

Scripting Tracker

Development Tool for SAP GUI Scripting

Version 5.18

Scripting Tracker is a utility to support the development of SAP GUI Scripting. The UI of the program is designed to offer a better overview by splitting up the work space into tabs. The Analyzer tab shows a well arranged tree with all SAP sessions and their scripting objects. Also it shows for each scripting object, after the selection in the tree with a single mouse click, a lot of technical details like ID, position etc. The Recorder tab shows a basic editor to load, edit and execute SAP GUI scripts. You can select and use the session you want, to run your script with this session.

The Analyzer offers the possibility to identify each scripting object of the SAP GUI with a red frame. There are two ways to achieve this: The first is to select an object from the hierarchy and to press right mouse button. The second is to select an object from the hierarchy and to press the identify button. Next it is necessary to move the mouse pointer to the selected session window. After the identifying of the scripting object it is possible to copy its technical name, called ID, to the clipboard and to use them in another context. This functionality is equal to the SAP GUI Scripting wizard.

With the Recorder the program offers the possibility to record, edit and execute your SAP GUI activities in PowerShell Windows and PowerShell Core, C# (.NET Framework and .NET 5 or 6), VB.NET, Python, JShell for Java, AutoIt and Visual Basic Script. Also you can use the code sequences inside RPA platforms.

Scripting Tracker supports the SAP GUI for Windows and the NetWeaver Business Client (NWBC) for Desktop.

Current Version from 22.07.2022

First Version from 20.10.2012

Benefit of Scripting Tracker

Under normal circumstances you can do with the SAP GUI Scripting recorder the standard to record and replay your manual SAP GUI activities. But sometimes it is not enough. You need an extra editor to customize your scripts. Also, if you record your script, you have no visual contact to the generated code. It is a blind flying to record your activities. Scripting Trackers recorder has the same functionality as SAP GUI Scripting recorder to record and replay SAP GUI Scripts.

Scripting Tracker brings more transparency. With the recorder of Scripting Tracker you have full visual control about the generated scripting code. You see in the integrated basic editor each line which is generated from recorder. And you have the possibility to enrich the code automatically with additional information. Scripting Tracker adds comment lines about the transaction, title, dynpro - program name and screen number - and the session number into your source code.

And Scripting Tracker supports different scripting engines. With Scripting Tracker it is possible to use *PowerShell Windows*, *PowerShell Core*, *C# (.NET Framework and .NET 5)*, *VB.NET*, *VBScript* of Windows Script Host (WSH), *AutoIt*, *Python* and *Java Shell* (JShell). You can record and replay sources of this engines. These different platforms offers now a wide base for total new integration scenarios. With Scripting Tracker it is now easy possible to integrate SAP GUI activities.

Microsoft stops with Windows 7 the delivery of the agents, also known as wizards. But the SAP GUI Scripting tools needs it. Therefore the SAP stops, with the SAP GUI for Windows release 7.20 patch level 9, also the support of the SAP GUI Scripting tools - you can find more information at OSS note 1633639. The Analyzer of Scripting Tracker is an alternative. Scripting Trackers Analyzer has the same functionality as SAP GUI Scripting wizard to identify the scripting objects. It shows all scripting objects in a clearly well arranged tree and, after a selection of one object, a lot of technical details or its position on the GUI with a red frame of the scripting object.

Scripting Tracker supports different SAP UI strategies. Primary it supports SAP GUI Scripting with SAP GUI for Windows, but also with NetWeaver Business Client (NWBC) for Desktop. And it works with Windows 7 and higher.

On the one hand Scripting Tracker optimizes your development process with SAP GUI Scripting. And on the other hand Scripting Tracker offers new horizons of integration between an SAP system and your presentation server. After all, Scripting Tracker brings you a step forward in independence and it increases your efficiency with SAP GUI Scripting.

Supported Languages of Scripting Tracker

Scripting Tracker supports the languages

- PowerShell Windows
- PowerShell Core
- C# (.NET Framework and .NET 5)
- VB.NET
- VBScript of the Windows Script Host (WSH)
- Autolt
- Python
- JShell (Java Shell)

To use Scripting Tracker it is necessary to enable and allow the execution of PowerShell or VBScript on the presentation server, or you can install and use PowerShell Core, Autolt or Python scripting engine as well as JShell. To use C# with .NET 5 or .NET 6 it is necessary to install the appropriate SDK.

Hint: PowerShell Windows is in a normal case available on any Windows system, but it is necessary to [set the execution policy](#).

Enable SAP GUI Scripting

Scripting Tracker uses SAP GUI for Windows with the SAP GUI Scripting API. So it is necessary to enable SAP GUI Scripting on the application and presentation server.

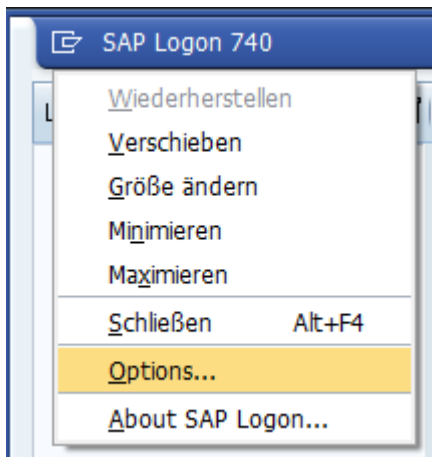
Hint: If the SAP GUI Scripting is disabled on one application server, you don't see its sessions in the tree.

[On Presentation Server](#)

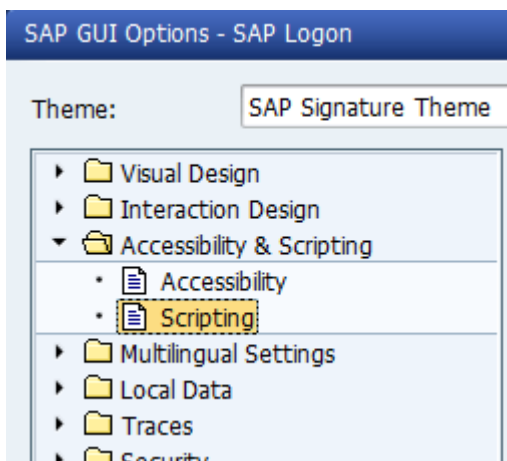
[On Application Server](#)

Enable SAP® GUI Scripting on Presentation Server

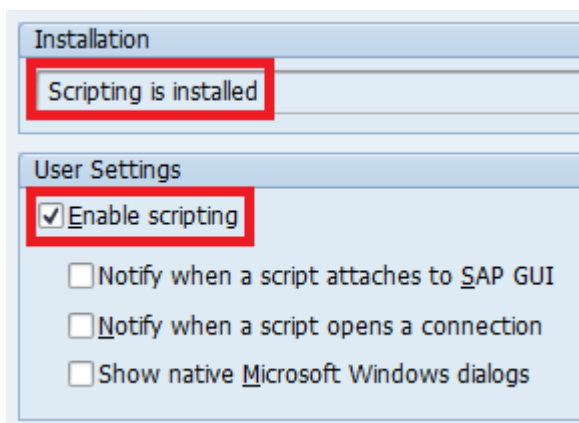
- Choose the menu item Options... from the system menu of the SAP® Logon.



- Choose the node Scripting.



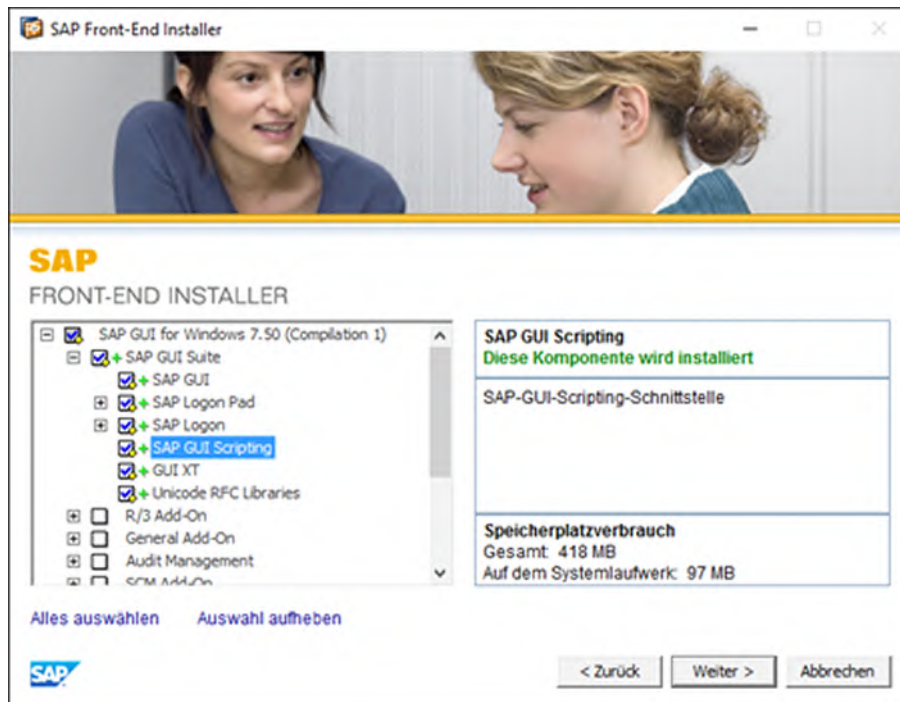
- The Scripting must be installed and activated.



Hint: It is necessary to disable the notifications, otherwise you got a requester for each script execution.

Hint: It is necessary to disable the using of native Windows dialogs. On this way the native Windows dialogs, e.g. like Save as or Open, are replaced with a dynpro-based dialog. So you have the possibility to record your activities also with these dialogs. An example [how it works in the SAP back-end is here](#) available.

Hint: The SAP GUI Scripting is an optional component from the SAP GUI Suite, so it is possible to install the SAP GUI Suite without SAP GUI Scripting and therefore it is necessary to check it.



Registry Entries of the SAP® GUI Scripting on the Presentation Server

You can find [more information about SAP GUI family at the Wiki](#).

Enable Scripting

HKEY_CURRENT_USER\Software\SAP\SAPGUI Front\SAP Frontend
Server\Security\UserScripting
from type REG_DWORD, Default: 1, 0 = inactive, 1 = active

Notify when a script attaches to SAP GUI

HKEY_CURRENT_USER\Software\SAP\SAPGUI Front\SAP Frontend
Server\Security\WarnOnAttach
from type REG_DWORD, Default: 1, 0 = inactive, 1 = active

Notify when a script opens a connection

HKEY_CURRENT_USER\Software\SAP\SAPGUI Front\SAP Frontend
Server\Security\WarnOnConnection
from type REG_DWORD, Default: 1, 0 = inactive, 1 = active

Show native MS Windows dialogs

HKEY_CURRENT_USER\Software\SAP\SAPGUI Front\SAP Frontend
Server\Scripting\ShowNativeWinDlgs
from type REG_DWORD, Default: 0, 0 = inactive, 1 = active

Registry Entries - File Save Dialog

In the ABAP code of the method FILE_SAVE_DIALOG from the class CL_GUI_FRONTEND_SERVICES is an example available how the registry entry ShowNativeWinDlgs is requested. At first a method is called which detects if scripting is active or not. If it is active it calls the function module GUI_FILE_SAVE_DIALOG, instead of FileSaveDialog of the SAPInfo Control module, which calls the native Windows dialog. This works independently from the SAP GUI for Windows version and from its settings.

```
call method IS_SCRIPTING_ACTIVE receiving result = rt_value EXCEPTIONS others
= 1.

if rt_value = 1.

* check the registry key
call method cl_gui_frontend_services=>registry_get_dword_value
  exporting root = cl_gui_frontend_services=>HKEY_CURRENT_USER
            key = 'Software\SAP\SAPGUI Front\SAP Frontend Server\Scripting'
            value = 'ShowNativeWinDlgs'
  importing reg_value = RCCU
  exceptions
    others = 1.

call method cl_gui_frontend_services=>registry_get_dword_value
  exporting root = cl_gui_frontend_services=>HKEY_LOCAL_MACHINE
            key = 'Software\SAP\SAPGUI Front\SAP Frontend Server\Scripting'
            value = 'ShowNativeWinDlgs'
  importing reg_value = RCLM
  exceptions
    others = 1.

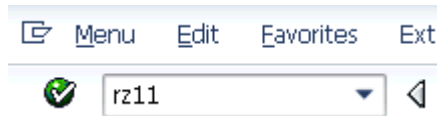
call method cl_gui_cfw=>flush.

if RCCU = 0 or ( RCCU ne 1 and RCLM ne 1 ).
  RT_VALUE = 'X'.
  call FUNCTION 'GUI_FILE_SAVE_DIALOG'
    exporting
      WINDOW_TITLE      = WINDOW_TITLE
      DEFAULT_EXTENSION = DEFAULT_EXTENSION
      DEFAULT_FILE_NAME = DEFAULT_FILE_NAME
      WITH_ENCODING      = WITH_ENCODING
      INITIAL_DIRECTORY = INITIAL_DIRECTORY
      FILE_FILTER        = FILTER
    importing
      FULLPATH      = FULLPATH
      FILE_ENCODING = FILE_ENCODING
      USER_ACTION   = USER_ACTION.
endif.

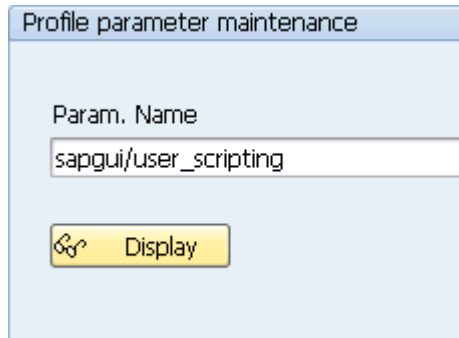
endif.
```


Enable SAP® GUI Scripting on Application Server





- Use the transaction code RZ11 in the ok field.



- Use the profile parameter sapgui/user_scripting and press the Display button.



- The current value must be TRUE.

Param. Name	sapgui/user_scripting	
Short description(Engl)	Enable or disable user scripting on the frontend.	
Appl. area	GUI Frontend	
ParameterTyp	Logical value	
Changes allowed	Change permitted	
Valid for oper. system	All operating systems	
DynamicallySwitchable	<input checked="" type="checkbox"/>	
Same on all servers	<input type="checkbox"/>	
Dflt value	FALSE	
ProfileVal	TRUE	
Current value	TRUE	

If it is FALSE, press the Change Value button and change it to TRUE on all servers.

Change Parameter Value

Parameter values

Param. Name: sapgui/user_scripting

Default value: FALSE

ProfileVal: TRUE

Current value: TRUE

New value: TRUE

☒ Switch on all servers

Important hint: It is necessary to use only uppercase characters.

- Or to view all profile parameters use the report RSPARAM with transaction code SA38 or SE38.

se38

Program: RSPARAM

Display Profile Parameter

Parameter Name	User-Defined Value	System Default Value	System Default Value(Unsubstituted Form)	Comment
sapgui/user_scripting	TRUE	FALSE	FALSE	Enable or disable user scripting on the frontend.
sapgui/user_scripting_disable_recording		FALSE	FALSE	Disable the recording capabilities of SAP GUI Scripting
sapgui/user_scripting_force_notification		FALSE	FALSE	Prevent users from disabling the SAP GUI Scripting notifications.
sapgui/user_scripting_per_user		FALSE	FALSE	Check user privileges to determine if user scripting should be enabled.
sapgui/user_scripting_set_readonly		FALSE	FALSE	Enable or disable a read-only version of SAP GUI Scripting.

- To set the profile parameter permanent, change to the directory SAP\<SID>\SYS\profile and append to the file <SID>_<INSTANCE>_<HOST> e.g. NSP_DVEBMGS00_ABAP the line sapgui/user_scripting = TRUE.
- You can control the SAP GUI Scripting via the following profile parameters:
 sapgui/user_scripting = Enable or disable user scripting on the frontend (TRUE)
 sapgui/user_scripting_disable_recording = Disable the recording capabilities of SAP GUI Scripting (FALSE)
 sapgui/user_scripting_force_notification = Prevent users from disabling the SAP GUI Scripting notifications (FALSE)
 sapgui/user_scripting_per_user = Check user privileges to determine if user scripting should be enabled (FALSE)
 sapgui/user_scripting_set_readonly = Enable or disable a read-only version of SAP GUI Scripting (FALSE)

Personal Settings on the Application Server

In the menu Help > Settings of the SAP GUI for Windows there is the possibility to customize personal settings. To ensure the automation of the F1 and F4 help, settings must be made at this point.

F1 Help


If it is necessary to automate the F1 help, switch from the Performance Assistant to the Modal Dialog Box.

F1 Help Application help F4 Help

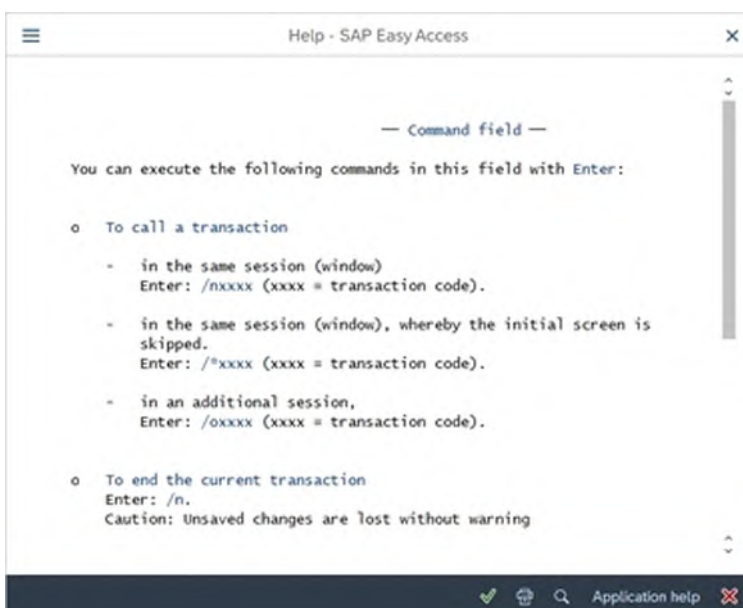
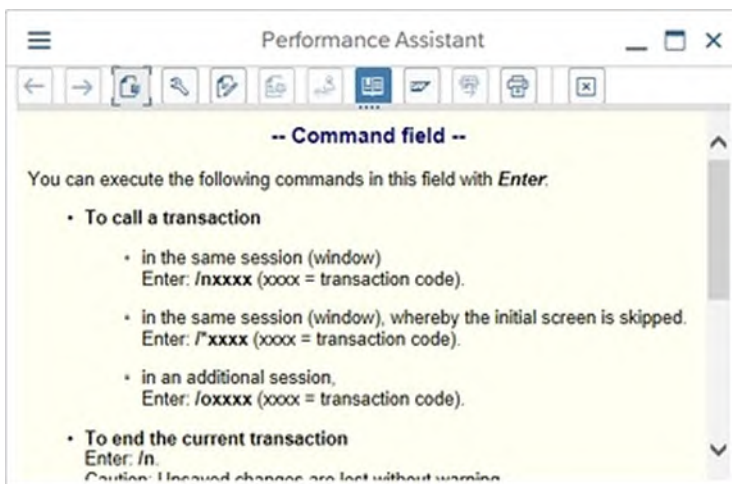
Display

☐ in Performance Assistant

☒ in Modal Dialog Box



The difference between the Performance Assistant and the Modal Dialog Box is, that the Performance Assistant is an ActiveX control which is not supported by SAP GUI Scripting.



Hint: Only in exceptional cases should it be necessary to automate the F1 help.

F4 Help

To use the F4 help, in the context of an automation, it is necessary to select the Dialog (modal) option in the Display settings.

F1 Help Application help **F4 Help**

User-specific settings

☐ Do not display pers. value list automatic.

☐ Only return value directly if only one hit

Max. number of hits to be displayed

Maximum Width Hit List (in chars.)

Display
☐ Control (amodal)
☒ **Dialog (modal)** ←
☐ System defaults

System defaults


☐ Pers. value list

☐ Return value dir.

500
----- changeable ----

255


Display
☒ Control
☐ Dialog



The difference between the Control (amodal) and the Dialog (modal) is, that the Control (amodal) is an ActiveX control which is not supported by SAP GUI Scripting.

Airline Code (1) 18 Entries found

Restrictions




ID	Airline
AA	American Airlines
AB	Air Berlin
AC	Air Canada
AF	Air France
AZ	Alitalia
BA	British Airways
CO	Continental Airlines
DL	Delta Airlines
FJ	Air Pacific
JL	Japan Airlines
LH	Lufthansa
NG	Lauda Air
NW	Northwest Airlines

18 Entries found

Airline Code 18 Entries

ID	Airline
AA	American Airlines
AB	Air Berlin
AC	Air Canada
AF	Air France
AZ	Alitalia
BA	British Airways
CO	Continental Airlines
DL	Delta Airlines
FJ	Air Pacific
JL	Japan Airlines
LH	Lufthansa
NG	Lauda Air
NW	Northwest Airlines
QF	Qantas Airways
SA	South African Air.
SQ	Singapore Airlines
SR	Swiss
UA	United Airlines



Hint: To test the different dialogs use TAC SE38 with the report DEMO_SELECTION_SCREEN_F4.

Program

[Menu](#)

[Toolbar](#)

[Analyzer](#)

[Recorder](#)

[Scripting API](#)

[Composer](#)

[Comparator](#)

[DumpState](#)

[Customizing](#)

[Notes](#)

[Statusbar](#)

[Keyboard Shortcuts](#)





[Preference file](#)

[Snippets file](#)




Menu

Menu	Description
File Exit	Quits Tracker
Tools Scan	Scans scripting objects of all sessions
Tools Always on top	Tracker window always on top
Tools Running Object Table (ROT)...	Opens a dialog which shows the display names of the running instances which are registered in the running object table (ROT).
Tools Export SAP GUI Scripting API...	Saves the SAP GUI Scripting API as CSV file.
Help Help...	Opens this help file
Help VBScript help...	Optional menu item. If the file VBScript.chm is in the same directory as Tracker it will be shown. It opens this VBScript® help file.
Help SAP GUI Scripting help...	Optional menu item. If the file SAPGUIScripting.chm is in the same directory as Tracker it will be shown. It opens this SAP® GUI Scripting help file.
Help Autolt help...	Optional menu item. If the keyword AutoltHelp in the ScriptingEngines section of the preference file is set to the Autolt help file in CHM format, this will be open.
Help About...	Shows an additional window with information about Tracker

Toolbar

Item	Description
 Scan scripting objects of all sessions	Actualize the content of the tree. The progress bar under the toolbar shows the scan activities.
 Tracker window always on top	This is a toggle button. It makes the program window sticky on the desktop.
 About Tracker...	Shows an additional window with information about Tracker.
 Opens help...	Opens this help file.

Analyzer

Item	Description
 Identify scripting object from SAP® GUI in Tracker hierarchy	This is a toggle button. Put the session to be analysed in foreground, select any object of this session in Tracker hierarchy and switch the button on. Move the mouse pointer to the session and if it is over an scripting object, the object will be marked with a red frame. Also the scripting object and its technical details will be shown in Tracker.
 Find scripting object in Tracker hierarchy	Opens a dialog to input a text to find a scripting object in Tracker hierarchy.
 Find next scripting object in Tracker hierarchy	Continues the search to find a scripting object in Tracker hierarchy.












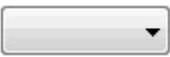



Right Mouse Click in the Analyzer Tree












A right mouse click on a session item opens a popup menu.


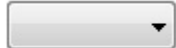
Menu	Description
Window in foreground	Brings the selected session window in foreground.
Get information	Shows a lot of technical information about the selected session in a message box.
Export IDs to clipboard	Exports all IDs or only the IDs of the user screen of the selected session to the clipboard.
Export IDs to file	Exports all IDs or only the IDs of the user screen of the selected session to a file.

A right mouse click on a scripting object visualize this object with a red frame in the SAP® GUI. This means it shows a red frame around the scripting object in the SAP® GUI of the selected item in Tracker hierarchy tree.

Recorder

Item	Description
 Clear editor	Clears the editor. If source was changed, the file save dialog will be opened.
 Open file...	Opens a dialog to choose a file to load it in the editor.
 Save file...	Opens a dialog to save the source code as file.
 Save file with standard code...	Opens a dialog to save the source code as file, with additional standard code for SAP GUI Scripting in the head and foot.
 Cut to clipboard	Cuts the selected text from the editor to the clipboard.
 Copy to clipboard	Copies the selected text to the clipboard.
 Paste from clipboard	Pastes text from the clipboard to the actual position of the text cursor.
 Undo	Undo the last activity.
 Redo	Redo the last activity.
 Open source in external editor	Opens the source code with an external editor. If you press the shift button, you add a few lines of code and information. Configure the editors in the section ProgramConfiguration of the Tracker.ini file.
 Reload source from external editor	Reloads the source code from an external editor.
 Code snippet	Inserts a code snippet from Snippets.xml into the editor at the actual cursor position. Look here for further information.
 Playback script	Executes the script from the editor.
 Record SAP® GUI Script	Records SAP® activities to a script in the editor.
	Stops the executing of the scripting process.

Stop script process	
 or  Antimalware scan	Scans the script code via Antimalware Scan Interface (AMSI) before it is stored. Configure the risk level in the section ScriptingEngines of the Tracker.ini file. Use the keyword AMSIAcceptedRiskLevel. Green check = AMSI activated Red cross = AMSI deactivated Hint: Activate with the Group Policy Editor (gpedit.msc) the setting Computer Configuration > Administrative Templates > Windows Components > Windows PowerShell > PowerShell Script Block Logging.
 Use PowerShell® Core Script	Records and executes the script as PowerShell® Core script file. Configure the path and file name of the PowerShell® engine in the section ScriptingEngines of the Tracker.ini file. Use the keyword PowerShellCore.
 Use PowerShell® Windows Script	Records and executes the script as PowerShell® Windows script file. Configure the path and file name of the PowerShell® engine in the section ScriptingEngines of the Tracker.ini file. Use the keyword PowerShell.
 Use C#	Records the script as C# code for .NET 5.0 or 6.0.
 Use C#	Records the script as C# code for the .NET Framework.
 Use VB.NET	Records the script as VB.NET code for the .NET Framework.
 Use Python	Records and executes the script as Python source. Configure the path and file name of the Python engine in the section ScriptingEngines of the Tracker.ini file. Use the keyword Python.
 Use JShell	Records and executes the script as JShell source. Configure the path and file name of the JShell engine in the section ScriptingEngines of the Tracker.ini file. Use the keyword JShell.
 Use AutoIt Script	Records and executes the script as AutoIt script file. Configure the path and file name of the AutoIt engine in the section ScriptingEngines of the Tracker.ini file. Use the keyword AutoIt.
 Use VBA or VBS via Windows® Script Host	Records the script as Visual Basic for Applications (VBA) or records and executes VBScript® (VBS) file via Windows® Script Host (WSH).

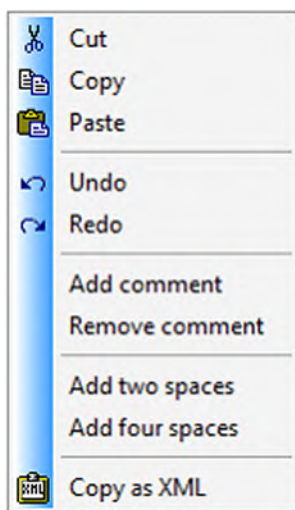
 Additional information in source	Enriches the source with information comment lines about the transaction, title, dynpro - program name and screen number - and the session number.
 SAP® session	Chooses the SAP® session to execute or record the script in it. If a session is selected, the window is set into foreground and some code is added automatically, to identify the connection and session. If no session is chosen the script will be executed as normal VBScript®.






Recorder Editor


- With the key combination Alt + Shift + Arrows it is possible to select a block.

Right Mouse Click in the Editor

A right mouse click in the editor opens a popup menu.



Item	Description
 Cut to clipboard	Cuts the selected text from the editor to the clipboard.
 Copy to clipboard	Copies the selected text to the clipboard.
 Paste from clipboard	Pastes text from the clipboard to the actual position of the text cursor.
 Undo	Undo the last activity.
 Redo	Redo the last activity.

Add comment	Add a comment sign at the begin of the selected lines.
Remove comment	Remove the comment sign from the begin of the selected lines.
Add two spaces	Add two spaces at the begin of the selected lines.
Add four spaces	Add four spaces at the begin of the selected lines.
 Copy as XML	<p>Copies the selected text to the clipboard and converts it to XML</p> <pre> & to &amp; < to &lt; > to &gt; " to &quot; ' to &apos; </pre>

Keyboard Shortcuts









Shortcut	Description
Ctrl + G	Inserts Get-Property code for PowerShell.
Ctrl + I	Inserts Invoke-Method code for PowerShell.
Ctrl + S	Inserts Set-Property code for PowerShell.

Scripting API

The Scripting API shows in a tree all classes, with its methods and properties, and the enumerations of the SAP® GUI Scripting API. Also it shows the arguments and the types of the methods and properties, also the constants of the enumerations. With a double click on a node the text is copied into the clipboard. With a single right click you open the SAP® GUI Scripting API help. It is necessary to set the sapfewse variable in the preference file, here it must set the path to sapfewse.ocx file, e.g. like `C:\Program Files (x86)\SAP\FrontEnd\SAPgui`. In the section below you see the Scripting API sorted by methods and properties, to see in which classes are they available. With a double click on one of the classes it will open the class in the tree above.

Composer

With the composer is it possible to arrange all snippets on an easy way. Choose the type of UI and the programming language. Now you can choose the snippet you like which is inserting at the actual caret position.

Item	Description
 Clear editor	Clears the editor.
 Open file...	Opens a dialog to choose a file to load it in the editor.
 Save file...	Opens a dialog to save the source code as file.
 Cut to clipboard	Cuts the selected text from the editor to the clipboard.
 Copy to clipboard	Copies the selected text to the clipboard.
 Paste from clipboard	Pastes text from the clipboard to the actual position of the text cursor.
 Undo	Undo the last activity.
 Redo	Redo the last activity.
C# to PowerShell	Converts selected C# WebDriver code to PowerShell convention
UTF8 / ASCII	Encoding of the file, default UTF8.

SAPGUI

A set of snippets to handle SAP® GUI for Windows UI automation via SAP® GUI Scripting API. These are the same snippets as in the recorder.

Web

A set of snippets to handle web UI automation via Selenium.


- [Selenium](#)
- [Chrome Browser \(Offline Installer\)](#)
- [Chrome WebDriver](#) or from [Storage](#)
- [Firefox Browser](#)
- [Mozilla Gecko WebDriver](#)
- [Edge WebDriver](#)
- [Katalon Automation Recorder](#) or from [Chrome Web Store](#)

Mobile

A set of snippets to handle mobile UI automation for Android devices via Appium.

- [Appium](#)
- [Appium Client Library](#) (Selenium Webdriver extension for Appium)
- [Selenium WebDriver](#) (Supporting browser automation)
- [Selenium WebDriver Support](#) (Supporting Selenium WebDriver)
- [Newtonsoft JSON](#) (JSON framework for .NET)
- [Castle Core](#) (DynamicProxy, Logging Abstractions and DictionaryAdapter)

Comparator

Item	Description
 Compare screen elements	Compares the selected screens to find different screen elements. This functionality compares the ID, the type and the changeable attribute. If a file is selected, only the IDs are compared.

DumpState

Item	Description
Dump Dumps the state of an object	<p>Delivers a hierarchy of collections with information about the state of an object.</p> <p>The parameter InnerObject may be used to specify for which internal object the data should be dumped. The most complex components support this parameter. In the most cases it is an empty string.</p> <p>The following OpCodes are used:</p> <ul style="list-style-type: none">• GPR = Get Property and Return value• MR = Method and Return value• GP = Get Property• M = Method

Customizing

The button "Edit Preference File" opens the Note tab and the Tracker.ini file.

The button "Edit Snippet File" opens the Note tab and the Snippet.xml file.

Program

- Path for temporary files
With the customizing is it possible to change the path of the temporary files on the runtime of Scripting Tacker on restricted areas.
- Delete temporary files
- Execute script without session
Here you can decide if you want to executes the scripts without a choosen session.
- Add SAP[®] standard code in source
If this checkbox is enabled, Tracker enriches the external source file with standard code.

PowerShell

- Minimized window style for PowerShell session
Sets the window style of PowerShell to minimized.
- PowerShell session does not exit after running
Does not close the PowerShell session after executing.

C#

- Use lambda expressions instead of methods
To use late binding a set of methods is used, with this option a set of lambda expressions is used instead. The generated source code of the main routine can then be copied directly into the UiPath Invoke Code activity. Uncomment the lines and delete the following line and add a variable from the type Microsoft.VisualBasic.

Python

- Python session does not exit after running
Does not close the Python session after executing.

JShell

- JShell session does not exit after running
Does not close the JShell session after executing.

WScript

- Use CScript instead of WScript
Use console application instead of windows application.
- Shows the registry information about the customizing of enabling Windows Script Host in SysWOW64 node for x86 application, value of Enabled in path
`SOFTWARE\Microsoft\Windows Script Host\Settings.`
-









SAP GUI Scripting User Settings

Shows a few registry information about the customization of SAP GUI Scripting in SysWOW64 node for x86 application. You can find more information [here](#).

- Enable Scripting
- Notify when a script attaches to SAP GUI
- Notify when a script opens a connection
- Show native Microsoft Windows dialogs

Notes

Notes is nothing more than a tiny editor where you can store different text informations.

Item	Description
 Clear notes	Clears the note.
 Open file...	Opens a dialog to choose a file to load it in the note.
 Save file...	Opens a dialog to save the note as file.
 Cut to clipboard	Cuts the selected text from the note to the clipboard.
 Copy to clipboard	Copies the selected text to the clipboard.
 Paste from clipboard	Pastes text from the clipboard to the actual position of the text cursor.
 Undo	Undo the last activity.
 Redo	Redo the last activity.
UTF8 / ASCII	Encoding of the file, default UTF8.

Statusbar

The statusbar on the bottom of the UI is segmented in four areas:

1. Status of the program - Ready or Active.
2. Version of the SAP GUI Scripting.
3. SAPGUI if an instance exists.
4. SAPGUI SERVER if one or more instances exists, and in brackets the number of instances.
5. Message from Anti Malware Scan Interface (AMSI)

Keyboard Shortcuts

Shortcut	Description
Alt + S	Scans the SAP® GUI Scripting objects of all sessions and refresh the content of the tree.
Alt + R	Shrinks the window to the title bar and vis-à-vis.
Alt + Q	Disable the identify scripting object button
Alt + F4	Quits Tracker.
F1	Opens this help file.

Preference file

It is possible to configure Scripting Tracker via the preference file Tracker.ini. The preference file must be in the same directory as Tracker.exe.

The preference file has two sections. With the first `ProgramConfiguration` it is possible to configure Tracker and with the second `ScriptingEngines` it is possible to set the path to the different scripting engines.

- `ProgramConfiguration`

Keyword	Description
EditorFont	Name of the using font in the editor, default Consolas.
EditorFontSize	Size of the using font in the editor, default 10.
EditorExternalWSH	Path and name of the VisualBasic® editor, default notepad.exe.
EditorExternalAU3	Path and name of the AutoIt editor, default notepad.exe.
EditorExternalPS1	Path and name of the PowerShell® Windows editor, default C:\Windows\System32\WindowsPowerShell\v1.0\powershell_ise.exe.
EditorExternalCorePS1	Path and name of the PowerShell® Core editor, default notepad.exe.
EditorExternalCS	Path and name of the C# editor, default notepad.exe.
EditorExternalVB	Path and name of the VB.NET editor, default notepad.exe.
EditorExternalPY	Path and name of the Python editor, default notepad.exe.
EditorExternalJSH	Path and name of the JShell editor, default notepad.exe
WindowPosSave	0 or 1 to save the window position, default 0
WindowPosX	X position of the window, default 10
WindowPosY	Y position of the window, default 10
WindowPosWidth	Width of the window, default 800
WindowPosHeight	Height of the window, default 800
sapfewse	Path to SAP® frontend Windows® scripting

	engine (sapfewse.ocx)
CodePage	Number of the codepage for the VBS ANSI files, default 1252
NotesFont	Name of the using font in the notes, default Calibri.
NotesFontSize	Size of the using font in the notes, default 12.

- **ScriptingEngines**

Keyword	Description
Autolt	Path and name of the Autolt engine.
AutoltHelp	Path and name of the Autolt help.
PowerShell	Path and name of the PowerShell® Windows engine.
PowerShellCore	Path and name of the PowerShell® Core engine.
Python	Path and name of the Python engine.
JShell	Path and name of the JShell engine.
VBScriptHelp	Path and name of the VBScript help.
AMSIAcceptedRiskLevel	The antimalware provider returns a result between 0 and 32767, inclusive an estimated risk level. With this keyword it is possible to set the sensitivity of the risk level. If the risk level is exceeded, a request dialog appears.

- **Tools**

In the [Tools] section you have the possibility to implement tools you like in the toolbar of Scripting Tracker. The keyword is shown as tooltip and the value is the program name you want to start.

Hint: You can use for the engines, exception PowerShell, and for the external editor paths the %userprofile% environment variable. This contains the profile directory of the user. Typical path is C:\Users\Username.

You can use for JShell path the %java_home% environment variable.

You can use %programfiles%, %programfiles(x86)%, %windir% and %systemroot% environment variable.

Example:

```
[ProgramConfiguration]
EditorFont = Consolas
EditorFontSize = 11
EditorExternalPS1 = %WINDIR%\sysnative\WindowsPowerShell\v1.0\powershell_ise.exe
EditorExternalCorePS1 = %SYSTEMROOT%\SysWOW64\notepad.exe
EditorExternalCS = %SYSTEMROOT%\SysWOW64\notepad.exe
EditorExternalVB = %SYSTEMROOT%\SysWOW64\notepad.exe
EditorExternalWSH = %SYSTEMROOT%\SysWOW64\notepad.exe
EditorExternalAU3 = %SYSTEMROOT%\SysWOW64\notepad.exe
EditorExternalPY = %SYSTEMROOT%\SysWOW64\notepad.exe
EditorExternalJSH = %SYSTEMROOT%\SysWOW64\notepad.exe
WindowPosSave = 0
WindowPosX = 5
WindowPosY = 10
WindowPosWidth = 640
WindowPosHeight = 760
sapfewse = %PROGRAMFILES(X86)%\SAP\FrontEnd\SAPgui
CodePage = 1252
NotesFont = Calibri
NotesFontSize = 12
[ScriptingEngines]
PowerShell = %WINDIR%\sysnative\WindowsPowerShell\v1.0\powershell.exe
PowerShellCore = %USERPROFILE%\PowerShellCore\pwsh.exe
AutoIt = %PROGRAMFILES(X86)%\AutoIt3\AutoIt3.exe
AutoItHelp = %PROGRAMFILES(X86)%\AutoIt3\AutoIt.chm
Python = %USERPROFILE%\Python\python.exe
JShell = %JAVA_HOME%\bin\jshell.exe
VBScriptHelp = C:\Language\VBScript\VBScript.chm
[Tools]
AutoItRecorder = C:\Language\AutoIt\Au3Recorder.exe
```

Snippets file

It is possible to define code snippets via the XML file Snippets.xml. The snippets file must be in the same directory as Tracker.exe.

With the title tag you define the text which is shown in the combobox. With the language tag you define the programming language in whose context the snippet is shown, allowed is here PowerShell, VBNet, CSharp, WScript, AutoIt, Python and Java. In the ui tag you can use any type you like, Scripting Tracker uses SAPGUI, Web and All. With the code tag you define the code which is copied into the editor at the actual cursor position.

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<!DOCTYPE snippets [
<!ELEMENT snippets (snippet)+>
<!ELEMENT snippet (title, language, ui, code)>
<!ELEMENT title (#PCDATA)>
<!ELEMENT language (#PCDATA)>
<!ELEMENT ui (#PCDATA)>
<!ELEMENT code (#PCDATA)>
]>

<snippets>

  <snippet>
    <title>Begin</title>
    <language>WScript</language>
    <ui>SAPGUI</ui>
    <code>
'-Begin-----
</code>
  </snippet>

</snippets>
```

Hint: Don't use comments in the XML file.

Robotic Process Automation

Robotic Process Automation (RPA) platforms, like UiPath or Blue Prism, uses the SAP GUI Scripting API to automate SAP GUI for Windows. The combination of RPA with SAP GUI Scripting offers interesting automation possibilities. The code, generated by Scripting Tracker, can be easily transferred to RPA. This speeds up the development of SAP GUI for Windows code sequences and reduces the possibilities of errors.

Integration scenarios of Scripting Tracker in the development workflow of UiPath on the example of different programming and scripting languages.

[Integration Scenario - UiPath - C#](#)

[Integration Scenario - UiPath - PowerShell Windows](#)

[Integration Scenario - UiPath - VBScript](#)

[Integration Scenario - UiPath - AutoIt](#)

Integration scenarios of Scripting Tracker in the development workflow of Blue Prism with different programming languages.

[Integration Scenario - Blue Prism - C#](#)

[Integration Scenario - Blue Prism - VB.NET](#)

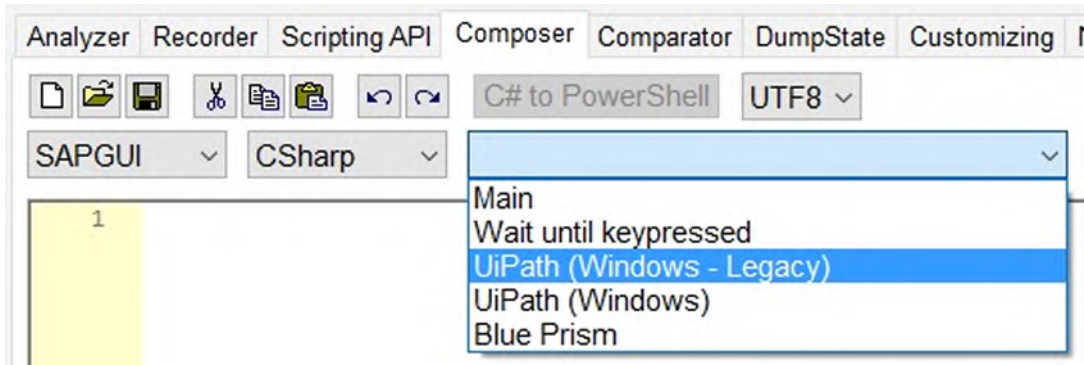
Identifiers play an essential role at automation in the SAP GUI for Windows environment. The highest hierarchical levels are the connection and the session. RPA platforms detects these themselves to ensure high flexibility. Here approaches to use this information in integration scenarios.

[Connection and Session Number - UiPath](#)

[Session ID and Session Number - UiPath](#)

Integration Scenario - UiPath - C#

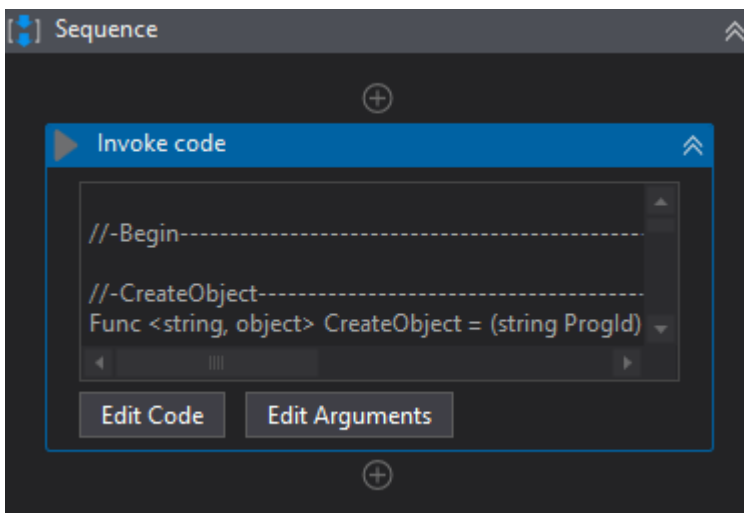
To use C# code with the UiPath Invoke Code activity is very easy. At first you must record your SAP activities with C#. Switch to the Composer tab and insert the SAPGUI > CSharp > UiPath (Windows - Legacy) or UiPath (Windows) code sequence.



Move your recorded SAP activities into the code sequence and replace with it the line
//>Insert your SAP GUI Scripting code here<.

```
56 dynamic ID = null;
57
58 //>Insert your SAP GUI Scripting code here<
59
60 FreeObject(session);
```

Copy now this whole code from the Composer into your UiPath Invoke Code activity.



Hint: If your UiPath project is Windows - Legacy it is necessary to add a dummy variable from type Microsoft.VisualBasic.* to your workflow sequence.

Name	Variable type	Scope	Default
Dummy1	Microsoft.VisualBasic.AppWinStyle	Sequence	Enter a VB expression
Create Variable			

Integration Scenario - UiPath - PowerShell Windows

PowerShellScript

Hint: Don't forget to [set the execution policy](#) first.

```
#-Begin-----

#-Parameters-----
Param(
    [String]$ConnectionNumber = "0",
    [String]$SessionNumber = "0"
)

#-Includes-----
."$PSScriptRoot\COM.ps1";

#-Main-----
$SapGuiAuto = Get-Object( , "SAPGUI");
If ($SapGuiAuto -isnot [__ComObject]) {
    Exit;
}

$application = Invoke-Method $SapGuiAuto "GetScriptingEngine";
If ($application -isnot [__ComObject]) {
    Free-Object $SapGuiAuto;
    Exit;
}

$connection = Get-Property $application "Children"
@([convert]::ToInt32($ConnectionNumber, 10));
If ($Null -eq $connection) {
    Free-Object $SapGuiAuto;
    Exit;
}

$session = Get-Property $connection "Children"
@([convert]::ToInt32($SessionNumber, 10));
If ($Null -eq $session) {
    Free-Object $SapGuiAuto;
    Exit;
}

$ID = Invoke-Method $session "findById" @("wnd[0]/tbar[0]/okcd");
Set-Property $ID "text" @("/nSE16");
$ID = Invoke-Method $session "findById" @("wnd[0]");
Invoke-Method $ID "sendVKey" @(0);
$ID = Invoke-Method $session "findById"
@("wnd[0]/usr/ctxtDATABROWSE-TABLENAME");
Set-Property $ID "text" @("TADIR");
$ID = Invoke-Method $session "findById"
@("wnd[0]/usr/ctxtDATABROWSE-TABLENAME");
Set-Property $ID "caretPosition" @(5);
$ID = Invoke-Method $session "findById" @("wnd[0]");
Invoke-Method $ID "sendVKey" @(0);
$ID = Invoke-Method $session "findById" @("wnd[0]");
Invoke-Method $ID "sendVKey" @(31);

$ID = Invoke-Method $session "findById" @("wnd[1]/usr/txtG_DBCOUNT");
$dbCount = Get-Property $ID "text";
```

```

$ID = Invoke-Method $session "findById" @("wnd[1]/tbar[0]/btn[0]");
Invoke-Method $ID "press";
$ID = Invoke-Method $session "findById" @("wnd[0]");
Invoke-Method $ID "sendVKey" @(3);
$ID = Invoke-Method $session "findById" @("wnd[0]");
Invoke-Method $ID "sendVKey" @(3);

Free-Object $SapGuiAuto;

Set-Content dbCount.txt $dbCount;

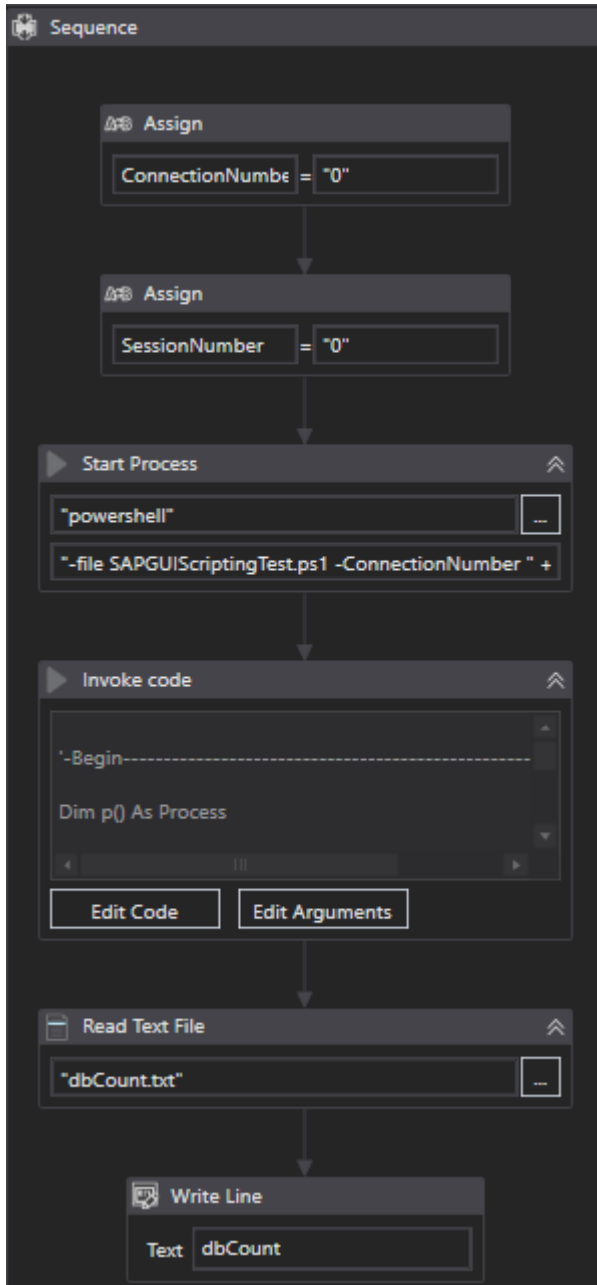
#-End-----

```

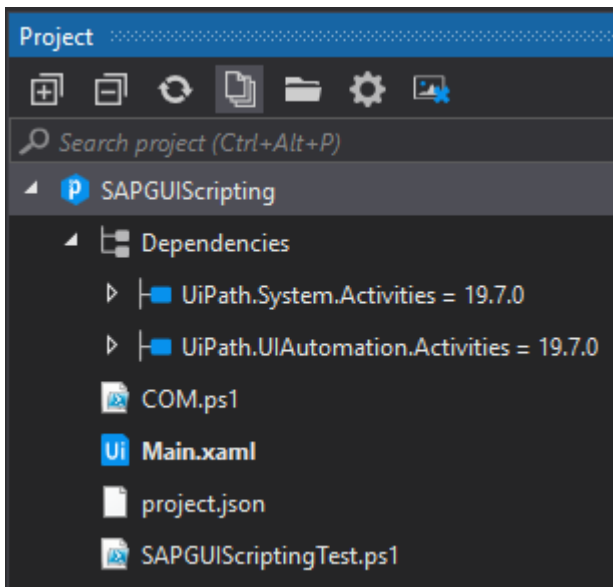
Variables in UiPath

Name	Variable type	Scope	Default
dbCount	String	Sequence	Enter a VB expression
ConnectionNumber	String	Sequence	Enter a VB expression
SessionNumber	String	Sequence	Enter a VB expression
Create Variable			

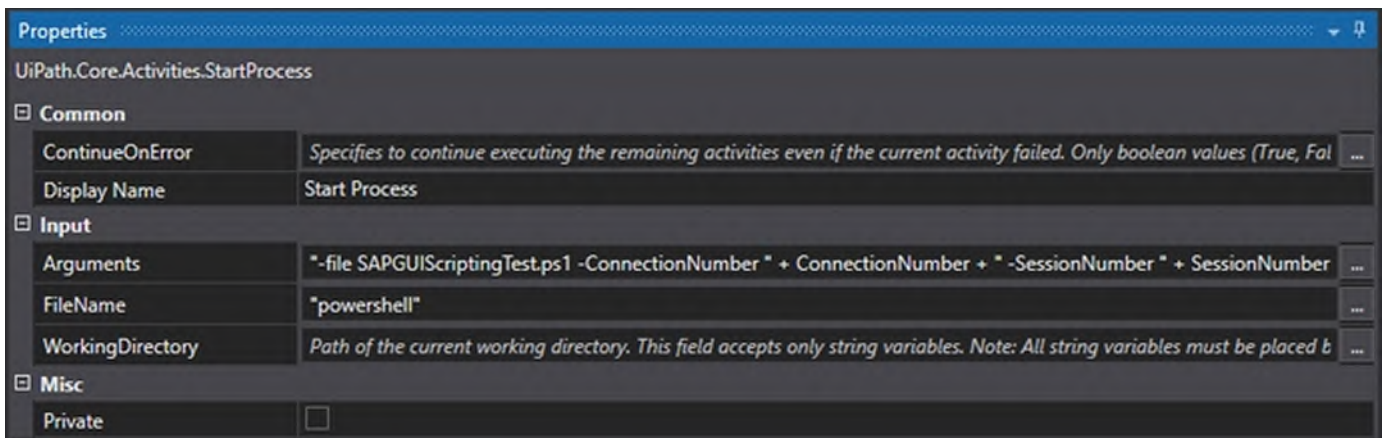
Sequence in UiPath



Hint: Store the PowerShell script file and the include into your project folder.



Properties of Start-Process activity



Code for Invoke Code activity

```
'-Begin-----  
  
Dim p() As Process  
  
Do  
    p = System.Diagnostics.Process.GetProcessesByName("powershell")  
    System.Threading.Thread.Sleep(500)  
Loop Until p.Length = 0  
  
'-End-----
```

Hint: To get the result from the PowerShell script the content of the file dbCount.txt is read.

Integration Scenario - UiPath - VBScript

VBScript

```
'-Begin-----

Option Explicit

Dim ConnectionNumber, SessionNumber
Dim SapGuiAuto, application, connection, session, dbCount

ConnectionNumber = WScript.Arguments.Item(0)
SessionNumber = WScript.Arguments.Item(1)

If Not IsObject(application) Then
    Set SapGuiAuto = GetObject("SAPGUI")
    Set application = SapGuiAuto.GetScriptingEngine
End If

If Not IsObject(connection) Then
    Set connection = application.Children(CInt(ConnectionNumber))
End If

If Not IsObject(session) Then
    Set session = connection.Children(CInt(SessionNumber))
End If

session.findById("wnd[0]/tbar[0]/okcd").text = "/nSE16"
session.findById("wnd[0]").sendVKey 0
session.findById("wnd[0]/usr/ctxtDATABROWSE-TABLENAME").text = "TADIR"
session.findById("wnd[0]/usr/ctxtDATABROWSE-TABLENAME").caretPosition = 5
session.findById("wnd[0]").sendVKey 0
session.findById("wnd[0]").sendVKey 31

dbCount = session.findById("wnd[1]/usr/txtG_DBCOUNT").Text

session.findById("wnd[1]/tbar[0]/btn[0]").press
session.findById("wnd[0]").sendVKey 3
session.findById("wnd[0]").sendVKey 3

WScript.Echo CStr(dbCount)

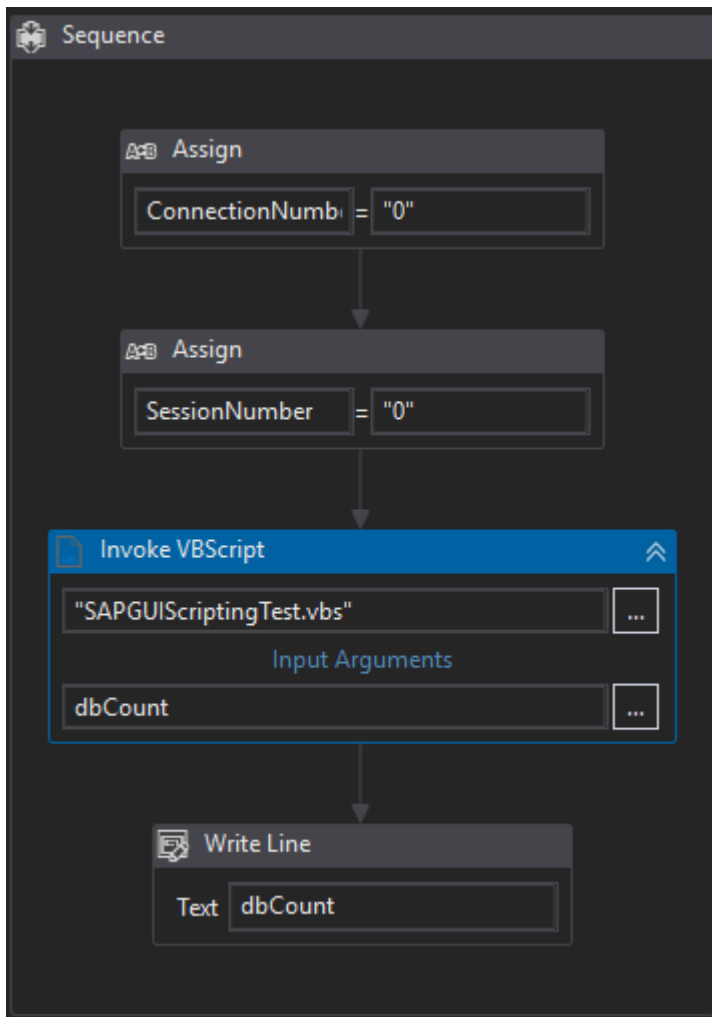
'-End-----
```

Hint: If you use Option Explicit you must define the arguments of Invoke VBScript activity too.

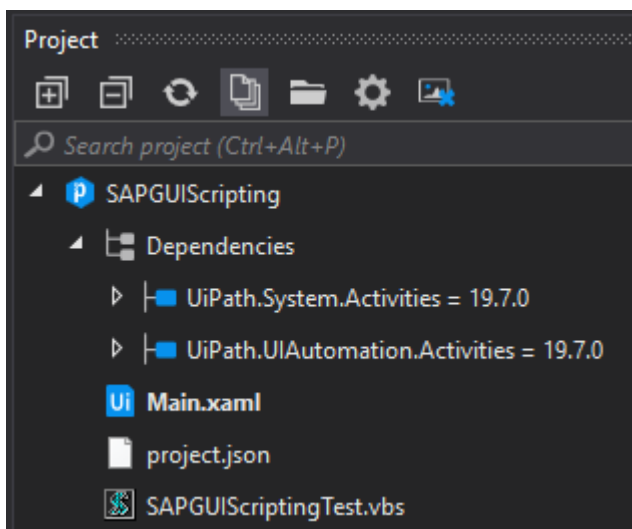
Variables in UiPath

Name	Variable type	Scope	Default
dbCount	String	Sequence	Enter a VB expression
ConnectionNumber	String	Sequence	Enter a VB expression
SessionNumber	String	Sequence	Enter a VB expression
Create Variable			

Sequence in UiPath



Hint: Store the VBScript file into your project folder.



Properties of Invoke VBScript activity

The screenshot shows the 'Properties' window for the 'Invoke VBScript' activity. The window is titled 'Properties' and has a search bar. The activity name 'UiPath.Core.Activities.InvokeVBScript' is displayed at the top. The properties are organized into sections: Common, Input, Misc, Options, and Output.

Section	Property	Value
Common	DisplayName	Invoke VBScript
	Timeout	Specifies the amount of time (in milliseconds) for the invoked ...
Input	Arguments	(Collection) ...
	VBScriptFileName	"SAPGUIScriptingTest.vbs" ...
Misc	Private	<input type="checkbox"/>
Options	HidePopups	Default: False. Determines if the display alerts, scripting errors ...
	KillOnTimeout	Default: False. Determines if the VBScript process will be killed ...
	UnicodeSupport	Default: False. Allows usage of special characters in input and ...
	WaitForOutput	True ...
Output	Result	dbCount ...

Arguments of Invoke VBScript activity

The screenshot shows the 'Arguments' window for the 'Invoke VBScript' activity. The window has a title bar with standard window controls. It contains a table with three columns: Direction, Type, and Value. There are two arguments listed: 'ConnectionNumber' and 'SessionNumber', both with 'In' direction and 'String' type. Below the table is a 'Create Argument' button. At the bottom right are 'OK' and 'Cancel' buttons.

Direction	Type	Value
In	String	ConnectionNumber
In	String	SessionNumber

Create Argument

OK Cancel

Integration Scenario - UiPath - Autolt

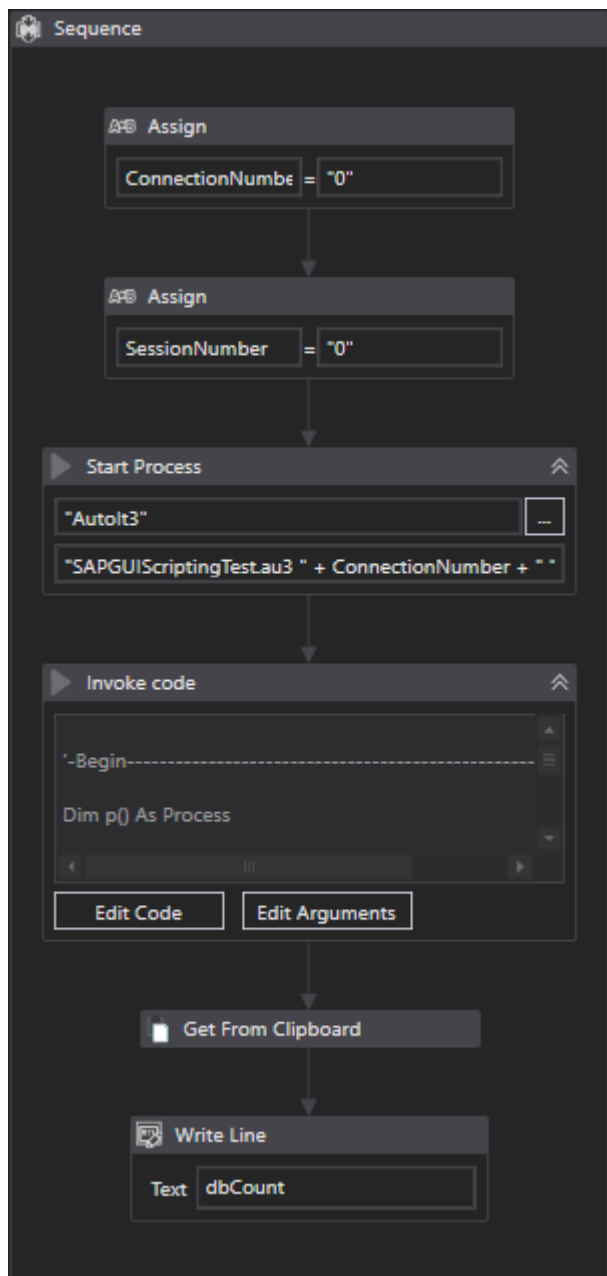
AutoltScript

```
;--Begin-----  
  
AutoItSetOption("MustDeclareVars", 1)  
  
Dim $ConnectionNumber, $SessionNumber  
Dim $SapGuiAuto, $Application, $connection, $session, $dbCount  
  
$ConnectionNumber = Number($CmdLine[1])  
$SessionNumber = Number($CmdLine[2])  
  
$SapGuiAuto = ObjGet("SAPGUI")  
If Not IsObj($SapGuiAuto) Or @Error Then  
    Exit  
EndIf  
  
$Application = $SapGuiAuto.GetScriptingEngine()  
If Not IsObj($Application) Then  
    Exit  
EndIf  
  
$connection = $Application.Children($ConnectionNumber)  
If Not IsObj($connection) Then  
    Exit  
EndIf  
  
$session = $connection.Children($SessionNumber)  
If Not IsObj($session) Then  
    Exit  
EndIf  
  
$session.findById("wnd[0]/tbar[0]/okcd").text = "/nSE16"  
$session.findById("wnd[0]").sendVKey(0)  
$session.findById("wnd[0]/usr/ctxtDATABROWSE-TABLENAME").text = "TADIR"  
$session.findById("wnd[0]/usr/ctxtDATABROWSE-TABLENAME").caretPosition = 5  
$session.findById("wnd[0]").sendVKey(0)  
$session.findById("wnd[0]").sendVKey(31)  
  
$dbCount = $session.findById("wnd[1]/usr/txtG_DBCOUNT").text  
  
$session.findById("wnd[1]/tbar[0]/btn[0]").press  
$session.findById("wnd[0]").sendVKey(3)  
$session.findById("wnd[0]").sendVKey(3)  
  
Clipput($dbCount)  
  
;--End-----
```

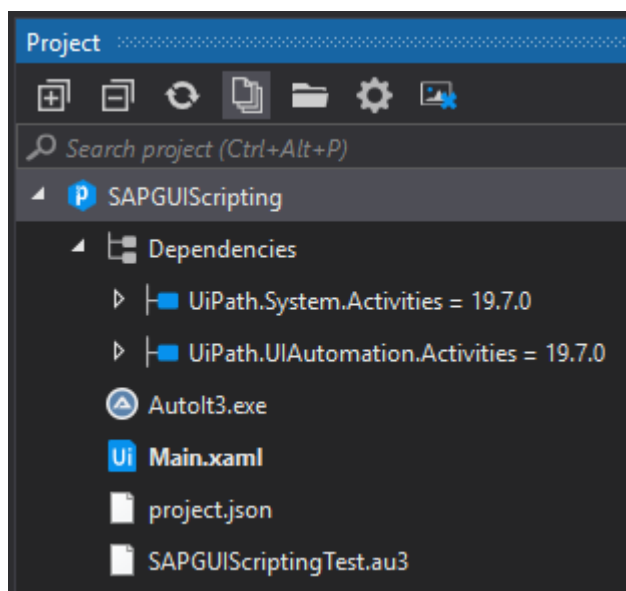
Variables in UiPath

Name	Variable type	Scope	Default
dbCount	String	Sequence	Enter a VB expression
ConnectionNumber	String	Sequence	Enter a VB expression
SessionNumber	String	Sequence	Enter a VB expression
Create Variable			

Sequence in UiPath



Hint: Store the Autolt script file and the Autolt3.exe into your project folder.



Properties of Start Process activity

The screenshot shows the 'Properties' window for the 'UiPath.Core.Activities.StartProcess' activity. It is divided into three sections: 'Common', 'Input', and 'Misc'. The 'Common' section has 'ContinueOnError' set to 'Specifies to continue executing the remaining activities even if the current activity failed' and 'Display Name' set to 'Start Process'. The 'Input' section has 'Arguments' set to '"SAPGUIScriptingTest.au3 " + ConnectionNumber + " " + SessionNumber', 'FileName' set to '"Autolt3"', and 'WorkingDirectory' set to 'Path of the current working directory. This field accepts only string variables. Note: All st'. The 'Misc' section has 'Private' set to an unchecked checkbox.

UiPath.Core.Activities.StartProcess	
Common	
ContinueOnError	Specifies to continue executing the remaining activities even if the current activity failed ...
Display Name	Start Process
Input	
Arguments	"SAPGUIScriptingTest.au3 " + ConnectionNumber + " " + SessionNumber ...
FileName	"Autolt3" ...
WorkingDirectory	Path of the current working directory. This field accepts only string variables. Note: All st ...
Misc	
Private	<input type="checkbox"/>

Hint: For the Inter Process Communication (IPC) with the Autolt interpreter the clipboard is using. To synchronize the Start Process activity an Invoke Code activity is used.

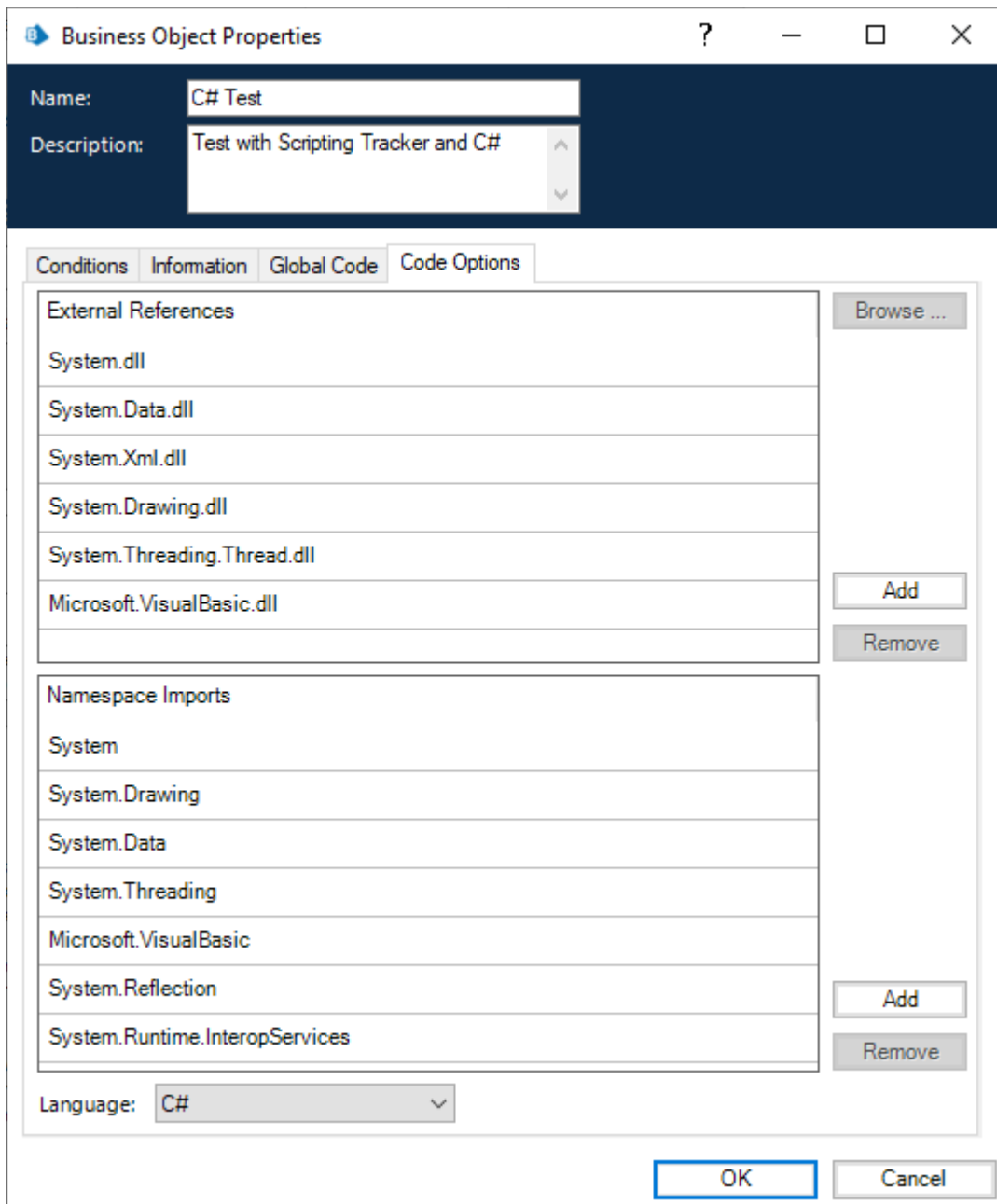
Code for Invoke Code activity

```
'-Begin-----  
  
Dim p() As Process  
  
Do  
    p = System.Diagnostics.Process.GetProcessesByName("AutoIt3")  
    System.Threading.Thread.Sleep(500)  
Loop Until p.Length = 0  
  
'-End-----
```

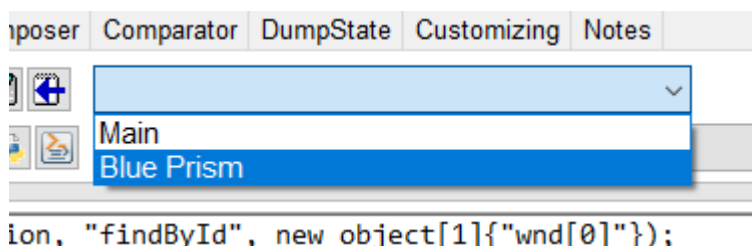
Hint: To get the result from the Autolt script the content of the clipboard is read.

Integration Scenario - Blue Prism - C#

Before you can use the C# code, which was recorded with Scripting Tracker, you must add an external reference. It is necessary to add the library Microsoft.VisualBasic.dll, because the GetObject method is used. Also the namespaces Microsoft.VisualBasic, System.Reflection and System.Runtime.InteropServices must be imported.



Record your activities with C#, add the code snippet Blue Prism, ...



... move your recorded code to the correct position and copy and paste your C# code from Scripting Tracker to your code stage.

Code Properties

Name: Code1

Description:

Inputs Outputs Code

```

1 //Begin-----
2
3
4 dynamic InvokeMethod(object obj, string methodName, object[] methodParams = null) {
5     return obj.GetType().InvokeMember(methodName, BindingFlags.InvokeMethod, null, obj, methodParams);
6 }
7
8 dynamic GetProperty(object obj, string propertyName, object[] propertyParams = null) {
9     return obj.GetType().InvokeMember(propertyName, BindingFlags.GetProperty, null, obj, propertyParams);
10 }
11
12 dynamic SetProperty(object obj, string propertyName, object[] propertyParams = null) {
13     return obj.GetType().InvokeMember(propertyName, BindingFlags.SetProperty, null, obj, propertyParams);
14 }
15
16 object session = null;
17 try {
18     object SapGuiAuto = Interaction.GetObject("SAPGUI");
19     object app = InvokeMethod(SapGuiAuto, "GetScriptingEngine");
20     object connection = GetProperty(app, "Children", new object[]{});
21     session = GetProperty(connection, "Children", new object[]{});
22 } catch {
23     return;
24 }
25
26 dynamic ID = null;
27
28 ID = InvokeMethod(session, "findById", new object[]{"wnd[0]"});
29 InvokeMethod(ID, "resizeWorkingPane", new object[]{{84, 29, 0}});
30 ID = InvokeMethod(session, "findById", new object[]{"wnd[0]/tbar[0]/okcd"});
31 SetProperty(ID, "text", new object[]{"nselid"});
32 ID = InvokeMethod(session, "findById", new object[]{"wnd[0]"});
33 InvokeMethod(ID, "sendVKey", new object[]{});

```

Check Code

Parameters


Stage logging: Disabled

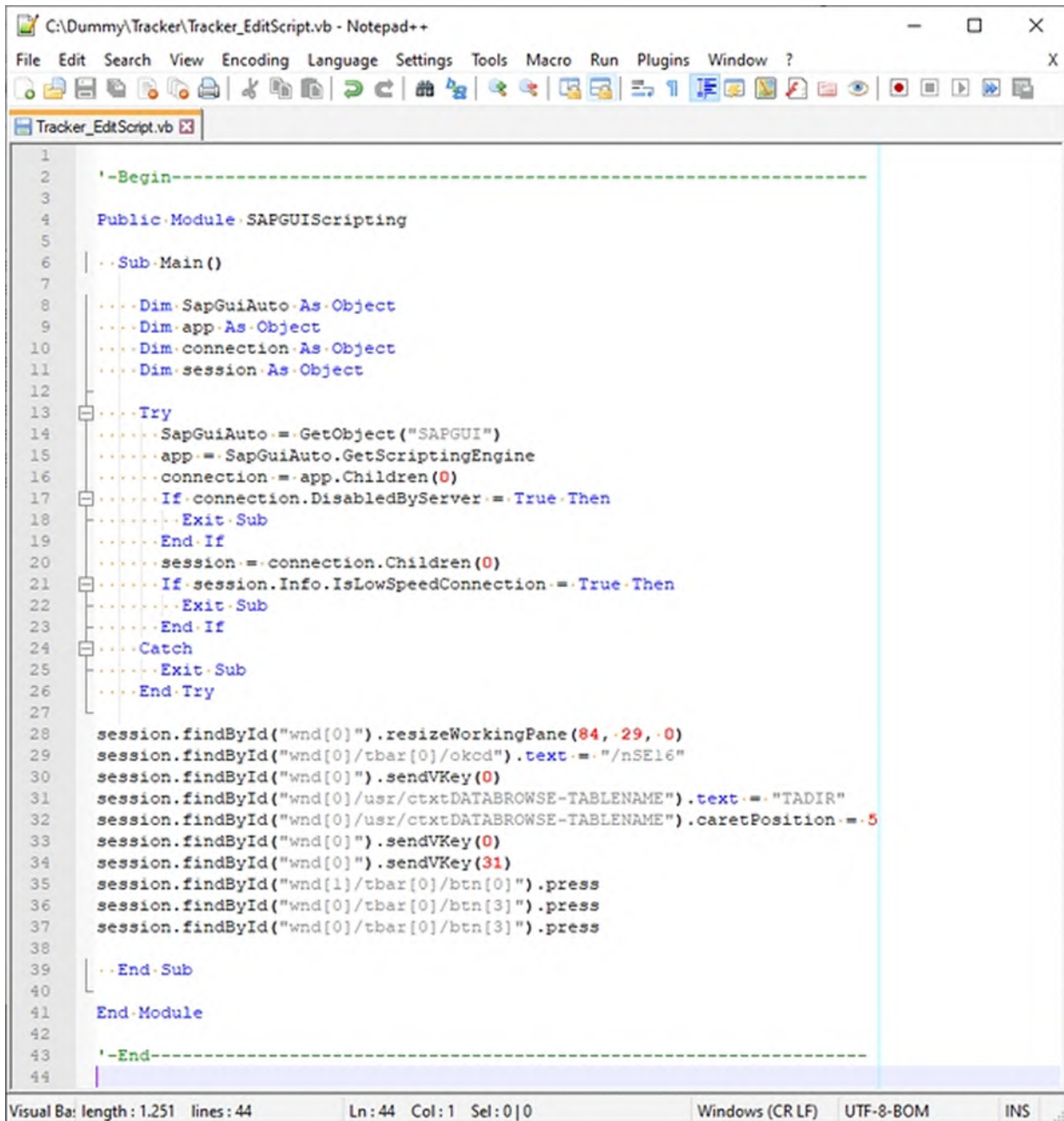
Warning threshold: System Default

Number of minutes: 5 (0 to disable)

OK Cancel

Integration Scenario - Blue Prism - VB.NET

To work with VB.NET is the most comfortable way to execute SAP GUI Scripting with Blue Prism. Record your activities with VB.NET, press the button open source in external editor  and ...



```
1  '-Begin-----
2
3
4  Public Module SAPGUIScripting
5
6  ..Sub Main()
7
8      ....Dim SapGuiAuto As Object
9      ....Dim app As Object
10     ....Dim connection As Object
11     ....Dim session As Object
12
13     ....Try
14         ....SapGuiAuto = GetObject("SAPGUI")
15         ....app = SapGuiAuto.GetScriptingEngine
16         ....connection = app.Children(0)
17         ....If connection.DisabledByServer = True Then
18             ....Exit Sub
19         ....End If
20         ....session = connection.Children(0)
21         ....If session.Info.IsLowSpeedConnection = True Then
22             ....Exit Sub
23         ....End If
24         ....Catch
25             ....Exit Sub
26         ....End Try
27
28     session.findById("wnd[0]").resizeWorkingPane(84, 29, 0)
29     session.findById("wnd[0]/tbar[0]/okcd").text = "/nSE16"
30     session.findById("wnd[0]").sendVKey(0)
31     session.findById("wnd[0]/usr/ctxtDATABROWSE-TABLENAME").text = "TADIR"
32     session.findById("wnd[0]/usr/ctxtDATABROWSE-TABLENAME").caretPosition = 5
33     session.findById("wnd[0]").sendVKey(0)
34     session.findById("wnd[0]").sendVKey(31)
35     session.findById("wnd[1]/tbar[0]/btn[0]").press
36     session.findById("wnd[0]/tbar[0]/btn[3]").press
37     session.findById("wnd[0]/tbar[0]/btn[3]").press
38
39 ..End Sub
40
41 End Module
42
43 '-End-----
44
```

Visual Ba: length: 1.251 lines: 44 Ln: 44 Col: 1 Sel: 0|0 Windows (CR LF) UTF-8-BOM INS

... copy the code sequence between Sub Main() and End Sub into your code stage.

Code Properties

Name: Code1

Description:

Inputs

Outputs

Code

Parameters

```

1 Dim SapGuiAuto As Object
2 Dim app As Object
3 Dim connection As Object
4 Dim session As Object
5
6 Try
7     SapGuiAuto = GetObject("SAPGUI")
8     app = SapGuiAuto.GetScriptingEngine
9     connection = app.Children(0)
10    If connection.DisabledByServer = True Then
11        Exit Sub
12    End If
13    session = connection.Children(0)
14    If session.Info.IsLowSpeedConnection = True Then
15        Exit Sub
16    End If
17 Catch
18    Exit Sub
19 End Try
20
21 session.findById("wnd[0]").resizeWorkingPane(34, 29, 0)
22 session.findById("wnd[0]/tbar[0]/okcd").text = "/nSE16"
23 session.findById("wnd[0]").sendVKey(0)
24 session.findById("wnd[0]/usr/ctxtDATABROWSE-TABLENAME").text = "TADIR"
25 session.findById("wnd[0]/usr/ctxtDATABROWSE-TABLENAME").caretPosition = 5
26 session.findById("wnd[0]").sendVKey(0)
27 session.findById("wnd[0]").sendVKey(31)
28 session.findById("wnd[0]/tbar[0]/btn[0]").press
29 session.findById("wnd[0]/tbar[0]/btn[3]").press
30 session.findById("wnd[0]/tbar[0]/btn[3]").press

```

Check Code

Stage logging: Disabled

Warning threshold: System Default

Number of minutes 5 (0 to disable)

OK

Cancel

Connection and Session Number - UiPath

In the area of SAP GUI for Windows automation plays the identifier (ID) of the UI elements an important role. It looks e.g. like this:

```
/app/con[0]/ses[0]/wnd[0]/usr/cntlIMAGE_CONTAINER/shellcont/shell
```

The ID starts always with /app, the GuiApplication. Then follows /con[0], which is the connection with its number, the GuiConnection. Then follows /ses[0], which is the session with its number, the GuiSession. You can find [more information here about the SAP GUI Scripting Object Model](#). When you log on to an SAP system, you establish a connection.

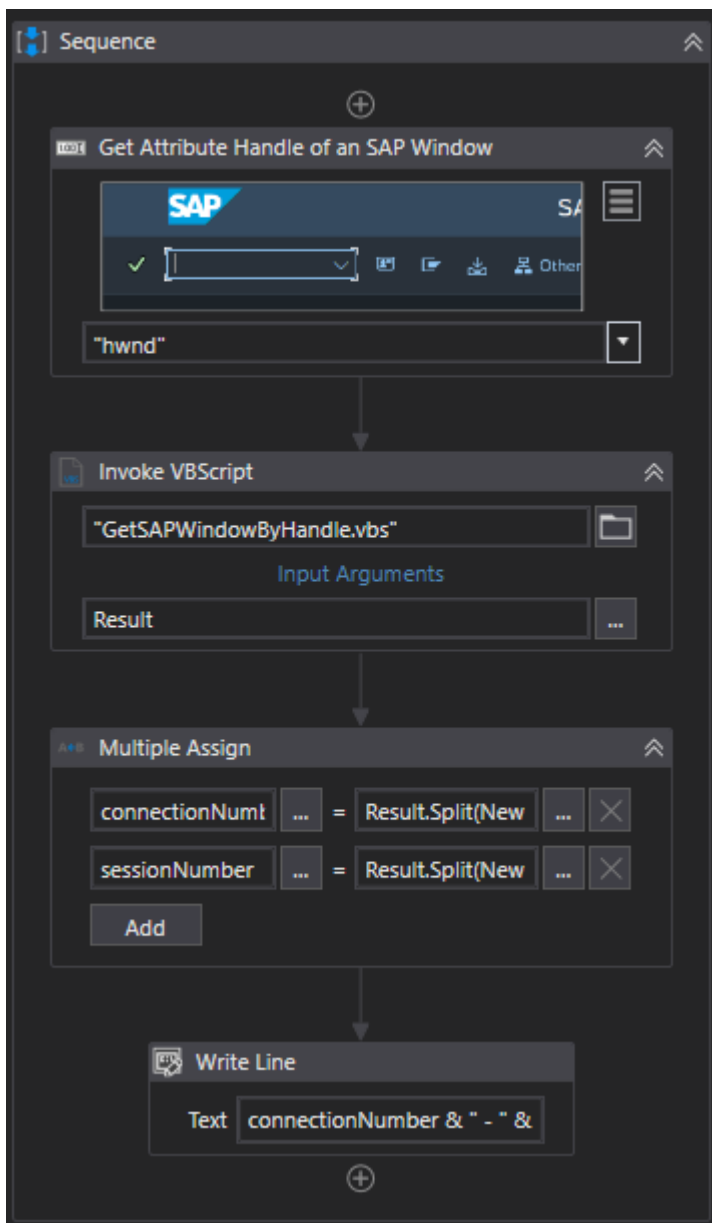
In the context of UiPath are the application, connection and session without significance. UiPath detects the window and an UI element with a selector e.g. like this:

```
<wnd app='saplogon.exe' cls='SAP_FRONTEND_SESSION' title='SAP Easy Access*' />  
<sap id='usr/cntlIMAGE_CONTAINER/shellcont/shell' />
```

As we can see the window is identified with other selectors and below the window the SAP GUI Scripting ID is used, beginning with the user screen /usr.

To get the connection and session number in UiPath it is possible to use the Get Attribute activity with the handle of the window, hwnd. The handle is transfered to a VBScript and the script delivers the session ID, the connection and session number back to the project. But this is not necessary at all. A recorded SAP GUI script can also be inserted in the main routine instead.

With this approach we have the possibility to use existing SAP GUI scripts seamlessly in UiPath. All we have to do is change the recorded connection and session number with variables. The advantages of this approach are, that we can use all the possibilities of the SAP GUI Scripting API and that we can use existing consolidated scripts furthermore. The disadvantage of this approach is, that we pass the control of the automation to another engine, so the complexity increases. It depends on the use case.



```
'-Begin-----
'-Directives-----
Option Explicit

'-Function FindSAPWindowsByHandle-----
Function FindSAPWindowByHandle(hSAPWnd)

    Dim SapGuiAuto, app, CollCon, oCon, CollSes, oSes, hWnd, i, j

    Set SapGuiAuto = GetObject("SAPGUI")
    If Not IsObject(SapGuiAuto) Then
        Exit Function
    End If

    Set app = SapGuiAuto.GetScriptingEngine
    If Not IsObject(app) Then
        Exit Function
    End If

    '-Get all connections-----
    Set CollCon = app.Connections()
    If Not IsObject(CollCon) Then
```

```

Exit Function
End If

'-Loop over connections-----
For i = 0 To CollCon.Count() - 1
    Set oCon = app.Children(CLng(i))
    If Not IsObject(oCon) Then
        Exit Function
    End If

    '-Get all sessions of a connection-----
    Set CollSes = oCon.Sessions()
    If Not IsObject(CollSes) Then
        Exit Function
    End If

    '-Loop over sessions-----
    For j = 0 To CollSes.Count() - 1

        Set oSes = oCon.Children(CLng(j))
        If Not IsObject(oSes) Then
            Exit Function
        End If

        If oSes.Busy() = vbFalse Then
            hWnd = oSes.FindById("wnd[0]").Handle
            If hSAPWnd = hWnd Then
                FindSAPWindowByHandle = oSes.ID
            End If
        End If

    Next

Next

End Function

'-Sub Main-----
Sub Main()

    Dim HandleSAPWindow
    Dim sessionId, connectionNumber, sessionNumber
    Dim pos, Len

    HandleSAPWindow = CLng(WScript.Arguments.Item(0))
    sessionId = CStr(FindSAPWindowByHandle(HandleSAPWindow))

    pos = InStr(sessionId, "con[") + 4
    Len = InStr(pos, sessionId, "]") - pos
    connectionNumber = CLng(Mid(sessionId, pos, Len))

    pos = InStr(sessionId, "ses[") + 4
    Len = InStr(pos, sessionId, "]") - pos
    sessionNumber = CLng(Mid(sessionId, pos, Len))

    WScript.Echo(sessionId & ";" & connectionNumber & ";" & sessionNumber)

End Sub

'-Main-----
Main

'-End-----

```

Session ID and Session Number - UiPath

Beside the handle of the window, which is described [here, how to handle Connection and Session Number](#), it is also possible to use instead the session ID and session number to identify an SAP session window unique.

The script detects the session via SystemSessionID and SessionNumber. Both information delivers the Get Attribute activity. It loops over all connections and sessions to find the correct one. When it has found the right session it calls the Sub Action. In this sub routine you can use the attributes which are not supported by the Get Attribute activity, in this example VisibleRowCount.

```
'-Begin-----  
  
'-Directives-----  
Option Explicit  
  
'-Sub Action-----  
Sub Action(session)  
  
    Dim Id, oElement  
    Id = "wnd[0]/" + WScript.Arguments.Item(2)  
    Set oElement = session.FindById(Id)  
    WScript.Echo CStr(oElement.VisibleRowCount)  
  
End Sub  
  
'-Sub GetSession-----  
Sub GetSession(SessionID, SessionNumber)  
  
    Dim SapAppl, SapGuiAuto, CollCon, i, oCon, CollSes, j, oSes  
    Dim oSesInf, SessID, SessNumber  
  
    Set SapGuiAuto = GetObject("SAPGUI")  
    If Not IsObject(SapGuiAuto) Then  
        WScript.Echo "Error: GetObject"  
        Exit Sub  
    End If  
  
    Set SapAppl = SapGuiAuto.GetScriptingEngine  
    If Not IsObject(SapAppl) Then  
        WScript.Echo "Error: GetScriptingEngine"  
        Exit Sub  
    End If  
  
    Set CollCon = SapAppl.Connections()  
    If Not IsObject(CollCon) Then  
        WScript.Echo "Error: No Connections"  
        Exit Sub  
    End If  
  
    '-Loop over connections-----  
    For i = 0 To CollCon.Count() - 1  
  
        Set oCon = SapAppl.Children(CLng(i))  
        If Not IsObject(oCon) Then  
            WScript.Echo "Error at Connection"  
            Exit Sub  
        End If  
  
        Set CollSes = oCon.Sessions()  
        If Not IsObject(CollSes) Then
```

```

WScript.Echo "Error: No Sessions"
Exit Sub
End If

'-Loop over sessions-----
For j = 0 To CollSes.Count() - 1

    Set oSes = oCon.Children(CLng(j))
    If Not IsObject(oSes) Then
        WScript.Echo "Error at Session"
        Exit Sub
    End If

    If oSes.Busy() = vbFalse Then

        Set oSesInf = oSes.Info()

        If IsObject(oSesInf) Then

            SessID = oSesInf.SystemSessionID()
            SessNumber = CStr(oSesInf.SessionNumber() - 1)

            If SessID = SessionID And SessNumber = SessionNumber Then
                Action oSes
            End If

        End If

    End If

End If

Next

Next

End Sub

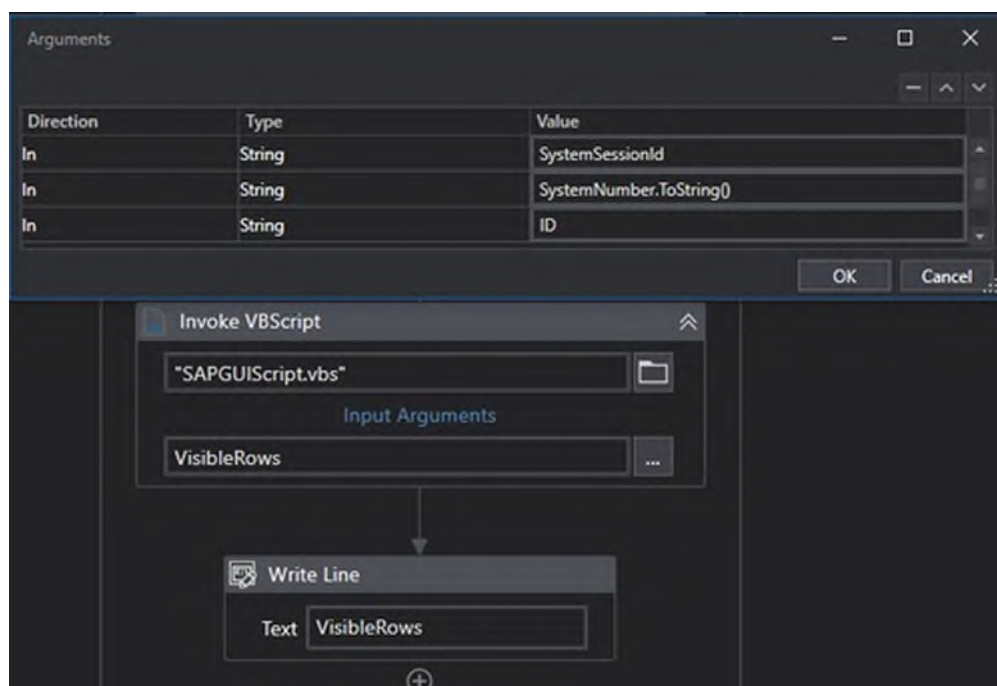
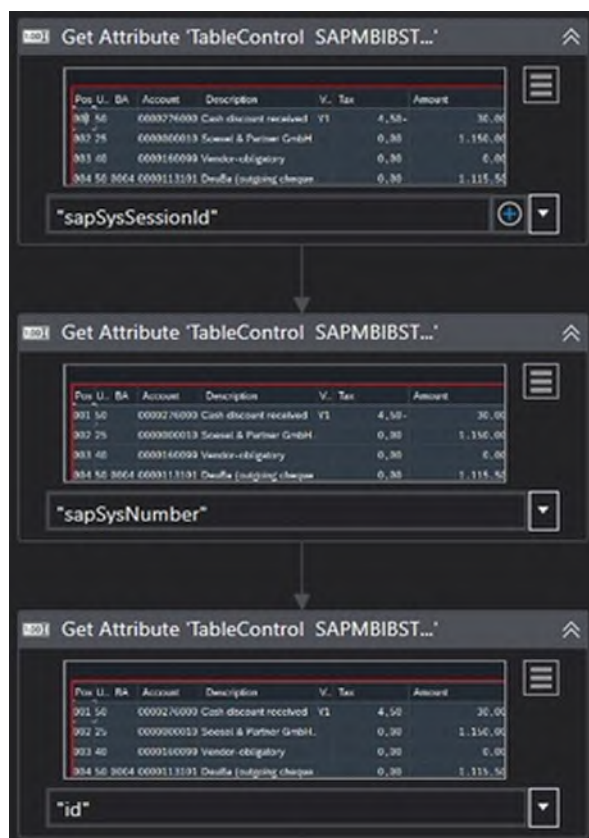
'-Sub Main-----
Sub Main()
    GetSession WScript.Arguments.Item(0), WScript.Arguments.Item(1)
End Sub

'-Main-----
Main

'-End-----

```

In the UiPath workflow I detect at first the attributes SystemSessionId, SessionNumber and the ID of the UI element, in this example a table control.



Hints, Tips and Tricks

- If you got an error, it is possible that the line number of the error message is different from the line number in the editor, because Tracker adds a few lines header automatically.

- If you want to compile the C# or VB.NET code, you can use the command lines

```
vbc.exe [Name of your script file].vb
```

or

```
csc.exe /reference:Microsoft.VisualBasic.dll [Name of your script file].cs
```

For C# it is necessary to add a reference to Microsoft.VisualBasic.dll.

You can download the Roslyn .NET compiler platform from the Nuget Gallery:

<https://www.nuget.org/packages/Microsoft.CodeDom.Providers.DotNetCompilerPlatform>

- To test different UI elements use transaction code GUIBIBS, which starts the program SAPMBIBS, or use transaction code SE38 with the program DEMO_DYNPRO*. These reports are for GUI test and it contains different examples of user interface design.
- Scripting Tracker uses VBScript® only in the context of Windows® Script Host (WSH). If you want to start the SAP® GUI Script via Customize Local Layout > Script Recording and Playback or via drag-and-drop to the session be sure that you don't use any possibilities of the WSH, otherwise you will get an error.
- The WSH offers a lot of additional possibilities, look at the help file, item Windows Script Host Object Model.
- If the program crashes it tries to write a Panic.sav file in the directory of Scripting Tracker.
- If you use another font size in the display setting as 100%, it could be possible that not all field descriptions are fully visible.
- Do not forget to switch the identify button off.
- If you use Scripting Tracker with Windows PowerShell version 2 you must add the following stub in front of your recorded code, because in PowerShell 2 is the variable \$PSScriptRoot not available :

```
If ($PSVersionTable.PSVersion.Major -eq 2) {  
    $PSScriptRoot = Split-Path $($MyInvocation.InvocationName)  
-Parent  
}
```

- Scripting Tracker offers for transparency different information via OutputDebugString
 - External program calls
 - Details about recordingUse [Sysinternals DebugView](#) to get the information.

List of Objects

Hints

- For different types of controls it is possible to use the Demo Center via TAC DWDM or ABAP Examples via TAC ABAPDOCU.
- As alternative to TAC GUIBIBS you can use TAC BIBS dito.
- Another alternative is the TAC GUIT, which calls the report SAPM_GUITEST_PORTABLE.
- Examples for different types of graphics via TAC RGRAPALL, but most examples opens an additional window which can't control via SAP GUI Scripting.
- More examples are available at the package SABAPDEMOS.
- To experiment with program name and screen number use TAC BRFACS02, e.g. SAPLBRF_MAINTENANCE:3006, or SE38 with the report name /SAPDMC/SAPMLSMW.

UI Object	Type	Transaction Code / Script / Library
GUIABAPeEditor	GuiShell SubType ABAPeEditor	SE80 AbapEditorScripting.dll
GUIApoGrid		
GUIApplication	GuiApplication	
GUIBarChart	GuiShell SubType BarChart	SE38 - BARCOCX1 SapBarcScripting.dll
GUIBox	GuiBox	SE38 - DEMO_DYNPRO_SPLITTER_CONTROL
GUIButton	GuiButton	GUIBIBS SE38 - DEMO_DYNPRO_PUSH_BUTTON
GUICalendar	GuiShell SubType Calendar	SE38 - SAPCALENDAR_DEMO1 SapCalenScripting.dll
GUICart	GuiShell SubType Chart	SE38 - GFW_PROG_TUTORIAL SE38 - GFW_PROG_PIE ChartScripting.dll

GUICheckBox	GuiCheckBox	GUIBIBS
GUICollection	GuiCollection	
GUIColorSelector	GuiShell SubType ColorSelector	SE38 - DEMO_COLORSEL SapSelScripting.dll
GUIComboBox	GuiComboBox	GUIBIBS SE38 - DEMO_DYNPRO_DROPDOWN_LISTBOX ListControlScripting.dll
GUIComboBoxControl		
GUIComboBoxEntry		
GUIComponent		
GUIComponentCollection		
GUIConnection	GuiConnection	
GUIContainer		
GUIContainerShell	GuiContainerShell	SE38 - GRAPHICS_GUI_CE_DEMO
GUIContextMenu		
GUITextField		
GUICustomControl	GuiCustomControl	SE38 - RSDemo_CUSTOM_CONTROL
GUIDialogShell		
GUIEAViewer2D		EAI2DScripting.dll
GUIEAViewer3D		EAI3DScripting.dll
GUIFrameWindow		

GUIGOSShell	GuiShell SubType ToolBar	SGOSTEST SE38 - GOS_TOOLBOX_TEST
GUIGraphAdapt		
GUIGridView	GuiShell SubType GridView (ALV-Grid)	SE80 - Package SLIS Programs BCALV_GRID* SE38 - BCALV_TEST_SUITE GridViewScripting.dll
GUIHTMLViewer	GuiShell SubType HTMLViewer	SE38 - DEMO_CREATE_HTML_MODERN SE38 - SAPHTML_DEMO1 SE80 - Package SAPHTML Programs SAPHTML_* To try Edge (based on Chromium) use SE38 - RSDemo_HTML_VIEWER Open, with a right mouse click, the context menu and choose Inspect to open the developer tools, before you load a site. SapHtmlScripting.dll
GUIInputFieldControl		ListControlChildScripting.dll
GUILabel	GuiLabel	GUIBIBS
GUIMainWindow	GuiMainWindow	SESSION_MANAGER GUIBIBS
GUIMap		SapMapScripting.dll
GUIMenu	GuiMenu	SESSION_MANAGER
GUIMenuBar		
GUIMessageWindow		
GUIModalWindow	GuiModalWindow	SE38 - DEMO_CALCULATOR_MODERN1 SE37 - POPUP_TO_INFORM
GUINetChart		SapNetzScripting.dll
GUIOfficeIntegration	GuiShell SubType	SE38 - SAPRDEMO_FORM_INTERFACE

	OfficeIntegration	SAPSDCCScripting.dll
GUIOkCodeField	GuiOkCodeField	SESSION_MANAGER
GUIPasswordField	GuiPasswordField	SESSION_MANAGER
GUIPicture	GuiShell SubType Picture	SE38 - SAP_PICTURE_DEMO SapImageScripting.dll
GUIRadioButton	GuiRadioButton	GUIBIBS
GUISapChart		SapChartScripting.dll
GUIScrollbar	GuiScrollbar	GUIBIBS
GUIScrollContainer	GuiScrollContainer	SE38 - DEMO_DYNPRO_SPLITTER_CONTROL
GUISession	GuiSession	
GUISessionInfo	GuiSessionInfo	GuiSessionInfo
GUIShell		
GUISimpleContainer	GuiSimpleContainer	SE38 - DEMO_DYNPRO_SUBSCREENS
GUISplit		SE38 - DEMO_CFW SE38 - DEMO_CFW2 SapSplitScripting.dll
GUISplitterContainer	GuiSplitterContainer	SE38 - DEMO_DYNPRO_SPLITTER_CONTROL
GUIStage		SapStageScripting.dll
GUIStatusbar	GuiStatusbar	GUIBIBS
GUIStatusbarLink		
GUIStatusPane	GuiStatusPane	GUIBIBS

GUITab	GuiTab	GUIBIBS SE38 - DEMO_DYNPRO
GUITableColumn		
GUITableControl	GuiTableControl	GUIBIBS
GUITableRow		
GUITabStrip	GuiTabStrip	SE38 - DEMO_DYNPRO
GUITextEdit	GuiTextEdit	SE80 - Package SAPTEXTEDIT Programs SAPTEXTEDIT_* SE38 - SAP_FULLSCREEN_CONTAINER_DEMO TextEditScripting.dll
GUITextField	GuiTextField	GUIBIBS
GUITitleBar	GuiTitleBar	
GUIToolBar	GuiShell SubType ToolBar	SE38 - BCALV_TREE_DND_MULTIPLE SapToolbScripting.dll
GUIToolBarControl		
GUITree	GuiShell SubType Tree	SE80 - Package SLIS Programs BCALV_TREE* WdtTreeScripting.dll
GUIUserArea	GuiUserArea	
GUIUtils	GuiUtils	GuiUtils
GUIVComponent		
GUIVContainer		
GUIViewSwitchTarget		

List Of Objects - GuiSessionInfo

```
#-Begin-----  
  
$Info = Get-Property -object $session "Info";  
$Transaction = Get-Property -object $Info -propertyName "Transaction";  
Write-Host "Tansaction:    " $Transaction;  
$Program = Get-Property -object $Info -propertyName "Program";  
Write-Host "Program:          " $Program;  
$ScreenNumber = Get-Property -object $Info -propertyName "ScreenNumber";  
Write-Host "ScreenNumber: " $ScreenNumber;  
$CodePage = Get-Property -object $Info -propertyName "CodePage";  
Write-Host "CodePage:      " $CodePage;  
$GuiCodePage = Get-Property -object $Info -propertyName "GuiCodePage";  
Write-Host "GuiCodePage:   " $GuiCodePage;  
$I18NMode = Get-Property -object $Info -propertyName "I18NMode";  
Write-Host "I18NMode:      " $I18NMode;  
$Language = Get-Property -object $Info -propertyName "Language";  
Write-Host "Language:      " $Language;  
$IsLowSpeed = Get-Property -object $Info -propertyName "IsLowSpeedConnection";  
Write-Host "IsLowSpeed:    " $IsLowSpeed;  
[Void] [Console]::WriteLine("Press key...");  
[Void] [Console]::ReadKey("NoEcho, IncludeKeyDown");  
  
#-End-----
```

List of Objects - GuiUtils

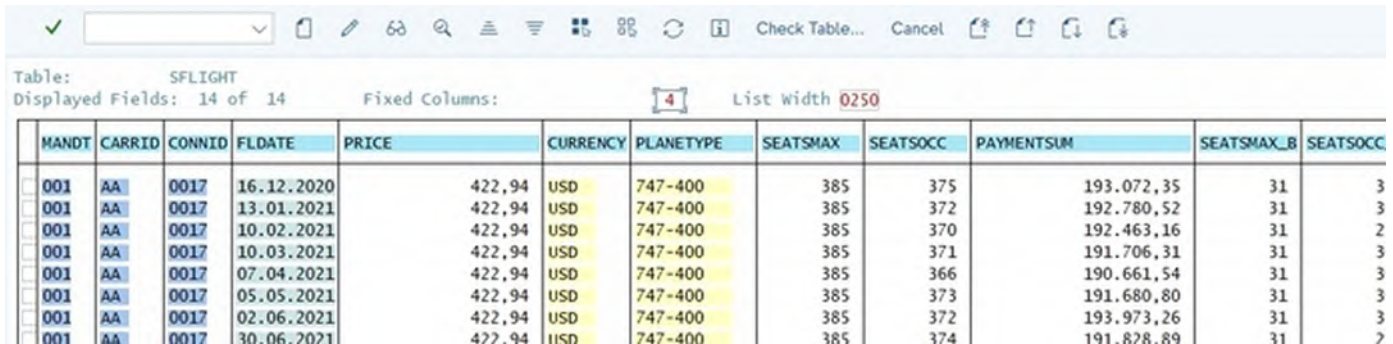
```
#-Begin-----  
  
$Utils = Get-Property -object $Application -propertyName "Utils";  
$hFile = Invoke-Method -object $Utils -methodName "OpenFile" -methodParam  
@("Test.txt");  
If ($hFile -ne 0) {  
    Invoke-Method -object $Utils -methodName "WriteLine" -methodParam @($hFile,  
"This is a test");  
    Invoke-Method -object $Utils -methodName "CloseFile" -methodParam @($hFile)  
}  
$MsgIcon = Get-Property -object $Utils -propertyName "MESSAGE_OPTION_OK";  
$MsgType = Get-Property -object $Utils -propertyName "MESSAGE_TYPE_PLAIN";  
Invoke-Method -object $Utils -methodName "ShowMessageBox" -methodParam  
@("Hint", "Ready", $MsgIcon, $MsgType) > $Null;  
  
#-End-----
```


List of Objects - GuiGridView (ALV)

The ABAP List Viewer (ALV) offers the possibility to view data in a tabular or hierarchical format. The ALV is also known as SAP List Viewer. It offers a friendly interface with a toolbar that allows to adjust the presented layout, to sort or filter data and to export data very easily. SAP offers three different types of the table display.

ABAP List

This is not an ALV display, this output using the ABAP WRITE command. The difference to the ALV List is the toolbar, e.g. you can not sum, export or filter.



MANDT	CARRID	CONNID	FLDATE	PRICE	CURRENCY	PLANETYPE	SEATSMAX	SEATSOCC	PAYMENTSUM	SEATSMAX_B	SEATSOCC
001	AA	0017	16.12.2020	422,94	USD	747-400	385	375	193.072,35	31	3
001	AA	0017	13.01.2021	422,94	USD	747-400	385	372	192.780,52	31	3
001	AA	0017	10.02.2021	422,94	USD	747-400	385	370	192.463,16	31	2
001	AA	0017	10.03.2021	422,94	USD	747-400	385	371	191.706,31	31	3
001	AA	0017	07.04.2021	422,94	USD	747-400	385	366	190.661,54	31	3
001	AA	0017	05.05.2021	422,94	USD	747-400	385	373	191.680,80	31	3
001	AA	0017	02.06.2021	422,94	USD	747-400	385	372	193.973,26	31	3
001	AA	0017	30.06.2021	422,94	USD	747-400	385	374	191.828,89	31	2

ALV List

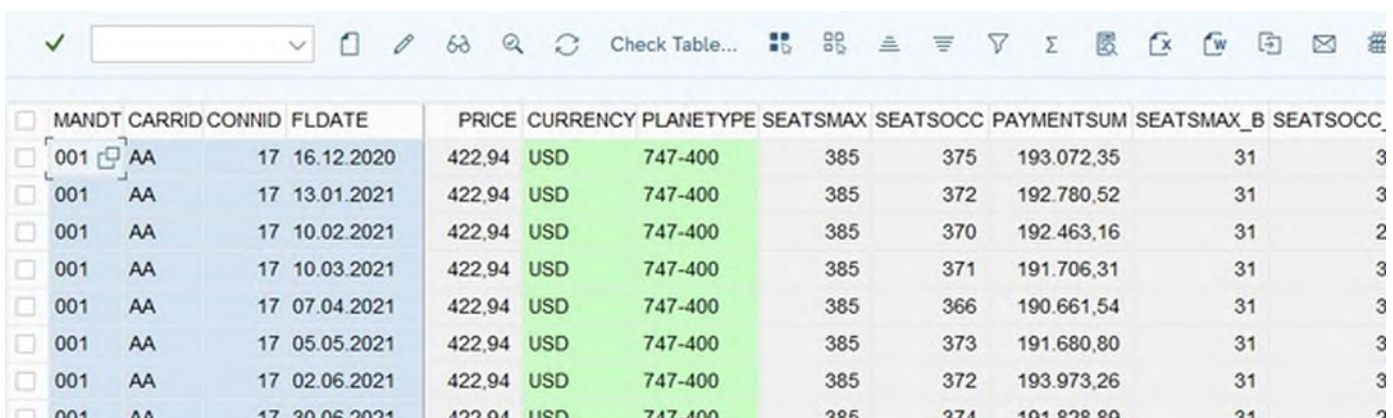
The ALV List, also known as Classic SAP List Viewer, is very comparable to the ABAP list from the automation perspective, it uses labels too. The data presentation is different and the toolbar offers more features.



MANDT	CARRID	CONNID	FLDATE	PRICE	CURRENCY	PLANETYPE	SEATSMAX	SEATSOCC	PAYMENTSUM	SEATSMAX_B	SEATSOCC
001	AA	17	16.12.2020	422,94	USD	747-400	385	375	193072,35	31	3
001	AA	17	13.01.2021	422,94	USD	747-400	385	372	192780,52	31	3
001	AA	17	10.02.2021	422,94	USD	747-400	385	370	192463,16	31	2
001	AA	17	10.03.2021	422,94	USD	747-400	385	371	191706,31	31	3
001	AA	17	07.04.2021	422,94	USD	747-400	385	366	190661,54	31	3
001	AA	17	05.05.2021	422,94	USD	747-400	385	373	191680,80	31	3
001	AA	17	02.06.2021	422,94	USD	747-400	385	372	193973,26	31	3
001	AA	17	30.06.2021	422,94	USD	747-400	385	374	191828,89	31	2

ALV Grid

The ALV Grid is more like an Microsoft Excel table format



MANDT	CARRID	CONNID	FLDATE	PRICE	CURRENCY	PLANETYPE	SEATSMAX	SEATSOCC	PAYMENTSUM	SEATSMAX_B	SEATSOCC
001	AA	17	16.12.2020	422,94	USD	747-400	385	375	193.072,35	31	3
001	AA	17	13.01.2021	422,94	USD	747-400	385	372	192.780,52	31	3
001	AA	17	10.02.2021	422,94	USD	747-400	385	370	192.463,16	31	2
001	AA	17	10.03.2021	422,94	USD	747-400	385	371	191.706,31	31	3
001	AA	17	07.04.2021	422,94	USD	747-400	385	366	190.661,54	31	3
001	AA	17	05.05.2021	422,94	USD	747-400	385	373	191.680,80	31	3
001	AA	17	02.06.2021	422,94	USD	747-400	385	372	193.973,26	31	3
001	AA	17	30.06.2021	422,94	USD	747-400	385	374	191.828,89	31	2

List of Objects - GuiTree

Development Class
SEU_TREE_MODEL





Reports

SAPSIMPLE_TREE_MODEL_DEMO
SAPTLIST_TREE_MODEL_DEMO
SAPTLIST_TREE_CONTROL_DEMO_HDR
SAPCOLUMN_TREE_MODEL_DEMO

Simple Tree

Root
Chld 1
New 1
New 2








List Tree without header

✓	Objects		
✓	Dynpros		
			0100 MUELLER Comment for Dynpro 100
			0200 HARRYHIRSCH Comment for Dynpro 200
✓	Programs		
			SAPTROX1 Comment on SAPTROX1
			SAPTRIXTROX Comment on SAPTRIXTROX

List Tree with header

Hierarchy Header	List Header
Objekte	
Dynpros	
Mask 1	010 MUELLER Comment to Dynpro 100
Mask 2	020 HARRYHIRSC Comment to Dynpro 200
Programme	
Prog 1	SAPTROX1 Comment to SAPTROX1
Prog 2	SAPTRIXTRO Comment to SAPTRIXTROX

Column Tree

Hierarchy Header	Column2	Column3
√  Root Column1	Root Column2	Root Column3
√  Chld1 Column1	 Chld1 Column2	Chld1 Column3  <input type="checkbox"/>
 New1 Column1		New1 Column3
 New2 Column1	New2 Column2	New2 Column3

Object, Prefix and Dynpro

No.	UI Object	Prefix	Dynpro Element Type
1	GUIABAPEditor	cntl	CUCTR
2	GUIApoGrid		
3	GUIApplication	app	
4	GUIBarChart		
5	GUIBox	box	FRAME
6	GUIButton	btn	PUSH
7	GUICalendar	cntl	CUCTR
8	GUICart		
9	GUICheckBox	chk	CHECK
10	GUICollection		
11	GUIColorSelector	cntl	CUCTR
12	GUIComboBox	cmb	I/O
13	GUIComboBoxControl		
14	GUIComboBoxEntry		
15	GUIComponent		
16	GUIComponentCollection		
17	GUIConnection	con	
18	GUIContainer		
19	GUIContainerShell	shellcont	

20	GUIContextMenu		
21	GUITextField	ctxt	I/O
22	GUICustomControl	cntl	CUCTR
23	GUIDialogShell	shellcont	
24	GUIEAViewer2D		
25	GUIEAViewer3D		
26	GUIFrameWindow	wnd	
27	GUIGOSShell	shellcont	
28	GUIGraphAdapt		
29	GUIGridView		
30	GUIHTMLViewer	cntl	CUCTR
31	GUIInputFieldControl		
32	GUILabel	lbl	TEXT
33	GUIMainWindow	wnd	
34	GUIMap		
35	GUIMenu	menu	
36	GUIMenuBar	mbar	
37	GUIMessageWindow		
38	GUIModalWindow	wnd	
39	GUINetChart		

40	GUIOfficeIntegration	cntl	CUCTR
41	GUIOkCodeField	okcd	OK
42	GUIPasswordField	pwd	
43	GUIPicture		
44	GUIRadioButton	rad	RADIO
45	GUI SapChart		
46	GUIScrollbar		
47	GUI ScrollContainer	ssub	SUBSC
48	GUI Session	ses	
49	GUI SessionInfo	:	
50	GUI Shell	shell	
51	GUI SimpleContainer	sub	
52	GUI Split		
53	GUI SplitterContainer	spkc	SPCTR
54	GUI Stage		
55	GUI StatusBar	sbar	
56	GUI StatusBarLink		
57	GUI StatusPane	pane	
58	GUI Tab	tabp	PUSH
59	GUI TableColumn		

60	GUITableControl	tbl	TABLE
61	GUITableRow		
62	GUITabStrip	tabs	TBSTR
63	GUITextEdit		
64	GUITextField	txt	I/O
65	GUITitleBar	titl	
66	GUIToolBar	tbar	
67	GUIToolBarControl		
68	GUITree		
69	GUIUserArea	usr	
70	GUIUtils	.	
71	GUIVComponent		
72	GUIVContainer		
73	GUIViewSwitchTarget		

Object, Prefix and Dynpro - Z_EXPORT_FIELDS

```
"-Begin-----
"-
"- ABAP program to export all dynpro fields of a development class as
"- a csv file
"-
"-----
REPORT z_export_fields.

INCLUDE MSEUSBIT.

DATA: BEGIN OF id,
      p TYPE progname,
      d TYPE sydynnr,
    END OF id.

TYPES: BEGIN OF ty_id,
      prog TYPE progname,
      dnum TYPE sydynnr,
    END OF ty_id.

TYPES: BEGIN OF ty_prog,
      object TYPE trobjtype,
      devclass TYPE devclass,
      obj_name TYPE sobj_name,
    END OF ty_prog.

TYPES: BEGIN OF ty_res,
      devclass TYPE devclass,      "Development Class
      prog TYPE progname,          "Programname
      obj_type TYPE trobjtype,     "PROG or FUGR
      dnum TYPE sychar04,          "Dynpro-Number
      cupo TYPE fnam____4,        "Dynpro-Name
      fname TYPE fnam____4,       "Fieldname
      type_short TYPE scrfgtyp,    "Fieldtype short
      type_long TYPE scrfgtyp,    "Fieldtype long
      stext TYPE stxt____1,       "Fieldtext
      ddicfield TYPE boolean,     "Flag if data dictionary field
      rollname TYPE rollname,     "Data element
      checktable TYPE checktable, "Table name of the foreign key
      inttype TYPE inttype,       "ABAP data type
      intlen TYPE intlen,         "Length in Bytes
    END OF ty_res.

DATA:
  lv_header TYPE d020s,
  ls_field TYPE d021s,
  lt_field TYPE TABLE OF d021s,
  lt_flow_logic TYPE TABLE OF d022s,
  lt_matchcode_info TYPE TABLE OF d023s,
  ls_id TYPE ty_id,
  lt_id TYPE STANDARD TABLE OF ty_id,
  ls_res TYPE ty_res,
  lt_res TYPE STANDARD TABLE OF ty_res,
  lv_res_fname TYPE fnam____4,
  lv_file TYPE string,
  ls_prog TYPE ty_prog,
  lt_prog TYPE STANDARD TABLE OF ty_prog,
  lv_tablename TYPE tabname,
  lv_fieldname TYPE fieldname,
  ls_dd03l TYPE dd03l,
```



```

lv_off TYPE i,
lv_len TYPE i.

FIELD-SYMBOLS:
<ls_prog> TYPE ty_prog,
<ls_res> TYPE ty_res.

SELECTION-SCREEN BEGIN OF SCREEN 1001.
SELECTION-SCREEN BEGIN OF LINE.
SELECTION-SCREEN COMMENT 1(30) cm_devcl FOR FIELD p_devcl.
PARAMETERS: p_devcl TYPE DEVCLASS OBLIGATORY.
SELECTION-SCREEN END OF LINE.
SELECTION-SCREEN BEGIN OF LINE.
SELECTION-SCREEN COMMENT 1(30) cm_file FOR FIELD p_file.
PARAMETERS: p_file TYPE sapb-sappfad OBLIGATORY LOWER CASE.
SELECTION-SCREEN END OF LINE.
SELECTION-SCREEN END OF SCREEN 1001.

CALL SELECTION-SCREEN 1001.

INITIALIZATION.
cm_devcl = 'Development Class:'.
cm_file = 'Filename:'.

START-OF-SELECTION.

"-Select programs (PROG) and function groups (FUGR)-----
SELECT object, devclass, obj_name
FROM TADIR
INTO CORRESPONDING FIELDS OF TABLE @lt_prog
WHERE devclass LIKE @p_devcl AND
( object = 'PROG' OR object = 'FUGR' )
ORDER BY devclass, obj_name.
CHECK sy-subrc = 0.

"-Modify program names of function groups-----
LOOP AT lt_prog ASSIGNING <ls_prog>.
CHECK <ls_prog>-object = 'FUGR'.
IF <ls_prog>-obj_name(1) = '/'.
FIND FIRST OCCURRENCE OF REGEX '(?!.*\//).*' IN <ls_prog>-obj_name
MATCH OFFSET lv_off MATCH LENGTH lv_len.
<ls_prog>-obj_name = <ls_prog>-obj_name+0(lv_off) && 'SAPL' &&
<ls_prog>-obj_name+lv_off(lv_len).
CONDENSE <ls_prog>-obj_name.
ELSE.
<ls_prog>-obj_name = 'SAPL' && <ls_prog>-obj_name.
ENDIF.
ENDLOOP.

LOOP AT lt_prog INTO ls_prog.

SELECT prog, dnum
FROM D020S
INTO CORRESPONDING FIELDS OF TABLE @lt_id
WHERE prog = @ls_prog-obj_name
ORDER BY prog, dnum.
CHECK sy-subrc = 0.

LOOP AT lt_id INTO ls_id.

id-p = ls_id-prog.
id-d = ls_id-dnum.

"-Gets data from DYNPSOURCE table-----
IMPORT DYNPRO lv_header lt_field lt_flow_logic lt_matchcode_info ID

```

id.

```
LOOP AT lt_field INTO ls_field.

    ls_res-devclass = ls_prog-devclass.
    ls_res-prog = lv_header-prog.
    ls_res-obj_type = ls_prog-object.
    ls_res-dnum = lv_header-dnum.
    ls_res-cupo = lv_header-cupo.
    ls_res-fname = ls_field-fnam.
    IF ls_field-stxt CN '_ '.
        ls_res-stext = ls_field-stxt.
    ELSE.
        ls_res-stext = ''.
    ENDIF.

    CALL FUNCTION 'RS_SCRP_GET_FIELD_TYPE_TEXT'
        EXPORTING
            field = ls_field
            text_kind = 'SHORT'
        IMPORTING
            field_type_without_modif = ls_res-type_short
        EXCEPTIONS
            OTHERS = 1.
    TRANSLATE ls_res-type_short TO UPPER CASE.

    CALL FUNCTION 'RS_SCRP_GET_FIELD_TYPE_TEXT'
        EXPORTING
            field = ls_field
        IMPORTING
            field_type_without_modif = ls_res-type_long
        EXCEPTIONS
            OTHERS = 1.

    IF ls_field-flg1 O FLG1DDF.
        CASE ls_res-type_short.
            WHEN 'I/O' OR 'TEXT' OR 'OK' OR 'CHECK' OR 'RADIO'.
                ls_res-ddicfield = abap_true.
            WHEN OTHERS.
                ls_res-ddicfield = abap_false.
        ENDCASE.
    ELSE.
        ls_res-ddicfield = abap_false.
    ENDIF.

    APPEND ls_res TO lt_res.

ENDLOOP.

ENDLOOP.

ENDLOOP.

LOOP AT lt_res ASSIGNING <ls_res> WHERE ddicfield = abap_true.
    IF <ls_res>-fname(1) = '*'.
        lv_len = strlen( <ls_res>-fname ) - 1.
        lv_res_fname = <ls_res>-fname+1(lv_len).
    ELSE.
        lv_res_fname = <ls_res>-fname.
    ENDIF.
    SPLIT lv_res_fname AT '-' INTO lv_tablename lv_fieldname.
    SELECT SINGLE tablename, fieldname, as4local, rollname, checktable,
        inttype, intlen
    FROM dd03l
    INTO CORRESPONDING FIELDS OF @ls_dd03l
```

```

WHERE tablename = @lv_tablename AND
      fieldname = @lv_fieldname AND
      as4local = 'A'.
CHECK sy-subrc = 0.
<ls_res>-rollname = ls_dd03l-rollname.
<ls_res>-checktable = ls_dd03l-checktable.
<ls_res>-inttype = ls_dd03l-inttype.
<ls_res>-intlen = ls_dd03l-intlen.
ENDLOOP.

lv_file = p_file.

Call Method cl_gui_frontend_services=>gui_download
EXPORTING
    filename                = lv_file
    filetype                 = 'ASC'
    write_field_separator   = 'X'
    trunc_trailing_blanks   = 'X'
    trunc_trailing_blanks_eol = 'X'
CHANGING
    data_tab                = lt_res
EXCEPTIONS
    others                  = 1.

```

"-End-----

SAP GUI Scripting Object Model

The ID of an UI element represents a object hierarchy for a unique identification.

An example:

/app/con[0]/ses[0]/wnd[0]/usr/cntlIMAGE_CONTAINER

The ID contains the following elements:

app = GuiApplication

Represents the process in which all SAP GUI activities runs, of the SAPlogon process.

con = GuiConnection

Represents the connection between SAP GUI and an application server.

ses = GuiSession

Represents the context in which a user performs their tasks, e.g. working with a transaction.

wnd = GuiMainWindow or GuiModalWindow

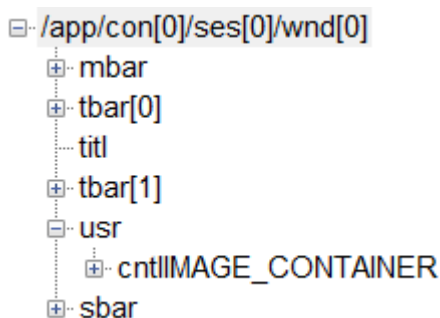
Represents the main window of a SAP GUI session.

After the window follows the [element of the session](#), in this example **usr** for the user screen. But it could also be mbar (menu bar) or tbar (tool bar).

The sum of all these elements are called *parent object ID*.

Now it follows the name of the object, consisting of a [prefix](#), in our case **cntl**, and its technical name, **IMAGE_CONTAINER**.

If an ID is not unique an one or two dimensional index is added, [0] or [1,2].



Java™ and JShell

- The approach to use SAP GUI Scripting with Java™ or JShell needs an JDK version 9 or higher.
- Set the JAVA_HOME environment variable to your JDK directory.
- Add to your path environment variable the bin directory of the JDK directory.
- The approach to use SAP GUI Scripting with Java™ or JShell needs Java COM Bridge (Jacob).
Jacob is delivered with Scripting Tracker.
You can find it here: github.com/freemansoft/jacob-project.
- It is necessary to add the path of the Jacob.jar file to the class path of the Java™ compiler, if you want to compile your code, e.g.

```
javac -CP C:\Dummy\JaCoB SAPGUIScripting.java
```
- It is necessary to add the path of the Jacob.jar file to the class path and the path to the native Jacob-DLLs to the java.exe via -Djava.library.path=[Path], if you want to execute your code with Java™ e.g.

```
java -CP .;C:\Dummy\JaCoB -Djava.library.path=C:\Dummy\JaCoB SAPGUIScripting
```
- It is necessary to add the path of Jacob.jar file to the class path of JShell, Scripting Tracker does that for you automatically, e.g.

```
/env -class-path C:\Dummy\JaCoB\jacob.jar
```
- It is necessary to add the path to the native Jacob-DLLs to the Windows Path environment variable.

Selenium WebDriver

- To combine SAP GUI Scripting with some activities in the web browser you can use Selenium from <http://www.seleniumhq.org/>.
- To use in this context Google Chrome browser you can load the web driver from <https://sites.google.com/a/chromium.org/chromedriver/downloads>.
- You can find the web driver specification here <https://w3c.github.io/webdriver/>

Check SAP GUI Class Instanciation

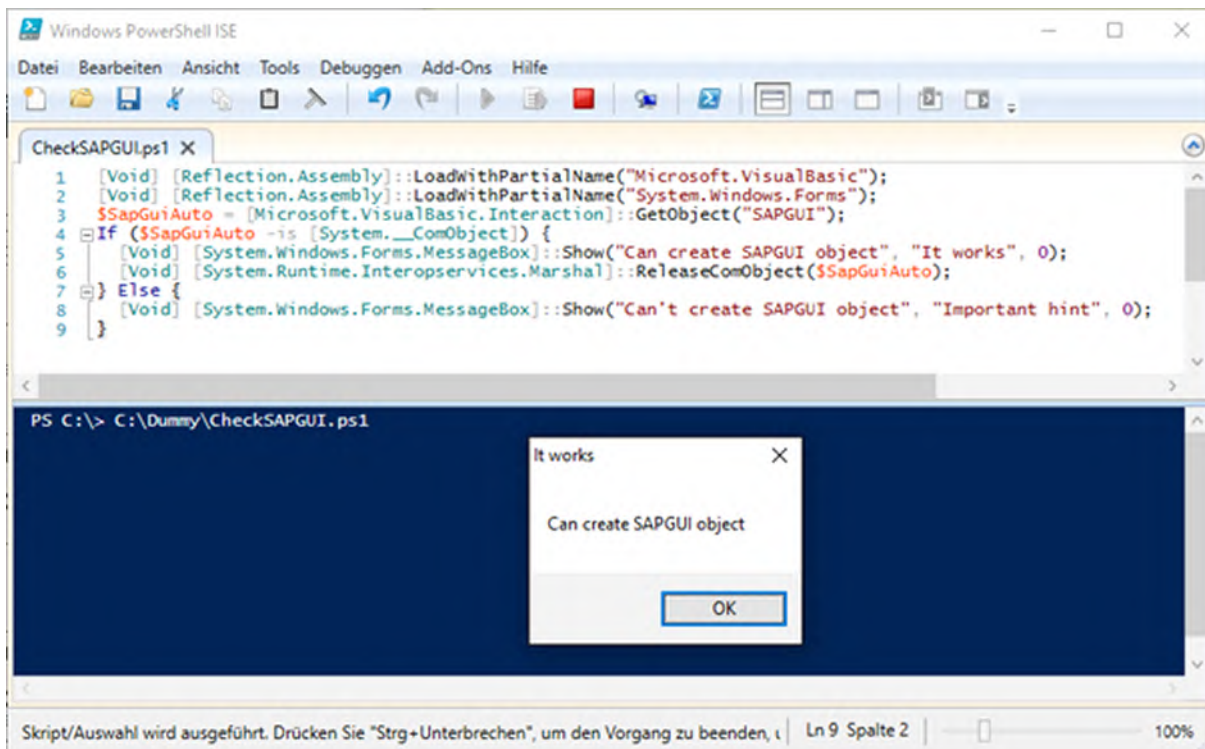
Execute the following code inside your PowerShell ISE to check the SAP GUI class instanciation.

```
Add-Type -AssemblyName "Microsoft.VisualBasic";
Add-Type -AssemblyName "System.Windows.Forms";

$SapGuiAuto = [Microsoft.VisualBasic.Interaction]::GetObject("SAPGUI");

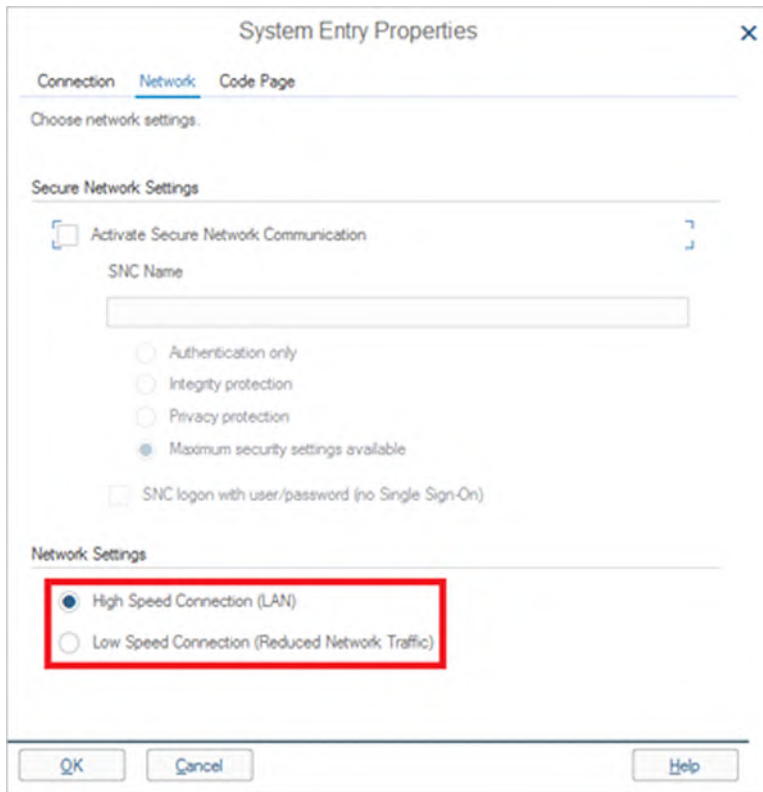
If ($SapGuiAuto -is [System.__ComObject]) {
    [Void] [System.Windows.Forms.MessageBox]::Show("Can create SAPGUI object",
    "It works", 0);
    [Void]
[System.Runtime.InteropServices.Marshal]::ReleaseComObject($SapGuiAuto);
} Else {
    [Void] [System.Windows.Forms.MessageBox]::Show("Can't create SAPGUI object",
    "Important hint", 0);
}
```

You must see a message box like in the image below.



Check Network Settings

It is necessary to set the network settings of a system entry to high speed connection. If it is set to low speed connection the names of the SAP GUI Scripting objects are not transmitted and therefore IDs don't work.



You can switch between high and low speed of your LAN connection in the properties of each connection in the SAP Logon.

Network Settings

- ☒ High Speed Connection (LAN)
- ☐ Low Speed Connection (Reduced Network Traffic)

With low speed connection you lose in some cases information of the ID, here an example. At first recorded code with high speed LAN connection:

```
session.findById("wnd[0]/tbar[0]/okcd").text = "/nSE80"
session.findById("wnd[0]").sendVKey 0
session.findById("wnd[0]/shellcont/shell/shellcont[3]/shell/shellcont[2]/shell").selectNode "1"
session.findById("wnd[0]/shellcont/shell/shellcont[3]/shell/shellcont[2]/shell").nodeContextMenu "1"
session.findById("wnd[0]/shellcont/shell/shellcont[3]/shell/shellcont[2]/shell").selectContextMenuItem "_P__WB_CREATE"
session.findById("wnd[1]/usr/chkRSEUR-WITH_TOP").selected = false
session.findById("wnd[1]/usr/txtRSEUR-TDPROGRAM").text = "Z_TEST"
session.findById("wnd[1]/usr/txtRSEUR-TDPROGRAM").caretPosition = 6
session.findById("wnd[1]/tbar[0]/btn[0]").press
session.findById("wnd[1]/usr/cmbTRDIR-RSTAT").setFocus
session.findById("wnd[1]/usr/cmbTRDIR-RSTAT").key = "T"
session.findById("wnd[1]/tbar[0]/btn[0]").press
```


Here now the same code with low speed LAN connection:

```
session.findById("wnd[0]/tbar[0]/okcd").text = "/nSE80"
session.findById("wnd[0]").sendVKey 0
session.findById("wnd[0]/shellcont/shell/shellcont[3]/shell/shellcont[2]/shell").selectNode "1"
session.findById("wnd[0]/shellcont/shell/shellcont[3]/shell/shellcont[2]/shell").nodeContextMenu "1"
session.findById("wnd[0]/shellcont/shell/shellcont[3]/shell/shellcont[2]/shell").selectContextMenuItem "_P__WB_CREATE"
session.findById("wnd[1]/usr/chk").selected = false
session.findById("wnd[1]/usr/txt").text = "Z_TEST"
session.findById("wnd[1]/usr/txt").caretPosition = 6
session.findById("wnd[1]/tbar[0]/btn[0]").press
session.findById("wnd[1]/usr/cmb[1]").setFocus
session.findById("wnd[1]/usr/cmb[1]").key = "T"
session.findById("wnd[1]/tbar[0]/btn[0]").press
```

Here the explanation from SAP note 161053:

When activating the "Low Speed Connection", the dataset sent to the front end is reduced at the expense of the usability. In addition, if you use the "Low Speed Connection" flag, problems can occur in SAP GUI Scripting, since the field names are no longer available in full. Specifically, this results in problems with the use of the command FindById, but also with other commands.

Restrictions when Using SAP GUI Scripting

This text is an citation from SAP note 587202.

Some technologies are not supported during scripting.

- *F4 search help control (amodal)*
The control is not supported in the scripting. Instead, a standard dialog is opened. In some transactions, this dialog does not open and a short dump occurs due to an error in the application. Until a Support Package corrects the application, you can work around this error by using the menu path "Help->Settings->F4" to select the modal dialog manually.
- *SAPscript*
The text control of the SAPscript component is not supported. It is replaced by a line editor, as described in SAP Notes 64634 and 100358 (point 10). If you have additional questions, contact the component BC-SRV-SCR.
- *Drag and Drop*
Drag and Drop is not supported in scripting. However, you should have the option of using the function without drag and drop in all applications.
- [Low-speed connection](#)
If the low-speed connection indicator is set for a connection, the system transfers less information to the SAP GUI. As a result, the scripting component is missing the field names that are required for the names and IDs of the objects in the scripting model. Errors then occur (for example, with FindById).
- *Missing support in individual transactions*
Certain transactions use dynamic keywords when communicating with the SAP system; these dynamic keywords change each time the transaction is called. This problem may occur when you select entries from the menu of the toolbar control in particular. If the script that is recorded in this transaction is run again, errors occur due to invalid parameters (for example, in the method SelectMenuItem).
- *Missing support for certain ActiveX components*
In order for you to reach an ActiveX control from scripting, scripting support must be made available explicitly. This has already been done for the standard controls. However, some applications contain controls that were developed by customers; no support for scripting exists for them.
- *No support for Microsoft common dialogs*
Scripting for common dialogs (such as FileSave, FileOpen) is not supported.
- *No recording of key combinations or actions that do not change the status of the control*
The key combinations or other actions that do not cause standard changes of the control - for example, "Copy to Clipboard" (CTRL + C) - are not recorded.
- The "advanced search" in input fields is not supported while scripting is active.

PowerShell - Set Execution Policy

It is necessary to set the execution policy of PowerShell. Open the PowerShell Integrated Scripting Environment (ISE) as administrator and type the following command into the commandline and press return.

```
Set-ExecutionPolicy -ExecutionPolicy RemoteSigned
```

Hint: It is necessary to set the execution policy twice, one time for x64 and one time for x86 version. Start one time the ISE from System32 subdirectory and one time from SysWOW64.

Python

- The approach to use SAP GUI Scripting with Python needs PyWin32
<https://sourceforge.net/projects/pywin32/>.

Web Dynpro for ABAP

To test integration scenarios with Web Dynpro for ABAP you can use the Web Dynpro Applications from the following development packages:

- SWDP_DEMO
- SWDP_DEMO_TUTORIALS

SFLIGHT

Generate Data

It is possible to create or reset the data for the SAP SFLIGHT model via transaction code SE38 and the report SAPBC_DATA_GENERATOR. Or you can use the transaction code BC_DATA_GEN. The additional SFLIGHT_DATA_GEN report fills the database tables STICKET and SNVOICE.

Dataset

		Approximate Number of Entries		
		SPFLI	SFLIGHT	SBOO...
Delete Table Entries	<input type="radio"/>	0	0	0
Minimum Data Record	<input type="radio"/>	14	95	28,500
Standard Data Record	<input checked="" type="radio"/>	26	350	100,000
Maximum Data Record	<input type="radio"/>	46	1300	274,000
Monster Data Record	<input type="radio"/>	46	4900	1,300,000

Large data records can only be created in the background.

☐ Canceled Entries in SBOOK

The following tables will be filled with the report:

Number	Tablename	Description
1	SCARR	Airline
2	SCURX	Currency
3	SCURR	Exchange rates
4	SGEOCITY	Geographical position of a city
5	SAIRPORT	Airports
6	SCITAIRP	City-Airport assignment
7	SAPLANE	Plane
8	SCPLANE	Cargo plane
9	SCUSTOM	Flight customers
10	STRAVELAG	Travel agency
11	SBUSPART	Airline partner
12	SCOUNTER	Sales counter
13	SPFLI	Flight schedule
14	SFLCONNPOS	Stage of a flight connection
15	SFLCONN	Flight connection offered by travel agency

16	SCARPLAN	Plain-airline assignment
17	SMEAL	Inflight meal
18	SMEALT	Inflight meal description
19	SSTARTER	Inflight meal / Appetizer
20	SMACOURSE	Inflight meal / Main course
21	SDESSERT	Inflight meal / Dessert
22	SMENU	Menu
23	SBOOK	Single flight booking
24	SFLIGHT	Flight

Add Records

With the transaction code BC_GLOBAL_SFLGH_CREA, which calls the report SAPBC_GLOBAL_SFLIGHT_CREATE, it is possible to add additional records to the SFLIGHT table.

1. Select a flight and press Create.

Airline	LH
Connection Number	400
Flight Date	20.04.2020
<input type="button" value="Create"/>	

2. Add the necessary data and press the save button.

Airline	LH
Connection Number	400
Flight Date	20.04.2020
Airfare	888,00
Airline local currency	EUR
Plane Type	737-800

3. Now you can find the additional record in the table SFLIGHT.

<input type="checkbox"/>	500	LH	0400	28.03.2020	666,00	EUR	A340-600	330	310	258.174,
<input type="checkbox"/>	500	LH	0400	11.04.2020	666,00	EUR	A340-600	330	319	266.200,
<input type="checkbox"/>	500	LH	0400	20.04.2020	888,00	EUR	737-800	140	0	0,
<input type="checkbox"/>	500	LH	0400	28.04.2020	666,00	EUR	A340-600	330	321	270.489,
<input type="checkbox"/>	500	LH	0400	13.05.2020	666,00	EUR	A340-600	330	314	262.910,

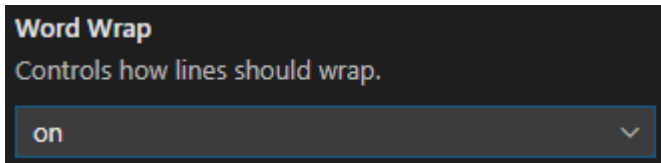
Visual Studio Code

[Visual Studio Code](#) is an open source code editor from Microsoft. It is possible to integrate VSCode in Scripting Tracker.
Here a few hints to configure VSCode.

To set a visible right margin add to settings.json:

```
"editor.rulers": [72]
```

To set the word wrap function change in Files > Preferences > Settings



Install [PowerShell Visual Studio extension](#).

To integrate PowerShell Core add to settings.json:

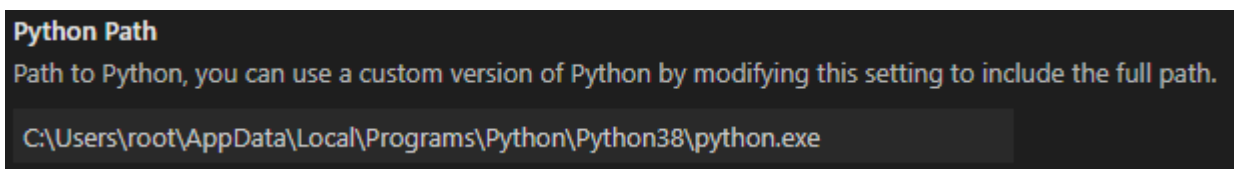
```
"powershell.powerShellAdditionalExePaths": [
  {
    "exePath" : "C:\\Users\\UserName\\Program
Files\\PowerShell\\pwsh.exe",
    "versionName": "PowerShell Core 7.0.0"
  }
]
```

Now you have the possibility to switch between different PowerShell versions with a click in the statusbar of VSCode.



Install [Python Visual Studio extension](#). and install [Python for Windows extension](#).

To set the path to Python interpreter change in Files > Preferences > Settings



This adds to settings.json:

```
"python.pythonPath":
"C:\\Users\\root\\AppData\\Local\\Programs\\Python\\Python38\\python.exe"
```


eCATT Export Scripts

```
*****
*
* Export of all or a selection eCATT scripts
*
*****

"-Begin-----
REPORT z_get_all_ecatt_scripts.

DATA:
  lt_line TYPE STANDARD TABLE OF ecscr_line,
  ls_line TYPE ecscr_line,
  ls_ec_line TYPE ecscr_line,
  lt_lines TYPE STANDARD TABLE OF string,
  lv_filename TYPE string,
  lt_ecscr_xml TYPE STANDARD TABLE OF ecscr_xml_str,
  lo_conv_in TYPE REF TO cl_abap_conv_in_ce,
  lv_str TYPE string,
  lv_path TYPE string.

FIELD-SYMBOLS:
  <ls_ecscr_xml> TYPE ecscr_xml_str.

SELECTION-SCREEN BEGIN OF BLOCK ui.
  SELECTION-SCREEN SKIP 1.
  SELECTION-SCREEN BEGIN OF LINE.
    SELECTION-SCREEN COMMENT 2(15) path FOR FIELD pa_path.
    PARAMETERS pa_path TYPE sapb-sappfad LOWER CASE.
  SELECTION-SCREEN END OF LINE.
  SELECTION-SCREEN SKIP 1.
  SELECTION-SCREEN BEGIN OF LINE.
    SELECTION-SCREEN COMMENT 2(15) scrname FOR FIELD pa_sname.
    PARAMETERS pa_sname(31) TYPE c.
  SELECTION-SCREEN END OF LINE.
  SELECTION-SCREEN SKIP 1.
  SELECTION-SCREEN BEGIN OF LINE.
    SELECTION-SCREEN COMMENT 2(37) xml FOR FIELD pa_xml.
    PARAMETERS pa_xml AS CHECKBOX.
  SELECTION-SCREEN END OF LINE.
SELECTION-SCREEN END OF BLOCK ui.

INITIALIZATION.
  path = 'Outputpath'.                "#EC NOTEXT
  pa_path = 'C:\Dummy\eCATT\'.        "#EC NOTEXT
  scrname = 'Scriptname'.              "#EC NOTEXT
  xml = 'Export SAP GUI Scripts as XML files'.    "#EC NOTEXT

AT SELECTION-SCREEN ON VALUE-REQUEST FOR pa_path.

  CALL METHOD cl_gui_frontend_services=>directory_browse
    EXPORTING
      window_title      = 'Outputpath'
    CHANGING
      selected_folder = lv_path
    EXCEPTIONS
      OTHERS           = 1.            "#EC NOTEXT
  IF sy-subrc = 0.
    pa_path = lv_path.
  ENDIF.
```

AT SELECTION-SCREEN.

```
SELECT *
  FROM ecscr_line
 INTO TABLE lt_line
 WHERE name LIKE pa_sname
 ORDER BY name version xml_lnr.
CHECK sy-subrc = 0.
```

```
LOOP AT lt_line INTO ls_line GROUP BY ( name = ls_line-name
  version = ls_line-version ).
```

```
CLEAR lt_lines.
```

```
LOOP AT GROUP ls_line INTO ls_ec_line.
  APPEND ls_ec_line-xml_line TO lt_lines.
ENDLOOP.
```

```
lv_filename = ls_line-name.
REPLACE ALL OCCURRENCES OF '/' IN lv_filename WITH '_'.
```

```
cl_gui_frontend_services=>gui_download(
  EXPORTING
    filename = pa_path && lv_filename && '_' &&
      ls_line-version && '.eCATT'
  CHANGING
    data_tab = lt_lines
  EXCEPTIONS
    OTHERS    = 1
).
IF sy-subrc <> 0.
ENDIF.
```

"#EC NOTEXT

```
IF pa_xml = 'X'. "-Save SAP GUI Scripting data in XML format-----
```

```
SELECT * FROM ecscr_xml_str INTO TABLE lt_ecscr_xml
 WHERE name = ls_line-name AND version = ls_line-version
 ORDER BY name version pname ptyp varid.
CHECK sy-subrc = 0.
```

```
LOOP AT lt_ecscr_xml ASSIGNING <ls_ecscr_xml>.
```

```
lo_conv_in = cl_abap_conv_in_ce=>create(
  input = <ls_ecscr_xml>-pxml_stream
).
lo_conv_in->read( IMPORTING data = lv_str ).
```

```
CHECK lv_str CS '<GuiScripting'.
"
```

"#EC NOTEXT

```
CLEAR lt_lines.
APPEND lv_str TO lt_lines.
```

```
cl_gui_frontend_services=>gui_download(
  EXPORTING
    filename = pa_path && lv_filename && '_' &&
      ls_line-version && '.' && <ls_ecscr_xml>-pname && '.xml'
  CHANGING
    data_tab = lt_lines
  EXCEPTIONS
    OTHERS    = 1
).
IF sy-subrc <> 0.
```

"#EC NOTEXT

```
ENDIF.
```

ENDLOOP.

ENDIF.

ENDLOOP.

"-End-----

Speed Comparison VBS and dotNET

VBScript

```
'-Begin-----  
  
'-Sub Main-----  
Sub Main()  
  
    Dim SapGuiAuto, app, connection, session  
    Dim StartTime, EndTime  
  
    On Error Resume Next  
    Set SapGuiAuto = GetObject("SAPGUI")  
    On Error GoTo 0  
    If Not IsObject(SapGuiAuto) Then  
        MsgBox "Can not get SapGuiAuto", vbOkOnly, "Hint"  
        Exit Sub  
    End If  
  
    On Error Resume Next  
    Set app = SapGuiAuto.GetScriptingEngine  
    On Error GoTo 0  
    If Not IsObject(app) Then  
        MsgBox "Can not get application", vbOkOnly, "Hint"  
        Exit Sub  
    End If  
  
    On Error Resume Next  
    Set connection = app.Children(0)  
    On Error GoTo 0  
    If Not IsObject(connection) Then  
        MsgBox "Can not get connection", vbOkOnly, "Hint"  
        Exit Sub  
    End If  
  
    If connection.DisabledByServer = True Then  
        MsgBox "Scripting is disabled by server", vbOkOnly, "Hint"  
        Exit Sub  
    End If  
  
    On Error Resume Next  
    Set session = connection.Children(0)  
    On Error GoTo 0  
    If Not IsObject(session) Then  
        MsgBox "Can not get session", vbOkOnly, "Hint"  
        Exit Sub  
    End If  
  
    If session.Info.IsLowSpeedConnection = True Then  
        MsgBox "Connection is low speed", vbOkOnly, "Hint"  
        Exit Sub  
    End If  
  
    StartTime = Timer()  
    For i = 1 To 10  
        session.findById("wnd[0]/tbar[0]/okcd").text = "/nSE16"  
        session.findById("wnd[0]").sendVKey 0  
        session.findById("wnd[0]/usr/ctxtDATABROWSE-TABLENAME").text = "TADIR"  
        session.findById("wnd[0]").sendVKey 0  
        session.findById("wnd[0]").sendVKey 31  
        session.findById("wnd[1]/tbar[0]/btn[0]").press
```

```

        session.findById("wnd[0]/tbar[0]/btn[3]").press
        session.findById("wnd[0]/tbar[0]/btn[3]").press
    Next
    EndTime = Timer()

    MsgBox CStr(FormatNumber(EndTime - StartTime, 2))

End Sub

'-Main-----
Main

'-End-----

```

VB.NET

```

'-Begin-----

Imports System.Windows.Forms
Imports System.Diagnostics

Public Module SAPGUIScripting

    Sub Main()

        Dim SapGuiAuto As Object
        Dim app As Object
        Dim connection As Object
        Dim session As Object

        Try
            SapGuiAuto = GetObject("SAPGUI")
            app = SapGuiAuto.GetScriptingEngine
            connection = app.Children(0)
            If connection.DisabledByServer = True Then
                Exit Sub
            End If
            session = connection.Children(0)
            If session.Info.IsLowSpeedConnection = True Then
                Exit Sub
            End If
        Catch
            Exit Sub
        End Try

        Dim watch As Stopwatch = Stopwatch.StartNew()
        For i = 1 To 10
            session.findById("wnd[0]/tbar[0]/okcd").text = "/nSE16"
            session.findById("wnd[0]").sendVKey(0)
            session.findById("wnd[0]/usr/ctxtDATABROWSE-TABLENAME").text = "TADIR"
            session.findById("wnd[0]").sendVKey(0)
            session.findById("wnd[0]").sendVKey(31)
            session.findById("wnd[1]/tbar[0]/btn[0]").press
            session.findById("wnd[0]/tbar[0]/btn[3]").press
            session.findById("wnd[0]/tbar[0]/btn[3]").press
        Next
        watch.Stop()
        MessageBox.Show(CStr(watch.Elapsed.TotalMilliseconds))

    End Sub

End Module

```

'-End-----

C#

```
//-Begin-----

using System;
using System.Reflection;
using System.Runtime.InteropServices;
using System.Diagnostics;
using Microsoft.VisualBasic;

public class SAPGUIScripting {

    static dynamic InvokeMethod(object obj, string methodName, object[]
methodParams = null) {
        return obj.GetType().InvokeMember(methodName, BindingFlags.InvokeMethod,
null, obj, methodParams);
    }

    static dynamic GetProperty(object obj, string propertyName, object[]
propertyParams = null) {
        return obj.GetType().InvokeMember(propertyName, BindingFlags.GetProperty,
null, obj, propertyParams);
    }

    static dynamic SetProperty(object obj, string propertyName, object[]
propertyParams = null) {
        return obj.GetType().InvokeMember(propertyName, BindingFlags.SetProperty,
null, obj, propertyParams);
    }

    static void FreeObject(object obj) {
        Marshal.ReleaseComObject(obj);
    }

    static void Main() {

        object session = null;
        try {
            object SapGuiAuto = Interaction.GetObject("SAPGUI");
            object app = InvokeMethod(SapGuiAuto, "GetScriptingEngine");
            object connection = GetProperty(app, "Children", new object[1]{0});
            session = GetProperty(connection, "Children", new object[1]{0});
        } catch {
            return;
        }
        dynamic ID = null;

        Stopwatch watch = Stopwatch.StartNew();
        for(var i = 1; i <= 10; i++) {
            ID = InvokeMethod(session, "findById", new
object[1]{"wnd[0]/tbar[0]/okcd"});
            SetProperty(ID, "text", new object[1]{"nSE16"});
            ID = InvokeMethod(session, "findById", new object[1]{"wnd[0]"});
            InvokeMethod(ID, "sendVKey", new object[1]{0});
            ID = InvokeMethod(session, "findById", new
object[1]{"wnd[0]/usr/ctxtDATABROWSE-TABLENAME"});
            SetProperty(ID, "text", new object[1]{"TADIR"});
            ID = InvokeMethod(session, "findById", new object[1]{"wnd[0]"});
            InvokeMethod(ID, "sendVKey", new object[1]{0});
        }
    }
}
```

```

        ID = InvokeMethod(session, "findById", new object[1]{"wnd[0]"});
        InvokeMethod(ID, "sendVKey", new object[1]{31});
        ID = InvokeMethod(session, "findById", new
object[1]{"wnd[1]/tbar[0]/btn[0]"});
        InvokeMethod(ID, "press");
        ID = InvokeMethod(session, "findById", new
object[1]{"wnd[0]/tbar[0]/btn[3]"});
        InvokeMethod(ID, "press");
        ID = InvokeMethod(session, "findById", new
object[1]{"wnd[0]/tbar[0]/btn[3]"});
        InvokeMethod(ID, "press");
    }
    watch.Stop();
    Console.WriteLine(watch.Elapsed.TotalMilliseconds.ToString());

}

//-End-----

```

Result

Time in seconds

Try	WSH	VB.NET	C#
1	7,05	7,04	7,52
2	6,89	6,92	7,46
3	6,92	6,96	7,55
4	7,06	6,85	7,58
5	6,95	6,90	7,74
6	6,84	6,82	7,60
7	7,07	6,83	7,65
8	7,13	6,69	7,36
9	6,98	6,85	7,37
10	6,93	7,02	7,33
Average	6,98	6,89	7,52
Percent	100%	99%	108%

HistoryEnabled

The HistoryEnabled property is set to false, to improve the performance of the SAP GUI.

<div>HistoryEnabled (Read-write)</div> <div><div>Public Property HistoryEnabled As Byte</div></div>	<div>The local history function can be enabled or disabled using this property. Disabling it will significantly improve the performance of SAP GUI, which may be crucial during load tests, for example.</div>
---	--

Unpack Tracker.zip

To unpack Tracker.zip use an external unpacker e.g. like 7-Zip. The internal Windows unpacker creates additional files with the extension Zone.Identifier of the unpacked files. These Zone.Identifier files contains information about the source of the file, it is an Alternate Data Stream (ADS). It describes the security zone associated with the file, which may be the Internet (3), local intranet (1), trusted site (2), restricted site (4) or local computer (0). Zone identifier files are generated automatically by Internet Explorer and other programs when files are downloaded to a Windows computer. You can see this files with the command `DIR /R`. The content looks e.g. like this:

```
[ZoneTransfer]
ZoneId=3
```

Hint: You can view the file with notepad e.g. like this:

```
notepad Tracker.exe:Zone.Identifier
```

When Tracker.zip is unpacked with an external unpacker, the file properties dialog no longer displays the security notice that the file came from another computer.

Attribute:	<input type="checkbox"/> Schreibgeschützt	Erweitert...
	<input type="checkbox"/> Versteckt	
Sicherheit:	Die Datei stammt von einem anderen Computer. Der Zugriff wurde aus Sicherheitsgründen eventuell blockiert.	
	Zulassen	

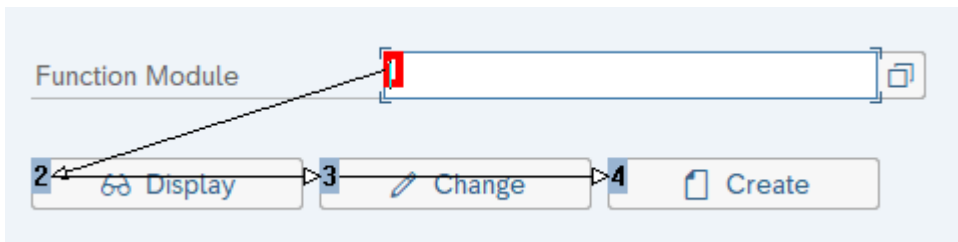
Local Tab Order Customizing

The SAP GUI for Windows allows to define for each screen a customizing of the tab order. If you press Ctrl and right mouse button you can find at the end of the context menu these items:

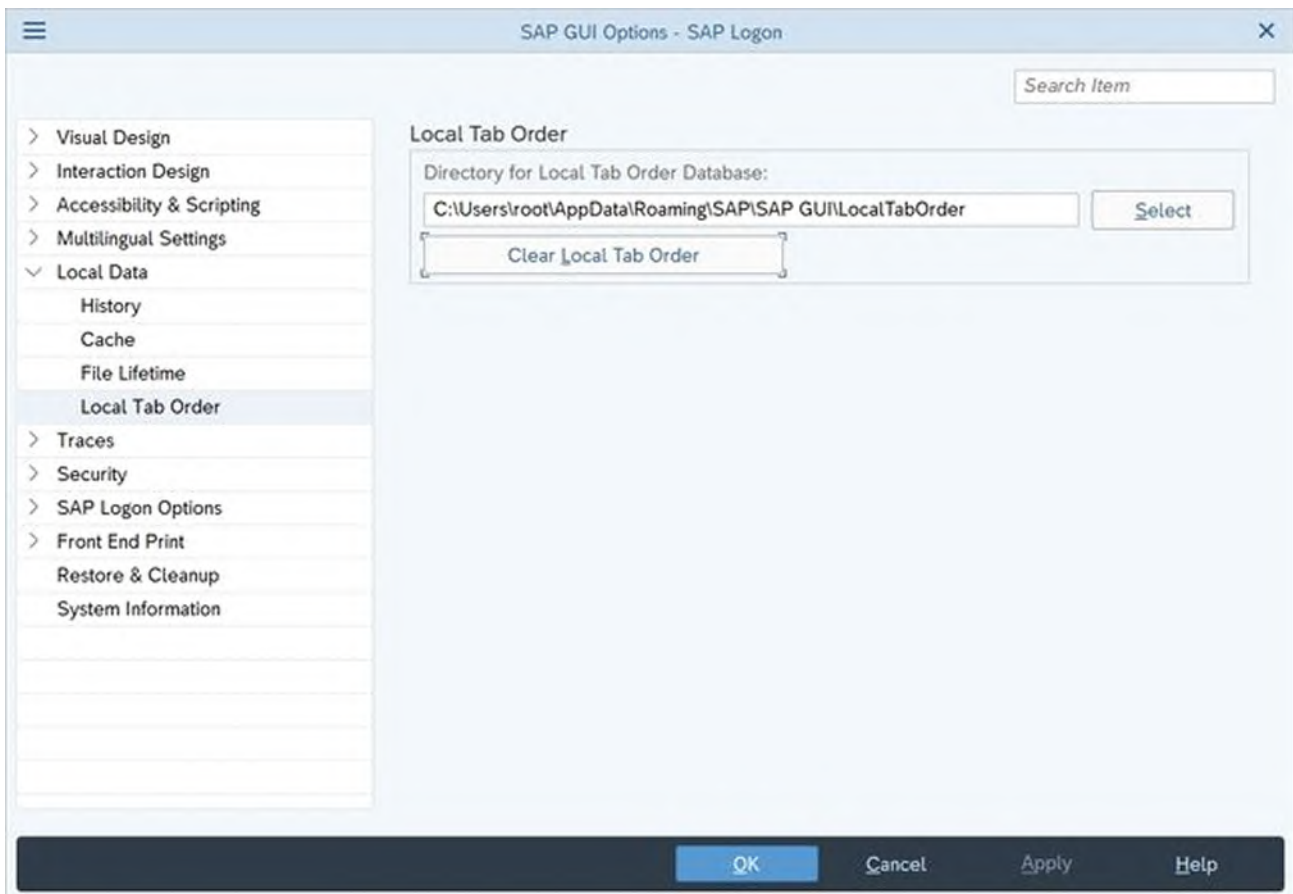
Configure Local Tab Order (Shift+Ctrl+L)

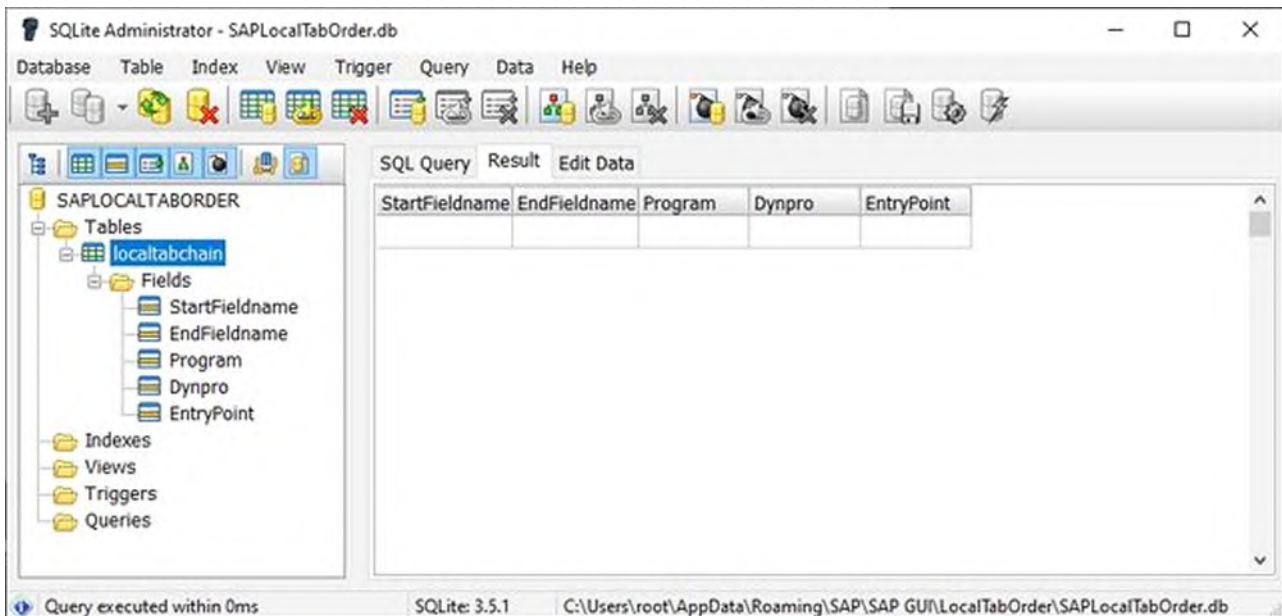
Visualize Local Tab Order (Shift+Ctrl+A to G)

With Visualize Local Tab Order you can see the sequence of fields if you press the tab key.

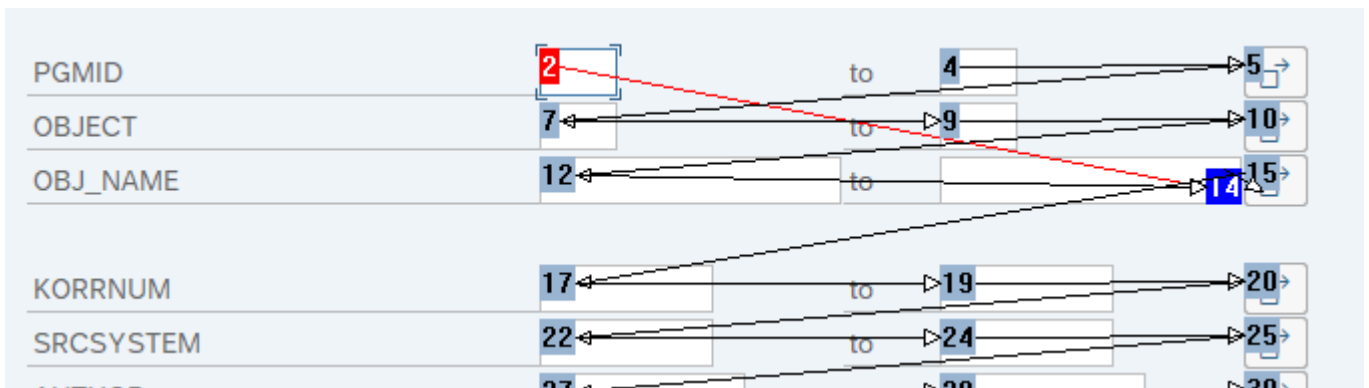


All sequences are stored in a SQLite database, which is defined in the SAP GUI Options:

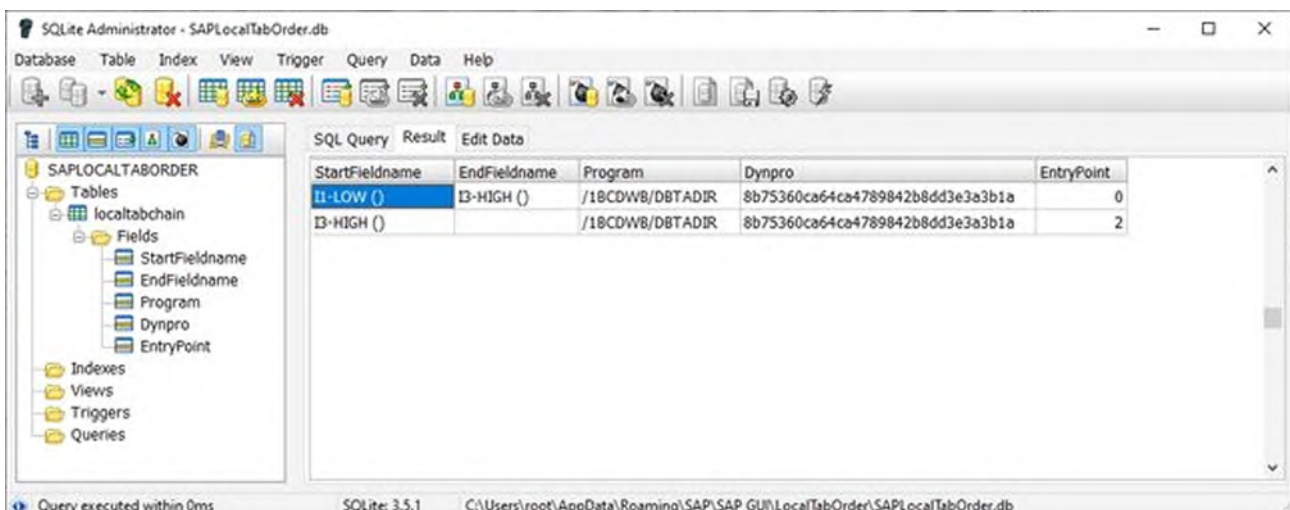




Each user has the possibility to define its own tab order for each screen, with the menu item Configure Local Tab Order. The following example jumps directly from field 2 to 14.



In the SQLite database you can find exact this as Start- and EndFieldname.



Also it is possible to define an exit point at field 14, this means the pressing of the tab key jumps to back button. Here a different perspective via the Configure Local Tab Order dialog.

Configure Local Tab Order Shift+Ctrl+L

X

Entry Point/Technical Name

Delete

Exit Point/Technical Name

to

I3-HIGH ()

Delete

From Element	From Element (Tech. Name)	To Element	To Element (Tech. Name)	
PGMID	I1-LOW ()	to	I3-HIGH ()	

Delete Local Tab Order Shortcut

Help

Close

Effects on Automation Workflows

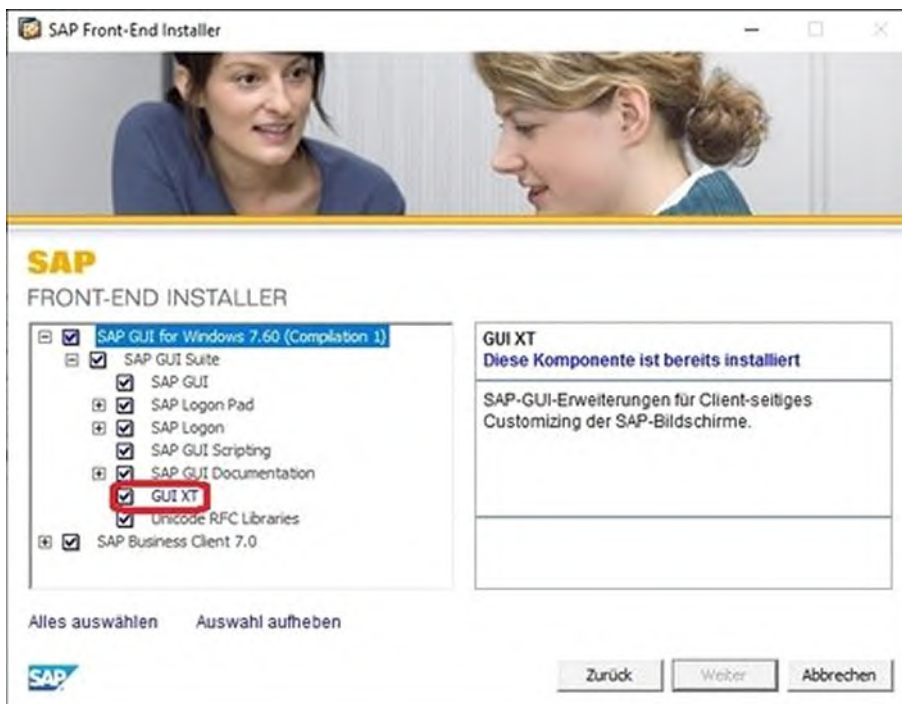
It is possible that your automation doesn't run correctly, when the tab key is used to switch to the fields and the user which executes the workflow has defined a local tab order. There are a lot of conditions. Perhaps there is a small chance that they will occur. But when it does occur, you can spend a lot of time to analyze it to find the reasons of this behavior.

How can we react to that?

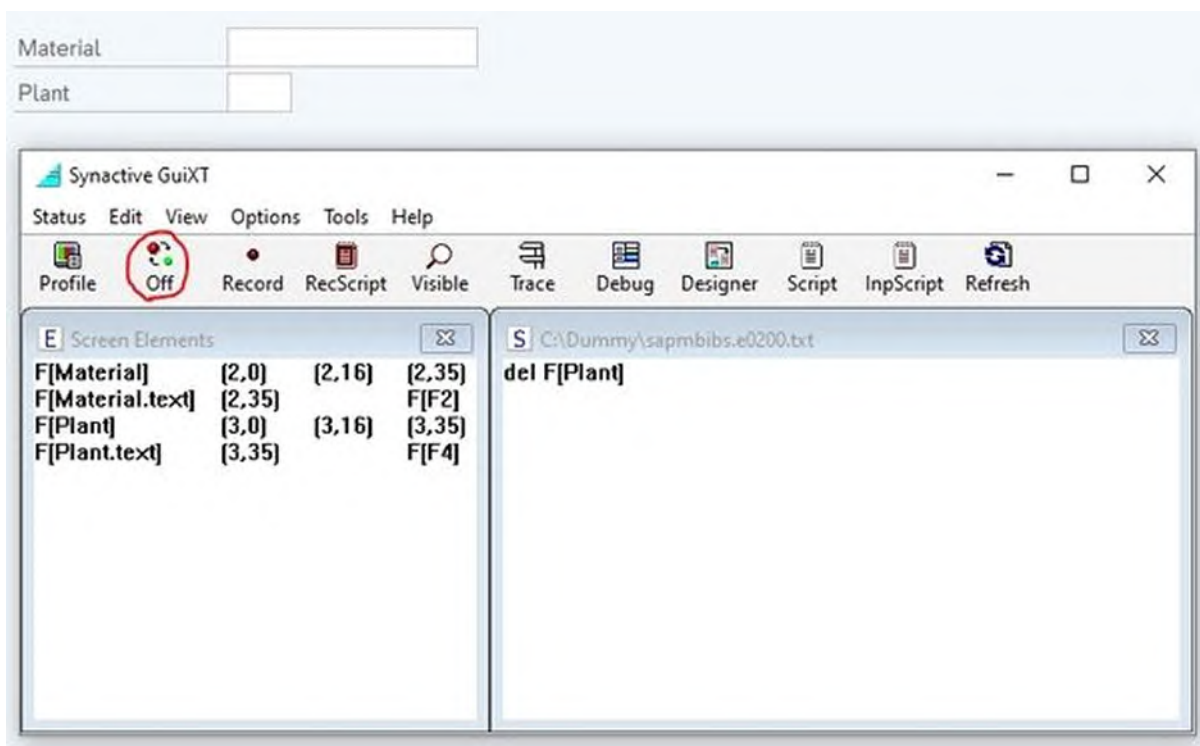
- Don't use the tab key to switch to the fields.
- Use only the IDs of the fields.
- It would also be possible to simply clear the database. But here we intervene in the sovereignty of the user settings. In my opinion not a good idea.

GuiXT

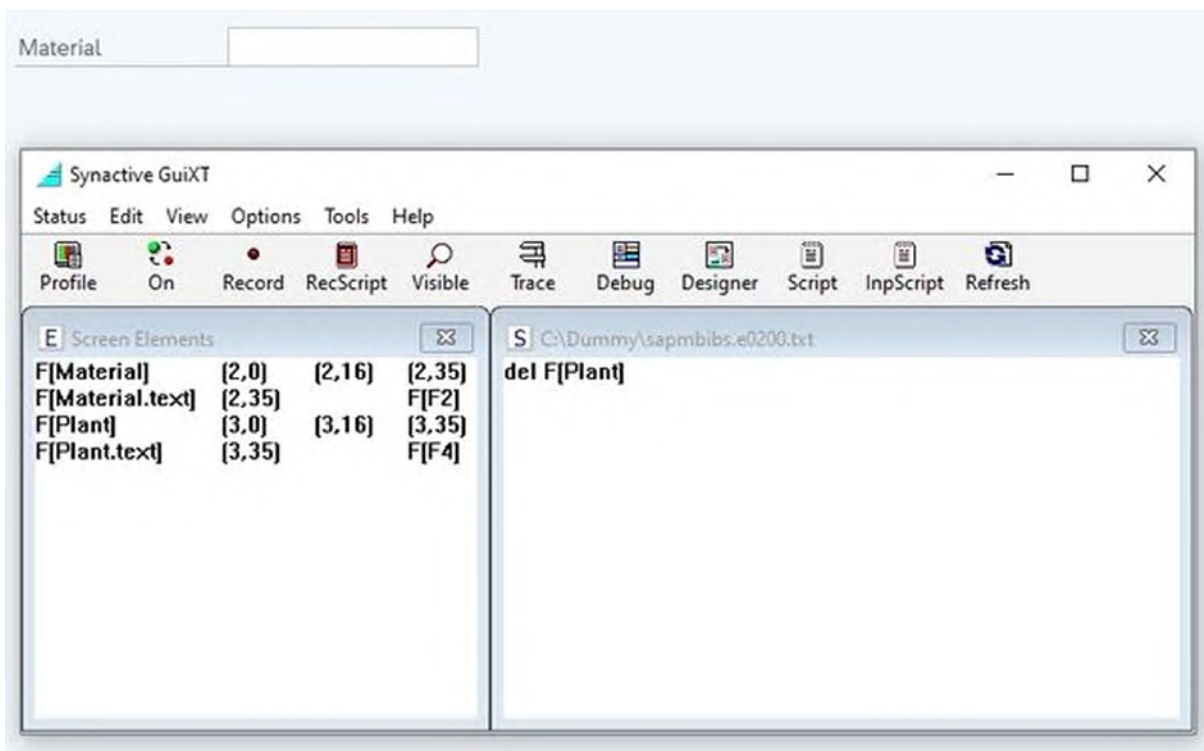
GuiXT is a very impressive SAP GUI for Windows extension. With GuiXT you have the possibility to change existing SAP GUI user screens as you like. A basis functionality is part of the installation of the SAP GUI for Windows.



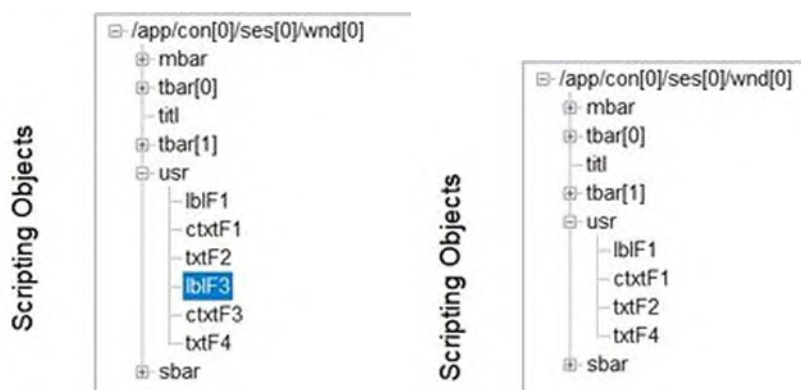
With GuiXT it is possible to change any user screen you like. Here an easy example with labels and fields. In the screen elements you see all UI elements which are available on the user screen. On the right side you see a tiny script, which deletes the Plant label and field, but GuiXT is off.



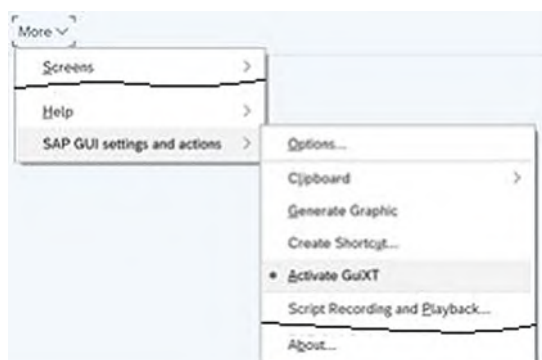
Now GuiXT is on and the screen is refreshed. The label and field Plant are no more longer available.



It is the same in the SAP GUI Scripting object hierarchy. On the left side the label and text F3 are available, but on the right side not, with activated GuiXT.



You can activate GuiXT very easily in the menu.



But what does that possibility mean for SAP automation?

There is a possibility, if you design an automation and the user which runs it uses GuiXT, that it "meets" screens, that it can't process. Fields can't be visible and other fields can exist. GuiXT offers fantastic possibilities here. When analyzing errors, it is always necessary to ask about the use of GuiXT. This can sometimes save you a lot of time and confusion.

Backslash

In a few programming languages it is necessary to escape the backslash \ in a string, e.g. in Python. This means it is necessary to add to a single backslash \ character the escape character \. On this way you double the backslash to \\. This prevents that the following character is recognized as a function character.

In SAP it is possible that IDs contain a backslash. Therefore it is necessary to add here an escape character.

An example: The following ID ...

```
wnd[0]/usr/tabsTAXI_TABSTRIP_HEAD/tabpT\13/ssubSUBSCREEN_BODY:SAPMV45A:4312/sub8309:SAPMV45A:8309/txtVBAK-ZZCON
```

... must be changed.

```
wnd[0]/usr/tabsTAXI_TABSTRIP_HEAD/tabpT\\13/ssubSUBSCREEN_BODY:SAPMV45A:4312/sub8309:SAPMV45A:8309/txtVBAK-ZZCON
```

Otherwise the error message "Control ID not found" occurs.

PowerShell - Array of Objects

In a few cases it is possible that PowerShell delivers, instead of one COM object, an array of COM objects. In this case it is not possible to get or set a single property or to call a single method.

The following example, which selects a row in a GuiTableControl and which was recorded, doesn't work:

```
$ID = Invoke-Method -object $ID -methodName "getAbsoluteRow" -methodParameter @ (14);  
$ID.selected = -1;
```

To set the selected property it is necessary to use the following code respectively the code must be adjusted manually:

```
$ID.getAbsoluteRow(14).selected = -1;
```

Because getAbsoluteRow delivers in PowerShell a collection of the columns in the row.

Restrictions

- Changing's in long texts with the full screen editor are not recorded, because no change event is fired from SAP GUI for Windows.
- In ALV-Grid the first position of the context menu is not recorded, because no change event is fired from SAP GUI for Windows.
- The text of a TextEdit control is not read because text lengths over 16702 characters cause a crash.
- Scripting Trackers recorder use the change event from SAP GUI Scripting. If no event is fired from SAP GUI for Windows, Scripting Tracker can't record the activities, equally the SAP standard.
- Scripting Tracker is an UTF8 version, it supports ANSI only with VBS files.
- Differences between SAP GUI for Windows and NetWeaver Business Client (NWBC)
 - The correct entry in the running object table (ROT) for the SAP GUI Scripting inside NWBC is SAPGUISEVER.
 - In the NWBC is no Toolbar[1] visible and not useable. Older SAP GUI Scripts will not work and there is no equivalent.
 - If a second NWBC window is open and the method `Connections` from the `Application` object with the property `Count` is used, it is not possible to detect more than one connection. SAP GUI Scripting uses only the first NWBC client window, because NWBC creates multiple instances of SAPGUISEVER in the running object table (ROT) and `GetObject` gets the first entry.
 - If a SAP® GUI Script is running in an NWBC window and the script is calling the method `GetScriptingEngine` a SAPGUI entry is registering in the ROT.

Momentary conclusion: SAP GUI Scripting in the context of NWBC offers not the same possibilities as in SAP GUI for Windows context now.

- Differences between SAP GUI for Windows and SAP GUI window in ABAP in Eclipse
 - The correct entry in the running object table (ROT) for the SAP GUI Scripting inside ABAP in Eclipse is SAPGUISEVER.
- Using SAP GUI for Windows, NWBC and ABAP in Eclipse with SAP GUI Scripting parallel at the same time

Momentary recommendation: Don't use SAP GUI for Windows, NWBC and/or ABAP in Eclipse with Scripting Tracker parallel and use only one instance of NWBC or ABAP in Eclipse at the same time with Scripting Tracker.

Frames

Collection of code snippets as frames to use SAP GUI Scripting easily.

[PowerShell](#)

[C#](#)

[VB.NET](#)

[WSH](#)

[VBA](#)

[Autolt](#)

[Python](#)

[JShell](#)

Frames - PowerShell

```
#-Begin-----  
  
#-Includes-----  
."$PSScriptRoot\COM.ps1"  
  
#-Sub Main-----  
Function Main {  
  
    $SapGuiAuto = Get-Object( , "SAPGUI")  
    If ($SapGuiAuto -isnot [System.__ComObject]) {  
        Return  
    }  
  
    $application = Invoke-Method $SapGuiAuto "GetScriptingEngine"  
    If ($application -isnot [System.__ComObject]) {  
        Free-Object -object $SapGuiAuto;  
        Return  
    }  
  
    Set-Property -object $application -propertyName "HistoryEnabled"  
-propertyParameter @($False);  
  
    $connection = Get-Property $application "Children" @(0)  
    If ($connection -eq $Null) {  
        Free-Object -object $application;  
        Free-Object -object $SapGuiAuto;  
        Return  
    }  
  
    $DisabledByServer = Get-Property $connection "DisabledByServer";  
    If ($True -eq $DisabledByServer) {  
        Return;  
    }  
  
    $session = Get-Property $connection "Children" @(0)  
    If ($session -eq $Null) {  
        Free-Object -object $connection;  
        Free-Object -object $application;  
        Free-Object -object $SapGuiAuto;  
        Return  
    }  
  
    $Busy = Get-Property $session "Busy";  
    If ($True -eq $Busy) {  
        Free-Object -object $session;  
        Free-Object -object $connection;  
        Free-Object -object $application;  
        Free-Object -object $SapGuiAuto;  
        Return;  
    }  
  
    $Info = Get-Property $session "Info";  
    $IsLowSpeedConnection = Get-Property $Info "IsLowSpeedConnection";  
    If ($True -eq $IsLowSpeedConnection) {  
        Free-Object -object $session;  
        Free-Object -object $connection;  
        Free-Object -object $application;  
        Free-Object -object $SapGuiAuto;  
        Return;  
    }  
}
```

```
Set-Property -object $application -propertyName "HistoryEnabled"  
-propertyValue @($True);
```

```
Free-Object -object $session;  
Free-Object -object $connection;  
Free-Object -object $application;  
Free-Object -object $SapGuiAuto;
```

```
}
```

```
#-Main-----  
Main
```

```
#-End-----
```

COM

```
#-Begin-----
```

```
#-Load assembly-----
```

```
If($PSVersionTable.PSVersion.Major -le 5) {  
    Add-Type -AssemblyName "Microsoft.VisualBasic";  
    Add-Type -AssemblyName "System.Windows.Forms";  
} ElseIf ($PSVersionTable.PSVersion.Major -ge 7) {  
    Add-Type -AssemblyName "System.Windows.Forms";  
}
```

```
#-Function Create-Object-----
```

```
Function Create-Object {
```

```
    Param(  
        [String]$objectName  
    )
```

```
    Try {  
        New-Object -ComObject $objectName;  
    } Catch {  
        If(($PSVersionTable.PSVersion.Major -le 5) -or `(  
            $PSVersionTable.PSVersion.Major -ge 7)) {  
            [Void][System.Windows.Forms.MessageBox]::Show(  
                "Can't create object", "Important hint", 0);  
            }  
        }  
    }
```

```
}
```

```
#-Function Get-Object-----
```

```
Function Get-Object {
```

```
    Param(  
        [String]$class  
    )
```

```
    If($PSVersionTable.PSVersion.Major -le 5) {  
        [Microsoft.VisualBasic.Interaction]::GetObject($class);  
    } ElseIf ($PSVersionTable.PSVersion.Major -ge 6) {  
        $SapROTWrr = New-Object -ComObject "SapROTWrr.SapROTWrrWrapper";  
        $SapROTWrr.GetROTErr($class);  
    }
```

```

}

#-Sub Free-Object-----
Function Free-Object {

    Param(
        [__ComObject]$Object
    )

    [Void][System.Runtime.InteropServices.Marshal]::ReleaseComObject($Object);
}

#-Function Get-Property-----
Function Get-Property {

    Param(
        [__ComObject]$Object,
        [String]$propertyName,
        $propertyParameter
    )

    $ObjectType = [System.Type]::GetType($Object);
    $ObjectType.InvokeMember($propertyName,
        [System.Reflection.BindingFlags]::GetProperty,
        $null, $Object, $propertyParameter);
}

#-Sub Set-Property-----
Function Set-Property {

    Param(
        [__ComObject]$Object,
        [String]$propertyName,
        $propertyValue
    )

    $ObjectType = [System.Type]::GetType($Object);
    [Void] $ObjectType.InvokeMember($propertyName,
        [System.Reflection.BindingFlags]::SetProperty,
        $null, $Object, $propertyValue);
}

#-Function Invoke-Method-----
Function Invoke-Method {

    Param(
        [__ComObject]$Object,
        [String]$methodName,
        $methodParameter
    )

    $ObjectType = [System.Type]::GetType($Object);
    $output = $ObjectType.InvokeMember($methodName,
        [System.Reflection.BindingFlags]::InvokeMethod,
        $null, $Object, $methodParameter);
    if ( $output ) { $output }
}

#-End-----

```

WebDriver

```
#-Begin-----

#-Includes-----
."$PSScriptRoot\COM.ps1"

#-Sub Main-----
Function Main {

    #-Set SapGuiAuto = GetObject("SAPGUI")-----
    $SapGuiAuto = Get-Object( , "SAPGUI")
    If ($SapGuiAuto -isnot [__ComObject]) {
        Return
    }

    #-Set application = SapGuiAuto.GetScriptingEngine-----
    $application = Invoke-Method $SapGuiAuto "GetScriptingEngine"
    If ($application -isnot [__ComObject]) {
        Free-Object $SapGuiAuto
        Return
    }

    #-Set connection = application.Children(0)-----
    $connection = Get-Property $application "Children" @(0)
    If ($connection -eq $Null) {
        Free-Object $SapGuiAuto
        Return
    }

    #-Set session = connection.Children(0)-----
    $session = Get-Property $connection "Children" @(0)
    If ($session -eq $Null) {
        Free-Object $SapGuiAuto
        Return
    }

    #-Your activities in the SAP GUI for Windows -----

    #-Load libraries-----
    Add-Type -Path "C:\Program
Files\Selenium\Selenium.WebDriverBackedSelenium.dll";
    Add-Type -Path "C:\Program Files\Selenium\WebDriver.dll";
    Add-Type -Path "C:\Program Files\Selenium\WebDriver.Support.dll";

    #-Set path to chrome browser-----
    $Options = New-Object OpenQA.Selenium.Chrome.ChromeOptions
    $Options.BinaryLocation = "C:/Program
Files/Google/Chrome/Application/chrome.exe"

    #-Opens a web browser window-----
    $WebDriver = New-Object OpenQA.Selenium.Chrome.ChromeDriver("C:\Program
Files\Selenium", $Options)
    $WebDriver.Url = "
http://nsp.stschnell.de:8630/sap/bc/webdynpro/sap/demo\_wd\_car\_rental"

    #-Your activities in the browser-----
```

```

$WebDriver.Close()
$WebDriver.Quit()

}

#-Main-----
Main

#-End-----

```

Parallel

```

#-Begin-----

#-Parameters-----
Param($SessionNo)

#-Includes-----
."$PSScriptRoot\COM.ps1"

#-Main-----
$SapGuiAuto = Get-Object( , "SAPGUI")
If ($SapGuiAuto -isnot [__ComObject]) {
    Exit
}

$application = Invoke-Method $SapGuiAuto "GetScriptingEngine"
If ($application -isnot [__ComObject]) {
    Free-Object $SapGuiAuto
    Exit
}

$connection = Get-Property $application "Children" @(0)
If ($connection -eq $Null) {
    Free-Object $SapGuiAuto
    Exit
}

$session = Get-Property $connection "Children" @(0)
If ($session -eq $Null) {
    Free-Object $SapGuiAuto
    Exit
}

#-Your Script here-----

Free-Object $SapGuiAuto

#-End-----

```

Here the script to execute it parallel:

```

#-Begin-----

start-job -Name job1 -FilePath C:\Dummy\test.ps1 -ArgumentList 0
start-job -Name job2 -FilePath C:\Dummy\test.ps1 -ArgumentList 1

wait-job -name job1

```

```
wait-job -name job2
```

```
receive-job -name job1
```

```
receive-job -name job2
```

```
#-End-----
```


Frames - C#

```
//-Begin-----  
  
using Microsoft.VisualBasic;  
using System.Reflection;  
using System.Runtime.InteropServices;  
  
public class SAPGUIScripting {  
  
    static dynamic InvokeMethod(object obj, string methodName, object[]  
methodParams = null) {  
        return obj.GetType().InvokeMember(methodName, BindingFlags.InvokeMethod,  
null, obj, methodParams);  
    }  
  
    static dynamic GetProperty(object obj, string propertyName, object[]  
propertyParams = null) {  
        return obj.GetType().InvokeMember(propertyName, BindingFlags.GetProperty,  
null, obj, propertyParams);  
    }  
  
    static dynamic SetProperty(object obj, string propertyName, object[]  
propertyParams = null) {  
        return obj.GetType().InvokeMember(propertyName, BindingFlags.SetProperty,  
null, obj, propertyParams);  
    }  
  
    static void FreeObject(object obj) {  
        Marshal.ReleaseComObject(obj);  
    }  
  
    static void Main() {  
  
        object SapGuiAuto = null;  
        object app = null;  
        object connection = null;  
        object session = null;  
  
        try {  
            SapGuiAuto = Interaction.GetObject("SAPGUI");  
            app = InvokeMethod(SapGuiAuto, "GetScriptingEngine");  
            SetProperty(app, "HistoryEnabled", new object[1]{false});  
            connection = GetProperty(app, "Children", new object[1]{0});  
            session = GetProperty(connection, "Children", new object[1]{0});  
            if(GetProperty(session, "Busy") == true) {  
                return;  
            }  
            object info = GetProperty(session, "Info");  
            if(GetProperty(info, "IsLowSpeedConnection") == true) {  
                return;  
            }  
        } catch {  
            return;  
        }  
  
        dynamic ID = null;  
  
  
        SetProperty(app, "HistoryEnabled", new object[1]{true});  
        FreeObject(session);  
    }  
}
```

```
FreeObject(connection);  
FreeObject(app);  
FreeObject(SapGuiAuto);  
  
}  
  
}  
  
// -End-----
```

To compile this code and execute it use the following batch file

```
c:\Windows\Microsoft.NET\Framework64\v4.0.30319\csc.exe /target:exe  
/platform:x64 /out:Tracker_RunScript.cs.exe  
/reference:"C:\Windows\Microsoft.NET\Framework64\v4.0.30319\Microsoft.VisualBasic.dll" Tracker_RunScript.cs  
start /min Tracker_RunScript.cs.exe
```

Frames - VB.NET

```
'-Begin-----  
  
Public Module SAPGUIScripting  
  
    Sub Main()  
  
        Dim SapGuiAuto As Object  
        Dim app As Object  
        Dim connection As Object  
        Dim session As Object  
  
        Try  
            SapGuiAuto = GetObject("SAPGUI")  
            app = SapGuiAuto.GetScriptingEngine  
            app.HistoryEnabled = False  
            connection = app.Children(0)  
            If connection.DisabledByServer = True Then  
                Exit Sub  
            End If  
            session = connection.Children(0)  
            If session.Busy = True Then  
                Exit Sub  
            End If  
            If session.Info.IsLowSpeedConnection = True Then  
                Exit Sub  
            End If  
        Catch  
            Exit Sub  
        End Try  
  
        app.HistoryEnabled = True  
  
    End Sub  
  
End Module  
  
'-End-----
```

To compile this code use the following batch file:

```
c:\Windows\Microsoft.NET\Framework64\v4.0.30319\vbc.exe /target:exe  
/platform:x64 /out:Tracker_RunScript.vb.exe Tracker_RunScript.vb  
start /min Tracker_RunScript.vb.exe
```

Frames - WSH

The Windows Script Host (WSH) is an automation technology from Microsoft and it is independent from the scripting language. Visual Basic Scripting, also known as VBScript, is an Active Scripting language which bases on the WSH. The language of VBScript is modeled on Visual Basic. VBScript is widely used and the SAP GUI Scripting Recorder, of the SAP GUI for Windows, uses VBScript as its standard language.

```
'-Begin-----  
  
'-Directives-----  
Option Explicit  
  
'-Sub Main-----  
Sub Main()  
  
    Dim SapGuiAuto, app, connection, session  
  
    Set SapGuiAuto = GetObject("SAPGUI")  
    If Not IsObject(SapGuiAuto) Then  
        Exit Sub  
    End If  
  
    Set app = SapGuiAuto.GetScriptingEngine  
    If Not IsObject(app) Then  
        Exit Sub  
    End If  
  
    app.HistoryEnabled = False  
  
    Set connection = app.Children(0)  
    If Not IsObject(connection) Then  
        Exit Sub  
    End If  
  
    If connection.DisabledByServer = True Then  
        Exit Sub  
    End If  
  
    Set session = connection.Children(0)  
    If Not IsObject(session) Then  
        Exit Sub  
    End If  
  
    If session.Busy = True Then  
        Exit Sub  
    End If  
  
    If session.Info.IsLowSpeedConnection = True Then  
        Exit Sub  
    End If  
  
    app.HistoryEnabled = True  
  
End Sub  
  
'-Main-----  
Main
```

```
'-End-----
```

Here an example [how to use VBScript with JScript](#).

Frames - WSH - JScript

JScript is a JavaScript dialect from Microsoft, based on the ECMAScript standard. It is implemented as an Active Scripting language in the Windows Script Host (WSH) and built on the COM/OLE automation platform.

```
//-Begin-----  
  
//Function hereString-----  
//  
// Simulate here-strings in JScript, like in PowerShell  
//  
//-----  
function hereString(f) {  
    return f.toString().replace(/^[^\\\/]+\//, '').replace(/\\*\\/[^\//]+$/, '');  
}  
  
//Visual Basic Script code to execute SAP GUI Scripting-----  
var VBSCode = hereString(function() {  
    Option Explicit  
  
    Sub SAPGUIScripting()  
        Set SapGuiAuto = GetObject("SAPGUI")  
        Set application = SapGuiAuto.GetScriptingEngine  
        Set connection = application.Children(0)  
        Set session = connection.Children(0)  
  
    End Sub  
  
    *});  
  
/*  
//Alternative it is possible to use a JavaScript string-----  
var VBSCode = '                                \n\  
Sub SAPGUIScripting()                        \n\  
    Set SapGuiAuto = GetObject("SAPGUI")      \n\  
    Set application = SapGuiAuto.GetScriptingEngine \n\  
    Set connection = application.Children(0)   \n\  
    Set session = connection.Children(0)      \n\  
End Sub';  
*/  
  
//Sub Main-----  
//  
// Hint: The MSScriptControl is only as x86 version available  
//  
//-----  
function Main() {  
    var MSScrCtrl = new ActiveXObject("MSScriptControl.ScriptControl");  
    MSScrCtrl.AllowUI = 1;  
    MSScrCtrl.Language = 'VBScript';  
    MSScrCtrl.AddCode(VBSCode);  
    MSScrCtrl.Run('SAPGUIScripting');  
}  
  
//Main-----  
Main();  
  
//End-----
```

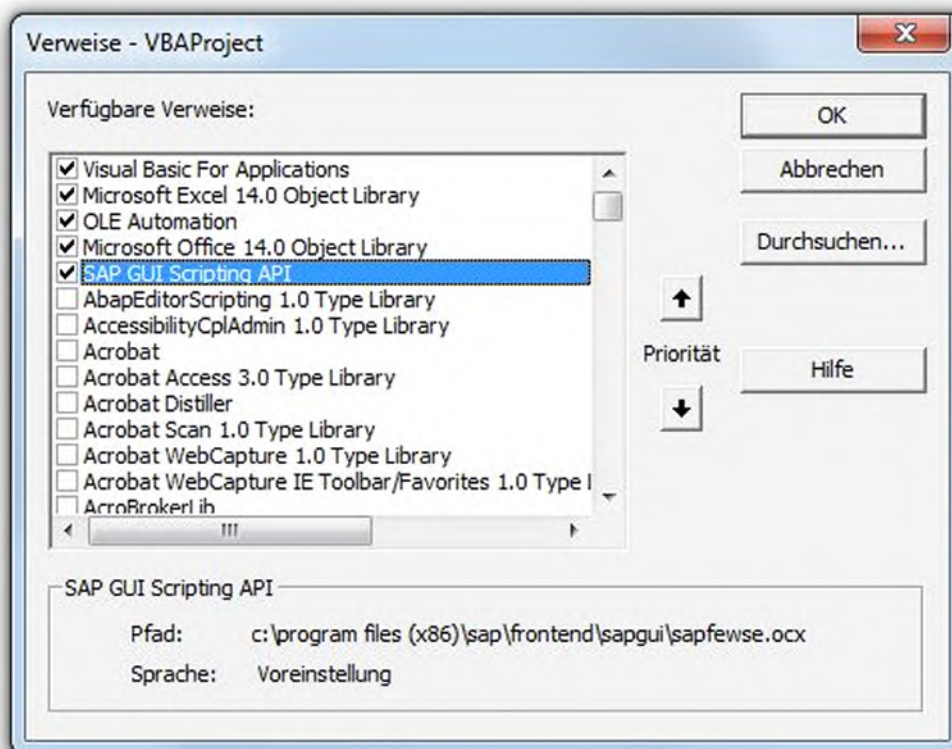
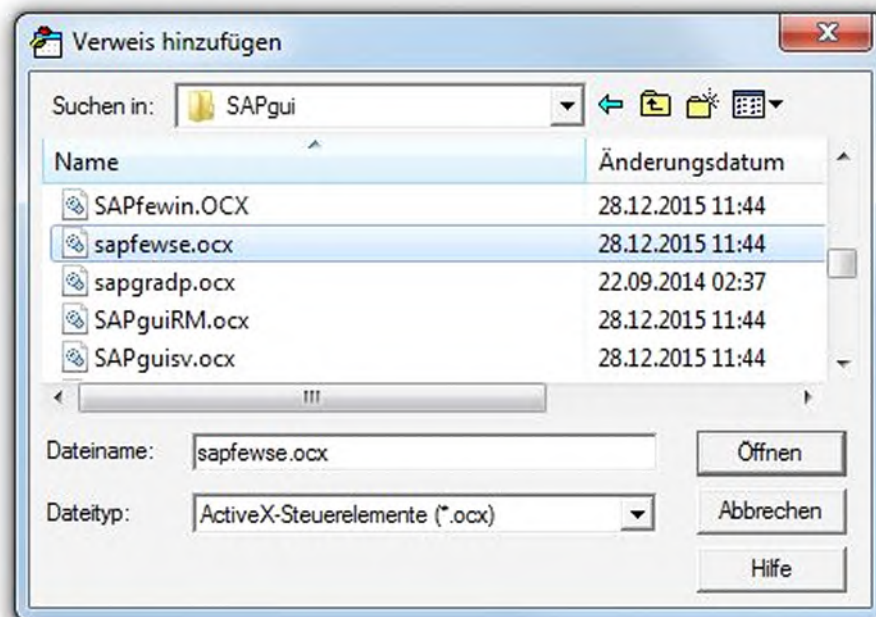
This example was created for the direct transfer of VBScript code, from the SAP GUI Scripting Recorder, into JScript, which is used by SAP Intelligent RPA.

Frames - VBA

```
'-Begin-----  
  
'-Directives-----  
Option Explicit  
  
'-Main-----  
Sub Main()  
  
    Dim SapGuiAuto As Object  
    Dim app As SAPFEWSELlib.GuiApplication  
    Dim connection As SAPFEWSELlib.GuiConnection  
    Dim session As SAPFEWSELlib.GuiSession  
  
    If app Is Nothing Then  
        Set SapGuiAuto = GetObject("SAPGUI")  
        Set app = SapGuiAuto.GetScriptingEngine  
    End If  
  
    If connection Is Nothing Then  
        Set connection = app.Children(0)  
    End If  
  
    If session Is Nothing Then  
        Set session = connection.Children(0)  
    End If  
  
End Sub  
  
'-End-----
```


Preparation for VBA

To use SAP GUI Scripting inside VBA you can reference to the ActiveX library. In this case the VBA-IDE supports you with code completion, of the methods and attributes, and with the library browser (F2).



Frames - Autolt

```
;--Begin-----  
  
;--Directives-----  
AutoItSetOption("MustDeclareVars", 1)  
  
;--Sub Main-----  
Func Main()  
  
    Local $SapGuiAuto, $application, $connection, $session  
  
    $SapGuiAuto = ObjGet("SAPGUI")  
    If Not IsObj($SapGuiAuto) Or @Error Then  
        Return  
    EndIf  
  
    $application = $SapGuiAuto.GetScriptingEngine()  
    If Not IsObj($application) Then  
        Return  
    EndIf  
  
    $application.HistoryEnabled = False  
  
    $connection = $application.Children(0)  
    If Not IsObj($connection) Then  
        Return  
    EndIf  
  
    If $connection.DisabledByServer = True Then  
        Return  
    EndIf  
  
    $session = $connection.Children(0)  
    If Not IsObj($session) Then  
        Return  
    EndIf  
  
    If $session.Busy = True Then  
        Return  
    EndIf  
  
    If $session.Info.IsLowSpeedConnection = True Then  
        Return  
    EndIf  
  
    $application.HistoryEnabled = True  
  
EndFunc  
  
;--Main-----  
Main()  
  
;--End-----
```

Frames - Python

```
#-Begin-----  
  
#-Includes-----  
import sys, win32com.client  
  
#-Sub Main-----  
def main():  
  
    try:  
  
        SapGuiAuto = win32com.client.GetObject("SAPGUI")  
        if not type(SapGuiAuto) == win32com.client.CDispatch:  
            return  
  
        application = SapGuiAuto.GetScriptingEngine  
        if not type(application) == win32com.client.CDispatch:  
            SapGuiAuto = None  
            return  
  
        application.HistoryEnabled = False  
  
        connection = application.Children(0)  
        if not type(connection) == win32com.client.CDispatch:  
            application = None  
            SapGuiAuto = None  
            return  
  
        if connection.DisabledByServer == True:  
            connection = None  
            application = None  
            SapGuiAuto = None  
            return  
  
        session = connection.Children(1)  
        if not type(session) == win32com.client.CDispatch:  
            connection = None  
            application = None  
            SapGuiAuto = None  
            return  
  
        if session.Busy == True:  
            session = None  
            connection = None  
            application = None  
            SapGuiAuto = None  
            return  
  
        if session.Info.IsLowSpeedConnection == True:  
            session = None  
            connection = None  
            application = None  
            SapGuiAuto = None  
            return  
  
    except:  
        print(sys.exc_info()[0])  
  
    finally:
```

```
application.HistoryEnabled = True
```

```
session = None
```

```
connection = None
```

```
application = None
```

```
SapGuiAuto = None
```

```
#-Main-----
```

```
if __name__ == "__main__":
```

```
    main()
```

```
#-End-----
```

Frames - JShell

```
//-Begin-----  
  
//-Classpath-----  
/env --class-path C:\Users\MyUser\Tracker\jacob\jacob.jar  
  
//-Import-----  
import com.jacob.activeX.*;  
import com.jacob.com.*;  
import javax.swing.JOptionPane;  
  
//-Sub Main-----  
void Main() {  
  
    ActiveXComponent SAPROTWr, application, connection, session, obj;  
    Dispatch ROTEntry;  
    Variant Value, ScriptEngine;  
    Variant[] arg;  
    boolean DisabledByServer, Busy, IsLowSpeedConnection;  
  
    ComThread.InitSTA();  
  
    application = null;  
  
    try {  
  
        // -Set SapGuiAuto = GetObject("SAPGUI")-----  
        SAPROTWr = new ActiveXComponent("SapROTWr.SapROTWrapper");  
        ROTEntry = SAPROTWr.invoke("GetROTEntry", "SAPGUI").toDispatch();  
  
        // -Set application = SapGuiAuto.GetScriptingEngine-----  
        ScriptEngine = Dispatch.call(ROTEntry, "GetScriptingEngine");  
        application = new ActiveXComponent(ScriptEngine.toDispatch());  
  
        application.setProperty("HistoryEnabled", false);  
  
        // -Set connection = application.Children(0)-----  
        connection = new ActiveXComponent(  
            application.invoke("Children", 0).toDispatch()  
        );  
  
        DisabledByServer =  
connection.getProperty("DisabledByServer").changeType(Variant.VariantBoolean).  
getBoolean();  
        if(DisabledByServer == true) {  
            System.out.println("Scripting is disabled by server");  
            ComThread.Release();  
            return;  
        }  
  
        // -Set session = connection.Children(0)-----  
        session = new ActiveXComponent(  
            connection.invoke("Children", 0).toDispatch()  
        );  
  
        Busy =  
session.getProperty("Busy").changeType(Variant.VariantBoolean).getBoolean();  
        if(Busy == true) {  
            System.out.println("Session is busy");  
            ComThread.Release();  
            return;  
        }  
    }  
}
```

```

    }

    IsLowSpeedConnection =
session.getPropertyAsComponent("Info").getProperty("IsLowSpeedConnection").cha
ngeType(Variant.VariantBoolean).getBoolean();
    if(IsLowSpeedConnection == true) {
        System.out.println("Connection is low speed");
        ComThread.Release();
        return;
    }

} catch (Exception e) {
    System.out.println(e.getMessage().toString());
} finally {
    application.setProperty("HistoryEnabled", true);
    ComThread.Release();
}

}

//--Main-----
Main();
/exit

//--End-----

```

Examples

Collection of additional examples.

[PowerShell](#)

[VBA](#)

[WSH](#)

[Autolt](#)

[C#](#)

Examples - PowerShell

[ComboBox](#)

[Excel](#)

[Generic Object Services \(GOS\)](#)

[OpenConnection](#)

[StatusBar](#)

[Read TableControl](#)

[Start SAP GUI](#)

[Start SAP GUI with Logon](#)

Examples - PowerShell - ComboBox

```
#-Begin-----

#-Includes-----
."$PSScriptRoot\COM.ps1";

#-Main-----
$SapGuiAuto = Get-Object -class "SAPGUI";
If ($SapGuiAuto -isnot [__ComObject]) {
    Exit;
}

$application = Invoke-Method -object $SapGuiAuto -methodName
"GetScriptingEngine";
If ($application -isnot [__ComObject]) {
    Free-Object -object $SapGuiAuto;
    Exit;
}

$connection = Get-Property -object $application -propertyName "Children"
-propertyParameter @(0);
If ($Null -eq $connection) {
    Free-Object -object $SapGuiAuto;
    Exit;
}

$session = Get-Property -object $connection -propertyName "Children"
-propertyParameter @(0);
If ($Null -eq $session) {
    Free-Object -object $SapGuiAuto;
    Exit;
}

#-Start TAC GUIBIBS-----
$ID = Invoke-Method -object $session -methodName "findById" -methodParameter
@("wnd[0]/tbar[0]/okcd");
Set-Property -object $ID -propertyName "text" -propertyValue @("/nGUIBIBS");
$ID = Invoke-Method -object $session -methodName "findById" -methodParameter
@("wnd[0]");
Invoke-Method -object $ID -methodName "sendVKey" -methodParameter @(0);

#-Goto Possible Entries-----
For($i = 1; $i -le 29; $i++) {
    $ID = Invoke-Method -object $session -methodName "findById" -methodParameter
@("wnd[0]/tbar[1]/btn[19]");
    Invoke-Method -object $ID -methodName "press";
}

$ID = Invoke-Method -object $session -methodName "findById" -methodParameter
@("wnd[0]/usr/cmbT005X-LAND");
$Entries = Get-Property -object $ID -propertyName "Entries";
If($PSVersionTable.PSVersion.Major -le 5) {
    $Count = $Entries.Count;
} Else {
    $Count = Get-Property -object $Entries -propertyName "Count";
}

For($i = 0; $i -lt $Count; $i++) {
    If($PSVersionTable.PSVersion.Major -le 5) {
        $Item = $Entries[$i];
    } Else {
```

```
$Item = Get-Property -object $Entries -propertyName "ElementAt"
-propertyParameter @($i);
}
$Pos = Get-Property -object $Item -propertyName "Pos";
$Key = Get-Property -object $Item -propertyName "Key";
$Value = Get-Property -object $Item -propertyName "Value";
Write-Host $Pos " " $Key " " $Value;
}

Free-Object -object $SapGuiAuto;

#-End-----
```

Examples - PowerShell - Excel

```
#-Begin-----
#-Function CreateExcel-----
Function CreateExcel {

    $Excel = New-Object -ComObject Excel.Application;
    $Excel.Visible = $True;
    $WorkBook = $Excel.Workbooks.Add();
    $WorkSheet = $Excel.ActiveSheet;
    Return $Excel, $WorkBook, $WorkSheet;

}

#-Function OpenExcel-----
Function OpenExcel {

    Param(
        [String]$FilePath,
        [String]$SheetName
    )

    $Excel = New-Object -ComObject Excel.Application;
    $Excel.Visible = $True;
    $WorkBook = $Excel.Workbooks.Open($FilePath);
    $WorkSheet = $WorkBook.Sheets($SheetName);
    Return $Excel, $WorkBook, $WorkSheet;

}

#-Sub Main-----
Function Main {

    $Excel, $WorkBook, $WorkSheet = OpenExcel -FilePath "C:\Dummy\Test.xlsx"
    -SheetName "Tabelle1";

    $LastCol =
$WorkSheet.UsedRange.Columns($WorkSheet.UsedRange.Columns.Count).Column;
    $LastRow = $WorkSheet.UsedRange.Rows($WorkSheet.UsedRange.Rows.Count).Row;

    $Range = $WorkSheet.Range($WorkSheet.Cells(1,1), $WorkSheet.Cells($LastRow,
$LastCol));

    For($i = 1; $i -le $LastRow; $i++) {
        For($j = 1; $j -le $LastCol; $j++) {
            Write-Host -NoNewline $Range.Cells($i, $j).Text;
        }
        Write-Host;
    }

    #$WorkBook.SaveAs("C:\Dummy\Test.xlsx");
    $Excel.Quit();

}

#-Main-----
Main;

#-End-----
```

Examples - PowerShell - Generic Object Services (GOS)

```
#-Begin-----  
#-  
#- Important hint: To use this example it is necessary to set the option  
#- "Show native Microsoft Windows dialogs" in the SAP Logon  
#-  
#-----
```

Installation

Scripting is installed

User Settings

- ☒ Enable scripting
- ☐ Notify when a script attaches to SAP GUI
- ☐ Notify when a script opens a connection
- ☒ Show native Microsoft Windows dialogs

```
#-Includes-----  
."$PSScriptRoot\COM.ps1"  
  
#-Includes AutoItX-----  
Add-Type -Path "$($PSScriptRoot)\AutoItX\AutoItX3.Assembly.dll";  
  
#-Sub Main-----  
Function Main() {  
  
    $SapGuiAuto = Get-Object "SAPGUI";  
    If ($SapGuiAuto -IsNot [System.__ComObject]) {  
        Return;  
    }  
  
    $Application = Invoke-Method $SapGuiAuto "GetScriptingEngine";  
    If ($Application -IsNot [System.__ComObject]) {  
        Return;  
    }  
  
    $Connection = Get-Property $Application "Children" @(0);  
    If ($Null -eq $Connection) {  
        Return;  
    }  
  
    $Session = Get-Property $Connection "Children" @(0);  
    If ($Null -eq $Session) {  
        Return;  
    }  
  
    $ID = Invoke-Method -object $session -methodName "findById" -methodParameter  
    @("wnd[0]/tbar[0]/okcd");  
    Set-Property -object $ID -propertyName "text" -propertyValue  
    @("/nsgostest");  
    $ID = Invoke-Method -object $session -methodName "findById" -methodParameter  
    @("wnd[0]");  
    Invoke-Method -object $ID -methodName "sendVKey" -methodParameter @(0);  
    $ID = Invoke-Method -object $session -methodName "findById" -methodParameter  
    @("wnd[0]/tbar[1]/btn[8]");  
}
```

```

Invoke-Method -object $ID -methodName "press";

$Path = "C:\Dummy";
$FilesToAttach = $Path + "\Files2Attach";
$Files = Get-ChildItem $FilesToAttach;

ForEach($File In $Files) {

    $ID = Invoke-Method -object $session -methodName "findById"
-methodParameter @("wnd[0]/titl/shellcont/shell")
    Invoke-Method -object $ID -methodName "pressContextButton"
-methodParameter @("%GOS_TOOLBOX")

    $Job = Start-Job -ArgumentList $File.FullName, $PSScriptRoot -ScriptBlock
{

    Param($FileName, $PSScriptRoot);

    Add-Type -Path "$($PSScriptRoot)\AutoItX\AutoItX3.Assembly.dll";
    If ([AutoIt.AutoItX]::WinWait("Import file") -ne 1) {
        Return;
    }

    [AutoIt.AutoItX]::WinActivate("Import file");

    #-These shortcuts are dependent from the OS language!-----
    [AutoIt.AutoItX]::Send("!n$FileName!f");

    If ([AutoIt.AutoItX]::WinWait("SAP GUI Security", "", 5) -ne 1) {
        Return;
    }
    If ([AutoIt.AutoItX]::WinExists("SAP GUI Security") -eq 1) {
        [AutoIt.AutoItX]::Send("!a");
    }

}

    $ID = Invoke-Method -object $session -methodName "findById"
-methodParameter @("wnd[0]/titl/shellcont/shell");
    Invoke-Method -object $ID -methodName "selectContextMenuItem"
-methodParameter @("%GOS_PCATTA_CREA");

    #-Now the native modal dialog box is open and served by the job-----

    $Job | Remove-Job

    $ID = Invoke-Method -object $Session -methodName "findById"
-methodParameter @("wnd[0]/sbar/pane[0]");
    $StatusBar = Get-Property -object $ID -propertyName "text";

    If($StatusBar -eq "Document created") {
        Write-Host $File.FullName "created successfully" -ForegroundColor Green;
    } Else {
        Write-Host $File.FullName "not created" -ForegroundColor Red;
    }

}

}

#-Main-----
Main

#-End-----

```

Examples - PowerShell - OpenConnection

```
#-Begin-----  
  
#-Includes-----  
."$PSScriptRoot\COM.ps1";  
  
#-Main-----  
$SapGuiAuto = Get-Object -class "SAPGUI";  
If ($SapGuiAuto -isnot [__ComObject]) {  
    Exit;  
}  
  
$application = Invoke-Method -object $SapGuiAuto -methodName  
"GetScriptingEngine";  
If ($application -isnot [__ComObject]) {  
    Free-Object -object $SapGuiAuto;  
    Exit;  
}  
  
$connection = Invoke-Method -object $application -methodName "OpenConnection"  
-methodParameter @("NSP")  
If ($Null -eq $connection) {  
    Free-Object -object $SapGuiAuto;  
    Exit;  
}  
  
$session = Get-Property -object $connection -propertyName "Children"  
-propertyParameter @(0);  
If ($Null -eq $session) {  
    Free-Object -object $SapGuiAuto;  
    Exit;  
}  
  
$ID = Invoke-Method -object $session -methodName "findById" -methodParameter  
@("wnd[0]/usr/txtRSYST-MANDT");  
Set-Property -object $ID -propertyName "text" -propertyValue @("001");  
$ID = Invoke-Method -object $session -methodName "findById" -methodParameter  
@("wnd[0]/usr/txtRSYST-BNAME");  
Set-Property -object $ID -propertyName "text" -propertyValue @("bcuser");  
$ID = Invoke-Method -object $session -methodName "findById" -methodParameter  
@("wnd[0]/usr/pwdRSYST-BCODE");  
Set-Property -object $ID -propertyName "text" -propertyValue @("minisap");  
$ID = Invoke-Method -object $session -methodName "findById" -methodParameter  
@("wnd[0]/usr/txtRSYST-LANGU");  
Set-Property -object $ID -propertyName "text" -propertyValue @("EN");  
$ID = Invoke-Method -object $session -methodName "findById" -methodParameter  
@("wnd[0]");  
Invoke-Method -object $ID -methodName "sendVKey" -methodParameter @(0);  
  
Free-Object -object $SapGuiAuto;  
  
#-End-----
```

Examples - PowerShell - StatusBar

```
#-Begin-----
#-Includes-----
."$PSScriptRoot\COM.ps1";

#-Main-----
$SapGuiAuto = Get-Object -class "SAPGUI";
If ($SapGuiAuto -isnot [__ComObject]) {
    Exit;
}

$application = Invoke-Method -object $SapGuiAuto -methodName
"GetScriptingEngine";
If ($application -isnot [__ComObject]) {
    Free-Object -object $SapGuiAuto;
    Exit;
}

$connection = Get-Property -object $application -propertyName "Children"
-propertyParameter @(0);
If ($Null -eq $connection) {
    Free-Object -object $SapGuiAuto;
    Exit;
}

$session = Get-Property -object $connection -propertyName "Children"
-propertyParameter @(0);
If ($Null -eq $session) {
    Free-Object -object $SapGuiAuto;
    Exit;
}

#-Start TAC GUIBIBS-----
$ID = Invoke-Method -object $session -methodName "findById" -methodParameter
@("wnd[0]/tbar[0]/okcd");
Set-Property -object $ID -propertyName "text" -propertyValue @("/nGUIBIBS");
$ID = Invoke-Method -object $session -methodName "findById" -methodParameter
@("wnd[0]");
Invoke-Method -object $ID -methodName "sendVKey" -methodParameter @(0);

#-Goto MessagesIn Primary Windows-----
For($i = 1; $i -le 39; $i++) {
    $ID = Invoke-Method -object $session -methodName "findById" -methodParameter
@("wnd[0]/tbar[1]/btn[19]");
    Invoke-Method -object $ID -methodName "press";
}

#-Button Success-----
$ID = Invoke-Method -object $session -methodName "findById" -methodParameter
@("wnd[0]/usr/btnPB1");
Invoke-Method -object $ID -methodName "press";
$ID = Invoke-Method -object $session -methodName "findById" -methodParameter
@("wnd[0]/sbar");
$StatusBarText = Get-Property -object $ID -propertyName "Text";
$MsgType = Get-Property -object $ID -propertyName "MessageType";
[Void][System.Windows.Forms.MessageBox]::Show($StatusBarText, $MsgType, 0);

#-Button Warning-----
$ID = Invoke-Method -object $session -methodName "findById" -methodParameter
@("wnd[0]/usr/btnPB2");
```

```

Invoke-Method -object $ID -methodName "press";
$ID = Invoke-Method -object $session -methodName "findById" -methodParameter
@("wnd[0]/sbar");
$StatusBarText = Get-Property -object $ID -propertyName "Text";
$MsgType = Get-Property -object $ID -propertyName "MessageType";
[Void][System.Windows.Forms.MessageBox]::Show($StatusBarText, $MsgType, 0);

#-Button Error-----
$ID = Invoke-Method -object $session -methodName "findById" -methodParameter
@("wnd[0]/usr/btnPB3");
Invoke-Method -object $ID -methodName "press";
$ID = Invoke-Method -object $session -methodName "findById" -methodParameter
@("wnd[0]/sbar");
$StatusBarText = Get-Property -object $ID -propertyName "Text";
$MsgType = Get-Property -object $ID -propertyName "MessageType";
[Void][System.Windows.Forms.MessageBox]::Show($StatusBarText, $MsgType, 0);

#-Button Status bar-----
$ID = Invoke-Method -object $session -methodName "findById" -methodParameter
@("wnd[0]/usr/btnPB5");
Invoke-Method -object $ID -methodName "press";
$ID = Invoke-Method -object $session -methodName "findById" -methodParameter
@("wnd[0]/sbar");
$StatusBarText = Get-Property -object $ID -propertyName "Text";
$MsgType = Get-Property -object $ID -propertyName "MessageType";
[Void][System.Windows.Forms.MessageBox]::Show($StatusBarText, $MsgType, 0);

Free-Object -object $SapGuiAuto;

#-End-----

```


Examples - PowerShell - Read TableControl

```
#-Begin-----
#-
#- TAC GUIBIBS
#-
#-----

#-Includes-----
."$PSScriptRoot\COM.ps1";

#-Main-----
$SapGuiAuto = Get-Object -class "SAPGUI";
If ($SapGuiAuto -isnot [__ComObject]) {
    Exit;
}

$application = Invoke-Method -object $SapGuiAuto -methodName
"GetScriptingEngine";
If ($application -isnot [__ComObject]) {
    Free-Object -object $SapGuiAuto;
    Exit;
}

$connection = Get-Property -object $application -propertyName "Children"
-propertyParameter @(0);
If ($Null -eq $connection) {
    Free-Object -object $SapGuiAuto;
    Exit;
}

$session = Get-Property -object $connection -propertyName "Children"
-propertyParameter @(0);
If ($Null -eq $session) {
    Free-Object -object $SapGuiAuto;
    Exit;
}

#-Start TAC GUIBIBS-----
$ID = Invoke-Method -object $session -methodName "findById" -methodParameter
@("wnd[0]/tbar[0]/okcd");
Set-Property -object $ID -propertyName "text" -propertyValue @("/nGUIBIBS");

#-Go to overview screen-----
$wnd0 = Invoke-Method -object $session -methodName "findById" -methodParameter
@("wnd[0]");
Invoke-Method -object $wnd0 -methodName "sendVKey" -methodParameter @(0);
Invoke-Method -object $wnd0 -methodName "sendVKey" -methodParameter @(19);
Invoke-Method -object $wnd0 -methodName "sendVKey" -methodParameter @(19);
Invoke-Method -object $wnd0 -methodName "sendVKey" -methodParameter @(19);
Invoke-Method -object $wnd0 -methodName "sendVKey" -methodParameter @(19);

#-Read GuiTableControl-----
$Table = Invoke-Method -object $session -methodName "findById"
-methodParameter @("wnd[0]/usr/tblSAPMBIBSTC535");
$vScrollBar = Get-Property -object $Table -propertyName "VerticalScrollbar";
$RowCount = Get-Property -object $Table -propertyName "RowCount";
$Columns = Get-Property -object $Table -propertyName "Columns";

If($PSVersionTable.PSVersion.Major -le 5) {
    $ColCount = $Columns.Count;
} Else {
```

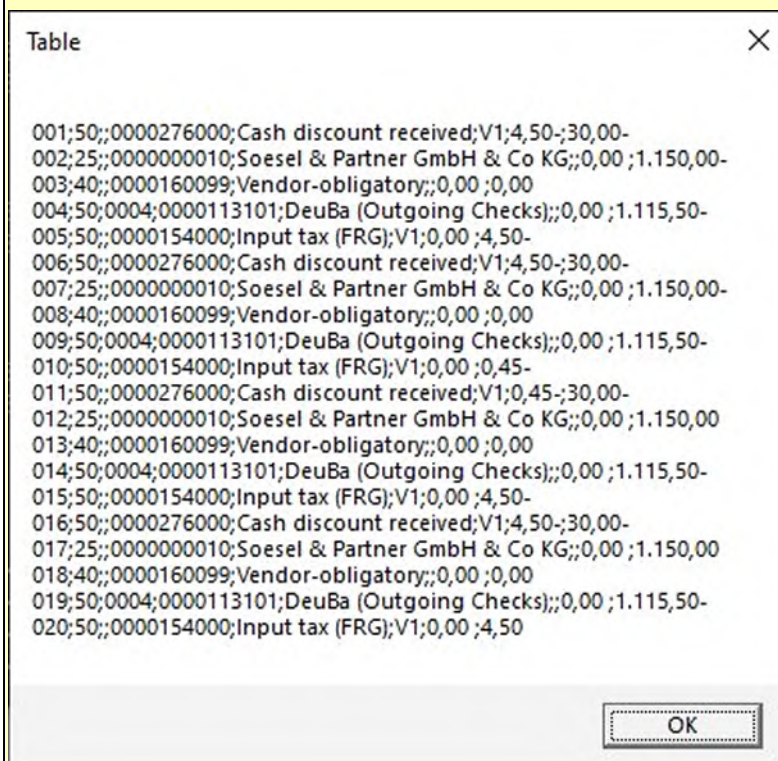
```

$ColCount = Get-Property -object $Columns -propertyName "Count";
}

For($Row = 0; $Row -lt $RowCount; $Row++) {
    Set-Property -object $vScrollBar -propertyName "position" -propertyValue
    @($Row)
    $Table = Invoke-Method -object $session -methodName "findById"
    -methodParameter @("wnd[0]/usr/tblSAPMBIBSTC535");
    $vScrollBar = Get-Property -object $Table -propertyName "VerticalScrollbar";
    For($Col = 0; $Col -lt $ColCount; $Col++) {
        $Cell = Invoke-Method -object $Table -methodName "getCell"
        -methodParameter @(0, $Col);
        $CellText = Get-Property -object $Cell -propertyName "Text";
        If($Col -lt $ColCount - 1) {
            $Out += $CellText + ";";
        } Else {
            $Out += $CellText + "`n" ;
        }
    }
    $vScrollBarPosition = Get-Property -object $vScrollBar -propertyName
    "position";
    $vScrollBarMaximum = Get-Property -object $vScrollBar -propertyName
    "Maximum";
    If($vScrollBarPosition -eq $vScrollBarMaximum) {
        Break;
    }
}

If($PSVersionTable.PSVersion.Major -ne 6) {
    #-Output in a message box-----
    [Void][System.Windows.Forms.MessageBox]::Show($Out, "Table", 0);
}

```



```

#-Output in a grid view-----
$OutGrid = $Out | ConvertFrom-Csv -Delimiter ";" -Header
"Pos", "UV", "BA", "Account", "Description", "VA", "Tax", "Amount";
$OutGrid | Out-GridView;

```

SOutGrid Out-GridView;							
Filter							
+ Add criteria							
Pos	UV	BA	Account	Description	VA	Tax	Amount
001	50		0000276000	Cash discount received	V1	4,50-	30,00-
002	25		0000000010	Soesel & Partner GmbH & Co KG		0,00	1.150,00-
003	40		0000160099	Vendor-obligatory		0,00	0,00
004	50	0004	0000113101	DeuBa (Outgoing Checks)		0,00	1.115,50-
005	50		0000154000	Input tax (FRG)	V1	0,00	4,50-
006	50		0000276000	Cash discount received	V1	4,50-	30,00-
007	25		0000000010	Soesel & Partner GmbH & Co KG		0,00	1.150,00-
008	40		0000160099	Vendor-obligatory		0,00	0,00
009	50	0004	0000113101	DeuBa (Outgoing Checks)		0,00	1.115,50-
010	50		0000154000	Input tax (FRG)	V1	0,00	0,45-
011	50		0000276000	Cash discount received	V1	0,45-	30,00-
012	25		0000000010	Soesel & Partner GmbH & Co KG		0,00	1.150,00
013	40		0000160099	Vendor-obligatory		0,00	0,00
014	50	0004	0000113101	DeuBa (Outgoing Checks)		0,00	1.115,50-
015	50		0000154000	Input tax (FRG)	V1	0,00	4,50-
016	50		0000276000	Cash discount received	V1	4,50-	30,00-
017	25		0000000010	Soesel & Partner GmbH & Co KG		0,00	1.150,00
018	40		0000160099	Vendor-obligatory		0,00	0,00
019	50	0004	0000113101	DeuBa (Outgoing Checks)		0,00	1.115,50-
020	50		0000154000	Input tax (FRG)	V1	0,00	4,50

```
} Else {
    Write-Output $Out;
}
```

```
Free-Object -object $SapGuiAuto;
```

```
#-End-----
```

Examples - Start SAP GUI

```
#-Begin-----

$Sig = @'
[DllImport("user32.dll", CharSet = CharSet.Auto)]
public static extern IntPtr FindWindow(string lpClassName, string
lpWindowName);
'@

#-Add FindWindow function-----
$Win32 = Add-Type -Namespace Win32 -Name Funcs -MemberDefinition $Sig
-PassThru;

#-Set the path to the SAP GUI directory-----
$SAPGUIPath = "C:\Program Files (x86)\SAP\FrontEnd\SAPgui\";

#-Set the SAP system ID or the IP address-----
$SID = "NSP";

#-Set the instance number of the SAP system-----
$instanceNo = "00";

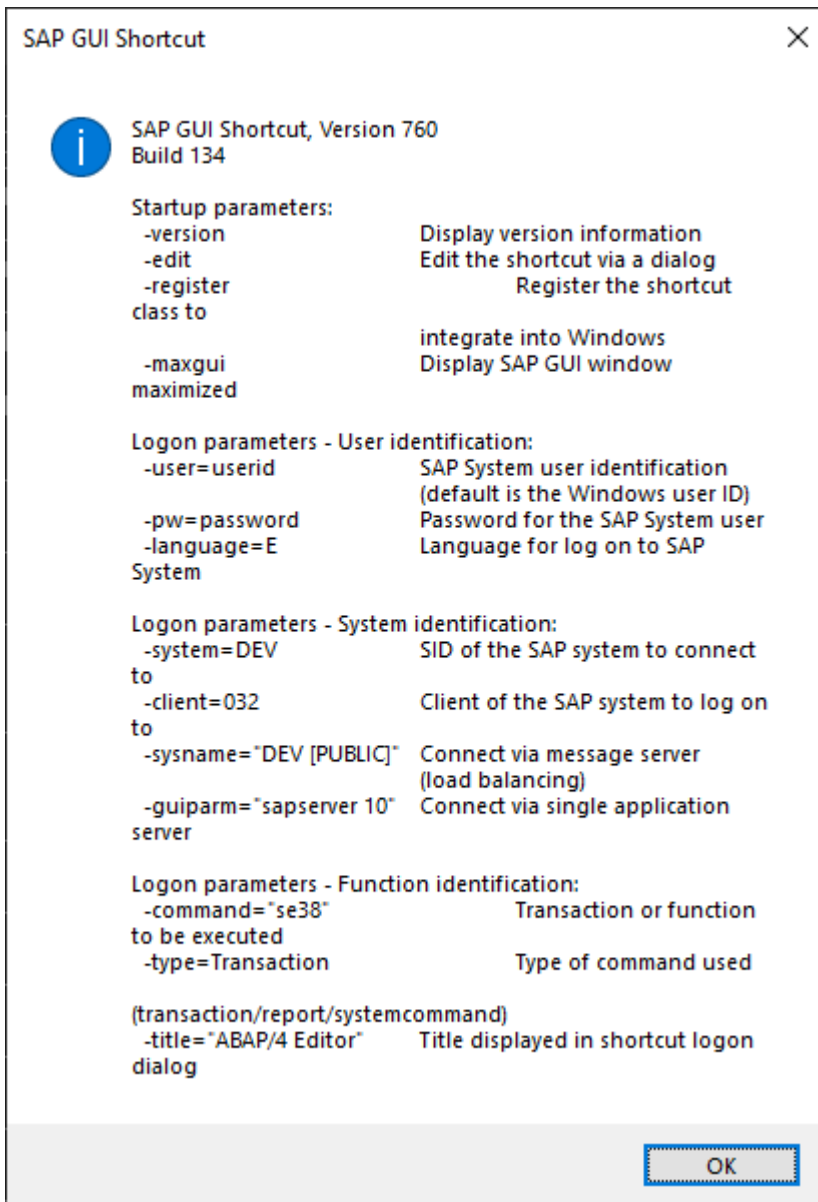
#-Starts the SAP GUI-----
$SAPGUI = $SAPGUIPath + "sapgui.exe";
& $SAPGUI $SID $InstanceNo;

While ($Win32::FindWindow("SAP_FRONTEND_SESSION", "SAP") -eq 0) {
    Start-Sleep -Milliseconds 250;
}
Write-Host "Here now your script...";

#-End-----
```

Examples - Start SAP GUI with Logon

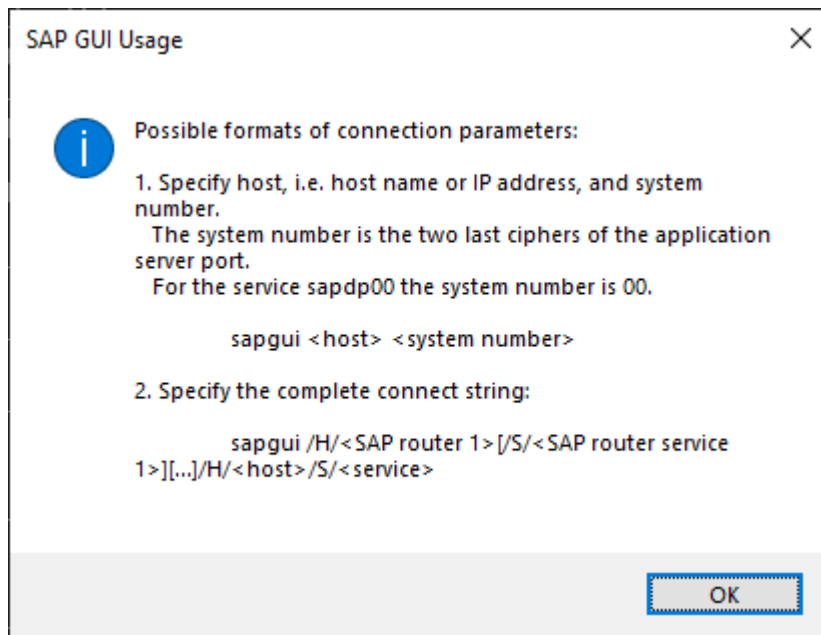
Variant 1 - Via SAP GUI Shortcut



```
#-Begin-----  
  
$Sig = @'  
[DllImport("user32.dll", CharSet = CharSet.Auto)]  
public static extern IntPtr FindWindow(string lpClassName, string  
lpWindowName);  
'@  
  
$Win32 = Add-Type -Namespace Win32 -Name Funcs -MemberDefinition $Sig  
-PassThru;  
  
& 'C:\Program Files (x86)\SAP\FrontEnd\SAPgui\sapshcut.exe' -system=NSP ` -  
-client=001 -user=bcuser -PW=minisap -language=E -maxgui -command=SE16;  
  
While ($Win32::FindWindow("SAP_FRONTEND_SESSION", "Data Browser: Initial  
Screen") -eq 0) {  
    Start-Sleep -Milliseconds 250;  
}
```

```
#-End-----
```

Variant 2 - Via SAP GUI



```
#-Begin-----
#-Includes-----
."COM.ps1"

#-Signatures-----
$Sig = @'
[DllImport("user32.dll", CharSet = CharSet.Auto)]
public static extern IntPtr FindWindow(string lpClassName, string
lpWindowName);
'@

#-Add FindWindow function-----
$Win32 = Add-Type -Namespace Win32 -Name Funcs -MemberDefinition $Sig
-PassThru;

#-Set the path to the SAP GUI directory-----
$SAPGUIPath = "C:\Program Files (x86)\SAP\FrontEnd\SAPgui\";

#-Set the SAP system ID-----
$SID = "localhost";

#-Set the instance number of the SAP system-----
$instanceNo = "00";

#-Start the SAP GUI-----
$SAPGUI = $SAPGUIPath + "sapgui.exe";
& $SAPGUI $SID $InstanceNo;

#-Wait until the session is available-----
While ($Win32::FindWindow("SAP_FRONTEND_SESSION", "SAP") -eq 0) {
    Start-Sleep -Milliseconds 250;
}

#-Logon to SAP GUI session-----
$SapGuiAuto = [Microsoft.VisualBasic.Interaction]::GetObject("SAPGUI");
$application = Invoke-Method $SapGuiAuto "GetScriptingEngine";
$connection = Get-Property $application "Children" @(0);
```

```
$session = Get-Property $connection "Children" @(0);

$ID = Invoke-Method $session "findById" @("wnd[0]/usr/txtRSYST-MANDT");
Set-Property $ID "Text" @("001");
$ID = Invoke-Method $session "findById" @("wnd[0]/usr/txtRSYST-BNAME");
Set-Property $ID "Text" @("BCUSER");
$ID = Invoke-Method $session "findById" @("wnd[0]/usr/pwdRSYST-BCODE");
Set-Property $ID "Text" @("minisap");
$ID = Invoke-Method $session "findById" @("wnd[0]/usr/txtRSYST-LANGU");
Set-Property $ID "Text" @("EN");
$ID = Invoke-Method $session "findById" @("wnd[0]");
Invoke-Method $ID "sendVKey" @(0);

Write-Host "Here now your script...";

#-End-----
```

Examples - C#

[Start Multiple TACs](#)

Examples - C# - Start Multiple TACs

```
//-Begin-----  
  
using System;  
using System.Reflection;  
using System.Runtime.InteropServices;  
using System.Threading;  
using Microsoft.VisualBasic;  
  
namespace ThreadingDemo {  
  
    class Program {  
  
        //-COM Interface-----  
        static dynamic InvokeMethod(object obj, string methodName, object[]  
methodParams = null) {  
            return obj.GetType().InvokeMember(methodName, BindingFlags.InvokeMethod,  
null, obj, methodParams);  
        }  
  
        static dynamic GetProperty(object obj, string propertyName, object[]  
propertyParams = null) {  
            return obj.GetType().InvokeMember(propertyName, BindingFlags.GetProperty,  
null, obj, propertyParams);  
        }  
  
        static dynamic SetProperty(object obj, string propertyName, object[]  
propertyParams = null) {  
            return obj.GetType().InvokeMember(propertyName, BindingFlags.SetProperty,  
null, obj, propertyParams);  
        }  
  
        static void FreeObject(object obj) {  
            Marshal.ReleaseComObject(obj);  
        }  
  
        //-Main-----  
        static void Main(string[] args) {  
  
            Console.WriteLine("Main Thread Started");  
  
            Thread SE37 = new Thread(MethodSE37) {  
                Name = "ThreadSE37"  
            };  
            Thread SE16 = new Thread(MethodSE16) {  
                Name = "ThreadSE16"  
            };  
  
            SE37.Start();  
            SE16.Start();  
  
            Console.WriteLine("Main Thread Ended");  
  
        }  
  
        //-SE37-----  
        static void MethodSE37() {  
  
            object SapGuiAuto = null;  
            object app = null;  
            object connection = null;
```

```

object session = null;

try {
    SapGuiAuto = Interaction.GetObject("SAPGUI");
    app = InvokeMethod(SapGuiAuto, "GetScriptingEngine");
    SetProperty(app, "HistoryEnabled", new object[1]{false});
    connection = GetProperty(app, "Children", new object[1]{0});
    session = GetProperty(connection, "Children", new object[1]{1});
} catch {
    return;
}

dynamic ID = null;

for (int i = 1; i <= 5; i++) {
    ID = InvokeMethod(session, "findById", new
object[1]{"wnd[0]/tbar[0]/okcd"});
    SetProperty(ID, "text", new object[1]{"nse37"});
    ID = InvokeMethod(session, "findById", new object[1]{"wnd[0]"});
    InvokeMethod(ID, "sendVKey", new object[1]{0});
    ID = InvokeMethod(session, "findById", new
object[1]{"wnd[0]/usr/ctxtRS38L-NAME"});
    SetProperty(ID, "text", new object[1]{"RFC_READ_TABLE"});
    ID = InvokeMethod(session, "findById", new object[1]{"wnd[0]"});
    InvokeMethod(ID, "sendVKey", new object[1]{7});
    ID = InvokeMethod(session, "findById", new
object[1]{"wnd[0]/usr/tabsFUNC_TAB_STRIP/tabpHEADER"});
    InvokeMethod(ID, "select");
    ID = InvokeMethod(session, "findById", new
object[1]{"wnd[0]/usr/tabsFUNC_TAB_STRIP/tabpIMPORT"});
    InvokeMethod(ID, "select");
    ID = InvokeMethod(session, "findById", new
object[1]{"wnd[0]/usr/tabsFUNC_TAB_STRIP/tabpEXPORT"});
    InvokeMethod(ID, "select");
    ID = InvokeMethod(session, "findById", new
object[1]{"wnd[0]/usr/tabsFUNC_TAB_STRIP/tabpCHANGE"});
    InvokeMethod(ID, "select");
    ID = InvokeMethod(session, "findById", new
object[1]{"wnd[0]/usr/tabsFUNC_TAB_STRIP/tabpTABLES"});
    InvokeMethod(ID, "select");
    ID = InvokeMethod(session, "findById", new
object[1]{"wnd[0]/usr/tabsFUNC_TAB_STRIP/tabpEXCEPT"});
    InvokeMethod(ID, "select");
    ID = InvokeMethod(session, "findById", new
object[1]{"wnd[0]/usr/tabsFUNC_TAB_STRIP/tabpSOURCE"});
    InvokeMethod(ID, "select");
    ID = InvokeMethod(session, "findById", new object[1]{"wnd[0]"});
    InvokeMethod(ID, "sendVKey", new object[1]{3});
    ID = InvokeMethod(session, "findById", new object[1]{"wnd[0]"});
    InvokeMethod(ID, "sendVKey", new object[1]{3});
}

SetProperty(app, "HistoryEnabled", new object[1]{true});
FreeObject(session);

FreeObject(connection);
FreeObject(app);
FreeObject(SapGuiAuto);
}

//SE16-----
static void MethodSE16() {

    object SapGuiAuto = null;

```

```

object app = null;
object connection = null;
object session = null;

try {
    SapGuiAuto = Interaction.GetObject("SAPGUI");
    app = InvokeMethod(SapGuiAuto, "GetScriptingEngine");
    SetProperty(app, "HistoryEnabled", new object[1]{false});
    connection = GetProperty(app, "Children", new object[1]{0});
    session = GetProperty(connection, "Children", new object[1]{2});
} catch {
    return;
}

dynamic ID = null;

for (int i = 1; i <= 10; i++) {
    ID = InvokeMethod(session, "findById", new
object[1]{"wnd[0]/tbar[0]/okcd"});
    SetProperty(ID, "text", new object[1]{"nse16"});
    ID = InvokeMethod(session, "findById", new object[1]{"wnd[0]"});
    InvokeMethod(ID, "sendVKey", new object[1]{0});
    ID = InvokeMethod(session, "findById", new
object[1]{"wnd[0]/usr/ctxtDATABROWSE-TABLENAME"});
    SetProperty(ID, "text", new object[1]{"TADIR"});
    ID = InvokeMethod(session, "findById", new
object[1]{"wnd[0]/usr/ctxtDATABROWSE-TABLENAME"});
    SetProperty(ID, "caretPosition", new object[1]{5});
    ID = InvokeMethod(session, "findById", new object[1]{"wnd[0]"});
    InvokeMethod(ID, "sendVKey", new object[1]{7});
    ID = InvokeMethod(session, "findById", new object[1]{"wnd[0]"});
    InvokeMethod(ID, "sendVKey", new object[1]{31});
    ID = InvokeMethod(session, "findById", new object[1]{"wnd[1]"});
    InvokeMethod(ID, "sendVKey", new object[1]{0});
    ID = InvokeMethod(session, "findById", new object[1]{"wnd[0]"});
    InvokeMethod(ID, "sendVKey", new object[1]{3});
    ID = InvokeMethod(session, "findById", new object[1]{"wnd[0]"});
    InvokeMethod(ID, "sendVKey", new object[1]{3});
}

SetProperty(app, "HistoryEnabled", new object[1]{true});

FreeObject(session);
FreeObject(connection);
FreeObject(app);
FreeObject(SapGuiAuto);

}

}

}

// -End-----

```

Examples - WSH (Windows Script Host)

[GetConnectionSessionNumber](#)

[FindSAPWindowByHandle](#)

[FindSAPWindowBySIDTAC](#)

[FindByTypeName](#)

[FindByText](#)

[FindByIdPart](#)

[Session](#)

[Splitter](#)

[Sum All Column Numbers](#)

[Table](#)

[Tree](#)

[Start Multiple TACs \(1\)](#)

[Start Multiple TACs \(2\)](#)

[HTMLViewer](#)

Examples - WSH - GetConnectionSessionNumber

```
'-Begin-----
'_
'- Example how to get connection and session number from session ID
'_
'-----

pos = InStr(session.Id(), "con[") + 4
len = InStr(pos, session.Id(), "]" ) - pos
connectionNumber = CLng(Mid(session.Id(), pos, Len))

pos = InStr(session.Id(), "ses[") + 4
len = InStr(pos, session.Id(), "]" ) - pos
sessionNumber = CLng(Mid(session.Id(), pos, Len))

MsgBox "ConnectionNumber: " & CStr(connectionNumber) & _
      " SessionNumber: " & CStr(sessionNumber)

'-End-----
```

Examples - WSH - FindSAPWindowByHandle

```
'-Function FindSAPWindowByHandle-----
'-
'- Function to find an SAP window by its handle
'- hSAPWnd = Handle of the SAP window
'-
'-----
Function FindSAPWindowByHandle(hSAPWnd)

    Dim SapGuiAuto, app, CollCon, oCon, CollSes, oSes, hWnd, i, j

    Set SapGuiAuto = GetObject("SAPGUI")
    If Not IsObject(SapGuiAuto) Then
        Exit Function
    End If

    Set app = SapGuiAuto.GetScriptingEngine
    If Not IsObject(app) Then
        Exit Function
    End If

    '-Get all connections-----
    Set CollCon = app.Connections()
    If Not IsObject(CollCon) Then
        Exit Function
    End If

    '-Loop over connections-----
    For i = 0 To CollCon.Count() - 1

        Set oCon = app.Children(CLng(i))
        If Not IsObject(oCon) Then
            Exit Function
        End If

        '-Get all sessions of a connection-----
        Set CollSes = oCon.Sessions()
        If Not IsObject(CollSes) Then
            Exit Function
        End If

        '-Loop over sessions-----
        For j = 0 To CollSes.Count() - 1

            Set oSes = oCon.Children(CLng(j))
            If Not IsObject(oSes) Then
                Exit Function
            End If

            If oSes.Busy() = vbFalse Then

                hWnd = oSes.findById("wnd[0]").Handle

                If hSAPWnd = hWnd Then
                    FindSAPWindowByHandle = oSes.ID
                End If

            End If

        Next

    Next
```

Next

End Function

Examples - WSH - FindSAPWindowBySIDTAC

```
'-Function FindSAPWindowBySIDTAC-----
Function FindSAPWindowBySIDTAC(SID, TAC)

    Dim SapAppl, SapGuiAuto, CollCon, i, oCon, CollSes, j
    Dim oSes, oSesInf

    Set SapGuiAuto = GetObject("SAPGUI")
    If Not IsObject(SapGuiAuto) Then
        Exit Sub
    End If

    Set SapAppl = SapGuiAuto.GetScriptingEngine
    If Not IsObject(SapAppl) Then
        Exit Sub
    End If

    Set CollCon = SapAppl.Connections()
    If Not IsObject(CollCon) Then
        Exit Sub
    End If

    '-Loop over connections-----
    For i = 0 To CollCon.Count() - 1

        Set oCon = SapAppl.Children(CLng(i))
        If Not IsObject(oCon) Then
            Exit Sub
        End If

        Set CollSes = oCon.Sessions()
        If Not IsObject(CollSes) Then
            Exit Sub
        End If

        '-Loop over sessions-----
        For j = 0 To CollSes.Count() - 1

            Set oSes = oCon.Children(CLng(j))
            If Not IsObject(oSes) Then
                Exit Sub
            End If

            If oSes.Busy() = vbFalse Then

                Set oSesInf = oSes.Info()
                If IsObject(oSesInf) Then
                    '-----
                    '-
                    '- With the session info object is it possible to
                    '- select a specific session which executes the
                    '- activities. In our example it is the system name
                    '- and the transaction code, but you can use all
                    '- properties of the session info object you want.
                    '-
                    '-----

                    If SID = oSesInf.SystemName() And TAC = oSesInf.Transaction() Then

                        FindSAPWindowBySIDTAC = oSes.ID
                    End If
                End If
            End If
        Next j
    Next i
End Function
```



```
        End If
    End If
End If
Next
Next
End Function
```

Examples - WSH - FindByTypeName

```
'-Function FindByTypeName-----
'-
'- Function to find an UI element by its type and name, independently
'- from program names and screen numbers
'- oApp      = SAP application
'- oArea     = Container to be searched
'- strType   = Type of UI element which is searched
'- strName   = Full or part of a name from UI element which is searched
'-
'-----
Function FindByTypeName(oApp, oArea, strType, strName)

    For i = 0 To oArea.Children().Count() - 1

        Set Obj = oArea.Children(CInt(i))
        If Obj.Type = strType And InStr(Obj.Name, strName) Then
            'MsgBox Obj.Name & " " & Obj.Type & " " & Obj.Text
            Set FindByTypeName = Obj
            Exit Function
        End If

        If Obj.ContainerType() = True Then
            Set ObjChildren = Obj.Children()
            If ObjChildren.Count() > 0 Then
                Set NextArea = oApp.findByID(Obj.ID)
                Set FindByTypeName = FindByTypeName(oApp, NextArea, strType, strName)
                If Not FindByTypeName Is Nothing Then
                    Exit Function
                End If
                Set NextArea = Nothing
            End If
            Set ObjChildren = Nothing
        End If

        Set Obj = Nothing

    Next

    Set FindByTypeName = Nothing

End Function
```

Examples - WSH - FindByText

```
'-Function FindByText-----
'-
'- Function to find an UI element by its text, independently from
'- program names and screen numbers, and delivers the ID
'- oArea    = Container to be searched
'- strText  = Text of UI element which is searched
'-
'- Example call:
'- Id = FindByText(session.findById("wnd[0]/usr"), "monitoring")
'-
'-----
Function FindByText(oArea, strText)

    On Error Resume Next
    cntObj = oArea.Children.Count()

    If cntObj > 0 Then
        For i = 0 To cntObj - 1
            Set Child = oArea.Children.Item(CLng(i))
            If InStr(UCase(Child.Text), UCase(strText)) Then
                Id = Child.Id
                Exit For
            End If
        Next
    End If

    On Error Goto 0
    FindByText = Id

End Function
```

Examples - WSH - FindByIdPart

```
'-Begin-----

'-Function FindByIdPart-----
'_
'- Function to find an UI element by its Id via Regular Expressions,
'- independently from program names and screen numbers
'- oApp = SAP application
'- oArea = Container to be searched
'- regexId = Regular Expression of Id of UI element which is searched
'_
'-----
Function FindByIdPart(oApp, oArea, regexId)

    Set oRegEx = New RegExp
    oRegEx.Pattern = regexId
    oRegEx.IgnoreCase = True
    oRegEx.Global = False

    On Error Resume Next
    If oArea.Children.Count() > 0 Then
        For i = 0 To oArea.Children.Count() - 1
            Set Child = oArea.Children.Item(CLng(i))
            If oRegEx.Test(Child.Id) Then
                FindByIdPart = Child.Id
                On Error GoTo 0
                Exit Function
            End If
            If Child.ContainerType() And Child.Children().Count() > 0 Then
                FindByIdPart = _
                    FindByIdPart(oApp, oApp.findByID(Child.Id), regexId)
                If FindByIdPart <> "" Then
                    On Error GoTo 0
                    Exit Function
                End If
            End If
        Next
    End If
    On Error Goto 0

    FindByIdPart = ""

End Function

'-Sub Main-----
Sub Main()

    Set SapGuiAuto = GetObject("SAPGUI")
    If Not IsObject(SapGuiAuto) Then
        Exit Sub
    End If

    Set app = SapGuiAuto.GetScriptingEngine
    If Not IsObject(app) Then
        Exit Sub
    End If

    app.HistoryEnabled = False
    Set connection = app.Children(0)
    If Not IsObject(connection) Then
        Exit Sub
    End If
End Sub
```

```

End If

If connection.DisabledByServer = True Then
    Exit Sub
End If

Set session = connection.Children(1)
If Not IsObject(session) Then
    Exit Sub
End If

If session.Info.IsLowSpeedConnection = True Then
    Exit Sub
End If

'Search for
'wnd[0]/usr/subSSA1:SAPLBRF_MAINTENANCE:3006/txtSBRF170-VERSION
Id = FindByIDPart(app, session.findById("wnd[0]/usr"), _
    "wnd\[0\]\usr\subSSA1:.*txtSBRF170-VERSION")

MsgBox Id

app.HistoryEnabled = True

End Sub

'-Main-----
Main

'-End-----

```

Examples - WSH - Session

```
'-Begin-----
'_
'- Example to show how to select a specific session to do SAP GUI
'- Scripting activities inside it. It scans all connections with all
'- sessions to find the correct one.
'_
'- Author: Stefan Schnell
'_
'-----

'-Directives-----
Option Explicit

'-Sub Action-----
'_
'- Get the selected session and do the action inside it
'_
'-----
Sub Action(session)

    'Insert your SAP GUI Scripting code from recorder here
    MsgBox session.findById("wnd[0]/titl").text

End Sub

'-Function GetSession-----
'_
'- Detects the session
'_
'-----
Function GetSession(SID, TAC)

    Dim SapGuiAuto, application, connections, connection, sessions
    Dim session, sessionInfo, j, i

    Set SapGuiAuto = GetObject("SAPGUI")
    If Not IsObject(SapGuiAuto) Then
        Exit Function
    End If

    Set application = SapGuiAuto.GetScriptingEngine
    If Not IsObject(application) Then
        Set SapGuiAuto = Nothing
        Exit Function
    End If

    Set connections = application.Connections()
    If Not IsObject(connections) Then
        Set SapGuiAuto = Nothing
        Set application = Nothing
        Exit Function
    End If

    '-Loop over connections-----
    For Each connection In connections

        Set sessions = connection.Sessions()

        '-Loop over sessions-----
        For Each session In sessions
```

```

If session.Busy() = vbFalse Then

    '-----
    '-
    '- With the session info object is it possible to select a
    '- specific session which executes the activities. In our
    '- example it is the system name and the transaction code, but
    '- you can use all properties of the session info object.
    '-
    '-----

    Set sessionInfo = session.Info()
    If sessionInfo.SystemName() = SID And _

        sessionInfo.Transaction() = TAC Then
            Set GetSession = session

    End If

End If

Next

Next

End Function

'-Sub Main-----
'-
'- Main procedure to select the session
'-
'------
Sub Main()

    Dim session

    Set session = GetSession("NSP", "SE80")
    Action session

End Sub

'-Main-----
Main()

'-End-----

```

Examples - WSH - Splitter

```
'-Begin-----  
  
Set Splitter = session.findById("wnd[0]/usr/shell")  
  
If Splitter.IsVertical = 0 Then  
    MsgBox "Horizontal Splitter" & vbCrLf & Splitter.GetRowSize(1), _  
        vbOkOnly + vbInformation  
    Splitter.SetRowSize 1, 250  
ElseIf Splitter.IsVertical = 1 Then  
    MsgBox "Vertical Splitter" & vbCrLf & Splitter.GetColSize(1), _  
        vbOkOnly + vbInformation  
    Splitter.SetColSize 1, 500  
ElseIf Splitter.IsVertical = 2 Then  
    MsgBox "Horizontal and vertical Splitter" & vbCrLf & _  
        Splitter.GetRowSize(1) & " - " & Splitter.GetColSize(1), _  
        vbOkOnly + vbInformation  
End If  
  
'-End-----
```


Examples - WSH - Sum All Column Numbers

```
'-Begin-----  
  
'-Sum all numbers in a column of a GridView (ALV grid)-----  
Set table =  
session.findById("wnd[0]/usr/ctrlBCALV_GRID_DEMO_0100_CONT1/shellcont/shell")  
Set Columns = table.ColumnOrder()  
rowTitle = CStr(Columns(7))  
For i = 0 To table.RowCount - 1  
    table.firstVisibleRow = i  
    seatsOCC = seatsOCC + CInt(table.GetCellValue(i, rowTitle))  
Next  
MsgBox CStr(seatsOCC)  
  
'-Sum all numbers in a column of a TableControl-----  
Set scrollBar =  
session.findById("wnd[0]/usr/tblSAPMBIBSTC535").VerticalScrollbar  
For i = 0 To scrollBar.Maximum  
  
session.findById("wnd[0]/usr/tblSAPMBIBSTC535").VerticalScrollbar.Position(i)  
    Steuer = Steuer +  
Cdbl(session.findById("wnd[0]/usr/tblSAPMBIBSTC535").GetCell(0, 6).Text)  
Next  
MsgBox CStr(Steuer)  
  
'-End-----
```

Examples - WSH - Table

[GridView](#)

[Read GridView in File](#)

[Read TableControl](#)

Examples - WSH - Table - GridView

Scroll and Search

```
'-Begin-----  
  
Set GridView =  
session.findById("wnd[0]/usr/cntlBCALV_GRID_DEMO_0100_CONT1/shellcont/shell")  
For i = 0 To GridView.RowCount - 1  
    GridView.SetCurrentCell i, "FUNCNAME"  
    If GridView.GetCellValue(i, "FUNCNAME") = "/OSP/GET_CHKSUM_BKT" Then  
        Exit For  
    End If  
Next  
  
'-End-----
```

Get Column Names

```
'-Begin-----  
  
Set ALV =  
session.findById("wnd[0]/usr/cntlBCALV_GRID_DEMO_0100_CONT1/shellcont/shell")  
  
'-Get all columns-----  
Set Columns = ALV.ColumnOrder  
  
'-Loop over columns-----  
For i = 0 To ALV.ColumnCount - 1  
    '-Column name and the maximum length of the cell-----  
    MsgBox CStr(Columns(i)) & " - Length: " & _  
        CStr(ALV.GetCellMaxLength(1, Columns(i)))  
Next  
  
'-End-----
```

Examples - WSH - Table - Read GridView in File

```
'-Begin-----
'-Constants-----
Const Delimiter = ";"

'-ReadTableInFile-----
Sub ReadTableInFile(session, TableName, FileName)

    '-Reset the session-----
    session.findById("wnd[0]/tbar[0]/okcd").text = "/n"
    session.findById("wnd[0]/tbar[0]/btn[0]").press

    '-Open TAC SE16-----
    session.findById("wnd[0]/tbar[0]/okcd").text = "/nSE16"
    session.findById("wnd[0]/tbar[0]/btn[0]").press

    '-View table-----
    session.findById("wnd[0]/usr/ctxtDATABROWSE-TABLENAME").text = TableName
    session.findById("wnd[0]/tbar[1]/btn[7]").press
    session.findById("wnd[0]/tbar[1]/btn[8]").press

    '-Set display to ALV Grid view-----

    '-Open user specific parameters dialog-----
    '- Attention: Here is a language specific code, customize it
    '-----

    '-German language-----
    'Set Einstellungen = Menu.FindByName("Einstellungen", "GuiMenu")
    'Set BenutzerPar = Einstellungen.FindByName("Benutzerparameter...", _
    '    "GuiMenu")

    '-English language-----
    Set Einstellungen = Menu.FindByName("Settings", "GuiMenu")
    Set BenutzerPar = Einstellungen.FindByName("User Parameters...", _
        "GuiMenu")
    BenutzerPar.Select()

    '-Set the display-----
    Set ALVGridView = session.findById("wnd[1]/usr/tabsG_TABSTRIP/" & _
        "tabp0400/ssubTOOLAREA:SAPLWB_CUSTOMIZING:0400/radRSEUMOD-TBALV_GRID")
    If ALVGridView.Selected = vbFalse Then
        ALVGridView.select()
    End If

    session.findById("wnd[1]/tbar[0]/btn[0]").press
    Set BenutzerPar = Nothing
    Set Einstellungen = Nothing
    Set Menu = Nothing

    '-Get rows and columns-----
    Set table = session.findById("wnd[0]/usr/ctrlGRID1/shellcont/shell")
    Rows = table.RowCount() - 1
    Cols = table.ColumnCount() - 1

    '-Write the table to a CSV file-----
    Set oFile = CreateObject("Scripting.FileSystemObject")
    If IsObject(oFile) Then
        Set SFlightFile = oFile.CreateTextFile(FileName, True)
        If IsObject(SFlightFile) Then
```

```

'-Get the technical title of all columns in the first line-----
Set Columns = table.ColumnOrder()
For j = 0 To Cols
    If j = Cols Then
        SFlightFile.Write(CStr(COLUMNS(j)))
    Else
        SFlightFile.Write(CStr(COLUMNS(j)) & Delimiter)
    End If
Next
SFlightFile.WriteLine("")

'-Get the title of all columns in the second line-----
For j = 0 To Cols
    Set ColumnTitle = table.GetColumnTitles(CStr(COLUMNS(j)))
    If j = Cols Then
        SFlightFile.Write(CStr(COLUMNTITLE(0)))
    Else
        SFlightFile.Write(CStr(COLUMNTITLE(0)) & Delimiter)
    End If
Next
SFlightFile.WriteLine("")

For i = 0 To Rows

    For j = 0 To Cols
        If j = Cols Then
            SFlightFile.Write(table.GetCellValue(i, CStr(COLUMNS(j))))
        Else
            SFlightFile.Write(table.GetCellValue(i, CStr(COLUMNS(j))) & _
                Delimiter)
        End If
    Next

    '-Each 32 lines actualize the grid-----
    If i Mod 32 = 0 Then
        table.SetCurrentCell i, CStr(COLUMNS(0))
        table.firstVisibleRow = i
    End If

    '-Carriage and return after a line-----
    If i <> Rows Then
        SFlightFile.WriteLine("")
    End If

Next
SFlightFile.Close

End If

End If

Set ALVGridView = Nothing
Set Columns = Nothing
Set table = Nothing

End Sub

'-Sub Main-----
Sub Main

    If Not IsObject(application) Then
        Set SapGuiAuto = GetObject("SAPGUI")
        Set application = SapGuiAuto.GetScriptingEngine
    End If

```

```
If Not IsObject(connection) Then
    Set connection = application.Children(0)
End If

If Not IsObject(session) Then
    Set session = connection.Children(0)
End If

'-Read the table SFLIGHT in a file-----
ReadTableInFile session, "SFLIGHT", "C:\\Dummy\\SFlight.csv"

End Sub

'-Main-----
Main

'-End-----
```

Examples - WSH - Table - Read TableControl

```
'-Begin-----
'_
'- TAC GUIBIBS
'_
'-----

'-Directives-----
Option Explicit

'-Sub Main-----
Sub Main

    Dim SapGuiAuto, application, connection, session
    Dim TableId, Table, RowCount, ColCount, Row, Col, Cell, Out

    If Not IsObject(application) Then
        Set SapGuiAuto = GetObject("SAPGUI")
        Set application = SapGuiAuto.GetScriptingEngine
    End If

    If Not IsObject(connection) Then
        Set connection = application.Children(0)
    End If

    If Not IsObject(session) Then
        Set session = connection.Children(0)
    End If

    TableId = "wnd[0]/usr/tblSAPMBIBSTC535"
    Set Table = session.findById(TableId)
    RowCount = Table.RowCount
    ColCount = Table.Columns.Count

    For Row = 0 To RowCount - 1

        Table.verticalScrollbar.position = Row
        Set Table = session.findById(TableId)
        For Col = 0 To ColCount - 1

            Set Cell = Table.GetCell(0, Col)
            If Col < ColCount - 1 Then
                Out = Out & Cell.Text & ";"
            Else
                Out = Out & Cell.Text
            End If

        Next

        Out = Out & vbNewLine

        If Table.verticalScrollbar.Position = Table.verticalScrollbar.Maximum Then
            Exit For
        End If

    Next

    MsgBox Out

End Sub
```

```
'-Main-----
Main
'-End-----
```

Oberflächengestaltung: Übersichtsbild



Belegnummer	1500005500	Buchungskreis	0001
Belegdatum	24.01.1994	Geschäftsjahr	1994
Referenz		Übergreifd.Nr	
Währung	DM	Soll/Haben	

Pos	BS	GsB	Kontonr	Bezeichnung	M.	Steuer	Betrag
001	50		0000276000	Skonto-Ertrag	V1	4,50-	30,00-
002	25		0000000010	Soesel & Parnter GmbH...		0,00	1.150,00-
003	40		0000160099	Kreditoren-Verbindlichk...		0,00	0,00
004	50	0004	0000113101	DeuBa (Ausgangsscheck...		0,00	1.115,50-
005	50		0000154000	Vorsteuer (BRD)	V1	0,00	4,50-

Position 1 von 20

```
'-Begin-----
'-
'- TAC DWDM
'-
'------
```

```
'-Directives-----
Option Explicit
```

```
'-Sub Main-----
Sub Main()
```

```
Dim SapGuiAuto, app, connection, session
Dim TableId, Table, RowCount, ColCount, Row, Col, Cell, Out
Dim StartTime, EndTime, ExecTime
```

```
On Error Resume Next
Set SapGuiAuto = GetObject("SAPGUI")
On Error GoTo 0
If Not IsObject(SapGuiAuto) Then
    MsgBox "Can not get SapGuiAuto", vbOkOnly, "Hint"
    Exit Sub
End If
```

```
On Error Resume Next
Set app = SapGuiAuto.GetScriptingEngine
On Error GoTo 0
If Not IsObject(app) Then
    MsgBox "Can not get application", vbOkOnly, "Hint"
    Exit Sub
End If
```

```
app.HistoryEnabled = False
```

```
On Error Resume Next
```



```

Set connection = app.Children(0)
On Error GoTo 0
If Not IsObject(connection) Then
    MsgBox "Can not get connection", vbOkOnly, "Hint"
    Exit Sub

End If
If connection.DisabledByServer = True Then
    MsgBox "Scripting is disabled by server", vbOkOnly, "Hint"
    Exit Sub
End If

On Error Resume Next
Set session = connection.Children(0)
On Error GoTo 0
If Not IsObject(session) Then
    MsgBox "Can not get session", vbOkOnly, "Hint"
    Exit Sub
End If

If session.Busy = True Then
    MsgBox "Session is busy", vbOkOnly, "Hint"
    Exit Sub
End If

If session.Info.IsLowSpeedConnection = True Then
    MsgBox "Connection is low speed", vbOkOnly, "Hint"
    Exit Sub
End If

StartTime = Timer

TableID = "wnd[0]/usr/tblRSDEMO_TABLE_CONTROLTABLE_CONTROL"
set Table = session.findById(TableID)
RowCount = Table.RowCount
ColCount = Table.Columns.Count
For Row = 0 To RowCount - 1

    Table.verticalScrollbar.position = Row

    Set Table = session.findById(TableID)
    For Col = 0 To ColCount - 1
        Set Cell = Table.GetCell(0, Col)
        If Col < ColCount - 1 Then
            Out = Out & Cell.Text & ";"
        Else
            Out = Out & Cell.Text
        End If
    Next

    Out = Out & vbNewLine

    If Table.verticalScrollbar.Position = _
        Table.verticalScrollbar.Maximum Then
        Exit For
    End If

Next

EndTime = Timer
ExecTime = EndTime - StartTime
MsgBox ExecTime

app.HistoryEnabled = True

```

End Sub

'-Main-----

Main

'-End-----

Table control title



	Airline	Fli...	Depart.city	D...	Arrival city	Flight...	Dej
<input type="checkbox"/>	AA American Airline	17	NEW YORK	JFK	SAN FRANCISCO	0:00 11:	
<input type="checkbox"/>	AA American Airli...	64	SAN FRANCISCO	SFO	NEW YORK	0:00 09:	
<input type="checkbox"/>	AZ Alitalia	555	ROME	FCO	FRANKFURT	0:00 19:	
<input type="checkbox"/>	AZ Alitalia	788	ROME	FCO	TOKYO	0:00 12:	
<input type="checkbox"/>	AZ Alitalia	789	TOKYO	TYO	ROME	0:00 11:	
<input type="checkbox"/>	AZ Alitalia	790	ROME	FCO	OSAKA	0:00 10:	
<input type="checkbox"/>	DL Delta Airlines	106	NEW YORK	JFK	FRANKFURT	0:00 19:	
<input type="checkbox"/>	DL Delta Airlines	1699	NEW YORK	JFK	SAN FRANCISCO	0:00 17:	
<input type="checkbox"/>	DL Delta Airlines	1984	SAN FRANCISCO	SFO	NEW YORK	0:00 10:	
<input type="checkbox"/>	JL Japan Airlines	407	TOKYO	NRT	FRANKFURT	0:00 13:	
<input type="checkbox"/>	JL Japan Airlines	408	FRANKFURT	FRA	TOKYO	0:00 20:	
<input type="checkbox"/>	LH Lufthansa	400	FRANKFURT	FRA	NEW YORK	0:00 10:	
<input type="checkbox"/>	LH Lufthansa	401	NEW YORK	JFK	FRANKFURT	0:00 18:	
<input type="checkbox"/>	LH Lufthansa	402	FRANKFURT	FRA	NEW YORK	0:00 13:	
<input type="checkbox"/>	LH Lufthansa	2402	FRANKFURT	FRA	BERLIN	0:00 10:	

Examples - WSH - Tree

[Detect Type](#)

[Get All Node Keys Text](#)

[Read List Items](#)

[Read Description \(1\)](#)

[Read Description \(2\)](#)

[Open All Nodes](#)

Examples - WSH - Tree - Detect Type

```
'-Begin-----  
'_  
'- Detects the tree type (simple, list or column)  
'_  
'-----  
  
If Not IsObject(application) Then  
    Set SapGuiAuto = GetObject("SAPGUI")  
    Set application = SapGuiAuto.GetScriptingEngine  
End If  
  
If Not IsObject(connection) Then  
    Set connection = application.Children(0)  
End If  
  
If Not IsObject(session) Then  
    Set session = connection.Children(0)  
End If  
  
Set Tree = session.findById("wnd[0]/usr/cntlTREE_CONTAINER/shellcont/shell")  
  
Select Case Tree.GetTreeType  
    Case 0  
        MsgBox "Simple tree"  
    Case 1  
        MsgBox "List tree"  
    Case 2  
        MsgBox "Column tree"  
End Select  
  
'-End-----
```

Examples - WSH - Tree - Get All Node Keys Text

```
'-Begin-----
'-
'- TAC SESSION_MANAGER or one of the demo reports
'-
'-----

If Not IsObject(application) Then
    Set SapGuiAuto = GetObject("SAPGUI")
    Set application = SapGuiAuto.GetScriptingEngine
End If

If Not IsObject(connection) Then
    Set connection = application.Children(0)
End If

If Not IsObject(session) Then
    Set session = connection.Children(0)
End If

Set Tree =
session.findById("wnd[0]/usr/cntlIMAGE_CONTAINER/shellcont/shell/shellcont[0]/
shell")
Set AllNodeKeys = Tree.GetAllNodeKeys()

'Get text of last node
MsgBox Tree.GetNodeTextByKey(AllNodeKeys(AllNodeKeys.Count - 1))

'Get key of last node
MsgBox AllNodeKeys(AllNodeKeys.Count - 1)

'Loop over all nodes
For Each NodeKey In AllNodeKeys
    MsgBox CStr(NodeKey) & " - " & Tree.GetNodeTextByKey(NodeKey)
Next

'-End-----
```

Examples - WSH - Tree - Read List Items

```
'-Begin-----

If Not IsObject(application) Then
    Set SapGuiAuto = GetObject("SAPGUI")
    Set application = SapGuiAuto.GetScriptingEngine
End If

If Not IsObject(connection) Then
    Set connection = application.Children(0)
End If

If Not IsObject(session) Then
    Set session = connection.Children(0)
End If

session.findById("wnd[0]/tbar[0]/okcd").text = "/nSE38"
session.findById("wnd[0]").sendVKey 0
session.findById("wnd[0]/usr/ctxtRS38M-PROGRAMM").text =
"SAPTLIST_TREE_MODEL_DEMO"
session.findById("wnd[0]").sendVKey 8
Set Tree = session.findById("wnd[0]/usr/ctlTREE_CONTAINER/shellcont/shell")

'-Expands the nodes of the second level-----
Set AllNodeKeys = Tree.GetAllNodeKeys()
For Each NodeKey In AllNodeKeys
    If Tree.IsFolderExpandable(NodeKey) Then
        Tree.ExpandNode(NodeKey)
    End If
Next

'-Reads the items of the nodes-----
Set AllNodeKeys = Tree.GetAllNodeKeys()

For Each NodeKey In AllNodeKeys

    MsgBox Tree.GetItemText(NodeKey, "1") + " - " + _
        Tree.GetItemText(NodeKey, "2") + " - " + _
        Tree.GetItemText(NodeKey, "3") + " - " + _
        Tree.GetItemText(NodeKey, "4")

    If InStr(Tree.GetNodeTextByKey(NodeKey), "SAPTRIXTROX") Then
        MsgBox "Ziel erreicht"
    End If

Next

'-End-----
```

✓	Objekte		
✓	Dynpros		
✓	0100 MUELLER	Comment to Dynpro 100	
✗	0200 HARRYHIRSCH	Comment to Dynpro 200	
✓	Programme		
✓	SAPTROX1	Comment to SAPTROX1	
✓	SAPTRIXTROX	Comment to SAPTRIXTROX	

Examples - WSH - Tree - Read Description (1)

```
'-Begin-----
'_
'- Read description of a column tree with TAC SE80
'_
'-----

If Not IsObject(application) Then
    Set SapGuiAuto = GetObject("SAPGUI")
    Set application = SapGuiAuto.GetScriptingEngine
End If

If Not IsObject(connection) Then
    Set connection = application.Children(0)
End If

If Not IsObject(session) Then
    Set session = connection.Children(0)
End If

Set Tree =
session.FindById("wnd[0]/shellcont/shell/shellcont[3]/shell/shellcont[2]/shell
")

'Column 2 is Beschreibung, but the index starts with 0
'To get the correct column 2 minus 1
colName = Tree.GetColumnNames.Item("1")
Set col = Tree.GetColumnCol(colName)

'Get top node
topNode = CStr(Tree.TopNode)

'Counts all sub nodes of the top node
cntSubNodes = Tree.GetNodeChildrenCountByPath(topNode)

'Scan all sub nodes
For i = 1 To cntSubNodes

    'The path of the subnodes is 1/1, 1/2 etc.
    NodeName = Tree.GetNodeTextByPath(topNode & "/" & CStr(i))

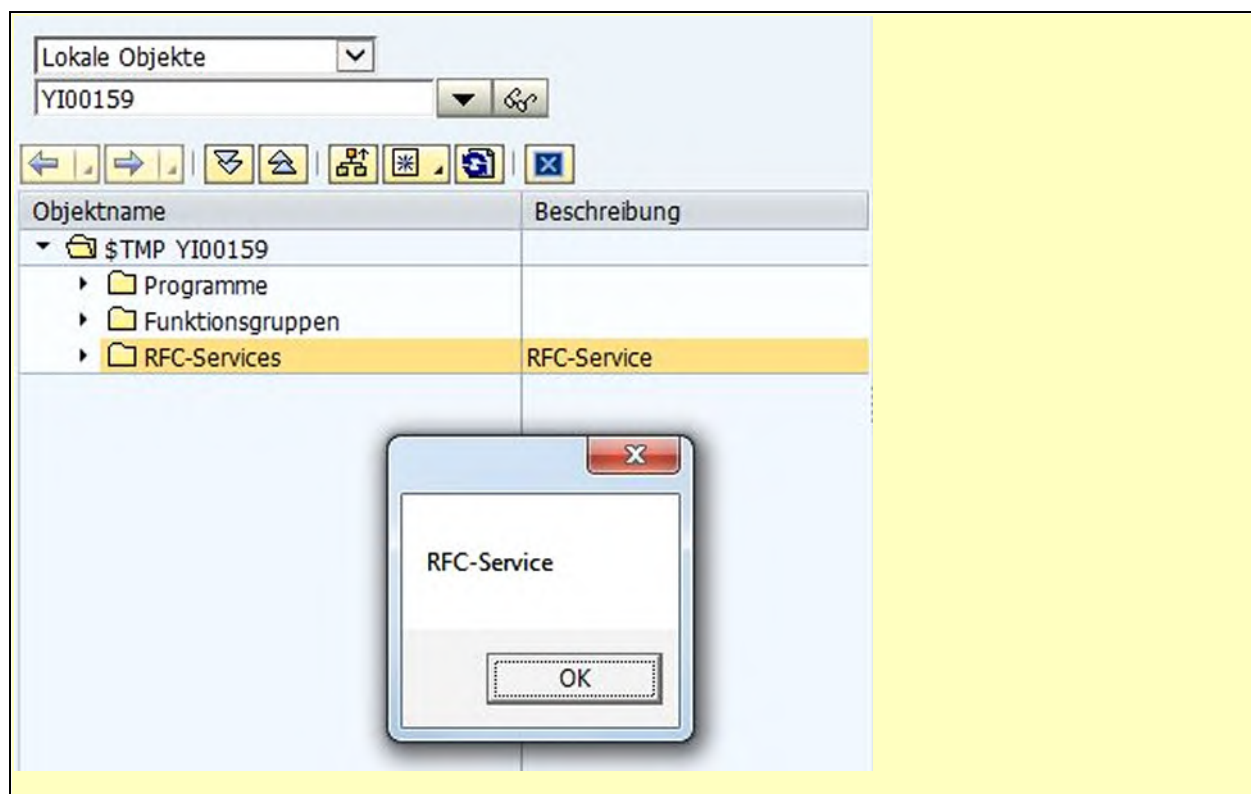
    'Search for the correct node name
    If NodeName = "RFC-Services" Then

        'Get the key of the node, index starts also with 0 therefore -1
        Key = CLng(Tree.GetNodeKeyByPath(topNode & "/" & CStr(i))) - 1
        'Get the description
        Beschreibung = col.Item(Key)
        MsgBox Beschreibung

    End If

Next

'-End-----
```



Examples - WSH - Tree - Read Description (2)

```
'-Begin-----
'_
'- Read description of a column tree with TAC SRMREGEDIT
'_
'-----

If Not IsObject(application) Then
    Set SapGuiAuto = GetObject("SAPGUI")
    Set application = SapGuiAuto.GetScriptingEngine
End If

If Not IsObject(connection) Then
    Set connection = application.Children(0)
End If

If Not IsObject(session) Then
    Set session = connection.Children(0)
End If

Set Tree = session.findById("wnd[0]/shellcont/shell/shellcont[2]/shell")
colName = Tree.GetColumnNames.Item("2")
Set col = Tree.GetColumnCol(colName)
topNode = CStr(Tree.TopNode) : Key = topNode

'-Counter to get correct index of node-----
cnt = 1 + Tree.GetNodeChildrenCountByPath(topNode)

'-Scan nodes on the first level-----
For i = 1 To Tree.GetNodeChildrenCountByPath(topNode)

    cnt = cnt + 1
    nodeName = Tree.GetNodeTextByKey(Key)
    nodePath = Tree.GetNodePathByKey(Key)

    If nodeName = "Anwendungs-Registry" Then

        '-Scan nodes on the second level-----
        For j = 1 To Tree.GetNodeChildrenCount(Key)
            cnt = cnt + 1
            subNode = Tree.GetNodeKeyByPath(nodePath & "/" & CStr(j))
            subNodeName = Tree.GetNodeTextByKey(subNode)
            Select Case subNodeName
                Case "S_AREA_GDMA"
                    MsgBox col.Item(cnt)
                Case "S_AREA_RMS"
                    MsgBox col.Item(cnt)
            End Select
        Next

    End If

    If i < Tree.GetNodeChildrenCountByPath(topNode) Then
        Key = Tree.GetNextNodeKey(Key)
    End If

Next

'-End-----
```

Examples - WSH - Tree - Open All Nodes

```
'-Sub OpenAllNodes-----  
'_  
'- Opens all nodes of a tree  
'_  
'-----  
Sub OpenAllNodes(Tree)  
    Dim ErrNumber  
    Set AllNodeKeys = Tree.GetAllNodeKeys()  
    For Each NodeKey In AllNodeKeys  
        If Not Tree.IsFolderExpanded(NodeKey) Then  
            On Error Resume Next  
            Tree.ExpandNode(NodeKey)  
            ErrNumber = Err.number  
            On Error GoTo 0  
            If ErrNumber = 0 Then  
                OpenAllNodes(Tree)  
            End If  
        End If  
    Next  
End Sub
```

Examples - WSH - Start Multiple TACs (1)

```
'-Begin-----
'-Directives-----
Option Explicit

'-Sub Action-----
Sub Action(con, ses)

    Dim objShell, RegEx, Matches, con_no, ses_no

    Set RegEx = New RegExp
    RegEx.Pattern = "[\d]+"
    Set Matches = RegEx.Execute(con)
    con_no = Matches(0).Value
    Set Matches = RegEx.Execute(ses)
    ses_no = Matches(0).Value
    Set objShell = Wscript.CreateObject("WScript.Shell")
    objShell.Run "YourScript.vbs " + con_no + " " + ses_no

End Sub

'-Function GetSession-----
Function GetSession(connection, TAC)

    Dim sessions, session, sessionInfo, j, i

    Set sessions = connection.Sessions()

    '-Loop over sessions-----
    For Each session In sessions
        If session.Busy() = vbFalse Then
            Set sessionInfo = session.Info()
            If sessionInfo.Transaction() = TAC Then
                Set GetSession = session
            End If
        End If
    Next

End Function

'-Sub Main-----
Sub Main()

    Dim SapGuiAuto, app, connection, session
    Dim session_SE16, session_SE37, session_SE38
    Dim arr

    Set SapGuiAuto = GetObject("SAPGUI")
    If Not IsObject(SapGuiAuto) Then
        Exit Sub
    End If

    Set app = SapGuiAuto.GetScriptingEngine
    If Not IsObject(app) Then
        Exit Sub
    End If

    Set connection = app.Children(0)
    If Not IsObject(connection) Then
        Exit Sub
    End If
```

```

End If

If connection.DisabledByServer = True Then
    Exit Sub
End If

Set session = connection.Children(0)
If Not IsObject(session) Then
    Exit Sub
End If

If session.Info.IsLowSpeedConnection = True Then
    Exit Sub
End If

session.FindById("wnd[0]/tbar[0]/okcd").text = "/oSE16"
session.FindById("wnd[0]").sendVKey 0
session.FindById("wnd[0]/tbar[0]/okcd").text = "/oSE37"
session.FindById("wnd[0]").sendVKey 0
session.FindById("wnd[0]/tbar[0]/okcd").text = "/oSE38"
session.FindById("wnd[0]").sendVKey 0

Set session_SE16 = GetSession(connection, "SE16")
arr = Split(session_SE16.ID, "/")
WScript.Sleep 500
Action arr(2), arr(3)

Set session_SE37 = GetSession(connection, "SE37")
arr = Split(session_SE37.ID, "/")
WScript.Sleep 500
Action arr(2), arr(3)

Set session_SE38 = GetSession(connection, "SE38")
arr = Split(session_SE38.ID, "/")
WScript.Sleep 500
Action arr(2), arr(3)

End Sub

'-Main-----
Main()

'-End-----

```

YourScript.vbs

```

'-Begin-----

Set Args = WScript.Arguments
con = Args(0)
ses = Args(1)

Set SapGuiAuto = GetObject("SAPGUI")
If Not IsObject(SapGuiAuto) Then
    WScript.Quit
End If

Set app = SapGuiAuto.GetScriptingEngine
If Not IsObject(app) Then
    WScript.Quit
End If

```

```
Set connection = app.Children(CLng(con))
If Not IsObject(connection) Then
    WScript.Quit
End If

If connection.DisabledByServer = True Then
    WScript.Quit
End If

Set session = connection.Children(CLng(ses))
If Not IsObject(session) Then
    WScript.Quit
End If

If session.Info.IsLowSpeedConnection = True Then
    WScript.Quit
End If

MsgBox session.Info.Transaction()

'-End-----
```

Examples - WSH - Start Multiple TACs (2)

To be able to process a higher amount of data in the same time, parallelization is a valid approach. The SAP GUI scripting, as an automation interface to the SAP GUI for Windows, is a basis for many RPA platforms. In this context it is worth taking a look at the parallelization capabilities of the SAP GUI Scripting.

The following example code has three routines:

1. Main

Connects the open session, just like the standard does. Then two additional sessions are opened with different transaction codes, in our example the SE16 and SE37. After that the session object via GetSession is detected, based on the transaction code, and an external script is started via Action.

2. GetSession

Loops over all the sessions of the connection to detect the session with the transaction code and delivers the session object.

3. Action

Starts an external script with the connection and session number as parameters.

```
'-Begin-----
'-Directives-----
Option Explicit

'-Sub Action-----
'_
'- Starts external script with the connection nubmer,session number
'- and transaction code as parameters
'_
'-----
Sub Action(con, ses, TAC)

    Dim objShell, RegEx, Matches, con_no, ses_no

    '-Extracts connection and session number-----
    Set RegEx = New RegExp
    RegEx.Pattern = "[\d]+"
    Set Matches = RegEx.Execute(con)
    con_no = Matches(0).Value
    Set Matches = RegEx.Execute(ses)
    ses_no = Matches(0).Value

    '-Executes the external script-----
    Set objShell = Wscript.CreateObject("WScript.Shell")
    objShell.Run "YourScript." + TAC + ".vbs " + con_no + " " + ses_no

End Sub

'-Function GetSession-----
'_
'- Detects a session by transaction code (TAC)
'_
'-----
Function GetSession(connection, TAC)

    Dim sessions, session, sessionInfo, j, i
```

```

Set sessions = connection.Sessions()
'-Loop over sessions-----
For Each session In sessions
    If session.Busy() = vbFalse Then
        Set sessionInfo = session.Info()
        If sessionInfo.Transaction() = TAC Then
            Set GetSession = session
        End If
    End If
End For
Next

End Function

'-Sub Main-----
Sub Main()

    Dim SapGuiAuto, app, connection, session
    Dim session_SE16, session_SE37
    Dim arr

    Set SapGuiAuto = GetObject("SAPGUI")
    If Not IsObject(SapGuiAuto) Then
        Exit Sub
    End If

    Set app = SapGuiAuto.GetScriptingEngine
    If Not IsObject(app) Then
        Exit Sub
    End If

    Set connection = app.Children(0)
    If Not IsObject(connection) Then
        Exit Sub
    End If

    If connection.DisabledByServer = True Then
        Exit Sub
    End If

    Set session = connection.Children(0)
    If Not IsObject(session) Then
        Exit Sub
    End If

    If session.Info.IsLowSpeedConnection = True Then
        Exit Sub
    End If

    '-Open different sessions with different TACs-----
    session.findById("wnd[0]/tbar[0]/okcd").text = "/oSE16"
    session.findById("wnd[0]").sendVKey 0
    session.findById("wnd[0]/tbar[0]/okcd").text = "/oSE37"
    session.findById("wnd[0]").sendVKey 0
    WScript.Sleep 500

    '-Detects the session with TAC SE16 and calls an external script-----
    Set session_SE16 = GetSession(connection, "SE16")
    arr = Split(session_SE16.ID, "/")
    Action arr(2), arr(3), "SE16"

    '-Detects the session with TAC SE37 and calls an external script-----
    Set session_SE37 = GetSession(connection, "SE37")
    arr = Split(session_SE37.ID, "/")
    Action arr(2), arr(3), "SE37"

```

```
End Sub
```

```
'-Main-----  
Main()  
  
'-End-----
```

YourScript.SE16.vbs

Here the script which automates the TAC SE16. First the connection and session number of the arguments are read. Then we have the standard sequence for establishing the connection to the session, with use of the passed arguments. Last but not least a for loop that does the same thing over and over again.

```
'-Begin-----  
  
Set Args = WScript.Arguments  
con = Args(0)  
ses = Args(1)  
  
Set SapGuiAuto = GetObject("SAPGUI")  
If Not IsObject(SapGuiAuto) Then  
    WScript.Quit  
End If  
  
Set app = SapGuiAuto.GetScriptingEngine  
If Not IsObject(app) Then  
    WScript.Quit  
End If  
  
Set connection = app.Children(CLng(con))  
If Not IsObject(connection) Then  
    WScript.Quit  
End If  
  
If connection.DisabledByServer = True Then  
    WScript.Quit  
End If  
  
Set session = connection.Children(CLng(ses))  
If Not IsObject(session) Then  
    WScript.Quit  
End If  
  
If session.Info.IsLowSpeedConnection = True Then  
    WScript.Quit  
End If  
  
For i = 1 To 50  
    session.findById("wnd[0]/usr/ctxtDATABROWSE-TABLENAME").text = "TADIR"  
    session.findById("wnd[0]/usr/ctxtDATABROWSE-TABLENAME").caretPosition = 5  
    session.findById("wnd[0]").sendVKey 0  
    session.findById("wnd[0]").sendVKey 31  
    session.findById("wnd[1]/tbar[0]/btn[0]").press  
    session.findById("wnd[0]/tbar[0]/btn[3]").press  
Next  
  
'-End-----
```


YourScript.SE37.vbs

Here the script which automates the TAC SE37. It does exactly the same as above.

```
'-Begin-----

Set Args = WScript.Arguments
con = Args(0)
ses = Args(1)

Set SapGuiAuto = GetObject("SAPGUI")
If Not IsObject(SapGuiAuto) Then
    WScript.Quit
End If

Set app = SapGuiAuto.GetScriptingEngine
If Not IsObject(app) Then
    WScript.Quit
End If

Set connection = app.Children(CLng(con))
If Not IsObject(connection) Then
    WScript.Quit
End If

If connection.DisabledByServer = True Then
    WScript.Quit
End If

Set session = connection.Children(CLng(ses))
If Not IsObject(session) Then
    WScript.Quit
End If

If session.Info.IsLowSpeedConnection = True Then
    WScript.Quit
End If

For i = 1 To 25
    session.findById("wnd[0]/usr/ctxtRS38L-NAME").text = "RFC_READ_TABLE"
    session.findById("wnd[0]/usr/ctxtRS38L-NAME").caretPosition = 14
    session.findById("wnd[0]/usr/btnBUT3").press
    session.findById("wnd[0]/usr/tabsFUNC_TAB_STRIP/tabpHEADER").select
    session.findById("wnd[0]/usr/tabsFUNC_TAB_STRIP/tabpIMPORT").select
    session.findById("wnd[0]/usr/tabsFUNC_TAB_STRIP/tabpEXPORT").select
    session.findById("wnd[0]/usr/tabsFUNC_TAB_STRIP/tabpCHANGE").select
    session.findById("wnd[0]/usr/tabsFUNC_TAB_STRIP/tabpTABLES").select
    session.findById("wnd[0]/usr/tabsFUNC_TAB_STRIP/tabpEXCEPT").select
    session.findById("wnd[0]/tbar[0]/btn[3]").press
Next

'-End-----
```

Now when the script is started, the sessions are opened with the transaction codes. Then other Windows Script Hosts are opened parallel and the scripts for SE16 and SE37 are executed parallel.

Examples - WSH - HTMLViewer

```
'-Begin-----  
'_  
'- Get and set the body text of an HTML control on a screen  
'_  
'-----  
  
'-Sub Main-----  
Sub Main()  
  
    Dim SapGuiAuto, app, connection, session  
    Dim oBrowser, BrowserCtrlType  
  
    On Error Resume Next  
    Set SapGuiAuto = GetObject("SAPGUI")  
    On Error GoTo 0  
    If Not IsObject(SapGuiAuto) Then  
        MsgBox "Can not get SapGuiAuto", vbOkOnly, "Hint"  
        Exit Sub  
    End If  
  
    On Error Resume Next  
    Set app = SapGuiAuto.GetScriptingEngine  
    On Error GoTo 0  
    If Not IsObject(app) Then  
        MsgBox "Can not get application", vbOkOnly, "Hint"  
        Exit Sub  
    End If  
  
    On Error Resume Next  
    Set connection = app.Children(0)  
    On Error GoTo 0  
    If Not IsObject(connection) Then  
        MsgBox "Can not get connection", vbOkOnly, "Hint"  
        Exit Sub  
    End If  
  
    If connection.DisabledByServer = True Then  
        MsgBox "Scripting is disabled by server", vbOkOnly, "Hint"  
        Exit Sub  
    End If  
  
    On Error Resume Next  
    Set session = connection.Children(0)  
    On Error GoTo 0  
    If Not IsObject(session) Then  
        MsgBox "Can not get session", vbOkOnly, "Hint"  
        Exit Sub  
    End If  
  
    If session.Info.IsLowSpeedConnection = True Then  
        MsgBox "Connection is low speed", vbOkOnly, "Hint"  
        Exit Sub  
    End If  
  
    '-Call TAC SE38-----  
    session.findById("wnd[0]/tbar[0]/okcd").text = "/nse38"  
    session.findById("wnd[0]").sendVKey 0  
  
    '-Execute report SAPHTML_DEMO1-----  
    session.findById("wnd[0]/usr/ctxtRS38M-PROGRAMM").text = "SAPHTML_DEMO1"
```

```

session.findById("wnd[0]/tbar[1]/btn[8]").press

'-Get the handle of the InternetExplorer.Application object-----
'_
'- This object controls an instance of Windows Internet Explorer
'- through automation
'_
'- Important hint:
'- The Internet Explorer (IE) 11 desktop application will end support
'- June 15, 2022, customers are encouraged to move to Microsoft Edge
'- with IE mode, it enables backward compatibility and will be
'- supported through at least 2029
'_
'-----
Set oBrowser =
session.findById("wnd[0]/usr/cntlHTML/shellcont/shell").BrowserHandle

'-Detect browser control type-----
'_
'- 0 = Internet Explorer, 1 = Edge (based on Chromium)
'_
'- Access to the Document Object Model (DOM) works only with
'- IE control
'_
'-----
BrowserCtrlType =
session.findById("wnd[0]/usr/cntlHTML/shellcont/shell").GetBrowserControlType(
)

If BrowserCtrlType = 0 Then

    '-Get the innerText of the body tag of the HTML site-----
    MsgBox oBrowser.document.body.innerText

    '-Add an additional tag with a text to the HTML site-----
    oBrowser.document.body.innerHTML = oBrowser.document.body.innerHTML + _
        "<div>Hello World!</div>"

End If

End Sub

'-Main-----
Main

'-End-----

```

```

'-Begin-----
'_
'- Get the body text of an HTML control from a dialog
'_
'-----

'-Sub Main-----
Sub Main()

    Dim SapGuiAuto, app, connection, session
    Dim ID, oBrowser, InnerText

    On Error Resume Next
    Set SapGuiAuto = GetObject("SAPGUI")
    On Error GoTo 0
    If Not IsObject(SapGuiAuto) Then
        MsgBox "Can not get SapGuiAuto", vbOkOnly, "Hint"
        Exit Sub
    End If

    On Error Resume Next
    Set app = SapGuiAuto.GetScriptingEngine
    On Error GoTo 0
    If Not IsObject(app) Then
        MsgBox "Can not get application", vbOkOnly, "Hint"
        Exit Sub
    End If

    On Error Resume Next
    Set connection = app.Children(0)
    On Error GoTo 0
    If Not IsObject(connection) Then
        MsgBox "Can not get connection", vbOkOnly, "Hint"
        Exit Sub
    End If

    If connection.DisabledByServer = True Then
        MsgBox "Scripting is disabled by server", vbOkOnly, "Hint"
        Exit Sub
    End If

    On Error Resume Next
    Set session = connection.Children(0)
    On Error GoTo 0
    If Not IsObject(session) Then
        MsgBox "Can not get session", vbOkOnly, "Hint"
        Exit Sub
    End If

    If session.Info.IsLowSpeedConnection = True Then
        MsgBox "Connection is low speed", vbOkOnly, "Hint"
        Exit Sub
    End If

    '-Call TAC SE37-----
    session.findById("wnd[0]/tbar[0]/okcd").text = "/nSE37"
    session.findById("wnd[0]").sendVKey 0

    '-Call FM POPUP_TO_CONFIRM-----
    session.findById("wnd[0]/usr/ctxtRS38L-NAME").text = "POPUP_TO_CONFIRM"
    session.findById("wnd[0]").sendVKey 8

    '-Fill the arguments-----

```

```

session.findById("wnd[0]/usr/txt[34,7]").text = "Information"
session.findById("wnd[0]/usr/txt[34,9]").text = "This is a test"
session.findById("wnd[0]/usr/txt[34,10]").text = "Yes"
session.findById("wnd[0]/usr/txt[34,12]").text = "No"
session.findById("wnd[0]/usr/txt[34,15]").text = ""
session.findById("wnd[0]").sendVKey 8

WScript.Sleep(2500)

'-Get the handle of the InternetExplorer.Application object-----
ID =
"wnd[1]/usr/subSUBSCREEN:SAPLSPO1:0502/cntlHTML_CONTROL_CON/shellcont/shell"
Set oBrowser = session.findById(ID).BrowserHandle

'-Get the innerText of the body tag of the HTML site-----
MsgBox oBrowser.document.body.innerText

'-Get the innerText-----
InnerText = session.findById(ID).BrowserHandle.document.all(0).innertext
MsgBox InnerText

End Sub

'-Main-----
Main

'-End-----

```

Examples - VBA (Visual Basic for Applications)

You can find information to prepare VBA [here](#).

[ClearAllChangeable Fields](#)

Examples - VBA - ClearAllChangeableFields

In some cases users have defined specific default values. In the case of automation, this can lead to processing errors. To exclude this, all contents of fields of a screen can be deleted with this approach.

```
'-Begin-----
'-Directives-----
Option Explicit

'-ClearAllChangeableFields-----
'-
'- Clear all changeable fields of an SAP GUI session
'-
'------
Sub ClearAllChangeableFields(obj As Object)

    Dim cntSess As Integer
    Dim i As Integer
    Dim Child As Object

    On Error Resume Next

    cntSess = obj.Children.Count()
    If cntSess = 0 Then
        On Error GoTo 0
        Exit Sub
    End If

    For i = 0 To cntSess - 1
        Set Child = obj.Children.Item(CLng(i))
        ClearAllChangeableFields Child
        If Child.Changeable = vbTrue And Child.ContainerType = vbFalse Then
            Select Case Child.Type()
                Case "GuiCheckBox"
                    Child.Selected = 0
                Case "GuiCTextField", "GuiTextField"
                    Child.Text = ""
                Case "GuiComboBox"
                    Child.Key = " "
            End Select
        End If
    Next

    On Error GoTo 0
End Sub

'-Test-----
Sub Test()

    Dim SapGuiAuto As Object
    Dim app As SAPFEWSELib.GuiApplication
    Dim connection As SAPFEWSELib.GuiConnection
    Dim session As SAPFEWSELib.GuiSession

    '-Main-----
    Set SapGuiAuto = GetObject("SAPGUI")
    Set app = SapGuiAuto.GetScriptingEngine
    Set connection = app.Children(1)
    Set session = connection.Children(0)
```

```
ClearAllChangeableFields session
```

```
End Sub
```

```
'-End-----
```


Examples - Autolt

[CheckTAC](#)

[Copy Table Data to Clipboard](#)

Examples - Autolt - CheckTAC

```
;-Begin-----
;-Directives-----
AutoItSetOption("MustDeclareVars", 1)

;-Includes-----
#include <StringConstants.au3>

;-CheckTAC-----
Func CheckTAC()

    Local $SapGuiAuto, $application, $connections, $connection
    Local $sessions, $session, $UserArea, $OrderType, $cmbOrderType

    $SapGuiAuto = ObjGet("SAPGUI")
    If Not IsObj($SapGuiAuto) Or @Error Then
        Return
    EndIf

    $application = $SapGuiAuto.GetScriptingEngine()
    If Not IsObj($application) Then
        Return
    EndIf

    $connections = $application.Connections()
    If Not IsObj($connections) Then
        Return
    EndIf

    For $connection In $connections

        If $connection.DisabledByServer = True Then
            ContinueLoop
        EndIf

        $sessions = $connection.Sessions()
        If Not IsObj($sessions) Then
            ContinueLoop
        EndIf

        For $session In $sessions

            If $session.Busy = True Then
                ContinueLoop
            EndIf

            If $session.Info.IsLowSpeedConnection = True Then
                ContinueLoop
            EndIf

            Select
                Case $session.Info.Transaction = "ME21N"
                    ;-Create Purchase Order-----

                Case $session.Info.Transaction = "ME22N"
                    ;-Change Purchase Order-----
```

```

Case $session.Info.Transaction = "ME23N"
;-Display Purchase Order-----

    $UserArea = $session.findById("wnd[0]/usr")
    $cmbOrderType = $UserArea.findByName("MEPO_TOPLINE-BSART",
GuiComboBox")
    $OrderType = $cmbOrderType.Text
    $OrderType = StringStripWS($OrderType, $STR_STRIPALL)

    Select
        Case $OrderType = "Normalbestellung"

            MsgBox(0, "Belegart", "Normalbestellung")

        Case $OrderType = "Rahmenbestellung"

            MsgBox(0, "Belegart", "Rahmenbestellung")

    EndSelect

EndSelect

Next

Next

EndFunc

;-Sub Main-----
Func Main()

    While 1
        CheckTAC()
        Sleep(1000)
    Wend

EndFunc

;-Main-----
Main()

;-End-----

```

Examples - Autolt - Copy Table Data to Clipboard

```
'-Open SE16 with table TADIR-----
session.findById("wnd[0]/tbar[0]/okcd").text = "/nse16"
session.findById("wnd[0]").sendVKey 0
session.findById("wnd[0]/usr/ctxtDATABROWSE-TABLENAME").text = "TADIR"
session.findById("wnd[0]/tbar[1]/btn[7]").press
session.findById("wnd[0]").sendVKey 8

'-Select two columns-----
session.findById("wnd[0]/usr/cntlGRID1/shellcont/shell").selectColumn("OBJ_NAME")
session.findById("wnd[0]/usr/cntlGRID1/shellcont/shell").selectColumn("SRCSYSTEM")

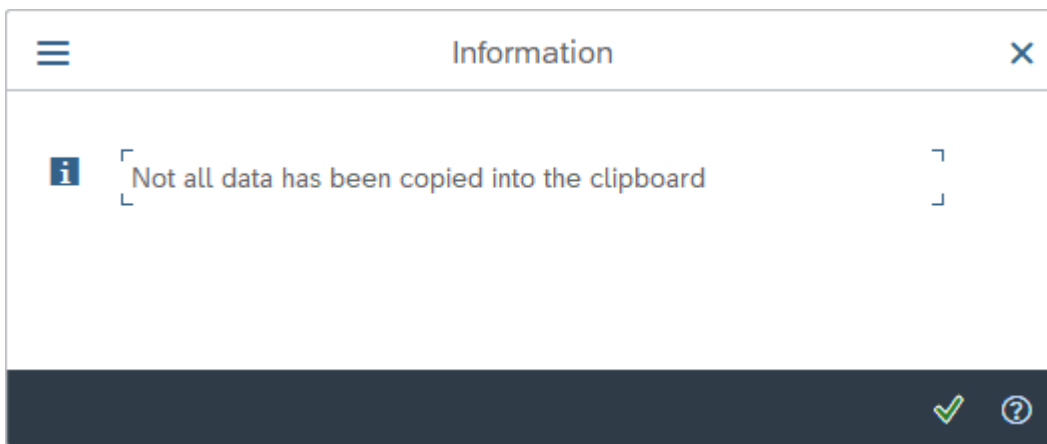
'-Bring SAP window to front-----
Set oAutoIt = WScript.CreateObject("AutoItX3.Control")
oAutoIt.WinActivate session.findById("wnd[0]").Text

'-Set the focus to the table-----
session.findById("wnd[0]/usr/cntlGRID1/shellcont/shell").setFocus

oAutoIt.Sleep 250

'-Send keys Ctrl + C to copy the content to the clipboard-----
oAutoIt.Send "{CTRLDOWN}c{CTRLUP}"
```

If too much data is to be copied, a message is displayed.



Hint: With VBScript SendKeys command is it not possible to send the keystrokes Ctrl + C to the SAP control.

Testing

Operating System	SAP GUI	Programming Language	Version	Available in Standard	Additional Library Necessary	Architecture	Recording	Playback
Windows 10 21H2 19044.1566 x64	SAP GUI for Windows 7.70 Compilation 1 Patch 5 x86	PowerShell Windows	5.1.19041.1320	✓	none	x64	✓	✓
						x86		✓
		PowerShell Core	7.2.1	✗	none	x64	✓	✓
			7.2.2				✓	✓
		C# 5 dotNET Framework	4.8.4084	✓	none	x64	✓	✓
						x86		✓
		C# 10 dotNET SDK	6.0.2	✗	none	x64	✓	✓
		Visual Basic 2012 dotNET Framework	14.8.4084	✓	none	x64	✓	✓
						x86		✓
		Windows Script Host VBScript	5.812	✓	none	x64	✓	✓
						x86		✓
		Autolt	3.3.14.5	✗	none	x64	✓	✓
						x86		✓
Windows 11 21H2 22000.527 x64	SAP GUI for Windows 7.70 Compilation 1 Patch 5 x86	PowerShell Windows	5.1.22000.581	✓	none	x64	✓	✓
						x86		✓
		C# 5 dotNET Framework	4.8.4161	✓	none	x64	✓	✓
						x86		✓
		Visual Basic 2012 dotNET Framework	14.8.4161	✓	none	x64	✓	✓
						x86		✓
		Windows Script Host VBScript	5.812	✓	none	x64	✓	✓
						x86		✓

Tracker.ini

Programming Language	Architecture	Section	Keyword	Comment
PowerShell Windows	x64	ProgramConfiguration	EditorExternals1	%WINDIR%\sysnative\WindowsPowerShell\v1.0\powershell_ise.exe
		ScriptingEngines	PowerShell	%WINDIR%\sysnative\WindowsPowerShell\v1.0\powershell.exe
	x86	ProgramConfiguration	EditorExternals1	%WINDIR%\SysWOW64\WindowsPowerShell\v1.0\powershell_ise.exe
		ScriptingEngines	PowerShell	%WINDIR%\SysWOW64\WindowsPowerShell\v1.0\powershell.exe
PowerShell Core	x64	ScriptingEngines	PowerShellCore	Set path to pwsh.exe
Autolt	x64	ScriptingEngines	Autolt	Set path to AutoIt3_x64.exe
	x86	ScriptingEngines	Autolt	Set path to AutoIt3.exe
Python	x86	ScriptingEngines	Python	Set path to python.exe

Java Shell	x64	ScriptingEngines	JShell	Set path to JShell.exe and add path to JaCoB.dll to PATH environment variable.
------------	-----	------------------	--------	--

Tracker.*2exe.cmd (dotNET)

C# dotNET Framework	x64	%WINDIR%\Microsoft.NET\Framework64\v4.0.30319\cs.c.exe /target:winexe /platform:x64 /out:Tracker_RunScript.cs.exe /reference:"C:\Windows\Microsoft.NET\Framework64\v4.0.30319\Microsoft.VisualBasic.dll" Tracker_RunScript.cs
	x86	%WINDIR%\Microsoft.NET\Framework\v4.0.30319\csc.exe /target:winexe /platform:x86 /out:Tracker_RunScript.cs.exe /reference:"C:\Windows\Microsoft.NET\Framework\v4.0.30319\Microsoft.VisualBasic.dll" Tracker_RunScript.cs
C# dotNET	x64	
Visual Basic dotNET Framework	x64	%WINDIR%\Microsoft.NET\Framework64\v4.0.30319\vb.c.exe /target:winexe /platform:x64 /out:Tracker_RunScript.vb.exe Tracker_RunScript.vb
	x86	%WINDIR%\Microsoft.NET\Framework\v4.0.30319\vb.c.exe /target:winexe /platform:x86 /out:Tracker_RunScript.vb.exe Tracker_RunScript.vb

Windows Script Host (VBScript)

Windows Script Host	x64	Start console with %WINDIR%\System32\cmd.exe
		Command echo %PROCESSOR_ARCHITECTURE% delivers AMD64
		%WINDIR%\System32\wscript.exe [Path\]Tracker_RunScript.vbs
	x86	Start console with %WINDIR%\SysWOW64\cmd.exe
		Command echo %PROCESSOR_ARCHITECTURE% delivers x86
		%WINDIR%\SysWOW64\wscript.exe [Path\]Tracker_RunScript.vbs

Requirements

- Operating system Windows® 7 or higher.
- Full standard installation of SAP® GUI for Windows® 7.40 or higher.
- Activated SAP® GUI Scripting on the presentation and application server.

Trademarks

- SAP, NetWeaver, NetWeaver Business Client (NWBC), ABAP and SAP GUI Scripting are registered trademarks of SAP AG
- Windows, Visual Basic for Application (VBA), VBScript, Script Host, PowerShell and Edge are registered trademarks of Microsoft, C# and VB.NET are product names from Microsoft
- Autolt and AutoltX is property of Jonathan Bennet and the Autolt team
- Java and JShell are registered trademarks of Oracle
- Jacob is property of Joe Freeman
- Google, Chrome and Android are registered trademarks of Google
- Python is a registered trademark of Python Software Foundation
- UiPath is a registered trademark of UiPath SRL
- Blue Prism is a registered trademark of Blue Prism Ltd.

Scintilla

Scripting Tracker uses Scintilla.

License for Scintilla and SciTE

Copyright 1998-2003 by Neil Hodgson <neilh@scintilla.org>

All Rights Reserved

Permission to use, copy, modify, and distribute this software and its documentation for any purpose and without fee is hereby granted, provided that the above copyright notice appear in all copies and that both that copyright notice and this permission notice appear in supporting documentation.

Neil Hodgson disclaims all warranties with regard to this software, including all implied warranties of merchantability and fitness, in no event shall Neil Hodgson be liable for any special, indirect or consequential damages or any damages whatsoever resulting from loss of use, data or profits, whether in an action of contract, negligence or other tortious action, arising out of or in connection with the use or performance of this software.

AutoltX

Scripting Tracker contains AutoltX.

It is allowed to reproduce and distribute an unlimited number of copies of AutoltX either in whole or in part. Scripting Tracker contains the help file of AutoltX and therewith a copy of all copyright and [trademark](#) notices.

AutoltX is [available at Autolt homepage](#).

Jacob

Scripting Tracker contains Jacob (Java COM Bridge).

It is allowed to distribute copies of the library as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and distribute a copy of this license along with the library. Scripting Tracker contains the license text.

Jacob is [available at GibHub](#).

Contact

WebSite: tracker.stschnell.de

Support: mail@stschnell.de

Guarantee exclusion

No guarantee for the actuality, correctness, completeness or quality of Scripting Tracker is taken. Liability claims, which refer to damage by the use or not-use of this program and its libraries, are principally impossible. This program and its libraries are provided 'as-is', without any express or implied warranty.