JD Edwards World

Java Database Connectivity 3.1 Guide

Release A9.4

E58808-01

April 2015

Describes how to use the JD Edwards World JDBC driver to allow non-JD Edwards World applications to access JD Edwards World data while maintaining the level of security and the flexibility built into the JD Edwards World software.

ORACLE

JD Edwards World Java Database Connectivity 3.1 Guide, Release A9.4

E58808-01

Contents

3 World JDBC Driver Administration

4 World JDBC Driver Data Features

Index

Preface

Welcome to the JD Edwards World Java Database Connectivity 3.1 Guide.

Audience

This guide is intended for implementers and end users of JD Edwards World Java Database Connectivity driver.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

Related Information

For additional information about JD Edwards World applications, features, content, and training, visit the JD Edwards World pages on the JD Edwards Resource Library located at:

http://learnjde.com

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Indicates cautionary information or terms defined in the glossary.
italic	Indicates book titles or emphasis.

1

World JDBC Driver Overview

This chapter contains these topics:

- Section 1.1, "Java Database Connectivity,"
- Section 1.2, "Terms."

1.1 Java Database Connectivity

The JD Edwards World JDBC driver (JDBC driver) provides the ability for non-JD Edwards World applications to access JD Edwards World data while maintaining the level of security and the flexibility built into the JD Edwards World software.

While the JD Edwards World JDBC driver adheres to the Java™ JDBC™ 4.0 specification interface, it is not a completely functional JDBC driver. The JDBC driver does provide great flexibility in selecting, formatting, and presenting JD Edwards World data to outside applications. The JDBC driver provides the features listed in this section.

1.1.1 Security Features

The JDBC driver provides the following security features for the user profile that logs into JD Edwards World via JD Edwards World JDBC driver:

- Restricts SQL to SELECT, INSERT, UPDATE, and DELETE statements only.
- Restricts access to only those tables in the JD Edwards World user library list.
- Applies JD Edwards World Action Code Security to control access to SELECT, INSERT, UPDATE, and DELETE SQL statements.
- Applies JD Edwards World Business Unit Security.
- Recognizes JD Edwards World Column Security based on World Writer Column Security definitions.
- Continues to take full advantage of IBM® object security. Each IBM database connection is based on the connected user credentials.

1.1.2 Additional Features

The JDBC driver provides the following additional features:

- Works with JD Edwards World software releases from A7.3 forward.
- Installs easily (less than 1 Mb jar file) and is dependent only on the JT400.jar file from the IBM JTOpen Toolbox for Java (version 6.1 or later).

- Performs basic JD Edwards World data transformations (decimal position, date format) based on JD Edwards World Data Dictionary definitions of the table column.
- Supports presumptive joins for the following JD Edwards World tables:
  - User Defined Codes
  - Company Constants
  - Currency Codes
  - Payment Terms
- Supports customized JD Edwards World and third-party data tables.
- SQL statements that identify transaction boundaries are also supported.

1.2 Terms

The following table contains synonymous terms that you find in SQL and the IBM i file system:

SQL	IBM i File System
Database	Server
Schema	Library
Table	File
Column	Field

Note: In this document, the name IBM i includes IBM servers named AS/400, eServer iSeries, System i5, System i, or Power Servers running the IBM i for Business operating system.

2

World JDBC Driver Installation

This chapter contains these topics:

2.1 System Requirements

Prior to using the JDBC driver, verify the following:

2.1.1 IBM i Server

The JDBC driver supports the JD Edwards World database on IBM i servers with the following:

- V5R4M0 operating system and later releases of the IBM i operating system

2.1.2 Client Systems

The JDBC driver is certified with JTOpen Toolbox for Java version 6.1 or later and is supported on several client OS platforms. You can download JTOpen Toolbox for Java from SourceForge JTOpen: IBM Toolbox for Java Website.

MS Windows

Runs natively on any version of Windows that supports Java 1.6 or later runtime JVM.

IBM i OS

- Runs natively on the classic 64 bit JVM provided by IBM i in all OS releases prior to V7R1.
- Runs natively the 32 bit J9 JVM provided by IBM i beginning with V7R1 and made available in V6R1.
- Requires Java J2SE™ Runtime Environment 6.0 (JRE 1.6) or later.
- Use either the JTOpen Toolbox for Java jt400.jar or the jt400Native.jar Java libraries.

Oracle Enterprise Linux

- Oracle Enterprise Linux 6 and higher.

Red Hat Linux

- RedHat Enterprise Linux 6 and higher.

2.1.3 Standards

The JDBC driver:

- Adheres to the Java JDBC 4.1 API interface.
- Implements the Java SE 7 JDBC interface standards.
- Adheres to the SQL standards found in the IBM i DB2 for I SQL Reference 7.1.

2.1.4 What You Should Know About

Item	Description
Prepared Statements	Prepared statements are fully supported.
Callable Statements	The JDBC driver does not support callable statements.
Single Currency Mode	This release of the JDBC driver performs currency decimal placement for single currency mode only. This is based on the standard decimal position placement for the data dictionary item of the database column for any given table. See Section 4.2.4, "Decimal Position Placement" for more information.
Multi-Language	This release of the JDBC driver does not support multi-language configurations within the JD Edwards World data files.

2.2 World JDBC Driver Installation

2.2.1 Driver Jar Files

Follow these steps to install the JDBC driver jar files:

1. Download the JD Edwards JDBC 3.1 driver package from My Oracle Support.
2. Unzip the JDBC driver files to a folder location of your choice. The jar file that is used is named JDEWorldJDBC.jar.
3. Download the IBM JTOpen Toolbox for Java package.
4. Unzip to a folder location of your choice. The jar file that is used is named jt400.jar. If you are running java on the IBM i system, you can choose to use the jt400Native.jar instead.
5. Copy the JDEWJDBCDriver.jar file and the jt400.jar files to the desired location within your Java application.

You can save the driver jar files anywhere on your application system; however, you should base the decision where to save the files on your client application needs. Oracle recommends that you refer to your application software documentation for this information.

Place the JDEWorldJDBC.jar driver file and jt400.jar in a folder that your Java application recognizes. Most applications prompt you for the location of the driver jar files when you define the JDBC driver or JDBC connection. Other applications have a specific location for Java jar files.

For example, JD Edwards World Business Intelligence (BI) Publisher has a specific directory it adds automatically to the classpath:

- Oracle BI Publisher Enterprise: <Install_directory>\j2ee\home\applib
- Oracle BI Publisher StandAlone: <Install_directory>\oc4j_bi\j2ee\home\applib

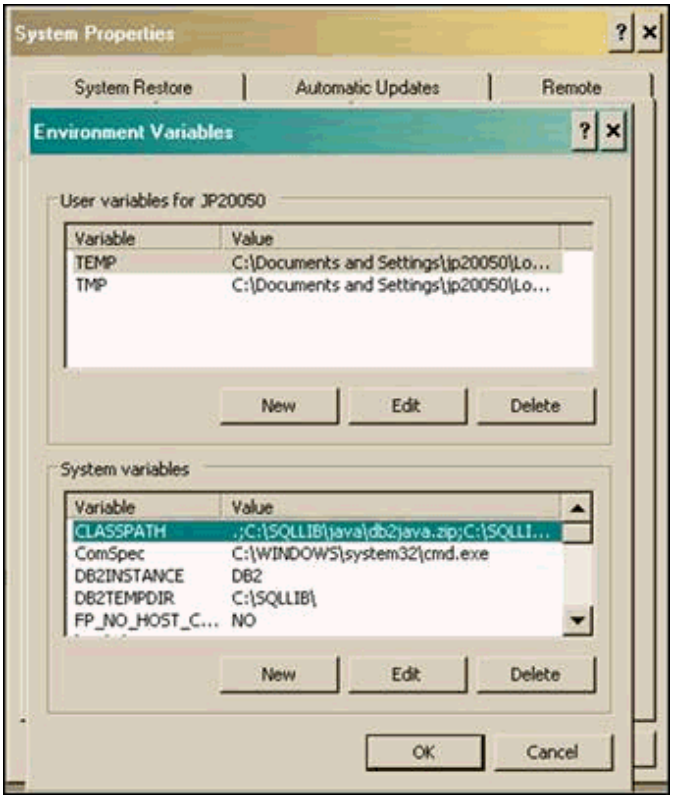
Copy and paste both jars into the specific folder for the BI Publisher installation.

2.2.2 Other Applications

If you are running a Java program from the command line, you can alternatively change your system classpath setting to include both jar libraries or alter the script batch program where the java.exe program is executed. Your classpath should include both the JDEWorldJDBC.jar and JT400.jar Java libraries respectively.

In Windows, use the System Properties dialog to change the classpath settings as follows:

Figure 2–1 System Properties screen



See your application software documentation for more information on the location for third party Java libraries.

2.3 JD Edwards World Database Connections

JD Edwards World JDBC provides several solutions for creating a JDBC Connection to JD Edwards World. Each has a different objective and purpose. The solution that is right for your given situation will depend on your application and requirements.

JD Edwards Class Name	Java API Implementation	Purpose
JDEWJDBCDriver	java.sql.Driver	General purpose database access. This class is based on JTOpen com.ibm.as400.access.AS400JDBCDriver
JDEWJDBCDataSource	javax.sql.DataSource	Used for configuring data source connections within application servers where connection pooling is not required. This data source is based on JTOpen com.ibm.as400.access.AS400JDBCDataSource. See JTOpen documentation for details.
JDEWJDBCCPDataSource	javax.sql.ConnectionPoolDataSource	Used for configuring data source connection pools within application servers where connection pooling is to be used. This data source is based on JTOpen com.ibm.as400.access.AS400JDBCConnectionPoolDataSource. See JTOpen documentation for details.
JDEWJDBCMngdDataSource	javax.sql.DataSource	Used with custom applications where connection pooling is not required. IBM recommends not using this class where connection pooling is already provided. This data source is based on JTOpen com.ibm.as400.access.AS400JDBCManagedDataSource. JTOpen provides automatic connection pooling. See JTOpen documentation for details.
JDEWJDBCMngdCPDataSource	javax.sql.ConnectionPoolDataSource	Used with custom applications where connection pooling is being managed by the client application. IBM recommends not using this class where connection pooling is already provided by another service such as an application server. This data source is based on JTOpen com.ibm.as400.access.AS400JDBCManagedConnectionPoolDataSource. JTOpen provides connection pooling. See JTOpen documentation for details.
JDEWJDBCXADataSource	javax.sql.XADataSource	Used for distributed or "global" transactions which may span multiple resources. Requires the use of a coordinating transaction manager.

2.3.1 Defining a JDBC Connection Using JDEWJDBCDriver

Follow these guidelines when defining the JDBC driver connection.

2.3.1.1 Connection String

Every JDBC driver uses a connection string to identify how the driver connects to the remote system. The JD Edwards World base URL connection string is a specific format. It begins with the character string jdbc:jdew://<IBM i Name >. A list of parameter

values that meet your connection needs follows. See Section 2.3.5.3, "Connection String or Property Options" for more information. The following illustrates a basic JD Edwards World connection string:

- jdbc:jdew://<IBM i Name>; JDEWEnvironment=<UserEnvironmentName>; JDEWRole=<UserRoleName>; user=<mylogin>; pwd=<mypassword>

You replace the following:

- <IBM i Name> with the network name or IP address of the IBM i server
- <UserEnvironmentName> with the 10 character JD Edwards World environment name. See Section 2.3.3, "Identifying a JD Edwards World Environment" for details.
- < UserRoleName > with the 10 character JD Edwards World user role name. Role is optionally used beginning with release A93 and later. See Section 2.3.3, "Identifying a JD Edwards World Environment" for details.
- <mylogin> with any valid JD Edwards World User ID. This is an optional parameter within the URL. Many applications handle the login and password separate from the URL; however, you must enter a User ID in order to connect.
- <mypassword> with the user password. This is an optional parameter within the URL. Many applications handle the login and password separate from the URL. However, you must enter a password in order to connect.

  The client application typically adds the user ID and password so they might not be necessary in the URL as you create a connection definition. If the client application does not provide the user ID and password, the IBM Toolbox for Java driver prompts you for the missing values or it will fail to connect.

Adding the user and password to the URL connection string is not secure and not recommended. Most applications manage the user ID and password as separate data entry fields. If so, then you do not need to enter them as part of the connection string.

2.3.1.2 Database Driver Class

When defining a JDBC driver connection in your client application, you must also supply the database driver class name. This is the program in the JDEWorldJDBC.jar library that is the initial entry point of the driver. Java is case-sensitive and you must enter the database driver class name exactly as it is below. Where the application prompts you for the JDBC driver class name, enter the following:

- com.jdedwards.as400.access.JDEWJDBCDriver

2.3.2 Defining a JDBC Connection Using a Data Source

Most web and Java based business applications are deployed on an application server such as WebLogic, OC4J, or WebSphere. Such applications require, or at least encourage, the use of data sources for obtaining database connections.

The main advantages of using a DataSource object to make a connection are:

- Applications do not need to hard code a driver class.
- Changes can be made to a data source's properties, which mean that it is not necessary to make changes in application setup or code when something about the data source or driver changes.
- Connection pooling and distributed transactions are available through a javax.sql.DataSource interface object that is implemented to work with the

middle-tier infrastructure. Connections made through the java.sql.Driver interface do not have connection pooling or distributed transaction capabilities.

Driver vendors provide DataSource implementations. A particular DataSource object represents a particular physical data source, and each connection the DataSource object creates is a connection to that physical data source.

A logical name for the data source is registered with a naming service that uses the Java Naming and Directory Interface™ (JNDI) API, usually by a system administrator or someone performing the duties of a system administrator. An application can retrieve the DataSource object it wants by doing a lookup on the logical name that has been registered for it. The application can then use the DataSource object to create a connection to the physical data source it represents.

A DataSource object can be implemented to work with the middle tier infrastructure so that the connections it produces will be pooled for reuse. An application that uses such a DataSource implementation will automatically get a connection that participates in connection pooling. A DataSource object can also be implemented to work with the middle tier infrastructure so that the connections it produces can be used for distributed transactions without any special coding.

JD Edwards World provides four different data source implementations. Each of these is a wrapper to a corresponding IBM JTOpen data source implementation. The four data source types are explained on the previous page.

Use the following guidelines when defining the JDBC data source connection:

2.3.2.1 Data Source Connection Properties

Every JDBC data source definition uses connection properties to identify how the data source connects to the remote system. The connection properties are a list of parameter values that meet your connection needs. See Section 2.3.5.3, "Connection String or Property Options" for more information. The following illustrates basic JD Edwards World connection properties for a typical JDEWJDBC Data Source connection:

- Servername= <IBM i Name>
- JDEWEnvironment=<UserEnvironmentName>
- JDEWRole=<UserRoleName> (if A93 or later and user connects using a JDE World role)
- Translate binary=true
- Prompt=false
- Any other connection properties desired

When using data sources, user and password values are normally handled separate from the connection properties, however, they may be included in the list of connection properties as well. The user and password connection properties may be defined as part of the data source definition as well, but this is not secure and not recommended.

2.3.2.2 Database Data Source Class Name

When defining a JDBC data source connection for your client application, you must also supply the database data source class name. This is the program in the JDEWorldJDBC.jar library that is used to create the database connection to JD Edwards World. Java is case-sensitive and you must enter the database driver class name exactly as it is below. Where the application server JDBC prompts you for the JDBC data source class name, enter one of the following:

- For a general purpose data source use:

  com.jdedwards.as400.access.JDEWJDBCDataSource

- For a ConnectionPool data source use:

  com.jdedwards.as400.access.JDEWJDBCCPDataSource

- For an XA data source use:

  com.jdedwards.as400.access.JDEWJDBCXADataSource

2.3.3 Identifying a JD Edwards World Environment

Your JD Edwards World system administrator should be very familiar with your specific JD Edwards World Environments and users of those environments. (The terms JD Edwards World environment and library list are synonymous.) The JDBC driver uses your existing JD Edwards World environments to identify and describe the files that may be accessed. With this information, the JDBC driver establishes a predefined library list for the JDBC connection. The JDBC driver uses this library list to identify the JD Edwards World database tables that the JDBC connection uses. All JD Edwards World information that the JDBC driver uses for the connection includes only the tables in this environment library list.

There are two types of JD Edwards World environments. JD Edwards World bases the environment type on the initial program that it uses from the users IBM user profile definition. You can use the IBM i command

DSPUSRPRF <mylogin>

to identify the initial program that you use for your JD Edwards World user profile. The two types of initial programs the system uses to log into JD Edwards World include the following:

2.3.3.1 Users with J98INIT as Initial Program

If the User ID (IBM user profile) uses J98INIT as the initial program in your user profile definition, then the JD Edwards World environment name is the same as your JD Edwards World User ID. You enter JDEWEnvironment=<mylogin> in the URL connection string to identify your login environment.

2.3.3.2 Users with J98INITA as Initial Program

J98INITA supports multiple JD Edwards World environments and the user typically has the option of selecting from several valid working environments. When using the JDBC driver, you indicate the environment you want to use by entering JDEWEnvironment=<UserEnvironmentName> in the URL connection string. After you log in to JD Edwards World with your standard IBM 5250 client application using the green screen platform, you can locate the UserEnvironmentName value.

To identify a JD Edwards World environment with J98INITA, perform one of the following:

- If the first screen that displays is JD Edwards World Library List Selection:
  - Enter 2 in the Option field to access the Library List Selection. The library list displays.
  - The JD Edwards World environment name displays above the library list. For example, it might display as: Library List - TESTA91, where TESTA91 is the library list (JD Edwards World environment) name.
  - Enter JDEWEnvironment =TESTA91 in your JDBC connection string.

- If the first screen that displays is the JD Edwards World Main Menu:

  You must access the Multi-Libl - Library List Selection Revisions program (P0093) to locate the name. You might need assistance from your system administrator to complete this task.

  - On the JD Edwards World Master Directory menu (G), enter G944 on the command line. On the Library List Control menu, choose User Signon List Revisions.
  - On User Signon List Revisions, locate your User ID. The User Signon List Revisions screen displays those environments that you can access. Note the library list name in the Library List field and choose Exit (F3).
  - Enter JDEWEnvironment=<myLibraryListl> in your JDBC driver connection string.

2.3.4 JDBC Driver and Data Source Properties

JDBC driver properties are parameters that affect the manner in which the JDBC driver or data source connects to the database server. The properties provide instructions to the driver that governs the behaviors of the JDBC driver connection. It is not necessary to include every property in your connection URL. Most properties have a default value; if the user does not provide the property in the URL then the JDBC driver uses the default value.

The JDBC driver uses the JTOpen Toolbox for Java JDBC driver as a foundation. Your connection URL might include any of the properties from the following:

- JDBC Driver Properties
- IBM Toolbox for Java JDBC driver properties. See Section 2.3.4.2, "IBM Toolbox for Java JDBC Properties" documentation for more information about these driver properties and choices.

You add all driver properties to the URL connection string or, optionally, you can add them to the connection properties list. Some applications allow you to define JDBC driver connection properties as a list of keywords and values, however, most applications simply use the URL connection string.

Following are all of the properties that the JDBC driver utilizes. Some of these are JD Edwards World only and IBM Toolbox for Java JDBC (jt400.jar) does not reference these. Others are IBM properties which JD Edwards World has overridden to behave in a slightly different manner.

For a complete list of IBM properties see Section 2.3.4.2, "IBM Toolbox for Java JDBC Properties." If the property is not in the following table, the IBM property functions as intended.

2.3.4.1 JD Edwards World JDBC Driver Properties

All JDBC driver properties are specific to the JDBC driver and they all begin with JDEW. JDBC driver properties are not case sensitive.

JD Edwards World Property	Description	Required	Choices	Default
JDEWEnvironment	Specifies the JD Edwards World user environment that the connection must use. The JDEWEnvironment value is the same value you use to define the user in the programs on the Security Officer menu (G94). If you enter an invalid environment name, the connection to the IBM i system fails. The JDEWEnvironment parameter identifies the IBM i library list that the system uses to make the connection. You must enter the JD Edwards World library list and this replaces any environment library list that might be in the IBM libraries connection property. JDEWEnvironment changes the connection library list to the user library list set up by the JD Edwards World System Administrator using the Multi-Libl - Library List Revisions program (P0094) on the Library List Control menu (G944). See Section 2.3.3, "Identifying a JD Edwards World Environment" for information about identifying your JD Edwards World Environment name.	Yes	Use any 10-character JD Edwards World environment name	None
JDEWRole	Beginning with release A93, if a user has a role defined in JDE World, then the connection string must include the JDEWROLE connection property. The Role property identifies the name of the JDE World role the will be used for the connection.	Maybe	Any valid role that the user has been assigned. If no role is assigned, then this parameter is options. See you system administrator for valid values.	None
JDEWTableNomenclature	Specifies an alternate naming system to identify table objects in the system. This alternate name renames the database tables so that externally, they have a more descriptive textual name. If users are not familiar with the JD Edwards World database structures, the standard file names can make it difficult to identify the business information in a table. It might be helpful to use an alternate nomenclature when referring to the tables in the JD Edwards World database. This allows the same database table to use a more descriptive textual name that is based on the descriptive information in JD Edwards World software. The descriptive text is available in the driver metadata that describes the database tables and columns. This can be very convenient when using 3rd party tools to build SQL statements. There are certain aspects you need to be aware of when using an alternate nomenclature. See Section 2.3.5, "Alternate Object Reference Nomenclature."	No	Use any combination of the following keywords, separated by an underscore (_) character: **OBJN** Object Description Name (DSPOBJD) **OBJT** Object Description Text (DSPOBJD) **OBJA** Object Description Attribute (DSPOBJD) **SVRD** File SVR Description (SVR)	OBJN

JD Edwards World Property	Description	Required	Choices	Default
JDEWColumnNomenclature	Specifies an alternate naming system to identify column names in the system. This alternate name renames the database table columns so that externally, they have a more descriptive textual name. If you are not familiar with the JD Edwards World database structures, you might have difficulty identifying the business information in a table column from the standard column names. It might be helpful to use an alternate nomenclature when referring to the columns in the JD Edwards World database. This allows the same database column name to use more descriptive textual names that are based on the descriptive information in JD Edwards World. The descriptive text is available in the driver metadata that describes the database tables and columns. This can be very convenient when using 3rd party tools to build SQL statements. There are certain aspects you need to be aware of when using an alternate nomenclature. See Section 2.3.5, "Alternate Object Reference Nomenclature."	No	Enter any combination of the following keywords, separated by an underscore (_) character: **FDFN** File Description Field Name (DSPFFD) **FDFT** File Description Field Text (DSPFFD) **FDTN** File Field Description Table Name **DDCD** Data Dictionary Code **DDAD** Data Dictionary Alpha Description **DDCT** Data Dictionary Column Title **DDCO** Data Dictionary Column Title Override else Data Dictionary Column Title **DDRD** Data Dictionary Row Description **DDRO** Data Dictionary Row Description Override else Data Dictionary Row Description	FDFN
JDEWFunctionSchema	Specifies the IBM i library that the driver might use to create driver objects (SQL functions). By default the driver creates any SQL Functions in QGPL. This parameter option changes the location where you create any JD Edwards World driver SQL functions. This library, either the default QGPL or the library you provide is added to the user library list. Currently, this is only necessary to support Business Unit Security. (JDEIsNumber function) All JDBC driver users must have *USE rights to the IBM i library. The first user to connect with the JDBC driver should have rights to create a SQL function (IBM i *SRVPGM object). It might be helpful if the System Administrator is the first user to connect with the JDBC driver. The System Administrator can create all of the necessary supporting SQL functions. After you create the JD Edwards World functions all connected users can subsequently use these.	No	Enter any existing IBM i library name.	QGPL

JD Edwards World Property	Description	Required	Choices	Default
JDEWBUSEvaluateNumerics	Employs the use of the JDEIsNumber function within the SQL statement to insure that all business units that it identifies with a numeric business unit range are numeric business unit values. Business Unit Security does not permit you to combine numeric and alphanumeric business units in the same range. If you set up business unit security properly, without combining numeric and alphanumeric business units, you can set this value to False. If you set this value to True, it might cause a significant impact on system performance. The system stores numeric business unit ranges in the F0001 table and they contain only numbers. For example 100000 - 999999 is a numeric business unit range. Business Unit Security requires only numeric business unit values within that range. For example, if a user creates business unit 3BEAR9, this business unit is within the 100000 - 999999 range of alphanumeric characters and the SQL might select it. By setting this value to TRUE, the JDBC driver does omit 3BEAR9 from the records it selects, which is consistent with normal Business Unit Security operations. By setting this value to FALSE, the JDBC driver includes 3BEAR9 in the records it selects, which is not consistent with normal Business Unit Security operations. See Work with Business Unit Security in the *JD Edwards World Technical Foundation Guide* for information about numeric business units and ranges within business unit security.	No	You enter: **True** to include the use of JDEIsNumber when validating business units for Business Unit Security. **False** to include the use of JDEIsNumber when validating numeric business units for Business Unit Security.	True
JDEWTraceSQL	Prints the original and the SQL statement that you revise into the application standard output.	No	You enter: **True** to log SQL statements to standard output. **False** and SQL are not logged.	False
JDEWTraceParser	Prints the detail parsing engine Token log and Node Tree log to the application standard output.	No	You enter: **True** to log parsing engine detail to standard output. **False** and this is not logged.	False
JDEWVerbose	Prints status messages during the connection phase of the driver execution.	No	**True** - logs additional status messages as the connection establishes. **False** - no additional logging occurs.	False

JD Edwards World Property	Description	Required	Choices	Default
JDEWMetaDataLoad	Identifies how the JDBC drive should behave when loading driver table specifications and other database metadata. The driver must extract the table, data dictionary, and other JD Edwards World specifications when a table is referenced by a SQL statement. Extracting the driver metadata is the biggest single performance factor in processing an SQL statement as additional data must be extracted from the database to support the JDBC driver decision making. The process of extracting metadata is only performed once by the first connection that references a given metadata value. That metadata value is subsequently cached at the JVM level and made available for later use by any and all JDBC connections. Once the Java application is closed (that is, when the JVM that runs the application is ended) all of the cached information is dropped. Note that cached data requires the JVM to use more system memory. This may necessitate changes to you JVM startup instructions.	No	**AsNeeded** - metadata will be extracted as it is needed. This option is best for single users using only a few tables. This solution is also better for users that are connected with slower network connection speeds (< 1 MB connections). **Preload** - preloads all table specifications for all environments defined in QJDF Security library. This option may be preferred when the driver is used within application servers such as OC4J and WebLogic which provide access to many JDEWorld environments. **Reload** - Forces an update to all driver metadata specifications for all environments defined in QJDF Security library. This option will refresh the existing metadata in the JVM using the current driver database connection. Note that a reload will not affect existing open JDBC Statements and ResultSet objects that already refer to the previous metadata definitions. However, it will take affect with any subsequent new reference to driver metadata.	AsNeeded

2.3.4.2 IBM Toolbox for Java JDBC Properties

IBM Toolbox for Java (IBM JDBC driver) is the foundation of the JD Edwards World JDBC driver. You can use any of the IBM Toolbox for Java properties as well. The JD Edwards World JDBC driver transmits these properties to the underlying IBM JDBC driver. However, JD Edwards World amended or revoked a few connection properties so that users must use the JD Edwards World environment.

IBM Toolbox for Java Server Property	Description	Required	Choices	Default
libraries	This property is disabled. The JD Edwards World JDBC driver overrides this libraries property and disregards any value you enter. The connection uses the library list in the JD Edwards World environment from the JDEWEnvironment property. See Section 2.3.4, "JDBC Driver and Data Source Properties" for more information	No	None	
naming	Users can use the naming convention of choice. However, the JD Edwards World JDBC driver adds additional capabilities to support JD Edwards World environments and enforce the use of the JD Edwards World library list. The JD Edwards World JDBC driver changes the inbound SQL statement so that the tables the SQL statement references only those tables in the JD Edwards World library list. The connection retains any library which is in the original SQL statement. This allows users to specifically address any table in the user's library list. In this manner, both SQL naming and system naming behave the same as they did previously. If a library is in the original SQL statement, but the library does not exist in the users JD Edwards World library list, then the SQL statement fails. In this respect, the connection restricts both SQL naming and system naming behavior. If the SQL statement does not provide a schema name with the table name, then the JDBC driver amends the SQL statement to use the first occurrence of the table in the connection library list. This results in the SQL naming behaving the same as system naming and the connection automatically locates the table in the JD Edwards World environment. This provides transparency and portability to SQL statements imbedded in user applications. If a table does not exist in the user library list, the SQL cannot reference it in the current environment and an error occurs.	No	SQL (as in schema.table) system (as in schema/table)	SQL
date format	See JTOpen documentation on date format. This parameter effects all JD Edwards World Julian dates.	No	Same as IBM property	Defaults from JD Edwards World User preferences or the System data format.
date separator	See JTOpen documentation on date format. This parameter effects all JD Edwards World Julian dates.	No	Defaults from JD Edwards World User preferences or the System data format.	

IBM Toolbox for Java Server Property	Description	Required	Choices	Default
Translate Binary	The JD Edwards World JDBC driver assumes that the user will always translate the data from EBCDIC to ASCII. Translate binary should be set to true in all circumstances.	Yes; must be True	True	

2.3.4.3 Examples of Driver Connection Strings

Note that these examples are discussed specifically as JDBC Driver connections (using JDEWJDBCDriver class). JDBC data source connections can be created using the same attributes. See Chapter 2, "World JDBC Driver Installation" for specific uses of data sources.

jdbc:jdew://MyIBMi;JDEWEnvironment=A92PROD;user=AB1234;password=inlikeflynn; libraries= AB1234 jdfOBJ JDFDTA

- Server is MyIBMi
- JD Edwards World Environment is A92PROD
- User is AB1234 and password is inlikeflynn.
- Libraries property will be ignored and set to the A92PROD library list anyway.

jdbc:jdew://10.2.1.10;JDEWEnvironment=AB1234; user=AB1234;password=inlikeflyn; prompt=false

- Server is IP address 10.2.1.10
- IBM user profile is AB1234
- JD Edwards World Environment is AB1234. This would be necessary for a user who's User Profile is configured with J98INIT as the initial program.
- Prompt = false means that IBM jt400 will not pop up a login window to request User and password.

jdbc:jdew://MyIBMi; JDEWEnvironment=A92PROD; JDEWFunctionSchema=JDFOBJ;JDEWBUSEvaluateNumerics=false; JDEWTraceSQL=true; JDEWVerbose=true; JDEWMetaDataLoad=preload; date format=ISO;JDEWRole=SOAAPPS

- Server is MyIBMi
- JD Edwards World Environment is A92PROD
- User and password would need to be provided separately by the application.
- Driver based functions/objects will be created in and used from library JDFOBJ.
- Business unit security within JDBC Driver will not validate that business units found within a given numeric Business Security range are indeed numeric values. This option will improve performance of some SQL statements; however, it should only be used if you have followed the rules for numeric ranges. (See JD Edwards World Help for JD Edwards World Menu option 5/G94 Business Unit User Security).
- JDETraceSQL=True will cause the driver to log before and after images of every SQL statement that is processed by the driver. This is helpful for diagnosing SQL statements.
- JDEWVerbose=true will provide a lot of detail about the JD Edwards World JDBC connection, the JDBC driver and the JDBC jar files that are being used.

- JDEWMetaDataLoad=preload will populate ALL JD Edwards World metadata on the very first connection attempt. This improves performance of all subsequent driver connections by developing a complete cache up front. This is very helpful for Application Server based applications.
- Date format for all dates returned by driver (including JD Edwards World Julian dates) will be in the ISO (YYYY-MM-DD) date format. These settings would override the User preferences in JD Edwards World.
- JDBC driver will respond to the Action Code, Column Security, and Business Unit Security setup for the SOAAPPS role.

jdbc:jdew://10.2.1.10; JDEWEnvironment=A92PROD; JDEWTableNomenclature= OBJT_OBJN; JDEWColumnNomenclature=FDFT_FDFN;

- Server is IP address 10.2.1.10
- JD Edwards World Environment is A92PROD.
- All table names will use the name format OBJT_OBJN. (See Section 2.3.5, "Alternate Object Reference Nomenclature.")
- All column names will use the name format FDFT_FDFN. (See Section 2.3.5, "Alternate Object Reference Nomenclature.")

2.3.5 Alternate Object Reference Nomenclature

You refer to every object in any database by its object reference. For example, F0101 is the object reference for the Address Book table and ABAN8 is the object reference for the Address Book Number field (column) in the F0101 table. Using these objects, a simple select statement using the Address Book table might resemble the following:

```
select ABAN8, ABALPH from F0101
```

These are the table and column (field) names in the JD Edwards World database and are the standard reference to these values in the JD Edwards World Address Book. However, if you are not familiar with the JD Edwards World database structures, these naming conventions might seem confusing when attempting to identify the business information in the database. Instead, it might be helpful to use an alternate nomenclature when referring to the tables and columns in the JD Edwards World database. The alternate nomenclature allows more descriptive names for the same database table and column names. The descriptive text is available in the JDBC driver metadata that describes the database tables and columns. This can be very convenient when using 3rd party tools to build SQL statements.

By using the JDETableNomenclature and JDEColumnNomenclature properties in the JDBC driver connection string, you can use alternate nomenclatures to refer to the table and column object references.

2.3.5.1 Table Nomenclature Keywords

Keyword	Description
OBJN	Object Description Name (DSPOBJD) (OBJN is the default.)
OBJT	Object Description Text (DSPOBJD)
OBJA	Object Description Attribute (DSPOBJD)

Keyword	Description
SVRD	Object SVR Description

2.3.5.2 Column Nomenclature Keywords

Keyword	Description
FDFN	File Description Field Name (DSPFFD) (FDFN is the default)
FDFT	File Description Field Text (DSPFFD)
FDTN	File Field Description Table Name
DDCD	Data Dictionary Code
DDAD	Data Dictionary Alpha Description
DDCT	Data Dictionary Column Title
DDCO	Data Dictionary Column Title Override else Data Dictionary Column Title
DDRD	Data Dictionary Row Description
DDRO	Data Dictionary Row Description Override else Data Dictionary Row Description

2.3.5.3 Connection String or Property Options

JDEWTableNomenclature. This is the connection string keyword or property you use to define the nomenclature for table names that the connection uses for the JD Edwards World database.

JDEWColumnNomenclature. This is the connection string keyword or property you use to define the nomenclature for column names that the connection uses for the JD Edwards World database.

Examples

Using the default nomenclature (OBJN and FDFN), a simple select statement in the Address Book table might resemble the following:

```
select ABAN8, ABALPH from F0101
```

Example 1:

With the connection string property

- JDEWColumnNomenclature= FDFT

in the connection string or the connection properties list, the same select statement is:

- select Address_Number_ABAN8,Alpha_Name_ABALPH from F0101

The keywords define the structure of the column name nomenclature, separated by an underscore.

Example 2:

When the connection string or connection properties include:

- JDEWTableNomenclature= OBJN_OBJT
- JDEWColumnNomenclature=FDFT_FDFN

The same SQL select statement is:

```
select
Address_Number_ABAN8,
Alpha_Name_ABALPH
from F0101_Address_Book_Master
```

2.3.5.4 What You Should Know About Alternate Nomenclature

Connection String or Property Options	Description
Alternate Nomenclature	Alternate Nomenclature can be a very effective tool for the individual that is a proficient user or a novice user that is becoming familiar with the JD Edwards World database. However, you should only use this after you finish customizing the data dictionary values and they are static. If you create an SQL statement based on the descriptive information and then the column description information changes, you must revise the original SQL statement in order for it to work. Consider this when developing reports for long term use
Different Environments and Releases	You should also be aware of changes when using the same SQL statements for different environments and releases. Different environments can contain different descriptive text for the same object reference. If the descriptions are different in different environments, the SQL statement does not work in either environment.
Table and Column Names	The OBJN or FDFN defaults must exist somewhere in the table and column nomenclature definition, respectively. If you choose a nomenclature definition that does not include OBJN or FDFN, the JDBC driver adds these attributes to the end of the nomenclature definition in the URL property.
SQL Statements	Using alternate nomenclatures can create very long SQL statements for queries in the tables. On rare occasions, this might result in an SQL statement exceeding the maximum length of 65535 characters. See the *IBM SQL Reference* documentation.

2.3.6 Global JDBC Driver Properties

Many JDBC driver connection options are settings that you should use globally with every JDBC driver connection. The JDBC driver uses UDC 98/JD to identify common settings that all JDBC connections use for the current database environment. Because the JDBC driver forwards all IBM properties to the JTOpen driver, this feature is effective for any IBM JTOpen driver properties as well.

The JDBC driver accesses UDC 98/JD to locate global options. Any options the driver locates here supersede this option setting on the local workstation. The setting applies to any connections that you establish subsequent to adding the global setting. Users need to disconnect and reconnect for the global settings to take effect. Existing active JDBC connections continue to utilize their current settings.

You can use the global settings with both JDBC connection options and IBM related options. When you add the IBM related options to the JD Edwards World connection string they apply to the underlying IBM JTOpen database connection string for use as well. This feature is only available with the JDBC driver connection.

The 98/JD UDC table allows you to use the Global JDBC driver settings to:

- Enforce standard business decisions.
- Implement changes to existing database connection definitions for all database connection users.

- Maintain consistency of JDBC Connection usage.
- Implement performance improvement to all JDBC connections.

Consider the following examples as you set up the global JDBC driver UDC table.

Global Setting Example 1:

Every JDBC driver database connection most likely converts EBCDIC to ASCII and therefore all connections should use the connections string option that follows:

- translate binary=true

To establish this as a rule for all JDBC connections in all cases, add the following to the 98/JD UDC table:

- 10 Digit Code: any number
- Description 1: translate binary
- Description 2: true

This forces all database connections made through the JDBC driver to include the translate binary=true property setting. The client application URL does not require this property setting.

Global Setting Example 2:

If your enterprise makes a business decision to use JDEWBUSEvaluateNumerics = false to more efficiently execute the SQL statements, then you might want to use this option with all JDBC connections to the JD Edwards World database. Instead, you can add "JDEWBUSEvaluateNumerics = false" to the UDC 98/JD table. All subsequent database connections that use the JDBC driver use this option without changing the application setup.

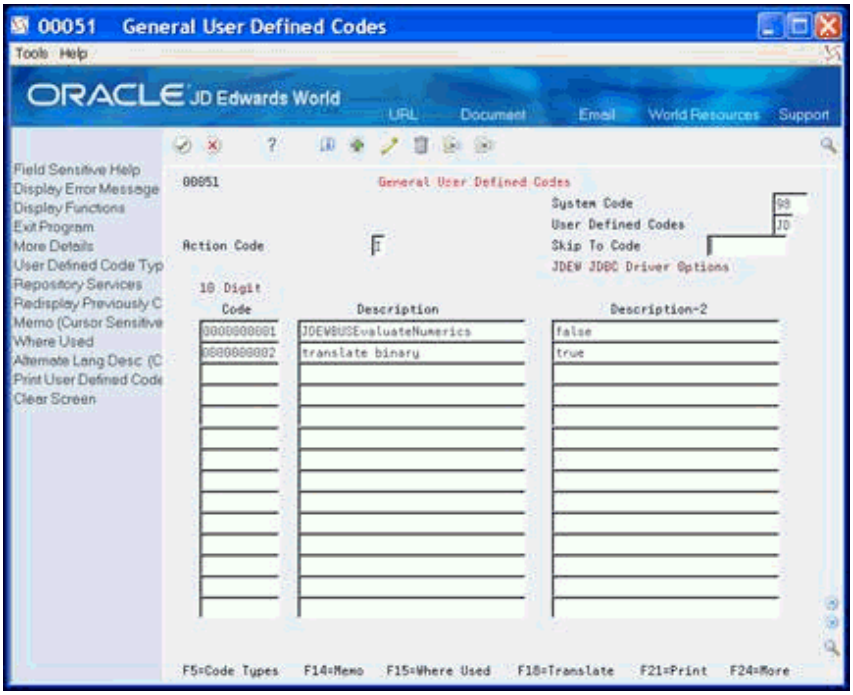
Note: See JDEWBUSEvaluateNumerics in Section 2.3.4, "JDBC Driver and Data Source Properties" before you use this option.

To implement a global JDBC driver connection option, add the connection option to the UDC 98/JD.

Note: Do not add user ID and password as a global setting. Doing so gives all connections the same user and password for all connection environments that use this UDC 98/JD table.

- Enter any number in the 10 Digit Code field.
- Enter the JDBC driver property keyword in the Description field. See Section 2.3.4, "JDBC Driver and Data Source Properties" for more information.
- Enter the JDBC connection value in the Description 2 field. See Section 2.3.4, "JDBC Driver and Data Source Properties" for a list of properties.

Figure 2–2 General User Defined Codes screen



2.3.7 JDBC Property Information Sources and Precedence for JDBC Connections

Property sources refer to the sources of information that can be used to set a JDBC property value.

Property precedence refers to the hierarchy by which these sources are employed.

During the initial connection to the JD Edwards World application, properties are collated from various sources to formulate the complete list of properties that are in effect for each specific JDBC Connection.

Properties may also be set using different solutions defined by the JDBC interface being used when creating a JDBC connection; these include specific property setters, URL strings and Property lists. These are explained further below.

Once all of the properties are identified, the connection is created using the property options identified according to the precedence order. The precedence order dictates which properties source and settings are observed. The setting source that has the highest precedence order is the value that is used. At each precedence level, if the value is set several difference times with different values, the last value received is the value that applies for any given property setting.

The following settings outline the precedence order for various property settings from highest precedence to lowest precedence.

2.3.7.1 Global Properties - (UDC 98/UD)

See Section 2.3.6, "Global JDBC Driver Properties" above for details on Global properties.

- Global properties are loaded from the 98/JD UDC table. This is the way to govern outside connections from within JD Edwards World.
- Applies to both driver and data source connections.

2.3.7.2 JDBC Java VM Parameters

Only applies to JDEWVerbose property only. This is to allow some fine detail logging of connection events at the initiation of the JDBC driver application startup and is not normally used.

2.3.7.3 JDBC Property Settings

- Via DataSource Setters
  - These are the individual DataSource option settings (e.g. setUser(…), setJDEWEnvironment(…)).
  - These property setting options are utilized by all data source classes and apply to data source connections only.
- Via URL String
  - With a data source connection, properties may be set using a URL style connection string with setProperties(String propertiesString) method. URL properties will have the same weight as a DataSource setter method.
  - With a Driver connection, URL string properties take precedent over the Properties list in standard Driver connections as specified by IBM JTOpen.
- Via Properties List
  - With a data source connection, property list entries may also be set using the setProperties(Properties propertieslist) method and have the same weight as a DataSource setters.
  - With a Driver connection, the Properties list is subordinate to the settings defined in the URL string.

2.3.7.4 User Preferences (Format Properties Only)

- Used when User Preference information is defined in JD Edwards World.
- Applies to both driver and data source connections.

2.3.7.5 IBM System Values (Format Properties Only)

- Used when User Preference information is not defined in JD Edwards World.
- Applies to both driver and data source connections.

2.4 BI Publisher 11G JDBC Data Source Setup with WebLogic Application Server

For customers implementing BI Publisher 11g and later, you may want to use either a standard JDBC Driver connection or a JNDI data source connection for your database connections to JD Edwards World.

Using a simple JDEWJDBCDriver based connection uses the URL connection string, which was previously described in this user guide.

2.4.1 Install Jar Files to the WebLogic Server

Follow these steps to install the jar files to the WebLogic server:

1. Locate the Domain folder for your WebLogic Server instance.

   Your BI Publisher domain might be found at the following location:

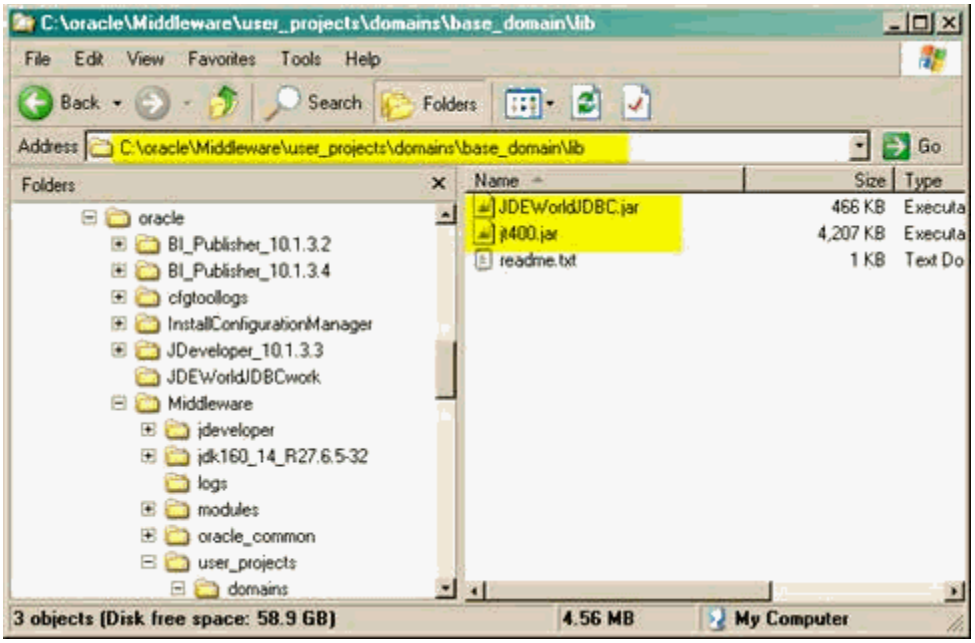
- C:\OracleMW\user_projects\domains\bifoundation_domain

where "bifoundation_domain" is the name of your specific WebLogic domain. Your setup may vary.

2. Copy the JDEWorldJDBC.jar and jt400.jar to the following folder on your BI Publisher 11G application server domain:

   domains\<domain_name>\ lib

Figure 2–3 BI Publisher 11G Server Domain Folder



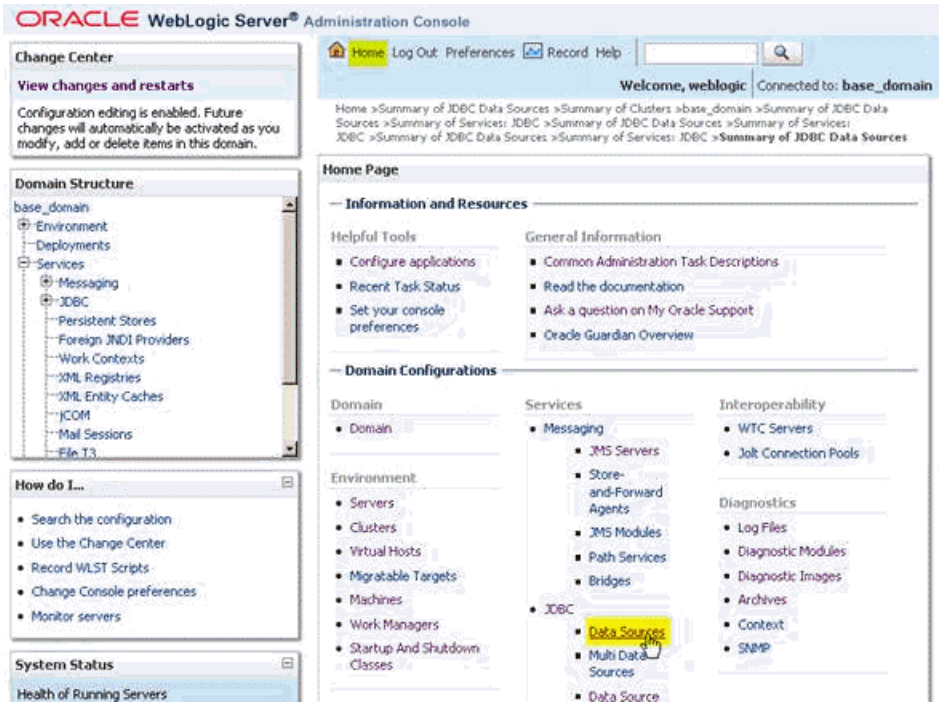
3. Stop and restart WEBLOGIC server and applications.

2.4.2 Define a Standard JDBC Data Source

Follow these steps to define a standard JDBC data source:

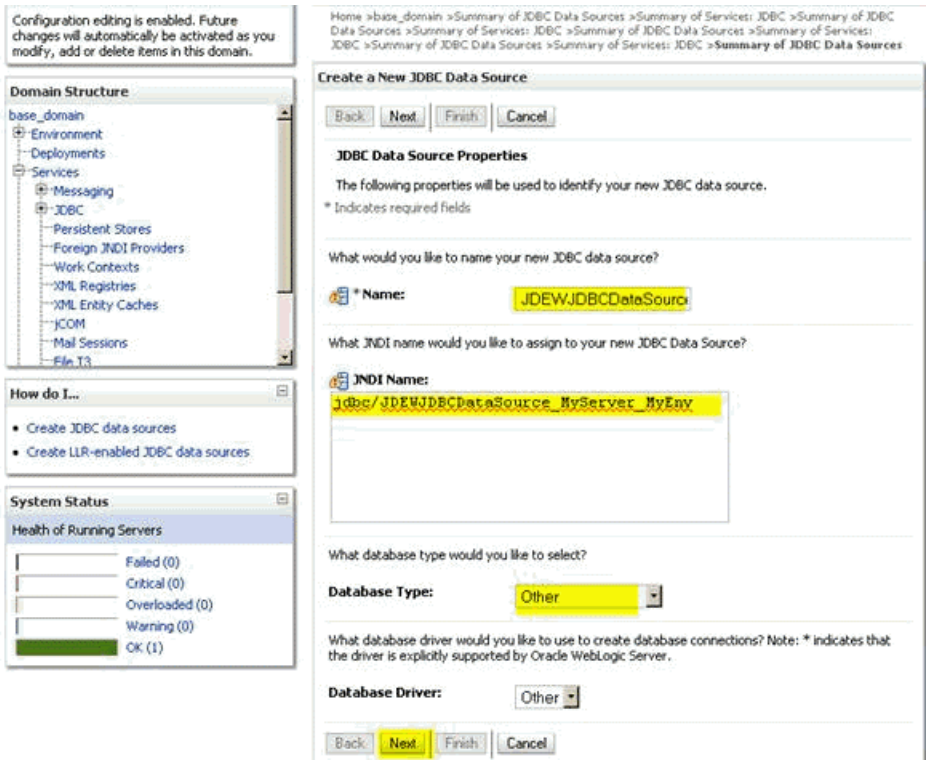
1. In your browser, login to your WebLogic Server Administration Console Home page and click the JDBC Data Sources link.

Figure 2–4 WebLogic Server Administration Console Home screen



2. On the Summary of Data Sources page, click the **New** button.
3. On the Create a New JDBC Data Source form:

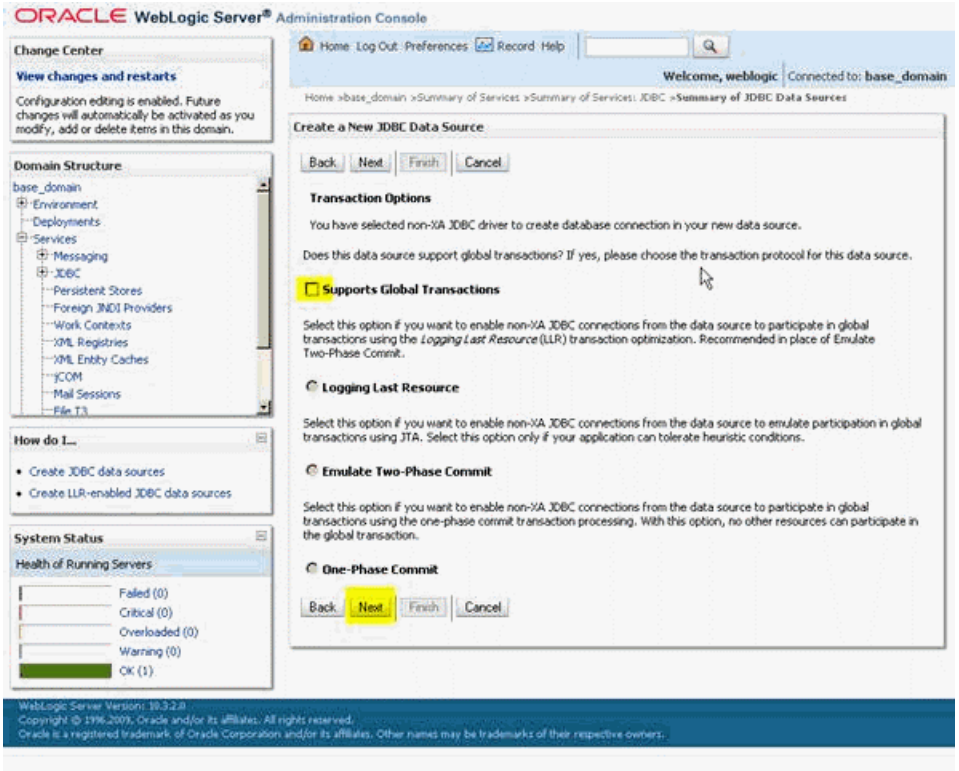
Figure 2–5 Create a New JDBC Data Source screen



- Enter a data source name. Oracle suggests that you use a name that references the JDBC service, System name and JDE environment as illustrated above.
- Enter a JNDI name.
- Choose the "Other" database type.
- Click the Next Button.

4. Choose Transaction Type.

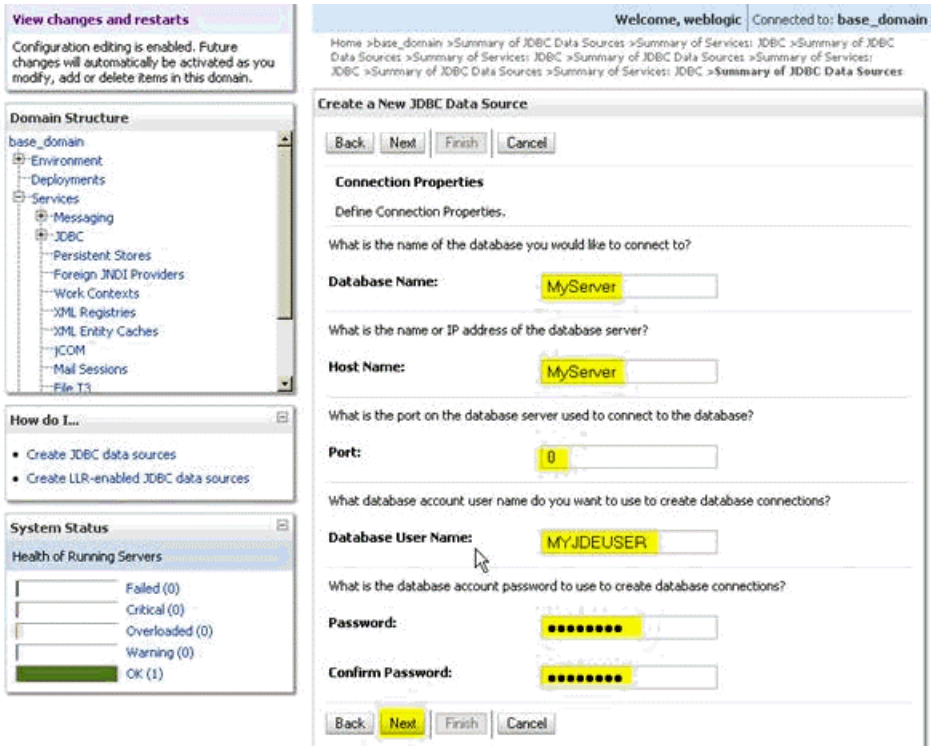
Figure 2–6 Create a New JDBC Data Source (Supports Global Transactions Unchecked) screen



- Uncheck the Supports Global Transactions option.
- Click the **Next** button.

5. Enter the database server and user information.

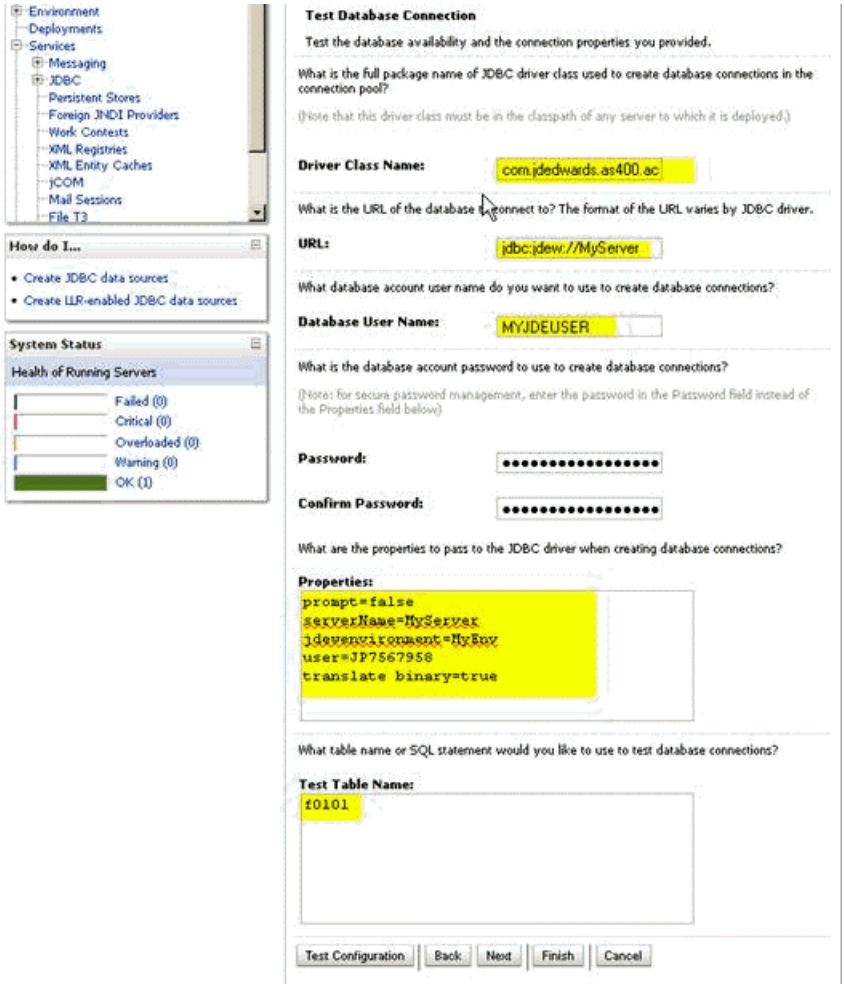
Figure 2–7 Create a New JDBC Data Source (Server and User Information) screen



- Database Name is the name of your JD Edwards World system.
- Host Name is the database server name or IP address.
- Set the Port number to 0.
- For Database User Name, enter a valid JD Edwards World user login.
- Enter password.
- Click the **Next** button.

6. Enter the database server and user information.

Figure 2–8 Create a New JDBC Data Source (Database and Server Information) screen



- Enter the data source class name:

  com.jdedwards.as400.access.JDEWJDBCDataSource

  or

  com.jdedwards.as400.access.JDEWJDBCCPDataSource.

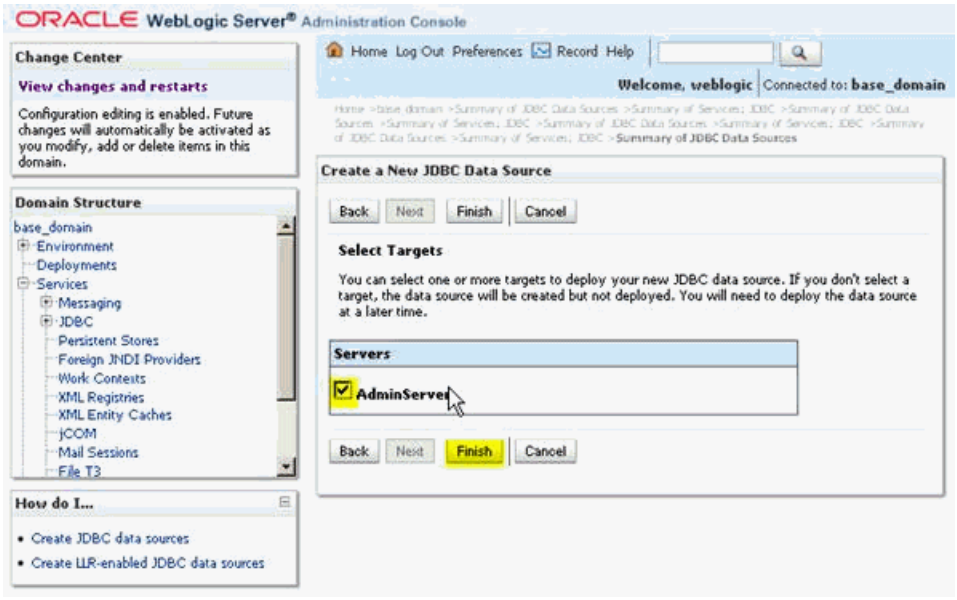
  or

  com.jdedwards.as400.access.JDEWJDBCXADataSource.
- Enter the data source URL.
- Enter data source properties as necessary.

  Note: Beginning with JDE World A9.3, if your user must log in with a user Role, then you need to include the JDEWRole property as well.

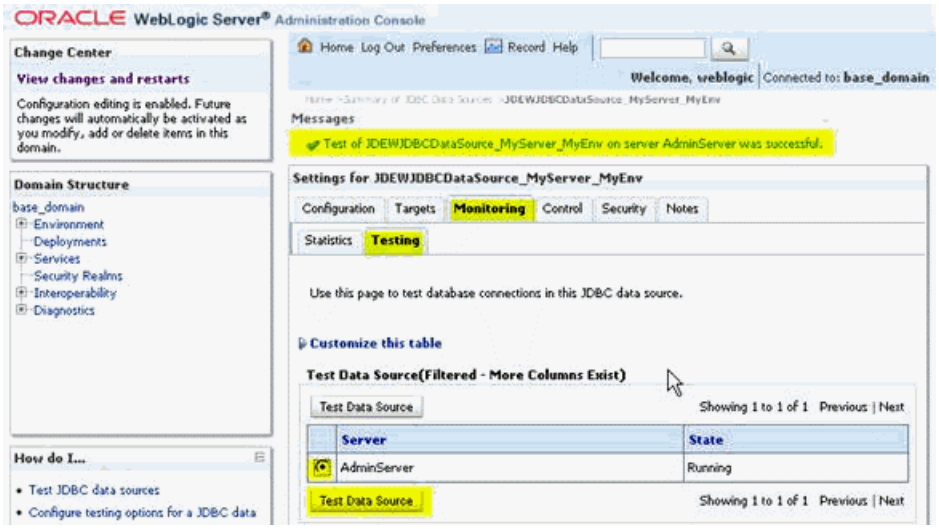
7. Assign the data source to the WebLogic server instance(s) where it will be used.

Figure 2–9 WebLogic Server Administration Console screen



8. After clicking the **Finish** button, the new data source definition will be tested using the assigned credentials. If everything is satisfactory, all of the Verify that the connection succeeded Messages will be green.
9. Test the Connection.
   - Click the data source name on the Summary of Data Sources page.
   - Click the Monitoring tab and click the Testing tab.

Figure 2–10 WebLogic Server Administration (Testing) screen



   - Select the server and click the **Test Data Source** button.
   - Observe the message that is returned at the top of the form.

2.4.3 Define a Connection Pool JDBC Data Source

Creating a Connection Pool data source is identical to creating a Standard data source described in the previous topic except that a different Driver Class Name is used. To define a true Connection Pool data source, use the following class name instead:

- com.jdedwards.as400.access.JDEWJDBCCPDataSource

Connection Pool attributes may be revised on the **Advanced** subsection of the Connection Pool tab.

2.4.4 Define an XA JDBC Data Source

Creating an XA data source is identical to creating a Standard data source described in the previous topic except that a different Driver Class Name is used. To define an XA data source, use the following class name instead:

- com.jdedwards.as400.access.JDEWJDBCXADataSource

2.5 BI Publisher 10G JDBC Data Source Setup on OC4J

For customers continuing to use BI Publisher 10g, you may now configure and use JNDI data sources to create a database connection to JD Edwards World. Here are the steps required to set up a JNDI Data Source Connection in BI Publisher 10G.

2.5.1 Install Jar Files to the OC4J Server

Follow these steps to install the jar files to the OC4J server:

1. Copy the JDEWorldJDBC.jar and jt400.jar to the following folder on your BI Publisher application server:

   <BI Publisher Install home>\j2ee\home\applib

2. Copy the JDEWorldJDBC.jar and jt400.jar to the following folder on your BI Publisher application server: Stop and restart OC4J.

2.5.2 Define a Standard JDBC Data Source

Creating a usable data source in OC4J is a two-step process:

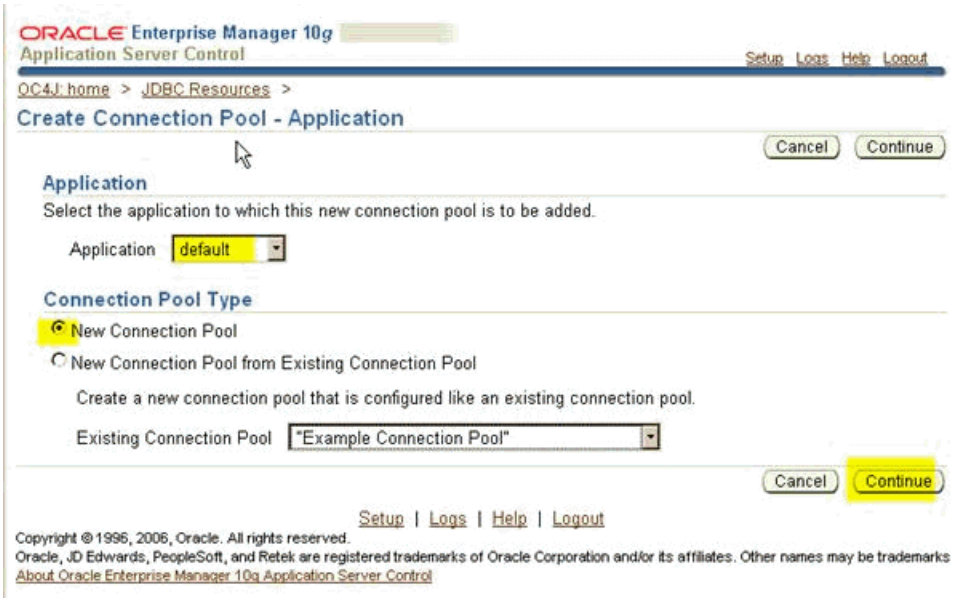
- Create a Connection Pool definition.
- Create a Named Data Source instance of the Connection Pool definition.

2.5.2.1 Create a Connection Pool Definition

Follow these steps to create a Connection Pool definition:

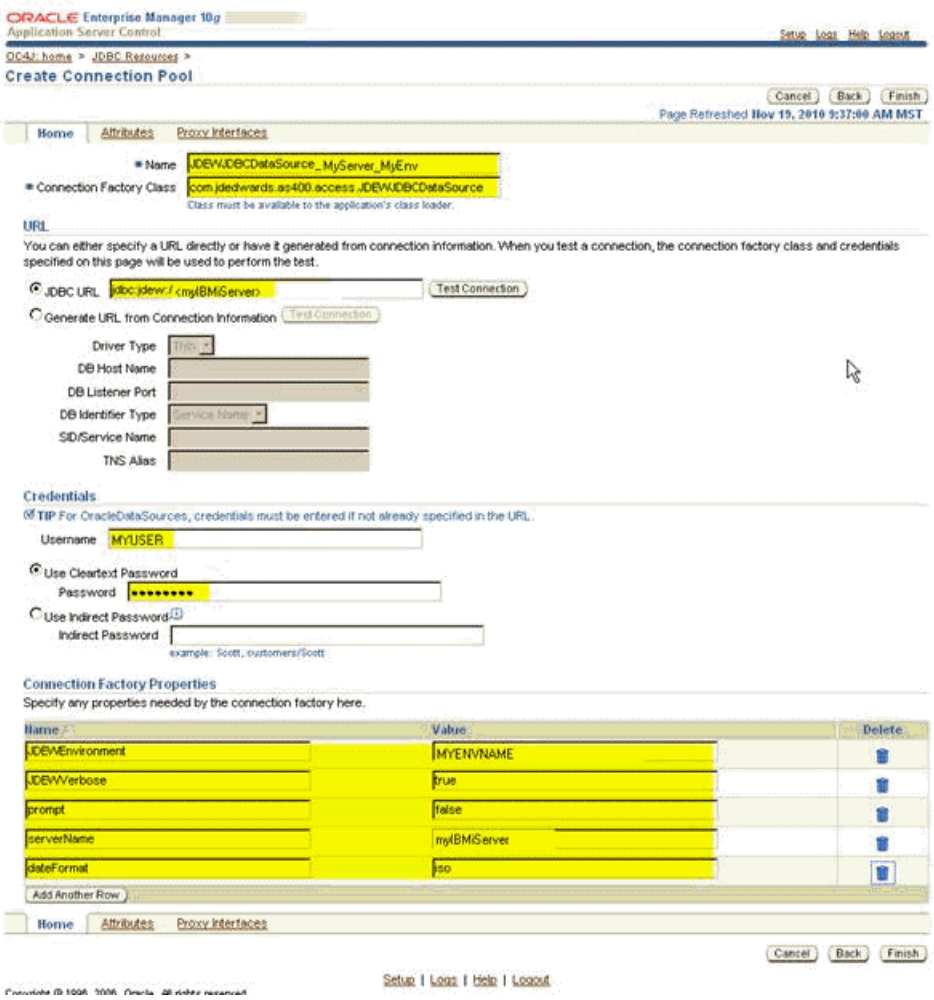
1. Launch Application Server Control and log in using your Administrator account.
2. Click the "Administration" tab.
3. Navigate to Administration Tasks/Services and click the Go To JDBC Resources task.
4. Click the **Create** button for Connection Pools.
5. Complete the entry of the Connection Pool definition using the following examples.

Figure 2–11 Create Connection Pool - Application screen



6. Choose the appropriate application and click the **Continue** button.
7. Complete the Create Connection Pool page.

Figure 2–12 Create Connection Pool screen



Note: Beginning with JDE World release A9.3, the data source setup may also require that the JDEWRole property be used.

8. Click the **Add Another Row** button to add any additional properties.

   Note that you will want to set the date format to ISO with the following option. ISO is the standard data format for XML documents user by BI Publisher.

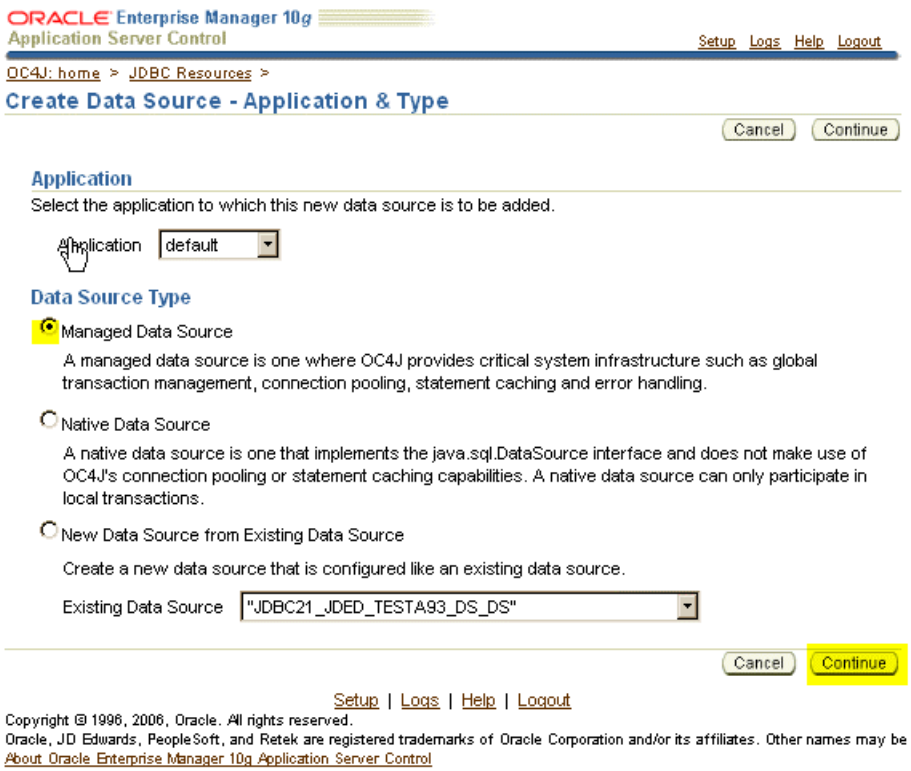
   - dateFormat=iso

9. Click the **Finish** button to save the data source definition.

2.5.2.2 Create a Named Data Source Instance of the Connection Pool Definition

1. Launch Application Server Control and log in using your Administrator account.
2. Launch Application Server Control and log in using your Administrator account.
3. Click the "Administration" tab.
4. Navigate to Administration Tasks/Services and click the Go To JDBC Resources task.
5. Click the **Create** button for Data Sources.

Figure 2–13 Create Data Source - Application & Type screen



- Enter any name for the data source. You may wish to include the server and environment name in the Data Source name.
- Enter a JNDI name. It is recommended that the name be prefaced with "jdbc/". This helps organized the information stored in the JNDI Browser.
- Choose the Connection Pool name that you created in the prior step form the dropdown list.
- Click the Finish button to save the data source.

2.5.3 Define a JDBC Connection Pool Data Source

Creating a Connection Pool Data Source is identical to creating a Standard data source described in the previous topic except that a different Connection Factory Class is used. To define a true Connection Pool data source, use the following class name instead:

- com.jdedwards.as400.access.JDEWJDBCCPDataSource

Connection Pool attributes may be revised on the Attributes Tab.

3

World JDBC Driver Administration

This chapter contains these topics:

- Section 3.1, "World JDBC Driver and JD Edwards World Security Features,"
- Section 3.2, "World JDBC Driver and JD Edwards World Data-Specific Features."

3.1 World JDBC Driver and JD Edwards World Security Features

The JDBC driver utilizes the following JD Edwards World security features:

- User access security
- Business unit security
- Field security for select
- File/field level security for insert/update/delete

3.1.1 Action Code User Access Security

The JDBC driver restricts access to JD Edwards World software using many of the same features JD Edwards World software uses. The JDBC driver user access security also includes the following:

- It does not cache or maintain user logins and passwords in the setup it uses during every database connection. The JDBC driver requires the client application to provide the appropriate login and password when requesting a connection. All existing IBM user security setup in place at your installation continues. Every user must log into the IBM i with their existing IBM i user account.
- The JDBC driver restricts access tables in the user library list in the JD Edwards World environment tables F0092, F0093, F0094, and F00944. The System Administrator controls which tables that the JDBC driver can access. See Section 3.2.2, "Environments and Specifications" for more information.
- The connection library list is the concatenation of the IBM i QSYSLIBL libraries and the libraries in the environment the user chooses.
- The JDBC driver validates every table in an SQL statement against the connection library list. See JDEWEnvironment in Section 2.3.4, "JDBC Driver and Data Source Properties."
- The JDBC driver retrieves table and column metadata as well as related control data from the current connection library list.
- If the JDBC driver references an object that does not exist in the user library list, the JDBC driver causes an error. This includes system files.

3.1.2 Business Unit Security

The JDBC driver performs Business Unit Security validation according to the business unit security rules for the user in the Business Unit Security table (F0001) and conforms to the logic in the Business Unit Security (C0000) standard RPG subroutine. See Work with Business Unit Security in the *JD Edwards World Technical Foundation Guide* for information.

The F0001 is the primary repository for business unit security rules and the user and table maintain these rules. See Section 3.2.2, "Environments and Specifications" for more information about identifying the specific instance of the F0001 file that the JDBC driver uses.

The JDBC driver extracts the business unit security rules for a given user and table and caches them for future reference. The JDBC driver converts all of the rules for a given user and table and formulates an SQL relational expression of the data values. The JDBC driver appends the SQL expression to the user's SQL statement before it is sent to the IBM i to execute. The JDBC driver always extracts the user ID from the current connection object.

The JDBC driver applies security to all database columns whose associated data dictionary item is class COSTCTRSEC. Some tables use more than one COSTCTRSEC related column. The JDBC driver examines all of these columns to ensure that all values meet the business unit security rules. The JDBC driver might disregard specific business unit security rules for specific columns. The JDBC driver uses UDC 98/UN to identify columns that the driver does not consider for Business Unit Security. The JDBC driver disregards Business Unit Security for a particular column if you enter the table column in UDC 98/UN. The value in the Code field is an arbitrary next number value. The table and column values you enter in the Description field, as shown in the following example, must include one space between them. The value in the Description 2 field indicates whether the exclusion rule is active or not. If the exclusion rule is active then the JDBC driver does not apply Business Unit Security to the database column you indicate. If the exclusion rule is not active, or does not exist in this UDC, then the JDBC driver applies Business Unit Security to the table column. You enter 1 to exclude column from Business Unit Security or another value to use Business Unit Security for the column. Following is an example of the pattern the UDC entries must use:

Code	Description	Description 2
0001	F55XX ABMCU	1
0002	F55XX ABHMCU	0

Business Unit Security differentiates numeric business unit security ranges from alphanumeric business unit security ranges. If you enter numeric values in the From and To fields in the F0001, then the JDBC driver considers only numeric business unit values when it searches for records that are within the range of the rule.

Using the SQL database function, JDEisNumber, enforces the rule that every row it identifies within the numeric business unit range is a numeric value. The JDEisNumber is a separate standalone program SQL function. The first user that attempts to use the JDBC driver automatically creates the JDEisNumber function. The JDBC driver creates this database function only once and all users of the same JD Edwards World instance use this function. The JDBC driver creates the JDEisNumber function in the QGPL library. Users can indicate a different library location for JDEisNumber using the JDEWFunctionSchema driver property. See Section 2.3.4, "JDBC Driver and Data Source Properties" for more information.

3.1.3 Field Security for Select

The JDBC driver performs field (column) level security based on information in the File/Field Level Security File (F9401) table.

When you set up Field Security, you should be aware that it uses the same mechanism as World Writer for defining secured data fields in the JD Edwards World software. You can access the File/Field Level Security - Inquiry program (P94011) by choosing Field Level Security on the World Writer Advanced Operations menu (G8231). The system maintains data values in the File/Field Level Security File (F9401) table. The JDBC driver queries this table for data related to a given user and files the SQL uses. See Work with Security in the *JD Edwards World World Writer Guide* for more information.

When you use an excluded column in an SQL statement the JDBC driver responds differently depending on the context. When you use the excluded column:

- As a column in the SQL statement Select List, the ResultSet always returns NULL for the ResultSet column value.
- In any other form, such as a Where Clause or Order By Clause, the World JDBC driver issues a parsing error and disallows the SQL statement.

3.1.4 File/Field Security for Insert/Update/Delete

The JDBC driver performs File/Field Security based on information in the File/Field Level Security File (F9401) and the Software Versions Repository (F9801). File/Field Level Security is required to allow the JDBC driver to perform insert, update and delete actions to files other than Batch Input Files (those files with a Function Use Code between 231 and 239 in the Software Versions Repository - F9801). You are allowed to insert/update/delete records in Batch Input Files by default.

To set up File/Field level security for a particular file

1. Navigate to the File/Field level Security maintenance program, Option 15 on the World Writer Advanced Operations Menu (G8231).
2. Add security records by user and file to allow insert/update/delete for any file which that user requires updating through the JDBC driver. To allow insert or delete, all fields in the file must be marked Y for Update. If the file will be updated only, only those fields which you will be updating need to be marked Y for Update.

Figure 3–1 Field Level Security screen

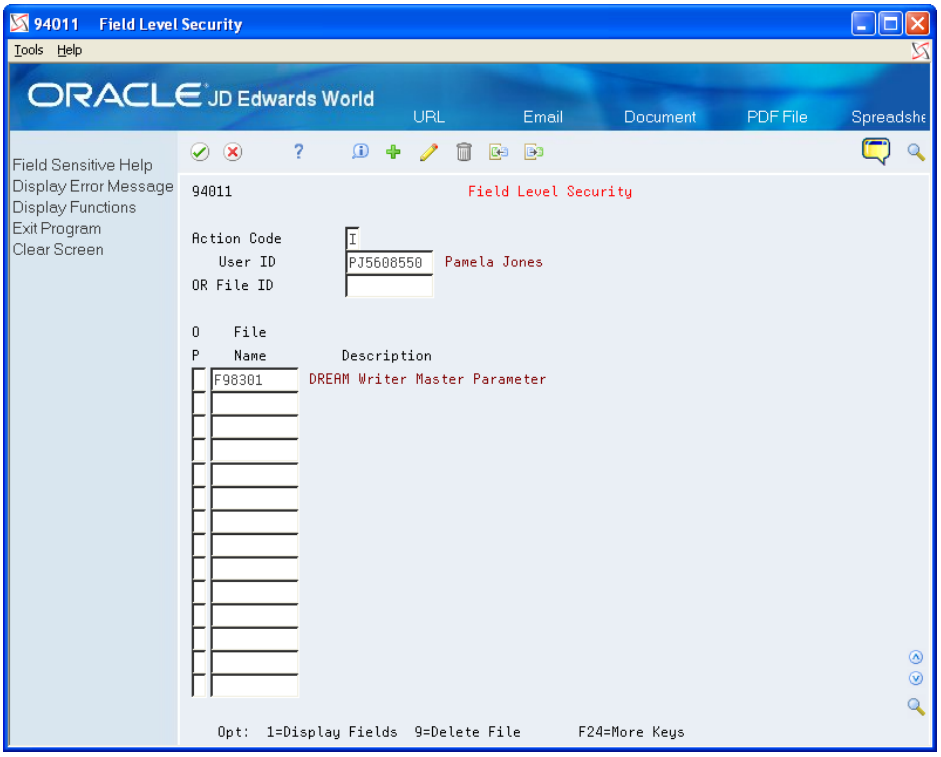
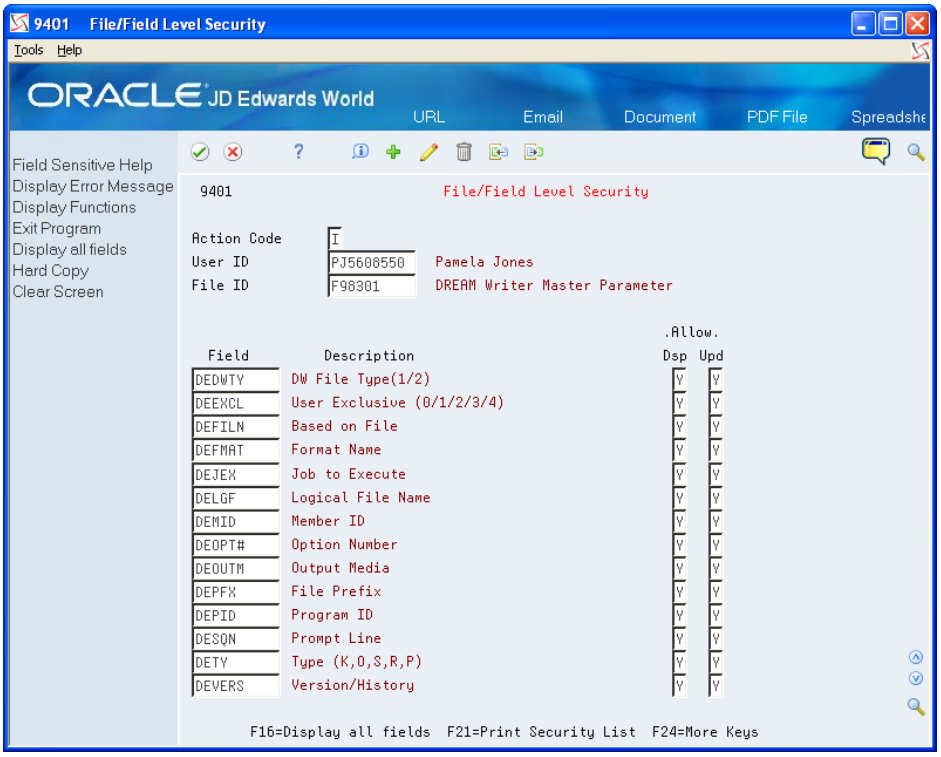


Figure 3–2 File/Field Level Security screen



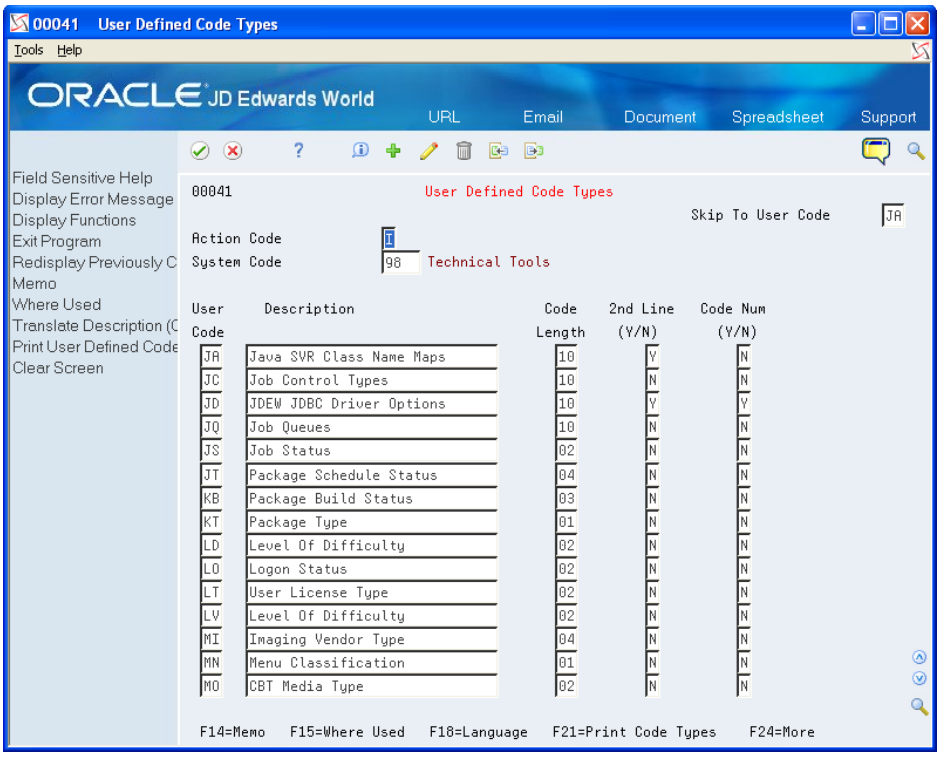
3.1.5 Action Security

The JDBC driver performs Action Security based on information in the Software Version \Repository File (F9801), Action Code Security File (F0003) and the 98/JA User Defined Code. Action Security is required to allow the JDBC driver to perform insert, update and delete actions.

To set up Action Security for a particular Java application

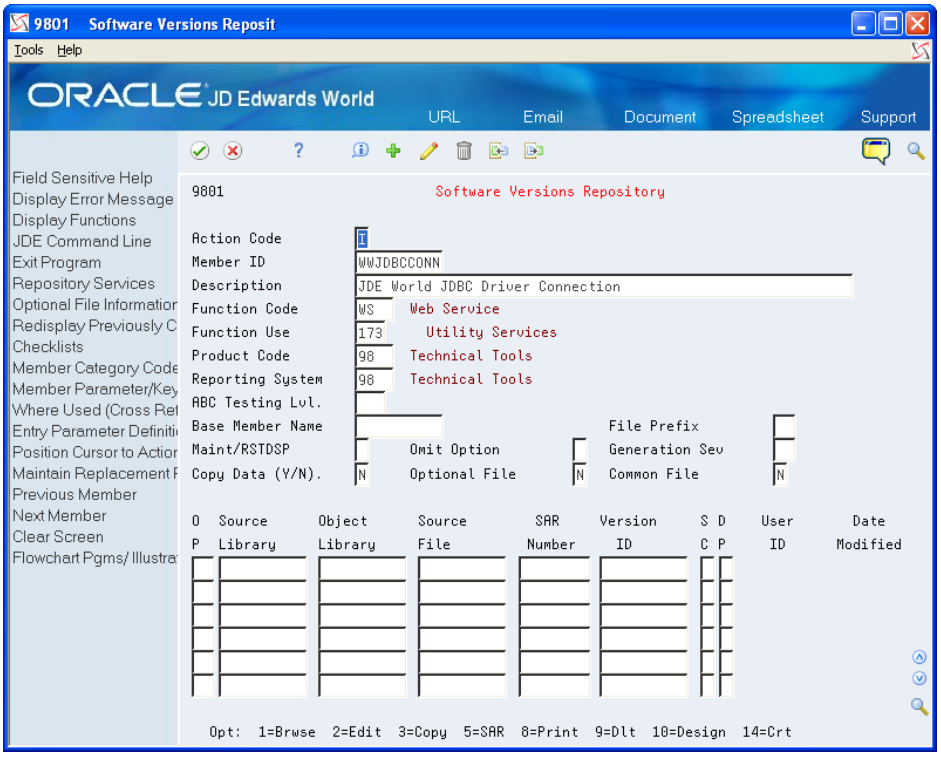
1. Create UDC code 98/JA which defines the Java Program map UDC table. The purpose is to provide a cross-reference between a World SVR object name and the Java class names. The Code length must be 10, Second Line must be "Y" and Code Num must be "N".

Figure 3–3 User Defined Code Types screen



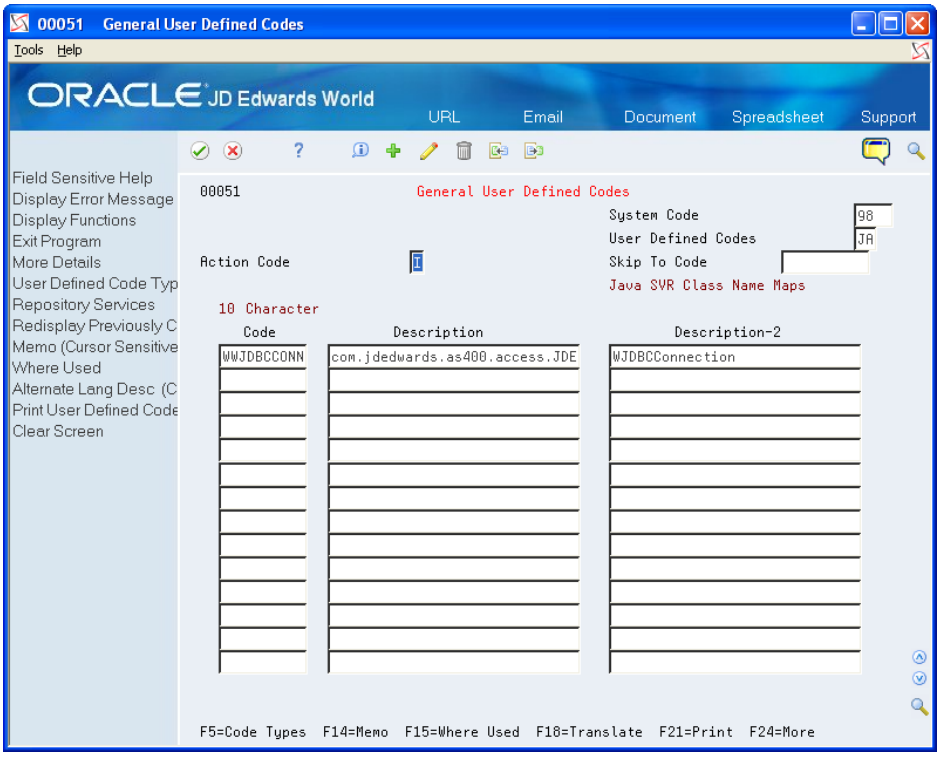
2. Create an SVR Entry to define an object name (Member ID) that represents the Java class name. This name may be anything you want, but Oracle recommends that you make it something that will relate to the Java application.

Figure 3–4 Software Versions Repository screen



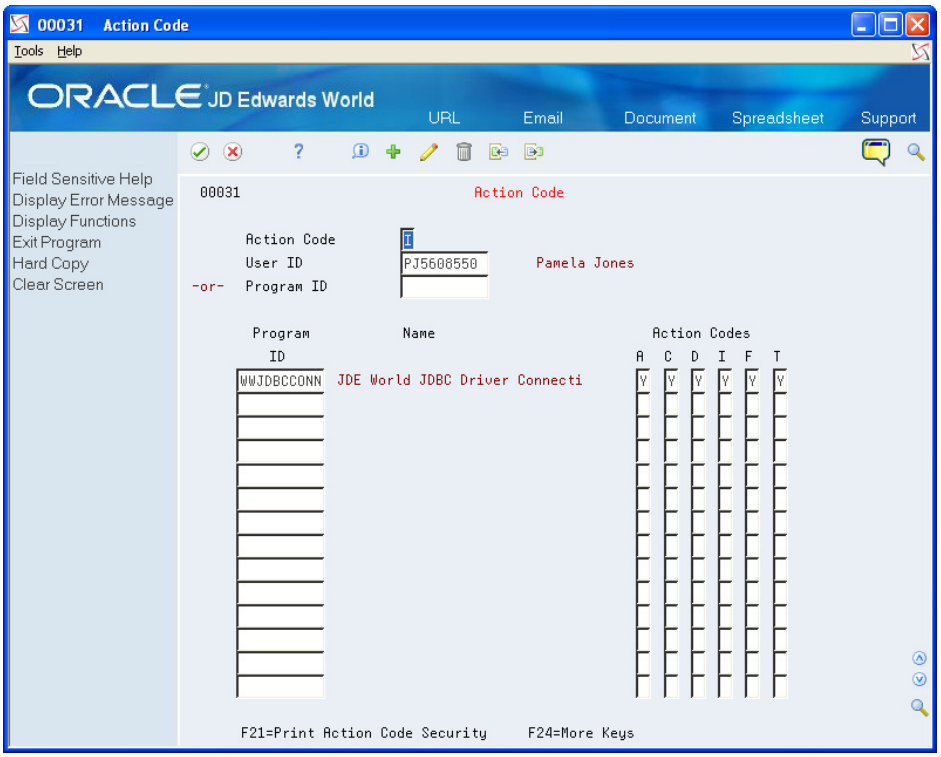
3. Create a UDC Table entry in UDC 98/JA to map a Java class to its corresponding SVR entry. This entry will map the SVR object name to a specific Java class name. Enter the name you used for the SVR Member ID in the 10 Character Code. The Description and Description-2 fields are concatenated together to make up the Java class name. Remember that Java class names are case sensitive.

Figure 3–5 General User Defined Codes screen



4. Add an Action Code security record for the User and Java application in the Action Code Security file: Only the A, C and D Action Codes are used by the JDBC driver, for Insert, Update, and Delete respectively.

Figure 3–6 Action Code screen



3.2 World JDBC Driver and JD Edwards World Data-Specific Features

The JDBC driver includes the following data-specific features.

3.2.1 Metadata Files

These features are operational with all releases from A7.3 forward that JD Edwards World supports. These features affect any metadata that the JDBC driver extracts from the system.

The JDBC driver maintains release independence by using SQL to extract data from the following tables. This eliminates 99% of any compatibility issues with the data files. Each of these tables must exist in the user's environment for the driver to function properly.

Specification Type	Metadata Source Table	Release Dependencies
JD Edwards World User and Environment	User Profile F0092/F0093/F0094/F00944	None
Business Unit Security	F0001	None
Column Security	F9401 (prior to A93) F8202 (A93 and later)	None
Company Constants	F0010	None
Currency Codes	F0013	None
Data Item Master	F9200 F9201 or F9210 F9202 F9203	The JDBC driver determines which table to use by attempting to locate the F9210 table. If it locates this table in the library list then the JDBC driver assumes that the environment release is A8.1 or later and uses this table. If it cannot locate the F9210, then it assumes that F9201 is the active Data Item Master. Expect all dictionary tables to be in the same data library.
Payment Terms	F0014	None
Table Definitions	SYSTABLES and F9801	None
User Defined Codes	F0005	None
User Display Preference	F00921	None
Table Column Definitions	SYSCOLUMNS	None
Server	N/A	The JDBC driver extracts the server name from the connection data.

3.2.2 Environments and Specifications

The JDBC driver must be able to identify the user's current JD Edwards World environment. Any user login that can log into the JD Edwards World software system using their normal user interface, can also log using the JDBC driver. As long as the JD Edwards World environment contains the minimum required files from the metadata source table, a user should be able to connect using the JDBC driver. The JDBC driver's ability to identify a JD Edwards World environment follows the same logic that the JD

Edwards World J98INIT and J98INITA programs use. The user can choose only one environment for a given database connection.

The JDBC driver determines the user from the connection object which is available after a login and creates the connection object. A jt400.jar API extracts the IBM user profile (USRPRF) from the IBM i. This allows access to all information for the current user. You can identify the JD Edwards World software instance by reviewing the user profile Initial Program Library of the connection. The JDBC driver examines the QJDF data area in this library to determine:

- The library in positions 131-140 of the QJDF data area is the JD Edwards World Object library for the instance.
- The library in positions 181-190 of the QJDF data area is the JD Edwards World Security library for the instance.
- The JD Edwards World Environment data is in the JD Edwards World Security library.
- The JDBC driver creates new objects in the he JD Edwards World Object library.

Within the JD Edwards World Security library tables are the valid environments the user has authority to use. The JDBC driver assumes that the following tables are in the library: F0092, F0093, F0094, and F00944. If any of these tables are missing from the library found at position 181 of the QJDF data area then the user cannot connect.

- If the user's initial program is J98INIT, then the correct environment name is the same as the user login name. JD Edwards World software permits only one environment in this scenario.
- If the user's initial program is J98INITA, then the JDBC driver can use any environment name for the user from the F0093 table.

The JDBC driver makes an environment choice for a connection via the driver property JDEWEnvironment. The JDBC driver identifies and constructs the environment choice. The connection library list is the concatenation of the system library list found in the IBM i QSYSLIBL system value and the libraries in the environment you choose. If the JD Edwards World Object library in the QJDF data area is not currently in the user library list, then the JDBC driver adds it at the bottom of the list. The JDBC driver applies the Library list to the current user connection by disconnecting the initial database connection and reconnecting it using the original URL with the properties in the original connection. It also changes the IBM JTOpen libraries property to that of the JD Edwards World environment. The JDBC driver extracts table specifications for all tables for all libraries in the library list. The JDBC driver performs this only once per library. If the JDBC driver extracts specifications for a library with the first user, then when the next user logs in, the JDBC driver already contains the table specification in the cache.

> **Note:** The SYSTABLES and SYSCOLUMNS database engine files support the driver. These files exist in the QSYS2 library and QSYS2 must exist in either the QSYSLIBL system value or in the JDEWEnvironment library list definition.

Table specification definitions for tables in the environment:

- The JDBC driver restricts all client SQL queries to the IBM i to libraries and files in the JDEWEnvironment library list. The JDBC driver processes each SQL statement and verifies the tables you request against the user's environment library list.

- The JDBC driver extracts table specifications for tables in the library list. The driver performs this only once per library.
  - If you indicate a specific schema (library) for the table in the SQL statement, for example select * from JDFDTA.F0101, then the JDBC driver validates the schema against the environment library list. If the library does not exist in the user library list then the JDBC driver delivers an SQL Exception and rejects the SQL statement.
  - If you do not provide a schema, for example select * from F0101, then the JDBC driver uses the first occurrence of the table in the library list.
  - If the table does not exist in the user library list then the JDBC driver delivers an SQL Exception and rejects the SQL statement.
- The JDBC driver identifies the exact instance of the table, extracts table and column definitions, and specifications from the SYSTABLES and SYSCOLUMNS database tables.

4

World JDBC Driver Data Features

This chapter contains theses topics:

- Section 4.1, "JDBC Driver Overview,"
- Section 4.2, "World JDBC Driver Data-Specific Features,"
- Section 4.3, "Other Considerations."

4.1 JDBC Driver Overview

The JD Edwards World JDBC driver can be used by any Java-based reporting tool or software application that needs to access business data using JDBC. A good example for JD Edwards World customers is Oracle Business Intelligence (BI) Publisher.

The World JDBC driver automates most of the work of transforming World data to a usable form that can be used by your Java based business applications, work that would otherwise require additional custom software development to perform similar tasks. This saves the JD Edwards World customer time and effort to create Java based software and reports.

In addition to the security features that were described in chapter 3, the driver handles data transformation of JD Edwards data as it is stored in the World database automatically. However, in order to make proper use of the World JDBC driver, users should understand how SQL is to be used within the World JDBC driver. This chapter illustrates how the World JDBC driver handles the processing of particular aspects of JD Edwards World data and how to use SQL to obtain the correct data from the JDE World database.

4.2 World JDBC Driver Data-Specific Features

World JDBC Driver data-specific features include:

- Presumptive joins
- Data transformations
- Identifying tables, columns, and data dictionary items
- Decimal position placement
- Julian date conversion

4.2.1 Presumptive Joins

In the JD Edwards World product, Presumptive Joins are used within the World Writer application to quickly allow users to pull in descriptions from parent tables for code

and ID values in child tables (foreign key references). The JD Edwards World JDBC driver provides a similar feature that makes it easy to add the descriptions for specific tables.

JDBC Presumptive Join capability is dependent upon the information in the JD Edwards World Data Dictionary tables. It is in effect for any column in the database where the related data dictionary item is defined as:

- Data Edit Rules. . UDC (any UDC code)
- Data Edit Rules. . FILE F0010 (Company Codes)
- Data Edit Rules. . FILE F0013 (Currency Codes)
- Data Edit Rules. . FILE F0014 (Payment Terms)

If you use presumptive joins, then the supporting tables indicated above must exist in the user library list.

To use a presumptive join, use any text string that begins with 'VC0' in a separate column immediately following the data column for which you wish to have a text description. For example:

```
select
SDKCOO, 'VC00001',
SDDCTO, 'VC00002',
SDBCRC, 'VC00003',
FROM F4211
```

will return results that look like those in this table:

SDKC000	00002	SDDCTO	00004	SCBCRC	00006
10	Company 10	SO	Sales Order	BEF	Belgian Francs
43	Intergalatic Megacorp. Inc.	SD	Direct Ship	UFO	Martian Monetary Unit
43	Intergalatic Megacorp. Inc	SO	Sales Order	?	
43	Intergalatic Megacorp. Inc	SO	Sales Order	USD	U.S. Dollar
63	Company 00063	VS	Purchasing Return - Brazil	BRL	Brazilian Real
67	Feimo's test	SO	Sales Order	ARA	Argentina Peso
100	Model Finan/Distrib Co (Mktg)	SD	Direct Ship	USD	U.S. Dollar
100	Model Finan/Distrib Co (Mktg)	SI	Interbranch Sales	USD	U.S. Dollar
100	Model Finan/Distrib Co (Mktg)	SO	Sales Order	ECU	European Currency Unit
100	Model Finan/Distrib Co (Mktg)	SO	Sales Order	GBP	Pound Sterling

If the preceding column does not support presumptive join, then the presumptive join text, (for the above example, 'VC00001'), will be returned for all values of the trailing column.

If the preceding column does support presumptive join but the code value does not exist in the lookup table, then the presumptive join text will return a null value.

4.2.2 Data Transformations

The main purpose of all JDBC data transformations is to present and use all database references to a database column from the end user perspective. The JDBC driver performs these data transformations automatically and the user or the client application should not need to manipulate data values in order for the system to recognize the meaning. That is, dates display as dates, currency displays as currency, and numbers have correct decimal positions. Users should be able to create SQL statements which are easy to read, write, and reflect common business values.

JD Edwards World currently supports three types of data transformations based on JD Edwards World Data Dictionary definitions of the table column:

- Decimal position placement
- Date translation
- Alternate nomenclature

Consider the following SQL example statement:

```
select shotot , shtotc, shcrr, shdrqj  from  F4201
where  shdrqj  between '18/08/2006'  and '23/08/2006'
and shotot > 1000.00  and SHCRR >0
fetch first 5 rows only
```

4.2.2.1 IBM i

This SQL statement extracts all records in the six day date range between Aug 18, 2006 and Aug 23, 2006 and whose total dollar value exceeds $1000.00.

shotot	shtotc	shcrr	shdrqj
1100.00	0.00	2.0000000	18.08.2006
12500.00	0.00	0.5000000	23.08.2006
17500.00	0.00	0.5000000	23.08.2006
17500.00	0.00	0.5000000	23.08.2006
12500.00	0.00	0.5000000	23.08.2006

The JDBC driver converts the inbound date format (DDMMYYYY in this case, where Y represents year, M for month and D for day) to the JD Edwards World Julian date format.

Note: Your specific date format will depend on your connection definition, the JD Edwards World user preferences and the IBM i system values. In general terms, the date format will be the same as you use within JD Edwards World, unless it is specifically overridden with the "data format" connection property.

Finally, users should have the option of referencing objects in the system using an alternate nomenclature. If users are not familiar with the JD Edwards World database structures, an alternate nomenclature allows users to refer to the tables and columns in the JD Edwards World database using descriptive text that is in the driver metadata.

4.2.3 Identifying Tables, Columns, and Data Dictionary Items

- JD Edwards World tables are tables that exist in JD Edwards World Software Versions Repository (SVR), for which an F9801 table entry exists. The driver only revises values for JD Edwards World tables.
- The JDBC driver identifies the JD Edwards World Data Dictionary item for a table column by removing the first two characters of the table column name. Therefore, Tables and table columns must follow the standard table definition rules for JDE table columns. Typically, the columns in the table use the following format:
  - ttdddd

    where
  - "tt" is the table prefix for table name as defined in the table SVR record and
  - "dddd" is the data dictionary data item JD Edwards World uses to define the table columns.
- The exception to this is a JD Edwards World field reference file. The JDBC driver uses columns names in JD Edwards World field reference files (F98FRF* files) as you define them in the file because they represent the data dictionary item names without a prefix.
- The JDBC driver considers only data dictionary items in glossary group D or S. The JDBC driver excludes all other data dictionary items.
- If the JDBC driver cannot locate a data dictionary item for a column in a JD Edwards World table, then it makes no changes to the JD Edwards World table column data; the data is passed to the client application unaltered.

4.2.4 Decimal Position Placement

One of the most difficult issues to address with the use of the JDBC driver with the JD Edwards World database is the placement of decimals for numeric values. The JD Edwards World JDBC driver handles this automatically for the driver users. Additionally, the decimal placement position is based on the data dictionary of the current connection environment. This allows client software that uses the driver to remain independent of the current setup of the environment being used. This is particularly useful for World instances that use different data dictionary information in their environments.

Again, all of this is handled automatically, and it is not necessary for the average user to worry about decimal placement. This topic describes in detail how the JDBC driver task is performed.

4.2.4.1 Definitions of Decimal Placement

In computer terminology, every number is defined by its numeric data type, precision, scale, and numeric value itself.

- Numeric data type represents how a numeric data value is represented in memory or saved on disk: On the IBM i, the valid values for numbers are: Packed Decimal data, Zoned Decimal data, Binary data, Floating Point data.
- The precision of a number is how many significant digits are in the number. This is the total number of digits that a number can have both left and right of the decimal place.
- The scale of a number is the number of digits to the right of the decimal place.
- The value is the actual data value such as 100.00

The first three terms are used to define a database column where a number is saved and the last is the actual value itself. Therefore a database column that has a maximum data value of 99999.999 would have the following characteristics:

Datatype: Might be Zoned Decimal in this example

Precision: 8

Scale: 3

A specific data row in the table might have the value of 100.000 in this column.

4.2.4.2 JD Edwards World Data Dictionary

In the JD Edwards World software, a numeric data value can have two different representations or perspectives:

- The user's perspective is the intrinsic business value of the number. It is also called the "Display" value.
- The "database" perspective is how the value is stored in the data files.

The benefits of having two perspectives within the JD Edwards World software system is that JDE can provide very flexible software that can easily adapt to many different applications and international requirements. However, having a "Display" perspective that differs from the "database" perspective make it difficult for client applications to use common database tools such as SQL to access the data. The database number values are not necessarily the correct intrinsic business value.

The display and database perspectives for values are defined by the associated data dictionary item for number represented in the JDE software system. The data dictionary identifies data type (FRDTAT), the value precision (FRDTAS), the display decimal (FRCDEC), and the file decimal (FRDTAD) values that represent a number in the JD Edwards World software system.

The numeric value from the user perspective is the display value and is defined as:

- numericDataType (precision, displayDecimal)

  or

- FRDTAT (FRDTAS, FRCDEC)

The numeric value that resides in the data file is the file decimal representation of the number and is:

- numericDataType (precision, fileDecimal)

  or

- FRDTAT (FRDTAS, FRDTAD)

Therefore, whenever the JDBC driver references a number column in the database it must adjust the value by a *scale offset*. The scale offset is always a power of 10 and the JDBC driver uses it as a multiplier or divisor in the column value calculations. The formula for the scale offset that the JDBC driver must apply is:

- *scaleoffset* = 10 (Display Decimal minus File Decimal)

  or

- *scaleoffset* = 10 (FRCDEC - FRDTAD)

The JDBC driver performs decimal position changes by altering the inbound SQL statement to reflect the expectations of the database engine and revises the ResultSet

information that you expect in the result set. The JDBC driver performs both of these tasks in the following manner:

- Revises the ResultSet values to report the correct scale.
- Revises the ResultSetMetaData so that it reports the correct column scale and precision.
- Changes outbound SQL statement column references, such as SelectList items, from:
  - columnName

    to
  - CAST(columnName / (scaleOffset) AS DECIMAL(FRDTAS, FRCDEC)) AS columnName

Changes inbound SQL statement column references, such as those in the WHERE and JOIN clauses, from:

- columnName

  to
- columnName * scaleOffset

Revisions are not made to the GROUP BY or ORDER BY clause of the SQL statement.

The following example illustrates how the JDBC driver converts a value to the user perspective. The dictionary item OTOT has the following data dictionary information:

OTOT Type	P
OTOT Size	15
OTOT Data File Decimals	0
OTOT Display Decimals	2

The scale offset then is:

- 2-0 = 2

If the original SQL statement is:

```
SELECT SHOTOT FROM F4201
```

Then the driver revises the SQL statement to become:

```
SELECT CAST (SHOTOT/100 AS DECIMAL (15, 2)) AS SHOTOT FROM F4201
```

The original ResultSet:

SHOTOT
20900

Is then returned as:

SHOTOT
209.00

4.2.4.3 Decimal Position Relational Expression Considerations

A relational expression is any SQL statement clause which compares two values. You use relational expressions extensively in JOIN and WHERE clauses to match, evaluate, and select data rows. There are performance considerations that you must be aware of when considering changes to SQL statement relational expressions because the DB2 for IBM i evaluates calculations for every record. To minimize these issues, the following rules apply:

- When you use a constant in a relationship (SHOTOT > 10.00), the JDBC driver revises the constant value and not the data column in the comparison. For decimal scale conversions, this means that the JDBC driver multiplies the constant value by the scale offset.
- For relationships between two columns with the same scale offset, the JDBC driver makes no changes to either side of the relationship. Since both columns have the same relative decimal position, the comparison succeeds without revising the database values.
- If the two columns have different scale offsets, then the JDBC driver adjusts the column with the smaller offset by the difference between the two offsets. In this manner, changes are only necessary to one column instead of both columns which minimizes the performance impact of the relationship comparison.
- JD Edwards World disabled and does not support some highly complex SQL statement relationship structures. In these situations, the JDBC driver issues an SQL exception error.

4.2.5 Julian Date Conversion

- JD Edwards World stores most date values in the database as a JD Edwards style Julian date. Some exceptions exist such as the use of text based date columns for Z-files.
- Julian date columns have a data dictionary Data Item Class of type "DATEW" (i.e. FRCLAS is set to "DATEW")
- JDE Julian date format is not the same as IBM Julian date format:
  - JDE Julian date = (IBM Julian date - 1900000)
- JD Edwards World stores date format as a CYYDDD format where:
  - C = the century where 0=1900 and 1 = 2000
  - YY is the year within the century
  - DDD is the nth day of the year
  - The CYY value can also be thought of as current year minus 1900, therefore 2010 - 1900 = 110.
- When the JDBC driver uses Date values in a ResultSet object, if the database column is based on a data Dictionary item with a DATEW class dictionary item, the JDBC driver reformats the Julian date column to either a numeric representation or a textual user date format. The actual date format value will be presented based on the chosen data format of the connected user.

- Beginning with JDBC 3.1, JDBC considers all DATEW database columns to be of database datatype DATE. The value is returned as a java.sql.Date object.
- If the JDE date value is 0, then a null date is returned by the JDBC driver.
- If the database contains a data value that is not a valid JDE Julian date , then the JDBC driver returns a null data value.

The following example illustrates how the JDBC driver retrieves a value from the database and converts it to reflect the user display perspective. Assume that the date format for the connection is set to "MDY" and the date separator is "-".

Users SQL statement request:

```
SELECT SHUPMJ FROM F4201 where SHUPMJ = '01-31-2006'
```

Revised SQL:

```
SELECT case when SHUPMJ=0 then NULL
when SHUPMJ=999999 then date('9999365')
else date(char(SHUPMJ+1900000)) end   AS "SHUPMJ"
FROM F4201
WHERE SHUPMJ =  106031
```

Original ResultSet returns a:

SHUPMJ
106031

Revised ResultSet as seen by user. Note, the actual text result will depend on whether the application software requests the ResultSet value as a string representation or as a number:

SHUPMJ as string
01/31/2006

4.2.5.1 Julian Date Relational Expression Considerations

Constant value Relationships (SHUPMJ > '12-31-1995')

- When the SQL statement uses a constant in a relationship, the JDBC driver revises the constant value and not the data column in the comparison. For JD Edwards World date columns, the JDBC driver converts date numbers to Julian date format. If the constant value is not a valid date, the JDBC driver issues an SQL exception error.
- When relationships exist between two date columns, the JDBC driver makes no changes to either side of the relationship. Since both columns have the same relative data values, the comparison succeeds without revising the database values.
- When relationships exist between a JD Edwards World date and a non-date column, the JDBC driver issues the following SQL exception message:
  - JDE: Incompatible data types for Date comparison

4.3 Other Considerations

The JDBC driver manages database object names that contain special characters, for example @1QRYG in F82100. The JDBC driver treats all column and table object references as string object data.

The JDBC driver supports any JD Edwards World table that uses JD Edwards World data dictionary to define the table columns. The JDBC driver only considers tables that use a field reference file which maps to JD Edwards World Data Dictionary for data transformation.

The JDBC driver operates independent of the JD Edwards World release level. The JDBC driver extracts table specifications for all files from the DB2 SYSTABLES and SYSCOLUMNS system tables. This provides release independence for the table structures. All access to data files is via SQL statements and SQL is not dependent on file record formats or format structure. The JDBC driver achieves release independence for all other specification files by extracting specifications from the appropriate JD Edwards World table in the user's connection library list. As the data dictionary files changed between A7.3 and A8.1, the driver searches for the F9210 table in the library list. If it detects F9210 table, it assumes this is an A8.1 or later release and use the F9210 table for data dictionary information.

The JDBC driver dynamically identifies and extracts database specifications (table specs) and identifies JD Edwards World Data Dictionary specifications for a table attribute. The JDBC driver extracts table specifications for all files from the SYSTABLES and SYSCOLUMNS DB2 for i tables. This provides release independence for the table structures.

The JDBC driver is able to handle multiple data item prefixes in the same file, such as with the Z-files, with tables that can contain SHUPMJ, and SDUPMJ columns in the same physical or logical table definition.

Index

M

N

O

P

R

S

T

U

W