Crystal Enterprise Report Application Server (CE RAS) 9

Database Connectivity using the Java SDK

Overview

This document discusses logging on to databases, setting table location, and using command tables and stored procedures in conjunction with the Crystal Enterprise Report Application Server (CE RAS) 9 Java SDK.

NOTE	Crystal Enterprise Report Application Server (CE RAS) 9 is shipped on a separate CD with Crystal Reports 9 Professional, Developer, and Advanced editions.

Contents

Introduction

Although there are many different database management systems (DBMS) available, Crystal Reports eliminates many of the differences when it connects to the actual database. The process of working with databases, tables, fields, and records is much the same regardless of the actual type of data being accessed.

Similarly, this document discusses some of the ways in which the Crystal Enterprise Report Application Server (CE RAS) Java SDK is used to manage different data sources, including database tables, stored procedures, active data, and command tables.

Database Logon

Many database management systems (DBMS) require you to provide logon credentials when connecting, such as server name, database name, user name, and password. However, the exact types of information differ depending on the connection method. For example, ODBC connections require the data source name (DSN), whereas OLEDB connections require a server name.

Logging on by table and by connection

The Crystal Enterprise Report Application Server (CE RAS) Java SDK offers two ways to ensure that a report has the logon credentials that it requires to run:

- Logging on by table
- Logging on by connection

Logging on by table

When logging on by table, use the CE RAS Java SDK to provide logon criteria on a table-by-table basis as follows:

```
ITable tbl = rptDoc.getDatabase().getTables().getTable(0);
// Make copy of the table’s ConnectionInfo object
IConnectionInfo ci = tbl.getConnectionInfo();

// Set the user name and password information
ci.setUserName("myUserName");
ci.setPassword("myPassword");

// Modify the ConnectionInfo object for the table using the
table alias
rptDoc.getDatabaseController().modifyTableConnectionInfo(tb
l.getAlias(), ci);
```

Logging on by connection

When logging on by connection, use the CE RAS Java SDK to retrieve a collection of connections, add logon information to a connection, and then modify the connection collection. The following sample code demonstrates how to log on by connection:

```
PropertyBag pb = new PropertyBag();
pb.putBooleanValue(PropertyBagHelper.PROMPTPROPERTY_ALL,
true);
pb.putBooleanValue(PropertyBagHelper.PROMPTPROPERTY_INCLUDE
ONDEMANDSUBREPORT, false);

// Retrieve the collection of ConnectionInfos from the
// report using the DatabaseController
ConnectionInfos cInfos =
rptDoc.DatabaseController.getConnectionInfos(pb);

// The GetConnectionInfos method takes a PropertyBag object
// with a value that specifies what ConnectionInfo objects
// to retrieve
IConnectionInfo ci = cInfos.getConnectionInfo(0);
ci.setUserName("myUserName");
ci.setPassword("myPassword");

// Modify the ConnectionInfos using the SetConnectionInfos
// method of the DatabaseController
rptDoc.getDatabaseController().setConnectionInfos(cInfos);
```

Logging on using the same user name and password

If all the tables in your report are from the same database, or use the same user name and password, then use the **logon** method of the **DatabaseController** object to connect to the tables. The **logon** method iterates through each table in the report and updates the user name and password. The following sample code demonstrates how to use this method:

```
rptDoc.getDatabaseController().logon("myUserName",
"myPassword");
```

Logging on using the Viewer SDK

Crystal Enterprise Report Application Server (CE RAS) 9 contains two SDKs: the RAS SDK and Viewer SDK. The RAS SDK directs the CE RAS Server to make changes to actual report documents. Most of the methods and properties in the RAS SDK are part of the Report Creation API (RCAPI), and thus require an RCAPI license that is only available with the Advanced Edition of CE RAS. The Viewer SDK is available in the Advanced and Developer editions.

The Viewer SDK, however, does not enable you to change actual report documents. The viewer objects render a report into an HTML representation of the original report. They have the ability, as a presentation layer, to provide logon credentials, set selection formula criteria, and set parameter values. Essentially, the viewer objects are able to provide the information required to display a report, but do not have the ability to modify the original report (the report is opened in a read-only format).

If any of the Viewer SDK classes based on the **ReportServerControl** class are used to view a report, and the **ReportClientDocument** object does not have sufficient logon credentials, the viewer will prompt you for the logon credentials. When this happens, one of two things occur:

- The viewer prompts for the user name and password.
- The viewer uses the logon credentials passed to it in the application code.

NOTE	The ActiveX Viewer and Java Viewer do not have the ability to prompt for user name and password, as well as to have the information passed to them at runtime. Therefore it is not recommended that you use these viewers because you would not be able to access the database to retrieve current information.

The viewer's logon behavior uses the logon by connection model. That is, the **ReportClientDocument** object retrieves information for each of the connections, rather than for each of the tables. The viewer accepts a collection of **ConnectionInfo** objects rather than table-specific logon credentials.

You have the option of writing client-side code to bypass the logon prompt, as the following sample code demonstrates:

```
rptDoc.open("c:\\Reports\\Report1.rpt",
OpenReportOptions._openAsReadOnly);

ReportServerControl ServerControl = new
ReportServerControl();


// The ServerControl object collects information about the
// required inputs to run a report such as parameter fields
// and database logon credentials. The viewer prompts for
// these values if they are not set programmatically
// through either the RAS or Viewer SDK
```

```
ServerControl.setReportSource(rptDoc.getReportSource());

ServerControl.setEnableLogonPrompt(false);

ServerControl.processHttpRequest(request, response,
getServletConfig().getServletContext(), out);


// Disable the default logon prompt and iterate through the

// ServerControl object's ReportLogon collection, setting

// the user name and password


ConnectionInfos reportLogons =
ServerControl.getDatabaseLogonInfos();

for(int i = 0; i < reportLogons.size(); i++)

{

    reportLogons.getConnectionInfo(i).setUserName("myUserNam
    e");

    reportLogons.getConnectionInfo(i).setPassword("myPasswor
    d");

}


// Send the report over to the viewer to preview it

CrystalReportViewer crViewer = new CrystalReportViewer();

crViewer.setReportSource(rptDoc.getReportSource());

crViewer.setDatabaseLogonInfos(reportLogons);


crViewer.processHttpRequest(request, response,
getServletConfig().getServletContext(), out);
```

Active Data

If you wish to pass a Java ResultSet to a report, use the **setDataSource** method of the **DatabaseController** object, as the following sample code demonstrates:

```
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");

java.sql.Connection con =
DriverManager.getConnection("jdbc:odbc:Xtreme Sample
Database 9");

Statement stmt = con.createStatement();


// Create a java ResultSet to contain the desired data

ResultSet rs = stmt.executeQuery("SELECT * FROM Orders");


// setDataSource takes the ResultSet, the alias of the
// existing table and a new table alias which is ALWAYS
"Reports"

// It is highly recommended that you destroy the ResultSet,

// Statement, and Connection objects after invoking the

// setDataSource method

rptDoc.getDatabaseController().setDataSource(rs,
"Customer", "Reports");
```

Limitations

Only one "table" of Active Data may be specified. That is, you cannot create a **ResultSet** object for the Orders table and another for the Orders Detail table, and then pass both to the report using the **setDataSource** method. To work around this limitation, link the two tables to create a single **ResultSet** object.

Making Changes to Data Connections

You may wish to make additional changes to the connection information. The **ConnectionInfo** object contains a set of connection attributes that are accessed using the **Attributes** property. This property returns a **PropertyBag** object.

Using PropertyBags

Since the types and values of the attributes contained in the **ConnectionInfo** object vary depending on the type of database and database connection, **PropertyBag** objects are used rather than a collection of fixed properties. A **PropertyBag** is a collection of key-value pairs representing the connection attributes, where the key is always a string value and the value can be a number, string, Boolean, or object (including another PropertyBag). These attributes are specific to the connection type. For example, Table 1 shows the attributes for a native Oracle connection. Note that the QE_LogonProperties is matched with a value that is a PropertyBag.

Key	**Value**
Database DLL	crdb_oracle.dll
QE_ServerDescription	ORCL_RCON1
QE_LogonProperties	PropertyBag
QE_SQLDB	True
QE_DatabaseType	Oracle Server

Table 1 – Attributes for Native Oracle Connection

For a native Oracle connection, the PropertyBag value for the QE_LogonProperties contains one key-value pair (see Table 2).

Key	Value
Server	ORCL_RCON1

Table 2 – PropertyBag value for Native Oracle Connection

Setting or changing QE_LogonProperties

Another PropertyBag key is the QE_LogonProperties. Recall that this key returns a value that is a PropertyBag object containing one or more objects. To change or set the QE_LogonProperties use the following sample code:

```
// Create the inner PropertyBag containing the key-value

// pair for the Server attribute


PropertyBag connAttributes = new PropertyBag();

PropertyBag qeLogonProps = new PropertyBag();

qeLogonProps.put(PropertyBagHelper.CONNINFO_SERVER_NAME,
"ORCL_RCON1");


// Use the Item property of the PropertyBag to add the

// inner property bag with the ConnInfo_CRQE_LogonProperties
key.


connAttributes.put(PropertyBagHelper.CONNINFO_CRQE_LOGONPRO
PERTIES, qeLogonProps);
```

Setting or changing the database DLL

The following sample code demonstrates how to set or change the database DLL for a report:

```
PropertyBag connAttributes = new PropertyBag();

connAttributes.put(PropertyBagHelper.CONNINFO_DATABASE_DLL,
"crdb_oracle.dll");
```

Changing the Table Location

You may also wish to change the table location of a report. If a report is designed against a particular table, but must be populated with data from another table, use the **setTableLocation** method to replace a table with a modified version of itself.

For example, there are two tables in an Oracle database called SALES.SALESEAST and SALES.SALESWEST. A report is designed using the SALESEAST table, but must be viewed with data from SALESWEST. The table schemas are identical in that the column names and types are identical. To change from the SALESEAST to SALESWEST table, use the following sample code:

```
Table origtbl =
(Table)rptDoc.getDatabase().getTables().getTable(0);

Table newtbl = (Table)origtbl.clone(true);


// Use the Name and QualifedName properties, especially

// when the qualified name for the table differs between

// the source and destination data sources


newtbl.setName("SALESEAST");

newtbl.setQualifiedName("SALES.SALESEAST");

rptDoc.getDatabaseController().setTableLocation(origtbl,
newtbl);
```

Changing databases

Rather than reporting off the same database, you can also report off a different database. For example, the following sample code demonstrates how to change to a different Oracle database server:

```
Table origtbl = (Table)
rptDoc.getDatabase().getTables.gettable(0);

Table newtbl = (Table) origtbl.clone(true);

IConnectionInfo connInfo = newtbl.getConnectionInfo();

PropertyBag connAttributes = connInfo.getAttributes();

PropertyBag connQEProps =
connAttributes.get(PropertyBagHelper.CONNINFO_CRQE_LOGONPRO
PERTIES);

connQEProps.put(PropertyBagHelper.CONNINFO_SERVER_NAME,
"NewServerName");

rptDoc.getDatabaseController().setTableLocation(origtbl,
newtbl);
```

NOTE	For a native Oracle connection, the **Server** attribute of the **QE_LogonProperties** attribute of the **ConnectionInfo** object controls to which server you are connecting. To change this attribute, complete the following steps: 1. Retrieve the **ConnectionInfo** attributes. 2. Retrieve the **QE_LogonProperties** PropertyBag. 3. Put the server name into the Server attribute. 4. Call the **SetTableLocation** method to modify the table.

Command objects

If the database you are using supports a query language such as SQL, you can write your own command using the **Add Command** node in the Crystal Reports **Database Expert**. This creates a virtual table that represents the results of processing the command. Command objects enable you to have complete control over the data processing that gets pushed down to the database server.

The CE RAS SDK treats a command object as a type of table. It is modeled using the **CommandTable** object. The **CommandTable** object inherits all the properties and methods of a regular **Table** object and adds a **CommandText** property that contains the SQL used in the expression. To display the **CommandText** property, use the following sample code:

```
CommandTable cmdTable = (CommandTable)
rptDoc.getDatabase().getTables().getTable(0);

System.out.println(cmdTable.getCommandText());
```

Stored Procedures

Procedures are stored in a database to help minimize network traffic. That is, a long SQL statement may be stored to avoid passing it back and forth over the network. The CE RAS SDK treats a stored procedure as a type of table. Stored procedures are modeled using the **Procedure** object. The **Procedure** object inherits all the properties and methods of a regular **Table** object, and adds a **Parameters** property that contains a collection of stored procedure parameters.

Parameterized stored procedures

It is also possible for the stored procedure to have parameters. In this case, only the argument values need to be transferred across the network. If a report connects to a database that has a stored procedure, and that stored procedure has parameters, the **ParameterFields** property is updated to include these parameters. You cannot change the definition of these parameters, but it is possible to modify their current values. Refer to the document Ceras9_java_parameters.pdf on our support site for more information.

Changing location of stored procedure

When changing location from one stored procedure to another, use the **Name** and **QualifiedName** properties, just as you would for a regular table. If the stored procedure is parameterized, it is recommended to use the **VerifyDatabase** method of the **ReportClientDocument** object. Otherwise, the report will fail to run if the names of the parameters of the source and destination stored procedures are different.

To change the location to a different stored procedure, use the following code:

```
Table origtbl = (Table)
rptDoc.getDatabase().getTables().getTable(0);

Table newtbl = (Table) origtbl.clone(true);

newtbl.setName("TEST_PROCEDURE");

newtbl.setQualifiedName("VANTECH.TEST_PROCEDURE");

newtbl.setUserName("myUserName");

newtbl.setPassword("myPassword");

rptDoc.getDatabaseController().setTableLocation(origtbl,
newtbl);

rptDoc.verifyDatabase();
```

NOTE	To highlight text in a PDF document for copying and pasting code, click the **Text Select Tool** toolbar button in Adobe Acrobat. This procedure applies to Adobe Acrobat 4.0 and 5.0.

DISCLAIMER	The third party products discussed in this white paper were not fully tested in conjunction with Crystal Enterprise prior to its release. Officially supported Crystal Enterprise platforms are listed in the text file Platforms.txt found in the root of the Crystal Enterprise CD as well as in the Knowledge Base article c2009003 (search for this article number at http://support.crystaldecisions.com/kbase), which is more current. The information in this document is provided as a courtesy to assist our customers with the configuration of our product in conjunction with these third party platforms. In the event issues arise with an unsupported configuration, there is no escalation support; however, they will be considered during the development of the next generation of our product.

Finding More Information

For sample Crystal Enterprise Report Application Server (CE RAS) 9 Java applications, search for Dev_ras_9_jsp_samples.exe and Adv_ras_9_jsp_samples.exe on our support site at:

http://support.crystaldecisions.com/search

For more information on the objects, methods, and properties discussed in this document, consult the CE RAS 9 Javadocs.

For more information and resources, refer to the product documentation and visit the support area of the web site at: www.businessobjects.com

This section lists titles and links to additional material. If possible, provide a short description with each resource. You will probably end up providing a lot of cross-references in this section.

www.businessobjects.com