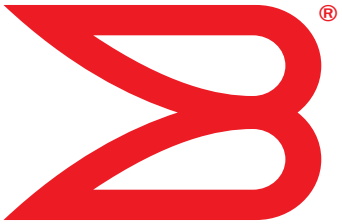
53-1002435-03
June, 2012



ServerIron ADX

Advanced Server Load Balancing Guide

Supporting Brocade ServerIron ADX version 12.4.00a

BROCADE

Brocade Communications Systems, Incorporated

Corporate and Latin American Headquarters
Brocade Communications Systems, Inc.
130 Holger Way
San Jose, CA 95134
E-mail: info@brocade.com

Asia-Pacific Headquarters
Brocade Communications Systems China HK, Ltd.
No. 1 Guanghua Road
Chao Yang District
Units 2718 and 2818
Beijing 100020, China
Tel: +8610 6588 8888
Fax: +8610 6588 9999
E-mail: china-info@brocade.com

European Headquarters
Brocade Communications Switzerland Sàrl
Centre Swissair
Tour B - 4ème étage
29, Route de l'Aéroport
Case Postale 105
CH-1215 Genève 15
Switzerland
Tel: +41 22 799 5640
Fax: +41 22 799 5641
E-mail: emea-info@brocade.com

Asia-Pacific Headquarters
Brocade Communications Systems Co., Ltd. (Shenzhen WFOE)
Citic Plaza
No. 233 Tian He Road North
Unit 1308 – 13th Floor
Guangzhou, China
Tel: +8620 3891 2000
Fax: +8620 3891 2111
E-mail: china-info@brocade.com

Document History

Title	Publication number	Summary of changes	Date
ServerIron ADX Advanced Server Load Balancing Guide	*53-1002435-01*	Release 12.3.03 has been updated with enhancements in release 12.4.00	January, 2012
ServerIron ADX Advanced Server Load Balancing Guide	*53-1002435-02*	Updated to support 12.4.00a release	April, 2012
ServerIron ADX Advanced Server Load Balancing Guide	*53-1002435-03*	Release 12.4.00 documentation has been updated	June, 2012

Contents

About This Document

Audience vii

Supported hardware and software vii

Document conventions vii
- Text formatting vii
- Command syntax conventions viii
- Notes, cautions, and danger notices viii

Notice to the reader ix

Related publications ix

Getting technical help ix
- To contact Technical Support, got to http://www.brocade.com/services-support/index.page for the latest e-mail and telephone contact information.. ix

Chapter 1 SIP Server Load Balancing

SIP overview 1
- SIP packet flow 2
- SIP client registration 4
- SIP terminology 4
- SIP message headers 4

SIP SLB and call persistence using ServerIron ADX 6
- SIP and call persistence specifications 7
- Sample deployment topologies 7
- SIP server health monitoring 11

Configuring SIP SLB 11
- SIP SLB over UDP (Stateless SLB mode) 11
- SIP SLB over UDP (stateful SLB mode) 15
- SIP SLB over TCP 17
- SIP SLB over TCP sample configuration 22
- Other SIP SLB over TCP options 22
- Debug commands 25

SIP SLB command reference 27
- Real server configuration mode 27
- Virtual server configuration mode 28
- Sample configuration 28

Chapter 2 Transparent Cache Switching

Transparent cache switching overview 29
Operation of transparent cache switching 29
Response to cache server failures 31
Stateful caching 31
Advanced statistics 32

Sample deployment topologies 32
Basic TCS 32
TCS with spoofing 33
TCS with destination NAT 33
TCS with source NAT 34
VIPs with reverse proxy 35

Configuring transparent cache switching 36
Configuration notes 36
Defining a cache server 37
Identify application ports for caching 37
Assigning web cache servers to cache groups 39
Enabling transparent cache switching 40

Other transparent cache switching options 41
Resetting the server cache table 41
Disabling a cache group or a server in a cache group 41
Removing or re-assigning an interface 41
Controlling traffic distribution among cache servers 42
Selecting server selection methods with cache groups 45
Resilient hashing for maximum cache persistence 47
Cache route optimization 48
Enabling destination NAT 51
Destination NAT for TCS 51
Source MAC address tracking for TCS 52
Configuring source NAT 52
Increasing the TCS hash bucket count 54
Enabling cache server spoofing support 56
Configuring the maximum connections for a cache server . . . 57
Setting cache server maximum TCP connection rates 58
Setting the cache server weight 59
Enabling FastCache 59
Enabling Remote Cache 59
Shutting down a cache server 60
Forceful shutdown on cache servers 61
Passive FTP for TCS 61
Streaming media support 65

Policy-based caching 66
Creating a set of filters using access lists 66
Configuring default cache groups 66
Configuring an ACL to bypass caching 67

Content-aware cache switching 68
How content-aware switching works. 68
Basic example of content-aware cache switching 69
Configuring policies for dynamic content 75
Bypassing embedded protocols 77
HTTP 1.1 support for cache switching 78
Cache persistence using URL hashing 80
Cache persistence using hashing on a portion of the URL. . . . 81
Supporting multiple pattern search for the same rule 86
Force rehash. 87
Traffic distribution based on cache server capacity 87
Configuring SNMP for cache server load 88
Displaying cache information 91
Sample configurations 94
Basic TCS configuration. 95
POP using caching to minimize WAN costs 97
Policy-based caching 99
Asymmetric TCS (FastCache) 100
Policy-based cache failover 102
TCS with reverse proxy. 104
High availability designs with TCS 107
Layer 3 TCS. 107
Active-standby TCS. 124
Interoperability issues with cache servers 126
CacheFlow server version 2.x.x and 3.x.x. 126
NetCache servers. 127
NetCache C720 cache server 127
CSW with NetCache cache servers. 127

About This Document

~~Audience~~

~~This document is designed for system administrators with a working knowledge of Layer 2 and Layer 3 switching and routing.~~

~~If you are using a Brocade Layer 3 Switch, you should be familiar with the following protocols if applicable to your network – IP, RIP, OSPF, BGP, ISIS, IGMP, PIM, DVMRP, and VRRP.~~

Supported hardware and software

Although many different software and hardware configurations are tested and supported by Brocade Communications Systems, Inc. for Brocade ServerIron ADX version 12.4.00 documenting all possible configurations and scenarios is beyond the scope of this document.

The following hardware platforms are supported by this release of this guide:

- ServerIron ADX 1000
- ServerIron ADX 4000
- ServerIron ADX 8000
- ServerIron ADX 1000

Document conventions

This section describes text formatting conventions and important notice formats used in this document.

Text formatting

The narrative-text formatting conventions that are used are as follows:

bold text	Identifies command names
	Identifies the names of user-manipulated GUI elements
	Identifies keywords
	Identifies text to enter at the GUI or CLI
italic text	Provides emphasis
	Identifies variables
	Identifies document titles
code text	Identifies CLI output

For readability, command names in the narrative portions of this guide are presented in bold: for example, **show version**.

Command syntax conventions

Command syntax in this manual follows these conventions:

command and **parameters**	Commands and parameters are printed in bold.
[ ]	Optional parameter.
variable	Variables are printed in italics enclosed in angled brackets < >.
...	Repeat the previous element, for example “member[;member...]”
|	Choose from one of the parameters.

Notes, cautions, and danger notices

The following notices and statements are used in this manual. They are listed below in order of increasing severity of potential hazards.

NOTE
A note provides a tip, guidance, or advice, emphasizes important information, or provides a reference to related information.

CAUTION

A Caution statement alerts you to situations that can be potentially hazardous to you or cause damage to hardware, firmware, software, or data.

DANGER

A Danger statement indicates conditions or situations that can be potentially lethal or extremely hazardous to you. Safety labels are also attached directly to products to warn of these conditions or situations.

Notice to the reader

This document may contain references to the trademarks of the following corporations. These trademarks are the properties of their respective companies and corporations.

These references are made for informational purposes only.

Corporation	Referenced Trademarks and Products
Sun Microsystems	Solaris
Microsoft Corporation	Windows NT, Windows 2000
The Open Group	Linux

Related publications

The following Brocade documents supplement the information in this guide:

- *Release Notes for ServerIron Switch and Router Software TrafficWorks 12.2.00*
- *ServerIron ADX Graphical User Interface*
- *ServerIron ADX Server Load Balancing Guide*
- *ServerIron ADX Global Server Load Balancing Guide*
- *ServerIron ADX Security Guide*
- *ServerIron ADX Administration Guide*
- *ServerIron ADX Switching and Routing Guide*
- *ServerIron ADX Installation Guide*
- *ServerIron ADX Firewall Load Balancing Guide*
- *Ironware MIB Reference Manual*

NOTE
For the latest edition of these documents, which contain the most up-to-date information, see Product Manuals at kp.foundrynet.com.

Getting technical help

To contact Technical Support, got to http://www.brocade.com/services-support/index.page for the latest e-mail and telephone contact information.

Chapter

SIP Server Load Balancing

1

In this chapter

- SIP overview 1
- SIP SLB and call persistence using ServerIron ADX 6
- Configuring SIP SLB. 11
- SIP SLB command reference 27

SIP overview

The Session Initiation Protocol (SIP) is a signaling protocol used by numerous IP communication products to create session-oriented connections between two or more endpoints in an IP network. SIP is emerging as the preferred technology for Voice over IP (VoIP) implementations.

Application-aware network switches play a vital role in increasing the uptime and availability of IP-based services such as VoIP. Many customers rely on this technology to meet mission-critical application requirements. Together with advanced SIP intelligence, ServerIron ADX switches offer a highly scalable, available, and secure load balancing infrastructure for SIP applications.

SIP is an application-layer protocol that can establish, modify, and terminate multimedia sessions, such as Internet telephony. In this implementation, ServerIron ADX SIP server load balancing balances SIP requests and responses, based on a Call-ID.

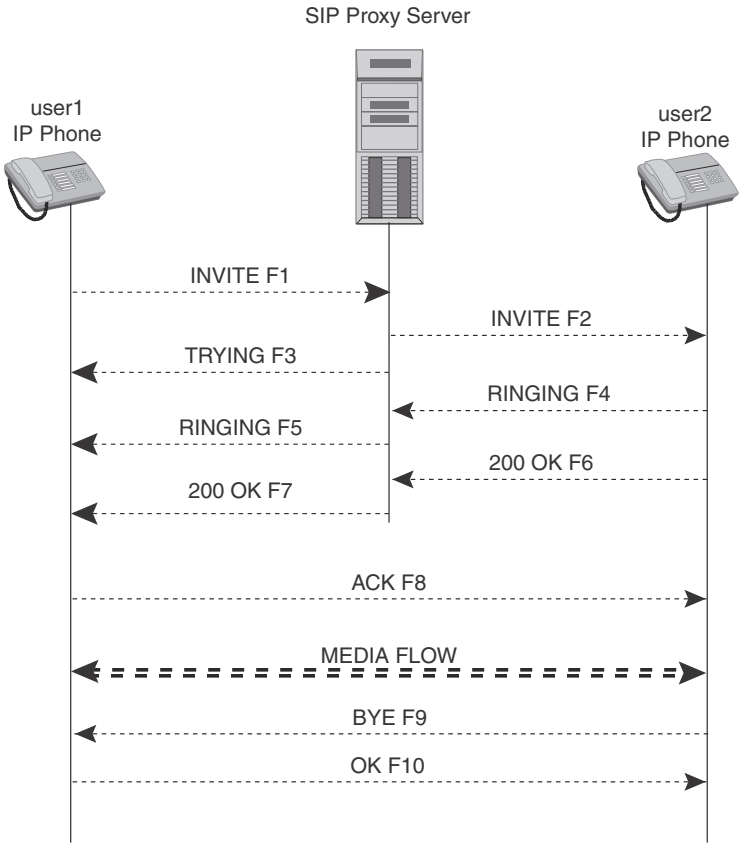
SIP Server Load Balancing is based on a request-and-response transaction model that is similar to HTTP. Each transaction consists of a request that invokes a particular method on the server, and at least one response. The method is carried within the request message.

For more information, see SIP: Session Initiation Protocol - RFC 3261.

SIP packet flow

Figure 1 demonstrates the basic operation of SIP; location of an endpoint, signal of a desire to communicate, negotiation of session parameters to establish the session, and tear-down of the session after completion.

FIGURE 1 SIP packet flow



The example in Figure 1 shows packet exchange between two SIP clients, also known as User Agent Clients (UACs). In the figure each message is labeled with the letter "F" and a number for reference. The session established between the two end-clients is facilitated by the SIP proxy server. User1 "calls" User2 using a SIP identity, a type of Uniform Resource Identifier (URI) called a *SIP URI*. The SIP URI is similar to an e-mail address, typically containing a username and a host name. In this case, it is *sip:user1@brocade.com*, where brocade.com is the domain of User1's SIP service provider.

SIP is based on an HTTP-like request-and-response transaction model. Each transaction consists of a request that invokes a particular method, or function, on the server, and at least one response. In this example, the transaction begins with User1's SIP phone sending an INVITE request addressed to User2's SIP URI. The INVITE request contains a number of header fields. The fields present in an INVITE request include a unique identifier for the call (Call-ID), the destination address, User1's address, and information about the type of session that User1 wishes to establish with User2. The INVITE (message F1 in Figure 1) would look like the following example:

```
INVITE sip:user2@brocade.com SIP/2.0
Via: SIP/2.0/UDP pcuser1.brocade.com;branch=dkDKdkDKdkDK1111
Max-Forwards: 50
To: User2 <sip:user2@brocade.com>
From: User1 <sip:user1@brocade.com>;tag=1122334455
Call-ID: 12341234123412@pcuser1.brocade.com
CSeq: 123456 INVITE
Contact: <sip:user1@pcuser1.brocade.com>
Content-Type: application/sdp
Content-Length: 142
```

Because User1's SIP phone does not know the location of User2's SIP phone, it sends the INVITE message to the SIP proxy server that is serving the brocade.com domain. The address of the brocade.com proxy server is known to the SIP phone through static configuration or through Dynamic Host Configuration Protocol (DHCP). The proxy server receives the INVITE request and sends a 100 (Trying) response back to User1's SIP phone. This response contains the same To, From, Call-ID, CSeq, and branch parameter in the Via field as the INVITE, which allows User1's SIP phone to correlate this response to the previously sent INVITE. The proxy server consults a database, generally called a *location service*, that contains the current IP address of User2. It then forwards (or proxies) the INVITE request there. Before forwarding the request, the proxy server adds an additional Via header field value with its own address (the INVITE already contains User1's address in the first Via field).

User2's SIP phone receives the INVITE and alerts User2 of the incoming call from User1; that is, User2's phone rings. User2's SIP phone indicates this by a 180 (Ringing) response, which is routed back through the SIP proxy server in the reverse direction. When User1's SIP phone receives the 180 (Ringing) response, it passes this information to User1, using an audio ringback tone.

If User2 decides to answer the call (User2 picks up the handset), the SIP phone sends a 200 OK response to indicate that the call has been answered. The 200 OK contains the Via, To, From, Call-ID, and CSeq header fields that are copied from the INVITE request, and a message body with the Session Description Protocol (SDP) media description of the type of session that User2 is willing to establish with User1. The 200 OK (message F6 in Figure 1) would look like the following example.

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP pcproxy.brocade.com
   ;branch= dkDKdkDKdkDK2222;received=172.1.1.2
Via: SIP/2.0/UDP pcuser1.brocade.com
   ;branch= dkDKdkDKdkDK1111;received=172.1.1.1
To: User2 <sip:user2@brocade.com>;tag=dkdkdk1
From: User1 <sip:user1@brocade.com>;tag=1122334455
Call-ID: 12341234123412@pcuser1.brocade.com
CSeq: 123456 INVITE
Contact: <sip:user2@172.1.1.3>
Content-Type: application/sdp
Content-Length: 131
```

The 200 OK message is routed back through the SIP proxy server to the User1's SIP phone, which then stops the ringback tone and indicates that the call has been answered. Finally, User1's SIP phone sends an acknowledgement (ACK) message to User2's SIP phone to confirm the reception of the final response (200 OK). This ACK is sent directly from User1's SIP phone to User2's SIP phone, bypassing the SIP proxy server. This occurs because the endpoints have now learned each other's IP addresses from the Contact header fields through the INVITE/200 OK exchange, which was not known when the initial INVITE was sent. This completes the INVITE/200/ACK three-way handshake used to establish SIP sessions.

The media exchange between User1 and User2 now begins using the format that they have agreed upon through SDP. In general, the end-to-end media packets take a different path from the SIP signaling messages. At the end of the call, User2 disconnects (hangs up) the phone and generates a BYE message. This BYE is routed directly to User1's SIP phone, again bypassing the SIP proxy. User1 confirms receipt of the BYE with a 200 OK response, which terminates the session and the BYE transaction. No ACK is sent. (An ACK is only sent in response to an INVITE request.)

SIP client registration

Registration is another common SIP operation. Registration is the means through which the SIP domain's registrar server learns the current location of SIP clients (UACs). Upon initialization, and at periodic intervals, the SIP clients send REGISTER messages to the domain's SIP registrar server. The REGISTER messages associate an individual SIP URI (sip:user@brocade.com) with the machine (IP address) into which the user is currently logged. The registrar server writes this association to a database, called the *location service*, where it can be used by the SIP proxy server of the domain. Often, a registrar server and the location service for a domain are co-located with the proxy server for that domain.

SIP terminology

This section describes terms and concepts that you might find useful when configuring SIP SLB.

- **Request-URI**: Every SIP user has a URI. One SIP user calls another by setting the SIP URI of the latter in the request message, also called the *request-URI*, which appears before all message headers.
- **UAC**: A User Agent Client (UAC) is a logical entity that creates a new request. The role of UAC lasts only for the duration of the transaction.
- **UAS**: A User Agent Server (UAS) is a logical entity that generates a response to a SIP request. The response accepts, rejects, or redirects the request.
- **Proxy server**: A proxy server is an intermediary entity that acts as both a server and a client for the purpose of making requests on behalf of other clients. A proxy server is primarily a router, which means its job is to ensure that a request is sent to another entity nearer to the targeted user. A proxy interprets and, if necessary, rewrites specific parts of a request message before forwarding it.
- **Redirect server**: A redirect server is a UAS that generates 3xx responses to requests it receives, directing the client to contact an alternate set of URIs.
- **Registrar server**: A registrar server accepts REGISTER requests and places the information it receives in those requests into the location service for the domain it handles.

SIP message headers

This section describes SIP message headers that you might find useful when making decisions about SIP server load balancing.

Call-ID

The Call-ID is a header field that appears in all SIP requests and responses. This header field acts as a unique identifier to group together a series of messages. It must be the same for all requests and responses sent by either the UAC or UAS in a dialog.

Call-ID is generated by the combination of a random string and the host name or IP address of a particular UAC. There is no length restriction on Call-ID. In the first implementation, a real server is selected based on the hash value of Call ID (stateless mode) or the value of Call ID (stateful mode).

- **Record-Route** : The Record-Route header field is inserted by a proxy in a request to force future requests in the dialog to be routed through the proxy; for example, Record-Route: <sip: server10.Biloxi.com; 1r>
- **From**: The From header field indicates the logical identity of the initiator of the request. It contains a URI and, optionally, a display name. This field must contain a "tag" parameter, chosen by the UAC.

  IP addresses of the host on which the UAC is running should not be used as FROM URIs, as these are not logical names.
- **To**: The To header field specifies the desired logical recipient of the request. This might not be the ultimate recipient of the request. Normally, the initial To field is set to be the value of the Request-URI. One exception is the REGISTER method.
- **Via**: The Via header field indicates the path taken by the request so far and indicates the path that should be followed in routing responses. A Via header field value contains the transport protocol used to send the message, the client's host name or network address, and possibly the port number at which it wishes to receive responses. It is a mandatory field for the UAC or UAS SIP proxies, and guarantees that the responses traverse through the same route as the requests.

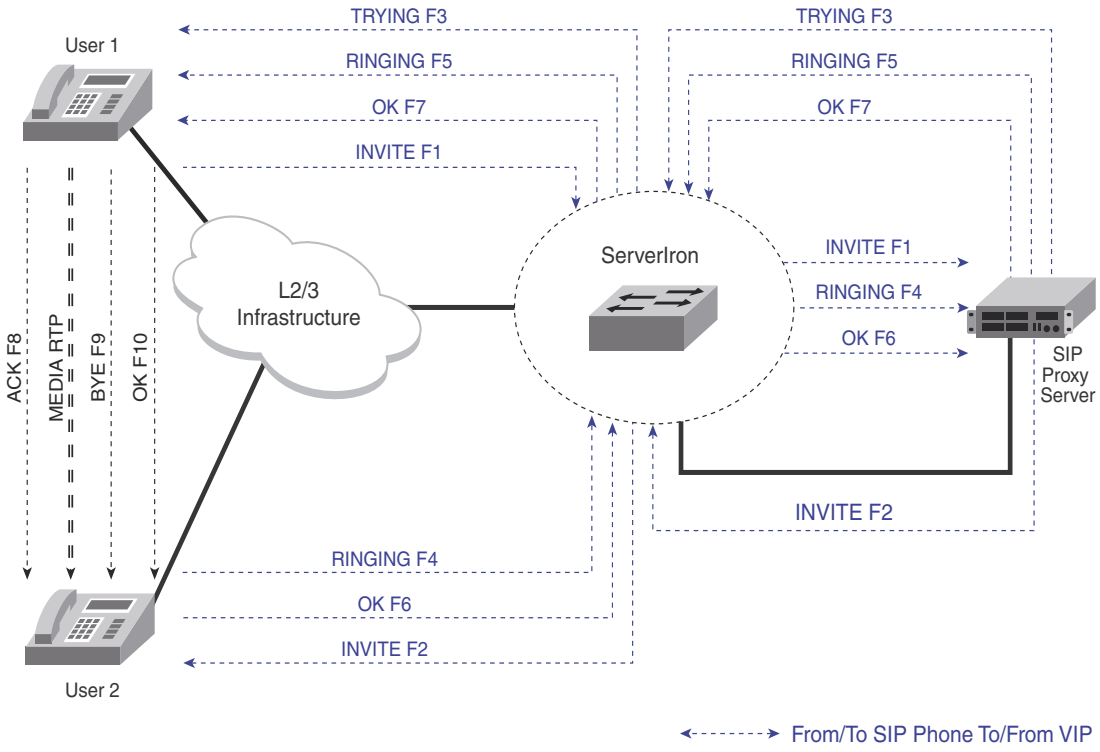
  The branch ID parameter in the Via header field value serves as a transaction identifier, and is used by proxies to detect loops.
- **Max-Forwards**: The Max-Forwards header field must be used with any SIP method to limit the number of proxies or gateways that can forward the request to the next downstream server.

  The Max-Forwards value is an integer from 1 through 255 indicating the remaining number of times that a request message is allowed to be forwarded. The recommended initial value is 70.

SIP SLB and call persistence using ServerIron ADX

Figure 2 shows an overview of a ServerIron ADX SIP server load balancing implementation.

FIGURE 2 ServerIron ADX SIP Server Load Balancing implementation



There are three kinds of SIP servers:

- Proxy server
- Redirect server
- Registrar server

In Figure 2, the ServerIron ADX SIP SLB uses the Domain-1 VIP to load balance SIP requests from Client A (user1) or Client B (user2) among Domain 1 proxy servers and registrar servers.

The SIP Server Load Balancing uses the Domain-2 VIP to load balance SIP requests from Client A (user1) or Client B (user2) among Domain 2 proxy servers and registrar servers.

The ServerIron ADX offers support for the following SIP servers in accordance with RFC 3261:

- Proxy
- Redirect
- Registrar

The ServerIron ADX supports the following methods in accordance with RFC 3261:

- INVITE
- REGISTER
- ACK

- CANCEL
- BYE
- OPTIONS

Additionally, the following methods are supported:

- SUBSCRIBE
- NOTIFY
- Other proprietary methods

SIP and call persistence specifications

The SIP server load balancing has the following specifications:

- By default, server selection is persistent on Call-ID.
- Pass-through SIP traffic from real SIP servers to SIP clients gets translated. The ServerIron ADX replaces the source IP (SIP server real IP) with Virtual IP (VIP).
- The ServerIron ADX does not modify any of the SIP header fields.
- The ServerIron ADX does not perform SIP-aware NAT.
- This implementation is based on RFC 3261.

NOTE
The ServerIron ADX SIP SLB is not implemented as a SIP proxy server, but rather as a load balancer of proxy or registrar traffic.

Sample deployment topologies

ServerIron ADX switches offer application-aware advanced intelligence for SIP server load balancing. The following sections describe some SIP server load balancing scenarios.

SIP server load balancing with DSR mode

Figure 3 shows an SIP server farm built around ServerIron ADX application switches.

FIGURE 3 SIP server farm with DSR mode

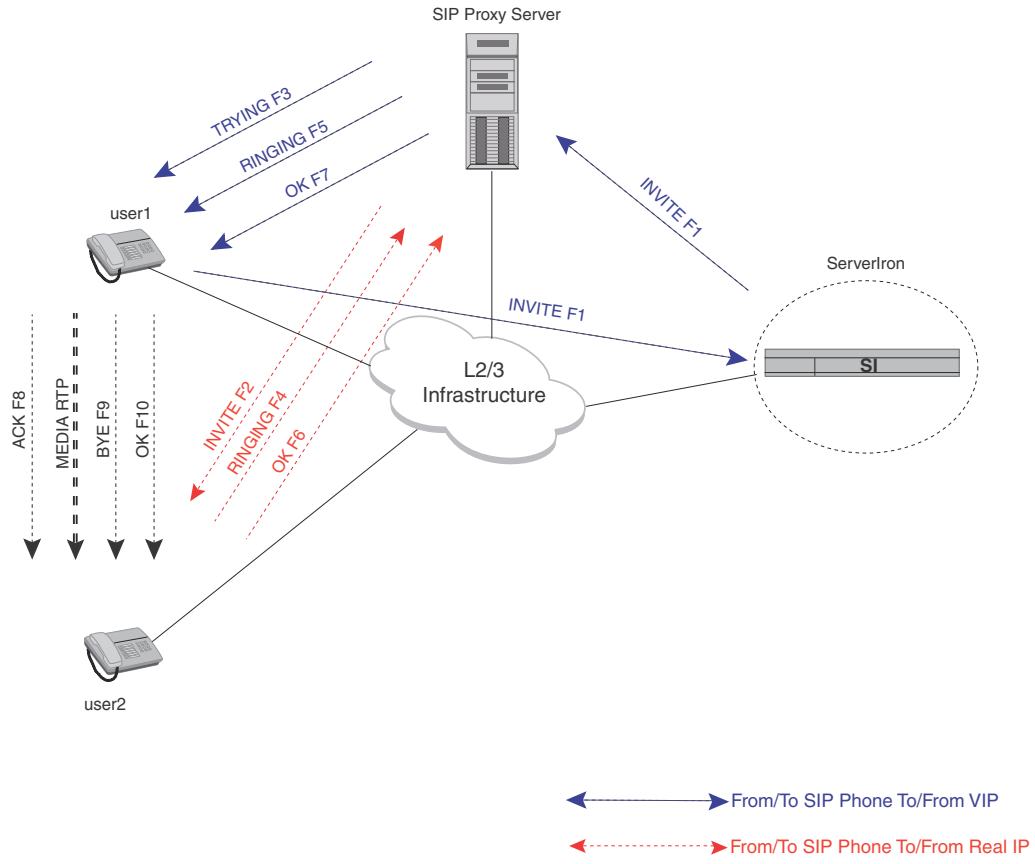


Figure 3 demonstrates a typical use case in which the ServerIron ADX application switch provides Call-ID based server persistence for UDP SIP traffic. The Call-ID attribute that uniquely identifies a SIP call is used to maintain session persistence. Due to the unique call flow requirements of SIP, most SIP implementations require you to enable Direct Server Return (DSR) mode on the ServerIron ADX switch.

Because User1's SIP phone does not know the location of User2's SIP phone, it initiates a new SIP session by sending an INVITE request to the SIP proxy server. It also generates a unique identifier (Call-ID) for the call. Because the SIP proxy server used by User1's SIP phone is actually the virtual IP address hosted on the ServerIron ADX switch, the ServerIron ADX switch receives the INVITE request and, using a server selection mechanism, identifies the *best available* SIP server for this INVITE. The ServerIron ADX uses the Call-ID attribute value to select one of the SIP servers in either stateless or stateful mode. For all SIP transactions within a dialog that use the same Call-ID, the ServerIron ADX selects the same SIP server. A new INVITE message with a different Call-ID is again subjected to server load balancing and may be forwarded to a different SIP server.

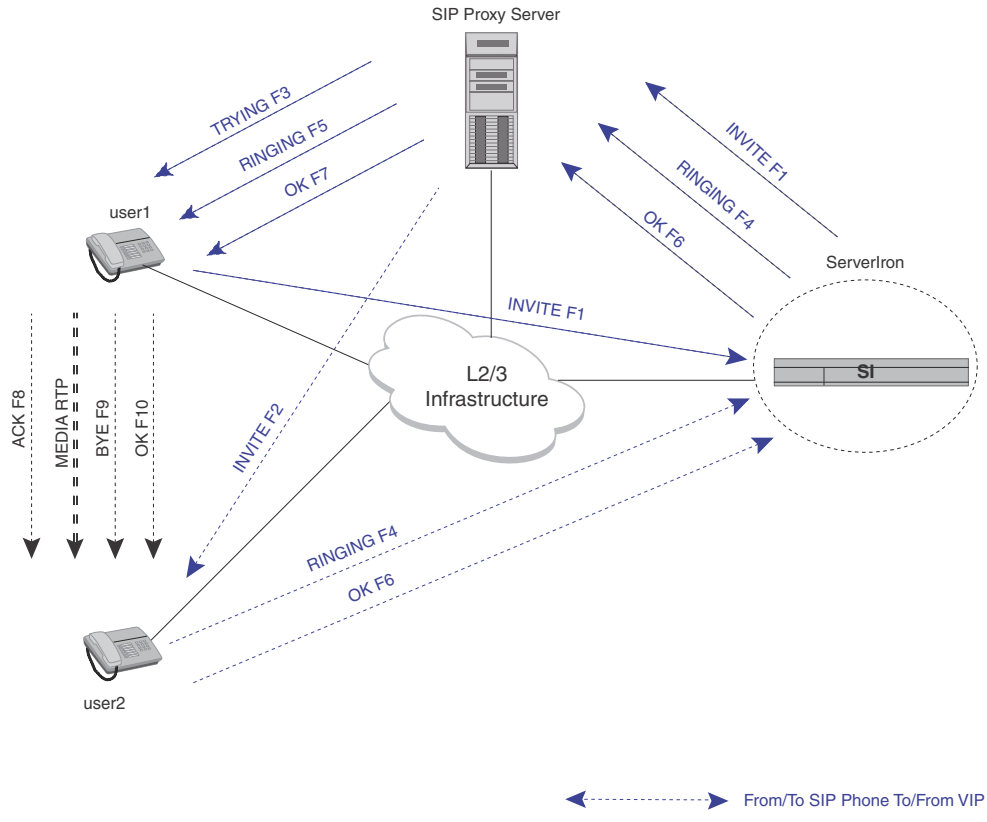
The proxy server receives the INVITE request and sends a 100 (Trying) message to User1's SIP phone. Because the ServerIron ADX switch is configured in DSR mode, the response message that is sourced from the virtual IP address flows directly to User1's SIP phone, bypassing the ServerIron ADX. The proxy server then consults the location service and forwards the INVITE request directly to User2's SIP phone, again bypassing the ServerIron ADX, and is sourced from the proxy server's own IP address.

NOTE
The proxy server's IP address must be reachable from all SIP clients.

User2's SIP phone receives the INVITE and alerts User2 of an incoming call. User2 replies with a Ringing message to the proxy server. If User2 answers the call, a 200 OK message is sent to the proxy server. The proxy server forwards this message to User1's SIP phone. Upon receiving the 200 OK message, User1's SIP phone sends an acknowledgement (ACK) message directly to User2's SIP phone, bypassing the proxy server. User1 and User2 SIP phones now begin media exchange and, upon completion, a BYE message closes the call.

Some SIP servers may be configured to use a virtual IP address (VIP) as the source address for all communications. Figure 4 shows SIP packet flows in this type of configuration.

FIGURE 4 SIP server farm with DSR mode and SIP server using VIP as source address

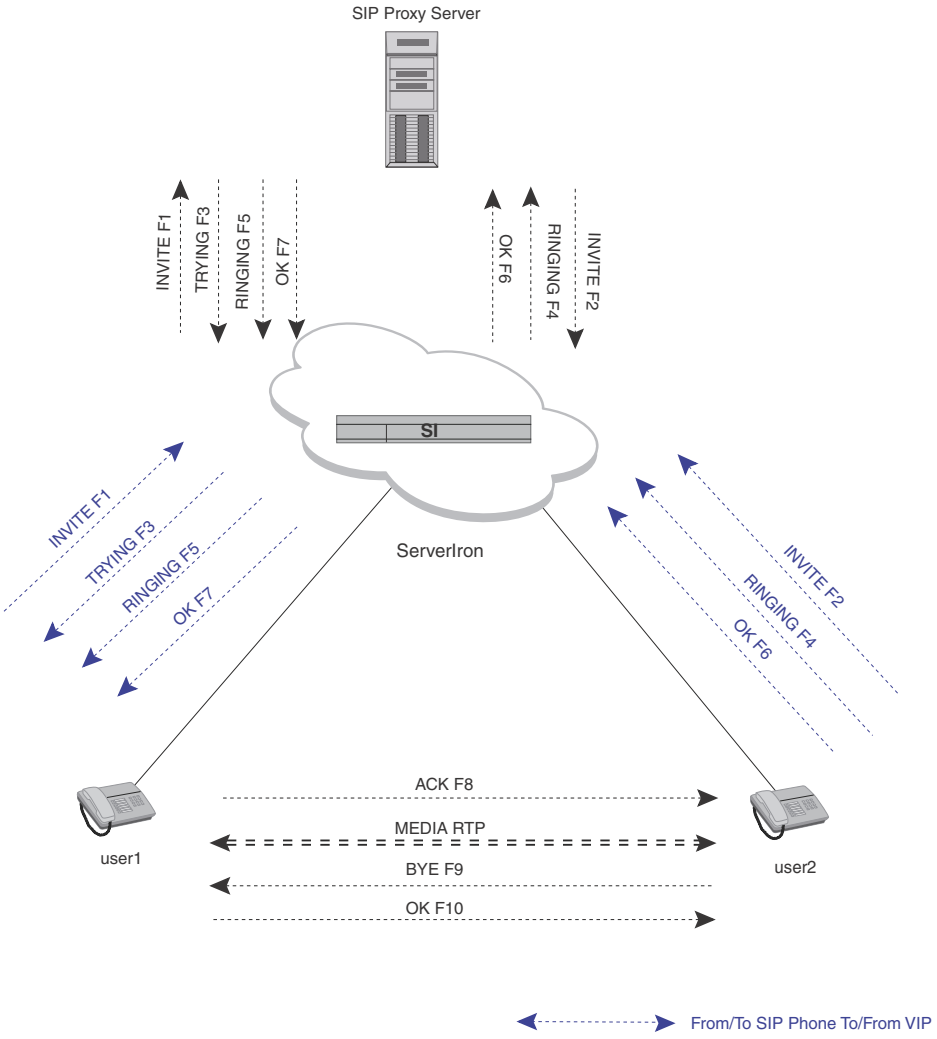


In this implementation, the SIP proxy server must use the same Call-ID for both legs of communication (the same Call-ID for message exchange with both SIP clients within a given SIP dialog). Session persistence and transaction integrity can only be achieved if the proxy server uses the same Call-ID.

SIP server load balancing with non-DSR mode

Figure 5 shows a SIP server farm with proxy servers connected inline (non-DSR mode) with the ServerIron ADX switch.

FIGURE 5 SIP Server Load Balancing with non-DSR mode



To maintain session persistence and transaction integrity, this implementation has the following requirements:

- The SIP proxy server should use the same Call-ID for both legs of communication (for example, for message exchange with both SIP clients within a given SIP dialog).
- For all outbound SIP communications, the proxy server should use the same UDP/TCP source port as that used as the destination port for all inbound communications.

NOTE
If the proxy server uses a source port other than the one used as the destination port for inbound communications, then the packets arriving from the proxy server go untranslated by the ServerIron ADX. The proxy server IP address must be reachable from all SIP clients in such cases.

SIP server health monitoring

There are two types of SIP servers of particular importance: SIP proxy servers and SIP registrar servers. The ServerIron ADX supports advanced UDP Layer-7 application health checks for both server types.

ServerIron ADX switches can be enabled to send REGISTER or OPTION messages to SIP servers to track their health. When an error-free response status (default is 200 OK) is received, then the ServerIron ADX marks the SIP server as being available, and starts assigning new SIP sessions to the available servers.

The switches can also be configured to send health-monitoring messages at user-defined frequency and retrial attempts. By default, 200 OK is considered a valid response code. Optionally, you can configure the switch to accept other response codes that indicate a healthy and available server.

SIP messages with specific SIP methods are switched to the appropriate SIP server. As an example, REGISTER messages are forwarded only to the SIP registrar server; whereas INVITE messages are distributed among SIP proxy servers.

Configuring SIP SLB

SIP server load balancing over UDP is available in stateless SLB mode and stateful SLB mode.

SIP SLB over UDP features the following elements in both stateless and stateful modes:

- Persistence parameter can be extended: The persistence parameter can be extended to other key header fields, such as the VIA header. This applies to both SIP stateful and SIP stateless modes.
- Support for fragmented UDP support: Applies to both SIP stateful and SIP stateless modes
- TCP SIP requests result in a TCP reset packet: According to RFC 3261, this will force the sender to retry the request using UDP. This applies to both SIP stateful and stateless modes.

The following elements are supported in stateful SLB mode only:

- SIP stateful support: Server selection is based on round-robin persistence on a user-configured key header, such as Call-ID, and corresponding SIP sessions are created for this purpose.
- Server initiated SIP requests handling: Sessions are created so that subsequent transactions with the same persistence parameter are directed to the same real server. This applies to SIP stateful mode only. This enables the ServerIron ADX to load balance the back-to-back user agent (B2BUA) SIP servers.
- A SIP session created in ServerIron ADX with multiple barrel processors are synched to all barrel processors (BPs).
- Redundancy support: SIP stateful supports both hot-standby and symmetric configuration by synchronizing SIP sessions to peer box. Sym-Active configuration is not recommended.
- A SIP session created in ServerIron ADX with multiple barrel processors are synched to all barrel processors (BPs).

SIP SLB over UDP (Stateless SLB mode)

The following sections discuss SIP over UDP.

Configuring a SIP proxy server and enabling health check

Complete the following steps to configure a real SIP proxy server and its health check.

1. Configure a real server and IP address for a proxy server and enter the real server configuration mode.

   ```
   ServerIronADX(config)# server real proxy-server-1 1.1.3.1
   ```

   Syntax: [no] server real <*name*> <*ip address*>

2. Specify the SIP port.

   ```
   ServerIronADX(config-rs-proxy-server-1)# port sip
   ```

   Syntax: [no] port sip

   NOTE
   You can specify SIP port number 5060 or the keyword **sip**.

3. Specify a proxy server and a health check method with options.

   ```
   ServerIronADX(config-rs-redirect-server-1)# port sip sip-proxy-server
   health-check-method options
   ```

   Syntax: port sip [sip-proxy-server] [options | health-check-method register] [health-check-no-dsr]

   - **sip-proxy-server**— Identifies the server as a SIP proxy server.
   - **health-check-method**— Specifies the SIP health check method.
   - **options**— Enables health check through OPTION messages.
   - **register**— Enables health check through REGISTER messages (default method).
   - **health-check-no-dsr**— Specifies for health check to be sent to a real server rather than a virtual server.

Configuring a SIP registrar server and enabling health check

Complete the following steps to configure a real SIP registrar server and its health check.

1. Configure a real server and IP address for a registrar server and enter the real server configuration mode.

   ```
   ServerIronADX(config)# server real registrar-1 1.1.5.1
   ```

   Syntax: [no] server real <*name*> <*ip address*>

2. Specify the SIP port.

   ```
   ServerIronADX(config-rs-registrar-1)#port sip
   ```

   Syntax: [no] port sip

3. Specify a registrar server and a health check method with no DSR. In this scenario, health check messages are sent directly to a real server IP address.

   ```
   ServerIronADX(config-rs-registrar-1)#port sip sip-registrar
   health-check-no-dsr
   ```

   Syntax: port sip [sip-registrar] [options | health-check-method register] [health-check-no-dsr]

   - **sip-registrar**— Identifies the server as a SIP registrar.

- **health-check-method**— Specifies the SIP health check method.
- **options**— Enables health check through OPTION messages.
- **register**— Enables health check through REGISTER messages (default method).
- **health-check-no-dsr**— Specifies for health check to be sent to a real server rather than a virtual server.

Configuring a SIP proxy plus registrar server and enabling health check

Complete the following steps to configure a real SIP proxy plus registrar server and its health check.

1. Configure a real server name and IP address for a registrar/proxy server and enter the real server configuration mode.

   ```
   ServerIronADX(config)# server real registrar-proxy-server-5 1.1.9.5
   ```

   Syntax: [no] server real <*name*> <*ip address*>

2. Specify the SIP port.

   ```
   ServerIronADX(config-rs-registrar-proxy-server-5)# port sip
   ```

   Syntax: [no] port sip

   NOTE
   You can specify SIP port number 5060 or the keyword **sip**.

3. Specify a registrar proxy server.

   ```
   ServerIronADX(config-rs-registrar-proxy-server-5)# port sip
   sip-both-registrar-proxy-server
   ```

   Syntax: port sip [sip-both-registrar-proxy-server] [options | health-check-method register] [health-check-no-dsr]

   - **sip-both-registrar-proxy-server**— Identifies the server as a SIP registrar server or a proxy server.
   - **health-check-method**— Specifies the SIP health check method.
   - **options**— Enables health check through OPTION messages.
   - **register**— Enables health check through REGISTER messages (default method).
   - **health-check-no-dsr**— Specifies for health check to be sent to a real server rather than a virtual server.

Configuring a SIP redirect server and enabling health check

Complete the following steps to configure a real SIP redirect server and its health check.

1. Configure a real server name and IP address for a redirect server and enter the real server configuration mode.

   ```
   ServerIronADX(config)# server real redirect-server-1 1.1.1.1
   ```

   Syntax: [no] server real <*name*> <*ip address*>

2. Specify the SIP port.

   ```
   ServerIronADX(config-rs-redirect-server-1)# port sip
   ```

   Syntax: [no] port sip

NOTE
You can specify SIP port number 5060 or the keyword **sip**.

3. Specify a redirect server and a health check method.

```
ServerIronADX(config-rs-redirect-server-1)# port sip sip-redirect-server
health-check-method register
```

Syntax: port sip [sip-redirect-server] [options | health-check-method register] [health-check-no-dsr]

- **sip-redirect-server**— Identifies the server as a SIP redirect server.
- **health-check-method**— Specifies the SIP health check method.
- **options**— Enables health check through OPTION messages.
- **register**— Enables health check through REGISTER messages (default method).
- **health-check-no-dsr**— Specifies for health check to be sent to a real server rather than a virtual server.

Configuring a SIP virtual server

Complete the following steps to configure SIP SLB virtual redirect-proxy servers and virtual proxy domains, and bind real servers to virtual servers.

1. Configure a virtual proxy domain name and IP address for Domain 1 and enter the virtual server configuration mode.

```
ServerIronADX(config)# server virtual-name-or-ip proxy-domain-1 1.1.6.9
```

Syntax: [no] server virtual-name-or-ip <*name*> <*ip address*>

2. Specify the SIP port and SIP switch.

```
ServerIronADX(config-vs-proxy-domain-1)# port sip sip-switch
```

Syntax: [no] port sip sip-switch

This command must be used to enable the SIP switch for the virtual port.

NOTE
You can specify the logical SIP port number 5060 or the keyword **sip**.

3. Configure a domain and specify a SIP domain name and dummy user.

```
ServerIronADX(config-vs-proxy-domain-1)# port sip sip-user-name sipuser
domain-name domain-1
```

Syntax: [**no**] **port sip** [**sip-user-name** <*user-name*> [**domain-name** <*domain-name*>]]

NOTE
The domain name is optional. If you do not specify a domain name, the server IP address is used.

4. Bind the real SIP registrar servers.

```
ServerIronADX(config-vs-proxy-domain-1)# bind sip registrar-1 sip registrar-2
sip
```

Syntax: bind sip <*registrar-name*> **bind sip** <*registrar-name*> **sip**

5. Bind the real SIP proxy servers.

```
ServerIronADX(config-vs-proxy-domain-1)# bind sip proxy-server-1 sip
proxy-server-2 sip
```

6. Return to global configuration mode.

```
ServerIronADX(config-vs-proxy-domain-1)# exit
```

Configuring health check

SIP health check can be performed by either the SIP REGISTER or OPTIONS method. Configure the method according to your needs. The default method is REGISTER.

To configure SIP health check correctly, you must configure the **sip-domain-name** and **dummy-user** at the virtual port level. SIP health check is only enabled at Layer 7 using UDP as the transport layer.

~~*SIP stateless sample configuration*~~

SIP SLB over UDP (stateful SLB mode)

SIP SLB over UDP makes SIP stateful and adds the intelligence needed to handle different Caller-ID situations.

SIP stateful basic configuration

1. Configure a real server.

```
ServerIronADX(config)# server real rs 2.2.2.2
ServerIronADX(config)# port sip sip-proxy-server
ServerIronADX(config)# port sip sip-both-registrar-proxy-server
health-check-method register
```

2. Configure a virtual server.

```
ServerIronADX(config)# server virtual-name-or-ip sip_vip 1.1.1.1
ServerIronADX(config)# port sip sip-stateful sip-keyfield-call-id
```

Syntax: [no] port sip sip-stateful [sip-keyfield-call-id | sip-keyfield-via]

3. Configure a SIP stateful port.

```
ServerIronADX(config)# port sip sip-stateful
```

Syntax: [no] port sip sip-stateful

4. Bind a virtual server to a real server.

```
ServerIronADX(config)# bind sip rs sip
```

Additional SIP session-specific commands

The following global configuration commands are related to SIP SLB. They will affect all virtual servers with stateful SIP SLB. The commands will not have any effect on the old stateless SIP SLB. Stateless SIP SLB will still follow the hash table and ages accordingly.

Configuring the maximum number of SIP sessions

```
ServerIronADX(config)# server sip session max-sip-sessions 1000000
```

Syntax: [no] server sip session max-sip-sessions <*sip-sessions*>

The <*sip-sessions*> variable can be from 10 through 2,000,000 sessions.

NOTE
This command requires a reload. Removing this configuration using **no** command will reset the session to 500,000.

Configuring SIP session age

```
ServerIronADX(config)# server sip session session-max-age 50
```

Syntax: [no] server sip session session-max-age <*age*>

The <*age*> variable can be from 0 through 60 minutes. The default age is 60 minutes.

Disabling partial SIP support for stateless SIP switching

```
ServerIronADX# server sip no-create-forward-l7-session
```

Syntax: [no] server sip no-create-forward-l7-session

This command does not create forward SIP sessions for the second leg of the call initiation by real servers, if the real server is bound to a SIP switch-enabled virtual server.

Clearing SIP sessions

```
ServerIronADX# clear server sip session rs1
```

Syntax: clear server sip session <*real-server*>

This command clears all SIP sessions load balanced to <*real-server*> by setting up the age to the maximum age. This command can only be accessed from the management processor (MP).

Show commands for displaying SIP sessions

NOTE
The **show sip session** command does not count SIP sessions. Use the **show sip server** command to display SIP-related real server information.

- "showing sip session info"
- "show sip server"

Showing sip session info

```
ServerIronADX# show sip session info
```

Syntax: show sip session info

This command displays information on SIP session usage.

```
ServerIronADX# show sip session info
slot-9 cup-1 Avail. SIP Ses =  0 total Sip Sessi= 0
```

Show sip server

```
ServerIronADX# show sip server
```

Syntax: show sip server

Use this command to display real server-related information.

```
ServerIronADX# show sip server

Avail. SIP sessions         =          0 Total SIP sessions      =     500000

Server State - 0: disabled, 1:enabled, 2:failed, 3:test, 4:suspect, 5:grace_dn,
6:active

Real Server   St CurrConn          TotConn    CurrSess   PeakConn

r31           6  28723             344489     28723      91687
r32           6  28722             344505     28722      91632
r33           6  28722             344549     28722      91951
r34           6  28722             344596     28722      91566
r35           6  28723             344702     28723      91870
```

~~SIP Stateful sample configuration~~

SIP SLB over TCP

Like HTTP, SIP follows a request-and-response model. However, SIP transactions are independent of the underlying transport layer protocol. For example, some transactions run over UDP connections, some get transported over TCP connections, and some use a combination of TCP and UDP interchangeably, depending on the size of the data.

Generally most SIP deployments are observed over UDP transport. However, some service providers use TCP instead of UDP to offer advanced SIP-based voice and video services. The primary reason for this method is to avoid the fragmentation experienced over UDP connections and to use the congestion avoidance mechanism of TCP.

In most cases, TCP and UDP are used interchangeably, depending on the data length. Support for TCP in addition to UDP is provided for seamless deployment of advanced SIP services. This implementation is based on RFC 3261.

Connection handling with SIP requests initiated by client

When a SIP client initiates a call using TCP, it either uses a separate TCP connection for each call or groups multiple calls together over a single TCP connection.

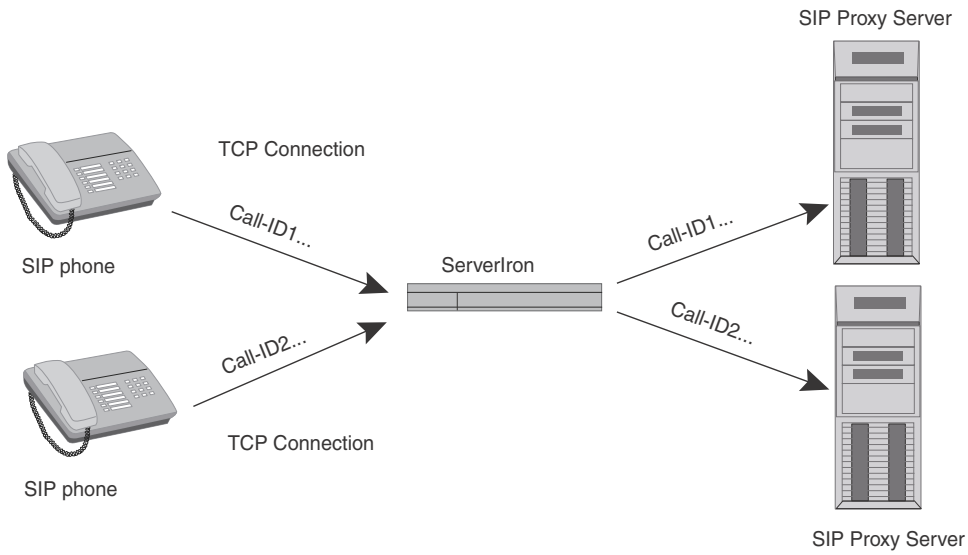
When a ServerIron ADX receives these calls over a single or multiple TCP connections, it load balances them among back-end proxy servers. The call persistence is maintained using the SIP Call-ID.

For the server-side connection, the ServerIron ADX uses either a single or multiple TCP connections with or without using source Network Address Translation (NAT) of the client IP address.

Client side: No connection reuse

Figure 6 shows that each SIP call uses a separate TCP connection. Each request has its unique Call-ID. ServerIron ADX load balances these requests among back-end SIP proxy servers.

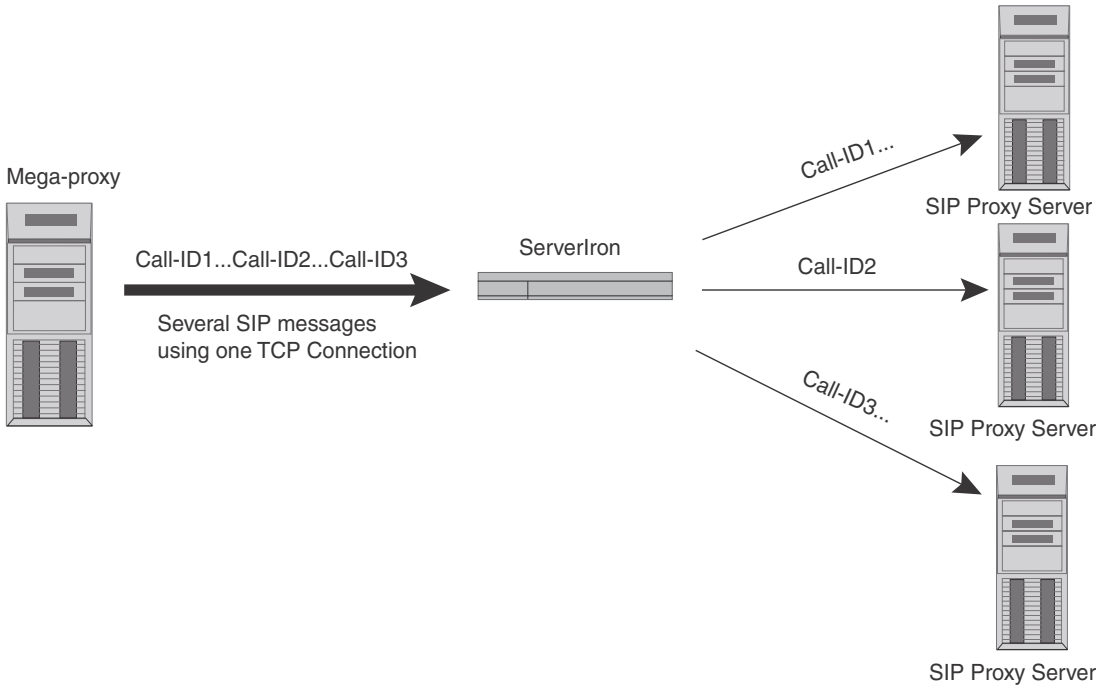
FIGURE 6 Single TCP connection for each SIP request



Client side: Connection reuse with Mega Proxy client

Figure 7 shows several SIP requests being initiated by a Mega Proxy client. The Mega Proxy uses a single TCP connection to send all these requests. Each of these requests is identified using a unique SIP Call-ID. The ServerIron ADX separates these requests and load balances them among different back-end proxy servers.

FIGURE 7 Reuse of TCP connection



Server-side connection: With source NAT

If source NAT is enabled, then the ServerIron ADX uses source NAT IP addresses and ports for establishing connections with proxy servers.

Server-side connection: Without source NAT

A ServerIron ADX may be configured with a maximum number of TCP connections to a specific proxy server.

If source NAT is disabled on the ServerIron ADX, then the ServerIron ADX establishes new TCP connections based on the connection reuse setting:

- Connection reuse disabled: If a new SIP request is received and the configured maximum number of server connections has been reached, then ServerIron ADX drops any new request. This is the default behavior.
- Connection reuse enabled: If a new SIP request is received and the configured maximum number of server connections has been reached, then ServerIron ADX reuses an established connection to process the client request.

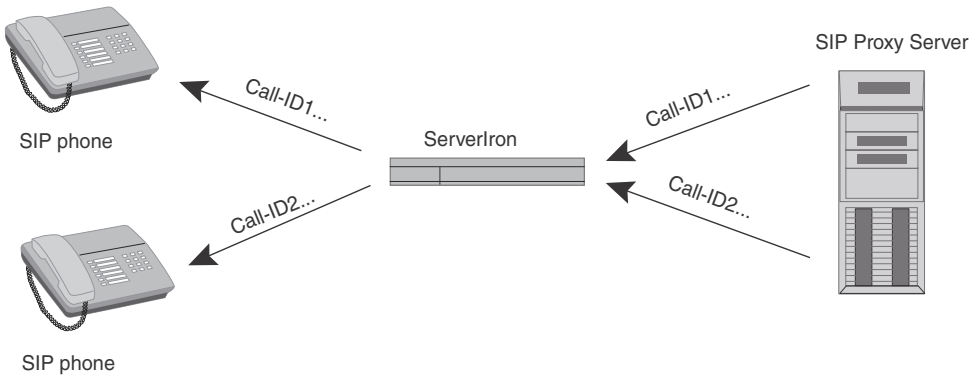
NOTE
With either setting, the client identity (IP address and port) is preserved because source NAT is not enabled.

Connection handling with SIP requests initiated by proxy server

Topologies involving back-to-back user agent (B2BUA) are also supported.

Figure 8 shows how the ServerIron ADX performs reverse source NAT on SIP server-initiated traffic.

FIGURE 8 SIP proxy server-initiated SIP requests



Load balancing modes

The ServerIron ADX can be enabled in one of the following modes while performing TCP SIP server load balancing:

- Stateless: The system maintains an internal hash table to ensure SIP call persistence.
- Stateful: The system creates session table entries to ensure SIP call persistence.

Global SIP over TCP commands

Use the following command to configure the maximum number of TCP-based client connections.

```
ServerIronADX(config)# server sip tcp max-client-tcp-connections 25
```

Syntax: [no] server sip tcp max-client-tcp-connections <*#-of-connections*>

The <#-of-connections> variable can be from 1 through 64 connections, which specifies the maximum number of concurrent TCP connections that ServerIron ADX can actively open to a specific event.

Use the following command to configure the maximum number of TCP-based server connections.

```
ServerIronADX(config)# server sip max-server-tcp-connections 25
```

Syntax: [no] server sip max-server-tcp-connections <*#-of-connections*>

The <#-of-connection> variable can be from 1 through 64 connections, which specifies the maximum number of concurrent TCP connections that ServerIron ADX can actively open to a specific server.

Use the following command to limit the maximum number of TCP transactions.

```
ServerIronADX(config)# server sip tcp max-tcp-transaction-limit 200,000
```

Syntax: [no] server sip tcp max-tcp-transaction-limit <*#-of-transactions*>

The <*#-of-transactions*> is the concurrent SIP transactions that a ServerIron ADX can manage. Enter 10 - 300,000 for <*#-of-transactions*>. The default value is 100,000.

The command should adjust the default number of concurrent SIP transactions supported on a local barrel processor (BP). Normally, the default value is more than sufficient. Changing this value is not recommended.

Real server commands for SIP over TCP

The commands in this section are entered under the SIP real server configuration level.

Use the following command to specify the maximum number of SIP connections allowed for a real server.

```
ServerIronADX(config-rs1)#sip max-tcp-connections 15
```

Syntax: [no] sip max-tcp-connections <*#-of-connections*>

The <*#-of-connections*> can be from 1 through 64 connections. The default is 64 for connection reuse cases (source NAT or non-source NAT). For non-connection reuse cases, the default value is determined by the value configured for the **max-conn** command.

NOTE
This command takes precedence over the **server sip tcp max-client-tcp-connections** and **server sip max-server-tcp-connections** (global) commands.

To enable server-side connection reuse when source NAT is not configured, enter the following command.

```
ServerIronADX(config-rs1)# sip enable-tcp-connection-reuse
```

Syntax: [no] sip enable-tcp-connection-reuse

Use the following command to specify an alternate real server IP address for the SIP real server. This is the IP address that the real server will use to send out SIP requests and responses. This configuration is required for those SIP servers that may use an IP address other than the configured real server IP address. This configuration helps a ServerIron ADX to identify the traffic as real server traffic.

```
ServerIronADX(config-rs-sip)# sip alternative-server-ip 10.120.5.34
```

Syntax: [no] sip alternative-server-ip <*alternate-ip*>

The <*alternate-ip*> variable is the alternate real server IP address.

By default, real server traffic is deeply scanned by the ServerIron ADX SIP parser. In some cases, you may want to prevent traffic from being deeply scanned because the real server initiated traffic other than SIP traffic (such as HTTP, DNS, and others). For these cases, configure an ACL for SIP. If the ACL action is "permit", packets should be forwarded without being deeply scanned by the SIP parser.

```
ServerIronADX(config-rs-sip)# sip tcp-access-list 2
```

Syntax: [no] sip tcp-access-list <*acl-id*>

The <*acl-id*> variable is the access list ID that will be used only for TCP.

Virtual server commands for SIP over TCP

Configure an alternate virtual port to be used by TCP-based SIP and the maximum local connections for a client. This port is configured under the SIP virtual server configuration level.

```
ServerIronADX(config-vs1)# port 5060 sip-alternative-port-start 5061
max-tcp-connections 3
```

Syntax: port <*sip-port*> **sip-alternative-port-start** <*start-port*> **max-tcp-connections** <*#connections*>

The <*sip-port*> is the virtual SIP port. Typically, this is port 5060.

The <*start-port*> variable specifies the beginning port of the alternate virtual port. This port will be used as the source-port of the client-side connection. The default value is the next unused port that is greater than the virtual port.

The <*#connections*> variable specifies the maximum number of local connections allowed for a client. This value takes precedence over the **server sip tcp max-client-tcp-connections** and **server sip max-server-tcp-connections** (global) commands. The default number of connections is 64.

The port range between <*start-port*> and <*start-port*>+<*max-tcp-connection*> should be reserved for alternative virtual port usage.

SIP SLB over TCP sample configuration

1. Configure a source IP address for the SIP proxy server.

   ```
   ServerIronADX(config)# server source-ip 172.28.8.3 255.255.255.0 0.0.0.0
   ```

2. Configure a SIP real server.

   ```
   ServerIronADX(config)# server real rs1 172.28.8.67
   ServerIronADX(config-rs-sip)# source-nat
   ServerIronADX(config-rs-sip)# sip max-tcp-connections 2
   ServerIronADX(config-rs-sip)# port sip
   ServerIronADX(config-rs-sip)# port sip sip-both-registrar-proxy-server
   ```

3. Configure a SIP virtual server and bind it to a SIP real server.

   ```
   ServerIronADX(config)#server virtual vs-sip 172.28.8.100
   ServerIronADX(config-vs-sip)# port sip
   ServerIronADX(config-vs-sip)# port sip sip-stateful
   ServerIronADX(config-vs-sip)# port sip sip-keyfield-call-id
   ServerIronADX(config-vs-sip)# port sip sip-alternative-port-start 5061
   max-connections 3
   ServerIronADX(config-vs-sip)# bind sip rs1 sip
   ```

Other SIP SLB over TCP options

The following section describes SIP SLB over TCP options.

Configuring periodic keepalive

Use the keepalive commands to periodically check the real server state after it is brought up. You can enable keepalive from the port profile configuration or from the real server port configuration.

SIP port profile configuration

To configure keepalive from the SIP port profile configuration, use the following commands.

```
ServerIronADX(config)# server port sip
ServerIronADX(config-port-sip)# udp keepalive protocol sip
```

Syntax: udp keepalive protocol sip

Real server port configuration

To configure keepalive from the real server port configuration, use the following commands.

```
ServerIronADX(config)# server real proxy-server-1
ServerIronADX(config-rs-proxy-server-1)# port sip keepalive
```

Syntax: port sip keepalive

Configuring a DSCP value for SIP health checks

During periods of network congestion, SIP health checks do not get high enough priority in a network, which causes servers to fail needlessly. Consequently, high-priority services such as VoIP can suffer service interruptions. This feature allows you to set a Differentiated Services Code Point (DSCP) value in the IP header of SIP health-check packets. The network can then be configured to grant sufficient priority to the SIP heath check packets to ensure robust service.

In the following example, a DSCP value of 46 is set for health check packets sent to SIP proxy service 1.1.9.5:SIP.

```
ServerIronADX(config)# server real si-server 1.1.9.5
ServerIronADX(config-rs-sip-server)# port sip sip-proxy-server
ServerIronADX(config-rs-sip-server)# port sip hc-dscp-mark 46
```

Syntax: port sip hc-dscp-mark <*DSCP-value*>

The <*DSCP-value*> variable can be set to a value from 0 through 63.

NOTE
When this command is configured for real service ports other than that of a SIP service, the value is ignored during health check. Examples include: sip-proxy-server, sip-redirect-server, sip-registrar, and sip-both-registrar-proxy-server.

Rehashing the SIP hash table

This section describes the commands for hash table management.

Manual rehash

Use the following command to manually rehash the hash table.

```
ServerIronADX(config)# server sip-hash-table-rehash vip1 sip  registrar-table
```

Syntax: server sip-hash-table-rehash <*virtual server name*> <*virtual port*> <*registrar-table* | *proxy-table*>

Automatic rehash

Use the following command to set the hash table idle time for real server replacement.

```
ServerIronADX# server sip-hash-table-idle 300000
```

Syntax: server sip-hash-table-idle <*sip-hash-table-idle-time*>

The <*sip-hash-table-idle-time*> variable is measured in seconds. The default is 1000 seconds.

NOTE
With this command, if the real server has been idling for the <*sip-hash-table-idle-time*> value or is not reachable due to health check, then it will be replaced with a new healthy real server on arrival of new packets to the hash bucket. The replacement algorithm attempts to achieve equal distribution of hash-buckets among available healthy real servers. The system will continue to assign idle or failed server hash-buckets to another healthy server until it reaches equilibrium.

NOTE
Setting the idle time to 0 redistributes the hash buckets to include new healthy real servers.

Displaying SIP real server connection rate

Use the **show server sip-conn-rate** command to display the connectivity rate for a real server.

```
ServerIronADX(config)# show server sip-conn-rate proxy-server-1
```

Syntax: show server sip-conn-rate <*real-server-name*>

Example

```
ServerIronADX# show server sip-conn proxy-server-1
Real Server: proxy-server-1            :
Real Port: sip(BOTH REGISTRAR AND PROXY SERVER):
Type:          Cur Local Rate:Last Conn Rate:Max Local Rate:TTL Conns
INVITE*              0              0/0      1               1/1
ACK                  0              0/0      0               0/0
OPTIONS              0              0/0      0               0/0
BYE*                 0              0/0      0               0/0
CANCEL*              0              0/0      0               0/0
REGISTER*            0              0/0      1               1/1
RQST UNKNOWN         0              0/0      0               0/0
RESP INFO            0              0/0      0               0/0
RSP SUCCESS          0              0/0      0               0/0
RSP REDIRECT         0              0/0      0               0/0
RESP C ERR           0              0/0      0               0/0
RSP S ERR            0              0/0      0               0/0
RESP FAIL            0              0/0      0               0/0
```

Displaying SIP virtual server connection rate

Use the **show server sip-hash proxy vs** command to display the hash table for a virtual server.

```
ServerIronADX(config)# show server sip-hash proxy-domain-1
```

Syntax: show server sip-hash <*virtual-server-name*>

Example

```
ServerIronADX# show server sip-hash proxy proxy-domain-1
SIP HashTable for virtual server :<proxy-domain-1>
Summary for virtual port <5060>:
==============================
RS|Port         #buckets  RS|Port         #bucketss
172.16.1.155|sip          0

Total active real ports bound to vport <5060>: 0
# of average buckets(proxy) for each real port: 0/256
Details for virtual port <5060>
==============================
Hash Real-Server|Port             TS Hash Real-Server|Port             TS

Total used buckets for vport <5060> :      0
```

Stateful SIP session handling in the event of a proxy server failure

ServerIron ADX can seamlessly handle failure of proxy servers while running in stateful mode. The ServerIron ADX can be enabled to reroute subsequent SIP packets on an existing flow for a failed proxy server to an available healthy proxy server. However, note that the back-end SIP proxy server should have the capability of handling such SIP calls that were originally serviced by a different proxy server.

This feature is generally useful when back-end proxy servers are configured in cluster configuration.

```
ServerIronADX(config)# server sip enable-session-failover-on-server-failure
```

Syntax: [no] server sip enable-session-failover-on-server-failure

---

NOTE
This command is not required when ServerIron ADX is configured in SIP stateless mode. With stateless mode, ServerIron ADX automatically assigns a new proxy server when packets arrive on an existing flow to failed server.

Debug commands

The following commands and the counters they display are useful for internal debugging purposes.

Showing various SIP counters

```
ServerIronADX# show sip debug parser
SIP Parser Counters:
PARSER ERR                  :1         PARSER PKT CORRUPT          :4
PARSER MEM ALLOC ERR        :0         PARSER MULTI CALLID         :0
PARSER ABNORMAL HDR END     :0         PARSER PKT HDR TOO LONG     :8
PARSER UNKNOWN PKT          :21        PARSER CONTENT TOO LONG     :0
PARSER CSEQ NOT FOUND       :1         PARSER PACKET MALFORMED     :0
```

Syntax: show sip debug [parser | session| sip-transaction | tcp-connection | udp-process]

Debugging SIP sessions

```
ServerIronADX# show sip debug session
SIP Session Debug Counters
SIP SESSION GET             :4381      SIP SESSION GET FAILURE     :0
SIP SESS FREE LIST CORUPT   :0         SIP SESS INDEX INVALID      :0
SIP SESS FINAL FREE         :84        SIP SESS DEALLOC            :4179
SIP SESS AGED               :60        SIP SESS ERR                :0
SIP SESS CORRUPT            :0         SIP HA AGE SYNC ERR         :0
SIP HA AGE SYNC RECV        :0         SIP HA AGE SYNC SENT        :0
SIP HA OUT OF IPC BUF       :0         SIP HA SESS CREAT SENT      :0
SIP HA SESS CREAT RECV      :0         SIP HA ASP SENT             :0
SIP HA DEL MSG RECV         :0         SIP HA DEL ATTEMPT RECV     :0
SIP HA DEL ATTEMPT SENT     :0         SIP HA DELETE MSG SENT      :0
SIP HA CREATE EXIST         :0         SIP HA DELETE NON-EXIST     :3910
```

Syntax: show sip debug session

Debugging SIP transactions

```
ServerIronADX# show sip debug sip-tran
SIP Transaction Counters:
TRANSACT ERR                :0         TRANSACT ENTRY GET          :10585
TRANSACT ENTRY CORRUPTED    :0         TRANSACT ENTRY GET FAIL     :0
TRANSACT INDEX INVALID      :0         TRANSACT FINAL FREED        :2828
TRANSACT FROMFLOW NOT FOUND :206       TRANSACT TOFLOW NOT FOUND   :354
TRANSACT ENTRY AGED         :3533      TRANSACT HEADER INCOMPLETE  :0
TRANSACT CONTENT ERR        :5         TRANSACT INVALID SIP HEADER :0
TRANSACT PKT TOO BIG        :0         TRANSACT PKT DROPPED        :5
TRANSACT CLIENT NOT FOUND   :0         TRANSACT RESPONSE ENTRY NOT :0
TRANSACT MSG OUTOF BOUND    :0
```

Syntax: show sip debug sip-transaction

Debugging SIP TCP connections

```
ServerIronADX# show sip debug tcp-connection
SIP Connection Counters:
TCP CONNECT ERR             :130       TCP CONNECT HOST ERR        :0
TCP CONNECT ALLOC FAILED    :0         TCP CONN BUSY               :0
TCP CONN RPORT FAILED       :5         TCP DYNAMIC BACKEND LISTEN  :0
TCP DYNAMIC FRONTEND LISTEN :0         TCP CONN PKT CHAIN ERR      :0
TCP CONN DBL FREE           :0         TCP MAX CONN REACHED        :0
TCP SESSION ERR             :0
SERVERIRON 1/1#
```

Syntax: show sip debug tcp-connection

Debugging UDP processes

```
ServerIronADX# show sip debug udp-process
SIP UDP Process Counters:
UDP ERR                     :0         UDP NO HEADER              :0
UDP UNKNOWN PKT             :0         UDP PKT TO MP              :0
UDP FWD DROP                :55        UDP REV DROP               :2
UDP NO ACTION               :0
```

Syntax: show sip debug udp-process

Debugging SIP packet traces

The **show sip debug packet-trace** command shows the packets with the IP address of either <*src-or-dest-IP-1*> or <*src-or-dest-IP-2*>. The output displays packet processing starting from parsing to forwarding. The command is very useful in debugging; however, it only displays in the BP console not on rconsole.

Syntax: show sip debug packet-trace <*src-or-dest-IP-1*> <*src-or-dest-IP-2*>

The following example is a display of "INVITE" packet processing shown on a BP console.

```
ServerIronADX4/2 # show sip debug packet-trace 172.28.8.67 172.28.8.100
4/2 #
->Start processing Incoming Flow [172.28.8.67:3798->172.28.8.100:5060] Data len
[1460]...
Start SIP Parsing for chain len [1460]...
SIP header parsing for flow COMPLETED with sip header - orig chain len [1460]
SIP packet header dump:
 Packet type: INVITE
 URI: sip:service@172.28.8.100:5060  version: SIP/2.0  response code: 0
 parse completed: 1  num pkts processed: 1  method: INVITE
 VIA header: SIP/2.0/TCP 172.28.8.67:5060;branch=z9hG4bKnashds8
 Max forward: 70
 Call ID: 1.3424.172.28.8.67@sipp.call.id
 Cseq: 1 INVITE
 Top branch id: z9hG4bKnashds8
 Top Sent By: 172.28.8.67  Top Sent By Port: 5060
In sip_process_incoming_frontend_request...
 Current transaction: branch [z9hG4bKnashds8] client[172.28.8.67]
 Forwarding request from flow [172.28.8.67:3798->172.28.8.100:5060]  to Server
[172.28.8.155:5060->172.28.8.67:3798]
```

SIP SLB command reference

This section describes the syntax and usage for each SIP Server Load Balancing command in the real server configuration mode and the virtual server configuration mode.

Real server configuration mode

Use the **port sip** command in the real server or virtual server configuration mode to configure a proxy, redirect, registrar, or registrar-proxy server. These commands are issued from the real server configuration.

Syntax: **port sip {sip-redirect-server | sip-proxy-server | sip-registrar | sip-both-registrar-proxy-server} [health-check-method [options | register]] | [health-check-no-dsr]**

- **sip-redirect-server**— Identifies the server as a SIP redirect server.
- **sip-proxy-server**— Identifies the server as a SIP proxy server.
- **sip-registrar**— Identifies the server as a SIP registrar.
- **sip-both-registrar-proxy-server**— Identifies the server as a SIP registrar or a proxy server.
- **health-check-method**— Specifies the SIP health check method.
- **options**— Enables health check through OPTION messages.
- **register**— Enables health check through REGISTER messages (default method).
- **health-check-no-dsr**— Specifies for health check to be sent to a real server rather than a virtual server.

Virtual server configuration mode

Use the **port sip** command in the virtual server configuration mode to configure. The commands are issued from the virtual server configuration.

Syntax: **port sip sip-switch | sip-domain-name** <*domain-name*>

- **sip-switch**— Enables SIP switching.
- **sip-domain-name** <*domain-name*>— Specifies SIP domain name.

Sample configuration

The following example shows the configuration details for SIP Server Load Balancing.

```
server real rs1 20.20.20.1
 port sip
 port sip sip-both-registrar-proxy-server health-check-method register
 port sip keepalive
!
server real rs2 20.20.20.2
 port sip
 port sip sip-both-registrar-proxy-server health-check-method register
 port sip keepalive
!
server virtual-name-or-ip vs1 10.10.0.100
 port sip
 port sip dsr
 port sip sip-switch
 bind sip rs1 sip rs2 sip
```

Chapter*

Transparent Cache Switching

2

In this chapter

- Transparent cache switching overview 29
- Other transparent cache switching options. 41
- Other transparent cache switching options. 41
- Policy-based caching. 66
- Content-aware cache switching 68
- Cache persistence using URL hashing. 80
- Traffic distribution based on cache server capacity 87
- Displaying cache information 91
- Sample configurations 94
- High availability designs with TCS 107
- Interoperability issues with cache servers. 126

Transparent cache switching overview

Cache servers process web queries faster and more efficiently by temporarily storing details about repetitive web queries locally, thereby reducing the number of external inquiries required to process a web query. The Brocade ServerIron ADX utilizes transparent cache switching (TCS) to distribute traffic among multiple cache servers in a local network

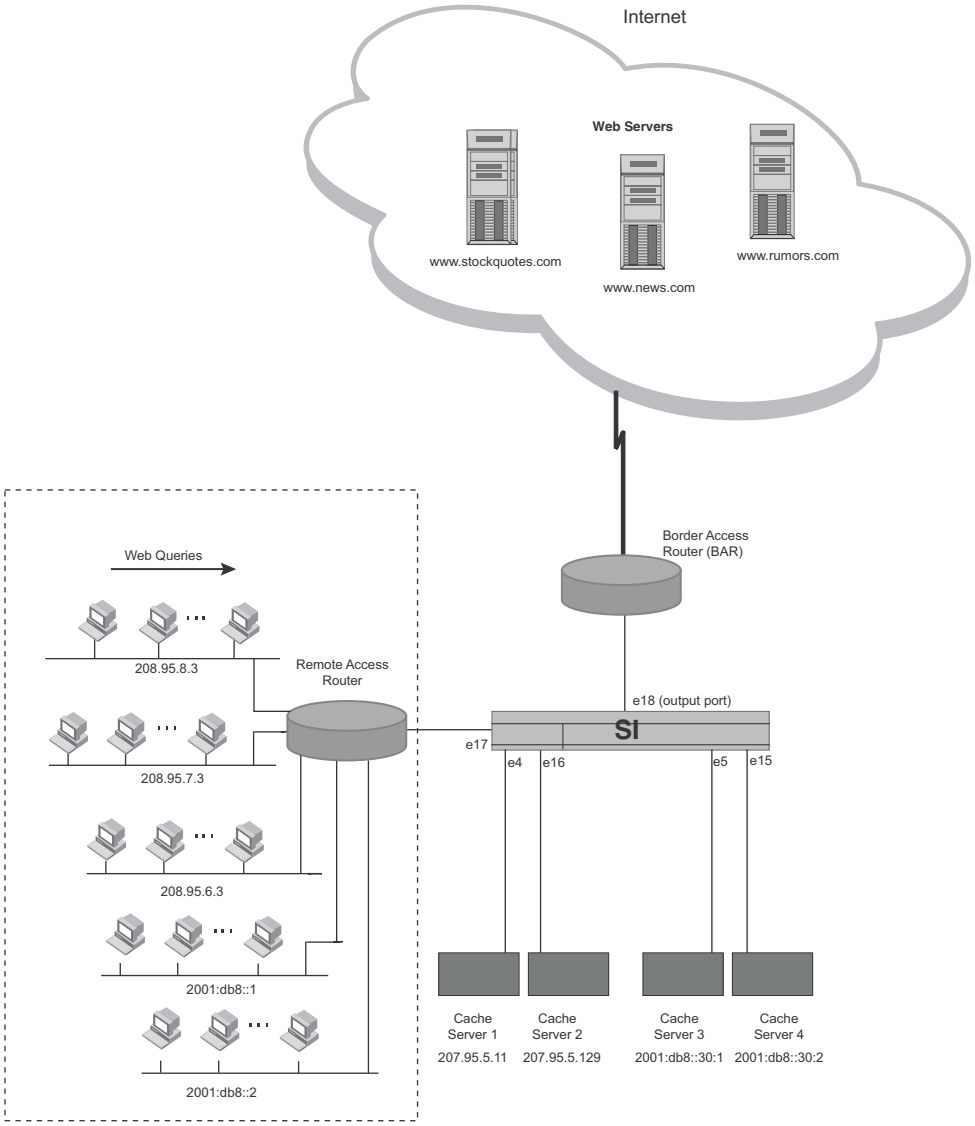
The Brocade ADX application delivery controllers increase the reliability of provider's cache infrastructure, and transparently redirect client queries to a cache farm to improve response time and overall efficiency.

ServerIron ADX supports transparent cache switching for both IPv4 and IPv6 cache servers.

Operation of transparent cache switching

A sample Transparent Cache Switching network diagram utilizing Brocade ServerIron ADX is shown below in Figure 9.

FIGURE 9 Logical representation of transparent caching



Four cache servers, server1, server2, server3, and server4, are installed within the network to handle the transparent caching of HTTP traffic. Server1 and server 2 are IPv4 cache servers and server3 and server4 are IPv6 cache servers. TCS is enabled on the ServerIron ADX to direct all HTTP traffic to the cache servers for processing.

A ServerIron ADX or backbone switch operating as a transparent cache switch detects and forwards all HTTP traffic to an available cache server. IPv4 HTTP traffic is directed to the IPv4 cache servers and IPv6 HTTP traffic is directed to the IPv6 cache servers. The cache server then processes the query and forwards the response back to the user through the attached ServerIron ADX.

The cache server determines how the web query will be handled by pulling from its local information stores and facilitating that information with external web queries on the WAN, as needed, to complete the query.

The ServerIron ADX provides the detection and switching of those HTTP packets forwarded from the cache server. This process is known as “transparent” cache switching because it is transparent to the end user. The end user continues to see the website pages as expected in answer to the query and is unaware that the access point to the information is through the cache server. Additionally, because TCS works with the default settings of web browsers, no configuration changes are required on the client station, which further adds to the transparency of the feature.

Response to cache server failures

Web cache servers are grouped with other cache servers to provide for automatic recovery from a failed or otherwise out-of-service web cache server. The ServerIron ADX monitors the availability of the cache servers in the group. If a web cache server failure occurs, the switch detects the failure and directs subsequent requests to the next available cache server or forwards the request directly to the WAN link. You can gain further reliability by using redundant ServerIron ADXs, thereby eliminating any single point of failure in the server group network path.

Stateful caching

Stateful caching provides the following services:

- Minimization of route flap problems.
- Graceful shutdown of transparent cache servers.
- Ability to set maximum connections allowed for a cache server.
- Use of access list to control caching based on source and destination addresses.
- Advanced statistics for TCS.

In stateful TCS, the ServerIron ADX creates session table entries for the client connections redirected to cache servers. The ServerIron ADX uses the return traffic as one means to assess the health of a cache server.

Minimization of route flap problems

When a route change causes web query traffic to be moved from an non-cached path to a cached path, no TCS is performed on the active connections.

NOTE
When the opposite transition occurs— web query traffic moving from a cached to non-cached path — the ServerIron ADX takes no action because the traffic is no longer visible to the ServerIron ADX.

Configurable maximum connections for cache server

You can set the maximum number of connections that a cache server will accept. By setting a limit, you can avoid a condition where the capacity threshold of a cache server is exceeded.

When a server reaches the maximum defined connection threshold, an SNMP trap is sent. When all the cache servers in a cache group reach the maximum connection threshold, the ServerIron ADX sends the client requests to the Internet.

Advanced statistics

Our TCS implementation provides the following advanced statistics:

- Current connections on a per cache basis
- Peak connections on a per cache basis
- Total connections on a per cache basis
- Packet counts to and from cache on a per-cache basis
- Octet counts to and from cache on a per-cache basis

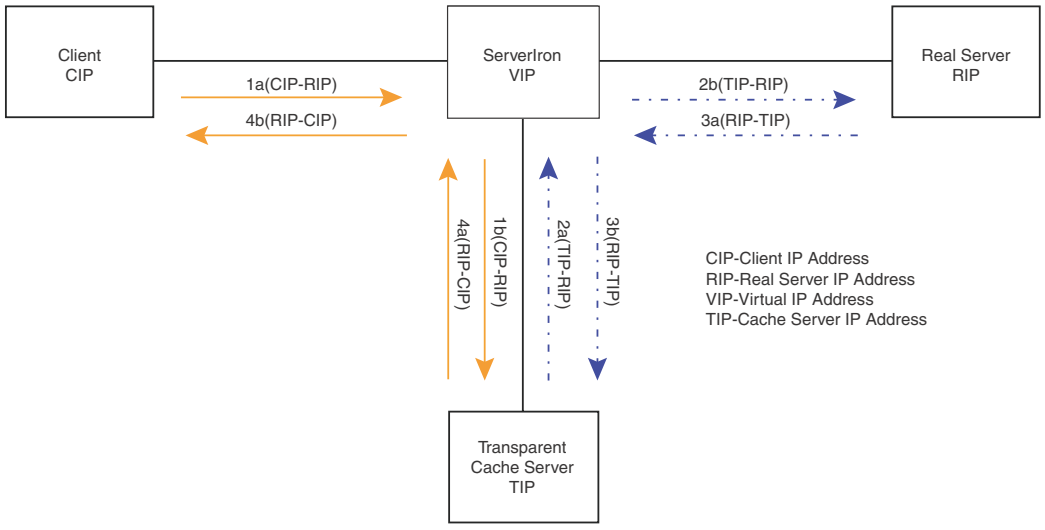
Sample deployment topologies

ServerIron ADX supports TCS in the following example topologies.

- "Basic TCS" on page 32
- "TCS with spoofing" on page 33
- "TCS with destination NAT" on page 33
- "TCS with source NAT" on page 34
- "VIPs with reverse proxy" on page 35

Basic TCS

The following example configuration shows simple TCS on the ServerIron ADX.



The above illustration shows the packet flow in a basic TCS configuration. In this example, Flow 1 is the client request getting forwarded to the cache server. If the cache server has the required information, the request is sent to the client via Flow 4. If the cache server does not have the information, it accesses the real server via Flow 2 and the traffic from the real server to the cache server comes from Flow 3.

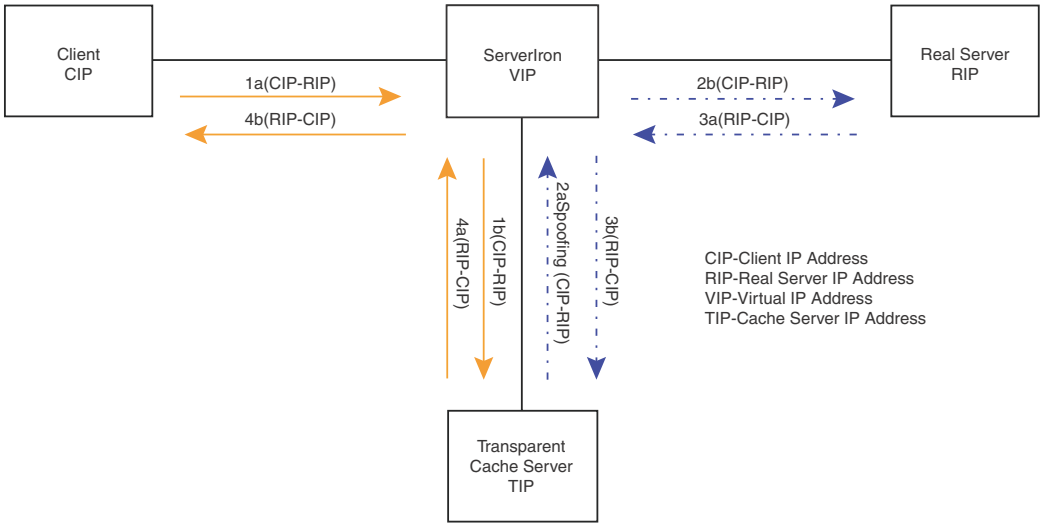
The following table lists the entries that need to be programmed in the CAM for hardware forwarding of pass-through traffic.

TABLE 1 Required CAM programming for simple TCS configurations

Level	Match	Hash
1	Destination port	Source IP address
2	Source port	Destination IP address

TCS with spoofing

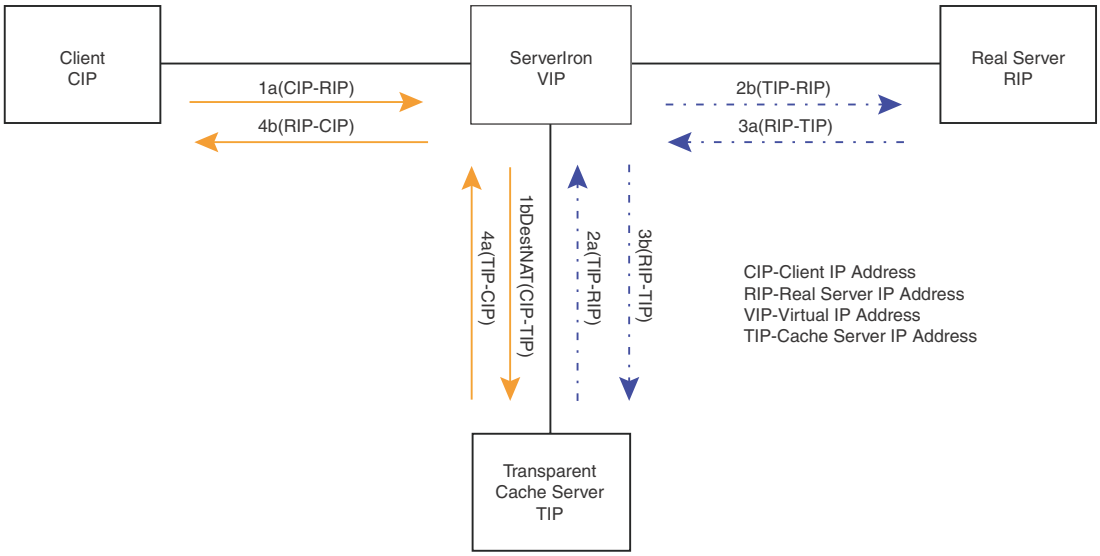
In the following example configuration, the cache server is spoofing the client's IP address instead of using its own IP address when accessing the real server.



In Flow 2a, the cache server is using the client's IP address as the source address instead of using its own IP address. There is no difference in the CAM programming for spoofing and non-spoofing cases. Refer to Table 1 for CAM programming details.

TCS with destination NAT

If the cache server is operating in the promiscuous or transparent mode, it can receive packets for any IP address. But, if the cache server requires that the client traffic arrive at the IP address of the cache server, destination NAT must be enabled on the ServerIron ADX. The following diagram illustrates this.

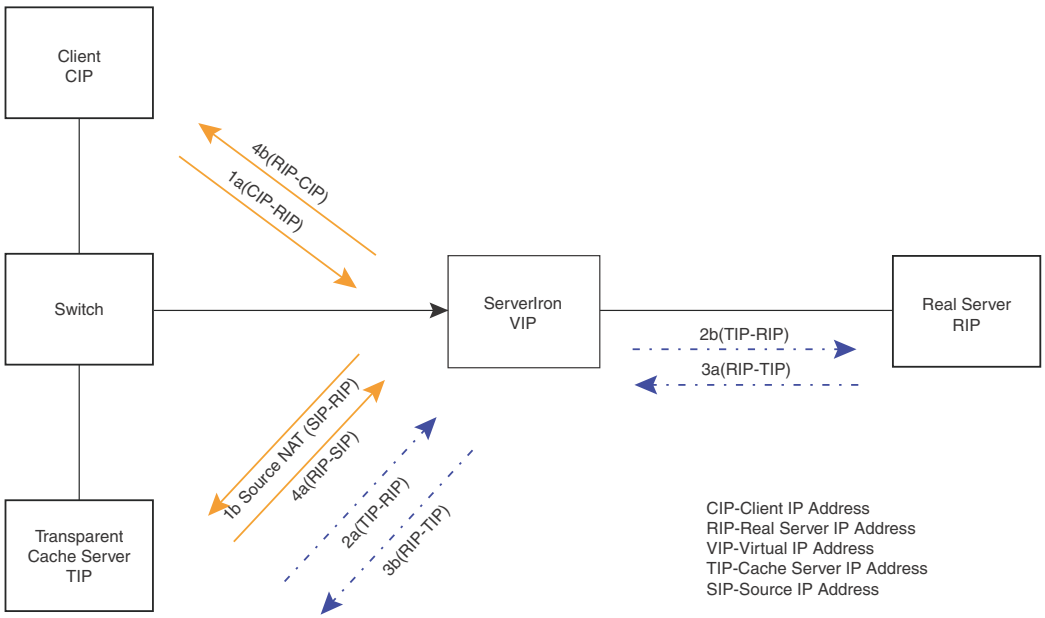


In Flow 1b, the ServerIron ADX changes the destination address in Flow 1a to that of the cache server. CAM programming is the same as for basic TCS, as detailed in Table 1.

NOTE
FTP is not supported for TCS with destination NAT.

TCS with source NAT

To make sure that the reverse traffic from the cache server hits the ServerIron ADX, source NAT may be used as shown in the following diagram:



In Flow 1b, the ServerIron ADX changes the source address to a configured IP address. The ServerIron ADX applies source NAT to the requests going from the ServerIron ADX to the cache server. Table 2 illustrates the CAM programming for this example.

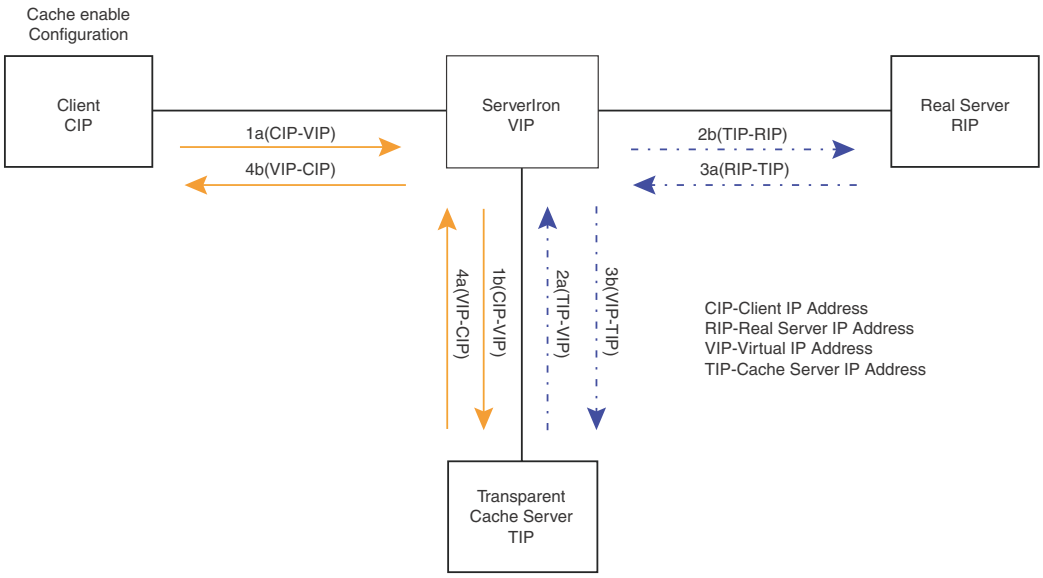
TABLE 2 Required CAM programming for simple TCS with source NAT configuration

Level	Match	Hash
1	Source NAT IP address	Destination port
2	Destination port	Source IP address
	Source port	Destination IP address

VIPs with reverse proxy

In the following example, SLB is configured and one or more of the virtual ports have reverse proxy SLB enabled (**cache-enable**). This is usually the case with server side caching topologies, when the cache server is running in transparent mode. This means that traffic destined to that particular VIP will be redirected to the cache server. If the cache server does not have the requested data, it makes a connection to the VIP, which is then load balanced across the real servers, and retrieves the data.

NOTE
FTP is not supported for TCS with source NAT.



The above diagram illustrates the packet flow in a TCS configuration with a VIP that has reverse proxy SLB enabled. Flow 1 shows the client request getting forwarded to the cache server. If the cache server has the required information, it is sent to the client via Flow 4. If the flows cache server does not have the information, it accesses the VIP via Flow 2 and the traffic from the load balanced real server to the cache server comes from Flow 3.

The following table lists the entries that need to be programmed in the CAM for hardware forwarding of pass-through traffic.

TABLE 3 Required CAM programming for VIPs with reverse proxy SLB enabled

Level	Match	Hash
1	Destination IP = cache enabled VIP	Source IP address
2	Source IP = cache enabled VIP	Destination IP address
3	Destination port = cache port	Source IP address
4	Source port = cache port	Destination IP address

Configuring transparent cache switching

Transparent cache switching (TCS) is disabled by default. To set up TCS, perform the following tasks:

1. Assign a name and IP address to each web cache server.
2. Configure port to each web cache server.
3. Configure cache server group and include previously configured cache servers under different cache groups.

NOTE
IPv4 cache servers and IPv6 cache servers must be included under different cache groups.

4. Enable TCS. Refer to "Enabling transparent cache switching" on page 40.

NOTE
You cannot enable the web cache feature on both a global (switch) and local (interface) basis.

5. Assign an interface to a cache group (optional).
6. Define distribution of web requests within a cache group (optional).
7. Modify default settings for TCS parameters (optional).
8. Save the configuration to flash.

Configuration notes

Consider the following:

- Once TCS is enabled on a switch, all ports on the switch are members of cache group 1 by default.
- You can configure up to 14 cache groups.
- Up to 256 web cache servers can be configured per cache group.
- Web cache servers must be members of a cache group.

NOTE
A single ServerIron ADX can provide transparent cache switching for up to 256 web cache servers per cache group.

- A cache group is defined in terms of input ports to the ServerIron ADX. To give a client access to a group of cache servers, the input port connecting the client to the ServerIron ADX must be in the cache group that contains the cache servers. If you plan to have only one cache group, you do not need to add the input ports to the cache group because all ports are members of cache group 1 by default.
- If you do not want a specific port to support TCS (for example, you want to redirect HTTP traffic for that port directly to the Internet instead), then you need to remove that port from default cache group 1.
- You must apply an IP policy to redirect Internet traffic to the cache servers. You can apply a global or local policy. A global policy takes effect on all ports as soon as you configure the policy. A local policy does not take effect until it is assigned to an interface. If you assign a local policy, assign it to the output port connected to the Internet. The policy sends all HTTP traffic addressed as output traffic to the port to the CPU instead for processing and forwarding.
- Although TCS can be concurrently be configured for cache servers with either IPv4 or IPv6 addresses, IPv4 traffic can only be directed to an IPv4 cache server and IPv6 traffic can only be directed to an IPv6 cache server.

Defining a cache server

Once you have enabled TCS on the ServerIron ADX, assign a name and IP address to each cache server. Once you have assigned the name and IP address, you can reference the server in CLI commands by either the server's name or its IP address.

To assign the names and IP addresses to the cache servers shown in Figure 9 on page 30, enter commands such as the following:

For IPv4

```
ServerIronADX(config)# server cache-name server1 207.95.5.11
ServerIronADX(config-rs-server1)# server cache-name server2 207.95.5.129
ServerIronADX(config-rs-server2)# end
ServerIronADX# write memory
```

For IPv6

```
ServerIronADX(config)# server cache-name server3 2001:db8::30:1
ServerIronADX(config-rs-server3)# server cache-name server4 2001:db8::30:2
ServerIronADX(config-rs-server4)# end
ServerIronADX# write memory
```

Syntax: [no] server cache-name <*text*> { <*ipv4-addr*> | <*ipv6-addr*> }

The <*text*> variable specifies a name for the server. It can be any alphanumeric string of up to 42 characters.

The <*ipv4-addr*> variable specifies the IPv4 address of the web cache server.

The <*ipv6-addr*> variable specifies the IPv6 address of the web cache server.

Identify application ports for caching

For each defined cache server you must specify the ports whose traffic you want to cache. The following example configures the previously named "server1" cache server to cache traffic from the port: "http", "ssl" and port number "8080".

```
ServerIronADX(config)# server cache-name server1
ServerIronADX(config-rs-server1)# port http
ServerIronADX(config-rs-server1)# port ssl
ServerIronADX(config-rs-server1)# port 8080
```

Syntax: [no] port <*portname or portnum*>

NOTE
A maximum of 256 non-well-known ports (port number >1024) can be configured.

NOTE
Where a non-well-known port is configured for the <*portname or portnum*> variable, a policy with port "0" needs to be configured as described in "Enabling transparent cache switching" on page 40.

Assigning web cache servers to cache groups

TCS requires all cache servers to belong to a cache group. To assign cache servers to a different cache group, use the **server cache-group** <*number*> command.

The ServerIron ADX uses one of two methods to distribute requests among the servers in a cache group: a hashing algorithm based on source and destination IP addresses or the least-connection method.

- The hashing algorithm is the default method for the HTTP and SSL protocols.
- The least-connection method is the default method for all other protocols.

The method can be changed from the default setting for all protocols. Refer to "Increasing the TCS hash bucket count" on page 54.

In addition, cache groups provide automatic recovery from a failed or otherwise out-of-service web cache server. If a web cache server failure occurs, ServerIron ADX detects the failure and directs subsequent requests to the next available cache server or forwards the request directly to the WAN link. For more information see, "csw-policy <policy-name>" on page 73 and "Redirecting client requests to an available cache server" on page 57.

To assign Cache Server 1 and Cache Server 2 to the same cache group (as in Figure 9 on page 30), create the server group and assign the servers to the group, as shown in the following.

For IPv4

```
ServerIronADX(config)# server cache-group 1
ServerIronADX(config-tc-1)# cache-name server1
ServerIronADX(config-tc-1)# cache-name server2
```

For IPv6

```
ServerIronADX(config)# server cache-group 2 ipv6
ServerIronADX(config-tc-2)# cache-name server3
ServerIronADX(config-tc-2)# cache-name server4
```

Syntax: server cache-group <*number*> [**ipv6**]

The <*number*> variable specifies an integer that identifies cache group you are creating. Up to 14 server cache groups can be assigned to a ServerIron ADX.

The **ipv6** parameter specifies that the cache group being created is for IPv6 servers.

NOTE
If IPv6 TCS is enabled, please enable the optimized mode for IPv6 (using the **ipv6 optimized-mode enable** command), save the configuration (using the **write memory** command), and reload the ServerIron ADX.

NOTE
You can gain additional reliability by using redundant ServerIron ADXs, thus eliminating any single point of failure in the network path to the web cache server group.

NOTE
While TCS can be configured to support IPv4 and IPv6 cache servers separately or concurrently,

NOTE
Define Pv4 cache servers and IPv6 cache servers in separate cache groups.

Enabling transparent cache switching

When TCS is enabled, the feature detects web traffic addressed for output to the Internet and redirects the traffic to the CPU, which processes the traffic and forwards it to the cache servers instead. TCS can be defined on either a global or local basis as described:

- **globally** – If TCS is enabled on a global basis, all ports redirect web traffic toward the cache servers. Globally assigning TCS to all ports eliminates the need to individually configure ports added in the future.
- **locally** – If TCS is enabled on a local basis only web traffic that is outbound from a specified ethernet port (or ports) will be directed toward the cache servers.

To enable TCS for HTTP traffic on all interfaces (globally) of the ServerIron ADX shown in Figure 9 on page 30, enter a command such as the following.

NOTE
The command for enabling TCS is different if you are running a switch image or a router image on the ServerIron ADX as shown.

When running a switch image

```
ServerIronADX(config)# ip policy 1 cache tcp 80 global
```

Syntax: ip policy <*index*> **cache {tcp | udp}** <*portnum*> **{ global | local }**

When running a router image

```
ServerIronADX(config)# ip l4-policy 1 cache tcp 80 global
```

Syntax: ip l4-policy <*index*> **cache {tcp | udp}** <*portnum*> **{ global | local }**

The <*index*> variable specifies the index value for the policy. This number can be any unused number from 1 – 64. Thus, up to 64 IP policies can be defined on a ServerIron ADX. You can use the **show ip policy** command to display the session policies that have been defined.

The <*portnum*> variable after TCP/UDP refers to the traffic which should be redirected to a cache server.

Where the <*portnum*> **variable is a non-wellknown-port**

To enable caching for any traffic destined to a non-well-known port (>1024), do not include the port number. Instead, configure a policy where the <*portnum*> variable is set to "0". For example, where you want to configure traffic to be redirected from port 8080 to a cache server, the <*portnum*> variable will be set to "0" as shown in the following.

```
ServerIronADX(config)# ip policy 1 cache tcp 0 global
```

NOTE
In this configuration, the policy with a <*portnum*> variable of "0" only needs to be configured once to support all non-well-known ports.

Where the **global** parameter is used, web traffic from all ports on the ServerIron ADX will be redirected toward the cache servers.

Where the **local** parameter is used, only web traffic that is outbound from a configured ethernet port will be directed to the cache servers. In this configuration, the **ip policy** command specifies that TCS is enabled locally. You must then configure each port whose outbound traffic you want to direct to the cache servers to use the previously configured IP policy.

In the following example, the **ip policy** command is configured to direct traffic locally. That IP policy is then configured under the interface configuration for ethernet port 18.

```
ServerIronADX(config)# ip policy 2 cache tcp 80 local
ServerIronADX(config)# int e 18
ServerIronADX(config-if-18)# ip-policy 2
```

Syntax: ip-policy <*policy-index*>

The <*policy-index*> variable specifies index of the IP policy that defines the web traffic that you want to direct towards the cache servers.

Other transparent cache switching options

Resetting the server cache table

You can configure the ServerIron ADX to reset the server cache table when a new cache server is added to a cache group or one cache server recovers from a failure, effectively enabling all cache servers to participate in the load balancing algorithm.

To enable the ServerIron ADX to automatically reset the server cache table when a new cache server is added to a cache group or one cache server recovers from a failure, enter the following command.

```
ServerIronADX(config)# server force-cache-rehash
```

Syntax: [no] server force-cache-rehash

Disabling a cache group or a server in a cache group

You can disable a cache group or server within a cache group to allow for maintenance.

To disable cache-group 1, enter the following commands.

```
ServerIronADX(config)# server cache-group 1
ServerIronADX(config-tc-1)# disable
```

Syntax: [no] disable

To disable a server (server2) within an active cache group, enter the following commands at the cache server level.

```
ServerIronADX(config)# server cache-name server2
ServerIronADX(config-rs-server2)# disable
```

Syntax: [no] disable

Removing or re-assigning an interface

By default, all ports (physical interfaces) on the ServerIron ADX belong to cache-group 1. An interface however, can assigned to more than one cache group.

Removing an interface from a cache group

You can remove an interface from a cache group to assign it to another cache group or to bias its traffic away from cache servers entirely.

```
ServerIronADX(config)# interface ethernet 3
ServerIronADX(config-if-3)# no cache-group 1
```

Syntax: [no] cache-group <*group-#*>

Assigning an interface to a cache group

To assign an interface to cache group, enter the following command.

```
ServerIronADX(config)# interface ethernet 3
ServerIronADX(config-if-3)# cache-group 1
```

NOTE
You must use **cache-group 1** to remove **no cache-group** command.

NOTE
You must create the cache group before you can assign an interface to the group.

Controlling traffic distribution among cache servers

To define how requests are distributed among multiple cache servers within a cache group, you can use the **hash-mask** <*destination-ip-mask*> <*source-ip-mask*> CLI command at the transparent cache level.

The ServerIron ADX uses the source and destination IP addresses as hash values. By default, the destination IP mask is 255.255.255.0, and the source IP mask is 0.0.0.0 which means only first three octets of destination IP are used to calculate the hash. All other services (known or unknown port) including SSL uses the predictor which is by default least connection.

The hash mechanism minimizes duplication of content on the cache servers by ensuring that a particular website is always cached on the same cache server.

NOTE
This is the default mechanism while performing Layer-4 TCS.

NOTE
If you configure the ServerIron ADX for server load balancing in addition to TCS, and the SLB configuration provides load balancing for your cache servers, then content will be duplicated on the cache servers as a result of the SLB predictor (load balancing metric). The SLB predictor works differently from the TCS hash mechanism and assumes that content is duplicated across the load balanced server.

NOTE
Traffic controlled by policy-based caching on an individual server level is load balanced, whereas traffic for the other cache servers is partitioned according to the hash feature. Refer to "Policy-based caching" on page 99.

NOTE
If you use content-aware cache switching (CSW in a TCS environment), URL string hashing is used to select a cache server within a server group. Content duplication is minimized because requests for cached content always go to the same cache server. Refer to "Active-standby TCS" on page 124 for more information.

Distribution algorithm

When a cache group contains multiple cache servers, the ServerIron ADX distributes traffic across the caches. The ServerIron ADX distributes the traffic using a hashing feature. The hashing feature uses a source hash mask and a destination hash mask for each cache group. The ServerIron ADX maintains a separate hash table for each cache group.

The masks determine how much of the source and destination IP addresses are used by the hash function to select a cache server.

For IPv4

The ServerIron ADX uses the following hash masks by default for IPv4 addresses:

- Destination Hash Mask: 255.255.255.0
- Source Hash Mask: 0.0.0.0

In the default hash mask, the first three octets of the destination address are significant and the source address is not significant. Therefore, traffic addressed to any of the addresses in a Class C sub-net always goes to the same cache server, regardless of the source address.

The ServerIron ADX uses the following algorithm for distributing traffic among the cache servers:

- "AND" the destination IPv4 address and destination IPv4 mask to get d1.d2.d3.d4.
- "AND" the source IPv6 address and source IPv6 mask to get s1.s2.s3.s4.
- Add each of the 1-byte values together sequentially: d1 + d2 +d3 +d4 + s1 + s2 +s3 +s4. This yields a 1-byte value that is used as the hash value.
- This 1-byte hash value is used to map to an entry in the hash table. Each entry maps to an active cache server.

For IPv6

The ServerIron ADX uses the following hash masks by default for IPv6 addresses:

- Destination Hash Mask: ffff:ffff:ffff:ffff:: (/64)
- Source Hash Mask::: (/0)

The ServerIron ADX uses the following algorithm for distributing traffic among the cache servers:

- "AND" the destination IPv6 address and destination IPv6 mask to get 16 1-byte values: d1, d2 ... d15, d16
- "AND" the source IP address and source IP mask to get 16 1-byte values: s1, s2 ... s15, s16
- Add each of the 1-byte values together sequentially: d1 + d2 + ...+ d15 + d16 + s1 + s2 + ... + s15 + s16. This yields a 1-byte value that is used as the hash value.
- This 1-byte hash value is used to map to an entry in the hash table. Each entry maps to an active cache server.

The ServerIron ADX contains 256 hash slots. If you do not assign weights to the cache servers (refer to "Setting the cache server weight" on page 59), the software divides the hash slots evenly among the cache servers. If you assign differing weights to the cache servers, the software assigns hash slots to the cache servers based on the ratios of their relative weights.

The hashing feature allows the switch to spread the traffic across the caches and minimizes duplicate data on the cache servers.

If all the cache servers become unavailable, traffic flows across the switch at Layer 2 and users go directly out to the Internet. The ServerIron ADX does not drop the traffic.

To change the hash mask, use the following command.

For IPv4

```
ServerIronADX(config)# server cache-group 1
ServerIronADX(config-tc-1)# hash-mask 255.255.255.0 255.255.0.0
```

For IPv6

```
ServerIronADX(config)# server cache-group 1
ServerIronADX(config-tc-1)# hash-mask ffff:ffff:ffff:ffff:: ffff:ffff:ffff::
```

Syntax: **hash-mask** { *<IPv4-destination-mask>* *<IPv4-source-mask>* **|** *<IPv6-destination-mask>* *<IPv6-source-mask>* }

The *<IPv4-destination-mask>* *<IPv4-source-mask>* variables specify the IPv4 destination and source masks to be used for the distribution algorithm.

The *<IPv6-destination-mask>* *<IPv6-source-mask>* variables specify the IPv6 destination and source masks to be used for the distribution algorithm.

Table 4 shows other examples of how the hash masks work.

TABLE 4 Example TCS hash masks (IPv4+IPv6)

Destination mask	Source mask	Destination IP address	Source IP address	Cache server
255.255.255.0	0.0.0.0	125.24.32.12	Any	C1
		125.24.32.210	Any	C1
		125.24.33.210	Any	C2
		125.24.34.210	Any	C3
255.255.255.192	0.0.0.0	125.24.32.12	Any	C1
		125.24.32.70	Any	C2
		125.24.32.190	Any	C3
255.255.255.0	0.0.0.255	125.24.32.12	149.165.16.233	C1
		125.24.32.12	189.12.122.233	C1
		125.24.32.12	189.12.122.200	C2
ffff:ffff:ffff:ffff::	::	2001:db8:0102:0303:6::1	Any	C1
		2001:db8:0102:0303:7::1	Any	C1
		2001:db8:0102:0304:8::1	Any	C2
		2001:db8:0102:0305:9::1	Any	C3
ffff:ffff:ffff:ffff:ffff::	::	2001:db8:0102:0303:6::1	Any	C1

TABLE 4 Example TCS hash masks (IPv4+IPv6) (Continued)

Destination mask	Source mask	Destination IP address	Source IP address	Cache server
		2001:db8:0102:0303:7::1	Any	C2
		2001:db8:0102:0304:8::1	Any	C3
ffff:ffff:ffff:ffff::	::ffff	2001:db8:0102:0303:6::1	2001:db8:0102:1	C1
		2001:db8:0102:0303:7::1	2001:db8:0304:1	C1
		2001:db8:0102:0304:8::1	2001:db8:0506:2	C2

Selecting server selection methods with cache groups

By default, the SSL and HTTP protocols use a hash method based on the source and destination IP addresses to select a cache server within a cache group. All other protocols use the least connection method by default.

To change the method of selecting a server within a cache group, enter a command such as the following:

```
ServerIronADX(config)# server cache-group 1
ServerIronADX(config-tc-1)# predictor least-connection port http
ServerIronADX(config-tc-1)# predictor hash port ftp
```

Syntax: predictor { hash | least-connection } port <*portname or number*>

The default predictor for SSL and HTTP is the **hash** method described in "Increasing the TCS hash bucket count" on page 54.

The default predictor for all other protocols is **least-connection**.

NOTE
If multiple services use the hash method to select a cache server, Brocade recommends that you group those services together using the **hc-track-group** command to maintain persistence of traffic across those services. For example, both http and https use hashing by default. To ensure that traffic persists to the same cache servers, you should group these services together. For more information on track port group health checks, see "Application port grouping" in the *ServerIron ADX Server Load Balancing Guide.*

Examples

In the following example, SSL traffic uses the hash mechanism to select a cache server which is the default action. SSL and HTTP traffic can use one hash table, This means that traffic having the same hash value will go to the same cache server whether it is SSL traffic or HTTP traffic.

NOTE
This example shows commands that are valid on the ServerIron ADX device only when it is running the Layer 3 router image.

```
ServerIronADX(config)# server cache-name cs1 192.168.0.1
ServerIronADX(config-rs-cs1)# port http
ServerIronADX(config-rs-cs1)# port http url "HEAD/"
ServerIronADX(config-rs-cs1)# port ssl
ServerIronADX(config-rs-cs1)# hc-track-group 80 443
ServerIronADX(config-rs-cs1)# exit
```

```
ServerIronADX(config)# server cache-name cs2 192.168.0.2
ServerIronADX(config-rs-cs2)# port http
ServerIronADX(config-rs-cs2)# port http url "HEAD/"
ServerIronADX(config-rs-cs2)# port ssl
ServerIronADX(config-rs-cs2)# hc-track-group 80 443
ServerIronADX(config-rs-cs2)# exit
```

In the following example, the cache selection method for the FTP protocol is changed to the hash method and it uses the same hash table as the HTTP protocol. The telnet protocol continues to use the least connection method, which is the default method for that protocol.

```
ServerIronADX(config)# server cache-name cs1 192.168.0.1
ServerIronADX(config-rs-cs1)# port http
ServerIronADX(config-rs-cs1)# port http url "HEAD/"
ServerIronADX(config-rs-cs1)# port ftp
ServerIronADX(config-rs-cs1)# port telnet
ServerIronADX(config-rs-cs1)# hc-track-group 80 21
ServerIronADX(config-rs-cs1)# exit
ServerIronADX(config)# server cache-name cs2 192.168.0.2
ServerIronADX(config-rs-cs2)# port http
ServerIronADX(config-rs-cs2)# port http url "HEAD/"
ServerIronADX(config-rs-cs2)# port ftp
ServerIronADX(config-rs-cs2)# port telnet
ServerIronADX(config-rs-cs2)# hc-track-group 80 21
ServerIronADX(config-rs-cs2)# exit
```

Resilient hashing for maximum cache persistence

In order to the maximize 'cache-hit ratio' which is a critical metric for architectures utilizing cache servers, most vendors of application delivery controllers (ADC) utilize a mechanism that provides persistence for traffic flows. When a new cache server is added to the mix for increased capacity however or when a failed cache server is recovered from a failure event, then the traffic persistency to cache is impacted. The network administrator then ends up either cleaning up the assembled intelligence on traffic persistency or newly-available cache servers are not utilized until the next available maintenance window.

Prior to software release 12.4.00a, the Brocade ServerIron ADX used a similar hashing mechanism that required re-hashing of the hash table which resulted in the loss of persistency data with the addition of new cache servers. Starting with software release 12.4.00a, the Brocade ServerIron ADX is empowered with a new resilient hashing mechanism that allows the seamless addition of newly-available cache servers while minimizing the impact on traffic persistency, which improves the cache-hit ratio.

The new mechanism also reduces the burden of full hash-table sync-up between two high-availability (HA) peer Brocade ServerIron ADX systems. Although this new mechanism doesn't completely eliminate loss of data to ensure cache persistency for new flows, it drastically reduces the impact by restricting the effect to a smaller sub-set of traffic flows.

The default settings for new resilient hashing mechanism are optimized to address common TCS deployments. There can however be situations that require modification of the default settings to improve cache re-use efficiency. Network configurations where you might want to alter the default settings include: where a deployment has a very small number of cache servers or where the IP addressing scheme of cache servers is causing non-optimal use of cache. In these situations, you can alter the default settings to see if efficiency is improved.

The two settings that can be adjusted are described below:

Hash Size: This setting controls the overall size of internal hash. The default size is 4096. The available settings are 2048, 4096, 8192 and 16384.

The following command is used to change the hash size.

```
ServerIronADX(config-tc-1)#resilient-hash hash-size 2048
```

Syntax: [no] resilient-hash hash-size <*hash-size*>

Hash Multiplier: This setting controls the granularity of hash usage against cache servers. The default value is 60. The available settings are 20, 40, 60, 80 and 100.

The following command is used to change the hash multiplier.

```
ServerIronADX(config-tc-1# resilient-hash multiplier 100
```

Syntax: [no] resilient-hash multiplier <*hash-multiplier*>

NOTE
This mechanism is available only with Layer-7 TCS. The Layer-4 TCS utilizes the hash-mechanism described in "Controlling traffic distribution among cache servers".

Cache route optimization

Typically the ServerIron ADX sits between a border access router (BAR) and a remote access server (RAS) where the BAR connects to the Internet/intranet. The RAS forwards the client HTTP traffic to the ServerIron ADX, which redirects the traffic to the cache servers. When a border router is configured as the default router for the cache servers, all traffic sent towards the browsing clients behind the RAS must first go to the BAR.

At Layer 3, the cache server sends its response to the IP address of the client (or to the ServerIron ADX if source NAT is enabled on the ServerIron ADX). However, at Layer 2, the cache server sends its response to the MAC address of its default gateway. In configurations where the default gateway is the BAR, this traffic pattern can cause significant (and unnecessary) BAR performance degradation and poor response time as perceived by the clients.

Cache route optimization (CRO) sends traffic from a cache server toward the RAS. When you enable the feature, the ServerIron ADX uses information in its Layer 4 session table and its traffic pattern recognition capabilities to redirect the traffic directly toward the clients instead of sending the traffic needlessly to the BAR.

Cache route optimization is disabled by default.

Enabling cache route optimization

Cache route optimization is useful for situations in which a cache server's default gateway is the border access router (BAR) that goes to the Internet, instead of the remote access server (RAS) that goes to the HTTP clients.

When you enable cache route optimization, the ServerIron ADX intelligently sends cache responses directly to the RAS at Layer 2 instead of sending them to the BAR for switching back through the ServerIron ADX to the RAS.

Cache route optimization is enabled on a ServerIron ADX as shown.

```
ServerIronADX(config)#server cache router-offload
```

Syntax: [no] server cache router-offload

NOTE
FTP is not supported when cache-router-offload is enabled.

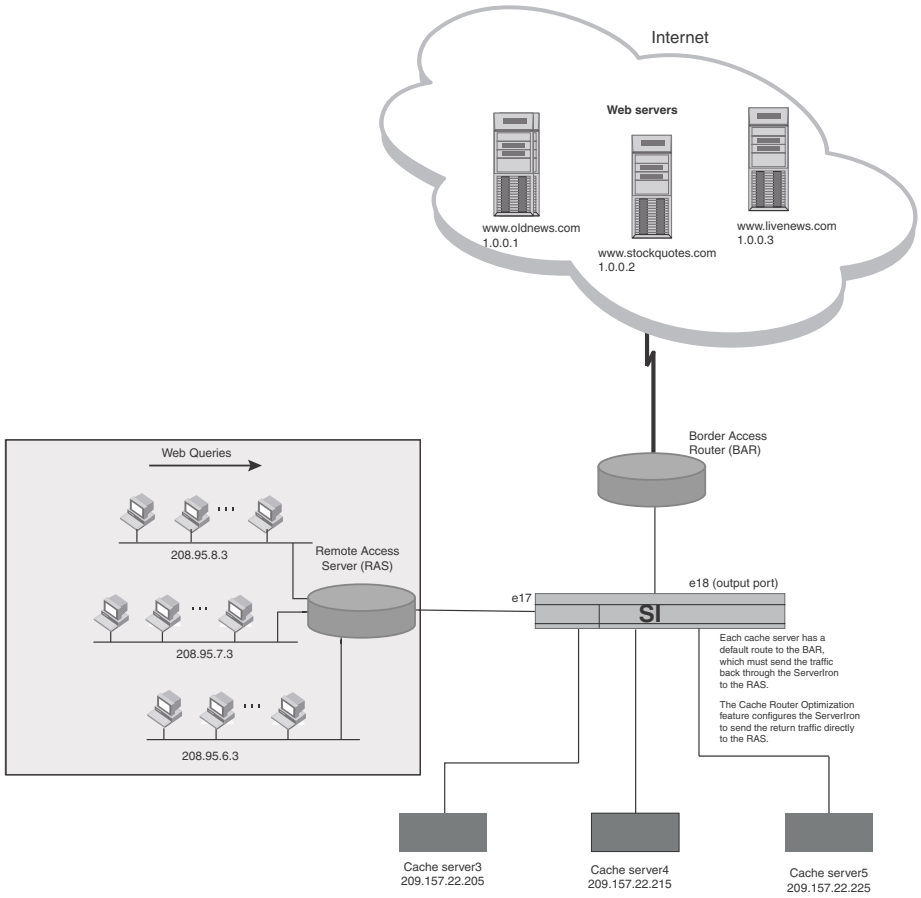
Cache route optimization example

Cache route optimization (CRO) solves a typical network topology dilemma, in which a cache server's default gateway is not the most direct route to the client. Figure 10 shows an example of a network with this topology.

In this example, return traffic from the cache servers passes through the ServerIron ADX to the border access router (BAR) because the BAR is the default gateway for the cache servers. However, the traffic is destined for the clients on the other side of the remote access server (RAS). The ServerIron ADX can switch the traffic at wire-speed, causing no perceivable response delays for the clients even if their return traffic must pass through the ServerIron ADX twice. However, the client return traffic might add noticeable overhead to the BAR, especially if the BAR is also the default gateway for other devices on the network.

You can reduce the burden on the BAR by enabling cache route optimization. This feature configures the ServerIron ADX to use the information in its Layer 4 session table to recognize that the return traffic actually should go to the RAS instead of the BAR, and send the return traffic directly to the RAS. Thus, the return traffic does not needlessly add overhead to the BAR.

FIGURE 10 Cache route optimization



Why ICMP redirects do not solve the problem

The ServerIron ADX redirects HTTP traffic destined for the Internet to the cache server. When the cache server responds to the client, it does so by sending its packets to its default gateway because the users are not in the same subnet as the cache server. However, at Layer 3, the packet is addressed to a client that is actually accessible through the remote access server (RAS). The border access router (BAR) knows the proper next hop router is the RAS, through a routing protocol, and retransmits the packet to the RAS, at Layer 2. The RAS forwards the packet to the client. Thus every packet to every client must go to the BAR and then be retransmitted. The BAR port is already carrying all the fetch and refresh traffic from that cache and this additional traffic can overload it.

- The BAR does not send an ICMP redirect in this case, as you might expect. A router sends ICMP redirects only if the following conditions are met:
- The interface on which the packet comes into the router is the same as the interface on which the packet gets routed out.

- The route being used for the outgoing packet must not have been created or modified by an ICMP redirect, and must not be the router's default route.
- The sub-net of the source IP address is the same as the sub-net of the next-hop IP address of the routed packet.
- The packet must not be source routed.
- The router must be configured to send redirects.

The third rule is violated here because caches put the web server's address in the source address field rather than the cache's address. Thus in this scenario, the packet is retransmitted to the best next hop router (the RAS) but no ICMP redirect is sent.

The ServerIron ADX solution

Cache route optimization of the ServerIron ADX is a Layer 2 mechanism that solves the problem described above. When the cache server responds to a client, the first packet is forwarded to the border access router (BAR) as discussed above. The BAR then retransmits the packet with the remote access server (RAS) as the destination MAC address and the BAR as the source MAC. The ServerIron ADX examines the packet at Layer 4. The ServerIron ADX finds a session table entry for this packet and knows it came from the cache server. The ServerIron ADX knows the packet has been re-transmitted because the packet's source MAC address isn't the cache server's MAC address and the input port isn't the cache server's port. The ServerIron ADX also recognizes that for this particular TCP session, it has seen the same packet with two different destination MAC addresses and knows that the second MAC address is the more appropriate one to use.

The ServerIron ADX contains a state table that includes a field for a MAC address. Initially this field is blank. If the ServerIron ADX sees that a packet has been re-transmitted, the ServerIron ADX places the new destination MAC address (the RAS MAC address) in the state table. When subsequent packets are sent from the cache server, the ServerIron ADX sees that there is a MAC address in the state table and replaces the destination MAC address with this address and forwards the packet.

How cache route optimization works

Each TCP connection between the cache and a client is tracked by the ServerIron ADX in a state table. The state table uses a key made up of the Layer 4 header: source IP address, source TCP port, destination IP address, and destination TCP port. The state table also has a field for a MAC address. This field is initially set to null (empty). When the cache server sends a packet a client, the ServerIron ADX examines its Layer 4 header and checks to see whether it matches an entry in the state table. The ServerIron ADX also examines the source MAC address to verify that the cache sent the packet. If the MAC address field in the state table is null, and it will be for the first packet, the ServerIron ADX simply forwards the packet at Layer 2 to the cache's default gateway, the BAR.

When the packet is re-transmitted by the BAR, the ServerIron ADX examines the Layer 4 header again, and sees that it matches an existing connection. The ServerIron ADX also examines the source MAC address to be sure the cache server sent the packet. In this case, the source MAC address is the BAR's MAC, not the cache server's. The ServerIron ADX concludes that this packet has been retransmitted and places the destination MAC address of the packet, the RAS's MAC, into the state table's MAC address field for this connection. Then the packet is forwarded to the RAS at Layer 2.

When the cache server transmits the next packet, the ServerIron ADX compares its Layer 4 header to the state table and gets a match and now the entry has a MAC address in the MAC address field. The ServerIron ADX replaces the destination address with the stored MAC address and transmits the packet at Layer 2 using the new "optimum" MAC address. Thus all packets except the first packet are sent directly to the optimum router.

Because this scheme works at the MAC layer, it is compatible with all routing protocols. Moreover, because the scheme is session specific, it can handle any number or RAS. When a session is terminated, the table entry is deleted and so is the “optimization”. Thus changes in the network at Layer 3 are immediately implemented.

Enabling destination NAT

By default, the ServerIron ADX translates the destination MAC address of a client request into the MAC address of the cache server. However, the ServerIron ADX does not translate the IP address of the request to the cache server’s IP address. Instead, the ServerIron ADX leaves the destination IP address untranslated.

This behavior assumes that the cache server is operating in promiscuous mode, which allows the cache server to receive requests for any IP address so long as the MAC address in the request is the cache server’s. This behavior works well in most caching environments. However, if your cache server requires that the client traffic arrive in directed IP unicast packets, you can enable destination NAT.

NOTE
This option is rarely used. If your cache server operates in promiscuous mode, you probably do not need to enable destination NAT. Otherwise, enable destination NAT. Consult your cache server documentation if you are unsure whether you need to enable destination NAT.

To enable destination NAT, enter commands such as the following:

```
ServerIronADX(config)# server cache-name server1
ServerIronADX(config-rs-server1)# dest-nat
```

Syntax: dest-nat

Destination NAT is disabled by default.

NOTE
FTP is not supported when destination NAT is enabled.

Destination NAT for TCS

If the cache server is remote (not directly connected to the ServerIron ADX), or is running in proxy mode (client connection is terminated completely from the back-end Internet connection), destination NAT can be configured for TCS.

When destination NAT is enabled under a cache server, the destination IP of client TCS traffic (client-to-Internet) is translated to that of the cache server (client-to-cache). The cache server then serves the web content from its cache, if available. If content is not available locally, the cache server will request content from the original destination (embedded in the HTTP host header).

When destination NAT is configured under a cache group, the destination IP of TCS traffic to all cache servers under this cache group is translated to a selected cache server's IP address. This is applicable for a remote cache server or when a cache server is running in proxy mode (not transparent mode).

Configuring destination NAT

Destination NAT can be configured either under a cache server or a cache group as shown in the following commands:

For a cache server

```
ServerIronADX(config)# server cache-name server1 207.95.5.11
ServerIronADX(config-rs-server1)# dest-nat
ServerIronADX(config-rs-server2)#
```

For a cache group

```
ServerIronADX(config)# server cache-group 1
ServerIronADX(config-tc-1)# dest-nat
```

Syntax: [no] dest-nat

Source MAC address tracking for TCS

If content is not found locally, some cache servers will initiate a connection to the Internet using the same TCP parameters as the incoming client connection (source IP:source port destination IP:destination port), which can cause a conflict with the original client connection.

For such situations, Brocade enables you to track the source MAC address of the incoming packet. If the source MAC address is that of the cache server, ServerIron ADX TCS will consider this connection as separate from the conflicting client connection.

To configure source MAC address tracking, enter the **server cache track-src-mac** command such as in the following example:

```
ServerIronADX(config)# server cache track-src-mac
```

Syntax: [no] server cache track-src-mac

If a cache server has two interfaces connected to a ServerIron ADX including one for Internet traffic, the “alias-name” must be configured as shown in the following:

```
ServerIronADX(config)# server cache-name cs1-dummy 1.1.1.2
ServerIronADX(config-rs-cs1-dummy)# exit
ServerIronADX(config)# server cache-name cs1 1.1.1.1
ServerIronADX(config-rs-cs1)# alias-cache cs1-dummy
ServerIronADX(config-rs-cs1)#
```

Configuring source NAT

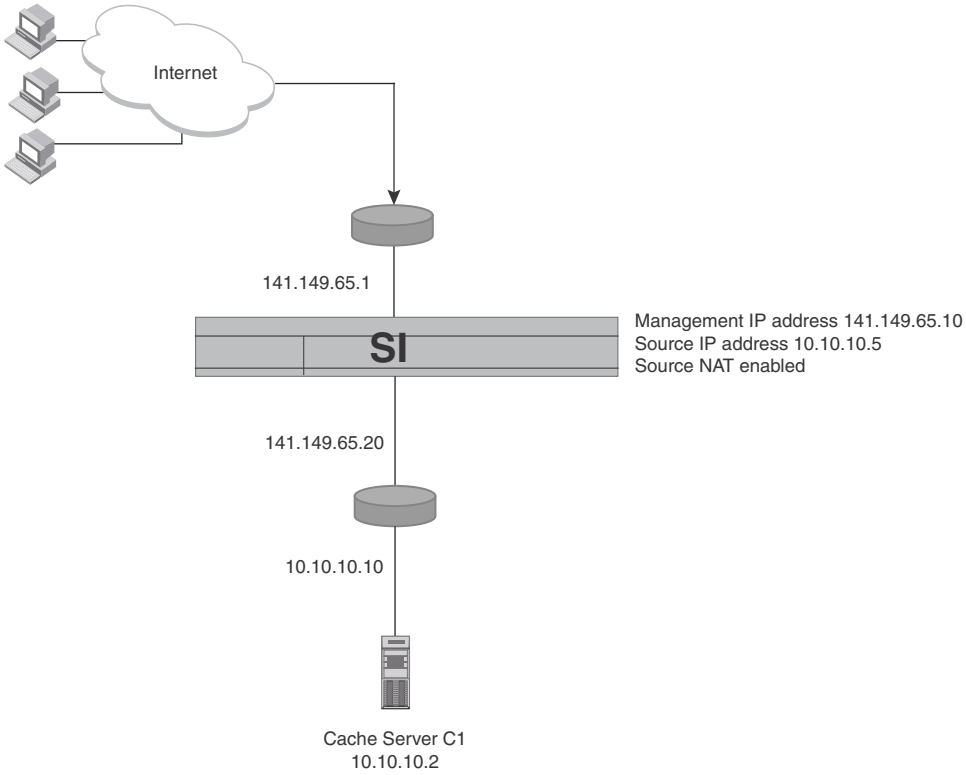
Normally, when the ServerIron ADX redirects a client’s web request to a cache server, the ServerIron ADX translates the destination MAC address of a client request into the MAC address of the cache server. However, the ServerIron ADX does not translate the source or destination IP addresses in the client’s request.

Generally, in network topologies where the ServerIron ADX and cache server are directly connected or connected through a Layer 2 switch or bridge, the cache’s response to a client query always passes back through the ServerIron ADX. The ServerIron ADX uses the cache response to assess the health of the cache server. When the ServerIron ADX passes a cache response to the client, the ServerIron ADX assumes the cache server is healthy.

However, if the time since the last packet the ServerIron ADX sent to the cache server and the cache server's response increases significantly, or the cache server's reply never reaches the ServerIron ADX but instead takes an alternate path to the client, the ServerIron ADX assumes that the cache server has stopped responding. When this occurs, the ServerIron ADX marks the cache server FAILED and stops redirecting client queries to the cache server.

You can ensure that cache server replies always pass back through the ServerIron ADX by configuring source NAT.

FIGURE 11 Using source NAT with TCS



In this example, the ServerIron ADX and cache server are connected by a router and are in different sub-nets. In a topology where the cache server's response is guaranteed to pass back through the ServerIron ADX, you may not need to configure source NAT. However, if the cache server's reply can reach the client by a path that does not pass through the ServerIron ADX, you need to configure source NAT.

To configure source NAT:

- Enable the source NAT feature. You can enable the feature at the cache group level for all cache servers or at the cache server level for individual servers.
- Configure a source IP address. A source IP address allows the ServerIron ADX to be a member of more than one sub-net. If the cache server and ServerIron ADX are in different sub-nets, configure a source IP address that is in the cache server's sub-net.

To enable source NAT globally for all cache servers and configure a source IP address, enter commands such as the following:

```
ServerIronADX(config)# server source-ip 10.10.10.5
```

```
ServerIronADX(config)# server cache-group 1
ServerIronADX(config-tc-1)# source-nat
ServerIronADX(config-tc-1)# dest-nat
```

These commands configure a source IP address at the global CONFIG level of the CLI, then change the CLI to the cache group configuration level and enable source NAT and destination NAT. Source NAT configures the ServerIron ADX to change the source IP address in a client query from the client's IP address to configured source IP address. Destination NAT configures the ServerIron ADX to change the destination IP address of the client's request to the IP address of the cache server.

Syntax: [no] source-ip <*ip-addr*> <*network-mask*> <*default-gateway*>

NOTE
The gateway parameter is required. If you do not want to specify a gateway, enter "0.0.0.0".

Syntax: [no] source-nat

To enable source NAT on a specific cache server instead of at the cache group configuration level for all cache servers, enter commands such as the following:

```
ServerIronADX(config)# server cache-name C1
ServerIronADX(config-rs-C1)# source-nat
ServerIronADX(config-rs-C1)# dest-nat
```

The commands in this example enable source NAT and destination NAT on cache server C1 only. This example assumes that the source IP address also is configured as shown in the previous example.

NOTE
FTP is not supported when source NAT is configured.

Increasing the TCS hash bucket count

The ServerIron ADX supports 256 hash buckets by default. If you do not assign weights to the cache servers, the software divides the hash buckets evenly among the cache servers. With the default hash bucket count of 256, there is a limitation of traffic distribution. In a setup with a large number of cache servers, if one of the cache servers fails, then their remaining cache servers may get hit by traffic spikes due to limited load balancing by hash buckets.

Consider an example where there are 64 cache servers within one cache group (CS1 through CS64). Because there are 256 buckets, each server is assigned four buckets (256 / 64 = 4). If one cache server (CS1), goes down, the four buckets assigned to CS1 are re-assigned to cache servers "CS2-CS5". Consequently, "CS2-CS5" have five buckets each while CS6 through CS64 still have four buckets. This means that the original traffic handled by the cache server going down is not distributed evenly among the rest of the cache servers. The traffic on CS2 through CS5 increases by 25 percent ([5 - 4] / 4= 25%).

You can increase the TCS hash bucket count to a higher number to ensure a more reasonable distribution of excess traffic among remaining cache servers when a cache server goes down.

Using the previous example in which there are 64 cache servers(CS1-CS64) you can upgrade the TCS hash bucket count to 8192. Because 8192 / 64 = 128, each server is now assigned 128 buckets. If one cache server (CS1) goes down, then the 128 buckets assigned to CS1 are re-assigned to the other 63 servers. In this situation, CS2 and CS3 get three additional buckets

each while CS4 through CS64 get two buckets each ([2 * 3] + [61 * 2] = 128). The result is that the original traffic handled by the cache server that went down is now distributed evenly among the remaining functional cache servers. The traffic on CS2 and CS3 increases by 2.34 percent ([131 - 128] / 128 = 2.34%), and by 1.56 percent on CS4 through S64 ([130 - 128] / 128 = 1.56%).

Configuring an increased hash bucket count

You can set the size of the TCS hash bucket count to a value from 256 through 16384 using the **system-max** command.

```
ServerIronADX(config)# system-max tcs-hash-table-size 2048
```

Syntax: [no] system-max tcs-hash-table-size <*hash-table-size*>

The <*hash-table-size*> variable specifies the size that you want to configure the hash table. Acceptable values are 256, 512, 1024, 2048, 4096, 8192, and 16384. Entering a value for this variable other than an acceptable value will cause an error message to be displayed. The default value is 256.

NOTE
The hash table size will only be effective after saving the configuration using the **write mem** command and reloading the ServerIron ADX.

Displaying the hash values per BP

You can display the hash values for a specific source and destination IP pair. The output first displays the current hash table size, then the hash value which is the bucket number for the given destination and source IP addresses. If a cache server is selected for this bucket, the selected cache server name is displayed and if no cache server is selected for this bucket, "empty" is displayed, as shown in the following examples.

```
ServerIronADX1/1# show cache-hash 1 1.2.3.4 192.168.1.1
Cache-group 1:          Hash table size:   512
Hash_info: Dest_mask = 255.255.255.255 Src_mask = 0.0.0.0
  bucket#:    38 -> cs1

ServerIronADX1/1# show cache-hash 1 1.2.3.4 192.168.1.1
Cache-group 1:          Hash table size:   512
Hash_info: Dest_mask = 255.255.255.255 Src_mask = 0.0.0.0
  bucket#:    39 -> empty
```

Syntax: show cache-hash <*cache-group*> <*destination-IP*> <*source-IP*>

NOTE
This command is available at the BP console only.

NOTE
The hash table is not synced between ServerIron ADX switches for both Hot-Standby and Active-Active mode.

Enabling cache server spoofing support

In TCS, when a client makes a request for HTTP content on the Internet, the ServerIron ADX directs the request to a cache server, rather than to the Internet. If the requested content is not on a cache server, it is obtained from a Web server of origin on the Internet, stored on a cache server to accommodate future requests, and sent from the cache server back to the requesting client.

NOTE
You cannot use the cache server spoofing feature with the reverse proxy SLB feature on the same ServerIron ADX.

When a cache server makes a request for content from the origin server, it can perform one of the following actions:

- The cache server replaces the requesting client's IP address with its own before sending the request to the Internet. The origin server then sends the content to the cache server. The cache server stores the content and sends it to the requesting client, changing the source IP address from its own to the origin server's IP address.
- The cache server does not replace the requesting client's IP address with its own. Instead, the cache server sends the request to the Internet using the requesting client's IP address as the source. This allows the origin server to perform authentication and accounting based on the client's IP address, rather than the cache server's IP address. This functionality is known as *cache server spoofing*.

When cache server spoofing support is enabled, the ServerIron ADX does the following with requests sent from a cache server to the Internet.

1. The ServerIron ADX looks at the MAC address to see if the packet is from a cache server. Note that the ServerIron ADX and the cache server cannot be separated by any router hops; they must be on the same physical segment. The ServerIron ADX uses an ARP request to get the MAC address of each configured cache server.
2. If the MAC address indicates that the packet is from a cache server, the ServerIron ADX checks the source IP address. If the source IP address does not match the cache server's IP address, the ServerIron ADX concludes that this is a spoofed packet.
3. The ServerIron ADX creates a session entry for the source and destination (IP address and port) combination, and then sends the request to the Internet.

When the origin server sends the content back, the ServerIron ADX looks for a session entry that matches the packet. If the session entry is found, the ServerIron ADX sends the packet to the appropriate cache server.

To enable cache server spoofing support, enter commands such as the following.

```
ServerIronADX(config)# server cache-group 1
ServerIronADX(config-tc-1)# spoof-support
```

Syntax: [no] spoof-support

The **no** form of the command disables cache server spoofing support. Cache server spoofing support is disabled by default.

To display the number of spoofed packets encountered by the ServerIron ADX, enter the following command.

```
ServerIronADX# show cache-group

Cache-group 1 has 1 members Admin-status = Enabled Active = 0
Hash_info: Dest_mask = 255.255.255.0 Src_mask = 0.0.0.0

Cache Server Name               Admin-status L4-Hash-Buckets L7-Hash-Buckets
cs1                             6            0               0


Name: cs1             IP: 192.168.1.1          State: 6   Groups =   1

Spoof Enable                                Cache->Web-Server    Web-Server->Cache
                    CurCon TotCon           Packets   Octets     Packets   Octets
Total               0      0                0         0          0         0

                                            Client->Cache        Cache->Client
           State    CurCon TotCon           Packets   Octets     Packets   Octets
http       active   0      0                0         0          0         0
Total               0      0                0         0          0         0
```

Syntax: show cache-group

Configuring the maximum connections for a cache server

You can limit the maximum number of connections supported on a server-by-server basis. By setting a limit, you can avoid a condition where the capacity threshold of a cache server is exceeded.

When a cache server reaches the maximum defined connection threshold, the ServerIron ADX sends an SNMP trap. When the cache server reaches its maximum connection threshold, the ServerIron ADX sends client requests to the Internet by default. Optionally, you can direct the ServerIron ADX to send client requests to another available cache server that has not reached its maximum connection threshold.

Up to 1,000,000 sessions are supported. This is the default.

To limit the connections to a maximum of 100,000 for cache server1 and 200,000 for server2 in the network seen in Figure 9 on page 30, enter the following commands.

```
ServerIronADX(config)# server cache-name server1
ServerIronADX(config-rs-server1)# max-conn 100000
ServerIronADX(config-rs-server1)# server cache-name server2
ServerIronADX(config-rs-server2)# max-conn 200000
ServerIronADX(config-rs-server2)# end
ServerIronADX# write mem
```

Syntax: max-conn <*connections-supported*>

The <*connections-supported*> variable specifies the number of connections supported. Acceptable values are from 1 to 2000000.

Redirecting client requests to an available cache server

When the limit specified in the **max-conn** command is reached for a cache server, the default behavior is that the ServerIron ADX redirects any traffic intended for this cache server (according to the allocated hash distribution) directly to the Internet.

Optionally, you can direct the ServerIron ADX to send client requests to another available cache server in the same cache group that has not reached its maximum connection threshold without changing the current hash distribution. Use the **reselect-server-if-overloaded** command under the cache group configuration as shown in the following example.

```
ServerIronADX(config)# server cache-group 1
ServerIronADX(config-tc-1)# reselect-server-if-overloaded
```

Syntax: [**no**] **reselect-server-if-overloaded**

Setting cache server maximum TCP connection rates

You can modify the maximum connection rate for for individual TCP/UDP ports.

Configuring the connection rate control

Connection rate control enables you to limit the connection rate to a cache server. The ServerIron ADX limits the number of new port connections per second to the number you specify.

The ServerIron ADX increments the connection counter for cache connections only after the ServerIron ADX selects one for the connection. If the ServerIron ADX cannot serve a client request because a cache already has the maximum number of connections for the current second for the requested port, the ServerIron ADX tries another cache. If there are no caches available, the ServerIron ADX directs the request to the Internet.

If you configure a limit for TCP and also for an individual application port, the ServerIron ADX uses the lower limit. For example, if you limit new TCP connections to a real server to 1000 per second and also limit new HTTP connections to 600 per second, the ServerIron ADX limits connections to TCP port HTTP to 600 per second.

NOTE
The ServerIron ADX counts only the new connections that remain in effect at the end of the one-second interval. If a connection is opened and terminated within the interval, the ServerIron ADX does not include the connection in the total for the server.

To limit the number of new TCP connections a cache can receive each second, enter commands such as the following.

```
ServerIronADX(config)# server cache-name C1 5.6.7.8
ServerIronADX(config-rs-C1)# max-tcp-conn-rate 2000
```

Syntax: **max-tcp-conn-rate** <*num*>

The <*num*> variable specifies the maximum number of connections per second. There is no default.

Syntax: **port** <*TCP/UDP-portnum*> **max-tcp-conn-rate** <*num*>

The <*TCP/UDP-portnum*> variable specifies the application port.

The <*num*> variable specifies the maximum number of connections per second.

You also can specify the connection rate for an individual port.

```
ServerIronADX(config)# server cache-name C1 5.6.7.8
ServerIronADX(config-rs-C1)# port http
ServerIronADX(config-rs-C1)# port http max-tcp-conn-rate 2000
```

Setting the cache server weight

You can assign a performance weight to each server. Servers assigned a higher weight receive a larger percentage of connections.

To set the weight for cache server1 to 5 from the default value of 1, enter the following commands.

```
ServerIronADX(config)# server cache-name server1
ServerIronADX(config-rs-server1)# weight 5
```

Syntax: weight <*server-weight*>

The <*server-weight*> variable may have any value from 1 though 65000. The default value is 1.

Enabling FastCache

By default, the ServerIron ADX uses cache responses to client requests as a means to assess the health of the cache server. However, in an asymmetric topology where the cache server uses a path to the client that does not pass through the ServerIron ADX, the ServerIron ADX does not observe the return traffic. As a result, the ServerIron ADX concludes that the cache server has failed even though the server may still be healthy.

When the ServerIron ADX concludes that a cache server is unavailable, the ServerIron ADX stops sending client requests to the cache server. You can override this behavior by enabling the FastCache feature. The FastCache feature configures the ServerIron ADX to continue sending client requests to a cache server even if the ServerIron ADX does not see responses from the server.

To enable FastCache, enter commands such as the following.

```
ServerIronADX(config)# server cache-name server1
ServerIronADX(config-rs-server1)# asymmetric
```

Syntax: asymmetric

Enabling Remote Cache

When Proxy ARP is enabled on the router, the router informs the ServerIron ADX that it can respond on behalf of the cache server. The ServerIron ADX uses ARP requests as part of the keep-alive health-checking mechanism, so Proxy ARP enables the keep-alive health-checking mechanism to function.

If Proxy ARP is disabled on the router, the keep-alive health-checking mechanism believes the cache server cannot be reached, and does not mark the server ACTIVE or direct requests to the cache server.

You can enable the ServerIron ADX to overcome the limitation posed by the absence of Proxy ARP by enabling the remote cache feature for the cache server. To do this, enter commands such as the following.

```
ServerIronADX(config)# server cache-name C1
ServerIronADX(config-rs-C1)# remote-cache
```

Syntax: [no] remote-cache

This example enables remote cache on cache server C1. With remote cache enabled, the ServerIron ADX can perform health checks on the cache server, even though Proxy ARP is disabled on the router connecting the ServerIron ADX to the cache server.

This example assumes that a source IP address is configured and source NAT and destination NAT also are enabled, if applicable.

Shutting down a cache server

The force shutdown feature (sometimes called the force delete feature) allows you to force termination of existing SLB connections. This feature assumes that you already have shut down a TCP/UDP service on the real server or you have shut down the real server itself.

There are several methods for shutting down a cache server. Some methods involve changing the ServerIron ADX configuration while other methods involve shutting down the cache server itself. Each method has consequences, so choose the method that works best in your situation:

- Edit the cache server configuration on the ServerIron ADX to disable the HTTP (or other) port on the server. For example, to disable port 80 (HTTP), you can use the **port http disable** command at the cache level of the CLI. If you use this method, you must redefine the cache server to add the server back to TCS. However, you do need to re-enable the disabled TCP/UDP ports.

  Although the HTTP port is disabled in the ServerIron ADX definition of the cache server, all the sites mapped to the cache server before the port was disabled remain mapped to the cache server. When the cache server comes back up, it gets the same traffic it used to have. While the cache server is disabled, the remaining cache servers temporarily handle caching for the down cache server's sites, but stop when the cache is restored. This behavior is the same as if the cache actually died.

  NOTE
  You may need to set the maximum connections parameter for the remaining cache servers, especially if the servers already run at a high percentage of their capacity when all cache servers are available. Refer to "Configuring the maximum connections for a cache server" on page 57.

- Delete the cache server from the ServerIron ADX. This option immediately prevents new connections. The ServerIron ADX ends existing connections after two minutes or, if you have enabled the force shutdown option, immediately.

  Do not use this method unless you have only one cache server. If you use this method, to re-add the cache server to the ServerIron ADX, you must redefine the cache server and re-assign it to a cache group. Moreover, because the ServerIron ADX uses a hashing function to allocate contents among cache servers, the ServerIron ADX allocates traffic to the remaining caches. If the deleted cache server is down for a while in a busy network, the traffic might be unevenly balanced between the cache server that was down and the other cache servers. To reset the hash function and thus rebalance the serving load, you need to reboot the ServerIron ADX.
- Shut down the cache server itself, rather than change definitions on the ServerIron ADX. When the cache server stops responding to health checks, the ServerIron ADX removes the server from TCS. If you have only one cache server, user traffic is switched at Layer 2 to the Internet until the cache server comes back. If you have more than one cache server, the remaining cache servers provide service until the disabled cache server comes back.

This option is simple because it does not require any configuration changes on the ServerIron ADX. However, this option immediately disconnects all users from the cache server, whereas the other options allow the server or service to gracefully shut down (unless you use the force shutdown option).

NOTE
You may need to set the maximum connections parameter for the remaining cache servers, especially if the servers already run at a high percentage of their capacity when all cache servers are available. Refer to "Configuring the maximum connections for a cache server" on page 57.

Forceful shutdown on cache servers

SLB and TCS allow the graceful shutdown of servers and services. By default, when a service is disabled or deleted, the ServerIron ADX does not send new connections to the real servers for that service. However, the ServerIron ADX does allow existing connections to complete normally, however long that may take.

You can use the force shutdown option (sometimes called the force delete option) to force the existing connections to be terminated within two minutes.

NOTE
If you disable or delete a service, do not enter an additional command to reverse the command you used to disable or delete the service while the server is in graceful shutdown.

NOTE
Refer to "*" on page 90 for important information about shutting down services or servers.

Suppose you have unbound the Telnet service on real server 15 but you do not want to wait until the service comes down naturally. To force TCS connections to be terminated, enter the following command.

```
ServerIronADX(config)# server force-delete
```

Syntax: server force-delete

Passive FTP for TCS

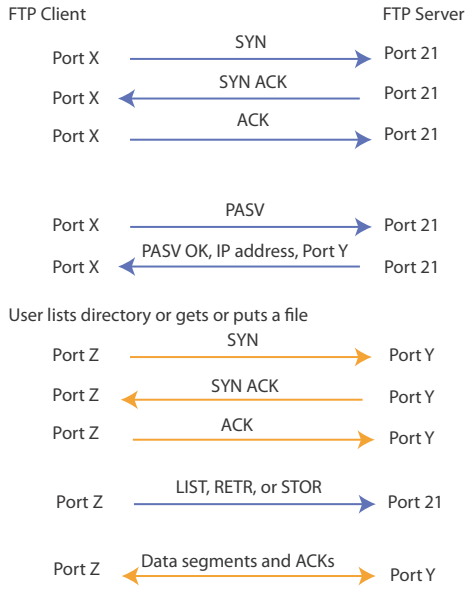
Passive FTP (sometimes referred to as PASV FTP because it involves the FTP PASV command) is a more secure form of data transfer in which the flow of data is set up and initiated by the File Transfer Protocol (FTP) client rather than by the FTP server program. Most web browsers (which act as FTP clients) use passive FTP by default because corporations prefer it as a safety measure. As a general rule, any corporate firewall server that exists in order to protect an internal network from the outside world recognizes input from the outside only in response to user requests that were sent out requesting the input. The use of passive FTP ensures all data flow initiation comes from inside the network rather than from the outside.

Traffic flow of passive FTP

Using normal or passive FTP, a client begins a session by sending a request to communicate through TCP port 21, the port that is conventionally assigned for this use at the FTP server. This communication is known as the Control Channel connection.

Figure 12 shows passive FTP packet flow. Using passive FTP, a PASV command is sent instead of a PORT command. Instead of specifying a port that the server can send to, the PASV command asks the server to specify a port it wishes to use for the Data Channel connection. The server replies on the Control Channel with the port number which the client then uses to initiate an exchange on the Data Channel. The server will thus always be responding to client-initiated requests on the Data Channel and the firewall can correlate these.

FIGURE 12 Traffic flow for passive FTP



Topologies supported

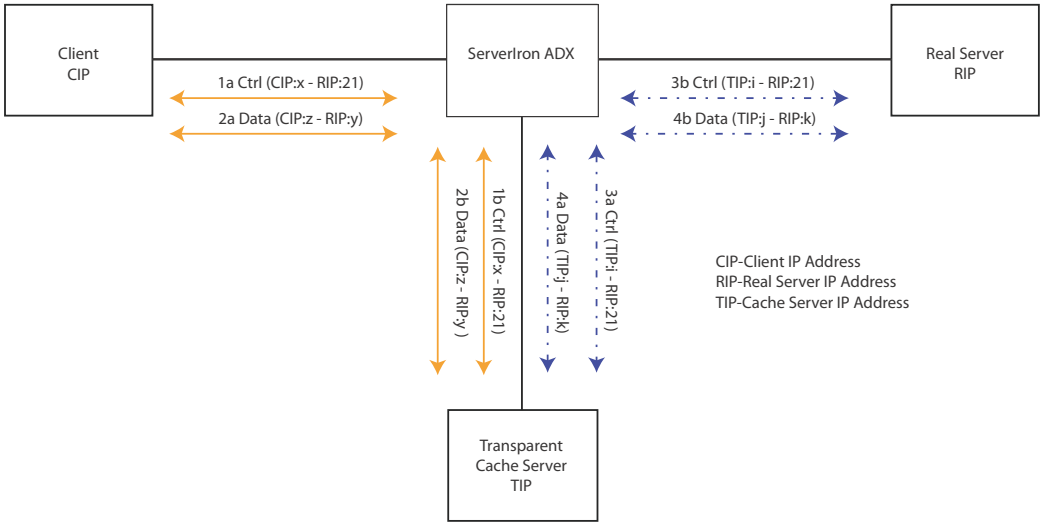
The following topologies are supported by passive FTP for TCS on the ServerIron ADX:

- Basic TCS
- TCS with spoofing

Basic TCS

Figure 13 shows the packet flow in a basic TCS configuration. In this example, Flows 1 and 2 are the Control Channel and Data Channel between the client and cache servers. Both flows are opened by the client. If the cache server does not have the information, it establishes Flows 3 and 4, which are the Control Channel and Data Channel between the cache server and the real server.

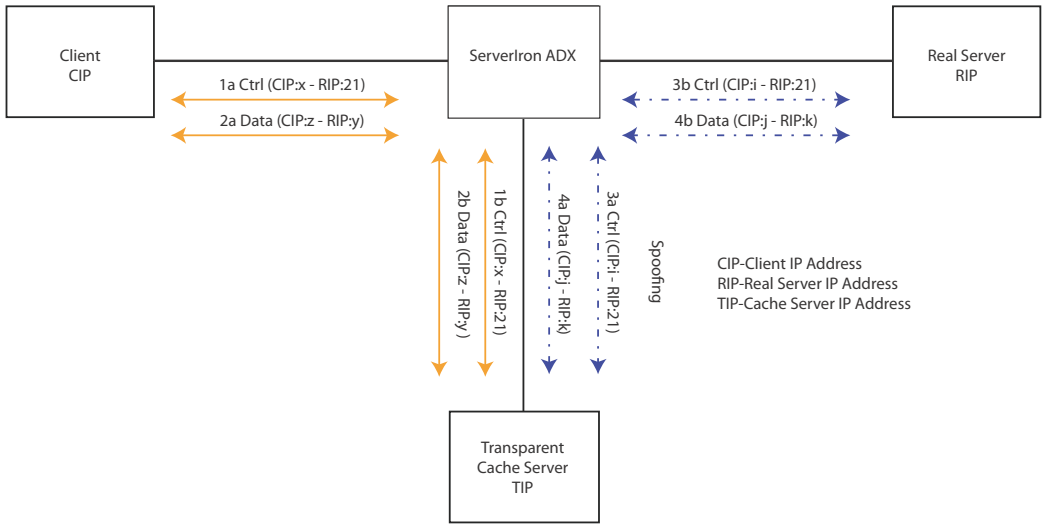
FIGURE 13 Basic TCS for passive FTP



TCS with spoofing

In Figure 14, the cache server is spoofing the client's IP address instead of using its own IP address when accessing the real server. In Flows 3 and 4, the cache server is using the client's IP address as the source address instead of using its own IP address.

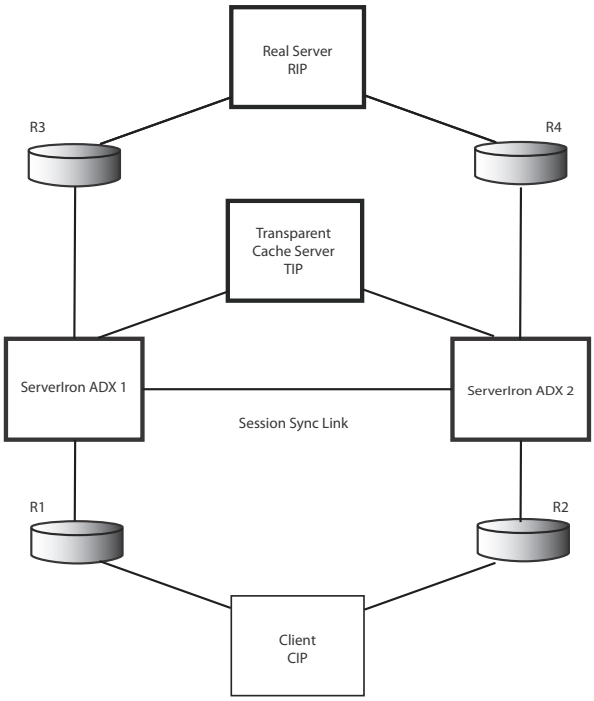
FIGURE 14 TCS with spoofing for passive FTP



High availability support

Because sessions are synced between ServerIron ADX devices, passive FTP for TCS also supports the high availability (HA) topology. The most common HA setup is Active-Active mode. Figure 15 shows an example of a Layer 2 Active-Active setup.

FIGURE 15 Layer 2 Active-Active high availability setup



Asymmetric flow

The control and data traffic between a client and cache server or between a cache server and a real server can be asymmetric, as described in Figure 16.

FIGURE 16 Asymmetric flow example

ServerIron ADX failover

Failover of the router or ServerIron ADX device is supported.

Configure VRRP on routers R1 and R2. Where R1 is the master or default gateway for both the client and cache server, the flow will be as shown in Figure 17.

FIGURE 17 Normal flow before failover

If the R1 router or “ServerIron ADX 1” fails, the flow will switch as shown in Figure 18.

FIGURE 18 Failover flow

During the failover, any traffic of the control or data channels will not be corrupted. After a short break, all transport will continue.

Enabling passive FTP caching

There is no specific CLI command to enable passive FTP caching. To enable passive CLI caching, Use the **port ftp** command within a cache server configuration, as shown in the following example.

```
ServerIronADX(config)# server cache-name CacheServer6
ServerIronADX(config-rs-CacheServer6)# port ftp
```

Whether the data channel is in active mode or passive mode depends on the operation of the FTP client and server. If they support passive mode, the ServerIron ADX can automatically adapt to it for Data Channel traffic.

Streaming media support

TCS can be used with streaming media content. The RTSP, MMS, and Real streaming media protocols are supported. The source NAT and destination NAT features are applied correctly to streams using these protocols, both for the parent TCP connection as well as the actual data stream.

NOTE
RTSP SLB and source NAT are not supported if a Darwin Streaming server is used.

To configure TCS for streaming media content, specify a streaming media port (RTSP, PNM, or MMS) as part of the definition of the cache server and configure an IP policy statement for the specified port.

For example, to configure TCS for RTSP, enter commands such as the following.

```
ServerIronADX(config)# server cache-name CacheServer1 192.168.1.101
ServerIronADX(config-rs-CacheServer1)# port rtsp
ServerIronADX(config-rs-CacheServer1)# exit
ServerIronADX(config)# ip policy 1 cache tcp rtsp global
```

If you use TCS with MMS (TCP port 1755 with random UDP ports) or PNM (TCP port 7070), you must specify port 0 in the **ip policy** command, because the command accepts TCP or UDP port numbers no higher than 1023. For example,

```
ServerIronADX(config)# server cache-name CacheServer1 192.168.1.101
ServerIronADX(config-rs-CacheServer1)# port mms
ServerIronADX(config-rs-CacheServer1)# port pnm
ServerIronADX(config-rs-CacheServer1)# exit
ServerIronADX(config)# ip policy 1 cache tcp 0 global
ServerIronADX(config)# ip policy 2 cache udp 0 global
```

NOTE
Streaming media protocols are not supported for IPv6 traffic of TCS.

Policy-based caching

Policy-based caching allows configuration of a separate set of filters for each cache group. Users can use access lists to define a set of filters and apply these to different cache groups.

Creating a set of filters using access lists

You first must create a set of filters using the **access-list** command at the CLI. You can use regular or extended access lists.

Applying access lists to cache group

Apply the access list to the desired cache group as follows. In the example, the filters defined in access list 1 are used for cache-group 1.

```
ServerIronADX(config)# server cache-group 1
ServerIronADX(config-tc-1)# filter-acl 1
ServerIronADX(config-tc-1)# exit
```

In the above example, the filters defined in access list 2 are used for cache-group 2.

```
ServerIronADX(config)# server cache-group 2
ServerIronADX(config-tc-2)#filter-acl 2
ServerIronADX(config-tc-2)#exit
```

Syntax: filter-acl { <Access List ID> | < Access List Name > }

NOTE
Although IPv4 and IPv6 ACLs are supported, an IPv4 ACL can only be bound to an IPv4 cache group and an IPv6 ACL can only be bound to an IPv6 cache group.

Configuring default cache groups

You can configure a default cache group. If the traffic does not match the ACL IDs for any of the cache groups, then it is sent to the default cache group. You do not need to explicitly associate an ACL ID with the default cache group; the behavior of the default cache group is "permit any any".

```
ServerIronADX(config)# server cache-group 3
ServerIronADX(config-tc-1)# cache-name Cache-Server1
ServerIronADX(config-tc-1)# cache-name Cache-Server2
ServerIronADX(config-tc-1)# default-group
ServerIronADX(config-tc-1)# exit
```

Syntax: default-group

NOTE
If default cache group is not configured or if no cache servers are associated with the default cache group, then the traffic is sent to the Internet if the traffic does not match any of the group ACLs.

Configuring an ACL to bypass caching

You can configure a bypass filter to redirect traffic to the Internet instead of sending it to the cache servers. Configure an access list using the existing access-list CLI if you want to designate it as the bypass filter as follows.

```
ServerIronADX(config)# server cache-bypass 3
```

Syntax: server cache-bypass <*acl-id*> [**ipv6]**

The <*acl-id*>variable specifies the ID of the IPv4 or IPv6 ACL being used for bypass caching.

You must use the **ipv6** parameter if the ACL being used for bypass caching is an IPv6 ACL.

The configured bypass caching ACL will be evaluated first. If traffic matches this ACL, this traffic will be sent directly to the Internet.

NOTE
This bypass caching ACL is global in scope; that is, it will apply to all cache groups. It should be configured as a "permit" ACL.

Summary of configuration constraints

- You can configure the filters using the **access-list** command and associate the ID with a cache group. Note that the filters in the acl-id will apply to all cache servers in the cache group. If you do not want the ACL ID to apply to a particular server in the cache group, you must create another cache group and move the server to this group.
- If you do not configure a default cache group, then traffic that does not match any of the group ACLs will be sent to the Internet.
- When policy-based caching is enabled, you should not disable any cache groups. Disabling a cache group could result in disruption of traffic. If you need to prevent traffic from being serviced by a particular group, you should update the filter ACL associated with the group accordingly instead of disabling the group.
- When policy-based caching is enabled, you should not turn on spoofing for a subset of the cache groups. You can either turn on spoofing for all the cache groups or turn it off for all of them.

Show commands

The **show cache-group** command displays the ACL ID, if one is associated with the group and the hit count for the associated policy number.

Syntax: show cache-group [cache-group number]

Debug commands

You can configure the following command to enable debugging for enhanced policy-based caching.

```
ServerIronADX(config)# server debug-policy-caching
```

Syntax: server debug-policy-caching

NOTE
The **server debug-policy-caching** command impacts performance and should be used for debugging purposes only. Output for this command is sent to the BP.

Content-aware cache switching

Content-aware cache switching (CSW in a TCS environment) uses information in the header of an HTTP request to determine how or if content should be retrieved from a cache server. Using the text in a URL string, the ServerIron ADX sends a request from a client to a cache server or to the Internet according to user-defined policies.

You can configure content-aware cache switching on the ServerIron ADX to do the following:

- Group cache servers by content; for example, GIF files can be cached on one cache server and HTML files on another
- Cause HTTP requests containing a given URL string to alwasys go to the same cache server, minimizing content duplication among cache servers
- Use information in the URL string or Host header field of an HTTP request to determine how the requested content should be cached
- Explicitly direct requests for dynamic content to the Internet, rather than to a cache server
- Use directives in the HTTP 1.0 or 1.1 header to determine whether requested content should be cached

NOTE
Content-aware cache switching utilizes the resilient hashing mechanism as described in "Controlling traffic distribution among cache servers".

How content-aware switching works

Content-aware switching (CSW) is the ServerIron ADX's ability to direct HTTP requests to a server, or group of servers, using information in the text of a URL string. The ServerIron ADX examines the contents of a URL string and makes a decision about where to send the packet based on selection criteria in user-defined policies. If text in the URL string matches the selection criteria, the HTTP request is sent to a server group specified in the policy.

NOTE
"URL string" is defined as the contents of the Request-URI part of the Request-Line in an HTTP request message. This information usually consists of the absolute pathname (directory and filename) of a resource; for example,
/doc/ServerIron ADX/1199/url_switching.html.

The URL string can also be the input to a process running on a remote server; for example,
/quote.cgi?s=BRCD&d=1d.

The network location of the resource is specified in the Host header field in an HTTP request message; for example,
Host: www.brocade.com.

The ServerIron ADX can examine both the URL string and Host header field when determining where to send the HTTP request. Refer to RFC 1945 or RFC 2616 for more information on HTTP request messages.

The selection criteria in a policy can be a string of characters starting from the beginning of the URL string, the end of the URL string, or within any part of the URL string. For example, selection criteria can be a URL string that starts with the text “/home”. In a TCS environment, when a client sends an HTTP request that has a URL string beginning with the text “/home", the policy can direct that request to a specific group of cache servers (or to another CSW policy for additional matching).

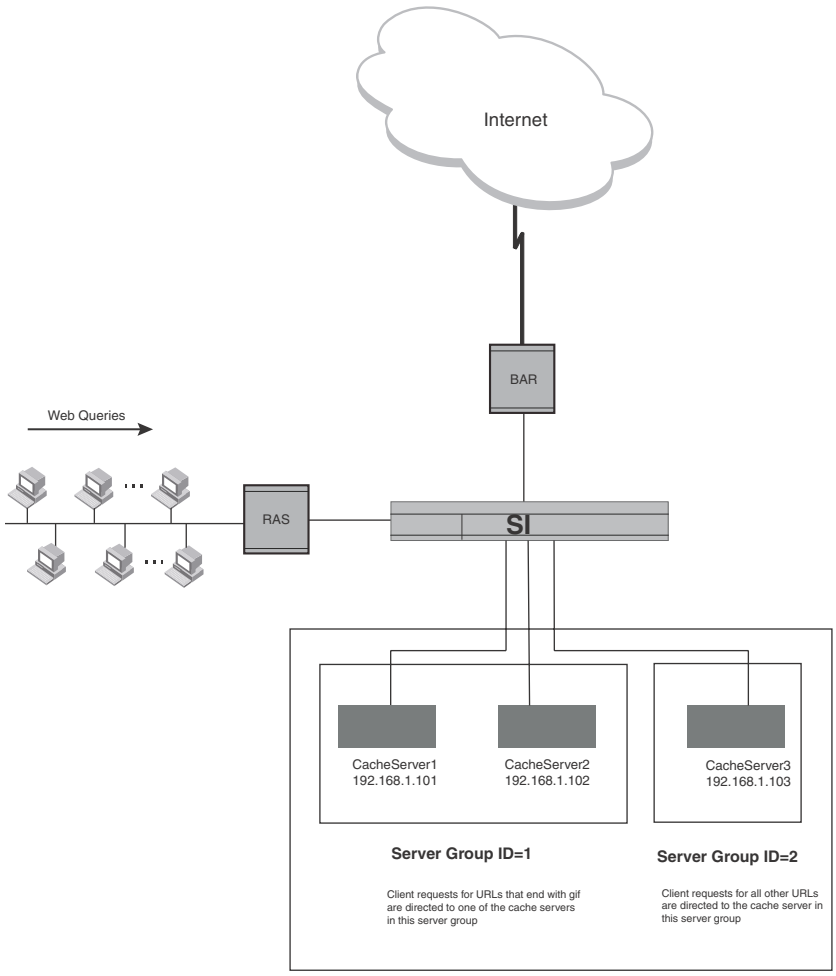
Basic example of content-aware cache switching

The diagram in Figure 19 illustrates a configuration that uses content-aware cache switching to cache GIF files on one set of cache servers and different kinds of files on another set.

In this configuration, Cache Group 1 consists of three cache servers. CacheServer1 and CacheServer2 are allocated to Server Group ID = 1, and CacheServer3 is allocated to Server Group ID = 2. The ServerIron ADX has CSW policies in place that cause HTTP requests to be directed to the cache servers as follows:

- HTTP requests containing URL strings that end with the text "gif" are sent to one of the cache servers in Server Group ID = 1.
- If a URL string does not end with the text "gif", the HTTP request is sent to the cache server in Server Group ID = 2.

FIGURE 19 Content-aware cache switching



The first time a client requests a URL that ends with "gif" (for example, /home/main/banner.gif) the following events take place.

1. Because the URL ends with "gif", the CSW policy on the ServerIron ADX directs the request to one of the cache servers in Server Group ID=1.
2. When a server group consists of more than one cache server, the ServerIron ADX uses a hashing algorithm to select one of the cache servers, and directs the request to the selected cache server.
3. Because this is the first time the content is requested, the selected cache server does not have the content stored, so the cache server retrieves it from the Internet.
4. The cache server receives the content, caches it, and sends it to the requesting client.

The next time a client requests the content, the following events take place.

1. Because the URL begins with "gif", the CSW policy directs the request to one of the cache servers in Server Group ID=1.
2. The ServerIron ADX hashes the URL string, selecting the same server it selected previously.
3. This time the cache server has the content and does not have to go to the Internet to get it; it sends the cached content to the requesting client.

Setting up content-aware cache switching consists of the following steps.

1. Enabling TCS on the ServerIron ADX
2. Setting up CSW policies
3. Configuring the cache servers
4. Assigning the cache servers to a cache group

Enabling TCS

To enable TCS on all interfaces (globally) of the ServerIron ADX shown in Figure 19, enter the following command.

```
ServerIronADX(config)# ip policy 1 cache tcp 80 global
```

Syntax: ip policy <*index*> **cache | normal | high tcp | udp** <*tcp/udp-portnum*> **global | local**

Setting up the CSW policies

The CSW policies define selection criteria for URL strings and specify what happens when a URL string matches the selection criteria. In content-aware cache switching, if an HTTP request contains a URL string that matches a policy's selection criteria, the HTTP request can be sent to a load balanced cache server group or to another policy for additional matching.

NOTE
The CSW policies discussed in this section apply to the example in Figure 19 on page 70.

The following commands define a CSW policy called "p1".

```
ServerIronADX(config)# csw-rule r1 url suffix "gif"
ServerIronADX(config)# csw-policy p1
ServerIronADX(config-csw-p1)# match r1 forward 1
ServerIronADX(config-csw-p1)# default forward 2
ServerIronADX(config-csw-p1)# exit
```

Syntax: csw-rule r1 url prefix | suffix | pattern"<*selection-criteria*>**"**

The **csw-rule r1 url suffix gif** command consists of two parts. The first part specifies what kind of matching the policy does on the selection criteria. Three kinds of matching methods are supported:

- The **prefix** keyword compares the selection criteria to the beginning of the URL string.
- The **suffix** keyword compares the selection criteria to the end of the URL string.
- The **pattern** keyword looks for the selection criteria anywhere within the URL string.

The second part of the specifies the selection criteria, which can be up to 80 characters in length; In this example, the selection criteria is the text string "gif". Because the matching method is **suffix**, the policy looks at the end of the URL string. If the URL string ends with the text "gif", then the URL string meets the selection criteria.

NOTE
In addition to using text as selection criteria, you can use an asterisk (*) as a wildcard character to specify one or more characters at the end of a URL string. For example, using "/ho*" as the selection criteria matches /home, /hotels, and /home/main/index.html.

If you are using the **suffix** matching method, you cannot use an asterisk (*) as a wildcard character. The asterisk wildcard character is valid for the **prefix** and **pattern** matching methods only.

Syntax: csw-policy *<policy-name>*

The **csw-policy p1** command sets the name of the policy.

Syntax: match r1 forward *<server-group-id>*

If the URL string meets the selection criteria, the second part of the **match** command specifies what to do with the HTTP request. In this example, the **1** in the command causes the HTTP request to be sent to the cache server group whose ID = 1. Specifying **0** in the **match** command causes the request to be directed to the Internet. A CSW policy can contain multiple **match** commands, each with different selection criteria.

Syntax: default forward*<server-group-id>*

The **default forward 2** command specifies what happens when the URL string does not meet any of the selection criteria in a CSW policy's **match** command. With a **match** command, you can specify a server group ID number. In this example, if a URL string does not match the selection criteria in policy p1, it is sent to Server Group 2 for evaluation.

NOTE
As the diagram in Figure 19 illustrates, there is only one cache server in Server Group ID = 2. Even so, the **match** command must refer to a server group rather than an actual cache server. Server groups can consist of one or more cache servers.

NOTE
By default, if no cache server is found in the server group then the request is bypassed to the Internet. To override this behavior, you need to enable 'group-failover' under the cache-group. Refer to the section '"Configuring group-failover" on page 73 for further details.

Configuring the cache servers

The cache servers return the content to the requesting clients. When configuring content-aware cache switching, you place the cache servers into logical server groups. CSW policies direct HTTP requests to one of the cache servers in these logical groups.

A server group can contain one or more cache servers. When a server group consists of more than one cache server, the ServerIron ADX uses a hashing algorithm to select one of the cache servers, and directs the request to the selected cache server. When configuring content-aware cache switching, you establish the IP address of each cache server and specify the server group to which it belongs.

To configure CacheServer1 in Figure 19 on page 70, enter commands such as the following

```
ServerIronADX(config)# server cache-name CacheServer1 192.168.1.101
ServerIronADX(config-rs-CacheServer1)# port http group-id 1 1
ServerIronADX(config-rs-CacheServer1)# exit
```

Syntax: port http group-id *<server-group-id-pairs>*

The **port http group-id** command indicates the server groups to which the cache server belongs. The server group is expressed as a pair of numbers, indicating a range of server group IDs. The first number is the lowest-numbered server group ID, and the second is the highest-numbered server group ID. For example, if a cache server belongs only to the server group with ID = 1, the last two numbers in the **port http group-id** command would be **1 1**. (Note the space between the two numbers.) If a cache server belongs to server groups 1 through 10, the last two numbers in the command would be **1 10**. Valid numbers for server group IDs 0 through 1023.

To include a cache server in groups that are not consecutively numbered, you can enter up to four server group ID pairs. For example, to include a cache server in groups 1 through 5 and 11 through 15, you would enter the following command.

```
ServerIronADX(config-rs-CacheServer1)# port http group-id 1 5 11 15
```

You can also specify the server group ID pairs on separate lines by entering commands such as in the following example.

```
ServerIronADX(config-rs-CacheServer1)# port http group-id 1 5
ServerIronADX(config-rs-CacheServer1)# port http group-id 11 15
```

The following commands configure the remaining cache servers in Figure 19 is shown below. These commands place CacheServer2 in server group ID = 1 (along with CacheServer1) and CacheServer3 in server group ID = 2.

```
ServerIronADX(config)# server cache-name CacheServer2 192.168.1.102
ServerIronADX(config-rs-CacheServer2)# port http group-id 1 1
ServerIronADX(config-rs-CacheServer2)# exit
ServerIronADX(config)# server cache-name CacheServer3 192.168.1.103
ServerIronADX(config-rs-CacheServer3)# port http group-id 2 2
ServerIronADX(config-rs-CacheServer3)# exit
```

Assigning the cache servers to a cache group

To activate content-aware cache switching (as in Figure 19), you create a cache group, assign the cache servers to that group, and specify a CSW policy to be active for the cache group, such as in the following example.

```
ServerIronADX(config)# server cache-group 1
ServerIronADX(config-tc-1)# cache-name CacheServer1
ServerIronADX(config-tc-1)# cache-name CacheServer2
ServerIronADX(config-tc-1)# cache-name CacheServer3
ServerIronADX(config-tc-1)# csw
ServerIronADX(config-tc-1)# csw-policy p1
```

Syntax: csw

Syntax: csw-policy <*policy-name*>

Configuring group-failover

When CSW rules are enabled with TCS, the ServerIron ADX tries to select a cache server with a 'server group-id' as specified by the CSW policy. By default, if no cache server is found in the server group then the request is bypassed to the Internet.

When the group-failover feature is enabled, the ServerIron ADX behaves in the following manner:

- If a cache server is found with the matching server group-id, then the HTTP request is forwarded to that cache server.

- If a cache server with the matching server group-id is not found, but a default CSW rule with forward action is specified, then the HTTP request is forwarded to a cache server defined inside the default rule.
- If a cache server with the matching server group-id is not found, and a default CSW rule with forward action is not specified, then the HTTP request is forwarded to any available cache server within the cache-group.

This process is demonstated in the configuration described in: “group-failover example”.

To enable group failover, enter commands such as the following:

```
ServerIronADX(config)#server cache-group 1
ServerIronADX(config-tc-1)#group-failover
ServerIronADX(config-tc-1)#
```

Syntax: [no] group-failover

group-failover example

Given the following TCS configuration:

```
ServerIronADX(config)# csw-policy policy1
ServerIronADX(config-csw-policy1)# match rule1 forward 1
ServerIronADX(config-csw-policy1)# default forward 2
ServerIronADX(config-csw-policy1)# exit

ServerIronADX(config)# server cache-name Cache1 10.1.1.11
ServerIronADX(config-rs-Cache1)# port http url "HEAD /"
ServerIronADX(config-rs-Cache1)# port http group-id  1 1

ServerIronADX(config)# server cache-name Cache1 10.1.1.12
ServerIronADX(config-rs-Cache1)# port http url "HEAD /"
ServerIronADX(config-rs-Cache1)# port http group-id 2 2

ServerIronADX(config)# server cache-name Cache1 10.1.1.13
ServerIronADX(config-rs-Cache1)# port http url "HEAD /"
ServerIronADX(config-rs-Cache1)# port http group-id 3 3

ServerIronADX(config)#server cache-group 1
ServerIronADX(config-tc-1)# cache-name cache1
ServerIronADX(config-tc-1)# cache-name cache2
ServerIronADX(config-tc-1)# cache-name cache3
ServerIronADX(config-tc-1)# spoof-support

ServerIronADX(config)# csw-policy policy1
```

Suppose the client sends an HTTP request to the remote host: “abc.com”. The ServerIron ADX will then forward the request to 'cache1' due to the configured csw-policy: 'policy1'. If cache1 goes down, subsequent requests to “abc.com” will then be directed to the Internet. The default rule will not be evaluated here since a csw-rule match was already found. The default rule will only be evaluated when the incoming request does not match against any configured csw-rule.

```
ServerIronADX(config)# server cache-group 1
ServerIronADX(config-tc-1)# group failover
```

When **group-failover** is enabled: if cache1 goes down, then the request is matched against the default rule, and directed to 'cache2'. If cache2 was also down, then the request is sent to any available cache server defined under 'cache-group 1' (such as 'cache3' in this example).

Configuring policies for dynamic content

For dynamic web pages such as Active Server Pages, it may be preferable not to cache the content. You can configure CSW policies on the ServerIron ADX that cause requests for these kinds of pages to bypass the cache servers and go directly to the Internet.

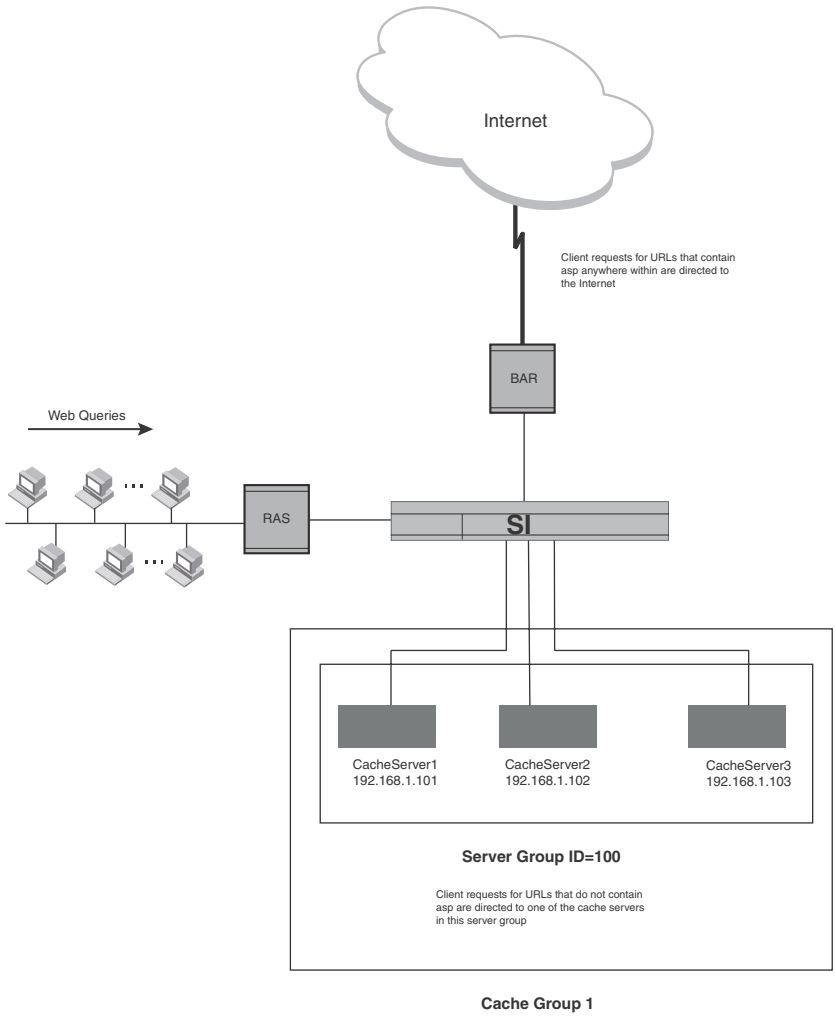
In addition, the ServerIron ADX examines directives in the HTTP 1.0 or 1.1 header to determine whether a request should be sent to the cache servers or to the Internet. When this feature is enabled (the default), a request is sent to the origin server regardless of the URL string if one of the following is true:

- The request contains a pragma:no-cache header (HTTP 1.0 requests).
- The Cache-Control header in the request contains a no-cache directive (HTTP 1.1 requests).

In the configuration in Figure 20 on page 76, the ServerIron ADX has CSW policies in place that cause HTTP requests to be directed to the cache servers as follows:

- Requests that have URL strings with the text "asp" anywhere within go directly to the Internet.
- Requests for all other content are directed to one of the cache servers in server group ID = 100.
- HTTP 1.0 requests that have a pragma:no-cache header are sent to the Internet regardless of the URL string.
- HTTP 1.1 requests that have a Cache-Control header containing a no-cache directive are sent to the Internet regardless of the URL string.

FIGURE 20 Sending requests for Active Server Pages to the Internet



The following sections explain how to set up this configuration.

Enabling TCS

To enable TCS on all interfaces (globally) of the ServerIron ADX shown in Figure 20, enter the following command.

```
ServerIronADX(config)# ip policy 1 cache tcp 80 global
```

Setting up the CSW policies

To implement the configuration in Figure 20, you would create a CSW policy that sends all requests containing URL strings ending with "asp" directly to the Internet, bypassing the cache servers. All other requests are sent to one of the cache servers in Server Group ID = 100.

The following commands define a CSW policy called “policyA1”.

```
ServerIronADX(config)# csw-rule r1 url pattern “asp”
ServerIronADX(config)# csw-policy policyA1
ServerIronADX(config-csw-policyA1)# match forward 0
ServerIronADX(config-csw-policyA1)# default forward 100
ServerIronADX(config-csw-policyA1)# exit
```

The **pattern** method in the **csw-rule r1 url** command causes the policy to look for the selection criteria anywhere within the URL string.

The **match forward 0** command looks for URL strings that contain the text "asp"; for example, /active/q.asp?ln=fdry. These HTTP requests are sent to the Internet.

The **default forward 100** command sends HTTP requests that do not meet the selection criteria in policyA1's **match** command Server Group ID = 100.

Configuring the cache servers

To place the cache servers in Figure 20 on page 76 into server group ID = 100, enter the following commands.

```
ServerIronADX(config)# server cache-name CacheServer1 192.168.1.101
ServerIronADX(config-rs-CacheServer1)# port http group-id 100 100
ServerIronADX(config-rs-CacheServer1)# exit
ServerIronADX(config)# server cache-name CacheServer5 192.168.1.102
ServerIronADX(config-rs-CacheServer5)# port http group-id 100 100
ServerIronADX(config-rs-CacheServer5)# exit
ServerIronADX(config)# server cache-name CacheServer6 192.168.1.103
ServerIronADX(config-rs-CacheServer6)# port http group-id 100 100
ServerIronADX(config-rs-CacheServer6)# exit
```

Assigning the cache servers to a cache group

To activate the configuration shown in Figure 20, enter the following commands.

```
ServerIronADX(config)# server cache-group 1
ServerIronADX(config-tc-1)# cache-name CacheServer1
ServerIronADX(config-tc-1)# cache-name CacheServer2
ServerIronADX(config-tc-1)# cache-name CacheServer3
ServerIronADX(config-tc-1)# http-cache-control
ServerIronADX(config-tc-1)# csw
ServerIronADX(config-tc-1)# csw-policy policyA1
```

Syntax: [no] http-cache-control

The **http-cache-control** command ensures that HTTP 1.0 requests that have a pragma:no-cache header and HTTP 1.1 requests that have a Cache-Control header containing a no-cache directive are sent to the Internet. This is the default behavior. To configure the ServerIron ADX to ignore the pragma:no-cache or Cache-Control header in an HTTP request, use the **no http-cache-control** command.

Bypassing embedded protocols

In network environments where non-HTTP protocols such as FTP and RTSP are masquerading over port HTTP, ServerIron ADX enables network administrators to bypass such embedded traffic to the Internet rather than forwarding it to cache servers.

Once this feature is enabled, the ServerIron ADX categorizes port 80 traffic as either *true HTTP traffic* or *non-HTTP traffic*:

- If the traffic is identified as HTTP traffic, then subsequent packets are processed normally using transparent cache switching.
- If the traffic is identified as non-HTTP traffic masquerading over port 80, then subsequent packets are bypassed to the Internet.

Use the **bypass-embedded-protocols** command, to enable the ServerIron ADX to classify incoming traffic as either HTTP or non-HTTP traffic. Once enabled, the embedded protocol traffic is bypassed to the Internet server.

Enter a command such as the following under the cache group configuration level:

```
ServerIronADX(config-tc-1)# bypass-embedded-protocols
```

Syntax: [no] bypass-embedded-protocols

The **no** option disables bypass embedded protocols. There are no other operands for this command.

HTTP 1.1 support for cache switching

Beginning with release 12.4.00, HTTP keep-alive mode is enabled by default for content-aware cache switching. This has been added to fully support HTTP 1.1 for TCS CSW. Unlike the operation in previous releases of ServerIron ADX, the client request is not downgraded to HTTP 1.0 and the connection header is unmodified. The client will send subsequent requests over the same TCP connection if the server supports HTTP keep-alive. The ServerIron ADX software will maintain the HTTP state machine to track HTTP request and response transactions. Every HTTP request will be analyzed and forwarded according to the CSW configuration. When CSW makes a decision to switch from one server to another server, it will send a TCP reset to the previously chosen server and establish a TCP connection to the newly selected server

Disabling HTTP keep-alive mode

HTTP keep-alive mode is the default mode for TCS CSW. You can disable keep-alive to return the ServerIron ADX to HTTP 1.0 mode which was supported in releases prior to 12.4.00, as shown in the folowing example.

```
ServerIronADX(config)# server cache-group 1
ServerIronADX(config-tc-1)# no-keep-alive
```

Syntax: [no] no-keep-alive

Use the **no** parameter to re-enable HTTP keep-alive if you have disabled it.

Displaying HTTP keep-alive statistics

You can use the **show server proxy keep-alive** command to display HTTP keep-alive statistics. The statistics relevant to HTTP keep-alive are shown (in bold) in the following abbreviated output and described in Table 5.

```
ServerIronADX 1000#show server proxy keep-alive
Keep-alive connection statistics:
...

    TCB status:
    Total in mem         =     100000   Current in pool      =          0
    Allocated from mem   =          2   Freed to mem         =          1
    Allocated from pool  =         36   Freed to pool        =         36
    Allocated from FS po =          0   Freed to FS pool     =          0
    Free to FS mem       =          0   Clean non-reusables  =          0
    Clean reusables      =          0   Clean in wrong state =          0
    None-reusable to mem =          0   Rate exceeded to mem =          0

    Connection unreusable reasons:
    Small window         =          0   No rev sess          =          0
    Not reusable         =          0   Fin/RST received     =          0
    Image                =          0                        =          0

    Delayed ACK list status:
    Total TCBs in list   =         12   Curr TCBs in list    =          0
    Generated ack num    =          0

...
SYN_RECV             =          0   WAIT_REQ             =          0
    NOT_COMPLETE         =          0   REQ_STORED           =          0
    SYN_SENT             =          0   REQ_SENT             =          0
    PAGE_REPLIED         =          0   STATE_UNKNOWN        =          0
    Reply_Sent           =          0   sock free to pool er =          0
    sock get from mem    =          0   send SYN to server   =          0
    send SYN to server f =          0   send reset to server =          0
    KA switch real port  =          0   KA reuse connection  =          0
    Curr TCBs in list    =          0   Total TCBs in list   =         12
    Generated ack num    =          0   Curr buffered data   =          0
    Curr buffered pkts   =          0   Total pipeline reqs  =          0
    Unknown              =          0                        =          0
KA DEBUG: URL_MULTI_STATE_FREED [   31] =                   37
```

Syntax: show server proxy keep-alive

The fields described in Table 5 provide statistics about HTTP keep-alive.

TABLE 5 HTTP keep-alive statistics

Field	Description
Allocated from pool	The number of instances when a server port was allocated from the Keep-Alive pool.
Freed to pool	The number of instances when a server port was freed to the Keep-Alive pool.
Total TCBs in list (appears in two locations)	The total number of Keep-Alive connections received by the ServerIron ADX.

You can clear the HTTP keep-alive counter on a ServerIron ADX as shown in the following example.

```
ServerIronADX# clear server keep-alive statistics
```

Syntax: clear server keep-alive statistics

Cache persistence using URL hashing

The ServerIron ADX enables traffic distribution among cache servers in a TCS setup after inspecting and hashing based on the request URL. This enables cache persistence based on the requested URL and minimizes duplication of content among multiple cache servers.

The hashing can be done on a complete URL or a portion of the URL. The following choices are available:

- Complete URL
- Path only
- Path and parameters only
- Host only
- Host and path only

Additionally, you can identify a sub-portion of either the complete URL or the host portion of the URL to perform the hashing on.

The hashing method is specified within a CSW policy definition as an action to be taken when a match is made.

NOTE
In previous versions of the ServerIron ADX software, this feature was configured using the **csw-hash url** command within the **server cache-group** configuration. This command is no longer available.

The following example directs the ServerIron ADX to hash on the complete URL when the conditions in the "r1" CSW rule is met.

NOTE
Because hash-persist action is a secondary action, you must add a forward action as shown in the following example, before adding a **hash-persist** action.

```
ServerIronADX(config)# csw-policy p1
ServerIronADX(config-csw-p1)# match r1 forward 1
ServerIronADX(config-csw-p1)# match r1 hash-persist url complete
```

Syntax: [no] hash-persist url <*method*>

The <*method*> variable can be one of the following:

- complete
- path-only
- path-and-parameters - default value
- host-only
- host-and-path

For example, when a client tries to connect to http://login.yahoo.com/config/mail?.src=ym, the http requests can come in two different ways:

1. Get http://login.yahoo.com/config/mail?.src=ym HTTP 1.1
2. Get /config/mail?.src=ym

Host: login.brocade.com

The following table demonstrates which part of the string would be used for hashing either of the prior examples depending the method selected with the **csw-hash url** command.

TABLE 6 Hashing methods

Method	String used for hashing
Complete	login.brocade.com/confg/mail?.src=ym
path-only	config/mail
path-and-parameters	confg/mail?.src=ym
host-only	login.brocade.com
host-and-path	login.brocade.com/confg/mail

Cache persistence using hashing on a portion of the URL

You can be more specific about the information in the URL that you want to determine traffic distribution for by parsing the URL string. This is done by specifying a "pattern-string" and parsing the URL string using a delimiter or an length limit. You can either parse the entire URL string or only the "host-only" portion of the string.

- the **hash-persist url search-url** command is used where the entire URL is being parsed.
- the **hash-persist url search-host** command is used where the host portion of the URL is being parsed.

Parsing the entire URL

Parsing the entire URL string allows you to select a portion of a URL to perform hashing on. The URL is searched to find a pattern string. This pattern string is combined with a configured offset value to identify a starting point. From this starting point in the URL, a string that will be used for hashing is derived by a method that is specified by the options you select for the **hash-persist url search-url** command. These options are described in the following.

Parsing using the "pattern string" only – With this option, the ServerIron ADX begins at the starting point in the URL and includes all of the characters up-to the end of the URL to form the string used for hashing.

Search URL string by starting at a "pattern string" and ending at a delimiter – With this option, the ServerIron ADX begins at the starting point in the URL and includes all of the characters up-to a specified delimiter character in the URL to form the string used for hashing.

Search URL string by starting at a "pattern string" and continuing for a specified length – With this option, the ServerIron ADX begins at the starting point in the URL and includes the number of characters specified to form the string used for hashing.

Parsing using the "pattern string" only

The following example identifies a pattern string to perform hashing on with an offset of "0".

NOTE
Because hash-persist action is a secondary action, you must add a forward action as shown in the following example, before adding a **hash-persist** action.

```
ServerIronADX(config)# csw-policy p1
ServerIronADX(config-csw-p1)# match r1 forward 1
ServerIronADX(config-csw-p1)# match r1 hash-persist url search-url “v=” offset 0
```

Syntax: [no] hash-persist url search-url <pattern-string> offset <offset-value>

The contents of the **<pattern-string>** variable are used to define the starting place on the URL string. If the URL string has multiple such pattern strings, only the first one will be used. If the pattern string is not found, the system will abort the rest of the searching steps, and choose the default hashing method. If the pattern string is found, the system will start the second step after skipping the length of the pattern string. The second step in the process is to adjust the starting place on the URL string by moving it by the number of characters defined by a value specified in the **<offset-value>**. If the pattern string is empty, the system will not search the pattern part, and use the beginning of URL string as the start point of the third step.

The **<offset-value>** is used to define how many characters will be skipped after the start point that is defined by the pattern-string. Normally this value is 0 (zero) which places the start point directly after the pattern string. A negative value can be used to move the starting place to the left in the URL string. If the **<offset-value>** is greater than the length of the rest of the URL string, the system will abort the rest of the searching steps, and start the next search (if configured). If another search isn’t configured, the default hashing method is used. If the value of the **<offset-value>** variable is within the length of the rest of the URL string, the system will skip to the offset place, and start the third step.

With this command, the third step is for the system to look to the end of the URL to define the string used for hashing.

Parsing using the “pattern string” with a delimiter

The following example identifies a pattern string to perform hashing on with an offset of “0” and a delimiter with the value “&”.

NOTE
Because hash-persist action is a secondary action, you must add a forward action as shown in the following example, before adding a **hash-persist** action.

```
ServerIronADX(config)# csw-policy p1
ServerIronADX(config-csw-p1)# match r1 forward 1
ServerIronADX(config-csw-p1)# match r1 hash-persist url search-url “v=” offset 0
delimiter “&”
```

Syntax: [no] hash-persist url search-url <pattern-string> offset <offset-value> delimiter <delimiter-string>

The contents and operation of the **<pattern-string>** variable is the same as described in “Parsing using the “pattern string” only”.

The contents and operation of the **<offset-value>** is the same as described in “Parsing using the “pattern string” only”.

With this command, the third step is for the system to parse the URL string up-to a character defined by the value of the **<delimiter-string>**. The delimiter string is used to define the end pattern after finding the substring. If the rest of URL string has multiple delimiter strings, only the first one will be used. If the **<delimiter-string>** is not found, or the value of the **<delimiter-string>** value is empty, the system will look to the end of the URL to define the string used for hashing.

Parsing using the "pattern string" with a specified string length

The following example identifies a pattern string to perform hashing on with an offset of "0" and a length value of "8".

NOTE
Because hash-persist action is a secondary action, you must add a forward action as shown in the following example, before adding a **hash-persist** action.

```
ServerIronADX(config)# csw-policy p1
ServerIronADX(config-csw-p1)# match r1 forward 1
ServerIronADX(config-csw-p1)# match r1 hash-persist url search-url "v=" offset 0
length 8
```

Syntax: [no] hash-persist url search-url <pattern-string> offset <offset-value> length <length>

The contents and operation of the **<pattern-string>** variable is the same as described in "Parsing using the "pattern string" only".

The contents and operation of the **<offset-value>** variable is the same as described in "Parsing using the "pattern string" only".

With this command, the third step is for the system to parse the URL string up-to the number of characters defined by the value of the **<length>** variable. The value of the **<length>** variable must be greater than 0 (zero). If the value of the **<length>** variable extends beyond the length of the URL, the system will look to the end of the URL to define the string used for hashing.

Examples for parsing on the entire URL

In the following example, the client tries to connect to youtube at the following URL.

```
http://www.youtube.com/watch?v=bUfp24dwzOA&playnext=1&videos=XA7MyzKoQXQ&feature=
featured
```

Table 7 displays the contents of the strings used for hashing if the **<pattern-string>** variable is set to "v=" and the offset value is set to "0" depending on the method used. The methods and commands required are described as follows.

Parsing using the "pattern string" only

```
ServerIronADX(config)# csw-policy p1
ServerIronADX(config-csw-p1)# match r1 forward 1
ServerIronADX(config-csw-p1)# match r1 hash-persist url search-url "v=" offset 0
```

Parsing using the "pattern string" with a delimiter

```
ServerIronADX(config)# csw-policy p1
ServerIronADX(config-csw-p1)# match r1 forward 1
ServerIronADX(config-csw-p1)# match r1 hash-persist url search-url "v=" offset 0
delimiter "v="
```

Parsing using the "pattern string" with a specified string length

```
ServerIronADX(config)# csw-policy p1
```

```
ServerIronADX(config-csw-p1)# match r1 forward 1
ServerIronADX(config-csw-p1)# match r1 hash-persist url search-url “v=” offset 0
length 16
```

TABLE 7 Results for parsing the entire URL by method

Method	String used for hashing
pattern string only	bUfp24dwzOA&playnext=1&videos=XA7MyzKoQXQ&feature=featured
pattern string with a delimiter of “&”	cbUfp24dwzOA
pattern string with a length value of “16”	bUfp24dwzOA&play

Parsing the host string

Parsing the host string allows you to select a “pattern-string” within a URL to determine a starting point for identifying a URL sting to perform hashing on. This pattern string is combined with a configured offset value to identify a starting point. From this starting point in the URL, a string that will be used for hashing is derived by a method that is specified by the options you select for the **hash-persist url search-host** command. These options are described in the following.

Parsing using the “pattern string” only – With this option, the ServerIron ADX begins at the starting point in the URL and includes all of the characters up-to the end of the host string to form the string used for hashing.

Search host string by starting at a “pattern string” and ending at a delimiter – With this option, the ServerIron ADX begins at the starting point in the URL and includes all of the characters up-to a specified delimiter character in the host string to form the string used for hashing.

Search host string by starting at a “pattern string” and continuing for a specified length – With this option, the ServerIron ADX begins at the starting point in the URL and includes the number of characters specified to form the string used for hashing.

Parsing using the “pattern string” only

The following example identifies a pattern string to perform hashing on with an offset of “0”.

NOTE
Because hash-persist action is a secondary action, you must add a forward action as shown in the following example, before adding a **hash-persist** action.

```
ServerIronADX(config)# csw-policy p1
ServerIronADX(config-csw-p1)# match r1 forward 1
ServerIronADX(config-csw-p1)# match r1 hash-persist url search-host “www.” offset
0
```

Syntax: [no] hash-persist url search-url <pattern-string> offset <offset-value>

The contents of the **<pattern-string>** variable string are used with this command to define the starting place in the hash string portion of the URL string. If the url string has multiple such pattern strings, only the first one will be used. If the pattern string is not found, the system will abort the rest of searching steps, and choose the default hashing method. If the pattern string is found, the

system will start the second step. The second step in the process is to adjust the starting place on the URL string by moving it by the number of characters defined by a value specified in the **<offset-value>**. If the pattern string is empty, the system will not search the pattern part, and use the beginning of URL string as the start point of the third step.

The **<offset-value>** is used to define how many characters will be skipped after the start point that is defined by the pattern string. Normally this value is 0 (zero) which places the start point at the beginning of the host string. A negative value can be used to move the starting place to the left in the host string. If the **<offset-value>** is greater than the length of the rest of the URL, the system will abort the rest of the searching steps, and start the next search (if configured). If another search isn't configured, the default hashing method is used. If the value of the **<offset-value>** variable is within the length of the rest of the URL string, the system will skip to the offset place, and start the third step.

With this command, the third step is for the system to look to the end of the host string to define the string used for hashing.

Parsing using the "pattern string" with a delimiter

The following example identifies a pattern string to perform hashing on with an offset of "0" and a delimiter value of ".".

NOTE
Because hash-persist action is a secondary action, you must add a forward action as shown in the following example, before adding a **hash-persist** action.

```
ServerIronADX(config)# csw-policy p1
ServerIronADX(config-csw-p1)# match r1 forward 1
ServerIronADX(config-csw-p1)# match r1 hash-persist url search-host "www." offset
0 delimiter "."
```

Syntax: [no] hash-persist url search-url <pattern-string> offset <offset-value> delimiter <delimiter-string>

The contents of the **<pattern-string>** variable is the same as described in "Parsing using the "pattern string" only".

The contents and operation of the **<offset-value>** is the same as described in "Parsing using the "pattern string" only".

With this command, the third step is for the system to parse the URL string up-to a character defined by the value of the **<delimiter-string>**. The delimiter string is used to define the end pattern after finding the substring. If the rest of URL string has multiple delimiter strings, only the first one will be used. If the **<delimiter-string>** is not found, or the value of the **<delimiter-string>** variable is empty, the system will look to the end of the host string to define the string used for hashing.

Parsing using the "pattern string" with a specified string length

The following example identifies a pattern string to perform hashing on with an offset of "0" and a length of "8".

NOTE
Because hash-persist action is a secondary action, you must add a forward action as shown in the following example, before adding a **hash-persist** action.

```
ServerIronADX(config)# csw-policy p1
ServerIronADX(config-csw-p1)# match r1 forward 1
```

```
ServerIronADX(config-csw-p1)# match r1 hash-persist url search-host “www.” offset
0 length 5
```

Syntax: [no] hash-persist url search-url <pattern-string> offset <offset-value> length <length>

The contents of the **<pattern-string>** variable is the same as described in “Parsing using the “pattern string” only”.

The contents and operation of the **<offset-value>** is the same as described in “Parsing using the “pattern string” only”.

With this command, the third step is for the system to parse the URL string up-to the number of characters defined by the value of the **<length>** variable. The value of the **<length>** variable must be greater than 0 (zero). If the value of the **<length>** variable extends beyond the length of the host string, the system will look to the end of the host string to define the string used for hashing.

Examples for parsing on the host string of the URL

In the following example, the client tries to connect to Brocade at the following URL.

```
http://www.brocade.com
```

Table 8 displays the contents of the stings used for hashing if the **<pattern-string>** variable is set to “yahoo.com” and the offset value is set to “0” depending on the method used. The methods and commands required are described as follows.

Parsing using the “pattern string” only

```
ServerIronADX(config)# csw-policy p1
ServerIronADX(config-csw-p1)# match r1 forward 1
ServerIronADX(config-csw-p1)# match r1 hash-persist url search-host “www.” offset
0
```

Parsing using the “pattern string” with a delimiter

```
ServerIronADX(config)# csw-policy p1
ServerIronADX(config-csw-p1)# match r1 forward 1
ServerIronADX(config-csw-p1)# match r1 hash-persist url search-host “www.” offset
0 delimiter “.”
```

Parsing using the “pattern string” with a specified string length

```
ServerIronADX(config)# csw-policy p1
ServerIronADX(config-csw-p1)# match r1 forward 1
ServerIronADX(config-csw-p1)# match r1 hash-persist url search-host “www.” offset
0 length 5
```

TABLE 8 Results for parsing the host string of a URL by method

Method	String used for hashing
pattern string only	brocade.com
pattern string with a delimiter of “.”	brocade
pattern string with a length value of 4”	broc

Supporting multiple pattern search for the same rule

ServerIron ADX supports multiple pattern search for the same rule. Support is provided for up to 4 different pattern strings as shown in the following example.

```
ServerIronADX(config)# csw-policy p1
ServerIronADX(config-csw-p1)# match r1 forward 10
ServerIronADX(config-csw-p1)# match r1 hash-persist url search-url “v1” offset 0
ServerIronADX(config-csw-p1)# match r1 hash-persist url search-url “v2” offset 0
ServerIronADX(config-csw-p1)# match r1 hash-persist url search-url “v3” offset 0
delimiter “&”
ServerIronADX(config-csw-p1)# match r1 hash-persist url search-url “v4” offset 0
length 3
ServerIronADX(config-csw-p1)# match r2 forward 20
ServerIronADX(config-csw-p1)# match r2 hash-persist url search-host “m1” offset 0
ServerIronADX(config-csw-p1)# match r2 hash-persist url search-host “m2” offset 0
```

Force rehash

By default, the following will happen when the cache server goes down and goes up again:

- When the cache server goes down, its hash bucket will be taken by other active cache server.
- When a failed cache server comes up, the hash bucket will be allocated if, and only if, there are hash buckets without server association. In most cases, where traffic is evenly spread, none of the hash buckets will be available to the new server.

When a cache server goes down and up, the group hash table is not rehashed and the other server won't be affected. This will make the ServerIron ADX maintain the persistence of the cache servers. Though in a situation where all of the hash table entries are taken by other cache servers, the failed cache server may not be able to process traffic after it comes back to active. This also applies to the situation that the new cache server is added to the group.

You can configure the entire hash table to be re-hashed when a new cache server is added or a failed cache server recovers. Use the csw-force-rehash command within the cache group configuration to enable automatic rehash of hash buckets.

```
ServerIronADX(config)# server cache-group 1
ServerIronADX(config-tc-1)# csw-force-rehash
```

Syntax: [no] csw-force-rehash

NOTE
This command generally disrupts the existing cache persistence.

Traffic distribution based on cache server capacity

In order to determine the health of a cache server, the Brocade ServerIron ADX performs basic health monitoring functions such as check IP connectivity and check reachability of application ports. Basic monitoring of these metrics however, may not provide an aggregated view of the health of cache servers. Brocade ServerIron ADX utilizes an advanced capacity check mechanism for Blue Coat CacheFlow servers by polling the state of Blue Coat servers through SNMP.

The above mentioned capacity check feature is available with Layer-4 switching of TCS traffic. Starting release 12.4.00a release, it can also be enabled in environments where Brocade ServerIron ADX is configured for cache selection through application-specific Layer-7 rules.

The following are some guidelines for using this feature:

- This feature can only be used for HTTP traffic over port 80.
- By default, the cache server state is set to OFFLINE. If an out-of-range SNMP MIB value is received, the cache server state is set to the default state (OFFLINE).
- Only SNMPv2 is currently supported
- Layer 7 SNMP functionality (as shown in Table 11) only applies at the time a new server is selected. It does not affect existing connections.

This feature operates by performing queries to cache server states that are held in the SNMP agent on the cache server. The ServerIron ADX then uses the cache server state obtained from the query to determine the load balancing action to take, as described in Table 9.

TABLE 9 Load balancing action as determined by cache server state

Cache server states	ServerIron ADX load balancing action
UNDERUSED (1): Cache is operating normally with low resource utilization.	For UNDERUSED (1) or NORMAL (2) or BURDENED (3), the action is the same: The ServerIron ADX will allow new connections to these cache servers and will also forward connections that were originally hashed to other stressed cache servers to these cache servers.
NORMAL (2): Cache is operating normally with no resource constraints.	
BURDENED (3): Cache is operating normally but nearing resource constraints.	
STRESSED (4): Cache has reached its effective capacity limit and cannot stably handle additional load.	The ServerIron ADX will not allow new connections to these cache servers if other underused or normal cache servers are available. If no underused or normal cache servers are available, ServerIron ADX will continue to use these cache servers.
OVERTAXED (5): Cache load has exceeded its maximum sustainable capacity and should be decreased.	For OVERTAXED (5) or OVERLOADED (6), the action is the same: The ServerIron ADX will not allow new connections to these cache servers but will continue servicing existing connections. Any new connections will instead be forwarded to the underused and normal cache servers.
OVERLOADED (6): Cache load has reached an unsustainable level and must be decreased immediately.	
HALTING (7): Cache is in the process of shutting down.	The ServerIron ADX will neither allow any new connections nor continue servicing existing connections on these cache servers. Any new connections will instead be forwarded to the underused and normal cache servers.
OFFLINE (8): Cache is out-of-service and unable to handle any traffic.	The ServerIron ADX will neither allow any new connections nor continue servicing existing connections on these cache servers. Any new connections will instead be forwarded to the underused and normal cache servers.
INITIALIZING (9): Cache is preparing to enter service but not ready to handle traffic.	The ServerIron ADX will not allow any new connections to these cache servers.

Configuring SNMP for cache server load

To configure SNMP-based cache server load balancing you must perform the following procedures:

- Within the cache server configuration:
  - Set the SNMP request community string to the same as the cache server.
  - Set the SNMP request oid of the cache server.
- Within the cache group configuration, set the predictor to be SNMP weighted.

The following configuration sets the SNMP request community string and OID.

NOTE
This example shows commands that are valid on the ServerIron ADX device only when it is running the Layer 3 router image.

```
ServerIronADX(config)# server cache-name cs100 20.20.20.100
ServerIronADX(config-rs-cs100)# snmp-request community public
ServerIronADX(config-rs-cs100)# snmp-request oid 1 1.3.6.1.4.1.14501.3.2.1.42.42.0
ServerIronADX(config-rs-cs100)# port http
ServerIronADX(config-rs-cs100)#exit
```

Syntax: [no] snmp-request community <*community-string*>

The <*community-string*> variable specifies the community string name. The string can be up to 32 characters long.

Syntax: [no] snmp-request oid <*oid-index*> <*oid-value*>

The <*oid-index*> variable specifies an index number for the SNMP object identifier that you are configuring.

The <*oid-value*> variable specifies the SNMP object identifier for the cache server.

By default, the ServerIron ADX polls the cache server for the configured OID every three seconds. To change the SNMP poll interval, you can use the following command.

```
ServerIronADX(config)# server snmp-poll 30
```

Syntax: [no] server snmp-poll <*seconds*>

The <*seconds*> variable specifies the poll interval for the SNMP queries sent by the ServerIron ADX to the cache servers. The ServerIron ADX then sets the cache state based on the reply received from the cache server.

The following configuration sets Server Cache Group 1 to use the SNMP predictor configured with the OID index 1 with the cs100 cache server.

```
ServerIronADX(config)# server cache-group 1
ServerIronADX(config-tc-1)# hash-mask 255.255.255.255 0.0.0.0
ServerIronADX(config-tc-1)# cache-name cs100
ServerIronADX(config-tc-1)# predictor snmp-weighted oid 1
```

Syntax: [no] predictor snmp-weighted oid <*oid-index*>

The <*oid-index*> variable specifies the index number for the SNMP object identifier that you want to use for operating SNMP weighted load sharing for this cache group. This must be the same object identifier as that defined under the cache servers.

A typical flow for SNMP-based cache server load balancing would include the following steps.

1. ServerIron ADX sends periodic SNMP queries to the cache server.
2. ServerIron ADX sets the cache server state based on the reply it receives from the cache server.
3. For an incoming connection, a hash is computed based on its load state.

TABLE 10 Hash allocation and load balancing performed by ServerIron ADX doing Layer-4 TCS based on the load state of the cache server

Cache server state	Allocated hash bucket	New connections	Overflow connections**	Existing connections	Clears hash buckets
Underused	Yes	Allows	Allows	Allows	No
Normal	Yes	Allows	Allows	Allows	No
Burdened	Yes	Allows	Allows	Allows	No
Stressed	Yes	Allows (If no other cache is available)	Does not allow	Allows	No
Overloaded/ overtaxed	Yes	Does not allow	Does not allow	Allows	No
Halting	No	Does not allow	Does not allow	Allows	Yes
Offline	No	Does not allow	Does not allow	Deletes*	Yes
Initializing	Yes	Does not allow	Does not allow	Deletes*	No

* In order to delete the existing connection on an offline/initializing cache server immediately the **server force-delete** command needs to be configured.

** Overflow connections indicates that the connection that previously hashed to a cache server that is currently either burdened, stressed, overloaded, overtaxed, halting or offline.

TABLE 11 Transaction allocation and load balancing performed by ServerIron ADX doing Layer-7 TCS based on the load state of the cache server

Cache Server state	New transactions	Existing transactions
Underused	Allows	Allows
Normal	Allows	Allows
Burdened	Allows	Allows
Stressed	Allows (If no other cache in SNMP state 1 to 3 exists. Otherwise, all the transactions will go to caches whose state is in between 1 to 3.	Allows
Overloaded/ overtaxed	Does not allow	Allows
Halting	Does not allow	Allows
Offline	Does not allow	Allows
Initializing	Does not allow	Allows

Displaying cache information

To display cache information, enter the following command at any level of the CLI.

```
Cache-group 1 has 1 members Admin-status = Enabled Active = 0
Hash_info: Dest_mask = 255.255.255.0 Src_mask = 0.0.0.0

Cache Server Name               Admin-status L4-Hash-Buckets L7-Hash-Buckets
cs1                             6           0                 0


Name: cs1             IP: 192.168.1.1          State: 6   Groups =   1

Spoof Enable                          Cache->Web-Server    Web-Server->Cache
                    CurCon TotCon     Packets   Octets     Packets   Octets
Total               0      0          0         0          0         0

                                      Client->Cache        Cache->Client
           State    CurCon TotCon     Packets   Octets     Packets   Octets
http       active   0      0          0         0          0         0
Total               0      0          0         0          0         0



Uncached traffic total
Connection          Client->Web-Server    Web-Server->Client
                    Packets    Octets     Packets    Octets
0                   0          0          0  Cd        0
```

```
ServerIronADX# show cache-group
 Cache-group 1 has 1 members Admin-status = Enabled
 Hash_info: Dest_mask = 255.255.255.0 Src_mask = 0.0.0.0

Cache Server Name   Admin-status Hash-distribution L7-hash-distribution
cs1                 6            4                 21
cs2                 6            3                 22
cs3                 6            4                 21

HTTP Traffic  From <-> to  Web-Caches
=====================================

Name: cf              IP: 209.157.23.195   State: 6   Groups =   1

                                          Host->Web-cache       Web-cache->Host
              State CurConn TotConn    Packets     Octets    Packets     Octets
Client       active       0  386581    1932917  185657048    1547981  393357745
Web-Server   Active       0       0          0          0          0          0
Total                     0  386581    1932917  185657048    1547981  393357745

HTTP Uncached traffic
=====================

Traffic to Web-server port 1
                        Client->Web-Server    Web-Server->Client
Client-port             Packets     Octets    Packets     Octets
 2                         8230     670375       8038    7348299
 4                           97       8129         92      83257
Total                      8327     678504       8130    7431556
```

Syntax: show cache-group

To clear the statistics displayed by the **show cache-group** command, use the following command.

```
ServerIronADX# clear server traffic
```

Syntax: clear server traffic

Table 12 describes the output of the **show cache-group** command.

TABLE 12 TCS information

Field	Description
Global cache group information This section of the display lists global information for the cache group.	
Admin-status	The administrative status of the cache group. The status can be one of the following: • Disabled • Enabled
Hash_info	The source and destination mask for the cache distribution hash value. The ServerIron ADX compares the website's IP address to the hash mask to determine to which cache server to send a request for a given website to. As long as the cache server is available, the ServerIron ADX always sends requests for a given IP address to the same cache. If a cache becomes unavailable, the ServerIron ADX directs requests for websites normally served by that cache to the remaining cache servers until the unavailable cache server becomes available again.

TABLE 12 TCS information (Continued)

Field	Description
Cache Server Name	The names of the cache servers in the cache group. These are the names you assigned when you configured cache server information on the ServerIron ADX.
Admin-status	The administrative state of the cache server, which can be one of the following: • 1 – Enabled • 2 – Failed • 3 – Testing • 4 – Suspect • 5 – Graceful shutdown • 6 – Active
L4-Hash-Buckets	The number of hash distribution slots used by the cache server. The ServerIron ADX has 256 hash distribution slots available. A hash distribution slot associates a website destination IP address with a specific cache server. Refer to "Increasing the TCS hash bucket count" on page 54 for more information about hash values.
L7-Hash-Buckets	The number of Layer 7 cache buckets.
Traffic statistics for traffic between clients and the cache	
Name	The cache server name
IP	The cache server's IP address
State	The administrative state of each of the services on the cache server, which can be one of the following: • 1 – Enabled • 2 – Failed • 3 – Testing • 4 – Suspect • 5 – Graceful shutdown • 6 – Active
Groups	The cache groups of which the cache server is a member.
State	The state of the service, which can be one of the following: • 1 – Enabled • 2 – Failed • 3 – Testing • 4 – Suspect • 5 – Graceful shutdown • 6 – Active
CurConn	The number of currently active connections between hosts and the cache server.
TotConn	The total number of connections between hosts and the cache server.
Cache->Web-Server	The total number of packets and octets from the cache server to web servers.
Web-Server->Cache	The total number of packets and octets from web servers to the cache server.
Client->Cache	The total number of packets and octets from clients to the cache server.
Cache->Client	The total number of packets and octets from the cache server to clients.

TABLE 12 TCS information (Continued)

Field	Description
Traffic statistics for traffic between clients and web servers The statistics for this section are for traffic that did not go to the cache server. Generally, statistics in this section accumulate when the cache server is not available. When the cache server is not available, the ServerIron ADX sends client requests directly to the network or Internet for servicing by the web servers.	
Connection	The total number of connections that ServerIron ADX has sent directly to web servers.
Client->Web-Server	The total number of packets and octets from clients that the ServerIron ADX has sent directly to the web servers.
Web-Server->Client	The total number of packets and octets from web servers to clients.

Sample configurations

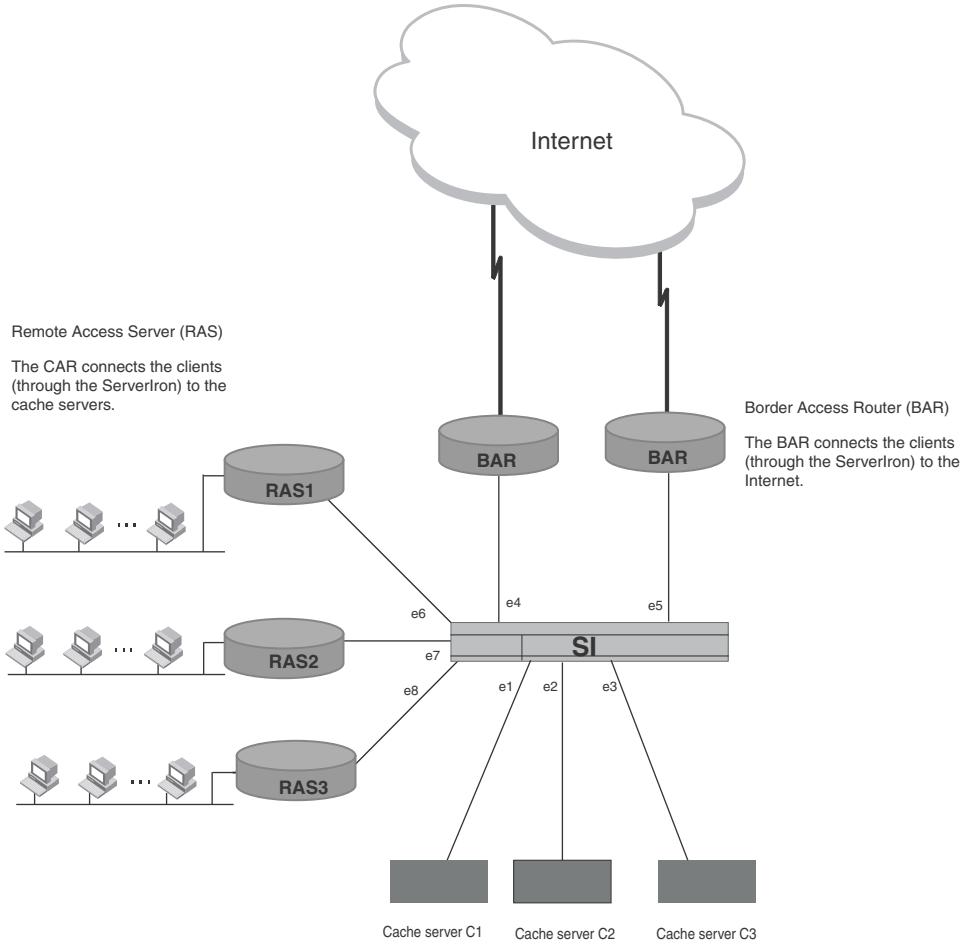
This section describes the following topics:

- Basic TCS configuration
- POP belonging to an ISP using caching to minimize WAN costs
- Policy-based caching
- Asymmetric TCS (FastCache)
- Policy-based cache failover
- TCS with reverse proxy

Basic TCS configuration

Figure 21 shows a configuration in which HTTP traffic flows into a Point-of-Presence (POP) from Remote Access Servers (RASs) and out of the POP to the Internet through a Border Access Router (BAR). The cache servers are labeled C1, C2, and C3.

FIGURE 21 Basic TCS configuration example



In the most basic setup, HTTP traffic flowing across the ServerIron ADX, in any direction, is redirected to the cache servers. If a cache server has the requested content, the server returns the content to the client. If the cache server does not have the content, the cache server goes to the Internet to get the requested content, then caches the content and sends it to the client.

The client never accesses the Internet directly, unless all the cache servers in the cache group are unavailable. In that case, traffic flows across the ServerIron ADX at Layer 2 and out to the Internet in the normal way.

In a transparent caching scheme, the ServerIron ADX acts as the traffic redirector and the cache servers accept requests for any destination IP address. A cache server that accepts requests for any IP address are running in promiscuous mode. The client does not have to configure anything on their web browser. Thus, the caching is “transparent” to the client. It is this transparent characteristic that sets proxy-based caching and transparent caching apart.

In this example, suppose you want all traffic to be cached and you want to use the ServerIron ADX's default settings. To configure the ServerIron ADX for this example, you define the caches, assign them to cache groups, and apply an IP policy.

Applying IP policies

For the simple case in which you want to cache everything no matter where it comes from or where it is going to, use a global policy, such as the following.

```
ServerIronADX(config)# ip policy 1 cache tcp 80 global
```

By using a global policy, you can make rule 2 true for all ports. Rule 1 is true by default because all ports are in cache group 1. Any HTTP traffic flowing across the switch is redirected to the caches.

You can accomplish the same thing with a local policy. With local policies you have to first define and then apply the policy to the appropriate output ports. In this case, since you want to cache all traffic, you need to apply the policy to the RAS and BAR ports.

```
ServerIronADX(config)# ip policy 1 cache tcp 80 local
ServerIronADX(config)# int e 4
ServerIronADX(config-if-4)# ip-policy 1
ServerIronADX(config-if-4)# int e 5
ServerIronADX(config-if-5)# ip-policy 1
ServerIronADX(config-if-5)# int e 6
ServerIronADX(config-if-6)# ip-policy 1
ServerIronADX(config-if-6)# int e 7
ServerIronADX(config-if-7)# ip-policy 1
ServerIronADX(config-if-7)# int e 8
ServerIronADX(config-if-8)# ip-policy 1
```

NOTE
Note the subtle syntax difference between the commands to create a local policy and apply a policy to a port. If you leave the dash out of the command, the command does not work.

The local policies make rule 2 true for the BAR and RAS ports. Rule 1 is true by default. Local policies provide better control at the cost of more configuration steps. If you add a BAR to port 10, traffic destined for it is not redirected because you have not applied the policy to port 10. With a global policy, traffic is redirected automatically.

Defining the caches

To make caching work, you need to apply an IP policy and you need to define the caches and assign them to a cache group. You define cache servers as follows.

```
ServerIronADX(config)# server cache-name C1 11.11.11.11
ServerIronADX(config)# server cache-name C2 11.11.11.12
ServerIronADX(config)# server cache-name C3 11.11.11.13
```

The ServerIronADX ARPs for these addresses to determine which ports the caches are on.

Defining the cache groups

A ***cache group*** is a collection of ServerIron ADX input ports and cache servers. You can define up to four cache groups on a ServerIron ADX. Each cache group can have a cache server farm with up to 256 caches. If a cache group has more than one cache server, the ServerIron ADX distributes traffic among the servers using a hashing algorithm. (Refer to "Increasing the TCS hash bucket count" on page 54.)

All ports on the ServerIron ADX are assigned to cache group 1 by default. Ports can be assigned to any cache group (only one at a time) or removed from all cache groups. If a port is removed from all cache groups, traffic entering on that port is not be redirected to a cache because rule 1 in this example is not true.

Once the caches have been defined, they must be associated (bound) with a particular cache group. The following CLI commands bind the cache servers shown in Figure 21 with a cache group.

```
ServerIronADX(config)# server cache-group 1
ServerIronADX(config-tc-1)# cache-name C1
ServerIronADX(config-tc-1)# cache-name C2
ServerIronADX(config-tc-1)# cache-name C3
```

POP using caching to minimize WAN costs

This example assumes a POP that belongs to an ISP. The RASs are actually remote access routers for customer dial-in. The ISP does not pay the phone company for access to the RASs; the ISP customers pay the phone company for this access. However, the ISP does pay for the WAN links connecting the BARs to the Internet. The ISP wants to introduce caching to improve user response time without the need to increase the size of the WAN links.

The ISP does not want to fill up the cache servers with content in its customer's web sites. The ISP wants to cache only the content on the other side of the BARs. The ISP wants only the traffic entering from a RAS destined for a BAR to be cached. The ISP does not want to cache RAS-to-RAS or BAR-to-RAS traffic.

In this example, the configuration requires more control than a global policy allows. Therefore, local policies are used. Only one cache group, the default cache group 1, is required.

To configure the ServerIron ADX for this application, apply IP policies only to the BAR ports (4 and 5), define the caches, and place them in cache group 1. Here are the CLI commands for creating this configuration.

```
ServerIronADX(config)# server cache-name C1 11.11.11.11
ServerIronADX(config)# server cache-name C2 11.11.11.12
ServerIronADX(config)# server cache-name C3 11.11.11.13
ServerIronADX(config)# ip policy 1 cache tcp 80 local
ServerIronADX(config)# int e 4
ServerIronADX(config-if-4)# ip-policy 1
ServerIronADX(config-if-4)# int e 5
ServerIronADX(config-if-5)# ip-policy 1
ServerIronADX(config-if-5)# ser cache-group 1
ServerIronADX(config-tc-1)# cache-name c1
ServerIronADX(config-tc-1)# cache-name c2
ServerIronADX(config-tc-1)# cache-name c3
```

Traffic entering from a BAR destined for a RAS is not cached because rule 2 (output redirection enabled) is not true for the RAS ports. Traffic from RAS-to-RAS is not cached because rule 2 is false in this case as well. Traffic from RAS-to-BAR is cached because both rules are true.

Both rules are true for BAR-to-BAR traffic as well. This type of traffic rarely, if ever, occurs. However, if this type of traffic does occur and you do not want to cache the traffic, you cannot turn off the output policy on the BAR ports or nothing will get cached. Instead, make rule 1 false by removing the BAR ports from all cache groups. These ports are in the default cache group 1.

```
ServerIronADX(config)# int e 4
ServerIronADX(config-if-4)# no cache-group 1
ServerIronADX(config-if-4)# int e 5
ServerIronADX(config-if-5)# no cache-group 1
```

Now RAS-to-BAR traffic is still cached because the input ports are in the default cache group and the output ports have the IP policy applied. BAR-to-RAS and RAS-to-RAS traffic is not cached because rule 2 is still false. BAR-to-BAR traffic is not cached because rule 1 is false.

Policy-based caching

Policy-based caching enables you to selectively cache some web sites but not others, on specific cache servers. For example, an ISP can use a ServerIron ADX configured for policy-based caching to redirect HTTP traffic to a series of web cache servers made by different vendors with different caching criteria.

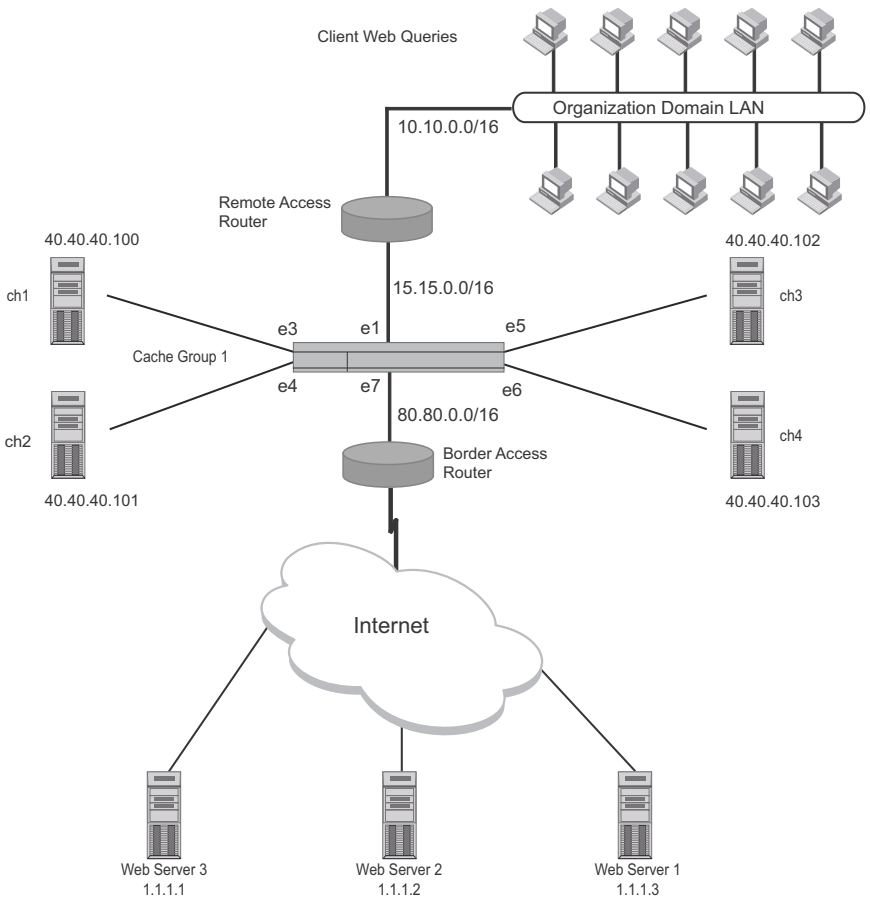
In the example shown in Figure 22, there are four cache servers with 2 cache servers in one group (cache group 1) and 2 cache servers in another group (cache group 2). Policy-based caching is applied for traffic destined to Web Server 1, Web Server 2, and Web Server 3.

Access Control List "101" is tied under Cache Group 1. The **filter-acl 101** command diverts traffic to cache 1 and cache 2 that was originally destined to Web Server 3 (IP address 1.1.1.1).

In the same way, Access List "102" is tied under Cache Group 2. The **filter-acl 102** command diverts all the traffic that was originally destined to Web Server 2 (IP address 1.1.1.2) to cache 3 (ch3) and cache 4 (ch4).

The **server cache-bypass 103** command divert all the traffic to Internet Web Server 3 (IP address 1.1.1.3). The fundamental use of the **server cache-bypass** command is to skip the caching mechanism and send web queries directly to the Internet.

FIGURE 22 Policy-based caching topology



Policy-based caching configuration

The following configuration implements the topology described in Figure 22.

NOTE
This example shows commands that are valid on the ServerIron ADX device only when it is running the Layer 3 router image.

```
ServerIronADX(config)#server cache-name ch1 40.40.40.100
ServerIronADX(config-rs-ch1)#port http
ServerIronADX(config-rs-ch1)#exit
ServerIronADX(config)#server cache-name ch2 40.40.40.101
ServerIronADX(config-rs-ch2)#port http
ServerIronADX(config-rs-ch2)#exit
ServerIronADX(config)#server cache-name ch3 40.40.40.102
ServerIronADX(config-rs-ch3)#port http
ServerIronADX(config-rs-ch3)#exit
ServerIronADX(config)#server cache-name ch4 40.40.40.103
ServerIronADX(config-rs-ch4)#port http
ServerIronADX(config-rs-ch4)#exit

ServerIronADX(config)#server cache-group 1
ServerIronADX(config-tc-1)#cache-name ch1
ServerIronADX(config-tc-1)#cache-name ch2
ServerIronADX(config-tc-1)#filter-acl 101
ServerIronADX(config-tc-1)#exit

ServerIronADX(config)# server cache-group 2
ServerIronADX(config-tc-2)#cache-name ch3
ServerIronADX(config-tc-2)#cache-name ch4
ServerIronADX(config-tc-2)#filter-acl 102
ServerIronADX(config-tc-2)#exit

ServerIronADX(config)# server cache-bypass 103
ServerIronADX(config)# access-list 101 permit tcp any host 1.1.1.1
ServerIronADX(config)# access-list 102 permit tcp any host 1.1.1.2
ServerIronADX(config)# access-list 103 permit tcp any host 1.1.1.3
```

Asymmetric TCS (FastCache)

Traffic in typical TCS configurations passes through the ServerIron ADX both from the client to the cache and from the cache to the client. The ServerIron ADX uses the cache responses to the client to diagnose the health of the cache server.

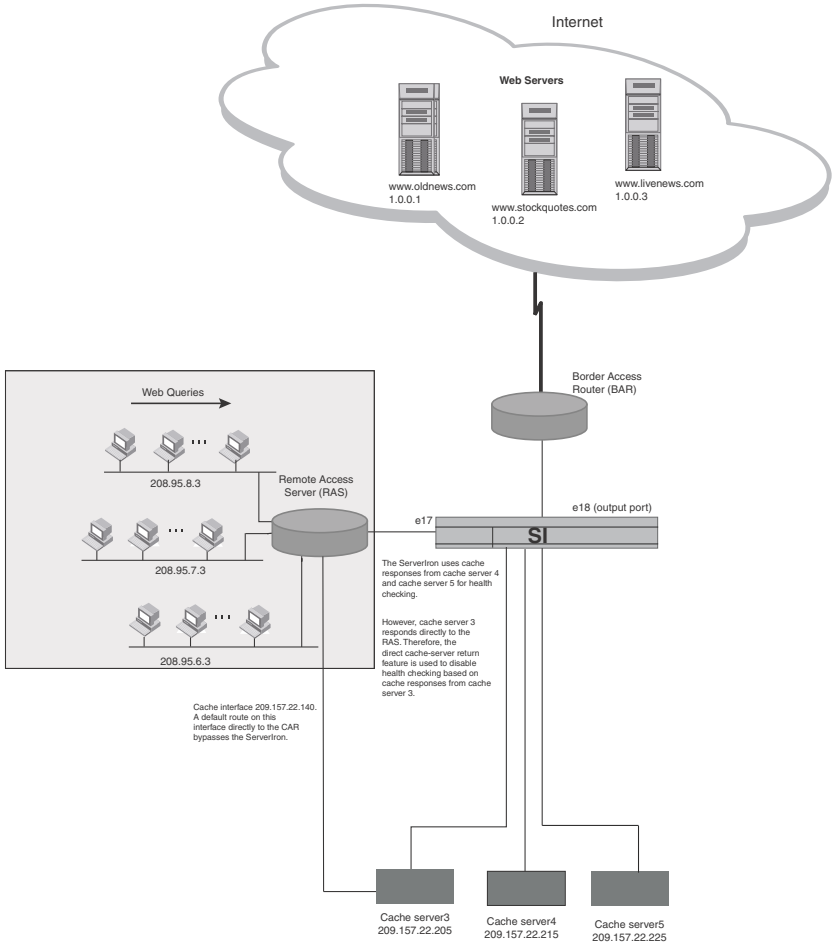
If the cache server responds to the client requests the ServerIron ADX redirects to the cache server, the ServerIron ADX knows that the cache server is healthy. However, if the cache server stops sending replies to the client requests, the ServerIron ADX assumes that the cache server is down and stops redirecting requests to that cache server.

Some configurations are asymmetric—traffic from the cache server to the client does not pass back through the ServerIron ADX. For example, caches that support multiple NICs might at the same time support only one default gateway. Figure 23 shows a configuration in which a cache server's default gateway is configured to go to the customer access router (RAS) instead of the ServerIron ADX. In this configuration, the ServerIron ADX does not see cache responses to client requests. Because the ServerIron ADX does not see responses coming from the cache server, the ServerIron ADX assumes that the cache server is down and stops redirecting requests to that cache server.

You can override this behavior by enabling the FastCache feature. This feature configures the ServerIron ADX to continue redirecting client requests to a cache server even though the ServerIron ADX does not see responses from the cache server. You enable the feature individually for real servers.

NOTE
Even when use the FastCache feature, the ServerIron ADX still performs a Layer 3 health check by regularly pinging the cache server. In addition, you can continue to use HTTP health checking.

FIGURE 23 FastCache feature used for asymmetric topology



Here are the commands for configuring the ServerIron ADX for the topology shown in Figure 23. The line that enables the FastCache feature is shown in bold.

NOTE
This example shows commands that are valid on the ServerIron ADX device only when it is running the Layer 3 router image.

```
ServerIronADX(config)# server cache-name cacheserver3 209.157.22.205
ServerIronADX(config-rs-cacheserver3)# asymmetric
ServerIronADX(config-rs-cacheserver3)# exit
```

```
ServerIronADX(config)# server cache-name cacheserver4 209.157.22.215
ServerIronADX(config-rs-cacheserver4)# exit
ServerIronADX(config)# server cache-name cacheserver5 209.157.22.225
ServerIronADX(config-rs-cacheserver5)# exit
```

This example assumes that the cache contains the contents requested by the client. However, if the cache does not contain the requested page, the cache tries to get the page from the live web site. In this case, the source address for the request is the IP address of the cache server, instead of the IP address of the client. Moreover, this behavior can result in a loop from the cache server to the RAS to the ServerIron ADX and back to the cache server.

To prevent this situation from occurring:

- Define the other interface on the cache server as a cache, but do not place the cache in a cache group.

Policy-based cache failover

In some TCS configurations, the ServerIron ADX is connected to the clients and also to the Internet through the same router. Moreover, in some cases the router contains a policy to forward HTTP requests to a next-hop IP address (virtual IP address) if the packet containing the request matches a filter configured in the router. Cache Failover (CFO) prevents client requests from becoming lost in a “black hole” when the cache servers are unavailable. When you configure the ServerIron ADX for CFO, the ServerIron ADX forwards client requests back to the router for forwarding to the Internet. Thus, clients still receive the requested content even though the cache servers are unavailable.

Normally, cache groups on the ServerIron ADX do not have virtual IP addresses. Instead, the ServerIron ADX selects a cache server from the cache group that contains the port to which the router is connected. Within the cache group, the ServerIron ADX uses a hashing algorithm to select a specific cache server.

NOTE
The virtual servers in SLB use virtual IP addresses, but TCS does not use virtual IP addresses unless you are using CFO.

To configure CFO, make sure you do the following.

1. Set up the router and aim the policy on the router at the virtual address on the ServerIron ADX rather than at the address of the cache.
2. Define the cache or caches on the ServerIron ADX and place them into cache group 1.
3. Define the virtual IP address in cache group 1.
4. Define the IP cache policy as a global cache.

NOTE
For CFO, you must define a global policy, not a local policy.

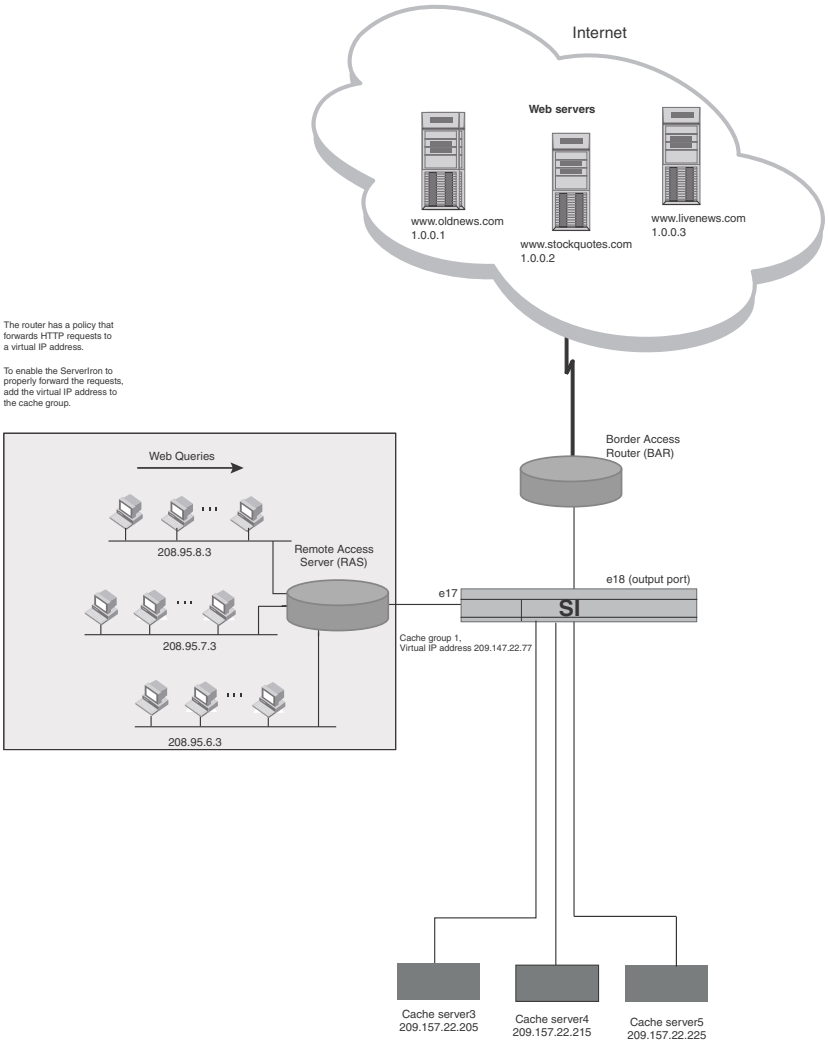
When you add the virtual IP address to the cache group:

- If the cache server to which the ServerIron ADX sends the HTTP traffic has the requested page, the cache server sends the page back to the client, typically through the ServerIron ADX. (This is the normal behavior regardless of whether you have added a virtual IP address.)

- If the cache server is unavailable or does not have the page and thus attempts to send the request back through the ServerIron ADX to the Internet, the ServerIron ADX sends the request to the router for forwarding to the Internet. If the virtual IP address is not configured on the ServerIron ADX, the ServerIron ADX drops the request from the cache server.

Figure 24 shows an example of a configuration that requires CFO.

FIGURE 24 Configuration using policy-based Cache Failover (CFO)



Here are the CLI commands for adding a virtual IP address to a cache group. Add the virtual IP address to which your router forwards the clients' HTTP requests.

```
ServerIronADX(config)# ip policy 1 cache tcp 80 global
ServerIronADX(config)# server cache-group 1
ServerIronADX(config-tc-1)# virtual-ip 209.157.22.77
```

Syntax: [**no**] **virtual-ip** { <ipv4_address> | <ipv6_address> }

The <ipv4_address> variable specifies an IPv4 address for the virtual IP address of the cache group. An IPv4 virtual IP address can only be configured under an IPv4 cache group.

The <ipv6_address> variable specifies an IPv6 address for the virtual IP address of the cache group. An IPv6 virtual IP address can only be configured under an IPv6 cache group.

TCS with reverse proxy

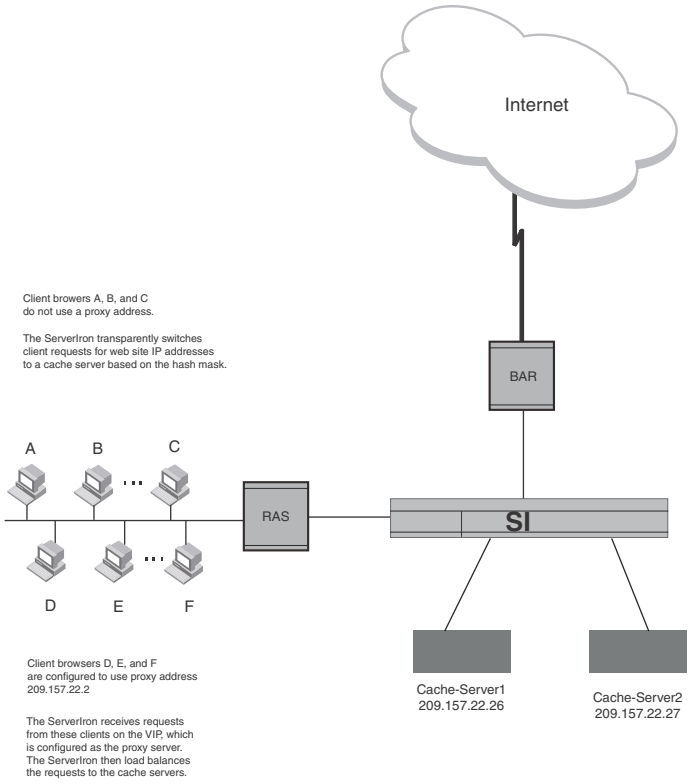
TCS with reverse proxy relieves clients who have configured their web browsers to point to a proxy server from the need to reconfigure their browsers. When you configure the ServerIron ADX for this feature, the ServerIron ADX performs TCS for clients whose browsers do use a proxy and for clients whose browsers do not use a proxy:

- For clients whose browsers do not use a proxy, the ServerIron ADX performs transparent TCS, using the normal hash mechanism to map requests to a cache server based on the source and destination information in the mask and the IP address of the requested site.
- For clients whose browsers use a proxy, the ServerIron ADX load balances the requests across the cache servers. The clients are served transparently by a virtual IP address (VIP) that you configure on the ServerIron ADX with the same IP address as the proxy. Although this is different from the hash mechanism used for transparent TCS, the results for the client are exactly the same. The ServerIron ADX sends the request to a cache server that either has the requested content and sends it back to the client or does not have the requested content but quickly obtains it from the Internet, then sends it back to the client. In addition, the hash mechanism not only distributes traffic, but also ensures that duplication of content is minimized. The hash mechanism minimizes duplication by ensuring that a particular website is always cached on the same cache server.

In either case, the ServerIron ADX provides the requested content to the client.

Figure 25 shows an example of a TCS configuration in which some clients have browsers configured to use a proxy while other clients' browsers are not thus configured.

FIGURE 25 Example Proxy Server Cache Load Balancing Configuration



As shown in Figure 25, some clients' web browsers are configured to use proxy IP address 209.157.22.2, while other client's web browsers are not configured to use a proxy server. You can configure the ServerIron ADX to satisfy both sets of clients.

Follow the steps given below to configure Proxy Server Cache Load Balancing.

1. Add the cache servers as customary, using the **server cache-name** <*string*> <*ip-addr*> command.
2. Add the HTTP ports and configure port-specific health check parameters at the Cache Server level, using the **port http** | <*num*> commands.
3. Create the proxy virtual IP address (VIP) and bind the HTTP ports of the cache servers to the VIP. Use the **server virtual-name-or-ip** <*string*> <*ip-addr*> and bind... commands.
4. Add the cache servers to a cache group using the **server cache-group 1** command.
5. Save the configuration changes to the startup-config file using the **write memory** command.

NOTE
If you have already configured your cache servers and cache group, you do not need to change their configuration. You only need to add the VIP for the proxy and bind the HTTP ports to it, then save the changes.

To configure the ServerIron ADX for the example shown in Figure 25 on page 104, enter the following commands on the ServerIron ADX.

NOTE
This example shows commands that are valid on the ServerIron ADX device only when it is running the Layer 3 router image.

```
ServerIronADX(config)# server port 4199
ServerIronADX(config-port-4199)# tcp
ServerIronADX(config-port-4199)# exit
ServerIronADX(config)# server port 8080
ServerIronADX(config-port-8080)# tcp
ServerIronADX(config-port-8080)# exit
```

The commands above add port profiles for the two HTTP ports in this example that are using port numbers other than the well-known port 80: 4199 and 8080. The **tcp** command at each port's configuration level is required. If you do not identify the ports as TCP ports, the ServerIron ADX assumes the ports are UDP ports and thus does not use an appropriate health check for the ports. You do not need to add a port profile for port 80, since that is the well-known HTTP port.

```
ServerIronADX(config)# server cache-name Cache-Server1 209.157.22.26
ServerIronADX(config-Cache-Server1)# port 4199
ServerIronADX(config-Cache-Server1)# port 8080
ServerIronADX(config-Cache-Server1)# port http
ServerIronADX(config-Cache-Server1)# exit
ServerIronADX(config)# server cache-name Cache-Server2 209.157.22.27
ServerIronADX(config-Cache-Server2)# port 4199
ServerIronADX(config-Cache-Server2)# port 8080
ServerIronADX(config-Cache-Server2)# port http
ServerIronADX(config-Cache-Server2)# exit
```

The commands above add cache servers Cache-Server1 and Cache-Server2. The **port** commands add the HTTP ports to the cache servers. This example does not include optional modification of the HTTP health check parameters for specific servers. For information about customizing an HTTP health check for a specific server.

```
ServerIronADX(config)# server virtual-name-or-ip Proxy 209.157.22.2
ServerIronADX(config-vs-Proxy)# port 4199 sticky
ServerIronADX(config-vs-Proxy)# port 8080 sticky
ServerIronADX(config-vs-Proxy)# bind 4199 Cache-Server1 4199 Cache-Server2 4199
ServerIronADX(config-vs-Proxy)# bind 8080 Cache-Server1 8080 Cache-Server2 8080
ServerIronADX(config-vs-Proxy)# exit
```

The commands above configure a virtual IP address (VIP) to take the place of the Proxy IP address to which some of the client browsers are directing their web requests. The IP address specified with the **server virtual-name-or-ip** command is the IP address that is configured as the proxy on some clients' web browsers. The port 4199 sticky and port 8080 sticky commands add the ports and also make them "sticky". When a port is sticky, once a client session is established on the port, the ServerIron ADX's load balancing mechanism (used for the proxy) sends subsequent packets in the same session to the same cache server. The sticky parameter is not required in this configuration but it can streamline cache performance by keeping client sessions on the same cache servers.

The **bind** commands create table entries in the ServerIron ADX that associate the cache servers and their HTTP ports with the Proxy VIP.

```
ServerIronADX(config)# server cache-group 1
ServerIronADX(config-tc-1)# cache-name Cache-Server1
ServerIronADX(config-tc-1)# cache-name Cache-Server2
ServerIronADX(config-tc-1)# write mem
```

The commands above add the cache servers to a cache group, then save the configuration changes to the ServerIron ADX's startup-config file.

High availability designs with TCS

Layer 3 TCS

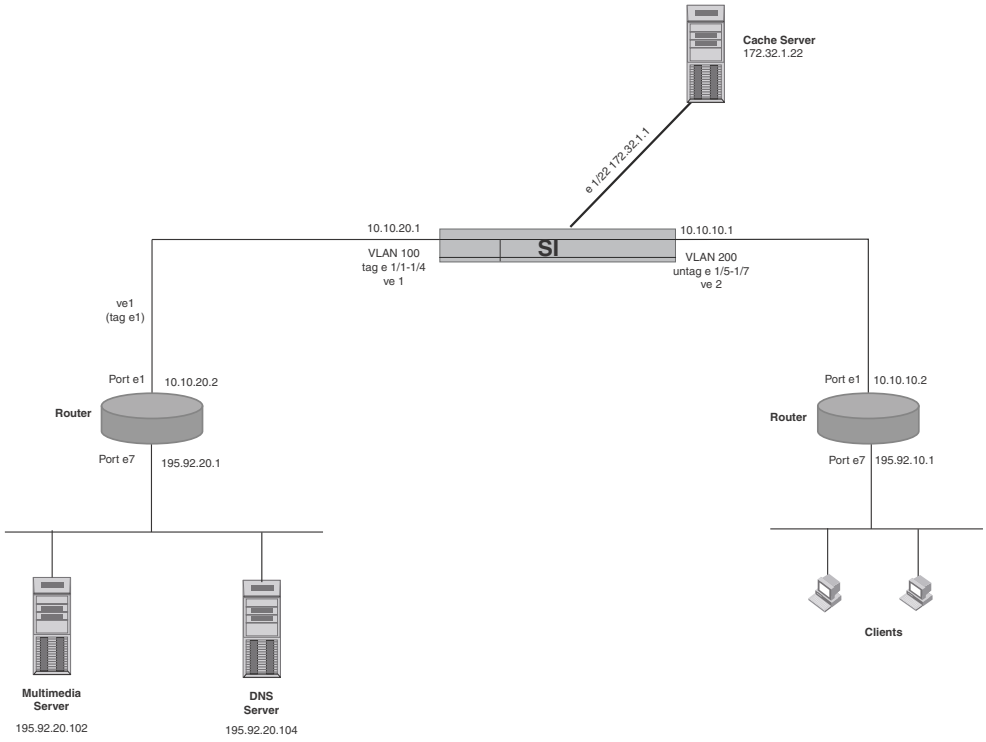
The following sections illustrate Layer 3 TCS support in the following configurations:

- "Layer 3 basic TCS configuration" on page 107
- "Layer 3 active-active TCS configuration" on page 108
- "Layer 3 active-active TCS configuration with a remote cache server" on page 112
- "Layer 3 Sym-Active SLB with TCS" on page 115

Layer 3 basic TCS configuration

Figure 26 illustrates a basic Layer 3 TCS configuration.

FIGURE 26 Basic TCS configuration



The following commands configure the ServerIron ADX in Figure 26.

NOTE
This example shows commands that are valid on the ServerIron ADX device only when it is running the Layer 3 router image.

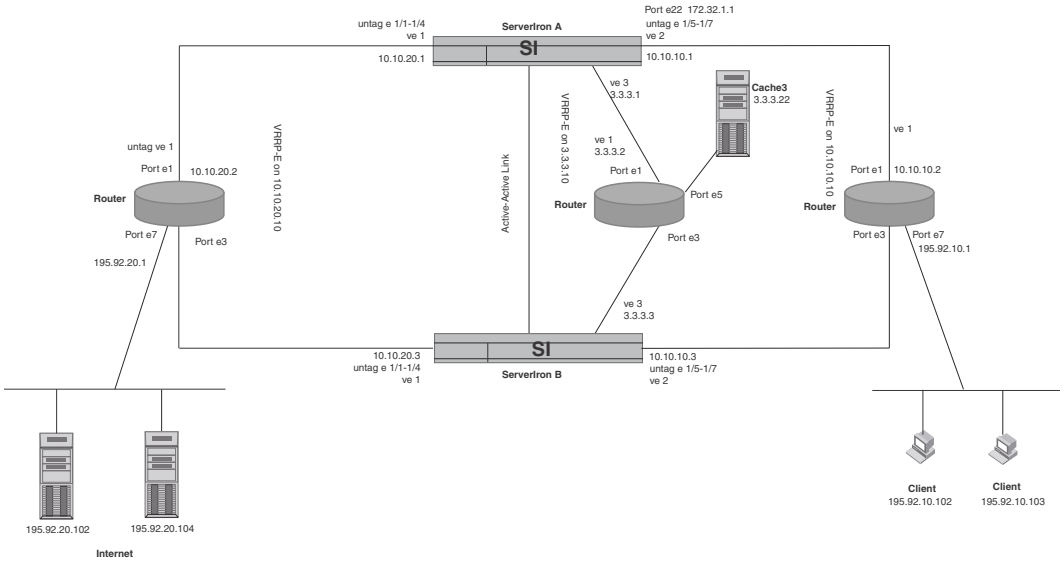
```
ServerIronADX(config)# vlan 1 name DEFAULT-VLAN by port
ServerIronADX(config-vlan-1)# exit
```

```
ServerIronADX(config)# vlan 100 by port
ServerIronADX(config-vlan-100)# untagged ethe 1/1 to 1/4 ethe 2/1 to 2/4
ServerIronADX(config-vlan-100)# router-interface ve 1
ServerIronADX(config-vlan-100)# exit
ServerIronADX(config)# vlan 200 by port
ServerIronADX(config-vlan-200)# untagged ethe 1/5 to 1/7 ethe 2/5 to 2/7
ServerIronADX(config-vlan-200)# router-interface ve 2
ServerIronADX(config-vlan-200)# exit
ServerIronADX(config)# interface ve 1
ServerIronADX(config-ve-1)# ip address 10.10.20.1 255.255.255.0
ServerIronADX(config)# interface ve 2
ServerIronADX(config-ve-2)# ip address 10.10.10.1 255.255.255.0
ServerIronADX(config)# server cache-name cache3 172.32.1.22
ServerIronADX(config-rs-cache3)# port mms
ServerIronADX(config-rs-cache3)# port rtsp
ServerIronADX(config-rs-cache3)# port pnm
ServerIronADX(config-rs-cache3)# port http
ServerIronADX(config-rs-cache3)# port http url "HEAD /"
ServerIronADX(config)# server cache-group 1
ServerIronADX(config-tc-1)# prefer-router-cnt 0
ServerIronADX(config-tc-1)# cache-name cache3
ServerIronADX(config-tc-1)# exit
ServerIronADX(config)# ip l4-policy 1 cache tcp 0 global
ServerIronADX(config)# ip l4-policy 2 cache udp 0 global
ServerIronADX(config)# ip l4-policy 3 cache tcp mms global
ServerIronADX(config)# ip l4-policy 4 cache tcp rtsp global
ServerIronADX(config)# ip l4-policy 5 cache tcp pnm global
ServerIronADX(config)# ip l4-policy 6 cache tcp http global
ServerIronADX(config)# interface ethernet 1/22
ServerIronADX(config-if-1/22)# ip address 172.32.1.1 255.255.255.0
ServerIronADX(config-if-1/22)# exit
```

Layer 3 active-active TCS configuration

Figure 27 illustrates an active-active TCS configuration.

FIGURE 27 Active-active TCS configuration



Commands for ServerIron ADX A

The following commands configure ServerIron ADX A in Figure 27.

NOTE

This example shows commands that are valid on the ServerIron ADX device only when it is running the Layer 3 router image.

```
ServerIronADX(config)# vlan 1 name DEFAULT-VLAN by port
ServerIronADX(config-vlan-1)# exit
ServerIronADX(config)# vlan 100 by port
ServerIronADX(config-vlan-100)# untagged ethe 1/1 to 1/4
ServerIronADX(config-vlan-100)# router-interface ve 1
ServerIronADX(config-vlan-100)# exit
ServerIronADX(config)# vlan 200 by port
ServerIronADX(config-vlan-200)# untagged ethe 1/5 to 1/7
ServerIronADX(config-vlan-200)# router-interface ve 2
ServerIronADX(config-vlan-200)# exit
ServerIronADX(config)# vlan 500 by port
ServerIronADX(config-vlan-500)# untagged ethe 1/8 to 1/12
ServerIronADX(config-vlan-500)# router-interface ve 3
ServerIronADX(config-vlan-500)# exit
ServerIronADX(config)# vlan 16 by port
ServerIronADX(config-vlan-16)# untagged ethe 1/16
ServerIronADX(config-vlan-16)# static-mac-address 00e0.52c2.8b00 ethernet 1/16
ServerIronADX(config-vlan-16)# exit
ServerIronADX(config)# interface ve 1
ServerIronADX(config-ve-1)# ip address 10.10.20.1 255.255.255.0
ServerIronADX(config-ve-1)# ip vrrp-extended vrid 1
ServerIronADX(config-ve-1-vrid-1)# backup
ServerIronADX(config-ve-1-vrid-1)# ip-address 10.10.20.10
ServerIronADX(config-ve-1-vrid-1)# track-port ve 2
ServerIronADX(config-ve-1-vrid-1)# track-port ve 3
ServerIronADX(config-ve-1-vrid-1)# enable
```

```
ServerIronADX(config)# interface ve 2
ServerIronADX(config-ve-2)# ip address 10.10.10.1 255.255.255.0
ServerIronADX(config-ve-2)# ip vrrp-extended vrid 2
ServerIronADX(config-ve-2-vrid-2)# backup
ServerIronADX(config-ve-2-vrid-2)# ip-address 10.10.10.10
ServerIronADX(config-ve-2-vrid-2)# track-port ve 1
ServerIronADX(config-ve-2-vrid-2)# track-port ve 3
ServerIronADX(config-ve-2-vrid-2)# enable
ServerIronADX(config)# interface ve 3
ServerIronADX(config-ve-3)# ip address 3.3.3.1 255.255.255.0
ServerIronADX(config-ve-3)# ip vrrp-extended vrid 3
ServerIronADX(config-ve-3-vrid-3)# backup
ServerIronADX(config-ve-3-vrid-3)# ip-address 3.3.3.10
ServerIronADX(config-ve-3-vrid-3)# track-port ve 1
ServerIronADX(config-ve-3-vrid-3)# track-port ve 2
ServerIronADX(config-ve-3-vrid-3)# enable
ServerIronADX(config)# server cache-name cache3 3.3.3.22
ServerIronADX(config-rs-cache2)# port mms
ServerIronADX(config-rs-cache2)# port rtsp
ServerIronADX(config-rs-cache2)# port pnm
ServerIronADX(config-rs-cache2)# port http
ServerIronADX(config-rs-cache2)# port http url "HEAD /"
ServerIronADX(config)# server cache-group 1
ServerIronADX(config-tc-1)# prefer-router-cnt 0
ServerIronADX(config-tc-1)# cache-name cache2
ServerIronADX(config-tc-1)# no http-cache-control
ServerIronADX(config-tc-1)# exit
ServerIronADX(config)# ip l4-policy 1 cache tcp 0 global
ServerIronADX(config)# ip l4-policy 2 cache udp 0 global
ServerIronADX(config)# ip l4-policy 3 cache tcp mms global
ServerIronADX(config)# ip l4-policy 4 cache tcp rtsp global
ServerIronADX(config)# ip l4-policy 5 cache tcp pnm global
ServerIronADX(config)# ip l4-policy 6 cache tcp http global
ServerIronADX(config)# ip route 195.92.20.0 255.255.255.0 10.10.20.2
ServerIronADX(config)# ip route 195.92.10.0 255.255.255.0 10.10.10.2
ServerIronADX(config)# router vrrp-extended
ServerIronADX(config)# server active-active-port ethe 1/16 vlan-id 16
ServerIronADX(config)# server force-delete
ServerIronADX(config)# no server l4-check
```

Commands for ServerIron ADX B

The following commands configure ServerIron ADX B in Figure 27.

NOTE

This example shows commands that are valid on the ServerIron ADX device only when it is running the Layer 3 router image.

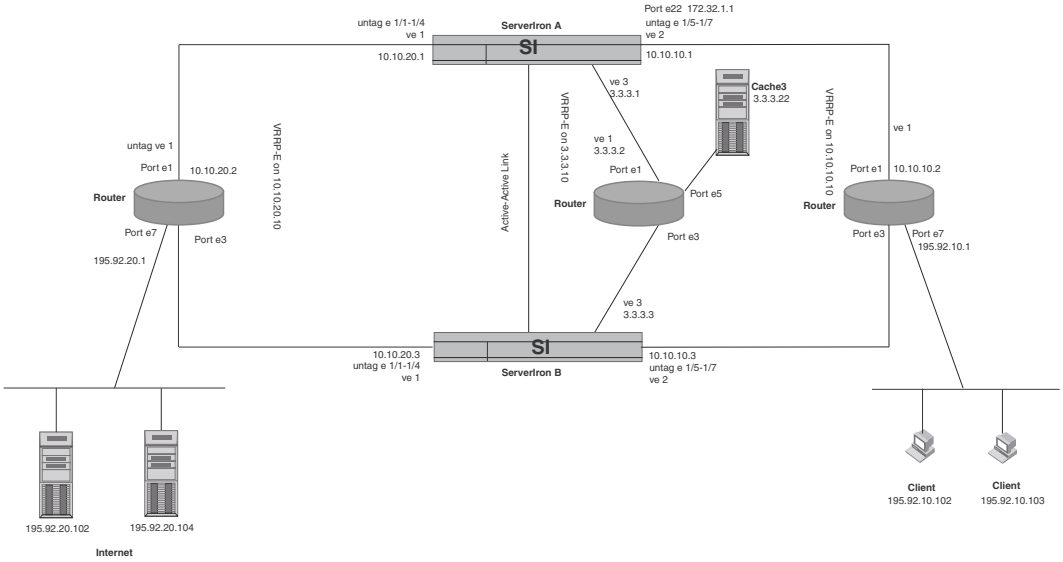
```
ServerIronADX(config)# vlan 1 name DEFAULT-VLAN by port
ServerIronADX(config-vlan-1)# exit
ServerIronADX(config)# vlan 100 by port
ServerIronADX(config-vlan-100)# untagged ethe 1/1 to 1/4
ServerIronADX(config-vlan-100)# router-interface ve 1
ServerIronADX(config-vlan-100)# exit
ServerIronADX(config)# vlan 200 by port
ServerIronADX(config-vlan-200)# untagged ethe 1/5 to 1/7
ServerIronADX(config-vlan-200)# router-interface ve 2
ServerIronADX(config-vlan-200)# exit
```

```
ServerIronADX(config)# vlan 500 by port
ServerIronADX(config-vlan-500)# untagged ethe 1/8 to 1/12
ServerIronADX(config-vlan-500)# router-interface ve 3
ServerIronADX(config-vlan-500)# exit
ServerIronADX(config)# vlan 16 by port
ServerIronADX(config-vlan-16)# untagged ethe 1/16
ServerIronADX(config-vlan-16)# static-mac-address 00e0.52ee.6900 ethernet 1/16
ServerIronADX(config-vlan-16)# exit
ServerIronADX(config)# interface ve 1
ServerIronADX(config-ve-1)# ip address 10.10.20.3 255.255.255.0
ServerIronADX(config-ve-1)# ip vrrp-extended vrid 1
ServerIronADX(config-ve-1-vrid-1)# backup
ServerIronADX(config-ve-1-vrid-1)# ip-address 10.10.20.10
ServerIronADX(config-ve-1-vrid-1)# track-port ve 2
ServerIronADX(config-ve-1-vrid-1)# track-port ve 3
ServerIronADX(config-ve-1-vrid-1)# enable
ServerIronADX(config)# interface ve 2
ServerIronADX(config-ve-2)# ip address 10.10.10.3 255.255.255.0
ServerIronADX(config-ve-2)# ip vrrp-extended vrid 2
ServerIronADX(config-ve-2-vrid-2)# backup
ServerIronADX(config-ve-2-vrid-2)# ip-address 10.10.10.10
ServerIronADX(config-ve-2-vrid-2)# track-port ve 1
ServerIronADX(config-ve-2-vrid-2)# track-port ve 3
ServerIronADX(config-ve-2-vrid-2)# enable
ServerIronADX(config)# interface ve 3
ServerIronADX(config-ve-3)# ip address 3.3.3.3 255.255.255.0
ServerIronADX(config-ve-3)# ip vrrp-extended vrid 3
ServerIronADX(config-ve-3-vrid-3)# backup
ServerIronADX(config-ve-3-vrid-3)# ip-address 3.3.3.10
ServerIronADX(config-ve-3-vrid-3)# track-port ve 1
ServerIronADX(config-ve-3-vrid-3)# track-port ve 2
ServerIronADX(config-ve-3-vrid-3)# enable
ServerIronADX(config)# server cache-name cache3 3.3.3.22
ServerIronADX(config-rs-cache3)# port mms
ServerIronADX(config-rs-cache3)# port rtsp
ServerIronADX(config-rs-cache3)# port pnm
ServerIronADX(config-rs-cache3)# port http
ServerIronADX(config-rs-cache3)# port http url "HEAD /"
ServerIronADX(config)# server cache-group 1
ServerIronADX(config-tc-1)# prefer-router-cnt 0
ServerIronADX(config-tc-1)# cache-name cache3
ServerIronADX(config-tc-1)# no http-cache-control
ServerIronADX(config-tc-1)# exit
ServerIronADX(config)# ip l4-policy 1 cache tcp 0 global
ServerIronADX(config)# ip l4-policy 2 cache udp 0 global
ServerIronADX(config)# ip l4-policy 3 cache tcp mms global
ServerIronADX(config)# ip l4-policy 4 cache tcp rtsp global
ServerIronADX(config)# ip l4-policy 5 cache tcp pnm global
ServerIronADX(config)# ip l4-policy 6 cache tcp http global
ServerIronADX(config)# ip route 195.92.10.0 255.255.255.0 10.10.10.2
ServerIronADX(config)# ip route 195.92.20.0 255.255.255.0 10.10.20.2
ServerIronADX(config)# router vrrp-extended
ServerIronADX(config)# server active-active-port ethe 1/16 vlan-id 16
ServerIronADX(config)# server force-delete
ServerIronADX(config)# no server l4-check
```

Layer 3 active-active TCS configuration with a remote cache server

Figure 28 illustrates an active-active TCS configuration with a remote cache server.

FIGURE 28 Active-active TCS configuration with remote cache server



Commands for ServerIron ADX A

The following commands configure ServerIron ADX A in Figure 28.

NOTE

This example shows commands that are valid on the ServerIron ADX device only when it is running the Layer 3 router image.

```
ServerIronADX(config)# vlan 1 name DEFAULT-VLAN by port
ServerIronADX(config-vlan-1)# exit
ServerIronADX(config)# vlan 100 by port
ServerIronADX(config-vlan-100)# untagged ethe 1/1 to 1/4
ServerIronADX(config-vlan-100)# router-interface ve 1
ServerIronADX(config-vlan-100)# exit
ServerIronADX(config)# vlan 200 by port
ServerIronADX(config-vlan-200)# untagged ethe 1/5 to 1/7
ServerIronADX(config-vlan-200)# router-interface ve 2
ServerIronADX(config-vlan-200)# exit
ServerIronADX(config)# vlan 500 by port
ServerIronADX(config-vlan-500)# untagged ethe 1/8 to 1/12
ServerIronADX(config-vlan-500)# router-interface ve 3
ServerIronADX(config-vlan-500)# exit
ServerIronADX(config)# vlan 16 by port
ServerIronADX(config-vlan-16)# untagged ethe 1/16
ServerIronADX(config-vlan-16)# static-mac-address 00e0.52ee.6900 ethernet 1/16
ServerIronADX(config-vlan-16)# exit
ServerIronADX(config)# interface ve 1
ServerIronADX(config-ve-1)# ip address 10.10.20.1 255.255.255.0
ServerIronADX(config-ve-1)# ip vrrp-extended vrid 1
ServerIronADX(config-ve-1-vrid-1)# backup
```

```
ServerIronADX(config-ve-1-vrid-1)# ip-address 10.10.20.10
ServerIronADX(config-ve-1-vrid-1)# track-port ve 2
ServerIronADX(config-ve-1-vrid-1)# track-port ve 3
ServerIronADX(config-ve-1-vrid-1)# enable
ServerIronADX(config)# interface ve 2
ServerIronADX(config-ve-2)# ip address 10.10.10.1 255.255.255.0
ServerIronADX(config-ve-2)# ip vrrp-extended vrid 2
ServerIronADX(config-ve-2-vrid-2)# backup
ServerIronADX(config-ve-2-vrid-2)# ip-address 10.10.10.10
ServerIronADX(config-ve-2-vrid-2)# track-port ve 1
ServerIronADX(config-ve-2-vrid-2)# track-port ve 3
ServerIronADX(config-ve-2-vrid-2)# enable
ServerIronADX(config)# interface ve 3
ServerIronADX(config-ve-3)# ip address 3.3.3.1 255.255.255.0
ServerIronADX(config-ve-3)# ip vrrp-extended vrid 3
ServerIronADX(config-ve-3-vrid-3)# backup
ServerIronADX(config-ve-3-vrid-3)# ip-address 3.3.3.10
ServerIronADX(config-ve-3-vrid-3)# track-port ve 1
ServerIronADX(config-ve-3-vrid-3)# track-port ve 2
ServerIronADX(config-ve-3-vrid-3)# enable
ServerIronADX(config)# server cache-name cache1 172.32.1.20
ServerIronADX(config-rs-cache1)# remote-cache
ServerIronADX(config-rs-cache1)# port mms
ServerIronADX(config-rs-cache1)# port rtsp
ServerIronADX(config-rs-cache1)# port pnm
ServerIronADX(config-rs-cache1)# port http
ServerIronADX(config-rs-cache1)# port http url "HEAD /"
ServerIronADX(config)# server cache-group 1
ServerIronADX(config-tc-1)# prefer-router-cnt 0
ServerIronADX(config-tc-1)# dest-nat
ServerIronADX(config-tc-1)# cache-name cache1
ServerIronADX(config-tc-1)# no http-cache-control
ServerIronADX(config-tc-1)# exit
ServerIronADX(config)# ip l4-policy 1 cache tcp 0 global
ServerIronADX(config)# ip l4-policy 2 cache udp 0 global
ServerIronADX(config)# ip l4-policy 3 cache tcp mms global
ServerIronADX(config)# ip l4-policy 4 cache tcp rtsp global
ServerIronADX(config)# ip l4-policy 5 cache tcp pnm global
ServerIronADX(config)# ip l4-policy 6 cache tcp http global
ServerIronADX(config)# ip route 172.32.1.0 255.255.255.0 3.3.3.4
ServerIronADX(config)# router vrrp-extended
ServerIronADX(config)# server active-active-port ethe 1/16 vlan-id 16
ServerIronADX(config)# server force-delete
ServerIronADX(config)# no server l4-check
```

Commands for ServerIron ADX B

The following commands configure ServerIron ADX B in Figure 28.

NOTE

This example shows commands that are valid on the ServerIron ADX device only when it is running the Layer 3 router image.

```
ServerIronADX(config)# vlan 1 name DEFAULT-VLAN by port
ServerIronADX(config-vlan-1)# exit
ServerIronADX(config)# vlan 100 by port
ServerIronADX(config-vlan-100)# untagged ethe 1/1 to 1/4
ServerIronADX(config-vlan-100)# router-interface ve 1
ServerIronADX(config-vlan-100)# exit
```

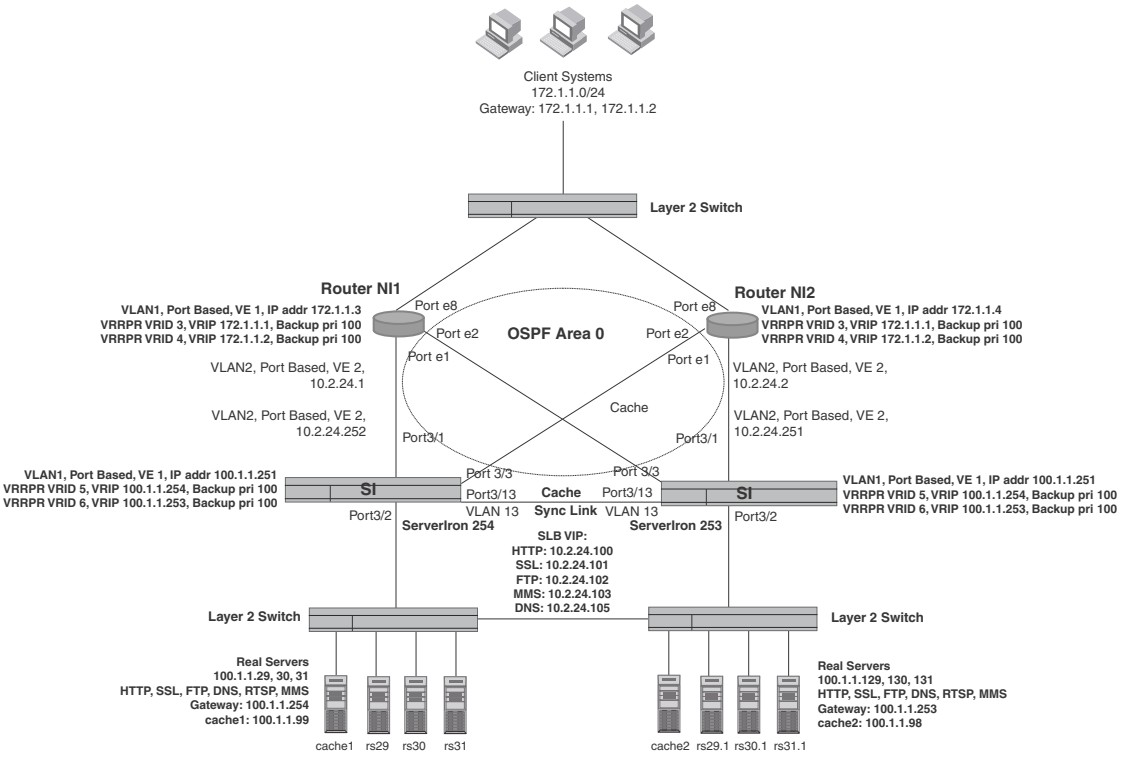
```
ServerIronADX(config)# vlan 200 by port
ServerIronADX(config-vlan-200)# untagged ethe 1/5 to 1/7
ServerIronADX(config-vlan-200)# router-interface ve 2
ServerIronADX(config-vlan-200)# exit
ServerIronADX(config)# vlan 500 by port
ServerIronADX(config-vlan-500)# untagged ethe 1/8 to 1/12
ServerIronADX(config-vlan-500)# router-interface ve 3
ServerIronADX(config-vlan-500)# exit
ServerIronADX(config)# vlan 16 by port
ServerIronADX(config-vlan-16)# untagged ethe 1/16
ServerIronADX(config-vlan-16)# static-mac-address 00e0.52ee.d600 ethernet 1/16
ServerIronADX(config-vlan-16)# exit
ServerIronADX(config)# interface ve 1
ServerIronADX(config-ve-1)# ip address 10.10.20.3 255.255.255.0
ServerIronADX(config-ve-1)# ip vrrp-extended vrid 1
ServerIronADX(config-ve-1-vrid-1)# backup
ServerIronADX(config-ve-1-vrid-1)# ip-address 10.10.20.10
ServerIronADX(config-ve-1-vrid-1)# track-port ve 2
ServerIronADX(config-ve-1-vrid-1)# track-port ve 3
ServerIronADX(config-ve-1-vrid-1)# enable
ServerIronADX(config)# interface ve 2
ServerIronADX(config-ve-2)# ip address 10.10.10.3 255.255.255.0
ServerIronADX(config-ve-2)# ip vrrp-extended vrid 2
ServerIronADX(config-ve-2-vrid-2)# backup
ServerIronADX(config-ve-2-vrid-2)# ip-address 10.10.10.10
ServerIronADX(config-ve-2-vrid-2)# track-port ve 1
ServerIronADX(config-ve-2-vrid-2)# track-port ve 3
ServerIronADX(config-ve-2-vrid-2)# enable
ServerIronADX(config)# interface ve 3
ServerIronADX(config-ve-3)# ip address 3.3.3.3 255.255.255.0
ServerIronADX(config-ve-3)# ip vrrp-extended vrid 3
ServerIronADX(config-ve-3-vrid-3)# backup
ServerIronADX(config-ve-3-vrid-3)# ip-address 3.3.3.10
ServerIronADX(config-ve-3-vrid-3)# track-port ve 1
ServerIronADX(config-ve-3-vrid-3)# track-port ve 2
ServerIronADX(config-ve-3-vrid-3)# enable
ServerIronADX(config)# server cache-name cache1 172.32.1.20
ServerIronADX(config-rs-cache1)# remote-cache
ServerIronADX(config-rs-cache1)# port mms
ServerIronADX(config-rs-cache1)# port rtsp
ServerIronADX(config-rs-cache1)# port pnm
ServerIronADX(config-rs-cache1)# port http
ServerIronADX(config-rs-cache1)# port http url "HEAD /"
ServerIronADX(config)# server cache-group 1
ServerIronADX(config-tc-1)# prefer-router-cnt 0
ServerIronADX(config-tc-1)# dest-nat
ServerIronADX(config-tc-1)# cache-name cache1
ServerIronADX(config-tc-1)# no http-cache-control
ServerIronADX(config-tc-1)# exit
ServerIronADX(config)# ip l4-policy 1 cache tcp 0 global
ServerIronADX(config)# ip l4-policy 2 cache udp 0 global
ServerIronADX(config)# ip l4-policy 3 cache tcp mms global
ServerIronADX(config)# ip l4-policy 4 cache tcp rtsp global
ServerIronADX(config)# ip l4-policy 5 cache tcp pnm global
ServerIronADX(config)# ip l4-policy 6 cache tcp http global
ServerIronADX(config)# ip route 172.32.1.0 255.255.255.0 3.3.3.4
ServerIronADX(config)# router vrrp-extended
ServerIronADX(config)# server active-active-port ethe 1/16 vlan-id 16
ServerIronADX(config)# server force-delete
ServerIronADX(config)# no server l4-check
```

Layer 3 Sym-Active SLB with TCS

Figure 29 illustrates a Sym-Active configuration with TCS, using VRRPE.

NOTE
To allow failover and session synchronization in an Sym-Active configuration to work properly, there must be a Layer 2 connection between the two ServerIron ADXs. This connection is required so that Layer 2 broadcasts, including ARP to the VIP from the ServerIron ADX with lower symmetric priority, can be exchanged between the two ServerIron ADXs. In configurations with multiple VLANs, the Layer 2 link must be on the sub-net where the VIPs are configured.

FIGURE 29 Sym-Active configuration with TCS and VRRPE



Commands for router NI1

The following commands configure router NI1 in Figure 29,

NOTE
The **route-only** command is omitted on this NetIron's configuration to allow Layer 2 connectivity between the two ServerIron ADXs on the VIP's sub-net.

NOTE
This example shows commands that are valid on the ServerIron ADX device only when it is running the Layer 3 router image.

```
NetIron(config)# vlan 1 name DEFAULT-VLAN by port
NetIron(config-vlan-1)# router-interface ve 1
NetIron(config-vlan-1)# exit
NetIron(config)# vlan 2 by port
NetIron(config-vlan-2)# untagged ethe 1 to 2
NetIron(config-vlan-2)# router-interface ve 2
NetIron(config-vlan-2)# exit
NetIron(config)# vlan 23 by port
NetIron(config-vlan-23)# untagged ethe 23
NetIron(config-vlan-23)# exit
NetIron(config)# interface ve 1
NetIron(config-ve-1)# ip address 172.1.1.3 255.255.255.0
NetIron(config-ve-1)# ip ospf area 0
NetIron(config-ve-1)# ip vrrp-extended vrid 3
NetIron(config-ve-1-vrid-3)# backup
NetIron(config-ve-1-vrid-3)# ip-address 172.1.1.1
NetIron(config-ve-1-vrid-3)# track-port e 1
NetIron(config-ve-1-vrid-3)# track-port e 2
NetIron(config-ve-1-vrid-3)# track-port e 8
NetIron(config-ve-1-vrid-3)# enable
NetIron(config-ve-1)# ip vrrp-extended vrid 4
NetIron(config-ve-1-vrid-4)# backup
NetIron(config-ve-1-vrid-4)# ip-address 172.1.1.2
NetIron(config-ve-1-vrid-4)# track-port e 1
NetIron(config-ve-1-vrid-4)# track-port e 2
NetIron(config-ve-1-vrid-4)# track-port e 8
NetIron(config-ve-1-vrid-4)# enable
NetIron(config-ve-1)# exit
NetIron(config)# interface ve 2
NetIron(config-ve-2)# ip address 10.2.24.1 255.255.255.0
NetIron(config-ve-2)# ip ospf area 0
NetIron(config-ve-2)# exit
NetIron(config)# interface ethernet 23
NetIron(config-if-23)# ip address 173.1.1.1 255.255.255.0
NetIron(config-if-23)# exit
NetIron(config)# ip route 0.0.0.0 0.0.0.0 10.2.24.254
NetIron(config)# ip router-id 10.2.24.1
NetIron(config)# router vrrp-extended
NetIron(config)# router ospf
NetIron(config-ospf-router)# area 0
NetIron(config-ospf-router)# redistribution connected
NetIron(config-ospf-router)# redistribution static
NetIron(config-ospf-router)# exit
```

Commands for ServerIron ADX 254

The following commands configure ServerIron ADX 254 in Figure 29.

NOTE

This example shows commands that are valid on the ServerIron ADX device only when it is running the Layer 3 router image.

```
ServerIronADX(config)# vlan 1 name DEFAULT-VLAN by port
ServerIronADX(config-vlan-1)# router-interface ve 1
ServerIronADX(config-vlan-1)# exit
ServerIronADX(config)# vlan 2 by port
ServerIronADX(config-vlan-2)# untagged ethe 3/1 ethe 3/3 ethe 4/1 ethe 4/3
ServerIronADX(config-vlan-2)# router-interface ve 2
ServerIronADX(config-vlan-2)# exit
```

```
ServerIronADX(config)# vlan 16 by port
ServerIronADX(config-vlan-16)# untagged ethe 3/16
ServerIronADX(config-vlan-16)# static-mac-address 00e0.5212.3400 ethernet 3/16
ServerIronADX(config-vlan-16)# exit
ServerIronADX(config)# vlan 999 by port
ServerIronADX(config-vlan-999)# untagged ethe 3/24
ServerIronADX(config-vlan-999)# exit
ServerIronADX(config)# interface ve 1
ServerIronADX(config-ve-1)# ip address 100.1.1.252 255.255.255.0
ServerIronADX(config-ve-1)# ip vrrp-extended vrid 5
ServerIronADX(config-ve-1-vrid-5)# backup
ServerIronADX(config-ve-1-vrid-5)# ip-address 100.1.1.254
ServerIronADX(config-ve-1-vrid-5)# track-port e 3/1
ServerIronADX(config-ve-1-vrid-5)# track-port e 3/2
ServerIronADX(config-ve-1-vrid-5)# track-port e 3/3
ServerIronADX(config-ve-1-vrid-5)# enable
ServerIronADX(config-ve-1)# ip vrrp-extended vrid 6
ServerIronADX(config-ve-1-vrid-6)# backup
ServerIronADX(config-ve-1-vrid-6)# ip-address 100.1.1.253
ServerIronADX(config-ve-1-vrid-6)# track-port e 3/1
ServerIronADX(config-ve-1-vrid-6)# track-port e 3/2
ServerIronADX(config-ve-1-vrid-6)# track-port e 3/3
ServerIronADX(config-ve-1-vrid-6)# enable
ServerIronADX(config-ve-1-vrid-6)# exit
ServerIronADX(config)# interface ve 2
ServerIronADX(config-ve-2)# ip address 10.2.24.252 255.255.255.0
ServerIronADX(config-ve-2)# ip ospf area 0
ServerIronADX(config-ve-2)# exit
ServerIronADX(config)# ip l4-policy 1 cache tcp 0 global
ServerIronADX(config)# ip l4-policy 2 cache udp 0 global
ServerIronADX(config)# ip l4-policy 3 cache tcp http global
ServerIronADX(config)# ip l4-policy 4 cache tcp ftp global
ServerIronADX(config)# ip l4-policy 5 cache tcp mms global
ServerIronADX(config)# router vrrp-extended
ServerIronADX(config)# server active-active-port ethe 3/13 vlan-id 13
ServerIronADX(config)# router ospf
ServerIronADX(config-ospf-router)# area 0
ServerIronADX(config-ospf-router)# redistribution connected
ServerIronADX(config-ospf-router)# exit
```

The following commands enable session synchronization on the ports where the active-active SLB feature is used. This is required both to ensure continued service following a failover and to enable each ServerIron ADX to send server replies back to the clients, regardless of which ServerIron ADX load balanced the request.

```
ServerIronADX(config)# server port 80
ServerIronADX(config-port-80)# session-sync
ServerIronADX(config-port-80)# tcp
ServerIronADX(config-port-80)# exit
ServerIronADX(config)# server port 21
ServerIronADX(config-port-21)# session-sync
ServerIronADX(config-port-21)# exit
ServerIronADX(config)# server port 1755
ServerIronADX(config-port-1755)# session-sync
ServerIronADX(config-port-1755)# tcp
ServerIronADX(config-port-1755)# udp
ServerIronADX(config-port-1755)# exit
ServerIronADX(config)# server port 53
ServerIronADX(config-port-53)# session-sync
ServerIronADX(config-port-53)# exit
```

```
ServerIronADX(config)# server port 443
ServerIronADX(config-port-443)# session-sync
ServerIronADX(config-port-443)# tcp
ServerIronADX(config-port-443)# exit
ServerIronADX(config)# server router-ports ethernet 3/1
ServerIronADX(config)# server router-ports ethernet 3/3
ServerIronADX(config)# server real rs29 100.1.1.29
ServerIronADX(config-rs-rs29)# port ssl
ServerIronADX(config-rs-rs29)# port mms
ServerIronADX(config-rs-rs29)# port http
ServerIronADX(config-rs-rs29)# port http url "HEAD /"
ServerIronADX(config-rs-rs29)# port ftp
ServerIronADX(config-rs-rs29)# port dns
ServerIronADX(config-rs-rs29)# exit
ServerIronADX(config)# server real rs30 100.1.1.30
ServerIronADX(config-rs-rs30)# port ssl
ServerIronADX(config-rs-rs30)# port mms
ServerIronADX(config-rs-rs30)# port http
ServerIronADX(config-rs-rs30)# port http url "HEAD /"
ServerIronADX(config-rs-rs30)# port ftp
ServerIronADX(config-rs-rs30)# port dns
ServerIronADX(config-rs-rs30)# exit
ServerIronADX(config)# server real rs31 100.1.1.31
ServerIronADX(config-rs-rs31)# port ssl
ServerIronADX(config-rs-rs31)# port mms
ServerIronADX(config-rs-rs31)# port http
ServerIronADX(config-rs-rs31)# port http url "HEAD /"
ServerIronADX(config-rs-rs31)# port ftp
ServerIronADX(config-rs-rs31)# port dns
ServerIronADX(config-rs-rs31)# exit
ServerIronADX(config)# server real rs29.1 100.1.1.129
ServerIronADX(config-rs-rs29.1)# port dns
ServerIronADX(config-rs-rs29.1)# port ftp
ServerIronADX(config-rs-rs29.1)# port http
ServerIronADX(config-rs-rs29.1)# port http url "HEAD /"
ServerIronADX(config-rs-rs29.1)# port mms
ServerIronADX(config-rs-rs29.1)# port ssl
ServerIronADX(config-rs-rs29.1)# exit
ServerIronADX(config)# server real rs30.1 100.1.1.130
ServerIronADX(config-rs-rs30.1)# port dns
ServerIronADX(config-rs-rs30.1)# port ftp
ServerIronADX(config-rs-rs30.1)# port http
ServerIronADX(config-rs-rs30.1)# port http url "HEAD /"
ServerIronADX(config-rs-rs30.1)# port mms
ServerIronADX(config-rs-rs30.1)# port ssl
ServerIronADX(config-rs-rs30.1)# exit
ServerIronADX(config)# server real rs31.1 100.1.1.131
ServerIronADX(config-rs-rs31.1)# port dns
ServerIronADX(config-rs-rs31.1)# port ftp
ServerIronADX(config-rs-rs31.1)# port http
ServerIronADX(config-rs-rs31.1)# port http url "HEAD /"
ServerIronADX(config-rs-rs31.1)# port mms
ServerIronADX(config-rs-rs31.1)# port ssl
ServerIronADX(config-rs-rs31.1)# exit
ServerIronADX(config)# server virtual-name-or-ip www 10.2.24.100
ServerIronADX(config-vs-www)# sym-priority 254
ServerIronADX(config-vs-www)# sym-active
ServerIronADX(config-vs-www)# predictor round-robin
ServerIronADX(config-vs-www)# port http
```

```
ServerIronADX(config-vs-www)# bind http rs31.1 http rs30.1 http rs29.1 http rs30
http
ServerIronADX(config-vs-www)# bind http rs31 http rs29 http
ServerIronADX(config-vs-www)# exit
ServerIronADX(config)# server virtual-name-or-ip ftp 10.2.24.102
ServerIronADX(config-vs-ftp)# sym-priority 254
ServerIronADX(config-vs-ftp)# sym-active
ServerIronADX(config-vs-ftp)# port ftp
ServerIronADX(config-vs-ftp)# bind ftp rs31.1 ftp rs30.1 ftp rs29.1 ftp rs29 ftp
ServerIronADX(config-vs-ftp)# bind ftp rs30 ftp rs31 ftp
ServerIronADX(config-vs-ftp)# exit
ServerIronADX(config)# server virtual-name-or-ip mms 10.2.24.103
ServerIronADX(config-vs-mms)# sym-priority 254
ServerIronADX(config-vs-mms)# sym-active
ServerIronADX(config-vs-mms)# port mms
ServerIronADX(config-vs-mms)# bind mms rs31.1 mms rs30.1 mms rs29.1 mms rs29 mms
ServerIronADX(config-vs-mms)# bind mms rs30 mms rs31 mms
ServerIronADX(config-vs-mms)# exit
ServerIronADX(config)# server virtual-name-or-ip dns 10.2.24.105
ServerIronADX(config-vs-dns)# sym-priority 254
ServerIronADX(config-vs-dns)# sym-active
ServerIronADX(config-vs-dns)# port dns
ServerIronADX(config-vs-dns)# bind dns rs31.1 dns rs30.1 dns rs29.1 dns rs29 dns
ServerIronADX(config-vs-dns)# bind dns rs30 dns rs31 dns
ServerIronADX(config-vs-dns)# exit
ServerIronADX(config)# server virtual-name-or-ip ssl 10.2.24.101
ServerIronADX(config-vs-ssl)# sym-priority 254
ServerIronADX(config-vs-ssl)# sym-active
ServerIronADX(config-vs-ssl)# port ssl sticky
ServerIronADX(config-vs-ssl)# bind ssl rs31.1 ssl rs30.1 ssl rs29.1 ssl rs31 ssl
ServerIronADX(config-vs-ssl)# bind ssl rs30 ssl rs29 ssl
ServerIronADX(config-vs-ssl)# exit
ServerIronADX(config)# server cache-name cache1 100.1.1.99
ServerIronADX(config-cs-cache1)# port ssl
ServerIronADX(config-cs-cache1)# port mms
ServerIronADX(config-cs-cache1)# port http
ServerIronADX(config-cs-cache1)# port http url "HEAD /"
ServerIronADX(config-cs-cache1)# port ftp
ServerIronADX(config-cs-cache1)# exit
ServerIronADX(config)# server cache-name cache2 100.1.1.98
ServerIronADX(config-cs-cache2)# port ssl
ServerIronADX(config-cs-cache2)# port mms
ServerIronADX(config-cs-cache2)# port http
ServerIronADX(config-cs-cache2)# port http url "HEAD /"
ServerIronADX(config-cs-cache2)# port ftp
ServerIronADX(config-cs-cache2)# exit
ServerIronADX(config)# server cache-group 1
ServerIronADX(config-tc-1)# prefer-router-cnt 0
ServerIronADX(config-tc-1)# cache-name cache1
ServerIronADX(config-tc-1)# cache-name cache2
ServerIronADX(config-tc-1)# no http-cache-control
ServerIronADX(config-tc-1)# exit
```

Commands for router NI2

The following commands configure router NI2 in Figure 29.

NOTE
This example shows commands that are valid on the ServerIron ADX device only when it is running the Layer 3 router image.

```
NetIron(config)# vlan 1 name DEFAULT-VLAN by port
NetIron(config-vlan-1)# router-interface ve 1
NetIron(config-vlan-1)# exit
NetIron(config)# vlan 2 by port
NetIron(config-vlan-2)# untagged ethe 1 to 2
NetIron(config-vlan-2)# router-interface ve 2
NetIron(config-vlan-2)# exit
NetIron(config)# vlan 23 by port
NetIron(config-vlan-23)# untagged ethe 23
NetIron(config-vlan-23)# exit
NetIron(config)# interface ve 1
NetIron(config-ve-1)# ip address 172.1.1.4 255.255.255.0
NetIron(config-ve-1)# ip ospf area 0
NetIron(config-ve-1)# ip vrrp-extended vrid 3
NetIron(config-ve-1-vrid-3)# backup
NetIron(config-ve-1-vrid-3)# ip-address 172.1.1.1
NetIron(config-ve-1-vrid-3)# track-port e 1
NetIron(config-ve-1-vrid-3)# track-port e 2
NetIron(config-ve-1-vrid-3)# track-port e 8
NetIron(config-ve-1-vrid-3)# enable
NetIron(config-ve-1)# ip vrrp-extended vrid 4
NetIron(config-ve-1-vrid-4)# backup
NetIron(config-ve-1-vrid-4)# ip-address 172.1.1.2
NetIron(config-ve-1-vrid-4)# track-port e 1
NetIron(config-ve-1-vrid-4)# track-port e 2
NetIron(config-ve-1-vrid-4)# track-port e 8
NetIron(config-ve-1-vrid-4)# enable
NetIron(config-ve-1)# exit
NetIron(config)# interface ve 2
NetIron(config-ve-2)# ip address 10.2.24.2 255.255.255.0
NetIron(config-ve-2)# ip ospf area 0
NetIron(config-ve-2)# exit
NetIron(config)# interface ethernet 23
NetIron(config-if-23)# ip address 173.1.1.1 255.255.255.0
NetIron(config-if-23)# exit
NetIron(config)# ip route 0.0.0.0 0.0.0.0 10.2.24.251
NetIron(config)# ip router-id 10.2.24.2
NetIron(config)# router vrrp-extended
NetIron(config)# route-only
NetIron(config)# router ospf
NetIron(config-ospf-router)# area 0
NetIron(config-ospf-router)# redistribution connected
NetIron(config-ospf-router)# redistribution static
NetIron(config-ospf-router)# exit
```

Commands for ServerIron ADX 253

The following commands configure ServerIron ADX 253 in Figure 29.

NOTE
This example shows commands that are valid on the ServerIron ADX device only when it is running the Layer 3 router image.

```
ServerIronADX(config)# vlan 1 name DEFAULT-VLAN by port
ServerIronADX(config-vlan-1)# router-interface ve 1
ServerIronADX(config-vlan-1)# exit
ServerIronADX(config)# vlan 2 by port
ServerIronADX(config-vlan-2)# untagged ethe 3/1 ethe 3/3 ethe 4/1 ethe 4/3
ServerIronADX(config-vlan-2)# router-interface ve 2
ServerIronADX(config-vlan-2)# exit
ServerIronADX(config)# vlan 16 by port
ServerIronADX(config-vlan-16)# untagged ethe 3/16
ServerIronADX(config-vlan-16)# static-mac-address 00e0.52ee.c700 ethernet 3/16
ServerIronADX(config-vlan-16)# exit
ServerIronADX(config)# vlan 999 by port
ServerIronADX(config-vlan-999)# untagged ethe 3/24
ServerIronADX(config-vlan-999)# exit
ServerIronADX(config)# interface ve 1
ServerIronADX(config-ve-1)# ip address 100.1.1.251 255.255.255.0
ServerIronADX(config-ve-1)# ip vrrp-extended vrid 5
ServerIronADX(config-ve-1-vrid-5)# backup
ServerIronADX(config-ve-1-vrid-5)# ip-address 100.1.1.254
ServerIronADX(config-ve-1-vrid-5)# track-port e 3/1
ServerIronADX(config-ve-1-vrid-5)# track-port e 3/2
ServerIronADX(config-ve-1-vrid-5)# track-port e 3/3
ServerIronADX(config-ve-1-vrid-5)# enable
ServerIronADX(config-ve-1)# ip vrrp-extended vrid 6
ServerIronADX(config-ve-1-vrid-6)# backup
ServerIronADX(config-ve-1-vrid-6)# ip-address 100.1.1.253
ServerIronADX(config-ve-1-vrid-6)# track-port e 3/1
ServerIronADX(config-ve-1-vrid-6)# track-port e 3/2
ServerIronADX(config-ve-1-vrid-6)# track-port e 3/3
ServerIronADX(config-ve-1-vrid-6)# enable
ServerIronADX(config-ve-1-vrid-6)# exit
ServerIronADX(config)# interface ve 2
ServerIronADX(config-ve-2)# ip address 10.2.24.251 255.255.255.0
ServerIronADX(config-ve-2)# ip ospf area 0
ServerIronADX(config-ve-2)# exit
ServerIronADX(config)# ip l4-policy 1 cache tcp 0 global
ServerIronADX(config)# ip l4-policy 2 cache udp 0 global
ServerIronADX(config)# ip l4-policy 3 cache tcp http global
ServerIronADX(config)# ip l4-policy 4 cache tcp ftp global
ServerIronADX(config)# ip l4-policy 5 cache tcp mms global
ServerIronADX(config)# ip router-id 10.2.24.251
ServerIronADX(config)# router vrrp-extended
ServerIronADX(config)# server active-active-port ethe 3/13 vlan-id 13
ServerIronADX(config)# router ospf
ServerIronADX(config-ospf-router)# area 0
ServerIronADX(config-ospf-router)# redistribution connected
ServerIronADX(config-ospf-router)# exit
```

The following commands enable session synchronization on the ports where the active-active SLB feature is used. This is required both to ensure continued service following a failover and to enable each ServerIron ADX to send server replies back to the clients, regardless of which ServerIron ADX load balanced the request.

```
ServerIronADX(config)# server port 80
ServerIronADX(config-port-80)# session-sync
ServerIronADX(config-port-80)# tcp
ServerIronADX(config-port-80)# exit
ServerIronADX(config)# server port 21
ServerIronADX(config-port-21)# session-sync
ServerIronADX(config-port-21)# exit
```

```
ServerIronADX(config)# server port 1755
ServerIronADX(config-port-1755)# session-sync
ServerIronADX(config-port-1755)# tcp
ServerIronADX(config-port-1755)# udp
ServerIronADX(config-port-1755)# exit
ServerIronADX(config)# server port 53
ServerIronADX(config-port-53)# session-sync
ServerIronADX(config-port-53)# exit
ServerIronADX(config)# server port 443
ServerIronADX(config-port-443)# session-sync
ServerIronADX(config-port-443)# tcp
ServerIronADX(config-port-443)# exit
ServerIronADX(config)# server router-ports ethernet 3/1
ServerIronADX(config)# server router-ports ethernet 3/3
ServerIronADX(config)# server real rs29 100.1.1.29
ServerIronADX(config-rs-rs29)# port ssl
ServerIronADX(config-rs-rs29)# port mms
ServerIronADX(config-rs-rs29)# port http
ServerIronADX(config-rs-rs29)# port http url "HEAD /"
ServerIronADX(config-rs-rs29)# port ftp
ServerIronADX(config-rs-rs29)# port dns
ServerIronADX(config-rs-rs29)# exit
ServerIronADX(config)# server real rs30 100.1.1.30
ServerIronADX(config-rs-rs30)# port ssl
ServerIronADX(config-rs-rs30)# port mms
ServerIronADX(config-rs-rs30)# port http
ServerIronADX(config-rs-rs30)# port http url "HEAD /"
ServerIronADX(config-rs-rs30)# port ftp
ServerIronADX(config-rs-rs30)# port dns
ServerIronADX(config-rs-rs30)# exit
ServerIronADX(config)# server real rs31 100.1.1.31
ServerIronADX(config-rs-rs31)# port ssl
ServerIronADX(config-rs-rs31)# port mms
ServerIronADX(config-rs-rs31)# port http
ServerIronADX(config-rs-rs31)# port http url "HEAD /"
ServerIronADX(config-rs-rs31)# port ftp
ServerIronADX(config-rs-rs31)# port dns
ServerIronADX(config-rs-rs31)# exit
ServerIronADX(config)# server real rs29.1 100.1.1.129
ServerIronADX(config-rs-rs29.1)# port dns
ServerIronADX(config-rs-rs29.1)# port ftp
ServerIronADX(config-rs-rs29.1)# port http
ServerIronADX(config-rs-rs29.1)# port http url "HEAD /"
ServerIronADX(config-rs-rs29.1)# port mms
ServerIronADX(config-rs-rs29.1)# port ssl
ServerIronADX(config-rs-rs29.1)# exit
ServerIronADX(config)# server real rs30.1 100.1.1.130
ServerIronADX(config-rs-rs30.1)# port dns
ServerIronADX(config-rs-rs30.1)# port ftp
ServerIronADX(config-rs-rs30.1)# port http
ServerIronADX(config-rs-rs30.1)# port http url "HEAD /"
ServerIronADX(config-rs-rs30.1)# port mms
ServerIronADX(config-rs-rs30.1)# port ssl
ServerIronADX(config-rs-rs30.1)# exit
ServerIronADX(config)# server real rs31.1 100.1.1.131
ServerIronADX(config-rs-rs31.1)# port dns
ServerIronADX(config-rs-rs31.1)# port ftp
ServerIronADX(config-rs-rs31.1)# port http
```

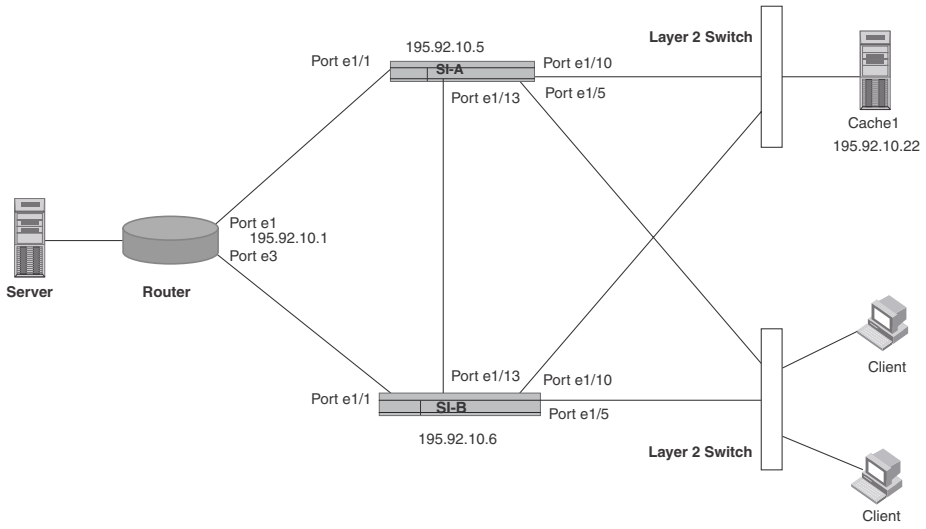
```
ServerIronADX(config-rs-rs31.1)# port http url "HEAD /"
ServerIronADX(config-rs-rs31.1)# port mms
ServerIronADX(config-rs-rs31.1)# port ssl
ServerIronADX(config-rs-rs31.1)# exit
ServerIronADX(config)# server virtual-name-or-ip www 10.2.24.100
ServerIronADX(config-vs-www)# sym-priority 254
ServerIronADX(config-vs-www)# sym-active
ServerIronADX(config-vs-www)# predictor round-robin
ServerIronADX(config-vs-www)# port http
ServerIronADX(config-vs-www)# bind http rs31.1 http rs30.1 http rs29.1 http rs30
http
ServerIronADX(config-vs-www)# bind http rs31 http rs29 http
ServerIronADX(config-vs-www)# exit
ServerIronADX(config)# server virtual-name-or-ip ftp 10.2.24.102
ServerIronADX(config-vs-ftp)# sym-priority 254
ServerIronADX(config-vs-ftp)# sym-active
ServerIronADX(config-vs-ftp)# port ftp
ServerIronADX(config-vs-ftp)# bind ftp rs31.1 ftp rs30.1 ftp rs29.1 ftp rs29 ftp
ServerIronADX(config-vs-ftp)# bind ftp rs30 ftp rs31 ftp
ServerIronADX(config-vs-ftp)# exit
ServerIronADX(config)# server virtual-name-or-ip mms 10.2.24.103
ServerIronADX(config-vs-mms)# sym-priority 254
ServerIronADX(config-vs-mms)# sym-active
ServerIronADX(config-vs-mms)# port mms
ServerIronADX(config-vs-mms)# bind mms rs31.1 mms rs30.1 mms rs29.1 mms rs29 mms
ServerIronADX(config-vs-mms)# bind mms rs30 mms rs31 mms
ServerIronADX(config-vs-mms)# exit
ServerIronADX(config)# server virtual-name-or-ip dns 10.2.24.105
ServerIronADX(config-vs-dns)# sym-priority 254
ServerIronADX(config-vs-dns)# sym-active
ServerIronADX(config-vs-dns)# port dns
ServerIronADX(config-vs-dns)# bind dns rs31.1 dns rs30.1 dns rs29.1 dns rs29 dns
ServerIronADX(config-vs-dns)# bind dns rs30 dns rs31 dns
ServerIronADX(config-vs-dns)# exit
ServerIronADX(config)# server virtual-name-or-ip ssl 10.2.24.101
ServerIronADX(config-vs-ssl)# sym-priority 254
ServerIronADX(config-vs-ssl)# sym-active
ServerIronADX(config-vs-ssl)# port ssl sticky
ServerIronADX(config-vs-ssl)# bind ssl rs31.1 ssl rs30.1 ssl rs29.1 ssl rs31 ssl
ServerIronADX(config-vs-ssl)# bind ssl rs30 ssl rs29 ssl
ServerIronADX(config-vs-ssl)# exit
ServerIronADX(config)# server cache-name cache1 100.1.1.99
ServerIronADX(config-cs-cache1)# port ssl
ServerIronADX(config-cs-cache1)# port mms
ServerIronADX(config-cs-cache1)# port http
ServerIronADX(config-cs-cache1)# port http url "HEAD /"
ServerIronADX(config-cs-cache1)# port ftp
ServerIronADX(config-cs-cache1)# exit
ServerIronADX(config)# server cache-name cache2 100.1.1.98
ServerIronADX(config-cs-cache2)# port ssl
ServerIronADX(config-cs-cache2)# port mms
ServerIronADX(config-cs-cache2)# port http
ServerIronADX(config-cs-cache2)# port http url "HEAD /"
ServerIronADX(config-cs-cache2)# port ftp
ServerIronADX(config-cs-cache2)# exit
```

```
ServerIronADX(config)# server cache-group 1
ServerIronADX(config-tc-1)# prefer-router-cnt 0
ServerIronADX(config-tc-1)# cache-name cache1
ServerIronADX(config-tc-1)# cache-name cache2
ServerIronADX(config-tc-1)# no http-cache-control
ServerIronADX(config-tc-1)# exit
```

Active-standby TCS

TCS is supported in an active-standby configuration. Figure 27 illustrates a sample active-standby TCS configuration. In this configuration, one of the ServerIron ADXs serves as the active ServerIron ADX, while the other remains in standby mode. If the active ServerIron ADX fails, the standby ServerIron ADX assumes the duties of the failed ServerIron ADX and becomes the new active ServerIron ADX.

FIGURE 30 Active-standby TCS configuration



Configuring router R1

The following commands configure router R1 in Figure 27.

NOTE
This example shows commands that are valid on the ServerIron ADX device only when it is running the Layer 3 router image.

```
ServerIronADX(config)# vlan 1 name DEFAULT-VLAN by port
ServerIronADX(config-vlan-1)# router-interface ve 1
ServerIronADX(config-vlan-1)# exit
ServerIronADX(config)# vlan 200 by port
ServerIronADX(config-vlan-1)# untagged ethe 9 to 16 ethe 26
ServerIronADX(config-vlan-1)# router-interface ve 2
ServerIronADX(config-vlan-1)# exit
ServerIronADX(config)# vlan 3 by port
ServerIronADX(config-vlan-1)# untagged ethe 17 to 18
ServerIronADX(config-vlan-1)# router-interface ve 3
ServerIronADX(config-vlan-1)# exit
```

```
ServerIronADX(config)# vlan 24 by port
ServerIronADX(config-vlan-1)# untagged ethe 24
ServerIronADX(config-vlan-1)# router-interface ve 4
ServerIronADX(config-vlan-1)# exit
ServerIronADX(config)# ip route 195.90.5.0 255.255.255.0 195.92.10.7
ServerIronADX(config)# interface ve 1
ServerIronADX(config-ve-1)# ip address 195.92.10.1 255.255.255.0
ServerIronADX(config-ve-1)# exit
ServerIronADX(config)# interface ve 2
ServerIronADX(config-ve-2)# ip address 195.92.20.1 255.255.255.0
ServerIronADX(config-ve-2)# exit
ServerIronADX(config)# interface ve 3
ServerIronADX(config-ve-3)# ip address 172.32.1.1 255.255.255.0
ServerIronADX(config-ve-3)# exit
```

Configuring ServerIron ADX A

The following commands configure ServerIron ADX A in Figure 27.

NOTE

This example shows commands that are valid on the ServerIron ADX device only when it is running the Layer 3 router image.

```
ServerIronADX(config)# vlan 1 name DEFAULT-VLAN by port
ServerIronADX(config-vlan-1)# no spanning-tree
ServerIronADX(config-vlan-1)# exit
ServerIronADX(config)# vlan 13 by port
ServerIronADX(config-vlan-13)# untagged ethe 1/13
ServerIronADX(config-vlan-13)# no spanning-tree
ServerIronADX(config-vlan-13)# exit
ServerIronADX(config)# ip address 195.92.10.5 255.255.255.0
ServerIronADX(config)# ip default-gateway 195.92.10.1
ServerIronADX(config)# ip policy 1 cache tcp 0 global
ServerIronADX(config)# ip policy 2 cache udp 0 global
ServerIronADX(config)# ip policy 3 cache tcp http global
ServerIronADX(config)# ip policy 4 cache tcp rtsp global
ServerIronADX(config)# ip policy 5 cache tcp mms global
ServerIronADX(config)# ip policy 6 cache tcp ftp global
ServerIronADX(config)# server backup ethe 1/13 00e0.52c2.8b00
ServerIronADX(config)# server tcp-age 20
ServerIronADX(config)# server udp-age 20
ServerIronADX(config)# no server l4-check
ServerIronADX(config)# server cache-name cache1 195.92.10.22
ServerIronADX(config-rs-cache1)# port http
ServerIronADX(config-rs-cache1)# port http url "HEAD /"
ServerIronADX(config-rs-cache1)# port mms
ServerIronADX(config-rs-cache1)# port rtsp
ServerIronADX(config-rs-cache1)# port ftp
ServerIronADX(config-rs-cache1)# exit
ServerIronADX(config)# server cache-group 1
ServerIronADX(config-tc-1)# cache-name cache1
ServerIronADX(config-tc-1)# exit
```

Configuring ServerIron ADX B

The following commands configure ServerIron ADX B in Figure 27.

NOTE
This example shows commands that are valid on the ServerIron ADX device only when it is running the Layer 3 router image.

```
ServerIronADX(config)# vlan 1 name DEFAULT-VLAN by port
ServerIronADX(config-vlan-1)# no spanning-tree
ServerIronADX(config-vlan-1)# exit
ServerIronADX(config)# vlan 13 by port
ServerIronADX(config-vlan-13)# untagged ethe 1/13
ServerIronADX(config-vlan-13)# no spanning-tree
ServerIronADX(config-vlan-13)# exit
ServerIronADX(config)# ip address 195.92.10.6 255.255.255.0
ServerIronADX(config)# ip default-gateway 195.92.10.1
ServerIronADX(config)# ip policy 1 cache tcp 0 global
ServerIronADX(config)# ip policy 2 cache udp 0 global
ServerIronADX(config)# ip policy 3 cache tcp http global
ServerIronADX(config)# ip policy 4 cache tcp rtsp global
ServerIronADX(config)# ip policy 5 cache tcp mms global
ServerIronADX(config)# ip policy 6 cache tcp ftp global
ServerIronADX(config)# mirror ethernet 1/5
ServerIronADX(config)# server backup ethe 1/13 00e0.52c2.8b00
ServerIronADX(config)# server tcp-age 20
ServerIronADX(config)# server udp-age 20
ServerIronADX(config)# no server l4-check
ServerIronADX(config)# server cache-name cache1 195.92.10.22
ServerIronADX(config-rs-cache1)# port http
ServerIronADX(config-rs-cache1)# port http url "HEAD /"
ServerIronADX(config-rs-cache1)# port mms
ServerIronADX(config-rs-cache1)# port rtsp
ServerIronADX(config-rs-cache1)# port ftp
ServerIronADX(config-rs-cache1)# exit
ServerIronADX(config)# server cache-group 1
ServerIronADX(config-tc-1)# cache-name cache1
ServerIronADX(config-tc-1)# exit
```

Interoperability issues with cache servers

The defaults for many of the ServerIron ADX's parameters are applicable to most TCS environments. However, depending on the brand of cache server you use, you might need to make minor modifications either to the cache server or to the ServerIron ADX for interoperability.

CacheFlow server version 2.x.x and 3.x.x

These versions send HTTP error code 503 in response to an HTTP keep-alive health check sent by the ServerIron ADX. By default, the ServerIron ADX does not consider this to be a valid health check response. To work around this issue, configure status code 503 to be a valid response to the health check. To do so, enter the **port http status_code 503 503** command at the configuration level for the cache server. This issue has not been observed with CacheFlow version 1.x.x.

NetCache servers

When the ServerIron ADX sends a packet to a NetCache server, by default the server addresses its replies to the packet to the source MAC address of the packet, instead of replying to the MAC address of the server's default gateway. This causes problems, especially in one-arm routing configurations.

To work around this issue, enter the following commands on the NetCache server.

```
priv set advanced
show config.system.fast_ip.enable
set config.system.fast_ip.enable off
```

NetCache C720 cache server

This model of cache server sends HTTP error code 500 in response to an HTTP keep-alive health check sent by the ServerIron ADX. By default, the ServerIron ADX does not consider this to be a valid health check response. To work around this issue, configure status code 500 to be a valid response to the health check. To do so, enter the **port http status_code 500 500** command at the configuration level for the cache server.

CSW with NetCache cache servers

When the ServerIron ADX redirects a TCP SYN to a cache server, the ServerIron ADX uses the ServerIron ADX port's MAC address as the source MAC address and the cache server's MAC address as the destination MAC address. The source and destination IP addresses are not changed. The source IP address is the client's and the destination IP address is the website the client is requesting.

When the ServerIron ADX receives the SYN-ACK from the cache server, the ServerIron ADX expects the destination MAC address in the cache server's reply to belong to the destination IP address (the requested Web site). The ServerIron ADX does not expect the destination MAC address to be the ServerIron ADX port's MAC address.

If you are using a NetCache cache server, you must enable the server to respond appropriately to the ServerIron ADX. To do so, access the following URL on the Netcache server:

- http:// cache_IP_address:3132/cache_config/ui_toggle_basic

On this page, disable IP Fast Output. When this feature is disabled, the cache server performs a route lookup and sends the packet to the correct destination address.