



MYOB ODBC Direct

How to write

Writing and committing information to an MYOB company file

This section provides an overview of writing and committing information to an MYOB company file.

Writing to an MYOB company file

To write to an MYOB company file, first connect to the file, then use SQL insert statements to insert information into import tables. The import tables are write-only. Information is committed by importing it from the import tables into an MYOB company file.

Note: There are differences in the import routine fields available in the multi-user and single-user MYOB products. Ensure that when you develop applications or solutions using the MYOB ODBC Direct driver that these solutions take into account the field differences between the multi-user and the single-user products.

Before you can write to a company file, the file must have been activated. The driver will check that the company file has been activated before importing any data. If the company file has not been activated, the import will not proceed and the driver will return an error code.

- For further information about connecting to a company file, see [‘Connecting to a data source’ on page 3](#)
- For further information about structuring the SQL insert statements see [‘Structuring the SQL INSERT command to insert information’ on page 5](#)

Committing information to an MYOB company file

Information is written to the company file after each insert statement or after a commit, depending on the `SQL_ATTR_AUTOCOMMIT` setting. For information about setting the commit method, see [‘Setting the commit method’ on page 7](#)

Errors and warnings

The MYOB ODBC Direct driver returns a code on completion of each SQL command.

For more information about MYOB ODBC import errors and warnings, see the *Testing or retrieving an error code* document.

Connecting to a data source

You can connect to an MYOB company file (the data source) for read and write access in two ways: either by setting up a Data Source Name (DSN) for the file using the **Data Sources (ODBC)** administrative tool in the Windows Control Panel, or without a DSN.

When connecting to the data source for writing, a connection string is required. The parameters within the connection string, defined by keywords and their assigned values, override default DSN values.

A connection string is a concatenation of a number of keywords and assigned values in the format of `[KEYWORD]=[VALUE];`

Keywords and their assigned values are separated from other keyword/value pairs with semi-colons. A semi-colon is also required to terminate a connection string.

Connection strings can be used when connecting to a data source with a DSN or, when the DRIVER keyword is set, connecting directly to the data source.

For further information about connection string keyword values, see [‘Keywords and supported values’ on page 8](#)

Connecting using a DSN

When connecting to a data source for writing using a DSN, the following keywords must be set:

```
DSN=<dsn name>;  
ACCESS_TYPE=READ_WRITE
```

Note: When connecting using a DSN, keywords and values in a connection string take precedence over the equivalent values set in the DSN.

Connecting without using a DSN

To connect to a data source without a DSN, use the DRIVER keyword and set the following additional keywords:

```
ACCESS_TYPE=READ_WRITE
TYPE=MYOB;
UID=<user ID>;
PWD or PASSWORD=<password>;
DATABASE=<path and name of company file>;
HOST_EXE_PATH=<path and name of MYOB application>;
KEY=<complete path and file name of key file>;
or
KEY=<key>;
```

For more information on connection string keywords and their values, see ['Keywords and supported values' on page 8](#)

An example connection string

An example connection string is shown below:

```
" Driver={MYOB ODBC}; TYPE=MYOB; UID=Administrator; PWD=MyPassword;
KEY=C:\Premier\123456700000.key; DATABASE
=C:\Premier\Test.myo;HOST_EXE_PATH=C:\Premier\MYOBP.exe; NETWORK_PROTOCOL
=NONET; DRIVER_COMPLETION=DRIVER_NOPROMPT; ACCESS_TYPE=READ_WRITE; "
```

Note: The driver name will depend on the version of the driver. A connection string can be a maximum of 16k (16384) characters in length. A password can be a maximum length of 31 characters.

Structuring the SQL INSERT command to insert information

When structuring the INSERT command, the MYOB file import structure must be observed. Refer to the *Write data dictionary* document for information about structuring an import record set for an import table.

The INSERT command is in the standard SQL format for single and multiple line inserts. However, when there are several transactions, such as sales or invoices, the individual transactions need to be separated by executing the END TRANSACTION command.

The END TRANSACTION command allows the addition of multiple transactions. This command cannot be parsed to the ODBC driver as part of an insert statement block, but must be parsed separately. The ODBC driver buffers all insert statement blocks to the driver, only committing them to the import routines of the MYOB application when a COMMIT is called.

Example: Multiple INSERT into the Tax Code table (.NET)

The following code snippet demonstrates a .NET Tax Code table INSERT that passes multiple items to the driver and commits them in one action.

```
cmdSQLInsert.CommandText = "INSERT INTO Import_Consolidated_TaxCodes
VALUES ('TST', '', 'WET')";
cmdSQLInsert.ExecuteNonQuery();

cmdSQLInsert.CommandText = "END TRANSACTION";
cmdSQLInsert.ExecuteNonQuery();

cmdSQLInsert.CommandText = "INSERT INTO Import_Consolidated_TaxCodes
VALUES ('TST', '', 'GST')";
cmdSQLInsert.ExecuteNonQuery();

cmdSQLInsert.CommandText = "END TRANSACTION";
cmdSQLInsert.ExecuteNonQuery();

cmdSQLInsert.CommandText = "INSERT INTO Import_Consolidated_TaxCodes
VALUES ('XXP', '', 'GST')";
cmdSQLInsert.ExecuteNonQuery();

myTrans.Commit();
```

Note: MYTRANS is an ODBC transaction object in the .NET environment. For more information regarding the MYTRANS ODBC transaction object, please refer to the MSDN help files. A full example on how to create the transaction object is available within the sample source code examples.

Example: Multiple INSERT into the Miscellaneous Purchases table (SQL)

The following example assigns values to both mandatory and non-mandatory fields in the Import_Miscellaneous_Purchases table.

As in the above example, INSERT statement blocks are separated by an END TRANSACTION statement block, enabling the driver to identify and buffer the two separate insert statements for execution when COMMIT is called.

```
INSERT INTO Import_Miscellaneous_Purchases
(CoLastName, FirstName, PurchaseNumber, PurchaseDate, SuppliersNumber,
Inclusive, Memo, Description, AccountNumber, ExTaxAmount, IncTaxAmount,
Job, TaxCode, NonGSTImportAmount, GSTAmount, ImportDutyAmount,
PurchaseStatus, CurrencyCode, ExchangeRate, PaymentIsDue, DiscountDays,
BalanceDueDays, PercentDiscount, AmountPaid, Category)
VALUES
(('Mountain Spring', '', 'PJ000001', '05/05/2004', 'Supplier Num', 'X',
'Purchase; Mountain Spring', 'Description Line 1', '61100', 90.91, 100.00,
'', 'GST', 0.00, 9.09, 0.00, 'B', 'AUD', 1.000000, 5, 1, 30, 0, 0.00, ''),
('', '', '', '', '', '', '', 'Description Line 2', '62100', 181.82, 200.00,
'', 'GST', 0.00, 18.18, 0.00, '', '', '', '', '', '', '', ''),
('', '', '', '', '', '', '', 'Description Line 3', '62100', 272.72, 300.00,
'', 'GST', 0.00, 27.28, 0.00, '', '', '', '', '', '', '', ''));
END TRANSACTION
INSERT INTO Import_Miscellaneous_Purchases
(CoLastName, FirstName, PurchaseNumber, PurchaseDate, SuppliersNumber,
Inclusive, Memo, Description, AccountNumber, ExTaxAmount, IncTaxAmount,
Job, TaxCode, NonGSTImportAmount, GSTAmount, ImportDutyAmount,
PurchaseStatus, CurrencyCode, ExchangeRate, PaymentIsDue, DiscountDays,
BalanceDueDays, PercentDiscount, AmountPaid, Category)
VALUES
('Supplier', '', 'PJ000002', '05/05/2004', 'XYZ', 'X', 'Purchase;
Supplier', 'Description Line 1', '91000', 123.45, 123.45, '', 'N-T', 0.00,
0.00, 0.00, 'B', 'AUD', 1.000000, 5, 1, 30, 0, 0.00, '')
COMMIT
```

Rules for writing to an MYOB company file

When inserting records via SQL, the following rules must be observed:

- Mandatory fields must be included.
- General fields are only required within the first line of a record set, or the Total Line. Refer to the *Write data dictionary* document for information about general fields for an import record set.
- All records inserted into an import table are loaded into the company file via an import function. The records must comply with the import file format. Some import files require a double carriage return to indicate the end of one record set and the start of a new record set. To indicate the end of a record set and the start of a new record set, use the END TRANSACTION command.
- The driver only provides for the INSERT of records via SQL (UPDATE is not supported).
- DO NOT apply currency formatting to dollar values.
- When AUTO COMMIT is turned on, only one import transaction can be performed at a time.
- When AUTO COMMIT is turned off, all inserts for the transaction must be to the same table.
- An END TRANSACTION statement only marks the end of a set of records.
- A COMMIT or ROLLBACK must be sent before INSERT parameters can be changed (e.g. change table, or change field selections).
- The driver can handle a maximum of 32k (32768) characters in an SQL statement block.

Setting the commit method

Data is committed to the company file either after each SQL statement (auto commit) or as a batch of transactions (manual commit).

The commit method is set using the **SQL_ATTR_AUTOCOMMIT** keyword in the connection string, sent when you connect to the company file.

For further information about connection string keyword values, see [‘Keywords and supported values’ on page 8](#)

Keywords and supported values

The following table describes connection string keywords and values that are supported by the MYOB ODBC Direct driver.

Mandatory keywords

When using a connection without passing a DSN name, certain keywords must be assigned values.

For a read connection, DRIVER; TYPE; DATABASE; UID; and if there is a password, PWD; must have assigned values.

For a write connection, HOST_EXE; ACCESS_TYPE; DRIVER; TYPE; DATABASE; UID; KEY; and if there is a password, PWD; must have assigned values.

Keyword	Attribute value description	Values	Example
DRIVER	Name of the driver. If DRIVER keyword is set, DATABASE keyword must also be set.	{MYOB ODBC x.x}	DRIVER={MYOB ODBC 6.0.0};
TYPE	Set this value to "MYOB".	MYOB	TYPE=MYOB;
DATABASE	Explicit path, including the filename where the MYOB company file is located.	Value is the location and name of the company file	DATABASE=C:\Premier\Test.myo;
DSN	Use only if a DSN has been set up for a data source. If DSN is set, do not use the DATABASE keyword, as it will be ignored.	Value is the DSN name	DSN=<DSN name>;
UID	A user ID. The User name as entered in MYOB. Note: If this value is not NULL, it will overwrite the username entered in the DSN.	Value is an appropriate User ID for the company file	UID=Administrator;
PWD or Password	The password for the user ID (UID) set up in the DSN, or an empty string if there is no password. Note: If this value is not NULL, it will override the password entered when setting up a the DSN. A password can be no more than 31 characters in length.	Value is the appropriate password for the User ID of the company file	PWD=MyPassword; or PWD;

Keyword	Attribute value description	Values	Example
KEY	Use only if you have a valid developer key, provided by MYOB as a key file. The key file contains the 128 bit key value.	Value is either a key value, or the explicit path and filename of a key file containing the key value. OEMs must use the key value, not the path and file name. By using the explicit path and filename of a key file: <ul style="list-style-type: none"> the contents of a key file can be changed without needing to recompile an application or modify a connection string the actual key value is hidden when used within a DSN set up. By using the key value a key file does not need to be installed on a client machine.	KEY=C:\Premier\123456700000.key; or KEY=<key value>;
HOST_EXE_PATH	Explicit path to the location of the MYOB executable file. This executable must be compatible with the company file. (that is, same country code and same version number). Note: Should the application already be running when executing it, it must be using the same company file as that specified in the DATABASE keyword.	Value is the location and name of MYOB executable file.	HOST_EXE_PATH=C:\Premier\MYOBP.exe;
NETWORK_PROTOCOL	Network Protocol to be used while using MYOB. NETWORK_PROTOCOL defaults to NONET if not explicitly set	NONET NETBIOS TCPIP	NETWORK_PROTOCOL= NONET;
DRIVER_COMPLETION	DRIVER_NOPROMPT – Do not prompt the user for additional login information (username and password). An error will be returned if insufficient login information is provided. DRIVER_PROMPT – Display the dialog box prompting the user for additional login information (username and password). The Default value is DRIVER_NOPROMPT	DRIVER_NOPROMPT DRIVER_PROMPT	DRIVER_COMPLETION= DRIVER_NOPROMPT;

Keyword	Attribute value description	Values	Example
ACCESS_TYPE	<p>READ – Connect with read access only. (Default)</p> <p>READ_WRITE – Connect with both read and write access.</p> <p>Setting the ACCESS_TYPE to READ_WRITE will force NETWORK_PROTOCOL to TCPIP</p>	<p>READ</p> <p>READ_WRITE</p>	ACCESS_TYPE=READ_WRITE;
IDENTIFY_BY	<p>When importing cards and invoices:</p> <p>CARD_NAME uses the name on the card as an identifier.</p> <p>CARD_ID uses the card ID as an identifier.</p> <p>Note: The Australian version of the ODBC Direct driver defaults to CARD_ID if this parameter is not set.</p> <p>When importing invoices, use card_name as an identifier, as the invoices import will not allow card_ID as the identifier.</p>	<p>CARD_NAME</p> <p>CARD_ID</p>	IDENTIFY_BY=CARD_NAME;
INSERT_TYPE	Determines how duplicate records are handled by an import.	<p>REJECT_DUPLICATES</p> <p>UPDATE_DUPLICATES</p> <p>ADD_DUPLICATES</p>	INSERT_TYPE= REJECT_DUPLICATES;
SQL_LOGIN_TIMEOUT	Connection timeout in seconds. This value cannot be less than 30 seconds as MYOB needs a significant amount of time to initialise its Java libraries when it launches.		
SQL_ATTR_AUTOCOMMIT	<p>If this keyword is set each individual statement is automatically committed when it completes successfully. No explicit transaction management functions are necessary. However, the return code from the function must still be checked as it is possible for the implicit transaction to fail.</p> <p>This keyword value defaults to 0 (manual commit). With manual commit all executed statements are included in the same transaction until calling END TRANSACTION specifically completes it.</p>	<p>0</p> <p>1</p>	<p>SQL_ATTR_AUTOCOMMIT=0;</p> <p>SQL_ATTR_AUTOCOMMIT=1;</p>

Keyword	Attribute value description	Values	Example
SUPPRESS_WARNINGS	<p>If a warning message is issued after an import statement, the error buffer is not automatically cleared. To clear the buffer, a rollback statement is required.</p> <p>If a second import is attempted without using a rollback statement, where a warning was received from the initial import, the second import will not go ahead.</p> <p>Setting the keyword SUPPRESS_WARNINGS to the value of TRUE (SUPPRESS_WARNINGS=TRUE;) enables consecutive import warnings to occur without requiring the rollback statement to be applied after each warning.</p>	TRUE FALSE	SUPPRESS_WARNINGS=TRUE;