September 2015
53-1003955-01

Brocade Virtual Traffic Manager and Oracle WebLogic Applications

Deployment Guide

Contents

Preface

Welcome to the Brocade Virtual Traffic Manager (vTM) and Oracle WebLogic Applications Deployment Guide. Read this preface for an overview of the information provided in this guide and contact information. This preface includes the following sections:

- About This Guide
- Contacting Brocade

About This Guide

The Brocade Virtual Traffic Manager and Oracle WebLogic Applications Solution Guide describes how to configure Virtual Traffic Manager, to load balance and optimize applications running in Oracle WebLogic environment.

Example WebLogic Applications

Sample applications like Oracle PeopleSoft & Blackboard's Academic Suite that run in WebLogic Environment can be deployed according to the best practices documented in this Deployment guide.

Audience

This guide is written for network administrators, WebLogic server administrators and developer-operations (DevOps) professionals familiar with administering and managing both Application Delivery Controllers (ADCs) and Oracle WebLogic and the applications that run within.

You must also be familiar with:

- WebLogic Administration
- Installing and configuring applications in WebLogic environment
- Brocade Virtual Traffic Manager

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 Web site: 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
- Why Use vTM to Manage Oracle WebLogic Applications

Virtual Traffic Manager Overview

Brocade Virtual Traffic Manager (vTM) is a software-based application delivery controller (ADC) designed to deliver faster and more reliable access to public web sites and private applications. vTM frees applications from the constraints of legacy, proprietary, hardware-based load balancers, which enables them 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, CSRF, and more
- Control applications effectively with built-in application intelligence and 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 to leverage existing features in Virtual Traffic Manager in a specialized way. With vTM, organizations can deliver:

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 by 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 more. Development teams use TrafficScript to enable a point of control in distributed applications, while 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. It dynamically groups activities for fewer long distance round trips, resamples and sprites images 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 out errors in web requests, and protecting against external threats, with the option of a comprehensive Layer-7 firewall to defend against deliberate attacks.

Why Use vTM to Manage Oracle WebLogic Applications

Oracle WebLogic is a powerful, mature J2EE Application Server, forming the basis of one of the leading Internet application development platforms. Several customers worldwide use Oracle WebLogic Server to provide a reliable framework for scalable and secure applications.

vTM can further improve the reliability of WebLogic applications, double their performance (and hence double ROI), protect them from direct attacks and flash floods, and dramatically reduce the operational costs in managing servers and application upgrades.

Reliability

vTM provides a single point-of-entry to a cluster of Oracle WebLogic servers hosting applications. With a combination of passive and active health monitoring and performance measurements, vTM can route incoming requests to the fastest-responding servers and avoid failed or underperforming servers.

Request rate shaping ensures that your WebLogic servers can never be overloaded with requests, and advanced Session Persistence reliably pins users' sessions to individual servers for maximum reliability.

Acceleration

vTM has been proven to:

- As much as double the transaction rate that can be achieved from WebLogic, with no additional software or server tuning.
- Double the response speed, under sustained load from multiple clients.
- Eliminate all errors for all but the most demanding tests.

Security

Total server isolation, request and response scrubbing, request validation and request rate shaping helps to place your WebLogic servers in as secure an environment as possible, protecting them from direct attacks from internet clients, invalid or malformed HTTP requests and malicious or incidental flash floods that would severely impact the levels of service you provide.

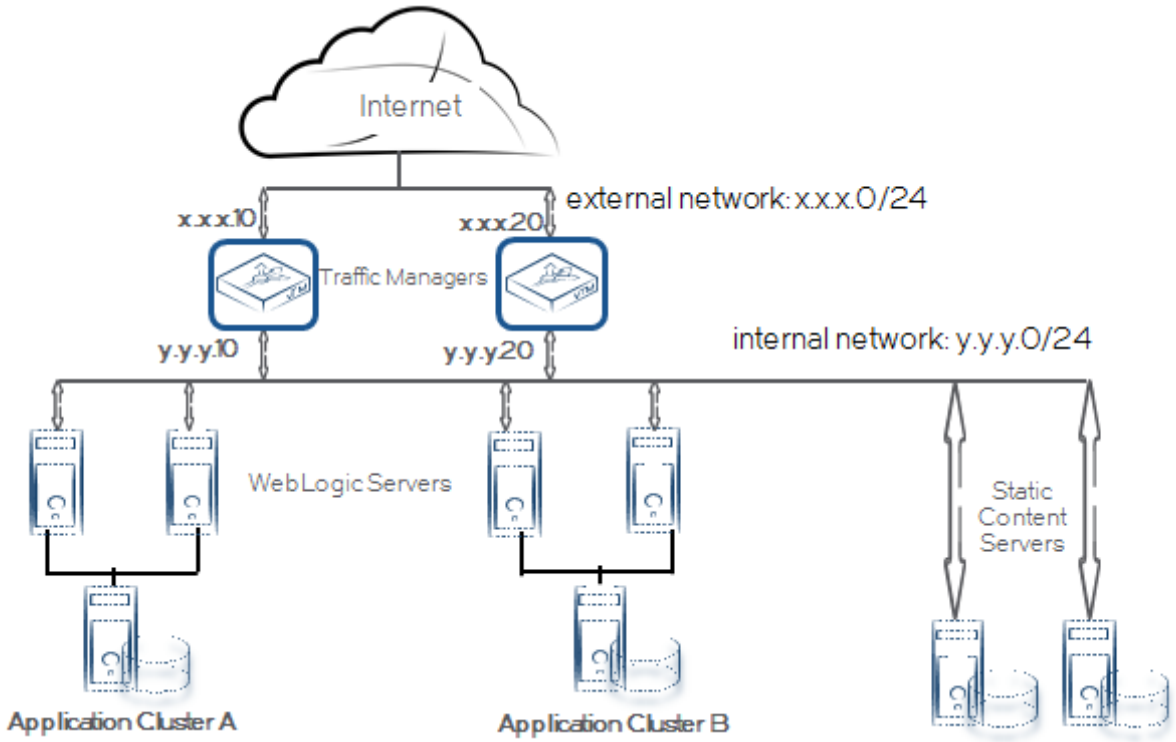
Management

vTM's advanced request routing and manipulation, driven by a fully programmable TrafficScript rules engine, gives you full flexibility to manage both your network traffic and server resources:

- Route traffic over multiple domains, draining traffic from one domain to another as applications are upgraded and user sessions complete.
- Load-balance over WebLogic servers and other servers to optimize resource utilization and eliminate unnecessary load on your application.
- Drain traffic from individual WebLogic servers, so that they can be taken out of service without any interruption to user's sessions.

Chapter 2: Oracle WebLogic Architecture

The diagram below shows a typical deployment of vTM with WebLogic with a pair of vTM servers installed in front of two application server clusters and a pair of web servers that serve static content.



The examples given in the rest of this document will refer to this diagram, and will use the IP addresses specified in it.

The vTM servers are shown operating in active-active mode, using a pair of Traffic IP addresses (x.x.x.10 and x.x.x.20). The DNS for the sites hosted by this architecture would list two A records (one for x.x.x.10 and one for x.x.x.20) for each website hostname.

vTM may also be configured to operate in active-passive mode; only a single IP address (x.x.x.10, for example) would be required for each hostname.

Although two application server clusters have been shown here in order to demonstrate request routing, vTM can be deployed in front of a single cluster. Similarly, a separate web-server cluster for static content is not essential, although it is recommended in many deployments since it reduces the load on the application servers.

Chapter 3: Deploying vTM for Oracle WebLogic Applications

This chapter describes the process and procedures for deploying Virtual Traffic Manager for load balancing applications deployed in a WebLogic environment. It includes the following sections:

- Requirements
- Understanding the Deployment Process

Requirements

- Brocade Virtual Traffic Manager (10.1 or later)
- Oracle WebLogic Server (Version 12.1.2.0.0)

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

Understanding the Deployment Process

The following table displays the process for deploying and configuring Virtual Traffic Manager for Oracle WebLogic:

Component	Procedure	Description
Virtual Traffic Manager (once)	Create a Traffic IP Group	A single Traffic IP Group must be created with the IP addresses used to resolve the hostnames of the website. For details, see “Creating Traffic IP Groups”
Virtual Traffic Manager (repeat for each set of servers)	Create a Pool	A Pool needs to have a set of servers to load-balance. Enter the hostname or IP address of the node along with the TCP/UDP port For details, see “Creating Pools”
Virtual Traffic Manager (repeat for each pool)	Create a Virtual Server	Create and associate the Virtual Server to the server pool of choice and the Traffic IP Group to listen on. For details, see “Creating Virtual Servers“
Virtual Traffic Manager (once in Traffic Script)	Create a request rule to preserve Client IP address	A Traffic Script rule to pass the client IP address to the application on the WebLogic Server For details, see “Passing the Client IP address to the WebLogic Application”
Virtual Traffic Manager (once)	Enable Session Persistence	Enable session persistence to preserve server side session state For details, see “Enabling Session Persistence”
Virtual Traffic Manager (as required)	HTTP and HTTPs versions of a website	Create another Virtual Server with port 443 in case of the need for HTTPs access. For details, see “Running both HTTP and HTTPs”
Virtual Traffic Manager (as required)	SSL decryption of HTTPs requests	Configure SSL Decryption to enable SSL offloads. For details, see “SSL Decryption”

Component	Procedure	Description
Virtual Traffic Manager (as required)	Traffic Routing	A Traffic Script rule to separate static and dynamic content. For details, see "Traffic Routing"

Creating Traffic IP Groups

A Traffic IP Group (also known as a Virtual IP) will need to be created on which the Virtual server will be listening on. To create a new Traffic IP Group:

1. Navigate to Services->Traffic IP Groups and scroll down to Create a new Traffic IP Group.
2. Fill in the fields as follows:
   - Name: A descriptive name for the applications inside WebLogic environment
   - IP Addresses: An IP Address that is mapped to FQDN of the application.
3. Click Create Traffic Group.

Creating Pools

A Pool has to be created for each application managed by the Traffic Manager. To create a new Pool:

1. Navigate to Services->Pools and scroll down to Create a new Pool.
2. Fill in the fields as follows:
   - Pool Name: A descriptive name for the pool.
   - Nodes: hostname:80 or ipaddress:80
   - Monitor: Select Full HTTP
3. In the next screen, click on Load Balancing.
4. Under Algorithm, select Perceptive.
5. Click on the Update button to apply changes.

Repeat this step for each pool of application servers and web servers.

Creating Virtual Server

Create a Virtual server that will handle the application Traffic. To create a new Virtual Server:

1. Navigate to 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: 80
   - Default Traffic Pool: Select the pool created in the previous step
3. Click on 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.

5. Set Enabled: to Yes.
6. Click on the Update button to apply changes.

Passing the Client IP address to the WebLogic Application

For logging purposes, preserving Client IP address in the header is important. This is accomplished by creating a Traffic Script and associating the Traffic Script to this virtual server. In order to create the Traffic Script:

1. Navigate to Catalogs→Rules.
2. Scroll down to Create new rule.
3. Give the new rule a name.
4. Select Use TrafficScript Language and click Create Rule.
5. Under the Rule: space enter the following text.
6. Click Update button to create the rule.

For WebLogic versions 6.x and later:

```
$debug = 0; // Change value to 1 if debug needed
$client_ip = request.getRemoteIP();
http.setHeader("WL-Proxy-Client-IP" , $client_ip);
if ($debug > 0) { log.info("WL-Proxy-Client-IP header added");}
```

For WebLogic versions 5.x and earlier:

```
$debug = 0; // Change value to 1 if debug needed
$client_ip = request.getRemoteIP();
http. setHeader("X-Forwarded-For", $client_ip);
if ($debug > 0) { log.info("WL-Proxy-Client-IP header added");}
```

Alternatively, for Virtual Traffic Manager versions 9.3 or later, the "X-Forwarded-For" option can be selected in the HTTP specific settings of the configured Virtual Server's Connection Management options in the UI.

Enabling Session Persistence

It is mandatory to enable session persistence for web applications to work correctly; this causes the second and subsequent requests from a client to be sent to the same back-end server as the first request, which prevents server side session state being lost. More recent versions of WebLogic Server share session state automatically between servers in a cluster; with these versions, enabling session persistence increases the performance of the application by reducing the number of times session data must be replicated between machines in a cluster.

There are three different types of session persistence that are appropriate for use with WebLogic Server:

- Transparent Session Affinity—With this method, vTM simply inserts a cookie into the response header, and uses this cookie to ensure that future requests received from the same client are sent to the same server as the first. This method has the advantage that it is trivial to configure, but the disadvantage that the application has no control over the cookie (and so cannot unset it when the user logs out, for example).
- Monitor Application Cookie—vTM can be set to monitor the JSESSIONID cookie that is set by WebLogic. This has the advantage that the application retains full control of the cookie.

- URL Rewriting—Both of the previous methods rely on the client's browser supporting HTTP cookies. Since not all browsers support cookies, and some users disable cookies as a privacy measure, WebLogic Server can be configured to add the JSESSIONID to the end of URLs in links generated by the application, when it detects that the user's browser is refusing cookies. vTM can extract the JSESSIONID from the requested URL, and use this as its session persistence key. Persistence on rewritten URLs can be combined with Monitor Application Cookie persistence to ensure that sessions are persistent regardless of whether the clients support cookies or not.

The combination of the "Monitor Application Cookies" and "URL Rewriting" methods is recommended for general-purpose use with WebLogic Server, but all three methods are documented below for reference.

When using the Monitor Application Cookies or Transparent Session Affinity methods, session persistence must be enabled for each of the pools that send traffic to WebLogic Server.

Monitor Application Cookies

1. Go to Services→ Pools → WebLogic Pool and click on the Session Persistence link. Click the Manage Session Persistence Classes link, and create a class.
2. Set this class to Monitor Application Cookies and set the cookie name to JSESSIONID.
3. Leave the failure mode set to choose a new node to use. This will cause vTM to send the request to a different node if the persistent node isn't available; in this circumstance, the chosen WebLogic server will recover a replica of the session state from a secondary server; the end user should not notice that a failure has occurred.
4. Click Update.

URL Rewriting Persistence

Configuring URL Rewriting Persistence is a two-stage process. First, a persistence class using "Universal Session Persistence" must be created, and then two TrafficScript rules written that detect a rewritten URL, extract the JSESSIONID from it and persist on this ID.

To create the session persistence class, go to Catalogs -> Persistence and create a new class called "url_rewriting":

1. Set this class to use the Universal Session Persistence method and the failure mode of choose a new node to use.
2. Click Update to finish.

   Note that you should not associate the url_rewriting class with any particular pool - the TrafficScript rule below will associate it with a request as and when it is required.
3. Go to Services → Virtual Servers → Your WebLogic Virtual Server → Rules.
4. Click Manage Rules in Catalog link in the Add New Request Rule section.
5. Create a new TrafficScript rule called "url_rewriting_persistence".
6. Cut and paste the following into the rule's text box, and click Update.

```
#// TS Rule for extracting JSESSION ID
$debug = 0; // Change value to 1 if debug needed
$cookie = http.getCookie( "JSESSIONID" );
if( $cookie ) break;

$url = http.getPath();
```

```
if (string.regexmatch($url, ".*;JSESSIONID=([\\w.]*).*", "i")) {
        $sessionid = $1;
        connection.setPersistence( "url_rewriting" );
        connection.setPersistenceKey( $sessionid );
        if ($debug > 0) { log.info("Persistence Values Set");}
}
```

Note that the argument to connection.setPersistence()must match the name of the persistence class you created above.

Finally, create a new response rule:

1. Go to Services → Virtual Servers → Your WebLogic Virtual Server → Rules.
2. Click the Manage Rules in Catalog link in the Add New Response Rule section.
3. Create a new TrafficScript rule called "url_rewriting_response".
4. Cut and paste the following into the rule's text box, and click Update.

```
#// TS Rule for rewriting responses
$debug = 0; // Change value to 1 if debug needed
# We're only interested in intercepting html responses
$contenttype = http.getResponseHeader( "Content-Type" );
if( ! string.startsWith( $contenttype, "text/html" ) ) break;

# Don't need to do this if the client accepted the cookie
$cookie = http.getCookie( "JSESSIONID" );
if( $cookie ) break;

# Weblogic will try to set the cookie on every request, so
# even if the client rejects it we can still persist on it.
$cookie = http.getResponseCookie( "JSESSIONID" );
if ( $cookie ) {
  connection.setPersistence( "url_rewriting" );
  connection.setPersistenceKey( $cookie );
  if ($debug > 0) { log.info("Persistence Values Set");}
}
```

Transparent Session Affinity

To enable Transparent Session Affinity, create a new session persistence class named "transparent", of type Transparent Session Affinity, and set this as the pool's persistence class.

1. The procedure to create a persistence class is similar to that described in the Monitor Application Cookies section above.
2. Go to Services → Pool → Your WebLogic Pool → Session Persistence to set the new class as your pool's session persistence class.

Running both HTTP and HTTPs

If required to run both an HTTP (unencrypted) and HTTPS (encrypted) version of an application, simply create two virtual servers that use the same default pool, one set to listen to port 443, with SSL Decryption enabled as described in the preceding section, and the other set to listen to port 80 without SSL Decryption enabled. Persistence must be enabled for both virtual servers.

SSL Decryption

In order to perform SSL decryption, the certificate and the private key must be imported into the Traffic Manager.

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

After importing the certificate, enable SSL decryption on the Virtual Server created:

1. Navigate to Services->Virtual Servers and select the virtual server that will be performing SSL decryption.
2. Scroll down and click on SSL Decryption.
3. Set ssl_decrypt to Yes.
4. Select the certificate imported in the previous step.
5. Scroll down to the bottom of the page and click Update.

Notifying WebLogic that the connection was encrypted

When SSL Decryption is enabled, WL-Proxy-SSL header must be set in the Traffic Manager. The following TrafficScript will enable that. Add the following TrafficScript rule to the Virtual Server as a Request Rule.

```
#// TS Rule for setting encryption flag in the header

$debug = 0; // Change value to 1 if debug needed

# Tell WebLogic if the connection to the client was encrypted
# and remove the Wl-Proxy-SSL header if not
if ( ssl.isSSL() ) {
        http.setHeader("WL-Proxy-SSL", "true");

        if ($debug > 0) { log.info("WL-Proxy-SSL header Set");}
} else {
        http.removeHeader("WL-Proxy-SSL");

        if ($debug > 0) { log.info("WL-Proxy-SSL header removed");}

}
```

Traffic Routing

TrafficScript can be used to distinguish requests for static content from those for dynamic content, sending the former to web servers and the latter to one or more of your application server clusters.

The principle is straightforward—write a TrafficScript rule that reads details of the request (usually the requested URL or Host header) and that uses these to decide which pool to send the requests to. Since the precise details of the rule will vary from application to application, the following is given by way of example only.

To use this rule, first create new pools:

- Pool cluster_a contains the WebLogic servers in Application Cluster A.
- Pool cluster_b contains the WebLogic servers in Application Cluster B.

- Pool static_servers contains static web servers.

Session persistence for cluster_b and cluster_a would be enabled; session persistence is not normally needed for static content.

Then add the rule below as the final request rule for the "weblogic_example" virtual server:

```
#// TS Rule for Routing Requests
$debug = 0; // Change value to 1 if debug needed
# URLs beginning /news/ /search/ or /cart/, and those ending .jsp and .do
# are dynamic, everything else is static.
$regex = "(/(news|search|cart)/|(.jsp|.do)$)";
$path = http.getPath();

# Send requests for anything other than dynamic content to the static cluster
if (! string.regexmatch($path, $regex)) {
    pool.use("static_servers");
    if ($debug > 0) { log.info("Redirected to Static Servers");}
}


# Split dynamic requests between the two application clusters, according
# to the host header - requests for www.example.com and sales.example.com
# are sent to cluster_a, all other requests to cluster_b.
$host = http.getHeader("Host");
if (string.regexmatch($host, "^(www|sales).example.com$")) {
    pool.use("cluster_a");
    if ($debug > 0) { log.info("Redirected to Cluster A");}
} else {
    # Everything else to the second application server cluster
    pool.use("cluster_b");
    if ($debug > 0) { log.info("Redirected to Cluster B");}
}
```

Chapter 4: Conclusion

This document briefly discusses how to configure Traffic Manager to load balance applications in a WebLogic environment. Traffic Manager is able to make intelligent load balancing decisions and improve the performance, security, reliability and integrity of the web traffic in this environment. 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 meet a range of service hosting problems.