IBM WebSphere Business Connection

Web Services Troubleshooting

Version 1.1.0

Note!
Before using this information and the product it supports, be sure to read the general information under "Notices" on page 9.

First Edition (September 2002)

This edition applies to Version 1, Release 1, Modification 0, of *IBM® WebSphere® Business Connection* (5724-D26) and to all subsequent releases and modifications until otherwise indicated in new editions.

IBM welcomes your comments. You can send them to the following address:

IBM Canada Ltd. Laboratory
Information Development
8200 Warden Avenue
Markham, Ontario, Canada L6G 1C7

Include the title and order number of this book, and the page number or topic related to your comment.

When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

Contents

Troubleshooting

This document describes various troubleshooting techniques you can use when you are developing and deploying Web services.

Using TcpMon

Complex messaging systems can be difficult to troubleshoot because of the many machines and software applications involved. Determining where the problem occurred can be a challenge, especially if the error information provided by a component is not passed reliably from the point of failure to the originator of the message.

A message monitor, therefore, can be a useful tool for troubleshooting message systems. The monitor allows you to intercept messages as they pass through a transport from one component or machine to another. While in transit, messages are said to be in their ***wire format.*** The wire format in the case of Business Connection is typically XML.

When an error occurs at the service end, it may or may not be reliably returned to the caller as a SOAP fault message. Generally when a service collaboration has a failure, a SOAP fault message is provided as the response. The various subsystems between the caller and the service (the CrossWorlds(R) ICS, Web Services Gateway, or WSIF, for example) are usually able to reliably return a SOAP fault to the caller.

Other errors are not handled as well by all of the subsystems. Sometimes an error message in transit back to the caller is not handled correctly by one of the subsystems, and it throws an exception. The caller might see the exception from the subsystem instead of the real error. The message monitor, however, can intercept the real error message so that you can see it in the monitor.

One such message monitor that is useful for SOAP messages is called **TcpMon**, which is part of the Apache AXIS project. TcpMon allows you to hook into the Http transport at the port level. For example, when you start TcpMon, you tell it which port number to listen on and where to forward all messages that arrive on that port. Generally for debugging, it is configured to listen on port 4040 and to forward to port 80. To make this useful, the caller is configured to send to port 4040 instead of the usual default Http port 80. In this way, the wire between the caller and the server can be monitored.

Example: Catching an error message

Suppose you want to send CrossWorlds-to-CrossWorlds using the BCT_SampleData scenario developed in the tutorial. In the URL configuration file, you normally enter the following line that sends to default port 80:

```
BCT_WS_BCT_SampleData_Create_URL=http://localhost/bctwssamplesweb
  /servlet/messagerouter
```

Now suppose that the application server for this URL was not started. Sending a message would fail in this case. The error that you see in the SOAP Connector agent window tells you this:

```
[Time: 2002/06/15 10:45:46.806] [System: Server] [Thread: VBJ ThreadPool Worker
(#6651535)] [Mesg: [Time: 2002/06/15 10:45:46.806] [System: ConnectorAgent] [SS:
 BCTSampleSOAPConnector] [Thread: VBJ ThreadPool Worker (#6651535)] [Type: Trace
] [Mesg:  :[9543] DoVerbFor cookie=CxCommon.DeliveryItem@330bfSat Jun 15 10:16:08
 EDT 2002 rc=-1 rtnObj.status=4 rtnObj.message=An error occurred while reading
data from URL [http://localhost:4040/bctwssamplesweb/servlet/messagerouter]
Exception thrown is [java.io.IOException: Cannot parse Web Service response: org
.xml.sax.SAXParseException: White space is required between the public identifier
and the system identifier.]] ]
```

Your first response to this information is probably to debug a SAXParseException. You will not get far with this approach, because the problem is that the DataHandler had this exception trying to parse the real error message, which is related to the fact that the application server is not started.

To see the real error in this case, you can change the URL to send to port 4040 and use TcpMon to monitor traffic between ports 4040 and 80.

1. First change the URL:

   ```
   BCT_WS_BCT_SampleData_Create_URL=http://localhost:4040

   /bctwssamplesweb/servlet/messagerouter
   ```

2. Start TcpMon. (You need access to axis.jar. This is available from http://www.apache.org).
3. Write a batch script similar to the following and run it to start TcpMon:

   ```
   @start d:\websphere\appserver\java\bin\java -classpath c:\utility\axis.jar
     org.apache.axis.utils.tcpmon 4040 localhost 80
   ```

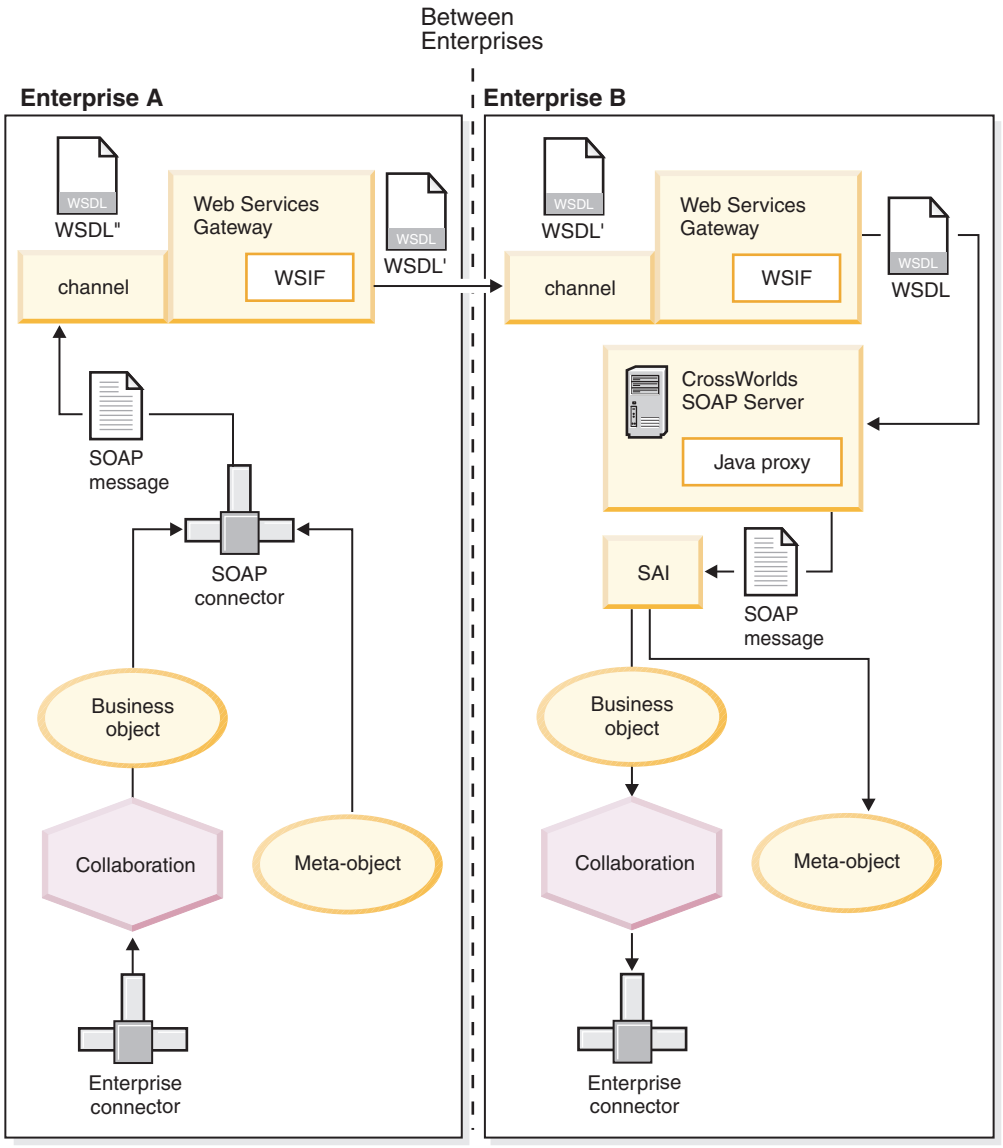
You can see that the Web Server returned an HTTP 500 error, along with HTML intended for a web browser. *This* is the error to debug. The fix is to start the application server.

The reason the SOAP agent window displayed a SAXParseException is because the CrossWorlds data handler could not parse the response message, which has an improper MIME type (text/html) for the data handler as configured in this case.

This example shows why it can be useful to view the actual messages on the wire. You can use this technique to watch messages flowing between two Web Services Gateways, between the server Web Services Gateway and the SOAP-enabled application server where the proxy resides, or between the client Web Services Gateway and the sending SOAP connector.

Monitoring Web Services Gateway-to-SOAP service messages

Refer to the figure below:



The closest point to a real service error that a message can be caught is at the line labeled **wsdl** that points to the **CrossWorlds Soap Server**. To catch this, TcpMon must be running on the same machine as the **CrossWorlds Soap Server**. The service section of the wsdl that is deployed in Web Services Gateway B must be modified before deployment to send to port 4040 rather than the default port 80, as shown below:

```
<service name="BCT_SampleData_Create">
<port name="BCT_SampleData_CreatePort"
binding="tns:BCT_SampleData_CreateBinding">
<soap:address
location="http://cwhostname:4040/crossworlds/servlet/messagerouter"/>
       </port>
</service>
```

Deployment of this wsdl causes messages to be sent from the Web Services Gateway B to port 4040 on the machine hosting the CrossWorlds Soap Server. With TcpMon configured to listen on port 4040, you can see the message traffic to and from the CrossWorlds Soap Server.

Monitoring Web Services Gateway-to-Web Services Gateway messages

This time, the line in the figure above between Web Services Gateway A and Web Services Gateway B is to be monitored.

Recall that wsdl' is the external gateway service form of the target wsdl deployed in Web Services Gateway B. To monitor messages between Web Services Gateways, you need to edit the service section of wsdl'. This requires you to have a physical file representation of wsdl' so you can edit it before deploying in Web Services Gateway A.

See Web Services Overview and Samples for information on how to obtain the two files that represent wsdl'. After obtaining these files and changing the import statement as required, change the service section in the service implementation file to use port 4040.

To catch these messages, TcpMon must be running on the same machine as Web Services Gateway B.

Monitoring SOAP connector-to-Web Services Gateway messages

This time, the line in the figure above between the SOAP Connector and the outbound channel of Web Services Gateway A is to be monitored.

The SOAP connector must be given the URL of the outbound channel of Web Services Gateway A in order to send messages to this channel. In the sample and the tutorial, this URL is contained in file bct_ws_configuration.txt. A map looks up the URL using a key hashed from the generic business-object type and the verb. Your code probably uses a different way to obtain the URL. However it is done, this URL must be modified to use port 4040 if you want to monitor these messages on the wire.

To catch these messages, TcpMon must be running on the same machine as Web Services Gateway A.

Parsing Errors

Another situation that can arise is when an error response comes back in a format that is not handled cleanly. For example, the CrossWorlds SOAP data handler expects the response to be an XML message. Normal successful responses are XML SOAP messages. Fault responses from the service that is called are provided as soap:fault messages in the body of the XML SOAP response. The SOAP data handler can parse and handle these without a problem.

The problem for the CrossWorlds SOAP data handler occurs when the response that it receives is not an XML message. This can occur if some other part of the message transport infrastructure fails. For example, if the Application Server that hosts the Web service proxy for a CrossWorlds Web service is not started and a request is sent to it, its response is an HTML page with a Http 500 error. This HTML cannot be parsed by the SOAP data handler, so you will see in the CrossWorlds SOAP agent window a parsing error. The real error is masked by this parse error.

To find the real error in this case, turn tracing to the highest level (5) for the SOAP connector agent. When you do this, the request SOAP message and the response to

it are recorded in the agent command window. The response is listed immediately after the request. The request is displayed as a nicely formatted XML string. You can identify it by watching the window just as a message is sent. The window will display the request XML and pause briefly before the response comes back. When this occurs and you see a parse error, you should scroll up in the window to read the real response, which presumably was not parsed by the CrossWorlds data handler. In this way you can get some insight to what actually went wrong.

Log and trace options

Tracing options are available to help trace the flow of messages through the gateway and the CrossWorlds application server where the java proxy class is deployed.

Using CrossWorlds log4j tracing

The CrossWorlds Java Proxy provides tracing that you can use to see when the proxy is called by the SOAP messagerouter servlet, when the proxy calls the SAI, and when the response has returned from the CrossWorlds collaboration.

The CrossWorlds tracing uses the Apache Log4J package. A properties file provided with the **bctwssamples.ear** is configured to trace events that occur for the BCT_TestAllTypes_Retrieve proxy class. The configuration writes trace messages to the "console." For the proxy running in WebSphere(R), the console is WebSphere stdout.

Information about how to use the Log4J facility is contained in the prologue of the properties file. Read this to understand the options available for tracing the proxy activity.

The following is an example of the type of tracing that is produced with the default log4j.properties file. Each message sent and responded to by the CrossWorlds inbound collaboration produces this tracing to the stdout file:

```
Bye[6/7/02 9:47:44:375 EDT] 7b7b0d31 SystemOut     U [2002-06-07 09:47:44,295] INFO
33719[Servlet.Engine.Transports:25](ProxyClassParam.java:220) - Loading parameters
from config-file: BCT_TestAllTypes_Retrieve.cfg

[6/7/02 9:47:44:455 EDT] 7b7b0d31 SystemOut     U [2002-06-07 09:47:44,375] INFO
33799[Servlet.Engine.Transports:25](WebServicesAccessClient.java:129) - Initializing
Session with ICS using parameters:cwldVersion= 4.x icsName= SSHCrossWorlds userName=
admin passWord= null collabName=SAI_to_BCTSampleConnector2_BCT_TestAllTypesInbound
collabPort= From mimeType=xml/soap boVerb=Retrieve iorFileName=
D:\CrossWorlds\SSHCrossWorldsInterchangeServer.ior

[6/7/02 9:47:46:698 EDT] 7b7b0d31 SystemOut     U [2002-06-07 09:47:46,608] INFO
36032[Servlet.Engine.Transports:25](WebServicesAccessClient.java:221) - Entering
executingCollab. CollabName: SAI_to_BCTSampleConnector2_BCT_TestAllTypesInbound

[6/7/02 9:47:46:798 EDT] 7b7b0d31 SystemOut     U [2002-06-07 09:47:46,708] INFO
36132[Servlet.Engine.Transports:25](WebServicesAccessClient.java:230) - Executing
collaboration with parameters: cwldVersion= 4.x icsName= SSHCrossWorlds userName=
admin passWord= null collabName =SAI_to_BCTSampleConnector2_BCT_TestAllTypesInbound
collabPort= From mimeType= xml/soap boVerb=Retrieve iorFileName=
D:\CrossWorlds\SSHCrossWorldsInterchangeServer.ior

[6/7/02 9:47:59:717 EDT] 7b7b0d31 SystemOut     U [2002-06-07 09:47:59,627] INFO
49051[Servlet.Engine.Transports:25](WebServicesAccessClient.java:199) - Closing
Access Session. CollabName: SAI_to_BCTSampleConnector2_BCT_TestAllTypesInbound
```

You can see that each line provides information about an event that has occurred in the proxy class. The specific information used by the proxy to connect to the ICS

and to call a collaboration is provided. This can be helpful when debugging connectivity problems between the WebSphere machine and the CrossWorlds machine.

Refer to CrossWorlds Web Services Documentation and the Apache Log4J documentation (http://jakarta.apache.org/log4j) for more information about the use of the Log4J tracing with the CrossWorlds java proxy.

Enabling WebSphere trace for Business Connection components

This section describes how you can enable WebSphere trace for the following components:

- Routing Filter
- Authentication Filter
- Message Warehouse
- Exception Handler

To debug problems with the Web Services Gateway handling of messages, there are tracing hooks written in the code for these components. These components use the built-in tracing facility of WebSphere. There are entry, exit, and information trace messages provided in each component.

To configure WebSphere tracing for these components, follow these steps:

1. Make sure the EAR file for each is installed in the Web Services Gateway App Server. Note that the Web Services Gateway Core ear file (wsgw.ear) must always be installed before any other Web Services Gateway-related EAR files.
2. With the filter and support EAR files installed, start the Web Services Gateway App Server.
3. Click the **Web Services Gateway App Server** in the left-hand pane of the WebSphere Application Server Admin Console, and then click the **Services** tab on the right-side pane.
4. Select **Trace Service** in the list box, and click **Edit Properties.**
5. When the Trace Service dialog box is displayed, click **Standard Output**and then click the small button to the right of the Trace Specification field.
6. When the Trace dialog is displayed, expand **Groups** and you will see a group named **WebSphere Business Connection Trace Loggers**.
7. Expand this group to see four classes whose names indicate their function.
8. Right-click the group name and select **All**.
9. Click **OK**.
10. Click **OK** again. Then be sure to click the **Apply** button on the main Admin Console window; otherwise the changes are lost.

    This will cause all trace messages that are in the code for these four support services to be written to the application server stdout file. Normally there are trace messages for method entry, method exit, and when something interesting is done by a method. Inspection of the stdout file will allow you to trace execution for each function.

Before tracing is activated, you must stop and restart the application server.

The following is the example of the output obtained when tracing is enabled for these four classes. You can see method entry and exit as well as information about message processing.

```
[6/14/02 14:48:17:949 EDT] 55a94e57 BCTWSMessageW > class
com.ibm.wsgw.beans.BCTWSMessageWarehouseBean logRequest Servlet.Engine.
Transports:10
Entry
                                ApacheSOAPChannel2
                                127.0.0.1
                                BCT_TestAllTypes_Retrieve
                                m_BCT_TestAllTypes
                                2002-06-14 14:48:17.278
                                [WSIFRequest:
    serviceID = 'urn:ibmwsgw:BCT_TestAllTypes_Retrieve'
    operationName = 'm_BCT_TestAllTypes'
    incomingMessage = 'org.apache.wsif.util.WSIFDefaultMessage@2e034e4f'
    contextMessage = 'org.apache.wsif.util.WSIFDefaultMessage@59b30e4f']
                                [Ljava.io.Serializable;@20a9ce4e
[6/14/02 14:48:17:959 EDT] 55a94e57 BCTWSMessageW > class
com.ibm.wsgw.beans.BCTWSMessageWarehouseBean sendToAuditLog
Servlet.Engine.Transports:10 Entry
                                1000
                                [WSIFRequest:
    serviceID = 'urn:ibmwsgw:BCT_TestAllTypes_Retrieve'
    operationName = 'm_BCT_TestAllTypes'
    incomingMessage = 'org.apache.wsif.util.WSIFDefaultMessage@2e034e4f'
    contextMessage = 'org.apache.wsif.util.WSIFDefaultMessage@59b30e4f']
                                ApacheSOAPChannel2
                                127.0.0.1
                                BCT_TestAllTypes_Retrieve
                                m_BCT_TestAllTypes
                                2002-06-14 14:48:17.278
[6/14/02 14:48:18:009 EDT] 55a94e57 SystemOut     U Executing
BCMLog:writeAudit::BCMObj.sendMsg(logMsgID:<5174739229438d86:35dc8e54:
ee6ff44a60:-7ffe>
applicationID:<BCT_WS_WSGW>
MsgType:<WAS>
MsgHeaderVersion:<BCM1>
HostNameIP:<stevesh/9.163.15.232>
BodySize:<248>
BodyData:<[WSIFRequest:
    serviceID = 'urn:ibmwsgw:BCT_TestAllTypes_Retrieve'
    operationName = 'm_BCT_TestAllTypes'
    incomingMessage = 'org.apache.wsif.util.WSIFDefaultMessage@2e034e4f'
    contextMessage = 'org.apache.wsif.util.WSIFDefaultMessage@59b30e4f']>
LogMessaeType:<SMAuditLog>
ProcessCorrelationID:<5174739229438d86:35dc8e54:ee6ff44a60:-7fff>
TransportCorrelationID:<null>
TransportMessageID:<null>
UserSecurityArea:<null>
InternalID:<null>
ExternalID:<127.0.0.1>
RelatedSubjectID:<null>
SenderInstanceID:<null>
ProcessingCategory:<null>
TransactionRequired:<false>
LogPriority:<High>
BodyCategory:<BCT_TestAllTypes_Retrieve>
BodyType:<m_BCT_TestAllTypes>
EventType:<1000>
ProcessInstanceID:<null>
InternalTradingPartnerID:<0>
UserArea:<null>
PublicationTopic:<null>
SessionID:<null>
```

```
ModelID:<DEFAULT>
TimeStampCreated:<1024080497989:-18000000>
TimeStampExpires:<-1>
Super:<com.ibm.wbi.bcm.sm.message.BCMMessage@1e6c30a>
,JMS,BCTSAMPLESQUEUEMANAGER,BCTAUDITLOG,file:/c:/bctws/jms/context);
[6/14/02 14:48:18:690 EDT]
55a94e57 BCTWSMessageW X class
com.ibm.wsgw.beans.BCTWSMessageWarehouseBean sendToAuditLog
Servlet.Engine.Transports:10 BCT_WS_TRACE00001: Received exception
<com.ibm.wbi.bcm.common.BCMException> Message information: <EpicException: caught
MissingResourceException getting message from ResourceBundle with information
-- class name <java.util.PropertyResourceBundle> key <4245> message <Can't find
resource for bundle java.util.PropertyResourceBundle, key 4245>
Resource name being used <com.ibm.wbi.bcm.common.BCT_SMExceptionMessages>
Original Inputs
MessageID<4245>Object[]<<4245><BCMLog::writeAudit(String,
String)><com.ibm.wbi.bcm.common.BCMException><EpicException: caught
MissingResourceException getting message from ResourceBundle with information
-- class name <java.util.PropertyResourceBundle> key <BCT_SM0116> message <Can't
find resource for bundle java.util.PropertyResourceBundle, key BCT_SM0116>
Resource name being used <com.ibm.wbi.bcm.common.BCT_SMExceptionMessages>
Original Inputs
MessageID<BCT_SM0116>Object[]<<BCT_SM0116><com.ibm.wbi.bcm.sm.bccm.LMSJMS::
getQueueConnection(MQAOAddress)><MQJMS2005: failed to create MQQueueManager
for 'BCTSAMPLESQUEUEMANAGER'>>><BCMLog writeAudit(String, String) Failed>>>
Additional information <>.
[6/14/02 14:48:18:730 EDT] 55a94e57 BCTWSMessageW > class
com.ibm.wsgw.beans.BCTWSMessageWarehouseBean sendToAuditLog
Servlet.Engine.Transports:10 Exit

[6/14/02 14:48:18:730 EDT] 55a94e57 BCTWSMessageW > class
com.ibm.wsgw.beans.BCTWSMessageWarehouseBean logRequest Servlet.Engine.
Transports:10
Exit
                                Message passed to Audit Log
[6/14/02 14:48:20:833 EDT] 55a94e57 BCTWSAuthenti >
com.ibm.bct.ws.filter.ejb.BCTWSAuthenticationFilterBean filterRequest
Servlet.Engine.Transports:10 Entry
                                [WSIFRequest:
    serviceID = 'urn:ibmwsgw:BCT_TestAllTypes_Retrieve'
    operationName = 'm_BCT_TestAllTypes'
    incomingMessage = 'org.apache.wsif.util.WSIFDefaultMessage@22540e4a'
    contextMessage = 'org.apache.wsif.util.WSIFDefaultMessage@45630e4b']
                                [WSIFResponse:
    serviceID = 'urn:ibmwsgw:BCT_TestAllTypes_Retrieve'
    operationName = 'm_BCT_TestAllTypes'
    isFault = 'false'     outgoingMessage = 'null'
    faultMessage = 'null'
    contextMessage =  'null']
[6/14/02 14:48:20:904 EDT]  55a94e57 BCTWSAuthenti >
com.ibm.bct.ws.filter.ejb.BCTWSAuthenticationFilterBean filterRequest
Servlet.Engine.Transports:10 Found destinationType: <PartnerName> and
destinationID:<stevesh> in SOAP Header
[6/14/02 14:48:20:914 EDT] 55a94e57 SystemOut     U Properties file not found,
using CMS values
[6/14/02 14:48:20:914 EDT] 55a94e57 SystemOut     U User id= uid:stevesh:
PartnerName
[6/14/02 14:48:20:914 EDT] 55a94e57 BCTWSAuthenti >
com.ibm.bct.ws.filter.ejb.BCTWSAuthenticationFilterBean filterRequest
Servlet.Engine.Transports:10 Exit
      [WSIFRequest:
    serviceID = 'urn:ibmwsgw:BCT_TestAllTypes_Retrieve'
    operationName = 'm_BCT_TestAllTypes'
    incomingMessage = 'org.apache.wsif.util.WSIFDefaultMessage@22540e4a'
    contextMessage = 'org.apache.wsif.util.WSIFDefaultMessage@45630e4b']
                                com.ibm.wsgw.beans.FilterAction@7063ce49
```

Notices

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

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

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

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

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

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

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

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

WebSphere Business Connection Lab Director
IBM RTP Laboratory
3039 Cornwallis Road
P.O. BOX 12195

Raleigh, NC 27709-2195
U.S.A

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurement may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

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

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

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

COPYRIGHT LICENSE:

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

Programming interface information

Programming interface information is intended to help you create application software using this program.

General-use programming interfaces allow you to write application software that obtain the services of this program's tools.

However, this information may also contain diagnosis, modification, and tuning information. Diagnosis, modification and tuning information is provided to help you debug your application software.

Warning: Do not use this diagnosis, modification, and tuning information as a programming interface because it is subject to change.

Trademarks and service marks

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

IBM
alphaWorks
AIX
CrossWorlds
DB2
DB2 OLAP Server
DB2 Universal Database
DeveloperWorks
MQSeries
SecureWay
WebSphere

Lotus is a trademark of International Business Machines Corporation and Lotus Development Corporation in the United States, other countries, or both.

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

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

MMX, Pentium and ProShare are trademarks of Intel Corporation in the United States, other countries, or both.

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