Manual

External Control for WinOLS

(as of: 07.03.2018)

Contents

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 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 line	require ("library");
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 = table.getn(a)
For / Next loop	for i=1, size do MessageBox ("a["..i.."]".."="..a[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 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

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.

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

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 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));

2.2.3 fromhex

Syntax: fromhex (string HexNumberAsString)
Return value: Number

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

Example: adr = fromhex("C000”);

2.2.4 tohex

Syntax: tohex (int number)
Return value: String

Converts the parameter into a hex string

Example: MessageBox(tohex(49152));

2.2.5 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.

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

2.2.6 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.

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

2.2.7 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.

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

2.2.8 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();

2.2.9 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 ();

2.2.10 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 ();

2.2.11 SetClient

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

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

Beispiel: SetClient ();

2.2.12 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 ();

2.2.13 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));

2.2.14 Sleep

Syntax: Sleep (Number Milliseconds)
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.

Example: Sleep (5000);

2.2.15 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 = table.getn(a)
for i=1, size do
  MessageBox ("a["..i.."]".."="..a[i]);
end
```

2.2.16 GetProjectVersions

Syntax: GetProjectVersions (String Filename)
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.

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

2.2.17 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%");

2.2.18 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%");

2.2.19 OpenAndExport

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");

2.2.20 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 ();

2.2.21 StartUrl

Syntax: StartUrl (string url)

Opens a browser window with the requested URL.

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

2.2.22 ReadDirectory

Syntax: ReadDirectory (string filenameandwildcards)

Returns an array with file and foldernames.

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

2.2.23 SetSolutionInformation

Syntax: SetSolutionInformation (number mode, int/string value)

Sets the return values if the script should be seen as Solution.

Mode:

eSolutionInfoReturncode:	TRUE if the script should be applicable, otherwise FALSE
eSolutionInfoName:	The name of the script
eSolutionInfoDescription:	A description
eSolutionInfoDebugUser:	Optional: Debug-informationen. Visible for the end user.
eSolutionInfoDebugDeveloper:	Optional: Debug-informationen. Visible only when run locally.
eSolutionInfoFullFile:	Optional: A different SolutionFull filename than the current.
eSolutionInfoLockKey:	Optional: A string to which the purchase is bound.
eSolutionInfoInfoKey:	Optional: Information, that is stored in the purchase DB
eSolutionInfoCredits:	Required for SolutionFull: The purchase price in credits
eSolutionInfoEnd:	Required for SolutionFull: Marks the end of the info header.

Note: SetSolutionInformation cuts off all values after 255 characters.

See also: Chapter “Solutions”

2.2.24 GetSystemInformation

Syntax: GetSystemInformation (number mode)

Returns informationen about the system. Supported value for „mode“:
eSysInfoLanguageDeEn: Returns the selected WinOLS language. Either „de“ or „en“.

2.2.25 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.

2.3 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.

2.4 Context: Project

2.4.1 projectGetProperty

Syntax: projectGetProperty (Number id)
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, ePrjPropClientLicenceplace
- ePrjPropVehicleType, ePrjPropVehicleProducer, ePrjPropVehicleChassis, ePrjPropVehicleModel, ePrjPropVehicleModelyear, ePrjPropVehicleVIN
- ePrjPropEcuProducer, ePrjPropEcuBuild, ePrjPropEcuProdNr, ePrjPropEcuStgNr, ePrjPropEcuSoftwareversion, ePrjPropEcuSoftwareversionVersion, ePrjPropEcuChecksum, ePrjPropEcuSoftwaresize, ePrjPropEcuUse
- ePrjPropEngineName, ePrjPropEngineType, ePrjPropEngineDisplacement, ePrjPropEngineTransmission, ePrjPropEngineOutputPS, ePrjPropEngineOutputKW, ePrjPropEngineEmissionStd
- ePrjPropCommunicationsReadhardware
- ePrjPropProjectType, ePrjProjectStatus
- ePrjFileCreatedBy, ePrjFileModifiedBy, ePrjFileCreatedOn, ePrjFileModifiedOn
- ePrjUserdef1, ePrjUserdef2, ePrjUserdef3, ePrjUserdef4, ePrjUserdef5

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

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 ()
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.

Example: Save ();

2.4.5 projectExport

Syntax: projectExport (String Filename, Number Typ, String ZippedFileiname=““, Number BdmToGoFlags=0, Number olsxCustomerNumber=0, String olsxPassword=””)
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 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).

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 projectImport

Syntax: projectImport (String Filename, Number Typ=eFiletypeAuto, Number Offset=0, String olsxPassword="")
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, eFiletypeODX

If no type is give, the value “eFiletypeAuto” will be assumed and the file type will be detected automatically.

The optional parameter “Offset” names an offset which moves the imported data back in the file. This parameter is use 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 the for the original).

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");

2.4.7 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.");

2.4.8 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 ();

2.4.9 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 ();

2.4.10 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 ();

2.4.11 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 ();

2.4.12 projectGetChecksumOptionStatus

Syntax: projectGetChecksumOptionStatus (Number index)
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 (0);

2.4.13 projectGetChecksumOptionText

Syntax: projectGetChecksumOptionText (Number index)
Return value: string

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

Example: rc = projectGetChecksumOptionText (0);

2.4.14 projectGetChecksumOptionType

Syntax: projectGetChecksumOptionStatus (Number index)
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 (0);

2.4.15 projectSetChecksumOptionStatus

Syntax: projectSetChecksumOptionStatus (Number index)
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.

Example: projectSetChecksumOptionStatus (0, 1);

2.4.16 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 ();

2.4.17 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 ();

2.4.18 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 = projectGetElement ();

2.4.19 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");

2.4.20 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: projecAddCommentAt (100, 200, "Test");

2.4.21 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);

2.4.22 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);

2.4.23 projectGetAt

Syntax: projectGetAt (Number Address, Number Datatype=eByte)
Return value: Number

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.

Example: rc = projectGetAt (100);

2.4.24 projectSetAt

Syntax 1: projectSetAt (Number Address, Number Value, Number Datatype=eByte)
Syntax 2: projectSetAt (Number Address, String Values)
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.

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

Example 1: rc = projectSetAt (100, 5000, eLoHi);
Example 2: rc = projectSetAt (100, "AF 46 C0 01 48 46 ?? 00");

2.4.25 projectFindBytes

Syntax 1: projectFindBytes (Number StartAddress, Number OrgVer, Number FirstByte, …)
Syntax 2: projectFindBytes (Number StartAddress, Number OrgVer, String AllBytes)
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.

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");

2.4.26 projectFindSimilarProjects

Syntax: projectFindSimilarProjects (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.

Example:

```
list = projectFindSimilarProjects (80, 9999, ePrjFilename, ePrjPropVehicleProducer, ePrjUserdef1);
nColumns = 4;              -- Four, because we get the relevance and 3 desired columns
size = table.getn(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
```

2.4.27 projectAddMap

Syntax: projectAddMap ()
Return value: bool

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

Example: projectAddMap();

2.4.28 projectDelMap

Syntax: projectDelMap (Number StartAddress)
Return value: Number

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

Example: projectDelMap(1000);

2.4.29 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");

2.4.30 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

2.4.31 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);

2.4.32 projectCountDifferentBytes

Syntax: projectCountDifferentBytes ()
Return value: int

Counts the number of different bytes between current version and original of the project. If a reference project is currently selected, it will be ignored.

Example: nBytesDifferent = projectCountDifferentBytes ();

2.5 Context: Version

2.5.1 versionGetProperty

Syntax: versionGetProperty (Number id)
Return value: String

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

The parameter “id” must have one of the following values: eVerPropName, eVerPropComment

Example: MessageBox (versionGetProperty (eVerPropName));

2.5.2 versionSetProperty

Syntax: versionSetProperty (Number id, String 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 windowSetMapProperties

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

Changes the property of the current map (if bLastNew==TRUE: of the last map that was created by LUA). 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