WebSphere software

IBM

Redbooks Paper

Thu-Giang Pham

WebSphere Application Server V6: JCA Connection Problem Determination

IBM® WebSphere® Application Server V6 provides implementation and support for the new J2EE™ Connector Architecture (JCA) specification V1.5 as part of the J2EE 1.4 platform. This paper discusses the most common problem areas that are associated with JCA connections in WebSphere Application Server V6. Issues that are caused by JCA components or JCA connection configuration errors can appear as one or more of the following initial symptoms:

- A JDBC™ call returns incorrect data to the application
- An application cannot connect to or access a database or EIS
- WebSphere error messages with the prefixes `DSRA`, `WSCL`, `J2CA`, `WTRN`, `CONM`, or `SQLException` or with database error codes.

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

Introduction

The JCA specification provides a standard mechanism that allows modern J2EE applications to connect and use heterogeneous resources from various existing enterprise information systems (EIS) as well as modern relational database systems. Based on the JCA, WebSphere provides client applications all system services regarding connection, transaction, and security management on behalf of the resource managers.

JCA involves four main components:

- Application server
- Application component
- Resource adapter
- EIS

It also specifies a requirement for packaging and deployment facilities for a resource adapter to plug into an application server. Figure 1 illustrates the contracts or relationships between these four components.

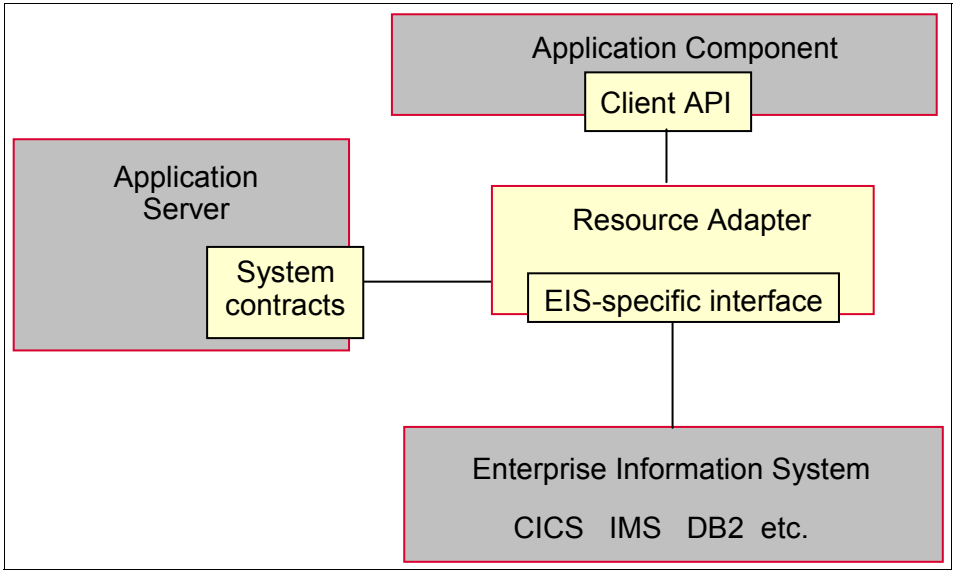

Figure 1 JCA overview

Based on the JCA specification, an EIS vendor can develop a standard resource adapter (RA) for its EIS to plug into any application sever that supports JCA. A resource adapter runs within the address space of an application server, while the EIS itself runs in a separate address space. For example, a DB2® database

(EIS) and WebSphere Application Server each run in a separate machine. An application component is able to access the EIS through the resource adapter.

The relationships between the four major components can be described as follows:

- The contracts between the application component and the resource adapter are provided through some form of client API. The client API can either be specific to a particular type of resource adapter, for example JDBC for relational data base, or a standard common client interface (CCI). The JCA recommends, but does not require, that a resource adapter implement the CCI. WebSphere Application Server V6 provides a relational resource adapter (RRA) that has an implementation for both the CCI and the traditional JDBC interfaces.
- The resource adapter and application server implement system contracts to provide the common mechanisms for connection, transaction, and security management.
- The contracts between the resource adapter and the EIS are specific to each underlying EIS. Thus, JCA does not impose any requirement on this proprietary relationship. For example, an RRA for a relational database accesses the resources through a JDBC driver that is supported by the database, or a resource adapter for an SAP R/3 systems accesses the business objects and functions through SAP's Business Application Programming Interface and Remote Function Call (BAPI/RFC).

JCA technical overview

The JCA allows applications to access the EIS in both managed and non-managed environments.

In a managed environment, the application components run within the address space of the application server. For example, an application component can be a servlet, or it can be an EJB™ that runs within a Web container or an EJB container of the WebSphere Application Server. The servlet or the EJB accesses the EIS through a resource adapter that is plugged into WebSphere Application Server. Figure 2 on page 4 shows this architecture.

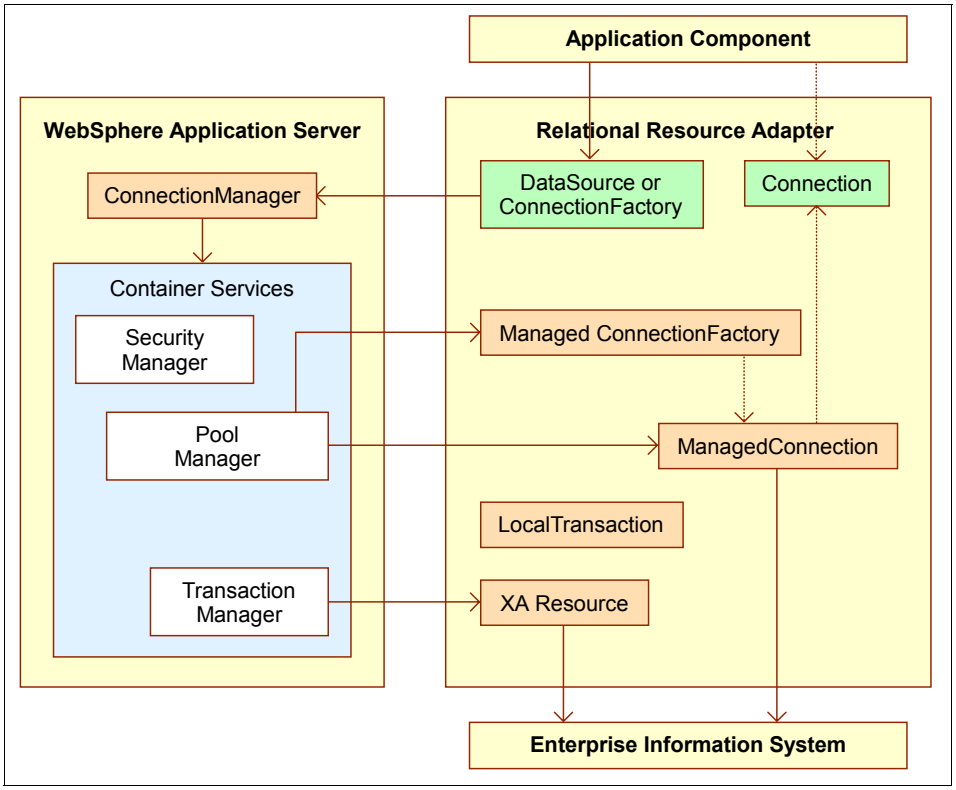

Figure 2 Managed application scenario

The application component acquires a ConnectionFactory through JNDI. The ConnectionFactory is used to get a Connection that provides access to the EIS. The ConnectionFactory and Connection interfaces are part of the CCI (from Java™ package javax.resource.cci) that are implemented by the resource adapter. In addition to the CCI, the WebSphere RRA provides implementation for the existing JDBC interfaces javax.sql.DataSource and javax.sql.Connection to allow existing applications to fit in the new JCA structure without changing application client code.

The remaining interfaces that are discussed in this section are specified in javax.resource.spi with the implementation that is provided by either WebSphere Application Server or the resource adapter.

The ConnectionFactory delegates the connection allocation request to the ConnectionManager (implemented by WebSphere). Through the ConnectionManager, WebSphere can provide applications with many system services such as security, transaction, pooling etc. The ConnectionManager

looks up the connection pool maintained by WebSphere to see if a free ManagedConnection (implemented by the resource adapter) is available. If there is none, the ManagedConnectionFactory (implemented by the resource adapter) will be asked to create a new ManagedConnection to add to the pool. In either case, WebSphere Application Server receives a ManagedConnection to create a Connection as its application-level handle to return to client. The returned Connection is just a logical handle that represents the physical connection within the underlying EIS.

The resource adapter also provides implementation for the XAResource interface to support transactions that involve multiple resource managers and the LocalTransaction interface to support transactions internal to the resource manager.

In a non-managed environment, application clients run outside the address space of the application server. The application programming model for the client code is the same as in a managed scenario. However, applications use the resource adapter library directly. The ConnectionFactory still delegates the task of allocating a Connection to the ConnectionManager, but it involves the resource adapter's default implementation for the ConnectionManager instead of the application server's.

This paper addresses problems that are experienced during connection to enterprise information systems or databases using JCA.

Users with the following initial symptoms might be experiencing a JCA-related problem:

- Symptom: A JDBC call returns incorrect data
- Symptom: Failure to connect to a new data source
- Symptom: Failure to connect to an existing data source
- Symptom: Failure to access a resource through JDBC
- Symptom: Failure to access a non-relational resource

Such symptoms can be observed in the WebSphere Application Server JVM™ logs in error messages with any of the following prefixes: `WTRN`, `J2CA`, `WSCL`, or `DSRA`.

This paper helps you find the cause of the problem if the cause is in one of the following areas:

- Application
- Configuration and tuning
- WebSphere components and resource adapters that implement the JCA system contracts

Work the problem

If you are experiencing problems during the execution of an application or task that impact the operation, the initial symptoms you encounter can help you determine what software components might be involved. An initial symptom is just an event that allows you to immediately identify a problem in a general and broad sense without getting into much detailed analysis. Thus, our problem determination process begins with a list of initial symptoms that can identify a JCA connection problem. The initial symptoms serve as the starting point for the subsequent analysis.

In the analysis of each symptom, this problem determination (PD) process guides you through the following tasks:

- Collecting diagnostic files
- Looking for specific warnings, errors, or other status information that is produced by the system
- Evaluating the error messages to see if they are related to your problem
- Assessing and applying a resolution as appropriate. Finding the root cause of the problem is the goal of the problem determination process, but suggesting a solution is not. However, during the analysis, we occasionally suggest solutions when they naturally follow from the analysis.

The problem determination process puts emphasis on actions or tasks. Performing a task produces a result, causes an event to happen, or gives a symptom for you to see what was happening. The outcome, event, or symptom in turn leads to another action or task. This process repeats until a root cause for the problem is found.

In the flow diagrams in this paper, a rectangular box with rounded corners represents an action that you perform and an oval-shaped element represents the detailed analysis of a problem area. (This detailed analysis is simply a series of actions that you need to perform for the problem area.) The transition between the elements in the flow diagram is qualified and labeled with the symptom (or outcome or event) that you receive. The diamond-shaped element is a branch that leads to more than one possible transition. When you have identified the problem area, you can proceed to the corresponding subsection of “Analyzing problem areas” on page 17 to continue the process.

Figure 3 shows the initial symptoms that you might experience when you have a JCA connection problem.

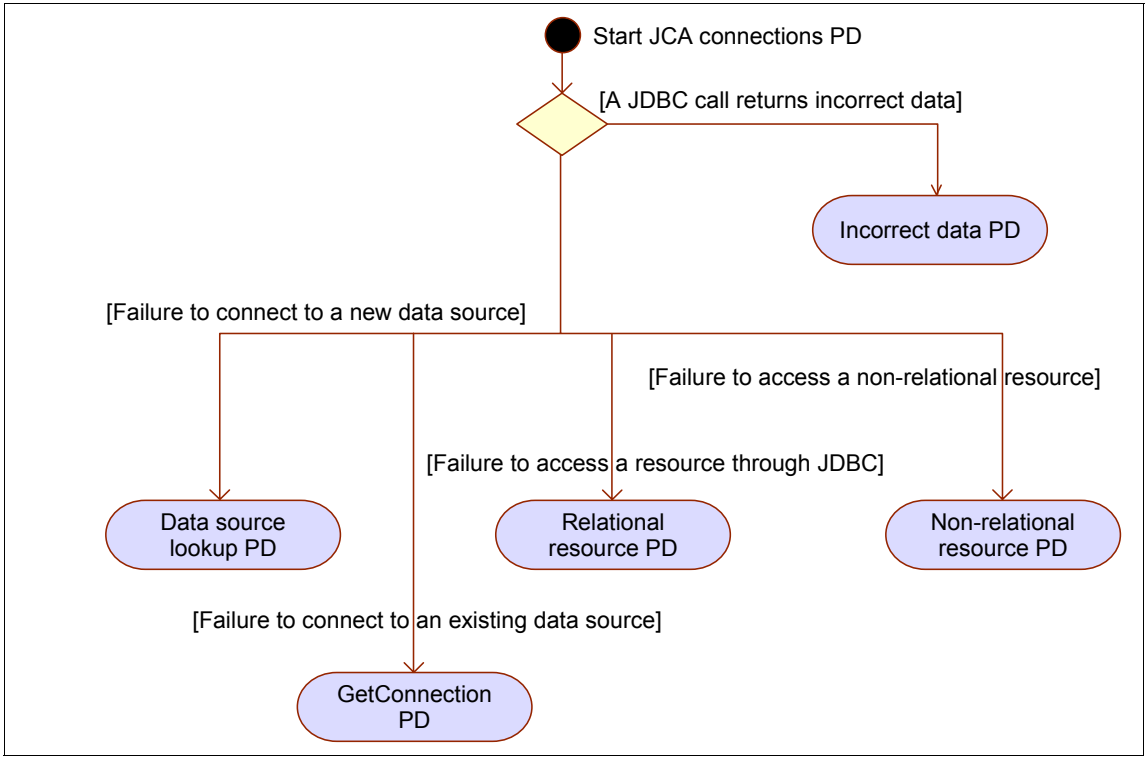

Figure 3 JCA connections - initial symptom analysis

Symptom: A JDBC call returns incorrect data

With this symptom, your application is running normally, but you receive incorrect data. For example, your database maintains a list of product inventory, but some product quantities or prices that are reported from your application are not in line with your expectation.

Rule out a database or JDBC driver problem

If you suspect data is being incorrectly returned to the application, try running the same query using the database's native client tools (such as Oracle SQL*Plus, Sybase isql, Informix® dbaccess, DB2 Command Line Processor, or SQL Server Query Analyzer). Alternatively, try to run the same query in a stand-alone JDBC program using the JDBC driver directly. If the data returned from the client tool is the same as the data that your application returned when running in WebSphere

Application Server, then it is likely that the database has a problem. In this case, you need to engage the database vendor for further help.

If the data in the database is correct, try to write a stand-alone JDBC program that runs the same query to see if the same symptom occurs. If the data returned from the stand-alone JDBC program is the same as the data that your application returned, then problem is with the JDBC driver. In this case, you need to contact the JDBC vendor for further help.

Tips: To create a stand-alone JDBC program to run outside of WebSphere Application Server, you can:

- Execute ***<WAS_install_root>*/bin/setupCmdLine.bat** on Windows® or ***<WAS_install_root>*/bin/setupCmdLine.sh** on UNIX® to set up the JAVA_HOME environment variable.
- Set the CLASSPATH environment variable to include *<WAS_install_root>*/lib/j2ee.jar and the jar file(s) for the JDBC driver. (See the examples for the JAR file names that are associated with the JDBC drivers included in this paper).
- If you are using a type 2 JDBC driver, then you might also need to set other required environment variables. For example, if you are using Oracle OCI driver, you must set ORACLE_HOME and LIBPATH.
- Your JDBC program must use the same data source implementation class as the one used by WebSphere connection manager.

The following examples show how to connect to the database for various platforms.

Example 1 Oracle JDBC Thin

```
import java.sql.*;
import javax.sql.*;
import oracle.jdbc.driver.*;
import oracle.jdbc.pool.OracleConnectionPoolDataSource;
.....
OracleConnectionPoolDataSource ds = new OracleConnectionPoolDataSource();
ds.setUser("scott");
ds.setPassword("tiger");
ds.setDriverType("thin");/
ds.setURL("jdbc:oracle:thin:@localhost:1521:swanlake");
Connection con = ods.getConnection();
```

Example 2 Oracle OCI

```
import java.sql.*;
import javax.sql.*;
import oracle.jdbc.pool.OracleConnectionPoolDataSource;
...
OracleConnectionPoolDataSource ds = new OracleConnectionPoolDataSource();
ds.setUser("scott");
ds.setPassword("tiger");
ds.setDriverType("oci");
ds.setURL("jdbc:oracle:oci8@tnsnames");
// tnsnames is the connect alias defined in your Oracle's network
// client file $ORACLE_HOME/network/admin/tnsnames.ora
Connection con = ds.getConnection();
```

Example 3 DB2 Legacy CLI

```
import java.sql.*;
import javax.sql.*;
import COM.ibm.db2.jdbc.*;
...
DB2ConnectionPoolDataSource ds = new DB2ConnectionPoolDataSource();
ds.setDatabaseName("sample");
ds.setUser("db2admin");
ds.setPassword("db2admin");
PooledConnection pCon = ds.getPooledConnection();
Connection con = pCon.getConnection();
```

Example 4 DB2 Universal JDBC (JCC)

```
import java.sql.*;
import javax.sql.*;
import com.ibm.db2.jcc.*;
...
DB2ConnectionPoolDataSource ds = new DB2ConnectionPoolDataSource();
ds.setUser("db2admin");
ds.setPassword("db2admin");
ds.setDatabaseName("sample");
ds.setServerName("9.27.40.128");
ds.setPortNumber(50000);
ds.setDriverType(4);
PooledConnection pCon = ds.getPooledConnection();
Connection con = pCon.getConnection();
```

Example 5 Informix JDBC

```
import java.sql.*;
import javax.sql.*;
import com.informix.jdbcx.*;
...
IfxConnectionPoolDataSource ds = new IfxConnectionPoolDataSource();
ds.setIfxIFXHOST("perfdobo.rchland.ibm.com");
ds.setPortNumber(1526);
ds.setUser("tgpham");
ds.setPassword("snOw1ce");
ds.setServerName("ol_perfdobo");
ds.setDatabaseName("tgpham");
PooledConnection pCon = ds.getPooledConnection();
Connection con = pCon.getConnection();
```

Example 6 Sybase jConnect

```
import java.sql.*;
import javax.sql.*;
import com.sybase.jdbc2.jdbc.*;
...
SybConnectionPoolDataSource ds = new SybConnectionPoolDataSource();
ds.setDatabaseName("test");
ds.setUser("test01");
ds.setPassword("test01");
ds.setServerName("mySybaseServerHostname");
ds.setPortNumber(4100);
PooledConnection pCon = ds.getPooledConnection();
Connection con = pCon.getConnection();
```

Example 7 SQL Server

```
import java.sql.*;
import javax.sql.*;
// For WebSphere embedded Connect JDBC
import com.ibm.websphere.jdbcx.sqlserver.*;
// For DataDirect Connect JDBC
// import com.ddtek.jdbcx.sqlserver.*;
// For SequeLink JDBC
// import com.ddtek.jdbcx.*;
SQLServerDataSource ds = new SQLServerDataSource();
ds.setDatabaseName("sample");
ds.setServerName("localhost");
ds.setPortNumber(1433);
ds.setUser("dbuser1");
ds.setPassword("dbpwd1");
ds.setSelectMethod("cursor");
Connection con = ds.getConnection();
```

What to look for

If you have determined that the problem is not caused by the database or JDBC driver, you can continue the analysis by looking for the situation in which the error occurs. The best way to do this is to perform a number of runs through the application, both from a fresh restart and repeated runs within the same application session.

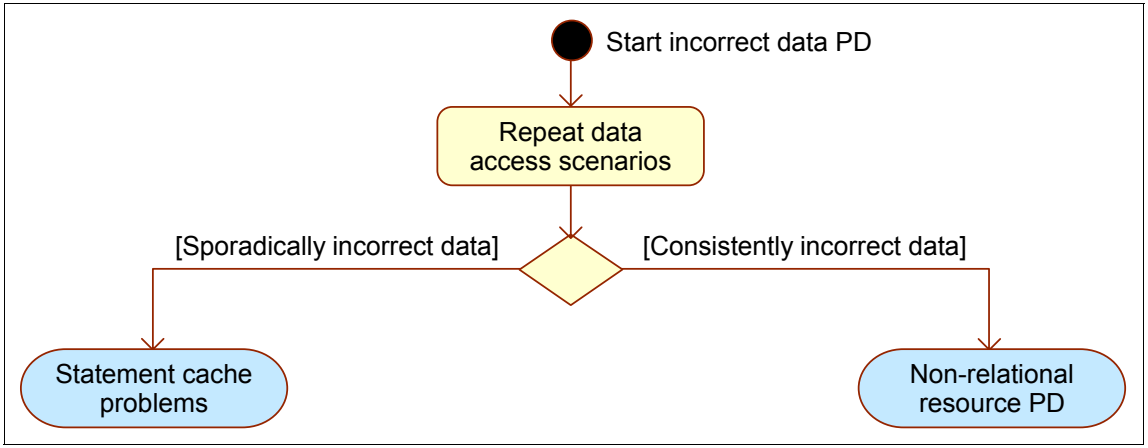

Figure 4 Incorrect data problem determination

As shown in Figure 4, you might be in one of the following two situations:

- Sporadically incorrect data, which means that the data that is returned to you is correct sometimes but not always.

  If your application uses a prepared statement or callable statement, you can experience a problem with WebSphere's statement cache.

  A prepared statement is a precompiled SQL statement that is stored in a prepared statement object for parameterized queries. This object is used to run the given SQL statement efficiently multiple times. A callable statement is used to invoke stored procedures.

  In general, the more prepared statements and callable statements that your application has, the larger the cache should be. Be aware, however, that specifying a larger statement cache size than needed wastes application memory and does not improve performance.

  Determine the value for your cache size by adding the number of uniquely prepared statements and callable statements (as determined by the SQL string, concurrency, and the scroll type) for each application that uses this data source on a particular server. This value is the maximum number of possible prepared statements and callable statements that are cached on a given connection over the life of the server.

You can try to disable the statement cache by setting `Statement cache size` to 0. This setting can be found in the administrative console by selecting **Resources** → **JDBC Providers** → <*JDBC_provider*> → **Data sources** → <*data_source*> → **WebSphere Application Server connection properties**.

You should call IBM Technical Support to help you with this scenario.

- Consistently incorrect data is when the data that is returned to you is always incorrect. In this case, it is highly possible that your application is connected to the wrong database or resource manager. In this case, you might have a problem with your environment or configuration. Proceed to “Configuration problems” on page 18 for further analysis.

Symptom: Failure to connect to a new data source

When you start a new working session in your application, the application first needs to look for a data source that provides the data. Any error at this early stage can be broadly identified as a failure on new data source.

Data to collect

All relevant error messages and exceptions needed for our analysis are available in SystemOut.log and SystemErr.log.

What to look for

To proceed for further analysis, you need to run a tool called TestConnection service. WebSphere Application Server provides a test connection service for testing connections to the data sources that you configure for database access. This test connection service can be activated in three different ways:

- Through the administrative console
- Using the wsadmin tool
- With a Java stand-alone program

For more information, see *Test connection service* in the WebSphere Information Center at:

`http://publib.boulder.ibm.com/infocenter/wasinfo/v6r0/topic/com.ibm.websphere.base.doc/info/aes/ae/cdat_testcon.html`

Depending on the messages or exceptions produced by TestConnection, you might need to perform the additional actions as shown in Figure 5.

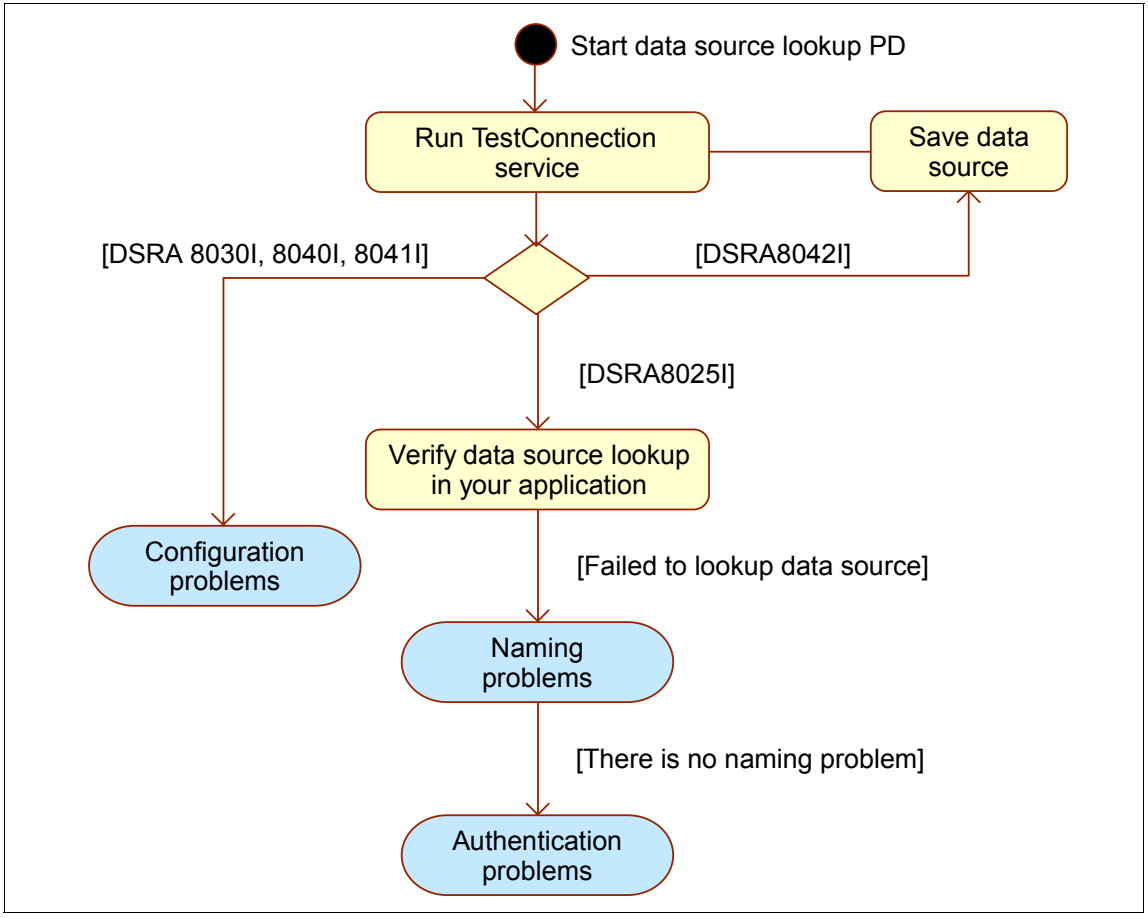

Figure 5 Data source lookup problem determination

Testing the connection results in one of the following outcomes:

- Message `DSRA8042I`

  This message indicates that the data source does not exist.

  If you created the data source but have not saved the configuration, the test connection will not find the data source. You need to save the data source and then re-run the TestConnection service.

- Messages DSRA8030I, DSRA8040I, or DSRA8041I

  These messages indicate that your connection is either failing or having problems. In this case, the TestConnection service produces additional DSRA messages in the range of 8000 to 8499 in the SystemOut.log.

  Proceed to “Configuration problems” on page 18 for further analysis.

- Message DSRA8025I

  This message indicates that the TestConnection service can connect to the data source successfully. There is nothing wrong with the data source configuration. So, the next step is to verify the data source lookup code in your application. One possible way to perform this verification is to run the application through a debugger with a breakpoint set after the lookup() method call on your java.naming.Context object. Another way is to create and run a small program with the data source lookup code cut and pasted from your application.

  If the data source lookup is successful, the new data source is OK, and you should go to “The next step” on page 39.

  If the data source lookup is not successful, you might have a naming or authentication problem. To determine if you have a naming problem, see “Naming problems” on page 24.

  If there is no problem with naming, see “Authentication problems” on page 25 to determine if you have a problem authenticating with the database.

If you reach this point without identifying the problem, go to “The next step” on page 39.

Symptom: Failure to connect to an existing data source

In the situation where a current session with the application has been working or a new session is started with no errors on naming or authentication, you can assume that the application was successful finding the data source (or connection factory). A failure to establish connectivity at this stage is broadly identified as a failure to get a new connection to an existing data source.

Data to collect

All relevant error messages and exceptions that are needed for our analysis are available in SystemOut.log and SystemErr.log.

What to look for

You need to evaluate the situation when there is failure on getting a connection. One possible way to do this is to run the application through a debugger with a breakpoint set after the getConnection() method call on your

javax.resource.cci.ConnectionFactory or javax.sql.DataSource object. Another way is to create and run a small program with the getting connection code cut and pasted from your application. You can run this test case many times and observe the outcomes as shown in Figure 6.

,

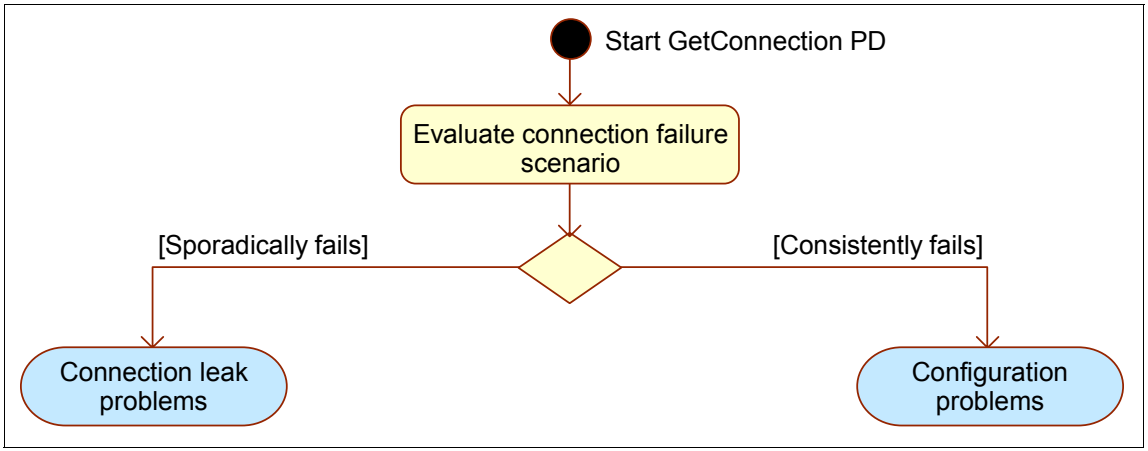

Figure 6 Problem determination path for failed connections

Examining the situation where there is a failure getting a new connection can lead you to one of the following two cases:

- Connection consistently fails, which means that you might have a problem with your application configuration. Proceed to “Configuration problems” on page 18 for further analysis.
- Connection fails sporadically, which means that it is very likely that you have a connection leak. Proceed to “Connection leak problems” on page 26 for further analysis.

If none of these symptoms apply, go to “The next step” on page 39.

Symptom: Failure to access a resource through JDBC

When an application uses JDBC to access a relational database and the access fails, you most likely receive error messages with prefixes `WTRN` (transaction problem) or `DSRA` (data source resource adapter).

Data to collect

All relevant error messages and exceptions that are needed for our analysis are available in SystemOut.log and SystemErr.log.

What to look for

Examine the exceptions in SystemOut.log and SystemErr.log for more information, as shown in Figure 7.

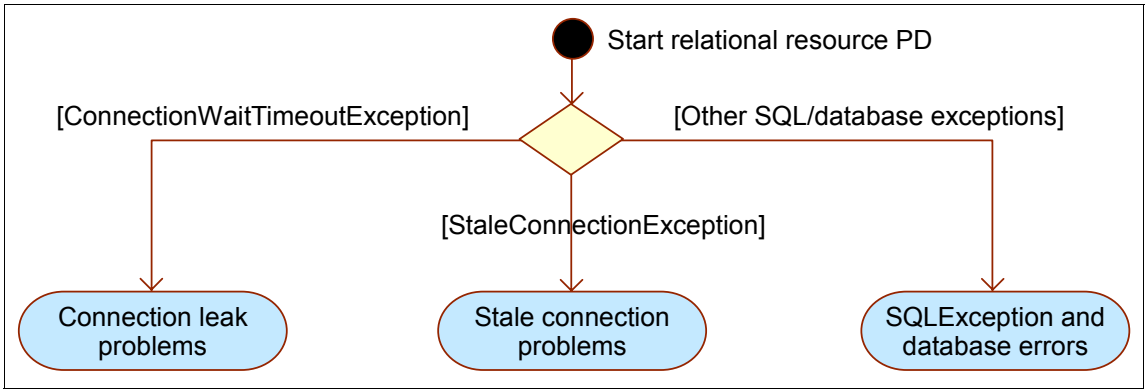

Figure 7 Relational resource problem determination

Depending on what you find, take one of the following actions:

- ConnectionWaitTimeoutException

  Proceed to “Connection leak problems” on page 26 for further analysis.

- StaleConnectionException

  Proceed to “Stale connection problems” on page 33 for further analysis.

- Other SQL/database exceptions

  Proceed to “SQLException and database errors” on page 35 for further analysis.

If none of these symptoms apply, go to “The next step” on page 39.

Symptom: Failure to access a non-relational resource

When your application uses non-relational resource adapters, such as CICS® ECI, Siebel, or SAP, the failure to access a resource usually involves the XAResource that manages transactions or connection factory that creates connections to the resource.

Data to collect

All relevant error messages and exceptions that are needed for our analysis are available in SystemOut.log and SystemErr.log.

What to look for

In most situations, you receive error messages with prefixes `WTRN` (transaction problem) or `J2CA` (general JCA connector problem). The errors can be classified into scenarios as shown in Figure 8.

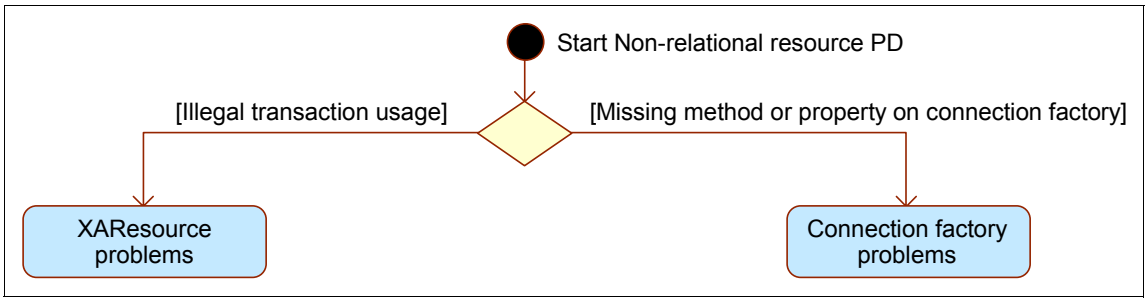

Figure 8 Non-relational resource problem determination

Depending on what you find, do one of the following:

- Illegal transaction usage

  Proceed to “XAResource problems” on page 36 for further analysis.

- Missing method or property on connection factory

  Proceed to “Connection factory problems” on page 37 for further analysis.

If none of these symptoms apply, go to “The next step” on page 39.

Analyzing problem areas

The symptom analysis from the previous section is designed to help you identify specific areas to explore that might be relevant to your problem. This section continues the process of symptom analysis but with more emphasis on the components that might be at the root of the problem. For each problem area, you perform a series actions to help you determine the cause of the error and are given possible resolutions or advice on how to proceed.

If you reached this point without being able to identify possible problem areas, see “The next step” on page 39.

Configuration problems

Use the steps discussed in this section to diagnose configuration problems.

Review the configuration

The first step in addressing configuration problems is to ensure that you understand how the resource should be configured. If this is a new resource, review the following article in the WebSphere Information Center for information about how to create and configure a JDBC provider and data source:

http://publib.boulder.ibm.com/infocenter/wasinfo/v6r0/topic/com.ibm.websphere.base.doc/info/aes/ae/tdat_tccrtprovds.html

Review the WebSphere Information Center for known issues

The next step is to check the WebSphere Information Center for information about common data source configuration problems:

http://publib.boulder.ibm.com/infocenter/wasinfo/v6r0/topic/com.ibm.websphere.nd.doc/info/ae/ae/rtrb_dsaccess.html

Tip for Oracle users:

Oracle 10g is supported officially in WebSphere Application Server V6. If you are using the Oracle 10g JDBC driver, your data source must use the com.ibm.websphere.rsadapter.Oracle10gDataStoreHelper data store helper class.

Oracle 10g resolves the following known issues with the Oracle 8i or 9i JDBC drivers:

- XA with Serializable isolation level is now supported.
- Oracle lifted the 4 KB limitation on BLOB data type with Oracle Thin JDBC driver.
- Oracle now returns a connection which allows connections with auto commit mode set to `true` to start an XA transaction.

If the configuration appears to be correct and none of the known issues apply to your situation, the next step is to search the JVM logs for meaningful error messages or exceptions.

Examine the JVM logs for error messages and exceptions

Search SystemOut.log and SystemErr.log for `DSRA` messages ranging from 8000 through 8499. If you find messages in this range, it is still likely that your data source configuration needs correction.

The following WebSphere Information Center article contains information about these messages, including an explanation and user response:

http://publib.boulder.ibm.com/infocenter/wasinfo/v6r0/topic/com.ibm.websphere.messages.doc/doc/DSRA.html

The various resolution options depends on the types of exceptions that you encounter. The most common exceptions that are related to data source's configuration are:

- ClassNotFoundException

  The JVM throws a ClassNotFoundException when an application tries to load in a class using its string name in the following three methods but no definition for the class with the specified name could be found:

  - Class.forName()
  - ClassLoader.findSystemClass()
  - ClassLoader.loadClass()

  Verify that the class path and the JAR file(s) for the JDBC provider are correct and exist on the server where the application with the associated data source is running.

  If the definition of the data source includes a WebSphere variable, make sure that the variable is defined and set at a correct scope (from the administrative console, go to **Environment** → **WebSphere Variables**).

  Remember that directories, file names, symbolic links, and so forth on UNIX platforms are case sensitive. Make sure that you have defined the directory and file names exactly as you see them on the UNIX file system.

- ClassCastException

  The JVM throws a ClassCastException to indicate that the code has attempted to cast an object to a subclass of which it is not an instance:

  ```
  (targetClass)sourceObject;
  ```

  ClassCastException occur whenever:

  - The type (class) of sourceObject is not a subclass of targetClass
  - The type (class) of sourceObject is in the class ancestry of class targetClass, but the class loader which loaded the class of sourceObject is different than the class loader that loaded class targetClass

  When dealing with a data source configuration, the latter case is the most likely. WebSphere Application Server's runtime is a multi-class loader environment and when targetClass is visible to more than one class loader, more than one instance is deployed.

  Check to make sure that there is only one copy of each JAR file that is required for the JDBC provider. Search the file system starting from the

<WAS_install_root> directory for duplication. Make sure the application's EAR file does not include the JDBC driver's JAR files.

- UnsatisfiedLinkError

  The JVM throws UnsatisfiedLinkError when it cannot find the native library or when the JVM has already loaded the native library.

  This exception only occurs when you are using type 2 JDBC drivers (for example, DB2 CLI Legacy, DB2 Universal Type 2, and Oracle OCI drivers). Type 2 JDBC drivers require that some binary code is loaded on each client machine (WebSphere Application Server machine). These native libraries are:

  – *.dll files on Windows platforms
  – *.so files on Solaris™
  – *.sl files on HP-UX
  – *.a or *.so files on AIX®

  For more information about different types of JDBC drivers, see the Sun™ Developer Network at:

  `http://java.sun.com/products/jdbc/driverdesc.html`

  If there is no value specified for the native library path in your data source's configuration, you might get the UnsatisfiedLinkError:

  – On Windows platforms, set the system environment PATH to include the path where the *.dll files can be loaded.
  – On UNIX platforms, you might need to set up the user's profile to set the environment variables before starting the WebSphere Application Server processes.

  Examine the native file name from the error message and confirm that it has the correct format and exists on the server.

  Table 1 on page 21 shows the list of JDBC drivers that are supported by WebSphere Application Server V6 and their JAR files, dependent environment variables, and so forth.

Table 1 Driver type-to-JAR file mapping

Database	Driver	Type	Required Environment Variable(s)	Default Jar Files
IBM				
DB2	DB2 CLI Legacy	2	DB2INSTANCE LD_LIBRARY_PATH[5]	db2java.zip
	DB2 Universal JDBC (or JCC)	2, 4	LD_LIBRARY_PATH[5]	db2jcc.jar db2jcc_license_cu.jar db2jcc_license_cisuz.jar
	OS/400® Toolbox	4		jt400.jar
	OS/400 Native	2		db2_classes.jar
Informix	Informix JDBC	4		ifxjdbc.jar and ifxjdbcx.jar[1]
Cloudscape™	Universal JDBC	4		db2jcc.jar, db2jcc_license_cu.jar and db2jcc_license_cisuz.jar
DataDirect Technology				
SQL Server	Sequelink 5.4	3		sljc.jar, spy-sl.ja2[2]
	Connect JDBC	4		base.jar, sqlserver.jar, util.jar and spy.jar[3]
Oracle Corp				
Oracle	OCI	2	ORACLE_HOME LD_LIBRARY_PATH[5]	ojdbc14.jar[4]
	Thin	4		ojdbc14.jar[4]
Microsoft® Corp				
SQL Server	SQL Server 2000	4		msbase mssqlserver.jar msutil.jar
Sybase Corp				
Sybase	jConnect	4		jconn2.jar

1. Informix: ifxjdbc_g.jar and ifxjdbcx_g.jar (debug version provides JDBC trace)
2. SQL Server: spy-sl.jar provides JDBC trace
3. SQL Server: spy.jar provides JDBC trace
4. Oracle: ojdbc14_g.jar (debug version provides JDBC trace)
5. LD_LIBRARY_PATH for Solaris and Linux®, LIBPATH for AIX, SHLIB_PATH for HP-UX and PATH for Windows.

If you have verified that the JDBC provider and data source configuration are correct and you still get the error connecting to the database, write a stand-alone JDBC program to see if you can connect to the back-end database using the same JDBC driver.

If the stand-alone JDBC program gives the same error, then the JDBC driver version might be incompatible with the back-end database version. If this is the case, you might need to re-install the JDBC driver and database client.

If the stand-alone JDBC program can connect to the back-end database, compare the settings of the environment variables (for example CLASSPATH, LIBPATH, and so forth) to make sure the settings from the successful session are the same as those settings in WebSphere Application Server.

- java.sql.SQLException: invalid arguments in call

  This exception occurs when the database requires a user ID and password for a connection but the data source does not have the custom properties for user and password defined or there is no J2C authentication alias specified for the component-managed authentication alias (used only when the application resource reference is using `res-auth = Application`) .

  To specify the user ID and password using a J2C authentication alias, you first need to create the alias (if it does not exist). To create new J2C authentication alias using the administrative console:

  a. Navigate to **JDBC providers** → ***<provider>*** → **Data sources** → ***<datasource>***.

  b. In the Related Items section, select **J2EE Connector Architecture (J2C) authentication data entries.**

  c. Click **New** to create a new entry.

  When the alias is created, you need to select it in the component-managed authentication alias field in the data source.

Tips when using an XA connection:

- Oracle

  Oracle databases must be configured with the Oracle JVM. See the following Technote for one method that you can use to configure an Oracle database with the Oracle JVM:

  http://www-1.ibm.com/support/docview.wss?rs=180&uid=swg21046276

  As an alternative, refer to the Oracle documentation for instructions on configuring the database with the Oracle JVM using scripts.

- SQL Server

  The Microsoft Distributed Transaction Coordinator service must be running on the server where SQL Server is running.

  When using DataDirect Connect JDBC or WebSphere embedded Connect JDBC Driver, a set of stored procedures must be installed on the master database. See the *DataDirect Connect for JDBC User's Guide and Reference*, that is available from DataDirect Technologies at URL:

  http://www.datadirect.com/techres/jdbcproddoc/index.ssp

  In particular, see *Installing Stored Procedures for JTA Instructions* in Chapter 6, *The Microsoft SQL Server Driver*.

- DB2

  If the DB2 server is on OS/390®, you must configure the Sync Point Manager for multi-site updates from DB2 Connect™ to DB2 OS/390. See technote:

  http://www-1.ibm.com/support/docview.wss?rs=180&uid=swg21114514

- Informix

  If your application requires an XA connection to an Informix database, then you should use Informix JDBC 3.0 because there are known problems with an XA connection using Informix JDBC 2.2.x. See the following Technotes:

  – *XA transactions are not working properly with Informix Dynamic Server*

    http://www-1.ibm.com/support/docview.wss?rs=180&uid=swg21204793

  – *setTransactionIsolation on a connection does not work for XA transactions in Informix 9.4 and Informix 10.0*

    http://www-1.ibm.com/support/docview.wss?rs=180&uid=swg21210281

Naming problems

This section discusses the situation where you see `NamingException` when the application accesses a data source. Proceed with the following steps to diagnose a naming problem.

Ensure that the application uses resource references

Applications are required to use a resource reference to access a data source or connection factory. For information about this, see the WebSphere Information Center item *Looking up data sources with resource references for relational access* at:

`http://publib.boulder.ibm.com/infocenter/wasinfo/v6r0/topic/com.ibm.websphere.nd.doc/info/ae/ae/cdat_datsorres.html`

Ensure that an indirect JNDI name is used

When defining a connection factory or data source, a JNDI name by which the connection factory or data source can be looked up by a component is identified. Preferably an indirect name with the *java:comp/env* prefix should be used (and must be used in future releases). An indirect name makes the resource reference data associated with the application available to the connection management runtime. This enhances resource management through the use of the res-auth, res-isolation-level, res-sharing-scope, and res-resolution-control settings.

Though you can still use a direct JNDI name (for example jdbc/myDataSource), this naming method is deprecated in WebSphere Application Server V6. WebSphere Application Server assigns default values to the resource reference data when you use this method. You see the warning message `J2CA0294W` logged in the JVM's SystemOut.log to document the defaults, as shown in Example 8.

Example 8 Warning message J2CA0294W when using direct JNDI name

```
J2CA0294W: Deprecated usage of direct JNDI lookup of resource jdbc/IOPEntity.
The following default values are used: [Resource-ref settings]
   res-auth:                 1 (APPLICATION)
   res-isolation-level:      0 (TRANSACTION_NONE)
   res-sharing-scope:        true (SHAREABLE)
   loginConfigurationName:   null
   loginConfigProperties:    null
[Other attributes]
   res-resolution-control:   999 (undefined)
isCMP1_x:                 false (not CMP1.x)
isJMS:                false (not JMS)
```

The application should be changed to use an indirect JNDI name (for example, java:comp/env/jdbc/myDataSource) instead of the direct JNDI name

(jdbc/myDataSource), and a resource reference should be created. See the WebSphere Information Center item *Connection factory JNDI name tips* at:

http://publib.boulder.ibm.com/infocenter/wasinfo/v6r0/topic/com.ibm.websphere.nd.doc/info/ae/ae/rdat_jnditips.html

It is possible to suppress the J2CA0294W warning message from the SystemOut.log by setting the logging level to:

```
*=info:com.ibm.ejs.j2c.ConnectionFactoryBuilderImpl=severe
```

A setting of `severe` for that class suppresses the logging of warning and lower level messages. However, this should be considered as an interim solution until the application can be updated.

For more information, see the WebSphere Information Center section *Naming* at:

http://publib.boulder.ibm.com/infocenter/wasinfo/v6r0/topic/com.ibm.websphere.nd.doc/info/ae/ae/welc6tech_nam.html

Authentication problems

J2C connector authentication data entries are used by resource adapters and JDBC data sources. Authentication is the process that identifies the caller. If authentication fails, it is likely that you have incorrectly configured the user's identity.

For more information, see the *Troubleshooting security configuration* section in the WebSphere Information Center at:

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

To verify the required configuration, do the following steps.

Check user registry configuration

First, you need to ensure the following:

- The user name and password that are specified in the J2C authentication alias are correct and can be used to access the database.
- The user registry that you defined the user and user ID in is the same registry that is specified in the WebSphere global security settings.

Check the J2C authentication data entry

When configuring resource adapters or data sources, the administrator specifies the alias for the J2C authentication data entry to use to access the resource. In the General properties of the data source, verify that you have a J2C authentication data entry specified in the `Component-managed authentication alias` field.

Check the application

Any client running in the same cell that can look up a resource in the JNDI namespace can obtain connections without explicitly providing authentication data on the getConnection() call. If the component's res-auth setting is `Application`, authentication is taken from the component-managed authentication alias defined on the connection factory. With res-auth set to `Container`, authentication is taken from the login configuration defined on the component's resource-reference. It is important to note that J2C authentication alias is defined per cell. An enterprise bean or servlet in one application server cannot look up a resource in another server process in a different cell, because the alias would not be resolved.

If the application explicitly provides a user ID and password on the getConnection() when obtaining the connection (that is getConnection(*user*,*password*) and the component's res-auth setting is set to `Application`, the authentication data that is used is the user ID and password that is passed in the getConnection() method. You need to ensure that the specified user ID and password can be used to access the database.

Enable traces

If the configuration appears to be correct and you continue to have authentication problems, you should enable WAS.j2c and RRA traces, recreate the problem, and then proceed to "The next step" on page 39.

Connection leak problems

A connection leak occurs when the application uses a connection but it never explicitly calls the close() method on the connection object to return the connection back to the connection pool for reuse. Example 9 demonstrates the proper closing of connection after usage.

Example 9 Closing a connection

```
Connection pooledCcon = null;
DataSource ds = null;
InitialContext ic = null;
try {
   ds = (DataSource)ic.lookup("java:comp/env/jdbc/myDataSource");
```

```
   pooledCon = ds.getConnection("username", "password");
   // Processing Code goes here
} catch (Exception ignored) {
   // catch JNDI or JDBC exceptions here
} finally {
   // If not calling close, it is considered connection leak
   if(pooledCon != null)
   pooledCon.close();
}
```

You might have a connection leak if your application receives exceptions such as com.ibm.websphere.ce.cm.ConnectionWaitTimeoutException or com.ibm.websphere.ce.j2c.ConnectionWaitTimeoutException when attempting to access a data source or JCA-compliant resource adapter, respectively.

In Example 10, the application receives a ConnectionWaitTimeoutException after waiting for 1800030 milliseconds for a connection, but no free connection from the pool is available. If the connection pool has a high value for the connection timeout (for example, 1800 seconds), you might not see the ConnectionWaitTimeOutException immediately.

Example 10 ConnectionWaitTimeoutException

```
[7/12/05 14:04:58:328 AST] 3ad6086e ConnectionMan E J2CA0020E: The
Connection Pool Manager could not allocate a Managed Connection:
com.ibm.websphere.ce.j2c.ConnectionWaitTimeoutException: Connection
not available, Timed out waiting for 1800030
at java.lang.Throwable.<init>(Throwable.java:195)
at java.lang.Exception.<init>(Exception.java:41)
at
javax.resource.ResourceException.<init>(ResourceException.java:73)
at
javax.resource.spi.ResourceAllocationException.<init>(ResourceAlloc
ationException.java:55)
at
com.ibm.websphere.ce.j2c.ConnectionWaitTimeoutException.<init>(Conn
ectionWaitTimeoutException.java:38)
at
com.ibm.ejs.j2c.poolmanager.FreePool.createOrWaitForConnection(Free
Pool.java:1100)
at
com.ibm.ejs.j2c.poolmanager.PoolManager.reserve(PoolManager.java:17
47)
```

Before deciding this problem is a connection leak, you should rule out any performance tuning problems. When you have ruled out these and if the problem still exists, you have to do further analysis.

Review your current connection pool use

Examining the connection pool settings can reveal some possible causes for ConnectionWaitTimeoutException due to the pool over use. The following are possibilities:

- 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 of this problem is that you receive these exceptions regularly, but your process use is not high. This exception indicates that there are too few connections available to keep the threads in the server busy.

- Connection timeout 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 are 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 connections, or you close them long after you are finished using them, thus 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 that 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 back-end system has 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 process is nearly 100% busy.

Tune the connection pool settings

Before continuing, review the WebSphere Information Center item *Connection pool settings* at:

http://publib.boulder.ibm.com/infocenter/wasinfo/v6r0/topic/com.ibm.websphere.express.doc/info/exp/ae/udat_conpoolset.html

You can adjust the following parameters for a data source:

- Maximum Connections

  Specifies the maximum number of ManagedConnections that can be created in the pool.

  ManagedConnections represent the physical connection to the back-end resource. When the maximum number is reached, no new ManagedConnections are created, and the requester waits until a ManagedConnection currently in use is returned to the pool or a ConnectionWaitTimeoutException is thrown. The default value is 10, allowing the number of ManagedConnections to grow to 10. If the maximum connections is changed to 0, the number of ManagedConnections can grow infinitely, and Connection Timeout will not be used.

  For example, if Maximum Connections is set to 5 and there are 5 ManagedConnections in use, the pool manager waits for a managed connection to become free for Connection Timeout seconds. A ConnectionWaitTimeoutException is thrown if a connection is not returned to the pool in time.

- Connection Timeout

  Specifies the interval, in seconds, after which a connection request times out and a ConnectionWaitTimeoutException is thrown.

  This value indicates the number of seconds that a request for a connection waits when there are no connections available in the free pool and no new connections can be created, usually because the maximum value of connections in the particular connection pool has been reached. For example, if Connection Timeout is set to 300, and the maximum number of connections are all in use, the pool manager waits for 300 seconds for a physical connection to become available.

  If a physical connection is not available within this time, the pool manager initiates a ConnectionWaitTimeoutException. It usually does not make sense to retry the getConnection() method. If a longer wait time is required, you should increase the Connection Timeout setting value. If a ConnectionWaitTimeoutException is caught by the application, the administrator should review the expected connection pool usage of the application and tune the connection pool and database accordingly.

  If the Connection Timeout is set to 0, the pool manager waits as long as necessary until a connection becomes available. This happens when the application completes a transaction and returns a connection to the pool, or when the number of connections falls below the value of Maximum Connections, allowing a new physical connection to be created.

  If Maximum Connections is set to 0, which enables an infinite number of physical connections, then the Connection Timeout value is ignored.

Enable connection leak trace facility

If you have ruled out performance tuning problems and the problem still exists, you need to enable the connection leak trace logic for further analysis.

After adjusting your connection pool settings, if it takes longer to receive the ConnectionWaitTimeoutException, then there is a high possibility that your application has a connection leak. The WebSphere Application Server V6 connection manager provides a diagnostic feature called the connection leak trace logic that gathers information about which application methods could potentially lead to leaking connections (not closing connection) or holding on to connections longer than expected (for example, long running queries).

To enable the connection leak trace logic, set the log detail level to ConnLeakLogic=finest.

Note: The WAS.j2c trace includes the ConnLeakLogic trace. If you enable WAS.j2c trace, then you do not have to enable ConnLeakLogic.

When you enable the ConnLeakLogic trace, for every time interval (the default is 10 seconds), WebSphere connection manager checks how long a connection has been in use and prints the stack trace to trace log. Currently, the default time interval is unchangeable. If you have a need to change the default value, contact IBM technical support to obtain an iFix that allows you to add a custom property at the data source.

What to look for in the trace

When a connection is leaked or held longer than the trace time interval (10 seconds), the trace contains the string `Connection Leak Logic Information` followed by a stack trace with all methods involved up to the getConnection(), including the application method that acquired the connection, as shown in Example 11.

Example 11 Connection Leak Logic Information with stack trace

```
Connection Leak Logic Information:
  MCWrapper id 16089ef2  Managed connection
com.ibm.ws.rsadapter.spi.WSRdbManagedConnectionImpl@6e909ef2
State:STATE_ACTIVE_INUSE Thread Id: 15861ecd Thread Name:
Servlet.Engine.Transports : 1
     Start time inuse Wed Nov 03 08:04:54 CST 2004 Time inuse 20 (seconds)
     Last allocation time Wed Nov 03 08:04:54 CST 2004
       getConnection stack trace information:
  at
com.ibm.ejs.j2c.ConnectionManager.allocateConnection(ConnectionManager.java:565
)
```

```
  at
com.ibm.ws.rsadapter.jdbc.WSJdbcDataSource.getConnection(WSJdbcDataSource.java:
215)
  at
com.ibm.ws.rsadapter.jdbc.WSJdbcDataSource.getConnection(WSJdbcDataSource.java:
306)
  at SnoopServlet.doGet(SnoopServlet.java:130)
  at javax.servlet.http.HttpServlet.service(HttpServlet.java:740)
  at javax.servlet.http.HttpServlet.service(HttpServlet.java:853)
  at
com.ibm.ws.webcontainer.servlet.StrictServletInstance.doService(StrictServletIn
stance.java:110)
  at
com.ibm.ws.webcontainer.servlet.StrictLifecycleServlet._service(StrictLifecycle
Servlet.java:174)
  at
com.ibm.ws.webcontainer.servlet.IdleServletState.service(StrictLifecycleServlet
.java:313)
  at
com.ibm.ws.webcontainer.servlet.StrictLifecycleServlet.service(StrictLifecycleS
ervlet.java:116)
  at
com.ibm.ws.webcontainer.servlet.ServletInstance.service(ServletInstance.java:28
3)
  at
com.ibm.ws.webcontainer.servlet.ValidServletReferenceState.dispatch(ValidServle
tReferenceState.java:42)
  at
com.ibm.ws.webcontainer.servlet.ServletInstanceReference.dispatch(ServletInstan
ceReference.java:40)
  at
com.ibm.ws.webcontainer.webapp.WebAppRequestDispatcher.handleWebAppDispatch(Web
AppRequestDispatcher.java:1059)
  at
com.ibm.ws.webcontainer.webapp.WebAppRequestDispatcher.dispatch(WebAppRequestDi
spatcher.java:588)
  at
com.ibm.ws.webcontainer.webapp.WebAppRequestDispatcher.forward(WebAppRequestDis
patcher.java:206)
  at com.ibm.ws.webcontainer.srt.WebAppInvoker.doForward(WebAppInvoker.java:80)
  at
com.ibm.ws.webcontainer.srt.WebAppInvoker.handleInvocationHook(WebAppInvoker.ja
va:214)
  at
com.ibm.ws.webcontainer.cache.invocation.CachedInvocation.handleInvocation(Cach
edInvocation.java:71)
  at
com.ibm.ws.webcontainer.srp.ServletRequestProcessor.dispatchByURI(ServletReques
tProcessor.java:182)
```

```
  at
com.ibm.ws.webcontainer.oselistener.OSEListenerDispatcher.service(OSEListener.j
ava:334)
  at
com.ibm.ws.webcontainer.http.HttpConnection.handleRequest(HttpConnection.java:5
6)
  at
com.ibm.ws.http.HttpConnection.readAndHandleRequest(HttpConnection.java:615)
  at com.ibm.ws.http.HttpConnection.run(HttpConnection.java:439)
  at com.ibm.ws.util.ThreadPool$Worker.run(ThreadPool.java:672)
```

After enabling the connection leak trace logic, let the application run and monitor the trace. Search the trace file for string `Connection Leak Logic Information`. **If** you see it, there are connections that have been in use for more than 10 seconds. You can analyze the stack trace for the suspected application methods. In Example 11 on page 30, the doGet() method of the SnoopServlet.java is the suspected source for leaking connections.

Check shareable connection settings

WebSphere Application Server supports both *unshareable* and *shareable* connections. An unshareable connection is not shared with other components in the application. The component that uses this connection has full control of this connection. On the other hand, a shareable connection can be shared with other components within the same transaction, as long as each getConnection() request has the same connection properties.

If you use shareable connections, incorrect settings can cause a connection leak. So, ensure that the following connection properties are the same:

- JNDI name. While not actually a connection property, this requirement simply means that you can only share connections from the same data source in the same server.
- Resource authentication.
- In relational databases:
  - Isolation level (corresponds to access intent policies applied to CMP beans)
  - Readonly
  - Catalog
  - TypeMap

To enable connection sharing for resource adapters within the same transaction, the following connection properties must be the same:

- JNDI name. While not actually a connection property, this requirement simply means that you can only share connections from the same resource adapter in the same server.
- Resource authentication.

Resolution

After the source for leaking connections has been identified, you should ask your application developer to review the source and fix the problem. Some other possible solutions to correct the ConnectionWaitTimeoutException are:

- Modify your application to use fewer connections.
- Check your application to make sure the connections are properly closed.

WebSphere Application Server establishes a queuing network that is comprised of the Web server, Web container, EJB container, Object Request Broker (ORB), and data source components. You should review and adjust these queues to improve WebSphere performance. See the following for more information:

- *IBM WebSphere Application Server - Performance Tuning: Adjusting WebSphere Application Server System Queues*

  http://www.redbooks.ibm.com/abstracts/tips0244.html

- *IBM WebSphere Application Server - Performance Tuning: Determining Optimum Queue Sizes*

  http://www.redbooks.ibm.com/abstracts/tips0245.html

Also, review the following articles in the WebSphere Information Center:

- *Tuning the application serving environment*

  http://publib.boulder.ibm.com/infocenter/wasinfo/v6r0/topic/com.ibm.websphere.nd.doc/info/ae/ae/tprf_tuneprf.html

- *Tuning parameters for data access resources*

  http://publib.boulder.ibm.com/infocenter/wasinfo/v6r0/topic/com.ibm.websphere.nd.doc/info/ae/ae/rdat_datobjtune.html

Stale connection problems

You have a stale connection problem when you receive the exception com.ibm.websphere.ce.cm.StaleConnectionException. This exception indicates that the connection currently held is no longer valid.

When a connection to a database becomes stale, any operation that is performed on that connection receives an SQLException from the JDBC driver. SQLException is a generic exception. It contains SQLState and SQLCode values that you can use to determine the meaning of the exception. The meaning of the SQLState and SQLCode varies depending on the database vendor. The WebSphere RRAs maintain the mapping of SQLStates and SQLCodes to StaleConnectionExceptions for each of the supported relational databases. When the connection pooling runtime catches an SQLException, it checks to see if this SQLException is considered a StaleConnection exception for the database server in use.

Before proceeding, see the *Stale connections* item in the WebSphere Information Center for information about how to detect stale connections and how to recover from stale connections:

http://publib.boulder.ibm.com/infocenter/wasinfo/v6r0/topic/com.ibm.websphere.base.doc/info/aes/ae/cdat_stalecon.html

To analyze the stale connection problems, proceed with the steps discussed in the following sections.

Check timeout settings

Check with your database administrator or network system administrator to determine if there are timeout settings on the database or firewall. Timeouts on the database and firewall can close your connections and thus cause the StaleConnectionException to occur. You might need to disable your database or firewall timeouts or review the connection pool settings so that they are suited to your environment. For example, the connection pool aged timeout should be less than the firewall timeout, which should be less than the database timeout.

Determine if a specific query is getting the exception

If you are getting StaleConnectionException when executing a certain query, it is likely that the query causes the JDBC driver to return the SQLException with an SQLState and SQLCode that are mapped to StaleConnectionException. The SQLState and SQLCode can help you determine the root cause of the problem.

Trace the problem

A JDBC driver returns an SQLException when there is problem with the query that is executed on the connection. Sometimes the connections are unusable after the SQLException occurs, but the application server does not throw the StaleConnectionException because the SQLState and SQLCode are not the ones from the mapping list for StaleConnectionException.

If this is the case, you should try to recreate the problem with a simple test case and trace the problem with the WAS.database, RRA, and WAS.j2c trace options

enabled. The trace should help you identify the query that is causing the problem and the SQLException that is returned by JDBC driver.

SQLException and database errors

If you receive an SQLException (not caused by connection leaks or stale connections) or database-specific exceptions, perform the steps that are discussed in this section.

Check your database error code documentation

An SQLException is often accompanied with a database-specific error code. In this case, you should check the database product documentation for an explanation and actions to take on the error.

Example 12 shows an SQLException for a DB2 database.

Example 12 SQLException caused by a database-specific error

```
java.sql.SQLException: [IBM][CLI Driver] CLI0108E  Communication link
failure. SQLSTATE=40003DSRA0010E: SQL State = 40003, Error Code =
-99,999
```

You can check DB2's CLI messages in the DB2 Information Center at:

http://publib.boulder.ibm.com/infocenter/db2help/topic/com.ibm.db2.udb.doc/core/r0climsg.htm

If you look up the message found in the SQLException, you find the following explanation:

```
CLI0108E Communication link failure.
Explanation: The connection between the driver and the data source failed
during execution of this function.
User Response: Establish a new connection.
```

Based on the information from your database documentation, you can decide on a suitable course of action. As in this example, one of the possible causes is a communication link failure. In this case, you need to check your network connection and then re-run the application to re-establish the connection.

Example 13 shows another instance of a DB2-specific error.

Example 13 SQL1040N

```
COM.ibm.db2.jdbc.DB2Exception: [IBM][CLI Driver] SQL1040N  The maximum number
of applications is already connected to the database.  SQLSTATE=57030
```

You can check DB2's SQL messages in the DB2 Information Center at:

http://publib.boulder.ibm.com/infocenter/db2help/topic/com.ibm.db2.udb.doc/core/rsql1000.htm

Example 14 shows the explanation for this message.

Example 14 Message SQL1040N explanation

```
SQL1040N The maximum number of applications is already connected to the
database.
Explanation: The number of applications connected to the database is equal to
the maximum value defined in the configuration file for the database. The
command cannot be processed.
User Response: Wait for other applications to disconnect from the database. If
more applications are required to run concurrently, increase the value for
maxappls. After all applications disconnect from the database and the database
is restarted, the new value takes effect.
```

In this case, the DB2 parameters, maxappls and maxagents, are set too low for the DB2 database. When configuring the data source settings for the database, ensure that the DB2 maxappls setting is greater than the maximum number of connections for the data source. The DB2 maxagents setting should be the sum of maxappls for all databases.

Trace the problem

When the suggested user response to the error code is not sufficient to resolve the problem, you might need to enable JDBC and database traces and engage the JDBC driver and database vendors to help you further determine the cause of the problem.

XAResource problems

The resource adapter has implementation code for the XAResource interface that supports transactions involving multiple resource managers. If your application has an error relating to invalid use of a transaction, you receive com.ibm.ws.Transaction.IllegalResourceIn2PCTransactionException (Example 15). This exception indicates that the application has incorrectly attempted to enlist multiple 1PC (1-phase commit or local transaction) resources in a 2PC (two-phase commit or XA) operation.

Example 15 XAResource transaction exception

```
[20/01/05 12:41:35:920 EST] 00000036 RegisteredRes E   WTRN0062E: An
illegal attempt to use multiple resources that have only one-phase
capability has occurred within a global transaction.
[20/01/05 12:41:36:326 EST] 00000036 LocalTransact E   J2CA0030E: Method
```

```
enlist caught
com.ibm.ws.Transaction.IllegalResourceIn2PCTransactionException: Illegal
attempt to enlist multiple 1PC XAResources
 at
com.ibm.ws.Transaction.JTA.RegisteredResources.enlistResource(Registered
Resources.java:362)
 at
com.ibm.ws.Transaction.JTA.TransactionImpl.enlistResource(TransactionImp
l.java:2693)
 at
com.ibm.ws.Transaction.JTA.TranManagerSet.enlistOnePhase(TranManagerSet.
java:417)
 at
com.ibm.ejs.j2c.LocalTransactionWrapper.enlist(LocalTransactionWrapper.j
ava:514)
 at
com.ibm.ejs.j2c.ConnectionManager.initializeForUOW(ConnectionManager.jav
a:1250)
 at
com.ibm.ejs.j2c.ConnectionManager.involveMCInTran(ConnectionManager.java
:938)
 at
com.ibm.ejs.j2c.ConnectionManager.allocateConnection(ConnectionManager.j
ava:571)
 at
com.ibm.connector2.cics.CICSConnectionFactory.getConnection(CICSConnecti
onFactory.java:211)
```

If you have this error, check the application to make sure that you do not have a resource that supports only local transactions (1PC) included accidentally within a context of a 2PC XA transaction. This is illegal.

Connection factory problems

Problems on ManagedConnectionFactory that are related to missing methods, missing connection factory properties, or failure to instantiate an object usually produce error messages with prefix `J2CA`. Most problems of this type can be tracked to the resource adapter's deployment descriptor (ra.xml) and WebSphere Application Server's resource configuration file (resources.xml).

The ra.xml file is supplied by the resource adapter vendor and can be found in the following location:

<WAS_install_root>/profiles/*<profile>*/installedConnectors/*<rarfile>*/META-INF/ra.xml

To find the resources.xml file, you need to be aware of the scope at which the resource adapter was defined (node, cell, server). For example, if you defined the resource adapter at the node level, the file is at:

<WAS_install_root>/profiles/*<profile>*/config/cells/*<cell>*/nodes/*<node>*/resources.xml

Proceed with the following actions.

Check ra.xml

If the error message indicates that a missing method or property is the problem, check for a mismatch between the resource adapter's XML definition in the ra.xml file and the actual Java class that is provided for the JavaBean's implementation. If there is a mismatch, you need to provide this failure notification to the resource adapter provider.

The error shown in Example 16 is caused because the Siebel-managed connection factory implementation class does not provide the code for the method set UserName.

Example 16 Connection factory problem: missing method

```
[2/22/05 15:19:26:508 EST] 3c6d43cd J2CXAResource W J2CA0008W: Class
com.ibm.wps.wpai.jca.siebel.SiebelManagedConnFactory used by resource siebel
did not contain method set UserName. Processing continued.
```

If the connection factory fails to instantiate an instance, as shown in Example 17, you might have an invalid definition in the ra.xml file that causes the class loader to fail to load the ManagedConnectionFactory class.

Example 17 Connection factory problem: missing class

```
[6/28/05 10:20:27:216 HKT] 3b0010ba ConnectionFac E J2CA0009E: An exception
occurred while trying to instantiate the ManagedConnectionFactory class
com.ibm.ws.rsadapter.spi.WSManagedConnectionFactoryImpl used by resource
```

Check resources.xml

It is also possible that the resources.xml file might be corrupted. You should verify that resources.xml contains the correct entry for the resource adapter. For example:

```
<resources.j2c:J2CResourceAdapter xmi:id="J2CResourceAdapter_1125056353109"
name="IMS Connector for Java" archivePath="${CONNECTOR_INSTALL_ROOT}/
jca15_ims9102.rar">
```

If not, or if it looks incorrect, delete and re-create the resource adapter using the administrative console (**Resource** → **Resource Adapters)**.

The next step

The symptoms and problem areas included in this paper are some that you are more likely to experience. However, there are other things that can go wrong, or the cause of the problem might be related to something other than JCA components.

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

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

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

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

Next, collect all of the data that is outlined in the appropriate MustGather documents and raise a problem record with IBM as follows:

- MustGather: Database connections or connection pooling problems for all releases of V4.0, V5.0, and V6.0 at:

  http://www-1.ibm.com/support/docview.wss?rs=180&uid=swg21145599

- JVM logs
- FFDC logs
- A WebSphere Application Server trace with the following components enabled:
  - WAS.j2c
  - RRA
  - WAS.database

Note: The WAS.database trace group traces the JDBC calls for any of the supported JDBC drivers. You must use the debug jar file(s) for the associated JDBC driver if you enable WAS.database (see Table 1 on page 21).

You can use this documentation when you contact IBM support.

Notices

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

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

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

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

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

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

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

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

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

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

This document created or updated on September 29, 2005.

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

Trademarks

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

@server®	CICS®	OS/390®
@server®	DB2 Connect™	OS/400®
Redbooks (logo) ™	DB2®	Redbooks™
AIX®	Informix®	WebSphere®
Cloudscape™	IBM®	

The following terms are trademarks of other companies:

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

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

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

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

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