Data access -- table of contents

Development

Administration

4.2.4.2: Obtaining and using database connections

IBM WebSphere Application Server Version 3.5 provides two options for accessing database connections:

- Connection pooling (model based on JDBC 2.0)
- Connection manager (now deprecated, model based on JDBC 1.0)

Because connection pooling is the most efficient model for Web applications, it is recommended that you use connection pooling for all new applications requiring database access. You should consider migrating existing applications to connection pooling if your applications use connection manager or the standard JDBC 1.0 methods for getting database connections.

IBM WebSphere Application Server also provides data access beans, which offer a rich set of features for working with relational database queries and result sets.

For a comprehensive treatment of WebSphere connection pooling and data access,be sure to read the IBM whitepaper to be published on the Webduring the summer of 2001.

4.2.4.2.1: Accessing data with the JDBC 2.0 Optional Package APIs

In JDBC 1.0 and the JDBC 2.0 Core API, the DriverManager class is used exclusively for obtaining a connection to a database. The database URL, user ID, and password are used in the getConnection() call. In the JDBC 2.0 Optional Package API, the DataSource object provides a means for obtaining connections to a database. The benefit of using datasourcesis that the creation and management of the connection factory is centralized. Applications do not need to have specific information like the database name, user ID, or password in order to obtain a connection to the database.

The steps for obtaining and using a connection with the JDBC 2.0 Optional Package API differ slightly from those in the JDBC 2.0 Core API example. Using the extensions, you access a relational database as follows:

1. Retrieve a DataSource object from the JNDI naming service
2. Obtain a Connection object from the datasource
3. Send SQL queries or updates to the database management system
4. Process the results

The connection obtained from the datasource is a pooled connection. This means that the Connection object is obtained from a pool of connections managed by IBM WebSphere Application Server.The following code fragment shows how to obtain and use a connection with the JDBC 2.0 Optional Package API:

```
try {// Retrieve a DataSource through the JNDI Naming Service       java.util.Properties parms = new
java.util.Properties();   parms.setProperty(Context.INITIAL_CONTEXT_FACTORY,
"com.ibm.ejs.ns.jndi.CNInitialContextFactory");      // Create the Initial Naming Context
javax.naming.Context ctx = new javax.naming.InitialContext(parms);      // Lookup through the naming
service to retrieve a DataSource object   javax.sql.DataSource ds =
(javax.sql.DataSource)ctx.lookup("jdbc/SampleDB");     // Obtain a Connection from the DataSource
java.sql.Connection conn = ds.getConnection(); // query the database   java.sql.Statement stmt =
conn.createStatement();   java.sql.ResultSet rs =       stmt.executeQuery("SELECT EMPNO, FIRSTNME,
LASTNAME FROM SAMPLE");      // process the results   while (rs.next()) {      String empno =
rs.getString("EMPNO");      String firstnme = rs.getString("FIRSTNME");      String lastname =
rs.getString("LASTNAME");   // work with results   }} catch (java.sql.SQLException sqle) {// handle
SQLException} finally {   try {      if (rs != null) rs.close();   }   catch (java.sql.SQLException
sqle) {   // can ignore   }   try {      if (stmt != null) stmt.close();   }    catch
(java.sql.SQLException sqle) {   // can ignore   }   try {      if (conn != null) conn.close();   }
catch (SQLException sqle) {   // can ignore   }} // end finally
```

In the previous example, the first action is to retrieve a DataSource object from the JNDI namespace. This is done by creating a Properties object of parameters used to set up an InitialContext object. After a context is obtained, a lookup on the context is performed to find the specific datasource necessary, in this case, SampleDB.

(In this example, it is assumed the datasource has already been created and bound into JNDI by the WebSphere administrator. For information about doing thisin application code, see the Related information.)

After a DataSource object is obtained, the application code calls getConnection()on the datasource to get a Connection object. After the connection is acquired, the querying and processing steps are the same as for theJDBC 1.0 example.

4.2.4.2.1.1: Creating datasources with the WebSphere connection pooling API

IBM WebSphere Application Serverprovides a public API to enable you to configure a WebSphere datasource in application code.This is necessary only when the application must create a datasource on demand.Otherwise, the datasource is configured by the administrator in the administrative console.

The complete API specification can be found in javadoc for the class com.ibm.websphere.advanced.cm.factory.DataSourceFactory. See the Related information.

To create a datasource on demand in an application, the application must do the following:

1. Create a Properties object with datasource properties
2. Obtain a datasource from the factory
3. Bind the datasource into JNDI

The following code fragment shows how an application would create a datasource and bind it into JNDI:

```
import com.ibm.websphere.advanced.cm.factory.DataSourceFactory;...try {   // Create a properties
file for the DataSource   java.util.Properties prop = new java.util.Properties();
prop.put(DataSourceFactory.NAME, "SampleDB");   prop.put(DataSourceFactory.DATASOURCE_CLASS_NAME,
"COM.ibm.db2.jdbc.DB2ConnectionPoolDataSource");   prop.put(DataSourceFactory.DESCRIPTION, "My
sample                datasource");   prop.put("databaseName", "sample");// Obtain a DataSource from
the factory   DataSource ds = DataSourceFactory.getDataSource(prop);// Bind the DataSource into JNDI
DataSourceFactory.bindDataSource(ds);} catch (ClassNotFoundException cnfe) {// check the class path
for all necessary classes} catch (CMFactoryException cmfe) {// Example of exception: incorrect
properties} catch (NamingException ne) {// Example of exception:  datasource by this name may
already exist}
```

To create a datasource for binding into JNDI, the application must firstcreate a Properties object to hold the DataSource configuration properties.The only properties required for the datasource from a WebSphere perspective are:

- NAME -The name of the datasource. This is used to identify the datasource when it is bound into JNDI.
- DATASOURCE_CLASS_NAME - The complete name of the DataSource class that can be found in the JDBC resource archive file(often referred to as the JDBC driver package). This DataSource class will be used to create connections to the database. The class specified here must implement javax.sql.ConnectionPoolDataSource or javax.sql.XADataSource.

However, depending on the DataSource class specified in the DATASOURCE_CLASS_NAME property, there may be other vendor-specific properties required. In this example, the databaseName property is also required,because DB2ConnectionPoolDataSource is being used. For more information on these vendor-specific properties, see the vendor's documentation for the complete list of properties supported for a datasource.

After a properties object is created, the application can create a new DataSource object by calling getDataSource() on the factory, passing in the Properties object as a parameter. This creates an object of type DataSource, but it is not yet bound into JNDI. To bind a datasource into JNDI,call bindDataSource() on the factory.At this point, other applications can share the datasource by retrieving it from JNDI with the name property specified on creation.

All other APIs specific to connection pooling are not public APIs. Applications that use a WebSphere datasource should followthe JDBC 2.0 Core and JDBC 2.0 Optional Package APIs.

4.2.4.2.1.2: Tips for using connection pooling

Most best practices have been documented elsewhere in Related information.The following are additional items that have not been explicitly called out:

Obtain and close connection in the same method. An application should obtain and close its connection in the method that requires the connection. This keeps the application from holding resources not being used and leaves more available connections in the pool for other applications. In addition, this practice removes the temptation to use the same connection in multiple transactions, which, by default, is not allowed. This practice does not cost the application much in performance,because the Connection object is from a pool of connections, where the overhead for establishing the connection has already been incurred.Lastly, make sure to declare the Connection object in the same method as the getConnection() call in a servlet;otherwise, the Connection object works as if it is a static variable(see "Worst Practices" later in this article for problems with this).

If you opened it, close it. All JDBC resources that have been obtained by an application should be explicitly closed by that application. The product tries to clean up JDBC resources on a connection after the connection has been closed. However, this behavior should not be relied upon, especially if the application might be migrated to another platform in the future.

For servlets, obtain DataSource in the init() method. For performance reasons, it is usually a good idea to put the JNDI lookup for the datasource into the init() method of the servlet. Because the datasource is simply a factory for connections that does not typically change,retrieving it in this method ensures that the lookup happens only once.

Worst practices

The following are some very common problems with applications that should be avoided, because they most often result in unexpected failures:

Do not close connections in a finalize() method. If an application waits to close a connection or other JDBC resource until the finalize() method, the connection is not closed until the object that obtained it is garbage-collected. This leads to problems if the application is not very thorough about closing its JDBC resources, such as ResultSet objects. Databases can quickly run out of the memory required to store the information about all of the JDBC resources it currently has open. In addition, the pool can quickly run out of connections to service other requests.

Do not declare connections as static objects. It is never recommended that connections be declared as static objects. If a connection is declared as static, the same connection might get used on different threads at the same time. This causes a great deal of difficulty, within both the product and the database.

In servlets, do not declare Connection objects as instance variables. In a servlet, all variables declared as instance variables act as if they are class variables. For example, in a servlet with an instance variable

```
Connection conn = null;
```

this variable acts as if it were static. In this case, all instances of the servlet use the same Connection object. This is because a single servlet instance can be used to serve multiple Web requests in different threads.

In CMP beans, do not manage data access. If a Container Managed Persistence (CMP) bean is writtenso that it manages its own data access, this data access may be part of a global transaction. Generally, if specialized data access is required,use a BMP session bean.

4.2.4.2.1.3: Handling data access exceptions

For data access, the standard Java exception class to catch is java.sql.SQLException.IBM WebSphere Application Servermonitors for specific SQL exceptions thrown from the database. Several of these exceptions are mapped to WebSphere-specific exceptions. The product provides WebSphere-specific exceptions to ease development by not requiring you to know all of the database-specific SQL exceptions that could be thrown in typical situations. In addition, monitoring SQL exceptions enables the product and application to recover from common problems like intermittent network or database outages.

ConnectionWaitTimeoutException

This exception (com.ibm.ejs.cm.pool.ConnectionWaitTimeoutException) indicates that the application has waited for the connectionTimeout (CONN_TIMEOUT) number of seconds and has not been returned a connection. This can occur when the pool is at its maximum size and all of the connections are in use by other applications for the duration of the wait. In addition, there are no connections currently in use that the application can share, because either the user ID and password are differentor it is in a different transaction.The following code fragment shows how to use this exception:

```
java.sql.Connection conn = null;javax.sql.DataSource ds = null;...try {// Retrieve a DataSource
through the JNDI Naming Service       java.util.Properties parms = new java.util.Properties();
setProperty.put(Context.INITIAL_CONTEXT_FACTORY,
"com.ibm.websphere.naming.WsnInitialContextFactory");      // Create the Initial Naming Context
javax.naming.Context ctx = new       javax.naming.InitialContext(parms);      // Lookup through the
naming service to retrieve a DataSource object   javax.sql.DataSource ds =
(javax.sql.DataSource)ctx.lookup("jdbc/SampleDB");   conn = ds.getConnection();    // work on
connection} catch (com.ibm.ejs.cm.pool.ConnectionWaitTimeoutException cw) {// notify the user that
the system could not provide a // connection to the database}  catch (java.sql.SQLException sqle)
{// deal with exception}
```

In all cases in which the ConnectionWaitTimeoutException is caught, there is very little to do in terms of recovery. It usually doesn't make sense to retry the getConnection() method, because if a longer wait time is required, the connectionTimeout datasource property should be set higher. Therefore, if this exception is caught by the application, the administrator should review the expected usage of the application and tune the connection pool and the database accordingly.

StaleConnectionException

This exception (com.ibm.ejs.cm.portability.StaleConnectionException)indicates that the connection currently being held is no longer valid. This can occur for numerous reasons, including:

- The application fails to get a connectionbecause of a problem such as the database not being started.
- A connection is no longer usable because of a database failure. When an application tries to use a connection it previously obtained, the connection is no longer valid. In this case, all connections currently in use by the application may prompt this exception.
- The application using the connection has already called close() and then tries to use the connection again.
- The connection has been orphaned, andthe application tries to use the orphaned connection.
- The application tries to use a JDBC resource, such as Statement, obtained on a now-stale connection.

When application code catches StaleConnectionException, it should take explicit steps to handle the exception. StaleConnectionException extends SQLException, so it can be thrown from any method that is declared to throw SQLException. The most common occasion for a StaleConnectionException to be thrown is the first time a connection is used, just after it has been retrieved. Because connections are pooled, a database failure is not detected until the operation immediately following its retrieval from the pool, which is the first time communication with the database is attempted. It is only when a failure is detected thatthe connection is marked stale. StaleConnectionException occurs less often if each method that accesses the database gets a new connection from the pool. Typically, this occurs because all connections currently allocatedto an application are marked stale; the more connections the application has, the more connections can be stale.

Generally,when a StaleConnectionException is caught, the transaction in which the connection was involvedneeds to be rolled back and a new transaction begun with a new connection.

For more information and detailed code samples,be sure to read the IBM whitepaper to be published on the Webduring the summer of 2001.

4.2.4.2.2: Accessing data with the JDBC 1.0 reference model

The reference model that uses the JDBC 1.0 APIs, which still work under JDBC 2.0 and Application Server Version 3.x, is based on the code fragments shown in the following steps:

1. Load the driver for a specific relational database product. The specific driver class should be available from the WebSphere administrator.

   This step is typically performed once, during the init() method of the servlet.

   ```
   Class.forName("COM.ibm.db2.jdbc.app.DB2Driver");
   ```
2. Use the static getConnection() method of the DriverManager class to get a JDBC connection to the relational database product, again using parameters for the specific database product. The WebSphere administrator can provide the subprotocol, database, user ID, and password information.

   This step is performed for each client request made to the servlet, typically in the doGet() or doPost() method. (The subprotocol and database information are combined into what is called the database URL, shown as "jdbc:subprotocol:database" in the following code.)

   ```
   Connection conn =  DriverManager.getConnection("jdbc:subprotocol:database",     // database URL
   "userid",                              "password");
   ```
3. Given the connection, do the necessary data server interactions for each client request. This step is typically performed in the doGet() or doPost() method.
4. At the end of each client request, free the connection resource. This step is typically performed at the end of the doGet() or doPost() method.

   ```
   conn.close();
   ```

4.2.4.2.3: Accessing relational databases with the IBM data access beans

Java programs that access JDBC-compliant relational databases typically use the classes and methods in the java.sql package to access data. Instead of using the java.sql package, you can use the classes and methods in the package com.ibm.db, the IBM data access beans. This gives you additional features for data access beyond those available in the java.sql package.

The Related information discusses what the data access beans are, their advantages, and how to use them. A data access bean uses a connection that you provide to it, such as a connection from a connection pool that you get through a DataSource object.

4.2.4.2.3.1: Example: Servlet using data access beans

The sample servlet uses the data access beans and is based on the sample servlet discussed in Article 4.2.4.2.1.1. The connection pooling sample servlet uses classes such as Connection, Statement, and ResultSet from the java.sql package to interact with the database. In contrast, this sample servlet uses the data access beans, instead of the classes in the java.sql package, to interact with the database. For convenience, call this sample servlet the DA (for data access beans) and call the sample servlet on which it is based the CP (for connection pooling).

The CP and DA sample servlets benefit from the performance and resource management enhancements made possible by connection pooling. The programmer coding the DA sample servlet benefits from the additional features and functions provided by the data access beans.

The DA sample servlet differs slightly from the CP sample servlet. This discussion covers only the changes. See Article 4.2.4.2.1.1 for the discussion of the CP sample servlet. The DA sample servlet shows the basics of connection pooling and the data access beans, but keeps other code to a minimum. Therefore, the servlet is not entirely realistic. You are expected to be familiar with basic servlet and JDBC coding.

The changes

This section describes how the DA sample servlet differs from the CP sample servlet. To view the coding in one or both of the samples while you read this section, click these links:

- DA sample
- CP sample

Steps 1 through 6 of the CP sample servlet are mostly unchanged in the DA sample servlet. The main changes to the DA sample servlet are:

- New package

  The com.ibm.db package (containing the data access beans classes) must be imported. The classes are in the databeans.jar file, found in the lib directory under the Application Server root install directory. You will need this jar file in your CLASSPATH in order to compile a servlet using the data access beans.

- The metaData variable

  This variable is declared in the Variables section at the start of the code, outside of all methods. This allows a single instance to be used by all incoming user requests. The full specification of the variable is completed in the init() method.

- The init() method

  New code has been appended to the init() method to do a one-time initialization on the metaData object when the servlet is first loaded. The new code begins by creating the base query object sqlQuery as a String object. Note the two "?" parameter placeholders. The sqlQuery object specifies the base query within the metaData object. Finally, the metaData object is provided higher levels of data (metadata), in addition to the base query, that will help with running the query and working with the results. The code sample shows:

  - The addParameter() method notes that when running the query, the parameter idParm is supplied as a Java Integer datatype, for the convenience of the servlet, but that idParm should be converted (through the metaData object) to do a query on the SMALLINT relational datatype of the underlying relational data when running the query.

    A similar use of the addParameter() method for the deptParm parameter notes that for the same underlying SMALLINT relational datatype, the second parameter will exist as a different Java datatype within the servlet - as a String rather than as an Integer. Thus parameters can be Java datatypes convenient for the Java application and can automatically be converted by the metaData object to be consistent with the required relational datatype when the query is run.

    Note that the "?" parameter placeholders in the sqlQuery object and the addParameter() methods are related. The first addParameter() attaches idParm to the first "?", and so on. Later, a setParameter() will use idParm as an argument to replace the first "?" in the sqlQuery object with an actual value.

    - The addColumn() method performs a function somewhat similar to the addParameter() method. For each column of data to be retrieved from the relational table, the addColumn() method maps a relational datatype to the Java datatype most convenient for use within the Java application. The mapping is used when reading data out of the result cache and when making changes to the cache (and then to the underlying relational table).
    - The addTable() method explicitly specifies the underlying relational table. This information is needed if changes to the result cache are to be propagated to the underlying relational table.

- Step 5

  Step 5 has been rewritten to use the data access beans to do the SQL query instead of the classes in the java.sql package. The query is run using the selectStatement object, which is a SelectStatement data access bean.

  Step 5 is part of the process of responding to the user request. When steps 1 through 4 have run, the conn Connection object from the connection pool is available for use. The code shows:

    1. The dataAccessConn object (a DatabaseConnection bean) is created to establish the link between the data access beans and the database connection - the conn object.
    2. The selectStatement object (a SelectStatement bean) is created, pointing to the database connection through the dataAccessConn object, and pointing to the query through the metaData object.
    3. The query is "completed" by specifying the parameters using the setParameter() method. The "?" placeholders in the sqlQuery string are replaced with the parameter values specified.
    4. The query is executed using the execute() method.
    5. The result object (a SelectResult bean) is a cache containing the results of the query, created using the getResult() method.
    6. The data access beans offer a rich set of features for working with the result cache - at this point the code shows how the first row of the result cache (and the underlying relational table) can be updated using standard Java coding, without the need for SQL syntax.
    7. The close() method on the result cache breaks the link between the result cache and the underlying relational table, but the data in the result cache is still available for local access within the servlet. After the close(), the database connection is unnecessary. Step 6 (which is unchanged from the CP sample servlet) closes the database connection (in reality, the connection remains open but is returned to the connection pool for use by another servlet request).

- Step 7

  Step 7 has been entirely rewritten (with respect to the CP sample servlet) to use the query result cache retrieved in Step 5 to prepare a response to the user. The query result cache is a SelectResult data access bean.

  Although the result cache is no longer linked to the underlying relational table, the cache can still be accessed for local processing. In this step, the response is prepared and sent back to the user. The code shows the following:

    - The nextRow() and previousRow() methods are used to navigate through the result cache.

    Additional navigation methods are available.

    - The getColumnValue() method is used to retrieve data from the result cache. Because of properties set earlier in creating the metaData object, the data can be easily cast to formats convenient for the needs of the servlet.

A possible simplification

If you do not need to updatethe relational table, you can simplify the sample servlet:

- At the end of the init() method, you can drop the lines with the addColumn() and addTable() methods, since the metaData object does not need to know as much if there are no relational table updates.
- You will also need to drop the lines with the setColumnValue() and updateRow() methods at the end of step 5, because you are no longer updating the relational table.
- Finally, you can remove most of the type casts associated with the getColumnValue() methods in step 7. You will, however, need to change the type cast to (Short) for the "ID" and "DEPT" use of the getColumnValue() method.

4.2.4.2.4: Database access by servlets and JSP files

Servlets using getConnection() to access a data source

When used without parameters, getConnection() assumes the default user ID and password for a data source. The WebSphere administrative clients do not offer a way to configure a default user ID and password for a data source to be used by a servlet.

Therefore, servlets using getConnection() to access a data source should specify a user ID and password:

```
getConnection(userid,password);
```

6.6.14: Administering database connections (overview)

The WebSphere administrator has an important role in establishingand maintaining connection pools through data sources and drivers:

- Configuring the resources used in connection pooling -- JDBC drivers and data sources

  The administrator needs to configure data sources and JDBC drivers for each brand and version of database from which enterprise applications or individual resources will require connections.

- Adjusting connection pooling parameters for optimal performance.

  Connection pools provide a way to share the connection overhead among multiple requests, but it is possible to configure too large a connection pool. The administrator needs to determine the optimal value for the pool size and other settings, based on environmental factors such as the operating system in use.

JDBC Drivers and data sources are used by Java components requiringdatabase access, such asservlets.

Procedure for introducing JDBC drivers

To introduce JDBC drivers into the WebSphere Application Server environment:

1. Create a WebSphere JDBC driver configuration that specifies where tofind the driver code on the physical machine.
2. Create a WebSphere data source configuration.
3. Perform an administrative "installation" of the JDBC driver into WebSphere.

Use the administrative client of your choice to perform the above steps (seeRelated information). The Create Data Source task wizard inthe Java administrative console is recommended because it leads you through all of the above steps.

6.6.14.0: Properties of JDBC drivers: WebSphere Application Server

Description

A description of the driver, for your administrative records.

Driver Implementation Class, Implementation Class

Specifies the name of the Java class that implements the driver.

Jar file, Classpath

Specifies the path to the JAR file containing the implmentation code for the database driver, such as db2java.zip file for DB2.

See the reference below for a list of default locations.

JTA Enabled

Specifies whether the driver can handle Java-based two-phase commit transactions. JTA is a transaction API for Java applications.

If **not** performing distributed transactions, set this value to False.

Note that for Application Server Version 3.0x, the URL Prefix setting had to contain "jta," in addition to selecting JTA Enabled. For Version 3.5, selecting JTA Enabled is sufficient. The URL Prefix value is independent of whether JTA is enabled.

Name

Specifies a name for the driver. It is recommended you enter a name that is suggestive of the database product you are using, such as DB2JdbcDriver.

Node

Specifies the node (machine) on which to install the driver.

URL prefix

Specifies the URL prefix with which this driver is associated. The URL prefix is comprised of the protocol and subprotocol, separated by a colon (":").

The Database Name of the data source is appended to the URL prefix to produce the full JDBC URL of the database, such as jdbc:db2:was.

Locating JDBC providers on your operating system

The following table lists the default locations of JDBC provider files. Seethe product prerequisites Web site for the most up-to-date information on the operating system brands and databases supported by IBM WebSphere Application Server. Column 1 in the table lists the operating system, and column 2 shows a list of the drivers that are available for use with each database supported for the operating system.

Operating system	Drivers

AIX	● DB2: *$DB2_HOME*/java12/db2java.zip or *$DB2_HOME*/java/db2java.zip ● Oracle: *$ORACLE_HOME*/jdbc/lib/classes12.zip ● Sybase: *sybase_install_root*/jConnect-5_2/classes/jconn2.jar ● InstantDB: *product_installation_root*/lib/idb.jar FP 3.5.3 Merant SequelLink: ❍ Complimentary copies with WebSphere Application Server: ■ *product_installation_root*/lib/sljc.jar ■ *product_installation_root*/lib/sljcx.jar
HP-UX	● DB2: *$DB2_HOME*/java12/db2java.zip or *$DB2_HOME*/java/db2java.zip ● Oracle: *$ORACLE_HOME*/jdbc/lib/classes12.zip ● InstantDB: *product_installation_root*/lib/idb.jar ● FP 3.5.3 Merant SequelLink: ❍ Complimentary copies with WebSphere Application Server: ■ *product_installation_root*/lib/sljc.jar ■ *product_installation_root*/lib/sljcx.jar
Linux (Intel)	● DB2: *$DB2_HOME*/java12/db2java.zip or *$DB2_HOME*/java/db2java.zip ● Oracle: *$ORACLE_HOME*/jdbc/lib/classes12.zip ● InstantDB: *product_installation_root*/lib/idb.jar
Linux on S/390	● Oracle: *$ORACLE_HOME*/jdbc/lib/classes12.zip ● InstantDB: *product_installation_root*/lib/idb.jar
NetWare	● DB2: *$DB2_HOME*/lib/db2java.zip ● Oracle: *$ORACLE_HOME*/jdbc/lib/classes12.zip ● InstantDB: *product_installation_root*/lib/idb.jar
Solaris	● DB2: *$DB2_HOME*/java12/db2java.zip or *$DB2_HOME*/java/db2java.zip ● Oracle: *$ORACLE_HOME*/jdbc/lib/classes12.zip ● Sybase: *sybase_install_root*/jConnect-5_2/classes/jconn2.jar ● InstantDB: *product_installation_root*/lib/idb.jar ● FP 3.5.3 Merant SequelLink: ❍ Complimentary copies with WebSphere Application Server: ■ *product_installation_root*/lib/sljc.jar ■ *product_installation_root*/lib/sljcx.jar

Windows	- DB2: C:\SQLLIB\java\db2java.zip - Oracle: C:\Oracle\Ora81\jdbc\lib\classes12.zip - InstantDB: *product_installation_root*/lib/idb.jar - FP 3.5.3 Merant SequelLink: - Complimentary copies with WebSphere Application Server: - *product_installation_root*\lib\sljc.jar - *product_installation_root*\lib\sljcx.jar Note that the same data source implementation class is used for both non JTA and JTA enabled datasources. Also, this class is found in the sljcx.jar, not the sljc.jar.

6.6.14.0.1: Properties of data sources

Connection timeout

Specifies the maximum time in seconds that requests for a connection wait if the maximum number of connections is reached and all connections are in use.

This value must be a positive integer.

Database Name

Specifies the name of the database used to store entity bean data.

For example, enter WAS. This would make your data source point to jdbc:db2:WAS.

Driver

Specifies the name of the JDBC driver that this data source is using.

Idle timeout

Specifies the maximum time in seconds that an idle (unallocated) connection can remain in the pool before being removed to free resources.

This value must be a positive integer.

JNDI Name, JNDI Binding Path

The JNDI name for the resource, including any naming subcontexts. This name is used as the linkage between the platform's binding information for resources defined in the a client applications deployment descriptor and actual resources bound into JNDI by the platform.

Maximum connection pool size

Specifies the maximum number of connections that can be in the pool. If the maximum number of connections is reached and all connections are in use, additional requests for a connection wait up to the number of seconds specified in the Connection timeout property.

This value must be a positive integer.

Minimum connection pool size

Specifies the minimum number of connections in the pool.

This value must be a positive integer.

Name

Specifies a name by which to administer the data source.

It is recommended that you enter a name that is suggestive of the database you will use to store entity bean data, such as WASDataSource, where WAS is the database name. The JNDI lookup for such a data source would be jdbc/WASDataSource.

Orphan timeout

Specifies the maximum time in seconds that an inactive (but allocated) connection can remain in the pool before being removed.

This value must be a positive integer.

⚠ The administrative server and console canexhibit odd behavior in response to database URLs that are not valid. When youuse a database URL, ensure you have typed it correctly.

6.6.14.1: Administering database connections with the Java console

This article extends article 6.6.14 (the overview of administering database connections) with information specific to the Java console. As discussed in 6.6.14, database connections available to applicationsare represented by administrative "JDBC driver" and "data source" configurations.

The table answers the most basic questions. See the related informationfor links to detailed instructions and resource properties.

Does the console provide full functionality for administering this resource?	Yes
How is this resource representedin the console tree views?	The Type tree contains JDBC Drivers and Data Sources folder objects. The Topology tree can contain zero or moreexisting JDBC drivers and data sources. Their names vary;they are supplied by the administrator. Use the View menu on the console menu bar to toggle between tree views.
Any task wizards for manipulatingthis resource?	On the console menu bar: Console -> Task -> Create Data Source

6.6.14.1.1: Configuring new database connections with the Java administrative console

The product offers a couple of approaches to configuring JDBC driversand data sources, the resources necessary to allow applications to connect to databases:

- By clicking Console -> Tasks -> Create Data Source from theconsole menu bar.
- Using menus on resources in the Topology and Type trees (see Related information)

The first two methods lead to the Create Data Sourcetask wizard, for which detailed help is provided here.

1. Follow the wizard instructions. On the first page, specifywhether to use a JDBC driver you have already configured, orto configure a new one as part of the wizard.
2. Click Next to proceed. If you specified to use anexisting JDBC driver, skip to the next step in these instructions.Otherwise, complete the next two panels to configure a newJDBC driver:
    - Specify JDBC driver properties, including a name by which to administer the driver.

      View data access properties help
    - Select a node on which the driver will reside. Then Browse for the JAR file containing the driver code.

   Click Next to proceed. Note, the wizard will not let you proceed until the "Jar file"field contains a value.
3. Specify a name by which to administer the data source. Specifythe name of the database to which the data source should provideconnections.

   View data access properties help
4. Click Finish.

6.6.14.3: Administering database connections with the Web console

Use the Web console to edit the configurations of JDBC drivers and data sources, which are used byyour installed applications to access data fromdatabases. You can create, modify, and install JDBC drivers. You can create andmodify data sources.

Click **Tasks** -> **Create Objects** -> **Create JDBC Driver** to create a new JDBC driver.

Click **Tasks** -> **Create Objects** -> **Create Data Source** to create a new data source.

In order to create a data source, a driver must already exist with which you can associate the data source. Existing JDBC drivers and data sources in the administrative domain are displayed in the **Resources** section of the navigation tree.

⚠ Creations and changes made with this console are not appliedto the administrative domain until you Commit them. Refer to section 6.6.0.3.5 for details.

6.6.14.4: Property files pertaining to database connections

The database connection properties are in file:

- *admin.config*

This file is located in directory:

`<WebSphere/Appserver>/bin`

The following entries in the ***admin.config*** file apply to database connections:

com.ibm.ejs.sm.adminServer.dbUrl	URL for JDBC access
com.ibm.ejs.sm.util.adminServer.dbSchema	database schema name
com.ibm.ejs.sm.adminServer.dbDriver	Classname of JDBC driver
com.ibm.ejs.sm.adminServer.dbPassword	Password for database access
com.ibm.ejs.sm.adminServer.dbUser	User ID for database access

See the Related information for a description of database connections and their properties.

6.6.14.5: Additional administrative tasks for specific databases

For your convenience, this article provides instructions for enabling some popular database drivers, and performing other administrative tasks often required in order to provide dataaccess to applications running on WebSphere Application Server. These tasks are performedoutside of the WebSphere Application Server administrative tools, often using the databaseproduct tools. Always refer to the documentation accompanying your database driver as the authoritative andcomplete source of driver information.

See the WebSphere Application Server product prerequisites for the latest information about supported databases, drivers, and operating systems.

- Enabling JDBC 2.0
  - For DB2 on Windows NT
  - For DB2 on UNIX
- Sourcing the db2profile script on UNIX
- Using JTA drivers
  - For DB2 on Windows NT
  - For DB2 on UNIX
- For Oracle via Merant SequeLink 5.1 on Windows NT
- For Sybase on AIX
- Tips for selecting JDBC drivers

Enabling JDBC 2.0

Ensure that your operating system environment is set up to enable JDBC 2.0 use.This is required in order to use data sources created through WebSphere ApplicationServer.

The following steps make it possible to find the appropriate JDBC 2.0 driverfor use with WebSphere Application Server administration:

Enabling JDBC 2.0 with DB2 on Windows NT

To enable JDBC 2.0 use on Windows NT systems:

1. Stop the DB2 JDBC Applet Server service.
2. Run the following batch file:
   `SQLLIB\java12\usejdbc2.bat`
3. Stop WebSphere Application Server (if it is running) andstart it again.

Perform the steps once for each system.

Determining the level of JDBC in use for DB2 on Windows NT

To see whether the JDBC level in use on your system:

- If JDBC 2.0 is in use, this file will exist:
  `SQLLIB\java12\inuse`
- If JDBC 1.0 is in use, this file will exist:
  `SQLLIB\java11\inuse`
  or there will be no java11 directory

Enabling JDBC 2.0 with DB2 on UNIX

Before starting WebSphere Application Server, you need to call $INSTHOME/sqllib/java12/usejdbc2 to use JDBC 2.0. For convenience, you might want to put this in your root user's startup script.For example on AIX, add the following to the root user's .profile:

```
if [ -f /usr/lpp/db2_07_01/java12/usejdbc2 ] ; then    . /usr/lpp/db2_07_01/java12/usejdbc2fi
```

Determining the level of JDBC in use for DB2 on your UNIX system

To determine if you are using JDBC 2.0, you can echo $CLASSPATH. If it contains

`$INSTHOME/sqllib/java12/db2java.zip`

then JDBC 2.0 is in use.

If it contains

`$INSTHOME/sqllib/java/db2java.zip`

then JDBC 1.0 is in use.

Sourcing the db2profile script on UNIX

Before starting WebSphere Application Server to host applications requiringdata access, source the db2profile:

```
.~db2inst1/sqllib/db2profile
```

where *db2inst1* is the user created during DB2 installation.

Using JTA drivers

Instructions are available for using JTA drivers on particular operatingsystems. See your operating system's documentation for more information.

Using JTA drivers for DB2 on Windows NT

To enable JTA drivers for DB2 on Windows NT, follow these steps:

1. Stop all DB2 services.
2. Stop the IBM WebSphere Application Server administrative service.
3. Stop any other processes that use the db2java.zip file. (Note: If the **JVIEW** process is active, you must use the **Task Manager** utility to stop it.)
4. Make sure that you already enabled JDBC 2.0.
5. Start the DB2 services.
6. Configure DB2 to use JTS as the transaction processing (TP) monitor. From the DB2 Control Center, follow these steps:
    a. Right-click the DB2 instance that contains the database that is to be enabled for JTA access.
    b. Click **Multisite Update**, **Configure** to start the Smartguide utility.
    c. Click the **Use the TP monitor named below** radio button.
    d. Select **JTS** as the TP monitor.
    e. Click **Done**.
7. Bind the necessary packages to the database. From the **DB2 Command Line Processor** window, issue the following commands:

```
db2=> connect to mydb2jtadb2=> bind db2home\bnd\@db2cli.lstdb2=> bind db2home\bnd\@db2ubind.lstdb2=>
disconnect mydb2jta
```

   where *mydb2jta* is the name of the database that is to be JTA enabled, and *db2home* is the DB2 root installation directory path (for example, D:\ProgramFiles\SQLLIB\bnd\@db2cli.lst).
8. When you use an IBM WebSphere Application Server administrative client (suchas the WebSphere Administrative Console) to configure a JDBC driver, specifythe following settings:
    - **Class name** = `COM.ibm.db2.jdbc.app.DB2Driver`
    - **URL prefix** = `jdbc:jta:db2`
    - **JTA enabled** = `True`

Using JTA drivers for DB2 on UNIX

To enable JTA drivers on UNIX, follow these steps:

1. Stop all DB2 services.
2. Stop the IBM WebSphere Application Server administrative service.
3. Stop any other processes that use db2java.zip file.
4. Make sure that you already enabled JDBC 2.0.
5. Start the DB2 services.
6. When you use an IBM WebSphere Application Server administrative client (suchas the WebSphere Administrative Console) to configure a JDBC driver, specifythe following settings:
    - **Class name** = `COM.ibm.db2.jdbc.app.DB2Driver`
    - **URL prefix** = `jdbc:jta:db2`
    - **JTA enabled** = `True`

Using JTA drivers for Oracle via Merant SequeLink 5.1 on Windows NT

To enable JTA drivers for use with Oracle via SequeLink on the Windows NT operating system, follow these steps:

1. Setting up Oracle 8.1.6:
    - Use the following command at the Oracle command promt to create the user:

```
create user dbuser1 identified by dbpwd1 default tablespace users \temporary tablespace temp profile
default account unlock;
```

    - Ensure XA connectivity for the user by issuing the following command at the Oracle command prompt:

```
grant select on "SYS"."DBA_PENDING_TRANSACTIONS" to dbuser1;
```

2. Setting up direct database authentication through SequeLink Manager:
    - In **SequeLink Services**, **SLOracle51**, **Configuration**, **Service Settings**, and **User Security**, set **ServiceAuthMethods** to **Anonymous**.
    - In **SequeLink Services**, **SLOracle51**, **Configuration**, **Data Source Settings**, **Default**, and **User Security**, set **DataSourceLogonMethod** to *DBMSLogon(UserID,Password)*.
    - Because the XAOpen string, containing the user ID and password, is storedin the trace log, to ensure security, enter the following setting:In **SequeLink Services**, **SLOracle51**, **Configuration**, **Service Settings**, **Logging** set **ServiceDebugLogLevel**to either **Fatal** or **Errors**.
3. Setting up WebSphere Application Server:
    - When you create your new JDBC driver, the URL prefix defaults to the following:

```
jdbc:sequelink//hostname:19996
```

Change *hostname* to the name of your machine, and enter the port number you are using, represented here by the SequeLink default of *:19996*.

- When you create the data source, enter the Oracle environment variable ORACLE_SID in the Database Name field.

  Enter the Oracle environment variable ORACLE_SID.
- Select your server, and on the **General** tab, click **Environment**. Add the CLASSPATH variable, and set its value to the path in which the SequeLink Client is installed, for example:

  ```
  #$WAS_HOME/driver/lib/sljcx.jar
  ```

  where *$WAS_HOME* is the location where WebSphere Application Server is installed.

Using JTA drivers for Sybase on AIX

To enable JTA drivers for use with Sybase on the AIX operating system,follow these steps:

1. At a command prompt, enable the Data Transaction Manager (DTM) by issuing these commands (one per line):

   ```
   isql -Usa -Ppassword -Sservername       sp_configure "enable DTM", 1    go
   ```
2. Stop the Sybase Adaptive Server database and start it again.
3. At a command prompt, grant the appropriate role authorization to the EJB user:

   ```
   isql -Usa -Ppassword -Sservername       grant role dtm_tm_role to EJB                   go
   ```

Notes about Sybase JTA drivers

Do not use a Sybase JTA connection in an enterprise bean method with an unspecified transaction context. A Sybase JTA connection does not support the local transaction mode. The implication is that the Sybase JTA connection must be used in a global transaction context.

Tips for selecting JDBC drivers

Here are some tips for selecting JDBC drivers:

- For DB2, consider type 2 application drivers, which are typically fasterthan type 2 network drivers.
- DB2 6.1 on HPUX does not support JDBC 2.0.
- For Sybase, consider type 4 thin drivers.
- On Sybase, JDBC 2.0 support is provided by the jConnect 5.2 component
- For Oracle, consider type 4 thin drivers.

6.6.14.6: Notes about various databases

This article provides miscellaneous tips for using supported databases. See also the related links.

Alwaysconsult the product prerequisites Web site for a list of the database brands and versions that aresupported by your particular Application Server version, edition, and fix pack.

- Do not drop the administrative database while the administrative server is running.
- DB2 performance on a local machine can be improved by setting up a local database as a remote instance. On UNIX systems, this is required. The configuration uses TCP/IP instead of shared memory.

  Oracle and Sybase also support client/server connections. Consult their product documentation for specifics.
- To use SQLJ static queries with DB2/MVS,be sure to add sqlj.jar and runtime.jarto the application server classpath only.
- When using Sybase 11.x, you might encounter the following error when HttpSession persistence is enabled:

  ```
  DBPortability W Could not create database table: "sessions" com.sybase.jdbc2.jdbc.SybSQLException:
  The 'CREATE TABLE' command is notallowed within a multi-statement transaction in the 'database_name'
  database
  ```

  where 'database_name' is the name of the database for holding sessions.

  If you encounter the error, issue the following commands at the Sybase command line:

  ```
  use database_namegosp_dboption db,"ddl in tran ",truego
  ```
- Sybase 12.0 does not support local transaction modes with a JTA enabled data source. To usea connection from a JTA enabled data source in a local transaction, Sybase patch EBF9422 mustbe installed.

6.6.14.7: Notes about InstantDB

InstantDB limitations

InstantDB is provided only for running the Hit Count enterprise bean example, andthe WebSphere Samples Gallery.

Two instances of InstantDB databases are created when the product is installed:

- sampleDB (created in *product_installation_root*/bin)
- IDBDatabase *product_installation_root*/installedApps/Samples.ear)

⚠ The use of InstantDB in a development, test, or production environment (otherthan to run the WebSphere Samples) is not supported.

Do not use InstantDB in a production environment. Do not use InstantDB ina development or test environment that requires functionality beyond the limitations described below.

InstantDB does not support:

- Persistence of application data
- CMP function for enterprise beans
- Access by administrative clients on remote machines (unless using administrative server agent)
- Multinode function
- Cloning of application servers
- Redirection of application requests from Web servers on one machine to application servers on another
- Distributed transactions
- Workload manager persistence

InstantDB tracing

InstantDB places its tracing information in the file:

```
product_installation_root/bin/trace.log
```

where *product_installation_root* refers to the IBM WebSphereApplication Server installation root.

To turn logging off:

1. Open the following file in a text editor:*product_installation_root*/bin/was.prp
2. Locate the line:
   ```
   traceFile=./trace.log
   ```
3. Comment the line out:
   ```
   ! traceFile=./trace.log
   ```

6.6.14.10: Establishing database failover support with HACMP

This document describes a high availability database failover scenario for IBM WebSphere Application Server Advanced Edition using two AIX systems and a shared external disk that stores the databases used by WebSphere applications. The scenario uses an IBM product called High Availability Cluster Multi-Processing (HACMP).

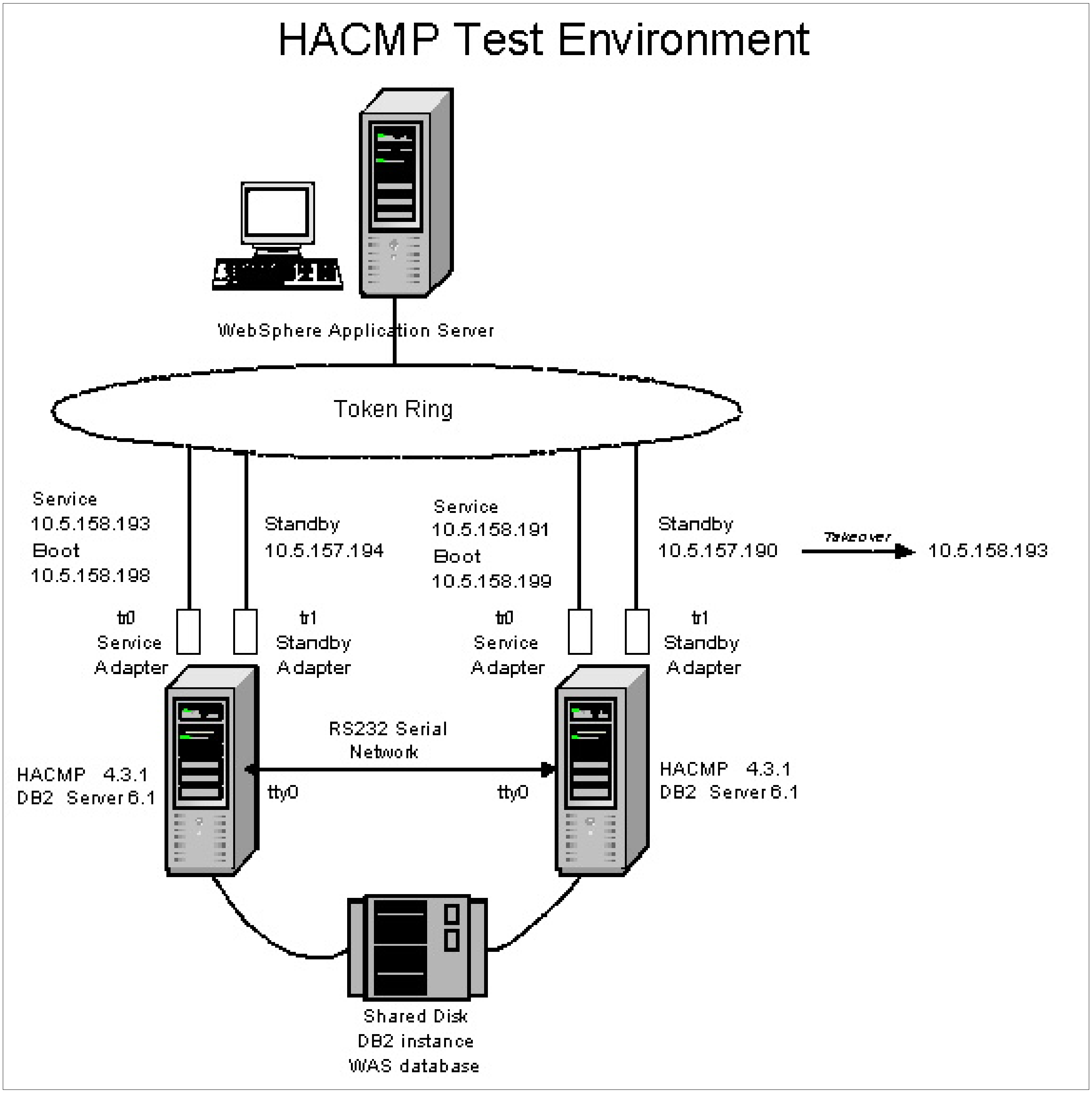
- Configuration overview
- Hardware and software
- Building a two node hot standby HACMP cluster
- WebSphere and Web server configuration options
- Performing a controlled failover to verify the configuration
- Current limitations and considerations

For some steps in the setup procedure, you are referred to related documents for more detailed instructions. If you need these documents, see the **Related information** at the bottom of this article for links.

Configuration overview

A cluster of two AIX systems can be used to build a high availability database environment. One AIX system is used as the primary DB2 server. The second system is used as the backup DB2 server system providing standby failover support for when the first system has a failure.

The cluster in the figure below demonstrates a two node **Hot Standby** configuration. It shows a common cascading scenario. Resources move to the second "hot standby" node if the primary node fails. The two RS/6000 machines run DB2 server software and the HACMP software.



The HACMP software is configured to control the DB2 software that accesses the databases located on the shared disk. By using a shared disk, either system can access the same databases, including the WebSphere administrative database (named "WAS" by default) and any other databases supporting applications managed by WebSphere Application Server.

This eliminates the first primary AIX system and its DB2 server software from being a single point of failure.

IBM WebSphere Application Server is installed on one or more other systems that will access the DB2 databases remotely over the network. Initially, it will connect through the first DB2 server system. When that first DB2 primary system has a failure, the backup system takes over as if it is the primary system and assumes the TCP/IP address of the first system.

When a failure occurs, HACMP will:

1. Detect the failure, such as a system, network, or application failure
2. Stop the DB2 server on the first system
3. Release the shared resources from the first system (disk volume groups)
4. Assume the service IP address on the standby adapter of the second system
5. Assign the shared resources to the second system
6. Start DB2 server on the second system

Hardware and software

In addition to IBM WebSphere Application Server, this scenario requires:

Software	Hardware
● Supported AIX version ● HACMP Version 4.3.1 ● Supported DB2 version	● 2 RS/6000 workstations ● 4 Token Ring 16/4 adapters ● 1 shared external disk configuration using Serial Storage Architecture (SSA) ● 1 IBM serial null modem cable

See the product prerequisites for information about supported software.

Building a two node hot standby HACMP cluster

The following procedure demonstrates how to set up a two node hot standby HACMP cluster. For more detail, consult the *HACMP for AIX - Installation Guide*.

1. Install network adapters in the cluster nodes

   Install two adapters on each node:
   - service/boot adapter
   - standby adapter.
2. Configure the network settings

   Configure the four adapters on the cluster nodes. First, configure the two standby adapters with the standby TCP/IP addresses.

   Second, configure the two service adapters with the boot addresses. The service addresses for the same two service adapters will be configured later by HACMP.

   Notice that the service adapter and standby adapters have to be on the *same* physical network but on *different* subnets. Some details are shown on the HACMP Test Environment figure in the configuration overview section.
3. Interconnect the workstations for HACMP

   Use the null modem cable to connect the two nodes through their serial ports. This serial connection will act as a private network between the two HACMP cluster nodes and will carry the "keep alive" packets between them without using the public TCP/IP network.

   Test the RS232 network by issuing the command:

   ```
   stty < /dev/ttyx
   ```

   on each system. The stty attributes should be displayed on both systems.

As an alternative to using the RS232 connection, if you use a SCSI device or SSA shared disk system, you can set the Target Mode SCSI/SSA connections to provide an alternative serial network. For details, see the *HACMP for AIX - Installation Guide*.

4. Install shared disk devices

   The application data for applications being managed by IBM WebSphere Application Server needs to be on a shared device that both nodes can access. It can be a shared disk or a network file system. The device itself should be mirrored or have data protection to avoid data corruption.

   The configuration described in the configuration overview depicts an IBM SSA Disk Subsystem for this purpose.

5. Define shared volume groups and file systems

   Creating the volume groups, logical volumes, and file systems shared by the nodes in an HACMP cluster requires steps to be performed on all nodes in the cluster.

   In general, define the components on one node and then import the volume group on the other nodes in the cluster. This ensures that the ODM definitions of the shared components are the same on all nodes in the cluster.

   Whether to define a Non-concurrent access or Concurrent access volume group depends on how you set up the cluster. In the hot standby configuration, a shared Non-Concurrent access volume group with a Journaled file system was used so that only one node can access the volume group and the file system at a time. HACMP will switch the resource from one node to the other node.

   To learn more about defining shared volume groups and file systems, see the *HACMP for AIX - Installation Guide* and your AIX documentation.

6. Install HACMP software

   Use "smit install_latest" to install:

   - cluster.adt
   - cluster.base
   - cluster.clvm
   - cluster.cspoc

   and related files on both nodes. HAView, a monitor tool, is not needed in the configuration.

7. Install DB2 server

   Install DB2 server on both nodes. The DB2 installation path can either be in a directory shared by both nodes or on a non-shared file system. When using a none-shared file system, the installation level must be identical.

8. Create DB2 instance

   Create a DB2 instance for the database. The DB2 instance path, as with the installation path can either be on a shared file system or on a manually mirrored file system. For the configuration discussed in the overview, the DB2 instance was created on the shared SSA disk system.

9. Confirm that the WAS database exists for the IBM WebSphere administrative server to use

   Make sure the WAS database exists. If not, create one. In either case, ensure that the application heap size (APPLHEAPSZ) of the database is set to 256.

   To manually create the WAS database and set the application heap size, execute these commands:

   ```
   db2 CREATE DATABASE wasdb2 UPDATE DB CFG FOR was USING APPLHEAPSZ 256
   ```

   If you later need to repeat the installation procedure, be sure to drop the WAS database before you install again. Use IBM DB2 Control Center or the following command to drop the database:

```
db2 DROP DATABASE WAS
```

10. Define the cluster topology

    Use "smit hacmp" to define clusters, cluster nodes, network adapters and network modules. In the configuration above, "cascade" was used so node 1 always has higher priority than node 2.

    For details, see the *HACMP for AIX - Installation Guide*.

11. Configure cluster resources

    In HACMP terms, an "application server" is a cluster resource made highly available by the HACMP software. In the configuration shown in the overview section, the DB2 instance on the shared disk is the "application server."

    An application server has a start script and a stop script. The start script starts the application server. The stop script stops the application server so that the application resource can be released, allowing the second node to take it over and restart the application.

Sample start script (start.sh):	`db2start`
Sample stop script (stop.sh):	`db2 force application all; db2stop`
Sample start usage:	`su - db2inst1   start.sh`
Sample stop usage:	`su - db2inst1   stop.sh`

    Create the start and stop scripts for both cluster nodes. Configure the application server with the path to them.

12. Start the HACMP cluster

    Start HACMP on the first node. Start HACMP on the second node after the start is complete on the first node. Use the /tmp/cm.log file to monitor the cluster events.

WebSphere and Web server configuration options

Now you can install and configure WebSphere and a Web server on other systems. You can set up the front end systems in a variety of ways. It is most simple to use just one system to run both a Web server and WebSphere Version 3.02. This is appropriate for a test environment.

1. Install the DB2 client software
2. Configure the DB2 client to connect to the remote WAS database
3. Install the Web server and WebSphere Application Server.

   When the Application Server installation prompts you for information about the administrative database, specify the previously configured remote database. (Alternatively, you can change the database setting by modifying the "com.ibm.ejs.sm.adminServer.dbUrl" directive in the admin.config file).

Performing a controlled failover to verify the configuration

1. Start HACMP on the first node. Monitor the /tmp/cm.log file for completion messages.
2. Start HACMP on the second node.
3. On the WebSphere system, start the WebSphere administrative server. Monitor the /logs/tracefile for the message:

   ```
   WebSphere Administration server open for e-business
   ```

4. Start the WebSphere administrative console and any application servers.

5. Initiate a controlled failover on the first node:
    1. Issue the command:

       ```
       smit hacmp
       ```

    2. From the menus, select Cluster Services -> Stop Cluster Services -> Stop now with "Shutdown mode - graceful with takeover." You can monitor the "/tmp/cm.log" on both systems to watch the progress.
6. When the failover is complete, refresh or start a new administrative console. Check the topology to see that the servers are functional.

Current limitations and considerations

During times when there is no active database connection, such as when the remote database server is stopped, and soon after the database connection is reestablished, using a Java administrative console (WebSphere Administrative Console) will produce some warning and error messages. The most common messages are:

```
getAttributeFailureFailed to roll back ... Connection is closed
```

In such circumstances, it is recommended that you close the console and start a new one.

6.6.14.11: Recovering from data source configuration problems using the XAResources file

When a WebSphere application server starts, the product stores information about any JTA-enabled data sources used by the server. This information is needed to recover from server failures. The information is kept in the file:

`product_installation_root/properties/application_server_nameXAResources`

Every time the server comes up, the server runtime examines thisfile and attempts any needed recovery on the resources describedin this file.

In the event that a JTA-enabled DataSource is configured incorrectly, you might see server startup failures orwarning messages due to the incorrectly-configured resources.

For example, you might see the warning message:

```
WTRN0025W: Can not create XAResource object
```

There is a work-around for this problem, *if* you are sure that you do not have other JTA-enabled data sources on this server for which recovery needs to be driven. You can simply removethe *application_server_name*XAResources file from the properties directory, correct the configuration error, and attempt to start the server again. The XAResources file will be re-created when theserver starts.