opentext™

Relativity Java Client for RM/COBOL™

2023-11-10

Contents

Introduction to the Relativity Java Client

Relativity Java Client is the universal JDBC client-side driver component of Relativity Client/Server. It provides relational access of legacy data to a Java application.

The Relativity Java Client can be installed on a Windows or UNIX server that has a Java runtime environment and can be used from a Java application to access data on a Relativity Data Server. The Relativity Data Server component can be installed on a Windows or UNIX server and provides access to legacy data through its server data sources.

Relativity Java Client is a Type 4 JDBC driver, written completely in Java, capable of connecting to any Relativity Data Server data source from Java JDBC-enabled applications without using a JDBC-ODBC Bridge.

About JDBC

Java Database Connectivity (JDBC) is a set of standard interfaces that enable Java applications to access multiple database management systems using Structured Query Language (SQL). The JDBC Driver Manager handles multiple drivers connecting to different data sources.

Relativity Java Client implements the JDBC 4.3 standard and throws SQLFeatureNotSupportedException for any unimplemented methods from JDBC 4.3 Specifications. Also, it does not implement pooled PreparedStatements. If you find an unimplemented method from the standard that is important to your business needs, please contact *OpenText Support for Micro Focus Products*.

About the Documentation

This manual contains information specific to installing Relativity Java Client, deploying it with Java applications, and setting up data source connections. For information about installing and using the Relativity Data Server, see the *Client/Server for Windows Installation Guide* or *Client/Server for UNIX Installation Guide*. Note, however, that the chapters in those guides referring to the data client are ODBC-specific, and you should consult this manual for JDBC-specific information.

This document contains visual cues to help the reader identify important information

Table Convention	**Indicates**
Initial Capitals	Menu names, command names, and dialog box, window, and form titles.
Bold	Menu, command, and button names. Other elements to be selected or typed to accomplish an action. Label on input media used during installation.
Italic	Reference to a topic in the current document or another Relativity document. Reference to another document. Emphasis.
Bold Italic	Variables.

Technical Support

OpenText is dedicated to helping you achieve the highest possible performance from the RM/COBOL family of products. The technical support staff are committed to providing prompt and professional service to you when you have problems or questions about your Micro Focus products.

Technical support services are subject to OpenText's prices, terms, and conditions in place at the time the service is requested.

While it is not possible to maintain and support specific releases of all software indefinitely, we offer priority support for the most current release of each product. For customers who elect not to upgrade to the most current release of the products, there is free support available on the OpenText Community Forum: *community.microfocus.com*

Support Guidelines

When you need assistance, you can expedite your call by having the following information available for the technical support representative:

- Company name and contact information.
- Micro Focus Relativity product serial number (found in the Electronic Product Delivery email, or in the License Certificate).
- Micro Focus Relativity product version number.
- Operating system and version number.
- Hardware, and related equipment.
- Exact message appearing on screen.
- Concise explanation of the problem and process involved when the problem occurred.

Test Cases

You may be asked for an example (test case) of the source that demonstrates the problem.

- The smaller the test case is, the faster we will be able to isolate the cause of the problem.
- Do not send full applications.
- Reduce the test case to the smallest possible combination of components required to reproduce the problem.
- If you have very large data files, write a small program to read in your current data files and to create new data files with as few records as necessary to reproduce the problem.
- Test the test case before sending it to us to ensure that you have included all the necessary components to run the test case.

When submitting your test case, please include the following items:

1. README text file that explains the problems. This file must include information regarding the hardware, operating system, and versions of all relevant software (including the operating system and all Micro Focus products). It must also include step-by-step instructions to reproduce the behavior.
2. Program source files. We require source for any program that is called during the course of the test case. Be sure to include any copy files necessary for recompilation.
3. Relativity catalog. We require a Relativity catalog file that demonstrates the problem.
4. Data files required by the programs. These files should be as small as possible to reproduce the problem described in the test case.

Contact Information

Our Web site gives up-to-date details of contact numbers and addresses.

Additional technical information or advice is available from several sources.

The product support pages contain considerable additional information, including the WebSync service, where you can download fixes and documentation updates. To connect, enter *http://www.microfocus.com* in your browser to go to the OpenText home page.

If you are a OpenText Support for Micro Focus Products customer, please see your Support Handbook for contact information. You can download it from our Web site or order it in printed form from your sales representative. Support from OpenText may be available only to customers who have maintenance agreements.

You may want to check these URLs in particular:

- *https://www.microfocus.com/products/relativity/* (trial software download and OpenText Community files)
- *https://www.microfocus.com/support-and-services/documentation/* (documentation updates and PDFs)

Installation

This chapter lists the system requirements and describes how to install the Relativity Java Client component on a Microsoft Windows or UNIX server.

You can install the Relativity Java Client to a hard drive for use with a Java application. If you are installing to an internet server, be sure to specify the internet server's www directory as the installation directory. If you install the documentation or sample java application to this directory, they will be available on your web site.

While this is desirable in a development environment, it is not desirable when deploying your application.

System Requirements

Installation of the Relativity Java Client on a Windows-based or UNIX workstation for application development requires the following components:

- Java Development Kit version 1.8.*x*.
- The CLASSPATH environment variable should contain the complete pathname of the `RelJDBC.jar` file. See *Setting the CLASSPATH Variable* for more information.
- Relativity Data Server Version 12.20 for Relativity for RM/COBOL installed on a Windows or UNIX server.

Supported Windows platforms are Windows 7 or later, Windows Server 2008 R2, 2012 R2, 2016 and 2019. Supported UNIX platforms are AIX 7.1 TL4 SP0 or later, RedHat 5.9 or later and SUSE 11 SP2 or later.

Before You Install Relativity Java Client

Before you install, check to make sure that the following system requirements have been met. The client and server may not be ready to handle data access until these recommended steps have been taken. This checklist is a suggestion to minimize disruption to end users.

Recommended Server Configuration

- Relativity Data Server for Windows or UNIX must be installed and running on at least one server that is accessible from the client machine.
- TCP/IP networking option must be installed and running.
- Configure at least one server data source using the Relativity Server Administration utility to test with Relativity Java Client after installation.

See the *Relativity Client/Server for Windows Installation Guide* or *Relativity Client/Server for UNIX Installation Guide* for details. The examples in this document assume that the Verify data source (created as part of the *Installation and Verification of Relativity Data Client for Catalog Development* chapter) has been set up.

Recommended Client Configuration

- TCP/IP networking option must be installed and running. The client must be able to connect to the Relativity Data Server.

Setting the CLASSPATH Variable

Before using Relativity Java Client and the sample JdbcSample, you need to set the CLASSPATH variable to include the following files:

- `RelJDBC.jar`
- `JdbcSample.jar`

On Windows, the default installation location for `RelJDBC.jar` is `C:\Program Files (x86)\Micro Focus\RM\Relativityv12\JDBC43` installed by the Relativity Client installer. The default installation location for `JdbcSample.jar` is `%PUBLIC%\Documents\Micro Focus\RM\Relativity \JClient43`.

Installing Relativity Java Client on Windows

To install Relativity Java Client on a Windows PC, perform the following steps:

1. If you don't have the installation program, visit *Micro Focus Support*, click **Downloads** and enter your login details, and then download the **Relativity Java Client for Windows** from your order. This installer will have a name like `rmrelativity`*`version`*`_reljdbc.msi`.
2. Start the installer by double-clicking the `.msi` file.
3. Follow the instructions presented by the installation program.

   Important: If you are installing the Relativity Java Client on an Internet server, select the root document directory as the destination directory. For example, for IIS, this is often `C:\inetpub \wwwroot`.
4. Continue to follow the instructions until you have completed the installation process.
5. Click **Finish** to close the Relativity Installation window or click the Micro Focus icon to visit the Micro Focus Web site to register your product.
6. To test the installation, proceed to the section *Verifying Installation of the Relativity Java Client* in this chapter.

Installation Messages

The following messages may appear during the installation of the Relativity Java Client.

```
Beginning installation.
```

This message indicates that the installation script is beginning installation of the Java Client.

```
Installing files in /home/user/RelJDBC.
```

This message precedes the copying of the installation files into the installation directory. If the copy fails, a message indicating the source of the failure will appear instead, and the installation will terminate.

```
Install was successful.
```

This message indicates that the installation was successful.

Verifying Installation of the Relativity Java Client

The installation can be verified by running the sample application, `JdbcSample`. This application runs directly from the Java Application Launcher (`java`). A Java Development Kit (JDK) 1.8 or later is required.

1. On the Relativity server, create a data source. For example, create a data source **Shirt3Server** using the sample `Shirt3.rcg`.

2. Launch the `JdbcSample` JDBC application. To launch the sample application from the Java Application Launcher, make the installation directory be your current directory and issue the following command:

   ```
    java -cp JdbcSample.jar;RelJDBC.jar JdbcSample -d Shirt3Server -u
   "DB3OWNER" -p 1234 -q "SELECT * from backorder"
   ```

3. The `-cp` sets the classpath. The remaining options are command line arguments of `JdbcSample`. To see the full list of `JdbcSample` command line arguments enter:

   ```
    java -cp JdbcSample.jar;RelJDBC.jar JdbcSample
   ```

4. You may need to grant some permissions to `JdbcSample`. To do so, add a Java command line option to redefine `java.security.policy`. For example:

   ```
    java -Djava.security.policy=JdbcSample.pol -cp JdbcSample.jar;RelJDBC.jar
   JdbcSample -d Shirt3Server -u "DB3OWNER" -p 1234 -q "SELECT * from
   backorder"
   ```

Connecting to Data Sources

This chapter provides instructions for and examples of developing and connecting to the Relativity Java Client driver. It assumes you have a basic knowledge of Java and SQL. For a detailed description of Java and the JDBC standard, visit *JDBC 4.3 Specification*. For further information about SQL, consult the documentation of your database management system.

Instructions

You can connect to a data source in four easy steps. Refer to the Sample Code at the end of these steps for more details.

1. Load and register the driver with the JDBC Driver Manager. A driver can be loaded and registered in three ways:

   a. Call the Class.forName() with the following syntax:

      ```
      Class.forName("relativity.jdbc.Driver");
      ```

      The driver will automatically register itself with the JDBC Driver Manager when it is loaded.

   b. Create an instance of the driver with the following syntax:

      ```
      relativity.jdbc.Driver sd = new
      relativity.jdbc.Driver();
      ```

      Add the driver to the java.lang.System property "jdbc.drivers". This is a list of driver class names, separated by colons, that the DriverManager class loads. When the DriverManager class is initialized, it looks for the system property "jdbc.drivers", and if the user has entered one or more drivers, the DriverManager class attempts to load them.

2. Create a JDBC URL specifying which data source with which you want to connect. The format of a JDBC URL is:

   ```
   jdbc:relativity://<server name>:<server port>/<data source name>
   ```

   Table 1: Table 1: JDBC URL parameters

JDBC URL parameters	Description
jdbc:relativity	Identifies the driver.
<server name>	The name of the Data Server.
<server port>	The port that the Data Server is listening on. This value must be an integer. The standard value is 1583.
<data source name>	The ODBC name of the data source.

3. Create and set the connection properties. Properties can simply be the user name and password or more detailed properties for more advanced data sources. For a list of advanced properties that can be set for the Relativity Data Server, see *Table 2: Java Client Properties*.

4. Call DriverManager.getConnection to specify the URL and any data source-specific properties and Relativity-specific connection properties. The following table lists the Relativity-specific properties.

   Table 2: Table 2: Java Client Properties

Connection Property	Description

ArrayFetchOn	Turns array fetching on or off. Setting this value to 1 turns array fetching on and 0 turns it off. Default is 1 (optional).
ArrayBufferSize	Sets the network transmission size when array fetching is on. Default is 8 KB. Note that the range for this value should be between 1 and 64. Values greater than 16 have not shown to give any performance benefit.
User	Specifies the username with which to connect to the data source.
Password	Specifies the password with which to attempt to connect to the data source.
ExecDesc	Short description of application/applet that will appear in Server Status dialog of the Relativity Server control panel applet.

Note: Connection properties that are not recognized by Java Client will be added to the ODBC connection string passed to the Data Server.

Sample Code

```
import java.sql.*;
import java.util.Properties;
public class sample {
    public static void main(String[] args) {
    Connection m_Connection = null;
    String szURL = new String("jdbc:relativity://DataServerName:1583/
Verify");
    // Load and register the driver
    try {
        Class.forName ("relativity.jdbc.Driver");
    } catch (java.lang.ClassNotFoundException e) {
        System.out.println("The driver could not be loaded : " +
e.getMessage());
    }
    //Create the connection properties
    Properties props = new Properties();
    props.put("user", "DB3 Owner");          //username
    props.put("password", "1234");          //password
    props.put("ExecDesc", "Sample");          //Executable Description
    props.put("ArrayFetchOn", "1");          //Turn array fetching on
    props.put("ArrayBufferSize", "8");         //Set array fetching size to
8K
    try {
        m_Connection = DriverManager.getConnection(szURL, props);
        /* Below is an alternative means of obtaining a connection.
        m_Connection = DriverManager.getConnection(szURL, "DB3 Owner",
"1234"); */
        m_Connection.clearWarnings();
    } catch (SQLException e) {
        System.out.println("The driver could not connect : " +
e.getMessage());
    }
    try {
        m_Connection.close();
    } catch (java.sql.SQLException e) {
        System.out.println("Cannot close connection: " + e);
        System.exit(1);
    }
    }
}
```

Implementing Multi-Threaded Access

Relativity Java Client is thread safe and supports multi-threaded applications. However, if you are using multi-threaded database access regularly, it is better to create multiple connections for each thread. If only one connection is used with multiple threads, only one thread is allowed to communicate over the network at once. This potential bottleneck is removed if each thread has its own connection.

Using the Sample

Included with the Relativity Java Client is a sample applet. Use this sample, JdbcSample, to test your data source connection.

Required Directory Structure

For JdbcSample to work, the `RelJDBC.jar` file must be in the same directory as `JdbcSample.jar` or referenced by CLASSPATH. (See *Setting the CLASSPATH Variable* in the Installation chapter for more information.)

File	Description
`JcbcSample.java`	JdbcSample.class source code.
`JdbcSample.jar`	Sample application.
`RelJDBC.jar`	The Relativity Java Client JDBC Driver itself.

Usage

```
java JdbcSample -d dsn [-u user] [-p pwd] [-port port] [-s server] [-q query]
```

where:

- `dsn` is the data source name of the Relativity database. This is required.
- `user` is the username to log in as
- pwd is the password for the user
- `server` is the Relativity server name or IP
- `port` is the port of the Relativity protocol
- `query` is the SELECT statement

All arguments except `dsn` are optional. `user`, pwd and `query` default to empty strings. `server` defaults to localhost. `port` defaults to 1583. If `query` is the empty string, then TestConnection connects to the Relativity database and immediately disconnects. Otherwise, it connects, runs the query, shows the results, and disconnects.

Given that Shirt3Server provides access to the sample Shirt3 catalog, the following is a console capture of the sample. The sample retrieves and displays the rows from the backorder table.

```
c:\Users\Public\Documents\ Documents\Micro Focus\RM\Relativity
\JClient43>java.exe -cp "JdbcSample.jar;c:\Program Files\ Micro Focus\RM
\Relativityv12\JDBC43\RelJDBC.jar" JdbcSample -d Shirt3Server -u "DB3 OWNER" -
p 1234 -q "SELECT * from backorder"
Relativity URL: jdbc:relativity:localhost:1583/Shirt3Server
User: DB3 OWNER
Connecting to database...
Connected.
Creating query statement...
Created.
Extracting data...
ProductNumber,ProductSize,Color,PricePerUnit,Price4OrMore,BackOrderQuantity,Da
teStockExpected
AP1927367D,20W ,17,32.00,29.99,22,19940803
AP1927466D,24W ,03,12.00,10.00,0,19940705
AP2823987D,ML  ,03,19.99,17.00,20,19940721
```

```
AP2824597D,ML  ,17,28.00,25.99,30,19940701
AP2824621D,MS  ,45,20.00,16.99,120,19940824
AP2824712D,MM  ,03,12.00,10.00,26,19940724
PF5430319B,S   ,37,18.00,15.30,120,19940615
PF5430442B,S   ,37,16.00,13.60,100,19940615
PF5430467B,M   ,73,16.00,13.60,50,19940601
PF5431036D,L   ,78,22.99,18.79,2,19940630
PF5432000D,L   ,26,24.00,20.40,102,19940715
PG5430418D,XLXT,34,20.00,17.00,40,19940630
PG5430418D,XLXT,37,20.00,17.00,10,19940614
PG5430418D,XLXT,73,20.00,17.00,130,19940701
Extracted.
Disconnected.
```

JDBC Core API Interfaces

A set of interfaces is included in the standard JDBC API for application programmers to open connections to data sources, execute SQL, and process results. A brief description of the interfaces and classes is given here. For more detailed information, see the *JDBC 4.3 Specification*.

Table 3: Table 3: JDBC Interfaces

Interface Name	Description
java.sql.Driver	The Driver interface is responsible for creating a connection to the data source.
java.sql.Connection	The Connection interface represents a connection to the data source. It can create all types of statements as well as provide information about the data source through the java.sql.DatabaseMetaData interface.
java.sql.DatabaseMetaData	The DatabaseMetaData interface provides detailed information about a data source and a database management system's (DBMS) capabilities and requirements.
java.sql.Statement	The Statement interface is created from a connection. It allows the user to execute simple SQL statements.
java.sql.PreparedStatement	The PreparedStatement interface is a statement that is pre-compiled by the DBMS for higher performance. It has input parameters that can be changed between executions.
java.sql.CallableStatement	The CallableStatement is much like a prepared statement. It also allows output parameters as well as input parameters. This interface is useful for executing stored procedures that have output parameters.
java.sql.ResultSet	The ResultSet interface is returned by any of the statement interfaces when a SQL query is executed. This interface allows the use to retrieve the results of the query in several formats.
java.sql.ResultSetMetaData	The ResultSetMetaData interface describes a result set. It provides information like how many rows there are and what data type is in each row.
javax.sql.DataSource	A factory for connections to the physical data source that this DataSource object represents. An alternative to the DriverManager facility, a DataSource object is the preferred means of getting a connection. An object that implements the DataSource interface will typically be registered with a naming service based on the Java™ Naming and Directory (JNDI) API.
javax.sql.PooledConnection	An object that provides hooks for connection pool management. A PooledConnection object represents a physical connection to a data source. The connection can be recycled rather than being closed when an application is finished with it, thus reducing the number of connections that need to be made.
javax.sql.ConnectionPoolDataSource	A factory for PooledConnection objects. An object that implements this interface will typically be registered with a

	naming service that is based on the Java™ Naming and Directory Interface (JNDI).
javax.sql.RowSet	A RowSet object contains a set of rows from a result set or some other source of tabular data, like a file or spreadsheet. Because a RowSet object follows the JavaBeans™ model for properties and event notification, it is a JavaBeans component that can be combined with other components in an application.

Table 4: Table 4: JDBC Classes

Class Name	**Description**
java.sql.DriverManager	The DriverManager class handles all of the JDBC drivers. It provides methods to get connections to data sources. It decides if any driver is appropriate for the specific connection requested and then communicates with that driver. Any JDBC driver that may be used should be registered with the DriverManager.
java.sql.DriverPropertyInfo	The DriverPropertyInfo class contains methods to determine the information the driver needs to connect with a data source.
java.sql.Types	The Types class is an enumeration of all of the SQL data types.
java.sql.Time	The Time class represents a SQL time data type.
java.sql.TimeStamp	The TimeStamp class represents a SQL timestamp data type.
java.sql.Date	The Date class represents a SQL date data type.

Appendix B: JDBC Extensions

The following member functions are available from the Relativity Java Client classes, in addition to the functionality provided by the *JDBC 4.3 Specification*.

Note: To use the Relativity Java Client-specific extensions to JDBC, you must cast the java.sql interface to the Relativity type.

For example:

```
import java.sql.DriverManager;
import relativity.jdbc.*;

java.sql.Connection connection = DriverManager.getConnection(url, "sa", "");
java.sql.PreparedStatement preparedStmt;
relativity.jdbc.RelativityParameterMetaData parameterMetaData;
    preparedStmt = connection.prepareStatement(sql);
    parameterMetaData =
(RelativityPreparedStatement)preparedStatement).getParameterMetaData();
```

RelativityPreparedStatement

Function Name	Function Description
getParameterMetaData()	This function returns a class that describes parameters in a prepared statement if it is available. If the statement contains no parameters this function returns null. The RelativityParameterMetaData class is provided below.

RelativityParameterMetaData

The RelativityParameterMetaData class is provided to describe parameters for prepared and callable statements.

Name	Description
int getParameterCount()	Returns the number of parameters in the statement.
int getParameterType(int param)	Returns the SQL type of the parameter, will be one of the enumerated types in java.sql.Types.
String getParameterTypeName(int param)	Returns the name of the parameter's SQL type.
int getPrecision(int param)	Returns the precision of the parameter.
int getScale(int param)	Returns the scale of the parameter.
int isNullable(int param)	Returns whether the parameter can be null or not.

Copyright and Disclaimer