



Deploying the BIG-IP LTM with Apache Tomcat and Apache HTTP Server

Table of Contents

Deploying the BIG-IP LTM with Tomcat application servers and Apache web servers

Prerequisites and configuration notes	1
Product versions and revision history	2
Configuration example	2
Typical management tasks for Apache Tomcat and BIG-IP LTM	4
Scalability	4
Monitoring	5
Maintenance	5
Configuring the Apache Web Server for load balancing	7
Statically compiled Apache server	7
Dynamically compiled Apache server	7
Configuring the Tomcat application server for load balancing	9
Configuring the BIG-IP LTM system for the Tomcat application servers	10
Creating the HTTP health monitor	10
Creating the pool	12
Creating profiles	13
Creating the persistence iRule	16
Creating the virtual server	17
Appendix A: Configuring the BIG-IP LTM to offload SSL	20
Using SSL certificates and keys	20
Creating a Client SSL profile	21
Creating the SSL persistence profile	21
Modifying the HTTP virtual server	22
Creating the HTTPS virtual server	23

Deploying the BIG-IP LTM with Tomcat application servers and Apache web servers

This deployment guide provides step-by-step procedures for configuring F5 devices with Apache web servers and Tomcat application servers.

In today's three-tiered application environments a typical architecture has a series of web and application servers separated by firewalls. The purpose of this guide is to instruct users how to load balance between the web servers and application servers by using the BIG-IP LTM. The benefits of using BIG-IP instead of software based plug-in modules include:

- Scalability - The ability to scale the back-end in more flexible ways.
- Monitoring - The ability to use advanced monitoring.
- Maintenance - The ability to perform maintenance without impacting the web server.
- Administration - The ability to use the BIG-IP system to ease administrative tasks.

Software plug-in solutions for load balancing using Apache jServ Protocol (AJP) are tested and reliable solutions, however using BIG-IP application delivery controllers provide greater flexibility, higher performance and a much wider range of options ranging from security to acceleration.

For more information on the Apache web server or the Apache Software Foundation, see <http://httpd.apache.org/>.

For more information on the Tomcat application server, see: <http://tomcat.apache.org/>

For more information on F5 devices described in this guide, see <http://www.f5.com/products/big-ip/>.

Prerequisites and configuration notes

The following are prerequisites and configuration notes:

- ◆ BIG-IP LTM must be running v9.4.x or later. We recommend using BIG-IP v10.1 or later.
- ◆ We recommend you have a thorough understanding of Tomcat application server. This guide is not intended to replace Tomcat documentation and best practices.
- ◆ We recommend a three-tiered architecture: web, application and database (see *Configuration example*, on page 2 for more information).
- ◆ We recommend reviewing the Apache deployment guide to optimize the web tier: <http://www.f5.com/pdf/deployment-guides/f5-apache-dg.pdf>
- ◆ This configuration described in this deployment requires that JSESSION ID based cookies are supported in your application, and that your Tomcat applications have their session information stored in a shared database so that load balancing will work properly during Tomcat outages.

Product versions and revision history

Product and versions tested for this deployment guide:

Product Tested	Version Tested
BIG-IP LTM	10.1, 10.2 (applicable to 9.x and later)
Apache Server	2.0.63 (also applicable to Apache 1.x and 2.2.x)
Tomcat Server	5 and 6

Revision history:

Document Version	Description
1.0	New deployment guide

Configuration example

In this deployment guide, the BIG-IP system is optimally configured to optimize and direct traffic to both the Apache web server and the Tomcat application server. Figure 1 shows a simple, logical configuration example with a redundant pair of BIG-IP LTM devices in front of a group of Apache web servers and Tomcat application servers.

Four devices are not required however. Through the use of Secure Network Address Translation (SNAT), the Apache web servers can simply proxy the requests directly to another set of virtual servers on the same BIG-IP system. We cover this as the recommended solution in this guide. Certain architectures may require separate BIG-IP devices for security reasons and in that scenario the use of SNAT should not be required.

In this document we use two sample applications, a bookstore and a healthcare application. The Apache web server is used as the HTTP protocol server. Apache is a far more robust and quicker HTTP server than Tomcat and many sites prefer to separate the functions. The Tomcat application server is used as the application server. Databases or Enterprise Java Bean (EJB) servers may also be included in this type of configuration, but they are out of the scope of this document.

In our flow, a client makes a request of the sample bookstore application (<http://example.com/bookstore>). The request is resolved to a virtual IP address on the BIG-IP system. The BIG-IP system terminates the TCP connection and directs the connection to one of the Apache web servers in the load balancing pool. Depending on the configuration, BIG-IP WebAccelerator may provide object caching to speed these requests, as well

as TCP optimization, SSL offload and other optimizations as described in the Apache deployment guide

(<http://www.f5.com/pdf/deployment-guides/f5-apache-dg.pdf>).

In the Apache configuration, we describe in this guide how to configure the Apache proxy module. Here, the Uniform Resource Identifier (URI) /bookstore is mapped by the Apache proxy module's configuration to a Tomcat context fronted by the BIG-IP system. The request for /bookstore then is proxied back to the BIG-IP LTM, on the specific Tomcat port (in our case initially 8080). The BIG-IP system then analyzes any JSession cookies for persistence and directs the request to the appropriate Tomcat instance in the pool. If the request is new, we configure the BIG-IP system to send the request to the Tomcat server with the least number of connections.

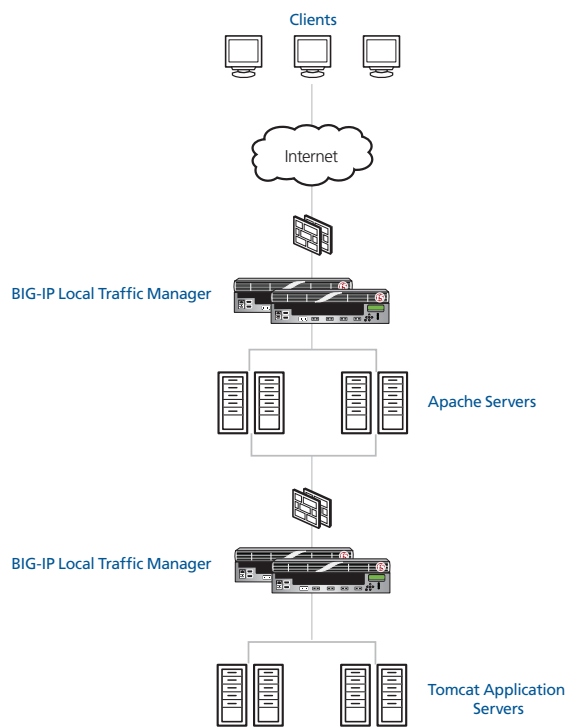


Figure 1 Logical configuration example

◆ **Note**

In this diagram we are showing the logical flow. In many physical installations, requests from the Apache servers simply go back through the same set of BIG-IP LTMs. In this scenario Secure Network Address Translations (SNAT) can be used to ensure that the Apache web application servers can reach the BIG-IP LTMs they are behind.

Typical management tasks for Apache Tomcat and BIG-IP LTM

In this section we provide some examples of typical management tasks for Tomcat applications with BIG-IP LTM. In cases where there are equivalent processes for Apache HTTP server with AJP protocol, we give some contrasts to make the processes clearer for users. This section contains only guidance for administrators already familiar with AJP methodology, and does not include step-by-step procedures.

Scalability

Using BIG-IP LTM new applications can easily be added and applications can be scaled by adding new servers.

Adding a new Tomcat context (applications)

- ◆ **On the BIG-IP LTM**

To add a new application to the BIG-IP LTM, the context for that application is added as a new *virtual server*, and then Apache HTTP server is modified, through the addition of a ProxyPass statement, to direct requests for this context to the virtual Server on BIG-IP. ProxyPass inclusion can be included in a separate file, so a restart of Apache HTTP Server is not required.

Therefore, the steps to create a new application context on the BIG-IP (described in this deployment guide in detail), are creation of a monitor, pool, profiles, virtual server, and then adding the ProxyPass statement to your HTTP server.

- ◆ **Apache HTTP using AJP equivalent**

Edit configuration files (such as httpd.conf) on each Apache HTTP server to indicate the new context. This information must include its port and each Tomcat application server. A restart of Apache HTTP server may be necessary.

Adding a new server for increased capacity

- ◆ **BIG-IP LTM**

To add a new server after Tomcat has been configured, simply edit the BIG-IP load balancing pools for each of your Tomcat applications to add the new pool member. Pool members can be left disabled until the appropriate time.

- ◆ **Apache HTTP using AJP equivalent**

Edit configuration files (such as httpd.conf) on each Apache HTTP server to indicate the new server. This information includes each new Tomcat server. A restart of Apache HTTP server may be necessary.

Monitoring

Using BIG-IP LTM Monitoring the health of each Tomcat context can be closely tracked so that requests are not sent to hosts that are not up or available.

- ◆ **BIG-IP LTM**

To manage uptime using HTTP monitoring on the BIG-IP LTM, a monitor is created for a specific application context. This monitor can be as simple as just expecting a default output from the given context, or it can be as complicated as calling a custom health check JSP page that exercises many aspects of the application. The output of this script can be parsed by BIG-IP to determine the application health, and then the specific instance that is not responding properly can be marked down.

- ◆ **Apache HTTP using AJP equivalent**

Custom health checks need to be written and configured.

Monitoring specific issues with an application

- ◆ **BIG-IP LTM**

If a particular Tomcat context is experiencing issues, advanced monitoring can be scripted on the BIG-IP LTM to measure and monitor the application performance. For scripted and advanced monitors, see detailed instructions in BIG-IP product documentation or refer your question to the opensource forum of DevCentral.

<http://devcentral.f5.com/Community/TopicGroups/tabid/1082223/asg/46/showtab/groupforums/Default.aspx?CSRT=7907073502817677552>

- ◆ **Apache HTTP using AJP equivalent**

Not applicable.

Maintenance

Using the BIG-IP LTM, you can easily and gracefully take servers and applications down for maintenance.

Taking down an application server for maintenance

- ◆ **BIG-IP LTM**

On BIG-IP LTM, shutting of an entire Tomcat server for maintenance (.e.g, upgrades, changes, etc) is easily accomplished through the user interface. An entire pool can be disabled and connections bled-off automatically so that there are no disconnects for users with long lived activity.

- ◆ **Apache HTTP using AJP equivalent**

Edit configuration files on each HTTP server, comment out server and restart Apache if necessary.

Taking down an application for maintenance

- ◆ **BIG-IP LTM**
On BIG-IP LTM, shutting off just one Tomcat context for maintenance (such as upgrades of the application) is also easily accomplished through the user interface. An entire pool can be disabled and connections bled-off automatically so there are no disconnects for users with long lived activity.
- ◆ **Apache HTTP using AJP equivalent**
Edit configuration files on each HTTP server, comment out server and restart Apache if necessary.

Configuring the Apache Web Server for load balancing

In this section, we modify the Apache web server to allow proxying of connections to the BIG-IP system. For this requirement, there are two possibilities depending on your web server configuration, a statically or dynamically compiled Apache server.

Statically compiled Apache server

If you have a statically compiled Apache server (for performance or other reasons) make sure **mod_proxy** is both compiled in and enabled (instructions for enabling **mod_proxy** are included below).

To check your statically compiled Apache instance for this feature, in a UNIX operating system, first locate the **httpd** binary, then execute the following command:

```
./httpd -l
```

If you see **mod_proxy.c**, your web server is configured with the required proxy component. If you do not see **mod_proxy.c**, you need to recompile and add **mod_proxy** at compile time. For instructions, see Apache web server documentations at <http://http.apache.org>.

Dynamically compiled Apache server

For dynamically configured Apache web server, ensure the following line is uncommented in your **httpd** configuration file (typically **httpd.conf** or **apache.conf**):

```
LoadModule proxy_module /usr/lib/apache2/modules/mod_proxy.so
```

In our example, the shared object being loaded is located in the **/usr/lib/apache2/modules/** directory. In your installation, the location may differ, so do not copy this line into your installation without checking the actual location of the dynamic shared object.

Next, either uncomment or add a proxy configuration to your web server. In our example, we use a **IFModule** command section to load the necessary configuration if **mod_proxy** is loaded:

```
<IfModule mod_proxy.c>
    ProxyRequests On
    <Proxy *>
        AddDefaultCharset off
        #Order deny,allow
        #Deny from all
        Order allow,deny
        Allow from all
        #Allow from .example.com
```

```
</Proxy>

# Enable/disable the handling of HTTP/1.1 "Via:" headers.
# ("Full" adds the server version; "Block" removes all outgoing Via: headers)
# Set to one of: Off | On | Full | Block

ProxyPass /bookstore http://TestTomcat.local:8080/bookstore
ProxyPassReverse /bookstore http://TestTomcat.local:8080/bookstore
ProxyPass /healthcarestore http://TestTomcat.local:8090/healthcarestore
ProxyPassReverse /healthcarestore http://TestTomcat.local:8090/healthcarestore

ProxyVia On
</IfModule>
```

Configuring and enabling the Proxy module only needs to be done once per Apache server. However, the **ProxyPass** and **ProxyPassReverse** statements are repeated for every Tomcat instance hosted on your Tomcat application serves.

Let's examine this in closer detail.

In our Apache web server configuration above, we have the following lines:

```
ProxyPass /bookstore http://TestTomcat.local:8080/bookstore
ProxyPassReverse /bookstore http://TestTomcat.local:8080/bookstore
ProxyPass /healthcarestore http://TestTomcat.local:8090/healthcarestore
ProxyPassReverse /healthcarestore http://TestTomcat.local:8090/healthcarestore
```

We configure Apache to proxy every request for the Tomcat context **/bookstore** to a virtual server instance on BIG-IP called **TestTomcat.local** on TCP port **8080** and every request for Tomcat context **/healthcarestore** to another virtual server instance on port **8090**.

Later in this guide we configure the BIG-IP system to run two virtual servers that answer on the IP address for **TestTomcat.local** on ports **8080** and **8090**.

For optimum performance, we recommend an IP address be used instead of a name in the ProxyPass setup. Or, if a DNS name is used, the name is locally mapped via the hosts file facility (**/etc/hosts** in UNIX and **c:\windows\system32\drivers\etc\hosts** using Microsoft Windows).

If a new Tomcat application is setup, our recommendation is to simply run that server on another TCP port. This allows for good flexibility between management and scalability. In this final example, if a third application is added, another new ProxyPass statement would be added to the Apache configuration:

```
ProxyPass /exampleapp http://TestTomcat.local:8100/exampleapp
ProxyPassReverse /exampleapp http://TestTomcat.local:8100/exampleapp
```

We cover the configuration of Apache Tomcat in the following sections.

This concludes the configuration of the Apache Web Server for forwarding to Tomcat Instances. For complete information how to setup BIG-IP LTM for Apache Web servers, see <http://www.f5.com/pdf/deployment-guides/f5-apache-dg.pdf>.

Configuring the Tomcat application server for load balancing

There are multiple ways to configure the Tomcat application server for any particular installation. To take advantage of BIG-IP's health monitoring, load balancing and persistence features, the only requirements for your Tomcat installation are that applications listen on a unique port or IP address, and that persistence be configurable with either JSession or similar cookie.

For a brief overview, here are the typical recommendations for running multiple tomcat applications on one server.

1. Partial separation using port numbers - One instance of Tomcat with contexts listening on different ports (This is the most common configuration and is covered in this guide).
2. Complete separation - Individual user account, individual instances of Tomcat calling a JVM or individual JVMs.
3. Partial separation using IP addresses - One instance of Tomcat listening on multiple IP addresses.

Each method has pros and cons, and unique configuration complexities. In this guide, we give examples using multiple Tomcat instances running on the same box, listening on multiple ports.

Application Name	Servers	Port
/book_store	10.133.81.100-105	8080
/healthcare_store	10.133.81.100-105	8090
/electronic_store	10.133.81.100-105	8100

Configuring the BIG-IP LTM system for the Tomcat application servers

In this section, we configure the BIG-IP LTM system to monitor Apache Tomcat Instances. This section also includes optional configuration for offloading SSL on the BIG-IP LTM.

The following table shows the BIG-IP LTM monitor, pool, profiles, and virtual server for each of our example Tomcat contexts.

Tomcat Context	Monitor	Pool	Profiles	Virtual Server
/book_store	bookstore-Tomcat	bookstore-Tomcat-pool	Tomcat-app-opt Tomcat-tcp-lan Tomcat-cookie*	bookstore-Tomcat-vs
/healthcare_store	healthcare-Tomcat	healthcare-Tomcat-pool	Tomcat-app-opt Tomcat-tcp-lan Tomcat-cookie*	healthcare-Tomcat-vs
/electronic_store	electronic-Tomcat	electronic-Tomcat-pool	Tomcat-app-opt Tomcat-tcp-lan Tomcat-cookie*	electronic-Tomcat-vs

* cookie persistence profile is optional

You may have more Tomcat contexts than this table, but we are showing that you must have a monitor, pool, and virtual server on the BIG-IP LTM for each context. The profiles and the iRule can be used across contexts.

Creating the HTTP health monitor

The first task is to set up a health monitor for each Tomcat instance. In this example we create the health monitor for the bookstore instance. Although the monitor in the following example is quite simple, you can configure optional settings such as Send and Receive Strings to make the monitor much more specific.

To create a health monitor

1. On the Main tab, expand **Local Traffic**, and then click **Monitors**.
2. Click the **Create** button. The New Monitor screen opens.
3. In the **Name** box, type a name for the Monitor.
In our example, we type **bookstore-Tomcat**.
4. From the **Type** list, select **http**.
5. In the Configuration section, in the **Interval** and **Timeout** boxes, type an Interval and Timeout. We recommend at least a (1:3) +1 ratio between the interval and the timeout (for example, the default setting has an interval of **5** and an timeout of **16**). In our example, we use a **Interval** of **30** and a **Timeout** of **91** (see Figure 2).

- In the **Send String** box, we recommend configuring the Send String to exercise the application in a functional way by calling a URL that makes calls to the application and the database. At the least the Send String should contain the context being called and be a valid call for your specific Tomcat instance.

In our example, we exercise the bookstore application by sending the string:

```
GET /bookstore/healthcheck.jsp HTTP/1.0\r\n\r\n
```

In this case, **healthcheck.jsp** is a custom JSP page that was written by the application developer with functions specific to the application.

- In the **Receive String** box, type the expected result from the Send String. In our example, the healthcheck.jsp returns Passed, so we type **Passed** in the box.
- Click the **Finished** button.
The new monitor is added to the Monitor list. This concludes the configuration of the health monitor for the bookstore instance.
- Repeat this procedure for every Tomcat instance and application that Tomcat hosts. In this guide we use bookstore and healthcarestore as two examples, so we configure another monitor for healthcarestore.

Local Traffic >> Monitors >> New Monitor...

General Properties

Name	bookstore-Tomcat
Type	HTTP
Import Settings	http

Configuration: Basic

Interval	30 seconds
Timeout	91 seconds
Send String	GET /bookstore/healthcheck.jsp HTTP/1.0\r\n\r\n
Receive String	Passed
User Name	
Password	
Reverse	<input type="radio"/> Yes <input checked="" type="radio"/> No
Transparent	<input type="radio"/> Yes <input checked="" type="radio"/> No

Cancel Repeat Finished

Figure 2 Creating the HTTP Monitor

Creating the pool

The next step is to define a load balancing pool for the Tomcat instance. A BIG-IP pool is a set of devices grouped together to receive traffic according to a load balancing method. This pool uses the monitor you just created.

To create the pool

1. On the Main tab, expand **Local Traffic**, and then click **Pools**.
The Pool screen opens.
2. In the upper right portion of the screen, click the **Create** button.
The New Pool screen opens.
3. From the Configuration list, select **Advanced**.
4. In the **Name** box, type a name for your pool.
In our example, we use **bookstore-Tomcat-pool**.
5. In the **Health Monitors** section, select the name of the monitor you created in *Creating the HTTP health monitor*, and click the Add (<<) button. In our example, we select **Tomcat-http-monitor**.
6. In the **Slow Ramp Time** box, type a number of seconds that corresponds to the expected number of requests per second.
For example, if the server farm is receiving 2000 requests per second, and there are 5 servers, each device receives approximately 400 requests per second (using the round robin load balancing method). When a server that has been offline comes back online, we don't want the BIG-IP to immediately send 400 requests to that device. In our example, we set the Slow Ramp Time to **30** seconds.
*Note: The **Slow Ramp Time** option does not appear unless you have selected **Advanced** from the Configuration list.*
7. From the **Load Balancing Method** list, choose your preferred load balancing method (different load balancing methods may yield optimal results for a particular network).
In our example, we select **Round Robin**.
8. In this pool, we leave the Priority Group Activation **Disabled**.
9. In the New Members section, make sure the **New Address** option button is selected.
10. In the **Address** box, add the first Tomcat server to the pool. In our example, we type **10.133.81.100**.
11. In the **Service Port** box, type **8080** or select **HTTP** from the list.
12. Click the **Add** button to add the member to the list.
13. Repeat steps 8-10 for each server you want to add to the pool.
In our example, we repeat these steps five times for the remaining servers, **10.133.81.101 - .105**.
14. Click the **Finished** button (see Figure 3).

Local Traffic >> Pools >> New Pool...

Configuration: Advanced

Name	bookstore-Tomcat-pool	
Health Monitors	Active Tomcat-http-monitor	Available gateway_icmp http https https_443 inband
Availability Requirement	All Health Monitor(s)	
Allow SNAT	Yes	
Allow NAT	Yes	
Action On Service Down	None	
Slow Ramp Time	30 seconds	
IP ToS to Client	Pass Through	
IP ToS to Server	Pass Through	
Link QoS to Client	Pass Through	
Link QoS to Server	Pass Through	
Reselect Tries	0	

Resources

Load Balancing Method	Round Robin	
Priority Group Activation	Disabled	
New Members	Address:	10.133.81.105
	Service Port:	8080 Select...
	<input type="button" value="Add"/>	
	R:1 P:0 C:0 10.133.81.100 :8080 R:1 P:0 C:0 10.133.81.101 :8080 R:1 P:0 C:0 10.133.81.102 :8080 R:1 P:0 C:0 10.133.81.103 :8080 R:1 P:0 C:0 10.133.81.104 :8080	
<input type="button" value="Edit"/>		<input type="button" value="Delete"/>

Figure 3 Creating the pool for the Tomcat servers

Creating profiles

The BIG-IP system uses configuration objects called profiles. A *profile* is an object that contains user-configurable settings for controlling the behavior of a particular type of network traffic, such as HTTP connections. Using profiles enhances your control over managing network traffic, and makes traffic-management tasks easier and more efficient.

Although it is possible to use the default profiles, we strongly recommend you create new profiles based on the default parent profiles, even if you do not change any of the settings initially. Creating new profiles allows you to easily modify the profile settings specific to this deployment, and ensures you do not accidentally overwrite the default profile.

Creating an HTTP profile

The first new profile we create is an HTTP profile. The HTTP profile contains numerous configuration options for how the BIG-IP LTM system handles HTTP traffic. In the following example, we base our HTTP profile off of the **http-acceleration** parent profile, as we are using the WebAccelerator. If you are not using the WebAccelerator, we recommend using the **http-wan-optimized-compression-caching** parent.

To create a new HTTP profile

1. On the Main tab, expand **Local Traffic**, and then click **Profiles**. The HTTP Profiles screen opens.
2. In the upper right portion of the screen, click the **Create** button. The New HTTP Profile screen opens.
3. In the **Name** box, type a name for this profile. In our example, we type **Tomcat-app-opt**.
4. From the **Parent Profile** list, select **http-wan-optimized-compression-caching**. If you are using the WebAccelerator module, select **http-acceleration**.
5. *Optional:* If you using the BIG-IP LTM to offload SSL, in the Settings section, check the Custom box for **Redirect Rewrite**, and from the **Redirect Rewrite** list, select **Match**. See *Appendix A: Configuring the BIG-IP LTM to offload SSL*, on page 1-20 for more information.
6. Modify any of the other settings as applicable for your network. In our example, we leave the settings at their default levels.
7. Click the **Finished** button.

Creating the TCP profile

The next profile we create are the TCP profile. In our example, we create a LAN optimized TCP profile.

To create a new TCP profile

1. On the Main tab, expand **Local Traffic**, and then click **Profiles**. The HTTP Profiles screen opens.
2. On the Menu bar, from the **Protocol** menu, click **tcp**.

-
3. In the upper right portion of the screen, click the **Create** button. The New TCP Profile screen opens.
 4. In the **Name** box, type a name for this profile. In our example, we type **Tomcat-tcp-lan**.
 5. From the **Parent Profile** list, select **tcp-lan-optimized**.
 6. Modify any of the settings as applicable for your network. In our example, we leave the settings at their default levels.
 7. Click the **Finished** button.

Optional: Creating persistence profile

Persistence is essential in an application server environment, especially in Java environments managed by Tomcat. For every user session, memory space is allocated on a particular server, so returning a user to the same machine, creates efficiency for both the server and the user.

Because the front-end Apache servers are essentially just acting as HTTP protocol servers, persistence back to a particular server is not needed. Apache web servers typically serves images or other static content in this scenario. However, persistence for Apache web servers is covered in the Apache deployment guide (listed at the top of this document).

For Tomcat persistence, there are two options. If the only persistence being used by your application is based on jSession cookie, the BIG-IP system's built-in cookie hash method can be used to maintain persistence. If URI based session persistence is also used, then it is our recommendation to use iRules to track persistence.

To create a new cookie persistence profile

1. On the Main tab, expand **Local Traffic**, and then click **Profiles**. The HTTP Profiles screen opens.
2. On the Menu bar, click **Persistence**. The Persistence Profiles screen opens.
3. In the upper right portion of the screen, click the **Create** button. The New Persistence Profile screen opens.
4. In the **Name** box, type a name for this profile. In our example, we type **Tomcat-cookie**.
5. From the **Persistence Type** list, select **Cookie**. The configuration options for cookie persistence appear.
6. From the **Cookie Method** list, select **Hash**.
7. In the Cookie Name box, type **JSESSIONID**.
8. Click the **Finished** button

Important

If your application only uses JSession cookie, it is safe to use the build-in cookie persistence profile which yields the fastest persistence results.

Figure 4 Creating the cookie persistence profile

If using persistence, it is a good idea to have a backup persistence method. In this example, we use Source Address Affinity.

To create a Source Address Affinity persistence profile

1. On the Main tab, expand **Local Traffic**, and then click **Profiles**. The HTTP Profiles screen opens.
 2. On the Menu bar, click **Persistence**. The Persistence Profiles screen opens.
 3. In the upper right portion of the screen, click the **Create** button. The New Persistence Profile screen opens.
 4. In the **Name** box, type a name for this profile. In our example, we type **Tomcat-source**.
 5. From the **Persistence Type** list, select **Source Address Affinity**. The configuration options appear.
 6. Modify any of the settings as applicable for your network. In our example, we leave the settings at their default levels.
1. Click the **Finished** button.

Creating the persistence iRule

If the JSession ID may come from a cookie or a URI, follow these steps to configure an iRule on the BIG-IP LTM system that allows the BIG-IP LTM system to use the application's JSESSIONID for persistence. The iRule looks for the JSESSIONID in the cookie, but also checks the URI if the cookie does not exist.

To create the iRule

1. On the Main tab, expand Local Traffic, click iRules, and then click the Create button.
2. In the **Name** box, enter a name for your iRule. In our example, we use **Tomcat-jsessionid**.
3. In the Definition section, type the following iRule (you can also copy and paste, but remove the line numbers):

```
1  when CLIENT_ACCEPTED {
2      set add_persist 1
3  }
4  when HTTP_RESPONSE {
5      if { [HTTP::cookie exists "JSESSIONID"] and $add_persist } {
6          persist add uie [HTTP::cookie "JSESSIONID"]
7          set add_persist 0
8      }
9  }
10 when HTTP_REQUEST {
11     if { [HTTP::cookie exists "JSESSIONID"] } {
12         persist uie [HTTP::cookie "JSESSIONID"]
13     } else {
14         set jsess [findstr [HTTP::uri] "jsessionid" 11 ";"]
15         if { $jsess != "" } {
16             persist uie $jsess
17         }
18     }
19 }
```

4. Click the **Finished** button.

Creating the virtual server

Next, we create a virtual server instance for each application server context. In our example, we create the virtual server for the bookstore application.

To create the virtual server

1. On the Main tab, expand **Local Traffic**, and then click **Virtual Servers**. The Virtual Servers screen opens.
2. Click the **Create** button. The New Virtual Server screen opens.

3. In the **Name** box, type a name for this virtual server. In our example, we type **Tomcat-bookstore-vs**.
Note: We recommend naming your pool with function of the Tomcat server being load balanced, which makes administration and debugging simpler.
4. In the **Destination** section, select the **Host** option button.
5. In the **Address** box, type the IP address of this virtual server. In our example, we use **192.168.10.120**.
6. In the **Service Port** box, type **80**, or select **HTTP** from the list.
7. From the Configuration list, select **Advanced**.
8. Leave the **Type** list at the default setting: **Standard**.
9. From the **Protocol Profile (Client)** list select the name of the profile you created in the *Creating the TCP profile* section. In our example, we select **Tomcat-tcp-lan**.
10. From the **Protocol Profile (Server)** list, select the name of the profile you created in the *Creating the TCP profile* section. In our example, we select **Tomcat-tcp-lan**.
11. From the HTTP Profile list, select the name of the profile you created in the *Creating an HTTP profile* section. In our example, we select **Tomcat-app-opt**.

Local Traffic » Virtual Servers » New Virtual Server...	
General Properties	
Name	Tomcat-bookstore-vs
Destination	Type: <input checked="" type="radio"/> Host <input type="radio"/> Network Address: 192.168.10.120
Service Port	80 HTTP
State	Enabled
Configuration: Advanced	
Type	Standard
Protocol	TCP
Protocol Profile (Client)	Tomcat-tcp-lan
Protocol Profile (Server)	Tomcat-tcp-lan
OneConnect Profile	None
NTLM Conn Pool	None
HTTP Profile	Tomcat-app-opt

Figure 5 Tomcat virtual server configuration (truncated)

12. In the Resources section, from the **Default Pool** list, select the pool you created in the *Creating the pool* section. In our example, we select **bookstore-Tomcat-pool**.
13. From the **Default Persistence Profile** list, select the persistence profile you created in the *Optional: Creating persistence profile* section. In our example, we select **Tomcat-cookie**.
14. From the **Fallback Persistence Profile** list, select the fallback persistence profile you created in the *Optional: Creating persistence profile* section. In our example, we select **Tomcat-source**.

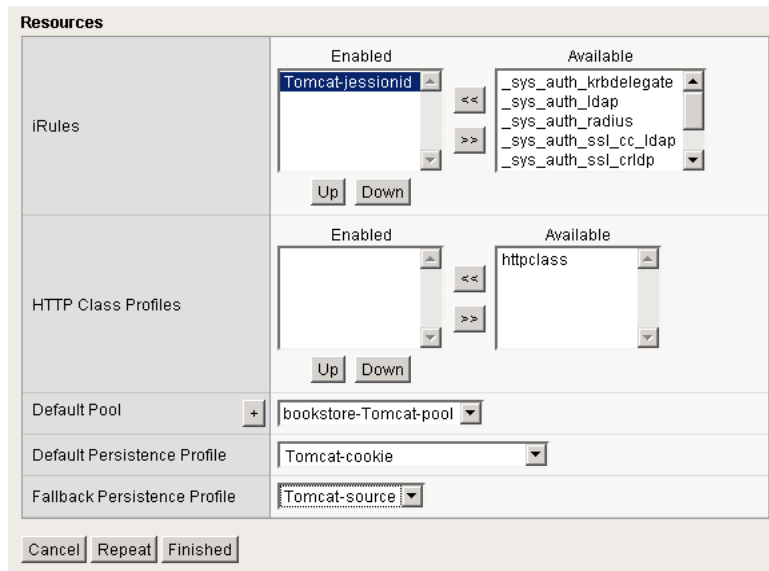


Figure 6 Adding the Pool and persistence profiles to the virtual server

15. Click the **Finished** button.
The BIG-IP LTM HTTP configuration for Tomcat deployment is now complete.
16. Repeat this entire procedure for each Tomcat context you are creating.

If you are using the BIG-IP system to offload SSL, continue with the following section.

Appendix A: Configuring the BIG-IP LTM to offload SSL

If you are using the BIG-IP LTM system to offload SSL from the Apache devices, there are additional configuration procedures you must perform on the BIG-IP LTM system.

◆ Important

Offloading SSL is not typically required inside of the firewall(s). We are including the instructions here for reference.

This section is optional, and only necessary if you are using the BIG-IP LTM system for offloading SSL.

In the following configuration, the BIG-IP LTM redirects all incoming traffic to the HTTP virtual server to the HTTPS virtual server. This is useful if a user types a URL in a browser, but forgets to change the protocol to HTTPS.

If your deployment does not require *all* traffic to be redirected to HTTPS, you do not need to configure the iRule or modify the HTTP virtual server as described below, nor configure the Rewrite Redirect setting in the HTTP profile in Step 5 of *Creating an HTTP profile*. You can have both an HTTP and HTTPS virtual server on the same address with the appropriate ports.

This section also includes an optional SSL persistence profile, see *Creating the SSL persistence profile*, on page 21.

Using SSL certificates and keys

Before you can enable the BIG-IP LTM system to act as an SSL proxy, you must install a SSL certificate on the virtual server that you wish to use for Apache connections on the BIG-IP LTM device. For this Deployment Guide, we assume that you already have obtained an SSL certificate, but it is not yet installed on the BIG-IP LTM system. For information on generating certificates, or using the BIG-IP LTM to generate a request for a new certificate and key from a certificate authority, see the **Managing SSL Traffic** chapter in the *Configuration Guide for Local Traffic Management*.

Importing keys and certificates

Once you have obtained a certificate, you can import this certificate into the BIG-IP LTM system using the Configuration utility. By importing a certificate or archive into the Configuration utility, you ease the task of managing that certificate or archive. You can use the Import SSL Certificates and Keys screen only when the certificate you are importing is in Privacy Enhanced Mail (PEM) format.

To import a key or certificate

1. On the Main tab, expand **Local Traffic**.
2. Click **SSL Certificates**. The list of existing certificates displays.
3. In the upper right corner of the screen, click **Import**.
4. From the **Import Type** list, select the type of import (Certificate or Key).
5. In the **Certificate** (or **Key**) **Name** box, type a unique name for the certificate or key.
6. In the **Certificate** (or **Key**) **Source** box, choose to either upload the file or paste the text.
7. Click **Import**.

If you imported the certificate, repeat this procedure for the key.

Creating a Client SSL profile

The next step in this configuration is to create a Client SSL profile. This profile contains the SSL certificate and Key information for decrypting the SSL traffic on behalf of the servers.

To create a new Client SSL profile

1. On the Main tab, expand **Local Traffic**, and then click **Profiles**. The HTTP Profiles screen opens.
2. On the Menu bar, from the SSL menu, select **Client**. The Client SSL Profiles screen opens.
3. Click the **Create** button. The New Client SSL Profile screen opens.
4. In the **Name** box, type a name for this profile. In our example, we type **Tomcat-clientssl**.
5. In the Configuration section, check the **Certificate** and **Key Custom** boxes.
6. From the **Certificate** list, select the name of the Certificate you imported in the *Importing keys and certificates* section.
7. From the **Key** list, select the key you imported in the *Importing keys and certificates* section.
8. Click the **Finished** button.

Creating the SSL persistence profile

You can create an optional SSL persistence profile.

- ◆ You *can* use SSL persistence with the following configurations:

- With an SSL virtual server, when the nodes are configured with the SSL certificate.
- With a virtual server configured with a `clientssl` profile, when the BIG-IP system terminates SSL connections.
- ◆ You *cannot* use SSL persistence with the following configurations:
 - With a virtual server configured with a `serverssl` profile. If the BIG-IP is configured to terminate and re-encrypt SSL connections, a different SSL session ID is used for the node-side connection than is used for the client-side connection. As a result, you cannot use SSL session ID persistence in combination with re-encryption.
 - With a virtual server configured for **Client Authentication**. For example, if the `clientssl` profile is configured to request a client SSL certificate for client authentication you cannot use SSL persistence.

To configure an SSL persistence profile

1. On the Main tab, expand **Local Traffic**, and then click **Profiles**.
2. On the Menu bar, click **Persistence**.
3. Click the **Create** button. The New Monitor screen opens.
4. In the **Name** box, type a name for the Monitor.
In our example, we type **Tomcat-SSL-persistence**.
5. From the **Type** list, select **SSL**.
6. Configure the options as applicable for your implementation.
7. Click **Finished**.

Modifying the HTTP virtual server

The next task is to modify the HTTP virtual server you created in *Creating the virtual server*, on page 1-17 to use the iRule you just created.

To modify the existing Tomcat virtual server

1. On the Main tab, expand **Local Traffic**, and then click **Virtual Servers**. The Virtual Servers screen opens.
2. From the Virtual Server list, click the Apache virtual server you created in the *Creating the virtual server* section.
In our example, we click **Tomcat-bookstore-vs**.
3. On the menu bar, click **Resources**.
4. From the **Default Pool** list, select **None**.
This virtual server no longer requires the load balancing pool, as traffic is redirected to the HTTPS virtual server we create in the following procedure.
5. Click the **Update** button.
6. In the iRules section, click the **Manage** button.

7. From the **Available** list, select **_sys_https_redirect**.
8. Click the **Finished** button.

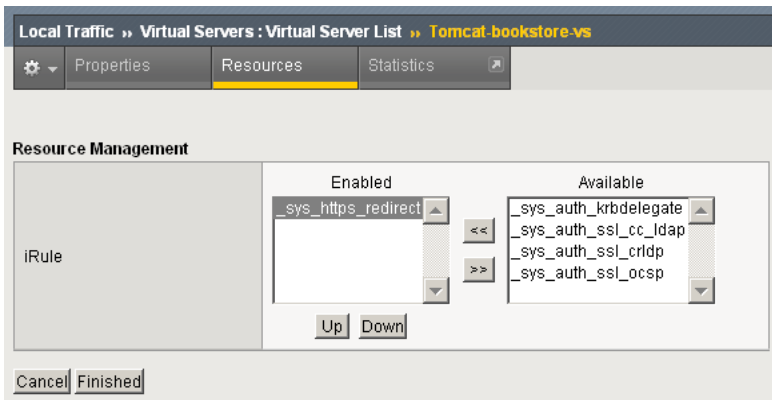


Figure 7 Adding the Redirect iRule to the HTTP virtual server

Creating the HTTPS virtual server

The final task in this section is to create a HTTPS virtual server.

To create a new HTTPS virtual server

1. On the Main tab, expand **Local Traffic**, and then click **Virtual Servers**. The Virtual Servers screen opens.
2. Click the **Create** button.
3. In the **Name** box, type a name for this virtual server. In our example, we type **Tomcat-bookstore-ssl-vs**.
4. In the **Destination** section, select the **Host** option button.
5. In the **Address** box, type the IP address of this virtual server. In our example, we use **192.168.104.146**.
6. In the **Service Port** box, type **443** or select **HTTPS** from the list.
7. From the Configuration list, select **Advanced**.
8. Leave the **Type** list at the default setting: **Standard**.
9. From the **Protocol Profile (Server)** list, select the name of the profile you created in the *Creating the TCP profile* section. In our example, we select **Tomcat-tcp-lan**.
10. From the HTTP Profile list, select the name of the profile you created in the *Creating an HTTP profile* section. In our example, we select **Tomcat-http-opt**.
Make sure you have the Rewrite Redirect box checked in the HTTP profile as described in Step 5 of *Creating an HTTP profile*.

11. From the **SSL Profile (Client)** list, select the name of the SSL profile you created in the *Creating a Client SSL profile* section. In our example, we select **Tomcat-clientssl**.
12. From the **Default Pool** list, select the pool you created in the *Creating the pool* section. In our example, we select **Tomcat-app-pool**.
13. From the **Default Persistence Profile** list, select the persistence profile you created in the *Optional: Creating persistence profile*. In our example, we select **Tomcat-SSL-persistence**.
14. From the **Fallback Persistence Profile** list, select the fallback persistence profile you created in the *Optional: Creating persistence profile*. In our example, we select **Tomcat-source**.
15. Click the **Finished** button.

This completes the BIG-IP LTM configuration.

To provide feedback on this deployment guide or other F5 solution documents, contact us at solutionsfeedback@f5.com.