Carla Sadtler
Rob Larsen
Giribabu Paramkusham

WebSphere Application Server V6.1: Web container problem determination

The runtime environment for Web components is called the *Web container*. If users receive unexpected results in a Web browser (such as errors or incorrect information), you might have a problem with the Web container.

Potential symptoms of a Web container problem can include:

- Users are not able to access a Web component.
- There is unexpected behavior when running a Web component.
- Errors occur when a Web component is started.
- There are problems with JSP™ compilation.
- Errors or exceptions are thrown during the running of a Web component.
- Messages that start with SRVE (Web container), JSPG (JSP), or JSFG (JSF pages) are received.

This paper takes you through the steps of diagnosing Web container problems for IBM® WebSphere® Application Server V6.1 on distributed and IBM i5/OS® platforms.

Introduction to Web containers

A Web container is a runtime environment for Web applications. It processes servlets, JSP files, and other types of server-side components. Each application server runtime has one logical Web container, which can be modified but not created or removed.

Web container overview

Figure 1 illustrates the Web container and its place in the application server.

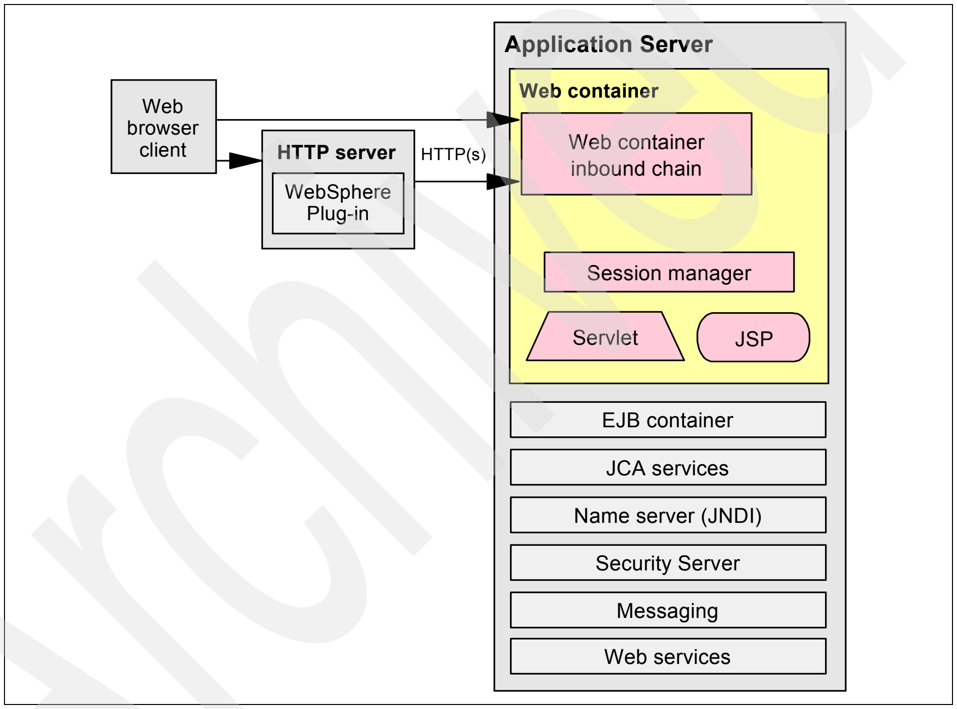


Figure 1 Web container overview

Each Web container provides:

- Web container transport chains

  The Web container inbound transport chain handles requests. It consists of a TCP inbound channel that provides the connection to the network, an HTTP inbound channel that serves HTTP 1.0 and 1.1 requests, and a Web container channel that sends requests for servlets and JSPs to the Web container for processing.

- Servlet processing

  When it handles servlets, a Web container creates a request object and a response object and then invokes the servlet service method. The Web container also invokes the servlet destroy method when appropriate and unloads the servlet, after which the JVM™ performs garbage collection.

- HTML and other static content processing

  Requests for HTML and other static content that are directed to the Web container are served by the Web container inbound chain. However, in most cases, using an external Web server and a Web server plug-in as a Web container interface is more appropriate for a production environment.

- Session management

  The Web container provides support for the javax.servlet.http.HttpSession interface as described in the API specification for the servlet.

Web applications

Servlets and JSP files are referred to as Web components. Static content files (such as HTML pages, image files, and XML files) are bundled with Web components during application assembly to create a Web module. A Web module is the single deployable and usable unit of Web resources and has a specific structure or archived format known as a *Web archive* (WAR) file. A J2EE™ Web module corresponds to a Web application as defined in the Java™ servlet specification.

The top-level directory of a Web module is the *document root* of the application. The document root is where JSP pages and static Web resources are stored. The document root has a subdirectory that is named /WEB-INF/, which contains the Web application deployment descriptor (web.xml) and the server-side classes used by the module.

Figure 2 on page 4 illustrates the directory structure of a Web application.

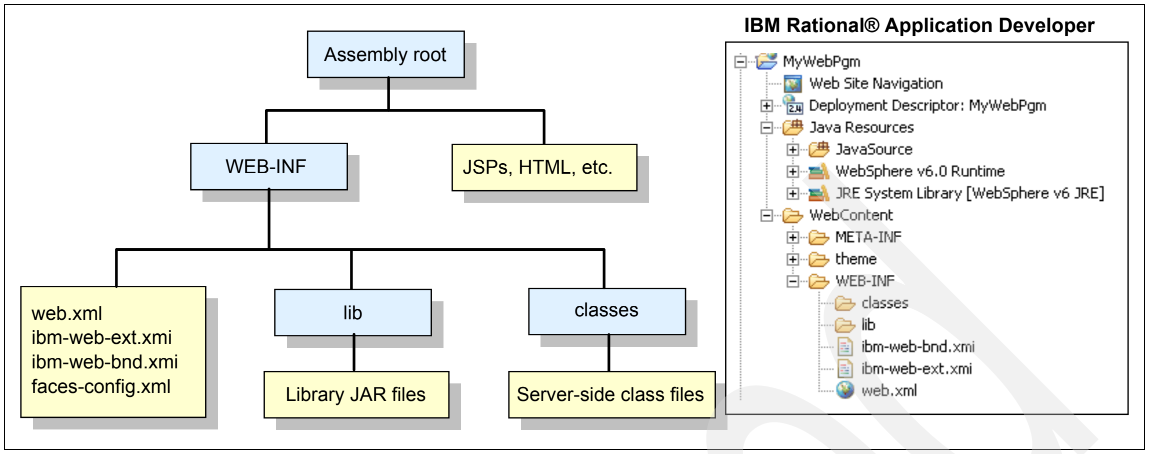


Figure 2 Web application components

Determine the Web container problem type

Many indicators of a Web container problem first appear to a user as an unexpected Web browser page or page contents, such as:

- HTTP 404 errors
- HTTP 500 errors
- Incorrect information displayed on Web pages

The unexpected information can be an error page that results from an HTTP error or a page that displays incorrect or incomplete information.

When an HTTP error occurs, the Web browser displays the error page with the error code. If a custom error page has been configured for the Web module, you must check the Web server log files to determine the HTTP error code because the true error might not be apparent.

Where to go from here: If you know the type of error, you can go directly to:

- "HTTP 404 errors" on page 5
- "HTTP 500 errors" on page 25
- "Incorrect information displayed on Web pages" on page 35

If you do not know the type of error, continue with"Collect and analyze the Web server log" on page 5.

Collect and analyze the Web server log

Collect the access log for the Web server. The Web server log file names and locations are specific to the product.

To analyze the log, search the Web server access log for 404 or 500 errors.

The following examples are from IBM HTTP Server logs:

- From the access.log:

  ```
  127.0.0.1 - - [28/Mar/2007:19:52:31 -0400] "GET url HTTP/1.1" 404 304
  127.0.0.1 - - [28/Mar/2007:20:03:48 -0400] "GET urlHTTP/1.1"500 5348
  ```

  Note the URL the Web server was trying to serve when the error occurred.
- If the Web server expected to serve the URL (versus passing the request to an application server), there is a corresponding error with information about where the Web server expected to find the file. Search the Web server error log for errors indicating that a file was not found, such as:

  ```
  - [Wed Mar 28 19:52:31 2007] [error] [client 127.0.0.1] File does
  not exist: file_location
  ```

HTTP 404 errors

HTTP 404 errors can have different underlying causes. Some examples of these causes are:

- External factors, such as a problem in the Web server
- Configuration problems, such as an incorrect Web server plug-in or virtual host configuration
- Runtime problems, such as an application or application server not started
- User or application problems, such as an incorrectly specified URL

Check system integrity

When users receive HTTP 404 error codes and you know the application at the root of the problem, the first thing to check is the integrity of the following system components:

- Web server
- Application server
- Application

If these are all working properly or you are not sure which application and server are involved, collecting more information about the symptoms is necessary.

Verify that the Web server is responding

Verify that the Web server is responding to requests. How you do this depends on your Web server product and its configuration.

If you are using IBM HTTP Server, a quick test is to access the URL for the Web server from a Web browser:

```
http://server_name
```

If you see the Welcome page, the Web server is running and responding to requests. If the Welcome page does not appear (that is, if you get a browser message such as page cannot be displayed or something similar), the problem is most likely in the Web server.

Resolve Web server problems

For general information about how to approach Web server problems, see “Approaching problems with Web servers and plug-ins” on page 49.

Verify that the application server is started

To verify the health of the hosting application server:

1. Determine the server or cluster that hosts the application.
2. Check the status of the application server.
3. Start the application server if it is not running.

If you are not sure about how to follow these steps, see “Managing servers and applications” on page 52.

Verify that the application is started

The status of an application can be seen in the administrative console. Select **Applications** → **Enterprise Applications**.

A Started status means the application is running, but it does not indicate if there were problems during the application startup.

If the application is not started, attempt to start it by selecting the application and clicking **Start**.

Collect diagnostics

If the type of error is not apparent to you or you did not obtain the URL information, collect the following diagnostic data:

- Errors displayed by the user browser, including the URL that failed
- SystemOut log data for the application server

  If your application is deployed to a cluster, you might need to collect the logs from each active server in the cluster. See “JVM logs” on page 46.
- Web server access and error logs

  If you are running an IBM HTTP Server, see “IBM HTTP Server, HTTP Server (powered by Apache) log files” on page 47.
- TRCTCPAPP trace (i5/OS)

  Because the HTTP Server (powered by Apache) is integrated into i5/OS, there is additional tracing available that can provide useful information. See “Trace Web server requests to WebSphere (i5/OS)” on page 47 for information about collecting this trace.

Analyze diagnostics

The two key determinations that you can make from the diagnostic data are:

- The URL that failed
- Where the failure occurred (Web server, plug-in, application server)

Analyze the diagnostics in the following order:

1. The browser error display

   The errors displayed by the browser can indicate the failure type and URL that failed. This might be enough information to resolve the problem.
2. SystemOut

   If the failure occurs in the Web container, SystemOut has the most informative error messages. These might be displayed by the browser also, depending on the error handling of the application. If no error messages are seen in the JVM logs, the request most likely did not reach the application server.
3. The TRCTCPAPP trace (i5/OS only)

   This trace helps you determine if the problem is caused by the plug-in.
4. Web server log

   If no other error indications are found, look at the Web server log, which can tell you the type of error (404 or 500) and the URL that caused the error.

Analyze the errors displayed by the browser

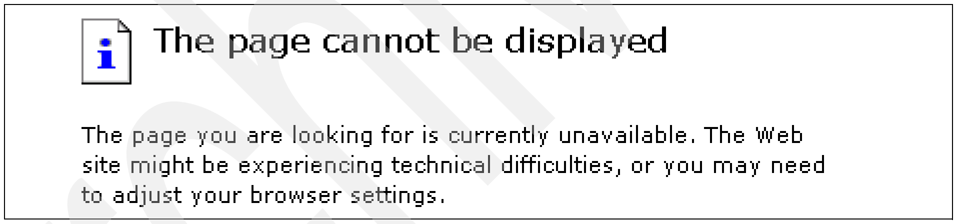
HTTP 404 errors often appear in the client browser with descriptive text and possibly with additional error information. If the following errors have been reported by users and you know the failing URL, you can determine the root cause of the problem:

- The page cannot be displayed
- JSP error or JSF error
- Failed to find resource
- File not found
- WebGroup/virtual host not defined

The type of error is not always be apparent to the user. Web applications often intercept and handle these types of errors, presenting a generic error page to the Web client. In that case, check the application server and Web server logs.

The page cannot be displayed or found

Figure 3 shows a typical example of a Web page with this error message. The display varies, depending on the browser; for example, instead of this page, you might see "The page cannot be found."


The page cannot be displayed

The page you are looking for is currently unavailable. The Web site might be experiencing technical difficulties, or you may need to adjust your browser settings.

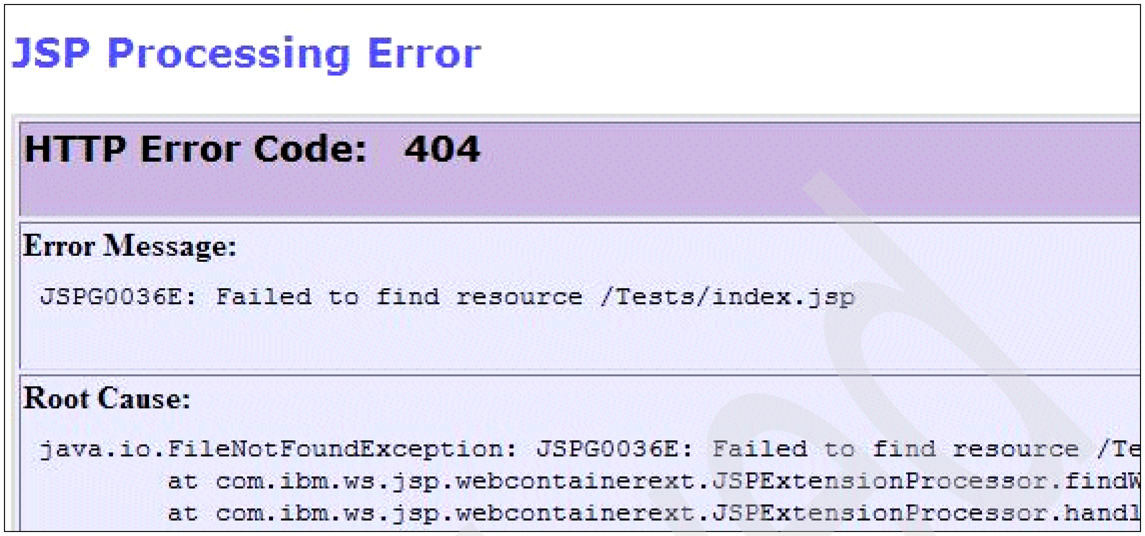
Figure 3 HTTP 404 Page cannot be displayed

Possible root causes for this error include a URL that was specified incorrectly and the unavailability of a component that is required to access the page (Web server, application server, for example). Note the URL in the browser at the time the error is displayed.

If the resource is a JSP or servlet, see "Page cannot be displayed or JSP/JSF error" on page 16.

JSP and JSF errors

JSP and JSF errors are usually accompanied by a message from the application server that indicates the problem. Figure 4 on page 9 shows a JSP error with a "JSPG0036E: Failed to find resource" message.


JSP Processing Error

HTTP Error Code: 404

Error Message:

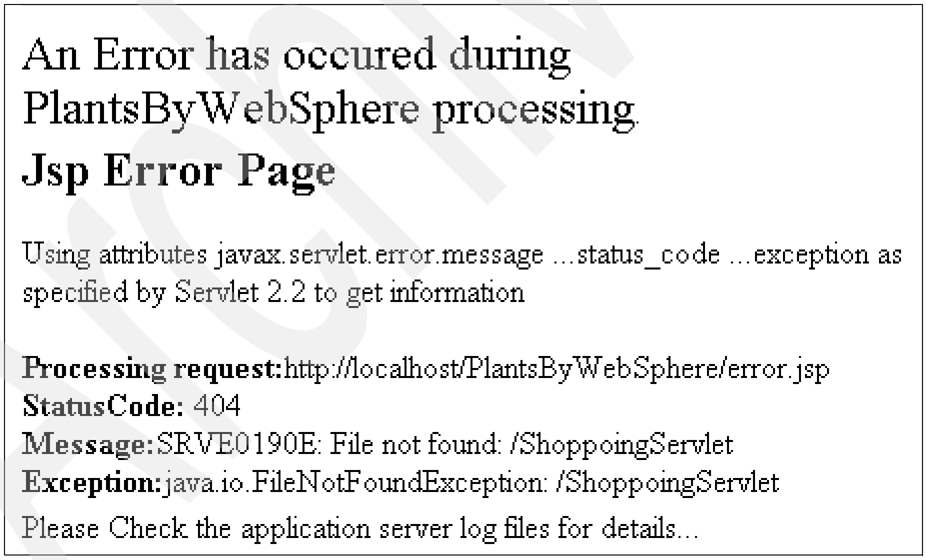
```
JSPG0036E: Failed to find resource /Tests/index.jsp
```

Root Cause:

```
java.io.FileNotFoundException: JSPG0036E: Failed to find resource /Te
        at com.ibm.ws.jsp.webcontainerext.JSPExtensionProcessor.findW
        at com.ibm.ws.jsp.webcontainerext.JSPExtensionProcessor.handl
```

Figure 4 HTTP 404 JSP error, JSPG0036E

Figure 5 shows a JSP error with an "SRVE0190E: File not found" message.


An Error has occured during PlantsByWebSphere processing.

Jsp Error Page

Using attributes javax.servlet.error.message ...status_code ...exception as specified by Servlet 2.2 to get information

Processing request:http://localhost/PlantsByWebSphere/error.jsp
StatusCode: 404
Message:SRVE0190E: File not found: /ShoppoingServlet
Exception:java.io.FileNotFoundException: /ShoppoingServlet
Please Check the application server log files for details...

Figure 5 JSP error, SRVE0190E

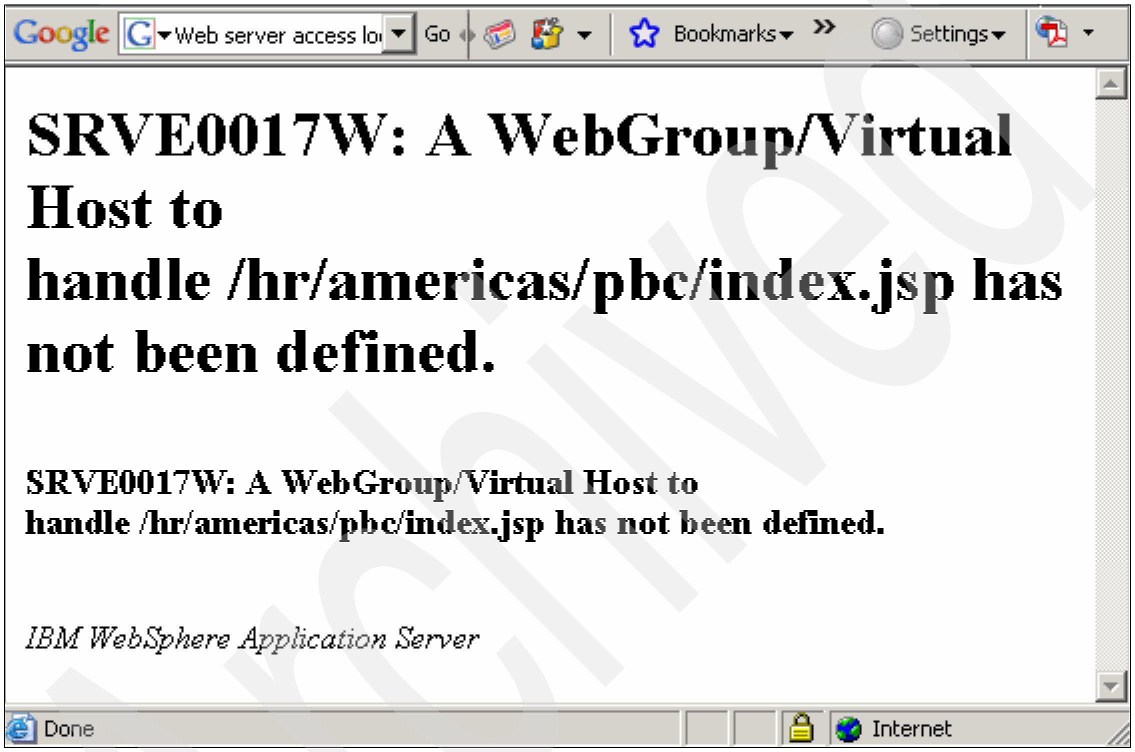
Possible root causes for this error include an incorrectly specified URL, the page is not available on the server, and a Web server plug-in configuration problem. Note the URL that caused the error. For more information, see “Page cannot be displayed or JSP/JSF error” on page 16.

WebGroup/Virtual Host has not been defined

The primary reasons for this error are virtual host configuration errors and an incorrectly specified URL. It is usually accompanied by the following messages:

- `SRVE0255E: WebGroup/Virtual Host has not been defined` (V6.1)
- `SRVE0017W: WebGroup/Virtual Host has not been defined` (V6.0)

Figure 6 is an example of what the SRVE0017W message looks like.



SRVE0017W: A WebGroup/Virtual Host to handle /hr/americas/pbc/index.jsp has not been defined.

SRVE0017W: A WebGroup/Virtual Host to handle /hr/americas/pbc/index.jsp has not been defined.

IBM WebSphere Application Server

Figure 6   WebGroup/Virtual Host failure

Note the URL that caused the error. For more information, see "WebGroup/virtual host not defined" on page 16.

Analyze SystemOut

To analyze SystemOut, search for:

- Web container messages: SRVExxxxE or SRVExxxxW messages
- JSP messages: JSPGxxxxE or JSPGxxxxW messages
- JSF messages: JSFGxxxxE or JSFGxxxxW messages
- Error messages that are related to the application startup

Check for successful application start

The sequence of messages in Example 1 shows a normal application and Web module startup sequence.

Example 1 Normal application startup messages

```
ApplicationMg A   WSVR0200I: Starting application: [application_name]
WebContainer  A   SRVE0161I: IBM WebSphere Application Server - Web
Container.
Copyright IBM Corp. 1998-2004
WebContainer  A   SRVE0162I: Servlet Specification Level: 2.4
WebContainer  A   SRVE0163I: Supported JSP Specification Level: 2.0
WebGroup      A   SRVE0169I: Loading Web Module: [web_module_name]
ApplicationMg A   WSVR0221I: Application started: [application_name]
```

If you received error or warning messages during application startup, you must address the problem with the application.

Web group/virtual host errors

If you see one of the following messages, you might have a problem with the virtual host configuration:

- `SRVE0255E: A WebGroup/Virtual Host to handle` *`url`* `has not been defined.` (V6.1)
- `SRVE0017W: A WebGroup/Virtual Host to handle` *`url`* `has not been defined.` (V6.0)

For more information, see "WebGroup/virtual host not defined" on page 16.

Other error messages

Error messages with JSPG, JSFG, or SRVE prefixes have information about the error, including the URL that caused the problem. Note the URL and see "Page cannot be displayed or JSP/JSF error" on page 16.

Where to go from here: If you found an error message that is not in the list, see "The next step" on page 48. If you did not find any errors, see "Analyze the Web server logs" on page 12.

Analyze the Web server logs

You can search for error indications in the Web server logs and analyze them.

Search for 404 status codes in the access log

A log entry in the NCSA format looks like this:

```
Remote_host- - date_time "request" status_code bytes
```

For 404 errors:

1. Search for “404” in *status_code*.
2. Note the *remote_host* address, the *date_time* stamp, and the *request* URL.

Search for errors in the error log

An error entry in the error log resembles this example:

```
[date_time] [error] [client Remote_host] error_message
```

For errors in this entry:

1. Search for “error.”
2. Correlate the access log entry that you noted with the error log entry by comparing the *date_time* and *remote_host* entries.
3. Note the *error_message* text.

Evaluate the access log

HTTP errors that are displayed by the Web browser are logged in the access log. These errors can originate in the Web server or application server.

The primary value of analyzing the access log entry is to determine the failing URL.

Static resources (for example, HTML pages and images) are normally served by the Web server. However, requests for static resources are sent to the application server when the fileServingEnabled property is set to **true** in the Web module extended deployment descriptor, which is ibm-web-ext.xmi. When an error occurs during the process of serving static resources from the application server, the only indication of the failure is in the access log.

If you see a 404 error in the access code, but no corresponding error in the Web server error log or SystemOut, check the fileServingEnabled property. If it is set to **true**, verify that the resources exist in the proper path in the application server.

Evaluate the error log

The presence of a corresponding message in the error log indicates that the Web server attempted to handle the request and failed (as opposed the WebSphere Application Server). The message contains more information about the error, including the file that the Web server was trying to serve. This indicates that the problem is in the Web server.

This section provides some examples of error messages.

> **Note:** If there is a 404 error in the access log, file serving is not enabled, and no error is seen in the error log, see "Page cannot be displayed or JSP/JSF error" on page 16 for more information.

Example 1

If errors similar to those that are shown here appear in the Web server logs, they indicate that the URL /sdfsdf.dsf was not found.

- access.log:

  ```
  127.0.0.1 - - [28/Mar/2007:19:52:31 -0400] "GET /sdfsdf.dsf HTTP/1.1"
  404 304
  ```

- error.log

  ```
  - [Wed Mar 28 19:52:31 2007] [error] [client 127.0.0.1] File does
  not exist: C:/IBM/IBM HTTP Server/htdocs/en_US/sdfsdf.dsf
  ```

The message in the error log tells you what file the URL translates to.

The root of this problem is that the file does not exist in the Web server. Verify that the URL is correct and make sure that the file is in the proper location on the Web server.

Example 2

If you see an error similar to the one that is shown here in the access log, but no corresponding error in the error log, it indicates that the Web server did not consider this a request that it should handle:

```
127.0.0.1 - - [28/Mar/2007:19:54:49 -0400] "GET /HitCount2.jsp
HTTP/1.1" 404 2876
```

The most likely cause is a problem with the Web server plug-in or at the application server.

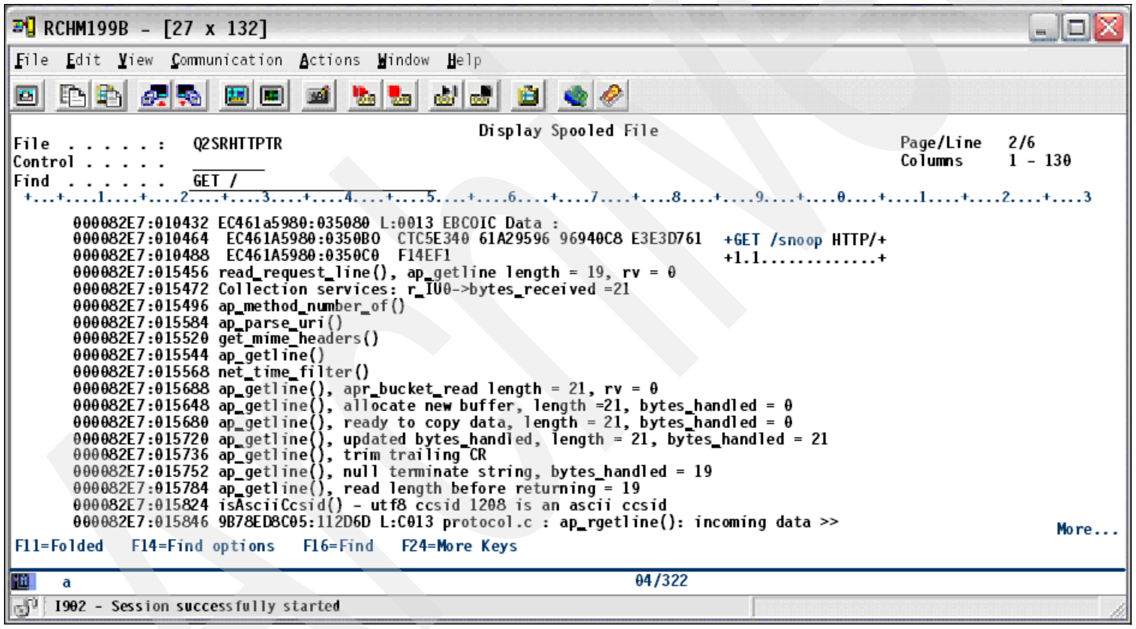
Verify that the plug-in configuration is correct (see "Verify that the Web server plug-in is working correctly" on page 21) and that the URL is correct (see "Verify that the URL is correct" on page 17).

Where to go from here: If you still feel sure that you have a Web container problem, but your symptoms are not listed here, go to “The next step” on page 48 to find information about searching online support and preparing to contact IBM if necessary.

Analyze the TRCTCPAPP trace

If your problem occurred in i5/OS and you collected this trace, analyze the trace contents:

1. Display the spool file.
2. On the Find line at the top, type **GET /** and press **F16** to locate the first request in the trace (Figure 7).



```
                                        Display Spooled File
File  . . . . . :   Q2SRHTTPTR                                                    Page/Line   2/6
Control . . . . .   ________                                                      Columns     1 - 130
Find  . . . . . .   GET /
 +...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8....+....9....+....0....+....1....+....2....+....3

      000082E7:010432 EC461a5980:035080 L:0013 EBCOIC Data :
      000082E7:010464  EC461A5980:0350B0  CTC5E340 61A29596 96940C8 E3E3D761  +GET /snoop HTTP/+
      000082E7:010488  EC461A5980:0350C0  F14EF1                              +1.1.............+
      000082E7:015456 read_request_line(), ap_getline length = 19, rv = 0
      000082E7:015472 Collection services: r_IU0->bytes_received =21
      000082E7:015496 ap_method_number_of()
      000082E7:015584 ap_parse_uri()
      000082E7:015520 get_mime_headers()
      000082E7:015544 ap_getline()
      000082E7:015568 net_time_filter()
      000082E7:015688 ap_getline(), apr_bucket_read length = 21, rv = 0
      000082E7:015648 ap_getline(), allocate new buffer, length =21, bytes_handled = 0
      000082E7:015680 ap_getline(), ready to copy data, length = 21, bytes_handled = 0
      000082E7:015720 ap_getline(), updated bytes_handled, length = 21, bytes_handled = 21
      000082E7:015736 ap_getline(), trim trailing CR
      000082E7:015752 ap_getline(), null terminate string, bytes_handled = 19
      000082E7:015784 ap_getline(), read length before returning = 19
      000082E7:015824 isAsciiCcsid() - utf8 ccsid 1208 is an ascii ccsid
      000082E7:015846 9B78ED8C05:112D6D L:C013 protocol.c : ap_rgetline(): incoming data >>
                                                                                                                  More...
F11=Folded   F14=Find options   F16=Find   F24=More Keys

      a                                                           04/322
I902 - Session successfully started
```

Figure 7 Trace data

The column on the right with request information is called the “eye-catcher.” The data is surrounded by “*” to help it stand out; for example:

```
*GET /snoop HTTP/*
*1.1.............*
```

3. Scroll down to see additional HTTP header information in the request in the eye-catcher.

4. The following entry type indicates that a request was passed to WebSphere:

   ```
   mod_was_ap20_http: as_translate_name: WebSphere will handle: /snoop
   ```

   This line tells you that the HTTP server matched the incoming request to the rules in the Web server plug-in configuration file and sent the request to WebSphere for processing.

5. Continue to scroll down until you see the outgoing HTTP response to the request.

   A successful response looks like that in Example 2.

Example 2 Successful response from WebSphere

```
*HTTP/1.1 200 OK.*
*.Date: Thu, 05 A*
*pr 2007 14:54:00*
* GMT..Server: We*
*bSphere Applicat*
*ion Server/6.1..*
```

If the request is unsuccessful, the Server header on the outgoing HTTP response tells you whether HTTP or WebSphere is throwing the error.

Example 3 is a sample of a 404 response sent by WebSphere that indicates that the plug-in sent the request to WebSphere but the application could not be found on the WebSphere side:

Example 3 HTTP 404 response from WebSphere

```
*HTTP/1.1 404 Not*
* Found..Date: Th*
*u, 05 Apr 2007 1*
*5:16:49 GMT..Ser*
*ver: WebSphere A*
*pplication Serve*
*r/6.1..........*
```

Example 4 shows an outgoing HTTP response that was sent from the HTTP server.

Example 4 HTTP 404 response from the HTTP server to the user

```
*HTTP/1.1 404 Not*
 * Found..Date: Th*
 *u, 05 Apr 2007 1*
 *5:35:56 GMT..Ser*
 *ver: Apache.....*
```

If the plug-in correctly sent the request to WebSphere, but an error occurred at the server, see “Page cannot be displayed or JSP/JSF error” for more information. If the plug-in did not send the request to WebSphere, but it should have, see “WebGroup/virtual host not defined” for more information.

Root causes for HTTP 404 errors

Based on symptoms that users report, you can usually narrow the problem down to one of two types:

- Page cannot be displayed or JSP/JSF error
- WebGroup/virtual host not defined

Determining the root cause of each type involves a series of checks that ensure that the components of the system (Web server, plug-in, etc.) are working properly and that there are no configuration problems. Some of these system checks are performed for more than one problem type.

Page cannot be displayed or JSP/JSF error

When you have a “Page cannot be displayed” error and the resource is a JSP page, JSF page, or servlet, or the browser is displaying a JSP error, take the following actions:

- Verify that the Web server plug-in is working correctly (see “Verify that the Web server plug-in is working correctly” on page 21).
- Verify that the URL that is causing the error is specified correctly (see “Verify that the URL is correct” on page 17).
- Verify that the Web server is responding (see “Verify that the Web server is responding” on page 6).
- Verify that the application is running (see “Verify that the application is started” on page 6).

If these actions do not resolve your problem, see “The next step” on page 48.

WebGroup/virtual host not defined

For a “WebGroup/virtual host not defined” error, take the following actions:

- Verify that the virtual host configuration is correct (see “Verify that the virtual host configuration is correct” on page 23).
- Verify that the URL causing the error is specified correctly (see “Verify that the URL is correct” on page 17).
- Verify that the application is running (see “Verify that the application is started” on page 6).

If these actions do not resolve your problem, see “The next step” on page 48.

Verify that the URL is correct

In a new installation or new application, it is possible that the URL of the application is not being specified correctly. To determine the URL of the installed application, use the administrative console to view the configuration of a number of items.

The format of the URL is as follows:

```
http://host:port/context_root/servlet_url_pattern
```

To find the URL for a servlet or JSP:

1. Find the URL pattern for the servlet.
2. Find the context root of the Web module that contains the servlet.
3. Find the virtual host where the Web module is installed.
4. Find the aliases for the virtual host.
5. Compare the failing URL with the correct URL for the servlet.

The examples in this section use the ShoppingServlet of the PlantsByWebSphere sample application.

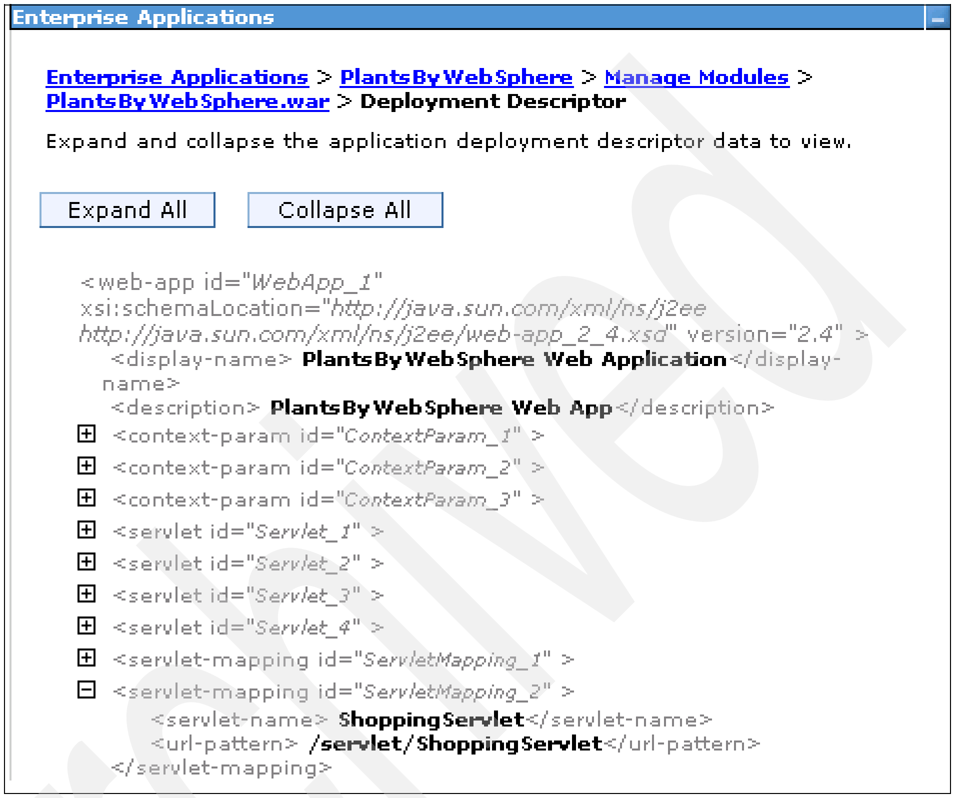
Find the URL pattern (*servlet_url_pattern)*

Find the URL pattern for the servlet in the deployment descriptor of the Web module as follows:

1. In the administrative console, select **Applications** →**Enterprise Applications**.
2. Click the application name to open the Configuration page for the application.
3. Select **Manage Modules** to list the Web and EJB™ modules in the application. If there is more than one Web module, view the deployment descriptor for each until you find the servlet.
4. Click ***Web_module_name*** **Web application** to see the general properties.
5. Click **View Deployment Descriptor**.

   This opens the Web module properties window. Look for the servlet in a <servlet-mapping> entry and note the **url-pattern** value.
6. If the Web module has security configured, check the *<security-constraint>* and *<security-role>* deployment descriptor tags for the role that is needed for access to the selected Web resource.

Figure 8 shows the portion of the PlantsByWebSphere Web module deployment descriptor that includes the ShoppingServlet mapping. From this, you can see that the URL pattern is **/servlet/ShoppingServlet**.


Enterprise Applications
Enterprise Applications > PlantsByWebSphere > Manage Modules > PlantsByWebSphere.war > Deployment Descriptor
Expand and collapse the application deployment descriptor data to view.
Expand All
Collapse All

```
<web-app id="WebApp_1"
xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd" version="2.4" >
   <display-name> PlantsByWebSphere Web Application</display-
  name>
   <description> PlantsByWebSphere Web App</description>
⊞  <context-param id="ContextParam_1" >
⊞  <context-param id="ContextParam_2" >
⊞  <context-param id="ContextParam_3" >
⊞  <servlet id="Servlet_1" >
⊞  <servlet id="Servlet_2" >
⊞  <servlet id="Servlet_3" >
⊞  <servlet id="Servlet_4" >
⊞  <servlet-mapping id="ServletMapping_1" >
⊟  <servlet-mapping id="ServletMapping_2" >
       <servlet-name> ShoppingServlet</servlet-name>
       <url-pattern> /servlet/ShoppingServlet</url-pattern>
   </servlet-mapping>
```

Figure 8 PlantsByWebSphere Web module deployment descriptor

The URL for the example now looks like:

`http://host:port/context_root/servlet/ShoppingServlet`

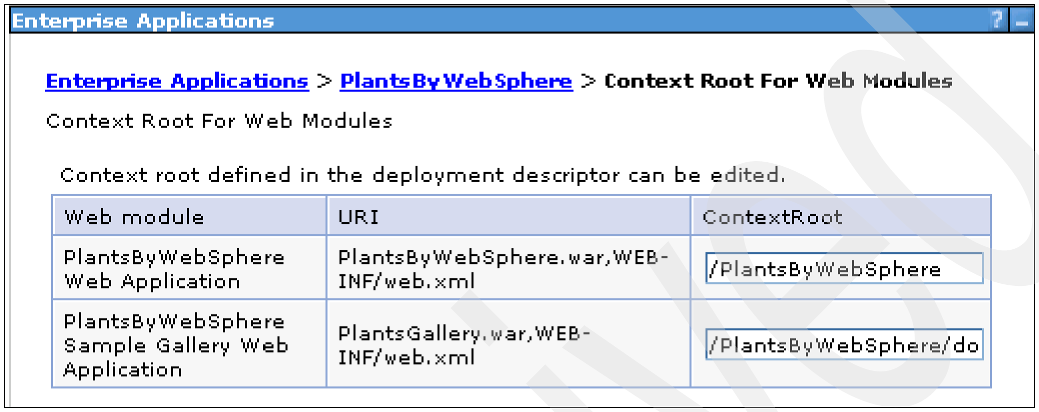
Find the context root (*context_root*)

To find the context root for the URL:

1. Select **Applications →Enterprise Applications**.
2. Click the application name.
3. Click **Context Root for Web Modules**.
4. Note the context root for the appropriate Web module.

In Figure 9, you can see that:

- There are two Web modules in the PlantsByWebSphere enterprise application. From the previous step, you know the Web module that contains the ShoppingServlet is the PlantsByWebSphere Web module.
- The context root for the PlantsByWebSphere Web module is /PlantsByWebSphere.


Enterprise Applications

Enterprise Applications > PlantsByWebSphere > Context Root For Web Modules

Context Root For Web Modules

Context root defined in the deployment descriptor can be edited.

Web module	URI	ContextRoot
PlantsByWebSphere Web Application	PlantsByWebSphere.war,WEB-INF/web.xml	/PlantsByWebSphere
PlantsByWebSphere Sample Gallery Web Application	PlantsGallery.war,WEB-INF/web.xml	/PlantsByWebSphere/do

Figure 9 Context root for the Web modules in DefaultApplication

When you put the context root together with the resource name, you obtain the following result:

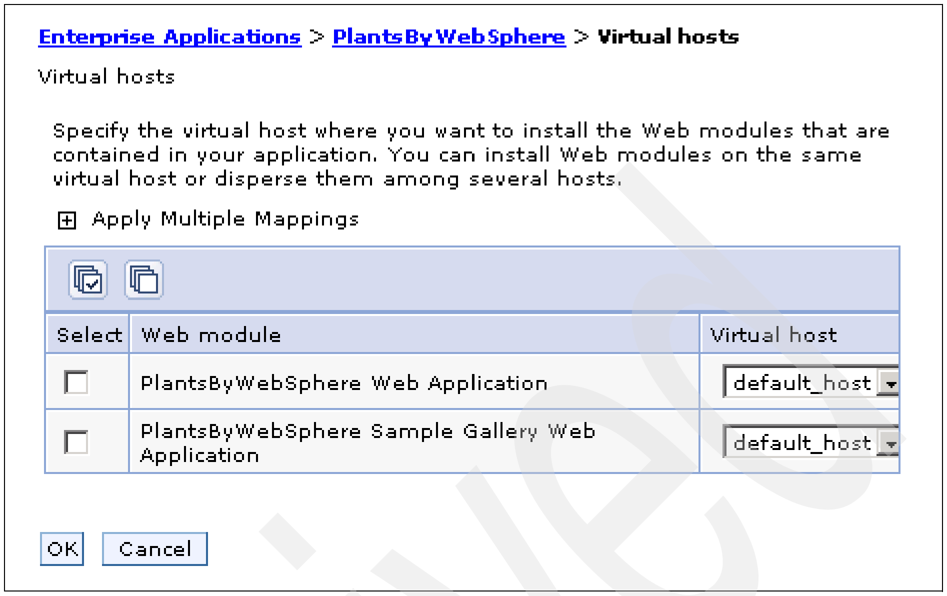
```
http://host:port/PlantsByWebSphere/servlet/ShoppingServlet
```

Find the virtual host (*host:port*)

To find the virtual host where the Web module is installed:

1. Select **Applications** →**Enterprise Applications**.
2. Click the application name.
3. Click **Virtual hosts** under Web Module Properties to see the Web modules in the application and the virtual hosts in which they have been installed.
4. Note the virtual host for the Web module.

Figure 10 on page 20 shows the Web modules in the PlantsByWebSphere application and the virtual hosts in which they have been installed. Note that the PlantsByWebSphere Web module has been installed in *default_host*.


Enterprise Applications > PlantsByWebSphere > Virtual hosts

Virtual hosts

Specify the virtual host where you want to install the Web modules that are contained in your application. You can install Web modules on the same virtual host or disperse them among several hosts.

⊞ Apply Multiple Mappings

Select	Web module	Virtual host
☐	PlantsByWebSphere Web Application	default_host
☐	PlantsByWebSphere Sample Gallery Web Application	default_host

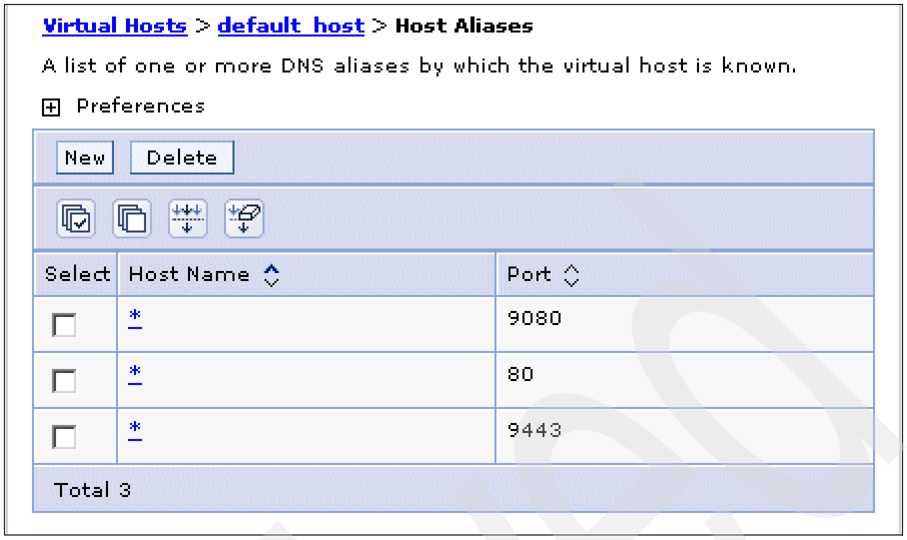
Figure 10 List of virtual hosts

Find the aliases for the virtual host

To find the host aliases for the virtual host:

1. From the console navigation tree, select **Environment** →**Virtual Hosts**.
2. Click the virtual host.
3. Select **Host Aliases** under Additional Properties.
4. Note each alias. An alias is composed of a host name and port number. Each alias represents a valid *host:port* combination for the Web module.

Figure 11 on page 21 shows the list of aliases by which *default_host* is known. The host aliases are *:80, *:9080 and *:9443. The asterisk means that any host name can be used.


Virtual Hosts > default_host > Host Aliases

A list of one or more DNS aliases by which the virtual host is known.

⊞ Preferences

Select	Host Name	Port
☐	*	9080
☐	*	80
☐	*	9443
Total 3		

Figure 11 Default_host virtual host aliases

Compare the result with the failing URL

To compare the failing URL with the correct URL for the servlet:

1. Combine the virtual host alias, context root, and URL pattern to form the correct URL request for the servlet.
2. Compare the correct URL for the servlet with the URL that was reported in the problem.
3. If they are the same, verify that the resource file exists in the deployed application. If they are not, correct the application that is calling the resource.

In the PlantsByWebSphere ShoppingServlet example, requests for the servlet with any of the following URLs map to the default_host virtual host:

- `http://`*host*`:80/PlantsByWebSphere/servlet/ShoppingServlet`
- `http://`*host*`:9080/PlantsByWebSphere/servlet/ShoppingServlet`
- `https://`*host*`:9443/PlantsByWebSphere/servlet/ShoppingServlet`

Verify that the Web server plug-in is working correctly

If you have recently updated or installed the application, ensure that the Web server plug-in was regenerated and propagated to the Web server. Also, ensure that the Web server is using the new plug-in configuration file.

Check other applications on the server

Attempt to access other applications running on the same application server through the Web server. If you can access another application, this verifies that the plug-in and Web server are working with the application server, but not necessarily that the plug-in configuration is correct.

Use Default Application for access

If Default Application is installed, you can try accessing the snoop or hitcount servlets through the Web server as a quick test of the plug-in configuration.

The URLs to access these servlets are:

- http://*Web_server_host*/snoop
- http://*Web_server_host*/hitcount

Access the application directly, bypassing the plug-in

If you have regenerated the plug-in and are sure it is in use, but you still have a problem, you can bypass the Web server and access the application directly from the application server. This is not the recommended method for serving a production Web site. However, it provides a good diagnostic tool when it is not clear whether a problem is in the Web server, WebSphere Application Server, or the Web server plug-in.

To bypass the Web server plug-in and access the failing application directly through the application server Web container:

1. Find the port for the Web container in the WebSphere administrative console:
   a. Select **Servers** → **Application servers**.
   b. Click the server name.
   c. Expand **Ports** in the Communications section.
   d. Note 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://*host*:9080/myAppContext/myJSP.jsp.

If you can access the application through the application server but not the Web server, you are most likely experiencing a problem with the Web server plug-in.

Resolve Web server plug-in problems

These actions can help you resolve Web server plug-in problems:

1. Review the virtual host mapping for the Web modules to make sure that the virtual host mapped to the module has the alias definitions required to reach the server. (see “Verify that the virtual host configuration is correct” on page 23).

2. Review the Web module mapping to ensure that the Web module is mapped to the correct the Web server.

   To see the mapping for the modules using the administrative console, select **Applications** →**Enterprise Applications**. Click the application name to open the configuration page and click **Manage Modules**.

3. Correct any problems.
4. Regenerate the plug-in and propagate it to the Web server.
5. Restart the Web server or wait for the new configuration to be activated.

If this doesn't resolve the problem, review *WebSphere Application Server V6: Web Server Plug-in Problem Determination:*

http://www.redbooks.ibm.com/redpapers/pdfs/redp4045.pdf

Verify that the virtual host configuration is correct

When you install an application, you associate a virtual host with each Web module in the application. The virtual host has a set of defined host aliases, each consisting of a host name and port number. Requests that match a host alias for a virtual host are processed by servlets/JSPs in the Web module.

Web server plug-in processing

When the Web server receives a request, the plug-in checks the URI of the request against those defined in the host aliases to determine which, if any, Web module can handle the request.

If the Web server plug-in receives a request that does not match one of the virtual hosts, then the user receives an HTTP error.

Determine the virtual host for the Web module

To determine the virtual host associated with the Web module, in the administrative console, select **Enterprise Applications** → ***application_name*** → **Virtual Hosts.**

The list that results shows the virtual host mapping for each Web module in the application.

In i5/OS, you can also use the IBM Web Administration for i5/OS console to view Web module virtual host mappings by selecting **Applications** → **Manage Installed Applications**. Select the application and click **Properties**.

View and modify the virtual host definition

To view and modify a virtual host definition using the WebSphere administrative console:

1. Select **Environment** → **Virtual Hosts**.
2. Click the virtual host name.
3. Click **Host Aliases**.
4. Click the host alias name that you want to modify.
5. Edit the host name or port as needed.
6. Restart the application server.
7. Regenerate the plug-in and propagate it to the Web server.

You can also use the IBM Web Administration for i5/OS console to view and manage virtual host definitions. In the console, select **Resource Configuration** → **Manage Virtual Hosts**.

Host alias definitions

Follow these guidelines when you are verifying the host alias definitions:

- Make sure that the host aliases that are defined for a virtual host include the host name and port number of the WebSphere Application Server and the host names and port numbers that the Web server plug-in is expecting to receive from the browser.
- Mapping HTTP requests to host aliases is case sensitive and the match must be alphabetically exact. For example, the request http://www.myhost.com/myservlet does not map to any of the following:
  - http://myhost/myservlet
  - http://www.myhost.com/MyServlet
  - http://www.myhost.com:9876/myservlet
- A * wild card can be used for the host name, the port, or both. When it is used for both, any request matches this rule.

What to look for

Check the virtual host definitions to make sure that the virtual host that is associated with the Web module has a host alias defined that contains the host name and the port that matches the host name and port in the URL that is causing the failure.

If `localhost` is used as an alias entry, check the etc/hosts file to ensure that all host names (aliases) that are associated with the loop back address (127.0.0.1) are part of the same virtual host grouping (for example, *default_host*). Also, verify that no duplicate host aliases have been defined in multiple virtual hosts.

For example, in Example 5, the host alias `test:80` is duplicated in both virtual hosts because a URI that contains `test:80` would match aliases in both virtual hosts (`*:80` and `test:80`).

Example 5 Virtual host definitions

```
<VirtualHostGroup Name="default_host">
       <VirtualHost Name="*:80"/>
       <VirtualHost Name="*:9080"/>
</VirtualHostGroup>

<VirtualHostGroup Name="test_host">
       <VirtualHost Name="test:80"/>
       <VirtualHost Name="*:9081"/>
</VirtualHostGroup>
```

HTTP 500 errors

HTTP 500 errors indicate that an internal server problem has occurred during the process of serving the requested page. Examples of the types of problems that can cause this in WebSphere Application Server include:

- JSP processor errors
- Application code (JSP, JSF, servlet) errors
- Session errors

In this section, we show you how to identify the root cause of an HTTP 500 error and how to resolve it.

Collect diagnostics

Begin by collecting the following diagnostics:

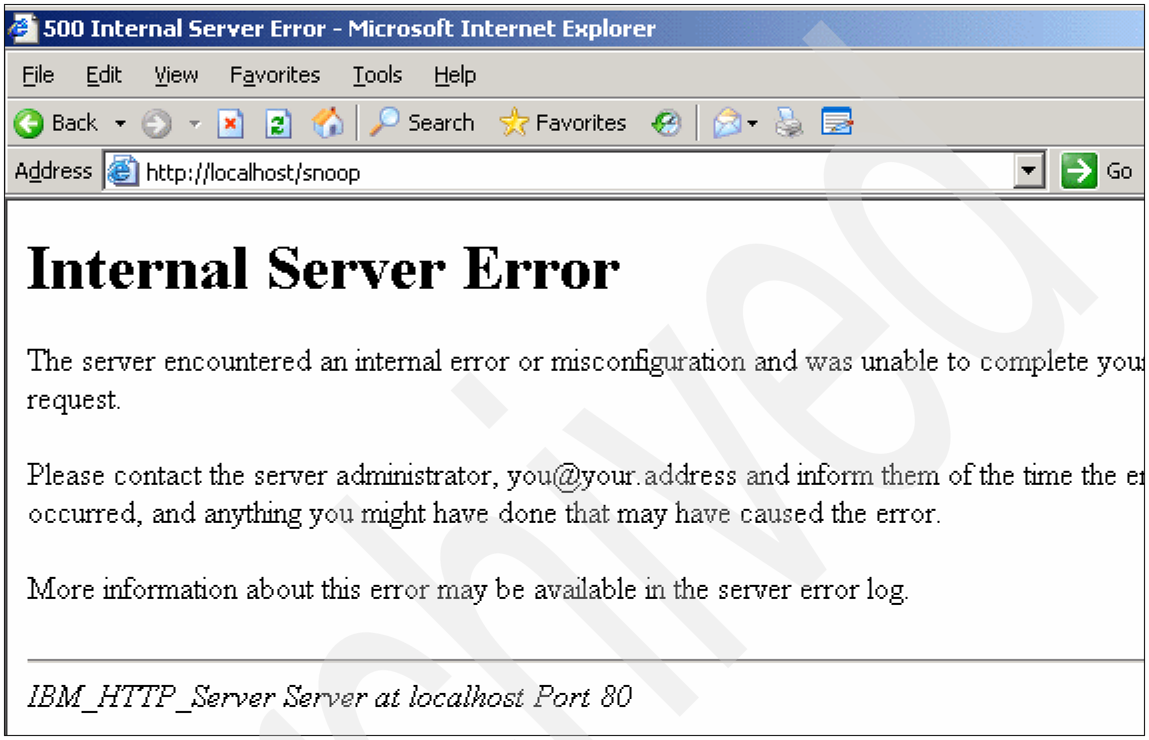
- Errors displayed by the browser, including the failing URL
- SystemOut log data for the application server (see “JVM logs” on page 46 for more information)

Analyze errors displayed by the browser

HTTP 500 errors often appear in the client browser with descriptive text and possibly with additional error information. The primary message is an HTTP 500 and sometimes there is additional information. However, the type of error is not always apparent to the user. Web applications often intercept and handle these types of errors, sending a generic error page to the Web client.

Internal server errors with no accompanying messages

When users receive HTTP 500 error codes and no accompanying descriptive messages (Figure 12), it is possible that no application servers are available to serve the file.


500 Internal Server Error - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost/snoop
Go

Internal Server Error

The server encountered an internal error or misconfiguration and was unable to complete you request.

Please contact the server administrator, you@your.address and inform them of the time the e occurred, and anything you might have done that may have caused the error.

More information about this error may be available in the server error log.

IBM_HTTP_Server Server at localhost Port 80

Figure 12 HTTP 500 Internal Server Error

To verify that an application server is available to serve the file:

1. Determine the server or cluster that hosts the application.
2. Check the status of the application server.
3. Start the application server if it is not running.

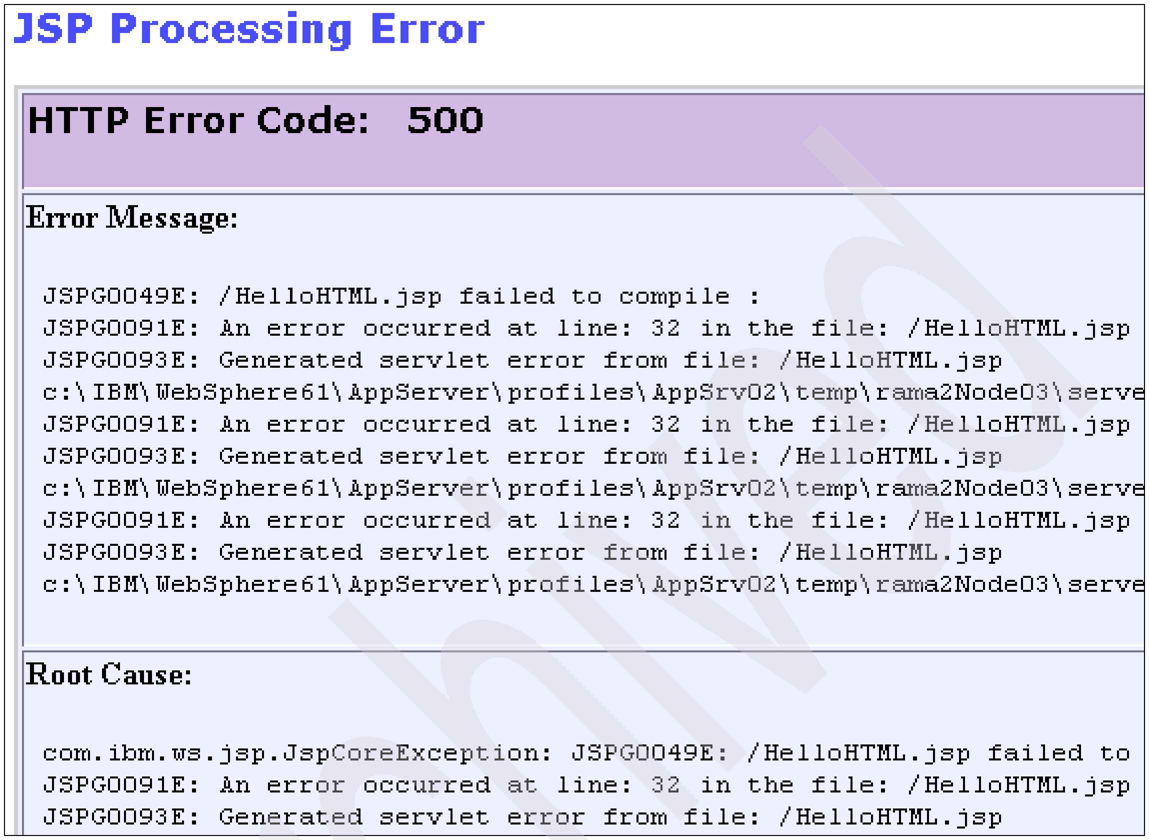
For more information, see "Managing servers and applications" on page 52.

JSP processing errors

JSP processing errors can be caused by application coding errors or a runtime error in the JSP processor in the Web container. Examples of messages for these errors include:

- JSPG0076E: Missing required attribute page...
- JSPG0049E: *jsp_name* failed to compile

Figure 13 shows how this might look in a Web browser.



JSP Processing Error

HTTP Error Code: 500

Error Message:

```
JSPG0049E: /HelloHTML.jsp failed to compile :
JSPG0091E: An error occurred at line: 32 in the file: /HelloHTML.jsp
JSPG0093E: Generated servlet error from file: /HelloHTML.jsp
c:\IBM\WebSphere61\AppServer\profiles\AppSrv02\temp\rama2Node03\serve
JSPG0091E: An error occurred at line: 32 in the file: /HelloHTML.jsp
JSPG0093E: Generated servlet error from file: /HelloHTML.jsp
c:\IBM\WebSphere61\AppServer\profiles\AppSrv02\temp\rama2Node03\serve
JSPG0091E: An error occurred at line: 32 in the file: /HelloHTML.jsp
JSPG0093E: Generated servlet error from file: /HelloHTML.jsp
c:\IBM\WebSphere61\AppServer\profiles\AppSrv02\temp\rama2Node03\serve
```

Root Cause:

```
com.ibm.ws.jsp.JspCoreException: JSPG0049E: /HelloHTML.jsp failed to
JSPG0091E: An error occurred at line: 32 in the file: /HelloHTML.jsp
JSPG0093E: Generated servlet error from file: /HelloHTML.jsp
```

Figure 13 JSP Processing Error message

Note: If you have JSPGxxxxE or JSFGxxxxE messages, see “Code error” on page 31 for more information.

IllegalStateException errors

IllegalStateException errors generally show up in a Java stack trace in the SystemOut log and can be caused by several things; the most common are invalid session objects and response generation problems.

For example, you might see “SRVE0068ESRVE0068E: could not invoke the *method_name* method on servlet *servlet_name.* Exception thrown: java.lang.IllegalStateException” with one of the following suffixes:

- Session object internals: *session_info*
- Response already committed

Analyze SystemOut

To analyze SystemOut, search for the following messages related to the error:

- SRVE0068E: could not invoke the *method_name* method on servlet *servlet_name.* Exception thrown: java.lang.IllegalStateException. If you have an illegal state exception error, the exception should give you an indication of the illegal state causing the problem.
- SRVE0239I and SRVE0240I, indicating that the JSP processor started normally.
- JSPGxxxxE messages, indicating a JSP error.
- JSFGxxxxE messages, indicating a JSF error.

Note the error text and the text that follows each message.

If you do not see any error messages or you see error messages other than those with the prefixes listed here, see "The next step" on page 48 for more information.

JSP processor error

Browse the SystemOut log of the server that hosts the JSP files to determine if the JSP processor has started successfully. Failure to start normally indicates a JSP processor exception. The messages in Example 6 indicate that the JSP processor has started normally.

Example 6 JSP processor messages in SystemOut.log file

```
WebContainer  A   SRVE0239I: Extension Factory [class
com.ibm.ws.jsp.webcontainerext.JSPExtensionFactory] was registered
successfully.
WebContainer  A   SRVE0240I: Extension Factory [class
com.ibm.ws.jsp.webcontainerext.JSPExtensionFactory] has been associated with
patterns [*.jsp *.jspx *.jsw *.jsv ].
```

If you have a JSP processor error, collect MustGather documentation and see "The next step" on page 48 for more information.

Code errors

If the JSP processor starts normally, the problem might be with the application code itself. For example, the JSP might have invalid JSP syntax that cannot be processed by the JSP processor.

Example 7 on page 29 shows a message that indicates a problem with JSP directive syntax.

Example 7 JSP directive syntax error message

```
Message: /test.jsp(2,1)JSPG0076E: Missing required attribute page for jsp
element jsp:include
```

Example 8 shows a message that indicates invalid Java syntax.

Example 8 Invalid Java syntax error message

```
com.ibm.ws.jsp.JspCoreException: JSPG0049E: /test.jsp failed to compile :
JSPG0091E: An error occurred at line: 16 in the file: /test.jsp
JSPG0093E: Generated servlet error from file: /test.jsp
```

For more information about a problem in the code, see “Code error” on page 31.

IllegalStateException: Invalid session object

When the application tries to use an invalidated session object, the IllegalStateException occurs (Example 9).

Example 9 IllegalStateException in invalid session object

```
[7/7/05 16:41:30:627 ART] 00000028 ServletWrappe E   SRVE0068E: Could not
invoke the service() method on servlet TestServlet. Exception thrown :
java.lang.IllegalStateException:
Session Object Internals:
id : pi55X7syi-ExTjyyhFn5Cu7
hashCode : 1449590567
create time : Thu Jul 07 16:24:54 ART 2005
last access : Thu Jul 07 16:41:30 ART 2005
max inactive interval : 1800
user name : anonymous
valid session : false
new session : true
overflowed : false
non-serializable app specific session data : {}
serializable app specific session data : {}
   at
com.ibm.ws.webcontainer.httpsession.SessionData.getValueGuts(SessionData.java(C
ompiled Code))
   at
com.ibm.ws.webcontainer.httpsession.SessionData.getValue(SessionData.java(Inlin
ed Compiled Code))
   at
com.ibm.ws.webcontainer.httpsession.SessionData.getAttribute(SessionData.java(I
nlined Compiled Code))
   at
com.ibm.ws.webcontainer.httpsession.HttpSessionFacade.getAttribute(HttpSessionF
acade.java(Compiled Code))
```

For errors with session objects, see “Invalid session object” on page 31 for more information.

IllegalStateException - Response generation problem

Examples of errors that are thrown by the HttpServletResponse interface during the response generation problem are:

- `java.lang.IllegalStateException: Response already committed`
- `java.lang.IllegalStateException: Header already sent`
- `java.lang.IllegalStateException: Cannot forward as Output Stream or Writer has already been obtained`

Example 10 shows the error messages generated for a “Response already committed” error.

Example 10 IllegalStateException in response generation

```
[7/8/05 20:36:25:694 ART] 0000004f ServletWrappe E   SRVE0068E: Could not
invoke the service() method on servlet TestServlet. Exception thrown :
java.lang.IllegalStateException
   at
com.ibm.ws.webcontainer.webapp.WebAppDispatcherContext.sendRedirect(WebAppDispa
tcherContext.java:486)
   at
com.ibm.ws.webcontainer.srt.SRTServletResponse.sendRedirect(SRTServletResponse.
java:810)
   at web.TestServlet.doGet(TestServlet.java:56)
   at javax.servlet.http.HttpServlet.service(HttpServlet.java:743)
   at javax.servlet.http.HttpServlet.service(HttpServlet.java:856)

[7/8/05 20:36:25:764 ART] 0000004f LocalTranCoor E   WLTC0017E: Resources
rolled back due to setRollbackOnly() being called.
[7/8/05 20:36:25:774 ART] 0000004f WebApp        E   SRVE0026E: [Servlet
Error]-[TestServlet]: java.lang.IllegalStateException
   at
com.ibm.ws.webcontainer.webapp.WebAppDispatcherContext.sendRedirect(WebAppDispa
tcherContext.java:486)
   at
com.ibm.ws.webcontainer.srt.SRTServletResponse.sendRedirect(SRTServletResponse.
java:810)
   at web.TestServlet.doGet(TestServlet.java:56)
   at javax.servlet.http.HttpServlet.service(HttpServlet.java:743)
   at javax.servlet.http.HttpServlet.service(HttpServlet.java:856)

[7/8/05 20:36:25:784 ART] 0000004f SRTServletRes W   WARNING: Cannot set
status. Response already committed.
```

For response generation problems, see “Response generation errors” on page 33.

Root causes

Some root causes for HTTP 500 errors are:

- Code error
- Invalid session object
- Response generation errors

Code error

The root cause of this problem is usually a programming error.

For help, look up the extended error definitions

- JSPG messages:

  http://publib.boulder.ibm.com/infocenter/wasinfo/v6r0/index.jsp?topic=/com.ibm.websphere.messages.doc/doc/JSPG.html

- JSFG messages:

  http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.messages.doc/com.ibm.ws.jsf.resources.messages.html

Correct the error and retry the file.

If you have not identified the problem, check to see if the problem has been documented by looking at the available online support (hints and tips, technotes, and fixes) for JSP problems at:

http://www-1.ibm.com/support/search.wss?rs=180&tc=SSEQTP&tc1=SSCC2GL&rankprofile=8&dc=DB520+D800+D900+DA900+DA800&dtm

If the searches fail to resolve your problem, see “The next step” on page 48.

Invalid session object

Check the servlet or JSP code that threw the exception to make sure that the session is not being invalidated too early. If that is not the case, check the session timeout interval to ensure that it is not too short.

About session objects

The session manager component uses the HttpSession interface to create a session between an HTTP client and the server. When a new session object is created, a unique session ID is assigned to it. The session ID, which is then passed as part of every request, matches the user with the session object.

Session tracking is what servlets use to maintain state and user information for multiple requests.

Sessions might be invalidated automatically if there is a session timeout or they can be ended explicitly by application code. The HttpSession interface provides the following method to terminate a session explicitly:

```
public void invalidate();
```

When a session terminates, the session object and the information that is stored in it are lost permanently. The session manager unbinds any objects that are bound to the session before it destroys the session.

HttpSession interface life cycle

Figure 14 illustrates the HttpSession interface life cycle.

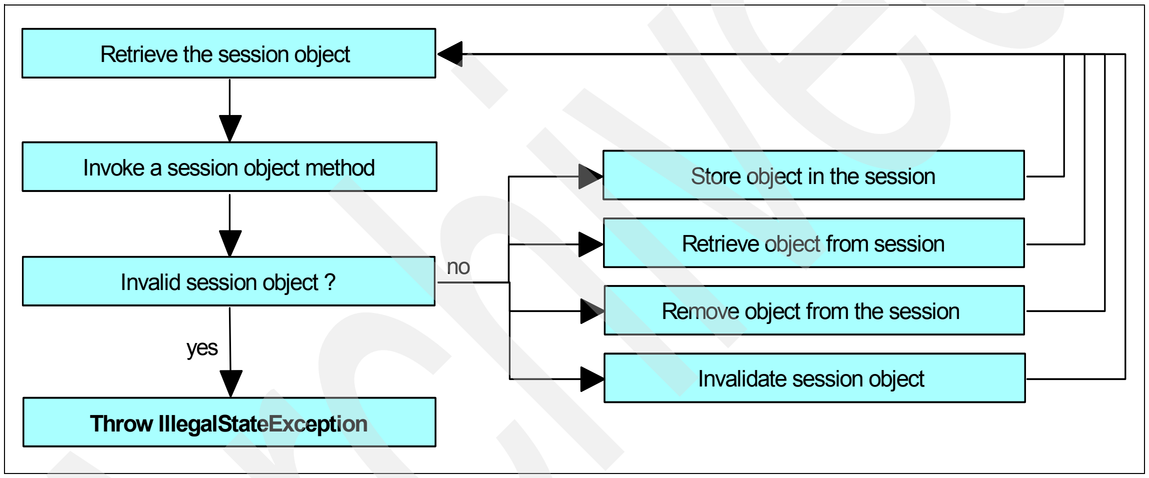

Retrieve the session object
Invoke a session object method
Invalid session object ?
no
yes
Throw IllegalStateException
Store object in the session
Retrieve object from session
Remove object from the session
Invalidate session object

Figure 14 HttpSession interface life cycle

Look at available online support

For more information, review the Java Servlet Specification Version 2.4, Section SRV.15.1.7 that relates to the HttpSession interface and methods definitions, to obtain more details about IllegalStateException creation causes:

`http://jcp.org/aboutJava/communityprocess/final/jsr154/index.html`

For current information that is available from IBM Support about known problems and their resolution related to session management, visit:

`http://www.ibm.com/support/search.wss?rs=180&tc=SSEQTP&tc1=SSCMPDS&rank profile=8&dc=DB520+D800+D900+DA900+DA800&dtm`

If these actions fail to identify your problem, see “The next step” on page 48.

Response generation errors

When an HTTP request from a client is delegated to a servlet, the service() method of the HttpServlet class is invoked. The HttpServlet class adds additional methods, such as doGet(), doPost(), doPut() and doHead(), for HTTP-based request processing.

Response already committed

The java.lang.IllegalStateException: Response already committed exception is thrown by the HttpServletResponse interface during the response generation process. If the response has been committed, you cannot execute any method that is related to HttpServletResponse object modification. For example, if you have written something in the response buffer, you cannot forward a page using the RequestDispatcher interface methods.

Other issues to look for in an application that can cause a java.lang.IllegalStateException are the following calls when the response has already been committed:

- Calling setBufferSize()
- Calling ServletResponse.reset() or ServletResponse.resetBuffer()
- Calling HttpServletResponse.sendError() or HttpServletResponse.sendRedirect().
- Calling RequestDispatcher.forward(), which includes performing a jsp:forward

Note: Remember that if you call forward() or sendRedirect() in your code, any lines of code following these still runs.

Header already sent

If you see the following message, it means that one or more headers have been committed to the client, so you cannot set that header again:

```
java.lang.IllegalStateException: Header already sent
```

Cannot forward as Output Stream, Writer already obtained

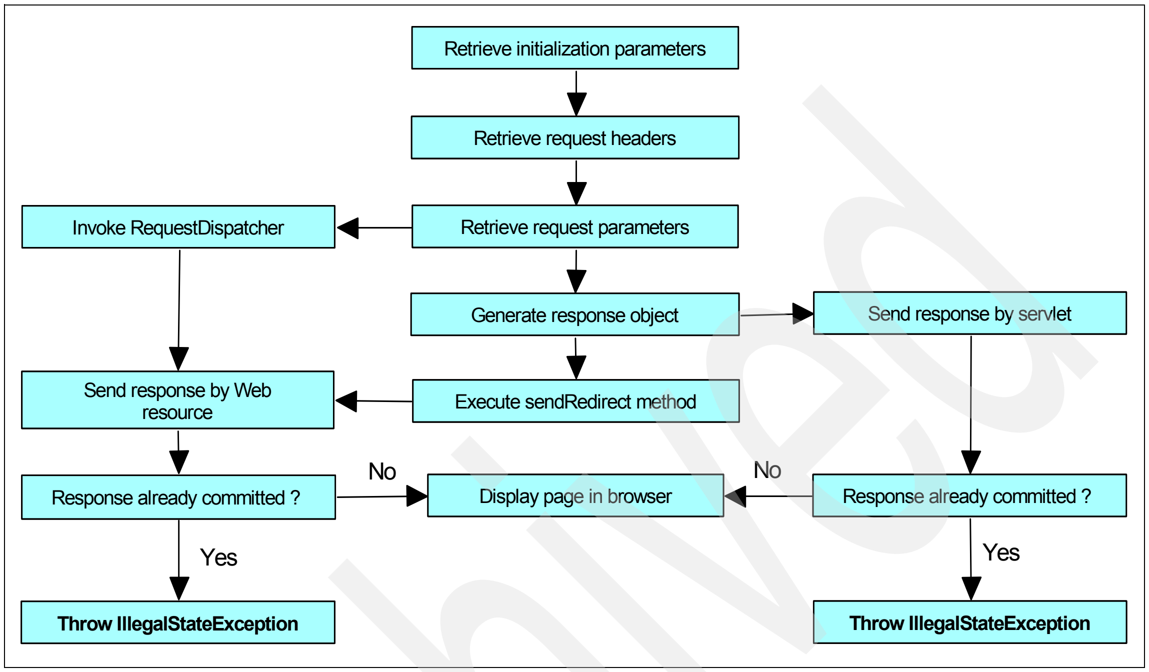
If you see the following message, it means that the calling servlet has called response.getWriter() or response.getOutputStream():

```
java.lang.IllegalStateException: Cannot forward as Output Stream or
Writer has already been obtained
```

Because the response has been written, it is unsuitable for forwarding.

The service() method life cycle

Figure 15 illustrates the life cycle of the service() method.



Retrieve initialization parameters
Retrieve request headers
Invoke RequestDispatcher
Retrieve request parameters
Generate response object
Send response by servlet
Send response by Web resource
Execute sendRedirect method
No
No
Response already committed ?
Display page in browser
Response already committed ?
Yes
Yes
Throw IllegalStateException
Throw IllegalStateException

Figure 15 Servlet service() method life cycle

Look at available online support

If you still have not identified the cause of the problem, see the Java Servlet Specification Version 2.4 at the following URL to obtain more details about the causes of IllegalStateException generation in the response generation process:

`http://jcp.org/aboutJava/communityprocess/final/jsr154/index.html`

Review the following sections:

- SRV.14.2.5 RequestDispatcher interface
- SRV.14.2.16 ServletRequest interface
- SRV.14.2.22 ServletResponse interface

For current information from IBM Support about known issues and resolutions that relate to session management, go to the following URL:

`http://www.ibm.com/support/search.wss?rs=180&tc=SSEQTP&tc1=SSCMPDF&rankprofile=8&dc=DB520+D800+D900+DA900+DA800&dtm`

If these steps do not resolve your problem, see "The next step" on page 48 for more information.

Incorrect information displayed on Web pages

If users are not receiving HTTP 404 or 500 errors, but are seeing incorrect page results in the Web browser, you could be experiencing a Web container problem. These problems are often a result of improper settings for the application in the deployment descriptors, application packaging, or Web container settings in the runtime environment.

The following symptoms can indicate a problem with the Web container:

- Users report that pages appear with missing elements.

  If a Web page appears but is missing static resources such as text, images, or file segments, see "Static resources not displayed" on page 35.

- Users report seeing an old version of application pages.

  If a JSP file has been modified and deployed to the server but the changes do not appear in the browser interface, make sure that the application has been enabled for JSP reloading. See "Web resources not reloading" on page 38.

- Users report that pages contain invalid information such as multiple question marks or garbage characters or that input is not interpreted correctly.

  It is possible that the application uses Double Byte Character Set (DBCS) characters (Japanese, Chinese, Korean languages) or specific characters for other languages that are not included in the default character encoding. In these cases, a correct character encoding configuration is necessary to display and process this information without problems. For more details related to encoding configurations, go to "Encoding and internationalization issues" on page 40.

- Users report lost data during a session.

  Users repeatedly have to enter information that should be saved during the session, lose shopping cart information, or have other short-term data loss.

  This data loss might be caused by a problem with HTTP sessions. For information about troubleshooting HTTP session problems, see:

  `http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.base.doc/info/aes/ae/rtrb_httpsessprobs.html`

Static resources not displayed

If the browser displays text output that is related to a JSP or servlet Web page but not images or HTML files, you could have a problem in the Web module packaging or with how the files are referenced in the application. Another possibility is that the file serving feature needs to be turned on for your application.

Verify the static resource file locations

Verify that your files are in the right place and that the document root directory of the Web application module follows the J2EE standard (the document root is in the *web_module*.war directory of the deployed application EAR file). Typically this directory is in this location:

app_server_root/profiles/*profile*/installedApps/*node*/*application*.ear/*web_module*.war/

If the image files are in a subdirectory of the document root, verify that the reference to the image reflects that (Example 11).

Example 11 Image reference in HTML tags

```
File Location: <web_module_name.war>/images/test.gif
Correct HTML tag: <img SRC="images/test.gif">
Incorrect HTML tag: <img SRC="test.gif">
```

Note: Do not place files to be served to the client in the WEB-INF directory.

Check the file serving feature

File serving is how a Web application serves static file types (HTML, images, and style sheets). This process uses the enable file servlet, also known as the file serving servlet or file serving enabler. This servlet serves up any resource file that is packaged in the WAR file, and the file serving attribute is set to **true** by default.

File serving is implemented by setting the fileServingEnabled property to **true** when you configure the Web module. If it is set to **false**, the Web server plug-in does not send requests for static content to the application server but leaves it up to the Web server to serve them. Serving the page from the Web server provides a shorter path to the page and usually provides more customization options than the file servlet can offer.

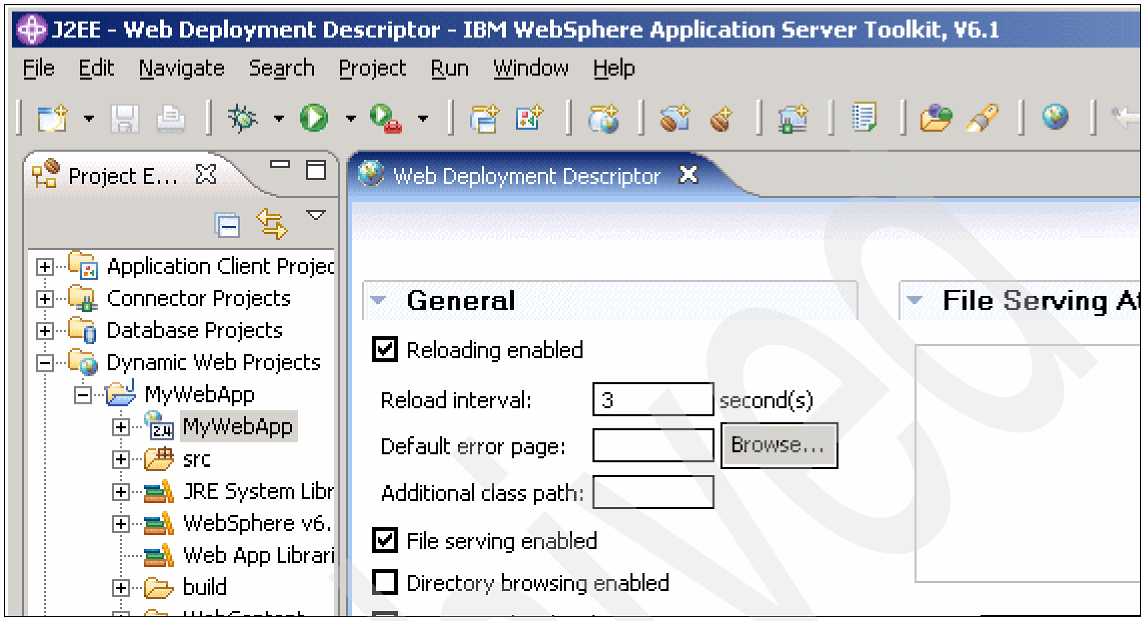
The fileServingEnabled property is in the ibm-web-ext.xmi configuration file at:

profile_root/installedApps/*node*/*application*.ear*web_module*.war/WEB-INF/ibm-web-ext.xmi

To update the property using the Application Server Toolkit:

1. Go to the Project Explorer view in the J2EE perspective and select the target Web application module.
2. Double-click the Web deployment descriptor and click the Extensions tab to see the IBM Web module extensions.

3. In the General section, select **File serving enabled** (Figure 16) to enable the static file serving or clear it to disable the static file serving.


J2EE - Web Deployment Descriptor - IBM WebSphere Application Server Toolkit, V6.1
File Edit Navigate Search Project Run Window Help
Project E...
Application Client Projec
Connector Projects
Database Projects
Dynamic Web Projects
MyWebApp
MyWebApp
src
JRE System Libr
WebSphere v6.
Web App Librari
build
Web Deployment Descriptor
General
File Serving At
Browse...

☑ Reloading enabled

Reload interval: 3 second(s)

Default error page:

Additional class path:

☑ File serving enabled

☐ Directory browsing enabled

Figure 16 Enabling file serving

4. Save the Web deployment descriptor file.
5. Redeploy the Web module.
6. Regenerate the Web server plug-in and propagate it to the Web server.
7. Stop and restart the Web server or allow enough time for the new plug-in configuration to be reloaded.
8. Retry the Web request.

For more information about this feature, see:

- *File serving*

  http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.base.doc/info/aes/ae/cweb_filserv.html

- *Customizing SimpleFileServlet* to disable file serving at:

  http://www.ibm.com/support/docview.wss?uid=swg21116838

- Supported assembly tools in WebSphere Application Server V6:

  http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.base.doc/info/aes/ae/catk_assemblytools.html

Web resources not reloading

If, after modifying and saving a servlet or JSP file, the change does not show up in the browser, you need to check the reload settings in the Web module configuration and the JSP runtime reload settings.

Web module reloading

For Web resources such as servlets and JSPs, the Web container reloads a Web module only when the IBM extension reloadingEnabled in the ibm-web-ext.xmi file is set to **true**. You can do this when you are editing the extended deployment descriptors of the Web module in a development or assembly tool.

To configure these settings with the Application Server Toolkit:

1. From the Project Explorer view of the J2EE perspective, double-click the Web deployment descriptor and click the Extensions tab.
2. In the General section, select **Reloading enabled** (see Figure 16 on page 37). If it is already selected, set the **Reload interval** lower.
3. Save the Web deployment descriptor file.
4. Redeploy the Web module.
5. Regenerate the Web server plug-in and propagate it to the Web server.
6. Stop and restart the Web server or wait for the new plug-in file to take effect.
7. Retry the Web request.

The entries in ibm-web-ext.xmi look similar to those in Example 12.

Example 12 Web module reloading settings in ibm-web-ext.xmi

```
<?xml version="1.0" encoding="UTF-8"?>
<webappext:WebAppExtension xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI"
xmlns:webappext="webappext.xmi" xmi:id="WebAppExtension_1176282340062"
reloadInterval="3" reloadingEnabled="true" additionalClassPath=""
fileServingEnabled="true" directoryBrowsingEnabled="false"
serveServletsByClassnameEnabled="true">
  <webApp href="WEB-INF/web.xml#WebApp_ID"/>
</webappext:WebAppExtension>
```

Configure JSP runtime reloading

JSP files can be translated and compiled at runtime when the JSP file or its dependencies are modified. This is known as JSP reloading, and it is enabled through the reloadEnabled JSP engine parameter in the ibm-web-ext.xmi configuration file.

To configure JSP reload using the Application Server Toolkit:

1. From the Project Explorer view of the J2EE perspective, double-click the Web deployment descriptor and select the Extensions tab.
2. In the JSP Attributes section (Figure 17), click **Add**.

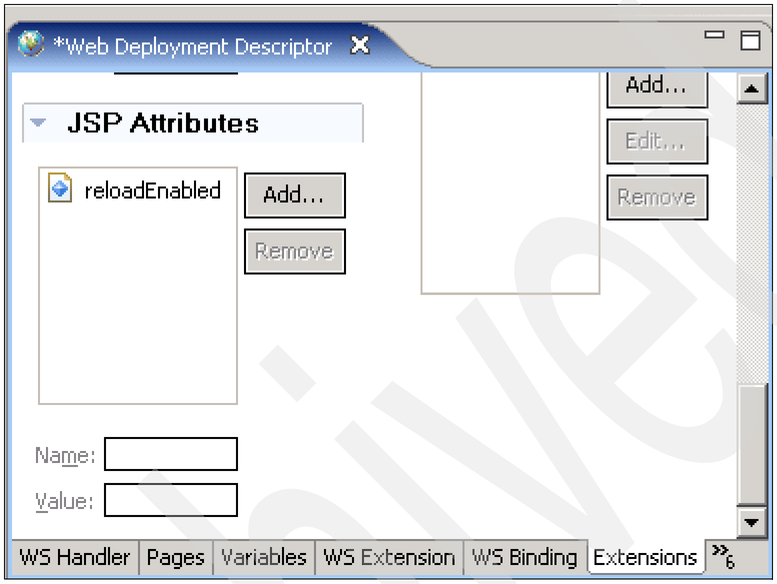


Figure 17 JSP attributes in IBM Web module extensions

3. Enter `reloadEnabled` in the Name field, `true` in the Value field, and click **Finish**.
4. Save the Web deployment descriptor file.
5. Redeploy the Web module.
6. Regenerate the Web server plug-in and propagate it to the Web server.
7. Stop and restart the Web server.
8. Retry the Web request.

The JSP engine settings are stored in ibm-web.ext.xmi (Example 13).

Example 13 JSP engine settings in ibm-web-ext.xmi

```
?xml version="1.0" encoding="UTF-8"?>
<webappext:WebAppExtension xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI"
xmlns:webappext="webappext.xmi" xmi:id="WebAppExtension_1176282340062"
reloadInterval="3" reloadingEnabled="true" additionalClassPath=""
fileServingEnabled="true" directoryBrowsingEnabled="false"
serveServletsByClassnameEnabled="true">
```

```
  <webApp href="WEB-INF/web.xml#WebApp_ID"/>
  <jspAttributes xmi:id="JSPAttribute_1176283583578" name="reloadEnabled" 
value="true"/>
</webappext:WebAppExtension>
```

For more available JSP attributes and details about the reload processing sequence, see “JavaServer™ Pages™ (JSP) runtime reloading settings” in the WebSphere Information Center at:

http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.base.doc/info/aes/ae/rweb_jspreloading.html

Encoding and internationalization issues

Encoding and internationalization problems usually appear as garbage characters in Web pages. Another common symptom is that input from the user is interpreted incorrectly.

DBCS character processing is a common issue for encoding problems. DBCS is used for languages such as Chinese, Korean, and Japanese, where a single byte is not sufficient to represent all characters in the alphabet. Proper coding and configuration usually resolves these problems.

Character encoding

Web components usually use the java.io.PrintWriter object to produce responses; PrintWriter automatically encodes with ISO 8859-1 character encoding. Servlets can also output binary data using java.io.OutputStream objects, which perform no encoding.

An application that cannot use the default encoding must explicitly set a different encoding.

Figure 18 on page 41 shows a Web page with improper encoding settings.

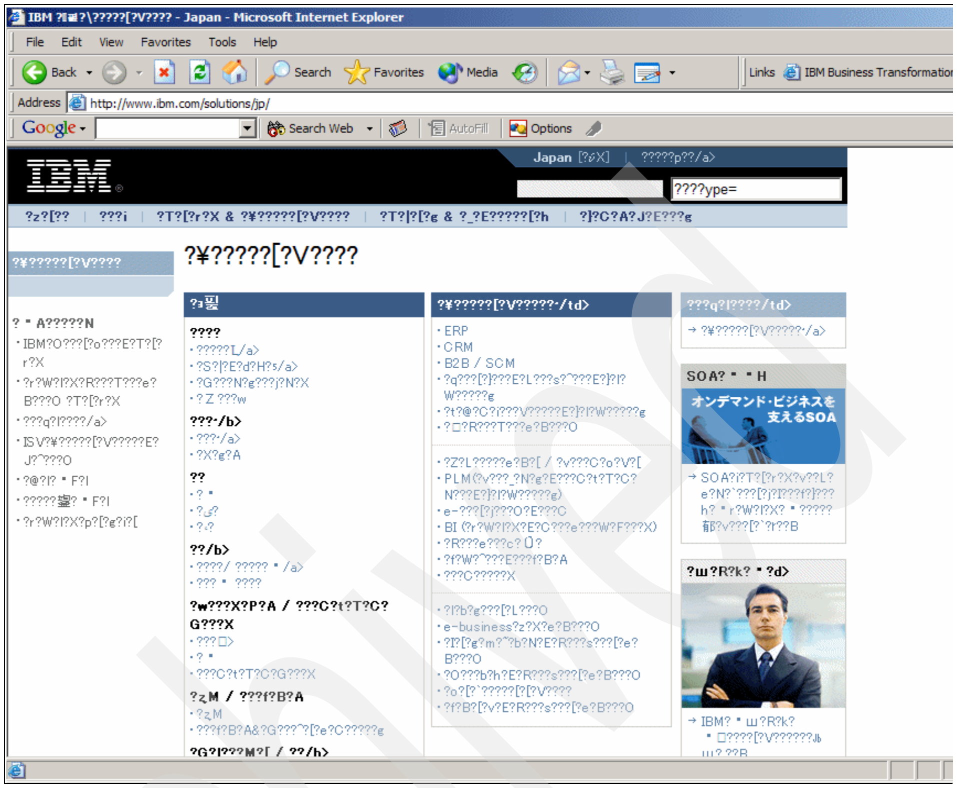

IBM ?ﾘ■?\?????[?V???? - Japan - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Links IBM Business Transformation
Address http://www.ibm.com/solutions/jp/
Google Search Web AutoFill Options
IBM
Japan
?¥?????[?V????
ERP
CRM
B2B / SCM
SOA
オンデマンド・ビジネスを支えるSOA

Figure 18 Web page with invalid character encoding settings

For Web components, three encodings must be considered:

- Request
- JSP
- Response

Request encoding

Request encoding is the character encoding where parameters in an incoming request are interpreted. Currently, many browsers do not send a request encoding qualifier with the content-type HTTP header. In such cases, a Web container uses the default encoding, which is ISO-8859-1, to parse request data.

If the client has not set character encoding and the request data is encoded with a different encoding from the default, the data is not interpreted correctly.

To correct this situation, you can use the setCharacterEncoding(String enc) method to override the character encoding that is supplied by the container (Example 14 on page 42).

Example 14 setCharacterEncoding() method implementation

```
public class TestServlet extends HttpServlet {

   public void doPost(HttpServletRequest req, HttpServletResponse resp)
      throws ServletException, IOException {

         req.setCharacterEncoding("UTF-8");
         String name = req.getParameter("name");
         resp.setContentType("text/html;charset=UTF-8");
         PrintWriter out = resp.getWriter();
         out.println("<html><body><h1>");
         out.println("Your name is "+name);
         out.println("</h1></body></html>");
   }
}
```

You must call the method before parsing any request parameters or reading any input from the request. Calling the method or tag after data has been read does not affect the encoding.

Page encoding

For JSP, page encoding is the character encoding where the file is encoded. For JSP in standard syntax, it is determined from the following sources:

- The pageEncoding attribute of the page directive of the page
- The charset value of the contentType attribute of the page directive

If none of these is provided, ISO-8859-1 is used as the default page encoding (Example 15). The pageEncoding and contentType attributes determine the page character encoding only for the file that physically contains the page directive.

Example 15 Page directive implementation

```
<HTML>
<HEAD>
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="ISO-8859-15" %>

<META http-equiv="Content-Type" content="text/html; charset=UTF-8">
<TITLE>Receiving parameters</TITLE>
</HEAD>
<BODY>
<H1>
Your data is <%= request.getParameter("name") %>
</H1>
</BODY>
</HTML>
```

Response encoding

Response encoding is the character encoding of the textual response that is generated from a Web component. A Web container sets an initial response encoding for a JSP page from the following sources:

- The charset value of the contentType attribute of the page directive
- The encoding specified by the pageEncoding attribute of the page directive

If none of these is provided, ISO-8859-1 is used as the default response encoding.

The setContentType() method in a servlet can be called to change the character encoding. Calls made after the getWriter() method has been called or after the response is committed do not affect the character encoding.

Struts character encoding settings

For Struts, consider the following methods for encoding:

- Request character encoding

  In a Struts environment, call the setCharacterEncoding() method in the ActionForm (Example 16).

Example 16 Request encoding in Struts implementation

```
public class PostMessageForm extends ActionForm {
   public void reset(ActionMapping mapping, HttpServletRequest request) {
      try {
         request.setCharacterEncoding("UTF-8");
      }catch (UnsupportedEncodingException e) {
         e.printStackTrace();
      }
   }
}
```

- Response character encoding

  Specify the response encoding using the contentType attribute with a charset value in the JSP page directive.

WebSphere Application Server settings

In normal circumstances, programmers are expected to determine the encoding settings in the application. However, when you are diagnosing problems, it is useful to set them temporarily or override them in the runtime configuration.

Request/response character encoding

WebSphere determines the character encoding that is used for request/response by parsing the client input values in getParameter() and writing the output based on the value in the accept-language header of the incoming request.

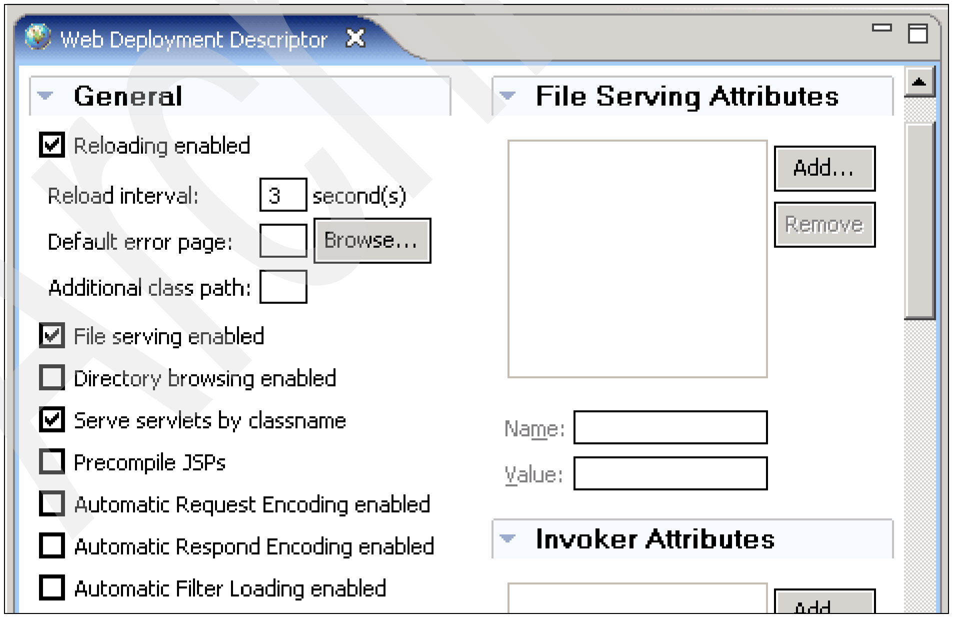
The language value and corresponding character encoding names are associated in the encoding.properties file in the *app_server_root*/properties directory. You can override the definition in this file by setting the client.encoding.override JVM command line argument.

For information about using this argument, see “Configuring application servers for UCS Transformation Format” at:

```
http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/
com.ibm.websphere.base.doc/info/aes/ae/trun_svr_utf.html
```

autoRequestEncoding and autoResponseEncoding

Programmers set request and response encodings and response content types using available methods in the Servlet specification. However, you can specify the autoRequestEncoding and autoResponseEncoding extensions so that the application server sets the encoding values and content type, using the Application Server Toolkit. The settings are found in the Web Deployment Descriptor on the Extensions tab (Figure 19).


Web Deployment Descriptor

General

- [✓] Reloading enabled
- Reload interval: 3 second(s)
- Default error page:
- Additional class path:
- [✓] File serving enabled
- [ ] Directory browsing enabled
- [✓] Serve servlets by classname
- [ ] Precompile JSPs
- [ ] Automatic Request Encoding enabled
- [ ] Automatic Respond Encoding enabled
- [ ] Automatic Filter Loading enabled

File Serving Attributes

Name:
Value:

Invoker Attributes

Figure 19 Automatic request/response encoding settings

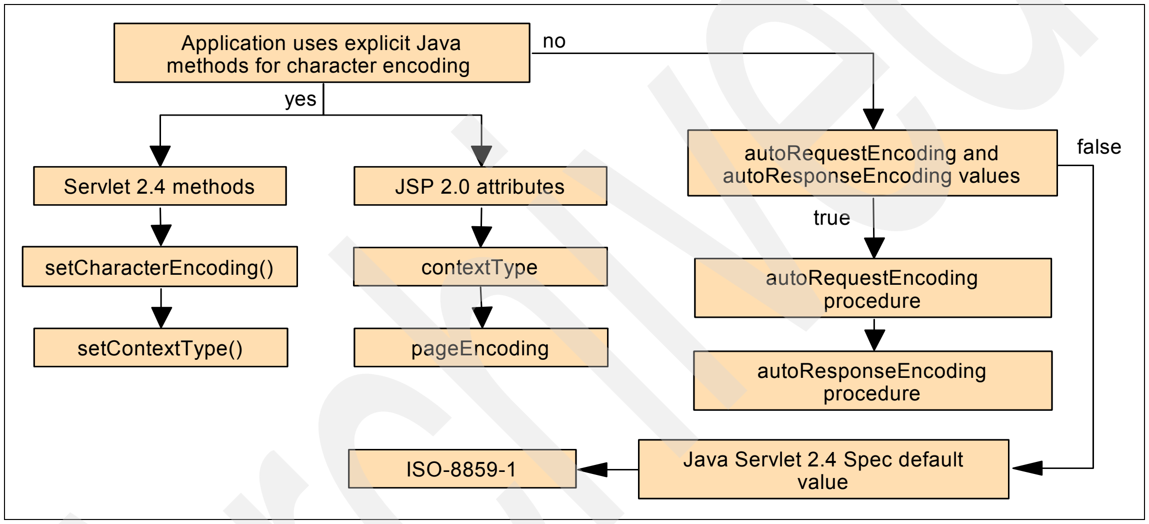
The default value for both is false, which means that the request and response character encoding is set to ISO-8859-1, the Servlet 2.4 Specification default.

For a description of Web container behavior when these values are set to **true,** see "autoRequestEncoding and autoResponseEncoding," an article in the WebSphere Information Center at:

`http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.nd.doc/info/ae/ae/cweb_autoreq.html`

Summary

Figure 20 shows a summary of how the encoding values are determined.



Application uses explicit Java methods for character encoding
no
yes
Servlet 2.4 methods
setCharacterEncoding()
setContextType()
JSP 2.0 attributes
contextType
pageEncoding
autoRequestEncoding and autoResponseEncoding values
false
true
autoRequestEncoding procedure
autoResponseEncoding procedure
Java Servlet 2.4 Spec default value
ISO-8859-1

Figure 20   Encoding determination process

Look at available online resources and support

If none of these steps fixes your problem, the following resources might be helpful:

- *Globalize your On Demand Business* for tips on character encoding:

  `http://www-306.ibm.com/software/globalization/j2ee/encoding.jsp`

- *JSR-000154 Java Servlet 2.4 Specification* for details about character encoding methods:

  `http://jcp.org/aboutJava/communityprocess/final/jsr154/index.html`

- *JSR-000152 JavaServer Pages 2.0 Specification*, for issues related to internationalization:

  http://jcp.org/aboutJava/communityprocess/final/jsr152/index.html
- *Internationalization: Resources for learning:*

  http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.base.doc/info/aes/ae/rin_resources.html

For current information available from IBM Support about known issues and resolutions that are related to encoding, see:

http://www.ibm.com/support/search.wss?rs=180&tc=SSEQTP&tc1=SSCVRZX&rankprofile=8&dc=DB520+D800+D900+DA900+DA800&dtm

If these steps still do not resolve your problem, see "The next step" on page 48..

Collect diagnostic data

This section provides guidance for collecting log files and trace data that are useful in diagnosing Web container problems.

JVM logs

JVM logs, often referred to as SystemOut and SystemErr logs, are created for every WebSphere Application Server process (application server, cluster member, node agent, and deployment manager). They can be found in the following locations:

- WebSphere Application Server V6.x for IBM z/OS®

  The JVM logs are located in the address space output. Usually a section labeled SYSOUT has diagnostic data from the JVM that runs in the servant region.
- WebSphere Application Server V6.x (distributed and i5/OS)

  The JVM log files are by default named SystemOut.log and SystemErr.log. The default location for the SystemOut and SystemErr logs is:

  – *profile_root*/logs/*server_name*/SystemOut.log
  – *profile_root*/logs/*server_name*/SystemErr.log:

The location of application server logs is configurable.

1. Select **Troubleshooting** → **Logs and Trace** in the navigation bar.
2. Click the server name.
3. Select **JVM logs**.

IBM HTTP Server, HTTP Server (powered by Apache) log files

These Web servers log every client request in the access.log file. If client users experience errors, such as an error logging on to a content server, or errors importing a document, you can use the access.log file to verify the resource manager Web address and to verify that the client request is getting through to the Web server.

By default, two logs in *Web_server_root*/logs are used:

- access.log
- error.log

You can use the ErrorLog and CustomLog directives to configure the location of these logs. You can also use the LogFormat directive to configure the format and combine all the logs into one log. For information about using these directives, see the comments in the httpd.conf file for the Web server.

Trace Web server requests to WebSphere (i5/OS)

When an HTTP 404 error occurs in the browser, the problem is often caused by a problem with the plug-in. The plug-in file has the rules that the HTTP server follows to determine what content it can pass to the WebSphere Application Server. When HTTP is unable to match any of these rules in the plug-in file, it tries to serve the page based on its own configuration and the HTTP server sends the 404 message.

Collect the TRCTCPAPP trace

To see whether the HTTP Apache server is passing requests properly to WebSphere, trace the HTTP as follows:

1. On the i5/OS command line, issue the following command to enable the trace (replace *server_name* with the name of your HTTP server instance):

   ```
   TRCTCPAPP APP(*HTTP) HTTPSVR(server_name) TRCLVL(*VERBOSE)
   ```

2. Reproduce the error (make sure that the browser cache has been cleared so that it makes a fresh request to the server).
3. To turn off the trace, issue the following command:

   ```
   TRCTCPAPP APP(*HTTP) SET(*OFF)
   ```

   This will produce a spool file with a file name of QZSRHTTPTR in the out queue of the user profile that issued the trace commands.

The next step

The symptoms and problem areas in this paper are those 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 container. This section provides information about obtaining additional support.

Search online support

For hints and tips, technotes, and fixes about Web container problems, see:

http://www.ibm.com/support/search.wss?rs=180&tc=SSEQTP&tc1=SSCMPDF

If the problem is performance related, these references are useful for configuring the JSP engine for optimal performance:

- *JSP engine configuration parameters*

  http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.nd.doc/info/ae/ae/rweb_jspengine.html

- *JSP class file generation*

  http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.nd.doc/info/ae/ae/cweb_jspclassfiles.html

- *JSP run time compilation settings* (to disable run time compilation)

  http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.nd.doc/info/ae/ae/rweb_jspdis.html

- *Packages and directories for generated .java and .class files*

  http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.nd.doc/info/ae/ae/cweb_javapkg.html

Note: These URLs are for WebSphere Application Server Network Deployment topics on distributed platforms. If you are using a different package or platform, you can find the comparable item by searching the title in the Information Center.

Reevaluate the symptoms

If, after going through this process, you still have an undiagnosed problem, see *Approach to Problem Determination in WebSphere Application Server V6* at:

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

Review the problem classifications to see if there are any other components that might be causing the problem.

Contact IBM

If your search does not provide any information that is relevant to your problem and you feel sure that you have a Web container problem, it might be time to contact IBM support.

Select the MustGather document from the following list that most closely resembles your problem. If you are not sure which that is, use the first MustGather listed (Web container errors).

The MustGather documents for Web container problems are:

- *MustGather: Web container errors on WebSphere Application Server V6.1*

  `http://www-1.ibm.com/support/docview.wss?uid=swg21252138`
- *MustGather: JavaServer Pages (JSP) problems*

  `http://www-1.ibm.com/support/docview.wss?uid=swg21255205`
- *MustGather: Java Server Faces (JSF) problems in V6.x*

  `http://www.ibm.com/support/docview.wss?uid=swg21198110`
- *MustGather: i18n (Internationalization) / Double Byte Character Set (DBCS)*

  `http://www-1.ibm.com/support/docview.wss?rs=180&uid=swg21141732`

Be sure to note all of the diagnostic work that you have done to minimize the amount of time that it takes IBM Support to assist you.

Approaching problems with Web servers and plug-ins

WebSphere Application Server supplies Web server plug-ins for a range of Web server types. You can find the supported Web server products in “System Requirements for WebSphere Application Server V6.1” at:

`http://www-1.ibm.com/support/docview.wss?rs=180&uid=swg27007651`

If you think you have a problem with a Web server or plug-in:

- Make sure that you are using a supported Web server.
- Make sure that the Web server is responding.
- Make sure that the plug-in configuration file is current.

This section gives you a general approach to problem determination for the Web servers and Web server plug-ins that are supplied with WebSphere Application Server or IBM operating systems. For other Web server types, see the documentation for that specific product.

Web servers

When an application server does not respond to incoming requests, check to see if static HTML pages can be served by the HTTP Server. If not, the problem is probably caused by the HTTP Server. The HTTP Server process might have stopped unexpectedly (a crash) or it might be running but not responding to requests (a hang). You can check to see if the Apache process (httpd) is running on your operating system to determine if the process has crashed or if it is hanging. Other possible problems with HTTP Server include configuration issues, caching issues, authentication problems, and SSL problems. If the HTTP Server is able to serve static pages, the root cause of the problem is probably with the Web server plug-in or the application server itself.

Note: There is a possibility that the HTTP Server could fail to start due to a Web server plug-in problem. For more information about this, see *WebSphere Application Server V6: Web Server Plug-in Problem Determination* at:

`http://www.redbooks.ibm.com/redpapers/pdfs/redp4045.pdf`

IBM HTTP Server

IBM HTTP Server V6.1 is included with WebSphere Application Server V6.1, and you use the administrative console to administer it. Other Web servers can be configured with WebSphere Application Server. In general, the same problems scenarios occur with them.

You can find problem determination strategies for IBM HTTP Server issues in the following resources:

- IBM HTTP Server Support

  `http://www-306.ibm.com/software/webservers/httpservers/support`

- IBM HTTP Server, Version 6.1 Information Center

  `http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.ihs.doc/info/welcome_ihs.html`

- IBM HTTP Server, Version 6.1 Information Center, *Troubleshooting IBM HTTP Server*

  `http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.ihs.doc/info/ihs/ihs/tihs`

- *MustGather: Read first for IBM HTTP Server*

  `http://www-1.ibm.com/support/docview.wss?rs=177&uid=swg21192683`

HTTP Server (powered by Apache)

HTTP Server (powered by Apache) is included with the i5/OS licensed programs and must be selected for installation. You can find problem determination strategies for HTTP Server (powered by Apache) issues in the following resources:

- IBM HTTP Server for iSeries® Support

  `http://www-03.ibm.com/servers/eserver/iseries/software/http/services/service.html`

- IBM HTTP Server for iSeries documentation

  `http://www-03.ibm.com/servers/eserver/iseries/software/http/docs/doc.html`

Web server plug-in

The Web server plug-in enables the Web server to send requests for dynamic content to Web applications (servlets and JSPs) that are installed in WebSphere Application Server. The configuration file used by the plug-in is created in the WebSphere Application Server system and propagated to the Web server.

To determine if you have a plug-in problem, first ensure that the Web server is responding. Then, try to access a servlet or JSP directly on the application server, bypassing the plug-in. You can do this by using the HTTP transport port (port 9080 by default) for the application server in the URL to access the servlet or JSP. For example, you can try to access the snoop servlet, which is one of the WebSphere Application Server samples:

```
http://localhost:9080/snoop
```

If the servlet or JSP can be accessed through the HTTP transport but not through the Web server plug-in, you are likely experiencing a problem with the plug-in. On the other hand, if it cannot be accessed through the HTTP transport or the plug-in, it is probably an application server problem. Potential Web server plug-in problems include configuration issues (especially with the plugin-cfg.xml configuration file) and failures to send requests to an application server.

WebSphere system messages for the plug-in begin with PLGN, PLGC, and PLPR.

Note: For problem determination strategies for the Web server plug-in, see *WebSphere Application Server V6: Web Server Plug-in Problem Determination* at:

http://www.redbooks.ibm.com/redpapers/pdfs/redp4045.pdf

Managing servers and applications

This section includes information about managing applications and application servers in WebSphere Application Server.

Check the status of an application

To see the status of each module on the server, select **Applications** → **Enterprise Applications.**

This action displays a table of the applications that are deployed in the cell. The status icon in the Application Status field indicates the status of the application. Moving your mouse over the icon displays the status. Any status other than Started indicates a problem. However, an Unavailable status might simply mean that the node agent for the server that the application is running on is down.

Determine the server or cluster that hosts the application

To view the mapping of the application modules to servers, select **Enterprise Applications** → ***application_name*** → **Manage Modules.**

The list that results from this action shows the server mapping for each module in the application.

Check the status of application modules on the target servers

To see the status of each module on the server it is mapped to:

1. Select **Enterprise Applications** → ***application_name*** → **Manage Modules.**
2. Click the module and select **Target specific application status.**

Check the status of application servers

You can check the status of an application server using the administrative console or by command.

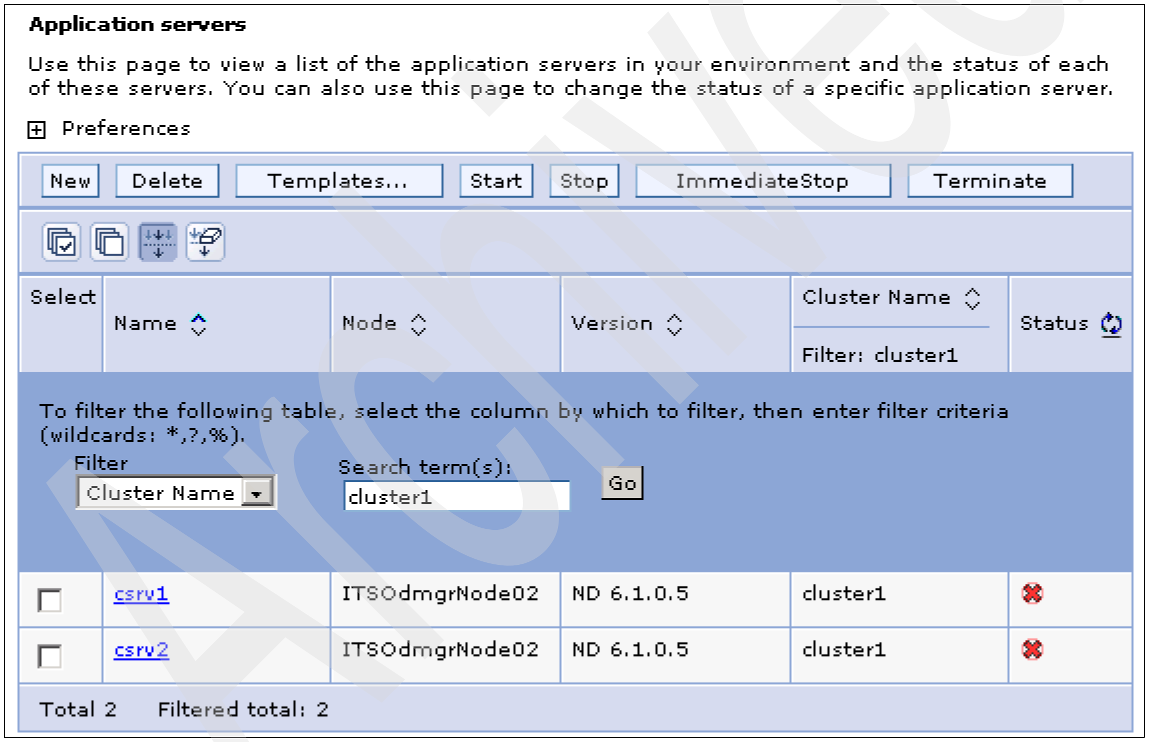
Note: In a single server environment, the administrative process runs on the application server. If you can access the administrative console, the application server is active.

Using the administrative console

To use the administrative console:

1. Select **Servers** →**Application Servers** to list the servers (Figure 21).

Tip: Use the filter function to manage how the servers are listed. For example, you can filter on a cluster name to see all the servers in a cluster.



Application servers

Use this page to view a list of the application servers in your environment and the status of each of these servers. You can also use this page to change the status of a specific application server.

⊞ Preferences

To filter the following table, select the column by which to filter, then enter filter criteria (wildcards: *,?,%).

Filter: Cluster Name

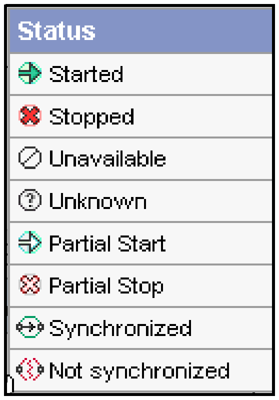
Search term(s): cluster1

Select	Name	Node	Version	Cluster Name Filter: cluster1	Status
☐	csrv1	ITSOdmgrNode02	ND 6.1.0.5	cluster1	✖
☐	csrv2	ITSOdmgrNode02	ND 6.1.0.5	cluster1	✖

Total 2 Filtered total: 2

Figure 21 Display of servers in a cluster

2. The last column to the right has an icon that indicates the status of each server. Figure 22 on page 54 shows the icons and the corresponding status.



Status
Started
Stopped
Unavailable
Unknown
Partial Start
Partial Stop
Synchronized
Not synchronized

Figure 22 Status icons

If the server status is Unavailable, the node agent on the node where the application server is installed is not active. The server cannot be managed from the administrative console unless its node agent is active.

Using the command line

Distributed and z/OS platforms supply the serverStatus command in the *profile_root*/bin directory.

The syntax of the serverStatus command is:

```
serverStatus.bat(sh) <server>|-all [options]
```

Check a specific server with the serverStatus command as follows:

- For Windows®: *profile_root*\bin\serverStatus *server_name*
- For UNIX® and z/OS: *profile_root*/serverStatus.sh *server_name*

For i5/OS, issue:

```
WRKACTJOB SBS(QWAS61)
```

Start application servers

In a single server environment, you can use a command to start the application server.

In a managed environment, you can use a command in the system where the server runs or you can start the server from the administrative console of the deployment manager. However, the node agent for the application server node must be up before you can use the console.

Using the administrative console

To start one or more servers:

1. Select **Servers →Application Servers** to list the servers.
2. Select the server you want to start and click **Start**.

Application servers in a cluster can be managed as independent servers. A second option is to manage all the servers in the cluster using a single button. In the administrative console:

1. Select **Servers →Clusters**.
2. Select a cluster and select one of the following options:
   - **Start**: Use this option to start all servers in the cluster.
   - **Ripplestart**: Use this option to stop and start all servers in the cluster.

If you want to stop and restart all the application servers on a node::

1. From the administrative console, select **System Administration →Node Agents**.
2. Select the node agent.
3. Click **Restart all Servers on the Node**.

Using the command line

You can use the startServer command to start the application server on distributed and z/OS platforms:

- Windows: *profile_root*\bin\startServer *server_name*
- i5/OS: *profile_root*/bin/startServer *server_name*
- UNIX and z/OS: *profile_root*/bin/startServer.sh *server_name*

Starting a job (z/OS)

Use the following command to start the server:

```
START appserver_procname,JOBNAME=server_shortname,
ENV=cell_shortname.node_shortname.server_shortname
```

Notices

Send us your comments in one of the following ways:

- Use the online **Contact us** review Redbooks form found at:
  ibm.com/redbooks
- Send your comments in an e-mail to:
  redbook@us.ibm.com
- Mail your comments to:
  IBM Corporation, International Technical Support Organization
  Dept. HYTD Mail Station P099, 2455 South Road
  Poughkeepsie, NY 12601-5400 U.S.A.

Redpaper

Trademarks

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

Redbooks (logo) ®	Rational®	WebSphere®
iSeries®	AIX®	
i5/OS®	IBM®	

The following terms are trademarks of other companies:

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

Internet Explorer, Windows, 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.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

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