Technical Note

Database Drivers Options

Overview

You can change the behaviour of Rep++ database drivers using options. These options can be set in two ways:

Method A: Using the configuration file.

Method B: Using the Connection.SetOption method.

This technical note describes the two methods along with the new options.

Database Drivers Options

Method A: using the configuration file

The sqld.ini configuration file contains information about your global configuration and the connections that you use. It is typically located in the *Rep++installationfolder*\bin\sdwin32\ or \bin\sdwin64\ folder, but can also be found in the \bin folder.

The configuration file is divided in different sections identified by a title in brackets. One section is named [CONSYST], which provides global information to the Rep++ and SQL Design applications. The connection sections, identified as [CONNECTION:*ConnectionName*] provide information pertaining specifically to *ConnectionName*. There can be as many as you have connections and you can have different settings for each. Note that these settings represent the default options for your connections, but they can be modified programmatically through the Connection.SetOption method (see Method B).

A [CONNECTION:*ConnectionName*] section can contain one or several options for the related database driver in the form of OptionName=*OptionValue*. The option name must be one of the valid options for the related database driver listed in the Options section below. For compatibility purposes, the database prefix 'xxx_' can be removed from the name.

Example:

```
[CONNECTION:MySqlServer]
SS_CursorMode=Static
MultiRowSize=128
TransMode=Auto
```

You can use the *Login* trace to see if the options are correctly set.

For more information about the configuration file, see Configuration Files in the Rep++ online help (in Visual Studio®), under Appendix Rep++.

Method B

You can use the Connection.SetOption method to set database driver options. It uses two parameters, the option name and the option value (see **Options** below). The full option name must be specified.

The method returns the following code:

0	(ERR_NO_ERR)	Parameter set.
12	(ERR_BAD_PARAM)	Bad parameter value for this option.
67	(ERR_NOT_SUPPORTED)	This option is not supported by this database driver.

Example:

```
app.DataConnection.SetOption("SS_CursorMode", "DefResSet");
```

Note that the SetOption method only overrides the values of the configuration file without modifying it.

Options

General

The following options can be used on all database drivers.

MultiRowSize	Sets the amount of memory allocated for multi-row fetching. The value is expressed in bytes. The value range must be between 0 and 16 384. The default value is 0, which lets the driver decide on the amount of memory to allocate. The buffer is allocated only if the multi-row fetching option is specified when the SQL command is compiled. Changing the value of this option has an impact on newly compiled SQL commands. Commands already compiled (or kept in the cache) are not modified.
NbMaxCursor	Specifies the maximum number of SQL commands that can be open at a given time for this connection. The value range must be between 4 and 1024. The default value is 32.

Oracle V8 Driver

ORA_NumStrict	Indicates whether the database specified by this connection supports transparent cast from CHAR to NUM type. If Oracle is used as a gateway to a foreign database (like DB/2) and this database does not support automatic conversion, this option must be set to True. The default value for this option is False. • Auto: Same as False. • True: Strict. • False: Not Strict.
ORA_NumWithComma	Indicates whether the driver converts the comma contained in the received numeric value to a period. Set this option to True if the language setting of the database uses the comma as the decimal separator. The default value of this option is False. • Auto: Same as False. • True: Converts the comma to a period. • False: Does not convert.

SQL Server OLEDB and ODBC Drivers

SS_CursorMode	Sets the type of cursor used to fetch data. SQL Server supports many options to optimize the way the data is returned from a request. You can use a default result set (firehose cursor) or one of the cursor flavors. The Rep++ driver for SQL Server (using OLEDB) supports the following options: • Auto: Use a fast forward cursor if connected to a SQL Server 2000 or older database. A default result set is used if connected to a SQL Server 2005 or newer database. This is the default option. • Static: Use a static cursor. • FastFoward: Use a fast forward cursor. • DefResSet: Use a default result set. MARS (see option below) will be used if

	connected to SQL Server 2005 or later. In all cases, the driver uses a default result set if the SQL command processed is a stored procedure or a SELECT accessing BLOB values. The fastest method to access data is normally achieved using a default result set. However, this method also has severe drawbacks with versions prior to SQL Server 2005, for which only one statement can be executed at a time for each SQL Server connection. If you use a SELECT statement, all rows must be fetched before you can execute another SQL statement of any kind. If you try to execute a second SQL statement while another one is still active, the OLEDB driver silently creates a second connection. When this second statement terminates, the connection is automatically closed. Note that these automatically opened connections are not pooled. This behaviour can cause scalability problems. For SQL Server 2005 or later, MARS (Multiple Active Result Sets) mode can be used (see option below). Using multi-row fetching can improve dramatically the performance when fetching a large result set with a static or fast forward cursor. In this case, specifying a large MultiRowSize value will have a significant impact on the performance. Using a default result set, the multi-row fetching parameter has no impact on performance, or might have a negative impact if a large value is specified for the MultiRowSize option. This is not a big surprise knowing that SQL Server already caches all the result sets on the client side. In this case, we are doing double caching, which increases overhead without benefits. For more information about these different options, please refer to the SQL Server documentation.
SS_Mars	Indicates whether the MARS (Multiple Active Result Sets) mode is enabled. Used with SQL Server 2005 and later versions. This option allows multiple active result sets at a time. With this option on, the driver does not silently open new connections when more than one statement needs to be executed at the same time. • Auto: Same as On (default). • On: MARS is enabled. • Off: MARS is disabled.
SS_SPDirectExec	Indicates whether SQL commands are prepared (validated) before execution. This option only applies to stored procedures. • True: The SQL command is executed directly without validation. • False: The SQL command is validated before execution (default).
SS_StartTrans	Starts a new transaction if not already started. Only for OLEDB.
SS_TransMode	Indicates when to start a new transaction. By default, a transaction is automatically started when a non-SELECT statement is processed. • Auto: Starts a transaction when a non-SELECT statement is executed, if a transaction is not already started. • All: Starts a transaction when a SQL statement is executed, if a transaction is not already started.

SQLite

DBFileName	Database file name.
TmpTable	Indicates where the temporary tables are kept. *File*: Temporary tables are kept in a file. *Memory*: Temporary tables are kept in memory.

CacheSize	Cache size, in bytes.