WebSphere® Application Server for z/OS Version, 6.0.1

Troubleshooting and support

GA22-7964-03

Note
Before using this information, be sure to read the general information under "Notices" on page 397.

Compilation date: March 11, 2005

Contents

How to send your comments

Your feedback is important in helping to provide the most accurate and highest quality information.

- To send comments on articles in the WebSphere Application Server Information Center
  1. Display the article in your Web browser and scroll to the end of the article.
  2. Click on the **Feedback** link at the bottom of the article, and a separate window containing an e-mail form appears.
  3. Fill out the e-mail form as instructed, and click on **Submit feedback** .
- To send comments on PDF books, you can e-mail your comments to: **wasdoc@us.ibm.com** or fax them to 919-254-0206.

  Be sure to include the document name and number, the WebSphere Application Server version you are using, and, if applicable, the specific page, table, or figure number on which you are commenting.

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.

Chapter 1. Overview and new features for troubleshooting

"What is new for troubleshooters" on page 3

This topic provides an overview of new and changed features in troubleshooting tools and support.

Chapter 5, "Diagnosing problems (using diagnosis tools)," on page 131

This topic provides a place to start your search for troubleshooting information.

Presentations from IBM Education Assistant

- Where to start with problem determination
- Serviceability enhancements

Troubleshooting overview

Troubleshooting is the process of finding and eliminating the cause of a problem. Whenever you have a problem with your IBM software, the troubleshooting process begins as soon as you ask yourself what happened? A basic troubleshooting strategy at a high level involves:

1. Recording the symptoms.
2. Recreating the problem.
3. Eliminating possible causes.
4. Using diagnostic tools.

Recording the symptoms of the problem

Depending on the type of problem you have, whether it be with your application, your server, or your tools, you might receive a message that indicates something is wrong. Always record the error message that you see. As simple as this sounds, error messages sometimes contain codes that might make more sense as you investigate your problem further. You might also receive multiple error messages that look similar but have subtle differences. By recording the details of each one, you can learn more about where your problem exists.

Recreating the problem

Think back to what steps you were doing that led you to this problem. Try those steps again to see if you can easily recreate this problem. If you have a consistently repeatable test case, you will have an easier time determining what solutions are necessary.

- How did you first notice the problem?
- Did you do anything different that made you notice the problem?
- Is the process that is causing the problem a new procedure, or has it worked successfully before?
- If this worked before what has changed? The change can refer to any type of change made to the system, ranging from adding new hardware or software, to configuration changes to existing software.

- What was the first symptom of the problem you witnessed? Were there other symptoms occurring around that point of time?
- Does the same problem occur elsewhere? Is only one machine experiencing the problem or are multiple machines experiencing the same problem?
- What messages are being generated that could indicate what the problem is?

Eliminating possible causes

Narrow the scope of your problem by eliminating components that are not causing the problem. By using a process of elimination, you can simplify your problem and avoid wasting time in areas that are not culprits. Consult the information in this product and other available resources to help you with your elimination process.

- Has anyone else experienced this problem? See the topic on searching knowledge bases.
- Is there a fix you can download? See the topic on getting fixes.
- You can use the WebSphere Application Server Troubleshooting Guide to work through the cause of a problem

Using diagnostic tools

As a more advanced task, there are various tools that you can use to analyze and diagnose problems with your system. To learn how to use these tools see Chapter 5, "Diagnosing problems (using diagnosis tools)," on page 131.

Contents of this section: Troubleshooting

Related concepts

Product overview and quick start
This topic provides shortcuts to information for obtaining a high level understanding of the product, then getting started quickly. Links include various conceptual overviews, descriptions of what is new, and how the documentation is organized. ″How do I?″ links lead to task descriptions, including ShowMe demonstrations, tutorials, and presentations when they are available.

Related tasks

Chapter 2, “How do I troubleshoot?,” on page 53

What is new for troubleshooters

This topic highlights what is new or changed in Version 6.0.x for users who are going to use the troubleshooting tools and serviceability features.

The biggest change in troubleshooting support and serviceability is increased ability to automatically detect and recover from problems. Many serviceability improvements are described in the IBM Education Assistant presentation:Serviceability enhancements.

The troubleshooting section has been revamped and expanded to include extensive support information, including the ability to search live, Web-based support resources by using the customized query fields in the ″Web search″ page.

New troubleshooting technology on the Support site

- MustGather: Read first for all WebSphere Application Server products
- Mustgather: Read first for WebSphere Application Server for z/OS
- Troubleshooting Guide for WebSphere Application Server

Improved message text, new message prefixes

Messages for key product components used during installation, migration, and initial configuration have been improved. Additional components have messages now. Message prefixes have changed. The message reference provides a mapping of Version 5.1.x prefixes to Version 6.0.x prefixes.

Java logging framework from JSR47 is exploited

In J2SE 1.4, the Java logging framework was introduced via JSR47. In WebSphere Application Server, messages and trace logged to both JRAS and JSR47 logging APIs are passed into the JSR47 logging infrastructure. This allows JSR47 Handlers connected to the root JSR47 Logger to receive all WebSphere Application Server log content. JSR47 and JRAS Logger levels can be controlled via the admin console troubleshooting section. WebSphere Application Server also builds its logs from the JSR47 framework by connecting its Handlers to the root Logger.

The JSR47 Logging infrastructure allows for flexible pluggability of custom Handlers into the logging infrastructure to enable custom logs. By appropriate configuration, the Handlers can receive WebSphere Application Server's logged events, and events logged to Loggers instantiated by your applications.

See "Logging and tracing with Java logging" on page 61.

JRAS is deprecated

The JRAS API is deprecated. Users are directed to use the JSR47 logging infrastructure instead. See Deprecated and removed features for more information about this and other deprecated items.

Common Base Events describe system situations

Common Base Events are data structures used to describe situations that occur in the system. Common Base Events are used for various purposes, including representing things such as business events, configuration events, error events, and so on. The WebSphere Application Server now uses Common Base Events as the internal representation of logged messages.

Common Base Events are logged via JSR47 and as such can be received and operated on from JSR47 Handlers. Handlers which are not programmed to the Common Base Event specification will also be able to consume these events as CommonBaseEventLogRecords. Handlers which are programmed to the Common Base Event specification can take advantage of fields within the Common Base Events.

See "The Common Base Event in WebSphere Application Server" on page 83.

Thread names can be included in logs	Thread name has been added to the Advanced log format and Log Analyzer format. The advanced log format is available as an output format for the trace log and system out log. The thread name is now included in this format to enable easier correlation with other types of diagnostic data. The log analyzer format is available as an output format for the trace log. The thread name is now included in this format to enable easier correlation with other types of diagnostic data.

Troubleshooting by component

Select a component from the list.

Related reference

"Installation component troubleshooting tips" on page 233

Administration and administrative console troubleshooting tips

Web module or application server dies or hangs

If an application server dies (its process spontaneously closes), or freezes (its Web modules stop responding to new requests):

- Isolate the problem by installing Web modules on different servers, if possible.
- To detect memory leak problems, enable verbose garbage collection on the application server. This feature adds detailed statements to the JVM error log file of the application server about the amount of available and in-use memory. To set up verbose garbage collection:
  1. Select **Servers > Application Servers >** *server_name* **> Process Definition > Java Virtual Machine**, and enable **Verbose Garbage Collection**.
  2. Stop and restart the application server.
  3. Periodically, or after the application server stops, browse the log file for garbage collection statements. Look for statements beginning with ″allocation failure″. The string indicates that a need for memory allocation has triggered a JVM garbage collection (freeing of unused memory). Allocation failures themselves are normal and not necessarily indicative of a problem. The allocation failure statement is followed by statements showing how many bytes are needed and how many are allocated.

     If there is a steady increase in the total amount of free and used memory (the JVM keeps allocating more memory for itself), or if the JVM becomes unable to allocate as much memory as it needs (indicated by the bytes needed statement), there might be a memory leak.
- If the verbose garbage collection output indicates that the application server is running out of memory, one of the following problems might be present:
- If an application server spontaneously dies, there will be an SDUMP. See Using the Error Dump and Cleanup interface for instructions on how to analyze the dump.

For current information available from IBM Support on known problems and their resolution, see the IBM Support page.

IBM Support has documents that can save you time gathering information needed to resolve this problem. Before opening a PMR, see the IBM Support page.

Related tasks

Chapter 6, “Troubleshooting by task,” on page 233

Web container troubleshooting tips

If you are having problems starting a Web module, or accessing resources within a particular Web module:

- View the logs for the application server which hosts the problem Web modules, and look for messages in the JVM output file which indicate that the web module has started successfully. You should see messages similar to the following:

```
WebContainer  A SRVE0161I: IBM WebSphere Application Server - Web Container.
Copyright IBM Corp. 1998-2002
WebContainer  A SRVE0169I: Loading Web Module: [module_name]
ApplicationMg A WSVR0221I: Application started: [application_name]
HttpTransport A SRVE0171I: Transport http is listening on port [port_number]
[server_name] open for e-business in [install_root]/log/[server_name]/SystemOut.log
```

- For specific problems that can cause servlets, HTML files, and JavaServer Pages (JSP) files not to be served, seeWeb resource (JSP file, servlet, HTML file, image) does not display .

If application server related calls fail during Servlet.init method, you can either:

- Start the server manually when the server is ready for e-business instead of starting the servlet upon startup or
- You can choose not to make application server related calls in the servlet’s init method.

If the property to start servlets during application server startup is enabled, part of its startup process calls the Servlet.init method on its servlets when you start the Web container. Therefore, when the Web container is starts and calls the init method, other components such as Naming and Work Load Management may not be fully started yet. As a result, application server related calls may not work since all of the application server components may not be ready yet. Once the application server is ’ready for e-business’, it is completely ready.

For current information available from IBM Support on known problems and their resolution, see the IBM Support page.

IBM Support has documents that can save you time gathering information needed to resolve this problem. Before opening a PMR, see the IBM Support page.

Related tasks

Chapter 6, “Troubleshooting by task,” on page 233

“Troubleshooting by component” on page 5

JavaServer Pages source code shown by the Web server

Problem

If you share the document root of the WebSphere Application Server with the Web server document root, a security exposure can result as the Web server might display the JavaServer Pages (JSP) source file as plain text.

You can use the WebSphere Web server plug-in set of rules to determine whether a given request will be handled by the WebSphere Application Server. When an

incoming request fails to match those rules, the Web server plug-in returns control to the Web server so that the Web server can fulfill the request. In this case, the unknown host header causes the Web server plug-in to return control to the Web server because the rules do not indicate that the WebSphere Application Server should handle it. Therefore, the Web server looks for the request in the Web server document root. Since the JSP source file is stored in the document root of the Web server, the Web server finds the file and displays it as plain text.

Suggested solution

Move the WebSphere Application Server JSP source file outside of the Web server document root. Then, when this request comes in with the unknown host header, the plug-in returns control to the Web server and the JSP source file is not found in the document root. Therefore, the Web server returns a `404 File Not Found` error rather than the JSP source file.

JavaServer Pages engine troubleshooting tips

If you are having difficulty using the JavaServer Pages (JSP) engine, try these steps:

1. Determine whether other resources such as `.html` files or servlets are being requested and displayed correctly. If they are not, the problem probably lies at a deeper level, such as with the HTTP server.
2. If other resources are being displayed correctly, determine whether the JSP processor has started normally:
   - Browse the logs of the server hosting the JSP files you are trying to access. The following messages indicate that the JSP processor has started normally:

     ```
     Extension Processor [class com.ibm.ws.jsp.webcontainerext.JSPExtensionProcessor]
     was initialized successfully.
     Extension Processor [class com.ibm.ws.jsp.webcontainerext.JSPExtensionProcessor]
     has been associated with patterns [*.jsp *.jspx *.jsw *.jsv ].
     ```

     If the JSP processor fails to load, you will see a message such as

     ```
     No Extension Processor found for handling JSPs.
     JSP Processor not defined. Skipping : jspfilename.
     ```

     in the server log files.
3. If the JSP engine has started normally, the problem may be with the JSP file itself.
   - The JSP may have invalid JSP syntax and could not be processed by the JSP Processor. Examine the server log files of the target application for invalid JSP directive syntax messages. Errors similar to the following in a browser indicate this kind of problem:

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

     This example indicates that line 2, column 1 of the named JavaServer Pages file is missing a mandatory attribute for the jsp:include action. Similar messages are displayed for other syntax errors.
   - Examine the target application server's `SystemErr.log` files for problems with invalid Java syntax. Errors similar to **`Message: Unable to compile class for JSP`** in a browser indicate this kind of problem.

     The error message output from the Javac compiler will be found in the `SystemErr.log`. It might look like:

```
JSPG0091E: An error occurred at line: 2 in the file: /myJsp.jsp
JSPG0093E: Generated servlet error: c:\WASROOT\temp\ ...
test.war\_myJsp.java:16: myInt is already defined in com.ibm.ws.jsp20._myJsp
int myInt = 122; String myString = "number is 122"; static int myStaticInt=22;
int myInt=121;
         ^  1 error
```

Correct the error in the JSP file and retry the file.

- Examine the target application server's server log files files for problems with invalid Java syntax. Errors similar to **Message: Unable to compile class for JSP** in a browser indicate this kind of problem.

  The error message output from the Javac compiler will be found in the SystemErr.log server log files. It might look like:

```
JSPG0091E: An error occurred at line: 2 in the file: /myJsp.jsp
JSPG0093E: Generated servlet error: c:\WASROOT\temp\ ...
test.war\_myJsp.java:16: myInt is already defined in com.ibm.ws.jsp20._myJsp
int myInt = 122; String myString = "number is 122"; static int myStaticInt=22;
int myInt=121;
         ^  1 error
```

  Correct the error in the JSP file and retry the file.

If none of these steps solves the problem, check to see if the problem is identified and documented using the links in Diagnosing and fixing problems: Resources for learning. If you do not see a problem that resembles yours, or if the information provided does not solve your problem, contact IBM support for further assistance.

For current information available from IBM Support on known problems and their resolution, see the IBM Support page. The IBM Support page contains documents that can save you time gathering information needed to resolve this problem.

Related tasks

Chapter 6, "Troubleshooting by task," on page 233

Related reference

Troubleshooting installation problems

HTTP session manager troubleshooting tips

If you are having problems creating or using HTTP sessions with your Web application hosted by WebSphere Application Server, here are some steps to take:

- View the logs for the application server which hosts the problem application:
  - first, look at messages written while each application is starting. They will be written between the following two messages:

```
Starting application: application
.....................
Application started: application
```

  - Within this block, look for any errors or exceptions containing a package name of com.ibm.ws.webcontainer.httpsession. If none are found, this is an indication that the session manager started successfully.
  - Error "**SRVE0054E: An error occurred while loading session context and Web application**" indicates that SessionManager didn't start properly for a given application.
  - Look within the logs for any Session Manager related messages. These messages will be in the format SESNxxxxE and SESNxxxxW for errors and warnings, respectively, where xxxx is a number identifying the precise error. Look up the extended error definitions in the Session Manager message table.
- See Best practices for using HTTP Sessions.
- Alternatively, a special servlet can be invoked that displays the current configuration and statistics related to session tracking.

  - Servlet name: **com.ibm.ws.webcontainer.httpsession.IBMTrackerDebug**.
  - It can be invoked from any web module which is enabled to serve by class name. For example, using default_app, **http://localhost:9080/servlet/com.ibm.ws.webcontainer.httpsession.IBMTrackerDebug**.
  - If you are viewing the module via the serve-by-class-name feature, be aware that it may be viewable by anyone who can view the application. You may wish to map a specific, secured URL to the servlet instead and disable the serve-servlets-by-classname feature.
- If you are using **database-based persistent sessions**, look for problems related to the **data source** the Session Manager relies on to keep session state information. For details on diagnosing database related problems see Errors accessing a datasource or connection pool

Error message `SRVE0079E Servlet host not found` after you define a port

Error message SRVE0079E can occur after you define the port in WebContainer > HTTP Transports for a server, indicating that you do not have the port defined in your virtual host definitions. To define the port,

1. On the administrative console, go to Environment > Virtual Hosts > default_host> Host Aliases> New
2. Define the new port on host "*"

The application server gets EC3 - 04130007 ABENDs

To prevent an EC3 - 04130007 abend from occuring on the application server, change the HTTP Output timeout value. The custom property *ConnectionResponseTimeout* specifies the maximum number of seconds the HTTP port for an individual server can wait when trying to read or write data. For instructions on how to set *ConnectionResponseTimeout*, see HTTP transport custom properties.

If none of these steps fixes your problem, check to see if the problem has been identified and documented by looking at the available online support (hints and tips, technotes, and fixes).

For current information available from IBM Support on known problems and their resolution, see the IBM Support page.

IBM Support has documents that can save you time gathering information needed to resolve this problem. Before opening a PMR, see the IBM Support page.

Related tasks

Chapter 6, "Troubleshooting by task," on page 233

"Troubleshooting by component" on page 5

Problems creating or using HTTP sessions

Note: To view and update the Session Manager settings discussed here, use the administrative console. Select the application server that hosts the problem application, then under **Additional properties**, select **Web Container**, then **Session manager**.

What kind of problem are you having?
- HTTP Sessions are not getting created, or are lost between requests.

- HTTP Sessions are not persistent (session data lost when application server restarts, or not shared across cluster).
- Session is shared across multiple browsers on same client machine.
- Session is not getting invalidated immediately after specified Session timeout interval.
- Unwanted sessions are being created by jsps.
- Session data intended for one client is seen by another client

If your problem is not described here, or none of these steps fixes the problem:

HTTP Sessions are not getting created, or are lost between requests

By default, the Session Manager uses cookies to store the session ID on the client between requests. Unless you intend to avoid cookie-based session tracking, ensure that cookies are flowing between WebSphere Application Server and the browser:
- Make sure the **Enable cookies** checkbox is checked under the **Session tracking Mechanism** property.
- Make sure cookies are enabled on the browser you are testing from or from which your users are accessing the application.
- Check the Cookie domain specified on the SessionManager (to view the or update the cookie settings, in the **Session tracking mechanism->enable cookies** property, click **Modify**).
  - For example, if the cookie domain is set as ".myCom.com", resources should be accessed using that domain name, e.g. http://www.myCom.com/myapp/servlet/sessionservlet.
  - If the domain property is set, make sure it begins with a dot (.). Certain versions of Netscape do not accept cookies if domain name doesn't start with a dot. Internet Explorer honors the domain with or without a dot. For example, if the domain name is set to *mycom.com*, change it to *.mycom.com* so that both Netscape and Internet Explorer honor the cookie.
- Check the **Cookie path** specified on the SessionManager. Check whether the problem url is hierarchially below the Cookie path specified. If not correct the Cookie path.
- If the `Cookie maximum age` property is set, ensure that the client (browser) machine's date and time is the same as the server's, including the time zone. If the client and the server time difference is over the "Cookie maximum age" then every access would be a new session, since the cookie will "expire" after the access.
- If you have multiple web modules within an enterprise application that track sessions:
  - If you want to have different session settings among web modules in an enterprise application, ensure that each web module specifies a different cookie name or path, or
  - If Web modules within an enterprise application use a common cookie name and path, ensure that the HTTP session settings, such as Cookie maximum age, are the same for all Web modules. Otherwise cookie behavior will be unpredictable, and will depend upon which application creates the session. Note that this does not affect session data, which is maintained separately by Web module.
- Check the cookie flow between browser and server:
  1. On the browser, enable "cookie prompt". Hit the servlet and make sure cookie is being prompted.
  2. Access the session servlet from the browser.
  3. The browser will prompt for the cookie; note the jsessionid.
  4. Reload the servlet, note down the cookie if a new cookie is sent.

5. Check the session trace and look for the session id and trace the request by the thread. Verify that the session is stable across web requests:
   – Look for **getIHttpsession(...)** which is start of session request.
   – Look for **releaseSesson(..)** which is end of servlet request.

- If you are using URL rewriting instead of cookies:
  – Ensure there are no static HTML pages on your application's navigation path.
- If you are using SSL as your session tracking mechanism:
  – Ensure that you have SSL enabled on your IBM HTTP Server or iPlanet HTTP server.
- If you are in a clustered (multiple node) environment, ensure that you have session persistence enabled.

HTTP Sessions are not persistent

If your HTTP sessions are not persistent, that is session data is lost when the application server restarts or is not shared across the cluster:

- Check the Datasource.
- Check the SessionManager's Persistence Settings properties:
  – If you intend to take advantage of Session Persistence, verify that Persistence is set to **Database**.
  – If you are using **Database-based persistence**:
    - Check the jndi name of the datasource specified correctly on SessionManager.
    - Specify correct userid and password for accessing the database.

      Note that these settings have to be checked against the properties of an existing Data Source in the admin console. The Session Manager does not automatically create a session database for you.
    - The Datasource should be non-JTA, i.e. non XA enabled.
    - Check the logs for appropriate database error messages.
    - With DB2, for row sizes other than 4k make sure specified row size matches the DB2 page size. Make sure tablespace name is specified correctly.

Session is shared across multiple browsers on same client machine

This behavior is browser-dependent. It varies between browser vendors, and also may change according to whether a browser is launched as a new process or as a subprocess of an existing browser session (for example by hitting Ctl-N on Windows).

The Cookie maximum age property of the Session Manager also affects this behavior, if cookies are used as the session-tracking mechanism. If the maximum age is set to some positive value, all browser instances share the cookies, which are persisted to file on the client for the specifed maximum age time.

Session is not getting invalidated immediately after specified Session timeout interval

The SessionManager invalidation process thread runs every *x* seconds to invalidate any invalid sessions, where x is determined based on the Session timeout interval specified in the Session manager properties. For the default value of 30 minutes , *x* is around 300 seconds. In this case, it could take up to 5 minutes (300 seconds) beyond the timeout threshold of 30 minutes for a particular session to become invalidated.

Unwanted sessions are being created by jsps

As required by the JavaServer Page specification, jsps by default perform a request.getSession(true), so that a session is created if none exists for the client. To prevent jsps from creating a new session, set the session scope to **false** in the jsp file using the page directive as follows:

```
<% @page session="false" %>
```

Session data intended for one client is seen by another client

In rare situations, usually due to application errors, session data intended for one client might be seen by another client. This situation is referred to as session data crossover. When the *DebugSessionCrossover* custom property is set to true, code is enabled to detect and log instances of session data crossover. Checks are performed to verify that only the session associated with the request is accessed or referenced. Messages are logged if any discrepancies are detected. These messages provide a starting point for debugging this problem. This additional checking is only performed when running on the WebSphere-managed dispatch thread, not on any user-created threads.

For additional information on how to set this property, see article, Web container custom properties.

For current information available from IBM Support on known problems and their resolution, see the IBM Support page.

IBM Support has documents that can save you time gathering information needed to resolve this problem. Before opening a PMR, see the IBM Support page.

Related tasks

Chapter 6, “Troubleshooting by task,” on page 233

Enterprise bean and EJB container troubleshooting tips

If you are having problems starting an EJB container, or encounter error messages or exceptions that appear to be generated on by an EJB container, follow these steps to resolve the problem:

- Browse the relevant log files for clues:
  - Use the Administrative Console to verify that the application server which hosts the container is running.
  - Browse the logs for the application server which hosts the container. Look for the message **server *server_name* open for e-business** in the server log files. If it does not appear, or if you see the message **problems occurred during startup**, browse the server log files for details.

If none of these steps solves the problem, check to see if the problem is identified and documented using the links in Diagnosing and fixing problems: Resources for learning.

For current information available from IBM Support on known problems and their resolution, see the IBM Support page.

IBM Support has documents that can save you time gathering information needed to resolve this problem. Before opening a PMR, see the IBM Support page.

Related tasks

Chapter 6, “Troubleshooting by task,” on page 233

Related reference

Troubleshooting installation problems

Cannot access an enterprise bean from a servlet, a JSP file, a stand-alone program, or another client

What kind of error are you seeing?

- **javax.naming.NameNotFoundException: Name** *name* **not found in context** ″**local**″ **message** when access is attempted
- **BeanNotReentrantException** is thrown
- **CSITransactionRolledbackException / TransactionRolledbackException** is thrown
- Call fails, Stack trace beginning **EJSContainer E Bean method threw exception [exception_name]** found in JVM log file.
- Call fails, **ObjectNotFoundException or ObjectNotFoundLocalException** when accessing stateful session EJB found in JVM log file.
- Attempt to start CMP EJB module fails with **javax.naming.NameNotFoundException:** *dataSourceName*
-
- **Message BBOT0003W is issued**
- Symptom: **CNTR0001W: A Stateful SessionBean could not be passivated**

If the client is remote to the enterprise bean, which means, running in a different application server or as a stand-alone client, browse the logs of the application server hosting the enterprise bean as well as log files of the client.

ObjectNotFoundException or ObjectNotFoundLocalException when accessing stateful session EJB

A possible cause of this problem is that the stateful session bean timed out and was removed by the container. This event must be addressed in the code, according to the EJB 2.1 specification (available at `http://java.sun.com/products/ejb/docs.html`), section 7.6.2, Dealing with exceptions.

Stack trace beginning "EJSContainer E Bean method threw exception [exception_name]" found in JVM log file

If the exception name indicates an exception thrown by an IBM class that begins with ″com.ibm...″, then search for the exception name within the information center, and in the online help as described below. If ″exception name″ indicates an exception thrown by your application, contact the application developer to determine the cause.

javax.naming.NameNotFoundException: Name name not found in context "local"

A possible reason for this exception is that the enterprise bean is not local (not running in the same Java virtual machine [JVM] or application server) to the client JSP, servlet, Java application, or other enterprise bean, yet the call is to a ″local″ interface method of the enterprise bean . If access worked in a development environment but not when deployed to WebSphere Application Server, for example, it might be that the enterprise bean and its client were in the same JVM in development, but are in separate processes after deployment.

To resolve this problem, contact the developer of the enterprise bean and determine whether the client call is to a method in the local interface for the enterprise bean. If so, have the client code changed to call a remote interface method, or to promote the local method into the remote interface.

References to enterprise beans with local interfaces are bound in a name space local to the server process with the URL scheme of `local:`.

BeanNotReentrantException is thrown

This problem can occur because client code (typically a servlet or JSP file) is attempting to call the same stateful SessionBean from two different client threads. This situation often results when an application stores the reference to the stateful session bean in a static variable, uses a global (static) JSP variable to refer to the stateful SessionBean reference, or stores the stateful SessionBean reference in the HTTP session object. The application then has the client browser issue a new request to the servlet or JSP file before the previous request has completed.

To resolve this problem, ask the developer of the client code to review the code for these conditions.

CSITransactionRolledbackException / TransactionRolledbackException is thrown

An enterprise bean container throws these high-level exceptions to indicate that an enterprise bean call could not successfully complete. When this exception is thrown, logs to determine the underlying cause.

Some possible causes include:

- The enterprise bean might throw an exception that was not declared as part of its method signature. The container is required to roll back the transaction in this case. Common causes of this situation are where the enterprise bean or code that it calls throws a NullPointerException, ArrayIndexOutOfBoundsException, or other Java runtime exception, or where a BMP bean encounters a JDBC error. The resolution is to investigate the enterprise bean code and resolve the underlying exception, or to add the exception to the problem method signature.
- A transaction might attempt to do additional work after being placed in a `"Marked Rollback"`, `"RollingBack"`, or `"RolledBack"` state. Transactions cannot continue to do work after they are set to one of these states. This situation occurs because the transaction has timed out which, often occurs because of a database deadlock. Work with the application database management tools or administrator to determine whether database transactions called by the enterprise bean are timing out.
- A transaction might fail on commit due to dangling work from local transactions. The local transaction encounters some "dangling work" during commit. When a local transactions encounters an "unresolved action" the default action is to "rollback". You can adjust this action to "commit" in an assembly tool. Open the enterprise bean `.jar` file (or the `EAR` file containing the enterprise bean) and select the Session Beans or Entity Beans object in the component tree on the left. The Unresolved Action property is on the IBM Extensions tab of the container properties.

Attempt to start EJB module fails with "javax.naming.NameNotFoundException dataSourceName_CMP"exception

This problem can occur because:

- When the DataSource resource was configured, container managed persistence was not selected.
  - To confirm this problem, in the administrative console, browse the properties of the data source given in the NameNotFoundException. On the Configuration panel, look for a check box labeled **Container Managed Persistence**.
  - To correct this problem, select the check box for **Container Managed Persistence**.
- If container managed persistence is selected, it is possible that the CMP DataSource could not be bound into the namespace.
  - Look for additional naming warnings or errors in the status bar, and in the hosting application server logs. Check any further naming-exception problems that you find by looking at the topic "Cannot look up an object hosted by WebSphere Application Server from a servlet, JSP file, or other client" on page 46.

Message BBOT0003W is issued

Message BBOT0003W indicates a transaction timeout. The timeout might result in the abnormal termination of the servant where the transaction is executing.

- The default timeout value for enterprise bean transactions is 120 seconds. After this time, the transaction times out and the connection closes.
- If the transaction legitimately takes longer than the specified timeout period, on the administrative console:
  1. Go to **Manage Application Servers > server_name**
  2. Select the **Transaction Service properties** page
  3. Increase the **Total transaction lifetime timeout** value
  4. **Save** the configuration

  Note: z/OS will use the value you set for **Total transaction lifetime timeout** as the default transaction timeout setting. If you set a value for this property that is greater than the maximum transaction timeout value, z/OS will use the maximum transaction timeout value as the default.

Symptom:CNTR0001W: A Stateful SessionBean could not be passivated

This error can occur when a Connection object used in the bean is not closed or nulled out.

To confirm this is the problem, look for an exception stack in the logs for the EJB container that hosts the enterprise bean, and looks similar to:

```
StatefulPassi W CNTR0001W:
A Stateful SessionBean could not be passivated: StatefulBeanO
(BeanId(XXX#YYY.jar#ZZZZ),
state = PASSIVATING)
java.io.NotSerializableException: com.ibm.ws.rsadapter.jdbc.WSJdbcConnection
 at java.io.ObjectOutputStream.outputObject((Compiled Code))
 at java.io.ObjectOutputStream.writeObject(ObjectOutputStream.java(Compiled Code))
 at java.io.ObjectOutputStream.outputClassFields((Compiled Code))
 at java.io.ObjectOutputStream.defaultWriteObject((Compiled Code))
 at java.io.ObjectOutputStream.outputObject((Compiled Code))
 at java.io.ObjectOutputStream.writeObject(ObjectOutputStream.java(Compiled Code))
 at com.ibm.ejs.container.passivator.StatefulPassivator.passivate((Compiled Code))

 at com.ibm.ejs.container.StatefulBeanO.passivate((Compiled Code)
 at com.ibm.ejs.container.activator.StatefulASActivationStrategy.atUnitOfWorkEnd
                     ((Compiled Code))
 at com.ibm.ejs.container.activator.Activator.unitOfWorkEnd((Compiled Code))
 at com.ibm.ejs.container.ContainerAS.afterCompletion((Compiled Code)
```

where *XXX,YYY,ZZZ* is the Bean's name.

To correct this problem, the application must close all connections and set the reference to null for all connections. Typically this activity is done in the `ejbPassivate()` method of the bean. See the enterprise bean specification mandating this requirement, specifically section 7.4 in the EJB specification Version 2.1. Also, note that the bean must have code to reacquire these connections when the bean is reactivated. Otherwise, there are NullPointerExceptions when the application tries to reuse the connections.

Related tasks

Chapter 6, "Troubleshooting by task," on page 233

A client program does not work

What kind of problem are you seeing?

ActiveX client fails to display ASP files, or WebSphere Application Server resources (JSP files, servlet, or HTML pages) or both

A possible cause of this problem is that both IIS for serving Active Server Pages (ASP) files and an HTTP server that supports WebSphere Application Server (such as IBM HTTP Server) are deployed on the same host. This deployment leads to misdirected HTTP traffic if both servers are listening on the same port (such as the default port 80).

To resolve this problem, either:

- Open the IIS administrative panel, and edit the properties of the default Web server to change the port number to a value other than 80
- Install IIS and the HTTP server on separate servers.

For current information available from IBM Support on known problems and their resolution, see the IBM Support page.

IBM Support has documents that can save you time gathering information needed to resolve this problem. Before opening a PMR, see the IBM Support page.

Related tasks

Chapter 6, "Troubleshooting by task," on page 233

"Troubleshooting by component" on page 5

Debugging client exceptions

Start with the client and work your way backward to find the problem. When tracing exceptions back to the original problem, be aware that the RMI/IIOP protocol requires that some exceptions undergo conversion from one type to another as the exception passes through the runtime. Usually this transformation is between CORBA::SystemExceptions and RMI RemoteExceptions. Pay special attention to the CORBA::SystemException minor codes which indicate that a type transformation has occurred.

Caused by:	System exception (thrown by runtime)	User exception (thrown by application code)

Look for:	• CEEDUMPs in controller (region) or servant (region). These dumps indicate that the runtime had an error • JRAS error log entries, which can narrow the error down the exception to a specific function within the runtime	• CEEDUMPs • JRAS error log entries and traces
Actions:	• Look at the minor code that is listed. • Look for fixes that address similar symptoms or minor codes. • System exceptions usually represent the detection of an unexpected error, and therefore (unless directed by the documentation of the minor code) will often require IBM assistance to identify the problem.	• Look at your application for any sign of error. • Look for system failures, such as a system exception in the controller (region). If you find a system exception, follow the steps to the left for diagnosing a system exception.

Universal Discovery, Description, and Integration, Web Service, and SOAP component troubleshooting tips

If you are having problems deploying or executing applications that use WebSphere Application Server Web Services, Universal Discovery, Description, and Integration (UDDI), or SOAP, try these steps:

- Review the troubleshooting documentation for messaging in the information center:
  - WSIF troubleshooting tips
- Investigate the following areas for SOAP-related problems:
  - View the error log of the HTTP server to which the SOAP request is sent.
  - Browse the Web site `http://xml.apache.org/soap/` for FAQs and known SOAP issues.

If none of these steps solves the problem, check to see if the problem has been identified and documented using the links in Diagnosing and fixing problems: Resources for learning.

For current information available from IBM Support on known problems and their resolution, see the IBM Support page.

IBM Support has documents that can save you time gathering information needed to resolve this problem. Before opening a PMR, see the IBM Support page.

Related tasks

Chapter 6, “Troubleshooting by task,” on page 233

Related reference

Troubleshooting installation problems

Web services security troubleshooting tips

Troubleshooting Web services security is best done by reviewing the configurations with assembly tools so that you can match up the client and server request and the response configurations. These configurations must match. A client request sender configuration must match a server request receiver configuration. For encryption to

successfully occur, the public key of the receiver must be exported to the sender and this key must be configured properly in the encryption information. For authentication, you must specify the method used by the client in the login mapping of the server. Also, you must correctly specify the actor URI at each point in the configuration with the same URI string. The following includes a list of generic troubleshooting steps that you can perform. A listing of specific symptoms and solutions is provided after these steps.

Steps for this task:

1. Verify that the client security extensions and server security extensions match on each downstream call for the following senders and receivers:
   - Request sender and request receiver
   - Response sender and response receiver
2. Verify that when the **Add Created Time Stamp** option is enabled on the client-side that the server has the **Add Received Time Stamp** option configured. You must configure the security extensions with an assembly tool.
3. Verify that the client security bindings and the server security bindings are correctly configured. When the client authentication method is signature, make sure that the server has a login mapping. When the client uses the public key `cn=Bob,o=IBM,c=US` to encrypt the body, verify that this Subject is a personal certificate in the server key store so that it can decrypt the body with the private key. You can configure the security bindings using an assembly tool or the WebSphere Application Server administrative console.
4. Check the `SystemOut.log` file in the `${`*`USER_INSTALL_ROOT`*`}/logs/`*`server1`* directory (*server1* changes depending upon the server name) for messages that might provide information about the problem.
5. Enable trace for Web services security by using the following trace specification:
   ```
   com.ibm.xml.soapsec.*=all=enabled:com.ibm.ws.webservices.*=all=enabled:
   com.ibm.wsspi.wssecurity.*=all=enabled:com.ibm.ws.security.*=all=enabled:
   SASRas=all=enabled
   ```
   Type the previous three lines as one continuous line.

- Specific symptons:

  Symptom: `CWWSI5061E: The SOAP Body is not signed`

  Solution: This error usually occurs whenever the SOAP security handler does not load properly, and does not sign the SOAP body not to be signed. The SOAP security handler is typically the first validation that occurs on the server-side, so a multitude of problems can cause this message to display. The error might be caused by invalid actor URI configurations. You can configure the actor Universal Resource Identifier (URI) at the following locations within the assembly tool:

  From the Web services client editor within the assembly tool for client configurations:

  – Click **Security Extensions > Client Service Configuration Details** and indicate the actor information in the **ActorURI** field.

  – Click **Security Extensions > Request Sender Configuration section > Details** and indicate the actor information in the **Actor** field.

  From the Web Services Editor within the assembly tool for server configurations:

  – Click **Security Extensions > Server Service Configuration** section. Verify that the actor URI has the same actor string as the client-side.

  – Click **Security Extensions > Response Sender Service Configuration Details > Details** and indicate the actor information in the **Actor** field.

The actor information on both the client and the server must refer to the same string. When the actor fields on the client and the server match, the request or response is acted upon instead of being forwarded downstream. The actor fields might be different when you have Web services acting as a gateway to other Web services. However, in all other cases, verify that the actor information matches on the client and server. When the Web services implementation is acting as a gateway and it does not have the same actor configured as the request passing through the gateway, this Web services implementation does not process the message from the client. Instead, it sends the request downstream. The downstream process that contains the correct actor string processes the request. The same situation occurs for the response. Therefore, it is important that you verify that the appropriate client and server actor fields are synchronized.

Additionally, the error can appear when you do not specify that the body is signed in the client configuration. To sign the body part of the message using the Web service client editor in the assembly tool, click **Security Extensions > Request Sender Configuration > Integrity** and select the message parts to sign.

- **Symptom:** CWWSI5075E: No security token found that satisfies any one of the authentication methods.

  Solution: Verify that the client and server login configuration information matches in the security extensions. Also, verify that the client has a valid login binding and that the server has a valid login mapping in the security bindings. You can check this information by looking at the following locations in the assembly tool:

  From the Web services client editor within the assembly tool for client configurations:

  – Click **Security Extensions > Request Sender Configuration > Login Configuration** verify the authentication method.

  – Click **Port Binding > Security Request Sender Binding Configuration > Login Binding** verify the authentication method and other parameters.

  From the Web Services Editor within the assembly tool for server configurations:

  – Click **Security Extensions > Request Receiver Service Configuration Details > Login Configuration** and verify the authentication method.

  – Click **Binding Configurations > Request Receiver Binding Configuration Details > Login Mapping** and verify the authentication method and other parameters.

  Also, make sure that the actor URI specified on the client and server matches. You can configure the actor URI at the following locations within the assembly tool:

  From the Web services client editor within the assembly tool for client configurations:

  – Click **Security Extensions > Client Service Configuration Details** and indicate the actor information in the **ActorURI** field.

  – Click **Security Extensions > Request Sender Configuration section > Details** and indicate the actor information in the **Actor** field.

  From the Web services editor within the assembly tool for server configurations:

  – Click **Security Extensions > Server Service Configuration** section. Make sure that the **Actor URI** field has the same actor string as the client side.

  – Click **Security Extensions > Response Sender Service Configuration Details > Details** and indicate the actor information in the **Actor** field.

- **Symptom:** `CWWSI5094E: No UsernameToken of trusted user was found or the login failed for the user while the TrustMode is BasicAuth.`

Solution: This situation occurs when you have IDAssertion configured in the login configuration as the authentication method. On the sending Web service, configure a trusted basic authentication entry in the login binding. Then, on the server side, verify that the trusted ID evaluator has a property set that contains the user name of this basic authentication entry. To configure the client for identity assertion, see:

- Configuring the client for identity assertion: specifying the method
- Configuring the client for identity assertion: collecting the authentication method

To configure the server for identity assertion, see:

- Configuring the server to handle identity assertion authentication
- Configuring the server to validate identity assertion authentication information

- **Symptom:** The following authorization error occurs with UNAUTHENTICATED as the security name: `CWSCJ0053E: Authorization failed for /UNAUTHENTICATED while invoking (Home)com/ibm/wssvt/tc/pli/ejb/Beneficiary findBeneficiaryBySsNo(java.lang.String):2 securityName: /UNAUTHENTICATED;accessID: null is not granted any of the required roles: AgentRole`

  Solution: This situation occurs because a login configuration is not being configured or Web services Security is not configured from a client to a server. When the request arrives at the server and authentication information is not received, the UNAUTHENTICATED user is set on the thread. Authorization returns this error if there are any roles assigned to the resource except for the special ″Everyone″ role, which supports access by anyone.

  If the client successfully authenticates to an EJB file but the EJB file calls a downstream Enterprise JavaBean (EJB) file that is not configured with Web services security or transport security, such as HTTP user ID and password, an error can occur for this downstream request. Using the assembly tool, verify that the enterprise archive (EAR) file for both client and server has the correct security extensions and security bindings. For more information, see:

  - Configuring the client security bindings using an assembly tool
  - Configuring the security bindings on a server acting as a client using the administrative console
  - Configuring the server security bindings using an assembly tool
  - Configuring the server security bindings using the administrative console

  Related tasks

  Configuring the client for identity assertion: specifying the method

  Configuring the client for identity assertion: collecting the authentication method

  Configuring the server to handle identity assertion authentication

  Configuring the server to validate identity assertion authentication information

  Configuring the client security bindings using an assembly tool

  Configuring the security bindings on a server acting as a client using the administrative console

  Configuring the server security bindings using an assembly tool

  Configuring the server security bindings using the administrative console

  Chapter 6, “Troubleshooting by task,” on page 233

  Related reference

  Troubleshooting installation problems

Errors returned to a client sending a SOAP request

What kind of problem are you seeing?
- SOAPException: faultCode=SOAP-ENV:Client; msg=Error opening socket; java.net.ConnectException: Connection refused: connect
- javax.security.cert.CertPathBuilderException: No end-entity certificate matching the selection criteria could be found.

If none of these errors match the one you see:
- Browse the application server logs.
- Look up any error or warning messages in the message table.

SOAPException: faultCode=SOAP-ENV:Client; msg=Error opening socket; java.net.ConnectException: Connection refused: connect

The most likely cause of this refused connection is that it was sent to the default port, 80, and an HTTP server is not installed or configured.

To verify this situation, send the message directly to the SOAP port; for example, to `http://`*`hostname`*`:9080`. If the message is sent correctly, there are two ways to resolve the problem:
- Continue specifying port 9080 on SOAP requests.
- If an HTTP server is not installed, install one and the associated plug-in component.
- If an HTTP server is installed:
  - Regenerate the HTTP plug-in configuration in the administrative console by clicking **Environment > Update WebServer Plugin**, and restarting the HTTP server.
  - If the problem persists, view the HTTP server access and error logs, as well as the *plugin_install_root*`/logs/`*web_server_name*`/http_plugin.log` file for more information.

javax.security.cert.CertPathBuilderException: No end-entity certificate matching the selection criteria could be found

This error usually indicates that new or updated security keys are needed. The security key files are:
- SOAPclient
- SOAPserver
- sslserver.p12

In an installed application, these files are located in the: *install_dir*/installedApps/*application_name*.ear/soapsec.war/key/ directory. After replacing these files, you must stop and restart the application.

To replace these files in a SOAP-enabled application that is not yet installed:
- Expand the *application_name*.ear file.
- Expand the soapsec.war file.
- Replace the security key files in thekey/ directory.
- After you replace these files, install the application and restart the server.

For current information available from IBM Support on known problems and their resolution, see the IBM Support page.

IBM Support has documents that can save you time gathering information needed to resolve this problem. Before opening a PMR, see the IBM Support page.

Related tasks

Chapter 6, "Troubleshooting by task," on page 233

Related reference

Troubleshooting testing and first time run problems

JDBC and data source troubleshooting tips

To see whether your specific problem has been addressed, review the Cannot access a data source topic.

Trace strings for JDBC data sources

Turn on JDBC tracing by using the following trace strings:

- **com.ibm.ws.database.logwriter** Trace string for databases that use the GenericDataStoreHelper. You can also use this trace string for unsupported databases.
- **com.ibm.ws.db2.logwriter** Trace string for DB2 databases.
- **com.ibm.ws.oracle.logwriter** Trace string for Oracle databases.
- **com.ibm.ws.cloudscape.logwriter** Trace string for Cloudscape databases.
- **com.ibm.ws.informix.logwriter** Trace string for Informix databases.
- **com.ibm.ws.sqlserver.logwriter** Trace string for Microsoft SQL Server databases.
- **com.ibm.ws.sybase.logwriter** Trace string for Sybase databases.

The trace group that includes the trace strings is WAS.database.

JDBC trace properties

Use a back-end database that supports JDBC tracing. Setting trace strings does not result in a trace if the database does not support JDBC tracing. The following databases offer JDBC tracing at this time:

- DB2
- Oracle
- SQL Server

Set the level of trace desired for DB2 Universal database and Oracle as custom properties on the datasource.

- **DB2 Universal JDBC driver provider** Custom properties for DB2 are:
  - **traceLevel** Possible traceLevel values are:
    - TRACE_NONE = 0
    - TRACE_CONNECTION_CALLS = 1
    - TRACE_STATEMENT_CALLS = 2
    - TRACE_RESULT_SET_CALLS = 4
    - TRACE_DRIVER_CONFIGURATION = 16
    - TRACE_CONNECTS = 32
    - TRACE_DRDA_FLOWS = 64
    - TRACE_RESULT_SET_META_DATA = 128
    - TRACE_PARAMETER_META_DATA = 256
    - TRACE_DIAGNOSTICS = 512
    - TRACE_SQLJ = 1024
    - TRACE_ALL = -1

      Note: This trace level provides real data that sets to the PreparedStatement or gets from the ResultSet object.
  - **traceFile** Use this property to integrate the DB2 trace with the WebSphere Application Server trace. If you do not set the value, traces are integrated. Otherwise, DB2 traces are directed to the desired file. You can dynamically

enable or disable trace. You can run an application and turn on the DB2 trace if there is a problem. Use the run time trace enablement provided with the Application Server by specifying a trace string of com.ibm.ws.db2.logwriter=all=enabled.

- **Oracle JDBC provider** Custom properties for Oracle are:
  - **oraclelogCategoryMask** Controls the output category. The default is 47, which is (OracleLog.USER_OPER 1 | OracleLog.PROG_ERROR 2 | OracleLog.ERROR 4 | OracleLog.WARNING 8 | OracleLog.DEBUG1 32).

    Possible values are:
    - OracleLog.USER_OPER 1
    - OracleLog.PROG_ERROR 2
    - OracleLog.ERROR 4
    - OracleLog.WARNING 8
    - OracleLog.FUNCTION 16
    - OracleLog.DEBUG1 32
    - OracleLog.SQL_STR 128
  - **oraclelogModuleMask** Controls which modules write debug output. The default is 1, which is OracleLog.MODULE_DRIVER 1.

    Possible values are:
    - OracleLog.MODULE_DRIVER 1,
    - OracleLog.MODULE_DBACCESS 2
  - **oraclelogPrintMask** Controls which information to print with each trace message. The default is 62, which is ([OracleLog.FIELD_OBJECT for 9i / OracleLog.FIELD_CONN for 8i] 32 | OracleLog.FIELD_CATEGORY 16 | OracleLog.FIELD_SUBMOD 8 | OracleLog.FIELD_MODULE 4 | OracleLog.FIELD_TIME 2).

    Possible values are:
    - OracleLog.FIELD_TIME 2
    - OracleLog.FIELD_MODULE 4
    - OracleLog.FIELD_SUBMOD 8
    - OracleLog.FIELD_CATEGORY 16
    - OracleLog.FIELD_OBJECT 32
    - OracleLog.FIELD_THREAD 64

  Notes for Oracle JDBC tracing:
  1. Oracle 9i requires the use of `classes12_g.zip` to display traces. With Oracle8i, the `classes12_g.zip` is optional.
  2. You can dynamically enable or disable trace. You can run an application and turn on the Oracle trace if there is a problem. Use the run-time trace enablement provided with the WebSphere Application Server products, by specifying a trace string of com.ibm.ws.oracle.logwriter=all=enabled.

For current information available from IBM Support on known problems and their resolution, see the IBM Support page.

IBM Support has documents that can save you time gathering information needed to resolve this problem. Before opening a PMR, see the IBM Support page.

Related tasks

Chapter 6, “Troubleshooting by task,” on page 233

Related reference

Troubleshooting installation problems

“Cannot access a data source” on page 24

Cannot access a data source

What kind of database are you trying to access?
- Oracle
- DB2
- SQL Server
- Cloudscape
- Sybase
- General data access problems

If the errors described in the previous articles do not match the errors you see:
1. Browse the log files of the application server for clues.
2. Browse the Helper Class property of the data source to verify that it is correct and that it is on the WebSphere Application Server class path. Mysterious errors or behavior might result from a missing or misnamed Helper Class name. If WebSphere Application Server cannot load the specified class, it uses a default helper class that might not function correctly with your database manager.
3. Verify that the Java Naming and Directory Interface (JNDI) name of the data source matches the name used by the client attempting to access it. If error messages indicate that the problem might be naming-related, such as referring to the **name server** or **naming service**, or including error IDs beginning with **NMSV**, look at the Naming related problems and Troubleshooting the naming service component topics.
4. Enable tracing for the resource adapter using the trace specification, RRA=all=enabled. Follow the instructions for dumping and browsing the trace output, to narrow the origin of the problem.

If none of these steps fixes your problem, see if the problem is identified and documented in available online support (hints and tips, technotes, and fixes). If none of the online resources listed in the topic describes your problem, see "Obtaining help from IBM" on page 228.

General data access problems
- An exception "IllegalConnectionUseException" occurs
- WTRN0062E: An illegal attempt to enlist multiple one phase capable resources has occurred.
- ConnectionWaitTimeoutException.
- com.ibm.websphere.ce.cm.StaleConnectionException: [IBM][CLI Driver] SQL1013N The database alias name or database name "NULL" could not be found. SQLSTATE=42705
- java.sql.SQLException: java.lang.UnsatisfiedLinkError:
- "J2CA0030E: Method enlist caught java.lang.IllegalStateException" wrapped in error "WTRN0063E: An illegal attempt to enlist a one phase capable resource with existing two phase capable resources has occurred" when attempting to execute a transaction.
- java.lang.UnsatisfiedLinkError:xaConnect exception when attempting a database operation
- "J2CA0114W: No container-managed authentication alias found for connection factory or datasource *datasource*" when attempting a database operation
- An error is thrown if you use the ws_ant command to perform the database customization for Structured Query Language in Java on HP platforms

IllegalConnectionUseException

This error can occur because a connection obtained from a WAS40DataSource is being used on more than one thread. This usage violates the J2EE 1.3 programming model, and an exception generates when it is detected on the server. This problem occurs for users accessing a data source through servlets or bean-managed persistence (BMP) enterprise beans.

To confirm this problem, examine the code for connection sharing. Code can inadvertently cause sharing by not following the programming model recommendations, for example by storing a connection in an instance variable in a servlet, which can cause use of the connection on multiple threads at the same time.

WTRN0062E: An illegal attempt to enlist multiple one phase capable resources has occurred

This error can occur because:
- An attempt was made to share a single-phase connection, when each **getConnection** method has different connection properties; such as the AccessIntent. This attempt causes a non-shareable connection to be created.
- An attempt was made to have more than one unshareable connection participate in a global transaction, when the data source is not an XA resource.
- An attempt was made to have a one-phase resource participate in a global transaction while an XA resource or another one-phase resource already participated in this global transaction.
  - Within the scope of a global transaction you try to get a connection more than once and at least one of the resource-refs you use specifies that the connection is unshareable, and the data source is not configured to support two-phase commit transactions. It does not support an XAResource. If you do not use a resource-ref, you default to unshareable connections.
  - Within the scope of a global transaction you try to get a connection more than once and at least one of the resource-refs you use specifies that the connection is shareable and the data source is not configured to support two-phase commit transactions. That is, it does not support an XAResource. In addition, even though you specify that connections are shareable, each getConnection request is made with different connection properties (such as IsolationLevel or AccessIntent). In this case, the connections are not shareable, and multiple connections are handed back.
  - Multiple components (servlets, session beans, BMP entity beans, or CMP entity beans) are accessed within a global transaction. All use the same data source, all specify shareable connections on their resource-refs, and you expect them to all share the same connection. If the properties are different, you get multiple connections. AccessIntent settings on CMP beans change their properties. To share a connection, the AccessIntent setting must be the same. For more information about CMP beans sharing a connection with non-CMP components, see the *Data access application programming interface support* and *Example: Accessing data using IBM extended APIs to share connections between container-managed and bean-managed persistence beans* topics in the DataAccess section of the information center.

To correct this error:
- Check what your client code passes in with its getConnection requests, to ensure they are consistent with each other.
- Check the connection sharing scope from the resource binding, using anassembly tool.

  - If you are running an unshareable connection scope, verify that your data source is an XA data source.
  - If you are running a shareable connection scope, verify that all connection properties, including AccessIntent, are sharable.
- Check the JDBC provider implementation class from the Manage JDBC resource panel of the administrative console to ensure that it is a class that supports XA-type transactions.

ConnectionWaitTimeoutException accessing a data source or resource adapter

If your application receives exceptions like a com.ibm.websphere.ce.cm.ConnectionWaitTimeoutException or com.ibm.websphere.ce.j2c.ConnectionWaitTimeoutException when attempting to access a WebSphere Application Server data source or JCA-compliant resource adapter, respectively, some possible causes are:

- The maximum number of connections for a given pool is set too low. The demand for concurrent use of connections is greater then the configured maximum value for the connection pool. One indication that this situation is the problem is that you receive these exceptions regularly, but your CPU utilization is not high. This exception indicates that there are too few connections available to keep the threads in the server busy.
- Connection Wait Time is set too low. Current demand for connections is high enough such that sometimes there is not an available connection for short periods of time. If your connection wait timeout value is too low, you might timeout shortly before a user returns a connection back to the pool. Adjusting the connection wait time can give you some relief. One indication of this problem is that you use close to the maximum number of connections for an extended period and receiving this error regularly.
- You are not closing some connections or you are returning connections back to the pool at a very slow rate. This situation can happen when using unshareable connections, when you forget to close them, or you close them long after you are finished using them, keeping the connection from returning to the pool for reuse. The pool soon becomes empty and all applications get ConnectionWaitTimeoutExceptions. One indication of this problem is you run out of connections in the connection pool and you receive this error on most requests.
- You are driving more load than the server or backend system have resources to handle. In this case you must determine which resources you need more of and upgrade configurations or hardware to address the need. One indication of this problem is that the application or database server CPU is nearly 100% busy.

To correct these problems, either:

- Modify an application to use fewer connections
- Properly close the connections.
- Change the pool settings of MaxConnections or ConnnectionWaitTimeout.
- Adjust resources and their configurations.

com.ibm.websphere.ce.cm.StaleConnectionException: [IBM][CLI Driver] SQL1013N The database alias name or database name "NULL" could not be found. SQLSTATE=42705

This error occurs when a data source is defined but the **databaseName** attribute and the corresponding value are not added to the custom properties panel.

To add the **databaseName** property:

1. Click **Resources>Manage JDBC Providers** link in the administrative console.
2. Select the JDBC provider that supports the problem data source.
3. Select **Data Sources** and then select the problem data source.
4. Under **Additional properties** click **Custom Properties**.
5. Select the **databaseName** property, or add one if it does not exist, and enter the actual database name as the value.
6. Click **Apply** or **OK**, and then click **Save** from the action bar.
7. Access the data source again.

java.sql.SQLException: java.lang.UnsatisfiedLinkError:

This error indicates that the directory containing the binary libraries which support a database are not included in the LIBPATH environment variable for the environment in which the WebSphere Application Server starts.

The path containing the DBM vendor libraries vary by dbm. One way to find them is by scanning for the missing library specified in the error message. Then you can correct the LIBPATH variable to include the missing directory, either in the `.profile` of the account from which WebSphere Application Server is executed, or by adding a statement in a `.sh` file which then executes the `startServer` program.

"J2CA0030E: Method enlist caught java.lang.IllegalStateException" wrapped in error "WTRN0063E: An illegal attempt to enlist a one phase capable resource with existing two phase capable resources has occurred" when attempting to execute a transaction.

This error can occur when last participant support is missing or disabled. last participant support allows a one-phase capable resource and a two-phase capable resource to enlist within the same transaction.

Last participant support is only available if the following are true:
- WebSphere Application Server Programming Model Extensions (PME) is installed. PME is included in the Application Server Integration Server product.
- The Additional Integration Server Extensions option is enabled when PME is installed. If you perform a typical installation, this function is enabled by default. If you perform a custom installation, you have the option to disable this function, which disables last participant support.
- The application enlisting the one-phase resource is deployed with the **Accept heuristic hazard** option enabled. This deployment is done with an assembly tool such as the Application Server Toolkit (AST) or Rational Web Developer.

java.lang.UnsatisfiedLinkError:xaConnect exception when attempting a database operation

This problem has two main causes:
- The most common cause is that the jdbc driver which supports connectivity to the database is missing, or is not the correct version, or that native libraries which support the driver are on the system's path.
  - To resolve this problem on a Windows platform, verify that the JDBC driver jar file is on the system PATH environment variable:
    - If you are using DB2,verify that at least the DB2 client product has been installed on the WebSphere host
      - On DB2 version 7.2 or earlier, the file where the client product is installed on the WebSphere Application Server is db2java.zip. Verify that

the usejdbc2.bat program has been executed after the database install and after any upgrade to the database product.

- On DB2 version 8.1 or later, use the DB2 Universal JDBC Provider Driver when defining a JDBC provider in WebSphere Application Server. The driver file is db2jcc.jar. If you use the type 2 (default) option, verify that at least the DB2 client product is installed on the WebSphere Application Server host. If you specify the type 4 option, the DB2 client does not need to be installed, but the file db2jcc.jar still must be present.

  When specifying the location of the driver file, it is recommended to that you specify the path and file name of the target DB2 installation, rather than simply copying the file to a local directory, if possible. Otherwise, you may be exposed to problems if the target DB2 installation is upgraded and the driver used by WebSphere Application Server is not. If you choose **DB2 Legacy CLI-based type 2 JDBC Driver** when defining a JDBC provider in WebSphere Application Server, then you must still follow the steps to ensure that you have the correct version of the db2java.zip file (see instructions for DB2 7.2 or earlier).

– On a Unix platform, ensure that any native libraries required to support the database client of your database product are specified in the LD_LIBRARY_PATH environment variable in the profile of the account under which WebSphere Application Server executes.

  If you are using DB2 The native library is libdb2jdbc.so. The best way to ensure that this library is accessed correctly by WebSphere is to call the db2profile script supplied with DB2 from the .profile script of the account (such as ″root″) under which WebSphere runs.

  - If you are using DB2 version 7.2 or earlier, ensure that the usejdbc2,script provided with DB2 is called from the profile of the account under which WebSphere Application server is launched.
  - If you are using DB2 version 8.1 or later, see the previous instructions for the Windows operating system.

- If the database manager is DB2, you may have chosen the option to create a 64-bit instance. A 64-bit configuration is not supported. If this has happened, remove the database instance and create a new one with the default 32-bit setting.

  Note: For general help in configuring JDBC drivers and data sources in WebSphere Application Server, see the topic Accessing data from applications.

"J2CA0114W: No container-managed authentication alias found for connection factory or datasource *datasource*" when attempting a database operation

This error might occur in the SystemOut.log file when you run an application to access a data source after creating the data source using JACL script.

The error message occurs because the JACL script did not set container-managed authentication alias for CMP connection factory. The JACL is missing the following line:

```
$AdminConfig create MappingModule $cmpConnectorFactory  "{mappingConfigAlias
 DefaultPrincipalMapping} {authDataAlias $authDataAlias}
```

To correct this problem, add the missing line to the JACL script and run the script again. See Example: Creating a JDBC provider and data source using Java Management Extensions API and the scripting tool for a sample JACL script.

An error is thrown if you use the ws_ant command to perform the database customization for Structured Query Language in Java on HP platforms

If you use the `ws_ant` command to perform the database customization for Structured Query Language in Java (SQLJ) on HP platforms, you can receive an error similar to the following:

```
[java] [ibm][db2][jcc][sqlj]
[java] [ibm][db2][jcc][sqlj] Begin Customization
[java] [ibm][db2][jcc][sqlj] encoding not supported!!
```

The cause of this error might be that your databases were created using the HP default character set. The Java Common Client (JCC) driver depends on the software development kit (SDK) to perform the codepage conversions. The SDK shipped with this product, however, does not support the HP default codepage.

You need to set your LANG to the ISO locale before creating the databases. It should be similar to the following:

```
export LANG=en_US.iso88591
```

Refer to the DB2 Tech Notes at `http://www-3.ibm.com/software/data/db2/udb/ad/v8/bldg/t0004877.htm` for details.

Related tasks

Chapter 6, "Troubleshooting by task," on page 233

"Troubleshooting by component" on page 5

Example: Accessing data using IBM extended APIs to share connections between container-managed and bean-managed persistence beans

Related reference

Extensions to data access APIs

Problems accessing an Oracle data source

What kind of error do you see when you try to access your Oracle-based data source

- An Invalid Oracle URL is specified
- ″DSRA0080E: An exception was received by the Data Store Adapter. See original exception message: ORA-00600″ when connecting to or using an Oracle data source.
- ″DSRA8100E: Unable to get a {0} from the DataSource. Explanation: See the linkedException for more information.″
- An ″Error while trying to retrieve text for error″ message occurs when connecting to an Oracle data source.
- A java.lang.UnsatisfiedLinkError occurs when connecting to an Oracle data source.
- The exception java.lang.NullPointerException or an ″internal error: oracle.jdbc.oci8.OCIEnv″ occurs when connecting to an Oracle data source.
- WSVR0016W: Classpath entry, ${ORACLE_JDBC_DRIVER_PATH}/classes12.zip, in Resource, Oracle JDBC Thin Driver, located at cells/BaseApplicationServerCell/nodes/wasrtp/resources.xml has an invalid variable.

An invalid Oracle URL is specified

This error might be caused by an incorrectly specified URL on the URL property of the target data source.

Examine the URL property for the data source object in the administrative console. For the 8i OCI driver, verify that **oci8** is used in the URL. For the 9i OCI driver, you can use either **oci8** or **oci**.

Examples of Oracle URLs:
- For the thin driver: `jdbc:oracle:thin:@hostname.rchland.ibm.com:1521:IBM`
- For the thick (OCI) driver: `jdbc:oracle:oci8:@tnsname1`

"DSRA0080E: An exception was received by the data store adapter. See original exception message: ORA-00600" when connecting to or using an Oracle data source "DSRA0080E: An exception was received by the data store adapter. See original exception message: ORA-00600" when connecting to or using an Oracle data source "DSRA0080E: An exception was received by the Data Store Adapter. See original exception message: ORA-00600" when connecting to or using an Oracle data source

A possible reason for this exception is that the version of the Oracle JDBC driver being used is older than the Oracle database. It is possible that more than one version of the Oracle JDBC driver is configured on the WebSphere Application Server.

Examine the version of the JDBC driver. Sometimes you can determine the version by looking at the class path to determine what directory the driver is in.

If you cannot determine the version this way, use the following program to determine the version. Before running the program, set the class path to the location of your JDBC driver files.

```
import java.sql.*;
     import oracle.jdbc.driver.*;
     class JDBCVersion
     {
         public static void main (String args[])
         throws SQLException
         {
             // Load the Oracle JDBC driver
             DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());
             // Get a connection to a database
             Connection conn = DriverManager.getConnection
             ("jdbc:oracle:thin:@appaloosa:1521:app1","sys","change_on_install");
             // Create Oracle DatabaseMetaData object
             DatabaseMetaData meta = conn.getMetaData();
             // gets driver info:
             System.out.println("JDBC driver version is " + meta.getDriverVersion());
         }
     }
```

If the driver and the database are at different versions, replace the JDBC driver with the correct version. If multiple drivers are configured, remove any that occur at the incorrect level.

DSRA8100E: Unable to get a {0} from the DataSource. Explanation: See the linkedException for more information.

When using an oracle thin driver, Oracle throws a ″java.sql.SQLException: invalid arguments in call″ error if no user name or password is specified when getting a connection. If you see this error while running WebSphere Application Server, the alias is not set.

To remove the exception, define the alias on the data source.

"Error while trying to retrieve text for error" error when connecting to an Oracle data source

The most likely cause of this error is that the Oracle 8i OCI driver is being used with an ORACLE_HOME property that is either not set or is set incorrectly.

To correct the error, examine the user profile that WebSphere Application Server is running under to verify that the $ORACLE_HOME environment variable is set correctly.

"java.lang.UnsatisfiedLinkError:" connecting to an Oracle data source

The environment variable LIBPATH might not be set or is set incorrectly, if your data source throws an **UnsatisfiedLinkError** error, and the full exception indicates that the problem is related to an Oracle module, as in the following examples.

Example of invalid an LIBPATH for the 8i driver:

```
Exception in thread "main" java.lang.UnsatisfiedLinkError:
/usr/WebSphere/AppServer/java/jre/bin/libocijdbc8.so:
load ENOENT on shared library(s)
/usr/WebSphere/AppServer/java/jre/bin/libocijdbc8.so libclntsh.a
```

Example of an invalid LIBPATH for the 9i driver:

```
Exception in thread "main" java.lang.UnsatisfiedLinkError:
no ocijdbc9  (libocijdbc9.a or .so) in java.library.path
at java.lang.ClassLoader.loadLibrary(ClassLoader.java(Compiled Code))
at java.lang.Runtime.loadLibrary0(Runtime.java:780)
```

To correct the problem, examine the user profile under which the WebSphere Application Server is running to verify that the LIBPATH environment variable includes Oracle libraries. Scan for the `lobocijdbc8.so` file to find the right directory.

java.lang.NullPointerException referencing 8i classes, or " internal error: oracle.jdbc.oci8. OCIEnv" connecting to an Oracle data source

The problem might be that the 9i OCI driver is being used on an AIX 32-bit machine, the LIBPATH is set correctly, but the ORACLE_HOME environment variable is not set or is set incorrectly. You can encounter an exception similar to either of the following when your application attempts to connect to an Oracle data source:

Exception example for the java.lang.NullPointerException:

```
Exception in thread "main" java.lang.NullPointerException
at oracle.jdbc.oci8.OCIDBAccess.check_error(OCIDBAccess.java:1743)
at oracle.jdbc.oci8.OCIEnv.getEnvHandle(OCIEnv.java:69)
at oracle.jdbc.oci8.OCIDBAccess.logon(OCIDBAccess.java:452)
at oracle.jdbc.driver.OracleConnection. <init>(OracleConnection.java:287)
```

Exception example for the java.sql.SQLException:

```
Exception in thread "main" java.sql.SQLException:
internal error: oracle.jdbc.oci8. OCIEnv@568b1d21
at oracle.jdbc.dbaccess.DBError.throwSqlException(DBError.java:184)
at oracle.jdbc.dbaccess.DBError.throwSqlException(DBError.java:226)
at oracle.jdbc.oci8.OCIEnv.getEnvHandle(OCIEnv.java:79)
```

To correct the problem, examine the user profile that WebSphere Application Server is running under to verify that it has the `$ORACLE_HOME` environment variable

set correctly, and that the $LIBPATH includes $ORACLE_HOME/lib.

WSVR0016W: Classpath entry, ${ORACLE_JDBC_DRIVER_PATH}/classes12.zip, in Resource, Oracle JDBC Thin Driver, located at cells/BaseApplicationServerCell/nodes/wasrtp/resources.xml has an invalid variable

This error occurs when there is no environment variable defined for the property, ORACLE_JDBC_DRIVER_PATH.

Verify this problem in the administrative console. Go to **Environment > Manage WebSphere Variables** to verify whether the variable ORACLE_JDBC_DRIVER_PATH is defined.

To correct the problem, click **New** and define the variable. For example, name : **ORACLE_JDBC_DRIVER_PATH** , value : **c:\oracle\jdbc\lib** Use a value that names the directory in your operating system and directory structure that contains the classes12.zip file.

Related tasks

Chapter 6, "Troubleshooting by task," on page 233

Related reference

"Cannot access a data source" on page 24

"Problems accessing a DB2 database"

"Problems accessing a SQL server data source" on page 39

"Problems accessing a Cloudscape database" on page 40

"Problems accessing a Sybase data source" on page 43

Problems accessing a DB2 database

What kind of problem are you having accessing your DB2 database?

- "SQL0567N "DB2ADMIN " is not a valid authorization ID. SQLSTATE=42602" on page 33
- "SQL0805N Package package-name was not found" on page 33
- "SQL0805N Package "NULLID.SQLLC300" was not found. SQLSTATE=51002" on page 33
- "SQL30082N Attempt to establish connection failed with security reason "17" ("UNSUPPORTED FUNCTION') SQLSTATE=08001" on page 33
- "SQLException, with ErrorCode -99,999 and SQLState 58004, with Java "StaleConnectionException: COM.ibm.db2.jdbc.DB2Exception: [IBM][CLI Driver] CLI0119E Unexpected system failure. SQLSTATE=58004", when using WAS40-type data source" on page 34
- "Error message java.lang.reflect.InvocationTargetException: com.ibm.ws.exception.WsException: DSRA0023E: The DataSource implementation class "COM.ibm.db2.jdbc.DB2XADataSource" could not be found. when trying to access a DB2 database" on page 34
- "CLI0119E System error. SQLSTATE=58004 - DSRA8100 : Unable to get a XAconnection or DSRA0011E: Exception: COM.ibm.db2.jdbc.DB2Exception: [IBM][CLI Driver] CLI0119E Unexpected system failure. SQLSTATE=5800" on page 35
- "COM.ibm.db2.jdbc.DB2Exception: [IBM][CLI Driver][DB2/NT] SQL0911N The current transaction has been rolled back because of a deadlock or timeout. Reason code "2". SQLSTATE=40001" on page 35
- ""COM.ibm.db2.jdbc.DB2ConnectionPoolDataSource" could not be found for data source ([data-source-name])"" on page 36
- "Problems accessing a DB2 database"

- “ java.sql.SQLException: Failure in loading T2 native library db2jcct2 DSRA0010E: SQL State = null, Error Code = -99,999 ” on page 36
- Lock contention exception occurs in database when data source implementation type is XA
- “"DSRA8050W: Unable to find the DataStoreHelper class specified" exception occurs when trying to use a DB2 Universal Datasource in a mixed release cell.” on page 38
- “Receive "’SYSTEM’ is not a valid authorization ID" message when trying to access DB2 on a Windows machine where WebSphere Application Server is also installed.” on page 38
- Check the IBM support page for information on problems that occur when using connection pooling with DB2.

SQL0567N "DB2ADMIN " is not a valid authorization ID. SQLSTATE=42602

If you encounter this error when attempting to access a DB2 Universal Database (UDB):
1. Verify that your user name and password in the data source properties page in the administrative console are correct.
2. Ensure that the user ID and password do not contain blank characters before, in between, or after.

SQL0805N Package *package-name* **was not found**

Possible reasons for these exceptions:
- If the package name is `NULLID.SQLLC300`, see SQL0805N Package ″NULLID.SQLLC300″ was not found. SQLSTATE=51002. for the reason.
- You are attempting to use an XA-enabled JDBC driver on a DB2 database that is not XA-ready.

To correct the problem on a DB2 Universal Database (UDB), run this one-time procedure, using the db2cmd interface while connected to the database in question:
1. **DB2 bind @db2ubind.lst blocking all grant public**
2. **DB2 bind @db2cli.lst blocking all grant public**

The `db2ubind.lst` and `db2cli.lst` files are in the `bnd` directory of your DB2 installation root. Run the commands from that directory.

SQL0805N Package "NULLID.SQLLC300" was not found. SQLSTATE=51002

This error can occur because:
- The underlying database was dropped and recreated.
- DB2 was ugpraded and its packages are not rebound correctly.

To resolve this problem, rebind the DB2 packages by running the db2cli.lst script found in the bnd directory. For example: db2>@db2cli.lst.

SQL30082N Attempt to establish connection failed with security reason "17" ("UNSUPPORTED FUNCTION’) SQLSTATE=08001

This error can occur when the security mechanism specified by the client is not valid for this server. Some typical examples:
- The client sent a new password value to a server that does not support the change password function.
- The client sent SERVER_ENCRYPT authentication information to a server that does not support password encryption.

- The client sent a userid, but no password, to a server that does not support authentication by userid only.
- The client has not specified an authentication type, and the server has not responded with a supported type. This can include the server returning multiple types from which the client is unable to choose.

To resolve this problem, ensure that your client and server use the same security mechanism. For example, if this is an error on your data source, verify that you have assigned a user id and password or authentication alias.

SQLException, with ErrorCode -99,999 and SQLState 58004, with Java "StaleConnectionException: COM.ibm.db2.jdbc.DB2Exception: [IBM][CLI Driver] CLI0119E Unexpected system failure. SQLSTATE=58004", when using WAS40-type data source

An unexpected system failure usually occurs when running in XA mode (two-phase commit). Among the many possible causes are:
- An invalid username or password was provided.
- The database name is incorrect.
- Some DB2 packages are corrupted.

To determine whether you have a user name or password problem, look in the `db2diag.log` file to view the actual error message and SQL code. A message like the following example, with an SQLCODE of -1403, indicates an invalid user ID or password:

```
2002-07-26-14.19.32.762905   Instance:db2inst1   Node:000
PID:9086(java)   Appid:*LOCAL.db2inst1.020726191932
XA DTP Support  sqlxa_open   Probe:101
DIA4701E Database "POLICY2" could not be opened
for distributed transaction processing.
String Title: XA Interface SQLCA  PID:9086 Node:000
SQLCODE = -1403
```

To resolve these problems:
1. Correct your user name and password. If you specify your password on the GUI for the data source, ensure that the user name and password you specify on the bean are correct. The user name and password you specify on the bean overwrite whatever you specify when creating the data source.
2. Use the correct database name.
3. Rebind the packages (in the `bnd` directory) as follows:

   ```
   db2connect to dbname
   c:\SQLLIB\bnd>DB2 bind @db2ubind.lst blocking all grant public
   c:\SQLLIB\bnd>DB2 bind @db2cli.lst blocking all grant public
   ```
4. Ensure that the \WebSphere\AppServer\properties\wsj2cdpm.properties file has the right user ID and password.

Error message `java.lang.reflect.InvocationTargetException: com.ibm.ws.exception.WsException: DSRA0023E: The DataSource implementation class "COM.ibm.db2.jdbc.DB2XADataSource" could not be found.` when trying to access a DB2 database

One possible reason for this exception is that a user is attempting to use a JDBC 2.0 DataSource, but DB2 is not JDBC 2.0-enabled. This situation frequently happens with new installations of DB2 because DB2 provides separate drivers for JDBC 1.X and 2.0, with the same physical file name. By default, the JDBC 1.X driver is on the class path.

To confirm this problem:
- On Windows systems, look for the inuse file in the java12 directory in your DB2 installation root. If the file missing, you are using the JDBC 1.x driver.
- On UNIX systems, check the class path for your data source. If the class path does not point to the `db2java.zip` file in the `java12` directory, you are using the JDBC 1.x driver.

To correct this problem:
- On Windows systems, stop DB2. Run the usejdbc2.bat file from the java12 directory in your DB2 instalation root. Run this file from a command line to verify that it completes successfully.
- On UNIX systems, change the class path for your data source to point to the `db2java.zip` file in the `java12` directory of your DB2 installation root.

CLI0119E System error. SQLSTATE=58004 - DSRA8100 : Unable to get a XAconnection or DSRA0011E: Exception: COM.ibm.db2.jdbc.DB2Exception: [IBM][CLI Driver] CLI0119E Unexpected system failure. SQLSTATE=5800

If you encounter this error when attempting to access a DB2 Universal Database (UDB) data source:
1. Check your user name and password custom properties in the data source properties page in the administrative console. Verify that they are correct.
2. Ensure the user ID and password do not contain any blank characters, before, in between, or after.
3. Check that the `WAS.policy` file exists for the application, for example, `D:\WebSphere\AppServer\installedApps\markSection.ear\META-INF\was.policy`.
4. View the entire exception listing for an underlying SQL error, and look it up using the DBM vendor message reference.

If you encounter this error while running DB2 on Red Hat Linux, the **max queues system wide** parameter is too low to support DB2 while it acquires the necessary resources to complete the transaction. When this problem exists, the exceptions J2CA0046E and DSRA0010E can precede the exception DSRA8100E.

To correct this problem, edit the /proc/sys/kernal/msgmni file to increase the value of the **max queues system wide** parameter to a value greater than 128.

COM.ibm.db2.jdbc.DB2Exception: [IBM][CLI Driver][DB2/NT] SQL0911N The current transaction has been rolled back because of a deadlock or timeout. Reason code "2". SQLSTATE=40001

This problem is probably an application-caused DB2 deadlock, particularly if you see an error similar to the following when accessing a DB2 data source:

```
ERROR CODE: -911
COM.ibm.db2.jdbc.DB2Exception: [IBM][CLI Driver][DB2/NT] SQL0911N
The current transaction has been rolled back because of a deadlock or timeout.
Reason code "2".  SQLSTATE=40001
```

To diagnose the problem:
1. Execute these DB2 commands:
   a. `db2 update monitor switches using LOCK ON`
   b. `db2 get snapshot for LOCKS on dbName >`

   The *directory_name*`\lock_snapshot.log` now has the DB2 lock information.
2. Turn off the lock monitor by executing: `db2 update monitor switches using LOCK OFF`

To verify that you have a deadlock:
1. Look for an application handle that has a lock-wait status, and then look for the ID of the agent holding lock to verify the ID of the agent.
2. Go to that handle to verify it has a lock-wait status, and the ID of the agent holding the lock for it. If it is the same agent ID as the previous one, then you know that you have a circular lock (deadlock).

To resolve the problem:
1. Examine your application and use a less restrictive isolation level if no concurrency access is needed.
2. Use caution when changing the **accessIntent** value to move to a lower isolation level. This change can result in data integrity problems.
3. For DB2/UDB Version 7.2 and earlier releases, you can set the DB2_RR_TO_RS flag from the DB2 command line window to eliminate unnecessary deadlocks, such as when the accessIntent defined on the bean method is too restrictive, for example, PessimisticUpdate. The DB@_RR_TO_RS setting has two impacts:
   - If RR is your chosen isolation level, it is effectively downgraded to RS.
   - If you choose another isolation level, and the DB2_RR_TO_RS setting is on, a scan skips over rows that are deleted but not committed, even though the row might qualify for the scan. The skipping behavior affects the RR, Read Stability (RS), and Cursor Stability (CS) isolation levels.

   For example, consider the scenario where transaction A deletes the row with column1=10 and transaction B does a scan where column1>8 and column1<12. With DB2_RR_TO_RS off, transaction B waits for transaction A to commit or rollback. If transaction A rolls back, the row with column1=10 is included in the result set of the transaction B query. With DB2_RR_TO_RS on, transaction B does not wait for transaction A to commit or rollback. Transaction B immediately receives query results that do not include the deleted row. Setting DB2_RR_TO_RS effectively changes locking behavior, thus avoiding deadlocks.

"COM.ibm.db2.jdbc.DB2ConnectionPoolDataSource" could not be found for data source ([data-source-name])"

This error is denoted by message DSRA8040I: Failed to connect to the DataSource.

This error usually occurs when the class path of the DB2 JDBC driver is set correctly to `${DB2_JDBC_DRIVER_PATH}/db2java.zip` but the environment variable DB2_JDBC_DRIVER_PATH is not set.

This error can also occur if you are using DB2 Version 7.1 or 7.2 and you have not yet run *usejdbc2*. This might be the problem if your path is correct but you still receive this error.

To confirm this problem:
1. Go to the **Manage WebSphere Variables** panel.
2. Select **Environment** to verify that there is no entry for the variable DB2_JDBC_DRIVER_PATH.

To correct this problem: Add the variable DB2_JDBC_DRIVER_PATH with **value** equal to the directory path containing the `db2java.zip` file.

java.sql.SQLException: Failure in loading T2 native library db2jcct2 DSRA0010E: SQL State = null, Error Code = -99,999

The *Failure in loading* message indicates one of two things:

- Usually this happens when the machine was not rebooted after installing DB2. Reboot the machine getting the error and try it again.
- The DB2 context is not getting set up correctly for the user running WebSphere Application Server. Source the db2profile file on the machine, and ensure that the environment contains pointers to the DB2 native libraries.

Lock contention exception occurs in database when data source implementation type is XA

Note: Because a lock contention exception can be caused by many factors, consider the following explanation and recommended response as a strategy for eliminating the possible reasons for your lock contention problem.

Symptom	A lock contention exception occurs in a DB2 database that your application accesses through a data source of implementation type XA.
Problem	Your application is trying to access database records that are locked by an XA transaction that is in ended (e) state, but cannot be prepared by the transaction manager.
Description	An XA transaction to DB2 that ends, but cannot be prepared, is in ended (e) state. Because it is *not* considered to be *in doubt*, the transaction manager cannot recover this transaction. DB2 does not return it in the list of in doubt transactions. DB2 also does not roll the transaction back immediately; it waits until all connections to the database are released. During this period of inaction, the transaction continues to hold locks on the database. Due to certain policies of WebSphere Application Server workload management, your application server might not disconnect all connections from the database to allow rollback of the transaction. Therefore the ended transaction persists in locking the same database records. If your application attempts to access these locked records, a lock contention exception occurs in DB2.
Recommended response	DB2 Version 8.2 is shipped with a sample application that connects to a defined DB2 server and uses the available DB2 APIs to obtain a list of these particular ended transactions. The application offers a configuration setting that enables you to designate an amount of time after which the application rolls these transactions back. Locate the sample application in the `sqllib/samples/db2xamon.c` directory of DB2 Version 8.2 and run it.

"DSRA8050W: Unable to find the DataStoreHelper class specified" exception occurs when trying to use a DB2 Universal Datasource in a mixed release cell.

This error usually occurs when you are using WebSphere Application Server Version 6 in conjunction with a previous version and attempt to create a DB2 Universal Datasource on the previous version.

This can happen because the DB2 Universal Datasource was not available on Version 5 and previous versions, but the Version 6 administrative console allows you to build one.

To correct this problem: create the datasource on Version 6.

Receive "'SYSTEM' is not a valid authorization ID" message when trying to access DB2 on a Windows machine where WebSphere Application Server is also installed.

Symptom	For a WebSphere Application Server on Windows installation that uses DB2 as the backend, you see the following exception in the JVM log:
	`java.sql.SQLException: [IBM][CLI Driver] SQL0567N "SYSTEM" is not a valid authorization ID. SQLSTATE=42602 DSRA0010E: SQL State = 42602, Error Code = -567`
	`at COM.ibm.db2.jdbc.app.SQLExceptionGenerator. throw_SQLException(Unknown Source)`
	`at COM.ibm.db2.jdbc.app.SQLExceptionGenerator. check_return_code(Unknown Source)`
	`at COM.ibm.db2.jdbc.app.DB2Connection.connect (Unknown Source)`
	`at COM.ibm.db2.jdbc.app.DB2Connection.<init> (Unknown Source)`
	`at COM.ibm.db2.jdbc.app.DB2ReusableConnection. <init>(Unknown Source)`
	`at COM.ibm.db2.jdbc.DB2PooledConnection. getConnection(Unknown Source)`
	`at com.ibm.ws.rsadapter.spi.WSRdbDataSource. getConnection(WSRdbDataSource.java:1035)`
	`at com.ibm.ws.rsadapter.spi. WSManagedConnectionFactoryImpl. createManagedConnection (WSManagedConnectionFactoryImpl.java:937)`
	`at com.ibm.ejs.j2c.poolmanager.FreePool. createManagedConnectionWithMCWrapper (FreePool.java:1502)`

Problem	This exception occurs for configurations in which WebSphere Application Server is a client to the DB2 server. The underlying problem is an authorization conflict between WebSphere Application Server on Windows and DB2 that arises when an application attempts to connect to DB2 without providing a user ID and a password.

Description	When a DB2 client and the DB2 database run on the same machine, DB2 allows the client to connect without a user ID and password. The connection is made under the credentials of the user that owns the client process: in this case, the application server JVM. However, if WebSphere Application Server runs as a Windows service, and the ″Log on as″ option is set to ″Local System Account″, the application server JVM is categorized as a subcomponent of a special Windows user called SYSTEM. This user is not allowed to connect to DB2, resulting in the previously shown exception.
Recommended response	You have two options: • Modify the WebSphere Application Server service to use a **Log on as** option of **This account**, and provide an account with permission to connect to DB2. **Or** • Configure your application server to provide credentials on the DB2 connection by using container-managed or component-managed authentication.

Related tasks

Chapter 6, “Troubleshooting by task,” on page 233

“Troubleshooting by component” on page 5

Example: Accessing data using IBM extended APIs to share connections between container-managed and bean-managed persistence beans

Related reference

“Cannot access a data source” on page 24

“Problems accessing an Oracle data source” on page 29

“Problems accessing a SQL server data source”

“Problems accessing a Cloudscape database” on page 40

“Problems accessing a Sybase data source” on page 43

Extensions to data access APIs

Problems accessing a SQL server data source

What kind of problem are you having accessing your SQL Server database?

- ERROR CODE: 20001 and SQL STATE: HY000.
- Application fails with message stating ″Cannot find stored procedure...″
- ERROR CODE: SQL5042 when running a Java application

ERROR CODE: 20001 and SQL STATE: HY000 accessing SQLServer database

The problem might be that the distributed transaction coordinator service is not started. Look for an error similar to the following example when attempting to access an SQL server database:

```
ERROR CODE: 20001
SQL STATE:  HY000
java.sql.SQLException: [Microsoft][SQLServer JDBC Driver]
[SQLServer]xa_open (0) returns -3
at com.microsoft.jdbc.base.BaseExceptions.createException(Unknown Source) ...
at com.microsoft.jdbcx.sqlserver.SQLServerDataSource.getXAConnection
(Unknown Source) ...
```

To confirm this problem:

1. Go to the Windows **Control Panel** and click **Services**(or click **Control Panel > Administrative Tools > Services**)
2. Verify whether the service **Distributed Transaction Coordinator** or **DTC** is started.
3. If not, start the Distributed Transaction Coordinator service.

Application fails with message stating "Cannot find stored procedure..." accessing an SQLServer database

This error can occur because the stored procedures for the Java Transaction API (JTA) feature are not installed on the Microsoft SQL Server.

To resolve the problem: Repeat the installation for the stored procedures for the JTA feature, according to the ConnectJDBC installation guide.

ERROR CODE: SQL5042 when running a Java application

This error can occur when you configure your application to run in the following manner:

1. you use a type 2 (application) driver running on the gateway to the OS 390
2. your application is an XA application.

OS 390 does not use XA, but uses SPM. To resolve the problem:

1. Check your dbm cfg to see that the SPM is not started on the gateway.
2. Assign a port and set the *db2comm* variable to **TCPIP**.
3. Update the dbm cfg value *SPM_NAME* to use your machine name.
4. Start the SPM on the gateway.

Related tasks

Chapter 6, “Troubleshooting by task,” on page 233

“Troubleshooting by component” on page 5

Example: Accessing data using IBM extended APIs to share connections between container-managed and bean-managed persistence beans

Related reference

“Cannot access a data source” on page 24

“Problems accessing an Oracle data source” on page 29

“Problems accessing a DB2 database” on page 32

“Problems accessing a Cloudscape database”

“Problems accessing a Sybase data source” on page 43

Extensions to data access APIs

Problems accessing a Cloudscape database

What kind of problem are you having accessing your Cloudscape database?

- Unexpected IOException wrapped in SQLException, accessing Cloudscape database.
- The ″Select for update″ operatioin on one row causes table to become locked, triggering a deadlock condition.
- ″ERROR XSDB6: Another instance of Cloudscape might have already booted the database *databaseName*.″ error starting application server.
- Error ″The version of the IBM Universal JDBC driver in use is not licensed for connectivity to Cloudscape databases″

- Running an application causes a runtime exception which produces an unreadable message.

Tip: Cloudscape errorCodes (2000, 3000, 4000) indicate levels of severity, not specific error conditions. In diagnosing Cloudscape problems, pay attention to the given sqlState value.

Unexpected IOException wrapped in SQLException, accessing Cloudscape database

This problem can occur because Cloudscape databases use a large number of files. Some operating systems, such as the Solaris Operating Environment, limit the number of files an application can open at one time. If the default is a low number, such as 64, you can get this exception.

If your operating system lets you configure the number of file descriptors, you can correct the problem by setting the number to a high value, such as 1024.

The "select for update" operation causes table lock and deadlock when accessing Cloudscape

If a select for update operation on one row locks the entire table, which creates a deadlock condition, there might be undefined indexes on that table. The lack of an index on the columns you use in the `where` clause can cause Cloudscape to create a table lock rather than a row level lock.

To resolve this problem, create an index on the affected table.

ERROR XSDB6: Another instance of Cloudscape may have already booted the database "database"

This problem occurs because Cloudscape embedded framework only allows one Java virtual machine (JVM) to access the database instance at a time.

To resolve this problem:
1. Verify that you do not have other JDBC client programs, such as **ij** or **cview** running on that database instance, when WebSphere Application Server is running.
2. Verify that you do not use the same instance of the database for more than one data source or use the networkServer framework, which doesn't have this limitation.
3. If there are no connections to Cloudscape, delete the db.lck lock file. This file can be found in the directory where the Cloudscape database is mounted, under the schema directory. For example, if the database is mounted at /myCloudscapeDB, issue the command: rm /myCloudscapeDB/schemaName/db.lck

Error ″The version of the IBM Universal JDBC driver in use is not licensed for connectivity to Cloudscape databases″

Error "The version of the IBM Universal JDBC driver in use is not licensed for connectivity to Cloudscape databases"

At the client runtime, an error similar to the following occurs:

```
The version of the IBM Universal JDBC driver in use is not
licensed for connectivity to Cloudscape databases.  To connect
to this DB2 server, please obtain a licensed copy of the IBM DB2
Universal Driver for JDBC and SQLJ.  An appropriate license file
db2jcc_license_*.jar for this target platform must be installed to
the application classpath.  Connectivity to Cloudscape databases is
enabled by any of the following license files:
{ db2jcc_license_c.jar, b2jcc_license_cu.jar, db2jcc_license_cisuz.jar }
```

The problem occurs because an incorrect JDBC driver jar file name is specified in the class path for JDBC provider. For example, the jar file name may have an extra '_', as follows:

```
${UNIVERSAL_JDBC_DRIVER_PATH}/db2jcc_license__cu.jar
```

To resolve the problem:

1. Correct the UNIVERSAL_JDBC_DRIVER_PATH jar file name in the JACL script
2. Restart the cluster.
3. Rerun the client.

Running an application causes a runtime exception which produces an unreadable message.

At client runtime, you may receive a message similar to the following: `Caused by: com.ibm.db2.jcc.a.SqlException: DB2 SQL error: SQLCODE: -1, SQLSTATE: 42X05, SQLERRMC: ANNUITYHOLDER20^T42X05`

The problem occurs because the property *retrieveMessagesfromServerOnGetMessage*, which is required by WebSphere Application Server, has not been set.

To resolve the problem, on the admininstrative console

1. Click **Resources -> JDBC Providers**
2. Click on a Cloudscape provider
3. Scroll down and click on **Data Sources**
4. Select your data source (or add a new one)
5. Scroll down and select **Custom Properties**
6. If the property *retrieveMessagesFromServerOnGetMessage* already exists, set its value to true. If the property does not exist, select **New**and add the property *retrieveMessagesFromServerOnGetMessage* with a value **true**
7. Rerun the client

The SystemOut.log will now generate readable messages so that you can resolve the underlying problem.

Related tasks

Chapter 6, "Troubleshooting by task," on page 233

Related reference

"Cannot access a data source" on page 24
"Problems accessing an Oracle data source" on page 29
"Problems accessing a DB2 database" on page 32
"Problems accessing a SQL server data source" on page 39
"Problems accessing a Sybase data source" on page 43

Problems accessing a Sybase data source

What kind of problem are you having accessing your Sybase database?

- ″Sybase Error 7713: Stored Procedure can only be executed in unchained transaction mode″ error.
- ″JZ0XS: The server does not support XA-style transactions. Please verify that the transaction feature is enabled and licensed on this server.″
- A container managed persistence (CMP) enterprise bean is causing exceptions.

"Sybase Error 7713: Stored Procedure can only be executed in unchained transaction mode" error

This error occurs when either:

- The JDBC attempts to put the connection in **autocommit(true)** mode.
- A stored procedure is not created in a compatible mode.

To fix the **autocommit(true)** mode problem, let the application change the connection to chained mode using the **Connection.setAutoCommit(false)** mode, or use a **set chained on** language command.

To resolve the stored procedure problem, use the `sp_procxmode` *procedure_name* `"anymode"` command.

"JZ0XS: The server does not support XA-style transactions. Please verify that the transaction feature is enabled and licensed on this server."

This error occurs when XA-style transactions are attempted on a server that does not have Distributed Transaction Management (DTM) installed.

To resolve this problem, use the instructions in the Sybase Manual titled: *Using Adaptive Server Distributed Transaction Management Features* to enable Distributed Transaction Management (DTM). The main steps in this procedure are:

1. Install the DTM option.
2. Check the `license.dat` file to verify that the DTM option is installed.
3. Restart the license manager.
4. Enable DTM in ISQL.
5. Restart the ASE service.

A container managed persistence (CMP) enterprise bean is causing exceptions

This error is caused by improper use of reserved words. Reserved words cannot be used as column names.

To correct this problem: Rename the variable to remove the reserved word. You can find a list of reserved words in the *Sybase Adaptive Server Enterprise Reference Manual; Volume 1: Building Blocks*, Chapter 4. This manual is available online at: http://manuals.sybase.com/onlinebooks/group-as/asg1250e/refman.

Related tasks

Chapter 6, “Troubleshooting by task,” on page 233

Related reference

“Cannot access a data source” on page 24

“Problems accessing an Oracle data source” on page 29

“Problems accessing a DB2 database” on page 32

“Problems accessing a SQL server data source” on page 39

“Problems accessing a Cloudscape database” on page 40

Errors in messaging

What kind of problem are you seeing?
- javax.jms.JMSException: MQJMS2008: failed to open MQ queue in JVM log.

javax.jms.JMSException: MQJMS2008: failed to open MQ queue in JVM log

This error can occur when the MQ queue name is not defined in the internal Java Message Service (JMS) server queue names list. This problem can occur if a WebSphere Application Server queue destination is created, without adding the queue name to the internal JMS server queue names list.

To resolve this problem:
1. Open the WebSphere Application Server administrative console.
2. Click **Servers > Manage Application Servers >** *server_name*> **Server Components > JMS Servers**.
3. Add the queue name to the list.
4. Save the changes and restart the server.

For current information available from IBM Support on known problems and their resolution, see the IBM Support page.

IBM Support has documents that can save you time gathering information needed to resolve this problem. Before opening a PMR, see the IBM Support page.

Related tasks

Chapter 6, "Troubleshooting by task," on page 233

Related reference

Troubleshooting testing and first time run problems

Debugging problems related to Java Message Service (JMS) support

You might encounter JMS-related errors in the WebSphere Application Server for z/OS environment. To debug these errors, use any of the various WebSphere console settings that controls the type of trace data collected.

com.ibm.ejs.*=all=enabled
: Turns on all container tracing

com.ibm.ejs.j2c.*=all=enabled
: Turns on all tracing for connector support in WebSphere Application Server for z/OS

Messaging=all=enabled
: Collects trace data for the JMS and Message-driven bean (MDB) components of WebSphere Application Server for z/OS

If your installation configures WebSphere MQ to provide Java Message Service (JMS) support, you might need to use specific MQ tools for diagnosis:
- WebSphere variable *JMSApi=all=enabled*, which turns on all tracing for applications that use the JMS application programming interface (API).
- MQSC commands, which allow you to display information and perform other operations.
- The CSQUTIL utility program, which helps you to perform backup, restoration, and reorganization tasks, and to issue WebSphere MQ commands.

For more information about these diagnostic tools, refer to WebSphere MQ books, which are available through the Web site:

http://www.ibm.com/software/ts/mqseries/library/

The most useful for diagnostic information are:
- WebSphere MQ for z/OS Problem Determination, GC34-6054
- WebSphere MQ for z/OS Messages and Codes, SC34-6056
- WebSphere MQ Script (MQSC) Command Reference, SC34-6055
- WebSphere MQ for z/OS System Administration Guide, SC34-6053

All of these publications can be accessed from the IBM Publications Center.

Related tasks

"Steps for configuring WebSphere variables" on page 167

Errors connecting to WebSphere MQ and creating WebSphere MQ queue connection factory

The following exception may occur when trying to create the MDBListener instance:

```
6/23/03 22:45:58:232 CDT] 673106a8 MsgListenerPo W WMSG0049E:
Failed to start MDB PSSampleMDB against listener port SamplePubSubListenerPort
[6/23/03 22:47:58:289 CDT] 673106a8 FreePool E J2CA0046E:
Method createManagedConnctionWithMCWrapper caught an exception
during creation of the ManagedConnection for resource
JMS$SampleJMSQueueConnectionFactory, throwing ResourceAllocationException.
Original exception:  javax.resource.spi.ResourceAdapterInternalException:
 createQueueConnection failed
com.ibm.mq.MQException: MQJE001: An MQException occurred:
Completion Code 2, Reason 2009
MQJE003: IO error transmitting message buffer at
com.ibm.mq.MQManagedConnectionJ11.(MQManagedConnectionJ11.java:239)
```

This problem occurs because the MQ manager userid does not have write access to the /tmp directory. To correct this problem, before you use a Jacl procedure to configure WebSphere Application Server resources and install an application:

1. Ensure that all applications have write access to /tmp directory. Use the chmod 1777 command on the directory if necessary.
2. Create another subdirectory under /tmp (for example, /tmp/mydir). Use this directory as a "working directory" for the Jacl.
3. Restart the server.

Applications that use messaging on startup should start successfully.

Related tasks

Chapter 6, "Troubleshooting by task," on page 233

Related reference

Troubleshooting testing and first time run problems

Naming services component troubleshooting tips

Naming is a J2EE service which publishes and provides access to resources such as connection pools, enterprise beans, and message listeners to client processes. If you have problems in accessing a resource which otherwise appears to be healthy, the naming service might be involved. To investigate problems with the WebSphere Application Server Naming service:
- With WebSphere Application Server running, run the **dumpNameSpace** command for Windows systems, or the **dumpNameSpace.sh** command for UNIX and z/OS systems, and pipe, redirect, or "more" the output so that it is easily viewed. This

command results in a display of objects in the WebSphere Application Server namespace, including the directory path and object name.
- If the object a client needs to access does not appear, use the administrative console to verify that:
  - The server hosting the target resource is started.
  - The Web module or EJB container, if applicable, hosting the target resource is running.
  - The JNDI name of the target resource is correct and updated.
  - If the problem resource is remote, that is, not on the same node as the Name Server node, that the JNDI name is fully qualified, including the host name. This is especially applicable to Network Deployment configurations
- If you see an exception that appears to be CORBA related ("CORBA" appears as part of the exception name) look for a naming-services-specific CORBA minor code, further down in the exception stack, for information on the real cause of the problem. For a list of naming service exceptions and explanations, see the class com.ibm.websphere.naming.WsnCorbaMinorCodes in the Javadoc.

If none of these steps solve the problem:
- For specific problems that can cause access to named object hosted in WebSphere Application Server to fail, see Cannot look up an object hosted by WebSphere Application Server from a servlet, JSP file, or other client.
- Check to see if the problem has been identified and documented using the links in Diagnosing and fixing problems: Resources for learning.

For current information available from IBM Support on known problems and their resolution, see the IBM Support page.

IBM Support has documents that can save you time gathering information needed to resolve this problem. Before opening a PMR, see the IBM Support page.

Related tasks

Chapter 6, "Troubleshooting by task," on page 233

Related reference

Troubleshooting installation problems

Cannot look up an object hosted by WebSphere Application Server from a servlet, JSP file, or other client

To resolve problems encountered when a servlet, JSP file, standalone application or other client attempts to access an enterprise bean, ConnectionPool, or other named object hosted by WebSphere Application Server, you must first verify that the target server can be accessed from the client:
- From a command prompt on the client's server, enter "`ping` *`server_name`*" and verify connectivity.
- Use the WebSphere Application Server administrative console to verify that the target resource's application server and, if applicable, EJB module or Web module, is started.

Continue only if there is no problem with connectivity and the target resource appears to be running.

What kind of error are you seeing?
- "NameNotFoundException from JNDI lookup operation " on page 47
- "CannotInstantiateObjectException from JNDI lookup operation " on page 47
- "Message NMSV0610I appears in the server's log file, indicating that some Naming exception has occurred " on page 48

NameNotFoundException from JNDI lookup operation

If you encounter this exception in trying to access an enterprise bean, data source, messaging resource, or other resource:

- Browse the properties of the target object in the administrative console, and verify that the jndi name it specifies matches the JNDI name the client is using.
- If you are looking up an object that resides on a server different from the one from which the initial context was obtained, you must use the fully qualified name.
  - If access is from another server object such as a servlet accessing an enterprise bean and you are using the default context, not specifying the fully qualified JNDI name, you may get this error if the object is being hosted on a different server.
  - If access is from a stand-alone client, it may be that the object you are attempting access is on a server different from the server from which you obtained the initial context.

To correct this problem, use the fully-qualified JNDIname:

- If the object is in a single server: `cell/nodes/`*`nodeName`*`/servers/`*`serverName`*`/`*`jndiName`*. Objects are not supported in this release.
- If the object is on a server cluster: `cell/clusters/`*`clusterName`*`/`*`jndiName`*.

CannotInstantiateObjectException from JNDI lookup operation

If you encounter this exception in trying to access an enterprise bean, data source, messaging resource, or other resource, possible causes include:

- A serialized Java object is being looked up, but the necessary classes required to deserialize it are not in the runtime environment.
- A Reference object is being looked up, and the associated factory used to process it as part of the lookup processing is failing.

To determine the precise cause of the problem:

- Look in the logs of the server hosting the target resource. Look for exceptions immediately preceding the CannotInstantiateObjectException. If it is a java.lang.NoClassDefFoundError or java.lang.ClassNotFoundException, make sure the class referenced in the error message can be located by the class loader.
- Print out the stack trace for the root cause and look for the factory class. It will be called by `javax.naming.NamingManager.getObjectInstance()`. The reason for the failure will depend on the factory implementation, and may require you to contact the developer of the factory class.

Message NMSV0610I appears in the server's log file, indicating that some Naming exception has occurred

This error is informational only and is provided in case the exception is related to an actual problem. Most of the time, it is not. If it is, the log file should contain adjacent entries to provide context.

- If no problems are being experienced, ignore this message. Also ignore the message if the problem you are experiencing does not appear to be related to the exception being reported and if there are no other adjacent error messages in the log.
- If a problem is being experienced, look in the log for underlying error messages.
- The information provided in message NMSV0610I can provide valuable debug data for other adjacent error messages posted in response to the Naming exception that occurred.

OperationNotSupportedException from JNDI Context operation

This error has two possible causes:

- An update operation, such as a bind, is being performed with a name that starts with "`java:comp/env`". This context and its subcontexts are read-only contexts.
- A Context bind or rebind operation of a non-CORBA object is being performed on a remote name space that does not belong to WebSphere Application Server. Only CORBA objects can be bound to these CosNaming name spaces.

To determine which of these errors is causing the problem, check the full exception message.

WSVR0046E: Failed to bind, ejb/jndiName: ejb/jndiName. Original exception : org.omg.CosNaming.NamingContextPackage.AlreadyBound

This error occurs two enterprise bean server applications were installed on the same server such that a binding name conflict occurred. That is, a jndiName value is the same in the two applications' deployment descriptors. The error will surface during server startup when the second application using that jndiName value is started.

To verify that this is the problem, examine the deployment descriptors for all enterprise bean server applications running in the server in search for a jndiName that is specified in more than one enterprise bean application.

To correct the problem, change any duplicate jndiName values to ensure that each enterprise bean in the server process is bound with a different name.

ConfigurationException from "new InitialContext" operation or from a JNDI Context operation with a URL name

If you are attempting to obtain an initial JNDI context, a configuration exception can occur because an invalid JNDI property value was passed to the InitialContext constructor. This includes JNDI properties set in the System properties or in some jndi.properties file visible to the class loader in effect. A malformed provider URL is the most likely property to be incorrect. If the JNDI client is being run as a thin client such that the `CLASSPATH` is set to include all of the individual jar files required, make sure the `.jar` file containing the properties file `com/ibm/websphere/naming/jndiprovider.properties` is in the `CLASSPATH`.

If the exception is occurring from a JNDI Context call with a name in the form of a URL, the current JNDI configuration may not be set up properly so that the required factory class name cannot be determined, or the factory may not be visible to the class loader currently in effect. If the name is a Java: URL, the JNDI client must be running in a J2EE client or server environment. That is, the client must be running in a container.

Check the exception message to verify the cause.

If the exception is being thrown from the InitialContext constructor, correct the property setting or the `CLASSPATH`.

If the exception is being thrown from a JNDI Context method, make sure the property `java.naming.factory.url.pkgs` includes the package name for the factory required for the URL scheme in the name. URL names with the Java scheme can only be used while running in a container.

ServiceUnavailableException from "new InitialContext" operation

This exception indicates that some unexpected problem occurred while attempting to contact the name server to obtain an initial context. The ServiceUnavailableException, like all NamingException objects, can be queried for a root cause. Check the root cause for more information. It is possible that some of the problems described for CommunicationExceptions may also result in a ServiceUnavailableException.

Since this exception is triggered by an unexpected error, there is no probable cause to confirm. If the root cause exception does not indicate what the probable cause is, investigate the possible causes listed for CommunicationExceptions.

CommunicationException thrown from a "new InitialContext" operation

The name server identified by the provider URL cannot be contacted to obtain the initial JNDI context. There are many possible causes for this problem, including:
- The host name or port in the provider URL is incorrect.
- The host name cannot be resolved into an IP address by the domain name server, or the IP address does not match the IP address which the server is actually running under.
- A firewall on the client or server is preventing the port specified in the provider URL from being used.

To correct this problem:
- Make sure the provider URL and the network configurations on the client and server machines are correct.
- Make sure the host name can be resolved into an IP address which can be reached by the client machine. You can do this using the ping command.
- If you are running a firewall, make sure that use of the port specified in the provider URL will be allowed.

Related tasks

Chapter 6, “Troubleshooting by task,” on page 233

Object request broker component troubleshooting tips

Enabling tracing for the Object Request Broker component

The object request broker (ORB) service is one of the WebSphere Application Server run time services. Tracing messages that are sent and received by the ORB is a useful starting point for troubleshooting the ORB service. You can selectively enable or disable tracing of ORB messages for each server in a WebSphere Application Server installation, and for each application client.

For instructions on how to set trace controls so that tracing occurs for the ORB subcomponent, see Setting trace controls for IBM service.

Log files and messages associated with Object Request Broker

For a summary of how messages get routed in WebSphere Application Server, see Message routing.

Java packages containing the Object Request Broker service

The ORB service resides in the following Java packages:
- com.ibm.com.CORBA.*
- com.ibm.rmi.*
- com.ibm.ws.orb.*
- com.ibm.ws.orbimpl.*
- com.ibm.ws390.orb.*
- org.omg.CORBA.*
- javax.rmi.CORBA.*

JAR files that contain the previously mentioned packages include:
- *java_install_dir*/jre/lib/ext/ibmorb.jar
- *java_install_dir*/jre/lib/ext/iwsorbutil.jar
- *install_dir*/lib/iwsorb.jar
- *install_dir*/lib/runtimews390.jar
- *install_dir*/lib/bootstrap.jar

Tools used with Object Request Broker

The tools used to compile Java remote interfaces to generate language bindings used by the ORB at runtime reside in the following Java packages:
- com.ibm.tools.rmic.*
- com.ibm.idl.*

The JAR file that contains the packages is *install_dir*/java/lib/ibmtools.jar.

Object Request Broker properties

The ORB service requires a number of ORB properties for correct operation. It is not necessary for most users to modify these properties, and only the system administrator should modify them when required. Consult IBM Support personnel for assistance. The properties reside in the orb.properties file, located at *install_dir*/java/jre/lib/orb.properties.

If none of these steps fixes your problem, check to see if the problem has been identified and documented by looking at the available online support (hints and tips, technotes, and fixes).

For current information available from IBM Support on known problems and their resolution, see the IBM Support page.

IBM Support has documents that can save you time gathering information needed to resolve this problem. Before opening a PMR, see the IBM Support page.

Related tasks

Chapter 6, "Troubleshooting by task," on page 233

Related reference

Troubleshooting installation problems

Messaging component troubleshooting tips

Problems deploying or executing applications which use the WebSphere Application Server messaging capabilities

If you are having problems deploying or executing applications which use the WebSphere Application Server messaging capabilities, review these articles in the WebSphere Application Server information center:

- "Troubleshooting WebSphere messaging" on page 342
- "Troubleshooting message-driven beans" on page 346
- Troubleshooting service integration technologies

On a Linux platform, embedded messaging does not get removed after uninstalling the product

On a Linux platform, you may find that embedded messaging is not removed when you uninstall the product silently using the **SetRetainMQToTrue.active**=false option.

To prevent this problem, either issue the uninstall command from a local machine, or use ssh instead of telnet to logon to the machine where you issue the uninstall command.

To correct this problem after logging in, issue the following command for a real root login: su -

If none of these steps solves the problem, check to see if the problem has been identified and documented using the links in Diagnosing and fixing problems: Resources for learning.

For current information available from IBM Support on known problems and their resolution, see the IBM Support page.

IBM Support has documents that can save you time gathering information needed to resolve this problem. Before opening a PMR, see the IBM Support page.

Related tasks

Chapter 6, "Troubleshooting by task," on page 233

Related reference

Troubleshooting installation problems

Chapter 2. How do I troubleshoot?

Legend for "How do I?..." links

Documentation	Show me	Tell me	Guide me	Teach me
Refer to the detailed steps and reference	Watch a brief multimedia demonstration	View the presentation for an overview	Be led through the console pages	Perform the tutorial with sample code
Approximate time: Varies	**Approximate time:** 3 to 5 minutes	**Approximate time:** 10 minutes+	**Approximate time:** 1/2 hour+	**Approximate time:** 1 hour+

Set traces, logs, and other controls

Use trace to obtain detailed information about the execution of product components, including application servers, clients, and other processes in the environment.

Documentation Show me Tell me

- Console
- Scripting

Detect hung threads

A common error in J2EE applications is a hung thread. The hang detection option is turned on by default. You can configure a hang detection policy to accommodate your applications and environment so that potential hangs can be reported, providing earlier detection of failing servers. When a hung thread is detected, the product notifies you so that you can troubleshoot the problem.

Documentation

Detect product configuration file problems

Use an administrative console page to identify and view problems that exist in the current configuration.

Documentation

Troubleshoot problems that occur during a task

Troubleshoot problems that crop up during a main task such as migrating, installing, administering, securing, or deploying applications.

Documentation

Troubleshoot particular WebSphere application problems

Expand **Troubleshooting > Troubleshooting WebSphere applications** in the information center table of contents for information about troubleshooting specific application components and their administrative configurations (such as EJB container settings).

Debug WebSphere applications during development

In order to debug your application, you must use your application development tool (such as WebSphere Studio Application Developer) to create a Java project or a project with a Java nature. You must then import the program that you want to debug into the project.

Documentation

Add tracing and logging to your applications

Designers and developers of applications that run on the application server might find it useful to use Java logging for generating their application logging. This approach has advantages over simply adding System.out.println statements to your code.

Documentation

Collect troubleshooting details for IBM Support

WebSphere Application Server includes a number of troubleshooting tools that are designed to help you isolate the source of problems. Many of these tools are designed to generate information to be used by IBM Support, and their output might not be understandable by the customer.

Documentation

Using JSR47 for logging

The Logging API Specification (or Java logging), which is defined with Java Specification Request (JSR) 47, is the logging toolkit that is provided by the java.util.logging package. Java logging provides a standard logging API for your applications.

Documentation

Using JSR47 for logging: Writing a handler

There may be occasions when you only want to propagate log records to your own log handlers, rather than participate in integrated logging; in other words, to use a stand alone log handler.

Documentation

Using JSR47 for logging: Writing a formatter

A formatter formats events according to type. Handlers are associated with one or more formatters.

Documentation

Using JSR47 for logging: Writing a filter

A Filter provides optional, secondary control over what is logged, beyond the control provided by the level.

Documentation

Using Common Base Events for logging

WebSphere Application Server uses Common Base Events within its logging framework. Common Base Events can be created explicitly and logged via the Java logging API, or can be created implicitly by using the Java logging API directly. For Common Base Event creation, the application server environment provides a Common Base Event factory with a Content Handler that provides both runtime data and template data for Common Base Events.

Documentation

Creating Common Base Events

In cases where the events generated by the Java logging API are insufficient to describe the event that needs to be captured, Common Base Events can be created using the Common Base Event factory APIs. When you create a Common Base Event you can add data to the Common Base Event before it is logged. WebSphere Application Server is configured to use an event factory that automatically populates WebSphere Application Server specific information into the Common Base Events that it generates.

Documentation

Related concepts

"Contents of this section: Troubleshooting" on page 2

Chapter 1, "Overview and new features for troubleshooting," on page 1

Product overview and quick start
This topic provides shortcuts to information for obtaining a high level understanding of the product, then getting started quickly. Links include various conceptual overviews, descriptions of what is new, and how the

documentation is organized. ″How do I?″ links lead to task descriptions, including ShowMe demonstrations, tutorials, and presentations when they are available.

Chapter 3. Debugging applications

To debug your application, you must use your application development tool (such as Rational Application Developer) to create a Java project or a project with a Java nature. You must then import the program that you want to debug into the project. By following the steps below, you can import the WebSphere Application Server examples into a Java project.

There are two debugging styles available:

- **Step-by-step** debugging mode prompts you whenever the server calls a method on a Web object. A dialog lets you step into the method or skip it. In the dialog, you can turn off step-by-step mode when you are finished using it.
- **Breakpoints** debugging mode lets you debug specific parts of programs. Add breakpoints to the part of the code that you must debug and run the program until one of the breakpoints is encountered.

Breakpoints actually work with both styles of debugging. Step-by-step mode just lets you see which Web objects are being called without having to set up breakpoints ahead of time.

You need not import an entire program into your project. However, if you do not import all of your program into the project, some of the source might not compile. You can still debug the project. Most features of the debugger work, including breakpoints, stepping, and viewing and modifying variables. You must import any source that you want to set breakpoints in.

The inspect and display features in the source view do not work if the source has build errors. These features let you select an expression in the source view and evaluate it.

1. Create a Java Project by opening the New Project dialog.
2. Select **Java** from the left side of the dialog and **Java Project** in the right side of the dialog.
3. Click **Next** and then specify a name for the project (such as WASExamples).
4. Press **Finish** to create the project.
5. Select the new project, choose **File > Import > File System**, then **Next** to open the import file system dialog.
6. Select the directory Browse pushbutton and go to the following directory: *installation_root*\installedApps*node_name* \DefaultApplication.ear\DefaultWebApplication.war.
7. Select the checkbox next to DefaultWebApplication.war in the left side of the Import dialog and then click **Finish**. This will import the JavaServer Pages files and Java source for the examples into your project.
8. Add any JAR files needed to build to the Java Build Path. To do this, select **Properties** from the right-click menu. Choose the Java Build Path node and then select the Libraries tab. Use the Add External JARs pushbutton to add the following JAR files:
   - *installation_root*\installedApps*node_name*\DefaultApplication.ear\Increment.jar.

     Once you have added this JAR file, select it and use the **Attach Source** pushbutton to attach Increment.jar as the source - Increment.jar contains both the source and class files.

- *installation_root*\lib\j2ee.jar
- *installation_root*\lib\pagelist.jar
- *installation_root*\\lib\webcontainer.jar

Click **OK** when you have added all of the JARs.

9. You can set some breakpoints in the source at this time if you like, however, it is not necessary as step-by-step mode will prompt you whenever the server calls a method on a Web object. Step-by-step mode is explained in more detail below.
10. To start debugging, you need to start the WebSphere Application Server in debug mode and make note of the JVM debug port. The default value of the JVM debug port is 7777.
11. Once the server is started, switch to the debug perspective by selecting **Window > Open Perspective > Debug**. You can also enable the debug launch in the Java Perspective by choosing **Window > Customize Perspective** and selecting the **Debug** and **Launch** checkboxes in the **Other** category.
12. Select the workbench toolbar **Debug** pushbutton and then select **WebSphere Application Server Debug** from the list of launch configurations. Click the **New** pushbutton to create a new configuration.
13. Give your configuration a name and select the project to debug (your new WASExamples project). Change the port number if you did not start the server on the default port (7777).
14. Click **Debug** to start debugging.
15. Load one of the examples in your browser (for example, http://localhost:9080/hitcount).

Attaching WebSphere Studio Application Developer to a remote debug session

The steps below describe how to attach WebSphere Studio Application Developer to a remote debug session on WebSphere Application Server for z/OS. Remote debugging can prove useful when the program you are debugging behaves differently on a z/OS system than on your local system.

1. Enable the debug engine on WebSphere Application Server for z/OS using the adminstrative console. See “Debugging Service details” on page 230.
2. Import the java source code you wish to debug into WebSphere Studio Application Developer and set breakpoints. See the *WebSphere Studio Application Developer information center* at `http://publib.boulder.ibm.com/infocenter/wsphelp/` for instructions on setting breakpoints.
3. Open a WebSphere Studio Application Developer debug perspective and create a debug session configuration.
4. Attach WebSphere Studio Application Developer to the WebSphere Application Server for z/OS debug runtime. See ″Connecting to a remote VM with the remote Java application launch configuration″ in the *WebSphere Studio Application Developer information center* at `http://publib.boulder.ibm.com/infocenter/wsphelp/`
5. Execute the java code in WebSphere Application Server for z/OS to hit the breakpoints set in WebSphere Studio Application Developer.
6. Use WebSphere Studio Application Developer debugger controls and features to debug the application.

Related concepts

Chapter 3, “Debugging applications,” on page 57

Related information

WebSphere combined information center

Unit testing with DB2

The steps below describe how to setup a unit test environment that would allow you to develop and unit test code with DB2 z/OS to support Container Managed Persistence (CMP) development and access DB2 test data that resides on z/OS.

1. Configure DB2 Distributed Data Facility (DDF) on z/OS to allow remote TCP/IP connections from your WebSphere Studio Application Developer workstation. See the DB2 Information Center for information on DDF.
2. Install the DB2 Client Configuration Assistant on the workstation where WebSphere Studio Application Developer is installed. The DB2 Client Configuration Assistant is shipped with DB2.
3. Use the DB2 Client Configuration Assistant to define a DB2 alias.
4. Use the DB2 alias you defined to access the DB2 subsystem on z/OS using the DB2 Distributed Data Facility (DDF).

   Related information

   WebSphere combined information center

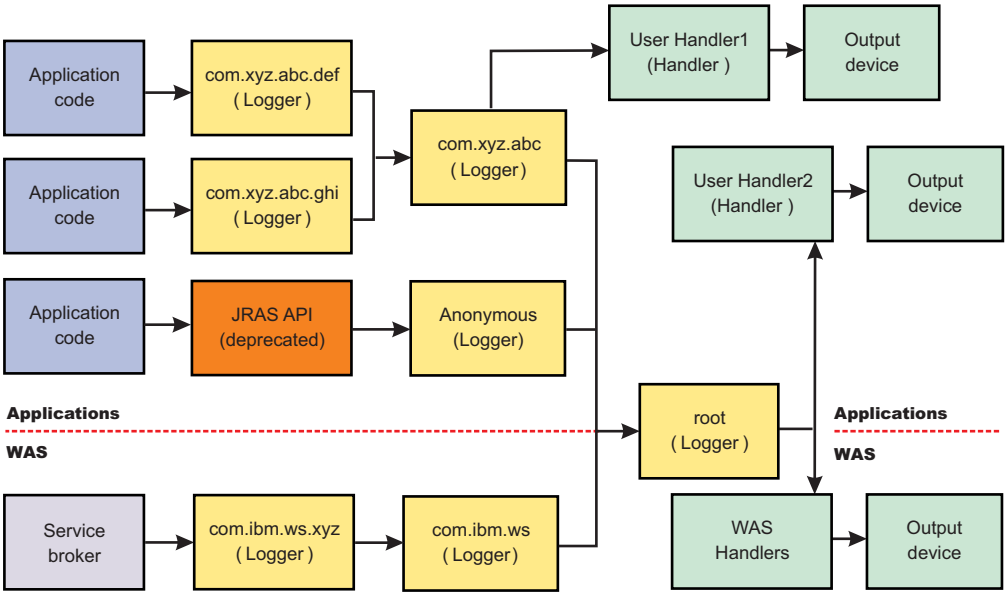
Chapter 4. Adding logging and tracing to your application

Deprecation: The JRas framework that is described in this information center is deprecated. However, you can achieve the same results using Java logging.

Designers and developers of applications that run with or under WebSphere Application Server, such as servlets, JavaServer Pages (JSP) files , enterprise beans, client applications, and their supporting classes, might find it useful to use Java logging for generating their application logging.

This approach has advantages over simply adding `System.out.println` statements to your code:

- Your messages are displayed in the WebSphere Application Server standard log files, using standard message format with additional data, such as a date and time stamp that are added automatically.
- You can more easily correlate problems and events in your own application to problems and events that are associated with WebSphere Application Server components.
- You can take advantage of the WebSphere Application Server log file management features.



Logging and tracing with Java logging

Developing, deploying and maintaining applications are complex tasks. When an application encounters an unexpected condition it might not be able to complete a requested operation. You might want the application to inform the administrator that the operation has failed and tell the administrator why the operation failed. This information enables the administrator to take the proper corrective action. Application developers might need to gather detailed information that relates to the path of a running application to determine the root cause of a failure that is due to a code bug. The facilities that are used for these purposes are typically referred to as *logging* and *tracing*.

Java logging is the logging toolkit that is provided by the java.util.logging package. Java logging provides a standard logging API for your applications.

Message logging (messages) and diagnostic trace (trace) are conceptually similar, but do have important differences. These differences are important for application developers to understand to use these tools properly. The following operational definitions of messages and trace are provided.

Message
A message entry is an informational record that is intended for end users, systems administrators, and support personnel to view. The text of the message must be clear, concise, and interpretable by an end user. Messages are typically localized, meaning they are displayed in the national language of the end user. Although the destination and lifetime of messages might be configurable, some level of message logging should be enabled in normal system operation. Message logging should be used judiciously because of performance considerations and the size of the message repository.

Trace A trace entry is an information record that is intended for service engineers or developers to use. As such, a trace record might be considerably more complex, verbose and detailed than a message entry. Localization support is typically not used for trace entries. Trace entries can be fairly inscrutable, understandable only by the appropriate developer or service personnel. It is assumed that trace entries are not written during normal runtime operation, but can be enabled as needed to gather diagnostic information.

To use Java logging, see the following topics:
• "Configuring logging properties using the administrative console" on page 66.

See the Java documentation for the java.util.logging class for a full description of the syntax and the construction of logging methods.

Loggers

Loggers are used by applications and runtime components to capture message and trace events. When situations occur that are significant either due to a change in state (for example when a server has completed startup) or because a potential problem has been detected (such as a timeout waiting for a resource) a message will be written to the logs. Trace events are logged in debugging scenarios, where a developer needs a clear view of what is occurring in each component to understand what might be going wrong. Logged events are often the only events available when a problem is first detected, and are used during both problem recovery and problem resolution.

Loggers are organized hierarchically. Each logger can have zero or more child loggers.

Loggers can be associated with a resource bundle. If specified, the resource bundle is used by the logger to localize messages logged to it. If the resource bundle is not specified, a logger will use the same resource bundle as its parent.

Loggers can be configured with a level. If specified the level is compared by the logger to incoming events -- events less severe than the level set for the logger are ignored by the logger. If the level is not specified, a logger will take on the level used by its parent. The default level for loggers is Level.INFO.

Loggers can have zero or more attached handlers. If supplied, all events logged to the logger will be passed to the attached handlers (for more details see “Log handlers”). Handlers write events to output destinations such as log files or network sockets. When a logger has finished passing a logged event to all of the handlers attached to that logger, the logger then passes the event to the handlers attached to the parents of the logger. This process will stop if a parent logger has been configured not to use its parent handlers. Handlers in WebSphere Application Server are attached to the root logger. Set the useParentHandlers logger property to false to prevent the logger from writing events to handlers that are higher up in the hierarchy.

Loggers can have a Filter. If supplied, the filter is invoked for each incoming event to tell the logger whether or not to ignore it.

Applications interact directly with loggers to log events. To obtain or create a logger, a call is made to the Logger.getLogger method with a name for the logger. Typically, the logger name is either the package qualified class name or the name of the package that the logger is used by. The hierarchical logger namespace is automatically created by using the dots in the logger name. For example, the logger ″com.ibm.websphere.ras″ has a parent logger named ″com.ibm.websphere″, which has a parent named ″com.ibm″, and so on. The parent at the top of the hierarchy is referred to as the root logger. This root logger is created during initialization. The root logger is the parent of the logger ″com″.

Loggers are structured in a hierarchy. Every logger except the root logger has one parent. Each logger can also have 0 or more children. A logger inherits log handlers, resource bundle names, and event filtering settings from its parent in the hierarchy. The logger hierarchy is managed by the LogManager function.

Loggers create log records. A log record is the container object for the data of an event. This object is used by filters, handlers, and formatters in the logging infrastructure.

The logger provides several sets of methods for generating log messages. Some log methods take only a level and enough information to construct a message. Other, more complex logp (log precise) methods support the caller in passing class name and method name attributes in addition to the level and message information. The logrb (log with ResourceBundle) methods add the capability of specifying a ResourceBundle as well as the level, message information, class name and method name. Using methods such as severe, warning, fine, finer, and finest you can log a message at a particular level. For a complete list of methods, see the java.util.logging documentation at http://java.sun.com/j2se/.

Related concepts

“Log handlers”

“Log levels” on page 64

“Log filters” on page 65

“Log formatters” on page 66
This article describes what a log formatter is.

Log handlers

Handlers write log record objects to an output device. Some examples of an output device are log files, sockets, and notification mechanisms.

Loggers can have zero or more attached handlers. All objects logged to the logger will be passed to the attached handlers, if handlers are supplied.

Handlers can be configured with a level. The handler compares the level specified in the logged object to the level specified for the handler. If the level of the logged object is less severe than the level set in the handler, the object is ignored by the handler. The default Level for handlers is ALL.

Handlers can have a filter. If a filter is supplied, the filter is invoked for each incoming object to tell the handler whether or not to ignore it.

Handlers can have a formatter. If a formatter is supplied, the formatter controls how the logged objects are formatted. For example, the formatter could decide to first include the timestamp, followed by a string representation of the level, followed by the message included in the logged object. The handler writes this formatted representation to the output device.

Both loggers and handlers can have levels and filters, and a logged object must pass all of these in order to be output. For example, you can set the logger level to FINE, but if the handler level is set at WARNING, then only WARNING level messages will appear in the output for that handler. Conversely, if your log handler is set to output all messages (level=All), but the logger level is set to WARNING, then the logger never sends messages lower than WARNING to the log handler.

Related concepts

"Loggers" on page 62

"Log levels"

"Log filters" on page 65

"Log formatters" on page 66
This article describes what a log formatter is.

Related tasks

"Configuring logging properties using the administrative console" on page 66
Use this task to browse or change the properties of Java logging.

Log levels

Levels control which events are processed by Java logging. WebSphere Application Server controls the levels of all loggers in the system. The level value is set from configuration data when the logger is created and can be changed at run time from the administrative console (see "Configuring logging properties using the administrative console" on page 66. If a level is not set in the configuration data, a level is obtained by proceeding up the hierarchy until a parent with a level value is found. You can also set a level for each handler to indicate which events are published to an output device. When you change the level for a logger in the administrative console, the change is propagated to the children of the logger.

Levels are cumulative; a logger can process logged objects at the level that has been set for the logger, and at all levels above the set level. Valid levels are:

Level	Content / Significance
Off	No events are logged.
Fatal	Task cannot continue and component cannot function.

Level	Content / Significance
Severe	Task cannot continue but component can still function
Warning	Potential error or impending error
Audit	Significant event affecting server state or resources
Info	General information outlining overall task progress
Config	Configuration change or status
Detail	General information detailing subtask progress
Fine	Trace information - General trace + method entry / exit / return values
Finer	Trace information - Detailed trace
Finest	Trace information - A more detailed trace - Includes all the detail needed to debug problems
All	All events are logged. If you have created custom levels, ″All″ would include your custom levels, and could provide a more detailed trace than ″finest″.

For instructions on how to set logging levels, see “Configuring logging properties using the administrative console” on page 66

Note: Trace information, which are events at levels Fine, Finer and Finest, can only be written to the trace log. Therefore, if you do not enable diagnostic trace, setting the log detail level to Fine, Finer or Finest will not have an effect on the data that is logged.

Related concepts

“Loggers” on page 62

“Log handlers” on page 63

“Log filters”

“Log formatters” on page 66
This article describes what a log formatter is.

Related tasks

“Configuring logging properties using the administrative console” on page 66
Use this task to browse or change the properties of Java logging.

Related reference

“Log level settings” on page 67
Use this page to configure and manage log level settings.

Log filters

A filter provides an optional, secondary control over what is logged, beyond the control that is provided by setting the level. Applications can apply a filter mechanism to control logging output through the logging APIs. An example of filter usage is to suppress all the events with a particular message key.

A filter is attached to a logger or log handler using the appropriate setFilter method. For a complete list of methods, see the java.util.logging documentation at http://java.sun.com/j2se/

Related concepts

"Loggers" on page 62

"Log handlers" on page 63

"Log levels" on page 64

"Log formatters"
This article describes what a log formatter is.

Log formatters

This article describes what a log formatter is.

Handlers may be configured with a formatter, which knows how to format log records. The event (represented by the log record object) is passed to the appropriate formatter by the handler. The formatter returns formatted output to the handler, which then writes it to the output device.

The formatter is responsible for rendering the event for output. This usually means the formatter uses the ResourceBundle specified in the event to look up the message in the appropriate language.

Formatters are attached to handlers using the setFormatter method.

You can find the java.util.logging documentation at http://java.sun.com/j2se/.

Related concepts

"Loggers" on page 62

"Log handlers" on page 63

"Log levels" on page 64

"Log filters" on page 65

Configuring logging properties using the administrative console

Use this task to browse or change the properties of Java logging.

Before applications can log diagnostic information, you need to specify how you want the server to handle log output, and what level of logging you require. Using the administrative console, you can enable or disable a particular log, specify where log files are stored and how many log files are kept, and specify a format for log output. You can also set a log level for each logger.

You can change the log configuration statically or dynamically. Static configuration changes affect applications when you start or restart the application server. Dynamic (or run-time) configuration changes apply immediately.

When a log is created, the level value for that log is set from the configuration data. If no configuration data is available for a particular log name, the level for that log is obtained from the parent of the log. If no configuration data exists for the parent log, the parent of that log is checked, and so on up the tree, until a log with a non-null Level value is found. When you change the level of a log, the change is propagated to the children of the log, which recursively propagate the change to their children, as necessary.

To configure loggers and log handlers for use by Java logging, use the administrative console to complete the following steps:

1. Set the logging levels for your logs.
   a. In the navigation pane, click **Servers > Application Servers**.
   b. Click the name of the server that you want to work with.
   c. Under Troubleshooting, click **Logging and tracing**.
   d. Click **Change Log Detail levels**.
   e. To make a static change to the configuration, click the **Configuration** tab. A list of well-known components, packages, and groups is displayed. To change the configuration dynamically, click the **Runtime** tab. The list of components, packages, and groups displays all the components that are currently registered on the running server.
   f. Select a component, package, or group to set a logging level. See "Log level settings" for a description of each level.
   g. Click **Apply**.
   h. Click **OK**.
2. To have static configuration changes take effect, stop then restart the Application Server.

Log level settings

Use this page to configure and manage log level settings.

Using log levels you can control which events are processed by Java logging. When you change the level for a logger, the change is propagated to the children of the logger.

Change Log Detail Levels

Specifies tracing details.

Enter a log detail level that specifies the components, packages, or groups to trace. The log detail level string must conform to the specific grammar described below. You can enter the log detail level string directly, or generate it using the graphical trace interface.

If you select the configuration tab, a list of well-known components, packages, and groups is displayed. This list might not be exhaustive.

If you select the runtime tab, the list of components, packages, and groups is displayed will all such components currently registered on the running server. This list is static; you will not see your application logger names on the runtime tab.

The format of the log detail level specification is:

```
<component> = <level>
```

where <component> is the component for which to set a log detail level, and <level> is one of the valid logger levels (off, fatal, severe, warning, audit, info, config, detail, fine, finer, finest, all). Separate multiple log detail level specifications with colons (:).

Components correspond to Java packages and classes, or to collections of Java packages. Use * as a wildcard to indicate components that include all classes in all packages contained by the specified component. For example:

* Specifies all traceable code running in the application server, including WebSphere Application Server system code and customer code.

com.ibm.ws.*
specifies all classes whose package name begins with com.ibm.ws.

com.ibm.ws.classloader.JarClassLoader
Specifies only the JarClassLoader class.

Note: An error can occur when setting a log detail level specification from the administrative console if selections are made from both the Groups and Components lists. In some cases, the selection made from one list is lost when adding a selection from the other list. To work around this problem, enter the desired log detail level specification directly into the log detail level entry field.

Select a component or group to set a log detail level. The table below lists the valid levels for application servers at WebSphere Application Server Version 6 and higher, and the valid logging and trace levels for earlier versions:

Version 6 Logging Level	Logging Level pre-Version 6	Trace Level pre-Version 6	Content / Significance
Off	Off	All disabled*	Logging is turned off. * In Version 6, a trace level of All disabled will turn off trace, but will not turn off logging. Logging will be enabled from the Info level.
Fatal	Fatal	-	Task cannot continue and component/ application/ server cannot function.
Severe	Error	-	Task cannot continue but component/ application/ server can still function. This level can also indicate an impending fatal error.
Warning	Warning	-	Potential error or impending error. This level can also indicate a progressive failure (for example, the potential leaking of resources).
Audit	Audit	-	Significant event affecting server state or resources
Info	Info	-	General information outlining overall task progress
Config	-	-	Configuration change or status

Detail	-	-	General information detailing subtask progress
Fine	-	Event	Trace information - General trace + method entry / exit / return values
Finer	-	Entry/Exit	Trace information - Detailed trace
Finest	-	Debug	Trace information - A more detailed trace that includes all the detail that is needed to debug problems
All		All enabled	All events are logged. If you create custom levels, ″All″ would include those levels, and could provide a more detailed trace than ″finest″.

When you enable a logging level in version 6, you are also enabling all of the levels above it. For example, if you set the logging level to warning on your version 6 application server, then warning, severe and fatal events will be processed.

Note: Trace information, which are events at levels Fine, Finer and Finest, can only be written to the trace log. Therefore, if you do not enable diagnostic trace, setting the log detail level to Fine, Finer or Finest will not have an effect on the data that is logged.

Related reference

“Enabling tracing and logging” on page 198

HTTP error and NCSA access log settings

To view this administrative console page, click **Application servers >** > *server name* **> HTTP error and NCSA access logging** .

Use this panel to configure an HTTP error log and NCSA access logs for an HTTP transport channel.

The HTTP error log logs HTTP errors. The level of error logging that occurs is dependent on the value selected for the Error log level field.

On a z/OS platform HTTP error and NCSA access logging is done at the servant level.

The NCSA access log contains a record of all inbound client requests that the HTTP transport channel handles. All of the messages contained in these logs are in NCSA format.

After you have configured the HTTP error log and the NCSA access logs, make sure the Enable NCSA access logging field is selected for the HTTP channels for

which you want logging to take place. To view the settings for an HTTP channel, click **Servers > Application Servers >***server* **> Web Container Transport Chains > HTTP Inbound Channel**.

Enable service at server startup: When selected, either an NCSA access log or an HTTP error log, or both will be initialized when the server is started.

Enable access logging: When selected, a record of inbound client requests that the HTTP transport channel handles is kept in the NCSA access log.

Access log file path: Specifies the directory path and name of the NCSA access log. Standard variable substitutions, such as $(SERVER_LOG_ROOT), can be used when specifying the directory path.

Access log maximum size: Specifies the maximum size, in megabytes, of the NCSA access log file. When this size is reached, an archive log named *logfile_name*.1 is created. However, every subsequent time that the original log file overflows this archive file is overwritten with the most current version of the original log file.

NCSA access log format: Specifies the NCSA format is used when logging client access information. If Common is selected, the log entries contain the requested resource and a few other pieces of information, but does not contain referral, user agent, or cookie information. If Combined in selected, referral, user agent, and or cookie information is included.

Enable error logging: When selected, HTTP errors that occur while the HTTP channel processes client requests are recorded in the HTTP error log.

Error log file path: Specifies the directory path and name of the HTTP error log. Standard variable substitutions, such as $(SERVER_LOG_ROOT), can be used when specifying the directory path.

Error log maximum size: Specifies the maximum size, in megabytes, of the HTTP error log file. When this size is reached, an archive log named *logfile_name*.1 is created. However, every subsequent time that the original log file overflows this archive file is overwritten with the most current version of the original log file.

Error log level:

Specifies the type of error messages that are included in the HTTP error log.

You can select:

Critical
Only critical failures that stop the Application Server from functioning properly are logged.

Error Errors in responses to clients are logged. These errors require Application Server administrator intervention if they are caused by server configuration settings.

Warning
Information on general errors, such as socket exceptions, that occur while handling client requests are logged. These errors do not typically require Application Server administrator intervention.

Information
The status of the various tasks performed while handling client requests is logged.

Debug
More verbose task status information is logged. This level of logging is not intended to replace RAS logging for debugging problems, but does provide a steady status report on the progress of individual client requests. If this level of logging is selected, you must specify a large enough log file size in the Error log maximum size field to contain all of the information that is logged.

Configuring logging properties for an application

The logger.properties file allows you to set logger attributes for specific loggers. The properties file is loaded the first time the method `Logger.getLogger(loggername)` is called within an application.

When an application calls the `Logger.getLogger` method for the first time, all the available logger properties files are loaded. Applications can provide logger.properties files in:

- the META-INF directory of the JAR file for the application
- directories included in the class path of application module
- directories included in the application class path

The properties file contains two categories of parameters - Logger control and Logger data:

- Logger control information
  - minimum localization level - minimum LogRecord level for which localization will be attempted
  - group - logical group that this component belongs to
  - eventfactory - The Common Base Event template file to use with the event factory. Note that the naming convention for this template is the fully qualified component name, with a file extension of ".event.xml". For example, a template that applies to package com.ibm.compXYZ would be called com.ibm.compXYZ.event.xml
- Logger data information
  - product name
  - organization name
  - component name
  - extensions – additional properties

Sample logger.properties file

In the following sample, event factory com.ibm.xyz.MyEventFactory will be used by any loggers in the com.ibm.websphere.abc package or any sub-packages which do not override this value in their own configuration file.

```
com.ibm.websphere.abc.eventfactory=com.ibm.xyz.MyEventFactory
```

Sample security

Purpose

The sample security policy that follows grants access to the file system and runtime classes. Include this security policy, with the entry `permission java.util.logging.LoggingPermission "control"`, in the META-INF directory of your application if you want to allow the application to programmatically alter controlled properties of loggers and handlers.

```
//////////////////////////////////////////////
//
// WebSphere Application Server Security Policy
//
//////////////////////////////////////////////

////////////////////////////////////////////////////////////////////
// Allow all access to the file system and runtime classes
////////////////////////////////////////////////////////////////////
grant codeBase "file:${application}" {
   permission java.util.logging.LoggingPermission "control";
};
```

Using loggers in an application

This article describes how to use Java logging within an application.

To instrument an application using Java logging, perform the following steps:

1. Create the necessary handler, formatter, and filter classes, if you need your own log files.
2. If localized messages will be used by the application, create a resource bundle as described in "Creating log resource bundles and message files" on page 77.
3. In the application code, get a reference to a logger instance as described in "Using a logger."
4. Insert the appropriate message and trace logging statements in the application as described in "Using a logger."

Using a logger

There are various ways in which to log messages or trace using Java logging. The following guidelines will help you to use Java logging to log messages and add tracing:

1. Use level WsLevel and above for messages, and lower levels for Trace. The WebSphere Application Server Extension API (the com.ibm.websphere.logging package) contains the WsLevel class.
   - For messages use:

     ```
     WsLevel.FATAL
     Level.SEVERE
     Level.WARNING
     WsLevel.AUDIT
     Level.INFO
     Level.CONFIG
     WsLevel.DETAIL
     ```

   - For trace use:

     ```
     Level.FINE
     Level.FINER
     Level.FINEST
     ```

2. Use the logp method instead of log or logrb since logp accepts parameters for class name and method name. The log and logrb methods will generally try to infer this information, but the performance penalty is prohibitive.
3. Avoid using the logrb method since this leads to inefficient caching of resource bundles which results in poor performance.
4. Use the isLoggable method to avoid creating data for a logging call that will not get logged. For example:

```
if (logger.isLoggable(Level.FINEST)) {
  String s = dumpComponentState(); // some expensive to compute method
  logger.logp(Level.FINEST, className, methodName, "componentX state
dump:\n{0}", s);
 }
```

The following sample applies to localized messages:

```
// note - generally avoid use of FINE, FINER, FINEST levels for
// messages to be consistent with WebSphere Application Server

String componentName = "com.ibm.websphere.componentX";
String resourceBundleName = "com.ibm.websphere.componentX.Messages";
Logger logger = Logger.getLogger(componentName, resourceBundleName);

// "Convenience" methods - not generally recommended due to lack of class
/ method names
//   - can't specify message substitution parameters
//   - can't specify class/method names
if (logger.isLoggable(Level.SEVERE))
 logger.severe("MSG_KEY_01");

if (logger.isLoggable(Level.WARNING))
 logger.warning("MSG_KEY_01");

if (logger.isLoggable(Level.INFO))
 logger.info("MSG_KEY_01");

if (logger.isLoggable(Level.CONFIG))
 logger.config("MSG_KEY_01");


// "log" methods - not generally recommended due to lack of class / method
names
//   - enable use of WebSphere Application Server specific levels
//   - enable use of message substitution parameters
//   - can't specify class/method names
if (logger.isLoggable(WsLevel.FATAL))
 logger.log(WsLevel.FATAL, "MSG_KEY_01", "parameter 1");

if (logger.isLoggable(Level.SEVERE))
 logger.log(Level.SEVERE, "MSG_KEY_01", "parameter 1");

if (logger.isLoggable(Level.WARNING))
 logger.log(Level.WARNING, "MSG_KEY_01", "parameter 1");

if (logger.isLoggable(WsLevel.AUDIT))
 logger.log(WsLevel.AUDIT, "MSG_KEY_01", "parameter 1");

if (logger.isLoggable(Level.INFO))
 logger.log(Level.INFO, "MSG_KEY_01", "parameter 1");

if (logger.isLoggable(Level.CONFIG))
 logger.log(Level.CONFIG, "MSG_KEY_01", "parameter 1");

if (logger.isLoggable(WsLevel.DETAIL))
 logger.log(WsLevel.DETAIL, "MSG_KEY_01", "parameter 1");
```

```
// "logp" methods - the recommended way to log
//   - enable use of WebSphere Application Server specific levels
//   - enable use of message substitution parameters
//   - enable use of class/method names
if (logger.isLoggable(WsLevel.FATAL))
 logger.logp(WsLevel.FATAL, className, methodName, "MSG_KEY_01",
"parameter 1");

if (logger.isLoggable(Level.SEVERE))
 logger.logp(Level.SEVERE, className, methodName, "MSG_KEY_01",
"parameter 1");

if (logger.isLoggable(Level.WARNING))
 logger.logp(Level.WARNING, className, methodName, "MSG_KEY_01",
"parameter 1");

if (logger.isLoggable(WsLevel.AUDIT))
 logger.logp(WsLevel.AUDIT, className, methodName, "MSG_KEY_01",
"parameter 1");

if (logger.isLoggable(Level.INFO))
 logger.logp(Level.INFO, className, methodName, "MSG_KEY_01",
"parameter 1");

if (logger.isLoggable(Level.CONFIG))
 logger.logp(Level.CONFIG, className, methodName, "MSG_KEY_01",
"parameter 1");

if (logger.isLoggable(WsLevel.DETAIL))
 logger.logp(WsLevel.DETAIL, className, methodName, "MSG_KEY_01",
"parameter 1");


// "logrb" methods - not generally recommended due to diminished performance
of switching resource bundles on the fly
//   - enable use of WebSphere Application Server specific levels
//   - enable use of message substitution parameters
//   - enable use of class/method names
String resourceBundleNameSpecial =
"com.ibm.websphere.componentX.MessagesSpecial";

if (logger.isLoggable(WsLevel.FATAL))
 logger.logrb(WsLevel.FATAL, className, methodName, resourceBundleNameSpecial,
"MSG_KEY_01", "parameter 1");

if (logger.isLoggable(Level.SEVERE))
 logger.logrb(Level.SEVERE, className, methodName, resourceBundleNameSpecial,
"MSG_KEY_01", "parameter 1");

if (logger.isLoggable(Level.WARNING))
 logger.logrb(Level.WARNING, className, methodName, resourceBundleNameSpecial,
"MSG_KEY_01", "parameter 1");

if (logger.isLoggable(WsLevel.AUDIT))
 logger.logrb(WsLevel.AUDIT, className, methodName, resourceBundleNameSpecial,
"MSG_KEY_01", "parameter 1");

if (logger.isLoggable(Level.INFO))
 logger.logrb(Level.INFO, className, methodName, resourceBundleNameSpecial,
"MSG_KEY_01", "parameter 1");

if (logger.isLoggable(Level.CONFIG))
 logger.logrb(Level.CONFIG, className, methodName, resourceBundleNameSpecial,
"MSG_KEY_01", "parameter 1");
```

```
if (logger.isLoggable(WsLevel.DETAIL))
 logger.logrb(WsLevel.DETAIL, className, methodName, resourceBundleNameSpecial,
"MSG_KEY_01", "parameter 1");
```

For trace, (or content that is not localized), the following sample applies:

```
// note - generally avoid use of FATAL, SEVERE, WARNING, AUDIT, INFO, CONFIG,
DETAIL levels for trace
// to be consistent with WebSphere Application Server

String componentName = "com.ibm.websphere.componentX";
Logger logger = Logger.getLogger(componentName);

// Entering / Exiting methods - recommended for non trivial methods
if (logger.isLoggable(Level.FINER))
 logger.entering(className, methodName);

if (logger.isLoggable(Level.FINER))
 logger.entering(className, methodName, "method param1");

if (logger.isLoggable(Level.FINER))
 logger.exiting(className, methodName);

if (logger.isLoggable(Level.FINER))
 logger.exiting(className, methodName, "method result");


// Throwing method - not generally recommended due to lack of message - use
logp with a throwable parameter instead
if (logger.isLoggable(Level.FINER))
 logger.throwing(className, methodName, throwable);


// "Convenience" methods - not generally recommended due to lack of class
/ method names
//   - can't specify message substitution parameters
//   - can't specify class/method names
if (logger.isLoggable(Level.FINE))
 logger.fine("This is my trace");

if (logger.isLoggable(Level.FINER))
 logger.finer("This is my trace");

if (logger.isLoggable(Level.FINEST))
 logger.finest("This is my trace");


// "log" methods - not generally recommended due to lack of class /
method names
//   - enable use of WebSphere Application Server specific levels
//   - enable use of message substitution parameters
//   - can't specify class/method names
if (logger.isLoggable(Level.FINE))
 logger.log(Level.FINE, "This is my trace", "parameter 1");

if (logger.isLoggable(Level.FINER))
 logger.log(Level.FINER, "This is my trace", "parameter 1");

if (logger.isLoggable(Level.FINEST))
 logger.log(Level.FINEST, "This is my trace", "parameter 1");


// "logp" methods - the recommended way to log
//   - enable use of WebSphere Application Server specific levels
//   - enable use of message substitution parameters
//   - enable use of class/method names
```

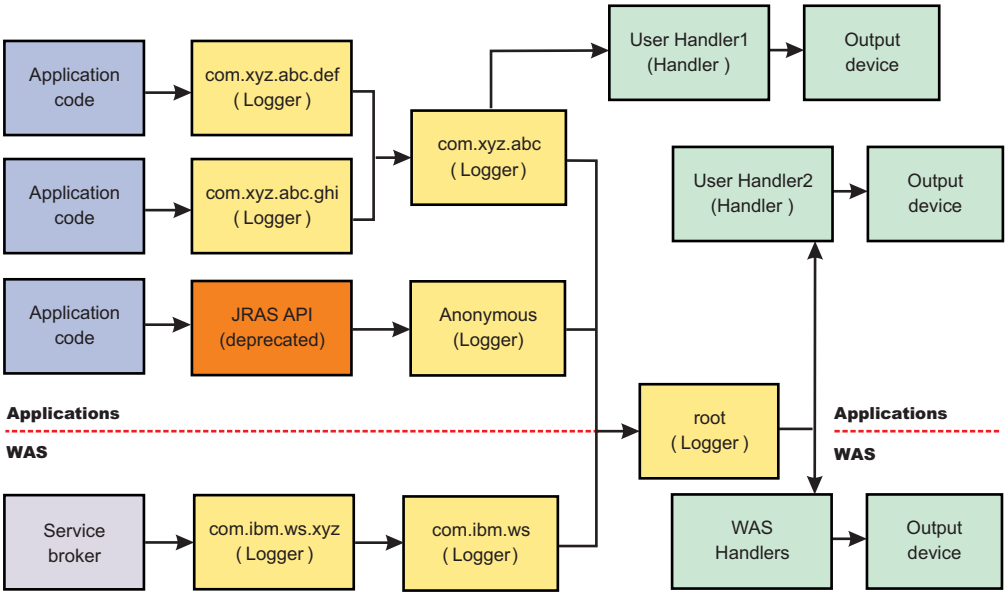
```
if (logger.isLoggable(Level.FINE))
 logger.logp(Level.FINE, className, methodName, "This is my trace",
"parameter 1");

if (logger.isLoggable(Level.FINER))
 logger.logp(Level.FINER, className, methodName, "This is my trace",
"parameter 1");

if (logger.isLoggable(Level.FINEST))
 logger.logp(Level.FINEST, className, methodName, "This is my trace",
"parameter 1");


// "logrb" methods - not applicable for trace logging since no localization
is involved
```

Understanding the logger hierarchy

WebSphere Application Server handlers are attached to the Java root logger in the logger hierarchy. As a result, any request from anywhere in the logger tree can be processed by WebSphere Application Server handlers. WebSphere Application You can configure the system to do the following tasks:

- Forward all application logging requests to the WebSphere Application Server handlers. This is the default behavior.
- Forward all application logging requests to your own custom handlers. To do this, set **useParentHandlers** to false on one of your custom loggers, and then attach your handlers to that logger.
- Forward all application logging requests to both WebSphere Application Server handlers, and your custom handlers, but do not forward WebSphere Application Server logging requests to your custom handlers. To do this, set **useParentHandlers** to true on one of your non-root custom loggers (true is the default setting) , and then attach your handlers to that logger.
- Forward all WebSphere Application Server logging requests to both WebSphere Application Server handlers, and your custom handlers. WebSphere Application Server logging requests are always forwarded to WebSphere Application Server handlers. To forward WebSphere Application Server requests to your custom handlers, attach your custom handlers to the Java root logger, so that they are at the same level in the hierarchy as the WebSphere Application Server handlers.

The example below shows how these requirements can be met using the Java logging infrastructure.



Creating log resource bundles and message files

Every method that accepts messages will localize those messages. The mechanism for providing localized messages is the Resource Bundle support provided by the IBM Developer Kit, Java Technology Edition. If you are not familiar with resource bundles as implemented by the Developer's Kit, you can get more information from various texts, or by reading the Javadoc for the `java.util.ResourceBundle`, `java.util.ListResourceBundle` and `java.util.PropertyResourceBundle` classes, as well as the `java.text.MessageFormat` class.

The `PropertyResourceBundle` is the preferred mechanism to use.

You can forward messages that are written to the internal WebSphere Application Server logs to other processes for display. For example, messages displayed on the administrator console, which can be running in a different location than the server process, can be localized using the *late binding* process. Late binding means that WebSphere Application Server does not localize messages when they are logged, but defers localization to the process that displays the message.

To properly localize the message, the displaying process must have access to the resource bundle where the message text is stored. This means that you must package the resource bundle separately from the application, and install it in a location where the viewing process can access it.

By default, the WebSphere Application Server runtime localizes all the messages when they are logged. This eliminates the need to pass a `.jar` file to the application, unless you need to localize in a different location. However, you can use the early binding technique to localize messages as they are logged. An application that uses early binding must localize the message before logging it. The application looks up the localized text in the resource bundle and formats the message. Use the early binding technique to package the application's resource bundles with the application.

To create a resource bundle, perform the following steps.

1. Create a text properties file that lists message keys and the corresponding messages. The properties file must have the following characteristics:
   - Each property in the file is terminated with a line-termination character.
   - If a line contains only white space, or if the first non-white space character of the line is the pound sign symbol (#) or exclamation mark (!), the line is ignored. The # and ! characters can therefore be used to put comments into the file.
   - Each line in the file, unless it is a comment or consists only of white space, denotes a single property. A backslash (\) is treated as the line-continuation character.
   - The syntax for a property file consists of a key, a separator, and an element. Valid separators include the equal sign (=), colon (:), and white space ( ).
   - The key consists of all characters on the line from the first non-white space character to the first separator. Separator characters can be included in the key by escaping them with a backslash (\), but doing this is not recommended, because escaping characters is error prone and confusing. It is instead recommended that you use a valid separator character that does not appear in any keys in the properties file.
   - White space after the key and separator is ignored until the first non-white space character is encountered. All characters remaining before the line-termination character define the element.

   See the Java documentation for the `java.util.Properties` class for a full description of the syntax and construction of properties files.
2. The file can then be translated into localized versions of the file with language-specific file names (for example, a file named `DefaultMessages.properties` can be translated into DefaultMessages_de.properties for German and DefaultMessages_ja.properties for Japanese).
3. When the translated resource bundles are available, put the bundle in a directory that is part of the application's classpath.
4. When a message logger is obtained from the log manager, it can be configured to use a particular resource bundle. Messages logged via the `Logger()` API will use this resource bundle when message localization is performed. At run time, the user's locale setting is used to determine the properties file from which to extract the message specified by a message key, thus ensuring that the message is delivered in the correct language.
5. If the message loggers `msg()` method is called, a resource bundle name must be explicitly provided.

The application locates the resource bundle based on the file's location relative to any directory in the classpath. For instance, if the property resource bundle named DefaultMessages.properties is located in the *baseDir*`/`*subDir1*`/`*subDir2*`/resources` directory and *baseDir* is in the class path, the name *subdir1*`.`*subdir2*`.resources.DefaultMessage` is passed to the message logger to identify the resource bundle.

Developing log resource bundles:
Resource bundle sample

You can create resource bundles in several ways. The best and easiest way is to create a properties file that supports a `PropertiesResourceBundle`. This sample shows how to create such a properties file.

For this sample, four localizable messages are provided. The properties file is created and the key-value pairs inserted into it. All the normal properties files

conventions and rules apply to this file. In addition, the creator must be aware of other restrictions imposed on the values by the Java MessageFormat class. For example, apostrophes must be ″escaped″ or they will cause a problem. Also avoid use of non-portable characters. WebSphere Application Server does not support usage of extended formatting conventions that the MessageFormat class supports, such as {1, date} or {0,number, integer}.

Assume that the base directory for the application that uses this resource bundle is ″baseDir″ and that this directory will be in the classpath. Assume that the properties file is stored in a subdirectory of baseDir that is not in the classpath (e.g. baseDir/subDir1/subDir2/resources). In order to allow the messages file to be resolved, the name subDir1.subDir2.resources.DefaultMessage is used to identify the PropertyResourceBundle and is passed to the message logger.

For this sample, the properties file is named DefaultMessages.properties.

```
# Contents of DefaultMessages.properties file
MSG_KEY_00=A message with no substitution parameters.
MSG_KEY_01=A message with one substitution parameter: parm1={0}
MSG_KEY_02=A message with two substitution parameters: parm1={0}, parm2 = {1}
MSG_KEY_03=A message with three parameter: parm1={0}, parm2 = {1}, parm3={2}
```

Once the file DefaultMessages.properties is created, the file can be sent to a translation center where the localized versions will be generated.

Creating a custom log handler

There may be occasions when you only want to propagate log records to your own log handlers, rather than participate in integrated logging. To use a stand alone log handler, set the ″useParentHandlers″ flag to false in your application.

The mechanism for creating a customer handler is the Handler class support provided by the IBM Developer Kit, Java Technology Edition. If you are not familiar with handlers as implemented by the Developer's Kit, you can get more information from various texts, or by reading the Javadoc for java.util.logging.

The following is a sample of a custom handler:

```
import java.io.FileOutputStream;
import java.io.PrintWriter;
import java.util.logging.Handler;
import java.util.logging.LogRecord;

/**
 * MyCustomHandler outputs contents to a specified file
 */
public class MyCustomHandler extends Handler {

 FileOutputStream fileOutputStream;
 PrintWriter printWriter;

 public MyCustomHandler(String filename) {
  super();

  // check input parameter
  if (filename == null)
   filename = "mylogfile.txt";

  try {
   // initialize the file
   fileOutputStream = new FileOutputStream(filename);
   printWriter = new PrintWriter(fileOutputStream);
  }
```

```
 catch (Exception e) {
  // implement exception handling...
 }
}

/* (non-Javadoc)
 * @see java.util.logging.Handler#publish(java.util.logging.LogRecord)
 */
public void publish(LogRecord record) {
 // ensure that this LogRecord should be logged by this Handler
 if (!isLoggable(record))
  return;

 // Output the formatted data to the file
 printWriter.println(getFormatter().format(record));
}

/* (non-Javadoc)
 * @see java.util.logging.Handler#flush()
 */
public void flush() {
 printWriter.flush();
}

/* (non-Javadoc)
 * @see java.util.logging.Handler#close()
 */
public void close() throws SecurityException {
 printWriter.close();
}
}
```

Creating a custom filter

A Filter provides optional, secondary control over what is logged, beyond the control provided by the level.

The mechanism for creating a customer filter is the Filter interface support provided by the IBM Developer Kit, Java Technology Edition. If you are not familiar with filters as implemented by the Developer's Kit, you can get more information from various texts, or by reading the Javadoc for java.util.logging.

The following is an example of a custom filter:

```
import java.util.Vector;
import java.util.logging.Filter;
import java.util.logging.LogRecord;

/**
 * MyCustomFilter rejects any LogRecords whose Level is not contained in the
 * configured list of Levels.
 */
public class MyCustomFilter implements Filter {

 private Vector acceptableLevels;

 public MyCustomFilter(Vector acceptableLevels) {
  super();
  this.acceptableLevels = acceptableLevels;
 }

 /* (non-Javadoc)
  * @see java.util.logging.Filter#isLoggable(java.util.logging.LogRecord)
  */
 public boolean isLoggable(LogRecord record) {
```

```
  return (acceptableLevels.contains(record.getLevel()));
 }

}
```

Creating a custom formatter

A formatter formats events. Handlers are associated with one or more formatters.

The mechanism for creating a customer formatter is the Formatter class support provided by the IBM Developer Kit, Java Technology Edition. If you are not familiar with formatters as implemented by the Developer's Kit, you can get more information from various texts, or by reading the Javadoc for java.util.logging.

The following is an example of a custom formatter:

```
import java.util.Date;
import java.util.logging.Formatter;
import java.util.logging.LogRecord;

/**
 * MyCustomFormatter formats the LogRecord as follows:
 * date   level   localized message with parameters
 */
public class MyCustomFormatter extends Formatter {

 public MyCustomFormatter() {
  super();
 }

 public String format(LogRecord record) {

  // Create a StringBuffer to contain the formatted record
  // start with the date.
  StringBuffer sb = new StringBuffer();

  // Get the date from the LogRecord and add it to the buffer
  Date date = new Date(record.getMillis());
  sb.append(date.toString());
  sb.append(" ");

  // Get the level name and add it to the buffer
  sb.append(record.getLevel().getName());
  sb.append(" ");

  // Get the formatted message (includes localization
  // and substitution of paramters) and add it to the buffer
  sb.append(formatMessage(record));

  return sb.toString();
 }
}
```

Using custom handlers, filters, and formatters

In some cases you may wish to have your own custom log files. Adding custom handlers, filters, and formatters enables you to customize your logging environment beyond what can be achieved by configuration of the default WebSphere Application Server logging infrastructure.

The following example demonstrates how to add a new handler to process requests to the *com.myCompany* subtree of loggers (see"Understanding the logger hierarchy" on page 76). The main method in this sample gives an example of how to use the newly configured logger.

```
import java.util.Vector;
import java.util.logging.Filter;
import java.util.logging.Formatter;
import java.util.logging.Handler;
import java.util.logging.Level;
import java.util.logging.Logger;

public class MyCustomLogging {

 public MyCustomLogging() {
  super();
 }

 public static void initializeLogging() {

  // Get the logger which we wish to attach a custom Handler to
  String defaultResourceBundleName = "com.myCompany.Messages";
  Logger logger = Logger.getLogger("com.myCompany", defaultResourceBundleName);

  // Set up a custom Handler (see MyCustomHandler example)
  Handler handler = new MyCustomHandler("MyOutputFile.log");

  // Set up a custom Filter (see MyCustomFilter example)
  Vector acceptableLevels = new Vector();
  acceptableLevels.add(Level.INFO);
  acceptableLevels.add(Level.SEVERE);
  Filter filter = new MyCustomFilter(acceptableLevels);

  // Set up a custom Formatter (see MyCustomFormatter example)
  Formatter formatter = new MyCustomFormatter();

  // Connect the filter and formatter to the handler
  handler.setFilter(filter);
  handler.setFormatter(formatter);

  // Connect the handler to the logger
  logger.addHandler(handler);

  // avoid sending events logged to com.myCompany showing up in WebSphere
  // Application Server logs
  logger.setUseParentHandlers(false);

 }

 public static void main(String[] args) {
  initializeLogging();

  Logger logger = Logger.getLogger("com.myCompany");

  logger.info("This is a test INFO message");
  logger.warning("This is a test WARNING message");
  logger.logp(Level.SEVERE, "MyCustomLogging", "main", "This is a test SEVERE message");
 }
}
```

When the above program is executed, the output of the program is written to the MyOutputFile.log file. The content of the log is in the expected log file, as controlled by the custom handler, and is formatted as defined by the custom formatter. The warning message has been filtered out, as specified by the configuration of the custom filter. The output is as follows:

```
C:\>type MyOutputFile.log
Sat Sep 04 11:21:19 EDT 2004 INFO This is a test INFO message
Sat Sep 04 11:21:19 EDT 2004 SEVERE This is a test SEVERE message
```

The Common Base Event in WebSphere Application Server

This topic describes how WebSphere Application Server takes advantage of the Common Base Events.

An application creates an event object whenever something happens that either should be recorded for later analysis or which may require additional work to be triggered. An *event* is a structured notification that reports information related to a situation. An event reports three kinds of information:

- The situation itself (what has happened)
- The identity of the affected component (for example, the server that has shut down)
- The identity of the component that is reporting the situation (which might be the same as the affected component)

The application creating the event object is called the event source. Event sources can use a common structure for the event. The accepted standard for such a structure is called the Common Base Event. The Common Base Event is a XML document defined as part of the Autonomic Computing initiative. The Common Base Event defines common fields, the values they can take and the exact meanings of these values.

The Common Base Event model is a standard defining a common representation of events that is intended for use by enterprise management and business applications. This standard, developed by the IBM Autonomic Computing Architecture Board, supports encoding of logging, tracing, management, and business events using a common XML-based format, making it possible to correlate different types of events that originate from different applications. For more information about the Common Base Event model, see the Common Base Event specification (*Canonical Situation Data Format: The Common Base Event V1.0.1*). The common event infrastructure currently supports version 1.0.1 of the specification.

The basic concept behind the Common Base Event model is the *situation*. A situation can be anything that happens anywhere in the computing infrastructure, such as a server shutdown, a disk-drive failure, or a failed user login. The Common Base Event model defines a set of standard situation types that accommodate most of the situations that might arise (for example, StartSituation and CreateSituation).

The Common Base Event should contain all of the information needed by the consumers to understand the event. This includes information about the runtime environment, the business environment and the instance of the application object that created the event.

For complete details on the Common Base Event format, see the XML schema included in the Common Base Event specification document, at ftp://www6.software.ibm.com/software/developer/library/ac-toolkitdg.pdf .

Related tasks

"Using the showlog command to view Common Base Events" on page 191

Types of problem determination events

This topic describes types of problem determination events.

Problem determination involves using multiple types of data, including at least two different classes of event data, log events and diagnostic events.

Log events, which are also referred to as message events, are typically emitted by components of a business application during normal deployment and operations. Log events may identify problems, but these events are also normally available and emitted while an application and its components are in production mode. The target audience for log and message events is users and administrators of the application and the components that make up the application. Log events are normally the only events available when a problem is first detected, and are typically used during both problem recovery and problem resolution.

Diagnostic events, which are commonly referred to as trace events, are used to capture internal diagnostic information about a component, and are usually not emitted or available during normal deployment and operations. The target audience for diagnostic events is the developers of the components that make up the business application. Diagnostic events are typically used when trying to resolve problems within a component, such as a software failure, but are sometimes used to diagnose other problems, especially when the information provided by the log events is not sufficient to resolve the problem. Diagnostic events are typically used when trying to resolve a problem.

Common Base Events are primarily used to represent log events.

The structure of the Common Base Event

A Common Base Event contains several structural elements. These are:

- Common header information
- Component Identification (both source and reporter)
- Situation information
- Message data
- Extended data
- Context data
- Associated events and association engine

Each of these structural elements has its own embedded elements and attributes.

The following table presents a summary of all fields in the Common Base Event and their usage requirements for problem determination events. It shows whether a particular element or attribute is required, recommended, optional, prohibited, or discouraged for log events, and the base specification.

Field Name	Log Events	Base Specification
Version	Required	Required
creationTime	Required	Required
severity	Required	Optional
Msg	Required	Optional
sourceComponentId*	Required	Required
sourceComponentId.location	Required	Required
sourceComponentId.locationType	Required	Required
sourceComponentId.component	Required	Required
sourceComponentId.subComponent	Required	Required
sourceComponentId.componentIdType	Required	Required

sourceComponentId.componentType	Required	Required
sourceComponentId.application	Recommended	Optional
sourceComponentId.instanceId	Recommended	Optional
sourceComponentId.processId	Recommended	Optional
sourceComponentId.threadId	Recommended	Optional
sourceComponentId.executionEnvironment	Optional	Optional
situation*	Required	Required
situation.categoryName	Required	Required
situation.situationType*	Required	Required
situation.situationType.reasoningScope	Required	Required
situation.situationType.(specific Situation Type elements)	Required	Required
msgDataElement*	Recommended	Optional
msgDataElement .msgId	Recommended	Optional
msgDataElement .msgIdType	Recommended	Optional
msgDataElement .msgCatalogId	Recommended	Optional
msgDataElement .msgCatalogTokens	Recommended	Optional
msgDataElement .msgCatalog	Recommended	Optional
msgDataElement .msgCatalogType	Recommended	Optional
msgDataElement .msgLocale	Recommended	Optional
extensionName	Recommended	Optional
localInstanceId	Optional	Optional
globalInstanceId	Optional	Optional
priority	Discouraged	Optional
repeatCount	Optional	Optional
elapsedTime	Optional	Optional
sequenceNumber	Optional	Optional
reporterComponentId*	Optional	Optional
reporterComponentId.location	Required (2)	Required (2)
reporterComponentId.locationType	Required (2)	Required (2)
reporterComponentId.component	Required (2)	Required (2)
reporterComponentId.subComponent	Required (2)	Required (2)
reporterComponentId.componentId Type	Required (2)	Required (2)
reporterComponentId.componentType	Required (2)	Required (2)
reporterComponentId.instanceId	Optional	Optional
reporterComponentId.processId	Optional	Optional
reporterComponentId.threadId	Optional	Optional
reporterComponentId.application	Optional	Optional

reporterComponentId.executionEnvironment	Optional	Optional
extendedDataElements*	Note 3	Optional
contextDataElements*	Note 4	Optional
associatedEvents*	Note 5	Optional

Note:

1. Items followed by an asterisk (*) are elements that consist of sub-elements and attributes. The fields in those elements are listed in the table directly following the parent element name.
2. Some of the elements are optional, but when included, they include sub-elements and attributes that are required. For example, reporterComponentId is of type ComponentIdentification. The component attribute in ComponentIdentification is required. Therefore, the reporterComponentId.component attribute is required, but only when the parent element (reporterComponentId) is included.
3. The extendedDataElements element can be included multiple times to supply extended data information. See the Extended Data section for more information on required and recommended extended data element values.
4. The contextDataElements element can be included multiple times to supply context data information.
5. The associatedEvents element can be included multiple times to supply correlation data. There are no recommended uses of this element for the producers of problem determination data, and it is in fact discouraged.

Common Header Information

The common header information in the Common Base Event includes the following information about an event:

- the version of this Common Base Event (version).
- the date and time when the event was generated (creationTime).
- the severity of the condition (situation) identified by the event (severity and priority).
- the type of event that was captured (extensionName).
- identifiers that can be used to quickly identify a specific event within a set of events (localInstanceId and globalInstanceId).
- information that allows a system to efficiently report multiple events of the same type, by consolidating those events into a single event (repeatCount and elapsedTime).
- sequence information that allows a system to order a set of events in other ways than time of capture (sequenceNumber).

The Common Base Event specification [CBE101] provides information on the required format of these fields and the Common Base Event Developer's Guide [CBEBASE] provides general usage guidelines. This article will provide additional information about how to format and use these fields for problem determination events, which should be used to clarify and extend the information provided in the other documents.

severity

All problem determination events must provide an indication as to the relative

severity of the condition (situation) being reported by providing appropriate values for the severity field in the Common Base Event. The severity field is required for problem determination events. This is more restrictive than the base specification for the Common Base Event, which lists this as an optional field because effective and efficient problem determination requires the ability to quickly identify the information needed to resolve a problem as well as prioritize the problems that need to be addressed. Typically the following values are used for problem determination events:

10	Information	Log information events (normal conditions, events supplied to clarify operations, for example, state transitions, operational changes). These events typically do not require administrator action or intervention.
20	Harmless	Similar to Information events, but used to capture 'audit' items, such as state transitions or operational changes. These events typically do not require administrator action or intervention.
30	Warning	Warnings typically represent recoverable errors, for example a failure that the system was able to correct. These events may require administrator action or intervention.
40	Minor	Minor errors describe events that represent an unrecoverable error within a component. The failure affects the component's ability to service some requests. The business application is able to continue to perform its normal functions, but its overall operation may be degraded. These events require administrator action or intervention to address the condition.

50	Critical	Critical errors describe events that represent an unrecoverable error within a component. The failure significantly affects the component's ability to service most requests. The business application is able to continue most (but not all) of its normal functions and its overall operation may be degraded. These events require administrator action or intervention to address the condition.
60	Fatal	Fatal errors describe events that represent an unrecoverable error within a component. The failure usually results in the complete failure of the component. The business application may be able to continue some normal functions, but its overall operation may be degraded. These events require administrator action or intervention to address the condition.

msg
Refer to "Message Data" on page 92 for information on this attribute.

priority
The usage of the priority field is discouraged for problem determination events. The severity field is typically used to communicate and evaluate the importance of problem determination events. When the priority field is used, it should only be used to enhance the information provided in severity field, i.e. prioritize events of the same severity.

extensionName
The extensionName field is used to communicate the 'type' of event being reported, for example, what general class of events is being reported. In many cases this field provides an indication of what additional data should be expected to be supplied with the event (for example, optional data values).

repeatCount
The repeatCount field is valid for problem determination events, but is not typically used or supplied by the event producers. This field is used for data reduction/consolidation by event management and analysis systems.

elapsedTime
The elapsedTime field is valid for problem determination events, but is not typically used or supplied by the event producers. This field is used for data reduction/consolidation by event management and analysis systems.

sequenceNumber
The sequenceNumber field is valid for problem determination events. It is typically only used by event producers when the granularity of the event time

stamp (the creationTime field) is not sufficient in ordering events. In other words, the sequenceNumber field is typically used to sequence events that have the same time stamp value.

Note: Event management and analysis systems may use the sequenceNumber field for a number of reasons, including providing alternative sequencing, not necessarily based on time stamp. The recommendations here are provided primarily for event producers.

Component Identification (Source and Reporter)

The component identification fields in the Common Base Event are used to indicate which component in the system is experiencing the condition described by the event (the sourceComponentId) and which component emitted the event (the reporterComponentId). Typically, these are the same component, in which case only the sourceComponentId is supplied. Some notes and scenarios on when these two elements in the Common Base Event should be used:

- the sourceComponentId is always used to identify the component experiencing the condition described by the event.
- the reporterComponentId is used to identify the component that actually produced and emitted the event. This element is typically only used within events emitted by a component that is monitoring another component and providing operational information regarding that component. The monitoring component (for example, a Tivoli agent or hardware device driver) is identified by the reporterComponentId and the component being monitored (for example, a monitored server or hardware device) is identified by the sourceComponentId.

  A potential misuse of the reporterComponentId is to identify a component that provides event conversion or management services for a component, for example, identifying an adapter that transforms the events captured by a component into Common Base Event format. The event conversion function is considered an extension of the component and should NOT be identified separately.

The information used to identify a component in the system is the same, regardless of whether it is the source component or reporter component:

location locationType	Component Location	Identifies the location of the component.
component componentType	Component Name	Identifies the asset name of the component, as well as the type of component.
subcomponent	Subcomponent Name	Identifies a specific part (i.e. subcomponent) of a component, e.g. a software module or hardware part.
application	Business Application Name	Identifies the business application or process the component is a part of and provides services for.
instanceId	Operational Instance	Identifies the operational instance of a component, i.e. the actual running instance of the component.

processId threadId	Operational Instance	Identifies the operational instance of a component within the context of a software operating system, i.e. the operating system process and thread running when the event was produced.
executionEnvironment	Operational Instance Component Location	Provides additional information about the operational instance of a component or its location by identifying the name of the environment hosting the operational instance of the component (e.g. the operating system name for a software application, the application server name for a J2EE application, or the hardware server type for a hardware part).

The Common Base Event specification [CBE101] provides information on the required format of these fields and the Common Base Event Developer's Guide [CBEBASE] provides general usage guidelines. This section will provide additional information about how to format and use some of these fields for problem determination events, which should be used to clarify and extend the information provided in the other documents.

Component

The component field in a problem determination event is used to identify the manageable asset associated with the event. A manageable asset is open for interpretation, but a good working definition is a manageable asset represents a hardware or software component that can be separately obtained or developed, deployed, managed and serviced. Examples of typical component names are:

- IBM eServer xSeries model x330
- IBM WebSphere Application Server#5.1 (5.1 is the version number)
- Microsoft Windows 2000
- The name of an internally developed software application for a component

subComponent

The subcomponent field in a problem determination event identifies the specific part of a component associated with the event. The subcomponent name is typically not a manageable asset, but provides internal diagnostic information when diagnosing an internal defect within a component, i.e. 'What part failed?'. Examples of typical subcomponents and their names are:

- Intel Pentium processor within a server system ("Intel Pentium IV Processor")
- the enterprise bean container within a web application server ("enterprise bean container")
- the task manager within an operating system ("Linux Kernel Task Manager")

- the name of a Java class and method (“myclass.mycompany.com” or “myclass.mycompany.com.methodname()”).

The format of a subcomponent name is determined by the component, but it is recommended that the convention shown above for naming a Java class or the combination of a Java class and method is followed. Subcomponent is a required field in the Common Base Event.

componentIdType
The componentIdType field is required by the base Common Base Event specification, but provides minimal value for problem determination events. The only recommendation regarding this field for problem determination events is to discourage the use of the “application” value. The componentIdType field identifies the type of component; the application is identified by the application field.

application
The application field is listed as an optional value within the Common Base Event specification, but it should be provided within problem determination events whenever it is available. The only reason this is not a required field for problem determination events is there are instances where the issuing component may not be aware of the overall business application.

instanceId
The instanceId field is listed as an optional value within the Common Base Event specification, but it should be provided within problem determination events whenever it is available.

The instanceID should always be provided when a software component is being identified and should identify the operational instance of the component (for example, which operation instance of an installed software image is actually associated with the event). This value should also be provided for hardware components, but not all hardware components support the concept of operational instances.

The format of the supplied value is defined by the component, but must be a value that can be used by an analysis system (either human or programmatic) to identify the specific running instance of the identified component. Examples include:

- **cell\node\server** name for the IBM WebSphere Application Server
- **deployed EAR file name** for a Java enterprise bean
- **serial number** for a hardware processor

processId
The processId field is listed as an optional value within the Common Base Event specification, but it should be provided for problem determination events whenever it is available and applicable. It should always be provided for software-generated events, and should identify the operating system process associated with the component identified in the event. The format of the thread ID should match the format of the operating system (or other execution environment, such as a Java Virtual Machine). This field is typically not applicable or used for events emitted by hardware (for example, firmware).

threadId
The threadId field is listed as an optional value within the Common Base Event specification, but it should be provided for problem determination events whenever it is available and applicable. It should always be provided for software-generated events, and should identify the operating system thread that was active when the event was detected or issued. A notable exception to

this recommendation is some operating systems or execution environments do not support threads. The format of the thread ID should match the format of the operating system (or other execution environment, such as a Java Virtual Machine). This field is typically not applicable or used for events emitted by hardware (for example, firmware).

executionEnvironment

The executionEnvironment field, when used, should identify the immediate execution environment used by the component being identified. Some examples are:

- the operating system name when the component is a native software application.
- the operating system/Java Virtual Machine name when the component is a Java J2SE application.
- the web server name when the component is a servlet.
- the portal server name when the component is a portlet.
- the application server name when the component is an enterprise bean.

The Common Base Event specification [CBE101] provides information on the required format of these fields and the Common Base Event Developer's Guide [CBEBASE] provides general usage guidelines.

Situation Information

The situation information is used to classify the condition being reported by an event into a common set of situations. The Common Base Event specification [CBE101] provides information on the set of situations defined for the Common Base Event, along with the values and formats used to describe these situations. The Common Base Event Developer's Guide [CBEBASE] provides general usage guidelines.

There are a few recommendations for situation information for problem determination events, such as:

- Whenever possible, use the situation categorizations and qualifiers described in the base Common Base Event specification. Avoid using your own situation definitions as much as possible.
- Not all messages and logs can be classified using the situation definitions supplied in the base Common Base Event specification. You can use the OtherSituation categorization to provide your own situation information, but the recommended course of action for problem determination events is to use the ReportSituation categorization, with reportCategory=Log.
- Warning events can be confusing. A warning event (i.e. an event with severity=warning) typically indicates a recoverable failure, but the situation settings can be interpreted as unrecoverable failures (e.g. ConnectSituation, successDisposition=UNSUCCESSFUL). The appropriate situation categorization should always be used and the severity setting will indicate the severity of the situation, i.e. whether the component recovered from the failure.
- The recommended setting for the reasoningScope value is EXTERNAL for all message events.

Message Data

All problem determination Common Base Events must provide human readable text describing the specific event being reported within the msg field of the Common Base Event. The text associated with events representing actual messages

or log entries is expected to be internationalized (in other words, translated and localized). We strongly recommend including the msgDataElement element in the Common Base Event whenever internationalized text is provided in the event. This element provides information about how the message text was created and how it should be interpreted, and is particularly invaluable when trying to interpret the event programmatically or when trying to interpret the message independent of the locale or language used to format the message text.

Note: Readers of this section of the document should be familiar with the concepts associated with creating internationalized messages (in other words, messages that are translated and localized). A good source of education on these concepts is provided by the documentation associated with internationalization of Java information and the usage of resource bundles within the Java language.

The msgDataElement element in the Common Base Event includes the following information about the text (i.e. the value of the msg field) provided with an event:

- The locale of the supplied message text, which identifies how the locale-independent fields within the message were formatted, as well as the language of the message (msgLocale).
- A locale-independent identifier associated with the message that can be used to interpret the message independent of the message language, message locale, and how the message was formatted (msgId and msgIdType).
- Information on how a translated message was created, including:
  - The identifier used to retrieve the message template (msgCatalogId).
  - The name and type of message catalog used to retrieve the message template (msgCatalog and msgCatalogType).
  - Any locale-independent information that was inserted into the message template to create the final message (msgCatalogTokens).

The Common Base Event specification [CBE101] provides information on the required format of these fields and the Common Base Event Developer's Guide [CBEBASE] provides general usage guidelines. This section will provide additional information about how to format and use these fields for problem determination events, which should be used to clarify and extend the information provided in the other documents.

msg

All message, log, and trace events must provide a human-readable message in the msg field of the Common Base Event. The msg field is required for problem determination events (both log events and diagnostic events). This is more restrictive than the base specification for the Common Base Event, which lists this as an optional field; because effective and efficient problem determination requires the ability to quickly identify the condition being reported by the event. The format and usage of this message is component-specific, but the following general guidelines should be followed:

- The message text supplied with messages and log events is expected to be internationalized.
- The locale of the supplied message text should be provided using the msgLocale field in the msgDataElement element of the Common Base Event.
- Additional information regarding the format and construction of internationalized messages should be provided whenever possible, using the msgDataElement element of the Common Base Event.

msgLocale
The message locale should be provided whenever message text is provided within the Common Base Event (as is the case with all problem determination events). The msgLocale field is listed as an optional value within the Common Base Event specification, but it should be provided within problem determination events whenever it is available. The only reason this is not a required field for problem determination events is there are instances where the locale information is not provided or available when formatting the Common Base Event.

msgId and msgIdType
Several companies include within internationalized message text a locale-independent identifier that can be used to interpret the condition being described by the message text, independent of the language of the message. For example, most messages issued by IBM software look like ″IEE890I WTO Buffers in console backup storage = 1024″, where a unique, locale-independent identifier ″IEE890I″ precedes the translated message text. This identifier provides a way to uniquely detect and identify a message independent of the location (and language) where it was issued, meaning it is invaluable for locale-independent and programmatic analysis. The msgId field is listed as an optional value within the Common Base Event specification, but it must be provided within problem determination events whenever this identifier is included in the message text. Likewise, the msgIdType field is listed as an optional value within the Common Base Event specification, but it must be provided within problem determination events whenever a value is supplied for msgId. These fields should not be supplied when the message text has not been translated or localized, for example, for trace events.

msgCatalogId
The msgCatalogId field is listed as an optional value within the Common Base Event specification, but it should be provided whenever the Common Base Event includes localized or translated message text, for example when providing problem determination events representing issued messages or log events. It is not a required field for problem determination events because not all problem determination events include translated message text, and there are cases where the value is not provided or available when formatting the Common Base Event. This field should not be supplied when the message text has not been translated or localized, for example, for trace events.

msgCatalogTokens
The msgCatalogTokens field is listed as an optional value within the Common Base Event specification, but it should be provided whenever the Common Base Event includes localized or translated message text, for example when providing problem determination events representing issued messages or log events. It is not a required field for problem determination events because not all problem determination events include translated message text, and there are cases where the value is not provided or available when formatting the Common Base Event. This value contains the list of locale independent values (message tokens) inserted into the localized message text when creating a translated message. It is very difficult to extract these values from a translated message without having knowledge of the translated message template used to create the message, meaning having these values separate from the message text is invaluable for locale-independent and programmatic analysis of the data supplied in the message text. This field should not be supplied when the message text has not been translated or localized, e.g. for trace events. Note: The Common Base Event provides several mechanisms for providing additional data about an event, including this field, extended data elements,

and extensions to the schema. The msgCatalogTokens field should always be used to supply the list of message tokens included in the message text associated with an event. These values can also be supplied in other parts of the Common Base Event as well, but they must be included in this field.

msgCatalog and msgCatalogType
The msgCatalog and msgCatalogType fields are listed as optional values within the Common Base Event specification, but they should be provided whenever the Common Base Event includes localized or translated message text, for example when providing problem determination events representing issued messages or log events. They are not required fields for problem determination events because not all problem determination events include translated message text, and there are cases where the values are not provided or available when formatting the Common Base Event. These fields should not be supplied when the message text has not been translated or localized, for example, for trace events.

Extended Data

The base information included in a Common Base Event may not be sufficient to represent all of the information captured by a component when creating a problem determination event. The Common Base Event provides several methods for including this additional data, including extending the Common Base Event schema or supplying one or more ExtendedDataElement elements within the Common Base Event. We recommend using the latter approach, ExtendedDataElement elements.

An ExtendedDataElement element is used to represent a single data item, and a Common Base Event can contain more than one of these elements (essentially one for each additional data item). A hint to the number and type of ExtendedDataElement elements is supplied by the extensionName value, but this is only a hint. The usage of the attributes in the ExtendedDataElement element for problem determination events is the same as those for any other Common Base Event.

Sample Common Base Event instance

The XML document shown below is an example of a Common Base Event (CBE) instance generated by a WebSphere Application Server application.

```
<CommonBaseEvent  creationTime="2004-09-18T04:03:28.484Z"
   globalInstanceId="myhost:1095479647062:1899"
   msg="WSVR0024I: Server server1 stopped"
   severity="10"
   version="1.0.1">

 ... several extendedDataElements for WebSphere Application Server internal use only ...

<sourceComponentId  component="com.ibm.ws.runtime.component.ServerCollaborator"
    componentIdType="Unknown"
    executionEnvironment="Windows 2000[x86]#5.0"
    instanceId="myhost\myhost\server1"
    location="myhost"
    locationType="Hostname"
    processId="1095479647062"
    subComponent="Unknown"
    threadId="Alarm : 0"
    componentType="http://www.ibm.com/namespaces/autonomic/WebSphereApplicationServer"/>

 <msgDataElement msgLocale="en_US">
  <msgCatalogTokens value="server1"/>
```

```
  <msgId>WSVR0024I< /msgId>
  <msgCatalogId>WSVR0024I< /msgCatalogId>
  <msgCatalog>com.ibm.ws.runtime.runtime< /msgCatalog>
 </msgDataElement>

 <situation categoryName="ReportSituation">
  <situationType xsi:type="ReportSituation" reasoningScope="EXTERNAL" reportCategory="LOG"/>
 </situation>

</CommonBaseEvent>
```

Note that there are a number of extendedDataElement elements in the actual XML which are used by WebSphere Application Server, but which are not for use by applications as they may change without notice in future releases.

The CommonBaseEvent element defines the CBE instance. This element has a set of attributes that are common for all CBEs. This includes the extensionName attribute that defines the type or class of the CBE instance, the creation time, severity and priority.

Nested within the CommonBaseEvent element are elements giving more detail about the situation. The first of these is the situation element. This is a standardized classification of the situation.

The CommonBaseEvent element also includes the sourceComponentId and the (optional) reporterComponentId elements. The sourceComponentId describes where the situation occurred; the reporterComponentId describes where the situation was detected. If the sourceComponentId and reporterComponentId are the same, the reporterComponentId is omitted.

The attributes of both the sourceComponentId and the reporterComponentId elements are the same. They identify the component's type, name, operating system and network location. The content of these attributes provides vertical correlation of the stack of IT resources active when the when the CBE was created.

Also included in the CommonBaseEvent element are contextDataElements that describe the context in which the situation occurred. This context correlates CBE instances that were part of the same piece of work. This is called horizontal correlation since an instance of a particular context type correlates events at the same level of abstraction (for example at the business level, or at the application level, or at the middleware level.)

Finally, there are the extended data elements. These contain additional data used to describe to situation. In this example, there is an extended data element added by WebSphere Application Server to describe the J2EE component that generated the CBE instance and some application data.

Sample Common Base Event template

Components that use the WebSphere Application Server event factory home can include a Common Base Event template XML file to provide data to populate Common Base Events. Template information is used by the content handler to fill in blanks in the Common Base Event when the Common Base Event complete method is called. Information that is already supplied in the event will not be overridden if the same field is supplied in the template.

The following is an example of a Common Base Event template:

```
<?xml version="1.0" encoding="UTF-8"?>

<TemplateEvent
  version="1.0.1"
  xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
  xsi:noNamespaceSchemaLocation="templateEvent.xsd">

 <CommonBaseEvent
  <sourceComponentId application="My Application" component="com.ibm.componentX"/>
      <extendedDataElements name="Sample ExtendedDataElement name" type="string">
   <values>Sample ExtendedDataElement value</values>
  </extendedDataElements>
   </CommonBaseEvent>

</TemplateEvent>
```

Component Identification for Problem Determination

This topic describes types of problem determination events.

A business application is made up of multiple components. A component can be made up of several internal subcomponents. Consistent application of these concepts is critical for effective problem determination of a business application; all of the parts of the application must use the same concepts and assumptions when creating and formatting events. The following definitions and examples should be used when creating Common Base Events for problem determination.

Business Application

A business application is the business logic and business data used to address a set of specific business requirements. A business application consists of several components of multiple types, combined in a unique manner by an enterprise, to provide the functions and resources needed to address those requirements. The primary creator and manager of a business application is the enterprise, and each enterprise or company creates unique business applications. Examples of business applications are the Payroll Application for the ACME Corporation and the Inventory Application for Spacely Sprockets.

Components

A business application is created and managed by the enterprise as a set of components. Components are deployable assets, developed either by the enterprise or a vendor, managed by the enterprise. A components may be created by the enterprise, typically for usage within a specific business application. For example, the ACME Corporation may create a set of enterprise beans to represent the business logic required by their Payroll Application. A component may also be an asset produced by a vendor and acquired by an enterprise. Examples of these components are hardware products, such as IBM eServers or Sun Solaris systems, or software products, such as IBM WebSphere Application Server, Oracle Database Servers.

Subcomponents

A specific component, depending on its complexity, may consist of several subcomponents. For example, the IBM WebSphere Application Server consists of many subcomponents, such as the enterprise bean container and the servlet engine. Subcomponent information is typically used only by the creator of the component to service the component, and as such are not separately deployable or manageable resources in the enterprise. The enterprise may deploy a change or update to a subcomponent, but only upon guidance from the component vendor and as part of the vendor's component. For example, a software fix for the enterprise bean container of the IBM WebSphere Application Server is packaged and deployed as a software update to the IBM

WebSphere Application Server. Replacement of the processor in an IBM eServer is deployed as a physical part, but only as a part of the original deployed component, the IBM eServer.

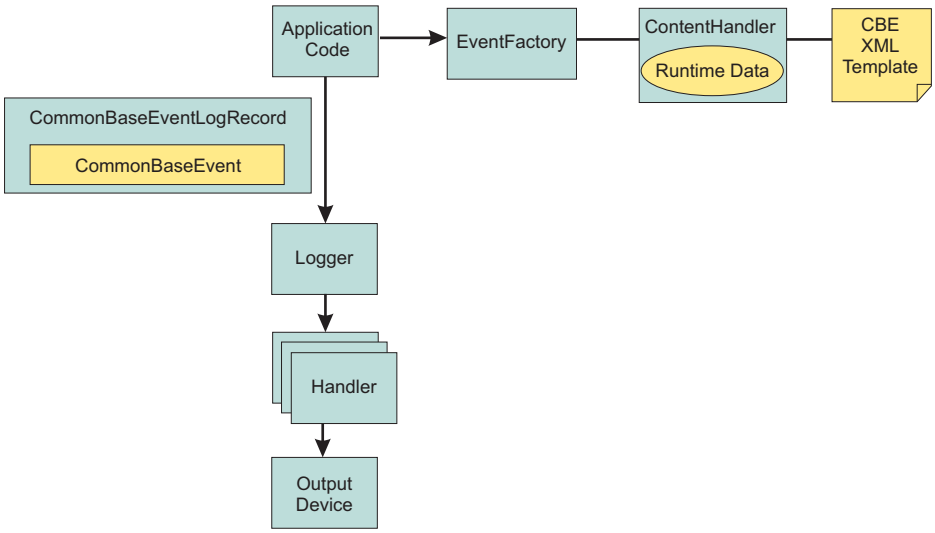
Logging Common Base Events in WebSphere Application Server

This topic describes how WebSphere Application Server takes advantage of the Common Base Events.

WebSphere Application Server uses Common Base Events within its logging framework. Common Base Events can be created explicitly and then logged via the Java logging API, or can be created implicitly by using the Java logging API directly. For Common Base Event creation, the application server environment provides a Common Base Event factory with a Content Handler that provides both runtime data and template data for Common Base Events.

Using the Common Base Event API with the Java Logging API to log Common Base Events

In cases where the events generated by the Java logging API are insufficient to describe the event that needs to be captured, Common Base Events can be created using the Common Base Event factory APIs. When you create a Common Base Event you can add data to the Common Base Event before it is logged. The following diagram illustrates how application code can create and log CommonBaseEvents:



The steps for generating a Common Base Event are as follows:

1. Application code invokes createCommonBaseEvent method on EventFactory to create a CommonBaseEvent.
2. Application code wraps CommonBaseEvent in a CommonBaseEventLogRecord, and adds event specific data.
3. Application code calls CommonBaseEvent's complete() method.
4. CommonBaseEvent invokes ContentHandler's completeEvent() method.
5. ContentHandler adds XML template data to CommonBaseEvent (including for example, the component name). Note that not all ContentHandlers support templates.

6. ContentHandler adds runtime data to CommonBaseEvent (including for example, the current thread name).
7. Application code passes CommonBaseEventLogRecord to Logger using Logger.log method.
8. Logger passes CommonBaseEventLogRecord to Handlers.
9. Handlers format data and write to output device.

WebSphere Application Server is configured to use an event factory that automatically populates WebSphere Application Server specific information into the Common Base Events that it generates. In general it is good practice to create events using the WebSphere Application Server default Common Base Event factory because this ensures consistency of Common Base Event content across events. However, other Common Base Event factories can be create and used. See "Configuring Common Base Events for an application" on page 102 for details on how to create and use custom event factories.

Related reference

"Using the Java Logging API to Generate and Log Common Base Events" on page 100

Common Base Event factory context:

The event factory context provides a service to look up event factory homes. The event factory context can be retrieved using a call to EventFactoryContext.getInstance(). Using this class, you can look up the event factory homes by name, and avoid the need to include the typed home in code. The EventFactoryHome must be located on the classpath to be found. The EventFactoryContext also stores an EventFactoryHome as a default, which can be obtained with a call to EventFactoryContext.getInstance().getEventFactoryHome().

In WebSphere Application Server, the EventFactoryContext is configured with a default EventFactoryHome which is associated to a ContentHandler capable of supplying both event template information, as well as WebSphere Application Server runtime default information.

More details can be found in the javadoc for org.eclipse.hyades.logging.events.cbe.EventFactory.

Common Base Event factory home:

Event Factory homes provide Event Factory instantiation based on a unique factory name. Event Factory home implementations are tightly coupled with content handlers which are used to populate Common Base Events with template or default data. Event Factory instances are maintained by the associated Event Factory home based on their unique name. For example, when application code requests a named Event Factory, the newly created Event Factory instance is returned and persisted for future requests for that named Event Factory. An abstract Event Factory home class provides the implementation for the APIs in the Event Factory home interface. Implementers extend the abstract Event Factory home class and implement the createContentHandler() API to create a typed content handler based on the type of the Event Factory home implementation.

In the WebSphere Application Server, the default Event Factory home obtained with a call to EventFactoryContext.getInstance().getEventFactoryHome() is associated with a ContentHandler capable of supplying both event template information, as well as WebSphere Application Server runtime default information.

More details can be found in the javadoc for org.eclipse.hyades.logging.events.cbe.EventFactoryHome at www.eclipse.org/hyades.

Common Base Event factory:

Event factories allow you to create Common Base Events and complete event properties using associated content handlers. Content handlers populate data into Common Base Events when the Common Base Event invokes the complete() method. All event properties set by the application code have priority over all properties specified by the content handler. Event factory implementations are tightly coupled with the content handler instance, which is associated with the event factory when the event factory is instantiated. Factory instances can only be retrieved from their associated event factory home. Event factory instances are retrieved and maintained based on unique names. Event factory names are hierarchal; they are represented using the standard Java dot-delimited name-space naming conventions.

More details can be found in the javadoc for org.eclipse.hyades.logging.events.cbe.EventFactory at www.eclipse.org/hyades.

Common Base Event content handler:

Content handlers populate data into Common Base Events when the Common Base Event complete() method is invoked. Content handlers can be associated with Common Base Event templates which provide default information to transfer into each Common Base Event. Content handlers may also provide any other information that is relevant to completing the population of the Common Base Event, such as runtime defaults deemed to be appropriate.

The use of content handlers is recommended to ensure consistency of field usage in the Common Base Event within a component or within a set of components sharing the same runtime. For example, some content handlers allow a template to be specified. If used consistently across a component, this would ensure that all events for that component would have the same template info filled in. Similarly, some content handlers can also supply runtime information to their associated Common Base Events. If consistently used throughout the entire runtime, this would ensure that all events use runtime data in a similar fashion.

The event factory home used in the WebSphere Application Server runtime is associated with a content handler that both reads from a template, and supplies runtime data. It is recommended that components use Event Factories obtained from this event factory home with their own templates, giving consistency between application events and server events.

More details can be found in the javadoc for org.eclipse.hyades.logging.events.cbe.ContentHandler at www.eclipse.org/hyades.

Using the Java Logging API to Generate and Log Common Base Events

In the WebSphere Application Server, the Java logging API (java.util.logging) automatically creates Common Base Events for events logged at level WsLevel.DETAIL or above (including WsLevel.DETAIL, Level.CONFIG, Level.INFO, WsLevel.AUDIT, Level.WARNING, Level.SEVERE, and WsLevel.FATAL). These Common Base Events are created using the event factory

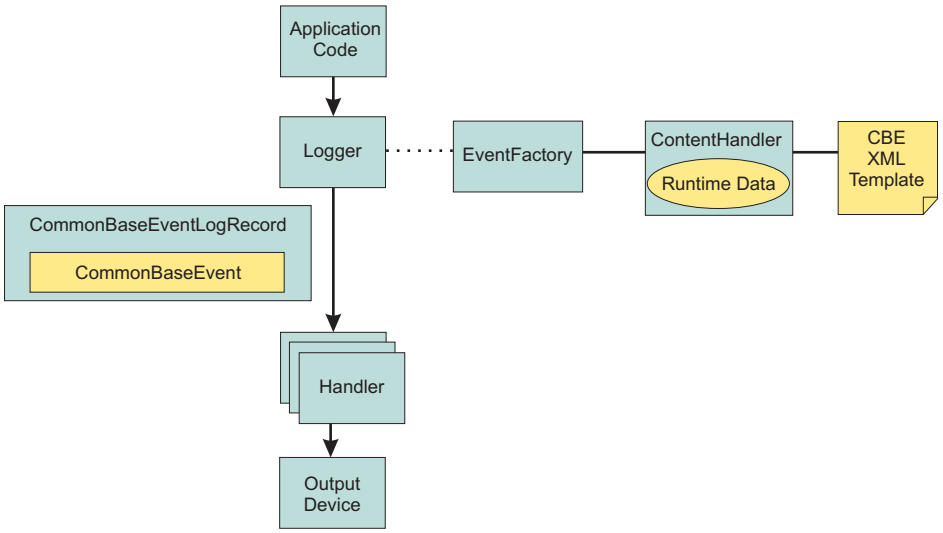
associated with the Logger to which the message was logged. If no event factory is specified, WebSphere Application Server uses a default event factory which will automatically fill in WebSphere Application Server specific information.

The java.util.logging.Logger class provides a variety of methods with which data can be logged. The WebSphere Application Server uses a special implementation of the Logger class that automatically creates Common Base Events for the following methods:

- config
- info
- warning
- severe
- log - all variants except log(LogRecord) when used with WsLevel.DETAIL or more severe levels
- logp - when used with WsLevel.DETAIL or more severe levels
- logrb - when used with WsLevel.DETAIL or more severe levels

WebSphere Application Server logger implementation is only used for named loggers (for example, loggers instantiated with calls such as Logger.getLogger("com.xyz.SomeLoggerName")). Loggers instantiated with calls to Logger.getAnonymousLogger() and Logger.getLogger(""), or Logger.global do not use the WebSphere Application Server implementation, and do not automatically create Common Base Events for logging requests made to them. LogRecords logged directly with Logger.log(LogRecord) are not automatically converted by WebSphere Application Server's Loggers into Common Base Events.

The following diagram illustrates how application code can create log CommonBaseEvents:



The Java logging API processing of named Loggers and message level events proceeds as follows:

1. Application code invokes named Logger (WsLevel.DETAIL or above) with event specific data.
2. Logger creates a CommonBaseEvent using createCommonBaseEvent method on EventFactory associated to the Logger (see Configuring Common Base Events for an application).

3. Logger creates a CommonBaseEvent using EventFactory associated to the Logger
4. Logger wraps CommonBaseEvent in a CommonBaseEventLogRecord, and adds event specific data.
5. Logger calls CommonBaseEvent's complete() method.
6. CommonBaseEvent invokes ContentHandler's completeEvent() method.
7. ContentHandler adds XML template data to CommonBaseEvent (including for example, the component name). Note that not all ContentHandlers support templates.
8. ContentHandler adds runtime data to CommonBaseEvent (including for example, the current thread name).
9. Logger passes CommonBaseEventLogRecord to Handlers.
10. Handlers format data and write to Output Device.

Configuring Common Base Events for an application

The logger.properties file allows you to set logger attributes for your component. The properties file is loaded the first time the method `Logger.getLogger(`*`loggername`*`)` is called within an application. The logger.properties file must be either on the WebSphere Application Server class path, or the context class path. See "Configuring logging properties for an application" on page 71 for details on the logger.properties file

To use the Common Base Events standard for events generated by your application, include the eventfactory property in your logger.properties file. The event factory property specifies the name of the Common Base Event template to use with the event factory. See "Sample Common Base Event template" on page 96 for details on the Common Base Event template. Note that the naming convention for this is the fully qualified component name with the extension ".event.xml". For example, a template that applies to package com.ibm.compXYZ would be called "com.ibm.compXYZ.event.xml"

Sample logger.properties file

In the following sample, event factory com.ibm.xyz.MyEventFactory will be used by any loggers in the com.ibm.websphere.abc package or any sub-packages which do not have their own configuration file.

```
com.ibm.websphere.abc.eventfactory=com.ibm.xyz.MyEventFactory
```

Related information

"Configuring logging properties for an application" on page 71

"Sample Common Base Event template" on page 96

"Log level settings" on page 67
Use this page to configure and manage log level settings.

Common Base Event content generated when using the WebSphere Application Server default event factory

When no other event factory is configured for a logger, the WebSphere Application Server uses its default event factory for creation of Common Base Events. The content handler associated with the default event factory populates fields as follows:

Field	Value	Notes
CommonBaseEvent. globalInstanceId	unique record id	only set if CommonBaseEvent.globalInstanceId was null before completeEvent() was called
CommonBaseEvent.msg	localized message based on MsgDataElement	only set if CommonBaseEvent.msg was null before completeEvent() was called
CommonBaseEvent.severity	set based on value of Level set on CommonBaseEventLogRecord if level >= Level.SEVERE set to 50 else if level >= Level.WARNING set to 30 default set to 10	only set if CommonBaseEvent.severity was null before completeEvent() was called
CommonBaseEvent. ComponentIdentification. component	set based on value of LoggerName set on CommonBaseEventLogRecord	only set if CommonBaseEvent.ComponentIdentification. component was null before completeEvent() was called
CommonBaseEvent. ComponentIdentification. componentIdType	″Unknown″	only set if CommonBaseEvent.ComponentIdentification. componentIdType was null before completeEvent() was called
CommonBaseEvent. ComponentIdentification. executionEnvironment	OSname[OSarch]#OSversion	only set if CommonBaseEvent.ComponentIdentification. executionEnvironment was null before completeEvent() was called
CommonBaseEvent. ComponentIdentification. instanceId	cellName\nodeName\ serverName	only set if CommonBaseEvent.ComponentIdentification. instanceId was null before completeEvent() was called only set in a server environment (ignored in a client application)
CommonBaseEvent. ComponentIdentification.location	hostname	only set if both CommonBaseEvent.ComponentIdentification. location and CommonBaseEvent.ComponentIdentification. locationType were null before completeEvent() was called
CommonBaseEvent. ComponentIdentification. locationType	″Hostname″	only set if both CommonBaseEvent.ComponentIdentification. location and CommonBaseEvent.ComponentIdentification. locationType were null before completeEvent() was called
CommonBaseEvent. ComponentIdentification. processId	internally generated representation of process number	only set if CommonBaseEvent.ComponentIdentification. processId was null before completeEvent() was called
CommonBaseEvent. ComponentIdentification. subComponent	set based on values of sourceClassName and sourceMethodName set on CommonBaseEventLogRecord sourceClassName. sourceMethodName	CommonBaseEvent.ComponentIdentification. subComponent was null before completeEvent() was called and both sourceClassName and sourceMethodName were set
CommonBaseEvent.Component Identification.threadId	set to value of JVM thread name	only set if CommonBaseEvent.ComponentIdentification. threadId was null before completeEvent() was called

Field	Value	Notes
CommonBaseEvent.ComponentIdentification.componentType	″http://www.ibm.com/namespaces/ autonomic/WebSphereApplication Server″	only set if CommonBaseEvent.ComponentIdentification.componentType was null before completeEvent() was called
CommonBaseEvent.MsgDataElement.msgLocale	set based on default locale of JVM	only set if CommonBaseEvent.msg was null before completeEvent() was called
CommonBaseEvent.Situation.categoryName	″ReportSituation″	only set if CommonBaseEvent.Situation was null before completeEvent() was called
CommonBaseEvent.Situation.situationType.type	″ReportSituation″	only set if CommonBaseEvent.Situation was null before completeEvent() was called
CommonBaseEvent.Situation.situationType.reasoningScope	″EXTERNAL″	only set if CommonBaseEvent.Situation was null before completeEvent() was called
CommonBaseEvent.Situation.situationType.reportCategory	″LOG″	only set if CommonBaseEvent.Situation was null before completeEvent() was called

Note that the sourceComponentIdentification is populated if no reporterComponentIdentification exists when completeEvent is invoked on the ContentHandler, otherwise the reporterComponentIdentification will be populated instead.

The same content handler that WebSphere Application Server uses for generating its events can be used in your applications. Event factory instances can be obtained as follows:

```
EventFactory eventFactory = EventFactoryContext.getInstance().
   getEventFactoryHome().getEventFactory(factoryName);
```

where factoryName is the name of the CommonBaseEvent template you wish to use with the factory.

The factoryName can also be specified as a configuration parameter for a logger. See “Configuring Common Base Events for an application” on page 102 for more details.

Related information

“Configuring Common Base Events for an application” on page 102

“Common Base Event factory” on page 100

Best Practices for Logging Common Base Events in WebSphere Application Server

The following practices will ensure consistent use of Common Base Events within your components and between your components and WebSphere Application Server components:

- Use a different Logger for each component. Sharing loggers across components gets in the way of being able to associate Loggers with component specific information
- Associate Loggers with event templates that specify source component identification. This ensures that the source of all events created with the Logger is properly identified.

- Use the same template for directly created Common Base Events (events created using the Common Base Event factories) and indirectly created Common Base Events (events created using the Java logging API) within the same component.
- Avoid calling the complete method on CommonBaseEvents until you are finished adding data to the CommonBaseEvent and are ready to log it. This ensures that any decisions made by the ContentHandler based on data already in the event will be made using the final data.

The following sample logger.properties file entry demonstrates how to associate Logger com.ibm.componentX with event factory com.ibm.componentX:

```
com.ibm.componentX.eventfactory=com.ibm.componentX
```

The following sample code demonstrates the use of the same event factory setting for direct (Part 1) and indirect (Part 2) Common Base Event logging:

```
<?xml version="1.0" encoding="UTF-8"?>

<TemplateEvent
  version="1.0.1"
  xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
  xsi:noNamespaceSchemaLocation="templateEvent.xsd">

 <CommonBaseEvent
   <sourceComponentId application="My application" component="com.ibm.componentX"/>
   <extendedDataElements CommonBaseEventname="Sample ExtendedDataElement name" type="string">
   <values>Sample ExtendedDataElement value</values>
  </extendedDataElements>
 < /CommonBaseEvent>

< /TemplateEvent>
```

Programming with the JRas framework

The JRas framework described in this task and its sub-tasks is deprecated. However, you can achieve similar results using Java logging.

Use the JRas extensions to incorporate message logging and diagnostic trace into WebSphere Application Server applications. The JRas extensions are based on the stand-alone JRas logging toolkit.

1. Retrieve a reference to the JRas manager.
2. Retrieve message and trace loggers by using methods on the returned manager.
3. Call the appropriate methods on the returned message and trace loggers to create message and trace entries, as appropriate.

Understanding the JRas facility

Note: The JRas framework described in this task and its sub-tasks is deprecated. However, you can achieve similar results using Java logging.

Developing, deploying and maintaining applications are complex tasks. For example, when a running application encounters an unexpected condition it might not be able to complete a requested operation. In such a case you might want the application to inform the administrator that the operation has failed and give information as to why. This enables the administrator to take the proper corrective action. Those who develop or maintain applications might need to gather detailed information relating to the execution path of a running application in order to

determine the root cause of a failure that is due to a code bug. The facilities that are used for these purposes are typically referred to as message logging and diagnostic trace.

Message logging (messages) and diagnostic trace (trace) are conceptually quite similar, but do have important differences. It is important for application developers to understand these differences in order to use these tools properly. To start with, the following operational definitions of messages and trace are provided.

Message
A message entry is an informational record intended to be viewed by end users, systems administrators and support personnel. The text of the message must be clear, concise and interpretable by an end user. Messages are typically localized, meaning they are displayed in the national language of the end user. Although the destination and lifetime of messages might be configurable, some level of message logging is always enabled in normal system operation. Message logging must be used judiciously due to both performance considerations and the size of the message repository.

Trace A trace entry is an information record that is intended to be used by service engineers or developers. As such a trace record may be considerably more complex, verbose and detailed than a message entry. Localization support is typically not used for trace entries. Trace entries may be fairly inscrutable, understandable only by the appropriate developer or service personnel. It is assumed that trace entries are not written during normal runtime operation, but may be enabled as needed to gather diagnostic information.

WebSphere Application Server provides a message logging and diagnostic trace API that can be used by applications. This API is based on the stand-alone JRas logging toolkit which was developed by IBM. The stand-alone JRas logging toolkit is a collection of interfaces and classes that provide message logging and diagnostic trace primitives. These primitives are not tied to any particular product or platform. The stand-alone JRas logging toolkit provides a limited amount of support (typically referred to as systems management support), including log file configuration support based on property files.

As designed, the stand-alone JRas logging toolkit does not contain the support required for integration into the WebSphere Application Server runtime or for usage in a J2EE environment. To overcome these limitations, WebSphere Application Server provides a set of extension classes to address these shortcomings. This collection of extension classes is referred to as the JRas extensions. The JRas extensions do not modify the interfaces introduced by the stand-alone JRas logging toolkit, but simply provide the appropriate implementation classes. The conceptual structure introduced by the stand-alone JRas logging toolkit is described below. It is equally applicable to the JRas extensions.

JRas Concepts

The following is a basic overview of important concepts and constructs introduced by the stand-alone JRas logging toolkit. It is not meant to be an exhaustive overview of the capabilities of this logging toolkit, nor is it intended to be a detailed discussion of usage or programming paradigms. More detailed

information, including code examples, is available in JRas extensions and its subtopics, including in the javadoc for the various interfaces and classes that make up the logging toolkit.

Event Types

The stand-alone JRas logging toolkit defines a set of event types for messages and a set of event types for trace. Examples of message types include informational, warning and error. Examples of trace types include entry, exit and trace.

Event Classes

The stand-alone JRas logging toolkit defines both message and trace event classes.

Loggers

A logger is the primary object with which the user code interacts. Two types of loggers are defined. These are message loggers and trace loggers. The set of methods on message loggers and trace loggers are different, since they provide different functionality. Message loggers create only message records and trace loggers create only trace records. Both types of loggers contain masks that indicates which categories of events the logger should process and which it should ignore. Although every JRas logger is defined to contain both a message and trace mask, the message logger only uses the message mask and the trace logger only uses the trace mask. For example, by setting a message logger's message mask to the appropriate state, it can be configured to process only Error messages and ignore Informational and Warning messages. Changing the state of a message logger's trace mask has no effect.

A logger contains one or more handlers to which it forwards events for further processing. When the user calls a method on the logger, the logger will compare the event type specified by the caller to its current mask value. If the specified type passes the mask check, the logger will create an event object to capture the information relating to the event that was passed to the logger method. This information may include information such as the names of the class and method which is logging the event, a message and parameters to log, among others. Once the logger has created the event object, it forwards the event to all handlers currently registered with the logger.

Methods that are used within the logging infrastructure itself should not make calls to the logger method. When an application uses an object that extends a thread class, implements the hashCode(), and makes a call to the logging infrastructure from that method, the result is a recursive loop.

Handlers

A handler provides an abstraction over an output device or event consumer. An example is a file handler, which knows how to write an event to a file. The handler also contains a mask that is used to further restrict the categories of events the handler will process. For example, a message logger may be configured to pass both warning and error events, but a handler attached to the message logger may be configured to only pass error events. Handlers also include formatters, which the handler invokes to format the data in the passed event before it is written to the output device.

Formatters

Handlers are configured with formatters, which know how to format events of certain types. A handler may contain multiple formatters, each of which knows how to format a specific class of event. The event object is passed to the appropriate formatter by the handler. The formatter returns formatted output to the handler, which then writes it to the output device.

JRas Extensions

The JRas framework described in this task and its sub-tasks is deprecated. However, you can achieve similar results using Java logging.

JRas extensions

The stand-alone JRas logging toolkit defines interfaces and provides a variety of concrete classes that implement these interfaces. Since the stand-alone JRas logging toolkit was developed as a general purpose toolkit, the implementation classes do not contain the configuration interfaces and methods necessary for use in the WebSphere Application Server product. In addition, many of the implementation classes are not written appropriately for use in a J2EE environment. To overcome these shortcomings, WebSphere Application Server provides the appropriate implementation classes that allow integration into the WebSphere Application Server environment. The collection of these implementation classes is referred to as the JRas extensions.

Usage Model

You can use the JRas extensions in three distinct operational modes:

Integrated
: In this mode, message and trace records are written only to logs defined and maintained by the WebSphere Application Server runtime. This is the default mode of operation and is equivalent to the WebSphere Application Server 4.0 mode of operation.

stand-alone
: In this mode, message and trace records are written solely to stand-alone logs defined and maintained by the user. You control which categories of events are written to which logs, and the format in which entries are written. You are responsible for configuration and maintenance of the logs. Message and trace entries are not written to WebSphere Application Server runtime logs.

Combined
: In this mode message and trace records are written to both WebSphere Application Server runtime logs and to stand-alone logs that you must define, control, and maintain. You can use filtering controls to determine which categories of messages and trace are written to which logs.

The JRas extensions are specifically targeted to an integrated mode of operation. The integrated mode of operation can be appropriate for some usage scenarios, but there many scenarios are not adequately addressed by these extensions. Many usage scenarios require a stand-alone or combined mode of operation instead. A set of user extension points has been defined that allow the JRas extensions to be used in either a stand-alone or combined mode of operations.

JRas extension classes

The JRas framework described in this task and its sub-tasks is deprecated. However, you can achieve similar results using Java logging.

WebSphere Application Server provides a base set of implementation classes that collectively are referred to as the JRas extensions. Many of these classes provide the appropriate implementations of loggers, handlers and formatters for use in a WebSphere Application Server environment. As previously noted, this collection of classes is targeted at an Integrated mode of operation. If you choose to use the

JRas extensions in either stand-alone or combined mode, you can reuse the logger and manager class provided by the extensions, but you must provide your own implementations of handlers and formatters.

WebSphere Application Server Message and Trace loggers

The message and trace loggers provided by the stand-alone JRas logging toolkit cannot be directly used in the WebSphere Application Server environment. The JRas extensions provide the appropriate logger implementation classes. Instances of these message and trace logger classes are obtained directly and exclusively from the WebSphere Application Server Manager class, described below. You cannot directly instantiate message and trace loggers. Obtaining loggers in any manner other than directly from the Manager is not allowed. Doing so is a direct violation of the programming model.

The message and trace loggers instances obtained from the WebSphere Application Server Manager class are subclasses of the `RASMessageLogger()` and `RASTraceLogger()` classes provided by the stand-alone JRas logging toolkit. The `RASMessageLogger()` and `RASTraceLogger()` classes define the set of methods that are directly available. Public methods introduced by the JRas extensions logger subclasses cannot be called directly by user code. Doing so is a violation of the programming model.

Loggers are named objects and are identified by name. When the Manager class is called to obtain a logger, the caller is required to specify a name for the logger. The Manager class maintains a name-to-logger instance mapping. Only one instance of a named logger will ever be created within the lifetime of a process. The first call to the Manager with a particular name will result in the logger being created and configured by the Manager. The Manager will cache a reference to the instance, then return it to the caller. Subsequent calls to the Manager that specify the same name will result in a reference to the cached logger being returned. Separate namespaces are maintained for message and trace loggers. This means a single name can be used to obtain both a message logger and a trace logger from the Manager, without ambiguity, and without causing a namespace collision.

In general, loggers have no predefined granularity or scope. A single logger could be used to instrument an entire application. Or users may determine that having a logger per class is more desirable. Or the appropriate granularity may lie somewhere in between. Partitioning an application into logging domains is rightfully determined by the application writer.

The WebSphere Application Server logger classes obtained from the Manager are thread-safe. Although the loggers provided as part of the stand-alone JRas logging toolkit implement the serializable interface, in fact loggers are not serializable. Loggers are stateful objects, tied to a Java virtual machine instance and are not serializable. Attempting to serialize a logger is a violation of the programming model.

Please note that there is no provision for allowing users to provide their own logger subclasses for use in a WebSphere Application Server environment.

WebSphere Application Server handlers

WebSphere Application Server provides the appropriate handler class that is used to write message and trace events to the WebSphere Application Server run-time logs. You cannot configure the WebSphere Application Server handler to write to

any other destination. The creation of a WebSphere Application Server handler is a restricted operation and not available to user code. Every logger obtained from the Manager comes preconfigured with an instance of this handler already installed. You can remove the WebSphere Application Server handler from a logger when you want to run in stand-alone mode. Once you have removed it, you cannot add the WebSphere Application Server handler again to the logger from which it was removed (or any other logger). Also, you cannot directly call any method on the WebSphere Application Server handler. Attempting to create an instance of the WebSphere Application Server handler, to call methods on the WebSphere Application Server handler or to add a WebSphere Application Server handler to a logger by user code is a violation of the programming model.

WebSphere Application Server formatters

The WebSphere Application Server handler comes preconfigured with the appropriate formatter for data that is written to WebSphere Application Server logs. The creation of a WebSphere Application Server formatter is a restricted operation and not available to user code. No mechanism exists that allows the user to obtain a reference to a formatter installed in a WebSphere Application Server handler, or to change the formatter a WebSphere Application Server handler is configured to use.

WebSphere Application Server manager

WebSphere Application Server provides a Manager class located in the `com.ibm.websphere.ras` package. All message and trace loggers must be obtained from this Manager. A reference to the Manager is obtained by calling the static `Manager.getManager()` method. Message loggers are obtained by calling the `createRASMessageLogger()` method on the Manager. Trace loggers are obtained by calling the `createRASTraceLogger()` method on the Manager class.

The manager also supports a *group* abstraction that is useful when dealing with trace loggers. The group abstraction allows multiple, unrelated trace loggers to be registered as part of a named entity called a group. WebSphere Application Server provides the appropriate systems management facilities to manipulate the trace setting of a group, similar to the way the trace settings of an individual trace logger.

For example, suppose component A consist of 10 classes. Suppose each class is configured to use a separate trace logger. Suppose all 10 trace loggers in the component are registered as members of the same group (for example Component_A_Group). You can then turn on trace for a single class. Or you can turn on trace for all 10 classes in a single operation using the group name if you want a component trace. Group names are maintained within the namespace for trace loggers.

Extending the JRas framework

Since the Jras extensions classes do not provide the flexibility and behavior required for many scenarios, a variety of extension points have been defined. You are allowed to write your own implementation classes to obtain the required behavior.

Note: The JRas framework described in this task and its sub-tasks is deprecated. However, you can achieve similar results using Java logging.

In general, the JRas extensions require you to call the Manager class to obtain a message logger or trace logger. No provision is made to allow you to provide your own message or trace logger subclasses. In general, user-provided extensions cannot be used to affect the integrated mode of operation. The behavior of the integrated mode of operation is solely determined by the WebSphere Application Server run-time and the JRas extensions classes.

Handlers

The stand-alone JRas logging toolkit defines the `RASIHandler` interface. All handlers must implement this interface. You can write your own handler classes that implement the `RASIHandler` interface. You should directly create instances of user-defined handlers and add them to the loggers obtained from the Manager.

The stand-alone JRas logging toolkit provides several handler implementation classes. These handler classes are inappropriate for usage in the J2EE environment. You cannot directly use or subclass any of the Handler classes provided by the stand-alone JRas logging toolkit. Doing so is a violation of the programming model.

Formatters

The stand-alone JRas logging toolkit defines the `RASIFormatter` interface. All formatters must implement this interface. You can write your own formatter classes that implement the `RASIFormatter` interface. You can only add these classes to a user-defined handler. WebSphere Application Server handlers cannot be configured to use user-defined formatters. Instead, directly create instances of your formatters and add them to the your handlers appropriately.

As with handlers, the stand-alone JRas logging toolkit provides several formatter implementation classes. Direct usage of these formatter classes is not supported.

Message event types

The stand-alone JRas toolkit defines message event types in the `RASIMessageEvent` interface. In addition, the WebSphere Application Server reserves a range of message event types for future use. The `RASIMessageEvent` interface defines three types, with values of 0x01, 0x02, and 0x04. The values 0x08 through 0x8000 are reserved for future use. You can provide your own message event types by extending this interface appropriately. User-defined message types must have a value of 0x1000 or greater.

Message loggers retrieved from the Manager have their message masks set to *pass* or process all message event types defined in the `RASIMessageEvent` interface. In order to process user-defined message types, you must manually set the message logger mask to the appropriate state by user code after the message logger has been obtained from the Manager. WebSphere Application Server does not provide any built-in systems management support for managing any message types.

Message event objects

The stand-alone JRas toolkit provides a `RASMessageEvent` implementation class. When a message logging method is called on the message logger, and the message type is currently enabled, the logger creates and distributes an event of this class to all handlers currently registered with that logger.

You can provide your own message event classes, but they must implement the `RASIEvent` interface. You must directly create instances of such user-defined message event classes. Once it is created, pass your message event to the message logger by calling the message logger's `fireRASEvent()` method directly. WebSphere Application Server message loggers cannot directly create instances of user-defined types in response to calling a logging method (`msg()`, `message()`...) on the logger. In addition, instances of user-defined message types are never processed by the WebSphere Application Server handler. You cannot create instances of the `RASMessageEvent` class directly.

Trace event types

The stand-alone JRas toolkit defines trace event types in the `RASITraceEvent` interface. You can provide your own trace event types by extending this interface appropriately. In such a case you must ensure that the values for the user-defined trace event types do not collide with the values of the types defined in the `RASITraceEvent` interface.

Trace loggers retrieved from the Manager typically have their trace masks set to reject all types. A different starting state can be specified by using WebSphere Application Server systems management facilities. In addition, the state of the trace mask for a logger can be changed at run-time using WebSphere Application Server systems management facilities.

In order to process user-defined trace types, the trace logger mask must be manually set to the appropriate state by user code. WebSphere Application Server systems management facilities cannot be used to manage user-defined trace types, either at start time or run-time.

Trace event objects

The stand-alone JRas toolkit provides a `RASTraceEvent` implementation class. When a trace logging method is called on the WebSphere Application Server trace logger and the type is currently enabled, the logger creates and distributes an event of this class to all handlers currently registered with that logger.

You can provide your own trace event classes. Such trace event classes must implement the `RASIEvent` interface. You must create instances of such user-defined event classes directly. Once it is created, pass the trace event to the trace logger by calling the trace logger's `fireRASEvent()` method directly. WebSphere Application Server trace loggers cannot directly create instances of user-defined types in response to calling a trace method (`entry()`, `exit()`, `trace()`) on the trace logger. In addition, instances of user-defined trace types are never processed by the WebSphere Application Server handler. You cannot create instances of the `RASTraceEvent` class directly.

User defined types, user defined events and WebSphere Application Server

By definition, the WebSphere Application Server handler will process user-defined message or trace types, or user-defined message or trace event classes. Message and trace entries of either a user-defined type or user-defined event class cannot be written to the WebSphere Application Server run-time logs.

Writing User Extensions:

The JRas framework described in this task and its sub-tasks is deprecated. However, you can achieve similar results using Java logging.

General Considerations

You can configure the WebSphere Application Server to use Java 2 security to restrict access to protected resources such as the file system and sockets. Since user written extensions typically access such protected resources, user written extensions must contain the appropriate security checking calls, using AccessController `doPrivileged()` calls. In addition, the user written extensions must contain the appropriate policy file. In general, it is recommended that you locate user written extensions in a separate package. It is your responsibility to restrict access to the user written extensions appropriately.

Writing a handler

User written handlers must implement the `RASIHandler` interface. The `RASIHandler` interface extends the `RASIMaskChangeGenerator` interface, which extends the `RASIObject` interface. A short discussion of the methods introduced by each of these interfaces follows, along with implementation pointers. For more in depth information on any of the particular interfaces or methods, see the corresponding product javadoc.

RASIObject interface

The `RASIObject` interface is the base interface for stand-alone JRas logging toolkit classes that are stateful or configurable, such as loggers, handlers and formatters.

- The stand-alone JRas logging tookit supports rudimentary properties-file based configuration. To implement this configuration support, the configuration state is stored as a set of key-value pairs in a properties file. The methods*`public Hashtable getConfig()`* and *`public void setConfig(Hashtable ht)`* are used to get and set the configuration state. The JRas extensions do not support properties based configuration and it is recommended that these methods be implemented as no-operations. You can implement your own properties based configuration using these methods.
- Loggers, handlers and formatters can be named objects. For example, the JRas extensions require the user to provide a name for the loggers that are retrieved from the manager. You can name your handlers. The methods *`public String getName()`* and *`public void setName(String name)`* are provided to get or set the name field. The JRas extensions currently do not call these methods on user handlers. You can implement these methods as you want, including as no operations.
- Loggers, handlers and formatters can also contain a description field. The methods *`public String getDescription()`* and *`public void setDescription(String desc)`* can be used to get or set the description field. The JRas extensions currently do not use the description field. You can implement these methods as you want, including as no operations.
- The method *`public String getGroup()`* is provided for usage by the `RASManager`. Since the JRas extensions provide their own Manager class, this method is never called. It is recommended you implement this as a no-operation.

RASIMaskChangeGenerator interface

The `RASIMaskChangeGenerator` interface is the interface that defines the implementation methods for filtering of events based on a mask state. This means that it is currently implemented by both loggers and handlers. By definition, an

object that implements this interface contains both a message mask and a trace mask, although both need not be used. For example, message loggers contain a trace mask, but the trace mask is never used since the message logger never generates trace events. Handlers however can actively use both mask values. For example a single handler could handle both message and trace events.

- The methods *public long getMessageMask()* and *public void setMessageMask(long mask)* are used to get or set the value of the message mask. The methods *public long getTraceMask()* and *public void setTraceMask(long mask)* are used to get or set the value of the trace mask.

In addition, this interface introduces the concept of *calling back* to interested parties when a mask changes state. The callback object must implement the `RASIMaskChangeListener` interface.

- The methods *public void addMaskChangeListener(RASIMaskChangeListener listener)* and *public void removeMaskChangeListener(RASIMaskChangeListener listener)* are used to add or remove listeners to the handler. The method *public Enumeration getMaskChangeListeners()* returns an Enumeration over the list of currently registered listeners. The method *public void fireMaskChangedEvent(RASMaskChangeEvent mc)* is used to call back all the registered listeners to inform them of a mask change event.

For efficiency reasons, the Jras extensions message and trace loggers implement the `RASIMaskChangeListener` interface. The logger implementations maintain a "composite mask" in addition to the logger's own mask. The logger's composite mask is formed by logically *or'ing* the appropriate masks of all handlers that are registered to that logger, then *and'ing* the result with the logger's own mask. For example, the message logger's composite mask is formed by or'ing the message masks of all handlers registered with that logger, then and'ing the result with the logger's own message mask.

This means that all handlers are required to properly implement these methods. In addition, when a user handler is instantiated, the logger it is to be added to should be registered with the handler using the *addMaskChangeListener()* method. When either the message mask or trace mask of the handler is changed, the logger must be called back to inform it of the mask change. This allows the logger to dynamically maintain the composite mask.

The `RASMaskChangedEvent` class is defined by the stand-alone JRas logging toolkit. Direct usage of that class by user code is allowed in this context.

In addition the `RASIMaskChangeGenerator` introduces the concept of caching the names of all message and trace event classes that the implementing object will process. The intent of these methods is to allow a management program such as a GUI to retrieve the list of names, introspect the classes to determine the event types that they might possibly process and display the results. The JRas extensions do not ever call these methods, so they can be implemented as no operations, if desired.

- The methods *public void addMessageEventClass(String name)* and *public void removeMessageEventClass(String name)* can be called to add or remove a message event class name from the list. The method *public Enumeration getMessageEventClasses()* will return an enumeration over the list of message event class names. Similarly, the *public void addTraceEventClass(String name)* and *public void removeTraceEventClass(String name)* can be called to add or remove a trace event class name from the list. The *method public Enumeration getTraceEventClasses()* will return an enumeration over the list of trace event class names.

RASIHandler interface

The `RASIHandler` interface introduces the methods that are specific to the behavior of a handler.

The `RASIHandler` interface as provided by the stand-alone JRas logging toolkit supports handlers that run in either a synchronous or asynchronous mode. In asynchronous mode, events are typically queued by the calling thread and then written by a worker thread. Since spawning of threads is not allowed in the WebSphere Application Server environment, it is expected that handlers will not queue or batch events, although this is not expressly prohibited.

- The methods *`public int getMaximumQueueSize()`* and *`public void setMaximumQueueSize(int size)`* throw `IllegalStateException` are provided to manage the maximum queue size. The method *`public int getQueueSize()`* is provided to query the actual queue size.
- The methods *`public int getRetryInterval()`* and *`public void setRetryInterval(int interval)`* support the notion of error retry, which again implies some type of queueing.
- The methods *`public void addFormatter(RASIFormatter formatter)`*, *`public void removeFormatter(RASIFormatter formatter)`* and *`public Enumeration getFormatters()`* are provided to manage the list of formatters that the handler can be configured with. Different formatters can be provided for different event classes, if appropriate.
- The methods *`public void openDevice()`*, *`public void closeDevice()`* and *`public void stop()`* are provided to manage the underlying device that the handler abstracts.
- The methods *`public void logEvent(RASIEvent event)`* and *`public void writeEvent(RASIEvent event)`* are provided to actually pass events to the handler for processing.

Writing a formatter

User written formatters must implement the `RASIFormatter` interface. The `RASIFormatter` interface extends the `RASIObject` interface. The implementation of the `RASIObject` interface is the same for both handlers and formatters. A short discussion of the methods introduced by the `RASIFormatter` interface follows. For more in depth information on the methods introduced by this interface, see the corresponding product API documentation.

RASIFormatter interface

- The methods *`public void setDefault(boolean flag)`* and *`public boolean isDefault()`* are used by the concrete RASHandler classes provided by the stand-alone JRas logging toolkit to determine if a particular formatter is the default formatter. Since these `RASHandler` classes must never be used in a WebSphere Application Server environment, the semantic significance of these methods can be determined by the user.
- The methods *`public void addEventClass(String name)`*, *`public void removeEventClass(String name)`* and *`public Enumeration getEventClasses()`* are provided to determine which event classes a formatter can be used to format. You can provide the appropriate implementations as you see fit.
- The method *`public String format(RASIEvent event)`* is called by handler objects and returns a formatted String representation of the event.

Programming model summary

The programming model described in this section builds upon and summarizes some of the concepts already introduced. This section also formalizes usage

requirements and restrictions. Use of the WebSphere Application Server JRas extensions in a manner that does not conform to the following programming guidelines is prohibited.

Note: The JRas framework described in this task and its sub-tasks is deprecated. However, you can achieve similar results using Java logging.

As described previously, you can use the WebSphere Application Server JRas extensions in three distinct operational modes. The programming models concepts and restrictions apply equally across all modes of operation.

- You must not use implementation classes provided by the stand-alone JRas logging toolkit directly, unless specifically noted otherwise. Direct usage of those classes is not supported. IBM Support will provide no diagnostic aid or bug fixes relating to direct usage of classes provided by the stand-alone JRas logging toolkit.
- You must obtain message and trace loggers directly from the Manager class. You cannot directly instantiate loggers.
- There is no provision that allows you to replace the WebSphere Application Server message and trace logger classes.
- You must guarantee that the logger names passed to the Manager are unique, and follow the naming constraints documented below. Once a logger is obtained from the Manager, you must not attempt to change the name of the logger by calling the `setName()` method.
- Named loggers can be used more than once. For any given name, the first call to the Manager results in the Manager creating a logger that is associated with that name. Subsequent calls to the Manager that specify the same name result in a reference to the existing logger being returned.
- The Manager maintains a hierarchical namespace for loggers. It is recommended but not required that a dot-separated, fully qualified class name be used to identify any given logger. Other than dots or periods, logger names cannot contain any punctuation characters, such as asterisk (*), comma(.), equals sign(=), colon(:), or quotes.
- Group names must comply with the same naming restrictions as logger names.
- The loggers returned from the Manager are subclasses of the RASMessageLogger and RASTraceLogger provided by the stand-alone JRas logging toolkit. You are allowed to call any public method defined by the RASMessageLogger and RASTraceLogger classes. You are not allowed to call any public method introduced by the provided subclasses.
- If you want to operate in either stand-alone or combined mode, you must provide your own Handler and Formatter subclasses. You are not allowed to use the Handler and Formatter classes provided by the stand-alone JRas logging toolkit. User written handlers and formatters must conform to the documented guidelines.
- Loggers obtained from the manager come with a WebSphere Application Server handler installed. This handler will write message and trace records to logs defined by the WebSphere Application Server runtime. Manage these logs using the provided systems management interfaces.
- You can programmatically add and remove user-defined handlers from a logger at any time. Multiple additions and removals of user defined handlers are allowed. You are responsible for creating an instance of the handler to add, configuring the handler by setting the handler's mask value and formatter appropriately, then adding the handler to the logger using the `addHandler()` method. You are responsible for programmatically updating the masks of user-defined handlers as appropriate.
- You may get a reference to the handler installed within a logger by calling the `getHandlers()` method on the logger and processing the results. You must not

call any methods on the handler obtained in this fashion. You are allowed to remove the WebSphere Application Server handler from the logger by calling the logger's `removeHandler()` method, passing in the reference to the WebSphere Application Server handler. Once removed, the WebSphere Application Server handler cannot be re-added to the logger.

- You are allowed to define your own message type. The behavior of user-defined message types and restrictions on their definitions is discussed in Extending the JRas framework.
- You are allowed to define your own message event classes. The usage of user-defined message event classes is discussed in Extending the JRas framework.
- You are allowed to define your own trace types. The behavior of user-defined trace types and restrictions on your definitions is discussed in Extending the JRas framework.
- You are allowed to define your own trace event classes. The usage of user-defined trace event classes is discussed in Extending the JRas framework.
- You must programmatically maintain the bits in the message and trace logger masks that correspond to any user-defined types. If WebSphere Application Server facilities are being used to manage the predefined types, these updates must not modify the state of any of the bits corresponding to those types. If you are assuming ownership responsibility for the predefined types then you can change all bits of the masks.

JRas Messages and Trace event types

This section describes JRas message and trace event types.

The JRas framework described in this task and its sub-tasks is deprecated. However, you can achieve similar results using Java logging.

Event types

The base message and trace event types defined by the stand-alone JRas logging toolkit are not the same as the ″native″ types recognized by the WebSphere Application Server run-time. Instead the basic JRas types are mapped onto the native types. This mapping may vary by platform or edition. The mapping is discussed below.

Platform Message Event Types

The message event types that are recognized and processed by the WebSphere Application Server runtime are defined in the RASIMessageEvent interface provided by the stand-alone JRas logging toolkit. These message types are mapped onto the native message types as follows.

WebSphere Application Server native type	JRas RASIMessageEvent type
Audit	TYPE_INFO, TYPE_INFORMATION
Warning	TYPE_WARN, TYPE_WARNING
Error	TYPE_ERR, TYPE_ERROR

Application developers can use JRas to issue an MVS WTO (write to operator) message by using a JRas RASIMessageEvent type of TYPE_INFO or TYPE_INFORMATION to issue a WebSphere Application Server for z/OS Audit trace. A WebSphere Application Server for z/OS Audit trace maps to an MVS route code 11 WTO.

Platform Trace Event Types

The trace event types recognized and processed by the WebSphere Application Server runtime are defined in the `RASITraceEvent` interface provided by the stand-alone JRas logging toolkit. The `RASITraceEvent` interface provides a rich and overly complex set of types. This interface defines both a simple set of levels, as well as a set of enumerated types.

- For a user who prefers a simple set of levels, `RASITraceEvent` provides `TYPE_LEVEL1`, `TYPE_LEVEL2`, and `TYPE_LEVEL3`. The implementations provide support for this set of levels. The levels are hierarchical (that is, enabling level 2 will also enable level 1, enabling level 3 also enables levels 1 and 2).
- For users who prefer a more complex set of values that can be *OR'd* together, `RASITraceEvent` provides `TYPE_API`, `TYPE_CALLBACK`, `TYPE_ENTRY_EXIT`, `TYPE_ERROR_EXC`, `TYPE_MISC_DATA`, `TYPE_OBJ_CREATE`, `TYPE_OBJ_DELETE`, `TYPE_PRIVATE`, `TYPE_PUBLIC`, `TYPE_STATIC`, and `TYPE_SVC`.

The trace event types are mapped onto the native trace types as follows:

Mapping WebSphere Application Server trace types to JRas `RASITraceEvent` ″Level″ types.

WebSphere Application Server native type	JRas RASITraceEvent level type
Event	TYPE_LEVEL1
EntryExit	TYPE_LEVEL2
Debug	TYPE_LEVEL3

Mapping WebSphere Application Server trace types to JRas `RASITraceEvent` enumerated types.

WebSphere Application Server native type	JRas RASITraceEvent enumerated types
Event	TYPE_ERROR_EXC, TYPE_SVC, TYPE_OBJ_CREATE, TYPE_OBJ_DELETE
EntryExit	TYPE_ENTRY_EXIT, TYPE_API, TYPE_CALLBACK, TYPE_PRIVATE, TYPE_PUBLIC, TYPE_STATIC
Debug	TYPE_MISC_DATA

For simplicity, it is recommended that one or the other of the tracing type methodologies is used consistently throughout the application. For users who decide to use the non-level types, it is further recommended that you choose one type from each category and use those consistently throughout the application to avoid confusion.

Message and Trace parameters

The various message logging and trace method signatures accept parameter types of `Object`, `Object[]` and `Throwable`. WebSphere Application Server will process and format the various parameter types as follows.

Primitives

Primitives, such as `int` and `long` are not recognized as subclasses of `Object` and cannot be directly passed to one of these methods. A primitive value must be transformed to a proper `Object` type (`Integer`, `Long`) before being passed as a parameter.

Object `toString()` is called on the object and the resulting `String` is displayed. The `toString()` method should be implemented appropriately for any object passed to a message logging or trace method. It is the responsibility of the caller to guarantee that the `toString()` method does not display confidential data such as passwords in clear text, and does not cause infinite recursion.

Object[]
The `Object[]` is provided for the case when more than one parameter is passed to a message logging or trace method. `toString()` is called on each `Object` in the array. Nested arrays are not handled. (i.e. none of the elements in the `Object` array should be an array).

Throwable
The stack trace of the `Throwable` is retrieved and displayed.

Array of Primitives
An array of primitive (e.g. `byte[]`, `int[]` is recognized as an `Object`, but is treated somewhat as a second cousin of `Object` by Java code. In general, arrays of primitives should be avoided, if possible. If arrays of primitives are passed, the results are indeterminate and may change depending on the type of array passed, the API used to pass the array and the release of the product. For consistent results, user code should preprocess and format the primitive array into some type of String form before passing it to the method. If such preprocessing is not performed, the following may result.

- [B@924586a0b - This is deciphered as ″a byte array at location X″. This is typically returned when an array is passed as a member of an `Object[]`. It is the result of calling `toString()` on the `byte[]`.
- Illegal trace argument : array of long. This is typically returned when an array of primitives is passed to a method taking an `Object`.
- 01040703... : the hex representation of an array of bytes. Typically this may be seen when a byte array is passed to a method taking a single `Object`. This behavior is subject to change and should not be relied on.
- ″1″ ″2″ ... : The String representation of the members of an `int[]` formed by converting each element to an Integer and calling `toString` on the Integers. This behavior is subject to change and should not be relied on.
- [Ljava.lang.Object;@9136fa0b : An array of objects. Typically this is seen when an array containing nested arrays is passed.

Controlling message logging

Writing a message to a WebSphere Application Server log requires that the message type passes three levels of filtering or screening.

1. The message event type must be one of the message event types defined in the `RASIMessageEvent` interface.
2. Logging of that message event type must be enabled by the state of the message logger's mask.
3. The message event type must pass any filtering criteria established by the WebSphere Application Server run-time itself.

When a WebSphere Application Server logger is obtained from the Manager, the initial setting of the mask is to forward all native message event types to the WebSphere Application Server handler. It is possible to control what messages get logged by programmatically setting the state of the message logger's mask.

Some editions of the product allow the user to specify a message filter level for a server process. When such a filter level is set, only messages at the specified severity levels are written to WebSphere Application Server logs. This means that

messages types that pass the message logger's mask check may be filtered out by the WebSphere Application Server itself.

Controlling Tracing

Each edition of the product provides a mechanism for enabling or disabling trace. The various editions may support static trace enablement (trace settings are specified before the server is started), dynamic trace enablement (trace settings for a running server process can be dynamically modified) or both.

Writing a trace record to a WebSphere Application Server requires that the trace type passes three levels of filtering or screening.

1. The trace event type must be one of the trace event types defined in the `RASITraceEvent` interface.
2. Logging of that trace event type must be enabled by the state of the trace logger's mask.
3. The trace event type must pass any filtering criteria established by the WebSphere Application Server run-time itself.

When a logger is obtained from the Manager, the initial setting of the mask is to suppress all trace types. The exception to this rule is the case where the WebSphere Application Server run-time supports static trace enablement and a non-default startup trace state for that trace logger has been specified. Unlike message loggers, the WebSphere Application Server may dynamically modify the state of a trace loggers trace mask. WebSphere Application Server will only modify the portion of the trace logger's mask corresponding to the values defined in the `RASITraceEvent` interface. WebSphere Application Server will not modify undefined bits of the mask that may be in use for user defined types.

When the dynamic trace enablement feature available on some platforms is used, the trace state change is reflected both in the Application Server run-time and the trace loggers trace mask. If user code programmatically changes the bits in the trace mask corresponding to the values defined by in the `RASITraceEvent` interface, the trace logger's mask state and the run-time state will become unsynchronized and unexpected results will occur. Therefore, programmatically changing the bits of the mask corresponding to the values defined in the `RASITraceEvent` interface is not allowed.

Instrumenting an application with JRas extensions

The JRas framework described in this task and its sub-tasks is deprecated. However, you can achieve similar results using Java logging.

To instrument an application using the WebSphere Application Server JRas extensions, perform the following steps:

1. Determine the mode the extensions will be used in: integrated, stand-alone or combined.
2. If the extensions will be used in either stand-alone or combined mode, create the necessary handler and formatter classes.
3. If localized messages will be used by the application, create a resource bundle as described in Creating JRas resource bundles and message files.
4. In the application code, get a reference to the Manager class and create the manager and logger instances as described in Creating JRas manager and logger instances.

5. Insert the appropriate message and trace logging statements in the application as described in Creating JRas manager and logger instances.

Creating JRas resource bundles and message files

The JRas framework described in this task and its sub-tasks is deprecated. However, you can achieve similar results using Java logging.

The WebSphere Application Server message logger provides the `message()` and `msg()` methods to allow the user to log localized messages. In addition, it provides the `textMessage()` method for logging of messages that are not localized. Applications can use either or both, as appropriate.

The mechanism for providing localized messages is the Resource Bundle support provided by the IBM Developer Java Technology Edition. If you are not familiar with resource bundles as implemented by the Developer's Kit, you can get more information from various texts, or by reading the Javadoc for the `java.util.ResourceBundle`, `java.util.ListResourceBundle` and `java.util.PropertyResourceBundle` classes, as well as the `java.text.MessageFormat` class.

The `PropertyResourceBundle` is the preferred mechanism to use. In addition, note that the JRas extensions do not support the extended formatting options such as {1, date} or {0,number, integer} that are provided by the `MessageFormat` class.

You can forward messages that are written to the internal WebSphere Application Server logs to other processes for display. For example, messages displayed on the administrator console, which can be running in a different location than the server process, can be localized using the *late binding* process. Late binding means that WebSphere Application Server does not localize messages when they are logged, but defers localization to the process that displays the message.

To properly localize the message, the displaying process must have access to the resource bundle where the message text is stored. This means that you must package the resource bundle separately from the application, and install it in a location where the viewing process can access it. If you do not want to take these steps, you can use the early binding technique to localize messages as they are logged.

The two techniques are described as follows:

Early binding
: The application must localize the message before logging it. The application looks up the localized text in the resource bundle and formats the message. When formatting is complete, the application logs the message using the `textMessage()` method. Use this technique to package the application's resource bundles with the application.

Late binding
: The application can choose to have the WebSphere Application Server runtime localize the message in the process where it is displayed. Using this technique,the resource bundles are packaged in a stand-alone `.jar` file, separately from the application. You must then install the resource bundle `.jar` file on every machine in the installation from which an administrator's console or log viewing program might be run. You must install the `.jar` file in a directory that is part of the extensions classpath. In addition, if you forward logs to IBM service, you must also forward the `.jar` file containing the resource bundles.

To create a resource bundle, perform the following steps.

1. Create a text properties file that lists message keys and the corresponding messages. The properties file must have the following characteristics:
   - Each property in the file is terminated with a line-termination character.
   - If a line contains only white space, or if the first non-white space character of the line is the pound sign symbol (#) or exclamation mark (!), the line is ignored. The # and ! characters can therefore be used to put comments into the file.
   - Each line in the file, unless it is a comment or consists only of white space, denotes a single property. A backslash (\) is treated as the line-continuation character.
   - The syntax for a property file consists of a key, a separator, and an element. Valid separators include the equal sign (=), colon (:), and white space ().
   - The key consists of all characters on the line from the first non-white space character to the first separator. Separator characters can be included in the key by escaping them with a backslash (\), but doing this is not recommended, because escaping characters is error prone and confusing. It is instead recommended that you use a valid separator character that does not appear in any keys in the properties file.
   - White space after the key and separator is ignored until the first non-white space character is encountered. All characters remaining before the line-termination character define the element.

   See the Java documentation for the `java.util.Properties` class for a full description of the syntax and construction of properties files.
2. The file can then be translated into localized versions of the file with language-specific file names (for example, a file named `DefaultMessages.properties` can be translated into DefaultMessages_de.properties for German and DefaultMessages_ja.properties for Japanese).
3. When the translated resource bundles are available, write them to a system-managed persistent storage medium. Resource bundles are then used to convert the messages into the requested national language and locale.
4. When a message logger is obtained from the JRas manager, it can be configured to use a particular resource bundle. Messages logged via the `message()` API will use this resource bundle when message localization is performed. At run time, the user's locale setting is used to determine the properties file from which to extract the message specified by a message key, thus ensuring that the message is delivered in the correct language.
5. If the message loggers `msg()` method is called, a resource bundle name must be explicitly provided.

The application locates the resource bundle based on the file's location relative to any directory in the classpath. For instance, if the property resource bundle named DefaultMessages.properties is located in the *baseDir*`/`*subDir1*`/`*subDir2*`/resources` directory and *baseDir* is in the class path, the name *subdir1*`.`*subdir2*`.resources.DefaultMessage` is passed to the message logger to identify the resource bundle.

Developing JRas resource bundles:

The JRas framework described in this task and its sub-tasks is deprecated. However, you can achieve similar results using Java logging.

Resource bundle sample

You can create resource bundles in several ways. The best and easiest way is to create a properties file that supports a `PropertiesResourceBundle`. This sample shows how to create such a properties file.

For this sample, four localizable messages are provided. The properties file is created and the key-value pairs inserted into it. All the normal properties files conventions and rules apply to this file. In addition, the creator must be aware of other restrictions imposed on the values by the Java `MessageFormat` class. For example, apostrophes must be ″escaped″ or they will cause a problem. Also avoid use of non-portable characters. WebSphere Application Server does not support usage of extended formatting conventions that the `MessageFormat` class supports, such as {1, date} or {0,number, integer}.

Assume that the base directory for the application that uses this resource bundle is ″`baseDir`″ and that this directory will be in the classpath. Assume that the properties file is stored in a subdirectory of `baseDir` that is not in the classpath (e.g. `baseDir/subDir1/subDir2/resources`). In order to allow the messages file to be resolved, the name `subDir1.subDir2.resources.DefaultMessage` is used to identify the `PropertyResourceBundle` and is passed to the message logger.

For this sample, the properties file is named `DefaultMessages.properties`.

```
# Contents of DefaultMessages.properties file
MSG_KEY_00=A message with no substitution parameters.
MSG_KEY_01=A message with one substitution parameter: parm1={0}
MSG_KEY_02=A message with two substitution parameters: parm1={0}, parm2 = {1}
MSG_KEY_03=A message with three parameter: parm1={0}, parm2 = {1}, parm3={2}
```

Once the file `DefaultMessages.properties` is created, the file can be sent to a translation center where the localized versions will be generated.

Creating JRas manager and logger instances

You can use the JRas extensions in integrated, stand-alone, or combined mode. Configuration of the application will vary depending on the mode of operation, but usage of the loggers to log message or trace entries is identical in all modes of operation.

Note: The JRas framework described in this task and its sub-tasks is deprecated. However, you can achieve similar results using Java logging.

Integrated mode is the default mode of operation. In this mode, message and trace events are sent to the WebSphere Application Server logs. See Setting up for integrated JRas operation for information on configuring for this mode of operation.

In the combined mode, message and trace events are logged to both WebSphere Application Server and user-defined logs. See Setting up for combined JRas operation for more information on configuring for this mode of operation.

In the stand-alone mode, message and trace events are logged only to user-defined logs. See Setting up for stand-alone JRas operation for more information on configuring for this mode of operation.

Using the message and trace loggers

Regardless of the mode of operation, the use of message and trace loggers is the same. See Creating JRas resource bundles and message files for more information on using message and trace loggers.

Using a message logger

The message logger is configured to use the `DefaultMessages` resource bundle. Message keys must be passed to the message loggers if the loggers are using the `message()` API.

```
msgLogger.message(RASIMessageEvent.TYPE_WARNING, this,
     methodName, "MSG_KEY_00");
... msgLogger.message(RASIMessageEvent.TYPE_WARN, this,
     methodName, "MSG_KEY_01", "some string");
```

If message loggers use the `msg()` API, you can specify a new resource bundle name.

```
msgLogger.msg(RASIMessageEvent.TYPE_ERR, this, methodName,
                  "ALT_MSG_KEY_00", "alternateMessageFile");
```

You can also log a text message. If you are using the `textMessage` API, no message formatting is done.

```
msgLogger.textMessage(RASIMessageEvent.TYPE_INFO, this, methodName,"String and Integer",
"A String", new Integer(5));
```

Using a trace logger

Since trace is normally disabled, trace methods should be guarded for performance reasons.

```
private void methodX(int x, String y, Foo z)
{
   // trace an entry point. Use the guard to make sure tracing is enabled.
Do this checking before we waste cycles gathering parameters to be traced.
   if (trcLogger.isLoggable(RASITraceEvent.TYPE_ENTRY_EXIT) {
        // since I want to trace 3 parameters, package them up in an Object[]
        Object[] parms = {new Integer(x), y, z};
        trcLogger.entry(RASITraceEvent.TYPE_ENTRY_EXIT, this, "methodX", parms);
  }
... logic
  // a debug or verbose trace point
  if (trcLogger.isLoggable(RASITraceEvent.TYPE_MISC_DATA) {
        trcLogger.trace(RASITraceEvent.TYPE_MISC_DATA, this, "methodX" "reached here");
  }
  ...
  // Another classification of trace event. Here an important state change
has been detected, so a different trace type is used.
  if (trcLogger.isLoggable(RASITraceEvent.TYPE_SVC) {
     trcLogger.trace(RASITraceEvent.TYPE_SVC, this, "methodX", "an important event");
  }
 ...
  // ready to exit method, trace. No return value to trace
    if (trcLogger.isLoggable(RASITraceEvent.TYPE_ENTRY_EXIT)) {
        trcLogger.exit(RASITraceEvent.TYPE_ENTRY_EXIT, this, "methodX");
   }
}
```

Setting up for integrated JRas operation

The JRas framework described in this task and its sub-tasks is deprecated. However, you can achieve similar results using Java logging.

In the integrated mode of operation, message and trace events are sent to WebSphere Application Server logs. This is the default mode of operation.

1. Import the requisite JRas extensions classes

   ```
   import com.ibm.ras.*;
   import com.ibm.websphere.ras.*;
   ```

2. Declare logger references.

   ```
   private RASMessageLogger msgLogger = null;
   private RASTraceLogger trcLogger = null;
   ```

3. Obtain a reference to the Manager and create the loggers. Since loggers are named singletons, you can do this in a variety of places. One logical candidate for enterprise beans is the `ejbCreate()` method. For example, for the enterprise bean named ″myTestBean″, place the following code in the `ejbCreate()` method.

   ```
   com.ibm.websphere.ras.Manager mgr = com.ibm.websphere.ras.Manager.getManager();
   msgLogger = mgr.createRASMessageLogger("Acme", "WidgetCounter", "RasTest",
            myTestBean.class.getName());



   // Configure the message logger to use the message file created
   // for this application.
   msgLogger.setMessageFile("acme.widgets.DefaultMessages");
   trcLogger = mgr.createRASTraceLogger("Acme", "Widgets", "RasTest",
            myTestBean.class.getName());
   mgr.addLoggerToGroup(trcLogger, groupName);
   ```

   Related concepts

   “Creating JRas manager and logger instances” on page 123
   You can use the JRas extensions in integrated, stand-alone, or combined mode. Configuration of the application will vary depending on the mode of operation, but usage of the loggers to log message or trace entries is identical in all modes of operation.

   Related tasks

   “Setting up for combined JRas operation”

   “Setting up for stand-alone JRas operation” on page 126

Setting up for combined JRas operation

The JRas framework described in this task and its sub-tasks is deprecated. However, you can achieve similar results using Java logging.

In combined mode, messages and trace are logged to both WebSphere Application Server logs and user-defined logs. The following sample assumes that you have written a user defined handler named `SimpleFileHandler` and a user defined formatter named `SimpleFormatter`. It also assumes that you are not using user defined types or events.

1. Import the requisite JRas extensions classes

   ```
   import com.ibm.ras.*;
   import com.ibm.websphere.ras.*;
   ```

2. Import the user handler and formatter.

   ```
   import com.ibm.ws.ras.test.user.*;
   ```

3. Declare the logger references.

   ```
   private RASMessageLogger msgLogger = null;
     private RASTraceLogger trcLogger = null;
   ```

4. Obtain a reference to the Manager, create the loggers and add the user handlers. Since loggers are named singletons, you can obtain a reference to the

loggers in a number of places. One logical candidate for enterprise beans is the `ejbCreate()` method. Make sure that multiple instances of the same user handler are not accidentally inserted into the same logger. Your initialization code must handle this. The following sample is a message logger sample. The procedure for a trace logger is similar.

```
com.ibm.websphere.ras.Manager mgr = com.ibm.websphere.ras.Manager.getManager();
    msgLogger = mgr.createRASMessageLogger("Acme", "WidgetCounter", "RasTest",
           myTestBean.class.getName());
    // Configure the message logger to use the message file defined
    // in the ResourceBundle sample.
    msgLogger.setMessageFile("acme.widgets.DefaultMessages");

   // Create the user handler and formatter. Configure the formatter,
   // then add it to the handler.
   RASIHandler handler = new SimpleFileHandler("myHandler", "FileName");
   RASIFormatter formatter = new SimpleFormatter("simple formatter");
    formatter.addEventClass("com.ibm.ras.RASMessageEvent");
    handler.addFormatter(formatter);

   // Add the Handler to the logger. Add the logger to the list of the
   //handlers listeners, then set the handlers
   // mask, which will update the loggers composite mask appropriately.
   // WARNING - there is an order dependency here that must be followed.
   msgLogger.addHandler(handler);
   handler.addMaskChangeListener(msgLogger);
   handler.setMessageMask(RASIMessageEvent.DEFAULT_MESSAGE_MASK);
```

Related concepts

"Creating JRas manager and logger instances" on page 123
You can use the JRas extensions in integrated, stand-alone, or combined mode. Configuration of the application will vary depending on the mode of operation, but usage of the loggers to log message or trace entries is identical in all modes of operation.

Related tasks

"Setting up for integrated JRas operation" on page 124

"Setting up for stand-alone JRas operation"

Setting up for stand-alone JRas operation

The JRas framework described in this task and its sub-tasks is deprecated. However, you can achieve similar results using Java logging.

In stand-alone mode, messages and traces are logged only to user-defined logs. The following sample assumes that you have a user-defined handler named `SimpleFileHandler` and a user-defined formatter named `SimpleFormatter`. It is also assumes that no user-defined types or events are being used.

1. Import the requisite JRas extensions classes

   ```
   import com.ibm.ras.*;
   import com.ibm.websphere.ras.*;
   ```

2. Import the user handler and formatter.

   ```
   import com.ibm.ws.ras.test.user.*;
   ```

3. Declare the logger references.

   ```
   private RASMessageLogger msgLogger = null;
     private RASTraceLogger trcLogger = null;
   ```

4. Obtain a reference to the Manager, create the loggers and add the user handlers. Since loggers are named singletons, you can obtain a reference to the loggers in a number of places. One logical candidate for enterprise beans is the `ejbCreate()` method. Make sure that multiple instances of the same user

handler are not accidentally inserted into the same logger. Your initialization code must handle this. The following sample is a message logger sample. The procedure for a trace logger is similar.

```
com.ibm.websphere.ras.Manager mgr = com.ibm.websphere.ras.Manager.getManager();
    msgLogger = mgr.createRASMessageLogger("Acme", "WidgetCounter", "RasTest",
             myTestBean.class.getName());
    // Configure the message logger to use the message file defined in
    //the ResourceBundle sample.
    msgLogger.setMessageFile("acme.widgets.DefaultMessages");

    // Get a reference to the Handler and remove it from the logger.
    RASIHandler aHandler = null;
    Enumeration enum = msgLogger.getHandlers();
    while (enum.hasMoreElements()) {
         aHandler = (RASIHandler)enum.nextElement();
         if (aHandler instanceof WsHandler)
              msgLogger.removeHandler(wsHandler);
    }

    // Create the user handler and formatter. Configure the formatter,
    // then add it to the handler.
    RASIHandler handler = new SimpleFileHandler("myHandler", "FileName");
    RASIFormatter formatter = new SimpleFormatter("simple formatter");
    formatter.addEventClass("com.ibm.ras.RASMessageEvent");
    handler.addFormatter(formatter);

   // Add the Handler to the logger. Add the logger to the list of the
   // handlers listeners, then set the handlers
   // mask, which will update the loggers composite mask appropriately.
   // WARNING - there is an order dependency here that must be followed.
   msgLogger.addHandler(handler);
   handler.addMaskChangeListener(msgLogger);
   handler.setMessageMask(RASIMessageEvent.DEFAULT_MESSAGE_MASK);
```

Related concepts

"Creating JRas manager and logger instances" on page 123
You can use the JRas extensions in integrated, stand-alone, or combined mode. Configuration of the application will vary depending on the mode of operation, but usage of the loggers to log message or trace entries is identical in all modes of operation.

Related tasks

"Setting up for integrated JRas operation" on page 124

"Setting up for combined JRas operation" on page 125

Logging messages and trace data for Java server applications

By using the WebSphere Application Server for z/OS support for logging application messages and trace data, you can improve the reliability, availability, and serviceability of any Java application that runs in a WebSphere Application Server for z/OS server. Through this support, your Java application's messages can appear on the MVS master console, in the error log stream, or in the component trace (CTRACE) data set for WebSphere Application Server for z/OS. Your application's trace entries can appear in the same CTRACE data set.

Determining where to issue the messages

You might want to issue messages to the MVS master console to report serious error conditions for mission-critical applications. Through the master console, an operator can receive and, if necessary, take action in response to a message that

indicates the status of an application. In addition, by directing messages to the master console, you can trigger automation packages to take action for specific conditions or events related to your application's processing.

Any messages that your application issues to the console also appear in either the error log stream or the CTRACE data set for WebSphere Application Server for z/OS, depending on the message type. Logging the messages in these system resources can help you more easily diagnose errors related to your application's processing. Similarly, issuing requests to log trace data in the CTRACE data set is another method of recording error conditions or collecting application data for diagnostic purposes.

System performance when logging messages and trace data

You can select the amount and types of trace data to be collected, which provides you with the ability to either run your application with minimal tracing when performance is a priority, or run your application with detailed tracing when you need to recreate a problem and collect additional diagnostic information.

The error log stream, the CTRACE data set for WebSphere Application Server for z/OS, and the master console are primarily intended for monitoring or recording diagnostic data for system components and critical applications. Depending on your installation's configuration, directing application messages and data to these resources might have an adverse affect on system performance. For example, if you send application data to the CTRACE data set, trace entries in that data set might wrap more quickly, which means you might lose some critical diagnostic data because the system writes new entries over existing ones when wrapping occurs. Use this logging support judiciously.

Note: You can only use WebSphere Application Server for z/OS support for logging messages and trace data for Java applications, not for Java applets.

Issuing application messages to the MVS master console

The JRas framework described in this task and its sub-tasks is deprecated. However, you can achieve similar results using Java logging.

With the WebSphere Application Server for z/OS reliability, availability, and serviceability support for Java (JRas) framework, you can issue messages from your Java application to the MVS master console. You might want to issue messages to the master console to report serious error conditions for mission-critical applications, or to trigger automation packages.

The messages your application issues also appear in either the error log stream or the component trace (CTRACE) data set that WebSphere Application Server for z/OS uses.

Logging the messages is another method of recording error conditions or collecting application data for diagnostic purposes.

Using a message logger

WebSphere Application Server for z/OS provides code that creates and manages a message logger, which processes your application's messages. WebSphere Application Server for z/OS creates only one message logger for each unique organization, product, or component, so that you can more easily identify the

messages recorded in the error log stream or CTRACE data set for a specific application. The message logger runs in the Java Virtual Machine (JVM) for the WebSphere Application Server for z/OS server in which your Java application will run.

To use a message logger, in your Java application:

1. Define the message logger.
2. Drive the method to instruct WebSphere Application Server for z/OS to create the message logger.
3. Code messages at appropriate points in your application.

Chapter 5. Diagnosing problems (using diagnosis tools)

The purpose of this section is to aid you in understanding why your enterprise application, application server, or WebSphere Application Server is not working and to help you resolve the problem. Unlike performance tuning which focuses on solving problems associated with slow processes and un-optimized performance, problem determination focuses on finding solutions to functional problems.

1. For tips on investigating common problems organized according to tasks within WebSphere Application server, see Troubleshooting by task.
2. For tips on how to investigate common kinds of problems based on the component that is causing the problem, see Troubleshooting by component.
3. For help in using WebSphere Application Server utilities to help you diagnose the problem, see Working with diagnostic tools and controls.
4. For help in viewing diagnostic information like dumps, error logs and CTRACE information, see Viewing diagnostic information
5. The IBM Developer Kit and Runtime Environment, Java 2 Technology Edition, Version 1.4.1 Diagnostics Guide describes debugging techniques and the diagnostic tools that are available to help you solve problems with Java. It also gives guidance on how to submit problems to IBM. You can find the guide at http://www.ibm.com/developerworks/java/jdk/diagnosis/.
6. For current information available from IBM Support on known problems and their resolution, see the IBM Support page.
7. IBM Support has documents that can save you time gathering information needed to resolve this problem. Before opening a PMR, see the IBM Support page.

Related tasks

Chapter 6, “Troubleshooting by task,” on page 233

“Troubleshooting by component” on page 5

Chapter 4, “Adding logging and tracing to your application,” on page 61

Chapter 3, “Debugging applications,” on page 57

Acquiring skills for problem determination

In a large-scale enterprise system such as the WebSphere Application Server for z/OS environment, diagnosis might require a variety of skills to progress from a symptom to fixing the underlying cause of that symptom. Because WebSphere Application Server for z/OS exploits many of the qualities and services that are unique to the z/OS operating system, diagnosing system-related problems might require skills in the following areas:

- Parallel sysplex
- TCP/IP
- Security Server (RACF) or the equivalent
- Database systems such as DB2 Universal Database for z/OS and OS/390
- UNIX Systems Services

You can find information for many of these topics in the publications available through the z/OS library Web site.

Similarly, diagnosing application-related problems might require a variety of skills because of the variety of application components that WebSphere Application

Server for z/OS supports. Programmers who diagnose application problems in the WebSphere Application Server for z/OS environment need some familiarity with the following:

- The roles defined in the Sun Microsystems Java 2 Enterprise Edition Specification V1.3.
- The programming models and specifications for application components (Enterprise beans, Web applications, and client programs) supported in the J2EE 1.3 environment.
- The process of assembling, deploying, installing, and running server applications and clients in the WebSphere Application Server for z/OS environment.
- Various tools such as the WebSphere Application Server for z/OS error log, and the job logs for programs running on z/OS.

Working with diagnostic tools and controls

The following list summarizes the z/OS tools you will use to access and work with diagnostic information.

- **z/OS console**

  The console displays configuration errors that cause the termination of the WebSphere Application Server for z/OS address spaces. Whatever goes to the console also goes to SYSLOG.
- **System log (SYSLOG)**

  SYSLOG is the repository for all messages that have appeared on the operator console. It also contains warning and informational messages that might be helpful after a failure has occurred.
- **Job log**

  The job log contains errors and warnings (non-termination) that are related to configuration. Anything that goes to the console and SYSLOG automatically goes to the job log.
- **System output (SYSOUT)**

  SYSOUT is a batch log that usually contains diagnostic data from the Java Virtual Machine (JVM) that runs in the servant (region). Any messages written to stderr will end up in SYSOUT. In addition, SYSOUT might contain error messages that usually appear in the log stream, but were redirected to SYSOUT because the log stream was not available.
- **Error log**

  The error log contains messages issued through Java logging and JRas support, if any. In addition, the error log usually contains messages intended for IBM use only; these are messages that support actions, problems, or issues that are usually externalized through additional messages in other sources. When you work with IBM service, you might be asked to supply the error log so that service personnel can use these support messages to diagnose the problem.

  Note: You must update the CFRM policy before using log streams that are CF-resident, such as the WebSphere error log and RRS logs. See Updating the CFRM policy for details.
- **SYSPRINT**

  SYSPRINT contains component trace (CTRACE) output for clients, and for servants when WebSphere Application Server for z/OS is configured to use SYSPRINT rather than CTRACE buffers and data sets.
- **Component trace (CTRACE) data set**

CTRACE data sets contain diagnostic trace entries for various processes, depending on the trace options configured for WebSphere Application Server for z/OS.

- **Logrec**

  When an error occurs, the system records information about the error in the logrec data set or the logrec log stream. The information provides you with a history of all hardware failures, selected software errors, and selected system conditions.
- **Transaction XA Partner Log**

  This log is used for recovery of XA resources. When an application accesses XA resources, WebSphere Application Server stores information about the resource to enable XA transaction recovery. For instructions on how to use the Customization dialog to configure the Transaction XA Partner Log see Customization Dialog variables: Stand-alone application server cell. For instructions on how to change the location of the Transaction XA Partner Log, see Transaction service settings.
- **SDSF**

  Use the SDSF DA panel to see how many application server address spaces are active, and observe at the CPU%, ECPU% and SIO rate. Use the ″ENC″ panel to see the enclaves running and what service classes they are running under.
- **RMF**

  See “Using RMF” on page 168 for instructions on starting and using RMF to monitor your transactions.
- **MODIFY command**

  See “Displaying WebSphere Application Server work” on page 174 for instructions on using the z/OS modify command to display information about WebSphere Application Server for z/OS servers or servants (regions).

To find additional information about these tools, and about the process of diagnosing problems on z/OS, use the z/OS Web Library to access the following books:

- *z/OS MVS Diagnosis: Procedures, GA22-7587* helps you diagnose problems in the MVS operating system, its subsystems, its components, and in applications running under the system.
- *z/OS MVS Diagnosis: Tools and Service Aids, GA22-7589* provides detailed information about tools and service aids that can help you diagnose problems. This book contains a guide on how to select the appropriate tool or service aid for your purposes, and also provides an overview of all the tools and service aids available.

Related tasks

Chapter 3, “Debugging applications,” on page 57

Updating the CFRM policy

You must update the coupling facility resource management (CFRM) policy before using log streams that are CF-resident, such as the WebSphere Application Server error log and RRS logs. If you have the source for the current active CFRM policy, update the source and use the IXCMIAPU Administrative Data utility to generate the new policy. If you do not have the source for the current active CFRM policy, rebuild the source from the active CFRM policy:

1. Find the active policy by issuing the command: D XCF,POL You will get output similar to this (partial display):

```
D XCF,POL
  IXC364I  10.57.49  DISPLAY XCF 061
  . . .
  TYPE: CFRM
       POLNAME:       POLCF1N1
       STARTED:       03/14/2003 11:32:22
       LAST UPDATED:  03/14/2003 11:31:52
  . . .
```

2. List the active CFRM Policy's structure definitions by using the Administrative Data utility:

```
//STEP1   EXEC PGM=IXCMIAPU
//SYSPRINT DD   SYSOUT=*
//SYSABEND DD   SYSOUT=*
//SYSIN    DD   *
DATA TYPE(CFRM) REPORT(YES)
/*
```

3. Extract the definitions for the ACTIVE policy only. Using the SYSPRINT from the above utility job, edit the output with the following steps so it can be used to define a new policy in the next job:
   - Extract the definitions for the ACTIVE policy only.
   - Delete the heading lines.
   - Add the new structure definition using the BBOWCFRM member of the target CNTL dataset as a model.
   - Copy it into a FB-LRECL(80) dataset to be used as SYSIN for the following job.
4. Use the Administrative Data utility to update the CFRM policy. The policy name can be the same as the ACTIVE CFRM policy or a new name. If you use the active policy name, REPLACE(YES) must be specified on the DEFINE control statement.

```
//STEP20   EXEC PGM=IXCMIAPU
//SYSPRINT DD   SYSOUT=*
//SYSABEND DD   SYSOUT=*
//SYSIN    DD   *
 DATA TYPE(CFRM) REPORT(YES)

 DEFINE POLICY NAME(POLCF1N1) REPLACE(YES)

    CF NAME(CF1LPAR) DUMPSPACE(5000) PARTITION(0E) CPCID(00)
       TYPE(009672) MFG(IBM) PLANT(02) SEQUENCE(000000051205)

    CF NAME(CF2LPAR) DUMPSPACE(5000) PARTITION(0F) CPCID(00)
       TYPE(009672) MFG(IBM) PLANT(02) SEQUENCE(000000051205)

    STRUCTURE NAME(CTS130_DFHLOG) SIZE(24000) INITSIZE(12000)
       REBUILDPERCENT(1)  PREFLIST(CF1LPAR, CF2LPAR)
    . . .
              <== Insert your new structure definitions here
```

5. Activate the new policy by issuing the following MVS Command:

```
SETXCF START,POLICY,TYPE=CFRM,POLNAME=POLCF1N1
```

Best practices for maintaining the run-time environment

Use the following guidelines to make sure that WebSphere Application Server for z/OS is customized and maintained correctly, to support your installation's application workload. Checking these basic software and hardware requirements can help you avoid problems with the run-time environment.

- **Check that you have the necessary prerequisite software up and running.** Check that they have the proper authorizations and that the definitions are correct.

- **Check for messages that signal potential problems.** Look for warning and error messages in the following sources:
  - SYSLOG from other z/OS subsystems and products, such as TCP/IP (especially the DNS, if in use), RACF, and so on
  - WebSphere Application Server for z/OS error log
  - SYSPRINT of the WebSphere Application Server for z/OS
  - Component trace (CTRACE) output for the server
- **Check the ports used by WebSphere Application Server.** The ports that are used by WebSphere Application Server must not be reserved by any other z/OS component.
- **Ensure that z/OS has enough DASD space for SVC dumps.** You might have to adjust the amount of space, because it depends on the size of your applications, on the configured Java virtual machine (JVM) heap size, and on the number of servant regions that might be included in one dump, and so on. For an SVC dump of one controller and one servant, you can start with a minimum of 512, but might have to increase the MAXSPACE to 1024 or higher, given the factors listed above.
- **Check your general environment.** Does your system have enough memory? Insufficient memory problems can show up as AUX shortages, abends, or exceptions from the WebSphere Application Server for z/OS run-time. Sometimes the heap size for Language Environment (LE) and for the Java virtual machine (JVM) needs to be increased. If you are using RRS and DB2, make sure your system has enough space for archive data sets.
- **Make sure all prerequisite fixes have been installed**; a quick check for a fix can save hours of debugging.

  For the most current information on fixes and service updates, see:
  - The Preventive Service Planning (PSP) buckets for both WebSphere Application Server for z/OS and JAVA subsets of the WebSphere Application Server for z/OS Upgrade. To obtain a copy of the most current versions of these PSP buckets, you can either contact the IBM Support Center, use S/390 SoftwareXcel or link to IBMLink.
  - The Support Web page of the WebSphere Application Server for z/OS Web site, which contains a table of the latest authorized program analysis reports (APARs).

  With the latest service information, check the following:
  - Ensure that all prerequisite PTFs (fixes) have been applied to the system.
  - Verify that all PTFs were actually present in the executables that were used at the time of error. Often, SMP can indicate that a fix is present and installed on the system when, in reality, the executables that were used at the time of error did not contain the fix.

Best practices for using system controls

- You have the option of using a z/OS system logger log stream as the WebSphere Application Server for z/OS error log. The WebSphere variable

  ```
  ras_log_logstreamName
  ```

  identifies which log stream you want to use for the error log; it has no default setting. If you do not use a log stream, however, messages that usually appear in the error log are directed to server's job log.
- You have the option of directing trace output to SYSPRINT or buffers. The WebSphere variable

  ```
  ras_trace_outputLocation
  ```

controls the location of trace output; its default values are SYSPRINT for client applications, and buffers to all other processes. Although you can change the default for other processes from buffers to SYSPRINT, performance is better when you use buffers.

- You can use the Resource Measurement Facility (RMF) to view status information that might indicate potential problems. WebSphere Application Server for z/OS uses Workload Manager (WLM) services to report transaction begin-to-end response times and execution delay times, which might indicate that changes are required for timeout values or tuning controls.

Related tasks

"Setting up the error log" on page 174

Related reference

Workload management (WLM) for z/OS
Workload management optimizes the distribution of incoming work requests to the application servers, enterprise beans, servlets, and other objects that can most effectively process the requests. Workload management also provides failover when servers are not available, improving application availability.

Collecting job-related information with System Management Facility (SMF)

This article gives an overview of how to enable and use the System Management Facility (SMF) to collect and record system and job-related information.

SMF can be enabled to collect and record system and job-related information on the WebSphere for z/OS system. This information can be used to bill users, report system reliability, analyze your configuration, schedule work, identify system resource usage, and perform other performance-related tasks that your organization may require.

You can enable SMF recording for:

- Capacity planning, to determine:
  - How many transactions have run?
  - What is the average and maximum completion time for methods running on each server?
  - How many clients are attached to each server instance? Of these clients, how many are active?
- Application profiling:
  - To show an application broken down into its component parts.
  - To provide timing information on the application's component parts.
- Error reporting:
  - To detect and record soft failures (those that are generated through an exception or those that are performance-related).
  - To use this error information to trigger an event that will cause an action to occur once a threshold has been reached.

Setting up SMF

Here is a sample SMFPRMxx member that will create interval records every 2 minutes, and record the following SMF record types:

- 30 - Address space
- 70-79 - RMF
- 82 - Crypto

- 88-90 - System Logger, Usage & System Data
- 101 - DB2
- 110 - CICS
- 120 - WebSphere

```
ACTIVE                       /*ACTIVE SMF RECORDING*/
    DSNAME(&SYSNAME..MAN1, &SYSNAME..MAN2)       /*TWO MAN DATASETS */
    LISTDSN                      /* LIST DATA SET STATUS AT IPL*/
    NOPROMPT                     /* DON'T PROMPT THE OPERATOR  */
    INTVAL(02)                   /* SMF GLOBAL RECORDING INTERVAL */
    SYNCVAL(00)                  /* GLOBAL SYNC VALUE             */
    MAXDORM(3000)                /* WRITE AN IDLE BUFFER AFTER 30 MIN*/
    STATUS(010000)               /* WRITE SMF STATS AFTER 1 HOUR*/
    SID(&SYSNAME(1:4))           /* USE SYSNAME AS SID     */
    SUBSYS(STC,INTERVAL(SMF,SYNC),
                          TYPE(0,30,70:79,88:90,101,110,120))
```

Set the SMF recording interval to 2 minutes by using the 'SET SMF=xx' command to activate the SMFPRMxx member from SYSx.PARMLIB. Use the 'D SMF,O' command to display the parameters in effect.

Use a tool like WSWS to simulate an application stress load.

While the transactions are running, switch to SDSF and RMF to observe the transactions.

Enabling SMF recording

Perform the following steps to enable SMF recording for WebSphere Application Server and select SMF type 120 records for output to the SMF data sets. For an overview of SMF recording, see Chapter 1 of z/OS MVS System Management Facilities (SA22-7630).

1. Use the WebSphere Application Server administrative console to enable properties for specific record types (see "Using the WebSphere Application Server administrative console to enable properties for specific SMF record types" for details).
2. Edit the SMFPRMxx parmlib member and update the SYS or SUBSYS(STC,...) statement to include the type 120 record.
   `SUBSYS(STC,EXITS(IEFU29,IEFACTRT),INTERVAL(SMF,SYNC),`
   `TYPE(0,30,70:79,88,89,120,245))`
3. Use the SET command to indicate which SMF parmlib member the system should use. You must issue the SET command before you start WebSphere Application Server. If you issue the command after the application server has started, SMF type 120 records will not be collected.
4. Format the output data set.

You have successfully enabled SMF recording when the SMF data is recorded in the data set which is specified in SMFPRMxx.

Related information

MVS System Management Facilities (SMF)

Using the WebSphere Application Server administrative console to enable properties for specific SMF record types:

Ensure that you have proper access to the administrative console.

To enable SMF you must first use the WebSphere Application Server administrative console to enable properties for specific record types. To view or set properties using the WebSphere Application Server administrative console:

1. Click **Server>Application Servers** in the navigation tree. The Application Servers page appears.
2. Click the **application server name** in the **Name** column of the Application Server collection table. The configuration panel of the application server selected appears.
3. On the configuration panel, under the Additional Properties section, click on **Custom Properties**.
4. To enable SMF type 120 records, click New, and specify one or more of the following properties:
   - name = server_SMF_server_activity_enabled = 1 (or server_SMF_server_activity_enabled = true)
   - name = server_SMF_server_interval_enabled = 1 (or true)
   - name = server_SMF_container_activity_enabled = 1 (or true)
   - name = server_SMF_container_interval_enabled = 1 (or true)
   - name = server_SMF_interval_length, value=n, where n is the interval, in seconds, that the system will use to write records for a server instance. Set this value to 0 to use the default SMF recording interval.
5. Click **OK** or **Apply**.
6. Save the changes and make sure a file sync is performed before restarting the servers.
7. For the changes to take effect, restart the application server.

You have successfully activated SMF recording for WebSphere Application Server when SMF type 120 records are being recorded.

Editing the SMFPRMxx parmlib member:

Make a working copy of the sample PARMLIB member SMFPRMYL.

Follow these steps to edit the SMFPRMxx parmlib member and enable SMF recording for WebSphere Application Server:

1. Insert an ACTIVE statement to indicate SMF recording. See *z/OS MVS Initialization and Tuning Guide* for more information.
2. Insert a SYS statement to indicate the types of SMF records you want the system to create. For example, use SYS(TYPE(120:120)) to select WebSphere Application Server type 120 records only. Keep the number of selected record types small to minimize the performance impact.
3. You can specify the interval in which you want the Server and Container interval records created (if no interval was specified in administrative console for the server or container definition) using the INTVAL (mm) statement in the SMFPRMxx parmlib member . The default SMF recording interval is 30 minutes. See *z/OS MVS Initialization and Tuning Reference* for more information.

   The server and container interval records will use either:
   - The value specified in the server/container definition as specified in the administrative console
   - The interval specified in the SMF parmlib member (from the SMF product settings) if you specify a length of 0.

Related information

z/OS MVS Initialization and Tuning Guide

z/OS MVS Initialization and Tuning Reference

Writing records to DASD:

Make sure you have your modified PARMLIB member SMFPRMxx.

Follow this step to start writing records to DASD:

Issue the following command: t smf=xx where xx is the suffix of the SMF parmlib member (SMFPRMxx). See *z/OS MVS System Management Facilities (SMF), SA22-7630* for more information.

Writing records to DASD has been completed successfully when the data is recorded in the data set which is specified in SMFPRMxx.

Related tasks

"Editing the SMFPRMxx parmlib member" on page 138

Related information

z/OS MVS Initialization and Tuning Guide

z/OS MVS Initialization and Tuning Reference

Formatting the output data set

Make sure SMF recording is running.

Perform the following steps to format the SMF recording output data set into a readable format for printing to the screen or other output device:

1. Switch the SMF data sets by entering i smf from the MVS console to switch the SMF data sets.
2. Run the SMF Dump program (IFASMFDP) to create a sequential data set from the raw dump. A sample JCL is shown in *z/OS MVS System Management Facilities (SMF), SA22-7630*.

You have successfully formatted the output data set when SMFDUMP ends with return code 0.

Related tasks

"Viewing the output data set"

Viewing the output data set

The data set should be viewed using a program that can display record type 120.

The Java SMF Record Interpreter is provided in the form of a jar file named bbomsmfv.jar. To use it from the z/OS or OS/390 UNIX environment:

1. Verify that the *JAVA_HOME* environment variable refers to the current java installation. JAVA_HOME=../usr/bin/java/J1.3 This should be at least Java 1.3 since this release is the first to implicitly contain the necessary record support needed by the interpreter.
2. Download the SMF Record Interpreter from the WebSphere for z/OS web site at: http://www.ibm.com/software/webservers/appserv/zos_os390/index.html
3. Copy the file bbomsmfv.jar to your tools directory. Be sure that any edits made to the file in the future are made to both copies of the file, or just execute from the installation directory in the first place.

4. To interpret SMF data from a cataloged z/OS or OS/390 sequential file named ″USER.SMFDATA″ (which was previously created using the IFASMFDP utility as described above), execute: java -cp bbomsmfv.jar com.ibm.ws390.sm.smfview.Interpreter ″USER.SMFDATA″ It is implicit in the java command parameterization that your current working directory is the tools directory. If this is not the case, you will receive a NoClassDefFoundError on com.ibm.ws390.sm.smfview.Interpreter. Java doesn't generate a diagnostic when it doesn't find bbomsmfv.jar in the current directory.

The SMF ViewTool has been successfully installed and invoked when you do not receive any Java error messages after the invocation and the Browser output is shown on the screen.

Related tasks

"Formatting the output data set" on page 139

Example of SMF Browser output:

Example of SMF Browser output

The SMF Browser available on the WebSphere for z/OS download site is able to display record type 120. To download the SMF Browser go to: http://www6.software.ibm.com/dl/websphere20/zosos390-p. For further information on the SMF Browser, download the browser package and read the associated documentation.

The following example shows sample output from the SMF Browser. The example features subtype 7 and subtype 8, in that order.

```
Record#: 14;
 Type: 120; Size: 820; Date: Fri Nov 23 04:54:17  EST 2001;
 SystemID: SY1; SubsystemID: WAS; Flag: 94;
 Subtype: 7 (WEB CONTAINER ACTIVITY);
# Triplets: 4;
Triplet #: 1; offset: 76; length: 32; count: 1;
Triplet #: 2; offset: 108; length: 140; count: 1;
Triplet #: 3; offset: 264; length: 556; count: 1;
Triplet #: 1; Type: ProductSection;
  Version: 1; Codeset: Unicode; Endian: 1; TimeStampFormat: 1 (S390STCK64);
  IndexOfThisRecord: 1; Total # records: 1; Total # triplets: 4;
Triplet #: 2; Type: WebContainerActivitySection;
  HostName : PLEX1;
  ServerName : BBOASR4;
  ServerInstanceName: BBOASR4A;
  WlmEnclaveToken * 00000020 00000242 -------- --------
                  * ^... * p1047
  ActivityID * b6c7a7b7 14e9bc85 000000b0 00000007
             * 0926306b -------- -------- --------
             *,............ *Cp1047
             ActivityStartTime * b6c7a7b7 14e9bc85 40404040 40404040 *
             ActivityStopTime * b6c7a7b7 53a8a645 40404040 40404040 *
Triplet #: 3; Type: HttpSessionManagerActivitySection;
  # http sessions created: 0; # http sessions invalidated: 0;
  # http sessions active: 0;
Average session life time: 0 [sec*10**-3];
 Triplet #: 4; Type: WebApplicationActivitySection;
  Name: PolicyIVP-localhost_1;
  # Servlets: 1;
  Triplet #: 4.1; offset: 272; length: 284; count: 1;
  Triplet #: 4.1; Type: ServletActivitySection;
   Name: SimpleFileServlet;
   ResponseTime: 48 [sec*10**-3];
   # errors: 0;
```

```
   Loaded by this request: 0;
   Loaded since (raw): ea54948e0d;
   Loaded since: Thu Nov 22 10:02:49 EST 2001;
Record#: 72;
  Type: 120; Size: 1744; Date: Fri Nov 23 05:01:02 EST 2001;
  SystemID: SY1; SubsystemID: WAS; Flag: 94;
  Subtype: 8 (WEB CONTAINER INTERVAL);
  # Triplets: 4;
  Triplet #: 1; offset: 76; length: 32; count: 1;
  Triplet #: 2; offset: 108; length: 112; count: 1;
  Triplet #: 3; offset: 264; length: 1480; count: 1;
  Triplet #: 1; Type: ProductSection;
    Version: 1; Codeset: Unicode; Endian: 1; TimeStampFormat: 1 (S390STCK64);
    IndexOfThisRecord: 1; Total # records: 1; Total # triplets: 4;
  Triplet #: 2; Type: WebContainerIntervalSection;
    HostName : PLEX1;
    ServerName : BBOASR4;
    ServerInstanceName: BBOASR4A;
    SampleStartTime * b6c7a6fd 655c0604 40404040 40404040 *
    SampleStopTime * b6c7a939 9a0e614c 40404040 40404040 *
Triplet #: 3; Type: HttpSessionManagerIntervalSection;
    http sessions #created: 1; #invalidated: 0;
    http sessions #active: 0; Min #active: 0; Max #active: 0;
    Average session life time: 0;
    Average session invalidate time: 0;
    http sessions #finalized: 0; #tracked: 0;
    http sessions #min live: 0; #max live: 0;
  Triplet #: 4; Type: WebApplicationIntervalSection;
    Name: PolicyIVP-localhost_1;
    # Servlets loaded: 0;
    # Servlets: 4;
    Triplet #: 4.1; offset: 312; length: 292; count: 1;
    Triplet #: 4.2; offset: 604; length: 292; count: 1;
    Triplet #: 4.3; offset: 896; length: 292; count: 1;
    Triplet #: 4.4; offset: 1188; length: 292; count: 1;
    Triplet #: 4.1; Type: ServletIntervalSection;
      Name: SimpleFileServlet;
      # requests: 6;
      AverageResponseTime: 764 [sec*10**-3];
      MinimumResponseTime: 18 [sec*10**-3];
      MaximumResponseTime: 4133 [sec*10**-3];
      # errors: 0;
      Loaded since (raw): ea54948e0d;
      Loaded since: Thu Nov 22 10:02:49 EST 2001;
    Triplet #: 4.2; Type: ServletIntervalSection;
      Name: Was40Ivp;
      # requests: 4;
      AverageResponseTime: 4664 [sec*10**-3];
      MinimumResponseTime: 1584 [sec*10**-3];
      MaximumResponseTime: 12572 [sec*10**-3];
      # errors: 0;
      Loaded since (raw): ea58a1509e;
      Loaded since: Fri Nov 23 04:55:14 EST 2001;
    Triplet #: 4.3; Type: ServletIntervalSection;
      Name: /cebit.jsp;
      # requests: 1;
      AverageResponseTime: 204 [sec*10**-3];
      MinimumResponseTime: 204 [sec*10**-3];
      MaximumResponseTime: 204 [sec*10**-3];
      # errors: 0;
      Loaded since (raw): ea58a24a69;
      Loaded since: Fri Nov 23 04:56:18 EST 2001;
    Triplet #: 4.4; Type: ServletIntervalSection;
      Name: JSP 1.1 Processor;
      # requests: 1;
      AverageResponseTime: 482 [sec*10**-3];
      MinimumResponseTime: 482 [sec*10**-3];
```

```
MaximumResponseTime: 482 [sec*10**-3];
# errors: 0;
Loaded since (raw): ea54948b66;
Loaded since: Thu Nov 22 10:02:48 EST 2001;
```

Disabling SMF recording for WebSphere Application Server

Ensure that you have proper access to the administrative console.

SMF recording can be enabled for WebSphere Application Server, and for z/OS. The following steps describe how to disable SMF recording for WebSphere Application Server:

1. Click **Server>Application Servers** in the navigation tree. The Application Servers page appears.
2. Click the **application server name** in the **Name** column of the Application Server collection table. The configuration panel of the application server selected appears.
3. On the configuration panel, under the Additional Properties section, click on **Custom Properties**.
4. To disable SMF type 120 records, set the following properties to false:
   - name = server_SMF_server_activity_enabled = 0 (or server_SMF_server_activity_enabled = false)
   - name = server_SMF_server_interval_enabled = 0 (or false)
   - name = server_SMF_container_activity_enabled = 0 (or false)
   - name = server_SMF_container_interval_enabled = 0 (or false)

   Alternatively, you could delete the SMF related properties. However, it will be easier for you to enable SMF recording later if you keep the properties in place and just change their values to false.
5. Click **OK** or **Apply**.
6. Save the changes and make sure a file sync is performed before restarting the servers.
7. For the changes to take effect, restart the application server.

You have successfully disabled SMF recording for WebSphere Application Server when SMF records of records type 120 are no longer being recorded.

Related tasks

"Using the WebSphere Application Server administrative console to enable properties for specific SMF record types" on page 137

Disabling SMF recording for the entire MVS system

Make sure that you have your own working copy of SMFPRMxx and SMF is running.

SMF recording can be enabled for WebSphere Application Server and for z/OS. The following steps describe how to disable SMF recording for your MVS System (z/OS):

Edit the SMFPRMxx parmlib member and set SMFPRMxx to ″NOACTIVE″ which will disable the writing of SMF records to DASD. Use the SET command to activate that SMF parmlib member on the MVS system.

SMF recording has successfully been disabled for the whole MVS system when SMF records for z/OS and WebSphere Application Server are no longer being written to DASD.

Related tasks

"Editing the SMFPRMxx parmlib member" on page 138

Overview of SMF record type 120

Information resulting from the SMF data gathering process for WebSphere Application Server for z/OS is held in SMF record type 120. This information is typically presented with the help of an SMF data viewing tool. This record format description is intended to enable your tool providers to design an SMF data viewing tool. Your system administrators will use an SMF data viewing tool with a description presented by your tool provider, since it requires them to make proper selections that limit the amount of presentation data. For example, they might want to view a specific time frame and only specific containers, classes, and methods. They may also occasionally need to refer to the record descriptions.

Two types of SMF records can be produced: activity records and interval records.

- Activity records are gathered as each activity within a server is completed. An activity is a logical unit of business function. It can be a server or user-initiated transaction.
- Interval records consist of data gathered at installation-specified intervals and provide capacity planning and reliability information.

Six records can be produced:

- the Server Activity record: Subtype 1
- the Server Interval record: Subtype 3
- the J2EE Container Activity Record: Subtype 5
- the J2EE Container Interval Record: Subtype 6
- the WebContainer Activity record: Subtype 7
- the WebContainer Interval record: Subtype 8

For additional information about using SMF records, see *z/OS MVS System Management Facilities (SMF), SA22-7630*.

Viewing records with the SMF Browser

The SMF Browser available on the WebSphere for z/OS download site is able to display record type 120. To download the SMF Browser go to: http://www6.software.ibm.com/dl/websphere20/zosos390-p. For further information on the SMF Browser, download the browser package and read the associated documentation.

Related information

MVS System Management Facilities (SMF)

Record Type 120 (78) - WebSphere Application Server performance statistics:

WebSphere Application Server writes record type 120 to collect WebSphere Application Server performance statistics. For more information about SMF record types, see *z/OS MVS System Management Facilities (SMF)*.

All subtypes of the record type 120 have the following format:

- Standard header section
- Individual header extension for subtype x
- Product section
- Subtype-specific sections listed below.

Record type 120 has the following subtypes:

- **Subtype 1: Server activity record**
  - **Server activity section** (one section per record):

    Contains information about each activity that occurred within one server.
  - **Communication session section** (zero, one, or multiple sections per record):

    Contains information about each communication session.
  - **JVM heap section** (zero, one, or multiple sections per record):

    Contains information about the heap in a server region.
- **Subtype 3: Server interval record**
  - **Server interval section** (one section per record):

    Contains aggregated information about all activities that occurred within the specified server interval.
  - **Server region section** (zero, one, or multiple sections per record):

    Contains information about server regions in the specified interval.
- **Subtype 5: J2EE container activity record**
  - **J2EE container activity section** (one section per record):

    Contains information about each activity that occurred within one J2EE container.
  - **Bean section** (multiple (0..n) sections per record):

    Contains information about all beans involved in this activity.
  - **Bean method section** (multiple (0..n) sections per bean section):

    Contains information about all methods of this bean involved in this activity.
- **Subtype 6: J2EE container interval record**
  - **J2EE container interval section** (one section per record):

    Contains aggregated information about all activities that occurred within one J2EE container in the specified interval.
  - **Bean section** (multiple (0..n) sections per record, see subtype 5):

    Contains information about all beans involved in this activity in the specified interval.
  - **Bean method section** (multiple (0..n) sections per bean section, see subtype 5):

    Contains information about all methods of this class involved in this activity in the specified interval.
- **Subtype 7: WebContainer activity record (Version 2)**
  - **WebContainer activity section** (one section per record):

    Contains information about each activity that occurred within one WebContainer.
  - **HttpSessionManager activity section** (one section per record):

    Contains information about all sessions involved in this activity.
  - **WebApplication section** (multiple (0..n) sections per record):

    Contains information about all WebApplications involved in this activity.
  - **Servlet section** (multiple (0..n) sections per WebApplication section):

    Contains information about all Servlets involved in this activity.
- **Subtype 8: WebContainer interval record (Version 2)**
  - **WebContainer interval section** (one section per record):

    Contains information about each activity that occurred within one WebContainer.

- **HttpSessionManager section** (one section per record):
  Contains information about all sessions involved in this activity.
- **WebApplication section** (multiple (0..n) sections per record):
  Contains information about all WebApplications involved in this activity.
- **Servlet section** (multiple (0..n) sections per WebApplication section):
  Contains information about all Servlets involved in this activity.

Triplets:

You can use triplets to build self-describing SMF records that contain various types of data sections and a varying number of each of these sections. All data sections are described by triplets that consist of:

1. An offset that specifies the start position of the data
2. A length that describes the length of the section
3. A count that describes how many instances of the section are included in this record.

The two triplets that describe the product section and the general record information section (for example, the section describing the container itself in a container activity record) are located at fixed positions within the record. This allows one to start evaluating the record right after having evaluated the record header.

Related reference

"Splitting SMF records"

"Header/self-defining section" on page 146

Splitting SMF records:

Since most of the WebSphere Application Server SMF records are used to describe variable-length data structures (for example, there might be hundreds of classes by container and hundreds of methods by class), the SMF records may be larger than the maximum record size supported by SMF (32KB). In this case, the logical records need to be split into several physical records.

Each of those physical records needs to be self-describing and self-contained. *Self-describing* indicates what we described in the paragraph on triplets before; it is a purely mechanical structure to help read a record. *Self-contained* indicates that, even if we have only a subset of the physical records at hand that together describe the original logical record, we need to be able to evaluate these records, combine the information stored in them, and set an 'incomplete' flag. This is required since, as we break up a logical record into physical records and write them to SMF one after the other, SMF might decide that only the first few physical records fit into the primary SMF dump dataset whereas the remaining physical records are written into an alternate SMF dump dataset. At the time when a formatted SMF dump dataset is evaluated, we may not assume that all physical records that make up one logical record are present. For example, self-containedness of a physical container activity record means that it contains the description of the container, but not necessarily all of its classes.

We use a similar splitting mechanism like the one that is currently used in the RMF product. Note that in the case of container records (subtype 2 and 4), we cannot assume that records will be split at a class boundary, but we must consider the case when the methods that belong to one class also need to be split over multiple physical records.

Note: The section length numbers used throughout the following diagrams are only for demonstrative purposes. In particular, the arrows indicating 32K boundaries or the total length of the records are placed at random. You can fit many more classes and methods into a physical record than suggested by the diagrams.

Related reference

"Triplets" on page 145

"Header/self-defining section"

Record environment and mapping:
Record environment

The following conditions exist for the generation of this record:

- **Macro** SMFWTM (record exit: IEFU83)
  Mode Task
  Addressing mode
  31-bit

Record mapping

For a description of the common SMF record header fields and the triplet fields (offset/length/number), if applicable, that locate the other sections on the record, see Header/Self-defining section.

For a description of triplets, see Using Triplets and MVS System Management Facilities (SMF)" (SA22-7630).

Related reference

"Triplets" on page 145

Related information

MVS System Management Facilities (SMF)

Header/self-defining section:

The tables below describe the header/self-defining section of an SMF record.

Offset (decimal)	Offset (hexadecimal)	Name	Length	Format	Description
0	0	SM120LEN	2	binary	Record length. This field and the next field (total of four bytes) form the RDW (record descriptor word). See "Standard SMF record header" in MVS System Management Facilities (SMF)" (SA22-7630), for a detailed description.
2	2	SM120SEG	2	binary	Segment descriptor (see record length field)

4	4	SM120FLG	1	binary	**Bit meaning when set** 0: New SMF record format 1: Subtypes used 2: Reserved 3-6: Version indicators* 7: Reserved *See "Standard SMF record header" in MVS System Management Facilities (SMF)" (SA22-7630), for a detailed description.
5	5	SM120RTY	1	binary	Record type 120(X'78')
6	6	SM120TME	4	binary	Time since midnight, in hundredths of a second, that the record was moved into the SMF buffer.
10	A	SM120DTE	4	packed	Date when the record was moved into the SMF buffer, in the form 0*cyydddF*. See "Standard SMF record header" in MVS System Management Facilities (SMF)" (SA22-7630), for a detailed description.
14	E	SM120SID	4	EBCDIC	System identification (from the SMFPRMxx SID parameter)
18	12	SM120SSI	4	EBCDIC	Subsystem identification from SUBSYS parameter
22	16	SM120RST	2	binary	Record subtype: 1: Server activity 2: Container activity 3: Server interval 4: Container interval. 5: J2EE container activity 6: J2EE container interval 7: WebContainer activity 8: WebContainer interval
24	18	SM120TRN	4	binary	Number of triplets in this record. A triplet is a set of three SMF fields (offset/length/number values) that defines a section of the record. The offset is the offset from the RDW. Subtypes: 1: Value is equal to the number of sessions +2 2 and 4: Value is equal to the number of classes +2.
28	1C	SM120PRS	4	binary	Offset to product section from RDW.
32	20	SM120PRL	4	binary	Length of product section.
36	24	SM120PRN	4	binary	Number of product sections.

Individual header extension for subtype 1

40	28	SM120SAS	4	binary	Offset to server activity section from RDW
44	2C	SM120SAL	4	binary	Length of server activity section
48	30	SM120SAN	4	binary	Number of server activity sections
52	34	SM120CSS	4	binary	Offset to communication session section from RDW
56	38	SM120CSL	4	binary	Length of communication session section
60	3C	SM120CSN	4	binary	Number of communication session sections
64	40	SM120JHS	4	binary	Offset to JVM heap section from RDW
68	44	SM120JHL	4	binary	Length of JVM heap section
72	48	SM120JHN	4	binary	Number of jvm heap sections

Individual header extension for subtype 3					
40	28	SM120SIS	4	binary	Offset to server interval section from RDW
44	2C	SM120SIL	4	binary	Length of server interval section
48	30	SM120SIN	4	binary	Number of server interval sections
The following triplet appears 0-n times; once for each server region section.					
52	34	SM120SRS	4	binary	Offset to server region section from RDW
56	38	SM120SRL	4	binary	Length of server region section
60	3C	SM120SRN	4	binary	Number of server region sections

Individual header extension for subtype 5					
40	28	SM120JA1	4	binary	Offset to J2EE container activity section from RDW
44	2C	SM120JA2	4	binary	Length of J2EE container activity section
48	30	SM120JA3	4	binary	Number of J2EE container activity sections

The following triplet appears 0-n times; once for each bean section.					
52	34	SM120JAS	4	binary	Offset to bean section from RDW
56	38	SM120JAL	4	binary	Length of bean section
60	3C	SM120JAN	4	binary	Number of bean sections

Individual header extension for subtype 6					
40	28	SM120JI1	4	binary	Offset to J2EE container interval section from RDW
44	2C	SM120JI2	4	binary	Length of J2EE container interval section
48	30	SM120JI3	4	binary	Number of J2EE container interval sections

The following triplet appears 0-n times; once for each bean section.					
52	34	SM120JIS	4	binary	Offset to bean section from RDW

56	38	SM120JIL	4	binary	Length of bean section
60	3C	SM120JIN	4	binary	Number of bean sections
Individual header extension for subtype 7					
40	28	SM120WA1	4	binary	Offset to WebContainer activity section from RDW.
44	2C	SM120WA2	4	binary	Length of WebContainer activity section.
48	30	SM120WA3	4	binary	Number of WebContainer activity sections.
52	34	SM120WA4	4	binary	Offset to HttpSessionManager activity section from RDW.
56	38	SM120WA5	4	binary	Length of HttpSessionManager activity section.
60	3C	SM120WA6	4	binary	Number of HttpSessionManager activity sections.
The following triplet appears 0-n times, once for each WebApplication section.					
64	40	SM120WA7	4	binary	Offset to WebApplication section from RDW.
68	44	SM120WA8	4	binary	Length of WebApplication section.
72	48	SM120WA9	4	binary	Number of WebApplication sections.
Individual header extension for subtype 8					
40	28	SM120WI1	4	binary	Offset to WebContainer interval section from RDW.
44	2C	SM120WI2	4	binary	Length of WebContainer interval section.
48	30	SM120WI3	4	binary	Number of WebContainer interval sections.
52	34	SM120WI4	4	binary	Offset to HttpSessionManager interval section from RDW.
56	38	SM120WI5	4	binary	Length of HttpSessionManager interval section.
60	3C	SM120WI6	4	binary	Number of HttpSessionManager interval sections.
The following triplet appears 0-n times, once for each WebApplication section.					
64	40	SM120WI7	4	binary	Offset to WebApplication section from RDW.
68	44	SM120WI8	4	binary	Length of WebApplication section.
72	48	SM120WI9	4	binary	Number of WebApplication sections.

Related reference

"Triplets" on page 145

"Product section" on page 150
This section includes the header/self-defining and product sections.

"Subtype 1: Server activity record" on page 150

"Subtype 3: Server interval record" on page 153

"Subtype 5: J2EE container activity record (Version 2)" on page 156

"Subtype 6: Container interval record (Version 2)" on page 158

"Subtype 7: WebContainer activity record (Version 2)" on page 160

"Subtype 8: WebContainer interval record (Version 2)" on page 162

Related information

MVS System Management Facilities (SMF)

Product section:

This section includes the header/self-defining and product sections.

Product section

Offset	Offset	Name	Length	Format	Description
0	0	SM120MFV	4	binary	CB SMF version
4	4	SM120COD	8	EBCDIC	Character codeset in which strings in the SMF record are encoded
12	C	SM120END	4	binary	Encode of numbers in the SMF record
16	10	SM120TSF	4	binary	Encoding of timestamps: 1: S390STCK64: The time values are encoded in 64-bit S/390 Store Clock format.

Reassembly information.					
20	14	SM120IXR	4	binary	Index of this record
24	18	SM120NRC	4	binary	Total number of records
28	1C	SM120NTR	4	binary	Total number of triplets

Related reference

"Triplets" on page 145
"Header/self-defining section" on page 146
"Subtype 1: Server activity record"
"Subtype 3: Server interval record" on page 153
"Subtype 5: J2EE container activity record (Version 2)" on page 156
"Subtype 6: Container interval record (Version 2)" on page 158
"Subtype 7: WebContainer activity record (Version 2)" on page 160
"Subtype 8: WebContainer interval record (Version 2)" on page 162

Related information

MVS System Management Facilities (SMF)

Subtype 1: Server activity record:

The server activity SMF record is used to record activity that is running inside a WebSphere Application Server for z/OS. This record can be used to perform basic charge-back accounting and to profile your applications to determine, in detail, what is happening inside the WebSphere Application Server transaction server.

A single record is created for each activity that is run inside a server or server instance. If the activity runs in multiple servers, then a record is written for each server.

You can activate this record through the administrative console by setting **server_SMF_server_activity_enabled=1 (or**

server_SMF_server_activity_enabled=true). See "Using the WebSphere Application Server administrative console to enable properties for specific SMF record types" on page 137 for instructions.

Server activity record schema

This section includes Subtype 1: Server activity record.

Server activity section

The Server activity section contains information about each activity that occurred within one server.

Offset (decimal)	Offset (hexadecimal)	Name	Length	Format	Description
0	0	SM120HNM	64	EBCDIC	WebSphere for z/OS transaction server host name
64	40	SM120SNA	8	EBCDIC	WebSphere for z/OS transaction server name
72	48	SM120INA	8	EBCDIC	WebSphere for z/OS transaction server instance name
80	50	SM120SNM	4	binary	Total number of server servants that were involved to process this activity. If applicable, up to the first five server servant address space IDs are listed within the next five fields.
84	54	SM120SR1	4	binary	The specific WebSphere for z/OS transaction server instance server servant where the request ran
88	58	SM120SR2	4	binary	The specific WebSphere for z/OS transaction server instance server servant where the request ran
92	5C	SM120SR3	4	binary	The specific WebSphere for z/OS transaction server instance server servant where the request ran
96	60	SM120SR4	4	binary	The specific WebSphere for z/OS transaction server instance server servant where the request ran
100	64	SM120SR5	4	binary	The specific WebSphere for z/OS transaction server instance server servant where the request ran
104	68	SM120CRE	8	EBCDIC	The user credentials under which the activity began.
112	70	SM120ATY	4	binary	Type of activity that this record references: 1: Method request: This record refers to a method request that is not part of a global transaction. 2: Transaction: This record refers to a transaction.
116	74	SM120AID	20	HEX	Identity of the activity
136	88	SM120WLM	8	HEX	WLM enclave token
144	90	SM120AST	16	S390STCK	Activity start time
160	A0	SM120AET	16	S390STCK	Activity stop time
176	B0	SM120NIM	4	binary	Number of input methods
180	B4	SM120NGT	4	binary	Number of global transactions that were started in the server servant
184	B8	SM120NLT	4	binary	Number of local transactions that were started in the server servant
188	BC	SM120J2E	4	binary	J2EE server
192	C0	SM120CEL	8	EBCDIC	WebSphere for z/OS cell name

200	C8	SM120NOD	8	EBCDIC	WebSphere for z/OS node name
208	D0	SM120WCP	8	binary	Total CPU time accumulated by the WLM enclave. TOD clock format (bit 51 = microseconds).

Communications session section

There are zero, one, or multiple sections per record. The Communications session section contains information about each communication session.

Offset (decimal)	Offset (hexadecimal)	Name	Length	Format	Description
0	0	SM120CSH	8	HEX	Communications session handle
8	8	SM120CSA	64	EBCDIC	Communications session address
72	48	SM120CSO	4	binary	Communications session optimization 1: Local communications session: The session is a local OS/390 optimized communications session. 2: Remote communications session: The session is a remote communications session. 3: Remote encrypted (SSL) 4: Remote within sysplex. 5: HTTP session. 6: HTTP encrypted session.
76	4C	SM120SDR	4	binary	Data received; the number of bytes received by the server. 'FFFFFFFF'X indicates the 4-byte field is too small. Use SM120CDR, an 8-byte field, instead.
80	50	SM120SDT	4	binary	Data transferred; the number of bytes transferred from the server back to the client. 'FFFFFFFF'X indicates the 4-byte field is too small. Use SM120CDT, an 8-byte field, instead.
84	54	SM120CDR	8	binary	Data received; the number of bytes received by the server.
92	5C	SM120CDT	8	binary	Data transferred; the number of bytes transferred from the server back to the client.

JVM Heap section

There are zero, one, or multiple sections per record. The JVM heap section contains information about the heap in each server servant.

The information in the JVM heap section comes from the QueryGCStatus() JNI function.

Offset (decimal)	Offset (hexadecimal)	Name	Length	Format	Description
0	0	SM120JHA	4	binary	Servant address space ID
4	4	SM120JHH	4	binary	The heap for which the following data applies.
8	8	SM120JHC	4	binary	The total number of allocation failures on this heap or, if querying shared storage, the subpool identifier. A negative value indicates the information is for the shared memory page pool.
12	C	SM120JHF	8	binary	The total number of free bytes in the heap/subpool/page pool.

20	14	SM120JHT	8	binary	The total number of bytes in the heap/subpool/page pool.

Related reference

"Triplets" on page 145

"Header/self-defining section" on page 146

"Product section" on page 150
This section includes the header/self-defining and product sections.

"Subtype 3: Server interval record"

"Subtype 5: J2EE container activity record (Version 2)" on page 156

"Subtype 6: Container interval record (Version 2)" on page 158

"Subtype 7: WebContainer activity record (Version 2)" on page 160

"Subtype 8: WebContainer interval record (Version 2)" on page 162

Related information

MVS System Management Facilities (SMF)

Subtype 3: Server interval record:

The purpose of the server interval SMF record is to record activity that is running inside a WebSphere Application Server for z/OS. This record is produced at regular intervals and is an aggregate of the work that ran inside the server instance during the interval.

A single record is created for each server instance that has interval recording active during the interval. If a server has multiple server instances, then a record for each server instance is written and the records must be merged after processing to get a complete view of the work that ran inside the server.

You can activate this record through the administrative console by setting **server_SMF_server_interval_enabled=1 (or server_SMF_server_interval_enabled=true)**. You can specify an interval through the administrative console by setting server_SMF_interval_length=n, where n is the desired number of seconds.

Server interval record schema

This section includes Subtype 3: Server interval record.

Server interval section

The server interval section contains information about each activity that occurred within one server.

Offset (decimal)	Offset (hexadecimal)	Name	Length	Format	Description
0	0	SM120HN2	64	EBCDIC	WebSphere for z/OS transaction server host name
64	40	SM120SNI	8	EBCDIC	WebSphere for z/OS transaction server name
72	48	SM120INI	8	EBCDIC	WebSphere for z/OS transaction server instance name
80	50	SM120SST	16	S390STCK	Time that the sample began in the server
96	60	SM120SET	16	S390STCK	Time that the sample ended

112	70	SM120NG2	4	binary	Number of global transactions that have run through the server instance during the interval that have been initiated by the server instance during the interval
116	74	SM120NL2	4	binary	Number of local transactions that have been initiated by the server instance during the interval
120	78	SM120NCS	4	binary	Reserved
124	7C	SM120NCA	4	binary	The number of communications sessions that have been active during the interval
128	80	SM120NLS	4	binary	Reserved
132	84	SM120NLA	4	binary	Number of active local communication sessions that have been attached and active within the server instance during the interval
136	88	SM120NRS	4	binary	Reserved
140	8C	SM120NRA	4	binary	Number of active remote communication sessions that have been attached and active within the server instance during the interval
144	90	SM120BTS	4	binary	Number of bytes that have been transferred to the server from all attached clients 'FFFFFFFF'X indicates the 4-byte field is too small. Use SM120ITS, an 8-byte field, instead.
148	94	SM120BFS	4	binary	Number of bytes that have been sent from the server to all attached clients 'FFFFFFFF'X indicates the 4-byte field is too small. Use SM120IFS, an 8-byte field, instead.
152	98	SM120BTL	4	binary	Number of bytes that have been transferred to the server from all locally attached clients 'FFFFFFFF'X indicates the 4-byte field is too small. Use SM120ITL, an 8-byte field, instead.
156	9C	SM120BFL	4	binary	Number of bytes that have been transferred from the server to all locally attached clients 'FFFFFFFF'X indicates the 4-byte field is too small. Use SM120IFL, an 8-byte field, instead.
160	A0	SM120BTR	4	binary	Number of bytes that have been transferred to the server from all remotely attached clients 'FFFFFFFF'X indicates the 4-byte field is too small. Use SM120ITR, an 8-byte field, instead.
164	A4	SM120BFR	4	binary	Number of bytes that have been transferred from the server to all remotely attached clients 'FFFFFFFF'X indicates the 4-byte field is too small. Use SM120IFR, an 8-byte field, instead.
168	A8	SM120J2	4	binary	J2EE server.
172	AC	SM120CL1	8	EBCDIC	WebSphere for z/OS transaction server cell name
180	B4	SM120ND1	8	EBCDIC	WebSphere for z/OS transaction server node name
188	BC	SM120NHS	4	binary	Reserved
192	C0	SM120NHA	4	binary	Number of HTTP communication sessions that have been attached and active within the server instance during the interval
196	C4	SM120BTH	4	binary	Number of bytes that have been transferred to the server from all HTTP attached clients 'FFFFFFFF'X indicates the 4-byte field is too small. Use SM120ITH, an 8-byte field, instead.

200	C8	SM120BFH	4	binary	Number of bytes that have been transferred from the server to all HTTP attached clients 'FFFFFFFF'X indicates the 4-byte field is too small. Use SM120IFH, an 8-byte field, instead.
204	CC	SM120TEC	8	binary	Total CPU time accumulated by the WLM enclaves. TOD clock format (bit 51 = microseconds).
212	D4	SM120ITS	8	binary	Number of bytes that have been transferred to the server from all attached clients.
220	DC	SM120IFS	8	binary	Number of bytes that have been sent from the server to all attached clients
228	E4	SM120ITL	8	binary	Number of bytes that have been transferred to the server from all locally attached clients
236	EC	SM120IFL	8	binary	Number of bytes that have been transferred from the server to all locally attached clients
244	F4	SM120ITR	8	binary	Number of bytes that have been transferred to the server from all remotely attached clients
252	FC	SM120IFR	8	binary	Number of bytes that have been transferred from the server to all remotely attached clients
260	104	SM120ITH	8	binary	Number of bytes that have been transferred to the server from all HTTP attached clients
268	10C	SM120IFH	8	binary	Number of bytes that have been transferred from the server to all HTTP attached clients
276	114	SM120IR1	32		Reserved

Server servant section

There are zero, one, or multiple sections per record. The server servant section contains information about each server servant in the specified server interval.

Offset	Offset	Name	Length	Format	Description
0	0	SM120SSA	4	binary	Servant address space ID
4	4	SM120SNT	4	binary	Number of triplets.
The following triplet appears 0-n times; once for each heap id section.					
8	8	SM120SSO	4	binary	Offset to heap id section from the beginning of this server servant section.
12	C	SM120SSL	4	binary	Length of heap id section.
16	10	SM120SSN	4	binary	Number of heap id sections.

Subtype 3: Heap ID section

There are multiple (0..n) sections per server servant section. The Heap id section contains information about all heaps of this server servant involved in this activity

Offset	Offset	Name	Length	Format	Description
0	0	SM120HIH	4	binary	The heap for which the following data applies.
4	4	SM120HIC	4	binary	Number of allocation failures on this heap during the interval.
8	8	SM120HI1	8	binary	Minimum number of bytes during the interval.
16	10	SM120HI2	8	binary	Maximum number of bytes during the interval.
24	18	SM120HI3	8	binary	Average number of bytes during the interval.

32	20	SM120HI4	8	binary	Minimum number of free bytes during the interval.
40	28	SM120HI5	8	binary	Maximum number of free bytes during the interval.
48	30	SM120HI6	8	binary	Average number of free bytes during the interval.

Related reference

"Triplets" on page 145

"Header/self-defining section" on page 146

"Product section" on page 150
This section includes the header/self-defining and product sections.

"Subtype 1: Server activity record" on page 150

"Subtype 5: J2EE container activity record (Version 2)"

"Subtype 6: Container interval record (Version 2)" on page 158

"Subtype 7: WebContainer activity record (Version 2)" on page 160

"Subtype 8: WebContainer interval record (Version 2)" on page 162

Related information

MVS System Management Facilities (SMF)

Subtype 5: J2EE container activity record (Version 2):

The purpose of the J2EE container activity SMF record is to record activity within a J2EE container that is located inside the WebSphere Application Server transaction server.

This record can be used to perform basic charge-back accounting, application profiling, problem determination, and capacity planning. A single record is created for each activity that is run within a J2EE container located inside a WebSphere Application Server transaction server.

You can activate this record through the administrative console by setting **server_SMF_container_activity_enabled=1 (or server_SMF_container_activity_enabled=true)** .

J2EE container activity record (Version 2) schema

This section includes Subtype 5: J2EE container activity record (Version 2).

J2EE container activity section

There is one section per record. The J2EE container activity section contains information about each activity that occurred within one J2EE container.

Offset	Offset	Name	Length	Format	Description
0	0	SM120JA4	64	EBCDIC	WebSphere for z/OS transaction server host name
64	40	SM120JA5	8	EBCDIC	WebSphere for z/OS transaction server name
72	48	SM120JA6	8	EBCDIC	WebSphere for z/OS transaction server instance name
80	50	SM120JA7	4	binary	The specific WebSphere for z/OS transaction server instance server servant where the request ran
84	54	SM120JA8	512	Unicode	WebSphere for z/OS container name.

596	254	SM120JA9	8	HEX	The WLM enclave token
604	25C	SM120JAA	4	binary	RESERVED
608	260	SM120JAB	20	HEX	The identity of the activity
628	274	SM120CL2	8	EBCDIC	Cell
636	27C	SM120ND2	8	EBCDIC	Node

Bean section

There are multiple sections per record. The bean section contains information about all beans involved in this activity.

Offset	Offset	Name	Length	Format	Description
0	0	SM120JB1	512	Unicode	AMCName of the bean activated by the container. **Note:** If the length of the AMCName exceeds 256 DBCS characters (512 bytes), the rightmost 256 characters are recorded.
512	200	SM120JB2	60	binary	UUID based AMC name
572	23C	SM120JB3	4	binary	The bean's type. 2: Stateless session bean. 3: Stateful session bean. 4: BMP entity bean. 5: CMP entity bean.
576	240	SM120JB4	4	binary	RESERVED
580	244	SM120JB5	4	binary	RESERVED
584	248	SM120JB6	4	binary	RESERVED
588	24C	SM120JB7	4	binary	The bean's reentrance policy. 0: Not reentrant within transaction. 1: Reentrant within transaction.
592	250	SM120JB8	4	binary	RESERVED
596	254	SM120JMC	4	binary	RESERVED
600	258	SM120JM6	4	binary	RESERVED
604	25C	SM120JB9	4	binary	Number of method triplets in this bean section
The following triplet appears 0-n times; once for each bean method section.					
608	260	SM120JBS	4	binary	Offset to bean method section from the beginning of this bean section
612	264	SM120JBL	4	binary	Length of bean method section
616	268	SM120JBN	4	binary	Number of bean method sections

Bean method section

There are multiple sections per bean section. The bean method section contains information about all methods of beans involved in this activity.

Offset	Offset	Name	Length	Format	Description
0	0	SM120JM1	1,024	Unicode	The name of the method including its signature in its externalized, human-readable form. If the length of the method exceeds 512 DBCS characters (1024 bytes), the leftmost 512 characters are recorded.
1024	400	SM120JM2	4	binary	The number of times the method was invoked during the activity.
1028	404	SM120JM3	4	binary	Average response time. The response time is measured in milliseconds (the granularity provided by the JVM - hopefully, it will be equal to 0 in most cases).

1032	408	SM120JM4	4	binary	Maximum response time. The response time is measured in milliseconds.

1036	40C	SM120JM5	4	binary	The bean method's transaction policy. Values from com.ibm.WebSphere for z/OS.csi. TransactionAttribute.java:
					`0: "TX_NOT_SUPPORTED"`
					`1: "TX_BEAN_MANAGED"`
					`2: "TX_REQUIRED"`
					`3: "TX_SUPPORTS"`
					`4: "TX_REQUIRES_NEW"`
					`5: "TX_MANDATORY"`
					`6: "TX_NEVER"`

1040	410	SM120JM8	4	binary	RESERVED.
1044	414	SM120JM9	4	binary	RESERVED.
1048	418	SM120JMA	512	Unicode	List of ejbRoles associated with the method. Separator character: ";" (semicolon). If the length of the concatenated string exceeds 256 characters (512 bytes), only its leftmost 256 characters are recorded.
1560	618	SM120JMB	4	binary	RESERVED.
1564	61C	SM120JMD	4	binary	RESERVED.
1568	620	SM120JME	4	binary	ejbLoad: # of invocations
1572	624	SM120JMF	4	binary	ejbLoad: avg execution time
1576	628	SM120JMG	4	binary	ejbLoad: max execution time
1580	62C	SM120JMH	4	binary	ejbStore: # of invocations
1584	630	SM120JMI	4	binary	ejbStore: avg execution time
1588	634	SM120JMJ	4	binary	ejbStore: max execution time
1592	638	SM120JMK	4	binary	ejbActivate: # of invocations
1596	63C	SM120JML	4	binary	ejbActivate: avg execution time
1600	640	SM120JMM	4	binary	ejbActivate: max execution time
1604	644	SM120JMN	4	binary	ejbPassivate: # of invocations
1608	648	SM120JMO	4	binary	ejbPassivate: avg execution time
1612	64C	SM120JMP	4	binary	ejbPassivate: max execution time
1616	650	SM120JMQ	8	binary	Average cpu time in microseconds.
1624	658	SM120JMR	8	binary	Minimum cpu time in microseconds.
1632	660	SM120JMS	8	binary	Maximum cpu time in microseconds.

Related reference

"Triplets" on page 145

"Header/self-defining section" on page 146

"Product section" on page 150

This section includes the header/self-defining and product sections.

"Subtype 1: Server activity record" on page 150

"Subtype 3: Server interval record" on page 153

"Subtype 6: Container interval record (Version 2)"

"Subtype 7: WebContainer activity record (Version 2)" on page 160

"Subtype 8: WebContainer interval record (Version 2)" on page 162

Related information

MVS System Management Facilities (SMF)

Subtype 6: Container interval record (Version 2):

The purpose of the J2EE container interval SMF record is to record activity within a J2EE container that is located inside the WebSphere Application Server transaction server.

This record is produced at regular intervals and is an aggregate of the activities running inside a J2EE container during the interval. This record can be used to perform application profiling, problem determination, and capacity planning.

A single record is created for each active J2EE container located in a WebSphere Application Server transaction server within the interval being recorded. If there is more than one server instance associated with a server, a record for the container will exist for each server instance. To get a common view of the work running in the J2EE container during the interval, you must merge the records after processing.

You can specify an interval through the WebSphere Application Server administrative console by setting **server_SMF_interval_length=n**, where n is the desired number of seconds.

You can activate this record by setting **server_SMF_container_interval_enabled=1 (or server_SMF_container_interval_enabled=true)** on the administrative console.

Container interval record (Version 2) schema

This section includes Subtype 6: Container interval record (Version 2).

J2EE container interval section

There is one section per record. The J2EE container interval section contains information about each activity that occurred within one J2EE container in the specified interval.

Offset (decimal)	Offset (hexadecimal)	Name	Length	Format	Description
0	0	SM120JI4	64	EBCDIC	The WebSphere for z/OS transaction server host name.
64	40	SM120JI5	8	EBCDIC	The WebSphere for z/OS transaction server name.
72	48	SM120JI6	8	EBCDIC	The WebSphere for z/OS transaction server instance name.
80	50	SM120JI7	512	Unicode	The WebSphere for z/OS container name. **Note:** This is hardcoded to ″Default″ for the 4.0.1 time frame.
592	250	SM120JI8	16	S390STCK	The time that the sample began in the server.
608	260	SM120JI9	16	S390STCK	The time that the sample ended.
624	270	SM120CL3	8	EBCDIC	Cell
632	278	SM120ND3	8	EBCDIC	Node

Subtype 6: Bean section:

See Subtype 5: Bean section

Subtype 6: Bean method section:

See Subtype 5: Bean method section

Related reference

"Triplets" on page 145

"Header/self-defining section" on page 146

"Product section" on page 150

This section includes the header/self-defining and product sections.

"Subtype 1: Server activity record" on page 150

"Subtype 3: Server interval record" on page 153

"Subtype 5: J2EE container activity record (Version 2)" on page 156

"Subtype 7: WebContainer activity record (Version 2)"

"Subtype 8: WebContainer interval record (Version 2)" on page 162

Related information

MVS System Management Facilities (SMF)

Subtype 7: WebContainer activity record (Version 2):

The purpose of the WebContainer activity SMF record is to record activity within a WebContainer running inside a WebSphere Application Server for z/OS transaction server.

The Web container is deployed within an EJB and runs within the EJB container. The WebContainer acts as a Web server handling HTTP sessions and servlets. The EJB container is not aware of the work the WebContainer does. Instead, the EJB container only records that the EJB has been dispatched. Meanwhile, the WebContainer gathers the detailed information, such as HTTP sessions, servlets, and their respective performance data. A single WebContainer Activity record is created for each activity that is run within a Web container.

WebContainer SMF recording is activated and deactivated along with the activation and deactivation of SMF recording for the J2EE container.

WebContainer activity record (Version 2) schema

This section includes Subtype 7: WebContainer activity record (Version 2).

WebContainer activity section

There is one section per record. The WebContainer activity section contains information about each activity that occurred within one web container.

Offset (decimal)	Offset (hexadecimal)	Name	Length	Format	Description
0	0	SM120WAA	64	EBCDIC	The WebSphere transaction server host name.
64	40	SM120WAB	8	EBCDIC	The WebSphere transaction server name.
72	48	SM120WAC	8	EBCDIC	The WebSphere transaction server instance name.
80	50	SM120WAD	8	HEX	The WLM enclave token.
88	58	SM120WAE	20	HEX	The identity of the activity.

108	6C	SM120WAF	16	S390STCK	The time the activity began in the server.
124	7C	SM120WAG	16	S390STCK	The time the activity ended.
140	8C	SM120CL4	8	EBCDIC	Cell
148	94	SM120ND4	8	EBCDIC	Node

HttpSessionManager section

There is one section per record. The HttpSessionManager section contains information about all (there may be zero or one) http sessions associated to one single activity.

Offset	Offset	Name	Length	Format	Description
0	0	SM120WAH	4	binary	″created Sessions″: Number of http sessions that were created.
4	4	SM120WAI	4	binary	″invalidatedSessions″: Number of http session that were invalidated.
8	8	SM120WAJ	4	binary	″activeSessions″: Number of http sessions that were referenced during this activity.
12	C	SM120WAK	4	binary	″sessionLifeTime″: lifetime of the session in milliseconds. If ″invalidatedSessions″ > 0, this is the average lifetime (in milliseconds) of the invalidated http session.

WebApplication section

There are multiple (0-n) sections per record. The WebApplication section contains information about all WebApplications involved in this activity.

Offset	Offset	Name	Length	Format	Description
0	0	SM120WAL	256	Unicode	The name of the WebApplication.
256	100	SM120WAM	4	binary	Number of servlet triplets in this web application section.
The following triplet appears 0-n times, once for each servlet section.					
260	104	SM120WAN	4	binary	Offset to servlet section from the beginning of this WebApplication section.
264	108	SM120WAO	4	binary	Length of servlet section.
268	10C	SM120WAP	4	binary	Number of servlet sections.

Servlet activity section

There are multiple (0-n) sections per WebApplication section. The Servlet activity section contains information about each servlet associated with WebApplications involved in this activity.

Offset	Offset	Name	Length	Format	Description
0	0	SM120WAQ	256	Unicode	The name of the servlet.

256	100	SM120WAR	4	binary	″responseTime″: Response time in milliseconds.
260	104	SM120WAS	4	binary	″numErrors″: The number of errors that were encountered during the servlet execution.
264	108	SM120WAT	4	binary	″loaded″: 0: The servlet did not have to be loaded as a result of this request. 1: The servlet had to be loaded as the result of this request.

268	10C	SM120WAU	16	EBCDIC	″loadedSince″: Timestamp from System.currentTimeMillis() when the servlet was loaded, in HEX format.
					
					Sample: The data as it appears in the record has the format
					`e7ef7c577c`
					
					, which needs to be converted to a Java long:
					`996155348860`
					
					. The Java long digits can be converted to java.util.Date:
					`Thu Jul 26 15:49:08 GMT+02:00 2001`

284	11C	SM120CPU	8	binary	Cpu time in microseconds.

Related reference

“Triplets” on page 145

“Header/self-defining section” on page 146

“Product section” on page 150

This section includes the header/self-defining and product sections.

“Subtype 1: Server activity record” on page 150

“Subtype 3: Server interval record” on page 153

“Subtype 5: J2EE container activity record (Version 2)” on page 156

“Subtype 6: Container interval record (Version 2)” on page 158

“Subtype 8: WebContainer interval record (Version 2)”

Related information

MVS System Management Facilities (SMF)

Subtype 8: WebContainer interval record (Version 2):

The purpose of the WebContainer interval SMF record is to record activity within a WebContainer running inside a WebSphere Application Server for z/OS transaction server.

The Web container execution environment consists of an EJB that is deployed into the EJB container. The WebContainer acts as a Web Server handling HttpSessions and Servlets. The EJB container is not aware of the purpose of the WebContainer activity record and only records that the EJB has been dispatched, but does not gather any of the detailed information, such as HttpSessions, Servlets, and their respective performance data. A single WebContainer record is created for each Web container.

In addition to data that is associated with an individual activity, there are some cases of Web container work that are performed outside the scope of an individual

request. For example, some instances of http session finalization and http session invalidation are performed asynchronously. In such a case a WebContainer interval record would record this data

WebContainer SMF recording is activated and deactivated along with the activation and deactivation of SMF recording for the J2EE container.

WebContainer interval record (Version 2) schema

WebContainer interval record (Version 2) schema.

WebContainer interval section

There is one section per record. The WebContainer interval section contains information about each activity that occurred within one WebContainer record.

Offset (decimal)	Offset (hexadecimal)	Name	Length	Format	Description
0	0	SM120WIA	64	EBCDIC	The WebSphere transaction server host name.
64	40	SM120WIB	8	EBCDIC	The WebSphere transaction server name.
72	48	SM120WIC	8	EBCDIC	The WebSphere transaction server instance name.
80	50	SM120WID	16	S390STCK	The time the sample began.
96	60	SM120WIE	16	S390STCK	The time the sample ended.
112	70	SM120CL5	8	EBCDIC	Cell
120	78	SM120ND5	8	EBCDIC	Node

HttpSessionManager section

There is one section per record. The HttpSessionManager section contains information about all (there may be zero or one) http sessions associated to one single activity.

Offset	Offset	Name	Length	Format	Description
0	0	SM120WIF	4	binary	"createdSessions": Number of http sessions that were created.
4	4	SM120WIG	4	binary	"invalidatedSessions": Number of http sessions that were invalidated.
8	8	SM120WIH	4	binary	"activeSessions": Current number of http sessions that are actively referenced in the server at the end of the interval.
12	C	SM120WII	4	binary	"minActiveSessions": Minimum number of active http sessions during the interval..
16	10	SM120WIJ	4	binary	"maxActiveSessions": Maximum number of active http sessions during the interval.
20	14	SM120WIK	4	binary	"sessionLifeTime": Average lifetime (in milliseconds) of invalidated http sessions.
24	18	SM120WIL	4	binary	"sessionInvalidateTime": Average time (in milliseconds) that was required to process the invalidation of http sessions.
28	1C	SM120WIM	4	binary	"finalizedSessions": Number of sessions that were finalized.

32	20	SM120WIN	4	binary	″liveSessions″: Total number of http sessions being tracked by the server at the end of the interval. This includes both active and inactive sessions.
36	24	SM120WIO	4	binary	″minLiveSessions″: Minimum number of live http sessions during the interval.
40	28	SM120WIP	4	binary	″maxLiveSessions″: Maximum number of live http sessions during the interval.

WebApplication interval section

There are multiple (0-n) sections per record. The WebApplication interval section contains information about all WebApplications involved in this activity.

Offset	Offset	Name	Length	Format	Description
0	0	SM120WIQ	256	Unicode	The WebApplication name.
256	100	SM120WIR	4	binary	″numLoadedServlets″: Number of servlets that were loaded. **Note:** This value might differ from the number of servlet sections in this record since servlets might exist that have been inactive during the interval.
260	104	SM120WIS	4	binary	Number of servlet triplets in this web application section.
The following triplet appears 0-n times, once for each servlet section.					
264	108	SM120WIT	4	binary	Offset to servlet section from the beginning of this WebApplication section.
268	10C	SM120WIU	4	binary	Length of the servlet section.
272	110	SM120WIV	4	binary	Number of servlet section.

Servlet section

There are multiple (0-n) sections per WebApplication section. The Servlet activity section contains information about all servlets involved per WebApplication in this activity.

Offset	Offset	Name	Length	Format	Description
0	0	SM120WIW	256	Unicode	The servlet name.
256	100	SM120WIX	4	binary	″totalRequests″: Number of times the servlet service was requested during the interval.
260	104	SM120WIY	4	binary	″responseTime″: Average response time in milliseconds.
264	108	SM120WIZ	4	binary	″minResponseTime″: Minimum response time in milliseconds.
268	10C	SM120WJ1	4	binary	″maxResponseTime″: Maximum response time in milliseconds.
272	110	SM120WJ2	4	binary	″numErrors″: The number of errors that were encountered during servlet execution.
276	114	SM120WJ3	16	EBCDIC	″loadedSince″: Timestamp when the servlet was loaded. Sample: Fri May 25 08:42:25 EDT 2001

292	124	SM120WJ4	8	binary	Average cpu time in microseconds.
300	12C	SM120WJ5	8	binary	Minimum cpu time in microseconds.
308	134	SM120WJ6	8	binary	Maximum cpu time in microseconds.

Related reference

"Triplets" on page 145

"Header/self-defining section" on page 146

"Product section" on page 150
This section includes the header/self-defining and product sections.

"Subtype 1: Server activity record" on page 150

"Subtype 3: Server interval record" on page 153

"Subtype 5: J2EE container activity record (Version 2)" on page 156

"Subtype 6: Container interval record (Version 2)" on page 158

"Subtype 7: WebContainer activity record (Version 2)" on page 160

Related information

MVS System Management Facilities (SMF)

Overview of SMF record type 80

As WebSphere Application Server becomes more capable of authentication and setting or changing the identity on a thread, so arises the need for the ability to audit these changes. Along with this also comes the need to audit the accompanying authorization requests made through EJBRoles checking, intending to produce audit records that include the original authenticated identity. This auditing in WebSphere Application Server is managed not through WebSphere Application Server itself, but through its External Security Manager (RACF or equivalent), where the SMF records are cut.

Preparing for audit support:

In order to take advantage of auditing in WebSphere Application Server, you need to set up SMF and RACF and have both running.

1. Set up SMF for audit support. For information on setting up and starting SMF, see *z/OS MVS System Management Facilities (SMF), SA22-7630*
2. Enable auditing for the EJB Roles by setting the RACF AUDIT attribute. This will set up RACF for auditing in WebSphere Application Server. You can turn on auditing for the ADMIN and PAYROLL classes with the following command:

   ```
   RALTER EJBROLE (ADMIN,PAYROLL) AUDIT(ALL)
   ```

   - Alternately, you could modify the RACFROLE job to put the AUDIT information there.
   - For more information and additional parameters for the AUDIT attribute, see the *z/OS Security Server RACF Auditor's Guide*.

Related information

MVS System Management Facilities (SMF)

z/OS Security Server RACF Auditor's Guide

Using audit support:

This file gives an overview of how to use audit support.

Auditing is performed using SMF records issued by RACF or an equivalent External Security Manager. This means that SMF audit records are cut as part of the WebSphere Application Server use of SAF interfaces such as IRRSIA00 (to manage ACEEs) and the RACROUTE macro.

The table below lists the various security authentication mechanisms and the corresponding data that is written to each part of the ACEE X500NAME field (this data is also in the RACO and SMF records). The information under ″Service Name″ is the constant string that is included in the ″Issuer's Distinguished Name″ field of X500NAME. The information under ″Authenticated Identity″ is the principal that is recorded in the ″Subject's Distinguished Name″ field.

Table 1. Security authentication mechanisms and the corresponding data that is written to each part of the ACEE X500NAME field

Authentication mechanism	Service name	Authenticated identity
Custom Registry	WebSphere Custom Registry	Custom registry principal name
Kerberos	WebSphere Kerberos	Kerberos principal, in the ″DCE″ format used for extracting the corresponding MVS userid using IRRSIM00 (/.../realm/principal)
RunAs Rolename	WebSphere Role Name	Role name
RunAs Server	WebSphere Server Credential	MVS userid
Trust Interceptor	WebSphere Authorized Login	MVS userid
RunAs Userid/Password	WebSphere Userid/Password	MVS Userid

In addition to tracking by MVS userid, events need to be traced to an originating userid. This is especially true for originating userids that are not MVS-based, such as EJB Roles, Kerberos principals, and Custom Registry principals.

Related information

MVS System Management Facilities (SMF)

z/OS Security Server RACF Auditor's Guide

Collecting performance diagnosis information

The following report options are listed here for information. IBM Service may request that you run one or more of these reports while assisting you with diagnosis. You do not need to collect this data unless it is requested by IBM Service.

- If you suspect that you are having throughput problems in a particular address space, for example by looking at some other real-time performance data, IBM Service may need to see a dump of one or more address spaces. This is done using the following parameters:

```
JOBNAME=(<jobname list>)
SDATA=(LSQA,PSA,SQA,SUM,SWA,TRT,WLM,CSA,RGN)
```

- If you suspect that the problem could be resulting from GRS latch or ENQ contention, check the RMF Enqueue Activity Report and enter the console command:

```
D GRS,CONTENTION
```

during a time period in which the performance problem is observed. SYS.BPX.A000.FSLIT.FILESYS.LSN represents HFS latches. Latch sets with a numeric suffix are file latches, specifically SYS.BPX.A000.FSLIT.FILESYS.LSN.01. If you detect file latch contention, the best way to determine the exact HFS file causing the problem is with an SVC dump, also collected during a time period in which contention occurred. You will need to dump one of the OMVS data spaces to get the file information.

```
DUMP COMM=(description of problem)
Reply to dump WTO, where serverproc is the name of your WebSphere Server
                   address space(s)
JOBNAME=(OMVS,Serverproc),DSPNAME=('OMVS'.SYSZBPX1,'OMVS'.SYSZBPX2),
SDATA=(CSA,GRSQ,LPA,NUC,PSA,RGN,SQA,TRT,SUM)
```

- Sometimes USS errors can cause performance problems. The USS Ctrace (SYSOMVS) MIN tracing option always records OMVS errors. You can take an SVC dump of the OMVS address space (as described in the previous bullet) and the data spaces and format the SYSOMVS CTRACE. Use IPCS options 7.2.1, suboption D, component SYSOMVS and the TALLY option (default is FULL). Look for trace events of errors in the TALLY report.
- To find delays in applications, collect application performance information
  - SMF 120 records.
  - Jinsight profile

Configuring WebSphere Application Server for z/OS variables

WebSphere Application Server for z/OS provides configuration variables that control server behavior. Specifically, these variables allow you to control:

- Output destinations and characteristics for the error log, and for CTRACE buffers, data sets and the external writer.
- Trace buffers, data sets, and the content of trace data.
- Types of dumps to be requested.
- Timeout values for system and application behavior.

Generally speaking, the default values are designed for normal operation in a production environment. Other circumstances might require different values:

- When you first customize and verify WebSphere Application Server for z/OS installation, or
- When you test application workloads in a test environment, or
- when you encounter a problem, and need to collect more diagnostic data.

Steps for configuring WebSphere variables

You should know that:

- Configuration variables may be set on a cell, node, or server level.
  - Variable values set on a cell level apply to all servers in all nodes in the cell, unless a different value for the same variable is set on a node or server level. Variable settings on a node or server level override values for the same variable set at the cell level.
  - Variables set on a node level apply to all servers in the node, unless a different value for the same variable is set on the server level. Variable settings on a server level override values for the same variable set at the node or cell level.
  - Variables set on a server level apply only to the specific server, not to any other servers in the same node or cell.

  When you are diagnosing particular problems, you are most likely to alter variable values on a server level, for a particular server. Specifying variable values on the server level affects both the controller and servant regions.

- You may use scripting interfaces, instead of the administrative console, to alter configuration variable values.

Depending on the types of problems you encounter, you might need to change the values set for configuration variables that control WebSphere Application Server behavior. The following procedure explains how to use the administrative console to change configuration variable values, commonly known as console settings.

1. Click **Environment -> Manage WebSphere Variables** in the console navigation tree.
2. On the **WebSphere Variables** page, select **Server** as the scope of the variable setting, and click **Apply**.
3. On the **WebSphere Variables** page, click **New**.
4. On the **Variable** page, specify a name and value for the variable. So other people can understand what the variable is used for, also specify a description for the variable. Then click **OK**.
5. Verify that the variable is shown in the list of variables.
6. Save your configuration.
7. To have the configuration take effect, stop the server and then start the server again.

Using RMF

RMF can usually be started with the simple 'S RMF' command from the MVS console.

Here is a typical RMF procedure:

```
//RMF      PROC
//IEFPROC  EXEC PGM=ERBMFMFC,REGION=0M,PARM='MEMBER(XS)'
//IEFPARM  DD   DDNAME=IEFRDER
//IEFRDER  DD   DSN=SYS1.PARMLIB,DISP=SHR
```

The following is a copy of the PARMLIB member ERBRMFXS: (parmeters beginning with a /* are not used in this example but may be useful to you.

```
CPU                     /* COLLECT CPU STATISTICS        */
     CHAN                    /* COLLECT CHANNEL STATISTICS    */
     CYCLE(1000)             /* SAMPLE AT 1 TIME / SECOND     */
     DEVICE(NOCHRDR)         /* NO CHARACTER RDR DEV STATS    */
     DEVICE(COMM)            /* ADDED COMM FOR 37X5           */
     DEVICE(DASD)            /* COLLECT DIRECT ACCESS DEVICE  */
     /* STATISTICS                 */
     DEVICE(NOGRAPH)         /* NO GRAPHICS DEVICE STATISTICS */
     DEVICE(NOTAPE)          /* NO TAPE DEVICE STATISTICS     */
     DEVICE(NOUNITR)         /* NO UNIT RECORD DEVICE STATS   */
     ENQ(SUMMARY)            /* ENQ REPORTING                 */
     INTERVAL(15M)           /* REPORT AT  15 MIN INTERVALS   */
     IOQ(DASD)               /* I/O Q'ING FOR DEV IN LOG CU   */
     IOQ(COMM)               /* I/O Q'ING FOR DEV IN LOG CU   */
     NOVSTOR                 /* NO RMF 3.2 AND LATER REL */
     OPTIONS                 /* OPERATOR MAY CHG RMF OPTIONS  */
     PAGING                  /* COLLECT PAGING STATISTICS     */
     PAGESP                  /* COLLECT PAGE/SWAP DATASET STAT*/
     RECORD                  /* RECORD INTO SMF DATASET       */
     NOSTOP                  /* STOP AFTER  90 MINUTES        */
     SYNC(SMF)               /* INTERVAL SYNCED WITH SMF      */

     SYSOUT(H)               /* SYSOUT CLASS OF OUTPUT REPORT */
```

```
 WKLD(PERIOD,SYSTEM)        /* COLLECT WORKLOAD MANAGER
                               STATISTICS AND REPORT AT THE
                               PERIOD LEVEL + TOTAL LINE     */
TRACE(CCVUTILP)
```

The Monitor III data gatherer can be started after RMF with the 'F RMF,S III' modify command.

To use the RMF Monitor 3 display, go to the "Sysplex Summary" display as follows:
1. Type RMF on ISPF command line.
2. Type 3 to see Monitor III choices.
3. Type S to get Sysplex reports.
4. Type 1 to see the Sysplex Summary report.

You can scroll back and forth in time with PF10 and PF11, or over-type the time field.

Look at the transactions in the WebSphere Application Server service and reporting class and note the Average Response time, Transactions per second, and Performance Index. You may also explore further in RMF Monitor III to see the "System Information" report.

Setting up component trace (CTRACE)

WebSphere Application Server for z/OS uses z/OS component trace (CTRACE) facilities to manage the collection and storage of trace data. Unless you configure specific CTRACE controls, WebSphere Application Server for z/OS records its trace data in address-space buffers in private (pageable) storage. This data is not accessible unless a dump of the address space is taken.

Although CTRACE data is primarily output for IBM service personnel to use, exploiting CTRACE capabilities at your installation allows you to have additional trace data available when a problem first occurs. Because CTRACE efficiently uses system resources, you can collect valuable trace data with minimal impact on performance.

When your installation first customizes and verifies WebSphere Application Server for z/OS installation, you have the option of defining CTRACE controls and resources. Using the ISPF customization dialog to configure a base application server node, you can specify:

- Data sets to contain CTRACE data collected for WebSphere Application Server for z/OS.
- CTRACE writer parameters that control the writer through which trace data moves from address-space buffers into trace data sets.
- The parmlib member that connects WebSphere Application Server for z/OS address spaces to trace data sets, and optionally turns on the CTRACE writer.
- WebSphere variables that control the characteristics of trace data.

The ISPF customization dialog generates instructions for:

1. Starting the CTRACE writer
2. Starting the WebSphere Application Server for z/OS application server

Following the instructions in sequence is quite important; you can lose valuable trace data if you do not start the CTRACE writer before starting the server.

For information about the advantages of using CTRACE facilities, see *z/OS MVS Diagnosis: Tools and Service Aids, GA22-7589*

Related reference

"Setting output destinations and characteristics" on page 212

"Setting trace controls" on page 213

"Viewing CTRACE and logged data through IPCS" on page 180

Related information

z/OS MVS Diagnosis: Tools and Service Aids

Steps for preparing CTRACE controls and resources

Before you start CTRACE activity for WebSphere Application Server for z/OS servers, you need to make some decisions about CTRACE controls and resources. You have the option of using default CTRACE values and resources, such as the IBM-supplied CTRACE parmlib member for WebSphere Application Server for z/OS, or you may alter default values and provide your own resources.

Perform the following steps to prepare CTRACE controls and resources:

1. Decide where you want to store CTRACE data. The location of trace data is set through the WebSphere variable `ras_trace_outputLocation`. Make a note of the value that you want to use; you will set the value later, when you start CTRACE activity.

   Note: Tips:

   - See Setting Trace Controls for an explanation of and default values for the `ras_trace_outputLocation` variable.
   - If you decide to use trace data sets, you can use existing or create new data sets now or later, as part of the WebSphere Application Server for z/OS customization process.
   - If you want the CTRACE data for each cell to go into separate data sets, use the `ras_trace_ctraceParms` variable described in Setting Trace Controls.
   - When you are installing WebSphere Application Server for z/OS, sending trace data to SYSPRINT can be helpful; however, tracing to SYSPRINT in a production environment can cause spool space to fill up quickly. For production, you can specify a different trace output location through the WebSphere variable `ras_trace_outputLocation`.
   - To separate trace data from other standard output messages, such as `System.output.println` from Java applications, you can direct trace data to a separate data set. To accomplish this separation, you need to:
     - Specify a TRCFILE DD statement in the JCL start procedure (proc) for the server.
     - Set the WebSphere variable `ras_trace_outputLocation` to TRCFILE. Note that you may specify the TRCFILE value with or without additional variable values.

2. Through modify commands, you have the ability to dynamically and temporarily direct trace data to SYSPRINT or TRCFILE.
   a. You can direct CTRACE data to buffers as part of normal operations.

b. To capture tracing related to an error, use the modify command to temporarily send trace data to SYSPRINT or TRCFILE.

c. Then you can use the modify command again, to dynamically reset the trace-output location

You can complete this sequence of actions without stopping the server that is encountering the problem.

3. Decide whether you want to accept defaults or provide other values for the following WebSphere variables. Make a note of the values that you want to use; you will set the values later, when you start CTRACE activity.
   - `ras_time_local`
   - `ras_trace_BufferCount`
   - `ras_trace_BufferSize`

   For defaults and valid values, see Setting Trace Controls
4. Decide whether you want to use the default JCL start procedure for the CTRACE writer, or code your own JCL proc. WebSphere Application Server for z/OS provides a CTRACE writer proc named `bbowtr`. If you decide to provide your own CTRACE writer procedure, create the JCL start proc using the rules for source JCL for an external writer. Place the start procedure in your system proclib.
5. Decide whether you want to use the default CTRACE parmlib member for WebSphere Application Server for z/OS, or provide your own. The WebSphere parmlib member is named CTIBBO00. If you decide to provide your own parmlib member, create one according to the following rules, and place it in your system parmlib. **Rules:**
   - You must follow the same naming convention; that is, the name must consist of the letters `CTIBBO`, but you may replace the two zeroes with any two alphanumeric characters.
   - Your parmlib member must contain the following parameters:

```
      TRACEOPTS       WTRSTART(xxxxxx)
  ON     /*CONNECT TO CTRACE EXTERNAL WRITER: */       WTR(xxxxxx)
```

Table 2. Parameters for the CTIBBOxx parmlib member

Parameter	Required or Optional?	Description
`TRACEOPTS`	Required and positional	This parameter must be specified first.
`WTRSTART` *(xxxxxx)*	Optional	This parameter automatically starts the CTRACE writer, using the specified JCL procedure. **Alternative:** You may use the operator command `TRACE CT,WTRSTART=xxxxx` to start the CTRACE writer. If you use this parameter in the parmlib member, and later issue the operator command, CTRACE issues an informational message to notify you that the writer already has been started.
`ON`	Required	This parameter must be specified before component options.

Table 2. Parameters for the CTIBBOxx parmlib member (continued)

WTR *(xxxxxx)*	Required	This parameter identifies the JCL procedure to be used to start the CTRACE external writer. The specified value must match the value of the WTRSTART parameter to capture the WebSphere Application Server for z/OS trace data into a trace data set.

6. Ensure that the DLL named BBORTSS5 is in the link-pack area (LPA).

After you have made these decisions and completed preparations, you are ready to start CTRACE activities using one of the following procedures:

- During customization of WebSphere Application Server for z/OS (see "Steps for starting CTRACE as part of WebSphere Application Server for z/OS customization") or
- While WebSphere Application Server for z/OS servers already are running on z/OS (see "Steps for starting CTRACE while WebSphere Application Server for z/OS servers are active" on page 173

Related reference

"Steps for starting CTRACE as part of WebSphere Application Server for z/OS customization"

"Steps for starting CTRACE while WebSphere Application Server for z/OS servers are active" on page 173

"Setting trace controls" on page 213

Steps for starting CTRACE as part of WebSphere Application Server for z/OS customization

Make sure you have properly prepared CTRACE controls and resources as described in "Steps for preparing CTRACE controls and resources" on page 170

Perform the following steps to start CTRACE as part of the customization process for WebSphere Application Server for z/OS:

1. Using the ISPF customization dialog to configure a base application server node, specify: Trace data set characteristics, CTRACE writer parameters, the CTRACE parmlib member and WebSphere variables, if you want values other than the defaults. The ISPF customization dialog generates instructions for starting the CTRACE writer and instructions for starting the WebSphere Application Server for z/OS application server
2. Start the CTRACE writer, using the generated instructions. You must start the writer first, or you might lose valuable trace data.
3. Start the WebSphere Application Server for z/OS application server, using the generated instructions.
4. When you need to collect trace data for analysis:
   a. Use the following operator command to disconnect WebSphere Application Server for z/OS from CTRACE: TRACE CT,ON,COMP= *cell_short_name* REPLY x,WTR=DISCONNECT,END. where *cell_short_name* is the value specified through the ISPF customization dialog to identify the location of server configuration files. The name must be 8 or fewer characters and all uppercase.

b. Use the following operator command to stop the CTRACE writer address space: `TRACE CT,WTRSTOP=writer_name` where *writer_name* is the name of the JCL start procedure for the CTRACE writer.

Related reference

"Viewing CTRACE and logged data through IPCS" on page 180

Steps for starting CTRACE while WebSphere Application Server for z/OS servers are active

Make sure you have properly prepared CTRACE controls and resources as described in "Steps for preparing CTRACE controls and resources" on page 170

If you start a WebSphere Application Server for z/OS server before starting the CTRACE writer for WebSphere, the server still gathers data in its trace buffers. This trace data is not available for use unless you follow this procedure, or until a dump of the server address space is taken.

Perform the following steps to start CTRACE when a WebSphere Application Server for z/OS server already is active:

1.
   - Use the following operator command to start the CTRACE writer address space: `TRACE CT,WTRSTART=writer_name` where *writer_name* is the name of the JCL start procedure for the CTRACE writer that is specified in the WebSphere Application Server for z/OS CTIBBO *xx* parmlib member.
   - To connect WebSphere Application Server for z/OS to a CTRACE writer other than the one specified in the CTIBBO *xx* parmlib member, also enter these operator commands: `TRACE CT,ON,COMP=cell_short_name REPLY x,WTR=writer_name, END` where *cell_short_name* is the value specified through the ISPF customization dialog to identify the location of server configuration files (the name must be 8 or fewer characters and all uppercase) and *writer_name* is the name of a JCL start procedure for a CTRACE external writer. The JCL start procedure must reside in the system's proclib.

   The CTRACE external writer begins writing the server's trace data to the location specified through the WebSphere variable *ras_trace_outputLocation*
2. When you need to collect trace data for analysis:
   a. Use the following operator command to disconnect WebSphere Application Server for z/OS from CTRACE: `TRACE CT,ON,COMP=cell_short_name REPLY x,WTR=DISCONNECT,END`
   b. Use the following operator command to stop the CTRACE writer address space: `TRACE CT,WTRSTOP=writer_name` where *writer_name* is the name of the JCL start procedure for the CTRACE writer.

Related reference

"Viewing CTRACE and logged data through IPCS" on page 180

Using CTRACE to collect trace data for Java server applications

Applications that run in WebSphere Application Server for z/OS can use JRas to provide tracing support that is consistent with WebSphere tracing. Instrumented applications use the JRas interfaces and classes for logging and tracing; trace data is written to the same component trace data set as the internal traces issued by the

WebSphere Application Server for z/OS runtime. So you can gather application trace data in the same locations, and use the same commands to start and stop CTRACE for these JRas applications as you do for WebSphere Application Server for z/OS server in which the applications are running.

Related reference

"Viewing CTRACE and logged data through IPCS" on page 180

"Programming with the JRas framework" on page 105

Setting up the error log

WebSphere Application Server for z/OS uses an error log to record error information when an unexpected condition or failure is detected within the product's own code. Such unexpected conditions or failures include:

- Assertion failures
- Unrecoverable error conditions
- Failures related to vital resources, such as memory
- Operating system exceptions
- Programming defects in WebSphere Application Server for z/OS code.

Because WebSphere Application Server for z/OS is predefined as a z/OS system logger application, you can use a log stream as the product's error log. Doing so offers the following flexibility:

- You can direct error information to:
  - A coupling facility log stream, which provides sysplex-wide error logging, or
  - A DASD-only log stream, which provides single system-only error logging.
- You can set up a common log stream for all WebSphere Application Server for z/OS servers, or individual log streams for each application server. Local z/OS client ORBs can also log data in log streams. The system logger APIs are unauthorized, but log stream resources can be protected using security products such as RACF.
- You can use the WebSphere variable

  `ras_time_local`

  to control whether timestamps in the error log appear in local time (`ras_time_local=1`) or Greenwich Mean Time (GMT) (`ras_time_local=1`), which is the default.

When your installation first customizes and verifies WebSphere Application Server for z/OS installation, you have the option of defining the error log as a log stream. Using the ISPF customization dialog to configure a base application server node, you can specify log stream characteristics, including sizes. After verifying installation, you can change the log stream used for normal operations.

For additional information about z/OS log stream requirements, access the *z/OS MVS Setting up a Sysplex, SA22-7625* available on the z/OS Library Web page

Related concepts

"Setting output destinations and characteristics" on page 212

Displaying WebSphere Application Server work

This topic lists parameters that allow you to dynamically change values related to tracing activity for a server or servant.

You can use either the WebSphere Application Server administrative console or the z/OS MVS console to accomplish many operator tasks that are related to WebSphere Application Server for z/OS servers. Entering the z/OS display or modify commands through the MVS console can provide information or perform tasks that are useful for diagnosing problems. With these commands you can perform the following actions:

- Use parameters to control WebSphere Application Server for z/OS operations
- Display information about WebSphere Application Server for z/OS servers or servants
- Dynamically change values related to tracing activity for a server or servant.

With the modify command, you can display information about the following:

- Active controllers (servers)
- Servants
- Sessions
- Trace settings
- Java trace
- JVM heap statistics
- Work Elements
- Error Log

Displaying the options on the modify command

You can display the options on the modify command with the help parameter. For example, the command

```
f azsr01a,help
```

generates the following output:

```
BBOO0178I THE COMMAND MODIFY MAY BE FOLLOWED BY ONE OF THE FOLLOWING KEYWORDS:
BBOO0179I CANCEL - CANCEL THIS CONTROL REGION
BBOO0179I TRACEALL - SET OVERALL TRACE LEVEL
BBOO0179I TRACEBASIC - SET BASIC TRACE COMPONENTS
BBOO0179I TRACEDETAIL - SET DETAILED TRACE COMPONENTS
BBOO0179I TRACESPECIFIC - SET SPECIFIC TRACE POINTS
BBOO0179I TRACEINIT - RESET TO INITIAL TRACE SETTINGS
BBOO0179I TRACENONE - TURN OFF ALL TRACING
BBOO0179I TRACETOSYSPRINT - SEND TRACE OUTPUT TO SYSPRINT (YES/NO)
BBOO0179I DISPLAY - DISPLAY STATUS
BBOO0179I TRACE_EXCLUDE_SPECIFIC - EXCLUDE SPECIFIC TRACE POINTS
BBOO0179I TRACEJAVA - SET JAVA TRACE OPTIONS
BBOO0179I TRACETOTRCFILE - SEND TRACE OUTPUT TO TRCFILE DD CARD (YES/NO)
BBOO0179I MDBSTATS - MDB DETAILED STATISTICS
```

You can display the DISPLAY options with the display,help parameter. For example, the command

```
f azsr01a,display,help
```

generates the following output:

```
BBOO0178I THE COMMAND DISPLAY, MAY BE FOLLOWED BY ONE OF THE FOLLOWING KEYWORDS:
BBOO0179I SERVERS - DISPLAY ACTIVE CONTROL PROCESSES
BBOO0179I SERVANTS - DISPLAY SERVANT PROCESSES OWNED BY THIS CONTROL PROCESS
BBOO0179I SESSIONS - DISPLAY INFORMATION ABOUT COMMUNICATIONS SESSIONS
BBOO0179I TRACE - DISPLAY INFORMATION ABOUT TRACE SETTINGS
BBOO0179I JVMHEAP - DISPLAY JVM HEAP STATISTICS
```

```
BBOO0179I WORK - DISPLAY WORK ELEMENTS
BBOO0179I ERRLOG - DISPLAY THE LAST 10 ENTRIES IN THE ERROR LOG
BBOO0188I END OF OUTPUT FOR COMMAND DISPLAY,HELP
```

Displaying work elements

The z/OS modify**server**,display,work command can provide useful information for diagnosing problems, or displaying what is going on inside your application servers. The syntax is:

```
f servername,display,work,display work parameters
```

where the **display work parameters** can be one of the following:

HELP
displays the DISPLAY,WORK parameters

CLINFO
Display classification information

See Classifying z/OS workload for more information.

EJB
EJB requests driven by IIOP: total, current, dispatched and timed out

EJB,SRS
EJB requests driven by IIOP by servant

SERVLET
Servlet requests driven by HTTP: total, current, dispatched and timed out

SERVLET,SRS
Servlet requests by each servant

MDB
Message-driven bean (MDB) requests driven by JMS: total, current, dispatched and timed out

MDB,SRS
MDB requests broken down by servant

ALL
Combines the above for EJBs, servlets, and MDBs

ALL,SRS
EJBs, servants, and MDBs by servant

SUMMARY
total requests, current in progress, and in dispatch for all types

SUMMARY,SRS
total received by each servant, current in dispatch in each servant for all types

The following examples demonstrate how to use various display,work parameters and the resulting output:

```
f azsr01a,display,work,servlet

BBOO0255I TIME OF LAST WORK DISPLAY Wed Dec 3 19:17:54 2003
BBOO0256I TOTAL SERVLET REQUESTS         150670    (DELTA 1654)
BBOO0257I CURRENT SERVLET REQUESTS       1

BBOO0258I SERVLET REQUESTS IN DISPATCH   0
BBOO0267I TOTAL SERVLET TIMEOUTS         0    (DELTA 0)
BBOO0188I END OF OUTPUT FOR COMMAND DISPLAY,WORK,SERVLET
```

```
f azsr01a,display,work,servlet,srs

BBOO0255I TIME OF LAST WORK DISPLAY Wed Dec 3 19:18:01 2003
BBOO0259I STC18964:  TOTAL SERVLET REQUESTS  152344   (DELTA 1675)
BBOO0260I STC18964:  SERVLET REQUESTS IN DISPATCH    0
BBOO0188I END OF OUTPUT FOR COMMAND DISPLAY,WORK,SERVLET,ALL

 (EJB and MDB displays would look the same except for the request type.)

f azsr01a,display,work,summary

BBOO0255I TIME OF LAST WORK DISPLAY Wed Dec  3 19:18:38 2003
BBOO0261I TOTAL REQUESTS TO SERVER     173591    (DELTA 13944)
BBOO0262I TOTAL CURRENT REQUESTS       0
BBOO0263I TOTAL REQUESTS IN DISPATCH   0
BBOO0268I TOTAL TIMED OUT REQUESTS     0    (DELTA 0)
BBOO0188I END OF OUTPUT FOR COMMAND DISPLAY,WORK,SUMMARY

f azsr01a,display,work,summary,srs

BBOO0255I TIME OF LAST WORK DISPLAY Wed Dec  3 19:27:01 2003
BBOO0264I STC18964:  TOTAL REQUESTS       173591    (DELTA 0)
BBOO0265I STC18964:  TOTAL REQUESTS IN DISPATCH  0
BBOO0188I END OF OUTPUT FOR COMMAND DISPLAY,WORK,SUMMARY,SRS
```

To display a combination of all information with one command, use the all keyword, as follows:

```
f azsr01a,display,work,all

BBOO0255I TIME OF LAST WORK DISPLAY Wed Dec  3 19:27:07 2003
BBOO0256I TOTAL EJB REQUESTS         0    (DELTA 0)
BBOO0257I CURRENT EJB REQUESTS       0
BBOO0258I EJB REQUESTS IN DISPATCH   0
BBOO0267I TOTAL EJB TIMEOUTS         0    (DELTA 0)
BBOO0256I TOTAL SERVLET REQUESTS         173591    (DELTA 0)




BBOO0257I CURRENT SERVLET REQUESTS       0
BBOO0258I SERVLET REQUESTS IN DISPATCH   0
BBOO0267I TOTAL SERVLET TIMEOUTS         0    (DELTA 0)
BBOO0256I TOTAL MDB REQUESTS         0    (DELTA 0)
BBOO0257I CURRENT MDB REQUESTS       0
BBOO0258I MDB REQUESTS IN DISPATCH   0
BBOO0267I TOTAL MDB TIMEOUTS         0    (DELTA 0)
BBOO0188I END OF OUTPUT FOR COMMAND DISPLAY,WORK,ALL

f azsr01a,display,work,all,srs

BBOO0255I TIME OF LAST WORK DISPLAY Wed Dec  3 19:30:06 2003
BBOO0259I STC18964:  TOTAL EJB REQUESTS          0    (DELTA 0)
BBOO0260I STC18964:  EJB REQUESTS IN DISPATCH    0
BBOO0259I STC18964:  TOTAL SERVLET REQUESTS      173591  (DELTA 0)
BBOO0260I STC18964:  SERVLET REQUESTS IN DISPATCH    0
BBOO0259I STC18964:  TOTAL MDB REQUESTS          0    (DELTA 0)
BBOO0260I STC18964:  MDB REQUESTS IN DISPATCH    0
BBOO0188I END OF OUTPUT FOR COMMAND DISPLAY,WORK,ALL,SRS
```

Note: Some points to consider:

- Adding,HELP at the end displays your choices (for example, DISPLAY,WORK,HELP).
- DISPLAY,WORK produces the same output as DISPLAY,WORK,SUMMARY.

- The Display EJB command displays IIOP requests. So, if you drive an HTTP request that runs a JSP or servlet that calls an EJB, only the servlet count increments. The counts are by protocol used to get to the controller.
- When displaying the work by servant, only the total and in dispatch requests display. Without the SRS keyword, the current requests and time outs are also displayed.
- Everything is not frozen while the numbers are collected. The old data is saved away and used to calculate the 'delta' in the next command. (Under a heavy load, the displayed values and deltas might not add up as you would expect.)
- TOTAL requests display the delta since last DISPLAY,WORK command, but it does not include the current "in flight" requests.

Displaying the error log

You can display the contents of the product error log with the DISPLAY,ERRLOG option of the modify command. This shows the last ten messages in the error log even if you are not routing them to a log stream. For example, the following command:

```
f x5sr01b,display,errlog
```

generates the following output:

```
BBOO0266I (STC18876) BossLog: { 0001} 2003/11/25 20:08:55.120 01
SYSTEM=SYSB SERVER=X5SR01B  PID=0X010201B2 TID=0X12FB3F00 00000000
c=UNK ./bborjtr.cpp+812 ... BBOO0222I TRAS0017I: The startup trace
state is *=all=disabled.
BBOO0266I (STC18876) BossLog: { 0002} 2003/11/25 20:09:08.255 01
SYSTEM=SYSB SERVER=X5SR01B  PID=0X010201B2 TID=0X12FB3F00 00000000
c=UNK ./bborjtr.cpp+812 ... BBOO0222I SECJ0231I: The Security
component's FFDC Diagnostic Module com.ibm.ws.security.core.SecurityDM
registered successfully: true.
BBOO0266I (STC18876) BossLog: { 0003} 2003/11/25 20:09:09.562 01
SYSTEM=SYSB SERVER=X5SR01B  PID=0X010201B2 TID=0X12FB3F00 00000000
c=UNK ./bborjtr.cpp+812 ... BBOO0222I SECJ0212I: WCCM JAAS
configuration information successfully pushed to login provider class.
BBOO0266I (STC18876) BossLog: { 0004} 2003/11/25 20:09:09.573 01
SYSTEM=SYSB SERVER=X5SR01B  PID=0X010201B2 TID=0X12FB3F00 00000000
c=UNK ./bborjtr.cpp+812 ... BBOO0222I SECJ0240I: Security service
initialization completed successfully
BBOO0266I (STC18876) BossLog: { 0005} 2003/11/25 20:09:18.304 01
SYSTEM=SYSB SERVER=X5SR01B  PID=0X010201B2 TID=0X12FB3F00 00000000

c=UNK ./bborjtr.cpp+812 ... BBOO0223I PMI0023W: Unable to register
PMI module due to duplicate name: SoapConnectorThreadPool
BBOO0266I (STC18876) BossLog: { 0006} 2003/11/25 20:09:29.451 01
SYSTEM=SYSB SERVER=X5SR01B  PID=0X010201B2 TID=0X12FB3F00 00000000
c=UNK ./bborjtr.cpp+812 ... BBOO0222I SECJ0243I: Security service
started successfully
BBOO0266I (STC18876) BossLog: { 0007} 2003/11/25 20:09:29.464 01
SYSTEM=SYSB SERVER=X5SR01B  PID=0X010201B2 TID=0X12FB3F00 00000000
c=UNK ./bborjtr.cpp+812 ... BBOO0222I SECJ0210I: Security enabled false
BBOO0266I (STC18876) BossLog: { 0008} 2003/11/25 20:09:35.772 01
SYSTEM=SYSB SERVER=X5SR01B  PID=0X010201B2 TID=0X12FB3F00 00000000
c=UNK ./bborjtr.cpp+812 ... BBOO0223I PMI0023W: Unable to register PMI
module due to duplicate name: ProcessDiscovery
. . . .
BBOO0188I END OF OUTPUT FOR COMMAND DISPLAY,ERRLOG
```

z/OS modify command parameters and their equivalent WebSphere variables

The z/OS modify command provides parameters that allow you to dynamically change values related to tracing activity for a server or servant. Table 2 lists the modify command parameters and the WebSphere variable that provides equivalent function.

Table 3. z/OS modify command parameters and their equivalent WebSphere variables

z/OS modify command parameter	Equivalent WebSphere variable	For more information, see..
TRACEALL	ras_trace_defaultTracingLevel	Internal tracing tips for WebSphere for z/OS
TRACEBASIC	ras_trace_basic **CAUTION:** **Do not change this variable unless directed by IBM service personnel.**	Setting trace controls for IBM service
TRACEDETAIL	ras_trace_detail **CAUTION:** **Do not change this variable unless directed by IBM service personnel.**	Setting trace controls for IBM service
TRACESPECIFIC	ras_trace_specific **CAUTION:** **Do not change this variable unless directed by IBM service personnel.**	Setting trace controls for IBM service
TRACE_EXCLUDE_SPECIFIC	ras_trace_exclude_specific **CAUTION:** **Do not change this variable unless directed by IBM service personnel.**	Setting trace controls for IBM service
TRACEINIT	n/a	Example: Getting help for the modify command
TRACENONE	n/a	Example: Getting help for the modify command
TRACETOSYSPRINT	ras_trace_outputLocation=SYSPRINT	Internal tracing tips for WebSphere for z/OS
TRACETOTRCFILE	ras_trace_outputLocation=TRCFILE	Internal tracing tips for WebSphere for z/OS
TRACEJAVA	n/a	Example: Getting help for the modify command

Related tasks

Classifying z/OS workload
Use this task to classify inbound requests by using a workload classification document, a common XML file that classifies inbound HTTP, IIOP, and message-driven bean (MDB) work for the z/OS workload manager.

Related information

MVS System Commands

Viewing diagnostic information

The following topics provide information about specific sources of diagnostic data, and the tools or resources you might need to view or work with that data.

Type of diagnostic tools or data:	Notes and instructions for use appear in:
CEEDUMPs	"Viewing CEEDUMPs in the job log"
SVC dumps	
CTRACE and JRas data	Viewing CTRACE and JRas data through IPCS
Error log data	Viewing error log contents through the Log Browse Utility (BBORBLOG)
z/OS display command	Using the z/OS display command
Java minor codes	Converting Java minor codes
SYSPRINT	Viewing SYSPRINT output, STDOUT and STDERR streams in an HFS File
Message routing	Message routing

Viewing CEEDUMPs in the job log

An error caught by LE or the Java run-time can result in the production of a CEEDUMP, which is written to a separate CEEDUMP specification in the job log. To view the dump contents, select the CEEDUMP portion of the output for the address space. The 'Traceback' section at the beginning of the dump can be very helpful.

Viewing CTRACE and logged data through IPCS

Once activated, the WebSphere Application Server for z/OS always writes trace data into memory buffers. The number and size of these buffers is controlled using WebSphere variables. You can get this trace data from a dump, which may be taken by the system or requested by the operator through DUMP or SLIP commands.

To view messages or application trace data from Component Trace, you must use the interactive problem control system (IPCS) to format the data. The source of the trace data can be a dump data set or a trace data set, and the command to use would be

```
IPCS CTRACE
```

. You can also use the IPCS CTRACE command to merge multiple trace entities together such as multiple WebSphere Application Server for z/OS address space traces, OMVS, and TCPIP.

Steps for using the IPCS dialog to format CTRACE data

When setting up IPCS, your installation may customize IPCS for its users. IBM recommends providing access to the IPCS dialog through an ISPF panel. If your installation has not customized IPCS as recommended, you need to start the IPCS dialog. See z/OS MVS IPCS User's Guide, SA22-7596 to find out how to start the IPCS dialog.

Perform the following steps to use the IPCS dialog to format application trace data:

1. From the IPCS Primary Option Menu panel, select option 6 (*COMMAND*).
2. On the IPCS Subcommand Entry panel:
   a. Issue the *SETDEF* subcommand to determine the default values for routing displays.
   b. Enter the *CTRACE* command, with the following required parameters: CTRACE COMP(*cell_short_name*)

      where *cell_short_name* is the value specified through the ISPF customization dialog to identify the location of server configuration files. The name must be 8 or fewer characters and all uppercase.

      Note: If you were interested in only JRAS data, you would enter the following:

      ```
      CTRACE COMP(cell_short_name
      )USEREXIT(JRAS)
      ```

      Specify additional parameters as necessary.

      Example: To direct trace data to the terminal only, you would append the NOPRINT and TERMINAL parameters to the CTRACE command.

   Tip: For a complete list of CTRACE command parameters, see z/OS MVS IPCS Commands, SA22-7594.
3. View your application's data, basing the method you choose on which one is appropriate for the location of the data:

If you directed output to the...	Then use the...
IPCS print data set (IPCSPRNT)	ISPF/PDF Browse option
Terminal	Dump Display Reporter panel

   Tips: To navigate through the trace data on the Dump Display Reporter panel, use the commands and PF keys listed in z/OS MVS IPCS User's Guide, SA22-7596.

   CTRACE enables you to view multiple traces together with the trace data from the various sources intermixed based on the time stamp. See z/OS MVS IPCS Commands, SA22-7594, for specifics on using this MERGE subcommand.

Finding the subname for IPCS CTRACE

If the trace data set is an SVC dump, the trace subname must also be specified. This subname is the aggregation of the address space's jobname with its ASID (address space identifier), in printable hexadecimal. An easy way to determine the subname is to query CTRACE for the data using the following IPCS subcommand:

```
CTRACE QUERY DSN('dump.data.set')
```

Once you get the subname you can view the WebSphere Application Server for z/OS trace data with the following IPCS subcommand:

```
CTRACE COMP(cell_short_name) SUB((subname)) FULL DSN('dump.data.set')
```

where *cell_short_name* is the value specified through the ISPF customization dialog to identify the location of server configuration files. The name must be 8 or fewer characters and all uppercase.

Note: The *subname* parameter is optional for only the trace data set. It is required when viewing the trace data using the dump data set.

Steps for using IPCS in batch mode to format CTRACE data

You must create an IPCS dump directory before you can use IPCS in batch mode. When setting up IPCS, your installation may customize IPCS for its users. This customization can include modifying the IBM-supplied BLSCDDIR CLIST with default values for creating an IPCS dump directory.

To view messages or application trace data from Component Trace, you must use the interactive problem control system (IPCS) to format the data. Using IPCS in batch mode is the easiest method of formatting data, especially if you do not have much experience with using IPCS, TSO/E and ISPF. Through batch mode, you can use IPCS to format trace data and write it to an MVS data set. Optionally, you may copy the contents of that data set into an HFS file for viewing.

When your installation has modified the BLSCDDIR CLIST the steps outlined herein will create an IPCS dump directory.

1. Decide on a fully-qualified data set name for the directory.
2. From the TSO/E command prompt, enter the BLSCDDIR command, specifying the data set name.

   For example, to create a dump directory named IBMUSER.DDIR, enter:

```
%blscddir dsn('ibmuser.ddir')
```

If your installation has not customized IPCS, you might need to alter other BLSCDDIR CLIST parameters. See the z/OS MVS IPCS User's Guide, SA22-7596 and z/OS MVS IPCS Commands, SA22-7594 for more details about using the BLSCDDIR CLIST to create a dump directory.

Perform the following steps to use IPCS in batch mode to format application trace data:

1. Create a file and copy the following sample JCL into it. This JCL invokes IPCS to extract and format JRAS trace data and write it into an MVS data set, and then uses the TSO/E OPUT command to copy the formatted data from the MVS data set into an HFS file.

```
//IBMUSERX   JOB ,
// CLASS=J,NOTIFY=&SYSUID,MSGCLASS=H
//IPCS       EXEC PGM=IKJEFT01,REGION=4096K,DYNAMNBR=50
//IPCSDDIR   DD DSN=IBMUSER.DDIR,DISP=SHR
//IPCSDOC    DD SYSOUT=H
//JRASTRC    DD DSN=IBMUSER.CB390.CTRACE,DISP=SHR
//IPCSPRNT   DD DSN=IBMUSER.IPCS.OUT,DISP=OLD
//SYSTSPRT   DD SYSOUT=*
//SYSTSIN    DD *
IPCS
DROPDUMP DDNAME(JRASTRC)
PROFILE LINESIZE(80)PAGESIZE(99999999)
SETDEF NOCONFIRM
CTRACE COMP(SYSBBOSS) DDNAME(JRASTRC) FULL PRINT +
       NOTERMINAL
DROPDUMP DDNAME(JRASTRC)
END
/*
//OPUT       EXEC PGM=IKJEFT01,REGION=4096K,DYNAMNBR=50
//SYSTSPRT   DD SYSOUT=*
//SYSTSIN    DD *
oput 'ibmuser.ipcs.out' '/u/ibmuser/ipcs/jrastrace.txt' TEXT
/*
```

2. Edit the sample JCL to replace *IBMUSER.DDIR* with the data set name that you used for the IPCS dump directory you created.
   a. Use the *PAGESIZE* parameter on the *PROFILE* statement only if you do not want to print the output data set.
   b. You may replace the HFS file name with the name of an existing HFS file, but you do not have to do so. The OPUT command processing will create a new HFS file, if the one specified does not exist, and grants read and write access to that file for your user ID only.

      If you do specify an existing HFS file, the OPUT command processing will write over any data that is already in that file. If you want to know more about the OPUT command, see the z/OS UNIX System Services Command Reference, SA22-7802.
   c. Change the data set name specified on the *JRASTRC DD* in the example to the name of the data set containing the CTRACE data.
   d. Change the name of the MVS data set on both the *JRASTRC DD* statement and the *OPUT* command in the SYSTSIN stream, as necessary. The formatted output of the JRAS CTRACE data is first written to the MVS data set specified by the IPCSPRNT DD statement and then (optionally) copied to the HFS data set. You must either pre-allocate this data set, or change the sample JCL to allocate the data set. This data set should have a record format of VBA and a record length of 133.
3. Submit the JCL to start the IPCS batch job.

Once you are done you can use a UNIX editor, such as vi, to view your trace data in the HFS file. If you want to know more about the UNIX editors, see z/OS UNIX System Services User's Guide, SA22-7801.

CTRACE enables you to view multiple traces together with the trace data from the various sources intermixed based on the time stamp. See z/OS MVS IPCS Commands, SA22-7594, for specifics on using this MERGE subcommand.

Related tasks

"Steps for using the IPCS dialog to format CTRACE data" on page 180

Sample JCL to display WebSphere for z/OS trace data: The following sample shows JCL that displays WebSphere for z/OS trace data.

Note: The JCL uses an IPCS dump directory (in VSAM data set `userid.DUMP.DIR`) that must be allocated before you run the JCL. See z/OS MVS IPCS Commands, SA22-7594 , for information about initializing a dump directory.

```
//SHOWTRC  JOB <job card info>
//JOBLIB   DD DISP=SHR,DSN=BBO.SBBOMIG
//       DD DISP=SHR,DSN=SYS1.MIGLIB
//PRINTIT EXEC PGM=IKJEFT01,REGION=OM
//IPCSDDIR DD DISP=(OLD,KEEP),DSN=userid.DUMP.DIR
//IPCSPARM DD DISP=SHR,DSN=SYS1.PARMLIB
//SYSTSPRT DD SYSOUT=*
//IPCSTOC  DD SYSOUT=*
//IPCSPRNT DD SYSOUT=*
//*---------------------------
//SYSTSIN DD *
IPCS NOPARM
  CTRACE COMP(SYSBBOSS) SUB((subname)) FULL DSN('dump.data.set')
/*
```

The following example shows JCL that displays WebSphere for z/OS trace data for multiple address spaces.

```
//SHOWTRC2 JOB <job card info>


//JOBLIB   DD DISP=SHR,DSN=BBO.SBBOMIG
//       DD DISP=SHR,DSN=SYS1.MIGLIB
//PRINTIT EXEC PGM=IKJEFT01,REGION=0M
//IPCSDDIR DD DISP=(OLD,KEEP),DSN=userid.DUMP.DIR
//IPCSPARM DD DISP=SHR,DSN=SYS1.PARMLIB
//SYSTSPRT DD SYSOUT=*
//IPCSTOC  DD SYSOUT=*
//IPCSPRNT DD SYSOUT=*
//*---------------------------
//SYSTSIN DD *
IPCS NOPARM
  MERGE
  CTRACE COMP(SYSBBOSS) SUB((subname)) FULL DSN('dump.data.set')
  CTRACE COMP(SYSBBOSS) SUB((subname2)) FULL DSN('dump.data.set')
  MERGEEND
/*
```

Viewing error log contents through the Log Browse Utility (BBORBLOG)

You can use the Log Browse Utility (BBORBLOG) to view the error log stream. If you need to look at the WebSphere Application Server for z/OS error log stream, use ISPF option 6 to enter the command:

1. Use ISPF option 6 to enter the proper command.

   ```
   ex 'BBO.SBBOEXEC(BBORBLOG)' 'BBO.BOSSXXXX'
   ```

   where the log-stream name is `BBO.BOSSXXXX`
2. The space allocation and the unit for the allocation are contained within the REXX code. If you keep a large amount of trace data, the allocation must be made larger.
3. The WebSphere Application Server for z/OS provides an ISPF REXX EXEC named BBORBLOG, that allows you to browse the error log stream.
4. Save the output.

   When you use the BBORBLOG browser, it creates a data set with your user ID followed by the log stream name. You should rename it if you wish to save your browser output. The contents of the current view of the log stream will remain until the stream reaches its retention date. The next time you invoke the browser, however, the current view of the log stream will be deleted (because it uses the same data set name). The previous data will exist in another record (not the current view) until its retention date.

Use the following information to determine viewing of the error log:

- By default, the macro formats the error records to fit a 3270 display.
- Timestamps are in Greenwich Mean Time (GMT) unless changed by setting the WebSphere Application Server variable `ras_time_local` to 1.
- Message BBOJ0051I, which appears in the job output, can help correlate error-log entries to the proper job output.

Using the log browse utility (BBORBLOG)

The browser takes two parameters:

Parameter	Description
log stream name	The name of the log stream. See the job messages for the name of the log stream.

format option	**80** The default. The log stream record will be formatted on a lrecl length of 80 characters. Additional lines will be wrapped. **NOFORMAT** Turns off formatting. The error log message appears as one log message string in the browse file.

View the error log stream output using the BBORBLOG browser. To invoke the browser, go to ISPF option 6 and enter:

```
'BBO.SBBOEXEC(BBORBLOG)' 'BBO.BOSSXXXX format option'
```

Note: In this example, BBORBLOG resides in BBO.SBBOEXEC.

The browser creates a browse data set named "userid.stream_name ", which contains the contents of the log stream. When the browser is executed, it:

1. Allocates a data set called userid.stream_name, which overwrites any duplicate data sets.
2. Populates the data set with the contents of the log stream.
3. Puts the user in "browse" mode on the data set.

Important: Each time BBORBLOG is invoked a static file is created which overwrites the existing file. In order to refresh the file, it is necessary to re-issue BBORBLOG

There are three valid ways (three separate commands to use) to invoke the browser. We will illustrate each of these using the following example:

Example: If the BBORBLOG member was in a data set named BBO.SBBOEXEC, then you would issue one of the following depending on your chosen format option:

```
ex 'BBO.SBBOEXEC(BBORBLOG)' 'BBO.BOSSXXXX'
ex 'BBO.SBBOEXEC(BBORBLOG)' 'BBO.BOSSXXXX  80'
ex 'BBO.SBBOEXEC(BBORBLOG)' 'BBO.BOSSXXXX  NOFORMAT'
```

Tip:: It is easier to invoke the browser if you add the target library (in our example, BBO.SBBOEXEC) to the SYSEXEC concatenation of the user logon procedure during the WebSphere Application Server for z/OS installation. If you concatenate the library during installation, you do not have to specify the library containing the browser REXX exec- you only need to specify BBORBLOG.

Error log stream record output

There are two error log stream records that we will look at:

- Server logstream
- CERR of a server.

Note: The numbers to the left of each sample were added to specify lines-they will not be in the actual output.

Sample output from a server logstream:

```
1  2000/06/01 16:01:06.683 01 SYSTEM=SY1 SERVER=BBOASR1A JobName=BBOASR1S
2            ASID=0X0033 PID=0X0100003C TID=0X24F858A0 0X000004 c=2.1010030
3            ./bbooreq.cpp+4437 ... BBOU0013W The function
4            make_user_exception(IIOP_protocolArea*)+4437 raised a user exception
5            CosNaming::NamingContext::NotFound.
```

The log stream record output fields from stream BBO.BOSSXXXX are:

Table 4. Parts table for a server logstream record output

Component	Description
line 1: 2000/06/01 16:01:06.683 01	Date / timestamp / 2-digit record version number
line 1: SYSTEM=SY1	System name
line 1: SERVER=BBOASR1A	Server name
line 1: JobName=BBOASR1S	Jobname
line 2: ASID=0X0033	ASID (address space identifier)
line 2: PID=0X0100003C	PID (Process ID)
line 2: TID=0X24F858A0 0X000004	TID (Thread ID)
line 2: c=2.1010030	Request correlation information
line 3: . /bbooreq.cpp+4437	File name & line
line 3: BBOU0013W	Log message number
line 3: The function. ..	Log message
lines 4-5: make_user_exception... CosNaming::Naming...	Continuation lines: Continuation of the Log Stream log message

Note: Each field is delimited by a blank.

Sample output from CERR of a server:

```
1|  BossLog: { 0017} 2000/06/01 15:58:25.557 01 SYSTEM=SY1 SERVER=BBOASR1A
2|     PID=0X0100003C TID=0X24F82920 00000000  c=3.C5D02
3|     ./bboiroot.cpp+1195 ... BBOU0012W The function IRootHomeImpl::findHome(
4|     const char*)+1195 received CORBA system exception CORBA::INTERNAL.
5|     Error code is C9C21200.
```

The CERR job message output fields are:

Table 5. Parts table for a CERR record output

Component	Description
line 1: BossLog: { 0017}	BossLog: {entry number}
line 1: 2000/06/01 15:58:25.557 01	Date / timestamp / 2-digit record version number
line 1: SYSTEM=SY1	System name
line 1: SERVER=BBOASR1A	Server name
line 2: PID=0X0100003C	PID (Process ID)
line 2: TID=0X24F82920 00000000	TID (Thread ID)
line 2: c=3.C5D02	Request correlation information
line 3: . /bboiroot.cpp+1195	File name & line
line 3: BBOU0012W	Log message number
line 3: The function IRootHomeImpl::find. ..	Log message
lines 4-5: const char*)+1195 received CORBA system exception CORBA::INTERNAL. Error code is C9C21200.	Continuation lines: Continuation lines of the CERR job message

- Each field is delimited by a blank.
- The CERR format is found in SYSOUT, not the logger.

Saving your BBORBLOG browser output

When you use the BBORBLOG browser, it creates a data set with your user ID followed by the log stream name. You should rename it if you wish to save your browser output. The contents of the current view of the log stream will remain until the stream reaches its retention date. The next time you invoke the browser, however, the current view of the log stream will be deleted (because it uses the same data set name). The previous data will exist in another record (not the current view) until its retention date.

Using the z/OS display command

You may use either the WebSphere Application Server administrative console or the z/OS MVS console to accomplish many operations tasks related to WebSphere Application Server for z/OS servers. Entering the z/OS display or modify commands through the MVS console can provide information or perform tasks that are useful for diagnosing problems.

The purpose of the DISPLAY command is to display information about the operating system, the jobs and application programs that are running, the processor, devices that are online and offline, central and expanded storage, workload management service policy and mode status, and the time of day.

For example, you could use the command

```
D WLM,APPLENV=*
```

to verify that WLM is defined and available for your applications.

To display the dynamic WLM application environment, use the command:

```
D WLM,DYNAPPL=*
```

You can check the WLM environment for a specific application using the following command:

```
D WLM,APPLENV=appname
```

These commands will return information similar to the following:

```
RESPONSE=SC42
IWM029I 13.09.47 WLM DISPLAY 075
APPLICATION ENVIRONMENT NAME STATE STATE DATA
appname AVAILABLE
ATTRIBUTES:PROC=procname SUBSYSTEM TYPE:CB
```

For more information on the display command see z/OS MVS System Commands, SA22-7627

Related information

z/OS MVS System Commands, SA22-7627

Converting Java minor codes

Occasionally, Java will take an WebSphere Application Server for z/OS error code (C9C2xxxx in hexadecimal) and convert it to a very large negative number. If you get a very large negative number, try converting it back to hexadecimal to find the correct code.

To convert the error codes back to hexadecimal:

- Add 2^{32} to the negative number and convert it into hexadecimal. This can be done using the OMVS command

  ```
  bc
  ```

  Example: Suppose you get the error code ″910022649″:
  1. Under OMVS, type the command:

     ```
     bc
     ```
  2. then type:

     ```
     obase=16
     2^32 - 910022649
     quit
     ```
- The bc program displays C9C22807, which is the hex value that you should look up.

Viewing SYSPRINT output, STDOUT and STDERR streams in an HFS File

If you are familiar with UNIX or NT environments, you may be reluctant to use the facilities of SDSF (or IOF) to view the SYSPRINT output from your Application Server regions. In the past, you may have used a familiar editor (such as vi) in a TELNET session to view the STDOUT and STDERR streams. These STDOUT and STDERR streams have been redirected to SYSPRINT, and it is possible to redirect them to files in the HFS for viewing using an HFS editor.

The JCL below for an Application Server region uses this facility. A new SET statement has been added to point to the LOGPATH directory, and the SYSPRINT DD statement has been changed to point to a file in the LOGPATH/servername directory, in this case named: `was.log.d &LYYMMDD..t&LHHMMSS` The extra period (.) between the date and time variables is not a typographical error, but rather an idiosyncrasy of JCL syntax that is necessary to terminate the first variable. &LYYMMDD will be replaced with the local date in YYMMDD format and &LHHMMSS will be replaced by the local time in HHMMSS format. Using the local date and time ensures a unique file for each instance of the Application Server region that is started. The PATHMODE subparameter sets the file mode to 775 and the PATHOPTS subparameter OWRONLY opens the file for WRITE access and the sub-parameter OCREAT indicates that if the file does not already exist, create it.

```
//TSTST1S PROC IWMSSNM='TSTST1S',PARMS='-ORBsrvname '
//  SET CBCONFIG='/WebSphere390/TSDCONF'
//  SET LOGPATH='/WebSphere390/logs'        <<----- Added to set path
//  SET RELPATH='controlinfo/envfile'
//WSDAS1S EXEC PGM=BBOSR,REGION=0M,TIME=NOLIMIT,
// PARM='/ &PARMS &IWMSSNM'
//BBOENV DD PATH='&CBCONFIG/&RELPATH/&SYSPLEX/&IWMSSNM/current.env'
//CEEDUMP  DD SYSOUT=*
//SYSOUT   DD SYSOUT=*
//*YSPRINT DD SYSOUT=*  Added the following 3 lines for SYSPRINT

//SYSPRINT DD PATHMODE=(SIRWXU,SIRWXG,SIROTH),          <<< 775
//            PATHOPTS=(OWRONLY,OCREAT),                <<<
//            PATH='&LOGPATH/TSTST1S/was.log.d&LYYMMDD..t&LHHMMSS'
```

In the previous example, if the file specified by the SYSPRINT DD statement is created on May 2, 2003 at 1:30:35 PM, the PATH parameter will resolve to: `/WebSphere390/logs/TSTST1S/was.log.d030503.t133035` which will be a unique file for this execution of this server instance.

Message routing

The following is a summary of how messages get routed in WebSphere Application Server:

- WTO messages issued by WebSphere Application Server during initialization are sent to hardcopy, but most can be routed to the data set specified by ras_default_msg_dd (see Setting Output destinations and characteristics).
- The Java "Audit" messages are also sent to hardcopy, but can be routed to the data set ras_hardcopy_msg_dd. (see Setting Output destinations and characteristics).
- Trace error, service, and fatal messages are sent to the error log specified by the ras_log_logstreamName. Otherwise, they go to CERR (SYSOUT). Some may also go to hardcopy. You must set the ras_log_logstreamName environment variable to the error log stream name. To set ras_log_logstreamName, on the administrative console select Environment -> Manage WebSphere Variables -> New.
- Early error messages will go to STDERR (SYSOUT) until WebSphere Application Server connects to the log stream. A WTO (BBOO0153I) is issued telling you how many messages went to STDERR before you connected to the log stream.
- Trace messages (such as BUFFER, SYSPRINT, and TRCFILE DD) are also routed to ras_trace_outputLocation.
- System.out.println, Tr.Sysout, stdout (cout), and client output go to SYSPRINT.(see Viewing SYSPRINT output, STDOUT and STDERR streams in an HFS File
- System.out.printerr, Tr.Syserr, and stderr (cerr) go to SYSOUT.

To use these new message routing variables, you must do two things:

1. Add these parameters to the server definitions using the Administrative Console under Environment -> Manage WebSphere Variables:
   - **ras_default_msg_dd =DEFALTDD**
   - **ras_hardcopy_msg_dd =HRDCPYDD**

   You can set these variables for individual control and servant processes, but it is easier to set them in the Environment variables for the entire cell. For the Daemon, you must prefix them with "DAEMON_" and set them at the cell level:
   - **DAEMON_ras_default_msg_dd =DEFALTDD**
   - **DAEMON_ras_hardcopy_msg_dd =HRDCPYDD**

Update the included &Z members in PROCLIB to add these new DD statements

```
//* Output DDs
//CEEDUMP DD SYSOUT=*,SPIN=UNALLOC,FREE=CLOSE
//SYSOUT DD SYSOUT=*,SPIN=UNALLOC,FREE=CLOSE
//SYSPRINT DD SYSOUT=*,SPIN=UNALLOC,FREE=CLOSE
//DEFALTDD DD SYSOUT=*,SPIN=UNALLOC,FREE=CLOSE
//HRDCPYDD DD SYSOUT=*,SPIN=UNALLOC,FREE=CLOSE
```

Note:

- If you specify the new environment variables, but do not specify the DD cards in the procedure, you will not get an error message indicating that the DD cards are missing, and the tracing output will not be written anywhere.

- If you try to direct the output for multiple streams to the same DD, such as setting both **ras_default_msg_dd** and **ras_hardcopy_msg_dd** to DEFALTDD (or to SYSPRINT,) then the allocation will fail and output will be sent to the default location (JOBLOG/SYSLOG).

For example, these DD files are used to segregate the messages and keep almost all of them off the hardcopy console

1. (SYSLOG.) JESMSGLG - a few start-up and shut-down messages
2. JESYSMSG - MVS allocation and deallocation messages
3. SYSOUT - a few start-up and shut-down messages
4. SYSPRINT - a few start-up and shut-down messages
5. HRDCPYDD - audit messages that would normally go to SYSLOG
6. DEFALTDD - informational messages that would normally go to SYSLOG

Using the Error Dump and Cleanup interface

The Error Dump and Cleanup (BBORLEXT) interface exists to call WebSphere Application Server for z/OS in a recovery environment to allow it to request a dump and to clean up its resources.

The interface will:
- Save the function and DLL names of the failing z/OS component into the SDWA.
- Determine whether or not to issue an SDUMP, if relevant to the time-of-failure environment.
- Clean up z/OS internal structures and connections.

Program requirements: This interface **must** be called from within a WebSphere Application Server for z/OS location service daemon, controller (region), or servant (region). There are no restrictions against in which recovery environment, such as an ESTAE or FRR routine, the caller must reside.

General information

Interface:	BALR to BBORLEXT
Address of routine:	(ECVT+'234'x)+'20'x
Address mode:	AMODE 31, RMODE any
State:	Allow problem program state, task mode
Cross memory mode:	PASN=HASN=SASN (non-cross memory)
Return codes:	No return codes
Function:	Clean-up various WebSphere for z/OS resources and possibly issue an SVC dump for the current address space

Input register information

The contents of the registers are as follows:

1	Contains the address of the SDWA
14	Contains the return address
15	Contains the entry point address of BBORLEXT

Output register information

When control returns to the caller, the contents of the registers are as follows:

0-1	Used as a work register by the system
2-14	Unchanged
15	Used as a work register by the system

Note: Some callers depend on register contents remaining the same before and after issuing a service. If the system changes the contents of registers on which the caller depends, the caller must save them before issuing the service and restore them after the system returns control.

Note: A dump will not occur for X22 abends or for certain reason codes from 0D6, 052, 067, CC3, and DC3 abends. There may also be other error conditions that will not create a dump.

Example:

```
Example Here is an example of how to call this routine in assembler:
LA 1,SDWA        Load SDWA@ in Reg 1
L 15,(0,16)      Load CVT address
L 15,140(,15)    Load ECVT address
L 15,564(,15)    Load address of z/OS structure
L 15,32(,15)     Load address of z/OS routine
BALR 14,15       Invoke z/OS routine
```

Using the showlog command to view Common Base Events

The showlog command converts the service log from binary format into plain text. Use the following showlog commands to produce output in Common Base Event XML format:

- On a Windows platform: showlog -format=CBE-XML-1.0.1 *filename*
- On a UNIX platform: showlog -format=CBE-XML-1.0.1 *filename*
- On a z/OS platform: showlog -start *startDateTime* -format=CBE-XML-1.0.1 *logStreamName*

where:

filename
Is the service log file name.

startDateTime
Specifies the start date and time, in yyyy-MM-ddTHH:mm:ss.SSSZ format. Milliseconds and time zone are optional.

logStreamName
Is the name of the configured error log stream.

For examples of showlog scripts, see Showlog Script.

Showlog Script

You can view the service log by using the showlog script to convert the contents of the service log to a text format that you can then write to a file or dump to the command shell window.

To run the showlog script:

1. Open a shell window on the machine where the service log resides.
2. Change the directory to *install_directory*/bin where *install_directory* is the fully qualified path where the WebSphere Application Server product is installed.
3. Run the showlog script:

   On z/OS systems, use the following format:

   ```
   showlog.sh {-start startDateTime [-end endDateTime] | -interval interval}
   [-format CBE-XML-1.0.1] [-encoding encoding] logStreamName
   [outputFilename
   ```

   where:

   -start Specifies the start date and time, in yyyy-MM-ddTHH:mm:ss.SSSZ format. Milliseconds and time zone are optional.

   -end Specifies the end date and time, in yyyy-MM-ddTHH:mm:ss.SSSZ format. Milliseconds and time zone are optional.

   -interval
   Specifies the start date as the system date and time minus interval milliseconds, and end date as the system date and time. Valid values are integers greater than 0.

   -format
   Specifies the output format. Currently only CBE-XML-1.0.1 format is supported (this complies with the Common Base Event specification version 1.0.1). If no format is given, showlog outputs in a tabular format.

   -encoding
   Specifies the output file encoding, a character encoding supported by the local Java Virtual Machine .

   logStreamName
   Is a log file name.

   outputFilename
   Is optional. If no file name is given, the showlog script creates a default showlog.out filename, outputFilename is created in the current directory unless it is a fully qualified file name.

The formatted contents of the service log are always written to a file. There are parameters to showlog.sh which control content and encoding of the output. Enter `showlog.sh` without parameters for parameter usage information.

The showlog script can return informational messages containing service names, return codes, and reason codes. For more information about using the z/OS log stream, or to look up service names, return codes, and reason codes, refer to z/OS MVS Authorized Assembler Services Reference ENF-IXG(SA22-7610). Return and reason codes are listed for each service.

Refer to the topic "Authorization for System Logger Application Programs" in *z/OS MVS Assembler Services Guide (SA22-7605)* for advice on permitting access to the log stream.

Examples of showlog scripts on z/OS systems

Here are examples of showlog scripts on z/OS systems

- To write all records from the file WAS.ERROR.LOG since July 14, 2004 in log analyzer format into myoutput.log, use the following format:

  ```
  showlog.sh -start 2004-07-14T00:00:00 WAS.ERROR.LOG myoutput.log
  ```

- To write all records from WAS.ERROR.LOG since July 14, 2004 in CBE XML 1.0.1 format into myoutput.log use the following format:

  ```
  showlog.sh -start 2004-07-14T00:00:00 -format CBE-XML-1.0.1
  WAS.ERROR.LOG myoutput.log
  ```

- To write all records from WAS.ERROR.LOG between July 14, 2004 and April 9, 2005 in CBE XML 1.0.1 format into myoutput.log, use the following format:

  ```
  showlog.sh -start 2004-07-14T00:00:00 -end 2005-04-09T00:00:00
  -format CBE-XML-1.0.1 WAS.ERROR.LOG myoutput.log
  ```

- To write all records from WAS.ERROR.LOG since December 6, 2004 at 9pm eastern standard time into showlog.out (the default output file), use the following format:

  ```
  showlog.sh -start 2004-12-06T21:00:00EST WAS.ERROR.LOG
  ```

Displaying user instructions

Run the showlog script with no parameters to display usage instructions.

On z/OS systems, the script is named

```
showlog.sh
```

Related tasks

"Generating messages in CBE format"

"Modifying logstream size" on page 194

Generating messages in CBE format

The z/OS logs can be stored in Common Base Event format. This enables the showlog tool to read the data in the logstream. In turn, the showlog output can be read by tools such as the log analyzer and log and trace analyzer (included as part of the Application Server Toolkit).

To enable writing of the logstream in Common Base Event format, open the administrative console, and complete the following steps.

1. Click **Application servers** → **server1** → **Process Definition** → **Control** → **Java Virtual Machine** → **Custom Properties**
2. Add a new custom property with `name="com.ibm.ws.logging.zOS.errorLog.format" and value "CBE-XML-1.0.1"`
3. Restart your application server for this setting to take effect.

When this property is set to CBE-XML-1.0.1, the messages written to the error logstream are in binary Common Base Event format. You can then use the showlog script to view the binary Common Base Event records in the logstream.

Note:

If you enable writing of the logstream in Common Base Event format, the error log is no longer viewable with the log browse utility. This action changes the format used to write to the logstream so that only the showlog tool can read it.

Related tasks

"Modifying logstream size"

"Viewing error log contents through the Log Browse Utility (BBORBLOG)" on page 184

Related reference

"Showlog Script" on page 192

"Using the showlog command to view Common Base Events" on page 191

Modifying logstream size

If the logstream record size is too small for a message being written to it, you see a message similar to the following written to your job log:

```
TRAS0024I: Log entry is of size 5012 bytes which is too large to be added to
log stream which is configured for 4096 byte records.  Log entry will not be
logged to the log stream.
```

The original message is also written to the job log and can be viewed there.

To resolve this issue and ensure your messages fit into your logstream, change the MAXBUFSIZE of the error log logstream.

The following code shows an example where the sample BBOERRLG job generated by the ISPF customization dialog is modified to set the MAXBUFSIZE to 8192:

```
//BBOERRLG  JOB (ACCTNO,ROOM),'USER10',CLASS=A,REGION=0M
//*
//*
//*
//BBORCLGS EXEC PGM=IXCMIAPU
//SYSPRINT DD  SYSOUT=*
//SYSIN    DD  *
    DATA TYPE(LOGR)
    DEFINE LOGSTREAM NAME(WAS.TY5.ERROR.LOG)
           DASDONLY(YES)
           HLQ(LOGGER)
           LS_SIZE(500)
           STG_SIZE(500)
           MAXBUFSIZE(8192)
           AUTODELETE(YES)
           RETPD(1)
           LS_DATACLAS(STANDARD)
```

Related tasks

"Generating messages in CBE format" on page 193

Related reference

"Showlog Script" on page 192

"Using the showlog command to view Common Base Events" on page 191

Detecting hung threads in J2EE applications

A common error in J2EE applications is a hung thread. A hung thread can result from a simple software defect (such as an infinite loop) or a more complex cause (for example, a resource deadlock). System resources, such as CPU time, might be consumed by this hung transaction when threads run unbounded code paths, such as when the code is running in an infinite loop. Alternately, a system can become unresponsive even though all resources are idle, as in a deadlock scenario. Unless an end user or a monitoring tool reports the problem, the system may remain in this degraded state indefinitely.

The hang detection option for WebSphere Application Server is turned on by default. You can configure a hang detection policy to accommodate your applications and environment so that potential hangs can be reported, providing earlier detection of failing servers. When a hung thread is detected, WebSphere Application Server notifies you so that you can troubleshoot the problem.

Using the hang detection policy, you can specify a time that is too long for a unit of work to complete. The thread monitor checks all managed threads in the system (for example, Web container threads and object request broker (ORB) threads) . Unmanaged threads, which are threads created by applications, are not monitored.

When WebSphere Application Server detects that a thread has been active longer than the time defined by the thread monitor threshold, the application server takes the following actions:

- Logs a warning in the WebSphere Application Server log that indicates the name of the thread that is hung and how long it has already been active. The following message is written to the log:

  ```
  WSVR0605W: Thread threadname has been active for
  hangtime and may be hung.  There are totalthreads
  threads in total in the server that may be hung.
  ```

  where: *threadname* is the name that appears in a JVM thread dump, *hangtime* gives an approximation of how long the thread has been active and *totalthreads* gives an overall assessment of the system threads.
- Issues a Java Management Extensions (JMX) notification. This notification enables third-party tools to catch the event and take appropriate action, such as triggering a JVM thread dump of the server, or issuing an electronic page or e-mail. The following JMX notification events are defined in the com.ibm.websphere.management.NotificationConstants class:
  - TYPE_THREAD_MONITOR_THREAD_HUNG This event is triggered by the detection of a (potentially) hung thread.
  - TYPE_THREAD_MONITOR_THREAD_CLEAR This event is triggered if a thread that was previously reported as hung completes its work. See "Detecting hung threads in J2EE applications."
- Triggers changes in the performance monitoring infrastructure (PMI) data counters. These PMI data counters are used by various tools, such as the Tivoli Performance Viewer, to provide a performance analysis.

False Alarms

If the work actually completes, a second set of messages, notifications and PMI events is produced to identify the false alarm. The following message is written to the log:

```
WSVR0606W: Thread threadname was previously reported to be
hung but has completed. It was active for approximately hangtime.
There are totalthreads threads in total in the server that still
may be hung.
```

where *threadname* is the name that appears in a JVM thread dump, *hangtime* gives an approximation of how long the thread has been active and *totalthreads* gives an overall assessment of the system threads.

Automatic adjustment of the hang time threshold

If the thread monitor determines that too many false alarms are issued (determined by the number of pairs of hang and clear messages), it can automatically adjust the threshold. When this adjustment occurs, the following message is written to the log:

```
WSVR0607W: Too many thread hangs have been falsely reported.  The hang
threshold is now being set to thresholdtime.
```

where: *thresholdtime* is the time (in seconds) in which a thread can be active before it is considered hung.

You can prevent WebSphere Application Server from automatically adjusting the hang time threshold. See "Configuring the hang detection policy" on page 197

Adjusting the hang detection policy of a running server

You can adjust the thread monitor settings by using the wsadmin scripting interface. These changes take effect immediately, but do not persist to the server configuration, and are lost when the server is restarted. The following script provides an example of how to adjust the properties for the thread monitor using the wsadmin tool:

```
# Read in the interval, threshold, false alarm from the command line
set interval [lindex $argv 0]
set threshold [lindex $argv 1]
set adjustment [lindex $argv 2]

# Get the object name of the server you want to change the values on
set server [$AdminControl completeObjectName "type=Server,*"]

# Read in the interval and print to the console
set i [$AdminControl getAttribute $server threadMonitorInterval]

# Read in the threshold and print to the console
set t [$AdminControl getAttribute $server threadMonitorThreshold]

# Read in the false alarm adjustment threshold and print to the console
set a [$AdminControl getAttribute $server threadMonitorAdjustmentThreshold]

# Set the new values using the command line parameters
$AdminControl setAttribute $server threadMonitorInterval  ${interval}

$AdminControl setAttribute $server threadMonitorThreshold ${threshold}

$AdminControl setAttribute $server threadMonitorAdjustmentThreshold ${adjustment}
```

Related concepts

"Detecting hung threads in J2EE applications" on page 195

Related tasks

"Configuring the hang detection policy" on page 197

Configuring the hang detection policy

The thread hang detection option is enabled by default. To adjust the hang detection policy values, or to disable hang detection completely:

1. From the administrative console, click **Servers** > **Application Servers** > *server_name*
2. Under Server Infrastructure, click **Administration** > **Custom Properties**
3. Click **New**.
4. Add the following properties:

   ```
   Name: com.ibm.websphere.threadmonitor.interval
   Value:  The frequency (in seconds) at which managed threads in the selected
           application server will be interrogated.
   Default: 180 seconds (three minutes).

   Name: com.ibm.websphere.threadmonitor.threshold
   Value:   The length of time (in seconds) in which a thread can be active
            before it is considered hung.  Any thread that is detected as
            active for longer than this length of time is reported as hung.
   Default: The default value is 600 seconds (ten minutes).

   Name:com.ibm.websphere.threadmonitor.false.alarm.threshold
   Value: The number of times (T) that false alarms can occur
          before automatically increasing the threshold. It is possible that a
          thread that is reported as hung eventually completes its work,
          resulting in a false alarm.  A large number of these events indicates
          that the threshhold value is too small. The hang detection facility can
          automatically respond to this situation: For every T false alarms, the
          threshold T is increased by a factor of 1.5. Set the value to
          zero (or less) to disable the automatic adjustment.
   Default: 100
   ```

   To disable the hang detection option, set the **com.ibm.websphere.threadmonitor.interval** property to less than or equal to zero.
5. Click **Apply**.
6. Click **OK**.
7. Save the changes and make sure a file synchronization is performed before restarting the servers.
8. Restart the Application Server for the changes to take effect.

Related concepts

"Detecting hung threads in J2EE applications" on page 195

Related reference

"Adjusting the hang detection policy of a running server" on page 196

Working with trace

Use trace to obtain detailed information about the execution of WebSphere Application Server components, including application servers, clients, and other processes in the environment. Trace files show the time and sequence of methods called by WebSphere Application Server base classes, and you can use these files to pinpoint the failure.

Collecting a trace is often requested by IBM technical support personnel. If you are not familiar with the internal structure of WebSphere Application Server, the trace output might not be meaningful to you.

1. Configure an output destination to which trace data is sent.
2. Enable trace for the appropriate WebSphere Application Server or application components.
3. Run the application or operation to generate the trace data.
4. Analyze the trace data or forward it to the appropriate organization for analysis.

Setting up trace for IBM WebSphere Application Server for z/OS is documented in detail in the WebSphere Application Server for z/OS Diagnosis Guide.

For current information available from IBM Support on known problems and their resolution, see the IBM Support page.

IBM Support has documents that can save you time gathering information needed to resolve this problem. Before opening a PMR, see the IBM Support page.

Enabling tracing and logging

You can configure the application server to start in a trace-enabled state by setting the appropriate configuration properties. You can only enable trace for an application client or standalone process at process startup.

In WebSphere Application Server version 6.0.x, a logging infrastructure, extending Java Logging, is used. This results in the following changes to the configuration of the logging infrastructure in WebSphere Application Server:

- Loggers defined in Java logging are equivalent to, and configured in the same way as, trace components introduced in previous versions of WebSphere Application Server. Both are referred to as "components."
- Both Java logging levels and WebSphere Application Server levels can be used. The following is a complete list of valid levels, ordered in ascending order of severity:
  1. all
  2. finest or debug
  3. finer or entryExit
  4. fine or event
  5. detail
  6. config
  7. info
  8. audit
  9. warning
  10. severe or error
  11. fatal
  12. off
- Setting the logging and tracing level for a component to all will enable all the logging for that component. Setting the logging and tracing level for a component to off will disable all the logging for that component.
- You can only configure a component to one level. However, configuring a component to a certain level enables it to perform logging on the configured level and any higher severity level.
- Several levels have equivalent names: finest is equivalent to debug; finer is equivalent to entryExit; fine is equivalent to event; severe is equivalent to error.

Java Logging does not distinguish between tracing and message logging. However, previous versions of WebSphere Application Server have made a clear distinction between those kind of messages. In WebSphere Application Server version 6.0.x, the differences between tracing and message logging are as follows:

- Tracing messages are messages with lower severity (for example, tracing messages are logged on levels fine, finer, finest, debug, entryExit, or event).
- Tracing messages are generally not localized.
- When tracing is enabled, a much higher volume of messages will be produced.
- Tracing messages provide information for problem determination.

Trace and logging strings

In WebSphere Application Server version 5.1.1 and earlier, trace strings were used for configuring tracing only. In WebSphere Application Server version 6.0.x, the ″trace string″ becomes a ″logging string″; it is used to configure both tracing and message logging.

In WebSphere Application Server version 5.1.1 and earlier, the trace service for all WebSphere Application Server components is disabled by default. To request a change to the current state of the trace service, a trace string is passed to the trace service. This trace string encodes the information detailing which level of trace to enable or disable and for which components.

In WebSphere Application Server, the tracing for all components is disabled by default. To change to the current state of the tracing and message logging, a logging string must be constructed and passed to the server. This logging string specifies what level of trace or logging to enable or disable for specific components.

You can type in trace strings (or logging strings), or construct them using the administrative console. Trace and logging strings must conform to a specific grammar.

For WebSphere Application Server version 5.1.1 and earlier, the specification of this grammar is as follows:

```
TRACESTRING=COMPONENT_TRACE_STRING[:COMPONENT_TRACE_STRING]*

 COMPONENT_TRACE_STRING=COMPONENT_NAME=LEVEL=STATE[,LEVEL=STATE]*

LEVEL = all | entryExit | debug | event

STATE = enabled | disabled

COMPONENT_NAME = COMPONENT | GROUP
```

For WebSphere Application Server version 6.0.x, the previous grammar is supported. However a new grammar has been added to better represent the underlying infrastructure:

```
LOGGINGSTRING=COMPONENT_LOGGING_STRING[:COMPONENT_LOGGING_STRING]*

 COMPONENT_TRACE_STRING=COMPONENT_NAME=LEVEL

LEVEL = all | (finest | debug) | (finer | entryExit) | (fine | event )
| detail | config | info | audit | warning | (severe | error) | fatal | off

COMPONENT_NAME = COMPONENT | GROUP
```

The COMPONENT_NAME is the name of a component or group registered with the trace service logging infrastructure. Typically, WebSphere Application Server components register using a fully qualified Java class name, for example com.ibm.servlet.engine.ServletEngine. In addition, you can use a wildcard character of asterisk (*) to terminate a component name and indicate multiple classes or packages. For example, use a component name of com.ibm.servlet.* to specify all components whose names begin with com.ibm.servlet. You can use a wildcard character of asterisk (*) at the end of the component or group name to make the logging string applicable to all components or groups whose names start with specified string. For example, a logging string specifying ″com.ibm.servlet.*″ as a component name will be applied to all components whose names begin with com.ibm.servlet. When ″*″ is used by itself in place of the component name, the level the string specifies, will be applied to all components.

Note:

- In WebSphere Application Server version 5.1.1 and earlier, you could set the level to ″all=disabled″ to disable tracing. This syntax in version 6 will result in LEVEL=info; tracing will be disabled, but logging will be enabled.
- The logging string is processed from left to right. During the processing, part of the logging string might be modified or removed if the levels they configure are being overridden by another part of the logging string.
- In WebSphere Application Server version 6.0.x, ″info″ is the default level. If the specified component is not present (*=xxx is not found), *=info is always implied. Any component that is not matched by the trace string will have its level set to info.
- If the logging string does not start with a component logging string specifying a level for all components, using the ″*″ in place of component name, one will be added, setting the default level for all components.
- STATE = enabled | disabled is not needed in version 6. However, if used, it has the following effect:
  - ″enabled″ sets the logging for the component specified to the level specified
  - ″disabled″ sets the logging for the component specified to one level above the level specified. The following examples illustrate the effect that disabling has on the logging level:

Logging string	Resulting logging level	Notes
com.ibm.ejs.ras=debug=disabled	com.ibm.ejs.ras=finer	debug (version 5) = finest (version 6)
com.ibm.ejs.ras=all=disabled	com.ibm.ejs.ras=info	″all=disabled″ will disable tracing; logging is still enabled.
com.ibm.ejs.ras=fatal=disabled	com.ibm.ejs.ras=off	
com.ibm.ejs.ras=off=disabled	com.ibm.ejs.ras=off	off is the highest severity

Examples of legal trace strings include:

Version 5 syntax	Version 6 syntax
`com.ibm.ejs.ras.ManagerAdmin=debug=enabled`	`com.ibm.ejs.ras.ManagerAdmin=finest`
`com.ibm.ejs.ras.ManagerAdmin=all=enabled,event=disabled`	`com.ibm.ejs.ras.ManagerAdmin=detail`
`com.ibm.ejs.ras.*=all=enabled`	`com.ibm.ejs.ras.*=all`

Version 5 syntax	Version 6 syntax
com.ibm.ejs.ras.*=all=enabled:com.ibm.ws.ras=debug=enabled,entryexit=enabled	com.ibm.ejs.ras.*=all:com.ibm.ws.ras=finer

Enabling trace on client and standalone applications

When standalone client applications (such as Java applications which access enterprise beans hosted in WebSphere Application Server) have problems interacting with WebSphere Application Server, it may be useful to enable tracing for the application. Enabling trace for client programs will cause the WebSphere Application Server classes used by those applications, such as naming-service client classes, to generate trace information. A common troubleshooting technique is to enable tracing on both the application server and client applications, and match records according to timestamp to try to understand where a problem is occurring.

1. To enable trace for the WebSphere Application Server classes in a client application, add the system properties shown in the following example to the startup script or command of the client application. The location of the output and the classes and detail included in the trace follow the same rules as for adding trace to WebSphere Application Servers. For example, trace the standalone client application program named `com.ibm.sample.MyClientProgram`, you would enter the following command:

   ```
   java -DtraceSettingsFile=MyTraceSettings.properties
   -Djava.util.logging.manager=com.ibm.ws.bootstrap.WsLogManager
   -Djava.util.logging.configureByServer=true com.ibm.samples.MyClientProgram
   ```

   The file identified by *filename* must be a properties file placed in the classpath of the application client or stand-alone process. An example file is provided in *install_root*`/properties/TraceSettings.properties`.

   You cannot use the **-DtraceSettingsFile=TraceSettings.properties** property to enable tracing of the ORB component for thin clients. ORB tracing output for thin clients can be directed by setting **com.ibm.CORBA.Debug.Output = debugOutputFilename** parameter in the command line.

   The java.util.logging.manager and java.util.logging.configureByServer system properties configure Java logging to use a WebSphere Application Server-specific LogManager class and to use the configuration from the file specified by the traceSettingsFile property. The default Java Logging properties file, located in the Java Runtime Environment (JRE), will not be applied.
2. You can configure the `MyTraceSettings.properties` file to send trace output to a file using the `traceFileName` property. Specify one of two options:
   - The fully qualified name of an output file. For example, `traceFileName=c:\\MyTraceFile.log`. You must specify this property to generate visible output.
   - `stdout`. When specified, output is written to `System.out`.
3. You can also specify a trace string for writing messages with the `Trace String` property, Specify a startup trace specification similar to that available on the server. For your convenience, you can enter multiple individual trace strings into the trace settings file, one trace string per line.

Here are the results of using each optional property setting:

- Specify a valid setting for the `traceFileName` property without a trace string to write messages to the specified file or `System.out` only.

- Specify a trace string without a `traceFileName` property value to generate no output.
- Specify both a valid `traceFileName` property and a trace string to write both message and trace entries to the location specified in the `traceFileName` property.

Automation and recovery scenarios and guidelines

The following section provides information on how to monitor and recover WebSphere Application Server for z/OS and the subsystems it uses. It provides startup, shutdown, and recovery procedures and scenarios. It also tells you how to determine if the subsystems are up or down, and tells you where to find more information.

APPC automation and recovery scenarios

Task	APPC automation and recovery scenarios
Startup	APPC should be started before WebSphere Application Server for z/OS. In theory, WebSphere Application Server for z/OS *could* be started before APPC, but only as long as no objects get dispatched in containers that have an IMS APPC LRMI associated with them. If APPC is not up before WebSphere Application Server for z/OS, and you want to use an APPC connector to talk to IMS, then you will have no connectivity. APPC/MVS does not have to be up for CICS. APPC does not have to be started after VTAM.
Shutdown	Reverse the startup procedure. Shutdown WebSphere Application Server for z/OS, APPC, then VTAM.
Handling in-flight or indoubt transactions if there is a failure	If you are using APPC for communications and it fails, do the following: 1. Shutdown all servers with APPC connectivity. 2. Restart APPC (if it totally failed). 3. Restart the WebSphere Application Server for z/OS server. **Note:** APPC will resynchronize itself. If your transaction is indoubt, IMS waits until you restart APPC. IMS relies on RRS for recovery. RRS will resolve transactions that are in doubt by handshaking with every subsystem it was communicating with before it went down. If you are using CICS, note that CICS has its own coordinator.
How to determine if APPC is running	Issue the `DISPLAY APPC,LU,ALL` command. If APPC is not active, it will say so. In addition, the status of the logical units used by WebSphere Application Server for z/OS and/or IMS should be active or no APPC work will be successful.
What happens to WebSphere for z/OS if APPC goes down?	Any objects attempting to use the IMS APPC PAA will not work. The server region running on behalf of the container attempting to use APPC will likely get a `C9C24C05` error, indicating that an APPC ALLOCATE request was attempted and failed. Additional APPC error diagnostic information that helps to pinpoint the APPC problem is contained in the logs associated with this region.

Task	APPC automation and recovery scenarios
What happens to other subsystems if APPC goes down?	Not applicable
Where to find more information	• *z/OS MVS Planning: Operations* • *z/OS MVS Planning: APPC/MVS Management* • *z/OS MVS Programming: Resource Recovery*

WLM automation and recovery scenarios

Task	WLM automation and recovery scenarios
Startup	WLM is automatically started by z/OS or OS/390 when you IPL your system. You do not have to start it.
Shutdown	You cannot shutdown WLM.
Handling in-flight and indoubt transactions if there is a failure	Not applicable
How to determine if WLM is running	Not applicable
What happens to WebSphere Application Server for z/OS if WLM goes down?	Not applicable
What happens to other subsystems if WLM goes down?	Not applicable

How to handle a catastrophic failure of the WebSphere Application Server for z/OS server regions	Following a catastrophic failure of the WebSphere Application Server for z/OS server regions, you can use one of the following resume commands, depending on your application environment type:
	• Static application environment:
	`v wlm,applenv=applenvname,`
	`resume`
	• Dynamic application environment:
	`v wlm,dynappl=applenvname,`
	`resume,options`

Where to find more information	• *z/OS MVS Planning: Workload Management* • *z/OS MVS Programming: Workload Management Services*

RACF automation and recovery scenarios

Task	RACF automation and recovery scenarios
Startup	If it is installed, RACF is started as a part of IPL.
Shutdown	RACF is not shutdown.
Handling in-flight and indoubt transactions if there is a failure	Not applicable
How to determine if RACF is running	Use the RACF SETROPTS command to display the status of RACF.

Task	RACF automation and recovery scenarios
What happens to WebSphere for z/OS if RACF goes down?	RACF goes into fail safe mode. This means that for every resource that is accessed, the operator is asked to verify if it is okay. In general, the system is IPLed if this occurs.
What happens to other subsystems if RACF goes down?	It depends on the subsystem and how RACF fails.
Where to find more information	• *z/OS Security Server RACF System Programmer's Guide* • *z/OS Security Server RACF Security Administrator's Guide*

RRS automation and recovery scenarios

Task	RRS automation and recovery scenarios
Startup	Ensure System Logger has been started before RRS. **Note:** RRS will display error messages indicating that System Logger must be started first if you try to start RRS without starting System Logger.Ensure RRS is started before WebSphere for z/OS. RRS does not start by itself. RRS will start automatically only if it was registered with the Automatic Restart Manager (ARM) and if ARM is running. To start RRS, issue the `start` command: `start atrrrs,sub=master` **Note:** RRS doesn't restart itself if you issue the cancel command, so you need to restart it manually if it was canceled or if ARM isn't running.
Shutdown	Shutdown RRS in the reverse order that you started RRS. Shutdown WebSphere Application Server for z/OS, then RRS, followed by System Logger. There is no controlled way to bring down RRS. The best approach is: 1. Quiesce WebSphere Application Server for z/OS. 2. Shutdown WebSphere Application Server for z/OS. 3. Cancel RRS. **Note:** You may want to bring down the DB2 you are using for WebSphere Application Server for z/OS before canceling RRS. To cancel RRS, issue the command: `setrrs cancel`

Task	RRS automation and recovery scenarios
Handling in-flight and indoubt transactions if there is a failure	Refer to the RRS system management panels to display in-flight and resolve indoubt transactions. You can display the resource managers on the RM panels in RRS, display all units of recovery (UR), filter the URs, and then resolve the indoubts. You cannot resolve in-flights. You can display all RRS-managed transactions. If you are using the IMS Connector for Java, this process applies only if IMS Connector for Java, IMS Connect, and the IMS subsystem have been configured locally on the same z/OS or OS/390 system image on which the WebSphere for z/OS J2EE server runs. The local configuration is the only configuration in which IMS Connector for Java runs as an RRS-transactional connector.
How to determine if RRS is running	Use the display command: `d a,atrrs` `atrrs` is the name of the default RRS procedure shipped with WebSphere Application Server for z/OS. Use the procedure name that you use to start RRS. The address space comes from the procedure.
What happens to WebSphere for z/OS if RRS goes down?	RRS is a required subsystem, so WebSphere Application Server for z/OS will not run without it. If RRS goes down, WebSphere Application Server for z/OS will get fatal errors. You need to get RRS started, then restart WebSphere Application Server for z/OS.
What happens to other subsystems if RRS goes down?	RRS is the z/OS or OS/390 transaction monitor. If you cancel RRS, you will have problems with any subsystems using it (for example, WebSphere Application Server for z/OS, DB2, IMS). You should understand the implications before you cancel RRS.
Where to find more information	*z/OS MVS Programming: Resource Recovery*

UNIX System Services automation and recovery scenarios

Task	UNIX System Services automation and recovery scenarios
Startup	UNIX System Services is a permanent component of MVS and is started automatically at IPL time.
Shutdown	UNIX System Services does not support a shutdown capability, so it is always available.
Handling in-flight or indoubt transactions if there is a failure	The only data that could be considered transactional in nature is data stored in the HFS.
How to determine if UNIX System Services is running	UNIX System Services is always available as long as the system is up and running.
What happens to WebSphere for z/OS if UNIX System Services goes down?	If UNIX System Services fails, the system must be re-IPLed. WebSphere Application Server for z/OS will get errors and terminate.
What happens to other subsystems if UNIX System Services goes down?	If UNIX System Services fails, the system must be re-IPLed.

Task	UNIX System Services automation and recovery scenarios
Where to find more information	*z/OS UNIX System Services Planning*

TCP/IP automation and recovery scenarios

Task	TCP/IP automation and recovery scenarios
Startup	TCP/IP must be up before starting WebSphere Application Server for z/OS.
Shutdown	Shutdown WebSphere Application Server for z/OS before shutting down TCP/IP.
Handling inflight or indoubt transactions if there is a failure	Methods in flight will have their transactions rolled back when the attempt to send a response to the method fails. Other transactions will wait for a timeout.
How to determine if TCP/IP is running	Use the display command looking for the TCP/IP procedure.
What happens to WebSphere Application Server for z/OS if TCP/IP goes down?	If TCP/IP goes down, then WebSphere Application Server for z/OS on the system must be restarted. You will get an SVC dump because the socket layer was destroyed.
What happens to other subsystems if TCP/IP goes down?	If TCP/IP goes down, sessions break and transactions react as described above. **Note:** WebSphere Application Server for z/OS will not be able to recognize that TCP/IP is back up. Therefore, WebSphere Application Server for z/OS must be restarted. **Note:** If TCP/IP goes down, you should recycle LDAP before restarting WebSphere Application Server for z/OS.

DB2 automation and recovery scenarios

Task	DB2 automation and recovery scenarios
Startup	DB2 is started after RRS but before LDAP, NFS, and WebSphere Application Server for z/OS.
Shutdown	Reverse of startup sequence.

Task	DB2 automation and recovery scenarios

Handling in-flight or indoubt transactions if there is a failure	Use the RRS panels to resolve. See *z/OS MVS Programming: Resource Recovery*. The RRS panels are the preferred way to resolve DB2 indoubts because they allow you to view all resource managers that have an interest in the transaction. However, you can also use DB2 to resolve indoubts. You can issue the command:
	`DISPLAY THREAD(*) TYPE(INDOUBT)`
	
	to display DB2 information about the indoubt threads it knows about (if there are too many, you can go into S.LOG to view the information). This display will give you a DB2 identifier called a "nid". Copy the nid and paste it into this command:
	`-RECOVER INDOUBT (RRSAF) ACTION(COMMIT) NID(B1D379D17ED6CF900000009401010000)`
	
	where the `nid` is the one that you cut from the display command. You can issue this command to roll back the transaction:
	`-RECOVER INDOUBT (RRSAF) ACTION(ABORT) NID(B1D379D17ED6CF900000009401010000)`

How to determine if DB2 is running	Use the display command to display the DB2 address space.
What happens to WebSphere Application Server for z/OS if DB2 goes down?	WebSphere Application Server for z/OS needs DB2 to run, so it abends if DB2 goes down. You need to restart the LDAP server, then restart DB2, and then restart WebSphere Application Server for z/OS.
What happens to other subsystems if DB2 goes down?	Not applicable
Where to find more information	See the DB2 books at the following Internet location:http://www.ibm.com/servers/eserver/zseries/zos/

CICS automation and recovery scenarios

Task	CICS automation and recovery scenarios
Startup	CICS and any required CICS products, such as CICS Transaction Gateway, need to be properly installed, initialized, and started before any work flows to a CICS-enabled WebSphere Application Server controller.
Shutdown	Shutdown the WebSphere Application Server controller that uses CICS as a backing store, then shut down the CICS service.

Task	CICS automation and recovery scenarios
Handling in-flight or indoubt transactions if there is a failure	If there is an error during processing, both CICS and WebSphere Application Server for z/OS rely on the underlying RRS subsystem to handle all rollback notifications to the registered interests. In the case of inflight transactions, RRS will notify all participants that a rollback is required, and normal rollback processing will occur in each registered party. In the case of indoubt transactions, it may be necessary to recycle the WebSphere Application Server for z/OS Application Control/Server region to release any pending transaction in CICS.
How to determine if CICS is running	This is installation dependent.
What happens to CICS if WebSphere Application Server for z/OS goes down?	Should WebSphere Application Server for z/OS happen to go down, one of two situations could occur: 1. If WebSphere Application Server for z/OS and CICS are currently engaged in a unit of work, then RRS processing as described above would occur and it may be necessary to recycle the application control server regions to release pending transactional work in CICS. 2. If WebSphere Application Server for z/OS and CICS are not currently engaged in a unit of work, CICS is not affected.
What happens to other subsystems if CICS goes down?	Not applicable
Where to find more information	*CICS Operations and Utilities Guide*

IMS automation and recovery scenarios

Task	IMS automation and recovery scenarios
Startup	IMS and any required IMS products, such as IMS Connect, need to be properly installed, initialized, and started before any work flows to an IMS-enabled WebSphere for z/OS application control server region are run.
Shutdown	Shutdown the WebSphere Application Server for z/OS application controller that uses IMS as a backing store, then shutdown the IMS service
Handling in-flight or indoubt transactions if there is a failure	If there is an error during processing, both IMS and WebSphere Application Server for z/OS rely on the underlying RRS subsystem to handle all rollback notifications to the registered interests. In the case of inflight transactions, RRS will notify all participants that a rollback is required and normal rollback processing will occur in each registered party. In the case of indoubt transactions, it may be necessary to recycle the WebSphere Application Server for z/OS Application Control/Server region to release any pending transaction in the IMS MPRs.
How to determine if IMS is running	This is installation-dependent.

Task	IMS automation and recovery scenarios
What happens to IMS if WebSphere Application Server for z/OS goes down?	Should WebSphere Application Server for z/OS happen to go down, one of two situations could occur: 1. If WebSphere Application Server for z/OS and IMS are currently engaged in a unit of work, then RRS processing as described above would occur and it may be necessary to recycle the application control server regions to release pending transactional work in the IMS MPR. 2. If WebSphere Application Server for z/OS and IMS are not currently engaged in a unit of work, IMS is not affected.
What happens to other subsystems if IMS goes down?	Not applicable
Where to find more information	*IMS/ESA Operator's Reference*

LDAP automation and recovery scenarios

Task	LDAP automation and recovery scenarios
Startup	LDAP, as used by WebSphere Application Server for z/OS, is completely run within the WebSphere Application Server for z/OS address spaces using something called "the local backend." This support takes the front side of the LDAP client APIs and the backend database implementation and runs them completely inside the WebSphere Application Server for z/OS Naming Server and Interface Repository. For Naming and IR, OMVS and DB2 must be up *before* Naming and IR. To run the LDAP server, TCPIP, OMVS, and DB2 must all be up before the LDAP server. **Note:** There are two LDAP modes supported: 1. Local LDAP backend. 2. Remote LDAP Server. The WebSphere Application Server for z/OS environment has to be set up correspondingly, and DB2, TCP/IP, and the remote server have to be up and running before WebSphere Application Server for z/OS is started.
Shutdown	Shutdown Naming and IR, then OMVS and DB2. For the LDAP server, shutdown the LDAP server, then TCPIP and DB2, and then OMVS.
Handling in-flight or indoubt transactions if there is a failure	If there is a failure during processing, Naming and IR rely on RRS to issue a rollback directly to DB2 and, as a result, any work done by the LDAP code is rolled back along with it. For the LDAP server, AUTOCOMMIT is set to NO, causing any error to ROLLBACK for that transaction. This ensures the atomicity characteristic of LDAP operations.

Task	LDAP automation and recovery scenarios
How to determine if LDAP is running	In the case of WebSphere Application Server for z/OS, if Naming and IR are operating, then LDAP is operating. In the case of the LDAP server, and a started task is used for the LDAP server, use the SDSF to see if the started task is running. Examine the output log for the started task to see if any error messages were displayed. Alternatively, the `LDAPSRCH` command (from TSO), or `LDAPSEARCH` command (from Unix System Services shell) can be used to perform a simple search to verify that the LDAP server is running.
What happens to WebSphere Application Server for z/OS if LDAP goes down?	• In J2EE server regions, the LDAP server **must** be active since it is a separate server that no longer runs inside the WebSphere Application Server for z/OS region. Recycle the JNDI, then restart the J2EE application servers.
What happens to other subsystems if LDAP goes down?	Most z/OS or OS/390 subsystems do not depend on LDAP, but this may change in the future. In the case of accessing LDAP through the LDAP server, there is a way to configure the LDAP server to operate in a sysplex environment such that (using sysplex-enabled DNS) LDAP requests will be sent to the LDAP server in the sysplex that is operating (assuming that there is one). As an alternative, subsystems that want to use LDAP could configure a backup LDAP server to be contacted in case the primary server is not accessible. In this case, the application would assume that it could retrieve all of the same data that it could get from the backup on the primary which would be handled by some replication mechanism. The LDAP server currently supports a master/slave replication mechanism, but you could also try duplicating the sysplex server using DB2 data sharing.
Where to find more information	*z/OS Security Server LDAP Server Administration and Use*

WebSphere Application Server for z/OS (Daemon) automation and recovery scenarios

Task	WebSphere for z/OS (daemon) automation and recovery scenarios
Startup	See the instructions for ″Starting the WebSphere for z/OS environment″.
Shutdown	See the instructions for ″Stopping the WebSphere for z/OS environment″.
Handling inflight or indoubt transactions if there is a failure	The daemon is a location agent. If the daemon fails during the course of a transaction, locate requests to the daemon will fail. These request failures will be surfaced by the client ORB. If the client is a WebSphere Application Server for z/OS client running in a sysplex, the locate request will be routed to another available daemon in the sysplex, if present.
How to determine if the daemon is running	Use the MVS display command.

Task	WebSphere for z/OS (daemon) automation and recovery scenarios
What happens to WebSphere Application Server for z/OS if the daemon goes down?	If the daemon goes down, all WebSphere Application Server for z/OS servers started on the same system will also be terminated.
What happens to other subsystems if the daemon goes down?	Other subsystems will continue to work fine. As a general rule, if the servers on multiple systems are defined in cluster and there is another one in the sysplex, if the daemon goes down , clients will not be affected.

Web server (servlet) automation and recovery scenarios

Task	WebServer automation and recovery scenarios
Startup	Web servers have a relationship with WebSphere Application Server for z/OS only in the sense that a client application program that is written to use WebSphere Application Server for z/OS facilities may be written as a servlet. Any implications for ordering of startup will be introduced by the applications. You probably want to have the WebSphere Application Server for z/OS object servers up and ready before starting the client application that the web server is hosting.
Shutdown	There are no dependencies from the product code. Similar to most applications, you may want to quiesce the clients prior to taking down the target WebSphere Application Server for z/OS servers. Shut down the web server to stop the port of entry.
Handling in-flight or indoubt transactions if there is a failure	Since a web server is stateless, there are no in-flight or indoubt transactions.
How to determine if a web server is running	Use the z/OS display commands and viewer tools (SDDF) to monitor the Application Server.
What happens to WebSphere Application Server for z/OS if the web server goes down?	WebSphere Application Server for z/OS requires an IBM HTTP J2EE server web server in order to provide full function servlets. So, if the web server goes down, applications that require a port of entry (like servlets and SOAP) cannot run.
What happens to other subsystems if Web Server goes down?	HTTP only provides information to other subsystems, so they are unaffected if the web server goes down.
Where to find more information	*z/OS HTTP Server Planning, Installing, and Using*

Types of configuration variables

WebSphere Application Server for z/OS provides configuration variables that allow you to control:

- Output destinations and characteristics for the error log, and for CTRACE buffers, data sets and the external writer.
- Trace buffers, data sets, and the content of trace data.
- Types of dumps to be requested.
- Timeout values for system and application behavior.

Setting output destinations and characteristics

client_ras_logstreamname= *name* Specifies the name of the log stream for an application client run-time to use for error information.

Default: If this variable is not specified, the client run-time uses STDERR.

Example:

```
client_ras_logstreamname=MY.CLIENT.ERROR.LOG
```

Tip: Do not put the log stream name in quotes. Log stream names are not data set names.

ras_default_msg_dd= *DD_card_name* Redirects write-to-operator (WTO) messages that use the default routing to hardcopy. These messages are redirected to the location identified through the DD card on the server's JCL start procedure. These WTO messages are primarily messages that WebSphere Application Server for z/OS issues during initialization.

Default: No default value is set; WTO messages that use default routing are sent to hardcopy.

Examples:

```
ras_default_msg_dd=MSGDD
ras_default_msg_dd=DFLTLOG
```

Example of the DFLTLOG DD card on the server's JCL start procedure:

```
//DFLTLOG    DD   SYSOUT=*
```

ras_hardcopy_msg_dd= *DD_card_name* Redirects write-to-operator (WTO) messages that WebSphere Application Server for z/OS routes to hardcopy. These messages are redirected to the location identified through the DD card on the server's JCL start procedure. These WTO messages are primarily audit messages issued from Java code during initialization.

Default: No default value is set; WTO messages that use hardcopy routing are sent to hardcopy.

Example:

```
ras_hardcopy_msg_dd=MSGDD
```

ras_log_logstreamName= *name* Specifies the log stream for WebSphere Application Server for z/OS to use for error information during bootstrap processing. If the specified log stream is not found or not accessible, a message is issued and errors are written to the server's job log.

Default: If this variable is not specified, WebSphere Application Server for z/OS uses STDERR.

Example:

```
ras_log_logstreamName=MY.CB.ERROR.LOG
```

Tip: Do not put the log stream name in quotes. Log stream names are not data set names.

Setting trace controls

The following trace options allow you to capture the information needed to detect problems. To view or set these options, use the WebSphere Application Server administrative console:

1. Select **Environment > Manage WebSphere Variables**.
2. Specify the variable name in the name field and specify the setting in the value field. You can also describe the setting in the description field on this tab.

ras_trace_outputLocation=SYSPRINT | BUFFER | TRCFILE
Specifies where you want trace records to be sent:

- To SYSPRINT
- To a memory buffer (BUFFER), the contents of which are later written to a CTRACE data set
- To a trace data set (TRCFILE) specified on the `TRCFILE DD` statement in the server's start procedure.

For servers, you may specify one or more values, separated by a space. For clients, you may specify SYSPRINT only.

Defaults:

- For clients, SYSPRINT
- For all other processes, BUFFER

Example: `ras_trace_outputLocation=SYSPRINT BUFFER`

ras_time_local=0 | 1
Specifies whether timestamps in trace records use Greenwich Mean Time (GMT) or local time. This variable setting controls timestamp format in the error log, and in traces sent to SYSPRINT or TRCFILE DD.

Default: 0 (GMT)

Example: `ras_time_local=1` sets timestamps to local time.

ras_trace_ctraceParms=SUFFIX | MEMBER_NAME
Identifies the CTRACE PARMLIB member. The value can be either:

- A two-character suffix, which is added to the string `CTIBBO` to form the name of the PARMLIB member, or
- The fully specified name of the PARMLIB member. A fully specified name must conform to the naming requirements for a CTRACE PARMLIB member.

If this environment variable is specified and the PARMLIB member is not found, the default PARMLIB member, CTIBBO00, is used. If neither the specified nor the default PARMLIB member is found, tracing is defined to CTRACE, but there is no connection to a CTRACE external writer.

Note: The Daemon is the only server that recognizes this environment variable.

Default: 00 (the default PARMLIB member, CTIBBO00)

Example: `ras_trace_ctraceParms=01` identifies PARMLIB member CTIBBO01

ras_trace_BufferCount= *n*
Specifies the number of trace buffers to allocate. Valid values are 4 through 8.

Default: 4

Example: `ras_trace_BufferCount=6`

ras_trace_BufferSize= *n*

Specifies the size of a single trace buffer in bytes. You can use the letters K (for kilobytes) or M (for megabytes). Valid values are 128K through 4M.

Default: 1M

Example: `ras_trace_BufferSize=2M`

Related reference

"Setting dump controls"

Setting dump controls

ras_dumpoptions_dumptype= *n*

Specifies the default dump used by the signal handler. Valid values and their meanings are:

- 0

  No dump is generated.
- 1

  A ctrace dump is taken.
- 2

  A cdump dump is taken.
- 3

  A csnap dump is taken.
- 4

  A CEE3DMP dump is taken.

  CEE3DMP generates a dump of Language Environment and the member language libraries. Sections of the dump are selectively included, depending on dump options specified, either by default or through the

  `ras_dumpoptions_ledumpoptons`

  variable. By default, this value passes

  `THREAD(ALL) BLOCKS`

  to CEE3DMP. You can override the default options for CEE3DMP through the

  `ras_dumpoptions_ledumpoptons`

  variable. For more information about CEE3DMP and its options, see z/OS Language Environment Programming Reference, SA22-7562..

Default: 3

Example:

`ras_dumpoptions_dumptype=2`

ras_dumpoptions_ledumpoptons= *options*

Specifies dump options to be used with a CEE3DMP. If you want more than one option, separate each option with a blank. Specifies dump options to be used with a CEE3DMP. If you want more than one option, separate each option with a blank.

This WebSphere variable is used only when you specify

`ras_dumpoptions_dumptype=4`

. For an explanation of CEE3DMP and valid dump options, see z/OS Language Environment Programming Reference, SA22-7562.

Rule: The maximum length of the option string on this environment variable is 255. If the option string is longer than 255, you receive message BBOM0011W and the CEE3DMP dump options are set to

```
THREAD(ALL) BLOCKS
```

.

Default:

```
THREAD(ALL) BLOCKS
```

Example:

```
ras_dumpoptions_ledumpoptons=NOTRACEBACK NOFILES
```

Controlling behavior through timeout values

Timeout variables let you control the amount of time you will allow for various requests to complete. Some of these variables map to internal variable names. The internal variable names are provided here to aid you with debugging.

ORB service advanced settings

ORB listener keep alive
This variable defines the value, in seconds, provided to TCP/IP on the *SOCK_TCP_KEEPALIVE* option for the IIOP listener. The function of this option is to verify if idle sessions are still valid by polling the client TCP/IP stack. If the client does not respond, the session is closed. If the client goes away without notifying the server, it would unnecessarily leave the session active on the server side. Use this option to clean up these unnecessary sessions.

- If the environment variable is not set, the TCP/IP option is not set.
- Setting the *SOCK_TCP_KEEPALIVE* option generates network traffic on idle sessions, which can be undesirable.

Default: 0

How to specify: Specify this setting in the administrative console using the path **Application Servers** > *server* > **ORB Service > Advanced Settings**

Internal variable name (for debugging purposes): Look for the internal variable name *protocol_iiop_server_session_keepalive* in the was.env file or the JES job log.

ORB SSL listener keep alive
This variable defines the value, in seconds, provided to TCP/IP on the *SOCK_TCP_KEEPALIVE* option for the IIOP listener. The function of this option is to verify if idle sessions are still valid by polling the client TCP/IP stack. If the client does not respond, the session is closed. If the client goes away without notifying the server, it would unnecessarily leave the session active on the server side. Use this option to clean up these unnecessary sessions.

- If the environment variable is not set, the TCP/IP option is not set.
- Setting the *SOCK_TCP_KEEPALIVE* option generates network traffic on idle sessions, which can be undesirable.

Default: 0

How to specify: Specify this custom property in the administrative console using the path **Application Servers** > *server* > **ORB Service > Advanced Settings**

Internal variable name (for debugging purposes): Look for the internal variable name *protocol_iiop_server_session_keepalive_ssl* in the was.env file or the JES job log.

WLM timeout

Specifies the maximum amount of time, in seconds, that WebSphere Application Server for z/OS will wait for IIOP requests to complete. This time limit includes:

- Time during which the IIOP request waits on the WLM queue until being dispatched to a servant (region), and
- Time during which an application component, running in the servant, processes the request and generates a response.

The server generates a failure response if this processing does not complete within the specified time.

Note: This variable setting does not apply for HTTP requests or Scalable Messaging Support; for that type of work, the value specified through the *ConnectionResponseTimeout* variable controls the time allowed for dispatching work to a servant (region).

Default: 300 seconds

How to specify: Specify this setting in the administrative console using the path **Application Servers** > *server* > **ORB Service > Advanced Settings**.

Internal variable name (for debugging purposes): Look for the internal variable name *control_region_wlm_dispatch_timeout* in the was.env file or the JES job log.

Example: `WLM timeout=600`

Request timeout

Specifies the maximum time, in tenths of seconds, that the client will wait for the response to a client request. This variable is the only timeout available for remote method dispatches made by clients only, not by application components within the servant region. Because the sysplex TCP/IP that runs through the coupling facility does not always tell the client when the other end of the socket is gone, clients can wait indefinitely for a response unless you set this variable. Setting *Request timeout* ensures that the client gets a response within the specified time, even if the response is a `COMM_FAILURE` exception.

Default: 0 (unlimited), which means no time-out value is set

How to specify: Specify this custom property in the administrative console using the path **Application Servers** > *server* > **ORB Service**

Internal variable name (for debugging purposes): Look for the internal variable name *protocol_iiop_local_timeout* in the was.env file or the JES job log.

Example: `Request timeout=20` sets the time to 2 seconds

Transaction service timeout variables

Total Transaction Lifetime Timeout

Specifies the maximum amount of time, in seconds, that the J2EE server will wait for an application transaction that originated in this server to complete. This default amount of time is given to the application transaction if it does

not set its own timeout value through the `UserTransaction.setTransactionTimeout()` method. If the application transaction is not committed or rolled back within the specified time, the application transaction will be marked for roll-back and will be allowed to continue running for a grace period of about four minutes. If the application transaction is committed or rolled back during the grace period, the outcome of the transaction will always be rolled back. If the application transaction does not complete after the grace period, the controller abnormally ends the servant (region) in which the application component is running with ABEND EC3 RSN=04130002 or 04130005.

Note: Only the total transaction lifetime timeout and the maximum transaction timeout have grace periods.

The maximum value is 2147483 seconds (24.85 days). Set this value to zero to disable the function.

Default: 120 seconds

How to specify: Specify this setting in the administrative console using the path **Application Servers** > *server* > **Transaction Service**

Internal variable name (for debugging purposes): Look for the internal variable name *transaction_defaultTimeout* in the was.env file or the JES job log.

Maximum Transaction Timeout

Specifies the maximum amount of time, in seconds, that your installation will allow an application to specify for its transactions to complete. If an application assigns a greater amount of time through the `UserTransaction.setTransactionTimeout()` method, the J2EE server overrides the application setting to the value specified for the *Maximum Transaction Timeout* variable. If the application does not set its own time-out value through the `current->set_timeout` method, the default value set through the *Maximum Transaction Timeout* variable applies.

The maximum value is 2147483 seconds (24.85 days). Set this value to zero to disable the function.

Default: 300 seconds

How to specify: Specify this setting in the administrative console using the path **Application Servers** > *server* > **Transaction Service**

Internal variable name (for debugging purposes): Look for the internal variable name *transaction_maximumTimeout* in the was.env file or the JES job log.

transaction_recoveryTimeout

Specifies the time, in minutes, that this controller (region) uses to attempt to resolve in-doubt transactions before issuing a write-to-operator-with-reply (WTOR) message to the console, requesting whether it should:

- Stop trying to resolve in-doubt transactions,
- Write transaction-related information to the job log or hard copy log and terminate.

If the operator replies that recovery should continue, the controller (region) will attempt recovery for the specified amount of time before re-issuing the WTOR message. Once all the transactions are resolved, the controller region terminates. This variable applies only to controllers in peer restart and recovery mode.

Default: 15 minutes

How to specify: Specify this custom property in the administrative console using the path **Application Servers** > *server* > **Custom Properties**

Internal variable name (for debugging purposes): The internal variable name is the same as the external variable name. Look for *transaction_recoveryTimeout* in the was.env file or the JES job log.

Example: `transaction_recoveryTimeout=7`

Server custom properties

control_region_mdb_request_timeout
Specifies the time, in seconds, that the server will wait for a message driven bean (MDB) request to receive a response. If the response is not received within the specified amount of time, the servant (region) may abend with ABEND EC3 RSN=04130008. Set this value to zero to disable the function.

Default: 120

How to specify: Specify this custom property in the administrative console using the path **Application Servers** > *server* > **Custom Properties**.

Internal variable name (for debugging purposes): The internal variable name is the same as the external variable name. Look for *control_region_mdb_request_timeout* in the was.env file or the JES job log.

Example: `control_region_mdb_request_timeout=180`

protocol_http_timeout_output_recovery
Controls the recovery action taken on timeouts for requests received over the HTTP transport. Specifying ″SERVANT″ allows for the termination of servant (region) when timeouts occur. If an HTTP request is under dispatch in a servant (region) when its timeout value is reached, the servant (region) terminates with an ABEND EC3 RSN=04130007. The HTTP request and socket are then cleaned up. A setting of ″SESSION″ only cleans up the HTTP request and socket. No attempt is made to disrupt the execution of a dispatched HTTP request within a servant (region). Be careful using this setting as it may lead to a loss of resources if the dispatched HTTP request loops or hangs.

Default: SERVANT

How to specify: Specify this custom property in the administrative console using the path **Application Servers** > *server* > **Custom Properties**.

Internal variable name (for debugging purposes): The internal variable name is the same as the external variable name. Look for *protocol_http_timeout_output_recovery* in the was.env file or the JES job log.

Example: `protocol_http_timeout_output_recovery=SERVANT`

protocol_https_timeout_output_recovery
Controls the recovery action taken on timeouts for requests received over the HTTP SSL transport. Specifying ″SERVANT″ allows for the termination of servant (region) when timeouts occur. If an HTTP SSL request is under dispatch in a servant (region) when its timeout value is reached, the servant (region) terminates with an ABEND EC3 RSN=04130007. The HTTP SSL request and socket are then cleaned up. A setting of ″SESSION″ only cleans up the HTTP SSL request and socket. No attempt is made to disrupt the execution of a dispatched HTTP SSL request within a servant (region). Be careful using this setting as it may lead to a loss of resources if the dispatched HTTP SSL request loops or hangs.

Default: SERVANT

How to specify: Specify this custom property in the administrative console using the path **Application Servers** > *server* > **Custom Properties**

Internal variable name (for debugging purposes): The internal variable name is the same as the external variable name. Look for *protocol_https_timeout_output_recovery* in the was.env file or the JES job log.

Example: `protocol_https_timeout_output_recovery=SESSION`

Secure Sockets Layer configuration repertoires

V3 Timeout
Specifies the length of time, in seconds, that a browser can reuse a System SSL Version 3 session ID without renegotiating encryption keys with the server. The repertoires that you define for a server require the same V3 timeout value.

Default: 100

How to specify: Specify this custom property in the administrative console using the path **Security > SSL Application Servers > New SSL Repertoire**

Internal variable name (for debugging purposes): The following SSL Configuration Repertoire timeout variables are set internally when you define your SSL repertoires:

- com_ibm_CSI_perform_ssl_sys_v3_timeout
- com_ibm_CSI_claim_ssl_sys_v3_timeout
- com_ibm_HTTP_claim_ssl_sys_v3_timeout
- com_ibm_DAEMON_claim_ssl_sys_v3_timeout

Look for these internal variables in the was.env file or the JES job log.

TCP transport channel timeout properties

Inactivity timeout property
Specifies the amount of time, in seconds, that the TCP transport channel waits for a read or write request to complete on a socket.

Note: The value specified for this property might be overridden by the wait times established for channels above this channel. For example, the wait time established for an HTTP transport channel overrides the value specified for the is property for every operation except the initial read on a new socket.

Default: 0 seconds

How to specify: Specify this custom property in the administrative console using the path **Servers > Application Servers >***server* **> Web Container Transport Chains > TCP Inbound Channel**.

HTTP transport channel timeout properties

Read timeout property
Specifies the amount of time, in seconds, the HTTP transport channel waits for a read request to complete on a socket after the first read request occurs. The read being waited for could be an HTTP body (such as a POST) or part of the headers if they were not all read as part of the first read request on the socket.

Default: 60 seconds

How to specify: Specify this custom property in the administrative console using the path **Servers > Application Servers >***server* **> Web Container Transport Chains > HTTP Inbound Channel**.

Write timeout property
Specifies the amount of time, in seconds, the HTTP transport channel waits for a write request to complete on a socket. Specifies the amount of time, in seconds, that the HTTP transport channel waits on a socket for each portion of response data to be transmitted. This timeout usually only occurs in situations where the writes are lagging behind new requests. This can occur when a client has a low data rate or the server's NIC is saturated with I/O.

Default: 60 seconds

How to specify: Specify this custom property in the administrative console using the path **Servers > Application Servers >***server* **> Web Container Transport Chains > HTTP Inbound Channel**.

Persistent timeout property
Specifies the amount of time, in seconds, that the HTTP transport channel allows a socket to remain idle between requests.

Default: 30 seconds

How to specify: Specify this custom property in the administrative console using the path **Servers > Application Servers >***server* **> Web Container Transport Chains > HTTP Inbound Channel**.

HTTP Transport timeout variables

ConnectionIOTimeOut
Sets a maximum amount of time, in seconds, that the J2EE server will wait for the complete HTTP request to arrive. Set this variable for each of the HTTP transport definitions on the server. You will need to set this variable for both SSL transport and non-SSL transport. The J2EE server starts the timer after the connection has been established, and cancels the connection if a complete request does not arrive within the specified maximum time limit. Specifying a value of zero disables the time out function.

Default: 10 seconds

How to specify: Specify this custom property in the administrative console using the path **Application Servers** > *server* > **Web Container > HTTP Transport>***host***>Custom Properties**.

Note: This panel is only available if an HTTP transport is defined for your Application Server environment. If an HTTP transport is not defined for your environment, you can use the wsadmin scripting tool to define one. However, it is recommended that you use an HTTP transport channel instead of an HTTP transport whenever possible.

Internal variable name (for debugging purposes): If you are debugging a problem in SSL-enabled transport, look for the internal variable name *protocol_https_timeout_input* in the was.env file or the JES job log. If you are debugging a problem in non-SSL transport, look for the internal variable name *protocol_http_timeout_input* in the was.env file or the JES job log.

ConnectionResponseTimeout
Sets a maximum amount of time, in seconds, that the J2EE server will wait for an application component to respond to an HTTP request. Set this variable for each of the HTTP transport definitions on the server. You will need to set this

variable for both SSL transport and non-SSL transport. If the response is not received within the specified length of time, the servant (region) might fail with ABEND EC3 and RSN=04130007. Setting this timer prevents client applications from waiting for a response from an application component that might be deadlocked, looping, or encountering other processing problems that cause the application component to hang.

Default: 120 seconds

How to specify: Specify this custom property in the administrative console using the path **Application Servers** > *server* > **Web Container > HTTP Transport >***host***> Custom Properties**.

Internal variable name (for debugging purposes): If you are debugging a problem in SSL-enabled transport, look for the internal variable name *protocol_https_timeout_output* in the was.env file or the JES job log. If you are debugging a problem in non-SSL transport, look for the internal variable name *protocol_http_timeout_output* in the was.env file or the JES job log.

ConnectionKeepAliveTimeout

Specifies the time, in seconds, that the J2EE server will wait for a subsequent request from an HTTP client on a persistent connection. If another request is not received from the same client within this time limit, the connection is closed.

Default: 30 seconds

How to specify: Specify this custom property in the administrative console using the path **Application Servers** > *server* > **Web Container > HTTP Transport >***host***> Custom Properties**.

Internal variable name (for debugging purposes): If you are debugging a problem in SSL-enabled transport, look for the internal variable name *protocol_https_timeout_persistentSession* in the was.env file or the JES job log. If you are debugging a problem in non-SSL transport, look for the internal variable name *protocol_http_timeout_persistentSession* in the was.env file or the JES job log.

Related reference

"Setting dump controls" on page 214

IBM Support Assistant

The IBM Support Assistant is a tool that helps you use various IBM Support resources from within the WebSphere Application Server administrative console. The IBM Support Assistant offers three components to help you with software questions:

- a Search component, which helps you access pertinent Support information in multiple locations
- a Support Links component, which provides a convenient location to access various IBM Web resources such as IBM product sites, IBM support sites and links to IBM news groups
- a Service component, which helps you submit an enhanced service request that includes key system data to IBM.

To download the IBM Support Assistant:

1. Log into the administrative console and select **Support** from the top menu. This will open the product support page, from where you can download the IBM

Support Assistant and also access various IBM Web resources such as IBM product sites, IBM support sites and links to IBM news groups.

2. Select "Click here to download the IBM Support Assistant" to download the tool and follow the instructions in the README file to install the tool.

Once the IBM Support Assistant is installed in the administrative console, you can launch it by selecting 'Support' from the top menu in the administrative console. The search component of the IBM Support Assistant will be displayed in a new browser window.

To learn more about how to use the IBM Support Assistant, click on the User Guide tab in the IBM Support Assistant window.

Note: The Support link in the administrative console is a context sensitive link. When the IBM Support Assistant is not installed, the Support link displays a link to the IBM Support Assistant download page and various IBM Web resources for the WebSphere Application Server product. Once the IBM Support Assistant is installed, the Support link launches the IBM Support Assistant main page.

Preparing for a call to IBM service

When you report a problem to IBM service, you will need to provide as much information as possible to help service personnel quickly resolve the problem. The information you might need to send depends, in part, on the type of problem you have encountered, and includes the following items:

- Job logs for affected address spaces; for example, the controller and any servant regions that the controller terminated
- Job output for affected address spaces, particularly WebSphere Application Server for z/OS messages that are written to the JESMSGLG data set
- The system log (SYSLOG), another source of WebSphere Application Server for z/OS messages
- WebSphere Application Server for z/OS error log
- The system logrec data set or log stream
- CTRACE external writer data sets
- SVC dumps, CEEDUMPs, or other types of dumps produced because of the problem.
- The affected server's environment file, WAS.env, which is located in the HFS:

  `AppServer/config/cells/cellname/nodes/nodename/servers/servername/was.env`

Additionally, IBM service might request you to:

- Provide a description of the circumstances or scenario under which the error occurs.
- Use the VERBEXIT CBDATA subcommand.
- Reset WebSphere variables that are for use only when directed by IBM service.
- Set WebSphere variable values for the location service daemon address space (same as those for servers, with the prefix "DAEMON_").

Using the IPCS VERBEXIT subcommand to display diagnostic data

The interactive problem control system (IPCS) is a tool that provides formatting and analysis support for dumps and traces produced by WebSphere Application Server for z/OS and the applications that it hosts. IBM service personnel might request that you use the IPCS subcommand `VERBEXIT` with the`CBDATA` verb name to

display dump information for WebSphere Application Server for z/OS. The CBDATA formatters reside in the SBBOMIG data set, which must be in the link list or LPA.

Entering VERBEXIT CBDATA results in a display of dump contents that can include the following WebSphere Application Server for z/OS structures:
- Global control blocks
- Address space control blocks
- Task control blocks (TCBs)
- ORB control block

Optional parameters control which of these structures are included in the dump display. If you enter VERBEXIT CBDATA without any optional parameters, the dump display includes only global control block contents.

To enter VERBEXIT CBDATA, you may use any of the methods for entering IPCS subcommands on z/OS, as described in z/OS MVS IPCS User's Guide, SA22-7596. Use the following syntax: VERBEXIT CBDATA ['parameter [,parameter]...']

Valid parameters are:
- **GLOBAL(bgvt-address)**

  Formats and displays cell-level global vector data for the specified address space. This display includes the following formatted control blocks:
  - BGVT address - z/OS Global Vector table
  - ASR Table and ASR Table entries - Active Server Resposity information
- **ASID(asid-number)**

  Formats and displays address space information for the daemon, the controller (region), or the servant (region). This display includes the following formatted control blocks:
  - BACB - z/OS address space control block
  - BTRC,TBUFSET,TBUF - z/OS Component trace control blocks
  - BOAM,BOAMX - z/OS BOA control blocks
  - ACRW queue - Application Controller Work element control blocks
  - BTCB queues - z/OS control information

  Along with ASID(asid-number), IBM service personnel might direct you to specify one of the following parameters, to include additional information in the dump display:
  - **BTCB(btcb_address)**

    Formats and displays the specified BTCB and ORB information for the WebSphere Application Server for z/OS TCB.
  - **COMMDATA**

    Formats and displays session information.
  - **CONFIG**

    Formats and displays configuration information for the address space.
  - **OBJADDR(object_address)** and **OBJTYPE(object_type_ID)**

    Formats and displays information for the specified object of the specified type. IBM service personnel will provide the values for you to supply for these parameters.
  - **ORBDATA**

    Formats and displays ORB information.
  - **TCB(tcb_address)**

    Formats and displays request summary information for the specified task.
  - **TRACEBACK**

Formats and displays ORB information.

- **SUMMARY**

  Summarizes information from some of the other CBDATA optional parameters. For example, for the GLOBAL parameter, specifying SUMMARY produces a list of active servers.

Example: Output from the command `ip VERBEXIT CBDATA 'ASID( xx ) TCB(yyyyyyyy )'`:

```
command ===> ip VERBX CBDATA 'ASID(xx) TCB(xxxxxxxx)'
******************************** TOP OF DATA ********************************
COMPON=WEBSPHERE Z/OS,COMPID=5655A9801,ISSUER=BBORMCDP,ERRNO=04006006
_____________________________________________________________________________
BBOR0012I Formatting Clssname
 Clssname: 2BE6947E
+0000 D9859496 A385E685 82C39695 A3818995 |RemoteWebContain|.................|
+0010 859900                              |er.             |...              |
BBOR0012I Formatting MethodNm
 MethodNm: 2BE69472
+0000 88A3A397 998598A4 85A2A300 00000000 |httprequest.....|.........?......|
BBOR0012I Formatting ComRtInf
 ComRtInf: 2BE69212
+0000 89974081 8484997E F94BF5F6 4BF4F24B |ip addr=9.56.42.|..@....~.K..K..K|
+0010 F1F6F840 979699A3 7EF1F0F8 F500     |168 port=1085.  |...@....~.....   |
BBOR0026I GMT Time Request was received into CTL region
 TODCLOCK: 00000000
    04/08/2003 12:58:02.926136
BBOR0026I GMT Time Request was Queued to WLM in CTL region
 TODCLOCK: 00000000
    04/08/2003 12:58:02.926263
BBOR0026I GMT Time Request will be Expired (approximated)
 TODCLOCK: 00000000
    04/08/2003 13:08:01.663032
BBOR0026I GMT Time Request was received into SR region
 TODCLOCK: 00000000
    04/08/2003 12:58:02.927729
```

Setting trace controls for IBM service

To view or set your trace settings, use the WebSphere administrative console:

- Click **Environment > Manage WebSphere Variables**.
- On the Configuration Tab check for any of these variables in the name field and observe the variable setting in the value field.
- To change or set a variable, specify the variable in the name field and specify the setting in the value field. You can also describe the setting in the description field on this tab.

Note:

- If you use any level of tracing, including ras_trace_defaultTracingLevel=1, ensure that you set ras_trace_outputLocation to BUFFER.

  ras_trace_defaultTracingLevel=1 will write exceptions to the trace log as well as to the ERROR log.
- Set the ras_trace_BufferCount=4 and ras_trace_BufferSize=128.

  This will get 512KB of storage for the trace buffers (the minimum allowed) and reduce memory requirements.
- It is best to trace to CTRACE.

If you are tracing to sysprint with ras_trace_defaultTracingLevel=3, you may experience an almost 100% throughput degradation. If you are tracing to CTRACE, however, you may only experience a 15% degradation in throughput.

- Make sure you disable JRAS tracing.

  To do this, look for the following lines in the trace.dat file pointed to by the JVM properties file:

  ```
  com.ibm.ejs.*=all=disable

  com.ibm.ws390.orb.*=all=disable
  ```

  Ensure that both lines are set to **=disable** or delete the two lines altogether.

Note: If ras_trace_outputLocation is set, you may be tracing and not know it.

ras_trace_defaultTracingLevel= *n*

Specifies the default tracing level for WebSphere Application Server for z/OS.

Valid values and their meanings are:

0	No tracing
1	Exception tracing
2	Basic and exception tracing
3	Detailed tracing, including basic and exception tracing

Use this variable together with the ras_trace_basic and ras_trace_detail variables to set tracing levels for WebSphere Application Server for z/OS subcomponents.Specifies the default tracing level for WebSphere Application Server for z/OS.

Default: 1

Example:

```
ras_trace_defaultTracingLevel=2
```

ras_trace_basic=n | (n,...)

Specifies tracing overrides for particular WebSphere Application Server for z/OS subcomponents.

Subcomponents, specified by numbers, receive basic and exception traces. If IBM service directs you to specify more than one subcomponent, use parentheses and separate the numbers with commas. IBM service provides the subcomponent numbers and their meanings.

Other parts of WebSphere Application Server for z/OS receive tracing as specified on the ras_trace_defaultTracingLevel variable.

Note: Valid values for **ras_trace_basic** are:

- 0: RAS
- 1: Common Utilities
- 2: COS/Naming
- 3: COMM
- 4: ORB
- 5: IM
- 6: OTS
- 7: Shasta
- 8: Systems Management

- 9: OS/390 Wrappers
- A: Daemon
- B: IR
- C: Test
- D: COS/Query
- E: Security
- F: Externalization
- G: Adapter
- H: Lifecycle
- I: Identity
- J: JRAS (internal tracing--via direction from IBM support)
- K: Reference collections
- L: J2EE

Default: (no default value)

Example:

```
ras_trace_basic=3
```

ras_trace_detail=n | (n,...)

Specifies tracing overrides for particular WebSphere Application Server for z/OS subcomponents.

Subcomponents, specified by numbers, receive detailed traces. If IBM service directs you to specify more than one subcomponent, use parentheses and separate the numbers with commas. IBM service provides the subcomponent numbers and their meanings.

Other parts of WebSphere Application Server for z/OS receive tracing as specified on the ras_trace_defaultTracingLevel variable.

Note: Valid values for **ras_trace_detail** are:

- 0: RAS
- 1: Common Utilities
- 2: COS/Naming
- 3: COMM
- 4: ORB
- 5: IM
- 6: OTS
- 7: Shasta
- 8: Systems Management
- 9: OS/390 Wrappers
- A: Daemon
- B: IR
- C: Test
- D: COS/Query
- E: Security
- F: Externalization
- G: Adapter
- H: Lifecycle
- I: Identity
- J: JRAS (internal tracing--via direction from IBM support)
- K: Reference collections
- L: J2EE

Default: (no default value)

Examples:

```
ras_trace_detail=3
ras_trace_detail=(3,4)
```

ras_trace_specific=n | (n,...)

Specifies tracing overrides for specific WebSphere Application Server for z/OS trace points.

Trace points are specified by 8-digit, hexadecimal numbers. If IBM service directs you to specify more than one trace point, use parentheses and separate the numbers with commas. You also can specify a WebSphere variable name by enclosing the name in single quotes. The value of the WebSphere variable will be handled as if you had specified that value on ras_trace_specific.

Default: (no default value)

Examples:

```
ras_trace_specific=03004020
ras_trace_specific=(03004020,04005010)
ras_trace_specific='xyz'
```

[where xyz is an environment variable name]

```
ras_trace_specific=('xyz','abc',03004021)
```

[where xyz and abc are environment variable names]

ras_trace_exclude_specific=n | (n,...)

Specifies WebSphere Application Server for z/OS trace points to exclude from tracing activity.

Trace points are specified by 8-digit, hexadecimal numbers. If IBM service directs you to specify more than one trace point, use parentheses and separate the numbers with commas. You also can specify a WebSphere variable name by enclosing the name in single quotes. The value of the WebSphere variable will be handled as if you had specified that value on ras_trace_exclude_specific.

Default: (no default value)

Examples:

```
ras_trace_exclude_specific=03004020
ras_trace_exclude_specific=(03004020,04005010)
ras_trace_exclude_specific='xyz'
```

[where xyz is an environment variable name]

```
ras_trace_exclude_specific=('xyz','abc',03004021)
```

[where xyz and abc are environment variable names]

Setting dump controls for IBM service

ras_minorcode_action= *value*

Determines the default behavior for gathering documentation about system exception minor codes.

CEEDUMP

Captures callback and offsets.

Tip: It takes time for the system to take CEEDUMPs and this may cause transaction timeouts. For instance, if the WebSphere variable transaction_defaultTimeout is set to 30 seconds, your application

transaction may time out because processing a CEEDUMP can take longer than 30 seconds. To prevent this from happening, either:
- Increase the transaction timeout value, or
- Code ras_minorcode_action=NODIAGNOSTICDATA and make sure the ras_trace_minorCodeTraceBacks variable is not specified.

TRACEBACK
Captures Language Environment and z/OS UNIX traceback data.

SVCDUMP
Captures an MVS dump (but will not produce a dump in the client).

NODIAGNOSTICDATA
Specifies that no diagnostic data will be collected, even if CEEDUMP, TRACEBACK, or SVCDUMP processing occurs because of another WebSphere variable setting. For example, if you code both of the following variables, traceback processing occurs but none of the traceback data is collected: ras_minorcode_action=NODIAGNOSTICDATA and ras_trace_minorCodeTraceBacks=ALL

Default: NODIAGNOSTICDATA

Example:

```
ras_minorcode_action=SVCDUMP
```

ras_trace_minorCodeTraceBacks= *value*
Enables traceback of system exception minor codes. Values are:

```
ALL|all
```

Enables traceback for all system exception minor codes.

Enables traceback of system exception minor codes. Values are:
- *minor_code* Enables traceback for a specific minor code.

  Example: Type

  ```
  1234
  ```

  for minor code

  ```
  C9C21234
  ```
- *(null value)* The default. This setting will not cause gathering of a traceback.

  Default: (null value)

  Example:

  ```
  ras_trace_minorCodeTraceBacks=all
  ```

Obtaining help from IBM

If you are not able to resolve a WebSphere Application Server problem by following the steps described in the Troubleshooting guide, by looking up error messages in the message reference, or looking for related documentation on the online help, contact IBM Technical Support.

Purchase of WebSphere Application Server entitles you to one year of telephone support under the Passport Advantage program. For details on the Passport Advantage program, visit http://www.lotus.com/services/passport.nsf/WebDocs/Passport_Advantage_Home.

The number for Passport Advantage members to call for WebSphere Application Server support is 1-800-237-5511. Please have the following information available when you call:
- Your Contract or Passport Advantage number.

- Your WebSphere Application Server version and revision level, plus any installed fixes.
- Your operating system name and version.
- Your database type and version.
- Basic topology data: how many machines are running how many application servers, and so on.
- Any error or warning messages related to your problem.

IBM Support has documents that can save you time gathering information needed to resolve this problem. Before opening a PMR, see the IBM Support page.

Tracing

WebSphere Application Server support engineers might ask you to enable tracing on a particular component of the product to diagnose a difficult problem.

Consulting

For complex issues such as high availability and integration with legacy systems, education, and help in getting started quickly with the WebSphere product family, consider using IBM consulting services. To learn about these services, browse the Web site http://www-1.ibm.com/services/fullservice.html.

Diagnosing and fixing problems: Resources for learning

In addition to the information center, there are several Web-based resources for researching and resolving problems related to the WebSphere Application Server.

The WebSphere Application Server support page

The official site for providing tools and sharing knowledge about WebSphere Application Server problems is the WebSphere Application Server support page: http://www.ibm.com/software/webservers/appserv/support.html. Among the features it provides are:

- A search field for searching the entire support site for documentation and fixes related to a specific exception, error message, or other problem. Use this search function before contacting IBM Support directly.
- *Hints and Tips*, *Technotes*, and *Solutions* links take you to specific problems and resolutions documented by WebSphere Application Server technical support personnel.
- A link *All fixes, fix packs, refresh packs, and tools* provides free WebSphere Application Server maintenance upgrades and problem determination tools.
  - *fixes* are software patches which address specific WebSphere Application Server defects. Selecting a specific defect from the list in the *All fixes, fix packs, refresh packs, and tools* page takes you to a description of what problem the fix addresses.
  - Fix packs are bundles of multiple fixes, tested together and released as a maintenance upgrade to WebSphere Application Server. Refresh packs are fix packs that also contain new function. If you select a fix pack from this page, you are taken to a page describing the target platform, WebSphere Application Server prerequisite level, and other related information. Selecting the *list defects* link on that page displays a list of the fixes which the fix pack includes. If you intend to install a fix which is part of a fix pack, it is usually better to upgrade to the complete fix pack rather than to just install the individual fix.

– Tools are free programs that help you analyze the configuration, behavior and performance of your WebSphere Application Server installation.

Accessing WebSphere Application Server support page resources

Some resources on the WebSphere Application Server support page are marked with a key icon. To access these resources, you must supply a user ID and password, or to register if do not already have an ID. When registering, you are asked for your contract number, which is supplied as part of a WebSphere Application Server purchase.

WebSphere Developer Domain

The Developer Domains are IBM-supported sites for enabling developers to learn about IBM software products and how to use them. They contain resources such as articles, tutorials, and links to newsgroups and user groups. You can reach the WebSphere Developer Domain at http://www7b.software.ibm.com/wsdd/.

The IBM Support page

IBM Support has documents that can save you time gathering information needed to resolve this problem. Before opening a PMR, see the IBM Support page.

Debugging Service details

Use this page to view and modify the settings used by the Debugging Service.

To view this administrative console page, click **Servers > Application Servers** > *server name* > **Debugging Service**.

The steps below describe how to enable a debug session on WebSphere Application Server. Debugging can prove useful when your program behaves differently on the application server than on your local system.

Enable service at server startup

Specifies whether the server will attempt to start the Debug service when the server starts.

JVM debug port

Specifies the port that the Java virtual machine will listen on for debug connections.

JVM debug arguments

Specifies the debugging argument string used to start the JVM in debug mode.

Debug class filters

Specifies an array of classes to ignore during debugging. When running in step-by-step mode, the debugger will not stop in classes that match a filter entry.

BSF debug port

Specifies the port that the BSF Debug Manager listens on.

BSF logging level

Specifies the level of logging provided by the BSF Debug Manager. The valid range is 0-3, with 3 being the highest level of logging.

Configuration problem settings

Use this page to identify and view problems that exist in the current configuration.

To view this administrative console page, click **Troubleshooting > Configuration Problems** in the console navigation tree.

Click a configuration problem in the Configuration Problems table to see more information about the problem.

Configuration document validation

Use these fields to specify the level of validation to perform on configuration documents.

Enable Cross validation

Enables cross validation of configuration documents.

Enabling cross validation enables comparison of configuration documents for conflicting settings.

Configuration Problems

Displays current configuration problem error messages. Click a message for detailed information about the problem.

Scope

Sorts the configuration problem list by the configuration file where each error occurs. Click a message for detailed information about the problem.

Fields that explain each configuration problem:

The following informational fields appear when you click a configuration problem message.

Message

Displays the message returned from the validator.

Explanation

A brief explanation of the problem.

User action

Specifies the recommended action to correct the problem.

Target Object

Identifies the configuration object where the validation error occurred.

Severity

Indicates the severity of the configuration error, with 1 being a severe error. Severity decreases as the severity descriptor increases.

Local URI

Specifies the local URI of the configuration file where the error occurred.

Full URI

Specifies the full URI of the configuration file where the error occurred.

Validator classname

The classname of the validator reporting the problem.

Chapter 6. Troubleshooting by task

These topics provide troubleshooting information based on the task or activity you were doing when you encountered the problem.

- Troubleshooting installation problems
- Troubleshooting administration
- "Troubleshooting security configurations" on page 353
- Troubleshooting performance

Related reference

Troubleshooting installation problems

Resolving timeout conditions
This file gives an overview of how to resolve timeout conditions

"Debugging client exceptions" on page 16

"Debugging problems related to Java Message Service (JMS) support" on page 44

Installation component troubleshooting tips

If you are having problems installing your WebSphere Application Server product, follow these steps to resolve the problem:

- Browse the relevant log files for clues:
  - The main installation log file: *install_root*/logs/log.txt.
  - The Profile creation wizard log file: *install_root*/profiles/*profile_name*/logs/pctLog.txt when you create a profile with the Profile creation wizard.
  - The profile creation log for creating the profile: *install_root*/logs/wasprofile/wasprofile_create_*profile_name*.log.
  - IBM HTTP Server log: *ihs_install_root*/log.txt and *ihs_install_root*/ihsv6_install.log.
  - The log files produced when the default application .ear file is installed are: *install_root*/profiles/*profile_name*/logs/defaultapp_config.log and *install_root*/profiles/*profile_name*/logs/defaultapp_deploy.log.
  - Other logs for other applications for a profile in the *install_root*/profiles/*profile_name*/logs directory.
- Verify that you have installed the correct level of dependent software, such as operating system version and revision level, by reviewing http://www.ibm.com/software/webservers/appserv/doc/latest/prereq.html.

If none of these steps solves the problem, check to see if the problem is identified and documented using the links in "Diagnosing and fixing problems: Resources for learning" on page 229.

For current information available from IBM Support on known problems and their resolution, see the IBM Support page.

IBM Support has documents that can save you time gathering information needed to resolve this problem. Before opening a PMR, see the IBM Support page.

Related tasks

Chapter 6, "Troubleshooting by task"

Related reference

Chapter 7. Learn about WebSphere applications

Use this section as a starting point to investigate the technologies used in and by applications that you deploy on the application server.

See Learn about WebSphere applications: Overview and new features for an introduction to each technology.

Web applications	How do I?...	Overview		Samples
EJB applications	How do I?...	Overview	Tutorials	Samples
Client applications	How do I?...	Overview		Samples
Web services	How do I?...	Overview	Tutorials	Samples
Data access resources	How do I?...	Overview	Tutorials	Samples
Messaging resources	How do I?...	Overview	Tutorials	Samples
Mail, URLs, and other J2EE resources	How do I?...	Overview		
Security	How do I?...	Overview	Tutorials	Samples
Naming and directory	How do I?...	Overview		
Object Request Broker	How do I?...	Overview		
Transactions	How do I?...	Overview		Samples
ActivitySessions	How do I?...	Overview		Samples
Application profiling	How do I?...	Overview		Samples
Asynchronous beans	How do I?...	Overview		Samples
Dynamic caching	How do I?...	Overview		
Dynamic query	How do I?...	Overview		Samples
Internationalization	How do I?...	Overview		Samples
Object pools	How do I?...	Overview		
Scheduler	How do I?...	Overview		Samples
Startup beans	How do I?...	Overview		
Work areas	How do I?...	Overview		

Web applications

Web module or application server dies or hangs

If an application server dies (its process spontaneously closes), or freezes (its Web modules stop responding to new requests):

- Isolate the problem by installing Web modules on different servers, if possible.
- To detect memory leak problems, enable verbose garbage collection on the application server. This feature adds detailed statements to the JVM error log file of the application server about the amount of available and in-use memory. To set up verbose garbage collection:

1. Select **Servers > Application Servers > *server_name* > Process Definition > Java Virtual Machine**, and enable **Verbose Garbage Collection**.
2. Stop and restart the application server.
3. Periodically, or after the application server stops, browse the log file for garbage collection statements. Look for statements beginning with ″allocation failure″. The string indicates that a need for memory allocation has triggered a JVM garbage collection (freeing of unused memory). Allocation failures themselves are normal and not necessarily indicative of a problem. The allocation failure statement is followed by statements showing how many bytes are needed and how many are allocated.

   If there is a steady increase in the total amount of free and used memory (the JVM keeps allocating more memory for itself), or if the JVM becomes unable to allocate as much memory as it needs (indicated by the bytes needed statement), there might be a memory leak.

- If the verbose garbage collection output indicates that the application server is running out of memory, one of the following problems might be present:
- If an application server spontaneously dies, there will be an SDUMP. See Using the Error Dump and Cleanup interface for instructions on how to analyze the dump.

For current information available from IBM Support on known problems and their resolution, see the IBM Support page.

IBM Support has documents that can save you time gathering information needed to resolve this problem. Before opening a PMR, see the IBM Support page.

Related tasks

Chapter 6, “Troubleshooting by task,” on page 233

JavaServer Pages source code shown by the Web server

Problem

If you share the document root of the WebSphere Application Server with the Web server document root, a security exposure can result as the Web server might display the JavaServer Pages (JSP) source file as plain text.

You can use the WebSphere Web server plug-in set of rules to determine whether a given request will be handled by the WebSphere Application Server. When an incoming request fails to match those rules, the Web server plug-in returns control to the Web server so that the Web server can fulfill the request. In this case, the unknown host header causes the Web server plug-in to return control to the Web server because the rules do not indicate that the WebSphere Application Server should handle it. Therefore, the Web server looks for the request in the Web server document root. Since the JSP source file is stored in the document root of the Web server, the Web server finds the file and displays it as plain text.

Suggested solution

Move the WebSphere Application Server JSP source file outside of the Web server document root. Then, when this request comes in with the unknown host header, the plug-in returns control to the Web server and the JSP source file is not found in the document root. Therefore, the Web server returns a `404 File Not Found` error rather than the JSP source file.

JavaServer Pages engine troubleshooting tips

If you are having difficulty using the JavaServer Pages (JSP) engine, try these steps:

1. Determine whether other resources such as `.html` files or servlets are being requested and displayed correctly. If they are not, the problem probably lies at a deeper level, such as with the HTTP server.
2. If other resources are being displayed correctly, determine whether the JSP processor has started normally:
   - Browse the logs of the server hosting the JSP files you are trying to access. The following messages indicate that the JSP processor has started normally:

     ```
     Extension Processor [class com.ibm.ws.jsp.webcontainerext.JSPExtensionProcessor]
     was initialized successfully.
     Extension Processor [class com.ibm.ws.jsp.webcontainerext.JSPExtensionProcessor]
     has been associated with patterns [*.jsp *.jspx *.jsw *.jsv ].
     ```

     If the JSP processor fails to load, you will see a message such as

     ```
     No Extension Processor found for handling JSPs.
     JSP Processor not defined. Skipping : jspfilename.
     ```

     in the server log files.
3. If the JSP engine has started normally, the problem may be with the JSP file itself.
   - The JSP may have invalid JSP syntax and could not be processed by the JSP Processor. Examine the server log files of the target application for invalid JSP directive syntax messages. Errors similar to the following in a browser indicate this kind of problem:

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

     This example indicates that line 2, column 1 of the named JavaServer Pages file is missing a mandatory attribute for the jsp:include action. Similar messages are displayed for other syntax errors.
   - Examine the target application server's `SystemErr.log` files for problems with invalid Java syntax. Errors similar to **Message: Unable to compile class for JSP** in a browser indicate this kind of problem.

     The error message output from the Javac compiler will be found in the `SystemErr.log`. It might look like:

     ```
     JSPG0091E: An error occurred at line: 2 in the file: /myJsp.jsp
     JSPG0093E: Generated servlet error: c:\WASROOT\temp\ ...
     test.war\_myJsp.java:16: myInt is already defined in com.ibm.ws.jsp20._myJsp
     int myInt = 122; String myString = "number is 122"; static int myStaticInt=22;
     int myInt=121;
              ^  1 error
     ```

     Correct the error in the JSP file and retry the file.
   - Examine the target application server's server log files files for problems with invalid Java syntax. Errors similar to **Message: Unable to compile class for JSP** in a browser indicate this kind of problem.

     The error message output from the Javac compiler will be found in the SystemErr.log server log files. It might look like:

     ```
     JSPG0091E: An error occurred at line: 2 in the file: /myJsp.jsp
     JSPG0093E: Generated servlet error: c:\WASROOT\temp\ ...
     test.war\_myJsp.java:16: myInt is already defined in com.ibm.ws.jsp20._myJsp
     int myInt = 122; String myString = "number is 122"; static int myStaticInt=22;
     int myInt=121;
              ^  1 error
     ```

     Correct the error in the JSP file and retry the file.

If none of these steps solves the problem, check to see if the problem is identified and documented using the links in Diagnosing and fixing problems: Resources for learning. If you do not see a problem that resembles yours, or if the information provided does not solve your problem, contact IBM support for further assistance.

For current information available from IBM Support on known problems and their resolution, see the IBM Support page. The IBM Support page contains documents that can save you time gathering information needed to resolve this problem.

Related tasks

Chapter 6, "Troubleshooting by task," on page 233

Related reference

Troubleshooting installation problems

HTTP session manager troubleshooting tips

If you are having problems creating or using HTTP sessions with your Web application hosted by WebSphere Application Server, here are some steps to take:

- View the logs for the application server which hosts the problem application:
  - first, look at messages written while each application is starting. They will be written between the following two messages:

    ```
    Starting application: application
    ......................
    Application started: application
    ```
  - Within this block, look for any errors or exceptions containing a package name of com.ibm.ws.webcontainer.httpsession. If none are found, this is an indication that the session manager started successfully.
  - Error "**SRVE0054E: An error occurred while loading session context and Web application**" indicates that SessionManager didn't start properly for a given application.
  - Look within the logs for any Session Manager related messages. These messages will be in the format SESNxxxxE and SESNxxxxW for errors and warnings, respectively, where xxxx is a number identifying the precise error. Look up the extended error definitions in the Session Manager message table.
- See Best practices for using HTTP Sessions.
- Alternatively, a special servlet can be invoked that displays the current configuration and statistics related to session tracking.
  - Servlet name: **com.ibm.ws.webcontainer.httpsession.IBMTrackerDebug**.
  - It can be invoked from any web module which is enabled to serve by class name. For example, using default_app, **http://localhost:9080/servlet/com.ibm.ws.webcontainer.httpsession.IBMTrackerDebug**.
  - If you are viewing the module via the serve-by-class-name feature, be aware that it may be viewable by anyone who can view the application. You may wish to map a specific, secured URL to the servlet instead and disable the serve-servlets-by-classname feature.
- If you are using **database-based persistent sessions**, look for problems related to the **data source** the Session Manager relies on to keep session state information. For details on diagnosing database related problems see Errors accessing a datasource or connection pool

Error message SRVE0079E Servlet host not found after you define a port

Error message SRVE0079E can occur after you define the port in WebContainer > HTTP Transports for a server, indicating that you do not have the port defined in your virtual host definitions. To define the port,

1. On the administrative console, go to Environment > Virtual Hosts > default_host> Host Aliases> New
2. Define the new port on host "*"

The application server gets EC3 - 04130007 ABENDs

To prevent an EC3 - 04130007 abend from occuring on the application server, change the HTTP Output timeout value. The custom property *ConnectionResponseTimeout* specifies the maximum number of seconds the HTTP port for an individual server can wait when trying to read or write data. For instructions on how to set *ConnectionResponseTimeout*, see HTTP transport custom properties.

If none of these steps fixes your problem, check to see if the problem has been identified and documented by looking at the available online support (hints and tips, technotes, and fixes).

For current information available from IBM Support on known problems and their resolution, see the IBM Support page.

IBM Support has documents that can save you time gathering information needed to resolve this problem. Before opening a PMR, see the IBM Support page.

Related tasks

Chapter 6, "Troubleshooting by task," on page 233

"Troubleshooting by component" on page 5

Problems creating or using HTTP sessions

Note: To view and update the Session Manager settings discussed here, use the administrative console. Select the application server that hosts the problem application, then under **Additional properties**, select **Web Container**, then **Session manager**.

What kind of problem are you having?
- HTTP Sessions are not getting created, or are lost between requests.
- HTTP Sessions are not persistent (session data lost when application server restarts, or not shared across cluster).
- Session is shared across multiple browsers on same client machine.
- Session is not getting invalidated immediately after specified Session timeout interval.
- Unwanted sessions are being created by jsps.
- Session data intended for one client is seen by another client

If your problem is not described here, or none of these steps fixes the problem:

HTTP Sessions are not getting created, or are lost between requests

By default, the Session Manager uses cookies to store the session ID on the client between requests. Unless you intend to avoid cookie-based session tracking, ensure that cookies are flowing between WebSphere Application Server and the browser:
- Make sure the **Enable cookies** checkbox is checked under the **Session tracking Mechanism** property.
- Make sure cookies are enabled on the browser you are testing from or from which your users are accessing the application.

- Check the Cookie domain specified on the SessionManager (to view the or update the cookie settings, in the **Session tracking mechanism->enable cookies** property, click **Modify**).
  - For example, if the cookie domain is set as ".myCom.com", resources should be accessed using that domain name, e.g. http://www.myCom.com/myapp/servlet/sessionservlet.
  - If the domain property is set, make sure it begins with a dot (.). Certain versions of Netscape do not accept cookies if domain name doesn't start with a dot. Internet Explorer honors the domain with or without a dot. For example, if the domain name is set to *mycom.com*, change it to *.mycom.com* so that both Netscape and Internet Explorer honor the cookie.
- Check the **Cookie path** specified on the SessionManager. Check whether the problem url is hierarchially below the Cookie path specified. If not correct the Cookie path.
- If the `Cookie maximum age` property is set, ensure that the client (browser) machine's date and time is the same as the server's, including the time zone. If the client and the server time difference is over the "Cookie maximum age" then every access would be a new session, since the cookie will "expire" after the access.
- If you have multiple web modules within an enterprise application that track sessions:
  - If you want to have different session settings among web modules in an enterprise application, ensure that each web module specifies a different cookie name or path, or
  - If Web modules within an enterprise application use a common cookie name and path, ensure that the HTTP session settings, such as Cookie maximum age, are the same for all Web modules. Otherwise cookie behavior will be unpredictable, and will depend upon which application creates the session. Note that this does not affect session data, which is maintained separately by Web module.
- Check the cookie flow between browser and server:
  1. On the browser, enable "cookie prompt". Hit the servlet and make sure cookie is being prompted.
  2. Access the session servlet from the browser.
  3. The browser will prompt for the cookie; note the jsessionid.
  4. Reload the servlet, note down the cookie if a new cookie is sent.
  5. Check the session trace and look for the session id and trace the request by the thread. Verify that the session is stable across web requests:
     - Look for **getIHttpsession(...)** which is start of session request.
     - Look for **releaseSesson(..)** which is end of servlet request.
- If you are using URL rewriting instead of cookies:
  - Ensure there are no static HTML pages on your application's navigation path.
- If you are using SSL as your session tracking mechanism:
  - Ensure that you have SSL enabled on your IBM HTTP Server or iPlanet HTTP server.
- If you are in a clustered (multiple node) environment, ensure that you have session persistence enabled.

HTTP Sessions are not persistent

If your HTTP sessions are not persistent, that is session data is lost when the application server restarts or is not shared across the cluster:
- Check the Datasource.
- Check the SessionManager's Persistence Settings properties:
  - If you intend to take advantage of Session Persistence, verify that Persistence is set to **Database**.

– If you are using **Database-based persistence**:
  - Check the jndi name of the datasource specified correctly on SessionManager.
  - Specify correct userid and password for accessing the database.

    Note that these settings have to be checked against the properties of an existing Data Source in the admin console. The Session Manager does not automatically create a session database for you.
  - The Datasource should be non-JTA, i.e. non XA enabled.
  - Check the logs for appropriate database error messages.
  - With DB2, for row sizes other than 4k make sure specified row size matches the DB2 page size. Make sure tablespace name is specified correctly.

Session is shared across multiple browsers on same client machine

This behavior is browser-dependent. It varies between browser vendors, and also may change according to whether a browser is launched as a new process or as a subprocess of an existing browser session (for example by hitting Ctl-N on Windows).

The Cookie maximum age property of the Session Manager also affects this behavior, if cookies are used as the session-tracking mechanism. If the maximum age is set to some positive value, all browser instances share the cookies, which are persisted to file on the client for the specifed maximum age time.

Session is not getting invalidated immediately after specified Session timeout interval

The SessionManager invalidation process thread runs every *x* seconds to invalidate any invalid sessions, where x is determined based on the Session timeout interval specified in the Session manager properties. For the default value of 30 minutes , *x* is around 300 seconds. In this case, it could take up to 5 minutes (300 seconds) beyond the timeout threshold of 30 minutes for a particular session to become invalidated.

Unwanted sessions are being created by jsps

As required by the JavaServer Page specification, jsps by default perform a request.getSession(true), so that a session is created if none exists for the client. To prevent jsps from creating a new session, set the session scope to **false** in the jsp file using the page directive as follows:

```
<% @page session="false" %>
```

Session data intended for one client is seen by another client

In rare situations, usually due to application errors, session data intended for one client might be seen by another client. This situation is referred to as session data crossover. When the *DebugSessionCrossover* custom property is set to true, code is enabled to detect and log instances of session data crossover. Checks are performed to verify that only the session associated with the request is accessed or referenced. Messages are logged if any discrepancies are detected. These messages provide a starting point for debugging this problem. This additional checking is only performed when running on the WebSphere-managed dispatch thread, not on any user-created threads.

For additional information on how to set this property, see article, Web container custom properties.

For current information available from IBM Support on known problems and their resolution, see the IBM Support page.

IBM Support has documents that can save you time gathering information needed to resolve this problem. Before opening a PMR, see the IBM Support page.

Related tasks

Chapter 6, "Troubleshooting by task," on page 233

Troubleshooting tips for Web application deployment

Deployment of a Web application is successful if you can access the application by typing a Uniform Resource Locator (URL) in a browser, or if you can access the application by following a link.

If you cannot access your application, follow these steps to eliminate some common errors that can occur during migration or deployment.

Web module does not run in WebSphere Application Server Version 5 or 6.

Symptom	Your Web module does not run when you migrate it to Version 5 or 6
Problem	In Version 4.x, the classpath setting that affected visibility was *Module Visibility Mode*. In Versions 5 and 6, you must use class loader policies to set visibility.
Recommended response	Reassemble an existing module, or change the visibility settings in the class loader policies. See the articles Class loaders and Class loading for more information.

Welcome page is not visible.

Symptom	You cannot access an application with a Web path of: `/webapp/myapp`
Problem	The default welcome page for a Web application is assumed to be *index.html*. You cannot access the default page of the *myapp* application unless it is named *index.html*.
Recommended response	To identify a different welcome page, modify the properties of the Web module during assembly. See the article Assembling Web applications for more information.

HTML files are not found.

Symptom	Your Web application ran successfully on prior versions, but now you encounter errors that the welcome page (typically *index.html*), or referenced HTML files are not found:
	`Error 404: File not found: Banner.html`
	`Error 404: File not found: HomeContent.html`

Problem	For security and consistency reasons, Web application URLs are now case-sensitive on all operating systems.
	
	Suppose the content of the index page is as follows:
	`<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 5.0 Frameset//EN">`
	`<HTML>`
	`<TITLE>`
	`Insurance Home Page`
	`</TITLE>`
	`<frameset rows="18,80">`
	`<frame src="Banner.html" name="BannerFrame" SCROLLING=NO>`
	`<frame src="HomeContent.html" name="HomeContentFrame">`
	`</frameset>`
	`</HTML>`
	
	However the actual file names in the `\WebSphere\AppServer\installedApps\...` directory where the application is deployed are:
	`banner.html`
	`homecontent.html`

Recommended response	To correct this problem, modify the *index.html* file to change the names *Banner.html* and *HomeContent.html* to *banner.html* and *homecontent.html* to match the names of the files in the deployed application.

For current information available from IBM Support on known problems and their resolution, see the IBM Support page.

IBM Support has documents that can save you time gathering information needed to resolve this problem. Before opening a PMR, see the IBM Support page.

Related concepts

Class loaders
Class loaders find and load class files. Class loaders enable applications that are deployed on servers to access repositories of available classes and resources. Application developers and deployers must consider the location of class and resource files, and the class loaders used to access those files, to make the files available to deployed applications.

Related tasks

Chapter 5, “Diagnosing problems (using diagnosis tools),” on page 131

Securing applications and their environments

Deploying and administering applications
Deploying an application file consists of installing the application file on a server configured to hold installable modules.

Related reference

“JavaServer Pages engine troubleshooting tips” on page 7

Web applications: Resources for learning

EJB applications

Enterprise bean and EJB container troubleshooting tips

If you are having problems starting an EJB container, or encounter error messages or exceptions that appear to be generated on by an EJB container, follow these steps to resolve the problem:

- Browse the relevant log files for clues:
  - Use the Administrative Console to verify that the application server which hosts the container is running.
  - Browse the logs for the application server which hosts the container. Look for the message **server *server_name* open for e-business** in the server log files. If it does not appear, or if you see the message **problems occurred during startup**, browse the server log files for details.

If none of these steps solves the problem, check to see if the problem is identified and documented using the links in Diagnosing and fixing problems: Resources for learning.

For current information available from IBM Support on known problems and their resolution, see the IBM Support page.

IBM Support has documents that can save you time gathering information needed to resolve this problem. Before opening a PMR, see the IBM Support page.

Related tasks

Chapter 6, "Troubleshooting by task," on page 233

Related reference

Troubleshooting installation problems

Cannot access an enterprise bean from a servlet, a JSP file, a stand-alone program, or another client

What kind of error are you seeing?

- **javax.naming.NameNotFoundException: Name** *name* **not found in context** "**local**" **message** when access is attempted
- **BeanNotReentrantException** is thrown
- **CSITransactionRolledbackException / TransactionRolledbackException** is thrown
- Call fails, Stack trace beginning **EJSContainer E Bean method threw exception [exception_name]** found in JVM log file.
- Call fails, **ObjectNotFoundException or ObjectNotFoundLocalException** when accessing stateful session EJB found in JVM log file.
- Attempt to start CMP EJB module fails with **javax.naming.NameNotFoundException:** *dataSourceName*
-
- **Message BBOT0003W is issued**
- Symptom: **CNTR0001W: A Stateful SessionBean could not be passivated**

If the client is remote to the enterprise bean, which means, running in a different application server or as a stand-alone client, browse the logs of the application server hosting the enterprise bean as well as log files of the client.

ObjectNotFoundException or ObjectNotFoundLocalException when accessing stateful session EJB

A possible cause of this problem is that the stateful session bean timed out and was removed by the container. This event must be addressed in the code, according to the EJB 2.1 specification (available at `http://java.sun.com/products/ejb/docs.html`), section 7.6.2, Dealing with exceptions.

Stack trace beginning "EJSContainer E Bean method threw exception [exception_name]" found in JVM log file

If the exception name indicates an exception thrown by an IBM class that begins with ″com.ibm...″, then search for the exception name within the information center, and in the online help as described below. If ″exception name″ indicates an exception thrown by your application, contact the application developer to determine the cause.

javax.naming.NameNotFoundException: Name name not found in context "local"

A possible reason for this exception is that the enterprise bean is not local (not running in the same Java virtual machine [JVM] or application server) to the client JSP, servlet, Java application, or other enterprise bean, yet the call is to a ″local″ interface method of the enterprise bean . If access worked in a development environment but not when deployed to WebSphere Application Server, for example, it might be that the enterprise bean and its client were in the same JVM in development, but are in separate processes after deployment.

To resolve this problem, contact the developer of the enterprise bean and determine whether the client call is to a method in the local interface for the enterprise bean. If so, have the client code changed to call a remote interface method, or to promote the local method into the remote interface.

References to enterprise beans with local interfaces are bound in a name space local to the server process with the URL scheme of `local:`.

BeanNotReentrantException is thrown

This problem can occur because client code (typically a servlet or JSP file) is attempting to call the same stateful SessionBean from two different client threads. This situation often results when an application stores the reference to the stateful session bean in a static variable, uses a global (static) JSP variable to refer to the stateful SessionBean reference, or stores the stateful SessionBean reference in the HTTP session object. The application then has the client browser issue a new request to the servlet or JSP file before the previous request has completed.

To resolve this problem, ask the developer of the client code to review the code for these conditions.

CSITransactionRolledbackException / TransactionRolledbackException is thrown

An enterprise bean container throws these high-level exceptions to indicate that an enterprise bean call could not successfully complete. When this exception is thrown, logs to determine the underlying cause.

Some possible causes include:
- The enterprise bean might throw an exception that was not declared as part of its method signature. The container is required to roll back the transaction in this case. Common causes of this situation are where the enterprise bean or code that it calls throws a NullPointerException, ArrayIndexOutOfBoundsException, or other Java runtime exception, or where a BMP bean encounters a JDBC error. The resolution is to investigate the enterprise bean code and resolve the underlying exception, or to add the exception to the problem method signature.
- A transaction might attempt to do additional work after being placed in a `″Marked Rollback″`, `″RollingBack″`, or `″RolledBack″` state. Transactions cannot

continue to do work after they are set to one of these states. This situation occurs because the transaction has timed out which, often occurs because of a database deadlock. Work with the application database management tools or administrator to determine whether database transactions called by the enterprise bean are timing out.
- A transaction might fail on commit due to dangling work from local transactions. The local transaction encounters some ″dangling work″ during commit. When a local transactions encounters an ″unresolved action″ the default action is to ″rollback″. You can adjust this action to ″commit″ in an assembly tool. Open the enterprise bean `.jar` file (or the `EAR` file containing the enterprise bean) and select the Session Beans or Entity Beans object in the component tree on the left. The Unresolved Action property is on the IBM Extensions tab of the container properties.

Attempt to start EJB module fails with "javax.naming.NameNotFoundException dataSourceName_CMP"exception

This problem can occur because:
- When the DataSource resource was configured, container managed persistence was not selected.
  - To confirm this problem, in the administrative console, browse the properties of the data source given in the NameNotFoundException. On the Configuration panel, look for a check box labeled **Container Managed Persistence**.
  - To correct this problem, select the check box for **Container Managed Persistence**.
- If container managed persistence is selected, it is possible that the CMP DataSource could not be bound into the namespace.
  - Look for additional naming warnings or errors in the status bar, and in the hosting application server logs. Check any further naming-exception problems that you find by looking at the topic “Cannot look up an object hosted by WebSphere Application Server from a servlet, JSP file, or other client” on page 46.

Message BBOT0003W is issued

Message BBOT0003W indicates a transaction timeout. The timeout might result in the abnormal termination of the servant where the transaction is executing.
- The default timeout value for enterprise bean transactions is 120 seconds. After this time, the transaction times out and the connection closes.
- If the transaction legitimately takes longer than the specified timeout period, on the administrative console:
  1. Go to **Manage Application Servers > server_name**
  2. Select the **Transaction Service properties** page
  3. Increase the **Total transaction lifetime timeout** value
  4. **Save** the configuration

  Note: z/OS will use the value you set for **Total transaction lifetime timeout** as the default transaction timeout setting. If you set a value for this property that is greater than the maximum transaction timeout value, z/OS will use the maximum transaction timeout value as the default.

Symptom:CNTR0001W: A Stateful SessionBean could not be passivated

This error can occur when a Connection object used in the bean is not closed or nulled out.

To confirm this is the problem, look for an exception stack in the logs for the EJB container that hosts the enterprise bean, and looks similar to:

```
StatefulPassi W CNTR0001W:
A Stateful SessionBean could not be passivated: StatefulBeanO
(BeanId(XXX#YYY.jar#ZZZZ),
state = PASSIVATING)
java.io.NotSerializableException: com.ibm.ws.rsadapter.jdbc.WSJdbcConnection
 at java.io.ObjectOutputStream.outputObject((Compiled Code))
 at java.io.ObjectOutputStream.writeObject(ObjectOutputStream.java(Compiled Code))
 at java.io.ObjectOutputStream.outputClassFields((Compiled Code))
 at java.io.ObjectOutputStream.defaultWriteObject((Compiled Code))
 at java.io.ObjectOutputStream.outputObject((Compiled Code))
 at java.io.ObjectOutputStream.writeObject(ObjectOutputStream.java(Compiled Code))
 at com.ibm.ejs.container.passivator.StatefulPassivator.passivate((Compiled Code))

 at com.ibm.ejs.container.StatefulBeanO.passivate((Compiled Code)
 at com.ibm.ejs.container.activator.StatefulASActivationStrategy.atUnitOfWorkEnd
                     ((Compiled Code))
 at com.ibm.ejs.container.activator.Activator.unitOfWorkEnd((Compiled Code))
 at com.ibm.ejs.container.ContainerAS.afterCompletion((Compiled Code)
```

where *XXX,YYY,ZZZ* is the Bean's name.

To correct this problem, the application must close all connections and set the reference to null for all connections. Typically this activity is done in the `ejbPassivate()` method of the bean. See the enterprise bean specification mandating this requirement, specifically section 7.4 in the EJB specification Version 2.1. Also, note that the bean must have code to reacquire these connections when the bean is reactivated. Otherwise, there are NullPointerExceptions when the application tries to reuse the connections.

Related tasks

Chapter 6, "Troubleshooting by task," on page 233

Access intent exceptions

The following exceptions are thrown in response to the application of access intent policies:

com.ibm.ws.ejbpersistence.utilpm.PersistenceManagerException

If the method that drives the ejbLoad() method is configured to be read-only but updates are then made within the transaction that loaded the bean's state, an exception is thrown during invocation of the ejbStore() method, and the transaction is rolled back. Likewise, the ejbRemove() method cannot succeed in a transaction that is set as read-only. If an update hint is applied to methods of entity beans with bean-managed persistence, the same behavior and exception results. The forwarded exception object contains the message string `PMGR1103E: update instance level read only bean` *`beanName`*

This exception is also thrown if the applied access intent policy cannot be honored because a finder, ejbSelect, or container-managed relationship (CMR) accessor method returns an inherently read-only result. The forwarded exception object contains the message string `PMGR1001: No such DataAccessSpec -` *`methodName`*

The most common occurrence of this error is when a custom finder that contains a read-only EJB Query Language (EJB QL) statement is called with an applied access intent of `wsPessimisticUpdate` or `wsPessimisticUpdate-Exclusive`. These policies require the use of a USE AND KEEP UPDATE LOCKS clause on the SQL SELECT statement to be executed, but a

read-only query cannot support USE AND KEEP UPDATE LOCKS. Other examples of read-only queries include joins; the use of ORDER BY, GROUP BY, and DISTINCT keywords.

To eliminate the exception, edit the EJB query so that it does not return an inherently read-only result or change the access intent policy being applied.

- If an update access is required, change the applied access intent setting to `wsPessimisticUpdate-WeakestLockAtLoad` or `wsOptimisticUpdate`.
- If update access is not truly required, use `wsPessimisticRead` or `wsOptimisticRead`.
- If connection sharing between entity beans is required, use `wsPessimisticUpdate-WeakestLockAtLoad` or `wsPessimisticRead`.

com.ibm.websphere.ejb.container.CollectionCannotBeFurtherAccessed
If a lazy collection is driven after it is no longer in scope, and beyond what has already been locally buffered, a CollectionCannotBeFurtherAccessed exception is thrown.

com.ibm.ws.exception.RuntimeWarning
If an application is configured incorrectly, a run-time warning exception is thrown as the application starts; startup is ended. You can validate an application's configuration by choosing the verify function. Some examples of misconfiguration include:

- A method configured with two different access intent policies
- A method configured with an undefined access intent policy

Related tasks

Using access intent policies

Frequently asked questions: Access intent

I have not applied any access intent policies at all. My application runs just fine with a DB2 database, but it fails with an Oracle database with the following message: ***com.ibm.ws.ejbpersistence.utilpm.PersistenceManagerException: PMGR1001E: No such DataAccessSpec :FindAllCustomers. The backend datastore does not support the SQLStatement needed by this AccessIntent: (pessimistic update-weakestLockAtLoad)(collections: transaction/25) (resource manager prefetch: 0) (AccessIntentImpl@d23690a).*** **Why?**

If you have not configured access intent, all of your data is accessed under the default access intent policy (`wsPessimisticUpdate-WeakestLockAtLoad`). On DB2 databases, the weakest lock is a shared one, and the query runs without a USE AND KEEP UPDATE LOCKS clause. On Oracle databases, however, the weakest lock is an update lock; this means that the SQL query must contain a USE AND KEEP UPDATE LOCKS clause. However, not every SQL statement necessarily supports USE AND KEEP UPDATE LOCKS; for example, if the query is being run against multiple tables in a join, USE AND KEEP UPDATE LOCKS is not supported. To avoid this problem, try either of the following:

- Modify your SQL query or reconfigure your application so that an update lock is supported
- Apply an access intent policy that supports optimistic concurrency

I am calling a finder method and I get an InconsistentAccessIntentException at run time. Why?

This can occur when you use method-level access intent policies to apply more control over how a bean instance is loaded. This exception indicates that the entity bean was previously loaded in the same transaction. This could happen if you

called a multifinder method that returned the bean instance with access intent policy X applied; you are now trying to load the second bean again by calling its findByPrimaryKey method with access intent Y applied. Both methods must have the same access intent policy applied.

Likewise, if the entity was loaded once in the transaction using an access intent policy configured on a finder, you might have called a container-managed relationship (CMR) accessor method that returned the entity bean configured to load using that entity's default access intent.

To avoid this problem, ensure that your code does not load the same bean instance twice within the same transaction with different access intent policies applied. Avoid the use of method-level access intent unless absolutely necessary.

I have two beans in a container-managed relationship. I call findByPrimaryKey() on the first bean and then call getBean2( ), a CMR accessor method, on the returned instance. At that point, I get an InconsistentAccessIntentException. Why?

You are probably using read-ahead. When you loaded the first bean, you caused the second bean to be loaded under the access intent policy applied to the finder method for the first bean. However, you have configured your CMR accessor method from the first bean to the second with a different access intent policy. CMR accessor methods are really finder methods in disguise; the run-time environment behaves as if you were trying to change the access intent for an instance you have already read from persistent store.

To avoid this problem, beans configured in a read-ahead hint are all driven to load with the same access intent policy as the bean to which the read-ahead hint is applied.

I have a bean with a one-to-many relationship to a second bean. The first bean has a pessimistic-update intent policy applied. When I try to add an instance of the second bean to the first bean's collection, I get an UpdateCannotProceedWithIntegrityException. Why?

The second bean probably has a read intent policy applied. When you add the second bean to the first bean's collection, you are not updating the first bean's state, you are implicitly modifying the second bean's state. (The second bean contains a foreign key to the first bean, which is modified.)

To avoid this problem, ensure that both ends of the relationship have an update intent policy applied if you expect to change the relationship at run time.

Related concepts

Access intent policies
An access intent policy is a named set of properties (access intents) that governs data access for Enterprise JavaBeans (EJB) persistence. You can assign policies to an entity bean and to individual methods on an entity bean's home, remote, or local interfaces during assembly. You can set access intents only within EJB Version 2.x-compliant modules for entity beans with CMP Version 2.x.

Concurrency control
Concurrency control is the management of contention for data resources. A concurrency control scheme is considered *pessimistic* when it locks a given resource early in the data-access transaction and does not release it until the

transaction is closed. A concurrency control scheme is considered *optimistic* when locks are acquired and released over a very short period of time at the end of a transaction.

Related tasks

Using access intent policies

Related reference

"Access intent exceptions" on page 247

Access intent -- isolation levels and update locks

Troubleshooting tips for EJBDEPLOY relationships

Problems may exist when EJBDeploy creates a data relationship in DB2 for z/OS Version 7.x. EJBDeploy creates a table with a composite of the two primary keys of the EJBs that are related to each other. If the composite keys are larger than 254 characters, DB2 for z/OS V7.x will not accept this relationship and the user will be conforonted with errors such as:

```
DSNT408I SQLCODE = -613, ERROR:  THE PRIMARY KEY OR A UNIQUE CONSTRAINT
IS TOO LONG OR HAS TOO MANY COLUMNS
DSNT418I SQLSTATE   = 54008 SQLSTATE RETURN CODE
```

This problem can be seen when the primary keys that are created for the two related beans have primary keys that are strings. This results in the composite being made up of 2 varchar(250) primary keys for a total of 500, which is greater than 254 maximum in DB2 for z/OS version 7.x.

Things to consider when utilizing top-down mappings to ensure you do not experience this problem:

- Top-down mappings are a guideline and must be reviewed with the DBA.
- Schemas created 'top-down' by EJBDeploy are designed only for testing, and as a guideline for the actual schema required. The use of the 'meet-in-the-middle' mapping does not present this problem.
- The composite key constraint problem is not experienced when using DB2 V8, which has 2K max key lengths.

EJBDEPLOY_JVM_OPTIONS

Set the `EJBDEPLOY_JVM_OPTIONS` property to override Java virtual machine (JVM) options that are passed to the code that deploys EJBs (ejbdeploy.sh). Set this property in one of the following locations:
deploymentmanager/bin/setupCmdLine.sh or *appServerHome*/bin/setupCmdLine.sh

For example, the following command increases the heap size of the JVM for ejbdeploy.sh:

```
export EJBDEPLOY_JVM_OPTIONS="-Xms128m -Xmx512m"
```

Related tasks

Chapter 6, "Troubleshooting by task," on page 233

"Troubleshooting by component" on page 5

Important file for message-driven beans

The following file in the WAS_HOME/temp directory is important for the operation of the WebSphere Application Server messaging service, so should not be deleted. If you do need to delete the WAS_HOME/temp directory or other files in it, ensure that you preserve the following file:

server_name-**durableSubscriptions.ser**
You should not delete this file, because the messaging service uses it to keep track of durable subscriptions for message-driven beans. If you uninstall an application that contains a message-driven bean, this file is used to unsubscribe the durable subscription.

Client applications

A client program does not work

What kind of problem are you seeing?

ActiveX client fails to display ASP files, or WebSphere Application Server resources (JSP files, servlet, or HTML pages) or both

A possible cause of this problem is that both IIS for serving Active Server Pages (ASP) files and an HTTP server that supports WebSphere Application Server (such as IBM HTTP Server) are deployed on the same host. This deployment leads to misdirected HTTP traffic if both servers are listening on the same port (such as the default port 80).

To resolve this problem, either:
- Open the IIS administrative panel, and edit the properties of the default Web server to change the port number to a value other than 80
- Install IIS and the HTTP server on separate servers.

For current information available from IBM Support on known problems and their resolution, see the IBM Support page.

IBM Support has documents that can save you time gathering information needed to resolve this problem. Before opening a PMR, see the IBM Support page.

Related tasks

Chapter 6, “Troubleshooting by task,” on page 233

“Troubleshooting by component” on page 5

Debugging client exceptions

Start with the client and work your way backward to find the problem. When tracing exceptions back to the original problem, be aware that the RMI/IIOP protocol requires that some exceptions undergo conversion from one type to another as the exception passes through the runtime. Usually this transformation is between CORBA::SystemExceptions and RMI RemoteExceptions. Pay special attention to the CORBA::SystemException minor codes which indicate that a type transformation has occurred.

Caused by:	System exception (thrown by runtime)	User exception (thrown by application code)
Look for:	• CEEDUMPs in controller (region) or servant (region). These dumps indicate that the runtime had an error • JRAS error log entries, which can narrow the error down the exception to a specific function within the runtime	• CEEDUMPs • JRAS error log entries and traces

Actions:	• Look at the minor code that is listed. • Look for fixes that address similar symptoms or minor codes. • System exceptions usually represent the detection of an unexpected error, and therefore (unless directed by the documentation of the minor code) will often require IBM assistance to identify the problem.	• Look at your application for any sign of error. • Look for system failures, such as a system exception in the controller (region). If you find a system exception, follow the steps to the left for diagnosing a system exception.

Application client troubleshooting tips

This section provides some debugging tips for resolving common Java 2 Platform Enterprise Edition (J2EE) application client problems. To use this troubleshooting guide, review the trace entries for one of the J2EE application client exceptions, and then locate the exception in the guide. Some of the errors in the guide are samples, and the actual error you receive can be different than what is shown here. You might find it useful to rerun the `launchClient` command specifying the `-CCverbose=true` option. This option provides additional information when the J2EE application client run time is initializing

Error: java.lang.NoClassDefFoundError

Explanation This exception is thrown when Java code cannot load the specified class.

Possible causes
- Invalid or non-existent class
- Class path problem
- Manifest problem

Recommended response

Check to determine if the specified class exists in a Java Archive (JAR) file within your Enterprise Archive (EAR) file. If it does, make sure the path for the class is correct. For example, if you get the exception:

```
java.lang.NoClassDefFoundError:
WebSphereSamples.HelloEJB.HelloHome
```

verify that the HelloHome class exists in one of the JAR files in your EAR file. If it exists, verify that the path for the class is WebSphereSamples.HelloEJB.

If both the class and path are correct, then it is a class path issue. Most likely, you do not have the failing class JAR file specified in the client JAR file manifest. To verify this situation, perform the following steps:

1. Open your EAR file with the Application Server Toolkit or the Rational Web Developer assembly tool, and select the Application Client.
2. Add the names of the other JAR files in the EAR file to the Classpath field.

This exception is generally caused by a missing Enterprise Java Beans (EJB) module name from the Classpath field.

If you have multiple JAR files to enter in the Classpath field, be sure to separate the JAR names with spaces.

If you still have the problem, you have a situation where a class is loaded from the file system instead of the EAR file. This error is difficult to debug because the offending class is not the one specified in the exception. Instead, another class is loaded from the file system before the one specified in the exception. To correct this error, review the class paths specified with the -CCclasspath option and the class paths configured with the Application Client Resource Configuration Tool. Look for classes that also exist in the EAR file. You must resolve the situation where one of the classes is found on the file system instead of in the `.ear` file. Remove entries from the classpaths, or include the `.jar` files and classes in the `.ear` file instead of referencing them from the file system.

If you use the -CCclasspath parameter or resource classpaths in the Application Client Resource Configuration Tool, and you have configured multiple JAR files or classes, verify they are separated with the correct character for your operating system. Unlike the Classpath field, these class path fields use platform-specific separator characters, usually a colon (on UNIX platforms) or a semi-colon (on Windows systems).

Note: The system class path is not used by the Application Client run time if you use the launchClient batch or shell files. In this case, the system class path would not cause this problem. However, if you load the launchClient class directly, you do have to search through the system class path as well.

Error: com.ibm.websphere.naming.CannotInstantiateObjectException: Exception occurred while attempting to get an instance of the object for the specified reference object. [Root exception is javax.naming.NameNotFoundException: xxxxxxxxxx]

Explanation

This exception occurs when you perform a lookup on an object that is not installed on the host server. Your program can look up the name in the local client Java Naming and Directory Interface (JNDI) name space, but received a NameNotFoundException exception because it is not located on the host server. One typical example is looking up an EJB component that is not installed on the host server that you access. This exception might also occur if the JNDI name you configured in your Application Client module does not match the actual JNDI name of the resource on the host server.

Possible causes

- Incorrect host server invoked
- Resource is not defined
- Resource is not installed
- Application server is not started
- Invalid JNDI configuration

Recommended response

If you are accessing the wrong host server, run the `launchClient` command again with the -CCBootstrapHost parameter specifying the correct host server name. If you are accessing the correct host server, use the product `dumpnamespace` command line tool to see a listing of the host server JNDI name space. If you do not see the failing object name, the resource is either not installed on the host server or the appropriate application server is not started. If you determine the resource is already installed and started, your JNDI name in your client application does not match the global JNDI name on the host server. Use the Application Server Toolkit to compare the JNDI bindings value of the failing object name in the client application to the JNDI bindings value of the object in the host server application. The values must match.

Error: javax.naming.ServiceUnavailableException: A communication failure occurred while attempting to obtain an initial context using the provider url: ″iiop://[invalidhostname]″. Make sure that the host and port information is correct and that the server identified by the provider URL is a running name server. If no port number is specified, the default port number 2809 is used. Other possible causes include the network environment or workstation network configuration. Root exception is org.omg.CORBA.INTERNAL: JORB0050E: In Profile.getIPAddress(), InetAddress.getByName[invalidhostname] threw an UnknownHostException. minor code: 4942F5B6 completed: Maybe

Explanation

This exception occurs when you specify an invalid host server name.

Possible causes

- Incorrect host server invoked
- Invalid host server name

Recommended response

Run the `launchClient` command again and specify the correct name of your host server with the -CCBootstrapHost parameter.

Error: javax.naming.CommunicationException: Could not obtain an initial context due to a communication failure. Since no provider URL was specified, either the bootrap host and port of an existing ORB was used, or a new ORB instance was created and initialized with the default bootstrap host of ″localhost″ and the default bootstrap port of 2809. Make sure the ORB bootstrap host and port resolve to a running name server. Root exception is org.omg.CORBA.COMM_FAILURE: WRITE_ERROR_SEND_1 minor code: 49421050 completed: No

Explanation

This exception occurs when you run the `launchClient` command to a host server that does not have the Application Server started. You also receive this exception when you specify an invalid host server name. This situation might occur if you do not specify a host server name when you run the launchClient tool. The default behavior is for the launchClient tool to run to the local host, because WebSphere Application Server does not know the name of your host server. This default behavior only works when you are running the client on the same machine with WebSphere Application Server is installed.

Possible causes

- Incorrect host server invoked
- Invalid host server name
- Invalid reference to `localhost`
- Application server is not started
- Invalid bootstrap port

Recommended response

If you are not running to the correct host server, run the `launchClient` command again and specify the name of your host server with the `-CCBootstrapHost` parameter. Otherwise, start the Application Server on the host server and run the `launchClient` command again.

Error: javax.naming.NameNotFoundException: Name comp/env/ejb not found in context ″java:″

Explanation

This exception is thrown when the Java code cannot locate the specified name in the local JNDI name space.

Possible causes

- No binding information for the specified name
- Binding information for the specified name is incorrect
- Wrong class loader was used to load one of the program classes
- A resource reference does not include any client configuration information

Recommended response

Open the EAR file with the Application Server Toolkit, and check the bindings for the failing name. Ensure this information is correct. If you are using Resource References, open the EAR file with the Application Client Resource Configuration Tool, and verify that the Resource Reference has client configuration information and the name of the Resource Reference exactly matches the JNDI name of the client configuration. If the values are correct, you might have a class loader error.

Error: java.lang.ClassCastException: Unable to load class: org.omg.stub.WebSphereSamples.HelloEJB._HelloHome_Stub at com.ibm.rmi.javax.rmi.PortableRemoteObject.narrow(portableRemoteObject.java:269)

Explanation

This exception occurs when the application program attempts to narrow to the EJB home class and the class loaders cannot find the EJB client side bindings.

Possible causes

- The files, *_Stub.class and _Tie.class, are not in the EJB `.jar` file
- Class loader could not find the classes

Recommended response

Look at the EJB `.jar` file located in the `.ear` file and verify the class contains the Enterprise Java Beans (EJB) client side bindings. These are class files with file names that end in `_Stub` and `_Tie`. If the binding classes are in the EJB `.jar` file, then you might have a class loader error.

Error: WSCL0210E: The Enterprise archive file [EAR file name] could not be found. com.ibm.websphere.client.applicationclient.ClientContainerException: com.ibm.etools.archive.exception.OpenFailureException

Explanation

This error occurs when the application client run time cannot read the Enterprise Archive (EAR) file.

Possible causes

The most likely cause of this error is that the system cannot find the EAR file cannot be found in the path specified on the `launchClient` command.

Recommended response

Verify that the path and file name specified on the `launchclient` command are correct. If you are running on the Windows operating system and the path and file name are correct, use a short version of the path and file name (8 character file name and 3 character extension).

The launchClient command appears to hang and does not return to the command line when the client application has finished.

Explanation	When running your application client using the `launchClient` command the WebSphere Application Server run time might need to display the security login dialog. To display this dialog, WebSphere Application Server run time creates an Abstract Window Toolkit (AWT) thread. When your application returns from its main method to the application client run time, the application client run time attempts to return to the operating system and end the Java virtual machine (JVM) code. However, since there is an AWT thread, the JVM code will not end until `System.exit` is called.
Possible causes	The JVM code does not end because there is an AWT thread. Java code requires that `System.exit()` be called to end AWT threads.
Recommended response	• Modify your application to call `System.exit(0)` as the last statement. • Use the `-CCexitVM=true` parameter when you call the `launchClient` command.

For current information available from IBM Support on known problems and their resolution, see the IBM customer support page.

IBM Support has documents that can save you time gathering information needed to resolve this problem. Before opening a PMR, see the IBM customer support page.

Web services

Viewing SYSPRINT output, STDOUT and STDERR streams in an HFS File

If you are familiar with UNIX or NT environments, you may be reluctant to use the facilities of SDSF (or IOF) to view the SYSPRINT output from your Application Server regions. In the past, you may have used a familiar editor (such as vi) in a TELNET session to view the STDOUT and STDERR streams. These STDOUT and STDERR streams have been redirected to SYSPRINT, and it is possible to redirect them to files in the HFS for viewing using an HFS editor.

The JCL below for an Application Server region uses this facility. A new SET statement has been added to point to the LOGPATH directory, and the SYSPRINT DD statement has been changed to point to a file in the LOGPATH/servername directory, in this case named: `was.log.d &LYYMMDD..t&LHHMMSS` The extra period (.) between the date and time variables is not a typographical error, but rather an idiosyncrasy of JCL syntax that is necessary to terminate the first variable. &LYYMMDD will be replaced with the local date in YYMMDD format and &LHHMMSS will be replaced by the local time in HHMMSS format. Using the local date and time ensures a unique file for each instance of the Application Server region that is started. The PATHMODE subparameter sets the file mode to

775 and the PATHOPTS subparameter OWRONLY opens the file for WRITE access and the sub-parameter OCREAT indicates that if the file does not already exist, create it.

```
//TSTST1S PROC IWMSSNM='TSTST1S',PARMS='-ORBsrvname '
//  SET CBCONFIG='/WebSphere390/TSDCONF'
//  SET LOGPATH='/WebSphere390/logs'        <<----- Added to set path
//  SET RELPATH='controlinfo/envfile'
//WSDAS1S EXEC PGM=BBOSR,REGION=0M,TIME=NOLIMIT,
// PARM='/ &PARMS &IWMSSNM'
//BBOENV DD PATH='&CBCONFIG/&RELPATH/&SYSPLEX/&IWMSSNM/current.env'
//CEEDUMP  DD SYSOUT=*
//SYSOUT   DD SYSOUT=*
//*YSPRINT DD SYSOUT=*  Added the following 3 lines for SYSPRINT

//SYSPRINT DD PATHMODE=(SIRWXU,SIRWXG,SIROTH),          <<< 775
//            PATHOPTS=(OWRONLY,OCREAT),                <<<
//            PATH='&LOGPATH/TSTST1S/was.log.d&LYYMMDD..t&LHHMMSS'
```

In the previous example, if the file specified by the SYSPRINT DD statement is created on May 2, 2003 at 1:30:35 PM, the PATH parameter will resolve to: `/WebSphere390/logs/TSTST1S/was.log.d030503.t133035` which will be a unique file for this execution of this server instance.

Universal Discovery, Description, and Integration, Web Service, and SOAP component troubleshooting tips

If you are having problems deploying or executing applications that use WebSphere Application Server Web Services, Universal Discovery, Description, and Integration (UDDI), or SOAP, try these steps:

- Review the troubleshooting documentation for messaging in the information center:
  - WSIF troubleshooting tips
- Investigate the following areas for SOAP-related problems:
  - View the error log of the HTTP server to which the SOAP request is sent.
  - Browse the Web site `http://xml.apache.org/soap/` for FAQs and known SOAP issues.

If none of these steps solves the problem, check to see if the problem has been identified and documented using the links in Diagnosing and fixing problems: Resources for learning.

For current information available from IBM Support on known problems and their resolution, see the IBM Support page.

IBM Support has documents that can save you time gathering information needed to resolve this problem. Before opening a PMR, see the IBM Support page.

Related tasks

Chapter 6, “Troubleshooting by task,” on page 233

Related reference

Troubleshooting installation problems

Errors returned to a client sending a SOAP request

What kind of problem are you seeing?

- SOAPException: faultCode=SOAP-ENV:Client; msg=Error opening socket; java.net.ConnectException: Connection refused: connect
- javax.security.cert.CertPathBuilderException: No end-entity certificate matching the selection criteria could be found.

If none of these errors match the one you see:
- Browse the application server logs.
- Look up any error or warning messages in the message table.

SOAPException: faultCode=SOAP-ENV:Client; msg=Error opening socket; java.net.ConnectException: Connection refused: connect

The most likely cause of this refused connection is that it was sent to the default port, 80, and an HTTP server is not installed or configured.

To verify this situation, send the message directly to the SOAP port; for example, to `http://hostname:9080`. If the message is sent correctly, there are two ways to resolve the problem:
- Continue specifying port 9080 on SOAP requests.
- If an HTTP server is not installed, install one and the associated plug-in component.
- If an HTTP server is installed:
  - Regenerate the HTTP plug-in configuration in the administrative console by clicking **Environment > Update WebServer Plugin**, and restarting the HTTP server.
  - If the problem persists, view the HTTP server access and error logs, as well as the *plugin_install_root*/logs/*web_server_name*/http_plugin.log file for more information.

javax.security.cert.CertPathBuilderException: No end-entity certificate matching the selection criteria could be found

This error usually indicates that new or updated security keys are needed. The security key files are:
- SOAPclient
- SOAPserver
- sslserver.p12

In an installed application, these files are located in the: *install_dir*/installedApps/*application_name*.ear/soapsec.war/key/ directory. After replacing these files, you must stop and restart the application.

To replace these files in a SOAP-enabled application that is not yet installed:
- Expand the *application_name*.ear file.
- Expand the soapsec.war file.
- Replace the security key files in thekey/ directory.
- After you replace these files, install the application and restart the server.

For current information available from IBM Support on known problems and their resolution, see the IBM Support page.

IBM Support has documents that can save you time gathering information needed to resolve this problem. Before opening a PMR, see the IBM Support page.

Related tasks

Chapter 6, “Troubleshooting by task,” on page 233

Related reference

Troubleshooting testing and first time run problems

Troubleshooting Web services

This topic provides an entry into the topics available to learn about ways that you can troubleshoot Web services applications.

This topic provides information for you to troubleshoot during different steps of the development, assembly, deployment, and security processes of a Web service.

Select the Web services topic area that you want to troubleshoot:

- Command-line tools

  This topic provides information on troubleshooting the **WSDL2Java** command-line tool and the **Java2WSDL** command-line tool.
- Java compiler errors

  This topic discusses troubleshooting compiled bindings of Web services.
- Serialization or deserialization errors

  This topic presents problems you might encounter performing serialization and deserialization in Web services.
- Authentication challenges and authorization failures with Web services security

  This topic discusses troubleshooting authentication and authorization when you are securing Web services.

  Related tasks

  Chapter 6, "Troubleshooting by task," on page 233

  Related reference

  "Universal Discovery, Description, and Integration, Web Service, and SOAP component troubleshooting tips" on page 17

Troubleshooting Web services command-line tools

This topic discusses troubleshooting the **WSDL2Java** and **Java2WSDL** command-line tools that are used when you develop Web services.

Each section in this topic is a problem that you might experience while using the WSDL2Java or Java2WSDL tool. A solution is provided to help you troubleshoot the problem.

Multiprotocol port component restrictions with JSR109 Version 1.0 and 1.1

Java Specification Requests (JSR) 109 specification validation errors occur when deploying an enterprise archive (EAR) file that contains a Web Services Description Language (WSDL) file with `http`, `jms` and `ejb` bindings generated by the **Java2WSDL** tool.

The JSR 109 specification requires each port component defined in the `webservices.xml` deployment descriptor file to refer to unique <servlet-class> elements in `web.xml` file for a JavaBeans implementation, or a unique <session> element in `ejb-jar.xml` file for an Enterprise JavaBeans (EJB) implementation. The servlet and session EJB are located in the `webservices.xml` file represented by <servlet-link> or <ejb-link> element.

The **WSDL2Java** tool maps the ports found in a WSDL file to port components that are in the generated `webservices.xml` file. If a single Web service has multiple bindings, in addition to a port for each of these bindings, the `webservices.xml` file contains multiple port components that should all point to the same EJB

(<session>) or Java bean (<servlet-class>). Because of the JSR 109 restrictions, the webservices.xml file is not valid and errors can occur during the deployment process.

The following example displays the error:

```
Error in <module> : CHKW6030E: Implementation class <class> referred to by port
components<port1> and <port2>. (JSR109 1.0: 7.1.2).
```

Here is the error with sample data:

```
Error in WebSvcsInSession20EJB.jar : CHKW6030E: Implementation class WSMultiProtocol
referred to by port components WSMultiProtocolJMS and WSMultiProtocolEJB.(JSR109 1.0: 7.1.2).
```

You can work around the restriction by creating multiple <session> EJB definitions within the ejb-jar.xml file that all point to the same implementation class, home interface and remote interface. You can still use the same classes, but the ejb-jar.xml file <session> definitions that reference the classes and the interfaces must be duplicated.

The following is an example of the webservices.xml file. Look for the classes and interfaces:

```
<webservices>
  <webservice-description>
    <webservice-description-name>WSMultiProtocolService</webservice-description-name>
    <wsdl-file>META-INF/wsdl/WSMultiProtocol.wsdl</wsdl-file>
    <jaxrpc-mapping file>META-INF/WSMultiProtocol_mapping.xml</jaxrpc-mapping file>
    <port-component>
      <port-component-name>WSMultiProtocolEjb</port-component-name>
      <wsdl-port>
        <namespaceURI>http://ejb.pli.tc.wssvt.ibm.com</namespaceURI>
        <localpart>WSMultiProtocolEjb</localpart>
      </wsdl-port>
      <service-endpoint-interface>com.ibm.wssvt.tc.pli.ejb.WSMultiProtocol
   </service-endpoint-interface>
      <service-impl-bean>
        <ejb-link>WSMultiProtocol</ejb-link>
      </service-impl-bean>
    </port-component>
    <port-component>
  <port-component-name>WSMultiProtocolJMS</port-component-name>
      <wsdl-port>
        <namespaceURI>http://ejb.pli.tc.wssvt.ibm.com</namespaceURI>
        <localpart>WSMultiProtocolJMS</localpart>
      </wsdlport>
      <service-endpoint-interface>com.ibm.wssvt.tc.pli.ejb.WSMultiProtocol
   </service-endpoint-interface>
      <service-impl-bean>
        <ejb-link>WSMultiProtocol_2</ejb-link>
      </service-impl_bean>
  </port-component>
        <port-component>
         <port-component-name>WSMultiProtocolJMS</port-component-name>
      <wsdl-port>
        <namespaceURI>http://ejb.pli.tc.wssvt.ibm.com</namespaceURI>
        <localpart>WSMultiProtocolJMS</localpart>
      </wsdlport>
      <service-endpoint-interface>com.ibm.wssvt.tc.pli.ejb.WSMultiProtocol
   </service-endpoint-interface>
      <service-impl-bean>
        <ejb-link>WSMultiProtocol_3</ejb-link>
      </service-impl_bean>
  </port-component>
 </webservice-description>
</webservices>
```

The following is an example of the ejb-jar.xml file. Look for the classes and interfaces, and how they are duplicated:

```
<ejb-jar-id="ejb-jar_ID">
 <display-name>WebSvcsInsSession20EJB</display-name>
 <enterprise-beans>
  <session-id="WSMultiProtocol">
   <ejb-name>WSMultiProtocol</ejb-name>
   <home>com.ibm.wssvt.tc.pli.ejb.WSMultiProtocolHome</home>
   <remote>com.ibm.wssvt.tc.pli.ejb.WSMultiProtocol</remote>
   <ejb-class>com.ibm.wssvt.tc.pli.ejb.WSMultiProtocolWebSvcsBean</ejb-class>
   <session-type>Stateless</session-type>
   <transaction-type>Container</transaction-type>
   <ejb-ref-id="EjbRef_1082407586720">
    <description></description>
    <ejb-ref-name>ejb/BeneficiarySession</ejb-ref-name>
    <ejb-ref-type>Session</ejb-ref-type>
    <home>com.ibm.wssvt.tc.pli.ejb.BeneficiarySessionHome</home>
    <remote>com.ibm.wssvt.tc.pli.ejb.BeneficiarySession</remote>
    <ejb-link>PolicySession20EJB.jar#BeneficiarySession</ejb-link>
   </ejb-ref>
   <ejb-ref-id="EjbRef_1082407586790">
    <description></description>
    <ejb-ref-name>ejb/PolicySession</ejb-ref-name>
    <ejb-ref-type>Session</ejb-ref-type>
    <home>com.ibm.wssvt.tc.pli.ejb.PolicySessionHome</home>
    <remote>com.ibm.wssvt.tc.pli.ejb.PolicySession</remote>
    <ejb-link>PolicySession20EJB.jar#PolicySession</ejb-link>
   </ejb-ref>


  <session-id="WSMultiProtocol_2">
   <ejb-name>WSMultiProtocol_2</ejb-name>
   <home>com.ibm.wssvt.tc.pli.ejb.WSMultiProtocolHome</home>
   <remote>com.ibm.wssvt.tc.pli.ejb.WSMultiProtocol</remote>
   <ejb-class>com.ibm.wssvt.tc.pli.ejb.WSMultiProtocolWebSvcsBean</ejb-class>
   <session-type>Stateless</session-type>
   <transaction-type>Container</transaction-type>
   <ejb-ref-id="EjbRef_1082407586720_2">
    <description></description>
    <ejb-ref-name>ejb/BeneficiarySession</ejb-ref-name>
    <ejb-ref-type>Session</ejb-ref-type>
    <home>com.ibm.wssvt.tc.pli.ejb.BeneficiarySessionHome</home>
    <remote>com.ibm.wssvt.tc.pli.ejb.BeneficiarySession</remote>
    <ejb-link>PolicySession20EJB.jar#BeneficiarySession</ejb-link>
   </ejb-ref>
   <ejb-ref-id="EjbRef_1082407586790_2">
    <description></description>
    <ejb-ref-name>ejb/PolicySession</ejb-ref-name>
    <ejb-ref-type>Session</ejb-ref-type>
    <home>com.ibm.wssvt.tc.pli.ejb.PolicySessionHome</home>
    <remote>com.ibm.wssvt.tc.pli.ejb.PolicySession</remote>
    <ejb-link>PolicySession20EJB.jar#PolicySession</ejb-link>
   </ejb-ref>


  <session-id="WSMultiProtocol_3">
   <ejb-name>WSMultiProtocol_3</ejb-name>
   <home>com.ibm.wssvt.tc.pli.ejb.WSMultiProtocolHome</home>
   <remote>com.ibm.wssvt.tc.pli.ejb.WSMultiProtocol</remote>
   <ejb-class>com.ibm.wssvt.tc.pli.ejb.WSMultiProtocolWebSvcsBean</ejb-class>
   <session-type>Stateless</session-type>
   <transaction-type>Container</transaction-type>
   <ejb-ref-id="EjbRef_1082407586790_3">
    <description></description>
    <ejb-ref-name>ejb/BeneficiarySession</ejb-ref-name>
    <ejb-ref-type>Session</ejb-ref-type>
```

```
    <home>com.ibm.wssvt.tc.pli.ejb.BeneficiarySessionHome</home>
    <remote>com.ibm.wssvt.tc.pli.ejb.BeneficiarySession</remote>
    <ejb-link>PolicySession20EJB.jar#BeneficiarySession</ejb-link>
   </ejb-ref>
   <ejb-ref-id="EjbRef_1082407586790_3">
    <description></description>
    <ejb-ref-name>ejb/PolicySession</ejb-ref-name>
    <ejb-ref-type>Session</ejb-ref-type>
    <home>com.ibm.wssvt.tc.pli.ejb.PolicySessionHome</home>
    <remote>com.ibm.wssvt.tc.pli.ejb.PolicySession</remote>
    <ejb-link>PolicySession20EJB.jar#PolicySession</ejb-link>
   </ejb-ref>
  </session>
```

Avoiding application errors after uninstalling an interim fix, a fix pack, or a refresh pack

If an application uses functions that are provided by a particular fix and you remove the fix, the application displays an error message. If you remove a fix, make sure that you retest your applications to check for errors. Redeploy any applications that display an error message because of the missing fix.

For example, suppose you install a fix pack on WebSphere Application Server. Then, you create the stock quote Web service, StockQuote. The **WSDL2Java** tool is used in a deployer role and generates a ServiceLocator class that extends the AgnosticService class.

If you uninstall the fix pack, the application is using a new Web services class (AgnosticService) that the version of WebSphere Application Server does not support. The application throws the following error:

```
java.lang.NoClassDefFoundError:
 Error while defining class:
 com.ibm.ws.wsfvt.test.stockquote.StockQuoteServiceLocator
 This error indicates that the class:
 com.ibm.webservices.multiprotocol.AgnosticService
 could not be located  while defining the class:
 com.ibm.ws.wsfvt.test.stockquote.StockQuoteServiceLocator
```

You need to redeploy the application on the Application Server to emit code that does not use the WebSphere Application Server version that is not supported by the Web services class that you use.

Using a proxy server to access the Internet while running the WSDL2Java command causes your connection to time out

If you use an environment that requires a proxy server to access the Internet during the run of the **WSDL2Java** command, the **WSDL2Java** command might not find the Internet information because the proxy server has the potential to time out. For example, if the input WSDL file is located on the Internet instead of a local drive, and you need to retrieve it from the Internet, the **WSDL2Java** command fails to find the file because the proxy server times out.

You can work around this problem by editing the `WSDL2Java.bat` file when using a Windows operating system or the `WSDL2Java.sh` file if you are using a Linux or Unix operating system. These files are located in the *<install_root>*`/WebSphere/AppServer/bin` directory.

If you use a Windows operating system, set your proxy host and port values in the WSDL2Java.bat file:

```
PROXY_INFO="-Dproxy.httpHost=yourPproxyHost -Dproxy.httpPort=yourProxyPort
```

If you use a Linux or Unix operating system, set your proxy host and port values in the WSDL2Java.sh file:

```
PROXY_INFO="-Dproxy.httpHost=yourProxyHost -Dproxy.httpPort=yourProxyPort
```

Emitter failure error occurs when running the WSDL2Java command on a WSDL document containing a JMS-style endpoint URL

If you run the **WSDL2Java** command-line tool on a WSDL document that contains a JMS-style endpoint Web address, for example, jms:/..., the urlprotocols.jar file that contains the custom protocol handler for the JMS protocol must be in the CLASSPATH variable statement. The error **WSWS3099E: Error: Emitter failure. Invalid endpoint address in port <x> in service <y>: <jms-url-string>** is avoided by making sure the urlprotocols.jar file is in the CLASSPATH variable statement.

To add the urlprotocols.jar file to the CLASSPATH variable statement:

On Windows platforms, edit the *install_root*\bin\setupCmdLine.bat file and locate the line that sets the *WAS_CLASSPATH* environment variable. Add *%install_root%*\lib\urlprotocols.jar to the end of the line that sets the *WAS_CLASSPATH* environment variable.

On Linux and Unix platforms, edit the *install_root*/bin/setupCmdLine.sh file and add $*install_root*/lib/urlprotocols.jar to the end of the line that sets the *WAS_CLASSPATH* environment variable.

Make sure to use the proper deliminator character for your platform, for example, use a semi-colon (;) for Windows platforms and a colon (:) for Linux and Unix platforms.

Related tasks

"Troubleshooting Web services" on page 260
This topic provides an entry into the topics available to learn about ways that you can troubleshoot Web services applications.

Troubleshooting Web services compiled bindings

This topic discusses troubleshooting compiled bindings of Web services that are developed and implemented based on the Web Services for Java 2 Platform, Enterprise Edition (J2EE) specification

Each section in this topic is a problem that you might experience with compiled bindings for Web services. A solution is provided to help you troubleshoot the problem.

Context root not recognized when mapping the default XML namespace to a Java package

When you map the default XML namespace to a Java package the context root is not recognized. If two namespaces are the same up to the first slash, they map to the same Java package. For example, the XML namespaces http://www.ibm.com/foo and http://www.ibm.com/bar both map to the www.ibm.com Java package . Use the -NStoPkg option of the **Java2WSDL** command to specify

the package for the fully qualified namespace.

Java code to WSDL mapping cannot be reversed to the original Java code

If you find that a WSDL file that you created with the **Java2WSDL** command-line tool cannot be compiled when regenerated into Java code using the **WSDL2Java** command-line tool, it is because the Java API for XML-based remote procedure call (JAX-RPC) mapping from Java code to WSDL is not reversible back to the original Java code.

To troubleshoot this problem, try specifying the -introspect option to the **WSDL2Java** command. The -instrospect option indicates to the **WSDL2Java** command to look into existing Java classes and gather information useful in generating artifacts that match the original Java code.

Related tasks

"Troubleshooting Web services" on page 260
This topic provides an entry into the topics available to learn about ways that you can troubleshoot Web services applications.

Troubleshooting the run time for a Web services client

This topic discusses troubleshooting Web services clients.

Each section in this topic is a problem that you might experience during the run-time of a Web services client. A solution is provided to help you troubleshoot the problem.

Runtime migration error

If you installed a Web service application that was developed for a WebSphere Application Server version prior to Version 6.0, you might get the following exception:

```
WSWS3701E: Error: An exception was encountered. Use wsdeploy to deploy
your application. This may correct the problem. The exception
is <exception data>.
```

This exception indicates that a problem occurred while running the application that was developed with tools supported by versions prior to Version 6.0. A solution to the problem is to uninstall the application, run the **wsdeploy** command and redeploy the application.

WebServicesFault exception displays during the application server run time for certain Web Services Description Language (WSDL) files

A `WebServicesFault` exception displays during the application server run time for WSDL files that define operations with `document style` and `literal use`, and use the SOAP header to transmit the input data.

If the WSDL files define the operation with `document style` and `literal use`, and this operation maps the input to the SOAP header, the Web services run time fails to find the correct operation for the target service and the `WebServicesFault` exception displays.

To solve the problem, change the WSDL files so that the operation does not have input that uses the SOAP header to transmit the data.

Increase the value of the ConnectionIOTimeOut parameter to avoid receiving an exception when hosting Web services on WebSphere Application Server

When hosting Web services on WebSphere Application Server, the following exception displays: `java.net.SocketTimeOutException: Read Timed Out`.

A slow network connection between the client and the Web service causes this problem. In such cases, the HTTP socket might time out before the Web service engine completely reads the SOAP request. In the majority of cases, a sudden increase in overall network activity causes this problem. The problem can also occur when the client is accessing the Web service from a slow network connection and when the SOAP request has a lot of data.

To solve the problem, increase the ConnectionIOTimeOut parameter for the Web container HTTP transport. The default value is 5 seconds. Increase the value to 30 seconds or greater. Set the value using the administrative console. Click **Servers** > **Application Servers** > ***server_name*** > ***Web Container*** > ***HTTP Transports*** > ***port_number*** > **Custom Properties** > **New**. Type the following property name and value:

- **Name:** ConnectionIOTimeOut
- **Value:** 30

If the Web service is hosted in a clustered environment, set the property on each application server in the cluster. If your application server is listening on more than one port number, set the property on all ports.

Related tasks

"Troubleshooting Web services" on page 260
This topic provides an entry into the topics available to learn about ways that you can troubleshoot Web services applications.

Troubleshooting serialization and deserialization in Web services

This topic discusses problems that you can have when you perform serialization and deserialization in Web services.

Each section in this topic is a problem that you might experience while serializing and deserializing Web services. A solution is provided to help you troubleshoot the problem.

Time zone information in deserialized java.util.Calendar is not as expected

When the client and server are based on Java code and a `java.util.Calendar` instance is received, the time zone in the received `java.util.Calendar` instance might be different from that of the `java.util.Calendar` instance that was sent.

This difference occurs because the `java.util.Calendar` is encoded as an xsd:dateTime for transmission. An xsd:dateTime is required to encode the correct time (base time plus or minus a time zone offset), but is not required to preserve locale information, including the original time zone.

The fact that the time zone for the current locale is not preserved needs to be accounted for when comparing Calendar instances. The `java.util.Calendar` class equals method checks that the time zones are the same when determining equality. Because the time zone in a deserialized Calendar instance might not match the current locale, use the before and after comparison methods to test that two Calendars refer to the same date and time as shown in the following examples:

```
java.util.Calendar c1 = .../// Date and time in time zone 1
java.util.Calendar c2 = .../// Same date and equivalent time, but in time zone 2

// c1 and c2 are not equal because their time zones are different
if (c1.equals (c2)) System.out.println("c1 and c2 are equal");

// but c1 and c2 do compare as "not before and not after" since they represent
the same date and time
if (!c1.after(c2) & !c1.before(c2) {
   System.out.println("c1 and c2 are equivalent");
}
```

Mixing Web services client and server bindings causes errors

Web Services for Java 2 Platform, Enterprise Edition (J2EE) and the Java API for XML-based remote procedure call (JAX-RPC) do not support *round-trip* mapping between Java code and a Web Services Description Language (WSDL) document for all Java types. For example, you cannot turn (serialize) a Java Date into XML code and then turn it back (deserialize) into a Java Date. This action deserializes as Java Calendar.

If you have a Java implementation that you create a WSDL document from, and you generate client bindings from the WSDL document, the client classes can be different from the server classes even though the client classes have the same package and class names. The Web service client classes must be kept separate from the Web service server classes. For example, do not place the Web service server bindings classes in a utility Java archive (JAR) file and then include a Web service client JAR file that references the same utility JAR file.

If you do not keep the Web services client and server classes separate, a variety of exceptions can occur, depending on the Java classes used. The following is a sample stack trace error that can occur:

```
com.ibm.ws.webservices.engine.PivotHandlerWrapper TRAS0014I: The following
 exception was loggedjava.lang.NoSuchMethodError:
 com.ibm.wssvt.acme.websvcs.ExtWSPolicyData: method
 getStartDate()Ljava/util/Date; not found
at com.ibm.wssvt.acme.websvcs.ExtWSPolicyData_Ser.addElements
  (ExtWSPolicyData_Ser.java: 210)
at com.ibm.wssvt.acme.websvcs.ExtWSPolicyData_Ser.serialize
  (ExtWSPolicyData_Swer.java:29)
at com.ibm.ws.webservices.engine.encoding.SerializationContextImpl.
   serializeActual (SerializationContextImpl.java 719)
at com.ibm.ws.webservices.engine.encoding.SerializationContextImpl.
   serialize (SerializationContextImpl.java: 463)
```

The problem is caused by using an interface as shown in the following example for the service endpoint interface in the service implementation:

```
package server:
public interface Test_SEI extends java.rmi.Remote {
 public java.util.Calendar getCalendar () throws java.rmi.RemoteException;
 public java.util.Date getDate() throws java.rmi.RemoteException;
}
```

When this interface is compiled and run through the **Java2WSDL** command-line tool, the WSDL document maps the methods as shown in the following example:

```
<wsdl:message name="getDateResponse">
 <wsdl:part name="getDateReturn" type="xsd:dateTime"/>
</wsdl:message>
```

```
<wsdl:message name="getCalendarResponse">
 <wsdl:part name="getCalendarReturn" type="xsd:dateTime"/>
</wsdl:message>
```

The JAX-RPC mapping implemented by the **Java2WSDL** tool has mapped both java.util.Date and java.util.Calendar instances to the xsd:dateTime XML type . The next step is to use the generated WSDL file to create a client for the Web service. When you run the **WSDL2Java** command-line tool on the generated WSDL, the generated classes include a different version of the server.Test_SEI interface, for example:

```
package server;
public interface Test_SEI extends java.rmi.Remote {
 public java.util.Calendar getCalendar() throws java.rmi.RemoteException;
 public java.util.Date getDate() throws java.rmi.RemoteException;
}
```

Note: The client version of the service.Test_SEI interface is different from the server version in that both the getCalendar and getDate methods return java.util.Calendar. The serialization and deserialization code that the client expects is the client version of the service endpoint interface. If the server version inadvertently appears in the client CLASSPATH variable, at either compilation or run time, an error occurs.

In addition to theNoSuchMethod error, the IncompatibleClassChangeError and ClassCastException can occur. However, almost any run-time exception can occur. The best practice is to be diligent about separating client Web services bindings classes from server Web services bindings classes. Always place the client bindings classes and server bindings classes in separate modules. If theses binding classes are in the same application, place the bindings classes in utility JAR files that are not shared between modules.

Related tasks

"Troubleshooting Web services" on page 260
This topic provides an entry into the topics available to learn about ways that you can troubleshoot Web services applications.

Troubleshooting authentication and authorization for Web services security based on Web Services for J2EE

These Web services are developed and implemented based on the Web Services for Java 2 platform, Enterprise Edition (J2EE) specification. This topic discusses the following troubleshooting authentication and authorization when you are securing Web services:

- Authentication challenge or authorization failure is displayed
- Web services security enabled application fails to start
- Applications with Web services security enabled cannot interoperate between WebSphere Application Server Version 6.0.x and Version 5.0.2

Authentication challenge or authorization failure is displayed

You might encounter an authentication challenge or an authorization failure if a thread switch occurs. For example, an application might create a new thread or a raw socket connection to a servlet might open. A thread switch is not recommended by the Java 2 Platform, Enterprise Edition (J2EE) specification because the security context information is stored in thread local. When a thread switch occurs, the authenticated identity is not passed from thread local to the new thread. As a result, WebSphere Application Server considers the identity to be

unauthenticated. If you must create a new thread, you must propagate the security context to the new thread. However, this process is not supported by WebSphere Application Server.

Web services security enabled application fails to start

When a Web services security-enabled application fails to start, you might receive an error message similar to the following:

```
[6/19/03 11:13:02:976 EDT] 421fdaa2 KeyStoreKeyLo E WSEC5156E: An exception
while retrieving the key from KeyStore object:
java.security.UnrecoverableKeyException: Given final block not properly padded
```

The cause of the problem is that the keypass value or password provided for a particular key in the key store is invalid. The key store values are specified in the `KeyLocators` elements of one of following binding files: `ws-security.xml`, `ibm-webservices-bnd.xmi` or `ibm-webservicesclient-bnd.xmi`. Verify that the keypass values for keys specified in the KeyLocators elements are correct.

Tracing Web services

You can trace the Web services run-time components, including an unmanaged client, a managed client and a server application. The procedure entry and exit, as well as the processing actions are traceable in the run-time components. You can also trace user-defined exceptions, as well as SOAP messages that use Java Message Service (JMS) or HTTP to request Web services.

The `com.ibm.ws.webservices.engine.*=all=enabled` specification traces the Web services run time only. See step 4 for settings that you can use to trace user-defined exceptions and SOAP messages or review Tracing SOAP messages with tcpmon to learn about tracing SOAP messages with the tcpmon process.

The following tasks describe how you can enable trace for Web services:

1. Enable trace for a Web services unmanaged client.
   a. Create a trace properties file by copying the *%install_root*`\properties\TraceSettings.properties` file to the same directory as your client application Java archive (JAR) file.
   b. Edit the properties file and change the traceFileName value to output the trace data. For example, `traceFileName=c:\\temp\\myAppClient.trc`.
   c. Edit the properties file to remove `com.ibm.ejs.ras.*=all=enabled` and add `com.ibm.ws.webservices.engine.*=all=enabled`.
   d. Add the -DtraceSettingsFile=<*trace_properties_file*> option to the Java command line that is used to run the client, where *trace_properties_file* represents the name of the properties file that you created in the substeps a through c. For example, `java -DtraceSettingsFile=TraceSettings.properties myApp.myAppMainClass`.
2. Enable trace for a Web services-managed client by invoking the **launchClient** command-line tool with the following options:

   -CCtrace=com.ibm.ws.webservices.engine.*=all=enabled

   -CCtracefile=traceFileName For example:

   %install_root%`\bin\launchClient MyAppClient.ear`

   `-CCtrace=com.ibm.ws.webservices.engine.*=all=enabled -CCtracefile=myAppClient.trc`

   See launchClient tool for more information.

3. Enable trace for a Web Services for Java 2 enterprise edition (J2EE) server application.
   a. Start WebSphere Application Server.
   b. Open the administrative console.
   c. Click **Servers** > **Application Servers** > *server*.
   d. Click **Change Log Detail Levels**.
   e. Add or delete the trace string in the text box. For this task, delete the trace string `*=info` and add the trace string `com.ibm.ws.webservices.engine.*=all=enabled`. You can specify the trace string in the text box in one of two ways:
      - Type the trace string directly into the text box. You must separate each trace string by a colon (:) with no spaces. For example:

        ```
        com.ibm.ws.webservices.trace.MessageTrace=finest:com.ibm.ws.webservices.
        engine.Message=finest
        ```
      - Choose a predefined trace string from the section that is listed. The predefined section starts with `*[All Components]`. The predefined tracing strings Web services component are listed under the com.ibm..ws.* section.
        – Click the plus (+) sign to expand the **com.ibm.ws.*** section.
        – Click the predefined trace string. For example, if you want to add a predefined trace string for the SOAP messaging trace, you can click: `com.ibm.ws.webservices.trace.MessageTrace`.
        – Click the trace option from the drop-down list. For example, you can choose off, fatal, severe, warning, audit, info, config, detail, fine, finer, finest, and all. The option, finest, is recommended. When you click on the option, the option is added to the end of the trace string. For example:

          ```
          com.ibm.ws.webservices.trace.MessageTrace=finest
          ```
   f. Click **Save** and **Apply**.

   For more information see, Enabling trace.
4. Enable trace for SOAP messages, user-defined exceptions, or both. The following trace specifications are used to trace SOAP messages:
   - `com.ibm.ws.webservices.trace.MessageTrace=all`

     This specification traces the contents of a SOAP message, including the binary attachment data.

     When the context-type of the SOAP message is not text and xml, the message probably contains attachments. In this case, the message is displayed in the trace file in the hex dump format. The following example illustrates a line in the hex dump format for non-text SOAP messages:

     ```
     0000: 0D 0A 2D 2D 2D 2D 2D 2D - 3D 5F 50 61 72 74 5F 36 ..------=_Part_6
     ```

     – In each trace file line, 16 bytes of the message are displayed
     – The first four digits are a hex number whose value is the byte offset into the SOAP message of the first byte on the line.
     – The next 16 two-digit hex numbers are the contents of each of the consecutive bytes in the message.
     – The ASCII representation of the bytes is displayed in the last 16 characters of the line, with unprintable characters that are represented by a period.
   - `*=off:com.ibm.ws.webservices.*=all`

     You can trace all Web services information, including the SOAP messages and the user-defined exceptions, with this setting.

You can enable logging of user-defined exceptions by specifying the `com.ibm.ws.webservices.trace.UserExceptionTrace=all` trace string. The user-defined exceptions are not logged by default. A user-defined exception is an exception that is defined in the Web Services Description Language (WSDL) file for an operation.

A user-defined exception often indicates an error-free condition. For example, the user-defined OverdrawnException exception, can occur for the service endpoint implementation of the makeWithdrawl method. This exception indicates an expected condition and does not indicate an error in the service endpoint implementation. Because these types of exceptions can occur during normal processing, they are not logged by default. When a user-defined exception is logged, the information is sent to the `trace.log` file and not to the `SystemOut.log` file.

You can also use the following trace strings to enable tracing for user-defined exceptions, as well as other trace points:

- com.ibm.ws.webservices.*=all

  Turns on all Web services run-time trace logs.
- com.ibm.ws.webservices.trace.*=all

  Turns on MessageTrace and UserExceptionTrace.

You have enabled trace for the unmanaged clients, managed clients, and the server applications. Depending on the trace string specification, the trace can include run-time components, user-defined exceptions and SOAP messages.

Analyze the message data.

Related tasks

"Troubleshooting Web services" on page 260
This topic provides an entry into the topics available to learn about ways that you can troubleshoot Web services applications.

"Tracing SOAP messages with tcpmon"
This topic discusses tracing SOAP messages that request Web services by using the tcpmon tool.

Tracing SOAP messages with tcpmon

This topic discusses tracing SOAP messages that request Web services by using the tcpmon tool.

You can use other trace tools to trace SOAP messages, similar to how you can trace Web services components. See Tracing Web services for more information about these other trace tools.

You can trace SOAP messages exchanged between a client and the server by installing a monitor or sniffer application to capture the HTTP traffic between the two points. The WebSphere product provides a utility class, `com.ibm.ws.webservices.engine.utils.tcpmon`, to trace the SOAP messages. The `com.ibm.ws.webservices.engine.utils.tcpmon` class redirects messages from a port, records the messages, and forwards the messages to another port.

WebSphere Application Server typically listens on port 9080, or port 80 if you are using IBM HTTP Server. The tcpmon process can be configured to listen on a particular port, such as 9088, while redirecting messages to another port, such as 9080 or port 80. The client is redirected to use port 9088 to access Web services.

Redirecting an application client to a different port is done by changing the SOAP address in the client Web Services Description Language (WSDL) file to use port 9088 and then running the **wsdeploy** command-line tool on the client enterprise archive (EAR) file to regenerate the service implementation.

Trace SOAP messages in Web services by following the actions listed in the steps for this task section.

1. Set up a development and unmanaged client run-time environment for Web services.
2. Run the **java -Djava.ext.dirs=%WAS_EXT_DIRS% com.ibm.ws.webservices.engine.utils.tcpmon** command. A window labeled TCPMonitor is displayed.
3. Configure the TCPMonitor to listen on port 9088 and forward messages to port 9080.
   a. In the Listen Port # field, enter `9088`.
   b. Click **Listener**.
   c. In the TargetHostname field, enter `localhost`.
   d. In the Target Port # field, enter `9080`.
   e. Click **Add**.
   f. Click the **Port 9088** tab that displays on the top of the page.

The messages exchanged between the client and server appear in the TCPMonitor Request and Response pane.

Save the message data and analyze it.

Related tasks

"Tracing Web services" on page 269
You can trace the Web services run-time components, including an unmanaged client, a managed client and a server application. The procedure entry and exit, as well as the processing actions are traceable in the run-time components. You can also trace user-defined exceptions, as well as SOAP messages that use Java Message Service (JMS) or HTTP to request Web services.

"Troubleshooting Web services" on page 260
This topic provides an entry into the topics available to learn about ways that you can troubleshoot Web services applications.

Frequently asked questions about Web services

This topic presents frequently asked questions about Web services that are developed and implemented based on the Web Services for Java 2 Platform, Enterprise Edition (J2EE) specification.

- What IBM development tools work with Web Services that are developed based on the Web Services for J2EE specification?
- Is Web Services for J2EE technology part of the J2EE specification?
- What is the relationship between the Web Services for J2EE specification and the Web Service Invocation Framework (WSIF)?
- What is the relationship between Apache SOAP 2.3 and the Web Services for J2EE specification?
- What standards does the Web Services for J2EE component of WebSphere Application Server Version 6.0 support?
- Does the Web Services for J2EE technology interoperate with other SOAP implementations, like .NET?
- Can I use a JavaBeans component to implement a Web service using SOAP Java Message Service (JMS) invocation?

- Does the SOAP and JMS support interoperate with other vendors?
- How does two-way messaging with a SOAP and JMS implementation work? Can it support multiple clients making simultaneous requests?

What IBM development tools work with Web Services that are developed based on the Web Services for J2EE specification?

The , Application Server Toolkit (AST) and Rational Web Developer, provide a graphical interface for developing code artifacts, assembling the code artifacts into various archives (modules) and configuring related Java 2 Platform, Enterprise Edition (J2EE) Version 1.2, 1.3 or 1.4 compliant deployment descriptors.

Is Web Services for J2EE technology part of the J2EE specification?

WebSphere Application Server Version 6.0 is based on J2EE 1.4. For WebSphere Application Server Version 5.0.2 and Version 5.1.x, the Web Services for J2EE Version 1.0 specification is an addition to J2EE 1.3. The J2EE specification 1.4 requires support for Web Services for J2EE Version 1.1. Minor differences exist between the J2EE 1.3 Version (JSR-109 Version 1.0) and the J2EE 1.4 Version (JSR-109 Version 1.1).

What is the relationship between the Web Services for J2EE specification and the Web Service Invocation Framework (WSIF)?

Web Services for J2EE and WSIF represent two different programming models for accessing Web services. Web Services for J2EE is standard, Java-centric, and more statically bound to Web Services Description Language (WSDL) documents because of the use of generated stubs. WSIF directly models WSDL. WSIF is more suitable when dynamically interpreting WSDL. WebSphere Application Server Version 6.0 leverages both technologies to achieve dynamic, high performing standards-based Web services implementations.

What is the relationship between Apache SOAP 2.3 and the Web Services for J2EE specification?

The development and implementation of a Web service in V6.x is based on the Web Services for J2EE specification. You are encouraged to migrate from Apache SOAP because this approach is not recommended for future releases. For information about migrating your Apache SOAP Web services, see Migrating Apache SOAP Web services to Web Services for J2EE standards.

What standards does the Web Services for J2EE component of WebSphere Application Server Version 6.0 support?

The following standards are supported by the Web Services for J2EE component of WebSphere Application Server Version 6.0:
- SOAP Version 1.1
- Web Services Description Language (WSDL) Version 1.1
- Web Services for J2EE Version 1.1
- Java API for XML-Based RPC (JAX-RPC) Version 1.1
- SOAP with attachments API for Java (SAAJ) Version 1.2

Does the Web Services for J2EE technology interoperate with other SOAP implementations, like .NET?

WebSphere Application Server Version 6.0 supports Web services that are consistent with the WS-I Basic Profile 1.0, and should interoperate with any other vendor conforming to this specification.

Can I use a JavaBeans component to implement a Web service using SOAP Java Message Service (JMS) invocation?

The SOAP and JMS support uses message-driven beans (MDB) to implement the JMS endpoint. You can use MDBs in the Enterprise JavaBeans (EJB) container and delegated to an enterprise bean. If you want to use a JavaBeans instead of an enterprise bean to implement the service endpoint, you must create a *facade* enterprise bean that delegates to the JavaBeans implementation.

Does the SOAP and JMS support interoperate with other vendors?

No. Currently no specification exists for SOAP and JMS invocationss, therefore each vendor chooses an implementation technique.

How does two-way messaging with a SOAP and JMS implementation work? Can it support multiple clients making simultaneous requests?

Before a client issues a two-way request, it creates a temporary JMS queue to receive the response. This temporary queue is specified as the replyTo destination that is in the outgoing JMS request message. After the server processes the request, it directs the response to the replyTo destination specified in the request message. The client deletes the temporary queue after the response is received. The server can handle simultaneous requests from multiple clients because each incoming request message contains the destination to which the reply is sent.

Related tasks

"Troubleshooting Web services" on page 260
This topic provides an entry into the topics available to learn about ways that you can troubleshoot Web services applications.

Troubleshooting the Web Services Invocation Framework

For information on resolving WebSphere-level problems, see Diagnosing and fixing problems.

To identify and resolve Web Services Invocation Framework (WSIF)-related problems, you can use the standard WebSphere Application Server trace and logging facilities. If you encounter a problem that you think might be related to WSIF, you can check for error messages in the WebSphere Application Server administrative console, and in the application server `stdout.log` file. You can also enable the application server debug trace to provide a detailed exception dump.

A list of the WSIF run-time system messages, with details of what each message means, is provided in Message reference for WSIF.

A list of the main known restrictions that apply when using WSIF is provided in WSIF - Known restrictions.

Here is a checklist of major WSIF activities, with advice on common problems associated with each activity:

Create service
Handcrafted WSDL can cause numerous problems. To help ensure that your WSDL is valid, use a tool such as WebSphere Studio to create your service.

Define transport mechanism
For the Java Message Service (JMS), check that you have set up the Java Naming and Directory Interface (JNDI) correctly, and created the necessary connection factories and queues.

For SOAP, make sure that the deployment descriptor file `dds.xml` is correct - preferably by creating it using WebSphere Studio or similar tooling.

Create client - Java code
Follow the correct format for creating a WSIF service, port, operation and message. For examples of correct code, see the Address Book Sample.

Compile code (client and service)
Check that the build path against code is correct, and that it contains the correct levels of JAR files.

Create a valid EAR file for your service in preparation for deployment to a Web server.

Deploy service
When you install and deploy the service EAR file, check carefully any messages given when the service is deployed.

Server setup and start
Make sure that the WebSphere Application Server `server.policy` file (in the `/properties` directory) has the correct security settings. For more information, see Enabling security for WSIF.

WSIF setup
Check that the `wsif.properties` file is correctly set up. For more information, see Maintaining the WSIF properties file.

Run client
Either check that you have defined the class path correctly to include references to your client classes, WSIF JAR files and any other necessary JAR files, or (preferably) run your client using the WebSphere Application Server launchClient tool.

Here is a list of common errors, and information on their probable causes:

- **"No class definition" errors received when running client code.**

  This problem usually indicates an error in the class path setup. Check that the relevant JAR files are included.

- **"Cannot find WSDL" error.**

  Some likely causes are:
  - The application server is not running.
  - The server location and port number in the WSDL are not correct.
  - The WSDL is badly formed (check the error messages in the application server `stdout.log` file).
  - The application server has not been restarted since the service was installed.

  You might also try the following checks:
  - Can you load the WSDL into your Web browser from the location specified in the error message?
  - Can you load the corresponding WSDL binding files into your Web browser?

- **Your Web service EAR file does not install correctly onto the application server.**

It is likely that the EAR file is badly formed. Verify the installation by completing the following steps:
  - For an EJB binding, run the WebSphere Application Server tool `\bin\dumpnamespace`. This tool lists the current contents of the JNDI directory.
  - For a SOAP over HTTP binding, open the `http://pathToServer/WebServiceName/admin/list.jsp` page (if you have the SOAP administration pages installed). This page lists all currently installed Web services.
  - For a SOAP over JMS binding, complete the following checks:
    - Check that the queue manager is running.
    - Check that the necessary queues are defined.
    - Check the JNDI setup.
    - Use the "display context" option in the `jmsadmin` tool to list the current JNDI definitions.
    - Check that the Remote Procedure Call (RPC) router is running.

- **There is a permissions problem or security error.**

  Check that the WebSphere Application Server `server.policy` file (in the `/properties` directory) has the correct security settings. For more information, see Enabling security for WSIF.

- **Using WSIF with multiple clients causes a SOAP parsing error.**

  Before you deploy a Web service to WebSphere Application Server, you must decide on the scope of the Web service. The deployment descriptor file `dds.xml` for the Web service includes the following line:

```
<isd:provider type="java" scope="Application" ......
```

  You can set the `Scope` attribute to `Application` or `Session`. The default setting is `Application`, and this value is correct if each request to the Web service does not require objects to be maintained for longer than a single instance. If `Scope` is set to `Application` the objects are not available to another request during the execution of the single instance, and they are released on completion. If your Web service needs objects to be maintained for multiple requests, and to be unique within each request, you must set the scope to `Session`. If `Scope` is set to `Session`, the objects are not available to another request during the life of the session, and they are released on completion of the session. If scope is set to `Application` instead of `Session`, you might get the following SOAP error:

```
SOAPException: SOAP-ENV:ClientParsing error, response was:
FWK005 parse may not be called while parsing.;
nested exception is:

[SOAPException: faultCode=SOAP-ENV:Client; msg=Parsing error, response was:

FWK005 parse may not be called while parsing.;
        targetException=org.xml.sax.SAXException:
FWK005 parse may not be called while parsing.]
```

- **Using the same names for JMS messaging queues and queue connection factories that run on application servers on different machines can cause JNDI lookup errors.** You should not use the same names for messaging queues and queue connection factories that run on application servers on different machines, because WSIF always looks first for JMS destinations locally, and only uses the full JNDI reference if it cannot find the destination locally. For example, if you run a Web service on a remote machine, and have an application server running locally that uses the same names for the messaging queues and queue connection factories, then WSIF will find and use the local queues even if the remote JNDI destination is provided in full in the WSDL service definition.
- **A JAX-RPC client running on WebSphere Application Server Version 5 uses SOAP over JMS to invoke a Web service running on a Version 5 application**

server. No username or password is required on the target MQ Series queue. After the application server is migrated to Version 6, and using Version 6 default messaging, client requests fail because basic authentication is now enabled.

The problem appears as a log message:

```
SibMessage    W   [:] CWSIT0009W: A client request failed in the application
server with endpoint <endpoint_name> in bus your_bus with
reason: CWSIT0016E: The user ID null failed authentication in bus your_bus.
```

When the application server is migrated to Version 6, and the default messaging provider (service integration technologies) is used, and global security is enabled for the server or cell, then by default the service integration bus queue destination inherits the security characteristics of the server or cell. So if the server or cell has basic authentication enabled, then the client request fails.

To resolve the problem, you have three choices (in order of security, from least secure to most secure):

– Disable global security.

– For an equivalent level of security to the configuration on Version 5, modify the settings for the service integration bus that hosts the queue destination so that bus security is disabled and therefore the bus does not inherit security characteristics from the server or cell.

– For a greater level of security than the configuration on Version 5, configure basic authentication on each client that uses the service.

To disable global security, refer to either of these topics:

– Global security settings.

– Enabling and disabling global security using scripting.

To disable bus security, use the administrative console to complete the following steps:

1. Navigate to **Service Integration** → **Buses [Content Pane]** → **your_bus**.
2. Clear the **Secure** check box.
3. Save your changes.

To configure basic authentication on each client, use either the administrative console or the `wsadmin` tool. To complete the task using the `wsadmin` tool, see Configuring Web service client port information with the wsadmin tool and use the `WebServicesClientBindPortInfo` `wsadmin` task option. To complete the task using the administrative console, complete the following steps:

1. Navigate to **Applications** → **Enterprise Applications** → **application_instance** → **Web Modules or EJB Modules** → **module_instance** → **Web services: Client security bindings**.
2. Click **HTTP basic authentication** to access the "Configuring HTTP basic authentication with the administrative console" panel.
3. Enter the values in the panel.
4. Save your changes.

- **The current WSIF default SOAP provider (the IBM Web Service SOAP provider) does not fully interoperate with services that are running on the former (Apache SOAP) provider.** This restriction is due to the fact that the IBM Web Service SOAP provider is designed to interoperate fully with a JAX-RPC compliant Web service, and Apache SOAP cannot provide such a service. To enable interoperation, modify either your Web service or the WSIF default SOAP provider as described in WSIF SOAP provider: working with legacy applications.

Trace and logging for WSIF

If you want to enable trace for the Web Services Invocation Framework (WSIF) API within WebSphere Application Server, and have trace, stdout and stderr for the application server written to a well-known location, see Setting up component trace (CTRACE).

WSIF offers trace points at the opening and closing of ports, the invocation of services, and the responses from services.

To trace the WSIF API, you need to specify the following trace string:

```
wsif=all=enabled
```

WSIF also includes a `SimpleLog` utility through which you can run trace when using WSIF outside of WebSphere Application Server. To enable this utility, complete the following steps:

1. Create a file named `commons-logging.properties` with the following contents:

```
org.apache.commons.logging.LogFactory=org.apache.commons.logging.impl.LogFactoryImpl
org.apache.commons.logging.Log=org.apache.commons.logging.impl.SimpleLog
```

2. Create a file named `simplelog.properties` with the following contents:

```
org.apache.commons.logging.simplelog.defaultlog=trace
org.apache.commons.logging.simplelog.showShortLogname=true
org.apache.commons.logging.simplelog.showdatetime=true
```

3. Put both these files, and the `commons-logging.jar` file, on the class path.

The `SimpleLog` uitility writes trace to the `System.err` file.

WSIF (Web Services Invocation Framework) messages

This topic contains a list of the WSIF run-time system messages, with details of what each message means.

WebSphere system messages are logged from a variety of sources, including application server components and applications. Messages logged by application server components and associated IBM products start with a unique message identifier that indicates the component or application that issued the message.

WSIF0001E: An extension registry was not found for the element type "{0}"

Explanation: Parameters: {0} element type. No extension registry was found for the element type specified.

User Response: Add the appropriate extension registry to the port factory in your code.

WSIF0002E: A failure occurred in loading WSDL from "{0}"

Explanation: Parameters: {0} location of the WSDL file. The WSDL file could not be found at the location specified or did not parse correctly

User Response: Check that the location of the WSDL file is correct. Check that any network connections required are available. Check that the WSDL file contains valid WSDL.

WSIF0003W: An error occurred finding pluggable providers: {0}

Explanation: Parameters: {0} specific details about the error. There was a problem locating a WSIF pluggable provider using the J2SE 1.3 JAR file extensions to support service providers architecture. The WSIF trace file will contain the full exception details.

User Response: Verify that a META-INF/services/org.apache.wsif.spi.WSIFProvider file exists in a provider jar, that each class referenced in the META-INF file exists in the class path, and

that each class implements org.apache.wsif.spi.WSIFProvider. The class in error will be ignored and WSIF will continue locating other pluggable providers.

WSIF0004E: WSDL contains an operation type "{0}" which is not supported for "{1}" **Explanation:** Parameters: {0} name of the operation type specified. {1} name of the portType for the operation. An operation type which is not supported has been specified in the WSDL.

User Response: Remove any operations of the unsupported type from the WSDL. If the operation is required then make sure all messages have been correctly specified for the operation.

WSIF0005E: An error occurred when invoking the method "{1}" . ("{0}")

Explanation: Parameters: {0} name of communication type. For example EJB or Apache SOAP. {1} name of the method that failed. An error was encountered when invoking a method on the Web service using the communication shown in brackets.

User Response: Check that the method exists on the Web service and that the correct parts have been added to the operation as described in the WSDL. Network problems might be a cause if the method is remote and so check any required connections.

WSIF0006W: Multiple WSIFProvider found supporting the same namespace URI "{0}" . Found ("{1}")

Explanation: Parameters: {0} the namespace URI. {1} a list of the WSIFProvider found.. There are multiple org.apache.wsif.spi.WSIFProvider classes in the service provider path that support the same namespace URI.

User Response: A following WSIF0007I message will be issued notifying which WSIPFProvider will be used. Which WSIFProvider is chosen is based on settings in the wsif.properties file, or if not defined in the properties, the last WSIFProvider found will be used. See the wsif.properties file for more details on how to define which provider should be used to support a namespace URI.

WSIF0007I: Using WSIFProvider "{0}" for namespaceURI "{1}"

Explanation: Parameters: {0} the classname of the WSIFProvider being used. {1} the namespaceURI the provider will be used to support.. Either a previous WSIF0006W message has been issued or the SetDynamicWSIFProvider method has been used to override the provider used to support a namespaceURI.

User Response: None. See also WSIF0006W.

WSIF0008W: WSIFDefaultCorrelationService removing correlator due to timeout. ID:"{0}"

Explanation: Parameters: {0} the ID of the correlator being removed from the correlation service. A stored correlator is being removed from the correlation service due to its timeout expiring.

User Response: Determine why no response has been received for the asynchronous request within the timeout period. The wsif.asyncrequest.timeout property of the wsif.properties file defines the length of the timeout period.

WSIF0009I: Using correlation service - "{0}"

Explanation: Parameters: {0} the name of the correlation service being used. This identifies the name of the correlation service that will be used to process asynchronous requests.

User Response: None. If a correlation service other than the default WSIF supplied one is required, ensure that it is correctly registered in the JNDI java:comp/wsif/WSIFCorrelationService namespace.

WSIF0010E: Exception thrown while processing asynchronous response - "{0}"

Explanation: Parameters: {0} the error message string of the exception. While processing the response from an executeRequestResponseAsync call an exception was thrown.

User Response: Use the exception error message string to determine the cause of the error. The WSIF trace will have more details on the error including the exception stack trace.

WSIF0011I: Preferred port "{0}" was not available

Explanation: Parameters: {0} the user's preferred port. The preferred port set by the user on org.apache.wsif.WSIFService is not available

User Response: None unless this message appears for long periods of time in which case the user might want to pick a different port as their preferred port.

WSIF - Known restrictions

This topic lists the main known restrictions that apply when using WSIF.

Threading

WSIF is not thread-safe.

External Standards

WSIF supports:

- SOAP Version 1.1 (not 1.2 or later).
- WSDL Version 1.1 (not 1.2 or later).

WSIF does not provide WS-I compliance, and it does not support the Java API for XML-based Remote Procedure Calls (JAX-RPC) Version 1.1 (or later).

Full schema parsing

WSIF does not support full schema parsing. For example, WSDL references in complex types in the schema are not handled, and attributes are not handled.

XML Schema "redefine" elements are not handled and are ignored.

SOAP WSIF does not support:

- SOAP headers that are passed as `<parts>`.
- Unreferenced attachments in SOAP responses.
- Document Encoded style SOAP messages.

Note: This is not primarily a WSIF restriction. Although you can specify Document Encoded style in WSDL, it is not generally considered to be a valid option and is not supported by the Web Services Interoperability Organization (WS-I).

SOAP provider interoperability

The current WSIF default SOAP provider (the IBM Web Service SOAP provider) does not fully interoperate with services that are running on the former (Apache SOAP) provider. This restriction is due to the fact that the IBM Web Service SOAP provider is designed to interoperate fully with a JAX-RPC compliant Web service, and Apache SOAP cannot provide such a service. For information on how to overcome this restriction, see WSIF SOAP provider: working with legacy applications.

WSIF's support for SOAP faults is restricted to SOAP faults originating from a Web service that runs using the IBM Web Service SOAP provider.

Note: This is not primarily a WSIF restriction. The current SOAP faults specification does not prescribe how to encode a SOAP fault so that it maps to a Java exception. Consequently, each Web service run-time environment currently decides on its own SOAP fault format. The IBM Web Service SOAP provider can understand its own response SOAP faults, but not the SOAP faults from another provider.

Type mappings

The current WSIF default SOAP provider (the IBM Web Service SOAP provider) conforms to the JAX-RPC type mapping rules that were finalized after the former (Apache SOAP) provider was created. The majority of types are mapped the same way by both providers. The exceptions are: `xsd:date`, `xsd:dateTime`, `xsd:hexBinary` and `xsd:QName`. Both client and service need to use the same mapping rules if any of these four types are used. Below is a table detailing the mapping rules for these four types:

XML Data Type	Apache SOAP Java Mapping	JAX-RPC Java Mapping
xsd:date	java.util.Date	Not supported
xsd:dateTime	Not supported	java.util.Calendar
xsd:hexBinary	Hexadecimal string	byte [ ]
xsd:QName	org.apache.soap.util.xml.QName	javax.xml.namespace.QName

Arrays and complex types

WSIF does not support general complex types, it only handles complex types that map to Java Beans. To use schema complex types, you must write your own custom serializers. The specific complex type and array support for WSIF outbound invocation of Web services is as follows:

- WSIF supports Java classes generated by WebSphere Studio Application Developer - Integration Edition (WSAD-IE) message generators (the normal case when WSDL files are downloaded from somewhere else). The WSAD-IE-based generation happens automatically when you use the BPEL editor, or the generation actions available on the Enterprise Services context menu, or the Business Integration toolbar.
- WSIF does not support Java beans generated by other tools, including the base WSAD tool.
- For WSAD-IE generated Java beans, attributes defined in the WSDL do not work. That is to say that these attributes, although they appear in the Java beans generated to represent the complex type, do not appear in the SOAP request created by WSIF.
- WSIF does not support arrays when they are a field of a Java bean. That is to say, WSIF only supports an array that is passed in as a named `<part>`. If an array is wrappered inside a Java bean, the array is not serialized in the same way.

Object Serialization

WSIF does not support serialization of objects across different releases.

Asynchronous invocation

WSIF supports synchronous invocation for all providers. For the JMS and the SOAP over JMS providers, WSIF also supports asynchronous invocation. You should call the `supportsAsync()` method before trying to execute an asynchronous operation.

The EJB provider
The target service of the WSIF EJB provider must be a remote-home interface, it cannot be an EJB local-home interface. In addition, the EJB stub classes must be available on the client class path.

Running outside WebSphere Application Server
WSIF is not supported for use outside WebSphere Application Server.

UDDI Registry troubleshooting

When the IBM WebSphere UDDI Registry is running, it might issue messages to report events or errors. You can use these messages, described in Messages as your first aid to problem determination. If you need more details about the causes of a problem, you can turn on tracing for UDDI, as described in:
- Turning on UDDI Registry trace

Some errors that you might encounter when setting up or using the UDDI Registry, and their causes, are described in Common Causes of Errors in the UDDI Registry

Turning on UDDI Registry trace

You can enable tracing of the UDDI Registry at several levels of granularity. Please refer to Enabling tracing and logging for more information about using the administrative console to enable or disable trace; the component for the UDDI Registry is `com.ibm.uddi`. For example, to enable all UDDI Registry tracing, specify

```
com.ibm.uddi.*=all
```

Common causes of errors in the UDDI Registry

Below are a few of the common causes of errors that might be found and their suggested solutions.
- The first start of the UDDI application may take some time to complete:
  - When you start the UDDI Registry application for the first time with a new UDDI Registry database, it must perform UDDI initialization, which occurs automatically for a default UDDI node, or when requested for a customized UDDI node. UDDI initialization populates the UDDI Registry with pre-defined data and entities, and can therefore take some time to complete. This is expected behavior. Note that this only occurs on the first start, not subsequent starts, of the UDDI application.
  - If you use WSADMIN to issue a command to start the UDDI application, then depending on your TCP timeout settings, this request might time out while waiting for UDDI to complete initialization. The UDDI initialization and the starting of the UDDI application will continue unaffected by this timeout, and will complete normally.
- There is a limitation concerning URL rewriting causing JavaScript syntax errors on several Web pages in the UDDI User Console. Because of this, cookies must be enabled in client browsers, the application server must have cookies enabled as the session tracking mechanism, and URL rewriting must be disabled.
- If you run the uddiDeploy.jacl with the default option, and you have not first deleted the UDDI Cloudscape database (from the databases subdirectory of your server profile) the UDDI database will not be recreated, and you will get an error message. This error can be ignored if you did not intend to replace the Cloudscape database.
- If attempting to use a remote DB2 database and you are experiencing problems attaching to the remote system, one of the possible causes might be IP

addressing. You should not have this problem if the remote system is using a static IP address. If, however, the remote system is using DHCP, the two systems must be aware of each others subnet mask.

- If you cannot see your UDDI node in the administrative console list of available nodes, check that the UDDI application is started on the relevant WebSphere node/server.
- If you are unable to issue UDDI requests; for example, you start the UDDI user console and get errors when trying to publish or inquire, it is possible that:
  1. the database is not currently loaded or configured. Check the output from creating the database.
  2. the database is not correctly configured. Check the JDBC provider and datasource definitions are correct. This can be validated by using the Test Connection button on the administrative console.
  3. the UDDI node is not initialized. Check the UDDI node page on the administrative console. If the entry for the node in question does not show as activated, or deactivated, try to initialize the node having set any policies or properties first.
  4. the UDDI node is currently deactivated, while UDDI runtime settings are being updated. Check the UDDI node page on the administrative console. If the entry for the node in question shows as deactivated, then wait for it to change to activated, and then retry the request.
  5. there are problems with your DB2 setup. In particular, check that the DB2 bind utility has been run and that a TEMP database has been defined in the DB2 location. See Using a DB2 Universal JDBC Driver Provider with WebSphere Application Server for z/OS for more information.
- Ensure that the database name specified when defining the data source is the location name and not the database name supplied to the createddl.sh script (see Creating a DB2 database). For example 'LOC1' not 'UDDI30'.

Reporting problems with the UDDI Registry

If you report a problem with the IBM WebSphere UDDI Registry component to IBM, supply the following information:

1. A detailed description of the problem. For example, if you were using a UDDI client, describe the client error seen and the type of client program, and provide details of the request issued. If you were using the UDDI user interface, describe the exact sequence you followed, and the result.
2. The build date and time of the version you are using. This can be obtained as follows:
   - In the *install_root*/profiles/*profile_name*/installedApps/*cell_name* subdirectory of the WebSphere installation location, you will find a subdirectory called UDDI_Registry.*node_name*.*server_name*.ear, where *node_name* is the name of the node into which the UDDI Registry application is installed, and *server_name* is the name of the server. Within that subdirectory, you will find a file called version.txt. Include the contents of this file as part of your information.
   - If the UDDI Registry has been started with tracing enabled for the UDDI component, you should find a trace entry in the WebSphere trace log that includes the strings "getUDDIMessageLogger" and "UDDI Build :" followed by the build date and time, and the build system. Also include this information.
3. Other environment information, such as:
   - Platform.
   - Database product.
   - Whether the database is local or remote, or for Cloudscape, whether it is embedded or networked.

- Whether the server is in a standalone or a network deployment configuration.
- If the server is in a network deployment configuration, whether it is in a cluster or not.
- Whether the UDDI node is default or customized.
- The language used, for example, Chinese.

4. Any relevant log files and trace files.
   - If the problem occurred while setting up and installing the UDDI Registry application using the setup script uddiDeploy.jacl, supply the log output from running the script. (If you did not redirect the output from the script file to a log file, rerun the script, this time redirecting the output as described in the section Setting up a customized UDDI node or Setting up a default UDDI node.) The log file is written to the directory from which you ran the setup script.
   - If the problem occurred while removing the UDDI Registry application using the remove script, uddiRemove.jacl, supply the log output from running the script. (If you did not redirect the output from the script file to a log file, rerun the script, this time redirecting the output as described in the section Removing a UDDI Registry node.) The log file is written to the directory from which you ran the remove script.
   - If the problem occurred while running the UDDI Registry, enable UDDI tracing (if not already enabled) and supply the trace log from the logs directory of the application server on which the UDDI Registry was running. See Turning on UDDI Trace for details on how to enable UDDI tracing. The Trace string that is most useful for initial analysis of problems by IBM is ″com.ibm.uddi.*=all″.
   - Also supply the WebSphere log files system.out and system.err.
   - Supply details of the version of IBM WebSphere Application Server you are running by executing the command `versioninfo.sh` on the application server node and directing the output to a log file.
5. If appropriate, any application code that you are using and the output produced by the application code.
6. UDDI utilizes First Failure Data Capture (FFDC) to capture data for unexpected UDDI errors. The FFDC data is saved in *install_root*/profiles/*profile_name*/logs/ffdc. Package all the files in this directory and send the package to IBM.

Data access resources

JDBC and data source troubleshooting tips

To see whether your specific problem has been addressed, review the Cannot access a data source topic.

Trace strings for JDBC data sources

Turn on JDBC tracing by using the following trace strings:

- **com.ibm.ws.database.logwriter** Trace string for databases that use the GenericDataStoreHelper. You can also use this trace string for unsupported databases.
- **com.ibm.ws.db2.logwriter** Trace string for DB2 databases.
- **com.ibm.ws.oracle.logwriter** Trace string for Oracle databases.
- **com.ibm.ws.cloudscape.logwriter** Trace string for Cloudscape databases.
- **com.ibm.ws.informix.logwriter** Trace string for Informix databases.
- **com.ibm.ws.sqlserver.logwriter** Trace string for Microsoft SQL Server databases.
- **com.ibm.ws.sybase.logwriter** Trace string for Sybase databases.

The trace group that includes the trace strings is WAS.database.

JDBC trace properties

Use a back-end database that supports JDBC tracing. Setting trace strings does not result in a trace if the database does not support JDBC tracing. The following databases offer JDBC tracing at this time:
- DB2
- Oracle
- SQL Server

Set the level of trace desired for DB2 Universal database and Oracle as custom properties on the datasource.
- **DB2 Universal JDBC driver provider** Custom properties for DB2 are:
  - **traceLevel** Possible traceLevel values are:
    - TRACE_NONE = 0
    - TRACE_CONNECTION_CALLS = 1
    - TRACE_STATEMENT_CALLS = 2
    - TRACE_RESULT_SET_CALLS = 4
    - TRACE_DRIVER_CONFIGURATION = 16
    - TRACE_CONNECTS = 32
    - TRACE_DRDA_FLOWS = 64
    - TRACE_RESULT_SET_META_DATA = 128
    - TRACE_PARAMETER_META_DATA = 256
    - TRACE_DIAGNOSTICS = 512
    - TRACE_SQLJ = 1024
    - TRACE_ALL = -1

    Note: This trace level provides real data that sets to the PreparedStatement or gets from the ResultSet object.
  - **traceFile** Use this property to integrate the DB2 trace with the WebSphere Application Server trace. If you do not set the value, traces are integrated. Otherwise, DB2 traces are directed to the desired file. You can dynamically enable or disable trace. You can run an application and turn on the DB2 trace if there is a problem. Use the run time trace enablement provided with the Application Server by specifying a trace string of com.ibm.ws.db2.logwriter=all=enabled.
- **Oracle JDBC provider** Custom properties for Oracle are:
  - **oraclelogCategoryMask** Controls the output category. The default is 47, which is (OracleLog.USER_OPER 1 | OracleLog.PROG_ERROR 2 | OracleLog.ERROR 4 | OracleLog.WARNING 8 | OracleLog.DEBUG1 32).

    Possible values are:
    - OracleLog.USER_OPER 1
    - OracleLog.PROG_ERROR 2
    - OracleLog.ERROR 4
    - OracleLog.WARNING 8
    - OracleLog.FUNCTION 16
    - OracleLog.DEBUG1 32
    - OracleLog.SQL_STR 128
  - **oraclelogModuleMask** Controls which modules write debug output. The default is 1, which is OracleLog.MODULE_DRIVER 1.

    Possible values are:
    - OracleLog.MODULE_DRIVER 1,
    - OracleLog.MODULE_DBACCESS 2

– **oraclelogPrintMask** Controls which information to print with each trace message. The default is 62, which is ([OracleLog.FIELD_OBJECT for 9i / OracleLog.FIELD_CONN for 8i] 32 | OracleLog.FIELD_CATEGORY 16 | OracleLog.FIELD_SUBMOD 8 | OracleLog.FIELD_MODULE 4 | OracleLog.FIELD_TIME 2).

  Possible values are:
  - OracleLog.FIELD_TIME 2
  - OracleLog.FIELD_MODULE 4
  - OracleLog.FIELD_SUBMOD 8
  - OracleLog.FIELD_CATEGORY 16
  - OracleLog.FIELD_OBJECT 32
  - OracleLog.FIELD_THREAD 64

Notes for Oracle JDBC tracing:

1. Oracle 9i requires the use of `classes12_g.zip` to display traces. With Oracle8i, the `classes12_g.zip` is optional.
2. You can dynamically enable or disable trace. You can run an application and turn on the Oracle trace if there is a problem. Use the run-time trace enablement provided with the WebSphere Application Server products, by specifying a trace string of com.ibm.ws.oracle.logwriter=all=enabled.

For current information available from IBM Support on known problems and their resolution, see the IBM Support page.

IBM Support has documents that can save you time gathering information needed to resolve this problem. Before opening a PMR, see the IBM Support page.

Related tasks

Chapter 6, “Troubleshooting by task,” on page 233

Related reference

Troubleshooting installation problems

“Cannot access a data source” on page 24

Cannot access a data source

What kind of database are you trying to access?

- Oracle
- DB2
- SQL Server
- Cloudscape
- Sybase
- General data access problems

If the errors described in the previous articles do not match the errors you see:

1. Browse the log files of the application server for clues.
2. Browse the Helper Class property of the data source to verify that it is correct and that it is on the WebSphere Application Server class path. Mysterious errors or behavior might result from a missing or misnamed Helper Class name. If WebSphere Application Server cannot load the specified class, it uses a default helper class that might not function correctly with your database manager.
3. Verify that the Java Naming and Directory Interface (JNDI) name of the data source matches the name used by the client attempting to access it. If error messages indicate that the problem might be naming-related, such as referring

to the **name server** or **naming service**, or including error IDs beginning with **NMSV**, look at the Naming related problems and Troubleshooting the naming service component topics.
4. Enable tracing for the resource adapter using the trace specification, RRA=all=enabled. Follow the instructions for dumping and browsing the trace output, to narrow the origin of the problem.

If none of these steps fixes your problem, see if the problem is identified and documented in available online support (hints and tips, technotes, and fixes). If none of the online resources listed in the topic describes your problem, see "Obtaining help from IBM" on page 228.

General data access problems
- An exception ″IllegalConnectionUseException″ occurs
- WTRN0062E: An illegal attempt to enlist multiple one phase capable resources has occurred.
- ConnectionWaitTimeoutException.
- com.ibm.websphere.ce.cm.StaleConnectionException: [IBM][CLI Driver] SQL1013N The database alias name or database name ″NULL″ could not be found. SQLSTATE=42705
- java.sql.SQLException: java.lang.UnsatisfiedLinkError:
- ″J2CA0030E: Method enlist caught java.lang.IllegalStateException″ wrapped in error ″WTRN0063E: An illegal attempt to enlist a one phase capable resource with existing two phase capable resources has occurred″ when attempting to execute a transaction.
- java.lang.UnsatisfiedLinkError:xaConnect exception when attempting a database operation
- ″J2CA0114W: No container-managed authentication alias found for connection factory or datasource *datasource*″ when attempting a database operation
- An error is thrown if you use the ws_ant command to perform the database customization for Structured Query Language in Java on HP platforms

IllegalConnectionUseException

This error can occur because a connection obtained from a WAS40DataSource is being used on more than one thread. This usage violates the J2EE 1.3 programming model, and an exception generates when it is detected on the server. This problem occurs for users accessing a data source through servlets or bean-managed persistence (BMP) enterprise beans.

To confirm this problem, examine the code for connection sharing. Code can inadvertently cause sharing by not following the programming model recommendations, for example by storing a connection in an instance variable in a servlet, which can cause use of the connection on multiple threads at the same time.

WTRN0062E: An illegal attempt to enlist multiple one phase capable resources has occurred

This error can occur because:
- An attempt was made to share a single-phase connection, when each **getConnection** method has different connection properties; such as the AccessIntent. This attempt causes a non-shareable connection to be created.
- An attempt was made to have more than one unshareable connection participate in a global transaction, when the data source is not an XA resource.

- An attempt was made to have a one-phase resource participate in a global transaction while an XA resource or another one-phase resource already participated in this global transaction.
  - Within the scope of a global transaction you try to get a connection more than once and at least one of the resource-refs you use specifies that the connection is unshareable, and the data source is not configured to support two-phase commit transactions. It does not support an XAResource. If you do not use a resource-ref, you default to unshareable connections.
  - Within the scope of a global transaction you try to get a connection more than once and at least one of the resource-refs you use specifies that the connection is shareable and the data source is not configured to support two-phase commit transactions. That is, it does not support an XAResource. In addition, even though you specify that connections are shareable, each getConnection request is made with different connection properties (such as IsolationLevel or AccessIntent). In this case, the connections are not shareable, and multiple connections are handed back.
  - Multiple components (servlets, session beans, BMP entity beans, or CMP entity beans) are accessed within a global transaction. All use the same data source, all specify shareable connections on their resource-refs, and you expect them to all share the same connection. If the properties are different, you get multiple connections. AccessIntent settings on CMP beans change their properties. To share a connection, the AccessIntent setting must be the same. For more information about CMP beans sharing a connection with non-CMP components, see the *Data access application programming interface support* and *Example: Accessing data using IBM extended APIs to share connections between container-managed and bean-managed persistence beans* topics in the DataAccess section of the information center.

To correct this error:
- Check what your client code passes in with its getConnection requests, to ensure they are consistent with each other.
- Check the connection sharing scope from the resource binding, using anassembly tool.
  - If you are running an unshareable connection scope, verify that your data source is an XA data source.
  - If you are running a shareable connection scope, verify that all connection properties, including AccessIntent, are sharable.
- Check the JDBC provider implementation class from the Manage JDBC resource panel of the administrative console to ensure that it is a class that supports XA-type transactions.

ConnectionWaitTimeoutException accessing a data source or resource adapter

If your application receives exceptions like a com.ibm.websphere.ce.cm.ConnectionWaitTimeoutException or com.ibm.websphere.ce.j2c.ConnectionWaitTimeoutException when attempting to access a WebSphere Application Server data source or JCA-compliant resource adapter, respectively, some possible causes are:
- The maximum number of connections for a given pool is set too low. The demand for concurrent use of connections is greater then the configured maximum value for the connection pool. One indication that this situation is the problem is that you receive these exceptions regularly, but your CPU utilization is not high. This exception indicates that there are too few connections available to keep the threads in the server busy.
- Connection Wait Time is set too low. Current demand for connections is high enough such that sometimes there is not an available connection for short

periods of time. If your connection wait timeout value is too low, you might timeout shortly before a user returns a connection back to the pool. Adjusting the connection wait time can give you some relief. One indication of this problem is that you use close to the maximum number of connections for an extended period and receiving this error regularly.
- You are not closing some connections or you are returning connections back to the pool at a very slow rate. This situation can happen when using unshareable connections, when you forget to close them, or you close them long after you are finished using them, keeping the connection from returning to the pool for reuse. The pool soon becomes empty and all applications get ConnectionWaitTimeoutExceptions. One indication of this problem is you run out of connections in the connection pool and you receive this error on most requests.
- You are driving more load than the server or backend system have resources to handle. In this case you must determine which resources you need more of and upgrade configurations or hardware to address the need. One indication of this problem is that the application or database server CPU is nearly 100% busy.

To correct these problems, either:
- Modify an application to use fewer connections
- Properly close the connections.
- Change the pool settings of MaxConnections or ConnnectionWaitTimeout.
- Adjust resources and their configurations.

com.ibm.websphere.ce.cm.StaleConnectionException: [IBM][CLI Driver] SQL1013N The database alias name or database name "NULL" could not be found. SQLSTATE=42705

This error occurs when a data source is defined but the **databaseName** attribute and the corresponding value are not added to the custom properties panel.

To add the **databaseName** property:
1. Click **Resources>Manage JDBC Providers** link in the administrative console.
2. Select the JDBC provider that supports the problem data source.
3. Select **Data Sources** and then select the problem data source.
4. Under **Additional properties** click **Custom Properties**.
5. Select the **databaseName** property, or add one if it does not exist, and enter the actual database name as the value.
6. Click **Apply** or **OK**, and then click **Save** from the action bar.
7. Access the data source again.

java.sql.SQLException: java.lang.UnsatisfiedLinkError:

This error indicates that the directory containing the binary libraries which support a database are not included in the LIBPATH environment variable for the environment in which the WebSphere Application Server starts.

The path containing the DBM vendor libraries vary by dbm. One way to find them is by scanning for the missing library specified in the error message. Then you can correct the LIBPATH variable to include the missing directory, either in the `.profile` of the account from which WebSphere Application Server is executed, or by adding a statement in a `.sh` file which then executes the `startServer` program.

"J2CA0030E: Method enlist caught java.lang.IllegalStateException" wrapped in error "WTRN0063E: An illegal attempt to enlist a one phase capable resource with existing two phase capable resources has occurred" when attempting to execute a transaction.

This error can occur when last participant support is missing or disabled. last participant support allows a one-phase capable resource and a two-phase capable resource to enlist within the same transaction.

Last participant support is only available if the following are true:
- WebSphere Application Server Programming Model Extensions (PME) is installed. PME is included in the Application Server Integration Server product.
- The Additional Integration Server Extensions option is enabled when PME is installed. If you perform a typical installation, this function is enabled by default. If you perform a custom installation, you have the option to disable this function, which disables last participant support.
- The application enlisting the one-phase resource is deployed with the **Accept heuristic hazard** option enabled. This deployment is done with an assembly tool such as the Application Server Toolkit (AST) or Rational Web Developer.

java.lang.UnsatisfiedLinkError:xaConnect exception when attempting a database operation

This problem has two main causes:
- The most common cause is that the jdbc driver which supports connectivity to the database is missing, or is not the correct version, or that native libraries which support the driver are on the system's path.
  - To resolve this problem on a Windows platform, verify that the JDBC driver jar file is on the system PATH environment variable:
    - If you are using DB2,verify that at least the DB2 client product has been installed on the WebSphere host
      - On DB2 version 7.2 or earlier, the file where the client product is installed on the WebSphere Application Server is db2java.zip. Verify that the usejdbc2.bat program has been executed after the database install and after any upgrade to the database product.
      - On DB2 version 8.1 or later, use the DB2 Universal JDBC Provider Driver when defining a JDBC provider in WebSphere Application Server. The driver file is db2jcc.jar. If you use the type 2 (default) option, verify that at least the DB2 client product is installed on the WebSphere Application Server host. If you specify the type 4 option, the DB2 client does not need to be installed, but the file db2jcc.jar still must be present.

        When specifying the location of the driver file, it is recommended to that you specify the path and file name of the target DB2 installation, rather than simply copying the file to a local directory, if possible. Otherwise, you may be exposed to problems if the target DB2 installation is upgraded and the driver used by WebSphere Application Server is not. If you choose **DB2 Legacy CLI-based type 2 JDBC Driver** when defining a JDBC provider in WebSphere Application Server, then you must still follow the steps to ensure that you have the correct version of the db2java.zip file (see instructions for DB2 7.2 or earlier).
  - On a Unix platform, ensure that any native libraries required to support the database client of your database product are specified in the LD_LIBRARY_PATH environment variable in the profile of the account under which WebSphere Application Server executes.

If you are using DB2 The native library is libdb2jdbc.so. The best way to ensure that this library is accessed correctly by WebSphere is to call the db2profile script supplied with DB2 from the .profile script of the account (such as ″root″) under which WebSphere runs.

- If you are using DB2 version 7.2 or earlier, ensure that the usejdbc2,script provided with DB2 is called from the profile of the account under which WebSphere Application server is launched.
- If you are using DB2 version 8.1 or later, see the previous instructions for the Windows operating system.

- If the database manager is DB2, you may have chosen the option to create a 64-bit instance. A 64-bit configuration is not supported. If this has happened, remove the database instance and create a new one with the default 32-bit setting.

  Note: For general help in configuring JDBC drivers and data sources in WebSphere Application Server, see the topic Accessing data from applications.

"J2CA0114W: No container-managed authentication alias found for connection factory or datasource *datasource*" when attempting a database operation

This error might occur in the SystemOut.log file when you run an application to access a data source after creating the data source using JACL script.

The error message occurs because the JACL script did not set container-managed authentication alias for CMP connection factory. The JACL is missing the following line:

```
$AdminConfig create MappingModule $cmpConnectorFactory  "{mappingConfigAlias
 DefaultPrincipalMapping} {authDataAlias $authDataAlias}
```

To correct this problem, add the missing line to the JACL script and run the script again. See Example: Creating a JDBC provider and data source using Java Management Extensions API and the scripting tool for a sample JACL script.

An error is thrown if you use the ws_ant command to perform the database customization for Structured Query Language in Java on HP platforms

If you use the `ws_ant` command to perform the database customization for Structured Query Language in Java (SQLJ) on HP platforms, you can receive an error similar to the following:

```
[java] [ibm][db2][jcc][sqlj]
[java] [ibm][db2][jcc][sqlj] Begin Customization
[java] [ibm][db2][jcc][sqlj] encoding not supported!!
```

The cause of this error might be that your databases were created using the HP default character set. The Java Common Client (JCC) driver depends on the software development kit (SDK) to perform the codepage conversions. The SDK shipped with this product, however, does not support the HP default codepage.

You need to set your LANG to the ISO locale before creating the databases. It should be similar to the following:

```
export LANG=en_US.iso88591
```

Refer to the DB2 Tech Notes at http://www-3.ibm.com/software/data/db2/udb/ad/v8/bldg/t0004877.htm for details.

Related tasks

Chapter 6, "Troubleshooting by task," on page 233

"Troubleshooting by component" on page 5

Example: Accessing data using IBM extended APIs to share connections between container-managed and bean-managed persistence beans

Related reference

Extensions to data access APIs

Problems accessing an Oracle data source

What kind of error do you see when you try to access your Oracle-based data source

- An Invalid Oracle URL is specified
- ″DSRA0080E: An exception was received by the Data Store Adapter. See original exception message: ORA-00600″ when connecting to or using an Oracle data source.
- ″DSRA8100E: Unable to get a {0} from the DataSource. Explanation: See the linkedException for more information.″
- An ″Error while trying to retrieve text for error″ message occurs when connecting to an Oracle data source.
- A java.lang.UnsatisfiedLinkError occurs when connecting to an Oracle data source.
- The exception java.lang.NullPointerException or an ″internal error: oracle.jdbc.oci8.OCIEnv″ occurs when connecting to an Oracle data source.
- WSVR0016W: Classpath entry, ${ORACLE_JDBC_DRIVER_PATH}/classes12.zip, in Resource, Oracle JDBC Thin Driver, located at cells/BaseApplicationServerCell/nodes/wasrtp/resources.xml has an invalid variable.

An invalid Oracle URL is specified

This error might be caused by an incorrectly specified URL on the URL property of the target data source.

Examine the URL property for the data source object in the administrative console. For the 8i OCI driver, verify that **oci8** is used in the URL. For the 9i OCI driver, you can use either **oci8** or **oci**.

Examples of Oracle URLs:

- For the thin driver: `jdbc:oracle:thin:@hostname.rchland.ibm.com:1521:IBM`
- For the thick (OCI) driver: `jdbc:oracle:oci8:@tnsname1`

"DSRA0080E: An exception was received by the data store adapter. See original exception message: ORA-00600" when connecting to or using an Oracle data source "DSRA0080E: An exception was received by the data store adapter. See original exception message: ORA-00600" when connecting to or using an Oracle data source "DSRA0080E: An exception was received by the Data Store Adapter. See original exception message: ORA-00600" when connecting to or using an Oracle data source

A possible reason for this exception is that the version of the Oracle JDBC driver being used is older than the Oracle database. It is possible that more than one version of the Oracle JDBC driver is configured on the WebSphere Application Server.

Examine the version of the JDBC driver. Sometimes you can determine the version by looking at the class path to determine what directory the driver is in.

If you cannot determine the version this way, use the following program to determine the version. Before running the program, set the class path to the location of your JDBC driver files.

```
import java.sql.*;
     import oracle.jdbc.driver.*;
     class JDBCVersion
     {
         public static void main (String args[])
         throws SQLException
         {
             // Load the Oracle JDBC driver
             DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());
             // Get a connection to a database
             Connection conn = DriverManager.getConnection
             ("jdbc:oracle:thin:@appaloosa:1521:app1","sys","change_on_install");
             // Create Oracle DatabaseMetaData object
             DatabaseMetaData meta = conn.getMetaData();
             // gets driver info:
             System.out.println("JDBC driver version is " + meta.getDriverVersion());
         }
     }
```

If the driver and the database are at different versions, replace the JDBC driver with the correct version. If multiple drivers are configured, remove any that occur at the incorrect level.

DSRA8100E: Unable to get a {0} from the DataSource. Explanation: See the linkedException for more information.

When using an oracle thin driver, Oracle throws a ″java.sql.SQLException: invalid arguments in call″ error if no user name or password is specified when getting a connection. If you see this error while running WebSphere Application Server, the alias is not set.

To remove the exception, define the alias on the data source.

"Error while trying to retrieve text for error" error when connecting to an Oracle data source

The most likely cause of this error is that the Oracle 8i OCI driver is being used with an ORACLE_HOME property that is either not set or is set incorrectly.

To correct the error, examine the user profile that WebSphere Application Server is running under to verify that the $ORACLE_HOME environment variable is set correctly.

"java.lang.UnsatisfiedLinkError:" connecting to an Oracle data source

The environment variable LIBPATH might not be set or is set incorrectly, if your data source throws an **UnsatisfiedLinkError** error, and the full exception indicates that the problem is related to an Oracle module, as in the following examples.

Example of invalid an LIBPATH for the 8i driver:

```
Exception in thread "main" java.lang.UnsatisfiedLinkError:
/usr/WebSphere/AppServer/java/jre/bin/libocijdbc8.so:
load ENOENT on shared library(s)
/usr/WebSphere/AppServer/java/jre/bin/libocijdbc8.so libclntsh.a
```

Example of an invalid LIBPATH for the 9i driver:

```
Exception in thread "main" java.lang.UnsatisfiedLinkError:
no ocijdbc9  (libocijdbc9.a or .so) in java.library.path
at java.lang.ClassLoader.loadLibrary(ClassLoader.java(Compiled Code))
at java.lang.Runtime.loadLibrary0(Runtime.java:780)
```

To correct the problem, examine the user profile under which the WebSphere Application Server is running to verify that the LIBPATH environment variable includes Oracle libraries. Scan for the `lobocijdbc8.so` file to find the right directory.

java.lang.NullPointerException referencing 8i classes, or " internal error: oracle.jdbc.oci8. OCIEnv" connecting to an Oracle data source

The problem might be that the 9i OCI driver is being used on an AIX 32-bit machine, the LIBPATH is set correctly, but the ORACLE_HOME environment variable is not set or is set incorrectly. You can encounter an exception similar to either of the following when your application attempts to connect to an Oracle data source:

Exception example for the java.lang.NullPointerException:

```
Exception in thread "main" java.lang.NullPointerException
at oracle.jdbc.oci8.OCIDBAccess.check_error(OCIDBAccess.java:1743)
at oracle.jdbc.oci8.OCIEnv.getEnvHandle(OCIEnv.java:69)
at oracle.jdbc.oci8.OCIDBAccess.logon(OCIDBAccess.java:452)
at oracle.jdbc.driver.OracleConnection. <init>(OracleConnection.java:287)
```

Exception example for the java.sql.SQLException:

```
Exception in thread "main" java.sql.SQLException:
internal error: oracle.jdbc.oci8. OCIEnv@568b1d21
at oracle.jdbc.dbaccess.DBError.throwSqlException(DBError.java:184)
at oracle.jdbc.dbaccess.DBError.throwSqlException(DBError.java:226)
at oracle.jdbc.oci8.OCIEnv.getEnvHandle(OCIEnv.java:79)
```

To correct the problem, examine the user profile that WebSphere Application Server is running under to verify that it has the `$ORACLE_HOME` environment variable set correctly, and that the `$LIBPATH` includes `$ORACLE_HOME/lib`.

WSVR0016W: Classpath entry, ${ORACLE_JDBC_DRIVER_PATH}/classes12.zip, in Resource, Oracle JDBC Thin Driver, located at cells/BaseApplicationServerCell/nodes/wasrtp/resources.xml has an invalid variable

This error occurs when there is no environment variable defined for the property, ORACLE_JDBC_DRIVER_PATH.

Verify this problem in the administrative console. Go to **Environment > Manage WebSphere Variables** to verify whether the variable `ORACLE_JDBC_DRIVER_PATH` is defined.

To correct the problem, click **New** and define the variable. For example, name : **ORACLE_JDBC_DRIVER_PATH** , value : **c:\oracle\jdbc\lib** Use a value that

names the directory in your operating system and directory structure that contains the `classes12.zip` file.

Related tasks

Chapter 6, “Troubleshooting by task,” on page 233

Related reference

“Cannot access a data source” on page 24

“Problems accessing a DB2 database” on page 32

“Problems accessing a SQL server data source” on page 39

“Problems accessing a Cloudscape database” on page 40

“Problems accessing a Sybase data source” on page 43

Problems accessing a DB2 database

What kind of problem are you having accessing your DB2 database?

- “SQL0567N "DB2ADMIN " is not a valid authorization ID. SQLSTATE=42602” on page 33
- “SQL0805N Package package-name was not found” on page 33
- “SQL0805N Package "NULLID.SQLLC300" was not found. SQLSTATE=51002” on page 33
- “SQL30082N Attempt to establish connection failed with security reason "17" ("UNSUPPORTED FUNCTION’) SQLSTATE=08001” on page 33
- “SQLException, with ErrorCode -99,999 and SQLState 58004, with Java "StaleConnectionException: COM.ibm.db2.jdbc.DB2Exception: [IBM][CLI Driver] CLI0119E Unexpected system failure. SQLSTATE=58004", when using WAS40-type data source” on page 34
- “Error message java.lang.reflect.InvocationTargetException: com.ibm.ws.exception.WsException: DSRA0023E: The DataSource implementation class "COM.ibm.db2.jdbc.DB2XADataSource" could not be found. when trying to access a DB2 database” on page 34
- “CLI0119E System error. SQLSTATE=58004 - DSRA8100 : Unable to get a XAconnection or DSRA0011E: Exception: COM.ibm.db2.jdbc.DB2Exception: [IBM][CLI Driver] CLI0119E Unexpected system failure. SQLSTATE=5800” on page 35
- “COM.ibm.db2.jdbc.DB2Exception: [IBM][CLI Driver][DB2/NT] SQL0911N The current transaction has been rolled back because of a deadlock or timeout. Reason code "2". SQLSTATE=40001” on page 35
- “"COM.ibm.db2.jdbc.DB2ConnectionPoolDataSource" could not be found for data source ([data-source-name])"” on page 36
- “Problems accessing a DB2 database” on page 32
- “ java.sql.SQLException: Failure in loading T2 native library db2jcct2 DSRA0010E: SQL State = null, Error Code = -99,999 ” on page 36
- Lock contention exception occurs in database when data source implementation type is XA
- “"DSRA8050W: Unable to find the DataStoreHelper class specified" exception occurs when trying to use a DB2 Universal Datasource in a mixed release cell.” on page 38
- “Receive "’SYSTEM’ is not a valid authorization ID" message when trying to access DB2 on a Windows machine where WebSphere Application Server is also installed.” on page 38
- Check the IBM support page for information on problems that occur when using connection pooling with DB2.

SQL0567N "DB2ADMIN " is not a valid authorization ID. SQLSTATE=42602

If you encounter this error when attempting to access a DB2 Universal Database (UDB):
1. Verify that your user name and password in the data source properties page in the administrative console are correct.
2. Ensure that the user ID and password do not contain blank characters before, in between, or after.

SQL0805N Package *package-name* **was not found**

Possible reasons for these exceptions:
- If the package name is `NULLID.SQLLC300`, see SQL0805N Package ″NULLID.SQLLC300″ was not found. SQLSTATE=51002. for the reason.
- You are attempting to use an XA-enabled JDBC driver on a DB2 database that is not XA-ready.

To correct the problem on a DB2 Universal Database (UDB), run this one-time procedure, using the db2cmd interface while connected to the database in question:
1. **DB2 bind @db2ubind.lst blocking all grant public**
2. **DB2 bind @db2cli.lst blocking all grant public**

The `db2ubind.lst` and `db2cli.lst` files are in the `bnd` directory of your DB2 installation root. Run the commands from that directory.

SQL0805N Package "NULLID.SQLLC300" was not found. SQLSTATE=51002

This error can occur because:
- The underlying database was dropped and recreated.
- DB2 was ugpraded and its packages are not rebound correctly.

To resolve this problem, rebind the DB2 packages by running the db2cli.lst script found in the bnd directory. For example: db2>@db2cli.lst.

SQL30082N Attempt to establish connection failed with security reason "17" ("UNSUPPORTED FUNCTION') SQLSTATE=08001

This error can occur when the security mechanism specified by the client is not valid for this server. Some typical examples:
- The client sent a new password value to a server that does not support the change password function.
- The client sent SERVER_ENCRYPT authentication information to a server that does not support password encryption.
- The client sent a userid, but no password, to a server that does not support authentication by userid only.
- The client has not specified an authentication type, and the server has not responded with a supported type. This can include the server returning multiple types from which the client is unable to choose.

To resolve this problem, ensure that your client and server use the same security mechanism. For example, if this is an error on your data source, verify that you have assigned a user id and password or authentication alias.

SQLException, with ErrorCode -99,999 and SQLState 58004, with Java "StaleConnectionException: COM.ibm.db2.jdbc.DB2Exception: [IBM][CLI Driver] CLI0119E Unexpected system failure. SQLSTATE=58004", when using WAS40-type data source

An unexpected system failure usually occurs when running in XA mode (two-phase commit). Among the many possible causes are:
- An invalid username or password was provided.
- The database name is incorrect.
- Some DB2 packages are corrupted.

To determine whether you have a user name or password problem, look in the `db2diag.log` file to view the actual error message and SQL code. A message like the following example, with an SQLCODE of -1403, indicates an invalid user ID or password:

```
2002-07-26-14.19.32.762905   Instance:db2inst1   Node:000
PID:9086(java)   Appid:*LOCAL.db2inst1.020726191932
XA DTP Support  sqlxa_open   Probe:101
DIA4701E Database "POLICY2" could not be opened
for distributed transaction processing.
String Title: XA Interface SQLCA  PID:9086 Node:000
SQLCODE = -1403
```

To resolve these problems:
1. Correct your user name and password. If you specify your password on the GUI for the data source, ensure that the user name and password you specify on the bean are correct. The user name and password you specify on the bean overwrite whatever you specify when creating the data source.
2. Use the correct database name.
3. Rebind the packages (in the `bnd` directory) as follows:

   ```
   db2connect to dbname
   c:\SQLLIB\bnd>DB2 bind @db2ubind.lst blocking all grant public
   c:\SQLLIB\bnd>DB2 bind @db2cli.lst blocking all grant public
   ```
4. Ensure that the \WebSphere\AppServer\properties\wsj2cdpm.properties file has the right user ID and password.

Error message `java.lang.reflect.InvocationTargetException: com.ibm.ws.exception.WsException: DSRA0023E: The DataSource implementation class "COM.ibm.db2.jdbc.DB2XADataSource" could not be found.` when trying to access a DB2 database

One possible reason for this exception is that a user is attempting to use a JDBC 2.0 DataSource, but DB2 is not JDBC 2.0-enabled. This situation frequently happens with new installations of DB2 because DB2 provides separate drivers for JDBC 1.X and 2.0, with the same physical file name. By default, the JDBC 1.X driver is on the class path.

To confirm this problem:
- On Windows systems, look for the inuse file in the java12 directory in your DB2 installation root. If the file missing, you are using the JDBC 1.x driver.
- On UNIX systems, check the class path for your data source. If the class path does not point to the `db2java.zip` file in the `java12` directory, you are using the JDBC 1.x driver.

To correct this problem:

- On Windows systems, stop DB2. Run the usejdbc2.bat file from the java12 directory in your DB2 instalation root. Run this file from a command line to verify that it completes successfully.
- On UNIX systems, change the class path for your data source to point to the `db2java.zip` file in the `java12` directory of your DB2 installation root.

CLI0119E System error. SQLSTATE=58004 - DSRA8100 : Unable to get a XAconnection or DSRA0011E: Exception: COM.ibm.db2.jdbc.DB2Exception: [IBM][CLI Driver] CLI0119E Unexpected system failure. SQLSTATE=5800

If you encounter this error when attempting to access a DB2 Universal Database (UDB) data source:

1. Check your user name and password custom properties in the data source properties page in the administrative console. Verify that they are correct.
2. Ensure the user ID and password do not contain any blank characters, before, in between, or after.
3. Check that the `WAS.policy` file exists for the application, for example, `D:\WebSphere\AppServer\installedApps\markSection.ear\META-INF\was.policy`.
4. View the entire exception listing for an underlying SQL error, and look it up using the DBM vendor message reference.

If you encounter this error while running DB2 on Red Hat Linux, the **max queues system wide** parameter is too low to support DB2 while it acquires the necessary resources to complete the transaction. When this problem exists, the exceptions J2CA0046E and DSRA0010E can precede the exception DSRA8100E.

To correct this problem, edit the /proc/sys/kernal/msgmni file to increase the value of the **max queues system wide** parameter to a value greater than 128.

COM.ibm.db2.jdbc.DB2Exception: [IBM][CLI Driver][DB2/NT] SQL0911N The current transaction has been rolled back because of a deadlock or timeout. Reason code "2". SQLSTATE=40001

This problem is probably an application-caused DB2 deadlock, particularly if you see an error similar to the following when accessing a DB2 data source:

```
ERROR CODE: -911
COM.ibm.db2.jdbc.DB2Exception: [IBM][CLI Driver][DB2/NT] SQL0911N
The current transaction has been rolled back because of a deadlock or timeout.
Reason code "2".  SQLSTATE=40001
```

To diagnose the problem:

1. Execute these DB2 commands:
   a. `db2 update monitor switches using LOCK ON`
   b. `db2 get snapshot for LOCKS on dbName >`

   The *directory_name*`\lock_snapshot.log` now has the DB2 lock information.
2. Turn off the lock monitor by executing: `db2 update monitor switches using LOCK OFF`

To verify that you have a deadlock:

1. Look for an application handle that has a lock-wait status, and then look for the ID of the agent holding lock to verify the ID of the agent.
2. Go to that handle to verify it has a lock-wait status, and the ID of the agent holding the lock for it. If it is the same agent ID as the previous one, then you know that you have a circular lock (deadlock).

To resolve the problem:

1. Examine your application and use a less restrictive isolation level if no concurrency access is needed.
2. Use caution when changing the **accessIntent** value to move to a lower isolation level. This change can result in data integrity problems.
3. For DB2/UDB Version 7.2 and earlier releases, you can set the DB2_RR_TO_RS flag from the DB2 command line window to eliminate unnecessary deadlocks, such as when the accessIntent defined on the bean method is too restrictive, for example, PessimisticUpdate. The DB@_RR_TO_RS setting has two impacts:
   - If RR is your chosen isolation level, it is effectively downgraded to RS.
   - If you choose another isolation level, and the DB2_RR_TO_RS setting is on, a scan skips over rows that are deleted but not committed, even though the row might qualify for the scan. The skipping behavior affects the RR, Read Stability (RS), and Cursor Stability (CS) isolation levels.

   For example, consider the scenario where transaction A deletes the row with column1=10 and transaction B does a scan where column1>8 and column1<12. With DB2_RR_TO_RS off, transaction B waits for transaction A to commit or rollback. If transaction A rolls back, the row with column1=10 is included in the result set of the transaction B query. With DB2_RR_TO_RS on, transaction B does not wait for transaction A to commit or rollback. Transaction B immediately receives query results that do not include the deleted row. Setting DB2_RR_TO_RS effectively changes locking behavior, thus avoiding deadlocks.

"COM.ibm.db2.jdbc.DB2ConnectionPoolDataSource" could not be found for data source ([data-source-name])"

This error is denoted by message DSRA8040I: Failed to connect to the DataSource.

This error usually occurs when the class path of the DB2 JDBC driver is set correctly to `${DB2_JDBC_DRIVER_PATH}/db2java.zip` but the environment variable DB2_JDBC_DRIVER_PATH is not set.

This error can also occur if you are using DB2 Version 7.1 or 7.2 and you have not yet run *usejdbc2*. This might be the problem if your path is correct but you still receive this error.

To confirm this problem:

1. Go to the **Manage WebSphere Variables** panel.
2. Select **Environment** to verify that there is no entry for the variable DB2_JDBC_DRIVER_PATH.

To correct this problem: Add the variable DB2_JDBC_DRIVER_PATH with **value** equal to the directory path containing the `db2java.zip` file.

java.sql.SQLException: Failure in loading T2 native library db2jcct2 DSRA0010E: SQL State = null, Error Code = -99,999

The *Failure in loading* message indicates one of two things:

- Usually this happens when the machine was not rebooted after installing DB2. Reboot the machine getting the error and try it again.
- The DB2 context is not getting set up correctly for the user running WebSphere Application Server. Source the db2profile file on the machine, and ensure that the environment contains pointers to the DB2 native libraries.

Lock contention exception occurs in database when data source implementation type is XA

Note: Because a lock contention exception can be caused by many factors, consider the following explanation and recommended response as a strategy for eliminating the possible reasons for your lock contention problem.

Symptom	A lock contention exception occurs in a DB2 database that your application accesses through a data source of implementation type XA.
Problem	Your application is trying to access database records that are locked by an XA transaction that is in ended (e) state, but cannot be prepared by the transaction manager.
Description	An XA transaction to DB2 that ends, but cannot be prepared, is in ended (e) state. Because it is *not* considered to be *in doubt*, the transaction manager cannot recover this transaction. DB2 does not return it in the list of in doubt transactions. DB2 also does not roll the transaction back immediately; it waits until all connections to the database are released. During this period of inaction, the transaction continues to hold locks on the database. Due to certain policies of WebSphere Application Server workload management, your application server might not disconnect all connections from the database to allow rollback of the transaction. Therefore the ended transaction persists in locking the same database records. If your application attempts to access these locked records, a lock contention exception occurs in DB2.
Recommended response	DB2 Version 8.2 is shipped with a sample application that connects to a defined DB2 server and uses the available DB2 APIs to obtain a list of these particular ended transactions. The application offers a configuration setting that enables you to designate an amount of time after which the application rolls these transactions back. Locate the sample application in the `sqllib/samples/db2xamon.c` directory of DB2 Version 8.2 and run it.

"DSRA8050W: Unable to find the DataStoreHelper class specified" exception occurs when trying to use a DB2 Universal Datasource in a mixed release cell.

This error usually occurs when you are using WebSphere Application Server Version 6 in conjunction with a previous version and attempt to create a DB2 Universal Datasource on the previous version.

This can happen because the DB2 Universal Datasource was not available on Version 5 and previous versions, but the Version 6 administrative console allows you to build one.

To correct this problem: create the datasource on Version 6.

Receive "'SYSTEM' is not a valid authorization ID" message when trying to access DB2 on a Windows machine where WebSphere Application Server is also installed.

Symptom	For a WebSphere Application Server on Windows installation that uses DB2 as the backend, you see the following exception in the JVM log:
	`java.sql.SQLException: [IBM][CLI Driver] SQL0567N "SYSTEM" is not a valid authorization ID. SQLSTATE=42602 DSRA0010E: SQL State = 42602, Error Code = -567`
	`at COM.ibm.db2.jdbc.app.SQLExceptionGenerator. throw_SQLException(Unknown Source)`
	`at COM.ibm.db2.jdbc.app.SQLExceptionGenerator. check_return_code(Unknown Source)`
	`at COM.ibm.db2.jdbc.app.DB2Connection.connect (Unknown Source)`
	`at COM.ibm.db2.jdbc.app.DB2Connection.<init> (Unknown Source)`
	`at COM.ibm.db2.jdbc.app.DB2ReusableConnection. <init>(Unknown Source)`
	`at COM.ibm.db2.jdbc.DB2PooledConnection. getConnection(Unknown Source)`
	`at com.ibm.ws.rsadapter.spi.WSRdbDataSource. getConnection(WSRdbDataSource.java:1035)`
	`at com.ibm.ws.rsadapter.spi. WSManagedConnectionFactoryImpl. createManagedConnection (WSManagedConnectionFactoryImpl.java:937)`
	`at com.ibm.ejs.j2c.poolmanager.FreePool. createManagedConnectionWithMCWrapper (FreePool.java:1502)`

Problem	This exception occurs for configurations in which WebSphere Application Server is a client to the DB2 server. The underlying problem is an authorization conflict between WebSphere Application Server on Windows and DB2 that arises when an application attempts to connect to DB2 without providing a user ID and a password.
Description	When a DB2 client and the DB2 database run on the same machine, DB2 allows the client to connect without a user ID and password. The connection is made under the credentials of the user that owns the client process: in this case, the application server JVM. However, if WebSphere Application Server runs as a Windows service, and the "Log on as" option is set to "Local System Account", the application server JVM is categorized as a subcomponent of a special Windows user called SYSTEM. This user is not allowed to connect to DB2, resulting in the previously shown exception.
Recommended response	You have two options: • Modify the WebSphere Application Server service to use a **Log on as** option of **This account**, and provide an account with permission to connect to DB2. **Or** • Configure your application server to provide credentials on the DB2 connection by using container-managed or component-managed authentication.

Related tasks

Chapter 6, “Troubleshooting by task,” on page 233

“Troubleshooting by component” on page 5

Example: Accessing data using IBM extended APIs to share connections between container-managed and bean-managed persistence beans

Related reference

“Cannot access a data source” on page 24

“Problems accessing an Oracle data source” on page 29

“Problems accessing a SQL server data source” on page 39

“Problems accessing a Cloudscape database” on page 40

“Problems accessing a Sybase data source” on page 43

Extensions to data access APIs

Problems accessing a SQL server data source

What kind of problem are you having accessing your SQL Server database?

- ERROR CODE: 20001 and SQL STATE: HY000.
- Application fails with message stating ″Cannot find stored procedure...″
- ERROR CODE: SQL5042 when running a Java application

ERROR CODE: 20001 and SQL STATE: HY000 accessing SQLServer database

The problem might be that the distributed transaction coordinator service is not started. Look for an error similar to the following example when attempting to access an SQL server database:

```
ERROR CODE: 20001
SQL STATE:  HY000
java.sql.SQLException: [Microsoft][SQLServer JDBC Driver]
[SQLServer]xa_open (0) returns -3
at com.microsoft.jdbc.base.BaseExceptions.createException(Unknown Source) ...
at com.microsoft.jdbcx.sqlserver.SQLServerDataSource.getXAConnection
(Unknown Source) ...
```

To confirm this problem:

1. Go to the Windows **Control Panel** and click **Services**(or click **Control Panel > Administrative Tools > Services**)
2. Verify whether the service **Distributed Transaction Coordinator** or **DTC** is started.
3. If not, start the Distributed Transaction Coordinator service.

Application fails with message stating "Cannot find stored procedure..." accessing an SQLServer database

This error can occur because the stored procedures for the Java Transaction API (JTA) feature are not installed on the Microsoft SQL Server.

To resolve the problem: Repeat the installation for the stored procedures for the JTA feature, according to the ConnectJDBC installation guide.

ERROR CODE: SQL5042 when running a Java application

This error can occur when you configure your application to run in the following manner:

1. you use a type 2 (application) driver running on the gateway to the OS 390

2. your application is an XA application.

OS 390 does not use XA, but uses SPM. To resolve the problem:

1. Check your dbm cfg to see that the SPM is not started on the gateway.
2. Assign a port and set the *db2comm* variable to **TCPIP**.
3. Update the dbm cfg value *SPM_NAME* to use your machine name.
4. Start the SPM on the gateway.

Related tasks

Chapter 6, "Troubleshooting by task," on page 233

"Troubleshooting by component" on page 5

Example: Accessing data using IBM extended APIs to share connections between container-managed and bean-managed persistence beans

Related reference

"Cannot access a data source" on page 24

"Problems accessing an Oracle data source" on page 29

"Problems accessing a DB2 database" on page 32

"Problems accessing a Cloudscape database" on page 40

"Problems accessing a Sybase data source" on page 43

Extensions to data access APIs

Problems accessing a Cloudscape database

What kind of problem are you having accessing your Cloudscape database?

- Unexpected IOException wrapped in SQLException, accessing Cloudscape database.
- The ″Select for update″ operatioin on one row causes table to become locked, triggering a deadlock condition.
- ″ERROR XSDB6: Another instance of Cloudscape might have already booted the database *databaseName*.″ error starting application server.
- Error ″The version of the IBM Universal JDBC driver in use is not licensed for connectivity to Cloudscape databases″
- Running an application causes a runtime exception which produces an unreadable message.

Tip: Cloudscape errorCodes (2000, 3000, 4000) indicate levels of severity, not specific error conditions. In diagnosing Cloudscape problems, pay attention to the given sqlState value.

Unexpected IOException wrapped in SQLException, accessing Cloudscape database

This problem can occur because Cloudscape databases use a large number of files. Some operating systems, such as the Solaris Operating Environment, limit the number of files an application can open at one time. If the default is a low number, such as 64, you can get this exception.

If your operating system lets you configure the number of file descriptors, you can correct the problem by setting the number to a high value, such as 1024.

The "select for update" operation causes table lock and deadlock when accessing Cloudscape

If a select for update operation on one row locks the entire table, which creates a deadlock condition, there might be undefined indexes on that table. The lack of an index on the columns you use in the `where` clause can cause Cloudscape to create a table lock rather than a row level lock.

To resolve this problem, create an index on the affected table.

ERROR XSDB6: Another instance of Cloudscape may have already booted the database "database"

This problem occurs because Cloudscape embedded framework only allows one Java virtual machine (JVM) to access the database instance at a time.

To resolve this problem:
1. Verify that you do not have other JDBC client programs, such as **ij** or **cview** running on that database instance, when WebSphere Application Server is running.
2. Verify that you do not use the same instance of the database for more than one data source or use the networkServer framework, which doesn't have this limitation.
3. If there are no connections to Cloudscape, delete the db.lck lock file. This file can be found in the directory where the Cloudscape database is mounted, under the schema directory. For example, if the database is mounted at /myCloudscapeDB, issue the command: rm /myCloudscapeDB/schemaName/db.lck

Error ″The version of the IBM Universal JDBC driver in use is not licensed for connectivity to Cloudscape databases″

Error "The version of the IBM Universal JDBC driver in use is not licensed for connectivity to Cloudscape databases"

At the client runtime, an error similar to the following occurs:

```
The version of the IBM Universal JDBC driver in use is not
licensed for connectivity to Cloudscape databases.  To connect
to this DB2 server, please obtain a licensed copy of the IBM DB2
Universal Driver for JDBC and SQLJ.  An appropriate license file
db2jcc_license_*.jar for this target platform must be installed to
the application classpath.  Connectivity to Cloudscape databases is
enabled by any of the following license files:
{ db2jcc_license_c.jar, b2jcc_license_cu.jar, db2jcc_license_cisuz.jar }
```

The problem occurs because an incorrect JDBC driver jar file name is specified in the class path for JDBC provider. For example, the jar file name may have an extra '_', as follows:

```
${UNIVERSAL_JDBC_DRIVER_PATH}/db2jcc_license__cu.jar
```

To resolve the problem:
1. Correct the UNIVERSAL_JDBC_DRIVER_PATH jar file name in the JACL script
2. Restart the cluster.
3. Rerun the client.

Running an application causes a runtime exception which produces an unreadable message.

At client runtime, you may receive a message similar to the following: `Caused by: com.ibm.db2.jcc.a.SqlException: DB2 SQL error: SQLCODE: -1, SQLSTATE: 42X05, SQLERRMC: ANNUITYHOLDER20^T42X05`

The problem occurs because the property *retrieveMessagesfromServerOnGetMessage*, which is required by WebSphere Application Server, has not been set.

To resolve the problem, on the admininstrative console

1. Click **Resources -> JDBC Providers**
2. Click on a Cloudscape provider
3. Scroll down and click on **Data Sources**
4. Select your data source (or add a new one)
5. Scroll down and select **Custom Properties**
6. If the property *retrieveMessagesFromServerOnGetMessage* already exists, set its value to true. If the property does not exist, select **New**and add the property *retrieveMessagesFromServerOnGetMessage* with a value **true**
7. Rerun the client

The SystemOut.log will now generate readable messages so that you can resolve the underlying problem.

Related tasks

Chapter 6, “Troubleshooting by task,” on page 233

Related reference

“Cannot access a data source” on page 24

“Problems accessing an Oracle data source” on page 29

“Problems accessing a DB2 database” on page 32

“Problems accessing a SQL server data source” on page 39

“Problems accessing a Sybase data source” on page 43

Problems accessing a Sybase data source

What kind of problem are you having accessing your Sybase database?

- ″Sybase Error 7713: Stored Procedure can only be executed in unchained transaction mode″ error.
- ″JZ0XS: The server does not support XA-style transactions. Please verify that the transaction feature is enabled and licensed on this server.″
- A container managed persistence (CMP) enterprise bean is causing exceptions.

"Sybase Error 7713: Stored Procedure can only be executed in unchained transaction mode" error

This error occurs when either:

- The JDBC attempts to put the connection in **autocommit(true)** mode.
- A stored procedure is not created in a compatible mode.

To fix the **autocommit(true)** mode problem, let the application change the connection to chained mode using the **Connection.setAutoCommit(false)** mode, or use a **set chained on** language command.

To resolve the stored procedure problem, use the `sp_procxmode` *`procedure_name`* `"anymode"` command.

"JZ0XS: The server does not support XA-style transactions. Please verify that the transaction feature is enabled and licensed on this server."

This error occurs when XA-style transactions are attempted on a server that does not have Distributed Transaction Management (DTM) installed.

To resolve this problem, use the instructions in the Sybase Manual titled: *Using Adaptive Server Distributed Transaction Management Features* to enable Distributed Transaction Management (DTM). The main steps in this procedure are:
1. Install the DTM option.
2. Check the `license.dat` file to verify that the DTM option is installed.
3. Restart the license manager.
4. Enable DTM in ISQL.
5. Restart the ASE service.

A container managed persistence (CMP) enterprise bean is causing exceptions

This error is caused by improper use of reserved words. Reserved words cannot be used as column names.

To correct this problem: Rename the variable to remove the reserved word. You can find a list of reserved words in the *Sybase Adaptive Server Enterprise Reference Manual; Volume 1: Building Blocks*, Chapter 4. This manual is available online at: http://manuals.sybase.com/onlinebooks/group-as/asg1250e/refman.

Related tasks

Chapter 6, “Troubleshooting by task,” on page 233

Related reference

“Cannot access a data source” on page 24

“Problems accessing an Oracle data source” on page 29

“Problems accessing a DB2 database” on page 32

“Problems accessing a SQL server data source” on page 39

“Problems accessing a Cloudscape database” on page 40

Provider coexistence considerations

Following are provider coexistence possibilities.

DB2 Legacy JDBC Providers and DB2 Universal JDBC Driver Providers

- Under WebSphere Application Server for z/OS, JDBC Provider definitions that use the Legacy DB2 for OS/390 and z/OS JDBC Driver (db2j2classes.zip) and JDBC Provider definitions that use the new DB2 Universal JDBC Driver (db2jcc.jar) must be carefully configured to ensure they never coexist on the same server. This is because some of the same class names are used in both drivers and these duplicate classes are functionally different. Adding jar files for the two types of drivers to the same CLASSPATH causes unpredictable results, since incorrect classes will be used for the provider whose CLASSPATH definition is added last.
- Carefully define the scope in which the two different types of driver providers are used. If you define one provider type under one scope (cell, node, or server), and the other provider type under another scope, separation is not ensured if the two scopes include the same server.

For example, if you define a DB2 for zOS Local JDBC Provider (RRS) at a node level, then define a DB2 Universal JDBC Driver Provider at a server level where the server is in the same node, the provider definition at the node level is propagated down to the server level. As a result, a conflict occurs between the JDBC drivers used by the two providers.

To help you understand which WebSphere Application Server for z/OS providers cannot coexist together, the providers are listed below under the DB2 JDBC driver that it uses:

1. Providers that use the DB2 for z/OS Legacy JDBC Driver
   - DB2 for zOS Local JDBC Provider (RRS)
2. Providers that use the DB2 Universal JDBC Driver
   - DB2 Universal JDBC Driver Provider
   - DB2 Universal JDBC Driver Provider (XA)
   - Cloudscape Network Server Using Universal JDBC Driver Provider

DB2 Drivers and Cloudscape

- The Cloudscape Network Server Using Universal JDBC Driver Provider uses an embedded copy of the DB2 Universal JDBC Driver that is shipped with Cloudscape. This provider is configured to automatically use the new level of the DB2 Universal JDBC Driver when installing the DB2 Version 7, Version 8, or standalone Type 4 version of the DB2 Universal JDBC Driver on the system where WebSphere Application Server for z/OS is configured.

  The Cloudscape Network Server Using Universal JDBC Driver Provider can coexist on the same server as the DB2 for Universal JDBC Driver Providers since it uses the same DB2 JDBC driver.
- The Cloudscape Network Server Using Universal JDBC Driver Provider cannot coexist in the same server as the DB2 for zOS Local JDBC Provider (RRS), because the Cloudscape provider uses the DB2 Universal JDBC Driver and the DB2 for zOS Local JDBC Provider (RRS) uses the DB2 for OS/390 and z/OS Legacy JDBC Driver. These drivers conflict with one another.

Related tasks

Using a DB2 Universal JDBC Driver Provider with WebSphere Application Server for z/OS

Related reference

"Vendor-specific data sources minimum required settings" on page 312

Connection and connection pool statistics

Performance Monitoring Infrastructure (PMI) method calls that are supported in the two existing Connection Managers (JDBC and J2C) are still supported in this version of WebSphere Application Server. The calls include:

- ManagedConnectionsCreated
- ManagedConnectionsAllocated
- ManagedConnectionFreed
- ManagedConnectionDestroyed
- BeginWaitForConnection
- EndWaitForConnection
- ConnectionFaults
- Average number of ManagedConnections in the pool
- Percentage of the time that the connection pool is using the maximum number of ManagedConnections
- Average number of threads waiting for a ManagedConnection
- Average percent of the pool that is in use

- Average time spent waiting on a request
- Number of ManagedConnections that are in use
- Number of Connection Handles
- FreePoolSize
- UseTime

Java Specification Request (JSR) 77 requires statistical data to be accessed through managed beans (Mbeans) to facilitate this. The Connection Manager passes the ObjectNames of the Mbeans created for this pool. In the case of Java Message Service (JMS) *null* is passed in. The interface used is :

```
PmiFactory.createJ2CPerf(
    String pmiName, // a unique Identifier for JCA /JDBC.  This is the
                    // ConnectionFactory name.

    ObjectName providerName,// the ObjectName of the J2CResourceAdapter
                            // or JDBCProvider Mbean

    ObjectName factoryName // the ObjectName of the J2CConnectionFactory
                           // or DataSourceMbean.
)
```

The following Unified Modeling Language (UML) diagram shows how JSR 77 requires statistics to be reported :

In WebSphere Application Server Version 5.x, the JCAStats interface was implemented by the J2CResourceAdapter Mbean, and the JDBCStats interface was implemented by the JDBCProvider Mbean. The JCAConnectionStats and JDBCConnectionStats interfaces are not implemented because they collect statistics for non pooled connections - which are not present in the JCA 1.0 Specification. JCAConnectionPoolStats, and JDBCConnectionPoolStats do not have a direct implementing Mbean; those statistics are gathered through a call to PMI. A J2C resource adapter, and JDBC provider each contain a list of ConnectionFactory or

DataSource ObjectNames, respectively. The ObjectNames are used by PMI to find the appropriate connection pool in the list of PMI modules.

The JCA 1.5 Specification allows an exception from the matchManagedConnection() method that indicates that the resource adapter requests that the connection not be pooled. In that case, statistics for that connection are provided separately from the statistics for the connection pool.

Example: Connection factory lookup

```
import javax.resource.cci.*;
import javax.resource.ResourceException;

import javax.naming.*;

import java.util.*;

/**
 * This class is used to look up a connection factory.
 */
public class ConnectionFactoryLookup {

   String jndiName = "java:comp/env/eis/SampleConnection";
   boolean verbose = false;

   /**
    * main method
    */
   public static void main(String[] args) {
      ConnectionFactoryLookup cfl = new ConnectionFactoryLookup();
      cfl.checkParam(args);

      try {
         cfl.lookupConnectionFactory();
      }
      catch(javax.naming.NamingException ne) {
   System.out.println("Caught this " + ne);
   ne.printStackTrace(System.out);
      }
      catch(javax.resource.ResourceException re) {
   System.out.println("Caught this " + re);
   re.printStackTrace(System.out);
      }
   }

 /**
  * This method does a simple Connection Factory lookup.
    *
    * After the Connection Factory is looked up, a connection is got from
  * the Connection Factory.  Then the Connection MetaData is retrieved
    * to verfiy the connection is workable.
  */
 public void lookupConnectionFactory()
  throws javax.naming.NamingException, javax.resource.ResourceException {

  javax.resource.cci.ConnectionFactory factory = null;
  javax.resource.cci.Connection conn = null;
  javax.resource.cci.ConnectionMetaData metaData = null;

  try {
   // lookup the connection factory
         if (verbose) System.out.println("Look up the connection factory...");

   InitialContext ic = new InitialContext();
   factory = (ConnectionFactory) ic.lookup(jndiName);
```

```
 // Get connection
       if (verbose) System.out.println("Get the connection...");
 conn = factory.getConnection();

 // Get ConnectionMetaData
 metaData = conn.getMetaData();

 // Print out the metadata Informatin.
 if (verbose) System.out.println("  **  EISProductName   :" + metaData.getEISProductName());
 if (verbose) System.out.println("      EISProductVersion:" + metaData.getEISProductName());
 if (verbose) System.out.println("      UserName         :" + metaData.getUserName());

       System.out.println("Connection factory "+jndiName+" is successfully looked up");
}
catch (javax.naming.NamingException ne) {
 // Connection factory cannot be looked up.
 throw ne;
}
catch (javax.resource.ResourceException re) {
 // Something wrong with connections.
 throw re;
}
finally {
 if (conn != null) {
  try {
   conn.close();
  }
  catch (javax.resource.ResourceException re) {
  }
 }
}
}

 /**
  * Check and gather all the parameters.
  */
 private void checkParam(String args[]) {
int i = 0, j;
String arg;
char flag;
    boolean help = false;

// parse out the options
while (i < args.length && args[i].startsWith("-")) {
 arg = args[i++];

 // get the database name
 if (arg.equalsIgnoreCase("-jndiName")) {
  if (i < args.length)
   jndiName = args[i++];
  else {
   System.err.println("-jndiName requires a J2C Connection Factory JNDI name");
   break;
  }
 }
 else { // check for verbose, cmp , bmp
  for (j = 1; j < arg.length(); j++) {
   flag = arg.charAt(j);
   switch (flag) {
    case 'v' :
    case 'V' :
     verbose = true;
     break;
                case 'h' :
                case 'H' :
                    help = true;
                    break;
```

```
      default :
       System.err.println("illegal option " + flag);
       break;
     }
    }
   }
  }

  if ((i != args.length) || help) {
   System.err.println("Usage: java ConnectionFactoryLookup [-v] [-h]");
   System.err.println("    [-jndiName the J2C Connection Factory JNDI name]");
   System.err.println("-v=verbose");
   System.err.println("-h=this information");
   System.exit(1);
  }
   }

}
```

Related concepts

Connection factory
An application component uses a *connection factory* to access a connection instance, which the component then uses to connect to the underlying enterprise information system (EIS).

Database exceptions resulting from foreign key conflicts, or deadlock when entity beans are configured for optimistic concurrency control

Exceptions resulting from foreign key conflicts, which signify violations of database referential integrity

A database *referential integrity* (RI) policy prescribes rules for how data is written to and deleted from the database tables to maintain relational consistency. Run-time requirements for managing bean persistence, however, can cause an EJB application to violate RI rules, which can cause database exceptions.

Your EJB application is violating database RI if you see an exception message in your WebSphere Application Server trace or log file that is similar to one of the following messages (which were produced in an environment running DB2):

`The insert or update value of the FOREIGN KEY` *`table1.name_of_foreign_key_constraint`*
`is not equal to any value of the parent key of the parent table.`

or

`A parent row cannot be deleted because the relationship` *`table1.name_of_foreign_key_constraint`*
`is not equal to any value of the parent key of the parent table.`

To prevent these exceptions, you must designate the order in which entity beans update relational database tables by defining sequence groups for the beans.

Exceptions resulting from deadlock caused by optimistic concurrency control schemes

Additionally, sequence grouping can minimize transaction rollback exceptions for entity beans that are configured for optimistic concurrency control. Optimistic concurrency control dictates that database locks be held for minimal amounts of time, so that a maximum number of transactions consistently have access to the data. In such a highly available database, concurrent transactions can attempt to

lock the same table row and create deadlock. The resulting exceptions can generate messages similar to the following (which was produced in an environment running DB2):

```
Unsuccessful execution caused by deadlock or timeout.
```

Use the sequence grouping feature to order bean persistence so that database deadlock is less likely to occur.

Related concepts

Sequence grouping for container-managed persistence

Vendor-specific data sources minimum required settings

Use this table as an at-a-glance reference of JDBC providers that can be defined for use with WebSphere Application Server Version 6.x, to establish data sources for transacting with relational databases. A list that contains detailed requirements for creating data sources with these providers follows the table. (The list also contains information about JDBC providers that are *deprecated* in WebSphere Application Server Version 6.x.)

Database type	JDBC Provider	Transaction support	Version and other considerations
DB2 on Windows, UNIX, or workstation-based LINUX	DB2 Universal JDBC Provider	One phase only	
	DB2 Universal JDBC Provider (XA)	One and two phase	The XA implementation is *not* supported in WebSphere Application Server run on workstation-based LINUX
	DB2 legacy CLI-based Type 2 JDBC Provider	One phase only	
	DB2 legacy CLI-based Type 2 JDBC Provider (XA)	One and two phase	

Database type	JDBC Provider	Transaction support	Version and other considerations
DB2 UDB for iSeries	DB2 UDB for iSeries (Native)	One phase only	Recommended for use with WebSphere Application Server **run on iSeries**
	DB2 UDB for iSeries (Native XA)	One and two phase	Recommended for use with WebSphere Application Server **run on iSeries**
	DB2 UDB for iSeries (Toolbox)	One phase only	
	DB2 UDB for iSeries (Toolbox XA)	One and two phase	
	DB2 Universal JDBC Provider (XA)	One and two phase	**-***Only* for use with WebSphere Application Server **run on z/OS** **-**Only driver type 4 is supported **-**Does *not* support Version 4 data sources
	DB2 legacy CLI-based Type 2 JDBC Provider	One phase only	**-***Only* for use with WebSphere Application Server run on Windows, UNIX, or workstation-based LINUX **-** Requires the DB2 Connect driver (available from DB2)
	DB2 legacy CLI-based Type 2 JDBC Provider (XA)	One and two phase	**-***Only* for use with WebSphere Application Server run on Windows, UNIX, or workstation-based LINUX **-** Requires the DB2 Connect driver (available from DB2)

Database type	JDBC Provider	Transaction support	Version and other considerations
DB2 on z/OS	DB2 for z/OS Local JDBC Provider (RRS)	One and two phase	*Only* for use with WebSphere Application Server **run on z/OS**
	DB2 Universal JDBC Provider	One phase only	
	DB2 Universal JDBC Provider (XA)	One and two phase	**-**Only driver type 4 is supported in WebSphere Application Server **run on z/OS** **-**Does *not* support Version 4 data sources in WebSphere Application Server **run on z/OS**
	DB2 legacy CLI-based Type 2 JDBC Provider	One phase only	**-***Only* for use with WebSphere Application Server run on Windows, UNIX, or workstation-based LINUX **-** Requires the DB2 Connect program (available from DB2)
	DB2 legacy CLI-based Type 2 JDBC Provider (XA)	One and two phase	**-***Only* for use with WebSphere Application Server run on Windows, UNIX, or workstation-based LINUX **-** Requires the DB2 Connect program (available from DB2)

Database type	JDBC Provider	Transaction support	Version and other considerations
Cloudscape	Cloudscape JDBC Provider	One phase only	- Not for use in clustering environment: Cloudscape is accessible from a single JVM only - Does not support Version 4 data sources
	Cloudscape JDBC Provider (XA)	One and two phase	- Not for use in clustering environment: Cloudscape is accessible from a single JVM only - Does not support Version 4 data sources
	Cloudscape Network Server using Universal JDBC driver	One phase only	Does not support Version 4 data sources
Informix	Informix JDBC Driver	One phase only	
	Informix JDBC Driver (XA)	One and two phase	
Sybase	Sybase JDBC Driver	One phase only	
	Sybase JDBC Driver (XA)	One and two phase	
Oracle	Oracle JDBC Driver	One phase only	
	Oracle JDBC Driver (XA)	One and two phase	

Database type	JDBC Provider	Transaction support	Version and other considerations
MS SQL Server	DataDirect ConnectJDBC type 4 driver for MS SQL Server	One phase only	Only for use with the corresponding driver from DataDirect Technologies
	DataDirect ConnectJDBC type 4 driver for MS SQL Server (XA)	One and two phase	Only for use with the corresponding driver from DataDirect Technologies
	WebSphere embedded ConnectJDBC driver for MS SQL Server	One phase only	- Not available for Application Server on z/OS - Cannot be used outside of WebSphere Application Server environment
	WebSphere embedded ConnectJDBC driver for MS SQL Server (XA)	One and two phase	- Not available for Application Server on z/OS - Cannot be used outside of WebSphere Application Server environment

Detailed data source requirements per JDBC provider and platform

The following list contains the requirements for creating data sources with every JDBC provider type that is supported in WebSphere Application Server Version 6.x. Specific fields are designated for the user and password properties. Inclusion of a property in the list does not imply that you should add it to the data source custom properties list. Rather, inclusion in the list means that a value is *typically* required for that field.

Important: After you determine the type of JDBC provider that suits your application and environment, ensure that you acquire the corresponding JDBC driver at a release level supported by this version of WebSphere Application Server. Consult the Supported hardware and software Web page at the http://www.ibm.com/software/webservers/appserv/doc/latest/prereq.html IBM Web address.

Use these links to find your provider and data source information:
- DB2 on z/OS, connecting to Application Server on z/OS
- "DB2 UDB for iSeries, connecting to Application Server on z/OS" on page 320
- Cloudscape
- Informix
- Sybase
- Oracle
- MS SQL Server

DB2 on z/OS, connecting to Application Server on z/OS
1. **DB2 for zOS Local JDBC Provider (RRS)**

The DB2 for zOS Local JDBC Provider (RRS) is for use with the DB2 for 390 and z/OS Legacy JDBC Driver. It can be used only with WebSphere Application Server for z/OS. This provider supports the creation of WebSphere Application Server for z/OS v5.0 and v4.0 data sources. It also uses z/OS Resource Recovery Services (RRS) to coordinate transactions across multiple resource managers using two-phase commit processing.

The DB2 for zOS Local JDBC Provider (RRS) allows applications to use both JDBC and Structured Query Language in Java (SQLJ) access to DB2 databases. Use of SQLJ with Container Managed Persistence (CMP) is not supported under this provider.

To use this provider, the legacy DB2 for z/OS JDBC Driver must be installed and configured to the WebSphere Application Server. Refer to the topic Using a DB2 for z/OS Local JDBC Provider (RRS) with WebSphere Application Server for z/OS.

The following configuration information is provided in a template for the DB2 for zOS Local JDBC Driver Provider (RRS), and is automatically filled in when you select this provider.

Data source implementation:

`com.ibm.db2.jcc.DB2ConnectionPoolDataSource`

This DB2 data source allows WebSphere Application Server for z/OS to perform connection pooling. Note that when you configure the data source, you must specify a name for the data source definition.

This provider requires JDBC driver files:

- `db2j2classes.zip`, which are DB2 for z/OS Legacy JDBC driver files. They can be retrieved using the following class path:

  `${DB2390_JDBC_DRIVER_PATH}/classes/db2j2classes.zip`
- The native files (`.so` type files) required by the DB2 for OS/390 and z/OS Legacy JDBC driver, retrievable from the following library path:

  `${DB2390_JDBC_DRIVER_PATH}/lib`

The driver requires DataStoreHelper class:

`com.ibm.websphere.rsadapter.DB2DataStoreHelper`

It also requires a valid authentication alias. When res-auth = CONTAINER is used, however, it is permissible to not specify any authentication alias. In this case, the user identity associated with a connection created by the data source is the user identity associated with the current thread at the time a connection request is made.

This driver requires the following properties:

- **databaseName** The location name of the target database, used when establishing connections using this data source.

2. **DB2 Universal JDBC Provider**

   The DB2 Universal JDBC Driver is an architecture-neutral JDBC driver for distributed and local DB2 access. Because the Universal Driver architecture is independent of any particular JDBC driver connectivity or target platform, it allows both Java connectivity (Type 4) or Java Native Interface (JNI) based connectivity (Type 2) in a single driver instance to DB2.

   Note: To use this provider, you must have the DB2 Universal JDBC Driver for DB2 Version 7 or DB2 Version 8 installed and configured for WebSphere Application Server for z/OS. Refer to the topic Using a DB2 Universal JDBC Driver Provider with WebSphere Application Server for z/OS.

   The DB2 Universal JDBC Provider allows applications to use both JDBC and Structured Query Language in Java (SQLJ) access to DB2 databases. SQLJ use with CMP is also supported.

It supports the following data source:

com.ibm.db2.jcc.DB2ConnectionPoolDataSource

Note that when you configure the data source, you must specify a name for the data source definition. This data source implementation class performs one-phase commit processing when you specify driver Type 4 in WebSphere Application Server for z/OS. When you specify driver Type 2, Application Server for z/OS uses RRS to coordinate transaction processing, and two-phase commit processing is performed for global transactions. The provider requires JDBC driver files:

- db2jcc.jar This is the DB2 Universal JDBC Driver jar file. After the DB2 installation, this jar file is located in DB2's install directory. The fully-qualified path of this jar must be specified as the value of the DB2UNIVERSAL_JDBC_DRIVER_PATH environment variable. Class path:

  ```
  ${DB2UNIVERSAL_JDBC_DRIVER_PATH}/db2jcc.jar
  ```
- db2jcc_License_cu.jar This is the DB2 Universal JDBC driver license file that allows access to DB2 Universal databases under Cloudscape and workstations. It is not used for WebSphere Application Server for z/OS, but is included to make the provider definition common between WebSphere Application Server for z/OS and WebSphere Application Server Distributed. Class path:

  ```
  ${UNIVERSAL_JDBC_DRIVER_PATH}/db2jcc_License_cu.jar
  ```
- db2jcc_License_cisuz.jar This is the DB2 Universal JDBC Driver license file that allows access to DB2 Universal databases under Cloudscape, workstations, and z/OS. After you install DB2, this jar file appears in the same DB2 directory as db2jcc.jar. Class path:

  ```
  ${DB2UNIVERSAL_JDBC_DRIVER_PATH}/db2jcc_License_cisuz.jar
  ```
- The native files (.so type files) required by the DB2 Universal JDBC Driver in WebSphere Application Server for z/OS. Use the following library path:

  ```
  ${DB2UNIVERSAL_JDBC_DRIVER_NATIVEPATH}
  ```

The DB2 Universal JDBC driver requires **DataStoreHelper** class:

```
com.ibm.websphere.rsadapter.DB2UniversalDataStoreHelper
```

This driver also requires a valid authentication alias if the driverType property (see properties below) is set to 4. If the driverType property is set to 2, a component-managed authentication alias must be specified to use the datasource with res-auth APPLICATION. In the case where driverType 2 is specified and the datasource is used with res-auth CONTAINER, you can specify a container-managed authentication alias; however, it is not required. If you do not specify a container-managed alias, the user identity associated with a connection created by the datasource will be the user identity associated with the current thread at the time the connection is obtained.

It requires the following properties:

- **databaseName** This is an actual database name if the **driverType** is set to **4**, or a locally cataloged database name if the **driverType** is set to **2**.
- **driverType** The JDBC connectivity type of a data source. *There are two permitted values: 2 and 4*. If you want to use Universal JDBC Type 2 driver, set this value to 2. If you want to use Universal JDBC Type 4 driver, set this value to 4.
- **serverName** The TCP/IP address or host name for the Distributed Relational Database Architecture (DRDA) server. Provide a value for this property only if your **driverType** is set to **4**. This property is not required if your **driverType** is set to **2**.
- **portNumber** The TCP/IP port number where the DRDA server resides. Provide a value for this property only if your **driverType** is set to **4**. This property is not required if your **driverType** is set to **2**.

3. **DB2 Universal JDBC Provider (XA)**

   This provider is the XA DB2 Universal JDBC provider that uses the DB2 Universal JDBC driver to provide access to DB2 databases. The Universal JDBC driver supports Java communication-based connectivity (driver Type 4), which allows distributed access to DB2. The driver also supports Java Native Interface (JNI) based connectivity (driver type 2), which allows local access to DB2. For XA capabilities, however, driver type 2 is not supported by the DB2 Universal JDBC Driver on WebSphere Application Server for z/OS. Therefore driver type 2 should *not* be used when defining an XA data source under this provider.

   Note: To use this provider, you must have the DB2 Universal JDBC Driver for DB2 Version 7 or DB2 Version 8 installed and configured for WebSphere Application Server for z/OS, or you must have the z/OS Application Connectivity to DB2 for z/OS feature installed and configured for WebSphere Application Server for z/OS. Refer to the topic Using a DB2 Universal JDBC Driver Provider with WebSphere Application Server for z/OS.

   The DB2 Universal JDBC Provider (XA) allows applications to use both JDBC and Structured Query Language in Java (SQLJ) access to DB2 databases. SQLJ use with CMP is also supported.

   This provider does not support the creation of Version 4.0 data sources.

   The DB2 Universal JDBC Provider (XA) supports the two phase data source:

   com.ibm.db2.jcc.DB2XADataSource

   Note that when you configure the data source, you must specify a name for the data source definition.

   It requires JDBC driver files:

   - `db2jcc.jar` This is the DB2 Universal JDBC Driver jar file. After the DB2 installation, this jar file is located in DB2's install directory. The fully-qualified path of this jar must be specified as the value of the DB2UNIVERSAL_JDBC_DRIVER_PATH environment variable:

     `${DB2UNIVERSAL_JDBC_DRIVER_PATH}/db2jcc.jar`
   - `db2jcc_License_cu.jar` This is the DB2 Universal JDBC driver license file that allows access to DB2 Universal databases under Cloudscape and workstations. It is not used for WebSphere Application Server for z/OS, but is included to make the provider definition common between WebSphere Application Server for z/OS and WebSphere Application Server Distributed. Class path:

     `${UNIVERSAL_JDBC_DRIVER_PATH}/db2jcc_License_cu.jar`
   - `db2jcc_License_cisuz.jar` This is the DB2 Universal JDBC Driver license file that allows access to DB2 Universal databases under Cloudscape, workstations, and z/OS. After you install DB2, this jar file appears in the same DB2 directory as db2jcc.jar. Class path:

     `${DB2UNIVERSAL_JDBC_DRIVER_PATH}/db2jcc_License_cisuz.jar`
   - The native files (`.so` type files) required by the DB2 Universal JDBC Driver in WebSphere Application Server for z/OS. Use the following library path:

     `${DB2UNIVERSAL_JDBC_DRIVER_NATIVEPATH}`

     (In cases that do not require native files, set the DB2UNIVERSAL_JDBC_DRIVER_NATIVEPATH to null.)

   The driver requires **DataStoreHelper** class:

   `com.ibm.websphere.rsadapter.DB2UniversalDataStoreHelper`

   It also requires a valid authentication alias.

   The DB2 Universal JDBC driver requires the following properties:

- **databaseName** This is a locally cataloged database name.
- **driverType** This is the JDBC connectivity type of a data source. *If you are running a version of DB2 prior to DB2 V8.1 FP6, you are restricted to using only the type 2 driver.*
- **serverName** The TCP/IP address or host name for the Distributed Relational Database Architecture (DRDA) server. Provide a value for this property only if your **driverType** is set to **4**. This property is not required if your **driverType** is set to **2**.
- **portNumber** The TCP/IP port number where the DRDA server resides. Provide a value for this property only if your **driverType** is set to **4**. This property is not required if your **driverType** is set to **2**.

For more information on DB2 for z/OS, visit the DB2 Web site at: http://www.ibm.com/software/data/db2/.

DB2 UDB for iSeries, connecting to Application Server on z/OS

In the rare case that you need to connect to DB2 UDB on iSeries to provide JDBC connectivity for an application run on WebSphere Application Server for z/OS, you can use the iSeries Toolbox driver for Java, the iSeries Toolbox XA-compliant driver for Java, or the DB2 JDBC Universal Driver XA.

1. **DB2 UDB for iSeries (Toolbox)**

   This JDBC driver, also known as iSeries Toolbox driver for Java, is provided in the DB2 for iSeries database server. Use this driver for remote DB2 connections on iSeries. We recommend you use this driver instead of the IBM Developer Kit for Java JDBC Driver to access remote DB2 UDB for iSeries systems.

   DB2 UDB for iSeries (Toolbox) supports one phase data source:

   com.ibm.as400.access.AS400JDBCConnectionPoolDataSource

   Requires JDBC driver files: **jt400.jar**

   Requires **DataStoreHelper** class:

   `com.ibm.websphere.rsadapter.DB2AS400DataStoreHelper`

   Requires an authentication alias.

   Requires properties:
   - **serverName** The name of the server from which the data source obtains connections. Example: *myserver.mydomain.com*.
2. **DB2 UDB for iSeries (Toolbox XA)**

   This XA compliant JDBC driver, also known as iSeries Toolbox XA-compliant driver for Java, is provided in the DB2 for iSeries database server. Use this driver for remote DB2 connections on iSeries. We recommend you use this driver instead of the IBM Developer Kit for Java JDBC Driver to access remote DB2 UDB for iSeries systems.

   DB2 UDB for iSeries (Toolbox XA) supports two phase data source:

   com.ibm.as400.access.AS400JDBCXADataSource

   Requires JDBC driver files: **jt400.jar**

   Requires **DataStoreHelper** class:

   `com.ibm.websphere.rsadapter.DB2AS400DataStoreHelper`

   Requires an authentication alias.

   Requires properties:
   - **serverName** The name of the server from which the data source obtains connections. Example: *myserver.mydomain.com*.
3. **DB2 Universal JDBC Provider (XA)**

This provider is the XA DB2 Universal JDBC provider that uses the DB2 Universal JDBC driver to provide access to DB2 databases. The Universal JDBC driver supports Java communication-based connectivity (driver Type 4), which allows distributed access to DB2. If you are running WebSphere Application Server for z/OS and connecting to DB2 UDB for iSeries, you *cannot* use Java Native Interface (JNI) based connectivity (driver type 2) with this provider.

Note: To use this provider, you must have the DB2 Universal JDBC Driver for DB2 Version 7 or DB2 Version 8 installed and configured for WebSphere Application Server for z/OS, or you must have the z/OS Application Connectivity to DB2 for z/OS feature installed and configured for WebSphere Application Server for z/OS. Refer to the topic Using a DB2 Universal JDBC Driver Provider with WebSphere Application Server for z/OS.

The DB2 Universal JDBC Provider (XA) allows applications to use both JDBC and Structured Query Language in Java (SQLJ) access to DB2 databases. SQLJ use with CMP is also supported.

This provider does not support the creation of Version 4.0 data sources.

The DB2 Universal JDBC Provider (XA) supports the two phase data source:

com.ibm.db2.jcc.DB2XADataSource

Note that when you configure the data source, you must specify a name for the data source definition.

It requires JDBC driver files:

- db2jcc.jar This is the DB2 Universal JDBC Driver jar file. After the DB2 installation, this jar file is located in DB2's install directory. The fully-qualified path of this jar must be specified as the value of the DB2UNIVERSAL_JDBC_DRIVER_PATH environment variable:

  ```
  ${DB2UNIVERSAL_JDBC_DRIVER_PATH}/db2jcc.jar
  ```

- db2jcc_License_cu.jar This is the DB2 Universal JDBC driver license file that allows access to DB2 Universal databases under Cloudscape and workstations. It is not used for WebSphere Application Server for z/OS, but is included to make the provider definition common between WebSphere Application Server for z/OS and WebSphere Application Server Distributed. Class path:

  ```
  ${UNIVERSAL_JDBC_DRIVER_PATH}/db2jcc_License_cu.jar
  ```

- db2jcc_License_cisuz.jar This is the DB2 Universal JDBC Driver license file that allows access to DB2 Universal databases under Cloudscape, workstations, and z/OS. After you install DB2, this jar file appears in the same DB2 directory as db2jcc.jar. Class path:

  ```
  ${DB2UNIVERSAL_JDBC_DRIVER_PATH}/db2jcc_License_cisuz.jar
  ```

- The native files (.so type files) required by the DB2 Universal JDBC Driver in WebSphere Application Server for z/OS. Use the following library path:

  ```
  ${DB2UNIVERSAL_JDBC_DRIVER_NATIVEPATH}
  ```

  (In cases that do not require native files, set the DB2UNIVERSAL_JDBC_DRIVER_NATIVEPATH to null.)

The driver requires **DataStoreHelper** class:

```
com.ibm.websphere.rsadapter.DB2UniversalDataStoreHelper
```

It also requires a valid authentication alias.

The DB2 Universal JDBC driver requires the following properties:

- **databaseName** This is a locally cataloged database name.

- **driverType** This is the JDBC connectivity type of a data source. Set this property to type 4 if you are running WebSphere Application Server for z/OS and connecting to DB2 UDB for iSeries.
- **serverName** The TCP/IP address or host name for the Distributed Relational Database Architecture (DRDA) server. Provide a value for this property only if your **driverType** is set to **4**. This property is not required if your **driverType** is set to **2**.
- **portNumber** The TCP/IP port number where the DRDA server resides. Provide a value for this property only if your **driverType** is set to **4**. This property is not required if your **driverType** is set to **2**.

For more information on DB2 for iSeries, visit the DB2 Web site at: http://www.ibm.com/software/data/db2/.

Cloudscape

1. **Cloudscape JDBC Provider**

   The Cloudscape JDBC Provider provides the JDBC access to the Cloudscape database. This Cloudscape JDBC driver used the embedded framework. You cannot use any Version 4.0 data sources with Cloudscape.

   Cloudscape JDBC Provider supports one phase data source:

   com.ibm.db2j.jdbc.DB2jConnectionPoolDataSource

   Requires JDBC driver files: **db2j.jar.**

   Requires **DataStoreHelper** class:

   `com.ibm.websphere.rsadapter.CloudscapeDataStoreHelper`

   Does not require a valid authentication alias.

   Requires properties:

   - **databaseName** The name of the database from which the data source obtains connections. If you do not specify a fully qualified path name, Application Server uses the default location of `WAS_HOME./cloudscape` (or the equivalent default for a UNIX or LINUX environment).
     - Example database path name for Windows: `c:\temp\sampleDB`
     - Example database path name for UNIX or LINUX: `/tmp/sampleDB`

     If no database currently exists for the path name you want to specify, simply append `;create=true` to the path name to create a database dynamically. (For example: `c:\temp\sampleDB;create=true`)

2. **Cloudscape JDBC Provider (XA)**

   The Cloudscape JDBC Provider (XA) provides the XA-compliant JDBC access to the Cloudscape database. This Cloudscape JDBC driver uses the embedded framework. You cannot use any Version 4.0 data sources with Cloudscape.

   Cloudscape JDBC Provider (XA) supports two phase data source:

   com.ibm.db2j.jdbc.DB2jXADataSource

   Requires JDBC driver files: **db2j.jar**

   Requires **DataStoreHelper** class:

   `com.ibm.websphere.rsadapter.CloudscapeDataStoreHelper`

   Does not require a valid authentication alias.

   Requires properties:

   - **databaseName** The name of the database from which the data source obtains connections. If you do not specify a fully qualified path name, Application Server uses the default location of `WAS_HOME./cloudscape` (or the equivalent default for a UNIX or LINUX environment).
     - Example database path name for Windows: `c:\temp\sampleDB`
     - Example database path name for UNIX or LINUX: `/tmp/sampleDB`

If no database currently exists for the path name you want to specify, simply append `;create=true` to the path name to create a database dynamically. (For example: `c:\temp\sampleDB;create=true`)

3. **Cloudscape Network Server using Universal JDBC driver**

   This Cloudscape driver takes advantage of the Network Server support that the DB2 universal Type 4 JDBC driver provides. You cannot use any Version 4.0 data sources with Cloudscape.

   Cloudscape uses the DB2 Universal Driver when using the Network Server. It supports one phase data source:

   com.ibm.db2.jcc.DB2ConnectionPoolDataSource

   Requires JDBC driver files:

   - **db2jcc.jar** If you install and run DB2, you must use the **db2jcc.jar** file that comes with DB2. To do that, the classpath in the JDBC template for Cloudscape network server is set to be:

     ```
     <classpath>${DB2UNIVERSAL_JDBC_DRIVER_PATH}/db2jcc.jar</classpath>

     <classpath>${CLOUDSCAPE_JDBC_DRIVER_PATH}/db2j.jar</classpath>

     <classpath>${CLOUDSCAPE51_JDBC_DRIVER_PATH}/db2j.jar</classpath>

     <classpath>${UNIVERSAL_JDBC_DRIVER_PATH}/db2jcc.jar</classpath>
     ```

     which means that the db2jcc.jar from DB2 always takes precedence. Note that this also means that you must set the DB2 environment variable **DB2UNIVERSAL_JDBC_DRIVER_PATH** in WebSphere when you set up your DB2 datasource. This is instead of hard coding the path of the db2jcc.jar for DB2 datasources.
   - **db2jcc_license_cu.jar** This file is the DB2 Universal JDBC license file that provides access to the Cloudscape databases using the **Network Server** framework. Use this file to gain access to the database. This file ships with WebSphere and is located in **${UNIVERSAL_JDBC_DRIVER_PATH}**.

     Note: **UNIVERSAL_JDBC_DRIVER_PATH** is a WebSphere environment variable that is already defined to the location in Websphere Application Server where the license jar file above is located, and will only be used if the **DB2UNIVERSAL_JDBC_DRIVER_PATH** is not set. DB2 users should ensure that **DB2UNIVERSAL_JDBC_DRIVER_PATH** is set to avoid loading multiple versions of the db2jcc.jar file.

     Note: **DB2UNIVERSAL_JDBC_DRIVER_PATH** is a WebSphere environment variable that you must set to point to the location of db2jcc.jar file (that comes with DB2). This variable is set only if you create a db2 provider.

     Note: Cloudscape requires only db2jcc_license_c.jar; however, WebSphere Application Server uses db2jcc_license_cu.jar because this works for both DB2 UDB and Cloudscape.

   Requires **DataStoreHelper** class:

   `com.ibm.websphere.rsadapter.CloudscapeNetworkServerDataStoreHelper`

   Note: The administrative console incorrectly lists the DB2UniversalDataStoreHelper as the default value for the **DataStoreHelper** class. You must change the default value to

com.ibm.websphere.rsadapter.CloudscapeNetworkServerDataStoreHelper. Also change the custom properties, using the instructions in the customer property section.

Requires a valid authentication alias.

Requires properties:

- **databaseName** The name of the database from which the data source obtains connections. If you do not specify a fully qualified path name, Application Server uses the default location of WAS_HOME./cloudscape (or the equivalent default for a UNIX or LINUX environment).
  - Example database path name for Windows: c:*temp**sampleDB*
  - Example database path name for UNIX or LINUX: /*tmp*/*sampleDB*

  If no database currently exists for the path name you want to specify, simply append ;create=true to the path name to create a database dynamically. (For example: c:*temp**sampleDB*;create=true)
- **driverType** Only the Type 4 driver is allowed.
- **serverName** The TCP/IP address or the host name for the Distributed Relational Database Architecture (DRDA) server.
- **portNumber** The TCP/IP port number where the DRDA server resides. The default value is port ***1527***.
- **retrieveMessagesfromServerOnGetMessage** This property is required by WebSphere Application Server, not the database. The default value is ***false***. You must set the value of this property to ***true***, to enable text retrieval using the **SQLException.getMessage()** method.

See the Cloudscape setup instructions for more information on configuring the Cloudscape Network Server.

For more information on IBM Cloudscape, visit the Cloudscape Web site at: http://www.ibm.com/software/data/cloudscape/

Informix

1. **Informix JDBC Driver**

   The Informix JDBC Driver is a Type 4 JDBC driver that provides JDBC access to the Informix database.

   Informix JDBC Driver supports one phase data source:

   com.informix.jdbcx.IfxConnectionPoolDataSource

   Requires JDBC driver files:

   ifxjdbc.jar
   ifxjdbcx.jar

   Requires **DataStoreHelper** class:

   com.ibm.websphere.rsadapter.InformixDataStoreHelper

   Requires a valid authentication alias.

   Requires properties:

   - **serverName** The name of the Informix instance on the server. Example: *ol_myserver*.
   - **portNumber** The port on which the instances listen. Example: *1526*.
   - **ifxIFXHOST** Either the IP address or the host name of the machine that is running the Informix database to which you want to connect. Example: *myserver.mydomain.com*.
   - **databaseName** The name of the database from which the data source obtains connections. Example: *Sample*.

- **informixLockModeWait** Although not required, this property enables you to set the number of seconds that Informix software waits for a lock. By default, Informix code throws an exception if it cannot immediately acquire a lock. Example: *2*.

2. **Informix JDBC Driver (XA)**

   The Informix JDBC Driver (XA) is a Type 4 JDBC driver that provides XA-compliant JDBC access to the Informix database.

   Informix JDBC Driver (XA) supports two phase data source:

   com.informix.jdbcx.IfxXADataSource

   Requires JDBC driver files:

   ifxjdbc.jar
   ifxjdbcx.jar

   Requires **DataStoreHelper** class:

   `com.ibm.websphere.rsadapter.InformixDataStoreHelper`

   Requires a valid authentication alias.

   Requires properties:
   - **serverName** The name of the Informix instance on the server. Example: *ol_myserver*.
   - **portNumber** The port on which the instances listen. Example: *1526*.
   - **ifxIFXHOST** Either the IP address or the host name of the machine that is running the Informix database to which you want to connect. Example: *myserver.mydomain.com*.
   - **databaseName** The name of the database from which the data source obtains connections. Example: *Sample*.
   - **informixLockModeWait** Although not required, this property enables you to set the number of seconds that Informix software waits for a lock. By default, Informix code throws an exception if it cannot immediately acquire a lock. Example: *2*.

For more information on Informix, visit the Informix Web site at: http://www.ibm.com/software/data/informix/

Sybase

1. **Sybase JDBC Driver**

   The Sybase JDBC Driver is a Type 4 JDBC driver that provides JDBC access to the Sybase database.

   Sybase JDBC Driver supports one phase data source:

   com.sybase.jdbc2.jdbc.SybConnectionPoolDataSource

   Requires JDBC driver files: **jconn2.jar**.

   Requires **DataStoreHelper** class:

   `com.ibm.websphere.rsadapter.SybaseDataStoreHelper`

   Requires a valid authentication alias.

   Requires properties:
   - **serverName** The name of the database server. Example: *myserver.mydomain.com*.
   - **databaseName** The name of the database from which the data source obtains connections. Example: *Sample*.
   - **portNumber** The TCP/IP port number through which all communications to the server take place. Example: *4100*.
   - **connectionProperties** A custom property required for applications containing EJB 2.0 enterprise beans. Value: `SELECT_OPENS_CURSOR=true`(Type: java.lang.String)

2. **Sybase JDBC Driver (XA)**

   The Sybase JDBC Driver (XA) is a Type 4 JDBC driver that provides XA-compliant JDBC access to the Sybase database.

   Sybase JDBC Driver (XA) supports two phase data source:

   com.sybase.jdbc2.jdbc.SybXADataSource

   Requires JDBC driver files: **jconn2.jar**.

   Requires **DataStoreHelper** class:

   com.ibm.websphere.rsadapter.SybaseDataStoreHelper

   Requires a valid authentication alias.

   Requires properties:
   - **serverName** The name of the database server. Example: *myserver.mydomain.com*
   - **databaseName** The name of the database from which the data source obtains connections. Example: *Sample*.
   - **portNumber** The TCP/IP port number through which all communications to the server take place. Example: *4100*.
   - **connectionProperties** A custom property required for applications containing EJB 2.0 enterprise beans. Value: SELECT_OPENS_CURSOR=true(Type: java.lang.String)

For more information on Sybase, visit the Sybase Web site at: http://www.sybase.com/

Oracle

1. **Oracle JDBC Driver**

   The Oracle JDBC Driver provides JDBC access to the Oracle database. This JDBC driver supports both Type 2 JDBC access and Type 4 JDBC access.

   Oracle JDBC Driver supports one phase data source:

   oracle.jdbc.pool.OracleConnectionPoolDataSource

   Requires JDBC driver files: **ojdbc14.jar**. (Note: If you require Oracle trace, use **ojdbc14_g.jar**.)

   Requires **DataStoreHelper** class:

   com.ibm.websphere.rsadapter.OracleDataStoreHelper

   (Note: If you are running Oracle10g, use com.ibm.websphere.rsadapter.Oracle10gDataStoreHelper.

   Requires a valid authentication alias.

   Requires properties:
   - **URL** The URL that indicates the database from which the data source obtains connections. Example: *jdbc:oracle:thin:@myServer:1521:myDatabase*, where *myServer* is the server name, *1521* is the port it is using for communication, and *myDatabase* is the database name.

2. **Oracle JDBC Driver (XA)**

   The Oracle JDBC Driver (XA) provides XA-compliant JDBC access to the Oracle database. This JDBC driver supports both Type 2 JDBC access and Type 4 JDBC access.

   Oracle JDBC Driver (XA) supports two phase data source:

   oracle.jdbc.xa.client.OracleXADataSource

   Requires JDBC driver files: **ojdbc14.jar**. (Note: If you require Oracle trace, use **ojdbc14_g.jar**.)

   Requires **DataStoreHelper** class:

   com.ibm.websphere.rsadapter.OracleDataStoreHelper

Requires a valid authentication alias.

Requires properties:

- **URL** The URL that indicates the database from which the data source obtains connections. Example: *jdbc:oracle:thin:@myServer:1521:myDatabase*, where *myServer* is the server name, *1521* is the port it is using for communication, and *myDatabase* is the database name.

For more information on Oracle, visit the Oracle Web site at:
http://www.oracle.com/

MS SQL Server

1. **DataDirect ConnectJDBC type 4 driver for MS SQL Server**

   DataDirect ConnectJDBC type 4 driver for MS SQL Server is a Type 4 JDBC driver that provides JDBC access to the MS SQL Server database. This provider is for use only with the Connect JDBC driver purchased from DataDirect Technologies.

   This JDBC provider supports this data source:

   com.ddtek.jdbcx.sqlserver.SQLServerDataSource

   Requires JDBC driver files:

   sqlserver.jar,
   base.jar and **util.jar**

   (The **spy.jar** file is optional. You need this file to enable spy logging. The **spy.jar** file is not in the same directory as the other three jar files. Instead, it is located in the **../spy/** directory.)

   Requires **DataStoreHelper** class:

   com.ibm.websphere.rsadapter.ConnectJDBCDataStoreHelper

   Requires a valid authentication alias.

   Requires properties:

   - **serverName** The name of the server in which MS SQL Server resides. Example: *myserver.mydomain.com*
   - **portNumber** The TCP/IP port that MS SQL Server uses for communication. Port 1433 is the default.
   - **databaseName** The name of the database from which the data source obtains connections. Example: *Sample*.

2. **DataDirect ConnectJDBC type 4 driver for MS SQL Server (XA)**

   DataDirect ConnectJDBC type 4 driver for MS SQL Server (XA) is a Type 4 JDBC driver which provides XA-compliant JDBC access to the MS SQL Server database. This provider is for use only with the Connect JDBC driver purchased from DataDirect Technologies.

   This JDBC provider supports this data source:

   com.ddtek.jdbcx.sqlserver.SQLServerDataSource.

   Requires JDBC driver files:

   sqlserver.jar,
   base.jar and **util.jar.**

   (The **spy.jar** file is optional. You need this file to enable spy logging. The **spy.jar** file is not in the same directory as the other three jar files. Instead, it is located in the **../spy/** directory.)

   Requires **DataStoreHelper** class:

   com.ibm.websphere.rsadapter.ConnectJDBCDataStoreHelper

   Requires a valid authentication alias.

   Requires properties:

- **serverName** The name of the server in which MS SQL Server resides. Example: *myserver.mydomain.com*
- **portNumber** The TCP/IP port that MS SQL Server uses for communication. Port 1433 is the default.
- **databaseName** The name of the database from which the data source obtains connections. Example: *Sample*.

For more information on the DataDirect ConnectJDBC driver, visit the DataDirect Web site at: http://www.datadirect-technologies.com/

3. **DataDirect SequeLink type 3 JDBC driver for MS SQL Server** -- Deprecated

   Because this JDBC provider is deprecated in WebSphere Application Server Version 6.0, it is no longer an available option in the administrative console. In its place, use one of the Connect JDBC providers, which are described previously in this section.

   DataDirect SequeLink type 3 JDBC driver for MS SQL Server is a type 3 JDBC driver that provides JDBC access to MS SQL Server via SequeLink server.

   This JDBC provider supports this data source:

   com.ddtek.jdbcx.sequelink.SequeLinkDataSource

   Requires JDBC driver files:

   sljc.jar and
   spy-sl.jar

   (The JDBC driver shipped with WebSphere Application Server requires the **sljc.jar** and the **spy-sl.jar** files. The JDBC driver purchased from DataDirect requires the **sljc.jar** and the **spy.jar** files. The **spy.jar** and **spy-sl.jar** files are optional. You need these files to enable spy logging.)

   Requires **DataStoreHelper** class:

   com.ibm.websphere.rsadapter.SequeLinkDataStoreHelper

   Requires a valid authentication alias.

   Requires properties:
   - **serverName** The name of the server in which SequeLink Server resides. Example: *myserver.mydomain.com*
   - **portNumber** The TCP/IP port that SequeLink Server uses for communication. By default, SequeLink Server uses port 19996.
   - **databaseName** The name of the database from which the data source obtains connections. Example: *Sample*.

4. **DataDirect SequeLink type 3 JDBC driver for MS SQL Server (XA)** -- Deprecated

   Because this JDBC provider is deprecated in WebSphere Application Server Version 6.0, it is no longer an available option in the administrative console. In its place, use one of the Connect JDBC providers, which are described previously in this section.

   DataDirect SequeLink type 3 JDBC driver for MS SQL Server (XA) is a type 3 JDBC driver that provides XA-compliant JDBC access to MS SQL Server via the SequeLink server.

   This JDBC provider supports this data source:

   com.ddtek.jdbcx.sequelink.SequeLinkDataSource

   Requires JDBC driver files:

   sljc.jar and
   spy-sl.jar

   (The JDBC driver shipped with WebSphere Application Server requires the **sljc.jar** and the **spy-sl.jar** files. The JDBC driver purchased from DataDirect requires the **sljc.jar** and the **spy.jar** files. The **spy.jar** and **spy-sl.jar** files are optional. You need these files to enable spy logging.)

Requires **DataStoreHelper** class:

`com.ibm.websphere.rsadapter.SequeLinkDataStoreHelper`

Requires a valid authentication alias.

Requires properties:

- **serverName** The name of the server in which SequeLink Server resides. Example: *myserver.mydomain.com*
- **portNumber** The TCP/IP port that SequeLink Server uses for communication. By default, SequeLink Server uses port 19996.
- **databaseName** The name of the database from which the data source obtains connections. Example: *Sample*.

Both of the WebSphere-embedded SequeLink JDBC drivers require installation of SequeLink Server on all machines running MS SQL Server. See the readme.html file found in the DataDirect folder on the WebSphere Application Server CD for instructions on how to install SequeLink Server. (Only install SequeLink Server from the WebSphere Application Server CD if you are using the SequeLink JDBC driver embedded in WebSphere. Otherwise, install a copy of SequeLink Server purchased from DataDirect Technologies.)

From the following FTP site, you can download the latest patches and upgrades for the version of SequeLink Server that is used with the WebSphere-embedded SequeLink JDBC driver:

`ftp://ftp.software.ibm.com/software/websphere/info/tools/DataDirect/datadirect.htm`

For more information on the DataDirect SequeLink type 3 JDBC driver, visit the DataDirect Web site at:

`http://www.datadirect-technologies.com/`

5. **Microsoft JDBC driver for MSSQLServer 2000** -- Deprecated

   Because this JDBC provider is deprecated in WebSphere Application Server Version 6.0, it is no longer an available option in the administrative console. In its place, use one of the Connect JDBC providers, which are described previously in this section.

   Microsoft JDBC driver for MSSQLServer 2000 is a type 4 JDBC driver that provides JDBC access to the MS SQL Server database.

   This JDBC provider supports this data source:

   com.microsoft.jdbcx.sqlserver.SQLServerDataSource

   Requires JDBC driver files:

   mssqlserver.jar,
   msbase.jar and **msutil.jar**

   (The **spy.jar** file is optional. You need it to enable spy logging. However, Microsoft does not ship the **spy.jar** file. Contact Microsoft about this issue.)

   Requires **DataStoreHelper** class:

   `com.ibm.websphere.rsadapter.ConnectJDBCDataStoreHelper`

   Requires a valid authentication alias.

   Requires properties:

   - **serverName** The name of the server in which MS SQL Server resides. Example: *myserver.mydomain.com*
   - **portNumber** The TCP/IP port that MS SQL Server uses for communication. Port 1433 is the default.
   - **databaseName** The name of the database from which the data source obtains connections. Example: *Sample*.

6. **Microsoft JDBC driver for MSSQLServer 2000 (XA)** -- Deprecated

Because this JDBC provider is deprecated in WebSphere Application Server Version 6.0, it is no longer an available option in the administrative console. In its place, use one of the Connect JDBC providers, which are described previously in this section.

Microsoft JDBC driver for MSSQLServer 2000 (XA) is a type 4 JDBC driver that provides XA-compliant JDBC access to the MS SQL Server database.

This JDBC provider supports this data source:

com.microsoft.jdbcx.sqlserver.SQLServerDataSource

Requires JDBC driver files:

mssqlserver.jar,
msbase.jar and **msutil.jar**

(The **spy.jar** file is optional. You need it to enable spy logging. However, Microsoft does not ship the **spy.jar** file. Contact Microsoft about this issue.)

Requires **DataStoreHelper** class:

`com.ibm.websphere.rsadapter.ConnectJDBCDataStoreHelper`

Requires a valid authentication alias.

Requires properties:

- **serverName** The name of the server in which MS SQL Server resides. Example: *myserver.mydomain.com*
- **portNumber** The TCP/IP port that MS SQL Server uses for communication. Port 1433 is the default.
- **databaseName** The name of the database from which the data source obtains connections. Example: *Sample*.

For more information on the Microsoft JDBC driver, visit the Microsoft Web site at:

`http://www.microsoft.com/sql`

Related concepts

Data sources
Installed applications uses a *data source* to access the data from the database.

Resource adapter
A resource adapter is a system-level software driver that a Java application uses to connect to an enterprise information system (EIS).

JDBC providers
Installed applications that must interact with *relational* databases use JDBC providers for data access. Together, the JDBC provider and data source objects are functionally equivalent to the J2EE Connector Architecture (JCA) connection factory (which provides access to non-relational databases).

Related tasks

Creating and configuring a JDBC provider and data source
This topic outlines the process for configuring access to a relational database, and provides links to more detailed instruction.

Creating and configuring a JDBC provider using the administrative console

Creating and configuring a data source using the administrative console

Creating and configuring a JDBC provider and data source using the Java Management Extensions API

Example: Using the Java Management Extensions API to create a JDBC driver and data source for container-managed persistence

```
//
// "This program may be used, executed, copied, modified and distributed
// without royalty for the
// purpose of developing, using, marketing, or distributing."
//
// Product 5630-A36,  (C) COPYRIGHT International Business Machines Corp., 2001,
//  2002
// All Rights Reserved * Licensed Materials - Property of IBM
//

import java.util.*;
import javax.sql.*;
import javax.transaction.*;
import javax.management.*;
import java.io.*;

import com.ibm.websphere.management.*;
import com.ibm.websphere.management.configservice.*;
import com.ibm.ws.management.*;
import com.ibm.ws.exception.*;

/**
 * Creates a node-scoped resource.xml entry for a
 * DB2 for zOS Local JDBC Provider (RRS) DataSource
 * when WebSphere security is not enabled
 *
 * The datasource created is for CMP use.
 *
 * To run this example, the following must be done:
 *
 * 1)  Set the WAS_HOME environment variable to the location of
 *     your WebSphere Application Server for z/OS Configuration
 *     directory
 *
 *     Example:  export WAS_HOME=/WebSphereV5R1M0/AppServer
 *
 * 2)  Set the following environment variables:
 *
 *     export WAS_LIB=$WAS_HOME/lib
 *     export WAS_CLASSPATH=[DIRECTORY_CONTAINING_THIS_FILE]
 *     export WAS_CLASSPATH=$WAS_CLASSPATH:$WAS_LIB/jmxc.jar
 *     export WAS_CLASSPATH=$WAS_CLASSPATH:$WAS_LIB/wsexception.jar
 *     export WAS_CLASSPATH=$WAS_CLASSPATH:$WAS_LIB/admin.jar
 *     export WAS_CLASSPATH=$WAS_CLASSPATH:$WAS_LIB/wasjmx.jar
 *     export WAS_CLASSPATH=$WAS_CLASSPATH:$WAS_HOME/java/jre/lib/ext/mail.jar
 *     export WAS_CLASSPATH=$WAS_CLASSPATH:$WAS_LIB/ibmjlog.jar
 *
 * 3)  Execute the following commands:
 *
 *     javac -classpath $WAS_CLASSPATH CreateDataSourceCMP.java
 *     java -classpath $WAS_CLASSPATH CreateDataSourceCMP
 */
public class CreateDataSourceCMP {

    String dsName           = "MyDataSourceCMP";   // ds display name ,
                                                    // also jndi name and CF name
    String dbName           = "LOC1";           // database name
    String authDataAlias    = "IBMUSER";        // an authentication data alias
    String uid              = "IBMUSER";        // userid
    String pw               = "IBMUSER";        // password
    String dbclasspath      = "/db2beta/db2710/classes/db2j2classes.zip";
                                // path to the db driver
```

```
String dblibpath        = "/db2beta/db2710/lib";
                          // path to the db lib directory

/**
 * Main method.
 */
public static void main(String[] args) {
    CreateDataSourceCMP cds = new CreateDataSourceCMP();
    try {
        cds.run(args);
    } catch (com.ibm.ws.exception.WsException ex) {
        System.out.println("Caught this " + ex);
        ex.printStackTrace();
        ex.getCause().printStackTrace();
    } catch (Exception ex) {
        System.out.println("Caught this " + ex);
        ex.printStackTrace();
    }
}

/**
 * This method creates the datasource using JMX.
 * The datasource created here is only written into resources.xml.
 * It is not bound into namespace until the server is restarted, or an
   application started
 */
public void run(String[] args) throws Exception {
    try {
        // Initialize the AdminClient.
        Properties adminProps = new Properties();
        adminProps.setProperty(AdminClient.CONNECTOR_TYPE, AdminClient.
              CONNECTOR_TYPE_SOAP);
        adminProps.setProperty(AdminClient.CONNECTOR_HOST, "localhost");
        adminProps.setProperty(AdminClient.CONNECTOR_PORT, "8880");
        AdminClient adminClient = AdminClientFactory.createAdminClient
                  (adminProps);

        // Get the ConfigService implementation.
        com.ibm.websphere.management.configservice.ConfigServiceProxy
            configService =
            new com.ibm.websphere.management.configservice.ConfigServiceProxy
                 (adminClient);

        Session session = new Session();

        // Use this group to add to the node scoped resource.xml.
        ObjectName node1 = ConfigServiceHelper.createObjectName(null, "Node",
                     null);
        ObjectName[] matches = configService.queryConfigObjects(session, null,
                           node1, null);
        node1 = matches[0];     // use the first node found

        // Use this group to add to the server1 scoped resource.xml.
        ObjectName server1 = ConfigServiceHelper.createObjectName(null,
                    "Server", "server1");
        matches = configService.queryConfigObjects(session, null, server1,
                    null);
        server1 = matches[0];   // use the first server found

        // Create the JDBCProvider
        String providerName = "My DB2 for zOS Local JDBC Provider (RRS)
            for CMP";
        System.out.println("Creating JDBCProvider " + providerName );

        // Prepare the attribute list
        AttributeList provAttrs = new AttributeList();
        provAttrs.add(new Attribute("name", providerName));
```

```
provAttrs.add(new Attribute("implementationClassName",
  "com.ibm.db2.jcc.DB2ConnectionPoolDataSource"));
provAttrs.add(new Attribute("description","Legacy DB2 for z/OS
   driver using RRS"));

//create it
ObjectName jdbcProv = configService.createConfigData(session,node1,
   "JDBCProvider", "resources.jdbc:JDBCProvider", provAttrs);
// now plug in the classpath
configService.addElement(session,jdbcProv,"classpath",dbclasspath,-1);
configService.addElement(session,jdbcProv,"nativepath",dblibpath,-1);

// Search for RRA so we can link it to the datasource
ObjectName rra = ConfigServiceHelper.createObjectName(null,
     "J2CResourceAdapter", null);
matches = configService.queryConfigObjects(session, node1, rra, null);
rra = matches[0]; // use the first J2CResourceAdapter segment
                     for builtin_rra

// Prepare the attribute list
AttributeList dsAttrs = new AttributeList();
dsAttrs.add(new Attribute("name", dsName));
dsAttrs.add(new Attribute("jndiName", "jdbc/" + dsName));
dsAttrs.add(new Attribute("datasourceHelperClassname",
   "com.ibm.websphere.rsadapter.DB2DataStoreHelper"));
dsAttrs.add(new Attribute("statementCacheSize", new Integer(10)));
dsAttrs.add(new Attribute("relationalResourceAdapter", rra));
// this is where we make the link to "builtin_rra"
dsAttrs.add(new Attribute("description", "JDBC Datasource for CMP
                    usage"));
dsAttrs.add(new Attribute("authDataAlias", authDataAlias));

// Create the datasource
System.out.println("  **  Creating datasource");
ObjectName dataSource = configService.createConfigData
(session,jdbcProv,"DataSource", "resources.jdbc:DataSource",dsAttrs);

// Add a propertySet.
AttributeList propSetAttrs = new AttributeList();
ObjectName resourcePropertySet = configService.createConfigData
  (session,dataSource,"propertySet","",propSetAttrs);

// Add resourceProperty databaseName
AttributeList propAttrs1 = new AttributeList();
propAttrs1.add(new Attribute("name", "databaseName"));
propAttrs1.add(new Attribute("type", "java.lang.String"));
propAttrs1.add(new Attribute("value", dbName));

configService.addElement(session,resourcePropertySet,
     "resourceProperties",propAttrs1,-1);

// Now Create the corresponding J2CResourceAdapter Connection
         Factory object.
ObjectName jra = ConfigServiceHelper.createObjectName(null,
     "J2CResourceAdapter",null);

// Get all the J2CResourceAdapter, and I want to add my datasource
System.out.println("  **  Get all J2CResourceAdapter's");
ObjectName[] jras = configService.queryConfigObjects(session, node1,
            jra, null);

int i=0;

for (;i< jras.length;i++) {
    System.out.println(ConfigServiceHelper.getConfigDataType(jras[i])+
              " " + i + " = "
        + jras[i].getKeyProperty(SystemAttributes._WEBSPHERE_CONFIG_
```

```
                    DATA_DISPLAY_NAME)
                + "\nFrom scope ="
                + jras[i].getKeyProperty(SystemAttributes._WEBSPHERE_CONFIG_
                        DATA_ID));
            // quit on the first builtin_rra
            if (jras[i].getKeyProperty(SystemAttributes._WEBSPHERE_CONFIG_
                  DATA_DISPLAY_NAME).equals("WebSphere Relational Resource
                     Adapter")) {
                break;
            }
        }


        if (i >= jras.length) {
            System.out.println("Did not find builtin_rra J2CResourceAdapter
               object creating CF anyways" );
        } else {
            System.out.println("Found builtin_rra J2CResourceAdapter object
               at index " + i + " creating CF" );
        }

        // Prepare the attribute list
        AttributeList cfAttrs = new AttributeList();
        cfAttrs.add(new Attribute("name", dsName + "_CF"));
        cfAttrs.add(new Attribute("authMechanismPreference","BASIC_PASSWORD"));
        cfAttrs.add(new Attribute("authDataAlias",authDataAlias));
        cfAttrs.add(new Attribute("cmpDatasource", dataSource ));
          // this is where we make the link to DataSource's xmi:id
        ObjectName cf = configService.createConfigData(session,jras[i],
            "CMPConnectorFactory", "resources.jdbc:CMPConnectorFactory",
                cfAttrs);

        // ===== start Security section
        System.out.println("Creating an authorization data alias " +
              authDataAlias);

        // Find the parent security object
        ObjectName security = ConfigServiceHelper.createObjectName(null,
                "Security", null);
        ObjectName[] securityName = configService.queryConfigObjects(session,
                 null, security, null);
        security=securityName[0];

        // Prepare the attribute list
        AttributeList authDataAttrs = new AttributeList();
        authDataAttrs.add(new Attribute("alias", authDataAlias));
        authDataAttrs.add(new Attribute("userId", uid));
        authDataAttrs.add(new Attribute("password", pw));
        authDataAttrs.add(new Attribute("description","Auto created alias
                for datasource"));

        //create it
        ObjectName authDataEntry = configService.createConfigData
           (session,security,"authDataEntries", "JAASAuthData",authDataAttrs);
        // ===== end Security section

        // Save the session
        System.out.println("Saving session" );
        configService.save(session, false);

        // reload resources.xml to bind the new datasource into the name space
        reload(adminClient,true);
    } catch (Exception ex) {
        ex.printStackTrace(System.out);
        throw ex;
    }
}
```

```
    /**
     * Get the DataSourceConfigHelperMbean and call reload() on it
     *
     * @param adminClient
     * @param verbose true - print messages to stdout
     */
    public void reload(AdminClient adminClient,boolean verbose) {
        if (verbose) {
            System.out.println("Finding the Mbean to call reload()");
        }

        // First get the  Mbean
        ObjectName handle = null;
        try {
            ObjectName queryName = new ObjectName("WebSphere:type=
                   DataSourceCfgHelper,*");
            Set s = adminClient.queryNames(queryName, null);
            Iterator iter = s.iterator();
            if (iter.hasNext()) handle = (ObjectName)iter.next();
        } catch (MalformedObjectNameException mone) {
            System.out.println("Check the program variable queryName" + mone);
        } catch (com.ibm.websphere.management.exception.ConnectorException ce) {
            System.out.println("Cannot connect to the application server" + ce);
        }

        if (verbose) {
            System.out.println("Calling reload()");
        }
        Object result = null;
        try {
            result = adminClient.invoke(handle, "reload", new Object[] {}, new
                  String[] {});
        } catch (MBeanException mbe) {
            if (verbose) {
                System.out.println("\tMbean Exception calling reload" + mbe);
            }
        } catch (InstanceNotFoundException infe) {
            System.out.println("Cannot find reload ");
        } catch (Exception ex) {
            System.out.println("Exception occurred calling reload()"  + ex);
        }

        if (result==null && verbose) {
            System.out.println("OK reload()");
        }
    }
}
```

Related concepts

Resource adapter
A resource adapter is a system-level software driver that a Java application uses to connect to an enterprise information system (EIS).

Connection factory
An application component uses a *connection factory* to access a connection instance, which the component then uses to connect to the underlying enterprise information system (EIS).

JDBC providers
Installed applications that must interact with *relational* databases use JDBC providers for data access. Together, the JDBC provider and data source objects are functionally equivalent to the J2EE Connector Architecture (JCA) connection factory (which provides access to non-relational databases).

Data sources
Installed applications uses a *data source* to access the data from the database.

Example: Using the Java Management Extensions API to create a JDBC driver and data source for bean-managed persistence, session beans, or servlets

```
// "This program may be used, executed, copied, modified and distributed
//       without royalty for the
// purpose of developing, using, marketing, or distributing."
//
// Product 5630-A36,  (C) COPYRIGHT International Business Machines Corp., 2001,
// 2002
// All Rights Reserved * Licensed Materials - Property of IBM
//
import java.util.*;
import javax.sql.*;
import javax.transaction.*;
import javax.management.*;

import com.ibm.websphere.management.*;
import com.ibm.websphere.management.configservice.*;
import com.ibm.ws.exception.WsException;

/**
 * Creates a node-scoped resource.xml entry for a
 * DB2 for zOS Local JDBC Provider (RRS) DataSource
 * when WebSphere security is not enabled
 *
 * To run this example, the following must be done:
 *
 * 1)  Set the WAS_HOME environment variable to the location of
 *     your WebSphere Application Server for z/OS Configuration
 *     directory
 *
 *     Example:  export WAS_HOME=/WebSphereV5R1M0/AppServer
 *
 * 2)  Set the following environment variables:
 *
 *     export WAS_LIB=$WAS_HOME/lib
 *     export WAS_CLASSPATH=[DIRECTORY_CONTAINING_THIS_FILE]
 *     export WAS_CLASSPATH=$WAS_CLASSPATH:$WAS_LIB/jmxc.jar
 *     export WAS_CLASSPATH=$WAS_CLASSPATH:$WAS_LIB/wsexception.jar
 *     export WAS_CLASSPATH=$WAS_CLASSPATH:$WAS_LIB/admin.jar
 *     export WAS_CLASSPATH=$WAS_CLASSPATH:$WAS_LIB/wasjmx.jar
 *     export WAS_CLASSPATH=$WAS_CLASSPATH:$WAS_HOME/java/jre/lib/ext/mail.jar
 *     export WAS_CLASSPATH=$WAS_CLASSPATH:$WAS_LIB/ibmjlog.jar
 *
 * 3)  Execute the following commands:
 *
 *     javac -classpath $WAS_CLASSPATH CreateDataSourceBMP.java
 *     java -classpath $WAS_CLASSPATH CreateDataSourceBMP
 */
 public class CreateDataSourceBMP {

    String dsName = "MyDataSourceBMP";  // ds display name , also jndi
          name and CF name
    String dbName = "LOC1";             // database name
    String authDataAlias = "IBMUSER";   // an authentication data alias
    String uid = "IBMUSER";             // userid
    String pw  = "IBMUSER";             // password
    String dbclasspath = "/db2beta/db2710/classes/db2j2classes.zip";
          // path to the db driver
    String dblibpath = "/db2beta/db2710/lib";
        // path to the db native library directory

    /**
     * Main method.
     */
```

```
public static void main(String[] args) {
    CreateDataSourceBMP cds = new CreateDataSourceBMP();
    try {
        cds.run(args);
    } catch (com.ibm.ws.exception.WsException ex) {
        System.out.println("Caught this " + ex   );
        ex.printStackTrace();
    } catch (Exception ex) {
        System.out.println("Caught this " + ex );
        ex.printStackTrace();
    }
}

/**
 * This method creates the datasource using JMX.
 *
 * The datasource created here is only written into resources.xml.
 * It is not bound into namespace until the server is restarted, or
        an application started
 */
public void run(String[] args) throws Exception {

    try {
        // Initialize the AdminClient.
        Properties adminProps = new Properties();
        adminProps.setProperty(AdminClient.CONNECTOR_TYPE,
              AdminClient.CONNECTOR_TYPE_SOAP);
        adminProps.setProperty(AdminClient.CONNECTOR_HOST, "localhost");
        adminProps.setProperty(AdminClient.CONNECTOR_PORT, "8880");
        AdminClient adminClient = AdminClientFactory.createAdminClient
              (adminProps);

        // Get the ConfigService implementation.
        com.ibm.websphere.management.configservice.ConfigServiceProxy
                  configService =
        new com.ibm.websphere.management.configservice.ConfigServiceProxy
                (adminClient);

        Session session = new Session();

        // Use this group to add to the node scoped resource.xml.
        ObjectName node1 = ConfigServiceHelper.createObjectName(null,
               "Node", null);
        ObjectName[] matches = configService.queryConfigObjects(session,
                null, node1, null);
        node1 = matches[0];     // use the first node found

        // Use this group to add to the server1 scoped resource.xml.
        ObjectName server1 = ConfigServiceHelper.createObjectName(null,
                      "Server", "server1");
        matches = configService.queryConfigObjects(session, null, server1,
                        null);
        server1 = matches[0];   // use the first server found

        // Create the JDBCProvider
        String providerName = "My DB2 for zOS Local JDBC Provider (RRS) for
                 BMP";
        System.out.println("Creating JDBCProvider " + providerName );

        // Prepare the attribute list
        AttributeList provAttrs = new AttributeList();
        provAttrs.add(new Attribute("name", providerName));
        provAttrs.add(new Attribute("implementationClassName",
                 "com.ibm.db2.jcc.DB2ConnectionPoolDataSource"));
        provAttrs.add(new Attribute("description","Legacy DB2 for z/OS
                   driver using RRS"));
```

```
//create it
ObjectName jdbcProv = configService.createConfigData(session,node1,
        "JDBCProvider", "resources.jdbc:JDBCProvider",provAttrs);
// now plug in the classpath
configService.addElement(session,jdbcProv,"classpath",dbclasspath,-1);
configService.addElement(session,jdbcProv,"nativepath",dblibpath,-1);

// Search for RRA so we can link it to the datasource
ObjectName rra = ConfigServiceHelper.createObjectName(null,
             "J2CResourceAdapter", null);
matches = configService.queryConfigObjects(session, node1, rra, null);
rra = matches[0]; // use the first J2CResourceAdapter segment for
           builtin_rra

// Prepare the attribute list
AttributeList dsAttrs = new AttributeList();
dsAttrs.add(new Attribute("name", dsName));
dsAttrs.add(new Attribute("jndiName", "jdbc/" + dsName));
dsAttrs.add(new Attribute("datasourceHelperClassname",
          "com.ibm.websphere.rsadapter.DB2DataStoreHelper"));
dsAttrs.add(new Attribute("statementCacheSize", new Integer(10)));
dsAttrs.add(new Attribute("relationalResourceAdapter", rra));
     // this is where we make the link to "builtin_rra"
dsAttrs.add(new Attribute("description", "JDBC Datasource for
            BMP usage"));
dsAttrs.add(new Attribute("authDataAlias",authDataAlias));

// Create the datasource
System.out.println("  **  Creating datasource");
ObjectName dataSource = configService.createConfigData(session,
    jdbcProv,"DataSource", "resources.jdbc:DataSource",dsAttrs);

// Add a propertySet.
AttributeList propSetAttrs = new AttributeList();
ObjectName resourcePropertySet =configService.createConfigData
     (session,dataSource,"propertySet","",propSetAttrs);

// Add resourceProperty databaseName
AttributeList propAttrs1 = new AttributeList();
propAttrs1.add(new Attribute("name", "databaseName"));
propAttrs1.add(new Attribute("type", "java.lang.String"));
propAttrs1.add(new Attribute("value", dbName));

configService.addElement(session,resourcePropertySet,
        "resourceProperties",propAttrs1,-1);


// ===== start Security section
System.out.println("Creating an authorization data alias " +
           authDataAlias);

// Find the parent security object
ObjectName security = ConfigServiceHelper.createObjectName(null,
         "Security", null);
ObjectName[] securityName = configService.queryConfigObjects(session,
         null, security, null);
security=securityName[0];

// Prepare the attribute list
AttributeList authDataAttrs = new AttributeList();
authDataAttrs.add(new Attribute("alias", authDataAlias));
authDataAttrs.add(new Attribute("userId", uid));
authDataAttrs.add(new Attribute("password", pw));
authDataAttrs.add(new Attribute("description","Auto created alias
         for datasource"));

//create it
```

```
            ObjectName authDataEntry = configService.createConfigData(session,
                 security,"authDataEntries", "JAASAuthData",authDataAttrs);
            // ===== end Security section

            // Save the session
            System.out.println("Saving session" );
            configService.save(session, false);

            // reload resources.xml
            reload(adminClient,true);

        } catch (Exception ex) {
            ex.printStackTrace(System.out);
            throw ex;
        }
    }

    /**
     * Get the DataSourceConfigHelperMbean and call reload() on it
     *
     * @param adminClient
     * @param verbose true - print messages to stdout
     */
    public void reload(AdminClient adminClient,boolean verbose) {
        if (verbose) {
            System.out.println("Finding the Mbean to call reload()");
        }

        // First get the  Mbean
        ObjectName handle = null;
        try {
            ObjectName queryName = new ObjectName("WebSphere:type=
                      DataSourceCfgHelper,*");
            Set s = adminClient.queryNames(queryName, null);
            Iterator iter = s.iterator();
            if (iter.hasNext()) handle = (ObjectName)iter.next();
        } catch (MalformedObjectNameException mone) {
            System.out.println("Check the program variable queryName" + mone);
        } catch (com.ibm.websphere.management.exception.ConnectorException ce) {
            System.out.println("Cannot connect to the application server" + ce);
        }

        if (verbose) {
            System.out.println("Calling reload()");
        }
        Object result = null;
        try {
            result = adminClient.invoke(handle, "reload", new Object[] {},
                    new String[] {});
        } catch (MBeanException mbe) {
            if (verbose) {
                System.out.println("\tMbean Exception calling reload" + mbe);
            }
        } catch (InstanceNotFoundException infe) {
            System.out.println("Cannot find reload ");
        } catch (Exception ex) {
            System.out.println("Exception occurred calling reload()"  + ex);
        }

        if (result==null && verbose) {
            System.out.println("OK reload()");
        }
    }
}
```

Related concepts

JDBC providers
Installed applications that must interact with *relational* databases use JDBC providers for data access. Together, the JDBC provider and data source objects are functionally equivalent to the J2EE Connector Architecture (JCA) connection factory (which provides access to non-relational databases).

Data sources
Installed applications uses a *data source* to access the data from the database.

Messaging resources

Errors in messaging

What kind of problem are you seeing?
- javax.jms.JMSException: MQJMS2008: failed to open MQ queue in JVM log.

javax.jms.JMSException: MQJMS2008: failed to open MQ queue in JVM log

This error can occur when the MQ queue name is not defined in the internal Java Message Service (JMS) server queue names list. This problem can occur if a WebSphere Application Server queue destination is created, without adding the queue name to the internal JMS server queue names list.

To resolve this problem:
1. Open the WebSphere Application Server administrative console.
2. Click **Servers > Manage Application Servers >** *server_name*> **Server Components > JMS Servers**.
3. Add the queue name to the list.
4. Save the changes and restart the server.

For current information available from IBM Support on known problems and their resolution, see the IBM Support page.

IBM Support has documents that can save you time gathering information needed to resolve this problem. Before opening a PMR, see the IBM Support page.

Related tasks

Chapter 6, “Troubleshooting by task,” on page 233

Related reference

Troubleshooting testing and first time run problems

Debugging problems related to Java Message Service (JMS) support

You might encounter JMS-related errors in the WebSphere Application Server for z/OS environment. To debug these errors, use any of the various WebSphere console settings that controls the type of trace data collected.

com.ibm.ejs.*=all=enabled
Turns on all container tracing

com.ibm.ejs.j2c.*=all=enabled
Turns on all tracing for connector support in WebSphere Application Server for z/OS

Messaging=all=enabled
Collects trace data for the JMS and Message-driven bean (MDB) components of WebSphere Application Server for z/OS

If your installation configures WebSphere MQ to provide Java Message Service (JMS) support, you might need to use specific MQ tools for diagnosis:

- WebSphere variable *JMSApi=all=enabled*, which turns on all tracing for applications that use the JMS application programming interface (API).
- MQSC commands, which allow you to display information and perform other operations.
- The CSQUTIL utility program, which helps you to perform backup, restoration, and reorganization tasks, and to issue WebSphere MQ commands.

For more information about these diagnostic tools, refer to WebSphere MQ books, which are available through the Web site:

`http://www.ibm.com/software/ts/mqseries/library/`

The most useful for diagnostic information are:

- WebSphere MQ for z/OS Problem Determination, GC34-6054
- WebSphere MQ for z/OS Messages and Codes, SC34-6056
- WebSphere MQ Script (MQSC) Command Reference, SC34-6055
- WebSphere MQ for z/OS System Administration Guide, SC34-6053

All of these publications can be accessed from the IBM Publications Center.

Related tasks

"Steps for configuring WebSphere variables" on page 167

Errors connecting to WebSphere MQ and creating WebSphere MQ queue connection factory

The following exception may occur when trying to create the MDBListener instance:

```
6/23/03 22:45:58:232 CDT] 673106a8 MsgListenerPo W WMSG0049E:
Failed to start MDB PSSampleMDB against listener port SamplePubSubListenerPort
[6/23/03 22:47:58:289 CDT] 673106a8 FreePool E J2CA0046E:
Method createManagedConnctionWithMCWrapper caught an exception
during creation of the ManagedConnection for resource
JMS$SampleJMSQueueConnectionFactory, throwing ResourceAllocationException.
Original exception:  javax.resource.spi.ResourceAdapterInternalException:
 createQueueConnection failed
com.ibm.mq.MQException: MQJE001: An MQException occurred:
Completion Code 2, Reason 2009
MQJE003: IO error transmitting message buffer at
com.ibm.mq.MQManagedConnectionJ11.(MQManagedConnectionJ11.java:239)
```

This problem occurs because the MQ manager userid does not have write access to the `/tmp` directory. To correct this problem, before you use a Jacl procedure to configure WebSphere Application Server resources and install an application:

1. Ensure that all applications have write access to /tmp directory. Use the chmod 1777 command on the directory if necessary.
2. Create another subdirectory under `/tmp` (for example, `/tmp/mydir`). Use this directory as a "working directory" for the Jacl.
3. Restart the server.

Applications that use messaging on startup should start successfully.

Related tasks

Chapter 6, "Troubleshooting by task," on page 233

Related reference

Troubleshooting testing and first time run problems

Messaging component troubleshooting tips

Problems deploying or executing applications which use the WebSphere Application Server messaging capabilities

If you are having problems deploying or executing applications which use the WebSphere Application Server messaging capabilities, review these articles in the WebSphere Application Server information center:
- "Troubleshooting WebSphere messaging"
- "Troubleshooting message-driven beans" on page 346
- Troubleshooting service integration technologies

On a Linux platform, embedded messaging does not get removed after uninstalling the product

On a Linux platform, you may find that embedded messaging is not removed when you uninstall the product silently using the **SetRetainMQToTrue.active**=false option.

To prevent this problem, either issue the uninstall command from a local machine, or use ssh instead of telnet to logon to the machine where you issue the uninstall command.

To correct this problem after logging in, issue the following command for a real root login: su -

If none of these steps solves the problem, check to see if the problem has been identified and documented using the links in Diagnosing and fixing problems: Resources for learning.

For current information available from IBM Support on known problems and their resolution, see the IBM Support page.

IBM Support has documents that can save you time gathering information needed to resolve this problem. Before opening a PMR, see the IBM Support page.

Related tasks

Chapter 6, "Troubleshooting by task," on page 233

Related reference

Troubleshooting installation problems

Important file for message-driven beans

The following file in the WAS_HOME/temp directory is important for the operation of the WebSphere Application Server messaging service, so should not be deleted. If you do need to delete the WAS_HOME/temp directory or other files in it, ensure that you preserve the following file:

server_name-**durableSubscriptions.ser**

You should not delete this file, because the messaging service uses it to keep track of durable subscriptions for message-driven beans. If you uninstall an application that contains a message-driven bean, this file is used to unsubscribe the durable subscription.

Troubleshooting WebSphere messaging

Use this overview task to help resolve a problem that you think is related to the WebSphere Messaging.

To identify and resolve problems that you think are related to WebSphere Messaging, you can use the standard WebSphere Application Server troubleshooting facilities. If you encounter a problem that you think might be related to WebSphere Messaging, complete the following stages. Some problems and their troubleshooting are specific to whether you are using the embedded WebSphere Messaging or WebSphere MQ as the JMS provider.

1. Check for error messages about messaging. For example, check for error messages that indicate a problem with JMS resources.

   Check in the SYSPRINT or SYSOUT log.

   The associated message reference information provides an explanation and any user actions to resolve the problem.
2. Check for more informational and error messages that might provide a clue to a related problem. For example, if you have problems accessing JMS resources, check for more error messages and extra details about any problem associated with the JMS provider or with the service integration technologies that the default messaging provider uses.

   For messages related to the resource adapter (JMS) of the default messaging provider, look for the prefix: CWSJR. For messages related to service integration technologies, see the related reference topics.

   If your message-driven bean uses WebSphere Application Server version 5 JMS resources, look for the prefixes: MSGS and WMSG.
3. If you suspect that problems might be related to application use of message-driven beans, see "Troubleshooting message-driven beans" on page 346.
4. Check the Release Notes for specific problems and workarounds The section *Possible Problems and Suggested Fixes* of the Release Notes, available from the WebSphere Application Server library web site, is updated regularly to contain information about known defects and their workarounds. Check the latest version of the Release Notes for any information about your problem. If the Release Notes do not contain any information about your problem, you can also search the Technotes database on the WebSphere Application Server web site.
5. Check your JMS resource configurations If the messaging services seem to be running properly, check that the JMS resources have been configured correctly. For example, check that the JMS activation specification against which a message-driven bean is deployed has been configured correctly. For more information about configuring JMS resources, see Using messaging.
6. Get a detailed exception dump for messaging. If the information obtained in the preceding steps is still inconclusive, you can enable the application server debug trace for the "Messaging" group to provide a detailed exception dump.

   Related tasks

   Troubleshooting service integration technologies

Tips for troubleshooting WebSphere Messaging

This topic provides a set of tips to help you troubleshoot problems with WebSphere Messaging.

- An MDB listener fails to start
- Problems running JMS applications with security enabled
- Queue manager fails to stop on Redhat Linux
- Application server fails to start in zh_TW.EUC locale on Solaris
- "Server memory consumption and java.lang.OutOfMemoryError exception when processing JMS messages" on page 345

- "TopicConnectionFactory attributes clash error when using "Basic" WebSphere MQ broker (MA0C SupportPac broker)" on page 346

For messaging problems specific to WebSphere Application Server nodes, see the information center and the Application Servers support web site; for example: Tips for troubleshooting WebSphere messaging [version 5].

An MDB listener fails to start

If an MDB listener deployed against a listener port fails to start, you should see the following message:

```
WMSG0019E: Unable to start MDB Listener {0}, JMSDestination {1} : {2}
```

To troubleshoot the cause of an MDB listener not starting, check the following factors:

- Check that the administrative resources have been configured correctly; for example, use the administrative console to check the listener port properties: Destination JNDI name and Connection factory JNDI name. Check that other properties of the listener port, destination, and connection factory are correct.
- Check that the queue exists and has been added to the JMS server.
- Check that the queue manager and JMS server have started.
- Check that the Remote Queue Manager Listener has started.
- If security is enabled, check that a component-managed authentication alias has been specified on the queue connection factory or topic connection factory used by the message-driven bean. This is not required if security is not enabled.
- Check that the user ID used to start the MDB listener is appropriately authorized.

Problems running JMS applications with security enabled

When trying to run a JMS application with security enabled, you can encounter authentication problems indicated by error messages; for example: `WMSG0019E: Unable to start MDB Listener PSSampleMDB, JMSDestination Sample/JMS/listen : javax.jms.JMSSecurityException:`. [This example indicates that the security credentials supplied are not valid.]

The problem can be removed by doing one of the following:

- If the authentication mechanism is set to `Application`, then the application needs to supply valid credentials.
- If the authentication mechanism is set to `Container`, then you need to configure the JMS ConnectionFactory with a container-managed Authentication Alias and ensure that the associated username and password are valid.

MQJMS2013 invalid security authentication supplied for MQQueueManager

If using WebSphere MQ as a JMS Provider, with JMS connection using Bindings transport mode, and the user specified is not the current logged on user for the WebSphere Application Server process, then the JMS Bindings authentication by WebSphere MQ throws the error `MQJMS2013 invalid security authentication supplied for MQQueueManager`.

If you want to use WebSphere MQ as a JMS Provider, with JMS connection using Bindings transport mode, set the property **Transport type**=BINDINGS on the WebSphere MQ Queue Connection Factory. You must also choose one of the following options:

- To use security credentials, ensure that the user specified is the currently logged on user for the WebSphere Application Server process.

- Do not specify security credentials. On the WebSphere MQ Connection Factory, ensure that both the **Component-managed Authentication Alias** and the **Container-managed Authentication Alias** properties are not set.

For more information about messaging security, see Asynchronous messaging - security considerations.

Queue manager fails to stop on Redhat Linux

When trying to stop an application server on Redhat Linux, the queue manager can hang with a Java core dump, and the last message in the SystemOut.log file is `Stopping Queue manager...`.

This is caused by a known RedHat problem (https://bugzilla.linux.ibm.com/show_bug.cgi?id=2336), that was introduced in libstdc++-2.96-116.7.2 and beyond.

The workaround is to go back to the libstdc++-2.96-108.1 level.

Application server fails to start in zh_TW.EUC locale on Solaris

If you have set the locale to zh_TW.EUC on Solaris, and are using WebSphere MQ as a JMS provider, you can encounter problems that stop application servers starting up.

If you intend using WebSphere MQ as a JMS provider on Solaris, do not set the LANG and LC_ALL variables to zh_TW.EUC (Traditional Chinese locale) to avoid problems when starting application servers. Set the LANG and LC_ALL variables to zh_TW instead of zh_TW.EUC.

Server memory consumption and java.lang.OutOfMemoryError exception when processing JMS messages

Intensive processing of JMS messages using the default JMS provider (for example, significant concurrent processing of large messages) can cause a java.lang.OutOfMemoryError exception and cause the application server to terminate.

Processing of JMS messages by the default provider is performed by a messaging engine within the application server process, and therefore consumes memory from the application server's JVM heap. This is in contrast with Version 5 where the support for the embedded JMS provider run in a separate process.

If the amount of memory available to the application server's JVM heap has not been configured large enough to handle the effect of the number of concurrent producers or consumers of messages and the message size, then a java.lang.OutOfMemoryError exception is thrown and the application server terminates.

Solution: When preparing to deploy applications that process JMS messages using the default messaging provider, you should plan for the potential consumption of the application server's memory for message processing. You should take into account the potential number of concurrent processors or consumers of messages and the message size, then set the size of the application server's JVM heap to handle the effect.

For example, when preparing to deploy a message-driven bean that is to be used to process messages concurrently, you should plan for the potential consumption of the application server's memory by concurrent endpoints. Each endpoint that is concurrently processing a message request adds at least two times the message size to the server's JVM heap and can add more, especially if a two-phase transaction is in place.

You can configure the amount of memory available to the application server's JVM heap by setting the Initial Heap Size and Maximum Heap Size properties of the application server. For example, on the WebSphere administrative console panel: **Servers** → **Application servers** → *server_name* → **Java and Process Management** → **Process Definition** → **Java Virtual Machine**.

You can configure the number of concurrent MDB endpoints that can process messages by setting the Maximum concurrent endpoints property of the activation specification used to deploy the message-driven bean. For example, on the WebSphere administrative console panel: **Resources** → **JMS Providers** → **Default messaging** → **JMS activation specification** → *activationspec_name*.

TopicConnectionFactory attributes clash error when using "Basic" WebSphere MQ broker (MA0C SupportPac broker)

When creating a JMS topic subscriber using the WebSphere MQ messaging provider, the following error message occurs in the SystemOut.log file:

```
"WSVR0017E: Error encountered binding the J2EE resource, TopicConnectionFactory,
as <JNDI_NAME> from file:<RESOURCES_FILE> com.ibm.ws.runtime.component.binder.
ResourceBindingException: invalid configuration passed to resource binding logic.
REASON: Failed to create connection factory: Error raised constructing
AdminObject, error code:  TopicConnectionFactory attributes clash  :
TopicConnectionFactory attributes clash"
```

This problem is causes by the configuration of the JMS topic connection factory used to create the subscriber, which specifies a broker version of ″Basic″ and a message selection value of ″Broker″. The ″Basic″ WebSphere MQ broker (MA0C SupportPac broker) does not support ″Broker″ message selection. -

Solution: Change the JMS topic connection factory to specify a message selection value of ″Client″, which is the only supported value for the WebSphere MQ Basic broker (MA0C SupportPac broker).

Related tasks

Troubleshooting service integration messaging

Related information

Tips for troubleshooting the default messaging provider

Troubleshooting message-driven beans

Use this overview task to help resolve a problem that you think is related to message-driven beans.

Message-driven beans support uses the standard WebSphere Application Server troubleshooting facilities. If you encounter a problem that you think might be related to the message-driven beans, complete the following steps.

1. Check for error messages about message-driven beans. For example, check for error messages that indicate a problem with JMS resources, such as activation specifications or listener ports, that are used by message-driven beans.

   Check in the SYSPRINT or SYSOUT log.

The associated message reference information provides an explanation and any user actions to resolve the problem.

2. Check for more informational and error messages that might provide a clue to a related problem. For example, if you have problems accessing JMS resources, check for more error messages and extra details about any problem associated with the JMS provider or with the service integration technologies that the default messaging provider uses.

   For messages related to the resource adapter (JMS) of the default messaging provider, look for the prefix: CWSJR. For messages related to service integration technologies, see the related reference topics.

   If your message-driven bean uses WebSphere Application Server version 5 JMS resources, look for the prefixes: MSGS and WMSG.

3. Check the Release Notes for specific problems and workarounds The section *Possible Problems and Suggested Fixes* of the Release Notes, available from the WebSphere Application Server library web site, is updated regularly to contain information about known defects and their workarounds. Check the latest version of the Release Notes for any information about your problem. If the Release Notes does not contain any information about your problem, you can also search the Technotes database on the WebSphere Application Server web site.

4. If your message-driven bean is deployed against a lister port, check that message listener service has started. The message listener service is an extension to the JMS functions of the JMS provider. It provides a listener manager that controls and monitors one or more JMS listeners, which each monitor a JMS destination on behalf of a deployed message-driven bean.

5. Check your JMS resource configurations If the messaging services seem to be running properly, check that the JMS resources have been configured correctly. For example, check that the JMS activation specification against which the message-driven bean is deployed has been configured correctly. For more information about configuring JMS resources for message-driven beans, see Administering support for message-driven beans.

6. Get a detailed exception dump for messaging. If the information obtained in the preceding steps is still inconclusive, you can enable the application server debug trace for the ″Messaging″ group to provide a detailed exception dump.

Related tasks

“Troubleshooting WebSphere messaging” on page 342
Use this overview task to help resolve a problem that you think is related to the WebSphere Messaging.

Mail, URLs, and other J2EE resources

Enabling debugger for a mail session

When you need to debug a JavaMail application, you can use the JavaMail debugging feature. Enabling the debugger triggers the JavaMail component of WebSphere Application Server to print the following data to the `stdout` output stream:

- interactions with the mail servers
- properties of the mail session

This output stream is redirected to the `SystemOut.log` file for the specific application server.

The mail debugger functions on a per session basis. To enable the JavaMail debugging feature:

1. Open the administrative console.
2. Click **Resources**>**Mail Providers**>*mail_session*>**Mail Session**>*mail session*.
3. Click **Debug**. Debug is enabled for just that session.
4. Click **Apply** or **OK**.

The following example shows sample JavaMail debugging output:

```
DEBUG: not loading system providers in <java.home>/lib
DEBUG  not loading optional custom providers file: /META-INF/javamail.providers
DEBUG: successfully loaded default providers

DEBUG  Tables of loaded providers
DEBUG: Providers listed by Class Name:
{com.sun.mail.smtp.SMTPTransport=javax.mail.Provider
     [TRANSPORT,smtp,com.sun.mail.smtp.SMTPTransport,Sun
Microsystems, Inc], com.sun.mail.imap.IMAPStore=javax.mail.Provider
     [STORE,imap,com.sun.mail.imap.IMAPStore,Sun
Microsystems, Inc], com.sun.mail.pop3.POP3Store=javax.mail.Provider
     [STORE,pop3,com.sun.mail.pop3.POP3Store,Sun
Microsystems, Inc]}
DEBUG:  Providers Listed By Protocol:
{imap=javax.mail.Provider[STORE,imap,com.sun.mail.imap.IMAPStore,Sun
Microsystems,Inc], pop3=javax.mail.Provider
[STORE,pop3,com.sun.mail.pop3.POP3Store,Sun
Microsystems, Inc], smtp=javax.mail.Provider
[TRANSPORT,smtp,com.sun.mail.smtp.SMTPTransport,Sun
Microsystems, Inc]}
DEBUG: not loading optional address map file: /META-INF/javamail.address.map
*** In SessionFactory.getObjectInstance,
  The default SessionAuthenticator is based on:
   store_user = john_smith
   store_pw = abcdef
*** In SessionFactory.getObjectInstance, parameters in the new session:
    mail.store.protocol="imap"
    mail.transport.protocol="smtp"
    mail.imap.user="john_smith"
    mail.smtp.host="smtp.coldmail.com"
    mail.debug="true"
    ws.store.password="abcdef"
    mail.from="john_smith@coldmail.com"
    mail.smtp.class="com.sun.mail.smtp.SMTPTransport"
    mail.imap.class="com.sun.mail.imap.IMAPStore"
    mail.imap.host="coldmail.com"
DEBUG: mail.smtp.class property exists and points to com.sun.mail.smtp.SMTPTransport
DEBUG SMTP: useEhlo true, useAuth false
DEBUG: SMTPTransport trying to connect to host "smtp.coldmail.com", port 25

javax.mail.SendFailedException: Sending failed;
      nested exception is:
      javax.mail.MessagingException: Unknown SMTP host: smtp.coldmail.com;
       nested exception is
       java.net.UnknownHostException: smtp.coldmail.com
       at javax.mail.Transport.send0(Transport.java:219)
       at javax.mail.Transport.send(Transport.java:81)
       at ws.mailfvt.SendSaveTestCore.runAll(SendSaveTestCore.java:48)
       at testers.AnyTester.main(AnyTester.java:130)
```

This output illustrates a connection failure to a Simple Mail Transfer Protocol (SMTP) server because a fictitious name, smtp.coldmail.com, is specified as the server name.

The following list provides tips on reading the previous sample of debugger output:

- The lines headed by *DEBUG* are printed by the JavaMail run-time, while the two lines headed by *** are printed by the WebSphere environment run-time.
- The first two lines say that some configuration files are skipped. At run-time the JavaMail component attempts to load a number of configuration files from different locations. All those files are not required. If a required file cannot be accessed, however, the JavaMail component creates an exception. In this sample, there is no exception and the third line announces that default providers are loaded.
- The next few lines, headed by either *Providers listed by Class Name* or *Providers Listed by Protocols*, show the protocol providers that are loaded. The three providers that are listed are the default protocol providers that come under the WebSphere built-in mail provider. They are the protocols SMTP, IMAP, and POP3, respectively. If you install special protocol providers (or, in JavaMail terminology, service providers) and these providers are used in the current mail session, you see them listed here with the default providers.
- The two lines headed by *** and the few lines below them are printed by WebSphere Application Server to show the configuration properties of the current mail session. Although these properties are listed by their internal name rather than the name you establish in the administrative console, you can easily recognize the relationships between them. For example, the property *mail.store.protocol* corresponds to the Protocol Name property in the Store Access section of the mail session configuration page.

  Note: Review the listed properties and values to verify that they correspond.
- The few lines above the exception stack show the JavaMail activities when sending a message. First, the JavaMail API recognizes that the transport protocol is set to SMTP and that the provider `com.sun.mail.smtp.SMTPTransport` exists. Next, the parameters used by SMTP, `useEhlo` and `useAuth`, are shown. Finally, the log shows the SMTP provider trying to connect to the mail server `smtp.coldmail.com`.
- Next is the exception stack. This data indicates that the specified mail server either does not exist or is not functioning.

Security

Troubleshooting authorization providers

This article describes the issues you might encounter using a Java Authorization Contract for Containers (JACC) authorization provider. Tivoli Access Manager is bundled with WebSphere Application Server as an authorization provider. However, you also can plug in your own authorization provider.

Using Tivoli Access Manager as a Java Authorization Contract for Containers authorization provider

You might encounter the following issues when using Tivoli Access Manager as a JACC authorization provider:

- The configuration of JACC might fail.
- The server might fail to start after configuring JACC.
- The application might not deploy properly.
- The startServer command might fail after you have configured Tivoli Access Manager or a clean uninstall did not take place after unconfiguring JACC.

- An ″HPDIA0202w An unknown user name was presented to Access Manager″ error might occur.
- An ″HPDAC0778E The specified user's account is set to invalid″ error might occur.
- An WASX7017E: Exception received while running file ″InsuranceServicesSingle.jacl″ error might occur.

Using an external provider for Java Authorization Contract for Containers authorization

You might encounter the following issues when you use an external provider for JACC authorization:

- An ″HPDJA0506E Invalid argument: Null or zero-length user name field for the ACL entry″ error might occur.

The configuration of JACC might fail

If you are having problems configuring JACC, check the following:

- Ensure that the parameters are correct. For example, there should not be a number after TAM_Policy_server_hostname:7135, but there should be a number after TAM_Authorization_server_hostname:7136 (for example, TAM_Authorization_server_hostname:7136:1).
- If a message such as "server can't be contacted" appears, it is possible that the host names or port numbers of the Tivoli Access Manager servers are incorrect, or that the Tivoli Access Manager servers have not been started.
- Ensure that the password for sec_master is correct.
- Check the SystemOut.log and search for the string `AMAS` to see if any error messages are present.

The server might fail to start after configuring JACC

If the server does not start after JACC has been configured, check the following:

- Ensure that the WebSphere Application Server and Tivoli Access Manager use the same Lightweight Directory Access Protocol (LDAP) server.
- If the message "Policy Director Authentication failed″ appears, ensure that the:
  - WebSphere Application Server LDAP serverID is the same as the "Administrator user" in the Tivoli Access Manager JACC configuration panel.
  - Tivoli Access Manager Administrator distinguished name (DN) is correct.
  - Password of the Tivoli Access Manager administrator has not expired and is valid.
  - Account is valid for the Tivoli Access Manager administrator.
- If a message such as "socket can't be opened for *xxxx*″ (where *xxxx* is a number) appears, do the following:
  1. Go to $WAS_HOME/profiles/profile_name/etc/tam.
  2. Change *xxxx* to an available port number in amwas.commomconfig.properties, and amwas*cellName_dmgr.properties if dmgr failed to start. If Node failed to start, change *xxx* to an available port number in amwas*cellName_nodeName_.properties. If appSever failed to start, change *xxxx* in Amwas*cellname_nodeName_serverName.properties.

The application might not deploy properly

When you click **Save**, the policy and role information is propagated to the Tivoli Access Manager policy. It might take some time to finish. If the save fails, you must uninstall the application and then reinstall it.

To access an application after it is installed, you must wait 30 seconds (by default) to start the application after you save.

The startServer command might fail after you have configured Tivoli Access Manager or a clean uninstall did not take place after unconfiguring JACC.

If the cleanup for JACC unconfiguration or start server fails after JACC has been configured, do the following:

- Remove Tivoli Access Manager properties files from WebSphere Application Server. For each application server in a network deployment (ND) environment with N servers defined (for example, server1, server2), the following files must be removed:

  ```
  $WAS_INSTALL/java/jre/PdPerm.properties
  $WAS_INSTALL/java/jre/PdPerm.ks
  $WAS_INSTALL/profiles/profile_name/etc/tam/*
  ```

- Use a utility to clear the security configuration and return the system to the state it was in before Tivoli Access Manager JACC was configured. The utility removes all of the PDLoginModuleWrapper entries as well as the Tivoli Access Manager authorization table entry from the security.xml file, effectively removing the Tivoli Access Manager JACC provider. Backup security.xml before running this utility.

  Enter the following commands:

  ```
  $WAS_HOME/java/jre/bin/java -classpath
   "$WAS_HOME/lib/AMJACCProvider.jar:CLASSPATH"
  com.tivoli.pd.as.jacc.cfg.CleanSecXML fully_qualified_path/security.xml
  ```

An "HPDIA0202w An unknown user name was presented to Access Manager" error might occur

You might encounter the following error message if you are attempting to use an existing user in a Local Directory Access Protocol (LDAP) user registry with Tivoli Access Manager:

```
AWXJR0008E   Failed to create a PDPrincipal for principal mgr1.:
AWXJR0007E   A Tivoli Access Manager exception was caught. Details are:
"HPDIA0202W   An unknown user name was presented to Access Manager."
```

This problem might be caused by the hostname exceeding predefined limits with Tivoli Access Manager when it is configured against MS Active Directory. In WebSphere Version 6.0, the maximum length of the hostname can not exceed 46 characters.

Check that the hostname is not fully qualified. Configure the machine so that the hostname does not include the host domain.

To correct this error, complete the following steps:

1. On the command line, type the following information to get a Tivoli Access Manager command prompt:

   ```
   pdadmin -a administrator_name -p administrator_password
   ```

   The pdadmin *administrator_name* prompt is displayed. For example:

   ```
   pdadmin -a administrator1 -p password
   ```

2. At the pdadmin command prompt, import the user from the LDAP user registry to Tivoli Access Manager by typing the following information:

   ```
   user import user_name cn=user_name,o=organization_name,c=country
   ```

   For example:

   ```
   user import jstar cn=jstar,o=ibm,c=us
   ```

After importing the user to Tivoli Access Manager, you must use the `user modify` command to set the user account to valid. The following syntax shows how to use this command:

```
user modify user_name account-valid yes
```

For example:

```
user modify jstar account-valid yes
```

For information on how to import a group from LDAP to Tivoli Access Manager, see the Tivoli Access Manager documentation.

An "HPDAC0778E The specified user's account is set to invalid" error might occur

You might encounter the following error message after you import a user to Tivoli Access Manager and restart the client:

```
AWXJR0008E   Failed to create a PDPrincipal for principal mgr1.:
AWXJR0007E   A Tivoli Access Manager exception was caught.
Details are: "HPDAC0778E   The specified user's account is set to invalid."
```

To correct this error, use the `user modify` command to set the user account to valid. The following syntax shows how to use this command:

```
user modify user_name account-valid yes
```

For example:

```
user modify jstar account-valid yes
```

An "HPDJA0506E Invalid argument: Null or zero-length user name field for the ACL entry" error might occur

You might encounter an error similar to the following message when you propagate the security policy information from the application to the provider using the wsadmin command propagatePolicyToJACCProvider:

```
AWXJR0035E   An error occurred while  attempting to add member, cn=agent3,
o=ibm,c=us, to role AgentRole
HPDJA0506E   Invalid argument: Null or zero-length user name field for the ACL entry
```

To correct this error, create or import the user, which is mapped to the security role to the Tivoli Access Manager. For more information on propagating the security policy information, see the documentation for your authorization provider.

An WASX7017E: Exception received while running file "InsuranceServicesSingle.jacl" error might occur

After the JACC provider and Tivoli Access Manager are enabled, when attempting to install the application (which is configured with security roles using the wsadmin command), the following error might occur:

```
WASX7017E: Exception received while running file "InsuranceServicesSingle.jacl"; exception
information: com.ibm.ws.scripting.ScriptingException: WASX7111E:
Cannot find a match for supplied option:
"[RuleManager, , , cn=mgr3,o=ibm,c=us|cn=agent3,o=ibm,c=us, cn=ManagerGro
up,o=ibm,c=us|cn=AgentGroup,o=ibm,c=us]" for task "MapRolesToUsers
```

The $AdminApp task option MapRolesToUsers becomes invalid when Tivoli Access Manager is used as the authorization server. To correct the error, change MapRolesToUsers to TAMMapRolesToUsers.

Related concepts

Authorization in WebSphere Application Server

Tivoli Access Manager integration as the JACC provider

JACC providers

JACC support in WebSphere Application Server

Related tasks

Enabling an external JACC provider

Configuring a JACC provider

Propagating security policy of installed applications to a JACC provider using wsadmin scripting

Related reference

Interfaces used to support JACC

Related information

IBM Tivoli Access Manager for e-business 5.1

Troubleshooting security configurations

Refer to Security components troubleshooting tips for instructions on how to troubleshoot errors related to security.

The following topics explain how to troubleshoot specific problems related to configuring and enabling security configurations:

- Errors when configuring or enabling security
- Errors or access problems after enabling security
- Errors after enabling Secure Sockets Layer (SSL) or SSL-related error messages

Errors when trying to configure or enable security

What kind of error are you seeing?

- “"LTPA password not set. validation failed" message displayed as error in the Administrative Console after saving global security settings ” on page 354
- “Errors when trying to configure or enable security”
- “The setupClient.bat or setupClient.sh file is not working correctly” on page 354

- "Java HotSpot Server VM warning: Unexpected Signal 11 occurred under user-defined signal handler 0x7895710a message occurs in the native_stdout.log file when enabling security on the HP-UX11i platform"
- "WebSphere Application Server Version 6 is not working correctly with Enterprise Workload Manager (EWLM)" on page 355
- If you have successfully configured security (made changes, saved the configuration, and enabled security with no errors), but are now having problems accessing Web resources or the administrative console, refer to Errors or access problems after enabling security.

"LTPA password not set. validation failed" message displayed as error in the Administrative Console after saving global security settings

This error can be caused if, when configuring WebSphere Application Server security, "LTPA" is selected as the authentication mechanism, and the LTPA password field is not set. To resolve this problem:
- Select Security **Authentication Mechanism > LTPA** in the console left-hand navigation pane.
- Complete the password and confirm password fields.
- Click **OK**.
- Try setting Global Security again.

The setupClient.bat or setupClient.sh file is not working correctly

The `setupClient.bat` file on Windows platforms and the `setupClient.sh` file on UNIX platforms incorrectly specify the location of the SOAP security properties file.

In the `setupClient.bat` file, the correct location should be:

```
set CLIENTSOAP=-Dcom.ibm.SOAP.ConfigURL=file:%WAS_HOME%/properties/soap.client.props
```

In the `setupClient.sh` file, the CLIENTSOAP variable should be:

```
CLIENTSOAP=-Dcom.ibm.SOAP.ConfigURL=file:$WAS_HOME/properties/soap.client.props
```

In the `setupClient.bat` and `setupClient.sh` files, complete the following steps:
1. Remove the leading / after file:.
2. Change `sas` to `soap`.

Java HotSpot Server VM warning: Unexpected Signal 11 occurred under user-defined signal handler 0x7895710a message occurs in the `native_stdout.log` file when enabling security on the HP-UX11i platform

After you enable security on HP-UX 11i platforms, the following error in the `native_stdout.log` file occurs, along with a core dump and WebSphere Application Server does not start:

```
Java HotSpot(TM) Server VM warning:
Unexpected Signal 11 occurred under user-defined signal handler 0x7895710a
```

To work around this error, apply the fixes recommended by HP for Java at the following URL:
http://www.hp.com/products1/unix/java/infolibrary/patches.html.

WebSphere Application Server Version 6 is not working correctly with Enterprise Workload Manager (EWLM)

To use WebSphere Application Server Version 6 with Enterprise Workload Manager (EWLM), you must manually update the WebSphere Application Serve server.policy files. For example:

```
grant codeBase "file:/<EWLM_Install_Home>/classes/ARM/arm4.jar" {
 permission java.security.AllPermission;
};
```

Otherwise, you might encounter a Java 2 security exception for violating the Java 2 security permission.

Refer to Configuring server.policy files for more information on configuring server.policy files.

For current information available from IBM Support on known problems and their resolution, see the IBM Support page.

IBM Support has documents that can save you time gathering information needed to resolve this problem. Before opening a PMR, see the IBM Support page.

Related tasks

Chapter 6, "Troubleshooting by task," on page 233

Access problems after enabling security

What kind of error are you seeing?

- I cannot access all or part of the administrative console or use the wsadmin tool after enabling security
- I cannot access a Web page after enabling security
- Authentication error accessing a Web page
- Authorization error accessing a Web page
- The client cannot access an enterprise bean after enabling security
- The client never gets prompted when accessing a secured enterprise bean
- I cannot stop an application server, node manager, or node after enabling security
- After enabling single signon, I cannot log on to the administrative console

I cannot access all or part of the administrative console or use the wsadmin tool after enabling security

- If you cannot access the administrative console, or view and update certain objects, look in the logs of the application server which hosts the administrative console page for a related error message.

  Note: You will need to use the administrative console to complete the next two items. If you are having a problem accessing the administrative console, you will have to turn off security and restart the administrative console. To turn off security, enter the following command at the system command prompt:

  ```
  wsadmin.sh -conntype NONE
  ```

  When the system command prompt reappears, enter:

  ```
  securityoff
  ```

  Restart the Deployment Manager after you turn off security.

- You might not have authorized your ID for administrative tasks. This problem is indicated by errors such as:
  - `[8/2/02 10:36:49:722 CDT] 4365c0d9 RoleBasedAuth A CWSCJ0305A: Role based authorization check failed for security name MyServer/myUserId, accessId MyServer/S-1-5-21-882015564-4266526380-2569651501-1005 while invoking method getProcessType on resource Server and module Server.`
  - Exception message: `"CWWMN0022E: Access denied for the getProcessType operation on Server MBean"`
  - `When running the command: wsadmin -username j2ee -password j2ee: CWWAX7246E: Cannot establish "SOAP" connection to host "BIRKT20" because of an authentication failure. Please ensure that user and password are correct on the command line or in a properties file.`

  To grant an ID administrative authority, from the administrative console, click **System Administration > Console Users** and validate that the ID is a member. If it is not, add the ID with at least monitor access privileges, for read-only access.
- Verify that the *enable_trusted_application* flag is set to true. To check the *enable_trusted_application* flag value using the administrative console, click **Security > Global Security**. Under Additional properties, click **Custom properties > EnableTrustedApplications**.

Remember: You could encounter synchronization problems if you are in a Network Deployment environment and save your security settings and the node agent was down.

I cannot access a Web page after enabling security

When secured resources are not accessible, probable causes include:
- Authentication errors - WebSphere Application Server security cannot identify the ID of the person or process. Symptoms of authentication errors include:

  On a Netscape browser:
  - `Authorization failed. Retry?` message displays after an attempt to log in.
  - Accepts any number of attempts to retry login and displays `Error 401` message when Cancel is clicked to stop retry.
  - A typical browser message displays: `Error 401: Basic realm='Default Realm'`.

  On an Internet Explorer browser:
  - Login prompt displays again after an attempt to log in.
  - Allows three attempts to retry login.
  - Displays `Error 401` message after three unsuccessful retries.
- Authorization errors - The security function has identified the requesting person or process as not authorized to access the secured resource. Symptoms of authorization errors include:
  - Netscape browser: "Error 403: AuthorizationFailed" message is displayed.
  - Internet Explorer:
    - "You are not authorized to view this page" message is displayed.
    - "HTTP 403 Forbidden" error is also displayed.
- SSL errors - WebSphere Application Server security uses Secure Socket Layer (SSL) technology internally to secure and encrypt its own communication, and incorrect configuration of the internal SSL settings can cause problems. Also you might have enabled SSL encryption for your own Web application or enterprise bean client traffic which, if configured incorrectly, can cause problems regardless of whether WebSphere Application Server security is enabled.

– SSL related problems are often indicated by error messages which contain a statement such as: `ERROR: Could not get the initial context or unable to look up the starting context.Exiting.` followed by `javax.net.ssl.SSLHandshakeException`

The client cannot access an enterprise bean after enabling security

If client access to an enterprise bean fails after security is enabled:

- Review the steps for securing and granting access to resources.
- Browse the server logs for errors relating to enterprise bean access and security. Look up any errors in the message table.

  Errors similar to `Authorization failed for /UNAUTHENTICATED while invoking` *`resource`* `securityName:/UNAUTHENTICATED;accessId:UNAUTHENTICATED not granted any of the required roles` *`roles`* indicate that:
  – An unprotected servlet or JavaSever Page (JSP) file accessed a protected enterprise bean. When an unprotected servlet is accessed, the user is not prompted to log in and the servlet runs as UNAUTHENTICATED. When the servlet makes a call to an enterprise bean that is protected the servlet fails.

    To resolve this problem, secure the servlet that is accessing the protected enterprise bean. Make sure the *runAs* property for the servlet is set to an ID that can access the enterprise bean.
  – An unauthenticated Java client program is accessing an enterprise bean resource that is protected. This situation can happen if the file read by the `sas.client.props` properties file used by the client program does not have the `securityEnabled` flag set to **true**.

    To resolve this problem, make sure that the `sas.client.props` file on the client side has its securityEnabled flag set to **true**.

  Errors similar to **Authorization failed for *valid_user* while invoking *resource* securityName:/username;accessId:xxxxxx not granted any of the required roles *roles*** indicate that a client attempted to access a secured enterprise bean resource, and the supplied user ID is not assigned the required roles for that enterprise bean.
  – Check that the required roles for the enterprise bean resource are accessed. View the required roles for the enterprise bean resource in the deployment descriptor of the Web resource.
  – Check the authorization table and make sure that the user or the group that the user belongs to is assigned one of the required roles. You can view the authorization table for the application that contains the enterprise bean resource using the administrative console.
- If you are using local OS and SAF authorization, check the SAF EJBROLEs setup. Specifically, verify that
  – the EJBROLE class has been activated.
  – The role has been defined to SAF.
  – The userid has been permitted to the role.
  – The class was refreshed after the permit operation.

Client program never gets prompted when accessing secured enterprise bean

Even though it appears security is enabled and an enterprise bean is secured, it can happen that the client executes the remote method without prompting. If the remote method is protected, an authorization failure results. Otherwise, execute the method as an unauthenticated user.

Possible reasons for this problem include:

- The server with which you are communicating might not have security enabled. Check with the WebSphere Application Server administrator to ensure that the server security is enabled. Access the global security settings from within the Security section of the administrative console.
- The client does not have security enabled in the sas.client.props file. Edit the sas.client.props file to ensure the property **com.ibm.CORBA.securityEnabled** is set to true.
- The client does not have a ConfigURL specified. Verify that the property **com.ibm.CORBA.ConfigURL** is specified on the command line of the Java client, using the -D parameter.
- The specified ConfigURL has an invalid URL syntax, or the `sas.client.props` that is pointed to cannot be found. Verify that the **com.ibm.CORBA.ConfigURL** property is valid, for example, similar to the C:/WebSphere/AppServer/properties/sas.client.props file on Windows systems. Check the Java documentation for a description of URL formatting rules. Also, validate that the file exists at the specified path.

Cannot stop an application server, node manager, or node after enabling security

If you use command line utilities to stop WebSphere Application Server processes, apply additional parameters after enabling security to provide authentication and authorization information.

After enabling single signon, I cannot log on to the administrative console

This problem occurs when single signon (SSO) is enabled, and you attempt to access the administrative console using the short name of the server, for example `http://myserver:9060/ibm/console`. The server accepts your user ID and password, but returns you to the log on page instead of the administrative console.

To correct this problem, use the fully qualified host name of the server, for example `http://myserver.mynetwork.mycompany.com:9060/ibm/console`.

Related tasks

Chapter 6, “Troubleshooting by task,” on page 233

Errors after enabling security

What kind of error are you seeing?

- Authentication error accessing a Web page
- Authorization error accessing a Web page
- Error Message: `CWSCJ0314E: Current Java 2 security policy reported a potential violation`
- `CWMSG0508E: The JMS Server security service was unable to authenticate user ID:` error displayed in SystemOut.log when starting an application server
- Error Message: `CWSCJ0237E: One or more vital LTPAServerObject configuration attributes are null or not available` after enabling security and starting the application server
- Error Message: `CWSCJ0336E: Authentication failed for user {0} because of the following exception {1}`

Authentication error accessing a Web page

Possible causes for authentication errors include:

- **Incorrect user name or passwords.** Check the user name and password and make sure they are correct.

- **Security configuration error : User registry type is not set correctly.** Check the user registry property in global security settings in the administrative console. Verify that it is the intended user registry.
- **Internal program error.** If the client application is a Java standalone program, this program might not gather or send credential information correctly.

Authorization error accessing a Web page

If a user who should have access to a resource does not, there is probably a missing configuration step. Review the steps for securing and granting access to resources.

Specifically:
- Check required roles for the accessed Web resource.
- Check the authorization table to make sure that the user, or the groups to which the user belongs, is assigned to one of the required roles.
- View required roles for the Web resource in the deployment descriptor of the Web resource.
- View the authorization table for the application that contains the Web resource, using the administrative console.
- Test with a user who is granted the required roles, to see if the user can access the problem resources.
- If the problem user is required to have one or more of the required roles, use the administrative console to assign that user to required roles. Then stop and restart the application.

Error Message: CWSCJ0314E: Current Java 2 security policy reported a potential violation on server

If you find errors on your server similar to:

```
Error Message: CWSCJ0314E: Current Java 2 Security policy reported a potential violation of
Java 2 Security Permission. Please refer to Problem Determination Guide for further information.
{0}Permission\:{1}Code\:{2}{3}Stack Trace\:{4}Code Base Location\:{5}
```

The Java security manager `checkPermission()` method has reported an exception, `SecurityException`.

The reported exception might be critical to the secure system. Turn on security trace to determine the potential code that might have violated the security policy. Once the violating code is determined, verify if the attempted operation is permitted with respect to Java 2 Security, by examining all applicable Java 2 security policy files and the application code.

A more detailed report is enabled by either configuring RAS trace into debug mode, or specifying a Java property.
- Specify the following property in the **Application Servers >** *server_name* **> ProcessDefinition > Java Virtual Machine** panel from the administrative console in the **Generic JVM arguments** panel:
  - Add the run-time flag **java.security.debug**
  - Valid values:
    access Print all debug information including: required permission, code, stack, and code base location.
    stack Print debug information including: required permission, code, and stack.
    failure Print debug information including: required permission and code.

For a review of Java security policies and what they mean , see the Java 2 Security documentation at http://java.sun.com/j2se/1.3/docs/guide/security/index.html .

Tip: If the application is running with a Java Mail API, this message might be benign. You can update the *installed Enterprise Application root*/META-INF/was.policy file to grant the following permissions to the application:

- permission java.io.FilePermission "${user.home}${/}.mailcap", "read";
- permission java.io.FilePermission "${user.home}${/}.mime.types", "read";
- permission java.io.FilePermission "${java.home}${/}lib${/}mailcap", "read";
- permission java.io.FilePermission "${java.home}${/}lib${/}mime.types", "read";

Error message: CWMSG0508E: The JMS Server security service was unable to authenticate user ID:" error displayed in SystemOut.log when starting an application server

This error can result from installing the JMS messaging API sample and then enabling security. You can follow the instructions in the Configure and Run page of the corresponding JMS sample documentation to configure the sample to work with WebSphere Application Server security.

You can verify the installation of the message-driven bean sample by launching the installation program, selecting **Custom**, and browsing the components which are already installed in the **Select the features you like to install** panel. The JMS sample is shown as **Message-Driven Bean Sample**, under **Embedded Messaging**.

You can also verify this installation by using the administrative console to open the properties of the application server which contains the samples. Select **MDBSamples** and click uninstall.

Error message: CWSCJ0237E: One or more vital LTPAServerObject configuration attributes are null or not available after enabling security and starting the application server.

This error message can result from selecting Lightweight Third Party Authentication (LTPA) as the authentication mechanism, but not generating the LTPA keys. The LTPA keys encrypt the LTPA token.

To resolve this problem:

1. Click **System Administration > Console users > LTPA**
2. Enter a password, which can be anything.
3. Enter the same password in **Confirm Password**.
4. Click **Apply**.
5. Click **Generate Keys**.
6. Click on **Save**.

The exception AccessControlException, is reported in the SystemOut.log

The problem is related to the Java 2 Security feature of WebSphere Application Server, the API-level security framework that is implemented in WebSphere Application Server Version 5. An exception similar to the following example displays. The error message and number can vary.

```
E CWSRV0020E: [Servlet Error]-[validator]: Failed to load servlet:
java.security.AccessControlException: access denied
(java.io.FilePermission
\WebSphere\AppServer\installedApps\maeda\adminconsole.ear\adminconsole.war\
WEB-INF\validation.xml read)
```

Possible causes of these errors include:

- Syntax errors in a policy file.
- Syntax errors in permission specifications in the ra.xml file bundled in a .rar file. This case applies to resource adapters that support connector access to CICS or other resources.
- An application is missing the specified permission in a policy file, or in permission specifications in an ra.xml file bundled in a .rar file
- The class path is not set correctly, preventing the permissions for the resource.xml file for SPI from being correctly created.
- A library called by an application, or the application, is missing a doPrivileged block to support access to a resource.
- Permission is specified in the wrong policy file.

To resolve these problems:

- Check all of the related policy files to verify that the permission shown in the exception, for example java.io.FilePermission, is specified.
- Look for a related ParserException in the SystemOut.log file which reports the details of the syntax error. For example:

  ```
  CWSCJ0189E: Caught ParserException while creating template for application policy
  \WAS\config\cells\xxx\nodes\xxx\app.policy.
  ```

  The exception is com.ibm.ws.security.util.ParserException: line 18: expected ';', found 'grant'
- Look for a message similar to: `CWSCJ0325W: The permission` *`permission`* `specified in the policy file is unresolved.`
- Check the call stack to determine which method does not have the permission. Identify the class path of this method. If it is hard to identify the method, enable the Java2 security Report.
  - The report shows:

    Permission
    : the missing permission.

    Code
    : which method has the problem.

    Stack Trace
    : where the access violation occurred.

    CodeBaseLocation
    : the detail of each stack frame.

    Usually, Permission and Code are enough to identify the problem. The following example illustrates a report:

    ```
    Permission:
    \WebSphere\AppServer\logs\server1\SystemOut_02.08.20_11.19.53.log :

    ccess denied (java.io.FilePermission

    WebSphere\AppServer\logs\server1\SystemOut_02.08.20_11.19.53.log delete)

    Code:
     com.ibm.ejs.ras.RasTestHelper$7  in
    {file:/WebSphere/AppServer/installedApps/maeda/JrasFVTApp.ear/RasLib.jar}

    Stack Trace:

    java.security.AccessControlException: access denied (java.io.FilePermission
    ```

```
\WebSphere\AppServer\logs\server1\SystemOut_02.08.20_11.19.53.log delete)
        at java.security.AccessControlContext.checkPermission
                                (AccessControlContext.java(Compiled Code))
        at java.security.AccessController.checkPermission
                                (AccessController.java(Compiled Code))
        at java.lang.SecurityManager.checkPermission
                                (SecurityManager.java(Compiled Code))
                                 .
Code Base Location:

com.ibm.ws.security.core.SecurityManager :

file:/WebSphere/AppServer/lib/securityimpl.jar
  ClassLoader: com.ibm.ws.bootstrap.ExtClassLoader
  Permissions granted to CodeSource
(file:/WebSphere/AppServer/lib/securityimpl.jar <no certificates>
  {
    (java.util.PropertyPermission java.vendor read);
    (java.util.PropertyPermission java.specification.version read);
    (java.util.PropertyPermission line.separator read);
    (java.util.PropertyPermission java.class.version read);
    (java.util.PropertyPermission java.specification.name read);
    (java.util.PropertyPermission java.vendor.url read);
    (java.util.PropertyPermission java.vm.version read);
    (java.util.PropertyPermission os.name read);
    (java.util.PropertyPermission os.arch read);
   }
    ( This list continues.)
```

- If the method is SPI, check the resources.xml file to ensure that the class path is correct.
- To confirm that all of the policy files are loaded correctly, or what permission each class path is granted, enable the trace with **com.ibm.ws.security.policy.*=all=enabled**. All loaded permissions are listed in the trace.log file. Search for the app.policy, was.policy and ra.xml files. To check the permission list for a class path, search for **Effective Policy for** *classpath*.
- If there are any syntax errors in the policy file or ra.xml file, correct them with the policytool. Avoid editing the policy manually, because syntax errors can result.
- If a permission is listed as Unresolved, it does not take effect. Verify that the specified permission name is correct.
- If the class path specified in the resource.xml file is not correct, correct it.
- If a required permission does not exist in either the policy files or the ra.xml file, examine the application code to see if you need to add this permission. If so, add it to the proper policy file or ra.xml file.
- If the permission should not be granted outside of the specific method that is accessing this resource, modify the code needs to use a doPrivileged block.

Tip: If the application is running with the Java Mail API, you can update the *installed Enterprise Application root*/META-INF/was.policy file to grant the following permissions to the application:

- `permission java.io.FilePermission "${user.home}${/}.mailcap", "read";`
- `permission java.io.FilePermission "${user.home}${/}.mime.types", "read";`
- `permission java.io.FilePermission "${java.home}${/}lib${/}mailcap", "read";`
- `permission java.io.FilePermission "${java.home}${/}lib${/}mime.types", "read";`

Error Message: CWSCJ0336E: Authentication failed for user {0} because of the following exception {1}

This error message results if the user ID indicated is not found in the LDAP user registry. To resolve this problem:
1. Verify that your user ID and password are correct.
2. Verify that the user ID exists in the registry.
3. Verify that the base distinguished name (DN) is correct.
4. Verify that the user filter is correct.
5. Verify that the bind DN and the password for the bind DN are correct. If the bind DN and password are not specified, add the missing information and retry.
6. Verify that the host name and LDAP type are correct.

Consult with the administrator of the user registry if the problem persists.

Related tasks

Chapter 6, "Troubleshooting by task," on page 233

Related reference

"Access problems after enabling security" on page 355

Errors after configuring or enabling Secure Sockets Layer

This article explains various problems you might encounter after configuring or enabling Secure Sockets Layer (SSL).

Stopping the deployment manager after configuring Secure Sockets Layer

After configuring the Secure Sockets Layer repertoires, if you stop the deployment manager without also stopping the node agents, you might receive the following error message when you restart the deployment manager:

```
CWWMU0509I: The server "nodeagent" cannot be reached. It appears to be stopped.
CWWMU0211I: Error details may be seen in the file:
           /opt/WebSphere/AppServer/logs/nodeagent/stopServer.log
```

The error occurs because the deployment manager did not propagate the new SSL certificate to the node agents. Thus, the node agents are using an older certificate files than the deployment manager and the certificate files are incompatible. To work around this problem, you must manually stop the node agent and deployment manager processes. To end the processes on Windows platforms, use the Task Manager. On UNIX platforms, run the command to end the process.

There are some things you need to consider when identifying the specific process that should be killed. For each process being killed, WebSphere Application Server stores the process ID in a `pid` file and you need to find these `*.pid` files. For example, the `server1.pid` for a standalone install might be found at: `<WAS_root>/AppServer/logs/server1.pid`

Accessing resources using HTTPS

If you are unable to access resources using a Secure Sockets Layer (SSL) URL (beginning with https:), or encounter error messages which indicate SSL problems, verify that your HTTP server is configured correctly for SSL by browsing the welcome page of the HTTP server using SSL by entering the URL: **https://***hostname*.

If the page works with HTTP, but not HTTPS, the problem is with the HTTP server.

- Refer to the documentation for your HTTP server for instructions on correctly enabling SSL. If you are using the IBM HTTP Server or Apache, go to:http://www.ibm.com/software/webservers/httpservers/library.html. Click **Frequently Asked Questions**> **SSL**.
- If you are use the IBM Key Management (IKeyman) tool to create certificates and keys, remember to stash the password to a file when creating the KDB file with the IBM Key Management Tool.
  1. Go to the directory where the KDB file was created, and see if there is a .sth file.
  2. If not, open the KDB file with the IBM Key Management Tool, and click **Key Database File > Stash Password**. The following message displays: `The password has been encrypted and saved in the file`.

If the HTTP server handles SSL-encrypted requests successfully, or is not involved (for example, traffic flows from a Java client application directly to an enterprise bean hosted by the WebSphere Application Server, or the problem appears only after enabling WebSphere Application Server security), what kind of error are you seeing?

System SSL

See *z/OS System Secure Sockets Layer Programming SC24-5901* for information on using the System Secure Sockets Layer (SSL) callable services programming interfaces.

- `javax.net.ssl.SSLHandshakeException - The client and server could not negotiate the desired level of security. Reason: handshake failure`
- `javax.net.ssl.SSLHandshakeException - The client and server could not negotiate the desired level of security. Reason: unknown certificate`
- `javax.net.ssl.SSLHandshakeException - The client and server could not negotiate the desired level of security. Reason: bad certificate`

javax.net.ssl.SSLHandshakeException - The client and server could not negotiate the desired level of security. Reason: handshake failure

If you see a Java exception stack similar to the following example:

```
[Root exception is org.omg.CORBA.TRANSIENT:  CAUGHT_EXCEPTION_WHILE_CONFIGURING_
SSL_CLIENT_SOCKET: CWWJE0080E:   javax.net.ssl.SSLHandshakeException - The client
and server could not negotiate the desired level of   security. Reason: handshake
failure:host=MYSERVER,port=1079 minor code: 4942F303 completed: No]   at
com.ibm.CORBA.transport.TransportConnectionBase.connect (TransportConnectionBase.java:NNN)
```

Some possible causes are:

- Not having common ciphers between the client and server.
- Not specifying the correct protocol.

To correct these problems:

1. Review the SSL settings. Click **WebSphere Administrative Console Security Settings > SSL Configuration Repertoires > DefaultSSLSettings** (or other named SSL settings).
2. Select the **Secure Sockets Layer (SSL)** option from the Additional Properties menu. You can also browse the file manually by viewing the *install_dir*/properties/sas.client.props file.
3. Check the property specified by the com.ibm.ssl.protocol file to determine which protocol is specified.

4. Check the cipher types specified by the com.ibm.ssl.enabledCipherSuites. You might want to add more cipher types to the list. To see which cipher suites are currently enabled: Fo to the properties page of the SSL settings as described above, and look for the **Cipher Suites** property. To see the list of all possible cipher suites, go to the properties page of the SSL settings as described above, then view the online help for that page. From the help page, click **Configure additional SSL settings**.
5. Correct the protocol or cipher problem by using a different client or server protocol and cipher selection. Typical protocols are SSL or SSLv3.

javax.net.ssl.SSLHandshakeException: unknown certificate

If you see a Java exception stack similar to the following example, it might be caused by not having the personal certificate for the server in the client truststore file:

```
ERROR: Could not get the initial context or unable to look up the starting context.
Exiting.  Exception received: javax.naming.ServiceUnavailableException: A communication
failure occurred while attempting to obtain an initial context using the provider
url: "corbaloc:iiop:localhost:2809". Make sure that the host and port information
is correct and that the server identified by the provider url is a running name
server. If no port number is specified, the default port number 2809 is used.
Other possible causes include the network environment or workstation network
configuration. [Root exception is org.omg.CORBA.TRANSIENT:
CAUGHT_EXCEPTION_WHILE_CONFIGURING_SSL_CLIENT_SOCKET: CWWJE0080E:
javax.net.ssl.SSLHandshakeException - The client and server could not
negotiate the desired level of security. Reason: unknown
certificate:host=MYSERVER,port=1940 minor code: 4942F303 completed: No]
```

To correct this problem:
1. Check the client truststore file to determine if the signer certificate from the server personal certificate is there. For a self-signed server personal certificate, the signer certificate is the public key of the personal certificate. For a certificate authority signed server personal certificate, the signer certificate is the root CA certificate of the CA that signed the personal certificate.
2. Add the server signer certificate to the client truststore file.

javax.net.ssl.SSLHandshakeException: bad certificate

If you see a Java exception stack similar to the following example, it can be caused by having a personal certificate in the client keystore used for SSL mutual authentication but not having extracted the signer certificate into the server truststore file so that the server can trust it whenever the SSL handshake is made:

```
ERROR: Could not get the initial context or unable to look up the starting context.
Exiting.  Exception received: javax.naming.ServiceUnavailableException: A communication
failure occurred while attempting to obtain an initial context using the provider
url: "corbaloc:iiop:localhost:2809". Make sure that the host and port information
is correct and that the server identified by the provider url is a running name
server. If no port number is specified, the default port number 2809 is used. Other
possible causes include the network environment or workstation network configuration.
[Root exception is org.omg.CORBA.TRANSIENT: CAUGHT_EXCEPTION_WHILE_CONFIGURING_SSL_
CLIENT_SOCKET: CWWJE0080E: javax.net.ssl.SSLHandshakeException - The client and
server could not negotiate the desired level of security. Reason:
bad certificate: host=MYSERVER,port=1940 minor code: 4942F303 completed: No]
```

To verify this problem, check the server truststore file to determine if the signer certificate from the client personal certificate is there. For a self-signed client personal certificate, the signer certificate is the public key of the personal certificate. For a certificate authority signed client personal certificate, the signer certificate is the root CA certificate of the CA that signed the personal certificate.

To correct this problem, add the client signer certificate to the server truststore file.

Related tasks

Chapter 6, "Troubleshooting by task," on page 233

Related reference

Troubleshooting testing and first time run problems

Security components troubleshooting tips

This document explains basic resources and steps for diagnosing security related issues in the WebSphere Application Server, including:

- What "SDSF output logs" to look at and what to look for in them.
- "General approach for troubleshooting security-related issues" on page 367 to isolating and resolving security problems.
- When and how to "Trace security" on page 374.

The following security-related problems are addressed elsewhere in the information center:

- Errors and access problems after enabling security

  After enabling global security, there was a degradation in performance. See Enabling global security for information about using the unrestricted policy files.
- Errors after enabling SSL, or SSL-related error messages

If none of these steps solves the problem, check to see if the problem has been identified and documented using the links in Diagnosing and fixing problems: Resources for learning.

Note: For an overview of WebSphere Application Server security components such as z/OS Secure Authentication Services (z/SAS) and how they work, see Getting started with security.

SDSF output logs

When troubleshooting the security component, browse the SDSF logs for the server that hosts the resource you are trying to access. The following is a sample of messages you would expect to see from a server in which the security service has started successfully:

Note: Messages beginning with BBOM0001I are related to z/OS-specific implementations of zSAS and CSIv2. They appear in both the controller and servant but are only applicable in the controller.

```
BBOM0001I com_ibm_Server_Security_Enabled: 1.
BBOM0001I com_ibm_CSI_claimTLClientAuthenticationSupported: 1.
BBOM0001I com_ibm_CSI_claimTLClientAuthenticationRequired: 0.
BBOM0001I com_ibm_CSI_claimTransportAssocSSLTLSSupported: 1.
BBOM0001I com_ibm_CSI_claimTransportAssocSSLTLSRequired: 0.
BBOM0001I com_ibm_CSI_claimMessageConfidentialityRequired: 0.
BBOM0001I com_ibm_CSI_claimClientAuthenticationSupported: 1.
BBOM0001I com_ibm_CSI_claimClientAuthenticationRequired: 0.
BBOM0001I com_ibm_CSI_claimClientAuthenticationtype:
SAFUSERIDPASSWORD.
BBOM0001I com_ibm_CSI_claimIdentityAssertionTypeSAF: 0.
BBOM0001I com_ibm_CSI_claimIdentityAssertionTypeDN: 0.
BBOM0001I com_ibm_CSI_claimIdentityAssertionTypeCert: 0.
BBOM0001I com_ibm_CSI_claimMessageIntegritySupported: NOT SET,DEFAULT=1.
BBOM0001I com_ibm_CSI_claimMessageIntegrityRequired: NOT SET,DEFAULT=1.
BBOM0001I com_ibm_CSI_claimStateful: 1.
BBOM0001I com_ibm_CSI_claimSecurityLevel: HIGH.
BBOM0001I com_ibm_CSI_claimSecurityCipherSuiteList: NOT SET.
```

```
BBOM0001I com_ibm_CSI_claimKeyringName: WASKeyring.
BBOM0001I com_ibm_CSI_claim_ssl_sys_v2_timeout: NOT SET, DEFAULT=100.
BBOM0001I com_ibm_CSI_claim_ssl_sys_v3_timeout: 600.
BBOM0001I com_ibm_CSI_performTransportAssocSSLTLSSupported: 1.
BBOM0001I security_sslClientCerts_allowed: 0.
BBOM0001I security_kerberos_allowed: 0.
BBOM0001I security_userid_password_allowed: 0.
BBOM0001I security_userid_passticket_allowed: 1.
BBOM0001I security_assertedID_IBM_accepted: 0.
BBOM0001I security_assertedID_IBM_sent: 0.
BBOM0001I nonauthenticated_clients_allowed: 1.
BBOM0001I security_remote_identity: WSGUEST.
BBOM0001I security_local_identity: WSGUEST.
BBOM0001I security_EnableRunAsIdentity: 0.

 Messages beginning with BBOO0222I are common to Java WebSphere security.
They appear in both the controller and servant but are applicable to the servant.

 BBOO0222I CWSCJ0240I: Security service initialization completed successfully
 BBOO0222I CWSCJ0215I: Successfully set JAAS login provider
 configuration class to com.ibm.ws.security.auth.login.Configuration.
 BBOO0222I CWSCJ0136I: Custom
 Registry:com.ibm.ws.security.registry.zOS.SAFRegistryImpl has been  initialized
 BBOO0222I CWSCJ0157I: Loaded Vendor AuthorizationTable:
 com.ibm.ws.security.core.SAFAuthorizationTableImpl
 BBOO0222I CWSCJ0243I: Security service started successfully
 BBOO0222I CWSCJ0210I: Security enabled true
```

General approach for troubleshooting security-related issues

When troubleshooting security-related problems, the following questions are very helpful and should be considered:

Does the problem occur when security is disabled?

This is a good litmus test to determine that a problem is security related. However, just because a problem only occurs when security is enabled does not always make it a security problem. More troubleshooting is necessary to ensure the problem is really security-related.

Did security appear to initialize properly?

A lot of security code is visited during initialization. So you will likely see problems there first if the problem is configuration related.

The following sequence of messages generated in the SDSF active log indicate normal code initialization of an application server. Non-security messages have been removed from the sequence that follows. This sequence will vary based on the configuration, but the messages are similar:

```
Trace: 2003/08/25 13:06:31.034 01 t=9EA930 c=UNK key=P8 (13007002)
  FunctionName: com.ibm.ws.security.auth.login.Configuration
  SourceId: com.ibm.ws.security.auth.login.Configuration
  Category: AUDIT
  ExtendedMessage: CWSCJ0215I: Successfully set JAAS login provider
  configuration class to com.ibm.ws.security.auth.login.Configuration.
Trace: 2003/08/25 13:06:31.085 01 t=9EA930 c=UNK key=P8 (13007002)
  FunctionName: com.ibm.ws.security.core.SecurityDM
  SourceId: com.ibm.ws.security.core.SecurityDM
  Category: INFO
  ExtendedMessage: CWSCJ0231I: The Security component's
  FFDC Diagnostic Module com.ibm.ws.security.core.SecurityDM
  registered success
fully: true.
Trace: 2003/08/25 13:06:31.086 01 t=9EA930 c=UNK key=P8 (0000000A)
  Description: Log Boss/390 Error
  from filename: ./bborjtr.cpp
  at line: 812
```

```
   error message: BBOO0222I CWSCJ0231I: The Security component's
FFDC Diagnostic Module com.ibm.ws.security.core.SecurityDM registered
  successfully: true.
 Trace: 2003/08/25 13:06:32.426 01 t=9EA930 c=UNK key=P8 (13007002)
   FunctionName: com.ibm.ws.security.core.SecurityComponentImpl
   SourceId: com.ibm.ws.security.core.SecurityComponentImpl
   Category: INFO
   ExtendedMessage: CWSCJ0309I: Java 2 Security is disabled.
 Trace: 2003/08/25 13:06:32.427 01 t=9EA930 c=UNK key=P8 (0000000A)
   Description: Log Boss/390 Error
   from filename: ./bborjtr.cpp
   at line: 812
   error message: BBOO0222I CWSCJ0309I: Java 2 Security is disabled.
 Trace: 2003/08/25 13:06:32.445 01 t=9EA930 c=UNK key=P8 (13007002)
   FunctionName: com.ibm.ws.security.core.SecurityComponentImpl
   SourceId: com.ibm.ws.security.core.SecurityComponentImpl
   Category: INFO
   ExtendedMessage: CWSCJ0212I: WCCM JAAS configuration information
successfully pushed to login provider class.
 Trace: 2003/08/25 13:06:32.445 01 t=9EA930 c=UNK key=P8 (0000000A)
   Description: Log Boss/390 Error
   from filename: ./bborjtr.cpp
   at line: 812
   error message: BBOO0222I CWSCJ0212I: WCCM JAAS configuration
 information successfully pushed to login provider class.
 Trace: 2003/08/25 13:06:32.459 01 t=9EA930 c=UNK key=P8 (13007002)
   FunctionName: SecurityComponentImpl
   SourceId: SecurityComponentImpl
   Category: WARNING
   ExtendedMessage: BBOS1000W  LTPA or ICSF are configured as the
authentication mechanism but SSO is disabled.
 Trace: 2003/08/25 13:06:32.459 01 t=9EA930 c=UNK key=P8 (0000000A)
   Description: Log Boss/390 Error
   from filename: ./bborjtr.cpp
   at line: 824
   error message: BBOS1000W  LTPA or ICSF are configured as the
authentication mechanism but SSO is disabled.
 Trace: 2003/08/25 13:06:32.463 01 t=9EA930 c=UNK key=P8 (13007002)
   FunctionName: com.ibm.ws.security.core.SecurityComponentImpl
   SourceId: com.ibm.ws.security.core.SecurityComponentImpl
   Category: INFO
   ExtendedMessage: CWSCJ0240I: Security service initialization completed
successfully
 Trace: 2003/08/25 13:06:32.463 01 t=9EA930 c=UNK key=P8 (0000000A)
   Description: Log Boss/390 Error
   from filename: ./bborjtr.cpp
   at line: 812
   error message: BBOO0222I CWSCJ0240I: Security service initialization
completed successfully
 Trace: 2003/08/25 13:06:39.718 01 t=9EA930 c=UNK key=P8 (13007002)
   FunctionName: com.ibm.ws.security.registry.UserRegistryImpl
   SourceId: com.ibm.ws.security.registry.UserRegistryImpl
   Category: AUDIT
   ExtendedMessage: CWSCJ0136I: Custom Registry:
com.ibm.ws.security.registry.zOS.SAFRegistryImpl has been initialized
 Trace: 2003/08/25 13:06:41.967 01 t=9EA930 c=UNK key=P8 (13007002)
   FunctionName: com.ibm.ws.security.core.WSAccessManager
   SourceId: com.ibm.ws.security.core.WSAccessManager
   Category: AUDIT
   ExtendedMessage: CWSCJ0157I: Loaded Vendor AuthorizationTable:
com.ibm.ws.security.core.SAFAuthorizationTableImpl
 Trace: 2003/08/25 13:06:43.136 01 t=9EA930 c=UNK key=P8 (13007002)
   FunctionName: com.ibm.ws.security.role.RoleBasedAuthorizerImpl
   SourceId: com.ibm.ws.security.role.RoleBasedAuthorizerImpl
   Category: AUDIT
   ExtendedMessage: CWSCJ0157I: Loaded Vendor AuthorizationTable:
com.ibm.ws.security.core.SAFAuthorizationTableImpl
```

```
Trace: 2003/08/25 13:06:43.789 01 t=9EA930 c=UNK key=P8 (13007002)
  FunctionName: com.ibm.ws.security.core.SecurityComponentImpl
  SourceId: com.ibm.ws.security.core.SecurityComponentImpl
  Category: INFO
  ExtendedMessage: CWSCJ0243I: Security service started successfully
Trace: 2003/08/25 13:06:43.789 01 t=9EA930 c=UNK key=P8 (0000000A)
  Description: Log Boss/390 Error
  from filename: ./bborjtr.cpp
  at line: 812
  error message: BBOO0222I CWSCJ0243I: Security service started successfully
Trace: 2003/08/25 13:06:43.794 01 t=9EA930 c=UNK key=P8 (13007002)
  FunctionName: com.ibm.ws.security.core.SecurityComponentImpl
  SourceId: com.ibm.ws.security.core.SecurityComponentImpl
  Category: INFO
  ExtendedMessage: CWSCJ0210I: Security enabled true
Trace: 2003/08/25 13:06:43.794 01 t=9EA930 c=UNK key=P8 (0000000A)
  Description: Log Boss/390 Error
  from filename: ./bborjtr.cpp
  at line: 812
  error message: BBOO0222I CWSCJ0210I: Security enabled true
Trace: 2003/08/25 13:07:06.474 01 t=9EA930 c=UNK key=P8 (13007002)
  FunctionName: com.ibm.ws.security.core.WSAccessManager
  SourceId: com.ibm.ws.security.core.WSAccessManager
  Category: AUDIT
  ExtendedMessage: CWSCJ0157I: Loaded Vendor AuthorizationTable:
com.ibm.ws.security.core.SAFAuthorizationTableImpl
Trace: 2003/08/25 13:07:09.315 01 t=9EA930 c=UNK key=P8 (13007002)
  FunctionName: com.ibm.ws.security.core.WSAccessManager
  SourceId: com.ibm.ws.security.core.WSAccessManager
  Category: AUDIT
  ExtendedMessage: CWSCJ0157I: Loaded Vendor AuthorizationTable:
com.ibm.ws.security.core.SAFAuthorizationTableImpl
```

Is there a stack trace or exception printed in the SystemOut.log?

A single stack trace tells a lot about the problem. What code initiated the code that failed? What is the failing component? Which class did the failure actually come from? Sometimes the stack trace is all that is needed to solve the problem and it can pinpoint the root cause. Other times, it can only give us a clue, and could actually be misleading. When support analyzes a stack trace, they may request additional trace if it is not clear what the problem is. If it appears to be security-related and the solution cannot be determined from the stack trace or problem description, you will be asked to gather the following trace specification: `SASRas=all=enabled:com.ibm.ws.security.*=all=enabled` from all processes involved.

Is this a distributed security problem or a local security problem?

- If the problem is local, that is the code involved does not make a remote method invocation, then troubleshooting is isolated to a single process. It is important to know when a problem is local versus distributed since the behavior of the ORB, among other components, is different between the two. Once a remote method invocation takes place, an entirely different security code path is entered.
- When you know that the problem involves two or more servers, the techniques of troubleshooting change. You will need to trace all servers involved simultaneously so that the trace shows the client and server sides of the problem. Try to make sure the timestamps on all machines match as closely as possible so that you can find the request and reply pair from two different processes. Enable both SAS or z/SAS and Security trace using the trace specification: `SASRas=all=enabled:com.ibm.ws.security.*=all=enabled`.

  For more information on enabling trace, see Enabling trace.

  For more information on enabling trace, see [Working with Trace

Is the problem related to authentication or authorization?

Most security problems fall under one of these two categories. Authentication is the process of determining who the caller is. Authorization is the process of validating that the caller has the proper authority to invoke the requested method. When authentication fails, typically this is related to either the authentication protocol, authentication mechanism or user registry. When authorization fails, this is usually related to the application bindings from assembly and/or deployment and to the caller's identity who is accessing the method and the roles required by the method.

Is this a Web or EJB request?

Web requests have a completely different code path than EJB requests. Also, there are different security features for Web requests than for EJB requests, requiring a completely different body of knowledge to resolve. For example, when using the LTPA authentication mechanism, the single signon feature (SSO) is available for Web requests but not for EJB requests. Web requests involve HTTP header information not required by EJB requests due to the protocol differences. Also, the Web container (or servlet engine) is involved in the entire process. Any of these components could be involved in the problem and all should be considered during troubleshooting, based on the type of request and where the failure occurs.

Secure EJB requests are passed from the controller to the servant. Web requests are mostly ignored by the controller. As a result, EJB requests are first processed and authenticated by the zSAS or CSIv2 layers of security. Authorization is done by the servant. If an authentication failure occurs, the zSAS type level of tracing must be turned on to diagnose the problem. Other problems can be diagnosed using the WebSphere Application Server component tracing (CTRACE) facility.

Does the problem seem to be related to the Secure Sockets Layer (SSL)?

The Secure Socket Layer (SSL) is a totally distinct separate layer of security. Troubleshooting SSL problems are usually separate from troubleshooting authentication and/or authorization problems. There are many things to consider. Usually, SSL problems are first time setup problems because the configuration can be difficult. Each client must contain the server's signer certificate. During mutual authentication, each server must contain the client's signer certificate. Also, there can be protocol differences (SSLv3 vs. TLS), and listener port problems related to stale IORs (i.e., IORs from a server reflecting the port prior to the server restarting).

In z/OS, two variations of SSL are used. To determine the cause of an SSL problem on z/OS, you will have to be aware of what protocol is being used. System SSL is used by the IIOP and HTTPS protocols. Java Secure Socket Extension (JSSE) is used by all other protocols, for example, Simple Object Access Protocol (SOAP). System SSL requests are handled in the controller and are used by z/SAS and CSIv1 security. JSSE is predominately used by the servant, but there are cases where it is used in the controller as well.

For SSL problems, we sometimes request an SSL trace to determine what is happening with the SSL handshake. The SSL handshake is the process which occurs when a client opens a socket to a server. If anything goes wrong with the key exchange, cipher exchange, etc. the handshake will fail and thus the socket is invalid. Tracing JSSE (the SSL implementation used in WebSphere Application Server) involves the following steps:

- Set the following system property on the client and server processes: `-Djavax.net.debug=true`. For the server, add this to the Generic JVM Arguments property of the Java virtual machine settings page.
- Recreate the problem.

  The SDSF active log of both processes should contain the JSSE trace. You will find trace similar to the following:

```
 JSSEContext: handleConnection[Socket
[addr=boss0106.plex1.l2.ibm.com/9.38.48.108,port=2139,localport=8878]]
 JSSEContext: handleConnection[Socket
[addr=boss0106.plex1.l2.ibm.com/9.38.48.108,port=2140,localport=8878]]
 TrustManagerFactoryImpl: trustStore is :
 /WebSphere/V5R0M0/AppServer/etc/DummyServerTrustFile.jks
 TrustManagerFactoryImpl: trustStore type is : JKS
 TrustManagerFactoryImpl: init truststore
 JSSEContext: handleConnection[Socket
[addr=boss0106.plex1.l2.ibm.com/9.38.48.108,port=2142,localport=8878]]
 KeyManagerFactoryImpl: keyStore is :
/WebSphere/V5R0M0/AppServer/etc/DummyServerKeyFile.jks
 KeyManagerFactoryImpl: keyStore type is : JKS
 KeyManagerFactoryImpl: init keystore
 KeyManagerFactoryImpl: init keystore
 JSSEContext: handleConnection[Socket
[addr=boss0106.plex1.l2.ibm.com/9.38.48.108,port=2143,localport=8878]]
 JSSEContext: handleSession[Socket
[addr=BOSSXXXX.PLEX1.L2.IBM.COM/9.38.48.108,port=8879,localport=2145]]
 JSSEContext:  confirmPeerCertificate
[Socket[addr=BOSSXXXX.PLEX1.L2.IBM.COM/9.38.48.108,port=8879,
  localport=2145]]
 X509TrustManagerImpl: checkServerTrusted
 X509TrustManagerImpl: Certificate [
 [
   Version: V3
   Subject: CN=jserver, OU=SWG, O=IBM, C=US
   Signature Algorithm: MD5withRSA, OID = 1.2.840.113549.1.1.4
0  Key:  IBMJCE RSA Public Key:
 modulus:
 100949966922395090747968287561185391075683695663138899555389506 68
66229530085897480010582163626382015770719020713112773657732526607 99
 128781182947273802312699983556527878615792292244995317112436562491
489904381884265119355037731265408654007388863303101746314438337601
 264540735679944205391693242921331551342247891
 public exponent:
 65537
0  Validity: [From: Fri Jun 21 20:08:18 GMT 2002,
               To: Thu Mar 17 20:08:18 GMT 2005]
   Issuer: CN=jserver, OU=SWG, O=IBM, C=US
   SerialNumber: [    3d1387b2 ]
0]
   Algorithm: [MD5withRSA]
   Signature:
 0000: 54 DC B5 FA 64 C9 CD FE   B3 EF 15 22 3D D0 20 31  T...d......"=. 1
 0010: 99 F7 A7 86 F9 4C 82 9F   6E 4B 7B 47 18 2E C6 25  .....L..nK.G...%
 0020: 5B B2 9B 78 D8 76 5C 82   07 95 DD B8 44 62 02 62  [..x.v\.....Db.b
 0030: 60 2A 0A 6D 4F B9 0A 98   14 27 E9 BB 1A 84 8A D1  `*.mO....'......
 0040: C2 22 AF 70 9E A5 DF A2   FD 57 37 CE 3A 63 1B EB  ."p.....W7.:c..
 0050: E8 91 98 9D 7B 21 4A B5   2C 94 FC A9 30 C2 74 72  .....!J.,...0.tr
 0060: 95 01 54 B1 29 E7 F8 9E   6D F3 B5 D7 B7 D2 9E 9B  ..T.)...m.......
 0070: 85 D8 E4 CF C2 D5 3B 64   F0 07 17 9E 1E B9 2F 79  ......;d....../y
0]
 X509TrustManagerImpl: Certificate [
 [
   Version: V3
   Subject: CN=jserver, OU=SWG, O=IBM, C=US
   Signature Algorithm: MD5withRSA, OID = 1.2.840.113549.1.1.4
0  Key:  IBMJCE RSA Public Key:
 modulus:
 100949966922395090747968287561185391075683695663138899555389506 6866
22953008589748001058216362638201577071902071311277365773252660 799
 128781182947273802312699983556527878615792292244995317112436562 4914
899043818842651193550377312654086540073888633031017463144383376 01
 264540735679944205391693242921331551342247891
 public exponent:
 65537
0  Validity: [From: Fri Jun 21 20:08:18 GMT 2002,
               To: Thu Mar 17 20:08:18 GMT 2005]
   Issuer: CN=jserver, OU=SWG, O=IBM, C=US
   SerialNumber: [    3d1387b2 ]
0]
   Algorithm: [MD5withRSA]
   Signature:
 0000: 54 DC B5 FA 64 C9 CD FE   B3 EF 15 22 3D D0 20 31  T...d......"=. 1
 0010: 99 F7 A7 86 F9 4C 82 9F   6E 4B 7B 47 18 2E C6 25  .....L..nK.G...%
 0020: 5B B2 9B 78 D8 76 5C 82   07 95 DD B8 44 62 02 62  [..x.v\.....Db.b
 0030: 60 2A 0A 6D 4F B9 0A 98   14 27 E9 BB 1A 84 8A D1  `*.mO....'......
 0040: C2 22 AF 70 9E A5 DF A2   FD 57 37 CE 3A 63 1B EB  ."p.....W7.:c..
 0050: E8 91 98 9D 7B 21 4A B5   2C 94 FC A9 30 C2 74 72  .....!J.,...0.tr
 0060: 95 01 54 B1 29 E7 F8 9E   6D F3 B5 D7 B7 D2 9E 9B  ..T.)...m.......
 0070: 85 D8 E4 CF C2 D5 3B 64   F0 07 17 9E 1E B9 2F 79  ......;d....../y
0]
 JSSEContext: handleConnection[Socket[addr=boss0106.plex1.l2.ibm.com
/9.38.48.108,port=2144,localport=8878]]
 JSSEContext: handleConnection[Socket[addr=BOSSXXXX.PLEX1.L2.IBM.COM
/9.38.48.108,port=8879,localport=2145]]
 JSSEContext: handleConnection[Socket[addr=BOSSXXXX.PLEX1.L2.IBM.COM
/9.38.48.108,port=8879,localport=2146]]
 JSSEContext: handleConnection[Socket[addr=BOSSXXXX.PLEX1.L2.IBM.COM
/9.38.48.108,port=8879,localport=2147]]
 JSSEContext: handleConnection[Socket[addr=BOSSXXXX.PLEX1.L2.IBM.COM
```

```
/9.38.48.108,port=8879,localport=2148]]
 JSSEContext: handleConnection[Socket[addr=BOSSXXXX.PLEX1.L2.IBM.COM
/9.38.48.108,port=8879,localport=2149]]
 JSSEContext: handleConnection[Socket[addr=BOSSXXXX.PLEX1.L2.IBM.COM
/9.38.48.108,port=8879,localport=2150]]
 JSSEContext: handleConnection[Socket[addr=BOSSXXXX.PLEX1.L2.IBM.COM
/9.38.48.108,port=8879,localport=2151]]
 JSSEContext: handleConnection[Socket[addr=BOSSXXXX.PLEX1.L2.IBM.COM
/9.38.48.108,port=8879,localport=2152]]
 JSSEContext: handleConnection[Socket[addr=BOSSXXXX.PLEX1.L2.IBM.COM
/9.38.48.108,port=8879,localport=2153]]
 JSSEContext: handleConnection[Socket[addr=BOSSXXXX.PLEX1.L2.IBM.COM
/9.38.48.108,port=8879,localport=2154]]
 JSSEContext: handleConnection[Socket[addr=BOSSXXXX.PLEX1.L2.IBM.COM
/9.38.48.108,port=8879,localport=2155]]
 JSSEContext: handleConnection[Socket[addr=BOSSXXXX.PLEX1.L2.IBM.COM
/9.38.48.108,port=8879,localport=2156]]
 JSSEContext: handleConnection[Socket[addr=BOSSXXXX.PLEX1.L2.IBM.COM
/9.38.48.108,port=8879,localport=2157]]
 JSSEContext: handleConnection[Socket[addr=BOSSXXXX.PLEX1.L2.IBM.COM
/9.38.48.108,port=8879,localport=2158]]
 JSSEContext: handleConnection[Socket[addr=BOSSXXXX.PLEX1.L2.IBM.COM
/9.38.48.108,port=8879,localport=2159]]
 JSSEContext: handleConnection[Socket[addr=BOSSXXXX.PLEX1.L2.IBM.COM
/9.38.48.108,port=8879,localport=2160]]
 JSSEContext: handleConnection[Socket[addr=BOSSXXXX.PLEX1.L2.IBM.COM
/9.38.48.108,port=8879,localport=2161]]
 JSSEContext: handleConnection[Socket[addr=BOSSXXXX.PLEX1.L2.IBM.COM
/9.38.48.108,port=8879,localport=2162]]
 JSSEContext: handleConnection[Socket[addr=BOSSXXXX.PLEX1.L2.IBM.COM
/9.38.48.108,port=8879,localport=2163]]
 JSSEContext: handleConnection[Socket[addr=BOSSXXXX.PLEX1.L2.IBM.COM
/9.38.48.108,port=8879,localport=2164]]
 JSSEContext: handleConnection[Socket[addr=BOSSXXXX.PLEX1.L2.IBM.COM
/9.38.48.108,port=8879,localport=2165]]
 JSSEContext: handleConnection[Socket[addr=BOSSXXXX.PLEX1.L2.IBM.COM
/9.38.48.108,port=8879,localport=2166]]

 JSSEContext: handleSession[Socket[addr=boss0106.plex1.l2.ibm.com
/9.38.48.108,port=9443,localport=2167]]
 JSSEContext:  confirmPeerCertificate[Socket[addr=boss0106.plex1.l2.ibm.com
/9.38.48.108,port=9443,localport=2167]]
 X509TrustManagerImpl: checkServerTrusted
 X509TrustManagerImpl: Certificate [
 [
   Version: V3
   Subject: CN=WAS z/OS Deployment Manager, O=IBM
   Signature Algorithm: SHA1withRSA, OID = 1.2.840.113549.1.1.5
0  Key:  IBMJCE RSA Public Key:
 modulus:
 12840948267119651469312486548020957441946413494498370439558603901582589
875503344841953410518313306436646682874151642817657944051 1007
 6258795528749232737808897160958348495006972731464152299032614592135114
193615399625559971360851405910982593456258536173893963406647 66
 649957749527841107121590352429348634287031501
 public exponent:
 65537
0  Validity: [From: Fri Jul 25 05:00:00 GMT 2003,
               To: Mon Jul 26 04:59:59 GMT 2004]
   Issuer: CN=WAS CertAuth, C=US
   SerialNumber: [    02]
0Certificate Extensions: 3
 [1]: ObjectId: 2.16.840.1.113730.1.13 Criticality=false
 Extension unknown: DER encoded OCTET string =
 0000: 04 3C 13 3A 47 65 6E 65   72 61 74 65 64 20 62 79  .<.:Generated by
 0010: 20 74 68 65 20 53 65 63   75 72 65 57 61 79 20 53   the SecureWay S
 0020: 65 63 75 72 69 74 79 20   53 65 72 76 65 72 20 66  ecurity Server f
 0030: 6F 72 20 7A 2F 4F 53 20   28 52 41 43 46 29        or z/OS (RACF)
-[2]: ObjectId: 2.5.29.14 Criticality=false
 SubjectKeyIdentifier [
 KeyIdentifier [
 0000: 05 6A CD 7F AE AF 89 78   99 A8 F1 5B 64 8B 9F AF  .j.....x...[d...
 0010: 73 1B 58 65                                        s.Xe
 ]
 ]
0[3]: ObjectId: 2.5.29.35 Criticality=false
 AuthorityKeyIdentifier [
 KeyIdentifier [
 0000: 7E D1 7B 17 74 D3 AD D1   7D D8 F8 33 85 19 04 F8  ....t......3....
 0010: 36 51 57 16                                        6QW.
 ]
0]
0]
   Algorithm: [SHA1withRSA]
   Signature:
 0000: 73 0D FC E1 8A B3 42 E1   04 73 72 B1 C6 C9 87 54  s.....B..sr....T
 0010: 87 57 02 FA 41 32 D8 B0   39 09 86 CB 6B 03 B6 F9  .W..A2..9...k...
 0020: 62 8D 95 36 56 0E D4 D2   F7 7A 8D 4B FB 0B FD 91  b..6V....z.K....
 0030: 89 A8 08 41 30 E2 27 DC   15 5F 2C F4 CD 2F 6B 8E  ...A0.'._,../k.
 0040: 21 2A 88 53 46 27 68 9B   55 14 38 8E 1F 50 95 BC  !*.SF'h.U.8..P..
 0050: A8 46 F6 68 97 9E 7B 65   9E E8 A7 34 B2 C8 63 CF  .F.h...e...4..c.
 0060: 73 C8 4E 25 0A EF C5 8F   04 A4 EB 8C CC 33 84 26  s.N%.........3.&
 0070: 5D FD 7C AD 7B 02 13 5A   86 A1 89 93 1E A4 93 63  ]......Z.......c
0]
 X509TrustManagerImpl: Certificate [
 [
   Version: V3
   Subject: CN=WAS CertAuth, C=US
   Signature Algorithm: SHA1withRSA, OID = 1.2.840.113549.1.1.5
0  Key:  IBMJCE RSA Public Key:
 modulus:
 11674085937333316022183855781833894964845874186386763528295600405299 18
405586812081999778334016098957482223690662303297851488832511 44
 2382911186804921983976695395381692334250582278359056431484427844566504
```

```
414917999525928648952429870379294084534556275527723173820770 15
 828713585220212502839546496071839496308430393
 public exponent:
 65537
0  Validity: [From: Fri Jul 25 05:00:00 GMT 2003,
               To: Sat Jul 24 04:59:59 GMT 2010]
   Issuer: CN=WAS CertAuth, C=US
   SerialNumber: [  0  ]
0Certificate Extensions: 4
 [1]: ObjectId: 2.16.840.1.113730.1.13 Criticality=false
 Extension unknown: DER encoded OCTET string =
 0000: 04 3C 13 3A 47 65 6E 65   72 61 74 65 64 20 62 79  .<.:Generated by
 0010: 20 74 68 65 20 53 65 63   75 72 65 57 61 79 20 53   the SecureWay S
 0020: 65 63 75 72 69 74 79 20   53 65 72 76 65 72 20 66  ecurity Server f
 0030: 6F 72 20 7A 2F 4F 53 20   28 52 41 43 46 29        or z/OS (RACF)
-[2]: ObjectId: 2.5.29.14 Criticality=false
 SubjectKeyIdentifier [
 KeyIdentifier [
 0000: 7E D1 7B 17 74 D3 AD D1   7D D8 F8 33 85 19 04 F8  ....t......3....
 0010: 36 51 57 16                                        6QW.
 ]
 ]
0[3]: ObjectId: 2.5.29.15 Criticality=true
 KeyUsage [
   Key_CertSign
   Crl_Sign
 ]
0[4]: ObjectId: 2.5.29.19 Criticality=true
 BasicConstraints:[
 CA:true
 PathLen:2147483647
 ]
0]
   Algorithm: [SHA1withRSA]
   Signature:
 0000: 43 88 AB 19 5D 00 54 57   5E 96 FA 85 CE 88 4A BF  C...].TW^.....J.
 0010: 6E CB 89 4C 56 BE EF E6   8D 2D 74 B5 83 1A EF 9C  n..LV....-t.....
 0020: B3 82 F2 16 84 FA 5C 50   53 2A B4 FD EB 27 98 5D  ......\PS*...'.]
 0030: 43 48 D3 74 85 21 D1 E1   F2 63 9E FB 58 2A F3 6A  CH.t.!...c..X*.j
 0040: 44 D2 F5 7D B2 55 B9 5E   32 11 78 B6 34 8E 4B 1D  D....U.^2.x.4.K.
 0050: F3 82 1D C1 5F 7B 3F AD   C9 29 FA FF D1 D1 13 2C  ...._.?..).....,
 0060: 57 F7 7B 51 02 99 6F ED   54 E1 51 34 B8 51 BE 97  W..Q..o.T.Q4.Q..
 0070: 30 AC 4F 89 AB AA 8A B2   E1 40 89 2E 18 C7 0E 15  0.O......@......
0]
 JSSEContext: handleConnection[Socket[addr=boss0106.plex1.l2.ibm.com
/9.38.48.108,port=9443,localport=2167]]
 JSSEContext: handleConnection[Socket[addr=BOSSXXXX.PLEX1.L2.IBM.COM
/9.38.48.108,port=8879,localport=2168]]

 JSSEContext: handleConnection[Socket[addr=boss0106.plex1.l2.ibm.com
/9.38.48.108,port=2235,localport=8878]]
 JSSEContext: handleConnection[Socket[addr=BOSSXXXX.PLEX1.L2.IBM.COM
/9.38.48.108,port=8879,localport=2236]]
 JSSEContext: handleSession[Socket[addr=BOSSXXXX.PLEX1.L2.IBM.COM
/9.38.48.108,port=8880,localport=2238]]
 JSSEContext:  confirmPeerCertificate[Socket
[addr=BOSSXXXX.PLEX1.L2.IBM.COM
/9.38.48.108,port=8880,localport=2238]]
 X509TrustManagerImpl: checkServerTrusted
 X509TrustManagerImpl: Certificate [

 [
   Version: V3
   Subject: CN=jserver, OU=SWG, O=IBM, C=US
   Signature Algorithm: MD5withRSA, OID = 1.2.840.113549.1.1.4
0  Key:  IBMJCE RSA Public Key:
 modulus:
 100949966922395090747968287561185391075683695663138899555389506686622953
008589748001058216362638201577071902071311277365773252660799
 128781182947273802312699983556527878615792292244995317112436562491 4
899043818842651193550377312654086540073888633031017463144383376 01
 264540735679944205391693242921331551342247891
 public exponent:
 65537
0  Validity: [From: Fri Jun 21 20:08:18 GMT 2002,
               To: Thu Mar 17 20:08:18 GMT 2005]
   Issuer: CN=jserver, OU=SWG, O=IBM, C=US
   SerialNumber: [    3d1387b2 ]
0]
   Algorithm: [MD5withRSA]
   Signature:
 0000: 54 DC B5 FA 64 C9 CD FE   B3 EF 15 22 3D D0 20 31  T...d......"=. 1
 0010: 99 F7 A7 86 F9 4C 82 9F   6E 4B 7B 47 18 2E C6 25  .....L..nK.G...%
 0020: 5B B2 9B 78 D8 76 5C 82   07 95 DD B8 44 62 02 62  [..x.v\.....Db.b
 0030: 60 2A 0A 6D 4F B9 0A 98   14 27 E9 BB 1A 84 8A D1  `*.mO....'......
 0040: C2 22 AF 70 9E A5 DF A2   FD 57 37 CE 3A 63 1B EB  .".p.....W7.:c..
 0050: E8 91 98 9D 7B 21 4A B5   2C 94 FC A9 30 C2 74 72  .....!J.,...0.tr
 0060: 95 01 54 B1 29 E7 F8 9E   6D F3 B5 D7 B7 D2 9E 9B  ..T.)...m.......
 0070: 85 D8 E4 CF C2 D5 3B 64   F0 07 17 9E 1E B9 2F 79  ......;d....../y
0]
 X509TrustManagerImpl: Certificate [
 [
   Version: V3
   Subject: CN=jserver, OU=SWG, O=IBM, C=US
   Signature Algorithm: MD5withRSA, OID = 1.2.840.113549.1.1.4
0  Key:  IBMJCE RSA Public Key:
 modulus:
 100949966922395090747968287561185391075683695663138899555389506
686622953008589748001058216362638201577071902071311277365773252660799
 128781182947273802312699983556527878615792292244995317112436562491
489904381884265119355037731265408654007388863303101746314438337601
 264540735679944205391693242921331551342247891
 public exponent:
 65537
```

```
0  Validity: [From: Fri Jun 21 20:08:18 GMT 2002,
               To: Thu Mar 17 20:08:18 GMT 2005]
   Issuer: CN=jserver, OU=SWG, O=IBM, C=US
   SerialNumber: [    3d1387b2 ]
0]
   Algorithm: [MD5withRSA]
   Signature:
 0000: 54 DC B5 FA 64 C9 CD FE   B3 EF 15 22 3D D0 20 31  T...d......"=. 1
 0010: 99 F7 A7 86 F9 4C 82 9F   6E 4B 7B 47 18 2E C6 25  .....L..nK.G...%
 0020: 5B B2 9B 78 D8 76 5C 82   07 95 DD B8 44 62 02 62  [..x.v\.....Db.b
 0030: 60 2A 0A 6D 4F B9 0A 98   14 27 E9 BB 1A 84 8A D1  `*.mO....'......
 0040: C2 22 AF 70 9E A5 DF A2   FD 57 37 CE 3A 63 1B EB  ."p.....W7.:c..
 0050: E8 91 98 9D 7B 21 4A B5   2C 94 FC A9 30 C2 74 72  .....!J.,...0.tr
 0060: 95 01 54 B1 29 E7 F8 9E   6D F3 B5 D7 B7 D2 9E 9B  ..T.)...m.......
 0070: 85 D8 E4 CF C2 D5 3B 64   F0 07 17 9E 1E B9 2F 79  ......;d....../y
0]
 JSSEContext: handleConnection[Socket[addr=BOSSXXXX.PLEX1.L2.IBM.COM/
9.38.48.108,port=8880,localport=2238]]
 JSSEContext: handleConnection[Socket[addr=BOSSXXXX.PLEX1.L2.IBM.COM/
9.38.48.108,port=8880,localport=2239]]
 JSSEContext: handleConnection[Socket[addr=BOSSXXXX.PLEX1.L2.IBM.COM/
9.38.48.108,port=8880,localport=2240]]
 JSSEContext: handleConnection[Socket[addr=BOSSXXXX.PLEX1.L2.IBM.COM/
9.38.48.108,port=8880,localport=2241]]
```

Trace security

The classes which implement WebSphere Application Server security are:
- com.ibm.ws.security.*
- com.ibm.websphere.security.*
- com.ibm.WebSphereSecurityImpl.*

For current information available from IBM Support on known problems and their resolution, see the IBM Support page.

IBM Support has documents that can save you time gathering information needed to resolve this problem. Before opening a PMR, see the IBM Support page.

Related tasks

Chapter 6, “Troubleshooting by task,” on page 233

JavaServer Pages source code shown by the Web server

Problem

If you share the document root of the WebSphere Application Server with the Web server document root, a security exposure can result as the Web server might display the JavaServer Pages (JSP) source file as plain text.

You can use the WebSphere Web server plug-in set of rules to determine whether a given request will be handled by the WebSphere Application Server. When an incoming request fails to match those rules, the Web server plug-in returns control to the Web server so that the Web server can fulfill the request. In this case, the unknown host header causes the Web server plug-in to return control to the Web server because the rules do not indicate that the WebSphere Application Server should handle it. Therefore, the Web server looks for the request in the Web server document root. Since the JSP source file is stored in the document root of the Web server, the Web server finds the file and displays it as plain text.

Suggested solution

Move the WebSphere Application Server JSP source file outside of the Web server document root. Then, when this request comes in with the unknown host header, the plug-in returns control to the Web server and the JSP source file is not found in the document root. Therefore, the Web server returns a `404 File Not Found` error rather than the JSP source file.

Naming and directory

Naming services component troubleshooting tips

Naming is a J2EE service which publishes and provides access to resources such as connection pools, enterprise beans, and message listeners to client processes. If you have problems in accessing a resource which otherwise appears to be healthy, the naming service might be involved. To investigate problems with the WebSphere Application Server Naming service:

- With WebSphere Application Server running, run the **dumpNameSpace** command for Windows systems, or the **dumpNameSpace.sh** command for UNIX and z/OS systems, and pipe, redirect, or "more" the output so that it is easily viewed. This command results in a display of objects in the WebSphere Application Server namespace, including the directory path and object name.
- If the object a client needs to access does not appear, use the administrative console to verify that:
  - The server hosting the target resource is started.
  - The Web module or EJB container, if applicable, hosting the target resource is running.
  - The JNDI name of the target resource is correct and updated.
  - If the problem resource is remote, that is, not on the same node as the Name Server node, that the JNDI name is fully qualified, including the host name. This is especially applicable to Network Deployment configurations
- If you see an exception that appears to be CORBA related ("CORBA" appears as part of the exception name) look for a naming-services-specific CORBA minor code, further down in the exception stack, for information on the real cause of the problem. For a list of naming service exceptions and explanations, see the class com.ibm.websphere.naming.WsnCorbaMinorCodes in the Javadoc.

If none of these steps solve the problem:

- For specific problems that can cause access to named object hosted in WebSphere Application Server to fail, see Cannot look up an object hosted by WebSphere Application Server from a servlet, JSP file, or other client.
- Check to see if the problem has been identified and documented using the links in Diagnosing and fixing problems: Resources for learning.

For current information available from IBM Support on known problems and their resolution, see the IBM Support page.

IBM Support has documents that can save you time gathering information needed to resolve this problem. Before opening a PMR, see the IBM Support page.

Related tasks

Chapter 6, "Troubleshooting by task," on page 233

Related reference

Troubleshooting installation problems

Cannot look up an object hosted by WebSphere Application Server from a servlet, JSP file, or other client

To resolve problems encountered when a servlet, JSP file, standalone application or other client attempts to access an enterprise bean, ConnectionPool, or other named object hosted by WebSphere Application Server, you must first verify that the target server can be accessed from the client:

- From a command prompt on the client's server, enter "`ping` *`server_name`*" and verify connectivity.

- Use the WebSphere Application Server administrative console to verify that the target resource's application server and, if applicable, EJB module or Web module, is started.

Continue only if there is no problem with connectivity and the target resource appears to be running.

What kind of error are you seeing?
- "NameNotFoundException from JNDI lookup operation " on page 47
- "CannotInstantiateObjectException from JNDI lookup operation " on page 47
- "Message NMSV0610I appears in the server's log file, indicating that some Naming exception has occurred " on page 48
- "OperationNotSupportedException from JNDI Context operation" on page 48
- "WSVR0046E: Failed to bind, ejb/jndiName: ejb/jndiName. Original exception : org.omg.CosNaming.NamingContextPackage.AlreadyBound " on page 48
- "ConfigurationException from "new InitialContext" operation or from a JNDI Context operation with a URL name" on page 48
- "ServiceUnavailableException from "new InitialContext" operation" on page 49
- "CommunicationException thrown from a "new InitialContext" operation" on page 49

NameNotFoundException from JNDI lookup operation

If you encounter this exception in trying to access an enterprise bean, data source, messaging resource, or other resource:
- Browse the properties of the target object in the administrative console, and verify that the jndi name it specifies matches the JNDI name the client is using.
- If you are looking up an object that resides on a server different from the one from which the initial context was obtained, you must use the fully qualified name.
  - If access is from another server object such as a servlet accessing an enterprise bean and you are using the default context, not specifying the fully qualified JNDI name, you may get this error if the object is being hosted on a different server.
  - If access is from a stand-alone client, it may be that the object you are attempting access is on a server different from the server from which you obtained the initial context.

To correct this problem, use the fully-qualified JNDIname:
- If the object is in a single server: `cell/nodes/nodeName/servers/serverName/jndiName`. Objects are not supported in this release.
- If the object is on a server cluster: `cell/clusters/clusterName/jndiName`.

CannotInstantiateObjectException from JNDI lookup operation

If you encounter this exception in trying to access an enterprise bean, data source, messaging resource, or other resource, possible causes include:
- A serialized Java object is being looked up, but the necessary classes required to deserialize it are not in the runtime environment.
- A Reference object is being looked up, and the associated factory used to process it as part of the lookup processing is failing.

To determine the precise cause of the problem:
- Look in the logs of the server hosting the target resource. Look for exceptions immediately preceding the CannotInstantiateObjectException. If it is a

java.lang.NoClassDefFoundError or java.lang.ClassNotFoundException, make sure the class referenced in the error message can be located by the class loader.
- Print out the stack trace for the root cause and look for the factory class. It will be called by `javax.naming.NamingManager.getObjectInstance()`. The reason for the failure will depend on the factory implementation, and may require you to contact the developer of the factory class.

Message NMSV0610I appears in the server's log file, indicating that some Naming exception has occurred

This error is informational only and is provided in case the exception is related to an actual problem. Most of the time, it is not. If it is, the log file should contain adjacent entries to provide context.
- If no problems are being experienced, ignore this message. Also ignore the message if the problem you are experiencing does not appear to be related to the exception being reported and if there are no other adjacent error messages in the log.
- If a problem is being experienced, look in the log for underlying error messages.
- The information provided in message NMSV0610I can provide valuable debug data for other adjacent error messages posted in response to the Naming exception that occurred.

OperationNotSupportedException from JNDI Context operation

This error has two possible causes:
- An update operation, such as a bind, is being performed with a name that starts with "`java:comp/env`". This context and its subcontexts are read-only contexts.
- A Context bind or rebind operation of a non-CORBA object is being performed on a remote name space that does not belong to WebSphere Application Server. Only CORBA objects can be bound to these CosNaming name spaces.

To determine which of these errors is causing the problem, check the full exception message.

WSVR0046E: Failed to bind, ejb/jndiName: ejb/jndiName. Original exception : org.omg.CosNaming.NamingContextPackage.AlreadyBound

This error occurs two enterprise bean server applications were installed on the same server such that a binding name conflict occurred. That is, a jndiName value is the same in the two applications' deployment descriptors. The error will surface during server startup when the second application using that jndiName value is started.

To verify that this is the problem, examine the deployment descriptors for all enterprise bean server applications running in the server in search for a jndiName that is specified in more than one enterprise bean application.

To correct the problem, change any duplicate jndiName values to ensure that each enterprise bean in the server process is bound with a different name.

ConfigurationException from "new InitialContext" operation or from a JNDI Context operation with a URL name

If you are attempting to obtain an initial JNDI context, a configuration exception can occur because an invalid JNDI property value was passed to the InitialContext constructor. This includes JNDI properties set in the System properties or in some

jndi.properties file visible to the class loader in effect. A malformed provider URL is the most likely property to be incorrect. If the JNDI client is being run as a thin client such that the CLASSPATH is set to include all of the individual jar files required, make sure the .jar file containing the properties file com/ibm/websphere/naming/jndiprovider.properties is in the CLASSPATH.

If the exception is occurring from a JNDI Context call with a name in the form of a URL, the current JNDI configuration may not be set up properly so that the required factory class name cannot be determined, or the factory may not be visible to the class loader currently in effect. If the name is a Java: URL, the JNDI client must be running in a J2EE client or server environment. That is, the client must be running in a container.

Check the exception message to verify the cause.

If the exception is being thrown from the InitialContext constructor, correct the property setting or the CLASSPATH.

If the exception is being thrown from a JNDI Context method, make sure the property java.naming.factory.url.pkgs includes the package name for the factory required for the URL scheme in the name. URL names with the Java scheme can only be used while running in a container.

ServiceUnavailableException from "new InitialContext" operation

This exception indicates that some unexpected problem occurred while attempting to contact the name server to obtain an initial context. The ServiceUnavailableException, like all NamingException objects, can be queried for a root cause. Check the root cause for more information. It is possible that some of the problems described for CommunicationExceptions may also result in a ServiceUnavailableException.

Since this exception is triggered by an unexpected error, there is no probable cause to confirm. If the root cause exception does not indicate what the probable cause is, investigate the possible causes listed for CommunicationExceptions.

CommunicationException thrown from a "new InitialContext" operation

The name server identified by the provider URL cannot be contacted to obtain the initial JNDI context. There are many possible causes for this problem, including:
- The host name or port in the provider URL is incorrect.
- The host name cannot be resolved into an IP address by the domain name server, or the IP address does not match the IP address which the server is actually running under.
- A firewall on the client or server is preventing the port specified in the provider URL from being used.

To correct this problem:
- Make sure the provider URL and the network configurations on the client and server machines are correct.
- Make sure the host name can be resolved into an IP address which can be reached by the client machine. You can do this using the ping command.
- If you are running a firewall, make sure that use of the port specified in the provider URL will be allowed.

Related tasks

Chapter 6, “Troubleshooting by task,” on page 233

Troubleshooting name space problems

Many naming problems can be avoided by fully understanding the key underlying concepts of WebSphere Application Server naming.

1. Review the key concepts of WebSphere Application Server naming, especially Name space logical view and Lookup names support in deployment descriptors and thin clients.
2. Review the programming examples that are included in the sections explaining the JNDI and CosNaming interfaces.
3. Read “Naming services component troubleshooting tips” on page 45 for additional general information.
4. If you “Cannot look up an object hosted by WebSphere Application Server from a servlet, JSP file, or other client” on page 46, read this article.

Related tasks

“Naming services component troubleshooting tips” on page 45

“Cannot look up an object hosted by WebSphere Application Server from a servlet, JSP file, or other client” on page 46

Related reference

Lookup names support in deployment descriptors and thin clients

dumpNameSpace tool

You can use the dumpNameSpace tool to dump the contents of a name space accessed through a name server. The dumpNameSpace tool is based on Java Naming and Directory Interface (JNDI).

When you run the dumpNameSpace tool, the naming service must be active. The dumpNameSpace tool cannot dump name spaces local to the server process, such as those with java: and local: URL schemes. The local: name space contains references to enterprise beans with local interfaces. Use the name space dump utility for java:, local: and server name spaces to dump java: and local: name spaces.

The tool dumps the server root context for the server at the specified host and port unless you specify a non-default starting context which precludes it. The tool does not dump the server root contexts for other servers.

Running dumpNameSpace

You can run the tool from a command line or using its program interface. This article describes command-line invocations. To access the dumpNameSpace tool through its program interface, refer to the class `com.ibm.websphere.naming.DumpNameSpace` in the WebSphere Application Server API documentation. “Example: Invoking the name space dump tool” on page 382 illustrates running the tool from a command line or using its program interface.

To run the tool from a command line, enter the following command from the *WebSphere/AppServer/bin* directory:

Platform	Command
UNIX	`dumpNameSpace.sh [[-keyword value]...]`

Platform	Command
Windows NT or later	`dumpNameSpace [[-keyword value]...]`

If you run the dumpNameSpace tool with security enabled, a login prompt is displayed. If you cancel the login prompt, the dumpNameSpace tool continues outbound with an "UNAUTHENTICATED" credential. Thus, by default, an "UNAUTHENTICATED" credential is used that is equivalent to the "Everyone" access authorization policy. You can modify this default setting by changing the value for the `com.ibm.CSI.performClientAuthenticationRequired` property to `true` in the *`install_dir`*`/properties/sas.client.props` file. If you change this property to `true`, rerun the dumpNameSpace tool and cancel the login prompt; the authorization fails and the command does not continue outbound.

Parameters

The keywords and associated values for the dumpNameSpace tool follow:

-host *myhost.company.com*
Indicates the bootstrap host or the WebSphere Application Server host whose name space you want to dump. The value defaults to `localhost`. Specify a value for `-host` if the tool is not run from the local machine. The `-host` parameter instructs the tool to connect to a server on a remote machine. For example, run

```
dumpNameSpace -host myhost.mycompany.com
```

to display the name space of the server running on *myhost.mycompany.com*.

-port *nnn*
Indicates the bootstrap port which, if not specified, defaults to `2809`.

-root { cell | server | node | host | legacy | tree | default }
Indicates the root context to use as the initial context for the dump. The applicable root options and default root context depend on the type of name server from which the dump is being obtained. Descriptions of root options follow:

cell	DumpNameSpace default. Dumps the tree starting at the cell root context.
server	Dumps the tree starting at the server root context.
node	Dumps the tree starting at the node root context. The `node` value is synonymous with `host`.

For WebSphere Application Servers Version 4.0 or later:

legacy	DumpNameSpace default. Dumps the tree starting at the legacy root context.
host	Dumps the tree starting at the bootstrap host root context. The `host` value is synonymous with `node`.
tree	Dumps the tree starting at the tree root context.

For all WebSphere Application Servers and other name servers:

default	Dumps the tree starting at the initial context which JNDI returns by default for that server type. This is the only `-root` choice that is compatible with WebSphere Application Servers prior to Version 4.0 and with non-WebSphere Application Server name servers.

-url *some_provider_URL*
Indicates the value for the java.naming.provider.url property used to get the initial JNDI context. This option can be used in place of the `-host`, `-port`, and `-root` options. If the `-url` option is specified, the `-host`, `-port`, and `-root` options are ignored.

-factory *com.ibm.websphere.naming.WsnInitialContextFactory*
Indicates the initial context factory to be used to get the JNDI initial context. The value defaults to *com.ibm.websphere.naming.WsnInitialContextFactory*. The default value generally does not need to be changed.

-startAt *some/subcontext/in/the/tree*
Indicates the path from the bootstrap host's root context to the top level context where the dump should begin. The tool recursively dumps subcontexts below this point. It defaults to an empty string, that is, the bootstrap host root context.

-format { jndi | ins }

jndi	The default. Displays name components as atomic strings.
ins	Shows name components parsed using Interoperable Naming Service (INS) rules (`id.kind`).

-report { short | long }

short	The default. Dumps the binding name and bound object type. This output is also provided by JNDI Context.list().
long	Dumps the binding name, bound object type, local object type, and string representation of the local object (that is, the IORs, string values, and other values that are printed). For objects of user-defined classes to display correctly with the long report option, you might need to add their containing directories to the list of directories searched. Set the environment variable **WAS_USER_DIRS** as shown in the following platform-specific commands. The value can include one or more directories. **UNIX** `WAS_USER_DIRS=/usr/classdir1:/usr/classdir2 export WAS_USER_DIRS` **Windows NT or later** `set WAS_USER_DIRS=c:\classdir1;d:\classdir2` All `.zip`, `.jar`, and `.class` files in the specified directories can then be resolved by the class loader when running the dumpNameSpace tool.

-traceString *"some.package.name.to.trace.*=all=enabled"*
Represents the trace string with the same format as that generated by the servers. The output is sent to the file `DumpNameSpaceTrace.out`.

Related tasks

"Troubleshooting name space problems" on page 379

Related reference

"Example: Invoking the name space dump tool"

"Name space dump utility for java:, local: and server name spaces" on page 383

"Example: Invoking the name space dump utility for java: and local: name spaces" on page 385

"Name space dump sample output" on page 386

Example: Invoking the name space dump tool

It is often helpful to view a dump of the name space to understand why a naming operation is failing. You can invoke the name space dump tool from the command line or from a program. Examples of each option follow.

Invoking the name space dump tool from a command line

Invoke the name space dump tool from a command line by entering either of the following commands:

```
dumpNameSpace -host myhost.mycompany.com -port 901
```

or:

```
dumpNameSpace -url corbaloc:iiop:myhost.mycompany.com:901
```

There are several command-line options to choose from. For detailed help on the options, enter either of the following commands:

```
dumpNameSpace -help
```

or:

```
dumpNameSpace -?
```

Use the dumpNameSpace.sh command. (Add .sh to the utility name.)

Invoking the name space dump tool from a Java program

You can dump name spaces from a program with the com.ibm.websphere.naming.DumpNameSpace API. Refer to the WebSphere Application Server API documentation for details on the DumpNameSpace program interface.

The following example illustrates how to invoke the name space dump tool from a Java program:

```
{
   ...
   import javax.naming.Context;
   import javax.naming.InitialContext;
   import com.ibm.websphere.naming.DumpNameSpace;
   ...
   java.io.PrintStream filePrintStream = ...
   Context ctx = new InitialContext();
   // Starting context for dump
   ctx = (Context) ctx.lookup("cell/nodes/node1/servers/server1");
   DumpNameSpace dumpUtil =
      new DumpNameSpace(filePrintStream, DumpNameSpace.SHORT);
   dumpUtil.generateDump(ctx);
   ...
}
```

Related tasks

"Troubleshooting name space problems" on page 379

Related reference

"dumpNameSpace tool" on page 379

Name space dump utility for `java:`, `local:` and `server` name spaces

Sometimes it is helpful to dump the `java:` name space for a J2EE application. You cannot use the dumpNameSpace command line utility for this purpose because the application's `java:` name space is accessible only by that J2EE application. From the WebSphere Application Server scripting tool, you can invoke a NameServer MBean to dump the `java:` name space for any J2EE application running in that same server process.

There is another name space local to server process which you cannot dump with the dumpNameSpace command line utility. This name space has the URL scheme of local: and is used by the container to bind objects locally instead of through the name server. The `local:` name space contains references to enterprise beans with local interfaces. There is only one `local:` name space in a server process. You can dump the `local:` name space by invoking the NameServer MBean associated with that server process.

Name space dump options

Name space dump options are specified in the MBean invocation as a parameter in character string format. The option descriptions follow.

-startAt *some/subcontext/in/the/tree*
Indicates the path from the name space root context to the top level context where the dump should begin. The utility recursively dumps subcontexts below this point. It defaults to an empty string, that is, the root context.

-report {short | long}

Option	Description
short	The default. Dumps the binding name and bound object type. This output is also provided by JNDI Context.list().
long	Dumps the binding name, bound object type, local object type, and string representation of the local object (that is, the IORs, string values, and other values that are printed).

-root {tree | host | legacy | cell | node | server | default}
Specify the root context of where the dump should start. The default value for `-root` is *cell*. This option is only valid for server name space dumps.

Option	Description
tree	Dump the tree starting at the tree root context.
host	Dump the tree starting at the server host root context (synonymous with "node").
legacy	Dump the tree starting at the legacy root context.
cell	Dump the tree starting at the cell root context. This is the default option.

Option	Description
node	Dump the tree starting at the node root context (synonymous with "host").
server	Dump the tree starting at the server root context. This is `-root` default.
default	Dump the tree starting at the initial context which JNDI returns by default for that server type.

-format {jndi | ins}

Specify the format to display name component as atomic strings or parsed according to INS rules (id.kind). This option is only valid for server name space dumps.

Option	Description
jndi	Display name components as atomic strings. This is `-format` default.
ins	Display name components parsed according to INS rules (id.kind).

NameServer MBean invocation

1. Enter the WebSphere Application Server scripting command prompt.

   Invoke a method on a NameServer MBean by using the WebSphere Application Server scripting tool. Enter the scripting command prompt by typing the following command:

Platform	Command
UNIX	wsadmin.sh
Windows NT	wsadmin

   Use the `-help` option for help on using the wsadmin command.

2. Select the NameServer MBean instance to invoke.

   Execute the following script commands to select the NameServer instance you want to invoke. For example,

   ```
   set mbean [$AdminControl completeObjectName WebSphere:*,type=NameServer,cell=
       cellName,node=nodeName,process=serverName]
   ```

   where *cellName*, *nodeName*, and *serverName* are the names of the cell, node, and server for the MBean you want to invoke. The specified server must be running before you can invoke a method on the MBean.

   You can see a list of all NameServer MBeans current running by issuing the following query:

   ```
   $AdminControl queryNames {*:*,type=NameServer}
   ```

3. Invoke the NameServer MBean.

   java: name space

   Dump a java: name space by invoking the dumpJavaNameSpace method on the NameServer MBean. Since each server application has its own `java:` name space, the application must be specified on the method invocation. An application is identified by the application name, module name, and component name. The method syntax follows:

```
$AdminControl invoke $mbean dumpJavaNameSpace {{appname}{modName}{compName}{opts}}
```

where *appName* is the application name, *modName* is the module name, and *compName* is the component name of the `java:` name space you want to dump. The value for *opts* is the list of name space dump options described earlier in this section. The list can be empty.

local: name space

Dump a `java:` name space by invoking the dumpLocalNameSpace method on the NameServer MBean. Because there is only one `local:` name space in a server process, you have to specify the name space dump options only.

```
$AdminControl invoke $mbean dumpLocalNameSpace {{opts}}
```

where *opts* is the list of name space dump options described earlier in this section. The list can be empty.

Server name space

Dump a server name space by invoking the dumpServerNameSpace method on an application server's NameServer MBean. This provides an alternative way to dump the name space on an application server, much like the dumpNameSpace command line utility.

```
$AdminControl invoke $mbean dumpServerNameSpace {{opts}}
```

where *opts* is the list of name space dump options described earlier in this section. The list can be empty.

Name space dump output

Name space dump output is sent to the console. It is also written to the file `DumpNameSpace.log`, in the server's log directory.

Related tasks

"Troubleshooting name space problems" on page 379

Related reference

"dumpNameSpace tool" on page 379

Example: Invoking the name space dump utility for `java:` and `local:` name spaces

It is often helpful to view the dump of a `java:` or `local:` name space to understand why a naming operation is failing. The NameServer MBean running in the application's server process can be invoked from the WebSphere Application Server scripting tool to generate a dump of these name spaces. Examples of NameServer MBean calls to generate dumps of java: and local: name spaces follow.

Dumping a `java:` name space

Assume you want to dump the `java:` name space of an application component running in server server1 on node node1 of the cell MyCell. The application name is AcctApp in module AcctApp.war, and the component name is Acct Servlet. The following script commands generate a long format dump of the application's `java:` name space of that application:

```
set mbean [$AdminControl completeObjectName WebSphere:*,type=NameServer,cell=MyCell,node=node1,process=server1]
$AdminControl invoke $mbean dumpJavaNameSpace {{DefaultApplication}{Increment.jar}{Increment}{-report long}}
```

Dumping a local: name space

Assume you want to dump the local: name space for the server server1 on node node1 of cell MyCell. The following script commands will generate a short format dump of that server's local name space:

```
set mbean [$AdminControl completeObjectName WebSphere:*type=NameServer,cell=MyCell,node=node1,process=server1]
$AdminControl invoke $mbean dumpLocalNameSpace {{-report short}}
```

Related tasks

"Troubleshooting name space problems" on page 379

Related reference

"dumpNameSpace tool" on page 379

Name space dump sample output

Name space dump output looks like the following example, which is the **SHORT** dump format:

```
Getting the initial context
Getting the starting context

================================================================================
Name Space Dump
   Provider URL: corbaloc:iiop:localhost:9810
   Context factory: com.ibm.websphere.naming.WsnInitialContextFactory
   Requested root context: cell
   Starting context: (top)=outpostNetwork
   Formatting rules: jndi
   Time of dump: Mon Sep 16 18:35:03 CDT 2002
================================================================================


================================================================================
Beginning of Name Space Dump
================================================================================

    1 (top)
    2 (top)/domain                                  javax.naming.Context
    2    Linked to context: outpostNetwork
    3 (top)/cells                                   javax.naming.Context
    4 (top)/clusters                                javax.naming.Context
    5 (top)/clusters/Cluster1                       javax.naming.Context
    6 (top)/cellname                                java.lang.String
    7 (top)/cell                                    javax.naming.Context
    7    Linked to context: outpostNetwork
    8 (top)/deploymentManager                       javax.naming.Context
    8    Linked to URL: corbaloc::outpost:9809/NameServiceServerRoot
    9 (top)/nodes                                   javax.naming.Context
   10 (top)/nodes/will2                             javax.naming.Context
   11 (top)/nodes/will2/persistent                  javax.naming.Context
   12 (top)/nodes/will2/persistent/SomeObject       SomeClass
   13 (top)/nodes/will2/nodename                    java.lang.String
   14 (top)/nodes/will2/domain                      javax.naming.Context
   14    Linked to context: outpostNetwork
   15 (top)/nodes/will2/cell                        javax.naming.Context
   15    Linked to context: outpostNetwork
   16 (top)/nodes/will2/servers                     javax.naming.Context
   17 (top)/nodes/will2/servers/server1             javax.naming.Context
   18 (top)/nodes/will2/servers/will2               javax.naming.Context
   19 (top)/nodes/will2/servers/member2             javax.naming.Context
   20 (top)/nodes/will2/node                        javax.naming.Context
   20    Linked to context: outpostNetwork/nodes/will2
   21 (top)/nodes/will2/nodeAgent                   javax.naming.Context
   22 (top)/nodes/outpost                           javax.naming.Context
   23 (top)/nodes/outpost/node                      javax.naming.Context
   23    Linked to context: outpostNetwork/nodes/outpost
```

```
24 (top)/nodes/outpost/nodeAgent                  javax.naming.Context
24    Linked to URL: corbaloc::outpost:2809/NameServiceServerRoot
25 (top)/nodes/outpost/persistent                 javax.naming.Context
26 (top)/nodes/outpost/nodename                   java.lang.String
27 (top)/nodes/outpost/domain                     javax.naming.Context
27    Linked to context: outpostNetwork
28 (top)/nodes/outpost/servers                    javax.naming.Context
29 (top)/nodes/outpost/servers/server1            javax.naming.Context
30 (top)/nodes/outpost/servers/server1/url        javax.naming.Context
31 (top)/nodes/outpost/servers/server1/url/CatalogDAOSQLURL
31                                                java.net.URL
32 (top)/nodes/outpost/servers/server1/mail       javax.naming.Context
33 (top)/nodes/outpost/servers/server1/mail/PlantsByWebSphere
33                                                javax.mail.Session
34 (top)/nodes/outpost/servers/server1/TransactionFactory
34                                                com.ibm.ejs.jts.jts.ControlSet$LocalFactory
35 (top)/nodes/outpost/servers/server1/servername java.lang.String
36 (top)/nodes/outpost/servers/server1/WSsamples  javax.naming.Context
37 (top)/nodes/outpost/servers/server1/WSsamples/TechSampDatasource
37                                                TechSamp
38 (top)/nodes/outpost/servers/server1/thisNode   javax.naming.Context
38    Linked to context: outpostNetwork/nodes/outpost
39 (top)/nodes/outpost/servers/server1/cell       javax.naming.Context
39    Linked to context: outpostNetwork
40 (top)/nodes/outpost/servers/server1/eis        javax.naming.Context
41 (top)/nodes/outpost/servers/server1/eis/DefaultDatasource_CMP
41                                                Default_CF
42 (top)/nodes/outpost/servers/server1/eis/WSsamples  javax.naming.Context
43 (top)/nodes/outpost/servers/server1/eis/WSsamples/TechSampDatasource_CMP
43                                                TechSamp_CF
44 (top)/nodes/outpost/servers/server1/eis/jdbc   javax.naming.Context
45 (top)/nodes/outpost/servers/server1/eis/jdbc/PlantsByWebSphereDataSource_CMP
45                                                PLANTSDB_CF
46 (top)/nodes/outpost/servers/server1/eis/jdbc/petstore
46                                                javax.naming.Context
47 (top)/nodes/outpost/servers/server1/eis/jdbc/petstore/PetStoreDB_CMP
47                                                PetStore_CF
48 (top)/nodes/outpost/servers/server1/eis/jdbc/CatalogDB_CMP
48                                                Catalog_CF
49 (top)/nodes/outpost/servers/server1/jta        javax.naming.Context
50 (top)/nodes/outpost/servers/server1/jta/usertransaction
50                                                java.lang.Object
51 (top)/nodes/outpost/servers/server1/DefaultDatasource
51                                                Default Datasource
52 (top)/nodes/outpost/servers/server1/jdbc       javax.naming.Context
53 (top)/nodes/outpost/servers/server1/jdbc/CatalogDB CatalogDB
54 (top)/nodes/outpost/servers/server1/jdbc/petstore  javax.naming.Context
55 (top)/nodes/outpost/servers/server1/jdbc/petstore/PetStoreDB
55                                                PetStoreDB
56 (top)/nodes/outpost/servers/server1/jdbc/PlantsByWebSphereDataSource
56                                                PLANTSDB
57 (top)/nodes/outpost/servers/outpost            javax.naming.Context
57    Linked to URL: corbaloc::outpost:2809/NameServiceServerRoot
58 (top)/nodes/outpost/servers/member1            javax.naming.Context
59 (top)/nodes/outpost/cell                       javax.naming.Context
59    Linked to context: outpostNetwork
60 (top)/nodes/outpostManager                     javax.naming.Context
61 (top)/nodes/outpostManager/domain              javax.naming.Context
61    Linked to context: outpostNetwork
62 (top)/nodes/outpostManager/cell                javax.naming.Context
62    Linked to context: outpostNetwork
63 (top)/nodes/outpostManager/servers             javax.naming.Context
64 (top)/nodes/outpostManager/servers/dmgr        javax.naming.Context
64    Linked to URL: corbaloc::outpost:9809/NameServiceServerRoot
65 (top)/nodes/outpostManager/node                javax.naming.Context
65    Linked to context: outpostNetwork/nodes/outpostManager
66 (top)/nodes/outpostManager/nodename            java.lang.String
```

```
   67 (top)/persistent                              javax.naming.Context
   68 (top)/persistent/cell                         javax.naming.Context
   68    Linked to context: outpostNetwork
   69 (top)/legacyRoot                              javax.naming.Context
   69    Linked to context: outpostNetwork/persistent
   70 (top)/persistent/AnotherObject                AnotherClass

================================================================================
End of Name Space Dump
================================================================================
```

Related tasks

"Troubleshooting name space problems" on page 379

Related reference

"dumpNameSpace tool" on page 379

Object Request Broker

Object request broker component troubleshooting tips

Enabling tracing for the Object Request Broker component

The object request broker (ORB) service is one of the WebSphere Application Server run time services. Tracing messages that are sent and received by the ORB is a useful starting point for troubleshooting the ORB service. You can selectively enable or disable tracing of ORB messages for each server in a WebSphere Application Server installation, and for each application client.

For instructions on how to set trace controls so that tracing occurs for the ORB subcomponent, see Setting trace controls for IBM service.

Log files and messages associated with Object Request Broker

For a summary of how messages get routed in WebSphere Application Server, see Message routing.

Java packages containing the Object Request Broker service

The ORB service resides in the following Java packages:
- com.ibm.com.CORBA.*
- com.ibm.rmi.*
- com.ibm.ws.orb.*
- com.ibm.ws.orbimpl.*
- com.ibm.ws390.orb.*
- org.omg.CORBA.*
- javax.rmi.CORBA.*

JAR files that contain the previously mentioned packages include:
- *java_install_dir*/jre/lib/ext/ibmorb.jar
- *java_install_dir*/jre/lib/ext/iwsorbutil.jar
- *install_dir*/lib/iwsorb.jar
- *install_dir*/lib/runtimews390.jar
- *install_dir*/lib/bootstrap.jar

Tools used with Object Request Broker

The tools used to compile Java remote interfaces to generate language bindings used by the ORB at runtime reside in the following Java packages:
- com.ibm.tools.rmic.*
- com.ibm.idl.*

The JAR file that contains the packages is *install_dir*/java/lib/ibmtools.jar.

Object Request Broker properties

The ORB service requires a number of ORB properties for correct operation. It is not necessary for most users to modify these properties, and only the system administrator should modify them when required. Consult IBM Support personnel for assistance. The properties reside in the orb.properties file, located at *install_dir*/java/jre/lib/orb.properties.

If none of these steps fixes your problem, check to see if the problem has been identified and documented by looking at the available online support (hints and tips, technotes, and fixes).

For current information available from IBM Support on known problems and their resolution, see the IBM Support page.

IBM Support has documents that can save you time gathering information needed to resolve this problem. Before opening a PMR, see the IBM Support page.

Related tasks

Chapter 6, "Troubleshooting by task," on page 233

Related reference

Troubleshooting installation problems

Transactions

Troubleshooting transactions

Use this overview task to help resolve a problem that you think is related to the Transaction service.

To identify and resolve transaction-related problems, you can use the standard WebSphere Application Server RAS facilities. If you encounter a problem that you think might be related to transactions, complete the following steps:

1. Check for transaction messages in the administrative console. The Transaction service produces diagnostic messages prefixed by "WTRN". The error message indicates the nature of the problem and provides some detail. The associated message information provides an explanation and any user actions to resolve the problem.
2. Check for Transaction messages in the activityerror log. Log messages produced by the Transaction service are accompanied by Log Analyzer descriptions.
3. Check for more messages in other potential message output repositories. For more information about a problem, check the standard output file configured by your administrator. This will contain more error messages and other detailed information about the problem.
4. Check for messages related to the application server's transaction log directory when the problem occurred.

Note: If you changed the transaction log directory and a problem caused the application server to fail (with in-flight transactions) before the server was restarted properly, the server will next start with the new log directory and be unable to automatically resolve in-flight transactions that were recorded in the old log directory. To resolve this, you can copy the transaction logs to the new directory then stop and restart the application server.

5. Check the RRS logs for any transaction activity involving RRS-compliant resources. IBM WebSphere Application Server for z/OS is capable of supporting both XA and RRS resource managers, and of coordinating a mix of RRSTransactional resource managers and XA capable resource managers under the same global transaction. If your installation uses XA resource managers, RRS resource managers, or a mixture of both, you can use the administrative console to view transaction logs that contain information about all transactions. You can find additional information about RRS transactions by using the RRS panels.
6. Check the hints and tips for troubleshooting transactions.

   The "Tips for troubleshooting transactions" is an ongoing collection of troubleshooting tips, based on test and user experience. If you have suggestions for other tips, please let the IBM WebSphere writing team know.

Tips for troubleshooting transactions

This topic provides a set of tips to help you troubleshoot problems with the WebSphere transaction service.

- Peer recovery fails to acquire a lock
- "XAER_NOTA exception logged after server fails" on page 391

For messaging problems specific to WebSphere Application Server nodes, see the information center and the Application Servers support web site; for example: Tips for troubleshooting WebSphere messaging [version 5].

Peer recovery fails to acquire a lock

If peer recovery of a transaction fails to acquire a file lock that is needed to perform recovery processing, you should see the following messages:

```
[10/26/04 8:41:38:887 CDT] 00000029 CoordinationL A   WTRN0100_GENERIC_ERROR
[10/26/04 8:41:39:100 CDT] 00000029 RecoveryHandl A   WTRN0100E: An attempt to
acquire a file lock need to perform recovery processing failed. Either the target
server is active or the recovery log configuration is incorrect
....
[10/26/04 8:42:34:921 CDT] 00000027 HAGroupImpl   I   HMGR0130I: The local member
of group GN_PS=fwsitkaCell01\fwwsaix1Node01\GriffinServer3,IBM_hc=GriffinCluster,type
=WAS_TRANSACTIONS has indicated that is it not alive. The JVM will be terminated.
[10/26/04 8:42:34:927 CDT] 00000027 SystemOut     O Panic:component requested panic
from isAlive
```

To troubleshoot the cause of failure to acquire the file lock, check the following factors:

- If you have "Enable high availability for persistent services" enabled on the server cluster and are using a NAS devise for the transaction logs, check that the DFS level on your machine is at a correct level for the NAS DFS level. If the two levels are not correct, the transaction logs cannot be accessed.
- If you are running as non-root, check that the id numbers of the non-root user and group match on all machines involved with peer recovery.
- If you have a policy defined for transaction, review the policy to ensure that you are giving control to the correct servers (perhaps you need to add to or reorder the preferred server list).

XAER_NOTA exception logged after server fails

If an application server fails, and the end transaction record is not forced to disk immediately, you may or may not recover a transaction.

WebSphere does not force the end record to the log, so it is up to the operating system/network file system to decide when to write to the disk. The record would be forced if the server was shutdown cleanly. The transaction service is designed to cope with the case of the end record never being written to disk - when it gets an XAER_NOTA returned from the databases.

```
[date time] 00000057 WSRdbXaResour E   DSRA0302E:  XAException occurred.
                                        Error code is: XAER_NOTA (-4).  Exception is: XAER_NOTA
```

If there is a transaction without an end record left in the transaction log, the transaction service tries to check with the database. If the transaction has completed, the database indicates that there is nothing to complete (XAER_NOTA). This is normal behavior, and not an error.

Transaction service exceptions

This topic lists the exceptions that can be thrown by the WebSphere Application Server transaction service. The exceptions are listed in the following groups:

- Standard exceptions
- Heuristic exceptions

If the EJB container catches a system exception from the business method of an enterprise bean, and the method is running within a container-managed transaction, the container rolls back the transaction before passing the exception on to the client. For more information about how the container handles the exceptions thrown by the business methods for beans with container-managed transaction demarcation, see the section *Exception handling* in the Enterprise JavaBeans 2.0 specification. That section specifies the container's action as a function of the condition under which the business method executes and the exception thrown by the business method. It also illustrates the exception that the client receives and how the client can recover from the exception.

Standard exceptions

The standard exceptions such as TransactionRequiredException, TransactionRolledbackException, and InvalidTransactionException are defined in the Java Transaction API (JTA) 1.0.1 Specification.

InvalidTransactionException
: This exception indicates that the request carried an invalid transaction context.

TransactionRequiredException exception
: This exception indicates that a request carried a null transaction context, but the target object requires an active transaction.

TransactionRolledbackException exception
: This exception indicates that the transaction associated with processing of the request has been rolled back, or marked for roll back. Thus the requested operation either could not be performed or was not performed because further computation on behalf of the transaction would be fruitless.

Heuristic exceptions

A heuristic decision is a unilateral decision made by one or more participants in a transaction to commit or rollback updates without first obtaining the consensus outcome determined by the Transaction Service. Heuristic decisions are an issue only after the participant has been prepared and the second phase of commit processing is underway. Heuristic decisions are normally made only in unusual circumstances, such as repeated failures by the transaction manager to communicate with a resource manage during two-phase commit. If a heuristic decision is taken, there is a risk that the decision differs from the consensus outcome, resulting in a loss of data integrity.

The following list provides a summary of the heuristic exceptions. For more detail, see the Java Transaction API (JTA) 1.0.1 Specification.

HeuristicRollback exception
: This exception is raised on the commit operation to report that a heuristic decision was made and that all relevant updates have been rolled back.

HeuristicMixed exception
: This exception is raised on the commit operation to report that a heuristic decision was made and that some relevant updates have been committed and others have been rolled back.

Related tasks

Troubleshooting transactions
Use this overview task to help resolve a problem that you think is related to the Transaction service.

Related information

Enterprise JavaBeans 2.0 specification

Java Transaction API (JTA) 1.0.1 Specification

Learn about WebSphere programming extensions

Use this section as a starting point to investigate the WebSphere programming model extensions for enhancing your application development and deployment.

See Learn about WebSphere applications: Overview and new features for an introduction to each WebSphere extension.

ActivitySessions	How do I?...	Overview	Samples	
Application profiling	How do I?...	Overview	Samples	
Asynchronous beans	How do I?...	Overview	Samples	
Dynamic caching	How do I?...	Overview		
Dynamic query	How do I?...	Overview	Samples	
Internationalization	How do I?...	Overview	Samples	
Object pools	How do I?...	Overview		
Scheduler	How do I?...	Overview	Samples	
Startup beans	How do I?...	Overview		
Work areas	How do I?...	Overview		

Related concepts

Learn about WebSphere applications: Overview and new features
Use the **Learn about WebSphere applications** section as a starting point to study technologies used in and by applications deployed on the application server.

Chapter 7, “Learn about WebSphere applications,” on page 235
Use this section as a starting point to investigate the technologies used in and by applications that you deploy on the application server.

ActivitySessions

Application profiling

Dynamic cache

Troubleshooting the dynamic cache service

Complete the steps below to resolve problems that you think are related to the dynamic cache service.

1. On the z/OS platform, check the job logs for the controller and servant. Messages that are prefaced with *DYNA* result from dynamic cache service operations.
   a. Find any messages prefaced with *DYNA* in the log you are viewing, and write down the message IDs. A sample message having the message ID *DYNA0030E* follows:

      ```
      DYNA0030E: "property" element is missing required attribute "name".
      ```

   b. Find the message for each message ID in the information center for WebSphere Application Server. In the information center navigation tree, click ***product_name*** **> Reference > Troubleshooter > Messages > DYNA** to view dynamic cache service messages.
   c. Read the message **Explanation** and **User Action** statements. A search for the message ID `DYNA0030E` displays a page having the following message:

```
DYNA0030E: "property" element is missing required attribute "name".
  Explanation: A required attribute was missing in the cache configuration.
  User Action: Add the required attribute to your cache configuration file.
```

      This explanation and user action suggests that you can fix the problem by adding or correcting a required attribute in the cache configuration file.
   d. Try the solutions stated under **User Action** in the *DYNA* messages.
2. Use the cache monitor to determine whether the dynamic cache service is functioning as expected. The cache monitor is an installable Web application that displays simple cache statistics, cache entries, and cache policy information.
3. If you have completed the preceding steps and still cannot resolve the problem, contact your IBM software support representative. The IBM representative might ask you to complete a diagnostic trace. To enable tracing in the administrative console, click **Troubleshooting > Logs and trace >** ***server_name*** **> Diagnostic trace** and specify **Enable trace with the following specification**. The IBM representative can tell you what trace specification to enter. Note that dynamic cache trace files can become large in a short period of time; you can limit the size of the trace file by starting the trace, immediately recreating the problem, and immediately stopping the trace.

   For current information available from IBM Support on known problems and their resolution, see the IBM Support page.

For technical support on dynamic cache service, see the IBM Support page.

Related tasks

Task overview: Using the dynamic cache service to improve performance
Use the dynamic cache to improve application performance by caching the output of servlets, commands, and JavaServer Pages (JSP) files.

Related information

Cache monitor
Cache monitor is an installable Web application that provides a real-time view of the current state of dynamic cache. You use it to help verify that dynamic cache is operating as expected. The only way to manipulate the data in the cache is by using the cache monitor. It provides a GUI interface to manually change data.

Displaying cache information
Use this task to monitor the activity of the dynamic cache service.

Troubleshooting tips for the dynamic cache service:

The dynamic cache service works within an application server Java virtual machine (JVM), intercepting calls to cacheable objects. This article describes some common runtime and configuration problems and remedies.

Servlets are not cached

Recommended response Enable servlet caching. On the Web container page of the administrative console, select the **Enable servlet caching** check box.

Cache entries are not written to disk

Explanation Cache entries are written to disk when the cache is full and new entries are added to the memory cache. Cache entries also are written to disk when **Flush to disk** is enabled in the administrative console and the server is stopped.

Recommended response Verify that **Disk offload** is enabled on the Dynamic cache service settings page of the administrative console. Also verify that cache entries written to disk are serializable and do not have the `PersistToDisk` configuration set to `false`.

Some servlets are not replicated or written to disk

Recommended response Ensure that the attributes and response are serializable. If you do not want to store the attributes, use the following property in your cache policy:

```
<property name="save-attributes">false</property>
```

Dynamic cache service does not cache fragments on the Edge

Recommended response Set the `EdgeCacheable` property to `true` in the cache policy for those entries that are to be cached on the Edge.

```
<property name="EdgeCacheable">true</property>
```

Dynamic cache invalidations are not sent to the IBM HTTP Server (IHS) plug-in

Explanation The `DynaCacheEsi.ear` file is required to send invalidations to external caches.

Recommended response Install `DynaCacheEsi.ear` using the administrative console.

Cache entries are evicted often

Problem The cache is full and new entries are added to the cache.

Explanation Cache entries are evicted when the cache is full and new entries are added to the cache. A least recently used (LRU) eviction mechanism removes the least recently used entry to make space for the new entries.

Recommended response Enable **Disk offload** on the Dynamic cache service settings page of the administrative console so the entries are written to disk. You can also increase the cache size to accommodate more entries in the cache.

Cache entries in disk with timeout set to 0 expire after one day

Explanation The maximum lifetime of an entry in disk cache is 24 hours. A timeout of `0` in the cache policy configures these entries to stay in disk cache for one whole day, unless they are evicted earlier.

Recommended response Set the timeout for the cache policy to a number less than `0`.

I cannot monitor cache entries on the Edge

Explanation Use the Cache monitor for monitoring contents in memory cache, disk cache and external caches (Edge cache). For the ESI processor's cache to be visible in the cache monitor, the `DynaCacheEsi.ear` application must be installed and the `esiInvalidationMonitor` property must be set to `true` in the `plugin-cfg.xml` file. Monitoring the ESI processor's cache entries on z/OS is not supported.

Recommended response See Displaying cache information and set the `esiInvalidationMonitor` property to `true` in the `plugin-cfg.xml` file.

I want to tune cache for my environment

Recommended response Use the Tivoli Performance viewer to study the caching behavior for your applications. Also consider performing the following actions:

- Increase the priority of cache entries that are expensive to regenerate.
- Modify timeout of entries so that they stay in memory as long as they are valid.
- Enable disk offload to store LRU evicted entries.
- Increase the cache size.

Cleaning the disk cache files after installing the fix pack or a new release if you use the disk cache function

Symptom
If the server is configured to use the disk cache, you must delete the disk cache files because the disk cache files are not compatible to the previous version.

Problem
Failure to remove the old disk cache files results in a ClassCastException error in the systemerr.log file when you access the cache from the disk.

Recommended response
To delete the disk cache, perform the following steps:

1. Note your disk offload location. If you do not know the disk cache offload location, perform the following steps:
   a. Click **Servers > Application servers >***server_name* **> Container services > Dynamic cache service** in the administrative console navigation tree.
   b. The location is specified in the Disk offload field. If the location is not specified, the default directory `WebSphere/AppServer/profiles/your_profile_name/temp/your_node/server_name/_dynacache` is used.
2. Make sure that the you stop the server and delete all the files under the offload location.
3. If you use the Network Deployment product, delete the disk cache files for each server.

Setting the flush attribute to true on every <jsp:include> tag in the cacheable JavaServer Pages file

Symptom
When you obtain the JavaServer Pages (JSP) file from the dynamic cache, a part of the page is not displayed.

Problem
The flush attribute is set to false on the `<jsp: include>` tag in the JSP file.

Description
When the cacheable JSP file includes another JSP file and if the flush attribute is set to false on the `<jsp: include>` tag, any data written to the parent output stream before the`<jsp: include>` tag are not cached.

Recommended response
Set `flush=true` on every `<jsp: include>` tag in the cacheable JSP file.

Related tasks

Task overview: Using the dynamic cache service to improve performance
Use the dynamic cache to improve application performance by caching the output of servlets, commands, and JavaServer Pages (JSP) files.

Related information

"Troubleshooting the dynamic cache service" on page 393

Displaying cache information
Use this task to monitor the activity of the dynamic cache service.

Internationalization

Notices

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only IBM's product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any of IBM's intellectual property rights may be used instead of the IBM product, program, or service. Evaluation and verification of operation in conjunction with other products, except those expressly designated by IBM, is the user's responsibility.

IBM may have patents or pending patent applications covering subject matter 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 Intellectual Property & Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
USA

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:

IBM Corporation
Mail Station P300
2455 South Road
Poughkeepsie, NY 12601-5400
USA
Attention: Information Requests

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

Trademarks and service marks

For trademark attribution, visit the IBM Terms of Use Web site (http://www.ibm.com/legal/us/).