

A general practice often seen in development environments is to have a web server to cater to the static pages and use the application server to deal with dynamic content. Pramati Server allows developers to integrate a number of third party products with its Server through a plug-in called WebGate.

In this chapter, we will look at how to configure Pramati Server WebGate, a plug-in for Apache. WebGate is available for Apache HTTP Servers as well as Microsoft IIS. The plug-in is provided by Pramati and can be downloaded from the website (<http://www.pramati.com/>). Apart from the normal behavior shown by the plug-in (i.e. redirecting pages to other back-end servers), the plug in can also be configured to perform LoadBalancing. We will cover both of these scenarios.

Apache is one of the most widely used webservers available for both Windows and Unix platforms. The WebGate plug-in allows requests to be proxied from an Apache Server to Pramati Server. WebGate enhances an existing Apache installation by providing a method to transparently access Pramati servlet engine.

Version and Platform Support

WebGate can be used with Apache Server 1.3.2x or 2.0 on Solaris, Linux, and Windows platforms. WebGate is installed as a module on Apache Server to load dynamic content from Pramati Server. WebGate is available for each of these platforms as:

Platform	WebGate File Name
Windows NT, Windows 2000	WGApache1.3Win.dll for Apache Versions 1.3 WGApache2.0Win.dll for Apache Versions 2.0
Solaris	WGApache1.3Solaris.so for Apache Version 1.3 WGApache2.0Solaris.so for Apache Version 2.0
Linux	WGApache1.3Linux.so for Apache Version 1.3 WGApache1.3Linux.so for Apache Version 2.0

Pramati WebGate for Apache is a plug-in that can be used in an environment where the Apache Server serves static pages while the dynamic content is redirected to the Pramati Server. WebGate is provided by Pramati and can be downloaded from <http://www.pramati.com/>. It is distributed as a dynamic shared object (DSO). The downloadable WebGate plug-in archive contains the following files:

- The plug-in itself (both for Windows and *NIX based systems)

- A plug-in properties file (`pramatiplugin.props` for Apache)
- Library Jars that are used to configure the WebGate plug-in for load balancing
- A copy of a `web-config.xml` file used by the LoadBalancer (enabled with the plug-in).

Note: The WebGate plug-in for Apache 1.3 on Solaris does not support load balancing.

Installing WebGate for Apache HTTP Server

Before installing WebGate, you should configure Apache Server for dynamic shared object (DSO) support, since the plug-in is built as a DSO that is loaded when the Apache HTTP Server starts.

To verify that DSO is enabled on Apache Server, run `httpd -l` at the prompt for Unix based systems. For Windows users run `apache -l` at the `<install-root>/Apache/` directory. All modules loaded on the Apache Server are listed. Look for `mod_so.c`. If `mod_so.c` is not present, contact your system administrator to enable DSO support for Apache Server.

Installing Apache Server with DSO Support in Unix

To work with the WebGate plug-in, the Apache server has to be installed with the Dynamic Shared Object (DSO) support. Apache supports building modules as shared objects on all major Unix platforms (see section "Supported Platforms" in document `htdocs/manual/dso.html` for details). APACI has a nice way of enabling the building of DSO-based modules and automatically installing them:

```
$ ./configure      --prefix=/path/to/apache \
                  --enable-module=rewrite \
                  --enable-shared=rewrite

$ make
$ make install
```

Enabling the DSO Module on Windows

With Apache 1.3 make sure the line `AddModule mod_so.c` is un-commented (by removing the `#` symbol at the beginning of the line).

```
#AddModule mod_usertrack.c
#AddModule mod_unique_id.c
AddModule mod_so.c
```

Installing WebGate

To install WebGate on Apache Server, extract the contents of the archive into a convenient location on your local system. Let it be called `<webgate_home>`.

- 1 Copy `pramatiplugin.props` into the `<APACHE_ROOT>/conf/` directory.

- 2 Open `<APACHE_ROOT>\conf\httpd.conf` in any text editor to view the Apache Server configuration file. Add the following information in the `LoadModule` and `AddModule` sections respectively, in the configuration file:

```
LoadModule webgate_module /path/to/webgate_home/WGApachex.xWin.dll
AddModule mod_webgate.c (ignore this for Apache version 2.0)
```

- 3 Requests can be redirected to Pramati Server based on a URI Pattern.

To redirect requests to Pramati Server based on a URI pattern, specify the URI pattern as shown in the example below:

```
<Location /bankWeb/*>
    SetHandler webgate-handler
</Location>
<Location /bankWeb/im/*>
    SetHandler webgate-handler
</Location>
<Location /bankWeb/im/*/*>
    SetHandler webgate-handler
</location>
```

In the `<Location>` tag above, wild cards (*) can be used in the URI pattern. The * indicates all files under that directory.

Alternatively, the `LocationMatch` tag can also be used:

```
<LocationMatch /bankWeb>
    SetHandler webgate-handler
</LocationMatch>
```

In the above configuration snippet, all URI requests that contain the sub string `"/bankWeb"` in its URI would be redirected to the `webgate-handler`. Note that in the `<LocationMatch>` entry, wildcards (*) cannot be used.

- 4 Save the configuration file and start/restart Apache Server.

WebGate is now plugged into Apache Server and allows you to load dynamic content from applications deployed on Pramati Server.

If you are using the WebGate plug-in 2.0, the following paths should be added to your system's path for Windows, and the `LD_LIBRARY_PATH` for Unix-based systems.

Operating System	File Path
Windows	<code><JAVA_HOME>/jre/bin/client</code>
Unix/Linux	<code><JAVA_HOME>/jre/lib/i386:<JAVA_HOME>/jre/lib/i386/client</code>
Solaris	<code><JAVA_HOME>/jre/lib/sparc</code>

Determining the Version of Plug-in

To determine the version of the WebGate Plug-in, please take a look at the log file. Here's a snippet from the log file that shows that the version of the plug-in is 2.1 and the build:

```
WebGate Plugin with Enhanced LoadBalancer for Apache2.0 Version 2.1
09-06-2004
```

Closing Idle Sockets on Pramati Server

With WebGate running, Pramati Server can be tuned for better performance with Apache. These parameters are mostly modified in the Pramati Server configuration file, `web-config.xml`, located under `<install-dir>\server\nodes\default\config` directory. The most significant property has to do with keeping idle sockets open on Pramati Server. The tag in the Server configuration file is shown below:

Table 1: Setting socket time out for Pramati Server with Apache

Parameter	Default Value	Set Tag in web-config.xml
keepalive-timeout-millis (in milliseconds)	1000 ms	<keepalive-timeout-millis>30000</keepalive-timeout-millis>

Configuring WebGate

The `pramatiplugin.props` file contains configuration parameters required for serving dynamic content from the Server. It is located in the directory `APACHE_ROOT/conf/`.

The parameters stored as key pairs are:

Parameter	Description
Logging	This parameter can take three values - <i>None</i> , <i>Medium</i> , or <i>Verbose</i> . The default log level is <i>medium</i> . Log files are not generated if this value is <i>None</i> . For log level <i>none</i> , no log file will be created.
MaxLogFileSizeInMB	Maximum size of each log file. If given less than 1 only one log is maintained
NoOfLogFiles	Number of old log files to be generated/kept after the size of the current log file exceeds its maximum size. <logfile> is the latest and the previous versions are renamed as <logfile>1, <logfile>2, <logfile>3,... and so on.
LogFileName	The logFile to which WebGate will write the logs. Here, the file name should be given relative to <APACHE_ROOT> directory. The default log file is <APACHE_ROOT>/logs/pramatiplugin.log. To override the default log file, the file path relative to" <APACHE_ROOT>/" should be provided. The string "<APACHE_ROOT>/" is attached by default the new logfilename.
AddClientHeaders	This option enables forwarding of client headers, this is useful to identify the IP headers. Add X-Forwarded-For Header.

WebGateHost	This is a comma separated list of machine names on which the Pramati Server is running. For example, <i>Host:port</i>
MaxSockPool	The maximum number of sockets used by the plug-in to connect Pramati Web Server. The default is 10.
EnhancedLBEnabled	To enable/disable the enhanced LoadBalancer which maintains Session stickiness, balances load in a round robin fashion and performs failover by resending the request to another node. It can also detect a restarted node. Default is false.
PluginInstallationHome	The plugin installation directory. The directory from which 'web-lbconfig.xml' is picked up. Note : this directory should also contain the required jars. Apart from these the web-config.xml file would be needed to configure enhanced LoadBalancing. Also, if tracing is used, the plugin-home directory should contain the trace.props file.

Using Connection Pooling of Pramati Server

Instead of creating a new socket from the plug-in to the web container of Pramati Server, a specified number of sockets can be created at start up and reused. This number is set in the `pramatiplugin.props` as `MaxSockPool` (see table above).

Failover of WebGate for Apache HTTP Server

When the Apache plug-in starts up, it tries to create a pool of connections to a node that is specified first in the comma separated list of nodes. If it successfully creates the connection, it serves the request using the pool.

When a node is shutdown, the plug-in performs a failover and tries to create a connection pool for the next node. If the connection fails or there is no response from any Server in the cluster, an error message is sent.

Sample Configuration

Following is a sample configuration file:

```
Logging medium
MaxLogFileSizeInMB 500
NoOfLogFiles 5
LogFileName logs/pramatiplugin.log
AddClientHeaders true
WebGateHost localhost:8181
MaxSockPool 10
EnhancedLBEnabled false
PluginInstallationHome D:/WebGateHome
```

Achieving Loadbalancing with the Apache plug-in

Pramati WebGate plug-in can be configured to balance load among several backend nodes. One advantage of configuring loadbalancing at the plug-in level is that it would avoid an extra hop to the Load balancer node and the loadbalancing takes place right at the plug-in level. To learn more about concepts of loadbalancing refer to the chapter on *Request Dispatcher Architecture* in Pramati Server 3.5 Technical Reference Guide.

To enable loadbalancing of the WebGate Plug-in, unzip the downloadable archive at a convenient location on the machine on which it should be configured. Let this directory be `<webgate_home>`. In this section we will look at how to configure the WebGate plug-in to balance load.

The loadbalancing feature can be enabled at the plug-in level by enabling the `EnhancedLBEnabled` property. Here's a sample `plugin.props` with descriptions of each properties inline:

Here's a sample `pramatiplugin.props` with descriptions of each properties inline:

Property Name	Description
<code>EnhancedLBEnabled</code>	To enable/disable the enhanced LoadBalancer which maintains Session stickiness, balances load in a round robin fashion and performs failover by resending the request to another node. It can also detect a restarted node. If enabled, the configuration property <code>WebgateHost</code> from <code>pramatiplugin.props</code> will not be read. The nodes information will be picked up from <code>web-lbconfig.xml</code> file.
<code>PluginInstallationHome</code>	The plugin installation directory. The ' <code>web-lbconfig.xml</code> ' file needed to configure the enhanced LB and the <code>LB.jar</code> will be picked up from this directory. Make sure that this path is in the system path.
<code>EnhancedLBContexts</code>	If ' <code>EnhancedLBEnabled</code> ' is enabled, then a list of contexts which need to be redirected to the back-end Pramati Servers is specified here. Eg. <code>EnhancedLBContexts /admin</code> . Use '@' for all contexts.

Note that the `web-lbconfig.xml` file (in the `<webgate_home>` directory) should be configured to provide information about various backend nodes and their port numbers. To learn more about configuring backend nodes, refer to the chapter Administration Guide.

Enabling Tracing for LoadBalancing

Pramati Server provides a tracing mechanism where by developers can troubleshoot or debug problems that may occur while using loadbalancing with the WebGate. When the plugin is started with Enhanced LB enabled, messages produced by the loadbalancer are present in `<PluginInstallationHome>/ApachePlugin1.3.log` or `<PluginInstallationHome>/ApachePlugin2.0.log` file. The log file with property `LogFileNames` generates logging information generated by the plugin alone.

By default only messages that are marked severe are logged. Pramati Server provides a `trace.props` file that enables logging. This file should be available in the plugin home.

The following is a sample `trace.props` file:

```
#~~~~~  
#Trace.props  
#~~~~~  
handlers=custom_handler  
handler.custom_handler.type=console  
handler.custom_handler.details=  
handler.custom_handler.isAbsolutePath=true  
handler.custom_handler.appendIfFileExists=true  
handler.custom_handler.loglevel=all  
handler.custom_handler.contexttype.packages=.,true;  
enable.debug=true
```

Table 2: Trace Properties

Property	Description
<code>handlers</code>	This is just a place holder for the kind of handlers. There can be a number of handlers available doing various types of tracing. Here the entry <code>custom_handler</code> provides the reference to the current handler.
<code>handler.custome_handler.type</code>	Tracing can happen either to a file or to console. Enter file for logging trace messages to a file, and console to the command shell.
<code>handler.custom_handler.details</code>	The path to the file where the trace messages should be logged in case of a file.
<code>handler.custom_handler.isAbsolutePath</code>	Set this to true if the path given above is an absolute path, and false otherwise.
<code>handler.custom_handler.appendIfFileExists</code>	This can be either true or false. If true, trace methods are appended at the end of the file, if it exists. If set to false, the trace file is overwritten.
<code>handler.custom_handler.loglevel</code>	Loglevels can be specified here. The following log levels are available : event, debug, off, info, user, finer, severe, warning, all.
<code>handler.custom_handler.contexttype.packages</code>	All relevant trace methods that occur in a given package are logged in the trace file.
<code>enable.debug</code>	Set this to true if debug messages should appear while tracing. Valid values are true or false.

