Catalyst 6500 Series Switch Content Switching Module with SSL Installation and Configuration Note

Software Release 2.1(x)
May 2005

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-7030-01

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

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

The following information is for FCC compliance of Class A devices: This equipment has been tested and found to comply with the limits for a Class A digital device, pursuant to part 15 of the FCC rules. These limits are designed to provide reasonable protection against harmful interference when the equipment is operated in a commercial environment. This equipment generates, uses, and can radiate radio-frequency energy and, if not installed and used in accordance with the instruction manual, may cause harmful interference to radio communications. Operation of this equipment in a residential area is likely to cause harmful interference, in which case users will be required to correct the interference at their own expense.

The following information is for FCC compliance of Class B devices: The equipment described in this manual generates and may radiate radio-frequency energy. If it is not installed in accordance with Cisco's installation instructions, it may cause interference with radio and television reception. This equipment has been tested and found to comply with the limits for a Class B digital device in accordance with the specifications in part 15 of the FCC rules. These specifications are designed to provide reasonable protection against such interference in a residential installation. However, there is no guarantee that interference will not occur in a particular installation.

Modifying the equipment without Cisco's written authorization may result in the equipment no longer complying with FCC requirements for Class A or Class B digital devices. In that event, your right to use the equipment may be limited by FCC regulations, and you may be required to correct any interference to radio or television communications at your own expense.

You can determine whether your equipment is causing interference by turning it off. If the interference stops, it was probably caused by the Cisco equipment or one of its peripheral devices. If the equipment causes interference to radio or television reception, try to correct the interference by using one or more of the following measures:

• Turn the television or radio antenna until the interference stops.

• Move the equipment to one side or the other of the television or radio.

• Move the equipment farther away from the television or radio.

• Plug the equipment into an outlet that is on a different circuit from the television or radio. (That is, make certain the equipment and the television or radio are on circuits controlled by different circuit breakers or fuses.)

Modifications to this product not authorized by Cisco Systems, Inc. could void the FCC approval and negate your authority to operate the product.

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

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

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

Catalyst 6500 Series Content Switching Module Secure Installation and Configuration Note

Preface

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

Note Except where specifically differentiated, the term *Catalyst 6500 series switche*s includes both Catalyst 6500 series and Catalyst 6000 series switches.

Note The term *SSL daughter card* is a Secure Socket Layer (SSL) termination daughter card for the CSM-S that accelerates SSL transactions.

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*.

Note For translations of the warnings in this publication, see the "Safety Overview" section on page xvi.

Audience

Only trained and qualified service personnel (as defined in IEC 60950 and AS/NZS3260) should install, replace, or service the equipment described in this publication.

Organization

This publication is organized as follows:

Chapter	Title	Description
Chapter 1	Product Overview	Presents an overview of the Catalyst 6500 Series Content Switching Module with SSL (CSM-S).
Chapter 2	Networking with the Content Switching Module with SSL	Describes how the supported hardware and software for the CSM-S operates on a network.
Chapter 3	Getting Started	Provides quick start guide to content switching on the supported hardware and software for the CSM-S.
Chapter 4	Configuring VLANs	Describes how to set up client and server VLANs for the CSM-S.
Chapter 5	Configuring Real Servers and Server Farms	Describes how to configure load balancing on the CSM-S.
Chapter 6	Configuring Virtual Servers, Maps, and Policies	Describes how to configure health monitoring on the CSM-S.
Chapter 7	Configuring the CSM-S SSL Services	Describes how to set up the SSL services for the CSM-S.
Chapter 8	Configuring SSL Services Secure Transactions	Describes how to configure services including keys on the CSM-S.
Chapter 9	Configuring Redundancy	Describes how to configure fault tolerance, HSRP, connection redundancy, and hitless upgrades.
Chapter 10	Configuring Additional Features and Options	Describes how to configure sticky groups and route health injection (RHI), Global Server Load Balancing (GSLB), and network management.
Chapter 11	Configuring Health Monitoring	Describes how to configure and monitor the health of servers and server farms.
Chapter 12	Using TCL Scripts with the CSM-S	Describes how to use Toolkit Command Language (TCL) scripts to configure the CSM-S.
Chapter 13	Configuring Firewall Load Balancing	Describes firewalls in a load-balancing configuration with the CSM-S.
Appendix A	CSM-S Configuration Examples	Lists sample CSM-S configurations.
Appendix B	SSL Configuration Examples	Lists SSL configurations for the CSM-S.
Appendix C	Troubleshooting and System Messages	Provides troubleshooting information and lists system messages.
Appendix D	CSM XML Document Type Definition	Lists CSM-S error messages with explanations about why they occurred and actions required to correct the problem.

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 it 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.

Safety Overview

Safety warnings appear throughout this publication in procedures that, if performed incorrectly, may harm you. A warning symbol precedes each warning statement.

IMPORTANT SAFETY INSTRUCTIONS

Warning **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

Waarschuwing **BELANGRIJKE VEILIGHEIDSINSTRUCTIES**

Dit waarschuwingssymbool betekent gevaar. U verkeert in een situatie die lichamelijk letsel kan veroorzaken. Voordat u aan enige apparatuur gaat werken, dient u zich bewust te zijn van de bij elektrische schakelingen betrokken risico's en dient u op de hoogte te zijn van de standaard praktijken om ongelukken te voorkomen. Gebruik het nummer van de verklaring onderaan de waarschuwing als u een vertaling van de waarschuwing die bij het apparaat wordt geleverd, wilt raadplegen.

BEWAAR DEZE INSTRUCTIES

Varoitus **TÄRKEITÄ TURVALLISUUSOHJEITA**

Tämä varoitusmerkki merkitsee vaaraa. Tilanne voi aiheuttaa ruumiillisia vammoja. Ennen kuin käsittelet laitteistoa, huomioi sähköpiirien käsittelemiseen liittyvät riskit ja tutustu onnettomuuksien yleisiin ehkäisytapoihin. Turvallisuusvaroitusten käännökset löytyvät laitteen mukana toimitettujen käännettyjen turvallisuusvaroitusten joukosta varoitusten lopussa näkyvien lausuntonumeroiden avulla.

SÄILYTÄ NÄMÄ OHJEET

Attention **IMPORTANTES INFORMATIONS DE SÉCURITÉ**

Ce symbole d'avertissement indique un danger. Vous vous trouvez dans une situation pouvant entraîner des blessures ou des dommages corporels. Avant de travailler sur un équipement, soyez conscient des dangers liés aux circuits électriques et familiarisez-vous avec les procédures couramment utilisées pour éviter les accidents. Pour prendre connaissance des traductions des avertissements figurant dans les consignes de sécurité traduites qui accompagnent cet appareil, référez-vous au numéro de l'instruction situé à la fin de chaque avertissement.

CONSERVEZ CES INFORMATIONS

Warnung **WICHTIGE SICHERHEITSHINWEISE**

Dieses Warnsymbol bedeutet Gefahr. Sie befinden sich in einer Situation, die zu Verletzungen führen kann. Machen Sie sich vor der Arbeit mit Geräten mit den Gefahren elektrischer Schaltungen und den üblichen Verfahren zur Vorbeugung vor Unfällen vertraut. Suchen Sie mit der am Ende jeder Warnung angegebenen Anweisungsnummer nach der jeweiligen Übersetzung in den übersetzten Sicherheitshinweisen, die zusammen mit diesem Gerät ausgeliefert wurden.

BEWAHREN SIE DIESE HINWEISE GUT AUF.

Avvertenza **IMPORTANTI ISTRUZIONI SULLA SICUREZZA**

Questo simbolo di avvertenza indica un pericolo. La situazione potrebbe causare infortuni alle persone. Prima di intervenire su qualsiasi apparecchiatura, occorre essere al corrente dei pericoli relativi ai circuiti elettrici e conoscere le procedure standard per la prevenzione di incidenti. Utilizzare il numero di istruzione presente alla fine di ciascuna avvertenza per individuare le traduzioni delle avvertenze riportate in questo documento.

CONSERVARE QUESTE ISTRUZIONI

Advarsel **VIKTIGE SIKKERHETSINSTRUKSJONER**

Dette advarselssymbolet betyr fare. Du er i en situasjon som kan føre til skade på person. Før du begynner å arbeide med noe av utstyret, må du være oppmerksom på farene forbundet med elektriske kretser, og kjenne til standardprosedyrer for å forhindre ulykker. Bruk nummeret i slutten av hver advarsel for å finne oversettelsen i de oversatte sikkerhetsadvarslene som fulgte med denne enheten.

TA VARE PÅ DISSE INSTRUKSJONENE

Aviso **INSTRUÇÕES IMPORTANTES DE SEGURANÇA**

Este símbolo de aviso significa perigo. Você está em uma situação que poderá ser causadora de lesões corporais. Antes de iniciar a utilização de qualquer equipamento, tenha conhecimento dos perigos envolvidos no manuseio de circuitos elétricos e familiarize-se com as práticas habituais de prevenção de acidentes. Utilize o número da instrução fornecido ao final de cada aviso para localizar sua tradução nos avisos de segurança traduzidos que acompanham este dispositivo.

GUARDE ESTAS INSTRUÇÕES

¡Advertencia! **INSTRUCCIONES IMPORTANTES DE SEGURIDAD**

Este símbolo de aviso indica peligro. Existe riesgo para su integridad física. Antes de manipular cualquier equipo, considere los riesgos de la corriente eléctrica y familiarícese con los procedimientos estándar de prevención de accidentes. Al final de cada advertencia encontrará el número que le ayudará a encontrar el texto traducido en el apartado de traducciones que acompaña a este dispositivo.

GUARDE ESTAS INSTRUCCIONES

Varning! **VIKTIGA SÄKERHETSANVISNINGAR**

Denna varningssignal signalerar fara. Du befinner dig i en situation som kan leda till personskada. Innan du utför arbete på någon utrustning måste du vara medveten om farorna med elkretsar och känna till vanliga förfaranden för att förebygga olyckor. Använd det nummer som finns i slutet av varje varning för att hitta dess översättning i de översatta säkerhetsvarningar som medföljer denna anordning.

SPARA DESSA ANVISNINGAR

Figyelem **FONTOS BIZTONSÁGI ELOÍRÁSOK**

Ez a figyelmezeto jel veszélyre utal. Sérülésveszélyt rejto helyzetben van. Mielott bármely berendezésen munkát végezte, legyen figyelemmel az elektromos áramkörök okozta kockázatokra, és ismerkedjen meg a szokásos balesetvédelmi eljárásokkal. A kiadványban szereplo figyelmeztetések fordítása a készülékhez mellékelt biztonsági figyelmeztetések között található; a fordítás az egyes figyelmeztetések végén látható szám alapján keresheto meg.

ORIZZE MEG EZEKET AZ UTASÍTÁSOKAT!

Предупреждение **ВАЖНЫЕ ИНСТРУКЦИИ ПО СОБЛЮДЕНИЮ ТЕХНИКИ БЕЗОПАСНОСТИ**

Этот символ предупреждения обозначает опасность. То есть имеет место ситуация, в которой следует опасаться телесных повреждений. Перед эксплуатацией оборудования выясните, каким опасностям может подвергаться пользователь при использовании электрических цепей, и ознакомьтесь с правилами техники безопасности для предотвращения возможных несчастных случаев. Воспользуйтесь номером заявления, приведенным в конце каждого предупреждения, чтобы найти его переведенный вариант в переводе предупреждений по безопасности, прилагаемом к данному устройству.

СОХРАНИТЕ ЭТИ ИНСТРУКЦИИ

警告 重要的安全性说明

此警告符号代表危险。您正处于可能受到严重伤害的工作环境中。在您使用设备开始工作之前，必须充分意识到触电的危险，并熟练掌握防止事故发生的标准工作程序。请根据每项警告结尾提供的声明号码来找到此设备的安全性警告说明的翻译文本。

请保存这些安全性说明

警告 安全上の重要な注意事項

「危険」の意味です。人身事故を予防するための注意事項が記述されています。装置の取り扱い作業を行うときは、電気回路の危険性に注意し、一般的な事故防止策に留意してください。警告の各国語版は、各注意事項の番号を基に、装置に付属の「Translated Safety Warnings」を参照してください。

これらの注意事項を保管しておいてください。

주의 중요 안전 지침

이 경고 기호는 위험을 나타냅니다. 작업자가 신체 부상을 일으킬 수 있는 위험한 환경에 있습니다. 장비에 작업을 수행하기 전에 전기 회로와 관련된 위험을 숙지하고 표준 작업 관례를 숙지하여 사고를 방지하십시오. 각 경고의 마지막 부분에 있는 경고문 번호를 참조하여 이 장치와 함께 제공되는 번역된 안전 경고문에서 해당 번역문을 찾으십시오.

이 지시 사항을 보관하십시오.

Aviso **INSTRUÇÕES IMPORTANTES DE SEGURANÇA**

Este símbolo de aviso significa perigo. Você se encontra em uma situação em que há risco de lesões corporais. Antes de trabalhar com qualquer equipamento, esteja ciente dos riscos que envolvem os circuitos elétricos e familiarize-se com as práticas padrão de prevenção de acidentes. Use o número da declaração fornecido ao final de cada aviso para localizar sua tradução nos avisos de segurança traduzidos que acompanham o dispositivo.

GUARDE ESTAS INSTRUÇÕES

Advarsel **VIGTIGE SIKKERHEDSANVISNINGER**

Dette advarselssymbol betyder fare. Du befinder dig i en situation med risiko for legemesbeskadigelse. Før du begynder arbejde på udstyr, skal du være opmærksom på de involverede risici, der er ved elektriske kredsløb, og du skal sætte dig ind i standardprocedurer til undgåelse af ulykker. Brug erklæringsnummeret efter hver advarsel for at finde oversættelsen i de oversatte advarsler, der fulgte med denne enhed.

GEM DISSE ANVISNINGER

تحذير **إرشادات الأمان الهامة**

يوضح رمز التحذير هذا وجود خطر. وهذا يعني أنك متواجد في مكان قد ينتج عنه التعرض لإصابات. قبل بدء العمل، احذر مخاطر التعرض للصدمات الكهربائية وكن على علم بالإجراءات القياسية للحيلولة دون وقوع أي حوادث. استخدم رقم البيان الموجود في أخر كل تحذير لتحديد مكان ترجمته داخل تحذيرات الأمان المترجمة التي تأتي مع الجهاز.
قم بحفظ هذه الإرشادات

Upozorenje **VAŽNE SIGURNOSNE NAPOMENE**

Ovaj simbol upozorenja predstavlja opasnost. Nalazite se u situaciji koja može prouzročiti tjelesne ozljede. Prije rada s bilo kojim uređajem, morate razumjeti opasnosti vezane uz električne sklopove, te biti upoznati sa standardnim načinima izbjegavanja nesreća. U prevedenim sigurnosnim upozorenjima, priloženima uz uređaj, možete prema broju koji se nalazi uz pojedino upozorenje pronaći i njegov prijevod.

SAČUVAJTE OVE UPUTE

Upozornění

DŮLEŽITÉ BEZPEČNOSTNÍ POKYNY

Tento upozorňující symbol označuje nebezpečí. Jste v situaci, která by mohla způsobit nebezpečí úrazu. Před prací na jakémkoliv vybavení si uvědomte nebezpečí související s elektrickými obvody a seznamte se se standardními opatřeními pro předcházení úrazům. Podle čísla na konci každého upozornění vyhledejte jeho překlad v přeložených bezpečnostních upozorněních, která jsou přiložena k zařízení.

USCHOVEJTE TYTO POKYNY

Προειδοποίηση

ΣΗΜΑΝΤΙΚΕΣ ΟΔΗΓΙΕΣ ΑΣΦΑΛΕΙΑΣ

Αυτό το προειδοποιητικό σύμβολο σημαίνει κίνδυνο. Βρίσκεστε σε κατάσταση που μπορεί να προκαλέσει τραυματισμό. Πριν εργαστείτε σε οποιοδήποτε εξοπλισμό, να έχετε υπόψη σας τους κινδύνους που σχετίζονται με τα ηλεκτρικά κυκλώματα και να έχετε εξοικειωθεί με τις συνήθεις πρακτικές για την αποφυγή ατυχημάτων. Χρησιμοποιήστε τον αριθμό δήλωσης που παρέχεται στο τέλος κάθε προειδοποίησης, για να εντοπίσετε τη μετάφρασή της στις μεταφρασμένες προειδοποιήσεις ασφαλείας που συνοδεύουν τη συσκευή.

ΦΥΛΑΞΤΕ ΑΥΤΕΣ ΤΙΣ ΟΔΗΓΙΕΣ

אזהרה

הוראות בטיחות חשובות

סימן אזהרה זה מסמל סכנה. אתה נמצא במצב העלול לגרום לפציעה. לפני שתעבוד עם ציוד כלשהו, עליך להיות מודע לסכנות הכרוכות במעגלים חשמליים ולהכיר את הנהלים המקובלים למניעת תאונות. השתמש במספר ההוראה המסופק בסופה של כל אזהרה כד לאתר את התרגום באזהרות הבטיחות המתורגמות שמצורפות להתקן.

שמור הוראות אלה

Opomena

постои кај електричните кола и треба да ги познавате стандардните постапки за спречување на несреќни случаи. Искористете го бројот на изјавата што се наоѓа на крајот на секое предупредување за да го најдете неговиот период во преведените безбедносни предупредувања што се испорачани со уредот.
ЧУВАЈТЕ ГИ ОВИЕ НАПАТСТВИЈА

Ostrzeżenie **WAŻNE INSTRUKCJE DOTYCZĄCE BEZPIECZEŃSTWA**

Ten symbol ostrzeżenia oznacza niebezpieczeństwo. Zachodzi sytuacja, która może powodować obrażenia ciała. Przed przystąpieniem do prac przy urządzeniach należy zapoznać się z zagrożeniami związanymi z układami elektrycznymi oraz ze standardowymi środkami zapobiegania wypadkom. Na końcu każdego ostrzeżenia podano numer, na podstawie którego można odszukać tłumaczenie tego ostrzeżenia w dołączonym do urządzenia dokumencie z tłumaczeniami ostrzeżeń.

NINIEJSZE INSTRUKCJE NALEŻY ZACHOWAĆ

Upozornenie **DÔLEŽITÉ BEZPEČNOSTNÉ POKYNY**

Tento varovný symbol označuje nebezpečenstvo. Nachádzate sa v situácii s nebezpečenstvom úrazu. Pred prácou na akomkoľvek vybavení si uvedomte nebezpečenstvo súvisiace s elektrickými obvodmi a oboznámte sa so štandardnými opatreniami na predchádzanie úrazom. Podľa čísla na konci každého upozornenia vyhľadajte jeho preklad v preložených bezpečnostných upozorneniach, ktoré sú priložené k zariadeniu.

USCHOVAJTE SI TENTO NÁVOD

Related Documentation

For more detailed installation and configuration information for the Content Switching Module with SSL, refer to the following publications:

- *Release Notes for the Catalyst 6500 Series Switch Content Switching Module with SSL*
- *Catalyst 6500 Series Switch Content Switching Module with SSL Installation Note*
- *Catalyst 6500 Series Switch Content Switching Module with SSL Command Reference*
- *Regulatory Compliance and Safety Information for the Catalyst 6500 Series Switches*

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

- *Release Notes for Catalyst 6500 Series SSL Services Module Software Release 2.x*
- *Catalyst 6500 Series Switch SSL Services Module Installation and Verification Note*
- *Catalyst 6500 Series Switch SSL Services Module Command Reference*
- *Catalyst 6500 Series Switch SSL Services Module System Messages*

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

- *Catalyst 6500 Series Switch Installation Guide*
- *Catalyst 6500 Series Switch Quick Software Configuration 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*
- *ATM Software Configuration and Command Reference—Catalyst 5000 Family and Catalyst 6500 Series Switches*
- *System Message Guide—Catalyst 6500 Series Switches*
- For information about MIBs, refer to this URL:

  http://www.cisco.com/public/sw-center/netmgmt/cmtk/mibs.shtml
- *Release Notes for Catalyst 6500 Series Switches and Cisco 7600 Series Router for Cisco IOS Release 12.1(8a)E3*

Cisco IOS Configuration Guides and Command References—Use these publications to help you configure the Cisco IOS software that runs on the MSFC and on the MSM and ATM modules.

Obtaining Documentation

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

Cisco.com

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

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

You can access the Cisco website at this URL:

http://www.cisco.com

You can access international Cisco websites at this URL:

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

Documentation DVD

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

Registered Cisco.com users (Cisco direct customers) can order a Cisco Documentation DVD (product number DOC-DOCDVD=) from the Ordering tool or Cisco Marketplace.

Cisco Ordering tool:

http://www.cisco.com/en/US/partner/ordering/

Cisco Marketplace:

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

Ordering Documentation

You can find instructions for ordering documentation at this URL:

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

You can order Cisco documentation in these ways:

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

  http://www.cisco.com/en/US/partner/ordering/

- Nonregistered Cisco.com users can order documentation through a local account representative by calling Cisco Systems Corporate Headquarters (California, USA) at 408 526-7208 or, elsewhere in North America, by calling 1 800 553-NETS (6387).

Documentation Feedback

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

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

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

We appreciate your comments.

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
- Nonemergencies—psirt@cisco.com

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 that has the most recent creation date in this public key server list:

http://pgp.mit.edu:11371/pks/lookup?search=psirt%40cisco.com&op=index&exact=on

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

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

Obtaining Technical Assistance

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

Cisco Technical Support Website

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

http://www.cisco.com/techsupport

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

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

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 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 TAC engineer. The TAC Service Request Tool is located at this URL:

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

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

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

Asia-Pacific: +61 2 8446 7411 (Australia: 1 800 805 227)
EMEA: +32 2 704 55 55
USA: 1 800 553-2447

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

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

Definitions of Service Request Severity

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

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

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

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

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

Obtaining Additional Publications and Information

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

- Cisco Marketplace provides a variety of Cisco books, reference guides, and logo merchandise. Visit Cisco Marketplace, the company store, at this URL:

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

- *Cisco Press* publishes a wide range of general networking, training and certification titles. Both new and experienced users will benefit from these publications. For current Cisco Press titles and other information, go to Cisco Press at this URL:

  http://www.ciscopress.com

- *Packet* magazine is the Cisco Systems technical user magazine for maximizing Internet and networking investments. Each quarter, Packet delivers coverage of the latest industry trends, technology breakthroughs, and Cisco products and solutions, as well as network deployment and troubleshooting tips, configuration examples, customer case studies, certification and training information, and links to scores of in-depth online resources. You can access Packet magazine at this URL:

  http://www.cisco.com/packet

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

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

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

  http://www.cisco.com/ipj

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

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

Licenses

This section contains information about software licenses.

Software License Agreement

THIS AGREEMENT IS AVAILABLE IN LANGUAGES OTHER THAN ENGLISH; PLEASE SEE YOUR CISCO SYSTEMS, INC. (“CISCO”) RESELLER OR VISIT OUR WEBSITE AT WWW.CISCO.COM. PLEASE READ THIS SOFTWARE LICENSE AGREEMENT CAREFULLY BEFORE DOWNLOADING, INSTALLING OR USING CISCO OR CISCO-SUPPLIED SOFTWARE. BY DOWNLOADING OR INSTALLING THE SOFTWARE, OR USING THE EQUIPMENT THAT CONTAINS THIS SOFTWARE, YOU ARE CONSENTING TO BE BOUND BY THIS AGREEMENT. IF YOU DO NOT AGREE TO ALL OF THE TERMS OF THIS AGREEMENT, THEN (A) DO NOT DOWNLOAD, INSTALL OR USE THE SOFTWARE, AND (B) YOU MAY RETURN THE SOFTWARE FOR A FULL REFUND, OR, IF THE SOFTWARE IS SUPPLIED AS PART OF ANOTHER PRODUCT, YOU MAY RETURN THE ENTIRE PRODUCT FOR A FULL REFUND. YOUR RIGHT TO RETURN AND REFUND EXPIRES 30 DAYS AFTER PURCHASE FROM CISCO OR AN AUTHORIZED CISCO RESELLER, AND APPLIES ONLY IF YOU ARE THE ORIGINAL PURCHASER.
The following terms govern your use of the Software except to the extent a particular program (a) is the subject of a separate written agreement with Cisco or (b) includes a separate “click-on” license agreement as part of the installation process.

License. Subject to the terms and conditions of and except as otherwise provided in this Agreement, Cisco Systems, Inc. ("Cisco") and its suppliers grant to Customer ("Customer") a nonexclusive and nontransferable license to use the specific Cisco program modules, feature set(s) or feature(s) for which Customer has paid the required license fees (the "Software"), in object code form only. In addition, the foregoing license shall also be subject to each of the following limitations:

- Unless otherwise expressly provided in the documentation, Customer shall use the Software solely as embedded in, for execution on, or (where the applicable documentation permits installation on non-Cisco equipment) for communication with Cisco equipment owned or leased by Customer;
- Customer's use of the Software shall be limited to use on a single hardware chassis, on a single central processing unit, as applicable, or use on such greater number of chassis or central processing units as Customer may have paid Cisco the required license fee; and
- Customer's use of the Software shall also be limited as applicable to the number of issued and outstanding IP addresses, central processing unit performance, number of ports, and any other restrictions set forth in Cisco's product catalog for the Software.

NOTE: For evaluation or beta copies for which Cisco does not charge a license fee, the above requirement to pay a license fee does not apply.

General Limitations. Except as otherwise expressly provided under this Agreement, Customer shall have no right, and Customer specifically agrees not to: (i) transfer, assign or sublicense its license rights to any other person, or use the Software on unauthorized or secondhand Cisco equipment, and any such attempted transfer, assignment or sublicense shall be void; (ii) make error corrections to or otherwise modify or adapt the Software or create derivative works based upon the Software, or to permit third parties to do the same; or (iii) decompile, decrypt, reverse engineer, disassemble or otherwise reduce the Software to human-readable form to gain access to trade secrets or confidential information in the Software. To the extent required by law, at Customer's request, Cisco shall provide Customer with the interface information needed to achieve interoperability between the Software and another independently created program, on payment of Cisco's applicable fee. Customer shall observe strict obligations of confidentiality with respect to such information.

Upgrades and Additional Copies. For purposes of this Agreement, "Software" shall include (and the terms and conditions of this Agreement shall apply to) any upgrades, updates, bug fixes or modified versions (collectively, "Upgrades") or backup copies of the Software licensed or provided to Customer by Cisco or an authorized distributor for which Customer has paid the applicable license fees. NOTWITHSTANDING ANY OTHER PROVISION OF THIS AGREEMENT: (1) CUSTOMER HAS NO LICENSE OR RIGHT TO USE ANY SUCH ADDITIONAL COPIES OR UPGRADES UNLESS CUSTOMER, AT THE TIME OF ACQUIRING SUCH COPY OR UPGRADE, ALREADY HOLDS A VALID LICENSE TO THE ORIGINAL SOFTWARE AND HAS PAID THE APPLICABLE FEE FOR THE UPGRADE; (2) USE OF UPGRADES IS LIMITED TO CISCO EQUIPMENT FOR WHICH CUSTOMER IS THE ORIGINAL END USER PURCHASER OR LESSEE OR WHO OTHERWISE HOLDS A VALID LICENSE TO USE THE SOFTWARE WHICH IS BEING UPGRADED; AND (3) USE OF ADDITIONAL COPIES IS LIMITED TO BACKUP PURPOSES ONLY.

Proprietary Notices. Customer agrees to maintain and reproduce all copyright and other proprietary notices on all copies, in any form, of the Software in the same form and manner that such copyright and other proprietary notices are included on the Software. Except as expressly authorized in this Agreement, Customer shall not make any copies or duplicates or any Software without the prior written permission of Cisco. Customer may make such backup copies of the Software as may be necessary for Customer's lawful use, provided Customer affixes to such copies all copyright, confidentiality, and proprietary notices that appear on the original.

Protection of Information. Customer agrees that aspects of the Software and associated documentation, including the specific design and structure of individual programs, constitute trade secrets and/or copyrighted material of Cisco. Customer shall not disclose, provide, or otherwise make available such trade secrets or copyrighted material in any form to any third party without the prior written consent of Cisco. Customer shall implement reasonable security measures to protect such trade secrets and copyrighted material. Title to Software and documentation shall remain solely with Cisco.

Limited Warranty. If Customer obtained the Software directly from Cisco, then Cisco warrants that during the Warranty Period (as defined below): (i) the media on which the Software is furnished will be free of defects in materials and workmanship under normal use; and (ii) the Software will substantially conform to its published specifications. The "Warranty Period means a period beginning on the date of Customer's receipt of the Software and ending on the later of (a) ninety (90) days from the date of initial shipment of the Software by Cisco, or (b) the end of the minimum period required by the law of the applicable jurisdiction. In addition, Cisco may provide an additional limited Year 2000 warranty for the Software; information regarding this warranty and its applicability to the Software may be found at the web site address www.cisco.com/warp/public/779/smbiz/service/y2k/y2k_comp.htm. The limited warranties extend only to Customer as the original licensee. Customer's sole and exclusive remedy and the entire liability of Cisco and its suppliers under these limited warranties will be, at Cisco or its service center's option, repair, replacement, or refund of the Software if reported (or, upon request, returned) to Cisco or its designee. Except as expressly granted in this Agreement, the Software is provided AS IS. Cisco does not warrant that the Software is error free or that Customer will be able to operate the Software without problems or interruptions. In addition, due to the continual development of new techniques for intruding upon and attacking networks, Cisco does not warrant that the Software or any equipment, system or network on which the Software is used will be free of vulnerability to intrusion or attack. This warranty does not apply if the Software (a) is licensed for beta, evaluation, testing or demonstration purposes for which Cisco does not receive a license fee, (b) has been altered, except by Cisco, (c) has not been installed, operated, repaired, or maintained in accordance with instructions supplied by Cisco, (d) has been subjected to abnormal physical or electrical stress, misuse, negligence, or accident, or (e) is used in ultrahazardous activities. If Customer obtained the Software from a Cisco reseller, the terms of any warranty shall be as provided by such distributor, and Cisco provides Customer no warranty with respect to such Software.

Disclaimer of Warranties. EXCEPT AS SPECIFIED IN THIS WARRANTY, ALL EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS, AND WARRANTIES INCLUDING, WITHOUT LIMITATION, ANY IMPLIED WARRANTY OR CONDITION OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, NONINFRINGEMENT, SATISFACTORY QUALITY OR ARISING FROM A COURSE OF DEALING, USAGE, OR TRADE PRACTICE, ARE HEREBY EXCLUDED TO THE EXTENT ALLOWED BY APPLICABLE LAW. TO THE EXTENT AN IMPLIED WARRANTY CANNOT BE EXCLUDED, SUCH WARRANTY IS LIMITED IN DURATION TO THE WARRANTY PERIOD. BECAUSE SOME STATES OR JURISDICTIONS DO NOT ALLOW LIMITATIONS ON HOW LONG AN IMPLIED WARRANTY LASTS, THE ABOVE LIMITATION MAY NOT APPLY TO YOU. THIS WARRANTY GIVES YOU SPECIFIC LEGAL RIGHTS, AND YOU MAY ALSO HAVE OTHER RIGHTS WHICH VARY FROM JURISDICTION TO JURISDICTION. Disclaimer of Liabilities. IN NO EVENT WILL CISCO OR ITS SUPPLIERS BE LIABLE FOR ANY LOST REVENUE, PROFIT, OR DATA, OR FOR SPECIAL, INDIRECT, CONSEQUENTIAL, INCIDENTAL, OR PUNITIVE DAMAGES HOWEVER CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY ARISING OUT OF THE USE OF OR INABILITY TO USE THE SOFTWARE EVEN IF CISCO OR ITS SUPPLIERS HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. In no event shall Cisco's or its suppliers' liability to Customer, whether in contract, tort (including negligence), or otherwise, exceed the price paid by Customer. The foregoing limitations shall apply even if the above-stated warranty fails of its essential purpose. BECAUSE SOME STATES OR JURISDICTIONS DO NOT ALLOW LIMITATION OR EXCLUSION OF CONSEQUENTIAL OR INCIDENTAL DAMAGES, THE ABOVE LIMITATION MAY NOT APPLY TO YOU.

Term and Termination. This Agreement is effective until terminated. Customer may terminate this Agreement at any time by destroying all copies of Software including any documentation. Customer's license rights under this Agreement will terminate immediately without notice from Cisco if Customer fails to comply with any provision of this Agreement. Upon termination, Customer must destroy all copies of Software in its possession or control.

Customer Records. Customer grants to Cisco and its independent accountants the right to examine Customer's books, records and accounts during Customer's normal business hours to verify compliance with this Agreement.In the event such audit discloses non-compliance with this Agreement, Customer shall promptly pay to Cisco the appropriate licensee fees.

Export. Software, including technical data, may be subject to U.S. export control laws, including the U.S. Export Administration Act and its associated regulations, and may be subject to export or import regulations in other countries. Customer agrees to comply strictly with all such regulations and acknowledges that it has the responsibility to obtain licenses to export, re-export, or import Software.

Restricted Rights. Cisco's commercial software and commercial computer software documentation is provided to United States Government agencies in accordance with the terms of this Agreement, and per subparagraph "(c)" of the "Commercial Computer Software - Restricted Rights" clause at FAR 52.227-19 (June 1987). For DOD agencies, the restrictions set forth in the "Technical Data-Commercial Items" clause at DFARS 252.227-7015 (Nov 1995) shall also apply. General. This Agreement shall be governed by and construed in accordance with the laws of the State of California, United States of America, as if performed wholly within the state and without giving effect to the principles of conflict of law. If any portion hereof is found to be void or unenforceable, the remaining provisions of this Agreement shall remain in full force and effect. Cisco hereby specifically disclaims the UN Convention on Contracts for the International Sale of Goods. Except as expressly provided herein, this Agreement constitutes the entire agreement between the parties with respect to the license of the Software and supercedes any conflicting or additional terms contained in the purchase order.

Product Overview

This documentation supports these modules:

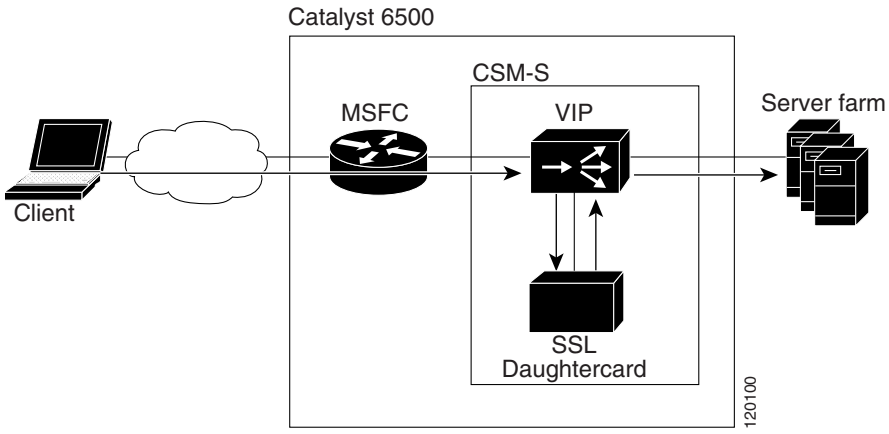
Product Number: WS-X6066-SLB-S-K9

The Catalyst 6500 Series Content Switching Module with SSL (CSM-S) combines high-performance server load balancing (SLB) with Secure Socket Layer (SSL) offload. The CSM-S can be used to distribute client requests using Layer 3 to Layer 7 information among groups of servers firewalls, caches, VPN termination devices, and other network devices. The CSM-S can also terminate and initiate SSL-encrypted traffic which allows the CSM-S to perform intelligent load balancing while ensuring secure end-to-end encryption.

Note The term *SSL daughter card* refers to the SSL termination daughter card for the CSM-S that accelerates SSL transactions.

Figure 1-1 shows an overview of how traffic flows through the CSM-S between the client and the server farm. Server farms are groups of load-balanced devices. Server farms that are represented as virtual servers can improve scalability and availability of services for your network. You can add new servers and remove failed or existing servers at any time without affecting the virtual server's availability.

Figure 1-1 CSM-S Traffic Flow Overview



Clients connect to the CSM-S directing their requests to the virtual IP (VIP) address of the virtual server. When a client initiates a connection to the virtual server, the CSM-S chooses a real server (a physical device that is assigned to a server farm) for the connection based on configured load-balancing algorithms and policies (access rules). Policies manage traffic by defining where to send client connections.

When a request arrives encrypted by SSL, the CSM-S can be configured to perform decryption, and eventually apply Layer 7 rules to the clear-text request to select the correct real server. Decryption only occurs if Layer 7 information is required to make the real server selection. If end-to-end encryption is required, the CSM-S re-encrypts the connection request after the real server selection has been made. This process allows the request to continue to the real server in its encrypted form.

Sticky connections limit traffic to individual servers by allowing multiple connections from the same client to *stick* (or attach) to the same real server using source IP addresses, source IP subnets, cookies, and the Secure Socket Layer (SSL) or by redirecting these connections using Hypertext Transfer Protocol (HTTP) redirect messages.

These sections describe the CSM-S:

Features

This software release contains feature sets supporting SSL (CSM-S) functionality from previous CSM releases. The tables in this section list these feature sets.

Table 1-1 lists the new CSM features in this release.

Table 1-1 New CSM Feature Set Description

Features New in this Release	Description
HTTP header sticky	Allows you to configure the CSM to perform stickiness based on the contents of the HTTP header (for example, the mobile station ISDN number [MSISDN], service key, session ID).
Configuration synchronization	Supports the synchronization of the configuration between the active and the standby CSM over the fault tolerant VLAN.
Failover tracking for interfaces and critical devices	Allows you to track the state of HSRP groups, physical interfaces, and gateways.
Private VLANs	Enables the use of private VLANs (PVLANs) with the CSM.

Table 1-1 New CSM Feature Set Description (continued)

Features New in this Release	Description
Partial server farm failover	When you configure a backup server farm, you can define threshold values so that the CSM fails over to the backup server farm if the primary server farm partially fails.
Server probe fail state improvements	Allows you to specify the number of successful retries needed to put a failed server back into service.
Real name option	Allows you to specify details about an entity. This option is applicable for probe, vserver, VLAN, and serverfarm modes
NAT configuration enhancements	Provides source NAT (NAT client) configuration rules to the policy level.
Infinite idle timeout	Allows you to keep a connection open for an indefinite time period.
VIP dependencies	Provides the ability to link VIPs together, providing the ability to automatically take a dependant VIP out of service if the specified VIP goes out of service.
Ordering of policies	Provides the ability to assign a priority value to a particular policy.
Maximum parse length reached behavior change	CSM load balances maximum parse length connection requests to the default policies.
Slow start improvements	Allows real servers to be in slow-start mode until the slow-start timer value expires or the conn_count is equal to that of the other real servers.
Non-secure router mode	Extends the environment variable to route a SYN packet, in addition to a non-SYN packet, that does not hit a VIP.
Increase vserver limit	Increases the number of virtual servers configurable with a particular VIP from 128 to 1000.

Table 1-2 lists the CSM features available in previous releases.

Table 1-2 CSM Feature Set Description

Features
Supported Hardware
Supervisor 2 with MSFC2
Supported Protocols
TCP load balancing
UDP generic IP protocol load balancing
Special application-layer support for FTP and the Real Time Streaming Protocol (RTSP)
Server Application State Protocol (SASP)
Layer 7 Functionality
Full regular expression matching

Table 1-2 CSM Feature Set Description (continued)

Features
URL, cookie switching, generic HTTP header parsing, HTTP method parsing
Miscellaneous Functionality
VIP connection watermarks
Backup (sorry server) and server farm
Optional port for health probes
IP reassembly
TCL (Toolkit Command Language) scripting
XML configuration interface
SNMP
GSLB (Global Server Load Balancing)—Requires a license
Resource usage display
Configurable idle and pending connection timeout
Idle timeout for unidirectional flows
SSL Services Module (SSLM) integration for SSL load balancing
Real server names
TCP connection redundancy for all types of flows (TCP, UDP, and IP)
Fault-tolerant **show** command enhancements
IOS SLB FWLB interoperation (IP reverse-sticky)
Multiple CSMs in a chassis
CSM and IOS-SLB functioning simultaneously in a chassis
Configurable HTTP 1.1 persistence (either all GETs are made to the same server or are balanced to multiple servers)
Fully configurable NAT
Server-initiated connections
Route health injection
Load-Balancing Algorithms
Round-robin
Weighted round-robin (WRR)
Least connections
Weighted least connections
URL hashing
Source IP hashing (configurable mask)
Destination IP hashing (configurable mask)
Source and destination IP hashing (configurable mask)
Load Balancing Supported
Server load balancing (TCP, UDP, or generic IP protocols)
Firewall load balancing

Table 1-2 CSM Feature Set Description (continued)

Features
DNS load balancing
Stealth firewall load balancing
Transparent cache redirection
Reverse proxy cache
SSL off-loading
VPN-IPSec load balancing
Generic IP devices and protocols
Stickiness
Cookie sticky with configurable offset and length
SSL ID
Source IP (configurable mask)
HTTP redirection
Redundancy
Sticky state
Full stateful failover (connection redundancy)
Health Checking
HTTP
ICMP
Telnet
TCP
FTP
SMTP
DNS
Return error-code checking
Inband health checking
User-defined TCL scripts
Management
SNMP traps
Full SNMP and MIB support
XML interface for remote CSM configuration
Back-end encryption support.
Workgroup Manager Support
Server Application State Protocol (SASP)

Table 1-3 lists the CSM-S features in this release.

Table 1-3 CSM-S Feature Set Description

Features
Supported Hardware
Supervisor 2 with MSFC2
Supported Software
Cisco IOS software Release 12.2(18)SXD with the Supervisor Engine 2 and MSFC2
SSL Features
SSL initiation
SSL version 2.0 forwarding
URL rewrite
HTTP header insertion
Wildcard proxy
Handshake Protocol
SSL 3.0
SSL 3.1/TLS 1.0
SSL 2.0 (only ClientHello support)
Session reuse
Session renegotiation
Session timeout
Symmetric Algorithms
ARC4
DES
3DES
Asymmetric Algorithms
RSA
Hash Algorithms
MD5
SHA1
Cipher Suites
SSL_RSA_WITH_RC4_128_MD5
SSL_RSA_WITH_RC4_128_SHA
SSL_RSA_WITH_DES_CSC_SHA
SSL_RSA_WITH_3DES_EDE_CBC_SHA
Public Key Infrastructure
RSA key pair generation for certificates up to 2048 bits
Secure key storage in CSM-S Flash memory device
Certificate enrollment for client and server-type proxy services

Table 1-3 CSM-S Feature Set Description (continued)

Features
Importing and exporting of key and certificate (PKCS12 and PEM)
Duplicating keys and certificates on standby CSM-S using the key and certificate import and export mechanism
Manual key archival, recovery, and backup
Key and certificate renewal using the CLI
Graceful rollover of expiring keys and certificates
Auto-enrollment and auto-renewal of certificates
Importing of certificate authority certificates by cut-and-paste or TFTP
Up to 8 levels of certificate authority in a certificate chain
Generating of self-signed certificate
Manual certificate enrollment using cut-and-paste or TFTP of PKCS10 CSR file
Peer (client and server) certificate authentication
Peer (client and server) certificates
Certificate security attribute-based access control lists
Certificate revocation lists (CRL)
Certificate expiration warning
TCP Termination
RFC 1323
Connection aging
Connection rate
NAT[1]/PAT[2]
Client and server
Redundancy
When the CSM-S module is in the standby state, you cannot access SSL services.
To have redundancy, you must use either two CSMs or two CSM-S. You cannot mix a CSM and a CSM-S for a supported redundancy configuration.
High Availability
Failure detection (SLB health monitoring schemes)
Module-level redundancy (stateless)
Serviceability
Password recovery
Statistics and Accounting
Total SSL connection attempts per proxy service
Total SSL connections successfully established per proxy service
Total SSL connections failed per proxy service
Total SSL alert errors per proxy service
Total SSL resumed sessions per proxy service

Table 1-3 CSM-S Feature Set Description (continued)

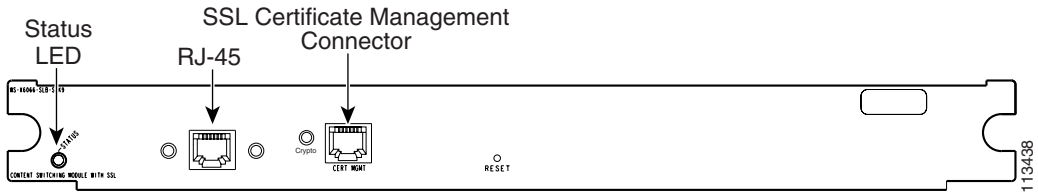
Features
Total encrypted/decrypted packets/bytes per proxy service
Statistics displayed at 1-second, 1-minute, and 5-minute traffic rates for CPU utilization and SSL-specific counters

1. NAT = Network Address Translation
2. PAT = Port Address Translation

Front Panel Description

Figure 1-2 shows the CSM-S front panel.

Figure 1-2 Content Switching Module Front Panel



Note The RJ-45 connector is covered by a removable plate.

Note You are required to make initial SSL daughter card configurations through a direct connection to the CSM-S Certificate Management port (Cert. Mgt). After the initial configurations, you can make an SSH or Telnet connection to further configure the module. See the "Initial SSL Daughter Card Configuration" section on page 5-2.

LEDs

When the CSM-S powers up, it initializes various hardware components and communicates with the supervisor engine. The Status LED indicates the supervisor engine operations and the initialization results. During the normal initialization sequence, the status LED changes from off to red, to orange, to green. The SSL daughter card Crypto LED is unused in this release.

Note For more information on the supervisor engine LEDs, refer to the *Catalyst 6500 Series Switch Module Installation Guide*.

Table 1-4 describes the Status LED operation.

Table 1-4 Content Switching Module LEDs

LED	Color	Description
Status	Off	• The module is waiting for the supervisor engine to provide power. • The module is not online. • The module is not receiving power, which could be caused by the following: – Power is not available to the CSM-S. – Module temperature is over the limit[1].
	Red	• The module is released from reset by the supervisor engine and is booting. • If the boot code fails to run, the LED stays red after power up.
	Orange	• The module is initializing hardware or communicating with the supervisor engine. • A fault occurred during the initialization sequence. • The module has failed to download its Field Programmable Gate Arrays (FPGAs) on power up but continues with the remainder of the initialization sequence and provides the module online status from the supervisor engine. • The module has not received module online status from the supervisor engine. This problem could be caused by the supervisor engine detecting a failure in an external loopback test that it issued to the CSM-S.
	Green	• The module is operational; the supervisor engine has provided module online status.
	Green to orange	• The module is disabled through the supervisor engine CLI [2] using the **set module disable** *mod* command.
Crypto	None.	• Not used. Reserved for future releases.

1. Enter the **show environment temperature** *mod* command to display the temperature of each of the four sensors on the CSM-S.
2. CLI = command-line interface.

RJ-45 Connector

The RJ-45 connector, which is covered by a removable plate, is used to connect a management station device or a test device. This connector is used by field engineers to perform testing and to obtain dump information.

SSL Connector

The Certificate Management (Cert. Mgt.) port connector is used for SSL certificate management and is available to make the necessary connection to the SSL daughter card for initial configuration purposes. After the initial configurations, you can make an SSH or Telnet connection to the SSL daughter card to further configure the module. See Chapter 5 in the *Catalyst 6500 Series Content Switching Module with SSL Installation and Configuration* Note.

CSM-S and SSL Services Module Command Differences

This section describes the differences in command functionality between the SSL Services Module and the CSM-S. The following commands or features in the SSL Services Module software are not available in the CSM-S:

- The **debug ssl-proxy pc** command.
- The stateless redundancy feature using HSRP in standalone mode.
- The **virtual ipaddr ...** command under the ssl-proxy service configuration mode requires the **secondary** keyword. The traffic flow will fail if this command is configured without the **secondary** keyword.

  For example,

  ```
  'virtual ipaddr 90.1.1.1 protocol tcp port 443'  is NOT supported.
  'virtual ipaddr 90.1.1.1 protocol tcp port 443 secondary'  is supported.
  ```

- The gateway forward feature from the SSL Services Module does not work with the CSM-S. This feature is used on the SSL Services Module to enable more traffic to flow to the SSL Services Module.

  For example,

  ```
  ssl-proxy vlan 2
  ipaddr 190.1.1.142 255.255.255.0
  gateway 190.1.1.100 forward
  ```

  This feature does not work on the CSM-S because the SSL daughter card receives packets only for the connections that are serviced by a VIP on the CSM. This feature is used on the SSL Services Module to enable more traffic to flow to the SSL Services Module.

Software Version Information

The CSM-S is a combination of the CSM and the SSL Services Module. The version number has these three parts:

A CSM-S version number

A CSM version number

An SSL Services Module version number.

The version number is in the following format:

<CSM-S version> <CSM version> <SSL Services Module version>

For example, the first software release for the CSM-S may appear as follows:

```
1.1(1) 4.1(3) 2.1(2
```

Note In the following examples, the version numbers are highlighted in **bold** text. There are two **show version** commands available. The **show version** command is available from the supervisor engine CLI and SSL daughter card CLI.

Note The **tech-support processor 0** command displays the CSM software version number.
The **show module** command displays the CSM and SSL bundled software version number.

You can display the version number for the software as follows:

- This example shows how to display the technical support information from the supervisor engine to display the CSM version:

```
Router# show module csm 4 tech-support processor 0
Software version: 4.1(3)
```

- Using the **show module** command from the supervisor engine, for example:

```
Router# show module
Mod Ports Card Type                                Model              Serial No.
--- ----- ---------------------------------------- ------------------ -----------
  1     2  Catalyst 6000 supervisor 2 (Active)    WS-X6K-SUP2-2GE    SAD055104SU
  2     2  Catalyst 6000 supervisor 2 (Hot)       WS-X6K-SUP2-2GE    SAL0702BJKF
  3     3  MWAM Module                            WS-SVC-MWAM-1      SAD071602TZ
  4     3  MWAM Module                            WS-SVC-MWAM-1      SAD071602UT
  5     3  MWAM Module                            WS-SVC-MWAM-1      SAD07200176
  7     0  Switching Fabric Module-136 (Active)   WS-X6500-SFM2      SAL06355FRR
  8    48  48 port 10/100 mb RJ-45 ethernet       WS-X6248-RJ-45     SAD03080474
  9     3  MWAM Module                            WS-SVC-MWAM-1      SAD0649019F
 11     0  CSM with SSL                           WS-X6066-SLB-S-K9  SAD07380300
 12     0  SLB Application Processor Complex      WS-X6066-SLB-APC   SAD061801NA
 13     1  SSL daughter card                             WS-SVC-SSL-1       SAD070303G2

Mod MAC addresses                       Hw     Fw           Sw           Status
--- ----------------------------------- ------ ------------ ------------ -------
  1  0001.6415.ab56 to 0001.6415.ab57   3.2    7.1(1)       12.2(TETONS_ Ok
  2  0006.d65c.5c78 to 0006.d65c.5c79   4.1    7.1(1)       12.2(TETONS_ Ok
  3  0003.feab.9738 to 0003.feab.973f   2.0    7.2(1)       2.1(0.3b)    Ok
  4  0002.fcbe.8500 to 0002.fcbe.8507   2.0    7.2(1)       2.1(0.3b)    Ok
  5  0003.feab.80c8 to 0003.feab.80cf   2.0    7.2(1)       2.1(0.1b)    Ok
  7  0001.0002.0003 to 0001.0002.0003   1.2    6.1(3)       8.3(0.63)TET Ok
  8  0060.09ff.f5c0 to 0060.09ff.f5ef   0.701  4.2(0.24)VAI 8.3(0.63)TET Ok
  9  0005.9a3b.9de8 to 0005.9a3b.9def   0.304  7.2(1)       1.0(0.1)     Ok
 11  0003.feac.a958 to 0003.feac.a95f   1.7                 1.1(1)       Ok
 12  00d0.d32f.03f8 to 00d0.d32f.03ff   1.5                 3.1(6)       Ok
 13  0040.0bf0.1c04 to 0040.0bf0.1c0b   1.2    7.2(1)       2.1(0.59)    Ok
```

- This example shows how to display the SSL proxy version from the SSL daughter card CLI:

```
ssl-proxy> show ssl-proxy version
Cisco Internetwork Operating System Software
IOS (tm) SVCSSL Software (SVCSSL-K9Y9-M), Version 12.2(14.6)SHK(0.28)  INTERIM TEST
SOFTWARE
Copyright (c) 1986-2004 by cisco Systems, Inc.
Compiled Tue 04-May-04 11:05 by integ
Image text-base: 0x00400078, data-base: 0x00B04000

ROM: System Bootstrap, Version 12.2(15)YS1 RELEASE SOFTWARE

ssl-proxy uptime is 0 minutes
System returned to ROM by power-on
System image file is "tftp://255.255.255.255/unknown"
AP Version 1.1(1) 4.1(1) 2.1(1)
```

Configuration Restrictions

SSL flows that are processed by the SSL daughter card are flows that are processed only by the CSM. The SSL daughter card cannot off-load flows that are not load balanced by the CSM.

All VLANs that are configured on the SSL daughter card must also be configured on the CSM. If the CSM is not configured, then the traffic for that VLAN will never arrive at the SSL daughter card.

Note There is no configuration verification between the CSM and SSL daughter card. If only the CSM portion of the configuration is completed, the local real servers will show as operational even before the SSL daughter card is configured. The status will always be operational for local real servers. Because these real servers are configured on the daughter card, they are always assumed to be available.

CSM-S Operation Overview

Clients and servers communicate through the CSM-S using Layer 2 and Layer 3 technology in a specific VLAN configuration. (See Figure 1-3.) In a simple Server Load Balancing (SLB) deployment, clients connect to the client-side VLAN and servers connect to the server-side VLAN. Servers and clients can exist on different subnets. Servers can also be located one or more Layer 3 hops away and connect to the CSM-S through routers.

A client sends a request to one of the module's VIP addresses. The CSM-S forwards this request to a server that can respond to the request. The server then forwards the response to the CSM-S, and the CSM-S forwards the response to the client.

When the client-side and server-side VLANs are on the same subnets, you can configure the CSM-S in single subnet (bridge) mode. For more information, see the "Configuring the Single Subnet (Bridge) Mode" section on page 2-1.

When the client-side and server-side VLANs are on different subnets, you can configure the CSM-S to operate in a secure (router) mode. For more information, see the "Configuring the Secure (Router) Mode" section on page 2-3.

You can set up a fault-tolerant configuration in either the secure (router) or single subnet (bridged) mode using redundant CSM-S modules. For more information, see the "Configuring Fault Tolerance" section on page 9-1.

Single subnet (bridge) mode and secure (router) mode can coexist in the same CSM-S with multiple VLANs.

Figure 1-3 Content Switching Module with SSL and Servers

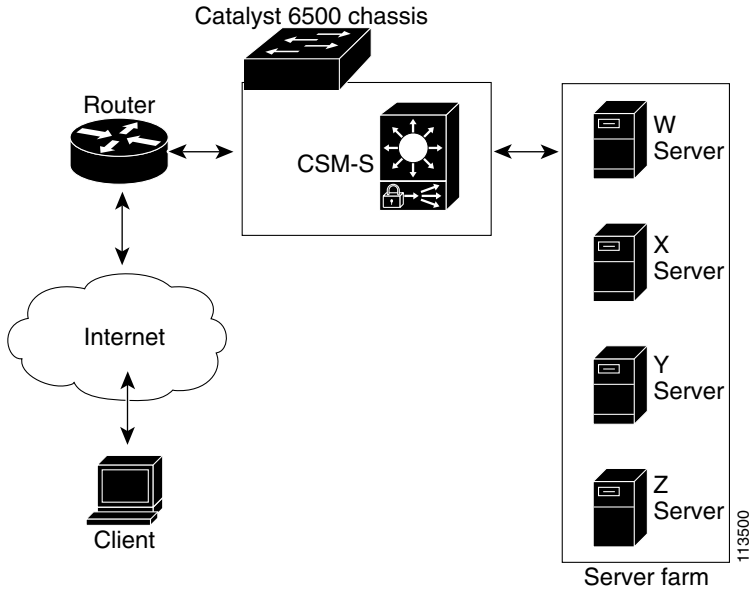
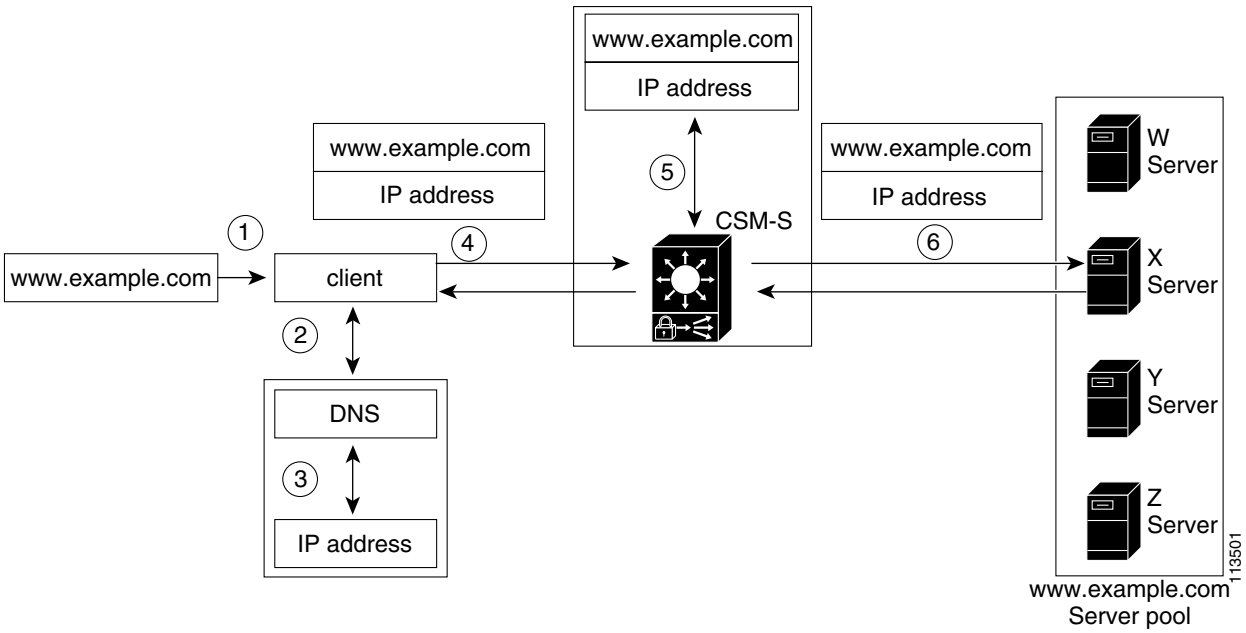


Figure 1-4 describes how the traffic flows between the client and server in a CSM-S environment.

Figure 1-4 Traffic Flow Between Client and Server



Note The numbers in Figure 1-4 correspond to the numbers in the following operation.

When you enter a request for information by entering a URL, the traffic flows as follows:

1. You enter a URL. (Figure 1-4 shows www.example.com as an example.)
2. The client contacts a DNS server to locate the IP address associated with the URL.
3. The DNS server sends the IP address of the virtual IP (VIP) to the client.
4. The client uses the IP address (CSM-S VIP) to send the HTTP request to the CSM-S.
5. The CSM receives the request with the URL, makes a load-balancing decision, and selects a server.

   For example, in Figure 1-4, the CSM-S selects a server (X server) from the www.example.com server pool, replacing its own VIP address with the address of the X server (directed mode), and forwards the traffic to the X server. If the NAT server option is disabled, the VIP address remains unchanged (dispatch mode).
6. The CSM-S performs Network Address Translation (NAT) and TCP sequence numbers translation.

CSM-S Operation with SSL

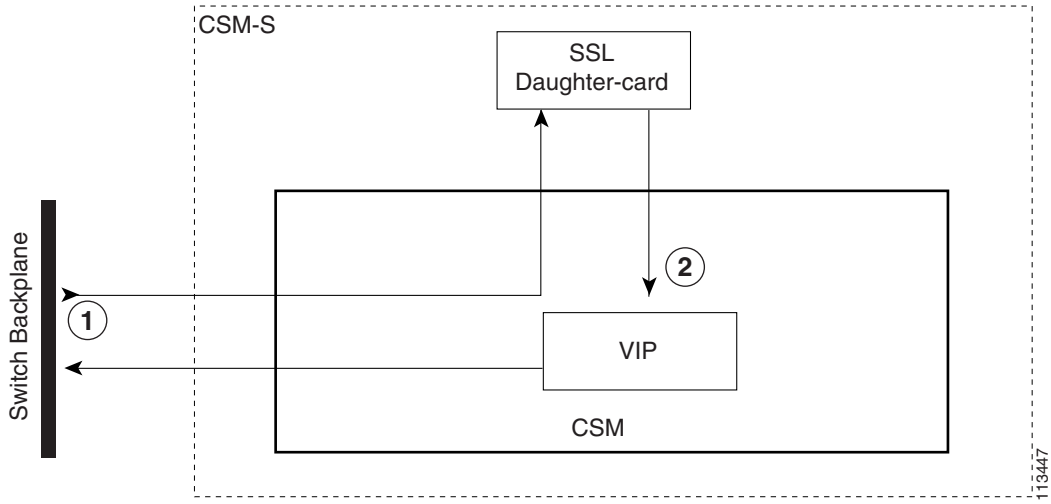
The CSM-S is a CSM with integrated SSL support on an internal daughter card so that communication between the load balancing and SSL modules is local to the CSM-S. The CSM-S configuration is a combination of a CSM and SSL Services Module configuration. See Figure 1-5.

Note SSL services are available only when virtual servers are configured for SSL operation and VLANs have been configured on the module.

All packets to and from the daughter card are routed through the CSM.

The CSM hardware and the SSL daughter card are loosely coupled but the CSM treats the SSL daughter card as a special real server that it knows is locally attached.

Figure 1-5 CSM-S Hardware Configuration



The daughter card runs the SSL software features from SSL releases up to and including SSL release 2.1 supported by Cisco IOS software Release 12.2(18)SXD. See the "Features" section on page 1-2 for a list of supported features for the CSM-S.

The software runs independently on both the CSM and the SSL daughter card. The CSM-S software allows for SSL configuration and flow processing to and from the daughter card. The Cisco IOS software enables the Public Key Infrastructure (PKI) allowing the CSM-S to load and generate certificates and keys for processing the SSL data flows and to configure the SSL software.

To configure the SSL feature, you must access the daughter card through the Certificate Management (Cert. Mgt.) port. The CSM-S baseboard includes a BOOTP server which upon a daughter card boot request supplies the boot information that includes the IP address and the SSL image to load.

Note When the CSM-S first starts up, the time starts at Jan 1, 1970. Once the CSM and SSL daughter card (CSM-S system) is up, then the time is synchronized with the time on the switch supervisor engine.

You may see syslog messages on the SSL daughter card console referring to expired certificates due to the time synchronization condition on first boot. Once the clock synchronization occurs from the supervisor engine, which occurs within a few seconds after the syslog messages are generated, the CSM-S is ready to pass traffic.

To determine when the CSM-S is ready to pass traffic, you can use the Router# **show module csm** *slot #* **status** command.

Two types of syslog messages are displayed when you enter the **show module csm** *slot #* **status** command:

1. When the module is ready to pass traffic, this message displays:

```
SLB Module is online in slot 4.
Configuration Download state: COMPLETE, SUCCESS
```

2. When the module is not ready to pass traffic, this message displays:

```
SLB Module is offline.
Requires CSM module version 3.1.
```

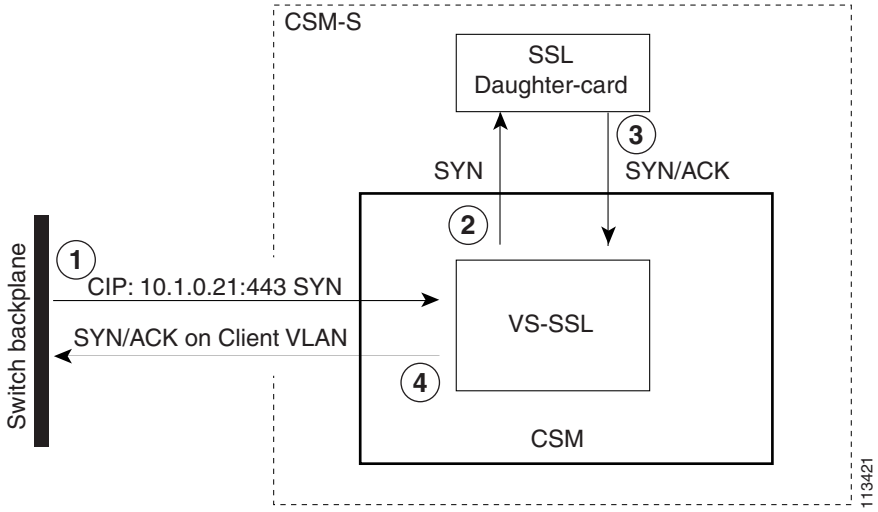
The runtime boot sequence for the CSM-S is as follows:

1. The CSM-S boots.
2. The CSM-S resets the daughter card. The daughter card runs a memory test.
3. When the memory test is complete, the daughter card ROMMON sends a BOOTP request to the CSM-S.
4. The CSM-S sends a BOOTP response containing the MAC address, EOBC IP address, and the flash location from where the daughter card runtime image loads.
5. The SSL console becomes active when the SSL Cisco IOS runtime starts.
6. The SSL software sends a time request to the CSM.
7. The CSM-S indicates to the switch supervisor engine that it is ready to go online.

Client-Side Configuration Traffic Flow

In Figure 1-6, the CSM requires a Layer 4 virtual server configured to accept client traffic on port 443. The server farm associated with this virtual server is configured with the same VIP address as the virtual server and must be marked as being local. Marking the virtual server as local tells the CSM that this server is located on the SSL daughter card. The tables are updated to properly forward traffic to the daughter card.

Figure 1-6 CSM-S Client-Side Configuration



Note The numbers in Figure 1-6 correspond to the numbers in the following operation.

The client-side configuration traffic flow is as follows:

1. When a client SYN-frame is received by the CSM and matches the SSL virtual server, the CSM treats it in the same manner as any Layer 4 virtual server.
2. The destination decision sets up the internal CSM tables to direct all subsequent client traffic on this connection to the SSL daughter card. The reverse tuple is also set up to direct traffic back to the client from the daughter card. The SYN packet is passed to the SSL daughter card for processing.
3. The SSL daughter card processes the SYN-frame and sets up an internal table for connection. The SSL daughter card then responds with a SYN/ACK to the client.
4. The SYN/ACK is received by the CSM and is processed with the reverse tuple. The SYN/ACK is then transmitted to the client through the client VLAN.

This example shows the client-side configuration for the CSM:

```
 vlan 420 client
  ip address 192.168.15.109 255.255.255.0
!
 serverfarm SSL
  nat server
  no nat client
  real 192.168.15.100 local
   inservice
!
 vserver V-SSL
  virtual 192.168.15.200 tcp https
  serverfarm SSL
  persistent rebalance
  inservice
```

This example shows the client-side configuration for the SSL daughter card:

```
ssl-proxy service server_proxy
 virtual ipaddr 192.168.15.100 protocol tcp port 443 secondary
 server ipaddr 192.168.15.200 protocol tcp port 80
 certificate rsa general-purpose trustpoint tier1_tp
 inservice
ssl-proxy vlan 420
 ipaddr 192.168.15.108 255.255.255.0
```

Server-Side Configuration Traffic Flow

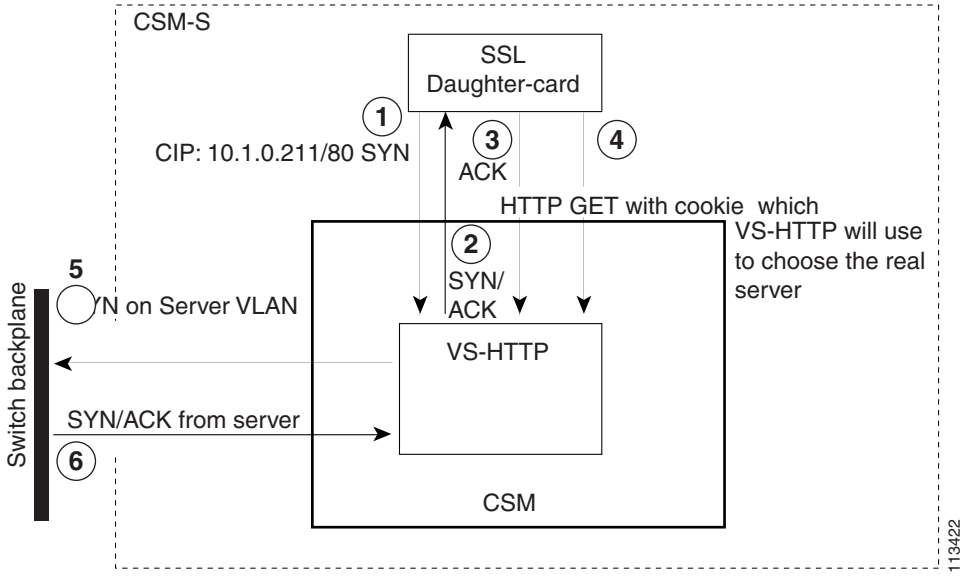
When the SSL daughter card terminates the SSL connection, it must establish a connection to a back-end server that services the request. The server is either a real server in the network or a virtual server configured on the CSM.

Note No configuration checking is done between the CSM and the SSL daughter card. You must make sure that the CSM and SSL daughter card configurations are set up correctly to allow the SSL daughter card to use a virtual server on the CSM for Layer 7 load balancing.

Configuring the CSM as the Back-End Server

Figure 1-7 shows the configuration where the back-end server is a Layer 7 virtual server. The virtual server, VS2, is configured to match the ssl-proxy server configuration of the SSL daughter card.

Figure 1-7 CSM-S Server-Side Configuration with CSM as the Back-End Server



Note The numbers in Figure 1-7 correspond to the numbers in the following operation.

The server-side configuration traffic flow with the CSM as the back-end server is as follows:

1. The SSL daughter card transmits a TCP SYN frame to the target address of the ssl-proxy service.
2. The CSM responds to the SYN that is sent to VS-HTTP with a SYN/ACK to the client IP address, and the SYN/ACK is sent to the SSL daughter card.
3. The SSL daughter card completes the TCP handshake by sending a TCP ACK to the CSM virtual server V-HTTP.
4. The SSL daughter card sends the decrypted HTTP GET request to the CSM virtual server V-HTTP. When the CSM receives this request, it uses the cookie value to determine the actual real server.
5. The CSM sends a TCP SYN to the real server as the client.
6. The real server responds with a TCP SYN/ACK.
7. The CSM continues to operate as it does for Layer 5 and Layer 7 flows for the system.

This example shows the server-side configuration for the CSM:

```
vlan 421 server
  ip address 192.168.17.109 255.255.255.0
!
 serverfarm SLB
  nat server
  no nat client
  real 192.168.17.13
   inservice
!
 vserver VS-HTTP
  virtual 192.168.15.200 tcp www
  serverfarm SLB
  persistent rebalance
  inservice
```

This example shows the server-side configuration for the SSL daughter card:

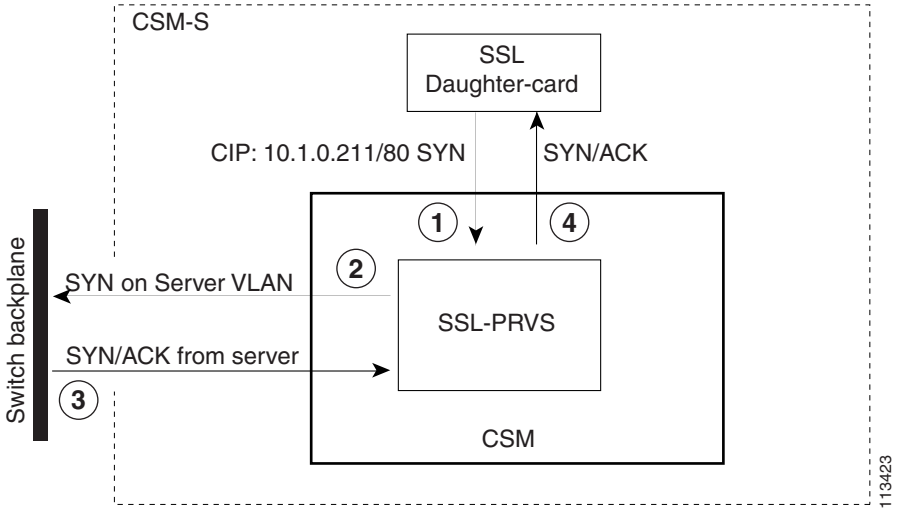
```
ssl-proxy service server_proxy
 virtual ipaddr 192.168.15.100 protocol tcp port 443 secondary
 server ipaddr 192.168.15.200 protocol tcp port 80
 certificate rsa general-purpose trustpoint tier1_tp
 inservice
```

Configuring the Real Server as the Back-End Server

When you configure a real server as the back-end server, the SSL daughter card is configured with the real server's IP address as the SSL-proxy server address. Traffic is sourced by the real server, and the CSM directs traffic from the SSL daughter card to and from the real server.

As shown in Figure 1-8, the CSM is configured with virtual server SSL-PRVS by using a server farm with the predictor-forward option. To properly forward traffic to the real server IP address, the CSM must perform Address Resolution Protocol (ARP) for all possible real servers. For ARP resolution to perform correctly, the server farm SSL real servers must contain the IP address of all possible real servers, but they must not be associated with any virtual server on the CSM. You can also associate health probes with the real servers.

Figure 1-8 CSM-S Server-Side Configuration with a Real Server as the Back-End Server



Note The numbers in Figure 1-8 correspond to the numbers in the following operation.

The server-side configuration traffic flow (with the real server as the back-end server) is as follows:

1. The SSL daughter card transmits the TCP SYN frame to the server address of the ssl-proxy service, and that frame is received by the CSM to match the virtual server SSL-PRVS.
2. A load-balancing decision is made, and the frame is forwarded to the server based on the predictor-forward server farm configuration. The reverse tuple is programmed to catch traffic from the server destined to the SSL daughter card. The frame is transmitted on the server VLAN.
3. When the SYN/ACK frame is received on the server VLAN, it matches the reverse path tuple setup and the frame is forwarded back to the client which is the SSL daughter card.
4. The SYN/ACK is sent to the SSL daughter card.

```
vlan 421 server
ip address 192.168.17.109 255.255.255.0
serverfarm SSLPF
  nat server
  no nat client
  predictor forward
vserver SSL-PFVS
  virtual 0.0.0.0 0.0.0.0 tcp 8888
  vlan local
  serverfarm SSLPF
  persistent rebalance
  inservice
```

This example shows the client-side and server-side configuration for the SSL daughter card:

```
ssl-proxy service server_proxy
 virtual ipaddr 192.168.15.100 protocol tcp port 443 secondary
 server ipaddr 192.168.17.13 protocol tcp port 8888
 certificate rsa general-purpose trustpoint tier1_tp
 inservice
ssl-proxy vlan 420
 ipaddr 192.168.15.108 255.255.255.0
ssl-proxy vlan 421
 ipaddr 192.168.17.108 255.255.255.0
```

Networking with the Content Switching Module with SSL

This chapter describes networking the CSM-S and contains these sections:

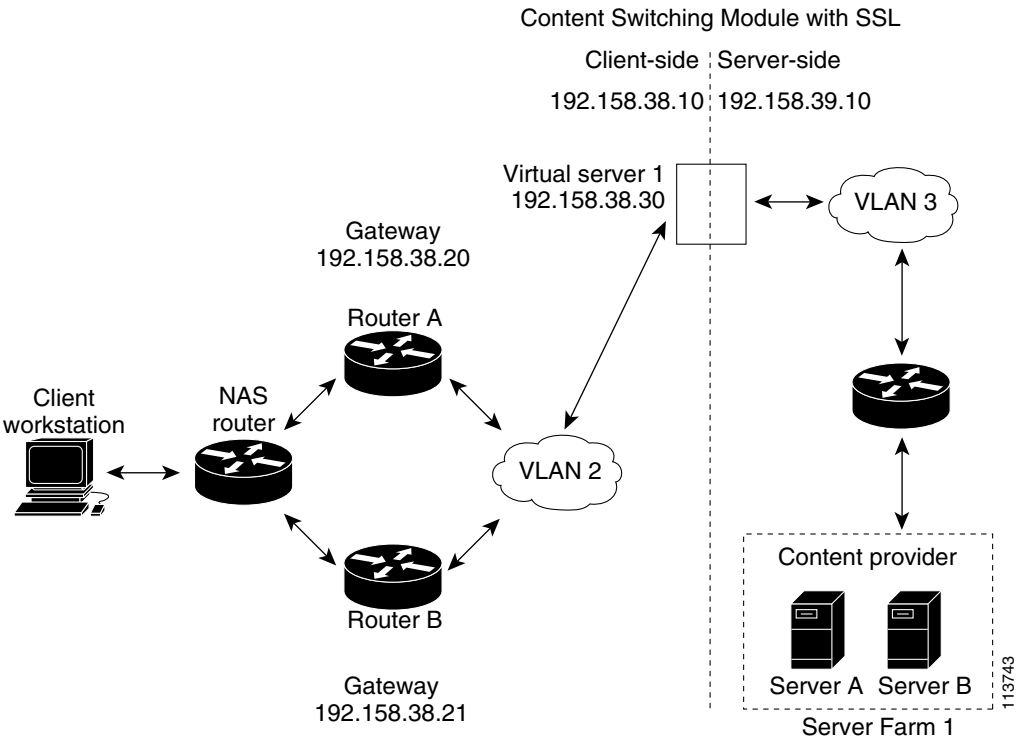
Configuring Modes for Networking

You can configure the CSM-S in a single subnet or bridged mode and a secure or router mode. These sections describe the modes:

Configuring the Single Subnet (Bridge) Mode

In the single subnet (bridge) mode configuration, the client-side and server-side VLANs are on the same subnets. Figure 2-1 shows how the single subnet (bridge) mode configuration is set up.

Figure 2-1 Single Subnet (Bridge) Mode Configuration



Note The addresses in Figure 2-1 refer to the steps in the following task table.

Note You configure single subnet (bridge) mode by assigning the same IP address to the CSM-S client and server VLANs.

To configure content switching for the single subnet (bridge) mode, perform this task:

	Command	Purpose
Step 1	`Router(config-module-CSM)# vlan database`	Enters the VLAN mode[1].
Step 2	`Router(vlan)# vlan 2`	Configures a client-side VLAN[2].
Step 3	`Router(vlan)# vlan 3`	Configures a server-side VLAN.
Step 4	`Router(vlan)# exit`	Exits the mode for the configuration to take effect.
Step 5	`Router(config-module-CSM)# vlan 2 client`	Creates the client-side VLAN 2 and enters the SLB VLAN mode[1].
Step 6	`Router(config-slb-vlan-client)# ip addr 192.158.38.10 255.255.255.0`	Assigns the CSM-S IP address on VLAN 2.
Step 7	`Router(config-slb-vlan-client)# gateway 192.158.38.20`	Defines the client-side VLAN gateway to Router A.
Step 8	`Router(config-slb-vlan-client)# gateway 192.158.38.21`	Defines the client-side VLAN gateway to Router B.

	Command	Purpose
Step 9	`Router(config-slb-vserver)# vlan 3 server`	Creates the server-side VLAN 3 and enters the SLB VLAN mode.
Step 10	`Router(config-slb-vlan-client)# ip addr 192.158.38.10 255.255.255.0`	Assigns the CSM-S IP address on VLAN 3.
Step 11	`Router(config-slb-vlan-client)# exit`	Exits the submode.
Step 12	`Router(config-module-CSM)# vserver VIP1`	Creates a virtual server and enters the SLB virtual server mode.
Step 13	`Router(config-slb-vserver)# virtual 192.158.38.30 tcp www`	Creates a virtual IP address.
Step 14	`Router(config-slb-vserver)# serverfarm farm1`	Associates the virtual server with the server farm[3].
Step 15	`Router(config-module-CSM)# inservice`	Enables the server.

1. Enter the **exit** command to leave a mode or submode. Enter the **end** command to return to the menu's top level.
2. The **no** form of this command restores the defaults.
3. This step assumes that the server farm has already been configured. (See the "Configuring Server Farms" section on page 5-1.)

Note Set the server default routes to Router A gateway (192.158.38.20) or Router B gateway (192.158.38.21).

Configuring the Secure (Router) Mode

In secure (router) mode, the client-side and server-side VLANs are on different subnets.

To configure content switching in secure (router) mode, perform this task:

	Command	Purpose
Step 1	`Router(config-module-CSM)# vlan database`	Enters the VLAN mode[1].
Step 2	`Router(vlan)# vlan 2`	Configures a client-side VLAN[2].
Step 3	`Router(vlan)# vlan 3`	Configures a server-side VLAN.
Step 4	`Router(vlan)# exit`	Exits the mode for the configuration to take effect.
Step 5	`Router(config-module-csm)# vlan 2 client`	Creates the client-side VLAN 2 and enters the SLB VLAN mode.
Step 6	`Router(config-slb-vlan-client)# ip addr 192.158.38.10 255.255.255.0`	Assigns the CSM-S IP address on VLAN 2.
Step 7	`Router(config-slb-vlan-client)# gateway 192.158.38.20`	Defines the client-side VLAN gateway to Router A.
Step 8	`Router(config-slb-vlan-client)# gateway 192.158.38.21`	Defines the client-side VLAN gateway to Router B.
Step 9	`Router(config-module-csm)# vlan 3 server`	Creates the server-side VLAN 3 and enters the SLB VLAN mode.
Step 10	`Router(config-slb-vlan-server)# ip addr 192.158.39.10 255.255.255.0`	Assigns the CSM-S IP address on VLAN 3.
Step 11	`Router(config-slb-vlan-server)# exit`	Exits the submode.

	Command	Purpose
Step 12	`Router(config-module-csm)# vserver VIP1`	Creates a virtual server and enters the SLB virtual server mode.
Step 13	`Router(config-slb-vserver)# virtual 192.158.38.30 tcp www`	Creates a virtual IP address.
Step 14	`Router(config-slb-vserver)# serverfarm farm1`	Associates the virtual server with the server farm[3].
Step 15	`Router(config-module-csm)# inservice`	Enables the server.

1. Enter the **exit** command to leave a mode or submode. Enter the **end** command to return to the menu's top level.
2. The **no** form of this command restores the defaults.
3. This step assumes that the server farm has already been configured. (See the "Configuring Server Farms" section on page 5-1.)

Note Set the server default routes to the CSM-S IP address (192.158.39.10).

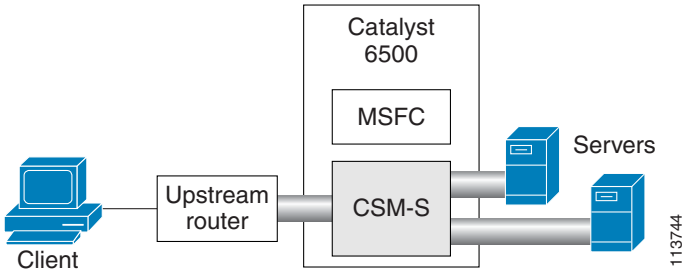
CSM-S Networking Topologies

This section describes CSM-S networking topologies and contains these sections:

CSM-S Inline and MSFC Not Involved

Figure 2-2 shows the CSM-S in a Layer 3 configuration without interaction with the MSFC.

Figure 2-2 CSM-S Inline, MSFC Not Involved



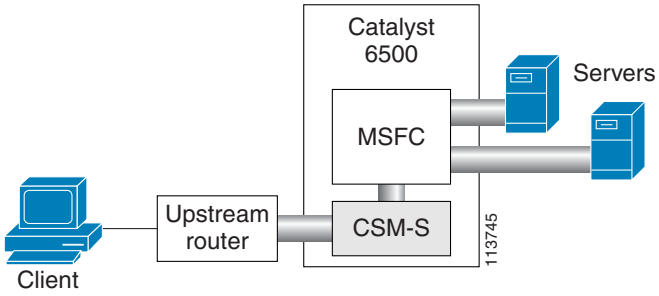
This configuration has these characteristics:

- The MSFC is not routing CSM-S VLANs.
- All server-to-server communications (direct Layer 3 or load balanced) are through the CSM-S.
- The CSM-S must use static routes to the upstream router (default gateway).

CSM-S Inline and MSFC on Server Side

Figure 2-3 shows the CSM-S in a configuration where the MSFC is located on the server side.

Figure 2-3 CSM-S Inline, MSFC Located on Server Side



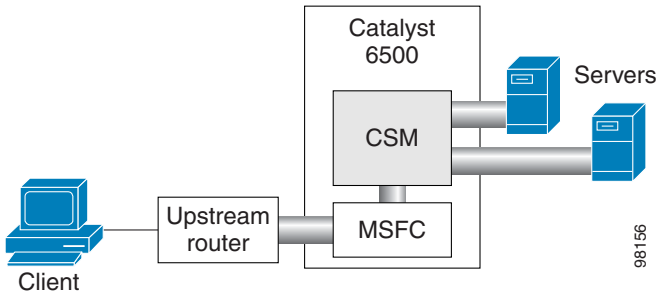
This configuration has these characteristics:

- Server-to-server direct communications bypass the CSM-S.
- Server-to-server load-balanced connections always require secure NAT (SNAT).
- The CSM-S must use static routes to the upstream router (default gateway).
- Routing protocols can be used in the back end.
- Layer-2 rewrite is not possible.

CSM-S Inline and MSFC on Client Side

Figure 2-4 shows the CSM-S in a configuration where the MSFC is located on the client side.

Figure 2-4 CSM-S Inline, MSFC Located on the Client Side



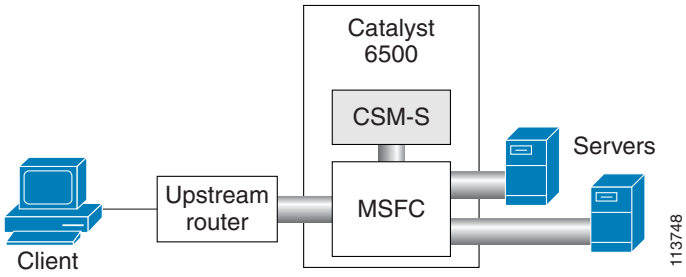
This configuration has these characteristics:

- The configuration is easy to deploy.
- Server-to-server Layer-3 communications pass through the CSM-S.
- Routing protocols can be used between the MSFC and the upstream router.
- All traffic to or from the servers passes through the CSM-S.

CSM-S in Aggregate Mode

Figure 2-5 shows the CSM-S in an aggregate-mode configuration.

Figure 2-5 CSM-S Located in Aggregate Mode



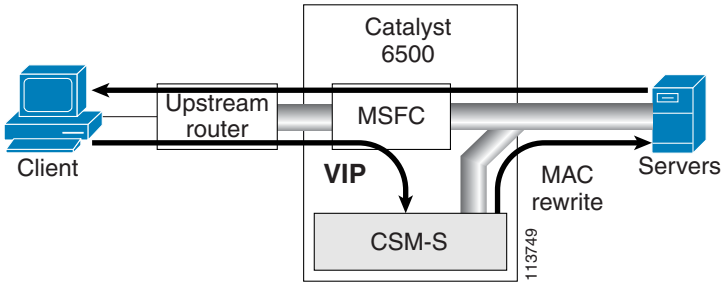
This configuration has these characteristics:

- The CSM-S is not inline, and the module does not see unnecessary traffic.
- Easy routing and CSM-S configuration.
- Requires PBR or client SNAT because return traffic is required.
- Server-to-server load-balanced connections always require SNAT.
- Layer-2 rewrite is not possible.

Direct Server Return

Figure 2-6 shows the CSM-S in a direct server return configuration.

Figure 2-6 Direct Server Return



This configuration has these characteristics:

- High throughput or bandwidth is not required in the load balancer.
- The load balancer does not recognize return traffic.
- TCP flows always have to be timed out.
- TCP termination is not possible (only Layer 4 load balancing).
- Inband health monitoring is not possible.
- Servers must be Layer-2 adjacent with a loopback address.

Routing with the CSM-S

When forwarding and maintaining load-balancing connections, the CSM-S must make routing decisions. However, the CSM-S does not run any routing protocols and does not have access to the MSFC routing tables. The CSM-S builds its own routing table with three types of entries:

- Directly attached IP subnets

  These subnets are configured on the CSM-S client or the server VLANs.

- Default gateways

  Default gateways are configured with the **gateway** keyword from within a client or server VLAN configuration submode. See Chapter 4, "Configuring VLANs." In this release, you may have up to 511 default gateways. However, you cannot have more than seven default gateways for the same VLAN.

  Most configurations have (or can be simplified to have) a single default gateway. This gateway points to the upstream router (or to an HSRP IP address that represents the upstream router pair) and eventually to various static routes.

- Static routes

  Static routes are configured with the **route** keyword from within a client or server VLAN configuration submode of configuration. See Chapter 4, "Configuring VLANs." Static routes are very useful when some servers are not Layer 2 adjacent.

Multiple default gateways are supported, however, they create a situation where if the CSM-S needs to make a routing decision to an unknown destination, the CSM-S will randomly select one of the gateways without your intervention or control. To control this behavior, use the predictor-forward option.

There are three situations in which the CSM-S must make a routing decision:

- Upon receiving a new connection.

  At this time, the CSM-S decides where to send the return traffic for that connection. Unlike other devices, the CSM-S does not perform a route lookup, but it memorizes the source MAC address from where the first packet of the connection was received. Return traffic for that connection is sent back to the source MAC address. This behavior also works with redundancy protocols between upstream routers, such as HSRP.

- The CSM-S is configured in router mode.

  The servers are pointing to the CSM-S as their default gateway, and the servers are originating connections.

- A server farm is configured with the predictor-forward option. (See Chapter 5, "Configuring Real Servers and Server Farms.") This predictor instructs the CSM-S to route the connection instead of load balancing it.

With multiple gateways, the first two situations can be simplified by using a server farm configured with the gateway as a unique real server. (See the "Configuring the Source NAT for Server-Originated Connections to the VIP" section on page A-7.)

Protecting Against Denial-of-Service Attacks

The CSM-S implements a variety of features to protect the devices that it is load balancing and to protect itself from a DoS attack. You cannot configure many of these features because they are controlled by the CSM-S and adjust to the amount of incoming traffic.

The CSM-S provides these DoS-protection features:

- SYN cookies

  Note Do not confuse a SYN cookie with synchronization of cookies because these are different features. This discussion refers only to SYN cookies.

  When the number of pending connections exceeds a configurable threshold, the CSM-S begins using SYN cookies, encrypting all of the connection state information in the sequence numbers that it generates. This action prevents the CSM-S from consuming any flow state for pending (not fully established) TCP connections. This behavior is fully implemented in hardware and provides a good protection against SYN attacks.

- Connection pending timeout

  This feature is configurable on a per-virtual server basis and allows you to time out connections that have not been properly established within the configured timeout value specified in seconds.

- Connection idle timeout

  This feature is configurable on a per-virtual server basis and allows you to time out established connections that have not been passing traffic for longer than an interval configured on a timer.

- Generic TCP termination

  Some connections may not require TCP termination for Layer 7 load balancing. You can configure any virtual server to terminate all incoming TCP connections before load balancing those connections to the real servers. This configuration allows you to take advantage of all the CSM-S DoS features located in Layer 4 load-balancing environments.

Getting Started

This chapter describes what is required before you begin configuring the CSM-S and contains these sections:

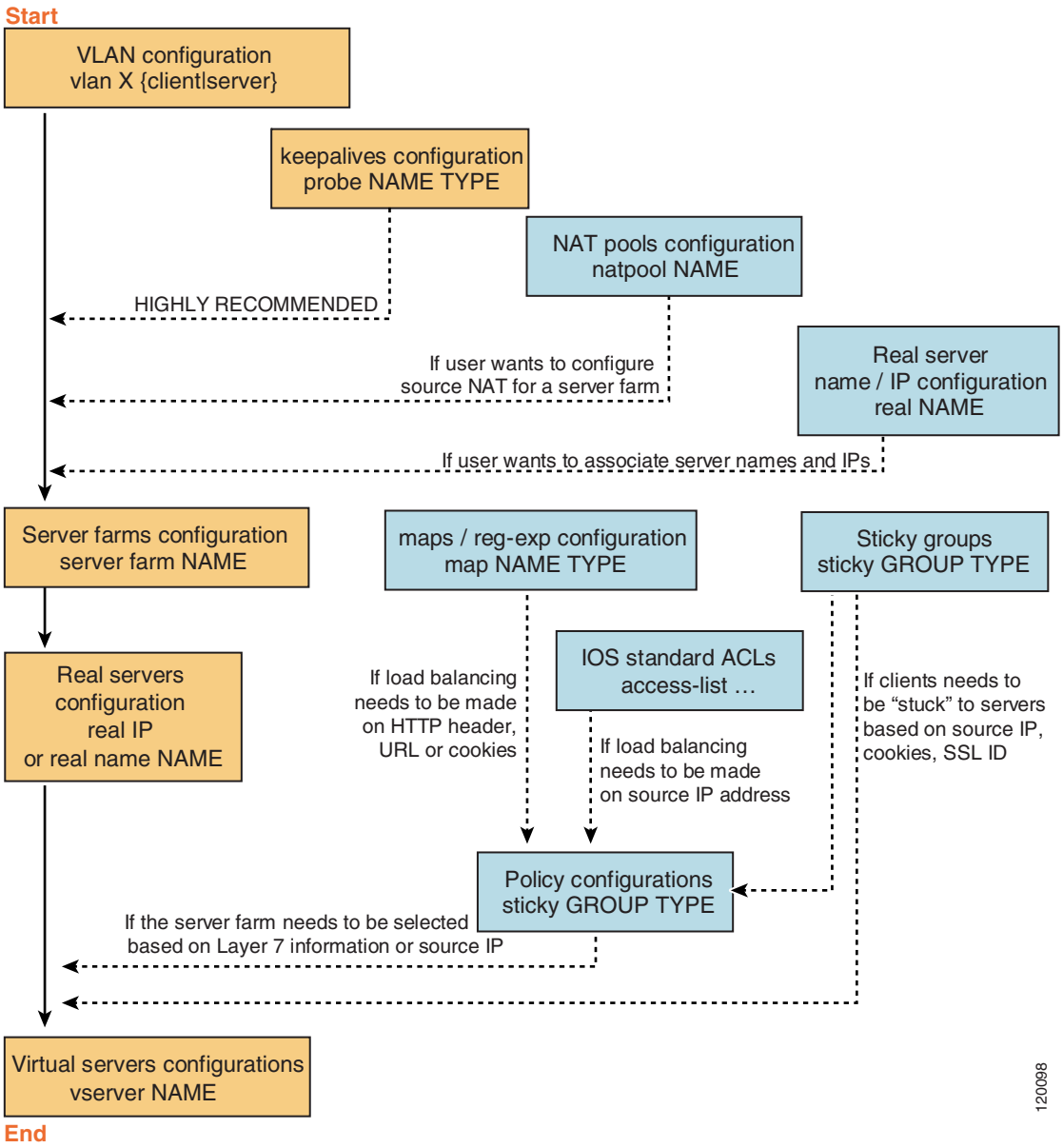
Configuration Overview

The configuration process assumes that the switch is in the RP mode. Figure 3-1 shows an overview of the required and optional operations in the basic CSM-S configuration process. Figure 3-2 shows an overview of the SSL portion of the configuration process.

Note Configuring policies is not necessary for Layer 4 load balancing.

These sections describe how to configure the required parameters:

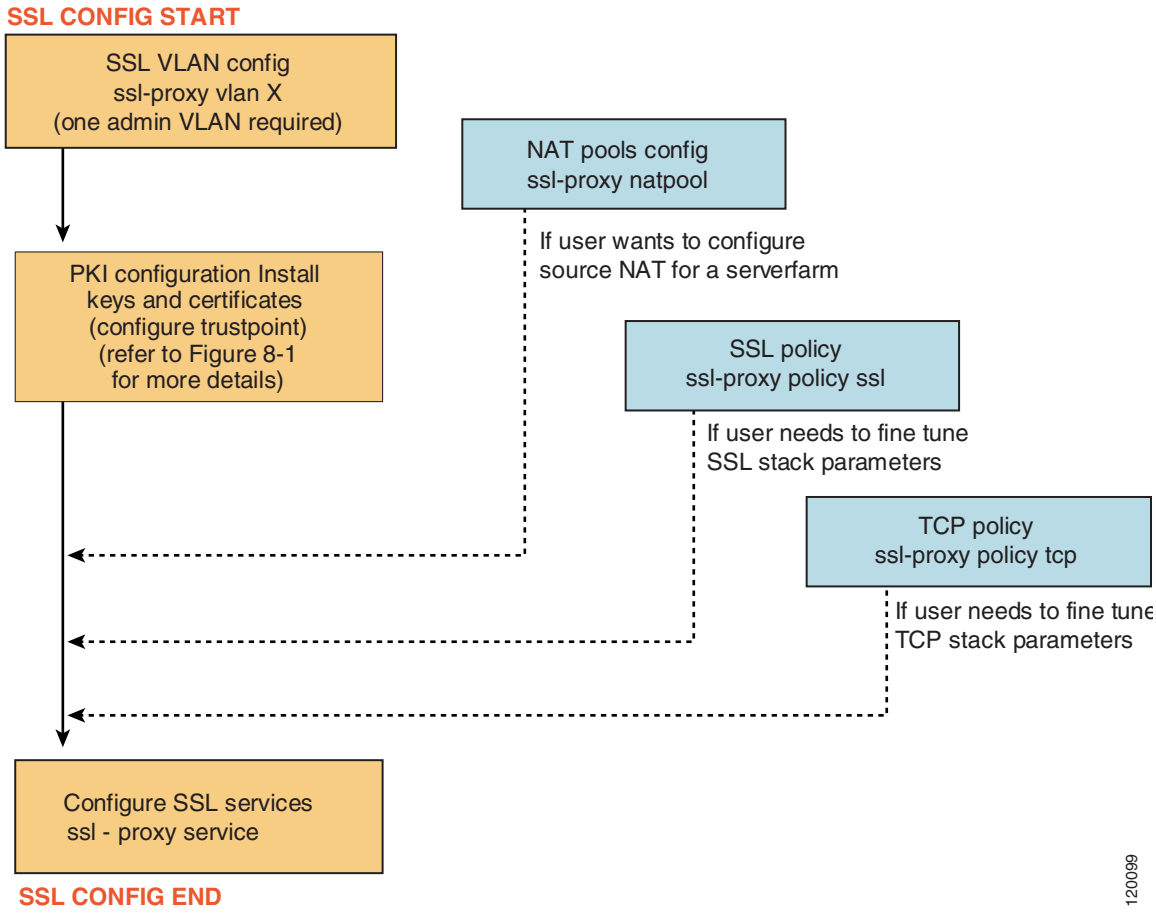
Figure 3-1 CSM-S Basic Configuration Overview



After you configure the required load-balancing parameters on the CSM-S, you can configure the optional parameters in the following sections:

- Configuring Redirect Virtual Servers, page 6-6
- Configuring Client NAT Pools, page 5-6
- Configuring Server-Initiated Connections, page 5-7
- Configuring TCP Parameters, page 6-4

Figure 3-2 CSM-S SSL Configuration Overview



To configure the SSL parameters, see the following sections:

- CSM-S and SSL Services Module Command Differences, page 1-10
- Initial SSL Daughter Card Configuration, page 7-2
- Configuring SSL for Client-Side and Server-Side Operation, page 7-6

To work with advanced configurations, see the following sections in Chapter 2 through Chapter 11:

- Configuring the Single Subnet (Bridge) Mode, page 2-1
- Configuring the Secure (Router) Mode, page 2-3
- Configuring URL Hashing, page 5-7
- Configuring Generic Header Parsing, page 6-12
- Configuring SSL for Client-Side and Server-Side Operation, page 7-6
- Configuring Route Health Injection, page 10-5
- Configuring Fault Tolerance, page 9-1
- Configuring Persistent Connections, page 10-13
- Configuring HSRP, page 9-5
- Configuring Connection Redundancy, page 9-9

Operating System Support

The CSM-S is supported on switches running Cisco IOS software only. Because the CSM-S is configured through the MSFC CLI, you must first session into the MSFC for access to the MSFC CLI. All Layer 2 configurations (such as VLAN and port associations) are performed on the supervisor engine when using a switch running the Cisco IOS software.

Note When running the CSM-S on a switch, configured VLANs are automatically added to the trunk or channel that connects the CSM-S to the switch backplane. In a switch running both the Catalyst operating system and Cisco IOS software, you will have to manually add the CSM-S VLANs to the trunk or channel.

Preparing to Configure the CSM-S

Before you configure the CSM-S, you must take these actions:

- Be sure that the Cisco IOS versions for the switch and the module match. Refer to the *Catalyst 6500 Series Switch Content Switching Module Installation Guide*.
- Before you can configure server load balancing, you must obtain the following information:
  - Network topology that you are using in your installation
  - Real server IP addresses
  - An entry for the CSM-S VIPs in the Domain Name Server (DNS) (if you want them to be reached through names)
  - Each virtual server's IP address
- Configure VLANs on the Catalyst 6500 series switch before you configure VLANs for the CSM-S. VLAN IDs must be the same for the switch and the module. Refer to the *Catalyst 6500 Series Switch Software Configuration Guide* for details.

  This example shows how to configure VLANs:

```
Router# configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
Router(config)# vlan 130
Router(config-vlan)# name CLIENT_VLAN
Router(config-vlan)# exit
Router(config)# vlan 150
Router(config-vlan)# name SERVER_VLAN
```

```
Router(config-vlan)# end
```

- Place the physical interfaces that connect to the servers or to the clients in the corresponding VLAN.

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

```
Router>
Router> enable
Router# config
Router(config)# interface 3/1
Router(config-if)# switchport
Router(config-if)# switchport access vlan 150
Router(config-if)# no shutdown
Router(vlan)# exit
```

- If the Multilayer Switch Function Card (MSFC) is used on the next-hop router on either the client or the server-side VLAN, then you must configure the corresponding Layer 3 VLAN interface.

Caution You cannot use the MSFC simultaneously as the router for both the client and the server side unless policy-based routing or source NAT is used and the CSM-S is configured in router mode. This situation occurs because the CSM-S must see both flow directions that load balances or forwards. If you use the CSM-S in bridge (single subnet) mode, do not configure the Layer 3 VLAN interface on the MSFC for both the client and the server side. If you use the CSM-S in router mode, do not configure the Layer 3 VLAN interface on the MSFC for both the client and the server side unless you properly configure policy-based routing or source NAT to direct return traffic back to the CSM-S.

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

```
Router>
Router> enable
Router# config
Router(config)# interface vlan 130
Router(config-if)# ip address 10.10.1.10 255.255.255.0
Router(config-if)# no shutdown
Router(vlan)# exit
```

Using the Command-Line Interface

The software interface for the CSM-S is the Cisco IOS command-line interface. To understand the Cisco IOS command-line interface and Cisco IOS command modes, refer to Chapter 2 in the *Catalyst 6500 Series Switch Cisco IOS Software Configuration Guide*.

Note Because each prompt has a character limit, some prompts may be truncated. For example Router(config-slb-vlan-server)# may appear as Router(config-slb-vlan-serve)#.

Accessing Online Help

In any command mode, you can get a list of available commands by entering a question mark (?) as follows:

```
Router> ?
```

or

```
Router(config)# module CSM 5
Router(config-module-CSM)# ?
```

Note Online help shows the default configuration values and ranges available to commands.

Saving and Restoring Configurations

For information about saving and restoring configurations, refer to the *Catalyst 6500 Series Switch Cisco IOS Software Configuration Guide*.

Configuring SLB Modes

Server load balancing on the Catalyst 6500 series switch can be configured to operate in two modes: the routed processor (RP) mode and the CSM mode. The switch configuration does not affect CSM-S operation. By default, the CSM-S is configured in RP mode. The RP mode allows you to configure one or multiple CSM-S modules in the same chassis and run Cisco IOS SLB on the same switch.

Note The RP mode is the default mode and is the recommended mode. The CSM mode is used for backward compatibility only for releases with CSM software images previous to release 2.1. When installing a new CSM or CSM-S image, use the RP mode.

The CSM mode allows you to configure a single CSM-S only. The CSM mode is supported for backward compatibility with previous software releases. The single CSM-S configuration will not allow Cisco IOS SLB to run on the same switch.

The following sections provide information about the modes:

- Mode Command Syntax, page 3-6
- Migrating Between Modes, page 3-7
- Differences Between the CSM and RP Modes, page 3-8
- Changing Modes, page 3-9

Mode Command Syntax

Before you can enter the CSM-S configuration commands on the switch, you must specify the CSM-S that you want to configure. To specify a CSM-S for configuration, use the **module csm** *slot-number* command. The *slot-number* value is the chassis slot where the CSM-S being configured is located.

The **module csm** command places you in CSM-S configuration submode. All additional configuration commands that you enter apply to the CSM-S that is installed in the slot you have specified.

Note Unless otherwise specified, all the examples in this publication assume that you have already entered this command and entered the configuration submode for the CSM-S that you are configuring.

The command syntax for the CSM-S mode and RP mode configuration is identical with these exceptions:

- When configuring in the CSM mode, you must prefix each top-level command with **ip slb**.
- Prompts are different for the CSM mode and RP mode configurations.

To configure a virtual server for a CSM-S in slot 5, perform this task:

	Command	Purpose
Step 1	`Router(config)# module csm 5`	Specifies the location of the CSM-S that you are configuring.
Step 2	`Router(config-module-csm)# vserver VS1`	Configures the virtual server.

This example shows the complete list of CSM-S commands in the config-module-csm mode:

```
Router# configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
Router(config)# module csm 5
Router(config-module-csm)# ?
SLB CSM module config
  arp          configure a static ARP entry
  capp         configure Content Application Peering Protocol
  default      Set a command to its defaults
  dfp          configure Dynamic Feedback Protocol manager
  exit         exit SLB CSM module submode
  ft           configure CSM fault tolerance (ft) feature
  map          configure an SLB map
  natpool      configure client nat pool
  no           Negate a command or set its defaults
  owner        configure server owner
  policy       configure an SLB policy
  probe        configure an SLB probe
  real         configure module real server
  script       configure script files and tasks
  serverfarm   configure a SLB server farm
  standby
  static       configure static NAT for server initiated connections
  sticky       configure a sticky group
  variable     configure an environment variable
  vlan         configure a vlan
  vserver      configure an SLB virtual server
  xml-config   settings for configuration via XML
```

Migrating Between Modes

Existing CSM-S configurations are migrated to the new configuration when the mode is changed from CSM to RP using the **ip slb mode** command. If a CSM-S configuration exists, you are prompted for the slot number.

You can migrate from an RP mode configuration to CSM mode configuration on the Catalyst 6500 series switch. You can migrate manually only from a Cisco IOS SLB configuration to a CSM-S configuration.

Differences Between the CSM and RP Modes

The CSM and RP modes only affect the way in which the CSM-S is configured from the CLI, not the operation and functionalities of the CSM-S itself. The RP mode is required to configure multiple CSM-S modules in one chassis as well as the Cisco IOS SLB in the same chassis with a CSM-S.

CSM Mode

You can use the **ip slb mode csm** command mode to configure a CSM-S in 1.x releases. This mode allows the configuration of a single CSM-S in the chassis. (Other CSMs or Cisco IOS SLB cannot be configured in the same chassis.)

In this mode, all the CSM-S configuration commands begin with **ip slb**.

The CSM-S **show** commands begin with **show ip slb**.

This mode is not recommended where it is provided as an option in the Cisco IOS CLI for backward compatibility if you are using CSM software 2.1 or later releases.

The following is an example of a configuration for a single CSM-S in the chassis:

```
Cat6k# show running-config
Building configuration...
Current configuration : 5617 bytes

ip slb mode csm
ip slb vlan 110 server
ip address 10.10.110.1 255.255.255.0

ip slb vlan 111 client
ip address 10.10.111.2 255.255.255.0
gateway 10.10.111.1

ip slb probe HTTP_TEST http
request method get url /probe/http_probe.html
expect status 200
interval 5
failed 5

ip slb serverfarm WEBFARM
nat server
no nat client
real 10.10.110.10
inservice
real 10.10.110.20
inservice
probe HTTP_TEST

ip slb vserver HTTPVIP
virtual 10.10.111.100 tcp www
persistent rebalance
serverfarm WEBFARM
inservice
```

RP Mode

You can use the **ip slb mode rp** command mode (the default) to configure multiple CSM-S modules in a chassis with Cisco IOS SLB. You can only configure the CSM-S using this mode starting from release 2.1.

In this mode, the CSM-S is configured from this command submode:

mod csm *X*

The *X* is the slot number of the CSM-S that you want to configure.

CSM-S **show** commands start with **show mod csm** *X*.

Beginning with CSM software release 2.1, the RP mode is the recommended mode when configuring the CSM-S. While in this mode, all the commands apply to Cisco IOS SLB and not to a CSM-S in the chassis. These commands begin with **ip slb**.

The following is an example of a configuration for a single CSM-S in the chassis:

```
Cat6k# show running-config
Building configuration...

Current configuration : 5597 bytes
!---

module ContentSwitchingModule 5
vlan 110 server
ip address 10.10.110.1 255.255.255.0

vlan 111 client
ip address 10.10.111.2 255.255.255.0
gateway 10.10.111.1

probe HTTP_TEST http
request method get url /probe/http_probe.html
expect status 200
interval 5
failed 5

serverfarm WEBFARM
nat server
no nat client
real 10.10.110.10
inservice
real 10.10.110.20
inservice
probe HTTP_TEST

vserver HTTPVIP
virtual 10.10.111.100 tcp www
persistent rebalance
serverfarm WEBFARM
inservice
```

Changing Modes

You can change the CSM operating mode from CSM mode to RP mode or RP mode to CSM mode. The next sections provide examples of how to change the modes.

CSM Mode to RP Mode

This example shows how to change from CSM mode to RP mode. This example is typical of a migration from CSM 1.x to 2.1 or later releases and does not require a module reset.

```
Cat6k# configure terminal
 Enter configuration commands, one per line.  End with CNTL/Z.

Cat6k(config)# ip slb mode ?
   csm  SLB in Content Switching Module
   rp   SLB in IOS system

Cat6k(config)# ip slb mode rp
 % The current SLB mode is CSM-SLB.
 % You are selecting RP-SLB mode.
 % All configuration for CSM-SLB will be moved to module submode.
 % Confirm switch to RP-SLB mode? [no]: yes
 % Enter slot number for CSM module configuration, 0 for none [5]: 5
 % Please save the configuration.
Cat6k(config)# end

Cat6k# write
 Building configuration...
 [OK]
Cat6k#
```

RP Mode to CSM Mode

This example shows how to migrate from RP mode to CSM mode and requires a module reset:

```
Cat6k# configure terminal
 Enter configuration commands, one per line.  End with CNTL/Z.

Cat6k(config)# ip slb mode ?
   csm  SLB in Content Switching Module
   rp   SLB in IOS system

Cat6k(config)# ip slb mode csm
 % The current SLB mode is RP-SLB.
 % You are selecting CSM-SLB.
 % All SLB configurations for RP will be ERASED.
 % After execution of this command, you must
 % write the configuration to memory and reload.
 % CSM-SLB module configuration will be moved to ip slb submodes.
 % Confirm switch to CSM-SLB mode? [no]: yes
 % Enter slot number for CSM module configuration, 0 for none [5]: 5
 % Please save the configuration and reload.

Cat6k(config)# end
Cat6k# write
 Building configuration...
Cat6k# reload
 Proceed with reload? [confirm] y
Verify Mode Operation
```

Verifying the Configuration

To confirm that your configuration is working properly, use these commands in RP mode:

```
Cat6k# show ip slb mode
     SLB configured mode = rp

Cat6k# configure terminal
 Enter configuration commands, one per line.  End with CNTL/Z.

Catk6-1(config)# ip slb ?
   dfp           configure Dynamic Feedback Protocol manager
   entries       initial and maximum SLB entries
   firewallfarm  configure an SLB firewall farm
   mode          configure SLB system mode
   natpool       define client nat pool
   probe         configure an SLB probe
   serverfarm    configure an SLB server farm
   vserver       configure an SLB virtual server
```

To confirm that your configuration is working properly, use these commands in Cisco IOS SLB mode:

```
Cat6k(config)# module csm 5
Cat6k(config-module-csm)# ?
 SLB CSM module config
   default     Set a command to its defaults
   dfp         configure Dynamic Feedback Protocol manager
   exit        exit SLB CSM module submode
   ft          configure CSM fault tolerance (ft) feature
   map         configure an SLB map
   natpool     configure client nat pool
   no          Negate a command or set its defaults
   policy      configure an SLB policy
   probe       configure an SLB probe
   serverfarm  configure an SLB server farm
   static      configure static NAT for server initiated connections
   sticky      configure a sticky group
   vlan        configure a vlan
   vserver     configure an SLB virtual server
```

To confirm that a single CSM-S in the chassis configuration is working properly, use these commands in CSM mode:

```
Cat6k# show ip slb mode
     SLB configured mode = csm

Catk6-1# configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.

Cat6k(config)# ip slb ?
   dfp         configure Dynamic Feedback Protocol manager
   ft          configure CSM fault tolerance (ft) feature
   map         configure an SLB map
   mode        configure SLB system mode
   natpool     configure client nat pool
   policy      configure an SLB policy
   probe       configure an SLB probe
   serverfarm  configure an SLB server farm
   static      configure static NAT for server initiated connections
   sticky      configure a sticky group
   vlan        configure a vlan
   vserver     configure an SLB virtual server
```

Upgrading to a New Software Release

This section describes three methods for upgrading the CSM-S:

- Upgrading from the Supervisor Engine Bootflash, page 3-12
- Upgrading from a PCMCIA Card, page 3-13
- Upgrading from an External TFTP Server, page 3-14

Note When upgrading to a new software release, you must upgrade the CSM-S image before upgrading the Cisco IOS image. Failure to do so prohibits the supervisor engine from recognizing the CSM-S. In this case, you would have to downgrade the Cisco IOS image, upgrade the CSM-S image, and then upgrade the Cisco IOS image.

Note During the CSM-S upgrade, both the CSM and the SSL daughter card images are upgraded. You cannot use a CSM image on the CSM-S, and you cannot use a CSM-S image on a CSM.

To upgrade the CSM-S, you need to session into the CSM-S module being upgraded. During the upgrade, enter all commands on a console connected to the supervisor engine. Enter each configuration command on a separate line. To complete the upgrade, enter the **exit** command to return to the supervisor engine prompt. See the "Configuring SLB Modes" section on page 3-6.

Caution You must enter the **exit** command to terminate sessions with the CSM-S that is being upgraded. If you do not terminate the session and you remove the CSM-S from the Catalyst 6500 series chassis, you cannot enter configuration commands to the CSM-S unless you press **Ctrl** + **^**, enter **x**, and enter the **disconnect** command at the prompt.

Upgrading from the Supervisor Engine Bootflash

Note Refer to the *Catalyst 6500 Series Supervisor Engine Flash PC Card Installation Note* for instructions on loading images into bootflash.

To upgrade the CSM-S from the supervisor engine bootflash, perform these steps:

Step 1 Enable the TFTP server to supply the image from bootflash as follows:

```
Router>
Router> enable
Router# configure terminal
Router(config)# tftp-server sup-bootflash:c6slb-apc.revision-num.bin
Router(config)
```

Step 2 Set up a session between the supervisor engine and the CSM-S:

```
Router# session slot csm-slot-number processor 0
```

Step 3 Load the image from the supervisor engine to the CSM-S:

```
CSM> upgrade 127.0.0.zz c6slb-apc.revision-num.bin
```

The *zz* is 12 if the supervisor engine is installed in chassis slot 1.
The *zz* is 22 if the supervisor engine is installed in chassis slot 2.

Note The supervisor engine can be installed in chassis slot 1 or slot 2 only.

Step 4 Close the session to the CSM-S, and return to the Cisco IOS prompt:

```
CSM> exit
```

Step 5 Reboot the CSM-S by power cycling the CSM-S or by entering the following commands on the supervisor engine console:

```
Router(config)# hw-module module csm-slot-number reset
```

Upgrading from a PCMCIA Card

Note Throughout this publication, the term *Flash PC card* is used in place of the term *PCMCIA card*.

To upgrade the CSM-S from a removable Flash PC card inserted in the supervisor engine, perform these steps:

Step 1 Enable the TFTP server to supply the image from the removable Flash PC card:

```
Router>
Router> enable
Router# configure terminal
Router(config)# tftp-server slotx:c6slb-apc.revision-num.bin
```

The *x* value is 0 if the Flash PC card is installed in supervisor engine PCMCIA slot 0.

Step 2 Set up a session between the supervisor engine and the CSM-S:

```
Router# session slot csm-slot-number processor 0
```

Step 3 Load the image from the supervisor engine to the CSM-S:

```
CSM> upgrade slot0: c6slb-apc.revision-num.bin
```

Note The supervisor engine can only be installed in chassis slot 1 or slot 2.

Step 4 Close the session to the CSM-S and return to the Cisco IOS prompt:

```
CSM> exit
```

Step 5 Reboot the CSM-S by power cycling the CSM-S or by entering the following commands on the supervisor engine console:

```
Router# hw-module module csm-slot-number reset
```

Upgrading from an External TFTP Server

To upgrade the CSM-S from an external TFTP server, perform these steps:

Step 1 Create a VLAN on the supervisor engine for the TFTP CSM-S run-time image download.

Note You can use an existing VLAN; however, for a reliable download, you should create a VLAN specifically for the TFTP connection.

Step 2 Configure the interface that is connected to your TFTP server.

Step 3 Add the interface to the VLAN.

Step 4 Enter the CSM-S **vlan** command.

See Chapter 4, "Configuring VLANs" for more information.

Step 5 Add an IP address to the VLAN for the CSM-S.

Step 6 Enter the **show csm** *slot* **vlan detail** command to verify your configuration.

See Chapter 4, "Configuring VLANs" for more information.

Step 7 Verify the CSM-S connectivity to the TFTP server:

```
Router# ping module csm csm-slot-number TFTP-server-IP-address
```

Step 8 Set up a session between the supervisor engine and the CSM-S:

```
Router# session slot csm-slot-number processor 0
```

Step 9 Upgrade the image:

```
CSM> upgrade TFTP-server-IP-address c6slb-apc.rev-number.bin
```

Step 10 Close the session to the CSM-S, and return to the Cisco IOS prompt:

```
CSM> exit
```

Step 11 Reboot the CSM-S by power cycling the CSM-S or by entering the following commands on the supervisor engine console:

```
Router# hw-module module csm-slot-number reset
```

Recovering a Lost Password

Recovering a password for SSL on the CSM-S does not require that you load a separate software image on the system. To recover passwords, use the special commands from the Certificate Management (Cert. Mgt.) port on the CSM-S front panel. Due to security concerns, you can recover the password through this port only.

When recovering lost SSL passwords, the following conditions apply:

- You must have a console connection to both the CSM and the SSL daughter card.
- All traffic to and from the SSL daughter card is interrupted when the SSL daughter card is rebooted during the password recovery process.

- Use the following prompts when recovering lost passwords:
  - `CSM>` or `VENUS` for the CSM console
  - `ssl-proxy#` for the SSL daughter card.

To recover the SSL daughter card password, perform this task:

	Command	Purpose
Step 1	`CSM>` **`venus`**	Enables the Venus command line interfaces.
Step 2	`VENUS#` **`set_ssl_password_recovery 1`**	Causes the SSL daughter card to reboot into the password recovery mode. In this mode, the SSL daughter card does not require a password to enter the enable mode.
Step 3	`ssl-proxy#` **`enable`**	Enters enable mode on the module.
Step 4	`ssl-proxy#` **`copy nvram:startup-config running-config`**	Copies the startup configuration to the running configuration.
Step 5	`ssl-proxy#` **`configure terminal`**	Enters the configuration mode.
Step 6	`ssl-proxy(config)#` **`enable password`** *`password`*	Enables the password. **Note** The password is the new password that you want to have as your enable password for the SSL daughter card. If you do not want an enable password, you can enter the **no enable password** command instead.
Step 7	`ssl-proxy(config)#` **`line vty`** *`start_line end_line`*	Sets the virtual terminal starting and stopping console line numbers that you want to reset for the enable password.
Step 8	`ssl-proxy(config-line)#` **`login`**	Login.
Step 9	`ssl-proxy(config-line)#` **`password`** *`password`*	Enter the new enable password that you want to set for the virtual terminal.
Step 10	`ssl-proxy(config-line)#` **`end`**	Ends the session.

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

This example shows how to recover a lost password on the SSL daughter card that is inserted in slot 4 from the SSL daughter card Certificate Management (Cert. Mgt.) console port:

```
CSM> venus
VENUS# set_ssl_password_recovery 4
ssl-proxy# enable
ssl-proxy# copy nvram:startup-config running-config
ssl-proxy# configure terminal
ssl-proxy(config)# enable password cisco
```

Note Enter the **enable password cisco** command to set the password to **cisco**.

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

From the SSL daughter card console port, import the keys from the backup image or regenerate the keys.

See the "Configuring the Keys and the Certificates" section on page 8-2 for information on generating keys and importing keys.

Configuring VLANs

This chapter describes how to configure VLANs on the CSM-S and contains these sections:

To configure VLANs on the SSL daughter card, see the "Configuring VLANs on the SSL Daughter Card" section on page 7-2.

When you install the CSM-S in a Catalyst 6500 series switch, you need to configure the client-side and server-side VLANs. (See Figure 4-1.)

Client-side or a server-side VLAN terminology logically distinguishes the VLANs facing the client-side and the VLANs connecting to the servers or destination devices. However, the CSM-S client and server VLANs function very similarly. For example, new connections can be received on a server VLAN and then be load-balanced to a client VLAN.

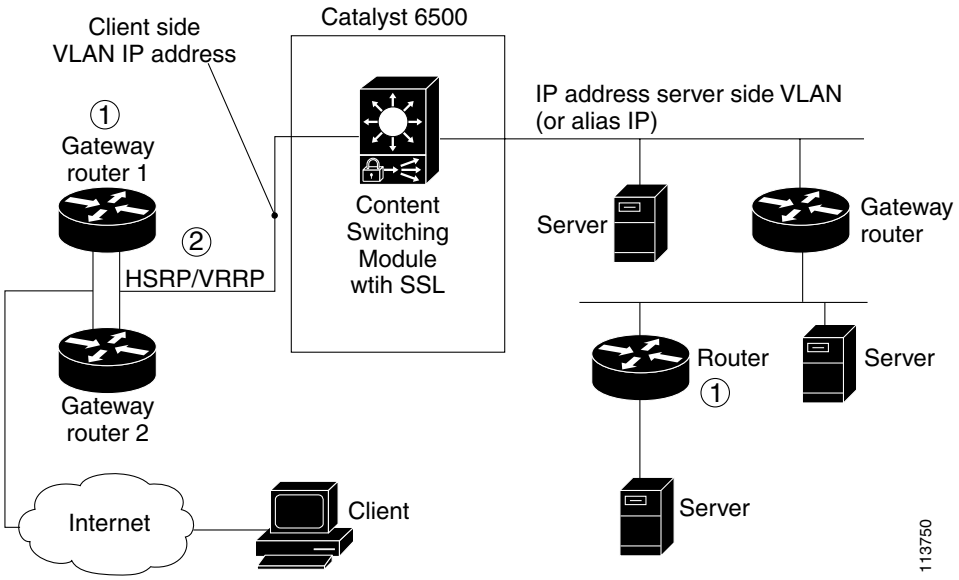
The differences between the client-side and server-side VLANs are as follows:

- When configuring bridge mode, you cannot bridge two server VLANs or two client VLANs. You can only bridge a client and a server VLAN.
- Denial of service (DoS) protection features are more aggressive on the client-side VLANs, especially when rate limiting control traffic is sent to the central processing unit.

Note You must configure VLANs on the Catalyst 6500 series switch before you configure VLANs for the CSM-S. The VLAN IDs must be the same for the switch and the module.

Note If you execute command **show module csm x conn**, the output shows an entry for VLAN 4095. You can ignore this VLAN, which the system creates for communication between the CSM and the SSL daughter card

Figure 4-1 Configuring VLANs



Note The numbers in Figure 4-1 correspond to the numbers in the following operation.

1. The CSM-S does not perform a Layer 3 lookup to forward traffic; the CSM-S cannot respond to ICMP redirects.
2. You can configure up to 7 gateways per VLAN for up to 511 client and server VLANs and up to 224 gateways for the entire system. If an HSRP gateway is configured, the CSM-S uses 3 of the 224 gateway entries because traffic can come from the virtual and physical MAC addresses of the HSRP group. (See the "Configuring HSRP" section on page 9-5.) The fault-tolerant VLAN does not use an IP interface, so it does not apply toward the 512 VLAN limit.

Configuring Client-Side VLANs

To configure the client-side VLANs, perform this task:

Caution You cannot use VLAN 1 as a client-side or server-side VLAN for the CSM-S.

	Command	Purpose
Step 1	`Router(config-module-csm)# vlan vlanid client`	Configures the client-side VLANs and enters the client VLAN mode[1].
Step 2	`Router(config-slb-vlan-client)# ip active_ip_addr [netmask] [alt standby_ip_addr [netmask]]`	Configures an IP address to the active CSM-S used by probes and ARP requests on this particular VLAN. When using redundant CSM-S modules, enter the **alt** keyword to specify an alternate IP address that is sent to the standby CSM-S.[2]
Step 3	`Router(config-slb-vlan-client)# description description`	(Optional) Specifies a description for the VLAN. Limit the *description* to 80 characters.
Step 4	`Router(config-slb-vlan-client)# gateway ip-address`	Configures the gateway IP address.

1. Enter the **exit** command to leave a mode or submode. Enter the **end** command to return to the menu's-top level.
2. The **no** form of this command restores the defaults.

This example shows how to configure the CSM-S for the client-side VLANs:

```
Router(config-module-csm)# vlan 130 client
Router(config-slb-vlan-client)# ip addr 123.44.50.6 255.255.255.0 alt 123.44.50.7
255.255.255.0
Router(config-slb-vlan-client)# gateway 123.44.50.1
Router(config-slb-vlan-client)# exit
Router# show module csm vlan 1
```

Configuring Server-Side VLANs

To configure the server-side VLANs, perform this task:

	Command	Purpose
Step 1	`Router(config-module-csm)# vlan vlanid server`	Configures the server-side VLANs and enters the server VLAN mode[1].
Step 2	`Router(config-slb-vlan-server)# ip active_ip_addr [netmask] [alt standby_ip_addr [netmask]]`	Configures an IP address for the server VLAN. When using redundant CSM-S modules, enter the **alt** keyword to specify an alternate IP address that is sent to the standby CSM-S[2].
Step 3	`Router(config-slb-vlan-server)# description description`	(Optional) Specifies a description for the VLAN. Limit the *description* to 80 characters.
Step 4	`Router(config-slb-vlan-server)# alias ip-address netmask`	(Optional) Configures multiple IP addresses to the CSM-S as alternate gateways for the real server[3].

	Command	Purpose
Step 5	`Router(config-slb-vlan-server)# route` *ip-address netmask* **gateway** *gw-ip-address*	Configures a static route to reach the real servers if they are more than one Layer 3 hop away from the CSM-S.
Step 6	`Router # show module csm` *slot* **vlan** [**client** | **server** | **ft**] [**id** *vlan-id*] [**detail**]	Displays the client-side and server-side VLAN configurations.

1. Enter the **exit** command to leave a mode or submode. Enter the **end** command to return to the menu's-top level.
2. The **no** form of this command restores the defaults.
3. The alias is required in the redundant configuration. See Chapter 9, "Configuring Redundancy."

This example shows how to configure the CSM-S for the server-side VLANs:

```
Router(config-module-csm)# vlan 150 server
Router(config-slb-vlan-server)# ip addr 123.46.50.6 255.255.255.0
Router(config-slb-vlan-server)# alias 123.60.7.6 255.255.255.0
Router(config-slb-vlan-server)# route 123.50.0.0 255.255.0.0 gateway 123.44.50.1
Router(config-slb-vlan-server)# exit
```

Configuring Real Servers and Server Farms

This chapter describes how to configure the servers and server farms and contains these sections:

Configuring Server Farms

A server farm or server pool is a collection of servers that contain the same content. You specify the server farm name when you configure the server farm and add servers to it, and when you bind the server farm to a virtual server. When you configure server farms, do the following:

- Name the server farm.
- Configure a load-balancing algorithm (predictor) and other attributes of the farm.
- Set or specify a set of real servers. (See the "Configuring Real Servers" section on page 5-3.)
- Set or specify the attributes of the real servers.

You also can configure inband health monitoring for each server farm. (See the "Understanding and Configuring Inband Health Monitoring" section on page 11-8.) You can assign a return code map to a server farm to configure return code parsing. (See the "Understanding and Configuring HTTP Return Code Checking" section on page 11-9.)

To configure server farms, perform this task:

	Command	Purpose
Step 1	`Router(config-module-csm)# serverfarm serverfarm-name`	Creates and names a server farm and enters the server farm configuration mode[1] [2].
Step 2	`Router(config-slb-sfarm)# predictor [roundrobin | leastconns | hash url | hash address [source | destination] [ip-netmask] | forward]]`	Configures the load-balancing prediction algorithm[2]. If not specified, the default is **roundrobin**. The **slowstart** *timer* command sets a timer for when the slow-start mechanism expires; valid values for *timer* are 1 to 65535 seconds. The **slowstart** *timer* is disabled by default.
Step 3	`Router(config-slb-sfarm)# nat client client-pool-name`	(Optional) Enables the NAT mode client[2]. (See the "Configuring Client NAT Pools" section on page 5-6.) **Note** If both the serverfarm and the policy are configured with client NAT, the policy takes precedence over the server farm.
Step 4	`Router(config-slb-sfarm)# no nat server`	(Optional) Specifies that the destination IP address is not changed when the load-balancing decision is made.
Step 5	`Router(config-slb-sfarm)# probe probe-name`	(Optional) Associates the server farm to a probe that can be defined by the **probe** command[2].
Step 6	`Router(config-slb-sfarm)# bindid bind-id`	(Optional) Binds a single physical server to multiple server farms and reports a different weight for each one[2]. The **bindid** command is used by DFP.
Step 7	`Router(config-slb-sfarm)# failaction {purge | reassign}`	(Optional) Sets the behavior of connections to real servers that have failed[2].
Step 8	`Router(config-slb-sfarm)# description description`	(Optional) Specifies a description for the server farm. Limit the *description* to 80 characters.
Step 9	`Router(config-slb-sfarm)# health retries 20 failed 600`	Configures inband health monitoring for all the servers in the server farm.
Step 10	`Cat6k-2(config-slb-sfarm)# retcode-map NAME_OF_MAP`	Configures HTTP return error code checking (requires the configuration of a map of type retcode).
Step 11	`Router(config-slb-sfarm)# real ip_address`	Defines a real server.
Step 12	`Router(config-slb-real)# inservice`	Enables the real servers.
Step 13	`Router# show module csm slot serverfarm serverfarm-name [detail]`	Displays information about one or all server farms.

1. Enter the **exit** command to leave a mode or submode. Enter the **end** command to return to the menu's top level.
2. The **no** form of this command restores the defaults.

When the least-connection predictor is configured, a slow-start mechanism is implemented to avoid sending a high rate of new connections to the servers that have just been put in service. The real server with the fewest number of active connections will get the next connection request for the server farm with the least-connection predictor.

An environment variable, REAL_SLOW_START_ENABLE, controls the rate at which a real server ramps up when it is put into service. The slow-start ramp up is only for a server farm configured with the least-connections method.

The configurable range for this variable is 0 to 10. The setting of 0 disables the slow-start feature. The value from 1 to 10 specifies how fast the newly activated server should ramp up. The value of 1 is the slowest ramp-up rate. The value of 10 specifies that the CSM would assign more requests to the newly activated server. The value of 3 is the default value.

If the configuration value is N, the CSM assigns 2 ^ N (2 raised to the N power) new requests to the newly active server from the start (assuming no connections were terminated at that time). As this server finishes or terminates more connections, a faster ramping occurs. The ramp up stops when the newly activated server has the same number of open connections as the other servers in a server farm or when the slowstart timer has expired.

This example shows how to configure a server farm, named p1_nat, using the least-connections (**leastconns**) algorithm.

```
Router(config-module-csm)# serverfarm pl_nat
Router(config-slb-sfarm)# predictor leastconns
Router(config-slb-sfarm)# real 10.1.0.105
Router(config-slb-real)# inservice
Router(config-slb-sfarm)# real 10.1.0.106
Router(config-slb-real)# inservice
```

Configuring Real Servers

Real servers are physical devices assigned to a server farm. Real servers provide the services that are load balanced. When the server receives a client request, it sends the reply to the CSM-S for forwarding to the client.

You configure the real server in the real server configuration mode by specifying the server IP address and port when you assign it to a server farm. You enter the real server configuration mode from the server farm mode where you are adding the real server.

A real server can be configured as follows:

- no inservice—The CSM-S is out of service. There are no sticky and no new connections being applied.

  Note If you specify no inservice, the CSM-S does not remove open connections. If you want to remove open connections, you must perform that task manually by using the **clear module csm** *slot* **connection** command.

- inservice—The CSM-S is in service. Sticky is allowed, and new connections to the module can be made.
- inservice standby—The CSM-S is in standby. Sticky is allowed. No new connections are allowed.

To configure real servers, perform this task:

	Command	Purpose
Step 1	`Router(config-slb-sfarm)# real ip-address [port][local]`	Identifies a real server as a member of the server farm and enters the real server configuration mode. An optional translation port can also be configured[1, 2]. An optional local keyword indicates that this server resides on the SSL daughter card.
Step 2	`Router(config-slb-real)# weight weighting-value`	(Optional) Sets the weighting value for the virtual server predictor algorithm to assign the server's workload capacity relative to the other servers in the server farm if the round-robin or least connection is selected[2]. **Note** The only time the sequence of servers starts over at the beginning (with the first server) is when there is a configuration or server state change (either a probe or DFP agent). When the least-connections predictor is configured, a slow-start mechanism is implemented to avoid sending a high rate of new connections to the servers that have just been put in service.
Step 3	`Router(config-slb-real)# maxconns max-conns`	(Optional) Sets the maximum number of active connections on the real server[2]. When the specified maximum is reached, no new connections are sent to that real server until the number of active connections drops below the minimum threshold.
Step 4	`Router(config-slb-real)# minconns min-conns`	(Optional) Sets the minimum connection threshold[2].
Step 5	`Router(config-slb-real)# inservice`	Enables the real server for use by the CSM[2, 3].
Step 6	`Router# show module csm slot [sfarm serverfarm-name] [detail]`	(Optional) Displays information about configured real servers. The **sfarm** option limits the display to real servers associated with a particular virtual server. The **detail** option displays detailed real server information.
Step 7	`Router# show module csm slot [vserver virtserver-name] [client ip-address] [detail]`	Displays active connections to the CSM-S. The **vserver** option limits the display to connections associated with a particular virtual server. The **client** option limits the display to connections for a particular client. The **detail** option displays detailed connection information.

1. Enter the **exit** command to leave a mode or submode. Enter the **end** command to return to the menu's top level.
2. The **no** form of this command restores the defaults.
3. Repeat Steps 1 through 5 for each real server that you are configuring.

This example shows how to create real servers:

```
Router(config-module-csm)# serverfarm serverfarm
Router(config-slb-sfarm)# real 10.8.0.7
Router(config-slb-real)# inservice
Router(config-slb-sfarm)# real 10.8.0.8
Router(config-slb-real)# inservice
Router(config-slb-sfarm)# real 10.8.0.9
Router(config-slb-real)# inservice
```

```
Router(config-slb-sfarm)# real 10.8.0.10
Router(config-slb-real)# inservice
Router(config-slb-sfarm)# real 10.1.0.105
Router(config-slb-real)# inservice
Router(config-slb-sfarm)# real 10.1.0.106
Router(config-slb-sfarm)# inservice
Router(config-slb-real)# end
Router# show mod csm slot reals detail
Router# show mod csm slot conns detail
```

The CSM-S performs a graceful server shutdown when a real server is taken out of service using the **no inservice** command. This command stops all new sessions from being load balanced to the real server while allowing existing sessions to complete or time out. New sessions are load balanced to other servers in the server farm for that virtual server.

Note If you specify no inservice, the CSM-S does not remove open connections. If you want to remove open connections, you must perform that task manually using the **clear module csm** *slot* **conn** command.

The standby state allows the fail action reassignment to reassign connections when a firewall fails. To configure the firewall connection reassignment, you have three options for graceful shutdown:

- Set up a fail action reassignment to a server farm.
- Assign a single real server as a backup for another real server in case of failure.
- The backup real server can be configured with inservice active or in the standby backup state. In standby, this real server would get new connections only when the primary real server failed.

This example shows how to remove a real server from service:

```
Router(config-slb-real)# no inservice
```

For more information on configuring server farms, see the "Configuring Server Farms" section on page 5-1.

The CSM-S also performs a graceful server shutdown when a real server fails a health probe and is taken out of service. For more information on configuring CSM-S health probes, see the "Configuring Probes for Health Monitoring" section on page 11-1.

If a client making a request is stuck to an out-of-service server (using a cookie, SSL ID, source IP, and so on), this connection is balanced to an in-service server in the farm. If you want to be stuck to an out-of-service server, enter the **inservice standby** command. When you enter the **inservice standby** command, no connections are sent to the standby real server with the exception of those connections that are stuck to that server and those servers with existing connections. After the specified standby time, you can use the **no inservice** command to allow only existing sessions to be sent to that real server. Sticky connections are then sent to an in-service real server in the server farm.

Configuring Dynamic Feedback Protocol

When you configure the Dynamic Feedback Protocol (DFP), the servers can provide feedback to the CSM-S to enhance load balancing. DFP allows host agents (residing on the physical server) to dynamically report the change in status of the host systems providing a virtual service.

Note A DFP agent may be on any host machine. A DFP agent is independent of the IP addresses and port numbers of the real servers that are managed by the agent. DFP manager is responsible for establishing the connections with DFP agents and receiving load vectors from DFP agents.

To configure DFP, perform this task:

	Command	Purpose
Step 1	`Router(config-module-csm)# dfp [password password]`	Configures DFP manager, supplies an optional password, and enters the DFP agent submode[1, 2].
Step 2	`Router(config-slb-dfp)# agent ip-address port [activity-timeout [retry-count [retry-interval]]]`	Configures the time intervals between keepalive messages, the number of consecutive connection attempts or invalid DFP reports, and the interval between connection attempts[2].
Step 3	`Router# show module csm slot dfp [agent [detail | ip-address port] | manager [ip_addr] | detail | weights]`	Displays DFP manager and agent information.

1. Enter the **exit** command to leave a mode or submode. Enter the **end** command to return to the menu's top level.
2. The **no** form of this command restores the defaults.

This example shows how to configure the dynamic feedback protocol:

```
Router(config-module-csm)# dfp password password
Router(config-slb-dfp)# agent 123.234.34.55 5 6 10 20
Router(config-slb-dfp)# exit
```

Configuring Client NAT Pools

When you configure client Network Address Translation (NAT) pools, NAT converts the source IP address of the client requests into an IP address on the server-side VLAN. Use the NAT pool name in the serverfarm submode of the **nat** command to specify which connections need to be configured for client NAT pools.

Note You can configure client NAT pools on the SSL daughter card. If you do so, there must be a matching client NAT pool on the CSM. If the matching client NAT pool is not configured, the client NAT will not work.

To configure client NAT pools, perform this task:

	Command	Purpose
Step 1	`Router(config-module-csm)# natpool pool-name start-ip end-ip netmask mask`	Configures a content-switching NAT. You must create at least one client address pool to use this command[1, 2].
Step 2	`Router(config-module-csm)# serverfarm serverfarm-name`	Enters the serverfarm submode to apply the client NAT.

	Command	Purpose
Step 3	`Router(config-slb-sfarm)# nat client clientpool-name`	Associates the configured NAT pool with the server farm.
Step 4	`Router# show module csm natpool [name pool-name] [detail]`	Displays the NAT configuration.

1. Enter the **exit** command to leave a mode or submode. Enter the **end** command to return to the menu's top level.
2. The **no** form of this command restores the defaults.

This example shows how to configure client NAT pools:

```
Router(config)# natpool pool1 102.36.445.2 102.36.16.8 netmask 255.255.255.0
Router(config)# serverfarm farm1
Router(config-slb-sfarm)# nat client pool1
```

Configuring Server-Initiated Connections

The NAT for the server allows you to support connections initiated by real servers and to provide a default configuration used for servers initiating connections that do not have matching entries in the server NAT configuration. By default, the CSM-S allows server-originated connections without NAT.

To configure NAT for the server, perform this task:

	Command	Purpose
Step 1	`Router(config)# static [drop | nat [ip-address | virtual]]`	Configures the server-originated connections. Options include dropping the connections, configuring them with NAT with a given IP address, or configuring them with the virtual IP address with which they are associated.[1, 2].
Step 2	`Router(config-slb-static)# real ip-address [subnet-mask]`	Configures the static NAT submode where the servers will have this NAT option. You cannot use the same real server with multiple NAT configuration options.

1. Enter the **exit** command to leave a mode or submode. Enter the **end** command to return to the menu's top level.
2. The **no** form of this command restores the defaults.

Configuring URL Hashing

When you choose a server farm for a connection, you can select a specific real server in that server farm. You can choose least connections, round robin, or URL hashing to select a real server.

URL hashing is a load-balancing predictor for Layer 7 connections. You can configure URL hashing on the CSM-S on a server farm-by-server farm basis. The CSM-S chooses the real server by using a hash value based on a URL. This hash value may be computed on the entire URL or on a portion of it. To select only a portion of the URL for hashing, you can specify the beginning and ending patterns in the URL so that only the portion of the URL from the specified beginning pattern through the specified ending pattern is hashed. The CSM-S supports URL hashing in software release 2.1(1).

Unless you specify a beginning and an ending pattern (see the "Configuring Beginning and Ending Patterns" section on page 5-9), the entire URL is hashed and used to select a real server.

Configuring a URL Hashing Predictor

You must configure URL hashing for all server farms that will be using the URL hashing predictor, regardless of whether they are using the entire URL or a beginning and ending pattern.

To configure URL hashing as a load-balancing predictor for a server farm, perform this task:

Command	Purpose
`Router(config-slb-sfarm)#` **`predictor hash url`**	Configures the URL hashing and load-balancing predictor for a server farm.

This example shows how to configure the URL hashing and load-balancing predictor for a server farm:

```
Router(config)# mod csm 2
Router(config-module-csm)# serverfarm farm1
Router(config-slb-sfarm)# predictor hash url
Router(config-slb-sfarm)# real 10.1.0.105
Router(config-slb-real)# inservice
Router(config-slb-real)# exit
```

Cache servers perform better using URL hashing. However, the hash methods do not recognize the weight for the real servers. The weight assigned to the real servers is used in the round-robin and least-connection predictor methods.

Note The only time the sequence of servers starts over at the beginning (with the first server) is when there is a configuration or server state change (either a probe or DFP agent).

To create different weights for real servers, you can list multiple IP addresses of the cache server in the server farm. You can also use the same IP address with a different port number.

Note Server weights are not used for hash predictors.

To configure real servers with a weight when using the URL hash predictor, perform this task:

	Command	Purpose
Step 1	`Router(config-slb-sfarm)#` **`serverfarm MYFARM`**	Creates a server farm named MYFARM.
Step 2	`Router(config-slb-sfarm)#` **`real 1.1.1.1 80`**	Specifies the real server at port 80.
Step 3	`Router(config-slb-sfarm)#` **`inservice`**	Enables the real server in service.
Step 4	`Router(config-slb-sfarm)#` **`real 1.1.1.1 8080`**	Specifies the real server at port 8080.
Step 5	`Router(config-slb-sfarm)#` **`inservice`**	Enables the real server in service.

Configuring Beginning and Ending Patterns

You configure a beginning and ending pattern at the virtual server level. The pattern that you define applies to all server farms assigned to all policies in that virtual server that have URL hashing enabled.

The beginning and ending pattern delimits the portion of the URL that will be hashed and used as a predictor to select a real server from a server farm that belongs to any policy assigned to that virtual server.

To hash a substring of the URL instead of the entire URL, specify the beginning and ending patterns in **vserver vserver-name** submode with the **url-hash begin-pattern** *pattern-a* command and **url-hash end-pattern** *pattern-b* command. Hashing occurs at the start of the beginning pattern and goes to the ending pattern.

For example, in the following URL, if the beginning pattern is **c&k=**, and the ending pattern is **&**, only the substring **c&k=c** is hashed:

http://quote.yahoo.com/q?s=csco&d=**c&k=c1**&t=2y&a=v&p=s&l=on&z=m&q=l\

Note Beginning and ending patterns are restricted to fixed constant strings. General regular expressions cannot be specified as patterns. If no beginning pattern is specified, hashing begins at the beginning of the URL. If no ending pattern is specified, hashing ends at the end of the URL.

This example shows how to configure beginning and ending patterns for URL hashing:

```
Router(config-module-csm)#
Router(config-module-csm)# vserver vs1
Router(config-slb-vserver)# virtual 10.1.0.81 tcp 80
Router(config-slb-vserver)# url-hash begin-pattern c&k= end-pattern &
Router(config-slb-vserver)# serverfarm farm1
Router(config-slb-vserver)# inservice
Router(config-slb-vserver)#
Router(config-slb-vserver)# exit
Router(config-module-csm)# exit
```

Configuring Virtual Servers, Maps, and Policies

This chapter describes how to configure content switching and contains these sections:

Configuring Virtual Servers

This section describes how to configure virtual servers and contains these sections:

Note When a virtual server is configured with an IP address, it starts replying to ARP requests for that specific IP, even if it is still out of service. This feature is important when migrating operational virtual servers from existing devices over to the CSM-S. Make sure that a virtual server on the CSM-S is never configured with the same IP of another device in the same network.

Virtual servers represent groups of real servers and are associated with real server farms through policies. Configuring virtual servers requires that you set the attributes of the virtual server by specifying the default server farm (default policy) and that you associate other server farms through a list of policies. The default server farm (default policy) is used if a request does not match any SLB policy or if there are no policies associated with the virtual server.

Before you can associate a server farm with the virtual server, you must configure the server farm. For more information, see the "Configuring Server Farms" section on page 5-1. Policies are processed in the order in which they are entered in the virtual server configuration. For more information, see the "Configuring Policies" section on page 6-10.

You can configure each virtual server with a pending connection timeout to terminate connections quickly if the switch becomes flooded with traffic. This connection applies to a transaction between the client and server that has not completed the request and reply process.

In a service provider environment in which different customers are assigned different virtual servers, you may need to balance the connections to prevent an individual server from absorbing most or even all of the connection resources on the CSM-S. You can limit the number of connections going through the CSM-S to a particular virtual server by using the VIP connection watermarks feature. With this feature, you may set limits on each virtual server, allowing a fair distribution of connection resources among all virtual servers.

Note You can configure a single virtual server to operate at either Level 4 or Level 7. To configure a virtual server to operate at Level 4, specify the server farm (default policy) as part of the virtual server configuration. (See Step 3 in the following task table.) To configure a virtual server to operate at Level 7, add SLB policies in the configuration of the virtual server. (See Step 7 in the following task table.)

The CSM-S can load balance traffic from any IP protocol. When you configure a virtual server in virtual server submode, you must define the IP protocol that the virtual server will accept.

Note Although all IP protocols have a protocol number, the CSM-S allows you to specify TCP or UDP by name instead of requiring you to enter their numbers.

Configure the virtual server in the virtual server configuration submode.

To configure virtual servers, perform this task:

	Command	Purpose
Step 1	`Router(config-module-csm)# owner` *owner-name* **address** *street-address-information* **billing-info** *billing-address-information* **email-address** *email-information* **maxconns** *1:MAXULONG*	Restricts access to virtual servers to a specific owner object.
Step 2	`Router(config-module-csm)# vserver` *virtserver-name*	Identifies the virtual server and enters the virtual server configuration mode[1, 2].
Step 3	`Router(config-slb-vserver)# vs-owner` *owner-name* **maxconns** *1:MAXULONG*	Sets the owner object name for this virtual server.
Step 4	`Router(config-slb-vserver)# virtual` *ip-address [ip-mask] protocol port-number* [**service ftp**]	Sets the IP address for the virtual server optional port number or name and the connection coupling and type[2]. The *protocol* value is **tcp**, **udp**, **Any** (no port number is required), or a *number* value (no port number is required).
Step 5	`Router(config-slb-vserver)# serverfarm` *serverfarm-name*	Associates the default server farm with the virtual server[2 3]. Only one server farm is allowed. If the server farm is not specified, all the requests not matching any other policies will be discarded.
Step 6	`Router(config-slb-vserver)# sticky` *duration*	(Optional) Configures connections from the client to use the same real server[2, 3]. The default is sticky off.
Step 7	`Router(config-slb-vserver)# sticky` *group-number* **reverse**	(Optional) Ensures that the CSM-S changes connections in the appropriate direction back to the same source.
Step 8	`Router(config-slb-vserver)# client` *ip-address network-mask* [**exclude**]	(Optional) Restricts which clients are allowed to use the virtual server[2, 3].

	Command	Purpose
Step 9	`Router(config-slb-vserver)# slb-policy` *policy-name*	(Optional) Associates one or more content switching policies with a virtual server[2].
Step 10	`Router(config-slb-vserver)# inservice`	Enables the virtual server for use by the CSM[2].
Step 11	`Router# show module csm` *slot* `vserver [details]`	Displays information for virtual servers that are defined for content switching.

1. Enter the **exit** command to leave a mode or submode. Enter the **end** command to return to the menu's top level.
2. The **no** form of this command restores the defaults.
3. These parameters refer to the default policy.

This example shows how to configure a virtual server named barnett, associate it with the server farm named bosco, and configure a sticky connection with a duration of 50 minutes to sticky group 12:

```
Router(config)# mod csm 2
Router(config-module-csm)# sticky 1 cookie foo timeout 100
Router(config-module-csm)# exit
Router(config-module-csm)#
Router(config-module-csm)# serverfarm bosco
Router(config-slb-sfarm)# real 10.1.0.105
Router(config-slb-real)# inservice
Router(config-slb-real)# exit
Router(config-slb-sfarm)#
Router(config-slb-sfarm)# vserver barnett
Router(config-slb-vserver)# virtual 10.1.0.85 tcp 80
Router(config-slb-vserver)# serverfarm bosco
Router(config-slb-vserver)# sticky 50 group 12
Router(config-slb-vserver)# inservice
Router(config-slb-vserver)# exit
Router(config-module-csm)# end
```

This example shows how to configure a virtual server, named vs1, with two policies and a default server farm when client traffic matches a specific policy. The virtual server will be load balanced to the server farm attached to that policy. When client traffic fails to match any policy, the virtual server will be load balanced to the default server farm named bosco.

```
Router(config)# mod csm 2
Router(config-module-csm)# map map3 url
Router(config-slb-map-url)# match protocol http url *finance*
Router(config-slb-map-url)#
Router(config-slb-map-url)# map map4 url
Router(config-slb-map-url)# match protocol http url *mail*
Router(config-slb-map-url)#
Router(config-slb-map-url)# serverfarm bar1
Router(config-slb-sfarm)# real 10.1.0.105
Router(config-slb-real)# inservice
Router(config-slb-real)#
Router(config-slb-real)# serverfarm bar2
Router(config-slb-sfarm)# real 10.1.0.106
Router(config-slb-real)# inservice
Router(config-slb-real)#
Router(config-slb-real)# serverfarm bosco
Router(config-slb-sfarm)# real 10.1.0.107
Router(config-slb-real)# inservice
Router(config-slb-real)#
Router(config-slb-real)# policy pc1
Router(config-slb-policy)# serverfarm bar1
Router(config-slb-policy)# url-map map3
Router(config-slb-policy)# exit
```

```
Router(config-module-csm)#
Router(config-module-csm)# policy pc2
Router(config-slb-policy)# serverfarm bar2
Router(config-slb-policy)# url-map map4
Router(config-slb-policy)# exit
Router(config-module-csm)#
Router(config-module-csm)# vserver bar1
Router(config-slb-vserver)# virtual 10.1.0.86 tcp 80
Router(config-slb-vserver)# slb-policy pc1
Router(config-slb-vserver)# slb-policy pc2
Router(config-slb-vserver)# serverfarm bosco
Router(config-slb-vserver)# inservice
Router(config-slb-vserver)#
```

Configuring TCP Parameters

Transmission Control Protocol (TCP) is a connection-oriented protocol that uses known protocol messages for activating and deactivating TCP sessions. In server load balancing, when adding or removing a connection from the connection database, the Finite State Machine correlates TCP signals such as SYN, SYN/ACK, FIN, and RST. When adding connections, these signals are used for detecting server failure and recovery and for determining the number of connections per server.

The CSM-S also supports User Datagram Protocol (UDP). Because UDP is not connection-oriented, protocol messages cannot be generically sniffed (without knowing details of the upper-layer protocol) to detect the beginning or end of a UDP message exchange. Detection of UDP connection termination is based on a configurable idle timer. Protocols requiring multiple simultaneous connections to the same real server are supported (such as FTP). Internet Control Management Protocol (ICMP) messages destined for the virtual IP address are also handled (such as ping).

To configure TCP parameters, perform this task:

	Command	Purpose
Step 1	`Router(config-module-csm)# vserver` *virtserver-name*	Identifies the virtual server and enters the virtual server configuration mode[1, 2].
Step 2	`Router(config-slb-vserver)# idle` *duration*	Configures the amount of time (in seconds) that connection information is maintained in the absence of packet activity for a connection[2]. Valid values for *duration* are 0 (connections remain open indefinitely) through 65535 seconds; the default value is 3600. **Note** If you specify **idle 0**, connections are created but are never automatically removed from the connection table, which can potentially consume all resources until you remove the connections. Use the **INFINITE_IDLE_TIME_MAXCONNS** environmental variable to specify the maximum number of connections.

1. Enter the **exit** command to leave a mode or submode. To return to the Router (config)> top level of the menu, enter the **end** command.
2. The **no** form of this command restores the defaults.

This example shows how to configure TCP parameters for virtual servers:

```
Router(config-module-csm)# vserver barnett
Router(config-slb-vserver)# idle 10
```

The CSM-S provides support for fragmented TCP packets. The TCP fragment feature only works with VIPs that have Level 4 policies defined and will not work for SYN packets or for Layer 7 policies. To support fragmented TCP packets, the CSM-S matches the TCP fragments to existing data flows or by matching the bridging VLAN ID. The CSM-S will not reassemble fragments for Layer 7 parsing. Because the CSM-S has a finite number of buffers and fragment ID buckets, packet resending is required when there are hash collisions.

When enabling TCP splicing, you must designate a virtual server as a Layer 7 device even when it does not have a Layer 7 policy. This option is only valid for the TCP protocol.

To configure TCP splicing, perform this task:

Configuring Partial Serverfarm Failover

When you configure a backup server farm, you can define two threshold values that specify how many active real servers are required for a server farm to remain healthy and how many active real servers are required for the server farm to be reactivated.

If you do not specify these threshold values, a server farm will fail when all real servers in the server farm fail. In this case, the primary server farm becomes operational again when one real server in the server farm becomes healthy.

	Command	Purpose
Step 1	`Router(config-module-csm)# vserver` *virtserver-name*	Identifies the virtual server and enters the virtual server configuration mode.
Step 2	`Router(config-slb-vserver)#` **serverfarm** *primary_serverfarm* [**backup** *backup_serverfarm* [**threshold outservice** *real_value* **inservice** *real_value*][**sticky**]]	Associates the default server farm with the virtual server and defines the backup server farm. The **outservice** value specifies the minimum number of active real servers required for the server farm to remain healthy. The **inservice** value specifies the number of active real servers required for the server farm to be reactivated. The valid range for both values is 1 up to the maximum number of real servers supported; the **outservice** value must be lower than the **inservice** value.

The following example shows how to configure the backup server farm to become active when there are less than three healthy real servers in the server farm, and it shows how the primary server farm becomes active again when there are six healthy real servers in the server farm:

```
Router(config-slb-sfarm)# vserver barnett
Router(config-slb-vserver)# serverfarm bosco backup BACKUP threshold outservice 3
inservice 6
```

Configuring Virtual Server Dependency

You can configure the CSM-S to track a virtual server. This feature provides the ability to automatically take the dependent virtual server out of service if the tracked virtual server is taken out of service or fails.

To configure virtual server dependency, perform this task:

	Command	Purpose
Step 1	`Router(config-module-csm)# vserver dependent_virtserver_name`	Identifies the dependant virtual server and enters the virtual server configuration mode.
Step 2	`Router(config-slb-vserver)# virtual ip-address [ip-mask] protocol port-number [service {ftp | rtsp | termination}]`	Sets the IP address for the dependant virtual server optional port number or name and the connection coupling and type[2]. The *protocol* value is **tcp**, **udp**, **any** (no port number is required), or a *number* value (no port number is required).
Step 3	`Router(config-slb-vserver)# status-tracking tracked_virtserver_name`	Identifies the tracked virtual server. When this virtual server is taken out of service or fails, the dependent virtual server identified in Step 1 is automatically taken out of service.

This example shows how to configure virtual servers A and C to be automatically taken out of service when virtual server B is taken out of service or fails:

```
Router(config-slb-sfarm)# vserver A
Router(config-slb-vserver)# virtual 10.1.0.85 tcp 80
Router(config-slb-vserver)# status-tracking B
Router(config-slb-vserver)# exit
Router(config-slb-sfarm)# vserver C
Router(config-slb-vserver)# virtual 10.1.0.86 tcp 80
Router(config-slb-vserver)# status-tracking B
```

Configuring Redirect Virtual Servers

	Command	Purpose
Step 1	`Router(config-module-csm)# vserver virtserver-name`	Identifies the virtual server and enters the virtual server configuration mode[1, 2].
Step 2	`Router(config-slb-vserver)# vserver tcp-protect`	Designates the virtual server for TCP splicing[2].
Step 3	`Router(config-slb-vserver)# virtual 100.100.100.100 tcp any service tcp-termination`	Enables TCP splicing.

1. Enter the **exit** command to leave a mode or submode. To return to the Router (config)> top level of the menu, enter the **end** command.
2. The **no** form of this command restores the defaults.

The **redirect-vserver** command is a server farm submode command that allows you to configure virtual servers dedicated to real servers. This mapping provides connection persistence, which maintains connections from clients to real servers across TCP sessions.

To configure redirect virtual servers, perform this task:

	Command	Purpose
Step 1	`Router(config-slb-sfarm)#` **`redirect-vserver`** *`name`*	Configures virtual servers dedicated to real servers and enters the redirect server submode[1, 2].
Step 2	`Router(config-slb-redirect-v)#` **`webhost relocation`** *`relocation string`*	Configures the destination URL host name when redirecting HTTP requests that arrive at this server farm. Only the beginning of the URL can be specified in the relocation string. The remaining portion is taken from the original HTTP request[2].
Step 3	`Router(config-redirect-v)#` **`webhost backup`** *`backup string`*	Configures the relocation string that is sent in response to HTTP requests in the event that the redirect server is out of service. Only the beginning of the relocation string can be specified. The remaining portion is taken from the original HTTP request[2].
Step 4	`Router(config-redirect-v)#` **`virtual`** *`v_ipaddress`* **`tcp`** *`port`*	Configures the redirect virtual server IP address and port[2].
Step 5	`Router(config-redirect-v)#` **`idle`** *`duration`*	Sets the CSM-S connection idle timer for the redirect virtual server[2].
Step 6	`Router(config-redirect-v)#` **`client`** *`ip-address network-mask`* [**`exclude`**]	Configures the combination of the IP address and network mask used to restrict those clients are allowed to access the redirect virtual server[2].
Step 7	`Router(config-redirect-v)#` **`inservice`**	Enables the redirect virtual server and begins advertisements[2].
Step 8	`Router(config-redirect-v)#` **`ssl`** *`port`*	(Optional) Enables SSL forwarding by the virtual server.
Step 9	`Router#` **`show module csm vserver redirect`** [**`detail`**]	Shows all redirect servers configured.

1. Enter the **exit** command to leave a mode or submode. Enter the **end** command to return to the menu's top level.
2. The **no** form of this command restores the defaults.

This example shows how to configure redirect virtual servers to specify virtual servers to real servers in a server farm:

```
Router (config)# serverfarm FARM1
Router (config-slb-sfarm)# redirect-vserver REDIR_1
Router (config-slb-redirect-)# webhost relocation 127.1.2.30 301
Router (config-slb-redirect-)# virtual 172.1.2.30 tcp www
Router (config-slb-redirect-)# inservice
Router (config-slb-redirect-)# exit
Router (config-slb-sfarm)# redirect-vserver REDIR_2
Router (config-slb-redirect-)# webhost relocation 127.1.2.31 301
Router (config-slb-redirect-)# virtual 172.1.2.31 tcp www
Router (config-slb-redirect-)# inservice
Router (config-slb-redirect-)# exit
Router (config-slb-sfarm)# real 10.8.0.8
Router (config-slb-real)# redirect-vserver REDIR_1
Router (config-slb-real)# inservice
```

```
Router (config-slb-sfarm)# real 10.8.0.9
Router (config-slb-real)# redirect-vserver REDIR_2
Router (config-slb-real)# inservice
Router (config-slb-real)# end
Router# show module csm serverfarm detail
```

Configuring Maps

You configure maps to define multiple URLs, cookies, HTTP headers, and return codes into groups that can be associated with a policy when you configure the policy. (See the "Configuring Policies" section on page 6-10.) Regular expressions for URLs (for example, *url1* and *url2*) are based on UNIX filename specifications. See Table 6-1 for more information.

To add a URL map, perform this task:

	Command	Purpose
Step 1	`Router(config-module-csm)#` **`map`** *`url-map-name`* **`url`**	Creates a group to hold multiple URL match criteria[1, 2].
Step 2	`Router(config-slb-map-url)#` **`match protocol http url`** *`url-path`*	Specifies a string expression to match against the requested URL[2].

1. Enter the **exit** command to leave a mode or submode. Enter the **end** command to return to the menu's top level.
2. The **no** form of this command restores the defaults.

Table 6-1 Special Characters for Matching String Expressions

Convention	Description
*	Zero or more characters.
?	Exactly one character. **Note** You must precede the question mark with a **Ctrl-V** command to prevent the CLI Parser from interpreting it as a help request
\	Escaped character.
Bracketed range [0-9]	Matching any single character from the range.
A leading ^ in a range	Do not match any in the range. All other characters represent themselves.
.\a	Alert (ASCII 7).
.\b	Backspace (ASCII 8).
.\f	Form-feed (ASCII 12).
.\n	New line (ascii 10).
.\r	Carriage return (ASCII 13).
.\t	Tab (ASCII 9).
.\v	Vertical tab (ASCII 11).
.\0	Null (ASCII 0).

Table 6-1 Special Characters for Matching String Expressions (continued)

Convention	Description
.\\	Backslash.
.\x##	Any ASCII character as specified in two-digit hex notation.

To add a cookie map, perform this task:

	Command	Purpose
Step 1	`Router(config)# map` *cookie-map-name* `cookie`	Configures multiple cookies into a cookie map[1].
Step 2	`Router(config-slb-map-cookie)# match protocol http cookie` *cookie-name* `cookie-value` *cookie-value-expression*	Configures multiple cookies[1].

1. The **no** form of this command restores the defaults.

This example shows how to configure maps and associate them with a policy:

```
Router(config-module-csm)# serverfarm pl_url_url_1
Router(config-slb-sfarm)# real 10.8.0.26
Router(config-slb-real)# inservice
Router(config-slb-real)# exit
Router(config-slb-sfarm)# exit
Router(config-slb-policy)# serverfarm pl_url_url_1
Router(config-slb-policy)# url-map url_1
Router(config-slb-policy)# exit
Router(config-module-csm)# serverfarm pl_url_url_2
Router(config-slb-sfarm)# real 10.8.0.27
Router(config-slb-real)# inservice
Router(config-slb-real)# exit
Router(config-slb-sfarm)# exit
Router(config-module-csm)# map url_1  url
Router(config-slb-map-url)# match protocol http url /url1
Router(config-slb-map-url)# exit
Router(config-module-csm)# map url_2 url
Router(config-slb-map-url)# match protocol http url /url/url/url
Router(config-slb-map-url)# match protocol http url /reg/*long.*
Router(config-slb-map-url)# exit
Router(config-module-csm)# policy policy_url_1
Router(config-module-csm)# policy policy_url_2
Router(config-slb-policy)# serverfarm pl_url_url_2
Router(config-slb-policy)# url-map url_2
Router(config-slb-policy)# exit
Router(config-module-csm)# vserver vs_url_url
Router(config-slb-vserver)# virtual 10.8.0.145 tcp 80
Router(config-slb-vserver)# slb-policy policy_url_1
Router(config-slb-vserver)# slb-policy policy_url_2
Router(config-slb-vserver)# inservice
Router(config-slb-vserver)# exit
```

Using the **map** command, you create a map group with the type HTTP header. When you enter the **map** command, you are placed in a submode where you can specify the header fields and values for the CSM-S to search for in the request.

To create a map for the HTTP header, perform this task:

Command	Purpose
`Router(config-module-csm)# map` *name* **header**	Creates and names an HTTP header map group.

For more information about header maps, see the "Configuring Generic Header Parsing" section on page 6-12.

To create a map for return code checking, perform this task:

Command	Purpose
`Router(config-module-csm)# map` *name* **retcode**	Creates and names a return code map group.

To configure HTTP return error code checking, perform this task:

Command	Purpose
`Router(config-slb-sfarm)# retcode-map` *name_of_map*	Configures HTTP return error code checking.

For more information about return code maps, see the "Understanding and Configuring HTTP Return Code Checking" section on page 11-9.

Configuring Policies

Policies are access rules that traffic must match when balancing to a server farm. Policies allow the CSM-S to balance Layer 7 traffic. Multiple policies can be assigned to one virtual server, creating multiple access rules for that virtual server. When configuring policies, you first configure the access rules (maps, client-groups, and sticky groups) and then you combine these access rules under a particular policy.

Note You must associate a server farm with a policy. A policy that does not have an associated server farm cannot forward traffic. The server farm associated with a policy receives all the requests that match that policy.

When the CSM-S is able to match policies, it selects the policy that appears first in the policy list. Policies are located in the policy list in the sequence in which they were bound to the virtual server.

A policy can be matched even if all the servers in the associated server farm are down. The default behavior of the policy in that case is not to accept those connections and send back a reset (RST) to the clients. To change this behavior, add a backup server farm for that policy.

When you add the **backup** *sorry-serverfarm* [**sticky**] option to the backup server farm, this option defines whether the sticky group applied to the primary server farm is also applied for the backup server farm. If you do not specify stickiness for the primary server farm, then stickiness is not applied to the backup server farm.

For example, if you have a sticky group configured for a policy, the primary server farm in this policy becomes sticky. The client will be stuck to the configured real server in the primary server farm. When all of the real servers in the primary server farm fail, new requests from this client are sent to the backup server farm. When the real server in the primary server farm comes back to the operational state, the following actions result:

- The existing connections to the backup real server continue to be serviced by the backup real server.
- The new requests from the client are sent to the backup real server if the sticky option is enabled for the backup server farm.
- The new requests go back to the primary real server if the sticky option is not used on the backup server farm.

You can reorder the policies in the list by removing policies and reentering them in the correct order. To remove and enter policies, enter the **no slb-policy** *policy name* command and the **slb-policy** *policy name* command in the virtual server submode.

To configure load-balancing policies, perform this task:

	Command	Purpose
Step 1	`Router(config-module-csm)# policy policy-name`	Creates the policy and enters the policy submode to configure the policy attributes[1].
Step 2	`Router(config-slb-policy)# url-map url-map-name`	Associates a URL map to a policy[2]. You must have previously created and configured the URL maps and cookie maps with the **map** command. See the "Configuring Generic Header Parsing" section on page 6-12.
Step 3	`Router(config-slb-policy)# cookie-map cookie-map-name`	Associates a cookie map to a policy[2].
Step 4	`Router(config-slb-policy)# header-map name`	Associates an HTTP header map to a policy.
Step 5	`Router(config-slb-policy)# sticky-group group-id`	Associates this policy to a specific sticky group[2].
Step 6	`Router(config-slb-policy)# client-group value | std-access-list-name`	Configures a client filter associated with a policy. Only standard IP access lists are used to define a client filter.
Step 7	`Router(config-slb-policy)# serverfarm serverfarm-name`	Configures the server farm serving a particular load-balancing policy. Only one server farm can be configured per policy[2].
Step 8	`Router(config-slb-policy)# set ip dscp dscp-value`	Marks traffic with a DSCP value if packets matched with the load-balancing policy[2].
Step 9	`Router(config-slb-policy)# nat client {client-pool-name | static}`	(Optional) Enables the NAT mode client[2]. **Note** If both the serverfarm and the policy are configured with client NAT, the policy takes precedence over the server farm.

1. Enter the **exit** command to leave a mode or submode. Enter the **end** command to return to the menu's top level.
2. The **no** form of this command restores the defaults.

This example assumes that the URL map, map1, has already been configured and shows how to configure server load-balancing policies and associate them to virtual servers:

```
Router(config-slb-policy)# serverfarm pl_sticky
```

```
Router(config-slb-sfarm)# real 10.1.0.105
Router(config-slb-sfarm)# inservice
Router(config-slb-policy)# exit
Router(config-module-csm)# policy policy_sticky_ck
Router(config-slb-policy)# serverfarm pl_sticky
Router(config-slb-policy)# url-map map1
Router(config-slb-policy)# exit
Router(config-module-csm)# vserver vs_sticky_ck
Router(config-slb-vserver)# virtual 10.1.0.80 tcp 80
Router(config-slb-vserver)# slb-policy policy_sticky_ck
Router(config-slb-sfarm)# inservice
Router(config-slb-policy)# exit
```

Configuring Generic Header Parsing

In software release 2.1(1), the CSM-S supports generic HTTP request header parsing. The HTTP request header contains fields that describe how content should be formatted to meet the user's requirements.

Understanding Generic Header Parsing

The CSM-S uses the information it learns by parsing and matching fields in the HTTP header along with policy information to make load-balancing decisions. For example, by parsing the browser-type field in the HTTP header, the CSM-S can determine if a user is accessing the content with a mobile browser and can select a server that contains content formatted for a mobile browser.

An example of a HTTP Get request header record is as follows:

```
GET /?u HTTP/1.1<0D><0A>
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg<0D><0A>
Referer: http://www.yahoo.com/<0D><0A>
Accept-Language: en-us<0D><0A>
Accept-Encoding: gzip, deflate<0D><0A>
User-Agent: Mozilla/4.0 (compatible; MSIE 5.0; Windows NT; DigExt)<0D><0A>
Host: finance.yahoo.com<0D><0A>
Connection: Keep-Alive<0D><0A>
Cookie: B=51g3cjstaq3vm; Y=1<0D><0A>
<0D><0A>
```

Generic Header Parsing Configuration

You configure generic header parsing by entering commands that instruct the CSM-S to perform policy matching on fields in the HTTP header. These sections describe how to configure generic header parsing on the CSM-S:

Creating a Map for the HTTP Header

Using the **map** command, you create a map group with the type HTTP header. When you enter the **map** command, you are placed in a submode where you can specify the header fields and values for the CSM-S to search for in the request.

To create a map for the HTTP header, perform this task:

Command	Purpose
`Router(config-module-csm)# map` *name* `header`	Creates and names a HTTP header map group.

Note Other map types include a URL and a cookie.

Specifying Header Fields and Match Values

You can specify the name of the field and the corresponding value for the CSM-S to match when receiving an HTTP request by using the **match** command.

To specify head fields and match values, perform this task:

Command	Purpose
`Router(config-slb-map-header)# match protocol http header` *field* `header-value` *expression*	Specifies the name of the field and value. The field can be any HTTP header except cookie. You can configure a cookie map if you want to configure a cookie header.

Note The CSM-S allows you to specify one or more fields in the HTTP header to be the criteria for policy matching. When multiple fields are configured in a single HTTP header group, all of the expressions in this group must match to satisfy this criteria.

Assigning an HTTP Header Map to a Policy

In policy submode, you specify the header map to include in that policy. The header map contains the HTTP header criteria to be included in a policy.

To assign an HTTP header map to a policy, perform this task:

	Command	Purpose
Step 1	`Router(config-module-csm)# policy` policy-*name*	Creates a policy.
Step 2	`Router(config-slb-policy)# header-map` *name*	Assigns an HTTP header map to a policy.

Note By default, a policy rule can be satisfied with any HTTP header information. The HTTP URL and HTTP cookie are specific types of header information and are handled separately by the CSM-S.

Assigning the Policy to a Virtual Server

In virtual server submode, specify the name of the policy that has the header map assigned by using the **vserver** *virtserver-name* command.

To specify a policy with a header map assigned, perform this task:

	Command	Purpose
Step 1	`Router(config-module-csm)#` **vserver** virtserver-*name*	Configures a virtual server.
Step 2	`Router(config-slb-policy)#` **header-map** *name*	Assigns an HTTP header map to a policy.

Generic Header Parsing Example

This example shows how to configure generic header parsing:

```
Router(config)# mod csm 2
Router(config-module-csm)# !!!configure generic header map
Router(config-module-csm)# map map2 header
Router(config-slb-map-heaer)# $col http header Host header-value *.yahoo.com

Router(config-slb-map-header)# !!! configure serverfarm
Router(config-slb-map-header)# serverfarm farm2
Router(config-slb-sfarm)# real 10.1.0.105
Router(config-slb-real)# inservice
Router(config-slb-real)# exit
Router(config-slb-sfarm)# exit

Router(config-module-csm)# !!! configurate policy
Router(config-module-csm)# policy pc2
Router(config-slb-policy)# serverfarm farm2
Router(config-slb-policy)# header-map map2
Router(config-slb-policy)# exit



Router(config-module-csm)# !!! config vserver
Router(config-module-csm)# vserver vs2
Router(config-slb-vserver)# virtual 10.1.0.82 tcp 80
Router(config-slb-vserver)# slb-policy pc2
Router(config-slb-vserver)# inservice
Router(config-slb-vserver)# end
Router(config)# show module csm 2 map det
```

Configuring the CSM-S SSL Services

This chapter describes the Command Line Interface (CLI) commands to configure, monitor, and debug the CSM-S software for SSL. These configuration commands are the same commands that are valid in the SSL Services Module.

Note Except where specifically differentiated, the term *Content Switching Module* and its acronym *CSM* refer to both the Content Switching Module and the Content Switching Module with SSL. The term *Content Switching Module with SSL* and its acronym *CSM-S* are used to refer to the CSM-S only.

This chapter describes configuration additions made to the CSM-S to support the SSL daughter card and contains these sections:

Note You must create a separate server farm for all back-end servers. Although the CSM-S is not associated with a virtual server, the CSM-S performs address resolution on each real server. A sample configuration is shown in the "Configuring the Real Server as the Back-End Server" section on page 1-18.

Initial SSL Daughter Card Configuration

This section describes how to make the initial configurations for the SSL daughter card.

Note You must make the following initial SSL daughter card configurations through a direct connection to the CSM-S Certificate Management Port connector. (See Figure 1-2 on page 1-8.) After the initial configurations, you can make an SSH or Telnet connection to the module in order to make further configurations for the module.

The initial SSL daughter card configuration consists of these tasks:

Configuring VLANs on the SSL Daughter Card

When you configure VLANs on the SSL daughter card, configure one of the VLANs as an administrative VLAN. The administrative VLAN is used for all management traffic, including SSH, public key infrastructure (PKI), secure file transfer (SCP), and TFTP operations. The system adds the default route through the gateway of the administrative VLAN.

Note Configure only one VLAN on the SSL daughter card as the administrative VLAN.

Note All VLANs configured on the SSL daughter card must also be configured on the CSM. All VLANs must match for the CSM virtual servers and the SSL real servers.

Note The VLAN IDs for the switch and the module must be identical. 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). You must limit the SSL daughter card configuration to the normal-range VLANs. Note that VLAN 4095 is automatically created for system communication between the CSM and the SSL daughter card. You can ignore this VLAN.

To configure VLANs on the SSL daughter card, perform this task:

	Command	Purpose
Step 1	`ssl-proxy(config)# ssl-proxy vlan` *vlan*	Configures the VLANs and enters VLAN mode.
Step 2	`ssl-proxy(config-vlan)# ipaddr` *ip_addr netmask*	Configures an IP address for the VLAN.

	Command	Purpose
Step 3	`ssl-proxy(config-vlan)# gateway gateway_addr`	Configures the client-side gateway IP address. **Note** Configure the gateway IP address in the same subnet as the VLAN IP address.
Step 4	`ssl-proxy(config-vlan)# route ip_addr netmask gateway ip_addr`	(Optional) Configures a static route for servers that are one or more Layer 3 hops away from the CSM-S.
Step 5	`ssl-proxy(config-vlan)# admin`	(Optional) Configures the VLAN as the administrative VLAN[1].

1. The administrative VLAN is for management traffic (PKI, SSH, SCP and TFTP). Specify only one VLAN as the administrative VLAN.

This example shows how to configure the VLAN and specify the IP address, the subnet mask, and the global gateway, and how to specify the VLAN as the administrative VLAN:

```
ssl-proxy(config)# ssl-proxy vlan 100
ssl-proxy(config-vlan)# ipaddr 10.1.0.20 255.255.255.0
ssl-proxy(config-vlan)# gateway 10.1.0.1
ssl-proxy(config-vlan)# admin
ssl-proxy(config-vlan)# ^Z
ssl-proxy#
```

Configuring Telnet Remote Access

To configure the SSL daughter card 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 daughter card 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#
```

Configuring the Fully Qualified Domain Name

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

To configure the FQDN, perform this task:

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

This example shows how to configure the FQDN on the SSL daughter card:

```
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 username and password for the SSH connection using either a simple username 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, page 7-4
- Configuring the Username and Password for SSH, page 7-5
- Configuring Authentication, Authorization, and Accounting for SSH, page 7-6

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 the RSA Key Pairs" section on page 8-5) 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 rsa keypair-name ssh_key_name`	Assigns the key pair name to SSH.
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 1.5
Authentication timeout: 120 secs; Authentication retries: 3
ssl-proxy#
```

Configuring the Username and Password for SSH

To configure the username 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 username 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 username and password for the SSH connection to the SSL daughter card:

```
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 username and password, see the "Recovering a Lost Password" section on page 3-14 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 the 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 daughter card:

```
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 "Recovering a Lost Password" section on page 3-14 to configure the switch.

Configuring SSL for Client-Side and Server-Side Operation

This section describes how to configure the CSM-S. These topics are discussed in this section:

When you configure the server farm, if the real server is the SSL daughter card, you must use the **local** keyword when defining the real server.

This example shows how to configure the CSM to support SSL:

```
Cat6k-2(config-module-csm)# serverfarm SSLfarm
Cat6k-2(config-slb-sfarm)# real 10.1.0.21 local
Cat6k-2(config-slb-real)# inservice

Cat6k-2(config-module-csm)# vserver VS1
Cat6k-2(config-slb-vserver)# virtual 10.1.0.21 tcp https
```

```
Cat6k-2(config-slb-vserver)# serverfarm SSLfarm
Cat6k-2(config-slb-vserver)# inservice
```

The local keyword on the real server is the only CSM configuration change. Additional configuration is required on the CSM-S for proper CSM-SSL daughter card communication.

Configuring the Client Side

This example shows how to configure the SSL proxy service on the SSL daughter card:

```
ssl-proxy(config)# ssl-proxy service S1
ssl-proxy(config-ssl-proxy)# virtual ipaddr 10.1.0.21 protocol tcp port 443 secondary
ssl-proxy(config-ssl-proxy)# server ipaddr 10.2.0.100 protocol TCP port 80
ssl-proxy(config-ssl-proxy)# inservice
```

This example shows how to configure the CSM virtual server:

```
Cat6k-2(config-module-csm)# serverfarm SSLfarm
Cat6k-2(config-slb-sfarm)# real 10.1.0.21 local
Cat6k-2(config-slb-real)# inservice

Cat6k-2(config-module-csm)# vserver VS1
Cat6k-2(config-slb-vserver)# virtual 10.1.0.21 tcp https
Cat6k-2(config-slb-vserver)# serverfarm SSLfarm
Cat6k-2(config-slb-vserver)# inservice
```

You can perform SSL load balancing between the SSL daughter card and an SSL Services Module in mixed mode.

The CSM uses SSL-ID sticky functionality to stick SSL connections to the same SSL Services Module. The CSM must terminate the client-side TCP connection to inspect the SSL-ID. The CSM must then initiate a TCP connection to either the SSL daughter card or the SSL Services Module when a load-balancing decision has been made.

The traffic flow has the CSM passing all traffic received on a virtual server to the SSL daughter card with TCP termination performed on the SSL daughter card itself. When you enable the SSL sticky function, the connection between the CSM and SSL daughter card becomes a full TCP connection.

This example shows how to configure mixed-mode SSL load balancing:

```
Cat6k-2(config-module-csm)# sticky 10 ssl timeout 60
Cat6k-2(config-module-csm)# serverfarm SSLfarm
Cat6k-2(config-slb-sfarm)# real 10.1.0.21 local
Cat6k-2(config-slb-sfarm)# inservice
Cat6k-2(config-slb-sfarm)# real 10.2.0.21
Cat6k-2(config-slb-sfarm)# inservice
Cat6k-2(config-module-csm)# vserver VS1
Cat6k-2(config-slb-vserver)# virtual 10.1.0.21 tcp https
Cat6k-2(config-slb-vserver)# sticky 60 group 10
Cat6k-2(config-slb-vserver)# serverfarm SSLfarm
Cat6k-2(config-slb-vserver)# persistent rebalance
Cat6k-2(config-slb-vserver)# inservice
```

Additionally, you must make an internally generated configuration to direct traffic at the SSL daughter card when the CSM must terminate the client-side TCP connection. You must create a virtual server with the same IP address or port of each local real server in the server farm *SSLfarm*. Internally, this virtual server is configured to direct all traffic that is intended for the virtual server to the SSL daughter card.

You must make an internally generated configuration because the IP address of the local real server and the SSL daughter card virtual server address must be the same. When the CSM initiates a connection to this local real server, the SYN frame is both sent and received by the CSM. When the CSM receives the SYN and the destination IP address or port is the same as the virtual server VS1, it matches VS1 unless a more-specific virtual server is added.

Configuring the Server Side

A standard virtual server configuration is used for Layer 4 and Layer 7 load balancing when the SSL daughter card uses the CSM as the back-end server.

To restrict this virtual server to receive traffic from the SSL daughter card only, use the VLAN local virtual server submode command as follows:

```
Cat6k-2(config-module-csm)# serverfarm SLBdefaultfarm
Cat6k-2(config-slb-sfarm)# real 10.2.0.20
Cat6k-2(config-slb-sfarm)# inservice

Cat6k-2(config-module-csm)# vserver VS2
Cat6k-2(config-slb-vserver)# virtual 10.2.0.100 tcp www
Cat6k-2(config-slb-vserver)# serverfarm SLBdefaultfarm
Cat6k-2(config-slb-vserver)# vlan local
Cat6k-2(config-slb-vserver)# inservice
```

You can configure the real server as the back end as shown in this example:

```
Cat6k-2(config-module-csm)# serverfarm SSLpredictorforward
Cat6k-2(config-slb-sfarm)# predictor forward

Cat6k-2(config-module-csm)# vserver VS3
Cat6k-2(config-slb-vserver)# virtual 0.0.0.0 0.0.0.0 tcp www
Cat6k-2(config-slb-vserver)# serverfarm SSLpredictorforward
Cat6k-2(config-slb-vserver)# inservice
```

Configuring the CSM as the Back-End Server

The virtual server and server farm configurations permit the SSL daughter card to use real servers as the back-end servers. Use the configuration that is described in the "Configuring the Client Side" section on page 7-7, and then configure the SSL daughter card to use the CSM as the back-end server:

This example shows the CSM virtual server configuration for Layer 7 load balancing:

```
Cat6k-2(config-module-csm)# serverfarm SLBdefaultfarm
Cat6k-2(config-slb-sfarm)# real 10.2.0.20
Cat6k-2(config-slb-real)# inservice

Cat6k-2(config-module-csm)# serverfarm SLBjpgfarm
Cat6k-2(config-slb-sfarm)# real 10.2.0.21

Cat6k-2(config-module-csm)# map JPG url
Cat6k-2(config-slb-map-cookie)# match protocol http url *jpg*

Cat6k-2(config-module-csm)# policy SLBjpg
Cat6k-2(config-slb-policy)# url-map JPG
Cat6k-2(config-slb-policy)#serverfarm SLBjpgfarm
```

```
Cat6k-2(config-module-csm)# vserver VS2
Cat6k-2(config-slb-vserver)# virtual 10.2.0.100 tcp www
Cat6k-2(config-slb-vserver)# serverfarm SLBdefaultfarm
Cat6k-2(config-slb-vserver)# slb-policy SLBjpg
Cat6k-2(config-slb-vserver)# inservice
```

This example shows the CSM virtual server configuration for Layer 4 load balancing:

```
Cat6k-2(config-module-csm)# serverfarm SLBdefaultfarm
Cat6k-2(config-slb-sfarm)# real 10.2.0.20
Cat6k-2(config-slb-real)# inservice

Cat6k-2(config-module-csm)# vserver VS2
Cat6k-2(config-slb-vserver)# virtual 10.2.0.100 tcp www
Cat6k-2(config-slb-vserver)# serverfarm SLBdefaultfarm
Cat6k-2(config-slb-vserver)# vlan local
Cat6k-2(config-slb-vserver)# inservice
```

Configuring the Real Server as the Back-End Server

The server side configuration traffic flow with the real server as the back end is similar to the client-side configuration. Use the configuration that is described in the "Configuring the Client Side" section on page 7-7, and then configure the SSL daughter card to use a real server as the back-end server:

No new configuration is required for the SSL daughter card proxy service configuration. This example shows how the configuration is internally initiated and hidden from the user:

```
ssl-proxy(config)# ssl-proxy service S1
ssl-proxy(config-ssl-proxy)# virtual ipaddr 10.1.0.21 protocol tcp port 443 secondary
ssl-proxy(config-ssl-proxy)# server ipaddr 10.2.0.20 protocol TCP port 80
ssl-proxy(config-ssl-proxy)# inservice
```

This example shows how to configure the CSM virtual server:

```
Cat6k-2(config-module-csm)# serverfarm SSLreals

Cat6k-2(config-slb-sfarm)# real 10.2.0.20
Cat6k-2(config-slb-sfarm)# inservice

Cat6k-2(config-module-csm)# serverfarm SSLpredictorforward
Cat6k-2(config-slb-sfarm)# predictor forward

Cat6k-2(config-module-csm)# vserver VS3
Cat6k-2(config-slb-vserver)# virtual 0.0.0.0 0.0.0.0 tcp www
Cat6k-2(config-slb-vserver)# serverfarm SSLpredictorforward
Cat6k-2(config-slb-vserver)# inservice
```

Configuring Policies

This section describes how to configure the SSL and TCP policies:

Configuring SSL Policy

Note The SSL commands for the SSL daughter card apply either globally or to a particular proxy server. See the "SSL Server Proxy Services" section on page 7-18 for procedures for applying policies to a proxy service.

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

One parameter that 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 the 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 daughter card expects a close-notify alert message. If an SSL peer does not send a close-notify alert, the SSL daughter card removes the session from the session cache so that the same session ID cannot be used for future SSL connections.

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

- strict—The SSL daughter card sends a close-notify alert message to the SSL peer, and the SSL daughter card expects a close-notify alert message from the SSL peer. If the SSL daughter card does not receive a close-notify alert, SSL resumption is not allowed for that session.
- none—The SSL daughter card does not send a close-notify alert message to the SSL peer, and the SSL daughter card does not expect a close-notify alert message from the SSL peer. If the SSL daughter card receives a close-notify alert from the SSL peer, the SSL daughter card preserves the session information so that SSL resumption can be used for future SSL connections. However, if the SSL daughter card does not receive a close-notify alert from the SSL peer, SSL resumption is not allowed for that session.
- disabled (default)—The SSL daughter card sends a close-notify alert to the SSL peer; however, the SSL peer does not expect a close-notify alert before removing the session. Whether or not the SSL peer sends a close-notify alert, 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 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 policy ssl` *ssl_policy_name*	Defines SSL policy templates.
Step 2	`ssl-proxy (config-ssl-policy)# cipher {rsa-with-rc4-128-md5 | rsa-with-rc4-128-sha | rsa-with-des-cbc-sha | rsa-with-3des-ede-cbc-sha | others...}`	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.

	Command	Purpose
Step 3	`ssl-proxy (config-ssl-policy)# protocol {ssl3 | tls1 | all}`	Defines the various protocol versions supported by the proxy server.
Step 4	`ssl-proxy (config-ssl-policy)# timeout handshake time`	Configures how long the module keeps the connection in the handshake phase. The valid range is 0 to 65535 seconds.
Step 5	`ssl-proxy (config-ssl-policy)# close-protocol {strict | none}`	Configures the SSL close-protocol behavior. The close-protocol behavior is disabled by default.
Step 6	`ssl-proxy (config-ssl-policy)# session-cache`	Enables the session-caching feature. Session caching is enabled by default.
Step 7	`ssl-proxy (config-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** You must use the **absolute** keyword to configure the session-cache size. The **absolute** keyword specifies that the session entry is kept in the session cache for the specified *timeout*. When the **absolute** keyword is specified, the new incoming connections are rejected, and no free entries are available in the session cache.
Step 8	`ssl-proxy (config-ssl-policy)# session-cache size size`	(Optional) Specifies the size of the session cache[1]. The valid range is from 1 to 262143 entries. **Note** Specify the session cache size when you enter the **absolute** keyword with the **timeout session** command. If you do not enter this command or specify a *size*, the session cache size is the maximum size (262,144).

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. You can limit the number of session entries by configuring the session-cache size.

Configuring TCP Policy

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

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

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

	Command	Purpose
Step 1	`ssl-proxy (config)# ssl-proxy policy tcp tcp_policy_name`	Defines TCP policy templates. All defaults are assumed unless otherwise specified.
Step 2	`ssl-proxy (config-tcp-policy)# timeout syn time`	Configures the connection establishment timeout. The default is 75 seconds. The valid range is from 5 to 75 seconds.

	Command	Purpose
Step 3	`ssl-proxy (config-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-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 from 0 to 960 seconds (0 = disabled).
Step 5	`ssl-proxy (config-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 to 960 seconds (0 = disabled).
Step 6	`ssl-proxy (config-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 7	`ssl-proxy (config-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 8	`ssl-proxy (config-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 9	`ssl-proxy(config-tcp-policy)#` **`delayed-ack-threshold`** *`delay`*	Configures the delayed ACK threshold. The default is 2. The valid range is from 1 to 10.
Step 10	`ssl-proxy(config-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 11	`ssl-proxy (config-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 12	`ssl-proxy (config-tcp-policy)#` [**`no`**] **`nagle`**	Enables or disables the Nagle algorithm.
Step 13	`ssl-proxy (config-tcp-policy)#` **`exit`**	Returns to configuration mode.

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

HTTP Header Insertion

In an SSL offloading environment, an SSL offloader terminates the secure client HTTP (HTTPS) connections, decrypts the SSL traffic into clear text, and forwards the clear text to a Web server through an HTTP connection. The HTTPS connections are not secure HTTP connections at the back-end server because the server does not know that the client connection came in as a secure connection.

You should configure the HTTP header insertion for the following reasons:

- HTTP header insertion allows the SSL daughter card to embed information into an HTTP header during a client connection. When the back-end server recognizes this header, the server returns all URLs as HTTPS.
- You can have a back-end 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 daughter card 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.

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

The custom headers and client IP and port headers are inserted in every HTTP request packet. Full session headers and decoded client certificate fields are inserted in the first HTTP request packets; only the session ID is inserted in subsequent HTTP requests that use the same session ID. The servers are expected to cache the session or client certificate headers 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, with each policy consisting of up to 32 prefixes or headers. The prefix and custom headers can include up to 240 characters.

These sections describe the information that can be inserted into the HTTP header:

- Prefix, page 7-13
- Client Certificate Headers, page 7-13
- Client IP and Port Address Headers, page 7-14
- Custom Headers, page 7-14
- SSL Session Headers, page 7-15

Prefix

When you specify **prefix** *prefix_string*, the SSL daughter card adds the specified prefix to every inserted HTTP header. Adding a prefix enables the server to identify connections as coming from the SSL daughter card and not from other appliances. A prefix is not added to standard HTTP headers from the client. The *prefix_string* can be up to 240 characters.

Client Certificate Headers

The client certificate header insertion allows the back-end server to see the attributes of the client certificate that the SSL daughter card has authenticated and approved. The 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.

When you specify **client-cert**, the SSL daughter card passes the following headers to the back-end 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	Algorithm used for the public key
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	Public RSA exponent
ClientCert-X509v3-Authority-Key-Identifier	X.509 authority key identifier
ClientCert-X509v3-Basic-Constraints	X.509 basic constraints
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 daughter card 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 daughter card 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 240 characters.

Note The syntax for *custom_string* is in the form *name*:*value*. The *custom_string* must be enclosed in quotation marks if it contains spaces as follows:

"SOFTWARE VERSION : 2.1(1)"

SSL Session Headers

The session headers, including the session ID, are used to cache client certificates based on the session ID. The 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 daughter card inserts the full session headers in the HTTP request during the full SSL handshake but inserts only the session ID when the session resumes.

When you configure the SSL daughter card as a client, the SSL daughter card inserts the session ID of the connection between the module and the back-end SSL server.

When you specify **session**, the SSL daughter card passes information specific to an SSL connection to the back-end server in the form of the following session headers.

Field to insert	Description
Session-Id	SSL session ID
Session-Cipher-Name	Symmetric cipher suite
Session-Cipher-Key-Size	Symmetric cipher key size
Session-Cipher-Use-Size	Symmetric cipher use

Configuring the HTTP Header Insertion

To configure the HTTP header insertion, perform this task:

	Command	Purpose
Step 1	`ssl-proxy (config)# ssl-proxy policy http-header` *policy_name*	Configures HTTP header insertion.
Step 2	`ssl-proxy(config-http-header-policy)# {prefix` *prefix_string* `| client-cert | client-ip-port | custom` *custom_string*`} | session]`	Specifies the prefix or type of header.
Step 3	`ssl-proxy(config-http-header-policy)# exit`	Returns to config mode.
Step 4	`ssl-proxy(config)# ssl-proxy service` *service_name*	Defines the name of the SSL proxy service. **Note** The *service_name* value is case sensitive.
Step 5	`ssl-proxy(config-ssl-proxy)# policy http-header` *http_header_policy_name*	Applies the HTTP header policy to the proxy server, for request.

This example shows how to configure the SSL daughter card to insert a prefix and session headers:

```
ssl-proxy (config)# ssl-proxy policy http-header ssl-offload
ssl-proxy(config-http-header-policy)# prefix SSL-OFFLOAD
ssl-proxy(config-http-header-policy)# session
ssl-proxy(config-http-header-policy)# custom "SOFTWARE VERSION:2.1(1)"
```

```
ssl-proxy(config-http-header-policy)# custom "module:SSL MODULE - CATALYST 6500"
ssl-proxy(config-http-header-policy)# custom type-of-proxy:server_proxy_1024_bit_key_size
ssl-proxy(config-http-header-policy)# exit
ssl-proxy(config)# ssl-proxy service ssl-offload
ssl-proxy(config-ssl-proxy)# policy http-header ssl-offload
```

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

```
SSL-OFFLOAD-SOFTWARE VERSION:2.1(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-Session-Cipher-Name:RC4-SHA
SSL-OFFLOAD-Session-Cipher-Key-Size:128
SSL-OFFLOAD-Session-Cipher-Use-Size:128
```

Configuring URL Rewrite

In a typical SSL offloading environment, an SSL offloader terminates secure client HTTP (HTTPS) connections, decrypts the SSL traffic into clear text, and forwards the clear text to a Web server through an HTTP connection. The HTTPS connections are not secure HTTP connections at the back-end server because the server does not know that the client connection came in as a secure connection.

If the 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 by using a clear text connection.

You can avoid problems with HTTP that are not secure redirects from the back-end server by configuring one or more URL rewrite rules. Each rewrite rule is associated with a service in the SSL proxy list. The URL rewrite rules resolve the problem of a website redirecting you to an HTTP URL that is not secure 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 URL rewrite 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. URL rewrite does not support embedded links.

URL rewrite rewrites the protocol and the nondefault port (default ports are port 80 for clear text and port 443 for SSL).

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

The guidelines for URL rewrite are as follows:

- 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.***, and 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 or ***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 overwrites the first rule.

- URL rewrite is performed for both offload and back-end servers (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 policy url-rewrite` *policy_name*	Configures the URL rewrite policy.
Step 2	`ssl-proxy(config-url-rewrite-policy)# url` *url* `[clearport` *port_number*`]`[1, 2] `[sslport` *port_number*`]`	Specifies the URL rewrite rules. You can configure up to 32 rewrite rules per SSL proxy service and up to 240 characters per rule. **Note** You should enter only one suffix or prefix wildcard character (*) once per rewrite rule.
Step 3	`ssl-proxy(config-url-rewrite-policy)# exit`	Returns to config mode.
Step 4	`ssl-proxy(config)# ssl-proxy service` *service_name*	Defines the name of the SSL proxy service. **Note** The *service_name* value is case sensitive.
Step 5	`ssl-proxy(config-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.

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

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

See Table 7-1 for examples that show URL rewrite.

Table 7-1 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/

Configuring the SSL Proxy Services

You define the SSL proxy services using the **ssl-proxy service** *ssl_proxy_name* command. You can configure the virtual IP address and port that is associated with the proxy service and the associated target IP address and port.

You define the TCP and SSL policies for both client (**virtual**) and server (**server**) sides of the proxy.

These sections describe how to configure the proxy services:

SSL Server Proxy Services

To configure the SSL server proxy services, perform this task:

	Command	Purpose
Step 1	`ssl-proxy(config)# ssl-proxy service` *service_name*	Defines the name of the SSL proxy service. **Note** The *service_name* value is case sensitive.
Step 2	`ssl-proxy(config-ssl-proxy)# virtual ipaddr` *ip_addr* [*mask_addr*][1,2] **protocol tcp port** *port* {**secondary**}	Defines the virtual server IP address, transport protocol (TCP), and port number for which the CSM-S is the proxy. **Note** The **secondary** keyword is always required.
Step 3	`ssl-proxy(config-ssl-proxy)# server ipaddr` *ip_addr* **protocol tcp port** *port*	Defines the IP address, port number, and 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 4	`ssl-proxy(config-ssl-proxy)# virtual policy tcp` *tcp_policy_name*[3]	(Optional) Applies a TCP policy to the client side of the proxy server. See the "Configuring TCP Policy" section on page 7-11 for TCP policy parameters.
Step 5	`ssl-proxy(config-ssl-proxy)# virtual policy ssl` *ssl_policy_name*[3]	(Optional) Applies an SSL policy to the client side of the proxy server. See the "Configuring SSL Policy" section on page 7-10 for SSL policy parameters.
Step 6	`ssl-proxy(config-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 7-11.
Step 7	`ssl-proxy(config-ssl-proxy)# policy http-header` *http_header_policy_name*	(Optional) Applies the HTTP header policy to the proxy server. See the "HTTP Header Insertion" section on page 7-13.
Step 8	`ssl-proxy(config-ssl-proxy)# policy url-rewrite` *url_rewrite_policy_name*	(Optional) Applies the URL rewrite policy. See the "Configuring URL Rewrite" section on page 7-16.

	Command	Purpose
Step 9	`ssl-proxy(config-ssl-proxy)#` **`trusted-ca`** *`ca_pool_name`*	(Optional) Associates the trusted certificate authority pool with the proxy service. See the "Client Certificate Authentication" section on page 8-41 for information on the certificate authority pools.
Step 10	`ssl-proxy(config-ssl-proxy)#` **`authenticate verify`** {**`signature-only`**[4] | **`all`**[5]}	(Optional) Enables the server certificate authentication and specifies the form of verification. See the "Server Certificate Authentication" section on page 8-43 for information on the server certificate authentication.
Step 11	`ssl-proxy(config-ssl-proxy)#` **`nat`** {**`server`** | **`client`** *`natpool_name`*}	(Optional) Specifies the usage of either server NAT[6] or client NAT for the server-side connection opened by the CSM-S. See the "Configuring NAT" section on page 7-22 and "Configuring NAT" section on page 7-22.
Step 12	`ssl-proxy(config-ssl-proxy)#` **`certificate rsa general-purpose trustpoint`** *`trustpoint_label`*	Applies a trustpoint configuration to the proxy server[7]. **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 8-7 for information on configuring the trust point.
Step 13	`ssl-proxy(config-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 daughter card does not respond to the ARP requests of the virtual IP address.
3. If you create a policy without specifying any parameters, the policy is created using the default values.
4. When you verify signature-only, authentication stops at the level that corresponds to one of the trusted certificate authority trustpoints in the trusted certificate authority pool.
5. When you verify all, the highest level issuer in the certificate chain must be configured as a trusted certificate authority trustpoint. The SSL daughter card 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.
6. NAT = network address translation
7. 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, and then repeat Step 12.

This example shows how to configure SSL proxy services:

```
ssl-proxy(config)# ssl-proxy service proxy1
ssl-proxy(config-ssl-proxy)# virtual ipaddr 10.1.1.100 protocol tcp port 443
ssl-proxy(config-ssl-proxy)# server ipaddr 10.1.1.1 protocol tcp port 80
ssl-proxy(config-ssl-proxy)# virtual policy tcp tcp2
ssl-proxy(config-ssl-proxy)# server policy tcp tcp2
ssl-proxy(config-ssl-proxy)# virtual policy ssl ssl1
ssl-proxy(config-ssl-proxy)# nat client t2
ssl-proxy(config-ssl-proxy)# certificate rsa general-purpose trustpoint tp1
ssl-proxy(config-ssl-proxy)# inservice
ssl-proxy(config-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 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-ssl-proxy)# server ipaddr 20.1.2.3 protocol tcp port 80
ssl-proxy(config-ssl-proxy)# virtual policy tcp tcp2
ssl-proxy(config-ssl-proxy)# server policy tcp tcp2
ssl-proxy(config-ssl-proxy)# virtual policy ssl ssl1
ssl-proxy(config-ssl-proxy)# inservice
ssl-proxy(config-ssl-proxy)# end
```

SSL Version 2.0 Forwarding

The SSL daughter card is not able to terminate SSL version 2.0 (SSLv2) connections. However, you can configure the SSL daughter card 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 daughter card 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, perform this task:

Command	Purpose
`ssl-proxy(config-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 the SSL proxy services to forward SSL v2.0 connections:

```
ssl-proxy(config)# ssl-proxy service frontend
ssl-proxy(config-ssl-proxy)# virtual ipaddr 35.200.200.102 protocol tcp port 443
ssl-proxy(config-ssl-proxy)# server ipaddr 26.51.51.1 protocol tcp port 80
ssl-proxy(config-ssl-proxy)# server ipaddr 26.51.51.2 protocol tcp port 443 sslv2
ssl-proxy(config-ssl-proxy)# certificate rsa general-purpose trustpoint test-cert
ssl-proxy(config-ssl-proxy)# inservice
ssl-proxy(config-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 back-end SSL server.

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

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

To configure SSL client proxy services, perform this task:

	Command	Purpose
Step 1	`ssl-proxy(config)# ssl-proxy service proxy_name client`	Defines the name of the SSL proxy service. The **client** keyword configures the SSL client proxy service. **Note** The *proxy-name* value is case sensitive.
Step 2	`ssl-proxy(config-ssl-proxy)# virtual ipaddr ip_addr [mask_addr]`[1] `protocol tcp port port secondary`	Defines the virtual server IP address, transport protocol (TCP), and port number for which the CSM-S is the proxy. **Note** The **secondary** keyword is required.
Step 3	`ssl-proxy(config-ssl-proxy)# server ipaddr ip_addr protocol tcp port port`	Defines the IP address, port number, and 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 4	`ssl-proxy(config-ssl-proxy)# virtual policy tcp tcp_policy_name`[2]	(Optional) Applies a TCP policy to the client side of the proxy server. See the "Configuring TCP Policy" section on page 7-11 for the TCP policy parameters.
Step 5	`ssl-proxy(config-ssl-proxy)# server policy ssl ssl_policy_name`[2]	(Optional) Applies an SSL policy to the server side of the proxy server. See the "Configuring SSL Policy" section on page 7-10 for the SSL policy parameters.
Step 6	`ssl-proxy(config-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 7-11.
Step 7	`ssl-proxy(config-ssl-proxy)# policy http-header http_header_policy_name`	(Optional) Applies the HTTP header policy to the proxy server. See the "HTTP Header Insertion" section on page 7-13.
Step 8	`ssl-proxy(config-ssl-proxy)# policy url-rewrite url_rewrite_policy_name`	(Optional) Applies the URL rewrite policy. See the "Configuring URL Rewrite" section on page 7-16.
Step 9	`ssl-proxy(config-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 8-41 for information on the certificate authority pools.
Step 10	`ssl-proxy(config-ssl-proxy)# authenticate verify {signature-only`[3] `| all`[4]`}`	Enables the client certificate authentication and specifies the form of verification. See the "Client Certificate Authentication" section on page 8-41 for information on the client certificate authentication.
Step 11	`ssl-proxy(config-ssl-proxy)# nat {server | client natpool_name}`	(Optional) Specifies the usage of either server NAT[5] or client NAT for the server-side connection opened by the SSL daughter card. See the "Configuring NAT" section on page 7-22.

	Command	Purpose
Step 12	`ssl-proxy(config-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 8-7 for information on configuring the trust point.
Step 13	`ssl-proxy(config-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. If you create a policy without specifying any parameters, the policy is created using the default values.
3. 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.
4. When you verify all, the highest level issuer in the certificate chain must be configured as a trusted certificate authority trustpoint. The SSL daughter card authenticates all 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.
5. NAT = network address translation

This example shows how to configure SSL client proxy services:

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

Configuring NAT

The client connections originate from the client and are terminated on the SSL daughter card. The server connections originate from the SSL daughter card.

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

Note If Client NAT is configured on the SSL daughter card, then you must also configure it on the CSM side for it to work.

Server NAT

The server IP address that is configured with the **ssl-proxy service** command specifies the IP address and port for the destination device, and either the SSL daughter card or the real server for which the SSL daughter card will act 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 the SSL daughter card is a proxy. The SSL daughter card always performs the port translation by using the port number entered in the **server ipaddress** subcommand.

To configure server NAT, perform this task:

	Command	Purpose
Step 1	`ssl-proxy (config)# ssl-proxy service ssl_proxy_name`	Defines the SSL proxy service.
Step 2	`ssl-proxy (config-ssl-proxy)# nat server`	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 daughter card (256,000 connections). Assuming that 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 daughter card, the command is rejected.

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 natpool natpool_name start_ip_addr end_ip_addr netmask`	Defines a pool of IP addresses that the SSL daughter card uses for implementing the client NAT.
Step 2	`ssl-proxy (config-ssl-proxy)# ssl-proxy service ssl_proxy_name`	Defines the SSL proxy service.
Step 3	`ssl-proxy (config-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 TACACS, TACACS+, and RADIUS

For information on configuring TACACS, TACACS+, and RADIUS, refer to these 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

For a list of supported MIBs, refer to this URL:

http://www.cisco.com/public/sw-center/netmgmt/cmtk/mibs.shtml

Note The Cisco product MIB ID for the CSM-S is ciscoproducts.610. This ID is different than the SSLM, which is ciscoproducts.554.

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) Enables the SSL proxy certificate expiration notification trap. **Note** If you set the certificate check-expiring interval to **0**, expiration notification traps are not sent. See the "Configuring the Certificate Expiration Warning" section on page 8-38 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) Enables 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 startup.

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 cryptographic 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 :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 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
```

This example shows how to display the forwarding data unit statistics:

```
ssl-proxy# show ssl-prox stats fdu
FDU Statistics:
    IP Frag Drops       : 0             IP Version Drops    : 0
    IP Addr Discards    : 0             Serv_Id Drops       : 0
    Conn Id Drops       : 0             Bound Conn Drops    : 0
    Vlan Id Drops       : 0             TCP Checksum Drops  : 0
    Hash Full Drops     : 0             Hash Alloc Fails    : 0
    Flow Creates        : 536701        Flow Deletes        : 536701
    Conn Id allocs      : 268354        Conn Id deallocs    : 268354
    Tagged Pkts Drops   : 0             Non-Tagg Pkts Drops : 0
    Add ipcs            : 3             Delete ipcs         : 0
    Disable ipcs        : 1             Enable ipcs         : 0
    Unsolicited ipcs    : 1345          Duplicate Add ipcs  : 0
    IOS Broadcast Pkts  : 43432         IOS Unicast Pkts    : 12899
    IOS Multicast Pkts  : 0             IOS Total Pkts      : 56331
    IOS Congest Drops   : 0             SYN Discards        : 0
FDU Debug Counters:
    Inv. Conn Drops     : 0             Inv. Conn Pkt Drops : 0
    Inv. TCP opcodes    : 0
    Inv. Fmt Pkt Drops  : 0             Inv. Bad Vlan ID    : 0
    Inv. Bad Ctl Command: 0             Inv. TCP Congest    : 0
    Inv. Bad Buffer Fmt : 0             Inv. Buf Undersized : 0
ssl-proxy#
```

Collecting Crash Information

The crash-info feature collects information 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. After you enter the **show ssl-proxy crash-info** command it takes from one to six 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 daughter card.

This example shows how to collect software-forced reset information:

```
ssl-proxy# show ssl-proxy crash-info

===== SSL daughter card - 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 daughter card - END OF CRASHINFO COLLECTION =======
```

Enabling VTS Debugging

A virtual terminal server (VTS) is built into the SSL daughter card for debugging different processors (FDU, TCP, SSL) 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 port 2001 to view debug information.

To display debugging information, perform this task:

Command	Purpose
ssl-proxy# [**no**] **debug ssl-proxy** {**fdu** | **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** {**tcp** | **fdu** | **ssl**} command from the SSL Certificate Management console. One connection is sent from a client and displays the logs found in TCP console.

This 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
```

This 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
```

CHAPTER 8

Configuring SSL Services Secure Transactions

This chapter describes how to configure the CSM-S from the command line interface (CLI) of the module and contains these sections:

Configuring the Public Key Infrastructure

The SSL daughter card on the CSM-S 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 and the clients to the server. The certificates, which are issued by certificate authorities, 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.

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.

Each SSL daughter card 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 to apply for a certificate for authentication.

We recommend that the certificates be stored in NVRAM so that the module does not need to query the certificate authority at startup to obtain the certificates or to automatically enroll. See the "Configuring a Root CA (Trusted Root)" section on page 8-28 for more information.

The SSL daughter card authenticates certificates that it receives from external devices when you configure the SSL daughter card as an SSL server and you configure the server proxy to authenticate the client certificate, or when you configure the SSL daughter card as an SSL client. The SSL daughter card 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. If a revocation check is necessary, the SSL daughter card downloads the certificate revocation list (CRL) from the certificate authority and looks up the serial number of the certificate received. See the "Certificate Revocation List" section on page 8-47 for information on 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 8-52 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 daughter card.

The SSL daughter card 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 username and password, bank account number, credit card number, or mother's maiden name.

If the certificate sender is an SSL client, the SSL daughter card 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 7-13 for information on configuring HTTP header insertion. See the "Client Certificate Authentication" section on page 8-41 for information on configuring client certificate authentication.

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

Configuring the Keys and the Certificates

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

- If you are using the 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 8-5 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 by using TFTP or cut-and-paste to create a PKCS10 file.
  - Request the SSL server certificate offline by using the PKCS10 package.
  - Import the SSL server certificate by using TFTP or cut-and-paste.

  See the "Manual Certificate Enrollment" section on page 8-11 for details.
- If you are using an external PKI system, do the following:
  - Generate PKCS12 or PEM files.
  - Import this file to the module.

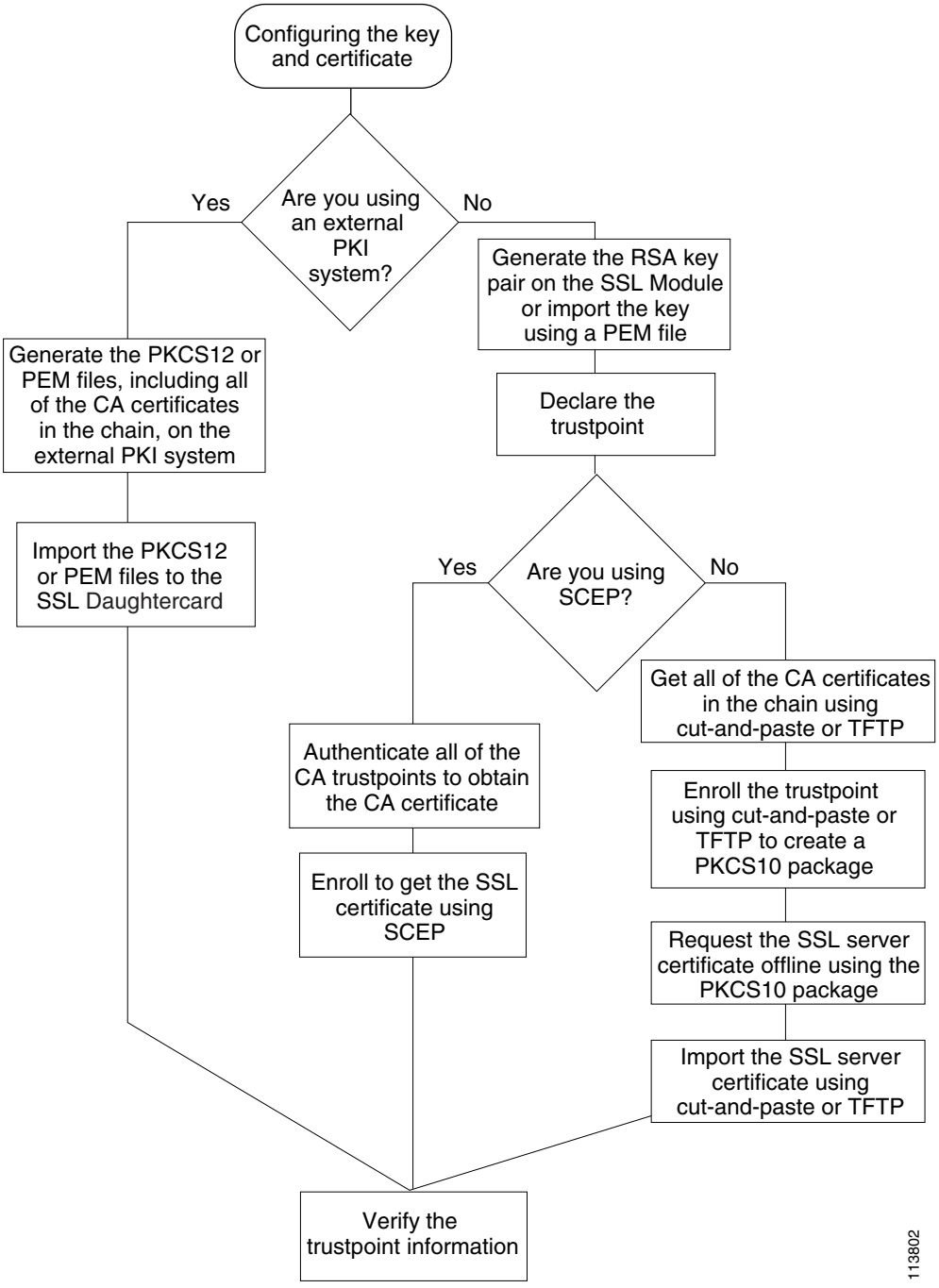
  See the "Importing and Exporting the Key Pairs and Certificates" section on page 8-19 for details.

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 trustpoin is created automatically if it does not already exist.

See Figure 8-1 for an overview on configuring keys and certificates.

Figure 8-1 Key and Certificate Configuration Overview



Configuring the Trustpoint Using SCEP

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

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

Generating the RSA Key Pairs

Note The first key pair that is generated enables SSH on the module. If you are using SSH, configure a key pair for SSH. See the "Configuring SSH" section on page 7-4.

RSA is the public key cryptographic system developed by Ron Rivest, Adi Shamir, and Leonard Aldeman. The RSA algorithm is widely used by the certificate authorities and the 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 the 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 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 daughter card 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 8-9.

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 that 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 daughter card 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 and the key pairs to the private configuration file in the module NVRAM.

To generate the RSA key pairs, perform this task:

Command	Purpose
ssl-proxy(config)# **crypto key generate rsa** [**usage-keys** | **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 the RSA keys, you are prompted to enter a modulus length in bits. The SSL daughter card 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 special-usage RSA keys:

```
crypto key generate rsa usage-keys

The name for the keys will be: myrouter.example.com
Choose the size of the key modulus in the range of 360 to 2048 for your Signature Keys.
Choosing a key modulus greater than 512 may take a few minutes.
How many bits in the modulus[512]? <return>

Generating RSA keys.... [OK].

Choose the size of the key modulus in the range of 360 to 2048 for your Encryption Keys.
Choosing a key modulus greater than 512 may take a few minutes.
How many bits in the modulus[512]? <return>

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

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

Note You cannot generate both special-usage and general-purpose keys; you can generate only one or the other.

```
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].
```

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 ca trustpoint` *trustpoint-label*[1]	Declares the trustpoint that your module should use. Enabling this command puts you in 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 that will use this certificate[2].
Step 5	`ssl-proxy(ca-trustpoint)# crl [best-effort | optional | query` *host:[port]*`]`	(Optional) Specifies how this trustpoint looks up a certificate revocation list when validating a certificate associated with this trustpoint. See the "Certificate Revocation List" section on page 8-47 for information on CRLs.
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 match is not required.
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 ca 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)# crl optional
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 ca 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 ca 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 ca 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 if 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 ca 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 ca 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 ca certificate' command will also show the fingerprint.
Fingerprint:  470DE382 65D8156B 0F84C2AF 4538B913

ssl-proxy(config)# end
```

After you configure the trustpoint, see the "Verifying the Certificates and the Trustpoints" section on page 8-27 to verify the certificate and trustpoint information.

Example of Three-Tier Certificate Authority Enrollment

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

This example shows how to configure three levels of certificate authority:

Generating the Keys

This example shows how to generate a key:

```
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

This example shows how to define a trustpoint:

```
ssl-proxy(config)# crypto ca 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 ca 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 ca 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 ca 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 ca 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 ca 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 ca 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 ca 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)—This method is the most common method for sending and receiving requests and certificates.
- A network connection between the router and the certificate authority is not possible—A router running Cisco IOS software obtains its certificate using this method.

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 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. Make sure that 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, you should enter the **crl optional** command for all trustpoints that are linked to the import process. Enter the **show crypto ca 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 **crl optional** 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 **crl optional** command for all the trustpoints associated with root CA. Similarly, when you import the subordinate CA2 certificate, enter the **crl optional** 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.

Configuring a Certificate Enrollment Using TFTP (One-Tier Certificate Authority)

To configure the certificate enrollment using TFTP, perform these steps:

Step 1 Configure the trustpoint:

```
ssl-proxy# configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
ssl-proxy(config)# crypto ca 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
```

Step 2 Request a certificate for the trustpoint:

```
ssl-proxy(config)# crypto ca 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 ca certificate' command will also show the fingerprint.
ssl-proxy(config)#    Fingerprint:  D012D925 96F4B5C9 661FEC1E 207786B7
!!
```

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

```
ssl-proxy(config)# crypto ca 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)#
```

Step 4 Import the server certificate:

```
ssl-proxy(config)# crypto ca 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)#
```

Configuring a Certificate Enrollment Using Cut-and-Paste (One-Tier Certificate Authority)

To configure the certificate enrollment using cut-and-paste, perform these steps:

Step 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]
```

Step 2 Configure the trustpoints:

```
ssl-proxy(config)# crypto ca 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
```

a. Request a certificate for the trustpoint:

```
ssl-proxy(config)# crypto ca 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
```

Step 3 Import the certificate authority certificate:

```
ssl-proxy(config)# crypto ca 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)#
```

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

```
ssl-proxy(config)# crypto ca 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
```

Configuring a Certificate Enrollment Using TFTP (Three-Tier Certificate Authority)

To configure certificate enrollment using TFTP, perform these steps:

Step 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]
```

Step 2 Configure the trustpoint:

```
ssl-proxy(config)# crypto ca 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
```

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

```
ssl-proxy(config)# crypto ca 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 ca certificate' command will also show the fingerprint.

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

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

Step 4 Use the CSR to acquire the SSL certificate offline from the third-level certificate authority.

Step 5 Authenticate the three certificate authorities (one root and two subordinate certificate authorities):

```
ssl-proxy(config)# crypto ca trustpoint test-1tier
ssl-proxy(ca-trustpoint)# enrollment url tftp://10.1.1.3/test-1tier
ssl-proxy(ca-trustpoint)# crl optional
ssl-proxy(ca-trustpoint)# exit
ssl-proxy(config)# crypto ca 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 ca trustpoint test-2tier
ssl-proxy(ca-trustpoint)# enrollment url tftp://10.1.1.3/test-2tier
ssl-proxy(ca-trustpoint)# crl optional
ssl-proxy(ca-trustpoint)# exit
ssl-proxy(config)# crypto ca 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 ca 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.
```

Step 6 Import the server certificate:

```
ssl-proxy(config)# crypto ca 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
```

Configuring a Certificate Enrollment Using Cut-and-Paste (Three-Tier Certificate Authority)

To configure a certificate enrollment using cut-and-paste, perform these steps:

Step 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]
```

Step 2 Configure the trustpoint:

```
ssl-proxy(config)# crypto ca 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
```

Step 3 Request a certificate for the trustpoint:

```
ssl-proxy(config)# crypto ca 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
```

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

Step 5 Define and import all certificate authorities (one root and two subordinate certificate authorities).

a. Define the two truspoints for the root certificate authority and subordinate 1 certificate authority.

Note Use **tp-proxy1** to import the subordinate 2 certificate authority certificate.

```
ssl-proxy(config)# crypto ca trustpoint 3tier-root
ssl-proxy(ca-trustpoint)# enrollment terminal
ssl-proxy(ca-trustpoint)# crl op
```

```
ssl-proxy(ca-trustpoint)# exit
ssl-proxy(config)# crypto ca trustpoint 3tier-sub1
ssl-proxy(ca-trustpoint)# enrollment terminal
ssl-proxy(ca-trustpoint)# crl op
ssl-proxy(ca-trustpoint)# exit
```

b. Import the root certificate authority certificate:

```
ssl-proxy(config)# crypto ca 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/lNVu0HjUORRdeCm1/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 the subordinate 1 certificate authority certificate:

```
ssl-proxy(config)# crypto ca 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 ca 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/3Z4y6w5WoeioIpCOCSydhnbd5wnwuethoyStVt91r6i61jWKl
d68Z8EoTg71daiV/WR/HAgMBAAGjggJjMIICXzAQBgkrBgEEAYI3FQEEAwIBADAd
BgNVHQ4EFgQU6FmJopqqzpbFMj6TaB2/wjlWlqEwCwYDVR0PBAQDAgHGMA8GA1Ud
EwEB/wQFMAMBAf8wgagGA1UdIwSBoDCBnYAUWaaNN2U14BaBoU9mY+ncuHpP922h
eaR3MHUxCzAJBgNVBAYTAlVTMRMwEQYDVQQIEwpjYWxpZm9ybmlhMREwDwYDVQQH
EwhzYW4gam9zZTEOMAwGA1UEChMFY2lzY28xDDAKBgNVBAsTA2hzczEgMB4GA1UE
AxMXc2ltcHNvbi1kZXZ0ZXN0LXJvb3QtQ0GCCho9HAcAAAAAAA4wgZcGA1UdHwSB
jzCBjDBoEGgP4Y9aHR0cDovL2Npc2NvLWcyNXVhNm80ZS9DZXJ0RW5yb2xsL3Np
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 ca 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 the 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.

These sections describe how to import or export key pairs and certificates:

- Importing and Exporting a PKCS12 File, page 8-20
- Importing and Exporting the PEM Files, page 8-21

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 and Exporting the Key Pairs and Certificates" section on page 8-19.

Note 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. Make sure that 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 **crl optional** command for all trustpoints that are linked to the import process. Enter the **show crypto ca 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 **crl optional** 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 **crl optional** command for all the trustpoints associated with root CA. Similarly, when you import the subordinate CA2 certificate, enter

the **crl optional** 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 ca {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 ca 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 ca 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 ca 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 ca 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 the Certificates and the Trustpoints" section on page 8-27 to verify the certificate and trustpoint information.

Importing and Exporting the PEM Files

Note The **crypto ca 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 that you use secure file transfer (SCP) 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 ca 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 ca 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 the PEM files using TFTP:

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

```
ssl-proxy(config)# crypto ca 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 ca 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 the Certificates and the Trustpoints" section on page 8-27 to verify the certificate and trustpoint information.

Importing the 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.

To import the PEM files for three levels of certificate authority, perform these steps:

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

```
ssl-proxy(config)# crypto ca trustpoint 3tier-root
ssl-proxy(ca-trustpoint)# enrollment terminal
ssl-proxy(ca-trustpoint)# crl optional
ssl-proxy(ca-trustpoint)# exit
ssl-proxy(config)# crypto ca 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
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
```

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

```
ssl-proxy(config)# crypto ca trustpoint 3tier-subca1
ssl-proxy(ca-trustpoint)# enroll terminal
ssl-proxy(ca-trustpoint)# crl optional
ssl-proxy(ca-trustpoint)# exit
ssl-proxy(config)# crypto ca 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
A1UdHwSBjzCBjDBoEGgP4Y9aHR0cDovL2Npc2NvLWw4ajZvaHBuci9DZXJ0RW5y
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
```

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

```
ssl-proxy(config)# crypto ca 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#
```

Step 4 Display the certificate information (optional):

```
ssl-proxy# show crypto ca 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 ca 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 ca 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 the Certificates and the Trustpoints

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

	Command	Purpose
Step 1	`ssl-proxy(ca-trustpoint)# **show crypto ca 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 ca trustpoints** [*trustpoint_label*]`	Displays information about all trustpoints or the specified trustpoint.

Sharing the Keys and the Certificates

The SSL daughter card supports the sharing of the same key pair by multiple certificates. However, this practice is not recommended 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 daughter card does not impose any restrictions on sharing certificates among multiple proxy services and multiple SSL daughter cards. 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 your legal staff regarding business contractual aspects.

Some web browsers compare the subject name of the server certificate with the host name or the IP address that appears on the URL. If the subject name does not match the host name or IP address, a dialog box appears, prompting the user to verify and accept the certificate. To avoid this step, you should limit the sharing of certificates based on the host name or IP address.

Configuring a Root CA (Trusted Root)

To configure a trusted root, perform this task beginning in global configuration mode:

	Command	Purpose
Step 1	`Router(config)# crypto ca trusted-root` *name*	Configures a root with a selected name and enters trusted root configuration mode.
Step 2	`Router(ca-root)# crl query` *url*	(Optional) Queries the CRL published by the configured root with the LDAP[1] URL.
Step 3	`Router(ca-root)# exit`	(Optional) Exits trusted root configuration mode.
Step 4	`Router(config)# crypto ca trustpoint` *name*	(Optional) Enters certificate authority identity configuration mode.
Step 5	`Router(ca-identity) #crl optional`	(Optional) Allows other peer certificates to be accepted by your switch, even if the appropriate CRL is not accessible to your switch.
Step 6	`Router(ca-identity)# exit`	(Optional) Exits certificate authority identity configuration mode.
Step 7	`Router(config)# crypto ca trusted-root` *name*	(Optional) Enters trusted root configuration mode.
Step 8	`Router(ca-root)# root {CEP` *url* `| TFTP` *server-hostname filename}*	Uses SCEP[2] with the given identity and URL, or uses TFTP to get a root certificate. **Note** Use TFTP only if your CA server does not support SCEP. **Note** When you are using TFTP, the server must be secured so that the downloading of the root certificate is not subject to any attack.
Step 9	`Router(ca-root)# root proxy` *url*	Defines the HTTP proxy server for getting a root certificate.

1. LDAP = Lightweight Directory Access Protocol.
2. SCEP = Simple Certificate Enrollment Protocol.

Saving Your Configuration

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

Tip 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. At startup, the module does not need to query the certificate authority to obtain the certificates or to auto-enroll. **Note** For security reasons, you should enter the **/erase** keyword to erase the public and the private configuration files before updating the NVRAM. If you do not enter the **/erase** keyword, the key pairs from the old private configuration file may remain in the NVRAM. **Caution** When you enter the **/erase** keyword, 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 two 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.

You should 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 hexadecimal 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 that are saved as hexadecimal dumps are also saved, resulting in two copies of the same certificate: one in hexadecimal 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 7 minutes to display.

Erasing the Saved Configuration

To erase a saved configuration, perform one of these tasks:

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 two minutes to finish.

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

Backing Up the Keys and the 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, but you cannot recover private keys.

If a secure server is available, back up the 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.

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 transfers.

Monitoring and Maintaining the Keys and Certificates

These tasks are optional:

- Deleting the RSA Keys from the Module, page 8-31
- Displaying the Keys and Certificates, page 8-32
- Deleting the Certificates from the Configuration, page 8-32

Deleting the RSA Keys from the Module

Caution Deleting the SSH key disables SSH on the module. If you delete the SSH key, generate a new key. For more information, see the "Configuring SSH" section on page 7-4.

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.

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 ca enroll** command when you originally obtained the certificates.
- Manually remove the trustpoint from the configuration, as described in the "Deleting the Certificates from the Configuration" section on page 8-32.

Displaying the Keys and Certificates

To display the keys and certificates, perform one of these tasks in EXEC mode:

Command	Purpose
`ssl-proxy# show crypto key mypubkey rsa`	Displays the RSA public keys for the module.
`ssl-proxy# show crypto ca certificates` [*trustpoint_label*]	Displays the 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** keyword is specified, the hexadecimal 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 the Certificates from the Configuration

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

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

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

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 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.

This example shows how to assign a trustpoint to a proxy service:

```
ssl-proxy# configure terminal
ssl-proxy(config)# ssl-proxy service s2
ssl-proxy(config-ssl-proxy)# virtual ip 10.1.1.2 p tcp p 443
ssl-proxy(config-ssl-proxy)# server ip 20.0.0.3 p tcp p 80
```

```
ssl-proxy(config-ssl-proxy)# inservice
ssl-proxy(config-ssl-proxy)# certficate rsa general trustpoint tp-1
ssl-proxy(config-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#
```

This 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
ssl-proxy(config)# ssl-proxy service s2
ssl-proxy(config-ssl-proxy)# certificate rsa general trustpoint tp-2
ssl-proxy(config-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 that you 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 CSM-S.

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

This example shows that the 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 service s2
ssl-proxy(config-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
```

This example shows that 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
```

This example shows that the existing and new connections use the old certificate until trustpoint t2 reenrolls. After trustpoint t2 reenrolls, the new connections use the new certificate; the 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 ca 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 ca 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
```

This example shows that 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
```

Configuring the Automatic Certificate Renewal and Enrollment

When you configure the 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 automatic 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 ca 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 ca timers`	Displays the time remaining before each timer expires.

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

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

Enabling the Key and Certificate History

Entering the **ssl-proxy pki history** command enables 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.

Entering the **show ssl-proxy certificate-history** command displays records. 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 the Peer Certificates

You can configure the SSL daughter card to cache the authenticated server and client (peer) certificates. Caching the authenticated peer certificates saves time by not requiring the SSL daughter card to authenticate the same certificate again. The SSL daughter card 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 you enter the **verifying all** command, the CRL lookup or an ACL filtering result could change within the specified timeout interval, causing the SSL daughter card to incorrectly accept a certificate that should be denied. For instance, the SSL daughter card could cache a peer certificate and, within the specified timeout, download an updated CRL in which the certificate is listed.

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

For example, in a site-to-site VPN environment, where two SSL daughter cards 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.

The peer certificates that expire within the specified time interval are not cached. The SSL daughter card uses the 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 the different proxy services at a 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 situation is considered a cache miss, and the peer certificate is verified.

To cache the 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 the Certificate Expiration Warning

You can configure the SSL daughter card 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 daughter card checks every 30 minutes for the expiration information for the following:

- All the proxy services
- The certificate authority certificates that are 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 daughter card 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 daughter card 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 that a proxy service certificate expiration warning is logged. For more information on configuring SNMP traps, see the "Configuring SNMP Traps" section on page 7-24.

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 a 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.
```

Configuring the 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 daughter card does not accept the certificate.

When you enter the **verify signature-only** command, the SSL daughter card 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 daughter card sorts the certificate chain in order, ignoring any unrelated or redundant certificates. The SSL daughter card 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 daughter card cannot trust the top-most certificate, the chain is rejected.

If the SSL daughter card trusts the top-most certificate, then the SSL daughter card 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 daughter card selects one of these trustpoints. Depending on the CRL and ACL map configuration for this trustpoint, the SSL daughter card performs revocation and certificate attribute filtering. If the CRL or ACL checking denies the certificate, the SSL daughter card rejects the chain.
- If the certificate is a X509 version 3 certificate authority certificate, the SSL daughter card 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 it takes longer and uses more CPU time.

You can download and update CRLs by entering CLI commands to reduce real-time delay. However, the CRL lookup is a slow process. See the "Certificate Revocation List" section on page 8-47 for information about the CRLs.

Client Certificate Authentication

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

To authenticate the SSL client, the SSL daughter card 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 daughter card 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 daughter card trusts only the certificates issued by the certificate authorities that you configure in the certificate authority pool.

Note For a proxy service to be operational, 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 goes 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 pool` *ca_pool_name*	Creates the certificate authority pool. **Note** You can create up to eight pools.
Step 2	`ssl-proxy(config-ssl-proxy)# 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 3	`ssl-proxy(config-ssl-proxy)# exit`	Returns to config mode.
Step 4	`ssl-proxy(config)# ssl-proxy service` *proxy_name*	Defines the name of the SSL server proxy service.
Step 5	`ssl-proxy(config-ssl-proxy)# trusted-ca` *ca_pool_name*	Associates the trusted certificate authority pool with the proxy service.

	Command	Purpose
Step 6	`ssl-proxy(config-ssl-proxy)# authenticate verify {signature-only`[2] `| all`[3]`}`	Enables the 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 7	`ssl-proxy(config-ssl-proxy)# exit`	Exits from configuration mode.

1. The SSL daughter card 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, the 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 daughter card 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 a client certificate authentication verifying the signature only:

```
ssl-proxy(config)# crypto ca trustpoint rootca
ssl-proxy(ca-trustpoint)# enrollment terminal
ssl-proxy(ca-trustpoint)# exit
ssl-proxy(config)#
ssl-proxy(config)# crypto ca 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(config)# ssl-proxy pool ca rootca
ssl-proxy(config-ca-pool)# ca trustpoint rootca
ssl-proxy(config-ca-pool)# ^Z
ssl-proxy(config)# ssl-proxy service client-auth-sig-only
ssl-proxy(config-ssl-proxy)# virtual ipaddr 14.0.0.1 protocol tcp port 443
ssl-proxy(config-ssl-proxy)# server ipaddr 24.0.0.1 protocol tcp port 80
```

```
ssl-proxy(config-ssl-proxy)# certificate rsa general-purpose trustpoint test-cert
ssl-proxy(config-ssl-proxy)# trusted-ca rootca
ssl-proxy(config-ssl-proxy)# authenticate verify signature-only
ssl-proxy(config-ssl-proxy)# inservice
ssl-proxy(config-ssl-proxy)# !
```

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

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

Server Certificate Authentication

When you configure the SSL daughter card as an SSL client (for example, for back-end encryption), the SSL daughter card always authenticates the SSL server.

To authenticate the SSL server, the SSL daughter card 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 daughter card 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. In this case, the SSL daughter card accepts only certificates that are 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 pool` *ca_pool_name*	Creates the certificate authority pool. **Note** You can create up to eight pools.
Step 2	`ssl-proxy(config-ssl-proxy)# 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 3	`ssl-proxy(config-ssl-proxy)# exit`	Returns to config mode.
Step 4	`ssl-proxy(config)# ssl-proxy service` *proxy_name* `client`	Defines the name of the SSL client proxy service. **Note** Enter the **client** keyword to configure the SSL client proxy services.
Step 5	`ssl-proxy(config-ssl-proxy)# trusted-ca` *ca_pool_name*	(Optional) Associates the certificate authority pool with the proxy service. **Note** If you specify **signature-only** in Step 6, you must configure a certificate authority pool.
Step 6	`ssl-proxy(config-ssl-proxy)# authenticate verify {signature-only`[2] `| all`[3]`}`	(Optional) Enables the server 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. You must configure a certificate authority pool to specify **signature-only**. The default is **all**.
Step 7	`ssl-proxy(config-ssl-proxy)# exit`	Exits configuration mode

1. The SSL daughter card 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, the 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 daughter card 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 the server certificate authentication verifying the signature only:

```
ssl-proxy(config)# crypto ca trustpoint rootca
ssl-proxy(ca-trustpoint)# enrollment terminal
ssl-proxy(ca-trustpoint)# exit
ssl-proxy(config)#
ssl-proxy(config)# crypto ca 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(config)# ssl-proxy pool ca rootca
ssl-proxy(config-ca-pool)# ca trustpoint rootca
ssl-proxy(config-ca-pool)# ^Z
ssl-proxy(config)# ssl-proxy service client-proxy-sig-only client
ssl-proxy(config-ssl-proxy)# virtual ipaddr 14.0.0.3 protocol tcp port 81
ssl-proxy(config-ssl-proxy)# server ipaddr 24.0.0.2 protocol tcp port 443
ssl-proxy(config-ssl-proxy)# trusted-ca rootca
ssl-proxy(config-ssl-proxy)# authenticate verify signature-only
ssl-proxy(config-ssl-proxy)# inservice
ssl-proxy(config-ssl-proxy)# !
```

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

```
ssl-proxy(config)# crypto ca trustpoint rootca
ssl-proxy(ca-trustpoint)# enrollment terminal
ssl-proxy(ca-trustpoint)# exit
ssl-proxy(config)#
ssl-proxy(config)# crypto ca 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(config)# ssl-proxy pool ca rootca
ssl-proxy(config-ca-pool)# ca trustpoint rootca
ssl-proxy(config-ca-pool)# ^Z
ssl-proxy(config)# ssl-proxy service client-proxy-verify-all client
ssl-proxy(config-ssl-proxy)# virtual ipaddr 14.0.0.4 protocol tcp port 81
ssl-proxy(config-ssl-proxy)# server ipaddr 24.0.0.2 protocol tcp port 443
ssl-proxy(config-ssl-proxy)# trusted-ca rootca
ssl-proxy(config-ssl-proxy)# authenticate verify all
ssl-proxy(config-ssl-proxy)# inservice
ssl-proxy(config-ssl-proxy)# ^Z
```

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

```
ssl-proxy(config)# crypto ca trustpoint rootca
ssl-proxy(ca-trustpoint)# enrollment terminal
ssl-proxy(ca-trustpoint)# crl optional
ssl-proxy(ca-trustpoint)# exit
ssl-proxy(config)#
ssl-proxy(config)# crypto ca 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(config)# ssl-proxy pool ca rootca
ssl-proxy(config-ca-pool)# ca trustpoint rootca
ssl-proxy(config-ca-pool)# ^Z
ssl-proxy(config)# ssl-proxy service client-proxy-sending-client-cert client
ssl-proxy(config-ssl-proxy)# virtual ipaddr 14.0.0.5 protocol tcp port 81
ssl-proxy(config-ssl-proxy)# server ipaddr 24.0.0.3 protocol tcp port 443
ssl-proxy(config-ssl-proxy)# certificate rsa general-purpose trustpoint test-cert
ssl-proxy(config-ssl-proxy)# trusted-ca rootca
ssl-proxy(config-ssl-proxy)# authenticate verify all
ssl-proxy(config-ssl-proxy)# inservice
ssl-proxy(config-ssl-proxy)# !
```

Certificate Revocation List

A certificate revocation list (CRL) is a time-stamped list that identifies the 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.

This section describes how to download and configure CRLs:

Downloading the CRL

If the certificate being validated contains a CRL distribution point (CDP) extension field, the module uses the CDP as the download path. The SSL daughter card 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 daughter card looks for a trustpoint that is associated with the certificate. If the SSL daughter card finds one or more associated trustpoints, the SSL daughter card uses one of the trustpoints to determine the download path and protocol using the SCEP enrollment URL. If there is no SCEP enrollment URL, the validation fails.

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

The SSL daughter card 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 a CRL request, the CRL download fails.
- The CRLs are signed by the root certificate authority. If the root certificate authority has been revoked, its CRL will probably become invalid. All trustpoints that are associated with a revoked root certificate authority should be deleted from the database as soon as the revocation is 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).

Configuring the CRL Options

Note More than one trustpoint can be associated with a root or subordinate certificate authority certificate. During certificate authentication, any of these trustpoints can be selected to determine CRL configuration. To obtain consistent authentication results, all of these trustpoints need to bear the same CRL configuration.

By default, the SSL daughter card always performs a CRL lookup if the trustpoint has been selected to validate a certificate. If the CRL is not in the database or has expired, the SSL daughter card downloads a CRL and saves it to the database for later use. If the CRL download fails, the SSL daughter card rejects the certificate that is being validated.

You can configure two options for CRL lookup:

- Best-effort

  If the SSL daughter card finds a CRL in the database and has not expired, then the SSL daughter card performs a CRL lookup. If the SSL daughter card does not find a CRL, the SSL daughter card attempts to download a CRL. However, if the CRL download fails, the SSL daughter card accepts the certificate.

- Optional

  If the SSL daughter card finds a CRL in the database that has not expired, then the SSL daughter card performs a CRL lookup. If the SSL daughter card does not find the CRL, the SSL daughter card accepts the certificate. The SSL daughter card makes no attempt to download a CRL.

To configure the CRL options, perform this task:

	Command	Purpose
Step 1	`ssl-proxy(config)# crypto ca 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)# crl` [**best-effort** | **optional**]	Specifies the CRL options.

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 ca 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, perform this task:

Command	Purpose
`ssl-proxy(config)# crypto ca 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 of the CRL.

This example shows how to download the CRL that is associated with the trustpoint “tp-root:”

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

Entering the X.500 CDP Information

You can enter the host name and port 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 certificate 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 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 the CRL download cannot be performed.

Note The CRL query must be sent on the administrative VLAN. You can either assign an IP address from the administrative VLAN to the LDAP server, or configure the LDAP server VLAN to be the administrative VLAN. To configure the administrative VLAN, refer to Configuring VLANs on the SSL Daughter Card, page 7-2.

To query the CRL with the X.500 URL, perform this task:

Command	Purpose
`ssl-proxy(ca-trustpoint)# crl query` *host:*`[`*port*`]`	Queries the CRL with the X.500 URL. The URL used to query the CRL must be an X.500 URL.

Entering a CRL Manually

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

To enter the CRL manually, perform this task:

	Command	Purpose
Step 1	`ssl-proxy(config)# crypto ca 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.

This example shows how to enter a CRL manually:

```
ssl-proxy(config)# crypto ca certificate chain tp
ssl-proxy(config-cert-chain)# crl
Enter the CRL in hexadecimalidecimal 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 the CRL Information

To display information about the CRLs, perform this task:

Command	Purpose
`ssl-proxy(config)# show crypto ca 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 ca 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 The CRLs are deleted globally and not per trustpoint.

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

To delete a CRL, perform this task:

Command	Purpose
`ssl-proxy(config-cert-chain)# no crl`	Deletes the CRL.

This example shows how to delete the CRL:

```
ssl-proxy(config)# crypto ca 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 you to specify an access control list (ACL) so that you can create a certificate-based ACL.

For information on configuring the 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

Configuring Redundancy

This chapter describes how to configure redundant connections and contains these sections:

Configuring Fault Tolerance

This section describes a fault-tolerant configuration. In this configuration, two separate Catalyst 6500 series chassis each contain a CSM-S.

Note You can also create a fault-tolerant configuration with two CSM-S modules in a single Catalyst 6500 series chassis. You also can create a fault-tolerant configuration in either the secure (router) mode or (bridge) mode that is not secure.

In the secure (router) mode, the client-side and server-side VLANs provide the fault-tolerant (redundant) connection paths between the CSM-S and the routers on the client side and the servers on the server side. In a redundant configuration, two CSM-S modules perform active and standby roles. Each CSM-S contains the same IP, virtual server, server pool, and real server information. From the client-side and server-side networks, each CSM-S is configured identically. The network sees the fault-tolerant configuration as a single CSM-S.

Note When you configure multiple fault-tolerant CSM-S pairs, do not configure multiple CSM-S pairs to use the same fault-tolerant VLAN. Use a different fault-tolerant VLAN for each fault-tolerant CSM-S pair.

Configuring fault tolerance requires the following:

- Two CSM-S modules that are installed in the Catalyst 6500 series chassis.
- Identically configured CSM-S modules. One CSM-S is configured as the active; the other is configured as the standby.
- Each CSM-S modules connected to the same client-side and server-side VLANs.

- Communication between the CSM-S modules provided by a shared private VLAN.
- A network that sees the redundant CSM-S modules as a single entity.
- Connection redundancy by configuring a link that has a 1-GB per-second capacity. Enable the calendar in the switch Cisco IOS software so that the CSM-S state change gets stamped with the correct time.

  The following command enables the calendar:

```
Cat6k-2# configure terminal
Cat6k-2(config)# clock timezone WORD offset from UTC
Cat6k-2(config)# clock calendar-valid
```

Because each CSM-S has a different IP address on the client-side and server-side VLAN, the CSM-S can send health monitor probes (see the "Configuring Probes for Health Monitoring" section on page 11-1) to the network and receive responses. Both the active and standby CSM-S modules send probes while operational. If the standby CSM-S assumes control, it knows the status of the servers because of the probe responses that it has received.

Connection replication supports both non-TCP connections and TCP connections. Enter the **replicate csrp** {**sticky** | **connection**} command in the virtual server mode to configure replication for the CSM-S modules.

Note The default setting for the **replicate** command is disabled.

To use connection replication for connection redundancy, enter these commands:

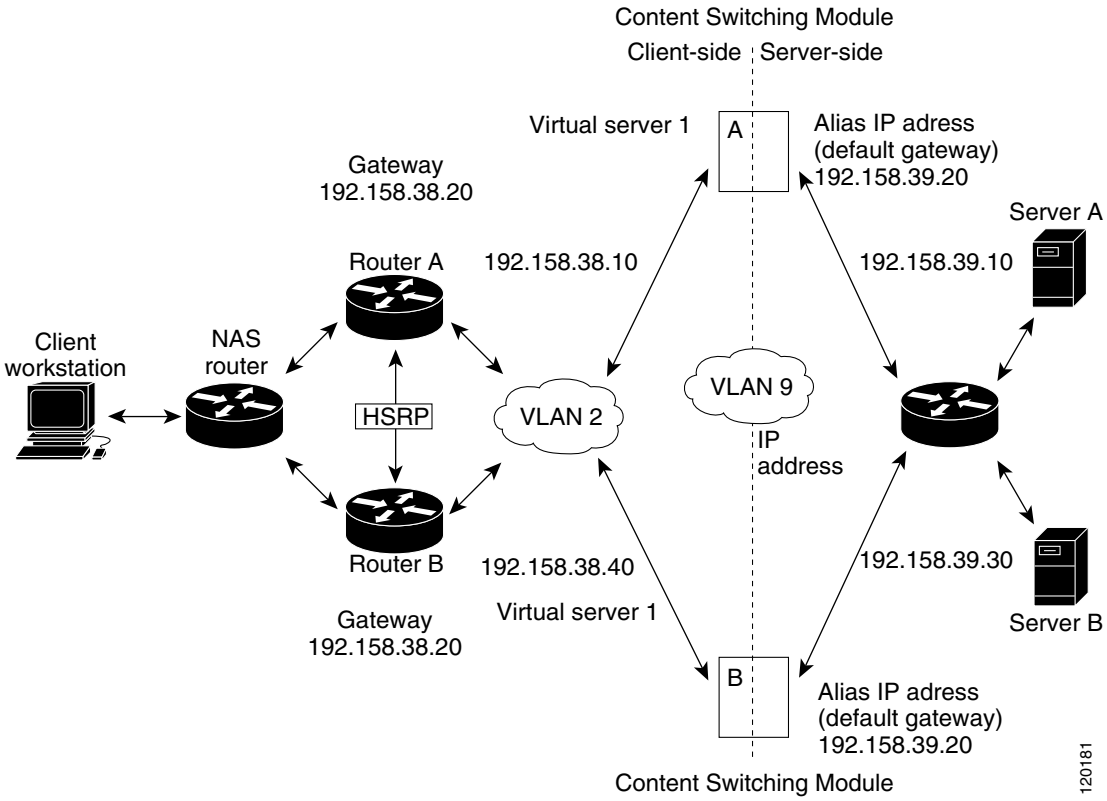
```
Cat6k-2# configure terminal
Cat6k-2(config)# no ip igmp snooping
```

You need to enter the **no ip igmp snooping** command because the replication frame has a multicast type destination MAC with a unicast IP address. When the switch listens to the Internet Group Management Protocol (IGMP) to find the multicast group membership and build its multicast forwarding information database (FIB), the switch does not find group members and prunes the multicast table. All multicast frames, from active to standby, are dropped causing erratic results.

If no router is present on the server-side VLAN, then each server's default route points to the aliased IP address.

Figure 9-1 shows how the secure (router) mode fault-tolerant configuration is set up.

Figure 9-1 Fault-Tolerant Configuration



Note The addresses in Figure 9-1 refer to the steps in the following two task tables.

To configure the active (A) CSM-S for fault tolerance, perform this task:

	Command	Purpose
Step 1	`Router(config-module-csm)# vlan 2 client`	Creates the client-side VLAN 2 and enters the SLB VLAN mode[1].
Step 2	`Router(config-slb-vlan-client)# ip addr 192.158.38.10 255.255.255.0`	Assigns the content switching IP address on VLAN 2.
Step 3	`Router(config-slb-vlan-client)# gateway 192.158.38.20`	(Optional) Defines the client-side VLAN gateway for an HSRP-enabled gateway.
Step 4	`Router(config-module-csm)# vserver vip1`	Creates a virtual server and enters the SLB vserver mode.

	Command	Purpose
Step 5	`Router(config-slb-vserver)# virtual 192.158.38.30 tcp www`	Creates a virtual IP address.
Step 6	`Router(config-module-csm)# inservice`	Enables the server.
Step 7	`Router(config-module-csm)# vlan 3 server`	Creates the server-side VLAN 3 and enters the SLB VLAN mode.
Step 8	`Router(config-slb-vlan-server)# ip addr 192.158.39.10 255.255.255.0`	Assigns the CSM-S IP address on VLAN 3.
Step 9	`Router(config-slb-vlan-server)# alias ip addr 192.158.39.20 255.255.255.0`	Assigns the default route for VLAN 3.
Step 10	`Router(config-slb-vlan-server) vlan 9`	Defines VLAN 9 as a fault-tolerant VLAN.
Step 11	`Router(config-module-csm)# ft group` *ft-group-number* `vlan 9`	Creates the content switching active and standby (A/B) group VLAN 9.
Step 12	`Router(config-module-csm)# vlan`	Enters the VLAN mode[1].
Step 13	`Router(vlan)# vlan 2`	Configures a client-side VLAN 2[2].
Step 14	`Router(vlan)# vlan 3`	Configures a server-side VLAN 3.
Step 15	`Router(vlan)# vlan 9`	Configures a fault-tolerant VLAN 9.
Step 16	`Router(vlan)# exit`	Enters the **exit** command to have the configuration take effect.

1. Enter the **exit** command to leave a mode or submode. Enter the **end** command to return to the menu's top level.
2. The **no** form of this command restores the defaults.

To configure the standby (B) CSM-S for fault tolerance, perform this task (see Figure 9-1):

	Command	Purpose
Step 1	`Router(config-module-csm)# vlan 2 client`	Creates the client-side VLAN 2 and enters the SLB VLAN mode[1].
Step 2	`Router(config-slb-vlan-client)# ip addr 192.158.38.40 255.255.255.0`	Assigns the content switching IP address on VLAN 2.
Step 3	`Router(config-slb-vlan-client)# gateway 192.158.38.20`	Defines the client-side VLAN gateway.
Step 4	`Router(config-module-csm)# vserver vip1`	Creates a virtual server and enters the SLB virtual server mode.
Step 5	`Router(config-slb-vserver)# virtual 192.158.38.30 tcp www`	Creates a virtual IP address.
Step 6	`Router(config-module-csm)# inservice`	Enables the server.
Step 7	`Router(config-module-csm)# vlan 3 server`	Creates the server-side VLAN 3 and enters the SLB VLAN mode.
Step 8	`Router(config-slb-vserver)# ip addr 192.158.39.30 255.255.255.0`	Assigns the CSM-S IP address on VLAN 3.
Step 9	`Router(config-slb-vserver)# alias 192.158.39.20 255.255.255.0`	Assigns the default route for VLAN 2.
Step 10	`Router(config-module-csm) vlan 9`	Defines VLAN 9 as a fault-tolerant VLAN.

	Command	Purpose
Step 11	`Router(config-module-csm)# ft group` *`ft-group-number`* `vlan 9`	Creates the CSM-S active and standby (A/B) group VLAN 9.
Step 12	`Router(config-module-csm)# show module csm all`	Displays the state of the fault-tolerant system.

1. Enter the **exit** command to leave a mode or submode. Enter the **end** command to return to the menu's top level.

Configuring HSRP

This section provides an overview of a Hot Standby Router Protocol (HSRP) configuration (see Figure 9-2) and describes how to configure the CSM-S modules with HSRP and CSM-S failover on the Catalyst 6500 series switches.

HSRP Configuration Overview

Figure 9-2 shows that two Catalyst 6500 series switches, Switch 1 and Switch 2, are configured to route from a client-side network (10.100/16) to an internal CSM-S client network (10.6/16, VLAN 136) through an HSRP gateway (10.100.0.1). The configuration shows the following:

- The client-side network is assigned an HSRP group ID of HSRP ID 2.
- The internal CSM-S client network is assigned an HSRP group ID of HSRP ID 1.

Note HSRP group 1 must have tracking turned on so that it can track the client network ports on HSRP group 2. When HSRP group 1 detects any changes in the active state of those ports, it duplicates those changes so that both the HSRP active (Switch 1) and HSRP standby (Switch 2) switches share the same knowledge of the network.

In the example configuration, two CSM-S modules (one in Switch 1 and one in Switch 2) are configured to forward traffic between a client-side and a server-side VLAN:

- Client VLAN 136

  Note The client VLAN is actually an internal CSM-S VLAN network; the actual client network is on the other side of the switch.

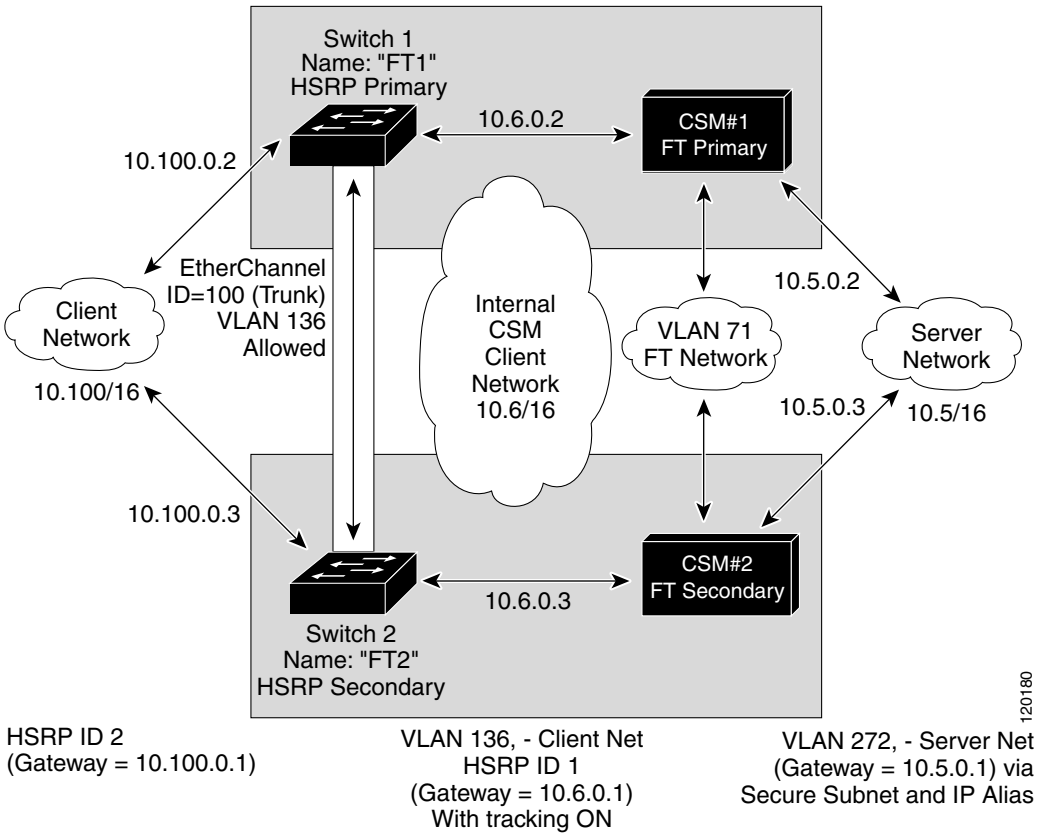
- Server VLAN 272

  The actual servers on the server network (10.5/1) point to the CSM-S server network through an alias gateway (10.5.0.1), allowing the servers to run a secure subnet.

  In the example configuration, an EtherChannel is set up with trunking enabled, allowing traffic on the internal CSM-S client network to travel between the two Catalyst 6500 series switches. The setup is shown in Figure 9-2.

  Note EtherChannel protects against a severed link to the active switch and a failure in a non-CSM-S component of the switch. EtherChannel also provides a path between an active CSM-S in one switch and another switch, allowing the CSM-S modules and switches to fail over independently, providing an extra level of fault tolerance.

Figure 9-2 HSRP Configuration



Creating the HSRP Gateway

This section describes how to create an HSRP gateway for the client-side network. The gateway is HSRP ID 2 for the client-side network.

Note In this example, HSRP is set on Fast Ethernet ports 3/6.

To create an HSRP gateway, perform these steps:

Step 1 Configure Switch 1—FT1 (HSRP active) as follows:

```
Router(config)# interface FastEthernet3/6
Router(config)# ip address 10.100.0.2 255.255.0.0
Router(config)# standby 2 priority 110 preempt
Router(config)# standby 2 ip 10.100.0.1
```

Step 2 Configure Switch 2—FT2 (HSRP standby) as follows:

```
Router(config)# interface FastEthernet3/6
Router(config)# ip address 10.100.0.3 255.255.0.0
Router(config)# standby 2 priority 100 preempt
Router(config)# standby 2 ip 10.100.0.1
```

Creating Fault-Tolerant HSRP Configurations

This section describes how to create a fault-tolerant HSRP secure-mode configuration. To create a nonsecure-mode configuration, enter the commands described with these exceptions:

- Assign the same IP address to both the server-side and the client-side VLANs.
- Do not use the **alias** command to assign a default gateway for the server-side VLAN.

To create fault-tolerant HSRP configurations, perform these steps:

Step 1 Configure VLANs on HSRP FT1 as follows:

```
Router(config)# module csm 5
Router(config-module-csm)# vlan 136 client
Router(config-slb-vlan-client)# ip address 10.6.0.245 255.255.0.0
Router(config-slb-vlan-client)# gateway 10.6.0.1
Router(config-slb-vlan-client)# exit

Router(config-module-csm)# vlan 272 server
Router(config-slb-vlan-server)# ip address 10.5.0.2 255.255.0.0
Router(config-slb-vlan-server)# alias 10.5.0.1 255.255.0.0
Router(config-slb-vlan-server)# exit

Router(config-module-csm)# vlan 71

Router(config-module-csm)# ft group 88 vlan 71
Router(config-slb-ft)# priority 30
Router(config-slb-ft)# preempt
Router(config-slb-ft)# exit

Router(config-module-csm)# interface Vlan136
ip address 10.6.0.2 255.255.0.0
standby 1 priority 100 preempt
standby 1 ip 10.6.0.1
standby 1 track Fa3/6 10
```

Step 2 Configure VLANs on HSRP FT2 as follows:

```
Router(config)# module csm 6
Router(config-module-csm)# vlan 136 client
Router(config-slb-vlan-client)# ip address 10.6.0.246 255.255.0.0
Router(config-slb-vlan-client)# gateway 10.6.0.1
Router(config-slb-vlan-client)# exit

Router(config-module-csm)# vlan 272 server
Router(config-slb-vlan-server)# ip address 10.5.0.3 255.255.0.0
Router(config-slb-vlan-server)# alias 10.5.0.1 255.255.0.0
Router(config-slb-vlan-server)# exit

Router(config-module-csm)# vlan 71

Router(config-module-csm)# ft group 88 vlan 71
```

```
Router(config-slb-ft)# priority 20
Router(config-slb-ft)# preempt
Router(config-slb-ft)# exit

Router(config-module-csm)# interface Vlan136
ip address 10.6.0.3 255.255.0.0
standby 1 priority 100 preempt
standby 1 ip 10.6.0.1
standby 1 track Fa3/6 10
```

Note To allow tracking to work, preempt must be on.

Step 3 Configure EtherChannel on both switches as follows:

```
Router(console)# interface Port-channel100
Router(console)# switchport
Router(console)# switchport trunk encapsulation dot1q
Router(console)# switchport trunk allowed vlan 136
```

Note By default, all VLANs are allowed on the port channel.

Step 4 To prevent problems, remove the server and fault-tolerant CSM-S VLANs as follows:

```
Router(console)# switchport trunk remove vlan 71
Router(console)# switchport trunk remove vlan 272
```

Step 5 Add ports to the EtherChannel as follows:

```
Router(console)# interface FastEthernet3/25
Router(console)# switchport
Router(console)# channel-group 100 mode on
```

Configuring Interface and Device Tracking

When you configure fault-tolerant HSRP, the active and standby state of the CSM-S does not follow the state of the active HSRP group. When the active HSRP is in one chassis and the active CSM-S is in another chassis, traffic traverses through the trunk ports between the two chassis.

You can configure tracking to track the state of HSRP groups, physical interfaces, and gateways.

Tracking an HSRP Group

You can configure HSRP group tracking so that when the HSRP state changes for a specified tracked group, the Cisco IOS software sends a message to the CSM-S to make an active switchover.

To configure HSRP group tracking, perform this task in the fault-tolerant submode:

Command	Purpose
`Router(config-slb-ft)# track group` *group_number*	Specifies the tracked HSRP group.

Tracking a Gateway

When you configure gateway tracking, the Cisco IOS software sends the configured gateway IP address and next hop IP address to the CSM-S. The CSM-S then periodically checks for the availability of the gateway. If the gateway is not available, the CSM-S forces an active switchover.

To configure gateway tracking, perform this task in the fault-tolerant submode:

Command	Purpose
`Router(config-slb-ft)# track gateway` *ip_addr*	Specifies the tracked gateway IP address.

Tracking an Interface

You can configure interface tracking so that when the specified physical interface goes down, the Cisco IOS software sends a message to the CSM-S to make an active switchover.

To configure interface tracking, perform this task in the fault-tolerant submode:

Command	Purpose
`Router(config-slb-ft)# track interface {async | ctunnel | dialer | fastethernet | gigabitethernet}`	Specifies the tracked interface.

Configure the Tracking Mode

To configure the tracking mode, perform this task in the fault-tolerant submode:

Command	Purpose
`Router(config-slb-ft)# track mode {any | all}`	Specifies the tracking mode. The **any** keyword forces a switchover if any of the tracking devices goes down or if the HSRP state changes to standby. The **all** keyword forces a switchover when at least one of the tracking devices goes down for every configured tracking feature (group, gateway, and interface).

Configuring Connection Redundancy

Connection redundancy prevents open connections from ceasing to respond when the active CSM-S fails and the standby CSM-S becomes active. With connection redundancy, the active CSM-S replicates forwarding information to the standby CSM-S for each connection that is to remain open when the active CSM-S fails over to the standby CSM-S.

To configure connection redundancy, perform this task:

	Command	Purpose
Step 1	`Router# configure terminal`	Enters router configuration mode.
Step 2	`Router(config)# no ip igmp snooping`	Removes IGMP snooping from the configuration.
Step 3	`Router(config-module-csm)# vserver virtserver-name`	Identifies a virtual server and enters the virtual server submode.
Step 4	`Router(config-slb-vserver)# virtual ip-address [ip-mask] protocol port-number [service ftp]`	Configures the virtual server attributes.
Step 5	`Router(config-slb-vserver)# serverfarm serverfarm-name`	Associates a server farm with a virtual server.
Step 6	`Router(config-slb-vserver)# sticky duration [group group-id] [netmask ip-netmask]`	Ensures that connections from the same client use the same real server.
Step 7	`Router(config-slb-vserver)# replicate csrp sticky`	Enables sticky replication.
Step 8	`Router(config-slb-vserver)# replicate csrp connection`	Enables connection replication.
Step 9	`Router(config-slb-vserver)# inservice`	Enables the virtual server for load balancing.
Step 10	`Router(config-module-csm)# ft group group-id vlan vlanid`	Configures fault tolerance and enters the fault-tolerance submode.
Step 11	`Router(config-slb-ft)# priority value`	Sets the priority of the CSM-S.
Step 12	`Router(config-slb-ft)# failover failover-time`	Sets the time for a standby CSM-S to wait before becoming an active CSM-S.
Step 13	`Router(config-slb-ft)# preempt`	Allows a higher priority CSM-S to take control of a fault-tolerant group when it comes online.

This example shows how to set fault tolerance for connection redundancy:

```
Router(config-module-csm)# vserver VS_LINUX-TELNET
Router(config-slb-vserver)# virtual 10.6.0.100 tcp telnet
Router(config-slb-vserver)# serverfarm SF_NONAT
Router(config-slb-vserver)# sticky 100 group 35
Router(config-slb-vserver)# replicate csrp sticky
Router(config-slb-vserver)# replicate csrp connection
Router(config-slb-vserver)# inservice
Router(config-slb-vserver)# exit
Router(config-module-csm)# ft group 90 vlan 111
Router(config-slb-ft)# priority 10
Router(config-slb-ft)# failover 3
Router(config-slb-ft)# preempt
Router(config-slb-ft)# exit
```

Synchronizing the Configuration

You can synchronize the configuration between the active and standby CSM-S in a single chassis or in separate chassis. Synchronization happens over the fault-tolerant VLAN.

Note Traffic over the fault-tolerant VLAN uses broadcast packets; therefore, we recommend that you remove all devices, other than those necessary for communication between the active and standby CSM-S, from the fault-tolerant VLAN.

Note It is important that you follow the procedures in this section as described. If you do not enter the **alt** *standby_ip_address* command on the active CSM-S (as described in Step 4 below) before you synchronize the configuration, the VLAN IP addresses on the standby CSM-S will be removed.

To configure synchronization on the active CSM-S, perform this task:

	Command	Purpose
Step 1	`Router# configure terminal`	Enters router configuration mode.
Step 2	`Router(config)# module csm slot-number`	Specifies the slot number of the active CSM-S.
Step 3	`Router(config-module-csm)# vlan vlan_ID {client | server}`	Configures the client-side and server-side VLAN.
Step 4	`Router(config-slb-vlan-client)# ip addr active_ip_addr netmask alt standby_ip_addr netmask`	Configures an IP address to the CSM-S on this particular VLAN. Enter the **alt** keyword to specify an alternate IP address that is sent to the standby CSM-S. **Note** If you do not enter the **alt** *standby_ip_address* command on the active CSM-S before you synchronize the configuration, the VLAN IP addresses on the backup CSM-S will be removed
Step 5	`Router(config-slb-vlan-client)# exit`	Exits VLAN config mode.
Step 6	`Router(config-module-csm)# ft group group-id vlan vlanid`	Configures fault tolerance and enters the fault-tolerance submode.
Step 7	`Router(config-slb-ft)# priority active_value alt standby_value`	Sets the priority of the CSM-S. Enter the **alt** keyword to specify an alternate priority value that is sent to the standby CSM-S.

To configure synchronization on the standby CSM-S, perform this task:

	Command	Purpose
Step 1	`Router# configure terminal`	Enters router configuration mode.
Step 2	`Router(config)# module csm slot-number`	Specifies the slot number of the standby CSM-S.
Step 3	`Router(config-module-csm)# ft group group-id vlan vlanid`	Configures fault tolerance and specifies the fault tolerant VLAN.

To synchronize the configuration, perform this task on the active CSM-S:

Command	Purpose
Router# **hw-module csm** *slot-number* **standby config-sync**	Synchronizes the configuration. Enter this command every time you want to synchronize the configuration.

This example shows how to configure both the active and the standby CSM-S for synchronization:

- Active CSM-S:

```
Router# configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
Router(config)# module csm 5
Router(config-module-csm)# vlan 130 client
Router(config-slb-vlan-client)# ip addr 123.44.50.5 255.255.255.0 alt 123.44.50.7
255.255.255.0
Router(config-slb-vlan-client)# gateway 123.44.50.1
Router(config-slb-vlan-client)# exit
Router(config-module-csm)# vlan 150 server
Router(config-slb-vlan-server)# ip addr 123.46.50.6 255.255.255.0 alt 123.44.40.8
255.255.255.0
Router(config-slb-vlan-server)# alias 123.60.7.6 255.255.255.0
Router(config-slb-vlan-server)# route 123.50.0.0 255.255.0.0 gateway 123.44.50.1
Router(config-slb-vlan-server)# exit
Router(config-module-csm)# ft group 90 vlan 111
Router(config-slb-ft)# priority 10 alt 15
Router(config-slb-ft)# end
```

- Standby CSM-S:

```
Router# configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
Router(config)# module csm 6
Router(config-module-csm)# ft group 90 vlan 111
Router(config-slb-ft)# end
```

This example shows how to synchronize the configuration between the active and standby CSM-Ss:

```
Router# hw-module csm 5 standby config-sync
%CSM_SLB-6-REDUNDANCY_INFO:Module 5 FT info:Active:Bulk sync started
%CSM_SLB-6-REDUNDANCY_INFO:Module 5 FT info:Active:Manual bulk sync completed
```

Configuring a Hitless Upgrade

A hitless upgrade allows you to upgrade to a new software version without major service disruption due to the downtime for the upgrade. To configure a hitless upgrade, perform these steps:

Step 1 If you have preempt enabled, turn it off.

Step 2 Perform a write memory on the standby CSM-S.

Step 3 Upgrade the standby CSM-S with the new release, and then reboot the CSM-S.

The standby CSM-S boots as standby with the new release. If you have sticky backup enabled, keep the standby CSM-S in standby mode for at least 5 minutes.

Step 4 Upgrade the active CSM-S.

Step 5 Reboot the active CSM-S.

When the active CSM-S reboots, the standby CSM-S becomes the new active CSM-S and takes over the service responsibility.

Step 6 The rebooted CSM-S comes up as the standby CSM.

Configuring Additional Features and Options

This chapter describes how to configure content switching and contains these sections:

Configuring Session Persistence (Stickiness)

Session persistence (or stickiness) refers to the functionality of sending multiple (simultaneous or subsequent) connections from the same client consistently to the same server. This is a typical requirement in certain load-balancing environments.

Complete application transactions (such as browsing a website, selecting various items for purchase, and then checking out) typically require multiple—sometimes hundreds or thousands—simultaneous or subsequent connections. Most of these transactions generate and require temporary critical information. This information is stored and modified on the specific server that is handling the transaction. For the entire duration of the transaction, which may take from minutes to hours, the client has to be consistently sent to the same server.

Multi-tier designs with a back-end shared database partially remove the problem, but a good stickiness solution improves the performance of the application by relying on the local server cache. Using the local server cache removes the requirement to connect to the database and get the transaction-specific information each time that a new server is selected.

Uniquely identifying a client across multiple connections is the most difficult part of the stickiness problem. Whatever might be the key information used to recognize and identify a client, the load-balancing device must store that information and associate it with the server that is currently processing the transaction.

Note The CSM-S can maintain a sticky database of 256.000 entries.

The CSM-S can uniquely identify the clients and perform stickiness with the following methods:

- Source IP address stickiness

  The CSM-S can be configured to learn the entire source IP address (with a netmask of 32 bits) or just a portion of it.

- SSL identification stickiness

  When the client and servers are communicating over SSL, they maintain a unique SSL identification number across multiple connections. SSL version 3.0 or TLS 1.0 specify that this identification number must be carried in clear text. The CSM-S can use this value to identify a specific transaction, but because this SSL ID can be renegotiated, it is not always possible to preserve stickiness to the correct server. SSL ID-based stickiness is used to improve performance of SSL termination devices by consistently allowing SSL ID reuse.

  Note When the CSM-S is used with the Catalyst 6500 SSL Module, SSL ID stickiness across SSL ID renegotiation is possible because each Catalyst 6500 SSL Module inserts its MAC address within the SSL ID, at a specific offset. This is configured through the **ssl-sticky** command under the virtual server configuration submode.

  Refer to the *Catalyst 6500 Series Switch SSL Services Module Configuration Note,* Chapter 5 “Configuring Different Modes of Operation” for sticky connection configuration information.

  Refer to the *Catalyst 6500 Series Switch Content Switching Module Command Reference* for information about the **ssl-sticky** command.

- Dynamic cookie learning

  The CSM-S can be configured to look for a specific cookie name and automatically learn its value either from the client request HTTP header or from the server “set cookie” message.

  By default, the entire cookie value is learned by the CSM-S. This feature has been enhanced in CSM-S software release 4.1.(1) by introducing an optional offset and length to instruct the CSM-S to only learn a portion of the cookie value. See the “Cookie Sticky Offset and Length” section on page 10-4.

  Dynamic cookie learning is useful when dealing with applications that store more than just the session ID or user ID within the same cookie. Only specific bytes of the cookie value are relevant to stickiness.

  CSM-S software release 4.1(1) also added the dynamic cookie stickiness feature that has the capability to search for (and eventually learn or stick to) the cookie information as part of the URL. (See the “URL-Learn” section on page 10-4.) URL learning is useful with applications that insert cookie information as part of the HTTP URL. In some cases, this feature can be used to work around clients that reject cookies.

- Cookie insert

  The CSM-S inserts the cookie on behalf of the server, so that cookie stickiness can be performed even when the servers are not configured to set cookies. The cookie contains information that the CSM-S uses to ensure persistence to a specific real server.

Configuring Sticky Groups

Configuring a sticky group involves configuring the sticky method (source IP, SSL ID, cookie) and parameters of that group and associating it with a policy. The sticky timeout specifies the period of time that the sticky information is kept in the sticky tables. The default sticky timeout value is 1440 minutes (24 hours). The sticky timer for a specific entry is reset each time that a new connection matching that entry is opened.

The sticky timer for a specific entry is reset from the point where the last session ends. This timeout policy applies to sessions using IP_Sticky only. Sessions using other forms of persistence (for example, cookie and url-hash) are not affected by this behavior.

Use this command to configure the sticky environment variable:

```
Router(config-module-csm)# variable NO_TIMEOUT_IP_STICKY_ENTRIES 1
```

Note Multiple policies or virtual servers potentially can be configured with the same sticky group. In that case, the stickiness behavior applies to all connections to any of those policies or virtual servers. These connections are also referred to as "buddy connections" because a client stuck to server A through policy or virtual server 1 also will be stuck to the same server A through policy or virtual server 2, if both policy or virtual server 1 and 2 are configured with the same sticky group.

Caution When using the same sticky group under multiple policies or virtual servers, it is important to make sure that all are using the same server farm or a different server farm with the same servers in it.

To configure sticky groups, perform this task:

Command	Purpose
Router(config-module-csm)# **sticky** *sticky-group-id* {**netmask** *netmask* | **cookie** *name* | **ssl**} [**address** [**source** | **destination** | **both**]][**timeout** *sticky-time*]	Ensures that connections from the same client matching the same policy use the same real server[1].

1. The **no** form of this command restores the defaults.

This example shows how to configure a sticky group and associate it with a policy:

```
Router(config-module-csm)# sticky 1 cookie foo timeout 100
Router(config-module-csm)# serverfarm pl_stick
Router(config-slb-sfarm)# real 10.8.0.18
Router(config-slb-real)# inservice
Router(config-slb-sfarm)# real 10.8.0.19
Router(config-slb-real)# inservice
Router(config-slb-real)# exit
Router(config-slb-sfarm)# exit
Router(config-module-csm)# policy policy_sticky_ck
Router(config-slb-policy)# serverfarm pl_stick
Router(config-slb-policy)# sticky-group 1
Router(config-slb-policy)# exit
Router(config-module-csm)# vserver vs_sticky_ck
Router(config-slb-vserver)# virtual 10.8.0.125 tcp 90
Router(config-slb-vserver)# slb-policy policy_sticky_ck
Router(config-slb-vserver)# inservice
Router(config-slb-vserver)# exit
```

Cookie Insert

Use cookie insert when you want to use a session cookie for persistence if the server is not currently setting the appropriate cookie. With this feature enabled, the CSM-S inserts the cookie in the response to the server from the client. The CSM-S then inserts a cookie in traffic flows from a server to the client.

This example shows how to specify a cookie for persistence:

```
Cat6k-2(config-module-csm)# sticky 5 cookie mycookie insert
```

Cookie Sticky Offset and Length

The cookie value may change with only a portion remaining constant throughout a transaction between the client and a server. The constant portion may be used to make persistent connections back to a specific server. To stick or maintain the persistence of that connection, you can specify the portion of the cookie that remains constant with the offset and length values of a cookie in the **cookie offset** *num* [**length** *num*] command.

You specify the offset in bytes, counting from the first byte of the cookie value and the length (also in bytes) that specifies the portion of the cookie that you are using to maintain the sticky connection. These values are stored in the sticky tables.

The offset and length can vary from 0 to 4000 bytes. If the cookie value is longer than the offset but shorter than the offset plus the length of the cookie, the CSM-S sticks the connection based on that portion of the cookie after the offset.

This example shows how to specify set the cookie offset and length:

```
Cat6k-1# configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
Cat6k-1(config)# module csm 4
Cat6k-1(config-module-csm)# sticky 20 cookie SESSION_ID
Cat(config-slb-sticky-cookie)# cookie offset 10 length 6
```

URL-Learn

The URL-learn cookie sticky feature allows the CSM-S to capture the session information of the set-cookie field or cookies embedded in URLs. The CSM-S creates a sticky table entry based on the value of a specified cookie embedded in the set-cookie HTTP header of the server's response.

When URL-learn is configured, the CSM-S can learn the cookie value in three different ways:

- Cookie message in the server to client direction
- Cookie in a client request
- Cookie value embedded in the URL

The behaviors in the first two bullets are already supported by the standard dynamic cookie learning feature, and the behavior in the last bullet is added with the URL-learn feature.

In most cases, the client then returns the same cookie value in a subsequent HTTP request. The CSM-S sticks the client to the same server based on that matching value. Some clients, however, disable cookies in their browser making this type of cookie sticky connection impossible. With the new URL cookie learn feature, the CSM-S can extract the cookie name and value embedded in the URL string. This feature works only if the server has embedded the cookie into the URL link in the web page.

If the client's request does not carry a cookie, the CSM-S looks for the session ID string (?session-id=) configured on the CSM-S. The value associated with this string is the session ID number that the CSM-S looks for in the cache. The session ID is matched with the server where the requested information is located and the client's request is sent.

Because the session cookie and the URL session ID may be different, the Cisco IOS **sticky** *id* **cookie** *name* command was updated. The example in this section shows the correct syntax.

Note The offset and length clauses were included in this updated command to support the cookie sticky offset feature in this release. See the "Cookie Sticky Offset and Length" section on page 10-4.

Depending on client and server behavior and the sequence of frames, the same cookie value may appear in the standard HTTP cookies appearing in the HTTP cookie, set-cookie headers, or cookies embedded in URLs. The name of a cookie may be different from the URL depending on whether the cookie is embedded in a URL or appears in an HTTP cookie header. The use of a different name for the cookie and the URL occurs because these two parameters are configurable on the server and are very often set differently. For example, the set-cookie name might be as follows:

```
Set-Cookie: session_cookie = 123
```

The URL might be as follows:

```
http://www.example.com/?session-id=123
```

The *name* field in the **sticky** command specifies the cookie name that appears in the cookie headers. The **secondary** *session_id* clause added to this command specifies the corresponding cookie name that appears in the URL.

This example shows how to configure the URL learning feature:

```
Cat6k-1# configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
Cat6k-1(config)# module csm 4
Cat6k-1(config-module-csm)# sticky 30 cookie session_cookie
Cat(config-slb-sticky-cookie)# cookie secondary session-id
Cat(config-slb-sticky-cookie)#
```

Configuring Route Health Injection

These sections describe how to configure route health injection (RHI):

Understanding RHI

These sections describe the RHI:

RHI Overview

RHI allows the CSM-S to advertise the availability of a VIP address throughout the network. Multiple CSM-S devices with identical VIP addresses and services can exist throughout the network. One CSM-S can override the server load-balancing services over the other devices if the services are no longer available on the other devices. One CSM-S also can provide the services because it is logically closer to the client systems than other server load-balancing devices.

Note RHI is restricted to intranets because the CSM-S advertises the VIP address as a host route and most routers do not propagate the host-route information to the Internet.

To enable RHI, configure the CSM-S to do the following:

- Probe real servers and identify available virtual servers and VIP addresses
- Advertise accurate VIP address availability information to the MSFC whenever a change occurs

Note At startup with RHI enabled, the CSM-S sends a message to the MSFC as each VIP address becomes available.

The MSFC periodically propagates the VIP address availability information that RHI provides.

Note RHI is normally restricted to intranets; for security reasons, most routers do not propagate host-route information to the Internet.

Routing to VIP Addresses Without RHI

Without RHI, traffic reaches the VIP address by following a route to the client VLAN to which the VIP address belongs. When the CSM-S starts up, the MSFC creates routes to client VLANs in its routing table and shares this route information with other routers. To reach the VIP, the client systems rely on the router to send the requests to the network subnet address where the individual VIP address lives.

If the subnet or segment is reachable but the virtual servers on the CSM-S at this location are not operating, the requests fail. Other CSM-S devices can be at different locations. However, the routers send the requests based on the logical distance to the subnet only.

Without RHI, traffic is sent to the VIP address without any verification that the VIP address is available. The real servers attached to the VIP might not be active.

Note By default, the CSM-S will not advertise the configured VIP addresses.

Routing to VIP Addresses With RHI

With RHI, the CSM-S sends advertisements to the MSFC when VIP addresses become available and withdraws advertisements for VIP addresses that are no longer available. The router looks in the routing table to find the path information it needs to send the request from the client to the VIP address. When the RHI feature is turned on, the advertised VIP address information is the most specific match. The request for the client is sent through the path where it reaches the CSM-S with active VIP services.

When multiple instances of a VIP address exist, a client router receives the information it needs (availability and hop count) for each instance of a VIP address, allowing it to determine the best available route to that VIP address. The router chooses the path where the CSM-S is logically closer to the client system.

Note With RHI, you must also configure probes because the CSM-S determines if it can reach a given VIP address by probing all the real servers that serve its content. After determining if it can reach a VIP address, the CSM-S shares this availability information with the MSFC. The MSFC, in turn, propagates this VIP availability information to the rest of the intranet.

Understanding How the CSM-S Determines VIP Availability

For the CSM-S to determine if a VIP is available, you must configure a probe (HTTP, ICMP, Telnet, TCP, FTP, SMTP, or DNS) and associate it with a server farm. When probes are configured, the CSM-S performs these checks:

- Probes all real servers on all server farms configured for probing
- Identifies server farms that are reachable (have at least one reachable real server)
- Identifies virtual servers that are reachable (have at least one reachable server farm)
- Identifies VIPs that are reachable (have at least one reachable virtual server)

Understanding Propagation of VIP Availability Information

With RHI, the CSM-S sends advertisement messages to the MSFC containing the available VIP addresses. The MSFC adds an entry in its routing table for each VIP address it receives from the CSM-S. The routing protocol running on the MSFC sends routing table updates to other routers. When a VIP address becomes unavailable, its route is no longer advertised, the entry times out, and the routing protocol propagates the change.

Note For RHI to work on the CSM-S, the MSFC in the chassis in which the CSM-S resides must run Cisco IOS Release 12.1.7(E) or later releases and must be configured as the client-side router.

Configuring RHI for Virtual Servers

To configure RHI for the virtual servers, perform these steps:

Step 1 Verify that you have configured the VLANs. See Chapter 4, “Configuring VLANs.”

Step 2 Associate the probe with a server farm. See the “Configuring Probes for Health Monitoring” section on page 11-1.

Step 3 Configure the CSM-S to probe real servers. See the "Configuring Probes for Health Monitoring" section on page 11-1.

Step 4 Enter the **advertise active** SLB virtual server command to enable RHI for each virtual server:

```
Router(config-module-csm)# vserver virtual_server_name
Router(config-slb-vserver)# advertise active
```

This example shows how to enable RHI for the virtual server named vserver1:

```
Router(config-module-csm)# vserver vserver1
Router(config-slb-vserver)# advertise active
```

Environmental Variables

You can enable the environmental variables in the configuration with the **variable** *name string* command. Table 10-1 describes the CSM-S environmental values.

Table 10-1 CSM-S Environmental Values

Name	Default	Valid Values	Description
ARP_INTERVAL	300	Integer (15 to 31536000)	Time (in seconds) between ARP requests for configured hosts.
ARP_LEARNED_INTERVAL	14400	Integer (60 to 31536000)	Time (in seconds) between ARP requests for learned hosts.
ARP_GRATUITOUS_INTERVAL	15	Integer (10 to 31536000)	Time (in seconds) between gratuitous ARP requests.
ARP_RATE	10	Integer (1 to 60)	Seconds between ARP retries.
ARP_RETRIES	3	Integer (2 to 15)	Count of ARP attempts before flagging a host as down.
ARP_LEARN_MODE	1	Integer (0 to 1)	Indicates whether the CSM-S learns MAC addresses on responses only (0) or all traffic (1).
ARP_REPLY_FOR_NO_INSERVICE_VIP	D	0	Integer (0 to 1).
ADVERTISE_RHI_FREQ	10	Integer (1 to 65535)	Frequency in second(s) that the CSM-S uses to check for RHI updates.
AGGREGATE_BACKUP_SF_STATE_TO_VS	0	Integer (0 to 1)	Specifies whether to include the operational state of a backup server farm into the state of a virtual server.
COOKIE_INSERT_EXPIRATION_DATE	Fri, 1 Jan 2010 01:01:50 GMT	String (2 to 63 chars)	Configures the expiration time and date for the HTTP cookie inserted by the CSM-S.

Table 10-1 CSM-S Environmental Values (continued)

Name	Default	Valid Values	Description
DEST_UNREACHABLE_MASK	65535	Integer (0 to 65535)	Bitmask defining which ICMP destination unreachable codes are to be forwarded.
FT_FLOW_REFRESH_INT	60	Integer (1 to 65535)	Interval for the fault-tolerant slow path flow refresh in seconds.
HTTP_CASE_SENSITIVE_MATCHING	1	Integer (0 to 1)	Specifies whether the URL (cookie, header) matching and sticky are to be case sensitive.
HTTP_URL_COOKIE_DELIMITERS	/?&#+	String (1 to 64 chars)	Configures the list of delimiter characters for cookies in the URL string.
MAX_PARSE_LEN_MULTIPLIER	1	Integer (1 to 16)	Multiplies the configured max-parse-len by this amount.
NAT_CLIENT_HASH_SOURCE_PORT	0	Integer (0 to 1)	Specifies whether to use the source port to pick client NAT IP address.
ROUTE_UNKNOWN_FLOW_PKTS	0	Integer (0 to 1)	Specifies whether to route non-SYN packets that do not match any existing flows.
NO_RESET_UNIDIRECTIONAL_FLOWS	0	Integer (0 to 1)	Specifies, if set, that unidirectional flows do not be reset when timed out.
SWITCHOVER_RP_ACTION	0	Integer (0 to 1)	Specifies whether to recover (0) or halt/reboot (1) after a supervisor engine route processor switchover occurs.
SWITCHOVER_SP_ACTION	0	Integer (0 to 1)	Specifies whether to recover (0) or halt/reboot (1) after a supervisor engine switch processor switchover occurs.
SYN_COOKIE_INTERVAL	3	Integer (1 to 60)	Specifies the interval, in seconds, at which a new syn-cookie key is generated.
SYN_COOKIE_THRESHOLD	5000	Integer (0 to 1048576)	Specifies the threshold (in number of pending sessions) at which syn-cookie is engaged.
TCP_MSS_OPTION	1460	Integer (1 to 65535)	Specifies the maximum segment size (MSS) value sent by CSM-S for Layer 7 processing.
TCP_WND_SIZE_OPTION	8192	Integer (1 to 65535)	Specifies the window size value sent by CSM-S for Layer 7 processing.

Table 10-1 CSM-S Environmental Values (continued)

Name	Default	Valid Values	Description
VSERVER_ICMP_ALWAYS_RESPOND	false	String (1 to 5 chars)	If "true," respond to ICMP probes regardless of virtual server state.
XML_CONFIG_AUTH_TYPE	Basic	String (5 to 6 chars)	Specifies the HTTP authentication type for xml-config: Basic or Digest.

This example shows how to display the environmental variables in the configuration:

```
Router# show mod csm 5 variable

variable                         value
-----------------------------------------------------------------
ARP_INTERVAL                     300
ARP_LEARNED_INTERVAL             14400
ARP_GRATUITOUS_INTERVAL          15
ARP_RATE                         10
ARP_RETRIES                      3
ARP_LEARN_MODE                   1
ARP_REPLY_FOR_NO_INSERVICE_VIP   0
ADVERTISE_RHI_FREQ               10
AGGREGATE_BACKUP_SF_STATE_TO_VS  0
DEST_UNREACHABLE_MASK            0xffff
FT_FLOW_REFRESH_INT              60
GSLB_LICENSE_KEY                 (no valid license)
HTTP_CASE_SENSITIVE_MATCHING     1
MAX_PARSE_LEN_MULTIPLIER         1
NAT_CLIENT_HASH_SOURCE_PORT      0
ROUTE_UNKNOWN_FLOW_PKTS          0
NO_RESET_UNIDIRECTIONAL_FLOWS    0
SYN_COOKIE_INTERVAL              3
SYN_COOKIE_THRESHOLD             5000
TCP_MSS_OPTION                   1460
TCP_WND_SIZE_OPTION              8192
VSERVER_ICMP_ALWAYS_RESPOND      false
XML_CONFIG_AUTH_TYPE             Basic
Cat6k-2#
```

To display all information for the current set of environmental variables in the configuration, use the **show module csm** *slot* **variable** [**detail**] command as follows:

```
Cat6k-2# show mod csm 5 variable detail
Name:ARP_INTERVAL  Rights:RW
Value:300
Default:300
Valid values:Integer (15 to 31536000)
Description:
Time (in seconds) between ARPs for configured hosts

Name:ARP_LEARNED_INTERVAL  Rights:RW
Value:14400
Default:14400
Valid values:Integer (60 to 31536000)
Description:
Time (in seconds) between ARPs for learned hosts

Name:ARP_GRATUITOUS_INTERVAL  Rights:RW
Value:15
Default:15
```

```
Valid values:Integer (10 to 31536000)
Description:
Time (in seconds) between gratuitous ARPs

Name:ARP_RATE  Rights:RW
Value:10
Default:10
Valid values:Integer (1 to 60)
Description:
Seconds between ARP retries

Name:ARP_RETRIES  Rights:RW
Value:3
Default:3
Valid values:Integer (2 to 15)
Description:
Count of ARP attempts before flagging a host as down

Name:ARP_LEARN_MODE  Rights:RW
Value:1
Default:1
Valid values:Integer (0 to 1)
Description:
Indicates whether CSM-S learns MAC address on responses only (0) or all traffic (1)

Name:ARP_REPLY_FOR_NO_INSERVICE_VIP  Rights:RW
Value:0
Default:0
Valid values:Integer (0 to 1)
Description:
Whether the CSM-S would reply to ARP for out-of-service vserver

Name:ADVERTISE_RHI_FREQ  Rights:RW
Value:10
Default:10
Valid values:Integer (1 to 65535)
Description:
The frequency in second(s) the CSM-S will check for RHI updates

Name:AGGREGATE_BACKUP_SF_STATE_TO_VS  Rights:RW
Value:0
Default:0
Valid values:Integer (0 to 1)
Description:
Whether to include the operational state of a backup serverfarm into the state of a
virtual server

Name:DEST_UNREACHABLE_MASK  Rights:RW
Value:0xffff
Default:65535
Valid values:Integer (0 to 65535)
Description:
Bitmask defining which ICMP destination unreachable codes are to be forwarded

Name:FT_FLOW_REFRESH_INT  Rights:RW
Value:60
Default:60
Valid values:Integer (1 to 65535)
Description:
FT slowpath flow refresh interval in seconds

Name:GSLB_LICENSE_KEY  Rights:RW
Value:(no valid license)
Default:(no valid license)
```

```
Valid values:String (1 to 63 chars)
Description:
License key string to enable GSLB feature

Name:HTTP_CASE_SENSITIVE_MATCHING  Rights:RW
Value:1
Default:1
Valid values:Integer (0 to 1)
Description:
Whether the URL (Cookie, Header) matching and sticky to be case sensitive

Name:MAX_PARSE_LEN_MULTIPLIER  Rights:RW
Value:1
Default:1
Valid values:Integer (1 to 16)
Description:
Multiply the configured max-parse-len by this amount

Name:NAT_CLIENT_HASH_SOURCE_PORT  Rights:RW
Value:0
Default:0
Valid values:Integer (0 to 1)
Description:
Whether to use the source port to pick client NAT IP address

Name:ROUTE_UNKNOWN_FLOW_PKTS  Rights:RW
Value:0
Default:0
Valid values:Integer (0 to 1)
Description:
Whether to route non-SYN packets that do not matched any existing flows

Name:NO_RESET_UNIDIRECTIONAL_FLOWS  Rights:RW
Value:0
Default:0
Valid values:Integer (0 to 1)
Description:
If set, unidirectional flows will not be reset when timed out

Name:SYN_COOKIE_INTERVAL  Rights:RW
Value:3
Default:3
Valid values:Integer (1 to 60)
Description:
The interval, in seconds, at which a new syn-cookie key is generated

Name:SYN_COOKIE_THRESHOLD  Rights:RW
Value:5000
Default:5000
Valid values:Integer (0 to 1048576)
Description:
The threshold (in number of pending sessions) at which syn-cookie is engaged

Name:TCP_MSS_OPTION  Rights:RW
Value:1460
Default:1460
Valid values:Integer (1 to 65535)
Description:
Maximum Segment Size (MSS) value sent by CSM-S for L7 processing

Name:TCP_WND_SIZE_OPTION  Rights:RW
Value:8192
Default:8192
Valid values:Integer (1 to 65535)
```

```
Description:
Window Size value sent by CSM-S for L7 processing

Name:VSERVER_ICMP_ALWAYS_RESPOND  Rights:RW
Value:false
Default:false
Valid values:String (1 to 5 chars)
Description:
If "true" respond to ICMP probes regardless of vserver state

Name:XML_CONFIG_AUTH_TYPE  Rights:RW
Value:Basic
Default:Basic
Valid values:String (5 to 6 chars)
Description:
HTTP authentication type for xml-config:Basic or Digest
```

Configuring Persistent Connections

The CSM-S allows HTTP connections to be switched based on a URL, cookies, or other fields contained in the HTTP header. Persistent connection support in the CSM-S allows for each successive HTTP request in a persistent connection to be switched independently. As a new HTTP request arrives, it may be switched to the same server as the prior request, it may be switched to a different server, or it may be reset to the client preventing that request from being completed.

As of software release 2.1(1), the CSM-S supports HTTP 1.1 persistence. This feature allows browsers to send multiple HTTP requests on a single persistent connection. After a persistent connection is established, the server keeps the connection open for a configurable interval, anticipating that it may receive more requests from the same client. Persistent connections eliminate the overhead involved in establishing a new TCP connection for each request.

HTTP 1.1 persistence is enabled by default on all virtual servers configured with Layer 7 policies. To disable persistent connections, enter the **no persistent rebalance** command. To enable persistent connections, enter the **persistent rebalance** command.

This example shows how to configure persistent connections:

```
Router# configure terminal
Enter configuration commands, one per line.  End with
CNTL/Z.
Router(config)# mod csm 2
!!! configuring serverfarm
Router(config-module-csm)# serverfarm sf3
Router(config-slb-sfarm)# real 10.1.0.105
Router(config-slb-real)# inservice
!!! configuring vserver
Router(config-slb-real)# vserver vs3
Router(config-slb-vserver)# virtual 10.1.0.83 tcp 80
Router(config-slb-vserver)# persistent rebalance
Router(config-slb-vserver)# serverfarm sf3
Router(config-slb-vserver)# inservice
Router(config-slb-vserver)# end
```

HTTP Header Insert

The HTTP header insert feature provides the CSM-S with the ability to insert information, such as the client's IP address, into the HTTP header. This feature is useful in situations where the CSM-S is performing source NAT and the application on the server side still requires visibility to the original source IP.

The CSM-S can insert the source IP address from the client into the header in the client-to-server direction.

Use the **insert protocol http header** *name* **header-value** *value* command to insert information into the HTTP header.

- *name*—Literal name of the generic field in the HTTP header. The name is a string with a range from 1 to 63 characters.
- *value*—Specifies the literal header value string to insert in the request.

  You can also use the %is and %id special parameters for the header values. The %is value inserts the source IP into the HTTP header and the %id value inserts the destination IP into the header. Each special parameter may only be specified once per header map.

Note A header map may contain multiple insert headers. If you insert header values that are made of multiple keywords that include spaces, you must use double quotes around the entire expression.

When configuring HTTP header insert, you must use a header map and a policy. You cannot use the default policy for HTTP header insert to work.

This example shows how to specify header fields and values to search upon a request:

```
Cat6k-2(config-module-csm)# natpool TESTPOOL 10.10.110.200 10.10.110.210 netmask
255.255.255.0
!
Cat6k-2(config-module-csm)# map HEADER-INSERT header
Cat6k-2(config-slb-map-header)# insert protocol http header Source-IP header-value %is
Cat6k-2(config-slb-map-header)# insert protocol http header User-Agent header-value
"MyBrowser 1.0"
!
Cat6k-2(config-module-csm)# real SERVER1
Cat6k-2(config-slb-real)# address 10.10.110.10
Cat6k-2(config-slb-real)# inservice
Cat6k-2(config-module-csm)# real SERVER2
Cat6k-2(config-slb-real)# address 10.10.110.20
Cat6k-2(config-slb-real)# inservice
!
Cat6k-2(config-module-csm)# serverfarm FARM-B
Cat6k-2(config-slb-sfarm)# nat server
Cat6k-2(config-slb-sfarm)# nat client TESTPOOL
Cat6k-2(config-slb-real)# real name SERVER1
Cat6k-2(config-slb-real)# inservice
Cat6k-2(config-slb-real)# real name SERVER2
Cat6k-2(config-slb-real)# inservice
!
Cat6k-2(config-module-csm)# policy INSERT
Cat6k-2(config-slb-policy)# header-map HEADER-INSERT
Cat6k-2(config-slb-policy)# serverfarm FARM-B
!
Cat6k-2(config-module-csm)# vserver WEB
Cat6k-2(config-slb-vserver)# virtual 10.10.111.100 tcp www
```

```
Cat6k-2(config-slb-vserver)# persistent rebalance
Cat6k-2(config-slb-vserver)# slb-policy INSERT
Cat6k-2(config-slb-vserver)# inservice
```

Configuring Global Server Load Balancing

This section contains the CSM global server load-balancing (GSLB) advanced feature set option and instructions for its use. You should review the terms of the software license agreement in the "Licenses" section on page xxvi in the Preface and on the back of the title page carefully before using the advanced feature set option.

Note By downloading or installing the software, you are consenting to be bound by the license agreement. If you do not agree to all of the terms of this license, then do not download, install, or use the software.

Using the GSLB Advanced Feature Set Option

To enable GSLB, perform this task in privileged mode:

Command	Purpose
`Router# config t` `Router(config)# mod csm 5`	Enters the configuration mode and enters CSM-S configuration mode for the specific CSM-S (for example, module 5, as used here).
`Router(config-module-csm)# variable` *name value*	Enables GSLB by using the name and value provided as follows: Name= [1] Value=
`Router(config-module-csm)# exit` `Router (config)# write mem`	Exits CSM-S module configuration mode and saves the configuration changes.
`Router#:hw-module` *slot number* `reset`	Reboots your CSM-S to activate changes.

1. GSLB requires a separately purchased license. To purchase your GSLB license, contact your Cisco representative.

Table 10-2 lists the GSLB environmental values used by the CSM-S.

Table 10-2 GSLB Environmental Values

Name	Default	Valid Values	Description
GSLB_LICENSE_KEY	(no valid license)	String (1 to 63 chars)	License key string to enable GSLB feature.
GSLB_KALAP_UDP_PORT	5002	Integer (1 to 65535)	Specifies the GSLB KAL-AP UDP port number.
GSLB_KALAP_PROBE_FREQ	45	Integer (45 to 65535)	Specifies the frequency of the GSLB KAL-AP probes.
GSLB_KALAP_PROBE_RETRIES	3	Integer (1 to 65535)	Specifies the maximum retries for GSLB KAL-AP probes.

Table 10-2 GSLB Environmental Values (continued)

Name	Default	Valid Values	Description
GSLB_ICMP_PROBE_FREQ	45	Integer (45 to 65535)	Specifies the frequency of the GSLB ICMP probes.
GSLB_ICMP_PROBE_RETRIES	3	Integer (1 to 65535)	Specifies the maximum retries for GSLB ICMP probes.
GSLB_HTTP_PROBE_FREQ	45	Integer (45 to 65535)	Specifies the frequency of the GSLB HTTP probes.
GSLB_HTTP_PROBE_RETRIES	3	Integer (1 to 65535)	Specifies the maximum retries for the GSLB HTTP probes.
GSLB_DNS_PROBE_FREQ	45	Integer (45 to 65535)	Specifies the frequency of the GSLB DNS probes.
GSLB_DNS_PROBE_RETRIES	3	Integer (1 to 65535)	Specifies the maximum retries for GSLB DNS probes.

Configuring GSLB

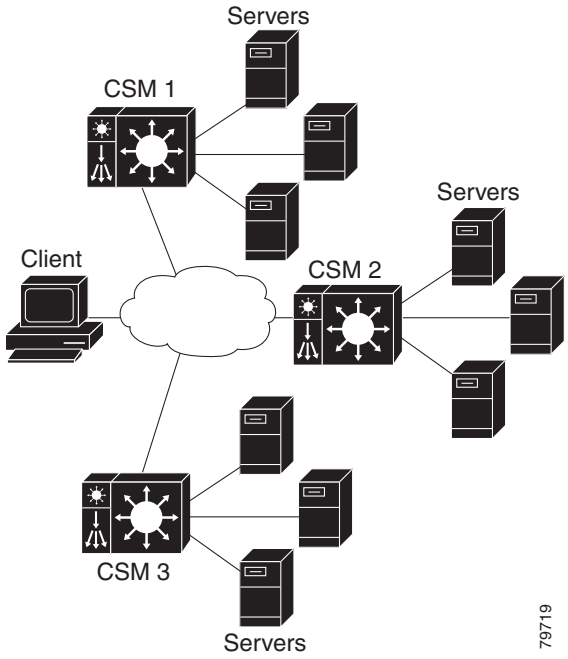
GSLB performs load balancing between multiple, dispersed hosting sites by directing client connections through DNS to different server farms and real servers based on load availability. GSLB is performed using access lists, maps, server farms, and load-balancing algorithms. Table 10-3 provides an overview of what is required for a GSLB configuration on the CSM-S.

Table 10-3 GSLB Operations

Client Request (From)	Domain (For)	Server farm (To)	Algorithm (Method)
Access lists can be used to filter incoming DNS requests, and policies are used to associate the configured maps, client groups, and server farms for incoming DNS requests.	A map is configured to specify the domain names that client requests must match. Regular expression syntax is supported. For example, domain names are cnn.com or yahoo.com that a client request must be matched against. If the domain name matches the specified map of a policy, the primary server farm is queried for a real server to respond to the request.	A server farm specifies a group of real servers where information is located that satisfies the client's request.	The GSLB probe is available for determining the availability of a target real server, using the probe type configured on the real server. GSLB server farm predictors are round-robin least load, ordered list, hash address source, hash domain, and hash domain address source.

Figure 10-1 shows a basic configuration for GSLB.

Figure 10-1 Global Server Load Balancing Configuration



In Figure 10-1, these guidelines apply to the configuration task and example:

- CSM-S 1 does both GSLB and SLB, while CSM-S 2 and CSM-S 3 only do SLB.
- CSM-S 1 has both a virtual server for SLB (where the real servers in the server farm are the IP addresses of the local servers) and a virtual server for GSLB.
- The DNS policy uses a primary server farm (where one of the real servers is local and the other two real servers are virtual servers configured on CSM-S 2 and CSM-S 3).
- Probes should be added for both the remote locations and the local real and virtual server.
- DNS requests sent to a CSM-S 1 management IP address (a CSM-S 1 VLAN address or alias IP) will receive as a response one of the three real server IPs configured in the server farm GSLBFARM.

To configure GSLB, perform this task:

	Command	Purpose
Step 1	`Router(config-slb-vserver)#` **`serverfarm`** *`serverfarm-name`*	Creates a server farm to associate with the virtual server.
Step 2	`Router(config-module-csm)#` **`vserver`** *`virtserver-name`*	Identifies a virtual server for SLB on CSM-S 1 and enters the virtual server submode.
Step 3	`Router(config-slb-vserver)#` **`virtual`** *`ip-address`* [*`ip-mask`*] *`protocol port-number`* [**`service ftp`**]	Configures the virtual server attributes.
Step 4	`Router(config-slb-vserver)#` **`inservice`**	Enables the virtual server for load balancing.
Step 5	`Router(config-module-csm)#` **`vserver`** *`virtserver-name`* **`dns`**	Identifies a virtual server for GSLB and enters the virtual server submode.

	Command	Purpose
Step 6	`Router(config-slb-vserver)# dns-policy [group group-id] [netmask ip-netmask]`	Ensures that connections from the same client use the same server farm.
Step 7	`Router(config-slb-vserver)# inservice`	Enables the virtual server for GSLB.
Step 8	`Router(config-module-csm)# serverfarm GSLBFARM dns-vip`	Creates and names the GSLBFARM server farm (which is actually a forwarding policy) and enters server farm configuration mode.
Step 9	`Router(config-slb-sfarm)# predictor hash address source`	Configures the hash address source for the load-balancing predictor for the server farm.
Step 10	`Router(config-module-csm)# real ip-addres`	Identifies the alias IP address of the real server and enters real server configuration submode.
Step 11	`Router(config-slb-real)# inservice`	Enables the virtual server for load balancing.
Step 12	`Router(config-module-csm)# map dns-map-name dns`	Configures a DNS map.
Step 13	`Router(config-dns-map)# match protocol dns domain name`	Adds a DNS name to the DNS map.
Step 14	`Router(config-module-csm)# policy policy name`	Configures a policy.
Step 15	`Router(config-slb-policy)# dns map map_name`	Adds the DNS map attribute to the policy.
Step 16	`Router(config-slb-policy)# serverfarm primary-serverfarm [backup sorry-serverfarm [sticky]]`	Associate the server farm with the policy.
Step 17	`Router(config-module-csm)# vserver virtserver-name`	Configures a virtual server on CSM-S 2 and enters the virtual server submode.
Step 18	`Router(config-slb-vserver)# virtual ip-address [ip-mask] protocol port-number [service ftp]`	Configures the virtual server attributes.
Step 19	`Router(config-slb-vserver)# serverfarm serverfarm-name`	Associates a server farm with the virtual server.
Step 20	`Router(config-slb-vserver)# inservice`	Enables the virtual server for load balancing.
Step 21	`Router(config-module-csm)# vserver virtserver-name`	Configures a virtual server on CSM-S 3 and enters the virtual server submode.
Step 22	`Router(config-slb-vserver)# virtual ip-address [ip-mask] protocol port-number [service ftp]`	Configures the virtual server attributes.
Step 23	`Router(config-slb-vserver)# serverfarm serverfarm-name`	Associates a server farm with the virtual server.
Step 24	`Router(config-slb-vserver)# inservice`	Enables the virtual server for load balancing.

This example shows how to configure GSLB:

On CSM1:

```
Router(config-module-csm)# serverfarm WEBFARM
Router(config-slb-sfarm)# predictor round-robin
Router(config-slb-sfarm)# real 3.5.5.5
Router(config-slb-real)# inservice
Router(config-slb-sfarm)# real 3.5.5.6
Router(config-slb-real)# inservice
Router(config-slb-real)# exit
Router(config-slb-sfarm)# exit

Router(config-module-csm)# vserver WEB
Router(config-slb-vserver)# virtual 10.10.10.10 tcp www
Router(config-slb-vserver)# serverfarm WEBFARM
Router(config-slb-vserver)# inservice

Router(config-module-csm)# serverfarm GSLBSERVERFARM dns-vip
Router(config-slb-sfarm)# predictor round-robine
Router(config-slb-sfarm)# real 10.10.10.10
Router(config-slb-real)# inservice
Router(config-slb-real)# exit
Router(config-slb-sfarm)# real 20.20.20.20
Router(config-slb-real)# inservice
Router(config-slb-real)# exit
Router(config-slb-sfarm)# real 30.30.30.30
Router(config-slb-real)# inservice
Router(config-slb-real)# exit

Router(config-module-csm)# map MAP1 dns
Router(config-dns-map)# match protocol dns domain foobar.com
Router(config-dns-map)# exit

Router(config-module-csm)# policy DNSPOLICY dns
Router(config-slb-policy)# dns map MAP1
Router(config-slb-policy)# serverfarm primary GSLBSERVERFARM ttl 20 responses 1
Router(config-slb-policy)# exit

Router(config-module-csm)# vserver DNSVSERVER dns
Router(config-slb-vserver)# dns-policy DNSPOLICY
Router(config-slb-vserver)# inservice
```

On CSM-S 2:

```
Router(config-module-csm)# serverfarm WEBFARM
Router(config-slb-sfarm)# predictor round-robin
Router(config-slb-sfarm)# real 4.5.5.5
Router(config-slb-real)# inservice
Router(config-slb-sfarm)# real 4.5.5.6
Router(config-slb-real)# inservice
Router(config-slb-real)# exit
Router(config-slb-sfarm)# exit

Router(config-module-csm)# vserver WEB
Router(config-slb-vserver)# virtual 20.20.20.20 tcp www
Router(config-slb-vserver)#s erverfarm WEBFARM
Router(config-slb-vserver)# inservice
```

On CSM-S 3:

```
Router(config-module-csm)# serverfarm WEBFARM
Router(config-slb-sfarm)# predictor round-robin
Router(config-slb-sfarm)# real 5.5.5.5
Router(config-slb-real)# inservice
```

```
Router(config-slb-sfarm)# real 5.5.5.6
Router(config-slb-real)# inservice
Router(config-slb-real)# exit
Router(config-slb-sfarm)# exit
Router(config-module-csm)# vserver WEB
Router(config-slb-vserver)# virtual 30.30.30.30 tcp www
Router(config-slb-vserver)# serverfarm WEBFARM
Router(config-slb-vserver)# inservice
```

Configuring Network Management

This section describes how to manage the CSM-S on the network and contains these sections:

Configuring SNMP Traps for Real Servers

When enabled, an SNMP trap is sent to an external management device each time that a real server changes its state (for example, each time that a server is taken in or out of service). The trap contains an object identifier (OID) that identifies it as a real server trap.

Note The real server trap OID is 1.3.6.1.4.1.9.9.161.2

The trap also contains a message describing the reason for the server state change.

Use the **snmp-server enable traps slb ft** command to enable or disable fault-tolerant traps associated with the SLB function of the Catalyst 6500 series switch. A fault-tolerant trap deals with the fault-tolerance aspects of SLB. For example, when fault-tolerant traps are enabled and the SLB device detects a failure in its fault-tolerant peer, it sends an SNMP trap as it transitions from standby to active.

To configure SNMP traps for real servers, perform this task:

	Command	Purpose
Step 1	`Router (config)# snmp-server community public`	Defines a password-like community string sent with the notification operation. The example string is **public**.
Step 2	`Router (config)# snmp-server host host-addr`	Defines the IP address of an external network management device to which traps are sent.
Step 3	`Router (config)# snmp-server enable traps slb csrp`	Enables SNMP traps for real servers[1].

1. The **no** form of this command disables the SNMP fault-tolerant traps feature.

Configuring the XML Interface

In previous releases, the only method available for configuring the CSM-S was the Cisco IOS CLI. With XML, you can configure the CSM-S using a Document Type Definition or DTD. (See Appendix D, "CSM XML Document Type Definition" for a sample of an XML DTD.)

These guidelines apply to XML for the CSM-S:

- Up to five concurrent client connections are allowed.
- The XML configuration is independent of the IP SLB mode with the following exception: The **csm_module slot='x' sense='no** command does have the desired effect and generates an XML error.
- Pipelined HTTP posts are not supported.
- There is a 30-second timeout for all client communication.
- Bad client credentials cause a message to be sent to the Cisco IOS system log.
- A single CSM-S can act as proxy for other CSM-S configurations by specifying a different slot attribute.

When you enable this feature, a network management device may connect to the CSM-S and send the new configurations to the device. The network management device sends configuration commands to the CSM-S using the standard HTTP protocol. The new configuration is applied by sending an XML document to the CSM-S in the data portion of an HTTP POST.

This example shows an HTTP conversation:

```
******** Client **************
POST /xml-config HTTP/1.1
Authorization: Basic VTpQ
Content-Length: 95

<?xml version="1.0"?>
<config><csm_module slot="4"><vserver name="FOO"/></csm_module></config>
******** Server **************
HTTP/1.1 200 OK
Content-Length: 21

<?xml version="1.0"?>
******** Client **************
POST /xml-config HTTP/1.1
Content-Length: 95

<?xml version="1.0"?>
<config><csm_module slot="4"><vserver name="FOO"/></csm_module></config>
******** Server **************
HTTP/1.1 401 Unauthorized
Connection: close
WWW-Authenticate: Basic realm=/xml-config
```

Table 10-4 lists the supported HTTP return codes.

Table 10-4 HTTP Return Codes for XML

Return Code	Description
200	OK
400	Bad Request
401	Unauthorized (credentials required, but not provided)
403	Forbidden (illegal credentials submitted; syslog also generated)
404	Not Found (“/xml-config” not specified)
408	Request Time-out (more than 30 seconds has passed waiting on receive)
411	Missing Content-Length (missing or zero Content-Length field)
500	Internal Server Error

Table 10-4 HTTP Return Codes for XML (continued)

Return Code	Description
501	Not Implemented (“POST” not specified)
505	HTTP Version Not Supported (“1.0” or “1.1” not specified)

These HTTP headers are supported:

- Content-Length (nonzero value required for all POSTs)
- Connection (*close* value indicates that a request should not be persistent)
- WWW-Authenticate (sent to client when credentials are required and missing)
- Authorization (sent from client to specify basic credentials in base 64 encoding)

For the XML feature to operate, the network management system must connect to a CSM-S IP address, not to a switch interface IP address.

Because the master copy of the configuration must be stored in Cisco IOS software, as it is with the CLI when XML configuration requests are received by the CSM-S, these requests must be sent to the supervisor engine.

Note XML configuration allows a single CSM-S to act as proxy for all the CSMs in the same switch chassis. For example, an XML page with configuration for one CSM-S may be successfully posted through a different CSM-S in the same switch chassis.

The Document Type Description (DTD), now publicly available, is the basis for XML configuration documents that you create. (See Appendix D, “CSM XML Document Type Definition.”) The XML documents are sent directly to the CSM-S in HTTP POST requests. To use XML, you must create a minimum configuration on the CSM-S in advance, using the Cisco IOS CLI. Refer to the *Catalyst 6500 Series Content Switching Module Command Reference* for information on the **xml-config** command.

The response is an XML document mirroring the request with troublesome elements flagged with child-error elements and with an error code and error string. You can specify which types of errors should be ignored by using an attribute of the root element in the XML document.

In addition to the ability to enable and disable the TCP port, security options for client access lists and HTTP authentication are supported.

To configure XML on the CSM, perform this task:

	Command	Purpose
Step 1	`Router(config-module-csm)# module csm slot`	Specifies the module and slot number.
Step 2	`Router(config-module-csm)# xml-config`	Enables XML on the CSM and enters the XML configuration mode.
Step 3	`Router(config-slb-xml)# port port-number`	(Optional) Specifies the TCP port where the CSM HTTP server listens.
Step 4	`Router(config-slb-xml)# vlan id`	(Optional) Restricts the CSM HTTP server to accept connections only from the specified VLAN.

	Command	Purpose
Step 5	`Router(config-slb-xml)# client-group [1-99 | name]`	(Optional) Specifies that only connections sourced from an IP address matching a client group are accepted by the CSM XML configuration interface.
Step 6	`Router(config-slb-xml)# credentials user-name password password`	(Optional) Configures one or more username and password combinations. When one or more **credentials** commands are specified, the CSM HTTP server authenticates user access using the basic authentication scheme described in RFC 2617.
Step 7	`Router(config-slb-xml)# inservice`	Enables the XML interface.
Step 8	`Router# show module csm 4 xml stats`	Displays a list of XML statistics. **Note** The statistics counters are 32 bit.

This example shows how to configure XML on the CSM:

```
Router(config-module-csm)# configure terminal
Router(config-module-csm)# module csm 4
Router(config-module-csm)# xml-config
Router(config-slb-xml)# port 80
Router(config-slb-xml)# vlan 200
Router(config-slb-xml)# credentials eric password @#$#%%@
Router(config-slb-xml)# inservice
Router# show module csm 4 xml stats
XML config: inservice, port = 80, vlan = 10 (10.0.0.247), client list = <none>
  connection stats:
    current = 0, total = 3
    failed = 3, security failed = 0
  requests: total = 5, failed = 3
Router#
```

When an untolerated XML error occurs, the HTTP response contains a 200 code. The portion of the original XML document with the error is returned with an error element that contains the error type and description.

This example shows an error response to a condition where a virtual server name is missing:

```
      <?xml version="1.0"?>
      <config>
       <csm_module slot="4">
         <vserver>
          <error code="0x20">Missing attribute name in element
vserver</error>
         </vserver>
       </csm_module>
      </config>
```

The error codes returned also correspond to the bits of the error-tolerance attribute of the configuration element. The following list contains the returned XML error codes:

```
XML_ERR_INTERNAL          = 0x0001,
XML_ERR_COMM_FAILURE      = 0x0002,
XML_ERR_WELLFORMEDNESS    = 0x0004,
XML_ERR_ATTR_UNRECOGNIZED = 0x0008,
XML_ERR_ATTR_INVALID      = 0x0010,
XML_ERR_ATTR_MISSING      = 0x0020,
XML_ERR_ELEM_UNRECOGNIZED = 0x0040,
XML_ERR_ELEM_INVALID      = 0x0080,
XML_ERR_ELEM_MISSING      = 0x0100,
```

```
XML_ERR_ELEM_CONTEXT      = 0x0200,
XML_ERR_IOS_PARSER        = 0x0400,
XML_ERR_IOS_MODULE_IN_USE = 0x0800,
XML_ERR_IOS_WRONG_MODULE  = 0x1000,
XML_ERR_IOS_CONFIG        = 0x2000
```

The default error_tolerance value is 0x48, which corresponds to ignoring unrecognized attributes and elements.

Configuring Server Application State Protocol

The Server Application State Protocol (SASP) allows the CSM-S to receive traffic weight recommendations from Workload Managers (WMs) register with the WMs, and enable the WMs to suggest new load-balancing group members to the CSM-S.

SASP is supported on Cisco IOS Release 12.1(13)E3 or later releases and a Cisco IOS release supporting 4.1.2 or later releases is required.

To configure SASP, you must associate a special bind_id with a server farm (for example, a SASP group) and a DFP agent (for example, a SASP Global Workload Manager [GWM]).

Configuring SASP Groups

A SASP group is equivalent to a server farm on the CSM-S. Use the **serverfarm** configuration command to configure the group. The members of the group are all the real servers configured under the server farm. To associate this group with a GWM, assign a SASP bind_id that matches the GWM. To configure SASP groups, use the **bindid** command when you are in the serverfarm configuration submenu as follows:

```
Router(config-slb-sfarm)# bindid 7
```

Configuring a GWM

A GWM is configured as a DFP agent. To configure a GWM, you must enter the DFP submenu under the CSM-S configuration command. This example shows how to configure the GWM as a DFP agent:

```
Router(config-slb-dfp)# agent ip.address port bind id
```

Note The CLI allows you to not enter a bind_id. However, the bind_id is required for the configuration of this agent as a GWM. The CLI describes the bind_id keyword as an “activity timeout” or a “keepalive.” It also allows you to enter two additional values. Do not enter any additional values unless you are troubleshooting an SASP environment.

Alternatively, the GWM can be configured as follows:

```
Router(config-slb-dfp)# agent ip.address port bind id flags
```

or

```
Router(config-slb-dfp)# agent ip.address port bind_id flags keep-alive-interval
```

The keepalive interval is a number that represents seconds and defaults to 180. The flags control how the CSM-S registers with the GWM. The default value is zero. See Table 10-5 for the meaning of the flags.

Table 10-5 SASP Flags

Flags Value	Meaning
0	Uses the CSM-S default registration flags (37).
32	Specifies the default load-balancing registration of the GWM. The load balancer sends a "Get Weights" message to get the new weights and pulls the weights from the GWM. The GWM must include the weights of all group members when sending the weights to this load balancer (including members whose weights have not changed).
33	Specifies that the load balancer should receive weights through the 'Send Weights" message. (The GWM pushes weights to the load balancer.)
34	Allows the GWM to trust any member-initiated registration and deregistration and immediately updates the registration or deregistration in the weights sent.
35	Same as 33 and 34.
36	Specifies that the GWM must not include members whose weights have not changed since the last time period.
37	Same as 33 and 36.
38	Same as 34 and 36.
39	Same as 33, 34, and 36.

Configuring Alternate bind_ids

By default, one bind_id is configured to be a SASP bind_id, 65520. The first bind_id can be any value between 1 and 65525. This example shows how to set the bind_id through the CSM-S configuration command:

```
Router(config-module-csm)# variable SASP_FIRST_BIND_ID value
```

The maximum number of bind_ids that can be used with SASP is eight, which is also the maximum number of supported GWMs. The maximum number of bind_ids can be any value between 0 and 8. This example shows how to set the maximum number of SASP bind_ids in use:

```
Router(config-module-csm)# variable SASP_GWM_BIND_ID_MAX value
```

Note Restart the CSM-S after modifying one of these environment variables.

Configuring a Unique ID for the CSM-S

By default, the CSM-S has a unique identifying string of "Cisco-CSM." This example shows how the string can be set through the CSM-S configuration command:

```
Router(config-module-csm)# variable SASP_CSM_UNIQUE_ID text
```

Note Restart the CSM-S after modifying one of these environment variables.

Configuring Weight Scaling

A weight for a real server on the CSM-S is a number between 0 and 100. SASP weights for members are between 0 to 65536. If the GWM is only producing weights in the CSM-S range, no scaling is needed. If the GWM is using the full SASP range, this range should be mapped. This example shows how to scale SASP weights:

```
Router(config-module-csm)# variable SASP_SCALE_WEIGHTS value
```

The range for SASP_SCALE_WEIGHTS is 0 through 12. Values 0 through 11 cause SASP weights to be divided by 2 raised to the n value. A value of 12 maps the entire 65536 values to the CSM-S 0-100 weight range.

This example shows how to display the SASP GWM details:

```
Router# show module csm 3 dfp detail
DFP Agent 64.100.235.159:3860  Connection state: Connected
   Keepalive = 65521  Retry Count = 33     Interval = 180   (Default)
   Security errors = 0
   Last message received: 03:33:46 UTC 01/01/70
   Last reported Real weights for Protocol any, Port 0
      Host 10.9.10.22       Bind ID 65521  Weight 71
      Host 10.10.12.10      Bind ID 65521  Weight 70
      Host 10.10.12.12      Bind ID 65521  Weight 68
   Last reported Real weights for Protocol any, Port 44
      Host 10.9.10.9        Bind ID 65521  Weight 69

DFP manager listen port not configured
No weights to report to managers.
```

This example shows how to display the SASP group:

```
Router# show module csm 3 serverfarms detail
SVRFARM2, type = SLB, predictor = RoundRobin, nat = SERVER
  virtuals inservice: 0, reals = 4, bind id = 65521, fail action = none
  inband health config: <none>
  retcode map = <none>
  Real servers:
    10.10.12.10, weight = 78, OUTOFSERVICE, conns = 0
    10.10.12.12, weight = 76, OPERATIONAL, conns = 0
    10.9.10.9:44, weight = 77, OPERATIONAL, conns = 0
    10.9.10.22, weight = 79, OUTOFSERVICE, conns = 0
  Total connections = 0
```

This example shows how to display the SASP environment variables:

```
Router# show module csm 3 variable

variable                        value
------------------------------------------------------------------
ARP_INTERVAL                    300
…
ROUTE_UNKNOWN_FLOW_PKTS         0
SASP_FIRST_BIND_ID              65520
SASP_GWM_BIND_ID_MAX            2
SASP_CSM_UNIQUE_ID              paula jones
…
XML_CONFIG_AUTH_TYPE            Basic
```

Back-End Encryption

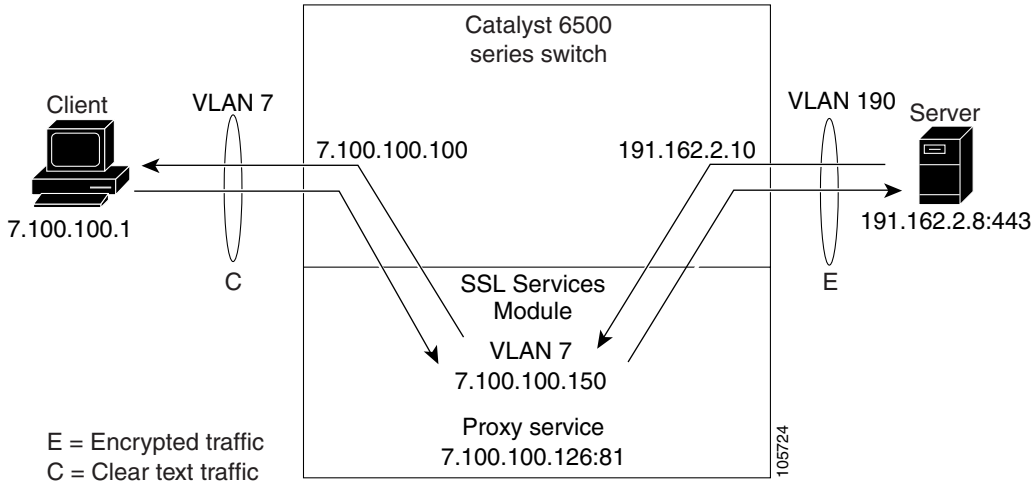
Back-end encryption allows you to create a secure end-to-end environment. In Figure 10-2, the client (7.100.100.1) is connected to switch port 6/47 in access VLAN 7. The server (191.162.2.8) is connected to switch port 10/2 in access VLAN 190.

The SSL proxy VLAN 7 has the following configuration:

- IP address—7.100.100.150
- Static route and gateway:
  - Route 191.0.0.0
  - Gateway 7.100.100.100

  The gateway IP address (the IP address of interface VLAN 7 on the MSFC) is configured so that the client-side traffic that is destined to an unknown network is forwarded to that IP address for further routing to the client.
- Client-side gateway—7.100.100.100 (the IP address of VLAN 7 configured on the MSFC)
- Virtual IP address of client proxy service—7.100.100.150:81
- Server IP address—191.162.2.8

Figure 10-2 Basic Back-End Encryption



Configuring the Client Side

This example shows how to configure the SSL proxy service:

```
ssl-proxy(config)# ssl-proxy service S1
ssl-proxy(config-ssl-proxy)# virtual ipaddr 10.1.0.21 protocol tcp port 443 secondary
ssl-proxy(config-ssl-proxy)# server ipaddr 10.2.0.100 protocol TCP port 80
ssl-proxy(config-ssl-proxy)# inservice
```

This example shows how to configure the CSM-S virtual server:

```
Cat6k-2(config-module-csm)# serverfarm SSLfarm
Cat6k-2(config-slb-sfarm)# real 10.1.0.21 local
Cat6k-2(config-slb-real)# inservice
```

```
Cat6k-2(config-module-csm)# vserver VS1
Cat6k-2(config-slb-vserver)# virtual 10.1.0.21 tcp https
Cat6k-2(config-slb-vserver)# serverfarm SSLfarm
Cat6k-2(config-slb-vserver)# inservice
```

You can perform SSL load balancing on the CSM-S and an SSL Services Module in mixed mode.

The CSM-S uses SSL-ID sticky functionality to stick SSL connections to the same SSL Services Module. The CSM-S must terminate the client-side TCP connection in order to inspect the SSL-ID. The CSM-S must then initiate a TCP connection to the SSL Services Module when a load-balancing decision has been made.

The traffic flow has the CSM-S passing all traffic received on a virtual server to the SSL Services Module with TCP termination performed on the SSL Services Module. When you enable the SSL sticky function, the connection between the CSM-S and the SSL Services Module becomes a full TCP connection.

This example shows how to configure mixed-mode SSL load balancing:

```
Cat6k-2(config-module-csm)# sticky 10 ssl timeout 60
Cat6k-2(config-module-csm)# serverfarm SSLfarm
Cat6k-2(config-slb-sfarm)# real 10.1.0.21 local
Cat6k-2(config-slb-sfarm)# inservice
Cat6k-2(config-slb-sfarm)# real 10.2.0.21
Cat6k-2(config-slb-sfarm)# inservice
Cat6k-2(config-module-csm)# vserver VS1
Cat6k-2(config-slb-vserver)# virtual 10.1.0.21 tcp https
Cat6k-2(config-slb-vserver)# sticky 60 group 10
Cat6k-2(config-slb-vserver)# serverfarm SSLfarm
Cat6k-2(config-slb-vserver)# persistent rebalance
Cat6k-2(config-slb-vserver)# inservice
```

You must make an internally generated configuration to direct traffic at the SSL Services Module when the CSM-S must terminate the client-side TCP connection. You must create a virtual server with the same IP address or port of each local real server in the server farm *SSLfarm*. Internally, this virtual server is configured to direct all traffic that is intended for the virtual server to the SSL Services Module.

You must make an internally generated configuration because the IP address of the local real server and the CSM-S virtual server address must be the same. When the CSM initiates a connection to this local real server, the SYN frame is both sent and received by the CSM-S. When the CSM-S receives the SYN, and the destination IP address or port is the same as the virtual server VS1, the CSM-S matches VS1 unless a more-specific virtual server is added.

Configuring the Server Side

A standard virtual server configuration is used for Layer 4 and Layer 7 load balancing when the SSL Services Module uses the CSM-S as the back-end server.

This example shows how to restrict this virtual server to receive only traffic from the SSL Services Module:

```
Cat6k-2(config-module-csm)# serverfarm SLBdefaultfarm
Cat6k-2(config-slb-sfarm)# real 10.2.0.20
Cat6k-2(config-slb-sfarm)# inservice

Cat6k-2(config-module-csm)# vserver VS2
Cat6k-2(config-slb-vserver)# virtual 10.2.0.100 tcp www
Cat6k-2(config-slb-vserver)# serverfarm SLBdefaultfarm
Cat6k-2(config-slb-vserver)# vlan local
Cat6k-2(config-slb-vserver)# inservice
```

This example shows how to configure the real server as the back-end server:

```
Cat6k-2(config-module-csm)# serverfarm SSLpredictorforward
Cat6k-2(config-slb-sfarm)# predictor forward

Cat6k-2(config-module-csm)# vserver VS3
Cat6k-2(config-slb-vserver)# virtual 0.0.0.0 0.0.0.0 tcp www
Cat6k-2(config-slb-vserver)# serverfarm SSLpredictorforward
Cat6k-2(config-slb-vserver)# inservice
```

Configuring the CSM-S as the Back-End Server

The virtual server and server farm configurations permits you to use real servers as the back-end servers. Use the configuration that is described in the "Configuring the Client Side" section on page 10-27 and then configure the SSL daughter card to use the CSM-S as the back-end server:

This example shows the CSM-S virtual server configuration for Layer 7 load balancing:

```
Cat6k-2(config-module-csm)# serverfarm SLBdefaultfarm
Cat6k-2(config-slb-sfarm)# real 10.2.0.20
Cat6k-2(config-slb-real)# inservice

Cat6k-2(config-module-csm)# serverfarm SLBjpgfarm
Cat6k-2(config-slb-sfarm)# real 10.2.0.21

Cat6k-2(config-module-csm)# map JPG url
Cat6k-2(config-slb-map-cookie)# match protocol http url *jpg*

Cat6k-2(config-module-csm)# policy SLBjpg
Cat6k-2(config-slb-policy)# url-map JPG
Cat6k-2(config-slb-policy)#serverfarm SLBjpgfarm

Cat6k-2(config-module-csm)# vserver VS2
Cat6k-2(config-slb-vserver)# virtual 10.2.0.100 tcp www
Cat6k-2(config-slb-vserver)# serverfarm SLBdefaultfarm
Cat6k-2(config-slb-vserver)# slb-policy SLBjpg
Cat6k-2(config-slb-vserver)# inservice
```

This example shows the CSM-S virtual server configuration for Layer 4 load balancing:

```
Cat6k-2(config-module-csm)# serverfarm SLBdefaultfarm
Cat6k-2(config-slb-sfarm)# real 10.2.0.20
Cat6k-2(config-slb-real)# inservice

Cat6k-2(config-module-csm)# vserver VS2
Cat6k-2(config-slb-vserver)# virtual 10.2.0.100 tcp www
Cat6k-2(config-slb-vserver)# serverfarm SLBdefaultfarm
Cat6k-2(config-slb-vserver)# vlan local
Cat6k-2(config-slb-vserver)# inservice
```

Configuring the Real Server as the Back-End Server

The server-side configuration traffic flow with the real server as the back-end server is similar to the client-side configuration. Use the configuration that is described in "Configuring the Client Side" section on page 10-27 and then configure the SSL Services Module to use a real server as the back-end server.

No new configuration is required for the SSL Services Module proxy service configuration. This example shows how the configuration is internally initiated and hidden from the user:

```
ssl-proxy(config)# ssl-proxy service S1
ssl-proxy(config-ssl-proxy)# virtual ipaddr 10.1.0.21 protocol tcp port 443 secondary
ssl-proxy(config-ssl-proxy)# server ipaddr 10.2.0.20 protocol TCP port 80
ssl-proxy(config-ssl-proxy)# inservice
```

This example shows how to configure the CSM-S virtual server:

```
Cat6k-2(config-module-csm)# serverfarm SSLreals

Cat6k-2(config-slb-sfarm)# real 10.2.0.20
Cat6k-2(config-slb-sfarm)# inservice

Cat6k-2(config-module-csm)# serverfarm SSLpredictorforward
Cat6k-2(config-slb-sfarm)# predictor forward

Cat6k-2(config-module-csm)# vserver VS3
Cat6k-2(config-slb-vserver)# virtual 0.0.0.0 0.0.0.0 tcp www
Cat6k-2(config-slb-vserver)# serverfarm SSLpredictorforward
Cat6k-2(config-slb-vserver)# inservice
```

Configuring Health Monitoring

This chapter describes how to configure the health monitoring on the CSM-S and contains these sections:

Configuring Probes for Health Monitoring

Configuring health probes to the real servers allows you to determine if the real servers are operating correctly. A real server's health is categorized as follows:

- Active—The real server responds appropriately.
- Suspect—The real server is unreachable or returns an invalid response. The probes are retried.
- Failed—The real server fails to reply after a specified number of consecutive retries. You are notified and the CSM-S adjusts incoming connections accordingly. Probes continue to a failed server until the server becomes active again.

The CSM-S supports probes used to monitor real servers. Configuring a probe involves the following:

- Entering the probe submode
- Naming the probe
- Specifying the probe type

The CSM-S supports a variety of probe types that monitor real servers, including FTP, DNS, or HTTP.

Note By default, no probes are configured on the CSM-S.

When configuring the CSM-S for health probe monitoring, you can use a multiple-tiered approach that includes the following actions:

- Active probes—These probes run periodically. ICMP, TCP, HTTP, and other predefined health probes fall into this category. Scripted health probes are included here as well. Active probes do not impact the session setup or teardown system.
- Passive monitoring (in-band health monitoring)—Monitors sessions for catastrophic errors that can remove a server from services. Catastrophic errors may be reset (RST) when there is no response from a server. These health checks operate at a full-session rate, and recognize failing servers quickly.

- Passive HTTP error code checking (in-band response parsing)—The CSM-S parses HTTP return codes and watches for codes such as "service unavailable" so that it can take a server out of service. Passive HTTP error code checking has a small impact on session performance.

To set up a probe, you must configure it by naming the probe and specifying the probe type while in probe submode.

After configuring a probe, you must associate it with a server farm for the probe to take effect. All servers in the server farm receive probes of the probe types that are associated with that server farm. You can associate one or more probe types with a server farm.

If you assign a port number when configuring either the real server or the virtual server, you do not need to specify a port number when you configure a probe. The probe inherits the port number from the real or virtual server configuration.

You can override the real server's and virtual server's port information by explicitly specifying a port to probe in the health probe configuration using the optional health probe port feature. This feature allows you to set a port for use by the health probes when no port is specified either in the real server or virtual server.

After you configure a probe, associate single or multiple probes with a server farm. All servers in the server farm receive probes of the probe types that are associated with that pool.

Note If you associate a probe of a particular type with a server farm containing real servers that are not running the corresponding service, the real servers send error messages when they receive a probe of that type. This action causes the CSM-S to place the real server in a failed state and disable the real server from the server farm.

To specify a probe type and name, perform this task:

	Command	Purpose
Step 1	`Router(config-module-csm)# probe probe-name {`**`http`** `|` **`icmp`** `|` **`telnet`** `|` **`tcp`** `|` **`ftp`** `|` **`smtp`** `|` **`dns`** `|` **`kal-ap-upd`**`}`	Specifies a probe type and a name[1] [2]: • *probe-name* is the name of the probe being configured; it has a character string of up to 15 characters. • **http** creates an HTTP probe with a default configuration. • **icmp** creates an ICMP probe with a default configuration. • **telnet** creates a Telnet probe with a default configuration. • **tcp** creates a TCP probe with a default configuration. • **ftp** creates an FTP probe with a default configuration. • **smtp** creates an SMTP probe with a default configuration. • **dns** creates a DNS probe with a default configuration. • **kal-ap-upd** creates a GSLB target probe.
Step 2	`Router(config-slb-probe-tcp)#` **`port`** *`port-number: 1-MAXUSHORT`*	Configures an optional port for a probe[3].
Step 3	`Router#` **`show module csm`** *`slot`* **`probe`**	Displays all probes and their configuration.
Step 4	`Router#` **`show module csm`** *`slot`* ***`tech-support probe`***	Displays probe statistics.

1. The **no** form of this command removes the probe type from the configuration.

2. Inband health monitoring provides a more scalable solution if you are receiving performance alerts.
3. The **port** command does not exist for the ICMP or PING health probe.

Note When you specify a probe name and type, it is initially configured with the default values. Enter the probe configuration commands to change the default configuration.

This example shows how to configure a probe:

```
Router(config-module-csm)# probe probe1 tcp
Router(config-slb-probe-tcp)# interval 120
Router(config-slb-probe-tcp)# retries 3
Router(config-slb-probe-tcp)# failed 300
Router(config-slb-probe-tcp)# open 10
Router(config-slb-probe-tcp)# serverfarm sf4
Router(config-slb-sfarm)# real 10.1.0.105
Router(config-slb-real)# inservice
Router(config-slb-real)# probe probe1
Router(config-slb-sfarm)# vserver vs4
Router(config-slb-vserver)# virtual 10.1.0.84 tcp 80
Router(config-slb-vserver)# serverfarm sf4
Router(config-slb-vserver)# inservice
Router(config-slb-vserver)# end
```

Note There are two different timeout values: open and receive. The open timeout specifies how many seconds to wait for the connection to open (that is, how many seconds to wait for SYN ACK after sending SYN). The receive timeout specifies how many seconds to wait for data to be received (that is, how many seconds to wait for an HTTP reply after sending a GET/HHEAD request). Because TCP probes close as soon as they open without sending any data, the receive timeout is not used.

Probe Configuration Commands

These commands are common to all probe types:

Command	Purpose
`Router(config-slb-probe)#` **`interval`** *`seconds`*	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[1] [2]. Range = 2–65535 seconds Default = 120 seconds
`Router(config-slb-probe)#` **`retries`** *`retry-count`*	Sets the number of failed probes that are allowed before marking the server as failed[1]. Range = 0–65535 Default = 3
`Router(config-slb-probe)#` **`failed`** *`failed-interval`*	Sets the time between health checks when the server has been marked as failed. The time is in seconds[1]. Range = 2–65535 Default = 300 seconds

Command	Purpose
`Router(config-slb-probe)#` **`recover`** *`recover_value`*	Sets the number of consecutive responses that are received before marking a failed server as healthy[1]. Range = 1–65535 Default = 1
`Router(config-slb-probe)#` **`open`** *`open-timeout`*	Sets the maximum time to wait for a TCP connection. This command is not used for any non-TCP health checks (ICMP or DNS[1]). Range = 1–65535 Default = 10 seconds
`Router(config-slb-probe)#` **`description`** *`description`*	(Optional) Specifies a description for the probe. Limit the *description* to 80 characters.

1. The **no** form of this command restores the defaults.
2. Inband health monitoring provides a more scalable solution if you are receiving performance alerts.

Configuring an HTTP Probe

An HTTP probe establishes an HTTP connection to a real server and then sends an HTTP request and verifies the response. The **probe** *probe-name* **http** command places the user in HTTP probe configuration submode.

To configure an HTTP probe, perform this task:

	Command	Purpose
Step 1	`Router(config-module-csm)#` **`probe`** *`probe-name`* **`http`**	Configures an HTTP probe and enters the HTTP probe submode[1].
Step 2	`Router(config-slb-probe-http)#` **`credentials`** *`username`* [*`password`*]	Configures basic authentication values for the HTTP SLB probe[1].
Step 3	`Router(config-slb-probe-http)#` **`expect status`** *`min-number`* [*`max-number`*]	Configures a status code to expect from the HTTP probe. You can configure multiple status ranges by entering one **expect status** command at a time[1]. *min-number*—If you do not specify a *max-number*, this number is taken as a single status code. If you specify a maximum number, this number is taken as the minimum status code of a range. *max-number*—The maximum status code in a range. The default range is 0–999. (Any response from the server is considered valid.) **Note** If no maximum is specified, this command takes a single number (min-number). If you specify both a minimum number and a maximum number, it takes the range of numbers.

	Command	Purpose
Step 4	`Router(config-slb-probe-http)# header field-name [field-value]`	Configures a header field for the HTTP probe. Multiple header fields may be specified[1].
Step 5	`Router(config-slb-probe-http)# request [method [get|head]] [url path]`	Configures the request method used by an HTTP probe[1]: • **get**—The HTTP **get** request method directs the server to get this page. • **head**—The HTTP **head** request method directs the server to get only the header for this page. • **url**—A character string of up to 1275 characters specifies the URL path; the default path is "/". **Note** The CSM-S supports only the **get** and **head** request methods; it does not support the **post** and other methods. The default method is **get**.

1. The **no** form of this command restores the defaults.

Configuring an ICMP Probe

An ICMP probe sends an ICMP echo (for example, ping) to the real server. The **probe icmp** command enters the ICMP probe configuration mode. All the common **probe** commands are supported except the **open** command, which is ignored.

To configure an ICMP probe, perform this task:

	Command	Purpose
Step 1	`Router(config-module-csm)# probe probe-name icmp`	Configures an ICMP probe and enters the ICMP probe submode[1].
Step 2	`Router(config-slb-probe-icmp)# interval`	Configures the intervals to wait between probes of a failed server and between probes.
Step 3	`Router(config-slb-probe-icmp)# receive`	Specifies the time to make a TCP connection to receive a reply from the server.
Step 4	`Router(config-slb-probe-icmp)# retries`	Limits the number of retries before considering the server as failed.

1. The **no** form of this command restores the defaults.

Configuring a UDP Probe

The UDP probe requires ICMP because otherwise the UDP probe will be unable to detect when a server has gone down or has been disconnected. You must associate UDP to the supervisor engine and then configure ICMP.

Because the UDP probe is a raw UDP probe, the CSM-S is using a single byte in the payload for probe responses. The CSM-S does not expect any meaningful response from the UDP application. The CSM-S uses the ICMP Unreachable message to determine if the UDP application is not reachable. If there is no ICMP unreachable reply in the receive timeout, the CSM-S assumes that the probe is operating correctly.

If the IP interface of the real server is down or disconnected, the UDP probe by itself would not know that the UDP application is not reachable. You must configure the ICMP probe in addition to the UDP probe for any given server.

The CSM-S uses the DNS probe as the high-level UDP application. You can use a TCL script to configure this probe. See Chapter 12, "Using TCL Scripts with the CSM-S."

To configure an ICMP probe, perform this task:

	Command	Purpose
Step 1	`Router(config-module-csm)# probe` *probe-name* **udp**	Configures a UDP probe and enters the UDP probe submode[1].
Step 2	`Router(config-slb-probe-icmp)# interval`	Configures the intervals to wait between probes of a failed server and between probes.
Step 3	`Router(config-slb-probe-icmp)# receive`	Specifies the time to make a TCP connection to receive a reply from the server.
Step 4	`Router(config-slb-probe-icmp)# retries`	Limits the number of retries before considering the server as failed.

1. The **no** form of this command restores the defaults.

Configuring a TCP Probe

A TCP probe establishes and removes connections. The **probe tcp** command enters the TCP probe configuration mode. All the common **probe** commands are supported.

To configure a TCP probe, perform this task:

	Command	Purpose
Step 1	`Router(config-module-csm)# probe` *probe-name* **tcp**	Configures a TCP probe and enters the TCP probe submode[1].
Step 2	`Router(config-slb-probe-icmp)# interval`	Configures the intervals to wait between probes of a failed server and between probes.
Step 3	`Router(config-slb-probe-icmp)# retries`	Limits the number of retries before considering the server as failed.

1. The **no** form of this command restores the defaults.

Configuring FTP, SMTP, and Telnet Probes

An FTP, SMTP, or Telnet probe establishes a connection to the real server and verifies that a greeting from the application was received. The **probe** (**ftp, smtp,** or **telnet**) command enters the corresponding probe configuration mode. All the **probe** common options are supported. Multiple status ranges are supported, one command at a time.

To configure a status code to expect from the FTP, SMTP, or Telnet probe, perform this task:

	Command	Purpose
Step 1	`Router(config-module-csm)# probe probe-name [ftp | smtp | telnet]`	Configures an FTP, SMTP, or Telnet probe and enters the FTP, SMTP, or Telnet probe submode[1].
Step 2	`Router(config-slb-probe-icmp)# interval`	Configures the intervals to wait between probes of a failed server and between probes.
Step 3	`Router(config-slb-probe-icmp)# receive`	Specifies the time to make a TCP connection to receive a reply from the server.
Step 4	`Router(config-slb-probe-icmp)# retries`	Limits the number of retries before considering the server as failed.

1. The **no** form of this command restores the defaults.

Specifying the DNS Resolve Request

A DNS probe sends a domain name resolve request to the real server and verifies the returned IP address. The **probe dns** command places the user in DNS probe configuration submode. All the probe common options are supported except **open,** which is ignored.

To specify the domain name resolve request, perform this task:

	Command	Purpose
Step 1	`Router(config-module-csm)# probe probe-name dns`	Configures a DNS probe and enters the tcp probe submode[1].
Step 2	`Router(config-slb-probe-dns)# [failed | interval | retries | receive]`	Configures the times to wait between probes to make a DNS connection, to receive a reply from the server, and to limit the number of retries before considering the real server as failed.

1. The **no** form of this command restores the defaults.

Configuring GSLB Probes

GSLB capabilities for the Cisco CSM were introduced in the CSM software release 3.1. With this capability, the CSM combines GSLB with route health injection.

When configuring Global Server Load Balancing (GSLB) type probes, the **port** submode command is not used to specify which destination UDP port to query. Use the CSM environment variable GSLB_KALAP_UDP_PORT instead. The default is port 5002.

To specify probe frequency and the number of retries for KAL-AP, ICMP, HTTP, and DNS probes when associated with a GSLB server farm environment, the following variables must be used instead of the probe configuration submode commands:

```
GSLB_KALAP_PROBE_FREQ           10
GSLB_KALAP_PROBE_RETRIES        3
GSLB_ICMP_PROBE_FREQ            10
GSLB_ICMP_PROBE_RETRIES         3
```

```
GSLB_HTTP_PROBE_FREQ            10
GSLB_HTTP_PROBE_RETRIES         2
GSLB_DNS_PROBE_FREQ             10
GSLB_DNS_PROBE_RETRIES          3
```

GSLB for the CSM is deployed such that the CSM becomes the DNS authoritative name server for the GSLB subdomain. Although the CSM does not use CAPP, it keeps track of local and remote devices by using health-checking probes. Theses probes are sent to the local real server within the data center and to the remote virtual server on the remote CSMs. The CSM should be configured with the same probes to monitor the availability of the local server and the remote virtual server at the active data center.

When DNS requests are sent to the CSM, the CSM assesses the load and availability of the local virtual servers and the remote virtual servers that support the requested domain. The CSM then responds with the IP address of the local or remote virtual servers, depending on the availability and load of these servers. If the information collected by the health probes confirms that the local virtual server is unavailable or overloaded, the CSM responds with the IP address of the remote virtual server.

Understanding and Configuring Inband Health Monitoring

These sections describe inband health monitoring:

Understanding Inband Health Monitoring

To efficiently balance connections, the CSM-S must continuously monitor the health of all real servers in its configuration. The inband health monitoring feature is configured for each server farm to monitor the health of the servers. The parameters configured per server farm are then applied to each real server in that server farm. You can configure the number of abnormal end sessions that occur before the system considers the real server unreachable. You also can specify a time to wait before a real server is reintroduced into the server farm and a connection attempt is made.

This feature works with health probes. If health probes and inband health monitoring are both configured on a particular server, both sets of health checks are required to keep a real server in service within a server farm. If either health-checking feature finds a server out of service, the server will not be selected by the CSM-S for load balancing.

Configuring Inband Health Monitoring

To configure inband health monitoring, perform these steps:

Step 1 Verify that you have configured server farms. (See the "Configuring Server Farms" section on page 5-1.)

Step 2 Enter the serverfarm submode command to enable inband health monitoring for each server farm:

```
Router(config-module-csm)# serverfarm serverfarm-name
Router(config-slb-sfarm)# health retries count failed seconds
```

Note Retries are the number of abnormal end sessions that the CSM-S will tolerate before removing a real server from service. The failed time is the number of seconds that the CSM-S waits before reattempting a connection to a real server that was removed from service by inband health checking.

This example shows how to enable inband health monitoring for a server farm named geo:

```
Router(config-module-csm)# serverfarm geo
Router(config-slb-sfarm)# health retries 43 failed 160
```

Understanding and Configuring HTTP Return Code Checking

These sections describe HTTP return code checking:

- Understanding HTTP Return Code Checking, page 11-9
- Configuring HTTP Return Code Checking, page 11-10

Understanding HTTP Return Code Checking

The return error code checking (return code parsing) feature is used to indicate when a server is not returning web pages correctly. This feature extends the capability of CSM-S to inspect packets, parse the HTML return codes, and act upon the return codes returned by the server.

After receiving an HTTP request from the CSM-S, the server responds with an HTTP return code. The CSM-S can use the HTTP return error codes to determine the availability of the server. The CSM-S can be configured to take a server out of use in response to receiving specific return codes.

A list of predefined codes (100 through 599) are in RFC 2616. For return code checking, some codes are more usable than others. For example, a return code of 404 is defined as a URL not found, which may be the result of the user entering the URL incorrectly. Error code 404 also might mean that the web server has a hardware problem, such as a defective disk drive preventing the server from finding the data requested. In this case, the web server is still alive, but the server cannot send the requested data because of the defective disk drive. Because of the inability of the server to return the data, you do not want future requests for data sent to this server. To determine the error codes you want to use for return code checking, refer to RFC 2616.

When HTTP return code checking is configured, the CSM-S monitors HTTP responses from all balanced HTTP connections and logs the occurrence of the return code for each real server. The CSM-S stores return code counts. When a threshold for a return code is reached, the CSM-S may send syslog messages or remove the server from service.

A default action, counting return codes, syslog messaging, or removing the real server from service or a set of these actions can be applied to a server farm. You also can bind a single virtual group to multiple server farms allowing you to reuse a single return code server farm policy on multiple server farms.

Note When you configure HTTP return code checking on a virtual server, the performance of that virtual server is impacted. Once return code parsing is enabled, all HTTP server responses must be parsed for return codes.

Configuring HTTP Return Code Checking

When you configure return error code checking, you configure the attributes of a server farm and associate it with a return code map.

To configure the return code checking, perform these steps:

Step 1 Verify that you have configured HTTP virtual servers. (See the "Configuring Redirect Virtual Servers" section on page 6-6.)

Step 2 Enter the map return code command to enable return code mapping and enter the return code map submode:

```
Router(config-module-csm)# map name retcode
```

Step 3 Configure the return code parsing:

```
Router(config-slb-map-retcode)# match protocol http retcode min max action [count | log |
remove] threshold [reset seconds]
```

You can set up as many matches as you want in the map.

Step 4 Assign a return code map to a server farm:

```
Router(config-slb-sfarm)# retcode-map name
```

This example shows how to enable return error code checking:

```
Router(config-module-csm)# map httpcodes retcode
Route(config-slb-map-retcode)# match protocol http retcode 401 401 action log 5 reset 120
Route(config-slb-map-retcode)# match protocol http retcode 402 415 action count
Route(config-slb-map-retcode)# match protocol http retcode 500 500 action remove 3 reset 0
Route(config-slb-map-retcode)# match protocol http retcode 503 503 action remove 3 reset 0
Route(config-slb-map-retcode)# exit
Router(config-module-csm)# serverfarm farm1
Router(config-slb-sfarm)# retcode-map httpcodes
Router(config-slb-sfarm)# exit
Router(config-module-csm)# end
```

Using TCL Scripts with the CSM-S

This chapter describes how to configure content switching and contains these sections:

The CSM-S allows you to upload and execute Toolkit Command Language (TCL) scripts on the CSM-S. Using TCL scripts, you can write customized TCL scripts to develop customized health probes or standalone tasks.

The TCL interpreter code in CSM-S is based on Release 8.0 of the standard TCL distribution. You can create a script to configure health probes (see the "Configuring Probes for Health Monitoring" section on page 11-1) or perform tasks on the CSM-S that are not part of a health probe. The CSM-S periodically executes the scripts at user-configurable intervals.

TCL is a widely used scripting language within the networking community. TCL also has huge libraries of scripts developed that can easily be found from various sites.

The CSM-S currently supports two script modes:

- Probe script mode—These scripts must be written using some simple rules. The execution of these scripts is controlled by health-monitoring module.

  As part of a script probe, the script is executed periodically, and the exit code that is returned by the executing script indicates the relative health and availability of specific real servers. Script probes operate similarly to other health probes available in the current implementation of CSM-S software.

- Standalone script mode—These scripts are generic TCL scripts. You control the execution of these scripts through the CSM-S configuration. A probe script can be run as a standalone task.

For your convenience, sample scripts are available to support the TCL feature. Other custom scripts will work, but these sample scripts are supported by Cisco TAC. The file with sample scripts is located at this URL:

http://www.cisco.com/cgi-bin/tablebuild.pl/cat6000-intellother

The file containing the scripts is named: c6slb-script.3-3-1.tcl.

Loading Scripts

Scripts are loaded onto the CSM-S through script files. A script file may contain zero, one, or more scripts. Each script requires 128 KB of stack space. Because there can be a maximum of 50 health scripts, the maximum stack space for script probes is 6.4 MB. Standalone scripts may also be running, which would consume more stack space.

Examples for Loading Scripts

Scripts can be loaded from a TFTP server, bootflash, slot0, and other storage devices using the **script file** [*file-url*] command.

This example shows how to load a script:

```
Router(config)# module csm 4
Router(config-module-csm)# script file tftp://192.168.1.1/httpProbe.test
```

The script name is either the filename of the script or a special name encoded within the script file. Each script file may contain a number of scripts in the same file. To run the script or create a health probe using that script, you must refer to the script name, not the script file from which the script was loaded.

In order to identify each relevant script, each script must start with a line:

```
#!name = script_name
```

This example shows a master script file in which the scripts are bundled:

```
#!name = SCRIPT1
puts "this is script1"
!name = SCRIPT2
puts "this is script2"
```

This example shows how to find the scripts available in a master script file:

```
Router(config)# configure terminal
Router(config-t)# module csm 4
Router(config-module-csm)# script file tftp://192.168.1.1/script.master
Router(config-module-csm)# end
```

This example shows three scripts available from the script.master file:

```
Router(config)# show module csm 4 file tftp://192.168.1.1/script.master
script1, file tftp://192.168.1.1/script.master
  size = 40, load time = 03:49:36 UTC 03/26/93
script2, file tftp://192.168.1.1/script.master
  size = 40, load time = 03:49:36 UTC 03/26/93
```

To show the contents of a loaded script file, use this command:

```
Router(config)# show module csm slot script full_file_URL code
```

This example shows how to display the code within a named script:

```
router1# show module csm 6 script name script1 code
script1, file tftp://192.168.1.1/script.master
  size = 40, load time = 03:04:36 UTC 03/06/93
#!name = script1
```

One major difference between a standalone script task and a script probe is that the health script is scheduled by the health monitoring CSM-S module. These conditions apply:

- A script can be modified while a script probe is active. The changes are applied automatically in the next script execution and for command line arguments.
- During probe configuration, a particular script is attached to the probe. If the script is unavailable at that time, the probe executes with a null script. If this situation occurs, a warning flag is generated. However, when the script is loaded again, the binding between the probe object and the script does not run automatically. You must use the **no script** and **script** commands again to do the binding.
- After a script is loaded, it remains in the system and cannot be removed. You can modify a script by changing a script and then by entering the **no script file** and **script file** commands again.
- Each script is always identified by its unique name. If two or more scripts have identical names, the last loaded script is used by the CSM-S. When there are duplicate script names, a warning message is generated by the CSM-S.

Reloading TCL Scripts

After a script file has been loaded, the scripts in that file exist in the CSM-S independent of the file from which that script was loaded. If a script file is subsequently modified, use the **script file** command to reload the script file and enable the changes on the CSM-S. (Refer to the *Content Switching Module with SSL Command Reference* for more information.) For example:

```
router(config)# module csm 4
router(config-module-csm)# no script file tftp://192.168.1.1/script.master
router(config-module-csm)# script file tftp://192.168.1.1/script.master
Loading script.master from 192.168.1.1 (via Vlan100): !!!!!!!!!!!!!!!
[OK - 74804 bytes]
router(config-module-csm)# end
```

The **no script file** command removes the **script file** command from the running configuration. This command does not unload the scripts in that file and does not affect scripts that are currently running on the CSM-S. You cannot unload scripts that have been loaded. If a loaded script is no longer needed, it is not necessary to remove it.

TCL Scripts and the CSM-S

The CSM-S Release 1.1(1) TCL script feature is based on the TCL 8.0 source distribution software. CSM-S TCL is modified so that it can be interrupted to call another process unlike the standard TCL library, allowing for concurrent TCL interpreter execution. The CSM-S TCL library does not support any standard TCL file I/O command, such as file, fcopy, and others.

Table 12-1 lists the TCL commands that are supported by the CSM-S.

Table 12-1 TCL Commands Supported by the CSM-S

Command			
Generic TCL Commands			
append	array	binary	break
catch	concat	continue	error
eval	exit	expr	fblocked

Table 12-1 TCL Commands Supported by the CSM-S (continued)

Command			
for	foreach	format	global
gets	if	incr	info
join	lappend	lindex	linsert
list	llength	lrange	lreplace
lsearch	lsort	proc	puts
regexp	regsub	rename	return
set	split	string	subst
switch	unset	uplevel	upvar
variable	while	namespace	
Time-Related Commands			
after	clock	time	
Socket Commands			
close	blocked	fconfigured	fileevent
flush	eof	read	socket
update	vwait		

Table 12-2 lists the TCL command not supported by the CSM-S.

Table 12-2 TCL Commands Not Supported by the CSM-S

Generic TCL Commands			
cd	fcopy	file	open
seek	source	tell	filename
load	package		

Table 12-3 lists the TCL commands specific to the CSM-S.

Table 12-3 CSM-S Specific TCL Commands

Command	Definition
disable_real *serverfarmName realIp port ,-1* | **all** *probeNumId probeNameId*	Disables a real server from the server farm by placing it in the PROBE_FAIL state. This command returns a 1 if successful and returns a 0 if it fails, as follows: `disable_real SF_TEST 1.1.1.1 -1 10 cisco` **Note** The server farm name must be uppercase per the caveat CSCec72471.
enable_real *serverfarmName realIp port ,-1* | **all** *probeNumId probeNameId*	Enables a real server from the PROBE_FAIL state to the operational state. This command returns a 1 if successful and returns a 0 if it fails, as follows: `enable_real SF_TEST 1.1.1.1 -1 10 cisco` **Note** The server farm name must be uppercase per the caveat CSCec72471.
gset *varname value*	Allows you to preserve the state of a probe by setting a variable that is global to all probe threads running from the same script. This command works properly only for probe scripts, not for standalone scripts. Variables in a probe script are only visible within one probe thread. Each time a probe exits, all variables are gone. For example, if a probe script contains a 'gset x 1 ; incr x', variable x would increase by 1 for each probe attempt. • To get the value of a variable from script, set *var* or $*var*. • To reset the value of a variable from script, unset *var*. • To display the current value of a variable, use the **show module csm** *slot* **tech script** command. See the "Debugging Probe Scripts" section on page 12-13 for additional details.

Table 12-3 CSM-S Specific TCL Commands (continued)

Command	Definition
socket -graceful host A.B.C.D port	By default, all CSM-S script probes close the TCP socket by sending a reset. This action is taken to avoid the TIME_WAIT state when the CSM-S initializes an active TCP close. Due to the limitation of 255 sockets available on vxworks, when there are too many probes running at the same time, the CSM-S can run out of system resources and the next probe attempt will fail when opening the socket. When the socket -graceful command is entered, the CSM-S closes TCP connections with a FIN instead of a reset. Use this command only when there are fewer then 250 probes on the system, as follows: `set sock [socket -graceful 192.168.1.1 23]`
ping [*numpacket*] *host A.B.C.D*	This command is currently disabled in CSM-S release 3.2. Allows you to ping a host from a script. This command returns a 1 if successful and returns a 0 if it fails, as follows: `set result [ping 3 1.1.1.1]` **Note** This command blocks the script if the remote host is not in the same subnet as the CSM-S per caveat CSCea67098.

xml *xmlConfigString*	Sends an XML configuration string to the CSM-S from a TCL script. This command works only when the XML server is enabled on the CSM-S. Refer to the XML configuration section.
	This command returns a string with the XML configuration result, as follows:
	`set cfg_result [ xml {`
	`<config>`
	`<csm_module slot="6">`
	`<serverfarm name="SF_TEST">`
	`</serverfarm>`
	`</config>`
	`}`
	`]`

The UDP command set allows Scotty-based TCL scripts to run on the CSM-S. Scotty is the name of a software package that allows you to implement site-specific network management software using high-level, string-based APIs. All UDP commands are thread safe (allowing you to share data between several programs) like the rest of the CSM-S TCL commands.

Table 12-4 lists the UDP commands used by the CSM-S.

Table 12-4 UDP Commands

Command	Definition
udp binary send *handle* [*host port*] *message*	Sends binary data containing a message to the destination specified by the host and port. The *host* and *port* arguments may not be used if the UDP handle is already connected to a transport endpoint. If the UDP handle is not connected, you must use these optional arguments to specify the destination of the datagram.
udp bind *handle* **readable** [*script*] **udp bind** *handle* **writable** [*script*]	Allows binding scripts to a UDP handle. A script is evaluated once the UDP handle becomes either readable or writable, depending on the third argument of the **udp bind** command. The script currently bound to a UDP handle can be retrieved by calling the **udp bind** command without a *script* argument. Bindings are removed by binding an empty string.
udp close *handle*	Closes the UDP socket associated with a handle.
udp connect *host port*	Opens a UDP datagram socket and connects it to a port on a remote host. A connected UDP socket only allows sending messages to a single destination. This usually allows shortening the code because there is no need to specify the destination address for each **udp send** command on a connected UDP socket. The command returns a UDP handle.
udp info [*handle*]	Without the *handle* argument, this command returns a list of all existing UDP handles. Information about the state of a UDP handle can be obtained by supplying a valid UDP handle. The result is a list containing the source IP address, the source port, the destination IP address, and the destination port.
udp open [*port*]	Opens a UDP datagram socket and returns a UDP handle. The socket is bound to a given port number or name. An unused port number is used if the *port* argument is missing.
udp receive *handle*	Receives a datagram from the UDP socket associated with the handle. This command blocks until a datagram is ready to be received.
udp send *handle* [*host port*] *message*	Sends ASCII data containing a message to the destination specified by the host and port. The *host* and *port* arguments may not be used if the UDP handle is already connected to a transport endpoint. If the UDP handle is not connected, you must use these optional arguments to specify the destination of the datagram.

Probe Scripts

The CSM-S supports several specific types of health probes, such as HTTP health probes, TCP health probes, and ICMP health probes when you need to use a diverse set of applications and health probes to administer your network. The basic health probe types supported in the current CSM-S software release often do not support the specific probing behavior that your network requires. To support a more flexible health-probing functionality, the CSM-S now allows you to upload and execute TCL scripts on the CSM-S.

You can create a script probe that the CSM-S periodically executes for each real server in any server farm associated with a probe. Depending upon the exit code of such a script, the real server is considered healthy, suspect, or failed. Probe scripts test the health of a real server by creating a network connection to the server, sending data to the server, and checking the response. The flexibility of this TCL scripting environment makes the available probing functions possible.

After you configure each interval of time, an internal CSM-S scheduler schedules the health scripts. Write the script as if you intend to perform only one probe. You must declare the result of the probe using the **exit** command.

A health script typically performs these actions:

- Opens a socket to an IP address.
- Sends one or more requests.
- Reads the responses.
- Analyzes the responses.
- Closes the socket.
- Exits the script by using exit 5000 (success) or exit 5001 for failure.

You can use the new **probe** *probe-name* **script** command for creating a script probe in Cisco IOS software. This command enters a probe submode that is similar to the existing CSM-S health probe submodes (such as HTTP, TCP, DNS, SMTP, and so on.). The probe script submode contains the existing probe submode commands **failed**, **interval**, **open**, **receive**, and **retries**.

A new **script** *script-name* command was added to the probe script submode. This command can process up to five arguments that are passed to the script when it is run as part of the health probe function.

Example for Writing a Probe Script

This example shows how a script is written to probe an HTTP server using a health script:

```
Router(config)# !name = HTTP_TEST

# get the IP address of the real server from a predefined global array csm_env
set ip $csm_env(realIP)
set port 80
set url "GET /index.html HTTP/1.0\n\n"

# Open a socket to the server. This creates a TCP connection to the real server
set sock [socket $ip $port]
fconfigure $sock -buffering none -eofchar {}

# Send the get request as defined
puts -nonewline $sock $url;

# Wait for the response from the server and read that in variable line
set line [ read $sock ]

# Parse the response
if { [ regexp "HTTP/1.. (\[0-9\]+) " $line match status  ] } {
  puts "real $ip server response : $status"
}

# Close the socket. Application must close the socket once the
# is over. This allows other applications and tcl scripts to make
# a good use of socket resource. Health monitoring is allowed to open
# only 200 sockets simultaneously.
close $sock
```

```
# decide the exit code to return to control module.
# If the status code is OK then script MUST do exit 5000
# to signal successful completion of a script probe.
# In this example any other status code means failure.
# User must do exit 5001 when a probe has failed.
if { $status == 200 } {
    exit 5000
} else {
    exit 5001
}
```

Environment Variables

Health probe scripts have access to many configured items through a predefined TCL array. The most common use of this array is to find the current real server IP addresses of the suspect during any particular launch of the script.

Whenever a script probe is executed on the CSM-S, a special array called csm_env is passed to the script. This array holds important parameters that may be used by the script.

Note The environmental variable information in these sections applies to only probe scripts, not standalone scripts.

Table 12-5 lists the members of the csm_env array.

Table 12-5 Member list for the csm_env Array

Member name	Content
realIP	Suspect IP address
realPort	Suspect IP port
intervalTimeout	Configured probe interval in seconds
openTimeout	Configured socket open timeout for this probe
recvTimeout	Configured socket receive timeout for this probe
failedTimeout	Configure failed timeout
retries	Configured retry count
healthStatus	Current suspect health status

Exit Codes

The probe script uses exit codes to signify various internal conditions. The exit code information can help you troubleshoot your scripts if they do not operate correctly. You can only use the **exit 5000** and **exit 5001** exit codes. A probe script indicates the relative health and availability of a real server using the exit code of the script. By calling exit (5000), a script indicates that the server successfully responded to the probe. Calling exit (5001) indicates that the server did not respond correctly to the health probe.

When a probe script fails and exits with 5001, the corresponding server is marked as PROBE_FAILED and is temporarily disabled from the server farm. The CSM-S continues to probe the server. When the probe successfully reconnects and exits with 5000, the CSM-S marks the server's status as OPERATIONAL and enables the server from the server farm again.

In addition to script exit 5001, these situations can cause a script to fail and mark the suspect PROBE_FAILED:

- TCL errors—Occurs when scripts contain errors that are caught by the TCL interpreter, for example, a syntax error. The syntax error message is stored in the special variable **erroInfo** and can be viewed using the **show mod csm X tech script** command.
- A stopped script—Caused by an infinite loop or caused when the script attempts to connect to an invalid IP address. Each script must complete its task within the configured time interval. If the script does not complete its task, the script controller terminates the script, and the suspect is failed implicitly.
- Error conditions—Occurs when a connection timeout or a peer-refused connection is also treated as an implicit failure.

Table 12-6 shows all exit codes used in the CSM-S.

Table 12-6 CSM-S Exit Codes

Exit Code	Meaning and Operational Effect on the Suspect
5000	Suspect is healthy. Controlled by user.
5001	Suspect has failed. Controlled by user.
4000	Script is aborted. The state change is dependent on other system status at that time. Reserved for system use.
4001	Script is terminated. Suspect is failed. Reserved for system use.
4002	Script panicked. Suspect is failed. Reserved for system use.
4003	Script has failed an internal operation or system call. Suspect is failed. Reserved for system use.
unknown	No change.

EXIT_MSG Variable

For debugging purposes, it is a good practice to set the script debug information in a special variable named EXIT_MSG. Using the EXIT_MSG variable, you can track the script execution point by entering specific Cisco IOS **show** commands.

This example shows how to use the EXIT_MSG variable to track script exit points to detect why a script is not working:

```
set EXIT_MSG "opening socket"
set s [socket 10.2.0.12 80]
set EXIT_MSG "writing to socket"
puts -nonewline $sock $url
```

Use the **show module csm** *slot* **tech script** command to check the EXIT_MSG variable.

This example shows that the EXIT_MSG was set to “opening socket” because EXIT_MSG is the last command that the script runs before the exit:

```
router1# show module csm 4 tech script
SCRIPT CONTROLLER STATS
: ==========================================================================
SCRIPT(0xcbcfb50) stat blk(0xcbcfbb0): TCL_test.tclcbcfb50
CMDLINE ARGUMENT:
 curr_id 1 argc 0 flag 0x0::
 type = PROBE
 task_id = 0x0: run_id = 512 ref count = 2
 task_status = TASK_DONE run status = OK
 start time = THU JAN 01 00:15:47 1970
 end time = THU JAN 01 00:17:02 1970
 runs = 1   +0
 resets = 1   +0
 notrel = 0   +0
 buf read err = 0   +0
 killed = 0   +0
 panicd = 0   +0
last exit status= 4000 last Bad status = 4000
Exit status history:
 Status (SCRIPT_ABORT) occured #(1) last@ THU JAN 01 00:17:02 1970
**TCL Controller:
------------------------
tcl cntrl flag = 0x7fffffff
#select(0) close_n_exit(0) num_sock(1)
 MEM TRACK last alloc(0) last size(0) alloc(0) size(0)
 hm_ver (1) flag(0x0) script buf(0xcbf8c00) new script buf(0x0)  lock owner(0x0) sig
taskdel:0 del:0 syscall:0 syslock:0 sig_select  script ptr (0xcbf88f0) id(0)
Config(0xcbcdd78) probe -> 10.1.0.105:80
 tclGlob(0xcbad050) script resource(0xcbcfa28)
#Selects(0) Close_n_exit(0) #Socket(1)
OPEN SOCKETS:
Last erroInfo = couldn't open socket: host is unreachable
    while executing
"socket 10.99.99.99 80 "
    (file "test.tcl" line 2)
Last errorCode = 65
Last panicInfo =
EXIT_MSG = opening socket
```

Running Probe Scripts

To run a probe script, you must configure a script probe type, and then associate a script name with the probe object (refer to the *Catalyst 6500 Series Content Switching Module Command Reference*).

To load, create, attach the script to a server farm and virtual server, run the probe scripts, and then display the results, perform theses steps:

Step 1 Load the script:

```
router1# conf t
Enter configuration commands, one per line.  End with CNTL/Z.
router1(config)# module csm 6
router1(config-module-csm)# script file tftp://192.168.10.102/csmTcl.tcl
Loading csmTcl.tcl from 192.168.10.102 (via Vlan100): !
[OK - 1933 bytes]
```

Step 2 Create a script probe:

```
router1(config-module-csm)# probe test1 script
rout(config-slb-probe-script)# script CSMTCL
rout(config-slb-probe-script)# interval 10
rout(config-slb-probe-script)# exit
```

Step 3 Attach the probe to the server farm and the virtual server:

```
router1(config-module-csm)# serverfarm test
router1(config-slb-sfarm)# real 10.1.0.105
router1(config-slb-real)# ins
router1(config-slb-real)# probe test1
router1(config-slb-sfarm)# exit
```

Step 4 Attach the server farm to a virtual server:

```
router1(config-module-csm)# vserver test
router1(config-slb-vserver)# virtual 10.12.0.80 tcp 80
router1(config-slb-vserver)# serverfarm test
router1(config-slb-vserver)# ins
router1(config-slb-vserver)# exit
```

At this point, the script probe should be set up. You can use the **show module csm slot tech probe** command to ensure that the scripts are running.

Step 5 Stop the script probe:

```
router1(config-module-csm)# serverfarm test
router1(config-slb-real)# no probe test1
router1(config-slb-sfarm)# exit
```

The examples that follow show how to verify the results of the script commands.

This example shows how to display script information:

```
router1# show module csm 6 script
CSMTCL, file tftp://192.168.10.102/csmTcl.tcl
  size = 1933, load time = 03:09:03 UTC 01/01/70
```

This example shows how to display information about probe scripts:

```
router1# show module csm 6 probe
probe           type    port  interval retries failed  open   receive
------------------------------------------------------------------------
TEST1           script        10       3       300     10     10
router1#
```

This example shows how to display detailed information about a specific probe script:

```
router1# show module csm 6 probe name TEST1 detail
probe           type    port  interval retries failed  open   receive
------------------------------------------------------------------------
TEST1           script        10       3       300     10     10
 Script: CSMTCL
 real                  vserver         serverfarm      policy          status
 ---------------------------------------------------------------------------
 10.1.0.105:80         TEST1           TEST            (default)       OPERABLE
router1#
```

This example shows how to display probe information for real servers:

```
router1# show module csm 6 probe real
  real = 10.1.0.105:80, probe = TEST1, type = script,
    vserver = TEST, sfarm = TEST
    status = FAILED, current = 03:26:04 UTC 01/01/70,
    successes = 1, last success = 03:15:33 UTC 01/01/70,
    failures = 4, last failure = 03:26:04 UTC 01/01/70,
    state = Unrecognized or invalid response
    script CSMTCL
    last exit code = 5001
```

Debugging Probe Scripts

To debug a script probe, you can do the following:

- Use the TCL **puts** command in the scripts running in verbose mode.

  In the verbose mode, the **puts** command causes each probe suspect to print a string to the CSM-S console. When there are many suspects running on the system, lots of output resources are required or the CSM-S console might hang. It is very important to make sure that this feature is enabled only when a single suspect is configured on the system.

- Use the special variable EXIT_MSG in the script.

  Each probe suspect contains its own EXIT_MSG variable. This variable allows you to trace the status of a script and check the status of the probe.

  This example shows how to use the EXIT_MSG variable in a script:

  ```
  set EXIT_MSG "before opening socket"
  set s [ socket $ip $port]
  set EXIT_MSG " before receive string"
  gets $s
  set EXIT_MSG "before close socket"
  close $s
  ```

  If a probe suspect fails when receiving the message, you should see EXIT_MSG = before you receive the string.

- Use the **show module csm** *slot* **probe real** [*ip*] command.

  This command shows you the current active probe suspects in the system:

  ```
  router1# show module csm 6 probe real
    real = 10.1.0.105:80, probe = TEST1, type = script,
      vserver = TEST, sfarm = TEST
      status = FAILED, current = 04:06:05 UTC 01/01/70,
      successes = 1, last success = 03:15:33 UTC 01/01/70,
      failures = 12, last failure = 04:06:05 UTC 01/01/70,
      state = Unrecognized or invalid response
      script CSMTCL
      last exit code = 5001
  ```

Note The last exit code displays one of the exit codes listed in Table 12-6 on page 12-10.

- Use the **show module csm** *slot* **tech probe** command.

  This command shows the current probe status (for both the standard and script probe):

```
router1# show module csm 6 tech probe

Software version: 3.2(1)
------------------------------------------------------------
------------------ Health Monitor Statistics ------------
------------------------------------------------------------
 Probe templates: 1
Suspects created: 1
    Open Sockets in System : 8 / 240
    Active Suspect(no ICMP): 0 / 200
    Active Script Suspect  : 0 / 50
    Num events  : 1

Script suspects: 1
  Healthy suspects: 0
Failures suspected: 0
Failures confirmed: 1

Probe attempts:        927  +927
Total recoveries:        3  +3
Total failures:          6  +6
Total Pending:           0  +0
```

- Use the **show module csm** *slot* **tech script** command, and look for the last exit status, persistent variables, errorInfo, and EXIT_MSG output.

```
router1# show module csm 6 tech script
SCRIPT(0xc25f7e0) stat blk(0xc25f848): TCL_csmTcl.tclc25f7e0
CMDLINE ARGUMENT:
 curr_id 1 argc 0 flag 0x0::
 type = PROBE
 task_id = 0x0: run_id = 521 ref count = 2
 task_status = TASK_DONE run status = OK
 start time = THU JAN 01 03:51:04 1970
 end time = THU JAN 01 03:51:04 1970

 runs = 13   +11
 resets = 13   +11
 notrel = 0   +0
 buf read err = 1   +1
 killed = 0   +0
 panicd = 0   +0

last exit status= 5001 last Bad status = 5001

Exit status history:

**TCL Controller:
-----------------------
tcl cntrl flag = 0x7fffffff
#select(0) close_n_exit(0) num_sock(2)
 MEM TRACK last alloc(0) last size(0) alloc(0) size(0)
 hm_ver (3) flag(0x0) script buf(0xc25ad80) new script buf(0xc25ad80)
 lock owner(0x0) sig taskdel:0 del:0 syscall:0 syslock:0 sig_select
 script ptr (0xc25f038) id(0)
Config(0xc2583d8) probe -> 10.1.0.105:80
 tclGlob(0xc257010)
SCRIPT RESOURCE(0xc25af70)-------
#Selects(0) Close_n_exit(0) #Socket(2)
OPEN SOCKETS:
```

```
Persistent Variables

--------------------
   x = 11

Last erroInfo =

Last errorCode =
Last panicInfo =
EXIT_MSG = ping failed : invalid command name "ping"
```

The last exit status displays the exit code number as shown in Table 12-6 on page 12-10.

The Persistent Variables information is set by the **gset varname value** command (as described in the "CSM-S Specific TCL Commands" section on page 12-5).

The erroInfo lists the error that is generated by the TCL compiler. When the script has a TCL runtime error, the TCL interpreter stops running the script and stores the error information in the erroInfo variable.

The EXIT_MSG (see the "EXIT_MSG Variable" section on page 12-10) displays detailed debug information for each probe suspected of failure. Because the output may be lengthy, you can try to filter the keyword first as shown in this example:

```
router1# show module csm slot tech script inc keyword
```

Standalone Scripts

A standalone script is a generic TCL script that loads and runs in the CSM-S. Because the standalone script is not configured like the probe script, and it is not attached to a server farm, the script will not be scheduled by the CSM-S as a periodically run task. To run the task, you must use the **script task** command.

The csm_env environment variables are not applied to a standalone script. You may use the **exit** command, however, if the exit code does not have special meaning for standalone scripts as it does in the probe script.

Example for Writing Standalone Scripts

This example shows how a generic TCL script can be written:

```
#!name = STD_SCRIPT
set gatewayList "1.1.1.1 2.2.2.2"
foreach gw $gatewayList {
    if { ![ ping $gw ]  } {
    puts "-WARNING : gateway $gw is down!!"
  }
}
```

Running Standalone Scripts

A standalone script is a TCL script that will be run once as a single task unlike script probes. The script will run and exit when it is finished. The standalone script will not be run by the CSM-S periodically unless you configure this script as a task. The **script file** command may be stored in the startup configuration so that it will run when the CSM-S boots. The script continues to run while the CSM-S is operating.

To run standalone scripts, perform these steps:

Step 1 Load the script:

```
router1# conf t
Enter configuration commands, one per line.  End with CNTL/Z.
router1(config)# module csm 6
router1(config-module-csm)# script file tftp://192.168.10.102/stdcsm.tcl
Loading stdcsm.tcl from 192.168.10.102 (via Vlan100): !
[OK - 183 bytes]
```

Step 2 Run the script as a standalone task:

```
router1(config-module-csm)# script task 1 STD_SCRIPT
```

Step 3 Rerun the script:

You can remove the old task and run it again as follows:

```
router1(config-module-csm)# no script task 1 STD_SCRIPT
router1(config-module-csm)# script task 1 STD_SCRIPT
```

You also can start a new task by giving it a new task ID as follows:

```
router1(config-module-csm)# script task 2 STD_SCRIPT
```

Step 4 Stop the script:

```
router1(config-module-csm)# no script task 1 STD_SCRIPT
```

Step 5 Use the **show** command to display the status of the script:

```
router1#sh mod csm 6 script
STD_SCRIPT, file tftp://192.168.10.102/stdcsm.tcl
router1#sh mod csm 6 script task

task  script                          runs       exit code  status
--------------------------------------------------------------------------
1     STD_SCRIPT                      1          4000       Not Ready
2     STD_SCRIPT                      1          4000       Not Ready
```

To display information about a specific running script, use the **show module csm** *slot* **script task index** *script-index* **detail** or the **show module csm** *slot* **script name** *script-name* **code** commands.

Debugging Standalone Scripts

Debugging a standalone script is similar to debugging a probe script. See the "Debugging Probe Scripts" section on page 12-13. You can use the **puts** command in the script to help debugging because running multiple threads do not cause problems.

TCL Script Frequently Asked Questions (FAQs)

These are some frequently asked questions about TCL scripting for the CSM-S:

- How are system resources used?

  The Vxworks support application has 255 file descriptors that are divided across all applications, for example, standard input and output, and any socket connections (to or from). When developing standalone scripts, you must be extremely careful when opening a socket. We recommend that you close a socket as soon as the operation is complete because you may run out of resources. The health monitoring module controls the number of open sockets by controlling the number of actively running scripts. Standalone scripts do not have this control.

  Memory, although a consideration, is not a big limiting factor because the module generally has enough memory available. Each script uses a 128-KB stack, and the rest of the memory is allocated at runtime by the script.

  The script tasks are given the lowest priority in the system so that the real-time characteristics of the system remain more or less the same while executing scripts. Unfortunately, scripts that have low priority also mean that if the system is busy doing non-TCL operations, all TCL threads may take longer to complete. This situation may lead to some health scripts being terminated and the unfinished threads marked as failed. To prevent scripts being failed, all script probes should have a retry value of 2 or more. You may want to use native CSM-S probes (for example, HTTP or DNS) whenever possible. The scripted health probes should be used to support unsupported applications.

  TCL supports both synchronous and asynchronous socket commands. Asynchronous socket commands return immediately without waiting for true connections. The internal implementation of the asynchronous script version involves a much more complicated code path with many more system calls per each such command. This condition generally slows down the system by causing some critical resources to wait while other commands are processing system calls. We do not recommend using the asynchronous socket for scripted probes unless this is a definite requirement. However, you may use this command in a standalone system.

- How do I know if a configured probe is running?

  You can run a sniffer on the real server side of the network. Also, you can use the following **show** commands to determine if probes are running on the CSM-S.

  - If the probe is running, the number of probe attempts should keep increasing as shown in this example:

```
router1# show module csm 6 tech probe
router1#sh mod csm 6 tech probe
Software version: 3.2(1)
------------------------------------------------------------
------------------ Health Monitor Statistics -----------------
------------------------------------------------------------
 Probe templates: 8
Suspects created: 24
    Open Sockets in System : 10 / 240
    Active Suspect(no ICMP): 2 / 200
    Active Script Suspect  : 2 / 50
    Num events  : 24
Script suspects: 24
  Healthy suspects: 16
Failures suspected: 0
Failures confirmed: 8
Probe attempts:        321  +220
Total recoveries:       16  +0
Total failures:          8  +2
Total Pending:           0  +0
```

– If the probe is running, the success or failures count should increase as shown in this example:

```
router1# show module csm 6 probe real
 real = 10.12.0.108:50113, probe = SCRIPT2_2, type = script,
   vserver = SPB_SCRIPT2, sfarm = SCRIPT2_GOOD, policy = SCRIPT2_GOOD,
   status = OPERABLE, current = 22:52:24 UTC 01/04/70,
   successes = 18, last success = 22:52:24 UTC 01/04/70,
   failures = 0, last failure = 00:00:00 UTC 01/01/70,
   state = Server is healthy.
   script httpProbe2.tcl GET /yahoo.html html 1.0 0
   last exit code = 5000
 real = 10.12.0.107:50113, probe = SCRIPT2_2, type = script,
   vserver = SPB_SCRIPT2, sfarm = SCRIPT2_GOOD, policy = SCRIPT2_GOOD,
   status = OPERABLE, current = 22:52:42 UTC 01/04/70,
   successes = 19, last success = 22:52:42 UTC 01/04/70,
   failures = 0, last failure = 00:00:00 UTC 01/01/70,
   state = Server is healthy.
   script httpProbe2.tcl GET /yahoo.html html 1.0 0
   last exit code = 5000
```

You can also close the socket using FIN in place of reset (RST).

- Why does the UDP probe fail to put the real server in the PROBE_FAIL state when a remote host is unreachable?

  A UDP probe must receive an “icmp port unreachable” message to mark a server as PROBE_FAIL. When a remote host is down or not responding, the UDP probe does not receive the ICMP message and the probe assumes that the packet is lost and the server is healthy.

  Because the UDP probe is a raw UDP probe, the CSM-S is using a single byte in the payload for probe responses. The CSM-S does not expect any meaningful response from the UDP application. The CSM-S uses the ICMP Unreachable message to determine if the UDP application is not reachable.

  If there is no ICMP unreachable reply in the receive timeout, the CSM-S assumes that the probe is operating correctly. If the IP interface of the real server is down or disconnected, the UDP probe by itself would not know that the UDP application is not reachable. You must configure the ICMP probe in addition to the UDP probe for any given server.

  Workaround: Always configure ICMP with a UDP type of probe.

- Where can I find a script example to download?

  Sample scripts are available to support the TCL feature. Other custom scripts will work, but these sample scripts are supported by Cisco TAC. The file with sample scripts is located at this URL:

  http://www.cisco.com/cgi-bin/tablebuild.pl/cat6000-intellother

  The file containing the scripts is named: c6slb-script.3-3-1.tcl.

- Where can I find TCL scripting information?

  The TCL 8.0 command reference is located at this URL:

  http://www.tcl.tk/man/tcl8.0/TclCmd/contents.html

  The TCL UDP command reference is located at this URL:

  http://wwwhome.cs.utwente.nl/~schoenw/scotty/

CHAPTER 13

Configuring Firewall Load Balancing

This chapter describes how to configure firewall load balancing and contains these sections:

Firewall load balancing allows you to scale firewall protection by distributing traffic across multiple firewalls on a per-connection basis. All packets belonging to a particular connection must go through the same firewall. The firewall then allows or denies transmission of individual packets across its interfaces.

Understanding How Firewalls Work

A firewall forms a physical barrier between two parts of a network, for example, the Internet and an intranet. When a firewall accepts a packet from one side (the Internet), it sends the packet through to the other side (the intranet). A firewall can modify a packet before passing it through or sending it through unaltered. When a firewall rejects a packet, it usually drops the packet and logs the dropped packet as an event.

After a session is established and a flow of packets begins, a firewall can monitor each packet in the flow or allow the flow to continue, unmonitored, depending on the policies that are configured on that firewall.

This section contains the following:

Firewall Types

The two basic types of firewalls are as follows:

- Regular firewalls
- Stealth firewalls

Regular firewalls have a presence on the network; they are assigned an IP address that allows them to be addressed as a device and seen by other devices on the network.

Stealth firewalls have no presence on the network; they are not assigned an IP address and cannot be addressed or seen by other devices on the network. To the network, a stealth firewall is part of the wire.

Both firewall types examine traffic moving in both directions (between the protected and the unprotected side of the network) and accept or reject packets based on user-defined sets of policies.

How the CSM-S Distributes Traffic to Firewalls

The CSM-S load balances traffic to devices configured in server farms. These devices can be servers, firewalls, or any IP-addressable object including an alias IP address. The CSM-S uses load-balancing algorithms to determine how the traffic is balanced among the devices configured in server farms, independent of device type.

Note We recommend that you configure Layer 3 load balancing on server farms that contain firewalls because of the interactions between higher-layer load-balancing algorithms and server applications.

Supported Firewalls

The CSM-S can load balance traffic to regular or stealth firewalls.

For regular firewalls, a single CSM-S or a pair of CSMs balances traffic among firewalls that contain unique IP addresses, similar to how the CSM-S balances traffic to servers.

For stealth firewalls, a CSM-S balances traffic among unique VLAN alias IP address interfaces on another CSM-S that provides paths through stealth firewalls. A stealth firewall is configured so that all traffic moving in both directions across that VLAN moves through the firewall.

Layer 3 Load Balancing to Firewalls

When the CSM-S load balances traffic to firewalls, the CSM-S performs the same function that it performs when it load balances traffic to servers. To configure Layer 3 load balancing to firewalls, follow these steps:

Step 1 Create a server farm for each side of the firewall.

Step 2 In serverfarm submode, enter the predictor **hash address** command.

Step 3 Assign that server farm to the virtual server that accepts traffic destined for the firewalls.

Note When you configure Layer 3 load balancing to firewalls, use source NAT in the forward direction and destination NAT in the reverse direction.

Types of Firewall Configurations

The CSM-S supports these two firewall configuration types:

- Dual-CSM-S configuration—Firewalls are located between two CSM modules. The firewalls accept traffic from one CSM-S and send it to a second CSM-S for load balancing to servers or return to the requesting device.
- Single-CSM-S configuration—Firewalls accept traffic from a CSM-S and send it back to the same CSM-S for load balancing to servers, or they can return traffic to the requesting device.

IP Reverse-Sticky for Firewalls

The CSM-S currently supports sticky connections. Sticky connections ensure that two distinct data flows originating from the same client are load balanced to the same destination.

Load-balanced destinations are often real servers. They may be firewalls, caches, or other networking devices. Sticky connections are necessary for the proper functioning of load-balanced applications. These applications utilize multiple connections from the same client to a server. The information transferred on one connection may affect the processing of information transferred on another connection.

The IP reverse-sticky feature is configured for balancing new connections from the same client to the same server, as described in the "Configuring Reverse-Sticky for Firewalls" section on page 13-24. This feature is especially important in the case of buddy connections, such as an FTP data channel or a streaming UDP data channel.

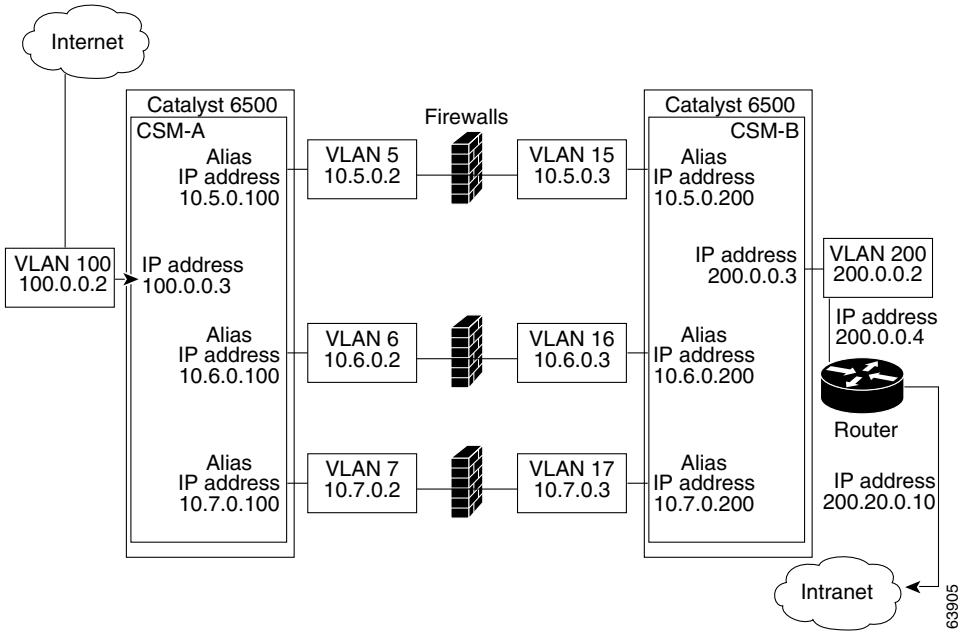
CSM-S Firewall Configurations

The CSM-S can support these firewall configurations:

- Stealth firewalls for dual CSM-S configurations (Figure 13-1)
- Regular firewalls for dual CSM-S configurations (Figure 13-2)
- Regular firewalls for single CSM-S configurations (Figure 13-3)
- Mixed firewalls (stealth and regular) for dual CSM-S configurations (Figure 13-4)

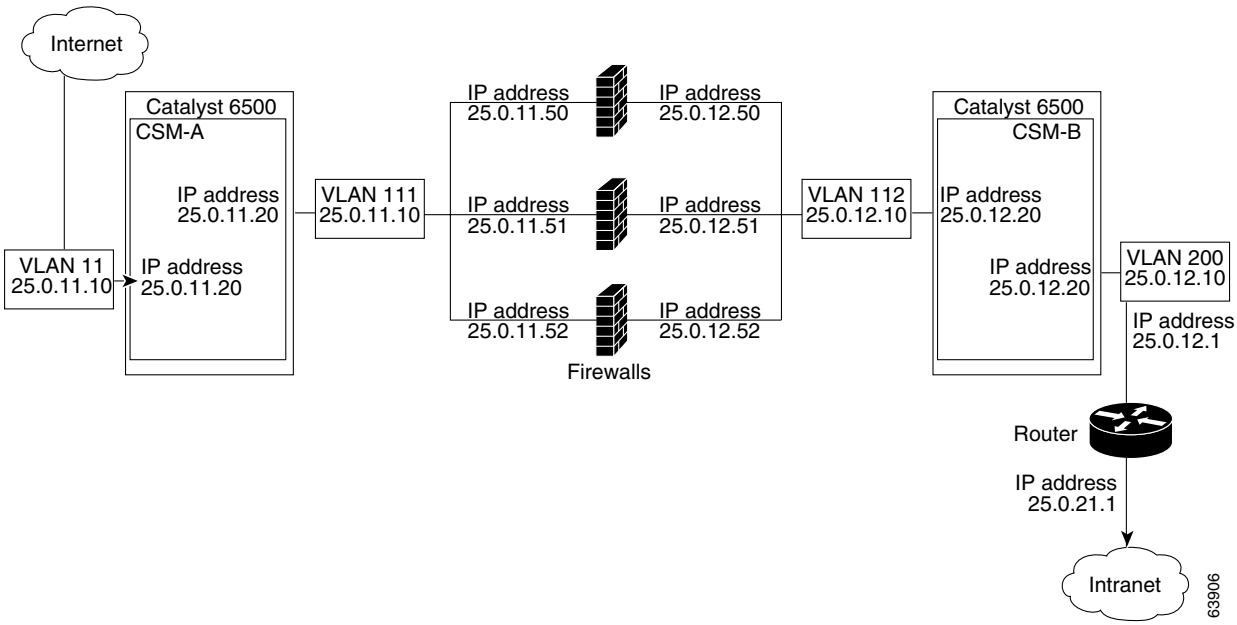
In Figure 13-1, traffic moves through the firewalls and is filtered in both directions. The figure shows the flow from the Internet to the intranet. On the path to the intranet, CSM-S A balances traffic across VLANs 5, 6, and 7 through firewalls to CSM-S B. On the path to the Internet, CSM-S B balances traffic across VLANs 15, 16, and 17 through firewalls to CSM-S A. CSM-S A uses the VLAN aliases of CSM-S B in its server farm, and CSM-S B uses the VLAN aliases of CSM-S A in its server farm.

Figure 13-1 Stealth Firewall Configuration (Dual CSM-S modules Only)



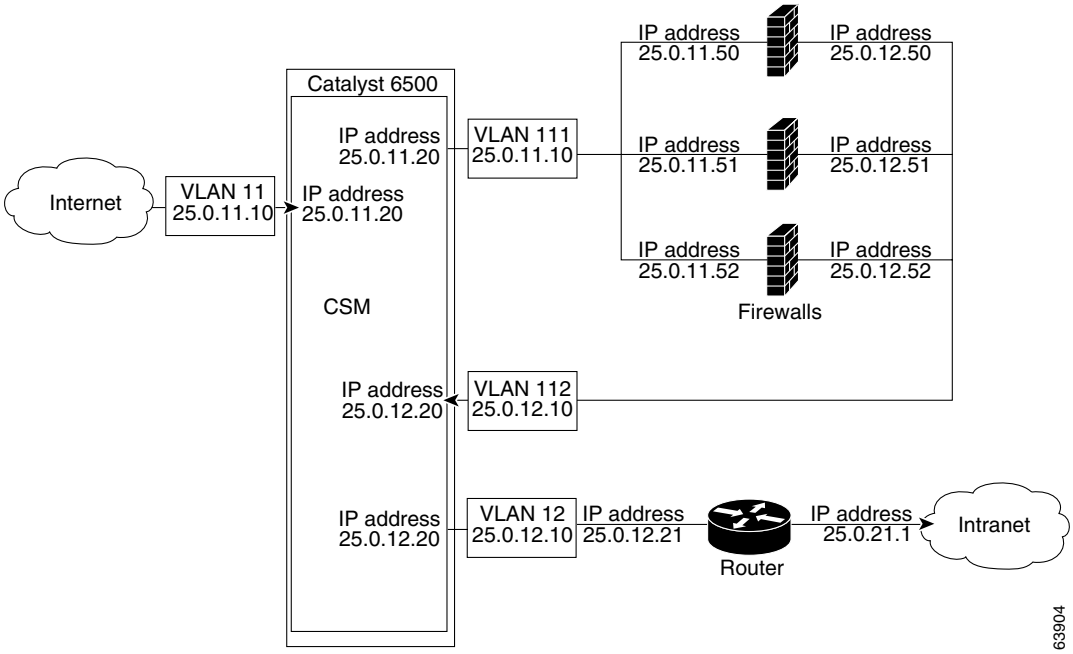
In Figure 13-2, traffic moves through the firewalls and is filtered in both directions. The figure shows the flow from the Internet to the intranet. VLANs 11 and 111 are on the same subnet, and VLANs 12 and 112 are on the same subnet.

Figure 13-2 Regular Firewall Configuration (Dual CSM-S modules)



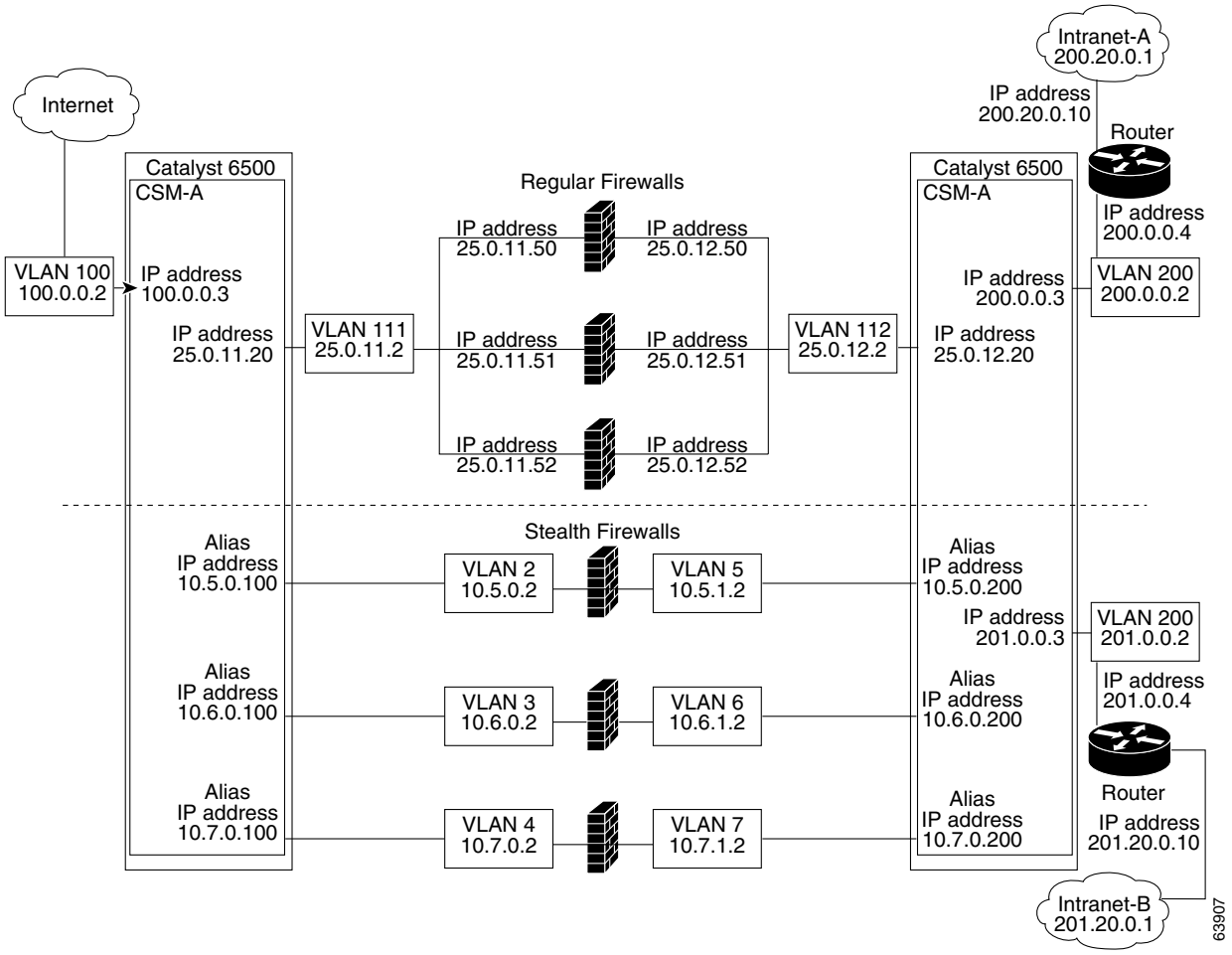
In Figure 13-3, traffic moves through the firewalls and is filtered in both directions. The figure shows only the flow from the Internet to the intranet, and VLANs 11 and 111 are on the same subnet. VLANs 12 and 112 are on the same subnet.

Figure 13-3 Regular Firewall Configuration (Single CSM-S)



In Figure 13-4, traffic moves through both the regular and stealth firewalls and is filtered in both directions. The figure shows the flow from the Internet to the intranet. VLANs 5, 6, and 7 are shared between CSM-S A and CSM-S B. On the path to the intranet, CSM-S A balances traffic across VLANs 5, 6, and 7 through firewalls to CSM-S B. On the path to the intranet, CSM-S B balances traffic across VLANs 5, 6, and 7 through firewalls to CSM-S A.

Figure 13-4 Mixed Firewall Configuration for Stealth and Regular Firewalls (Dual CSM-S Only)



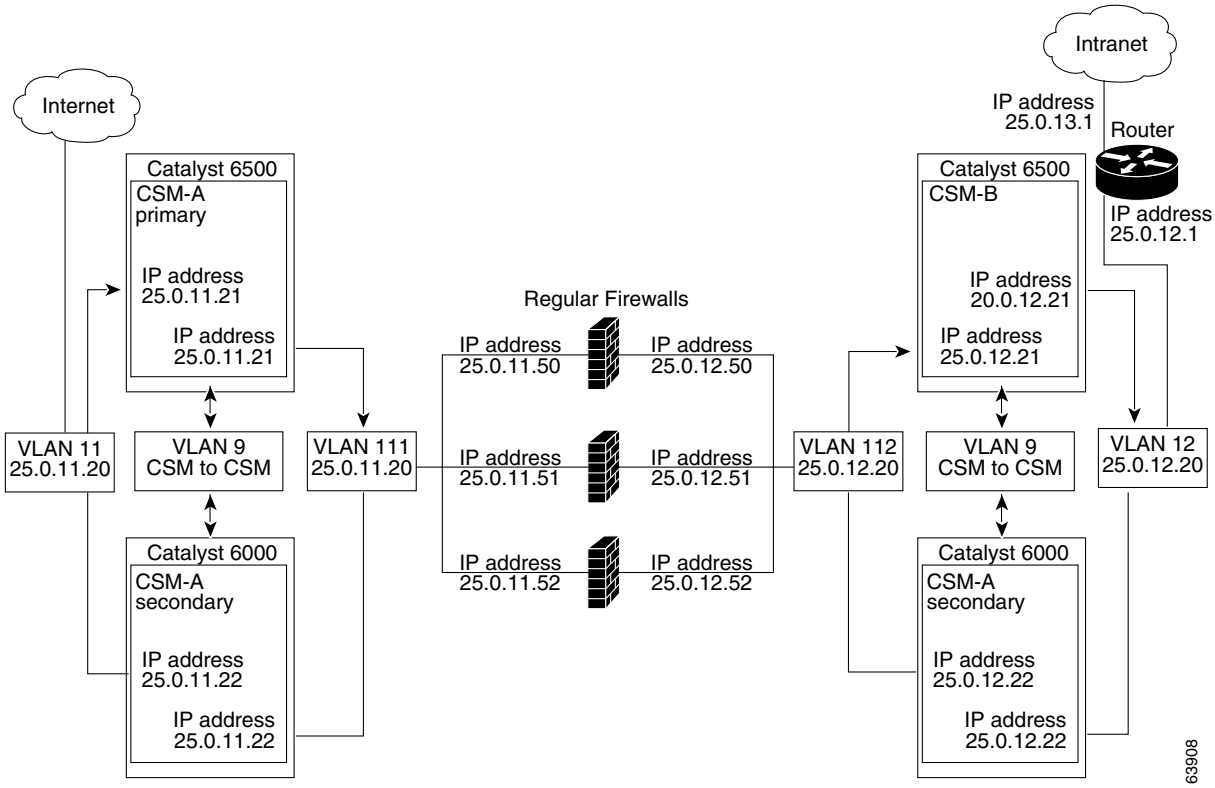
Fault-Tolerant CSM-S Firewall Configurations

The CSM-S supports fault tolerance for these configurations:

- Stealth firewalls in a fault-tolerant dual CSM-S configuration
- Regular firewalls in a fault-tolerant dual CSM-S configuration
- Regular firewalls in a fault-tolerant single CSM-S configuration
- Mixed firewalls (stealth and regular) in a fault-tolerant dual CSM-S configuration

In Figure 13-5, the traffic moves through the firewalls and is filtered in both directions. The figure only shows the flow from the Internet to the intranet through the primary CSMs, and VLANs 11 and 111 are on the same subnet. VLANs 12 and 112 are on the same subnet.

Figure 13-5 Fault-Tolerant, Regular Firewall Configuration–(Dual CSMs)



Configuring Stealth Firewall Load Balancing

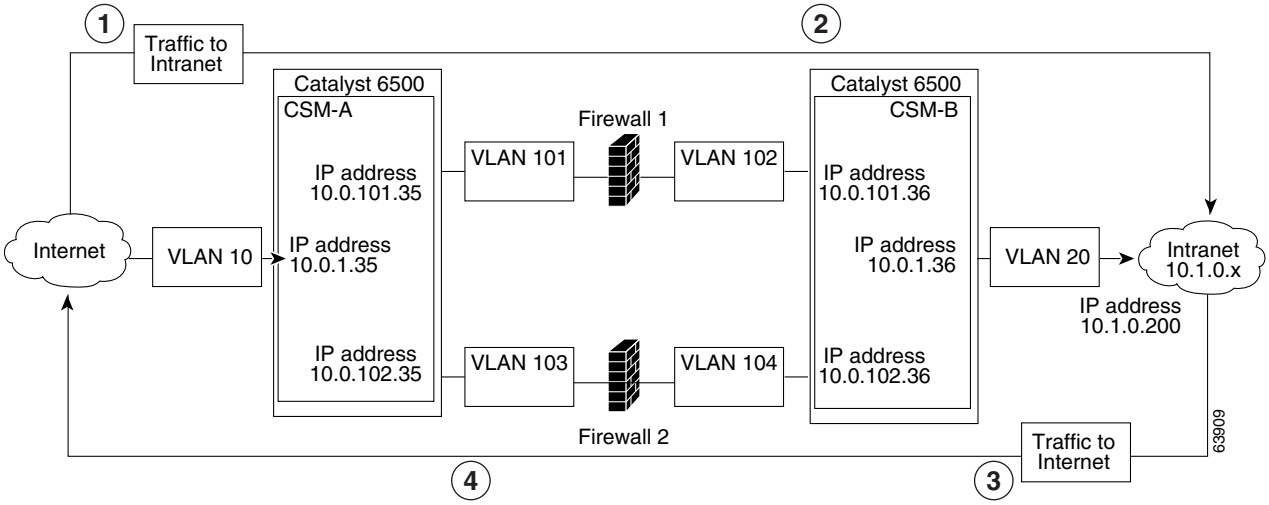
This section describes how to configure firewall load balancing for stealth firewalls and covers the following information:

- Stealth Firewall Configuration, page 13-7
- Stealth Firewall Configuration Example, page 13-8

Stealth Firewall Configuration

In a stealth firewall configuration, firewalls connect to two different VLANs and are configured with IP addresses on the VLANs to which they connect. (See Figure 13-6.)

Figure 13-6 Stealth Firewall Configuration Example



Location	Traffic Direction	Arrives On	Exits On
1	To intranet	VLAN 10	VLANs 101 and 103
2	To intranet	VLANs 101 and 103	VLAN 20
3	To Internet	VLAN 20	VLANs 102 and 104
4	To Internet	VLANs 101 and 103	VLAN 10

Figure 13-6 shows two regular firewalls (Firewall 1 and Firewall 2) located between two CSM modules (CSM-S A and CSM-S B).

Note Stealth firewalls do not have addresses on VLANs.

On the path from the Internet to the intranet, traffic enters the insecure side of the firewalls through separate VLANs, VLAN 101 and VLAN 103, and exits the secure side of the firewalls through separate VLANs, VLAN 102 and VLAN 104. On the path from the intranet to the Internet, the flow is reversed. VLANs also provide connectivity to the Internet (VLAN 10) and to the intranet (VLAN 20).

In a stealth configuration, CSM-S A and CSM-S B load balance traffic through the firewalls.

Stealth Firewall Configuration Example

The stealth firewall configuration example contains two CSM-S modules (CSM-S A and CSM-S B) installed in separate Catalyst 6500 series switches.

Note In a stealth firewall configuration, each CSM-S must be installed in a separate Catalyst 6500 series switch.

This section describes how to create the stealth firewall configuration for CSM-S A and CSM-S B.

Configuring CSM-S A (Stealth Firewall Example)

To create the regular configuration example, perform these tasks for CSM-S A:

- Creating VLANs on Switch A, page 13-9
- Configuring VLANs on CSM-S A, page 13-9
- Configuring Server Farms on CSM-S A, page 13-10
- Configuring Virtual Servers on CSM-S A, page 13-11

Note Although the configuration tasks are the same for both the CSM-S A and CSM-S B, the steps, commands, and parameters that you enter are different.

Creating VLANs on Switch A

To create two VLANs on Switch A, perform this task:

	Command	Purpose
Step 1	`Switch-A(config)# vlan`	Enters the VLAN mode[1].
Step 2	`Switch-A(vlan)# vlan 10`	Creates VLAN 10[2].
Step 3	`Switch-A(vlan)# vlan 101`	Creates VLAN 101[3].
Step 4	`Switch-A(vlan)# vlan 103`	Creates VLAN 103[4].

1. Perform this step on the switch console of the switch that contains CSM-S A.
2. VLAN 10 connects CSM-S A to the Internet.
3. VLAN 101 provides a connection through Firewall 1 to CSM-S B.
4. VLAN 103 provides a connection through Firewall 2 to CSM-S B.

Configuring VLANs on CSM-S A

To configure the three VLANs, perform this task:

	Command	Purpose
Step 1	`Switch-A(config)# module csm 5`	Enters multiple module configuration mode and specifies that CSM-S A is installed in slot 5.
Step 2	`Switch-A(config-module-csm)# vlan 10 client`	Specifies VLAN 10 as the VLAN that is being configured, identifies it as a client VLAN, and enters VLAN configuration mode.
Step 3	`Switch-A(config-slb-vlan-client)# ip address 10.0.1.35 255.255.255.0`	Specifies an IP address and netmask for VLAN 10.
Step 4	`Switch-A(config-slb-vlan-client)# alias 10.0.1.30 255.255.255.0`	Specifies an alias IP address and netmask for VLAN 10[1].
Step 5	`Switch-A(config-slb-vlan-client)# exit`	Returns to VLAN configuration mode.
Step 6	`Switch-A(config-module-csm)# vlan 101 server`	Specifies VLAN 101 as the VLAN that is being configured, identifies it as a server VLAN, and enters VLAN configuration mode.

	Command	Purpose
Step 7	`Switch-A(config-slb-vlan-server)# ip address 10.0.101.35 255.255.255.0`	Specifies an IP address and netmask for VLAN 101.
Step 8	`Switch-A(config-slb-vlan-server)# alias 10.0.101.100 255.255.255.0`	Specifies an alias IP address and netmask for VLAN 101[1].
Step 9	`Switch-A(config-slb-vlan-server)# exit`	Returns to VLAN configuration mode.
Step 10	`Switch-A(config-module-csm)# vlan 103 server`	Specifies VLAN 103 as the VLAN that is being configured, identifies it as a server VLAN, and enters VLAN configuration mode.
Step 11	`Switch-A(config-slb-vlan)# ip address 10.0.102.35 255.255.255.0`	Specifies an IP address and netmask for VLAN 103.
Step 12	`Switch-A(config-slb-vlan)# alias 10.0.102.100 255.255.255.0`	Specifies an alias IP address and netmask for VLAN 103[1].

1. This step provides a target for CSM-S B to use in making a load-balancing decision.

Configuring Server Farms on CSM-S A

Note Because the IP addresses of CSM-S B are listed in the INSIDE-SF server farm as real servers, CSM-S A will load balance the two firewalls that exist in the path to CSM-S B.

To configure two server farms on CSM-S A, perform this task:

	Command	Purpose
Step 1	`Switch-A(config)# module csm 5`	Enters multiple module configuration mode and specifies that CSM-S A is installed in slot 5.
Step 2	`Switch-A(config-module-csm)# serverfarm FORWARD-SF`	Creates and names the FORWARD-SF[1] server farm (actually a forwarding policy) and enters serverfarm configuration mode.
Step 3	`Switch-A(config-slb-sfarm)# no nat server`	Disables the NAT of server IP addresses and port numbers[2].
Step 4	`Switch-A(config-slb-sfarm)# predictor forward`	Forwards traffic in accordance with its internal routing tables rather than a load-balancing algorithm.
Step 5	`Switch-A(config-slb-sfarm)# exit`	Returns to multiple module configuration mode.
Step 6	`Switch-A(config-module-csm)# serverfarm TO-INSIDE-SF`	Creates and names the INSIDE-SF[3] server farm (that will contain alias IP addresses rather than real servers) and enters serverfarm configuration mode.
Step 7	`Switch-A(config-slb-sfarm)# no nat server`	Disables the NAT of the server IP address and port number[4].
Step 8	`Switch-A(config-slb-sfarm)# predictor hash address source 255.255.255.255`	Selects a server using a hash value based on the source IP address[5].
Step 9	`Switch-A(config-slb-sfarm)# real 10.0.101.200`	Identifies the alias IP address of CSM-S B that lies on the path to Firewall 1 as a real server and enters real server configuration submode.

	Command	Purpose
Step 10	`Switch-A(config-slb-real)# inservice`	Enables the firewall.
Step 11	`Switch-A(config-slb-real)# exit`	Returns to serverfarm configuration mode.
Step 12	`Switch-A(config-slb-sfarm)# real 10.0.102.200`	Identifies the alias IP address of CSM-S B that lies on the path to Firewall 2 as a real server and enters real server configuration submode.
Step 13	`Switch-A(config-slb-real)# inservice`	Enables the firewall.

1. FORWARD-SF is actually a route forwarding policy, not an actual server farm, that allows traffic to reach the Internet (through VLAN 10). It does not contain any real servers.
2. This step is required when configuring a server farm that contains a forwarding policy rather than real servers.
3. INSIDE-SF contains the two alias IP addresses of CSM-S B listed as real servers that allow traffic from the intranet to reach CSM-S B.
4. This step is required when configuring a server farm that contains firewalls.
5. We recommend that you perform this step when configuring insecure-side firewall interfaces in a server farm.

Configuring Virtual Servers on CSM-S A

To configure three virtual servers on CSM-S A, perform this task:

	Command	Purpose
Step 1	`Switch-A(config)# module csm 5`	Enters multiple module configuration mode and specifies that the CSM-S A is installed in slot 5.
Step 2	`Switch-A(config-module-csm)# vserver FORWARD-V101`	Specifies FORWARD-V101[1] as the virtual server that is being configured and enters virtual server configuration mode.
Step 3	`Switch-A(config-slb-vserver)# virtual 0.0.0.0 0.0.0.0 any`	Specifies a match for any IP address and any protocol[2].
Step 4	`Switch-A(config-slb-vserver))# vlan 101`	Specifies that the virtual server will only accept traffic arriving on VLAN 101, which is traffic arriving from the insecure side of the firewalls.
Step 5	`Switch-A(config-slb-vserver)# serverfarm FORWARD-SF`	Specifies the server farm for this virtual server[3].
Step 6	`Switch-A(config-slb-vserver)# inservice`	Enables the virtual server.
Step 7	`Switch-A(config-slb-vserver)# exit`	Returns to multiple module configuration mode.
Step 8	`Switch-A(config-module-csm)# vserver FORWARD-V103`	Specifies FORWARD-V103[4] as the virtual server that is being configured and enters virtual server configuration mode.
Step 9	`Switch-A(config-slb-vserver)# virtual 0.0.0.0 0.0.0.0 any`	Specifies a match for any IP address and any protocol[5].
Step 10	`Switch-A(config-slb-vserver))# vlan 103`	Specifies that the virtual server will only accept traffic arriving on VLAN 103, which is traffic arriving from the insecure side of the firewalls.
Step 11	`Switch-A(config-slb-vserver)# serverfarm FORWARD-SF`	Specifies the server farm for this virtual server[3].
Step 12	`Switch-A(config-slb-vserver)# inservice`	Enables the virtual server.
Step 13	`Switch-A(config-slb-vserver)# exit`	Returns to multiple module configuration mode.

	Command	Purpose
Step 14	`Switch-A(config-module-csm)# vserver OUTSIDE-VS`	Specifies OUTSIDE-VS[6] as the virtual server that is being configured and enters virtual server configuration mode.
Step 15	`Switch-A(config-slb-vserver)# virtual 10.1.0.0 255.255.255.0 any`	Specifies the IP address, netmask, and protocol (any) for this virtual server. Clients reach the server farm represented by this virtual server through this address.
Step 16	`Switch-A(config-slb-vserver))# vlan 10`	Specifies that the virtual server will only accept traffic arriving on VLAN 10, which is traffic arriving from the Internet.
Step 17	`Switch-A(config-slb-vserver)# serverfarm TO-INSIDE-SF`	Specifies the server farm for this virtual server[7].
Step 18	`Switch-A(config-slb-vserver)# inservice`	Enables the virtual server.

1. FORWARD-V101 allows Internet traffic to reach the insecure side of the firewalls (through VLAN 101).
2. Client matching is only limited by VLAN restrictions. (See Step 4.)
3. This server farm is actually a forwarding predictor rather than an actual server farm containing real servers.
4. FORWARD-V103 allows Internet traffic to reach the insecure side of the firewalls (through VLAN 103).
5. Clients will always match (see Step 9)–only being limited by VLAN restrictions. (See Step 10.)
6. OUTSIDE-VS allows traffic from the Internet to reach CSM-S A (through VLAN 10).
7. The server farm contains the alias IP addresses of CSM-S B that lie along the path of Firewall 1 and Firewall 2.

Configuring CSM-S B (Stealth Firewall Example)

To create the regular configuration example, perform the following configuration tasks for CSM-S B:

- Creating VLANs on Switch B, page 13-12
- Configuring VLANs on CSM-S B, page 13-13
- Configuring Server Farms on CSM-S B, page 13-13
- Configuring Virtual Servers on CSM-S B, page 13-15

Note Although the configuration tasks are the same for both CSM-S A and CSM-S B, the steps, commands, and parameters that you enter are different.

Creating VLANs on Switch B

To create three VLANs on Switch B, perform this task:

Note This example assumes that the CSM-S modules are in separate Catalyst 6500 series switches. If they are in the same chassis, you can create all of the VLANs on the same Catalyst 6500 series switch console.

	Command	Purpose
Step 1	`Switch-B(config)# vlan`	Enters the VLAN mode[1].
Step 2	`Switch-B(vlan)# vlan 102`	Creates VLAN 102[2].

	Command	Purpose
Step 3	`Switch-B(vlan)# vlan 104`	Creates VLAN 104[3].
Step 4	`Switch-B(vlan)# vlan 200`	Creates VLAN 200[4].

1. Do this step on the switch console of the switch that contains CSM-S B.
2. VLAN 102 provides a connection through Firewall 1 to CSM-S A.
3. VLAN 104 provides a connection through Firewall 2 to CSM-S A.
4. VLAN 200 provides the connection to the internal network.

Configuring VLANs on CSM-S B

To configure the three VLANs, perform this task:

	Command	Purpose
Step 1	`Switch-B(config)# module csm 6`	Enters multiple module configuration mode and specifies that CSM-S B is installed in slot 6.
Step 2	`Switch-B(config-module-csm)# vlan 102 server`	Specifies VLAN 102 as the VLAN that is being configured, identifies it as a server VLAN, and enters VLAN configuration mode.
Step 3	`Switch-B(config-slb-vlan-server)# ip address 10.0.101.36 255.255.255.0`	Specifies an IP address and netmask for VLAN 102.
Step 4	`Switch-B(config-slb-vlan-server)# alias 10.0.101.200 255.255.255.0`	Specifies an alias IP address and netmask for VLAN 102[1].
Step 5	`Switch-B(config-slb-vlan-server)# exit`	Returns to multiple module configuration mode.
Step 6	`Switch-B(config-module-csm)# vlan 104 server`	Specifies VLAN 104 as the VLAN that is being configured, identifies it as a server VLAN, and enters VLAN configuration mode.
Step 7	`Switch-B(config-slb-vlan-server)# ip address 10.0.102.36 255.255.255.0`	Specifies an IP address and netmask for VLAN 104.
Step 8	`Switch-B(config-slb-vlan)# alias 10.0.102.200 255.255.255.0`	Specifies an alias IP address and netmask for VLAN 104[1].
Step 9	`Switch-B(config-slb-vlan-server)# exit`	Returns to multiple module configuration mode.
Step 10	`Switch-B(config-module-csm)# vlan 20 server`	Specifies VLAN 20 as the VLAN that is being configured, identifies it as a server VLAN, and enters VLAN configuration mode.
Step 11	`Switch-B(config-slb-vlan-server)# ip address 10.1.0.36 255.255.255.0`	Specifies an IP address and netmask for VLAN 20.

1. This step provides a target for CSM-S A to use in making a load-balancing decision.

Configuring Server Farms on CSM-S B

To configure three server farms on CSM-S B, perform this task:

Note SERVERS-SF specifies that client NAT will be performed using a pool of client NAT addresses that are created earlier in the example using the **natpool** command. You must create the NAT pool before referencing the command.

	Command	Purpose
Step 1	`Switch-B(config)# module csm 6`	Enters multiple module configuration mode and specifies that CSM-S B is installed in slot 6.
Step 2	`Switch-B(config-module-csm)# serverfarm FORWARD-SF`	Creates and names the FORWARD-SF[1] server farm (actually a forwarding policy) and enters serverfarm configuration mode.
Step 3	`Switch-B(config-slb-sfarm)# no nat server`	Disables the NAT of server IP addresses and port numbers[2].
Step 4	`Switch-B(config-slb-sfarm)# predictor forward`	Forwards traffic in accordance with its internal routing tables rather than a load-balancing algorithm.
Step 5	`Switch-B(config-slb-sfarm)# exit`	Returns to multiple module configuration mode.
Step 6	`Switch-B(config-module-csm)# serverfarm TO-OUTSIDE-SF`	Creates and names the GENERIC-SF server farm and enters serverfarm configuration mode[3].
Step 7	`Switch-B(config-slb-sfarm)# no nat server`	Disables NAT of server IP addresses and port numbers[4].
Step 8	`Switch-B(config-slb-sfarm)# real 10.0.101.100`	Identifies the alias IP address of CSM-S A that is locked on the path to Firewall 1 as a real server and enters real server configuration submode.
Step 9	`Switch-B(config-slb-real)# inservice`	Enables the real server (actually an alias IP address).
Step 10	`Switch-B(config-slb-real)# exit`	Returns to the serverfarm configuration mode.
Step 11	`Switch-B(config-slb-sfarm)# real 10.0.102.100`	Identifies the alias IP address of CSM-S B that is located on the path to Firewall 2 as a real server and enters real server configuration submode.
Step 12	`Switch-B(config-slb-real)# inservice`	Enables the real server (actually an alias IP address).
Step 13	`Switch-B(config-slb-real)# exit`	Returns to serverfarm configuration mode.
Step 14	`Switch-B(config-module-csm)# serverfarm SERVERS-SF`	Creates and names the SERVERS-SF[5] server farm and enters serverfarm configuration mode.
Step 15	`Switch-B(config-slb-sfarm)# real 10.1.0.101`	Identifies a server in the intranet as a real server, assigns it an IP address, and enters real server configuration submode.
Step 16	`Switch-B(config-slb-real)# inservice`	Enables the real server.
Step 17	`Switch-B(config-slb-real)# exit`	Returns to serverfarm configuration mode.
Step 18	`Switch-B(config-slb-sfarm)# real 10.1.0.102`	Identifies a server in the intranet as a real server, assigns it an IP address, and enters real server configuration submode.
Step 19	`Switch-B(config-slb-real)# inservice`	Enables the real server.
Step 20	`Switch-B(config-slb-sfarm)# real 10.1.0.103`	Identifies a server in the intranet as a real server, assigns it an IP address, and enters real server configuration submode.
Step 21	`Switch-B(config-slb-real)# inservice`	Enables the real server.

1. FORWARD-SF is actually a route forwarding policy, not an actual server farm, that allows traffic to reach the intranet (through VLAN 20). It does not contain any real servers.

2. This step is required when configuring a server farm that contains a forwarding policy rather than real servers.
3. OUTSIDE-SF contains the two alias IP addresses of CSM-S A as the real servers allowing traffic from the intranet to reach CSM-S A.
4. This step is required when configuring a server farm that contains a forwarding policy rather than real servers.
5. SERVERS-SF contains the IP addresses of the real servers located within the intranet.

Configuring Virtual Servers on CSM-S B

To configure three virtual servers on CSM-S, perform this task:

	Command	Purpose
Step 1	`Switch-B(config)# module csm 6`	Enters multiple module configuration mode and specifies that CSM-S B is installed in slot 6.
Step 2	`Switch-B(config-module-csm)# vserver FORWARD-VS-102`	Specifies FORWARD-VS as the virtual server that is being configured and enters virtual server configuration mode.
Step 3	`Switch-B(config-slb-vserver)# virtual 0.0.0.0 0.0.0.0 any`	Specifies a match for any IP address and any protocol[1].
Step 4	`Switch-B(config-slb-vserver)# vlan 102`	Specifies that the virtual server will only accept traffic arriving on VLAN 102, which is traffic arriving from the secure side of the Firewall 1.
Step 5	`Switch-B(config-slb-vserver)# serverfarm FORWARD-SF`	Specifies the server farm for this virtual server[2].
Step 6	`Switch-B(config-slb-vserver)# inservice`	Enables the virtual server.
Step 7	`Switch-B(config-slb-vserver)# exit`	Returns to multiple module configuration mode.
Step 8	`Switch-B(config-module-csm)# vserver FORWARD-VS-104`	Specifies FORWARD-VS[3] as the virtual server that is being configured and enters virtual server configuration mode.
Step 9	`Switch-B(config-slb-vserver)# virtual 0.0.0.0 0.0.0.0 any`	Specifies a match for any IP address and any protocol[1].
Step 10	`Switch-B(config-slb-vserver)# vlan 104`	Specifies that the virtual server will only accept traffic arriving on VLAN 104, which is traffic arriving from the secure side of the Firewall 2.
Step 11	`Switch-B(config-slb-vserver)# serverfarm FORWARD-SF`	Specifies the server farm for this virtual server[2].
Step 12	`Switch-B(config-slb-vserver)# inservice`	Enables the virtual server.
Step 13	`Switch-B(config-slb-vserver)# exit`	Returns to multiple module configuration mode.
Step 14	`Switch-B(config-module-csm)# vserver INSIDE-VS`	Specifies INSIDE-VS[4] as the virtual server that is being configured and enters virtual server configuration mode.
Step 15	`Switch-B(config-slb-vserver)# virtual 0.0.0.0 0.0.0.0 any`	Specifies a match for any IP address and any protocol[1].
Step 16	`Switch-B(config-slb-vserver)# vlan 20`	Specifies that the virtual server will only accept traffic arriving on VLAN 20, which is traffic arriving from the intranet.

	Command	Purpose
Step 17	`Switch-B(config-slb-vserver)# serverfarm TO-OUTSIDE-SF`	Specifies the server farm for this virtual server (containing the alias IP addresses of CSM-S A as real servers and allowing traffic to flow through Firewalls 1 and 2) and enters real server configuration submode.
Step 18	`Switch-B(config-slb-vserver)# inservice`	Enables the virtual server.
Step 19	`Switch-B(config-slb-vserver)# exit`	Returns to multiple module configuration mode.
Step 20	`Switch-B(config-module-csm)# vserver TELNET-VS`	Specifies TELNET-VS[5] as the virtual server that is being configured and enters virtual server configuration mode. **Note** TELNET-VS does not use a VLAN limit; any source traffic (from firewalls or internal network) will be load balanced through this address.
Step 21	`Switch-B(config-slb-vserver)# virtual 10.1.0.200 255.255.255.0 tcp telnet`	Specifies the IP address, netmask, protocol (TCP), and port (Telnet) for this virtual server[6].
Step 22	`Switch-B(config-slb-vserver)# serverfarm SERVERS-SF`	Specifies the server farm containing real servers for this virtual server.
Step 23	`Switch-B(config-slb-vserver)# inservice`	Enables the virtual server.

1. Client matching is only limited by VLAN restrictions.
2. This server farm is actually a forwarding predictor rather than an actual server farm containing real servers.
3. FORWARD-VS allows traffic from the Internet to reach the intranet through VLAN 20.
4. INSIDE-VS allows traffic from the intranet to reach CSM-S A through Firewall 1 (through VLANs 102 and 101) or Firewall 2 (through VLANs 104 and 103).
5. TELNET-VS allows traffic from the Internet to reach Telnet servers in the internal network.
6. Clients reach the server farm represented by this virtual server through this address.

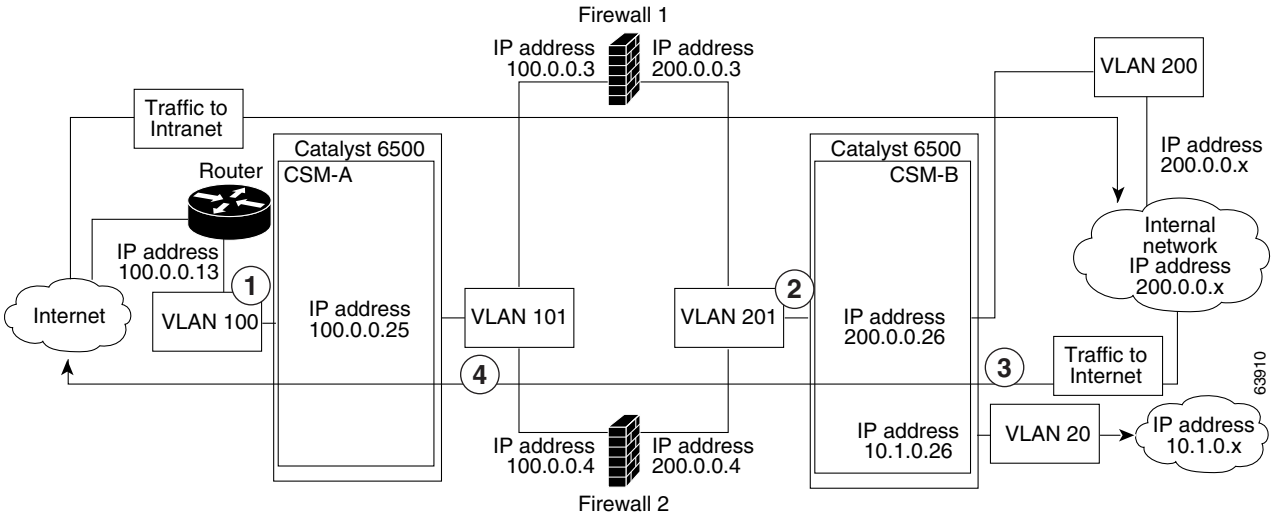
Configuring Regular Firewall Load Balancing

This section describes how to configure firewall load balancing for regular firewalls and provides the following information:

Packet Flow in a Regular Firewall Configuration

In a regular firewall configuration, firewalls connect to two different VLANs and are configured with IP addresses on the VLANs to which they connect. (See Figure 13-7.)

Figure 13-7 Regular Firewall Configuration Example



Item	Traffic Direction	Arrives On	Exits On
1	To intranet	VLAN 100	VLANs 101
2	To intranet	VLANs 201	VLAN 200 and 20
3	To Internet	VLAN 200 and 20	VLANs 201
4	To Internet	VLANs 101	VLAN 100

Figure 13-7 shows two regular firewalls (Firewall 1 and Firewall 2) located between two CSMs (CSM-S A and CSM-S B). Traffic enters and exits the firewalls through shared VLANs (VLAN 101 and VLAN 201). Both regular firewalls have unique addresses on each shared VLAN.

VLANs provide connectivity to the Internet (VLAN 100), the internal network (VLAN 200), and to internal server farms (VLAN 20).

The CSM-S balances traffic among regular firewalls as if they were real servers. Regular firewalls are configured in server farms with IP addresses like real servers. The server farms to which regular firewalls belong are assigned a load-balancing predictor and are associated with virtual servers.

Regular Firewall Configuration Example

The regular firewall configuration example contains two CSM-S modules (CSM-S A and CSM-S B) installed in separate Catalyst 6500 series switches.

Note You can use this example when configuring two CSM-S modules in the same Catalyst 6500 series switch chassis. You can also use this example when configuring a single CSM-S in a single switch chassis, assuming that you specify the slot number of that CSM-S when configuring both CSM-S A and CSM-S B.

Configuring CSM-S A (Regular Firewall Example)

To create the regular configuration example, perform the following configuration tasks for CSM-S A:

- Creating VLANs on Switch A, page 13-18
- Configuring VLANs on CSM-S A, page 13-18
- Configuring Server Farms on CSM-S A, page 13-19
- Configuring Virtual Servers on CSM-S A, page 13-20

Note Although the configuration tasks are the same for both CSM-S A and CSM-S B, the steps, commands, and parameters that you enter are different.

Creating VLANs on Switch A

The example, shown in Figure 13-7, requires that you create two VLANs on Switch A.

Note This example assumes that the CSM-S modules are in separate Catalyst 6500 series switch chassis. If they are in the same chassis, all of the VLANs can be created on the same Catalyst 6500 series switch console.

To configure VLANs on Switch A, perform this task:

	Command	Purpose
Step 1	`Switch-A(config)# vlan`	Enters the VLAN mode[1].
Step 2	`Switch-A(vlan)# vlan 100`	Creates VLAN 100[2].
Step 3	`Switch-A(vlan)# vlan 101`	Creates VLAN 101[3].

1. Do this step on the switch console of the switch that contains CSM-S A.
2. VLAN 100 connects CSM-S A to the Internet.
3. VLAN 101 connects CSM-S A to the insecure side of the firewalls.

Configuring VLANs on CSM-S A

To configure the two VLANs, perform this task:

	Command	Purpose
Step 1	`Switch-A(config)# module csm 5`	Enters multiple module configuration mode and specifies that CSM-S A is installed in slot 5.
Step 2	`Switch-A(config-module-csm)# vlan 100 client`	Specifies VLAN 100 as the VLAN that is being configured, identifies it as a client VLAN, and enters VLAN configuration mode.
Step 3	`Switch-A(config-slb-vlan-client)# ip address 100.0.0.25 255.255.255.0`	Specifies an IP address and netmask for VLAN 100.
Step 4	`Switch-A(config-slb-vlan-client)# gateway 100.0.0.13`	Configures a gateway IP address for the router on the Internet side of CSM-S A.
Step 5	`Switch-A(config-slb-vlan-client)# exit`	Returns to multiple module configuration mode.

	Command	Purpose
Step 6	`Switch-A(config-module-csm)# vlan 101 server`	Specifies VLAN 101 as the VLAN that is being configured, identifies it as a server VLAN, and enters VLAN configuration mode.
Step 7	`Switch-A(config-slb-vlan-server)# ip address 100.0.0.25 255.255.255.0`	Specifies an IP address and netmask for VLAN 101.
Step 8	`Switch-A(config-slb-vlan-server)# alias 100.0.0.20 255.255.255.0`	Specifies an alias IP address and netmask for VLAN 101[1].

1. This step provides a target for CSM-S B to use in making a load-balancing decision.

Configuring Server Farms on CSM-S A

Note Firewall 1 and Firewall 2 secure-side IP addresses are configured as real servers in the SEC-SF server farm associated with CSM-S B.

To configure two server farms on CSM-S A, perform this task:

	Command	Purpose
Step 1	`Switch-A(config)# module csm 5`	Enters multiple module configuration mode and specifies that CSM-S A is installed in slot 5.
Step 2	`Switch-A(config-module-csm)# serverfarm FORWARD-SF`	Creates and names the FORWARD-SF[1] server farm (actually a forwarding policy) and enters serverfarm configuration mode.
Step 3	`Switch-A(config-slb-sfarm)# no nat server`	Disables the NAT of server IP addresses and port numbers[2].
Step 4	`Switch-A(config-slb-sfarm)# predictor forward`	Forwards traffic by adhering to its internal routing tables rather than a load-balancing algorithm.
Step 5	`Switch-A(config-slb-sfarm)# exit`	Returns to multiple module configuration mode.
Step 6	`Switch-A(config-module-csm)# serverfarm INSEC-SF`	Creates and names the INSEC-SF[3] server farm (which will contain firewalls as real servers) and enters serverfarm configuration mode.
Step 7	`Switch-A(config-slb-sfarm)# no nat server`	Disables the NAT of the server IP address and port number[4].
Step 8	`Switch-A(config-slb-sfarm)# predictor hash address source 255.255.255.255`	Selects a server using a hash value based on the source IP address[5].
Step 9	`Switch-A(config-slb-sfarm)# real 100.0.0.3`	Identifies Firewall 1 as a real server, assigns an IP address to its insecure side, and enters real server configuration submode.
Step 10	`Switch-A(config-slb-real)# inservice`	Enables the firewall.
Step 11	`Switch-A(config-slb-real)# exit`	Returns to serverfarm configuration mode.
Step 12	`Switch-A(config-slb-sfarm)# real 100.0.0.4`	Identifies Firewall 2 as a real server, assigns an IP address to its insecure side, and enters real server configuration submode.
Step 13	`Switch-A(config-slb-real)# inservice`	Enables the firewall.

1. FORWARD-SF is actually a route forwarding policy, not an actual server farm, that allows traffic to reach the Internet (through VLAN 100); it does not contain any real servers.
2. This is a required step when configuring a server farm that contains a forwarding policy rather than real servers.
3. INSEC-SF contains (Firewall 1 and Firewall 2); their insecure-side IP addresses are configured as real servers in this server farm.
4. This is a required step when configuring a server farm that contains firewalls.
5. We recommend this step when configuring insecure-side firewall interfaces in a server farm.

Configuring Virtual Servers on CSM-S A

To configure two virtual servers on CSM-S A, perform this task:

	Command	Purpose
Step 1	`Switch-A(config)# module csm 5`	Enters multiple module configuration mode and specifies that the CSM-S A is installed in slot 5.
Step 2	`Switch-A(config-module-csm)# vserver FORWARD-VS`	Specifies FORWARD-VS[1] as the virtual server that is being configured and enters virtual server configuration mode.
Step 3	`Switch-A(config-slb-vserver)# virtual 0.0.0.0 0.0.0.0 any`	Specifies a match for any IP address and any protocol[2].
Step 4	`Switch-A(config-slb-vserver))# vlan 101`	Specifies that the virtual server will only accept traffic arriving on VLAN 101, which is traffic arriving from the insecure side of the firewalls.
Step 5	`Switch-A(config-slb-vserver)# serverfarm FORWARD-SF`	Specifies the server farm for this virtual server[3].
Step 6	`Switch-A(config-slb-vserver)# inservice`	Enables the virtual server.
Step 7	`Switch-A(config-slb-vserver)# exit`	Returns to multiple module configuration mode.
Step 8	`Switch-A(config-module-csm)# vserver INSEC-VS`	Specifies INSEC-VS[4] as the virtual server that is being configured and enters virtual server configuration mode.
Step 9	`Switch-A(config-slb-vserver)# virtual 200.0.0.0 255.255.255.0 any`	Specifies the IP address, netmask, and protocol (any) for this virtual server[5].
Step 10	`Switch-A(config-slb-vserver))# vlan 100`	Specifies that the virtual server will only accept traffic arriving on VLAN 100, which is traffic arriving from the Internet.
Step 11	`Switch-A(config-slb-vserver)# serverfarm INSEC-SF`	Specifies the server farm for this virtual server[6].
Step 12	`Switch-A(config-slb-vserver)# inservice`	Enables the virtual server.

1. FORWARD-VS allows Internet traffic to reach the insecure side of the firewalls (through VLAN 101).
2. Client matching is only limited by VLAN restrictions. (See Step 4.)
3. This server farm is actually a forwarding predictor rather than an actual server farm containing real servers.
4. INSEC-VS allows traffic from the Internet to reach CSM-S A (through VLAN 101).
5. Clients reach the server farm represented by this virtual server through this address.
6. The server farm contains firewalls rather than real servers.

Configuring CSM-S B (Regular Firewall Example)

To create the regular configuration example, perform the following configuration tasks for CSM-S B:

- Creating VLANs on Switch B, page 13-21
- Configuring VLANs on CSM-S B, page 13-21
- Configuring Server Farms on CSM-S B, page 13-22
- Configuring Virtual Servers on CSM-S B, page 13-23

Note Although the configuration tasks are the same for both CSM-S A and CSM-S B, the steps, commands, and parameters that you enter are different.

Creating VLANs on Switch B

Note This example assumes that the CSM-S modules are in separate Catalyst 6500 series switch chassis. If they are in the same chassis, all of the VLANs can be created on the same Catalyst 6500 series switch console.

To create three VLANs on Switch B, perform this task:

	Command	Purpose
Step 1	`Switch-B(config)# vlan`	Enters the VLAN mode[1].
Step 2	`Switch-B(vlan)# vlan 201`	Creates VLAN 201[2].
Step 3	`Switch-B(vlan)# vlan 200`	Creates VLAN 200[3].
Step 4	`Switch-B(vlan)# vlan 20`	Creates VLAN 20[4].

1. Do this step on the switch console of the switch that contains CSM-S B.
2. VLAN 201 provides the connection to the secure side of the firewalls.
3. VLAN 20 provides the connection to the internal server farms.
4. VLAN 200 provides the connection to the internal network.

Configuring VLANs on CSM-S B

To configure the three VLANs on CSM-S B, perform this task:

	Command	Purpose
Step 1	`Switch-B(config)# module csm 6`	Enters multiple module configuration mode and specifies that CSM-S B is installed in slot 6.
Step 2	`Switch-B(config-module-csm)# vlan 201 server`	Specifies VLAN 201 as the VLAN that is being configured, identifies it as a server VLAN, and enters VLAN configuration mode.
Step 3	`Switch-B(config-slb-vlan-server)# ip address 200.0.0.26 255.255.255.0`	Specifies an IP address and netmask for VLAN 201.
Step 4	`Switch-B(config-slb-vlan-server)# alias 200.0.0.20 255.255.255.0`	Specifies an alias IP address and netmask for VLAN 201[1].

	Command	Purpose
Step 5	`Switch-B(config-slb-vlan-server)# exit`	Returns to VLAN configuration mode.
Step 6	`Switch-B(config-module-csm)# vlan 20 server`	Specifies VLAN 20 as the VLAN that is being configured, identifies it as a server VLAN, and enters VLAN configuration mode.
Step 7	`Switch-B(config-slb-vlan-server)# ip address 10.1.0.26 255.255.255.0`	Specifies an IP address and netmask for VLAN 20.
Step 8	`Switch-B(config-slb-vlan-server)# exit`	Returns to VLAN configuration mode.
Step 9	`Switch-B(config-module-csm)# vlan 200 client`	Specifies VLAN 200 as the VLAN that is being configured, identifies it as a client VLAN, and enters VLAN configuration mode.
Step 10	`Switch-B(config-slb-vlan)# ip address 200.0.0.26 255.255.255.0`	Specifies an IP address and netmask for VLAN 200.

1. This step provides a target for CSM-S A to use in making a load-balancing decision.

Configuring Server Farms on CSM-S B

Note Firewall 1 and Firewall 2 secure-side IP addresses are configured as real servers in the INSEC-SF server farm associated with CSM-S A.

To configure two server farms on CSM-S B, perform this task:

	Command	Purpose
Step 1	`Switch-B(config)# module csm 6`	Enters multiple module configuration mode and specifies that CSM-S B is installed in slot 6.
Step 2	`Switch-B(config-module-csm)# serverfarm GENERIC-SF`	Creates and names the GENERIC-SF[1] server farm and enters serverfarm configuration mode.
Step 3	`Switch-B(config-slb-sfarm)# real 10.1.0.101`	Identifies a server in the internal server farm as a real server, assigns it an IP address, and enters real server configuration submode.
Step 4	`Switch-B(config-slb-real)# inservice`	Enables the real server.
Step 5	`Switch-B(config-slb-real)# exit`	Returns to serverfarm configuration mode.
Step 6	`Switch-B(config-slb-sfarm)# real 10.1.0.102`	Identifies a server in the internal server farm as a real server, assigns it an IP address, and enters real server configuration submode.
Step 7	`Switch-B(config-slb-real)# inservice`	Enables the real server.
Step 8	`Switch-B(config-slb-real)# exit`	Returns to serverfarm configuration mode.
Step 9	`Switch-B(config-slb-sfarm)# exit`	Returns to multiple module configuration mode.
Step 10	`Switch-B(config-module-csm)# serverfarm SEC-SF`	Creates and names the SEC-SF[2] server farm and enters serverfarm configuration mode.
Step 11	`Switch-B(config-slb-sfarm)# no nat server`	Disables the NAT of server IP address and port number[3].

	Command	Purpose
Step 12	`Switch-B(config-slb-sfarm)# predictor hash address destination 255.255.255.255`	Selects a server using a hash value based on the destination IP address[4].
Step 13	`Switch-B(config-slb-sfarm)# real 200.0.0.3`	Identifies Firewall 1 as a real server, assigns an IP address to its insecure side, and enters real server configuration submode.
Step 14	`Switch-B(config-slb-real)# inservice`	Enables the firewall.
Step 15	`Switch-B(config-slb-real)# exit`	Returns to serverfarm configuration mode.
Step 16	`Switch-B(config-slb-sfarm)# real 200.0.0.4`	Identifies Firewall 2 as a real server, assigns an IP address to its insecure side, and enters real server configuration submode.
Step 17	`Switch-B(config-slb-real)# inservice`	Enables the firewall.

1. GENERIC-SF contains the real servers in the internal server farm.
2. SEC-SF contains (Firewall 1 and Firewall 2); their secure-side IP addresses are configured as real servers in this server farm.
3. This is a required step when configuring a server farm that contains firewalls.
4. We recommend this step when configuring secure-side firewall interfaces in a server farm.

Configuring Virtual Servers on CSM-S B

To configure three virtual servers on CSM-S B, perform this task:

	Command	Purpose
Step 1	`Switch-B(config)# module csm 6`	Enters multiple module configuration mode and specifies that CSM-S B is installed in slot 6.
Step 2	`Switch-B(config-module-csm)# vserver GENERIC-VS`	Specifies GENERIC-VS[1] as the virtual server that is being configured and enters virtual server configuration mode.
Step 3	`Switch-B(config-slb-vserver)# virtual 200.0.0.127 tcp 0`	Specifies the IP address, protocol (TCP), and port (0=any) for this virtual server[2].
Step 4	`Switch-B(config-slb-vserver))# vlan 201`	Specifies that the virtual server will only accept traffic arriving on VLAN 201, which is traffic arriving from the secure side of the firewalls.
Step 5	`Switch-B(config-slb-vserver)# serverfarm GENERIC-SF`	Specifies the server farm for this virtual server[3].
Step 6	`Switch-B(config-slb-vserver)# inservice`	Enables the virtual server.
Step 7	`Switch-B(config-slb-vserver)# exit`	Returns to multiple module configuration mode.
Step 8	`Switch-B(config-module-csm)# vserver SEC-20-VS`	Specifies SEC-20-VS[4] as the virtual server that is being configured and enters virtual server configuration mode.
Step 9	`Switch-B(config-slb-vserver)# virtual 200.0.0.0 255.255.255.0 any`	Specifies the IP address, netmask, and protocol (any) for this virtual server[2].
Step 10	`Switch-B(config-slb-vserver))# vlan 20`	Specifies that the virtual server will only accept traffic arriving on VLAN 20, which is traffic arriving from the internal server farms.

	Command	Purpose
Step 11	`Switch-B(config-slb-vserver)# serverfarm SEC-SF`	Specifies the server farm for this virtual server[5].
Step 12	`Switch-B(config-slb-vserver)# inservice`	Enables the virtual server.
Step 13	`Switch-B(config-slb-vserver)# exit`	Returns to multiple module configuration mode.
Step 14	`Switch-B(config-module-csm)# vserver SEC-200-VS`	Specifies SEC-20-VS[6] as the virtual server that is being configured and enters virtual server configuration mode.
Step 15	`Switch-B(config-slb-vserver)# virtual 200.0.0.0 255.255.255.0 any`	Specifies the IP address, netmask, and protocol (any) for this virtual server[2].
Step 16	`Switch-B(config-slb-vserver))# vlan 200`	Specifies that the virtual server will only accept traffic arriving on VLAN 200, which is traffic arriving from the internal network.
Step 17	`Switch-B(config-slb-vserver)# serverfarm SEC-SF`	Specifies the server farm for this virtual server[5].
Step 18	`Switch-B(config-slb-vserver)# inservice`	Enables the virtual server.

1. GENERIC-VS allows traffic from the internal server farms and the internal network that is destined for the Internet to reach the secure side of the firewalls (through VLAN 101).
2. Clients reach the server farm represented by this virtual server through this address.
3. The server farm exists in the internal server farms network.
4. SEC-20-VS allows traffic from the Internet to reach the internal server farms (through VLAN 20).
5. The server farm contains firewalls rather than real servers.
6. SEC-200-VS allows traffic from the Internet to reach the internal network (through VLAN 20).

Configuring Reverse-Sticky for Firewalls

The reverse-sticky feature creates a database of load-balancing decisions based on the client's IP address. This feature overrides the load-balancing decision when a reverse-sticky entry is available in the database. If there is no reverse-sticky entry in the database, a load-balancing decision takes place, and the result is stored for future matching.

Understanding Reverse-Sticky for Firewalls

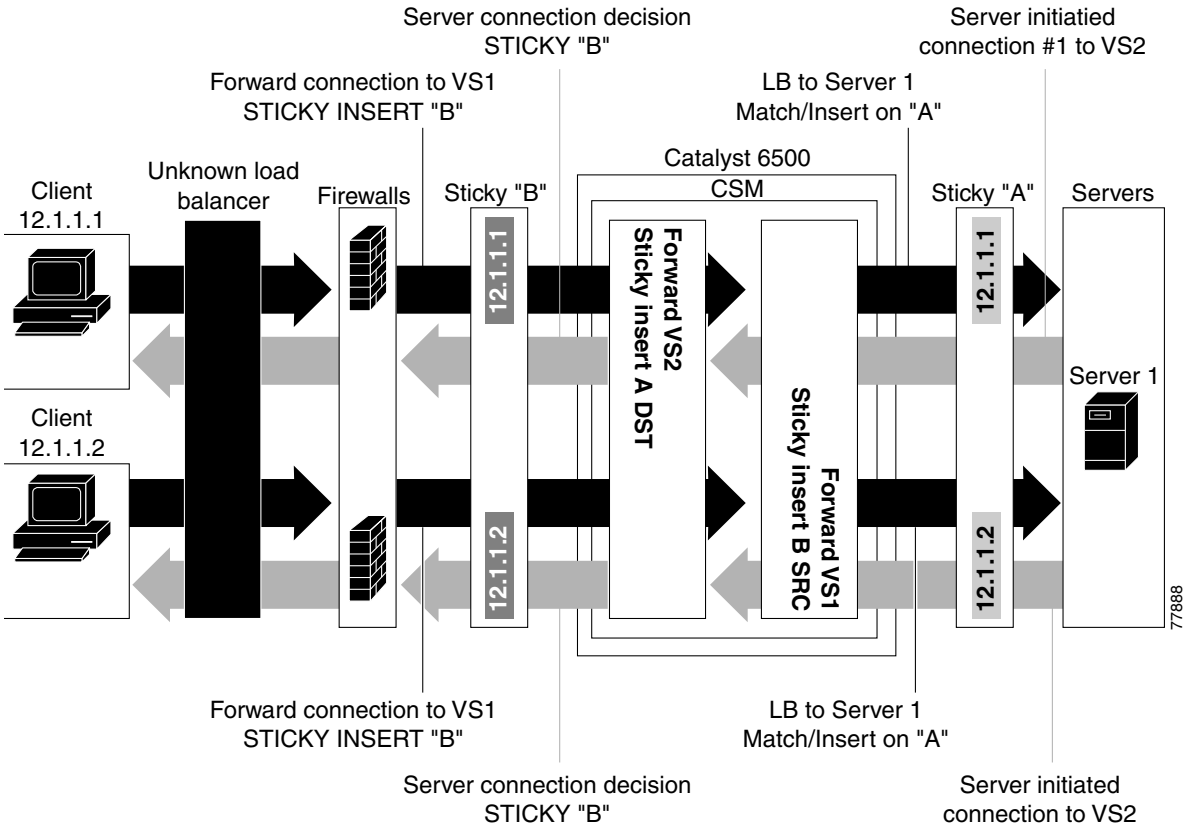
Reverse-sticky provides a way of inserting entries into a sticky database as if the connection came from the other direction. A virtual server with reverse-sticky places an entry into the specified database containing the inbound real server.

Note The inbound real server must be a real server within a server farm.

This entry is matched by a sticky command on a different virtual server. The other virtual server sends traffic to the client, based on this pregenerated entry.

The CSM-S stores reverse-sticky information as links from a source IP key to a real server. When the load balancer gets a new session on a virtual server with an assigned sticky database, it first checks the database for an existing entry. If a matching entry is found, the session is connected to the specified real server. Otherwise, a new entry is created linking the sticky key with the appropriate real server. Figure 13-8 shows how the reverse-sticky feature is used for firewalls.

Figure 13-8 Reverse-Sticky for Firewalls



As shown in Figure 13-8, the reverse-sticky process is as follows:

- A client connects to the CSM-S virtual server, VS1, through a load-balanced firewall. This load-balancing decision is made without interaction with the CSM-S.
- Server 1 creates a connection back to the original client. This connection matches virtual server VS2. VS2 uses the sticky information inserted by the original VS1 reverse-sticky. The connection now is forced to the same Firewall 1.
- A second client, coming in through a different firewall, connects to the same VS1. Reverse-sticky creates a new entry into database B for the second client, pointing to Firewall 2. VS1 also performs a normal sticky to Server 1.
- Server 1 creates a connection back to Client 2. The connection matches the connection in VS2. VS2 uses the sticky information inserted by the original VS1 reverse-sticky. This connection is used for the connection to Firewall 2.
- If the server had originated the first connection, the link back to the server would have been inserted by VS2, and a normal load-balancing decision would have generated a connection to one of the firewalls.

Note This configuration supports forward direction connections (client to server) using any balancing metric. However, the balancing metric to the firewalls from VS2 must match that of the unknown load balancer, or the unknown load balancer must stick new buddy connections in a similar manner if client responses to server-initiated traffic are to be sent to the correct firewall.

Configuring Reverse-Sticky for Firewalls

To configure IP reverse-sticky for firewall load balancing, perform this task:

	Command	Purpose
Step 1	`SLB-Switch(config)# module csm slot`	Associates load-balancing commands to a specific CSM-S module and enters the CSM-S module configuration submode for the specified slot.
Step 2	`SLB-Switch(config-module-csm)# vserver virtserver-name`	Identifies a virtual server and enters the virtual server configuration submode.
Step 3	`SLB-Switch(config-slb-vserver)# sticky duration [group group-id] [netmask ip-netmask] [source | destination | both]`	Defines the portion of the IP information (source, destination, or both) that is used for the sticky entry key.
Step 4	`SLB-Switch(config-slb-vserver)# reverse-sticky group-id`	Ensures that the CSM-S maintains connections in the opposite direction back to the original source.
Step 5	`SLB-Switch# show module csm slot sticky`	Displays the sticky database.

Configuring Stateful Firewall Connection Remapping

To configure the firewall reassignment feature, you must have an MSFC image from Cisco IOS Release 12.1(19)E.

To configure firewall reassignment, perform these steps:

Step 1 In the serverfarm submode for firewalls, configure the action:

```
Cat6k-2(config)# serverfarm FW-FARM
failaction reassign
```

Step 2 Assign a backup real server for each firewall if it failed (probe or ARP):

```
Cat6k-2(config-slb-sfarm)# serverfarm FW-FARM
Cat6k-2(config-slb-sfarm)# real 1.1.1.1
Cat6k(config-slb-module-real)# backup real 2.2.2.2
Cat6k(config-slb-module-real)# inservice
Cat6k-2(config-slb-sfarm)# real 2.2.2.2
Cat6k(config-slb-module-real)# backup real 3.3.3.3
Cat6k(config-slb-module-real)# inservice
Cat6k-2(config-slb-sfarm)# real 3.3.3.3
Cat6k(config-slb-module-real)# backup real 1.1.1.1
Cat6k(config-slb-module-real)# inservice
```

Step 3 Configure the ICMP probe (through the firewall) for this server farm.

Step 4 Configure the ICMP probes for the CSM-S modules outside and inside the firewall.

Make sure that the backup real server is configured in the same order in both CSM-S modules.

The inservice standby option assigned to a real server specifies that this server only receives connections if they are destined or load-balanced to the failed primary server. If you configure the real server designated as real 2.2.2.2 with inservice standby, then all connections would go to either of the real servers designated as real 1.1.1.1 or real 3.3.3.3. When real server real 1.1.1.1 failed, the real server designated as real 2.2.2.2 will be active in place of real server real 1.1.1.1.

CSM-S Configuration Examples

This chapter describes how to configure firewall load balancing and contains these sections:

Each example in this appendix includes only the relevant portions of the configuration. In some cases, some portions of the Layer 2 and Layer 3 Catalyst switch configuration are included. Lines with comments start with # and can be pasted in the configuration once you are in configuration mode after entering the **configuration terminal** command.

Make sure that you create all the VLANs used in the CSM-S configuration on the switch using the **vlan** command.

Configuring the Router Mode with the MSFC on the Client Side

This example provides configuration parameters for setting up the router mode:

```
module ContentSwitchingModule 5
 vlan 220 server
  ip address 10.20.220.2 255.255.255.0
  alias 10.20.220.1 255.255.255.0

# The servers' default gateway is the alias IP address
# Alias IP addresses are needed any time that you are
# configuring a redundant system.
```

```
# However, it is a good practice to always use a
# alias IP address so that a standby CSM-S can easily
# be added without changes to the IP addressing scheme

!
 vlan 221 client
  ip address 10.20.221.5 255.255.255.0
  gateway 10.20.221.1

# The CSM-S default gateway in this config is the
# MSFC IP address on that VLAN

!
 serverfarm WEBFARM
  nat server
  no nat client
  real 10.20.220.10
   inservice
  real 10.20.220.20
   inservice
  real 10.20.220.30
   no inservice
!
 vserver WEB
  virtual 10.20.221.100 tcp www
  serverfarm WEBFARM
  persistent rebalance
  inservice

# "persistence rebalance" is effective ONLY when performing
# L7 load balancing (parsing of URLs, cookies, header, ...)
# and only for HTTP 1.1 connections.
# It tells the CSM-S to parse and eventually make a new
# load balancing decision for each GET within the same
# TCP connection.

interface FastEthernet2/2
 no ip address
 switchport
 switchport access vlan 220

# The above is the port that connects to the real servers

interface FastEthernet2/24
 ip address 10.20.1.1 255.255.255.0

# The above is the interface that connects to the client side network

interface Vlan221
 ip address 10.20.221.1 255.255.255.0

# The above is the MSFC interface for the internal VLAN used
# for MSFC-CSM-S communication
```

This example shows the output of the **show** commands:

```
Cat6k-2# show module csm 5 arp

Internet Address  Physical Interface  VLAN      Type       Status
------------------------------------------------------------------
 10.20.220.1      00-02-FC-E1-68-EB   220       -ALIAS-    local
 10.20.220.2      00-02-FC-E1-68-EC   220       --SLB--    local
 10.20.220.10     00-D0-B7-A0-81-D8   220       REAL       up(0 misses)
 10.20.221.1      00-02-FC-CB-70-0A   221       GATEWAY    up(0 misses)
```

```
 10.20.221.5      00-02-FC-E1-68-EC   221       --SLB--    local
 10.20.220.20     00-D0-B7-A0-81-D8   220       REAL       up(0 misses)
 10.20.220.30     00-D0-B7-A0-81-D8   220       REAL       up(0 misses)
 10.20.221.100    00-02-FC-E1-68-EB   0         VSERVER    local

Cat6k-2# show module csm 5 vlan detail
vlan   IP address       IP mask          type
-------------------------------------------------
220    10.20.220.2      255.255.255.0    SERVER
  ALIASES
  IP address       IP mask
  -------------------------------
  10.20.220.1      255.255.255.0
221    10.20.221.5      255.255.255.0    CLIENT
  GATEWAYS
  10.20.221.1
Cat6k-2#
Cat6k-2# show module csm 5 real

real                  server farm      weight  state          conns/hits
-------------------------------------------------------------------------
10.20.220.10          WEBFARM          8       OPERATIONAL    0
10.20.220.20          WEBFARM          8       OPERATIONAL    0
10.20.220.30          WEBFARM          8       OUTOFSERVICE   0
Cat6k-2#
Cat6k-2# show module csm 5 real detail
10.20.220.10, WEBFARM, state = OPERATIONAL
  conns = 0, maxconns = 4294967295, minconns = 0
  weight = 8, weight(admin) = 8, metric = 0, remainder = 0
  total conns established = 5, total conn failures = 0
10.20.220.20, WEBFARM, state = OPERATIONAL
  conns = 0, maxconns = 4294967295, minconns = 0
  weight = 8, weight(admin) = 8, metric = 0, remainder = 0
  total conns established = 5, total conn failures = 0
10.20.220.30, WEBFARM, state = OUTOFSERVICE
  conns = 0, maxconns = 4294967295, minconns = 0
  weight = 8, weight(admin) = 8, metric = 0, remainder = 0
  total conns established = 0, total conn failures = 0

Cat6k-2#
Cat6k-2# show module csm 5 vserver detail
WEB, type = SLB, state = OPERATIONAL, v_index = 17
  virtual = 10.20.221.100/32:80 bidir, TCP, service = NONE, advertise = FALSE
  idle = 3600, replicate csrp = none, vlan = ALL, pending = 30, layer 4
  max parse len = 2000, persist rebalance = TRUE
  ssl sticky offset = 0, length = 32
  conns = 0, total conns = 10
  Default policy:
    server farm = WEBFARM, backup = <not assigned>
    sticky: timer = 0, subnet = 0.0.0.0, group id = 0
  Policy          Tot matches  Client pkts  Server pkts
  ------------------------------------------------------
  (default)       10           50           50

Cat6k-2#
Cat6k-2# show module csm 5 stats
Connections Created:        28
Connections Destroyed:      28
Connections Current:        0
Connections Timed-Out:      0
Connections Failed:         0
Server initiated Connections:
      Created: 0, Current: 0, Failed: 0
L4 Load-Balanced Decisions: 27
```

```
L4 Rejected Connections:    1
L7 Load-Balanced Decisions: 0
L7 Rejected Connections:
      Total: 0, Parser: 0,
      Reached max parse len: 0, Cookie out of mem: 0,
      Cfg version mismatch: 0, Bad SSL2 format: 0
L4/L7 Rejected Connections:
      No policy: 1, No policy match 0,
      No real: 0, ACL denied 0,
      Server initiated: 0
Checksum Failures:  IP: 0, TCP: 0
Redirect Connections: 0,  Redirect Dropped: 0
FTP Connections:            0
MAC Frames:
      Tx: Unicast: 345, Multicast: 5, Broadcast: 25844,
          Underflow Errors: 0
      Rx: Unicast: 1841, Multicast: 448118, Broadcast: 17,
          Overflow Errors: 0, CRC Errors: 0
```

Configuring the Bridged Mode with the MSFC on the Client Side

This example provides configuration parameters for configuring bridged mode:

```
module ContentSwitchingModule 5
 vlan 221 client
  ip address 10.20.220.2 255.255.255.0
  gateway 10.20.220.1
!
 vlan 220 server
  ip address 10.20.220.2 255.255.255.0

# Two VLANs with the same IP address are bridged together.

!
 serverfarm WEBFARM
  nat server
  no nat client
  real 10.20.220.10
   inservice
  real 10.20.220.20
   inservice
  real 10.20.220.30
   no inservice
!
 vserver WEB
  virtual 10.20.220.100 tcp www
  serverfarm WEBFARM
  persistent rebalance
  inservice

interface FastEthernet2/2
 no ip address
 switchport
 switchport access vlan 220

# The above is the port that connects to the real servers

interface FastEthernet2/24
 ip address 10.20.1.1 255.255.255.0
```

```
# The above is the MSFC interface that connects to the client side network

interface Vlan221
 ip address 10.20.220.1 255.255.255.0

# The above is the MSFC interface for the internal VLAN used
# for MSFC-CSM-S communication.
# The servers use this IP address as their default gateway
# since the CSM-S is bridging between the client and server VLANs
```

This example shows the output of the **show** commands:

```
Cat6k-2# show module csm 5 arp

Internet Address  Physical Interface  VLAN      Type       Status
------------------------------------------------------------------
 10.20.220.1      00-02-FC-CB-70-0A   221       GATEWAY    up(0 misses)
 10.20.220.2      00-02-FC-E1-68-EC   221/220   --SLB--    local
 10.20.220.10     00-D0-B7-A0-81-D8   220       REAL       up(0 misses)
 10.20.220.20     00-D0-B7-A0-81-D8   220       REAL       up(0 misses)
 10.20.220.30     00-D0-B7-A0-81-D8   220       REAL       up(0 misses)
 10.20.220.100    00-02-FC-E1-68-EB   0         VSERVER    local
```

Configuring the Probes

This example provides configuration parameters for configuring probes:

```
module ContentSwitchingModule 5
 vlan 220 server
  ip address 10.20.220.2 255.255.255.0
  alias 10.20.220.1 255.255.255.0
!
 vlan 221 client
  ip address 10.20.221.5 255.255.255.0
  gateway 10.20.221.1
!
 probe PING icmp
  interval 5
  failed 10
  receive 4

# Interval between the probes is 5 seconds for healthy servers
# while it is 10 seconds for failed servers.
# The servers need to reply within 4 seconds.

!
 probe TCP tcp
  interval 5
  failed 10
  open 4

# The servers need to open the TCP connection within 4 seconds.

!
 probe HTTP http
  request method head url /probe/http_probe.html
  expect status 200 299
  interval 20
  port 80

# The port for the probe is inherited from the vservers.
# The port is necessary in this case, since the same farm
```

```
# is serving a vserver on port 80 and one on port 23.
# If the "port 80" parameter is removed, the HTTP probe
# will be sent out on both ports 80 and 23, thus failing
# on port 23 which does not serve HTTP requests.

 probe PING-SERVER-30 icmp
  interval 5
  failed 10
!
 serverfarm WEBFARM
  nat server
  no nat client
  real 10.20.220.10
   inservice
  real 10.20.220.20
   inservice
  real 10.20.220.30
   health probe PING-SERVER-30
   inservice
  probe PING
  probe TCP
  probe HTTP
!
 vserver TELNET
  virtual 10.20.221.100 tcp telnet
  serverfarm WEBFARM
  persistent rebalance
  inservice
!
 vserver WEB
  virtual 10.20.221.100 tcp www
  serverfarm WEBFARM
  persistent rebalance
  inservice
!
```

This example shows the output of the **show** commands:

```
Cat6k-2# show module csm 5 probe

probe           type    port  interval retries failed  open   receive
------------------------------------------------------------------------
PING            icmp          5        3       10             4
TCP             tcp           5        3       10      4
HTTP            http    80    20       3       300     10     10
PING-SERVER-30  icmp          5        3       10             10

Cat6k-2# show module csm 5 probe detail
probe           type    port  interval retries failed  open   receive
------------------------------------------------------------------------
PING            icmp          5        3       10             4
 real                  vserver         serverfarm      policy          status
 ---------------------------------------------------------------------------
 10.20.220.30:80       WEB             WEBFARM         (default)       OPERABLE
 10.20.220.20:80       WEB             WEBFARM         (default)       OPERABLE
 10.20.220.10:80       WEB             WEBFARM         (default)       OPERABLE
 10.20.220.30:23       TELNET          WEBFARM         (default)       OPERABLE
 10.20.220.20:23       TELNET          WEBFARM         (default)       OPERABLE
 10.20.220.10:23       TELNET          WEBFARM         (default)       OPERABLE
TCP             tcp           5        3       10      4
 real                  vserver         serverfarm      policy          status
 ---------------------------------------------------------------------------
 10.20.220.30:80       WEB             WEBFARM         (default)       OPERABLE
 10.20.220.20:80       WEB             WEBFARM         (default)       OPERABLE
```

```
 10.20.220.10:80       WEB             WEBFARM         (default)       OPERABLE
 10.20.220.30:23       TELNET          WEBFARM         (default)       OPERABLE
 10.20.220.20:23       TELNET          WEBFARM         (default)       OPERABLE
 10.20.220.10:23       TELNET          WEBFARM         (default)       OPERABLE
HTTP            http    80    20       3       300     10     10
 Probe Request:  HEAD       /probe/http_probe.html
 Expected Status Codes:
  200 to 299
 real                  vserver         serverfarm      policy          status
 ---------------------------------------------------------------------------
 10.20.220.30:80       WEB             WEBFARM         (default)       OPERABLE
 10.20.220.20:80       WEB             WEBFARM         (default)       FAILED
 10.20.220.10:80       WEB             WEBFARM         (default)       OPERABLE
 10.20.220.30:80       TELNET          WEBFARM         (default)       OPERABLE
 10.20.220.20:80       TELNET          WEBFARM         (default)       FAILED
 10.20.220.10:80       TELNET          WEBFARM         (default)       OPERABLE
PING-SERVER-30  icmp          5        3       10             10
 real                  vserver         serverfarm      policy          status
 ---------------------------------------------------------------------------
 10.20.220.30:80       WEB             WEBFARM         (default)       OPERABLE
 10.20.220.30:23       TELNET          WEBFARM         (default)       OPERABLE

Cat6k-2# show module csm 5 real

real                  server farm      weight  state          conns/hits
-------------------------------------------------------------------------
10.20.220.10          WEBFARM          8       OPERATIONAL    0
10.20.220.20          WEBFARM          8       PROBE_FAILED   0
10.20.220.30          WEBFARM          8       OPERATIONAL    0
```

Configuring the Source NAT for Server-Originated Connections to the VIP

This example shows a situation where the servers have open connections to the same VIP address that clients access. Because the servers are balanced back to themselves, the source NAT is required. To set the source NAT, use the **vlan** parameter in the virtual server configuration to distinguish the VLAN where the connection is originated. A different server farm is then used to handle server-originated connections. Source NAT is configured for that server farm. No source NAT is used for client-originated connections so that the servers can log the real client IPs.

Note You should use a similar configuration when the server-to-server load-balanced connections need to be supported with the source and destination servers located in the same VLAN.

```
module ContentSwitchingModule 5
 vlan 220 server
  ip address 10.20.220.2 255.255.255.0
  alias 10.20.220.1 255.255.255.0
!
 vlan 221 client
  ip address 10.20.221.5 255.255.255.0
  gateway 10.20.221.1
!
 natpool POOL-1 10.20.220.99 10.20.220.99 netmask 255.255.255.0
!
 serverfarm FARM
```

```
  nat server
  no nat client
  real 10.20.220.10
   inservice
  real 10.20.220.20
   inservice
  real 10.20.220.30
   inservice
!
 serverfarm FARM2
  nat server
  nat client POOL-1
  real 10.20.220.10
   inservice
  real 10.20.220.20
   inservice
  real 10.20.220.30
   inservice
!
 vserver FROM-CLIENTS
  virtual 10.20.221.100 tcp telnet
  vlan 221
  serverfarm FARM
  persistent rebalance
  inservice
!
 vserver FROM-SERVERS
  virtual 10.20.221.100 tcp telnet
  vlan 220
  serverfarm FARM2
  persistent rebalance
  inservice
```

This example shows the output of the **show** commands:

```
Cat6k-2# show module csm 5 vser
vserver         type  prot virtual                 vlan state        conns
-------------------------------------------------------------------------------
FROM-CLIENTS    SLB   TCP  10.20.221.100/32:23     221  OPERATIONAL  1
FROM-SERVERS    SLB   TCP  10.20.221.100/32:23     220  OPERATIONAL  1

Cat6k-2# show module csm 5 conn detail

    prot vlan source               destination          state
-------------------------------------------------------------------------
In  TCP  220  10.20.220.10:32858    10.20.221.100:23     ESTAB
Out TCP  220  10.20.220.20:23       10.20.220.99:8193    ESTAB
    vs = FROM-SERVERS, ftp = No, csrp = False

In  TCP  221  10.20.1.100:42443     10.20.221.100:23     ESTAB
Out TCP  220  10.20.220.10:23       10.20.1.100:42443    ESTAB
    vs = FROM-CLIENTS, ftp = No, csrp = False

# The command shows the open connections and how they are translated.
#
# For each connection, both halves of the connection are shown.
# The output for the second half of each connection
# swaps the source and destination IP:port.
#
# The connection originated by server 10.20.220.10 is source-NAT'ed
# and source-PAT'ed (also its L4 source port needs to be translated)
# Its source IP changes from 10.20.220.10 to 10.20.220.99
# Its source L4 port changes from 32858 to 8193
```

```
Cat6k-2# show module csm 5 real

real                  server farm      weight  state          conns/hits
-------------------------------------------------------------------------
10.20.220.10          FARM             8       OPERATIONAL    1
10.20.220.20          FARM             8       OPERATIONAL    0
10.20.220.30          FARM             8       OPERATIONAL    0
10.20.220.10          FARM2            8       OPERATIONAL    0
10.20.220.20          FARM2            8       OPERATIONAL    1
10.20.220.30          FARM2            8       OPERATIONAL    0

Cat6k-2# show module csm 5 natpool
nat client POOL-1  10.20.220.99  10.20.220.99  netmask 255.255.255.0

Cat6k-2# show module csm 5 serverfarm

server farm      type     predictor    nat   reals   redirect  bind id
-----------------------------------------------------------------------
FARM             SLB      RoundRobin   S     3       0         0
FARM2            SLB      RoundRobin   S,C   3       0         0
```

Configuring Session Persistence (Stickiness)

This example provides configuration parameters for configuring session persistence or stickiness:

```
module ContentSwitchingModule 5
 vlan 220 server
  ip address 10.20.220.2 255.255.255.0
  alias 10.20.220.1 255.255.255.0
!
 vlan 221 client
  ip address 10.20.221.5 255.255.255.0
  gateway 10.20.221.1
!
 serverfarm WEBFARM
  nat server
  no nat client
  real 10.20.220.10
   inservice
  real 10.20.220.20
   inservice
  real 10.20.220.30
   inservice
!
 sticky 10 netmask 255.255.255.255 timeout 20
!
 sticky 20 cookie yourname timeout 30
!
 vserver TELNET
  virtual 10.20.221.100 tcp telnet
  serverfarm WEBFARM
  persistent rebalance
  inservice
!
 vserver WEB1
  virtual 10.20.221.101 tcp www
  serverfarm WEBFARM
  sticky 20 group 10
  persistent rebalance
  inservice
```

```
!
 vserver WEB2
  virtual 10.20.221.102 tcp www
  serverfarm WEBFARM
  sticky 30 group 20
  persistent rebalance
  inservice
!
```

This example shows the output of the **show** commands:

```
Cat6k-2# show module csm 5 sticky group 10

group   sticky-data              real                  timeout
-----------------------------------------------------------------
10      ip 10.20.1.100           10.20.220.10          793

Cat6k-2# show module csm 5 sticky group 20

group   sticky-data              real                  timeout
-----------------------------------------------------------------
20      cookie 4C656B72:861F0395 10.20.220.20          1597


Cat6k-2# show module csm 5 sticky

group   sticky-data              real                  timeout
-----------------------------------------------------------------
20      cookie 4C656B72:861F0395 10.20.220.20          1584
10      ip 10.20.1.100           10.20.220.10          778
```

Configuring Direct Access to Servers in Router Mode

This example shows how to configure a virtual server to give direct access to the back-end servers when you are using router mode:

Note In router mode, any connection that does not hit a virtual server is dropped.

```
module ContentSwitchingModule 5
 vlan 220 server
  ip address 10.20.220.2 255.255.255.0
  alias 10.20.220.1 255.255.255.0
!
 vlan 221 client
  ip address 10.20.221.5 255.255.255.0
  gateway 10.20.221.1
  alias 10.20.221.2 255.255.255.0

# The alias IP is only required in redundant configurations
# This is the IP address that the upstream router (the MSFC
# in this case) will use as next-hop to reach the
# backend servers
# See below for the static route added for this purpose.
#
!
 serverfarm ROUTE
  no nat server
  no nat client
```

```
  predictor forward

#
# This serverfarm is not load balancing, but is simply
# routing the traffic according to the CSM-S routing tables
# The CSM-S routing table in this example is very simple,
# there is just a default gateway and 2 directly attached
# subnets.
#
# The "no nat server" is very important, since you do not
# want to rewrite the destination IP address when
# forwarding the traffic.

!
 serverfarm WEBFARM
  nat server
  no nat client
  real 10.20.220.10
   inservice
  real 10.20.220.20
   inservice
!
 vserver DIRECT-ACCESS
  virtual 10.20.220.0 255.255.255.0 tcp 0
  serverfarm ROUTE
  persistent rebalance
  inservice

# This vserver is listening to all TCP connections destined to the
# serverfarm IP subnet.
# Note: ping to the backend servers will not work with this example

!
 vserver WEB
  virtual 10.20.221.100 tcp www
  serverfarm WEBFARM
  persistent rebalance
  inservice

interface Vlan221
 ip address 10.20.221.1 255.255.255.0

# vlan221 is the L3 interface on the MSFC that connects to the CSM-S
# Client requests are being routed by the MSFC, from its other
# interfaces (not shown in this example) to vlan221.

!
ip classless
ip route 10.20.220.0 255.255.255.0 10.20.221.2

# This static route is necessary to allow the MSFC to reach
# the backend servers.
```

This example shows the output of some of the **show** commands:

```
Cat6k-2# show module csm 5 conn detail

    prot vlan source                destination           state
------------------------------------------------------------------------
In  TCP  221  10.20.1.100:44268     10.20.220.10:23       ESTAB
Out TCP  220  10.20.220.10:23       10.20.1.100:44268     ESTAB
    vs = DIRECT-ACCESS, ftp = No, csrp = False

# The information displayed shows that the CSM-S is not rewriting any IP addresses while
```

```
# forwarding theconnection from VLAN 221 (client) to VLAN 220 (server) This connection has
# been created because it was destined to the virtual server DIRECT-ACCESS.

Cat6k-2# show module csm 5 vserver detail
WEB, type = SLB, state = OPERATIONAL, v_index = 14
  virtual = 10.20.221.100/32:80 bidir, TCP, service = NONE, advertise = FALSE
  idle = 3600, replicate csrp = none, vlan = ALL, pending = 30, layer 4
  max parse len = 2000, persist rebalance = TRUE
  ssl sticky offset = 0, length = 32
  conns = 0, total conns = 0
  Default policy:
    server farm = WEBFARM, backup = <not assigned>
    sticky: timer = 0, subnet = 0.0.0.0, group id = 0
  Policy          Tot matches  Client pkts  Server pkts
  ----------------------------------------------------
  (default)       0            0            0

DIRECT-ACCESS, type = SLB, state = OPERATIONAL, v_index = 15
  virtual = 10.20.220.0/24:0 bidir, TCP, service = NONE, advertise = FALSE
  idle = 3600, replicate csrp = none, vlan = ALL, pending = 30, layer 4
  max parse len = 2000, persist rebalance = TRUE
  ssl sticky offset = 0, length = 32
  conns = 1, total conns = 1
  Default policy:
    server farm = ROUTE, backup = <not assigned>
    sticky: timer = 0, subnet = 0.0.0.0, group id = 0
  Policy          Tot matches  Client pkts  Server pkts
  ----------------------------------------------------
  (default)       1            48           35
```

Configuring Server-to-Server Load-Balanced Connections

This example shows a CSM-S configuration with three VLANs, one client, and two server VLANs. This configuration allows server-to-server load-balanced connections. There is no need for the source NAT because the source and destination servers are in separate VLANs.

```
module ContentSwitchingModule 5
 vlan 220 server
  ip address 10.20.220.2 255.255.255.0
  alias 10.20.220.1 255.255.255.0
!
 vlan 221 client
  ip address 10.20.221.5 255.255.255.0
  gateway 10.20.221.1
!
 vlan 210 server
  ip address 10.20.210.2 255.255.255.0
  alias 10.20.210.1 255.255.255.0
!
 serverfarm TIER-1
  nat server
  no nat client
  real 10.20.210.10
   inservice
  real 10.20.210.20
   inservice
!
 serverfarm TIER-2
  nat server
  no nat client
```

```
  real 10.20.220.10
   inservice
  real 10.20.220.20
   inservice
!
 vserver VIP1
  virtual 10.20.221.100 tcp telnet
  vlan 221
  serverfarm TIER-1
  persistent rebalance
  inservice
!
 vserver VIP2
  virtual 10.20.210.100 tcp telnet
  vlan 210
  serverfarm TIER-2
  persistent rebalance
  inservice
!
```

This example shows the output of some of the **show** commands:

```
Cat6k-2# show module csm 5 arp

Internet Address  Physical Interface  VLAN      Type       Status
------------------------------------------------------------------
 10.20.210.1      00-02-FC-E1-68-EB   210       -ALIAS-    local
 10.20.210.2      00-02-FC-E1-68-EC   210       --SLB--    local
 10.20.210.10     00-D0-B7-A0-68-5D   210       REAL       up(0 misses)
 10.20.210.20     00-D0-B7-A0-68-5D   210       REAL       up(0 misses)
 10.20.220.1      00-02-FC-E1-68-EB   220       -ALIAS-    local
 10.20.220.2      00-02-FC-E1-68-EC   220       --SLB--    local
 10.20.210.100    00-02-FC-E1-68-EB   0         VSERVER    local
 10.20.220.10     00-D0-B7-A0-81-D8   220       REAL       up(0 misses)
 10.20.221.1      00-02-FC-CB-70-0A   221       GATEWAY    up(0 misses)
 10.20.221.5      00-02-FC-E1-68-EC   221       --SLB--    local
 10.20.220.20     00-D0-B7-A0-81-D8   220       REAL       up(0 misses)
 10.20.221.100    00-02-FC-E1-68-EB   0         VSERVER    local

Cat6k-2# show module csm 5 vser

vserver         type  prot virtual                  vlan state        conns
------------------------------------------------------------------------------
VIP1            SLB   TCP  10.20.221.100/32:23      221  OPERATIONAL  1
VIP2            SLB   TCP  10.20.210.100/32:23      210  OPERATIONAL  1

Cat6k-2# show module csm 5 conn detail

    prot vlan source               destination          state
-------------------------------------------------------------------
In  TCP  221  10.20.1.100:44240     10.20.221.100:23     ESTAB
Out TCP  210  10.20.210.10:23       10.20.1.100:44240    ESTAB
    vs = VIP1, ftp = No, csrp = False

In  TCP  210  10.20.210.10:45885    10.20.210.100:23     ESTAB
Out TCP  220  10.20.220.10:23       10.20.210.10:45885   ESTAB
    vs = VIP2, ftp = No, csrp = False

# The previous command shows a connection opened from a client coming in from VLAN 221
# (client is 10.20.1.100). That connection goes to virtual IP address 1 (VIP1) and is
# balanced to 10.20.210.10. Another connection is opened from server 10.20.210.10, goes to
# VIP2 and is balanced to 10.20.220.10
```

Configuring Route Health Injection

The CSM-S supports virtual servers in any IP subnet. If a virtual server is configured in a subnet that is not directly attached to the MSFC, you can configure the CSM-S to inject a static route into the MSFC routing tables, depending on the health of the server farm serving that virtual server.

You can use this mechanism also for disaster recovery or GSLB solutions, where two distinct CSMs inject a static route for the same VIP. The static routes can then be redistributed, eventually with different costs, to prefer a specific location.

```
module ContentSwitchingModule 5
 vlan 220 server
  ip address 10.20.220.2 255.255.255.0
  alias 10.20.220.1 255.255.255.0
!
 vlan 221 client
  ip address 10.20.221.5 255.255.255.0
  gateway 10.20.221.1
  alias 10.20.221.2 255.255.255.0
```

The alias IP is very important because it is the IP that the CSM-S instructs the MSFC to use as the next hop to reach the advertised virtual server.

```
!
 probe PING icmp
  interval 2
  retries 2
  failed 10
  receive 2
!
 serverfarm WEBFARM
  nat server
  no nat client
  real 10.20.220.10
   inservice
  real 10.20.220.20
   inservice
  probe PING
!
 vserver WEB
  virtual 10.20.250.100 tcp www
  vlan 221

# By default, a virtual server listens to traffic coming in on any VLAN. You can restrict
# access to a virtual server by defining a specific VLAN. When using Route Health
# Injection, it is required to specify the VLAN for the virtual server. This tells the
CSM-S
# which next-hop it needs to program in the static route that it will inject in the MSFC
# routing tables.

serverfarm WEBFARM
  advertise active

# This is the command that tells the CSM-S to inject the route for this virtual server.
The
# option "active" tells the CSM-S to remove the route if the backend serverfarm fails.

persistent rebalance
  inservice
```

This example shows the output of some of the **show** commands:

```
Cat6k-2# show module csm 5 probe detail
```

```
probe           type    port  interval retries failed  open   receive
-------------------------------------------------------------------
PING            icmp          2        2       10             2
 real                  vserver         serverfarm      policy          status
 --------------------------------------------------------------------------------
 10.20.220.20:80       WEB             WEBFARM         (default)       OPERABLE
 10.20.220.10:80       WEB             WEBFARM         (default)       OPERABLE

Cat6k-2# show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS inter area
       * - candidate default, U - per-user static route, o - ODR
       P - periodic downloaded static route

Gateway of last resort is 10.20.1.100 to network 0.0.0.0

     10.0.0.0/8 is variably subnetted, 8 subnets, 3 masks
C       10.21.1.0/24 is directly connected, Vlan21
S       10.20.250.100/32 [1/0] via 10.20.221.2, Vlan221

# The static route to 10.20.250.100 has been automatically created by the CSM-S, since
both
# servers were healthy.

C       10.20.221.0/24 is directly connected, Vlan221
S*   0.0.0.0/0 [1/0] via 10.30.1.100

Cat6k-2# show module csm 5 vser detail
WEB, type = SLB, state = OPERATIONAL, v_index = 14
  virtual = 10.20.250.100/32:80 bidir, TCP, service = NONE, advertise = TRUE
  idle = 3600, replicate csrp = none, vlan = 221, pending = 30, layer 4
  max parse len = 2000, persist rebalance = TRUE
  ssl sticky offset = 0, length = 32
  conns = 0, total conns = 6
  Default policy:
    server farm = WEBFARM, backup = <not assigned>
    sticky: timer = 0, subnet = 0.0.0.0, group id = 0
  Policy          Tot matches  Client pkts  Server pkts
  ----------------------------------------------------
  (default)       6            36           30

# Failing the servers causes the route to be removed This behaviour is configured with the
# advertise active command.

Cat6k-2# show module csm 5 probe detail
1d20h: %SYS-5-CONFIG_I: Configured from console by vty0 (probe detail
probe           type    port  interval retries failed  open   receive
-------------------------------------------------------------------
PING            icmp          2        2       10             2
 real                  vserver         serverfarm      policy          status
 --------------------------------------------------------------------------------
 10.20.220.20:80       WEB             WEBFARM         (default)       TESTING
 10.20.220.10:80       WEB             WEBFARM         (default)       TESTING

Cat6k-2#
1d20h: %CSM_SLB-6-RSERVERSTATE: Module 5 server state changed: SLB-NETMGT: ICMP health
probe failed for server 10.20.220.20:80 in serverfarm 'WEBFARM'
1d20h: %CSM_SLB-6-RSERVERSTATE: Module 5 server state changed: SLB-NETMGT: ICMP health
probe failed for server 10.20.220.10:80 in serverfarm 'WEBFARM'

\Cat6k-2#
```

```
Cat6k-2# show module csm 5 probe detail
probe           type    port  interval retries failed  open   receive
------------------------------------------------------------------------
PING            icmp          2        2       10             2
 real                  vserver         serverfarm      policy          status
 ---------------------------------------------------------------------------
 10.20.220.20:80       WEB             WEBFARM         (default)       FAILED
 10.20.220.10:80       WEB             WEBFARM         (default)       FAILED
Cat6k-2#

Cat6k-2# show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS inter area
       * - candidate default, U - per-user static route, o - ODR
       P - periodic downloaded static route

Gateway of last resort is 10.20.1.100 to network 0.0.0.0
     10.0.0.0/8 is variably subnetted, 8 subnets, 3 masks
C       10.21.1.0/24 is directly connected, Vlan21
C       10.20.221.0/24 is directly connected, Vlan221
S*   0.0.0.0/0 [1/0] via 10.30.1.100
```

Configuring the Server Names

This example shows a different way to associate the servers to the server farms by using the server names. This method is preferred when the same servers are associated to multiple server farms because it allows the user to take a server out of rotation from all the server farms with only one command.

```
module ContentSwitchingModule 5
 vlan 220 server
  ip address 10.20.220.2 255.255.255.0
  alias 10.20.220.1 255.255.255.0
!
 vlan 221 client
  ip address 10.20.221.5 255.255.255.0
  gateway 10.20.221.1
  alias 10.20.221.2 255.255.255.0
!
 probe PING icmp
  interval 2
  retries 2
  failed 10
  receive 2
!
 probe FTP ftp
  interval 5
  retries 2
  failed 20
  open 3
  receive 3
!
 probe HTTP http
  request method head
  expect status 200 299
  interval 5
  retries 2
  failed 10
  open 2
```

```
  receive 2
!
 real SERVER1
  address 10.20.220.10
  inservice
 real SERVER2
  address 10.20.220.20
  inservice
!
 serverfarm FTPFARM
  nat server
  no nat client
  real name SERVER1
   inservice
  real name SERVER2
   inservice
  probe PING
  probe FTP
!
 serverfarm WEBFARM
  nat server
  no nat client
  real name SERVER1
   inservice
  real name SERVER2
   inservice
  probe PING
  probe HTTP
!
 vserver FTP
  virtual 10.20.221.100 tcp ftp service ftp
  serverfarm FTPFARM
  persistent rebalance
  inservice
!
 vserver WEB
  virtual 10.20.221.100 tcp www
  serverfarm WEBFARM
  persistent rebalance
  inservice
!
```

This example shows the output of some of the **show** commands:

```
Cat6k-2# show module csm 5 probe detail
probe           type    port  interval retries failed  open   receive
-------------------------------------------------------------------
PING            icmp          2        2       10             2
 real                   vserver         serverfarm      policy          status
 --------------------------------------------------------------------------------
 10.20.220.20:21        FTP             FTPFARM         (default)       OPERABLE
 10.20.220.10:21        FTP             FTPFARM         (default)       OPERABLE
 10.20.220.20:80        WEB             WEBFARM         (default)       OPERABLE
 10.20.220.10:80        WEB             WEBFARM         (default)       OPERABLE
FTP             ftp           5        2       20      3      3
 Expected Status Codes:
  0 to 999
 real                   vserver         serverfarm      policy          status
 --------------------------------------------------------------------------------
 10.20.220.20:21        FTP             FTPFARM         (default)       OPERABLE
 10.20.220.10:21        FTP             FTPFARM         (default)       OPERABLE
HTTP            http          5        2       10      2      2
 Probe Request:  HEAD       /
 Expected Status Codes:
```

```
 200 to 299
real                  vserver         serverfarm      policy          status
--------------------------------------------------------------------------------
10.20.220.20:80       WEB             WEBFARM         (default)       OPERABLE
10.20.220.10:80       WEB             WEBFARM         (default)       OPERABLE

Cat6k-2# show module csm 5 real

real                  server farm      weight  state          conns/hits
--------------------------------------------------------------------------------
SERVER1               FTPFARM          8       OPERATIONAL    0
SERVER2               FTPFARM          8       OPERATIONAL    0
SERVER1               WEBFARM          8       OPERATIONAL    0
SERVER2               WEBFARM          8       OPERATIONAL    0

# Taking a server out of service at the server farm level will only take the server out of
# service for that specific farm

Cat6k-2# configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
Cat6k-2(config)# module csm 5
Cat6k-2(config-module-csm)# server webfarm
Cat6k-2(config-slb-sfarm)# real name server1
Cat6k-2(config-slb-real)# no inservice
Cat6k-2(config-slb-real)# end
1d20h: %CSM_SLB-6-RSERVERSTATE: Module 5 server state changed: SLB-NETMGT: Configured
server 10.20.220.10:0 to OUT-OF-SERVICE in serverfarm 'WEBFARM'
Cat6k-2#
1d20h: %SYS-5-CONFIG_I: Configured from console by vty0 (10.20.1.100)
Cat6k-2#
Cat6k-2# show module csm 5 real

real                  server farm      weight  state          conns/hits
--------------------------------------------------------------------------------
SERVER1               FTPFARM          8       OPERATIONAL    0
SERVER2               FTPFARM          8       OPERATIONAL    0
SERVER1               WEBFARM          8       OUTOFSERVICE   0
SERVER2               WEBFARM          8       OPERATIONAL    0
Cat6k-2#

# Taking the server out of service at the real server level will take the server out of
# service for all the server farms

Cat6k-2# confure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
Cat6k-2(config)# module csm 5
Cat6k-2(config-module-csm)# real server1
Cat6k(config-slb-module-real)# no inservice
Cat6k(config-slb-module-real)# end
Cat6k-2#
1d20h: %SYS-5-CONFIG_I: Configured from console by vty0 (10.20.1.100)
Cat6k-2# show module csm 5 real

real                  server farm      weight  state          conns/hits
--------------------------------------------------------------------------------
SERVER1               FTPFARM          8       OUTOFSERVICE   0
SERVER2               FTPFARM          8       OPERATIONAL    0
SERVER1               WEBFARM          8       OUTOFSERVICE   0
SERVER2               WEBFARM          8       OPERATIONAL    0
Cat6k-2#
```

Configuring a Backup Server Farm

This example shows you how to configure a backup server farm for a virtual server. If all the servers in the primary server farm fail, the CSM-S starts directing requests to the backup server farm. The sticky options allow you to control the backup operation if stickiness is configured for that virtual server.

```
module ContentSwitchingModule 5
 vlan 220 server
  ip address 10.20.220.2 255.255.255.0
  alias 10.20.220.1 255.255.255.0
!
 vlan 221 client
  ip address 10.20.221.5 255.255.255.0
  gateway 10.20.221.1
  alias 10.20.221.2 255.255.255.0
!
 vlan 210 server
  ip address 10.20.210.2 255.255.255.0
  alias 10.20.210.1 255.255.255.0
!
 probe PING icmp
  interval 2
  retries 2
  failed 10
  receive 2
!
 real SERVER1
  address 10.20.220.10
  inservice
 real SERVER2
  address 10.20.220.20
  inservice
 real SERVER3
  address 10.20.210.30
  inservice
 real SERVER4
  address 10.20.210.40
  inservice
!
 serverfarm WEBFARM
  nat server
  no nat client
  real name SERVER1
   inservice
  real name SERVER2
   inservice
  probe PING
!
 serverfarm WEBFARM2
  nat server
  no nat client
  real name SERVER3
   inservice
  real name SERVER4
   inservice
  probe PING
!
 vserver WEB
  virtual 10.20.221.100 tcp www
  serverfarm WEBFARM backup WEBFARM2
  persistent rebalance
  inservice
```

```
!
```

This example shows the output of some of the **show** commands:

```
Cat6k-2# show module csm 5 real

real                 server farm      weight  state          conns/hits
--------------------------------------------------------------------------
SERVER1              WEBFARM          8       OPERATIONAL    0
SERVER2              WEBFARM          8       OPERATIONAL    0
SERVER3              WEBFARM2         8       OPERATIONAL    0
SERVER4              WEBFARM2         8       OPERATIONAL    0

# All the servers are shown as operational.

Cat6k-2# show module csm 5 serverfarm detail
WEBFARM, type = SLB, predictor = RoundRobin
  nat = SERVER
  virtuals inservice = 1, reals = 2, bind id = 0, fail action = none
  inband health config: <none>
  retcode map = <none>
  Probes:
    PING, type = icmp
  Real servers:
    SERVER1, weight = 8, OPERATIONAL, conns = 0
    SERVER2, weight = 8, OPERATIONAL, conns = 0
  Total connections = 0

WEBFARM2, type = SLB, predictor = RoundRobin
  nat = SERVER
  virtuals inservice = 1, reals = 2, bind id = 0, fail action = none
  inband health config: <none>
  retcode map = <none>
  Probes:
    PING, type = icmp
  Real servers:
    SERVER3, weight = 8, OPERATIONAL, conns = 0
    SERVER4, weight = 8, OPERATIONAL, conns = 0
  Total connections = 0

Cat6k-2# show module csm 5 vserver detail
WEB, type = SLB, state = OPERATIONAL, v_index = 18
  virtual = 10.20.221.100/32:80 bidir, TCP, service = NONE, advertise = FALSE
  idle = 3600, replicate csrp = none, vlan = ALL, pending = 30, layer 4
  max parse len = 2000, persist rebalance = TRUE
  ssl sticky offset = 0, length = 32
  conns = 0, total conns = 0
  Default policy:
    server farm = WEBFARM, backup = WEBFARM2 (no sticky)
    sticky: timer = 0, subnet = 0.0.0.0, group id = 0
  Policy          Tot matches  Client pkts  Server pkts
  ------------------------------------------------------
  (default)       0            0            0

# No connections have been sent to the virtual server yet.

Cat6k-2# show module csm 5 vserver detail
WEB, type = SLB, state = OPERATIONAL, v_index = 18
  virtual = 10.20.221.100/32:80 bidir, TCP, service = NONE, advertise = FALSE
  idle = 3600, replicate csrp = none, vlan = ALL, pending = 30, layer 4
  max parse len = 2000, persist rebalance = TRUE
  ssl sticky offset = 0, length = 32
  conns = 0, total conns = 14
  Default policy:
```

```
    server farm = WEBFARM, backup = WEBFARM2 (no sticky)
    sticky: timer = 0, subnet = 0.0.0.0, group id = 0
  Policy          Tot matches  Client pkts  Server pkts
  ---------------------------------------------------
  (default)       14           84           70

# A total of 14 connections have been sent to the virtual server and have been balanced to
# the primary server farm. For each connection, the client has sent 6 packets and the #
server has sent 5 packets. Two servers are taken out of service

Cat6k-2#
1d21h: %CSM_SLB-6-RSERVERSTATE: Module 5 server state changed: SLB-NETMGT: ICMP health
probe failed for server 10.20.220.10:80 in serverfarm 'WEBFARM'
1d21h: %CSM_SLB-6-RSERVERSTATE: Module 5 server state changed: SLB-NETMGT: ICMP health
probe failed for server 10.20.220.20:80 in serverfarm 'WEBFARM'

Cat6k-2# show module csm 5 serverfarm detail
WEBFARM, type = SLB, predictor = RoundRobin
  nat = SERVER
  virtuals inservice = 1, reals = 2, bind id = 0, fail action = none
  inband health config: <none>
  retcode map = <none>
  Probes:
    PING, type = icmp
  Real servers:
    SERVER1, weight = 8, PROBE_FAILED, conns = 0
    SERVER2, weight = 8, PROBE_FAILED, conns = 0
  Total connections = 0

# The two servers have failed the probe but the CSM-S has not yet refreshed the ARP table
# for them, so the servers are not yet shown in the failed state

WEBFARM2, type = SLB, predictor = RoundRobin
  nat = SERVER
  virtuals inservice = 1, reals = 2, bind id = 0, fail action = none
  inband health config: <none>
  retcode map = <none>
  Probes:
    PING, type = icmp
  Real servers:
    SERVER3, weight = 8, OPERATIONAL, conns = 0
    SERVER4, weight = 8, OPERATIONAL, conns = 0
  Total connections = 0

Cat6k-2# show module csm 5 vserver detail
WEB, type = SLB, state = OUTOFSERVICE, v_index = 18
  virtual = 10.20.221.100/32:80 bidir, TCP, service = NONE, advertise = FALSE
  idle = 3600, replicate csrp = none, vlan = ALL, pending = 30, layer 4
  max parse len = 2000, persist rebalance = TRUE
  ssl sticky offset = 0, length = 32
  conns = 0, total conns = 14
  Default policy:
    server farm = WEBFARM, backup = WEBFARM2 (no sticky)
    sticky: timer = 0, subnet = 0.0.0.0, group id = 0
  Policy          Tot matches  Client pkts  Server pkts
  ---------------------------------------------------
  (default)       14           83           70

# The virtual server is displayed as out of service, even if it is configured with a
# backup server farm, which is healthy. This behaviour is useful if the backup server farm
# is configured as an HTTP redirect server farm to a different site and you are using some
# DNS-based GSLB method, where some connections are still being directed to the failed
# virtual server.
```

```
# If you want the CSM-S to consider the virtual server healthy and operational if the
backup
# server farm is healthy, you just need to change an environmental variable.

Cat6k-2# show module csm 5 variable

variable                        value
------------------------------------------------------------------
ARP_INTERVAL                    300
ARP_LEARNED_INTERVAL            14400
ARP_GRATUITOUS_INTERVAL         15
ARP_RATE                        10
ARP_RETRIES                     3
ARP_LEARN_MODE                  1
ARP_REPLY_FOR_NO_INSERVICE_VIP  0
ADVERTISE_RHI_FREQ              10
AGGREGATE_BACKUP_SF_STATE_TO_VS 0
DEST_UNREACHABLE_MASK           0xffff
FT_FLOW_REFRESH_INT             15
GSLB_LICENSE_KEY                (no valid license)
HTTP_CASE_SENSITIVE_MATCHING    1
MAX_PARSE_LEN_MULTIPLIER        1
NAT_CLIENT_HASH_SOURCE_PORT     0
ROUTE_UNKNOWN_FLOW_PKTS         0
NO_RESET_UNIDIRECTIONAL_FLOWS   0
SYN_COOKIE_INTERVAL             3
SYN_COOKIE_THRESHOLD            5000
TCP_MSS_OPTION                  1460
TCP_WND_SIZE_OPTION             8192
VSERVER_ICMP_ALWAYS_RESPOND     false
XML_CONFIG_AUTH_TYPE            Basic

# The variable that you want to change is AGGREGATE_BACKUP_SF_STATE_TO_VS

Cat6k-2#
1d21h: %CSM_SLB-6-RSERVERSTATE: Module 5 server state changed: SLB-NETMGT: Server
10.20.220.20 failed ARP request
Cat6k-2#

# The CSM-S has refreshed the ARP entry for 10.20.220.20 which is now reported in the
failed
state.

Cat6k-2# configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
Cat6k-2(config)# module csm 5
Cat6k-2(config-module-csm)# variable AGGREGATE_BACKUP_SF_STATE_TO_VS 1
Cat6k-2(config-module-csm)# end

1d21h: %SYS-5-CONFIG_I: Configured from console by vty0 (10.20.1.100)

Cat6k-2# show module csm 5 variable

variable                        value
------------------------------------------------------------------
ARP_INTERVAL                    300
ARP_LEARNED_INTERVAL            14400
ARP_GRATUITOUS_INTERVAL         15
ARP_RATE                        10
ARP_RETRIES                     3
ARP_LEARN_MODE                  1
ARP_REPLY_FOR_NO_INSERVICE_VIP  0
ADVERTISE_RHI_FREQ              10
AGGREGATE_BACKUP_SF_STATE_TO_VS 1
```

```
DEST_UNREACHABLE_MASK           0xffff
FT_FLOW_REFRESH_INT             15
GSLB_LICENSE_KEY                (no valid license)
HTTP_CASE_SENSITIVE_MATCHING    1
MAX_PARSE_LEN_MULTIPLIER        1
NAT_CLIENT_HASH_SOURCE_PORT     0
ROUTE_UNKNOWN_FLOW_PKTS         0
NO_RESET_UNIDIRECTIONAL_FLOWS   0
SYN_COOKIE_INTERVAL             3
SYN_COOKIE_THRESHOLD            5000
TCP_MSS_OPTION                  1460
TCP_WND_SIZE_OPTION             8192
VSERVER_ICMP_ALWAYS_RESPOND     false
XML_CONFIG_AUTH_TYPE            Basic

Cat6k-2# show module csm 5 vserver detail
WEB, type = SLB, state = OPERATIONAL, v_index = 18
  virtual = 10.20.221.100/32:80 bidir, TCP, service = NONE, advertise = FALSE
  idle = 3600, replicate csrp = none, vlan = ALL, pending = 30, layer 4
  max parse len = 2000, persist rebalance = TRUE
  ssl sticky offset = 0, length = 32
  conns = 0, total conns = 14
  Default policy:
    server farm = WEBFARM, backup = WEBFARM2 (no sticky)
    sticky: timer = 0, subnet = 0.0.0.0, group id = 0
  Policy          Tot matches  Client pkts  Server pkts
  ------------------------------------------------------
  (default)       14           83           70

# The virtual server is now shown as operational.

Cat6k-2# show module csm 5 real detail
SERVER1, WEBFARM, state = PROBE_FAILED
  address = 10.20.220.10, location = <NA>
  conns = 0, maxconns = 4294967295, minconns = 0
  weight = 8, weight(admin) = 8, metric = 0, remainder = 0
  total conns established = 7, total conn failures = 0
SERVER2, WEBFARM, state = FAILED
  address = 10.20.220.20, location = <NA>
  conns = 0, maxconns = 4294967295, minconns = 0
  weight = 8, weight(admin) = 8, metric = 0, remainder = 0
  total conns established = 7, total conn failures = 0
SERVER3, WEBFARM2, state = OPERATIONAL
  address = 10.20.210.30, location = <NA>
  conns = 0, maxconns = 4294967295, minconns = 0
  weight = 8, weight(admin) = 8, metric = 0, remainder = 0
  total conns established = 0, total conn failures = 0
SERVER4, WEBFARM2, state = OPERATIONAL
  address = 10.20.210.40, location = <NA>
  conns = 0, maxconns = 4294967295, minconns = 0
  weight = 8, weight(admin) = 8, metric = 0, remainder = 0
  total conns established = 0, total conn failures = 0
Cat6k-2#

1d21h: %CSM-S_SLB-6-RSERVERSTATE: Module 5 server state changed: SLB-NETMGT: Server
10.20.220.10 failed ARP request

# The ARP entry for the other server has been refreshed.

Cat6k-2# show module csm 5 real detail
SERVER1, WEBFARM, state = FAILED
  address = 10.20.220.10, location = <NA>
  conns = 0, maxconns = 4294967295, minconns = 0
  weight = 8, weight(admin) = 8, metric = 0, remainder = 0
```

```
  total conns established = 7, total conn failures = 0
SERVER2, WEBFARM, state = FAILED
  address = 10.20.220.20, location = <NA>
  conns = 0, maxconns = 4294967295, minconns = 0
  weight = 8, weight(admin) = 8, metric = 0, remainder = 0
  total conns established = 7, total conn failures = 0
SERVER3, WEBFARM2, state = OPERATIONAL
  address = 10.20.210.30, location = <NA>
  conns = 0, maxconns = 4294967295, minconns = 0
  weight = 8, weight(admin) = 8, metric = 0, remainder = 0
  total conns established = 0, total conn failures = 0
SERVER4, WEBFARM2, state = OPERATIONAL
  address = 10.20.210.40, location = <NA>
  conns = 0, maxconns = 4294967295, minconns = 0
  weight = 8, weight(admin) = 8, metric = 0, remainder = 0
  total conns established = 0, total conn failures = 0

# So far, each of the servers in the primary server farm have received 7 connections. New
# connections are now sent only to the backup server farm.

Cat6k-2# show module csm 5 real detail
SERVER1, WEBFARM, state = FAILED
  address = 10.20.220.10, location = <NA>
  conns = 0, maxconns = 4294967295, minconns = 0
  weight = 8, weight(admin) = 8, metric = 0, remainder = 0
  total conns established = 7, total conn failures = 0
SERVER2, WEBFARM, state = FAILED
  address = 10.20.220.20, location = <NA>
  conns = 0, maxconns = 4294967295, minconns = 0
  weight = 8, weight(admin) = 8, metric = 0, remainder = 0
  total conns established = 7, total conn failures = 0
SERVER3, WEBFARM2, state = OPERATIONAL
  address = 10.20.210.30, location = <NA>
  conns = 0, maxconns = 4294967295, minconns = 0
  weight = 8, weight(admin) = 8, metric = 0, remainder = 0
  total conns established = 6, total conn failures = 0
SERVER4, WEBFARM2, state = OPERATIONAL
  address = 10.20.210.40, location = <NA>
  conns = 0, maxconns = 4294967295, minconns = 0
  weight = 8, weight(admin) = 8, metric = 0, remainder = 0
  total conns established = 6, total conn failures = 0
Cat6k-2#
```

Configuring a Load-Balancing Decision Based on the Source IP Address

This example shows how to make a load-balancing decision based on the source IP address of the client. This configuration requires the use of SLB-policies.

```
module ContentSwitchingModule 5
 vlan 220 server
  ip address 10.20.220.2 255.255.255.0
  alias 10.20.220.1 255.255.255.0
!
 vlan 221 client
  ip address 10.20.221.5 255.255.255.0
  gateway 10.20.221.1
  alias 10.20.221.2 255.255.255.0
!
```

```
 probe PING icmp
  interval 2
  retries 2
  failed 10
  receive 2
!
 real SERVER1
  address 10.20.220.10
  inservice
 real SERVER2
  address 10.20.220.20
  inservice
 real SERVER3
  address 10.20.220.30
  inservice
 real SERVER4
  address 10.20.220.40
  inservice
!
 serverfarm WEBFARM
  nat server
  no nat client
  real name SERVER1
   inservice
  real name SERVER2
   inservice
  probe PING
!
 serverfarm WEBFARM2
  nat server
  no nat client
  real name SERVER3
   inservice
  real name SERVER4
   inservice
!
 policy SOURCE-IP-50
  client-group 50
  serverfarm WEBFARM2

# A policy consists of a series of conditions, plus the actions to take if those
# conditions are matched. In this case, the only condition is client-group 50 which
# requires the incoming connection to match the standard access-list 50. The only action
# to take is to use server farm WEBFARM2 to serve those requests.

!
 vserver WEB
  virtual 10.20.221.100 tcp www
  serverfarm WEBFARM
  persistent rebalance
  slb-policy SOURCE-IP-50

# Slb-policies associated to a virtual server are always examined in the order in which
# they are configured. The defintion of the server farm under the virtual server
# configuration is the default policy and is always used as a last resort if no policy
# matches, or if there are no policies configured.

# In this case, incoming requests are processed to see if they match the conditions of the
# slb-policy SOURCe-IP-50. If they do, then the server farm WEBFARM2 is used, otherwise
# the default policy is selected (for example, WEBFARM is used).

# If a default server farm is not configured, then connections that do not match any
# policy are dropped.
```

```
# This example shows how to configure the IOS standard access list. You can configure any
# of the 1-99 standard access lists, or you can configure named access lists

inservice
!
access-list 50 permit 10.20.1.100
```

This example shows the output of some of the **show** commands:

```
Cat6k-2# show module csm 5 vser detail
WEB, type = SLB, state = OPERATIONAL, v_index = 18
  virtual = 10.20.221.100/32:80 bidir, TCP, service = NONE, advertise = FALSE
  idle = 3600, replicate csrp = none, vlan = ALL, pending = 30, layer 4
  max parse len = 2000, persist rebalance = TRUE
  ssl sticky offset = 0, length = 32
  conns = 0, total conns = 0
  Default policy:
    server farm = WEBFARM, backup = <not assigned>
    sticky: timer = 0, subnet = 0.0.0.0, group id = 0
  Policy          Tot matches  Client pkts  Server pkts
  ------------------------------------------------------
  SOURCE-IP-50    0            0            0
  (default)       0            0            0

# This example shows that six connections have matched the slb-policy SOURCE-IP-50.

Cat6k-2# show module csm 5 vser detail
WEB, type = SLB, state = OPERATIONAL, v_index = 18
  virtual = 10.20.221.100/32:80 bidir, TCP, service = NONE, advertise = FALSE
  idle = 3600, replicate csrp = none, vlan = ALL, pending = 30, layer 4
  max parse len = 2000, persist rebalance = TRUE
  ssl sticky offset = 0, length = 32
  conns = 0, total conns = 6
  Default policy:
    server farm = WEBFARM, backup = <not assigned>
    sticky: timer = 0, subnet = 0.0.0.0, group id = 0
  Policy          Tot matches  Client pkts  Server pkts
  ------------------------------------------------------
  SOURCE-IP-50    6            36           30
  (default)       0            0            0

# This example shows that SERVER3 and SERVER4 have received 3 connections each.

Cat6k-2# show module csm 5 real detail
SERVER1, WEBFARM, state = OPERATIONAL
  address = 10.20.220.10, location = <NA>
  conns = 0, maxconns = 4294967295, minconns = 0
  weight = 8, weight(admin) = 8, metric = 0, remainder = 0
  total conns established = 0, total conn failures = 0
SERVER2, WEBFARM, state = OPERATIONAL
  address = 10.20.220.20, location = <NA>
  conns = 0, maxconns = 4294967295, minconns = 0
  weight = 8, weight(admin) = 8, metric = 0, remainder = 0
  total conns established = 0, total conn failures = 0
SERVER3, WEBFARM2, state = OPERATIONAL
  address = 10.20.220.30, location = <NA>
  conns = 0, maxconns = 4294967295, minconns = 0
  weight = 8, weight(admin) = 8, metric = 0, remainder = 0
  total conns established = 3, total conn failures = 0
SERVER4, WEBFARM2, state = OPERATIONAL
  address = 10.20.220.40, location = <NA>
  conns = 0, maxconns = 4294967295, minconns = 0
  weight = 8, weight(admin) = 8, metric = 0, remainder = 0
  total conns established = 3, total conn failures = 0
```

Configuring Layer 7 Load Balancing

This example shows how to make load-balancing decisions based on Layer 7 information. In this case, the CSM-S terminates the TCP connection, buffers the request, and parses it to see if the request matches the policy conditions. When a load-balancing decision is made, the CSM-S opens the connection to the selected server and splices the two flows together.

The configuration in this example requires the use of maps and policies. A policy is a list of conditions and actions that are taken if all the conditions are true.

```
Cat6k-2(config-module-csm)# policy test
Cat6k-2(config-slb-policy)# ?
SLB policy config
  client-group    define policy client group
  cookie-map      define policy cookie map
  default         Set a command to its defaults
  exit            exit slb policy submode
  header-map      define policy header map
  no              Negate a command or set its defaults
  reverse-sticky  define sticky group for reverse traffic
  serverfarm      define policy serverfarm
  set             set policy parameters
  sticky-group    define policy sticky group
  url-map         define policy URL map

# The conditions are:
# -client-group (source IP matches a certain ACL)
# -cookie-map (match based on cookies)
# -header-map (match based on HTTP headers)
# -url-map (match based on URLs)

# The actions are:
# -serverfarm (the most common: use this serverfarm)
# -sticky-group (use sticky)
# -reverse-sticky (use reverse sticky)
# -set (set ip dscp)

\module ContentSwitchingModule 5
 vlan 220 server
  ip address 10.20.220.2 255.255.255.0
  alias 10.20.220.1 255.255.255.0
!
 vlan 221 client
  ip address 10.20.221.5 255.255.255.0
  gateway 10.20.221.1
  alias 10.20.221.2 255.255.255.0
!
 probe PING icmp
  interval 2
  retries 2
  failed 10
  receive 2
!
 map TEST header
  match protocol http header Host header-value www.test.com
!
 map SPORTS url
  match protocol http url /sports/*

# The definition of maps is based on the header and the URL. The URL starts right after
# the host. For example, in the URL http://www.test.com/sports/basketball/ the URL portion
# that the URL map applies to is /sports/basketball/.
```

```
!
 real SERVER1
  address 10.20.220.10
  inservice
 real SERVER2
  address 10.20.220.20
  inservice
 real SERVER3
  address 10.20.220.30
  inservice
 real SERVER4
  address 10.20.220.40
  inservice
!
 serverfarm WEBFARM
  nat server
  no nat client
  real name SERVER1
   inservice
  real name SERVER2
   inservice
  probe PING
!
 serverfarm WEBFARM2
  nat server
  no nat client
  real name SERVER3
   inservice
  real name SERVER4
   inservice
!
 policy TEST-SPORTS-50
  url-map SPORTS
  header-map TEST
  client-group 50
  serverfarm WEBFARM2

# Three conditions need to match for this policy to have a match.

!
 vserver WEB
  virtual 10.20.221.100 tcp www
  serverfarm WEBFARM
  persistent rebalance
  slb-policy TEST-SPORTS-50
  inservice
!
# If the three conditions defined in the policy are true then WEBFARM2 is used otherwise
# WEBFARM is.
```

This example shows the output of some of the **show** commands:

```
# In this example, 17 requests have matched the policy Of those, 12 requests have not
# matched the policy

Cat6k-2# show module csm 5 vserver detail
WEB, type = SLB, state = OPERATIONAL, v_index = 18
  virtual = 10.20.221.100/32:80 bidir, TCP, service = NONE, advertise = FALSE
  idle = 3600, replicate csrp = none, vlan = ALL, pending = 30, layer 4
  max parse len = 2000, persist rebalance = TRUE
  ssl sticky offset = 0, length = 32
  conns = 0, total conns = 29
  Default policy:
```

```
    server farm = WEBFARM, backup = <not assigned>
    sticky: timer = 0, subnet = 0.0.0.0, group id = 0
  Policy          Tot matches  Client pkts  Server pkts
  ---------------------------------------------------
  TEST-SPORTS-50  17           112          95
  (default)       12           82           72

# This example shows that the 29 connections that were load balanced have been load
# balanced at Layer 7. For example, the CSM-S has to terminate TCP and parse Layer 5
through
# Layer 7 information.

Cat6k-2# show module csm 5 stats
Connections Created:        29
Connections Destroyed:      29
Connections Current:        0
Connections Timed-Out:      0
Connections Failed:         0
Server initiated Connections:
      Created: 0, Current: 0, Failed: 0
L4 Load-Balanced Decisions: 0
L4 Rejected Connections:    0
L7 Load-Balanced Decisions: 29
L7 Rejected Connections:
      Total: 0, Parser: 0,
      Reached max parse len: 0, Cookie out of mem: 0,
      Cfg version mismatch: 0, Bad SSL2 format: 0
L4/L7 Rejected Connections:
      No policy: 0, No policy match 0,
      No real: 0, ACL denied 0,
      Server initiated: 0
Checksum Failures:  IP: 0, TCP: 0
Redirect Connections: 0,  Redirect Dropped: 0
FTP Connections:            0
MAC Frames:
      Tx: Unicast: 359, Multicast: 0, Broadcast: 8,
          Underflow Errors: 0
      Rx: Unicast: 387, Multicast: 221, Broadcast: 1,
          Overflow Errors: 0, CRC Errors: 0
```

Configuring HTTP Redirect

This example shows how you can configure the CSM-S to send HTTP redirect messages:

```
# This configuration represents the configuration of site A

module ContentSwitchingModule 6
 vlan 211 client
  ip address 10.20.211.2 255.255.255.0
  gateway 10.20.211.1
!
 vlan 210 server
  ip address 10.20.210.1 255.255.255.0
!
 map SPORTMAP url
  match protocol http url /sports*
!
 serverfarm REDIRECTFARM
  nat server
  no nat client
```

```
  redirect-vserver WWW2
   webhost relocation www2.test.com 301
   inservice
!
 serverfarm WWW1FARM
  nat server
  no nat client
  real 10.20.210.10
   inservice
  real 10.20.210.20
   inservice
!
 policy SPORTPOLICY
  url-map SPORTMAP
  serverfarm REDIRECTFARM
!
 vserver WWW1VIP
  virtual 10.20.211.100 tcp www
  serverfarm WWW1FARM
  persistent rebalance
  slb-policy SPORTPOLICY
  inservice

# This configuration represents the configuration of site B

module ContentSwitchingModule 7
 vlan 221 client
  ip address 10.20.221.2 255.255.255.0
  gateway 10.20.221.1
!
 vlan 220 server
  ip address 10.20.220.1 255.255.255.0
!
 serverfarm WWW2FARM
  nat server
  no nat client
  real 10.20.220.10
   inservice
  real 10.20.220.20
   inservice
!
 vserver WWW2VIP
  virtual 10.20.221.100 tcp www
  serverfarm WWW2FARM
  persistent rebalance
  inservice
```

This example shows the output of some of the **show** commands:

```
# To test the configuration, the first nine requests are sent to www1.test.com requesting
# the home page "/." The 10th request is sent to http://www1.test.com/sports/.

Cat6k-2# show module csm 6 vser deta
WWW1VIP, type = SLB, state = OPERATIONAL, v_index = 11
  virtual = 10.20.211.100/32:80 bidir, TCP, service = NONE, advertise = FALSE
  idle = 3600, replicate csrp = none, vlan = ALL, pending = 30
  max parse len = 2000, persist rebalance = TRUE
  ssl sticky offset = 0, length = 32
  conns = 0, total conns = 10
  Default policy:
    server farm = WWW1FARM, backup = <not assigned>
    sticky: timer = 0, subnet = 0.0.0.0, group id = 0
  Policy          Tot Conn     Client pkts  Server pkts
  ------------------------------------------------------
```

```
  SPORTPOLICY     1            3            1
  (default)       9            45           45

Cat6k-2# show module csm 7 vser detail
WWW2VIP, type = SLB, state = OPERATIONAL, v_index = 26
  virtual = 10.20.221.100/32:80 bidir, TCP, service = NONE, advertise = FALSE
  idle = 3600, replicate csrp = none, vlan = ALL, pending = 30
  max parse len = 2000, persist rebalance = TRUE
  ssl sticky offset = 0, length = 32
  conns = 0, total conns = 1
  Default policy:
    server farm = WWW2FARM, backup = <not assigned>
    sticky: timer = 0, subnet = 0.0.0.0, group id = 0
  Policy          Tot Conn     Client pkts  Server pkts
  ------------------------------------------------------
  (default)       1            5            5

# Nine requests have matched the default policy for www1.test.com so they have been served
# by WWW1FARM. One request has matched the policy SPORTPOLICY and has been redirected to
# the second site that has then served the request.

# The following is an example of the request that was sent to www1.cisco.com asking for
# /sports/.

10.20.1.100.34589 > 10.20.211.100.80: P 1:287(286) ack 1 win 5840 (DF)
0x0000   4500 0146 763c 4000 4006 da85 0a14 0164        E..Fv<@.@......d
0x0010   0a14 d364 871d 0050 ec1d 69e6 7b57 aead        ...d...P..i.{W..
0x0020   5018 16d0 96b2 0000 4745 5420 2f73 706f        P.......GET./spo
0x0030   7274 732f 2048 5454 502f 312e 310d 0a43        rts/.HTTP/1.1..C
0x0040   6f6e 6e65 6374 696f 6e3a 204b 6565 702d        onnection:.Keep-
0x0050   416c 6976 650d 0a55 7365 722d 4167 656e        Alive..User-Agen
0x0060   743a 204d 6f7a 696c 6c61 2f35 2e30 2028        t:.Mozilla/5.0.(
0x0070   636f 6d70 6174 6962 6c65 3b20 4b6f 6e71        compatible;.Konq
0x0080   7565 726f 722f 322e 322d 3131 3b20 4c69        ueror/2.2-11;.Li
0x0090   6e75 7829 0d0a 4163 6365 7074 3a20 7465        nux)..Accept:.te
0x00a0   7874 2f2a 2c20 696d 6167 652f 6a70 6567        xt/*,.image/jpeg
0x00b0   2c20 696d 6167 652f 706e 672c 2069 6d61        ,.image/png,.ima
0x00c0   6765 2f2a 2c20 2a2f 2a0d 0a41 6363 6570        ge/*,.*/*..Accep
0x00d0   742d 456e 636f 6469 6e67 3a20 782d 677a        t-Encoding:.x-gz
0x00e0   6970 2c20 677a 6970 2c20 6964 656e 7469        ip,.gzip,.identi
0x00f0   7479 0d0a 4163 6365 7074 2d43 6861 7273        ty..Accept-Chars
0x0100   6574 3a20 416e 792c 2075 7466 2d38 2c20        et:.Any,.utf-8,.
0x0110   2a0d 0a41 6363 6570 742d 4c61 6e67 7561        *..Accept-Langua
0x0120   6765 3a20 656e 5f55 532c 2065 6e0d 0a48        ge:.en_US,.en..H
0x0130   6f73 743a 2077 7777 312e 7465 7374 2e63        ost:.www1.test.c
0x0140   6f6d 0d0a 0d0a                                 om....

# The following example is the message that the client has received back from
# www1.cisco.com. This message is the HTTP redirect message generated by the CSM-S

10.20.211.100.80 > 10.20.1.100.34589: FP 1:56(55) ack 287 win 2048 (DF)
0x0000   4500 005f 763c 4000 3e06 dd6c 0a14 d364        E.._v<@.>..l...d
0x0010   0a14 0164 0050 871d 7b57 aead ec1d 6b04        ...d.P..{W....k.
0x0020   5019 0800 8b1a 0000 4854 5450 2f31 2e30        P.......HTTP/1.0
0x0030   2033 3031 2046 6f75 6e64 200d 0a4c 6f63        .301.Found...Loc
0x0040   6174 696f 6e3a 2068 7474 703a 2f2f 7777        ation:.http://ww
0x0050   7732 2e74 6573 742e 636f 6d0d 0a0d 0a          w2.test.com....

# The redirect location sent back to the client matches exactly the string configured with
# the webhost relocation www2.test.com 301 command because the client was browsing
# www1.test.com/sports/ and is redirected to www2.test.com/.

# In some cases this might not be the desired behaviour and there might be the need to
# preserve the original URL that the browser requested.
```

```
# To preseerve the URL that the browser requested, you can use the %p parameter as part of
# the redirect string.

# The configuration would then appear as:

#   serverfarm REDIRECTFARM
#     nat server
#     no nat client
#     redirect-vserver WWW2
#       webhost relocation www2.test.com/%p
#       inservice

# The following example shows the resulting redirect message which is sent back to the
# client:

10.20.211.100.80 > 10.20.1.100.34893: FP 1:64(63) ack 329 win 2048 (DF)
0x0000   4500 0067 7d95 4000 3e06 d60b 0a14 d364        E..g}.@.>......d
0x0010   0a14 0164 0050 884d 7093 b53b 4e0b e8a8        ...d.P.Mp..;N...
0x0020   5019 0800 2800 0000 4854 5450 2f31 2e30        P...(...HTTP/1.0
0x0030   2033 3032 2046 6f75 6e64 200d 0a4c 6f63        .302.Found...Loc
0x0040   6174 696f 6e3a 2068 7474 703a 2f2f 7777        ation:.http://ww
0x0050   7732 2e74 6573 742e 636f 6d2f 7370 6f72        w2.test.com/spor
0x0060   7473 2f0d 0a0d 0a                              ts/....

# In other cases, you may need to redirect an HTTP request to an HTTPS VIP, on the same or
# on a remote CSM-S. In that case, the URL request must change from http:// to https://
# You can do this by using the parameter ssl 443

# The configuration would then be as follows:

#   serverfarm REDIRECTFARM
#     nat server
#     no nat client
#     redirect-vserver WWW2
#       webhost relocation www2.test.com/%p
#       ssl 443
#       inservice

# The following is the resulting redirect message sent back to the client.

10.20.211.100.80 > 10.20.1.100.34888: FP 1:65(64) ack 329 win 2048 (DF)
0x0000   4500 0068 2cda 4000 3e06 26c6 0a14 d364        E..h,.@.>.&....d
0x0010   0a14 0164 0050 8848 7088 b087 21e5 a627        ...d.P.Hp...!..'
0x0020   5019 0800 f39e 0000 4854 5450 2f31 2e30        P.......HTTP/1.0
0x0030   2033 3032 2046 6f75 6e64 200d 0a4c 6f63        .302.Found...Loc
0x0040   6174 696f 6e3a 2068 7474 7073 3a2f 2f77        ation:.https://w
0x0050   7777 322e 7465 7374 2e63 6f6d 2f73 706f        ww2.test.com/spo
0x0060   7274 732f 0d0a 0d0a                            rts/....
```

SSL Configuration Examples

This appendix contains these sections:

CSM-S Configuration Example (Bridge Mode, No NAT)

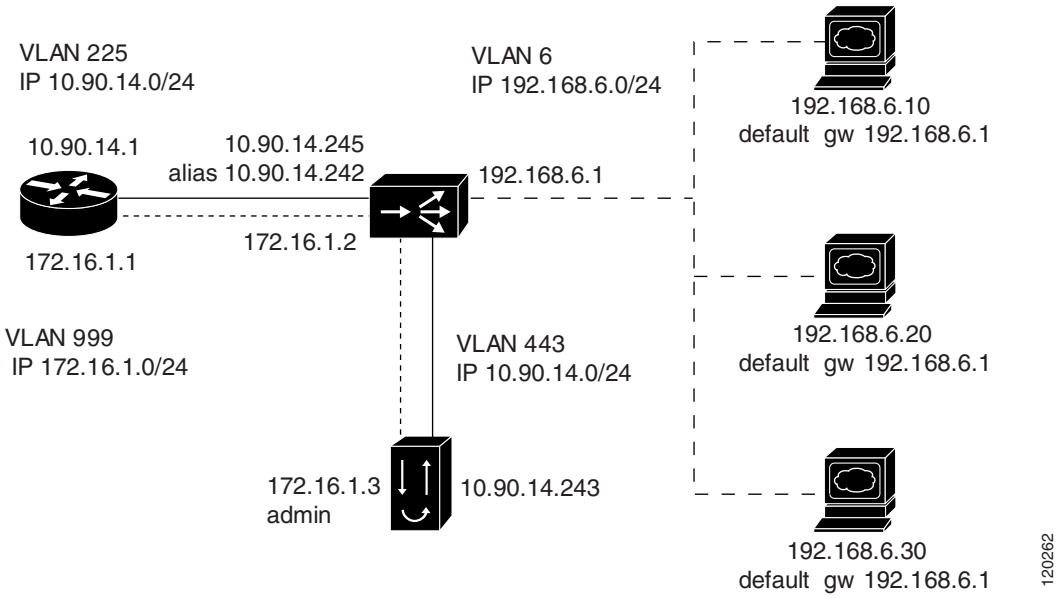
This section describes a CSM-S configuration, which allows a client to load balance HTTP to three web servers (IP addresses 192.168.6.10, 192.168.6.20, and 192.168.6.30) and offload HTTPS, and then load balance to the same three web servers.

In this example, the CSM-S client VLAN and the server VLAN for the SSL daughter card are configured in the same IP subnet (bridge mode), while the web servers are on a private IP network and reside in a separate VLAN. (See Figure B-1.)

The CSM-S is configured so that it does not perform NAT operations when it is directing encrypted traffic to the SSL daughter card. The SSL daughter card is also configured not to perform NAT operations when it is sending decrypted traffic back to the CSM-S for load balancing the decrypted traffic. The CSM-S is then configured to perform NAT for the decrypted traffic to the selected destination server.

The administration network is separate from the client traffic networks and must reside in its own administration VLAN, which must be configured on both the CSM-S and SSL daughter card.

Figure B-1 Bridge Mode, No NAT Configuration Example



The following addresses are configured on the CSM-S:

- Client clear text traffic—10.90.14.181:80
- Client SSL traffic—10.90.14.181:443
- Decrypted traffic from SSL daughter card—10.90.14.181:80
- Client VLAN 225 with IP address 10.90.14.245 for client communication
- Server VLAN 443 with IP address 10.90.14.245, and alias 10.90.14.242 for SSL daughter card communication
- Server VLAN 999 with IP address 172.16.1.2 to allow Administrative communication to reach the SSL daughter card
- Server VLAN 6 with IP address 192.168.6.2, and an alias 192.168.6.1 for real server communications.

The following address is configured on the SSL daughter card:

- 10.90.14.181:443 (This IP address is configured with the **secondary** keyword, which is a CSM-S and bridge mode requirement.)
- VLAN 443 with IP address 10.90.14.243 and a gateway of 10.90.14.1.
- VLAN 999with IP address 172.16.1.3, a gateway of 172.16.1.1, and admin enabled.

Figure B-1 shows VLAN 225and VLAN 443 in the same subnet and VLAN 6 in a separate subnet.

Add all the VLANs (listed above) to the VLAN database, and configure the IP address on the VLAN interface for VLAN 999, VLAN 225, and VLAN 6 on the MSFC.

Note While VLAN 999 (172.16.1.1) and VLAN 225 (10.90.14.1) exist as Layer 3 interfaces on the MSFC. VLAN 443 and VLAN 6 (192.168.6.1) exist as VLANs in the VLAN database, but they do not have corresponding Layer 3 interfaces on the MSFC.

This example shows how to create the Layer 2 and Layer 3 VLANs on the switch MSFC:

```
Cat6k# configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
Cat6k(config)# vlan 6
Cat6k(config-vlan)# name Server_communications
Cat6k(config-if)# vlan 225
Cat6k(config-vlan)# name Client_communications
Cat6k(config-vlan)# interface Vlan225
Cat6k(config-if)# ip address 10.90.14.1 255.255.255.0
Cat6k(config-if)# no shutdown
Cat6k(config-if)# vlan 443
Cat6k(config-vlan)# name SSL-DC_communications
Cat6k(config-if)# vlan 999
Cat6k(config-vlan)# name SSL-DC_administrative
Cat6k(config-vlan)# interface Vlan999
Cat6k(config-if)# ip address 172.16.1.1 255.255.255.0
Cat6k(config-if)# no shutdown
```

This example shows how to create the client and server VLANs on the CSM-S installed in slot number 5:

```
Cat6k# configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
Cat6k(config-module-csm)# module ContentSwitchingModule 5
Cat6k(config-module-csm)# vlan 999 server
Cat6k(config-slb-vlan-server)# ip address 172.16.1.2 255.255.255.0
Cat6k(config-slb-vlan-server)# vlan 225 client
Cat6k(config-slb-vlan-client)# description Client Traffic
Cat6k(config-slb-vlan-client)# ip address 10.90.14.245 255.255.255.0
Cat6k(config-slb-vlan-client)# gateway 10.90.14.1
Cat6k(config-slb-vlan-client)# !
Cat6k(config-slb-vlan-client)# vlan 6 server
Cat6k(config-slb-vlan-server)# description Server Traffic
Cat6k(config-slb-vlan-server)# ip address 192.168.6.2 255.255.255.0
Cat6k(config-slb-vlan-server)# alias 192.168.6.1 255.255.255.0
Cat6k(config-slb-vlan-server)# !
Cat6k(config-slb-vlan-server)# vlan 443 server
Cat6k(config-slb-vlan-server)# ip address 10.90.14.245 255.255.255.0
```

This example shows how to create real servers with names.

```
Cat6k(config-slb-vlan-server)# real LINUX
Cat6k(config-slb-module-real)# address 192.168.6.10
Cat6k(config-slb-module-real)# inservice
Cat6k(config-slb-module-real)# real WIN2K
Cat6k(config-slb-module-real)# address 192.168.6.20
Cat6k(config-slb-module-real)# inservice
Cat6k(config-slb-module-real)# real SUN
Cat6k(config-slb-module-real)# address 192.168.6.30
Cat6k(config-slb-module-real)# inservice
```

This example shows how to create the server farm of web servers (configured with server NAT) and the server farm of the SSL daughter card (configured with no server NAT and local):

```
Cat6k(config-slb-module-real)# serverfarm SSLOFFLOADERS
Cat6k(config-slb-sfarm)# no nat server
Cat6k(config-slb-sfarm)# real 10.90.14.243 local
```

Note The keyword local is required to configure the CSM-S to send traffic to this real over the local VLAN to the SSL daughter card.

```
Cat6k(config-slb-real)# inservice
Cat6k(config-slb-real)# serverfarm WEB
Cat6k(config-slb-sfarm)# real name LINUX
```

```
Cat6k(config-slb-real)# inservice
Cat6k(config-slb-real)# real name WIN2K
Cat6k(config-slb-real)# inservice
Cat6k(config-slb-real)# real name SUN
Cat6k(config-slb-real)# inservice
```

This example shows how to configure the two virtual servers to direct HTTPS traffic to the SSL daughter card for off loading and to load balance HTTP to web servers. In this example, the web servers are receiving traffic to port 80 only, either directly from the clients or as decrypted traffic from the SSL daughter cards (since no port translation is configured).

```
Cat6k(config-slb-module-real)# serverfarm SSLOFFLOADERS
Cat6k(config-slb-sfarm)# no nat server
Cat6k(config-slb-sfarm)# real 10.90.14.243 local
Cat6k(config-slb-real)# inservice
Cat6k(config-slb-real)# serverfarm WEB
Cat6k(config-slb-sfarm)# real name LINUX
Cat6k(config-slb-real)# inservice
Cat6k(config-slb-real)# real name WIN2K
Cat6k(config-slb-real)# inservice
Cat6k(config-slb-real)# real name SUN
Cat6k(config-slb-real)# inservice
```

This example shows how to configure the two virtual servers to direct HTTP traffic to the SSL daughter card for off loading and to load balance HTTP to web servers. In this example, the web servers are receiving traffic to port 80 only, either directly from the clients or as decrypted traffic from the SSL daughter cards (since no port translation is configured).

```
Cat6k(config-slb-real)# vserver SSLTERMINATION
Cat6k(config-slb-vserver)# virtual 10.90.14.181 tcp https
Cat6k(config-slb-vserver)# vlan 225
Cat6k(config-slb-vserver)# serverfarm SSLOFFLOADERS
Cat6k(config-slb-vserver)# persistent rebalance
Cat6k(config-slb-vserver)# inservice
Cat6k(config-slb-vserver)# vserver WEBSERVERS
Cat6k(config-slb-vserver)# virtual 10.90.14.181 tcp www
Cat6k(config-slb-vserver)# serverfarm WEB
Cat6k(config-slb-vserver)# persistent rebalance
Cat6k(config-slb-vserver)# inservice
Cat6k(config-slb-vserver)# exit
Cat6k(config-module-csm)# exit
Cat6k(config)# exit
```

This example shows how to configure the administration VLAN on the SSL daughter card to communicate over the VLAN 999:

```
SSL-DC# configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
SSL-DC(config)# ssl-proxy vlan 999
SSL-DC(config-vlan)# ipaddr 172.16.1.3 255.255.255.0
SSL-DC(config-vlan)# gateway 172.16.1.1
SSL-DC(config-vlan)# admin
```

Next the VLAN 443 is configured to allow communication with clients for off loading client SSL connections:

```
SSL-DC(config-vlan)# ssl-proxy vlan 443
SSL-DC(config-vlan)# ipaddr 10.90.14.243 255.255.255.0
SSL-DC(config-vlan)# gateway 10.90.14.1
```

To complete the configuration, enter the **ssl-proxy service** command to create a new service on the SSL daughter card (**sslterm**). This example shows how to configure a virtual IP address that matches the virtual server created on the CSM-S. (This virtual IP address is configured with the **secondary** keyword

so that the SSL daughter card does not reply to ARP requests for this IP address. This configuration is also a requirement for bridging network designs) The service is configured to send decrypted traffic back to the CSM-S without performing NAT.

```
SSL-DC(config-vlan)# ssl-proxy service sslterm
SSL-DC(config-ssl-proxy)# virtual ipaddr 10.90.14.181 protocol tcp port 443 secondary
SSL-DC(config-ssl-proxy)# server ipaddr 10.90.14.245 protocol tcp port 80
SSL-DC(config-ssl-proxy)# no nat server
SSL-DC(config-ssl-proxy)# certificate rsa general-purposetrustpoint certs-key

*Aug 19 20:52:11.487: %STE-6-PKI_SERVICE_CERT_INSTALL: Proxy: sslterm, Trustpoin
t: certs-key, Key: RSAKEY, Serial#: 1A65, Index: 2
*Aug 19 20:52:11.487: %STE-6-PKI_CA_CERT_INSTALL: Root, Subject Name: CN = Thawt
e Test CA Root, OU = TEST TEST TEST, O = Thawte Certification, ST = FOR TESTING
PURPOSES ONLY, C = ZA, Serial#: 00, Index: 3

SSL-DC(config-ssl-proxy)# inservice

*Aug 19 20:52:11.515: %STE-5-UPDOWN: ssl-proxy service sslterm changed state to
UP
```

These examples show the output of the various **show** commands on the MSFC and CSM:

```
Cat6k# show module csm 5 vlan detail
vlan   IP address       IP mask          type
-------------------------------------------------
6      192.168.6.2      255.255.255.0    SERVER
  Description: Server Traffic
  ALIASES
  IP address       IP mask
  -------------------------------
  192.168.6.1      255.255.255.0
225    10.90.14.245     255.255.255.0    CLIENT
  Description: Client Traffic
  GATEWAYS
  10.90.14.129
443    10.90.14.245     255.255.255.0    SERVER
999    172.16.1.2       255.255.255.0    SERVER

Cat6k# show module csm 5 real

real                  server farm      weight  state          conns/hits
-------------------------------------------------------------------------
10.90.14.243          SSLOFFLOADERS    8       OPERATIONAL    0
LINUX                 WEB              8       OPERATIONAL    0
WIN2K                 WEB              8       OPERATIONAL    0
SUN                   WEB              8       OPERATIONAL    0

Cat6k# show module csm 5 vserver detail
SSLTERMINATION, type = SLB, state = OPERATIONAL, v_index = 12
  virtual = 10.90.14.181/32:443 bidir, TCP, service = NONE, advertise = FALSE
  idle = 3600, replicate csrp = none, vlan = 225, pending = 30, layer 4
  max parse len = 2000, persist rebalance = TRUE
  ssl sticky offset = 0, length = 32
  conns = 1, total conns = 4
  Default policy:
    server farm = SSLOFFLOADERS, backup = <not assigned>
    sticky: timer = 0, subnet = 0.0.0.0, group id = 0
  Policy          Tot matches  Client pkts  Server pkts
  -----------------------------------------------------
  (default)           4            32           21
```

```
WEBSERVERS, type = SLB, state = OPERATIONAL, v_index = 13
  virtual = 10.90.14.181/32:80 bidir, TCP, service = NONE, advertise = FALSE
  idle = 3600, replicate csrp = none, vlan = ALL, pending = 30, layer 4
  max parse len = 2000, persist rebalance = TRUE
  ssl sticky offset = 0, length = 32
  conns = 1, total conns = 7
  Default policy:
    server farm = WEB, backup = <not assigned>
    sticky: timer = 0, subnet = 0.0.0.0, group id = 0
  Policy          Tot matches  Client pkts  Server pkts
  ---------------------------------------------------
  (default)       7            45           35
```

These examples show the output of the various **show** commands on the SSL daughter card:

```
SSL-DC# show ssl-proxy service sslterm
Service id: 1, bound_service_id: 257
Virtual IP: 10.90.14.181, port: 443 (secondary configured)
Server IP: 10.90.14.245, port: 80
rsa-general-purpose certificate trustpoint: certs-key
  Certificate chain for new connections:
    Certificate:
       Key Label: RSAKEY, 1024-bit, not exportable
       Key Timestamp: 02:03:11 UTC Aug 19 2004
       Serial Number: 1A65
    Root CA Certificate:
       Serial Number: 00
  Certificate chain complete
Admin Status: up
Operation Status: up

SSL-DC# show ssl-proxy stats
TCP Statistics:
    Conns initiated     : 4             Conns accepted       : 4
    Conns established   : 8             Conns dropped        : 4
    Conns Allocated     : 4             Conns Deallocated    : 4
    Conns closed        : 8             SYN timeouts         : 0
    Idle timeouts       : 0             Total pkts sent      : 43
    Data packets sent   : 19            Data bytes sent      : 5875
    Total Pkts rcvd     : 48            Pkts rcvd in seq     : 21
    Bytes rcvd in seq   : 3264

SSL Statistics:
    conns attempted     : 4             conns completed     : 4
    full handshakes     : 2             resumed handshakes  : 2
    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             session fails       : 0

FDU Statistics:
    IP Frag Drops       : 0             IP Version Drops    : 0
    IP Addr Discards    : 0             Serv_Id Drops       : 0
    Conn Id Drops       : 0             Bound Conn Drops    : 0
    Vlan Id Drops       : 0             TCP Checksum Drops  : 0
    Hash Full Drops     : 0             Hash Alloc Fails    : 0
    Flow Creates        : 8             Flow Deletes        : 8
    Conn Id allocs      : 4             Conn Id deallocs    : 4
    Tagged Pkts Drops   : 0             Non-Tagg Pkts Drops : 0
    Add ipcs            : 3             Delete ipcs         : 0
    Disable ipcs        : 2             Enable ipcs         : 0
```

```
Unsolicited ipcs     : 127          Duplicate Add ipcs  : 0
IOS Broadcast Pkts   : 613          IOS Unicast Pkts    : 1110
IOS Multicast Pkts   : 0            IOS Total Pkts      : 1723
IOS Congest Drops    : 0            SYN Discards        : 0
```

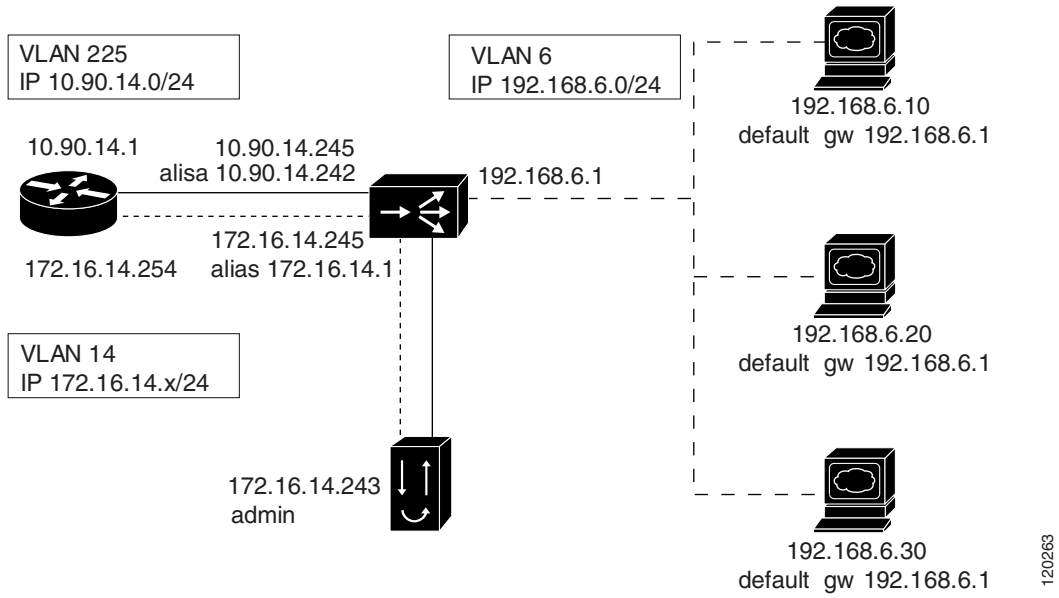
CSM-S Configuration Example (Router Mode, Server NAT)

This section describes a CSM-S configuration which allows a client to load balance HTTP to three web servers (IP addresses 192.168.6.10, 192.168.6.20, and 192.168.6.30) and offload HTTPS then load balance to the same three web servers.

In this example, the CSM-S client VLAN is on a public network, the server VLAN for the SSL daughter card is in the a private IP subnet, and the web servers are in a different private IP network and reside in a separate VLAN. (See Figure B-2.)

The CSM-S is configured to perform the default server NAT operations to direct encrypted client traffic to the SSL daughter card. The SSL daughter card is also configured to perform server NAT operations when sending decrypted traffic back to the CSM-S. The CSM-S is then configured to perform another NAT on the decrypted traffic to the selected destination server.

Figure B-2 Configuration Example—Router Mode, Server NAT



The following addresses are configured on the CSM-S virtual servers:

- Client clear text traffic—10.90.14.182:80
- Client SSL traffic—10.90.14.182:443
- Decrypted traffic from SSL daughter card—10.90.14.182:80
- Client VLAN 225 with IP address 10.90.14.245 for client communication
- Server VLAN 14 with IP address 172.16.14.245, and alias 172.16.14.1 for SSL daughter card communication

- Server VLAN 6 with IP address 192.168.6.2, and an alias 192.168.6.1 for real server communications

The following address is configured on the SSL daughter card:

- 172.16.14.182:443 (this IP address is configured with the **secondary** keyword a CSM-S requirement)
- VLAN 14 with IP address 172.16.14.243, a route for client traffic to the CSM-S VLAN interface, and a gateway of 172.16.14.254 for routing administrative traffic.

Figure B-2, shows VLAN 225, VLAN 14 and VLAN 6 are each in separate subnets.

Add all the VLANs (listed above) to the VLAN database, and configure the IP address on the VLAN interface for VLAN 14 and VLAN 225 on the MSFC.

Note VLAN 225 (10.90.14.1) exists as a Layer 3 interface on the MSFC to route Client traffic to the CSM-S. VLAN 14 (172.16.1.254) is also configured on the MSFC to allow administrative traffic to be routed to the SSL daughter card. VLAN 14 (172.16.14.1) is configured on the CSM-S to send and received SSL traffic to/from the SSL daughter card. VLAN 6 (192.168.6.1) exists only as a VLAN in the VLAN database and as CSM-S and SSL daughter card VLANs, but it does not have corresponding Layer 3 interfaces on the MSFC.

This example creates the Layer 2 and Layer 3 VLANs on the switch MSFC:

```
Cat6k# configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
Cat6k(config)# vlan 6
Cat6k(config-vlan)# name Server_communications
Cat6k(config)# vlan 14
Cat6k(config-vlan)# name SSL-DC_communications
Cat6k(config-vlan)# interface Vlan14
Cat6k(config-if)# ip address 172.16.14.254 255.255.255.0
Cat6k(config-if)# no shutdown
Cat6k(config-if)# vlan 225
Cat6k(config-vlan)# name Client_communications
Cat6k(config-vlan)# interface Vlan225
Cat6k(config-if)# ip address 10.90.14.1 255.255.255.0
Cat6k(config-if)# no shutdown
```

This example shows how to create the client and server VLANs on the CSM installed in slot number 5:

```
Cat6k# configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
Cat6k(config)# module ContentSwitchingModule 5
Cat6k(config-module-csm)# vlan 225 client
Cat6k(config-slb-vlan-client)# description Client Traffic
Cat6k(config-slb-vlan-client)# ip address 10.90.14.245 255.255.255.0
Cat6k(config-slb-vlan-client)# gateway 10.90.14.1
Cat6k(config-slb-vlan-client)# vlan 6 server
Cat6k(config-slb-vlan-server)# description Server Traffic
Cat6k(config-slb-vlan-server)# ip address 192.168.6.2 255.255.255.0
Cat6k(config-slb-vlan-server)# alias 192.168.6.1 255.255.255.0
Cat6k(config-slb-vlan-server)# vlan 14 server
Cat6k(config-slb-vlan-server)# ip address 172.16.14.245 255.255.255.0
Cat6k(config-slb-vlan-server)# alias 172.16.14.1 255.255.255.0
```

This example shows how to create real servers with names.

```
Cat6k(config-slb-vlan-server)# real LINUX
Cat6k(config-slb-module-real)# address 192.168.6.10
Cat6k(config-slb-module-real)# inservice
Cat6k(config-slb-module-real)# real WIN2K
```

```
Cat6k(config-slb-module-real)# address 192.168.6.20
Cat6k(config-slb-module-real)# inservice
Cat6k(config-slb-module-real)# real SUN
Cat6k(config-slb-module-real)# address 192.168.6.30
Cat6k(config-slb-module-real)# inservice
```

This example shows how to create the server farm of web servers (configured with server NAT) and the server farm of the SSL daughter card (configured with server NAT and local):

```
Cat6k(config-slb-module-real)# serverfarm SSLOFFLOADERS
Cat6k(config-slb-sfarm)# nat server
Cat6k(config-slb-sfarm)# no nat client
Cat6k(config-slb-sfarm)# real 172.16.14.182 local
```

Note The keyword local is required to configure the CSM-S to send traffic to this real over the local VLAN to the SSL daughter card.

```
Cat6k(config-slb-real)# inservice
Cat6k(config-slb-real)# serverfarm WEB
Cat6k(config-slb-sfarm)# nat server
Cat6k(config-slb-sfarm)# no nat client
Cat6k(config-slb-sfarm)# real name LINUX
Cat6k(config-slb-real)# inservice
Cat6k(config-slb-real)# real name WIN2K
Cat6k(config-slb-real)# inservice
Cat6k(config-slb-real)# real name SUN
Cat6k(config-slb-real)# inservice
```

This example shows how to configure the two virtual servers. In this example, the web servers receive requests to port 80 directly from the clients. HTTPS traffic is received on port 443 and sent to the SSL daughter card for decryption. Upon decryption, the HTTP traffic is sent to the public HTTP virtual for load balancing:

```
Cat6k(config-slb-real)# vserver SSLTERMINATION
Cat6k(config-slb-vserver)# virtual 10.90.14.182 tcp https
Cat6k(config-slb-vserver)# serverfarm SSLOFFLOADERS
Cat6k(config-slb-vserver)# persistent rebalance
Cat6k(config-slb-vserver)# inservice

Cat6k(config-slb-vserver)# vserver WEBSERVERS
Cat6k(config-slb-vserver)# virtual 10.90.14.182 tcp www
Cat6k(config-slb-vserver)# serverfarm WEB
Cat6k(config-slb-vserver)# persistent rebalance
Cat6k(config-slb-vserver)# inservice
```

This example shows how to configure the SSL daughter card to communicate with the CSM-S over VLAN 14 for client traffic and administrative traffic:

```
SSL-DC# configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
SSL-DC(config)# ssl-proxy vlan 14
SSL-DC(config-vlan)# ipaddr 172.16.14.243 255.255.255.0
SSL-DC(config-vlan)# gateway 172.16.14.254
SSL-DC(config-vlan)# route 10.90.14.0 255.255.255.0 gateway 172.16.14.1
SSL-DC(config-vlan)# admin
```

Note The **gateway** command is required for routing administrative communication. The router defined only effects traffic destined for the VLAN IP address (TELNET, SSH, and so on...)
The **route** statement is required to route traffic back to the CSM-S alias IP when the SSL daughter card is receiving encrypted traffic on one network and sending decrypted traffic to a different IP network.

Note The administrative VLAN can be configured separately if required by adding a new VLAN and the appropriate IP address on both the CSM-S and SSL daughter card.

To complete the configuration, enter the **ssl-proxy service** command to create a new service on the SSL daughter card (**sslterm**). This example shows how to configure a virtual IP address that matches the virtual server created on the CSM-S. (This virtual IP address is configured with the **secondary** keyword so that the SSL daughter card does not reply to ARP requests for this IP address. This configuration is also a requirement for bridging network designs.) The service is configured to send decrypted traffic back to the CSM-S while performing NAT on the destination address:

```
SSL-DC(config-vlan)# ssl-proxy service sslterm
SSL-DC(config-ssl-proxy)# virtual 172.16.14.182 protocol tcp port 443 secondary
SSL-DC(config-ssl-proxy)# server ipaddr 10.90.14.182 protocol tcp port 80
SSL-DC(config-ssl-proxy)# certificate rsa general-purpose trustpoint certs-key

*Aug 22 14:44:47.395: %STE-6-PKI_SERVICE_CERT_INSTALL: Proxy: sslterm, Trustpoin
t: certs-key, Key: RSAKEY, Serial#: 1A65, Index: 6
*Aug 22 14:44:47.395: %STE-6-PKI_CA_CERT_INSTALL: Root, Subject Name: CN = Thawt
e Test CA Root, OU = TEST TEST TEST, O = Thawte Certification, ST = FOR TESTING
PURPOSES ONLY, C = ZA, Serial#: 00, Index: 7

SSL-DC(config-ssl-proxy)# inservice

*Aug 22 14:44:47.423: %STE-5-UPDOWN: ssl-proxy service sslterm changed state to UP

Cat6k # configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
Cat6k(config)# interface Vlan14
Cat6k(config-if)# ip address 172.16.14.254 255.255.255.0
Cat6k(config-if)# no shutdown
```

These examples show the output of the various **show** commands on the MSFC and CSM:

```
cat6k# show mod csm 5 vlan detail
vlan   IP address       IP mask          type
-----------------------------------------------------
6      192.168.6.2      255.255.255.0    SERVER
  Description: Server Traffic
  ALIASES
  IP address       IP mask
  -------------------------------
  192.168.6.1      255.255.255.0
14     172.16.14.245    255.255.255.0    SERVER
  ALIASES
  IP address       IP mask
  -------------------------------
  172.16.14.1      255.255.255.0
225    10.90.14.245     255.255.255.0    CLIENT
  Description: Client Traffic
  GATEWAYS
  10.90.14.129

Cat6k# show mod csm 5 real
real                 server farm      weight  state          conns/hits
-------------------------------------------------------------------------
172.16.14.182        SSLOFFLOADERS    8       OPERATIONAL    0
LINUX                WEB              8       OPERATIONAL    0
WIN2K                WEB              8       OPERATIONAL    0
SUN                  WEB              8       OPERATIONAL    0
```

```
Cat6k# show mod csm 5 vserver detail
SSLTERMINATION, type = SLB, state = OPERATIONAL, v_index = 20
  virtual = 10.90.14.182/32:443 bidir, TCP, service = NONE, advertise = FALSE
  idle = 3600, replicate csrp = none, vlan = ALL, pending = 30, layer 4
  max parse len = 2000, persist rebalance = TRUE
  ssl sticky offset = 0, length = 32
  conns = 0, total conns = 8
  Default policy:
    server farm = SSLOFFLOADERS, backup = <not assigned>
    sticky: timer = 0, subnet = 0.0.0.0, group id = 0
  Policy          Tot matches  Client pkts  Server pkts
  -----------------------------------------------------
  (default)       8            75           46

WEBSERVERS, type = SLB, state = OPERATIONAL, v_index = 21
  virtual = 10.90.14.182/32:80 bidir, TCP, service = NONE, advertise = FALSE
  idle = 3600, replicate csrp = none, vlan = ALL, pending = 30, layer 4
  max parse len = 2000, persist rebalance = TRUE
  ssl sticky offset = 0, length = 32
  conns = 0, total conns = 11
  Default policy:
    server farm = WEB, backup = <not assigned>
    sticky: timer = 0, subnet = 0.0.0.0, group id = 0
  Policy          Tot matches  Client pkts  Server pkts
  -----------------------------------------------------
  (default)       11           58           38
```

These examples show the output of the various **show** commands on the SSL daughter card:

```
SSL-DC# show ssl-proxy service sslterm
Service id: 4, bound_service_id: 260
Virtual IP: 172.16.14.182, port: 443 (secondary configured)
Server IP: 10.90.14.182, port: 80
rsa-general-purpose certificate trustpoint: certs-key
  Certificate chain in graceful rollover, being renewed:
    Certificate:
       Key Label: RSAKEY, 1024-bit, not exportable
       Key Timestamp: 02:03:11 UTC Aug 19 2004
       Serial Number: 1A65
    Root CA Certificate:
       Serial Number: 00
  Service certificate in graceful rollover
Admin Status: up
Operation Status: up

SSL-DC# show ssl-proxy stats
TCP Statistics:
    Conns initiated     : 12            Conns accepted       : 12
    Conns established   : 24            Conns dropped        : 12
    Conns Allocated     : 12            Conns Deallocated    : 12
    Conns closed        : 24            SYN timeouts         : 0
    Idle timeouts       : 0             Total pkts sent      : 129
    Data packets sent   : 59            Data bytes sent      : 23001
    Total Pkts rcvd     : 146           Pkts rcvd in seq     : 57
    Bytes rcvd in seq   : 9826

SSL Statistics:
    conns attempted     : 12            conns completed     : 12
    full handshakes     : 10            resumed handshakes  : 2
    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             session fails       : 0

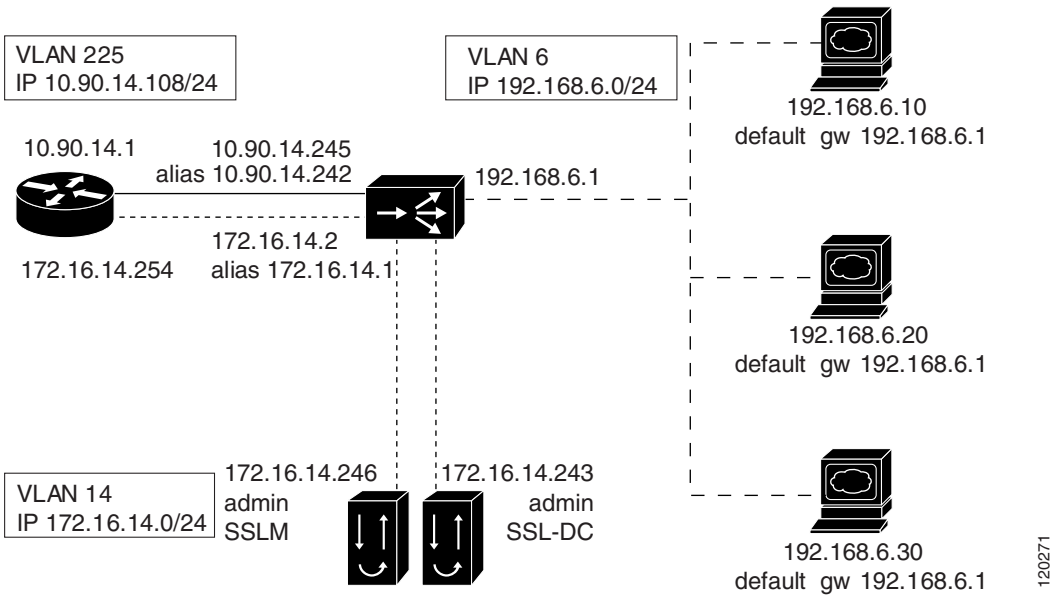
FDU Statistics:
    IP Frag Drops       : 0             IP Version Drops    : 0
    IP Addr Discards    : 0             Serv_Id Drops       : 2
    Conn Id Drops       : 0             Bound Conn Drops    : 0
    Vlan Id Drops       : 0             TCP Checksum Drops  : 0
    Hash Full Drops     : 0             Hash Alloc Fails    : 0
    Flow Creates        : 24            Flow Deletes        : 24
    Conn Id allocs      : 12            Conn Id deallocs    : 12
    Tagged Pkts Drops   : 0             Non-Tagg Pkts Drops : 0
    Add ipcs            : 7             Delete ipcs         : 0
    Disable ipcs        : 6             Enable ipcs         : 0
    Unsolicited ipcs    : 3579          Duplicate Add ipcs  : 0
    IOS Broadcast Pkts  : 17881         IOS Unicast Pkts    : 31780
    IOS Multicast Pkts  : 0             IOS Total Pkts      : 49661
    IOS Congest Drops   : 0             SYN Discards        : 0
```

CSM-S and SSLM Configuration Example (Router Mode, Server NAT)

This section scales the previous CSM-S configuration Appendix B, "CSM-S Configuration Example (Router Mode, Server NAT)" by adding a SSL Services Module (SSLM) to the design. The SSLM is added using the same VLAN and IP network as the SSL daughter card. The CMS-S will use weighted round robin to load balance traffic between the SSLM and SSL-DC. Since the SSLM is approximately three times faster than the SSL daughter card weighted round robin is needed to spread the traffic across SSL off loaders according to the performance of the SSL off loader. The CSM-S applies SSL sticky to the client connections to ensure the same SSL session continue to use the same SSL off loader for the duration of the SSL session. In this example the duration is thirty minutes.

In this example, the SSLM is added to the previous CSM-S configuration. The SSLM will accept client connections on the same network as the SSL daughter card. (See Figure B-3.)

Figure B-3 Configuration Example—CSM-S and SSLM Router Mode, Server NAT



The following address is configured on the SSLM:

- ssl-proxy service virtual of 172.16.14.10:443 and a server IP of 10.90.14.182:80
- VLAN 14 with IP address 172.16.14.246, a route for client traffic to the CSM-S VLAN interface, and a gateway of 172.16.14.254 for routing administrative traffic.

Along with he SSLM configuration the MSFC must be configured to allow VLAN 14 traffic to pass to the SSLM.

This example creates the Layer 2 and Layer 3 VLANs on the Cat6k MSFC:

```
Cat6k# configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
Cat6k(config)# ssl-proxy module 6 allowed-vlan 14
```

This example shows how to add the SSLM to the CSM-S.

```
Cat6k(config-slb-vlan-server)# real SSLM
Cat6k(config-slb-module-real)# address 172.16.14.10
Cat6k(config-slb-module-real)# inservice
Cat6k(config-slb-module-real)# inservice
```

This example shows how to add the SSLM real to the server farm of SSL off loaders and configure the weights for each real:

```
Cat6k(config-slb-module-real)# serverfarm SSLOFFLOADERS
Cat6k(config-slb-sfarm)# real 172.16.14.182 local
Cat6k(config-slb-sfarm)# weight 1
Cat6k(config-slb-real)# inservice
Cat6k(config-slb-sfarm)# real name SSLM
Cat6k(config-slb-sfarm)# weight 3
Cat6k(config-slb-real)# inservice
```

This example shows how to configure the CSM-S virtual server to apply SSL sticky for thirty minute sessions and use an SSL Session ID offset (SSL sticky sticky 10 ssl timeout 30).

```
Cat6k(config-slb-real)# vserver SSLTERMINATION
Cat6k(config-slb-vserver)# sticky 30 group 10
Cat6k(config-slb-vserver)# ssl-sticky offset 20 length 6
```

This example shows how to configure the SSLM to communicate with the CSM-S over VLAN 14 for client traffic and administrative traffic:

```
SSLM# configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
SSLM(config)# ssl-proxy vlan 14
SSLM(config-vlan)# ipaddr 172.16.14.246 255.255.255.0
SSLM(config-vlan)# gateway 172.16.14.254
SSLM(config-vlan)# route 10.90.14.0 255.255.255.0 gateway 172.16.14.1
SSLM(config-vlan)# admin
```

Note The **gateway** command is required for routing administrative communication. The router defined only effects traffic destined for the VLAN IP address (TELNET, SSH, and so on...)
The **route** statement is required to route traffic back to the CSM-S alias IP when the SSL daughter card is receiving encrypted traffic on one network and sending decrypted traffic to a different Ip network.

To complete the configuration, enter the **ssl-proxy service** command to create a new service on the SSL daughter card (**sslterm**). This example shows how to configure a virtual IP address that matches the virtual server created on the CSM-S. (This virtual IP address is configured with the **secondary** keyword so that the SSL daughter card does not reply to ARP requests for this IP address. This is also a requirement for bridging network designs) The service is configured to send decrypted traffic back to the CSM-S while performing NAT on the destination address:

```
SSLM(config)# ssl-proxy service sslterm
SSLM(config-ssl-proxy)# virtual ipaddr 172.16.14.10 protocol tcp port 443
SSLM(config-ssl-proxy)# server ipaddr 10.90.14.182 protocol tcp port 80
SSLM(config-ssl-proxy)# certificate rsa general-purpose trustpoint certs-key

*Aug 24 01:40:17.581: %STE-6-PKI_SERVICE_CERT_INSTALL: Proxy: sslterm, Trustpoint:
certs-key, Key: RSAKEY, Serial#: 1C2B, Index: 2
*Aug 24 01:40:27.637: %STE-6-PKI_SERVICE_CERT_DELETE: Proxy: , Trustpoint: certs-key, Key:
RSAKEY, Serial#: 1C2B, Index: 0

SSLM(config-ssl-proxy)# inservice

*Aug 24 01:40:34.165: %STE-5-UPDOWN: ssl-proxy service sslterm changed state to UP

SSLM(config-ssl-proxy)# exit

Cat6k# show mod csm 5 real

real                  server farm      weight  state          conns/hits
--------------------------------------------------------------------------
SSLM                  SSLOFFLOADERS    3       OPERATIONAL    0
172.16.14.182         SSLOFFLOADERS    1       OPERATIONAL    0
LINUX                 WEB              8       OPERATIONAL    0
SUN                   WEB              8       OPERATIONAL    0
WIN2K                 WEB              8       OPERATIONAL    0

Cat6k# show mod csm 5 vserver detail
SSLTERMINATION, type = SLB, state = OPERATIONAL, v_index = 22
  virtual = 10.90.14.182/32:443 bidir, TCP, service = NONE, advertise = FALSE
  idle = 3600, replicate csrp = none, vlan = ALL, pending = 30, layer 7
  max parse len = 2000, persist rebalance = TRUE
  ssl sticky offset = 20, length = 6
  conns = 0, total conns = 12
  Default policy:
    server farm = SSLOFFLOADERS, backup = <not assigned>
```

```
    sticky: timer = 30, subnet = 0.0.0.0, group id = 10
  Policy          Tot matches  Client pkts  Server pkts
  ------------------------------------------------------
  (default)       12           135          96

WEBSERVERS, type = SLB, state = OPERATIONAL, v_index = 23
  virtual = 10.90.14.182/32:80 bidir, TCP, service = NONE, advertise = FALSE
  idle = 3600, replicate csrp = none, vlan = ALL, pending = 30, layer 4
  max parse len = 2000, persist rebalance = TRUE
  ssl sticky offset = 0, length = 32
  conns = 0, total conns = 12
  Default policy:
    server farm = WEB, backup = <not assigned>
    sticky: timer = 0, subnet = 0.0.0.0, group id = 0
  Policy          Tot matches  Client pkts  Server pkts
  ------------------------------------------------------
  (default)       12           75           67

Cat6k# show mod csm 5 sticky

group   sticky-data              real                  timeout
-----------------------------------------------------------------
10      ssl ADB70000:000DBCAF    172.16.14.182         1680
10      ssl A03F0000:00602F30    172.16.14.10          1596
```

These examples show the output of the various **show** commands on the SSLM:

```
SSLM# show ssl-proxy service sslterm
Service id: 1, bound_service_id: 257
Virtual IP: 172.16.14.10, port: 443
Server IP: 10.90.14.182, port: 80
rsa-general-purpose certificate trustpoint: certs-key
  Certificate chain for new connections:
    Certificate:
       Key Label: RSAKEY, 1024-bit, exportable
       Key Timestamp: 13:12:48 UTC Aug 23 2004
       Serial Number: 1C2B
    Root CA Certificate:
       Serial Number: 00
  Certificate chain complete
Admin Status: up
Operation Status: up

SSLM# show ssl-proxy stats
TCP Statistics:
    Conns initiated     : 14            Conns accepted       : 14
    Conns established   : 28            Conns dropped        : 10
    Conns Allocated     : 14            Conns Deallocated    : 14
    Conns closed        : 28            SYN timeouts         : 0
    Idle timeouts       : 0             Total pkts sent      : 181
    Data packets sent   : 90            Data bytes sent      : 47214
    Total Pkts rcvd     : 196           Pkts rcvd in seq     : 85
    Bytes rcvd in seq   : 32480
SSL Statistics:
    conns attempted     : 14            conns completed     : 14
    full handshakes     : 11            resumed handshakes  : 3
    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             session fails       : 0
FDU Statistics:
```

```
IP Frag Drops        : 0              IP Version Drops     : 0
IP Addr Discards     : 0              Serv_Id Drops        : 0
Conn Id Drops        : 0              Bound Conn Drops     : 0
Vlan Id Drops        : 0              TCP Checksum Drops   : 1
Hash Full Drops      : 0              Hash Alloc Fails     : 0
Flow Creates         : 28             Flow Deletes         : 28
Conn Id allocs       : 14             Conn Id deallocs     : 14
Tagged Pkts Drops    : 0              Non-Tagg Pkts Drops  : 0
Add ipcs             : 2              Delete ipcs          : 0
Disable ipcs         : 1              Enable ipcs          : 0
Unsolicited ipcs     : 0              Duplicate Add ipcs   : 0
IOS Broadcast Pkts   : 68857          IOS Unicast Pkts     : 293
IOS Multicast Pkts   : 0              IOS Total Pkts       : 69150
IOS Congest Drops    : 0              SYN Discards         : 0
```

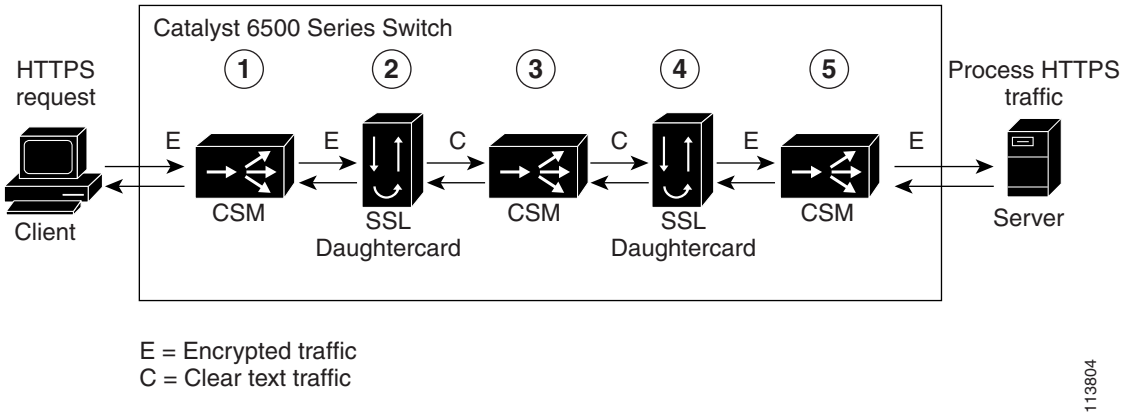
Integrated Secure Content-Switching Service Example

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 B-4, 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 "Configuring Session Persistence (Stickiness)" section on page 10-1 for information on configuring sticky connections, to an SSL daughter card.
2. The SSL daughter card 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 daughter card again for encryption to SSL traffic.
4. The SSL daughter card forwards the encrypted SSL traffic to the CSM.
5. The CSM forwards the SSL traffic to the HTTPS server.

Figure B-4 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 daughter card and the CSM.

```
Router# confure terminal
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 local
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
```

Configuring the SSL Daughter Card

This example shows how to create the VLAN between the SSL daughter card and the CSM:

```
ssl-proxy(config)# ssl-proxy vlan 35
ssl-proxy(config-vlan)# ipaddr 35.200.200.3 255.0.0.0
ssl-proxy(config-vlan)# gateway 35.200.200.100
ssl-proxy(config-vlan)# admin
```

This example shows how to configure a trusted certificate authority pool on the SSL daughter card:

```
ssl-proxy(config-vlan)# ssl-proxy pool ca net
ssl-proxy(config-ca-pool)# ca trustpoint keon-root
ssl-proxy(config-ca-pool)# ca trustpoint net-root
ssl-proxy(config-ca-pool)# ca trustpoint TP-1024-pcks12-root
```

This example shows how to configure a URL rewrite policy on the SSL daughter card:

```
ssl-proxy(config)# ssl-proxy policy url-rewrite frontend
ss(config-url-rewrite-policy)# url www.cisco.com clearport 80 sslport 443
ss(config-url-rewrite-policy)# url wwwin.cisco.com clearport 80 sslport 443
ss(config-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 daughter card directly opens a connection to the configured server.

```
ssl-proxy(config-ca-pool)# ssl-proxy service frontend
ssl-proxy(config-ssl-proxy)# virtual ipaddr 35.200.200.101 protocol tcp port 443 secondary
ssl-proxy(config-ssl-proxy)# server ipaddr 35.35.35.25 protocol tcp port 81
ssl-proxy(config-ssl-proxy)# server ipaddr 35.200.200.14 protocol tcp port 443 sslv2
ssl-proxy(config-ssl-proxy)# certificate rsa general-purpose trustpoint TP-1024-pkcs12
ssl-proxy(config-ssl-proxy)# policy url-rewrite frontend
ssl-proxy(config-ssl-proxy)# trusted-ca net
ssl-proxy(config-ssl-proxy)# authenticate verify all
ssl-proxy(config-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-ssl-proxy)# ssl-proxy service wildcard client
ssl-proxy(config-ssl-proxy)# virtual ipaddr 0.0.0.0 0.0.0.0 protocol tcp port 81 secondary
ssl-proxy(config-ssl-proxy)# server ipaddr 35.200.200.125 protocol tcp port 443
ssl-proxy(config-ssl-proxy)# certificate rsa general-purpose trustpoint client-cert
ssl-proxy(config-ssl-proxy)# no nat server
ssl-proxy(config-ssl-proxy)# trusted-ca net
ssl-proxy(config-ssl-proxy)# authenticate verify all
ssl-proxy(config-ssl-proxy)# inservice
ssl-proxy(config-ssl-proxy)# ^Z
```

This example shows how to display the status of the SSL server proxy service:

```
ssl-proxy# show ssl-proxy service frontend
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
Admin Status: up
Operation Status: up
ssl-proxy#
```

This example shows how to display the status of the SSL client proxy service:

```
ssl-proxy# show ssl-proxy service wildcard
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
Admin Status: up
Operation Status: up
ssl-proxy#
```

Certificate Security Attribute-Based Access Control Examples

Certificate security attribute-based access control 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

This example shows that the 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 service ssl-offload
ssl-proxy(config-ssl-proxy)# virtual ipaddr 8.100.100.126 protocol tcp port 443 secondary
```

```
ssl-proxy(config-ssl-proxy)# server ipaddr 191.162.2.8 protocol tcp port 80
ssl-proxy(config-ssl-proxy)# certificate rsa general-purpose trustpoint cert
ssl-proxy(config-ssl-proxy)# nat client client-nat
ssl-proxy(config-ssl-proxy)# trusted-ca root-ca
ssl-proxy(config-ssl-proxy)# authenticate verify all
ssl-proxy(config-ssl-proxy)# inservice
ssl-proxy(config-ssl-proxy)# exit
ssl-proxy(config)#
ssl-proxy(config)# 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 ca trustpoint root
ssl-proxy(ca-trustpoint)# enrollment mode ra
ssl-proxy(ca-trustpoint)# enrollment terminal
ssl-proxy(ca-trustpoint)# crl optional
ssl-proxy(ca-trustpoint)# match certificate acl
ssl-proxy(ca-trustpoint)# exit
ssl-proxy(config)#
ssl-proxy(config)# crypto ca certificate map acl 10
ssl-proxy(ca-certificate-map)# subject-name co .cisco.com
ssl-proxy(ca-certificate-map)# exit
```

This example shows that the 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 the string “certificate manager.”

```
ssl-proxy(config)# ssl-proxy service ssl-offload
ssl-proxy(config-ssl-proxy)# virtual ipaddr 8.100.100.126 protocol tcp port 443 secondary
ssl-proxy(config-ssl-proxy)# server ipaddr 191.162.2.8 protocol tcp port 80
ssl-proxy(config-ssl-proxy)# certificate rsa general-purpose trustpoint cert
ssl-proxy(config-ssl-proxy)# nat client client-nat
ssl-proxy(config-ssl-proxy)# trusted-ca root-ca
ssl-proxy(config-ssl-proxy)# authenticate verify all
ssl-proxy(config-ssl-proxy)# inservice
ssl-proxy(config-ssl-proxy)# exit
ssl-proxy(config)#
ssl-proxy(config)# 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 ca trustpoint root
ssl-proxy(ca-trustpoint)# enrollment mode ra
ssl-proxy(ca-trustpoint)# enrollment terminal
ssl-proxy(ca-trustpoint)# crl optional
ssl-proxy(ca-trustpoint)# match certificate acl
ssl-proxy(ca-trustpoint)# exit
ssl-proxy(config)#
ssl-proxy(config)# crypto ca 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 ca 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
```

This example shows that 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 service ssl-initiation client
ssl-proxy(config-ssl-proxy)# virtual ipaddr 8.100.100.126 protocol tcp port 81
ssl-proxy(config-ssl-proxy)# server ipaddr 191.162.2.8 protocol tcp port 443 secondary
ssl-proxy(config-ssl-proxy)# nat client client-nat
ssl-proxy(config-ssl-proxy)# trusted-ca root
ssl-proxy(config-ssl-proxy)# authenticate verify all
ssl-proxy(config-ssl-proxy)# inservice
ssl-proxy(config-ssl-proxy)# exit
ssl-proxy(config)#
ssl-proxy(config)# 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 ca trustpoint root
ssl-proxy(ca-trustpoint)# enrollment mode ra
ssl-proxy(ca-trustpoint)# enrollment terminal
ssl-proxy(ca-trustpoint)# crl optional
ssl-proxy(ca-trustpoint)# match certificate acl
ssl-proxy(ca-trustpoint)# exit
ssl-proxy(config)#
ssl-proxy(config)# crypto ca 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 sent to the server:

```
ssl-proxy# configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
ssl-proxy(config)# ssl-proxy policy http-header ssl-offload
ssl-proxy(config-http-header-policy)# custom "SOFTWARE VERSION :2.1(1)"
ssl-proxy(config-http-header-policy)# custom "module :SSL MODULE - CATALYST 6500"
ssl-proxy(config-http-header-policy)# custom
type-of-proxy:server_proxy_with_1024_bit_key_size
ssl-proxy(config-http-header-policy)# client-ip-port
ssl-proxy(config-http-header-policy)# prefix SSL-OFFLOAD
ssl-proxy(config-http-header-policy)# exit
ssl-proxy(config)#
ssl-proxy(config)# ssl-proxy service ssl-offload
ssl-proxy(config-ssl-proxy)# virtual ipaddr 8.100.100.126 protocol tcp port 443 secondary
ssl-proxy(config-ssl-proxy)# server ipaddr 191.162.2.8 protocol tcp port 80
```

```
ssl-proxy(config-ssl-proxy)# certificate rsa general-purpose trustpoint cert
ssl-proxy(config-ssl-proxy)# nat client client-nat
ssl-proxy(config-ssl-proxy)# policy http-header ssl-offload
ssl-proxy(config-ssl-proxy)# inservice
ssl-proxy(config-ssl-proxy)# exit
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 :2.1(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 :0          Custom Headers Inserted :2
    Session Id's Inserted    :0          Client Cert. Inserted   :0
    Client IP/Port Inserted  :2
    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 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 policy http-header ssl-offload
ssl-proxy(config-http-header-policy)# session
ssl-proxy(config-http-header-policy)# prefix SSL-OFFLOAD
ssl-proxy(config-http-header-policy)# exit
ssl-proxy(config)#
ssl-proxy(config)# ssl-proxy service ssl-offload
ssl-proxy(config-ssl-proxy)# virtual ipaddr 8.100.100.126 protocol tcp port 443 secondary
ssl-proxy(config-ssl-proxy)# server ipaddr 191.162.2.8 protocol tcp port 80
ssl-proxy(config-ssl-proxy)# certificate rsa general-purpose trustpoint cert
ssl-proxy(config-ssl-proxy)# nat client client-nat
ssl-proxy(config-ssl-proxy)# policy http-header ssl-offload
ssl-proxy(config-ssl-proxy)# inservice
ssl-proxy(config-ssl-proxy)# exit
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-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 the custom headers, the 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 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 policy http-header ssl-offload
ssl-proxy(config-http-header-policy)# custom "SOFTWARE VERSION :2.1(1)"
ssl-proxy(config-http-header-policy)# custom "module :SSL MODULE - CATALYST 6500"
ssl-proxy(config-http-header-policy)# custom
type-of-proxy:server_proxy_with_1024_bit_key_size
ssl-proxy(config-http-header-policy)# client-cert
ssl-proxy(config-http-header-policy)# client-ip-port
ssl-proxy(config-http-header-policy)# prefix SSL-OFFLOAD
ssl-proxy(config-http-header-policy)# exit
ssl-proxy(config)# ssl-proxy service ssl-offload
ssl-proxy(config-ssl-proxy)# virtual ipaddr 8.100.100.126 protocol tcp port 443 secondary
ssl-proxy(config-ssl-proxy)# server ipaddr 191.162.2.8 protocol tcp port 80
ssl-proxy(config-ssl-proxy)# certificate rsa general-purpose trustpoint cert
ssl-proxy(config-ssl-proxy)# nat client client-nat
ssl-proxy(config-ssl-proxy)# policy http-header ssl-offload
ssl-proxy(config-ssl-proxy)# trusted-ca root-ca
ssl-proxy(config-ssl-proxy)# authenticate verify all
ssl-proxy(config-ssl-proxy)# inservice
ssl-proxy(config-ssl-proxy)# exit
ssl-proxy(config)# exit
```

For the full SSL handshake, the custom headers, the 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 :2.1(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
```

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
```

URL Rewrite Examples

These examples show how to configure URL rewrite depending on the desired outcome and assume the following proxy configuration:

```
ssl-proxy service frontend
 virtual ipaddr 35.200.200.101 protocol tcp port 443 secondary
 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

This example shows how to configure a protocol rewrite (for example, HTTP to HTTPS) when the clear text port is a standard HTTP port 80. In this example, when the server sends the relocation string as http://ssl-136.cisco.com/index2.html, the SSL daughter card 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:

-
```
ssl-proxy policy url-rewrite test-url-rewrite
 url ssl-136.cisco.com
!
```
-
```
ssl-proxy policy url-rewrite test-url-rewrite
 url ssl*
!
```
-
```
Ssl-proxy policy url-rewrite test-url-rewrite
 url *com
!
```

Example 2

This example shows how to configure a protocol rewrite (for example, HTTP to HTTPS) when the clear text port is a nonstandard HTTP port. In this example, when the server sends the relocation string as http://ssl-136.cisco.com:100/index2.html, the SSL daughter card rewrites the string as https://ssl-136.cisco.com/index2.html.

To configure a protocol rewrite (HTTP to HTTPS) with a nonstandard clear text port, specify any of the following URL rewrite rules:

- ```
  ssl-proxy policy url-rewrite test-url-rewrite
   url ssl-136.cisco.com clearport 100
  !
  ```
- ```
  ssl-proxy policy url-rewrite test-url-rewrite
   url ssl* clearport 100
  !
  ```
- ```
  ssl-proxy policy url-rewrite test-url-rewrite
   url *com clearport 100
  !
  ```

Example 3

This example shows how to configure a protocol rewrite and SSL port rewrite when the clear text port is a standard HTTP port 80. In this example, when the server sends the relocation string as http://ssl-136.cisco.com/index2.html, the SSL daughter card rewrites the string as https://ssl-136.cisco.com:445/index2.html.

To configure a protocol rewrite (HTTP to HTTPS) with a nonstandard SSL text port, specify any of the following URL rewrite rules:

- ```
  ssl-proxy policy url-rewrite test-url-rewrite
   url ssl-136.cisco.com sslport 445
  !
  ```
- ```
  ssl-proxy policy url-rewrite test-url-rewrite
   url ssl* sslport 445
  !
  ```
- ```
  ssl-proxy policy url-rewrite test-url-rewrite
   url *com sslport 445
  !
  ```

Example 4

This example shows how to configure a protocol rewrite and SSL port rewrite when the clear text port is nonstandard. In this example, when the server sends the relocation string as http://ssl-136.cisco.com:100/index2.html, the SSL daughter card rewrites the string as https://ssl-136.cisco.com:445/index2.html.

To configure a protocol rewrite and SSL port rewrite with a nonstandard clear text port, specify any of the following URL rewrite rules:

- ```
  ssl-proxy policy url-rewrite test-url-rewrite
   url ssl-136.cisco.com clearport 100 sslport 445
  !
  ```
- ```
  ssl-proxy policy url-rewrite test-url-rewrite
   url ssl* clearport 100 sslport 445
  !
  ```
- ```
  ssl-proxy policy url-rewrite test-url-rewrite
   url *com clearport 100 sslport 445
  !
  ```
  ```

This example displays the above URL rewrite policy:

```
ssl-proxy# show ssl-proxy policy url-rewrite test-url-rewrite
Rule URL                                    Clearport SSLport
  1 *com                                      100        445
SSL proxy services using this policy:
          frontend
Usage count of this policy:1

ssl-proxy#
```

Troubleshooting and System Messages

This appendix describes how to troubleshoot the CSM-S and system messages.

Troubleshooting

CSM-S error messages may be received and reported in the system log (syslog). This section describes these messages. When a CSM-S is out of service, the module still replies to ARP requests but will not reply to pings.

System Messages

This section lists the system log (syslog) messages supported in the CSM-S.

The SSL daughter card will also generate system messages. The log levels are the same as those shown for the CSM.

Note The SSL daughter card messages have a prefix of "STE" instead of "CSM_SLB."

The Cisco IOS software message logs contain the warning level with this syntax:

`CSM_SLB_level-code`

Table C-1 lists the level codes.

Table C-1 Error Message Level Codes

Message Level	Code
LOG_EMERG	0 /* system is unusable */
LOG_ALERT	1 /* action must be taken immediately */
LOG_CRIT	2 /* critical conditions */
LOG_ERR	3 /* error conditions */
LOG_WARNING	4 /* warning conditions */
LOG_NOTICE	5 /* normal but signification condition */

Table C-1 Error Message Level Codes (continued)

Message Level	Code
LOG_INFO	6 /* informational */
LOG_DEBUG	7 /* debug-level messages */

When syslog messages are received, they are preceded by one of the following banners (where # is the slot number of the CSM-S module):

Error Message `CSM_SLB-3-IDB_ERROR Unknown error occurred while configuring IDB`

Explanation The MFSC could not create the internal interfaces for the CSM-S.

Recommended Action Either this version of the MFSC or the IDPROM were incorrectly programmed. Reprogram the MFSC or the IDPROM.

Error Message `CSM_SLB-3-OUTOFMEM Module [dec] memory error`

Explanation This problem is a general memory problem of the control module. The memory problem may lead to more serious operational problems in the CSM-S if it persists.

Recommended Action Run a memory check, or increase the memory size.

Error Message `CSM_SLB-3-PORTCHANNEL Portchannel allocation failed for module [dec]`

Explanation This problem occurs when there are more CSM-S modules inserted into the chassis than configured or when the slot number where the module was inserted was higher than anticipated.

Recommended Action Move the CSM-S module to a lower slot number to resolve the problem.

Error Message `CSM_SLB-3-RELOAD Module [dec] configuration reload failed`

Explanation The MSFC could not reload the existing configuration into the CSM-S module that came online. The cause of the problem may be the CLI error checking of the CSM-S.

Recommended Action Check the status of the CSM-S module such as diagnostic failure or version mismatch.

Error Message `CSM_SLB-3-UNEXPECTED Module [dec] unexpected error`
`CSM_SLB-3-REDUNDANCY Module [dec] FT error`
`CSM_SLB-4-REDUNDANCY_WARN Module [dec] FT warning`
`CSM_SLB-6-REDUNDANCY_INFO Module %d FT info`

```
CSM_SLB-3-ERROR Module [dec] error
CSM_SLB-4-WARNING Module [dec] warning
CSM_SLB-6-INFO Module [dec] info
```

Explanation These messages are generic headlines for error or warning messages. Additional details are located in the information string.

Recommended Action None.

Error Message `CSM_SLB-3-VERMISMATCH Module [dec] image version mismatch`

Explanation This problem indicates a version mismatch between the MFSC and the CSM-S code. This condition occurs only with the MFSC software versions earlier than the 12.1(8)EX release or CSM-S software versions earlier than the 2.1(1) release.

Recommended Action Upgrade or downgrade the MFSC version to match the CSM-S version to allow the CSM-S to come online.

Error Message `CSM_SLB-4-ARPCONFIG Module [dec] ARP configuration error`

Explanation There problem indicates an error in creating or removing static ARP configuration.

Recommended Action Recheck your ARP configuration.

Error Message `CSM_SLB-4-ERRPARSING Module [dec] configuration warning`
`SLB-REGEX: Syntactiv error in regular expression <x>.`
`SLB-REGEX: Parse error in regular expression <x>.`

Explanation This message is an error-checking message for the URL, cookie, or header regular expression matching.

Recommended Action Check the input matching strings.

Error Message `CSM_SLB-4-INVALIDID Module [dec] invalid ID`
`CSM_SLB-4-DUPLICATEID Module [dec] duplicate ID`

Explanation This message is an error-checking message between two modules when one module is calling another module.

Recommended Action Check the errors at the CLI level, which should prevent these errors from appearing.

Error Message `CSM_SLB-4-PROBECONFIG Module [dec] probe configuration error`

Explanation The CSM-S does not have enough memory to support the specified probe configuration.

Recommended Action Remove some of the probes from the server farm.

Error Message `CSM_SLB-4-REGEXMEM Module [dec] regular expression memory error`
`SLB-LCSC: Error detected while downloading URL configuration for vserver %s.`
`SLB-LCSC: Error detected while downloading COOKIE policy map for vserver <x>.`
`SLB-LCSC: Error detected while downloading COOKIE <x> for vserver <x>.`
`SLB-LCSC: There was an error downloading the configuration to hardware`
`SLB-LCSC: due to insufficient memory. Use the 'show ip slb memory'`
`SLB-LCSC: command to gather information about memory usage.`

Explanation These errors may occur if you configured complex URL, cookie, or header matching expressions. The CSM-S has a limited amount of space to compute the matching strings. Currently, the limit of 10 keywords (for example, "name*") are allowed per virtual server.

Recommended Action Combine (or remove) the expression strings to work around this problem.

Error Message `CSM_SLB-4-TOPOLOGY Module [dec] warning`

Explanation The CSM-S is detecting a "bridge loop" in the network.

Recommended Action Check the bridging device and the bridge-mode configurations of the multiple CSM-S modules located in the network.

Error Message `CSM_SLB-4-VERWILDCARD Received CSM-SLB module version wildcard on slot`

Explanation The CSM-S sends this message when you enter a debug command on the CSM-S console to work around the image version mismatch condition described in the previous error message.

Recommended Action This error is a debug condition only.

Error Message `SLB-DIAG: WatchDog task not responding.`
`SLB-DIAG: Fatal Diagnostic Error %x, Info %x.`
`SLB-DIAG: Diagnostic Warning %x, Info %x.`

Explanation Various diagnostic problems were encountered during the board boot procedure.

Recommended Action Check for a CSM-S hardware failure or corrupted software in the Flash memory.

Error Message `SLB-FT: Heartbeat intervals are not identical between ft pair.`
`SLB-FT: heartbeat interval is identical again`
`SLB-FT: The configurations are not identical between the members of the fault tolerant pair.`

Explanation These errors occur as a result of a misconfiguration between two redundant CSM-S modules.

Recommended Action Check the fault-tolerant configuration attributes and the real server and server farm configurations.

Error Message `SLB-FT: Standby is not monitoring active now.`

Explanation This problem is the result of a version mismatch of the fault-tolerance protocol between two versions of the CSM-S. The standby CSM-S stays as standby and does not take over as active if the primary CSM-S fails. The CSM-S does not support hitless (HA) upgrades in this situation.

Recommended Action Make sure that the fault-tolerance protocol versions match.

Server and Gateway Health Monitoring

Error Message `SLB-LCSC: No ARP response from gateway address A.B.C.D.`

Explanation The configured gateway A.B.C.D. did not respond to ARP requests.

Error Message `SLB-LCSC: No ARP response from real server A.B.C.D.`

Explanation The configured real server A.B.C.D. did not respond to ARP requests.

Error Message `SLB-LCSC: Health probe failed for server A.B.C.D on port P.`

Explanation The configured real server on port P of A.B.C.D. failed health checks.

Error Message `SLB-LCSC: DFP agent <x> disabled server <x>, protocol <x>, port <x>`

Explanation The configured DFP agent has reported a weight of 0 for the specified real server.

Error Message `SLB-LCSC: DFP agent <x> re-enabled server <x>, protocol <x>, port <x>`

Explanation The configured DFP agent has reported a non-zero weight for the specified real server.

Diagnostic Messages

Error Message `SLB-DIAG: WatchDog task not responding.`

Explanation A critical error occurred within the CSM hardware or software.

Error Message `SLB-DIAG: Fatal Diagnostic Error %x, Info %x.`

Explanation A hardware fault was detected. The hardware is unusable and must be repaired or replaced.

Error Message `SLB-DIAG: Diagnostic Warning %x, Info %x.`

Explanation A non-fatal hardware fault was detected.

Fault Tolerance Messages

Error Message `SLB-FT: No response from peer. Transitioning from Standby to Active.`

Explanation The CSM detected a failure in its fault-tolerant peer and has transitioned to the active state.

Error Message `SLB-FT: Heartbeat intervals are not identical between ft pair.`
`SLB-FT: Standby is not monitoring active now.`

Explanation Proper configuration of the fault-tolerance feature requires that the heartbeat intervals be identical between CSMs within the same fault-tolerance group, which is currently not the case. The fault-tolerance feature is disabled until the heartbeat intervals have been configured identically.

Error Message `SLB-FT: heartbeat interval is identical again`

Explanation The heartbeat intervals of different CSMs in the same fault-tolerance group have been reconfigured to be identical. The fault-tolerance feature will be re-enabled.

Error Message `SLB-FT: The configurations are not identical between the members of the fault tolerant pair.`

Explanation In order for the fault-tolerance system to preserve the sticky database, the different CSMs in the fault-tolerance group must be identically configured, which is not currently the case.

Regular Expression Errors

Error Message `SLB-LCSC: There was an error downloading the configuration to hardware`
`SLB-LCSC: due to insufficient memory. Use the 'show ip slb memory'`
`SLB-LCSC: command to gather information about memory usage.`
`SLB-LCSC: Error detected while downloading URL configuration for vserver %s.`

Explanation The hardware does not have sufficient memory to support the desired set of regular expressions. A different set of regular expressions must be configured for the system to function properly.

Error Message `SLB-REGEX: Parse error in regular expression <x>.`
`SLB-REGEX: Syntactic error in regular expression <x>.`

Explanation The configured regular expression does not conform to the regular expression syntax as described in the user manual.

Error Message `SLB-LCSC: Error detected while downloading COOKIE policy map for vserver <x>.`
`SLB-LCSC: Error detected while downloading COOKIE <x> for vserver <x>.`

Explanation An error occurred in configuring the cookie regular expressions for the virtual server. This error is likely due to a syntactic error in the regular expression (see below), or there is insufficient memory to support the desired regular expressions.

XML Errors

When an untolerated XML error occurs, the HTTP response contains a 200 code. The portion of the original XML document with the error is returned with an error element that contains the error type and description.

This example shows an error response to a condition where a virtual server name is missing:

```
<?xml version="1.0"?>
<config>
 <csm_module slot="4">
  <vserver>
   <error code="0x20">Missing attribute name in element

vserver</error>
  </vserver>
 </csm_module>
</config>
```

The error codes returned also correspond to the bits of the error tolerance attribute of the configuration element. Returned XML error codes are as follows:

```
XML_ERR_INTERNAL          = 0x0001,
XML_ERR_COMM_FAILURE      = 0x0002,
XML_ERR_WELLFORMEDNESS    = 0x0004,
XML_ERR_ATTR_UNRECOGNIZED = 0x0008,
XML_ERR_ATTR_INVALID      = 0x0010,
XML_ERR_ATTR_MISSING      = 0x0020,
XML_ERR_ELEM_UNRECOGNIZED = 0x0040,
XML_ERR_ELEM_INVALID      = 0x0080,
XML_ERR_ELEM_MISSING      = 0x0100,
XML_ERR_ELEM_CONTEXT      = 0x0200,
XML_ERR_IOS_PARSER        = 0x0400,
XML_ERR_IOS_MODULE_IN_USE = 0x0800,
XML_ERR_IOS_WRONG_MODULE  = 0x1000,
XML_ERR_IOS_CONFIG        = 0x2000
```

The default error_tolerance value is 0x48, which corresponds to ignoring unrecognized attributes and elements.

CSM XML Document Type Definition

You can use this DTD to configure the CSM as described in the "Configuring the XML Interface" section on page 10-20.

The CSM XML Document Type Definition (DTD) is as follows:

```
<!--
/*
 *  cisco_csm.dtd - XML DTD for CSM 3.2
 *
 *  January 2002  Paul Mathison
 *
 *  Copyright (c) 2002, 2003 by cisco Systems, Inc.
 *  All rights reserved
 */
-->

<!--
  Notes:
  Each element refers to a particular IOS CLI command.
  Each attribute refers to a command parameter.
  Except where noted, all "name" attributes are strings of length
    1 to 15, with no whitespace.
  IP address and mask attributes use standard "x.x.x.x" format.
-->

<!--
*************************************************************
  Elements and attributes required by various other elements
*************************************************************
-->

<!ELEMENT inservice EMPTY>
<!ATTLIST inservice
  sense (yes | no) #IMPLIED
>

<!ELEMENT inservice_standby EMPTY>
<!ATTLIST inservice_standby
  sense (yes | no) #IMPLIED
>

<!--
  backup_name is a string of length 1 to 15
  backup_sticky default is "no"
-->
<!ELEMENT serverfarm_ref EMPTY>
<!ATTLIST serverfarm_ref
  sense         (yes | no) #IMPLIED
```

```
  name          CDATA      #REQUIRED
  backup_name   CDATA      #IMPLIED
  backup_sticky (yes | no) #IMPLIED
>

<!--
  value is between 1 and 4294967295
-->
<!ELEMENT maxconns EMPTY>
<!ATTLIST maxconns
  sense (yes | no) #IMPLIED
  value NMTOKEN    #REQUIRED
>

<!--
  id is between 1 and 255
-->
<!ELEMENT reverse_sticky EMPTY>
<!ATTLIST reverse_sticky
  sense (yes | no) #IMPLIED
  id    NMTOKEN    #REQUIRED
>


<!--
************************************************************
  Elements and attributes required for env_variable
************************************************************
-->

<!--
  name is a string of length 1 to 31
  expression is a string of length 0 to 127
-->
<!ELEMENT env_variable EMPTY>
<!ATTLIST env_variable
  sense      (yes | no) #IMPLIED
  name       CDATA      #REQUIRED
  expression CDATA      #REQUIRED
>


<!--
************************************************************
  Elements and attributes required for owner
************************************************************
-->

<!--
  string is of length 1 to 200
-->
<!ELEMENT billing_info EMPTY>
<!ATTLIST billing_info
  sense  (yes | no) #IMPLIED
  string CDATA      #REQUIRED
>

<!--
  string is of length 1 to 200
-->
<!ELEMENT contact_info EMPTY>
<!ATTLIST contact_info
  sense  (yes | no) #IMPLIED
  string CDATA      #REQUIRED
```

```
>

<!ELEMENT owner (maxconns?, billing_info?, contact_info?)>
<!ATTLIST owner
  sense (yes | no) #IMPLIED
  name  CDATA      #REQUIRED
>


<!--
************************************************************
  Elements and attributes required for vlan
************************************************************
-->

<!ELEMENT vlan_address EMPTY>
<!ATTLIST vlan_address
  sense     (yes | no) #IMPLIED
  ipaddress NMTOKEN    #REQUIRED
  ipmask    NMTOKEN    #REQUIRED
>

<!ELEMENT gateway EMPTY>
<!ATTLIST gateway
  sense     (yes | no) #IMPLIED
  ipaddress NMTOKEN    #REQUIRED
>

<!--
  gateway uses standard x.x.x.x format
-->
<!ELEMENT route EMPTY>
<!ATTLIST route
  sense     (yes | no) #IMPLIED
  ipaddress NMTOKEN    #REQUIRED
  ipmask    NMTOKEN    #REQUIRED
  gateway   NMTOKEN    #REQUIRED
>

<!ELEMENT alias EMPTY>
<!ATTLIST alias
  sense     (yes | no) #IMPLIED
  ipaddress NMTOKEN    #REQUIRED
  ipmask    NMTOKEN    #REQUIRED
>

<!--
  id is between 2 and 4094
  Maximum of 7 gateways per vlan
  Maximum of 4095 routes per vlan
  Maximum of 255 aliases per vlan
  Global maximum of 255 unique vlan_addresses
  Global maximum of 255 vlan gateways (including routed gateways)
-->
<!ELEMENT vlan (vlan_address?, gateway*, route*, alias*)>
<!ATTLIST vlan
  sense (yes | no)        #IMPLIED
  id    NMTOKEN           #REQUIRED
  type  (client | server) #REQUIRED
>


<!--
************************************************************
```

```
  Elements and attributes required for script_file and script_task
************************************************************
-->

<!--
  url is a string of length 1 to 200
-->
<!ELEMENT script_file EMPTY>
<!ATTLIST script_file
  sense (yes | no) #IMPLIED
  url   CDATA      #REQUIRED
>

<!--
  id is between 1 and 100
  name is a string of length 1 to 31
  arguments is a string of length 0 to 199
-->
<!ELEMENT script_task EMPTY>
<!ATTLIST script_task
  sense     (yes | no) #IMPLIED
  id        NMTOKEN    #REQUIRED
  name      CDATA      #REQUIRED
  arguments CDATA      #IMPLIED
>


<!--
************************************************************
  Elements and attributes required for probe
************************************************************
-->

<!--
  value is between 2 and 65535 (default is 300)
-->
<!ELEMENT probe_failed EMPTY>
<!ATTLIST probe_failed
  sense (yes | no) #IMPLIED
  value NMTOKEN    #REQUIRED
>

<!--
  value is between 2 and 65535 (default is 120)
-->
<!ELEMENT probe_interval EMPTY>
<!ATTLIST probe_interval
  sense (yes | no) #IMPLIED
  value NMTOKEN    #REQUIRED
>

<!--
  value is between 0 and 65535 (default is 3)
-->
<!ELEMENT probe_retries EMPTY>
<!ATTLIST probe_retries
  sense (yes | no) #IMPLIED
  value NMTOKEN    #REQUIRED
>

<!--
  value is between 1 and 65535 (default 10)
-->
<!ELEMENT probe_open EMPTY>
```

```
<!ATTLIST probe_open
  sense (yes | no) #IMPLIED
  value NMTOKEN    #REQUIRED
>

<!--
  value is between 1 and 65535 (default 10)
-->
<!ELEMENT probe_receive EMPTY>
<!ATTLIST probe_receive
  sense (yes | no) #IMPLIED
  value NMTOKEN    #REQUIRED
>

<!--
  value is between 1 and 65535
-->
<!ELEMENT probe_port EMPTY>
<!ATTLIST probe_port
  sense (yes | no) #IMPLIED
  value NMTOKEN    #REQUIRED
>

<!--
  string is of length 1 to 64
-->
<!ELEMENT probe_domain EMPTY>
<!ATTLIST probe_domain
  sense  (yes | no) #IMPLIED
  string CDATA      #REQUIRED
>

<!ELEMENT probe_address EMPTY>
<!ATTLIST probe_address
  sense     (yes | no) #IMPLIED
  ipaddress NMTOKEN    #REQUIRED
  mode      (transparent | routed) "transparent"
>

<!ELEMENT probe_expect_address EMPTY>
<!ATTLIST probe_expect_address
  sense     (yes | no) #IMPLIED
  ipaddress NMTOKEN    #REQUIRED
>

<!--
  expression is a string of length 1 to 200
-->
<!ELEMENT probe_header EMPTY>
<!ATTLIST probe_header
  sense      (yes | no) #IMPLIED
  name       CDATA      #REQUIRED
  expression CDATA      #REQUIRED
>

<!--
  user is a string of length 1 to 15
  password is a string of length 1 to 15
-->
<!ELEMENT probe_credentials EMPTY>
<!ATTLIST probe_credentials
  sense    (yes | no) #IMPLIED
  user     CDATA      #REQUIRED
  password CDATA      ""
```

```
>

<!--
  url is a string of length 1 to 200
-->
<!ELEMENT probe_request EMPTY>
<!ATTLIST probe_request
  sense  (yes | no)   #IMPLIED
  method (get | head) #REQUIRED
  url    CDATA        "/"
>

<!--
  min_code is between 0 and 999
  max_code default is match min_code
-->
<!ELEMENT probe_expect_status EMPTY>
<!ATTLIST probe_expect_status
  sense    (yes | no) #IMPLIED
  min_code NMTOKEN    #REQUIRED
  max_code NMTOKEN    #IMPLIED
>

<!--
  name is a string of length 1 to 31
  arguments is a string of length 0 to 199
-->
<!ELEMENT script_ref EMPTY>
<!ATTLIST script_ref
  sense     (yes | no) #IMPLIED
  name      CDATA      #REQUIRED
  arguments CDATA      #IMPLIED
>

<!--
  secret is a string of length 1 to 32
-->
<!ELEMENT probe_secret EMPTY>
<!ATTLIST probe_secret
  sense  (yes | no) #IMPLIED
  secret CDATA      #REQUIRED
>

<!--
  Maximum of 255 probe_headers per http_probe
  probe_address must use mode "routed"
-->
<!ELEMENT http_probe (probe_failed?, probe_interval?, probe_retries?,
                      probe_open?, probe_receive?, probe_port?, probe_address?,
                      probe_request?, probe_credentials?, probe_header*,
                      probe_expect_status*)
>

<!--
  Maximum of 255 probe_expect_addresses per dns_probe
  probe_address must use mode "routed"
-->
<!ELEMENT dns_probe (probe_failed?, probe_interval?, probe_retries?,
                     probe_receive?, probe_port?, probe_address?, probe_domain?,
                     probe_expect_address*)
>

<!--
  probe_address must use mode "transparent"
```

```
-->
<!ELEMENT icmp_probe (probe_failed?, probe_interval?, probe_retries?,
                      probe_receive?, probe_address?)
>

<!ELEMENT tcp_probe (probe_failed?, probe_interval?, probe_retries?,
                     probe_open?, probe_port?)
>

<!ELEMENT udp_probe (probe_failed?, probe_interval?, probe_retries?,
                     probe_receive?, probe_port?)
>

<!ELEMENT smtp_probe (probe_failed?, probe_interval?, probe_retries?,
                      probe_open?, probe_receive?, probe_port?,
                      probe_expect_status*)
>

<!ELEMENT telnet_probe (probe_failed?, probe_interval?, probe_retries?,
                        probe_open?, probe_receive?, probe_port?,
                        probe_expect_status*)
>

<!ELEMENT ftp_probe (probe_failed?, probe_interval?, probe_retries?,
                     probe_open?, probe_receive?, probe_port?,
                     probe_expect_status*)
>

<!ELEMENT script_probe (probe_failed?, probe_interval?, probe_retries?,
                        probe_open?, probe_receive?, probe_port?, script_ref?)
>

<!--
  probe_address must use mode "routed"
-->
<!ELEMENT kalap_udp_probe (probe_failed?, probe_interval?, probe_retries?,
                           probe_receive?, probe_port?, probe_address?,
                           probe_secret?)
>

<!--
  probe_address must use mode "routed"
-->
<!ELEMENT kalap_tcp_probe (probe_failed?, probe_interval?, probe_retries?,
                           probe_open?, probe_receive?, probe_port?,
                           probe_address?, probe_secret?)
>

<!ELEMENT probe (http_probe | dns_probe | icmp_probe | tcp_probe | udp_probe |
                 smtp_probe | telnet_probe | ftp_probe | script_probe |
                 kalap_udp_probe | kalap_tcp_probe)
>
<!ATTLIST probe
  sense (yes | no)                #IMPLIED
  name  CDATA                     #REQUIRED
  type  (http | dns | icmp | tcp | udp |
         smtp | telnet | ftp | script |
         kal-ap-udp | kal-ap-tcp)   #REQUIRED
>


<!--
*************************************************************
  Elements and attributes required for natpool
```

```
************************************************************
-->

<!--
  Global maximum of 255 natpool addresses
-->
<!ELEMENT natpool EMPTY>
<!ATTLIST natpool
  sense    (yes | no) #IMPLIED
  name     CDATA      #REQUIRED
  first_ip NMTOKEN    #REQUIRED
  last_ip  NMTOKEN    #REQUIRED
  ipmask   NMTOKEN    #REQUIRED
>


<!--
************************************************************
  Elements and attributes required by maps
************************************************************
-->

<!--
  url is a string of length 1 to 200
  method is a string of length 1 to 15 (e.g. GET)
-->
<!ELEMENT url_rule EMPTY>
<!ATTLIST url_rule
  sense  (yes | no) #IMPLIED
  url    CDATA      #REQUIRED
  method CDATA      #IMPLIED
>

<!--
  name is a string of length 1 to 63
  expression is a string of length 1 to 127
-->
<!ELEMENT cookie_rule EMPTY>
<!ATTLIST cookie_rule
  sense      (yes | no) #IMPLIED
  name       CDATA      #REQUIRED
  expression CDATA      #REQUIRED
>

<!--
  name is a string of length 1 to 63
  expression is a string of length 1 to 127
-->
<!ELEMENT header_rule EMPTY>
<!ATTLIST header_rule
  sense      (yes | no) #IMPLIED
  name       CDATA      #REQUIRED
  expression CDATA      #REQUIRED
  type       (match | insert) "match"
>

<!--
  min_code and max_code are between 100 and 599
  threshold is between 1 and 4294967295, no effect for count action
  reset is between 0 and 4294967295 (0 means no reset)
-->
<!ELEMENT retcode_rule EMPTY>
<!ATTLIST retcode_rule
  sense     (yes | no)             #IMPLIED
```

```
  min_code  NMTOKEN                #REQUIRED
  max_code  NMTOKEN                #REQUIRED
  action    (count | log | remove) #REQUIRED
  threshold NMTOKEN                #REQUIRED
  reset     NMTOKEN                "0"
>

<!--
  domain is a string of length 1 to 127
-->
<!ELEMENT dns_rule EMPTY>
<!ATTLIST dns_rule
  sense  (yes | no) #IMPLIED
  domain CDATA      #REQUIRED
>

<!--
  Maximum of 1023 url_rules per map
-->
<!ELEMENT url_map (url_rule*)>
<!ATTLIST url_map
  sense (yes | no) #IMPLIED
  name  CDATA      #REQUIRED
>

<!--
  Maximum of 5 cookie_rules per map
-->
<!ELEMENT cookie_map (cookie_rule*)>
<!ATTLIST cookie_map
  sense (yes | no) #IMPLIED
  name  CDATA      #REQUIRED
>

<!--
  Maximum of 5 header_rules per map
-->
<!ELEMENT header_map (header_rule*)>
<!ATTLIST header_map
  sense (yes | no) #IMPLIED
  name  CDATA      #REQUIRED
>

<!--
  Maximum of 100 retcodes (not ranges) per map
-->
<!ELEMENT retcode_map (retcode_rule*)>
<!ATTLIST retcode_map
  sense (yes | no) #IMPLIED
  name  CDATA      #REQUIRED
>

<!--
  Maximum of 16 dns_rules per map
-->
<!ELEMENT dns_map (dns_rule*)>
<!ATTLIST dns_map
  sense (yes | no) #IMPLIED
  name  CDATA      #REQUIRED
>


<!--
*************************************************************
```

```
  Elements and attributes required for redirect_server
************************************************************
-->

<!--
  value is between 1 and 65535
-->
<!ELEMENT ssl_port EMPTY>
<!ATTLIST ssl_port
  sense (yes | no) #IMPLIED
  value NMTOKEN    #REQUIRED
>

<!--
  string is of length 1 to 127
-->
<!ELEMENT redirect_relocate EMPTY>
<!ATTLIST redirect_relocate
  sense  (yes | no)  #IMPLIED
  string CDATA       #REQUIRED
  code   (301 | 302) "302"
>

<!--
  string is of length 1 to 127
-->
<!ELEMENT redirect_backup EMPTY>
<!ATTLIST redirect_backup
  sense  (yes | no)  #IMPLIED
  string CDATA       #REQUIRED
  code   (301 | 302) "302"
>

<!ELEMENT redirect_server (ssl_port?, redirect_relocate?, redirect_backup?,
                           inservice?)
>
<!ATTLIST redirect_server
  sense (yes | no) #IMPLIED
  name  CDATA      #REQUIRED
>


<!--
************************************************************
  Elements and attributes required for named_real_server
************************************************************
-->

<!--
  string is of length 0 to 63
-->
<!ELEMENT location EMPTY>
<!ATTLIST location
  sense  (yes | no) #IMPLIED
  string CDATA      #REQUIRED
>

<!ELEMENT real_address EMPTY>
<!ATTLIST real_address
  sense     (yes | no) #IMPLIED
  ipaddress NMTOKEN    #REQUIRED
>

<!ELEMENT named_real_server (real_address?, location?)>
```

```
<!ATTLIST named_real_server
  sense (yes | no) #IMPLIED
  name  CDATA      #REQUIRED
>


<!--
************************************************************
  Elements and attributes required for real_server
************************************************************
-->

<!--
  value is between 0 and 100
-->
<!ELEMENT weight EMPTY>
<!ATTLIST weight
  sense (yes | no) #IMPLIED
  value NMTOKEN    #REQUIRED
>

<!--
  value is between 1 and 4294967295
-->
<!ELEMENT minconns EMPTY>
<!ATTLIST minconns
  sense (yes | no) #IMPLIED
  value NMTOKEN    #REQUIRED
>

<!--
  value is between 2 and 254 (default is 254)
-->
<!ELEMENT load_threshold EMPTY>
<!ATTLIST load_threshold
  sense (yes | no) #IMPLIED
  value NMTOKEN    #REQUIRED
>

<!--
  tag is a string of length 0 to 32
-->
<!ELEMENT real_probe_ref EMPTY>
<!ATTLIST real_probe_ref
  sense (yes | no) #IMPLIED
  name  CDATA      #REQUIRED
  tag   CDATA      #IMPLIED
>

<!--
  either ipaddress or named_real_server_ref is required
  port is between 0 and 65535 (0 means no port translation)
-->
<!ELEMENT real_server_backup EMPTY>
<!ATTLIST real_server_backup
  sense                (yes | no) #IMPLIED
  ipaddress            NMTOKEN    #IMPLIED
  named_real_server_ref CDATA     #IMPLIED
  port                 NMTOKEN    "0"
>

<!--
  either ipaddress or named_real_server_ref is required
  port is between 0 and 65535 (0 means no port translation)
```

```
  Global maximum of 4095 real_servers
-->
<!ELEMENT real_server (weight?, minconns?, maxconns?, load_threshold?,
                       real_probe_ref?, real_server_backup?, inservice?,
                       inservice_standby?)
>
<!ATTLIST real_server
  sense                  (yes | no) #IMPLIED
  ipaddress              NMTOKEN    #IMPLIED
  named_real_server_ref CDATA      #IMPLIED
  port                   NMTOKEN    "0"
>


<!--
************************************************************
  Elements and attributes required for serverfarm
************************************************************
-->

<!ELEMENT retcode_map_ref EMPTY>
<!ATTLIST retcode_map_ref
  sense (yes | no) #IMPLIED
  name  CDATA      #REQUIRED
>

<!--
  retries is between 0 and 65534
  failed is between 0 and 65535
-->
<!ELEMENT health EMPTY>
<!ATTLIST health
  sense     (yes | no) #IMPLIED
  retries   NMTOKEN    #REQUIRED
  failed    NMTOKEN    #REQUIRED
>

<!ELEMENT failaction EMPTY>
<!ATTLIST failaction
  sense (yes | no)         #IMPLIED
  value (purge | reassign) #REQUIRED
>

<!ELEMENT probe_ref EMPTY>
<!ATTLIST probe_ref
  sense (yes | no) #IMPLIED
  name  CDATA      #REQUIRED
>

<!ELEMENT natpool_ref EMPTY>
<!ATTLIST natpool_ref
  sense (yes | no) #IMPLIED
  name  CDATA      #REQUIRED
>

<!ELEMENT server_nat EMPTY>
<!ATTLIST server_nat
  sense (yes | no) #IMPLIED
>

<!--
  value is between 0 and 65533
-->
<!ELEMENT bind_id EMPTY>
```

```
<!ATTLIST bind_id
  sense (yes | no) #IMPLIED
  value NMTOKEN    #REQUIRED
>

<!--
  hash_ip_type and ipmask valid only when value = hash_ip
-->
<!ELEMENT predictor EMPTY>
<!ATTLIST predictor
  sense        (yes | no)                      #IMPLIED
  value        (roundrobin | leastconns |
                hash_ip | hash_url | forward) #REQUIRED
  hash_ip_type (source | destination | both)  "both"
  ipmask       NMTOKEN                        "255.255.255.255"
>

<!ELEMENT dns_predictor EMPTY>
<!ATTLIST dns_predictor
  sense        (yes | no)                      #IMPLIED
  value        (roundrobin | ordered-list |
                leastload | hash_domain |
                hash_ip | hash_ip_domain)     #REQUIRED
>

<!ELEMENT serverfarm (predictor?, natpool_ref?, server_nat?, health?,
                      bind_id?, retcode_map_ref?, failaction?,
                      redirect_server*, real_server*, probe_ref*)
>
<!ATTLIST serverfarm
  sense (yes | no) #IMPLIED
  name  CDATA      #REQUIRED
>

<!--
  real_server "port" attribute is ignored
-->
<!ELEMENT dns_serverfarm (dns_predictor?, real_server*)>
<!ATTLIST dns_serverfarm
  sense (yes | no) #IMPLIED
  name  CDATA      #REQUIRED
  type  (dns-vip | dns-ns) #REQUIRED
>


<!--
************************************************************
  Elements and attributes required for sticky_group
************************************************************
-->

<!--
  src_ip and dest_ip are necessary for IP-based sticky_groups
  expression is necessary for SSL, cookie, and header-based sticky_groups
  expression is a string of length 0 to 127
-->
<!ELEMENT static_sticky EMPTY>
<!ATTLIST static_sticky
  sense      (yes | no) #IMPLIED
  real_ip    NMTOKEN    #REQUIRED
  expression NMTOKEN    #IMPLIED
  src_ip     NMTOKEN    #IMPLIED
  dest_ip    NMTOKEN    #IMPLIED
>
```

```
<!--
  This only applies to cookie and header-based sticky_groups
  offset is between 0 and 3999
  length is between 1 and 4000
-->
<!ELEMENT sticky_offset EMPTY>
<!ATTLIST sticky_offset
  sense (yes | no) #IMPLIED
  offset NMTOKEN   #REQUIRED
  length NMTOKEN   #REQUIRED
>

<!--
  This only applies to cookie-based sticky_groups
  name is a string of length 1 to 63
-->
<!ELEMENT cookie_secondary EMPTY>
<!ATTLIST cookie_secondary
  sense (yes | no) #IMPLIED
  name  CDATA      #REQUIRED
>

<!--
  id is between 1 and 255
  timeout is between 1 and 65535
  ipmask required for ip types
  cookie is a string of length 1 to 63, req for type=cookie or cookie_insert
  header is a string of length 1 to 63, req for type=header
-->
<!ELEMENT sticky_group (sticky_offset?, cookie_secondary?, static_sticky*)>
<!ATTLIST sticky_group
  sense   (yes | no)          #IMPLIED
  id      NMTOKEN             #REQUIRED
  timeout NMTOKEN             "1440"
  type    (ip | cookie | ssl |
           ip_src | ip_dest | ip_src_dest |
           cookie_insert | header) #REQUIRED
  ipmask  NMTOKEN             #IMPLIED
  cookie  CDATA               #IMPLIED
  header  CDATA               #IMPLIED
>


<!--
************************************************************
  Elements and attributes required for policy
************************************************************
-->

<!ELEMENT url_map_ref EMPTY>
<!ATTLIST url_map_ref
  sense (yes | no) #IMPLIED
  name  CDATA      #REQUIRED
>

<!ELEMENT cookie_map_ref EMPTY>
<!ATTLIST cookie_map_ref
  sense (yes | no) #IMPLIED
  name  CDATA      #REQUIRED
>

<!ELEMENT header_map_ref EMPTY>
<!ATTLIST header_map_ref
```

```
  sense (yes | no) #IMPLIED
  name  CDATA      #REQUIRED
>

<!ELEMENT dns_map_ref EMPTY>
<!ATTLIST dns_map_ref
  sense (yes | no) #IMPLIED
  name  CDATA      #REQUIRED
>

<!--
  order is between 1 and 3 (corresponds to "primary", "secondary", "tertiary")
  ttl is between 1 and 604800 (default is 20)
  response_count is between 1 and 8 (default is 1)
-->
<!ELEMENT dns_serverfarm_ref EMPTY>
<!ATTLIST dns_serverfarm_ref
  sense (yes | no) #IMPLIED
  order NMTOKEN    #REQUIRED
  name  CDATA      #REQUIRED
  ttl   NMTOKEN    #IMPLIED
  response_count NMTOKEN #IMPLIED
>

<!--
  Reference to an IOS standard IP access list
  Specify either the id (range 1 to 99) or name
  name is a string of length 1 to 200
-->
<!ELEMENT client_group_ref EMPTY>
<!ATTLIST client_group_ref
  sense (yes | no) #IMPLIED
  name CDATA   #IMPLIED
  id   NMTOKEN #IMPLIED
>

<!--
  id is between 1 and 255
-->
<!ELEMENT sticky_group_ref EMPTY>
<!ATTLIST sticky_group_ref
  sense (yes | no) #IMPLIED
  id    NMTOKEN    #REQUIRED
>

<!--
  value is between 0 and 63
-->
<!ELEMENT dscp EMPTY>
<!ATTLIST dscp
  sense (yes | no) #IMPLIED
  value NMTOKEN    #REQUIRED
>

<!ELEMENT policy (serverfarm_ref?, client_group_ref?, sticky_group_ref?,
                  reverse_sticky?, dscp?, url_map_ref?, cookie_map_ref?,
                  header_map_ref?)
>
<!ATTLIST policy
  sense (yes | no) #IMPLIED
  name  CDATA      #REQUIRED
>

<!--
```

```
  Maximum of 3 dns_serverfarm_refs per dns_policy (one for each order)
-->
<!ELEMENT dns_policy (dns_serverfarm_ref*, client_group_ref?, dns_map_ref?)>
<!ATTLIST dns_policy
  sense (yes | no) #IMPLIED
  name CDATA       #REQUIRED
>


<!--
************************************************************
  Elements and attributes required for vserver
************************************************************
-->

<!--
  protocol is between 0 and 255 (0 = any, 1 = icmp, 6 = tcp, 17 = udp)
  port is between 0 and 65535 (0 means any)
  ftp and termination service valid only for tcp protocol
  rtsp service valid for tcp and udp protocol
  per-packet service valid only for non-tcp protocols
-->
<!ELEMENT virtual EMPTY>
<!ATTLIST virtual
  sense     (yes | no)          #IMPLIED
  ipaddress NMTOKEN             #REQUIRED
  ipmask    NMTOKEN             "255.255.255.255"
  protocol  NMTOKEN             #REQUIRED
  port      NMTOKEN             #REQUIRED
  service   (none | ftp | rtsp |
             termination | per-packet) "none"
>

<!ELEMENT client EMPTY>
<!ATTLIST client
  sense     (yes | no) #IMPLIED
  ipaddress NMTOKEN    #REQUIRED
  ipmask    NMTOKEN    "255.255.255.255"
  exclude   (yes | no) "no"
>

<!--
  timeout is between 1 and 65535
  group is between 0 and 255 (if nonzero, refers to an ip sticky_group)
-->
<!ELEMENT sticky EMPTY>
<!ATTLIST sticky
  sense   (yes | no) #IMPLIED
  timeout NMTOKEN    #REQUIRED
  group   NMTOKEN    "0"
  ipmask  NMTOKEN    "255.255.255.255"
>

<!ELEMENT policy_ref EMPTY>
<!ATTLIST policy_ref
  sense (yes | no) #IMPLIED
  name  CDATA      #REQUIRED
>

<!ELEMENT dns_policy_ref EMPTY>
<!ATTLIST dns_policy_ref
  sense (yes | no) #IMPLIED
  name  CDATA      #REQUIRED
>
```

```
<!--
  begin and end are strings, 0-length ok
  total length of begin and end should not exceed 200
-->
<!ELEMENT url_hash EMPTY>
<!ATTLIST url_hash
  sense (yes | no) #IMPLIED
  begin CDATA      #REQUIRED
  end   CDATA      #REQUIRED
>

<!--
  value is between 2 and 4094
-->
<!ELEMENT vlan_id EMPTY>
<!ATTLIST vlan_id
  sense (yes | no) #IMPLIED
  value NMTOKEN    #REQUIRED
>

<!--
  value is between 2 and 65535
-->
<!ELEMENT idle EMPTY>
<!ATTLIST idle
  sense (yes | no) #IMPLIED
  value NMTOKEN    #REQUIRED
>

<!--
  value is between 1 and 65535
-->
<!ELEMENT pending EMPTY>
<!ATTLIST pending
  sense (yes | no) #IMPLIED
  value NMTOKEN    #REQUIRED
>

<!ELEMENT replicate_csrp EMPTY>
<!ATTLIST replicate_csrp
  sense (yes | no)            #IMPLIED
  value (sticky | connection) #REQUIRED
>

<!ELEMENT advertise EMPTY>
<!ATTLIST advertise
  sense (yes | no)        #IMPLIED
  value (always | active) #REQUIRED
>

<!ELEMENT persistent EMPTY>
<!ATTLIST persistent
  sense (yes | no) #IMPLIED
>

<!--
  value is between 1 and 4000
-->
<!ELEMENT parse_length EMPTY>
<!ATTLIST parse_length
  sense (yes | no) #IMPLIED
  value NMTOKEN    #REQUIRED
>
```

```
<!--
  string is of length 1 to 127
-->
<!ELEMENT domain EMPTY>
<!ATTLIST domain
  sense  (yes | no) #IMPLIED
  string CDATA      #REQUIRED
>

<!ELEMENT unidirectional EMPTY>
<!ATTLIST unidirectional
  sense (yes | no | default) #IMPLIED
>

<!ELEMENT owner_ref EMPTY>
<!ATTLIST owner_ref
  sense (yes | no) #IMPLIED
  name  CDATA      #REQUIRED
>

<!--
  offset is between 0 and 3999
  length is between 1 and 4000
-->
<!ELEMENT ssl_sticky_offset EMPTY>
<!ATTLIST ssl_sticky_offset
  sense (yes | no) #IMPLIED
  offset NMTOKEN   #REQUIRED
  length NMTOKEN   #REQUIRED
>

<!--
  Maximum of 1023 domains per vserver
  Default idle is 3600
  Default pending is 30
-->
<!ELEMENT vserver (virtual?, vlan_id?, unidirectional?, owner_ref?,
                   maxconns?, ssl_sticky_offset?, idle?, pending?,
                   replicate_csrp?, advertise?, persistent?, parse_length?,
                   inservice?, url_hash?, policy_ref*, domain*,
                   serverfarm_ref?, sticky?, reverse_sticky?, client*)
>
<!ATTLIST vserver
  sense (yes | no) #IMPLIED
  name  CDATA      #REQUIRED
>

<!ELEMENT dns_vserver (inservice?, dns_policy_ref*)>
<!ATTLIST dns_vserver
  sense (yes | no) #IMPLIED
  name  CDATA      #REQUIRED
>


<!--
************************************************************
  Elements and attributes required for dfp
************************************************************
-->

<!--
  port is between 1 and 65535
-->
```

```
<!ELEMENT dfp_manager EMPTY>
<!ATTLIST dfp_manager
  sense (yes | no) #IMPLIED
  port  NMTOKEN    #REQUIRED
>

<!--
  port is between 1 and 65535
  timeout is between 0 and 65535
  retry is between 0 and 65535 (must specify timeout)
  interval is between 1 and 65535 (must specify retry)
-->
<!ELEMENT dfp_agent EMPTY>
<!ATTLIST dfp_agent
  sense     (yes | no) #IMPLIED
  ipaddress NMTOKEN    #REQUIRED
  port      NMTOKEN    #REQUIRED
  timeout   NMTOKEN    "0"
  retry     NMTOKEN    "0"
  interval  NMTOKEN    "180"
>

<!--
  password is a string of length 1 to 64
  timeout is between 0 and 65535
-->
<!ELEMENT dfp (dfp_manager?, dfp_agent*)>
<!ATTLIST dfp
  sense    (yes | no) #IMPLIED
  password CDATA      #IMPLIED
  timeout  NMTOKEN    "180"
>


<!--
*************************************************************
  Elements and attributes required for udp_capp
*************************************************************
-->

<!--
  secret is a string of length 1 to 32
-->
<!ELEMENT capp_options EMPTY>
<!ATTLIST capp_options
  sense      (yes | no) #IMPLIED
  ipaddress  NMTOKEN    #REQUIRED
  encryption (md5)      "md5"
  secret     CDATA      #REQUIRED
>

<!--
  value is between 1 and 65535
-->
<!ELEMENT capp_port EMPTY>
<!ATTLIST capp_port
  sense (yes | no) #IMPLIED
  value NMTOKEN    #REQUIRED
>

<!ELEMENT capp_secure EMPTY>
<!ATTLIST capp_secure
  sense (yes | no) #IMPLIED
>
```

```
<!--
  Maximum of 16 capp_options
  Default capp_port is 5002
-->
<!ELEMENT udp_capp (capp_port?, capp_secure?, capp_options*)>
<!ATTLIST udp_capp
  sense (yes | no) #IMPLIED
>


<!--
************************************************************
  Elements and attributes required for ft
************************************************************
-->

<!ELEMENT ft_preempt EMPTY>
<!ATTLIST ft_preempt
  sense (yes | no) #IMPLIED
>

<!--
  value is between 1 and 254
-->
<!ELEMENT ft_priority EMPTY>
<!ATTLIST ft_priority
  sense (yes | no) #IMPLIED
  value NMTOKEN    #REQUIRED
>

<!--
  value is between 1 and 65535
-->
<!ELEMENT ft_failover EMPTY>
<!ATTLIST ft_failover
  sense (yes | no) #IMPLIED
  value NMTOKEN #REQUIRED
>

<!--
  value is between 1 and 65535
-->
<!ELEMENT ft_heartbeat EMPTY>
<!ATTLIST ft_heartbeat
  sense (yes | no) #IMPLIED
  value NMTOKEN    #REQUIRED
>

<!--
  group is between 1 and 254
  vlan_id is between 2 and 4094, and must *not* match id of
    existing client or server vlan configured for csm_module
  Default ft_preempt is off
  Default ft_priority is 10
  Default ft_failover is 3
  Default ft_heartbeat is 1
-->
<!ELEMENT ft (ft_preempt?, ft_priority?, ft_failover?, ft_heartbeat?)>
<!ATTLIST ft
  sense   (yes | no) #IMPLIED
  group   NMTOKEN    #REQUIRED
  vlan_id NMTOKEN    #REQUIRED
>
```

```
<!--
************************************************************
  Elements and attributes required for static_nat
************************************************************
-->

<!ELEMENT static_real EMPTY>
<!ATTLIST static_real
  sense     (yes | no) #IMPLIED
  ipaddress NMTOKEN    #REQUIRED
  ipmask    NMTOKEN    "255.255.255.255"
>

<!--
  ipaddress is required for type=ip
  Global maximum of 16383 static_reals
-->
<!ELEMENT static_nat (static_real*)>
<!ATTLIST static_nat
  sense     (yes | no)            #IMPLIED
  type      (drop | ip | virtual) #REQUIRED
  ipaddress NMTOKEN               #IMPLIED
>


<!--
************************************************************
  Elements and attributes required for static_arp
************************************************************
-->

<!--
  macaddress has the form "hhhh.hhhh.hhhh", where h is a hex digit
  vlan_id is between 2 and 4094
-->
<!ELEMENT static_arp EMPTY>
<!ATTLIST static_arp
  sense      (yes | no) #IMPLIED
  ipaddress  NMTOKEN    #REQUIRED
  macaddress NMTOKEN    #REQUIRED
  vlan_id    NMTOKEN    #REQUIRED
>


<!--
************************************************************
  root definition for csm_module
************************************************************
-->

<!--
  slot is between 1 and MAXSLOT (depends on chassis)
  Maximum of 4095 probes
  Maximum of 1023 url_maps
  Maximum of 1023 cookie_maps
  Maximum of 1023 header_maps
  Maximum of 1023 retcode_maps
  Maximum of 1023 dns_maps
  Maximum of 4095 serverfarms and dns_serverfarms
  Maximum of 255 sticky_groups (including those id=0 groups created
                                implicitly for vservers)
  Maximum of 4000 vservers and dns_vservers
```

```
  Maximum of 255 owners
  Maximum of 16383 static_arp entries
-->
<!ELEMENT csm_module (env_variable*, owner*, vlan*, script_file*, script_task*,
                      probe*, natpool*, url_map*, cookie_map*, header_map*,
                      retcode_map*, dns_map*, named_real_server*,
                      serverfarm*, dns_serverfarm*, sticky_group*,
                      policy*, dns_policy*, vserver*, dns_vserver*,
                      dfp?, udp_capp?, ft?, static_nat*, static_arp*)
>
<!ATTLIST csm_module
  sense (yes | no) #IMPLIED
  slot  NMTOKEN    #REQUIRED
>

<!--
************************************************************
  actions
************************************************************
-->

<!--
  error_tolerance is a 32-bit value, specified
    in hex or decimal, which acts as a bitmask
    for specifying which error types should be
    ignored.  See valid error types below.  Default is 0x0048.
  dtd_version is a string that specifies the set of
    configurable CSM features, and should match the CSM version
    specified at the top of this DTD.  Default is "2.2".
    Note that if the version is higher than the CSM can
    handle, an error may be returned.  In most cases,
    the CSM will do its best to interpret the document,
    even if dtd_version is missing or higher than expected.
-->
<!ELEMENT config (csm_module)>
<!ATTLIST config
  error_tolerance NMTOKEN #IMPLIED
  dtd_version     NMTOKEN #IMPLIED

<!--
**************
In case of error, the response document will include an "error" child element
in the offending element.  The error element takes the form:
<!ELEMENT error EMPTY>
<!ATTLIST error
  code NMTOKEN #REQUIRED
>
The body of the error element is a description string.
Attribute "code" is a hex value representing a mask of possible error codes:
    XML_ERR_INTERNAL          = 0x0001 /* internal memory or coding error */
    XML_ERR_COMM_FAILURE      = 0x0002 /* communication failure */
    XML_ERR_WELLFORMEDNESS    = 0x0004 /* not a wellformed XML document */
    XML_ERR_ATTR_UNRECOGNIZED = 0x0008 /* found an unrecognized attribute */
    XML_ERR_ATTR_INVALID      = 0x0010 /* found invalid value in attribute */
    XML_ERR_ATTR_MISSING      = 0x0020 /* required attribute missing */
    XML_ERR_ELEM_UNRECOGNIZED = 0x0040 /* found an unrecognized element */
    XML_ERR_ELEM_INVALID      = 0x0080 /* found invalid element */
    XML_ERR_ELEM_MISSING      = 0x0100 /* required element missing */
    XML_ERR_ELEM_CONTEXT      = 0x0200 /* valid element found in wrong place */
    XML_ERR_IOS_PARSER        = 0x0400 /* IOS unable to parse command */
    XML_ERR_IOS_MODULE_IN_USE = 0x0800 /* Another user is configuring CSM */
    XML_ERR_IOS_WRONG_MODULE  = 0x1000 /* Tried to configure unavailable CSM */
    XML_ERR_IOS_CONFIG        = 0x2000 /* IOS configuration error */
**************
```

A

B

C

D

E

F

G

H

I

K

L

M

N

O

P

R

S

T

U

V

W