Configures the destination server farm.

	Command	Purpose
Step 5	`Router(config-slb-vserver)#` **`vlan`** *`vlan`*	Specifies the VLAN from which the CSM accepts traffic for a specified virtual server. **Note** For security reasons, this command is required for the decryption virtual server.
Step 6	`Router(config-slb-vserver)#` **`inservice`**	Puts the virtual server in service.

Sticky Connections

Note Configuring the SSL sticky feature requires CSM software release 3.1(1a) or later releases on the CSM.

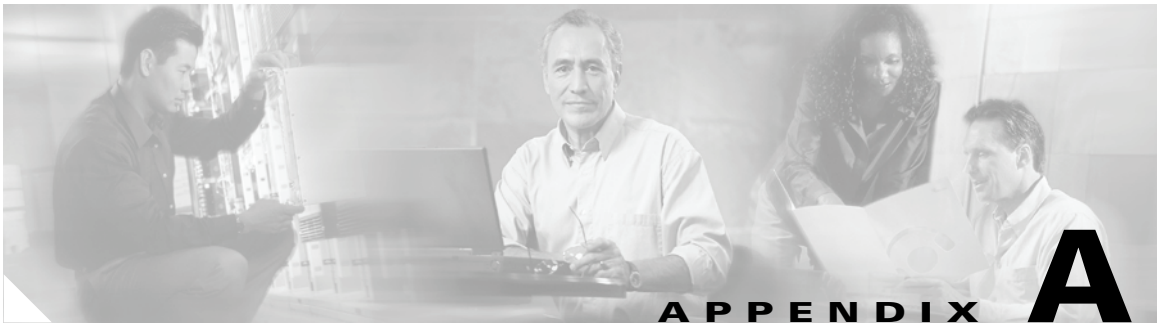
If a CSM and SSL Services Module configuration consists of multiple SSL Services Modules connected to a single CSM, configure the SSL sticky feature on the CSM to ensure that the CSM always forwards traffic from a particular client to the same SSL Services Module.

A 32-byte SSL session ID is created for each connection between a client and an SSL Services Module. With the SSL sticky feature configured, the CSM looks at a specific portion of the SSL session ID (the MAC address of the SSL Services Module) and load balances SSL traffic among the SSL Services Modules.

Note The MAC address of the SSL Services Module is always located at bytes 21 though 26 of the SSL session ID, even when the session ID is renegotiated.

To configure a sticky connection on the CSM, perform this task:

	Command	Purpose
Step 1	`Router(config)#` **`mod csm`** *`mod`*	Specifies the slot of the CSM.
Step 2	`Router(config-module-csm)#` **`sticky`** *`group`* **`ssl`**	Configures the sticky group ID.
Step 3	`Router(config-module-csm)#` **`vserver`** *`vserver`*	Associates the group ID with the virtual server.
Step 4	`Router(config-slb-vserver)#` **`sticky`** *`group`* **`timeout`** *`time`*	Specifies the amount of time, in minutes, that the connection remains sticky.
Step 5	`Router(config-slb-vserver)#` **`ssl-sticky offset 20 length 6`**	Specifies the location of the SSL Services Module MAC address in the SSL ID.



Example SSL Configurations

This appendix has the following sections:

Policy-Based Routing Configuration Example

This section shows a policy-based routing configuration example using a real client and a real server.

In Figure A-1, the SSL Services Module and the real server both have the IP address 3.100.100.151. The IP address on the SSL Services Module is configured with the **secondary** keyword and will not reply to ARP requests for this address, which avoids the problem of using duplicate IP addresses.

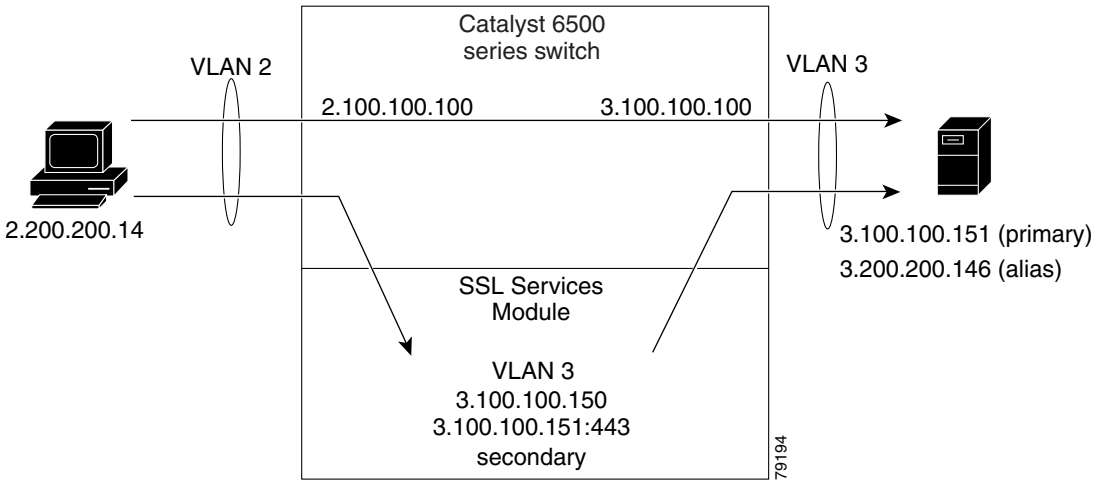
The client (2.200.200.14) is attached to a VLAN 2 switchport (access mode). The client's default gateway IP address is 2.100.100.100 (VLAN 2 IP address on the supervisor engine).

The real server is attached to a VLAN 3 switchport (access mode). The default gateway on the real server is 3.100.100.100 (VLAN 3 IP address on the supervisor engine). The real server has two addresses: 3.100.100.151 (primary) and 3.200.200.146 (alias).

Clear-text (HTTP) traffic destined for IP address 3.100.100.151 on port 80 is sent directly to the real server, which bypasses the SSL Services Module.

With policy-based routing, SSL traffic destined for IP address 3.100.100.151 on port 443 is redirected to the SSL Services Module for decryption. The decrypted traffic is sent to IP address 3.200.200.146 on port 81 (the alias IP address for the real server). The return traffic from the real server is forwarded to the SSL Services Module. The module encrypts the traffic and sends it to the client.

Figure A-1 Client-to-Server Traffic Flow Example



Configuring the Allowed VLANs

These examples show how to allow VLAN 3 between the SSL Services Module and the supervisor engine:

Cisco IOS Software

```
Router# configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
Router(config)# ssl-proxy module 8 allowed-vlan 3
Router(config)# ^Z
Router#
Router# show ssl-proxy module 8 state
SSL-proxy module 8 data-port:
 Switchport:Enabled
Administrative Mode:trunk
Operational Mode:trunk
Administrative Trunking Encapsulation:dot1q
Operational Trunking Encapsulation:dot1q
Negotiation of Trunking:Off
Access Mode VLAN:1 (default)
Trunking Native Mode VLAN:1 (default)
Trunking VLANs Enabled:3
Pruning VLANs Enabled:2-1001
Vlans allowed on trunk:3
Vlans allowed and active in management domain:3
Vlans in spanning tree forwarding state and not pruned:
   3
Allowed-vlan :3

Router#
```

Catalyst Operating System Software

```
Console> (enable) set trunk 8/1 3
Adding vlans 3 to allowed list.
Console> (enable) show trunk 8/1
* - indicates vtp domain mismatch
# - indicates dot1q-all-tagged enabled on the port
Port      Mode         Encapsulation  Status        Native vlan
--------  -----------  -------------  ------------  -----------
 8/1      nonegotiate  dot1q          not-trunking  1

Port      Vlans allowed on trunk
--------  ---------------------------------------------------------------------
 8/1      3

Port      Vlans allowed and active in management domain
--------  ---------------------------------------------------------------------
 8/1      3

Port      Vlans in spanning tree forwarding state and not pruned
--------  ---------------------------------------------------------------------
 8/1      3
```

Configuring the Access List and Route Map

This example shows how to configure the access list and route map for redirecting SSL traffic from the client to the SSL Services Module and for redirecting clear text traffic from the real server to the SSL Services Module:

```
Router# configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
Router(config)#
Router(config)# ip access-list extended redirect_ssl
Router(config-ext-nacl)# permit tcp any 3.0.0.0 0.255.255.255 eq 443
Router(config-ext-nacl)# !
Router(config-ext-nacl)# ip access-list extended reverse_traffic
Router(config-ext-nacl)# permit tcp 3.0.0.0 0.255.255.255 eq 81 any
Router(config-ext-nacl)# !
Router(config-ext-nacl)# route-map redirect_ssl permit
Router(config-route-map)# match ip address redirect_ssl
Router(config-route-map)# set ip next-hop 3.100.100.150
Router(config-route-map)# !
Router(config-route-map)# route-map reverse_traffic permit
Router(config-route-map)# match ip address reverse_traffic
Router(config-route-map)# set ip next-hop 3.100.100.150
Router(config-route-map)# !
Router(config-route-map)# interface Vlan2
Router(config-if)# ip address 2.100.100.100 255.0.0.0
Router(config-if)# ip policy route-map redirect_ssl
Router(config-if)# !
Router(config-if)# interface Vlan3
Router(config-if)# ip address 3.100.100.100 255.0.0.0
Router(config-if)# ip policy route-map reverse_traffic
Router(config-if)# !
Router(config-if)#^Z
Router#
```

Importing a Test Certificate

This example shows how to import the test certificate. For information on configuring a trustpoint and obtaining a certificate, see the "Configuring Keys and Certificates" section on page 3-3:

```
ssl-proxy# test ssl-proxy certificate install
% Opening file, please wait ...
% Writing, please wait .............
% Please use the following config command to import the file.
  "crypto pki import <trustpoint-name> pkcs12 nvram:test/testssl.p12 cisco"
% Then you can assign the trustpoint to a proxy service for testing.

*Oct  9 19:49:17.570:%STE-6-PKI_TEST_CERT_INSTALL:Test key and certificate was installed
into NVRAM in a PKCS#12 file.
ssl-proxy# configure terminal
ssl-proxy(config)# crypto pki import sample pkcs12 nvram: cisco
Source filename [sample]? test/testssl.p12
ssl-proxy(config)#
*Oct  9 19:51:04.674:%SSH-5-ENABLED:SSH 1.5 has been enabled
*Oct  9 19:51:04.678:%CRYPTO-6-PKCS12IMPORT_SUCCESS:PKCS #12 Successfully Imported.
ssl-proxy(config)# ^Z
ssl-proxy#
```

Configuring the SSL Proxy Subinterface

This example shows how to add an interface to VLAN 3 on the SSL Services Module:

```
ssl-proxy# configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
ssl-proxy(config)# interface SSL-Proxy 0.3
ssl-proxy(config-subif)# encapsulation dot1q 3
ssl-proxy(config-subif)# ip address 3.100.100.150 255.0.0.0
ssl-proxy(config-subif)# no shutdown
ssl-proxy(config-subif)# exit
ssl-proxy(config)# ^Z
ssl-proxy#
```

Configuring the SSL Proxy Service

This example shows how to add a specific proxy service that identifies a virtual IP address and a server IP address for each proxy:

```
ssl-proxy# configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
ssl-proxy(config)# ssl-proxy context Default
ssl-proxy(config-context)# service sample
ssl-proxy(config-ctx-ssl-proxy)# virtual ipaddr 3.100.100.151 protocol tcp port 443
secondary
ssl-proxy(config-ctx-ssl-proxy)# server ipaddr 3.200.200.146 protocol tcp port 81
ssl-proxy(config-ctx-ssl-proxy)# cert rsa general-purpose trustpoint sample
ssl-proxy(config-ctx-ssl-proxy)# inservice
ssl-proxy(config-ctx-ssl-proxy)# ^Z
ssl-proxy#
```

Verifying Service and Connections

This example shows how to verify the SSL proxy service and connections:

```
ssl-proxy# show ssl-proxy service sample
No context name provided, assuming context 'Default'…

Service id:3, bound_service_id:259
Virtual IP:3.100.100.151, port:443 (secondary configured)
Server IP:3.200.200.146, port:81
rsa-general-purpose certificate trustpoint:sample
  Certificate chain in use for new connections:
    Server Certificate:
       Key Label:sample
       Serial Number:01
    Root CA Certificate:
       Serial Number:00
  Certificate chain complete
Context name: Default
Context Id  : 0
Admin Status:up
Operation Status:up
ssl-proxy#

ssl-proxy# show ssl-proxy conn
Connections for TCP module 1
Local Address       Remote Address          VLAN   Conid  Send-Q Rwind  Recv-Q State
------------------- -------------------     -----  ------ ------ ------ ------ ------
3.100.100.151.443   2.200.200.14.37820      3      470     0     32768    0    ESTABLISHED
2.200.200.14.37820  3.200.200.146.81        3      471     0     32768    0    ESTABLISHED
ssl-proxy#
```

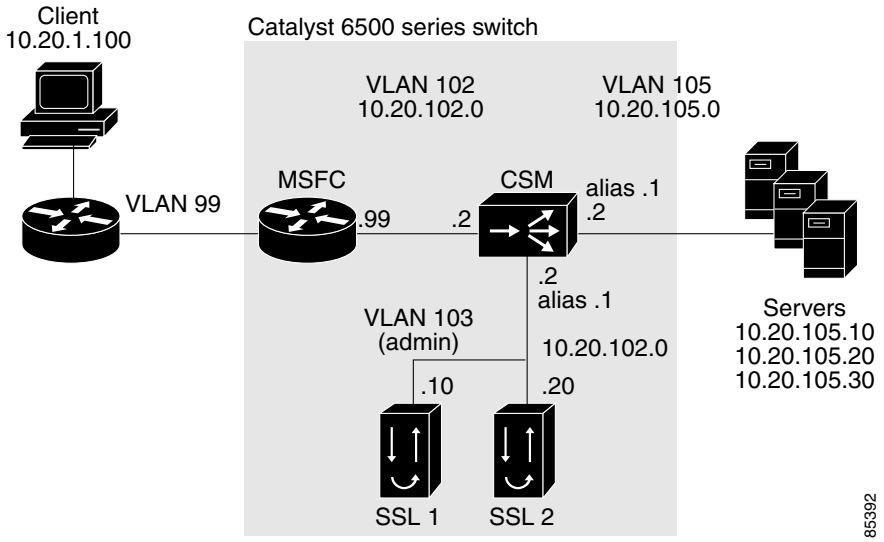
CSM and SSL Services Module Configuration Example (Bridge Mode, No NAT)

This section describes a CSM and SSL Services Module configuration that contains two SSL Services Modules, a CSM, a client network, and a server farm that has three web servers (IP addresses 10.20.105.10, 10.20.105.20, and 10.20.105.30).

In this example, the CSM client VLAN and CSM server VLAN for the SSL Services Modules are configured in the same IP subnet (bridge mode), while the CSM server VLAN for the web servers is in a separate IP subnet. (See Figure A-2.)

The CSM is configured so that it does not perform NAT operations when it is load balancing encrypted traffic to the SSL Services Modules. The SSL Services Modules are also configured not to perform NAT operations when they are sending decrypted traffic back to the CSM. The CSM is then configured to perform NAT for the decrypted traffic to the selected destination server.

Figure A-2 Bridge Mode, No NAT Configuration Example



The following addresses are configured on the CSM virtual servers:

- Client clear text traffic—10.20.102.100:80
- Client SSL traffic—10.20.102.100:443
- Decrypted traffic from SSL Services Modules—10.20.102.100:80

The following address is configured on the SSL virtual server:

- 10.20.102.100:443 (This IP address is configured with the **secondary** keyword.)

Figure A-2 shows VLAN 102 and VLAN 103 in the same subnet and VLAN 105 in a separate subnet.

Add all required VLANs to the VLAN database, and configure the IP interface for VLAN 102 on the MSFC. Configure VLAN 102, VLAN 103, and VLAN105 on the CSM. See the "Initial SSL Services Module Configuration" section on page 2-2 for information on how to configure VLANs and IP interfaces.

Note While VLAN 102 exists as Layer 3 interface on the MSFC, both VLAN 103 and VLAN 105 exist only as VLANs in the VLAN database and as CSM VLANs, but they do not have corresponding Layer 3 interfaces on the MSFC.

This example shows how to create the client and server VLANs on the CSM installed in slot number 5:

```
Router# configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
Router(config)# module csm 5
Router(config-module-csm)# vlan 102 client
Router(config-slb-vlan-client)# ip address 10.20.102.2 255.255.255.0
Router(config-slb-vlan-client)# gateway 10.20.102.99
Router(config-slb-vlan-client)# exit
Router(config-module-csm)# vlan 103 server
Router(config-slb-vlan-server)# ip address 10.20.102.2 255.255.255.0
Router(config-slb-vlan-server)# alias 10.20.102.1 255.255.255.0
Router(config-slb-vlan-server)# exit
```

```
Router(config-module-csm)# vlan 105 server
Router(config-slb-vlan-server)# ip address 10.20.105.2 255.255.255.0
Router(config-slb-vlan-server)# alias 10.20.105.1 255.255.255.0
Router(config-slb-vlan-server)# end
```

This example shows how to allow VLAN 103 between the SSL Services Module and the CSM:

Cisco IOS Software

```
Router(config)# ssl-proxy module 4 allowed-vlan 103
```

Catalyst Operating System Software

```
Console> (enable) set trunk 4/1 103
```

This example shows how to create the server farm of web servers (configured with server NAT) and the server farm of SSL Services Modules (configured with no server NAT):

```
Router# configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
Router(config)# module csm 5
Router(config-module-csm)# serverfarm SSLFARM
Router(config-slb-sfarm)# no nat server
Router(config-slb-sfarm)# real 10.20.102.10
Router(config-slb-real)# inservice
Router(config-slb-real)# real 10.20.102.20
Router(config-slb-real)# inservice
Router(config-slb-real)# exit
Router(config-slb-sfarm)# exit
Router(config-module-csm)# serverfarm WEBSERVERS
Router(config-slb-sfarm)# nat server
Router(config-slb-sfarm)# real 10.20.105.10
Router(config-slb-real)# inservice
Router(config-slb-real)# real 10.20.105.20
Router(config-slb-real)# inservice
Router(config-slb-real)# real 10.20.105.30
Router(config-slb-real)# inservice
Router(config-slb-real)# end
```

This example shows how to configure the three virtual servers. In this example, the web servers are receiving traffic on port 80 only, either directly from the clients or as decrypted traffic from the SSL Services Modules (since no port translation is configured).

The CSM distinguishes between requests received directly from the clients and requests received from the SSL Services Modules based on the VLAN from where the connections are received.

A sticky group is also configured to maintain stickiness based on the SSL ID.

```
Router# configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
Router(config)# module csm 5
Router(config-module-csm)# sticky 100 ssl timeout 30
Router(config-module-csm)# vserver CLEAR_VIP
Router(config-slb-vserver)# virtual 10.20.102.100 tcp www
Router(config-slb-vserver)# vlan 102
Router(config-slb-vserver)# serverfarm WEBSERVERS
Router(config-slb-vserver)# inservice
Router(config-slb-vserver)# exit
Router(config-module-csm)# vserver DECRYPT_VIP
Router(config-slb-vserver)# virtual 10.20.102.100 tcp www
Router(config-slb-vserver)# vlan 103
Router(config-slb-vserver)# serverfarm WEBSERVERS
Router(config-slb-vserver)# inservice
```

```
Router(config-slb-vserver)# exit
Router(config-module-csm)# vserver SSL_VIP
Router(config-slb-vserver)# virtual 10.20.102.100 tcp https
Router(config-slb-vserver)# vlan 102
Router(config-slb-vserver)# serverfarm SSLFARM
Router(config-slb-vserver)# sticky 30 group 100
Router(config-slb-vserver)# inservice
Router(config-slb-vserver)# end
```

This example shows how to configure the SSL Services Module to communicate with the CSM:

```
ssl-proxy# configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
ssl-proxy(config)# interface SSL-Proxy 0.103
ssl-proxy(config-subif)# encapsulation dot1q 103
ssl-proxy(config-subif)# ip address 10.20.102.10 255.255.255.0
ssl-proxy(config-subif)# no shutdown
ssl-proxy(config-subif)# end
```

To complete the configuration, enter the **service** command under ssl-proxy context to create a new service on the SSL Services Module (**test1**). This example shows how to configure a virtual IP address that matches the virtual server created on the CSM. (This virtual IP address is configured with the **secondary** keyword so that the SSL Services Module does not reply to ARP requests for this IP address.) The service is configured to send decrypted traffic back to the CSM without performing NAT.

```
ssl-proxy# configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
ssl-proxy(config)# ssl-proxy context Default
ssl-proxy(config-context)# service test1
ssl-proxy(config-ctx-ssl-proxy)# virtual ipaddr 10.20.102.100 protocol tcp port 443
secondary
ssl-proxy(config-ctx-ssl-proxy)# server ipaddr 10.20.102.1 protocol tcp port 80
ssl-proxy(config-ctx-ssl-proxy)# certificate rsa general-purpose trustpoint testtp
ssl-proxy(config-ctx-ssl-proxy)# no nat server
ssl-proxy(config-ctx-ssl-proxy)# inservice
ssl-proxy(config-ctx-ssl-proxy)# end
```

The following examples show the output of the various **show** commands on the MSFC and CSM:

```
Router# show module csm 5 vlan detail
vlan   IP address       IP mask          type
-----------------------------------------------------
102    10.20.102.2      255.255.255.0    CLIENT
  GATEWAYS
  10.20.102.99
103    10.20.102.2      255.255.255.0    SERVER
  ALIASES
  IP address       IP mask
  --------------------------------
  10.20.102.1      255.255.255.0
105    10.20.105.2      255.255.255.0    SERVER
  ALIASES
  IP address       IP mask
  --------------------------------
  10.20.105.1      255.255.255.0
```

```
Router# show module csm 5 vserver detail
SSL_VIP, type = SLB, state = OPERATIONAL, v_index = 13
  virtual = 10.20.102.100/32:443, TCP, service = NONE, advertise = FALSE
  idle = 3600, replicate csrp = none, vlan = 102, pending = 30
  max parse len = 600, persist rebalance = TRUE
  conns = 0, total conns = 2
  Default policy:
    server farm = SSLFARM, backup = <not assigned>
    sticky: timer = 30, subnet = 0.0.0.0, group id = 100
  Policy          Tot Conn     Client pkts  Server pkts
  ------------------------------------------------------
  (default)       2            22           15

CLEAR_VIP, type = SLB, state = OPERATIONAL, v_index = 14
  virtual = 10.20.102.100/32:80, TCP, service = NONE, advertise = FALSE
  idle = 3600, replicate csrp = none, vlan = 102, pending = 30
  max parse len = 600, persist rebalance = TRUE
  conns = 0, total conns = 0
  Default policy:
    server farm = WEBSERVERS, backup = <not assigned>
    sticky: timer = 0, subnet = 0.0.0.0, group id = 0
  Policy          Tot Conn     Client pkts  Server pkts
  ------------------------------------------------------
  (default)       0            0            0

DECRYPT_VIP, type = SLB, state = OPERATIONAL, v_index = 15
  virtual = 10.20.102.100/32:80, TCP, service = NONE, advertise = FALSE
  idle = 3600, replicate csrp = none, vlan = 103, pending = 30
  max parse len = 600, persist rebalance = TRUE
  conns = 0, total conns = 2
  Default policy:
    server farm = WEBSERVERS, backup = <not assigned>
    sticky: timer = 0, subnet = 0.0.0.0, group id = 0
  Policy          Tot Conn     Client pkts  Server pkts
  ------------------------------------------------------
  (default)       2            11           7
```

The following examples show the output of the various **show** commands on the SSL Services Module:

```
ssl-proxy# show ssl-proxy service test1
No context name provided, assuming context 'Default'…

Service id: 0, bound_service_id: 256
Virtual IP: 10.20.102.100, port: 443 (secondary configured)
Server IP: 10.20.102.1, port: 80
rsa-general-purpose certificate trustpoint: testtp
  Certificate chain in use for new connections:
    Server Certificate:
       Key Label: testtp
       Serial Number: 01
    Root CA Certificate:
      Serial Number: 00
  Certificate chain complete
Context name: Default
Context Id  : 0
Admin Status: up
Operation Status: up
ssl-proxy#
```

```
ssl-proxy# show ssl-proxy stats
TCP Statistics:
    Conns initiated   : 2             Conns accepted   : 2
    Conns established : 4             Conns dropped    : 4
    Conns closed      : 4             SYN timeouts     : 0
    Idle timeouts     : 0             Total pkts sent  : 26
    Data packets sent : 15            Data bytes sent  : 8177
    Total Pkts rcvd   : 27            Pkts rcvd in seq : 11
    Bytes rcvd in seq : 5142

SSL stats:
    conns attempted     : 2             conns completed     : 2
    full handshakes     : 2             resumed handshakes  : 0
    active conns        : 0             active sessions     : 0
    renegs attempted    : 0             conns in reneg      : 0
    handshake failures  : 0             data failures       : 0
    fatal alerts rcvd   : 0             fatal alerts sent   : 0
    no-cipher alerts    : 0             ver mismatch alerts : 0
    no-compress alerts  : 0             bad macs received   : 0
    pad errors          : 0

FDU Statistics
    IP Frag Drops     :  0              Serv_Id Drops     :  0
    Conn Id Drops     :  0              Checksum Drops    :  0
    IOS Congest Drops :  0              IP Version Drops  :  0
    Hash Full Drops   :  0              Hash Alloc Fails  :  0
    Flow Creates      :  4              Flow Deletes      :  4
    conn_id allocs    :  4              conn_id deallocs  :  4
    Tagged Drops      :  0              Non-Tagged Drops  :  0
    Add ipcs          :  0              Delete ipcs       :  0
    Disable ipcs      :  0              Enable ipcs       :  0
    Unsolicited ipcs  :  0              Duplicate ADD ipcs:  0
ssl-proxy#
```

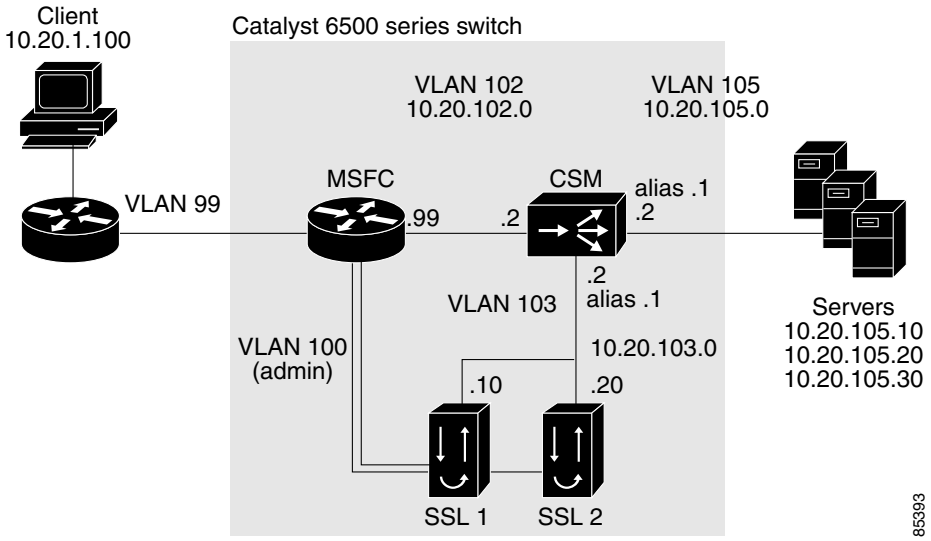
CSM and SSL Services Module Configuration Example (Router Mode, Server NAT)

This section describes a CSM and SSL Services Module configuration that contains two SSL Services Modules, a CSM, a client network, and a server farm that has three web servers (IP addresses 10.20.105.10, 10.20.105.20, and 10.20.105.30).

In this example, the three CSM VLANs (client VLAN, server VLAN for the SSL Services Modules, and server VLAN for the web servers) are configured in distinct IP subnets (router mode). (See Figure A-3.)

The CSM is configured to perform server NAT operations when it is load balancing the encrypted traffic to the SSL Services Modules. The SSL Services Modules are also configured to perform server NAT operations when they are sending decrypted traffic back to the CSM. The CSM is then configured to perform NAT on the decrypted traffic to the selected destination server.

Figure A-3 Configuration Example—Router Mode, Server NAT



The following addresses are configured on the CSM virtual servers:

- Client clear text traffic—10.20.102.100:80
- Client SSL traffic—10.20.102.100:443
- Decrypted traffic from SSL Services Modules—10.20.103.100:81

The following addresses are configured on the SSL virtual server:

- 10.20.103.110:443
- 10.20.103.120:443

In Figure A-3, VLAN 102, VLAN 103, and VLAN 105 are in separate subnets. VLAN 100 (admin) is set up as a separate VLAN for management purposes.

Add all the required VLANs to the VLAN database, and configure the IP interfaces for VLAN 100 and VLAN 102 on the MSFC. Configure VLAN 102, VLAN 103, and VLAN 105 on the CSM. See the "Initial SSL Services Module Configuration" section on page 2-2 for information on how to configure VLANs and IP interfaces.

Note While VLAN 100 and VLAN 102 exist as Layer 3 interfaces on the MSFC, both VLAN 103 and VLAN 105 exist only as VLANs in the VLAN database and as CSM VLANs, but they do not have corresponding Layer 3 interfaces on the MSFC.

This example shows how to create the client and server VLANs on the CSM installed in slot number 5:

```
Router# configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
Router(config)# module csm 5
Router(config-module-csm)# vlan 102 client
Router(config-slb-vlan-client)# ip address 10.20.102.2 255.255.255.0
Router(config-slb-vlan-client)# alias 10.20.102.1 255.255.255.0
Router(config-slb-vlan-client)# gateway 10.20.102.99
Router(config-slb-vlan-client)# exit
Router(config-module-csm)# vlan 103 server
Router(config-slb-vlan-server)# ip address 10.20.103.2 255.255.255.0
Router(config-slb-vlan-server)# alias 10.20.103.1 255.255.255.0
```

```
Router(config-slb-vlan-server)# exit
Router(config-module-csm)# vlan 105 server
Router(config-slb-vlan-server)# ip address 10.20.105.2 255.255.255.0
Router(config-slb-vlan-server)# alias 10.20.105.1 255.255.255.0
Router(config-slb-vlan-server)# end
```

This example shows how to allow VLAN 103 (client VLAN) between the SSL Services Module and the CSM, and VLAN 100 (admin VLAN) between the SSL Services Module and the MSFC:

Cisco IOS Software

```
Router(config)# ssl-proxy module 4 allowed-vlan 100,103
```

Catalyst Operating System Software

```
Console> (enable) set trunk 4/1 100,103
```

This example shows how to create the server farm of web servers (configured with server NAT) and the server farm of SSL Services Modules (configured with server NAT):

```
Router# configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
Router(config)# module csm 5
Router(config-module-csm)# serverfarm SSLFARM
Router(config-slb-sfarm)# nat server
Router(config-slb-sfarm)# real 10.20.103.110
Router(config-slb-real)# inservice
Router(config-slb-real)# real 10.20.103.120
Router(config-slb-real)# inservice
Router(config-slb-real)# exit
Router(config-slb-sfarm)# exit
Router(config-module-csm)# serverfarm WEBSERVERS
Router(config-slb-sfarm)# nat server
Router(config-slb-sfarm)# real 10.20.105.10
Router(config-slb-real)# inservice
Router(config-slb-real)# real 10.20.105.20
Router(config-slb-real)# inservice
Router(config-slb-real)# real 10.20.105.30
Router(config-slb-real)# inservice
Router(config-slb-real)# end
```

This example shows how to configure the three virtual servers. In this example, the web servers receive requests on port 80 directly from the clients, and decrypted requests on port 81 from the SSL Services Modules (since IP and port translation are configured).

This example also shows how to configure a sticky group to maintain stickiness based on the SSL ID:

```
Router# configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
Router(config)# module csm 5
Router(config-module-csm)# sticky 100 ssl timeout 30
Router(config-module-csm)# vserver CLEAR_VIP
Router(config-slb-vserver)# virtual 10.20.102.100 tcp www
Router(config-slb-vserver)# vlan 102
Router(config-slb-vserver)# serverfarm WEBSERVERS
Router(config-slb-vserver)# inservice
Router(config-slb-vserver)# exit
Router(config-module-csm)# vserver DECRYPT_VIP
Router(config-slb-vserver)# virtual 10.20.103.100 tcp 81
Router(config-slb-vserver)# vlan 103
Router(config-slb-vserver)# serverfarm WEBSERVERS
Router(config-slb-vserver)# inservice
Router(config-slb-vserver)# exit
Router(config-module-csm)# vserver SSL_VIP
```

```
Router(config-slb-vserver)# virtual 10.20.102.100 tcp https
Router(config-slb-vserver)# vlan 102
Router(config-slb-vserver)# serverfarm SSLFARM
Router(config-slb-vserver)# sticky 30 group 100
Router(config-slb-vserver)# inservice
Router(config-slb-vserver)# end
```

This example shows how to configure the SSL Services Module to communicate with the CSM over VLAN 103 and to communicate with the MSFC over VLAN 100:

```
ssl-proxy# configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
ssl-proxy(config)# interface SSL-Proxy 0.100
ssl-proxy(config-subif)# encapsulation dot1q 100
ssl-proxy(config-subif)# ip address 10.20.100.10 255.255.255.0
ssl-proxy(config-subif)# no shutdown
ssl-proxy(config-subif)# exit
ssl-proxy(config)# ip route 0.0.0.0 0.0.0.0 10.20.100.99
ssl-proxy(config)# interface SSL-Proxy 0.103
ssl-proxy(config-subif)# encapsulation dot1q 103
ssl-proxy(config-subif)# ip address 10.20.103.10 255.255.255.0
ssl-proxy(config-subif)# no shutdown
ssl-proxy(config-subif)# end
```

To complete the configuration, enter the **service** command under ssl-proxy context to create a new service on the SSL Services Module (**test1**). This example shows how to configure a virtual IP address, which acts as a real server for the CSM. (Since this virtual IP address is required to reply to ARP, the **secondary** keyword is not entered.) The service is configured to send decrypted traffic back to the CSM and to perform NAT on both the destination IP address and the port:

```
ssl-proxy# configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
ssl-proxy(config)# ssl-proxy contest Default
ssl-proxy(config-context)# service test1
ssl-proxy(config-ctx-ssl-proxy)# virtual ipaddr 10.20.103.110 protocol tcp port 443
ssl-proxy(config-ctx-ssl-proxy)# server ipaddr 10.20.103.100 protocol tcp port 81
ssl-proxy(config-ctx-ssl-proxy)# certificate rsa general-purpose trustpoint testtp
ssl-proxy(config-ctx-ssl-proxy)# nat server
ssl-proxy(config-ctx-ssl-proxy)# inservice
ssl-proxy(config-ctx-ssl-proxy)# end
```

The following examples show the output of the various **show** commands on the MSFC and CSM:

```
Router# show mod csm 5 vlan detail
vlan   IP address       IP mask          type
-----------------------------------------------------
102    10.20.102.2      255.255.255.0    CLIENT
  GATEWAYS
  10.20.102.99
  ALIASES
  IP address       IP mask
  -------------------------------
  10.20.102.1      255.255.255.0
103    10.20.103.2      255.255.255.0    SERVER
  ALIASES
  IP address       IP mask
  -------------------------------
  10.20.103.1      255.255.255.0
105    10.20.105.2      255.255.255.0    SERVER
  ALIASES
  IP address       IP mask
  -------------------------------
  10.20.105.1      255.255.255.0
```

```
Router# show mod csm 5 vser detail
CLEAR_VIP, type = SLB, state = OPERATIONAL, v_index = 10
  virtual = 10.20.102.100/32:80, TCP, service = NONE, advertise = FALSE
  idle = 3600, replicate csrp = none, vlan = 102, pending = 30
  max parse len = 600, persist rebalance = TRUE
  conns = 0, total conns = 1
  Default policy:
    server farm = WEBSERVERS, backup = <not assigned>
    sticky: timer = 0, subnet = 0.0.0.0, group id = 0
  Policy          Tot Conn     Client pkts  Server pkts
  ------------------------------------------------------
  (default)       1            6            4

DECRYPT_VIP, type = SLB, state = OPERATIONAL, v_index = 11
  virtual = 10.20.103.100/32:81, TCP, service = NONE, advertise = FALSE
  idle = 3600, replicate csrp = none, vlan = 103, pending = 30
  max parse len = 600, persist rebalance = TRUE
  conns = 0, total conns = 2
  Default policy:
    server farm = WEBSERVERS, backup = <not assigned>
    sticky: timer = 0, subnet = 0.0.0.0, group id = 0
  Policy          Tot Conn     Client pkts  Server pkts
  ------------------------------------------------------
  (default)       2            11           7

SSL_VIP, type = SLB, state = OPERATIONAL, v_index = 13
  virtual = 10.20.102.100/32:443, TCP, service = NONE, advertise = FALSE
  idle = 3600, replicate csrp = none, vlan = 102, pending = 30
  max parse len = 600, persist rebalance = TRUE
  conns = 0, total conns = 2
  Default policy:
    server farm = SSLFARM, backup = <not assigned>
    sticky: timer = 30, subnet = 0.0.0.0, group id = 100
  Policy          Tot Conn     Client pkts  Server pkts
  ------------------------------------------------------
  (default)       2            21           15
```

The following examples show the output of the various **show** commands on the SSL Services Module:

```
ssl-proxy# show ssl-proxy service test1
No context name provided, assuming context 'Default'…

Service id: 0, bound_service_id: 256
Virtual IP: 10.20.103.110, port: 443
Server IP: 10.20.103.100, port: 81
rsa-general-purpose certificate trustpoint: testtp
  Certificate chain in use for new connections:
    Server Certificate:
       Key Label: testtp
       Serial Number: 01
    Root CA Certificate:
      Serial Number: 00
  Certificate chain complete
Context name: Default
Context Id  : 0
Admin Status: up
Operation Status: up
ssl-proxy#
```

```
ssl-proxy# show ssl-proxy stats
TCP Statistics:
    Conns initiated   : 2             Conns accepted   : 2
    Conns established : 4             Conns dropped    : 4
    Conns closed      : 4             SYN timeouts     : 0
    Idle timeouts     : 0             Total pkts sent  : 26
    Data packets sent : 15            Data bytes sent  : 8212
    Total Pkts rcvd   : 26            Pkts rcvd in seq : 11
    Bytes rcvd in seq : 5177

SSL stats:
    conns attempted     : 2             conns completed     : 2
    full handshakes     : 2             resumed handshakes  : 0
    active conns        : 0             active sessions     : 0
    renegs attempted    : 0             conns in reneg      : 0
    handshake failures  : 0             data failures       : 0
    fatal alerts rcvd   : 0             fatal alerts sent   : 0
    no-cipher alerts    : 0             ver mismatch alerts : 0
    no-compress alerts  : 0             bad macs received   : 0
    pad errors          : 0

FDU Statistics
    IP Frag Drops     :  0              Serv_Id Drops     :  0
    Conn Id Drops     :  0              Checksum Drops    :  0
    IOS Congest Drops :  0              IP Version Drops  :  0
    Hash Full Drops   :  0              Hash Alloc Fails  :  0
    Flow Creates      :  4              Flow Deletes      :  4
    conn_id allocs    :  4              conn_id deallocs  :  4
    Tagged Drops      :  0              Non-Tagged Drops  :  0
    Add ipcs          :  0              Delete ipcs       :  0
    Disable ipcs      :  0              Enable ipcs       :  0
    Unsolicited ipcs  :  0              Duplicate ADD ipcs:  0
```

Basic Backend Encryption Example

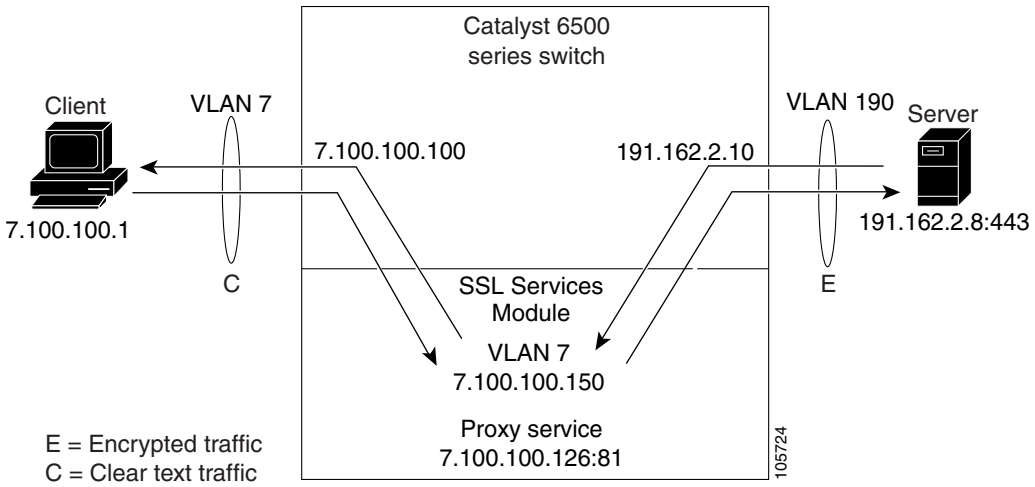
Backend encryption allows you to create a secure end-to-end environment. This example shows a basic backend encryption configuration.

In Figure A-4, the client (7.100.100.1) is connected to switchport 6/47 in access VLAN 7. The server (191.162.2.8) is connected to switchport 10/2 in access VLAN 190.

The SSL proxy 0.7 subinterface has the following configuration:

- IP address—7.100.100.150
- Static route—191.0.0.0
- Virtual IP address of client proxy service—7.100.100.150:81
- Server IP address—191.162.2.8

Figure A-4 Basic Backend Encryption



Configuring VLANS and Switchports

These examples show how to create VLANs and assign ports to the respective VLANs:

Cisco IOS Software

```
Router# configure terminal
Router(config)# vlan 7
Router(config-vlan)# exit
Router(config)# vlan 190
Router(config-vlan)# exit
Router(config)# interface FastEthernet6/47
Router(config-if)# switchport
Router(config-if)# switchport access vlan 7
Router(config-if)# switchport mode access
Router(config-if)# exit
Router(config)#
Router(config)# interface GigabitEthernet10/2
Router(config-if)# switchport
Router(config-if)# switchport access vlan 190
Router(config-if)# switchport mode access
Router(config-if)# exit
Router(config)#
```

Catalyst Operating System

```
Console> (enable) set vlan 7
VTP advertisements transmitting temporarily stopped,
and will resume after the command finishes.
Vlan 7 configuration successful
Console> (enable)
Console> (enable) set vlan 190
VTP advertisements transmitting temporarily stopped,
and will resume after the command finishes.
Vlan 190 configuration successful
Console> (enable)
```

```
Console> (enable) set vlan 7 6/47
VLAN  Mod/Ports
---- ------------------------
7     6/47
Console> (enable) set vlan 190 10/2
VLAN  Mod/Ports
---- ------------------------
190   10/2
Console> (enable)
```

Configuring the Allowed VLANs

This example shows how to allow VLAN 7 between SSL module in slot 12 and the supervisor engine:

Cisco IOS Software

```
Router# configure terminal
Router(config)# ssl-proxy module 12 allowed-vlan 7
Router(config)# exit
Router#

Router# show ssl-proxy mod 12 state
Router module 12 data-port:

Switchport:Enabled
Administrative Mode:trunk
Operational Mode:trunk
Administrative Trunking Encapsulation:dot1q
Operational Trunking Encapsulation:dot1q
Negotiation of Trunking:Off
Access Mode VLAN:1 (default)
Trunking Native Mode VLAN:1 (default)
Trunking VLANs Enabled:7
Pruning VLANs Enabled:2-1001
Vlans allowed on trunk:7
Vlans allowed and active in management domain:7
Vlans in spanning tree forwarding state and not pruned:
   7
Allowed-vlan :7

Router#
```

Catalyst Operating System

```
Console> (enable) show mod 12
Mod Slot Ports Module-Type               Model               Sub Status
--- ---- ----- ------------------------- ------------------- --- --------
12  12   1     SSL Module                WS-SVC-SSL-1        no  ok

Mod Module-Name          Serial-Num
--- -------------------- -----------
12                       SAD062004N0

Mod MAC-Address(es)                        Hw     Fw          Sw
--- -------------------------------------- ------ ----------- ------------------
12  00-e0-b0-ff-f0-c2                      0.305  7.2(1)      2.1(1)
Console> (enable)
Console> (enable) set trunk 12/1 7
Adding vlans 7 to allowed list.
Port(s) 12/1 allowed vlans modified to 7.
Console> (enable)
```

```
Console> (enable) show trunk 12/1
* - indicates vtp domain mismatch
# - indicates dot1q-all-tagged enabled on the port
$ - indicates non-default dot1q-ethertype value
Port      Mode         Encapsulation  Status        Native vlan
--------  -----------  -------------  ------------  -----------
12/1      nonegotiate  dot1q          trunking      1

Port      Vlans allowed on trunk
--------  ---------------------------------------------------------------------
12/1      7,190

Port      Vlans allowed and active in management domain
--------  ---------------------------------------------------------------------
12/1      7,190

Port      Vlans in spanning tree forwarding state and not pruned
--------  ---------------------------------------------------------------------
12/1      7,190
Console> (enable)
```

Configuring the Access List and Route Map

This example shows how to configure the access list and route map for redirecting SSL traffic from the server to the SSL Services Module and for redirecting clear text traffic from the client to the SSL Services Module:

```
Router(config)# ip access-list extended client
Router(config-ext-nacl)# permit tcp any host 7.100.100.126 eq 81
Router(config-ext-nacl)# exit
Router(config)#
Router(config)# ip access-list extended server
Router(config-ext-nacl)# permit tcp host 191.162.2.8 eq 443 any
Router(config-ext-nacl)# exit
Router(config)#
Router(config)# route-map server permit 10
Router(config-route-map)# match ip address server
Router(config-route-map)# set ip next-hop 7.100.100.150
Router(config-route-map)# exit
Router(config)#
Router(config)# route-map client permit 10
Router(config-route-map)# match ip address client
Router(config-route-map)# set ip next-hop 7.100.100.150
Router(config-route-map)# exit
Router(config)#
Router(config)# interface Vlan7
Router(config-if)# ip address 7.100.100.100 255.0.0.0
Router(config-if)# ip policy route-map client
Router(config-if)# end
Router#
Router# configure terminal
Router(config)# interface Vlan190
Router(config-if)# ip address 191.162.2.10 255.0.0.0
Router(config-if)# ip policy route-map server
Router(config-if)# end
```

Configuring the SSL Proxy Subinterface

This example shows how to add an interface to VLAN 7 on the SSL Services Module:

```
ssl-proxy# configure terminal
ssl-proxy(config)# interface ssl-proxy0.7
ssl-proxy(config-subif)# encapsulation dot1q 7
ssl-proxy(config-subif)# ip address 7.100.100.150 255.0.0.0
ssl-proxy(config-subif)# no shutdown
ssl-proxy(config-subif)# exit
ssl-proxy(config)# ip route 191.0.0.0 255.0.0.0 7.100.100.100
```

Configuring the Root Certificate Authority Trustpoint for Server Certificate Authentication

This example shows how to configure root certificate authority trustpoint for authenticating server certificates. See the "Server Certificate Authentication" section on page 3-55 for information on configuring server certificate authentication.

```
ssl-proxy(config)# crypto pki trustpoint root
ssl-proxy(ca-trustpoint)# enrollment terminal
ssl-proxy(ca-trustpoint)# exit
ssl-proxy(config)# crypto pki auth root

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

-----BEGIN CERTIFICATE-----
MIICSTCCAfOgAwIBAgIBATANBgkqhkiG9w0BAQUFADBxMQswCQYDVQQGEwJVUzET
MBEGA1UECBMKQ2FsaWZvcm5pYTERMA8GA1UEBxMIU2FuIEpvc2UxDjAMBgNVBAoT
BUNpc2NvMQwwCgYDVQQLEwNIU1MxHDAaBgNVBAMTE0NlcnRpZmljYXRlIE1hbmFn
ZXIwHhcNMDIwNDI3MDcwMDAwWhcNMDYwNDI3MDcwMDAwWjBxMQswCQYDVQQGEwJV
UzETMBEGA1UECBMKQ2FsaWZvcm5pYTERMA8GA1UEBxMIU2FuIEpvc2UxDjAMBgNV
BAoTBUNpc2NvMQwwCgYDVQQLEwNIU1MxHDAaBgNVBAMTE0NlcnRpZmljYXRlIE1h
bmFnZXIwXDANBgkqhkiG9w0BAQEFAANLADBIAkEAvadjzN/Z3SEe7dBktvG6U1/W
hoe1/vJ6uIV/goS/fwYnr0+Gk1Lw2DGEMN9P1li5qcM2Rgq6E+6AQv2UYerBjQID
AQABo3YwdDARBglghkgBhvhCAQEEBAMCAAcwDwYDVR0TAQH/BAUwAwEB/zAdBgNV
HQ4EFgQU7u9bvU3N9Wtgnc9GwuoleyKlCAAwHwYDVR0jBBgwFoAU7u9bvU3N9Wtg
nc9GwuoleyKlCAAwDgYDVR0PAQH/BAQDAgGGMA0GCSqGSIb3DQEBBQUAA0EAhYZX
upIv+ZRo639io4ZLaiCWePHA+6Oz2n4B1kX9Jqx9fS3Zbyc712BP6lM2Apk5fhBs
6z/WrDRR0GZhlOoAvg==
-----END CERTIFICATE-----
quit
Certificate has the following attributes:
Fingerprint:683F909E 0B9F1651 7AAB8E36 14DBE45F
% Do you accept this certificate? [yes/no]:yes
Trustpoint CA certificate accepted.
% Certificate successfully imported
ssl-proxy(config)# ssl-proxy context Default
ssl-proxy(config-context)# pool ca root-ca
ssl-proxy(config-ctx-ca-pool)# ca trustpoint root
ssl-proxy(config-ctx-ca-pool)# exit
```

Configuring the SSL Proxy Service

This example shows how to configure the SSL client proxy service to accept clear text connections to virtual IP address 7.100.100.126 with TCP port 81 and to initiate an SSL connection to the backend SSL server IP address 191.162.2.8 with destination TCP port 443. See the "SSL Client Proxy Services" section on page 3-48 for information on configuring client proxy services.

```
ssl-proxy(config)# ssl-proxy context Default
ssl-proxy(config-context)# service backend-ssl client
ssl-proxy(config-ctx-ssl-proxy)# virtual ipaddr 7.100.100.126 protocol tcp port 81
ssl-proxy(config-ctx-ssl-proxy)# server ipaddr 191.162.2.8 protocol tcp port 443
ssl-proxy(config-ctx-ssl-proxy)# trusted-ca root-ca
ssl-proxy(config-ctx-ssl-proxy)# authenticate verify all
ssl-proxy(config-ctx-ssl-proxy)# inservice
ssl-proxy(config-ctx-ssl-proxy)# ^Z
ssl-proxy#
```

Verifying Service and Connections

This example shows the successful initiation of the SSL connection to the backend SSL server:

```
ssl-proxy# show ssl-proxy stats ssl
SSL Statistics:
    conns attempted     :5              conns completed     :5
    conns in handshake  :0              conns in data       :0
    renegs attempted    :0              conns in reneg      :0
    active sessions     :0              max handshake conns :1
    rand bufs allocated :1              cached rand buf miss:0
    current device q len:0              max device q len    :1
    sslv2 forwards      :0              cert reqs processed :0
    fatal alerts rcvd   :0              fatal alerts sent   :0
    stale packet drops  :0              service_id discards :0
    session reuses      :0              hs handle in use    :0

    SSL3 Statistics:
      full handshakes    :0              resumed handshakes :0
      handshake failures :0              data failures      :0
      bad macs received  :0              pad errors         :0
      conns established with cipher rsa-with-rc4-128-md5        :0
      conns established with cipher rsa-with-rc4-128-sha        :0
      conns established with cipher rsa-with-des-cbc-sha        :0
      conns established with cipher rsa-with-3des-ede-cbc-sha   :0

    TLS1 Statistics:
      full handshakes    :3              resumed handshakes :2
      handshake failures :0              data failures      :0
      bad macs received  :0              pad errors         :0
      conns established with cipher rsa-with-rc4-128-md5        :5
      conns established with cipher rsa-with-rc4-128-sha        :0
      conns established with cipher rsa-with-des-cbc-sha        :0
      conns established with cipher rsa-with-3des-ede-cbc-sha   :0

SSL error statistics:
    session alloc fails :0              session limit exceed:0
    handshake init fails:0              renegotiation fails :0
    no-cipher alerts    :0              ver mismatch alerts :0
    no-compress alerts  :0              multi buf rec errors:0
    ssl peer closes     :0              non-ssl peer closes :0
    unexpected record   :0              rec formatting error:0
    rsa pkcs pad errors :0              premaster errors    :0
    failed rsa reqs     :0              failed random reqs  :0
```

```
    failed key-material :0            failed master-secret:0
    failed update hash  :0            failed finish hash  :0
    failed encrypts     :0            failed decrypts     :0
    bad record version  :0            bad record size     :0
    cert verify errors  :0            unsupported certs   :0
    conn aborted        :0
    overload drops      :0            hs limit exceeded   :0
    hs handle mem fails :0            conn reuse error    :0
    dev invalid params  :0            dev failed requests :0
    dev timeout         :0            dev busy            :0
    dev cancelled       :0            no dev fails        :0
    dev resource fails  :0            dev unknown errors  :0
    dev conn ctx fails  :0            dev cmd ctx fails   :0
    mem alloc fails     :0            buf alloc fails     :0
    invalid cipher algo :0            invalid hash algo   :0
    unaligned buf addr  :0            unaligned buf len   :0
    internal error      :0            unknown ipcs        :0
    double free attempts:0            alert-send fails    :0

SSL Crypto Statistics:
    blocks encrypted    :20           blocks decrypted    :249
    bytes encrypted     :4898         bytes decrypted     :25194
    rsa public key ops  :7            rsa private key ops :4
    crypto failures     :0            device dma errors   :0

SSL last 5 sec average Statistics:
    full handshakes     :0            resumed handshakes  :0
    handshake failures  :0            data failures       :0
    bytes encrypted     :0            bytes decrypted     :0

SSL last 1 min average Statistics:
    full handshakes     :0            resumed handshakes  :0
    handshake failures  :0            data failures       :0
    bytes encrypted     :0            bytes decrypted     :0

SSL last 5 min average Statistics:
    full handshakes     :0            resumed handshakes  :0
    handshake failures  :0            data failures       :0
    bytes encrypted     :0            bytes decrypted     :0

SSL PKI Statistics:
    number of malloc    :224          number of free      :209
    ssl buf allocated   :12           ssl buf freed       :8

    Peer Certificate Verify Statistics:
    cert approved       :3            cert disapproved    :0
    peer cert empty     :0            total num of request:3
    req being processed :0            req pending         :0
    longest queue       :1            longest pending     :0
    verify congestion   :0            req dropped, q full :0
    no memory for verify:0            verify data error   :0
    verify context error:0            context delete error:0
    timer expired error :0            timer expired count :0
    late verify result  :0            timer turned on     :3
    timer turned off    :3            context created     :3
    context deleted     :3
```

```
    High Priority IPC:
    ipc request received:23            ipc request dropped :0
    ipc req duplicated  :0             ipc req fragment err:0
    ipc req parm len err:0             ipc req op code err :0
    ipc req cert len err:0             ipc response sent   :23
    ipc resp no memory  :0             ipc resp no ssl buf :0
    ipc buffer allocated:0             ipc buffer freed    :0
    ipc buf alloc failed:0             ipc send msg failed :0

    Normal Priority IPC:
    ipc buffer allocated:3             ipc buffer freed    :3
    ipc request sent    :3             ipc request received:3
    ipc buf alloc failed:0             ipc send msg failed :0
    ipc requests dropped:0

ssl-proxy#
```

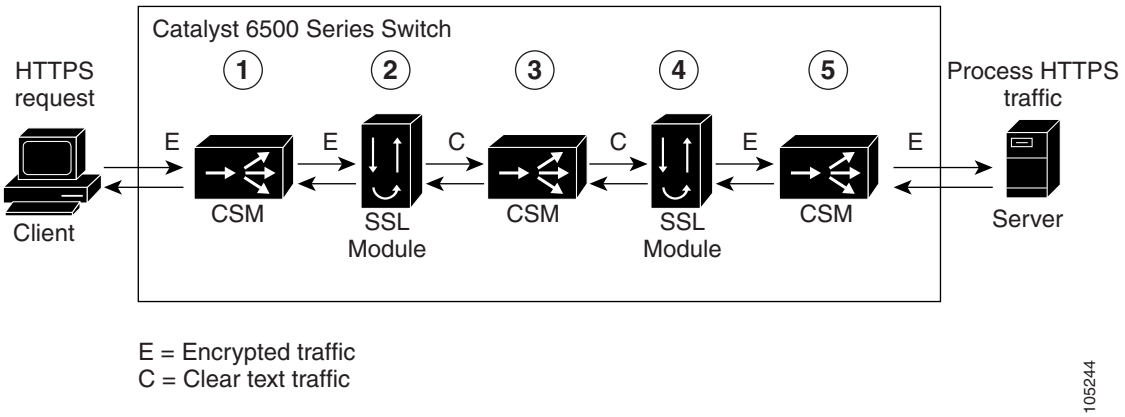
Integrated Secure Content-Switching Service Example

Configuring an integrated secure content-switching service (using a content switching module [CSM] as a server load balancer) with backend encryption has all the benefits of load-balancing and content switching, while securing data with full SSL coverage as it traverses paths of vulnerability.

As shown in Figure A-5, an integrated secure content-switching service configuration involves five processing steps:

1. The CSM load-balances the SSL traffic, based on either load-balancing rules or using the SSL sticky feature (see the "Sticky Connections" section on page 5-7 for information on configuring sticky connections), to an SSL Services Module.
2. The SSL Services Module terminates the SSL session, decrypts the SSL traffic into clear text traffic, and forwards the traffic back to the CSM.
3. The CSM content-switches the clear text traffic to the SSL Services Module again for encryption to SSL traffic.
4. The SSL Services Module forwards the encrypted SSL traffic to the CSM.
5. The CSM forwards the SSL traffic to the HTTPS server.

Figure A-5 Backend Encryption Example—Integrated Secure Content-Switching Service



Configuring the CSM

This example shows how to configure the VLANs on the CSM. VLAN 24 is the VLAN through which client traffic arrives. VLAN 35 is the VLAN between the SSL Services Module and the CSM.

```
Router# conf t
Enter configuration commands, one per line.  End with CNTL/Z.
Router(config)# module ContentSwitchingModule 6
Router(config-module-csm)# vlan 24 client
Route(config-slb-vlan-client)# ip address 24.24.24.24 255.0.0.0
Route(config-slb-vlan-client)# vlan 35 server
Route(config-slb-vlan-server)# ip address 35.35.35.35 255.0.0.0
Route(config-slb-vlan-server)# route 36.0.0.0 255.0.0.0 gateway 35.200.200.3
```

This example shows how to configure the URL policy for Layer 7 parsing:

```
Route(config-slb-vlan-server)# map URL url
Router(config-slb-map-url)# match protocol http method GET url /*
```

This example shows how to create server farms:

```
Router(config-slb-map-url)# serverfarm SSLCARDS
Router(config-slb-sfarm)# real 35.200.200.101
Router(config-slb-real)# inservice

Router(config-slb-real)# serverfarm VLAN36REALS
Router(config-slb-sfarm)# real 36.200.200.14
Router(config-slb-real)# inservice
Router(config-slb-real)# real 36.200.200.5
Router(config-slb-real)# inservice
```

This example shows how to create the virtual servers:

```
Router(config-slb-real)# vserver LB-HTTP-SSLMODS
Router(config-slb-vserver)# virtual 35.35.35.25 tcp 81
Router(config-slb-vserver)# vlan 35
Router(config-slb-vserver)# slb-policy URL
Router(config-slb-vserver)# inservice

Router(config-slb-vserver)# vserver LB-SSL-SSLMODS
Router(config-slb-vserver)# virtual 24.24.24.25 tcp https
Router(config-slb-vserver)# serverfarm SSLCARDS
Router(config-slb-vserver)# inservice
```

This example shows how to display the status of the real servers and virtual servers:

```
Router# sh module contentSwitchingModule all reals
---------------------- CSM in slot 6 ----------------------

real                  server farm      weight  state          conns/hits
------------------------------------------------------------------------
35.200.200.101        SSLCARDS         8       OPERATIONAL    0
36.200.200.14         VLAN36REALS      8       OPERATIONAL    0
36.200.200.5          VLAN36REALS      8       OPERATIONAL    0

Router# sh module contentSwitchingModule all vservers
---------------------- CSM in slot 6 ----------------------

vserver         type  prot virtual                 vlan state        conns
------------------------------------------------------------------------
LB-HTTP-SSLMODS SLB   TCP  35.35.35.25/32:81       35   OPERATIONAL  0
LB-SSL-SSLMODS  SLB   TCP  24.24.24.25/32:443      ALL  OPERATIONAL  0

Router#
```

Configuring the SSL Services Module

This example shows how to configure the SSL Services Module to communicate with the CSM over VLAN 35:

```
ssl-proxy(config)# interface SSL-Proxy 0.35
ssl-proxy(config-subif)# encapsulation dot1q 35
ssl-proxy(config-subif)# ip address 35.200.200.3 255.0.0.0
ssl-proxy(config-subif)# no shutdown
```

This example shows how to configure a trusted certificate authority pool on the SSL Services Module:

```
ssl-proxy(config)# ssl-proxy context Default
ssl-proxy(config-context)# pool ca net
ssl-proxy(config-ctx-ca-pool)# ca trustpoint keon-root
ssl-proxy(config-ctx-ca-pool)# ca trustpoint net-root
ssl-proxy(config-ctx-ca-pool)# ca trustpoint TP-1024-pcks12-root
```

This example shows how to configure a URL rewrite policy on the SSL Services Module:

```
ssl-proxy(config)# ssl-proxy context Default
ssl-proxy(config-context)# policy url-rewrite frontend
ssl-proxy(config-ctx-url-rewrite-policy)# url www.cisco.com clearport 80 sslport 443
ssl-proxy(config-ctx-url-rewrite-policy)# url wwwin.cisco.com clearport 80 sslport 443
ssl-proxy(config-ctx-url-rewrite-policy)# url wwwin.cisco.com clearport 81 sslport 443
```

This example shows how to configure the SSL server proxy that accepts client traffic coming through the CSM. This example also shows how to configure client authentication, SSL v2.0 forwarding, and URL rewrite policy.

Note For SSL V2.0 connections, the SSL Services Module directly opens a connection to SSL Services Module instead of giving it back to CSM.

```
ssl-proxy(config)# ssl-proxy context Default
ssl-proxy(config-context)# service frontend
ssl-proxy(config-ctx-ssl-proxy)# virtual ipaddr 35.200.200.101 protocol tcp port 443
ssl-proxy(config-ctx-ssl-proxy)# server ipaddr 35.35.35.25 protocol tcp port 81
ssl-proxy(config-ctx-ssl-proxy)# server ipaddr 35.200.200.14 protocol tcp port 443 sslv2
ssl-proxy(config-ctx-ssl-proxy)# certificate rsa general-purpose trustpoint TP-1024-pkcs12
ssl-proxy(config-ctx-ssl-proxy)# policy url-rewrite frontend
ssl-proxy(config-ctx-ssl-proxy)# trusted-ca net
ssl-proxy(config-ctx-ssl-proxy)# authenticate verify all
ssl-proxy(config-ctx-ssl-proxy)# inservice
```

This example shows how to configure the SSL client proxy that accepts clear text traffic from the CSM after the traffic completes Layer 7 parsing and decides the real server. This example also shows how to configure client certificates and a wildcard proxy.

Note The gateway address (35.200.200.125) is the address through which the real servers (36.200.200.14 and 36.200.200.5) are reached.

```
ssl-proxy(config-ctx-ssl-proxy)# service wildcard client
ssl-proxy(config-ctx-ssl-proxy)# virtual ipaddr 0.0.0.0 0.0.0.0 protocol tcp port 81
secondary
ssl-proxy(config-ctx-ssl-proxy)# server ipaddr 35.200.200.125 protocol tcp port 443
ssl-proxy(config-ctx-ssl-proxy)# certificate rsa general-purpose trustpoint client-cert
ssl-proxy(config-ctx-ssl-proxy)# no nat server
ssl-proxy(config-ctx-ssl-proxy)# trusted-ca net
ssl-proxy(config-ctx-ssl-proxy)# authenticate verify all
ssl-proxy(config-ctx-ssl-proxy)# inservice
ssl-proxy(config-ctx-ssl-proxy)# ^Z
```

This example shows how to display the status of the SSL server proxy service:

```
ssl-proxy# show ssl-proxy service frontend
No context name provided, assuming context 'Default'…
Service id: 2, bound_service_id: 258
Virtual IP: 35.200.200.101, port: 443
Server IP: 35.35.35.25, port: 81
SSLv2  IP: 35.200.200.14, port: 443
URL Rewrite Policy: frontend
Certificate authority pool: net
  CA pool complete
rsa-general-purpose certificate trustpoint: TP-1024-pkcs12
  Certificate chain for new connections:
    Certificate:
       Key Label: TP-1024-pkcs12, 1024-bit, not exportable
       Key Timestamp: 22:53:16 UTC Mar 14 2003
       Serial Number: 3C2CD2330001000000DB
    Root CA Certificate:
       Serial Number: 313AD6510D25ABAE4626E96305511AC4
  Certificate chain complete
Certificate authentication type: All attributes (like CRL) are verified
Context name: Default
Context Id  : 0
Admin Status: up
Operation Status: up
ssl-proxy#
```

This example shows how to display status of the SSL client proxy service:

```
ssl-proxy# show ssl-proxy service wildcard
No context name provided, assuming context 'Default'…

Service id: 267, bound_service_id: 11
Virtual IP: 0.0.0.0, port: 81 (secondary configured)
Virtual IP mask: 0.0.0.0
Server IP: 35.200.200.125, port: 443
Certificate authority pool: net
  CA pool complete
rsa-general-purpose certificate trustpoint: client-cert
  Certificate chain for new connections:
    Certificate:
       Key Label: client-cert, 1024-bit, not exportable
       Key Timestamp: 18:42:01 UTC Jul 14 2003
       Serial Number: 04
    Root CA Certificate:
       Serial Number: 01
  Certificate chain complete
Certificate authentication type: All attributes (like CRL) are verified
Context name: Default
Context Id  : 0
Admin Status: up
Operation Status: up
ssl-proxy#
```

Site-To-Site Transport Layer VPN Example

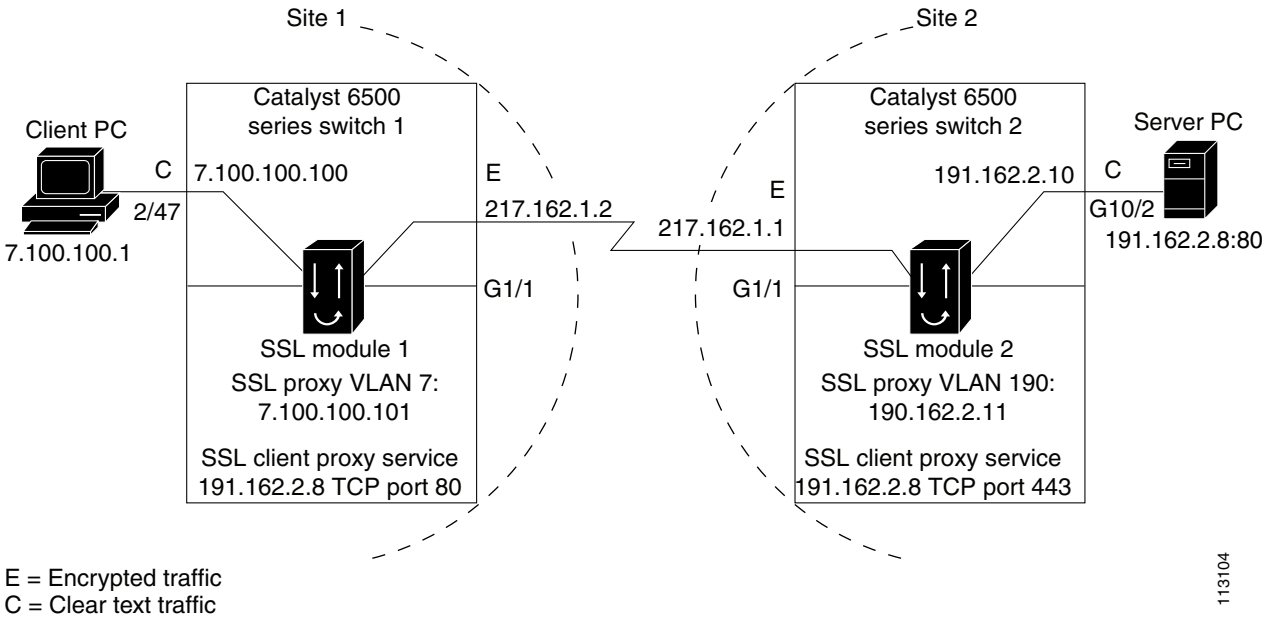
A site-to-site transport layer VPN configuration connects two trusted sites to support TCP-based applications.

In Figure A-6, SSL module 1 is configured with a client proxy service. SSL module 1 encrypts the client clear text traffic into SSL traffic and forwards the encrypted SSL traffic to SSL module 2 on the remote site through a backend SSL session. SSL module 2 is configured with a standard SSL offload virtual service, which decrypts the received SSL traffic into clear text and forwards it to the servers on the remote site.

When you configure a proxy service as either clear text-to-encryption or encryption-to-clear text mode, the proxy service acts in an SSL client role while communicating with the secure backend server. You need to configure SSL policies to describe the SSL client and the backend SSL session. See the "Configuring SSL Policy" section on page 4-2 for information on configuring SSL policies.

This section gives an example of how to tunnel HTTP traffic from the client to the server and back to the client through an SSL VPN.

Figure A-6 Backend Encryption Example—Site-to-Site Transport Layer VPN



In Figure A-6, Site 1 and Site 2 are connected by Gigabit Ethernet; however, both sites could also be connected through the Internet.

The client PC (7.100.100.1) is connected to switchport 2/47 in access VLAN 7. The server (191.162.2.8) is connected to switchport 10/2 in access VLAN 190.

Site 1 Configuration

Site 1 in Figure A-6 shows the SSL Services Module (SSL module 1) installed in slot 13 in Catalyst 6500 series switch 1.

The following example shows how to add a VLAN between the SSL Services Module and the supervisor engine:

```
cat6k-router-1# show mod 13
Mod Ports Card Type                              Model              Serial No.
--- ----- -------------------------------------- ------------------ -----------
 13    1  SSL Module                             WS-SVC-SSL-1       SAD062503FZ

Mod MAC addresses                       Hw    Fw           Sw           Status
--- ----------------------------------- ------ ------------ ------------ -------
 13  0010.7b00.0e00 to 0010.7b00.0e07   0.304 7.2(1)       2.1(1)       Ok

Mod Online Diag Status
--- -------------------
 13 Pass

cat6k-router-1# configure terminal
cat6k-router-1(config)# ssl-proxy module 13 allowed-vlan 7
cat6k-router-1(config)# exit
```

The following example shows to configure the VLAN, configure a port as a switchport, and assign the switchport to the access VLAN:

```
cat6k-router-1# configure terminal
cat6k-router-1(config)# vlan 7
cat6k-router-1(config-vlan)# exit
cat6k-router-1(config)# interface FastEthernet2/47
cat6k-router-1(config-if)# switchport
cat6k-router-1(config-if)# switchport access vlan 7
cat6k-router-1(config-if)# switchport mode access
cat6k-router-1(config-if)# exit
cat6k-router-1(config)#
```

The following examples show how to configure extended access lists:

- Access list “client” is used to match any traffic going to IP address 191.162.2.8 with destination TCP port 80 (HTTP traffic).

  ```
  cat6k-router-1(config)# ip access-list extended client
  cat6k-router-1(config-ext-nacl)# permit tcp any host 191.162.2.8 eq www
  cat6k-router-1(config-ext-nacl)# exit
  ```

- Access list “server” is used to match any traffic from IP address 191.162.2.8 with source TCP port 443 (encrypted traffic from site 2).

  ```
  cat6k-router-1(config)# ip access-list extended server
  cat6k-router-1(config-ext-nacl)# permit tcp host 191.162.2.8 eq 443 any
  cat6k-router-1(config-ext-nacl)# exit
  ```

The following examples show how to configure route maps to redirect traffic to the SSL Services Module for encryption and decryption:

- Route map "client" redirects the traffic that matches access-list "client" to the next hop IP address 7.100.100.101 (the IP address of SSL proxy subinterface 0.7 on SSL-module-1).

```
cat6k-router-1(config)# route-map client permit 10
cat6k-router-1(config-route-map)# match ip address client
cat6k-router-1(config-route-map)# set ip next-hop 7.100.100.101
cat6k-router-1(config-route-map)# exit
```

- Route map "server" redirects the traffic that matches access-list "server" to the next hop IP address 7.100.100.101 (the IP address of SSL proxy subinterface 0.7 on SSL-module-1).

```
cat6k-router-1(config)# route-map server permit 10
cat6k-router-1(config-route-map)# match ip address server
cat6k-router-1(config-route-map)# set ip next-hop 7.100.100.101
cat6k-router-1(config-route-map)# exit
```

The following example shows how to configure the routed interface and assign the route map:

```
cat6k-router-1(config)# interface Vlan7
cat6k-router-1(config-if)# ip address 7.100.100.100 255.0.0.0
cat6k-router-1(config-if)# ip policy route-map client
cat6k-router-1(config-if)# exit
cat6k-router-1(config)# interface GigabitEthernet1/1
cat6k-router-1(config-if)# ip address 217.162.1.2 255.255.255.0
cat6k-router-1(config-if)# ip policy route-map server
cat6k-router-1(config-if)# exit
```

SSL Module 1 Configuration

The following examples show how to configure the SSL client proxy service. The client proxy service is configured with virtual IP address 191.162.2.8, TCP port 80. The server is configured with IP address 7.100.100.100 so that server-side traffic is sent to 7.100.100.100 for further routing without changing the server IP address. See the "SSL Client Proxy Services" section on page 3-48 for information on configuring client proxy services. See the "Server Certificate Authentication" section on page 3-55 for more information on authenticating server certificates.

```
ssl-module1# configure terminal
ssl-module1(config)# ssl-proxy context Default
ssl-module1(config-context)# service encrypt-clear-text client
ssl-module1(config-ctx-ssl-proxy)# virtual ipaddr 191.162.2.8 protocol tcp port 80
secondary
ssl-module1(config-ctx-ssl-proxy)# server ipaddr 7.100.100.100 protocol tcp port 443
ssl-module1(config-ctx-ssl-proxy)# certificate rsa general-purpose trustpoint cert2048
ssl-module1(config-ctx-ssl-proxy)# no nat server
ssl-module1(config-ctx-ssl-proxy)# trusted-ca root-ca
ssl-module1(config-ctx-ssl-proxy)# authenticate verify all
ssl-module1(config-ctx-ssl-proxy)# inservice
ssl-module1(config-ctx-ssl-proxy)# exit
ssl-module1(config-context)# exit
ssl-module1(config)#
```

The following example shows how to configure the SSL proxy subinterface on the SSL Services Module:

```
ssl-module1(config)# interface SSL-Proxy 0.7
ssl-module1(config-subif)# encapsulation dot1q 7
ssl-module1(config-subif)# ip address 7.100.100.101 255.0.0.0
ssl-module1(config-subif)# no shutdown
ssl-module1(config-subif)# exit
```

The following example shows how to import the root-ca certificate to the SSL Services Module:

```
ssl-module1(config)# crypto pki trustpoint root-ca
ssl-module1(ca-trustpoint)# enroll terminal
ssl-module1(ca-trustpoint)# exit
ssl-module1(config)# crypto pki auth root-ca

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

-----BEGIN CERTIFICATE-----
MIIC1zCCAoGgAwIBAgIQadUxzU/i97hDmZRYJ1bBcDANBgkqhkiG9w0BAQUFADB1
MQswCQYDVQQGEwJVUzETMBEGA1UECBMKY2FsaWZvcm5pYTERMA8GA1UEBxMIc2Fu
IGpvc2UxDjAMBgNVBAoTBWNpc2NvMQwwCgYDVQQLEwNoc3MxIDAeBgNVBAMTF3Np
bXBzb24tZGV2dGVzdC1yb290LUNBMB4XDTAzMTExMTIxNDgwMloXDTEzMTExMTIx
NTczOVowdTELMAkGA1UEBhMCVVMxEzARBgNVBAgTCmNhbGlmb3JuaWExETAPBgNV
BAcTCHNhbiBqb3NlMQ4wDAYDVQQKEwVjaXNjbzEMMAoGA1UECxMDaHNzMSAwHgYD
VQQDExdzaW1wc29uLWRldnRlc3Qtcm9vdC1DQTBcMA0GCSqGSIb3DQEBAQUAA0sA
MEgCQQCWEibAnU1VqQNUn0Wb94qnHi8FKjmVhibLHGRl6J+V7gHgzmF2MTz5WP5l
VQ2/1NVu0HjUORRdeCm1/raKJ/7ZAgMBAAGjgewwgekwCwYDVR0PBAQDAgHGMA8G
A1UdEwEB/wQFMAMBAf8wHQYDVR0OBBYEFCYGLUBTKNd9EgUonHnoSvbHg0axMIGX
BgNVHR8EgY8wgYwwQ6BBoD+GPWh0dHA6Ly9jaXNjby1sOGo2b2hwbnIvQ2VydEVu
cm9sbC9zaW1wc29uLWRldnRlc3Qtcm9vdC1DQS5jcmwwRaBDoEGGP2ZpbGU6Ly9c
XGNpc2NvLWw4ajZvaHBuclxDZXJ0RW5yb2xsXHNpbXBzb24tZGV2dGVzdC1yb290
LUNBLmNybDAQBgkrBgEEAYI3FQEEAwIBADANBgkqhkiG9w0BAQUFAANBACBqe1wy
YjalelGZqLVu4bDVMFo6ELCV2AMBgi41K3ix+Z/03PJd7ct2BIAF4lktv9pCe6IO
EoBcmZteA+TQcKg=
-----END CERTIFICATE-----
quit
Certificate has the following attributes:
Fingerprint:AC6FC55E CC29E891 0DC3FAAA B4747C10
% Do you accept this certificate? [yes/no]:yes
Trustpoint CA certificate accepted.
% Certificate successfully imported
```

The following example shows how to configure a certificate authority pool. See the "Client Certificate Authentication" section on page 3-51 for information on configuring certificate authority pools.

```
ssl-module1(config)# ssl-proxy context Default
ssl-module1(config-context)# pool ca root-ca
ssl-module1(config-ctx-ca-pool)# ca trustpoint root
ssl-module1(config-ctx-ca-pool)# exit
```

The following example shows how to disable revocation checking by entering the **revocation-check none** command for the trustpoint. See the "Checking the Certificate Status" section on page 3-58 for information on configuring certificate revocation list options.

```
ssl-module1(config)# crypto pki trustpoint cert1024
ssl-module1(ca-trustpoint)# revocation-check none
ssl-module1(ca-trustpoint)# exit
ssl-module1(config)# exit
ssl-module1#
```

Site 2 Configuration

Site 2 in Figure A-6 shows the SSL Services Module (SSL module 2) installed in slot 3 in Catalyst 6500 switch 2.

The following example shows how to add VLAN 190 between SSL Services Module and the supervisor engine:

```
cat6k-router-2# show mod 3
Mod Ports Card Type                              Model              Serial No.
--- ----- -------------------------------------- ------------------ -----------
  3     1  SSL Module                             WS-SVC-SSL-1       SAD0722010N

Mod MAC addresses                       Hw    Fw           Sw           Status
--- ---------------------------------- ------ ------------ ------------ -------
  3  0002.fcbe.91f0 to 0002.fcbe.91f7   2.0   7.2(1)       2.1(1)       Ok

Mod Online Diag Status
--- -------------------
  3 Bypass

cat6k-router-2# config t
cat6k-router-2(config)# ssl-proxy module 3 allowed-vlan 190
cat6k-router-2(config)# exit
```

The following example shows how to configure the VLAN, configure the server port as a switchport, and assign the switchport to the access VLAN:

```
cat6k-router-2# config t
cat6k-router-2(config-vlan)# vlan 190
cat6k-router-2(config)# exit
cat6k-router-2# config t
cat6k-router-2(config)# interface GigabitEthernet10/2
cat6k-router-2(config-if)# switchport
cat6k-router-2(config-if)# switchport access vlan 190
cat6k-router-2(config-if)# switchport mode access
cat6k-router-2(config-if)# spanning-tree portfast
cat6k-router-2(config-if)# exit
cat6k-router-2(config)#
```

The following examples show how to configure the access lists:

- Access list “client” is used to match traffic going to host IP address 191.162.2.8 with destination TCP port 443 (the standard SSL port number).

  ```
  cat6k-router-2(config)# ip access-list extended client
  cat6k-router(config-ext-nacl)# permit tcp any host 191.162.2.8 eq 443
  cat6k-router(config-ext-nacl)# exit
  cat6k-router-2(config)#
  ```

- Access list “server” is used to match traffic from server IP address 191.162.2.8 with source port 80 (HTTP traffic).

  ```
  cat6k-router-2(config)# ip access-list extended server
  cat6k-router(config-ext-nacl)# permit tcp host 191.162.2.8 eq 80 any
  cat6k-router(config-ext-nacl)# exit
  cat6k-router-2(config)#
  ```

The following examples show how to configure route maps to redirect traffic to the SSL Services Module for encryption and decryption:

- Route map "client" redirects the traffic that matches access-list "client" to the next hop IP address 191.162.2.11 (the IP address of SSL proxy 0.190 subinterface on SSL-module-2). This configuration redirects encrypted HTTP traffic to the SSL Services Module for decryption.

```
cat6k-router-2(config)# route-map client permit 10
cat6k-route(config-route-map)# match ip address client
cat6k-route(config-route-map)# set ip next-hop 191.162.2.11
cat6k-route(config-route-map)# exit
cat6k-router-2(config)#
```

- Route map "server" redirects the traffic that matches access-list "server" to the next hop IP address 191.162.2.11 (the IP address of SSL proxy 0.190 subinterface on SSL-module-2). This configuration redirects clear text HTTP traffic to the SSL Services Module for encryption.

```
cat6k-router-2(config)# route-map server permit 10
cat6k-route(config-route-map)# match ip address server
cat6k-route(config-route-map)# set ip next-hop 191.162.2.11
cat6k-route(config-route-map)# exit
cat6k-router-2(config)#
```

The following example shows how to configure the routed-interface and assign the IP policy route maps:

```
cat6k-router-2(config)# interface GigabitEthernet1/1
cat6k-router-2(config-if)# ip address 217.162.1.1 255.255.255.0
cat6k-router-2(config-if)# ip policy route-map client
cat6k-router-2(config-if)# exit
cat6k-router-2(config)#
cat6k-router-2(config-if)# interface Vlan190
cat6k-router-2(config-if)# ip address 191.162.2.10 255.0.0.0
cat6k-router-2(config-if)# ip policy route-map server
cat6k-router-2(config-if)# exit
cat6k-router-2(config)# exit
```

SSL Module 2 Configuration

The following example shows how to configure the SSL server proxy to decrypt the encrypted HTTP traffic into clear text HTTP traffic:

```
ssl-module2# configure terminal
ssl-module2(config)# ssl-proxy context Default
ssl-module2(config-context)# service decrypt-ssl-traffic
ssl-module2(config-ctx-ssl-proxy)# virtual ipaddr 191.162.2.8 protocol tcp port 443
secondary
ssl-module2(config-ctx-ssl-proxy)# server ipaddr 191.162.2.10 protocol tcp port 80
ssl-module2(config-ctx-ssl-proxy)# certificate rsa general-purpose trustpoint cert1024
ssl-module2(config-ctx-ssl-proxy)# no nat server
ssl-module2(config-ctx-ssl-proxy)# trusted-ca root-ca
ssl-module2(config-ctx-ssl-proxy)# authenticate verify all
ssl-module2(config-ctx-ssl-proxy)# inservice
ssl-module2(config-ctx-ssl-proxy)# exit
ssl-module2(config)#
```

This example shows how to configure SSL proxy subinterface:

```
ssl-module2(config)# interface SSL-Proxy 0.190
ssl-module2(config-subif)# encapsulation dot1q 190
ssl-module2(config-subif)# ip address 191.162.2.11 255.255.0.0
ssl-module2(config-subif)# no shutdown
ssl-module2(config-subif)# exit
ssl-module2(config)#
```

The following example shows how to import the root-ca certificate to the SSL Services Module:

```
ssl-module2(config)# crypto pki trustpoint root-ca
ssl-module2(ca-trustpoint)# revocation-check none
ssl-module2(ca-trustpoint)# enrollment terminal
ssl-module2(ca-trustpoint)# exit
ssl-module2(config)# crypto pki authenticate root-ca

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

-----BEGIN CERTIFICATE-----
MIIC1zCCAoGgAwIBAgIQadUxzU/i97hDmZRYJ1bBcDANBgkqhkiG9w0BAQUFADB1
MQswCQYDVQQGEwJVUzETMBEGA1UECBMKY2FsaWZvcm5pYTERMA8GA1UEBxMIc2Fu
IGpvc2UxDjAMBgNVBAoTBWNpc2NvMQwwCgYDVQQLEwNoc3MxIDAeBgNVBAMTF3Np
bXBzb24tZGV2dGVzdC1yb290LUNBMB4XDTAzMTExMTIxNDgwMloXDTEzMTExMTIx
NTczOVowdTELMAkGA1UEBhMCVVMxEzARBgNVBAgTCmNhbGlmb3JuaWExETAPBgNV
BAcTCHNhbiBqb3NlMQ4wDAYDVQQKEwVjaXNjbzEMMAoGA1UECxMDaHNzMSAwHgYD
VQQDExdzaW1wc29uLWRldnRlc3Qtcm9vdC1DQTBcMA0GCSqGSIb3DQEBAQUAA0sA
MEgCQQCWEibAnUlVqQNUn0Wb94qnHi8FKjmVhibLHGRl6J+V7gHgzmF2MTz5WP5l
VQ2/1NVu0HjUORRdeCm1/raKJ/7ZAgMBAAGjgewwgekwCwYDVR0PBAQDAgHGMA8G
A1UdEwEB/wQFMAMBAf8wHQYDVR0OBBYEFCYGLUBTKNd9EgUonHnoSvbHg0axMIGX
BgNVHR8EgY8wgYwwQ6BBoD+GPWh0dHA6Ly9jaXNjby1sOGo2b2hwbnIvQ2VydEVu
cm9sbC9zaW1wc29uLWRldnRlc3Qtcm9vdC1DQS5jcmwwRaBDoEGGP2ZpbGU6Ly9c
XGNpc2NvLWw4ajZvaHBuclxDZXJ0RW5yb2xsXHNpbXBzb24tZGV2dGVzdC1yb290
LUNBLmNybDAQBgkrBgEEAYI3FQEEAwIBADANBgkqhkiG9w0BAQUFAANBACBqe1wy
YjalelGZqLVu4bDVMFo6ELCV2AMBgi41K3ix+Z/03PJd7ct2BIAF4lktv9pCe6IO
EoBcmZteA+TQcKg=
-----END CERTIFICATE-----
quit
Certificate has the following attributes:
Fingerprint:AC6FC55E CC29E891 0DC3FAAA B4747C10
% Do you accept this certificate? [yes/no]:yes
Trustpoint CA certificate accepted.
% Certificate successfully imported

ssl-module2(config)#
```

The following example shows how to configure a certificate authority pool. See the "Client Certificate Authentication" section on page 3-51 for information on configuring certificate authority pools. The example also show to disable certificate revocation checking by entering the **revocation-check none** command for the trustpoint. See the "Checking the Certificate Status" section on page 3-58 for information on configuring certificate revocation list options.

```
ssl-module2(config)# ssl-proxy context Default
ssl-module2(config-context)# pool ca root-ca
ssl-module2(config-ctx-ca-pool)# ca trustpoint root-ca
ssl-module2(config-ctx-ca-pool)# exit
ssl-module2(config-context)# exit
ssl-module2(config)# crypto pki trustpoint cert1024
ssl-module2(ca-trustpoint)# revocation-check none
ssl-module2(ca-trustpoint)# exit
ssl-module2(config)#
```

The following example show how to display statistics when connections are active:

- SSL module 1

```
ssl-module1# show ssl-proxy con
Connections for TCP module 1
Local Address         Remote Address        VLAN Conid  Send-Q Recv-Q State
--------------------- --------------------- ---- ------ ------ ------ ------
191.162.2.8:80        7.100.100.1:34472     7    9      0      0      ESTAB
7.100.100.1:34472     191.162.2.8:443       7    196617 0      0      ESTAB
```

- SSL module 12

```
ssl-module2# show ssl-proxy con
Connections for TCP module 1
Local Address         Remote Address        VLAN Conid  Send-Q Recv-Q State
--------------------- --------------------- ---- ------ ------ ------ ------
191.162.2.8:443       7.100.100.1:34472     190  9      0      0      ESTAB
7.100.100.1:34472     191.162.2.8:80        190  196617 0      0      ESTAB
```

Certificate Security Attribute-Based Access Control Examples

The Certificate Security Attribute-Based Access Control feature adds fields to the certificate that allow specifying an access control list (ACL) to create a certificate-based ACL.

For information on configuring certificate security attribute-based access control, refer to *Certificate Security Attribute-Based Access Control* at this URL:

http://www.cisco.com/univercd/cc/td/doc/product/software/ios122/122newft/122t/122t15/ftcrtacl.htm

In the following example, SSL connections for the SSL proxy service “ssl-offload” are successful only if the subject-name of the client certificate contains the domain name **.cisco.com**:

```
ssl-proxy(config)# ssl-proxy context Default
ssl-proxy(config-context)# service ssl-offload
ssl-proxy(config-ctx-ssl-proxy)# virtual ipaddr 8.100.100.126 protocol tcp port 443
ssl-proxy(config-ctx-ssl-proxy)# server ipaddr 191.162.2.8 protocol tcp port 80
ssl-proxy(config-ctx-ssl-proxy)# certificate rsa general-purpose trustpoint cert
ssl-proxy(config-ctx-ssl-proxy)# nat client client-nat
ssl-proxy(config-ctx-ssl-proxy)# trusted-ca root-ca
ssl-proxy(config-ctx-ssl-proxy)# authenticate verify all
ssl-proxy(config-ctx-ssl-proxy)# inservice
ssl-proxy(config-ctx-ssl-proxy)# exit
ssl-proxy(config-context)# pool ca root-ca
ssl-proxy(config-ctx-ca-pool)# ca trustpoint root
ssl-proxy(config-ctx-ca-pool)# end
ssl-proxy(config)#
ssl-proxy(config)# crypto pki trustpoint root
ssl-proxy(ca-trustpoint)# enrollment terminal
ssl-proxy(ca-trustpoint)# revocation-check none
ssl-proxy(ca-trustpoint)# match certificate acl
ssl-proxy(ca-trustpoint)# exit
ssl-proxy(config)#
ssl-proxy(config)# crypto pki certificate map acl 10
ssl-proxy(ca-certificate-map)# subject-name co .cisco.com
ssl-proxy(ca-certificate-map)# exit
```

In the following example, certificate ACLs are configured so that SSL connections for the proxy service "ssl-offload" are successful for the following conditions:

- the subject-name of the client certificate contains **ste3-server.cisco.com** or **ste2-server.cisco.com**.
- the valid-start of the client certificate is greater than or equal to 30th Jul 2003.
- the expiration date of the client certificate is less than 1st Jan 2007.
- the issuer-name of the client certificate contains "certificate manager" in the string.

```
ssl-proxy(config)# ssl-proxy context Default
ssl-proxy(config-context)# service ssl-offload
ssl-proxy(config-ctx-ssl-proxy)# virtual ipaddr 8.100.100.126 protocol tcp port 443
ssl-proxy(config-ctx-ssl-proxy)# server ipaddr 191.162.2.8 protocol tcp port 80
ssl-proxy(config-ctx-ssl-proxy)# certificate rsa general-purpose trustpoint cert
ssl-proxy(config-ctx-ssl-proxy)# nat client client-nat
ssl-proxy(config-ctx-ssl-proxy)# trusted-ca root-ca
ssl-proxy(config-ctx-ssl-proxy)# authenticate verify all
ssl-proxy(config-ctx-ssl-proxy)# inservice
ssl-proxy(config-ctx-ssl-proxy)# exit
ssl-proxy(config-context)# pool ca root-ca
ssl-proxy(config-ctx-ca-pool)# ca trustpoint root
ssl-proxy(config-ctx-ca-pool)# exit
ssl-proxy(config)#
ssl-proxy(config)# crypto pki trustpoint root
ssl-proxy(ca-trustpoint)# enrollment terminal
ssl-proxy(ca-trustpoint)# revocation-check none
ssl-proxy(ca-trustpoint)# match certificate acl
ssl-proxy(ca-trustpoint)# exit
ssl-proxy(config)#
ssl-proxy(config)# crypto pki certificate map acl 10
ssl-proxy(ca-certificate-map)# subject-name co ste3-server.cisco.com
ssl-proxy(ca-certificate-map)# valid-start ge Jul 30 2003 00:00:00 UTC
ssl-proxy(ca-certificate-map)# expires-on lt Jan 01 2007 00:00:00 UTC
ssl-proxy(ca-certificate-map)# issuer-name co certificate manager
ssl-proxy(ca-certificate-map)# exit
ssl-proxy(config)#
ssl-proxy(config)# crypto pki certificate map acl 20
ssl-proxy(ca-certificate-map)# subject-name co ste2-server.cisco.com
ssl-proxy(ca-certificate-map)# expires-on lt Jan 01 2007 00:00:00 UTC
ssl-proxy(ca-certificate-map)# issuer-name co certificate manager
ssl-proxy(ca-certificate-map)# valid-start ge Jul 30 2003 00:00:00 UTC
ssl-proxy(ca-certificate-map)# exit
```

In the following SSL initiation example, the server certificate is checked for the domain name in the certificate field. SSL initiation is successful only if the subject-name of the server certificate contains the domain name **.cisco.com**:

```
ssl-proxy# configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
ssl-proxy(config)# ssl-proxy context Default
ssl-proxy(config-context)# service ssl-initiation client
ssl-proxy(config-ctx-ssl-proxy)# virtual ipaddr 8.100.100.126 protocol tcp port 81
ssl-proxy(config-ctx-ssl-proxy)# server ipaddr 191.162.2.8 protocol tcp port 443
ssl-proxy(config-ctx-ssl-proxy)# nat client client-nat
ssl-proxy(config-ctx-ssl-proxy)# trusted-ca root
ssl-proxy(config-ctx-ssl-proxy)# authenticate verify all
ssl-proxy(config-ctx-ssl-proxy)# inservice
ssl-proxy(config-ctx-ssl-proxy)# exit
ssl-proxy(config-context)# ssl-proxy pool ca root-ca
ssl-proxy(config-ca-pool)# ca trustpoint root
ssl-proxy(config-ca-pool)# exit
ssl-proxy(config)#
```

```
ssl-proxy(config)# crypto pki trustpoint root
ssl-proxy(ca-trustpoint)# enrollment terminal
ssl-proxy(ca-trustpoint)# revocation-check none
ssl-proxy(ca-trustpoint)# match certificate acl
ssl-proxy(ca-trustpoint)# exit
ssl-proxy(config)#
ssl-proxy(config)# crypto pki certificate map acl 10
ssl-proxy(ca-certificate-map)# subject-name co .cisco.com
ssl-proxy(ca-certificate-map)# exit
ssl-proxy(config)#
```

HTTP Header Insertion Examples

The following examples show how to insert various HTTP headers and how to display header insertion statistics.

Example 1

This example shows how to insert custom headers, client IP address and TCP port number information, and a prefix string in HTTP requests that are sent to the server:

```
ssl-proxy# configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
ssl-proxy(config)# ssl-proxy context Default
ssl-proxy(config-context)# policy http-header ssl-offload
ssl-proxy(config-ctx-http-header-policy)# custom SOFTWARE VERSION :3.1(1)
ssl-proxy(config-ctx-http-header-policy)# custom module :SSL MODULE - CATALYST 6500
ssl-proxy(config-ctx-http-header-policy)# custom
type-of-proxy:server_proxy_with_1024_bit_key_size
ssl-proxy(config-ctx-http-header-policy)# client-ip-port
ssl-proxy(config-ctx-http-header-policy)# prefix SSL-OFFLOAD
ssl-proxy(config-ctx-http-header-policy)# exit
ssl-proxy(config-context)# service ssl-offload
ssl-proxy(config-ctx-ssl-proxy)# virtual ipaddr 8.100.100.126 protocol tcp port 443
ssl-proxy(config-ctx-ssl-proxy)# server ipaddr 191.162.2.8 protocol tcp port 80
ssl-proxy(config-ctx-ssl-proxy)# certificate rsa general-purpose trustpoint cert
ssl-proxy(config-ctx-ssl-proxy)# nat client client-nat
ssl-proxy(config-ctx-ssl-proxy)# policy http-header ssl-offload
ssl-proxy(config-ctx-ssl-proxy)# inservice
ssl-proxy(config-ctx-ssl-proxy)# end
ssl-proxy(config)# exit
```

Custom headers and client IP address and TCP port number information are added to every HTTP request and are prefixed by the prefix string, as shown below:

```
SSL-OFFLOAD-Client-IP:7.100.100.1
SSL-OFFLOAD-Client-Port:59008
SSL-OFFLOAD-SOFTWARE VERSION :3.1(1)
SSL-OFFLOAD-module :SSL MODULE - CATALYST 6500
SSL-OFFLOAD-type-of-proxy:server_proxy_with_1024_bit_key_size
```

This example shows how to display header insertion information:

```
ssl-proxy# show ssl-proxy stats hdr
Header Insert Statistics:
    Session Headers Inserted : 4          Custom Headers Inserted : 8
    Session Id's Inserted    : 4          Client Cert. Inserted   : 2
    Client IP/Port Inserted  : 4          PEM Cert. Inserted      : 0
    Aliased Hdrs Inserted    : 0
    No End of Hdr Detected   : 0          Payload no HTTP header  : 0
    Desc Alloc Failed        : 0          Buffer Alloc Failed     : 0
    Client Cert Errors       : 0          Malloc failed           : 0
    Service Errors           : 0          Conn Entry Invalid      : 0
    Buffers allocated        : 0          Buffers Scanned         : 4
    Insertion Points Found   : 4          Hdrs Spanning Records   : 0
    End of Header Found      : 4          Buffers Accumulated     : 4
    Multi-buffer IP Port     : 0          Multi-buffer Session Id : 0
    Multi-buffer Session Hdr : 0          Multi-buffer Custom Hdr : 0
    Scan Internal Error      : 0          Database Not Initialized: 0
```

This example shows how to display SSL statistics:

```
ssl-proxy# show ssl-proxy stats ssl
SSL Statistics:
    conns attempted     : 15            conns completed     : 15
    conns in handshake  : 0             conns in data       : 0
    renegs attempted    : 0             conns in reneg      : 0
    active sessions     : 0             max handshake conns : 3
    rand bufs allocated : 1             cached rand buf miss: 0
    current device q len: 0             max device q len    : 1
    sslv2 forwards      : 0             cert reqs processed : 6
    fatal alerts rcvd   : 0             fatal alerts sent   : 5
    stale packet drops  : 0             service_id discards : 0
    session reuses      : 0             hs handle in use    : 0
    netscape step-ups   : 0             SGC step-ups        : 0

    SSL3 Statistics:
      full handshakes    : 0             resumed handshakes : 0
      handshake failures : 0             data failures      : 0
      bad macs received  : 0             pad errors         : 0
      conns established with cipher rsa-with-rc4-128-md5          : 0
      conns established with cipher rsa-with-rc4-128-sha          : 0
      conns established with cipher rsa-with-des-cbc-sha          : 0
      conns established with cipher rsa-with-3des-ede-cbc-sha     : 0
      conns established with cipher rsa-with-null-md5             : 0
      conns established with cipher rsa-exp1024-with-des-cbc-sha  : 0
      conns established with cipher rsa-exp1024-with-rc4-56-sha   : 0
      conns established with cipher rsa-exp1024-with-rc4-56-md5   : 0
      conns established with cipher rsa-exp-with-rc4-40-md5       : 0
      conns established with cipher rsa-exp-with-des40-cbc-sha    : 0

    TLS1 Statistics:
      full handshakes    : 6             resumed handshakes : 2
      handshake failures : 0             data failures      : 0
      bad macs received  : 0             pad errors         : 0
      conns established with cipher rsa-with-rc4-128-md5          : 8
      conns established with cipher rsa-with-rc4-128-sha          : 0
      conns established with cipher rsa-with-des-cbc-sha          : 0
      conns established with cipher rsa-with-3des-ede-cbc-sha     : 0
      conns established with cipher rsa-with-null-md5             : 0
      conns established with cipher rsa-exp1024-with-des-cbc-sha  : 0
      conns established with cipher rsa-exp1024-with-rc4-56-sha   : 0
      conns established with cipher rsa-exp1024-with-rc4-56-md5   : 0
      conns established with cipher rsa-exp-with-rc4-40-md5       : 0
      conns established with cipher rsa-exp-with-des40-cbc-sha    : 0
```

Example 2

This example shows how to insert session headers and a prefix string. The full session headers are added to the HTTP request when the full SSL handshake occurs. However, only the session ID is inserted when the session resumes.

```
ssl-proxy# configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
ssl-proxy(config)# ssl-proxy context Default
ssl-proxy(config-context)# policy http-header ssl-offload
ssl-proxy(config-ctx-http-header-policy)# session
ssl-proxy(config-ctx-http-header-policy)# prefix SSL-OFFLOAD
ssl-proxy(config-ctx-http-header-policy)# exit
ssl-proxy(config-context)# service ssl-offload
ssl-proxy(config-ctx-ssl-proxy)# virtual ipaddr 8.100.100.126 protocol tcp port 443
ssl-proxy(config-ctx-ssl-proxy)# server ipaddr 191.162.2.8 protocol tcp port 80
ssl-proxy(config-ctx-ssl-proxy)# certificate rsa general-purpose trustpoint cert
ssl-proxy(config-ctx-ssl-proxy)# nat client client-nat
ssl-proxy(config-ctx-ssl-proxy)# policy http-header ssl-offload
ssl-proxy(config-ctx-ssl-proxy)# inservice
ssl-proxy(config-ctx-ssl-proxy)# end
ssl-proxy(config)# exit
```

For the full SSL handshake, the session headers, prefixed by the prefix string, are added to the HTTP request as shown below:

```
SSL-OFFLOAD-Session-Id:33:FF:2C:2D:25:15:3C:50:56:AB:FA:5A:81:0A:EC:E9:00:00:0A:03:00:60:
2F:30:9C:2F:CD:56:2B:91:F2:FF
SSL-OFFLOAD-Session-Step-Up: FALSE
SSL-OFFLOAD-Session-Initial-Cipher-Name:
SSL-OFFLOAD-Session-Initial-Cipher-Key-Size:
SSL-OFFLOAD-Session-Initial-Cipher-Use-Size:
SSL-OFFLOAD-Session-Cipher-Name:RC4-SHA
SSL-OFFLOAD-Session-Cipher-Key-Size:128
SSL-OFFLOAD-Session-Cipher-Use-Size:128
```

When the session resumes, only the session ID is inserted:

```
SSL-OFFLOAD-Session-Id:33:FF:2C:2D:25:15:3C:50:56:AB:FA:5A:81:0A:EC:E9:00:00:0A:03:00:60:
2F:30:9C:2F:CD:56:2B:91:F2:FF
```

This example shows how to display header insertion information:

```
ssl-proxy# show ssl-proxy stats hdr
Header Insert Statistics:
    Session Headers Inserted :1          Custom Headers Inserted :0
    Session Id's Inserted    :2          Client Cert. Inserted   :0
    Client IP/Port Inserted  :0
    No End of Hdr Detected   :0          Payload no HTTP header  :0
    Desc Alloc Failed        :0          Buffer Alloc Failed     :0
    Client Cert Errors       :0          No Service              :0
```

This example shows how to display SSL statistics:

```
ssl-proxy# show ssl-proxy stats ssl
SSL Statistics:
    conns attempted     :2             conns completed     :2
    conns in handshake  :0             conns in data       :0
    renegs attempted    :0             conns in reneg      :0
    active sessions     :0             max handshake conns :1
```

```
rand bufs allocated :0             cached rand buf miss:0
current device q len:0             max device q len    :2
sslv2 forwards      :0             cert reqs processed :0
fatal alerts rcvd   :0             fatal alerts sent   :0
stale packet drops  :0             service_id discards :0
session reuses      :0

SSL3 Statistics:
  full handshakes    :0             resumed handshakes :0
  handshake failures :0             data failures      :0
  bad macs received  :0             pad errors         :0
  conns established with cipher rsa-with-rc4-128-md5        :0
  conns established with cipher rsa-with-rc4-128-sha        :0
  conns established with cipher rsa-with-des-cbc-sha        :0
  conns established with cipher rsa-with-3des-ede-cbc-sha   :0

TLS1 Statistics:
  full handshakes    :1             resumed handshakes :1
  handshake failures :0             data failures      :0
  bad macs received  :0             pad errors         :0
  conns established with cipher rsa-with-rc4-128-md5        :0
  conns established with cipher rsa-with-rc4-128-sha        :2
  conns established with cipher rsa-with-des-cbc-sha        :0
  conns established with cipher rsa-with-3des-ede-cbc-sha   :0
```

Example 3

This example shows how to insert custom headers, decoded client certificate fields, and the IP address and destination TCP port number of the client-side connection, prefixed by the prefix string. The complete decoded client certificate fields are inserted for the full SSL handshake. However, only session ID is inserted when the SSL session resumes.

```
ssl-proxy# configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
ssl-proxy(config)# ssl-proxy context Default
ssl-proxy(config-context)# policy http-header ssl-offload
ssl-proxy(config-ctx-http-header-policy)# custom SOFTWARE VERSION :3.1(1)
ssl-proxy(config-ctx-http-header-policy)# custom module :SSL MODULE - CATALYST 6500
ssl-proxy(config-ctx-http-header-policy)# custom
type-of-proxy:server_proxy_with_1024_bit_key_size
ssl-proxy(config-ctx-http-header-policy)# client-cert
ssl-proxy(config-ctx-http-header-policy)# client-ip-port
ssl-proxy(config-ctx-http-header-policy)# prefix SSL-OFFLOAD
ssl-proxy(config-ctx-http-header-policy)# exit
ssl-proxy(config-context)# service ssl-offload
ssl-proxy(config-ctx-ssl-proxy)# virtual ipaddr 8.100.100.126 protocol tcp port 443
ssl-proxy(config-ctx-ssl-proxy)# server ipaddr 191.162.2.8 protocol tcp port 80
ssl-proxy(config-ctx-ssl-proxy)# certificate rsa general-purpose trustpoint cert
ssl-proxy(config-ctx-ssl-proxy)# nat client client-nat
ssl-proxy(config-ctx-ssl-proxy)# policy http-header ssl-offload
ssl-proxy(config-ctx-ssl-proxy)# trusted-ca root-ca
ssl-proxy(config-ctx-ssl-proxy)# authenticate verify all
ssl-proxy(config-ctx-ssl-proxy)# inservice
ssl-proxy(config-ctx-ssl-proxy)# end
ssl-proxy(config)# exit
```

For the full SSL handshake, the custom headers, decoded client certificate fields, the IP address and destination TCP port number of the client-side connection, prefixed by the prefix string, are added to the HTTP request, as shown below:

```
SSL-OFFLOAD-Client-IP:7.100.100.1
SSL-OFFLOAD-Client-Port:59011
SSL-OFFLOAD-Session-Id:0F:61:9C:F2:E5:98:70:9D:1B:C1:EA:1D:38:F5:A1:2B:00:00:0E:03:00:60:
2F:30:9C:2F:1D:7D:5A:82:30:F6
SSL-OFFLOAD-SOFTWARE VERSION :3.1(1)
SSL-OFFLOAD-module :SSL MODULE - CATALYST 6500
SSL-OFFLOAD-type-of-proxy:server_proxy_with_1024_bit_key_size
SSL-OFFLOAD-ClientCert-Valid:1
SSL-OFFLOAD-ClientCert-Error:none
SSL-OFFLOAD-ClientCert-Fingerprint:1B:11:0F:E8:20:3F:6C:23:12:9C:76:C0:C1:C2:CC:85
SSL-OFFLOAD-ClientCert-Subject-CN:a
SSL-OFFLOAD-ClientCert-Issuer-CN:Certificate Manager
SSL-OFFLOAD-ClientCert-Certificate-Version:3
SSL-OFFLOAD-ClientCert-Serial-Number:0F:E5
SSL-OFFLOAD-ClientCert-Data-Signature-Algorithm:sha1WithRSAEncryption
SSL-OFFLOAD-ClientCert-Subject:OID.1.2.840.113549.1.9.2 = ste2-server.cisco.com +
OID.2.5.4.5 = B0FFF22E, CN = a, O = Cisco
SSL-OFFLOAD-ClientCert-Issuer:CN = Certificate Manager, OU = HSS, O = Cisco, L = San Jose,
ST = California, C = US
SSL-OFFLOAD-ClientCert-Not-Before:22:29:26 UTC Jul 30 2003
SSL-OFFLOAD-ClientCert-Not-After:07:00:00 UTC Apr 27 2006
SSL-OFFLOAD-ClientCert-Public-Key-Algorithm:rsaEncryption
SSL-OFFLOAD-ClientCert-RSA-Public-Key-Size:1024 bit
SSL-OFFLOAD-ClientCert-RSA-Modulus-Size:1024 bit
SSL-OFFLOAD-ClientCert-RSA-Modulus:B3:32:3C:5E:C9:D1:CC:76:FF:81:F6:F7:97:58:91:4D:B2:0E:
C1:3A:7B:62:63:BD:5D:F6:5F:68:F0:7D:AC:C6:72:F5:72:46:7E:FD:38:D3:A2:E1:03:8B:EC:F7:C9:9A:
80:C7:37:DA:F3:BE:1F:F4:5B:59:BD:52:72:94:EE:46:F5:29:A4:B3:9B:2E:4C:69:D0:11:59:F7:68:3A:
D9:6E:ED:6D:54:4E:B5:A7:89:B9:45:9E:66:0B:90:0B:B1:BD:F4:C8:15:12:CD:85:13:B2:0B:FE:7E:8D:
F0:D7:4A:98:BB:08:88:6E:CC:49:60:37:22:74:4D:73:1E:96:58:91
SSL-OFFLOAD-ClientCert-RSA-Exponent:00:01:00:01
SSL-OFFLOAD-ClientCert-X509v3-Authority-Key-Identifier:keyid=EE:EF:5B:BD:4D:CD:F5:6B:60:
9D:CF:46:C2:EA:25:7B:22:A5:08:00
SSL-OFFLOAD-ClientCert-X509v3-Basic-Constraints:
SSL-OFFLOAD-ClientCert-Signature-Algorithm:sha1WithRSAEncryption
SSL-OFFLOAD-ClientCert-Signature:87:09:C1:F8:86:C1:15:C5:57:18:8E:B3:0D:62:E1:0F:6F:D4:9D:
75:DA:5D:53:E2:C6:0B:73:99:61:BE:B0:F6:19:83:F2:E5:48:1B:D2:6C:92:83:66:B3:63:A6:58:B4:5C:
0E:5D:1B:60:F9:86:AF:B3:93:07:77:16:74:4B:C5
SSL-OFFLOAD-ClientCert-X509v3-Subject-Alternative-Name:
ipAddress=192.168.1.100,rfc822Name=my@other.com
SSL-OFFLOAD-ClientCert-X509v3-Key-Usage: Digital Signature,Non-Repudiation,Key
Encipherment,
Data Encipherment,Key Agreement,Key Cert Sign,CRL Signature,Encipher Only,Decipher Only
SSL-OFFLOAD-ClientCert-X509v3-Authority-Information-Access: Access Method=OCSP,Access
Location=http://ocsp.my.host/"
SSL-OFFLOAD-ClientCert-X509v3-CRL-Distribution-Points: http://myhost.com/myca.crl
```

This example shows how to display header insertion information:

```
ssl-proxy# show ssl-proxy stats hdr
Header Insert Statistics:
    Session Headers Inserted :0          Custom Headers Inserted :1
    Session Id's Inserted    :1          Client Cert. Inserted   :1
    Client IP/Port Inserted  :1
    No End of Hdr Detected   :0          Payload no HTTP header  :0
    Desc Alloc Failed        :0          Buffer Alloc Failed     :0
    Client Cert Errors       :0          No Service              :0
```

This example shows how to display SSL statistics:

```
ssl-proxy# show ssl-proxy stats ssl
SSL Statistics:
    conns attempted     :1              conns completed     :1
    conns in handshake  :0              conns in data       :0
    renegs attempted    :0              conns in reneg      :0
    active sessions     :0              max handshake conns :1
    rand bufs allocated :0              cached rand buf miss:0
    current device q len:0              max device q len    :2
    sslv2 forwards      :0              cert reqs processed :1
    fatal alerts rcvd   :0              fatal alerts sent   :0
    stale packet drops  :0              service_id discards :0
    session reuses      :0

    SSL3 Statistics:
      full handshakes    :0              resumed handshakes :0
      handshake failures :0              data failures      :0
      bad macs received  :0              pad errors         :0
      conns established with cipher rsa-with-rc4-128-md5        :0
      conns established with cipher rsa-with-rc4-128-sha        :0
      conns established with cipher rsa-with-des-cbc-sha        :0
      conns established with cipher rsa-with-3des-ede-cbc-sha   :0

    TLS1 Statistics:
      full handshakes    :1              resumed handshakes :0
      handshake failures :0              data failures      :0
      bad macs received  :0              pad errors         :0
      conns established with cipher rsa-with-rc4-128-md5        :0
      conns established with cipher rsa-with-rc4-128-sha        :0
      conns established with cipher rsa-with-des-cbc-sha        :0
      conns established with cipher rsa-with-3des-ede-cbc-sha   :1
```

Example 4

This example shows how to configure the SSL Services Module to insert session headers, a header alias, and a prefix:

```
ssl-proxy# configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
ssl-proxy(config)# ssl-proxy context s1
ssl-proxy(config-context)# policy http-header ssl-offload
ssl-proxy(config-ctx-http-header-policy)# prefix SSL-OFFLOAD
ssl-proxy(config-ctx-http-header-policy)# session
ssl-proxy(config-ctx-http-header-policy)# alias My-Session-Cipher session-cipher-name
ssl-proxy(config-ctx-http-header-policy)# exit
ssl-proxy(config-context)# service ssl-offload
ssl-proxy(config-ctx-ssl-proxy)# policy http-header ssl-offload
```

In addition to the standard HTTP headers, the following header information is inserted:

Note The alias name (My-Session-Cipher) is used instead of the standard name (session-cipher-name).

```
SSL-OFFLOAD-Session-Id:33:FF:2C:2D:25:15:3C:50:56:AB:FA:5A:81:0A:EC:E9:00:00:0A:03:00:60:
2F:30:9C:2F:CD:56:2B:91:F2:FF
SSL-OFFLOAD-My-Session-Cipher:RC4-SHA
SSL-OFFLOAD-Session-Cipher-Key-Size:128
SSL-OFFLOAD-Session-Cipher-Use-Size:128
SSL-OFFLOAD-Session-Step-Up:FALSE
SSL-OFFLOAD-Session-Initial-Cipher-Key-Size:
SSL-OFFLOAD-Session-Initial-Cipher-Name:
SSL-OFFLOAD-Session-Initial-Cipher-Use-Size:
```

Example 5

This example shows how to insert the client certificate in PEM format as a header, and it adds a prefix string to the header. The PEM format client certificate is inserted for the full SSL handshake. However, only the session ID is inserted when the SSL session resumes.

```
ssl-proxy# configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
ssl-proxy(config)# ssl-proxy context Default
ssl-proxy(config-context)# policy http-header ssl-offload
ssl-proxy(config-ctx-http-header-policy)# client-cert pem
ssl-proxy(config-ctx-http-header-policy)# prefix SSL-OFFLOAD
ssl-proxy(config-ctx-http-header-policy)# exit
ssl-proxy(config-context)# service ssl-offload
ssl-proxy(config-ctx-ssl-proxy)# virtual ipaddr 8.100.100.126 protocol tcp port 443
ssl-proxy(config-ctx-ssl-proxy)# server ipaddr 191.162.2.8 protocol tcp port 80
ssl-proxy(config-ctx-ssl-proxy)# certificate rsa general-purpose trustpoint cert
ssl-proxy(config-ctx-ssl-proxy)# nat client client-nat
ssl-proxy(config-ctx-ssl-proxy)# policy http-header ssl-offload
ssl-proxy(config-ctx-ssl-proxy)# trusted-ca root-ca
ssl-proxy(config-ctx-ssl-proxy)# authenticate verify all
ssl-proxy(config-ctx-ssl-proxy)# inservice
ssl-proxy(config-ctx-ssl-proxy)# exit
ssl-proxy(config)# exit
```

For the full SSL handshake, the PEM format client certificate, prefixed by the prefix string, is added to the HTTP request, as shown below:

```
SSL-OFFLOAD-Session-Id: 3A:D9:E3:B7:6E:95:EA:B4:A8:9E:E0:BD:76:8D:43:3A:00:00:12:03:00:E0:
B0:FF:F1:48:1C:5B:B2:62:D9:40
SSL-OFFLOAD-ClientCert-PEM: -----BEGIN CERTIFICATE-----
MIIDcDCCAtmgAwIBAgIBDDANBgkqhkiG9w0BAQQFADCBhjEeMBwGA1UEAxMVTXkg
Q2VydGlmaWNhdGlvbiBBdXRoMQ8wDQYDVQQIEwZBbGFza2ExCzAJBgNVBAYTAlVT
MRwwGgYJKoZIhvcNAQkBFg1teWNhQG15Y2Eub3JnMSgwJgYDVQQKEx9NeSBSb290
IENlcnRpZmljYXRpb24gQXV0aG9yaXR5MB4XDTA1MDcyMDE5MTY1N1oXDTEwMDcx
OTE5MTY1N1owgYUxFDASBgNVBAMTC21vcmVleHQuY29tMQswCQYDVQQIEwJTVzEL
MAkGA1UEBhMCRVUxHzAdBgkqhkiG9w0BCQEWEG1vcmVAbW9yZWV4dC5jb20xHDAa
BgNVBAoTE01vcmUgRXh0ZW5zaW9ucyBMdGQxFDASBgNVBAsTC21vcmUgZXh0IG91
MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQDNHLRU6I2t7b7DFLFJbYLpFplj
D/l+9m1ZGtVp7Bf1K7YuXK8XztVs/XGk8ibmdBGyVTDyu/3gDCS6jKQUtAqwijGn
LitqEEtjl6e+7og6iB58NgEdO8UyT5wBwCX6Q5bdJuB460N3EhlEYlGs4lBib61A
HWPsiugF817nm/Ee2wIDAQABo4HsMIHpMAwGA1UdEwQFMAMBAf8wDwYDVR0PAQH/
BAUDAwf/gDBMBgNVHREERTBDgRBtb3JlQG1vcmVleHQuY29tgQxteUBvdGhlci5j
b22GFmh0dHA6Ly9teW90aGVydXJsLmhlcmWHBMCoAWSIAyoDBDAwBggrBgEFBQcB
AQQkMCIwIAYIKwYBBQUHMAKGFGh0dHA6Ly9vY3NwLm15Lmhvc3QvMEgGA1UdHwRB
```

```
MD8wIKAeoByGGmh0dHA6Ly9teWhvc3QuY29tL215Y2EuY3JsMBugGaAXhhVodHRw
Oi8vb3RoLmNvbS9teS5jcmwwDQYJKoZIhvcNAQEEBQADgYEAIABQBXZ5Ky5uZHuv
hxN8pXdr+vm0FRBZmFflEsd46HV4yug4W6pmDzBhW56913HzKYQOQvc0WoRTlZ0f
53rvQZq7UOSGZLtXvn/I9/epdEWCx/YYcWnCNGhTZycf6VRcmainW5prZVBfTUAq
Nlnar2TvW0bLOG/qtkYQZooZzR8=
-----END CERTIFICATE-----
```

URL Rewrite Examples

The following examples show how to configure URL rewrite depending on the desired outcome and assume the following proxy configuration:

```
service frontend
 virtual ipaddr 35.200.200.101 protocol tcp port 443
 server ipaddr 35.200.200.14 protocol tcp port 80
 certificate rsa general-purpose trustpoint TP-1024-pkcs12
 policy url-rewrite test-url-rewrite
 inservice
!
```

Example 1

The following example shows how to configure a protocol rewrite (for example, HTTP to HTTPS) when the clear text port is the standard HTTP port 80. In this example, when the server sends the relocation string as http://ssl-136.cisco.com/index2.html, the SSL Services Module rewrites the string as https://ssl-136.cisco.com/index2.html.

To configure a protocol rewrite (HTTP to HTTPS), specify any of the following URL rewrite rules:

-
```
policy url-rewrite test-url-rewrite
 url ssl-136.cisco.com
!
```
-
```
policy url-rewrite test-url-rewrite
 url ssl*
!
```
-
```
policy url-rewrite test-url-rewrite
 url *com
!
```

Example 2

The following example shows how to configure a protocol rewrite (for example, HTTP to HTTPS) when the clearport is a non-standard HTTP port. In this example, when the server sends the relocation string as http://ssl-136.cisco.com:100/index2.html, the SSL Services Module rewrites the string as https://ssl-136.cisco.com/index2.html.

To configure a protocol rewrite (HTTP to HTTPS) with a non-standard clear text port, specify any of the following URL rewrite rules:

- ```
  policy url-rewrite test-url-rewrite
   url ssl-136.cisco.com clearport 100
  !
  ```
- ```
  policy url-rewrite test-url-rewrite
   url ssl* clearport 100
  !
  ```
- ```
  policy url-rewrite test-url-rewrite
   url *com clearport 100
  !
  ```

Example 3

The following example shows how to configure a protocol rewrite and SSL port rewrite when the clear text port is the standard HTTP port 80. In this example, when the server sends the relocation string as http://ssl-136.cisco.com/index2.html, the SSL Services Module rewrites the string as https://ssl-136.cisco.com:445/index2.html.

To configure a protocol rewrite (HTTP to HTTPS) with a non-standard SSL text port, specify any of the following URL rewrite rules:

- ```
  policy url-rewrite test-url-rewrite
   url ssl-136.cisco.com sslport 445
  !
  ```
- ```
  policy url-rewrite test-url-rewrite
   url ssl* sslport 445
  !
  ```
- ```
  policy url-rewrite test-url-rewrite
   url *com sslport 445
  !
  ```

Example 4

The following example shows how to configure a protocol rewrite and SSL port rewrite when the clear text port is non-standard. In this example, when the server sends the relocation string as http://ssl-136.cisco.com:100/index2.html, the SSL Services Module rewrites the string as https://ssl-136.cisco.com:445/index2.html.

To configure a protocol rewrite and SSL port rewrite with a non-standard clear text port, specify any of the following URL rewrite rules:

- ```
  policy url-rewrite test-url-rewrite
   url ssl-136.cisco.com clearport 100 sslport 445
  !
  ```
- ```
  policy url-rewrite test-url-rewrite
   url ssl* clearport 100 sslport 445
  !
  ```
- ```
  policy url-rewrite test-url-rewrite
   url *com clearport 100 sslport 445
  !
  ```
  ```

The following example displays the above URL rewrite policy:

```
ssl-proxy# show ssl-proxy policy url-rewrite test-url-rewrite
No context name provided, assuming context 'Default'...
Rule URL                                      Clearport SSLport
  1 *com                                        100        445
SSL proxy services using this policy:
          frontend
Usage count of this policy:1

ssl-proxy#
```

HSRP Examples

In systems with an SSL Services Module and a Content Switching Module (CSM), the failover functionality on the CSM provides stateless redundancy on the SSL module. When the SSL module is used in a standalone configuration (using policy-based routing), you can configure HSRP to provide redundancy.

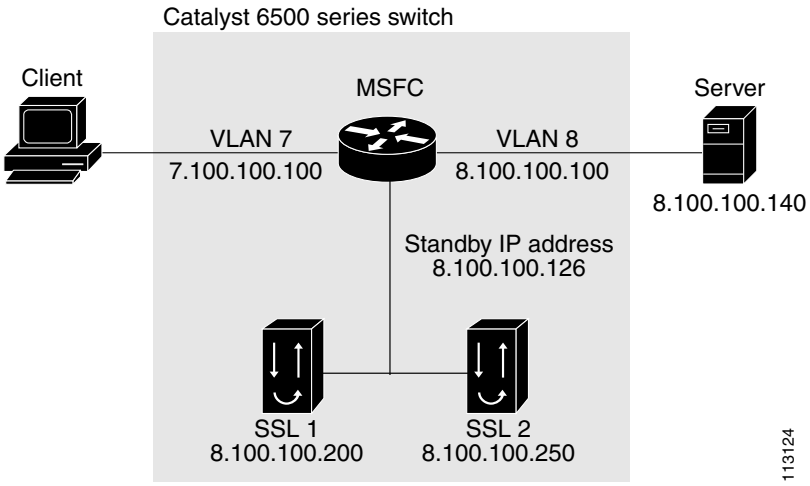
See the "Configuring Redundancy" section on page 4-16 for more information on configuring redundancy using HSRP.

Standalone Redundancy Example

In Figure A-7, both SSL Services Modules have the same proxy service configured and specify the **secondary** keyword for the virtual IP address and the same HSRP configuration. Both modules are configured with standby IP address 8.100.100.126. SSL 1 is the active module and accepts SSL connections. SSL 2 is the backup module and does not accept SSL connections until SSL 1 goes offline.

Policy-based routing is configured on the MSFC so that any TCP traffic destined for IP address 8.100.100.126:443 is redirected to the next-hop IP address 8.100.100.126.

Figure A-7 Standalone Redundancy



Supervisor Engine Configuration

This example shows how to configure the route maps and access lists:

```
Router# config t
Enter configuration commands, one per line.  End with CNTL/Z.
Router(config)# route-map client permit 10
Router(config-route-map)# set ip next-hop 8.100.100.126
Router(config-route-map)# match ip address client
Router(config-route-map)# exit
Router(config)# route-map server permit 10
Router(config-route-map)# match ip address server
Router(config-route-map)# set ip next-hop 8.100.100.126
Router(config-route-map)# exit
Router(config)#
Router(config)# ip access-list extended client
Router(config-ext-nacl)# permit tcp any host 8.100.100.126 eq 443
Router(config-ext-nacl)# exit
Router(config)#
Router(config)# ip access-list extended server
Router(config-ext-nacl)# permit tcp host 8.100.100.140 eq www any
Router(config-ext-nacl)# exit
Router(config)#
Router(config)# interface Vlan7
Router(config-if)# ip address 7.100.100.100 255.0.0.0
Router(config-if)# ip policy route-map client
Router(config-if)# exit
Router(config)#
Router(config)# interface Vlan8
Router(config-if)# ip address 8.100.100.100 255.0.0.0
Router(config-if)# ip policy route-map server
Router(config-if)# exit
Router(config)# exit
Router#
```

SSL 1 Configuration

This example shows how to configure the proxy service and the VLAN on SSL 1:

```
ssl-mod-1# configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
ssl-mod-1(config)# ssl-proxy context Default
ssl-mod-1(config-context)# service ssl-offload
ssl-mod-1(config-ctx-ssl-proxy)# virtual ipaddr 8.100.100.126 protocol tcp port 443
secondary
ssl-mod-1(config-ctx-ssl-proxy)# server ipaddr 8.100.100.140 protocol tcp port 80
ssl-mod-1(config-ctx-ssl-proxy)# certificate rsa general-purpose trustpoint cert
ssl-mod-1(config-ctx-ssl-proxy)# inservice
ssl-mod-1(config-ctx-ssl-proxy)# exit
ssl-mod-1(config-context)# exit
ssl-mod-1(config)# interface SSL-Proxy 0.8
ssl-mod-1(config-subif)# encapsulation dot1q 8
ssl-mod-1(config-subif)# ip address 8.100.100.200 255.0.0.0
ssl-mod-1(config-subif)# standby ip 8.100.100.126
ssl-mod-1(config-subif)# standby timers 1 3
ssl-mod-1(config-subif)# standby priority 90
ssl-mod-1(config-subif)# exit
ssl-mod-1(config)# ip route 191.0.0.0 255.0.0.0 8.100.100.100
ssl-mod-1(config)# exit
ssl-mod-1#
```

SSL 2 Configuration

This example shows how to configure the proxy service and the VLAN on SSL 2:

```
ssl-mod-2# configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
ssl-mod-2(config)# ssl-proxy context Default
ssl-mod-2(config-context)# service ssl-offload
ssl-mod-2(config-ctx-ssl-proxy)# virtual ipaddr 8.100.100.126 protocol tcp port 443
secondary
ssl-mod-2(config-ctx-ssl-proxy)# server ipaddr 8.100.100.140 protocol tcp port 80
ssl-mod-2(config-ctx-ssl-proxy)# certificate rsa general-purpose trustpoint cert
ssl-mod-2(config-ctx-ssl-proxy)# inservice
ssl-mod-2(config-ctx-ssl-proxy)# exit
ssl-mod-2(config-context)# exit
ssl-mod-2(config)# interface SSL-Proxy 0.8
ssl-mod-2(config-subif)# encapsulation dot1q 8
ssl-mod-2(config-subif)# ip address 8.100.100.250 255.0.0.0
ssl-mod-2(config-subif)# standby ip 8.100.100.126
ssl-mod-2(config-subif)# standby timers 1 3
ssl-mod-2(config-subif)# standby priority 110
ssl-mod-2(config-subif)# exit
ssl-mod-2(config)# exit
ssl-mod-2#
```

Load Balancing Example

In Figure A-8, each SSL Services Module is configured with more than one proxy service. Each SSL Services Module has a different HSRP group configured.

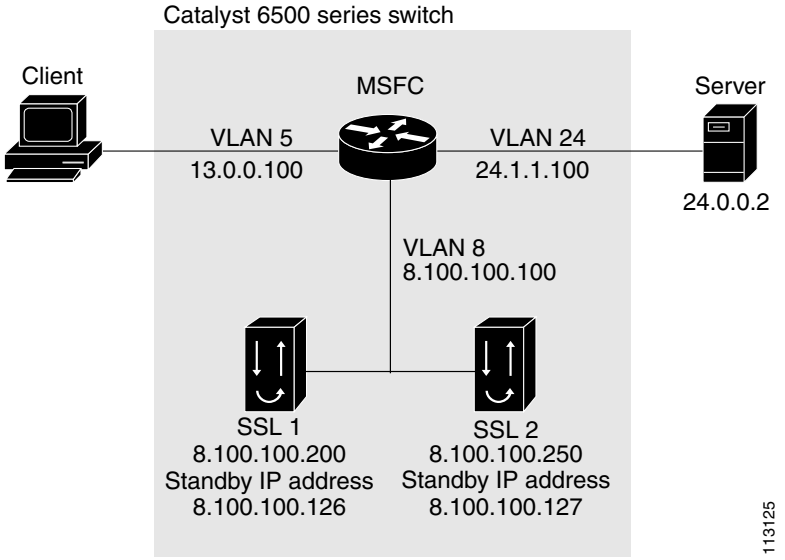
On the MSFC, configure policy-based routing so that traffic to the different proxy services is load balanced between the two SSL Services Modules.

On the SSL Services Modules, configure the **standby** *group_number* **preempt delay** *delay* command for the following reasons:

- When a module goes offline and comes back online, half of the traffic is switched back to the new (online) module for efficient load balancing.
- The new (online) module does not become immediately active, giving sufficient time for the proxy services to come online.

Configure client NAT for each proxy service so that when multiple proxies send traffic to the same server, the traffic from the server is sent back to the module that originated the traffic. See the "Client NAT" section on page 4-15 for information on configuring client NAT.

Figure A-8 Load Balancing



Supervisor Engine Configuration

This example shows how to configure the route maps and access lists:

```
Router# configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
Router(config)# ip access-list extended ssl-offload
Router(config-ext-nacl)# permit tcp any host 8.100.100.110 eq 443
Router(config-ext-nacl)# exit
Router(config)# ip access-list extended ssl-offload-checkout
Router(config-ext-nacl)# permit tcp any host 8.100.100.111 eq 443
Router(config-ext-nacl)#
Router(config-ext-nacl)# exit
Router(config)# route-map client permit 10
Router(config-route-map)# match ip address ssl-offload
Router(config-route-map)# set ip next-hop 8.100.100.126
Router(config-route-map)#
Router(config-route-map)# exit
Router(config)# route-map client permit 20
Router(config-route-map)# match ip address ssl-offload-checkout
Router(config-route-map)# set ip next-hop 8.100.100.127
Router(config-route-map)# exit
Router(config)# interface Vlan5
Router(config-if)# ip address 13.0.0.100 255.0.0.0
Router(config-if)# ip policy route-map client
Router(config-if)# no shutdown
Router(config-if)# exit
Router(config)# interface GigabitEthernet10/7
Router(config-if)# switchport
Router(config-if)# switchport access vlan 5
Router(config-if)# switchport mode access
Router(config-if)# spanning-tree portfast
Router(config-if)# no shutdown
Router(config-if)# exit
```

```
Router(config)# interface GigabitEthernet10/11
Router(config-if)# switchport
Router(config-if)# switchport access vlan 24
Router(config-if)# switchport mode access
Router(config-if)# spanning-tree portfast
Router(config-if)# no shutdown
Router(config-if)# exit
Router(config)# interface Vlan24
Router(config-if)# ip address 24.1.1.100 255.0.0.0
Router(config-if)# no shutdown
Router(config-if)# no ip redirects
Router(config-if)# ^Z
Router#
```

SSL 1 Configuration

This example shows how to configure the proxy services and the VLAN on SSL 1:

```
ssl-mod-1# configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
ssl-mod-1(config)# ssl-proxy context Default
ssl-mod-1(config-context)# natpool client-nat 8.100.1.1 8.100.1.8 netmask 255.0.0.0
ssl-mod-1(config-context)# service ssl-offload
ssl-mod-1(config-ctx-ssl-proxy)# virtual ipaddr 8.100.100.110 protocol tcp port 443
secondary
ssl-mod-1(config-ctx-ssl-proxy)# server ipaddr 24.0.0.2 protocol tcp port 80
ssl-mod-1(config-ctx-ssl-proxy)# certificate rsa general-purpose trustpoint cert
ssl-mod-1(config-ctx-ssl-proxy)# nat client client-nat
ssl-mod-1(config-ctx-ssl-proxy)# inservice
ssl-mod-1(config-ctx-ssl-proxy)# exit
ssl-mod-1(config-context)#
ssl-mod-1(config-context)# service ssl-offload-checkout
ssl-mod-1(config-ctx-ssl-proxy)# virtual ipaddr 8.100.100.111 protocol tcp port 443
secondary
ssl-mod-1(config-ctx-ssl-proxy)# server ipaddr 24.0.0.2 protocol tcp port 80
ssl-mod-1(config-ctx-ssl-proxy)# certificate rsa general-purpose trustpoint cert
ssl-mod-1(config-ctx-ssl-proxy)# nat client client-nat
ssl-mod-1(config-ctx-ssl-proxy)# inservice
ssl-mod-1(config-ctx-ssl-proxy)# exit
ssl-mod-1(config-context)# exit
ssl-mod-1(config)# interface SSL-Proxy 0.8
ssl-mod-1(config-subif)# encapsulation dot1 8
ssl-mod-1(config-subif)# ip address 8.100.100.200 255.0.0.0
ssl-mod-1(config-subif)# standby 1 ip 8.100.100.126
ssl-mod-1(config-subif)# standby 1 timers 1 3
ssl-mod-1(config-subif)# standby 1 priority 90
ssl-mod-1(config-subif)# standby 1 preempt delay minimum 60
ssl-mod-1(config-subif)# standby 2 ip 8.100.100.127
ssl-mod-1(config-subif)# standby 2 timers 1 3
ssl-mod-1(config-subif)# standby 2 priority 110
ssl-mod-1(config-subif)# standby 2 preempt delay minimum 60
ssl-mod-1(config-subif)# exit
ssl-mod-1(config)# ip route 24.0.0.0 255.0.0.0 8.100.100.100
ssl-mod-1(config)#
```

SSL 2 Configuration

This example shows how to configure the proxy services and the VLAN on SSL 2:

```
ssl-mod-2# configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
ssl-mod-2(config)# ssl-proxy context Default
ssl-mod-2(config-context)# natpool client-nat 8.100.2.1 8.100.2.8 netmask 255.0.0.0
ssl-mod-2(config-context)# service ssl-offload
ssl-mod-2(config-ctx-ssl-proxy)# virtual ipaddr 8.100.100.110 protocol tcp port 443
secondary
ssl-mod-2(config-ctx-ssl-proxy)# server ipaddr 24.0.0.2 protocol tcp port 80
ssl-mod-2(config-ctx-ssl-proxy)# certificate rsa general-purpose trustpoint cert
ssl-mod-2(config-ctx-ssl-proxy)# nat client client-nat
ssl-mod-2(config-ctx-ssl-proxy)# inservice
ssl-mod-2(config-ctx-ssl-proxy)# exit
ssl-mod-2(config-context)# service ssl-offload-checkout
ssl-mod-2(config-ctx-ssl-proxy)# virtual ipaddr 8.100.100.111 protocol tcp port 443
secondary
ssl-mod-2(config-ctx-ssl-proxy)# server ipaddr 24.0.0.2 protocol tcp port 80
ssl-mod-2(config-ctx-ssl-proxy)# certificate rsa general-purpose trustpoint cert
ssl-mod-2(config-ctx-ssl-proxy)# nat client client-nat
ssl-mod-2(config-ctx-ssl-proxy)# inservice
ssl-mod-2(config-ctx-ssl-proxy)# exit
ssl-mod-2(config-context)# exit
ssl-mod-2(config)# interface SSL-Proxy 0.8
ssl-mod-2(config-subif)# encapsulation dot1q 8
ssl-mod-2(config-subif)# ip address 8.100.100.250 255.0.0.0
ssl-mod-2(config-subif)# standby priority 110
ssl-mod-2(config-subif)# standby 1 ip 8.100.100.126
ssl-mod-2(config-subif)# standby 1 timers 1 3
ssl-mod-2(config-subif)# standby 1 priority 110
ssl-mod-2(config-subif)# standby 1 preempt delay minimum 60
ssl-mod-2(config-subif)# standby 2 ip 8.100.100.127
ssl-mod-2(config-subif)# standby 2 timers 1 3
ssl-mod-2(config-subif)# standby 2 priority 90
ssl-mod-2(config-subif)# standby 2 preempt delay minimum 60
ssl-mod-2(config-subif)# exit
ssl-mod-2(config)# ip route 24.0.0.0 255.0.0.0 8.100.100.100
ssl-mod-2(config)#
```

Displaying Statistics

These examples show how to display statistics to show that load balancing is occurring in two SSL Services Module:

SSL 1

```
ssl-mod-1# show ssl-proxy stats service
No context name provided, assuming context 'Default'

Service ssl-offload SSL Statistics:
    conns attempted     :0              conns completed     :0
    full handshakes     :0              resumed handshakes  :0
    conns in handshake  :0              conns in data       :0
    renegs attempted    :0              conns in reneg      :0
    blocks encrypted    :0              bytes encrypted     :0
    blocks decrypted    :0              bytes decrypted     :0
    valid cache entry   :0              session limit exceed:0
    handshake failures  :0              data failures       :0
    fatal alerts rcvd   :0              fatal alerts sent   :0
```

```
    bad macs received   :0             pad errors          :0
    no-cipher alerts    :0             no-compress alerts  :0
    ver mismatch alerts :0

Service ssl-offload-checkout SSL Statistics:
    conns attempted     :3288          conns completed     :3286
    full handshakes     :3287          resumed handshakes  :0
    conns in handshake  :1             conns in data       :1
    renegs attempted    :0             conns in reneg      :0
    blocks encrypted    :41468         bytes encrypted     :57831402
    blocks decrypted    :3287          bytes decrypted     :289256
    valid cache entry   :253152        session limit exceed:0
    handshake failures  :0             data failures       :0
    fatal alerts rcvd   :0             fatal alerts sent   :0
    bad macs received   :0             pad errors          :0
    no-cipher alerts    :0             no-compress alerts  :0
    ver mismatch alerts :0

ssl-mod-1# show standby
Ethernet0/0.8 - Group 1
  State is Standby
    7 state changes, last state change 00:03:37
  Virtual IP address is 8.100.100.126
  Active virtual MAC address is 0000.0c07.ac01
    Local virtual MAC address is 0000.0c07.ac01 (default)
  Hello time 1 sec, hold time 3 sec
    Next hello sent in 0.004 secs
  Preemption enabled, delay min 60 secs
  Active router is 8.100.100.250, priority 110 (expires in 2.000 sec)
  Standby router is local
  Priority 90 (configured 90)
  IP redundancy name is "hsrp-Et0/0.8-1" (default)
Ethernet0/0.8 - Group 2
  State is Active
    2 state changes, last state change 01:53:29
  Virtual IP address is 8.100.100.127
  Active virtual MAC address is 0000.0c07.ac02
    Local virtual MAC address is 0000.0c07.ac02 (default)
  Hello time 1 sec, hold time 3 sec
    Next hello sent in 0.648 secs
  Preemption enabled, delay min 60 secs
  Active router is local
  Standby router is 8.100.100.250, priority 90 (expires in 2.644 sec)
  Priority 110 (configured 110)
  IP redundancy name is "hsrp-Et0/0.8-2" (default)
ssl-mod-1#
```

SSL 2

```
ssl-mod-2# show ssl-proxy stats service
No context name provided, assuming context 'Default'...

Service ssl-offload SSL Statistics:
    conns attempted     :4128          conns completed     :4126
    full handshakes     :4127          resumed handshakes  :0
    conns in handshake  :1             conns in data       :1
    renegs attempted    :0             conns in reneg      :0
```

```
    blocks encrypted    :51757          bytes encrypted     :72085513
    blocks decrypted    :4127           bytes decrypted     :363176
    valid cache entry   :136076         session limit exceed:0
    handshake failures  :0              data failures       :0
    fatal alerts rcvd   :0              fatal alerts sent   :0
    bad macs received   :0              pad errors          :0
    no-cipher alerts    :0              no-compress alerts  :0
    ver mismatch alerts :0

Service ssl-offload-checkout SSL Statistics:
    conns attempted     :0              conns completed     :0
    full handshakes     :0              resumed handshakes  :0
    conns in handshake  :0              conns in data       :3
    renegs attempted    :0              conns in reneg      :0
    blocks encrypted    :0              bytes encrypted     :0
    blocks decrypted    :0              bytes decrypted     :0
    valid cache entry   :126001         session limit exceed:0
    handshake failures  :0              data failures       :0
    fatal alerts rcvd   :0              fatal alerts sent   :0
    bad macs received   :0              pad errors          :0
    no-cipher alerts    :0              no-compress alerts  :0
    ver mismatch alerts :0

ssl-mod-2# show standby
Ethernet0/0.8 - Group 1
  State is Active
    2 state changes, last state change 02:23:54
  Virtual IP address is 8.100.100.126
  Active virtual MAC address is 0000.0c07.ac01
    Local virtual MAC address is 0000.0c07.ac01 (default)
  Hello time 1 sec, hold time 3 sec
    Next hello sent in 0.232 secs
  Preemption enabled, delay min 60 secs
  Active router is local
  Standby router is 8.100.100.200, priority 90 (expires in 2.232 sec)
  Priority 110 (configured 110)
  IP redundancy name is "hsrp-Et0/0.8-1" (default)
Ethernet0/0.8 - Group 2
  State is Standby
    10 state changes, last state change 00:03:34
  Virtual IP address is 8.100.100.127
  Active virtual MAC address is 0000.0c07.ac02
    Local virtual MAC address is 0000.0c07.ac02 (default)
  Hello time 1 sec, hold time 3 sec
    Next hello sent in 0.876 secs
  Preemption enabled, delay min 60 secs
  Active router is 8.100.100.200, priority 110 (expires in 2.876 sec)
  Standby router is local
  Priority 90 (configured 90)
  IP redundancy name is "hsrp-Et0/0.8-2" (default)
ssl-mod-2#
```

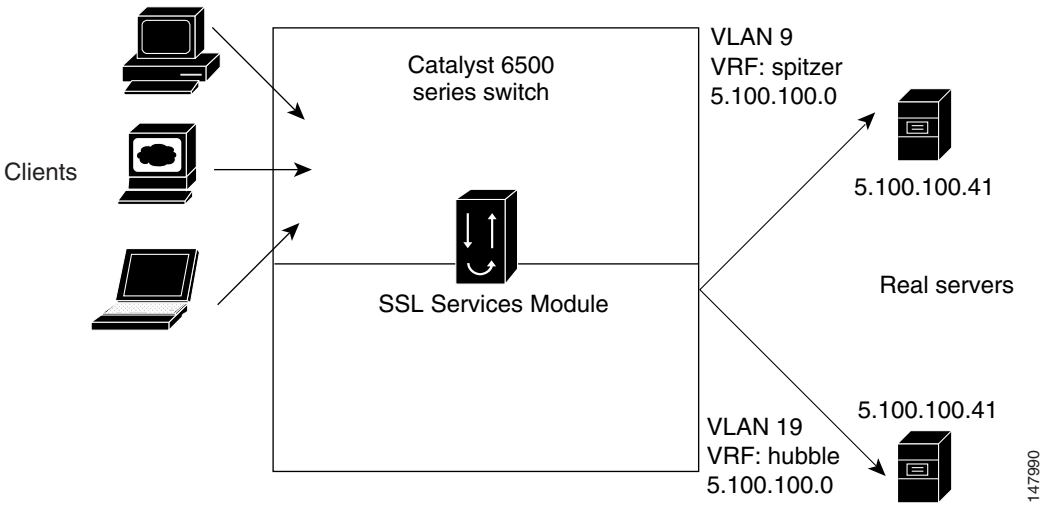
Virtualization with VRF Example

This section shows how to configure virtualization with VPN routing and forwarding (VRF).

Figure A-9 shows the following information:

- VRF “spitzer”
  - VLAN 9
  - IP address 5.100.100.0
  - assigned to context “spitzer”
- VRF “hubble”
  - VLAN 19
  - IP address 5.100.100.0
  - assigned to context “hubble”

Figure A-9 Virtualization with VRF



Configuring the Supervisor Engine

These examples show how to create VLAN 9 (for VRF “spitzer”) and VLAN 19 (for VRF “hubble”) and assign ports to the respective VLANs:

```
Router# configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
Router(config)# vlan 9
Router(config-vlan)# vlan 19
Router(config-vlan)# exit
Router(config)# interface fastEthernet 4/48
Router(config-if)# switchport
Router(config-if)# switchport access vlan 9
Router(config-if)# switchport mode access
Router(config-if)# no shutdown
Router(config-if)# exit
Router(config)# interface GigabitEthernet5/6
```

```
Router(config-if)# switchport
Router(config-if)# switchport access vlan 19
Router(config-if)# switchport mode access
Router(config-if)# no shutdown
Router(config-if)# exit
Router(config)#
```

This example shows how to allow VLANs 9 and 19 between the SSL Services Module and the supervisor engine:

```
Router# configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
Router(config)# ssl-proxy module 2 allowed-vlan 9,19
Router(config)# exit
Router#
```

Configuring the SSL Services Module

This example shows the initial configurations for the two different VRFs:

```
ssl-proxy# configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
ssl-proxy(config)# ip vrf hubble
ssl-proxy(config-vrf)# rd 300:300
ssl-proxy(config-vrf)# exit
ssl-proxy(config)# ip vrf spitzer
ssl-proxy(config-vrf)# rd 200:200
ssl-proxy(config-vrf)# end
ssl-proxy#
```

This example shows how to associate the VRFs with the SSL proxy subinterface:

```
ssl-proxy# configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
ssl-proxy(config)# interface SSL-Proxy0.9
ssl-proxy(config-subif)# encapsulation dot1Q 9
ssl-proxy(config-subif)# ip vrf forwarding spitzer
ssl-proxy(config-subif)# ip address 5.100.100.10 255.255.255.0
ssl-proxy(config-subif)# no cdp enable
ssl-proxy(config-subif)# no shut
ssl-proxy(config-subif)# no shutdown
ssl-proxy(config-subif)# exit
ssl-proxy(config)# interface SSL-Proxy0.19
ssl-proxy(config-subif)# encapsulation dot1Q 19
ssl-proxy(config-subif)# ip vrf forwarding hubble
ssl-proxy(config-subif)# ip address 5.100.100.10 255.255.255.0
ssl-proxy(config-subif)# no cdp enable
ssl-proxy(config-subif)# no shutdown
ssl-proxy(config-subif)# end
ssl-proxy#
```

This example shows how to configure context “hubble”:

```
ssl-proxy# configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
ssl-proxy(config)# ssl-proxy context hubble
ssl-proxy(config-context)# vrf-name hubble
ssl-proxy(config-context)# service hubble
ssl-proxy(config-ctx-ssl-proxy)# virtual ipaddr 3.100.100.108 protocol tcp port 443
ssl-proxy(config-ctx-ssl-proxy)# server ipaddr 5.100.100.41 protocol tcp port 80
ssl-proxy(config-ctx-ssl-proxy)# certificate rsa general-purpose trustpoint shuttle
```

```
ssl-proxy(config-ctx-ssl-proxy)# nat client hubble
ssl-proxy(config-ctx-ssl-proxy)# inservice
ssl-proxy(config-ctx-ssl-proxy)# exit
ssl-proxy(config-context)# natpool hubble 5.100.100.20 5.100.100.27 netmask 255.255.255.0
ssl-proxy(config-context)# end
ssl-proxy#
```

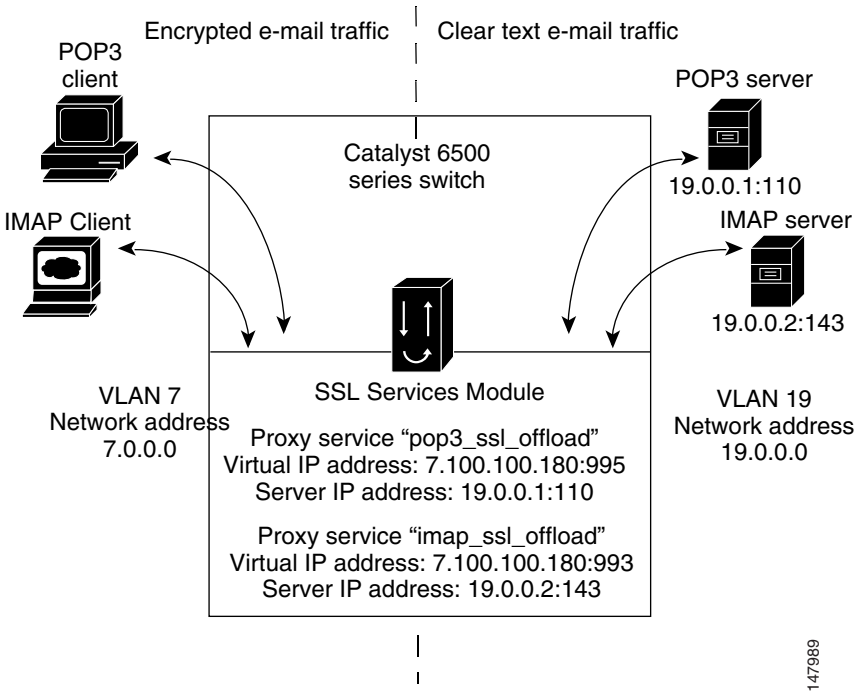
This example shows how to configure context “spitzer”:

```
ssl-proxy# configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
ssl-proxy(config)# ssl-proxy context spitzer
ssl-proxy(config-context)# vrf-name spitzer
ssl-proxy(config-context)# service spitzer
ssl-proxy(config-ctx-ssl-proxy)# virtual ipaddr 3.100.100.107 protocol tcp port 443
ssl-proxy(config-ctx-ssl-proxy)# server ipaddr 5.100.100.41 protocol tcp port 80
ssl-proxy(config-ctx-ssl-proxy)# certificate rsa general-purpose trustpoint shuttle
ssl-proxy(config-ctx-ssl-proxy)# nat client spitzer
ssl-proxy(config-ctx-ssl-proxy)# inservice
ssl-proxy(config-ctx-ssl-proxy)# exit
ssl-proxy(config-context)# natpool spitzer 5.100.100.20 5.100.100.27 netmask 255.255.255.0
ssl-proxy(config-context)# end
ssl-proxy#
```

Offloading Non-HTTP Protocols Example

This section shows how to configure the SSL Services Module to offload non-HTTP protocols. Figure A-10 shows the traffic flow from the POP3 and IMAP clients to the POP3 and IMAP servers.

Figure A-10 Offloading Non-HTTP Protocols



Configuring the Supervisor Engine

These examples show how to create client-side VLAN 7 and server-side VLAN 19 and assign ports to the respective VLANs:

```
Router# configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
Router(config)# vlan 7
Router(config-vlan)# vlan 19
Router(config-vlan)# interface FastEthernet3/29
Router(config-if)# switchport
Router(config-if)# switchport access vlan 19
Router(config-if)# switchport mode access
Router(config-if)# no shutdown
Router(config-if)# exit
Router(config)# interface FastEthernet3/38
Router(config-if)# switchport
Router(config-if)# switchport access vlan 7
Router(config-if)# switchport mode access
Router(config-if)# no shutdown
Router(config-if)# exit
```

This example shows how to allow VLANs 7 and 19 between the SSL Services Module and the supervisor engine:

```
Router(config)# ssl-proxy mod 7 allowed-vlan 7,19
Router(config)#
```

Configuring the SSL Services Module

This example shows how to configure the VLAN 7 (client-side) and VLAN 19 (server-side):

```
ssl-proxy(config)# interface ssl-proxy 0
ssl-proxy(config-if)# no shutdown
ssl-proxy(config-if)# interface SSL-Proxy0.7
ssl-proxy(config-subif)# encapsulation dot1Q 7
ssl-proxy(config-subif)# ip address 7.100.100.121 255.0.0.0
ssl-proxy(config-subif)# exit
ssl-proxy(config)# interface SSL-Proxy0.20
ssl-proxy(config-subif)# encapsulation dot1Q 19
ssl-proxy(config-subif)# ip address 19.0.0.100 255.0.0.0
ssl-proxy(config-subif)# exit
```

This example shows how to configure the SSL context:

```
ssl-proxy# configure terminal
ssl-proxy(config)# ssl-proxy context mail_ssl_offload
ssl-proxy(config-context)# description ssl pop3 /imap mail traffic offload
ssl-proxy(config-context)# exit
```

This example shows how to configure the NAT pool for client NAT:

```
ssl-proxy(config)# ssl-proxy context mail_ssl_offload
ssl-proxy(config-context)# natpool natpool 19.2.2.1 19.2.2.8 netmask 255.0.0.0
```

This example shows how to configure the SSL proxy service to offload encrypted IMAP traffic to the IMAP server at IP address 19.0.0.2 on port 143 (the clear text IMAP port) when the client connects to the proxy service at IP address 7.100.100.180 with SSL port 993:

```
ssl-proxy(config-context)# service imap_ssl_offload
ssl-proxy(config-ctx-ssl-proxy)# virtual ipaddr 7.100.100.180 protocol tcp port 993
ssl-proxy(config-ctx-ssl-proxy)# server ipaddr 19.0.0.2 protocol tcp port 143
ssl-proxy(config-ctx-ssl-proxy)# certificate rsa general-purpose trustpoint cert1024
ssl-proxy(config-ctx-ssl-proxy)# nat client natpool
ssl-proxy(config-ctx-ssl-proxy)# inservice
ssl-proxy(config-ctx-ssl-proxy)# exit
ssl-proxy(config-context)#
```

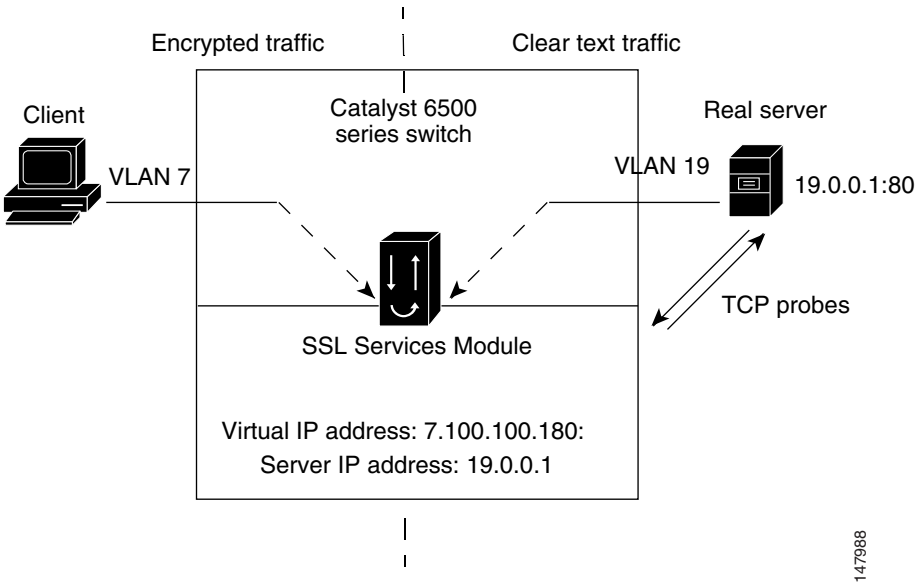
This example shows how to configure the SSL proxy service to offload encrypted POP3 traffic to the POP3 server at IP address 19.0.0.1 on port 110 (the clear text POP3 port) when the client connects to the proxy service at IP address at 7.100.100.180 with SSL port 995:

```
ssl-proxy(config-context)# service pop3_ssl_offload
ssl-proxy(config-ctx-ssl-proxy)# virtual ipaddr 7.100.100.180 protocol tcp port 995
ssl-proxy(config-ctx-ssl-proxy)# server ipaddr 19.0.0.1 protocol tcp port 110
ssl-proxy(config-ctx-ssl-proxy)# certificate rsa general-purpose trustpoint cert1024
ssl-proxy(config-ctx-ssl-proxy)# nat client natpool
ssl-proxy(config-ctx-ssl-proxy)# inservice
ssl-proxy(config-ctx-ssl-proxy)# exit
ssl-proxy(config-context)#
```

Health Probe Example

This section shows how to configure the SSL Services Module to probe the server to detect a server failure. Figure A-11 shows the traffic flow between the real server and the SSL Services Module.

Figure A-11 TCP Health Probe



Configuring the Supervisor Engine

These examples show how to create client-side VLAN 7 and server-side VLAN 19 and assign ports to the respective VLANs:

```
Router(config)# vlan 7
Router(config-vlan)# vlan 19
Router(config-vlan)# interface FastEthernet3/29
Router(config-if)# switchport
Router(config-if)# switchport access vlan 19
Router(config-if)# switchport mode access
Router(config-if)# no shutdown
Router(config-if)# exit
Router(config)# interface FastEthernet3/38
Router(config-if)# switchport
Router(config-if)# switchport access vlan 7
Router(config-if)# switchport mode access
Router(config-if)# no shutdown
Router(config-if)# exit
```

This example shows how to allow VLANs 7 and 19 between the SSL Services Module and the supervisor engine:

```
Router(config)# ssl-proxy mod 7 allowed-vlan 7,19
Router(config)#
```

Configuring the SSL Services Module

This example shows how to configure the VLAN 7 (client-side) and VLAN 19 (server-side):

```
ssl-proxy(config)# interface SSL-Proxy0
ssl-proxy(config-if)# no shutdown
ssl-proxy(config-if)# exit
ssl-proxy(config)# interface SSL-Proxy0.7
ssl-proxy(config-subif)# encapsulation dot1Q 7
ssl-proxy(config-subif)# ip address 7.100.100.121 255.0.0.0
ssl-proxy(config-subif)# no shutdown
ssl-proxy(config-subif)# exit
ssl-proxy(config)# interface SSL-Proxy0.20
ssl-proxy(config-subif)# encapsulation dot1Q 19
ssl-proxy(config-subif)# ip address 19.0.0.100 255.0.0.0
ssl-proxy(config-subif)# no shutdown
ssl-proxy(config-subif)# exit
ssl-proxy(config)#
```

This example shows how to configure the SSL context:

```
ssl-proxy(config)# ssl-proxy context ssl
ssl-proxy(config-context)# description ssl-offload with tcp health probe
```

This example shows how to configure the NAT pool with IP addresses from the server VLAN:

```
ssl-proxy(config-context)# natpool natpool 19.2.2.1 19.2.2.8 netmask 255.0.0.0
ssl-proxy(config-context)#
```

Health Probe with Default (Server) Port

This example shows how to configure the SSL proxy service "sslloffload" and apply the TCP health probe policy:

```
ssl-proxy(config-context)# service sslloffload
ssl-proxy(config-ctx-ssl-proxy)# virtual ipaddr 7.100.100.180 protocol tcp port 443
ssl-proxy(config-ctx-ssl-proxy)# server ipaddr 19.0.0.1 protocol tcp port 80
ssl-proxy(config-ctx-ssl-proxy)# certificate rsa general-purpose trustpoint cert1024
ssl-proxy(config-ctx-ssl-proxy)# policy health-probe tcp probe1
ssl-proxy(config-ctx-ssl-proxy)# nat client natpool
ssl-proxy(config-ctx-ssl-proxy)# inservice
ssl-proxy(config-ctx-ssl-proxy)# exit
ssl-proxy(config-context)#
```

This example shows how to configure a TCP health probe to check if the server that is listening on port 80 (server port) is able to accept TCP connections successfully:

```
ssl-proxy(config-context)# policy health-probe tcp probe1
ssl-proxy(config-ctx-tcp-probe)# exit
ssl-proxy(config-context)#
```

The following example shows the state of the SSL proxy service:

```
ssl-proxy# show ssl-proxy service sslloffload context ssl
Service id: 1, bound_service_id: 257
Virtual IP: 7.100.100.180, port: 443
Server IP: 19.0.0.1, port: 80
TCP Health Probe Policy: probe1
Nat pool: natpool
rsa-general-purpose certificate trustpoint: cert1024
  Certificate chain for new connections:
    Certificate:
       Key Label: cert1024.key, 1024-bit, exportable
       Key Timestamp: 05:18:23 UTC Dec 30 2005
       Serial Number: 12F332E200000000000D
    Root CA Certificate:
       Serial Number: 6522F512C30E078447D8AFC35567B101
  Certificate chain complete
Context name: ssl
Context Id  : 2
Admin Status: up
Operation Status: up
```

Health Probe with Port Mismatch

This example shows how to configure the SSL proxy service "sslloffload" and apply the TCP health probe policy:

```
ssl-proxy(config-context)# service sslloffload
ssl-proxy(config-ctx-ssl-proxy)# virtual ipaddr 7.100.100.180 protocol tcp port 443
ssl-proxy(config-ctx-ssl-proxy)# server ipaddr 19.0.0.1 protocol tcp port 80
ssl-proxy(config-ctx-ssl-proxy)# certificate rsa general-purpose trustpoint cert1024
ssl-proxy(config-ctx-ssl-proxy)# policy health-probe tcp probe1
ssl-proxy(config-ctx-ssl-proxy)# nat client natpool
ssl-proxy(config-ctx-ssl-proxy)# inservice
ssl-proxy(config-ctx-ssl-proxy)# exit
ssl-proxy(config-context)#
```

This example shows how to configure a TCP health probe to check if the server (19.0.0.1) that is listening on port 81 is able to accept TCP connections successfully:

```
ssl-proxy(config-context)# policy health-probe tcp probe1
ssl-proxy(config-ctx-tcp-probe)# 81
Warning: Port in the service ssloffload configuration (80) differs from the port in the
health probe configuration (81)
ssl-proxy(config-ctx-tcp-probe)# exit
ssl-proxy(config-context)#
```

Health Probe with No NAT Server Configured on Service

In this example, the port used for the TCP health probe is 80 and the IP address is 7.100.100.180.

This example shows how to configure the SSL proxy service "ssloffload" with **no nat server** and apply the TCP health probe policy:

```
ssl-proxy(config-context)# service ssloffload
ssl-proxy(config-ctx-ssl-proxy)# virtual ipaddr 7.100.100.180 protocol tcp port 443
secondary
ssl-proxy(config-ctx-ssl-proxy)# server ipaddr 19.0.0.1 protocol tcp port 80
ssl-proxy(config-ctx-ssl-proxy)# certificate rsa general-purpose trustpoint cert1024
ssl-proxy(config-ctx-ssl-proxy)# policy health-probe tcp probe1
ssl-proxy(config-ctx-ssl-proxy)# no nat server
ssl-proxy(config-ctx-ssl-proxy)# nat client natpool
ssl-proxy(config-ctx-ssl-proxy)# inservice
ssl-proxy(config-ctx-ssl-proxy)# exit
ssl-proxy(config-context)#
```

This example shows how to configure a TCP health probe to check if the server that is listening on port 80 is able to accept TCP connections successfully:

```
ssl-proxy(config-context)# policy health-probe tcp probe1
ssl-proxy(config-ctx-tcp-probe)# exit
ssl-proxy(config-context)#
```

Health Probe with Port Mismatch and No NAT Server Configured on Service

In this example, the port used for the TCP health probe is 444 and the IP address is 7.100.100.180.

This example shows how to configure the SSL proxy service "ssloffload" with **no nat server** and apply the TCP health probe policy:

```
ssl-proxy(config-context)# service ssloffload
ssl-proxy(config-ctx-ssl-proxy)# virtual ipaddr 7.100.100.180 protocol tcp port 443
secondary
ssl-proxy(config-ctx-ssl-proxy)# server ipaddr 19.0.0.1 protocol tcp port 80
ssl-proxy(config-ctx-ssl-proxy)# certificate rsa general-purpose trustpoint cert1024
ssl-proxy(config-ctx-ssl-proxy)# policy health-probe tcp probe1
ssl-proxy(config-ctx-ssl-proxy)# no nat server
ssl-proxy(config-ctx-ssl-proxy)# nat client natpool
ssl-proxy(config-ctx-ssl-proxy)# inservice
ssl-proxy(config-ctx-ssl-proxy)# exit
ssl-proxy(config-context)#
```

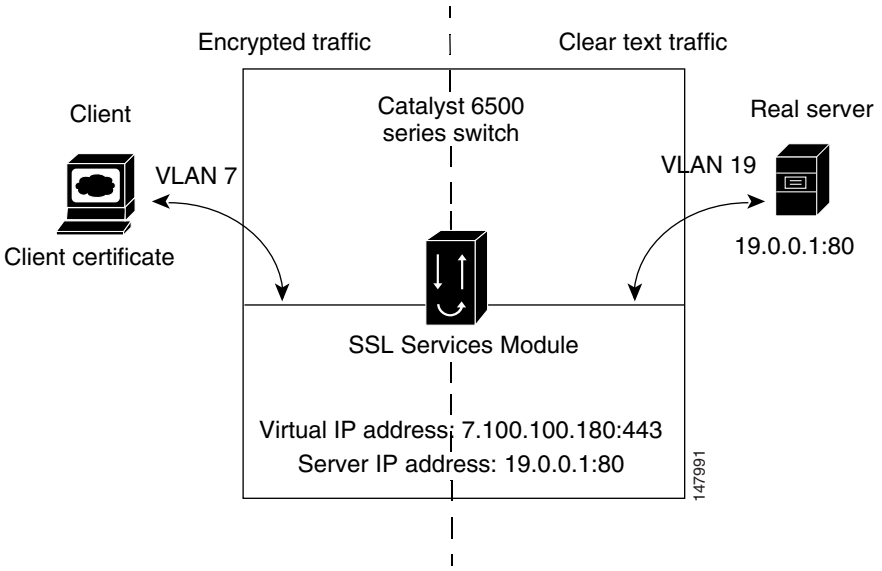
This example shows how to configure a TCP health probe to check if the server that is listening on port 81 is able to accept TCP connections successfully:

```
ssl-proxy(config-context)# policy health-probe tcp probe1
ssl-proxy(config-ctx-tcp-probe)# 444
ssl-proxy(config-ctx-tcp-probe)# exit
Warning: Port in the service sslloffload configuration (80) differs from the port in the
health probe configuration (444)
ssl-proxy(config-context)#
```

Client Authentication Example

This section shows how to configure client authentication. Figure A-12 shows the network layout and traffic flow to the SSL Services Module.

Figure A-12 Client Authentication



Configuring the Supervisor Engine

These examples show how to create client-side VLAN 7 and server-side VLAN 19 and assign ports to the respective VLANs:

```
Router# configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
Router(config)# vlan 7
Router(config-vlan)# vlan 19
Router(config-vlan)# interface FastEthernet3/29
Router(config-if)# switchport
Router(config-if)# switchport access vlan 19
Router(config-if)# switchport mode access
Router(config-if)# no shutdown
Router(config-if)# exit
```

```
Router(config)# interface FastEthernet3/38
Router(config-if)# switchport
Router(config-if)# switchport access vlan 7
Router(config-if)# switchport mode access
Router(config-if)# no shutdown
Router(config-if)# exit
```

This example shows how to allow VLANs 7 and 19 between the SSL Services Module and the supervisor engine:

```
Router(config)# ssl-proxy mod 7 allowed-vlan 7,19
Router(config)#
```

Configuring the SSL Services Module

This example shows how to configure the VLAN 7 (client-side) and VLAN 19 (server-side):

```
ssl-proxy# configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
ssl-proxy(config)# interface ssl-proxy0.7
ssl-proxy(config-subif)# encapsulation dot1Q 7
ssl-proxy(config-subif)# ip address 7.100.100.121 255.0.0.0
ssl-proxy(config-subif)# exit
ssl-proxy(config)# interface ssl-proxy0.20
ssl-proxy(config-subif)# encapsulation dot1Q 19
ssl-proxy(config-subif)# ip address 19.0.0.100 255.0.0.0
ssl-proxy(config-subif)# end
ssl-proxy(config)#
```

This example shows how to configure the CA certificate and assign a trustpoint to the certificate. This signed certificate from the certificate authority is used at the client browser.

Note This trustpoint cannot have server and private keys. The trustpoint that is used with this proxy service cannot be used for client-authentication purposes because the trustpoint contains a private key and a server certificate.

```
ssl-proxy(config)# crypto pki trustpoint ca_cert
ssl-proxy(ca-trustpoint)# enrollment terminal
ssl-proxy(ca-trustpoint)# revocation-check none
ssl-proxy(ca-trustpoint)# exit
ssl-proxy(config)# crypto pki auth ca_cert

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

-----BEGIN CERTIFICATE-----
MIIDjjCCAvegAwIBAgIQZSL1EsMOB4RH2K/DVWexATANBgkqhkiG9w0BAQUFADCB
jzEgMB4GCSqGSIb3DQEJARYRc2ltcHNvbkBjaXNjby5jb20xCzAJBgNVBAYTAlVT
MQswCQYDVQQIEwJDQTERMA8GA1UEBxMIU2FuIEpvc2UxDjAMBgNVBAoTBUNpc2Nv
MQwwCgYDVQQLEwNXQlUxIDAeBgNVBAMTF3NpbXBzb24tZGV2dGVzdC1yb290LWNh
MB4XDTA1MDYxNDA2NTUyMFoXDTE1MDYxNDA3MDMyN1owgY8xIDAeBgkqhkiG9w0B
CQEWEXNpbXBzb25AY2lzY28uY29tMQswCQYDVQQGEwJVUzELMAkGA1UECBMCQ0Ex
ETAPBgNVBAcTCFNhbiBKb3NlMQ4wDAYDVQQKEwVDaXNjbzEMMAoGA1UECxMDV0JV
MSAwHgYDVQQDExdzaW1wc29uLWRldnRlc3Qtcm9vdC1jYTCBnzANBgkqhkiG9w0B
AQEFAAOBjQAwgYkCgYEAsbMVYdM/XxtHLXRHT1S0mg8K/v/ZNQ5uCCCNZ7l1nBnj
7vAsIsGR8HoSsPQQFi3/iYO2GjiBEqdikFevWqyrOdehrahpVbkL3xp1dZZP08ZN
psZgbYqkyizNljuOKS5Y66jvZ2jMR5yJH0hQHBiNjtqcjCD4XT6JWGH6TIZ7veMC
```

```
AwEAAaOB6DCB5TALBgNVHQ8EBAMCAcYwDwYDVR0TAQH/BAUwAwEB/zAdBgNVHQ4E
FgQUTJGjnYT0iOGmpbcjmROSUo6AvhowgZMGA1UdHwSBizCBiDBBoD+gPYY7aHR0
cDovL3dpbjJrLXJvb3QtY2EvQ2VydEVucm9sbC9zaW1wc29uLWRldnRlc3Qtcm9v
dC1jYS5jcmwwQ6BBoD+GPWZpbGU6Ly9cXHdpbjJrLXJvb3QtY2FcQ2VydEVucm9s
bFxzaW1wc29uLWRldnRlc3Qtcm9vdC1jYS5jcmwwEAYJKwYBBAGCNxUBBAMCAQAw
DQYJKoZIhvcNAQEFBQADgYEAOoWOfrM/4H4ltFYpHtowGso+V+YuANw7OsWTR1Ay
qIYNzSTO4xDyApE25PrGL3vYHsMTWWRnXMsp+5xujU746/Y6pmnDvSWykJlerrfG
SkQVCkcm0ygFtm/0VUvA/4582wbhE6BjH1WYKIHTMVmhwkeQZk2StoCdhMwX9a2l
4S4=
-----END CERTIFICATE-----

Certificate has the following attributes:
Fingerprint: 9913F4FA 99CC239B C9D72DF1 6764B7CD
% Do you accept this certificate? [yes/no]: yes
Trustpoint CA certificate accepted.
% Certificate successfully imported
```

This example shows how to create the certificate authority pool and add a trusted certificate authority to the pool.

```
ssl-proxy(config-context)# pool ca trusted_ca_pool
ssl-proxy(config-ctx-ca-pool)# ca trustpoint ca_cert
ssl-proxy(config-ctx-ca-pool)# exit
```

This example shows how to configure the NAT pool for client-side NAT:

```
ssl-proxy(config)# ssl-proxy context ssl
ssl-proxy(config-context)# natpool natpool 19.2.2.1 19.2.2.8 netmask 255.0.0.0
```

This example shows the SSL proxy service with client authentication with full verification:

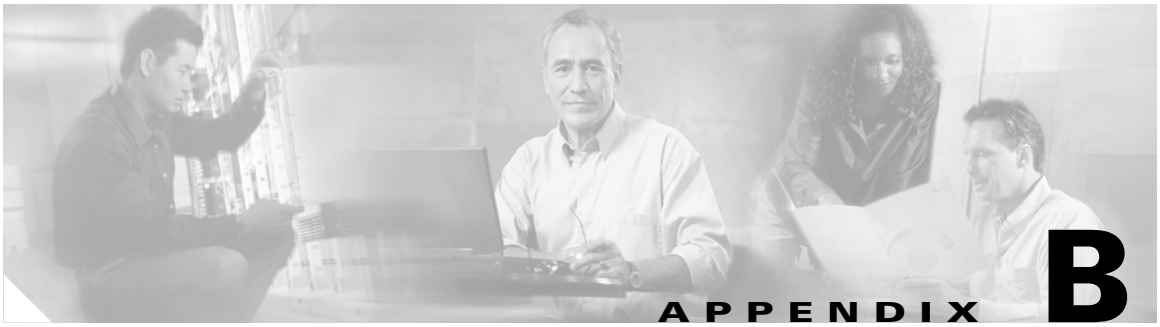
```
ssl-proxy(config-context)# service ssl-1
ssl-proxy(config-ctx-ssl-proxy)# virtual ipadd 7.100.100.180 pro tcp port 443
ssl-proxy(config-ctx-ssl-proxy)# server ipaddr 19.0.0.1 protocol tcp port 80
ssl-proxy(config-ctx-ssl-proxy)# certificate rsa general-purpose trustpoint cert1024
ssl-proxy(config-ctx-ssl-proxy)# nat client natpool
ssl-proxy(config-ctx-ssl-proxy)# inservice
ssl-proxy(config-ctx-ssl-proxy)# authenticate verify all
ssl-proxy(config-ctx-ssl-proxy)# trusted-ca trusted_ca_pool
ssl-proxy(config-context)# ^Z
ssl-proxy#
ssl-proxy# show ssl-proxy service ssl-1 context ssl
Service id: 6, bound_service_id: 262
Virtual IP: 7.100.100.180, port: 443
Server IP: 19.0.0.1, port: 80
Nat pool: natpool
Certificate authority pool: trusted_ca_pool
  CA pool complete
rsa-general-purpose certificate trustpoint: cert1024
  Certificate chain for new connections:
    Certificate:
       Key Label: cert1024.key, 1024-bit, exportable
       Key Timestamp: 05:18:23 UTC Dec 30 2005
       Serial Number: 12F332E200000000000D
    Root CA Certificate:
       Serial Number: 6522F512C30E078447D8AFC35567B101
  Certificate chain complete
Certificate authentication type: All attributes (like CRL) are verified
Context name: ssl
Context Id  : 6
Admin Status: up
Operation Status: up
ssl-proxy#
```

Successful client authentication causes the "cert approved" counter to increase when you enter the **show ssl-proxy stats ssl** command:

```
...

SSL PKI Statistics:
    number of malloc    : 6256          number of free      : 6226
    ssl buf allocated   : 4             ssl buf freed       : 1

    Peer Certificate Verify Statistics:
    cert approved       : 3943          cert disapproved    : 2101
```



Upgrading the Images

The compact Flash on the SSL Services Module has two bootable partitions: application partition (AP) and maintenance partition (MP). By default, the application partition boots every time. The application partition contains the binaries necessary to run the SSL image. The maintenance partition is booted if you need to upgrade the application partition.

You can upgrade both the application software and the maintenance software. However, you are not required to upgrade both images at the same time. Refer to the release notes for the SSL Services Module for the latest application partition and maintenance partition software versions.

The entire application and maintenance partitions are stored on the FTP or TFTP server. The images are downloaded and extracted to the application partition or maintenance partition, depending on which image is being upgraded.

To upgrade the application partition, change the boot sequence to boot the module from the maintenance partition. To upgrade the maintenance partition, change the boot sequence to boot the module from the application partition. Set the boot sequence for the module by using the supervisor engine CLI commands. The maintenance partition downloads and installs the application image. The supervisor engine must be executing the run-time image to provide network access to the maintenance partition.

Before starting the upgrade process, you will need to download the application partition image or maintenance partition image to the TFTP server.

A TFTP or FTP server is required to copy the images. The TFTP server should be connected to the switch, and the port connecting to the TFTP server should be included in any VLAN on the switch.

These sections describe how to upgrade the images:

- Upgrading the Application Software, page B-1.
- Upgrading the Maintenance Software, page B-5.

Upgrading the Application Software

How you upgrade the application software depends on whether you are using Cisco IOS software or the Catalyst operating system software.

The following sections describe how to upgrade the application software from the CLI for each switch operating system:

- Cisco IOS Software, page B-2
- Catalyst Operating System Software, page B-4

Cisco IOS Software

Note Do not reset the module until the image is upgraded. The total time to upgrade the image takes up to eight minutes.

To upgrade the application partition software, perform this task:

	Command	Purpose
Step 1	`Router# hw-module module` *mod* `reset cf:1`	Reboots the module from the maintenance partition. **Note** It is normal to see messages such as "Press Key" on the module console after entering this command.
Step 2	`Router# show module`	Displays that the maintenance partition for the module has booted.
Step 3	`Router# copy tftp: pclc#`*mod*`-fs:`	Downloads the image.
Step 4	`Router# hw-module module` *mod* `reset`	Resets the module. **Note** Do not reset the module until the "You can now reset the module" message displays on the console. Resetting the module before this message displays will cause the upgrade to fail.
Step 5	`Router# show module`	Verifies that the application partition for the module has booted.

This example shows how to upgrade the application partition software:

```
Router# hw-module module 6 reset cf:1
hw mod 6 reset cf:1
Device BOOT variable for reset = <cf:1>
Warning: Device list is not verified.

Proceed with reload of module? [confirm]y

% reset issued for module 6

02:11:18: SP: The PC in slot 6 is shutting down. Please wait ...
02:11:31: SP: PC shutdown completed for module 6
02:11:31: %C6KPWR-SP-4-DISABLED: power to module in slot 6 set off (Reset)
02:14:21: SP: OS_BOOT_STATUS(6) MP OS Boot Status: finished booting
02:14:28: %DIAG-SP-6-RUN_MINIMUM: Module 6: Running Minimum Online Diagnostics...
02:14:34: %DIAG-SP-6-DIAG_OK: Module 6: Passed Online Diagnostics
02:14:34: %OIR-SP-6-INSCARD: Card inserted in slot 6, interfaces are now online

Router# show module
Mod Ports Card Type                                Model              Serial No.
--- ----- -------------------------------------- ------------------ -----------
  1    2  Catalyst 6000 supervisor 2 (Active)    WS-X6K-S2U-MSFC2   SAD055006RZ
  2   48  48 port 10/100 mb RJ45                 WS-X6348-RJ-45     SAL052794UW
  6    1  SSL Module (MP)                        WS-SVC-SSL-1       SAD060702VK

...<output truncated>...

Router# copy tftp: pclc#6-fs:
copy tftp: pclc#6-fs:
Address or name of remote host []? 10.1.1.1
```

```
Source filename []? c6svc-ssl-k9y9.1-x-y.bin

Destination filename [c6svc-ssl-k9y9.1-x-y.bin]?

Accessing tftp://10.1.1.1/c6svc-ssl-k9y9.1-x-y.bin...
Loading c6svc-ssl-k9y9.1-x-y.bin from 10.1.1.1 (via Vlan2):
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

<output truncated>

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
[OK - 14918353 bytes]

14918353 bytes copied in 643.232 secs (23193 bytes/sec)
Router#
02:29:23: %SVCLC-SP-5-STRRECVD: mod 6: <Application upgrade has started>
02:29:23: %SVCLC-SP-5-STRRECVD: mod 6: <Do not reset the module till upgrade completes!!>
02:36:07: %SVCLC-SP-5-STRRECVD: mod 6: <Application upgrade has succeded>
02:36:07: %SVCLC-SP-5-STRRECVD: mod 6: <You can now reset the module>>

Router# hw-module module 6 reset
Device BOOT variable for reset = <empty>
Warning:Device list is not verified.

Proceed with reload of module? [confirm]y
% reset issued for module 6
Router#
02:36:57:SP:The PC in slot 6 is shutting down. Please wait ...
02:37:17:SP:PC shutdown completed for module 6
02:37:17:%C6KPWR-SP-4-DISABLED:power to module in slot 6 set off (Reset)
02:38:39:SP:OS_BOOT_STATUS(6) AP OS Boot Status:finished booting
02:39:27:%DIAG-SP-6-RUN_COMPLETE:Module 6:Running Complete Online Diagnostics...
02:39:29:%DIAG-SP-6-DIAG_OK:Module 6:Passed Online Diagnostics
02:39:29:%OIR-SP-6-INSCARD:Card inserted in slot 6, interfaces are now online

Router# show module

Mod Ports Card Type                              Model              Serial No.
--- ----- -------------------------------------- ------------------ -----------
  1    2  Catalyst 6000 supervisor 2 (Active)    WS-X6K-S2U-MSFC2   SAD055006RZ
  2   48  48 port 10/100 mb RJ45                 WS-X6348-RJ-45     SAL052794UW
  6    1  SSL Module                             WS-SVC-SSL-1       SAD060702VK

...<output truncated>...
```

Catalyst Operating System Software

Note Do not reset the module until the image is upgraded. The total time to upgrade the image takes up to eight minutes.

To upgrade the application partition software, perform this task:

	Command	Purpose
Step 1	`Console (enable) set boot device cf:1 mod`	Sets the module to boot the maintenance partition.
Step 2	`Console (enable) reset mod`	Resets the module to the maintenance partition. **Note** The SUP_OSBOOTSTATUS system message shows that the maintenance partition (MP) has booted.
Step 3	`Console (enable) session [mod]`	Access the MSFC from the switch CLI using a Telnet session[1].
Step 4	`Router# copy tftp: pclc#mod-fs:`	Downloads the image.
Step 5	`Router# exit`	Exits the MSFC CLI and returns to the switch CLI.
Step 6	`Console (enable) set boot device cf:4 mod`	Sets the module to boot the application partition.
Step 7	`Console (enable) reset mod`	Resets the module to the application partition. **Note** Do not reset the module until the "You can now reset the module" message displays on the console. Resetting the module before this message displays will cause the upgrade to fail. **Note** The SUP_OSBOOTSTATUS system message shows that the application partition (AP) has booted.

1. To access the MSFC from the switch CLI directly connected to the supervisor engine console port, enter the **switch console** *mod* command. To exit from the MSFC CLI and return to the switch CLI, press **Ctrl-C** three times at the Router> prompt.

This example shows how to upgrade the application partition software:

```
Console> (enable) set boot device cf:1 6
Device BOOT variable = cf:1
Memory-test set to PARTIAL
Warning:Device list is not verified but still set in the boot string.
Console> (enable)
Console> (enable) reset 6 cf:1
This command will reset module 6.
Unsaved configuration on module 6 will be lost
Do you want to continue (y/n) [n]? y
Module 6 shut down in progress, please don't remove module until shutdown completed.
Console> (enable) Module 6 shutdown completed. Module resetting...
2003 Jan 17 08:34:07 %SYS-3-SUP_OSBOOTSTATUS:MP OS Boot Status:finished booting
2003 Jan 17 08:34:23 %SYS-5-MOD_OK:Module 6 is online
2003 Jan 17 08:34:23 %DTP-5-TRUNKPORTON:Port 6/1 has become dot1q trunk
```

```
Console> (enable) session 15
Trying Router-15...
Connected to Router-15.
Type ^C^C^C to switch back...
Router>

Router# copy tftp: pclc#6-fs:
copy tftp: pclc#6-fs:
Address or name of remote host []? 10.1.1.1

Source filename []? c6svc-ssl-k9y9.1-x-y.bin

Destination filename [c6svc-ssl-k9y9.1-x-y.bin]?

Accessing tftp://10.1.1.1/c6svc-ssl-k9y9.1-x-y.bin...
Loading c6svc-ssl-k9y9.1-x-y.bin from 10.1.1.1 (via Vlan2):
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

<output truncated>

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
[OK - 14918353 bytes]

14918353 bytes copied in 643.232 secs (23193 bytes/sec)
Router#
02:29:23: %SVCLC-SP-5-STRRECVD: mod 6: <Application upgrade has started>
02:29:23: %SVCLC-SP-5-STRRECVD: mod 6: <Do not reset the module till upgrade completes!!>
02:36:07: %SVCLC-SP-5-STRRECVD: mod 6: <Application upgrade has succeded>
02:36:07: %SVCLC-SP-5-STRRECVD: mod 6: <You can now reset the module>>
Router# exit
Console> (enable) set boot device cf:4 6
Device BOOT variable = cf:4
Memory-test set to PARTIAL
Warning:Device list is not verified but still set in the boot string.
Console> (enable) reset 6
This command will reset module 6.
Unsaved configuration on module 6 will be lost
Do you want to continue (y/n) [n]? y
Module 6 shut down in progress, please don't remove module until shutdown completed.
Console> (enable) Module 6 shutdown completed. Module resetting...
2003 Jan 17 08:36:58 %SYS-3-SUP_OSBOOTSTATUS:AP OS Boot Status:finished booting
2003 Jan 17 08:37:51 %SYS-5-MOD_OK:Module 6 is online
2003 Jan 17 08:37:51 %DTP-5-TRUNKPORTON:Port 6/1 has become dot1q trunk
```

Upgrading the Maintenance Software

How you upgrade the maintenance software depends on whether you are using Cisco IOS software or the Catalyst operating system software.

The following sections describe how to upgrade the maintenance software from the CLI for each switch operating system:

- Cisco IOS Software, page B-6
- Catalyst OS Software, page B-7

Cisco IOS Software

Note Do not reset the module until the image is upgraded. The total time to upgrade the image takes up to eight minutes.

To upgrade the maintenance partition software, perform this task:

	Command	Purpose
Step 1	Router# **hw-module module** *mod* **reset**	Reboots the module from the application partition.
Step 2	Router# **copy tftp: pclc#***mod***-fs:**	Downloads the image.
Step 3	Router# **hw-module module** *mod* **reset cf:1**	Resets the module in the maintenance partition. **Note** Do not reset the module until the "Upgrade of MP was successful. You can now boot MP" message displays on the console. Resetting the module before this message displays will cause the upgrade to fail.
Step 4	Router# **show module**	Verifies that the maintenance partition for the module has booted.

This example shows how to upgrade the maintenance partition software:

```
Router# hw module 6 reset
Device BOOT variable for reset = <empty>
Warning:Device list is not verified.
Proceed with reload of module? [confirm]y
% reset issued for module 6
Router#
02:36:57:SP:The PC in slot 6 is shutting down. Please wait ...
02:37:17:SP:PC shutdown completed for module 6
02:37:17:%C6KPWR-SP-4-DISABLED:power to module in slot 6 set off (Reset)
1w0d:SP:OS_BOOT_STATUS(6) AP OS Boot Status:finished booting
1w0d:%OIR-SP-6-INSCARD:Card inserted in slot 6, interfaces are now online
Router# copy tftp:pclc#6-fs:
Address or name of remote host []? 10.1.1.1
Source filename []? mp.1-2-0-16.bin.gz
Destination filename [mp.1-2-0-16.bin.gz]?
Accessing tftp://10.1.1.1/mp.1-2-0-16.bin.gz...
Loading mp.1-2-0-16.bin.gz from 10.1.1.1 (via Vlan2):
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
<output truncated>
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
[OK - 9818951 bytes]
9818951 bytes copied in 164.388 secs (59730 bytes/sec)
ssl-proxy>
1w0d:%SVCLC-SP-6-STRRECVD:mod 6:<MP upgrade started. Do not reset the card.>
1w0d:%SVCLC-SP-6-STRRECVD:mod 6:<Upgrade of MP was successful. You can now boot MP.>
Router# hw mod 6 reset cf:1
Device BOOT variable for reset = <cf:1>
Warning:Device list is not verified.
Proceed with reload of module? [confirm]y
% reset issued for module 6
```

```
Router# show module
Mod Ports Card Type                                Model              Serial No.
--- ----- ---------------------------------------- ------------------ -----------
1   2     Catalyst 6000 supervisor 2 (Active)    WS-X6K-S2U-MSFC2   SAD055006RZ
2   48    48 port 10/100 mb RJ45                  WS-X6348-RJ-45     SAL052794UW
6   1     SSL Module (MP)                        WS-SVC-SSL-1       SAD060702VK

...<output truncated>...
```

Catalyst OS Software

Note Do not reset the module until the image is upgraded. The total time to upgrade the image takes up to 8 minutes.

To upgrade the maintenance partition software, perform this task:

	Command	Purpose
Step 1	`Console (enable) set boot device cf:4 mod`	Sets the module to boot the application partition.
Step 2	`Console (enable) reset mod`	Resets the module to the application partition. **Note** The SUP_OSBOOTSTATUS system message shows that the application partition (AP) has booted.
Step 3	`Console (enable) session [mod]`	Access the MSFC from the switch CLI using a Telnet session[1].
Step 4	`Router# copy tftp: pclc#mod-fs:`	Downloads the image.
Step 5	`Router# exit`	Exits the MSFC CLI and returns to the switch CLI.
Step 6	`Console (enable) set boot device cf:1 mod`	Sets the module to boot the maintenance partition.
Step 7	`Console (enable) reset mod`	Resets the module to the maintenance partition. **Note** Do not reset the module until the "Upgrade of MP was successful. You can now boot MP" message displays on the console. Resetting the module before this message displays will cause the upgrade to fail. **Note** The SUP_OSBOOTSTATUS system message shows that the maintenance partition (MP) has booted.

1. To access the MSFC from the console that is directly connected to the supervisor engine console port, enter the **switch console** *mod* command. To exit from the MSFC CLI and return to the switch console, press **Ctrl-C** three times at the Router> prompt.

This example shows how to upgrade the maintenance partition software:

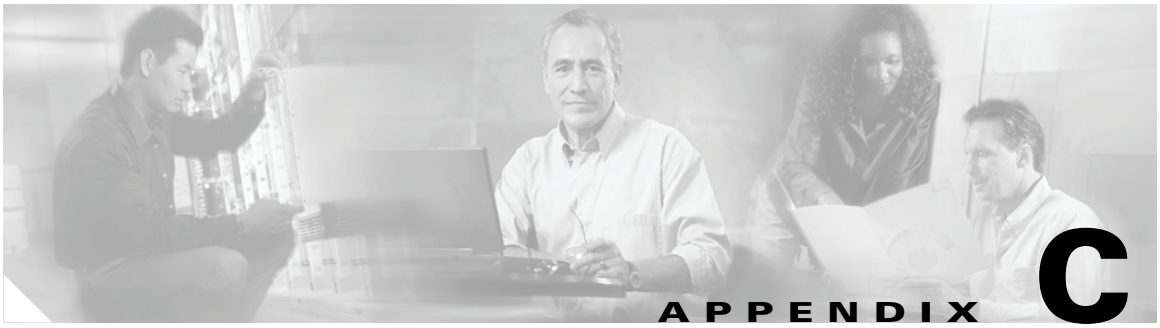
```
Console> (enable) set boot device cf:4 6
Device BOOT variable = cf:4
Memory-test set to PARTIAL
Warning:Device list is not verified but still set in the boot string.
Console> (enable) reset 6
This command will reset module 6.
Unsaved configuration on module 6 will be lost
Do you want to continue (y/n) [n]? y
Module 6 shut down in progress, please don't remove module until shutdown completed.
Console> (enable) Module 6 shutdown completed. Module resetting...
```

```
2003 Jan 17 08:36:58 %SYS-3-SUP_OSBOOTSTATUS:AP OS Boot Status:finished booting
2003 Jan 17 08:37:51 %SYS-5-MOD_OK:Module 6 is online
2003 Jan 17 08:37:51 %DTP-5-TRUNKPORTON:Port 6/1 has become dot1q trunk
Console> (enable) session 15
Trying Router-15...
Connected to Router-15.
Type ^C^C^C to switch back...
Router>
Router# copy tftp:pclc#6-fs:
Address or name of remote host []? 10.1.1.1
Source filename []? mp.1-2-0-16.bin.gz
Destination filename [mp.1-2-0-16.bin.gz]?
Accessing tftp://10.1.1.1/mp.1-2-0-16.bin.gz...
Loading mp.1-2-0-16.bin.gz from 10.1.1.1 (via Vlan2):
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

<output truncated>

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
[OK - 9818951 bytes]

9818951 bytes copied in 164.388 secs (59730 bytes/sec)
ssl-proxy>
1w0d:%SVCLC-SP-6-STRRECVD:mod 6:<MP upgrade started. Do not reset the card.>
1w0d:%SVCLC-SP-6-STRRECVD:mod 6:<Upgrade of MP was successful. You can now boot MP.>
Router# exit
Console> (enable) set boot device cf:1 6
Device BOOT variable = cf:1
Memory-test set to PARTIAL
Warning:Device list is not verified but still set in the boot string.
Console> (enable)
Console> (enable) reset 6 cf:1
This command will reset module 6.
Unsaved configuration on module 6 will be lost
Do you want to continue (y/n) [n]? y
Module 6 shut down in progress, please don't remove module until shutdown completed.
Console> (enable) Module 6 shutdown completed. Module resetting...
2003 Jan 17 08:34:07 %SYS-3-SUP_OSBOOTSTATUS:MP OS Boot Status:finished booting
2003 Jan 17 08:34:23 %SYS-5-MOD_OK:Module 6 is online
2003 Jan 17 08:34:23 %DTP-5-TRUNKPORTON:Port 6/1 has become dot1q trunk
```



Testing SSL Proxy Services

You can test or troubleshoot SSL proxy services by doing one of the following:

Generating a Self-Signed Certificate

You can generate multiple self-signed certificates for testing SSL proxy services, specifying the key label and the subject name, by entering the **test crypto pki self** command. You must generate a key pair with a label before you generate the self-signed certificate. See the "Generating RSA Key Pairs" section on page 3-5 for details on generating key pairs.

After you enter the **test crypto pki self** command, you are prompted for the key pair label and the subject name of the certificate. A trustpoint with the key pair label as the trustpoint name is automatically created, and the hexadecimal dump of the self-signed certificate displays on the console. You can then assign the trustpoint to a proxy service for testing. You can repeat the procedure after reboot if necessary. The certificate is stored in memory only and cannot be saved in NVRAM as part of the configuration.

Note You cannot save the self-signed certificates as part of the configuration.

Note You can assign a generated self-signed certificate to a proxy service, but you cannot assign an imported self-signed certificate to a proxy service because you cannot import the key pair of the certificate authority that signed the imported certificate.

Note The **show crypto ca certificate** command does not display the self-signed certificates.

To generate a self-signed certificate and assign a trustpoint to the proxy service, perform this task:

	Command	Purpose
Step 1	`ssl-proxy# test crypto pki self`	Generates a self-signed certificate. **Note** After you enter this command, you are prompted for the subject name (using the LDAP format) and key pair name.
Step 2	`ssl-proxy# show crypto ca trustpoint label`	Displays information for the trustpoint.
Step 3	`ssl-proxy# configure terminal`	Enters configuration mode, selecting the terminal option.
Step 4	`ssl-proxy(config)# ssl-proxy context name`	Enters the SSL context subcommand mode. Use the optional SSL context name to specify an SSL virtualization instance. The context name is case-sensitive.
Step 5	`ssl-proxy(config-context)# service service_name`	Defines the name of the SSL proxy service. **Note** The *service-name* is case-sensitive.
Step 6	`ssl-proxy(config-ctx-ssl-proxy)# certificate rsa general-purpose trustpoint trustpoint_label`	Assigns a trustpoint to a proxy service.
Step 7	`ssl-proxy(config-ctx-ssl-proxy)# end`	Exits configuration mode.
Step 8	`ssl-proxy# show ssl-proxy service service-name`	Displays the key pair and the serial number of the certificate chain used for a specified proxy service.

Note If the trustpoint already exists, it might be replaced by the test certificate. We recommend that you generate a unique key pair for the test certificate.

This example shows how to generate a key pair, generate a self-signed certificate, and assign the certificate to a proxy service:

```
ssl-proxy# configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
ssl-proxy(config)# crypto key generate rsa general-keys label k1 modulus 1024
The name for the keys will be:k1

% The key modulus size is 1024 bits
% Generating 1024 bit RSA keys ...[OK]

ssl-proxy(config)# end
ssl-proxy#
*Mar 20 14:34:01.543:%SYS-5-CONFIG_I:Configured from console by console
ssl-proxy#
ssl-proxy# test crypto pki self
Enter subject name for certificate
CN=testhost.my.com, O=lab, OU=testgroup
Enter name of key to be used
k1
ssl-proxy#
     30 82 02 06 30 82 01 6F 02 20 45 32 30 38 39 32
     45 37 38 31 42 41 46 45 45 45 44 45 37 37 41 36
     43 41 44 37 44 43 45 38 34 37 30 0D 06 09 2A 86
     48 86 F7 0D 01 01 04 05 00 30 3C 31 12 30 10 06
     03 55 04 0B 13 09 74 65 73 74 67 72 6F 75 70 31
```

```
      0C 30 0A 06 03 55 04 0A 13 03 6C 61 62 31 18 30
      16 06 03 55 04 03 13 0F 74 65 73 74 68 6F 73 74
      2E 6D 79 2E 63 6F 6D 30 1E 17 0D 30 33 30 33 32
      30 31 34 33 35 30 30 5A 17 0D 31 33 30 33 31 37
      31 34 33 35 30 30 5A 30 3C 31 12 30 10 06 03 55
      04 0B 13 09 74 65 73 74 67 72 6F 75 70 31 0C 30
      0A 06 03 55 04 0A 13 03 6C 61 62 31 18 30 16 06
      03 55 04 03 13 0F 74 65 73 74 68 6F 73 74 2E 6D
      79 2E 63 6F 6D 30 81 9F 30 0D 06 09 2A 86 48 86
      F7 0D 01 01 01 05 00 03 81 8D 00 30 81 89 02 81
      81 00 EC 21 35 B5 0E BF 9C 1C 71 05 05 B2 8A 47
      C8 F9 13 6C 5A 14 77 63 BD 0C B7 D3 35 6A DB B8
      0F C2 D2 39 A8 62 67 EE CB BC 8D 5E F8 C2 1E 8E
      D6 39 62 07 B4 64 20 D8 29 25 1E 9E 06 C8 F8 F9
      A6 29 05 19 CC D9 00 E9 2D 96 6D CE CA E0 D7 BF
      DC 9D 1B 7E 71 C1 D7 3F 25 28 41 5A F9 FB 98 66
      B9 A7 81 18 79 71 2A AC 55 F8 CC A4 4A 90 35 A7
      E9 BD 79 66 BC 5B C5 98 16 B0 63 5B D3 6E 85 65
      42 1B 02 03 01 00 01 30 0D 06 09 2A 86 48 86 F7
      0D 01 01 04 05 00 03 81 81 00 A3 93 7A E6 60 54
      8C 3A FF 6A 72 A8 1F 4B AD 79 53 C4 37 DF C4 D4
      F9 F4 58 3C E4 D8 BE FF BB C5 F9 CD B0 20 7F 3D
      0E B5 11 8E FA 33 02 9E 5E 52 36 4D 0F AB 21 41
      97 A4 2D 94 4D DF D2 A0 B4 DE B0 2E 1C BA 16 A9
      4C 28 34 72 8E D5 82 F6 B6 B2 D6 4E B5 1A F0 BB
      6B 65 E7 85 52 72 9F 9C BC A7 D9 B4 79 AB 6B C2
      DC FD AD 02 D3 28 87 CD 06 8B 11 3C 22 85 28 1B
      DC 04 05 8D 4F 1D 07 8D D0 BC

ssl-proxy# show crypto ca trustpoint k1
Trustpoint k1:
    Subject Name:
    CN = testhost.my.com
     O = lab
     OU = testgroup
          Serial Number:453230383932453738314241464545454445373741364341443744434538343 7
    Application generated trust point

ssl-proxy# configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
ssl-proxy(config)# ssl-proxy context Default
ssl-proxy(config-context)# service ser1
ssl-proxy(config-ctx-ssl-proxy)# certificate rsa general trustpoint k1
ssl-proxy(config-ctx-ssl-proxy)#
*Mar 20 14:36:09.567:%STE-6-PKI_SERVER_CERT_INSTALL:Proxy:ser1, Trustpoint:k1, Key:k1,
Serial#:453230383932453738314241464545454445373741364341443744434538343 7, Index:3
ssl-proxy(config-ctx-ssl-proxy)# end
ssl-proxy#
*Mar 20 14:36:16.363:%SYS-5-CONFIG_I:Configured from console by console
ssl-proxy#
ssl-proxy# show ssl-proxy service ser1
Service id:2, bound_service_id:258
Virtual IP address not configured
Server IP address not configured
rsa-general-purpose certificate trustpoint:k1
  Certificate chain for new connections:
    Server Certificate:
       Key Label:k1
       Serial Number:453230383932453738314241464545454445373741364341443744434538343 7
       Self-signed
Certificate chain complete
Admin Status:down
Operation Status:down
ssl-proxy#
```

Importing the Embedded Test Certificate

A test PKCS12 file (test/testssl.p12) is embedded in the SSL software on the module. You can install the file into NVRAM for testing purposes and for proof of concept. After you install the PKCS12 file, you can import it to a trustpoint and then assign it to a proxy service that is configured for testing.

To install and import the test file, perform this task:

	Command	Purpose
Step 1	`ssl-proxy# test ssl-proxy certificate install`	Installs the test PKCS12 file to NVRAM.
Step 2	`ssl-proxy(config)# crypto ca import` *trustpoint_label* `pkcs12 nvram:test/testssl.p12` *passphrase*	Imports the test PKCS12 file to the module. **Note** For the test certificate, the *passphrase* is **cisco**.
Step 3	`ssl-proxy(config)# ssl-proxy context` *name*	Enters the SSL context subcommand mode. Use the optional SSL context name to specify an SSL virtualization instance. The context name is case-sensitive.
Step 4	`ssl-proxy(config-context)# service` *test_name*	Defines the name of the test proxy service.
Step 5	`ssl-proxy(config-ctx-ssl-proxy)# certificate rsa general-purpose trustpoint` *trustpoint_label*	Assigns a trustpoint to a proxy service.
Step 6	`ssl-proxy(config-ctx-ssl-proxy)# end`	Exits configuration mode.
Step 7	`ssl-proxy# show ssl-proxy stats` *test_service*	Displays test statistics information.

This example shows how to import the test PKCS12 file:

```
ssl-proxy# test ssl-proxy certificate install
% Opening file, please wait ...
% Writing, please wait .............
% Please use the following config command to import the file.
  "crypto ca import <trustpoint-name> pkcs12 nvram:test/testssl.p12 cisco"
% Then you can assign the trustpoint to a proxy service for testing.
*Nov 18 22:59:06.331:%STE-6-PKI_TEST_CERT_INSTALL:Test key and certificate was installed
into NVRAM in a PKCS#12 file.

ssl-proxy# configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
ssl-proxy(config)# crypto ca import test-tp pkcs12 nvram:test/testssl.p12 cisco
Source filename [test/testssl.p12]?
CRYPTO_PKI:Imported PKCS12 file successfully.
ssl-proxy(config)#
*Nov 18 23:05:48.699:%CRYPTO-6-PKCS12IMPORT_SUCCESS:PKCS #12 Successfully Imported.
ssl-proxy(config)# ssl-proxy context Default
ssl-proxy(config-context)# service test-service
ssl-proxy(config-ctx-ssl-proxy)# certificate rsa general-purpose trustpoint test-tp
ssl-proxy(config-ctx-ssl-proxy)# end
ssl-proxy#
```

A

B

C

D

E

G

H

V