

---

---

# Configuring and Using the JDBC PDS

The JDBC pluggable data source (PDS) enables you to access any JDBC sources, such as:

- An RDBMS like Oracle, DB2, Sybase, or SQL Server
- A non-relational data source like Microsoft Excel
- Any ODBC data source through the JDBC-ODBC bridge

The JDBC PDS is installed, by default, with Oracle9i Reports to allow access to all of the JDBC supported data sources.

This chapter contains the following main sections:

- [JDBC Configuration File](#)

This section contains the necessary information for you to configure your drivers for use in the JDBC PDS.

- [Defining and Running a JDBC Query](#)

This section contains information on how to define and run a query against a JDBC data source.

- [Troubleshooting Information](#)

This section contains information on how to troubleshoot JDBC related issues.

- [Adding your own PDS](#)

This section contains information on how to implement your own PDS for use in Oracle9i Reports.

## JDBC Configuration File

The `jdbcpds.conf` file, located in the `ORACLE_HOME\reports\conf` directory is Oracle9i Reports' JDBC PDS configuration file. This file is pre-configured for the Merant DataDirect drivers provided by Oracle. You need to add or modify relevant entries in the `jdbcpds.conf` file to include any other JDBC drivers that you want to use.

Reports Builder displays a list of drivers in the **JDBC Query Connection** dialog based on the entries in the `jdbcpds.conf` file. Use this list to select specific drivers for your report's JDBC query.

Reports Builder reads and caches the entries in the `jdbcpds.conf` when it is invoked. Restart Reports Builder to view the result of any changes made to the `jdbcpds.conf` file, e.g., adding a new JDBC driver entry.

The `jdbcpds.conf` file has two sections:

- An Internal DTD section describing the XML format and driver configuration information

---

---

**Caution:** This section should not be modified.

---

---

- An XML section detailing the driver information like driver name, connect string format, driver class, etc.,

---

---

**Note:** You can modify or add your driver information in this section.

---

---

### Example

The following sample illustrates the contents of the `jdbcpds.conf` file:

```
<!-- DTD section - Not to be modified -->

<!DOCTYPE jdbcpds [
<!ELEMENT jdbcpds (driverInfo)>
<!ELEMENT driverInfo (driver+)>
<!ELEMENT driver (property*)>
<!-- ATTLLIST driver
      name          CDATA #REQUIRED
      mainProtocol   ( jdbc ) "jdbc"
      subProtocol    CDATA #REQUIRED
      connectString  CDATA #REQUIRED
```

```
        class          CDATA #REQUIRED
        connection     CDATA #REQUIRED
        loginTimeout   CDATA "5"
    >
<!ELEMENT property EMPTY>
<!ATTLIST property   name CDATA #REQUIRED
                  value CDATA #REQUIRED >

]>

<!-- Add or modify the following section for your driver information -->
<!-- Following drivers are available out-of-box in 9iAS -->

<jdbcps>
<driverInfo>
  <driver name = "oracleThin"
    subProtocol = "oracle:thin"
    connectString = "mainProtocol:subProtocol:@databaseName"
    class= "oracle.jdbc.driver.OracleDriver"
    connection = "oracle.reports.plugin.datasource.jdbcps.
      JDBCConnectionHandling">
  </driver>

  <driver name = "oracle"
    subProtocol = "oracle:oci8"
    connectString = "mainProtocol:subProtocol:@databaseName"
    class = "oracle.jdbc.driver.OracleDriver"
    connection = "oracle.reports.plugin.datasource.jdbcps.
      JDBCConnectionHandling">
  </driver>

  <driver name = "jdbc-odbc"
    subProtocol = "odbc"
    connectString = "mainProtocol:subProtocol:databaseName"
    class = "sun.jdbc.odbc.JdbcOdbcDriver"
    connection = "oracle.reports.plugin.datasource.jdbcps.
      JDBCConnectionHandling">
  </driver>

  <driver name = "sqlserver-merant"
    subProtocol = "merant:sqlserver"
    connectString = "mainProtocol:subProtocol://databaseName"
    class = "com.oracle.ias.jdbc.sqlserver.SQLServerDriver"
    connection = "oracle.reports.plugin.datasource.jdbcps.
```

```

        JDBCConnectionHandling">
</driver>

<driver name = "sybase-merant"
        subProtocol = "merant:sybase"
        connectString = "mainProtocol:subProtocol://databaseName"
        class = "com.oracle.ias.jdbc.sybase.SybaseDriver"
        connection = "oracle.reports.plugin.datasource.jdbcpds.
        JDBCConnectionHandling"
        loginTimeout = "0">
</driver>

<driver name = "db2-merant"
        subProtocol = "merant:db2"
        connectString = "mainProtocol:subProtocol://databaseName"
        class = "com.oracle.ias.jdbc.db2.DB2Driver"
        connection = "oracle.reports.plugin.datasource.jdbcpds.
        JDBCConnectionHandling"
        loginTimeout = "0">
</driver>

<driver name = "informix-merant"
        subProtocol = "merant:informix"
        connectString = "mainProtocol:subProtocol://databaseName"
        class = "com.oracle.ias.jdbc.informix.InformixDriver"
        connection = "oracle.reports.plugin.datasource.jdbcpds.
        JDBCConnectionHandling">
</driver>

</driverInfo>
</jdbcpds>

```

Table 1–1 outlines the various attributes that can be associated with a driver.

**Table 1–1 Driver Attributes**

Attribute Name	Description	Sample
name	A unique user-defined value used to refer to a specific JDBC driver in Oracle9i Reports.	sybase-merant

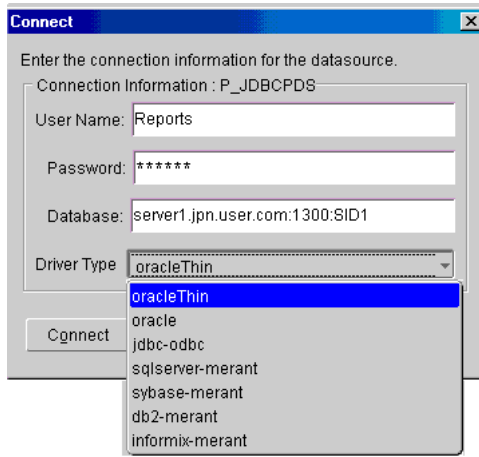
**Table 1–1 Driver Attributes**

Attribute Name	Description	Sample
subProtocol	Driver sub protocol added with the database URL before creating a database connection. This is driver specific information and can be found in the driver documentation. Example: The sub protocol used for connecting to the Merant driver: <ul style="list-style-type: none"> <li>■ Sybase is merant: sybase</li> <li>■ SQL Server is merant: sqlserver</li> </ul>	merant: sybase
connectString	Format of the driver's connect string format is mainProtocol: subProtocol://databaseURL. For example, jdbc: subProtocol://databaseName. Do not specify the actual values for subProtocol or databaseName, use the fixed placeholder names instead.	mainProtocol: subProtocol://databaseName
class	Driver class name used to register (REPORTS_CLASSPATH) and load the driver. This is driver specific information and can be found in the driver documentation.	com.oracle.ias.jdbc.informix.InformixDriver
connection	Driver's connection handling class. The JDBC PDS can have different connection handling classes for each driver. Oracle9i Reports' default connection handling class, which is sufficient for most drivers, is <i>oracle.reports.plugin.datasource.jdbcpds.JDBCConnectionHandling</i> . Please refer to the Reports Java document, for more information on how to extend your JDBC Connection class	oracle.reports.plugin.datasource.jdbcpds.JDBCConnectionHandling
loginTimeout (Optional)	Driver specific parameter. Specify the value in seconds. Please refer to the driver documentation for more information.	0
property	Specify any additional properties of your driver as <i>Attribute Name</i> and <i>Value</i> .	-

When you submit your report's connection details, the connection information is combined with the driver's configuration information specified in the `jdbcpds.conf` file. The resulting connection information is submitted to the

database as a complete connection URL. Refer to [Table 1–2](#), [Table 1–3](#), [Table 1–4](#), [Table 1–5](#), and [Table 1–6](#) for more information on sample connection information. [Figure 1–1](#) shows a list of all drivers configured in the `jdbcpds.conf` file.

**Figure 1–1** JDBC Connect dialog in Reports Builder



## Verifying Pre-installed Driver Entries

Drivers like SQL Server and Excel with `jdbc-odbc`, Oracle JDBC Thin, and Oracle JDBC OCI (thick) are installed and configured with Oracle9i Reports. These drivers do not require any additional `.jar` files to be installed.

- Oracle JDBC Thin driver
- Oracle JDBC OCI (thick) driver
- `jdbc-odbc` driver

You can use SQL Server / Excel with the `jdbc-odbc` driver. This entry is pre-configured in the `jdbcpds.conf` file. Before you can use SQL Server or Excel with `jdbc-odbc`, you need to create an ODBC data source. Refer to Windows help, for more information on how to create an ODBC data source.

---

---

**Note:** Oracle9i Application Server provides Merant DataDirect drivers which can also be used to access SQL Server.

---

---

## Installing and Configuring Merant DataDirect drivers

Oracle provides a set of Merant DataDirect drivers (Version 3.2) that can be downloaded from OTN (<http://otn.oracle.com>). The driver configuration file, i.e., `jdbcpds.conf` contains relevant entries for the Merant DataDirect drivers. Additionally, the JDBC Connect dialog (Table 1-1) lists the entries for the set of Merant DataDirect drivers provided by Oracle.

However, you need to install the appropriate `.jar` files and specify them in Oracle9i Reports specific classpath entries, in order to make them available to Reports Builder and Oracle9iAS Reports Services

The drivers provided by Oracle for use with Oracle9i Application Server / Oracle9i Developer Suite, version 9.0.4, are:

- [Sybase Driver](#)
- [DB2 Driver](#)
- [SQL Server Driver](#)
- [Informix Driver](#)

You can also install and configure a [Custom Driver](#) for use with Oracle9i Application Server and Oracle9i Developer Suite.

The following procedure outlines the generic steps involved in configuring the Merant DataDirect drivers. To configure specific Merant DataDirect drivers refer to the appropriate sections.

To configure the Merant DataDirect drivers:

1. Install the relevant `.jar` files in your Oracle9i Application Server and Oracle9i Developer Suite directory.
2. Include an entry in the `REPORTS_CLASSPATH` to make the files available to Reports Builder and Oracle9iAS Reports Services. Refer to the relevant driver in this section for information on the required `.jar` files.
  - a. **Reports Builder:** Prefix the driver location to the existing entries in `REPORTS_CLASSPATH`. This variable is located in the registry for Windows users and in the `reports.sh` file for Solaris users. Refer to the relevant driver in this section for an example.
  - b. **rwbuilder.conf:** Append the driver location to the engine `classPath` attribute in the `rwbuilder.conf` configuration file. Refer to the relevant driver in this section for an example.

- c. **Reports Server:** Append the driver location to the `classPath` attribute of the engine, in the Reports Server configuration file. Refer to the relevant driver in this section for an example
- d. **`jdbcpds.conf`:** Located in the `ORACLE_HOME\reports\conf` directory. Refer to [Table 1–1](#) for more information on the parameters. Refer to the relevant driver in this section for an example.

## Sybase Driver

1. Install the relevant `.jar` files in your Oracle9i Application Server and Oracle9i Developer Suite directory.

Jar files required: `YUtil.jar`, `Ysybase.jar`, and `Ybase.jar`.

2. Include an entry in the `REPORTS_CLASSPATH` to make the files available to Reports Builder and Oracle9iAS Reports Services.

- a. **Reports Builder:** Prefix the driver location to the existing entries in `REPORTS_CLASSPATH`. This variable is located in the registry for Windows users and in the `reports.sh` file for Solaris users.

**Example:**

```
D:\sybase_installed\YUtil.jar;D:\sybase_
installed\Ysybase.jar;D:\sybase_installed\Ybase.jar;existing classpath
entries
```

- b. **`rwbuilder.conf`:** Append the driver location to the engine `classPath` attribute in the `rwbuilder.conf` configuration file.

**Example:**

```
<engine id="rwEng" class="oracle.reports.engine.EngineImpl"
initEngine="1" maxEngine="1" minEngine="0" engLife="50" maxIdle="30"
callbackTimeOut="60000" classPath="D:\sybase_
installed\YUtil.jar;D:\sybase_installed\Ysybase.jar;D:\sybase_
installed\Ybase.jar;">
...
</engine>
```

- c. **Reports Server:** Append the driver location to the `classPath` attribute of the engine in the Reports Server configuration file.

**Example:**

```
<engine id="rwEng" class="oracle.reports.engine.EngineImpl"
initEngine="1" maxEngine="1" minEngine="0" engLife="50" maxIdle="30"
callbackTimeOut="60000" classPath="D:\sybase_
installed\YUtil.jar;D:\sybase_installed\Ysybase.jar;D:\sybase_
```



```
</engine>
```

- c. **Reports Server:** Append the driver location to the `classPath` attribute of the engine in the Reports Server configuration file.

**Example:**

```
<engine id="rwEng" class="oracle.reports.engine.EngineImpl"
initEngine="1" maxEngine="1" minEngine="0" engLife="50"
maxIdle="30" callbackTimeout="60000" classPath="D:\db2_
installed\YMutil.jar;D:\db2_installed\YMdb2.jar;D:\db2_
installed\YMbase.jar">
...
</engine>
```

- d. **jdbcpds.conf:** Located in the `ORACLE_HOME\reports\conf` directory. Refer to [Table 1-1](#) for more information on the parameters.

**Example:**

```
<driver name = "db2-merant"
subProtocol = "merant:db2"
connectString = "mainProtocol:subProtocol://databaseName"
class = "com.oracle.ias.jdbc.db2.DB2Driver"
connection = "oracle.reports.plugin.datasource.jdbcpds.
JDBCConnectionHandling"
loginTimeout = "0">
</driver>
```

## SQL Server Driver

1. Install the relevant `.jar` files in your Oracle9i Application Server and Oracle9i Developer Suite directory.

Jar files required: `YMutil.jar`, `YMsqserver.jar`, and `YMbase.jar`

2. Include an entry in the `REPORTS_CLASSPATH` to make the files available to Reports Builder and Oracle9iAS Reports Services.
  - a. **Reports Builder:** Prefix the driver location to the existing entries in `REPORTS_CLASSPATH`. This variable is located in the registry for Windows users and in the `reports.sh` file for Solaris users.

**Example:**

```
D:\sqlserver_installed\YMutil.jar;D:\sqlserver_
installed\YMsqserver.jar;D:\sqlserver_installed\YMbase.jar;existing
classpath entries
```

- b. **rwbuilder.conf**: Append the driver location to the engine `classPath` attribute in the `rwbuilder.conf` configuration file.

**Example:**

```
<engine id="rwEng" class="oracle.reports.engine.EngineImpl"
initEngine="1" maxEngine="1" minEngine="0" engLife="50"
maxIdle="30" callbackTimeOut="60000" classPath="D:\sqlserver_
installed\YUtil.jar;D:\sqlserver_
installed\YMsqserver.jar;D:\sqlserver_installed\YMbase.jar;">
...
</engine>
```

- c. **Reports Server**: Append the driver location to the `classPath` attribute of the engine in the Reports Server configuration file.

**Example:**

```
<engine id="rwEng" class="oracle.reports.engine.EngineImpl"
initEngine="1" maxEngine="1" minEngine="0" engLife="50"
maxIdle="30" callbackTimeOut="60000" classPath="D:\sqlserver_
installed\YUtil.jar;D:\sqlserver_
installed\YMsqserver.jar;D:\sqlserver_installed\YMbase.jar;">
...
</engine>
```

3. **jdbcpds.conf**: Located in the `ORACLE_HOME\reports\conf` directory. Refer to [Table 1-1](#) for more information on the parameters.

**Example:**

```
<driver name = "sqlserver-merant"
subProtocol = "merant:sqlserver"
connectString = "mainProtocol:subProtocol://databaseName"
class = "com.oracle.ias.jdbc.sqlserver.SQLServerDriver"
connection = "oracle.reports.plugin.datasource.jdbcpds.
JDBCConnectionHandling">
</driver>
```

## Informix Driver

1. Install the relevant `.jar` files in your Oracle9i Application Server and Oracle9i Developer Suite directory.

Jar files required: `YUtil.jar`, `YMinformix.jar`, and `YMbase.jar`

2. Include an entry in the `REPORTS_CLASSPATH` to make the files available to Reports Builder and Oracle9iAS Reports Services.

- a. **Reports Builder:** Prefix the driver location to the existing entries in `REPORTS_CLASSPATH`. This variable is located in the registry for Windows users and in the `reports.sh` file for Solaris users.

**Example:**

```
D:\informix_installed\YMutil.jar;D:\informix_
installed\YMinformix.jar;D:\informix_installed\YMbase.jar;existing
classpath entries
```

- b. **rwbuilder.conf:** Append the driver location to the engine `classPath` attribute in the `rwbuilder.conf` configuration file.

**Example:**

```
<engine id="rwEng" class="oracle.reports.engine.EngineImpl"
initEngine="1" maxEngine="1" minEngine="0" engLife="50"
maxIdle="30" callbackTimeOut="60000" classPath="D:\informix_
installed\YMutil.jar;D:\informix_installed\YMinformix.jar;D:\informix_
installed\YMbase.jar">
...
</engine>
```

- c. **Reports Server:** Append the driver location to the `classPath` attribute of the engine in the Reports Server configuration file.

**Example:**

```
<engine id="rwEng" class="oracle.reports.engine.EngineImpl"
initEngine="1" maxEngine="1" minEngine="0" engLife="50"
maxIdle="30" callbackTimeOut="60000" classPath="D:\informix_
installed\YMutil.jar;D:\informix_installed\YMinformix.jar;D:\informix_
installed\YMbase.jar">
...
</engine>
```

- d. **jdbcpds.conf:** Located in the `ORACLE_HOME\reports\conf` directory. Refer to [Table 1-1](#) for more information on the parameters.

**Example:**

```
<driver name = "informix-merant"
subProtocol = "merant:informix"
connectString = "mainProtocol:subProtocol://databaseName"
class = "com.oracle.ias.jdbc.informix.InformixDriver"
connection = "oracle.reports.plugin.datasource.jdbcpds.
JDBCConnectionHandling">
</driver>
```

## Custom Driver

Any driver that is not provided by Oracle must be installed and configured:

1. Install the relevant .jar files in your Oracle9i Application Server and Oracle9i Developer Suite directory.
2. Include an entry in the `REPORTS_CLASSPATH` to make the files available to Reports Builder and Oracle9iAS Reports Services.

Jar files required: Refer to the relevant driver documentation.

- a. **Reports Builder:** Prefix the driver location to the existing entries in `REPORTS_CLASSPATH`. This variable is located in the registry for Windows users and in the `reports.sh` file for Solaris users.

**Example:**

```
driver location\1st jar file;driver location\2nd jar file2;existing
classpath entries
```

- b. **rwbuilder.conf:** Append the driver location to the engine `classPath` attribute in the `rwbuilder.conf` configuration file.

**Example:**

```
<engine id="rwEng" class="oracle.reports.engine.EngineImpl"
initEngine="1" maxEngine="1" minEngine="0" engLife="50"
maxIdle="30" callbackTimeOut="60000" classPath="driver location\1st jar
file;driver location\2nd jar file;">
...
</engine>
```

- c. **Reports Server:** Append the driver location to the `classPath` attribute of the engine in the Reports Server configuration file.

**Example:**

```
<engine id="rwEng" class="oracle.reports.engine.EngineImpl"
initEngine="1" maxEngine="1" minEngine="0" engLife="50"
maxIdle="30" callbackTimeOut="60000" classPath="driver location\1st jar
file;driver location\2nd jar file;">
...
</engine>
```

- d. **jdbcpds.conf:** Located in the `ORACLE_HOME\reports\conf` directory. Add relevant driver configuration information to the `jdbcpds.conf` file. Refer to [Table 1-1](#) for more information on the required parameters.

**Example:**

```
<driver name = "<driver name>"
```

```
subProtocol = "<subProtocol>"
connectString = "mainProtocol:subProtocol://databaseName"
class = "<driver class name>"
connection = "<connection handling class>"
</driver>
```

---

---

**Note:** This value can still be `connection = "oracle.reports.plugin.datasource.jdbcpds.JDBCCConnectionHandling"` for your custom drivers, if you do not want to implement a custom connection dialog

---

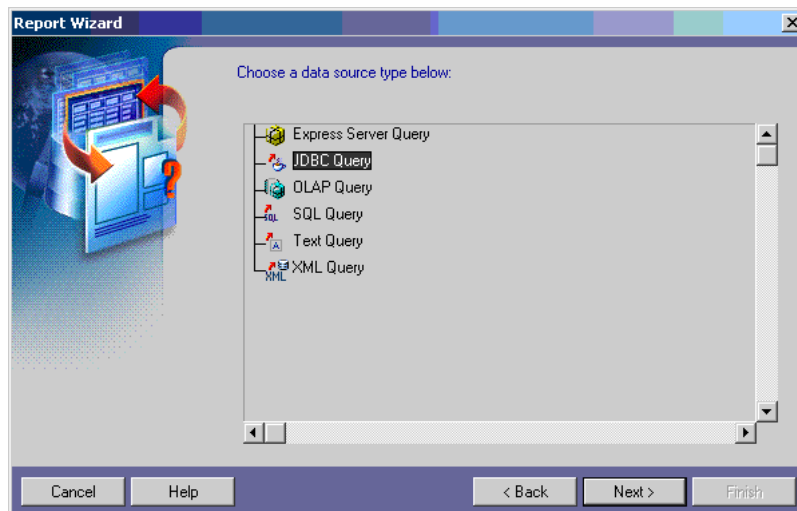
---

## Defining and Running a JDBC Query

After configuring the relevant JDBC drivers, you can define and run a JDBC query using either SQL or a stored procedure.

To define a JDBC query:

1. Start Reports Builder.
2. Invoke the Reports Wizard.
3. Select the data source type as **JDBC Query** and click **Next**. For more information on how to work with the Report Wizard, refer to the *Reports Builder online help*.

**Figure 1–2 Select a Data Source Type**

4. In the Data Source Definition window, click **Query Definition**.
5. Define one of the following:

- A SQL query:

```
SELECT * FROM DEPARTMENT;
```

- A stored procedure:

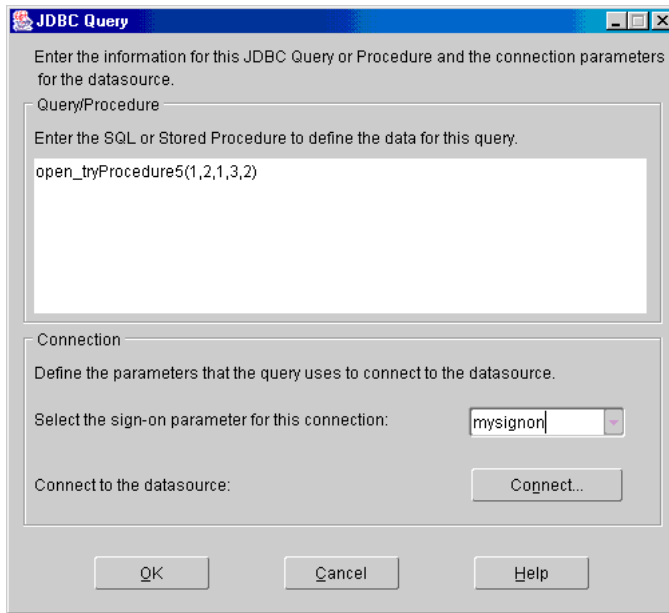
Enter the complete call syntax of your database's stored procedure. For example:

```
TestProc (40)
```

For more information on the call syntax, refer to your database documentation.

JDBC PDS submits the calling statement to the driver as specified, to invoke the stored procedure.

**Figure 1–3 Calling a stored procedure**



### Usage Notes

- To specify an Excel data source:
  - Query (Single worksheet):
 

```
SELECT * FROM [SHEET1$] or SELECT COL1, COL2, ...COLn
FROM [SHEET1$]
```

    - \* Where SHEET1\$ is the name of a .xls file
    - \* Where the first worksheet row value is taken as a column name for the query
  - Query (Multiple worksheets):
 

```
SELECT * FROM [WORKSHEETNAME$]
```

    - \* Where [WORKSHEETNAME\$] is the name of the worksheet
    - \* Where the first worksheet row is taken as a column name for the query

---

---

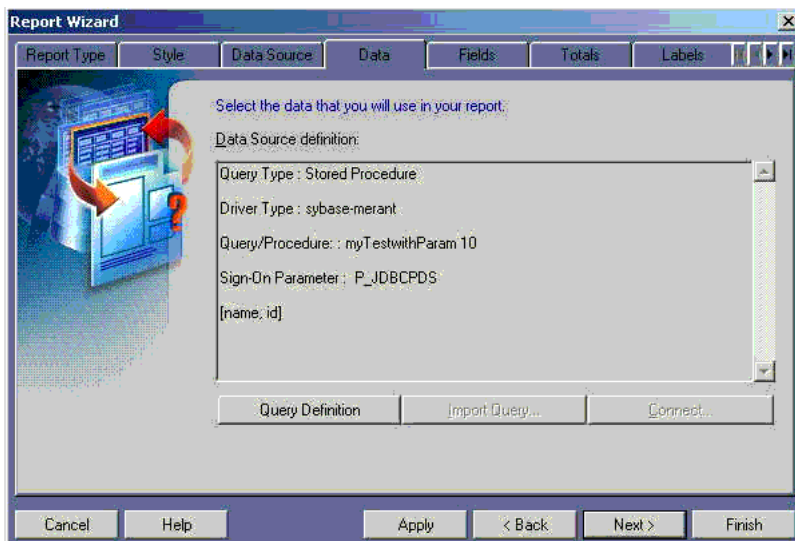
**Note:** If a value is not mentioned in any of the columns in the first row, then the default name is *FcolumnNumber*. For example, the 8th column will be F8, the ninth column will be F9, and so on.

---

---

6. Specify a sign-on parameter name. This sign-on parameter is associated with the connection information when run against a database. The default sign-on parameter value is `p_jdbcpds`:
  - a. Enter a new sign-on name and click **Connect**. Use this sign-on parameter to specify a database connection when you are running your report using Oracle*i*AS Reports Services.
  - b. Enter the connection information (user name, password, and database name) for the driver type. Refer to [Table 1-2](#), [Table 1-3](#), [Table 1-4](#), [Table 1-5](#), and [Table 1-6](#) for sample connection information.
  - c. Select the driver type. The driver list is displayed based on the values entered in the `jdbcpds.conf` file.
  - d. Click **Connect** to gain access to the database using the new sign-on. The connect string formed internally is a combination of:
    - \* The `connectString` driver attribute ([Table 1-1](#)) defined in the `jdbcpds.conf` file
    - \* The connection information supplied in the Connect dialog.
7. Click **OK** to execute the JDBC query.
8. The Reports Wizard displays the query description ([Figure 1-4](#)).

**Figure 1–4 Query Description**



9. Follow the steps in the wizard to define the layout and to run the report based on your JDBC query.

### Sample Connection Information

Table 1–2, Table 1–3, Table 1–4, Table 1–5, and Table 1–6 lists sample connection information for use with the pre-installed drivers.

**Table 1–2 Oracle Thin Driver**

Property	Value
Username	Reports
Password	Welcome
Database	hostname : The TCP/IP address or TCP/IP host name of the server you are connecting to. port : The TCP/IP port number. property : The connection properties. Refer to the driver documentation for a list of connection properties and their valid values. Example: server1.us.oracle.com:1300:session1

**Table 1–3 Oracle Thick Driver**

Property	Value
Username	Reports
Password	Welcome
Database	n123 where n123 is a tnsname entry in the tnsnames.ora file

**Table 1–4 JDBC-ODBC Driver**

Property	Value
Username	N/A
Password	This password is set at the time of establishing an ODBC connection.
Database	SQLSVR where SQLSVR is the ODBC Data entry in the ODBC data source

**Table 1–5 Sybase**

Property	Value
Username	Reports
Password	Welcome
Database	hostname: The TCP/IP address or TCP/IP host name of the server you are connecting to. port: The number of the TCP/IP port. Example: server1.us.oracle.com:1300

**Table 1–6 DB2**

Property	Value
Username	Reports
Password	Welcome

**Table 1-6 DB2**

Property	Value
Database	<p>hostname: The TCP/IP address or TCP/IP host name of the server you are connecting to.</p> <p>port: The TCP/IP port number.</p> <p>property: The connection properties. Refer to the driver documentation for a list of connection properties and their valid values.</p> <p>Example1: server1:1654</p> <p>Example2: server2:1721;PackageName=pkg1</p>

## Running a JDBC Report using Oracle9iAS Reports Services

When you run a report having a JDBC query (Reports Server or `rwr` engine), use the sign-on parameter to submit the connection information for the JDBC data source. This sign-on parameter is defined for your JDBC query in the design time.

For example, if your report has a JDBC query to a Sybase data source, a JDBC query to a DB2 data source, and a SQL query to an Oracle data source, then the request could be defined as:

```
http://your_ias
server:port//reports/rwservlet?report=my.rdf&userid=user/pwd@oracledb
&desformat-pdf&destype=cache&p_sybasepds=sybaseuser/pw@sybasehost:port
&p_db2pds=db2user/pwd@db2host:port
```

where:

- `p_sybasepds` is the sign-on parameter associated with the sybase JDBC query.
- `p_db2pds` is the sign-on parameter associated with the DB2 JDBC query defined in the report at design time.
- `userid` is the value for connecting the SQL query to the Oracle database. You do not need to specify the `userid` if your report does not have a SQL query or a REF cursor query.

The default sign-on parameter name `p_jdbcpds` will be used if you have not specified a name in the JDBC query dialog while designing the report.

## Troubleshooting Information

This section lists out:

- JDBC PDS error messages ([Error Messages](#))
- JDBC query troubleshooting ([Trace Information](#)).

## Error Messages

[Table 1–7](#), [Table 1–8](#), and [Table 1–9](#) lists troubleshooting information related to the JDBC PDS.

**Table 1–7 Error Messages related to the database connection**

Error Message	Cause	Action
Connection class {0} can't be loaded	Invalid connection class mentioned in the <code>jdbcpds.conf</code> file for the selected driver.	Ensure that the driver connection class specified in the <code>jdbcpds.conf</code> file is both valid and available.
Failed to connect to the datasource	Invalid connection information.	Ensure the validity of the username, password, database, and driver type.
Invalid sign-on parameter {0}	Invalid sign-on parameter for the specified query or procedure.	Ensure the sign-on parameter is available and valid for the report's JDBC query type.
Invalid value is given to the sign-on parameter {0}.	Invalid connect string for the specified sign-on parameter.	Ensure that the specified connect string for this sign-on parameter is valid for the selected driver.

**Table 1–8 Error messages related to executing the data source**

Error Message	Cause	Action
Reference parameter of type Date is not supported by JDBC driver used.	The driver used to connect to database does not support the <b>Date</b> data type as a reference parameter.	Use either: <ul style="list-style-type: none"> <li>■ The <b>String</b> data type as the reference parameter</li> <li>■ A different JDBC driver that supports the <b>Date</b> data type as a reference parameter.</li> </ul>
Invalid lexical parameter {0} is used in the query	Invalid lexical parameter used in the query or procedure.	Ensure that the query or procedure uses valid lexical parameters. Create a new parameter if it is not available.

**Table 1–8 Error messages related to executing the data source**

<b>Error Message</b>	<b>Cause</b>	<b>Action</b>
SQL Error:	SQL syntax error in the specified query or procedure.	Ensure that the syntax of the query or procedure is valid. Refer to the relevant data source's documentation.
Invalid query/procedure for the specified datasource.	Invalid query or procedure syntax.	Ensure that the syntax of the query or procedure is valid. Refer to the relevant data source's documentation
Invalid reference parameter value	Invalid reference parameter value.	Verify that the reference column types and values are correct.
No query/procedure is entered.	The query or procedure text field is empty.	Enter a valid query or procedure in the text field.
Database URL:	Invalid database URL.	Verify the validity of the specified database name and the selected driver type.
Either the number of columns or the types of columns does not match the query definition	The data fetched does not match the number of columns or column types specified in the query definition.	Ensure that the number of columns and the column types match the query definition.
The column type {0} used in the query/procedure is not supported by Reports JDBC query.	This column type is not supported by the Oracle9i Reports JDBC query interface.	Ensure that only column types supported by the Oracle9i Reports JDBC query interface are used. Refer to the JDBC specification and Oracle9i Reports documentation for a list of all supported types.

**Table 1–9 Isolating driver / pds issues**

<b>Error Message</b>	<b>Cause</b>	<b>Action</b>
The inline DTD section of the configuration file jdbcpds.conf has been modified.	The format of the inline DTD section in the jdbcpds.conf file has been altered.	If the DTD format is modified, ensure the validity of configuration file against the JDBC PDS requirement.

**Table 1–9 Isolating driver / pds issues**

<b>Error Message</b>	<b>Cause</b>	<b>Action</b>
Line Number:	An error was found on the specified line of the <code>jdbcpds.conf</code> file.	Correct the error on the specified line.
Configuration file <code>jdbcpds.conf</code> is not found	The <code>jdbcpds.conf</code> file is not found under the <code>reports/conf</code> directory.	Ensure that the <code>jdbcpds.conf</code> file is available in the <code>reports/conf</code> directory.
Parsing error in the configuration file <code>jdbcpds.conf</code> . Number of errors: {0}	The XML section in the <code>jdbcpds.conf</code> file does not conform with its inline DTD.	Ensure that the XML section in the <code>jdbcpds.conf</code> file refers to the correct inline DTD.
No entry is present for the driver {0} in the <code>jdbcpds.conf</code> file.	The driver used in the query is not mentioned in the <code>jdbcpds.conf</code> file.	Ensure that the entry for the required driver along with the related driver information is in the <code>jdbcpds.conf</code> file.

## Trace Information

Use the detailed trace information (`ORACLE_HOME\reports\logs\`) generated by Oracle9i Reports to debug your JDBC query.

- Design time (building a JDBC query) and run time (running a JDBC query)

The trace information generated is helpful to find out the following:

- Lexical and bind parameters.
- Final connect string formed to connect to the driver.
- Metadata information received from the driver.
- Final query submitted to the database.

See [Example 1–1](#) for a sample design-time trace output.

See [Example 1–2](#) for a sample run-time trace output.

### Sample trace output

#### **Example 1–1 Building a JDBC Query from JDBC Query Dialog**

Connection handling trace showing final connect string

```
[2003/4/7 5:41:38:686] Debug 50103 (jdbcpds): handleConnectButtonEvent : start
[2003/4/7 5:41:38:686] Debug 50103 (jdbcpds): handleConnectButtonEvent :
subProtocol :sybase-merant
[2003/4/7 5:41:38:686] Debug 50103 (jdbcpds): handleConnectButtonEvent :
connection class
:oracle.reports.plugin.datasource.jdbcpds.JDBCConnectionHandling
[2003/4/7 5:41:38:696] Debug 50103 (jdbcpds): handleConnectButtonEvent : combine
string :jdbc:merant:sybase://server1.us.oracle.com:1300
[2003/4/7 5:41:38:696] Debug 50103 (jdbcpds): JDBCDataSource : setJDBCQueryType:
sybase
[2003/4/7 5:41:41:350] Debug 50103 (jdbcpds): JDBCUIEventHandler :
handleConnectEvent : Valid Connection
com.oracle.ias.jdbc.sybase.SybaseConnection@56fc16
[2003/4/7 5:41:41:350] Debug 50103 (jdbcpds): JDBCUIEventHandler :
handleConnectEvent : END com.oracle.ias.jdbc.sybase.SybaseConnection@56fc16
```

**Design time metadata of query**

```
[2003/3/31 6:35:46:363] Debug 50103 (jdbcpds): JDBCUIEventHandler :
handleOKEvent : Serialize XML<jdbc pds DTDVersion="
1.0"><JDBCQuery>jdbc pds pkg.proc_with_
param(1,2,3,4,5)</JDBCQuery><QueryDefinition>1</QueryDefinition><driverType>orac
le</driverType><connectionClass>oracle.reports.plugin.datasource.jdbcpds.JDBCCon
nectionHandling</connectionClass><SignOnParameter>P_
JBCPDSPDS</SignOnParameter><jdbcElements><elementname = "EMPNO" type = "2"
typeName = "NUMBER" columnSize = "4" columnScale = "0" /><element name =
"ENAME" type = "12" typeName = "VARCHAR2" columnSize = "10" columnScale =
"0" /><element name = "JOB" type = "12" typeName = "VARCHAR2" columnSize =
"9" columnScale = "0" /><element name = "MGR" type = "2" typeName = "NUMBER"
columnSize = "4" columnScale = "0" /><element name = "HIREDATE" type = "93"
typeName = "DATE" columnSize = "16" columnScale = "0" /><element name = "SAL"
type = "2" typeName = "NUMBER" columnSize = "7" columnScale= "2" /><element
name = "COMM" type = "2" typeName = "NUMBER" columnSize = "7" columnScale =
"2" /><element name = "DEPTNO" type = "2" typeName = "NUMBER" columnSize = "2"
columnScale = "0"
/></jdbcElements><referenceColumns></referenceColumns></jdbc pds>
[2003/3/31 6:35:46:383] Debug 50103 (jdbcpds): JDBCUIEventHandler :handleOKEvent
END
```

**Example 1–2 Running a JDBC Query:**

```
[2003/3/18 5:45:17:707] Debug 50103 (jdbcpds): JDBCDataSource : startRuntime
method : START
```

**Describing the JDBC Query:**

```
[2003/3/18 5:45:17:707] Debug 50103 (jdbcpds): JDBCDataSource : describe :
```

START

```
[2003/3/18 5:45:17:707] Debug 50103 (jdbcpds): applyXML: Extract the Serilzed
XML containing Query Meta Data <jdbcps DTDVersion=" 1.0"><JDBCQuery>select *
from
emp</JDBCQuery><QueryDefinition>0</QueryDefinition><driverType>oracle</driverTyp
e><connectionClass>oracle.reports.plugin.datasources.jdbcps.JDBCConnectionHandli
ng</connectionClass>...
```

**ConnectionHandling At Runtime:**

```
[2003/3/18 5:45:17:737] Debug 50103 (jdbcpds): JDBCDataSource : startRuntime :
Create a new connection and handle it
[2003/3/18 5:45:17:737] Debug 50103 (jdbcpds): JDBCExecuteQuerySource :
handleConnection : START
[2003/3/18 5:45:17:778] Debug 50103 (jdbcpds): JDBCExecuteQuerySource :
handleConnection : set driver
[2003/3/18 5:45:17:778] Debug 50103 (jdbcpds): JDBCExecuteQuerySource :
handleConnection : Check if Connection for the sign on parameter is pooled
[2003/3/18 5:45:17:778] Debug 50103 (jdbcpds): JDBCExecuteQuerySource
:handleConnection : connection available in pool
[2003/3/18 5:45:17:778] Debug 50103 (jdbcpds): handleConnection : END
[2003/3/18 5:45:17:778] Debug 50103 (jdbcpds): JDBCDataSource : startRuntime :
END
```

**Runtime execution of jdbc query**

```
[2003/3/31 6:36:2:836] Debug 50103 (jdbcpds): JDBCDataSource : execute : run
Query
[2003/3/31 6:36:2:836] Debug 50103 (jdbcpds): JDBCExecuteQuerySource :
getOutputFromDatabase : START
[2003/3/31 6:36:2:836] Debug 50103 (jdbcpds): JDBCExecuteQuerySource :
getOutputFromDatabase: start Query stringto be submitted
jdbcpspkg.proc_with_param(1,2,3,4,5)
[2003/3/31 6:36:2:836] Debug 50103 (jdbcpds): JDBCExecuteQuerySource :
getOutputFromDatabase : check connection
[2003/3/31 6:36:2:836] Debug 50103 (jdbcpds): JDBCExecuteQuerySource :
getOutputFromDatabase : QSource Id: 1
[2003/3/31 6:36:2:836] Debug 50103 (jdbcpds): JDBCExecuteQuerySource:
executeOracleProcedure:Start
[2003/3/31 6:36:2:836] Debug 50103 (jdbcpds): JDBCExecuteQuerySource:
executeOracleProcedure:Procedure to be submitted { call
jdbcpspkg.proc_with_param(?,?,?,? ) }
[2003/3/31 6:36:2:836] Debug 50103 (jdbcpds): JDBCExecuteQuerySource:
executeOracleProcedure: Set parameters for the procedure call
[2003/3/31 6:36:2:836] Debug 50103 (jdbcpds): JDBCExecuteQuerySource:
executeOracleProcedure: execute procedure
[2003/3/31 6:36:2:847] Debug 50103 (jdbcpds): JDBCDataSource : execute : query
```

```
execution over andresultset object is
oracle.jdbc.driver.OracleResultSetImpl@751a9e
[2003/3/31 6:36:2:847] Debug 50103 (jdbcpds): JDBCDataSource : execute : END
```

**Running Report trace with Result set info**

```
2003/4/7 5:26:6:996] Debug 50103 (jdbcpds): JDBCDataSource : execute : replace
lexical columns with actual string for the query
[2003/4/7 5:26:6:996] Debug 50103 (jdbcpds): JDBCDataSource : execute : run
Query
[2003/4/7 5:26:6:996] Debug 50103 (jdbcpds): JDBCExecuteQuerySource :
getOutputFromDatabase : START
[2003/4/7 5:26:6:996] Debug 50103 (jdbcpds): JDBCExecuteQuerySource :
getOutputFromDatabase: start Query string to be submitted select * from reports
[2003/4/7 5:26:7:6] Debug 50103 (jdbcpds): JDBCExecuteQuerySource :
getOutputFromDatabase : check connection
[2003/4/7 5:26:7:6] Debug 50103 (jdbcpds): JDBCExecuteQuerySource :
getOutputFromDatabase : QSource Id: 4
[2003/4/7 5:26:7:6] Debug 50103 (jdbcpds): JDBCExecuteQuerySource :
getOutputFromDatabase : Query source is SQL query
[2003/4/7 5:26:7:6] Debug 50103 (jdbcpds): JDBCExecuteQuerySource:executeQuery
Start
[2003/4/7 5:26:7:6] Debug 50103 (jdbcpds): executeQuery prepareStatement select
* from reports
[2003/4/7 5:26:7:6] Debug 50103 (jdbcpds): executeQuery : bind parameters set
for the query
[2003/4/7 5:26:7:6] Debug 50103 (jdbcpds): executeQuery : JDBC Query executed
[2003/4/7 5:26:7:387] Debug 50103 (jdbcpds): JDBCExecuteQuerySource :
getOutputFromDatabase : Query result col 0 test col 1 10
[2003/4/7 5:26:7:387] Debug 50103 (jdbcpds): JDBCExecuteQuerySource:executeQuery
Start
[2003/4/7 5:26:7:387] Debug 50103 (jdbcpds): executeQuery prepareStatement
select * from reports
[2003/4/7 5:26:7:387] Debug 50103 (jdbcpds): executeQuery : bind parameters set
for the query
[2003/4/7 5:26:7:387] Debug 50103 (jdbcpds): executeQuery : JDBC Query executed
[2003/4/7 5:26:7:767] Debug 50103 (jdbcpds): JDBCDataSource : execute : query
execution over andresultset object is
com.oracle.ias.jdbc.base.BaseResultSet@56c3cf
[2003/4/7 5:26:7:767] Debug 50103 (jdbcpds): JDBCDataSource : execute : END
```

## Adding your own PDS

Oracle9i Reports has exposed the PDS API and also contains an RSDK tutorial (<http://otn.oracle.com/products/reports/apis/index.html>) that describes in detail how to implement or customize your own PDS. Using this API, you can implement an unlimited number of PDSs to access any kind of data sources that you have.

The main tasks you must perform to add your PDS are:

- [Registering the PDS](#)
- [Configuring the jdbcpds.conf file](#)
- [Installing your PDS' .jar files](#)
- [Installing the driver's .jar files](#)

## Registering the PDS

Register your PDS with the preferences file to make it available to Reports Builder and Oracle9iAS Reports Services.

**Table 1–10 Preferences file**

File	Location	Operating System
cauprefs.ora	ORACLE_HOME	Windows
prefs.ora	ORACLE_HOME	Unix

The preferences file should reference the factory class implementation of either Oracle9i Reports or your custom class file. Reports Builder displays the relevant PDS icon only if the factory class is registered:

```
Reports.PluggableDataSourceFactories =
("oracle.reports.plugin.datasources.xmlpds.XMLDataSourceFactory",
"oracle.reports.plugin.datasources.jdbcpds.JDBCDataSourceFactory",
"oracle.reports.plugin.datasources.textpds.TextDataSourceFactory",
"oracle.reports.plugin.datasources.mypds.mypdsDataSourceFactory",
"oracle.dss.pds.bibbeans.xrpds.XRPDSFactory",
"oracle.dss.pds.snapi.expresspds.ExpressPDSFactory")
```

To add your PDS:

1. Open the preferences file ([Table 1–10](#)) using any text editor.

2. Locate `Reports.PluggableDataSourceFactories`.
3. Include the name of any JDBC factory class using the following syntax:

```
Reports.PluggableDataSourceFactories = ("pluginClassname" [,  
"pluginClassname"]...)
```

`pluginClassname` is the name of the factory class containing the required implementation.

---

---

**Note:** Refer to the Oracle9i Reports SDK on OTN (<http://otn.oracle.com>), for more information on implementing your custom PDS class.

---

---

4. Restart Reports Builder / Oracle9iAS Reports Services for the changes to take effect.

## Configuring the `jdbcpds.conf` file

For more information on how to configure the `jdbcpds.conf` file, refer to the [JDBC Configuration File](#) section.

## Installing your PDS' .jar files

Specify the path to your PDS .jar files. This makes all the relevant classes available to Reports Builder and Reports Server.

### Reports Builder

Prefix the path of all dependent .jar files to the Oracle9i Reports environment variable `REPORTS_CLASSPATH`. This variable is located in the registry for Windows users and in the `reports.sh` file for Solaris users.

**Example:** `D:\mypds.jar;existing classpath entries`

### rwbuilder.conf

Add the jar file location to the engine `classPath` attribute in the `rwbuilder.conf` configuration file.

**Example:** `<engine id="rwEng" class="oracle.reports.engine.EngineImpl"  
initEngine="1" maxEngine="1" minEngine="0" engLife="50"  
maxIdle="30" callbackTimeOut="60000" classPath="d:\mypds.jar;">`

```
...  
</engine>
```

## Reports Server

Append the jar file location to the `classPath` attribute in the Reports Server configuration file.

**Example:** `<engine id="rwEng" class="oracle.reports.engine.EngineImpl"  
initEngine="1" maxEngine="1" minEngine="0" engLife="50"  
maxIdle="30" callbackTimeOut="60000" classPath="d:\mypds.jar;">  
...  
</engine>`

## Installing the driver's .jar files

Refer to the [Custom Driver](#) section for more information on how to install the driver's .jar files.

