1010data ODBC Driver User’s Guide
Introduction.................................................................................................. 3

Installing the ODBC Driver........................................................................ 4
  Installing on Linux.......................................................................................... 5

Configuration Options.................................................................................. 6
  Windows-Specific Instructions........................................................................ 6
  Linux-Specific Instructions............................................................................. 6
  Connection...................................................................................................... 7
  SQL.................................................................................................................. 10
  Driver........................................................................................................... 12
  Types............................................................................................................. 14
  Logging.......................................................................................................... 14

Setting Up a New Connection..................................................................... 17
  Creating a New Connection.......................................................................... 17

Best Practices.............................................................................................. 20
  Using 1010data Macro Language in 1010SQL........................................... 20
    Escaping within the SELECT clause......................................................... 20
    Escaping within the WHERE clause......................................................... 20
    Escaping within the FROM clause............................................................ 20
  Stored Procedures....................................................................................... 21
    Using a Quick Query as a Stored Procedure........................................... 21
    Calling a 1010data Stored Procedure from SQL....................................... 22

Troubleshooting and Support..................................................................... 23
  Upgrading the ODBC Driver....................................................................... 24
  Upgrading the ODBC Driver on *Nix........................................................ 24
  Known Issues and Workarounds................................................................. 24
  Requesting UTF-8 Data............................................................................... 24
The 1010data ODBC driver is a software utility that serves two primary functions. First, it provides a standard interface so that applications can connect to 1010data directly. Second, it allows the 1010data system to understand SQL. This functionality is very useful, but some care should be taken when configuring the ODBC driver to maintain optimal system speed and resource management. Doing so will prevent technical issues and provide the best possible user experience. When you build your queries, please refer to Best Practices on page 20 for tips on writing efficient queries.

**Note:** The 1010data ODBC driver conforms to the ODBC 3.0 specification.
Installing the ODBC Driver

The 1010data ODBC driver is only available for clients who have purchased API access along with their 1010data account. If you don't have API access, or aren't sure if you do, contact your 1010data sales representative.

With the newest version of the 1010data ODBC driver, both the 32-bit and 64-bit versions install simultaneously. Since both versions are installed, 64-bit applications will automatically use the 64-bit version of the driver, and 32-bit applications will use the 32-bit version of the driver. However, you must configure separate DSN profiles for each version.

For Windows: Once you download the correct version of the ODBC driver installer, simply double-click on the package and follow the on-screen instructions.

In order to install the ODBC driver on Windows you must have permission to write to both of the following folders:

- C:\Windows\System32
- C:\Windows\SysWOW64

The installation directory selected during the installation process contains files such as documents, licenses and the uninstaller for the driver. It does not contain the actual binary files. If you do not have write permission to both the directories listed above the ODBC driver installation will fail.

During the Windows installation, there are two options now available as shown in the screenshot below:

If you connect to 1010data via a proxy server, enter the proxy information in the format given in the installation wizard (pictured above). If you don't use a proxy, leave this box blank.

If you plan to use Stored Procedures with 1010data, check the I plan on using Store Procedures check box (circled in red above). If you're not sure if you're going to use Stored Procedures, leave this blank. If you don't check this box during installation, you may still used stored procedures by using the DSN configuration outlined in the Configuration Options section of this document.

If you call the 1010data ODBC installer from the Windows command line with the /S command it will trigger a "silent install." If you call the uninstaller with the /S switch it will trigger a "silent uninstall."

Note: A "Silent Installation" means that the installer will not ask you any questions during installation. Instead, it will simply install the driver with the default options.
Installing on Linux

For Linux: Make sure that UnixODBC is installed on your system and then run the 1010data ODBC Linux install script. If you need a more customized setup, please refer to the README file included with the installation package.
Configuration Options

Windows-Specific Instructions

The ODBC driver installation installs both a 32-bit and 64-bit version of the driver. Each version maintains DSN profile that must be configured separately. On a 64-bit Windows system, both drivers will be configurable. If you use a 32-bit Windows system, the 32-bit version of the driver is the only version available for configuration.

To open the **SQL1010 ODBC DSN Configuration** dialog, complete the following steps:

1. Open the **ODBC Data Source Administrator** dialog by clicking the **Start Menu > All Programs > 1010data > 1010odbc**.
   - To configure the 32-bit version of the driver, click the **ODBC 32-bit** option
   - To configure the 64-bit version of the driver, click the **ODBC Native** option

   **Note:** If you use a 32-bit Windows system, the only option will be: **ODBC Native**. Selecting this option will allow you to configure the DSN profile for the 32-bit version of the driver.

2. Click the **System DSN** tab.
3. Click the system DSN labeled: **www2.1010data.com**.
4. Click the **Configure...** button on the right-hand side of the dialog.

You should now see a configuration dialog with 5 tabs, as follows: **Connection, SQL, Driver, Types, and Logging**.

Finally, if you’re not certain of the driver you’re configuring, you can verify the version as shown in the screenshot below:

The number after the "." character tells you which version you’re configuring. Above is the 32-bit configuration dialog. A **64** will follow the "." character when configuring the 64-bit version.

Linux-Specific Instructions

Linux configuration for the 1010data ODBC driver is managed by the config file located in **{$HOME}/.1010data/config**. If it needs to be moved, the path to the config file may be specified in the **ODBCENTEN** environment variable. Most settings described below are configurable on Linux. For information on valid key values, reference the sample config file.
For the settings tables below, the left-most column contains the setting names for Windows on top and the configuration keys for Linux on the bottom.

**A Note for Developers:** If you are building a front-end application, the Linux configuration keys and associated values can be used for your connection string when connecting to the 1010data ODBC driver.

### Connection

The **Connection** tab contains the majority of the settings for the ODBC driver.

![1010data ODBC DSN Configuration](image)

The following table describes the function of each setting, as well as recommended use-cases:

**Table 1: 1010data ODBC driver: Connection tab**

<table>
<thead>
<tr>
<th>ODBC Driver Setting/ Linux Key</th>
<th>Default Value</th>
<th>Brief Description</th>
<th>Use Case</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DSN</strong></td>
<td><code>www2.1010data.com</code></td>
<td>A unique name for the ODBC driver connection.</td>
<td>Keep the default value unless there is a specific reason you need to change it.</td>
</tr>
<tr>
<td>{DSN}</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Host</strong></td>
<td><code>www2.1010data.com</code></td>
<td>The domain name of the server/service your computer connects to when using an ODBC connection.</td>
<td>Keep the default value, unless you need to use a specific version of 1010data. If you do need to use a specific version, specify it here (i.e., <code>www2.1010data.com/cgi-bin/{VERSION}/gw.k</code>).</td>
</tr>
<tr>
<td>{Host}</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>User ID</strong></td>
<td><code>blank</code></td>
<td>The 1010data user name of the person/computer connecting to the database.</td>
<td>If the application you're using sends authentication information, this field will be overwritten. In such a case, it doesn't matter if this field is populated or not.</td>
</tr>
<tr>
<td>{UID}</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ODBC Driver Setting/ Linux Key</td>
<td>Default Value</td>
<td>Brief Description</td>
<td>Use Case</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>---------------</td>
<td>------------------</td>
<td>---------</td>
</tr>
<tr>
<td><strong>Password</strong></td>
<td>{PWD} blank</td>
<td>The password for the 1010data account used in this connection.</td>
<td>If the application you're using sends authentication information, this field will be overwritten. In such a case, it doesn't matter if this field is populated or not.</td>
</tr>
<tr>
<td><strong>Port</strong></td>
<td>{Port} 443</td>
<td>The TCP/IP port on which the 1010data servers establish ODBC connections</td>
<td>Keep the default, or change to '80' if you don't want to send encrypted information. For instance, if you are connecting to 1010data via a secure VPN connection, you can safely change the port to 80.</td>
</tr>
<tr>
<td><strong>Kill prior session</strong></td>
<td>{KillP} No</td>
<td>If there is an active 1010data session when you log in, this option controls the behavior. If <strong>No</strong>, an error will be returned. If <strong>Yes</strong>, the session will be terminated and a new session will be established. If <strong>Possess</strong>, the existing session will be taken over by the ODBC interaction.</td>
<td></td>
</tr>
<tr>
<td><strong>Reconnect?</strong></td>
<td>{Persist} Unchecked</td>
<td>If the current session times out, attempt to reconnect to the 1010data session.</td>
<td>Keep the default unless you have a specific reason to change it. Binary data sends and receives more quickly than ASCII data.</td>
</tr>
<tr>
<td><strong>Binary mode</strong></td>
<td>{BinaryP} Checked</td>
<td>Sends all data in binary form instead of ASCII.</td>
<td>Keep the default unless you have a specific reason not to send compressed data.</td>
</tr>
<tr>
<td><strong>Compression</strong></td>
<td>{Compression} Checked</td>
<td>Send data using compression algorithms to improve send/receive times.</td>
<td>Keep the default unless you have a specific reason not to send compressed data.</td>
</tr>
<tr>
<td><strong>Treat root dirs as schemas</strong></td>
<td>{Schematize} Unchecked</td>
<td>Tells the ODBC driver to treat root directories in the 1010data system as schemas would be treated in a relational</td>
<td>Use this option if your application requires schemas to function.</td>
</tr>
<tr>
<td>ODBC Driver Setting/ Linux Key</td>
<td>Default Value</td>
<td>Brief Description</td>
<td>Use Case</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>---------------</td>
<td>-------------------</td>
<td>----------</td>
</tr>
<tr>
<td><strong>database. In an SQL database, schemas contain tables, while in 1010data, directories contain tables.</strong> Download Window Size</td>
<td>1000000</td>
<td>Sets the default number of records or rows downloaded to the local system each time the local system receives information.</td>
<td>If you are working in a table or set of tables that all share the same root directory, you can provide the name of that root directory here so that it doesn't need to be typed every time.</td>
</tr>
<tr>
<td><strong>{Window}</strong> Root Folder (Optional)</td>
<td>Blank</td>
<td>Provides a root folder/directory where all subsequent folders/tables will be assumed to be located. <strong>Note:</strong> The Mangle pathnames checkbox must be checked for this option to work. <strong>Note:</strong> The root folder name must contain an '@' symbol before the name. See the screenshot preceding this table for an example.</td>
<td>If the programmatically generated SQL code produced by your applications uses underscores as folder delimiters, you should check this box. <strong>Note:</strong> Using this feature can degrade client performance when large numbers of objects are visible in the database.</td>
</tr>
<tr>
<td><strong>{Database}</strong> Mangle pathnames</td>
<td>Unchecked</td>
<td>In SQL, both the underscore (_) character, and the period (.) character are valid as folder delimiters. In 1010data, the period (.) character is used for this purpose. This setting performs a “fuzzy” search for underscore delimiters in an SQL folder path and replaces underscores identified as delimiters with periods. <strong>Note:</strong> This setting is also required for the</td>
<td></td>
</tr>
<tr>
<td>ODBC Driver Setting/ Linux Key</td>
<td>Default Value</td>
<td>Brief Description</td>
<td>Use Case</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>---------------</td>
<td>------------------</td>
<td>---------</td>
</tr>
<tr>
<td>Group Name</td>
<td>Blank</td>
<td>This value specifies the name of the SAM pool group you use for shared access management.</td>
<td></td>
</tr>
<tr>
<td>{GRP}</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Retry Get UID if system is busy</td>
<td>Unchecked</td>
<td>If all the SAM pool resources for your group (i.e., the preset number of user IDs available) are in use, the system will attempt to reconnect until a connection is made.</td>
<td></td>
</tr>
<tr>
<td>{LoopGetUID}</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Clear cache if workspace is greater than n MB</td>
<td>Blank</td>
<td>For SAM pools, a UID's resources remain allocated even after the session has ended.</td>
<td></td>
</tr>
<tr>
<td>{ClearCache}</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>{XFF}</td>
<td>Empty</td>
<td>This value stores the IP address given from a user machine to a proxy application.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

If you're using a proxy to connect, prepend your host URL with the proxy credentials in square brackets []. For example:

```
[http://PROXYUSERNAME:PROXYUSERPASSWORD@www.corporateproxyserver.com:8080]
https://www2.1010data.com
```

This does not support auto-proxy.

**SQL**

The **SQL** tab contains settings specific to how the ODBC driver translates SQL logic to the 1010data Insights Platform.
Table 2: 1010data ODBC driver: SQL tab

<table>
<thead>
<tr>
<th>ODBC Driver Setting/ Linux Keys</th>
<th>Default Value</th>
<th>Brief Description</th>
<th>Use Case</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use three-value Boolean logic (slower)</td>
<td>Unchecked</td>
<td>Evaluate boolean expressions using SQL conventions as opposed to 1010data Macro Language conventions.</td>
<td>You should check this box if you want N/A values to never equal anything.</td>
</tr>
<tr>
<td>...only if columns are nullable</td>
<td>Unchecked</td>
<td>Apply the above option only to columns that allow null values.</td>
<td>See above.</td>
</tr>
<tr>
<td>Use metadata in SQL translator</td>
<td>Unchecked</td>
<td>Attempt to optimize queries with table attributes.</td>
<td>Use this option if you want to use stored procedures with your application.</td>
</tr>
<tr>
<td>...for join-order optimization</td>
<td>Unchecked</td>
<td>Utilize primary key information to optimize 1010data link order from SQL JOINs.</td>
<td></td>
</tr>
<tr>
<td>Allow scalar columns in summary query</td>
<td>Unchecked</td>
<td>Use to enable the use of scalar columns in queries sent to 1010data.</td>
<td></td>
</tr>
<tr>
<td>ODBC Driver Setting/Linux Keys</td>
<td>Default Value</td>
<td>Brief Description</td>
<td>Use Case</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>---------------</td>
<td>----------------------------------------------------------------------------------</td>
<td>--------------------------------------------------------------------------</td>
</tr>
<tr>
<td>...and group by them</td>
<td>Unchecked</td>
<td>Allows you to use the above option as a break column.</td>
<td></td>
</tr>
<tr>
<td>{GroupByThem}</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sequential conjunction</td>
<td>Unchecked</td>
<td></td>
<td>Note: Do not change this setting unless expressly told to do so by 1010data support.</td>
</tr>
<tr>
<td>{SequentialAnd}</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Allow quotes around stored procedures</td>
<td>Unchecked</td>
<td>Use double quotes (&quot;) with stored procedure FROM clauses.</td>
<td>Use quotes for stored procedures that contain parentheses SELECT * FROM &quot;my.stored.procedure(VALUE1, VALUE2, ... )&quot;.</td>
</tr>
<tr>
<td>Ignore trailing underscore warnings</td>
<td>Unchecked</td>
<td>Ignores trailing underscores in column names.</td>
<td>Tableau produces SQL that adds trailing underscores to the column names and aliases it produces. These names are not allowed in the platform.</td>
</tr>
</tbody>
</table>

**Driver**

The **Driver** tab contains settings for output scenarios.
### Table 3: 1010data ODBC driver: Driver tab

<table>
<thead>
<tr>
<th>ODBC Driver Setting/ Linux Keys</th>
<th>Default Value</th>
<th>Short Description</th>
<th>Use Case</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Excel(tm) compatibility kludges</strong></td>
<td>Unchecked</td>
<td>Required for MS Excel to utilize the ODBC driver for inputting into Excel from 1010data.</td>
<td></td>
</tr>
<tr>
<td>{ExcelKludge}</td>
<td></td>
<td><strong>Note:</strong> The {ExcelKludge} key does not exist in the Linux config file, but is valid for connection strings in Windows.</td>
<td></td>
</tr>
<tr>
<td><strong>SQL Cancel behavior</strong></td>
<td>Report Unimplemented</td>
<td>Changes the way the ODBC driver reports SQL Cancel functionality.</td>
<td>Change this option if you're connecting from MS SQL server.</td>
</tr>
<tr>
<td>{Cancel}</td>
<td></td>
<td><strong>Note:</strong> Changes the way the ODBC driver reports SQL Cancel functionality.</td>
<td></td>
</tr>
<tr>
<td><strong>On error clear cache and retry</strong></td>
<td>Unchecked</td>
<td>Resubmits a query after clearing the server memory cache.</td>
<td>If you've encountered an error this command will clear your cache and then resubmit whatever transaction encountered the error exactly once.</td>
</tr>
<tr>
<td>{AlwaysClear}</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>ADO SQL Columns underscore kludge</strong></td>
<td>Unchecked</td>
<td>The ADO standard doesn't support underscores ('<em>') in table names. This allows the user to use a &quot;*&quot; character instead of an '</em>' character when calling the SQLTables function.</td>
<td></td>
</tr>
<tr>
<td>{AdoKludge}</td>
<td></td>
<td><strong>Note:</strong> The {AdoKludge} key does not exist in the Linux config file, but is valid for connection strings in Windows.</td>
<td></td>
</tr>
<tr>
<td><strong>Report errors in SQLSetStmtAttr</strong></td>
<td>Unchecked</td>
<td>When this box is checked, attempts to set SQL statement attributes will return the following error message: SQL_ERROR, if the attribute cannot be set. If the box is left unchecked, any attempt to set SQL statement attributes will return the message, SQL_SUCCESS, whether the operation</td>
<td>This option was created to accommodate the behavior of some tools and the way they connect to the 1010data ODBC driver. Currently, Tableau is the only known tool that requires this option.</td>
</tr>
<tr>
<td>ODBC Driver Setting/ Linux Keys</td>
<td>Default Value</td>
<td>Short Description</td>
<td>Use Case</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>---------------</td>
<td>-------------------</td>
<td>---------</td>
</tr>
<tr>
<td>String width to report {StrWidth}</td>
<td>255</td>
<td>was performed successfully or not.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>This is the width that the 1010data ODBC driver reports for string columns.</td>
<td>If your data has very wide string columns you may need to increase this value.</td>
</tr>
</tbody>
</table>

## Types

The **Types** tab contains settings for handling Date, Time and Timestamp formats.

![SQL1010 ODBC DSN Configuration](image)

### Table 4: 1010data ODBC driver: Types tab

<table>
<thead>
<tr>
<th>ODBC Driver Setting/ Linux Keys</th>
<th>Default value</th>
<th>Short description</th>
<th>Use case</th>
</tr>
</thead>
<tbody>
<tr>
<td>HHMMSS</td>
<td>SQL_TIME</td>
<td>Time format</td>
<td>N/A</td>
</tr>
<tr>
<td>{MapHHMMSS}</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Timestamp</td>
<td>SQL_TIMESTAMP</td>
<td>Timestamp format</td>
<td>N/A</td>
</tr>
<tr>
<td>MapTimestamp</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>YYYYMMDD</td>
<td>SQL_DATE</td>
<td>Date format</td>
<td>N/A</td>
</tr>
<tr>
<td>{MapYYYYMMDD}</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

## Logging

The **Logging** tab contains settings that dictate how the system records actions and errors.
<table>
<thead>
<tr>
<th>ODBC Driver Setting/Linux Keys</th>
<th>Default value</th>
<th>Short description</th>
<th>Use case</th>
</tr>
</thead>
<tbody>
<tr>
<td>Log to</td>
<td>Unchecked/Blank</td>
<td>If the system should create log files and if so, the file path to the current log file.</td>
<td>You should check this box to keep the number of files in the log directory to a minimum. <strong>Note:</strong> When you experience a problem, you should uncheck this box to generate a clean log file that will only contain information pertaining to the error.</td>
</tr>
<tr>
<td>Append</td>
<td>Unchecked</td>
<td>Append new logs to the existing log file, as opposed to creating a new log file.</td>
<td></td>
</tr>
<tr>
<td>Log-level drop-down menu</td>
<td>Basic</td>
<td>Each option records a different level of log data.</td>
<td>For all troubleshooting and support issues, this option should be set to: <strong>Debug+XML</strong></td>
</tr>
<tr>
<td>Import Options</td>
<td>Blank</td>
<td>The file path to a settings file.</td>
<td>If you have previously installed and configured the ODBC driver and saved your settings, you can import those settings from the saved file here.</td>
</tr>
<tr>
<td>ODBC Driver Setting/Linux Keys</td>
<td>Default value</td>
<td>Short description</td>
<td>Use case</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>--------------</td>
<td>------------------</td>
<td>---------</td>
</tr>
<tr>
<td>Export Options</td>
<td>Blank</td>
<td>The file path where a configuration file is stored.</td>
<td>If you have properly configured your ODBC driver, you can output the settings here so you can import them and reapply them in the future.</td>
</tr>
</tbody>
</table>
Setting Up a New Connection

Once the 1010data ODBC driver is installed it is recommended to create a different connection for every use-case you may encounter. This allows end-users to maintain separate configurations for each client-application, and generally helps keep things organized. This section walks through the process of creating a new DSN connection with the recommended settings.

Creating a New Connection

To create a new connection, click the **Start** button, then click **All Programs > 1010data > 1010dataodbc**.

If you have the most current version of the 1010data ODBC driver, then both the 32-bit and 64-bit versions are installed and accessible here:

Before moving forward there's an important detail to keep in mind. Remember that it is a client application that will use the 1010data ODBC driver to connect to a 1010data data source. So you must know the "bitness" of the client application. Depending on many factors, the client application could be a 32-bit or 64-bit application. Make sure you know the bitness of the client, and then create a DSN connection with the corresponding ODBC driver (i.e., use **Configure 32-bit ODBC** for a 32-bit client application and **Configure Native ODBC** for a 64-bit application). For the rest of this section, it will be assumed the connection is being configured for a 64-bit application. However, the configuration options are identical.

After selecting the appropriate driver for the client application the **ODBC Data Source Administrator** will launch. Click the **System DSN** tab, then click the **Add** button, as shown below:

After clicking the **Add** button, the **Create New Data Source** dialog will open. Make sure you select the 1010dataODBC driver option available in this dialog, then click the **Finish** button, as shown below:
After clicking the **Finish** button the **SQL1010 ODBC DSN Connection** configuration dialog will automatically appear. There are several settings on the **Connection** tab that should be set unless you have a specific reason not to. The list below provides the configuration details that should be set when the configuration is complete:

- Complete the **DSN** field by giving the new connection a descriptive name
- Make sure the **Compression** checkbox is checked
- Make sure the **Reconnect?** checkbox is checked
- Make sure the **Binary mode** checkbox is checked

When the configuration is complete the **Connection** tab should look like the screenshot below (with your own descriptive connection name, of course):

When you've finished the configuration, click the **OK** button. The newly created connection will appear on the **System DSN** tab of the **ODBC Data Source Administrator** dialog, as shown below:
There are many other configuration options you may want to set depending on the specific use of the connection you’re creating. For full details on every available configuration options, see the Configuration Options section of this guide.
Best Practices

When working with SQL queries and the 1010data system, there are some important conventions to keep in mind. This section contains several helpful pieces of information you should consider when writing any 1010SQL query.

Using 1010data Macro Language in 1010SQL

The syntax of SQL expressions differs greatly from that of the 1010data Macro Language. Furthermore, the 1010data Macro Language provides a huge number of functions that simply aren't available in SQL. In order to retain the full power of the 1010data system, and in the interest of writing queries that will run optimally on the system, you can use the standard ODBC escape sequence ({} ) to incorporate 1010data Macro Language expressions in your query.

When you are building a 1010SQL query, you can insert a 1010data Macro Language expression at within FROM, WHERE, and SELECT clauses, by placing the expression inside curly braces {}. This is the standard syntax for escape sequences in ODBC. Value expressions may use any 1010data function available in the Macro Language.

When you build 1010SQL queries, try to use as many escape sequences as possible to do the heavy lifting and calculations for your query. Not only is the Macro Language a more powerful language for transforming data and performing calculations, it is also the native language of the 1010data system. Therefore, your queries will run faster and have less chance of encountering issues when you incorporate as much Macro Language code as possible.

Escaping within the SELECT clause

You can escape 1010data Macro Language code within a SELECT clause.

As an example, you can make use of the 1010data g_median group function (which returns the median value for a column within a group):

```
SELECT lahmanid, year, {hr-g_median(lahmanid;;hr)} AS homeRunDifference
FROM pub.demo.baseball.batting
```

The escaped 1010data Macro Language expression uses an alias, AS homeRunDifference. The alias is used to reference the Macro Language code, and it is required.

Notice here that the syntax of the escape sequence {hr-g_median(lahmanid;;hr)} is that of the 1010data Macro Language, not SQL. One example of how the two syntaxes differ is that in the 1010data Macro Language expression, a semicolon is used to separate the function arguments.

Escaping within the WHERE clause

This example shows how to escape to 1010data Macro Language code in a WHERE clause.

Examine the following query:

```
SELECT * FROM pub.demo.weather.stations
WHERE {sm(name;'S*')|sm(name;'A*')&sm(name;'*E')};
```

This query returns 9 rows, namely those stations whose name begins with either S or A, and end with E.

Escaping within the FROM clause

An even more powerful escape tool provided in 1010SQL is the macro escape mechanism. With macro escapes, any series of 1010data operations, no matter how large or complex, can be included in a 1010SQL statement.
The way this mechanism is implemented is fairly straightforward. The escape clause is simply appended to a table identifier in the `FROM` clause as a modifier, using the syntax:

```
table:{ops}
```

When the `FROM` clause is resolved, the table `table` is opened, and the 1010data operations (belonging to the Macro Language) specified by the `ops` are applied to the table before any further processing by 1010SQL. Below is an example:

```
SELECT *
FROM pub.demo.weather.stations:{<sort col="elev" dir="up"/>};
```

The effect the sample above is to sort the table by elevation and return all rows. The result is exactly the same as the SQL subquery below:

```
SELECT *
FROM (SELECT * FROM pub.demo.weather.stations ORDER BY elev);
```

The difference here is that the first `FROM` statement, which incorporates a 1010data Macro Language `<sort>` operation, is much faster than the SQL statement on the 1010data system.

It is important to note the differences in syntax between the two languages so you can build more efficient queries. If you need more information on the Macro Language and how to build 1010data expressions, take a look at the 1010data Reference Manual.

### Stored Procedures

Working with 1010data via the ODBC driver also allows you to use Quick Queries you have built as a stored procedure in 1010SQL.

As you may know if you're familiar with the standard 1010data interface, Quick Queries are 1010data Macro Language queries written in block statements that can be saved and invoked by the user when needed. They are very useful for building complex queries that you use with any regularity whatsoever.

#### Using a Quick Query as a Stored Procedure

1010SQL can invoke your Quick Queries using the SQL/ODBC stored procedure syntax.

In order to enable this feature, complete the following steps:

1. Open the ODBC Data Source Administrator dialog by clicking the Start Menu > All Programs > 1010data > 1010odbc > Configure 32-bit/Native ODBC.
2. Click the System DSN tab.
3. Click the system DSN labeled: www2.1010data.com.
4. Click the Configure... button.
5. Ensure that the Mangle Pathnames checkbox on the Connection tab is checked.
6. Click the SQL tab.
7. Make sure the Use metadata in SQL translator checkbox is checked.
8. Make sure the ...for join-order optimization checkbox is checked.
9. Click the OK button.

In order for a Quick Query to be used as a stored procedure in 1010SQL, the parameters of the query must be contained within a `<block>` operation as follows:

```
<block name="main" param1="default1" param2="default2" ...>
  <willbe name="newColumn" value="{[@param1]}"/>
  <sel value="oldColumn">{@param2}"/>
  ...
</block>
```

Note that the name of the block is `main`. The ODBC driver requires the block be called `main`. The parameters contained within the block, however, can be called whatever you decide.
Here are some other notes on using Quick Queries as stored procedures:

- Stored procedure variables cannot be mixed case. They must be in all lowercase characters.
- You can parameterize values in `<sel>`'s and `<willbe>`'s, but you can't parameterize a table to be linked to (or a base table).
- Only the code that contains the parameters for substitution needs to be inside the `<block>` tag.
- `<color>` is ignored unless it is outside the `<block>` tag.
- When you save the Quick Query as a stored procedure, you must have at least one value parameterized in the Quick Query code block. If there aren't any values you'd like to parameterize in the Quick Query, you can parameterize the base table. It will be ignored. The screenshot below is an example of parameterized values:

![Save As a Quick Query](image)

**Calling a 1010data Stored Procedure from SQL**

Once you have created a well-formed stored procedure in 1010data you can invoke it in your SQL code. There are two methods for doing this.

The first way to call a 1010data stored procedure in SQL is to use the `CALL` keyword, as follows:

```
CALL [STORED-PROC-FOO](arg1, arg2...argN);
```

In the example above, you provide your stored procedure with its parameters in a comma-separated list. This method is the ODBC standard method for calling a stored procedure. The next method is specific to 1010data's implementation of the ODBC standard. In this example, you can invoke the stored procedure with an SQL `SELECT` statement, as follows:

```
SELECT * FROM [STORED-PROC-FOO](arg1, arg2...argN);
```

In both cases you provide the parameters to the stored procedure in a comma-separated list. Both the above examples will produce identical results, so use which ever method you are most comfortable with.
If you are encountering issues with your ODBC driver configuration, 1010data does offer support for this service. In order to make sure you have captured all the information necessary for our support staff, make sure you take the following steps prior to contacting 1010data.

**Note:** Please make sure you have the most recent version of the ODBC driver installed on your system before you contact support. You can find the version number on the bottom of the ODBC Data Source Administrator dialog box. You can find out what the latest version of the driver is by checking the ODBC driver change log: [http://www.1010data.com/downloads/odbc/changelog.txt](http://www.1010data.com/downloads/odbc/changelog.txt)

### For Windows Users

1. Open the Logging Tab of the SQL1010ODBC DSN Configuration Dialog for your DSN (Start > All Programs > 1010data > 1010odbc > Configure 32-bit ODBC > System DSN Tab > Select the desired DSN > Configure... > Logging Tab).
2. Check "Log to".
3. If it is checked, uncheck "Append".
4. Click the top "Browse" button, and choose a *file* where the logs should be stored, ensure that the user using the DSN has permission to write to this file. We will call this file myLog.txt.
5. In the drop down menu select "Debug+XML" (note this will slightly impede performance and should be set back to "Basic" or "Detailed" for normal operation).
6. Click the bottom "Browse" button, and choose a *file* where the DSN's configuration options will be exported. We will call my file myConfig.txt.
7. Click the "Export Options" button.
8. If you get a "Success!" message click "OK", if you get another message take a screenshot.
9. Click "OK".
10. Consider deleting old log files in the folder where you have configured the ODBC driver to log, in step 12 we only want the log files that pertain to the issue you are experiencing.
11. Reproduce the issue you are experiencing. If you have configured your DSN with the settings described above the driver will produce two files myLog-########-#.txt and myLog-########-#.1010xml where the #'s can be any number or letter.
12. Take myLog-########-#.txt, myLog-########-#.txt.1010xml and myConfig.txt (if you didn't clean up your log folder during step 10 remember to only include the newest files!) and email them to support@1010data.com to open a support ticket for your issue. (If you already started a dialog via support@1010data.com regarding this issue feel free to reply to the dialog with the log files).

### For *Nix Users

1. Open libsql1010odbc.so's config file in a text editor (either in $HOME/.1010data/config or $TENTENODBC/config).
2. Set "LogP=1;".
3. Set "LogAppend=0;".
4. Set "LogFile=/some/folder/you/have/write/permissions/to/myLog.txt;".
5. Set "LogLevel=Debug+XML;".
6. Save the changes to your config file.
7. Make a copy of your config file, called myConfig.txt and delete the PWD line.
8. Consider deleting old log files in the folder where you have configured the ODBC driver to log, in step 9 we only want the log files that pertain to the issue you are experiencing.
9. Reproduce the issue you are experiencing. If you have setup your config with the settings described above the driver will produce two files myLog-########-#.txt and myLog-########-#.1010xml where the #'s can be any number or letter.
10. Take myLog-########-#.txt, myLog-########-#.txt.1010xml and myConfig.txt (if you didn't clean up your log folder during step 7 remember to only include the newest files!) and email them to
Upgrading the ODBC Driver

The 1010data ODBC driver is a fully supported interface product and should be kept up to date whenever possible. To update the ODBC driver Windows users can simply download and run the installer for the most current version. The installer will replace the existing ODBC driver installation on the system with the most current version. However, it isn't always practical to replace an entire installation. If you can't simply reinstall the driver, you can download and install the relevant .dll file on your system. However, you have to make sure you're downloading the correct .dll file for your system.

**Note:** Re-running the ODBC driver installer will not affect your existing DSN's.

If you have a 32-bit Windows installation, download the .dll file found here: [http://www2.1010data.com/downloads/tools/odbc/win32/sql1010odbc.dll](http://www2.1010data.com/downloads/tools/odbc/win32/sql1010odbc.dll). This file must be placed in the following folder in your Windows system:

C:\windows\system32

If, however, you have a 64-bit Windows configuration there are two possibilities. If you're using a 32-bit application/ODBC driver, download the file found here: [http://www2.1010data.com/downloads/tools/odbc/win32/sql1010odbc.dll](http://www2.1010data.com/downloads/tools/odbc/win32/sql1010odbc.dll). This file must be placed in the following folder on your system:

C:\windows\syswow64

Finally, if you're using a 64-bit Windows configuration and a 64-bit application/ODBC driver, download the .dll at the following location: [http://www2.1010data.com/downloads/tools/odbc/win64/sql1010odbc.dll](http://www2.1010data.com/downloads/tools/odbc/win64/sql1010odbc.dll). This file must be placed in the following location on your Windows system:

C:\windows\system32

Upgrading the ODBC Driver on *Nix

If you used the 1010data ODBC driver Linux installation script to install the driver on your Linux system. Download the file at the following link: [http://www2.1010data.com/downloads/tools/odbc/lin64/libsql1010odbc.so](http://www2.1010data.com/downloads/tools/odbc/lin64/libsql1010odbc.so) and overwrite the existing file found in the following location:

$HOME/.1010data/libsql1010odbc.so

If you did not use the installation script that 1010data provides for Linux users, download the file at the link above and place it in the directory where you initially installed the ODBC driver.

Known Issues and Workarounds

This section contains information about issues that have been identified by 1010data as suboptimal or inherent in the design of the 1010data ODBC driver. When practical, a workaround will be provided to avoid potential issues before they arise.

**Requesting UTF-8 Data**

1010data does not distinguish between wide and non-wide character columns, all data is stored as UTF-8 in the system, even strictly ASCII columns. However, the 1010data ODBC driver is an ANSI driver, meaning it doesn't support wide character encodings.

In practical terms, this means that the ODBC driver does not offer support for the SQL data type: `SQL_C_WCHAR`. If this data is requested, an error is thrown.
To avoid this issue, applications should be not be configured to request wide character data. At a lower level, if possible, applications should be configured to request `SQL_C_CHAR` data and anticipate that proper UTF-8 encoded data will be returned instead.