Manual

External Control for WinOLS

(as of: 09.09.2026)

Contents

1 PRELIMINARY NOTES 5
1.1 What is External Control for WinOLS? 5
1.2 What is LUA? 5
1.3 Lua in a nutshell 5
1.4 Requirements 5
1.5 Security advice 6
1.6 WinOLS Configuration 6
1.7 Starting LUA 6
1.8 Samples and server usage 6
1.9 WinOLSScript 7

2 LANGUAGE FEATURES 8
2.1 Notes 8
2.1.1 Return values 8
2.1.2 Filenames 8
2.2 Global functions 8
2.2.1 TextEntryDialog (old) 8
2.2.2 TextEntryDialog (new) 8
2.2.3 MessageBox 9
2.2.4 fromhex 9
2.2.5 tohex 10
2.2.6 fromJSON 10
2.2.7 toJSON 10
2.2.8 HttpStart 10
2.2.9 HttpAddHeader 10
2.2.10 HttpAddParam 10
2.2.11 HttpAddFile 11
2.2.12 HttpExecute 11
2.2.13 HttpResponseError 11
2.2.14 HttpResponseString 11
2.2.15 HttpResponseFile 11
2.2.16 binaryor 12
2.2.17 binaryxor 12
2.2.18 binaryand 12
2.2.19 Log 12
2.2.20 Quit 12
2.2.21 SaveAll 13
2.2.22 CloseAll 13
2.2.23 SetClient 13
2.2.24 GetClient 14
2.2.25 NewProject 14
2.2.26 GetVersion 14
2.2.27 Sleep 14
2.2.28 FindProjects 14
2.2.29 FindProjects2 15
2.2.30 DuplicateProject (Filename) 15
2.2.31 GetProjectVersions 16
2.2.32 OpenProjectVersion 16
2.2.33 DeleteProjectVersion 16
2.2.34 OpenAndExport 17
2.2.35 ReactivateChecksums 17
2.2.36 StartUrl 17

2.2.37 ReadDirectory 17
2.2.38 CreateDirectory 18
2.2.39 MessagePump 18
2.2.40 RemoveNonFilenameCharacters 18
2.2.41 GetLastError 18
2.2.42 timeGetTime() 18
2.2.43 requirex 19
2.2.44 usestrict 19
2.3 Context philosophy 19
2.3.1 SetThirdPartyConfig 19
2.4 Context: Project 20
2.4.1 projectGetProperty 20
2.4.2 projectSetProperty 20
2.4.3 projectClose 21
2.4.4 projectSave 21
2.4.5 projectExport 21
2.4.6 projectExportMaps 22
2.4.7 projectImport 22
2.4.8 projectMail 23
2.4.9 projectSearchChecksums 23
2.4.10 projectApplyChecksums 24
2.4.11 projectAddChecksum 24
2.4.12 projectStatChecksums 24
2.4.13 projectGetChecksumName 25
2.4.14 projectGetChecksumOptionStatus 25
2.4.15 projectGetChecksumOptionText 25
2.4.16 projectGetChecksumOptionType 25
2.4.17 projectSetChecksumOptionStatus 26
2.4.18 projectGetElementOffset 26
2.4.19 projectGetElement 26
2.4.20 projectSetElement 26
2.4.21 projectGetElementRanges 27
2.4.22 projectSetElementRanges 27
2.4.23 projectGetDataAreas 27
2.4.24 projectAddCommentAt 27
2.4.25 projectGetCommentAt 27
2.4.26 projectDelCommentAt 28
2.4.27 projectGetAt 28
2.4.28 projectSetAt 28
2.4.29 projectSetOrg 29
2.4.30 projectFindBytes 29
2.4.31 projectReplaceBytes 30
2.4.32 projectFindSimilarProjects 30
2.4.33 projectFindSimilarProjectsSql 30
2.4.34 projectImportFromOls 31
2.4.35 projectFindMap 32
2.4.36 projectAddMap 32
2.4.37 projectDelMap 32
2.4.38 projectImportCsvJson 32
2.4.39 projectImportMapPack 32
2.4.40 projectSetRight 33
2.4.41 projectSetRightsOwner 33
2.4.42 projectCountDifferentBytes 33
2.4.43 projectDelFolder 34
2.4.44 projectDelDuplicateMaps 34
2.4.45 projectAddrCpuToRaw 34
2.4.46 projectAddrRawToCpu 34

2.4.47 projectImportChanges 34
2.4.48 projectAutoUpdate 35
2.4.49 projectAutoImport 35
2.4.50 projectSearchVehicleData 36
2.4.51 projectCloneVehicleData 36
2.4.52 projectGetQuickFixState 36
2.4.53 projectSetQuickFixState 37
2.4.54 projectGetQuickFixes 37
2.4.55 projectImportMapFile 37
2.5 Context: Version 38
2.5.1 versionGetProperty 38
2.5.2 versionSetProperty 38
2.6 Context: Window 39
2.6.1 windowGetActive 39
2.6.2 windowSetActive 39
2.6.3 windowGetMapProperties 39
2.6.4 windowSetMapProperties 40

3 INDEX 41

1 Preliminary notes

1.1 What is External Control for WinOLS?

WinOLS has manifold methods to read data from ECUs, to edit them and write them back. The plugin "External Control" allows you to control several of these functions via scripts. For this, the plugin brings an entire programming language along, namely "LUA". This allows you to automate repetitive data processing steps.

Note: The plugin "External Control" has nothing to do with the scripts functions that are already integrated in WinOLS (*.winolsskript) in common. It is a lot more powerful.

1.2 What is LUA?

LUA is an existing programming language which was integrated into WinOLS. This manual documents the WinOLS functions which can be accessed by LUA. For a documentation of the language LUA itself, please refer to the LUA homepage:
http://www.lua.org/

1.3 Lua in a nutshell

Including a library	dofile ("library.lua");
Line comment	--Comment text
Concatenate strings	MessageBox("s="..st);
Strings equal / not equal	if (str=="equal" and str~="unequal") then
Number of array elements	size = #myarray
For / Next loop	for i=1, size do MessageBox ("myarray ["..i.."]".."=".. myarray [i]); end
Helpful function (see library.lua) for var. content	MessageBox_r(a);
Replace a string	s = string.gsub(s, "needle", "replacement");
Read a text file (one line)	INPUT = io.open("myfile.txt", "rb") inputfile = chomp(INPUT:read("*line")) INPUT:close();
Write a text file	OUTPUT = io.open ("myfile.txt", "w") OUTPUT:write ("my content"); OUTPUT:close();
Delete a file	os.remove ("myfile.txt");

1.4 Requirements

To use the plugin "External Control" you need:

- A registered WinOLS version (At least Version 1.219)
- A registered version of the "External Control" plugin
  (not included in the WinOLS license)

Please note:
To use the “External Control” plugin on a webserver you’ll need your own Windows webserver. The reason is that the WinOLS functions may take up a lot of computing power. Using it in a time-sharing system with other servers or in a process with a time limit would cause problems.

1.5 Security advice

LUA contains functions for the file management. This includes functions to delete files. Please do not use WinOLS to run LUA scripts if they do not come from a trusted source. When running it on a server, please make sure that scripts can only be entered or modified by authorized persons.

[Note: Access to the io library is disabled for solutions]

1.6 WinOLS Configuration

Due to performance reasons, you should configure WinOLS on your server, so that the following options are disabled:

- Search potential maps (Background)
- Generate overview information (Background)
- Suggest scripts, if applicable (When importing)

Normally you’ll want the following options to be enabled, or you might experience problems with your scripts:

- Search and accept vehicle data (When importing)
- Search checksums (When importing)

1.7 Starting LUA

You have 3 options to start a LUA Script:

- Use the complete path and filename as a parameter when starting WinOLS
- Drag the LUA file by Drag + Drop into the WinOLS frame window.
- Drag the LUA file by Drag + Drop into a WinOLS project window.

The first option requires that WinOLS currently isn’t running. If WinOLS is currently running (no matter if it’s idle or running a script), the start will be ignored – but the script will be executed in the currently running WinOLS if possible.

The third option will set the target window as default project (context) for the script. You’ll find more about this under “Context philosophy”.

1.8 Samples and server usage

At this address you can find sample programs for the use of LUA in WinOLS:
https://www.evc.de/en/product/ols/lua.asp

The use as server is worth mentioning. In this mode the script isn’t simply applied to the current project. Instead the script is running permanently since program start and reacts via “ticket” files to requests from outside, perhaps from a webserver.

1.9 WinOLSScript

You can combine LUA and WinOLSScript by using projectImport to import a WinOLSScript file into the current project.

2 Language features

This manual documents the WinOLS functions which can be accessed by LUA. For a documentation of the language LUA itself, please refer to the LUA homepage:
http://www.lua.org/

2.1 Notes

2.1.1 Return values

Some functions return a boolean value to show if they were successful. Unless documented differently, these functions return “TRUE” (equals 1) when successful and “FALSE” (equals 0) when not successful.

2.1.2 Filenames

When passing a filename to a function you should (if possible) always use the filename and the complete path, since WinOLS cannot guarantee that you always have the same current working directory.

2.2 Global functions

Global functions are not member of a certain context. They work independently of the current project or window.

2.2.1 TextEntryDialog (old)

Syntax: TextEntryDialog (int mode, String WindowTitle, String Description, String DefaultValue="")
Return value: String

This function allows it to display a window and query a string. The value entered by the user is returned as string. (Or an empty string if cancel was clicked.)

The following modes are supported:
eTextEntrySmall: A compact dialog
eTextEntryPassword: Same, but in password mode (stars instead of characters)

Example:
value = TextEntryDialog(eTextEntrySmall, "my title", "my description", "my default value");
MessageBox (value);

2.2.2 TextEntryDialog (new)

Syntax: TextEntryDialog (String WindowTitle, int mode, String Description, String DefaultValue, …)
Return value: Array

In this mode, the TextEntryDialog function is more flexible. The 3 parameters mode/Description/DefaultValue can be repeated several times to query several things simultaneously.

The mode supports the following values:
eTextEntryEdit: Text input field
eTextEntryPassword: Same, but in password mode (asterisks instead of letters)
eTextEntryCheckbox: A checkbox. The DefaultValue should be 1 or 0.

The dialog may contain up to 10 text input fields and 20 checkboxes. If successful, an array with the entered values is returned. If the user aborts the dialog, an array with the length 0 is returned.

Example:
a = TextEntryDialog("my title", eTextEntrySmall, "my description", "my default value", eTextEntryPassword, "my description 2", "secret", eTextEntryCheckbox, "my checkbox 1", "1", eTextEntryCheckbox, "my checkbox 2", "1");

```
value = ""
size = #a
for i=1, size do
  value = value..a[i].."\n";
end

MessageBox (value);
```

2.2.3 MessageBox

Syntax: MessageBox (String Text, Number Type = MB_OK)
Return value: Number

This function allows you to display a window and thus reflects the Windows function of same denominator. It is great for testing, but should be avoided on server as it will pause the process until the window is closed by the user.

Type can be entered as the binary combination of the following values:
MB_ICONERROR, MB_ICONQUESTION, MB_ICONWARNING, MB_ICONINFORMATION, MB_ABORTRETRYIGNORE, MB_OK, MB_OKCANCEL, MB_RETRYCANCEL, MB_YESNO, MB_YESNOCANCEL

The return value can be one of the following values: IDYES, IDNO, IDCANCEL, IDOK, IDABORT, IDRETRY, IDIGNORE.

Note: LUA does not support a “binary or”. Use the function “binaryor” instead.

Example 1: MessageBox ("Test");
Example 2: rc = MessageBox ("Test. Variable c="..c.. " Continue?“, binaryor(MB_YESNO, MB_ICONINFORMATION));

2.2.4 fromhex

Syntax: fromhex (string HexNumberAsString)
Return value: Number

Converts the given hex number (a string parameter) into a number

Example: adr = fromhex("C000”);

2.2.5 tohex

Syntax: tohex (int number)
Return value: String

Converts the parameter into a hex string

Example: MessageBox(tohex(49152));

2.2.6 fromJSON

Syntax: fromJSON (string strJSON)
Return value: Object

Receives a string with JSON data and creates a LUA object from it.

Example: myObj = fromJSON("{ \"n3\":[\"Test\", 1.23], \"n1\":true, \"n2\":42 }");

2.2.7 toJSON

Syntax: toJSON (Object data)
Return value: String

Receives a LUA object and generates a JSON string from it.

Example: MessageBox (toJSON({n1=true, n2=42, n3={"Test", 1.23 } }));

2.2.8 HttpStart

Syntax: HttpStart (string url, string verb=”GET”)
Return value: bool

First command to initiate an http request.

Example: rc = HttpStart("https://example.com"); -- true=Erfolg

2.2.9 HttpAddHeader

Syntax: HttpAddHeader (string name, string value)
Return value: bool

Adds a header to the http call. (optional)

Example: HttpAddHeader("content-type", "multipart/form-data"); -- true=Erfolg

2.2.10 HttpAddParam

Syntax: HttpAddParam (string name, string value)
Return value: bool

Adds a parameter to the http call. (optional)

Example: HttpAddParam("myParam", "myValue"); -- true=Erfolg

2.2.11 HttpAddFile

Syntax: HttpAddFile (string name, string path_and_filename)
Return value: bool

Adds a file as a parameter to the http call. (optional)

Example: HttpAddFile("myTransferFilename", "x:\\myPath\\MyFile.dat"); -- true=Erfolg

2.2.12 HttpExecute

Syntax: HttpExecute ()
Return value: Number

Executes the http call. Returns the http status code as a number. If successful, usually 200. Your HttpAdd… calls must be executed before this command. Your HttpResponse… calls afterwards.

Example: rc=HttpExecute ();

2.2.13 HttpResponseError

Syntax: HttpResponseError ()
Return value: String

Returns the http status text as a string. If successful, usually "OK".

Example: rc=HttpResponseError ();

2.2.14 HttpResponseString

Syntax: HttpResponseString ()
Return value: String

Returns the result of the http call as a string.

Example: rc=HttpResponseString ();

2.2.15 HttpResponseFile

Syntax: HttpResponseFile (string path_and_filename)
Return value: bool

Saves the result of the http call as a file.

Example: rc=HttpResponseFile („x:\\myPath\\MyOutputFile.dat“); -- true=Erfolg

2.2.16 binaryor

Syntax: binaryor (Number param1, Number param2, ...)
Return value: Number

Since LUA does not support a “binary or“ like C, this function was introduced. It supports any number of parameters. In C it corresponds to the code "param1|param2". Some WinOLS LUA functions allow several flags to be combined with binaryor.

Example: rc = MessageBox ("Test. Variable c="..c.. " Continue?“, binaryor(MB_YESNO, MB_ICONINFORMATION));

2.2.17 binaryxor

Syntax: binaryxor (Number param1, Number param2, ...)
Return value: Number

Since LUA does not support a “binary x-or“ like C, this function was introduced. It supports any number of parameters. In C it corresponds to the code "param1^param2".

Beispiel: rc = MessageBox (binaryxor(3,4)); --Result: 7

2.2.18 binaryand

Syntax: binaryand (Number param1, Number param2, ...)
Return value: Number

Since LUA does not support a “binary and“ like C, this function was introduced. It supports any number of parameters. In C it corresponds to the code "param1&param2".

Beispiel: rc = MessageBox (binaryand(3,7)); --Result: 3

2.2.19 Log

Syntax: Log (string)
Return value: -

Write a string into the WinOLS logfile (and the service console, if used).

Example: Log("test");

2.2.20 Quit

Syntax: Quit ()
Return value: -

This function terminates WinOLS.

Notes:

- All unsaved changes will be lost. Use the function SaveAll() first, if you don’t like that.
- All open projects will be restored during the next start. This isn’t a good idea for the automated

use, since the number of projects will grow and use up memory. Use the function CloseAll() before terminating WinOLS.
- If you run a WinOLS with a LUA script as parameter and expect WinOLS to close after running the script then script must end with this function.

Example: Quit();

2.2.21 SaveAll

Syntax: SaveAll ()
Return value: -

This function saves all changes in open WinOLS projects. If an original version with changes in the hexdump is saved, then a new version with the name “CreatedByLua” will be created. If a version with changes in the hexdump is saved, then the same version will be overwritten.

Example: SaveAll ();

2.2.22 CloseAll

Syntax: CloseAll ()
Return value: -

This function closes all open WinOLS projects. Unsaved changes will be lost.

Notes:
- All open projects will be restored during the next start. This isn’t a good idea for the automated use, since the number of projects will grow and use up memory. Use the function CloseAll() before terminating WinOLS.

Example: CloseAll ();

2.2.23 SetClient

Syntax: SetClient(string clientname)
Return value: bool (true for success)

Syntax: SetClient()
Return value: string (current value)

Changes the current client. This function influences subsequent calls to NewProject, FindProjects, etc. It has no influence on projects that are already created / open.

Instead of a client name, you can also provide a reseller name. This allows you to access the reseller’s project list, e.g. using projectFindSimilarProjects (but you can’t purchase using LUA). Please note that you MUST set the client back to a regular, local client before the next project is created.

If no parameter is passed, the current client is returned.

Beispiel: SetClient ("test");

2.2.24 GetClient

Syntax: GetClient(bool bReturnPath=false)
Return value: string

Returns the currently active client or (if true is passed as a parameter) its path.

Example: var path = GetClient (true); -- path is without backslash at the end

2.2.25 NewProject

Syntax: NewProject()
Return value: -

This function creates a new, empty WinOLS project. This new project (and window) will be used as active project (and window).

Example: NewProject ();

2.2.26 GetVersion

Syntax: GetVersion (Number VersionId)
Return value: Number

This function returns the versions number (major or minor) from WinOLS or from the “External Control” plugin. The parameter “VersionId” must have one of the following values: eWinOLSMajor, eWinOLSMinor, ePluginMajor, ePluginMinor.

Use this function to maintain compatibility for future extensions or changes.

Example: MessageBox ("You're using WinOLS "..GetVersion(eWinOLSMajor).."."..
GetVersion(eWinOLSMinor));

2.2.27 Sleep

Syntax: Sleep (Number Milliseconds)
        Sleep (Number Milliseconds, String pathAndWildcards)
Return value: -

This function lets WinOLS “sleep” for a defined time, so that neither computing power is used up, nor actions can be performed by WinOLS. Use it to check data at regular intervals without using up too much computing power. The parameter is defined in 1/1000 seconds.

If you pass a path including file name with wildcards as second parameter, the function will return immediately if such a file exists / is created or if WinOLS is getting closed.

Example 1: Sleep (5000);
Example 2: Sleep (5000, "x:\\test*.ticket");

2.2.28 FindProjects

Syntax: FindProjects (String Producer, String Chassis, String Model, String md5Cpu, String md5Eprom, Number ProjectStatus, string Softwareversion)

Return value: Array

This function returns an array with all filenames of projects that match the search criteria. Valid values for the ProjectStatus are the values ePrjDeveloping, ePrjFinished, ePrjMaster or 0.

Use the empty string (""), or the number 0 to show that this criterion should be ignored.

It's recommended to use the ProjektStatus to avoid confusion with temporarily created projects.

Example:
```
a = FindProjects("","","","1253b4de81311b81c05a623eaa5781ff", "", ePrjMaster, "");
size = #a
for i=1, size do
  MessageBox ("a["..i.."]".."="..a[i]);
end
```

2.2.29 FindProjects2

Syntax: FindProjects2 (Number MaxNumberOfResults, string GeneralSearch, Number idSearchColumn1, String Column1Value, Number idSearchColumn2, String Column2Value, Number idDesiredColumn1, Number idDesiredColumn2, …)

Return value: array

This function searches the current client for projects that meet the 3 specified conditions:

1. GeneralSearch:
   A general search string that searches all columns. The usual tricks like spaces or minus also work here.
2. idSearchColumn1+Column1Value:
   The id of a column (e.g. ePrjPropVehicleProducer) and the text value that in must contain. Instead of the Id, 0 can also be used to skip this search condition. If the text value is an empty string, then only projects where this field is empty fit.
3. idSearchColumn2+Column2Value:
   Like 2.

It returns a list with all desired columns.

Example:
```
-- Search for "Test" and manufacturer=Volvo; Returns 3 data fields per project.
list = FindProjects2 (80, "test", ePrjPropVehicleProducer, "Volvo", 0, "", ePrjFilename,
ePrjPropVehicleProducer, ePrjUserdef1);
nColumns = 3;
size = #list
for i=0, size/nColumns-1 do
  MessageBox ("File "..i.."=".. list [i*nColumns+1].."\n".. list [i*nColumns+2] .."\n".. list [i*nColumns+3]);
end
```

2.2.30 DuplicateProject (Filename)

Syntax: DuplicateProject (String Filename)
Return value: string

This function creates a binary copy of the specified ols file and automatically creates a new file name in the current client folder. The new path+filename is returned.

Example:
strNewFilename = DuplicateProject ("prj1000.ols"); --Diese Datei muss bereits existieren

2.2.31 GetProjectVersions

Syntax: GetProjectVersions (String Filename)
Syntax: GetProjectVersions (String Filename, int versionproperties, …)
Return value: Array

This function returns an array with all version names and descriptions of a given ols project file. The array contains alternately name and description (starting with index 1 for name). The name contains twice as much entries as the project has versions.

Alternatively, you can pass any number of version properties (see versionGetProperty). Then these properties are returned instead of the ones mentioned above.

Example:
```
versions = GetProjectVersions ("prj1000.ols");  --this file needs to exist already
size = #versions
for i=1, size/2 do
  MessageBox ("Version "..i.."=".. versions [i*2-1].."\n".. versions [i*2]);
end
```

2.2.32 OpenProjectVersion

Syntax1: OpenProjectVersion (string filename, string versionname)
Syntax2: OpenProjectVersion (string filename, number versionindex)
Return value: bool (true for success)

Opens the given project in the desired version. The project must be closed later with projectClose or CloseAll.

versionname: Name of the desired version. If not unique, the one with the smallest index will be used.
versionindex: 0 for original, 1 for first version…

Note: The *type* is important for parameter 2. If you have string (from a file) that contains a number (as text), it's a version name. Use "1*mystring" to use it as version index.

Example: OpenProjectVersion ("prj1000.ols", "+10%");

2.2.33 DeleteProjectVersion

Syntax1: DeleteProjectVersion (string filename, string versionname)
Syntax2: DeleteProjectVersion (string filename, number versionindex)
Return value: bool (true for success)

Like OpenProjectVersion, but the version is not opened, but deleted.

Note: Versions that are currently open cannot be deleted.

Note: Deleting a version will cause the following version numbers to shift. If you want to delete multiple versions, use the version names or delete by descending size numbers.

Example: DeleteProjectVersion ("prj1000.ols", "+10%");

2.2.34 OpenAndExport

Syntax: OpenAndExport string filename, string versionname, String Filename, Number Typ, String ZippedFileiname="")
Syntax: OpenAndExport string filename, number versionindex, String Filename, Number Typ, String ZippedFileiname="")

Return value: bool (true for success)

Combines the functions OpenProjectVersion, projectExport and projectClose. Parameters like for the listed functions.

Example: OpenAndExport ("prj1000.ols", "+10%", "%HOMEPATH%\\Desktop\\file.dat", eFiletypeBinary, "c:\\myfiles\\file.zip");

2.2.35 ReactivateChecksums

Syntax: ReactivateChecksums ()
Return value: -

If a checksum plugin causes an exception while searching or calculating WinOLS automatically disables it until WinOLS is restarted to protect the use against problems. If you program want to write a script to automatically search for checksum you might want to call this function after each finished project to re-enable all plugins.

Example: ReactivateChecksums ();

2.2.36 StartUrl

Syntax: StartUrl (string url)

Opens a browser window with the requested URL.

Example: StartUrl ("http://www.google.com/");

2.2.37 ReadDirectory

Syntax: ReadDirectory (string filenameandwildcards)

Returns an array with file and foldernames.

Example:

```
a = ReadDirectory ("c:\\*.txt");
size = #a
for i=1, size do
  MessageBox ("a["..i.."]".."="..a[i]);
end
```

2.2.38 CreateDirectory

Syntax: CreateDirectory (string full_path)

Returns 0 in case of an error, 1 if the folder was created and 2 if it already existed.

Example:
```
a = ReadDirectory ("c:\\*.txt");
size = #a
for i=1, size do
  MessageBox ("a["..i.."]".."="..a[i]);
end
```

2.2.39 MessagePump

Syntax: MessagePump ()

Handles pending Windows messages in the WinOLS main thread. This function should be called regularly if the LUA code takes a long time or runs in an endless loop. Otherwise Windows will show that WinOLS is not responding anymore.

2.2.40 RemoveNonFilenameCharacters

Syntax: RemoveNonFilenameCharacters (string input)
Return value: String

Takes the input string and removes characters that Windows won't accept in a filename, like "?". Returns the updated string.

2.2.41 GetLastError

Syntax: GetLastError(bool bAsString=FALSE)
Return value: Number / String

Returns the value of the Windows function with the same name. Due to automatic correction strategies, the reason for the failure of a WinOLS function does not necessarily have to be in here.

If an import fails because of NOREAD, projectImport sets the value to 30 (which corresponds to the Windows error ERROR_READ_FAULT).

If the optional parameter is set to TRUE, the function returns a string with the Windows error name for the most important common errors, e.g. "0x2=ERROR_FILE_NOT_FOUND" instead of "2.0".

2.2.42 timeGetTime()

Syntax: timeGetTime()
Return value: Number

Returns the value of the Windows function of the same name, i.e. the milliseconds since system startup.

2.2.43 requirex

Syntax: requirex(String filename)
Return value: boolean

Works like the normal require command, but can also handle encrypted files (see WinOLS help: luaX). It is not possible to enter a password, but if the current script has already been opened with a password, this is automatically tried for the file specified here.

Example:
```
requirex ("mylib");  -- tries mylib, mylib.lua, mylib.luax
```

2.2.44 usestrict

Syntax: usestrict ()
Return value: -

Similar to the JavaScript mode of the same name, variables must be explicitly declared. This protects against typing errors in variable names.

Example:
```
usestrict();
local myLocalVar=1          -- OK
declare ("myGlobalVar", 2) -- OK (Special syntax for global variables necessary)
myLcalVar = 2               -- Error! Note the typo. Without “usestrict()” this error might be overlooked
```

2.3 Context philosophy

Several functions use the current WinOLS state as a context to run. For example, the function “projectSave” will affect only the project that is currently selected. LUA for WinOLS works a bit like a macro recorder which replays keystrokes.

The plugin offers several functions to change different current contexts. If you want so save a project which is not the current one, you can poll the current project, change the active one, save the active one and then reactivate the old project.

If a necessary context doesn’t exist (e.g. when you’re calling projectSave() without having any projects open), the functions fails and returns an error value.

2.3.1 SetThirdPartyConfig

Syntax: SetThirdPartyConfig (Number id, String Wert)
Return value: bool

Sets the value of a configuration property of F12>ThirdParty. Reading the current value is not supported for security reasons.

Possible values for id: eThirdPartyAutotunerMasterToolId, eThirdPartyAutotunerApikey, eThirdPartyAutoflasherApikey, eThirdPartyBFlashApikey, eThirdPartyBFlashLifetime, eThirdPartyMagicmotorsportApikey, eThirdPartyMagicmotorsportXsn, eThirdPartyMagicmotorsportSn

```
SetThirdPartyConfig (eThirdPartyMagicmotorsportSn, "abc123");
```

2.4 Context: Project

2.4.1 projectGetProperty

Syntax: projectGetProperty (Number id, Number iOrgVer=0)
Return value: String

This function polls one of the many texts from the project properties. Use it e.g. to find out which checksum is currently entered for the project.

The parameter “id” must be one of the following values:

- ePrjPropClientName, ePrjPropClientNumber, ePrjPropClientLicenceplace
- ePrjPropVehicleType, ePrjPropVehicleProducer, ePrjPropVehicleChassis, ePrjVehicleBuild, ePrjPropVehicleModel, ePrjVehicleCharacteristic, ePrjPropVehicleModelyear, ePrjPropVehicleVIN
- ePrjPropEcuProducer, ePrjPropEcuBuild, ePrjPropEcuProdNr, ePrjPropEcuStgNr, ePrjPropEcuSoftwareversion, ePrjPropEcuSoftwareversionVersion, ePrjPropEcuChecksum, ePrjPropEcuSoftwaresize, ePrjPropEcuUse
- ePrjPropEngineName, ePrjPropEngineType, ePrjPropEngineDisplacement, ePrjPropEngineTransmission, ePrjPropEngineOutputPS, ePrjPropEngineOutputKW, ePrjPropEngineEmissionStd, ePrjPropEngineTorque
- ePrjPropCommunicationsReadhardware
- ePrjPropProjectType, ePrjProjectStatus
- ePrjFileCreatedBy, ePrjFileModifiedBy, ePrjFileCreatedOn, ePrjFileModifiedOn, ePrjComment
- ePrjPropNoreadTag, ePrjPropSpiTag, ePrjPropBdmTag, ePrjPropUserTag, ePrjPropUserTagText
- ePrjPropResellerCredits, ePrjPropResellerProjectType, ePrjPropResellerProjectDetails
- ePrjUserdef1, ePrjUserdef2, ePrjUserdef3, ePrjUserdef4, ePrjUserdef5, …

Furthermore the following values are allowed (for reading only): ePrjImportFilename, ePrjImportPath, ePrjFilename

As of WinOLS 5.62.01, the following project properties (for Original with 1 as the second parameter for the version) can be queried: ePrjPropChecksum8Bit, ePrjPropChecksum8BitCpu, ePrjPropChecksum8BitEpr, ePrjPropChecksumMD5, ePrjPropChecksumMD5Cpu, ePrjPropChecksumMD5Epr, ePrjPropChecksumSHA1, ePrjPropChecksumSHA256.

Note: This function returns the current values from the project properties. It does not scan the project hexdump for data.

Example1:
```
MessageBox (projectGetProperty(ePrjPropClientName));
```

Example 2: Implicit conversion of the return value into a number to allow a comparison:
```
if (ePrjDeveloping==1*projectGetProperty (ePrjProjectStatus)) then
  MessageBox ("ePrjDeveloping");
end
```

2.4.2 projectSetProperty

Syntax: projectSetProperty (Number id, String newvalue)
Return value: bool

This function changes one of the many texts from the project properties to a new value.

The parameter “id” must have one of the values like for projectGetProperty. The following values are not supported: ePrjPropEcuSoftwaresize, ePrjPropEcuChecksum, ePrjFileCreatedOn, ePrjFileModifiedOn.

Note: This function changes the project data and marks the project as changed.

Example: projectSetProperty(ePrjPropVehicleProducer, "VW");

2.4.3 projectClose

Syntax: projectClose (bool bDeleteFile=FALSE)
Return value: bool

This functions closes the active WinOLS project and all windows which belong to this project. Unsaved changes will be lost. If you use the value “TRUE” as parameter, the project is closed and delete from the harddisk.

Example: projectClose ();

2.4.4 projectSave

Syntax: Save (bool CreateNewVersion, String VersionName=”CreatedByLua”, String VersionDescription=””)
Return value: bool

This function saves the active WinOLS project, if there are any changes to save. If an original with changes is currently in the hexdump, then a version with the name “CreatedByLua” will be created. If a version with changes is currently in the hexdump, then the same version will be overwritten. You can force WinOLS to create a new version by passing TRUE as first parameter. The second and third parameter can be version name and description. They’re only observed if a new version is saved.

Beispiel: projectSave (TRUE); -- Don’t use true here. Use TRUE

2.4.5 projectExport

Syntax: projectExport (String Filename, Number Typ, String ZippedFileiname=““, Number BdmToGoFlags=0, Number olsxCustomerNumber=0, String olsxPassword=””, Number ExportFlags=0)
Return value: bool

This function exports the current project as a file. The parameter “Typ” defines the file format and must have one of the following values: eFiletypeBinary, eFiletypeWinOLS, eFiletypeWinOLSAll, eFiletypeIntelHex, eFiletypeMotorolaHex, eFiletypeBdmToGo, eFiletypeWinOLSX, eFiletypeWinOLSAllX. eFiletypeCMDSlave. If you have the required plugin, you can also use one of the following values: eFiletypeVBF, eFiletypeCMDSlave, eFiletypeFLS, eFiletypeNewGenius, eFiletypeODX, eFiletypeCFF, eFiletypeCRE, eFiletypeBFlashSlave, eFiletypeSMRF, eFiletypeAutotunerSlave, eFiletypeAutoflasherSlave, eFiletypeMagicmotorsportSlave.

If you pass a filename for the parameter “ZippedFilename”, then the exported file will be moved into that archive. You can prefix the ZippedFilename with “+” to add the file to an existing zip file.

For BdmToGo files:
The parameter „BdmToGoFlags“ can contain export options for BdmToGo files and can be defines as a combination of the following values: eBdmToGoProgramEprom, eBdmToGoProgramEprom2, eBdmToGoProgramEEprom, eBdmToGoProgramProcessor, eBdmToGoNoReimport. If areas should be compared before programming, they must be marked as comment BDM1, BDM2 or BDM3 (either by the import or by the function projectAddCommentAt).

ExportFlags:
This can be eExportRemoveChecksums or eExportActiveElementOnly (or a binaryor combination).
Note: This used to be the parameter “Bool bRemoveChecksums”, but since eExportRemoveChecksums equals 1, this is still compatible.

Of interest for plugins: SetThirdPartyConfig

Note: Just like in C, Backslashes that are within a string must be doubled in LUA, because they’d start a special character otherwise. See example.

Note: The file formats eFiletypeBinary, eFiletypeIntelHex and eFiletypeMotorolaHex always export the currently select element.

Note: The filename may contain environment variables enclosed in percent signs. WinOLS will replace these strings with their current value. Furthermore you can use placeholders to use the path / filename from the import (example 3+4). Since they aren’t always available you must specify reserve values. The import path always ends on a backslash and the import filename never includes the suffix.

Example 1: projectExport ("%HOMEPATH%\\Desktop\\file.dat", eFiletypeBinary, "c:\\myfiles\\file.zip“);
Example 2: projectExport ("c:\\myfiles\\file.BdmToGo", eFiletypeBdmToGo, false,
binaryor(eBdmToGoProgramEprom, eBdmToGoProgramEprom2, eBdmToGoProgramEEprom,
eBdmToGoProgramProcessor, eBdmToGoNoReimport));
Example 3: projectExport ("%HOMEPATH%\\Desktop**[ImportFilename**|file**]**.dat", eFiletypeBinary);
Example 4: projectExport ("**[ImportPath**|%HOMEPATH%\\Desktop**]**\\file.dat", eFiletypeBinary);

2.4.6 projectExportMaps

Syntax: projectExportmaps (String Dateiname)
Return value: bool

This function exports the maps of the current project as a csv or json file. The file name must end with .json or .csv accordingly.

Example: projectExport ("c:\\myfiles\\maps.json");

2.4.7 projectImport

Syntax: projectImport (String Filename, Number Typ=eFiletypeAuto, Number Offset=0, String olsxPassword="", Number Options=0)
Return value: bool

This function imports the given file as a version into the current project. By default, WinOLS tries to select the directory of the lua script as current directory. To be on the safe side, you should always use absolute paths (like in the example) and not relative paths (with incomplete names).

The optional parameter “Typ” defines the file format and can have one of the following values: eFiletypeAuto, eFiletypeBinary, eFiletypeWinOLS, eFiletypeIntelHex, eFiletypeMotorolaHex, eFiletypeEDX, eFiletypeBdmToGo, eFiletypeWinOLSX, eFiletypeVBF, eFiletypeCMDSlave,

eFiletypeVBF, eFiletypeFLS, eFiletypeODX, eFiletypeCFF, eFiletypeCRE, eFiletypeBFlashSlave, eFiletypeSMRF, eFiletypeAutotunerSlave, eFiletypeAutoflasherSlave, eFiletypeMagicmotorsportSlave, eFiletypeIni.

If no type is given, the value “eFiletypeAuto” will be assumed and the file type will be detected automatically. This works with most file types, including types from Import plugins and winolsskript-files. The "Options" parameter eOptionWinOLSSkriptTestOnly can be used for this file type. In this case, the script is not applied, but only checked to see whether the requires conditions are met.

The optional parameter “Offset” names an offset which moves the imported data back in the file. This parameter is used for file formats, where WinOLS would allow you to enter the offset. Furthermore, the import refers to the current element for most file formats. So, you’ll need the offset only if you don’t want to start your import at the beginning of an element. It is supported for versions only (not for the original).

The optional parameter “Options” can be eOptionCmdCutFF or eOptionCmdIgnore. It answers the confirmation when importing CMD files. (The default is eOptionCmdIgnore.)

Of interest for plugins: SetThirdPartyConfig

Note: Just like in C, Backslashes that are within a string must be doubled in LUA, because they’d start a special character otherwise. See example.

Note: New versions contain the text “Imported version” as version name. If a different name is wanted, use the functions versionGetProperty / versionSetProperty.

Note: The filename may contain environment variables enclosed in percent signs. WinOLS will replace these strings with their current value.

Example: projectImport ("%HOMEPATH%\\Desktop\\file.dat");
Example 2:

```
dir = "c:\\mySkriptFolder";
msg = "";
aFiles = ReadDirectory(dir.."\\*.winolsskript"); --Get all files
for c=1, #aFiles do
  if (projectImport(dir.."\\"..aFiles[c], 0, "", "", eOptionWinOLSSkriptTestOnly)) then
    msg = msg..aFiles[c].."\n";  --this file can be applied
  end
end
MessageBox(msg);
```

2.4.8 projectMail

Syntax: projectMail (String To, String Subject, String Message, String Filename=“”);
Return value: bool

This function sends a mail (optionally with a file as attachment) directly from WinOLS. To send the mail WinOLS uses the default MAPI application of the computer. This is the same application that is used when you’re clicking on a mail hyperlink on a website.

Example: projectMail ("me@mydomain.com", "Test", "This is a testmessage.\n\nSending Mail works.");

2.4.9 projectSearchChecksums

Syntax: projectSearchChecksums ()

Return value: bool

This function calls the checksum search in WinOLS. Depending on the configuration this may be done anyway automatically when importing the file.

The function returns TRUE, if it was possible to run the checksum search. (FALSE for example when no project was currently open). TRUE does not mean that a checksum was found. To question that property please use the function projectStatChecksums.

Example: rc = projectSearchChecksums ();

2.4.10 projectApplyChecksums

Syntax: projectApplyChecksums ()
Return value: bool

This function makes sure that all not calculated checksums get calculated. Depending on the project file this isn't always done automatically.

The function returns TRUE, if at least one checksum needs to be recalculated or false, if no changes were needed or no project file or checksums were available.

Example: rc = projectApplyChecksums ();

2.4.11 projectAddChecksum

Syntax: projectAddChecksum (Number iFrom, Number iTo, Number dataOrg, Number iCorrectAt=-1
, Boolean bAutoApply=FALSE, Number iCorrectAtTo=-1);
Return value: bool

Adds a manual (additive) checksum block. Depending on the number of parameters:
- only with display of the result
- with (optionally automatic) balancing of the result
- with (optionally automatic) balancing of the result over a range (full byte)

Example 1: projectAddChecksum (0,255, eHiLo);
Example 2: projectAddChecksum (0,255, eHiLo, 256, TRUE);
Example 3: projectAddChecksum (0,255, eHiLo, 256, TRUE, 511);

2.4.12 projectStatChecksums

Syntax: projectStatChecksums ()
Return value: bool

This function returns information about the current checksum. If the return value equals -1, then the current project does not contain any checksums or there is no current project. If the return value is bigger than 0, then this number equals the number of checksums that could not be calculated correctly. The return value 0 means that the project contains at least one checksum and all checksums were calculated correctly.

Normally any return value other than 0 should cause some attention and your script should make sure that the project doesn't get delivered in the current state.

Example: rc = projectStatChecksums ();

2.4.13 projectGetChecksumName

Syntax: projectGetChecksumName (EChkInfo ChkInfo=eChecksumName)
Return value: string

This function returns the name of the checksum found in the current project. This is the same text that is displayed in the project properties and the project list. If you pass the value “eChecksumNumber” as optional parameter, the function will not return the name of the checksum, but the number (e.g. “OLS265”). If you use “eChecksumVersion”, you’ll receive the checksum version (e.g. “2.05”).

If no checksum was found, the function returns an empty string.

Example: rc = projectGetChecksumName ();

2.4.14 projectGetChecksumOptionStatus

Syntax1: projectGetChecksumOptionStatus (Number index_or_id)
Syntax2: projectGetChecksumOptionStatus (String SearchStringWithWildcards)
Return value: bool

This function returns the status (0 or 1) of the "Checkbox" checksum option (for example, OLS285) specified by the 0-based index.

Example: rc = projectGetChecksumOptionStatus ("*Patch*“);

2.4.15 projectGetChecksumOptionText

Syntax1: projectGetChecksumOptionText (Number index_or_id)
Syntax2: projectGetChecksumOptionText (String SearchStringWithWildcards)
Return value: string

This function returns the label text of the checksum option specified by the 0-based index.

Example: rc = projectGetChecksumOptionText (0);

2.4.16 projectGetChecksumOptionType

Syntax1: projectGetChecksumOptionStatus (Number index_or_id)
Syntax2: projectGetChecksumOptionStatus (String SearchStringWithWildcards)
Return value: ECOptType

This function returns the type of the checksum option specified by the 0-based index. Possible return values are:

- eCOTNone => Not available
- eCOTCheckbox => This is a checkbox
- eCOTText => This is a text. (There is no status)
- eCOTCheckboxSearchAgain => Like eCOTCheckbox, but projectSearchChecksums must be

called again if you change the status of this checkbox.

Example: rc = projectGetChecksumOptionType ("*Patch*“);

2.4.17 projectSetChecksumOptionStatus

Syntax1: projectSetChecksumOptionStatus (Number index_or_id, Number NewValue)
Syntax2: projectSetChecksumOptionStatus (String SearchStringWithWildcards, Number NewValue)

Return value: bool

This function returns the status (0 or 1) of the "Checkbox" checksum option (for example, for OLS285) specified by the 0-based index.

Important: If the checksums is type eCOTCheckboxSearchAgain, the checksum search must be called again. (Starting with WinOLS 5.32, this is done automatically.)

Example: projectSetChecksumOptionStatus (0, 1);

2.4.18 projectGetElementOffset

Syntax: projectGetElementOffset ()
Return value: Number

This function returns the offset, i.e. the address that marks the first byte of the current elements. Often, this is 0. Checksums or manual changes at the elements may cause other result values.

Example: rc = projectGetElementOffset ();

2.4.19 projectGetElement

Syntax: projectGetElement ()
Return value: Number

This function returns the identifier of the current element. The return value is one of the following values: eElementNone, eElementHeader, eElementProcessor, eElementEprom, eElementEprom2, eElementEEprom, eElementConfiguration

Example: rc = projectGetElement ();

2.4.20 projectSetElement

Syntax: projectSetElement (Number ElementId)
Return value: bool

This function selects the currently visible element and thus the range that is used for an import or export process for several file formats. For the ElementId all values from projectGetElement are accepted.

Example: rc = projectSetElement (eElementEprom);

2.4.21 projectGetElementRanges

Syntax: projectGetElementRanges (bOptReturnECU=FALSE, bOptReturnLongFormat=FALSE)
Return value: string

This function returns the elements in the format expected by `projectSetElementRanges`. The long format also returns the fields `bHardware`, `bInvalid`, `ShowOffset`, and `InternalOffset`:

Example: rc = projectGetElementRanges();

2.4.22 projectSetElementRanges

Syntax: projectSetElementRanges (String RangeString)
Return value: bool

This function overwrites the currently configured definitions for the element ranges with the definitions from the parameter. That has the format ELEMENTNAME:FROM-TO. „From“ and „To“ are decimal values. Further ranges can be appended (separated by semicolons).

Example: rc = projectSetElementRanges ("Engine / Eprom:0-1048575; Engine / Processor:1048576-1507327");

2.4.23 projectGetDataAreas

Syntax: projectGetDataAreas (bOptReturnECU=FALSE)
Return value: int array (bool in case of an error)

This function returns the project's data areas as an array of numbers. Entry [1] is from1. [2]=to1, [2]=from2…

If TRUE is passed as a parameter, the numbers are converted to CPU addresses.

If no project is open, FALSE is returned.

Example: rc = projectGetDataAreas ();

2.4.24 projectAddCommentAt

Syntax: projectAddCommentAt (Number FromAdrress, Number ToAddress, String Text, Number FrameColor=-1, Number BackColor=-1)
Return value: bool

This function adds a comment for the range defined by the parameters. The frame and back colors are optional. The addresses are relative for the currently selected element.

Note: This function changes the project data and marks the project as changed.

Example: projectAddCommentAt (100, 200, "Test");

2.4.25 projectGetCommentAt

Syntax: projectGetCommentAt (Number Address)
Return value: String

This function checks if there is a comment at the given address and returns the comment text if there is one. If no comment is there, and empty string will be returned. The address is relative to the currently selected element.

Example: rc = projectGetCommentAt (100);

2.4.26 projectDelCommentAt

Syntax: projectDelCommentAt (Number Address)
Return value: bool

This function checks if there is a comment at the given address and deletes it, if there is one. The address is relative to the currently selected element.

Note: This function changes the project data and marks the project as changed.

Example: rc = projectDelCommentAt (100);

2.4.27 projectGetAt

Syntax: projectGetAt (Number Address, Number Datatype=eByte, Number nValues=1, bool bOrg=false)
Return value: Number/Array

This function reads the value stored in the hexdump at the given address and returns it. The address is relative to the currently selected element. The optional parameter “datatype” describes the organization and bitwidth that is used to store the value. Valid values for this parameter are eByte, eLoHi, eHiLo, eLoHiLoHi, eHiLoHiLo, eFloatLoHi, eFloatHiLo.

Note: In the case of an error the function will not return 0, but eEmptyvalue which (currently) equals the value -99999.

If you pass nValues>1, the function can query multiple consecutive values and return an array.

Example: rc = projectGetAt (100);

2.4.28 projectSetAt

Syntax 1: projectSetAt (Number Address, Number Value, Number Datatype=eByte, Number Mode=eSetAbsolute)
Syntax 2: projectSetAt (Number Address, String Values, Number Datatype =eByte, Number Mode=eSetAbsolute)
Return value: bool

This function writes one / multiple(s) value at the given address into the hexdump. The address is relative to the currently selected element. The optional parameter “datatype” describes the organization and bitwidth that is used to store the value. Valid values for this parameter are eByte, eLoHi, eHiLo, eLoHiLoHi, eHiLoHiLo, eFloatLoHi, eFloatHiLo, eDoubleLoHi, eDoubleHiLo, eBitLoHi, eBitHiLo.

If you’re passing multiple values as string, WinOLS expects hex values. You can specify decimal values with the postfix “M”. When using the eByte datatype, the string may contain ?? placeholders for values

which should not be replaced.

Note: This function changes the project data and marks the project as changed.

Note: You can pass eEmptyvalue for Value to reset the cell to the original value.

Example 1: rc = projectSetAt (100, 5000, eLoHi);
Example 2: rc = projectSetAt (100, "AF 46 C0 01 48 46 ?? 00");
Example 3: rc = projectSetAt (100, "10M 15M 20M", eHiLo, eSetPercent); -- eSetRelative is also allowed

2.4.29 projectSetOrg

Syntax 1: projectSetOrg ()
Syntax 2: projectSetOrg (Number Address)
Syntax 3: projectSetOrg (Number AddressStart, Number AddressEnd)
Return value: bool

This function resets the bytes of the entire projects / of the give address / address range to the original version.

Example: projectSetOrg();

Requires WinOLS 5.38.05

2.4.30 projectFindBytes

Syntax 1: projectFindBytes (Number StartAddress, Number OrgVer, Number FirstByte, …)
Syntax 2: projectFindBytes (Number StartAddress, Number OrgVer, String AllBytes, Datatype=eByte)
Syntax 3: projectFindBytes (Number StartAddress, Number EndAddress, Number OrgVer, Number FirstByte, …)
Syntax 4: projectFindBytes (Number StartAddress, Number EndAddress, Number OrgVer, String AllBytes, Datatype=eByte)
Return value: Number or Array

This function searches in the current element either in the original (0) or in the version (1) for the byte sequence. Each byte is appended as separate argument. The function accepts any number of arguments.

Alternatively you can pass all bytes in form of a string. In this case it doesn't matter if you use a space character, colon or tab character as separator. The „0x" prefix is optional, too, WinOLS always expects hex numbers. Furthermore you can replace individual byte in the search string with the ?? placeholder.

If StartAddres>=0 then the function will return the next occurrence at / after this address or -1 if the string isn't found. If StartAddress is -1, the function will return an array with all occurrences. (The array will be empty, if no occurrence was found.

The parameter EndAddress is optional. If you use it, it must not be 0 or 1 to avoid confusion with the OrgVer parameter.

Starting with WinOLS 5.15 you can optionally specify the data type after the string. Valid values are: eByte, eHiLo, eLoHi, eHiLoHiLo, eLoHiLoHi, eFloatHiLo, eDoubleLoHi, eDoubleHiLo, eBitLoHi, eBitHiLo.

All addresses are relative to the currently selected element.

Example 1: FoundAt = projectFindBytes (0, 1, 100, 101, 102);
Example 2: FoundAt = projectFindBytes (0, 1, "4C 46 30 01 48 46 ?? 00");
Example 3: AllPositions = projectFindBytes (-1, 1, "4C 46 30 01 48 46 ?? 00");

2.4.31 projectReplaceBytes

Syntax 1: projectReplaceBytes (Number StartAddress, Number EndAddress, String AllSearchCells, String AllReplaceCells, Number SearchOrgVer=0, Number LimitNumberResults =1, Number DataType=eByte, Number Mode=eSetAbsolute)
Syntax 2: projectReplaceBytes (Number StartAddress, Number EndAddress, String AlleSearchCells, Number ReplaceValue, Number SearchOrgVer=0, Number LimitNumberResults=1, Number DataType =eByte, Number Mode=eSetAbsolute)
Erfordert WinOLS 5.27

Return value: Number (number of replaced results).

This function searches for the sequence of cells in the current element either in the original (0) or in the version (1). The sequence will be passed in the form of a string. It does not matter if a space, comma or tab character is used as separator. Also the "0x" prefix is optional, WinOLS always expects hex numbers. Furthermore, you can replace single bytes in the search string with ?? as placeholder.

For replacement you can specify a string or a single value.

Decimal values can be specified in the string with the postfix "M". When using the eByte data type, the string can contain ?? placeholders for values that should not be searched/replaced.

Beispiel 1: rc=projectReplaceBytes (0, 4095, "2C 3C 4C ?? 00", 0);
Beispiel 2: rc=projectReplaceBytes (0, 4095, "2C 3C 4C ?? 00", "1 2 3", eByte, 1, 1, eSetRelative);

2.4.32 projectFindSimilarProjects

As of WinOLS 5.71.24 this function is obsolete, projectFindSimilarProjectsSql should be used instead, which is faster/more modern. Parameters and return values are identical.
The projects found may differ slightly in individual cases due to the different technology.

2.4.33 projectFindSimilarProjectsSql

Syntax: projectFindSimilarProjectsSql (Number MinPercent, Number MaxNumberOfResults, Number idDesiredColumn1, Number idDesiredColumn2, …)

Return value: array

This function searches for project similar to the active one the current client folder. It returns a sorted list, starting with the best match. The list contains values for the relevance and all desired properties. The similarity calculation is not influenced the columns that you specify. The active project is automatically excluded from the search results.

Data areas:
Starting with WinOLS 5.11 this function can also find projects with similar data areas. This is symbolized by negative values. To activate this feature, a minus sign must be placed in front of the value for MinPercent (example: -80). And in the result list, entries where only the data range is similar are also marked by a negative value (example: -100).

Property matches:

Starting with WinOLS 5.50 this function can also return projects where only a project property matches (and not the similarity, which is returned as 0). To activate this, pass the flag eFSPAllowPropertyMatches after MaxNumberOfResults.

Extended relevance info:
As of WinOLS 5.52, WinOLS can return three instead of one relevance info. The first remains the same, the next two are the project similarity and the data area similarity. To activate this, pass the eFSPTrippleRelevance flag after MaxNumberOfResults. (The exact position of the flag is ignored, WinOLS always returns the 3 relevance information in the first 3 columns.)

Example:

```
list = projectFindSimilarProjectsSql (80, 9999, ePrjFilename, ePrjPropVehicleProducer, ePrjUserdef1);
nColumns = 4;                -- Four, because we get the relevance and 3 desired columns
size = #list
for i=0, size/nColumns-1 do
   MessageBox ("File "..i.."="..  list [i*nColumns+1].."\n".. list [i*nColumns+2] .."\n".. list [i*nColumns+3]
.."\n".. list [i*nColumns+4]);
end
```

2.4.34 projectImportFromOls

Syntax: projectImportFromOls (String SourceFilename, Number iVersion=0, Number Flags=0)

Return value: Number (true if successful)

Imports maps and hexdump values from the source project to the current project in a new version. The projects must have the same size, an offset is not supported.

iVersion is the index of the version from which the data is imported. 0 is the original. If -1 is transferred here, no version data is transferred.

WinOLS5:
Instead of the version index, you can also use the version name instead.

Flags:
Several values (combined by binaryor(a, b)) can be transferred here:
eImportSkipEEprom: The content of the (virtual) EEproms is not transferred.
eImportOnlyDataArea: Only the contents of the data areas are imported.
eImportSkipMaps: The maps are not transferred.

WinOLS5:
The following parameters are additionally supported:
eImportMapsRelative: Transfer map changes relatively
eImportMapsPercent: Transfer map changes as percentages
eImportSkipInsideMaps: Don't transfer changes inside maps
eImportSkipOutsideMaps: Don't transfer changes outside maps
eImportOnlyChanged: Only import changed bytes (Can be combined with relative/percent for maps)
(The first 3 flags in this list are mutually exclusive.)

As of WinOLS 5.67.02:
eImportUseElementInformation: Uses element addresses+offset
eImportDontUseElementInformation: Uses raw addresses+offset (i.e. "All elements")
(These flags are mutually exclusive. It is recommended that you always specify one of the two, otherwise WinOLS will use for compatibility reasons the last setting used when manually importing from an ols file.)

Example:

adr = projectImportFromOls ("SourceProject.ols", 1, binaryor(eImportSkipEEprom, eImportOnlyDataArea));

2.4.35 projectFindMap

Syntax: projectFindMap (String Criterion, String Value, Number StartAddress=0)

Return value: Number/Array

Searches the current project for a map with the desired name or ID. Criterion can be "Name" or "IdName".

If StartAddress>=0: Returns the start address of the map (in all-element address mode) or -1.
If StartAddress==-1: Returns an array with all start addresses.

Example:
adr = projectFindMap ("IdName", "MyMapId");

2.4.36 projectAddMap

Syntax: projectAddMap ()
Return value: bool

Adds another map to the current project, which can be set up with windowSetMapProperties.

Example: projectAddMap();

2.4.37 projectDelMap

Syntax: projectDelMap (Number StartAddress)
Syntax: projectDelMap (String StringWithWildcards)
Return value: Number

Deletes all maps that begin at the given start address. Returns the number of maps that were deleted.

Example: projectDelMap(1000);
projectDelMap("*");
projectDelMap("MyUnimportantMaps_*");

2.4.38 projectImportCsvJson

Syntax: projectImportCsvJson (String Filename, String MatchModes="id,address")
Return value: bool
Requires WinOLS5

Imports the specified csv or json file into the current project. The MatchModes string defines which criteria are allowed for matching with existing maps and what should be preferred (if the file has both information).
Example: projectImportCsvJson ("d:\test.csv", "id");

2.4.39 projectImportMapPack

Syntax: projectImportMapPack (String Filename, Number Offset=0, String Prefix="", Number

bSkipDuplicates=false, Number bIgnoreAxis=false, Number bIgnoreTexts=false)
Return value: bool

Imports the given kp-file into the current project.

Example: projectImportMapPack ("d:\test.kp");

2.4.40 projectSetRight

Syntax: projectSetRight (Number idRight, bool bNewState)
Return value: bool

Changes the given right for the current project. Requires at least WinOLS 4.69 (see eWinOLSMajor, eWinOLSMinor).

This function will fail and return false if the registered WinOLS user isn't allowed to change the rights for this project (see projectSetRightsOwner).

Valid values for idRight are:
ePRExportBinary, ePRExportOLS, ePRExportBdm, ePRExportKp, ePRExportClipboard, ePRExportOther, ePRWriteEprom, ePRWriteBdm, ePROtherTransMapContent, ePROtherTransMapProp, ePRAccessHexdump, ePRAccessMaps, ePRAccessMapList

Example: projectSetRight (ePRExportKp, false); --Disallow map pack export

2.4.41 projectSetRightsOwner

Syntax: projectSetRightsOwner (Number EvcCustomerNumber)
Return value: bool

Changes the owner of the project rights. If this number doesn't match the customer number of the currently registered WinOLS user and at least one right not granted, all subsequent attempts to change the rights or the rights owner will fail. This applies both to lua commands and GUI use.

This function will fail and return false if the registered WinOLS user isn't allowed to change the rights for this project.

Example: projectSetRightsOwner (33333);

2.4.42 projectCountDifferentBytes

Syntax: projectCountDifferentBytes (bool bReallyReturnDifferences=FALSE)
Return value: int

Even if the name suggests otherwise, this function counts the number of **identical** bytes between the current version and the original of the project. For compatibility reasons, this cannot be corrected, but if you pass TRUE as a parameter, the different bytes are actually counted.

Any reference version is ignored. The function always considers the entire project (i.e. ignores the current element). It also takes changes made by checksums into account, even if these have been hidden (F12>Hexdump).

Example: nBytesDifferent = projectCountDifferentBytes (TRUE);

2.4.43 projectDelFolder

Syntax: projectDelFolder (String FolderName)
Return value: bool

Deletes the given folder and the maps inside.

Example: projectDelFolder ("Foldername");

2.4.44 projectDelDuplicateMaps

Syntax: projectDelDuplicateMaps (bool bOnlyCompareAddressesAndSizes, bool bOnlyWhenInTheSameFolder)
Return value: Number

Deletes duplicate maps in the current project. Corresponds to the map part of the WinOLS function "Project > Find duplicate objects" (but without the confirmations). The function returns the number of deleted maps or -1 in case of error.

Example: projectDelDuplicateMaps (FALSE, TRUE);

2.4.45 projectAddrCpuToRaw

Syntax: projectAddrCpuToRaw (Number Address)
Return value: Number

Converts the address from CPU format (as normally displayed in WinOLS) to the raw address you see when you select "all elements". If the address cannot be converted, 0xFFFFFFFF is returned.

Example: projectAddrCpuToRaw (fromhex("8000000"));

2.4.46 projectAddrRawToCpu

Syntax: projectAddrRawToCpu (Number Address)
Return value: Number

Converts the address from the raw format (which you see when you select "all elements") to the CPU address (as it is normally displayed in WinOLS). If the address cannot be converted, 0xFFFFFFFF is returned.

Example: projectAddrCpuToRaw (fromhex("1234"));

2.4.47 projectImportChanges

Syntax: projectImportChanges (string SourceFilename, string versionname, number flags=binaryor(eICImportMapStructure,eICImportMapContents), number transfermode=eAutmAbsolute, number tolerance=0)
Syntax: projectImportChanges (string SourceFilename, number versionindex, number flags, number transfermode=eAutmAbsolute, number tolerance=0, array ignoreRanges={})
Return value: bool
Rquires WinOLS5

Executes the function "Import changes: Automatic". The maps/values from the source project are

transferred to the target project. The offset is determined automatically. The correctness/completeness of the imported data cannot be guaranteed. The use of this function in an automatic process without human control is therefore susceptible to risk.

Valid values for flags are (binary-or-combinations off):
eICImportMapStructure (The map structure is transferred)
eICImportMapContents (The cell values in maps are transferred)
eICImportDataAreas (The cell values outside maps are transferred)
eICOnlyChangedMaps (Only maps with changes are considered)
eICCompareById (The offset is determined on the basis of the Id. The maps must already exist in the target project)
eICCompareByName (Dito. Keep in mind that the names must be unique)
eICAllowReturn4 (Unlocks return value 4 (to keep WinOLS compatible))
eICRemoveDuplicates (Deletes any existing duplicates in the target project)

Transfermode can have exactly one of these values:
eICTMAbsolute (Changed cell values are transmitted absolutely)
eICTMRelative (Changed cell values are transmitted relatively (delta))
eICTMPercent (Changed cell values are transmitted as percentages)
eICTMAllFromOrg (All original cell values are transmitted as absolute values)
eICTMAllFromVer (All cell values are transmitted absolutely)

The optional parameter ignoreRanges can specify from/to ranges in the source project that are not to be transferred. The array size must always be even. The addresses are expected in raw format (see projectAddrCpuToRaw).

Return value:
0=The source project could not be found/used
1=Not all maps/ranges could be imported
2=Not all map axes could be imported
3=Everything was imported
4=Everything was imported and there was only 1 offset. (Requires eICAllowReturn4)

Example 1: projectImportChanges ("x:\\myfolder\\source.ols", 1);
Example 2: projectImportChanges ("x:\\myfolder\\source.ols", 1 , eICImportDataAreas, eAutmAbsolute, 0, { 0,fromhex("0x400"), fromhex("0x1D80"),fromhex("0xFFFF")});

2.4.48 projectAutoUpdate

Syntax: projectAutoUpdate (bool bForce=FALSE)
Return value: bool
Requires WinOLS5.21 + FeatureUpdate

Triggers AutoUpdate (Like: Projekt > Ex-&Import > AutoUpdate) for the project.

bForce: If TRUE, then the data will be updated, even if the file timestamp doesn't require it.

Example: projectAutoUpdate ();

2.4.49 projectAutoImport

Syntax: projectAutoImport (bool bForce=FALSE)
Return value: bool
Requires WinOLS5.21 + FeatureUpdate

Triggers AutoImport (Like: Projekt > Ex-&Import > AutoImport) for the project.

bForce: If TRUE, then the data will be updated, even if the file timestamp doesn't require it.

Example: projectAutoImport ();

2.4.50 projectSearchVehicleData

Syntax: projectSearchVehicleData ()
Return value: bool

Triggers the search for project properties.

Example: projectSearchVehicleData ();

2.4.51 projectCloneVehicleData

Syntax: projectCloneVehicleData ();
Syntax: projectCloneVehicleData (String clientname, Number flags, String emptyvalues);

Return value: bool

Uses the "Update projects > Clone" function (and its settings by default) to transfer project properties from projects with identical search fields to the current project. All 3 parameters are optional. If they are not specified, WinOLS uses the current settings from the dialog. If the second parameter is specified, it must contain ALL flags (binaryor) that are to be used. In the third parameter, the individual values are separated by semicolons.

Die möglichen Flags für Parameter 2:
eClonePropCompareSoftware
eClonePropCompareSoftwareVersion
eClonePropCompareEngineDescription
eClonePropCompareECUProd
eClonePropVehicle
eClonePropUserdef
eClonePropECU
eClonePropEngine
eClonePropAcceptDissent
eClonePropSimpleMajority
eClonePropAllowOverwrite

Example 1: projectCloneVehicleData ();
Example 2: projectCloneVehicleData ("myclient", binaryor(eClonePropCompareECUProd, eClonePropECU, eClonePropVehicle), "nix;nada;empty");

2.4.52 projectGetQuickFixState

Syntax: projectGetQuickFixState (String name, bool bReturnInt=true)
Rückgabewert: Number oder String

Returns the status of the desired QuickFix. The name does not have to match exactly, parts are also sufficient.

The return value is either a number or a string. A 1 or 0 indicates on or off. (WinOLS recognizes most

common state names automatically.) Important: This is a number, a comparison with the Boolean "true" or "false" will not work! In string mode, the name of the state is returned. The return value -1 (or "undef") means that the current state is contradictory. -2 (or "unknown") means that no QuickFix with the name was found.

If this function is called without an open project, it returns false (Boolean).
Requires WinOLS 5.72

Example: stateNumber = projectGetQuickFixState (“FeatureX”);

2.4.53 projectSetQuickFixState

Syntax: projectSetQuickFixState (String name, StringOrBoolen newState)
Rückgabewert: Number

Sets the status of the QuickFix. The name does not have to match exactly, parts are also sufficient. If you pass the status as a string, it must match exactly (otherwise "applied" would also match "not applied"). If you pass the status as a Boolean (e.g. true), WinOLS automatically understands most of the textual designations commonly used in the project.

A return value of 1 indicates success. The return value -1 means that the desired state was not found. -2 means that no QuickFix with the name was found.

If this function is called without an open project, it returns false (Boolean).
Requires WinOLS 5.72

Example: rc = projectSetQuickFixState (“FeatureX”, true);

2.4.54 projectGetQuickFixes

Syntax: projectGetQuickFixes ()
Return value: Array

Returns a list of the names of all QuickFixes available in the current project.

If this function is called without an open project, it returns false (Boolean).
Requires WinOLS 5.72

Example:
```
a = projectGetQuickFixes();
size = #a;
firstentry = a[1];
```

2.4.55 projectImportMapFile

Syntax: projectImportMapFile (String filename, Number offset=0, Number flags)
Return value: bool or array

Imports the maps from a map file. (Currently only .a2l / *.asap2. Requires OLS521.)

If this function is called without an open project, it returns false (Boolean). Otherwise, it returns a Number array with 4 entries: nImported, nParseWarnings, nParseErrors, nUnimportable.

The following flags can be passed (combined using binaryor(..)):

eIMFUseElement: If element information is available, use it and ignore the offset. If no elements are present in the project, this flag is ignored and the offset parameter is used anyway.
eIMFDetectOffset: Uses the plugin's offset detection. (Only fast mode is used to avoid interrupting the script for too long.)
eIMFCreateRamElements: Creates RAM elements (if present in the file) and also imports the maps contained therein.

Example:
```
rc = projectImportMapFile("C:\\myfile.a2l", 0, binaryor(eIMFUseElement, eIMFCreateRamElements))

if (rc==false) then
  -- Error: you don't have a current project
else
  nImported        = rc[1];  -- Number or imported map. 0 if file (access) error.
  nParseWarnings   = rc[2];  -- number of parser warnings
  nParseErrors     = rc[3]   -- number of parser errors
  nUnimportable    = rc[4];  -- >0 usually means the file doesn't match or the offset is wrong
end
```

2.5 Context: Version

2.5.1 versionGetProperty

Syntax: versionGetProperty (Number id)
Return value: String or Number

This function queries one of the texts from the version properties.

The parameter "id" must have one of the following values: eVerPropName, eVerPropComment, eVerPropCreatedOn, eVerPropChangedOn, eVerPropChecksum, eVerPropCVN, eVerPropOutput, eVerPropTorque, eVerPropState, eVerPropCredits

The following return values are possible with eVerPropState: eVerStatNone, eVerStatExperiment, eVerStatToDo, eVerStatDev, eVerStatTestable, eVerStatErrors, eVerStatFinished, eVerStatMaster, eVerAutoExport, eVerAutoImport, eVerAutoUpdate, eVerAutoUpdateAndExport

Example: MessageBox (versionGetProperty (eVerPropName));

2.5.2 versionSetProperty

Syntax: versionSetProperty (Number id, String newvalue)
Syntax: versionSetProperty (Number id, Number newvalue)
Return value: bool

This function changes one of the texts from the version properties to a new value.

The parameter "id" must have one of the values from versionGetProperty. Note: This function changes the project data and marks the project as changed.

Example: versionSetProperty (eVerPropName, "Tuned version");

2.6 Context: Window

The context of the current window always implies the current project, since this is always connected to the window. Here, window only means the project windows (map, hexdump, ...), but not the special windows like search, overview, etc.

2.6.1 windowGetActive

Syntax: windowGetActive ()
Return value: Number

This function returns an identifier (Type: Number) for the current window. The identifier is valid only for this session. If the window or WinOLS get closed the identifier becomes invalid.

Example: aw = windowGetActive ();

2.6.2 windowSetActive

Syntax: windowSetActive(Number WindowIdentifier)
Return value: bool

This function activates the window that is identified by the parameter and brings it to the foreground. All functions that use the project or window as context now use this window / project.

Example: windowSetActive (aw);

2.6.3 windowGetMapProperties

Syntax: windowGetMapProperties (string PropertyName)
Syntax: windowGetMapProperties (string PropertyName, number MapStartAddress)
Syntax: windowGetMapProperties (string PropertyName, number MapStartAddress, number nSkip)
Syntax: windowGetMapProperties (string PropertyName, string MapId)
Return value: string

Returns the current value for the selected map property. The possible PropertyNames are the same as for the WinOLS-Script-command set_map_property. (See WinOLS help file)

If an address or a map ID is passed as the second parameter, the function does not refer to the active map window, but to the map that starts at the specified address / has the specified map ID. If there is more than one map that meets this condition, you can use nSkip to specify how many of them should be skipped.

This function requires that the project right "Transfer map structure" is given.

The possible PropertyNames are the same as for the WinOLS-Script-command set_map_property. (See WinOLS help file)

Example: windowGetMapProperties ("name");

2.6.4 windowSetMapProperties

Syntax: windowSetMapProperties (string PropertyName, string/number Wert)
Syntax: windowSetMapProperties (string PropertyName, string/number Wert, bool bLastNew)
Syntax: windowSetMapProperties (string PropertyName, string/number Wert, number MapStartAddress)
Syntax: windowSetMapProperties (string PropertyName, string/number Wert, number MapStartAddress, number nSkip)
Syntax: windowSetMapProperties (string PropertyName, string/number Wert, string MapId)
Return value: bool

Changes the property of the current map (if bLastNew==TRUE: of the last map that was created by LUA). If an address (>1) or a map ID is passed as the third parameter, the function does not refer to the active map window, but to the map that starts at the specified address / has the specified map ID. If there is more than one map that meets this condition, you can use nSkip to specify how many of them should be skipped.

The possible PropertyNames are the same as for the WinOLS-Script-command set_map_property. (See WinOLS help file)

Example: windowSetMapProperties ("Name", "Example ", TRUE);

3 Index

B

binaryor 9, 10, 12

C

CloseAll 13
Context
  project 19
  Version 36
  Window 37
Context philosophy 19

F

Filenames 8

G

GetVersion 14
Global functions 8

M

MessageBox 8, 9

N

NewProject 13, 14, 16, 17, 18, 19

P

projectAddCommentAt 26
projectApplyChecksums 23
projectClose 20
projectDelCommentAt 27
projectExport 21
projectFindBytes 15, 28, 29, 30, 31, 32, 33
projectGetAt 27
projectGetChecksummenName 24
projectGetCommentAt 27
projectGetElement 26
projectGetElementOffset 25
projectGetProperty 19
projectImport 22
projectMail 23
projectSave 21
projectSearchChecksums 23
projectSetAt 27, 28
projectSetElement 26
projectSetElementRanges 26
projectSetProperty 20
projectStatChecksums 24

Q

Quit 12

R

Requirements 5
Return values 8

S

SaveAll 13
Security advice 6
Sleep 14
Starting LUA 6

V

versionGetProperty 36
versionSetProperty 37

W

windowGetActive 37
windowSetActive 37, 38
WinOLS Configuration 6