Application Guide for SSL Acceleration Avaya VPN Gateway

9.0
NN46120-100, 04.01
August 2012

Notice

While reasonable efforts have been made to ensure that the information in this document is complete and accurate at the time of printing, Avaya assumes no liability for any errors. Avaya reserves the right to make changes and corrections to the information in this document without the obligation to notify any person or organization of such changes.

Documentation disclaimer

Avaya shall not be responsible for any modifications, additions, or deletions to the original published version of this documentation unless such modifications, additions, or deletions were performed by Avaya. End User agree to indemnify and hold harmless Avaya, Avaya's agents, servants and employees against all claims, lawsuits, demands and judgments arising out of, or in connection with, subsequent modifications, additions or deletions to this documentation, to the extent made by End User.

Link disclaimer

Avaya is not responsible for the contents or reliability of any linked Web sites referenced within this site or documentation(s) provided by Avaya. Avaya is not responsible for the accuracy of any information, statement or content provided on these sites and does not necessarily endorse the products, services, or information described or offered within them. Avaya does not guarantee that these links will work all the time and has no control over the availability of the linked pages.

Warranty

Avaya provides a limited warranty on this product. Refer to your sales agreement to establish the terms of the limited warranty. In addition, Avaya's standard warranty language, as well as information regarding support for this product, while under warranty, is available to Avaya customers and other parties through the Avaya Support Web site: http://www.avaya.com/support. Please note that if you acquired the product from an authorized Avaya reseller outside of the United States and Canada, the warranty is provided to you by said Avaya reseller and not by Avaya.

Licenses

THE SOFTWARE LICENSE TERMS AVAILABLE ON THE AVAYA WEBSITE, HTTP://SUPPORT.AVAYA.COM/LICENSEINFO/ ARE APPLICABLE TO ANYONE WHO DOWNLOADS, USES AND/OR INSTALLS AVAYA SOFTWARE, PURCHASED FROM AVAYA INC., ANY AVAYA AFFILIATE, OR AN AUTHORIZED AVAYA RESELLER (AS APPLICABLE) UNDER A COMMERCIAL AGREEMENT WITH AVAYA OR AN AUTHORIZED AVAYA RESELLER. UNLESS OTHERWISE AGREED TO BY AVAYA IN WRITING, AVAYA DOES NOT EXTEND THIS LICENSE IF THE SOFTWARE WAS OBTAINED FROM ANYONE OTHER THAN AVAYA, AN AVAYA AFFILIATE OR AN AVAYA AUTHORIZED RESELLER, AND AVAYA RESERVES THE RIGHT TO TAKE LEGAL ACTION AGAINST YOU AND ANYONE ELSE USING OR SELLING THE SOFTWARE WITHOUT A LICENSE. BY INSTALLING, DOWNLOADING OR USING THE SOFTWARE, OR AUTHORIZING OTHERS TO DO SO, YOU, ON BEHALF OF YOURSELF AND THE ENTITY FOR WHOM YOU ARE INSTALLING, DOWNLOADING OR USING THE SOFTWARE (HEREINAFTER REFERRED TO INTERCHANGEABLY AS "YOU" AND "END USER"), AGREE TO THESE TERMS AND CONDITIONS AND CREATE A BINDING CONTRACT BETWEEN YOU AND AVAYA INC. OR THE APPLICABLE AVAYA AFFILIATE ("AVAYA").

Copyright

Except where expressly stated otherwise, no use should be made of materials on this site, the Documentation(s) and Product(s) provided by Avaya. All content on this site, the documentation(s) and the product(s) provided by Avaya including the selection, arrangement and design of the content is owned either by Avaya or its licensors and is protected by copyright and other intellectual property laws including the sui generis rights relating to the protection of databases. You may not modify, copy, reproduce, republish, upload, post, transmit or distribute in any way any content, in whole or in part, including any code and software. Unauthorized reproduction, transmission, dissemination, storage, and or use without the express written consent of Avaya can be a criminal, as well as a civil, offense under the applicable law.

Third-party components

Certain software programs or portions thereof included in the Product may contain software distributed under third party agreements ("Third Party Components"), which may contain terms that expand or limit rights to use certain portions of the Product ("Third Party Terms"). Information regarding distributed Linux OS source code (for those Products that have distributed the Linux OS source code), and identifying the copyright holders of the Third Party Components and the Third Party Terms that apply to them is available on the Avaya Support Web site: http://www.avaya.com/support/Copyright/.

Trademarks

The trademarks, logos and service marks ("Marks") displayed in this site, the documentation(s) and product(s) provided by Avaya are the registered or unregistered Marks of Avaya, its affiliates, or other third parties. Users are not permitted to use such Marks without prior written consent from Avaya or such third party which may own the Mark. Nothing contained in this site, the documentation(s) and product(s) should be construed as granting, by implication, estoppel, or otherwise, any license or right in and to the Marks without the express written permission of Avaya or the applicable third party.

Avaya is a registered trademark of Avaya Inc.

All other trademarks are the property of their respective owners.

Downloading documents

For the most current versions of documentation, see the Avaya Support Web site: http://www.avaya.com/support

Contact Avaya Support

Avaya provides a telephone number for you to use to report problems or to ask questions about your product. The support telephone number is 1-800-242-2121 in the United States. For additional support telephone numbers, see the Avaya Web site: http://www.avaya.com/support

Contents

Chapter 1: Preface

This *Application Guide* contains example configurations and usage of the Avaya VPN Gateway (AVG) when using it for SSL acceleration. For instructions on how to deploy the VPN feature, see the *Application Guide for VPN* (available for both CLI and BBI configuration).

Who Should Use This Book

This *Application Guide* is intended for network installers and system administrators engaged in configuring and maintaining a network. It assumes that you are familiar with Ethernet concepts and IP addressing. All IP addresses are examples and should not be used as-is.

Related Documentation

For full documentation on installing, configuring and using the many features of the AVG, see the following manuals:

- *Avaya VPN Gateway 9.0 Users Guide* Describes the initial setup procedure, upgrades, operator user management, certificate management, troubleshooting and other general operations that apply to both SSL Acceleration and VPN.
- *Avaya VPN Gateway 9.0 Command Reference* Describes each command in detail. The commands are listed per menu, according to the order they appear in the Command Line Interface (CLI).
- *comment: Avaya VPN Gateway 9.0 CLI Application Guide for VPN* Provides examples on how to configure VPN deployment through the CLI.
- *Avaya VPN Gateway 9.0 BBI Application Guide for VPN* Provides examples on how to configure VPN deployment through the BBI (Browser-Based Management Interface).
- *Avaya VPN Gateway 9.0 VPN Administrators Guide* VPN management guide intended for end-customers in a Secure Service Partitioning configuration.
- *Avaya VPN Gateway 3050/3070 Hardware Installation Guide* Describes installation of the VPN Gateway 3050 and 3070 hardware models.
- *Avaya Configuration - Secure Portable Office Client* Gives the feature list and provides general information about Secure Portable Office Client.

- *Avaya VPN Gateway 9.0 Troubleshooting Guide* Describes the prerequisites and various tools used to troubleshoot the Avaya VPN Gateway (AVG).
- *Avaya VPN Gateway 9.0 Release Notes* Lists new features available in version 9.0 and provides up-to-date product information.

The preceding manuals are available for download (see Customer service on page 9).

Product Names

The software described in this manual runs on several different hardware models. Whenever the generic terms Avaya VPN Gateway, Avaya Gateway or AVG are used in the documentation, the following hardware models are implied:

- Avaya VPN Gateway 3050 (AVG 3050)
- Avaya VPN Gateway 3070 (AVG 3070)

Similarly, all references to the old product name – iSD-SSL or iSD – in commands or screen outputs should be interpreted as applying to the preceding hardware models.

Note:

Manufacturing of the Avaya SSL Accelerator (formerly Alteon SSL Accelerator) has been discontinued.

How This Book Is Organized

Public Key Infrastructure and SSL on page 13 provides a general overview of the basic concepts behind Secure Sockets Layer (SSL) transactions in general, as well as the SSL transactions involved when the VPN Gateway is used as a web server accelerator.

Basic Applications on page 21 provides basic scenarios and configuration examples for using the VPN Gateway for web server acceleration (without end to end encryption), content intelligent switching for secure sessions, and in redundant active-standby configurations.

Web Server Accelerator with End to End Encryption on page 65 describes how to configure the AVG for end to end encryption, where also the data transmitted between the VPN Gateway and the backend servers is encrypted.

Web Server Accelerator with Multiple Networks in Cluster on page 81 describes how to configure multiple networks within a cluster consisting of two VPN Gateways, used in combination with four Application Switches.

Configuring the AVG to Rewrite Client Requests on page 123 describes how to configure the VPN Gateway to rewrite client requests for the HTTPS service if the client browser does not meet the required cipher strength.

HTTP to HTTPS Redirection on page 127 describes how to configure the VPN Gateway for redirection of http requests to https.

Load Balancing of Backend Servers on page 131 describes how to configure the VPN Gateway to perform basic load balancing of real servers. Additional load balancing options such as metrics, health checks, string matching, and persistent client connections are also discussed.

String-Based Load Balancing and Blocking on page 149 describes how to create match strings used for load balancing or blocking. Configuration examples with detailed step-by-step instructions are provided in the chapter.

Script-Based Health Checks on page 161 describes how to create a customized health check script that can be used when the VPN Gateway is configured to perform load balancing of backend servers.

Stand-Alone Web Server Accelerator on page 177 describes how to set up the VPN Gateway as a stand-alone web server accelerator, without using an Application Switch.

Global Server Load Balancing on page 189 describes how to set up the VPN Gateway for Global server load balancing (GSLB), which allows you to balance server traffic load across multiple physical sites.

Customer service

Visit the Avaya Web site to access the complete range of services and support that Avaya provides. Go to www.avaya.com or go to one of the pages listed in the following sections.

- Getting technical documentation on page 9
- Getting product training on page 9
- Getting help from a distributor or reseller on page 10
- Getting technical support from the Avaya Web site on page 10

Getting technical documentation

To download and print selected technical publications and release notes directly from the Internet, go to www.avaya.com/support.

Getting product training

Ongoing product training is available. For more information or to register, you can access the Web site at http://www.avaya.com/support. From this Web site, you can locate the Training contacts link on the left-hand navigation pane.

Getting help from a distributor or reseller

If you purchased a service contract for your Avaya product from a distributor or authorized reseller, contact the technical support staff for that distributor or reseller for assistance.

Getting technical support from the Avaya Web site

The easiest and most effective way to get technical support for Avaya products is from the Avaya Technical Support Web site at www.avaya.com/support.

Chapter 2: New in this release

The following sections detail what's new in *Avaya VPN Gateway Application Guide for SSL Acceleration* (NN46120-100) Release 8.0. There are no new features or changes to this guide for Release 9.0.

Features

There are no new features for this release.

Other Changes

See the following sections for information about changes that are not feature-related:

- The hardware models were updated in Release 8.0. For more information, see Product Names on page 8.

Chapter 3: Public Key Infrastructure and SSL

This chapter describes some of the fundamentals behind the Avaya VPN Gateway.

Encryption

Encryption and decryption allow two communicating parties to disguise information they send to each other. The sender encrypts, or scrambles, information before sending it. The receiver decrypts, or unscrambles, the information after receiving it. While in transit, the encrypted information is unintelligible to an intruder. Because of the number of possible combinations that can be formed out of the 128 bits, it is extremely difficult for a third party to intercept and decrypt the messages being sent. Table 1: Statistical Time Required to Break Encryption on page 13 shows the time required to break the encryption of a message based on the key length.

Table 1: Statistical Time Required to Break Encryption

Key Length	Number of Possible Keys	Approximate Time to Break the Encryption
40 Bits (Exportable RC2/RC4)	1 Trillion	3.5 Hours
56 Bits (DES)	72 Quadrillion	2 Months
128 Bits (RC2/RC4)	340 Decillion	1.6 Trillion Years

Public Key Encryption

Public key encryption (also called asymmetric encryption) involves a pair of keys—a public key and a private key—associated with an entity that needs to authenticate its identity electronically or to sign or encrypt data. Each public key is published, and the corresponding private key is kept secret. Data encrypted with your public key can be decrypted only with your private key.

Public key cryptography facilitates the following tasks:

- Tamper detection allows the recipient of information to verify that it has not been modified in transit. Any attempt to modify data or substitute a false message will be detected.
- Authentication allows the recipient of information to determine its origin— that is, to confirm the sender's identity.
- Non-repudiation prevents the sender of information from later claiming that the information was never sent.

Digital Signatures

It is possible to use a private key for encryption and a public key for decryption. Although this is not desirable when encrypting sensitive information, it is a crucial part of digitally signing any data. Instead of encrypting the data itself, the signing software creates a one-way hash of the data and then uses a private key to encrypt the hash. The encrypted hash, along with other information, such as the hashing algorithm, is known as a digital signature.

When sending encrypted messages using public key encryption, digital signatures are used to ensure that the message originated with the person sending it, and that the message was not tampered with after the signature was applied.

Digital signatures are also used in digital certificates, where the certificate owner's public key is digitally signed with the private key of a certificate authority. A server certificate, along with other data, is sent to the client during the SSL handshake. The client then uses this information, along with the public key of the certificate authority, in order to authenticate the server.

Certificates

A certificate is an electronic document used to identify an individual, a server, a company, or some other entity and to associate that identity with a public key. A certificate provides recognized proof of a person's identity. Public key cryptography uses certificates to address the problem of impersonation. There are two kinds of certificates:

- Register Certificates: certificates that have been authenticated by an authenticating service, such as a certificate authority.
- Chain Certificates: certificates that have been authenticated by other certificates that have been authenticated by an authenticating service.

Certificate Authorities

A certificate authority (CA) is an entity that validates identities and issues certificates. They are issued by either independent third parties or independent organizations operating their own certificate-issuing server software (such as Netscape Certificate Server). The methods used

to validate an identity vary, depending on the policies of a given CA. In general, before issuing a certificate, the CA must use its published verification procedures for that type of certificate to ensure that an entity requesting a certificate is authentic.

Register Certificates

The certificate issued by the CA binds a particular public key to the name of the entity that the certificate identifies (such as the name of an employee or a server). Certificates help prevent the use of fake public keys for impersonation.

In addition to a public key, a certificate always includes the name of the entity it identifies, an expiration date, the name of the CA that issued the certificate, a serial number, and other information. Most importantly, a certificate always includes the digital signature of the issuing CA. The digital signature of the issuing CA allows the certificate to function as a "letter of introduction" for users who know and trust the CA.

Chain Certificates

Chain certificate allows a chain of trust to be created. Each certificate in the chain attests to the identity of the previous certificate. The final certificate will be a certificate that has been authenticated by a trusted CA. For example, client A trusts the CA, and the CA trusts client B, therefore, client A trusts client B.

Secure Sockets Layer (SSL)

The Secure Sockets Layer (SSL) protocol runs above the TCP/IP protocol and below higher-level protocols such as HTTP or IMAP. SSL uses TCP/IP on behalf of the higher-level protocols and, in the process, allows an SSL-enabled server to authenticate itself to an SSL-enabled client. The client then authenticates itself to the server, and both machines establish an encrypted connection. The current standard is TLS (Transport Layer Security) but the name SSL is still kept.

Example of an SSL Transaction

The steps involved in an SSL transaction can be done with a VPN Gateway (or an SSL-enabled server). The steps are summarized as follows:

1. The client sends the following information to the VPN Gateway: SSL version number, cipher settings, randomly generated data, and other information that the server needs to communicate with the SSL client.
2. The VPN Gateway sends the following information to the client: SSL version number, cipher settings, randomly generated data, and other information needed to communicate with the server over SSL. The server also sends its own certificate and, if the client is requesting a server resource that requires client authentication, requests the client's certificate.
3. The client uses some of the information sent by the VPN Gateway to authenticate it. If the VPN Gateway is not authenticated, the user is warned of the problem and informed that an encrypted and authenticated connection cannot be established. If the VPN Gateway is successfully authenticated, the client goes on to Step 4.
4. The client (with the cooperation of the VPN Gateway, depending on the cipher being used) creates the premaster secret for the session, encrypts it with the AVG 's public key (obtained from the AVG 's certificate, sent in Step 2), and sends the encrypted premaster secret to the VPN Gateway.
5. If the VPN Gateway has requested client authentication (an optional step in the handshake), the client also signs another piece of data that is unique to this handshake and is known by both the client and the VPN Gateway. In this case the client sends both the signed data and the client's own certificate to the AVG along with the encrypted premaster secret.
6. If the VPN Gateway has requested client authentication, it attempts to authenticate the client. If the client is not authenticated, the session is terminated. If the client can be successfully authenticated, the VPN Gateway uses its private key to decrypt the premaster secret, then performs a series of steps (which the client also performs, starting from the same premaster secret) to generate the master secret.
7. Both the client and the VPN Gateway use the master secret to generate the session key, which is a symmetric key. It is used to encrypt and decrypt information exchanged during the SSL session and to verify its integrity—that is, to detect any change in the data between the time it was sent and the time it is received over the SSL connection.
8. The client informs the VPN Gateway that future messages from the client will be encrypted with the session key. The client then sends a separate (encrypted) message indicating that the client portion of the handshake is finished.
9. The VPN Gateway informs the client that future messages will be encrypted with the session key. It then sends the client a separate (encrypted) message indicating that the AVG portion of the handshake is finished.
10. The SSL handshake is now complete, and the SSL session begins. The client and the VPN Gateway use the session keys to encrypt and decrypt the data they send to each other and to verify data integrity.

    Figure 1: SSL Handshake Procedure on page 17 depicts an outline of the preceding steps.

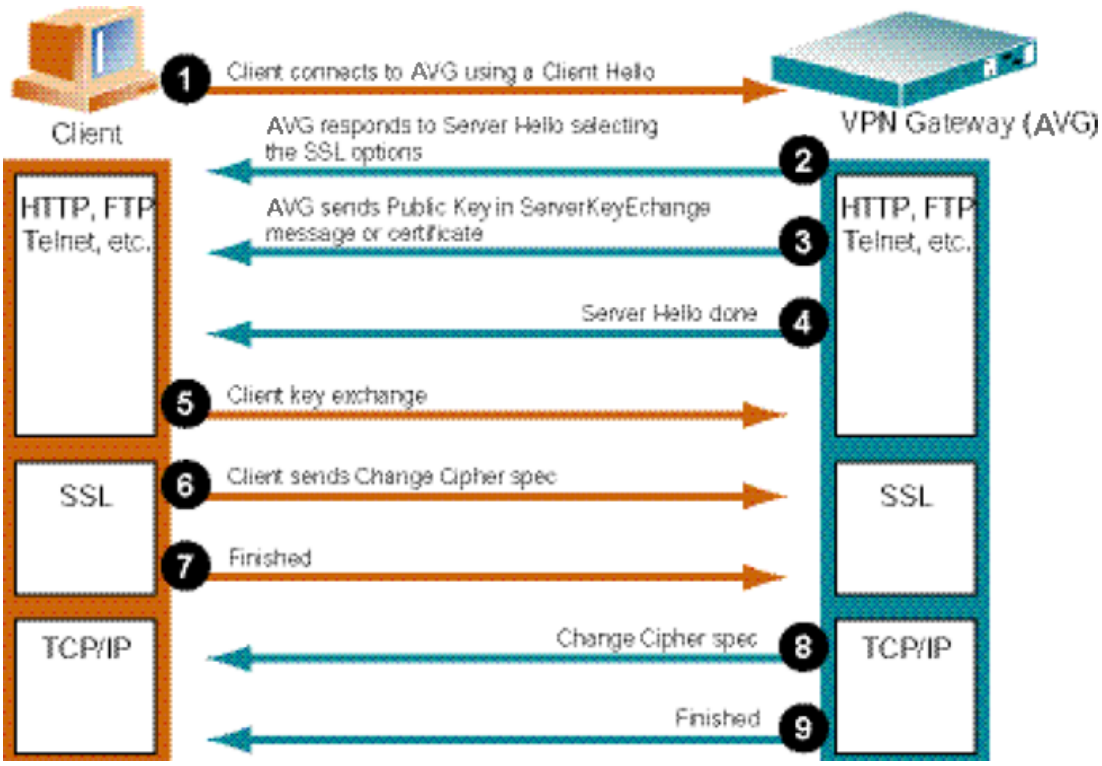

Figure 1: SSL Handshake Procedure

Basic Operation of AVG as Web Server Accelerator

The following diagram and steps describe the basic operation of a VPN Gateway when used as a Web server accelerator together with an Application Switch.

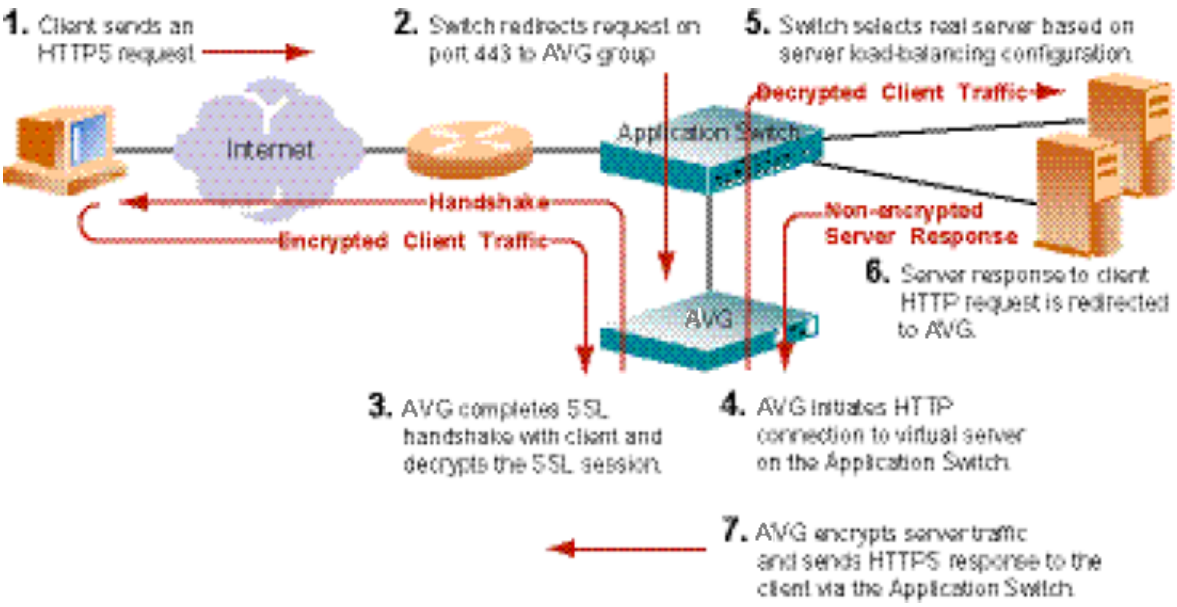

Figure 2: Basic HTTPS Offload Operation

1. Client requests secure information through HTTPS.

   When the client requires secure information, the client web browser sends an HTTPS (Hypertext Transfer Protocol Secure) request on TCP port 443. This request arrives at the Application Switch, to which the server containing the desired information is connected.

2. The Application Switch redirects the request to the AVG group.

   The Application Switch recognizes HTTPS traffic on port 443 and redirects the request to a VPN Gateway. This form of application redirection is described in detail in your *Web OS Application Guide* (provided for buyers of Application Switches).

3. The AVG completes the SSL handshake and decrypts the session.

   The AVG responds to the client's HTTPS request through the Application Switch and starts the SSL session.

4. The AVG initiates HTTP connection to the virtual server.

   The AVG receives the client's encrypted SSL traffic through the Application Switch. The AVG decrypts the secure traffic and forwards it as a regular HTTP request to a virtual server on the Application Switch.

5. The Application Switch selects a real server based on configured load-balancing options.

   Based on criteria such as server health status and configured load-balancing distribution metrics, the Application Switch selects a real server and forwards the client's decrypted HTTP traffic.

6. The server processes the HTTP request and replies to the client.

The server sends the requested non-encrypted HTTP information intended for the client's IP address. The Application Switch redirects this traffic to the VPN Gateway.

7. The AVG encrypts the server traffic and sends the HTTPS response to the client through the Application Switch.

Basic Operation of AVG as Web Server Accelerator with End to End Encryption

The main difference when using end to end encryption, as compared to using the VPN Gateway as a web server accelerator without end to end encryption, is that all traffic on the network segment between the AVG(s) and backend servers is encrypted as well.

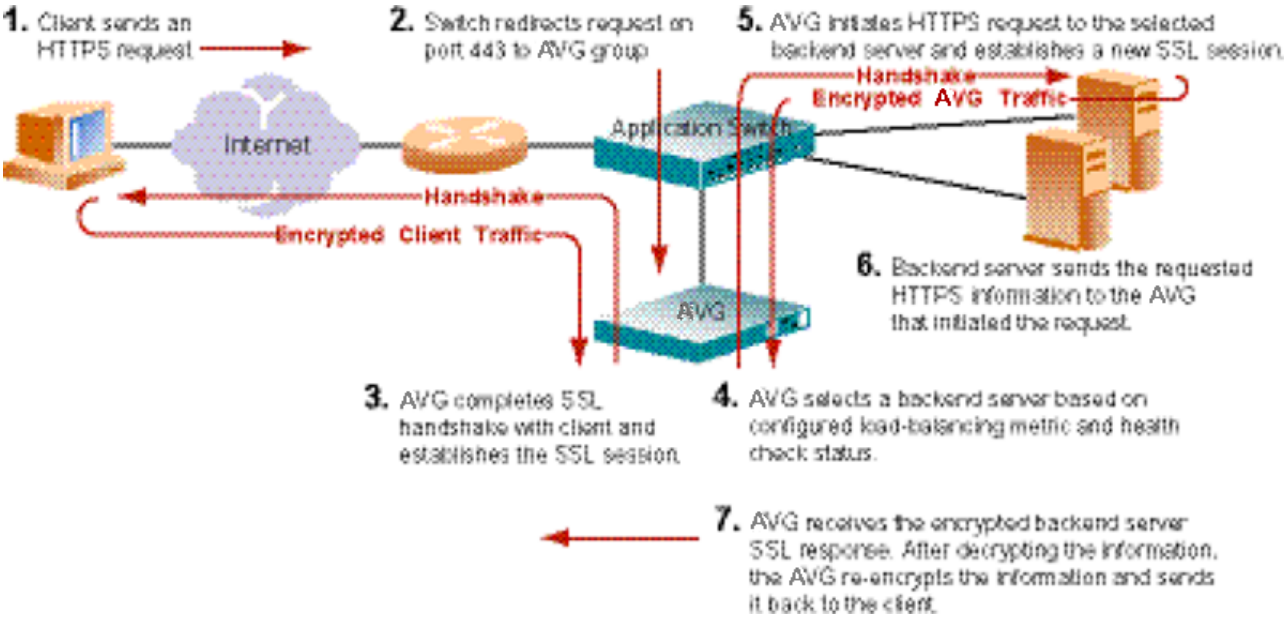

Figure 3: HTTPS Offload Operation With End to End Encryption

1. A client requests secure information through HTTPS.

   When the client requires secure information, the client Web browser sends a Hypertext Transfer Protocol Secure (HTTPS) request on TCP port 443. This request arrives at the Application Switch, to which the server containing the desired information is connected.

2. The Application Switch redirects the request to the AVG group.

   The Application Switch recognizes HTTPS traffic on port 443 and redirects the request to a VPN Gateway. This form of application redirection is described in detail in your *Web OS Application Guide* (provided for buyers of Application Switches).

3. The AVG initiates the SSL handshake and establishes the SSL session.

The AVG responds to the client's HTTPS request through the Application Switch and establishes an SSL session with the client. In order to perform a load-balancing decision, the SSL session is temporarily decrypted.

4. The AVG selects a backend server based on the configured load-balancing options and health check status.

   Based on criteria such as backend server health check status and configured load-balancing distribution metrics, the AVG selects a backend server.

5. The AVG establishes an SSL connection to the selected backend server.

   After having re-encrypted the client's request, the AVG initiates an HTTPS request over an SSL connection to the selected backend server on behalf of the client. On subsequent requests, the AVG will always reuse an existing SSL session in order to decrease the overhead involved in performing a full SSL handshake.

6. The backend server processes the HTTPS request and replies to the AVG.

   The backend server sends the requested encrypted HTTPS information to the AVG that initiated the request.

7. The AVG receives the backend server traffic and sends the HTTPS response to the client through the Application Switch.

   The AVG receives the backend server's encrypted SSL traffic through the Application Switch. After decrypting the response from the backend server, the AVG re-encrypts the information using the initial encryption algorithm, and sends it back to the client.

Chapter 4: Basic Applications

This chapter describes some basic network applications that make use of the VPN Gateway :

- Web Server Accelerator, using an Application Switch configured with redirect filters, and two or more load-balanced VPN Gateways for SSL offload configured to run in transparent proxy mode, on Web Server Accelerator on page 21.
- Web Server Accelerator, using an Application Switch configured to use the Return To Sender feature, and two or more load-balanced VPN Gateways for SSL offload configured to run in transparent proxy mode, on Web Server Accelerator, using Return to Sender (RTS) on page 34.
- Web Server Accelerator, using two or more load-balanced VPN Gateways for SSL offload, and where the AVGs are configured to run in non-transparent proxy mode, on Web Server Accelerator, with AVGs in Non-Transparent Proxy Mode on page 35.
- Content-Intelligent Switching for Secure Sessions, using single or multiple VPN Gateways for SSL offload with cookie processing, on Content-Intelligent Switching for Secure Sessions on page 38.
- Redundant Active-Standby Configuration, using multiple VPN Gateways for high-availability scenarios, on Redundant Active-Standby Configuration on page 39.
- Mail Server Accelerator, using two or more load-balanced VPN Gateways for SSL offload, on Mail Server Accelerator on page 49.
- The sample configurations discussed are merely recommendations and are not required. The first five examples are based on the configuration described in Web Server Accelerator on page 21 . Therefore, it is recommended that you read this chapter in sequence, and modify the base configuration as noted in each subsequent example.

In the Mail Server Accelerator example it is assumed that Web server acceleration has not been set up. However, it is possible to use mail server acceleration in parallel with Web server accelerating using the same group of AVGs.

Web Server Accelerator

Figure 4: Sample Web Server Accelerator Network Using Multiple AVGs on page 22 illustrates the most common network configuration for the AVG . This configuration example consists of two VPN Gateways, an Application Switch and two servers.

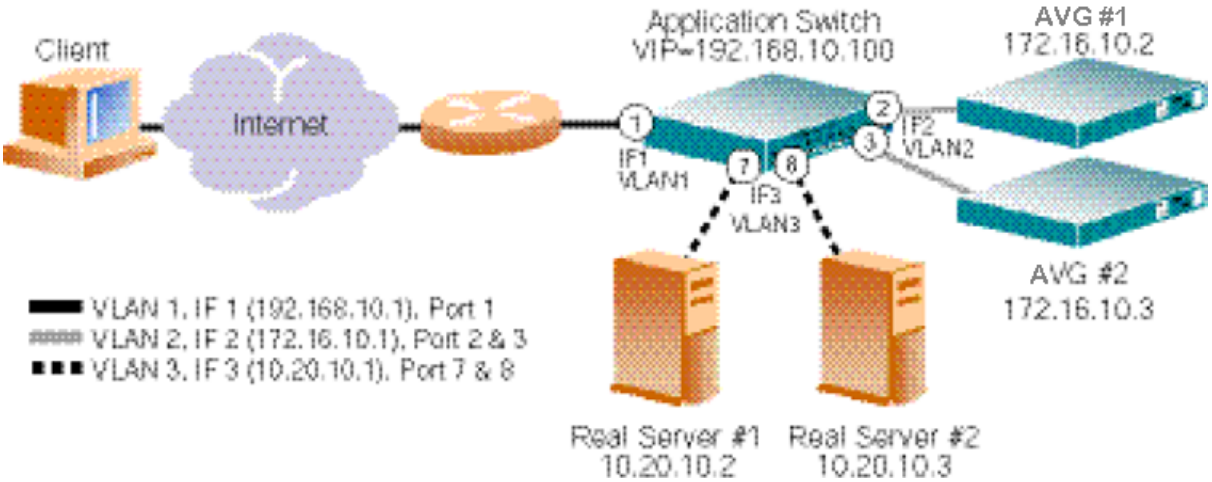

Figure 4: Sample Web Server Accelerator Network Using Multiple AVGs

Initial Setup of the AVGs

This configuration example assumes that you have performed the initial setup of the two VPN Gateways as described in the "Initial Setup" chapter in the *Users Guide*.

After having performed the initial setup, connect to the AVG cluster to add a certificate and configure the AVG parameters. For more information about the concept of AVG clusters and the propagation of configuration changes within a cluster, see the "Initial Setup" chapter in the *Users Guide* .

Add a Server Certificate to the AVG

This step presumes that you have a server certificate, signed by a certificate authority (CA), and a private key. The process for obtaining the required certificate file is covered in the "Certificates and Client Authentication" chapter in the *Users Guide*.

Once you have the appropriate certificate, use the following procedure to add the certificate to the VPN Gateway .

```
# /cfg/cert

Enter certificate number: (1-)1

Creating Certificate 1

>> Certificate 1#cert

Paste the certificate, press Enter to create a new line, and then type
"..."(without the quotation marks) to terminate.
```

The preceding example assumes that the certificate signing request (CSR) was generated from certificate number 1, which implies that the private key that corresponds to the public key in the certificate is already in place.

When prompted for the certificate, follow the instructions on-screen. Use Notepad or any other text editor to display the certificate. Then copy and paste the text of the certificate into the terminal window. For more detailed information about how to add certificates and keys to the AVG, see the "Certificates and Client Authentication" chapter in the *Users Guide* .

Important:

Once you have pasted the entire contents of the certificate file, press ENTER to create a new empty line and then type three periods (...). Press ENTER again to complete the installation of the certificate.

Note:

Under Microsoft Windows, HyperTerminal may be slow to complete the copy-and-paste operation. If your security policy permits enabling Telnet or SSH access to the AVG, use a Telnet or SSH client instead.

Configure the VPN Gateways

The configuration changes will automatically be propagated to all VPN Gateways in the cluster.

Create and Configure a Virtual SSL Server

1. Create a virtual SSL server.

   This step creates a new virtual SSL server on the AVG. Each virtual SSL server listens to a specific TCP port and is connected to a Virtual Server IP address on the Application Switch.

   ```
   # /cfg/ssl/server

   Enter virtual server number: (1-)1

   Creating new server 1
   >> Server 1#
   ```

2. Define a name for virtual SSL server 1.

   This step lets you specify a name, by which you can identify SSL server 1. To view the numbers and related names of all configured SSL servers, use the `/info/servers` command. The name you specify is mainly intended for your own

reference, and is not critical for the configuration itself. As the following example suggests, the name can indicate the service for which the SSL server is created.

```
>> Server 1#name

Current value:""

Enter new SSL server name:HTTPS
```

3. Set listen TCP port for SSL server 1.

   Each time you create a new SSL server, the listen port is automatically set to 443. Since you are setting up the AVG for HTTPS offload purposes in this example, it is not really necessary to configure the SSL server to listen port to 443. However, for using the AVG for any protocol other than HTTPS, a new virtual SSL server must be configured to listen to the TCP port of the intended service.

```
>> Server 1#port

>> Current value: 443 (https)
>> Enter listen port number:443
```

4. Connect the SSL server to the desired Virtual Server IP address on the Application Switch.

   This step connects SSL Server 1 to the IP address of the desired virtual server on the Application Switch.

```
>> Server 1#vips

Current value:""

Enter server ips (comma separated):192.168.10.100
```

5. Set the Real Server IP address to which SSL Server 1 should connect when initiating requests.

```
>> Server 1#rip

Current value: 0.0.0.0
Enter IP address to connect to:0.0.0.0
```

   Preserve the current value of the Real Server IP address, which should be 0.0.0.0. At first glance this configuration may perhaps seem odd. However, by specifying 0.0.0.0 as the Real Server IP address, the SSL server is instructed to use the destination IP address (in the received packets) when initiating requests sent to the virtual server. Since the destination IP address in the received packets corresponds

to the virtual server IP address, the requests will always reach the correct address.

6. Set the server port to which SSL server 1 should connect when initiating requests.

   This step sets the TCP port, to which SSL Server 1 connects when initiating requests.

```
>> Server 1#rport

Current value: 0 [81]
Enter port to connect to:81
```

Note:

If you have not configured any port mappings for the real Web servers on the Application Switch (like in this example), the real Web servers must also be set to listen for AVG traffic on port 81.

7. Specify the certificate to be used by SSL Server 1.

   You are prompted to type the index number of an existing certificate. To view all certificates currently added to the AVG by index number and name, use the `/info/certs` command. For more information on how to add a certificate to the AVG, see the "Certificates and Client Authentication" chapter in the *Users Guide* .

```
>> Server 1#ssl

>> SSL Settings for server 1#cert

Current value: <not set>
Enter certificate number: (1-)1
```

Note:

If the certificate you specify is a chained certificate, you need to first add the CA certificates up to and including the root CA certificate, and then specify the CA certificate chain of the server certificate. For more information on how to construct the server certificate chain, see the **`cachain`** command under "SSL Server SSL Configuration" in the *Command Reference* .

8. Apply the changes.

```
>> SSL Settings for Server 1#apply

Changes applied successfully.
```

Configure the Application Switch

Create the Necessary VLANs

In this configuration, there will be three VLANs: VLAN 1 for the Application Switch that connects to the Internet, VLAN 2 for the AVG units, and VLAN 3 for the real servers. Since VLAN 1 is the default, only VLAN 2 and VLAN 3 require additional configuration. Note that you will perform all of the following changes to the sample configuration on the Application Switch.

1. Configure VLAN 2 to include Application Switch ports leading to the AVG units.

```
# /cfg/vlan 2

>> VLAN 2#add 2

Port 2 is an UNTAGGED port and its current PVID is 1.
Confirm changing PVID from 1 to 2 [y/n]:y

>> VLAN 2#add 3

Port 3 is an UNTAGGED port and its current PVID is 1.
Confirm changing PVID from 1 to 2 [y/n]:y

>> VLAN 2#ena
```

2. Configure VLAN 3 to include Application Switch ports leading to the real servers.

```
# /cfg/vlan 3

>> VLAN 3#add 7

Port 7 is an UNTAGGED port and its current PVID is 1.
Confirm changing PVID from 1 to 2 [y/n]:y

>> VLAN 3#add 8

Port 8 is an UNTAGGED port and its current PVID is 1.
Confirm changing PVID from 1 to 2 [y/n]:y

>> VLAN 3#ena
```

3. Disable Spanning Tree Protocol (STP) for the AVG ports 2 and 3 and real server ports 7 and 8.

```
# /cfg/stp 1/port 2

>> Spanning Tree Port 2#off
```

```
>> Spanning Tree Port 2#../port 3

>> Spanning Tree Port 3#off

>> Spanning Tree Port 3#../port 7

>> Spanning Tree Port 7#off

>> Spanning Tree Port 7#../port 8

>> Spanning Tree Port 8#off
```

Configure One IP Interface for Each VLAN

Note:

If you prefer, you can reverse the order of the first two commands (addr and mask) in the following example. By entering the mask first, the Application Switch will automatically calculate the correct broadcast address for you. The calculated broadcast address is displayed immediately after you provide the IP address of the interface, and will be applied together with the other settings when you execute the **`apply`** command.

1. Configure an IP interface for client traffic on the Application Switch with VLAN 1.

```
# /cfg/ip/if 1

>> IP Interface 1#addr 192.168.10.1

>> IP Interface 1#mask 255.255.255.0

>> IP Interface 1#broad 192.168.10.255

>> IP Interface 1#vlan 1

>> IP Interface 1#ena
```

2. Configure an IP interface for AVG traffic with VLAN 2.

```
# /cfg/ip/if 2

>> IP Interface 2#addr 172.16.10.1

>> IP Interface 2#mask 255.255.0.0

>> IP Interface 2#broad 172.16.255.255

>> IP Interface 2#vlan 2

>> IP Interface 2#ena
```

3. Configure an IP interface for the real server traffic with VLAN 3.

```
# /cfg/ip/if 3

>> IP Interface 3#addr 10.20.10.1

>> IP Interface 3#mask 255.255.255.0

>> IP Interface 3#broad 10.20.10.255

>> IP Interface 3#vlan 3

>> IP Interface 3#ena
```

4. Apply the changes.

```
# apply
```

Note:

Make sure the VPN Gateways are configured to use the IP address of IP interface 2 on VLAN 2 as their default gateway. For more information about gateway configuration, see the **`gateway`** command under "System Configuration" in the *Command Reference* . Likewise, the Web servers must be configured to use the IP address of IP interface 3 on VLAN 3 as their default gateway.

Configure Web Server Load Balancing Parameters

1. Set and enable the IP addresses of the real Web servers.

```
# /cfg/slb/real 1

>> Real Server 1#rip 10.20.10.2

>> Real Server 1#ena

>> Real Server 1#../real 2

>> Real Server 2#rip 10.20.10.3

>> Real Server 2#ena
```

2. Add real Web servers 1 and 2 to real server group 1.

```
# /cfg/slb/group 1

>> Real server group 1#add 1
```

```
>> Real server group 1#add 2
```

3. Set and enable the IP address for Virtual Server 1, enable service on port 81, and connect real server group 1 to the virtual server.

   Enable service on port 81 for unencrypted communication between the VPN Gateways and the real Web servers. Recall that the real Web servers must also be configured to listen for AVG traffic on port 81. The preceding step also connects the real Web servers in server group 1 to the enabled virtual server. The HTTP service on port 80 for non-SSL Web traffic from clients to the real Web servers in real server group 1 is also enabled. Thus, the load balancing scheme for real server group 1 includes traffic on port 80 and 81.

```
# /cfg/slb/virt 1

>> Virtual Server 1#vip 192.168.10.100

>> Virtual Server 1#ena

>> Virtual Server 1#service 81

>> Virtual Server 1 81 Service#group 1

>> Virtual Server 1 81 Service#../service http

>> Virtual Server 1 http Service#group 1

>> Virtual Server 1http Service# apply
```

4. Enable client processing on port 1 leading to the Internet.

```
# /cfg/slb/port 1/client ena
```

5. Enable client processing on ports 2 and 3 leading to VPN Gateways.

```
# /cfg/slb/port 2

>> SLB Port 2#client ena

>> SLB Port 2#../port 3

>> SLB Port 3#client ena
```

6. Enable server processing on ports 7 and 8 leading to real servers.

```
# /cfg/slb/port 7

>> SLB Port 7#server ena

>> SLB Port 7#../port 8
```

```
>> SLB Port 8#server ena
```

7. Turn on Layer 4 processing.

```
# /cfg/slb/on
```

Configure AVG Load Balancing Parameters

Set and enable the IP addresses of the VPN Gateways, and create a group in the switch for load balancing.

1. For each VPN Gateway, create a Real Server IP address on the Application Switch.

```
# /cfg/slb/real 3

 >> Real server 3#rip 172.16.10.2

>> Real server 3#ena

>> Real server 3#../real 4

>> Real server 4#rip 172.16.10.3

>> Real server 4#ena
```

2. Create a Real Server Group and add the Real Servers (the AVGs in this case).

```
# /cfg/slb/group 2

>> Real server group 2#add 3

>> Real server group 2#add 4
```

3. Set the load balancing metric and health check type for real server group 2.

```
# /cfg/slb/group 2

>> Real server group 2#metric hash

>> Real server group 2#health tcp
```

4. Apply the changes.

```
# apply
```

Configure Filters

1. Create a filter to redirect client HTTPS traffic intended for port 443.

   When this filter is added to the switch port leading to the Internet, incoming HTTPS traffic is redirected to the VPN Gateways in real server group 2. Firewall redirect hash method is also enabled, using redirection based on hashing on both the source IP and the destination IP of the packets.

   The HTTPS traffic filter should be given a high number (a lower priority), such as 100, so as not to interfere with other filters.

```
# /cfg/slb/filt 100

>> Filter 100#proto tcp

>> Filter 100#dport https

>> Filter 100#action redir

>> Filter 100#group 2

>> Filter 100#rport https

>> Filter 100#adv/fwlb e

>> Filter 100Advanced# ../ena
```

2. Create a filter to deny client traffic intended for port 81.

   This filter, when placed on the client port leading to the Internet, blocks all incoming traffic destined for port 81. This blocking filter is required to ensure that traffic from clients outside your trusted network does not gain access to non-encrypted content on your real Web servers (content that would have been encrypted, had you not used the VPN Gateway for SSL offload purposes).

```
# /cfg/slb/filt 3

>> Filter 3#proto tcp

>> Filter 3#dport 81

>> Filter 3#action deny

>> Filter 3#ena
```

3. Create a default filter to allow all other traffic.

```
# /cfg/slb/filt 224
```

```
>> Filter 224#sip any
>> Filter 224#dip any
>> Filter 224#proto any
>> Filter 224#action allow
>> Filter 224#ena
```

4. Add the client filters to the client port leading to the Internet.

   This step adds the HTTPS redirect filter, the port 81 deny filter, and the default allow filter to the client port leading to the Internet.

```
# /cfg/slb/port 1
>> SLB Port 1#add 100
>> SLB Port 1#add 3
>> SLB Port 1#add 224
>> SLB Port 1#filt ena
```

5. Add an additional filter to allow for real server health checks.

   The health check filter should be given a smaller number (higher priority) than the redirection filter set in Step 6 on page 32 .

```
# /cfg/slb/filt 150
>> Filter 150#action allow
>> Filter 150#sip any
>> Filter 150#smask 0.0.0.0
>> Filter 150#dip 10.20.10.1
>> Filter 150#dmask 255.255.255.255
>> Filter 150#proto tcp
>> Filter 150#sport any
>> Filter 150#dport any
>> Filter 150#ena
```

6. Create a filter to redirect real server responses back to the VPN Gateway.

This filter, when added to the switch ports leading to the real Web servers, will redirect TCP traffic from port 81 back to the AVGs in group 2.

Firewall redirect hash method is also enabled, and the redirection is based on hashing using both the source IP and the destination IP of the packets. The return packets hash to the same IP address of the AVG in real server group 2, as from which the packets originated.

```
# /cfg/slb/filt 200

>> Filter 200#proto tcp

>> Filter 200#sport 81

>> Filter 200#action redir

>> Filter 200#group 2

>> Filter 200#adv/fwlb e

>> Filter 200 Advanced#../ena
```

7. Add the real server filter to the real server ports.

```
# /cfg/slb/port 7

>> SLB Port 7#add 150

>> SLB Port 7#add 200

>> SLB Port 7#add 224

>> SLB Port 7#filt ena

>> SLB Port 7#../port 8

>> SLB Port 8#add 150

>> SLB Port 8#add 200

>> SLB Port 8#add 224

>> SLB Port 8#filt ena
```

Apply, Save, and Verify the Configuration

1. Apply and save the Application Switch configuration changes.

```
# apply # save
```

2. Verify SSL Offload is working.

   Open a Web browser from the client side of the network. Access the following URLs:

   - `http://192.168.10.100`
   - `https://192.168.10.100`

   The second URL should prompt a security alert message.

Web Server Accelerator, using Return to Sender (RTS)

This configuration is similar to the one described in Web Server Accelerator on page 21, but uses the Return To Sender (RTS) feature on the Application Switch instead of redirect filters to ensure that the response from the real Web servers is sent back to the VPN Gateway that initiated the request. The RTS feature is available in Web OS 9.0 or later. For more information on the RTS feature, see your *Web OS Command Reference*.

With this configuration, you can use the regular HTTP port 80 also for the unencrypted information transmitted between the AVGs and the real Web servers. You can also use any of the available Application Switch load balancing metrics for the AVG group, without being restricted to using the `hash` metric.

Make the following modifications to the configuration example in Web Server Accelerator on page 21 in order to use the RTS feature.

Modify the AVG Configuration

1. On the VPN Gateway, change the TCP port used when initiating requests to the real Web servers.

```
# /cfg/ssl/server 1/rport

Current value: 0 [81]
Enter port to connect to:80
```

   Note:

   With this configuration change, the real Web servers only need to listen to one port; the default TCP port 80 used for HTTP.

2. Apply your changes.

```
>> Server 1#apply

Changes applied successfully.
```

Modify the Application Switch Configuration

1. On the Application Switch, enable Return To Sender on the switch ports leading to the VPN Gateways.

```
# /cfg/slb/port 2

>> SLB Port 2#rts ena

>> SLB Port 2#../port 3

>> SLB Port 3#rts ena
```

Note:

When RTS is enabled on a specific port, no filters can be used on that port. Filtering must however still be enabled on the port for RTS to work.

2. On the Application Switch, remove superfluous configuration settings remaining from the original Web Server Accelerator on page 21 configuration example.

```
# /cfg/slb/virt 1/service 81/del

# /cfg/slb/filt 3/del

# /cfg/slb/filt 150/del

# /cfg/slb/filt 200/del
```

3. Apply and save the configuration changes.

```
# apply
# save
```

Web Server Accelerator, with AVGs in Non-Transparent Proxy Mode

This configuration is similar to the one described in Web Server Accelerator on page 21. The main difference is that the VPN Gateway is configured to run in non-transparent proxy mode

—which means that the AVG's own IP address will be the source IP address when initiating a connection request towards a backend Web server. This configuration can be used with any Application Switch. The response from the real Web server is always sent back to the VPN Gateway that initiated the request, without the need of redirect filters or the RTS feature on an Application Switch.

With this configuration, you can use the regular HTTP port 80 also for the unencrypted information transmitted between the VPN Gateways and the real Web servers. You can also use any of the available Application Switch load balancing metrics for the AVG group, without being restricted to using the `hash` metric.

The main drawback with the AVGs running in non-transparent proxy mode is that logging of client IP addresses on the real Web servers can only be achieved when using HTTP or HTTPS, and then requires adding an extra HTTP header to the request initiated by the VPN Gateway. However, nearly all Web servers today have the capability to extract client IP information from this extra HTTP header, and add it to the log file.

To make the VPN Gateways run in non-transparent proxy mode when used as Web server accelerators, make the following modifications to the configuration example in Web Server Accelerator on page 21.

Modify the AVG Configuration

1. On the VPN Gateway, disable transparent proxy mode.

```
# /cfg/ssl/server 1/proxy

Current value: on

Proxy mode (on/off):off
```

2. Specify the real server IP address.

   Because transparent proxy is set to `off`, the real server IP address must explicitly be set to the virtual server IP address (as defined on the Application Switch).

```
>> Server 1#rip

Current value: 0.0.0.0
Enter IP address to connect to:192.168.10.100
```

(example virtual server IP address)

3. Change the TCP port used by the AVG when initiating requests to the real Web servers.

```
>> Server 1#rport
```

```
Current value: 0 [81]
Enter port to connect to (0-65534):80
```

Note:

With this configuration change, the real Web servers only need to listen to one port; the default TCP port 80 used for HTTP.

4. Change the virtual SSL server type from generic to http.

   This step is required in order to enable access to the HTTP Settings menu.

```
>> Server 1#type

Current value: generic
Type (generic/http):http
```

5. Configure the AVG to use the extra X-Forwarded-For HTTP header when initiating requests to the real Web servers.

```
>> Server 1#http

>> HTTP Settings#addxfor

Current value: off
Add X-Forwarded-For header (on/off/anonymous/remove):on
```

6. Apply your configuration changes.

```
>> HTTP Settings#apply
```

Modify the Application Switch Configuration

1. On the Application Switch, remove superfluous configuration settings remaining from the original Web Server Accelerator on page 21 configuration example.

```
# /cfg/slb/virt 1/service 81/del
# /cfg/slb/filt 3/del
# /cfg/slb/filt 150/del
# /cfg/slb/filt 200/del
# /cfg/slb/port 7/rem 224/filt dis
# /cfg/slb/port 8/rem 224/filt dis
```

2. Apply and save the Application Switch configuration changes.

```
# apply
# save
```

Content-Intelligent Switching for Secure Sessions

The following example configures the Application Switch to combine cookie-based persistence in cookie rewrite mode with SSL offload through the VPN Gateway. These steps assume that the network is already configured as described in Web Server Accelerator on page 21.

1. Connect to the Application Switch CLI.
2. Enable Direct Access Mode for the Application Switch.

   It is always necessary to enable Direct Access Mode (DAM) 52

   to be able to perform layer 7 functions like cookie inspection.

   ```
   # /cfg/slb/adv/direct ena
   ```

3. Configure cookie options for the HTTP service on TCP port 81.

   Active cookie mode (cookie rewrite mode) only works for cookies defined in the HTTP cookie header, not cookies defined in the URI. The switch can be configured to look for the cookie to rewrite in up to 16 server response packets in a TCP connection. This ensures that active cookie mode works well with HTTP 1.1, where multiple HTTP GET requests happen within the same TCP connection and the cookie may therefore not be present in the first server response packet.

   ```
   # /cfg/slb/virt 1/service 81

   >> Virtual Server 1 Service 81#pbind

   Enter client|cookie|sslid|disable persistence mode:cookie

   Enter passive|rewrite cookie persistance mode [p/r]:rewrite

   Enter Cookie Name:AlteonSession

   Enter the number of bytes to be extract:8

   Look for cookie in URL [e|d]:d
   ```

4. Apply and save the changes.

   ```
   # apply
   # save
   ```

Redundant Active-Standby Configuration

The following steps are used for configuring a redundant active-standby configuration with two Application Switches and two VPN Gateways, as illustrated. It is assumed that the first Application Switch is configured as in the example for Web Server Accelerator on page 21.

This configuration requires each VPN Gateway to be connected to one Application Switch. Even though only one Application Switch is active at a given time, the VPN Gateway connected to the Application Switch currently in standby mode will also process SSL traffic. A Layer 2 switch or hub should be placed between the real servers and the Application Switches, and also between the Application Switches and the clients.

Note:

Port 8 on VLAN 3 (configured in Web Server Accelerator on page 21) is not required for this example.

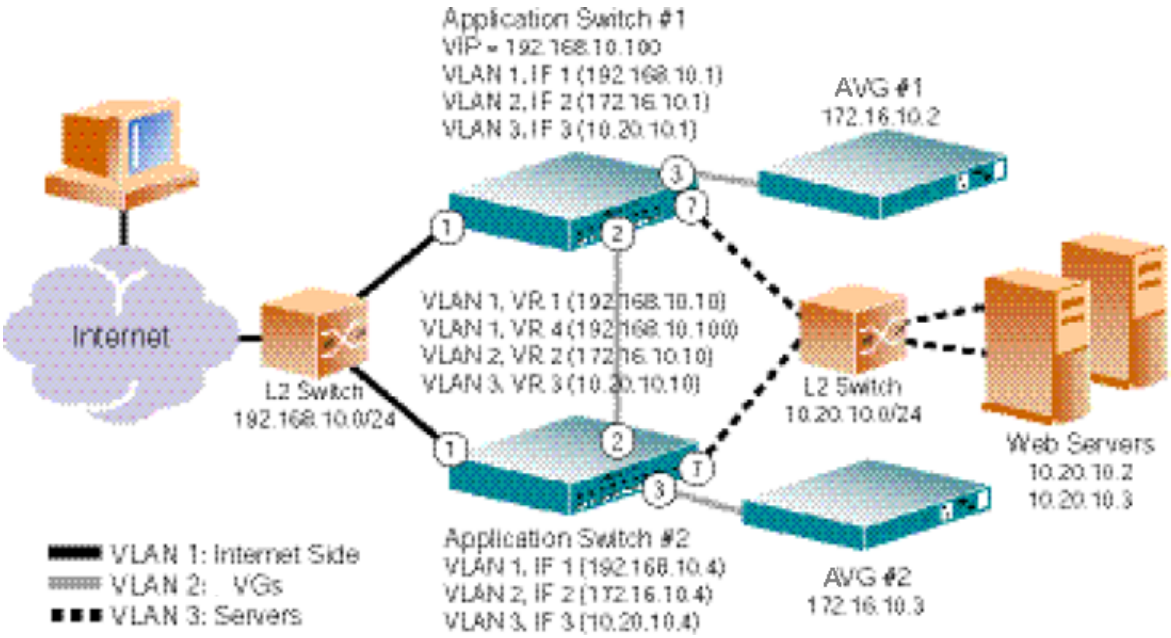

Figure 5: Redundant Active-Standby Configuration

In this process, you will perform the following tasks:

- Create three IP interfaces on the second Application Switch, each in a separate VLAN
- Enable VRRP and SLB on the second switch
- Configure the SLB sync peer

Configure Application Switch 2

Create the Necessary VLANs

In this configuration, there will be three VLANs: VLAN 1 for the Application Switch, VLAN 2 for the VPN Gateways, and VLAN 3 for the real servers. Since VLAN 1 is the default, only VLAN 2 and VLAN 3 require additional configuration.

1. On the second switch, log in as the administrator.
2. Configure VLAN 2 to include Application Switch ports leading to the VPN Gateways.

```
# /cfg/vlan 2

>> VLAN 2#add 2

Port 2 is an UNTAGGED port and its current PVID is 1.
Confirm changing PVID from 1 to 2 [y/n]:y

>> VLAN 2#add 3

Port 3 is an UNTAGGED port and its current PVID is 1.
Confirm changing PVID from 1 to 2 [y/n]:y

>> VLAN 2#ena
```

3. Configure VLAN 3 to include Application Switch ports leading to the real servers.

```
# /cfg/vlan 3

>> VLAN 3#add 7

Port 7 is an UNTAGGED port and its current PVID is 1.
Confirm changing PVID from 1 to 2 [y/n]:y

>> VLAN 3#ena
```

4. Disable Spanning Tree Protocol (STP) for the AVGs and real server ports.

   STP prevents loops in network topologies by removing redundant links. In active/standby configurations however, STP would eventually kill all links between the AVGs and the Application Switches, as well as between the real Web servers and the Application Switches. Therefore it must be disabled.

```
# /cfg/stp 1/port 2

>> Spanning Tree Port 2#off
```

```
>> Spanning Tree Port 2#../port 3
>> Spanning Tree Port 3#off
>> Spanning Tree Port 3#../port 7
>> Spanning Tree Port 7#off
```

Configure IP Interfaces for Each VLAN

1. Configure an IP interface for client traffic on the Application Switch.

```
# /cfg/ip/if 1
>> IP Interface 1#addr 192.168.10.4
>> IP Interface 1#mask 255.255.255.0
>> IP Interface 1#ena
```

2. Configure an IP interface for AVG traffic.

```
# /cfg/ip/if 2
>> IP Interface 2#addr 172.16.10.4
>> IP Interface 2#mask 255.255.255.0
>> IP Interface 2#vlan 2
>> IP Interface 2#ena
```

3. Configure an IP interface for the real server traffic.

```
# /cfg/ip/if 3
>> IP Interface 3#addr 10.20.10.4
>> IP Interface 3#mask 255.255.255.0
>> IP Interface 3#vlan 3
>> IP Interface 3#ena
```

4. Apply the changes.

```
# apply
```

Prepare to Receive Synchronization

Synchronize the configuration between two Application Switches.

1. Configure the synchronization parameters.

   Set the Application Switch 1 IP interface as peer 1 and disable synchronization of VRRP priorities.

```
# /cfg/slb/sync

>> Config Synchronization#prios d

>> Config Synchronization#peer 1

>> Peer Switch 1#addr 192.168.10.1

>> Peer Switch 1#ena
```

2. Apply and save the configuration changes on Application Switch 2.

```
# apply
# save
```

Configure Application Switch 1

Configure VRRP

VRRP is configured for failover (redundancy) between two Application Switches, in the event one of the Application Switches fails.

1. On Application Switch 1, log in as the administrator.
2. Globally turn on VRRP.

```
# /cfg/vrrp/on
```

3. Configure virtual router 1.

```
# /cfg/vrrp/vr 1

>> VRRP Virtual Router 1#vrid 1

>> VRRP Virtual Router 1#addr 192.168.10.10

>> VRRP Virtual Router 1#if 1
```

```
>> VRRP Virtual Router 1#prio 101
>> VRRP Virtual Router 1#share dis
>> VRRP Virtual Router 1#track/14pts e
>> VRRP Virtual Router 1 Priority Tracking#../ena
```

4. Configure virtual router 2.

```
# /cfg/vrrp/vr 2
>> VRRP Virtual Router 2#vrid 2
>> VRRP Virtual Router 2#addr 172.16.10.10
>> VRRP Virtual Router 2#if 2
>> VRRP Virtual Router 2#prio 101
>> VRRP Virtual Router 2#share dis
>> VRRP Virtual Router 2#track/14pts e
>> VRRP Virtual Router 2 Priority Tracking#../ena
```

5. Configure virtual router 3.

```
# /cfg/vrrp/vr 3
>> VRRP Virtual Router 3#vrid 3
>> VRRP Virtual Router 3#addr 10.20.10.10
>> VRRP Virtual Router 3#if 3
>> VRRP Virtual Router 3#prio 101
>> VRRP Virtual Router 3#share dis
>> VRRP Virtual Router 3#track/14pts e
>> VRRP Virtual Router 3 Priority Tracking#../ena
```

6. Configure virtual router 4, which is the virtual server router.

```
# /cfg/vrrp/vr 4
>> VRRP Virtual Router 4#vrid 4
>> VRRP Virtual Router 4#addr 192.168.10.100
```

```
>> VRRP Virtual Router 4#if 1

>> VRRP Virtual Router 4#prio 101

>> VRRP Virtual Router 4#share dis

>> VRRP Virtual Router 4#track/14pts e

>> VRRP Virtual Router 4 Priority Tracking#../ena
```

Note:

Make sure the AVGs are configured to use the IP address of Virtual Router 2 on VLAN 2 as their default gateway. For more information about gateway configuration, see the **gateway** command under "System Configuration" in the *Command Reference* . Likewise, the Web servers must be configured to use the IP address of Virtual Router 3 on VLAN 3 as their default gateway.

Prepare and Send Synchronization

1. Configure the Synchronization parameters.

```
# /cfg/slb/sync

>> Config Synchronization#prios d

>> Config Synchronization#peer 1

>> Peer Switch 1#addr 192.168.10.4

>> Peer Switch 1#ena
```

2. Apply and save the configuration changes on Application Switch 1.

```
# apply # save
```

3. Synchronize server load balancing configuration on peers.

```
# /oper/slb/sync

Synchronizing VRRP, FILT, PORT and SLB configuration
to 192.168.10.4

Confirm synchronizing the configuration to 192.168.10.4 [y/n]:y
```

Caution:

The `/oper/slb/sync` command will push the filter, port, server load balancing and VRRP configuration from the current switch to the peer switch.

Verify Failover

Check for the master switch.

```
# /info/vrrp
```

Check secure connection on the client. Disconnect link on master and verify failover to the backup switch.

Redundant Active-Standby Configuration with AVG Port Failover

The following steps are used for configuring a port failover solution, in which each VPN Gateway is connected to two Application Switches by using two physical network ports on each AVG. Should one network port fail on a VPN Gateway, the active link is immediately switched over to the other port and the AVG continues to process SSL requests uninterrupted.

The port failover configuration example assumes that you already have a redundant active-standby configuration with two Application Switches and two VPN Gateways, as described in Redundant Active-Standby Configuration on page 39. Note that the Redundant Active-Standby configuration example in turn assumes that the first Application Switch is configured as in the example for Web Server Accelerator on page 21.

If your AVG cluster consists of more than two devices, you may want to place Layer 2 switches between the AVGs and the Application Switches in order to decrease the number of ports used on the Application Switches. In that case, port 1 on each VPN Gateway should be connected to one Layer 2 switch, and port 2 on each VPN Gateway connected to another Layer 2 switch. Both Layer 2 switches must in turn be connected to both Application Switches. With Layer 2 switches placed between the AVGs and the Application Switches, no primary port should be configured in the AVG cluster.

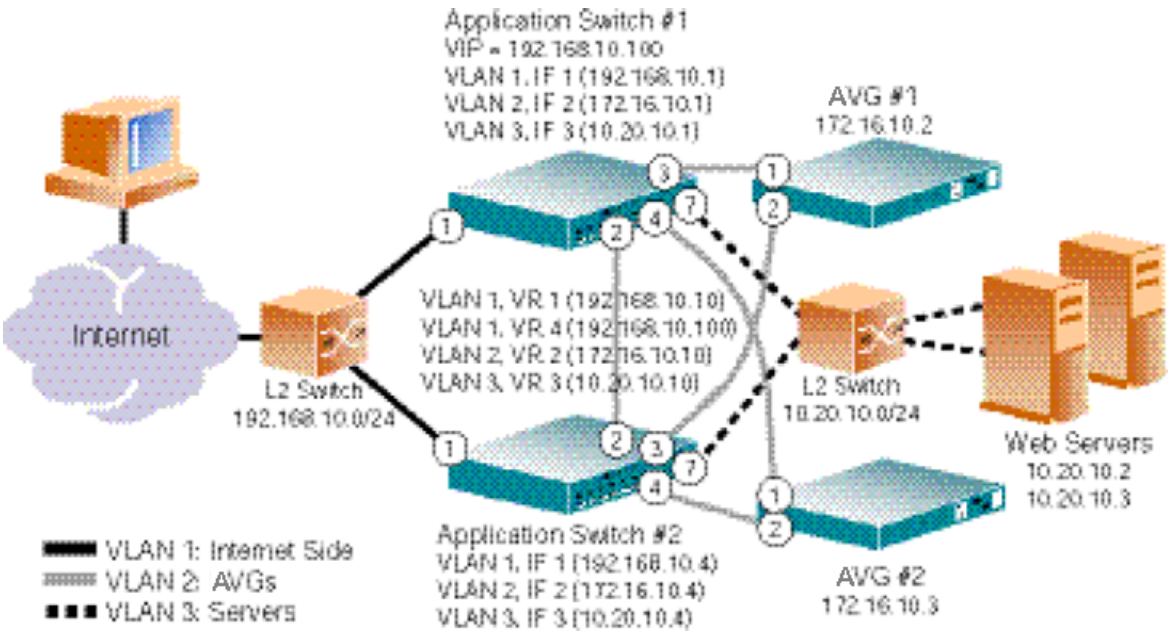

Figure 6: Redundant Active-Standby Configuration with Port Failover

In this process you will perform the following tasks:

- On Application Switch 1, configure VLAN 2 to include switch port 4
- On Application Switch 2, configure VLAN 2 to include switch port 4
- In the AVG cluster, add an extra port to Interface 1 on both VPN Gateways
- In the AVG cluster, specify a primary port for Interface 1 on both VPN Gateways

Configure Application Switch 1

1. On Application Switch 1, log in as the administrator.
2. Configure VLAN 2 to include switch port 4, which is used for the extra connection leading to the failover port on the first VPN Gateway.

```
# /cfg/vlan 2

>> VLAN 2#add 4

Port 4 is an UNTAGGED port and its current PVID is 1.

Confirm changing PVID from 1 to 2 [y/n]:y

>> VLAN 2#ena
```

3. Apply and save the configuration changes on Application Switch 1.

```
# apply
# save
```

Configure Application Switch 2

1. On Application Switch 2, log in as the administrator.
2. Configure VLAN 2 to include switch port 4, which is used for the extra connection leading to the failover port on the second VPN Gateway.

```
# /cfg/vlan 2

>> VLAN 2#add 4

Port 4 is an UNTAGGED port and its current PVID is 1.

Confirm changing PVID from 1 to 2 [y/n]:y

>> VLAN 2#ena
```

3. Apply and save the configuration changes on Application Switch 2.

```
# apply
# save
```

Configure the AVGs

All configuration changes will automatically be propagated to all VPN Gateways in the cluster.

1. Log in as the administrator to the AVG cluster by using a Telnet or SSH connection to the Management IP address (MIP). You can also use a direct serial connection to one of the AVGs.

```
login:admin

Password:
```

2. Examine the current network and port configuration in the cluster. (If necessary, scroll down in the terminal window to view the relevant information).

```
>> Main#/cfg/sys/cur
```

```
System:
Management IP (MIP) address = 10.1.82.144

iSD Host 1:
Type of the iSD = master
IP address = 10.1.82.145
License =
IPSEC user sessions: 10
TPS: unlimited
SSL user sessions: 10
Default gateway address = 10.1.82.2
Ports = 1 : 2
Hardware platform = 200
Host Routes:
No items configured
Host Interface 1:
IP address = 10.1.82.145
Network mask = 255.255.255.0
VLAN tag id = 0
Mode = failover
Primary port = 0
Interface Ports:
2
Host Port 1:
Autonegotiation = on
Speed = 0
Full or half duplex mode = full
Host Port 2:
Autonegotiation = on
Speed = 0
Full or half duplex mode = full
```

One interface is configured per VPN Gateway (iSD host). This is the default Interface 1. When the first VPN Gateway (iSD host 1) was installed in a new cluster it was assigned Interface 1, an IP address and a port. When the second VPN Gateway (iSD host 2) was joined to the cluster, this VPN Gateway was assigned its own Interface 1, another IP address and a port.

The line `Interface Ports:` shows the physical port number currently assigned to Interface 1 per iSD host. In this example, port 1 has been assigned to Interface 1 per iSD host. One more port must be added to Interface 1 for both VPN Gateways in order to create a failover configuration.

Each iSD host is equipped with 3 physical ports. The host-specific port information (Ports = 1,2: 3) shows that port 1 and port 2 can exist on the same network in a failover solution, while none of these ports can coexist with port 3 on the same interface. Ports that can coexist on the same interface appear grouped together separated by comma (,).

Given the preceding information, port 2 must now be added to Interface 1 on both VPN Gateways in order to create a failover configuration.

Note:

If port 3 had been assigned to interface 1 in the preceding example, you would need to first delete port 3, then add port 1 and port 2 to Interface 1 in order to configure a failover solution within Interface 1.

3. Add port 2 to Interface 1 for iSD host 1.

```
/cfg/sys/host 1/interface 1/ports/add

Port to add:2
```

4. Set port 1 as the primary port in Interface 1.

```
>> Interface Ports#../primary

Current value: 0

Enter the primary port of the Interface:1
```

5. Add port 2 to Interface 1 for iSD host 2.

```
/cfg/sys/host 2/interface 1/ports/add

Port to add:2
```

6. Set port 1 as the primary port in Interface 1.

```
>> Interface Ports#../primary

Current value: 0

Enter the primary port of the Interface:1
```

7. Apply your changes.

```
>> Host Interface 1#apply
```

Mail Server Accelerator

Figure 7: Sample Mail Server Accelerator Network Using Multiple AVGs on page 50 illustrates the same network configuration for the VPN Gateway as in the Web server accelerator example, using an Application Switch. However, this configuration example describes how to use the AVGs for mail server accelerator purposes.

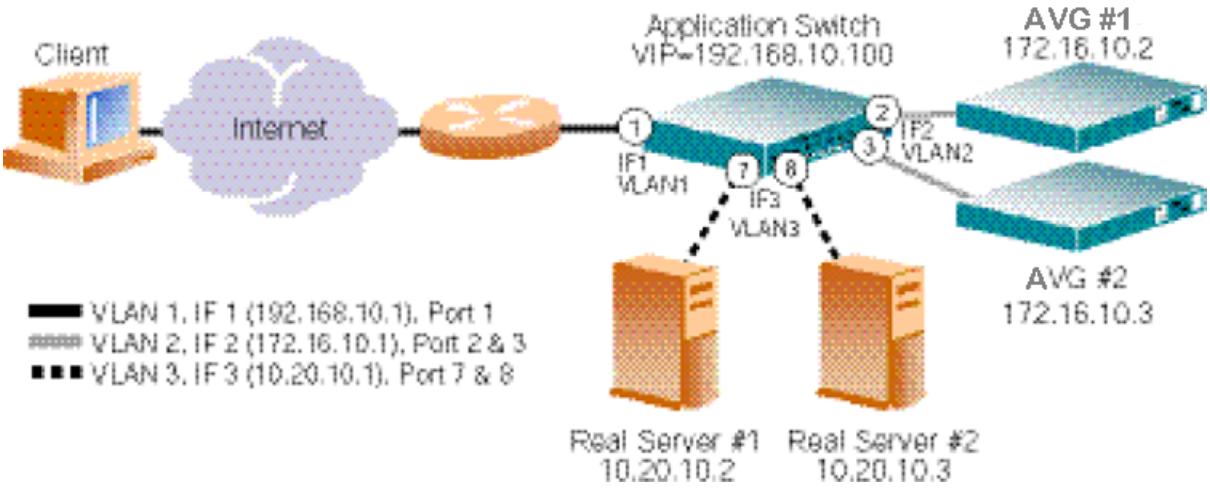

Figure 7: Sample Mail Server Accelerator Network Using Multiple AVGs

Initial Setup

This configuration example assumes that you have performed the initial setup of the two AVGs as described in the "Initial Setup" chapter in the *Users Guide* . The example also assumes that you have added a server certificate as described in Add a Server Certificate to the AVG on page 22.

Configure the AVGs

Log in as the administrator to the AVG cluster by using a Telnet or SSH connection to the Management IP address (MIP). The configuration changes will automatically be propagated to all VPN Gateways in the cluster.

Create a Virtual SSL Server for SMTPS

1. Create the necessary virtual SSL server for SMTPS.

   This step creates a new virtual SSL server in the VPN Gateway. Each virtual SSL server listens to a specific TCP port and is mapped to a virtual server on the Application Switch. If you have already created a virtual SSL server for HTTPS services, you must create additional virtual servers for SMTPS and POP3S mail services. Each virtual SSL server must be assigned a unique number.

```
# /cfg/ssl/server

Enter virtual server number: (1-)1
```

```
Creating new server 1
>> Server 1#
```

2. Define a name for virtual SSL server 1.

   This step lets you specify a name, by which you can identify SSL server 1. To view the numbers and related names of all configured SSL servers, use the `/cfg/ssl/cur` command. The name you specify is mainly intended for your own reference, and is not critical for the configuration itself. As the preceding example suggests, the name can indicate the service for which the SSL server was created.

```
>> Server 1#name

Current value:" "

Enter new SSL server name:SMTPS
```

3. Set listen TCP port for virtual SSL server 1.

   Each time you create a new SSL server, the listen port is automatically set to 443. Since you are setting up the VPN Gateway for secure mail offload purposes, configure the listen port to 465, which is the TCP port used by SMTPS.

```
>> Server 1#port

Current value: 443 (https)

Enter listen port number:465
```

4. Connect virtual SSL server 1 to the desired Virtual Server IP address on the Application Switch.

   This step connects SSL server 1 to the IP address of the desired virtual server on the Application Switch.

```
>> Server 1#vips

Current value:""

Enter server ips (comma separated):192.168.10.100
```

5. Set the Real Server IP address to which SSL Server 1 should connect when initiating requests.

   Preserve the current value of the Real Server IP address, which should be 0.0.0.0. At first glance this configuration may perhaps seem a bit odd. However, by specifying 0.0.0.0 as the Real Server IP address, the SSL server is instructed to use the destination IP address (in the received packets) when initiating requests sent to the Virtual Server IP address. Because the destination IP address in the

received packets corresponds to the IP address of the virtual server, the requests will always reach the correct Virtual Server IP address.

```
>> Server 1#rip

Current value: 0.0.0.0

Enter IP address to connect to: 0.0.0.0
```

6. Set the server port to which SSL server 1 should connect when initiating requests.

   This step sets the TCP port to which SSL Server 1 connects when initiating requests.

```
>> Server 1#rport

Current value: 0 [81]

Enter port to connect to (0-65534):25
```

7. Specify the certificate to be used by virtual SSL Server 1.

   Note that you are prompted to type the number of an existing certificate, not the name assigned to a certificate. To view all certificates currently added to the VPN Gateway, use the `/cfg/ssl/cur` command. For more information about how to add a certificate, see the "Certificates and Client Authentication" chapter in the *Users Guide* .

```
>> Server 1#ssl

>> SSL Settings for Server 1#cert

Current value: <not set>

Enter certificate number: (1-)1
```

Create the Virtual SSL Server for POP3S

In this configuration example, POP3S is used together with SMTPS in the messaging system. However, IMAPS is a viable alternative to POP3S and is supported by many mail clients. To use IMAPS, configure the appropriate virtual SSL server name in and TCP listen port in step 3 on page 53.

1. Create the necessary virtual SSL server for POP3S.

   This step creates a virtual SSL server intended for POP3S services. Recall that each virtual SSL server must be assigned a unique number.

```
# /cfg/ssl/server

Enter virtual server number: (1-)2

Creating new server 2
>> Server 2#
```

2. Define a name for virtual SSL server 2.

```
>> Server 2#name

Current value:""

Enter new SSL server name:POP3S
```

3. Set listen TCP port for virtual SSL server 2.

   This step sets the listen port of virtual SSL server 2 to 995, which is the TCP port used by POP3S.

```
>> Server 2#port

Current value: <not set> [443 (https)]

Enter listen port number:995
```

4. Map SSL server 2 to the Virtual Server IP address on the Application Switch.

   This step maps SSL server 2 to the same Virtual Server IP address on the Application Switch as the one to which you mapped SSL server 1.

```
>> Server 2#vips

Current value:""

Enter server ips (comma separated):192.168.10.100
```

5. Set the Real Server IP address to which SSL Server 2 should connect when initiating requests.

   As with virtual SSL server 1, preserve the current value of 0.0.0.0 for virtual SSL server 2.

```
>> Server 1#rip

Current value: 0.0.0.0

Enter IP address to connect to:0.0.0.0
```

6. Set the server port to which SSL server 2 should connect when initiating requests.

   This step sets the TCP port, to which the SSL Server 2 connects when initiating requests.

   As you will see further ahead, the virtual server will have services enabled on both port 25 (SMTP) and 110 (POP3) to match the settings on virtual SSL server 1 and 2 respectively.

```
>> Server 2#rport

Current value: 0 [81]

Enter port to connect to (0-65534):110
```

7. Specify the certificate to be used by virtual SSL Server 2.

   Specify the same certificate number as you did for virtual SSL server 1.

```
>> Server 2#ssl

>> SSL Settings for Server 2#cert

Current value: <not set>

Enter certificate number: (1-)1
```

8. Apply the changes.

```
>> SSL Settings for Server 2#apply
```

Configure the Application Switch

Create the Necessary VLANs

In this configuration, there will be three VLANs: VLAN 1 for the Application Switch that connects to the Internet, VLAN 2 for the VPN Gateways, and VLAN 3 for the real mail servers. Since VLAN 1 is the default, only VLAN 2 and VLAN 3 require additional configuration. Note that you will perform all of the following the sample configuration changes on the Application Switch.

1. Configure VLAN 2 to include Application Switch ports leading to the VPN Gateways.

```
# /cfg/vlan 2
```

```
>> VLAN 2#add 2

Port 2 is an UNTAGGED port and its current PVID is 1.
Confirm changing PVID from 1 to 2 [y/n]:y

>> VLAN 2#add 3

Port 3 is an UNTAGGED port and its current PVID is 1.
Confirm changing PVID from 1 to 2 [y/n]:y

>> VLAN 2#ena
```

2. Configure VLAN 3 to include Application Switch ports leading to the real servers.

```
# /cfg/vlan 3

>> VLAN 3#add 7

Port 7 is an UNTAGGED port and its current PVID is 1.
Confirm changing PVID from 1 to 2 [y/n]:y

>> VLAN 3#add 8

Port 8 is an UNTAGGED port and its current PVID is 1.
Confirm changing PVID from 1 to 2 [y/n]:y

>> VLAN 3#ena
```

3. Disable Spanning Tree Protocol (STP) for the AVG ports 2 and 3 and real server ports 7 and 8.

```
# /cfg/stp/port 2

>> Spanning Tree Port 2#off

>> Spanning Tree Port 2#../port 3

>> Spanning Tree Port 3#off

>> Spanning Tree Port 3#../port 7

>> Spanning Tree Port 7#off

>> Spanning Tree Port 7#../port 8

>> Spanning Tree Port 8#off
```

Configure One IP Interface for Each VLAN

Note:

If you prefer, you can reverse the order of the first two commands (addr and mask) in the following example. By entering the mask first, the Application Switch will automatically

calculate the correct broadcast address for you. The calculated broadcast address is displayed immediately after you provide the IP address of the interface, and will be applied together with the other settings when you execute the **apply** command.

1. Configure an IP interface for client traffic on the Application Switch with VLAN 1.

```
# /cfg/ip/if 1
>> IP Interface 1#addr 192.168.10.1
>> IP Interface 1#mask 255.255.255.0
>> IP Interface 1#broad 192.168.10.255
>> IP Interface 1#vlan 1
>> IP Interface 1#ena
```

2. Configure an IP interface for AVG traffic with VLAN 2.

```
# /cfg/ip/if 2
>> IP Interface 2#addr 172.16.10.1
>> IP Interface 2#mask 255.255.0.0
>> IP Interface 2#broad 172.16.255.255
>> IP Interface 2#vlan 2
>> IP Interface 2#ena
```

3. Configure an IP interface for the real server traffic with VLAN 3.

```
# /cfg/ip/if 3
>> IP Interface 3#addr 10.20.10.1
>> IP Interface 3#mask 255.255.255.0
>> IP Interface 3#broad 10.20.10.255
>> IP Interface 3#vlan 3
>> IP Interface 3#ena
```

4. Apply the changes.

```
# apply
```

Configure Mail Server Load Balancing Parameters

1. Set and enable the IP addresses of the real mail servers.

```
# /cfg/slb/real 1

>> Real Server 1#rip 10.20.10.2

>> Real Server 1#ena

>> Real Server 1#../real 2

>> Real Server 2#rip 10.20.10.3

>> Real Server 2#ena
```

2. Add real mail servers 1 and 2 to real server group 1.

```
# /cfg/slb/group 1

>> Real server group 1#add 1

>> Real server group 1#add 2
```

3. Set and enable the IP address for Virtual Server 1, enable services for SMTP and POP3, and connect real server group 1 to the Virtual Server.

   Enable services for SMTP and POP3 on Virtual Server 1 for unencrypted communication between the AVGs and the real mail servers. This step also connects the real mail servers in server group 1 to the enabled virtual server.

   Note:

   Assigning the same group to SMTP and POP3 services means that both services will be blocked if one service fails.

```
# /cfg/slb/virt 1

>> Virtual Server 1#vip 192.168.10.100

>> Virtual Server 1#ena

>> Virtual Server 1#service smtp

>> Virtual Server 1 smtp Service#group 1

>> Virtual Server 1 smtp Service#../service pop3

>> Virtual Server 1 pop3 Service#group 1
```

```
>> Virtual Server 1 pop3 Service#apply
```

4. Enable client processing on port 1 leading to the Internet.

```
# /cfg/slb/port 1/client ena
```

5. Enable client processing on ports 2 and 3 leading to AVGs.

```
# /cfg/slb/port 2
>> SLB Port 2#client ena
>> SLB Port 2#../port 3
>> SLB Port 3#client ena
```

6. Enable server processing on ports 7 and 8 leading to real servers.

```
# /cfg/slb/port 7
>> SLB Port 7#server ena
>> SLB Port 7#../port 8
>> SLB Port 8#server ena
```

7. Turn on Layer 4 processing.

```
# /cfg/slb/on
```

8. Apply the changes.

```
# apply
```

Configure AVG Load Balancing Parameters

Set and enable the IP addresses of the VPN Gateways, and create a group in the switch for load balancing.

1. For each VPN Gateway, create a Real Server IP address in the switch

```
# /cfg/slb/real 3
>> Real server 3#rip 172.16.10.2
>> Real server 3#ena
>> Real server 3#../real 4
```

```
>> Real server 4#rip 172.16.10.3

>> Real server 4#ena
```

2. Create a Real Server Group and add the Real Servers (the AVGs in this case)

```
# /cfg/slb/group 2

>> Real server group 2#add 3

>> Real server group 2#add 4
```

3. Set the load balancing metric and the health check type for Real Server Group 2.

```
# /cfg/slb/group 2

>> Real server group 2#metric hash

>> Real server group 2#health tcp
```

4. Apply the changes.

```
# apply
```

Configure Filters

1. Create a filter to redirect client SMTPS traffic intended for port 465.

   When this filter is added to the switch port leading to the Internet, all incoming SMTPS traffic is redirected to the AVGs in real server group 2. Firewall redirect hash method is also enabled, using redirection based on hashing on both the source IP and the destination IP of the packets.

   The SMTPS traffic filter should be given a high number (a lower priority), such as 110, so as not to interfere with other filters.

```
# /cfg/slb/filt 110

>> Filter 110#proto tcp

>> Filter 110#dport 465

>> Filter 110#action redir

>> Filter 110#group 2

>> Filter 110#rport 465
```

```
>> Filter 110#adv/fwlb e

>> Filter 110 Advanced#../ena
```

2. Create a filter to redirect client POP3S traffic intended for port 995.

   When this filter is added to the switch port leading to the Internet, all incoming POP3S traffic is redirected to the AVGs in real server group 2. Firewall redirect hash method is also enabled, using redirection based on hashing on both the source IP and the destination IP of the packets.

   The POP3S traffic filter should be given a high number (a lower priority), such as 120, so as not to interfere with other filters.

```
# /cfg/slb/filt 120

>> Filter 120#proto tcp

>> Filter 120#dport 995

>> Filter 120#action redir

>> Filter 120#group 2

>> Filter 120#rport 995

>> Filter 120#adv/fwlb e

>> Filter 120 Advanced#../ena
```

3. Create filter to deny client traffic intended for port 25 (SMTP).

   This filter, when placed on the client port leading to the Internet, blocks all incoming traffic destined for port 25. This blocking filter is optional but ensures that traffic from clients outside your trusted network does not gain non-encrypted access to your real servers (content that would have been encrypted, had you not used the VPN Gateway for SSL offload purposes).

```
# /cfg/slb/filt 4

>> Filter 4#proto tcp

>> Filter 4#dport 25

>> Filter 4#action deny

>> Filter 4#ena
```

4. Create filter to deny client traffic intended for port 110 (POP3).

This filter, when placed on the client port leading to the Internet, blocks all incoming traffic destined for port 110.

```
# /cfg/slb/filt 5

>> Filter 5#proto tcp

>> Filter 5#dport 110

>> Filter 5#action deny

>> Filter 5#ena
```

5. Create a default filter to allow all other traffic.

```
# /cfg/slb/filt 224

>> Filter 224#sip any

>> Filter 224#dip any

>> Filter 224#proto any

>> Filter 224#action allow

>> Filter 224#ena
```

6. Add the client filters to the client port leading to the Internet.

   This step adds the SMTPS and POP3S redirect filters, the optional SMTP and POP3 deny filters and the default allow filter to the client port leading to the Internet.

```
# /cfg/slb/port 1

>> SLB Port 1#add 110

>> SLB Port 1#add 120

>> SLB Port 1#add 4

>> SLB Port 1#add 5

>> SLB Port 1#add 224

>> SLB Port 1#filt ena
```

7. Add an additional filter to allow for real SMTP server health checks.

   The health check filter should be given a smaller number (higher priority) than the redirection filters set for the real server responses back to the AVGs.

```
# /cfg/slb/filt 150
```

```
>> Filter 150#action allow
>> Filter 150#sip any
>> Filter 150#smask 0.0.0.0
>> Filter 150#dip 10.20.10.1
>> Filter 150#dmask 255.255.255.255
>> Filter 150#proto tcp
>> Filter 150#sport any
>> Filter 150#dport any
>> Filter 150#ena
```

Note:

If you have configured the Application Switch for Web server accelerator purposes, the health check filter above already exists. In that case, the same filter can be used for mail server accelerator purposes.

8. Create a filter to redirect real server responses back to the AVG.

   This filter, when added to the switch ports leading to the real mail servers, will redirect TCP traffic from port 25 back to the VPN Gateways in group 2.

   Firewall redirect hash method is also enabled, which means the packets are hashed based on both the source IP and the destination IP. This makes the return packets hash to the same IP address of the VPN Gateway in real server group 2, as from which the packets originated.

```
# /cfg/slb/filt 210
>> Filter 210#proto tcp
>> Filter 210#sport 25
>> Filter 210#action redir
>> Filter 210#group 2
>> Filter 210#adv/fwlb e
>> Filter 210 Advanced#../ena
```

9. Create a filter to redirect real server responses back to the VPN Gateway.

   This filter, when added to the switch ports leading to the real Web servers, will redirect TCP traffic from port 110 back to the VPN Gateways in group 2.

Firewall redirect hash method is also enabled, which means the packets are hashed based on both the source IP and the destination IP. This makes the return packets hash to the same IP address of the VPN Gateway in real server group 2, as from which the packets originated.

```
# /cfg/slb/filt 220
>> Filter 220#proto tcp
>> Filter 220#sport 110
>> Filter 220#action redir
>> Filter 220#group 2
>> Filter 220#adv/fwlb e
>> Filter 220 Advanced#../ena
```

10. Add the real server filters to the real server ports.

```
# /cfg/slb/port 7
>> SLB Port 7#add 150
>> SLB Port 7#add 210
>> SLB Port 7#add 220
>> SLB Port 7#add 224
>> SLB Port 7#filt ena
>> SLB Port 7#../port 8
>> SLB Port 8#add 150
>> SLB Port 8#add 210
>> SLB Port 8#add 220
>> SLB Port 8#add 224
>> SLB Port 8#filt ena
```

Apply and Save the Configuration

Apply and save the Application Switch configuration changes.

```
# apply
# save
```

You should also instruct the intended users of the service that they must enable secure E-mail in their mail client application. If this feature is not supported, they may need to upgrade to a newer version.

Chapter 5: Web Server Accelerator with End to End Encryption

This chapter describes how to configure the Avaya VPN Gateway (AVG) to be used as a Web server accelerator, where the information exchanged on the network segment between the AVGs and the real Web servers is encrypted. Such a configuration can be desirable, or even required, in many high security environments such as the financial, medical, insurance, corporate, or government sector. For a description of the basic operation of the VPN Gateway when used as a Web server accelerator with endpoint encryption, see Basic Operation of AVG as Web Server Accelerator with End to End Encryption on page 19 .

However, only setting up a new SSL connection between the VPN Gateway and the backend servers would degrade the performance of the backend servers considerably when end to end encryption is involved, and the benefits of using the AVG as an offload device would be impaired. In order to alleviate the performance degradation, three different methods that interact with each other are part of the AVG:

- SSL Connect on page 65
- Connection Pooling on page 66
- Load Balancing on page 66

Concepts

SSL Connect

SSL Connect enables the VPN Gateway to set up an SSL connection to the backend servers, while at the same time terminating the SSL connection to the client. This provides the means for authenticating the backend server and securing the information that is exchanged also on this network segment. Because a new SSL connection is established between the VPN Gateway and the backend server, a lighter encryption algorithm can be used on this more protected network segment in order to improve performance. For server authentication purposes, you can generate a server certificate using the CLI on the VPN Gateway, and then install the certificate and the private key on the backend server. For client authentication purposes, you can generate a client certificate on the VPN Gateway, and then install the client certificate and the private key on the VPN Gateway. Provided the backend server is configured to require a client certificate, the VPN Gateway will present the configured client certificate.

Furthermore, since the VPN Gateway will be acting as a client towards the backend server, the AVG will always try to reuse previously established SSL sessions. The SSL session reuse

attempts will be successful, since the backend server recognizes the VPN Gateway as a client that connects repeatedly. SSL session reuse between the VPN Gateway and the backend server helps in lowering the overhead involved in performing a full SSL handshake.

Connection Pooling

Connection Pooling enables the VPN Gateway to reuse the server side of a socket for a future client request. Sockets can be pooled and reused when either the client closes its end of the socket, or when a specific time frame with no traffic on the client socket has elapsed. Sockets are only pooled when all the HTTP requests that were received on the client side socket have been fully replied to on the server side socket. If these criteria are met, the pooling occurs when the proxy is just about to close the server side socket. The next time a client initiates a request, a server side socket available in the pool can be reused instead of establishing a new socket with all the overhead involved. One pool of server side sockets is maintained for each backend server that responds positively to the health checking performed at regular intervals. This mechanism improves performance and enables the VPN Gateway to handle more connections per second when SSL connect is enabled (for end to end encryption purposes).

Load Balancing

Load Balancing of backend servers must be performed by the VPN Gateway when configured for end to end encryption, since the Application Switch will only see encrypted traffic. The available AVG load balancing metrics include a hash algorithm on the client source IP address, a round robin algorithm, or a real time measurement of which backend server currently has fewest open connections. Load balancing can also be used in combination with a health check mechanism to minimize the number of failed connections. The load balancing of backend servers can also be combined with cookie-based persistency or session-based persistency.

Figure 8: Sample Web Server Accelerator Network Using Multiple AVGs on page 67 illustrates the most common network configuration for the AVG. This configuration example consists of two VPN Gateways, an Application Switch and two servers.

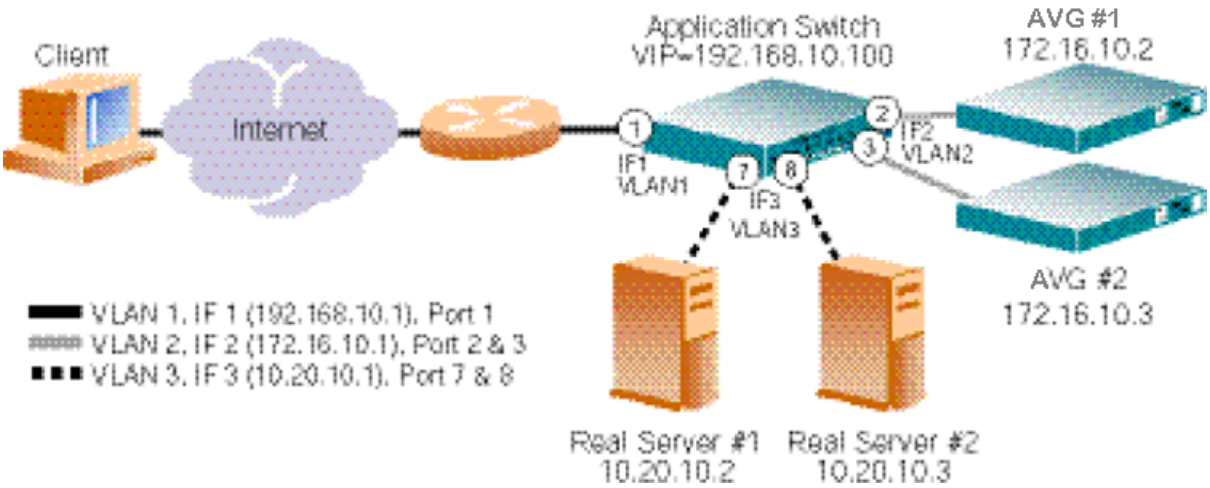

Figure 8: Sample Web Server Accelerator Network Using Multiple AVGs

Configuring End to End Encryption

Initial Setup of AVGs

This configuration example assumes that you have performed the initial setup of the two VPN Gateways as described in the "Initial Setup" chapter in the *Users Guide*. The example also assumes that you have added a server certificate as described in Add a Server Certificate to the AVG on page 22 .

Configure the AVGs

Log in as the administrator to the AVG cluster by using a Telnet or SSH connection to the Management IP address (MIP). The configuration changes will automatically be propagated to all VPN Gateways in the cluster.

Create and Configure a Virtual SSL Server

1. Create a virtual SSL server.

   This step creates a new virtual SSL server on the VPN Gateway. Each virtual SSL server listens to a specific TCP port and is connected to a virtual server IP address on the Application Switch.

```
# /cfg/ssl/server

Enter virtual server number: (1-)1

Creating Server 1
>> Server 1#
```

2. Define a name for virtual SSL server 1.

   This step lets you specify a name, by which you can identify SSL server 1. To view the numbers and related names of all configured SSL servers, use the `/cfg/ssl/cur` command. The name you specify is mainly intended for your own reference, and is not critical for the configuration itself. As the preceding example suggests, the name can indicate the service for which the SSL server was created.

```
>> Server 1#name

Current value :""

Enter new SSL server name :HTTPS
```

3. Set listen TCP port for SSL server 1.

   Each time you create a new SSL server, the listen port is automatically set to 443. Since you are setting up the AVG for HTTPS offload purposes in this example, it is not really necessary to configure the SSL server to listen to port 443. However, for using the AVG for any protocol other than HTTPS, a new virtual SSL server must be configured to listen to the TCP port of the intended service.

```
>> Server 1#port

>> Current value: <not set> [443 (https)]
>> Enter listen port number :443
```

4. Connect the SSL server to the desired Virtual Server IP address on the Application Switch.

   This step connects SSL Server 1 to the IP address of the desired virtual server on the Application Switch.

```
>> Server 1#vips

Current value :""

Enter server ips192.168.10.100
```

5. Specify the certificate to be used by SSL Server 1 in SSL handshakes with Internet clients.

   You are prompted to type the index number of an existing certificate. The certificate you specify in this step is the certificate that is sent to a client browser during the

SSL handshake. To view all certificates currently added to the AVG by index number and name, use the `/cfg/ssl/cur` command. For more information on how to add a certificate to the AVG, see the "Certificates and Client Authentication" chapter in the *Users Guide* .

```
>> Server 1#ssl

>> SSL Settings#cert

Current value: <not set>
Enter certificate number: (1-)1
```

Note:

If the certificate you specify is a chained certificate, you need to first add the CA certificates up to and including the root CA certificate, and then specify the CA certificate chain of the server certificate. For more information on how to construct the server certificate chain, see the **`cachain`** command under "SSL Server SSL Configuration" in the *Command Reference* .

Configure Connection Pooling and Load Balancing Parameters

1. Change the "type" of the virtual SSL server to HTTP.

   Changing the server type to HTTP is necessary in order to enable the pooling of SSL connections.

```
>> SSL Settings#../type

Current value: generic
Type (generic/http): http
```

2. Disable transparent proxy mode.

   This step is required in order to make use of the connection pooling capabilities of the VPN Gateway.

```
>> Server 1#proxy

Current value: on
Proxy mode (on/off) :off
```

3. Enable pooling of TCP connections between the AVG and the real Web servers.

   Enabling pooling of TCP connections will increase performance.

```
>> Server 1#adv/pool

>> Pool Settings#ena
```

4. Enable load balancing of real Web servers.

```
>> Pool Settings#../loadbalanc

>> Load Balancing Settings#ena
```

5. Specify the IP address and port number for each real Web server to load balance.

   This step adds the two real servers (also called backend servers) to the load balancing configuration. The backend servers are identified by their respective IP address. The port setting indicates the TCP port number the virtual SSL server uses when initiating requests towards the backend servers.

```
>> Load Balancing Settings#backend

Enter backend server number: (1-)1

>> Backend Server 1#ip

Current value: 0.0.0.0
Enter IP address: 10.20.10.2

>> Backend Server 1#port

Current value: 0
Enter backend port number: 443

>> Backend Server 1#../backend 2

>> Backend Server 2#ip

Current value: 0.0.0.0
Enter IP address: 10.20.10.3

>> Backend Server 2#port

Current value: 0
Enter backend port number: 443
```

Note:

The backend servers must be configured to listen to the same TCP port as the one the configured. Since the backend servers must be SSL-enabled for end to

end encryption purposes, they will typically listen to traffic on port 443 by default.

6. Specify the health check type to use for the load balanced real Web servers.

```
>> Backend Server 2#../health

Current value: auto
Enter health check type (none/tcp/ssl/auto/script):
```

The default health check type is `auto`, which uses a built-in script. If you want to use one of the other alternatives, see the "Load Balancing Settings" section in the *Command Reference* for more information.

If you select `script` as the health check method, you must first define a customized script by using the script editing functions in the Health Check Script menu. For more information on how to use the options available in the Health Check Script menu, see Customized Health Check Scripts on page 161 .

Configure SSL Connect and Certificate Settings

1. Enable the use of SSL for encrypting the information sent between the AVG and the backend servers.

```
>> Load Balancing Settings#../sslconnect

>> SSL Connect Settings#ena
```

2. If needed, specify a client certificate for the AVG to present to the backend servers.

```
>> SSL Connect Settings#cert

Enter certificate number: (1-)
```

Note:

Normally, you don't have to specify a client certificate. However, if the SSL software running on the Web server is configured to require a client certificate, you must specify a client certificate that the virtual SSL server on the VPN Gateway can present to the Web server. If you generate a client certificate using the CLI on the VPN Gateway, you must verify that the CA certificate you used for generating the client certificate also is added to the list of CA certificates on the Web server.

3. Specify the certificate verification level for backend server authentication.

This step will ensure that the backend server always presents its server certificate to the virtual SSL server for authentication.

```
>> SSL Connect Settings#verify

>> SSL Connect Verify Settings#verify

Current value: none
Certificate verification (none/require): require
```

4. Specify the common name found in the backend server certificate.

   By specifying the common name of the backend server (the common name typically equals the fully qualified domain name of the Web server, and is stated in the backend server certificate), the virtual SSL server will perform a check to see if the common name you specify corresponds with common name in the backend server certificate presented to the VPN Gateway each time an SSL connection is established.

```
>> SSL Connect Verify Settings#commonname

Current value :""

Give common name of server:<common name as stated in the backend
server certificate>
```

Note:

If you generated a server certificate using the CLI on the VPN Gateway and installed that certificate on the backend server, you specify the same common name as you did when generating the server certificate.

5. Specify the accepted signer of the backend server certificate.

```
>> SSL Connect Verify Settings#cacerts

Current value :""

Enter certificate numbers (separated by comma):<index number
representing the CA certificate>
```

Typically, all major client Web browsers are preloaded with the certificates of common CAs. Client Web browsers use the preloaded CA certificates to verify that the certificate submitted by the Web server is valid and issued by a trusted CA. If the client browser does not recognize the CA that signed and issued the Web server's certificate, a warning is displayed.

However, in this case it is the virtual SSL server on the VPN Gateway that acts as a client towards the backend server, and you must therefore explicitly specify which

certificate authorities are valid signers of the backend servers' certificate. The CA certificate you specify must be available on the VPN Gateway.

Note:

If you generated a server certificate using the CLI on the VPN Gateway and installed that certificate on the backend server, specify the index number of the CA certificate you used for generating the server certificate as the accepted signer.

Also, if you signed a CSR using the CLI on the VPN Gateway and installed the signed CSR as a server certificate on the backend server, specify the index number of the CA certificate you used for signing the server CSR.

6. Apply your settings.

```
>> SSL Connect Verify Settings#apply

Changes applied successfully.
```

Configure the Application Switch

Create the Necessary VLANs

In this configuration, there will be three VLANs: VLAN 1 for the Application Switch port that connects to the Internet, VLAN 2 for ports connecting to the VPN Gateways, and VLAN 3 for ports connecting to the real servers. Since VLAN 1 is the default, only VLAN 2 and VLAN 3 require additional configuration.

1. Configure VLAN 2 to include Application Switch ports leading to the VPN Gateways.

```
# /cfg/vlan 2

>> VLAN 2#add 2

Port 2 is an UNTAGGED port and its current PVID is 1.
Confirm changing PVID from 1 to 2 [y/n]: y

>> VLAN 2#add 3

Port 3 is an UNTAGGED port and its current PVID is 1.
Confirm changing PVID from 1 to 2 [y/n]: y

>> VLAN 2#ena
```

2. Configure VLAN 3 to include Application Switch ports leading to the real servers.

```
# /cfg/vlan 3

>> VLAN 3#add 7

Port 7 is an UNTAGGED port and its current PVID is 1.
Confirm changing PVID from 1 to 2 [y/n]: y

>> VLAN 3#add 8

Port 8 is an UNTAGGED port and its current PVID is 1.
Confirm changing PVID from 1 to 2 [y/n]: y

>> VLAN 3#ena
```

3. Disable Spanning Tree Protocol (STP) for the AVG ports 2 and 3 and real server ports 7 and 8.

```
# /cfg/stp 1/port 2

>> Spanning Tree Port 2#off

>> Spanning Tree Port 2#../port 3

>> Spanning Tree Port 3#off

>> Spanning Tree Port 3#../port 7

>> Spanning Tree Port 7#off

>> Spanning Tree Port 7#../port 8

>> Spanning Tree Port 8#off
```

Configure One IP Interface for Each VLAN

Note:

If you prefer, you can reverse the order of the first two commands (**`addr`** and **`mask`**) in the following example. By entering the mask first, the Application Switch will automatically calculate the correct broadcast address for you. The calculated broadcast address is displayed immediately after you provide the IP address of the interface, and will be applied together with the other configuration settings when you execute the **`apply`** command.

1. Configure an IP interface for client traffic on the Application Switch with VLAN 1.

```
# /cfg/ip/if 1

>> IP Interface 1#addr 192.168.10.1

>> IP Interface 1#mask 255.255.255.0
```

```
>> IP Interface 1#broad 192.168.10.255
>> IP Interface 1#vlan 1
>> IP Interface 1#ena
```

2. Configure an IP interface for AVG traffic with VLAN 2.

```
# /cfg/ip/if 2
>> IP Interface 2#addr 172.16.10.1
>> IP Interface 2#mask 255.255.0.0
>> IP Interface 2#broad 172.16.255.255
>> IP Interface 2#vlan 2
>> IP Interface 2#ena
```

3. Configure an IP interface for the real server traffic with VLAN 3.

```
# /cfg/ip/if 3
>> IP Interface 3#addr 10.20.10.1
>> IP Interface 3#mask 255.255.255.0
>> IP Interface 3#broad 10.20.10.255
>> IP Interface 3#vlan 3
>> IP Interface 3#ena
```

4. Apply the changes.

```
# apply
```

Note:

Make sure the VPN Gateways are configured to use the IP address of IP interface 2 on VLAN 2 as their default gateway. For more information about gateway configuration, see the **`gateway`** command under "System Configuration" in the *Command Reference*. Likewise, the Web servers must be configured to use the IP address of IP interface 3 on VLAN 3 as their default gateway.

Configure Web Server Load Balancing Parameters

Note:

Defining the real servers and a virtual IP address (VIP) on the Application Switch is only necessary if the switch should be able to reply to ARP requests for the VIP. Also, if the AVG cluster fails, the redirect filters will not be triggered and the configuration suggested will constitute a failover solution.

1. Set and enable the IP addresses of the real Web servers.

```
# /cfg/slb/real 1
>> Real Server 1#rip 10.20.10.2
>> Real Server 1#ena
>> Real Server 1#../real 2
>> Real Server 2#rip 10.20.10.3
>> Real Server 2#ena
```

2. Add real Web servers 1 and 2 to real server group 1.

```
# /cfg/slb/group 1
>> Real server group 1#add 1
>> Real server group 1#add 2
```

3. Set the load balancing metric and health check type for real server group 1.

```
# /cfg/slb/group 1
>> Real server group 1#metric hash
>> Real server group 1#health sslh
```

4. Enable client processing on port 1 leading to the Internet.

```
# /cfg/slb/port 1/client ena
```

5. Enable Direct Access Mode (DAM).

```
# /cfg/slb/adv/direct ena
```

6. Turn on Layer 4 processing.

```
# /cfg/slb/on
```

Configure AVG Load Balancing Parameters

Set and enable the IP addresses of the VPN Gateways, and create a group in the switch for load balancing.

1. For each VPN Gateway, create a Real Server IP address on the Application Switch.

```
# /cfg/slb/real 3

>> Real server 3#rip 172.16.10.2

>> Real server 3#ena

>> Real server 3#../real 4

>> Real server 4#rip 172.16.10.3

>> Real server 4#ena
```

2. Create a Real Server Group and add the Real Servers (the VPN Gateways in this case).

```
# /cfg/slb/group 2

>> Real server group 2#add 3

>> Real server group 2#add 4
```

3. Set the load balancing metric and health check type for real server group 2.

```
# /cfg/slb/group 2

>> Real server group 2#metric hash

>> Real server group 2#health sslh
```

4. Set and enable the IP address for Virtual Server 1, enable load balancing through Virtual Server 1 of server group 1 for the HTTPS service.

   Enable service on port 443 for HTTPS communication between the VPN Gateways and the real Web servers. Recall that the real Web servers must also be SSL-enabled and listen for AVG traffic on port 443. This step also connects the real Web servers in group 1 to the enabled Virtual Server 1.

```
# /cfg/slb/virt 1
```

```
>> Virtual Server 1#vip 192.168.10.100
>> Virtual Server 1#ena
>> Virtual Server 1#service https
```

(service `https` assumes use of TCP port 443)

```
>> Virtual Server 1 http Service#group 1
>> Virtual Server 1 https Service#apply
```

5. Apply the changes.

```
# apply
```

Configure Filters

1. Create a filter to redirect client HTTPS traffic intended for port 443.

   When this filter is added to the switch port leading to the Internet, incoming HTTPS traffic is redirected to the AVGs in real server group 2.

   The HTTPS traffic filter should be given a high number (a lower priority), such as 100, so as not to interfere with other filters.

```
# /cfg/slb/filt 100
>> Filter 100#proto tcp
>> Filter 100#dport https
>> Filter 100#action redir
>> Filter 100#group 2
>> Filter 100#rport https
>> Filter 100#ena
```

2. Create a default filter to allow all other traffic.

```
# /cfg/slb/filt 224
>> Filter 224#sip any
>> Filter 224#dip any
>> Filter 224#proto any
```

```
>> Filter 224#action allow

>> Filter 224#ena
```

3. Add the client filters to the client port leading to the Internet.

   This step adds the HTTPS redirect filter and the default allow filter to the client port leading to the Internet.

```
# /cfg/slb/port 1

>> SLB Port 1#add 100

>> SLB Port 1#add 224

>> SLB Port 1#filt ena
```

Apply and Save the Configuration

Apply and save the configuration changes.

```
# apply
# save
```

Verifying End to End Encryption

Using an OpenSSL client

1. Configure the AVG to add an extra X-ISD header to the client request.

```
# /cfg/ssl/server 1/http

>> HTTP Settings#addxisd on

>> HTTP Settings#apply
```

2. From the client side of the network, use an OpenSSL client and open a connection to the IP address of Virtual Server 1 on the Application Switch. Type the following commands.

```
openssl s_client -connect 192.168.10.100 TRACE / HTTP/
1.0
```

3. In the response, look for the **X-ISD** header. The output should resemble the following example.

   Note that the `type=https-https` entry that appears at the end of the X-ISD header shows `https` both for the incoming client request, and for the request that VPN Gateway #1 (with IP address 172.16.10.2) initiated to backend server #2 (with IP address 10.20.10.3). This entry verifies that end to end encryption has been performed.

```
Host: 192.168.10.100
Via: 1.0 192.168.10.100 Alteon iSD-SSL/6.0
X-ISD: 172.16.10.2 10.20.10.3 index=1; pool=on; lb=all-
roundrobin; type=https-https
X-SSL: decrypted=true, ciphers="TLSv1/SSLv3 DES-CBC3-SHA"
```

Using a Secure Telnet client

1. From the client side of the network, use a TelnetS (Secure Telnet) client and open a connection to the IP address of Virtual Server 1 on the Application Switch.

   - 192.168.10.100

2. Type the following command in your TelnetS client.

```
GET / HTTP/1.0
```

3. In the response from the backend server, look for the `Server` header. The output should resemble the following example.

```
HTTP/1.1 200 OK
Date: Thu, 15 Dec 2006 10:40:06 GMT
Server: Apache/1.3.20 (Unix) mod_ssl/2.8.4 OpenSSL/0.9.6
Content-Type: message/http
Connection: close
Via: 1.1 192.168.10.100 Alteon iSD-SSL/6.0
```

   The `Server` header contains information about the SSL software running on the backend server, which enables the VPN Gateway to transmit encrypted requests.

Chapter 6: Web Server Accelerator with Multiple Networks in Cluster

This chapter provides the configuration details for creating a Web server accelerator solution with the following characteristics:

- Fully redundant system with four Application Switches running in active-standby mode and a cluster of two Avaya VPN Gateways (AVGs). There is no single point-of-failure.
- Encrypted traffic that is separated from non-encrypted traffic and management traffic by configuring two different IP networks on the VPN Gateways.
- Higher level of security for the AVG cluster, since management access is only allowed through the internal trusted IP network.

Note:

The iSD100-SSL model cannot be used in this configuration, since that model is equipped with only one port NIC. You must use either the ASA 310 Copper NIC model (equipped with two integrated Intel PRO/100+ port NICs), or the ASA 310 Fiber NIC model (equipped with one 3Com Gigabit Ethernet fiber-optic port NIC, in addition to the dual integrated Intel PRO/100+ NICs).

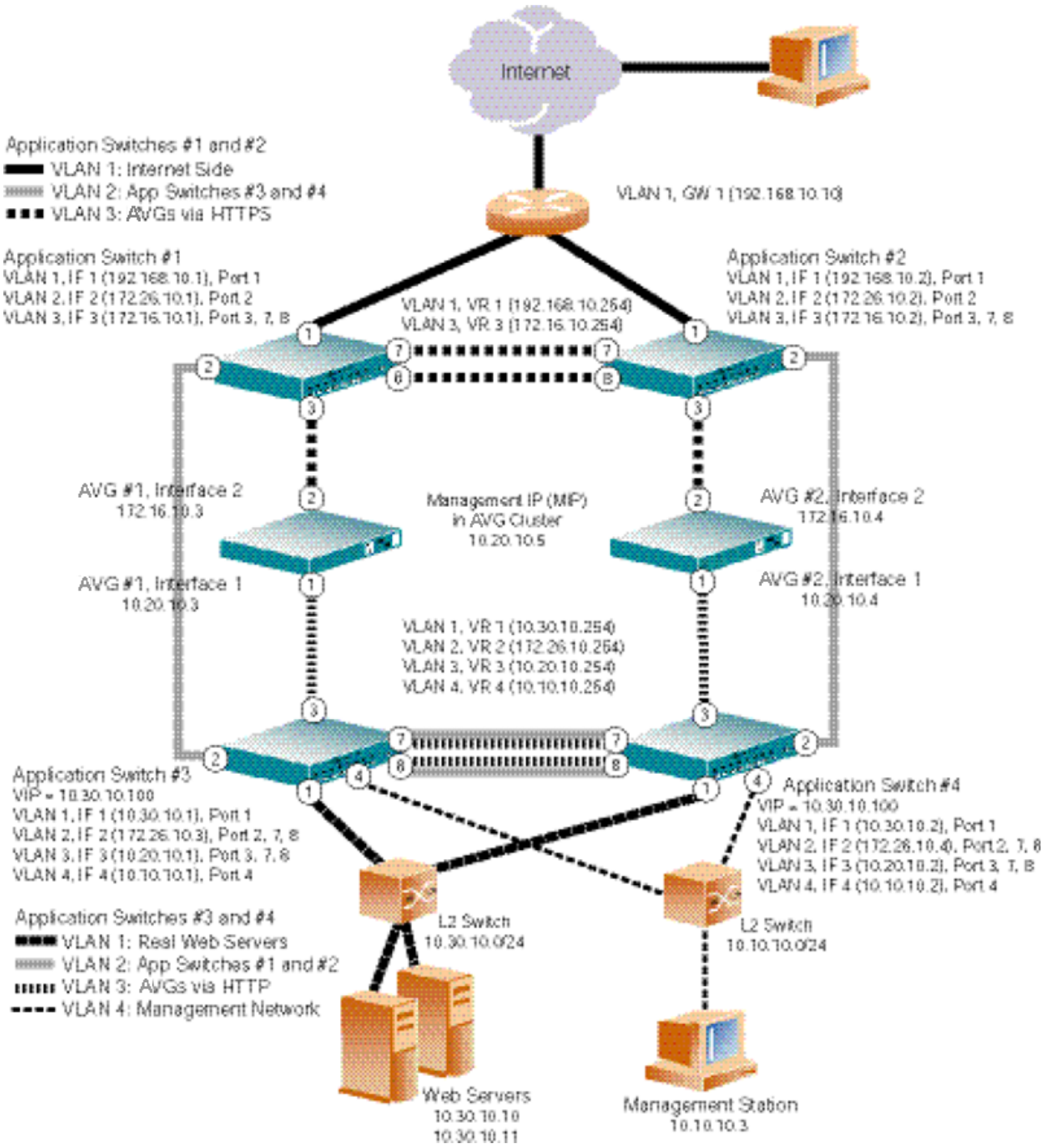

Figure 9: Sample Web Server Accelerator Network Using Application Switches in Active-Standby Mode and a Cluster of AVGs

Functional Description

AVG and Application Switch

The two VPN Gateways in Figure 9: Sample Web Server Accelerator Network Using Application Switches in Active-Standby Mode and a Cluster of AVGs on page 82 are both connected to a pair of Application Switches: Application Switch 1 and 2 connect to the Internet, whereas Application Switch 3 and 4 connect to the real Web servers and the management network.

Application Switches 1 and 2 are configured to perform load balancing of the VPN Gateways, to which all HTTPS client connection requests are redirected through a filter on Application Switch port 1. Application Switches 3 and 4, on the other hand, are configured to perform load balancing of the real Web servers which receive both regular HTTP requests from Internet clients, and requests from the VPN Gateways for content that would have been encrypted had not the AVGs served as SSL offload devices. All other traffic coming from the Internet is blocked, using a combination of filters on switch port 1 on the Internet side Application Switches.

Application Switches 1 and 2 will send regular HTTP traffic over a direct (crossover) link to the virtual server configured on Application Switches 3 and 4, which in turn forwards the traffic to the load balanced real Web servers.

Application Switches 3 and 4 are configured to use the Return To Sender (RTS) feature, which ensures that a Web server response is always sent back through the same VPN Gateway that forwarded the client request (and terminated the SSL session in the process).

Optionally, a firewall or reversed proxy server could be used to forward the regular HTTP traffic from the Internet side Application Switches to the real Web servers (through Application Switches 3 and 4). However, this requires a different configuration and is beyond the scope of this example.

Health Checks

Health checking is critical in any redundant solution. In this configuration, Application Switches 1 and 2 will perform health checks of Application Switches 3 and 4, respectively, and the health checks are forwarded through two HTTP proxy servers configured to run in non-transparent proxy mode in the AVG cluster. Since each VPN Gateway has a link to both an Application Switch on the Internet side and an Application Switch on the Web server side simultaneously, the health checking of Application Switches 3 and 4 includes verifying that both of these links are active.

The HTTP proxy servers on the VPN Gateways are used only for transmitting health checks to the HTTP responders that are enabled on Application Switches 3 and 4. The "health check" proxy servers are in effect regular virtual SSL servers, but where SSL and transparent proxy mode have been disabled.

Each VPN Gateway is configured with a health check proxy server that:

- Listens to the physical IP address as configured for one VPN Gateway on Interface 2, which connects to an Application Switch on the Internet side. This is the VIP configuration setting.
- Initiates requests to one Application Switch on the server side, using the Application Switch 's IP interface address that connects to VLAN 3. This is the RIP configuration setting.
- Uses port 80 when listening to and initiating HTTP requests. These are the PORT and RPORT configuration settings.

Thus, when one of the Internet side Application Switches initiates a HTTP health check, the health check is transmitted to one of the AVG proxy servers. The proxy server forwards the HTTP request to its server side Application Switch. The HTTP responder that is enabled on the Application Switch sends a "HEALTH CHECK OK" reply back to the initiating Internet side Application Switch, using the return path through the same VPN Gateway (achieved by configuring the proxy servers to run in non-transparent mode).

Trunked Interswitch Connection

The connection between each pair of Application Switches (over port 7 and 8) plays a key role in this network solution, especially when it comes to the connection between Application Switches 3 and 4. Because unexpected routing behavior will result from a failed connection, you are strongly recommended to use port trunking on port 7 and 8 between each pair of Application Switches.

Configuring Multiple Networks

Initial Setup of AVGs

This configuration example assumes that you have performed the initial setup of the two VPN Gateways as described in the "Initial Setup" chapter in the *Users Guide* . The example also assumes that you have added a server certificate as described in Add a Server Certificate to the AVG on page 22.

The default Interface 1 will be configured during the initial setup. Interface 1 connects to VLAN 3 as defined on Application Switches 3 and 4. Later on in this configuration example, you will

configure an additional interface within the cluster; Interface 2 which connects to VLAN 3 as defined on Application Switches 1 and 2.

Note:

When prompted for a gateway IP address during the initial setup, leave blank to skip. In this configuration example, you will configure a default gateway after having performed the initial setup.

Configure the AVGs

Log in as the administrator to the AVG cluster by using a Telnet or SSH connection to the Management IP address (MIP). The configuration changes will automatically be propagated to all VPN Gateways in the cluster.

Create and Configure a Virtual SSL Server

1. Create a virtual SSL server.

   This step creates a new virtual SSL server on the VPN Gateway. Each virtual SSL server listens to a specific TCP port and is connected to a Virtual Server IP address on the Application Switch.

```
# /cfg/ssl/server

Enter virtual server number: (1-)1

Creating new server 1
>> Server 1#
```

2. Define a name for virtual SSL server 1.

   This step lets you specify a name, by which you can identify SSL server 1. To view the numbers and related names of all configured SSL servers, use the `/cfg/ssl/cur` command. The name you specify is mainly intended for your own reference, and is not critical for the configuration itself. As the following example suggests, the name can indicate the service for which the SSL server was created.

```
>> Server 1#name

Current value:""

Enter new server name:HTTPS
```

3. Set listen TCP port for SSL server 1.

   Each time you create a new SSL server, the listen port is automatically set to 443. Since you are setting up the VPN Gateway for HTTPS offload purposes in this

example, it is not really necessary to configure the SSL server to listen port to 443. However, for using the VPN Gateway for any protocol other than HTTPS, a new virtual SSL server must be configured to listen to the TCP port of the intended service.

```
>> Server 1#port

>> Current value: 443 (https)
>> Enter listen port number (1-65534):443
```

4. Connect the SSL server to the desired Virtual Server IP address on Application Switch 3.

   This step connects SSL Server 1 to the IP address of the virtual server in VLAN 1, as defined on Application Switch 3.

```
>> Server 1#vips

Current value:""

Enter server ips (comma separated):10.30.10.100
```

5. Set the Real Server IP address to which SSL Server 1 should connect when initiating requests.

   Preserve the current value of the Real Server IP address, which should be 0.0.0.0. At first glance this configuration may perhaps seem odd. However, by specifying 0.0.0.0 as the Real Server IP address, the SSL server is instructed to use the destination IP address (in the received packets) when initiating requests sent to the virtual server. Since the destination IP address in the received packets corresponds to the IP address of the virtual server, the requests will always reach the correct Virtual Server IP address.

```
>> Server 1#rip

Current value: 0.0.0.0
Enter IP address to connect to:0.0.0.0
```

6. Set the server port to which SSL server 1 should connect when initiating requests.

   This step sets the TCP port, to which virtual SSL Server 1 connects when initiating requests.

```
>> Server 1#rport

Current value: 0 [81]
Enter port to connect to:81
```

Note:

If you have not configured any port mappings for the real Web servers on the Application Switch (like in this example), the real Web servers must also be set to listen for AVG traffic on port 81.

7. Specify the certificate to be used by SSL Server 1.

   You are prompted to type the index number of an existing certificate. To view all certificates currently added to the AVG by index number and name, use the `/info/certs` command. For more information on how to add a certificate to the AVG, see the "Certificates and Client Authentication" chapter in the *Users Guide* .

```
>> Server 1#ssl

>> SSL Settings#cert

Current value: <not set>
Enter certificate number: (1-)1
```

Note:

If the certificate you specify is a chained certificate, you need to first add the CA certificates up to and including the root CA certificate, and then specify the CA certificate chain of the server certificate. For more information on how to construct the server certificate chain, see the **`cachain`** command under "SSL Server SSL Configuration" in the *Command Reference*.

8. Apply the configuration changes.

```
>> SSL Settings#apply

Changes applied successfully.
```

Create and Configure two Extra Virtual SSL Servers

These two virtual SSL servers are necessary in order to let Application Switches 1 and 2 perform health checks through the VPN Gateways, where virtual SSL server 2 and 3 merely act as HTTP proxy servers. They are not used for SSL and only serve the purpose of ensuring that Application Switches 1 or 2 will never forward a client HTTPS connection request to a VPN Gateway which does not respond positively to the health check.

1. Create virtual SSL server 2.

```
# /cfg/ssl/server 2
```

```
Creating new server 2
>> Server 2#
```

2. Define a name for virtual SSL server 2.

```
>> Server 2#name

Current value:""

Enter new server name:HealthCheck_1
```

3. Set listen TCP port for SSL server 2.

   Each time you create a new SSL server, the listen port is automatically set to 443. Because you are setting up the AVG for HTTP health check purposes in this step, the listen port is changed to port number 80 (HTTP).

```
>> Server 2#port

>> Current value: 443 (https)
>> Enter listen port number (1-65534):80
```

4. Connect the SSL server to a Virtual Server IP address.

```
>> Server 2#vips

Current value:""

Enter server ips (comma separated):172.16.10.3
```

5. Set the Real Server IP address to which SSL Server 2 should connect when initiating requests.

   The RIP value corresponds to IP interface 3 on Application Switch 3.

```
>> Server 2#rip

Current value: 0.0.0.0
Enter IP address to connect to:10.20.10.1
```

6. Set the server port to which SSL server 2 should connect when initiating requests.

   This step sets the TCP port, to which SSL Server 2 connects when forwarding health check requests.

```
>> Server 2#rport
```

```
Current value: 0 [81]
Enter port to connect to (0-65534):80
```

7. Turn off transparent proxy mode.

   When transparent proxy mode is set to `off`, the VPN Gateway will forward the health check reply to Application Switches 1 and 2. If the AVG fails to forward the health check reply, Application Switches 1 and 2 will send client requests to the other VPN Gateway in the cluster instead.

```
>> Server 2#proxy off
```

8. Disable SSL for virtual SSL server 2.

   Because the virtual SSL server only will act as a proxy HTTP server and not use a certificate, SSL is disabled.

```
>> Server 2#ssl

>> SSL Settings#dis
```

9. Apply the configuration changes.

```
>> SSL Settings#apply

Changes applied successfully.
```

10. Create virtual SSL server 3.

```
# /cfg/ssl/server 3

Creating new server 3
>> Server 3#
```

11. Define a name for virtual SSL server 3.

```
>> Server 3#name

Current value:""

Enter new server name:HealthCheck_2
```

12. Set listen TCP port for SSL server 3.

    Each time you create a new SSL server, the listen port is automatically set to 443. Because you are setting up the AVG for HTTP health check purposes in this step, the listen port is changed to port number 80 (HTTP).

```
>> Server 3#port

>> Current value: 443 (https)
>> Enter listen port number (1-65534):80
```

13. Connect the SSL server to a Virtual Server IP address.

```
>> Server 3#vips

Current value:""

Enter server ips (comma separated):172.16.10.4
```

14. Set the Real Server IP address to which SSL Server 3 should connect when initiating requests.

    The RIP value corresponds to IP interface 3 on Application Switch 4.

```
>> Server 3#rip

Current value: 0.0.0.0
Enter IP address to connect to:10.20.10.2
```

15. Set the server port to which SSL server 3 should connect when initiating requests.

    This step sets the TCP port, to which SSL Server 3 connects when initiating requests.

```
>> Server 3#rport

Current value: 0 [81]
Enter port to connect to (0-65534):80
```

16. Turn off transparent proxy mode.

```
>> Server 3#proxy off
```

17. Disable SSL for virtual SSL server 3.

    Because the virtual SSL server only will act as a proxy HTTP server and not use a certificate, SSL is disabled.

```
>> Server 3#ssl
```

```
>> SSL Settings#dis
```

18. Apply the configuration changes.

```
>> SSL Settings#apply

Changes applied successfully.
```

Specify a Default Gateway and Configure Static Routes

1. Specify the default gateway, which in this example corresponds to Virtual Router 3 in VLAN 3, as defined on Application Switch 1

   The default gateway should be specified for both VPN Gateways in the cluster.

```
>> Server 3#/cfg/sys/host 1/gateway

Current value: 0.0.0.0
Enter default gateway address:172.16.10.254

>> iSD Host 1#../host 2/gateway

Current value: 0.0.0.0
Enter default gateway address:172.16.10.254
```

2. Static routes must be configured to the management network in VLAN 4, and to the real Web servers network in VLAN 1.

   The gateway address specified for both the management network (IP address 10.10.10.0) and the real Web servers network (IP address 10.30.10.0) corresponds to Virtual Router 3 in VLAN 3, as defined on Application Switches 3 and 4.

```
>> System#routes/add

Enter destination address:10.10.10.0

Enter network mask:255.255.255.0

Enter gateway address:10.20.10.254

>> Routes#add

Enter destination address:10.30.10.0

Enter network mask:255.255.255.0

Enter gateway address:10.20.10.254
```

Configure Interface 2 for Both AVGs

Currently, both VPN Gateways are configured with Interface 1 per host. Port number 1 is normally assigned to Interface 1 by default (during the initial setup).

Configure Interface 2 on both VPN Gateways to connect them to Application Switches 1 and 2 in VLAN 3. The AVGs will receive HTTPS client connection requests through Application Switches 1 and 2 on Interface 2.

1. Specify the parameters for Interface 2 on AVG host 1 and add a physical port to be used in Interface 2.

   This step configures the parameters for Interface 2 on AVG host 1, and enables the AVG to use port number 2 on Interface 2.

   Note that the network mask can be entered in number of bits, for example, 24 instead of 255.255.255.0.

```
>> Routes#/cfg/sys/host 1/interface 2

Creating Host Interface 2
Enter new IP address of the Interface:172.16.10.3

Enter the network mask:24

Enter VLAN tag id [0]:<press ENTER to skip>

Entering: Interface ports menu
Port to add:2

Leaving: Interface ports menu
```

2. Specify the parameters for Interface 2 on AVG host 2 and add a physical port to be used in Interface 2.

   This step configures the parameters for Interface 2 on AVG host 2, and enables the AVG to use port number 2 on Interface 2.

```
>> Host Interface 2#../../host 2/interface 2

Creating Host Interface 2
Enter new IP address of the Interface:172.16.10.4

Enter the network mask:24

Enter VLAN tag id [0]:<press ENTER to skip>

Entering: Interface ports menu
Port to add:2

Leaving: Interface ports menu
```

Restrict Access to the Management Network

1. Add the network address of the management network to the access list.

   This step assumes that you have already enabled Telnet (or SSH) access to the AVG cluster. By adding a machine address or network address to the access list, remote access to the AVG cluster is only granted the specified machine or range of machines. In this example, all management stations on network 10.10.10.0 will be granted access to the cluster.

   For more information about enabling remote access, see the "The Command Line Interface" chapter in your *Command Reference* .

   ```
   >> Host Interface 2#/cfg/sys/accesslist

   >> Access List#add

   Enter network address:10.10.10.0

   Enter netmask:255.255.255.0
   ```

   Note:

   If you add an entry to the access list, you must also add the Management IP address (MIP) and the Interface 1 IP addresses of the existing VPN Gateways in the cluster to the access list. Otherwise the AVGs will not be able to communicate.

2. Add the MIP and Interface 1 addresses of both VPN Gateways to the access list.

   The network address specified in the following example will include the MIP and Interface 1 addresses.

   ```
   >> Access List#add

   Enter network address:10.20.10.0

   Enter netmask:255.255.255.0
   ```

3. Apply your configuration changes.

   ```
   >> Access List#apply

   Changes applied successfully.
   ```

Configure Application Switch 1

Create the Necessary VLANs

In the configuration on Application Switches 1 and 2, there will be three VLANs: VLAN 1 that connects to the Internet, VLAN 2 for connections to Application Switches 3 and 4, and VLAN 3 for the connection to the VPN Gateways. Since VLAN 1 is the default, only VLAN 2 and VLAN 3 require additional configuration.

1. Configure VLAN 2 to include Application Switch port leading to Application Switches 3 and 4.

```
#/cfg/vlan 2

>> VLAN 2#add 2

Port 2 is an UNTAGGED port and its current PVID is 1.
Confirm changing PVID from 1 to 2 [y/n]:y

>> VLAN 2#ena
```

2. Configure VLAN 3 to include the Application Switch port leading to the VPN Gateways, and the trunking ports leading to Application Switch 2.

```
#/cfg/vlan 3

>> VLAN 3#add 3
```

Port 3 is an UNTAGGED port and its current PVID is 1. Confirm changing PVID from 1 to 3 [y/n]: **y**

```
>> VLAN 3#add 7

Port 7 is an UNTAGGED port and its current PVID is 1.
Confirm changing PVID from 1 to 3 [y/n]:y

>> VLAN 3#add 8

Port 8 is an UNTAGGED port and its current PVID is 1.
Confirm changing PVID from 1 to 3 [y/n]:y

>> VLAN 3#ena
```

Disable STP and Create a Port Trunk

1. Disable Spanning Tree Protocol (STP) globally.

```
#/cfg/stp 1

>> Spanning Tree#off
```

2. Enable trunking between Application Switch 1 and Application Switch 2 on port 7 and port 8.

```
#/cfg/trunk 1

>> Trunk group 1#add 7

Port 7 added.
>> Trunk group 1#add 8

Port 8 added.
>> Trunk group 1#ena
```

Configure One IP Interface for Each VLAN

When you configure the Application Switch IP interfaces, you can let the Application Switch automatically calculate the correct broadcast address for you. To do this, enter the subnet mask first, and then the address of the IP interface. Immediately after you provide the address of the IP interface, the calculated broadcast address will be displayed.

1. Configure an IP interface for client traffic on Application Switch 1 with VLAN 1.

```
#/cfg/ip/if 1

>> IP Interface 1#addr 192.168.10.1

>> IP Interface 1#mask 255.255.255.0

>> IP Interface 1#broad 192.168.10.255

>> IP Interface 1#vlan 1

>> IP Interface 1#ena
```

2. Configure an IP interface for the Application Switch traffic with VLAN 2.

```
#/cfg/ip/if 2
```

```
>> IP Interface 2#addr 172.26.10.1
>> IP Interface 2#mask 255.255.255.0
>> IP Interface 2#broad 172.26.10.255
>> IP Interface 2#vlan 2
>> IP Interface 2#ena
```

3. Configure an IP interface for the AVG traffic with VLAN 3.

```
#/cfg/ip/if 3
>> IP Interface 3#addr 172.16.10.1
>> IP Interface 3#mask 255.255.255.0
>> IP Interface 3#broad 172.16.10.255
>> IP Interface 3#vlan 3
>> IP Interface 3#ena
```

Configure the AVG Load Balancing Parameters

Set and enable the IP addresses of the virtual SSL servers on the AVG, and create a group in the switch for health checking.

Note:

The AVG load balancing configuration on Application Switch 1 is only intended to enable Application Switches 1 and 2 to perform health checks through the VPN Gateways. The real load balancing of the AVGs will be configured on Application Switch 3.

1. Configure the Application Switch to use server load balancing globally.

```
#/cfg/slb>> Layer 4#on
```

2. Create a Real Server IP address for each VPN Gateway on the Application Switch.

```
#/cfg/slb/real 1
(Real 1 corresponds to virtual SSL server 2 on SSL VPN)
>> Real server 1#name SSL-VPN_1
```

```
>> Real server 1#rip 172.16.10.3

>> Real server 1#ena

>> Real server 1#../real 2

(Real 2 corresponds to virtual SSL server 3 on SSL VPN)

>> Real server 2#name SSL-VPN_2

>> Real server 2#rip 172.16.10.4

>> Real server 2#ena
```

Real server 1 corresponds to virtual SSL server 2 on the AVG, whereas real server 2 corresponds to virtual SSL server 3 on the AVG. These real servers are only configured in order to let Application Switches 1 and 2 send health checks through the VPN Gateways. The virtual SSL servers to which the real servers are mapped only function as HTTP proxy servers that forward health checks to Application Switches 3 and 4.

3. Create a Real Server Group and add the AVGs.

```
#/cfg/slb/group 1

>> Real server group 1#name SSL_Accelerators

>> Real server group 1#add 1

>> Real server group 1#add 2
```

4. Set the load balancing metric and health check type for Real Server Group 1.

```
#/cfg/slb/group 1

>> Real server group 1#metric hash

>> Real server group 1#health script1
```

5. Define the health check script.

```
#/cfg/slb/adv/script 1

>> Health Script 1#open 80

>> Health Script 1#send "GET /health.html HTTP/1.0\\n\\n"

>> Health Script 1#expect "HEALTH CHECK OK"
```

```
>> Health Script 1#close
```

The health check script will be sent to the built-in HTTP servers on Application Switches 3 and 4, through the two HTTP proxy servers on each VPN Gateway. The health.html page is always available on the Application Switch, provided the HTTP responder is enabled (this will be done later on in this configuration example). The health checking provides the means for Application Switches 1 and 2 to detect if the physical network interfaces on the VPN Gateways are functional, and if the links between the AVGs and Application Switches 3 and 4 are active. This prevents Application Switches 1 and 2 from forwarding client requests to a VPN Gateway that has a faulty NIC, or where the link between a VPN Gateway and an Application Switch (on the remote side of a VPN Gateway) is down.

Note:

The health.html page is predefined and cannot be modified. The page only servers the purpose of allowing health checks, either through a script (as in this configuration example) or through the Web OS BBI (Browser Based Interface).

Specify a Default Gateway and Add a Static Route

1. Specify the default gateway for Application Switch 1.

   This step specifies the default gateway in VLAN 1 connecting to the Internet side.

```
#/cfg/ip/gw 1

>> Default gateway 1#addr 192.168.10.10
```

2. Add the required static route.

   The static route is required for client HTTPS requests to reach the real Web servers through the VPN Gateways. The gateway IP address for this static route corresponds to virtual router 2 in VLAN 2, as defined on Application Switches 3 and 4.

```
#/cfg/ip/route

>> IP Static Route#add

Enter destination IP address:10.30.10.0

Enter destination subnet mask:255.255.255.0

Enter gateway IP address:172.26.10.254
```

```
Enter interface number: (1-256)2
```

3. Specify a secondary gateway for Application Switch 1.

   If link 1 on Application Switch 1 in Figure 9: Sample Web Server Accelerator Network Using Application Switches in Active-Standby Mode and a Cluster of AVGs on page 82 fails, non-https traffic will not be routed back to the Internet unless a secondary gateway, pointing to Application Switch 2, is defined on Application Switch 1.

```
#/cfg/ip/gw 2

>> Default gateway 2#addr 172.16.10.2
```

Configure Filters

1. Create a filter to allow health check of the default gateway.

```
#/cfg/slb/filt 5

>> Filter 5#proto icmp

>> Filter 5#sip 192.168.10.10

>> Filter 5#smask 255.255.255.255

>> Filter 5#action allow

>> Filter 5#ena
```

2. Create a filter to redirect incoming client HTTPS traffic intended for port 443.

   When this filter is added to the switch port leading to the Internet, incoming HTTPS traffic is redirected to the VPN Gateways in real server group 1.

```
#/cfg/slb/filt 10

>> Filter 10#proto tcp

>> Filter 10#dport https

>> Filter 10#action redir

>> Filter 10#group 1

>> Filter 10#rport https
```

```
>> Filter 10#ena
```

3. Create a filter to allow HTTP traffic on the crossover links from Application Switches 1 and 2 to Application Switches 3 and 4 on VLAN 2.

```
#/cfg/slb/filt 20
>> Filter 20#proto tcp
>> Filter 20#dport http
>> Filter 20#action allow
>> Filter 20#ena
```

4. Create a filter to allow receiving VRRP updates on port 1.

   VRRP uses IP address 224.0.0.18 for updates by default.

```
#/cfg/slb/filt 30
>> Filter 30#dip 224.0.0.18
>> Filter 30#dmask 255.255.255.255
>> Filter 30#action allow
>> Filter 30#ena
```

5. Create a filter to disallow all other traffic.

```
#/cfg/slb/filt 40
>> Filter 40#action deny
>> Filter 40#ena
```

6. Add the filters to the client port leading to the Internet.

```
#/cfg/slb/port 1
>> SLB Port 1#add 5
>> SLB Port 1#add 10
>> SLB Port 1#add 20
>> SLB Port 1#add 30
```

```
>> SLB Port 1#add 40
>> SLB Port 1#filt ena
```

Configure VRRP

VRRP is configured for failover (redundancy) between two Application Switches, in the event one of the Application Switches fails.

1. Turn on VRRP.

```
#/cfg/vrrp/on
```

2. Configure virtual router 1.

```
#/cfg/vrrp/vr 1
>> VRRP Virtual Router 1#vrid 1
>> VRRP Virtual Router 1#addr 192.168.10.254
>> VRRP Virtual Router 1#if 1
>> VRRP Virtual Router 1#prio 101
>> VRRP Virtual Router 1#share dis
>> VRRP Virtual Router 1#track/ifs e
>> VRRP Virtual Router 1 Priority Tracking#../ena
```

3. Configure virtual router 3.

```
#/cfg/vrrp/vr 3
>> VRRP Virtual Router 3#vrid 3
>> VRRP Virtual Router 3#addr 172.16.10.254
>> VRRP Virtual Router 3#if 3
>> VRRP Virtual Router 3#prio 101
>> VRRP Virtual Router 3#share dis
>> VRRP Virtual Router 3#track/ifs e
>> VRRP Virtual Router 3 Priority Tracking#../ena
```

Note:

Make sure the VPN Gateways are configured to use the IP address of Virtual Router 3 on VLAN 3 as their default gateway. For more information about gateway configuration, see the **gateway** command under "System Configuration" in the *Command Reference* .

Configure the Synchronization Parameters

Configure the synchronization parameters.

```
#/cfg/slb/sync
>> Config Synchronization#prios d
>> Config Synchronization#peer 1
>> Peer Switch 1#addr 172.16.10.2
>> Peer Switch 1#ena
```

Apply and Save the Configuration

Apply and save the configuration changes on Application Switch 1.

```
#apply
#save
```

Configure Application Switch 2

Create the Necessary VLANs

The VLANs configured on Application Switch 2 are identical to those configured on Application Switch 1.

1. Configure VLAN 2 to include Application Switch port leading to Application Switches 3 and 4.

```
#/cfg/vlan 2
```

```
>> VLAN 2#add 2

Port 2 is an UNTAGGED port and its current PVID is 1.
Confirm changing PVID from 1 to 2 [y/n]:y

>> VLAN 2#ena
```

2. Configure VLAN 3 to include the Application Switch port leading to the VPN Gateways, and the trunking ports leading to Application Switch 2.

```
#/cfg/vlan 3

>> VLAN 3#add 3

Port 3 is an UNTAGGED port and its current PVID is 1.
Confirm changing PVID from 1 to 3 [y/n]:y

>> VLAN 3#add 7

Port 7 is an UNTAGGED port and its current PVID is 1.
Confirm changing PVID from 1 to 3 [y/n]:y

>> VLAN 3#add 8

Port 8 is an UNTAGGED port and its current PVID is 1.
Confirm changing PVID from 1 to 3 [y/n]:y

>> VLAN 3#ena
```

Disable STP and Create a Port Trunk

1. Disable Spanning Tree Protocol (STP) globally.

```
#/cfg/stp 1

>> Spanning Tree#off
```

2. Enable trunking between Application Switches 1 and 2 on port 7 and port 8.

```
#/cfg/trunk 1

>> Trunk group 1#add 7

Port 7 added.
>> Trunk group 1#add 8

Port 8 added.
>> Trunk group 1#ena
```

Configure One IP Interface for Each VLAN

When you configure the Application Switch IP interfaces, you can let the Application Switch automatically calculate the correct broadcast address for you. To do this, enter the subnet mask first, and then the address of the IP interface. Immediately after you provide the address of the IP interface, the calculated broadcast address will be displayed.

1. Configure an IP interface for the client traffic with VLAN 1.

```
#/cfg/ip/if 1
>> IP Interface 1#addr 192.168.10.2
>> IP Interface 1#mask 255.255.255.0
>> IP Interface 1#broad 192.168.10.255
>> IP Interface 1#vlan 1
>> IP Interface 1#ena
```

2. Configure an IP interface for the Application Switch traffic with VLAN 2.

```
#/cfg/ip/if 2
>> IP Interface 2#addr 172.26.10.2
>> IP Interface 2#mask 255.255.255.0
>> IP Interface 2#broad 172.26.10.255
>> IP Interface 2#vlan 2
>> IP Interface 2#ena
```

3. Configure an IP interface for the AVG traffic with VLAN 3.

```
#/cfg/ip/if 3
>> IP Interface 3#addr 172.16.10.2
>> IP Interface 3#mask 255.255.255.0
>> IP Interface 3#broad 172.16.10.255
>> IP Interface 3#vlan 3
>> IP Interface 3#ena
```

Specify a Default Gateway and Add a Static Route

The gateway and static route configuration on Application Switch 2 is identical to the one performed on Application Switch 1.

1. Specify the default gateway for Application Switch 2.

```
#/cfg/ip/gw 1

>> Default gateway 1#addr 192.168.10.10
```

2. Add the required static route.

```
#/cfg/ip/route

>> IP Static Route#add

Enter destination IP address:10.30.10.0


Enter destination subnet mask:255.255.255.0


Enter gateway IP address:172.26.10.254


Enter interface number: (1-256)2
```

3. Specify a secondary gateway for Application Switch 2.

   If link 1 on Application Switch 2 in Figure 9: Sample Web Server Accelerator Network Using Application Switches in Active-Standby Mode and a Cluster of AVGs on page 82 fails, non-https traffic will not be routed back to the Internet unless a secondary gateway, pointing to Application Switch 1, is defined on Application Switch 2.

```
#/cfg/ip/gw 2

>> Default gateway 2#addr 172.16.10.1
```

Configure and Enable the Synchronization Parameters

1. Enable layer 4 processing to allow SLB settings to be synchronized from the peer Application Switch (Application Switch 1 in this example).

```
#/cfg/slb
>> Layer 4#on
```

2. Configure the synchronization parameters.

```
#/cfg/slb/sync
>> Config Synchronization#prios d
>> Config Synchronization#peer 1
>> Peer Switch 1#addr 172.16.10.1
>> Peer Switch 1#ena
```

3. Enable VRRP globally.

```
#/cfg/vrrp
>> Virtual Router Redundancy Protocol#on
```

Apply and Save the Configuration

Apply and save the configuration changes on Application Switch 2.

```
#apply
```

```
#save
```

On Switch 1, Synchronize the Configuration to Application Switch 2

Before synchronizing the configuration, make sure the following requirements are met:

- Identical software versions running on both Application Switches.
- Identical password for the Admin user configured on both Application Switches.
- Absence of filters that block traffic on TCP port 3121.
- Accurate synchronization configuration applied and saved on both Application Switches.

Synchronize configuration on peers.

```
#/oper/slb/sync

Synchronizing VRRP, FILT, PORT and SLB configuration
to 172.16.10.2

Confirm synchronizing the configuration to 172.16.10.2 [y/n]:y
```

Caution:

The `/oper/slb/sync` command will synchronize the current filter, port, and server load balancing configuration between the two switches, besides the VRRP configuration.

Configure Application Switch 3

Create the Necessary VLANs

In the configuration on Application Switches 3 and Web 4, there will be four VLANs, where only one is shared with the VLAN configuration on Application Switches 1 and 2. VLAN 1 is used for connections to the real Web servers, VLAN 2 for connections to Application Switches 1 and 2, VLAN 3 for connections to the VPN Gateways, and VLAN 4 for connections to the management network. Since VLAN 1 is the default, only VLAN 2, VLAN 3, and VLAN 4 require additional configuration.

1. Enable VLAN tagging for switch port 7 and 8.

```
#/cfg/port 7

>> Port 7#tag e

Current VLAN tag support: disabled
New VLAN tag support: enabled
Port 7 changed to tagged.
>> Port 7#pvid 2

>> Port 7#../port 8

>> Port 8#tag e

Current VLAN tag support: disabled
New VLAN tag support:         enabled
Port 8 changed to tagged.
>> Port 8#pvid 2
```

2. Configure VLAN 2 to include the Application Switch port leading to Application Switch 1, and the trunking ports leading to Application Switch 4.

```
#/cfg/vlan 2

>> VLAN 2#add 2

Port 2 is an UNTAGGED port and its current PVID is 1.
Confirm changing PVID from 1 to 2 [y/n]:y

>> VLAN 2#add 7

>> VLAN 2#add 8

>> VLAN 2#ena
```

3. Configure VLAN 3 to include the Application Switch port leading to the VPN Gateways, and the trunking ports leading to Application Switch 4.

```
#/cfg/vlan 3

>> VLAN 3#add 3

Port 3 is an UNTAGGED port and its current PVID is 1.
Confirm changing PVID from 1 to 3 [y/n]:y

>> VLAN 3#add 7

>> VLAN 3#add 8

>> VLAN 3#ena
```

4. Configure VLAN 4 to include the Application Switch port leading to the management network.

```
#/cfg/vlan 4

>> VLAN 4#add 4

Port 4 is an UNTAGGED port and its current PVID is 1.
Confirm changing PVID from 1 to 4 [y/n]:y

>> VLAN 4#ena
```

Disable STP and Create a Port Trunk

1. Disable Spanning Tree Protocol (STP) globally.

```
#/cfg/stp 1

>> Spanning Tree#off
```

2. Configure port trunking between Application Switch 3 and Application Switch 4 on port 7 and port 8.

```
#/cfg/trunk 1

>> Trunk group 1#add 7

Port 7 added.
>> Trunk group 1#add 8

Port 8 added.
>> Trunk group 1#ena
```

Configure One IP Interface for Each VLAN

When you configure the Application Switch IP interfaces, you can let the Application Switch automatically calculate the correct broadcast address for you. To do this, enter the subnet mask first, and then the address of the IP interface. Immediately after you provide the address of the IP interface, the calculated broadcast address will be displayed.

1. Configure an IP interface for real Web server traffic on Application Switch 3 with VLAN 1.

```
#/cfg/ip/if 1

>> IP Interface 1#addr 10.30.10.1

>> IP Interface 1#mask 255.255.255.0
```

```
>> IP Interface 1#broad 10.30.10.255
>> IP Interface 1#vlan 1
>> IP Interface 1#ena
```

2. Configure an IP interface for the Application Switch traffic with VLAN 2.

```
#/cfg/ip/if 2
>> IP Interface 2#addr 172.26.10.3
>> IP Interface 2#mask 255.255.255.0
>> IP Interface 2#broad 172.26.10.255
>> IP Interface 2#vlan 2
>> IP Interface 2#ena
```

3. Configure an IP interface for the AVG traffic with VLAN 3.

```
#/cfg/ip/if 3
>> IP Interface 3#addr 10.20.10.1
>> IP Interface 3#mask 255.255.255.0
>> IP Interface 3#broad 10.20.10.255
>> IP Interface 3#vlan 3
>> IP Interface 3#ena
```

4. Configure an IP interface for the management traffic with VLAN 4.

```
#/cfg/ip/if 4
>> IP Interface 4#addr 10.10.10.1
>> IP Interface 4#mask 255.255.255.0
>> IP Interface 4#broad 10.10.10.255
>> IP Interface 4#vlan 4
>> IP Interface 4#ena
```

Configure Load Balancing Parameters on the Real Web Server and AVG

Set and enable the IP addresses of the VPN Gateways, and create a group in the switch for load balancing.

1. Configure Application Switch 3 to use server load balancing globally.

```
# /cfg/slb

>> Layer 4#on
```

2. For each VPN Gateway, create a Real Server IP address on the Application Switch.

   Defining the two VPN Gateways as real servers is necessary in order to make RTS (Return To Sender) work. The actual load balancing of the AVGs is performed by Application Switches 1 and 2.

```
# /cfg/slb/real 3

>> Real server 3#name iSD-SSL_1

>> Real server 3#rip 10.20.10.3

>> Real server 3#ena

>> Real server 3#../real 4

>> Real server 4#name iSD-SSL_2

>> Real server 4#rip 10.20.10.4

>> Real server 4#ena
```

3. For each Web server, create a Real Server IP address on the Application Switch.

```
# /cfg/slb/real 10

>> Real server 10#name Web_1

>> Real server 10#rip 10.30.10.10

>> Real server 10#ena

>> Real server 10#../real 11

>> Real server 11#name Web_2

>> Real server 11#rip 10.30.10.11
```

```
>> Real server 11#ena
```

4. Create a Real Server Group and add the real Web servers.

```
# /cfg/slb/group 1

>> Real server group 1#name Web_Servers

>> Real server group 1#add 10

>> Real server group 1#add 11
```

5. Set the load balancing metric and health check type for Real Server Group 1.

```
# /cfg/slb/group 1

>> Real server group 1#metric leastconns

>> Real server group 1#health tcp
```

Configure Port Processing

1. Enable server processing and filters on the port leading to the real Web servers.

   Enabling filter processing is necessary in order to make sure that RTS (Return To Sender) works.

```
# /cfg/slb/port 1

>> SLB port 1#server ena

>> SLB port 1#filt ena
```

2. Enable client processing on the port leading to Application Switch 1.

```
# /cfg/slb/port 2

>> SLB port 2#client ena
```

3. Enable client processing and Return To Sender (RTS) on the port leading to the VPN Gateway.

```
# /cfg/slb/port 3

>> SLB port 3#client ena
```

```
>> SLB port 3#rts ena
```

4. Enable client processing and RTS on the trunk ports connecting Application Switch 3 and Application Switch 4.

```
# /cfg/slb/port 7

>> SLB port 7#client ena

>> SLB port 7#rts ena

>> SLB port 7#../port 8

>> SLB port 8#client ena

>> SLB port 8#rts ena
```

Create and Configure a Virtual Server

1. Create a virtual server in VLAN 1 by specifying the IP address and enabling the server.

```
# /cfg/slb/virt 1

>> Virtual Server 1#vip 10.30.10.100

>> Virtual Server 1#ena
```

2. Enable service on virtual port 80 (HTTP) and 81 for the virtual server, and connect real server group 1 to the virtual server.

   Enable service also on port 81 for unencrypted communication between the VPN Gateways and the real Web servers. Recall that the real Web servers must also be configured to listen for AVG traffic on port 81. The preceding step also connects the real Web servers in server group 1 to the enabled virtual server. The HTTP service on port 80 for non-SSL Web traffic from clients to the real Web servers in real server group 1 is also enabled. Thus, the load balancing scheme for real server group 1 includes traffic on port 80 and 81.

```
# /cfg/slb/virt 1/service http

>> Virtual Server 1 http Service#group 1

>> Virtual Server 1 http Service#../service 81

>> Virtual Server 1 81 Service#group 1
```

Specify the Default Gateways and Enable HTTP Access

1. Specify the default gateways, which in this example correspond to the respective IP interface 2 in VLAN 2 on Application Switches 1 and 2.

```
# /cfg/ip/gw 1

>> Default gateway 1#addr 172.26.10.1

>> Default gateway 1#ena

>> Default gateway 1#../gw 2

>> Default gateway 2#addr 172.26.10.2

>> Default gateway 2#ena
```

2. Enable the HTTP responder in order to respond to the health checking performed by Application Switch 1.

```
# /cfg/sys

>> System#http ena
```

Note:

When enabling the HTTP responder you may also enable the Web browser based switch management interface (Web OS BBI), depending on the Web OS version and the Application Switch model. You should therefore make sure that you are not using the default password for the Administrator user on the Application Switch. If you change the password on Application Switch 3, make sure that you configure Application Switch 4 to use the same password or the configuration synchronization between the two Application Switches will fail.

Configure VRRP

VRRP is configured for failover (redundancy) between Application Switches 3 and 4, in the event one of the Application Switches fails. In this configuration, Application Switch 3 will be the master (achieved by setting `prio` to `101`).

1. Globally turn on the VRRP.

```
# /cfg/vrrp/on
```

2. Configure and enable virtual router 1 for IP interface 1 on VLAN 1.

```
# /cfg/vrrp/vr 1
>> VRRP Virtual Router 1#vrid 1
>> VRRP Virtual Router 1#addr 10.30.10.254
>> VRRP Virtual Router 1#if 1
>> VRRP Virtual Router 1#prio 101
>> VRRP Virtual Router 1#share dis
>> VRRP Virtual Router 1#track/ifs e
>> VRRP Virtual Router 1 Priority Tracking#../ena
```

3. Configure and enable virtual router 2 for IP interface 2 on VLAN 2.

```
# /cfg/vrrp/vr 2
>> VRRP Virtual Router 2#vrid 2
>> VRRP Virtual Router 2#addr 172.26.10.254
>> VRRP Virtual Router 2#if 2
>> VRRP Virtual Router 2#prio 101
>> VRRP Virtual Router 2#share dis
>> VRRP Virtual Router 2#track/ifs e
>> VRRP Virtual Router 2 Priority Tracking#../ena
```

4. Configure and enable virtual router 3 for IP interface 3 on VLAN 3.

```
# /cfg/vrrp/vr 3
>> VRRP Virtual Router 3#vrid 3
>> VRRP Virtual Router 3#addr 10.20.10.254
>> VRRP Virtual Router 3#if 3
>> VRRP Virtual Router 3#prio 101
>> VRRP Virtual Router 3#share dis
>> VRRP Virtual Router 3#track/ifs e
>> VRRP Virtual Router 3 Priority Tracking#../ena
```

5. Configure and enable virtual router 4 for IP interface 4 on VLAN 4.

```
# /cfg/vrrp/vr 4
>> VRRP Virtual Router 4#vrid 4
>> VRRP Virtual Router 4#addr 10.10.10.254
>> VRRP Virtual Router 4#if 4
>> VRRP Virtual Router 4#prio 101
>> VRRP Virtual Router 4#share dis
>> VRRP Virtual Router 4#track/ifs e
>> VRRP Virtual Router 4 Priority Tracking#../ena
```

6. Configure and enable virtual router 5 as the VIP virtual router instance.

```
# /cfg/vrrp/vr 5
>> VRRP Virtual Router 5#vrid 5
>> VRRP Virtual Router 5#addr 10.30.10.100
>> VRRP Virtual Router 5#if 1
>> VRRP Virtual Router 5#prio 101
>> VRRP Virtual Router 5#share dis
>> VRRP Virtual Router 5#track/ifs e
>> VRRP Virtual Router 5 Priority Tracking#../ena
```

Configure the Synchronization Parameters

Configure the synchronization parameters.

```
# /cfg/slb/sync
>> Config Synchronization#prios d
>> Config Synchronization#peer 1
>> Peer Switch 1#addr 172.26.10.4
```

```
>> Peer Switch 1#ena
```

Apply and Save the Configuration

Apply and save the configuration changes on Application Switch 3.

```
# apply
# save
```

Configure Application Switch 4

Create the Necessary VLANs

The VLANs configured on Application Switch 4 are identical to those configured on Application Switch 3.

1. Enable VLAN tagging for switch port 7 and 8.

```
# /cfg/port 7

>> Port 7#tag e

Current VLAN tag support: disabled
New VLAN tag support:               enabled
Port 7 changed to tagged.
>> Port 7#pvid 2

>> Port 7#../port 8

>> Port 8#tag e

Current VLAN tag support: disabled
New VLAN tag support:               enabled
Port 8 changed to tagged.
>> Port 8#pvid 2
```

2. Configure VLAN 2 to include the Application Switch port leading to Application Switch 2, and the trunking ports leading to Application Switch 3.

```
# /cfg/vlan 2

>> VLAN 2#add 2

Port 2 is an UNTAGGED port and its current PVID is 1.
Confirm changing PVID from 1 to 2 [y/n]:y
```

```
>> VLAN 2#add 7

>> VLAN 2#add 8

>> VLAN 2#ena
```

3. Configure VLAN 3 to include the Application Switch port leading to the VPN Gateways, and the trunking ports leading to Application Switch 3.

```
# /cfg/vlan 3

>> VLAN 3#add 3

Port 3 is an UNTAGGED port and its current PVID is 1.
Confirm changing PVID from 1 to 3 [y/n]:y

>> VLAN 3#add 7

>> VLAN 3#add 8

>> VLAN 3#ena
```

4. Configure VLAN 4 to include the Application Switch port leading to the management network.

```
# /cfg/vlan 4

>> VLAN 4#add 4

Port 4 is an UNTAGGED port and its current PVID is 1.
Confirm changing PVID from 1 to 4 [y/n]:y

>> VLAN 4#ena
```

Disable STP and Create a Port Trunk

1. Disable Spanning Tree Protocol (STP) globally.

```
# /cfg/stp 1

>> Spanning Tree#off
```

2. Enable trunking between Application Switches 3 and 4 on port 7 and port 8.

```
# /cfg/trunk 1

>> Trunk group 1#add 7
```

```
Port 7 added.
>> Trunk group 1#add 8

Port 8 added.
>> Trunk group 1#ena
```

Configure One IP Interface for Each VLAN

When you configure the Application Switch IP interfaces, you can let the Application Switch automatically calculate the correct broadcast address for you. To do this, enter the subnet mask first, and then the address of the IP interface. Immediately after you provide the address of the IP interface, the calculated broadcast address will be displayed.

1. Configure an IP interface in VLAN 1 for real Web server traffic on Application Switch 4.

```
# /cfg/ip/if 1

>> IP Interface 1#addr 10.30.10.2

>> IP Interface 1#mask 255.255.255.0

>> IP Interface 1#broad 10.30.10.255

>> IP Interface 1#vlan 1

>> IP Interface 1#ena
```

2. Configure an IP interface in VLAN 2 for the traffic with Application Switch 2.

```
# /cfg/ip/if 2

>> IP Interface 2#addr 172.26.10.4

>> IP Interface 2#mask 255.255.255.0

>> IP Interface 2#broad 172.26.10.255

>> IP Interface 2#vlan 2

>> IP Interface 2#ena
```

3. Configure an IP interface in VLAN 3 for the AVG traffic.

```
# /cfg/ip/if 3

>> IP Interface 3#addr 10.20.10.2

>> IP Interface 3#mask 255.255.255.0
```

```
>> IP Interface 3#broad 10.20.10.255

>> IP Interface 3#vlan 3

>> IP Interface 3#ena
```

4. Configure an IP interface in VLAN 4 for the management traffic.

```
# /cfg/ip/if 4

>> IP Interface 4#addr 10.10.10.2

>> IP Interface 4#mask 255.255.255.0

>> IP Interface 4#broad 10.10.10.255

>> IP Interface 4#vlan 4

>> IP Interface 4#ena
```

Specify the Default Gateways and Enable HTTP Access

1. Specify the default gateways, which in this example are Application Switches 1 and 2 using IP interface 2 on VLAN 2.

```
# /cfg/ip/gw 1

>> Default gateway 1#addr 172.26.10.1

>> Default gateway 1#ena

>> Default gateway 1#../gw 2

>> Default gateway 2#addr 172.26.10.2

>> Default gateway 2#ena
```

2. Enable the HTTP responder in order to respond to the health checking performed by Application Switch 2.

```
# /cfg/sys

>> System#http ena
```

Note:

When enabling the HTTP responder you may also enable the Web browser based switch management interface (Web OS BBI), depending on the Web OS version and the Application Switch model. You should therefore make sure that

you are not using the default password for the Administrator user on the Application Switch. If you change the password, make sure that you specify the same password as on Application Switch 3 or the configuration synchronization between the two Application Switches will fail.

Configure the Synchronization Parameters

1. Enable layer 4 processing to allow SLB settings to be synchronized from the peer Application Switch (Application Switch 3 in this example).

```
# /cfg/slb

>> Layer 4#on
```

2. Configure the synchronization parameters.

```
# /cfg/slb/sync

>> Config Synchronization#prios d

>> Config Synchronization#peer 1

>> Peer Switch 1#addr 172.26.10.3

>> Peer Switch 1#ena
```

3. Enable VRRP globally.

```
# /cfg/vrrp

>> Virtual Router Redundancy Protocol#on
```

Apply and Save the Configuration

Apply and save the configuration changes on Application Switch 4.

```
# apply
```

```
# save
```

On Switch 3, Synchronize the Configuration to Application Switch 4

Before synchronizing the configuration, make sure the following requirements are met:

- Identical software versions running on both Application Switches.
- Identical password for the Admin user configured on both Application Switches.
- Absence of filters that block traffic on TCP port 3121.

Accurate synchronization configuration applied and saved on both Application Switches.

Synchronize configuration on peers.

```
# /oper/slb/sync

Synchronizing VRRP, FILT, PORT and SLB configuration
to 172.26.10.4

Confirm synchronizing the configuration to 172.26.10.4 [y/n]:y
```

Caution:

The `/oper/slb/sync` command will synchronize the current filter, port, and server load balancing configuration between the two switches, besides the VRRP configuration.

Chapter 7: Configuring the AVG to Rewrite Client Requests

If the client's Web browser is not capable of meeting the cipher list requirement you have specified for a virtual SSL server on the AvayaVPN Gateway (AVG), you can enable the rewrite functionality in order to let the Web server display a customized error message. Without this functionality, the SSL session between the client and the VPN Gateway would simply be terminated during the SSL handshake.

Setting up Rewrite of Weak Cipher Client Requests

This example assumes that you already have configured a virtual SSL server for the HTTPS service, as described in the Web Server Accelerator example on Web Server Accelerator on page 21 .

Log in as the administrator to the AVG cluster by using a Telnet or SSH connection to the Management IP address (MIP). The configuration changes will automatically be propagated to all VPN Gateways in the cluster.

1. Decide which virtual SSL server to configure.

   View the displayed information about certificates and virtual SSL servers. If you have configured more than one virtual SSL server, identify the server used for HTTPS by verifying that the Listen port value is set to 443 (HTTPS). In this example virtual SSL server 1 is used as an example for the configuration changes.

   ```
   >> Main#cfg/ssl

   >> SSL#cur
   ```

2. Examine the current settings for the chosen virtual SSL server.

   Examine the settings and make sure that the virtual SSL server is enabled. Take special notice of the SSL settings and the current cipher list, which denotes the cipher strength. If you have not modified the cipher list, it should correspond to the default value of `ALL@STRENGTH`.

   ```
   >> SSL#server

   Enter virtual server number: (1-)1
   ```

   (example)

```
>> Server 1#cur

Server 1:
Server name = test_server
IP addr(s) of server = 192.168.10.100
Standalone mode = off
Listen port of server = 443 (https)
Real server IP addr = 0.0.0.0
Real server port = 81
Type (generic/http/socks) = generic
Transparent proxy mode (on/off) = on
Enable virtual server = enabled
SSL Settings:
Server certificate = 1
SSL cache size = 4000
SSL cache timeout = 5m
List of accepted signers of client certificates =
List of CA chain certificates =
Protocol version = ssl3
Certificate verification level = none
Cipher list = ALL@STRENGTH
Enable SSL = enabled

TCP Settings:
```

(continued on-screen)

3. Set the SSL server type to HTTP.

```
>> Server 1#type

Current value: generic
Type (generic/http/socks):http
```

In order to make use of high-level HTTP capabilities, such as rewriting client requests, the virtual SSL server type needs to be changed to `http`. Also, without changing the type to `http`, the HTTP Settings menu and Rewrite menu are not even activated.

4. Access the HTTP menu and the Rewrite menu, then enable the rewrite functionality.

```
>> Server 1#http

>> HTTP Settings#rewrite

>> Rewrite#rewrite

Current value: off
Enable rewrite (on/off):on

You may need to adjust the server #/ssl/cipher setting.
For step-up certificates we recommend ALL:-RC2:-SHA1:@STRENGTH
```

5. Specify which cipher list to accept.

```
>> Rewrite Menu for Server 1#ciphers

Current value: HIGH:MEDIUM

Enter cipher list:RC4:ALL:!EXPORT:!DH
```

Specify the rewrite cipher list (in uppercase letters) that you actually require here, or accept the default rewrite cipher list (`HIGH:MEDIUM`). The cipher strength that you actually require must always be higher than the cipher strength provided by the cipher list specified under the SSL Settings menu (you viewed these settings in step 2). This way, clients using a Web browser that does not meet the actual cipher strength requirements can still establish an SSL session and get a customized error message from the Web server (or a default error message from the AVG), even if the actual requirements are not met.

The rewrite function is only triggered when a client browser can perform the lower cipher strength specified under the SSL Settings menu, but is unable to perform the higher cipher strength you specify in this step. When both cipher requirements are met, the SSL session is established transparently without the involvement of the rewrite function.

Again, verify that the cipher list you specify in this step provides higher cipher strength than the cipher list you took note of in step 2. For more information about ciphers, see the "Supported Ciphers" appendix in the *Users Guide*. If you find that you need to change the cipher list specified under the SSL Settings menu, use the command `/cfg/ssl/server`, specify the desired virtual SSL server, and then type the command `ssl/ciphers`.

6. Configure the SSL server to let the Web server handle the client response.

   In this example, configure the SSL server to let the Web server handle the response sent back to the client browser when the rewrite function is triggered. This configuration also requires specifying a valid URI, pointing to a resource on the Web server (described in the next step).

```
>> Rewrite#response

Current value: iSD

Enter the source of response (iSD/WebServer):WebServer
```

If you choose to let the AVG (iSD) handle the response, it returns a predefined error message to the client browser if the required cipher strength is not met. An example of such a predefined error message emanating from the AVG can be:

```
                    http://www.example.com/
         requires stronger cryptography support of your
                              browser.
  Your browser used TLSv1/SSLv3, with the cipher suite RC4-MD5,
        which is not one of the accepted ones: NULL-SHA
```

7. Specify the URI to add when rewriting the client request.

```
>> Rewrite#URI

Current value: /cgi-bin/weakcipher

Enter the URI address (WebServer response only): /
cipheralert.asp
```

Specify a URI pointing to a resource on the Web server to which the client made its initial request. The URI can point to a static HTML page, or to a server-side script generating a dynamic error message. A dynamic error message can be based on the session specific information that the rewrite function automatically adds to the modified request that is passed on to the Web server.

If you specify `MEDIUM` as the cipher list (used only as an example) and `/cipheralert.asp` for the URI, the rewritten client request sent on to the Web server would look similar to this:

`"GET /cipheralert.asp?c=TLSv1/SSLv3:RC4-MD5&cs=MEDIUM HTTP/1.0"` Where `c` stands for client, and `cs` stands for (required) cipher suite.

8. Apply your settings.

```
>> Rewrite#apply

Changes applied successfully.
```

Chapter 8: HTTP to HTTPS Redirection

This chapter describes how to configure the Avaya VPN Gateway (AVG) to automatically transform an HTTP client request into the required HTTPS request. By configuring such a redirect service on the AVG, the user can simply enter the fully qualified domain name in the web browser's address field, without having to specify (or knowing) the protocol required to establish a secure connection.

The redirect service is configured by adding an additional virtual HTTP server. When the virtual HTTP server on the AVG receives a request, it will redirect the browser to the virtual HTTPS server by sending an HTTP Location header to the browser.

This configuration example assumes that you have already set up a working HTTPS server for web server acceleration. If not, see Basic Applications on page 21 in this manual.

Configure HTTP to HTTPS Redirection

Log in as the administrator to the AVG cluster.

1. Create a virtual HTTP server.

   This step creates a new virtual HTTP server on the AVG.

```
# /cfg/ssl/server

Enter virtual server number: (1-)2

Creating Server 2

>> Server 2#type

Current value: generic
Type (generic/http/socks): http
```

2. Define a name for the virtual HTTP server.

   This step lets you specify a name, by which you can identify the virtual HTTP server. To view the numbers and related names of all configured servers, use the `/info/servers` command. The name you specify is mainly intended for your own reference, and is not critical for the configuration itself. As the following example suggests, the name can indicate the service for which the virtual server is created.

```
>> Server 2#name
```

```
Current value:""

Enter new server name:redirect service
```

3. Set listen TCP port for the HTTP server.

   Each time you create a new virtual server, the listen port is automatically set to 443. For the HTTP to HTTPS redirect service in this example, the virtual HTTP server must be set to listen to port 80 (the default port used for HTTP).

```
>> Server 2#port

Current value:<not set>

[443 (https)]
Enter listen port number :80
```

4. Assign the desired virtual server IP address to the HTTP server.

   This is the address the HTTP client will connect to. It will typically be the same as the address of the virtual HTTPS server.

```
>> Server 2#vips

Current value:""

Enter server ips (comma separated): 192.168.10.100
```

5. Disable SSL for the virtual HTTP server.

```
>> Server 2#ssl

>> SSL Settings#dis
```

6. Enable HTTPS to HTTP redirection.

```
>> SSL Settings#../http/httpsredir

Current value: off
Perform http to https redirect (on/off): on
```

7. Apply the changes.

```
>> >> HTTP Settings#apply

Changes applied successfully.
```

Application Switch Configuration

When using the AVG with an Application Switch, the Application Switch must be configured to redirect HTTP traffic to the AVG HTTP server.

1. Create a filter to redirect client HTTP traffic intended for port 80 on the virtual server IP (VIP) address to the AVG real server group.

   When this filter is added to the switch port leading to the Internet, incoming HTTP traffic destined for the virtual server IP address is redirected to the VPN Gateways in real server group 2. Firewall redirect hash method is also enabled, using redirection based on hashing on both the source IP and the destination IP of the packets.

```
# /cfg/slb/filt 101
>> Filter 101#dip 192.168.10.100
>> Filter 101#dmask 255.255.255.255
>> Filter 101#proto tcp
>> Filter 101#dport http
>> Filter 101#action redir
>> Filter 101#group 2
>> Filter 101#rport http
>> Filter 101#adv/fwlb e
>> Filter 101 Advanced#../ena
```

2. Add the filter to the client port leading to the Internet.

```
# /cfg/slb/port 1
>> SLB Port 1#add 101
```

3. Apply and save the Application Switch configuration changes.

```
# apply # save
```

Chapter 9: Load Balancing of Backend Servers

This section describes the steps for configuring the Avaya VPN Gateway (AVG) to perform load balancing of backend servers. The VPN Gateway can load balance both encrypted and unencrypted backend server connections. When the VPN Gateway is configured to perform end to end encryption, it must load balance the backend servers, as the Application Switch cannot load balance encrypted traffic. The Application Switch however, can still load balance ordinary client HTTP traffic to the backend servers, as well as client HTTPS requests handled by the VPN Gateways.

AVG-based Server Load Balancing

In the following procedure, many of the server load balancing options are left to their default values. For more configuration options, see these sections:

- Metrics for Server Load Balancing on page 132
- Health Checks in Server Load Balancing on page 133
- String Matching in Server Load Balancing on page 135
- Persistent Client Connections in Server Load Balancing on page 136

A virtual SSL server must be configured on the VPN Gateway prior to configuring server load balancing.

Note:

For details about any of the menu commands appearing in this configuration example, see the *Command Reference* .

1. Select a virtual SSL server, through which load balancing is performed.

```
# /cfg/ssl/server

Enter virtual server number: (1-)1

>> Server 1#
```

   For a complete listing of all configured virtual SSL servers on your AVG, use the `/info/servers` command.

2. Define each backend server by specifying IP address and TCP port.

   The virtual SSL server will initiate requests to the specified IP address and TCP port of the backend server. In this configuration example, the TCP port is set to 443

(HTTPS) as the load balancing is thought to be part of a wider end to end encryption configuration.

```
>> Server 1#adv/loadbalanc/backend

Enter backend server number: (1-)1

Creating Backend Server 1
>> Backend Server 1#ip 10.20.10.2

>> Backend Server 1#port 443

>> Backend Server 1#../backend 2

>> Backend Server 2#ip 10.20.10.3

>> Backend Server 2#port 443
```

Note:

The backend servers must also be configured to listen for incoming requests on the same TCP ports as you specify in the load balancing configuration on the AVG.

3. Enable load balancing and verify the configuration.

```
>> Backend Server 2#../ena

>> Load Balancing Settings#cur
```

4. Apply your changes.

```
>> Load Balancing Settings#apply
```

Metrics for Server Load Balancing

Metrics are used for selecting which backend server, configured within the realm of a virtual SSL server, that will receive the next client connection. The available metrics are `hash`, `roundrobin` (round robin), and `leastconn` (least connections). Each of these are explained. The default metric is `hash`.

To change the load balancing metric for backend servers configured under a specific virtual SSL server, use the following command at any menu prompt:

```
>> #/cfg/ssl/server #/adv/loadbalanc/metric
```

```
Current value: hash
Enter metric (hash/roundrobin/leastconn):
```

Hash

A mathematical hash on the client source IP is used when selecting a backend server. All requests from a specific client will be sent to the same backend server, provided the health check mechanism declares the backend server as up during the complete session. Persistency in client connections is therefore inherent in the hash load balancing metric. However, hash is not recommended when many clients share the same source IP address (such as proxied clients), because all clients are directed to the same backend server without the benefit of load balancing the traffic across the available backend servers.

Round Robin

New client connections are issued to each available backend server in turn. The first backend server gets the first connection, the second backend server gets the next connection, followed by the third backend server, and so on. When all configured backend servers have received at least one connection, the issuing process starts over with the first backend server.

The `roundrobin` metric can be combined with persistency based on cookies or information in the SSL session.

Least Connections

With the `leastconn` metric, the number of connections currently open on each backend server is measured in real time. The backend server with the fewest connections is considered to be the best choice for the next incoming client connection request.

This option is the most self-regulating, with the fastest servers typically getting the most connections over time.

The `leastconn` metric can be combined with persistency based on cookies or information in the SSL session.

Health Checks in Server Load Balancing

By determining health for each backend server included in the load balancing configuration, load balancing is optimized. Only those backend servers that respond positively to the health

check will be included in the load balancing scheme at any given time. This ensures effective utilization of available network bandwidth and server resources.

The available health check methods are `none`, `tcp`, `ssl`, `auto`, and `script`. Each of these are explained. The default health check method is set to `auto`. If you have a cluster of VPN Gateways, remember that each VPN Gateway performs its own health checks of backend servers.

To change the health check method for backend servers configured under a specific virtual SSL server, use the following command at any menu prompt:

```
>> #/cfg/ssl/server #/adv/loadbalanc/health

Current value: auto
Enter health check type (none/tcp/ssl/auto/script):
```

None

Specifies that no health checking of backend servers should be performed. If you have enabled load balancing, all backend servers will be included in the load balancing scheme at all times, regardless of the actual backend server status. If a backend server is inaccessible, client connections issued to that backend server (based on the selected load balancing metric) will fail. Failed backend server connections are logged and can be viewed using the `/stats/server #/becnctfail` command.

TCP Health Checks

With the `tcp` health check option, each VPN Gateway opens a TCP connection to each backend server, using the IP address and port number specified in the backend server configuration.

SSL Health Checks

With the `ssl` health check option, each VPN Gateway first opens a TCP connection to each backend server, using the IP address and port number specified in the backend server configuration. After the TCP connection has been successfully opened, a SSL connection is established. Thereafter, the SSL connection is shut down and the TCP connection is closed.

Using the SSL health check option requires that SSL connect is enabled on both the virtual SSL server level, and the individual backend server level. SSL connect is enabled on both levels by default.

Auto Health Checking

With the `auto` health check option, each VPN Gateway first opens a TCP connection to each backend server, using the IP address and port number specified in the backend server configuration. After that, a SSL connection is established only to those backend servers on which SSL connect is enabled. Thereafter, possible SSL connections are shut down, and all TCP connections are closed.

The default `auto` health check method is well suited in a configuration where the SSL connect setting is not uniform among the load balanced backend servers. Each backend server is health checked by using one out of two methods: a TCP health check only, or a TCP health check in combination with an SSL health check. As the option name implies, the `auto` health check automatically determines whether SSL connect is enabled on a particular backend server.

Script-Based Health Checking

The `script` health check option requires that you have created a customized health check script by using the script editing functions available in the Health Check Script menu (`/cfg/ssl/server #/adv/loadbalanc/script`).

For more information about creating a customized health check script, see Script-Based Health Checks on page 161.

String Matching in Server Load Balancing

By selecting `string` as the load balancing type, only those backend servers for which you have specified one or more match strings are included in the load balancing scheme. Furthermore, load balancing of client connection requests among these backend servers occurs only if a match of the string definition is found in the specified location in a client connection request. If a match is found, the current load balancing metric (hash, round robin, or least connections) is applied. For more information about creating match strings and assigning them to individual backend servers, see String-Based Load Balancing and Blocking on page 149 .

To change the load balancing method from `all` to `string`, use the following command at any menu prompt:

```
>> #/cfg/ssl/server #/adv/loadbalanc/type

Current value: all
Enter load balancing type (all/string):string
```

Note:

String matching in load balancing overrides any persistency option (`cookie` or `session`), which are ignored.

Persistent Client Connections in Server Load Balancing

When a virtual SSL server on the AVG is configured to perform load balancing of backend servers, a persistency strategy can be selected to ensure that all connections from a specific client reach the same backend server within the complete session. When a client initiates a connection request to establish a new session, the connection is issued to a backend server according to the load balancing metric you have selected (hash, round robin, or least connections). For all subsequent client requests within an established session, however, the chosen persistency strategy comes into play and overrides the load balancing metric.

The available persistency options are **none**, **cookie** and **session**. Each of these are explained. The default persistency option is set to `none`.

To change the persistency option for backend servers configured under a specific virtual SSL server, use the following command at any menu prompt:

```
>> #/cfg/ssl/server #/adv/loadbalanc/persistenc

Current value: none

Enter persistence (none/cookie/session):
```

Cookie-Based Persistence

Cookies are strings passed through HTTP from servers to browsers. Based on the mode of operation, cookies are inserted by either the AVG or the backend server. After a client receives a cookie, a server can poll that cookie with a **`GET`** command, which allows the querying server to positively identify the client as the one that received the cookie earlier.

In the AVG, cookies are used to route client traffic back to the same physical backend server to maintain session persistence.

Permanent and Temporary Cookies

Cookies can be either permanent or temporary. A permanent cookie is stored on a client computer's hard drive, with the cooperation of the client Web browser. On each subsequent request to the same site, the client Web browser will send the appropriate cookie. A permanent cookie may contain a user ID, or user preferences for a Web site that allows their pages to be customized.

A temporary cookie is only valid for a specific Web browser session, and expires as soon as the client Web browser is shut down. By default, the AVG uses temporary cookies, which are sufficient for maintaining persistency between clients and backend servers within a given session. You can however specify an expiration date or expiration time using the **`expires`** and **`expiresdel`** commands.

Client Browsers that do not Accept Cookies

Under normal conditions, most Web browsers are configured to accept cookies. However, if a client browser is configured to not accept cookies, you must use `hash` as the load-balancing metric to maintain session persistence.

With the persistence strategy set to `cookie`, session persistence for browsers that do not accept cookies will be based on the client source IP address instead.

Cookie Format

A cookie can be defined in the HTTP header or placed in the URL for hashing.

The AVG uses only cookies that are defined in the HTTP header. The cookie is defined as a "`Name=Value`" pair and can appear along with other parameters and cookies.

Cookie Properties

Cookies are configured on the AVG by defining the following properties:

- `name` (cookie name, maximum 254 bytes)

  Identifies the cookie used by the virtual SSL server for persistence purposes. The maximum length of a cookie name is restricted to 254 bytes.

- `domain` (domain name)

  Sets a domain name for the cookie (for example, .example.com), to make sure the cookie gets returned to the server that initiated the cookie, even if the client browser sends the cookie to another server responding to the same host name (for example, www.example.com).

- `expires` (date and time)

  Sets an absolute expiration date for the cookie.

- `expiresdel` (value in seconds)

  Sets a time frame during which the cookie will be valid.

- `offset` (integer value indicating number of bytes, 1-64)

- Defines the starting point of the real cookie value within a longer string. The offset value directs the AVG to start looking for the real cookie value at the specified location in the string.
- `length` (integer value indicating number of bytes, 0-64)

  Defines the number of bytes to extract for the cookie value within a longer string. The specified length may vary depending on whether you want to include both backend server IP address and virtual server IP address (cookie length = 16) in the cookie information, or only backend server IP address (cookie length = 8).

Cookie properties are configured by using the related menu options in the Cookie Settings Menu:

```
>> Main#/cfg/ssl/server #/adv/loadbalanc/cookie

[Cookie Settings Menu]
mode                   - Set cookie mode
name                    - Set cookie name
domain               - Set cookie domain
expires            - Set cookie expires
expiresdel   - Set cookie expires delta
localvips      - Configure other local VIPs
offset              - Set cookie value offset
length              - Set cookie value length
cur                     - Display current settings
```

Cookie Mode of Operation

The AVG supports the following modes of operation for cookie-based persistence:

- `insert`

  When a client sends a connection request without a cookie, the backend server responds with the requested data, and the AVG inserts a cookie into the data packet. The AVG embeds the IP address of the backend server that received the initial request into this cookie, and forwards the cookie to the client. The AVG then uses this cookie on all subsequent connection requests from the same client (within a given session) to bind to the backend server that was first selected using the current load balancing metric.

  Insert mode is the default mode of cookie operation.

- `passive`

  With passive mode, the backend server must be configured to embed a cookie in the response to the client request. The AVG will then look for this cookie in all subsequent connection requests (within a given session) from the same client, before establishing a connection to a backend server.

  If you configure your backend server to include an IP address in hexadecimal form as the cookie value (requires a cookie length value of exactly 8 bytes), the AVG will use that IP

address to direct all subsequent traffic within a given session to the same backend server.

If you configure your backend server to embed a string of characters as the cookie value,the AVG will perform a hash on the cookie value to direct all subsequent traffic within a given session to the same backend server.

- `rewrite`

  With rewrite mode, the backend server must be configured to return a special persistence cookie, which the AVG is configured to recognize. When recognized, the AVG intercepts the cookie and rewrites the value to include server-specific information before sending it on to the client. Subsequent connection requests (within a given session) from the same client are sent to the same backend server.

The cookie mode of operation can be changed by using the **mode** menu option in the Cookie Settings Menu:

```
>> Cookie Settings#mode

Current value: i
Cookie mode (insert/passive/rewrite):
```

Configuring Cookie-Based Persistence

Example 1: Using Insert Cookie Mode

1. Set the load balancing metric to either `roundrobin` or `leastconn`.

   ```
   >> Main#/cfg/ssl/server 1/adv/loadbalanc/metric

   Current value: hash
   Enter metric (hash/roundrobin/leastconn):
   ```

   When using the `insert` cookie mode of operation, always select `roundrobin` or `leastconn` as the load balancing metric.

   - With `roundrobin` as the selected load balancing metric, initial client connection requests are issued to each backend server in turn.
   - With `leastconn` as the selected load balancing metric, initial client connection requests are issued to the backend server currently having the fewest open connections.

2. Enable cookie-based persistence on the virtual SSL server.

```
>> Load Balancing Settings#persistenc

Current value: none
Enter persistence (none/cookie/session):cookie
```

3. Specify `insert` as the cookie mode of operation.

```
>> Load Balancing Settings#cookie/mode

Cookie mode (insert/passive/rewrite):insert
```

4. Define a suitable name for the cookie.

```
>> Cookie Settings#name

Current value: ISDSSL
Enter cookie name:
```

When using the `insert` cookie mode, the cookie name you define is the cookie name that will be embedded in the cookie. If you change the cookie name from the default value, make sure the length does not exceed 254 bytes.

5. Specify the offset value for the cookie.

```
>> Cookie Settings#offset

Current value: 1
Enter starting point of cookie value [1-64]:1
```

With `insert` cookie mode, always set the cookie offset value to 1.

6. Specify the length value for the cookie.

```
>> Cookie Settings#length

Current value: 8
Enter number of bytes to extract [0-64]:
```

- To include only the backend server IP address, set the cookie length to 8 bytes (default value).
- To include both the IP address of the backend server and the IP address of the Virtual server on the Application Switch, set the cookie length to 16 bytes.

7. Verify your cookie configuration.

```
>> Cookie Settings#cur

Collecting data, please wait...

Cookie Settings:

Cookie mode = i<indicates "insert">

Cookie name =<your cookie name as embedded by the SSL VPN>

Cookie value offset = 1

Cookie value length =<your specified extract length of 8 or 16>
```

8. Apply your settings.

```
>> Cookie Settings#apply

Changes applied successfully.
```

Example 2: Using Passive Cookie Mode

1. Specify the appropriate load balancing metric for the backend servers.

```
>> Main#/cfg/ssl/server 1/adv/loadbalanc/metric

Current value: hash
Enter metric (hash/roundrobin/leastconn):
```

   - If the backend server is configured to embed an IP address in the cookie, select `roundrobin` or `leastconn` as the metric.
   - If the backend server is configured to embed a string of characters as the cookie value, select hash as the load balancing metric. For more information about load balancingmetrics, see Metrics for Server Load Balancing on page 132.

2. Enable cookie-based persistence on the virtual SSL server.

```
>> Load Balancing Settings#persistenc

Current value: none
Enter persistence (none/cookie/session):cookie
```

3. Specify `passive` as the cookie mode of operation.

```
>> Load Balancing Settings#cookie/mode
```

```
Cookie mode (insert/passive/rewrite):passive
```

4. Specify the cookie name to match.

```
>> Cookie Settings#name

Current value: ISDSSL
Enter cookie name:
```

When using the `passive` cookie mode, the cookie name you specify on the AVG is significant to the functionality. When a backend server generates a cookie, the AVG will intercept and act only on those cookies for which a match of the cookie name you specify is found.

You can use the wildcard character * at the end of the name you specify. If more than one cookie matches, only the first one will be used. When a match of the cookie name is found, the AVG will perform one of the following actions, depending on the information found in the cookie:

- If the backend server embeds an IP address in the cookie, the AVG will use the IP address information in the cookie to direct all subsequent traffic within a given session to the corresponding backend server.
- If the backend server embeds a string of characters as the cookie value, the AVG will perform a hash on the cookie value. The AVG will then select a backend server and direct all subsequent traffic within a given session to the same backend server, based on the hashed cookie value. To perform the hash, the load balancing metric must be set to `hash`. If the load balancing metric is not set to `hash`, the cookie is ignored.

5. Specify the offset value for the cookie.

```
>> Cookie Settings#offset

Current value: 1
Enter starting point of cookie value [1-64]:
```

- If an IP address is embedded in the cookie by the backend server, set the offset length to `1` byte.
- If the backend server is configured to embed a string of characters as the cookie value, the offset value specifies at which point in the string the AVG should start to extract information used for the hashing key.

  Example: For a cookie defined as `sid=012345abcdef` (`sid` being the cookie name, and `012345abcdef` being the cookie value) and the offset value set to `5`, the AVG will use information in the cookie value starting from the fifth byte (the character `4` in this example).

Note:

The offset value (in bytes) must not exceed the byte length of the cookie value.

6. Specify the number of bytes to extract from the cookie value to use as the hashing key.

```
>> Cookie Settings#length

Current value: 8
Enter number of bytes to extract [0-64]:
```

- If an IP address is embedded in the cookie by the backend server, set the `length` value to `8`bytes.
- If the backend server is configured to embed a string of characters as the cookie value, the length value specifies the number of bytes of the string (starting from the point specified by the offset value) the AVG should use as the hashing key.

  Example: For a cookie defined as `sid=012345abcdef`, the offset value set to `5`, and the length value set to `4`, the AVG will use only the characters `45ab` in the cookie name for the hashing key.

Note:

The number of bytes determined by the offset value and the length value must not exceed the length of the cookie value (in bytes) set by the backend server.

7. Verify your cookie configuration.

```
>> Cookie Settings#cur

Collecting data, please wait...

Cookie Settings:

Cookie mode = p<indicates "passive">

Cookie name =<your cookie name match string>

Cookie value offset =<your offset value in bytes>


Cookie value length =<your specified extract length in bytes>
```

8. Apply your settings.

```
>> Cookie Settings#apply
```

```
Changes applied successfully.
```

Example 3: Using Rewrite Cookie Mode

1. Specify the appropriate load balancing metric for the backend servers.

```
>> Main#/cfg/ssl/server 1/adv/loadbalanc/metric

Current value: hash
Enter metric (hash/roundrobin/leastconn):
```

When using the `rewrite` cookie mode of operation, always select `roundrobin` or `leastconn` as the load balancing metric.

- With `roundrobin` as the selected load balancing metric, initial client connection requests are issued to each backend server in turn.
- With `leastconn` as the selected load balancing metric, initial client connection requests are issued to the backend server currently having the fewest open connections.

2. Enable cookie-based persistence on the virtual SSL server.

```
>> Load Balancing Settings#persistenc

Current value: none
Enter persistence (none/cookie/session):cookie
```

3. Specify `rewrite` as the cookie mode of operation.

```
>> Load Balancing Settings#cookie/mode

Cookie mode (insert/passive/rewrite):rewrite
```

4. Specify the cookie name to match.

```
>> Cookie Settings#name

Current value: ISDSSL
Enter cookie name:
```

When using the `rewrite` cookie mode, the cookie name you specify on the AVG is significant to the functionality. When a backend server generates a cookie, the AVG will intercept and act only on those cookies for which a match of the cookie name you specify is found.

You can use the wildcard character * at the end of the name you specify. If more than one cookie matches, only the first one will be used. When a match of the cookie name is found, the AVG will rewrite the cookie value set by the backend server to include either:

- The backend server IP address (if cookie length=8 bytes)
- The backend server IP address and the virtual server IP address (if cookie length=16 bytes).

5. Specify the offset value for the cookie.

```
>> Cookie Settings#offset

Current value: 1
Enter starting point of cookie value [1-64]:
```

With the `rewrite` cookie mode, the offset value specifies at which point in the character string defining the cookie value, the AVG should start rewriting the existing cookie value.

Example: For a cookie defined as `sid=alteonpersistencecookie` (`sid` being the cookie name, and `alteonpersistencecookie` being the cookie value) an offset value of `7` will instruct the AVG to rewrite the existing cookie value starting from the character `p`.

Note:

The offset value (in bytes) must not exceed the byte length of the cookie value.

6. Specify the length value for the cookie.

```
>> Cookie Settings#length

Current value: 8
Enter number of bytes to extract [0-64]:
```

- To include only the backend server IP address, set the cookie length to `8` bytes (thedefault value).

  Example: For a cookie defined as `sid=alteonpersistencecookie`, the offset value set to `7`, and the length value set to `8`, the cookie value would change to the following after the AVG has performed its rewrite operation: `sid=alteoncdb20f04ncecookie`

- To include both the IP addresses of the backend server and the IP address of the Virtual server on the Application Switch, set the cookie length to `16` bytes.

  Example: For a cookie defined as `sid=alteonpersistencecookie`, the offset value set to `7`, and the length value set to 16, the cookie value would

change to the following after the AVG has performed its rewrite operation: `sid=alteoncdb20f04cdb20f0ae` (where the last character e is part of the original cookie value set by the backend server).

Note:

The number of bytes determined by the offset value and the length value must not exceed the length of the cookie value (in bytes) set by the backend server.

7. Verify your cookie configuration.

```
>> Cookie Settings#cur

Collecting data, please wait...

Cookie Settings:

Cookie mode = r<indicates "rewrite">

Cookie name =<your cookie name match string>

Cookie value offset =<your offset value in bytes>

Cookie value length =<your specified extract length of 8 or 16
bytes>
```

8. Apply your settings.

```
>> Cookie Settings#apply

Changes applied successfully.
```

SSL Session-Based Persistence

SSL is a set of protocols built on top of TCP/IP that allows a server application and a client application to communicate over an encrypted connection, providing mechanisms for authentication, non-repudiation, and security.

Using information in the SSL session, the AVG can forward a client connection request to the same backend server to which it was bound during the last session. Because the SSL protocol allows many TCP connections to use the same SSL session from the same client to a backend server, a key exchange need only be performed when the SSL session ID expires. Reusing the same SSL session reduces overhead, and enables the AVG to send all SSL sessions with the same ID to the same backend server.

How SSL Session-Based Persistence Works

Based on information unique for a specific SSL session, the virtual SSL server will forward a client connection request to the same backend server to which it was bound during the last

session. The length of time an SSL session can be reused in order to obtain persistency in client connections depends on the size of the SSL cache size and the SSL timeout value.

Configuring SSL Session-Based Persistency

1. Enable SSL connect towards the backend servers for the virtual SSL server.

```
>> Main#/cfg/ssl/server #/adv/sslconnect

>> SSL Connect Settings#ena

Remember to update the rport settings.
```

   When enabling `sslconnect` for SSL session-based client persistency in load balancing, the `rport` settings correspond to the TCP port numbers on the backend servers you have configured the virtual SSL server to use when initiating requests. For a reminder of this configuration step, see 2 on page 131.

2. Specify the appropriate load balancing metric for the backend servers.

```
>> SSL Connect Settings#../loadbalanc/metric

Current value: hash
Enter metric (hash/roundrobin/leastconn):
```

   With SSL session-based persistency, you can select any of the three available load balancing metrics. For more information about load balancing metrics, see Metrics for Server Load Balancing on page 132 .

3. Enable session-based persistence on the virtual SSL server.

```
>> Load Balancing Settings#persistenc

Current value: none

Enter persistence (none/cookie/session):session
```

4. Apply your changes.

```
>> Load Balancing Settings#apply
```

Chapter 10: String-Based Load Balancing and Blocking

This chapter provides information about creating match strings used to customize the load balancing of backend servers. In the following procedure, many of the backend server load balancing options are left to their default values. See Additional Match String Options on page 158 for more options.

Creating Match Strings

By using the menu options in the Load Balancing Strings menu, you can define up to 128 match strings per virtual server. Match strings are identified by their index number, and these index numbers are assigned to backend servers for string-based load balancing. To effectively perform string-based load balancing, the same match string index number must be assigned to at least two backend servers. When a match of the defined string is found in a client connection request, only those backend servers to which the match string has been assigned are included in the load balancing scheme. The matching client connection request is then issued to one of these backend servers, based on the current load balancing metric (hash, round robin, or least connections).

Note:

If a client connection request does not match any string assigned to any of the backend servers, the client connection is terminated.

The menu options in the Load Balancing Strings menu are also used for creating blocking strings. A blocking string is used for blocking client requests containing data that matches a string of data as defined in the blocking string. If a match is found, the client connection request is terminated. Unlike match strings, specified blocking strings apply to all backend servers configured under a particular virtual SSL server. The blocking feature can be useful in order to protect the backend servers from being infected with the "Code Red" worm, for example.

Match String Configuration Examples

The following is required prior to configuration:

- Load balancing enabled for the virtual SSL server handling the backend servers.
- Backend servers, configured with IP addresses and TCP ports on the AVG.
- The virtual SSL server performing string-based load balancing or blocking must be set to the HTTP type.

Note:

For details about any of the menu commands appearing in these configuration examples, see the *Command Reference* .

Example 1: Creating a Match String for Load Balancing Client Requests for Static Data

1. Select the virtual SSL server, to which the configured backend servers you want to load balance are assigned.

```
# /cfg/ssl/server

Enter virtual server number: (1-)1

>> Server 1#
```

   For a complete listing of all configured virtual SSL servers on the AVG, use the `/info/servers` command.

2. Enter the Load Balancing Strings menu and specify an unused index number.

   By specifying an unused index number, an empty match string placeholder is created. Whenever you want to reconfigure a match string, or assign a match string to a backend server, specify the index number representing the desired match string.

```
>> Server 1#adv/string

Enter string number: (1-)1

Creating LB String 1
>> LB String 1#
```

3. Define a first match string to be used in load balancing.

   The wildcard character "*" can be used for any match string. Only one match string can be defined for each match string index number. Instead, you can assign more than one match string to a backend server by separating the corresponding match string index numbers with a comma (,).

```
>> LB String 1#match

Current value: <not set>
Enter match string (may contain *):*.GIF
```

4. Specify the client request location(s), in which the AVG will search for a match.

```
>> LB String 1#location

Current value:""

Enter match locations (separated by comma):URL
```

The AVG will search for matches only in the locations you specify. The client request connection is issued to a load balanced backend server according to the current metric if a match is found in one or several of the specified locations. For load balancing to occur, it is thus not required that matches be found in all the specified locations.

A match location can be either a header name in a HTTP request (such as `User-Agent`), or an HTTP method (such as `GET`). When specifying `URL` as a match **`location`**, all valid HTTP methods are included in the search which saves you from having to enter each method name explicitly. For more information about valid match locations, see the location command in the "Load Balancing Strings Configuration" section in the *Command Reference* .

5. Define a second match string.

```
>> LB String 1#../string 2

Creating LB String 2
>> LB String 2#
Current value: <not set>
Enter match string (may contain *):*.JPG
```

6. Specify the client request location(s) for the second match string.

```
>> LB String 2#location

Current value:""

Enter match locations (separated by comma):URL
```

7. Assign the match strings to a first backend server.

```
>> LB String 2#../loadbalanc/backend 1

>> Backend Server 1#lbstrings

Current value:""

Enter string numbers (separated by comma):1,2
```

8. Assign the same match strings to a second backend server.

```
>> Backend Server 1#../backend 2

>> Backend Server 2#lbstrings

Current value:""

Enter string numbers (separated by comma):1,2
```

By assigning the match strings to at least two backend servers, the current load balancing metric is applied to those backend servers when a match of the defined strings is found in the specified location of an incoming client connection request.

You can also specify negative string numbers, which indicates that a match of the defined strings must not be found in a client request for the backend servers to be load balanced.

In this configuration example, client connection requests containing the string `*.GIF` or the string `*.JPG` in any HTTP method field will be load balanced among backend server 1 and backend server 2, using the current load balancing metric.

Note:

If you assign a match string(s) to only one backend server, all client connection requests matching the string(s) will be sent to that particular backend server. Load balancing will not take place.

9. Change the load balancing type to string.

```
>> Backend Server 2#../type

Current value: all
Enter load balancing type (all/string):string
```

10. Verify your settings.

```
>> Load Balancing Settings#cur

Collecting data, please wait...
```

11. Apply your changes.

```
>> Load Balancing Settings#apply
```

Example 2: Creating a Match String for Load Balancing Client Requests for Dynamic Data

1. Select the virtual SSL server, to which the configured backend servers you want to load balance using match strings are assigned.

```
# /cfg/ssl/server

Enter virtual server number: (1-)1

>> Server 1#
```

   For a complete listing of all configured virtual SSL servers on your AVG, use the `/info/servers` command.

2. Enter the Load Balancing Strings menu and specify an unused index number.

```
>> Server 1#adv/string

Enter string number: (1-)3

Creating LB String 3
>> LB String 3#
```

   By specifying an unused index number, an empty match string placeholder is created. Whenever you want to reconfigure a match string, or assign a match string to a backend server, specify the index number representing the desired match string.

3. Define a first match string to be used in load balancing.

```
>> LB String 3#match

Current value: <not set>
Enter match string (may contain *):*.cgi
```

   The wildcard character "*" can be used for any match string. However, only one match string can be defined for each match string index number. Instead, you can assign more than one match string to a backend server by separating the corresponding match string index numbers with a comma (,).

4. Specify the client request location(s), in which the AVG will search for a match.

```
>> LB String 3#location
```

```
Current value:""

Enter match locations (separated by comma):URL
```

The AVG will search for matches only in the locations you specify. The client request connection is issued to a load balanced backend server according to the current metric if a match is found in one or several of the specified locations. For load balancing to occur, it is thus not required that matches are found in all the specified locations.

A match location can be either a header name in a HTTP request (such as `User-Agent`), or an HTTP method (such as `GET`). When specifying `URL` as a match location, all valid HTTP methods are included in the search which saves you from having to enter each method name explicitly. For more information about valid match locations, see the **`location`** command in the *Command Reference* .

5. Define a second match string.

```
>> LB String 3#../string 4

Creating LB String 4
>> LB String 4#match

Current value: <not set>
Enter match string (may contain *):*cgi-bin*
```

6. Specify the client request location(s) for the second match string.

```
>> LB String 4#location

Current value:""

Enter match locations (separated by comma):URL
```

7. Assign the match strings to a first backend server.

```
>> LB String 4#../loadbalanc/backend 3

>> Backend Server 3#lbstrings

Current value:""

Enter string numbers (separated by comma):3,4
```

8. Assign the same match strings to a second backend server.

```
>> Backend Server 3#../backend 4
```

```
>> Backend Server 4#lbstrings

Current value:""

Enter string numbers (separated by comma):3,4
```

By assigning the match strings to at least two backend servers, the current load balancing metric is applied to those backend servers when a match of the defined strings is found in the specified location of an incoming client connection request.

In this configuration example, client connection requests containing the string `*.cgi` or the string `*cgi-bin*` in any HTTP method field will be load balanced among backend server 3 and backend server 4, using the current load balancing metric.

Note:

If you assign a match string(s) to only one backend server, all client connection requests matching the string(s) will be sent to that particular backend server. Load balancing will not take place.

9. Change the load balancing type to string.

```
>> Backend Server 4#../type

Current value: all
Enter load balancing type (all/string):string
```

10. Verify your settings.

```
>> Load Balancing Settings#cur

Collecting data, please wait...
```

11. Apply your changes.

```
>> Load Balancing Settings#apply
```

Example 3: Creating a Blocking String for Client Connections Containing the Code Red Worm

The "Code Red" worm exploits a vulnerability on unpatched Microsoft IIS servers that uses the Microsoft Indexing Service. This example provides information on how to intercept and terminate client requests containing the "Code Red" worm before reaching an MS IIS backend

server. Such a solution should always be used in conjunction with the IIS patches made available by Microsoft.

1. Select the virtual SSL server for which you want to configure a blocking string.

```
# /cfg/ssl/server

Enter virtual server number: (1-)1

>> Server 1#
```

For a complete listing of all configured virtual SSL servers on your AVG, use the `/info/servers` command.

2. Enter the Load Balancing Strings menu and specify an unused index number.

```
>> Server 1#adv/string

Enter string number: (1-)5

Creating LB String 5
>> LB String 5#
```

By specifying an unused index number, an empty match string placeholder is created. Whenever you want to reconfigure a match string used for blocking, or assign a match string to a virtual SSL server, specify the index number representing the desired string.

3. Define a first match string to be used for blocking.

```
>> LB String 5#match

Current value: <not set>
Enter match string (may contain *):*default.ida
```

The wildcard character "*" can be used for any match string. However, only one match string can be defined for each match string index number. Instead, you can assign more than one match string to a backend server by separating the corresponding match string index numbers with a comma (,).

Note:

Blocking all client requests matching the string `*default.ida` will also block legitimate requests to a MS IIS server using the Microsoft Indexing Service.

4. Specify the client request location, in which the AVG will search for a match.

```
>> LB String 5#location
```

```
Current value:""

Enter match locations (separated by comma):URL
```

The AVG will search for matches only in the locations you specify. The client request connection is blocked from all configured backend servers if a match is found in one or several of the specified locations. For blocking to occur, it is thus not required that matches are found in all the specified locations.

A match location can be either a header name in a HTTP request (such as `User-Agent`), or an HTTP method (such as `GET`). When specifying `URL` as a match location, all valid HTTP methods are included in the search which saves you from having to enter each method name explicitly. For more information about valid match locations, see the **`location`** command in the *Command Reference* .

5. Define a second match string to be used for blocking the "Code Red" worm

   The string in the following example is a UTF-encoded string (Unicode 16 bit). .

```
>> LB String 5# ../string 6

Creating LB String 6
>> LB string 6#match

Current value: <not set>
Enter match string (may contain *):*
%u6858%ucbd3%u7801%u9090%u685
8%ucbd3%u7801%u9090%u6858%ucbd3%u7801%u9090%u9090%u81
90%u00c3%u0003%u8b00%u531b%u53ff%u0078%u0000%u00=a*
```

6. Specify the client request location, in which the AVG will search for a match.

```
>> LB String 6#location

Current value:""

Enter match locations (separated by comma):Query
```

7. Assign the match strings that the virtual SSL server will use to block client connection requests.

```
>> LB String 6#../blockstrin

Current value:""

Enter string numbers (separated by comma):5,6
```

Any client connection request in which a match of the specified blocking strings are found will be terminated by the virtual SSL server. Unlike match strings used for

load balancing purposes, blocking strings automatically apply to all backend servers configured under the currently selected virtual SSL server.

8. Apply your changes.

```
>> Advanced Settings# apply
```

Additional Match String Options

In the previous section many of the match string options are left to their default values. The following configuration options can be used to customize the match strings used in backend server load balancing on your AVG:

- Ignore case in match
- Negate result in match

Ignore Case in Match

With the `icase` option set to `off`, case is considered when the AVG searches for a match of the defined string in the specified location of a client request. A match string defined as `*.GIF` is therefore considered different data than a `*.gif` string found in the URI of a client request. As a result, a match will not be induced and load balancing of backend servers to which the match string has been assigned will therefore not occur.

The default setting for the `icase` option is `on`, which means that case is ignored in matches.

To change the `icase` setting for a specific match string, use the following command at any menu prompt:

```
>> #/cfg/ssl/server #/adv/string #/icase

Current value: on
Ignore case in match (on/off):
```

Negate Result in Match

With the `negate` option set to `on`, all client requests that do not contain the defined match string in the specified location will induce a match. As a result, load balancing of backend servers to which the negated match string has been assigned will occur.

The default setting for the `negate` option is `off`.

To change the `negate` setting for a specific match string, use the following command at any menu prompt:

```
>> #/cfg/ssl/server #/adv/string #/negate

Current value: off
Negate match (on/off):
```

Additional String Load Balancing Options

When assigning match strings to backend servers for load balancing purposes, the following configuration options can be used to customize the load balancing operation involving the assigned match strings:

- `any`

  Specifies that a match of one or more of the load balancing strings assigned to the current backend server must be found in a client request for load balancing to occur. This is the default string load balancing operation setting.

- `all`

  Specifies that matches of all load balancing strings assigned to the current backend server must be found in a client request for load balancing to occur.

- `one`

  Specifies that a match of one, and only one, of the load balancing strings assigned to the current backend server must be found in a client request for load balancing to occur.

- `none`

  Specifies that no match of any of the load balancing strings assigned to the current backend server must be found in a client request for load balancing to occur. If a match is found, the backend server is not included in the load balancing scheme.

The default string load balancing operation setting is

`any`

.

To change the string load balancing operation setting for a specific backend server, use the following command at any menu prompt:

```
>> #/cfg/ssl/server #/adv/loadbalanc/backend #/lbop

Current value: any
Select one of any, all, one or none:
```

Chapter 11: Script-Based Health Checks

If you have enabled load balancing of the backend servers for a virtual SSL server, you also have the option to select a health check type. The load balancing decisions will in part be based on the outcome of the backend server health checks, which are performed at regular intervals (the default health check interval is 10 seconds).

The default health check type is **`auto`**, which uses a built-in preconfigured script. For more information about the built-in scripts used with **`auto`** health checking, see Built-In Health Check Scripts on page 172. Other available health check types include **`tcp, ssl`** and **`script`** (besides the **`none`** option). The health check type is selected using the following command:

```
/cfg/ssl/server #/adv/loadbalanc/health
```

For more information about available health check types, see the "Load Balancing Settings" section in the *Command Reference* .

If you select **`script`** as the health check type, it is assumed that you have created a customized script by using the script editing functions available in the Health Check Script menu. This chapter provides instructions and examples for creating your own customized health check scripts.

The script health check mechanism is based around a script language which can establish both TCP and SSL connections to the backend servers, and send and expect data on those connections. If a single script item in a health check script fails, the entire script fails and the backend server is reported as down.

Customized Health Check Scripts

The Health Check Script menu is used for creating customized health check scripts. The script editing functions available in the Health Check menu consist of a number of script commands. Timeout values can be set individually for each script command. Arguments can also be defined for a few script commands.

The Health Check Script menu is accessed by using the following command:

```
/cfg/ssl/server #/adv/loadbalanc/script
```

Script Commands

A timeout value can be set for each script command. The timeout value defines the maximum number of seconds in which the particular operation must complete for the health check to be considered successful.

Arguments can be defined for the **open**, **send**, and **expect** script commands. The argument associates a data field to one of these commands. The argument can consist of an IP address and TCP port number combination (**open** command), a HTTP GET request (**send** command), or an extended POSIX regular expression (**expect** command).

The following script commands and parameters are available:

- **auto_open** <*timeout*>

  This command will first perform a check to see whether SSL connect is enabled on the virtual SSL server for which you have set the health check type to **script**.

  If SSL connect is not enabled on the virtual SSL server, the command will open a TCP connection to each of the load balanced backend server.

  If SSL connect is enabled on the virtual SSL server, the command will perform a second check to see whether SSL connect is enabled on each of the backend servers included in the load balancing configuration. For those backend servers on which SSL connect is enabled, the **auto_open** command first opens a TCP connection and then establishes a full SSL session. For those backend servers on which SSL connect is not enabled, the **auto_open** command will only open a TCP connection.

  When creating a customized health-check script, you are recommended to use the **auto_open** (and corresponding **auto_close**) script command in a configuration where the SSL connect setting of the backend servers is not uniform. Given these conditions, using the **auto_open** and **auto_close** script commands will ensure the highest level of accuracy in the health checking of backend servers.

- **open** <*timeout*> <*argument (IP Address:Port)*>

  This command opens a TCP connection to the load balanced backend servers, or to the IP address specified in the argument. The timeout value (in seconds) defines the time frame for establishing the TCP connection. The argument is optional, and if no argument is specified the backend servers specified in the Backend Servers menu (`cfg/ssl/server #/adv/loadbalanc/backend`) are implicit. If an argument is specified, the argument must be stated as IP ADDRESS:PORT. Example: 192.168.0.4:443

- **ssl_open** <*timeout*>

  This command establishes a full SSL session to the load balanced backend servers, or to the IP address specified in the argument for the preceding **open** script command. Note that the **ssl_open** command cannot be used independently; it must be preceded by the regular **open** script command. (Likewise, the **ssl_close** command must be followed by the regular **close** script command.) The SSL parameters used for the SSL session are the same as the ones specified in the SSL Connect Settings menu.

  The timeout value (in seconds) defines the time frame for establishing the SSL connection. If the backend servers are under a heavy load this may take a considerable amount of time. The argument field is ignored for the **ssl_open** script command because the command must be preceded by the regular **open** command. If the preceding **open**

command is used together with an argument, the SSL session is established to the IP address and port that was specified in that argument.

- **`send`** *<timeout> <argument (HTTP request)>*

  This command sends data on the established connection, regardless of whether the connection is an ordinary TCP connection or an SSL connection. If the connection is an SSL connection, all transmitted data is encrypted. See Script Example 2 on Script Example 2: Health Checking a Server by Establishing an SSL Connection on page 169 for an example argument that can be used with the **`send`** script command.

- **`expect`** *<timeout> <argument (extended POSIX regular expressions)>*

  This command reads data from the established connection and matches the received data against the extended POSIX regular expression given as an argument. If the established connection is an SSL connection, the received data is decrypted. The **`expect`** script command is often preceded by the **`send`** script command, but as Script Example 1 on Script Example 1: Health Checking an Auxiliary Server on page 164 shows this is not a requirement. For more information about defining an argument for the **`expect`** command, see Extended POSIX Regular Expressions on page 164 .

- **`ssl_close`** *<timeout>*

  This command performs an SSL shutdown on the most recent SSL connection that was established using the **`ssl_open`** script command.

- **`close`** *<timeout>*

  This command terminates the most recent TCP connection that was established using the **`open`** script command. Both the **`close`** command and the **`ssl_close`** command operate on the most recently opened connection. Only one connection at a time can be open within a health check script. However, the script may contain more than one consecutive series of open and close sequences (as illustrated in Script Example 1 on Script Example 1: Health Checking an Auxiliary Server on page 164 .

- **`auto_close`** *<timeout>*

  This command will first perform an SSL shutdown on the most recent SSL connection, if established, and then continue with terminating the most recent TCP connection. Just as with the corresponding **`auto_open`** script command, you are recommended to use the **`auto_close`** command in a configuration where the SSL connect setting of the backend servers is not uniform. Given these conditions, using the **`auto_close`** and **`auto_open`** script commands will ensure the highest level of accuracy in the health checking of backend servers.

Extended POSIX Regular Expressions

Extended POSIX regular expressions are used as arguments for the **`expect`** command. In the following table, the meaning of the individual elements that constitute the complete expressions used in the script examples are detailed:

`^`	Beginning of line, we don't want to accept any message that contains the data inside the actual body.
`$`	End of line.
`HTTP/1`	Explicit character match.
`\.`	Explicit match for the dot (".") character. The dot character must be back-quoted by the backslash character since it is a special character.
`\?`	Explicit match for the question mark ("?") character. The question mark character must be back-quoted by the backslash character since it is a special character.
`[1,0]`	Either one of the character 1 or 0.
`+`	Space-plus, indicating one or several spaces.
`200`	Explicit character match.

Several descriptions of the syntax and semantics of extended POSIX regular expressions can be found on the Internet.

Script Configuration Examples

In order to effectively use script-based health checking, you also need to perform these tasks:

- Enable load balancing on the virtual SSL server through which the backend servers will be load balanced.
- Set the health check type to `script` if you want to use a customized health check script. If you want to make use of the built-in health check scripts, set the health check type to `tcp`, `ssl`, or `auto`.
- Add and configure backend servers to the load balancing configuration.

Script Example 1: Health Checking an Auxiliary Server

This example shows how an auxiliary server (a POP3 server in this case) can be health checked by the AVG, using a customized script that also includes extended POSIX regular

expressions. The health checking of the auxiliary server is performed alongside the regular health checking of the load balanced backend servers.

1. Type the following command (server 1 is used only as an example).

```
>> Main#/cfg/ssl/server 1/adv/loadbalanc/script
```

   The Health Check Script Menu is now displayed.

```
[Health Check Script Menu]
list       - List all values
del        - Delete a value by number
add        - Add a new value
insert     - Insert a new value
move       - Move a value by number
```

2. Add the first script command, including a timeout value.

   This sequence creates the first line in the script and opens a TCP connection to the load balanced backend servers. You only need to specify an IP address as an argument to the **open** command if you want to perform health checking on another machine than the backend servers. When no IP address is specified, the backend servers included in the load balancing configuration are always implied.

   The **open** script command must be able to complete within the timeout value of 8 seconds. If not, the complete health check is considered as having failed, and the backend servers are reported as down.

```
>> Health Check Script#add

Enter script command:open

Timeout in seconds:8

Argument:
```

3. Add a script command for establishing an SSL connection to the load balanced backend Web servers.

   This sequence creates the second line in the script, and opens an SSL connection towards the load balanced backend servers.

   Since the backend Web servers must be capable of handling SSL connections for end to end encryption purposes, it may be appropriate to include the **ssl_open** script command in the backend server health check.

```
>> Health Check Script#add

Enter script command:ssl_open
```

```
Timeout in seconds:8

Argument:
```

Note:

In order to use the **ssl_open** script command, the virtual SSL server must be SSL-enabled. You can verify whether this is the case by using the `/cfg/ssl/server #/ssl/cur` command. To enable SSL on the desired virtual SSL server, use the `/cfg/ssl/server #/ssl/ena` command.

4. Conclude the **ssl_open** command by adding a **ssl_close** command to the health script.

   This sequence creates the third line in the customized health check script, and closes the SSL connection that was opened by the previous line in the script.

```
>> Health Check Script#add

Enter script command:ssl_close

Timeout in seconds:8

Argument:
```

5. Conclude the initial **open** command by adding a final **close** command.

   This sequence creates the fourth line in the health check script, and closes the TCP connection that was opened by the **open** command in the first line in the script.

```
>> Health Check Script#add

Enter script command:close

Timeout in seconds:8

Argument:
```

6. Add a command to health check a POP3 server.

   This sequence creates the fifth line in the script and contains the **open** script command, combined with a timeout value and an argument. The **open** script command opens a TCP connection to the IP address and port number given as an argument. In this example, the IP address corresponds to a POP3 server on the same network segment as the backend servers.

   The timeout value specifies the time frame for the command to complete. If the command cannot complete within the specified time frame, the health check is considered having failed.

```
>> Health Check Script#add

Enter script command:open

Timeout in seconds:8

 Argument:192.168.128.88:110

(IP address of POP3 server:port number)
```

Note:

When stating an IP address as an argument for the **open** or **ssl_open** script commands, you should also specify which TCP port that should be used for the request.

7. Specify what reply to expect from the POP3 server in order to consider the health check successful.

   This sequence creates the sixth line in the script and contains the **expect** script command. The **expect** command reads data from the established connection and matches the received data against the regular expression that is specified in the argument. In this example, the data received from the POP3 server must contain the text " +OK " at the beginning of a line.

```
>> Health Check Script#add

Enter script command:expect

Timeout in seconds:8

Argument:^\+OK
```

8. Conclude the most recent **open** script command by adding a final **close** command to the health check script.

   This sequence creates the seventh and final line in the health check script. Whenever you have added an **open** script command, it must be accompanied by a **close** command (although they need not be on adjacent lines as you can see in the examples).

   The IP address specified as an argument for the previous **open** command does not have to be repeated for the **close** command, since the same IP address is implicit when the **close** command is performed.

```
>> Health Check Script#add

Enter script command:close

Timeout in seconds:2
```

```
Argument:
```

9. View the entire script by using the **`list`** command.

```
>> Health Check Script#list

Old:
Pending:
1: open 8s ""
2: ssl_open 8s ""
3: ssl_close 8s ""
4: close 8s ""
5: open 8s 192.168.128.88:110
6: expect 8s "^\+OK"
7: close 2s ""
```

Seven commands have now been added to the customized script. These commands will perform both a health check of the backend servers and an additional POP3 server.

Note:

Each line in the script is represented by an index number. All lines combined form one complete script. The commands in the script are executed in the order indicated by the index numbers. Lines can be moved up or down in the script by using the **`move`** command.

10. Configure the health check type to use the customized script.

```
>> Health Check Script#../health

Current value: auto
Enter health check type (none/tcp/ssl/auto/script):script
```

11. Apply your changes.

```
>> Load Balancing Settings#apply

Changes applied successfully.
```

The status of backend server health checks performed by virtual SSL servers can be viewed by using the `/stats/local/dump` command.

Script Example 2: Health Checking a Server by Establishing an SSL Connection

This health check script example performs an SSL-based health check of a backend Web server, and includes sending a request to the Web server. The received data is then matched against a regular expression that is specified in the argument for the **expect** command.

1. Type the following command (where server 1 is used only as an example).

```
>> Main#/cfg/ssl/server 1/adv/loadbalanc/script
```

   The Health Check Script Menu is now displayed.

```
[Health Check Script Menu]
list       - List all values
del        - Delete a value by number
add        - Add a new value
insert     - Insert a new value
move       - Move a value by number
```

2. Add the first script command.

   This sequence creates the first line in the script and opens a TCP connection to the load balanced backend servers. You only need to specify an IP address as an argument to the **open** command in case you want to perform health checking on another machine than the backend servers. When no IP address is specified, the backend servers included in the load balancing configuration are always implied.

   The **open** script command must be able to complete within the timeout value of 8 seconds. If not, the complete health check is considered as having failed, and the backend servers are reported as down.

```
>> Health Check Script#add

Enter script command:open

Timeout in seconds:8

Argument:
```

3. Add a script command for establishing an SSL connection to the load balanced backend Web servers.

   This sequence creates the second line in the script, and opens an SSL connection towards the load balanced backend servers.

```
>> Health Check Script#add

Enter script command:ssl_open

Timeout in seconds:8

Argument:
```

4. Add a script command for sending a request to the load balanced Web servers.

   This sequence creates the third line in the script, and sends the HTTP request specified in the argument to the Web server.

```
>> Health Check Script#add

Enter script command:send

Timeout in seconds:8

Argument:GET /checkdb.php\? HTTP/1.0\r\n\r\n
```

5. Specify what reply to expect from the request that was sent.

   This sequence creates the fourth line in the script and specifies that the received data from the backend Web servers must contain the text "PHP database is running" on a line of its own. If the received data does not contain the specified text, the health check is considered having failed.

```
>> Health Check Script#add

Enter script command:expect

Timeout in seconds:8

Argument:^PHP database is running$
```

6. Conclude the **ssl_open** command by adding the **ssl_close** command to the health script.

   This sequence creates the fifth line in the script, and closes the SSL connection that was opened by the **ssl_open** script command on the second line.

```
>> Health Check Script#add

Enter script command:ssl_close

Timeout in seconds:8

Argument:
```

7. Conclude the initial **open** script command by adding a final **close** command to the health check script.

   This sequence creates the sixth and final line in the health check script, and closes the TCP connection to the Web server that was opened by the initial **open** script command.

```
>> Health Check Script#add

Enter script command:close

Timeout in seconds:4

Argument:
```

8. View the entire script by using the **list** command.

```
>> Health Check Script# list

Old:
Pending:
1: open 8s ""
2: ssl_open 8s ""
3: send 8s "GET /checkdb.php\? HTTP/1.0 \r\n\r\n"
4: expect 8s "^PHP database is running$"
5: ssl_close 8s ""
6: close 4s ""
```

Note:

Each line in the script is represented by an index number. All lines combined form one complete script. The commands in the script are executed in the order indicated by the index numbers. Lines can be moved up or down in the script by using the **move** command.

9. Configure the health check type to use the customized script.

```
>> Health Check Script#../health

Current value: auto
Enter health check type (none/tcp/ssl/auto/script):script
```

10. Apply your changes.

```
>> Load Balancing Settings#apply

Changes applied successfully.
```

Built-In Health Check Scripts

When using the `tcp`, `ssl`, or `auto` health check options, a built-in script is used for health checking backend servers that have been configured for load balancing by using the Backend Servers menu (`/cfg/ssl/server #/adv/loadbalanc/backend`).

Depending on your settings for the virtual SSL server, one out of four different built-in health check scripts is selected and applied automatically. The four scripts and related virtual SSL server settings are listed following section in order of complexity.

Script 1

Script 1 contains the following script items:

```
open   8
close  2
```

This is a basic health check script that opens a TCP connection to the backend servers, and then closes the connection. If the TCP connection cannot be opened or closed within the specified time frames, the health check fails and the backend servers are reported as being down.

Script 1 is used when:

- The virtual SSL server type is set to **`generic`**.

  Command path in the CLI: `/cfg/ssl/server #/type generic`

- SSL connect is **`disabled`** on the virtual SSL server.

  Command path in the CLI: `/cfg/ssl/server #/adv/sslconnect/dis`

- The health check option is set to either **`tcp`** or **`auto`**.

  Command path in the CLI: `/cfg/ssl/server #/adv/loadbalanc/health tcp|auto`

Script 2

Script 2 contains the following script items:

```
open        8
ssl_open    8
ssl_close   8
close       2
```

This health check script first opens a TCP connection to the backend servers, then establishes a full SSL connection to the same backend servers. Thereafter, the SSL connection is shut down, and the TCP connection is closed.

Script 2 is used when:

- The virtual SSL server type is set to **`generic`**.

  Command path in the CLI: `/cfg/ssl/server #/type generic`

- SSL connect is **`enabled`** on the virtual SSL server.

  Command path in the CLI: `/cfg/ssl/server #/adv/sslconnect/ena`

- The health check option is set to either **`ssl`** or **`auto`**.

  Command path in the CLI: `/cfg/ssl/server #/adv/loadbalanc/health ssl|auto`

- SSL connect is set to **`on`** for a backend server load balanced by the virtual SSL server.

  Command path in the CLI: `/cfg/ssl/server #/adv/loadbalanc/backend #/sslconnect on`

  Note:

  If SSL connect is set to `off` for a particular backend server, all other settings being the same as above, that particular backend server will automatically receive Script 1 instead of Script 2 due to an auto-sensing feature in the health checking method.

Script 3

Script 3 contains the following script items:

```
open     8
send     8      "GET / HTTP/1.0 \r\n\r\n"
expect   8      "^HTTP/1\.[1,0] +200"
close    2
```

This health check script first opens a TCP connection to the backend servers, then sends a HTTP " `GET` " request. For the health check to succeed, the reply from the backend servers must contain the expected string " `HTTP 1.0 +200` ", or " `HTTP 1.1 +200` " at the beginning of a line. After having confirmed that the reply contains the expected string, the TCP connection is closed.

Script 3 is used when:

- The virtual SSL server type is set to **`http`**.

  Command path in the CLI: `/cfg/ssl/server #/type http`

- SSL connect is **`disabled`** on the virtual SSL server.

Command path in the CLI: `/cfg/ssl/server #/adv/sslconnect/dis`

- The health check option is set to either **`tcp`** or **`auto`**.

  Command path in the CLI: `/cfg/ssl/server #/adv/loadbalanc/health tcp|auto`

Script 4

Script 4 contains the following script items:

```
open       8
ssl_open   8
send       8      "GET / HTTP/1.0 \r\n\r\n"
expect     8      "^HTTP/1\.[1,0] +200"
ssl_close  8
close      2
```

This health check script first opens a TCP connection to the backend servers, then establishes a full SSL connection. Thereafter, a HTTP " `GET` " request is sent to the backend servers, and the reply is matched against the expected string. For the health check to succeed, the reply from the backend servers must contain the string " `HTTP 1.0 +200` ", or " `HTTP 1.1 +200` "at the beginning of a line. The SSL connection is then shut down, where after the TCP connection is closed.

Script 4 is used when:

- The virtual SSL server type is set to **`http`**.

  Command path in the CLI: `/cfg/ssl/server #/type http`

- SSL connect is **`enabled`** on the virtual SSL server.

  Command path in the CLI: `/cfg/ssl/server #/adv/sslconnect/ena`

- The health check option is set to either **`ssl`** or **`auto`**.

  Command path in the CLI: `/cfg/ssl/server #/adv/loadbalanc/health ssl|auto`

- SSL connect is set to **`on`** for a backend server load balanced by the virtual SSL server.

  Command path in the CLI: `/cfg/ssl/server #/adv/loadbalanc/backend #/sslconnect on`

Note:

If SSL connect is set to **`off`** for a particular backend server, and all other settings are the same as above, that particular backend server will automatically receive Script 3 instead of Script 4 due to an auto-sensing feature in the health checking method.

Verifying Script-Based Health Checks

To examine the outcome of health checks performed on all backend servers, you can use the `/stats/local/dump` command. If you have configured the VPN Gateway(s) to use a syslog server, changes in the health check status of backend servers can also be sent as syslog messages to the specified syslog server.

To examine the outcome of health checks performed by a specific AVG host only on those backend servers that are load balanced by a specific virtual SSL server, use the `/stats/local/isdhost #/server #/healthchec` command.

In case there are two load balanced backend servers and one passed the health check but not the other, the `healthchec` command will produce a screen output similar to the following example.

```
>> Single ISD Stats for Server 1#healthchec

Healthcheck status at ISD number '1'
BE      RIP                          UP         EXEC   FAILS   REASON
1       192.168.128.50:80     up      no
2       192.168.128.51:80     down   no      2         Can't connect:
Connection refused
```

Explanations of the health check status properties found in the screen output example:

- BE: Backend server index number.
- RIP: Real Server IP address and TCP port of backend servers being health checked.
- UP: Operational status (`up` or `down`) of backend server, determined by the health check.
- EXEC: Executing (`yes` or `no`). Provides information about whether the health check is currently being performed on the backend server.
- FAILS: Indicates the number of times the health check has failed on the backend server.
- REASON: States the reason to why the health check failed on the backend server.

Chapter 12: Stand-Alone Web Server Accelerator

In the typical network setup, an AvayaVPN Gateway (AVG) requires support of an Application Switch for traffic redirection. For small, low volume sites however, it is normally not economically feasible nor necessary to have both VPN Gateways and an Application Switch. To use the VPN Gateway in scenarios without an Application Switch, you should configure the VPN Gateway in "stand-alone" mode.

IP Address Migration

When using the VPN Gateway in conjunction with an Application Switch, the Application Switch holds the virtual server IP address (VIP), and redirects traffic to the AVG cluster. In stand-alone mode, the AVG cluster will hold the VIP and perform address resolution of the VIP. In a cluster of several AVG devices, it is not possible to do this using one VIP address common to all AVGs. In stand-alone mode, the virtual SSL server will therefore be bound to several addresses specified in an address list for each virtual SSL server. The address list should contain as many IP addresses as there are VPN Gateways in the cluster. Client requests will be redirected to one of the VPN Gateways by means of round robin load balancing. In other words, the SSL server address list replaces the function of the VIP used in a setup with an Application Switch. When stand-alone mode is enabled, the SSL server addresses in the list will be distributed among the VPN Gateways. If one or several VPN Gateways fail, the SSL server addresses currently held by these AVGs will migrate to functional VPN Gateways. This means that as long as there is one functional VPN Gateway in the cluster, all of the SSL server addresses in the list will be available.

DNS Round Robin Load Balancing

To distribute client traffic evenly between the VPN Gateways in the cluster, you should specify as many IP addresses in the address list as there are AVGs in the cluster. In the DNS server configuration, specify that the domain name assigned to the SSL server resolves to the IP addresses configured in the IP address list. Also configure the DNS server to perform round robin load balancing and reverse DNS lookups. DNS lookups for the domain name will then resolve to the IP list addresses in a round robin manner, with the effect that client requests will be directed to the AVGs on a turn basis.

Backend Server Load Balancing

If you have more than one real Web server (Backend server) in your network, you can configure the VPN Gateway to perform load balancing of the real Web servers as well. To add load balancing to the regular task of Web server acceleration in this example, see Add Load Balancing Support on page 184 .

The following illustration shows two VPN Gateways in a cluster where two IP addresses are added to the IP list of an SSL server in the AVG cluster. The IP addresses shown next to the AVGs are in fact not bound to any specific VPN Gateway, but could rather be considered as floating IP addresses.

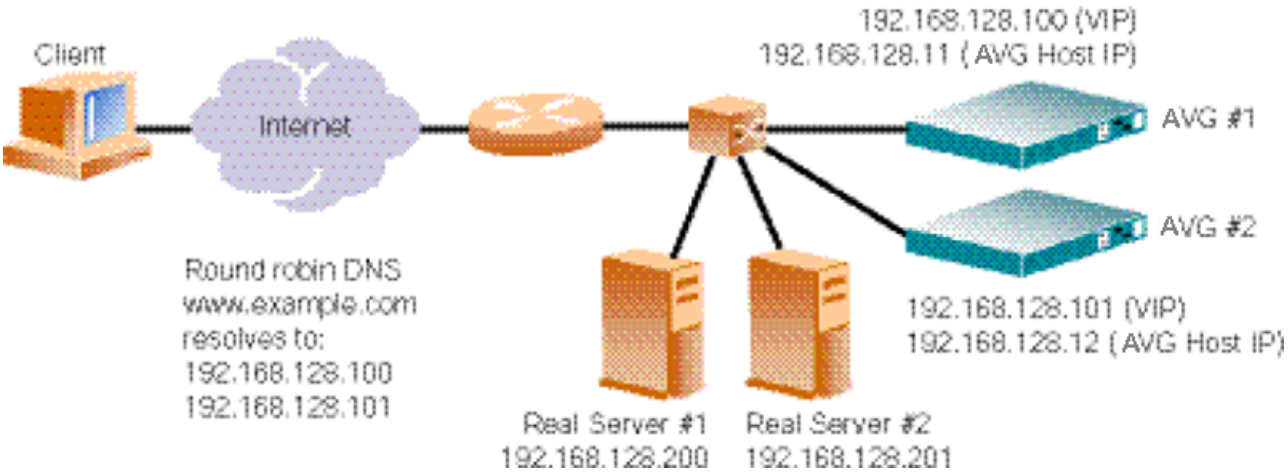

Figure 10: Two Stand-alone AVGs with Round Robin Load Balancing

If a VPN Gateway would fail, the IP addresses in the IP address list currently bound to the AVG will be migrated to the functional AVG.

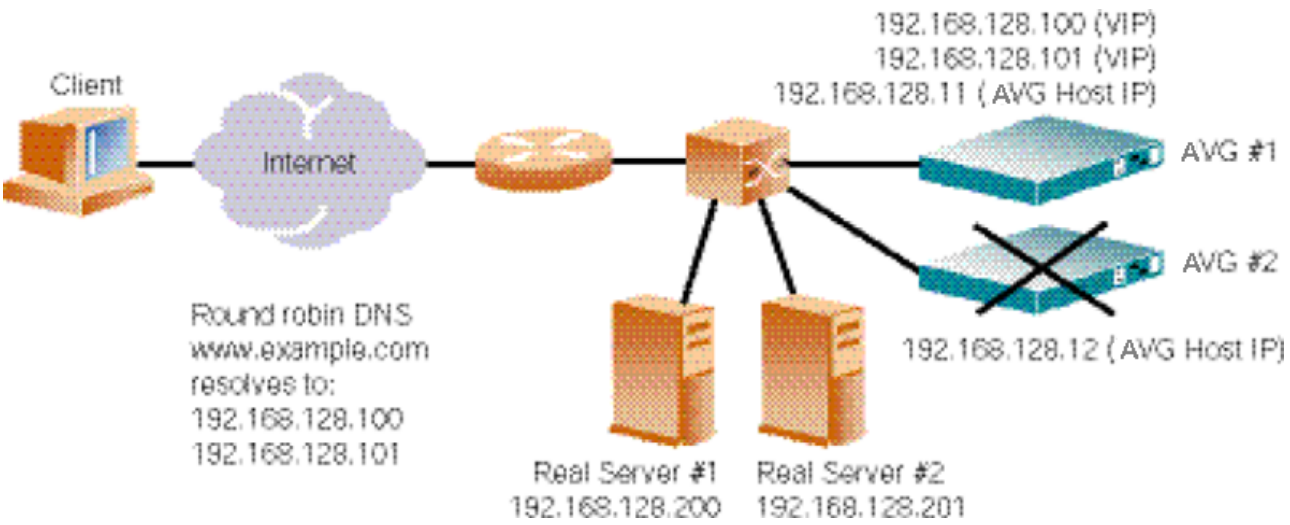

Figure 11: IP Address Migration to Functional VPN Gateway

Configuration Example

Initial Setup

This configuration example assumes that you have performed the initial setup of the AVG as described in the "Initial Setup" chapter in the *Users Guide* .

After having performed the initial setup, connect to the AVG cluster to add a certificate and configure the parameters.

Add a Server Certificate

This step presumes that you have a server certificate, signed by a certificate authority (CA), and a private key. The process for obtaining the required certificate files is covered in the "Certificates and Client Authentication" chapter in the *Users Guide* .

Once you have the appropriate certificate, use the following procedure to add the certificate to the AVG.

```
# /cfg/cert

Enter certificate number: (1-)1
```

```
Creating Certificate 1
>> Certificate 1#cert

Paste the certificate, press Enter to create a new line, and then type
"..."(without the quotation marks) to terminate.
```

The preceding example assumes that the certificate signing request (CSR) was generated from certificate number 1, which implies that the private key that corresponds to the public key in the certificate is already in place.

When prompted for the certificate, follow the instructions. Use Notepad or any other text editor to display the certificate. Then copy and paste the text of the certificate into the terminal window. For more detailed information about how to add certificates and keys to the AVG, see the "Certificates and Client Authentication" chapter in the *Users Guide* .

Important:

Once you have pasted the entire contents of the certificate file, press ENTER to create a new empty line and then type three periods (...). Press ENTER again to complete the installation of the certificate.

Configure the Virtual SSL Server Parameters

Connect through Telnet or SSH to the management IP address (MIP) of the AVG.

1. Create a virtual SSL server.

   This step creates a new virtual SSL server on the AVG. Each virtual SSL server listens to a specific TCP port, and is mapped to a virtual server IP address.

```
# /cfg/ssl/server

Enter virtual server number: (1-)1

Creating Server 1
>> Server 1#
```

2. Define a name for virtual SSL server 1.

   This step lets you specify a name, by which you can identify SSL server 1. The name you specify is mainly intended for your own reference, and is not critical for the configuration itself. As the preceding example suggests, the name can indicate the service for which the virtual SSL server is created.

```
>> Server 1#name

Current value:""
```

```
Enter new SSL server name:HTTPS
```

3. Set the server type to HTTP.

```
>> Server 1#type

Current value: generic
Type (generic/http/socks):http
```

4. Disable transparent proxy mode.

   Transparent proxy mode is supported only when the AVG is used together with an Application Switch that has been configured with the appropriate filters and load balancing algorithm. (For a Web server accelerator configuration example that includes using an Avaya Application Switch, see Web Server Accelerator on page 21).

```
>> Server 1#proxy

Current value: on
Proxy mode (on/off):off
```

5. Set the listen TCP port for SSL server 1.

   Enter the listen port number 443, which is the default value used for HTTPS connections.

```
>> Server 1#port

Current value: <not set> [443 (https)]
Enter listen port number (1-65534):443
```

Configure IP Address Migration

To enable IP address migration, you should enable stand-alone mode for the server you have just created and add the desired IP addresses to the IP address list. As long as at least one of the VPN Gateways is up and running, all of these IP addresses will be available.

When a new VPN Gateway joins the cluster, or an existing member of the cluster becomes functional again, some IP addresses are migrated over to that new machine. It is not possible to explicitly control from the CLI which IP addresses end up where in the cluster. The addresses are assigned automatically.

Load balancing of the VPN Gateways will be available if the DNS server is configured for round robin DNS and the domain name assigned to the virtual SSL server resolves to the VIPs configured in the IP lists.

Note:

Each VPN Gateway retains its own host IP address (in the diagrams called AVG Host IP), which is configured during the Initial Setup as described in the *Users Guide*. The host IP address for each VPN Gateway is fixed, and does not participate in IP address migration.

1. Enable stand-alone mode for the SSL server (Server 1), and define the virtual server IP addresses.

   This step defines the virtual server IP addresses on which client connection requests destined for port 443 are received. Enter as many IP addresses as there are VPN Gateways in the cluster.

   ```
   # /cfg/ssl/server 1/standalone

   Current value: off
   Standalone mode (on/off):on

   >> Server 1#vips

   Current value:""

   Enter server ips (comma separated):
   192.168.128.100,192.168.128.101
   ```

   The IP addresses configured in the preceding step correspond to the addresses found in the DNS server's A Record.

2. Apply the changes.

   ```
   >> Server 1#apply
   ```

3. If only one real server is used, set the Real Server IP address that SSL server 1 should connect to, when it initiates requests.

   This step instructs the AVG to initiate requests to the IP address of the real Web server.

   ```
   >> Server 1#rip

   Current value: 0.0.0.0
   Enter IP address to connect to:192.168.128.200
   ```

   (real Web server IP address)

   Note:

   If you want the AVG to perform load balancing of multiple real Web servers, retain the default `0.0.0.0` Real Server IP value.

4. Set the server port to which SSL server 1 should connect when initiating requests.

```
>> Server 1#rport

Current value: 81
Enter port to connect to:80
```

When using the AVG together with an Application Switch for Web server accelerating purposes, the `rport` value is normally set to TCP port 81. In that case, normal HTTP traffic uses TCP port 80, while HTTPS requests are redirected to the VPN Gateway(s) by the Application Switch. The AVG(s) then use TCP port 81 to send and receive decrypted HTTP information to and from the real Web servers (through the Application Switch).

When using the AVG as a stand-alone device, however, you are restricted to using HTTPS for all requests to the Web server. Therefore, you can use the standard TCP port 80, which does not require you to reconfigure the real Web server.

Note:

If you want the AVG to perform load balancing of multiple real Web servers, you do not need to perform this step. When including real Web servers in the load balancing configuration, you will specify each server by IP address and TCP port.

5. Specify the certificate to be used by SSL Server 1.

You are prompted to type the index number of an existing certificate. To view all certificates currently added to the AVG by index number and name, use the `/cfg/cur cert` command. For more information about how to add a certificate, see the "Certificates and Client Authentication" chapter in the *Users Guide*

```
>> Server 1#ssl

>> SSL Settings#cert

Current value:<not set>

Enter certificate number: (1-)1
```

Note:

If the certificate you specify is a chained certificate, you need to first add the CA certificates up to and including the root CA certificate, and then specify the CA certificate chain of the server certificate. For more information on how to construct

the server certificate chain, see the **cachain** command under "SSL Server SSL Configuration" in the *Command Reference*.

6. Apply the changes.

```
>> SSL Settings#apply

Changes applied successfully.
```

Add Load Balancing Support

If you have more than one real (backend) Web server, you can configure the AVG to perform load balancing of the backend servers. Load balancing of backend servers requires a few additional steps to complete the stand-alone Web server accelerator configuration preceding example.

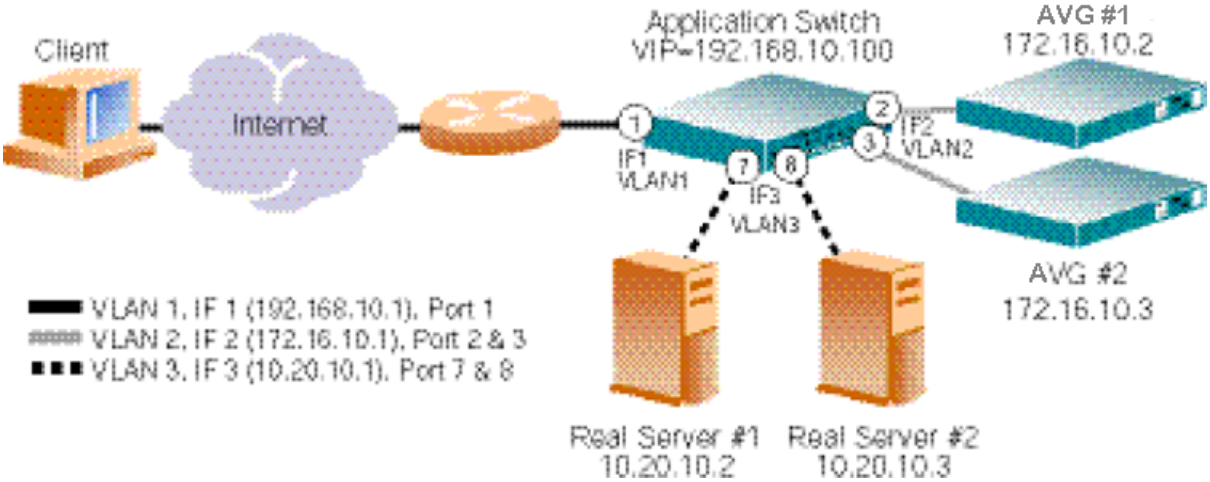

Figure 12: Adding Backend Server Load Balancing to the Stand-alone Configuration

To add support for load balancing, continue from 6 on page 184 on 6 on page 184 and follow the instructions.

1. The default value of 0.0.0.0 for the Real Server IP setting in 3 on page 182 on 3 on page 182 should be retained.

   If the current value corresponds to `0.0.0.0`, press ENTER to retain the default value. If another Real Server IP address is shown, enter `0.0.0.0` and press ENTER.

```
>> SSL Settings#../ rip
```

```
Current value: 0.0.0.0
Enter IP address to connect to:
```

2. Define each backend (real) server by specifying IP address and TCP port.

   The virtual SSL server will initiate requests to one of the specified backend servers, based on the default `hash` load balancing metric. For more information about available load balancing options, see Load Balancing of Backend Servers on page 131 .

```
>> Server 1#adv/loadbalanc/backend

Enter backend server number: (1-)1

Creating Backend Server 1
>> Backend Server 1#ip 192.168.128.200

>> Backend Server 1#port 80

>> Backend Server 1#../backend 2

>> Backend Server 2#ip 192.168.128.201

>> Backend Server 2#port 80
```

   Note:

   The backend servers must also be configured to listen for incoming requests on the same TCP ports as you specify in the load balancing configuration on the VPN Gateway.

3. Enable load balancing of the configured backend servers and verify the configuration.

```
>> Backend Server 2#../ena

>> Load Balancing Settings#cur
```

4. Apply your changes.

```
>> Load Balancing Settings#apply
```

Directing Traffic to Different SSL Servers

To configure several SSL servers on the AVG in stand-alone mode, for example, to enable access to different sites, add the required virtual server IP addresses to each SSL server's IP list, as described earlier in this chapter. For each SSL server that you have defined, enable

stand-alone mode and add the desired virtual server IP addresses to the IP list. In addition, you should configure the DNS server to perform round robin DNS lookups on the desired IP addresses and domain names.

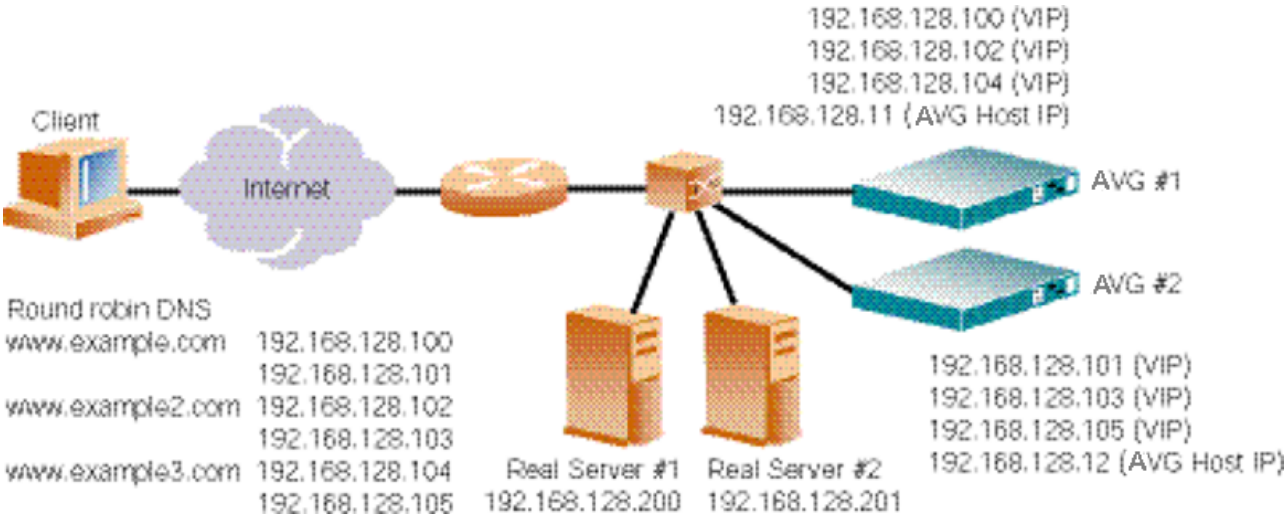

Figure 13: Three Different SSL Servers in a Redundant Stand-alone Solution

With the setup in Figure 13: Three Different SSL Servers in a Redundant Stand-alone Solution on page 186 , users can access three different virtual SSL servers, using three different domain names. If one VPN Gateway should fail, all virtual IP addresses will be migrated to the functional AVG(s), that is, they will still be available.

Follow these steps to configure your AVG cluster as exemplified.

- Configure the SSL servers you need in the AVG cluster.
- Enable stand-alone mode for each configured SSL server.
- Add the IP addresses (that the domain name resolves to) to the IP list of each SSL server.
- Set the real server IP address of the backend server, or configure load balancing if using more than one backend server
- Specify the certificate to be used by each SSL server.

For detailed instructions on how to perform the preceding steps, see Configure the Virtual SSL Server Parameters on page 180 , Configure IP Address Migration on page 181 and Add Load Balancing Support on page 184

Stand-alone Setup Using Two Networks

This example shows how to configure the AVG cluster to receive incoming traffic (from the internet) through one network interface, and setup a session to the requested server on the

intranet through another. The network interface (NIC) facing the Internet is called the traffic interface and should always reside on the same network as the default gateway.

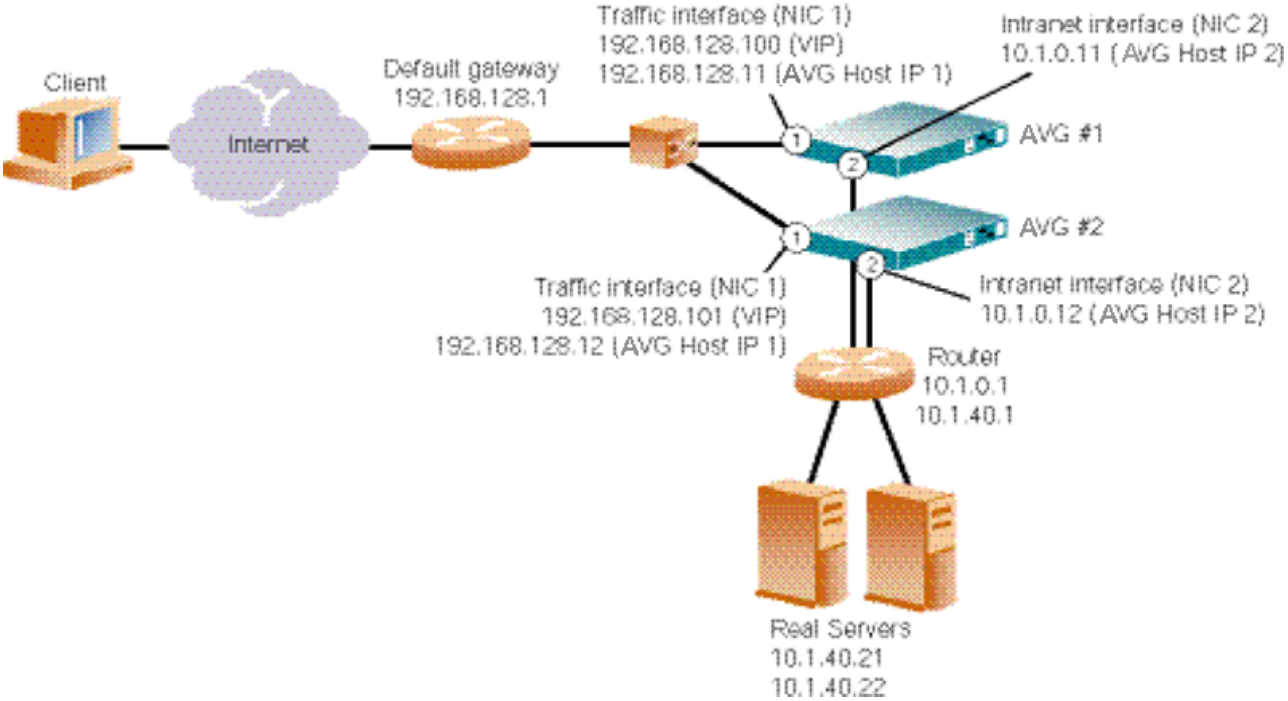

Figure 14: Stand-alone Setup Using Two Networks

Having created the desired number of SSL servers, enabled stand-alone mode, added IP addresses to the IP list and configured real server IP addresses (as described previously in this chapter), proceed with the following steps:

1. Configure Interface 2 on AVG host 1.

   This step describes how to add a new interface to AVG host 1. Note that the network mask can be entered in number of bits, e.g 24 instead of 255.255.255.0.

```
>> Main#cfg/sys/host 1/interface 2

Creating Host Interface 2
Enter new IP address of the Interface:10.1.0.11

Enter the network mask:24

Enter VLAN tag id [0]:<press ENTER to skip>

Entering: Interface ports menu
Port to add:2

Leaving: Interface ports menu
```

2. Configure Interface 2 on AVG host 2.

This step describes how to add a new interface to AVG host 2. Note that the network mask can be entered in number of bits, e.g 24 instead of 255.255.255.0.

```
>> Main#cfg/sys/host 2/interface 2

Creating Host Interface 2
Enter new IP address of the Interface:10.1.0.12

Enter the network mask:24

Enter VLAN tag id [0]:<press ENTER to skip>

Entering: Interface ports menu
Port to add:2

Leaving: Interface ports menu
```

3. Make sure the default gateway of both AVG hosts resides on the "dirty side", that is, the side facing the Internet.

```
>> Main#cfg/sys/host 1/gateway

Current value: 192.168.128.1
>> Main#cfg/sys/host 2/gateway

Current value: 192.168.128.1
```

4. Add static routes that point to the intranet.

   For incoming traffic to be routed to the requested server on the intranet, you have to add static routes pointing to the intranet. In this example, traffic to the backend servers (that is, IP addresses 10.1.40.21 and 10.1.40.22) should be routed through the router with the IP address 10.1.0.1.

```
>> Main#cfg/sys/routes

>> Routes#add

Enter destination address:10.1.40.0

Enter network mask:255.255.255.0

Enter gateway address:10.1.0.1
```

5. Apply your changes.

```
>> Routes#apply
```

Chapter 13: Global Server Load Balancing

Global server load balancing (GSLB) allows you to balance server traffic load across multiple physical sites. The GSLB implementation takes into account an individual site's health, response time and geographic location to smoothly integrate the resources of the dispersed server sites for complete global performance. For an introduction to GSLB, see the *Web OS Application Guide* version 9.0 or higher.

Note:

The following configuration examples assume that an ordinary Application Switch is configured for GSLB on the current site.

Configuration Example

Introduction

This configuration example is based on the following prerequisites: two sites—A and B—host the domain www.example.com. An ordinary Application Switch is configured for global server load balancing (GSLB).

There are several issues connected with GSLB and SSL that makes it impossible to rely on the switch to provide all GSLB functionality. The SSL box has to participate.

A typical situation when the issues primarily occur, is when a client has been sent to a site, say site A, by its initial DNS lookup of www.example.com. After a while, the client performs a new DNS lookup and is then sent to site B. If cookie persistence is used, the client should be sent back to site A.

The client is sent back to site A by using a HTTP redirect, traditionally to the VIP of site A. This is a redirect to https://VIP/path. If this is done for SSL, the browser will return an error, since it unsuccessfully tries to match the host part of the URL (VIP in this case) with the common name in the certificate of that site (in this case; www.example.com).

The only option is to redirect the client to the name of site A, that is, to www.example.com. However, since the client has already resolved www.example.com to site B, it will be sent back to site B instead of site A.

This situation is solved by setting up a secondary VIP bound to another DNS entry, www2.example.com. The VIP redirects the client to www2.example.com. Note that the

secondary VIP (www2.example.com) requires another certificate, one that has www2.example.com as significant name.

Maintaining cookie persistence

Cookie persistence is implemented by encoding the VIP and the RIP in the cookie. If a client arrives to a site with a cookie that contains a VIP that isn't local, it is redirected back to the remote site that has the VIP. The problem in this case, is that the session cannot be redirected back to the primary VIP of the remote site since the VIP cannot be used in the redirect. Instead, the session has to be redirected to a secondary VIP. This means that the remote site must recognize both its primary VIP and its secondary VIP as local when it receives a bounced session.

Setup

On site A: VIP 1, bound to www.example.com in DNS: 192.168.128.10 VIP 2, bound to www1.example.com in DNS: 192.168.128.20

On site B: VIP 2, bound to www.example.com in DNS: 192.168.128.30 VIP 3, bound to www2.example.com in DNS: 192.168.128.40.

This configuration example uses a local server named 192.168.128.100 on site A, and 192.168.128.200 on site B.

Configuring the Servers at site A and B

Now, configure each site, A and B.

Site A

Start loading the certificates for www.example.com and www1.example.com on the AVG for Site A (cert 1 and 2).

Add a Server Certificate

This step presumes a server certificate, signed by a certificate authority (CA), and a private key. The process for obtaining the required certificate files is covered in the "Certificates and Client Authentication" chapter in the *Users Guide* .

Once you have the appropriate certificate, use the following procedure to add the certificate to the AVG.

```
# /cfg/cert

Enter certificate number: (1-)1

Creating Certificate 1
>> Certificate 1#cert

Paste the certificate, press Enter to create a new line, and then type
"..." (without the quotation marks) to terminate.
```

The preceding example assumes that the certificate signing request (CSR) was generated from certificate number 1, which implies that the private key that corresponds to the public key in the certificate is already in place.

When prompted for the certificate, follow the instructions. Use Notepad or any other text editor to display the certificate. Then copy and paste the text of the certificate into the terminal window. For more detailed information about how to add certificates and keys to the AVG, see the "Certificates and Client Authentication" chapter in the *Users Guide* .

Important:

Once you have pasted the entire contents of the certificate file, press ENTER to create a new empty line and then type three periods (...). Press ENTER again to complete the installation of the certificate.

Configure the primary server (VIP 1: www.example.com) at site A

1. Configure the virtual server for VIP 1 (bound to www.example.com).

```
>> Main#/cfg/ssl/server 1

>> Server 1#vips

Current value: ""

Enter server ips (comma separated): 192.168.128.10
```

   The server must be of type http, since we rely on http redirects for passing users from one site to another.

```
>> Server 1#type http
```

2. Set up the certificate.

```
>> Server 1#ssl
```

```
>> SSL Settings#cert 1

>> SSL Settings# ..
```

3. Configure cookie persistence.

```
>> Server 1#adv/loadbalancing

>> Load Balancing Settings#persistence cookie

>> Load Balancing Settings#cookie

>> Cookie Settings#mode insert
```

4. Configure the domain to .example.com.

   Otherwise, the cookie will only be sent to one of the VIPs (the first contacted which will probably be www.example.com). If the domain is set to example.com, the cookie will be sent to all the VIPs; this is necessary so that persistence is maintained even when a user is redirected to the secondary VIP on a site.

```
>> Cookie Settings#domain .example.com
```

5. Configure **`localvips`** .

   The **`localvips`** are required so that the AVG will accept cookies that were initially set by the other VIP on the same site.

```
>> Cookie Settings#localvips

>> Local VIPs#add 192.168.128.10

>> Local VIPs#add 192.168.128.20

>> Local VIPs# ..
```

6. Set the cookie length to 16, so that both VIP and RIP are encoded within the cookie.

```
>> Cookie Settings#length 16

>> Cookie Settings# ..
```

7. Configure the backend servers. Start with the local backend server, 192.168.128.100.

```
>> Load Balancing Settings#backend 1
>> Backend Server 1#ip 192.168.128.100
>> Backend Server 1#port 80
>> Backend Server 1#ena
>> Backend Server 1# ..
```

8. Configure the remote backend server (vip 4 on site B).

```
>> Load Balancing Settings#backend 2
>> Backend Server 2#ip 192.168.128.40
>> Backend Server 2#port 443
>> Backend Server 2#remote true
```

9. Also configure the name that should be used when redirecting to VIP 4.

   The IP address of VIP 4 cannot be used because the host name in the URL redirection must match the name in the certificate for VIP 4.

```
>> Backend Server 2#rname www2.example.com
>> Backend Server 2#ena
>> Backend Server 2# ..
```

10. Finally, include the primary VIP of site B as a remote backend server.

    This VIP is required so that the primary site recognizes that cookies containing VIP 3 should be redirected back to site B. Because redirection to www.example.com would result in the client trying to connect to the primary site again, make sure to redirect the client to a name that exists only on site B; that is, www2.example.com.

```
>> Load Balancing Settings#backend 3
>> Backend Server 3#ip 192.168.128.30
>> Backend Server 3#port 443
>> Backend Server 3#remote true
>> Backend Server 3#rname www2.example.com
>> Backend Server 3#ena
```

```
>> Backend Server 3# ..
```

The next task is to configure the secondary (or local) VIP of the site.

Configure the local secondary server (VIP 2: www1.example.com) at site A

This VIP is used by site B for redirecting to, since it cannot redirect to www.example.com since that would simply send the client back to site B.

1. Configure the virtual server for VIP 2 (bound to www1.example.com.)

   It is almost identical to the configuration of the primary VIP.

```
>> Main#/cfg/ssl/server 2
>> Server 2#vips 192.168.128.20
>> Server 2#type http
```

2. Set up the certificate (www1.example.com).

```
>> Server 2#ssl
>> SSL Settings#cert 2
>> SSL Settings# ..
```

3. Configure cookie persistence.

```
>> Server 2#adv/loadbalancing
>> Load Balancing Settings#persistence cookie
>> Load Balancing Settings#cookie
>> Cookie Settings#mode insert
>> Cookie Settings#domain .example.com
```

4. Configure the `localvips` .

```
>> Cookie Settings#localvips
>> Local VIPs#add 192.168.128.10
>> Local VIPs#add 192.168.128.20
```

```
>> Local VIPs# ..
>> Cookie Settings#length 16
>> Cookie Settings# ..
>> Load Balancing Settings#backend 1
>> Backend Server 1#ip 192.168.128.100
>> Backend Server 1#port 80
>> Backend Server 1#ena
>> Backend Server 1# ..
>> Load Balancing Settings#backend 2
>> Backend Server 2#ip 192.168.128.40
>> Backend Server 2#port 443
>> Backend Server 2#remote true
>> Backend Server 2#rname www2.example.com

>> Backend Server 2#ena
>> Backend Server 2# ..
>> Load Balancing Settings#backend 3
>> Backend Server 3#ip 192.168.128.30
>> Backend Server 3#port 443
>> Backend Server 3#remote true
>> Backend Server 3#rname www2.example.com
>> Backend Server 3#ena
>> Backend Server 3# ..
```

Configuration of site A is now complete. Now configure site B similarly. The configuration for site B is a mirror image of the configuration of site A. Then, load certificate www.example.com (cert 1) and certificate www2.example.com (cert 3) into the AVG.

Configure the primary server (VIP 3: www.example.com) at site B

1. Set up the primary VIP (192.168.128.30).

```
>> Main#/cfg/ssl/server 1
>> Server 1#vips 192.168.128.30
>> Server 1#type http
```

2. Set up the certificate (www.example.com).

```
>> Server 1#ssl
>> SSL Settings#cert 1
>> SSL Settings# ..
```

3. Configure cookie persistence.

```
>> Server 1#adv/loadbalancing
>> Load Balancing Settings#persistence cookie
>> Load Balancing Settings#cookie
>> Cookie Settings#mode insert
>> Cookie Settings#domain .example.com
```

4. Configure the **`localvips`** .

```
>> Cookie Settings#localvips
>> Local VIPs#add 192.168.128.30
>> Local VIPs#add 192.168.128.40
>> Local VIPs# ..
>> Cookie Settings#length 16
>> Cookie Settings# ..
```

5. And configure the backend servers.

```
>> Load Balancing Settings#backend 1
>> Backend Server 1#ip 192.168.128.200
>> Backend Server 1#port 80
>> Backend Server 1#ena
>> Backend Server 1# ..
>> Load Balancing Settings#backend 2
>> Backend Server 2#ip 192.168.128.20
>> Backend Server 2#port 443
>> Backend Server 2#remote true
>> Backend Server 2#rname www1.example.com
>> Backend Server 2#ena
>> Backend Server 2# ..
>> Load Balancing Settings#backend 3
>> Backend Server 3#ip 192.168.128.10
>> Backend Server 3#port 443
>> Backend Server 3#remote true
>> Backend Server 3#rname www1.example.com
>> Backend Server 3#ena
>> Backend Server 3# ..
```

Configure the local secondary server (VIP 4: www2.example.com) at site B

1. Set up the primary VIP (192.168.128.40).

```
>> Main#/cfg/ssl/server 2
>> Server 2#vips 192.168.128.40
>> Server 2#type http
```

2. Set up the certificate (www2.example.com).

```
>> Server 2#ssl
>> SSL Settings#cert 3
>> SSL Settings# ..
```

3. Configure cookie persistence.

```
>> Server 2# adv/loadbalancing
>> Load Balancing Settings#persistence cookie
>> Load Balancing Settings#cookie
>> Cookie Settings#mode insert
>> Cookie Settings#domain .example.com
```

4. Configure the `localvips` .

```
>> Cookie Settings#localvips
>> Local VIPs#add 192.168.128.30
>> Local VIPs#add 192.168.128.40
>> Local VIPs# ..
>> Cookie Settings#length 16
>> Cookie Settings# ..
```

5. And configure the backend servers.

```
>> Load Balancing Settings#backend 1
>> Backend Server 1#ip 192.168.128.200
>> Backend Server 1#port 80
>> Backend Server 1#ena
>> Backend Server 1# ..
>> Load Balancing Settings#backend 2
>> Backend Server 2#ip 192.168.128.20
>> Backend Server 2#port 443
>> Backend Server 2#remote true
```

```
>> Backend Server 2#rname www1.example.com
>> Backend Server 2#ena
>> Backend Server 2# ..
>> Load Balancing Settings#backend 3
>> Backend Server 3#ip 192.168.128.10
>> Backend Server 3#port 443
>> Backend Server 3#remote true
>> Backend Server 3#rname www1.example.com
>> Backend Server 3#ena
>> Backend Server 3# ..
```

Test the setup

You can now test your setup by pointing your browser to www1.example.com and browse the site. This will cause a cookie to be generated which contains VIP2: RIP. Then, if you point the browser to www2.example.com, it should be redirected back to www1.example.com since site B will detect a cookie that contains a remote VIP.

Glossary

ARP	Address Resolution Protocol. A network layer protocol used to convert an IP address into a physical address, such as an Ethernet address. A host wishing to obtain a physical address broadcasts an ARP request onto the TCP/IP network. The host on the network that has the IP address in the request then replies with its physical hardware address.
AVG	Avaya VPN Gateway.
CA (Certificate Authority)	A trusted third-party organization or company that issues digital certificates. The role of the CA in this process is to guarantee that the entity granted the unique certificate is, in fact, who he or she claims to be.
CLI (Command Line Interface)	The text-based interface on the VPN Gateway, presented to the user after having logged in. The CLI can be accessed through a console connection or remote connection (Telnet or SSH). The CLI is used for collecting AVG information and configuring the AVG.
Cluster (of VPN Gateways)	A cluster is a group of VPN Gateways that share the same configuration parameters. There can be more than one AVG cluster in the network, each with its own set of parameters and services to be used with different real servers. Every cluster has a Management IP address (MIP), which is an IP alias to one of the master VPN Gateways in the cluster.
Console Connection	A connection to a VPN Gateway established through the console port.
CRL (Certificate Revocation List)	A list containing the serial numbers of revoked client certificates. Each CA issues and maintains their own CRLs. If you generate client certificates on the VPN Gateway, you can also create your own CRL.
CSR (Certificate Signing Request)	A request for a digital certificate, sent to a CA. On the VPN Gateway, you can generate a CSR from the command line interface by using the **`request`** command.
DCE (Data Communicatons Equipment)	A device that communicates with a Data Terminal Equipment (DTE) in RS-232C communications.
DER (Distinguished Encoding Rules)	A process for unambiguously converting an object specified in ASN.1 (such as an X.509 certificate, for example) into binary values for storage or transmission on a network.
Digital Certificate	The digital equivalent of an ID card used in conjunction with a public key encryption system. Digital certificates are issued by trusted third parties known as certificate

authorities (CAs), after verifying that a public key belongs to a certain owner. The certification process varies depending on the CA and the level of certification.

Digital Signature

A digital guarantee that a document has not been altered, as if it were carried in an electronically-sealed envelope. The "signature" is an encrypted digest of the text that is sent with the text message. The recipient decrypts the signature digest and also recomputes the digest from the received text. If the digests match, the message is proved intact and tamper free from the sender.

A digital signature ensures that the document originated with the person signing it and that it was not tampered with after the signature was applied. However, the sender could still be an impersonator and not the person he or she claims to be. To verify that the message was indeed sent by the person claiming to send it requires a digital certificate (digital ID) which is issued by a certification authority.

DIP (Destination IP) Address

The destination IP address of a frame.

DPort (Destination Port)

The destination port number, linking the incoming data to the correct service. For example, port 80 for HTTP, port 443 for HTTPS, port 995 for POP3S.

DTE (Data Terminal Equipment)

A device that controls data flowing to or from a computer. The term is most often used in reference to serial communications defined by the RS-232C standard. This standard defines the two ends of the communication channel as being a DTE and DCE device. However, using a null-modem cable, a DTE to DTE communication channel can also be established between, for example, two computers.

GSLB (Global Server Load Balancing)

An Application Switch feature that allows you to balance server traffic load across multiple physical sites. The GSLB implementation takes into account an individual site's health, response time, and geographical location to smoothly integrate the resources of the dispersed server sites for complete global performance.

IP Interface

IP interfaces are defined on the Application Switch and are used for defining the subnets to which the switch belongs. Up to 256 IP interfaces can be configured on an Application Switch. The IP address assigned to each IP interface provides the switch with an IP presence on your network. No two IP interfaces can be on the same IP subnet. The IP interfaces can be used for connecting to the switch for remote configuration, and for routing between subnets and VLANs (if used).

Master VPN Gateway

A VPN Gateway in a cluster that is in control of the MIP address, or can take over the control of the MIP address should another master fail. Configuration changes in the cluster are propagated to other members through the master VPN Gateways.

MIB (Management Information Base)

An SNMP structure that describes which groups and objects that can be monitored on a particular device.

MIP (Management IP) Address

An IP address that is an IP alias to a master VPN Gateway in a cluster of VPN Gateways. The MIP address identifies the cluster and is used when making configuration changes through a Telnet or SSH connection.

Nslookup A utility used to find the IP address or host name of a machine on a network. In order to use the **`nslookup`** command on the AVG, it must have been configured to use a DNS server.

NTP (Network Time Protocol) A protocol used to synchronize the real-time clock in a computer. There are numerous primary and secondary servers on the Internet that are synchronized to the Coordinated Universal Time (UTC) through radio, satellite or modem.

Passphrase Passphrases differ from passwords only in length. Passwords are usually short, from six to ten characters. Short passwords may be adequate for logging onto computer systems that are programmed to detect a large number of incorrect guesses, but they are not safe for use with encryption systems. Passphrases are usually much longer—up to 100 characters or more. Their greater length makes passphrases more secure.

PEM (Privacy Enhanced Mail) A standard for secure e-mail on the Internet. It supports encryption, digital signatures and digital certificates as well as both private and public key methods. Keys and certificates are often stored in the PEM format.

Ping (Packet INternet Groper) A utility used to determine whether a particular IP address is online.

PKCS #12 A standard for storing private keys and certificates.

PKI (public key infrastructure) Short for public key infrastructure, a system of digital certificates, Certificate Authorities, and other registration authorities that verify and authenticate the validity of each party involved in an Internet transaction. PKIs are currently evolving and there is no single PKI nor even a single agreed-upon standard for setting up a PKI. However, nearly everyone agrees that reliable PKIs are necessary before electronic commerce can become widespread.

A PKI is also called a trust hierarchy.

Portal Applies to the VPN feature. The Portal Web page is displayed following a successful login to a virtual SSL server configured as a portal server. The Portal contains five different tabs from where the user can access various intranet resources such as web, mail and file servers. For more information about the VPN feature, see the *Application Guide for VPN* .

Real Server Group A group of real servers that are associated with a virtual IP address (VIP) or filter on an Application Switch.

RIP (Real Server IP) Address A real server IP address that the Application Switch load balances to when requests are made to a virtual IP address (VIP).

RPort (Real Server Port) The real server port, which a virtual SSL server on the AVG uses when sending and receiving information to and from the real servers.

Setup Utility When turning on a VPN Gateway the very first time, the Setup utility starts up automatically. The Setup utility is used for performing a basic configuration of the

AVG cluster. The Setup utility first presents you with the choice of setting up the AVG as a single device, or to add the VPN Gateway to an existing cluster.

If you perform a reinstallation of the AVG software, you will also enter the Setup Utility after the VPN Gateway has rebooted.

SIP (Source IP) Address The source IP address of a frame.

Slave A VPN Gateway that depends on a master VPN Gateway in the same cluster for proper configuration.

SNMP (Simple Network Management Protocol) A network monitoring and control protocol. Data is passed from SNMP agents, which are hardware and/or software processes reporting activity in each network device (a VPN Gateway, for example), to the workstation console (or SNMP manager) used to oversee the network. The SNMP agents return information contained in a MIB (Management Information Base), which is a data structure that defines what information is obtainable from the device.

SOCKS A generic, proxy protocol for TCP/IP-based networking applications. The SOCKS protocol provides a flexible framework for developing secure communications by easily integrating other security technologies, for example, SSL.

SOCKS includes two components, the SOCKS server and the SOCKS client. The SOCKS server is implemented at the application layer, while the SOCKS client is implemented between the application and transport layers. The basic purpose of the protocol is to enable hosts on one side of a SOCKS server to gain access to hosts on the other side of a SOCKS server, without requiring direct IP reachability.

SPort (Source Port) The source destination port, linking the incoming data to the correct service. For example, port 80 for HTTP, port 443 for HTTPS, port 995 for POP3S.

SSH (Secure Shell) A program used to log into another computer over a network, execute commands in a remote machine, and move files from one machine to another. SSH provides strong authentication and secure communications over insecure channels.

SSL (Secure Sockets Layer) Protocol The SSL protocol is the leading security protocol on the Internet. It runs above the TCP/IP protocol and below higher-level protocols such as HTTP or IMAP. SSL uses TCP/IP on behalf of the higher-level protocols and, in the process, allows an SSL-enabled server to authenticate itself to an SSL-enabled client.

SSL VPN Feature allowing remote access to intranet resources (such as applications, mail, files, intranet web pages) through a secure connection. The underlying protocol used for these sessions is SSL.

With the SSL VPN feature enabled, mobile workers, telecommuters and partners can access information and/or applications on the intranet, either through the SSL VPN Portal Page (browser-based mode) or by using the SSL VPN SOCKS client software (transparent mode). What information should be accessible to the user is

determined through access rules. For more information about the VPN feature, see the *Application Guide for VPN* .

STP (Spanning Tree Protocol) An algorithm used in transparent bridges that dynamically determines the best path from source to destination. It avoids bridge loops (two or more paths linking one segment to another), which can cause the bridges to misinterpret results. The algorithm creates a hierarchical "tree" that "spans" the entire network including all switches. It determines all redundant paths and makes only one of them active at any given time.

TLS (Transport Layer Security) The TLS protocol provides communications privacy over the Internet. The protocol allows client/server applications to communicate in a way that is designed to prevent eavesdropping, tampering, or message forgery.

Traceroute A utility used to identify the route used for station-to-station connectivity across the network.

Trap If a trap is defined in the MIB, a trap message is sent from the SNMP agent to the SNMP manager when the trap is triggered. A trap can for example define a hardware failure in a monitored device.

URI (Uniform Resource Identifier) The addressing technology from which URLs are created. Technically, URLs such as HTTP:// and FTP:// are specific subsets of URIs, although the term URL is mostly heard.

VIP (Virtual IP) Address An IP address that the switch owns and uses to load balance particular service requests (like HTTP) to other servers.

Virtual Router A shared address between two devices utilizing VRRP, as defined in RFC 2338. One virtual router is associated with an IP interface defined on the Application Switch. All IP interfaces on the Application Switches must be in a VLAN. If there is more than one VLAN defined on the Application Switch, then the VRRP broadcast will only be sent out on the VLAN for which the associated IP interface is a member.

Virtual SSL Server A virtual SSL server handles a specific service on the AVG, such as HTTPS, SMTPS, IMAPS, or POP3S. You can create up to 256 virtual SSL servers per AVG cluster, and each virtual SSL server is mapped to a virtual server on the Application Switch. In order to authenticate itself towards clients making requests for the specified service, the virtual SSL server is configured to use a digital certificate.

VLAN (Virtual Local Area Network) VLANs are commonly used to split up groups of network users into manageable broadcast domains, to create logical segmentation of workgroups, and to enforce security policies among logical segments. Up to 246 VLANs are supported on an Application Switch running Web OS.

VRRP (Virtual Router A protocol that acts very similarly to Cisco's proprietary HSRP address-sharing protocol. The reason for both of these protocols is to ensure devices have a next hop or default gateway that is always available. For example, two or more devices

Redundancy Protocol)

sharing an IP interface are either advertising or listening for advertisements. These advertisements are sent through a broadcast message to address 224.0.0.18.

With VRRP, one switch is considered the master and the other is the backup. The master is always advertising through the broadcasts. The backup switch is always listening for the broadcasts. Should the master stop advertising, the backup will take over ownership of the VRRP IP and MAC addresses as defined by the specification. The switch announces this change in ownership to the devices around it by way of a gratuitous ARP and advertisements. If the backup switch didn't do the gratuitous ARP, the Layer 2 device attached to the switch would not know that the MAC address had moved in the network. For a more detailed description, refer to RFC 2338.

X.509

A widely-used specification for digital certificates that has been a recommendation of the ITU since 1988.

Index

M

N

P

R

S

T

W