Anleitung

External Control für WinOLS

(Stand: 14.04.2025)

Inhaltsverzeichnis

1 Vorbemerkungen

1.1 Was ist External Control für WinOLS?

WinOLS erlaubt es mit vielfältigen Methoden Daten aus Steuergeräten zu lesen, zu bearbeiten und zu schreiben. Das Plugin „External Control“ erlaubt es jetzt zahlreiche Funktionen von WinOLS über Skripte zu steuern. Dazu bringt das Plugin eine komplette Programmiersprache namens „LUA“ mit. Dadurch ist es möglich die Datenverarbeitung bei wiederkehrenden Vorfällen zu automatisieren.

Hinweis: Das Plugin „External Control“ hat nichts mit der bereits in WinOLS integrierten Skript-Funktionen zu tun (*.winolsskript), sondern bietet erheblich größere Möglichkeiten.

1.2 Was ist LUA?

LUA ist eine fertige Programmiersprache, die in WinOLS integriert wurde. Diese Anleitung dokumentiert die WinOLS Funktionen, die Lua zur Verfügung gestellt werden. Für eine Dokumentation der eigentlichen Sprache LUA, konsultieren Sie bitte die LUA Homepage:
http://www.lua.org/

1.3 Lua im Schnelldurchlauf

Datei-Include	getfile ("library.lua");
Zeilenkommentar	--Kommentartext
Strings verketten	MessageBox("s=“..st);
Strings Gleichheit / Ungleichheit	if (str==“equal“ and str~=“unequal“) then
Anzahl Array-Elemente	size = #myarray
For / Next Schleife	for i=1, size do MessageBox ("myarray ["..i.."]".."=".. myarray [i]); end
Hilfsfunktion (aus library.lua) für Variableninhalt	MessageBox_r(a);
Strings ersetzen	s = string.gsub(s, "needle", "replacement");
Textdatei lesen	INPUT = io.open("myfile.txt", "rb") inputfile = chomp(INPUT:read("*line")) INPUT:close();
Textdatei schreiben	OUTPUT = io.open ("myfile.txt", "w") OUTPUT:write ("my content"); OUTPUT:close();
Datei löschen	os.remove ("myfile.txt");

1.4 Voraussetzungen

Zum Betrieb des Plugins „External Control“ benötigen Sie:

- Eine registrierte Version von WinOLS (Mindestens Version 1.219)
- Eine registrierte Version von des „External Control“ Plugins
  (nicht im Lieferumfang von WinOLS enthalten)

Bitte beachten Sie:
Zum Betrieb des „External Control“ Plugins auf einem Webserver benötigen Sie i.d.R. einen eigenen Windows Webserver. Der Grund ist, daß die WinOLS Funktionen sehr aufwendig sein können. Ein Betrieb im Time-Sharing mit anderen Servern oder in einem Prozess mit Zeitbegrenzung führt meist zu Problemen.

1.5 Sicherheitshinweis

LUA enthält Funktionen zum Dateimanagement, darunter auch Funktionen zum Löschen von Dateien. Starten Sie daher LUA Skripte nur, wenn sie aus Quellen stammen, denen Sie vertrauen. Beim Betrieb auf Servern müssen Sie dafür sorgen, daß keine Skripte von nicht autorisierten Personen eingebracht werden können.

[Hinweis: Für Lösungen wird der Zugriff auf die io Library blockiert.]

1.6 WinOLS Konfiguration

Aus Performancegründen sollten Sie auf dem Server WinOLS so konfigurieren, daß folgende Optionen ausgeschaltet sind:

- Potentielle Kennfelder suchen (Im Hintergrund)
- Übersichtsinformationen erzeugen (Im Hintergrund)
- Skripte vorschlagen, falls passend (Beim Import)

In der Regel wird es für einen korrekten Ablauf erforderlich sein, daß folgende Optionen eingeschaltet sind:

- Fahrzeugdaten suchen & übernehmen (Beim Import)
- Checksummen suchen & aktualisieren (Beim Import)

1.7 LUA starten

Sie haben drei Möglichkeiten ein LUA Skript zu starten:

- Übergeben Sie den vollständigen Pfad und Dateinamen als Parameter beim Start von WinOLS
- Ziehen Sie die LUA-Datei per Drag + Drop in das WinOLS Rahmenfenster
- Ziehen Sie die LUA-Datei per Drag + Drop in ein WinOLS Projektfenster

Die erste Möglichkeit setzt voraus, dass WinOLS derzeit nicht läuft. Sollte WinOLS gerade laufen (egal ob im Leerlauf oder in der Skriptausführung), dann wird der Aufruf ignoriert.

Die dritte Möglichkeit setzt das Zielfenster als Standardprojekt (Kontext) für das Skript. Mehr dazu finden Sie unter „Kontextphilosophie“.

1.8 Beispiele und Serverbetrieb

Unter dieser Adresse finden Sie einige Beispielprogramme für den Einsatz LUA in WinOLS:
https://www.evc.de/de/product/ols/lua.asp

Erwähnenswert ist dabei der Betrieb als Server. In diesem Modus wird nicht einfach ein Skript auf das aktuelle Projekt angewendet. Stattdessen läuft ein Skript permanent ab Programmstart und reagiert über „Ticket“ Dateien auf Anfragen von außen, zum Beispiel von einem Webserver.

1.9 WinOLSSkript

Sie können LUA und WinOLSSkript kombinieren indem Sie mit projectImport eine WinOLSSkript-Datei in das aktuelle Projekt importieren.

2 Sprachumfang

Besprochen werden an dieser Stelle nur die WinOLS Funktionen, die LUA hinzugefügt wurden. Für eine Dokumentation der eigentlichen Sprache LUA, konsultieren Sie bitte die LUA Homepage: http://www.lua.org/

2.1 Hinweise

2.1.1 Rückgabewerte

Einige Funktionen geben einen boolean Wert zurück um anzuzeigen ob sie erfolgreich waren. Wenn nicht anders dokumentiert, geben diese Funktion im Erfolgsfall den Wert „TRUE“ (entspricht 1) zurück und im Fehlerfall „FALSE“ (entspricht 0).

2.1.2 Dateinamen

Wenn Sie einer Funktion einen Dateinamen übergeben, sollten Sie (sofern möglich) immer einen vollständigen Dateinamen inklusive Pfad übergeben, da nicht garantiert werden kann, dass Sie immer das gleiche Arbeitsverzeichnis haben.

2.2 Globale Funktionen

Globale Funktionen sind keinem besonderen Kontext zugeordnet. Sie funktionieren unabhängig vom aktuellen Projekt oder Fenster.

2.2.1 MessageBox

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

Diese Funktion erlaubt es Windows-Fenster anzuzeigen und spiegelt damit die gleichnamige Windows Funktion wieder. Sie eignet sich gut zum testen, sollte aber auf Servern gemieden werden um die Prozesse nicht anhalten zu lassen.

Type kann als binäre Kombination folgender Werte angegeben werden:
MB_ICONERROR, MB_ICONQUESTION, MB_ICONWARNING, MB_ICONINFORMATION, MB_ABORTRETRYIGNORE, MB_OK, MB_OKCANCEL, MB_RETRYCANCEL, MB_YESNO, MB_YESNOCANCEL

Der Rückgabewert kann einen der folgenden Werte annehmen: IDYES, IDNO, IDCANCEL, IDOK, IDABORT, IDRETRY, IDIGNORE.

Hinweis: LUA unterstützt kein „binäres oder“. Verwenden Sie die Funktion „binaryor“ stattdessen.

Beispiel 1: MessageBox ("Test");

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

2.2.2 TextEntryDialog (alt)

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

Diese Funktion erlaubt es Windows-Fenster anzuzeigen und einen Text abzufragen. Der eingegebene Wert wird als String zurückgegeben. (Oder ein Leerstring falls Abbrechen gedrückt wurde.)

Der Modus unterstützt folgende Werte:
eTextEntrySmall: Ein kompakter Dialog
eTextEntryPassword: Genauso, aber im Kennwortmodus (Sternchen statt Buchstaben)

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

2.2.3 TextEntryDialog (neu)

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

In diesem Modus ist die Funktion TextEntryDialog flexibler. Die 3 Parameter mode/Description/DefaultValue können mehrfach wiederholt werden um mehrere Dinge gleichzeitig abzufragen.

Der Modus unterstützt folgende Werte:
eTextEntryEdit: Texteingabefeld
eTextEntryPassword: Genauso, aber im Kennwortmodus (Sternchen statt Buchstaben)
eTextEntryCheckbox: Eine Checkbox. Der DefaultValue sollte 1 oder 0 sein.

Der Dialog darf bis zu 10 Texteingabefelder und 20 Checkboxen enthalten. Im Erfolgsfall wird ein Array mit den eingegebenen Werten zurückgegeben. Bricht der Anwender den Dialog ab, wird ein Array mit der Länge 0 zurückgegeben.

Beispiel:
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);
```

2.2.4 fromhex

Syntax: fromhex (string HexZahlAlsString)

Rückgabewert: Zahl

Rechnet die angegeben hex Zahl (ein String) in eine echte Zahl um.

Example: adr = fromhex("C000");

2.2.5 tohex

Syntax: tohex (int number)
Rückgabewert: String

Erzeugt aus der angegebenen Zahl einen String mit einer Hex-Zahl.

Example: MessageBox(tohex(49152));

2.2.6 fromJSON

Syntax: fromJSON (string strJSON)
Rückgabewert: Objekt

Erwartet einen String mit JSON-Daten und erzeugt ein LUA-Objekt daraus.

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

2.2.7 toJSON

Syntax: toJSON (Objekt daten)
Rückgabewert: String

Erwartet ein LUA-Objekt und erzeugt einen JSON-String daraus

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

2.2.8 HttpStart

Syntax: HttpStart (string url, string verb="GET")
Rückgabewert: bool

Erster Befehl um einen http-Request einzuleiten.

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

2.2.9 HttpAddHeader

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

Fügt dem http-Aufruf einen Header hinzu. (optional)

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

2.2.10 HttpAddParam

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

Fügt dem http-Aufruf einen Parameter hinzu. (optional)

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

2.2.11 HttpAddFile

Syntax: HttpAddFile (string name, string path_and_filename)
Rückgabewert: bool

Fügt dem http-Aufruf eine Datei als Paramter hinzu. (optional)

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

2.2.12 HttpExecute

Syntax: HttpExecute ()
Rückgabewert: Number

Führt den http-Aufruf aus. Gibt den http-statuscode als Zahl zurück. Im Erfolgsfall i.d.R. 200. Ihre HttpAdd… Aufrufe müssen vorher ausgeführt werden. Ihre HttpResponse… Aufrufe nachher.

Beispiel: rc=HttpExecute ();

2.2.13 HttpResponseError

Syntax: HttpResponseError ()
Rückgabewert: String

Gibt den http-statustext als String zurück. Im Erfolgsfall i.d.R. „OK“.

Beispiel: rc=HttpResponseError ();

2.2.14 HttpResponseString

Syntax: HttpResponseString ()
Rückgabewert: String

Gibt das Ergebnis des http-Aufrufs als String zurück

Beispiel: rc=HttpResponseString ();

2.2.15 HttpResponseFile

Syntax: HttpResponseFile (string path_and_filename)
Rückgabewert: bool

Speichert das Ergebnis des http-Aufrufs als Datei.

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

2.2.16 binaryor

Syntax: binaryor (Number param1, Number param2, ...)
Rückgabewert: Number

Da LUA kein „binäres oder“ wie beispielsweise C enthält, wurde diese Funktion eingefügt. Sie unterstützt beliebig viele Parameter.

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

2.2.17 binaryxor

Syntax: binaryxor (Number param1, Number param2, ...)
Rückgabewert: Number

Da LUA kein „binäres x-oder“ wie beispielsweise C enthält, wurde diese Funktion eingefügt. Sie unterstützt beliebig viele Parameter.

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

2.2.18 binaryand

Syntax: binaryand (Number param1, Number param2, ...)
Rückgabewert: Number

Da LUA kein „binäres und“ wie beispielsweise C enthält, wurde diese Funktion eingefügt. Sie unterstützt beliebig viele Parameter.

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

2.2.19 Log

Syntax: Log (string)
Rückgabewert: -

Schreibt einen String in das WinOLS logfile (und die Service-Konsole, falls die verwendet wird).

Beispiel: Log("test");

2.2.20 Quit

Syntax: Quit ()
Rückgabewert: -

Diese Funktion beendet WinOLS.

Hinweise:

- Alle ungespeicherten Änderungen gehen verloren. Verwenden Sie zuvor den Befehl SaveAll(), wenn Sie das nicht wollen.
- Alle offenen Projekte werden beim nächsten Start wieder hergestellt. Dies ist für den Serverbetrieb ungünstig, da es auf Dauer immer mehr werden können und so den Speicher belasten. Verwenden Sie daher die Funktion CloseAll() bevor Sie WinOLS beenden.
- Wenn Sie im Betrieb auf dem Server WinOLS mit einem LUA Skript starten und erwarten, dass WinOLS nach Beendigung des Skripts geschlossen wird, dann muss das Skript mit diesem Befehl enden.

Beispiel: Quit();

2.2.21 SaveAll

Syntax: SaveAll ()
Rückgabewert: -

Diese Funktion speichert alle Änderungen in offenen WinOLS Projekten. Wird ein Original mit Änderungen im Hexdump gespeichert, dann wird eine Version mit dem Namen „CreatedByLua“ angelegt. Wird eine Version mit Änderungen im Hexdump gespeichert, dann wird die gleiche Version überschrieben.

Beispiel: SaveAll ();

2.2.22 CloseAll

Syntax: CloseAll ()
Rückgabewert: -

Diese Funktion schließt alle offenen WinOLS Projekte. Ungespeicherte Änderungen gehen dabei verloren.

Hinweise:
- Alle offenen Projekte werden beim nächsten Start wieder hergestellt. Dies ist für den Serverbetrieb ungünstig, da es auf Dauer immer mehr werden können und so den Speicher belasten. Verwenden Sie daher die Funktion CloseAll() bevor Sie WinOLS beenden.

Beispiel: CloseAll ();

2.2.23 SetClient

Syntax: SetClient(string clientname)
Rückgabewert: bool (true bei Erfolg)

Syntax: SetClient()
Rückgabewert: string (aktueller Wert)

Ändert den aktiven Mandanten. Diese Funktion beeinflusst darauf folgende Aufrufe wie NewProject oder FindProjects. Auf bereits angelegte / geöffnete Projekte hat sie keinen Einfluss.

Anstelle eines Kundennamens können Sie auch einen Resellernamen angeben. Dadurch können Sie z.B. mit projectFindSimilarProjects auf die Projektliste des Wiederverkäufers zugreifen (aber Sie können nicht über LUA kaufen). Bitte beachten Sie, dass Sie den Client vor der Erstellung des nächsten Projekts auf einen regulären, lokalen Client zurücksetzen MÜSSEN.

Wenn man keinen Parameter übergibt wird der aktuelle Mandant zurückgegeben.

Beispiel: SetClient ("test");

2.2.24 NewProject

Syntax: NewProject()
Rückgabewert: -

Diese Funktion öffnet ein neues, leeres WinOLS Projekt. Das neue Projekt (bzw. Fenster) wird als aktives Projekt (bzw. Fenster) verwendet.

Beispiel: NewProject ();

2.2.25 GetVersion

Syntax: GetVersion (Number VersionId)
Rückgabewert: -

Diese Funktion gibt die Versionsnummer (Höherwertig oder Niederwertig) von WinOLS oder vom „External Control“ Plugin zurück. Der Parameter „VersionId“ muss einen der folgenden Werte annehmen: eWinOLSMajor, eWinOLSMinor, ePluginMajor, ePluginMinor.

Benutzen Sie den Befehl um bei zukünftigen Änderungen oder Erweiterung die Kompatibilität zu gewährleisten.

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

2.2.26 Sleep

Syntax: Sleep (Number Milliseconds)
Sleep (Number Milliseconds, String pathAndWildcards)
Rückgabewert: -

Diese Funktion läßt WinOLS eine bestimmte Zeitspanne „schlafen“, so dass weder Rechenpower verbraucht wird, noch Aktionen von WinOLS ausgeführt werden. Benutzen Sie sie um Daten regelmäßig zu prüfen ohne zu viel Rechenpower zu verbrauchen. Der Parameter wird in 1/1000 Sekunden angegeben.

Falls Sie als zweiten Parameter einen Pfad inklusive Dateiname mit Wildcards übergeben, wird die Funktion sofort beendet, wenn eine solche Datei existiert / erzeugt wird oder falls WinOLS beendet wird.

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

2.2.27 FindProjects

Syntax: FindProjects (String Hersteller, String Chassis, String Ausführung, String md5Cpu, String md5Eprom, Number ProjektStatus, string Softwareversion)
Rückgabewert: Array

Diese Funktion gibt ein Array mit allen Dateinamen von Projekten zurück, auf die die angegebenen Suchkriterien passen. Gültige Werte für den ProjektStatus sind die Werte ePrjDeveloping, ePrjFinished, ePrjMaster oder 0.

Verwenden Sie Leerstrings (""), bzw. die Zahl 0 um Indifferenz für ein Kriterium anzuzeigen.

Es wird empfohlen den ProjektStatus zu verwenden um Verwechslungen mit temporär angelegten Projekten auszuschließen.

Beispiel:
```
a = FindProjects("","","","1253b4de81311b81c05a623eaa5781ff", "", ePrjMaster, "");

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

2.2.28 FindProjects2

Syntax: FindProjects2 (Number MaxErgebnisAnzahl, string GenerelleSuche, Number idSuchSpalte1, String Spalte1Value, Number idSuchSpalte2, String Spalte2Value, Number idGewuenschteSpalte1, Number idGewuenschteSpalte2, …)

Rückgabewert: array

Diese Funktion sucht beim aktuellen Mandanten nach Projekten die den 3 angegebenen Bedingungen entsprechen:

1. GenerelleSuche:
   Ein genereller Suchstring der alle Spalten durchsucht. Die üblichen Tricks wie Leerzeichen oder Minus funktionieren auch hier.
2. idSuchSpalte1+Spalte1Value:
   Der Id einer Spalte (z.B. ePrjPropVehicleProducer) und der Text-Wert der enthalten sein muss. Statt dem Id kann auch 0 verwendet warden um diese Suchbedingung zu überspringen. Ist der Text-Wert ein Leerstring, dann passen nur Projekte wo dieses Feld auch leer ist.
3. idSuchSpalte2+Spalte2Value:
   Wie bei 2.

Es gibt eine Liste mit allen gewünschten Spalten zurück.

Beispiel:
```
-- Suche nach „Test“ und Hersteller=Volvo; Gibt 3 Datenfelder pro Projekt zurück.
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
```

2.2.29 DuplicateProject

Syntax: DuplicateProject (String Filename)
Rückgabewert: string

Diese Funktion erzeugt eine binäre Kopie der angegebenen ols-Datei und erzeugte dafür automatisch

einen neuen Dateinamen im aktuellen Mandanten-Ordner. Der neue Pfad+Dateiname wird zurückgegeben.

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

2.2.30 GetProjectVersions

Syntax: GetProjectVersions (String Filename)
Syntax: GetProjectVersions (String Filename, int versionseigenschaften, …)
Rückgabewert: Array

Diese Funktion gibt ein Array mit allen Versionsnamen und -Beschreibungen von der angegebenen ols-Projektdatei zurück. Das Array enthält dabei abwechselnd Name und Beschreibung (beginnend mit Index 1 für Name). Das Array enthält also doppelt so viele Einträge wie das Projekt Versionen hat.

Alternativ kann man eine beliebige Anzahl von Versionseigenschaften (siehe versionGetProperty) übergeben. Dann werden statt der o.g. Eigenschaften diese zurückgegeben.

Beispiel:
```
versions = GetProjectVersions ("prj1000.ols");  --Diese Datei muss bereits existieren
size = #versions
for i=1, size/2 do
  MessageBox ("Version "..i.."=".. versions [i*2-1].."\n".. versions [i*2]);
end
```

2.2.31 OpenProjectVersion

Syntax1: OpenProjectVersion (string filename, string versionname)
Syntax2: OpenProjectVersion (string filename, number versionindex)
Rückgabewert: bool (true bei Erfolg)

Öffnet das angegebene Projekt in der gewünschten Version. Das Projekt muss später mit projectClose oder CloseAll geschlossen werden.

versionname: Name der gewünschten Version. Falls nicht eindeutig, wird der geringste Index verwendet.
versionindex: 0 für Original, 1 für erste Version…

Achtung: Für den zweiten Parameter ist der *Typ* ausschlaggebend. Wenn Sie also einen String (aus einer Datei) haben der eine Zahl als Text zum Inhalt war, ist das ein Versionsname. Mit „1*meinstring“ können Sie ihn als Index verwenden.

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

2.2.32 DeleteProjectVersion

Syntax1: DeleteProjectVersion (string filename, string versionname)
Syntax2: DeleteProjectVersion (string filename, number versionindex)
Rückgabewert: bool (true bei Erfolg)

Wie OpenProjectVersion, aber die die Version wird nicht geöffnet, sondern gelöscht.

Hinweis: Versionen die derzeit geöffnet sind, können nicht gelöscht werden.

Hinweis: Durch das Löschen einer Version, verschieben sich die darauf folgenden Versionsnummern. Wenn Sie mehrere Versionen löschen möchten, verwenden Sie die Versionsnamen oder löschen Sie anhand von Nummern in absteigender Größe.

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

2.2.33 OpenAndExport

Rückgabewert: bool (true bei Erfolg)

Kombiniert die Funktionen OpenProjectVersion, projectExport und projectClose. Parameter wie bei den genannten Funktionen.

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

2.2.34 ReactivateChecksums

Syntax: ReactivateChecksums ()
Rückgabewert: -

Falls ein Checksummenplug ein Problem verursacht während es nach Checksummen sucht oder sie berechnet, dann wird es automatisch von WinOLS deaktiviert um den Anwender zu schützen. Falls ihr Skript automatisch Checksummen suchen soll, dann möchten Sie evtl. diese Funktion nach Abschluss eines jeden Projektes aufrufen um eventuell deaktivierte Plugins wieder zu aktivieren.

Beispiel: ReactivateChecksums ();

2.2.35 StartUrl

Syntax: StartUrl (string url)

Öffnet ein Browserfenster mit der angegebenen URL.

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

2.2.36 ReadDirectory

Syntax: ReadDirectory (string filenameandwildcards)

Gibt ein Array mit den Datei- und Ordnernamen zurück.

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

2.2.37 CreateDirectory

Syntax: CreateDirectory (string full_path)

Erzeugt einen Ordner. Gibt im Fehlerfall 0 zurück, 1 wenn der Ordner angelegt wurde, 2 wenn er bereits vorher existierte.

Beispiel:
CreateDirectory ("c:\\my_dir");

2.2.38 MessagePump

Syntax: MessagePump ()

Bearbeitet anstehende Windows-Nachrichten im WinOLS Hauptthread. Diese Funktion sollte regelmäßig aufgerufen werden falls der LUA-Code sehr lange dauert oder sogar in einer Endlos-Schleife läuft, da Windows andernfalls anzeigt, dass WinOLS nicht mehr reagiert.

2.2.39 RemoveNonFilenameCharacters

Syntax: RemoveNonFilenameCharacters (string input)
Rückgabewert: String

Nimmt die Eingabezeichenfolge und entfernt Zeichen, die Windows in einem Dateinamen nicht akzeptiert, wie z. B. "?". Gibt die aktualisierte Zeichenfolge zurück.

2.2.40 GetLastError

Syntax: GetLastError()
Rückgabewert: Number

Gibt den Wert der gleichnamigen Windows-Funktion zurück. Aufgrund von automatischen Korrekturstrategien muss nicht hier nicht zwangsläufig die Ursache für das Scheitern eine WinOLS-Funktion drinstehen.

Wenn ein Import wg. NOREAD scheitert, setzt projectImport den Wert auf 30 (was dem Windows-Fehler ERROR_READ_FAULT entspricht).

2.2.41 timeGetTime()

Syntax: timeGetTime()
Rückgabewert: Number

Gibt den Wert der gleichnamigen Windows-Funktion zurück, also die Millisekunden seit Systemstart.

2.2.42 requirex

Syntax: requirex(String filename)
Rückgabewert: boolean

Funktioniert wie der normale require-Befehl, kann aber auch mit verschlüsselten Dateien (siehe WinOLS-Hilfe: luaX) umgehen. Eine Kennworteingabe ist dabei nicht möglich, aber falls das laufende Skript bereits mit einem Kennwort geöffnet wurden, dann wird dies automatisch für die hier angegebene Datei probiert.

Beispiel:
requirex ("mylib"); -- probiert mylib, mylib.lua, mylib.luax

2.2.43 usestrict

Syntax: usestrict()
Rückgabewert: -

Ähnlich wie beim gleichnamigen JavaScript Modus müssen hierdurch Variablen explizit deklariert werden. Dies schützt einem vor Tippfehlern bei Variablennamen.

Beispiel:
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 Kontextphilosophie

Zahlreiche Funktionen verwenden zu Ihrer Ausführung einen Kontext, den Sie aus dem aktuellen Zustand von WinOLS beziehen. Beispielsweise bezieht sich die Funktion „projectSave()“ auf das Projekt, was derzeit in WinOLS gerade aktiv ist. LUA für WinOLS verhält sich also ähnlich wie ein Makro-Rekorder, der Tastenkombinationen abspielt.

Sie haben die Möglichkeit mit verschiedenen Funktionen die verschiedenen Kontexte zu ändern. Wollen Sie beispielsweise ein anderes Projekt als das aktive abspeichern, dann können Sie das aktive Projekt abfragen, ein anderes aktivieren, das aktive Projekt speichern und dann das alte Projekt wieder aktivieren.

Existiert ein notwendiger Kontext nicht (z.B. wenn Sie „projectSave()“ aufrufen, ohne dass überhaupt ein Projekt geöffnet ist), dann scheitert die Funktion und gibt einen Fehlerwert zurück.

2.4 Kontext: Projekt

2.4.1 projectGetProperty

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

Diese Funktion fragt eine der diversen Texte aus den Projekteigenschaften ab. Benutzen Sie sie z.B. um zu überprüfen welche Checksumme für das Projekt eingetragen ist.

Der Parameter „id“ muss einen der folgenden Werte haben:

- 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, …

Außerdem sind auch folgende Werte (nur zum Lesen) zulässig: ePrjImportFilename, ePrjImportPath, ePrjFilename

Ab WinOLS 5.62.01 können folgende Projekteigenschaften (für Original mit 1 als zweiten Parameter für die Version) abgefragt werden: ePrjPropChecksum8Bit, ePrjPropChecksum8BitCpu, ePrjPropChecksum8BitEpr, ePrjPropChecksumMD5, ePrjPropChecksumMD5Cpu, ePrjPropChecksumMD5Epr, ePrjPropChecksumSHA1, ePrjPropChecksumSHA256.

Hinweis: Diese Funktion fragt die aktuelle Werte aus den Projekteigenschaften ab. Eine Suche in den Projektdaten führt sie nicht durch.

Beispiel 1:
```
MessageBox (projectGetProperty(ePrjPropClientName));
```

Beispiel 2: Implizite Konvertierung des Rückgabewertes in eine Zahl um einen Vergleich zu ermöglichen:
```
if (ePrjDeveloping==1*projectGetProperty (ePrjProjectStatus)) then
  MessageBox ("ePrjDeveloping");
end
```

2.4.2 projectSetProperty

Syntax: projectSetProperty (Number id, String newvalue)
Rückgabewert: bool

Diese Funktion ändert einen der diversen Texte aus den Projekteigenschaften auf einen neuen Wert.

Der Parameter „id“ muss einen der folgender Werte wie bei projectGetProperty haben. Nicht unterstützt werden folgende Werte: ePrjPropEcuSoftwaresize, ePrjPropEcuChecksum, ePrjFileCreatedOn, ePrjFileModifiedOn.

Hinweis: Diese Funktion ändert die Projektdaten und markiert das Projekt als geändert.

Beispiel: projectSetProperty(ePrjPropVehicleProducer, "VW");

2.4.3 projectClose

Syntax: projectClose (bool bDeleteFile=FALSE)
Rückgabewert: bool

Diese Funktion schließt das aktive WinOLS Projekt und alle Fenster die zu diesem Projekt gehören. Ungespeicherte Änderungen gehen dabei verloren. Wenn Sie als Parameter den Wert „TRUE“ übergeben, wird die Projektdatei nicht nur geschlossen, sondern auch von der Festplatte gelöscht.

Beispiel: projectClose ();

2.4.4 projectSave

Syntax: projectSave (bool bNeueVersionErzeugen, String VersionsName=”CreatedByLua”, String VersionsBeschreibung=””)
Rückgabewert: bool

Diese Funktion speichert das aktive WinOLS Projekt sofern es Änderungen gibt. Wird ein Original mit Änderungen im Hexdump gespeichert, dann wird eine Version mit dem Namen „CreatedByLua“ angelegt. Wird eine Version mit Änderungen im Hexdump gespeichert, dann wird die gleiche Version überschrieben. Sie können das Erstellen einer neuen Version erzwingen indem Sie TRUE als ersten Parameter übergeben. Parameter 2 und 3 sind Versionsname und -Beschreibung und werden nur beachtet falls eine neue Version geschrieben wird.

Beispiel: projectSave (TRUE); -- Hier niemals true verwenden. Sondern TRUE

2.4.5 projectExport

Syntax: projectExport (String Dateiname, Number Typ, String ZippedDateiname=““, Number BdmToGoFlags=0, Number olsxCustomerNumber=0, String olsxPassword=””, Number ExportFlags=0)
Rückgabewert: bool

Diese Funktion exportiert das aktuelle Projekt als Datei. Der Parameter „Typ“ legt das Dateiformat fest und kann folgende Werte haben: eFiletypeBinary, eFiletypeWinOLS, eFiletypeWinOLSAll, eFiletypeIntelHex, eFiletypeMotorolaHex, eFiletypeBdmToGo, eFiletypeWinOLSX, eFiletypeWinOLSAllX. Wenn Sie die erforderlichen Plugins haben, sind auch folgende Werte zulässig: eFiletypeVBF, eFiletypeCMDSlave, eFiletypeFLS, eFiletypeNewGenius, eFiletypeODX, eFiletypeCFF, eFiletypeCRE, eFiletypeBFlashSlave, eFiletypeSMRF, eFiletypeAutotunerSlave, eFiletypeAutoflasherSlave, eFiletypeMagicmotorsportSlave.

Übergibt man dem Parameter „ZippedDateiname“ einen Dateinamen, dann wird die exportierte Datei dort rein verschoben. Um die Datei einer bestehenden zip-Datei hinzuzufügen, reicht es ein „+“ an den Anfang von ZippedDateiname zu schreiben.

Bei BdmToGo-Dateien:
Der Parameter „BdmToGoFlags“ gibt bei BdmToGo Dateien die Exportoptionen an und dann eine Kombination folgender Werte annehmen: eBdmToGoProgramEprom, eBdmToGoProgramEprom2, eBdmToGoProgramEEprom, eBdmToGoProgramProcessor, eBdmToGoNoReimport. Sollen Bereiche vor der Programmierung verglichen werden, dann müssen diese zuvor per Kommentar als BDM1, BDM2 oder BDM3 markiert worden sein (entweder durch einen Import oder durch die Funktion projectAddCommentAt).

ExportFlags:
Dies kann eExportRemoveChecksums oder eExportActiveElementOnly (oder eine binaryor-Kombination) sein.
Hinweis: Dies war früher der Parameter "Bool bRemoveChecksums", aber da eExportRemoveChecksums gleich 1 ist, ist dies immer noch kompatibel.

Hinweis: Genau wie in C, müssen auch in LUA Backslashes die direkt in Strings vorkommen doppelt eingegen werden, da sonst die Einleitung eines Steuerzeichens angenommen wird. Siehe Beispiel.

Hinweis: Die Dateiformate eFiletypeBinary, eFiletypeIntelHex und eFiletypeMotorolaHex exportieren immer nur das aktuell gewählte Element.

Hinweis: Der Dateiname darf Umgebungsvariablen, eingeschlossen von Prozentzeichen enthalten. WinOLS ersetzt diese Texte durch ihren aktuellen Wert. Außerdem können Sie Platzhalter verwenden um

den Pfad / Dateinamen vom Import zu verwenden (Beispiel 3+4). Weil diese nicht immer vorhanden sind, müssen Sie Reserve-Werte angeben. Der Import-Pfad endet immer auf einem Backslash, der Import-Dateiname ist immer ohne Suffix.

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

2.4.6 projectImport

Syntax: projectImport (String Dateiname, Number Typ=eFiletypeAuto, Number Offset=0, String olsxPassword="", Number Options=0)
Rückgabewert: bool

Diese Funktion importiert die angegebene Datei als Version in das aktuelle Projekt. Standardmäßig versucht WinOLS immer den Pfad des Lua Skripts als aktuelles Verzeichnis zu setzen. Sicherheitshalber sollte man trotzdem absolute Dateipfade (wie im Beispiel) und nicht relative (mit unvollständigem Namen) verwenden.

Der optionale Parameter „Typ" legt das Dateiformat fest und kann folgende Werte haben: eFiletypeAuto, eFiletypeBinary, eFiletypeWinOLS, eFiletypeIntelHex, eFiletypeMotorolaHex, eFiletypeEDX, eFiletypeBdmToGo, eFiletypeWinOLSX, eFiletypeVBF, eFiletypeCMDSlave, eFiletypeVBF, eFiletypeFLS, eFiletypeODX, eFiletypeCFF, eFiletypeCRE, eFiletypeBFlashSlave, eFiletypeSMRF, eFiletypeAutotunerSlave, eFiletypeAutoflasherSlave, eFiletypeIni.

Wird kein Typ angegeben, dann wird der Wert „eFiletypeAuto" angenommen und der Dateityp wird automatisch ermittelt. Dies funktioniert mit den meisten Dateiarten, einschließlich Typen die durch Import-Plugins unterstützt werden und winolsskript-Dateien. Für diese kann der „Options" Parameter eOptionWinOLSSkriptTestOnly verwendet werden. Dann wird das Skript nicht angewendet, sondern es wird nur geprüft, ob die requires-Bedingungen erfüllt werden.

Der optionale Parameter „Offset" gibt einen Offset an um den die importierten Daten nach hinten verschoben werden soll. Dieser Parameter wird nur bei den Dateiformaten beachtet, bei denen auch im WinOLS eine Eingabe möglich ist. Zusätzlich bezieht sich der Import bei den meisten Dateiformaten auch immer auf das aktuelle Element. Sie benötigen den Offset also nur, wenn Sie nicht am Anfang eines Elements mit dem Import beginnen möchten. Der Offset wird nur für Versionen unterstützt, nicht für das Original.

Der optionale Parameter "Options" kann eOptionCmdCutFF oder eOptionCmdIgnore sein. Er beantwortet die Rückfrage für den Import von CMD-Dateien. (Der Default ist eOptionCmdIgnore.)

Hinweis: Genau wie in C, müssen auch in LUA Backslashes die direkt in Strings vorkommen doppelt eingegen werden, da sonst die Einleitung eines Steuerzeichens angenommen wird. Siehe Beispiel.

Hinweis: Neue Versionen erhalten den Text „Importierte Version" als Versionsbezeichnung. Wird eine andere Bezeichnung gewünscht, verwenden Sie versionGetProperty / versionSetProperty.

Hinweis: Der Dateiname darf Umgebungsvariablen, eingeschlossen von Prozentzeichen enthalten. WinOLS ersetzt diese Texte durch ihren aktuellen Wert.

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

Beispiel 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);
```

2.4.7 projectMail

Syntax: projectMail (String An, String Subject, String Nachricht, String Filename="");
Rückgabewert: bool

Verschickt eine Mail (optional mit einer Datei als Anhang) direkt aus WinOLS heraus. Zum Versand des Mails wird die Standard MAPI Anwendung des Rechners verwendet. Das ist die gleiche Anwendung, die verwendet wird wenn Sie im Browser auf einer E-Mail Adresse in einer Webseite klicken.

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

2.4.8 projectSearchChecksums

Syntax: projectSearchChecksums ()
Rückgabewert: bool

Diese Funktion sorgt dafür, dass die Checksummen neu gesucht werden. Je nach Konfiguration geschieht das ohnehin beim Importieren.

Die Funktion gibt TRUE zurück, es möglich war die Checksummensuche auszuführen. (FALSE z.B. wenn kein Projekt offen war). TRUE bedeutet aber nicht, dass tatsächlich Checksummen gefunden wurden. Um das abzufragen verwenden Sie bitte die Funktion projectStatChecksums.

Beispiel: rc = projectSearchChecksums ();

2.4.9 projectAddChecksum

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

Fügt einen manuellen (additiven) Checksummenblock hinzu. Je nach Parameteranzahl:

- nur mit Anzeige des Ergebnisses
- mit (optional automatischem) Ausgleich des Ergebnisses
- mit (optional automatischem) Ausgleich des Ergebnisses über einen Bereich (Vollbyte)

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

2.4.10 projectApplyChecksums

Syntax: projectApplyChecksums ()
Rückgabewert: bool

Diese Funktion sorgt dafür, dass alle nicht ausgeglichenen Checksummen berechnet werden. Je nach Projektdatei geschieht das ohnehin automatisch.

Die Funktion gibt TRUE zurück, wenn mindestens eine Checksumme neu berechnet wurden oder false, wenn keine Änderungen notwendig waren oder keine Projektdatei oder Checksummen vorhanden waren.

Beispiel: rc = projectApplyChecksums ();

2.4.11 projectStatChecksums

Syntax: projectStatChecksums ()
Rückgabewert: bool

Diese Funktion gibt Informationen über die aktuelle Checksummen zurück. Ist der Rückgabewert -1, dann enthält das aktuelle Projekt keine Checksummen oder es gibt kein aktuelles Projekt. Ist der Rückgabewert größer als 0, dann beziffert dieser Wert die Anzahl der nicht ausgeglichenen Checksummen. Der Rückgabewert 0 bedeutet, dass es mindestens eine Checksumme gibt, aber alle Checksummen ausgeglichen sind.

In der Regel sollte ein Rückgabewert ungleich 0 Anlass zur Sorge geben und Ihr Skript sollte dafür sorgen, dass dieses Projekt im jetzigen Zustand nicht ausgeliefert wird.

Beispiel: rc = projectStatChecksums ();

2.4.12 projectGetChecksumName

Syntax: projectGetChecksumName (EChkInfo ChkInfo=eChecksumName)
Rückgabewert: string

Diese Funktion gibt den Namen der aktuell gefundenen Checksumme zurück. Dies ist der gleiche Text, der auch in den Projekteigenschaften und in der Projektliste angezeigt wird. Wird als optionaler Parameter „eChecksumNumber“ übergeben, dann wird nicht der Name, sondern die Checksummennummer zurückgegeben, z.B. „OLS265“. Wird der Parameter „eChecksumVersion“ übergeben, dann wird die Versionsnummer (z.B. „2.05“) zurückgegeben.

Wurde keine Checksumme gefunden, dann wird ein leerer String zurückgegeben.

Beispiel: rc = projectGetChecksumName ();

2.4.13 projectGetChecksumOptionStatus

Syntax1: projectGetChecksumOptionStatus (Number index_or_id)
Syntax2: projectGetChecksumOptionStatus (String SuchstringMitWildcards)
Rückgabewert: bool

Diese Funktion gibt den Status (0 oder 1) der Checksummen-Option vom Typ „Checkbox“ (gibt es z.B. bei OLS285) zurück, die durch den 0-basierten index angegeben wurde.

Beispiel: rc = projectGetChecksumOptionStatus (0);

2.4.14 projectGetChecksumOptionText

Syntax1: projectGetChecksumOptionText (Number index_oder_id)
Syntax2: projectGetChecksumOptionText (String SuchstringMitWildcards)
Rückgabewert: string

Diese Funktion gibt den Beschriftungstext der Checksummen-Option zurück, die durch den 0-basierten index angegeben wurde.

Beispiel: rc = projectGetChecksumOptionText (0);

2.4.15 projectGetChecksumOptionType

Syntax1: projectGetChecksumOptionStatus (Number index_oder_id)
Syntax2: projectGetChecksumOptionStatus (String SuchstringMitWildcards)

Rückgabewert: ECOptType

Diese Funktion gibt den Typ der Checksummen-Option zurück, die durch den 0-basierten index angegeben wurde. Mögliche Rückgabewerte sind:

- eCOTNone => Nicht vorhanden
- eCOTCheckbox => Das ist eine Checkbox
- eCOTText => Das ist ein Text. (Es gibt also keinen Status)
- eCOTCheckboxSearchAgain => Wie eCOTCheckbox, aber projectSearchChecksums muss erneut aufgerufen werden, wenn man den Status dieser Checkbox ändert.

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

2.4.16 projectSetChecksumOptionStatus

Syntax1: projectSetChecksumOptionStatus (Number index_oder_id, Number Wert)
Syntax2: projectSetChecksumOptionStatus (String SuchstringMitWildcards, Number Wert)
Rückgabewert: bool

Diese Funktion setzt den Status (0 oder 1) der Checksummen-Option vom Typ „Checkbox“ (gibt es z.B. bei OLS285) zurück, die durch den 0-basierten index angegeben wurde.

Wichtig: Bei Checksummen mit dem Typ eCOTCheckboxSearchAgain muss danach die Checksummensuche erneut aufgerufen werden. (Ab WinOLS 5.32 wird das automatisch gemacht.)

Beispiel: projectSetChecksumOptionStatus (0, 1);

2.4.17 projectGetElementOffset

Syntax: projectGetElementOffset ()
Rückgabewert: Number

Diese Funktion gibt den Offset zurück, also die Adresse mit der das erste Byte des aktuellen Elements bezeichnet wird. Oftmals ist das 0. Durch Checksummen oder manuelle Änderungen bei den Elementen sind auch andere Adressen möglich.

Beispiel: rc = projectGetElementOffset ();

2.4.18 projectGetElement

Syntax: projectGetElement ()
Rückgabewert: Number

Diese Funktion gibt den Identifier des aktuell Elements zurück. Der Rückgabe ist einer der folgenden Werte: eElementNone, eElementHeader, eElementProcessor, eElementEprom, eElementEprom2, eElementEEprom, eElementConfiguration

Beispiel: rc = projectGetElement ();

2.4.19 projectSetElement

Syntax: projectSetElement (Number ElementId)
Rückgabewert: bool

Diese Funktion wählt das aktuell sichtbare Element und damit den Bereich der bei einem Import- oder Exportvorgang verwendet wird. Als ElementId werden alle Werte aus projectGetElement akzeptiert.

Beispiel: rc = projectSetElement (eElementProcessor);

2.4.20 projectSetElementRanges

Syntax: projectSetElementRanges (String RangeString)
Rückgabewert: bool

Diese Funktion überschreibt alle derzeit eingetragenen Definitionen für die Element Bereiche mit den Definitionen aus dem Parameter. Dieser hat das Format ELEMENTNAME:VON-BIS. „Von“ und „Bis“ sind dabei Dezimalzahlen. Weitere Bereiche können (durch Semikola getrennt) angehängt werden.

Beispiel: rc = projectSetElementRanges ("Motor / Eprom:0-1048575;Motor / Prozessor:1048576-1507327");

2.4.21 projectAddCommentAt

Syntax: projectAddCommentAt (Number VonAdresse, Number BisAdresse, String Text, Number FrameColor=-1, Number BackColor=-1)
Rückgabewert: bool

Diese Funktion fügt einen Kommentar für den angegebenen Bereich ein. Die Rahmen- und Hintergrundfarben sind optional. Die Adresse bezieht sich auf das aktuell gewählte Element.

Hinweis: Diese Funktion ändert die Projektdaten und markiert das Projekt als geändert.

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

2.4.22 projectGetCommentAt

Syntax: projectGetCommentAt (Number Adresse)
Rückgabewert: String

Diese Funktion prüft ob es an der angegebenen Adresse einen Kommentar gibt und gibt ggf. den hinterlegten Text zurück. Im Fehlerfall wird ein Leerstring zurückgegeben. Die Adresse bezieht sich auf das aktuell gewählte Element.

Beispiel: rc = projectGetCommentAt (100);

2.4.23 projectDelCommentAt

Syntax: projectDelCommentAt (Number Adresse)
Rückgabewert: bool

Diese Funktion prüft ob es an der angegebenen Adresse einen Kommentar gibt und löscht ihn gegebenenfalls. Die Adresse bezieht sich auf das aktuell gewählte Element.

Hinweis: Diese Funktion ändert die Projektdaten und markiert das Projekt als geändert.

Beispiel: rc = projectDelCommentAt (100);

2.4.24 projectGetAt

Syntax: projectGetAt (Number Adresse, Number Datentyp=eByte, Number nWerte=1)
Rückgabewert: Number/Array

Diese Funktion gibt den Wert der Daten an der angegebenen Adresse zurück. Die Adresse bezieht sich auf das aktuell gewählte Element. Der optionale Parameter „Datentyp“ beschreibt die die Organisation und Bitbreite mit der der Wert an der angegebenen Adresse steht. Zulässige Werte für diesen Paramter sind eByte, eLoHi, eHiLo, eLoHiLoHi, eHiLoHiLo, eFloatLoHi, eFloatHiLo, eFloatHiLo, eDoubleLoHi, eDoubleHiLo, eBitLoHi, eBitHiLo.

Hinweis: Im Fehlerfall wird zur Unterscheidung nicht 0, sondern eEmptyvalue zurückgegeben was (derzeit) dem Wert -99999 entspricht.

Wenn 3 Parameter übergeben werden, dann können mehrere hintereinanderliegende Werte abgefragt werden und es wird ein Array zurückgegeben.

Beispiel: rc = projectGetAt (100);

2.4.25 projectSetAt

Syntax 1: projectSetAt (Number Adresse, Number Wert, Number Datentyp=eByte, Number Mode=eSetAbsolute)
Syntax 2: projectSetAt (Number Adresse, String Werte, Number Datentyp=eByte, Number Mode=eSetAbsolute)

Rückgabewert: bool

Diese Funktion schreibt den/die Werte an der angegebenen Adresse in den Hexdump. Die Adresse bezieht sich auf das aktuell gewählte Element. Der optionale Parameter „Datentyp“ beschreibt die die Organisation und Bitbreite mit der der Wert an der angegebenen Adresse steht. Zulässige Werte für diesen Paramter sind eByte, eLoHi, eHiLo, eLoHiLoHi, eHiLoHiLo, eFloatLoHi, eFloatHiLo, eDoubleLoHi, eDoubleHiLo, eBitLoHi, eBitHiLo.

Wenn Sie mehrere Werte als String übergeben, erwartet WinOLS Hex-Werte. Dezimalwerte können Sie mit dem Postfix "M" angeben. Bei Verwendung des eByte-Datentyps kann der String ??-Platzhalter für Werte enthalten, die nicht ersetzt werden sollen.

Hinweis: Diese Funktion ändert die Projektdaten und markiert das Projekt als geändert.

Hinweis: Sie können eEmptyvalue als Wert übergeben um die Zelle auf den Originalwert zurückzusetzen.

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

2.4.26 projectSetOrg

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

Diese Funktion setzt die Bytes des gesamten Projekts / der angegebenen Adresse / des Adressbereichs auf die ursprüngliche Version zurück.

Beispiel: projectSetOrg();

Erfordert WinOLS 5.38.05

2.4.27 projectFindBytes

Syntax 1: projectFindBytes (Number StartAdresse, Number OrgVer, Number ErstesByte, …)
Syntax 2: projectFindBytes (Number StartAdresse, Number OrgVer, String AlleBytes, Number Datentyp=eByte)
Syntax 3: projectFindBytes (Number StartAdresse, Number EndAdresse, Number OrgVer, Number ErstesByte, …)
Syntax 4: projectFindBytes (Number StartAdresse, Number EndAdresse, Number OrgVer, String AlleBytes, Number Datentyp=eByte)
Rückgabewert: Number oder Array

Diese Funktion sucht im aktuellen Element entweder im Original (0) oder in der Version (1) nach der Bytesequenz. Jedes Byte wird einzeln als Parameter angehängt. Die Funktion akzeptiert beliebig viele Parameter.

Alternativ können alle Bytes in Form eines Strings übergeben werden. Dabei ist egal ob als Trennzeichen ein Leerzeichen, Komma oder Tab-Zeichen verwendet wird. Auch das „0x“ Präfix ist optional, WinOLS erwartet immer Hex-Zahlen. Des Weiteren können Sie einzelne Bytes im Suchstring auch durch ?? als Platzhalter ersetzen.

Wenn StartAdresse >=0, dann gibt die Funktion die Adresse der nächsten Fundstelle an / nach dieser Adresse zurück. Ist StartAdresse -1, dann gibt die Funktion ein Array mit allen Fundstellen zurück. (Das Array ist leer, falls nichts gefunden wurde.)

Der Parameter EndAdresse ist optional. Wenn er verwendet wird, darf er nicht 0 oder 1 sein um Verwechslungen mit den OrgVer Parameter zu vermeiden.

Ab WinOLS 5.15 kann hinter dem String noch der Datentyp festgelegt werden. Zulässig sind: eByte, eHiLo, eLoHi, eHiLoHiLo, eLoHiLoHi.

Alle Adressen beziehen sich auf das aktuell gewählte Element.

Hinweis: Wird der Suchstring nicht gefunden, dann gibt die Funktion -1 oder ein Leeres Array zurück.

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

2.4.28 projectReplaceBytes

Syntax 1: projectReplaceBytes (Number StartAdresse, Number EndAdresse, String AlleSucheZellen, String AlleErsetzenZellen, Number SucheOrgVer=0, Number LimitAnzahlErgebnisse=1, Number Datentyp=eByte, Mode=eSetAbsolute)
Syntax 1: projectReplaceBytes (Number StartAdresse, Number EndAdresse, String AlleSucheZellen, Number ErsetzenWert, Number SucheOrgVer=0, Number LimitAnzahlErgebnisse=1, Number Datentyp=eByte, Mode=eSetAbsolute)
Erfordert WinOLS 5.27

Rückgabewert: Number (Anzahl der ersetzten Ergebnisse)

Diese Funktion sucht im aktuellen Element entweder im Original (0) oder in der Version (1) nach der Sequenz von Zellen. Diese wird in Form eines Strings übergeben werden. Dabei ist egal ob als Trennzeichen ein Leerzeichen, Komma oder Tab-Zeichen verwendet wird. Auch das „0x“ Präfix ist optional, WinOLS erwartet immer Hex-Zahlen. Des Weiteren können Sie einzelne Bytes im Suchstring auch durch ?? als Platzhalter ersetzen.

Zum Ersetzen können Sie einen String oder einen einzelnen Wert angeben.

Dezimalwerte können Sie im String mit dem Postfix "M" angeben. Bei Verwendung des eByte-Datentyps kann der String ??-Platzhalter für Werte enthalten, die nicht ersetzt werden sollen.

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);

2.4.29 projectFindSimilarProjects

Ab WinOLS 5.71.24 ist diese Funktion veraltet, stattdessen sollte projectFindSimilarProjectsSql verwendet werden, was schneller/moderner ist. Parameter und Rückgabewerte sind identisch.

Die gefundenen Projekte können aufgrund der anderen Technologie in Einzelfällen minimal abweichen.

2.4.30 projectFindSimilarProjectsSql

Syntax: projectFindSimilarProjectsSql (Number MinPercent, Number MaxErgebnisAnzahl, Number idGewuenschteSpalte1, Number idGewuenschteSpalte2, …)

Rückgabewert: array

Diese Funktion sucht beim aktuellen Mandanten nach Projekten die dem aktuellen ähnlich sind. Es gibt eine Liste zurück die nach absteigender Relevanz sortiert ist. Diese Liste enthält die Relevanz und alle gewünschten Spalten. Die Ähnlichkeitsberechnung wird nicht von den gewünschten Spalten beeinflusst. Das aktive Projekt wird automatisch aus den Ergebnissen entfernt.

Datenbereiche:
Ab WinOLS 5.11 kann diese Funktion auch Projekten mit ähnlichen Datenbereichen finden. Dies wird durch negative Werte symbolisiert. Um dieses Feature zu aktivieren muss für MinPercent also ein Minuszeichen vor den Wert gesetzt werden (Beispiel: -80). Und der Ergebnisliste werden Einträge bei denen nur der Datenbereich ähnlich ist, auch durch einen negativen Wert (Beispiel: -100) gekennzeichnet.

Übereinstimmende Eigenschaften:
Ab WinOLS 5.50 kann diese Funktion auch Projekte zurückgeben, bei denen nur eine Projekteigenschaft übereinstimmt (und nicht die Ähnlichkeit, sie wird als 0 zurückgegeben). Um dies zu aktivieren, übergeben Sie das Flag eFSPAllowPropertyMatches nach MaxNumberOfResults.

Erweiterte Relevanzinfos:
Ab WinOLS 5.52 kann WinOLS drei statt einer Relavanz-Infos zurückgeben. Die erste bleibt dabei gleich, die nächsten beiden sind die Projektähnlichkeit und die Datenbereichsähnlichkeit. Um dies zu aktivieren, übergeben Sie das Flag eFSPTrippleRelevance nach MaxNumberOfResults. Die genaue Position des Flags wird dabei ignoriert, WinOLS gibt die 3 Relevanzinfos immer in den ersten 3 Spalten zurück.

Beispiel:

```
list = projectFindSimilarProjectsSql (80, 9999, ePrjFilename, ePrjPropVehicleProducer, ePrjUserdef1);
nColumns = 4;                -- Vier, weil wir die Relevanz und 3 gewünschte Spalten erhalten
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
```

2.4.31 projectImportFromOls

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

Rückgabewert: Number (true im Erfolgsfall)

Importiert aus dem Quellprojekt Kennfelder und Hexdump-Werte in das aktuelle Projekt in eine neue Version. Die Projekte müssen die gleiche Größe haben, ein Offset wird nicht unterstützt.

iVersion ist der Index der Version von der die Daten übernommen wird. 0 ist das Original. Falls hier -1 übergeben wird, werden keine Versionsdaten übernommen.

WinOLS5:
Statt dem Index der Version können Sie auch den Namen der Version übergeben.

Flags:
Hier können mehrere Werte (kombiniert durch binaryor(a, b)) übergeben werden:
eImportSkipEEprom: Der Inhalt des (virtuellen) EEproms wird nicht übernommen
eImportOnlyDataArea: Nur der Inhalte den Datenbereiche wird übernommen

eImportSkipMaps: Die Kennfelder werden nicht übernommen.

WinOLS5:
Folgende Parameter werden zusätzlich unterstützt:
eImportMapsRelative: Kennfeldänderungen relativ übertragen
eImportMapsPercent: Kennfeldänderungen als Prozent übertragen
eImportSkipInsideMaps: Änderungen innerhalb von Kennfeldern nicht übertragen
eImportSkipOutsideMaps: Änderungen außerhalb von Kennfeldern nicht übertragen
eImportOnlyChanged: Nur geänderte Bytes importieren. (Kann für Kennfelder mit Relativ/Prozent kombiniert werden)
(Die ersten 3 Flags in dieser Liste schließen sich gegenseitig aus).

Ab WinOLS 5.67.02:
eImportUseElementInformation: Verwendet Elementadressen+Offset
eImportDontUseElementInformation: Verwendet Rohadressen+Offset (also „Alle Elemente“)
(Diese Flags schließen sich gegenseitig aus. Es wird empfohlen immer eines der beiden anzugeben, denn andernfalls nutzt WinOLS aus Kompatibilitätsgründen die zuletzt genutzte Einstellung beim manuellen Import aus einer ols-Datei.)

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

2.4.32 projectFindMap

Syntax: projectFindMap (String Kriterium, String Wert, Number StartAddress=0)

Rückgabewert: Number/Array

Sucht im aktuellen Projekt nach einem Kennfeld mit dem gewünschten Namen oder Id. Kriterium kann „Name“ oder „IdName“ sein.

Wenn StartAddress>=0: Gibt die Startadresse des Kennfelds (in alle-Elemente-Zählweise) zurück oder -1.
Wenn StartAddress==-1: Gibt ein Array mit allen Startadressen zurück.

Beispiel:
adr = projectFindMap ("IdName", "MyMapId”);

2.4.33 projectAddMap

Syntax: projectAddMap ()
Rückgabewert: bool

Fügt dem aktuellen Projekt ein Kennfeld hinzu, was dann mit windowSetMapProperties eingerichtet werden kann.

Beispiel: projectAddMap();

2.4.34 projectDelMap

Syntax: projectDelMap (Number StartAddress)
Syntax: projectDelMap (String StringWithWildcards)
Rückgabewert: Number

Löscht alle Kennfelder die an der gegebenen Startadresse anfangen. Gibt die Anzahl der gelöschten Kennfelder zurück.

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

2.4.35 projectImportCsvJson

Syntax: projectImportCsvJson (String Filename, String MatchModes=”id,address”)
Rückgabewert: bool
Erfordert WinOLS5

Importiert die angegebene csv oder json-Datei in das aktuelle Projekt. Der MatchModes-String definiert welche Kriterien zum Matching mit vorhandenen Kennfeldern zulässig sind und was bevorzugt werden soll (falls die Datei beide Information hat).

Beispiel: projectImportCsvJson ("d:\test.csv", "id");

2.4.36 projectImportMapPack

Syntax: projectImportMapPack (String Filename, Number Offset=0, String Prefix="", Number bSkipDuplicates=false, Number bIgnoreAxis=false, Number bIgnoreTexts=false)
Rückgabewert: bool

Importiert die angegebene kp-Datei in das aktuelle Projekt.

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

2.4.37 projectSetRight

Syntax: projectSetRight (Number idRight, bool bNewState)
Rückgabewert: bool

Ändert das angegebene Recht für das aktuelle Projekt. Erfordert mindestens WinOLS 4.69 (siehe eWinOLSMajor, eWinOLSMinor).

Diese Funktion scheitert und gibt false zurück, wenn der registrierte WinOLS Anwender nicht das Recht hat die Projektrechte zu ändern (siehe projectSetRightsOwner).

Gültige Werte für idRight sind:
ePRExportBinary, ePRExportOLS, ePRExportBdm, ePRExportKp, ePRExportClipboard, ePRExportOther, ePRWriteEprom, ePRWriteBdm, ePROtherTransMapContent, ePROtherTransMapProp, ePRAccessHexdump, ePRAccessMaps, ePRAccessMapList

Beispiel: projectSetRight (ePRExportKp, false); --Kennfeldpaket export verbieten

2.4.38 projectSetRightsOwner

Syntax: projectSetRightsOwner (Number EvcKundenNummer)
Rückgabewert: bool

Ändert den Eigentümer der Rechte des aktuellen Projektes. Wenn dieser nicht mit dem aktuell

registrierten WinOLS-Anwender identisch ist und mindestens ein Recht nicht gewährt ist, werden alle darauf folgenden Versuche die Projektrechte oder den Eigentümer der Projektrechte zu ändern, scheitern. Dies bezieht sich sowohl auf lua, als auch auf die GUI.

Diese Funktion scheitert und gibt false zurück, wenn der registrierte WinOLS Anwender nicht das Recht hat die Projektrechte zu ändern.

Beispiel: projectSetRightsOwner (33333);

2.4.39 projectCountDifferentBytes

Syntax: projectCountDifferentBytes (bool bReallyReturnDifferences=FALSE)
Rückgabewert: int

Auch wenn der Name etwas anderes suggeriert, zählt diese Funktion die Anzahl der **identischen** Bytes zwischen aktueller Version und dem Original des Projektes. Aus Kompatibilitätsgründen kann das nicht korrigiert werden, aber wenn Sie TRUE als Parameter übergeben, werden wirklich die unterschiedlichen Bytes gezählt.

Eine eventuelle Referenzversion wird ignoriert. Die Funktion betrachtet immer das gesamte Projekt (ignoriert also das aktuelle Element). Außerdem beachtet sie Änderungen durch Checksummen, auch wenn diese ausgeblendet wurden (F12>Hexdump).

Beispiel: nBytesDifferent = projectCountDifferentBytes (TRUE);

2.4.40 projectDelFolder

Syntax: projectDelFolder (String FolderName)
Rückgabewert: bool

Löscht den angegebenen Ordner und die Kennfelder darin.

Beispiel: projectDelFolder ("Foldername");

2.4.41 projectDelDuplicateMaps

Syntax: projectDelDuplicateMaps (bool bOnlyCompareAddressesAndSizes, bool bOnlyWhenInTheSameFolder)
Rückgabewert: Number

Löscht doppelte Kennfelder im aktuellen Projekt. Entspricht dem Kennfeld-Teil der WinOLS-Funktion „Projekt > Doppelte Objekte suchen“ (allerdings ohne die Rückfragen). Die Funktion gibt die Anzahl der gelöschten Kennfelder zurück oder im Fehlerfall -1.

Beispiel: projectDelDuplicateMaps (FALSE, TRUE);

2.4.42 projectAddrCpuToRaw

Syntax: projectAddrCpuToRaw (Number Address)
Rückgabewert: Number

Rechnet die Adresse vom CPU-Format (wie es normalerweise im WinOLS angezeigt wird) in die rohe Adresse um, die man sieht, wenn man „alle Elemente“ wählt. Wenn die Adresse nicht konvertiert werden

kann, wird 0xFFFFFFFF zurückgegeben.

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

2.4.43 projectAddrRawToCpu

Syntax: projectAddrRawToCpu (Number Address)
Rückgabewert: Number

Rechnet die Adresse vom rohen Format (das man sieht, wenn man „alle Elemente“ wählt) um in die CPU-Adresse (wie es normalerweise im WinOLS angezeigt wird). Wenn die Adresse nicht konvertiert werden kann, wird 0xFFFFFFFF zurückgegeben.

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

2.4.44 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)
Rückgabewert: Number
Erfordert WinOLS5

Führt die Funktion „Änderungen übernehmen: Automatisch“ aus. Dabei werden Kennfelder / Werte aus dem Quellprojekt in das Zielprojekt übernommen. Der Offset wird dabei automatisch ermittelt. Die Richtigkeit/Vollständigkeit der importierten Daten kann nicht garantiert werden. Die Verwendung dieser Funktion in einem automatischen Prozess ohne menschliche Kontrolle ist daher deutlich risikobehaftet.

Gültige Werte für flags sind (Binär-Oder-Kombinationen aus):
eICImportMapStructure (Die Kennfeldstruktur wird übertragen)
eICImportMapContents (Die Zellwerte in Kennfeldern werden übertragen)
eICImportDataAreas (Die Zellwerte außerhalb von Kennfeldern werden übertragen)
eICOnlyChangedMaps (Nur Kennfelder mit Änderungen werden beachtet)
eICCompareById (Die Offsetermittlung erfolgt anhand des Ids. Die Kennfelder müssen im Zielprojekt bereits existieren)
eICCompareByName (Dito. Beachten Sie, dass die Namen eindeutig sein müssen)
eICAllowReturn4 (Erlaubt Rückgabewert von 4 (wg. Kompatibilität))
eICRemoveDuplicates (Löscht ggf. bereits bestehende Duplikate im Zielprojekt)

Transfermode kann genau einen dieser Werte annehmen:
eICTMAbsolute (Geänderte Zellwerte werden absolut übertragen)
eICTMRelative (Geänderte Zellwerte werden relativ (delta) übertragen)
eICTMPercent (Geänderte Zellwerte werden als prozent übertragen)
eICTMAllFromOrg (Alle Original-Zellwerte werden absolut übertragen)
eICTMAllFromVer (Alle Zellwerte werden absolut übertragen)

Rückgabewert:
0=Das Quellprojekt konnte nicht gefunden/verwendet werden
1=Nicht alle Kennfelder / Bereiche konnten übernommen werden
2=Nicht alle Kennfeldachsen konnten übernommen werden
3=Alles konnte übernommen werden
4=Alles konnte übernommen werden und es gab nur 1 Offset. (Erfordert eICAllowReturn4)

Beispiel: projectImportChanges ("x:\\myfolder\\source.ols", 1);

2.4.45 projectAutoUpdate

Syntax: projectAutoUpdate (bool bForce=FALSE)
Rückgabewert: bool
Erfordert WinOLS5.21 + FeatureUpdate

Triggert das AutoUpdate (Entspricht: Projekt > Ex-&Import > AutoUpdate) für das Projekt.

bForce: Wenn TRUE, dann werden die Daten aktualisiert, auch dann das Dateidatum das eigentlich nicht erfordert.

Beispiel: projectAutoUpdate ();

2.4.46 projectAutoImport

Syntax: projectAutoImport (bool bForce=FALSE)
Rückgabewert: bool
Erfordert WinOLS5.21 + FeatureUpdate

Triggert den AutoImport (Entspricht: Projekt > Ex-&Import > AutoImport) für das Projekt.

bForce: Wenn TRUE, dann werden die Daten aktualisiert, auch dann das Dateidatum das eigentlich nicht erfordert.

Beispiel: projectAutoImport ();

2.4.47 projectSearchVehicleData

Syntax: projectSearchVehicleData ()
Rückgabewert: bool

Startet die Suche nach Projekteigenschaften.

Beispiel: projectSearchVehicleData ();

2.4.48 projectCloneVehicleData

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

Rückgabewert: bool

Verwendet die Funktion „Projekte aktualisieren > Klonen“ (und per default dessen Einstellungen) um Projekteigenschaften aus Projekten mit identischen Suchfeldern in das aktuelle Projekt zu übernehmen. Alle 3 Parameter sind optional. Werden sie nicht angegeben, dann verwendet WinOLS die aktuellen Einstellungen aus dem Dialog. Wird der zweite Parameter angegeben, dann muss er ALLE flags beinhalten (binaryor), die verwendet werden sollen. Im dritten Parameter werden die einzelnen Werte durch Semikolons getrennt.

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

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

2.4.49 projectGetQuickFixState

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

Gibt den Status des gewünschten QuickFixes zurück. Der Name muss nicht exakt passen, es reichen auch Teile.

Der Rückgabewert ist wahlweise eine Zahl oder ein String. Eine 1 oder 0 signalisiert ein an oder aus. (WinOLS erkennt die meisten üblichen Zustandsnamen automatisch.) Wichtig: Das ist eine Zahl, ein Vergleich mit den Boolean „true“ oder „false“ wird nicht funktionieren! Im String-Modus wird der Name des Zustands zurückgegen. Der Rückgabewert -1 (oder "undef") bedeutet, dass der aktuelle Zustand widersprüchlich ist. -2 (oder "unknown") bedeutet, dass kein QuickFix mit dem Namen gefunden wurde.

Wird diese Funktion ohne offenes Projekt aufgerufen, dann gibt sie false (Boolean) zurück.
Erfordert WinOLS 5.72

Beispiel: stateNumber = projectGetQuickFixState (“FeatureX”);

2.4.50 projectSetQuickFixState

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

Setzt den Zustand des QuickFixes. Der Name muss nicht exakt passen, es reichen auch Teile. Übergibt man den Zustand als String dann muss er exakt passen (weil sonst „applied“ auch auf „not applied“ passen würde). Übergibt man den Zustand als Boolen (zB true), dann versteht WinOLS automatisch die meisten im Projekt üblichen textuellen Bezeichnungen.

Eine Rückgabewert von 1 signalisiert Erfolg. Der Rückgabewert -1 bedeutet, dass der gewünschte Zustand nicht gefunden wurde. -2 bedeutet, dass kein QuickFix mit dem Namen gefunden wurde.

Wird diese Funktion ohne offenes Projekt aufgerufen, dann gibt sie false (Boolean) zurück.
Erfordert WinOLS 5.72

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

2.4.51 projectGetQuickFixes

Syntax: projectGetQuickFixes ()
Rückgabewert: Array

Gibt eine Liste der Namen aller im aktuellen Projekt verfügbaren QuickFixes zurück.

Wird diese Funktion ohne offenes Projekt aufgerufen, dann gibt sie false (Boolean) zurück.
Erfordert WinOLS 5.72

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

2.5 Kontext: Version

2.5.1 versionGetProperty

Syntax: versionGetProperty (Number id)
Rückgabewert: String oder Number

Diese Funktion fragt eine der Texte aus den Versionseigenschaften ab.

Der Parameter „id“ muss einen der folgender Werte haben: eVerPropName, eVerPropComment, eVerPropCreatedOn, eVerPropChangedOn, eVerPropChecksum, eVerPropCVN, eVerPropOutput, eVerPropTorque, eVerPropState, eVerPropCredits

Bei eVerPropState sind folgende Rückgabewerte möglich: eVerStatNone, eVerStatExperiment, eVerStatToDo, eVerStatDev, eVerStatTestable, eVerStatErrors, eVerStatFinished, eVerStatMaster, eVerAutoExport, eVerAutoImport, eVerAutoUpdate, eVerAutoUpdateAndExport

Beispiel: MessageBox (versionGetProperty (eVerPropName));

2.5.2 versionSetProperty

Syntax: versionSetProperty (Number id, String newvalue)
Syntax: versionSetProperty (Number id, Number newvalue)
Rückgabewert: bool

Diese Funktion ändert einen der Texte aus den Versionseigenschaften auf einen neuen Wert.

Der Parameter „id“ muss einen der Werte wie bei versionGetProperty haben. Hinweis: Diese Funktion ändert die Projektdaten und markiert das Projekt als geändert.

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

2.6 Kontext: Fenster

Der Kontext des aktuellen Fensters impliziert auch immer das aktuelle Projekt, da dies immer mit dem

Fenster verbunden ist. Als Fenster gelten hier nur die Projektfenster (Kennfeld, Hexdump, ...), nicht aber die Spezialfenster wie Suche, Übersicht, etc.

2.6.1 windowGetActive

Syntax: windowGetActive ()
Rückgabewert: Number

Diese Funktion gibt einen Identifier (Typ: Number) für das aktuelle Fenster zurück. Der Identifier ist nur für diese Session gültig. Wird das Fenster oder WinOLS einmal geschlossen, ist der Identifier ungültig.

Beispiel: aw = windowGetActive ();

2.6.2 windowSetActive

Syntax: windowSetActive(Number WindowIdentifier)
Rückgabewert: bool

Diese Funktion aktiviert das Fenster, was durch den Parameter identifiziert wird und bringt es in den Vordergrund. Alle Befehle die ein Projekt oder Fenster als Kontext erwarten, verwenden ab jetzt dieses Fenster / Projekt.

Beispiel: 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)
Rückgabewert: string

Gibt den aktuellen Wert für die gewählte Kennfeld-Eigenschaft zurück. Die möglichen PropertyNamen entsprechen denen des WinOLS-Skript-Befehls set_map_property. (Siehe WinOLS-Hilfedatei)

Wenn als zweiter Parameter eine Adresse oder ein Kennfeld-Id übergeben wird, dann bezieht sich die Funktion nicht auf das aktive Kennfeld-Fenster, sondern auf das Kennfeld was an der angegebenen Adresse startet / den angegebenen Kennfeld-Id besitzt. Wenn es mehrere Kennfelder gibt, die dieser Bedingung entsprechen, kann man mit nSkip angeben, wie viele davon übersprungen werden sollen.

Diese Funktion erfordert, dass das Projektrecht „Kennfeldstruktur übertragen“ gegeben ist.

Beispiel: 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)
Rückgabewert: bool

Ändert eine Eigenschaft des aktuellen Kennfeldes (falls bLastNew==TRUE: des zuletzt per LUA hinzugefügten Kennfelds). Wenn als dritter Parameter eine Adresse (>1) oder ein Kennfeld-Id übergeben wird, dann bezieht sich die Funktion nicht auf das aktive Kennfeld-Fenster, sondern auf das Kennfeld was an der angegebenen Adresse startet / den angegebenen Kennfeld-Id besitzt. Wenn es mehrere Kennfelder gibt, die dieser Bedingung entsprechen, kann man mit nSkip angeben, wie viele davon übersprungen werden sollen.

Die möglichen PropertyNamen entsprechen denen des WinOLS-Skript-Befehls set_map_property. (Siehe WinOLS-Hilfedatei)

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

3 Index