Introduction

If you use Moneyworks v5 or newer, and would like to integrate it with 4D v11 or newer, then you've come to the right place! Simply download and install the component, play with the example database (or read the documentation), and you're away.

This document is based on the Moneyworks command-line documentation from Cognito (http://cognito.co.nz/developer/cli-manual.

The component has been tested on 4D v11r8, v12.1 and v13.4, connecting to Moneyworks v5, v6 and v7 respectively. It will operate as a trial for one hour. For a license key, or to obtain the source code, please contact us (http://www.sussol.net/contact/). Pricing is a simple flat rate of $250 per year for unlimited use.

Now updated for 4D v15, including support for REST connections to versions of Moneyworks newer than v6.1 - see http://cognito.co.nz/developer/moneyworks-datacentre-rest-api/

Examples using the available methods are included in the accompanying example database, which also includes a form which can be used to experiment (use the **mwks_preferences** method to open it).

Downloads

Component (unicode) + example database for v11 is available here

Component only (without example database), with unicode switched off, for v11 is available here

Component only (without example database) for v12 (fixed for Moneyworks 7.1.9) is available here (Previous version available here)

Component only (without example database) for v13 (fixed for Moneyworks 7.1.9) is available here

Component only (without example database) for v14 (fixed for Moneyworks 7.1.9) is available here

Component only (without example database) for v15 (including Moneyworks REST support) is available here

This wiki will always have the latest version of the documentation, but an older PDF version is available here

Files and Strings

The full path must be given for any files, as the Moneyworks command line doesn't know about the Moneyworks plugins folder. Paths on the Macintosh should use the posix standard rather than the old OS9 standard. Spaces within paths may need to be escaped.

For non-REST connections, double quotes within strings should be escaped, or use Char(Double

Quote). Single quotes should also not be used (except in the *extra_parameters* parameter in the export/import/report and form methods, which is passed verbatim to Moneyworks) as they are used internally to delimit the parameters passed to the Moneyworks command line. For REST connections, text parameters are automatically URL encoded using standard HTTP escape sequences (except for ***mwks_run_command***)

Included Methods

mwks_register

(registration_key) → Boolean

Parameter	Type	Description
registration_key	String	Registration key for component

Return value

True if *registration_key* is valid, otherwise ***False***

Description

Registers the component using the given key string. If the key isn't valid, the component can still be used for an hour before it times out. This method should be called before attempting to use any other component method e.g. in the **On Startup** database method.

mwks_setup_connection

(p_connect; p_application; p_error; connection; mwks_location

[if connection=1 (Gold local): data_path; doc_username; doc_pass]

[if connection=2 (Gold server): client_license; doc_username; doc_pass; server_ip; server_port]

[if connection=3 (DC server) or 4 (DC REST server):data_path; doc_username; doc_pass; server_ip; server_port; dc_username; dc_pass;dc_os]

{; open_datafile} {; xml_output}) → Boolean

Parameter	Type	Description
p_connect	Pointer to string	**Returned** containing the connect string
p_application	Pointer to string	**Returned** containing the Moneyworks application string

p_error	Pointer to string	**Returned** containing the error string
connection	Integer	1=Gold local, 2=Gold server, 3=DC server, 4=DC REST server
mwks_location	String	Full path to Moneyworks executable (not needed if connection=4)
data_path	String	Full path to data file (if connection=1), or document filename (if connection=3 or 4)
doc_username	String	Username for data file (ignored if doc_pass is blank)
doc_pass	String	Password for data file (may be left blank)
client_key	String	License key for client of Gold server
server_ip	String	IP address of Gold server/DC server/DC REST server
server_port	String	If left blank, uses default port (6674 for Gold server, 6699 for DC server, 6710 for DC REST server)
dc_username	String	Username for DC or DC REST server login (may be left blank if server not running in ASP mode, otherwise it is required)
dc_pass	String	Password for DC or DC REST server (may be left blank)
dc_os	String	OS of DC server - either 'Windows' or 'Macintosh'
open_datafile	String	Optional parameter - if non blank, try to open the data file
xml_output	String	Optional parameter - if non blank, use XML output

Return value

True if the connection string has been setup correctly, otherwise ***False*** (in which case *p_error* is populated with the error message). If the optional *open_datafile* parameter has been passed and is not an empty string, it will only return ***True*** if the data file has been opened successfully.

Description

Sets up the necessary connection parameters for communicating with Moneyworks (*p_connect* and *p_application* are then used in most of the other component methods). The first five parameters are common for all connection types, and the remaining parameters depend on the connection type (three for Gold local, five for Gold server, and eight for DC or DC REST server).

The last two parameters (*open_datafile* and *xml_output*) are optional. If *open_datafile* is not an empty string, it will attempt to open the data file, otherwise it will just create the connection string. If *xml_output* is not an empty string, it will use the -x parameter to call the Moneyworks command-line (rather than -q). This gives more descriptive error messages in many cases e.g. when importing data. Note that if you use XML output, the result string will lose any tab or newline characters that you might have included in the *format* parameter e.g. when exporting data.

mwks_run_command

(connect_string; command_string; application_string; p_result; p_error {; blob_POST}) → Boolean

Parameter	Type	Description
connect_string	String	Connect string

command_string	String	Moneyworks command string
application_string	Pointer to string	Moneyworks application string
p_result	Pointer to string	For the result string
p_error	Pointer to string	For the error string
blob_POST	Blob	Optional - POST data for REST (as XML)

Return value

True if successful, otherwise ***False*** (in which case *p_error* is populated with the error message).

Description

Generic method to run any Moneyworks command (used in the Execute button on the example form). The first two parameters are the connection parameters from ***mwks_setup_connection*** (*p_connect* and *p_application*). The result of running the command is passed back in *p_result.* Note that for REST connections, the command must be formatted using the appropriate URL and HTTP escape characters.

mwks_evaluate

(connect_string; application_string; p_result; p_error; expression) → Boolean

Parameter	Type	Description
connect_string	String	Connect string
application_string	String	Moneyworks application string
p_result	Pointer to string	For the result string
p_error	Pointer to string	For the error string
expression	String	Expression to be evaluated

Return value

True if successful, otherwise ***False*** (in which case *p_error* is populated with the error message).

Description

Evaluate the given expression. The first two parameters are the connection parameters from ***mwks_setup_connection*** (*p_connect* and *p_application*). The result of evaluating the expression is passed back in *p_result.*

Note that, unless it is a REST connection, single quotes should not be used in the *expression* parameter as they are used internally to delimit the command-line parameters sent to Moneyworks. For example, setting *expression* to:

```
"Replace("+Char(Double quote)+"Product.custom3"+Char(Double quote)+",
```

```
`Custom4="+Char(Double quote)+"web"+Char(Double quote)+"` , `"+Char(Double quote)+"mod"+Char(Double quote)+"`)")
```

will set the product custom3 field to “mod” if custom4 field is “web”. This is equivalent to the direct Moneyworks command line statement:

```
evaluate expr='Replace("Product.custom3", `custom4="web"`,`"mod"`)'
```

Note that for REST connections, the *expression* parameter will automatically be URL encoded using standard HTTP escape sequences.

mwks_import_file

(connect_string; application_string; p_result; p_error; file_path; import_map {; extra_parameters}) → Boolean

mwks_import_text

(connect_string; application_string; p_result; p_error; import_text; import_map {; extra_parameters}) → Boolean

mwks_import_arrays

(connect_string; application_string; p_result; p_error; p_text_array; import_map {; extra_parameters} {; b_debug}) → Boolean

Parameter	Type	Description
connect_string	String	Connect string
application_string	String	Moneyworks application string
p_result	Pointer to string	For the result string
p_error	Pointer to string	For the error string
import_map	String	Full path to Moneyworks import map (see below for REST connections)
file_path	String	Full path to file to be imported
import_text	String	Text to be imported
p_text_array	Pointer to string array	Array of individual field data values to be imported
extra_parameters	String	Advanced import options - see below
b_debug	Boolean	Optional - for REST connections only - see below

Return value

True if successful, otherwise ***False*** (in which case *p_error* is populated with the error message).

Description

There are three different ways to import data, depending on whether the data is being imported from a file, a pre-prepared text string, or an array of individual field strings (which are combined internally into a single string) The first two parameters are the connection parameters from ***mwks_setup_connection*** (*p_connect* and *p_application*). The result of the import operation is passed back in *p_result.*

There is no metacharacter expansion when importing from text (or from an array of text), so you can't use \t, \n etc.. If you have multiple lines to import, write your data to a temporary file and import from the file instead. Output is a report of number of records created and updated (except for User, Contact and Build files, which will always report zero). For non-REST connections, note also that single quotes should not be used in *import_text*.

Additional arguments which can be combined into *extra_parameters* may be:

- filename='Account | User | Build | Memo' :- import into fixed format file (import map ignored)
- update='true' seqnum='num' {discard='true'} :- for transactions only. Modify the identified invoice rather than create a new one (default). Optionally discard imported invoice (i.e. just delete or cancel invoice being modified)
- return_seq='true' :- Output will be the sequence number of newly imported record
- post='true' :- For transactions only - post the imported transactions
- post seqnum='sequence number' :- Post the transaction identified by sequence number

For REST connections, standard Moneyworks import maps cannot be used and so the *import_map* parameter is ignored for ***mwks_import_file*** and ***mwks_import_text***. However, ***mwks_import_arrays*** uses a pseudo import map which uses placeholders to insert the array data into in a suitably formatted XML file which is then imported into Moneyworks. If the *b_debug* flag is set, this generated import file will be saved in the component's Resources\Moneyworks_import folder, so that you can import it manually later. In the example below, it inserts the 2nd array element into <*ourref*>, and it will split array element 17 (using *GS_ASCII_Code* as the separator) into one or more <*detail.stockcode*> values, with separate detail lines for each. Obviously, all of the detail line array elements should have the same number of separators!

```
<?xml version="1.0"?>
<table name="Transaction" count="1" start="0" found="1">
    <transaction>
        <ourref>{{{2}}}</ourref>
        <transdate>{{{4}}}</transdate>
        <user3>{{{8}}}</user3>
        <user2 calculated="true">if (user3="USD",1,if (user3="GBP",2,if
(user3="AUD",3,if (user3="EUR",4,if (user3="JPY",5,6)))))</user2>
        <namecode>{{{1}}}</namecode>
        <description>{{{16}}}</description>
        <currency calculated="true">user3</currency>
        <subfile name="Detail">
            <detail>
                <detail.taxcode>{{{22}}}</detail.taxcode>
```

```
                <detail.gross>{{{20}}}</detail.gross>
                <detail.tax work-it-out="true" />
                <detail.net calculated="true">detail.gross-
detail.tax</detail.net>
                <detail.stockcode>{{{17}}}</detail.stockcode>
                <detail.account work-it-out="true" />
            </detail>
        </subfile>
        <user1 />
        <user2>{{{11}}}</user2>
        <user3 />
        <gross work-it-out="true" />
    </transaction>
</table>
```

mwks_export

(connect_string; application_string; p_result;p_error; table;format; search; output_file {;extra_parameters}) → Boolean

$	**Parameter**	**Type**	**Description**
1	connect_string	String	Connect string
2	application_string	String	Moneyworks application string
3	p_result	Pointer to string	For the result string
4	p_error	Pointer to string	For the error string
5	table	String	Moneyworks table name
6	format	String	Format string (if blank, use default export format)
7	search	String	Search string (if blank, include all records)
8	output_file	String	Full path to output file (if blank, return in *p_result*)
9	extra_parameters	String	Advanced export options - see below

Return value

True if successful, otherwise ***False*** (in which case *p_error* is populated with the error message).

Description

Export data from the given file (“table”) in MoneyWorks. The first two parameters are the connection parameters from ***mwks_setup_connection*** (*p_connect* and *p_application*). The result of the export operation is passed back in *p_result* or written to *output_file* if supplied.

The *table* parameter may include a sort specifier (e.g. “Name.Code-”).

If the *search* parameter is an empty string, you will get all records, otherwise only those matching the search expression. Using “=” as the search expression will export a single “record” containing just the field names for the file.

To export with a particular format, use the *format* parameter. Everything in the format string is returned verbatim except for anything inside […] which is treated as an expression which can reference the fields of the file being exported. Thus if you want tab-delimited, then put tabs between the expressions. You can use metacharacters \t \r \n \xHH (hex) or \\. e.g. “[Code],[Phone]\n”. Using the *format* parameter is strongly recommended, since default export formats are subject to change.

For non-REST connections, note that single quotes should not be used in the *table*, *format*, or *search* parameters as they are used internally to delimit the corresponding command-line parameters sent to Moneyworks. You can use double quotes instead e.g. $t_search:=“Custom4=” + Char(Double Quote) + “web” + Char(Double Quote).

If the *output_file* parameter is not blank, a file is created using the full path supplied and the data is written there. Note that, in this case, the result string will be empty. If you supply an *output_file* parameter with an empty *format* parameter, you will get the format equivalent to a manual export. Note that if the *output_file* parameter is blank and you have used the XML switch for the Moneyworks connection, tabs and other white space will be stripped from the result string.

mwks_data_to_arrays

(result_string; p_arrays)

Parameter	Type	Description
result_string	String	Result string from mwks_export()
p_arrays	Pointer to 2D array	For an array of text arrays to hold results

Description

Process exported data from MoneyWorks. Create a 2D array of zero-element text arrays (one for each field to be returned) and pass a pointer to it as *p_arrays*. See example methods.

mwks_report

(connect_string; application_string; p_result; p_error; report_path; format; from; to; output_file {;extra_parameters}) → Boolean

Parameter	Type	Description
connect_string	String	Connect string
application_string	String	Moneyworks application string
p_result	Pointer to string	For the result string
p_error	Pointer to string	For the error string
report_path	String	Full path to report
format	String	'text','pdf' or 'html' (default to 'text' if blank)
from	String	From date string (may be blank)
to	String	To date string (may be blank)
output_file	String	Full path to output file (if blank, return in *p_result*)
extra_parameters	String	Advanced report options - see below

Return value

True if successful, otherwise ***False*** (in which case *p_error* is populated with the error message).

Description

Executes the given report. The first two parameters are the connection parameters from ***mwks_setup_connection*** (*p_connect* and *p_application*). The result of the report is passed back in *p_result* or written to *output_file* if supplied.

You should either supply both *from* and *to* as date strings (e.g. '01/01/2010') to specify a range for the report, or leave them both blank.

For non-REST connections,, note that single quotes should not be used in the *format* parameter, as they are used internally to delimit the command-line parameters sent to Moneyworks.

If *output_file* is not specified, or the *format* parameter is blank, the format is tab-delimited (i.e. 'text'). If an *output_file* is given and the filename ends with ".html", or *format* is 'html' then the output will be in HTML.

All parameters are inserted into the name table and are available to the report. Thus you can pass values for custom controls defined for a report in *extra_parameters*.

mwks_form

(connect_string; application_string; p_result; p_error; form_path; format; search; output_file {;extra_parameters}) → Boolean

Parameter	Type	Description
connect_string	String	Connect string
application_string	String	Moneyworks application string
p_result	Pointer to string	For the result string
p_error	Pointer to string	For the error string
form_path	String	Full path to form
format	String	'pdf' (default to 'pdf' if blank)
search	String	Search string (required unless form type is .rept)
output_file	String	Full path to output file (if blank, return path in *p_result*)
extra_parameters	String	Advanced form options - see below

Return value

True if successful, otherwise ***False*** (in which case *p_error* is populated with the error message).

Description

Only available in Moneyworks v6. Renders the form to a pdf file. The first two parameters are the connection parameters from ***mwks_setup_connection*** (*p_connect* and *p_application*). The output of the form is written to *output_file* if supplied, otherwise a file is created in the temp directory and *p_result* returns the path to the generated file.

For form types other than .rept the *search* parameter must be provided to select the record(s) to use. If multiple records are selected, all output still goes to the same file.

The only supported *format* at present is “pdf”.

For non-REST connections, note that single quotes should not be used in the *search* parameter as they are used internally to delimit the command-line parameters sent to Moneyworks. You can use double quotes instead e.g. $t_search:=“Custom4=” + Char(Double Quote) + “web” + Char(Double Quote).

For forms that use variables supplied by the form printing dialog (such as Message, Print_Copy etc) you may provide values for those variables by passing them in *extra_parameters* e.g. Message='Hello' Print_Copy='0' Include_Remit='0' for typical invoice forms; Stmt_Date='1/1/2010' for statements.

Migrating from previous Moneyworks 4D plugin

The main differences are:

- The connection to Moneyworks is handled differently - most methods take two extra parameters (*connect_string* and *application_string*) to pass the connection details.
- The return value is usually Boolean rather than text - text return values are passed via a string pointer instead (*p_result).*
- There is no separate method (***MWErrors***) to return errors - these are returned via a string pointer instead (*p_error*).
- There is no equivalent of ***MWLookup*** - this can be handled by ***mwks_evaluate*** using the Moneyworks Lookup command (see the “Calculations and things” section of the Moneyworks manual).

Taking into account the extra (four in most cases) parameters, the methods can be mapped as follows:

- Connecting to Moneyworks: ***mwks_setup_connection*** is roughly equivalent to a combination of ***MWConnect*** and ***MWOpen***.
- Evaluating expressions: The *expression* parameter in ***mwks_evaluate*** is equivalent to the *EvalStr* parameter in ***MWEvaluate***.
- Importing: The *import_map* parameter in ***mwks_import_file*** and ***mwks_import_text*** is equivalent to the *theMap* parameter in ***MWImportFile*** and ***MWImportText*** respectively. The *file_path* parameter in ***mwks_import_file*** is equivalent to the *theFile* parameter in ***MWImportFile***. The *import_text* parameter in ***mwks_import_text*** is equivalent to the *theText* parameter in ***MWImportText***.
- Exporting: The *table*, *search* and *output_file* parameters in ***mwks_export*** are equivalent to the

logicalFile, *searchString* and *destination* parameters in ***MWExport*** respectively.

- Reporting: The *report_path*, *from*, and *to* parameters in ***mwks_report*** are equivalent to the *theReport*, *FromPeriod*, and *ToPeriod* parameters in ***MWDoReport*** respectively. The *output_file* parameter in ***mwks_report*** is a combination of the *OutputTo* and *FileName* parameters in ***MWDoReport***. There is no equivalent of the *Interact* parameter in ***MWDoReport***.

From:
https://docs.sussol.net/ - **Sussol Docs**

Permanent link:
https://docs.sussol.net/doku.php/4d_moneyworks_component?rev=1475514014

Last update: **2016/10/03 17:00**