



Configuring Additional Features and Options

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

- [Configuring Session Persistence \(Stickiness\), page 10-1](#)
- [Configuring Route Health Injection, page 10-5](#)
- [Environmental Variables, page 10-8](#)
- [Configuring Persistent Connections, page 10-13](#)
- [HTTP Header Insert, page 10-14](#)
- [Configuring Global Server Load Balancing, page 10-15](#)
- [Configuring Network Management, page 10-20](#)
- [Configuring Server Application State Protocol, page 10-24](#)
- [Back-End Encryption, page 10-27](#)

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
<pre>Router(config-module-csm)# sticky sticky-group-id {netmask netmask cookie name ssl} [address [source destination both]] [timeout sticky-time]</pre>	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)# vsserver vs_sticky_ck
Router(config-slb-vsserver)# virtual 10.8.0.125 tcp 90
Router(config-slb-vsserver)# slb-policy policy_sticky_ck
Router(config-slb-vsserver)# inservice
Router(config-slb-vsserver)# 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, page 10-5](#)
- [Configuring RHI for Virtual Servers, page 10-7](#)

Understanding RHI

These sections describe the RHI:

- [RHI Overview, page 10-6](#)
- [Routing to VIP Addresses Without RHI, page 10-6](#)
- [Routing to VIP Addresses With RHI, page 10-7](#)

- [Understanding How the CSM-S Determines VIP Availability, page 10-7](#)
- [Understanding Propagation of VIP Availability Information, page 10-7](#)

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

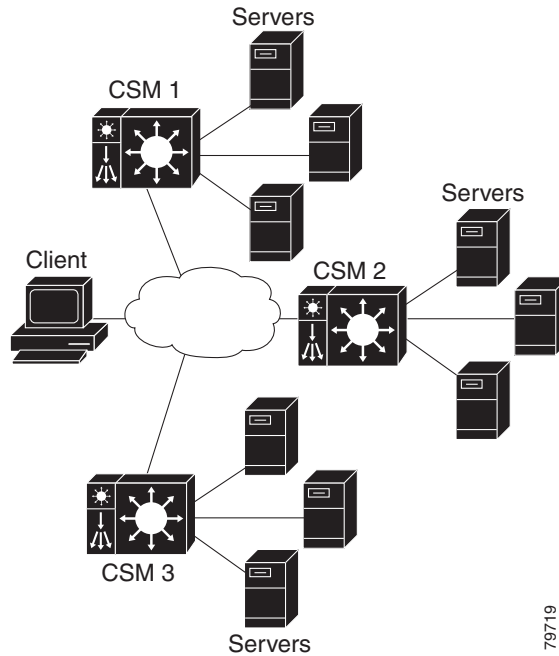
GSLB performs load balancing between multiple, dispersed hosting sites by directing client connections through DNS to different server farms and real servers based on load availability. GSLB is performed using access lists, maps, server farms, and load-balancing algorithms. Table 10-3 provides an overview 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 submenu.
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 submenu.

	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-address	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-robin
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) # serverfarm 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, page 10-20](#)
- [Configuring the XML Interface, page 10-20](#)

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 public casa slb	Defines the IP address of an external network management device to which traps are sent, enables casa and SLB event traps.
Step 3	Router (config)# snmp-server enable traps slb csrp	Enables SNMP traps for real servers ¹ for SLB FT Replication Protocol traps.

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 not 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 <i>user-name password password</i>	(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, OUTFSERVICE, 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, OUTFSERVICE, 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.

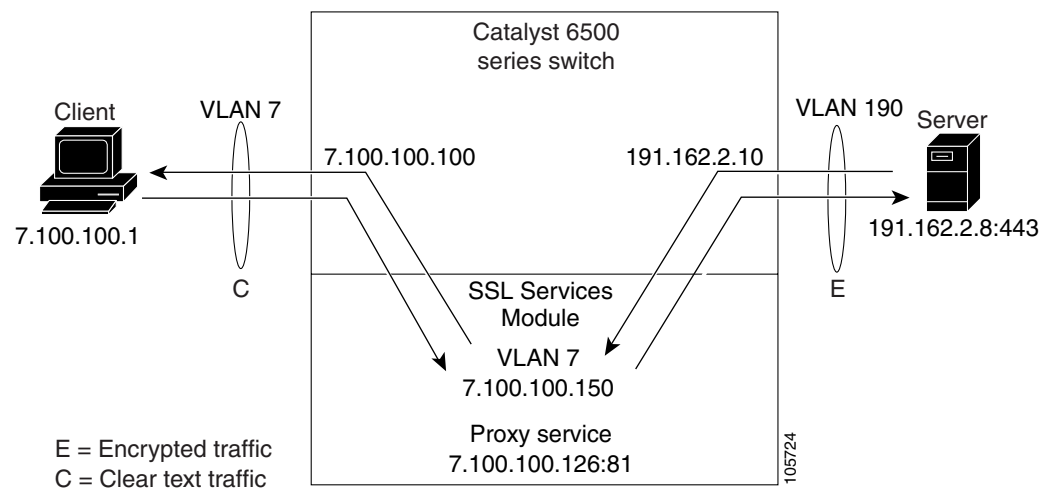
The SSL proxy VLAN 7 has the following configuration:

- 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
```