



Craig Scott

WebSphere Application Server V6: Web Server Plug-in Problem Determination

This paper discusses techniques for diagnosing problems that are related to the Web server plug-in for WebSphere Application Server V6, including those resulting from configuration errors. The Web server plug-ins handle communications between a Web server and the Web container in an application server or servers. Symptoms of a problem that involve the plug-in include:

- ▶ Users cannot access an application through the Web server
- ▶ Load balancing and failover are not working properly
- ▶ Session data is being lost
- ▶ Slow or intermittent application response
- ▶ The Web server will not start after plug-in installation or configuration

The Web server plug-in works with six different Web servers. However, this paper concentrates on the IBM® HTTP Server and its plug-in. It does not address Web server problems or problems that are related to the application server Web container.

Important: We recommend that you start your problem determination process by reading *Approach to Problem Determination in WebSphere Application Server V6* at <http://www.redbooks.ibm.com/redpapers/pdfs/redp4073.pdf>.

Introduction

Figure 1 shows how the Web server plug-in interacts with both the Web server and the Web container in the application server. Note that the Web server and application server need not reside on the same physical server.

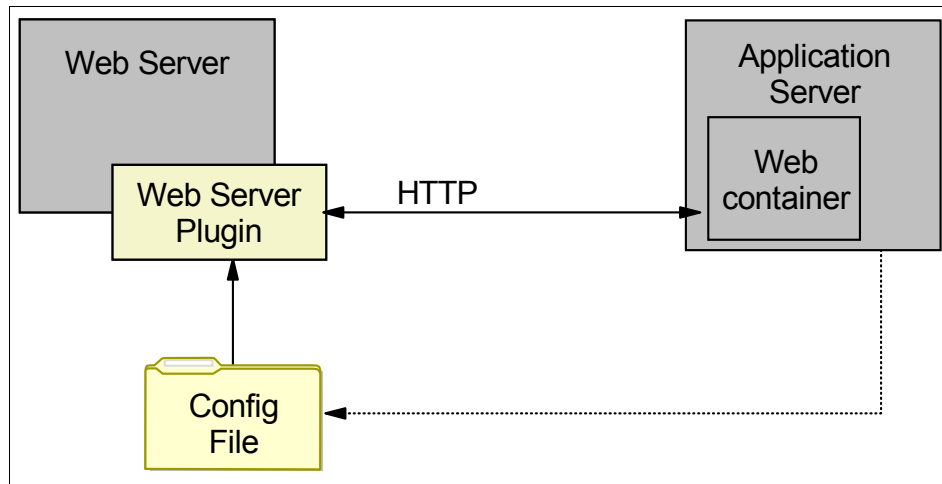


Figure 1 Web server plug-in interactions

As illustrated in Figure 1, the problem determination techniques in this paper focus on the Web server plug-in and its configuration file. Installing the plug-in requires that you change the Web server configuration and that you create the plug-in configuration file from using the WebSphere® administrative tools. Thus, this paper also discusses these areas.

We look at the following symptoms that a user would see:

- ▶ Users cannot reach the application (page not found and page cannot be displayed).
- ▶ Session data is lost (the user is asked to log on every time and shopping cart data lost).
- ▶ Slow application response, intermittent availability.

We also look at the following symptoms that an administrator would see:

- ▶ Web server will not start.
- ▶ Applications are not distributed properly to servers in clustered environments.

All of the Web servers with which the Web server plug-in is compatible allow external modules to be loaded by the Web server at runtime; the Web server

plug-in is one of these modules. For example, the IBM HTTP Server uses two directives in its configuration file to load the plug-in module and to refer to the plug-in configuration file.

In Example 1, the first directive loads the module and the second directive tells the plug-in module where to find the plug-in configuration file. The plug-in configuration file is an XML document that tells the Web server how to handle requests for URLs that are sent to the WebSphere Application Server.

Example 1 IBM HTTP Server plug-in directives

```
LoadModule was_ap20_module "C:\IBM\WAS6\Plugins\bin\mod_was_ap20_http.dll"  
WebSpherePluginConfig "C:\IBM\WAS6\Plugins\config\webserver1\plugin-cfg.xml"
```

The plug-in intercepts all requests that are received by the Web server . It compares the host name and port of the incoming URL to those that are defined in the VirtualHostGroup, and the requested URI to the list of URIs that are defined in the UriGroup. If a match is found in both the VirtualHostGroup and UriGroup, the plug-in then looks at the Route to determine what ServerCluster to use. When the Route is determined, the plug-in sends the request to the WebSphere Application Server that is defined in the ServerCluster. In Example 2, a request for http://server/snoop is directed to WebSphere Application Server. All other requests are passed back to the Web server to handle.

Example 2 Excerpt from the plug-in config file

```
<VirtualHostGroup Name="default_host">  
  <VirtualHost Name="*:80"/>  
</VirtualHostGroup>  
<UriGroup Name="default_host_cluster1_URIs">  
  <Uri AffinityCookie="JSESSIONID" AffinityURLIdentifier="jsessionid"  
    Name="/snoop/*"/>  
</UriGroup>  
<Route ServerCluster="cluster1"  
  UriGroup="default_host_cluster1_URIs"  
  VirtualHostGroup="default_host"/>
```

The plug-in configuration file is managed via the WebSphere administrative console. Whenever you make a change that affects the plug-in, you need to regenerate the plug-in. Changes that affect the plug-in configuration include:

- ▶ Changing the virtual hosts definitions
- ▶ Changing the HTTP transport type or port
- ▶ Creating or deleting a server
- ▶ Installing or uninstalling an enterprise application

Tip: You can set your server to generate and propagate the plug-in automatically. Refer to the WebSphere Information Center for further details.

Load balancing

The Web server plug-in can perform load balancing across multiple applications servers in a cluster.

Figure 2 shows a basic load balanced configuration involving three servers. The Web server plug-in is installed on the Web server machine and directs requests to each application server in turn, either on a round-robin or random basis. This is only one of many topologies that can be used for load balancing.

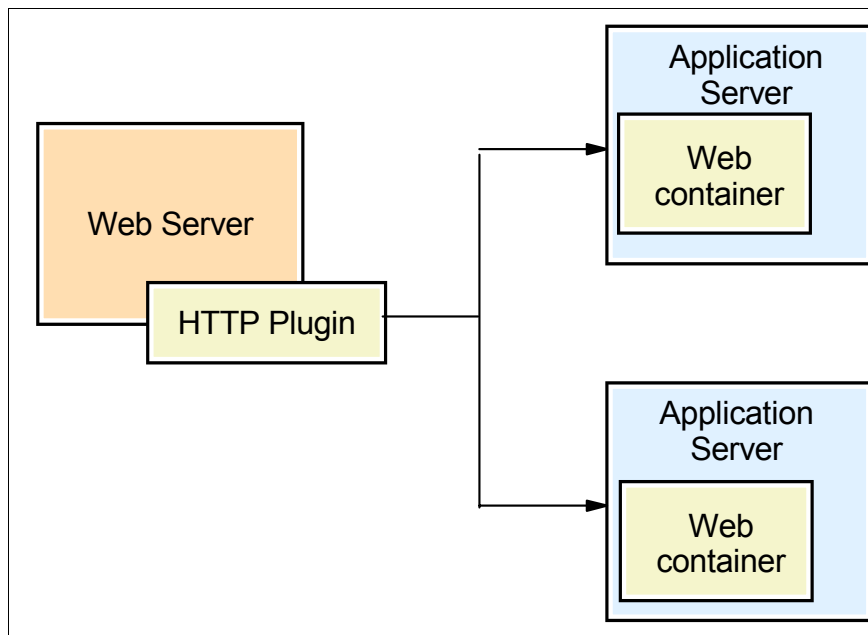


Figure 2 Load balancing

For more information, see *WebSphere Application Server V6 Scalability and Performance Handbook*, SG24-6392 at:

<http://publib-b.boulder.ibm.com/abstracts/sg246392.html>

In addition to defining the URLs that should be sent to the Web server plug-in, the plugin-cfg.xml file also describes the server clusters. Example 2 on page 3 shows the entries for the snoop URL and tells the plug-in that the snoop application should be handled by a UriGroup named default_host_cluster1_URIs.

Example 3 highlights the route entry that shows applications from the UriGroup named default_host_cluster1_URIs will be handled by a cluster named cluster1. The cluster named cluster1 has two separate servers. The first server is accessed via HTTP on port 9080 and HTTPS on port 9443, while the second server is accessed by HTTP on port 9081 and HTTPS on port 9444.

Example 3 Clustering excerpts from the plug-in configuration file

```
<Route ServerCluster="cluster1" UriGroup="default_host_cluster1_URIs"
  VirtualHostGroup="default_host"/>
...
<ServerCluster CloneSeparatorChange="false" LoadBalance="Round Robin" Name="cluster1"
PostSizeLimit="-1" RemoveSpecialHeaders="true" RetryInterval="60">
  <Server CloneID="10ig7jdv" ConnectTimeout="0" ExtendedHandshake="false"
LoadBalanceWeight="2" MaxConnections="-1" Name="k116571Node01_server1" ServerIOTimeout="0"
WaitForContinue="false">
    <Transport Hostname="k116571" Port="9080" Protocol="http"/>
    <Transport Hostname="k116571" Port="9443" Protocol="https">
      <Property Name="keyring" Value="C:\ibm\was6\plugins\etc\plugin-key.kdb"/>
      <Property Name="stashfile" Value="C:\ibm\was6\plugins\etc\plugin-key.sth"/>
    </Transport>
  </Server>
  <Server CloneID="10ig7jfqj" ConnectTimeout="0" ExtendedHandshake="false"
LoadBalanceWeight="2" MaxConnections="-1" Name="k116571Node01_server2" ServerIOTimeout="0"
WaitForContinue="false">
    <Transport Hostname="k116571" Port="9081" Protocol="http"/>
    <Transport Hostname="k116571" Port="9444" Protocol="https">
      <Property Name="keyring" Value="c:\ibm\was6\plugins\etc\plugin-key.kdb"/>
      <Property Name="stashfile" Value="c:\ibm\was6\plugins\etc\plugin-key.sth"/>
    </Transport>
  </Server>
```

The result of this is that a request for the snoop application is directed to an application server, either to a server listening on port 9080 or another server listening on port 9081.

The plug-in monitors the requests that it sends to each server. If either server cannot be contacted, then that server is marked as down and taken out of the cluster until it is restored.

Load balancing is configured partly by settings under the Web server entry in the WebSphere administrative console and partly through updating the plugin-cfg.xml file directly. The WebSphere Information Center contains details on configuring the various settings, see *Clusters and workload management* at:

http://publib.boulder.ibm.com/infocenter/wasinfo/v6r0/index.jsp?topic=/com.ibm.websphere.nd.doc/info/ae/ae/crun_srvgrp.html

Session affinity

A J2EE™ application supports the concept of sessions. A *session* is a way to maintain data about a user over subsequent requests. For example, a shopping cart application can maintain the list of items that you have selected to buy as session data.

Sessions are managed at the Web container level. However, the Web server plug-in has a role to play in maintaining *session affinity*. When a user connects to an application that uses sessions, such as a shopping cart application, the Web container starts a session and passes the user's session ID back to the browser, usually in a cookie.

You can maintain sessions across the servers in a cluster by a variety of methods (such as, by writing session data to a shared database after every request). If session affinity is used, the plug-in routes subsequent requests to the same application server that started the session. It does this by internally maintaining a list of servers and session IDs. Using session affinity provides the best application performance because you are eliminating the database write and read for each request. Session data is instead maintained in the application server's memory.

You do not have to do anything to the Web server plug-in to enable session affinity. If you have enabled session support in the application server and your plug-in configuration file contains the CloneID parameter as in the following example, then you have session affinity:

```
<Server CloneID="10ig7jdvd" ...>
```

The plug-in generation process adds the CloneID parameter by default. This is how the plug-in identifies each application server.

Work the problem

You begin the problem determination process by collecting the appropriate data that is required to diagnose the problem. We give you a list of all the documentation that might be required and how to collect it. If you have limited ability to recreate the problem, you might want to collect all the documentation at once, before starting the problem determination process. If you would rather only collect the relevant data as you move through the process, the data that you need is pointed out at each step.

Next, you go through a series of questions and actions that will help you define the high-level symptoms that you are experiencing. Each of these steps leads you to a more detailed procedure that is designed to take you through the

process of collecting and analyzing data to determine the most likely source of the problem.

And lastly, we provide guidance on the next step to take for resolution, whether it be a support site, contacting IBM, information about configuration, or some other suggestion as to how to proceed.

Collect the data

The logs and traces that will help you diagnose plug-in related problems include the following:

- ▶ Web server log files
- ▶ Web server plug-in log
- ▶ Plug-in trace (logged to the Web server plug-in log)
- ▶ WebSphere Application Server SystemOut and SystemErr logs
- ▶ WebSphere Application Server console messages
- ▶ Network protocol analysis (sometimes referred to as an iptrace)

If the problem is difficult to recreate or disruptive to business operations, see “The next step” on page 33 for a complete list of documentation to collect before continuing. In particular, you should review the MustGather documents for a complete list of documentation that is required by IBM support.

Web server log files

The Web server plug-in supports five different Web servers running on various platforms. Most Web servers write two log files: an access log that contains details of all accesses to the Web server and an error log that contains details of any errors. The default location of the logs is as follows:

- ▶ IBM HTTP Server, Apache, and SunOne
 - Windows®
 - Access log: <Web_server_home>\logs\access.log
 - Error log: <Web_server_home>\logs\error.log
 - UNIX
 - Access log: <Web_server_home>/logs/access_log
 - Error log: <Web_server_home>/logs/error_log
- ▶ Microsoft® IIS
 - Windows
 - Access log: C:\WINNT\SYSTEM32\LogFiles\W3SVC1\<date>.log
 - Error log: Windows event log

► Domino® Web server

The Domino Web server logs to a database. Refer to the Domino documentation for information about how to view these logs.

Web server plug-in log

The plug-in also writes its own log, which can be found in the Web server plug-in install directory path. The log you are looking for is under another directory structure that includes the logical name of the Web server that you chose when you installed the plug-in.

You can find the location of the log file by first looking at the Web server configuration. This refers to the plug-in configuration file as shown in Example 1 on page 3. The plug-in configuration file then tells you where the log file is as shown in Example 4. This example also shows you where you set the amount of detail that is logged.

Example 4 Location of plug-in log file

```
<Log LogLevel="Error"  
Name="c:\ibm\was6\plugins\logs\webserver1\http_plugin.log" />
```

The default LogLevel is Error, but you can set it to Trace to collect significantly more information. Should you need to raise this problem with IBM Support, they will request a plug-in trace.

Plug-in trace

To get an effective trace, you need to enable as much logging as possible on the Web server. For example, you can set the logging level to debug to capture verbose output in the IBM HTTP Server by modifying the LogLevel directive in the configuration file as shown:

```
LogLevel debug
```

You need to restart the IBM HTTP Server for this change to take effect.

Enable trace logging in the Web server plug-in by setting the LogLevel directive in the plugin-cfg.xml file as shown:

```
<Log LogLevel="Trace"  
Name="c:\ibm\was6\plugins\logs\webserver1\http_plugin.log" />
```

You do not need to restart the IBM HTTP Server for this change to take effect.

Tip: The plug-in trace generates significant amounts of data. Make your test as specific as possible and run it in isolation to reduce the number of lines that are generated.

WebSphere Application Server logs

You also need to look at the application server log files to determine if your application is actually being called by the plug-in but failing at the application server. You need to check the log files for the application server where the application is running. The log files are:

```
<WAS_install_root>/profiles/<profile>/logs/<server>/SystemOut.log  
<WAS_install_root>/profiles/<profile>/logs/<server>/SystemErr.log
```

Note: Applications often do some logging of their own to application specific log files.

Network trace

In rare cases, you might need to use a network protocol analyzer that will allow you to capture an iptrace. This might help you to determine where the problem lies. WebSphere Application Server does not supply such a tool, but there are third party tools available (for example, Ethereal that is available from <http://www.ethereal.com/>).

Analyze the high-level symptoms

Let's start working the problem by checking the following criteria in the order listed to define the high-level symptoms:

- ▶ You are not getting a response from the Web server.

Check to see if the Web server has started by looking for the process or by accessing the top level of the URL via a Web browser. For example:

```
http://localhost/
```

If you fail to get the Web server welcome page, check the operating system to see if the process for the Web server is active. If not, try starting the Web server manually.

If the Web server will not start, go to "Problem: Web server will not start" on page 11.

- ▶ The Web server has started, but you cannot access the application through it. For example, using the following URL to access the snoop servlet does not work:

```
http://Web_server/snoop
```

Check to see if the application can be accessed via the Web container directly. For example:

`http://Application_server:WC_port/snoop`

To access the application directly through the Web container:

1. Find the port for the Web container:
 - a. In the WebSphere administrative console, select **Servers** → **Application Servers**.
 - b. Click the server name.
 - c. Under the Communications section, expand Ports.
 - d. Use the port number listed for WC_defaulthost.
2. Use the port number to access the resource from a browser.

For example, if the port is 9080, the URL is:

`http://localhost:9080/snoop`

If the application can be accessed directly through the Web container but not through the Web server, go to “Problem: Failure between the Web server and plug-in” on page 13.

- ▶ The application relies on sessions, but these sessions appear to be getting lost between requests.

For example, if you are prompted to log in every time you access the application or saved data such as the items in a shopping cart are being lost, the application might be losing session data.

If you are experiencing this problem, go to “Problem: Sessions are being lost” on page 22.

- ▶ The application works sometimes but fails at others, and you have a clustered environment.

For example, two out of every three requests works, but the third times out.

If you are experiencing this problem, go to “Problem: The application works intermittently” on page 27.

- ▶ The application load is not being evenly distributed in a clustered environment.

For example, you might be seeing 80% CPU on one server in a cluster and very low usage on the other servers.

If you are experiencing this problem, go to “Problem: Application load is not being evenly distributed” on page 30.

If you are having a problem with the Web server plug-in that is not covered in this list, see “The next step” on page 33).

Analyzing problem areas

Your analysis of the data you gathered will most likely lead you to one of the following problem areas. If not, see “The next step” on page 33.

Problem: Web server will not start

If you have recently installed the plug-in, or regenerated the plug-in configuration and the Web server will not start, the problem could be related to the plug-in.

Data to collect

The following logs are required to determine why the Web server will not start:

- ▶ Operating system messages
- ▶ Web server logs
- ▶ Web server plug-in log

You should collect these logs and copy them to a place where you can view them. This is important because the original logs might be overwritten during the debugging process.

In some instances, the error message that identifies the problem might not appear in any logs. In this case, you need to start the Web server process from the command line and observe the messages that are written to the console directly.

What to look for

Browse through the logs for messages that indicate why the Web server did not start. In the event that there are no failure messages, try running the executable to start the Web server directly to see the message.

Example 5 shows how you would see the message that is generated by the IBM HTTP Server when the plugin-cfg.xml file is missing.

Example 5 Starting the HTTP Server from the command line

```
C:\IBM\HTTP\bin>apache
ws_common: websphereUpdateConfig: Failed to stat plugin config file for
C:\IBM\WAS6\Plugins\config\webserver1\plugin-cfg.xml
```

In a UNIX environment, you normally see any relevant messages on the command line when trying to start the Web server. Example 6 shows the messages you see on UNIX if the plug-in module is in some way corrupt.

Example 6 Messages indicating a corrupt plug-in module on UNIX

```
Syntax error on line 844 of /opt/IBMIHS/conf/httpd.conf:  
Cannot load /opt/IBM/WAS6/Plugins/bin/mod_was_ap20_http.so into server:  
/opt/IBM/WAS6/Plugins/bin/mod_was_ap20_http.so: undefined symbol: ap_palloc
```

In Windows, you rarely see a useful error message when starting the Web server from the Windows services panel. You need to look for the log files that are written by the Web server or the plug-in. A message similar to that shown in Example 6 appears in the http_plugin.log file in a Windows environment.

When the Web server plug-in stops the Web server from starting, the error messages are generally quite specific and accurate. When you have resolved the issue in the message, you will be able to start the Web server.

If the error message refers to a file, ensure that the file exists at the path shown in the message and that it is not corrupt. Try doing this as the user who runs the Web server to ensure that the problem is not related to file or directory permissions.

If the message refers to a problem with the plugin-cfg.xml file, you could try regenerating the plug-in from the WebSphere administrative console.

The plug-in is only tested with certain releases of the Web servers. If you have old or unsupported versions of the plug-in, Web server or GSKit, then you might need to upgrade that component. The following URL links to the software requirements page for WebSphere Application Server:

http://publib.boulder.ibm.com/infocenter/wasinfo/v6r0/index.jsp?topic=/com.ibm.websphere.base.doc/info/welcome_base.html

Tip: You can run the GSKit version command to get detailed version information. The command will be in the `<gskit_install>/bin` directory.

For example:

```
C:\Program Files\IBM\gsk7\bin\gsk7ver
```

If you do not see any messages similar to those shown in the examples in this section, then it is possible that the plug-in is not causing your Web server to fail. In this case, you should review the symptoms that led you to this conclusion.

Problem: Failure between the Web server and plug-in

If you have determined that the Web server is not sending HTTP requests to the WebSphere Application Server and you suspect the plug-in to be the cause of the problem, this section can help you determine the cause.

The most obvious symptom is the ability to reach the application directly through the Web container but not through the Web server.

The Web server error log will also contain a message that indicates that the file could not be found, as follows:

```
[Tue Jun 28 15:54:43 2005] [error] [client 127.0.0.1] File does not exist:  
C:/IBM/HTTP/htdocs/en_US/snoop
```

Data to collect

The following logs are required to determine why your application is not responding:

- ▶ Web server logs
- ▶ Web server plug-in log
- ▶ Plug-in trace (advanced debugging)
- ▶ Network trace (for advanced debugging)
- ▶ Plug-in configuration file

You should collect these logs and copy them to a place where you can view them. This action is important because the original logs might be overwritten during the debugging process.

What to look for

The first thing to check is that the plugin-cfg.xml file that the Web server is referencing contains the entries for the application that is not working. The Web server configuration will have an entry that points to the plugin-cfg.xml file that it is using:

```
WebSpherePluginConfig /opt/WAS6/Plugins/config/web1/plugin-cfg.xml
```

Open this file either in a text editor or Web browser, and check to see if the context root of the application that you are testing appears in the file. An example of this is shown in Example 2 on page 3.

Verify you are using the correct configuration file

If the application entry is missing, ensure that the configuration file the Web server is using is the same file that you generated. When you generate the Web server plug-in either through the WebSphere administrative console or the **GenPluginCfg.sh/bat** command, the output tells you the exact location of the

generated file. Figure 3 shows the location of the new plug-in file in the file system.

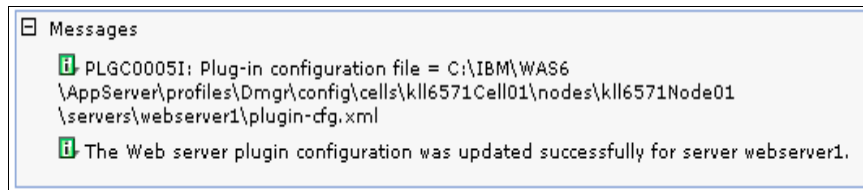


Figure 3 Location of generated plug-in file

You can compare this path to that in the Web server's configuration to ensure that they match:

```
WebSpherePluginConfig "C:\IBM\WAS6\Plugins\config\webserver1\plugin-cfg.xml"
```

In some configurations, the plug-in is propagated automatically to the correct locations on the local or remote host. The plug-in configuration file can only be propagated automatically if the following conditions are met:

- ▶ The remote Web server is IBM HTTP Server V6
- ▶ The plug-in configuration service is enabled
- ▶ Either
 - A WebSphere node agent must be running on the Web server machine
 - The IBM HTTP Server must be running on the Web server machine and an administrative password has been created
- ▶ The automatic propagation field must be checked as shown in Figure 4 on page 15

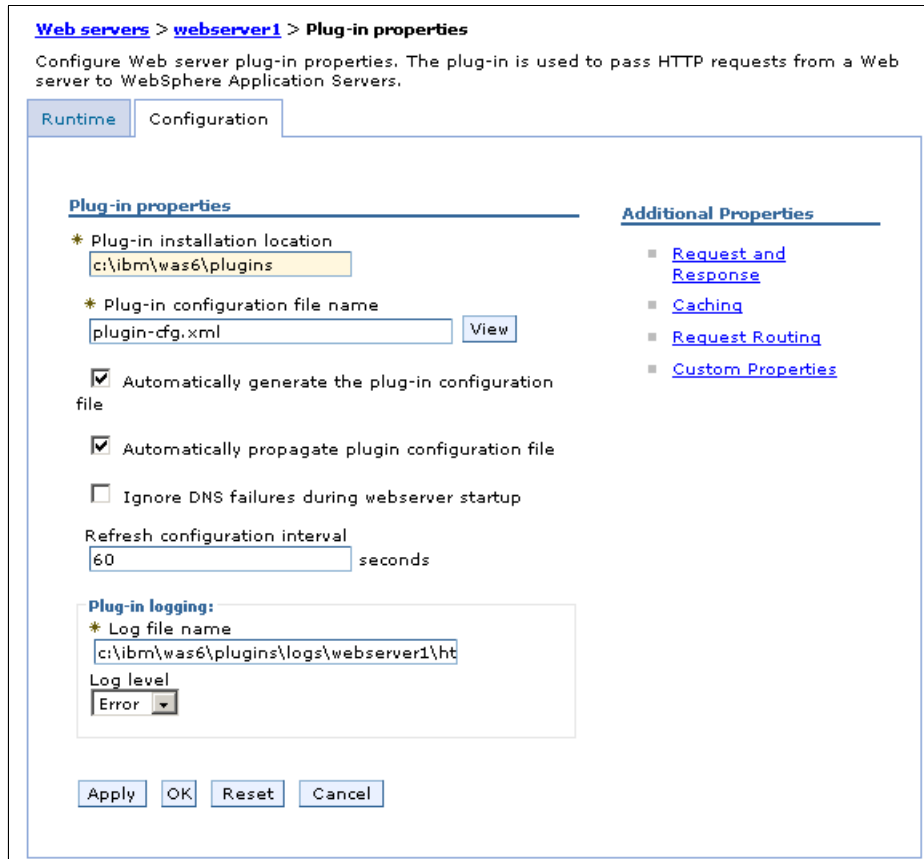


Figure 4 Plug-in configuration properties

For more information about automatic propagation of the plug-in, see *Web server plug-in properties* at:

http://publib.boulder.ibm.com/infocenter/wasinfo/v6r0/index.jsp?topic=/com.ibm.websphere.base.doc/info/aes/ae/uwsv_plugin_props.html

If the correct configuration file is in use, the next step is to try regenerating the Web server plug-in.

Verify that the plug-in configuration file is generated properly

If you have tried regenerating the plug-in, but the application's context root still does not appear in the generated plugin-cfg.xml file, you could be generating the configuration file incorrectly. The configuration included in a generated plugin-cfg.xml file depends on the method that you use to generate the file.

If you generate the Web server plug-in using the **GenPluginCfg.sh/bat** command line tool without parameters, every application and application server in the topology is generated into the configuration file. However, you can narrow the topology that is included in the configuration by using parameters. For example, if you generate the Web server plug-in using **GenPluginCfg** and specify a Web server name, or generate the plug-in through the administrative console, only those applications and application servers that are mapped to the Web server are generated into the configuration file.

The problem might be simply that you have not mapped the application or application server to the Web server. Use the administrative console to perform this mapping.

Check for Virtual Host and WebGroup not found error

If the Web server plug-in configuration appears to be correct, but the application is still not working, determine a Virtual Host / WebGroup not found message is being returned to the browser. If so, this error is covered in *WebSphere Application Server V6: Web Container Problem Determination* at:

<http://www.redbooks.ibm.com/redpapers/pdfs/redp4058.pdf>

Further analysis: Trace the request

At this point, you have confirmed that everything is configured correctly and as far as you can tell, the Web server plug-in should be sending your request to a WebSphere Application Server to process. You have verified that the plug-in configuration is correct and that the Web server is accepting the request. However, you are not getting any response from the application.

The next step is to trace a particular HTTP request through the topology to determine where the request is failing. You have to take a plug-in trace (see “Plug-in trace” on page 8).

After you have enabled the trace, send an HTTP request through the system, and check each log in turn to see where the transaction fails. Figure 5 on page 17 shows the path a HTTP request takes.

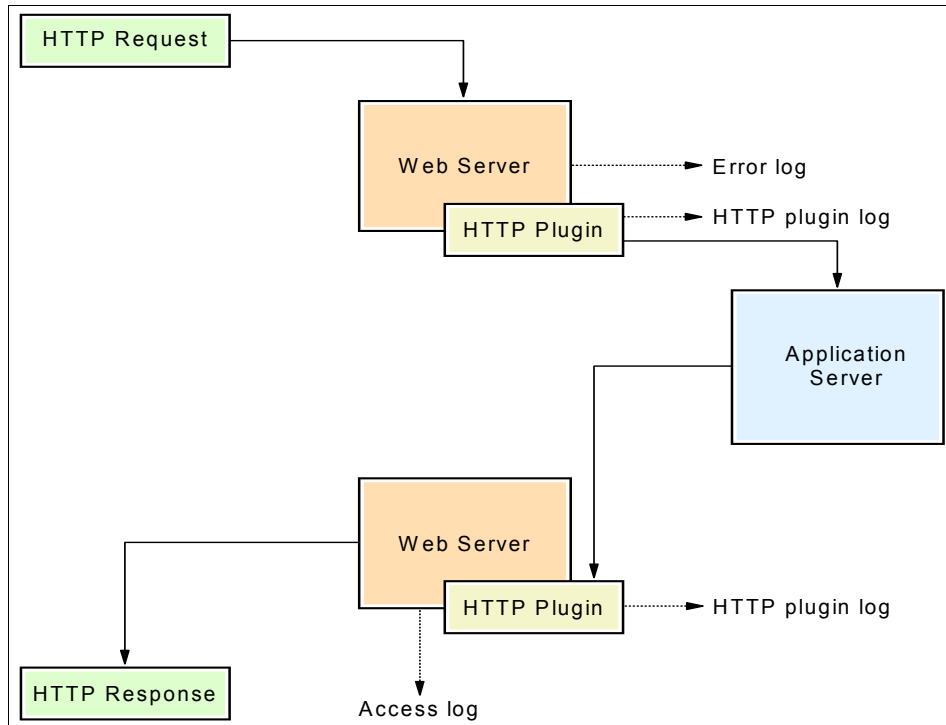


Figure 5 Following an HTTP request

Following a given request through that path shows you where the problems lie. Example 9 on page 18 shows the log entries that are generated by a successful request to the snoop servlet. These log entries show the successful transition of the request through each component.

The first entries written to http_plugin.log file show the plug-in building the structures it uses to parse URLs. If all is well, you can see it building the structures for your URL. Example 7 shows excerpts from the trace log for the snoop servlet.

Example 7 Plug-in trace initial entries

```
[Thu Jun 23 14:35:46 2005] 0000097c 00000ae4 - TRACE: ws_uri: uriCreate:
Creating uri
[Thu Jun 23 14:35:46 2005] 0000097c 00000ae4 - TRACE: ws_uri: uriSetName:
Setting the name /snoop/* with score 7
[Thu Jun 23 14:35:46 2005] 0000097c 00000ae4 - TRACE: ws_uri:
uriSetAffinityURL: Setting the affinity cookie jsessionid
[Thu Jun 23 14:35:46 2005] 0000097c 00000ae4 - TRACE: ws_uri:
uriSetAffinityCookie: Setting the affinity cookie JSESSIONID
```

```
[Thu Jun 23 14:35:46 2005] 0000097c 00000ae4 - TRACE: ws_uri_group:
uriGroupAddUri: Adding uri /snoop/* to front of list
```

After the plug-in has parsed the configuration file and is successfully initialized, it writes entries that include the build version and other data about the plug-in, as shown in Example 8.

Example 8 Plug-in initialization messages

```
-----System Information-----
[Thu Jun 23 14:35:46 2005] 0000097c 00000ae4 - PLUGIN: Bld version: 6.0.0
[Thu Jun 23 14:35:46 2005] 0000097c 00000ae4 - PLUGIN: Bld date: Oct 31 2004,
11:15:26
[Thu Jun 23 14:35:46 2005] 0000097c 00000ae4 - PLUGIN: Webserver:
IBM_HTTP_Server/6.0 Apache/2.0.47 (Win32)
[Thu Jun 23 14:35:46 2005] 0000097c 00000ae4 - PLUGIN: Hostname = KLL6571
[Thu Jun 23 14:35:46 2005] 0000097c 00000ae4 - PLUGIN: OS version 5.0, build
2195, 'Service Pack 4'
[Thu Jun 23 14:35:46 2005] 0000097c 00000ae4 - PLUGIN:
-----
```

The entries in Example 9 show the successful resolution of the URL that was requested by the plug-in. It shows the processing of the request through the selection of a server from the cluster through setting up the transport. It also shows that the request has been forwarded to an application server. Shortly after that, it shows the response coming back from the application server with a HTTP return code of 200, that is, successful. The time stamps show that the application server took seven seconds to process the request.

Example 9 Web server plug-in request trace entries for snoop

```
[Thu Jun 23 14:35:56 2005] 000007bc 000007d4 - TRACE: ws_common:
websphereShouldHandleRequest: trying to match a route for: vhost='localhost';
uri='/snoop'
...
[Thu Jun 23 14:35:56 2005] 000007bc 000007d4 - TRACE: ws_common:
websphereUriMatch: Found a match '/snoop' to '/snoop' in UriGroup:
default_host_cluster1_URIs with score 6
...
[Thu Jun 23 14:35:57 2005] 000007bc 000007d4 - TRACE: ws_common:
websphereExecute: Executing the transaction with the app server
...
[Thu Jun 23 14:35:57 2005] 000007bc 000007d4 - TRACE: lib_htrrequest:
htrrequestWrite: Writing the request:
[Thu Jun 23 14:35:57 2005] 000007bc 000007d4 - TRACE: GET /snoop HTTP/1.1
[Thu Jun 23 14:35:57 2005] 000007bc 000007d4 - TRACE: User-Agent: Wget/1.9
[Thu Jun 23 14:35:57 2005] 000007bc 000007d4 - TRACE: Host: localhost
[Thu Jun 23 14:35:57 2005] 000007bc 000007d4 - TRACE: Accept: */*
```

```

[Thu Jun 23 14:35:57 2005] 000007bc 000007d4 - TRACE: Connection: Keep-Alive
[Thu Jun 23 14:35:57 2005] 000007bc 000007d4 - TRACE: $WSIS: false
[Thu Jun 23 14:35:57 2005] 000007bc 000007d4 - TRACE: $WSSC: http
[Thu Jun 23 14:35:57 2005] 000007bc 000007d4 - TRACE: $WSPR: HTTP/1.0
[Thu Jun 23 14:35:57 2005] 000007bc 000007d4 - TRACE: $WSRA: 127.0.0.1
[Thu Jun 23 14:35:57 2005] 000007bc 000007d4 - TRACE: $WSRH: 127.0.0.1
[Thu Jun 23 14:35:57 2005] 000007bc 000007d4 - TRACE: $WSSN: localhost
[Thu Jun 23 14:35:57 2005] 000007bc 000007d4 - TRACE: $WSSP: 80
[Thu Jun 23 14:35:57 2005] 000007bc 000007d4 - TRACE: Surrogate-Capability:
WS-ESI="ESI/1.0+"
[Thu Jun 23 14:35:57 2005] 000007bc 000007d4 - TRACE: lib_htrequest:
htrequestWrite: Writing the request content
[Thu Jun 23 14:35:57 2005] 000007bc 000007d4 - TRACE: ws_common:
websphereExecute: Wrote the request; reading the response
[Thu Jun 23 14:35:57 2005] 000007bc 000007d4 - TRACE: lib_htresponse:
htresponseRead: Reading the response: 5397bc
...
[Thu Jun 23 14:35:57 2005] 000007bc 000007d4 - TRACE: lib_htresponse:
htresponseRead: Reading the response: 5397bc
[Thu Jun 23 14:36:04 2005] 000007bc 000007d4 - TRACE: HTTP/1.1 200 OK
[Thu Jun 23 14:36:04 2005] 000007bc 000007d4 - TRACE: Content-Type:
text/html; charset=ISO-8859-1
[Thu Jun 23 14:36:04 2005] 000007bc 000007d4 - TRACE: Content-Language:
en-US
[Thu Jun 23 14:36:04 2005] 000007bc 000007d4 - TRACE: Content-Length: 16166
[Thu Jun 23 14:36:04 2005] 000007bc 000007d4 - TRACE: lib_htresponse:
htresponseSetContentLength: Setting the content length |16166|

```

Immediately after this, the Web server sends the request back to the browser and then updates the HTTP access log to show that the transaction completed successfully, as shown in Example 10.

Example 10 Web server access log entry for snoop

```
127.0.0.1 - - [23/Jun/2005:14:35:56 -0400] "GET /snoop HTTP/1.0" 200 16166
```

The Web server writes to the access log at the end of processing.

Following your failing HTTP request through these log files shows you where the problem lies.

Example 11 on page 20 shows the messages that are written by the plug-in when it marks a server down. These messages appear even when you are not tracing requests. The message includes the error that is returned by the operating system, that is `err=10061`, which means connection refused. You should check your operating system documentation for a list of these error codes.

Example 11 Plug-in messages when a server goes down

```
[Thu Jul 07 13:53:20 2005] 0000d6c 000010b8 - ERROR: ws_common:
websphereGetStream: Failed to connect to app server on host 'k116571', OS
err=10061
[Thu Jul 07 13:53:20 2005] 0000d6c 000010b8 - ERROR: ws_common:
websphereExecute: Failed to create the stream
[Thu Jul 07 13:53:20 2005] 0000d6c 000010b8 - ERROR: ws_server:
serverSetFailoverStatus: Marking k116571Node01_server01 down
[Thu Jul 07 13:53:20 2005] 0000d6c 000010b8 - ERROR: ws_common:
websphereHandleRequest: Failed to execute the transaction to
'k116571Node01_server01' on host 'k116571'; will try another one
```

If the Web server plug-in trace shows it building the request and sending it to an application server, but does not show a reply coming back, then the problem is likely to be in the network between the Web server and the application server.

Example 12 shows the plug-in writing the request to a server and then waiting on the response. In this instance, the plug-in has been configured so that it times out waiting for a response after 10 seconds. The example shows the plug-in timing out the read and marking the server down.

Example 12 Marking a server down after time out

```
[Thu Jul 07 15:09:48 2005] 0000ed4 00000ffc - TRACE: lib_htrequest:
htrequestWrite: Writing the request content
[Thu Jul 07 15:09:48 2005] 0000ed4 00000ffc - TRACE: ws_common:
websphereExecute: Wrote the request; reading the response
[Thu Jul 07 15:09:48 2005] 0000ed4 00000ffc - TRACE: lib_htresponse:
htresponseRead: Reading the response: 4cf1504
[Thu Jul 07 15:09:58 2005] 0000ed4 00000ffc - TRACE: lib_htresponse:
htresponseSetError: Setting the error |1|
[Thu Jul 07 15:09:58 2005] 0000ed4 00000ffc - ERROR: ws_common:
websphereExecute: Failed to read from a new stream; App Server may have gone
down during read
```

In some instances, the plug-in does not detect that a server has gone down or hung after successfully establishing a connection and writing a request. This circumstance generally occurs when one server is Windows-based and the other is UNIX®-based.

Example 13 shows a request that responds normally. The plug-in writes the request and then waits for the response. The response arrives seven seconds later.

Example 13 A request that responds

```
[Thu Jul 07 14:27:00 2005] 0000a20 0000f80 - TRACE: lib_htrequest:
htrequestWrite: Writing the request content
[Thu Jul 07 14:27:00 2005] 0000a20 0000f80 - TRACE: ws_common:
websphereExecute: Wrote the request; reading the response
[Thu Jul 07 14:27:00 2005] 0000a20 0000f80 - TRACE: lib_htresponse:
htresponseRead: Reading the response: 5397cc
[Thu Jul 07 14:27:07 2005] 0000a20 0000f80 - TRACE: HTTP/1.1 200 OK
```

Example 14 shows where the plug-in writes the request and then waits for the response. Almost five minutes later, another request arrives and the plug-in begins parsing it. There was no response from the original request.

Example 14 A request that never responds

```
[Thu Jul 07 14:22:20 2005] 0000a20 0000f88 - TRACE: ws_common:
websphereExecute: Wrote the request; reading the response
[Thu Jul 07 14:22:20 2005] 0000a20 0000f88 - TRACE: lib_htresponse:
htresponseRead: Reading the response: 4cf1504
[Thu Jul 07 14:27:00 2005] 0000a20 0000f80 - TRACE: lib_util:
parseHostHeader: Host: 'localhost', port 80
```

At this point, you could use a network protocol analyzer to determine what is happening to the network packets after they leave the Web server. This is also called an *iptrace*. Network protocol analyzers are quite technical tools and generate a lot of output because there is a lot of network traffic, even on a seemingly idle network. If you are not confident about capturing and interpreting this output, you should consider engaging a network specialist to assist you.

Most problems that are perceived to be the fault of the plug-in are in fact configuration problems or network problems. Aside from the load balancing features, the plug-in is quite a simple component that simply takes a request, parses it for a match and then forwards matching requests to an application server.

If you have been through these diagnostic steps and not found the source of your problem, it is time to contact IBM Support (see “The next step” on page 33).

Problem: Sessions are being lost

If session data appears to be getting lost and you have a clustered environment, the plug-in is a potential source of the problem.

Data to collect

You should collect the following data:

- ▶ Plug-in trace
- ▶ Web server plug-in log

You should collect the log and copy it to a place where you can view it. This is important because the original log might be overwritten during the debugging process.

What to look for

The Web server plug-in trace shows you the processing of session affinity in detail. The sample applications that are provided with WebSphere Application Server allow you to test session affinity. The following URL displays a simple JSP™ demonstrating that WebSphere Application Server is alive and sets up a session:

`http://servername/HelloHTML.jsp`

The excerpt from the plug-in trace shown in Example 15 shows the session that is created. It shows the plug-in looking for the various cookies that it can use for session management and then using a round-robin load balancing algorithm because no session could be found.

Example 15 Creating a session

```
[Mon Jun 27 14:48:33 2005] 00000798 00000e00 - TRACE: ws_common:
websphereWriteRequestReadResponse: Enter
[Mon Jun 27 14:48:33 2005] 00000798 00000e00 - TRACE: ws_common:
websphereHandleSessionAffinity: Checking for session affinity
[Mon Jun 27 14:48:33 2005] 00000798 00000e00 - TRACE: ws_common:
websphereHandleSessionAffinity: Checking the SSL session id
[Mon Jun 27 14:48:33 2005] 00000798 00000e00 - TRACE: lib_htrequest:
htrequestGetCookieValue: Looking for cookie: 'SSLJSESSION'
[Mon Jun 27 14:48:33 2005] 00000798 00000e00 - TRACE: lib_htrequest:
htrequestGetCookieValue: No cookie found for: 'SSLJSESSION'
[Mon Jun 27 14:48:33 2005] 00000798 00000e00 - TRACE: ws_common:
websphereHandleSessionAffinity: Checking the cookie affinity: JSESSIONID
[Mon Jun 27 14:48:33 2005] 00000798 00000e00 - TRACE: lib_htrequest:
htrequestGetCookieValue: Looking for cookie: 'JSESSIONID'
[Mon Jun 27 14:48:33 2005] 00000798 00000e00 - TRACE: lib_htrequest:
htrequestGetCookieValue: No cookie found for: 'JSESSIONID'
```

```
[Mon Jun 27 14:48:33 2005] 00000798 00000e00 - TRACE: ws_common:
websphereHandleSessionAffinity: Checking the url rewrite affinity: jsessionid
[Mon Jun 27 14:48:33 2005] 00000798 00000e00 - TRACE: ws_common:
websphereParseSessionID: Parsing session id from '/HelloHTML.jsp'
[Mon Jun 27 14:48:33 2005] 00000798 00000e00 - TRACE: ws_common:
websphereParseSessionID: Failed to parse session id
[Mon Jun 27 14:48:33 2005] 00000798 00000e00 - TRACE: ws_common:
websphereHandleSessionAffinity: Bypassing check for partitionID cookie
affinity. No stored partition table.
[Mon Jun 27 14:48:33 2005] 00000798 00000e00 - TRACE: ws_server_group:
serverGroupNextRoundRobinServer: Round Robin load balancing
...
[Mon Jun 27 14:51:36 2005] 00000798 00000e18 - TRACE: ws_server_group:
serverGroupIncrementConnectionCount: Server k116571Node01_server2 picked,
pendingConnectionCount 1 totalConnectionsCount 5.
```

Example 16 shows a second request that is processed. This time, there is a session ID, and so the request is routed to the same server as that in Example 15 on page 22.

Example 16 Processing session affinity

```
[Mon Jun 27 14:51:51 2005] 00000798 00000e00 - TRACE: ws_common:
websphereHandleSessionAffinity: Checking for session affinity
[Mon Jun 27 14:51:51 2005] 00000798 00000e00 - TRACE: ws_common:
websphereHandleSessionAffinity: Checking the SSL session id
[Mon Jun 27 14:51:51 2005] 00000798 00000e00 - TRACE: lib_htrequest:
htrequestGetCookieValue: Looking for cookie: 'SSLJSESSION'
[Mon Jun 27 14:51:51 2005] 00000798 00000e00 - TRACE: lib_htrequest:
htrequestGetCookieValue: No cookie found for: 'SSLJSESSION'
[Mon Jun 27 14:51:51 2005] 00000798 00000e00 - TRACE: ws_common:
websphereHandleSessionAffinity: Checking the cookie affinity: JSESSIONID
[Mon Jun 27 14:51:51 2005] 00000798 00000e00 - TRACE: lib_htrequest:
htrequestGetCookieValue: Looking for cookie: 'JSESSIONID'
[Mon Jun 27 14:51:51 2005] 00000798 00000e00 - TRACE: lib_htrequest:
htrequestGetCookieValue: name='JSESSIONID',
value='00004RRFkLckLGWVw-37Cd-mEN7:10ig7jfqj'
[Mon Jun 27 14:51:51 2005] 00000798 00000e00 - TRACE: ws_common:
websphereParseCloneID: Parsing clone ids from
'00004RRFkLckLGWVw-37Cd-mEN7:10ig7jfqj'
[Mon Jun 27 14:51:51 2005] 00000798 00000e00 - TRACE: ws_common:
websphereParseCloneID: Adding clone id '10ig7jfqj'
[Mon Jun 27 14:51:51 2005] 00000798 00000e00 - TRACE: ws_common:
websphereParseCloneID: Returning list of clone ids
[Mon Jun 27 14:51:51 2005] 00000798 00000e00 - TRACE: ws_server_group:
serverGroupFindClone: Looking for clone
[Mon Jun 27 14:51:51 2005] 00000798 00000e00 - TRACE: ws_server_group:
serverGroupGetFirstPrimaryServer: getting the first primary server
```

```

[Mon Jun 27 14:51:51 2005] 00000798 00000e00 - TRACE: ws_server_group:
serverGroupFindClone: Comparing curCloneID '10ig7jfdvd' to server clone id
'10ig7jfdvd'
[Mon Jun 27 14:51:51 2005] 00000798 00000e00 - TRACE: ws_server_group:
serverGroupGetNextPrimaryServer: getting the next primary server
[Mon Jun 27 14:51:51 2005] 00000798 00000e00 - TRACE: ws_server_group:
serverGroupFindClone: Comparing curCloneID '10ig7jfdvd' to server clone id
'10ig7jfdvd'
[Mon Jun 27 14:51:51 2005] 00000798 00000e00 - TRACE: ws_server_group:
serverGroupFindClone: Match for clone 'k116571Node01_server2'
[Mon Jun 27 14:51:51 2005] 00000798 00000e00 - TRACE: ws_server:
serverHasReachedMaxConnections: currentConnectionsCount 0, maxConnectionsCount
-1.
[Mon Jun 27 14:51:51 2005] 00000798 00000e00 - STATS: ws_server_group:
serverGroupCheckServerStatus: Checking status of k116571Node01_server2,
ignoreWeights 1, markedDown 0, retryNow 0, wlbAllows -1
reachedMaxConnectionsLimit 0
[Mon Jun 27 14:51:51 2005] 00000798 00000e00 - TRACE: ws_server:
serverHasReachedMaxConnections: currentConnectionsCount 0, maxConnectionsCount
-1.
[Mon Jun 27 14:51:51 2005] 00000798 00000e00 - TRACE: ws_server_group:
serverGroupIncrementConnectionCount: Server k116571Node01_server2 picked,
pendingConnectionCount 1 totalConnectionsCount 7.
[Mon Jun 27 14:51:51 2005] 00000798 00000e00 - TRACE: ws_common:
websphereHandleSessionAffinity: Setting server to k116571Node01_server2

```

Reviewing a plug-in trace log shows whether session affinity is working correctly and whether the plug-in is routing sessions correctly to the servers.

If it appears that session affinity is not working, check the WebSphere Application Server configuration to ensure that the session management parameters are set correctly. Also, check the plugin-cfg.xml file to ensure that the CloneID parameter is set.

Figure 4 on page 15 shows the administrative console page for the WebSphere Application Server session configuration parameters. You get to this page by choosing **Application servers** → **servername** → **Web container settings** → **Session management**.

There are three different session tracking mechanisms:

- ▶ Enable SSL ID tracking, which can only be used when securing your applications with SSL.
- ▶ Enable cookies, which is the most common and generally accepted method of tracking sessions.
- ▶ Enable URL rewriting, which is only rarely used and has a high performance impact because every URL must be modified to include the session ID.

To enable sessions, you choose one or more of these session tracking mechanisms. The other parameters are for setting how long your sessions will last and other session management purposes.

The image shows a configuration dialog box titled "Configuration" with a "General Properties" section. Under "Session tracking mechanism:", there are four checkboxes: "Enable SSL ID tracking" (unchecked), "Enable cookies" (checked), "Enable URL rewriting" (unchecked), and "Enable protocol switch rewriting" (checked). Below this, the "Maximum in-memory session count:" is set to "1000 sessions". The "Allow overflow" checkbox is checked. Under "Session timeout:", the "Set timeout" radio button is selected, with a value of "30 minutes". The "Security integration" checkbox is unchecked. Under "Serialize session access:", the "Allow serial access" checkbox is unchecked, and the "Maximum wait time" is set to "5 seconds". The "Allow access on timeout" checkbox is checked. At the bottom, there are four buttons: "Apply", "OK", "Reset", and "Cancel".

Figure 6 WebSphere Application Server session configuration

You will only see errors or problems related to sessions with a plug-in trace.

Example 17 on page 26 shows a similar sequence of processing a request to that shown in Example 16 on page 23 except that the chosen server is down.

The plug-in then chooses another server in the cluster but the session on the server that is down is lost.

Example 17 Session lost after application server goes down

```
[Thu Jul 07 15:55:59 2005] 0000bcc 000010c0 - TRACE: ws_server_group:
serverGroupFindClone: Match for clone 'm23vnx60Craig01_server02'
...
[Thu Jul 07 15:55:59 2005] 0000bcc 000010c0 - TRACE: ws_server_group:
lockedServerGroupUseServer: Server m23vnx60Craig01_server02 picked, weight 0.
[Thu Jul 07 15:55:59 2005] 0000bcc 000010c0 - TRACE: ws_common:
websphereFindTransport: Finding the transport
[Thu Jul 07 15:55:59 2005] 0000bcc 000010c0 - TRACE: ws_common:
websphereFindTransport: Setting the transport(case 2): m23vnx60 on port 19082
[Thu Jul 07 15:55:59 2005] 0000bcc 000010c0 - TRACE: ws_common:
websphereExecute: Executing the transaction with the app server
[Thu Jul 07 15:55:59 2005] 0000bcc 000010c0 - TRACE: ws_common:
websphereGetStream: Getting the stream to the app server
[Thu Jul 07 15:55:59 2005] 0000bcc 000010c0 - TRACE: ws_transport:
transportStreamDequeue: Checking for existing stream from the queue
[Thu Jul 07 15:56:00 2005] 0000bcc 000010c0 - ERROR: ws_common:
websphereGetStream: Failed to connect to app server on host 'm23vnx60', OS
err=10061
[Thu Jul 07 15:56:00 2005] 0000bcc 000010c0 - TRACE: ws_common:
websphereGetStream: socket 6628 closed - failed to connect
[Thu Jul 07 15:56:00 2005] 0000bcc 000010c0 - ERROR: ws_common:
websphereExecute: Failed to create the stream
[Thu Jul 07 15:56:00 2005] 0000bcc 000010c0 - ERROR: ws_server:
serverSetFailoverStatus: Marking m23vnx60Craig01_server02 down
[Thu Jul 07 15:56:00 2005] 0000bcc 000010c0 - STATS: ws_server:
serverSetFailoverStatus: Server m23vnx60Craig01_server02 : pendingConnections 0
failedConnections 1 affinityConnections 1 totalConnections 0.
[Thu Jul 07 15:56:00 2005] 0000bcc 000010c0 - ERROR: ws_common:
websphereHandleRequest: Failed to execute the transaction to
'm23vnx60Craig01_server02'on host 'm23vnx60'; will try another one
...
[Thu Jul 07 15:56:00 2005] 0000bcc 000010c0 - TRACE: ws_server_group:
lockedServerGroupUseServer: Server k116571Node01_server01 picked, weight 0.
[Thu Jul 07 15:56:00 2005] 0000bcc 000010c0 - TRACE: ws_common:
websphereFindTransport: Finding the transport
[Thu Jul 07 15:56:00 2005] 0000bcc 000010c0 - TRACE: ws_common:
websphereFindTransport: Setting the transport(case 2): k116571 on port 9082
```

If session affinity appears to be working from the plug-in's point of view, then it is possible that the application is at fault. Review the application with the developers to ensure that they are handling sessions correctly.

Example 18 shows the entries from the plug-in trace when the application requires session affinity via cookies but the browser does not allow cookies. The plug-in is not able to maintain session affinity as it has no way of saving the session ID.

Example 18 Session failure as cookies are not allowed

```
[Thu Jul 07 15:59:04 2005] 00000978 00000ff4 - TRACE: ws_common:
websphereHandleSessionAffinity: Checking for session affinity
[Thu Jul 07 15:59:04 2005] 00000978 00000ff4 - TRACE: ws_common:
websphereHandleSessionAffinity: Checking the SSL session id
[Thu Jul 07 15:59:04 2005] 00000978 00000ff4 - TRACE: lib_htrequest:
htrequestGetCookieValue: Looking for cookie: 'SSLJSESSIONID'
[Thu Jul 07 15:59:04 2005] 00000978 00000ff4 - TRACE: lib_htrequest:
htrequestGetCookieValue: No cookie found for: 'SSLJSESSIONID'
[Thu Jul 07 15:59:04 2005] 00000978 00000ff4 - TRACE: ws_common:
websphereHandleSessionAffinity: Checking the cookie affinity: JSESSIONID
[Thu Jul 07 15:59:04 2005] 00000978 00000ff4 - TRACE: lib_htrequest:
htrequestGetCookieValue: Looking for cookie: 'JSESSIONID'
[Thu Jul 07 15:59:04 2005] 00000978 00000ff4 - TRACE: lib_htrequest:
htrequestGetCookieValue: No cookie found for: 'JSESSIONID'
[Thu Jul 07 15:59:04 2005] 00000978 00000ff4 - TRACE: ws_common:
websphereHandleSessionAffinity: Checking the url rewrite affinity: jsessionid
[Thu Jul 07 15:59:04 2005] 00000978 00000ff4 - TRACE: ws_common:
websphereParseSessionID: Parsing session id from '/HelloHTML.jsp'
[Thu Jul 07 15:59:04 2005] 00000978 00000ff4 - TRACE: ws_common:
websphereParseSessionID: Failed to parse session id
```

The WebSphere Information Center contains information about managing sessions in WebSphere Application Server clusters. In particular, see *Clusters and workload management* at:

http://publib.boulder.ibm.com/infocenter/wasinfo/v6r0/index.jsp?topic=/com.ibm.websphere.nd.doc/info/ae/ae/crun_srvgrp.html

Problem: The application works intermittently

In the event that users are having regular but inconsistent failures accessing the application and you have a clustered environment, you might have a plug-in problem.

Data to collect

The following logs are required to determine why your application is not responding:

- ▶ Web server logs
- ▶ Web server plug-in log
- ▶ WebSphere Application Server logs

If you cannot identify the cause of the problem from these logs, you need to take a plug-in trace. Trace entries are also written to the Web server plug-in log.

You should collect these logs and copy them to a place where you can view them. This action is important because the original logs might be overwritten during the debugging process.

What to look for

Intermittent failures in applications can be caused by the plug-in not detecting that a WebSphere Application Server has failed and thus continuing to route requests to that application server.

If the application server is down, the plug-in will not be able to establish a TCP/IP connection and will not be able to send the request. Under these conditions, the plug-in marks the server as down.

If the application server is hung, it can still accept requests but cannot respond to them. Under these conditions, the application server is not marked down until the operating system times out the connection. Depending on the operating system, this action could take up to 10 minutes. At some point after the plug-in has marked the server down, it checks again and might well find that the server is accepting requests again. In this case, the server is marked as up, at least until the next time out.

If the application server is not hung but running slowly, to the point where all TCP/IP connections have been used up in the application server, the plug-in is not able to establish a connection and marks the server as down.

A failure in an application will not be detected by the plug-in because the application server is accepting requests and responding to them. This is the way the plug-in is designed to work.

Note: With all of these conditions, you would not get a response when trying to access the application directly on the server, for example:

```
http://servername:9080/snoop
```

In a clustered environment, you need to try and access each server in the cluster in turn to determine which is hung. For further information about dealing with application server hangs, refer to *Approach to Problem Determination in WebSphere Application Server V6* at:

<http://www.redbooks.ibm.com/redpapers/pdfs/redp4073.pdf>

The cause of the problems that you are experiencing in this section is related to WebSphere Application Server not responding as you expect and so you should

be resolving that issue. However, you can minimize the impact of a problem in WebSphere Application Server by tuning the parameters that control how the plug-in responds to time-out conditions. You can use the following parameters to tune the plug-in:

- ▶ **RetryInterval**

The `RetryInterval` parameter sets how long the plug-in waits to check a server for availability after marking that server down.

- ▶ **ServerIOTimeout**

The `ServerIOTimeout` attribute of a server element enables the plug-in to set a time out value, in seconds, for sending requests to and reading responses from the application server. If a value is not set for the `ServerIOTimeout`, the plug-in uses blocked I/O to write the request to and read the response from the application server until the TCP connection times out. For example:

```
<Server Name="server1" ServerIOTimeout=300>
```

In this case, the plug-in waits 300 seconds (15 minutes) before timing out the TCP connection. Setting the `ServerIOTimeout` attribute to a reasonable value enables the plug-in to time out the connection sooner and transfers requests to another application server when possible.

When selecting a value for this attribute, remember that sometimes it might take a couple of minutes for an application server to process a request. Setting the value of the `ServerIOTimeout` attribute too low could cause the plug-in to send a false server error response to the client.

Note: The `ServerIOTimeout` attribute is ignored for a plug-in that is running on a Solaris™ platform.

- ▶ **ExtendedHandshake**

You should use the `ExtendedHandshake` parameter when there is a proxy firewall between the Web server plug-in and an application server. It forces the plug-in to do more extensive checking to determine if an application server should be marked as down.

- ▶ **MaxConnections**

The maximum number of concurrent connections that can be sent to an application server can be used to stop the plug-in from sending requests to a server that might be running slow.

- ▶ **ConnectTimeout**

The `ConnectTimeout` attribute of a `Server` element enables the plug-in to perform non-blocking connections with the application server. Non-blocking

connections are beneficial when the plug-in is unable to contact the destination to determine if the port is available or unavailable.

If no ConnectTimeout value is specified, the plug-in performs a blocking connect in which the plug-in sits until an operating system times out (as long as two minutes depending on the platform) and allows the plug-in to mark the server unavailable. A value of zero (0) causes the plug-in to perform a blocking connect. A value greater than zero (0) specifies the number of seconds you want the plug-in to wait for a successful connection. If a connection does not occur after that time interval, the plug-in marks the server down and fails over to one of the other servers defined in the cluster.

Where possible, use the administrative console to set these properties for a given Web server definition. Any manual changes that you make to the plug-in configuration file for a given Web server are overridden whenever the file is regenerated. You can update the RetryInterval parameter in the administrative console by navigating to **Web servers** → **webserver** → **Plug-in properties** → **Request routing** and set the RetryInterval as shown in Figure 7.



The image shows a screenshot of a web form in an administrative console. The form has a label "Retry interval" above a text input field. The input field contains the number "60" and is followed by the text "seconds". The input field has a light blue border and a small arrow on the right side, indicating it is a numeric input field.

Figure 7 Setting the RetryInterval parameter

All the other parameters must be set manually in the plugin-cfg.xml file. Thus, every time you regenerate the plug-in, you must edit the generated file to add these parameters.

For details on these and other plug-in properties, see *plugin-cfg.xml file* in the WebSphere Information Center at:

http://publib.boulder.ibm.com/infocenter/wasinfo/v6r0/index.jsp?topic=/com.ibm.websphere.express.doc/info/exp/ae/rwsv_plugincfg.html

Problem: Application load is not being evenly distributed

Uneven load distribution across servers in a cluster might be attributed to a plug-in problem.

Data to collect

Collect the following data:

- ▶ Plug-in trace
- ▶ Web server plug-in log

The plug-in trace data is written to the Web server plug-in log. You should collect the log and copy it to a place where you can view it. This action is important because the original log might be overwritten during the debugging process.

You need to analyze the trace to determine how many of each request was sent to each application server in the cluster.

What to look for

Each time the plug-in chooses an application server to which it sends a request, it writes a message in the trace log similar to the following:

```
TRACE: ws_server_group: serverGroupIncrementConnectionCount: Server servername
picked, pendingConnectionCount count totalConnectionsCount count.
```

You can manually count the number of times that the request is sent to each server in the cluster. Alternatively, you might write a simple script to parse this log and extract the data for you. This data shows exactly how many requests are being routed to each server in the cluster. If the distribution is not even, you should check the relative weight of each server in the cluster as defined in the plugin-cfg.xml file by the LoadBalanceWeight parameter:

```
<Server CloneID=cloneid ConnectTimeout=0 ExtendedHandshake=false
LoadBalanceWeight=3 MaxConnections=-1 Name=servername ServerIOTimeout=0
WaitForContinue=false>
```

Whenever the plug-in sends a request to an application server, it decrements a counter that it maintains for each server in a cluster. It starts with each server's LoadBalanceWeight. When a server's counter reaches zero, the plug-in stops sending requests to that server. When all server's counters reach zero, it resets them back to their starting value and continues. The default value for LoadBalanceWeight is 2.

For example, consider 2 servers (server1 and server2). Server1 has a LoadBalanceWeight of 2 and server2 has a LoadBalanceWeight of 4. Table 1 shows how the requests are sent to each server.

Table 1 Effect of differing load balancer weights

Request #	Sent to	server1 counter	server2 counter
-	-	2	4
1	server1	1	4
2	server2	1	3
3	server1	0	3
4	server2	0	2
5	server2	0	1
6	server2	0	0
	counters reset:	2	4
7	server1	1	4

The ratio of requests that are sent to servers is simply the ratio of the relative weights. So, in Table 1, two requests were sent to server1 and four requests were sent to server2, for a ratio of 1:2. This is equivalent to the relative weights, 2:4.

Compare the number of requests that are sent to each server in the cluster to the ratio of the load balance weight settings.

Other parameters might also influence load balancing. For example, if your applications are running slowly on one application server, then that server might reach its MaxConnections limit. So, the plug-in stops sending requests to it. This is clearly marked in the plug-in trace.

Session affinity, as discussed in “Problem: Sessions are being lost” on page 22, could also be contributing to uneven load balancing. Again, you can see messages that the plug-in is choosing a particular server to maintain session affinity.

If the number of requests that are sent to each server matches the ratio of the load balance weight settings, then plug-in load balancing is working correctly. The uneven CPU usage must be caused by some other factor, such as a rogue process on the server or a problem in the application.

If they do not match, then some other factor might be influencing the load balancing algorithm. For example, you might be reaching MaxConnections on one server, or a server might be marked down by the plug-in.

If you need more intelligent routing based on application server load or response time, you need to use another load balancing product, such as WebSphere Application Server Edge Component.

The next step

The symptoms and problem areas included in this paper are some that you are more likely to experience. However, there are other things that can go wrong, or the cause of the problem might be related to a component other than the Web server plug-in.

If, after going through this process, you still have an undiagnosed problem, it is recommended that you go back to *Approach to Problem Determination in WebSphere Application Server V6* at:

<http://www.redbooks.ibm.com/redpapers/pdfs/redp4073.pdf>

Review *Classify the problem and determine the root cause* to see if there are any other components that might be causing the problem.

If you feel sure that you have a plug-in related problem, there are things you can do before contacting IBM support. First, you should review the documentation that you have gathered for errors related to the plug-in that were not addressed in this paper and search support sites for information or fixes.

Next, you should collect all of the data that is outlined in the MustGather document for Web server plug-in problems and raise a problem record with IBM.

- ▶ MustGather for general problems with the Web server plug-in:
<http://www-1.ibm.com/support/docview.wss?rs=180&uid=swg21174894>
- ▶ MustGather for problems generating the plug-in configuration file:
<http://www-1.ibm.com/support/docview.wss?rs=180&uid=swg21199421>
- ▶ MustGather for problems installing the Web server plug-in:
<http://www-1.ibm.com/support/docview.wss?rs=180&uid=swg21199343>

Be sure to spell out all of the diagnostic work that you have done so far to minimize the amount of time it takes IBM Support to assist you in resolving your problem.

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:
IBM Director of Licensing, IBM Corporation, North Castle Drive Armonk, NY 10504-1785 U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrates programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. You may copy, modify, and distribute these sample programs in any form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to IBM's application programming interfaces.

This document created or updated on September 30, 2005.




Send us your comments in one of the following ways:

- ▶ Use the online **Contact us** review redbook form found at:
ibm.com/redbooks
- ▶ Send your comments in an email to:
redbook@us.ibm.com
- ▶ Mail your comments to:
IBM Corporation, International Technical Support Organization
Dept. HZ8 Building 662, P.O. Box 12195
Research Triangle Park, NC 27709-2195 U.S.A.

Trademarks

The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both:

Redbooks (logo) ™
Domino®

IBM®
Redbooks™

WebSphere®

JSP, J2EE, Solaris, and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

The following terms are trademarks of other companies:

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Microsoft, Windows, Win32, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other company, product, or service names may be trademarks or service marks of others.