



IBM Software Group

SW5706 Request flow problems: Where did my request go?



© 2007 IBM Corporation

4.0

This unit focuses on problem determination techniques and tools associated with request flow problems.

Unit objectives

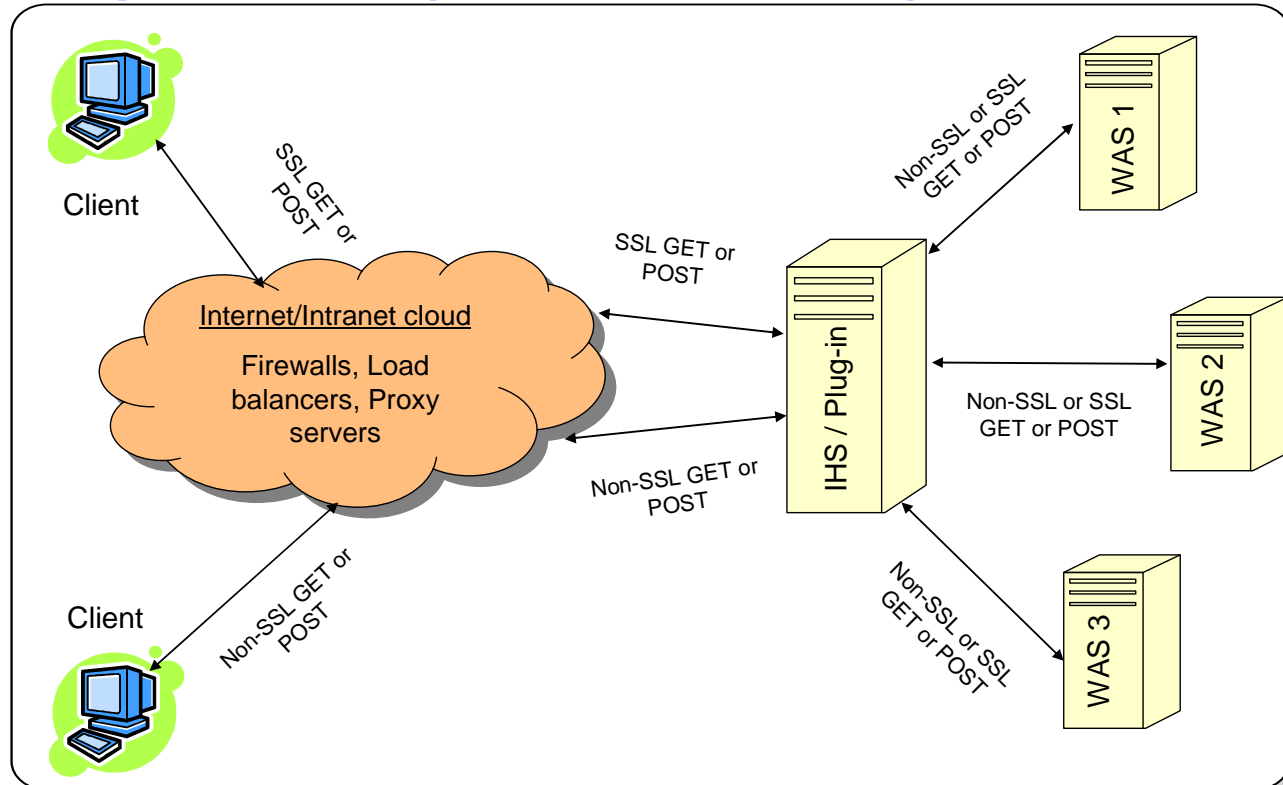
After completing this unit, you should be able to:

- Identify working and non-working requests
- Identify key questions needed to resolve request problems
- Determine the appropriate tracing necessary to debug problem requests
- Read logs to pinpoint the source of failure



This topic explores client requests and possible failure points that can occur. By the end of this unit, you will be able to identify the differences between working and non-working requests, know what questions are relevant to resolving request problems, as well as use the WebSphere® tracing and tooling to pinpoint the source of failure.

Request flows (SSL and non-SSL)



This diagram depicts how requests move between a client and Websphere Application Server. The Internet/Intranet cloud represents components in the request flow that can cause problems but are not discussed in this unit. If you do suspect a problem in one of these gateways, you can determine if a problem exists by taking a packet trace from the IHS to verify the content and movement of your requests.

Review of a working request: Outline

- Typical request methods
 - ▶ GET
 - ▶ POST
- Common protocols
 - ▶ HTTP
 - ▶ HTTPS
- Browser return codes
 - ▶ 200
 - ▶ 302
 - ▶ 304
- Identifying working requests through the logs
 - ▶ Review Web server access log (GET & POST)
 - ▶ Review plug-in log (GET & POST)

There are several paths a request can take and a number of different scenarios associated with each variation. This unit focuses in on the most common requests and a selection of the problems you are most likely to experience. The most common request methods are get and post but there are also head, put, and delete. In this unit, we focus on get and post. This course also focuses on the two most common protocols, http and secured https. Within this realm there are three possible browser return codes that we are going to explore.

Typical request methods: GET

Plants by WebSphere - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Address: http://localhost/PlantsByWebSphere/


PLANTS BY WEBSHERE

Flowers Fruits & Vegetables Trees Accessories

Gardens of Summer

They all start with the right flowers... and we've got them all

Tips
Preserve extra grass seed by keeping it dry. Tape boxes and bags closed, or seal them into plastic bags. Be sure to remove extra air from the bags. Store all seed in a cool, dry area such as a garage or basement.

Specials

Bonsai Tree
\$30.00 each

Powered by IBM WebSphere e-business software

Flowers : Fruits & Vegetables : Trees : Accessories
Home : Shopping Cart : My Account : Login : Help

Request flow problems © 2007 IBM Corporation 5

The get method is used when a request is seeking content from a host application. This is commonly used when loading a page in a browser. Here we see the sample application, Plants by WebSphere, that ships with WebSphere Application Server.

Typical request methods: POST

The screenshot shows a Microsoft Internet Explorer browser window displaying the 'Plants by WebSphere' website. The address bar shows 'http://localhost/PlantsByWebSphere/'. The page title is 'PLANTS BY WEBSHERE'. There are navigation tabs for 'Flowers', 'Fruits & Vegetables', 'Trees', and 'Accessories'. The 'Home' link is visible. The main heading is 'Shopping Cart'. Below the heading, there is a paragraph of instructions: 'Here are the items you have selected. To recalculate your total after changing the quantity of an item, select the 'Recalculate' button. To remove an item from your cart, enter "0" as the quantity. Select 'Checkout Now' to begin the checkout process.'

ITEM #	ITEM DESCRIPTION	PACKAGING	QUANTITY	PRICE	SUBTOTAL
T0003	Bonsai	0.5 gallon mature tree	<input type="text" value="2"/>	\$30.00	\$60.00

Order Subtotal:\$60.00

Buttons: Continue Shopping, Recalculate, Checkout Now

Powered by IBM WebSphere e-business software

Navigation: Flowers : Fruits & Vegetables : Trees : Accessories
Home : Shopping Cart : My Account : Login : Help

Request flow problems © 2007 IBM Corporation 6

The post method is used to submit data to a web based application. This is how user input is often submitted to an application. Here we see an example of post being used to submit the quantity of desired bonsai trees in the Plants by WebSphere application.

Common protocols: HTTP

- HyperText Transfer Protocol (HTTP) is the method used to transfer or convey information on the World Wide Web. It is a patented open internet protocol whose original purpose was to provide a way to publish and receive HTML pages.
- The default TCP port of an HTTP Uniform Resource Identifier (URI) is **80**.
- Sample: **http://localhost/PlantsByWebSphere**
http://localhost:80/PlantsByWebSphere

Both of the previous examples use the default http method to communicate between the client and WebSphere Application Server. The default port for this communication is port 80 and is implied whenever a port is not specified in a URL. Therefore, the two sample URLs will result in the exact same request.

Common protocols: HTTPS

- HTTPS is not a separate protocol, but refers to the combination of a normal HTTP interaction over an encrypted secure socket layer (SSL) or transport layer security
- The default TCP port of an HTTPS Uniform Resource Identifier (URI) is **443**.
- Sample: **https://localhost/PlantsByWebSphere**



Data served using the HTTPS protocol should result in a secure/encrypted connection. The padlock at the bottom browser bar in Internet Explorer indicates that the page displayed is secure. If it isn't displayed, and the request was over HTTPS, this indicates that part of the content wasn't encrypted and isn't secure. This typically is seen when applications use frames. The original request was made using the HTTPS protocol, but the follow-on requests to populate the other frames of the page were submitted using HTTP. If you observe this within your application, you will need to consult with the developer to change those requests to use the proper protocol.

Browser return code: “200 OK”

- The request has succeeded. The information returned with the response is dependent on the method used in the request.
 - ▶ **GET**: An entity corresponding to the requested resource is sent in the response
 - ▶ **HEAD**: The entity-header fields corresponding to the requested resource are sent in the response without any message-body
 - ▶ **POST**: An entity describing or containing the result of the action is returned
 - ▶ **TRACE**: An entity containing the request message as received by the end server is returned

```
Response Header  
HTTP/1.0 200 OK  
Date: Fri, 31 Dec 1999 23:59:59 GMT  
Content-Type: text/html  
Content-Length: 1354
```

The response header is a record at the beginning of an HTTP response from the Web server. The response header field allows the server to pass additional information about the response that cannot be placed in the status line. The header fields give information about the server and about further access to the resources identified by the Requested URI. This data is normally not viewable from the browser response page. The 200 response code means that the request was handled properly.

Browser return code: “302 Object moved”

- The requested resource resides temporarily under a different URI.
 - ▶ Since the redirection might be altered on occasion, the client SHOULD continue to use the Request-URI for future requests.
 - ▶ The temporary URI SHOULD be given by the Location field in the response.

Response Header

HTTP/1.1 302 Object moved

Location: <http://www.mycompany.com/newpage.html>

A 302 response indicates that the page is no longer at the specified location and has been moved. This is automatically generated by the host which reroutes the browser to the new page. This will typically happen when the web server or application sever has a redirect to point the browser queuest to another URI.

Browser return code: “304 Not Modified”

- Indicates that the requested page has not been modified since it was last requested.
- If the client has performed a conditional GET request and access is allowed, but the document has not been modified, the server SHOULD respond with this status code.

Response Header

HTTP/1.1 **304 Not Modified**

Date: Fri, 31 Dec 1999 23:59:59 GMT

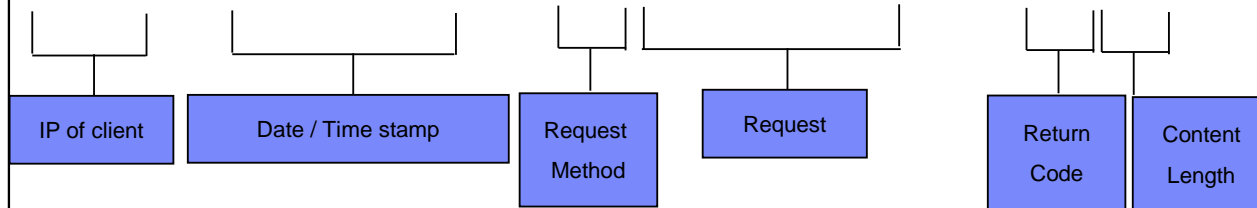
The 304 response code indicates that the requested page has not been modified since it was last requested. The page that is returned is pulled from the host system's cache. Caching is an effective mechanism for reducing overhead to the host system. The host cache's the content rather than having to locate or generate the content to be served. The host will continue to use the cache entry until it expires or is modified.

Identifying a working request: Web server access log

■ GET and POST requests

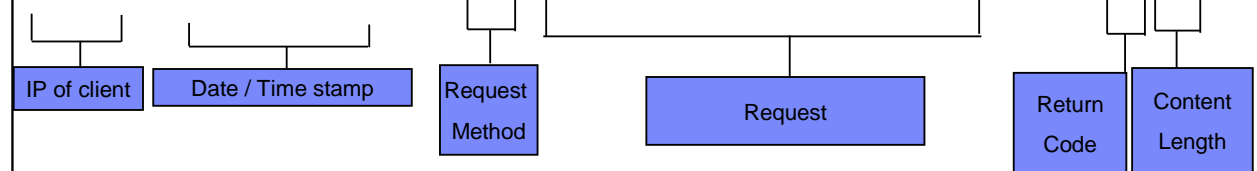
IHS Access Log for a Successful Get Request

127.0.0.1 -- [17/Apr/2006:13:07:00 -0400] "GET /PlantsByWebSphere/ HTTP/1.1" 200 1984



IHS Access Log for a Successful Post Request

127.0.0.1 -- [17/Apr/2006:13:26:10 -0400] "POST /PlantsByWebSphere/servlet/ShoppingServlet HTTP/1.1" 200 8307



All requests submitted to the IBM HTTP Server (IHS) are recorded in the Web server's access log at the time the request is completed. This is where you can verify the response to a specific requests. If a request fails to be processed on the Web server for any reason the response may not be posted to the access log. The example output on this slide conforms to the default IHS configuration format specified by the httpd.conf file. There are several key fields in an access log entry. Notice how both of these examples are from transactions with 200 response codes.

Identifying a working request: Plug-in log

■ GET request

[Mon Apr 17 13:07:00 2006] 00000d64 00000c0c - TRACE:

Date / Time stamp

Process / Thread ID

Message Type

1. [Mon Apr 17 13:07:00 2006] 00000d64 00000c0c - TRACE: ws_common: websphereShouldHandleRequest: trying to match a route for: vhost='localhost'; uri=/PlantsByWebSphere/
2. [Mon Apr 17 13:07:00 2006] 00000d64 00000c0c - TRACE: mod_was_ap20_http: as_translate_name: WebSphere will handle: /PlantsByWebSphere/
3. [Mon Apr 17 13:07:00 2006] 00000d64 00000c0c - TRACE: ws_common: websphereFindTransport: Setting the transport(case 2): App_Server_01 on port 9080
4. [Mon Apr 17 13:07:00 2006] 00000d64 00000c0c - TRACE: GET /PlantsByWebSphere/ HTTP/1.1
5. [Mon Apr 17 13:07:00 2006] 00000d64 00000c0c - TRACE: ws_common: websphereExecute: Wrote the request; reading the response
6. [Mon Apr 17 13:07:00 2006] 00000d64 00000c0c - TRACE: lib_htresponse: htresponseRead: Reading the response: 52965c
7. [Mon Apr 17 13:07:00 2006] 00000d64 00000c0c - TRACE: **HTTP/1.1 200 OK**
8. [Mon Apr 17 13:07:00 2006] 00000d64 00000c0c - TRACE: ws_common: websphereEndRequest: Ending the request

If a request is directed to a WebSphere Application Server then you can also trace the request as it passes through the IHS plugin. This information can be found in the plugin-log file. The examples on this slide are various log statements from a plugin-log file. They are not a complete request trace, but represent a few key entries to look for in order to validate that a request is received and preprocessed by the plug-in.

Identifying a working request: Plug-in log

■ POST request

[Mon Apr 17 15:43:18 2006] 00001128 00001130 - TRACE:

Date / Time stamp

Process / Thread ID

Message Type

[Mon Apr 17 15:43:18 2006] 00001128 00001130 - TRACE: ws_common: websphereFindServerGroup: trying to match a route for: vhost='localhost'; uri='/PlantsByWebSphere/servlet/ShoppingServlet'

[Mon Apr 17 15:43:18 2006] 00001128 00001130 - TRACE: mod_was_ap20_http: as_translate_name: WebSphere will handle: /PlantsByWebSphere/servlet/ShoppingServlet

[Mon Apr 17 15:43:18 2006] 00001128 00001130 - TRACE: ws_common: websphereFindTransport: Setting the transport(case 2): App_Server_01 on port 9080

[Mon Apr 17 15:43:18 2006] 00001128 00001130 - TRACE: POST /PlantsByWebSphere/servlet/ShoppingServlet HTTP/1.1

[Mon Apr 17 15:43:18 2006] 00001128 00001130 - TRACE: ws_common: websphereExecute: Wrote the request; reading the response

[Mon Apr 17 15:43:18 2006] 00001128 00001130 - TRACE: lib_htresponse: htresponseRead: Reading the response: 699964

[Mon Apr 17 15:43:18 2006] 00001128 00001130 - TRACE: **HTTP/1.1 200 OK**

[Mon Apr 17 15:43:18 2006] 00001128 00001130 - TRACE: ws_common: websphereEndRequest: Ending the request

These log entries are almost identical to the previous slide except that the previous slide was from GET requests and these are examples of POST requests. The log entries still follow the same general format as before.

Topic summary

Having completed this topic, you should be able to:

- Describe the typical request flow between a client, a Web server, and the WebSphere Application Server
- Read and interpret the necessary logs to identify working requests

By now you should be able to describe the typical request flow between a client, Web server, and the WebSphere Application Server as well as read and interpret the pertinent log files.

Review of non-working request

After completing this topic, you should be able to:

- Identify preliminary questions to ask at the start of problem determination
- Determine what trace data is needed and how to collect it
- Review trace data to identify key failure points
- Make a rational choice to correct the problem

The rest of this presentation focuses on how to troubleshoot non-working requests. This includes the initial steps, activating trace, and reviewing the trace to determine what steps are necessary to resolve the problem.

Preliminary questions (1 of 2)

1. Are all components running?
 - ▶ Verify that IHS and the WebSphere application are running.
2. DNS is working correctly?
 - ▶ Verify that the Web server hostname and the IP address of the box are set correctly within DNS.
3. Are there any firewall or front-end issues?
 - ▶ Know your topology and what components lie between the client and the Web server. One of these could be keeping the request from going through.
4. What are the Operating Systems (OS) of all components involved?
 - ▶ Sometimes updates to components are published for your particular OS. Also, you may have an OS related patch that is required to correct a problem.
5. What are the versions and maintenance levels of the involved products?
 - ▶ Important for researching possible updates to the Web server, plug-in and application server.

There are several questions that you should ask yourself whenever you begin troubleshooting a non-working request. These questions will help you focus your efforts on the components that are likely causing the problem.

Preliminary questions (2 of 2)

6. What did you type in?

- ▶ Need to know the exact request that was typed into the client browser.

7. What did you get back?

- ▶ Get a print screen of the results. Problems can result with the wrong response or no response.

8. What was the date and time of the request?

- ▶ This aids in locating the problematic request within the logs.

9. Who was supposed to respond to the request?

- ▶ Web server or application server



Here are the rest of the example questions that will help you track down the cause of the request flow problem. These questions cannot cover all the components since that would require more knowledge of your environment, but they provide an excellent starting position to focus your problem determination efforts.

Determining what trace data is needed (1 of 3)

1. Stop the IBM HTTP Server and WebSphere Application Server.
2. Clear all logs in the IBM HTTP Server directory:
 - ▶ <install_root>/log
3. Clear all logs in the WebSphere Application Server directory:
 - ▶ <install_root>/log
4. Edit the plugin-cfg.xml file and change Loglevel to Trace. For example:
 - ▶ <Log LogLevel="Trace" Name="c:\WebSphere\AppServer\logs\native.log"/>
5. Edit the httpd.conf file and change Loglevel to **debug**

The IHS Web server requires a restart for any changes to the *httpd.conf* file to take effect. For other Web server types, please consult with your vendor for the specifics on how to accomplish the same logging.

Any time a Web server restart is required for diagnostic purposes, you should archive the existing logs to have new logs created. Doing this will reduce the amount of data you will need to review to diagnose a problem request.

Most IHS configurations have one access log and error log. Scan the *httpd.conf* file for any of the following directives: *ErrorLog*, *CustomLog*, *RewriteLog* and make a note of each location. Various other logs can be generated depending on the environment's needs. Some may set up <VirtualHost> stanzas for SSL and have specific logs generated to trap that area's specific traffic.

The Plug-in can have its *LogLevel* changed without having to stop and restart the Web server. This change will take effect after the plug-in's *Refreshinterval* has expired and a new request has been submitted to the system.

Determine what trace data is needed (2 of 3)

6. Enable WebSphere tracing:

- ▶ Log on to the administrative console,
- ▶ In left panel, select **Troubleshooting -> Logs and Trace**.
- ▶ In right panel, select **Logging and Tracing -> your_application_server -> Diagnostic Trace**.
- ▶ In Configuration tab perform the following tasks:
 - Check **Enable Log** property.
 - In the Trace Output, choose File Name and type a file name. Increase the Maximum file size to 100 MB. Increase Maximum number of historical files to 10 as well.
 - Select **Basic**

Configuration Runtime

General Properties

Enable Log

Trace Output

Memory Buffer

* Maximum Buffer Size
8 thousand entries

File

* Maximum File Size
100 MB

* Maximum Number of Historical Files
10

* File Name
\${SERVER_LOG_ROOT}/my_trace.log

Additional Properties

[Change Log Detail Levels](#)

Trace Output Format
Basic (Compatible)

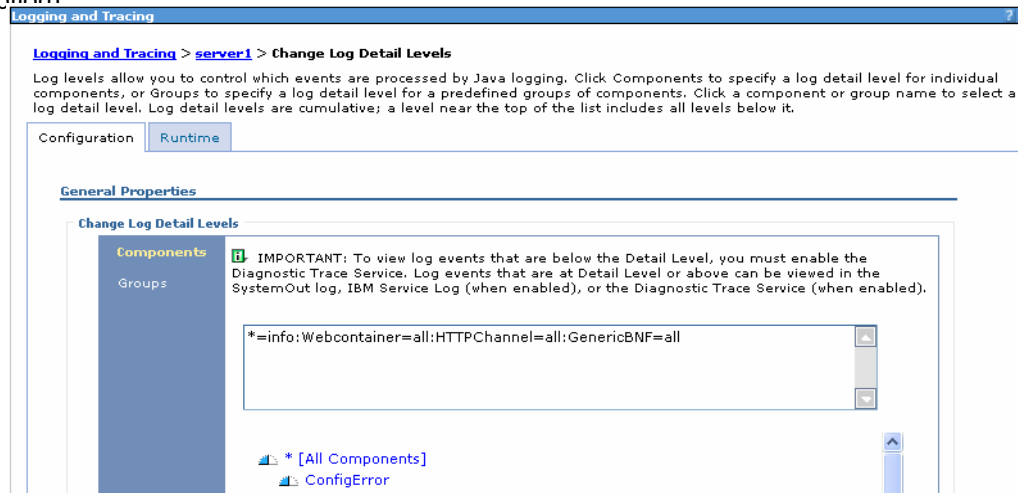
Apply OK Reset Cancel

The information in this slide provides the steps to enable an application server *Webcontainer* trace. This information is posted within an IBM *Mustgather* that can be found on the ibm support site by search for the phrase *Mustgather*.

Determine what trace data is needed (3 of 3)

6. Enable WebSphere tracing (continued):

- ▶ Navigate back to **Logging and Tracing** -> **your_application_server** -> **Change Log Detail Levels**
- ▶ In the Configuration tab **Trace Specification**, enter the following line:
 - `*=info:Webcontainer=all:HTTPChannel=all:GenericBNF=all`
- ▶ Click **Apply**, then **OK** and then **Save** your configuration (select Synchronize changes with Nodes option)



7. Restart the IBM HTTP Server and WebSphere Application Server.
8. Recreate the failure.

This slide provides the steps to enable an application server *Webcontainer* trace. This information can also be found by looking at the Mustgather information on the IBM support website.

Unit summary

Having completed this unit, you should be able to:

- Identify the difference between working and non-working requests
- Produce the necessary logging to research non-working requests
- Identify the root cause of a non-working request

By this point you have the ability to identify a non-working request and produce the necessary trace and log information so that you can identify the root cause of the non-working request.

Feedback

Your feedback is valuable

You can help improve the quality of IBM Education Assistant content to better meet your needs by providing feedback.

- Did you find this module useful?
- Did it help you solve a problem or answer a question?
- Do you have suggestions for improvements?

Click to send e-mail feedback:

<mailto:iea@us.ibm.com?subject= Feedback about SW5706G13 RequestFlowProbs.ppt>



You can help improve the quality of IBM Education Assistant content by providing feedback.

Trademarks, copyrights, and disclaimers

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

IBM WebSphere

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

Product data has been reviewed for accuracy as of the date of initial publication. Product data is subject to change without notice. This document could include technical inaccuracies or typographical errors. IBM may make improvements or changes in the products or programs described herein at any time without notice.

Information is provided "AS IS" without warranty of any kind. THE INFORMATION PROVIDED IN THIS DOCUMENT IS DISTRIBUTED "AS IS" WITHOUT ANY WARRANTY, EITHER EXPRESS OR IMPLIED. IBM EXPRESSLY DISCLAIMS ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT. IBM shall have no responsibility to update this information. IBM products are warranted, if at all, according to the terms and conditions of the agreements (for example, IBM Customer Agreement, Statement of Limited Warranty, International Program License Agreement, etc.) under which they are provided. Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products in connection with this publication and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products.

IBM makes no representations or warranties, express or implied, regarding non-IBM products and services.

The provision of the information contained herein is not intended to, and does not, grant any right or license under any IBM patents or copyrights. Inquiries regarding patent or copyright licenses should be made, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

© Copyright International Business Machines Corporation 2007. All rights reserved.

Note to U.S. Government Users - Documentation related to restricted rights-Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract and IBM Corp.