



MoneyWorks Automation

What is MoneyWorks Automation

MoneyWorks Automation provides MoneyWorks with the ability to communicate with other programs, allowing MoneyWorks to be interrogated by other systems, and data to be submitted by these systems to MoneyWorks.

It is provided because we realise that MoneyWorks cannot be all things to all people, and that there are things people need that MoneyWorks does not provide.

Using automation allows you to do things like:

- integrate MoneyWorks into Excel for reporting and budgeting;
- have external consolidation routines for multiple MoneyWorks documents;
- process direct debits;
- automate the import of invoices and price books from suppliers.

To create extensions using MoneyWorks automation you need (on Windows) a COM enabled development system such as VB, Delphi, RealBasic. On Mac you need to be able to send and receive AppleEvents—AppleScript is an ideal solution.

In addition there are free plug-ins available for MoneyWorks Automation for FileMaker Pro and 4D. These provide cross-platform compatibility (i.e. a single database solution should run on both Mac and Windows).

Note

This document is very much a work in progress.

Conventions used

Where a feature has been implemented in a point release, that is indicated by (for example) [MW4.0.7], indicating that you must be using MoneyWorks Gold 4.0.7 or later.

Examples are in courier:

```
namecode = "SMITH"
```

A note on Quotes

Some of the parameters you need to pass to MoneyWorks will contain embedded quotes. For example, to find all the transactions from customer SMITH you might use the search expression:

```
namecode = "SMITH"
```

When you come to embed this in a string to pass to MoneyWorks you will need to put escape marks around the quotes. In AppleScript, you might have an expression like:

```
export "Transaction" using search "namecode = \"SMITH\""
```

In VB, the equivalent command would be:

```
mw.export("transaction", "namecode = ""SMITH""", "")
```

On other platforms it might be different.

To simplify the unravelling of these (it can be really tedious to get the quotes correctly balanced in complex expressions), MoneyWorks will generally recognise the use of ` as a quote (that's the character at the top left of the keyboard). Thus the search expression above code be specified as:

```
"namecode = `SMITH`"
```

Note that the quotes must balance, so `SMITH" will not work.

The apostrophe (') is used to indicate a date. So '1/12/03' is not the same as "1/12/03".

Key differences between Windows and Macintosh

On **Windows**, MoneyWorks presents itself as a COM server, allowing COM clients access to a number of methods and properties. In the Windows environment, MoneyWorks is an object, and hence you have to create the object in your application before it can be used, as in the VB example below.

```
Sub test()  
    Dim mwObj as MoneyWorksApplication  
    ' attach to running MoneyWorks or create new instance  
    Set mwObj = GetObject("", "MoneyWorks.Application")  
    mwObj.Open ("c:\Program Files\MoneyWorks\Kiwi Widgets.mwd3")  
    x = mwObj.ImportFile("C:\import text.txt", "NameMap.imo")  
    mwObj.Quit  
End Sub
```

Note that the VB project must have a reference to the MoneyWorks type library (Tools:References).

For the MoneyWorks plug-ins for FileMaker and 4D (for Windows), the creation of the object is done by the "Connect" command (which is why you must call it first). The object is held until you call the "Disconnect" command, which you should do before quitting your FileMaker or 4D application (it is safe to call Disconnect without first calling Connect).

On **Macintosh**, MoneyWorks is an AppleEvent aware application, and can (normally) receive and respond to AppleEvents. There is no need to create or maintain an object as in Windows. The equivalent script is:

```
tell application "MoneyWorks Gold"  
    open file "Acme Widgets"  
    import file "text.txt" using map "NameMap"  
    quit  
end tell
```

For the Macintosh implementation of the plug-ins, you should still use the "Connect" and "Disconnect" commands for script compatibility with Windows. The Connect command is used to indicate which application you are opening.

The Open Command

The Open command allows you to open a MoneyWorks document.

Syntax

Windows:

Open("documentpath")

Mac:

Open file "documentpath"

If that document is already open, the command has no effect (i.e. MoneyWorks will not close and reopen the same document). If another document is open, it is closed and the requested document is open.

Connecting to a MoneyWorks server

MoneyWorks also has an undocumented facility for opening a network connection to a server from the command line or a shortcut. You do this by passing a URL instead of a file path on the command line in the form "moneyworks://user:pass@server:port" where server is the server IP or domain name and port is usually 6674.

Every part of this except server is optional, i.e

```
[ "moneyworks://" ] [ user [ ":" pass ] "@" ] server [ ":" port ]  
[ "?doc=" documentname ]
```

The Export Command

The Export command will export all the records that match the specified search function from the nominated logical file. Entire records are exported (you cannot specify individual fields). Use this when you want to extract information in bulk (the Lookup function can be used to extract single fields of specific records).

Syntax

Windows:

Export ("logicalfilename", "searchExpr", "destPath")

Mac:

export "logicalfilename" [**using search** "searchExpr"] [**into** "destPath"]

This causes MoneyWorks to export the information into "destPath", the nominated target file (the full path name must be specified)—the information in the file will be the same as if the File>Export command was used.

If destpath is empty, MoneyWorks will return the exported text as the result of the Export command, allowing it to be directly manipulated by the calling program. The information exported this way is different from that exported to a file. Each record in a memory export is preceded by 3 <record information> fields. These are:

Slot number -- physical record number in the file

Sequence number -- unique record number (incremented for every new record)

Date changed

Logical File

The logical file is the name of the internal MoneyWorks file, and is one of:

- Account
- Ledger
- General
- Department
- Transaction
- Detail
- Log
- TaxRate
- Name
- Payments
- Product
- Job
- Build
- JobSheet
- Memo

The Search Expression

The search expression can be any MoneyWorks search (as used in the MoneyWorks Advanced Find command). Note that it is important to balance quotation marks. Also note that there is no "Contains" search (as in the standard MoneyWorks Find command)—instead use the PositionInText function.

MoneyWorks Automation also supports special search commands:

- "*" The highlighted records in the nominated list, or all records if none highlighted (or list window is not open);
- "**" The highlighted records in the nominated list, or nothing if non highlighted (or list window is not open);
- "=" A single record containing the field names [MW4.1].

Examples:

```
Export("Name", "*", "")
    all highlighted names (or all if none highlighted)
Export("Transaction", "**", "")
    all highlighted transactions (or none)
Export("Transaction", "Type=`DII`", "")
    all debtor invoices with outstanding balances;
Export("Name", "BalanceD90 > 0", "")
    all debtors with outstanding invoices from three or more
    cycles ago.
```

Notes

The amount of text that can be received by other applications can be limited, For example 4D is restricted to about 32K (using the MWExportText command) and FileMaker Pro v6 and earlier is restricted to about 63K (using the MWKSEExportText command). If MoneyWorks returns more text than can be handled, the associated plugin will return a -999 error. For this reason if you are processing large amounts of text, it may be preferable to Export the information to a file.

Special Exports

Account

When exported through memory, you will get essentially the account list (with no balances). If you export it to a file, you will get a file with the current balances in the right hand column, and with a separate record for each account-department. You will also get other GL information encoded at the end of the file.

Ledger

This gives the complete ledger record (all balances and budgets). You can't export this to a file: The result is always returned directly by the export command.

If you provide a search based on the accountcode (e.g. "Accountcode = `3600`") and the accountcode is departmentalised, you will get all all accountcode-department records exported.

To export a single departmentalised record, you need to base your search on the concat field (e.g. "concat = `3600-JB`").

Note: If you are looking for a specific account balance, it is better to use the MoneyWorks GetBalance() and GetMovement() functions.

Build

Exports the recipes for the nominated product(s), or all recipes if no product specified.

When exporting to a file, the fields exported are:

- Product code of parent
- Quantity
- Product code of part

When exporting through memory, the fields are:

- <record info>
- Sequence number of parent product
- Quantity
- Product code of part

Memo

Exports the memo fields for the nominated names records. At present [MW4.0.9] this cannot be output to a file.

The field order is:

- <record info>
- Sequence number of parent (Name) record
- line number
- memo date
- recall date
- flags (unused)
- memo text

Import

Sending information to MoneyWorks is normally achieved by the use of the import command (although in certain situations you can use the Replace function, or the SetBudget function).

Syntax is:

Windows:

Import("filePath", "MapName") — import from file "filePath"
ImportText("text", "MapName") — import text

Mac:

Import file "filePath" **using map** "Mapname" — import from file "filePath"
Import "text" **using map** "MapName" — import text

Mapname is usually the name of the import map (which should be in the Import Maps folder in the MoneyWorks Custom Plug-ins folder). You need to set this up in advance of scripting the import. For files with a fixed import field order you just specify the logical file name preceded by the characters "://" (e.g. "://Accounts")

Mapname can also be a special XML string (see below).

Setting up an import map

The easiest way to set up an import map is to have some sample data in the form that you want to import. Copy it onto the clipboard, and paste it into the appropriate MoneyWorks list (for Payments/Receipts against invoices, hold down the shift (win) or option (mac) keys and copy/paste). An import map will be presented. Align the fields (this may take several goes), and when all is working OK use the Save button to save the map. If you need to change the map later, use the Load button, reconfigure the map, and then resave.

Importing using XML instead of Map [MW4.0.6]

The map parameter may be xml text containing a single "args" tag with the following attributes:

```
<?xml version="1.0"?>
<args file="transaction"
map="updatemap"
update="true"
post="true"
seqnum="999"/>
```

Yes, the <?xml version="1.0"?> is required.

File must be present. Useful values are "transaction", "account", "user", or "build". You can import into other files this way but there is no point—just use the regular syntax.

For file = "transaction" you must also specify the name of the import map in the Plug-Ins (using the map attribute). The update and post attributes are optional and are used to "recall" invoices (see Recalling an Invoice);

For file = "account", the import data should be in the same format you get when you

Copy accounts from the accounts list;

For file = "build", the import data has to be in the order for builds (see Importing Builds);

For file = "user", the import data must be in the order for user data (see Importing User);

Recalling an Invoice

An invoice in MoneyWorks can be "recalled", allowing existing invoices to be updated from other programs. To do this you need to set `update="true"`, and specify the sequence number of the invoice (`seqnum = "999"` in the XML import map). If the `post="true"` option is specified, the new invoice will be posted upon import.

If the invoice identified by `seqnum` is not posted, then it gets deleted upon successful import of the new transaction (new one effectively replaces it).

For posted debtor invoices, providing any payments are not processed for GST, it is cancelled. If there were any payments on it, they become overpayments which may then be (manually) allocated to the new (or any other) invoice.

For posted creditor invoices [MW4.1], providing there are no payments against it, the invoice is cancelled. If there are payments against it an error is raised.

Note that the newly created transaction will have a different sequence number to the original, even if the original was deleted in the recall.

Importing Accounts

You can import accounts using the pseudo map `"/Account"` (or the XML file = `"account"`). The import data must be in the same order as you get when you copy accounts, i.e.

- Account
- Description
- Type
- Group
- Tax Code
- Category
- P&L
- Accountant's Code
- Category2
- Category3
- Category4
- Bank Account Number

The fields after P&L are optional (but the file must have the same number of fields in each record).

The first line is assumed to contain headings and, apart from the number of fields, is ignored.

The account meta-data that is supplied when you copy an account is optional, and is used to create categories, departments etc.

Importing Builds

Builds can be imported using the `"/Build"` pseudo-map. A build is a list of items and

quantities that are used to manufacture an item (and can be seen in the Inventory tab of the product record).

The file format for importing builds is fixed, and is:

```
Product Code (of the item being built)
Quantity
Product Code (of component item).
```

The component item must have an expense account associated with it in the product record.

When you import a build, it replaces any existing build (i.e. all the existing build records are deleted, and the new ones added).

Importing User

The User file is a special internal MoneyWorks file that can be used by external applications to store their own data in. There are two fields, a key field and a data field. Use the pseudo-map ":/User" for this logical file.

Note: To avoid conflicts with other external applications, please check with Cognito for key field availability.

When a record is imported with the same key value as an existing record, the existing record is overwritten (this is the mechanism for updating the information). If the record has a new key, it is added to the file.

The fields are:

```
Key   9 characters
Data 245 characters
```

is not allowed as the first character of the key (reserved for use by MW).

[MW4.0] The user file does also contain the users set up in Sharing & users. If you export it, you will get encoded user data.

The Evaluate Function

Use the Evaluate function to interrogate one of the MoneyWorks environment variables, or to invoke one of the special internal MoneyWorks functions. The Evaluate function always returns a text string.

Syntax

```
Evaluate("Expression")
```

Examples:

```
Evaluate("DocumentPath")
    — returns full path to current Moneyworks document
Evaluate("Name")
    — returns the company name
Evaluate("Today()")
    — returns today's date (using the MoneyWorks Today() function)
Evaluate ("1+2*3")
```


— returns 7.00

Useful Parameters:

File location:

- Documentpath
- DocumentName
- Applicationpath
- CustomPluginsPath
- StandardPluginsPath

Company Details (from Show>Company Details):

- Name
- Address1
- Address2
- Address3
- Delivery1
- Delivery2
- Delivery3
- Phone
- Fax
- email
- gstNum

Document

- AgingCycle
- Username
- Initials

Country details:

- Locale — first three characters of MoneyWorks registration
- Regnum — the MoneyWorks registration code
- GSTRegname
- CoRegname
- TaxName — returns "GST" for NZ, Aus, "VAT" for UK etc
- TaxName2 — for two-tier tax systems—"PST" in Canada

Dates and Periods

- CurrentPeriod — the internal number of the current period
- DateToPeriod(date) — returns the period number of the date
- PeriodToDate(period) — returns the period-end date of period number

The Lookup function

This function returns the value of the nominated field based on the key value passed. The syntax is:

Lookup("KeyValue", "file.field")

For example

Lookup("SMITH", "Name.DBalance")

returns the current balance of the debtor whose code is SMITH.

The Lookup command is not recognised by COM or AppleEvents, but must be passed as a parameter to the Evaluate command (and this is where the quotes start getting tricky). The previous example would have to be invoked as:

Evaluate("Lookup(`SMITH`, `Name.DBalance`)")

One use of the Evaluate function is to test if a record with a known key value exists. For example the following VB returns true if there is a Name record in the MoneyWorks file with the namecode passed:

```
Function NameExists(Code As String) As Boolean
    Dim res As String
    res = MW.Evaluate("Lookup(`" & Code & "`," & Name.code`)")
    NameExists = IIf(StrComp(Code, res, vbTextCompare) = 0, True, False)
End Function
```

The Lookup function is implemented as a standalone function in the 4D Plug-in.

The GetBalance Function

Use this function to get a balance for a nominated range of general ledger records for the nominated period (as determined by the date).

Syntax:

Getbalance("LedgerSearch", 'Date' [, "Budget"])

It is important to realise that this function operates on the Ledger file, not the Accounts file. The Budget parameter (if present) should be "A" or "B", to return the budgeted movement.

Example:

```
Evaluate("GetBalance(`accountcode = \"3@\"`, '30/9/1')")
```

Returns the total balance of all ledger records starting with "3" for the period ends on or after 30/9/01. Note that the date is delimited by apostrophes, not quotes.

```
Evaluate("GetBalance(`accountcode = \"3@\"`, '30/9/1', `A`")
```

returns the budgeted (A) balance instead of the actual.

```
Evaluate("GetBalance(`Department = \"jb\"`, '30/9/1')")
```

returns the balance for the department "JB" in the same period.

Note that if accountcode 3600 is departmentalised, the following will return the balance of all subledgers:

```
"GetBalance(`accountcode = \"3600\"`, '30/9/1')"
```

but to determine a specific departmentalised account you need to use:

```
"GetBalance(`concat = \"3600-JB\"`, '30/9/1')"
```

The GetMovement Function

This is similar to the GetBalance function, but returns the movement for the nominated ledger records between the nominated periods. The syntax is:

GetMovement("ledgerSearch", 'fromperiod', 'toperiod' [, "Budget"])

Example:

```
Evaluate("GetMovement(`accountcode = \"3@\"`, '30/4/1', '30/9/1')")
```

Returns the movement for all ledger records starting with 3 between the April and September periods. If the financial year started on Apr 1st, this would represent the year to date figure.

The SetBudget Function

The SetBudget function is used to set the budget (A or B) for the nominated ledger record for the nominated period (as determined by the date). This must be passed as a parameter to the Evaluate function.

Syntax:

```
SetBudget(Budget, ConcatCode, PeriodDate, Value)
```

Example:

```
Evaluate("SetBudget(`A`, `3600-JB`, '30/9/01', `3900.00`")
```

Sets the A budget for ledger record "3600-JB" in the Sept 01 period to \$3,900.00.

You can use the Setbudget function to populate the MoneyWorks budgets from a spreadsheet (see the "Budget Spreadsheet" in the Office examples on the Developers page of the Cognito website).

The Replace function

The Replace function is similar to the Select>Replace command in MoneyWorks in that it replaces the nominated fields in the records as determined by the search function by the nominated value. Must be passed as a parameter to the Evaluate function.

Syntax:

```
Replace("file.field", "search", "value")
```

Example:

```
set x to Evaluate("Replace(\"name.colour\", `dbalance > 2000`,  
`\"red\"`")
```

An AppleScript command that sets the colour to red of every debtor with an outstanding balance of more than \$2000. Note the embedded quotes around red.

```
set x to Evaluate("Replace(\"transaction.colour\", `type = \"DII\"  
and duedate < '1/8/01' and gross > 100`, `\"red\"`")
```

An AppleScript command that sets the colour to red of every overdue invoice that was due before 1st Aug 2001 and is for more than \$100.

Notes:

- Not all fields can be replaced (for example attempting to replace "transaction.gross" will not work).
- The Replace command should be used with extreme caution.
- This Replace command takes precedence over the internal Replace command in MoneyWorks (which replaces a part of a string with another). The internal

Replace cannot therefore be invoked from outside MoneyWorks.

The DoReport Command

Use the DoReport command to have MoneyWorks run a nominated report.

Syntax:

Windows:

DoReport(reportName as string, startDate as string, endDate as string,
outputMode as integer, destinationFilePath as string, jobDialogs as
Boolean)

e.g. `x = mw.DoReport("Profit & Loss for Month.crep", "1/4/03",
"1/4/03", "", "", "false")`

puts the contents of the Profit & Loss for Month report for April 2003 into X
(as a tab delimited string)

Mac

Do Report [from period "date"] [to period "date"] [outputting to
printer/preview/text file] [spooling to "file"] [job dialogs boolean]

e.g. `set x to do report "Profit & Loss for Month" from date "1/4/03"
to date "1/4/03"`

puts the contents of the Profit & Loss for Month report for April 2003 into X as
a tab delimited string.

Note that, apart from date, the report settings (such as Omit Zero Balance and Breakdown) cannot be specified remotely—the report is run with the last settings used to print the report. If you need specific settings, make a copy of the report (it can go into your Custom Plug-ins) with the required settings.

The MoneyWorks Command Menu

You can invoke externals directly from MoneyWorks by placing them in the Scripts folder (in your Custom Plug-ins). They will then appear at the bottom of the MoneyWorks Command menu.

On Mac, any compiled AppleScript in the Script menu will appear in the Command menu.

On Windows, and .exe, .bat, .xls, .doc will appear.