Rocket Model 204 Connect

Installation and Programming Guide

Version 7 Release 5.0

October 2014
CSTAR–75–UG-01

Notices

Edition

Publication date: October 2014
Book number: CSTAR–0705–UG-01
Product version: Version 7 Release 5.0

Copyright

Trademarks

Examples

This information might contain examples of data and reports. The examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

License agreement

Note: This product may contain encryption technology. Many countries prohibit or restrict the use, import, or export of encryption technologies, and current use, import, and export regulations should be followed when exporting this product.

Corporate Information

Rocket Software, Inc. develops enterprise infrastructure products in four key areas: storage, networks, and compliance; database servers and tools; business information and analytics; and application development, integration, and modernization.

Website: www.rocketsoftware.com

Rocket Global Headquarters
77 4th Avenue, Suite 100
Waltham, MA 02451-1468
USA

Contacting Technical Support

If you have current support and maintenance agreements with Rocket Software and CCA, contact Rocket Software Technical support by email or by telephone:

Email: m204support@rocketsoftware.com

Telephone :

North America	+1.800.755.4222
United Kingdom/Europe	+44 (0) 20 8867 6153

Alternatively, you can access the Rocket Customer Portal and report a problem, download an update, or read answers to FAQs. You will be prompted to log in with the credentials supplied as part of your product maintenance agreement.

To log in to the Rocket Customer Portal, go to:

www.rocketsoftware.com/support

Contents

About this Guide

This guide contains information about installing and using the Connect★ Suite.

Audience

The audience for this document includes end users, application developers, installers, system managers, and file managers who want to access data stored in Model 204.

A note about User Language and SOUL

Model 204 version 7.5 provides a significantly enhanced, object-oriented, version of User Language called SOUL. All existing User Language programs will continue to work under SOUL, so User Language can be considered to be a subset of SOUL, though the name "User Language" is now deprecated. In this guide, the name "User Language" has been replaced with "SOUL."

Rocket Model 204 documentation set

To access the Rocket Model 204 documentation, see the Rocket Documentation Library (http://docs.rocketsoftware.com/), or go directly to the Rocket Model 204 documentation wiki (http://m204wiki.rocketsoftware.com/).

Documentation conventions

This guide uses the following standard notation conventions in statement syntax and examples:

Convention	Description
`TABLE`	Uppercase represents a keyword that you must enter exactly as shown.
`TABLE` *`tablename`*	In text, italics are used for variables and for emphasis. In examples, italics denote a variable value that you must supply. In this example, you must supply a value for *tablename*.
`READ [SCREEN]`	Square brackets ([ ]) enclose an optional argument or portion of an argument. In this case, specify READ or READ SCREEN.
`UNIQUE | PRIMARY KEY`	A vertical bar (|) separates alternative options. In this example, specify either UNIQUE or PRIMARY KEY.
`TRUST | NOTRUST`	Underlining indicates the default. In this example, NOTRUST is the default.

Convention	Description
`IS {NOT | LIKE}`	Braces ({ }) indicate that one of the enclosed alternatives is required. In this example, you must specify either IS NOT or IS LIKE.
`item ...`	An ellipsis (. . .) indicates that you can repeat the preceding item.
`item ,...`	An ellipsis preceded by a comma indicates that a comma is required to separate repeated items.
`All other symbols`	In syntax, all other symbols (such as parentheses) are literal syntactic elements and must appear as shown.
`nested-key` `::=` *`column_name`*	A double colon followed by an equal sign indicates an equivalence. In this case, *nested-key* is equivalent to *column_name*.
`Enter your account:` `sales11`	In examples that include both system-supplied and user-entered text, or system prompts and user commands, boldface indicates what you enter. In this example, the system prompts for an account and the user enters **sales11**.
File > Save As	A right angle bracket (>) identifies the sequence of actions that you perform to select a command from a pull-down menu. In this example, select the Save As command from the File menu.
E`DIT`	Partial bolding indicates a usable abbreviation, such as E for EDIT in this example.

1

Before You Install

Overview

The Connect★ Suite supports three communications interfaces:

- .NET Framework
- JDBC
- ODBC

Each interface has its own database connectivity program that supports both SQL and RCL connections. The Connect★ Visual Interface is provided to populate the SQL catalog file (CCACAT) and for quick query capability to Model 204.

SQL conformance level

The SQL conformance level is ANSI 89 with most ANSI 92 functionality.

Communication protocol

Each interface uses the TCP/IP protocol to connect a PC workstation to Model 204 running on a mainframe.

IPv6 address support for Connect★

All Connect★ clients support Internet Protocol Version 6 (IPv6) 128-bit addresses as well as IPv4 32-bit addresses.

Connect★ clients have the option of specifying explicit server addresses in either of the following formats:

- IPv4 dotted decimal format; for example:

  ```
  74.6.238.254
  ```

- IPv6 colon-separated hexadecimal format; for example:

  ```
  2001:0DB8:AC10:FE01:0000:0000:0000:0000
  ```

  Consecutive zero halfwords can be replaced by a double colon, for example:

  ```
  2001:0DB8:AC10:FE01::
  ```

If an application specifies the host address as a symbolic name, then the client first attempts an IPv6 connection to the server. If that fails, the client attempts an IPv4 connection. An error is not returned to the application unless both attempts fail.

Requirements In order to take advantage of IPv6 addresses, you might have to install and enable IPv6 support on your client operating system. See your client O/S documentation for instructions.

Installation requirements

System requirements

For the Model 204 mainframe

- TCP/IP for operating systems z/OS, z/VM, or z/VSE
- IBM operating system: z/OS, z/VM, or z/VSE
- IBM TCP/IP V2 or higher
- Model 204 V6R1 or higher, configured for use with Connect ★ SQL and/or RCL connections. For more information, see the Rocket Model 204 installation instructions for your operating system and the *Rocket Model 204 SQL Connectivity Guide*.

For Connect★ workstation or server

The Connect* ODBC interface is 32-bit and provides compatibility with any program or operating system capable of running or supporting 32-bit applications. This includes Windows XP (32-bit or 64-bit), Windows 7 (32-bit or 64-bit), Java Virtual Machine (32-bit or 64-bit), or other similar environments.

The Connect* JDBC driver operates with either a 64-bit or 32-bit Java Virtual Machine.

The Connect* .NET interface version 7.5 provides both 32-bit and 64-bit drivers that can be used with any Web or GUI-driven application.

ODBC support

- Microsoft TCP/IP
- Microsoft platforms:
  - Windows XP
  - Windows Vista
  - Windows 2003
  - Windows 7

JDBC support

- Microsoft TCP/IP
- Model 204 7.4.0 -- required if you are using BLOBs and CLOBs

  Otherwise, Model 204 7.4.0 is not required; Connect★ version 7.4.0 is compatible with previous supported versions of Model 204.

  Model 204 7.4.0 is compatible with previous supported versions of Connect★.
- Java SE Development Kit (JDK) or Java SE Runtime Environment (JRE) Version 1.5 or later
- Platforms: Any operating system that supports Java Runtime Environment

.NET Framework support

- Microsoft TCP/IP
- .NET Framework 2.0 or higher
- Microsoft platforms:
  - Windows XP
  - Windows Vista
  - Windows 7
  - Windows Server 2003
  - WIndows Server 2008

Preparing to install

Before installing Connect★, make sure to take care of the following items:

Take care of…	Consult…
Installing TCP/IP software on the mainframe and the PC	Model 204 installation instructions for your site and the appropriate installation guide for your PC
Getting an IP address for the z/OS system Model 204 runs in. There is only one IP address for each system.	TCP/IP manager at your site.
Getting a SERVPORT number, the local port number that remote clients use to connect to the Model 204 link.	TCP/IP manager at your site.
Defining input threads, such as IODEVs 19 and/or 49, and setting CCAIN parameters	Rocket Model 204 documentation wiki system management pages: http://m204wiki.rocketsoftware.com/index.php/Defining_the_User_Environment_(CCAIN) Rocket Model 204 documentation wiki parameter pages for a complete description of parameters and their values: http://m204wiki.rocketsoftware.com/index.php/List_of_Model_204_parameters
Defining links and processgroups for your applications	Rocket Model 204 documentation wiki for the DEFINE LINK and DEFINE PROCESSGROUP commands for TCP/IP: http://m204wiki.rocketsoftware.com/index.php/Category:Commands
Making sure you have your username and password for ftp.rocketsoftware.com	Rocket Technical Support
Removing your existing copy of Connect★ ODBC	“Uninstalling a previous version of Connect★ for ODBC” on page 14
Removing your existing copy of Connect★ for JDBC	“Removing a previous version of JDBC” on page 30
Removing your existing copy of .NET Framework	“Uninstalling a previous version of Connect★ for .NET Framework” on page 42

Most of the tasks in the previous table are handled by the Model 204 system administrator or Model 204 database administrator. The person who installs Connect★ might or might not also be the Model 204 system administrator or database administrator.

TCP/IP software requirements

To connect using TCP/IP, define a TCP/IP link in the Model 204 Online using a DEFINE LINK command. For correct coding, see the DEFINE LINK command page of the Rocket M204wiki: http://m204wiki.rocketsoftware.com/index.php/DEFINE_LINK_command:_Horizon_for_TCP/IP

Security interfaces for UNIX System Services (formerly OpenEdition)

If you are running with the sockets interface (IBM stack and z/OS 1.4 or above) then you must make security definitions within Open MVS to define Model 204 as a sockets application.

A useful reference manual is the IBM document SC28-1890, *UNIX System Services Planning* manual (formerly *OpenEdition Planning*). Among other subjects, this manual deals with security issues, including the setup of user IDs (UIDs) and group IDs (GIDs). This planning manual illustrates the security setup from a Security Server (formerly RACF) point of view.

As the planning manual mentions, you can use an equivalent security package instead of Security Server (RACF). However, you must verify that the version of the security package that you are running can handle UNIX System Services security.

Sample definitions for Security Server (RACF) users:

- For the owning USERID for the ONLINE address space, specify the following to define it to the OMVS segment:

  ```
  UID=0000000000
  HOME=/
  PROGRAM=/bin/sh
  ```

  When defining UNIX System Services users to Security Server (RACF), such as assigning a UID to the user, the USERID must have a UID of 0, as shown in this first example.

- For the group that owns the owning USERID, specify a group ID:

  ```
  GID=0000000002
  ```

  The value of 0000000002 in this example is the ID of the particular Security Server (RACF) group as defined to UNIX System Services.

In Security Server (RACF) terms, you can set up the UID and GID definitions with certain panels under TSO/ISPF that are used to maintain users and groups. There is an equivalent facility for Top Secret or ACF2.

2

Connecting to Model 204

Overview

Before you can use one or more of the Connect★ Suite communications interfaces (ODBC, JDBC, and/or .NET Framework), you must verify your connection to the Model 204 data and the ability to access the data in a query. The general steps are as follows:

1. Confirm the mainframe connection with the **ping** command.
2. Use the Connect★ Visual Interface (CVI) to confirm the connection to your Model 204 Online.

Confirming workstation-to-mainframe communication

Confirm the workstation-to-mainframe TCP/IP communication by using the workstation TCP/IP software to “ping” the mainframe on which the Model 204 Online resides:

1. Execute the **ping** command using one of the following methods:
   - In the DOS interface, enter the **ping** command with the IP address of the mainframe:

     `ping` *`IP-address`*

   - Double-click on a ping icon for your installed TCP/IP software and fill in the IP address.
2. Check the message displayed. If your command reached the TCP/IP address and was returned, you have a communication line to the mainframe. However, this does not mean that you have

connected to the Model 204 Online.

3. Use the CVI to confirm a connection to the Online. See “Installing the Connect★ Visual Interface (CVI)” on page 8 and “Defining the demonstration database to CCACAT” on page 9.

SQL, RCL and Freeway/204 connections

You can open multiple RCL and SQL connections in the same session.

SQL

When an SQL connection is established to the Model 204 database, the SQL statements are translated by Model 204 at the host, and the results are passed back to the client.

Before executing an SQL statement, check with your Connect★ administrator to make sure that either the demonstration database is installed and has been defined to the SQL catalog file, CCACAT, or you have other available tables defined in CCACAT. (See “SQL catalog population” on page 9.)

RCL

Using the RCL feature, you can communicate with Model 204 using SOUL and command syntax, wrapped in an SQL delivery vehicle. RCL returns Model 204 responses in a single-column variable-character-width table (up to 255 characters per row). The results are returned logically a line at a time (not using full-screen output).

RCL threads do not require an SQL catalog.

Freeway/204

With Freeway/204 you have access to two SQL and two RCL connections for free. This will require IODEVs 19 and 49 to support SQL and RCL connections. See the Rocket Model 204 documentation wiki for a discussion of IODEVs and how to set them up:

http://m204wiki.rocketsoftware.com/index.php/Defining_the_User_Environment_(CCAIN)

You must define a process, process groups and links in Model 204.

Installing the Connect★ Visual Interface (CVI)

Model 204 provides you with demonstration databases and the CVI, so you can run queries as soon as you have installed the Connect★ software. The CVI must be installed first.

Prerequisite

Before you install you must have the following in your working environment: Java Runtime 32-bit JRE (minimum version 1.6).

To install the CVI:

1. Navigate to the Rocket FTP site:

   ```
   ftp.rocketsoftware.com
   ```

2. Enter the userid and password provided by Rocket.
3. Open the ConnectStar/CVI folder and download the zip files to your PC, in **binary** format.
4. Run the .exe file to install the CVI application.

By installing the CVI you can enable a connection to a data source on Model 204.

Next you will use the CVI to define a data source. Then you can use the CVI to check whether you have successfully connected to Model 204 by entering a query.

SQL catalog population

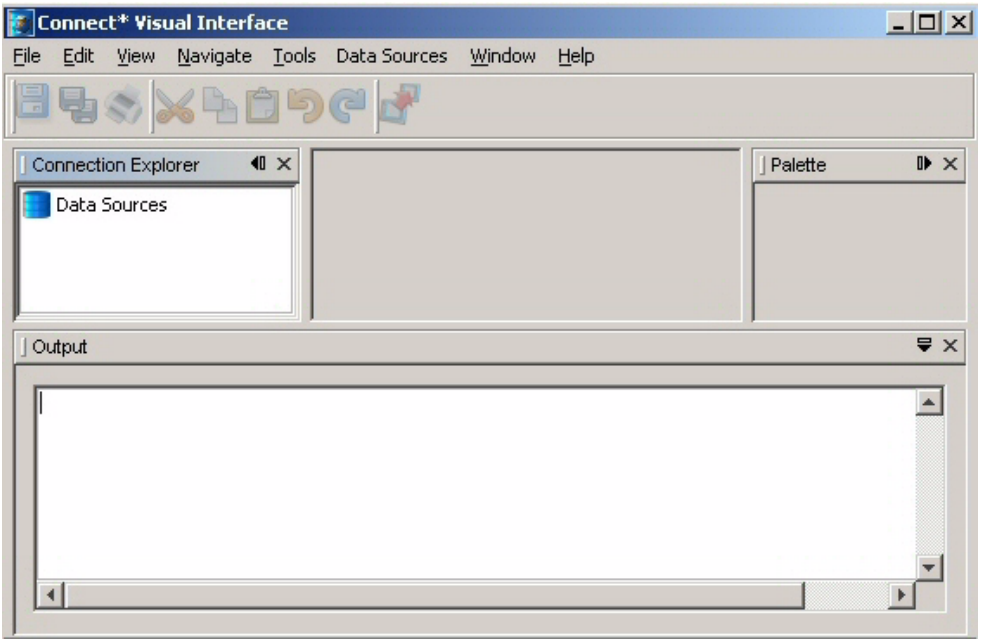
The file DEMOTAB.DDL is automatically copied into the C:\\Program Files\CCA\Connect Star for Model 204\CVI folder of each workstation when you install the CVI. The DEMOTAB.DDL file contains SQL Data Definition Language (DDL) statements, which define the Model 204 demonstration databases to the SQL catalog, CCACAT.

Note: The DEMOTAB.DDL file uses DEMO as the schema name. You can either use this name or choose your own. If you choose to use a different schema name, be sure to edit any files you use to replace DEMO with your own schema name.

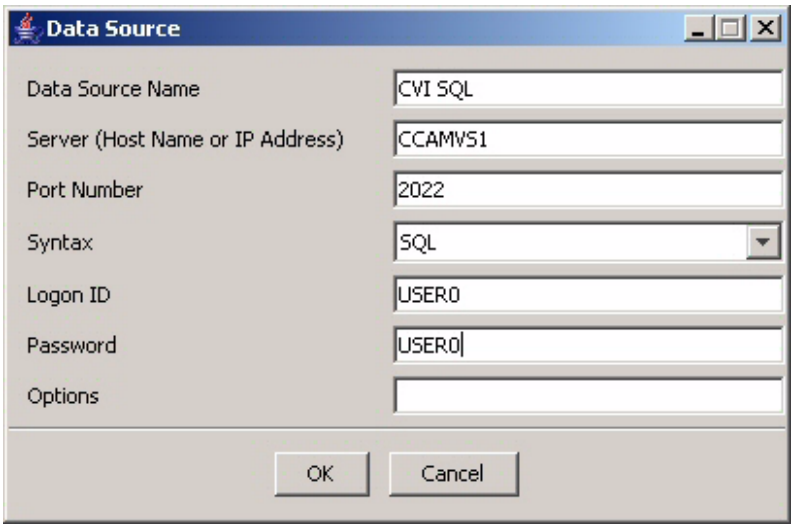
Defining the demonstration database to CCACAT

To define the demonstration database files to CCACAT:

1. Open the Connect Star Visual Interface software by double-clicking on the icon for the Connect Star For Model 204 program group in the Start > Programs menu.
2. Click on CVI, then on Connect Star Visual Interface. The following Connect★ Visual Interface dialog box opens.

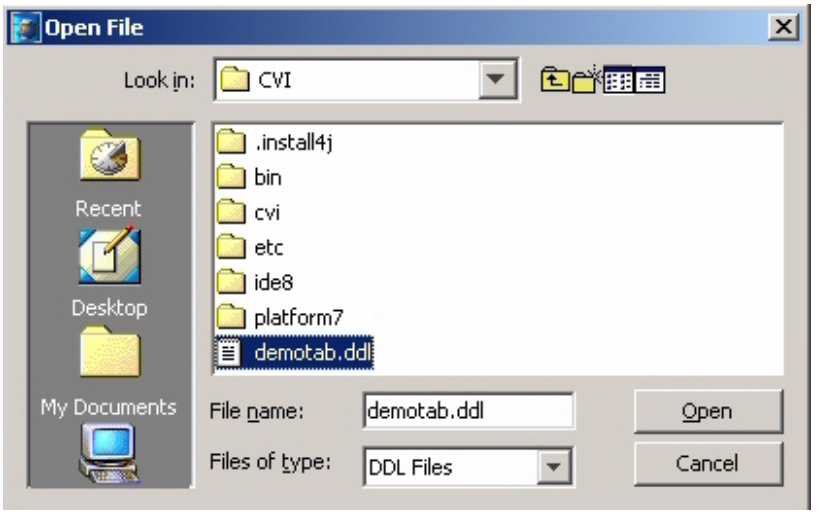


3. In the Connect★ Visual Interface dialog box, select the Data Sources > New option. The Data Source dialog box opens.

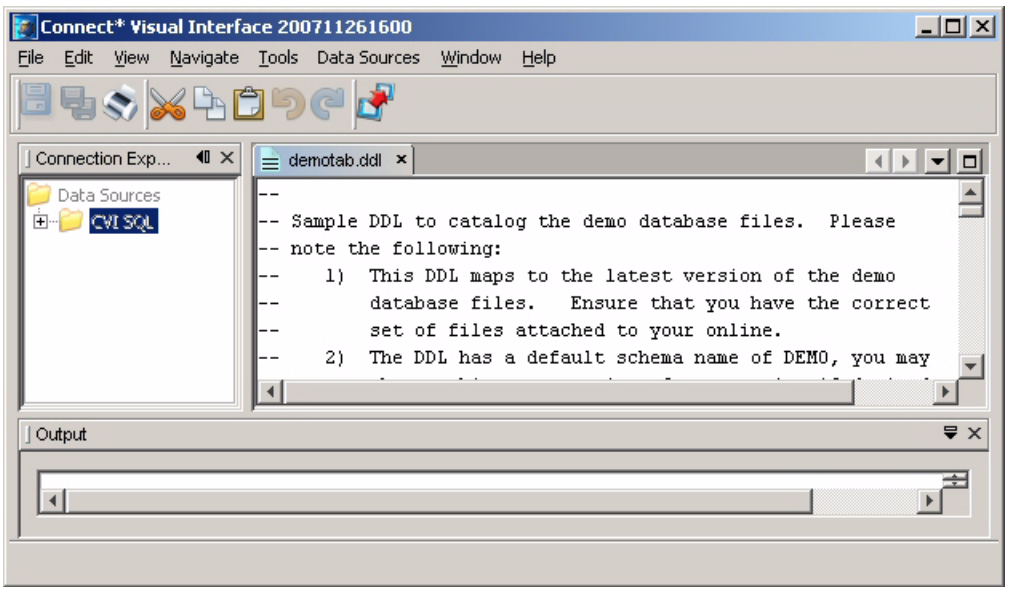


4. Define an SQL data source to your Model 204 Online and click OK.
5. The CVI dialog box is displayed. On the menu bar click Data Sources > Connect to establish a connection, then click OK in the message box. This step verifies your Model 204 Online connection.
6. Click the rightmost execute icon to execute the displayed "SELECT * FROM CATALOG.TABLES". This step verifies your SQL Catalog installation.

7. To use DEMOTAB.DDL as the input file, Select File > Open File and navigate to the installation folder. Select the demotab.ddl file.



8. Click on the Execute icon or enter Ctrl + Shift + X.

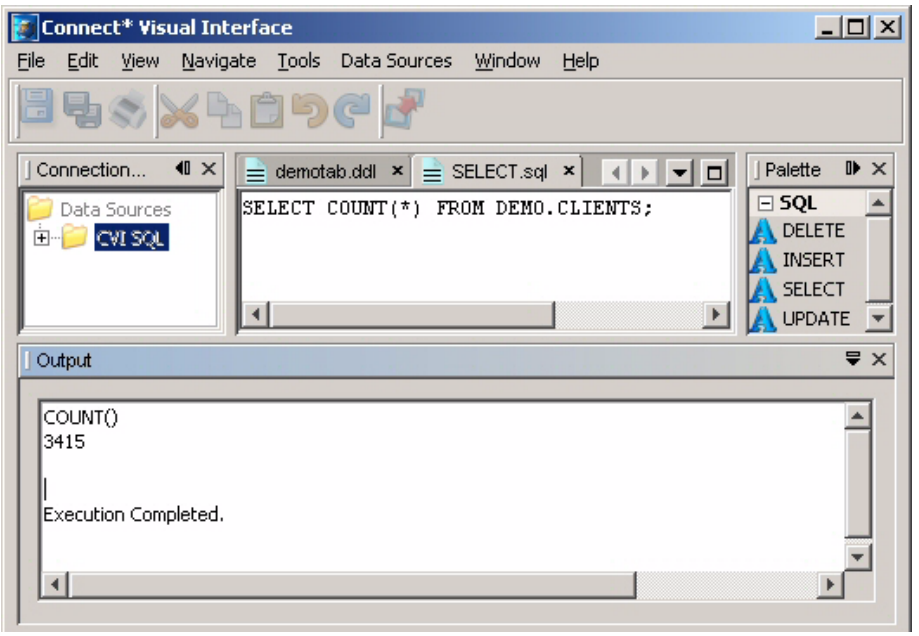


Successful execution of the demotab.ddl displays a message *Execution Completed* in the Output panel.

9. To verify definitions in CCACAT click on File > New File. Select SQL as the file type, then enter a file name.

10. In the New File dialog box type the following SQL query and click Execute.

```
SELECT COUNT (*) FROM DEMO.CLIENTS;
```



11. Do one of the following: click in the Output field to Clear, Save As...., or Print the output.

Creation of your own DDL input file

You can create the input file using one of the following:

- Table Specification Facility (CCATSF) subsystem running under Model 204
- Text editor to create DDL statements manually
- Third-party application that produces standard SQL DDL

Note: The DDL input file that you submit to CVI must use statement delimiters. See the *Rocket Model 204 SQL Server User's Guide* for information about using the CCATSF subsystem. You can run queries against this database using an application such as Microsoft Query.

You can use another optional Model 204 mainframe subsystem, CCACATREPT, to report on CCACAT.

See the *Rocket Model 204 SQL Server User's Guide* for information about using the CCACATREPT subsystem.

3

Connect★ for ODBC

Overview

Connect★ for ODBC is *multithreaded capable*. Connect★ provides a common memory stack that will be shared during access of a multithreaded environment or application. Connect★ is not multithreading.

The 32-bit Connect* for ODBC driver offers you the flexibility to use generic code with multiple ODBC-compliant database interfaces on the Windows Platform. To develop applications, use a Windows application such as:

- Microsoft Office (Excel, Access, Word)
- Microsoft Visual Studio (Visual Basic, or any ODBC compatible language)

Setting up the Connect★ for ODBC environment

ODBC conformance

The Model 204 32-bit ODBC driver supports conformance to ODBC Version 1.1 and runs with ODBC 2.0 and ODBC 3.0 applications. An informational message is returned for ODBC 2.0 and ODBC 3.0 requests that are not supported.

ODBC Installation steps

Complete these steps when installing Connect★:

Step	Action	See...
1.	Uninstall an earlier version of Connect★for ODBC	page 14
2.	Install Connect★ for ODBC	page 14
3.	Confirm the workstation-to-mainframe communication.	page 7
4.	Define an ODBC data source.	page 18
5.	Populate or load the SQL catalog and verify the Connect★ connection.	page 9

Uninstalling a previous version of Connect★ for ODBC

Running the Connect★ uninstaller removes the files, program groups, program folders, and registry entries created during Connect★ installation. It does not remove work files that you add to a Connect★ workstation.

Run the uninstaller for Connect★ using the Windows Control Panel to select the Add/Remove Programs icon.

In the Add/Remove Programs Properties dialog box, select the Connect★ software. Click on the Add/Remove button.

The uninstaller does not remove the following:

- any files added to the directory after the installation completed
- any items added to the program folder after the installation completed
- existing ODBC data sources

Installing the product for ODBC

1. Ensure that you have fulfilled the preinstallation requirements as described in “Preparing to install” on page 4.
2. Navigate to the FTP server:

   ftp.rocketsoftware.com
3. Enter the userid and password provided by Rocket.
4. Navigate to the ConnectStar File folder.
5. Open the ODBC folder and download the files to your PC, in **binary** format.
6. Run the .exe file to install the application.

Files installed

In addition to the files installed in your installation destination folder, Connect★ installs the following files in Program Files under Common Files/CCA:

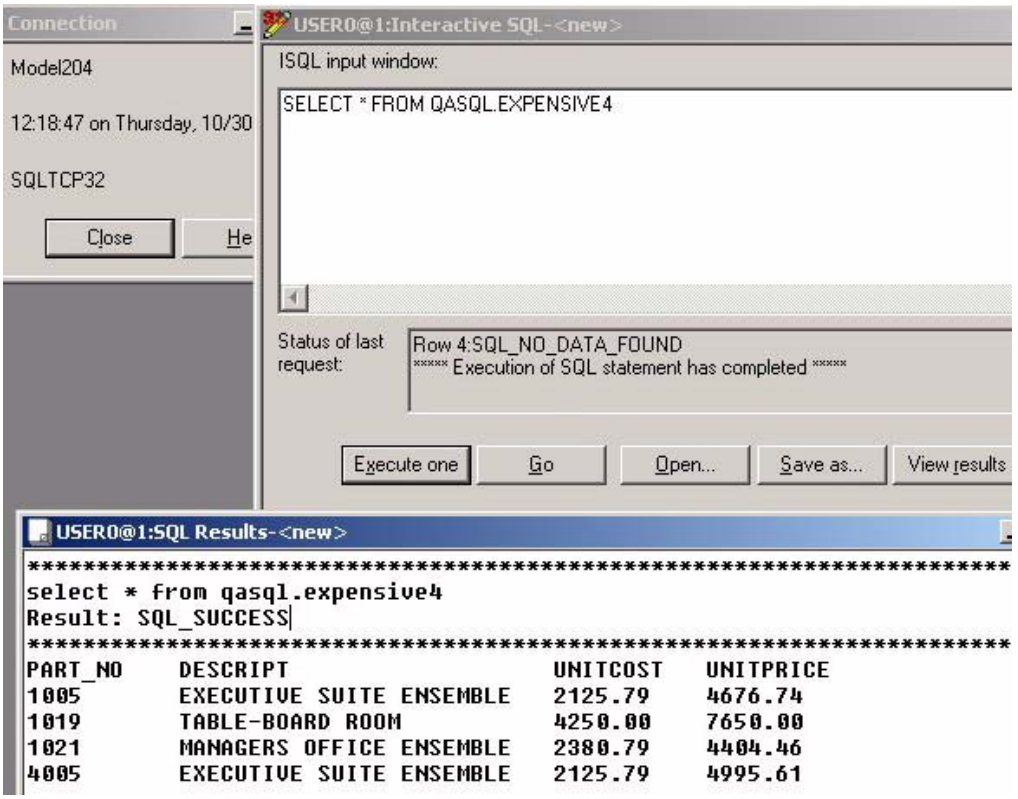
Installed for TCP/IP
M204IN32.DLL
M204OD32.DLL
M204RX32.DLL
M204SC32.DLL
M204TP32.DLL
M204UT32.DLL

Installation complete

When the installation is complete, you can use the Start > Programs > Connect Star for Model 204 > ODBC program group to access one of the following:

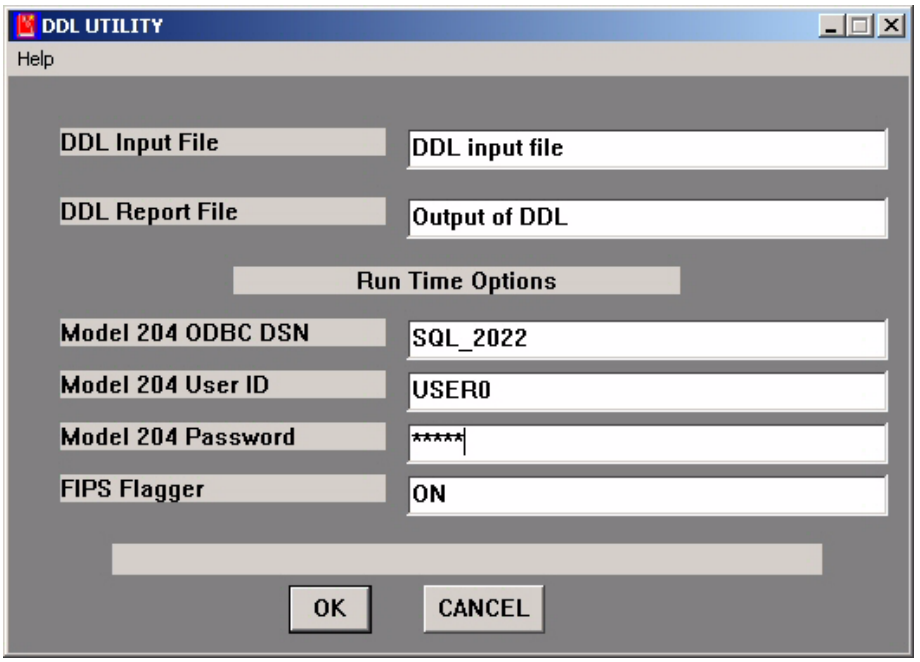
- Audit for
  - Turn logging on
  - Turn logging off

- Unsupported
  - CLIIVP



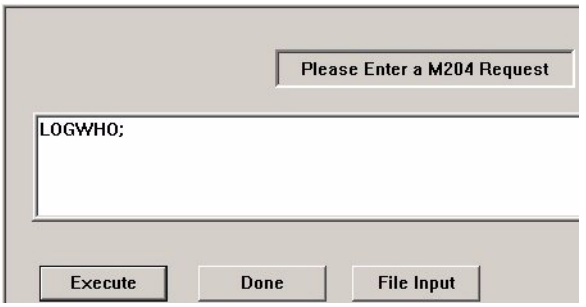
CLIIVP uses either an SQL or an RCL ODBC data source to connect to Model 204. It accepts a file or a command-at-a-time as input. Output is returned in a results window. You can edit and save both input and output.

– DDLWIN

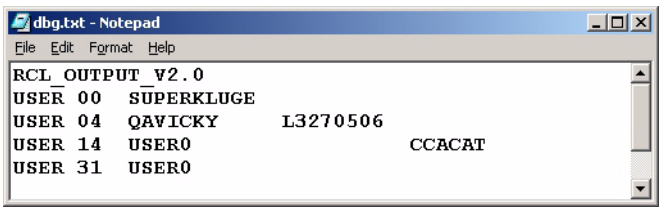


DDLWIN runs the DDL UTILITY which accepts a file of DDL statements and updates the CCACAT Model 204 file.

– TTRW



TTRW lets you select an ODBC data source and runs the SQL or RCL request you enter or provide in an input file. Output is written to dbg.txt in the ODBC > Unsupported folder.



- Catalog 2
  – Connect to data source, for a display of all tables
- Readme File

Defining an ODBC data source

Before executing an SQL statement, check with your Connect★ administrator to make sure that either the demonstration database is installed and has been defined to the SQL catalog file, CCACAT, or you have other available tables defined in CCACAT. (See “SQL catalog population” on page 9.)

Selecting an ODBC data source is required to establish an ODBC connection to the Model 204 Online.

Use the Model 204 ODBC Driver — Configure Data Source dialog box to create data sources that can be accessed by ODBC or by other applications such as Microsoft Access, Visual Basic, or PowerBuilder.

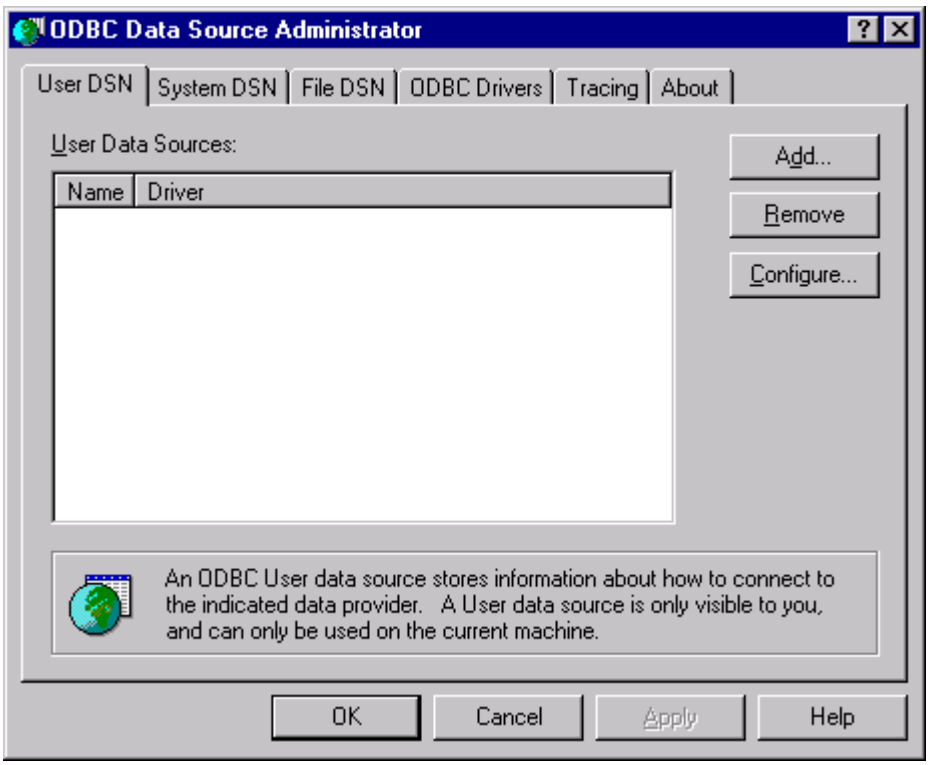
You can define more than one data source, each identified by its data source name, with different settings in each one, to the same Model 204 Online.

To give your workstation greater flexibility, you might want to configure the same Model 204 Online more than once, as SQL and RCL (see “Connection Type” on page 21).

Creating a data source

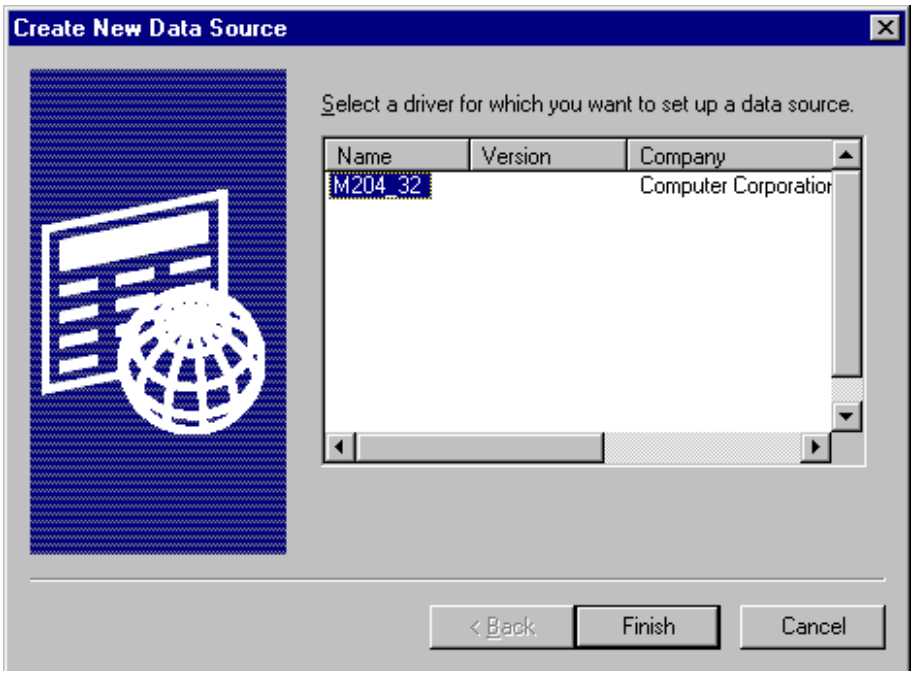
To create a data source:

1. In the Control Panel click on Administrative Tools. Double-click on the Data Sources (ODBC) icon to display the ODBC Data Source Administrator dialog box.

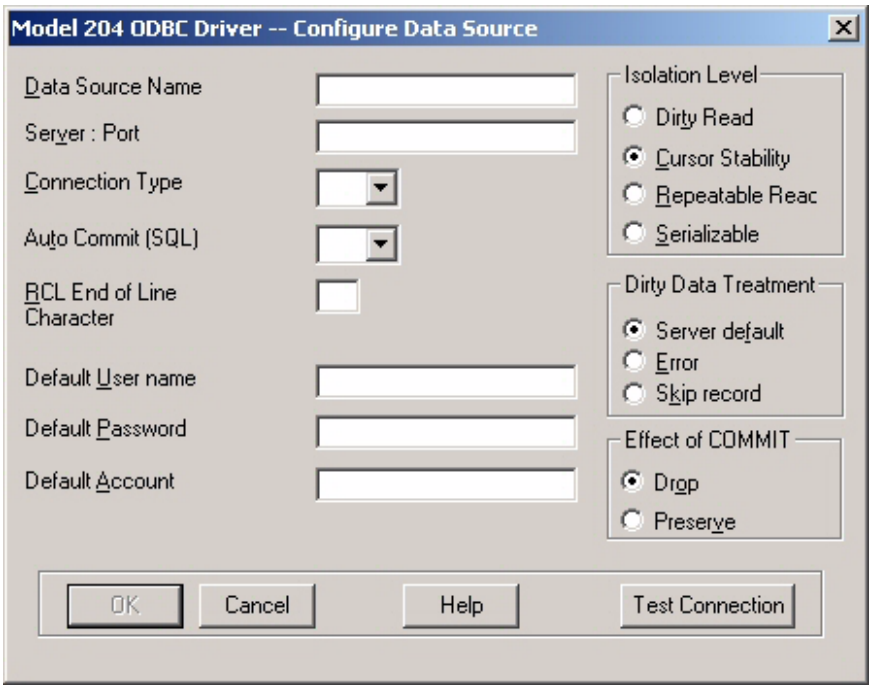


2. Create a user or system data source by clicking on the Add button in the User DSN or System DSN tab.

3. Select the M204_32 driver in the Create New Data Source dialog box.



4. Click on Finish to display the Model 204 ODBC Driver — Configure Data Source dialog box.



5. Define the data source by filling in the text boxes (see the section “Model 204 ODBC driver data source text boxes” on page 20) and click Test Connection button.

Changing a data source

To change an existing data source definition:

1. Open the 32-bit ODBC Data Source Administrator to display the ODBC Data Source Administrator dialog box.
2. Select a data source in the Data Source list box.
3. Click on Configure to display the Model 204 ODBC Driver — Configure Data Source dialog box.
4. Change the information in the text boxes and click on OK.

Model 204 ODBC driver data source text boxes

This section describes the individual text boxes in the Model 204 ODBC Driver — Configure Data Source dialog box.

Data Source Name

Each data source must have a unique name.

The name you enter is the name applications use to access the set of configuration information in the data source. This is the name that applications pass to the SQLDriverConnect function or the SQLConnect function. It corresponds to the DSN keyword.

You can enter any alphanumeric characters up to a maximum of 30 characters. Do not use blank spaces in the Data Source Name.

You can give the data source a name that is more meaningful to you than the (default) Model 204 name in the Server ID/Address field. The Data Source Name value resides on your workstation and is accessible only from your workstation. The specification for the data source name is in the ODBC.INI section of the registry.

The ODBC standard lets you create multiple data sources. If you want a default data source, you must explicitly identify the Data Source Name as “Default.” You are not required to create a default data source.

Server ID/ Address

The Server ID value specifies the IP address and the port number used by the Model 204 32-bit ODBC driver to establish a connection to a Model 204 server. The IP address and port number are typically assigned by the local systems programmer. Follow these guidelines:

- The IP address must be the IP address of the IBM z/OS or z/VM mainframe system as entered in the Model 204 Online to which you are connecting. The address is defined in the Define Link LOCALID=*xxxx*.
- The port number must be the number specified in the Model 204 TCP/IP link defined in the Online:

  ```
  Define Link SERVPORT=xxxx
  ```

- The IP address and port number are separated by a colon.
- Two IP address formats are possible: either the domain name or the numeric IP address. For example:

  ```
  M204MVS1:3001

  192.207.28.129:3001
  ```

Connection Type

Use the Connection Type text box to specify the use of SQL or RCL to establish a connection (IODEV) to the mainframe:

- With SQL, you can use SQL statements as specified in the *Rocket Model 204 SQL Server User's Guide*.
- With RCL, you can use SOUL requests and Model 204 commands and procedures.

Auto Commit (SQL)

The default SQL_AUTOCOMMIT setting for the M204_32 Driver is ON. This default setting implies that all inserts, updates, and deletes are committed to the database immediately after the transaction is applied, without the need to issue an SQLTransact call with SQL_COMMIT.

The AUTOCOMMIT default (ON) returns SQL_SUCCESS for update or delete transactions that find no rows to update or delete in the table.

To turn the SQL_AUTOCOMMIT option OFF, select the OFF option in the text box.

RCL End-of-Line Character

If you are establishing an RCL connection, accept the end-of-line character, the semicolon (;) character in the text box.

Default User Name

The Default User Name value is used by the Model 204 32-bit ODBC driver to log on to a Model 204 server. The Default User Name must be a valid Model 204 logon name for the requested Model 204 server.

If you do not enter a Default User Name, the application prompts the user for a valid name at connection time. The Default User Name corresponds to the UID keyword in either the SQLConnect or SQLDriverConnect call.

Default Password

The Default Password value must be the valid Model 204 password for the name entered as the Default User Name in the same data source.

If no Default Password is entered here, the application prompts the user for a valid password at connection time. The Default Password corresponds to the PWD keyword in either the SQLConnect or SQLDriverConnect call.

Default Account

The Default Account value sets the default account character string used by the Model 204 32-bit ODBC driver when establishing a connection to a Model 204 server. The Default Account value can be any account string acceptable to the requested server.

This value is optional. You can assign it to a project, for example, to record the tasks and the number of hours spent. If you omit it, the driver does not pass an Account string to the Model 204 server.

The Default Account corresponds to the ACCOUNT keyword in the SQLDriverConnect call.

Default settings installed with Connect★ software

Technical Support recommends that non-programmers do not change the following default settings provided with the installed Connect★ software:

- Isolation Level
- Dirty Data Treatment
- Effect of COMMIT

Getting Model 204 data

Before executing an SQL statement, check with your Connect★ administrator to make sure that either the demonstration database is installed and has been defined to the SQL catalog file, CCACAT, or you have other available tables defined in CCACAT. (See “SQL catalog population” on page 9.)

When the Model 204 ODBC driver installation is complete, use Catalog2, an ODBC data source schema browser, to verify that the connection to Model 204 works successfully:

1. Define an ODBC user or system data source before attempting to send a query to Model 204 (see “Defining an ODBC data source” on page 18).

2. In the Connect Star For Model 204 program group under ODBC, click on Catalog2.
3. Use File > Open to display the Select Data Source dialog box. Select a defined data source to display all the tables or views in the Model 204 SQL catalog (CCACAT). You can determine the types of tables and information included in the display by selecting Setting in the View menu.
4. Double-click on a table or view to display all the columns that are included in the table and information about the columns.

ODBC third-party software

The Model 204 32-bit ODBC driver works with nearly all ODBC 32-bit applications.

Application development using ODBC

If your application requires an ODBC connection, you must register your Model 204 as a data source with the Microsoft ODBC administrator on your workstation. Use the Microsoft 32-bit ODBC administrator to define a data source after the Model 204 ODBC driver has been installed. For information on:

- Installing the driver, see “Installing the product for ODBC” on page 14
- Defining a data source, see “Defining an ODBC data source” on page 18.

Connect★ for ODBC is multi threaded capable. Connect★ provides a common memory stack that will be shared during access of a multi threaded environment or application. Connect★ is not multithreading.

If you are a knowledgeable programmer, you can override the default data source settings, as described in this chapter.

Overriding data source settings

An ODBC application can override the isolation level by calling the SQLSetConnectOption function with the SQL_TXN_ISOLATION option and setting it explicitly.

You can also override the settings for Isolation Level, Dirty Data Treatment, and Effect of COMMIT fields by passing explicit keyword values to SQLDriverConnect at connect time.

- The ODBC driver can direct SQLDriverConnect to display the SQLDriverConnect dialog box, where the user can enter a password.
- If an ODBC application calls SQLConnect, it can pass an explicit password in the third string argument to SQLConnect. (The three string arguments to SQLConnect are interpreted as DSN, UID, and PWD, respectively.) When

SQLConnect is called, the driver looks in the registry to find the Default User Name and Default Password for the specified Data Source Name.

Note: Take care in selecting Isolation Level, Dirty Data Treatment, and Effect of COMMIT. Some ODBC applications are written to expect certain server behavior that is determined by these three settings. Such applications might behave unpredictably if these expectations are not met. Robust applications, however, usually adjust their expectations to the settings you specify in the data source, because ODBC provides a way for applications to query these settings.

Data source default parameters

SQL programmers who are knowledgeable about these settings might want to change the defaults described here (to affect all queries using the defined data source) or change the settings for individual queries.

Use the Model 204 ODBC Driver — Configure Data Source dialog box to change data source default parameters that can be accessed by ODBC or by other applications such as Visual Basic or Visual C++.

The specifications for data source parameters are stored in your system's registry under HKEY_LOCAL_MACHINE\SOFTWARE\ODBC\OCBC.INI.

Isolation Level

The Isolation Level parameter sets the default transaction isolation level used by the Model 204 32-bit ODBC driver to establish a connection to a Model 204 server. Isolation Level corresponds to the ISOLATION keyword.

Choose...	If you want to...
Dirty Read	See another user's uncommitted updates. In this case, Model 204 does not lock the selected records.
Cursor Stability (default)	Ensure that you only see updates that have already been committed. In this case, Model 204 locks selected records when a cursor is opened.
Serializable	Have complete control over a table by enforcing strict serializability. This option locks any table that you use during an SQL transaction and prevents other users from accessing those SQL tables. Rocket Software suggests that you use this option rarely, if at all.

Dirty Data Treatment

The Dirty Data Treatment parameter controls the default behavior of SQLFetch when the data contained in a Model 204 record cannot be converted to the SQL

data types mapped onto it (this is known as “dirty data”). Dirty Data Treatment corresponds to the DIRTYREC keyword.

Choose...	If you want to...
Server default (default)	Use the behavior specified in the SQLCNVER parameter of the CCAIN stream as the default for the Model 204 SQL Server being accessed. (For a description of the SQLCNVER parameter, see the *Rocket Model 204 SQL Connectivity Guide*.)
Error	Make SQLFetch return an error whenever it encounters dirty data.
Skip record	Skip any Model 204 record containing dirty data and try the next. If one or more records are skipped, issue a warning message.

Effect of COMMIT

The Effect of COMMIT parameter controls the behavior of the SQL_COMMIT option of the SQLTransact function.

Choose...	If you want SQL_COMMIT to...
Drop (default)	Close cursors and drop statements.
Preserve	Preserve open cursors and prepared statements.

Note: SQL_ROLLBACK always closes cursors and drops statements.

Debug tracing for ODBC

You can generate a debugging trace log that provides detailed information about your program run. Turn logging on if Technical Support requests it to research a problem. Turn logging off unless you are trying to document a problem. Do not activate logging during volume testing or production runs, because it slows Connect★ significantly.

Warning: The ODBC trace log file is cumulative and can grow to be very large. The log is best used for debugging specific programs with a small sample of data. If you use the log file, periodically erase it or edit out portions that are no longer needed.

Turning logging on and off

To turn logging on and off, do one of the following:

- use the CCACHECK program
- in the Connect Star for Model 204 program group, ODBC folder, Audit subfolder, select one of the following shortcuts:
  - Turn Logging Off
  - Turn Logging On

Note: Each connection writes its own set of log files with a date/time stamp as part of the file name into the Windows or WINNT folder.

CCACHECK program

The CCACHECK utility logs the following information to the file named LOGFILE.LOG in the same directory where you execute CCACHECK:

- Start and stop date for the program's execution
- Windows operating-system version
- Selected system information
- File and version information for files critical to 32-bit Connect★ ODBC
- Registry keys critical to 32-bit Connect★ ODBC
- Status of logging options

Creating your own shortcuts or overriding the defaults

Turn logging off by executing CCACHECK with the following parameters:

```
CCACHECK ~ALOGALL ~BOFF
```

Turn all the logs on by executing CCACHECK with the following parameters:

```
CCACHECK ~ALOGMSG ~B*IT ~CLOGRX ~DLOGTCP
```

You can turn on any combination of options, as required. Three logs are written to your Windows or WINNT folder:

- M204OD32.LOG
- M204RX32.LOG
- M204TC32.LOG

When you use the CCACHECK utility with the options listed in Table 3-1, the results are written to the specified logs.

Table 3-1. CCACHECK options

CCACHECK *Option*	Writes the…
`~ALOGMSG ~B*It`	M204OD32.LOG with Connect★ ODBC trace information. Use both the ~A and ~B options.
`~ALANGUSER ~BIBM1047, or ~BJAPAN, or ~BTURKISH, or ~BUS`	M204OD32.LOG with the code for the character set you want to use: • IBM1047: ASCII characters 0–256 • JAPAN: DBCS “Kanji” • TURKISH: ASCII characters 0–256, with Turkish syntax • US: ASCII characters 0–128
`~ATRANSTIME ~B*`	M204OD32.LOG with the connection time out value. Zero is the default.
`~CLOGRX`	M204RX32.LOG with communications information.
`~DLOGTCP`	M204TC32.LOG with socket level information.

LOGMSG message types

The LOGMSG message types are described in Table 3-2.

Table 3-2. LOGMSG message types

e =	Entry and exit from CLI or ODBC functions
x =	Diagnostic information
p =	Parameter values
b =	Bound buffer values
f =	Fetched values
t =	Internal trace
s =	SCFE (communications layer) request block
i =	Time intervals at entry and exit to every CLI/ODBC function and SCFE call

You can also use asterisks to indicate either “log all” or “log all except”, as shown in Table 3-3.

Table 3-3. Use of asterisks

LOGMSG=	Logs…
*	All message types
ext	Types e, x, and t
*ipf	All message types except i, p, and f

4

Connect★ for JDBC

Overview

Connect★ for JDBC supports Connect★ functionality to access Model 204. It includes all SQL and RCL statement syntax.

Related Rocket Model 204 documentation is available for download on the Rocket Customer Portal.

Connect★ for JDBC environment setup

Limitations and specifications

Limitations

A complete listing of the Connect★ for JDBC limitations is provided in the Connect Star for Model 204 program group under the JDBC > J204 API.

JDBC API specifications

JDBC API incorporates JDBC 2.5 with no extended functionality.

Platforms tested

The following platforms have been tested successfully for connectivity with Model 204 using Connect★ for JDBC:

- z/OS 1.5
- Red Hat Linux 8.0 (Kernel 2.4)
- Mandrake 9.2 & 10.0 (Kernel 2.6)
- Sun Solaris 2.8
- Windows NT, 2000, XP, Vista

These platforms are supported by Rocket Software.

Environment requirements

Before you install Connect★ for JDBC you must have the following installed in your working environment:

- Java2 Software Development Kit (J2SDK) Version 1.5 or greater.
- Model 204 Version 6.1.0 or greater with TCP/IP and SQL.

Removing a previous version of JDBC

1. Go to the Control Panel > Add/Remove Programs.
2. Click on Connect Star for JDBC and remove.
3. Go to the installation folder for the previous version of J204 and delete the j204.jar file.

Installing Connect★ for JDBC

1. Ensure that you have fulfilled the preinstallation requirements as described in “Preparing to install” on page 4.
2. Navigate to the FTP server site:

   ftp.rocketsoftware.com
3. Enter the userid and password provided by Rocket.
4. Navigate to the M204 > 7.5 > ConnectStar folder.
5. Open the JDBC folder.
6. Choose the folder for the driver that you want: 32bit or 64bit.
7. Download the files to your PC, in **binary** format.
8. Run the .exe file to install the application.

Connection parameters

To connect a J204 client application to a Model 204 server, the application must specify connection parameters in the URL and/or java.util.Properties object supplied as parameters to DriverManager.getConnection().

See J204 API for DriverManager and J204Driver classes. If a parameter is specified in both the URL and the properties object, the value in the properties object takes precedence.

The URL includes both keyword and positional parameters. Positional parameters must be entered in the specified order, while keyword parameters may appear in any order. All keyword parameters are optional.

Note: If you specify parameters in the properties object, you must enter the key entirely in uppercase or lowercase. Otherwise, the property will not be found. The case of parameters in the URL can be upper, lower, or mixed; it does not matter.

The URL has the following format:

Syntax

```
JDBC:J204:// {hostname} : {port} {/connection type}
     [/user/password]
     [/eoc=eoc character]
     [/cursors=commit option]
     [/dirty data=dirty data option]
     [/isolation level=isolation level option]
     [/trace=trace option]
```

Where

- *hostname* specifies the server's TCP/IP name, entered in dotted decimal or string format.

  Required; in the URL

- *port* specifies the port entered for the SERVPORT parameter in the DEFINE LINK command for the Model 204 server.

  Required; in the URL

- *connection type* is either SQL or RCL.

  Required; in either the URL or properties list

  Properties key: connection type

- *user* and *password* specify a valid Model 204 user name and password.

  Optional; no defaults

  Properties keys: user, password

- The *eoc character* for the `eoc` parameter specifies the end-of-line character for RCL connection.

Default character is a semicolon (;)

Properties key: eoc

- The *commit option* for the `cursors` parameter determines the effect of commit on cursors. Choose one of the following options:
  - DROP specifies to close the cursor and drop the statement.
  - PRESERVE specifies to preserve open cursors and prepared statements.

  Default is DROP

  Properties key: cursors

- The *dirty data option* for the `dirty data` parameter specifies the treatment of dirty data by a Fetch statement. Choose one of the following options:
  - DEFAULT uses the server default specified by the SQLCNVER parameter in the CCAIN stream; this is the default.
  - SKIP specifies to bypass a record containing dirty data and continue on to the next.
  - ERROR specifies to return an error message whenever dirty data is encountered.

  Default is DEFAULT

  Properties key: dirty data

- The *isolation level options* for the `isolation level` parameter are as follows:

Option	Specifies
RC, the default	Read Committed (cursor stability)
RU	Read Uncommitted (dirty read)
SR	Serializable

Properties key: isolation level

- The `trace` parameter controls logging. The *trace options* are as follows:

Option	Specifies
ALL	Write all trace messages to the log file
NONE	Write no trace messages

No log file is created if this parameter is omitted. For more information, see “Data source default parameters” on page 24.

Usage

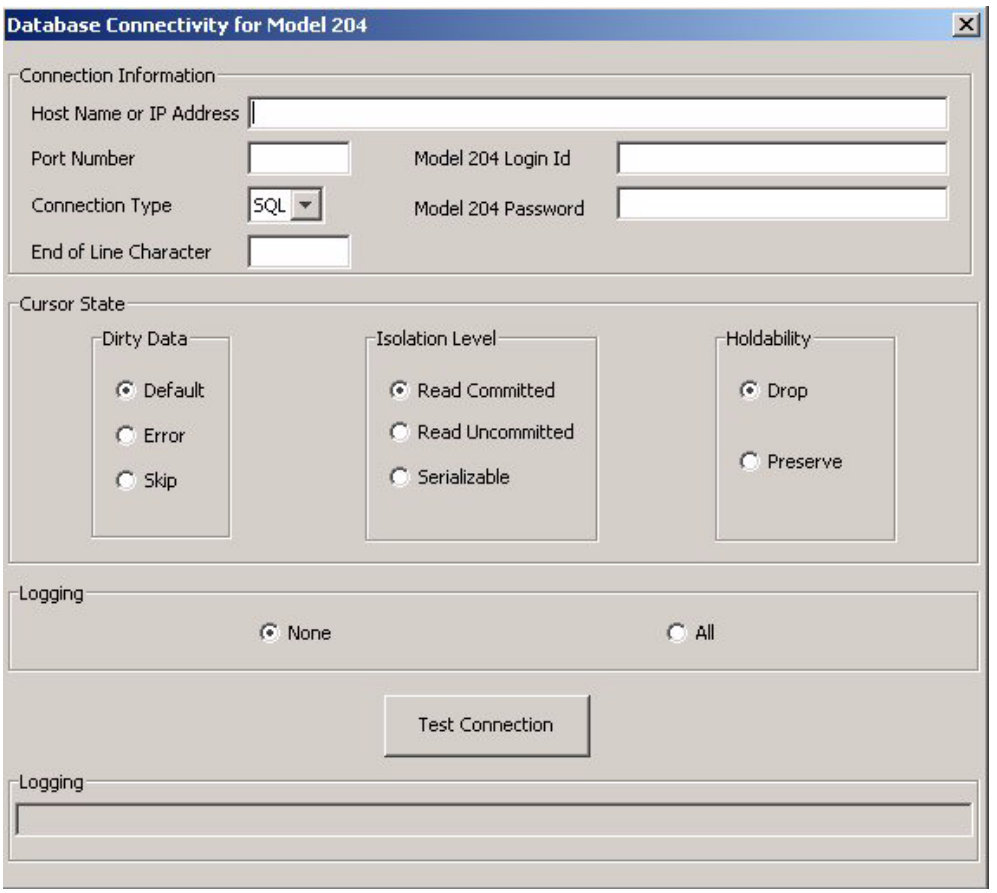
- You must specify the host name and port in the URL. You can specify the other parameters in the URL or in the properties object. If you specify a

parameter in both places, the value specified in the properties list takes precedence.

- Positional parameters must be entered in the specified order. *Hostname, port,* and *connection type* must be specified in the URL before other options.
- All keyword parameters are optional.
- When specifying parameters in a properties list, you must enter the key (property name) in either uppercase or lowercase characters, but not mixed case. If the key is in mixed case, the property cannot be found. When specifying parameters in the URL, you can enter uppercase, lowercase, or mixed case.

Verifying your JDBC connection

1. Navigate to Connect Star for Model 204 program group > JDBC > DatabaseConnectivity.
2. Complete the Connection Information screen and click the Test Connection button.



Connect★ for JDBC Compilation and Execution

Execution prerequisite

Before executing an SQL statement, check with your Connect★ administrator to make sure that either the demonstration database is installed and has been defined to the SQL catalog file, CCACAT, or you have other available tables defined in CCACAT. (See “SQL catalog population” on page 9.)

Online help

Connect* for JDBC includes HTML help files that provide complete documentation of the J204 JDBC driver. For detailed documentation, browse the index.HTML file in the \doc directory where the driver is installed.

RCL and JDBC

Java CallableStatements provide a means of calling stored procedures. Model 204 SQL does not support stored procedures; therefore, Connect★ for JDBC does not implement the CallableStatement interface.

Calling any CallableStatement method will result in an Unsupported Method SQLException.

An alternative to CallableStatement is the RCLStatement class, which allows execution of SOUL statements and procedures. You can use the following methods for RCL processing:

```
close()

createStatement()

execute()

executeQuery()

getResultSet()

getResultSetType()

getTransactionIsolation()

nativeSQL()
```

BLOB and CLOB support

Connect* for JDBC supports Model 204 BLOB and CLOB data types in SQL update and retrieval statements.

Use the following ResultSet methods to retrieve large object column data after SELECT statement execution.

```
BLOB
 byte[] getBytes()
 Object getObject()
 InputStream getBinaryStream()
CLOB
 byte[] getBytes()
 Object getObject()
 String getString()
 InputStream getAsciiStream()
 InputStream getBinaryStream()
 java.io.Reader getCharacterStream()
 InputStream getUnicodeStream()
```

In INSERT and UPDATE statements, use parameters for large object columns. First, create a prepared statement, then set the values of the parameters with the following PreparedStatement methods.

```
BLOB
 setBytes()
 setObject()
 setBinaryStream()
CLOB
 setBytes()
 setObject()
 setAsciiStream()
 setBinaryStream()
 setCharacterStream()
 setUnicodeStream()
```

j204.jar file

You must include the j204.jar file in the classpath for both compilation and execution. See your Java documentation for a complete list of options.

Note that the j204.jar file can be placed anywhere as long as the classpath points to its directory.

Compilation example

```
javac –classpath c:\jdbc\yourClasses;c:\jdbc\j204.jar
   yourProgram.java
```

Execution example

```
java -cp .;c:\jdbc\yourClasses;c:\jdbc\j204.jar yourProgram
```

Debug tracing for JDBC

To create a trace log file, use the TRACE=ALL parameter in the connection string. See “Connection parameters” on page 31.

J204Driver log file location

For applications run under Windows Vista and Windows 7, the driver places the log file in the user's home directory.

- Under Windows 7, the directory would be:

  C:\Users*user_name*

- Under Windows XP, the directory would be:

  C:\Documents and Settings*user_name*

For other operating systems, the log file is placed in the same directory as the j204.jar file.

Connection pooling

Connect* 7.5 for JDBC offers a "connection pooling" class to interface with other pooling infrastructures provided by pooling packages and web application servers. This new feature allows for a pool of connections that remain active and open during execution of an SQL and RCL (SOUL) statement.

Using connection pooling helps eliminate the overhead of creating initial connections from scratch or trying to manage each connection using the JDBC API.

Each Connection Pool can create a pool of active connections maintainable throughout the connection cycle of the pooling service.

Connect* 7.5 for JDBC has been tested with the JNDI, Apache DBCP™, BoneCP, and C3PO pooling packages.

Connection pooling example

Below is an example of pooling connections using the Apache DBCP™. There are other examples provided in the install directory of Connect* 7.5 for JDBC:

```
/**
* Example: Pooling Connection program using Apache DBCP™ (Database Connection
* Pooling) Package
*/

package com.cca.j204.tests;

import javax.sql.DataSource;
import java.sql.Connection;
import java.sql.Statement;
```

```
import java.sql.ResultSet;
import java.sql.SQLException;

// Using the Apache DBCP BasicDataSource.
// The example is using the following libraries:
//   commons-dbcp-1.x.jar
//   commons-pool-1.x.jar

import org.apache.commons.dbcp.BasicDataSource;

public class SimpleDBCPExample {

 public static void main(String[] args) {

  BasicDataSource basicDatasource = new BasicDataSource();
  basicDatasource.setDriverClassName("com.cca.j204.J204Driver");
  basicDatasource.setUsername("user");
  basicDatasource.setPassword("password");
  basicDatasource.setUrl("jdbc:j204://mainframe.domain.com:2403/rcl");
  basicDatasource.setInitialSize(5); // Set up 5 connections at initialization

  DataSource dataSource = basicDatasource;
  String statement = "VIEW ALL;";    // SOUL Statement

  // Show statistics before the connections are established
  System.out.println("Statistics before the connections are established.");
  showStatistics(dataSource);

  // Now, we can use JDBC DataSource as we normally would.
  Connection connectionHandle = null;
  Statement statementHandle = null;
  ResultSet results = null;

  try {
    System.out.println("Creating connection.");
    connectionHandle = dataSource.getConnection();//Start all 5 connections at
once.

  // Show statistics while the connections are established
  System.out.println("Statistics after initializing the connections.");
  showStatistics(dataSource);

  System.out.println("Creating statement.");
  statementHandle = connectionHandle.createStatement();

  System.out.println("Executing statement.");
  results = statementHandle.executeQuery(statement);

  System.out.println("Show Results:");
  int numcols = results.getMetaData().getColumnCount();
  while(results.next()) {
    for(int i=1;i<=numcols;i++) {
      System.out.print("\t" + results.getString(i));
    }
    System.out.println("");
  }
```

```
  // Show statistics while the connections are established
  System.out.println("Statistics while the connections are established.");
  showStatistics(dataSource);

  results.close();
  statementHandle.close();
  connectionHandle.close();

  // Show statistics after the connections have been closed
  System.out.println("Statistics after the connections have been closed.");
  showStatistics(dataSource);

  // Clean up and close all of the pooled connections
  basicDatasource.close();

  } catch(SQLException ex1) {
    System.out.println("SQLException in main(): " + ex1.getMessage());
  }

    }

  // Print out the statistics from the BasicDataSource class
  public static void showStatistics(DataSource ds) {
    BasicDataSource bds = (BasicDataSource) ds;
    System.out.println("Number of Active Connections: " + bds.getNumActive());
    System.out.println("Number of Idle Connections: " + bds.getNumIdle());
  }

}
```

The sample code generates this output:

```
Statistics before the connections are established.
Number of Active Connections: 0
Number of Idle Connections: 0

Creating connection.

Statistics after initializing the connections.
Number of Active Connections: 0
Number of Idle Connections: 5

Creating statement.
Executing statement.

Show Results:

  VERSION   7.4.0K      RELEASE OF MODEL 204
  CMSVERSN              VERSION OF THE CMS INTERFACE
  SYSOPT    X'AB'       SYSTEM OPTIONS
  SYSOPT2   X'00'       SYSTEM OPTIONS
  OPSYS     X'D209E8'   OPERATING SYSTEM
  SYSID     RS26        SMF SYSTEM IDENTIFICATION
  JOBNM     M204MWD     JOB NAME
```

```
STEPNM    ONLINE      STEP NAME
JOBSTEP   MPONLINE    JOB STEP NAME
JESID     J0919076    JES JOB ID
LOCATION              D204 LOCATION
LSERVPD   8176        LENGTH OF SCHEDULER PDL
NUSERS    432         NUMBER OF USERS
NSERVS    15          NUMBER OF SERVERS
SERVNSA   X'00000000' NON SWAPPABLE SERVER AREAS
SERVNSSZ  0           NON SWAPPABLE SERVER AREA SIZE
NJBUFF    36          NUMBER OF JOURNAL BUFFERS

.... (continue output)

Statistics while the connections are established.
Number of Active Connections: 1
Number of Idle Connections: 4

Statistics after the connections have been closed.
Number of Active Connections: 0
Number of Idle Connections: 5
```

5

Connect★ for .NET Framework

Overview

Connect★ for .NET Framework supports Connect★ functionality for Model 204. This includes all SQL and RCL statement syntax.

Related Rocket Model 204 documentation is available for download on the Rocket Customer Portal.

Connect★ for .NET Framework environment

.NET Framework conformance

Connect★ for .NET Framework is compliant with Microsoft .NET Framework 2.0.

Specifications and limitations

The Model204Client Help file is created during installation and can be found in the Connect Star for Model 204 program group under .NET.

Platforms tested

The following platforms have been tested successfully for connectivity with Connect★ for .NET Framework: Windows XP, Windows Vista, and Windows 7.

These platforms are supported by Rocket Software.

Environment requirements

Before you install Connect★ for .NET Framework you must have the following installed in your working environment:

- Model 204 Version 7 Release 4.0 or greater with TCP/IP and SQL
- Microsoft .NET Framework 2.0

Microsoft Visual Studio Integration

Connect* for .NET Framework supports integration of the Model 204 .NET Framework data provider with Microsoft Visual Studio™ 2005. Later versions are not yet supported.

Connect* for .NET Framework installation registers Model204Client as a Data Designer Extensibility (DDEX) provider. This enables you to use Model204Client with Visual Studio™ wizard and visual designer data tools to build Windows applications. Model 204 data objects appear as an object hierarchy in the Server Explorer window, and they can be dragged and dropped into various designer tools provided by the IDE.

For documentation on using the data provider in Visual Studio™, see the *Using Model204Client in Microsoft Visual Studio* topic in the CCA.Data.chm help file that is installed with Model204Client.

Uninstalling a previous version of Connect★ for .NET Framework

Go to Control Panel, Add/Remove Programs, click on Connect★ Model 204 for .NET Framework and remove.

Connect★ for .NET Framework installation

Installing Connect★ for .NET Framework

1. Ensure that you have fulfilled the preinstallation requirements as described in “Preparing to install” on page 4.
2. Navigate to the FTP server site:

   ftp.rocketsoftware.com
3. Enter the userid and password provided by Rocket.
4. Navigate to the M204 > 7.5 > ConnectStar folder.
5. Open the DotNet File folder.
6. Choose the folder for the driver that you want: 32bit or 64bit.

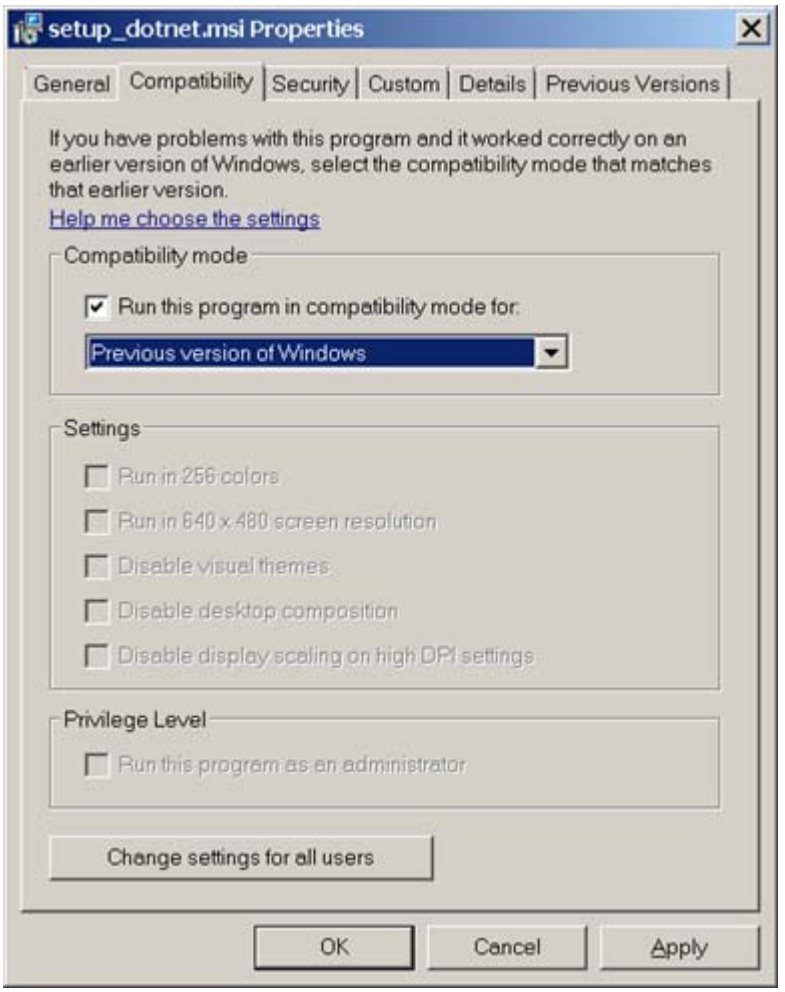
7. Download the files to your PC, in **binary** format.
8. Run the .exe file to install the application.

Installation under Microsoft Windows 7

You can run the Model 204 .NET installation using either setup.dotnet.exe or setup_dotnet.msi:

- To run the Model 204 .NET installation using setup.dotnet.exe, just click on setup.dotnet.exe.
- To run the Model 204 .NET installation using setup_dotnet.msi:

1. Right-click on setup_dotnet.msi in the installation directory and select Properties.
2. Go to the Compatibility tab (see below) and check “Run this program in compatibility mode”.
3. Press apply.
4. Click on setup_dotnet.msi to install.



Connection parameters

Consult the .NET Framework online help for the specific connection parameters.

For a complete discussion of the connection parameters see “Connection parameters” on page 31.

Documentation

The Model204Client Help file is created during installation and can be found in the Connect Star for Model 204 program group under .NET. The Help file provides complete documentation of the new Connect* for .NET Framework features.

Using Connect★ for .NET Framework

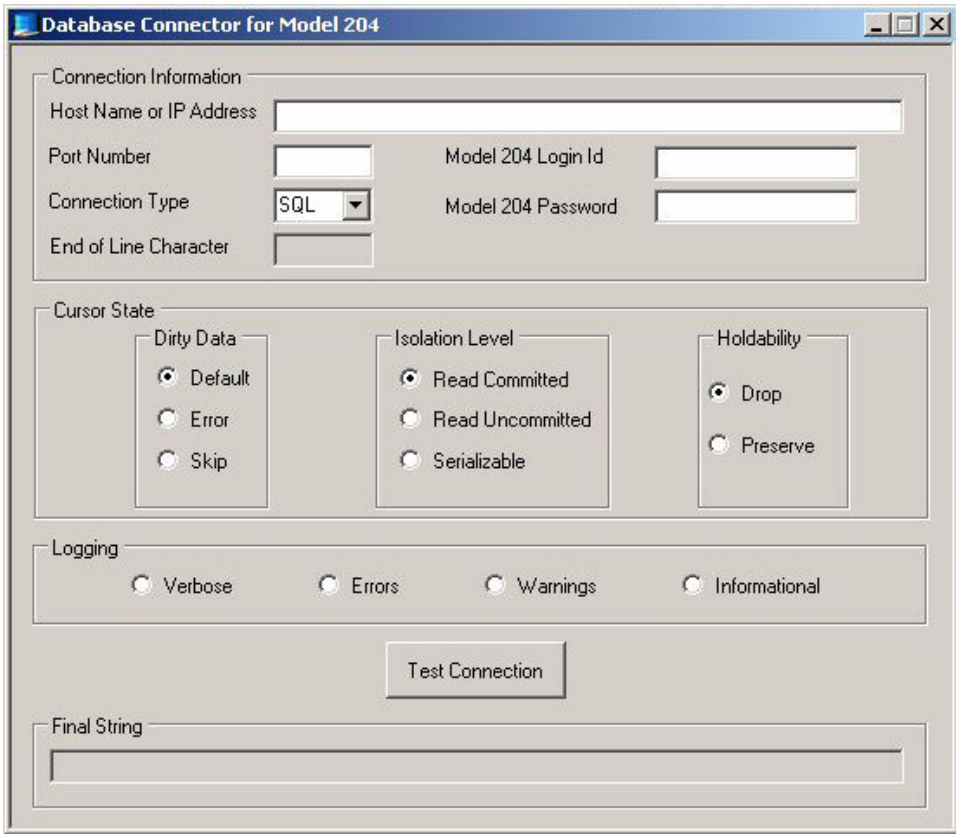
Prerequisite

Before executing an SQL statement, check with your Connect★ administrator to make sure that either the demonstration database is installed and has been defined to the SQL catalog file, CCACAT, or you have other available tables defined in CCACAT. (See “SQL catalog population” on page 9.)

Verifying the .NET Framework connection

Navigate to Programs.

1. Click on the Connect Star for Model 204 program group.
2. Then, click on .NET. > DatabaseConnectivity.
3. Complete the Connection Information screen and click the Test Connection button.



Debug trace for .NET Framework

To create a trace log file use the loglevel parameter in the .NET connection string.

Log file location under Windows

The driver places the Model204Client log file in the user's home directory.

- Under Windows 7, the directory would be:

  ```
  C:\Users\user_name
  ```

- Under Windows XP, the directory would be:

  ```
  C:\Documents and Settings\user_name
  ```

Large object (BLOB and CLOB) data type support

Connect* for .NET Framework now provides support for Model 204 BLOB and CLOB data types in SQL update and retrieval statements.

Use these DataReader methods to retrieve data from BLOB columns:

```
long GetBytes()
object GetValue()
```

Use these DataReader methods to retrieve data from CLOB columns:

```
long GetBytes()
long GetChars()
string GetString()
object GetValue()
```

In INSERT and UPDATE statements, use parameters for large object columns. First prepare the statement, then set the values of the parameters with the Parameter.Value property. The CCA.Data help file includes examples of storing and retrieving LOB columns.

Model204Client Connection Pooling

Connection pooling can reduce the number of times that you need to establish new connections to the server. When pooling is enabled and a new connection is opened, the pool manager obtains a server connection from an existing pool, if a suitable pool is available. The new connection must have exactly the same connection string as the pool.

Model204Client pools connections by default. You can disable pooling by setting pooling=false in the connection string. An application may have both pooled and non-pooled connections.

If no pool exists for a specified connection string, the pool manager creates a new pool and adds connection objects to the new pool until the minimum pool size requirement is satisfied. New connections with the same connection string are obtained from the pool, adding connection objects as necessary up to the maximum pool size. When connections are closed or disposed, they are added back to the pool. A connection pool is accessible until it is explicitly cleared or the client process terminates.

The static method `CdmDbConnection.ClearAllPools` disposes all connection pools for the provider, and `CdmDbConnection.ClearPool` clears the connection pool associated with a specific connection string. These methods sever idle connections on the server. Any connection in use at the time of the call is not closed until the application explicitly closes it. When that connection is subsequently closed, it is discarded instead of being returned to the pool.

Note: Transactions are specific to a connection and do not exist across connections.

Applications coded with connection pooling

An application should close or dispose connections when it finishes using them so that they are returned to the pool. Applications should also close active connections and clear all pools before terminating application processing. If this is not done, any connection open at termination will be severed by the server and receive the following error in the audit trail.

```
M204.2010: COMM ERROR STATUS, STATUSD = 53 1

M204.2012: REMOTE SQL SERVER COMMUNICATION ERROR RECEIVE
WAIT STATUS
```

Connection string properties

The following connection string properties pertain to connection pooling.

Pooling

Determines whether connection pooling is enabled for connections created with the same connection string.

True enables pooling; false disables it.

Type: boolean

Default: true

Max Pool Size

Maximum number of server connections in a pool.

If a new connection is opened and the maximum number of connections in the appropriate pool are already in use, the pool manager waits **Waittime** seconds for one to become available. If one does not become available within this time, it throws an exception.

Type: int

Default: 100

Min Pool Size

Minimum number of server connections in a pool.

If the minimum is larger than the maximum, the maximum value is used.

Type: int

Default: 0

Max Idle Time

Number of seconds a pooled connection can remain idle before the connection between the server and client is severed and the connection object is dropped from the pool.

When an application opens a new connection, the pool manager searches available connections in the associated pool and compares the time that data were last received from the server with the current time. It severs connections with a time span exceeding the value specified by **Max Idle Time**.

A value of zero indicates an infinite idle time.

Type: int

Default: 0

Waittime

The number of seconds the client will wait for a pooled connection to become available.

If no connection is available within this period of time, `CdmDbConnection.Open` will throw an exception.

Type: int

Default: 10

Properties summary

If `Pooling` is false, all other pooling parameters are ignored. If `Pooling` is true:

- A new connection is taken from an existing pool, if the connection strings match exactly.
- If the connection string does not match an existing pool, a new pool is created.
- When a pooled connection is closed or disposed, it remains logged into Model 204 with the user ID and password specified in the connection string.
- Model204Client severs a server connection when
  - it has been idle longer than `MaxIdleTime` seconds
  - `CdmDbConnection.CloseAllPools or CdmDbConnection.ClosePool` is executed for the pool.
- Each connection is independent of all others. Transactions exist only within the context of a single connection.

MaxIdleTime property setting

The purpose of the `MaxIdleTime` property is to reduce the number of idle connections in a pool. This provides a way for applications with large numbers of idle connections to limit the overuse of resources.

In setting the `MaxIdleTime` value, you should take account of the value of the `TIMEOUT` parameter on the server process definition associated with the connection. See the "DEFINE PROCESS command: SQL" page on the Model 204 wiki:

http://m204wiki.rocketsoftware.com/index.php/DEFINE_PROCESS_command:_SQL

If the server does not receive input for more than TIMEOUT seconds, then it abnormally terminates and unbinds the session, making the connection unusable. `MaxIdleTime` on the client, therefore, should be less than the TIMEOUT value on the server. If not, an application might get a pooled connection that has been severed by the server. The request will throw an exception:

```
System.Exception: Fatal communications error: a Negative
Response was received
```

Also, the server audit trail will show a timeout on that connection:

```
***  M204.1968: PROCESS TIMED OUT WAITING FOR COMPLETION
OF READ()
M204.2010: COMM ERROR STATUS, STATUSD = 53 2
***  M204.2010: COMM ERROR STATUS, STATUSD = 53 2
***  M204.2012: REMOTE SQL SERVER COMMUNICATION ERROR
RECEIVE WAIT STATUS
```

This will not happen if `TIMEOUT` is unlimited (the default). If `TIMEOUT` is unlimited, however, the server does not reclaim unused connections.

Note that `MaxIdleTime` pertains only to pooled connections that have been closed or disposed on the client, and `TIMEOUT` pertains to all existing server connections. If a client connection has not been closed but sends no data to the server for a period of time longer than `TIMEOUT`, the next request executed on that connection will fail.

Index

Numerics

A

C

D

E

I

J

L

M

O

P

R

S

T

U

V