January 2016
53-1003954-02

Brocade Virtual Traffic Manager and VMware Horizon View Servers

Deployment Guide

Contents

Preface

Welcome to the *Brocade Virtual Traffic Manager and VMware Horizon View Servers Deployment Guide*. Read this preface for an overview of the information provided in this guide and for contact information. This preface includes the following sections:

- About This Guide
- Contacting Brocade

About This Guide

The *Brocade Virtual Traffic Manager and VMware Horizon View Servers Deployment Guide* describes the different ways of load-balancing different View Server components. The guide also details the reference architecture of an AlwaysOn mechanism of deploying View Servers.

Audience

This guide is written for network operations professionals, server administrators, and DevOps professionals familiar with administering and managing application delivery controllers (ADCs), servers, and applications.

You must also be familiar with:

- VMware Horizon View Server components
- Brocade Virtual Traffic Manager (vTM)

For more details on the Brocade vADC product family, see http://www.brocade.com/vADC.

Contacting Brocade

This section describes how to contact departments within Brocade.

Internet

You can learn about Brocade products through the company website: http://www.brocade.com.

Technical Support

If you have problems installing, using, or replacing Brocade products, contact Brocade Support or your channel partner who provides support. To contact Brocade Support, see http://www.brocade.com/en/support.html.

Professional Services

Brocade Global Services has the expertise to help organizations build scalable and efficient cloud infrastructures. Leveraging 15 years of expertise in storage, networking, and virtualization, Brocade Global Services delivers world-class professional services, technical support, and education services, enabling organizations to maximize their Brocade investments, accelerate new technology deployments, and optimize the performance of networking infrastructures.

Chapter 1: Solution Overview

This chapter includes the following sections:

- Virtual Traffic Manager Overview
- VMware Horizon View

Virtual Traffic Manager Overview

Brocade Virtual Traffic Manager (vTM) is a software-based application delivery controller (ADC) that is designed to deliver faster and more reliable access to public websites and private applications. vTM frees applications from the constraints of legacy, proprietary, hardware-based load balancers, which enables applications to run on any physical, virtual, or cloud environment. With vADC products from Brocade, organizations can:

- Make applications more reliable with local and global load balancing.
- Scale application servers by up to 3x by offloading TCP and SSL connection overhead.
- Accelerate applications by up to 4x by using web content optimization (WCO).
- Secure applications from the latest application attacks, including SQL injection, XSS, and CSRF.
- Control applications effectively with built-in application intelligence and a full-featured scripting engine.

Virtual Traffic Manager offers much more than basic load balancing. It controls and optimizes end-user services by inspecting, transforming, prioritizing, and routing application traffic. The powerful TrafficScript® engine facilitates the implementation of traffic management policies that are unique to an application by allowing organizations to build custom functionality or leverage existing features in Virtual Traffic Manager in a specialized way. With vTM, organizations can deliver the following.

Performance

Improve application performance for users by offloading encryption and compression from the web server by dynamic caching and reducing the number of TCP sessions on the application.

Reliability and Scalability

Increase application reliability by load-balancing traffic across web and application servers, balancing load across multiple data centers (private or public clouds), monitoring the response time of servers in real-time to decide the fastest way to deliver a service, protecting against traffic surges, and managing the bandwidth and rate of requests used by different classes of traffic.

Advanced Scripting and Application Intelligence

Manage application delivery more easily with fine-grained control of users and services using TrafficScript, an easy-to-use scripting language that can parse any user transaction and take specific, real-time action based on user, application, request, or other criteria. Development teams use TrafficScript to enable a point of control in distributed applications, whereas operations teams use it to quickly respond to changing business requirements or problems within an application before developers can fix it.

Application Acceleration

Dramatically accelerate web-based applications and websites in real-time with optional web content optimization (WCO) functionality. WCO dynamically groups activities for fewer long-distance round trips, resamples and uses image sprites to reduce bandwidth, and minifies JavaScript and combines style sheets to give the best possible response time for loading a web page on any browser or device.

Application-Layer Security

Enhance application security by filtering errors in web requests and protecting against external threats, with the option of a comprehensive Layer 7 firewall to defend against deliberate attacks.

VMware Horizon View

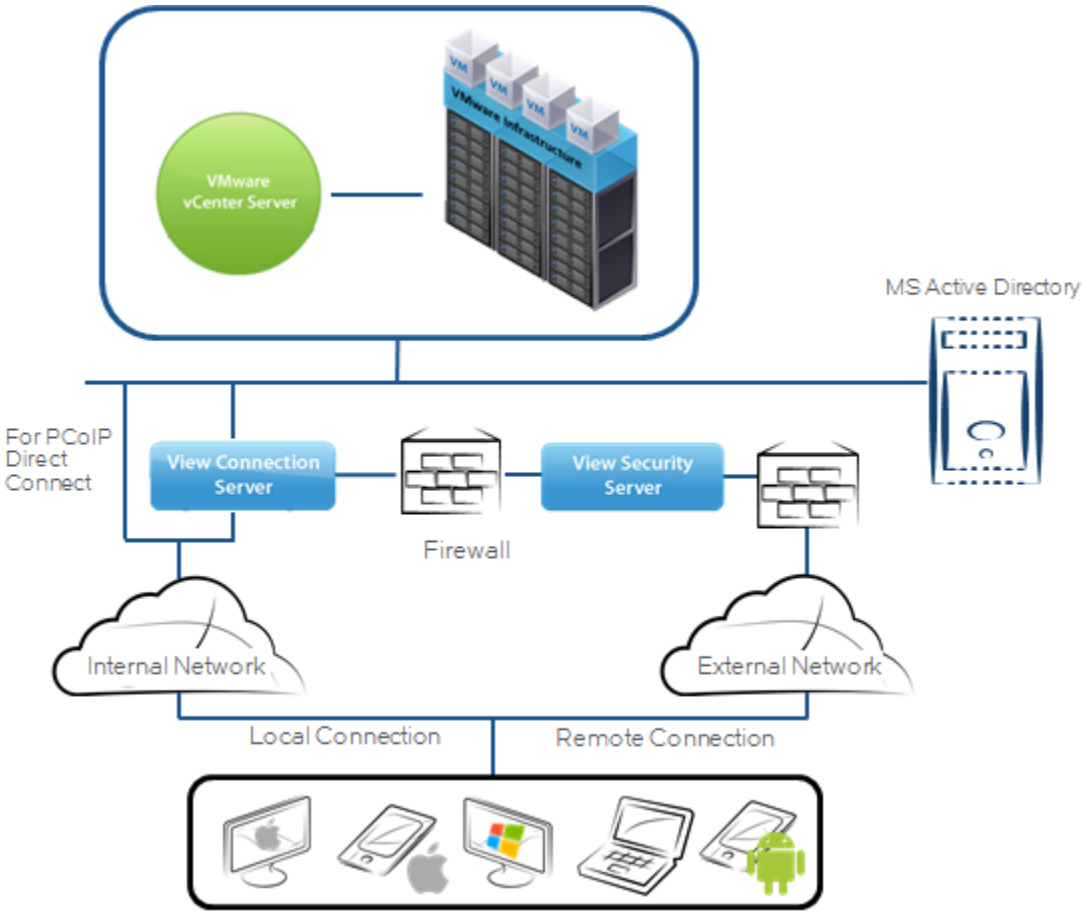
The VMware Horizon View portfolio of products delivers personalized virtual desktops as a managed service from a virtualization platform built to deliver the entire desktop, including the operating system, applications, and data. With Horizon View, desktop administrators virtualize the operating system, applications, and user data and deliver modern desktops to end users across a variety of network conditions.

Chapter 2: VMware Horizon View Architecture

The VMware Horizon View high-level network architecture, as depicted in the following figure, has a few key components relevant to this deployment guide.

- View Connection Server—View Connection Server acts as a broker for client connections. View Connection Server authenticates users through Windows Active Directory and directs the requests to the appropriate virtual machine, physical or blade PC, or Windows Terminal Services server. View Connection Server provides the following management capabilities:
- Authenticating users.
- Entitling users to specific desktops and pools.
- Assigning applications packaged with VMware ThinApp to specific desktops and pools.
- Managing local and remote desktop sessions.
- Establishing secure connections between users and desktops.
- View Security Server—View Security Server is a special instance of View Connection Server that runs a subset of View Connection Server functions. View Security Server provides an additional layer of security between the Internet and the internal network. A security server resides within a DMZ and acts as a proxy host for connections inside your trusted network. Each security server is paired with an instance of View Connection Server and forwards all traffic to that instance.

Figure 2-1: VMware Horizon View Architecture



In order to provide scalability and availability, a load balancer is deployed to load-balance both security servers and connection servers.

Chapter 3: Deploying Virtual Traffic Manager for View Servers

This chapter describes the process of deploying Virtual Traffic Manager in the VMware Horizon View architecture. It includes the following sections:

- Requirements
- Load-balance Connection Servers
- Configure vTM for View Connection Servers
- Load-balance Security Servers
- Configure vTM for View Security Servers

Requirements

- Brocade Virtual Traffic Manager (10.1 or later)
- SSL Certificates
- VMware View Servers (Connection and Security Servers, 6.x or earlier)
- VMware View Clients (3.x or earlier)

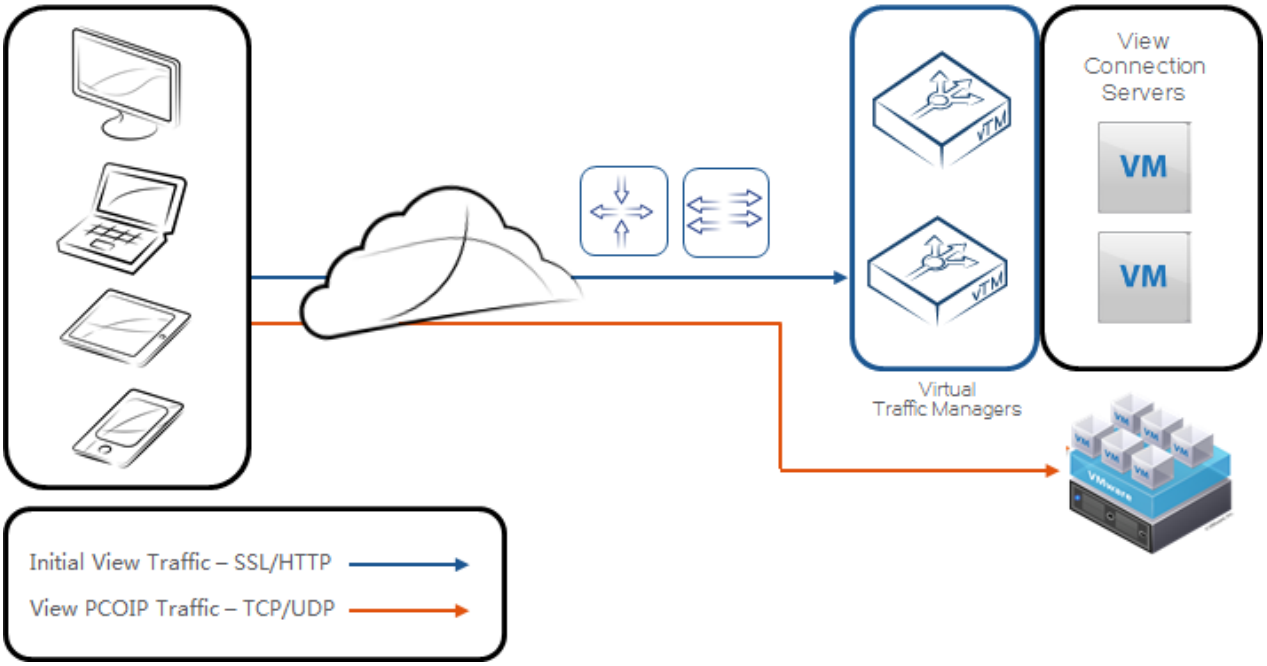
Note: This deployment guide was certified while the product was with Riverbed and for 9.x or earlier versions of the Virtual Traffic Manager.

Load-balance Connection Servers

The following traffic flow figure shows the Virtual Traffic Manager deployment with a VMware View using Connection Servers only. In this deployment, Virtual Traffic Manager does not handle PCoIP traffic, and the traffic flow is as follows:

1. The client machine makes an SSL connection to the Virtual Traffic Manager's Traffic IP address for the VMware Connection Servers.
2. Virtual Traffic Manager decrypts the SSL connection and load-balances the connection among View Connection Servers. Optionally, Virtual Traffic Manager can be configured to re-encrypt the connection established to the backend View Connection Server.
3. After authentication, desktop entitlement, and selection are complete, desktop (PCoIP) connections proceed to the appropriate View Desktop directly, bypassing Virtual Traffic Manager.

Figure 3-1: Load-balancing Connection Servers—TM Deployment with VMware Using Connection Servers Only



Configure the VMware Horizon View Connection Server for SSL Offload on Virtual Traffic Manager

The following steps allow the Virtual Traffic Manager to offload SSL transactions and send unencrypted web traffic directly to the View Connection Servers.

1. Log in to the **View Manager Administrator** tool.
2. From the navigation pane, click to expand **View Configuration** and then click **Servers**.
3. Click the **Connection Servers** tab on the main pane.
4. Select each Connection server, and click the **Edit** button. The **Edit View Connection Server** settings box opens.
5. On the **General** tab, deselect the **Use Secure Tunnel connection to desktop** check box if checked.
6. Deselect the **Use PCoIP Secure Gateway for PCoIP connections to desktop** check box if checked.
7. Click **OK** to close the window.

Allow HTTP Connections from Virtual Traffic Manager

When SSL is off-loaded to an intermediate server like Virtual Traffic Manager, you can configure View Connection Server instances to allow HTTP connections from the client-facing, intermediate devices. The intermediate devices must accept HTTPS for View Client connections.

To allow HTTP connections between View servers and intermediate devices, you must configure the config.properties (locked.properties, in older versions) file on each View Connection Server instance and security server on which HTTP connections are allowed.

Even when HTTP connections between View servers and intermediate devices are allowed, you cannot disable SSL in View. View servers continue to accept HTTPS connections as well as HTTP connections.

Note: If your View Clients use smart card authentication, the clients must make HTTPS connections directly to the View Connection Server or security server. SSL off-loading is not supported with smart card authentication.

To configure the config.properties (locked.properties) file:

1. Create or edit the config.properties (locked.properties) file in the SSL gateway configuration folder on the View Connection Server or security server host.

   For example: install_directory\VMware\VMware View\Server\sslgateway\conf\config.properties (locked.properties)
2. To configure the View server's protocol, add the **serverProtocol** property and set it to **http**. The value **http** must be typed in lower case.
3. (Optional) Add properties to configure a nondefault HTTP listening port and a network interface on the View server.
4. To change the HTTP listening port from 80, set **serverPortNonSSL** to another port number to which the intermediate device is configured to connect.
5. If the View server has more than one network interface, and you intend the server to listen for HTTP connections on only one interface, set **serverHost** to the IP address of that network interface.
6. Save the config.properties (locked.properties) file.
7. Restart the View Connection Server service to effect your changes.

This completes preparing the VMware Horizon View Connection servers configuration prerequisite for enabling SSL offloading onto Virtual Traffic Manager.

Configure vTM for View Connection Servers

The following table displays the process for configuring the Virtual Traffic Manager:

Component	Procedure	Description
Virtual Traffic Manager (once)	Create a traffic IP group.	A single traffic IP group must be created to front the View Connection Server pool. For details, see the "Create a Traffic IP Group" section.
	Create a pool.	A pool must be created per port. The IP address of each individual View Connection Server should be added to the pool. For details, see the "Create a Pool" section.
	Configure J2EE session persistence on the pool.	For details, see the "Configure Session Persistence" section.
	Configure the HTTP health monitor.	For details, see the "Configure a Health Monitor" section.
	Create a virtual server.	A virtual server must be created per port. For details, see the "Create a Virtual Server" section.
	Configure SSL decryption.	Import the certificate to perform SSL decryption on the vTM. For details, see the "Configure SSL Decryption" section.

Create a Traffic IP Group

A traffic IP group (also known as a virtual IP) must be created on which the virtual server will be listening on. To create a new traffic IP group, select **Services > Traffic IP Groups** and scroll down to **Create a new Traffic IP Group**. Fill in the fields as follows:

- **Name:** A descriptive name for the traffic IP group, i.e., *view.company.com* for the View Connection Servers.
- **IP Addresses:** An IP address to be associated to the FQDN of the View Connection Servers.

Create a Pool

A pool must be created for each service managed by the Virtual Traffic Manager. To create a new pool, select **Services > Pools** and scroll down to **Create a new Pool**. Fill in the fields as follows:

- **Pool Name:** A descriptive name for the pool.
- **Nodes:** *hostname:80 or ipaddress:80* if SSL decryption is enabled on Virtual Traffic Manager for each of the actual back-end Connection Servers. **Monitor:** Leave as **Ping** for now.

Configure Session Persistence

VMware Horizon View requires some form of session persistence. This section details the steps to configure session persistence.

Create a new session persistence class. A session persistence class must be created only once per type of persistence. For the VMware Horizon View application, J2EE Session Persistence should be used. The appropriate persistence class must be created.

1. Select **Catalogs > Persistence**.
2. Scroll down and create a new session persistence class.
3. Set the type to **J2EE Session Persistence**.

Attach the session persistence class to the pool.

1. Select **Services > Pools**, and select the pool created earlier.
2. Scroll down and click **Session Persistence**.
3. Choose the appropriate session persistence class.

Configure a Health Monitor

This section details the steps to create a health monitor. The HTTP monitor is used for port 80 on the View Connection Servers.

1. Select **Catalogs > Monitors**.
2. Scroll down to **Create new monitor**.
3. Give the new monitor a name. Set the type to **HTTP** and the scope to **Node**.
4. Click **Create Monitor** to create the monitor.
5. In the subsequent configuration page, scroll down and change the path to */*.
6. Change **status_regex** to **^200$**.
7. Change **body_regex** to **VMware View Portal**.

Attach the Monitor to a Pool

After the monitor has been created, it must be attached to the appropriate pool.

1. Select **Services > Pools**, and choose the pool that the monitor will be attached to.
2. Scroll down and click **Health Monitoring**.
3. Add the appropriate health monitor.

Create a Virtual Server

Create a virtual server to handle all View Client traffic.

1. Select **Services > Virtual Servers**, and scroll down to **Create a new Virtual Server**.
2. Enter the following:
    - **Virtual Server Name:** A descriptive name for the virtual server
    - **Protocol:** HTTP

- Port: 443
- Default Traffic Pool: The pool created earlier

3. Click Create Virtual Server.
4. In the next screen under Listening on, select Traffic IP Groups, and check the appropriate traffic IP group that was created earlier.

Configure SSL Decryption

Import the Certificate

In order to perform SSL decryption, the certificate and private key used for the View Virtual Server created earlier must be imported to the Virtual Traffic Manager.

1. Select the Catalogs > SSL > SSL Certificates catalog.
2. Click Import Certificate to import the appropriate certificate.

Enable SSL Decryption on the Virtual Server

After importing the certificate, enable SSL decryption on the virtual server created with the following steps:

1. Select Services > Virtual Servers, and choose the virtual server created earlier that will be doing the SSL decryption.
2. Scroll down and click SSL Decryption.
3. Set ssl_decrypt to Yes.
4. Select the certificate imported earlier.

Configuration Summary

By accessing the Services > Config Summary on the webGUI, a complete snapshot of all configured services is provided. This is a very useful table to glance through to get a good understanding of how the services are configured.

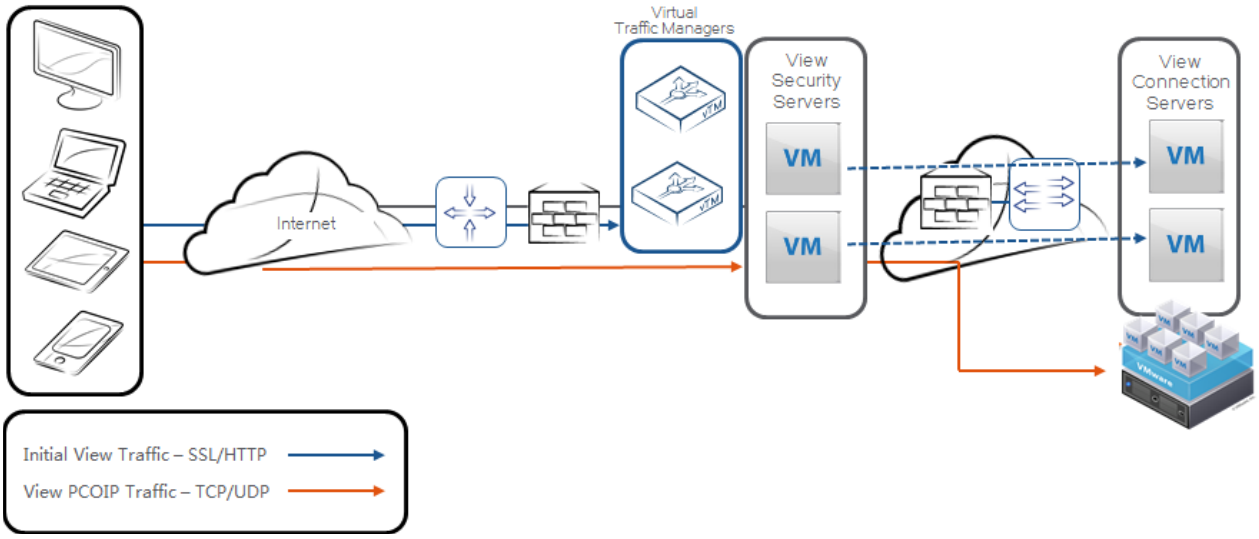
Virtual Servers ▽	Rules	Pools	Nodes	Monitors	Persistence
view.company.com View:443	*Use default pool*	**View-Connection-Pool**	10.5.100.182:80 10.5.100.183:80	**View Connection**	**view**

Load-balance Security Servers

The following traffic flow figure shows the Virtual Traffic Manager deployment with a VMware View using Security Servers only. In this deployment, Virtual Traffic Manager does not handle PCoIP traffic. The traffic flow is as follows:

1. The client machine makes a connection to the Virtual Traffic Manager's TrafficIP address for the VMware Security Servers.
2. Virtual Traffic Manager decrypts the SSL connection and load-balances the connection among View Security Servers. Virtual Traffic Manager re-encrypts the connection established to the backend View Security Server.
3. After authentication, desktop entitlement, and selection are complete, desktop (PCoIP) connections proceed directly to the View Security Server that was selected in Step 1 bypassing Virtual Traffic Manager.

Figure 3-2: Load-balancing Security Servers—TM Deployment with VMware Using Security Servers Only



Modify VMware Horizon View Security Server Settings

To modify the VMware configuration for View Security Server:

1. Log in to the **View Manager Administrator** tool.
2. From the navigation pane, click to expand **View Configuration** and then click **Servers**.
3. Click the **Connection Servers** tab on the main pane.
4. Select each Connection Server, and click the **Edit** button. The **Edit View Connection Server** settings dialog box opens.
5. On the **General** tab, in the **HTTP(S) Secure Tunnel External URL** box, type the Traffic IP address or DNS FQDN that you will configure on Virtual Traffic Manager for the Security Server, followed by a colon and the port.
6. Check **Use Secure Tunnel connection to Desktop**.
7. Click **OK** to close the window.
8. For each Security Server, configure the following:
    a. Select each Security Server, and click the **Edit** button. The **Edit View Security Server settings** dialog box opens.
    b. On the **General** tab, in the **HTTP(S) Secure Tunnel External URL** box, type the Traffic IP address or DNS FQDN that you will configure on Virtual Traffic Manager for the Security Server, followed by a colon and the port.
    c. Click **OK** to close the window.
    d. In **PCoIP External URL**, leave the IP address of the Security Server as is and do not configure the Traffic IP address because PCoIP traffic from the client will be handled directly by the Security Server and not by Virtual Traffic Manager.

Configure vTM for View Security Servers

The following table displays the process for configuring the Virtual Traffic Manager.

Component	Procedure	Description
Virtual Traffic Manager (once)	Create a traffic IP group.	A single traffic IP group must be created to front the View Security Server pool. For details, see the "Create a Traffic IP Group" section.
	Create a pool.	A pool needs to be created per port. The IP Address of each individual View Security server should be added to the pool. For details, see the "Create a Pool" section.
	Enable SSL encryption on the pool.	For details, see the "Enable SSL Encryption on the Pool" section.
	Configure J2EE session persistence on the pool.	For details, see the "Configure Session Persistence" section.
	Configure the HTTP health monitor.	For details, see the "Configure a Health Monitor" section.
	Create a virtual server.	A virtual server must be created per port. For details, see the "Create a Virtual Server" section.
	Configure SSL decryption.	Import the certificate to perform SSL decryption on the vTM. For details, see the "Configure SSL Decryption" section.

Create a Traffic IP Group

A traffic IP group (also known as a virtual IP) must be created on which the virtual server will listen on. To create a new traffic IP group, select **Services > Traffic IP Groups** and scroll down to **Create a new Traffic IP Group**. Fill in the fields as follows:

- **Name:** A descriptive name for the traffic IP group, i.e., *view.company.com* for the View Security Servers.
- **IP Addresses:** An IP address to be associated to the FQDN of the View Security Servers.

Create a Pool

A pool must be created for each service managed by the Virtual Traffic Manager. To create a new pool, select **Services > Pools** and scroll down to **Create a new Pool**. Fill in the fields as follows:

- **Pool Name:** A descriptive name for the pool.
- **Nodes:** hostname:443 or ipaddress:443 for each of the actual back-end Security Servers.
- **Monitor:** Leave as **Ping** for now.

Enable SSL Encryption on the Pool

Because Virtual Traffic Manager encrypts the traffic to SSL before sending it to View Security Servers, SSL encryption must be enabled under the pool configuration, as follows:

1. Select **Services > Pools** and click **SSL Settings**.
2. Set **ssl_encrypt** to **Yes**, and click **Update**.

Configure Session Persistence

VMware Horizon View requires some form of session persistence. This section details the steps to configure session persistence.

Create a new session persistence class. A session persistence class must only be created once per type of persistence. For the VMware Horizon View application, J2EE Session Persistence should be used. The appropriate persistence class must be created.

1. Select **Catalogs > Persistence**.
2. Scroll down and create a new session persistence class.
3. Set the type to **J2EE Session Persistence**.

Attach the session persistence class to the pool.

1. Select **Services > Pools** and select the pool created earlier.
2. Scroll down and click **Session Persistence**.
3. Choose the appropriate session persistence class.

Configure a Health Monitor

This section details the steps to create health monitors.

Create an HTTP monitor. The HTTP monitor is used for port 443 on the View Security Servers.

1. Select **Catalogs > Monitors**.
2. Scroll down to **Create new monitor**.
3. Give the new monitor a name. Set the type to **HTTP** and the scope to **Node**.
4. Click **Create Monitor** to create the monitor.
5. In the subsequent configuration page, scroll down and change the path to **/**.
6. Change **status_regex** to **^200$**.
7. Change **body_regex** to **VMware View Portal**.

After the monitor has been created, it must be attached to the appropriate pool.

1. Select **Services > Pools** and select the pool that the monitor will be attached to.
2. Scroll down and click **Health Monitoring**.
3. Add the appropriate health monitor.

Create a Virtual Server

1. Create a virtual server to handle all the View Client traffic.
2. Select **Services > Virtual Servers**, and scroll down to **Create a new Virtual Server**.

3. Enter the following:
   - Virtual Server Name: A descriptive name for the virtual server
   - Protocol: HTTP
   - Port: 443
   - Default Traffic Pool: The pool created earlier
4. Click Create Virtual Server.
5. In next screen under Listening on, select Traffic IP Groups, and check the appropriate traffic IP group that was created earlier.

Configure SSL Decryption

In order to perform SSL decryption, the certificate and private key used for View Virtual Server created earlier must be imported to the Virtual Traffic Manager.

1. Select the Catalogs > SSL > SSL Certificates catalog.
2. Click Import Certificate to import the appropriate certificate.

After importing the certificate, enable SSL decryption on the virtual server created with following steps:

1. Select Services > Virtual Servers, and select the virtual server created earlier that will be doing the SSL decryption.
2. Scroll down and click SSL Decryption.
3. Set ssl_decrypt to Yes.
4. Select the certificate imported earlier.

Configuration Summary

By accessing Services > Config Summary on the webGUI, a complete snapshot of all configured services is provided. This is a very useful table to glance through to get a good understanding of how the services are configured. The table will appear with a summary of the configuration, as shown below.

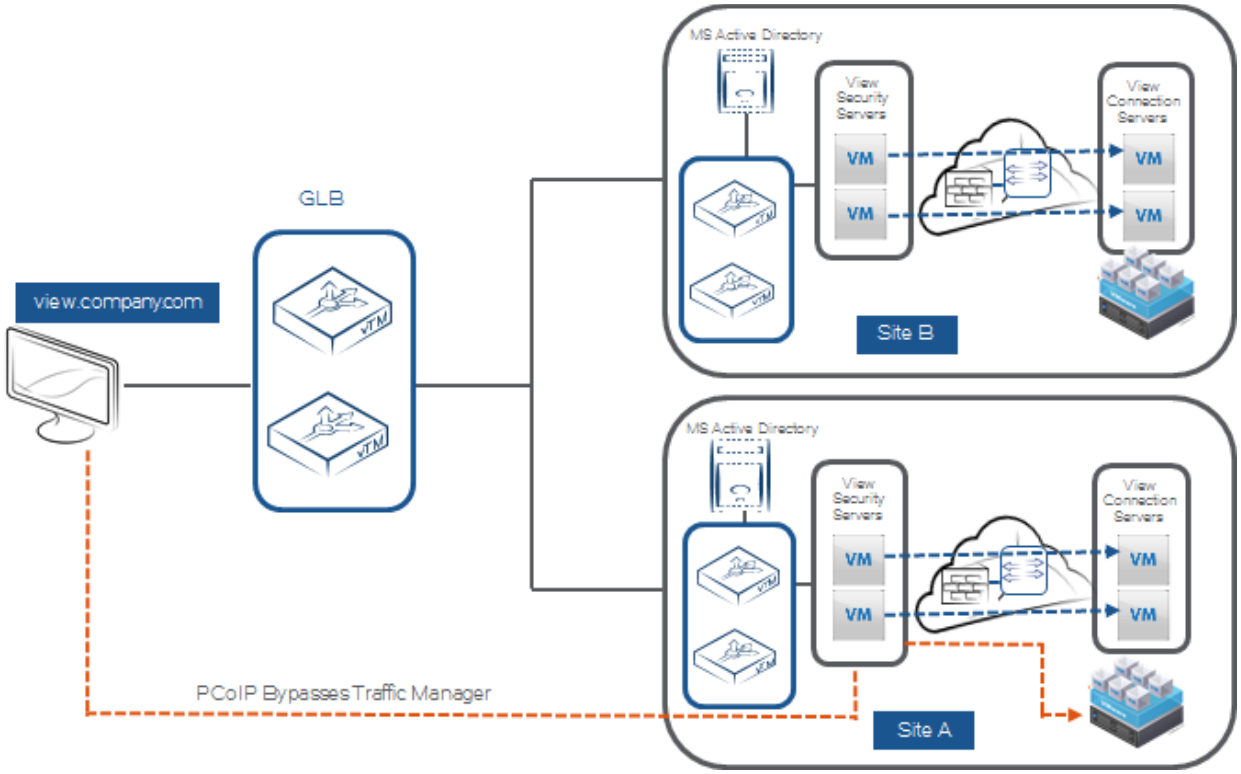
Virtual Servers ▽	Rules	Pools	Nodes	Monitors	Persistence
view.company.com View:443	*Use default pool*	**View-Security-Pool**	10.5.100.184:443 10.5.100.185:443	**View Security**	**View Persistence**

Chapter 4: Virtual Traffic Manager Deployment—VMware's AlwaysOn Desktop Reference Architecture

VMware AlwaysOn Desktop is a solution architecture that provides continuous availability of virtual desktops, applications, across devices, locations, and networks. This solution is an active-active configuration with multilevels of redundancy as well as continuous monitoring and load balancing between sites.

This AlwaysOn Desktop reference architecture use case is for organizations requiring a single namespace solution for VMware Horizon View, which allows the use of a single URL, while maintaining user persistence to the user's defined data center. This enables users to have access to their assigned virtual desktops as well as maintain persistence when accessing from different network connections such as Wi-Fi, a private LAN, a public LAN, and cellular connections. Additionally, Virtual Traffic Manager's GLB feature can be leveraged to direct clients to their closest data center based on geo-location. The following is high-level logical network diagram for this deployment.

Figure 4-1: VMware's AlwaysOn Desktop Reference Architecture—High-Level logical Network Diagram for This Deployment



In this deployment, the traffic flow is as follows:

1. A View Client attempts to access View Desktop using a secure connection to https://view.company.com. A DNS request is sent from the client to resolve the view.company.com FQDN.
2. If the Virtual Traffic Manager GLB feature is deployed, Virtual Traffic Manager receives the DNS query and determines the best site location based on the client's public IP address.
3. Let's say that in this flow the GLB picks Site A for the client; it then sends the Traffic IP of Site A Virtual Traffic Manager as the DNS response.

4. The client now makes a secure connection to the Site A TrafficIP. Virtual Traffic Manager in Site A is configured to query the Active Directory to determine the user association with the site based on Active Directory group membership.
5. Based on the user membership association, the request is sent to a specific pool. Based on our example, if the user is a member of Site A, it is sent to the Site A pool; else, it is sent to the Site B pool. The Site B pool consists of a single node, which is the TrafficIP of the Site B Virtual Traffic Manager. The default pool on the Site A Virtual Traffic Manager is set to the Site A pool.
6. The Site A pool node authenticates, does desktop entitlement, and is configured for PCoIP Gateway as the node's IP address and *not* the Virtual Traffic Manager's TrafficIP.
7. View client desktop (PCoIP) connections proceed directly to the View Security Server, completely bypassing the Virtual Traffic Manager.

Virtual Traffic Manager Configuration for Each Site

This section contains step-by-step instructions for configuring the Site A and Site B Virtual Traffic Managers based on the VMware Horizon View AlwaysOn Desktop reference architecture.

Component	Procedure	Description
Virtual Traffic Manager (each site)	Set up the authenticator.	For details, see the "Set Up the Authenticator" section.
	Create a traffic IP group.	A single traffic IP group must be created to front the View Security Server pool. For details, see the "Create a Traffic IP Group" section.
	Create pools.	A pool must be created per port. The IP address of each individual View Security Server should be added to the pool. For details, see the "Create Pools" section.
	Enable SSL encryption on the pool.	For details, see the "Enable SSL Encryption on the Pool" section.
	Configure the HTTP health monitor.	For details, see the "Configure a Health Monitor" section.
	Create a virtual server.	A virtual server must be created per port. For details, see the "Create a Virtual Server" section.
	Configure SSL decryption.	Import the certificate to perform SSL decryption on the vTM. For details, see the "Configure SSL Decryption" section.

Set Up the Authenticator

One of the key configuration steps on Virtual Traffic Manager is to set up Virtual Traffic Manager to query Active Directory for the user's group membership. For this configuration, the IP address of the Active Directory server, a user name with the privilege to make an LDAP query, and a password for the user name are required.

Configure the authenticator by accessing **Catalogs > Authenticators** in the Virtual Traffic Manager UI and under create new Authenticator, fill in the name, the host with the IP address of the AD server, and leave the port 389. Click the **Create Authenticator** button. In the **Authenticator** screen, fill in the following relevant information:

1. Enter the value for **ldap!bind!dn**. For example, **CN=vtm-bind,CN=Users,DC=company,DC=com**.
2. Enter the password under **ldap!bind!password** for the above-mentioned user. For the example, fill in the password for user **vtm-bind**.
3. Enter the value for **ldap!filter** as **sAMAccountName=%u**.
4. Enter the value for **ldap!filterbasedn**. For the example, **DC=company,DC=com**.
5. Enter the value for **ldap!attr** as * to request all attributes.
6. Click **Update** to complete the authenticator configuration.

Create a Traffic IP Group

A traffic IP group (also known as a virtual IP) must be created on which the virtual server will listen on. To create a new traffic IP group, select **Services > Traffic IP Groups** and scroll down to **Create a new Traffic IP Group**. Fill in the fields as follows:

- **Name:** A descriptive name for the traffic IP group, i.e., *view.company.com* for the View Security Servers.
- **IP Addresses:** An IP address to be associated to the FQDN of the View Security Servers.

Create Pools

Two pools, one for Site A and one for Site B, must be created on each site's Virtual Traffic Manager pair. On the Site A Virtual Traffic Manager pair, the default pool should consist of the Site A local View Security Server nodes. The second pool should be a Site B pool with the node address being the TrafficIP of the Site B Virtual Traffic Manager. On the Site B Virtual Traffic Manager pair, the default pool should consist of the Site B local View Security Server nodes. The second pool should be a Site A pool with the node address being the TrafficIP of the Site A Virtual Traffic Manager.

To create two pools on the Site A Virtual Traffic Manager pair, select **Services > Pools** and scroll down to **Create a new Pool**. Fill in the fields as follows:

1. **Pool Name:** A descriptive name for the Site A local pool.
2. **Nodes:** *hostname:443 or ipaddress:443* for each backend View Security Server if the pool that you are configuring is local to the vTM, for example: the Site A pool should consist of local View Security Server IP addresses or hostnames.
3. **Monitor:** Leave as **Ping** for now.
4. Click **Create Pool** to create the Site A local pool.

Create the Site B pool on the Site A Virtual Traffic Manager pair by selecting **Services > Pools** and scrolling down to **Create a new Pool**. Fill in the fields as follows:

1. **Pool Name:** A descriptive name for the Site B pool.
2. **Nodes:** The hostname of the TrafficIP on Site B:443 or the TrafficIP address of Site B:443.
3. **Monitor:** Leave as **Ping** for now.
4. Click **Create Pool** to create the Site B pool.
5. Under Pool **Basic Settings**, for **Failure Pool**, select the Site A pool created earlier from the drop-down.

Similarly, repeat these steps to create pools in the Site B Virtual Traffic Manager. In this case, the default pool will have the security servers of Site B as nodes, and the secondary pool will have the Traffic IP of the Site A Virtual Traffic Manager as its node.

Enable SSL Encryption on the Pool

Because the Virtual Traffic Manager encrypts the traffic to SSL before sending it to the View Security Servers, SSL encryption must be enabled under each pool (both the Site A pool and the Site B pool on the Virtual Traffic Manager pair) configuration as follows. To enable SSL encryption, select **Services > Pools** and choose the pool, and click **SSL Settings**. Set **ssl_encrypt** to **Yes**, and click **Update**.

Replicate this configuration on the Virtual Traffic Managers at both sites.

Configure a Health Monitor

This section details the steps to create the health monitor.

The HTTP monitor is used for port 443 on the View Security Servers.

1. Select **Catalogs > Monitors**.
2. Scroll down to **Create new monitor**.
3. Give the new monitor a name. Set the type to **HTTP** and the scope to **Node**.
4. Click **Create Monitor** to create the monitor.
5. In the subsequent configuration page, scroll down and set **use_ssl** to **Yes**.
6. Change the path to /.
7. Change **status_regex** to ^200$.
8. Change **body_regex** to **VMware View Portal**.

After the monitor has been created, it must be attached to both pools.

1. Select **Services > Pools** and select the pool that the monitor will be attached to.
2. Scroll down and click **Health Monitoring**.
3. Add the appropriate health monitor created earlier.

Replicate this configuration on the Virtual Traffic Managers at both sites.

Create a Virtual Server

Create a virtual server to handle all View Client traffic.

1. Select **Services > Virtual Servers**, scroll down to **Create a new Virtual Server**, and enter the following:
   - **Virtual Server Name:** A descriptive name for the Virtual Server
   - **Protocol:** HTTP
   - **Port:** 443
   - **Default Traffic Pool:** The pool created earlier
2. Click **Create Virtual Server**.
3. In next screen under **Listening on**, select **Traffic IP Groups**, and check the appropriate traffic IP group that was created earlier.

After creating the virtual server, HTTP-specific settings for this virtual server must be changed:

1. Select **Services > Virtual Servers** and select the virtual server created earlier.
2. Click **Connection Management**.
3. Click **HTTP-Specific Settings**.
4. From the drop-down, set **http_chunk_overhead_forwarding** to **eager**.

Replicate this configuration on the Virtual Traffic Managers at both sites.

Configure SSL Decryption

In order to perform SSL decryption, the certificate and private key used for the View Virtual Server created earlier must be imported to the Virtual Traffic Manager.

1. Select the **Catalogs > SSL > SSL Certificates** catalog.
2. Click **Import Certificate** to import the appropriate certificate.

After importing the certificate, SSL decryption can now be performed.

1. Select **Services > Virtual Servers**, and choose the virtual server created earlier that does the SSL decryption.
2. Scroll down and click **SSL Decryption**.
3. Set **ssl_decrypt** to **Yes**.
4. Select the certificate imported earlier.

Replicate this configuration on the Virtual Traffic Managers at both sites.

TrafficScript Rules and Virtual Server Association

This section walks through the steps to create a TrafficScript rule, which can query the AD server and create a persistence record based on the user's membership with AD groups. A TrafficScript rule contains a few important variables that must be changed based on your specific deployment.

Variable	Value
$authenticator = "*View_AD*"	Replace this variable with the authenticator name that is configured in Virtual Traffic Manager.
$view_siteA_pool = "*View-SiteA-Pool*";	Replace this variable with the pool name that is configured in Virtual Traffic Manager for Site A.
$view_siteB_pool = "*View-SiteB-Pool*";	Replace this variable with the pool name that is configured in Virtual Traffic Manager for Site B.
$view_guid = "624384c9-111e-459a-a4ab-f84aa0d48af9";	This value can be determined from Connection Server Cluster by viewing the following registry key value on the Standard Connection Server. HKLM\SOFTWARE\VMware, Inc.\VMware VDM\Node Manager\ConnectionServer Cluster GUID. Or you can use the value provided in this example.
$view_server = "*VSS1*";	Replace this variable with the hostname of one of the Connection Servers.

Variable	Value
$view_dns = "*company.com*";	Replace this variable with the full DNS domain.
$view_domain = "*MYDOMAIN*";	Replace this variable with the domain name used when a user logs in with Domain\username syntax.
$debug = 0;	Set the value to 1 if you want to enable informational logging. Set the value to 2 if debugs are needed and to 3 if extensive debugs are needed (like passwords).
$time_out = 120;	Timeout value, in seconds, for each persistence record. The default is set to 2 minutes.

Request and Response Rule

The following are the steps to create a request rule and associate it with a virtual server.

1. Copy the rule at the following website, and paste it into a text editor of choice:

   http://community.brocade.com/t5/vADC-Docs/VMware-View-AlwaysOn-Desktop-Reference-Architecture/ta-p/73959

2. Modify the variables in the rule based on your specific deployment configuration.
3. In the Virtual Traffic Manager WebUI, select **System > Global Settings**, scroll to the **Other Settings** section, and set **trafficscript!variable_pool_use** to **yes**.
4. Select **Catalogs > Rules**, scroll to the **Create new Rule** section, fill in the name for the rule, and select **Use Traffic Script Language**. Click the **Create rule** button to create the rule.
5. Copy the text from the file created in Step 2, and paste it under the **Rule** section. Click **Update** to save the rule.
6. Select **Services > Virtual Servers**, and click the virtual server for which to associate the rule created earlier for both requests and responses. Under the **Virtual Server** configuration screen, click **rules**, and from under the **Rules** configuration screen, under both the **Request Rules** and **Response Rules** section drop-down menu, select the rule that was created earlier and click **Add Rule**.

This completes the rule creation and association with the virtual server.

Replicate this configuration on the Virtual Traffic Managers at both sites.

Chapter 5: Conclusion

This document briefly discusses how to configure Virtual Traffic Manager to load-balance traffic to a pool of View Servers. Virtual Traffic Manager can make intelligent load-balancing decisions to direct View clients to View servers and improve the performance, security, reliability, and integrity of View traffic. Please refer to the product documentation on the Brocade Community Forums (http://community.brocade.com) for examples of how Brocade Virtual Traffic Manager can be deployed to solve a range of service hosting problems.

Appendix

TrafficScript Rule Associated to the Virtual Server

```
#// TS Rule for LB VMware View

#// Set the debug level (possible values: 0,1,2,3):
#// 0 = Logging Off
#// 1 = Informational Logging
#// 2 = Full Debug
#// 3 = Full Debug INCLUDING PASSWORDS - USE WITH EXTREME CARE
$debug = 2;

#// Rule to direct traffic based on AD group membership.
#// Please declare the names of the pools that you have configured, and ensure
#// that the trafficscript!variable_pool_use global setting is set to 'yes'.

#// What is the name of your VTM AD authenticator?
#// This can be set up under "Catalogs > Authenticators" in the Virtual Traffic Manager GUI.
$authenticator = "AD_AUTHENTICATOR";

#// What are the names of your pools to differentiate between Site A and Site B?
#// These are set up under "Services > Pools" in the Virtual Traffic Manager GUI.
$view_siteA_pool = "SITE-A-POOL";
$view_siteB_pool = "SITE-B-POOL";

#// Following are some View-specific variables that we need.
#// 'server' is the name of your View Connection Server.
#// 'dns' is the DNS search suffix (single suffix only) used in your setup.
#// 'domain' is the NT4 style name of your active directory.
#// 'guid' is the GUID of the View Connection Server, and can be found by checking
#// 'HKLM\SOFTWARE\VMware, Inc.\VMware VDM\Node Manager\ConnectionServer Cluster GUID' with
#// regedit.exe on your connection server.
$view_info =
   [ "guid" => "648a7ef8-dcba-abcd-efgh-58973ad94085",
```

```
    "server" => "rvbdvTMs1",
    "dns" => "brocade.local",
    "domain" => "brocade"];

#// Here we need to know about the names of the AD groups to map to each site.
$siteA_AD_Groupname = "brcd_DC0";
$siteB_AD_Groupname = "brcd_DC1";

#// Here we need to know about the IP addresses of your Site B Virtual Traffic Manager.
#// Site B VTM IP addresses (up to two supported). If your setup requires more, please
#// contact your Brocade sales representative to arrange for Brocade Professional
#// Services to provide a quote for customizing this deployment guide.
$site_B_VTM1_IP = "10.10.10.6";
$site_B_VTM2_IP = "";

#// Is this a request or response rule?
$rulestate = rule.getstate();

#// What is the client's IP address (useful for logging)?
$client_ip = request.getRemoteIP();

if ($rulestate == "REQUEST"){
if( $debug > 0 ) {log.info("#*#* Start Request Section *#*#");}
#// We need to extract the tunnel ID for persistence purposes and then break out of the rule to let the tunnel
through.
$path = http.getPath();
if (string.startswith($path, "/ice/tunnel")){
   $tunnelID = http.getQueryString();
   if ($tunnelID){
    if ($debug > 0){log.info("Request from client: " . $client_ip . " Using tunnelID:". $tunnelID . " as persistence
value");}
    connection.setPersistenceKey( $tunnelID );
    $choose_pool = data.get( $tunnelID );
    if( $debug > 0 ) { log.info( "Matching this to pool: ".$choose_pool ); }
    if( $choose_pool ) {
```

```
        pool.select( $choose_pool );
      }
    } else {
      log.error("Request from client: " . $client_ip . " Found no tunnelID:". $tunnelID . " in request for /ice/tunnel: This shouldn't happen, contact support.");
    }
    if ($debug > 1){log.info("Request from client: " . $client_ip . " Request is for /ice/tunnel - exiting rule to let it through");}
    #// Bypass script if the path is for "/ice/tunnel" to let the desktop session start.
    break;
}



$JSESSIONID = http.getCookie("JSESSIONID");
if($JSESSIONID){
  if ($debug > 0) {log.info("Request from client: " . $client_ip . " JSESSIONID=" .$JSESSIONID." found in the request...");}
  connection.setPersistenceKey($JSESSIONID);
  $choose_pool = data.get( $JSESSIONID );
  if( $debug > 0 ) { log.info( "Matching this to pool: ".$choose_pool ); }
  if( $choose_pool ) {
    pool.select( $choose_pool );
  }
} else {
  if ($debug > 0) {log.info("Request from client: " . $client_ip . " No JSESSIONID found in the request...");}
}

#// Bypass script if the path is for / as it could be GLB health monitor.
if( $path == "/") { break;}

#// Collect the HTTP request headers and body.
$req_headers = http.getRequest();
if(!http.getHeader("Expect")=="100-continue"){
     $req_body= http.getBody();
}
```

```
if ($debug > 1){log.info("Request from client: " . $client_ip . " HTTP Request is:\n" . $req_headers);}
if ($debug > 1){log.info("Request from client: " . $client_ip . " HTTP BODY is:\n" . $req_body);}

#// Reset flags to the needed defaults.
$username = '';
$password = '';

#// Inspect the request and see if it is something we are interested in:
#// syntax is xml.xpath.matchNodeSet( doc, nspacemap, query)
$is_xml = false;
if (!string.regexmatch($req_body, '\<\?xml',"i")){
  #// Document is _not_ an XML doc.
      if ($debug > 1){log.info("Request from client: " . $client_ip . " Request is NOT an XML document - exiting");}
      $is_xml = false;
      break;
} else {
  #// Document is an XML doc.
  $is_xml = true;
  if ($debug > 0){log.info("Request from client: " . $client_ip . " Request is an XML document");}
}

#// Test to see if we have been sent a "<get-configuration>" request.
$get_configuration = xml.xpath.matchNodeCount( $req_body, "", "//broker/get-configuration");
if ($debug > 0){log.info("Request from client: " . $client_ip . " get-config is:" . $get_configuration);}

#// Test to see if we have been sent a "<get-tunnel-connection>" request.
$get_tunnel_connection = xml.xpath.matchNodeCount( $req_body, "", "//broker/get-tunnel-connection");
if ($debug > 0){log.warn("Request from client: " . $client_ip . " get-tunnel-connection is:" . $get_tunnel_connection);}

#// Test to see if we have been sent a "<do-submit-authentication>" request.
$do_submit_authentication = xml.xpath.matchNodeCount( $req_body, "", "//broker/do-submit-authentication");
if ($debug > 0){log.info("Request from client: " . $client_ip . " do-submit-authentication is:" .
$do_submit_authentication);}

#// Test to see if we have been sent a "<do-logout>" request.
```

```
$do_logout = xml.xpath.matchNodeCount( $req_body, "", "//broker/do-logout" );
if ($debug > 0){log.info("Request from client: " . $client_ip . " do-logout is:" . $do_logout);}


#// Test to see if we have been sent a "<get-tunnel-connection>" request.
if ($get_tunnel_connection == 1 && $is_xml == true){
  if( $debug > 0 ){ log.info( "Request from client: " . $client_ip . " <get-tunnel-connection> identified from: " .
$client_ip );}
    connection.data.set("connection_state", "get-tunnel-connection");
 }
#// If we have a <get-configuration> query, we will send the first response and exit.
if ($get_configuration == 1 && $is_xml == true){
  if( $debug > 0 ){ log.info( "Request from client: " . $client_ip . " <get-configuration> response - Sending
first_response to client: " . $client_ip );}
  sendFirstResponse($view_info, $debug);
  break;
}
#// If we have been sent authentication credentials, we will go to work.
if ($do_submit_authentication == 1 && $is_xml == true){
  if( $debug > 0 ){ log.info( "Request from client: " . $client_ip . " <do-submit-authentication> identified from: " .
$client_ip );}


  $xml_user_credentials_data = xml.xpath.matchNodeSet($req_body,"","//broker/do-submit-
authentication/screen/params/param/values/value/text()");
  $xml_user_credentials_fieldnames = xml.xpath.matchNodeSet($req_body,"","//broker/do-submit-
authentication/screen/params/param/name/text()");


  #// We check that $xml_user_credentials_fieldnames contains "username, domain, password":
  if(string.regexmatch( $xml_user_credentials_fieldnames, "username, domain, password")){
    if ($debug > 0){log.info("Request from client: " . $client_ip . " <do-submit-authentication>: extracted username,
domain, password fields in submitted request.");}
    if ($debug > 1){log.info("Request from client: " . $client_ip . " <do-submit-authentication> extracted XML Fields:
" .$xml_user_credentials_fieldnames);}
    if ($debug > 2){log.info("Request from client: " . $client_ip . " <do-submit-authentication> extracted XML Values:
" .$xml_user_credentials_data);}


    #// Let's extract the username and password:
    $credentials = string.split($xml_user_credentials_data,",");
```

```
$username = $credentials[0];
$password = $credentials[2];
#// Currently we don't need the domain name, so we won't extract it, but it is here for future use if needed.
#//$cred_domain = $credentials[1];

$auth = auth.query( $authenticator, $username, $password );
#// We should check to ensure auth.query returned successfully:
#// If $auth returns 'Error' then something is wrong with the authenticator and the admin needs to investigate.
if( $auth['Error'] ) {
    log.error( "Request from client: " . $client_ip ." Error with authenticator " . $authenticator . ": " . $auth['Error'] );
}

#// Let's extract the list of groups the user is a 'memberOf'.
$groups = $auth['memberOf'];
#// If there is only one group, "$auth['memberOf']" will return a string,
#// not an array, so we need to force the $groups value to be an array.
$group_isArray = lang.isarray($groups);
if ($group_isArray != true){
  if ($debug > 1){log.info("Connection From: " . $client_ip .": $auth['memberOf'] returned a single group, forcing
$group to be an array");}
  $groups = lang.toArray($groups);
}

if ($debug > 1){log.info("Request from client: " . $client_ip ." Full Auth Info: " . lang.dump($auth));}
if ($debug > 1){log.info("Request from client: " . $client_ip ." Group Info: " . lang.dump($groups));}



#// Map Site B users to the Site B pool of servers.
foreach ( $group in $groups){
  if( $debug > 0 ) {log.info("$group is" . lang.dump($group));}


   if( string.contains( $group, $siteB_AD_Groupname ) ){
     if( $debug > 0) { log.info( "Request from client: " . $client_ip ." User: ".$username." member of SiteB Users
group");}
     pool.select( $view_siteB_pool );
```

```
        break;
      } else{
       if( $debug > 0) { log.info( "Request from client: " . $client_ip ." User: ".$username." is NOT a member of SiteB
Users group");}

      }
    }
    #// Map Site A users to the Site A pool of servers.
    foreach ( $group in $groups){
      if( string.contains( $group, $siteA_AD_Groupname ) ) {
        if( $debug > 0 ) { log.info( "Request from client: " . $client_ip ." User: ".$username." member of Default SiteA
Users group") ;}
        pool.select( $view_siteA_pool );
        break;
      } else{
       if( $debug > 0) { log.info( "Request from client: " . $client_ip ." User: ".$username." is NOT a member of SiteA
Users group");}

      }
    }
  }
} #// End do-submit-authentication


 if( $debug > 0 ) {log.info("#*#* End Reqeust Section *#*#");}
} #// End of REQUEST Section


sub sendFirstResponse( $info, $debug )  {
 #// $first_response = '<?xml version="1.0"?><broker version="7.0"><configuration><result>ok</result><broker-
guid>'.$info["guid"].'</broker-guid><broker-service-
principal><type>kerberos</type><name>'.$info["server"].'$@'.$info["dns"].'</name></broker-service-
principal><authentication><screen><name>windows-
password</name><params><param><name>domain</name><values><value>'.$info["domain"].'</value></values>
</param></params></screen></authentication></configuration></broker>';
$first_response = '<?xml version="1.0"?><broker version="7.0"><set-locale><result>ok</result></set-
locale><configuration><result>ok</result><broker-guid>'.$info["guid"].'</broker-guid><broker-service-
principal><type>kerberos</type><name>'.$info["server"].'$@'.$info["dns"].'</name></broker-service-
principal><authentication><screen><name>windows-
password</name><params><param><name>domain</name><values><value>'.$info["domain"].'</value></values>
```

```
</param></params></screen></authentication></configuration></broker>';
  if( $debug > 1 ){ log.info( "first_response data:\n" .$first_response); }
  http.sendResponse( "200 OK", "text/xml;charset=UTF-8", $first_response, "XFF: VTM_SiteA" );
}
###// AK:  should be ==
if ($rulestate == "RESPONSE" ){
if( $debug > 0 ) {log.info("#*#* Start Response Section *#*#");}
$debug_node = connection.getNode();
$debug_pool = connection.getPool();
$debug_http_response_code=http.getResponseCode();

$resp_headers = http.getResponseHeaders();
if ($debug > 1){log.info("Response to: " . $client_ip . " HTTP Response Headers are:" .
lang.dump($resp_headers));}

#// $resp_body= http.getResponseBody( 96 );  #// What is the nature of the response?
$content_type = http.getResponseHeader( "Content-Type" );
if ($debug > 1) { log.info("Content type of response is" . $content_type ); }
if( string.startsWith( $content_type, "text/xml" ) ) {
  #// TODO: What if the response doesn't contain a </broker> tag?
  $resp_body= http.stream.readResponse( 4096, "</broker>"); #// Limit was arbitrarily chosen
} else {
    if( $debug > 0 ) {log.warn( "This response was not XML - not extracting content for logging." );}
}

#// ASSUMPTION: Any XML response we care about (i.e., the one with the session ID in it) is less than 4096 bytes.
#// If it's longer, then we'll just stream it to the client.
if( string.length( $content_type ) < 4096 ) {

 if ($debug > 1) { log.info( "Grabbed response body:\n" . $resp_body); }

 if( $debug > 0 ) {log.info("RESPONSE: connection was sent to node:" . $debug_node);}
 if( $debug > 0 ) {log.info("RESPONSE: connection was sent to pool:" . $debug_pool);}
 if( $debug > 0 ) {log.info("RESPONSE: Server Responded with code:" . $debug_http_response_code);}
   $response_cookies = http.getResponseCookies();
```

```
  if($response_cookies){
    if($response_cookies["JSESSIONID"]){
      $JSESSIONID = $response_cookies["JSESSIONID"];
      #// Need to bypass for <set-locale> message.
       if (string.regexmatch($resp_body, '\<\?xml',"i")){
          if( $debug > 0 ) {log.info("#### PARSING XML RESPONSE ###");}
          $set_locale = xml.xpath.matchNodeCount($resp_body,"","//broker/set-locale/result");
          if( $debug > 0 ) {log.info("#### SET LOCALE IS: ". $set_locale ." ###");}
          if (!$set_locale){
             connection.setPersistenceKey( $JSESSIONID );
             data.set( $JSESSIONID, $debug_pool );
             if ($debug > 0) {log.info("Response Rule - Request from client: " . $client_ip ." Response set
JSESSIONID=" . $JSESSIONID .". Extracting and using for persistence key");}
          } else {
             if ($debug > 0) {log.info("Response Rule - Request from client: " . $client_ip ." Response set
JSESSIONID=" . $JSESSIONID .". _NOT_ extracting and using for persistence key");}
          }
       } else {
          if( $debug > 0 ) {log.warn( "Response data did not seem to be XML." );}
       }
    }
  } else {
    if ($debug > 0) {log.info("Response Rule - Request from client: " . $client_ip ." No JSESSIONID found in
response...");}
  }


  if (connection.data.get("connection_state") ==  "get-tunnel-connection"){
    if ($debug > 0){log.info("Response to: " . $client_ip ." HTTP Response Headers are:" .
lang.dump($resp_headers));}
    if ($debug > 0){log.info("Response to: " . $client_ip ." HTTP Response BODY is:" . $resp_body);}


    #// Inspect the response and see if it is something we are interested in:
    #// Syntax is xml.xpath.matchNodeSet( doc, nspacemap, query )
    if (string.regexmatch($resp_body, '\<\?xml',"i")){
       #// Document is an XML doc
```

```
    if ($debug > 0){log.info("Request from client: " . $client_ip . " Response to <get-tunnel-connection> is an XML
document");}
    $tunnelID = xml.xpath.matchNodeSet($resp_body,"","//broker/tunnel-connection/connection-id/text()");
    if ($tunnelID){
      if ($debug > 0) {log.info("Response Rule - Request from client: " . $client_ip . " Response set tunnelID=" .
$tunnelID .". Extracting and using for persistence key");}
       connection.setPersistenceKey( $tunnelID );
       data.set( $tunnelID, $debug_pool );
     }
   } else {
      lang.warn( "Didn't think response body was XML." );
   }
 } #// End of connection_state == get-tunnel-connection


 if ($debug > 1){log.info("Response to: " . $client_ip . " HTTP Response Headers are:" .
lang.dump($resp_headers));}
 if ($debug > 1){log.info("Response to: " . $client_ip . " HTTP Response BODY is:" . $resp_body);}
}


if( $resp_body ) {
   http.stream.startResponse( http.getResponseCode(), $content_type, "" );
   http.stream.writeResponse( $resp_body);
   http.stream.continueFromBackend();
}


if( $debug > 0 ) {log.info("#*#* End Response Section *#*#");}
} ##// End of RESPONSE Section #//
```