Scripting Tracker

Development Tool for SAP® GUI Scripting

Version 5.00

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 Analyser 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 Analyser 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, Visual Basic Script®, AutoIt, Python and JShell for Java™. Also you can record and edit the dotNET languages C# and VB.NET, to use this code sequences inside RPA platforms.

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

Benefit

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 Tracker brings here more transparency. With the recorder of Scripting Tracker you have full visual control about the generated SAP® GUI Scripting code. You see in the 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. The standard uses PowerShell® Windows. With Scripting Tracker it is possible to use, beside PowerShell® Windows, PowerShell ® Core, VBScript® of Windows Scripting Host (WSH), AutoIt, Python and Java Shell (JShell). You can record and replay sources of this engines. Also it is possible to record C# and VB.NET code. 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 7.20 patch level 9, also the support of the SAP® GUI Scripting tools - look at OSS note 1633639. The Analyser of Scripting Tracker is an alternative. It shows all scripting objects in a clearly arranged tree and, after a selection of one object, a lot of technical details or its position on the SAP® GUI with a red frame.

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.

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.

In Headwords

Scripting Trackers recorder has the same functionality as SAP® GUI Scripting recorder to record and replay SAP® GUI Scripts.

In addition
• an integrated basic editor,
• full visual control about the generated code,
• the possibility to enrich the code automatically with additional information and
• beside PowerShell® Windows, support of different scripting engines like PowerShell® Core, VBScript® of Windows Scripting Host (WSH), AutoIt, Python and JShell for Java™.
• *Scripting Trackers Analyser* has the same functionality as SAP® GUI Scripting wizard to identify SAP® GUI Scripting objects.

In addition
• it works with Windows® 7 and higher,
• shows all scripting objects in a well arranged tree and
• shows a lot of technical details of the scripting object.

Scripting Tracker optimizes your development process with SAP® GUI Scripting and offers new horizons of integration between an SAP® system and your presentation server. It brings you one step closer in independence and increases your efficiency with SAP® GUI Scripting.

Enable SAP® GUI Scripting

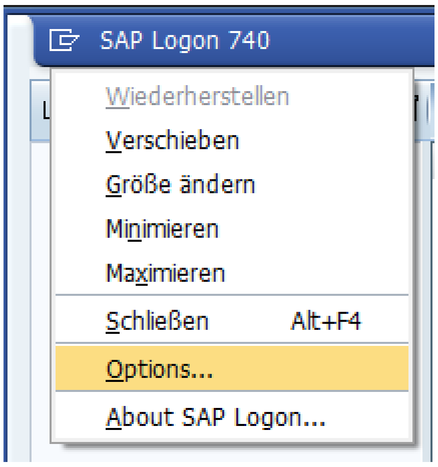
Scripting Tracker uses SAP® GUI and PowerShell® Windows or PowerShell® Core, Windows Scripting Host (WSH) VBScript®, AutoIt scripting, Python engine or JShell. Also it can create code for C# or VB.NET programming language. So it is necessary to enable SAP® GUI Scripting on the application and presentation server. Also it is necessary to enable PowerShell® or VBScript® on the presentation server, or you can install and use PowerShell® Core, AutoIt or Python scripting engine as well as JShell. PowerShell® Windows is in a normal case available on any Windows system, but it is necessary to set the execution policy.

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

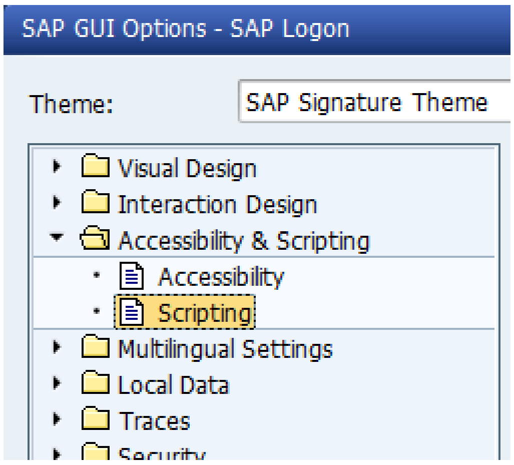
On Presentation Server
On Application Server

Enable SAP® GUI Scripting on Presentation Server

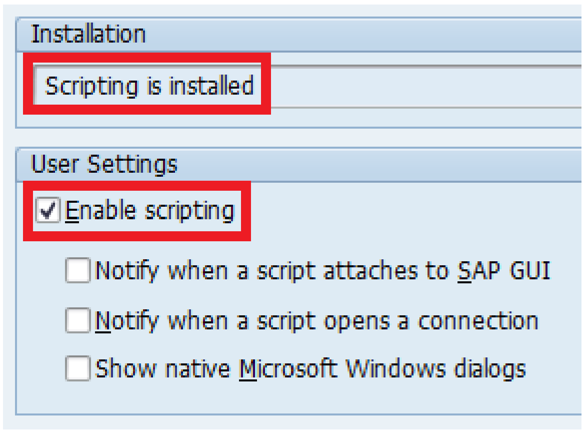
- Choose the menu item Options... from the system menu of the SAP® Logon.



- Choose the node Scripting.



- The Scripting must be installed and activated.



Installation

Scripting is installed

User Settings

☑ Enable scripting

☐ Notify when a script attaches to SAP GUI

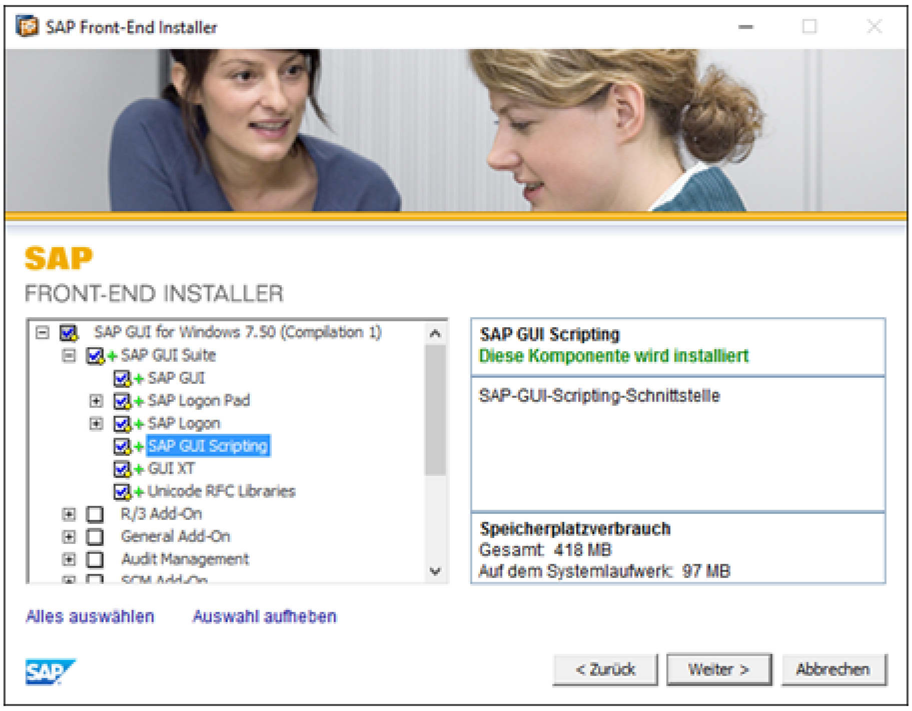
☐ Notify when a script opens a connection

☐ Show native Microsoft Windows dialogs

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

Hint: It is better 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.

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.



- ☑ SAP GUI for Windows 7.50 (Compilation 1)
  - ☑ SAP GUI Suite
    - ☑ SAP GUI
    - ☑ SAP Logon Pad
    - ☑ SAP Logon
    - ☑ SAP GUI Scripting
    - ☑ GUI XT
    - ☑ Unicode RFC Libraries
  - ☐ R/3 Add-On
  - ☐ General Add-On
  - ☐ Audit Management
  - ☐ SCM Add-On

SAP GUI Scripting
Diese Komponente wird installiert

SAP-GUI-Scripting-Schnittstelle

Speicherplatzverbrauch
Gesamt: 418 MB
Auf dem Systemlaufwerk: 97 MB

Registry Entries of the SAP® GUI Scripting

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

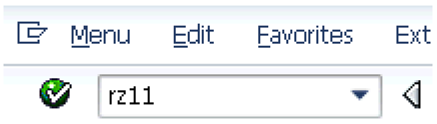
Notify when a script attaches to SAP GUI
HKEY_CURRENT_USER\Software\SAP\SAPGUI Front\SAP Frontend Server\Security\WarnOnAttach
from type REG_DWORD, Default: 1, 0 = inactive, 1 = active

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

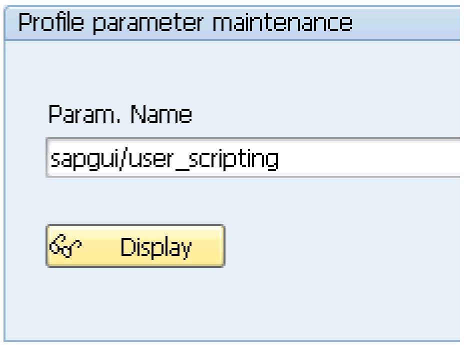
Show native MS Windows dialogs
HKEY_CURRENT_USER\Software\SAP\SAPGUI Front\SAP Frontend Server\Scripting\ShowNativeWinDlgs
from type REG_DWORD, Default: 0, 0 = inactive, 1 = active

Enable SAP® GUI Scripting on Application Server

- Use the transaction code RZ11 in the ok field.

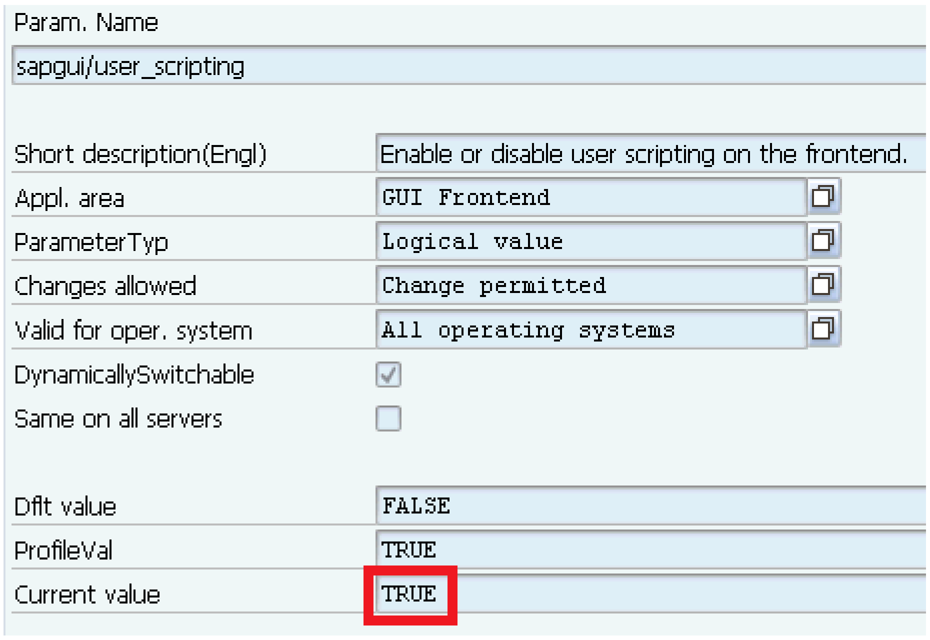


- Use the profile parameter sapgui/user_scripting and press the Display button.



Param. Name
sapgui/user_scripting

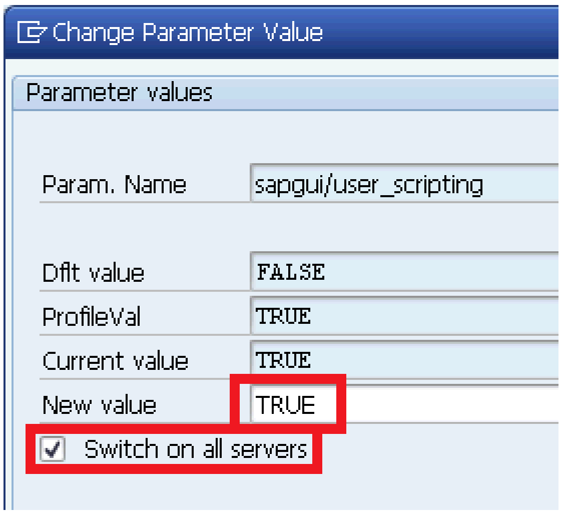
- The current value must be TRUE.



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	✓
Same on all servers	
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

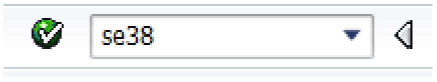
Parameter values

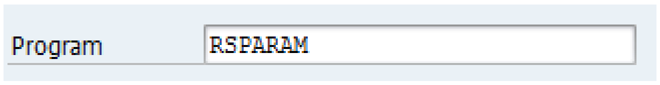
Param. Name	sapgui/user_scripting
Dflt value	FALSE
ProfileVal	TRUE
Current value	TRUE
New value	TRUE

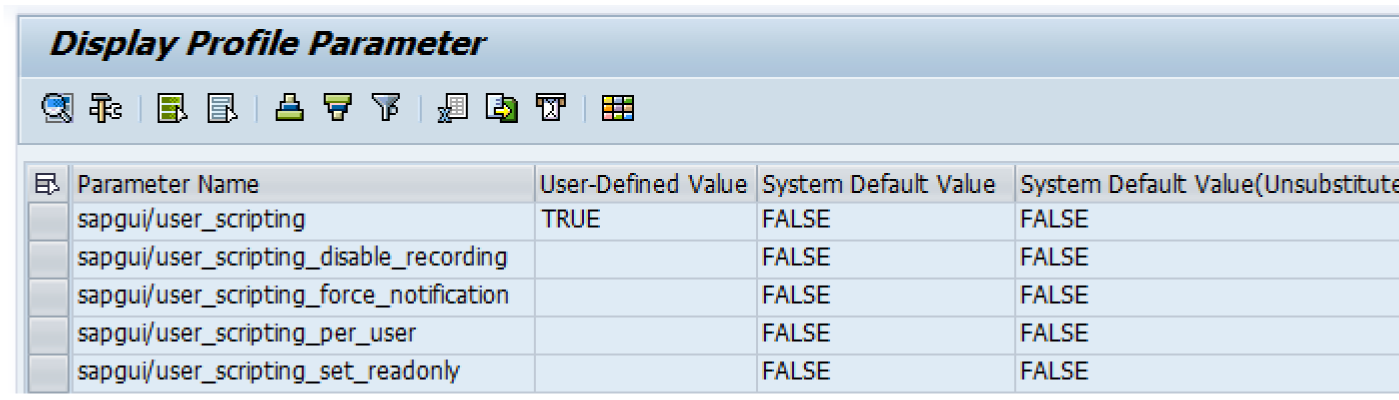
[✓] Switch on all servers

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

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


se38


Program RSPARAM


Display Profile Parameter

Parameter Name	User-Defined Value	System Default Value	System Default Value(Unsubstitute
sapgui/user_scripting	TRUE	FALSE	FALSE
sapgui/user_scripting_disable_recording		FALSE	FALSE
sapgui/user_scripting_force_notification		FALSE	FALSE
sapgui/user_scripting_per_user		FALSE	FALSE
sapgui/user_scripting_set_readonly		FALSE	FALSE

- 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 priviledges 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)

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).
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 AutoIt help...	Optional menu item. If the keyword AutoItHelp in the ScriptingEngines section of the preference file is set to the AutoIt 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.

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 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.

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. If you press the shift button, you add a few lines of code and information.
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	Reloades 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.
▶	Executes the script from the editor.

Playback script	
Record SAP® GUI Script	Records SAP® activities to a script in the editor.
Stop script process	Stops the executing of the scripting process.
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 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 C#	Records the script as C# code.
Use VB.NET	Records the script as VB.NET code.
Use Windows® Scripting Host	Records and executes the script as VBScript® file via Windows® Scripting Host (WSH).
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 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

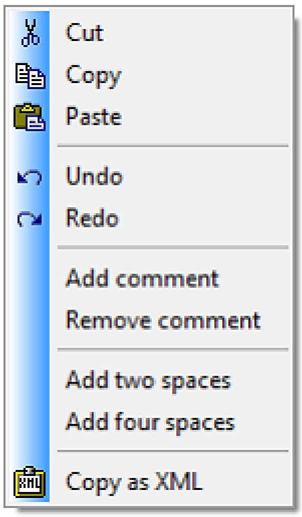
	JShell engine in the section ScriptingEngines of the Tracker.ini file. Use the keyword JShell.
+ Additional information in source	Enriches the source with information comment lines about the transaction, title, dynpro - program name and screen number - and the session number.
SAP® session	Chooses the 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®.
☐ Add SAP® standard code in source	If the checkbox is enabled Tracker enriches the external source file with standard code.
About recorder module...	Shows an additional window with information about recorder module of Tracker.

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	Copies the selected text to the clipboard and converts it to XML

	`& to &amp;`
	`< to &lt;`
	`> to &gt;`
	`" to &quot;`
	`' to &apos;`

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.

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)

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

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	Delivers a hierarchy of collections with information about the state of an object. The parameter InnerObject may be used to secify 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. The following OpCodes are used: • 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.

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.
-

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.

SAP GUI Scripting User Settings

Shows a few information about the customization of SAP GUI Scripting. 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. SAPGUISERVER if one or more instances exists, and in brackets the number of instances.

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.

Robotic Process Automation (RPA)

UiPath Integration Scenarios

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

PowerShell
VBScript
Python
AutoIt

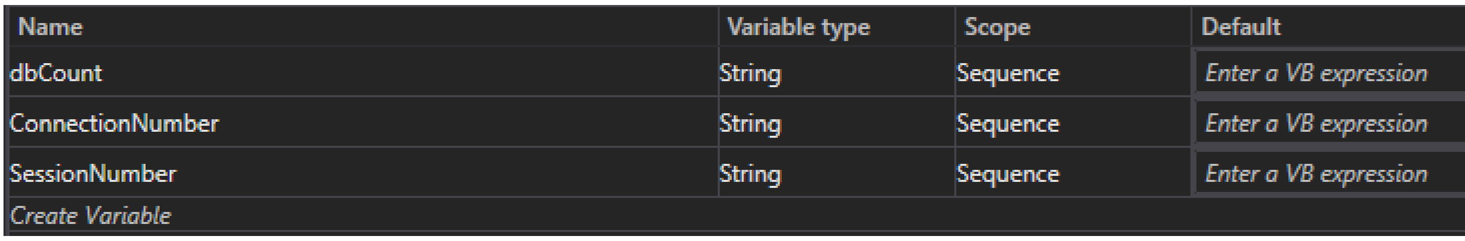
UiPath Integration Scenario (PowerShell)

PowerShellScript

```
#-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);
```

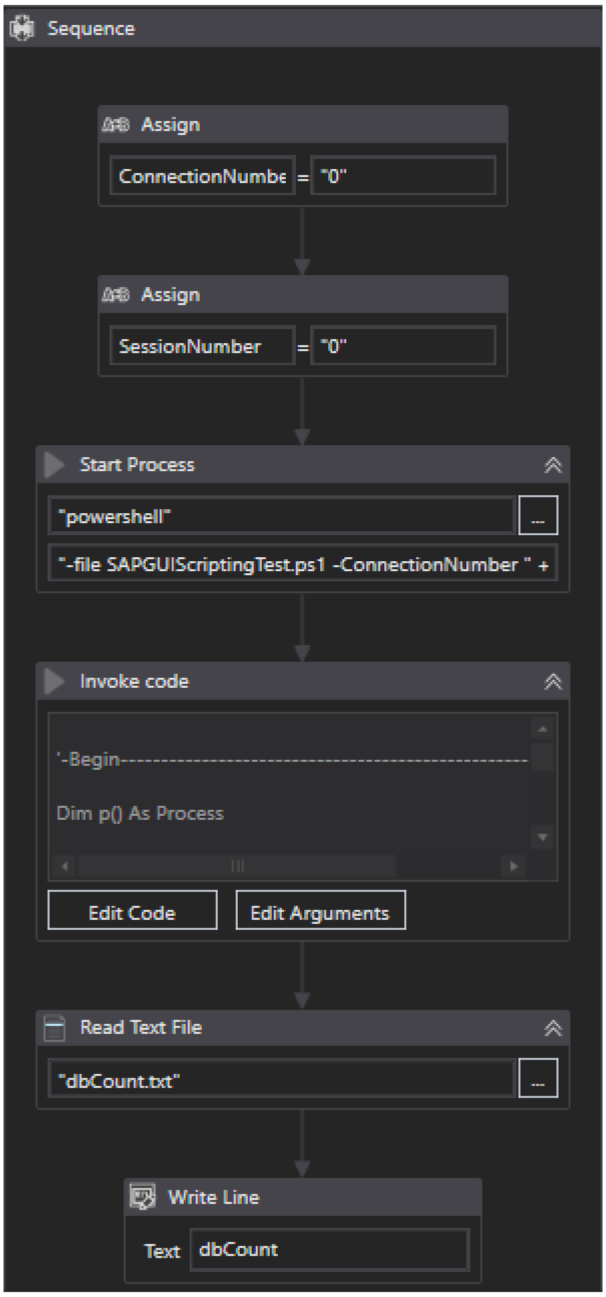
```
$ID = Invoke-Method $session "findById" @("wnd[0]");
Invoke-Method $ID "sendVKey" @(3);

Free-Object $SapGuiAuto;

Set-Content dbCount.txt $dbCount;

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

Variables in UiPath



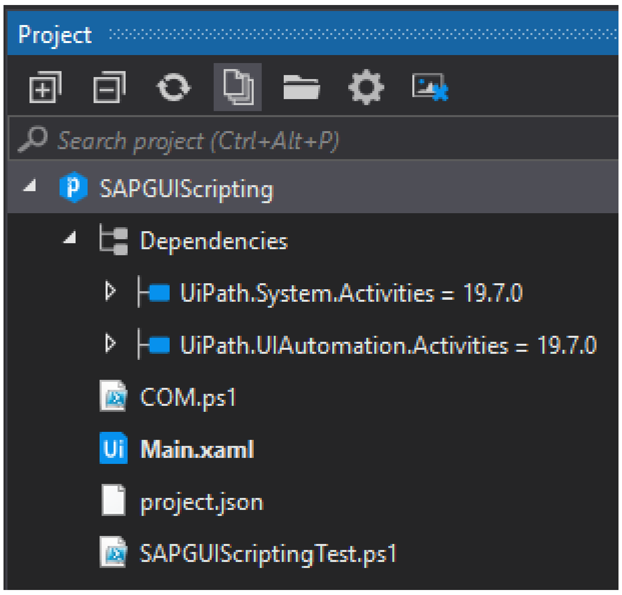
Name	Variable type	Scope	Default
dbCount	String	Sequence	*Enter a VB expression*
ConnectionNumber	String	Sequence	*Enter a VB expression*
SessionNumber	String	Sequence	*Enter a VB expression*
Create Variable			

Sequence in UiPath



```
ConnectionNumbe = "0"
SessionNumber = "0"
"powershell"
"-file SAPGUIScriptingTest.ps1 -ConnectionNumber " +
'-Begin-----------------------------------------------
Dim p() As Process
"dbCount.txt"
dbCount
```

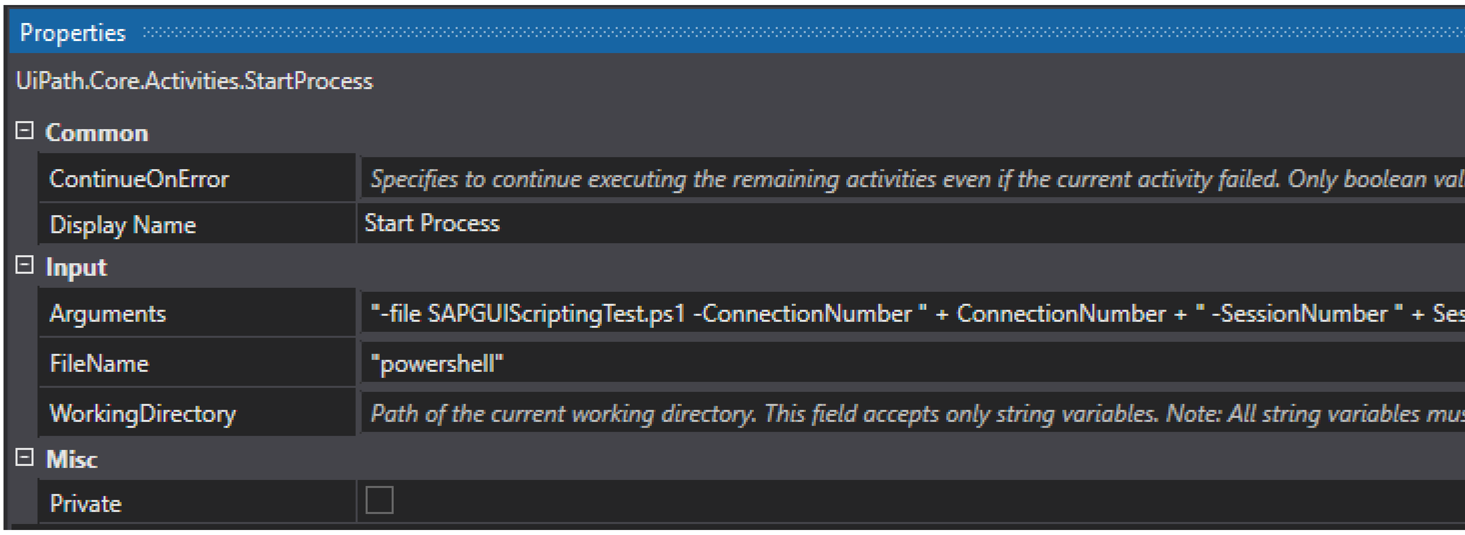
Hint: Store the PowerShell script file and the include into your project folder.


Project

Search project (Ctrl+Alt+P)

SAPGUIScripting
- Dependencies
  - UiPath.System.Activities = 19.7.0
  - UiPath.UIAutomation.Activities = 19.7.0
- COM.ps1
- Main.xaml
- project.json
- SAPGUIScriptingTest.ps1

Properties of Start-Process activity


Properties

UiPath.Core.Activities.StartProcess

Common	
ContinueOnError	Specifies to continue executing the remaining activities even if the current activity failed. Only boolean val
Display Name	Start Process
Input	
Arguments	"-file SAPGUIScriptingTest.ps1 -ConnectionNumber " + ConnectionNumber + " -SessionNumber " + Ses
FileName	"powershell"
WorkingDirectory	Path of the current working directory. This field accepts only string variables. Note: All string variables mus
Misc	
Private	☐

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.

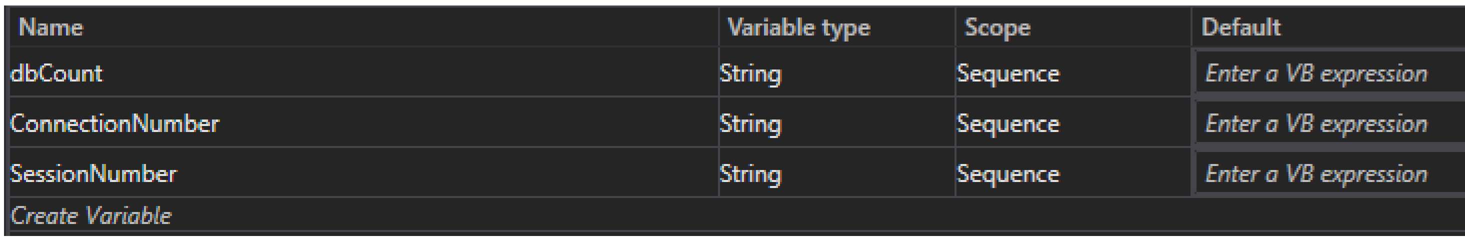
UiPath Integration Scenario (VBScript)

VBScript

```
'-Begin-----------------------------------------------------------------

Option Explicit

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

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-------------------------------------------------------------------
```

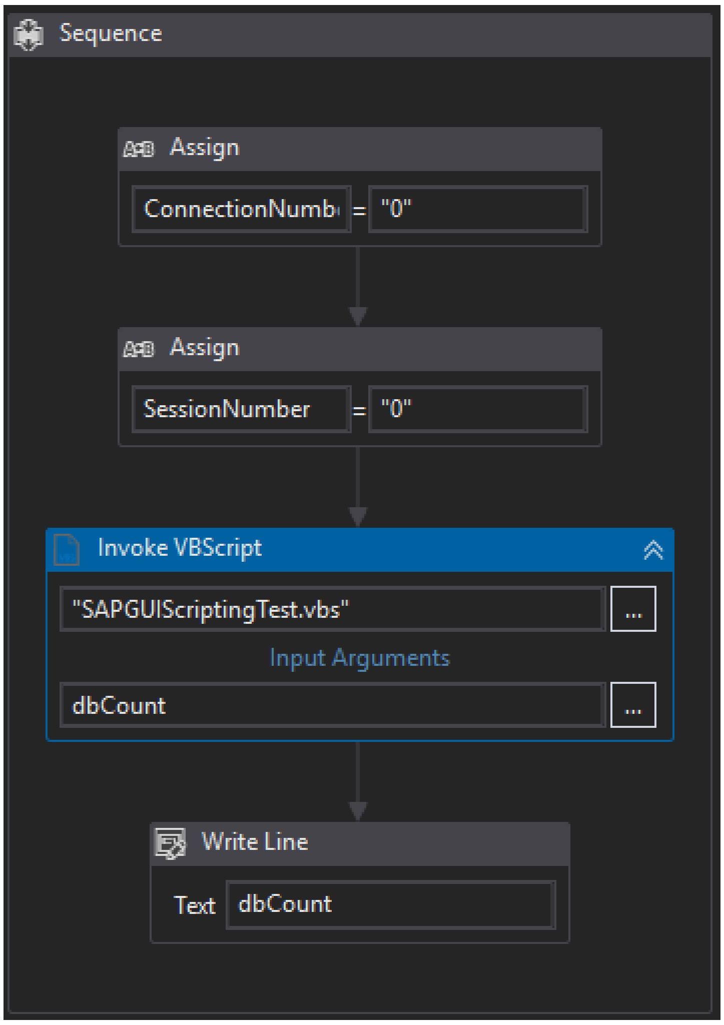
Hint: If you use Option Explicit you must define the arguments of Invoke VBScript activity too.

Variables in UiPath

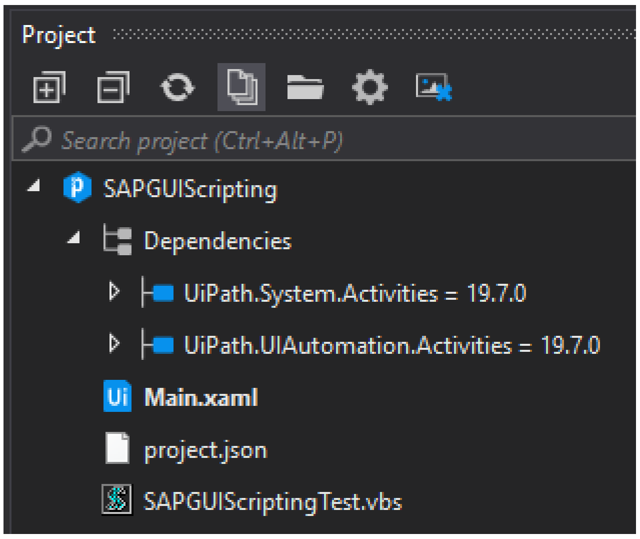


Name	Variable type	Scope	Default
dbCount	String	Sequence	*Enter a VB expression*
ConnectionNumber	String	Sequence	*Enter a VB expression*
SessionNumber	String	Sequence	*Enter a VB expression*
Create Variable			

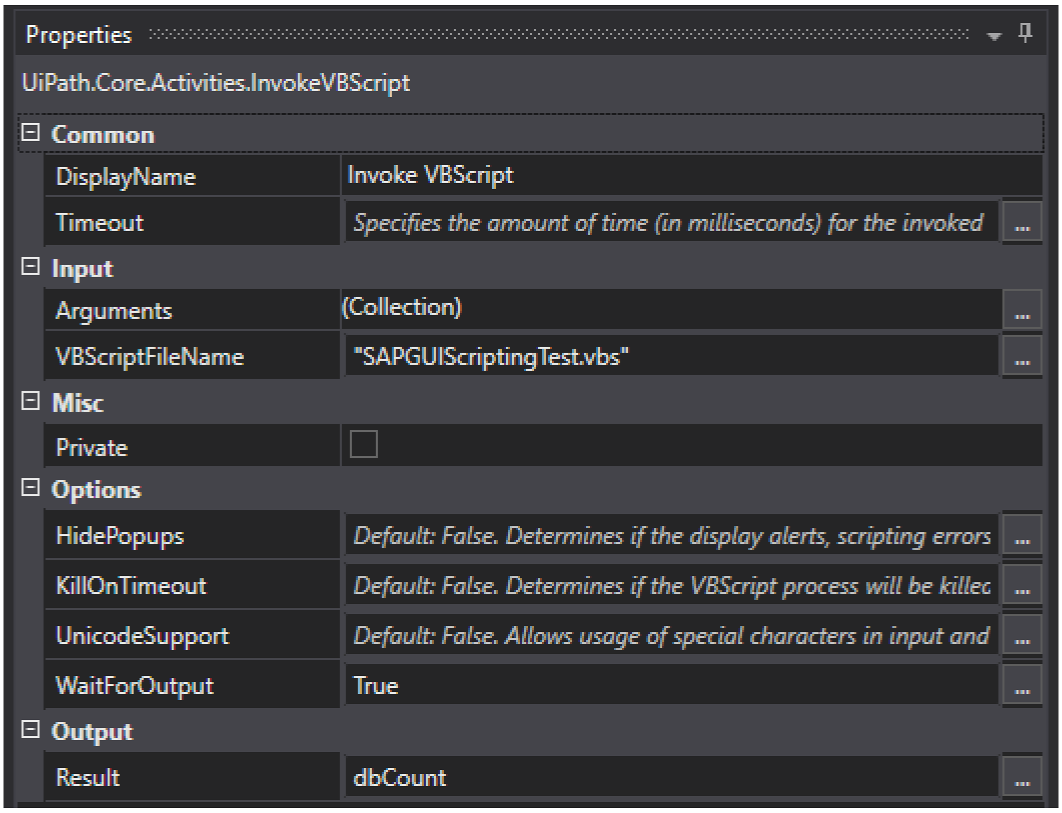
Sequence in UiPath



Hint: Store the VBScript file into your project folder.


Project
Search project (Ctrl+Alt+P)
SAPGUIScripting
Dependencies
UiPath.System.Activities = 19.7.0
UiPath.UIAutomation.Activities = 19.7.0
Main.xaml
project.json
SAPGUIScriptingTest.vbs

Properties of Invoke VBScript activity

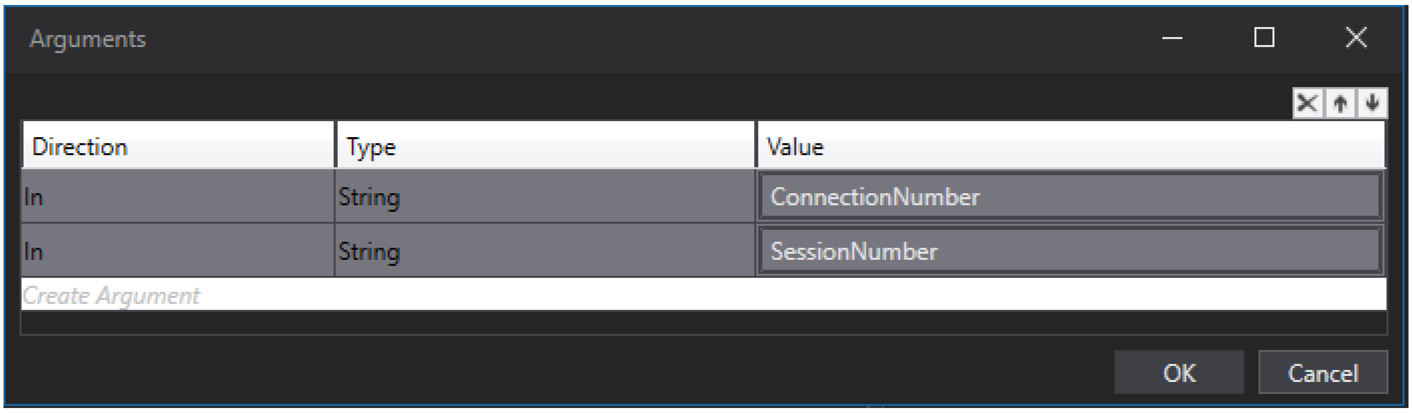


Properties

UiPath.Core.Activities.InvokeVBScript

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

Arguments of Invoke VBScript activity



Arguments

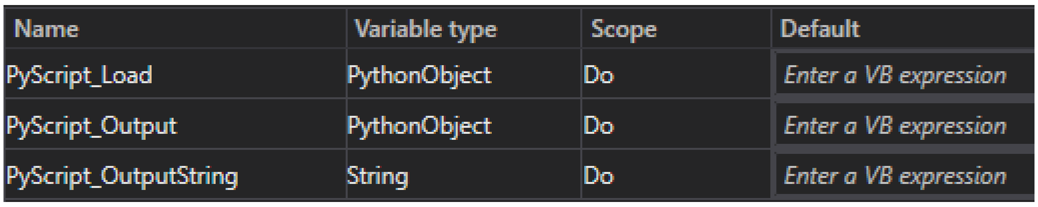
Direction	Type	Value
In	String	ConnectionNumber
In	String	SessionNumber

UiPath Integration Scenario (Python)

PythonScript

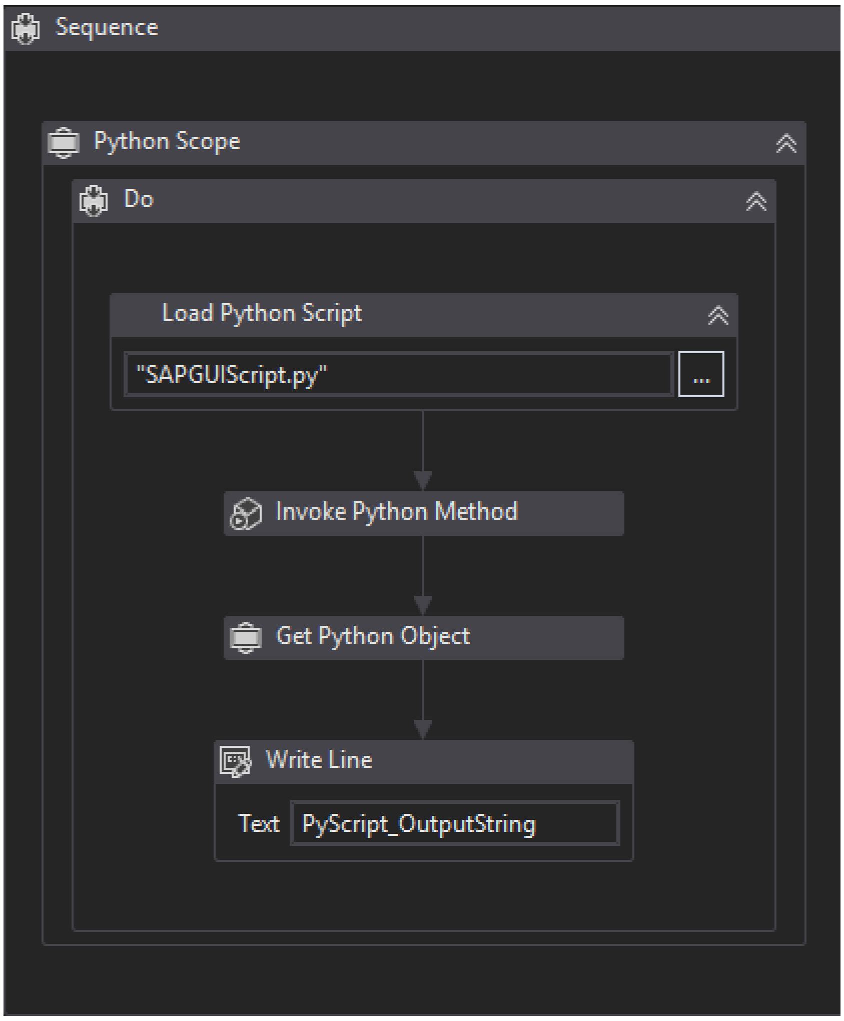
```
#-Begin-----------------------------------------------------------------

#-Includes--------------------------------------------------------------
import sys, win32com.client

#-Function test---------------------------------------------------------
def test():

  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

    connection = application.Children(0)
    if not type(connection) == win32com.client.CDispatch:
      application = None
      SapGuiAuto = None
      return

    session = connection.Children(0)
    if not type(session) == win32com.client.CDispatch:
      connection = None
      application = None
      SapGuiAuto = None
      return

    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)
    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)

  except:
    return sys.exc_info()[0]

  finally:
    session = None
    connection = None
    application = None
    SapGuiAuto = None

  return dbCount

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

Variables in UiPath

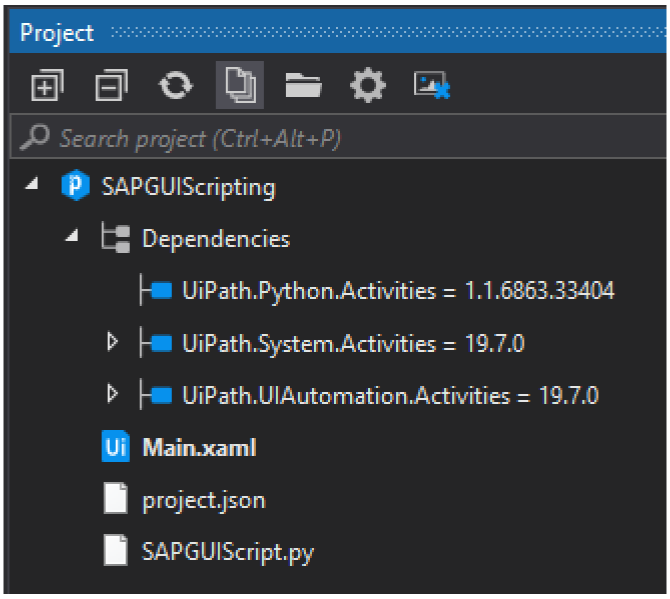


Name	Variable type	Scope	Default
PyScript_Load	PythonObject	Do	Enter a VB expression
PyScript_Output	PythonObject	Do	Enter a VB expression
PyScript_OutputString	String	Do	Enter a VB expression

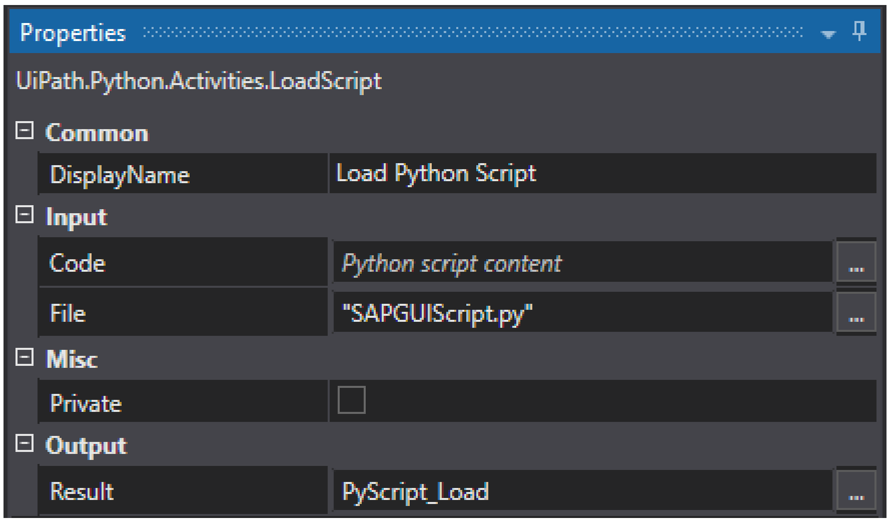
Sequence in UiPath



Hint: Store the Python script file into your project folder.


Project
Search project (Ctrl+Alt+P)
SAPGUIScripting
Dependencies
UiPath.Python.Activities = 1.1.6863.33404
UiPath.System.Activities = 19.7.0
UiPath.UIAutomation.Activities = 19.7.0
Main.xaml
project.json
SAPGUIScript.py

Properties of Load Python Script

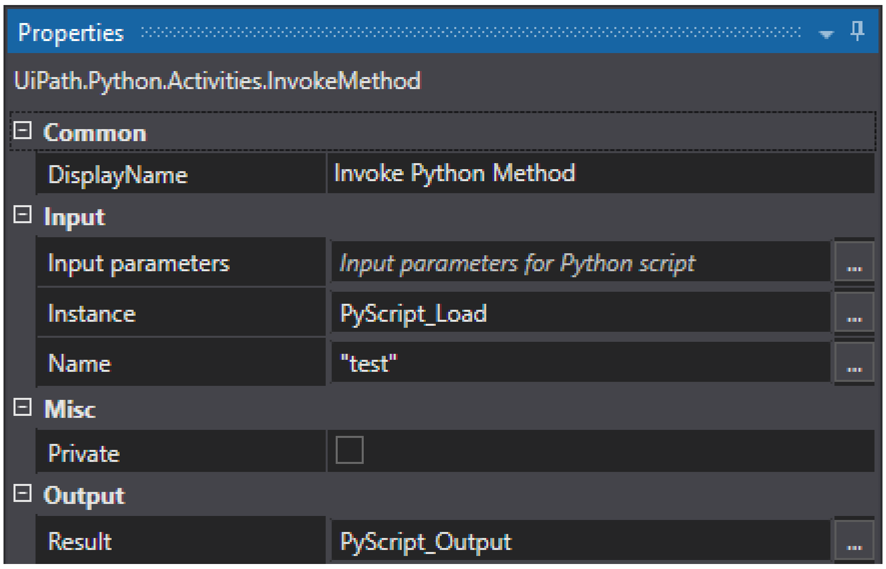


Properties

UiPath.Python.Activities.LoadScript

Common	
DisplayName	Load Python Script
Input	
Code	*Python script content*
File	"SAPGUIScript.py"
Misc	
Private	☐
Output	
Result	PyScript_Load

Properties of Invoke Python Method

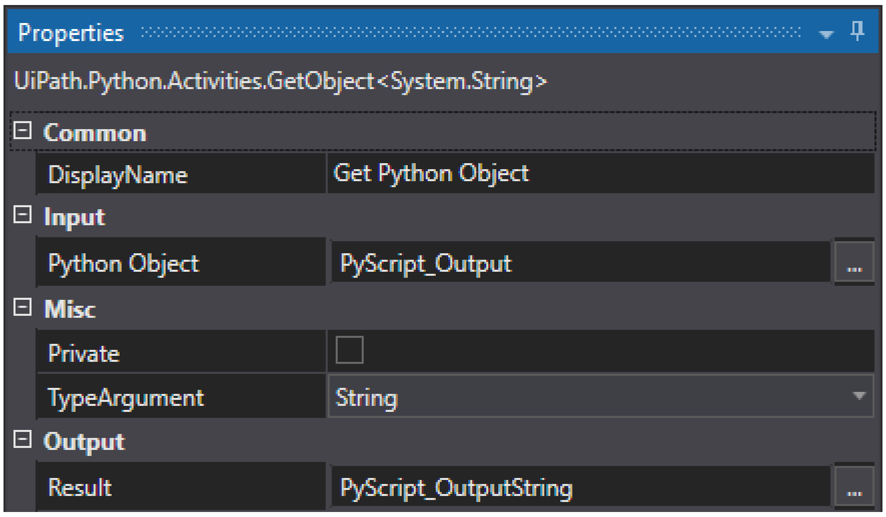


Properties

UiPath.Python.Activities.InvokeMethod

Common	
DisplayName	Invoke Python Method
Input	
Input parameters	*Input parameters for Python script*
Instance	PyScript_Load
Name	"test"
Misc	
Private	☐
Output	
Result	PyScript_Output

Properties of Get Python Object



Properties

UiPath.Python.Activities.GetObject<System.String>

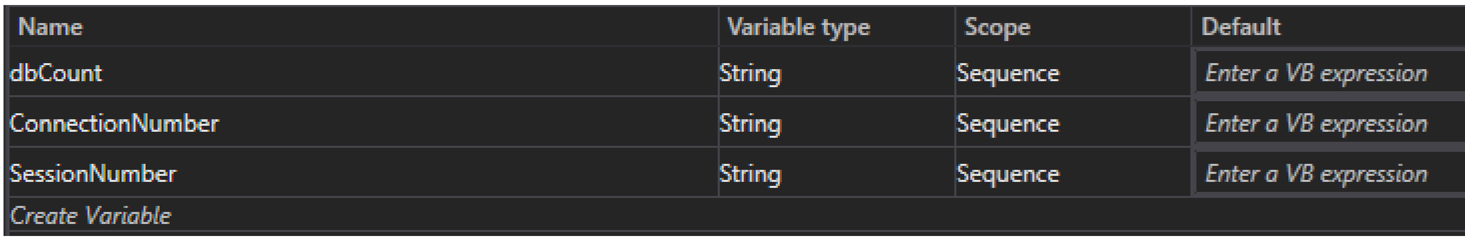
Common	
DisplayName	Get Python Object
Input	
Python Object	PyScript_Output
Misc	
Private	☐
TypeArgument	String
Output	
Result	PyScript_OutputString

UiPath Integration Scenario (AutoIt)

AutoItScript

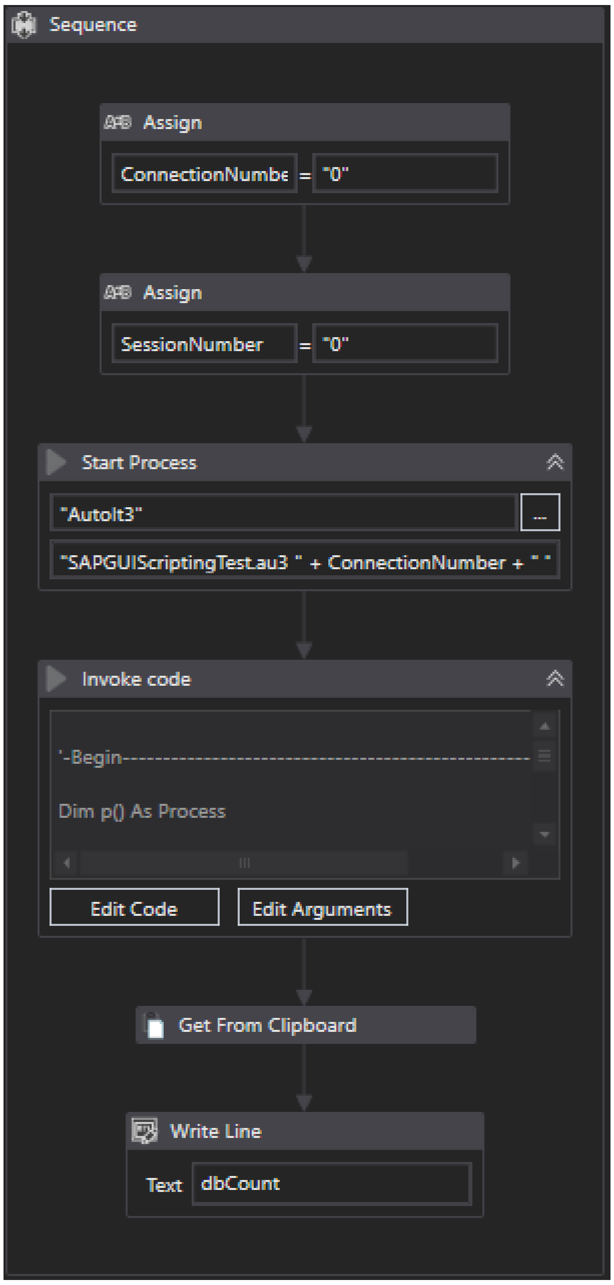
```
;-Begin-----------------------------------------------------------------

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



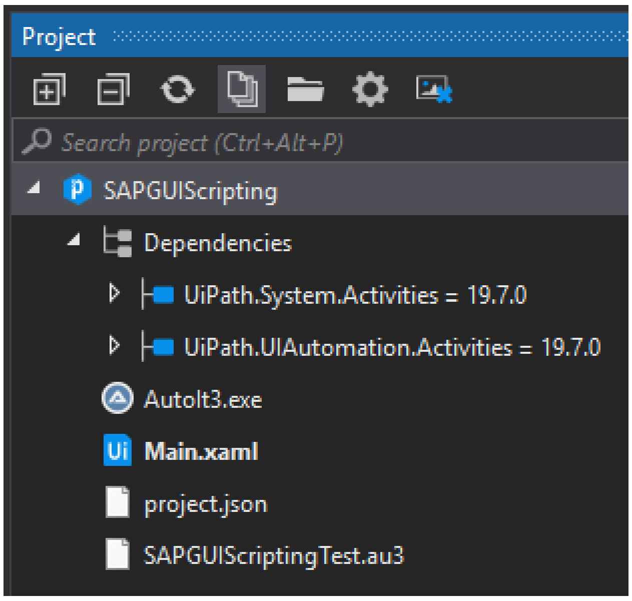
Name	Variable type	Scope	Default
dbCount	String	Sequence	Enter a VB expression
ConnectionNumber	String	Sequence	Enter a VB expression
SessionNumber	String	Sequence	Enter a VB expression
Create Variable			

Sequence in UiPath

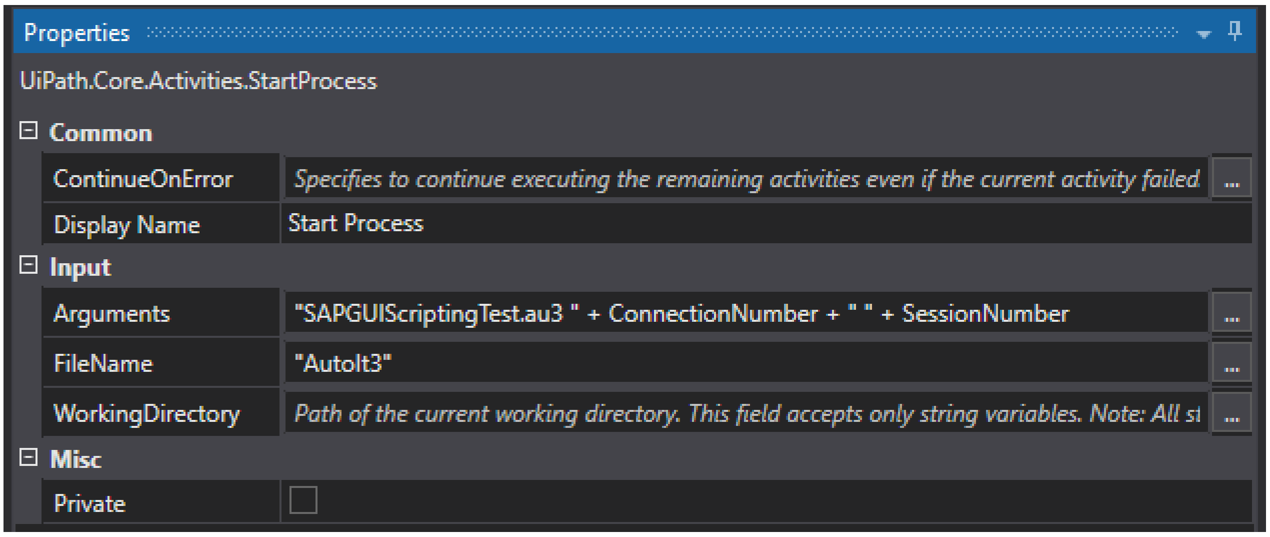


```
'-Begin-----------------------------------------------

Dim p() As Process
```

Hint: Store the AutoIt script file and the AutoIt3.exe into your project folder.


Project
Search project (Ctrl+Alt+P)
SAPGUIScripting
Dependencies
UiPath.System.Activities = 19.7.0
UiPath.UIAutomation.Activities = 19.7.0
AutoIt3.exe
Main.xaml
project.json
SAPGUIScriptingTest.au3

Properties of Start Process activity



Properties

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	"AutoIt3"
WorkingDirectory	*Path of the current working directory. This field accepts only string variables. Note: All st*
Misc	
Private	☐

Hint: For the Inter Process Communication (IPC) with the AutoIt 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 AutoIt script the content of the clipboard is read.

Blue Prism Integration Scenarios

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

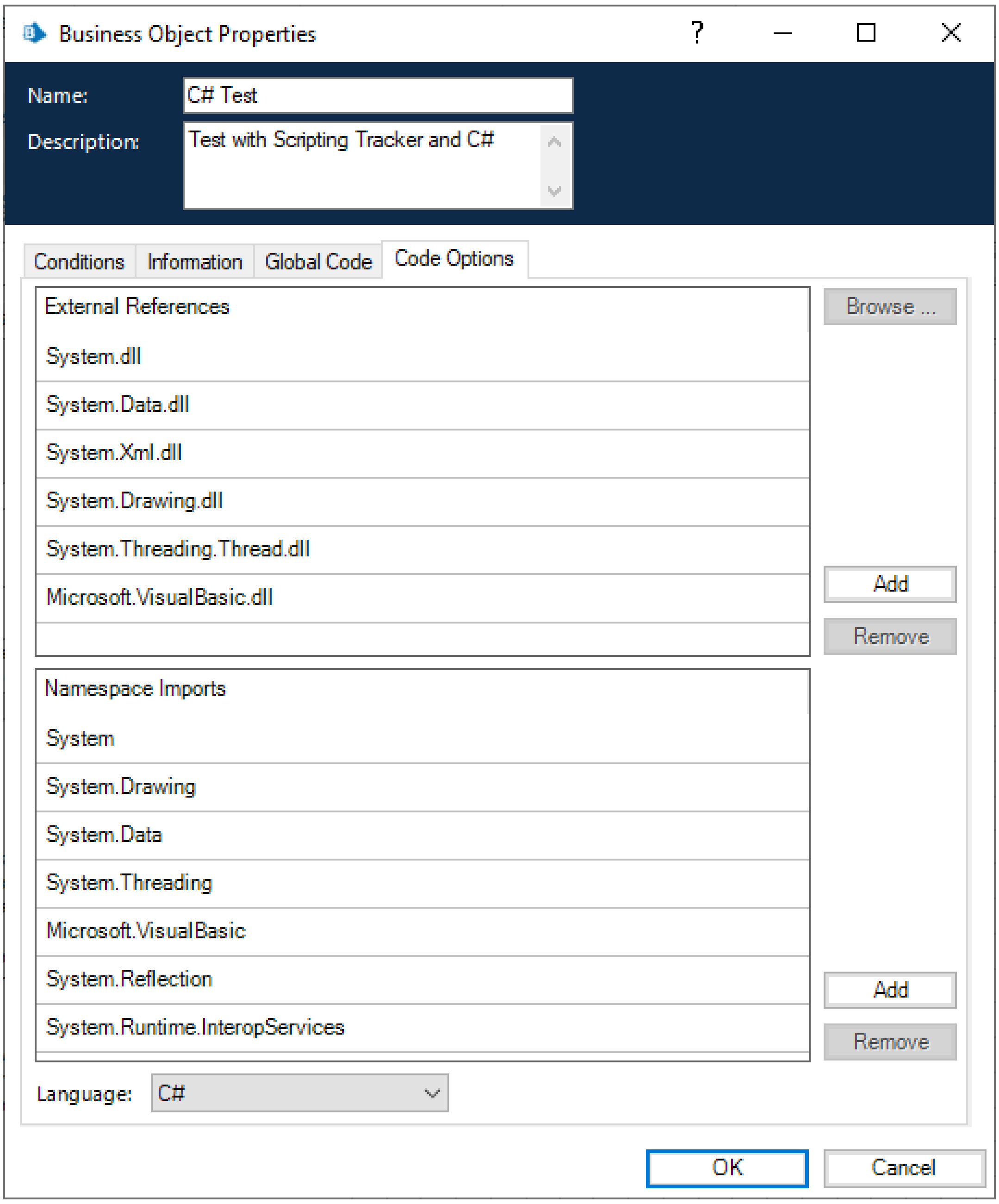
VB.NET
C#

Blue Prism Integration Scenario (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 copy the code sequence between Sub Main() and End Sub into your code stage.

Blue Prism Integration Scenario (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.



Name: C# Test

Description: Test with Scripting Tracker and C#

External References
System.dll
System.Data.dll
System.Xml.dll
System.Drawing.dll
System.Threading.Thread.dll
Microsoft.VisualBasic.dll

Namespace Imports
System
System.Drawing
System.Data
System.Threading
Microsoft.VisualBasic
System.Reflection
System.Runtime.InteropServices

Language: C#

Record your activities with C#, add the code snippet Blue Prism and move your recorded code to the correct position.

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.

- 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® Scripting 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 recording
  Use 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.
- 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.

UI Object	Type	Transaction Code / Script
GUIABAPEditor	GuiShell SubType ABAPEditor	SE80
GUIApoGrid		
GUIApplication	GuiApplication	
GUIBarChart	GuiShell SubType BarChart	SE38 - BARCOCX1
GUIBox	GuiBox	SE38 - DEMO_DYNPRO_SPLITTER_CON TROL
GUIButton	GuiButton	GUIBIBS SE38 - DEMO_DYNPRO_PUSH_BUTTON
GUICalendar	GuiShell SubType Calendar	SE38 - SAPCALENDAR_DEMO_BEGIN
GUIChart	GuiShell SubType Chart	SE38 - GFW_PROG_TUTORIAL SE38 - GFW_PROG_PIE
GUICheckBox	GuiCheckBox	GUIBIBS
GUICollection	GuiCollection	
GUIColorSelector	GuiShell	SE38 - DEMO_COLORSEL

	SubType ColorSelector	
GUIComboBox	GuiComboBox	GUIBIBS SE38 - DEMO_DYNPRO_DROPDOWN_LISTBOX
GUIComboBoxControl		
GUIComboBoxEntry		
GUIComponent		
GUIComponentCollection		
GUIConnection	GuiConnection	
GUIContainer		
GUIContainerShell	GuiContainerShell	SE38 - GRAPHICS_GUI_CE_DEMO
GUIContextMenu		
GUICTextField		
GUICustomControl	GuiCustomControl	SE38 - RSDEMO_CUSTOM_CONTROL
GUIDialogShell		
GUIEAIViewer2D		
GUIEAIViewer3D		
GUIFrameWindow		
GUIGOSShell	GuiShell SubType ToolBar	SGOSTEST SE38 - GOS_TOOLBOX_TEST
GUIGraphAdapt		

GUIGridView	GuiShell SubType GridView (ALV-Grid)	SE80 - Package SLIS Programs BCALV_GRID*
GUIHTMLViewer	GuiShell SubType HTMLViewer	SE38 - DEMO_CREATE_HTML_MODERN SE38 - SAPHTML_DEMO1
GUIInputFieldControl		
GUILabel	GuiLabel	GUIBIBS
GUIMainWindow	GuiMainWindow	SESSION_MANAGER GUIBIBS
GUIMap		
GUIMenu	GuiMenu	SESSION_MANAGER
GUIMenuBar		
GUIMessageWindow		
GUIModalWindow	GuiModalWindow	SE38 - DEMO_CALCULATOR_MODERN1 SE37 - POPUP_TO_INFORM
GUINetChart		
GUIOfficeIntegration	GuiShell SubType OfficeIntegration	SE38 - SAPRDEMO_FORM_INTERFACE
GUIOkCodeField	GuiOkCodeField	SESSION_MANAGER
GUIPasswordField	GuiPasswordField	SESSION_MANAGER
GUIPicture	GuiShell SubType Picture	SE38 - SAP_PICTURE_DEMO
GUIRadioButton	GuiRadioButton	GUIBIBS
GUISapChart		

GUIScrollbar	GuiScrollbar	GUIBIBS
GUIScrollContainer	GuiScrollContainer	SE38 - DEMO_DYNPRO_SPLITTER_CONTROL
GUISession	GuiSession	
GUISessionInfo	GuiSessionInfo	GuiSessionInfo
GUIShell		
GUISimpleContainer	GuiSimpleContainer	SE38 - DEMO_DYNPRO_SUBSCREENS
GUISplit		
GUISplitterContainer	GuiSplitterContainer	SE38 - DEMO_DYNPRO_SPLITTER_CONTROL
GUIStage		
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
GUITextField	GuiTextField	GUIBIBS
GUITitleBar	GuiTitleBar	
GUIToolBar	GuiShell SubType ToolBar	SE38 - BCALV_TREE_DND_MULTIPLE
GUIToolBarControl		
GUITree	GuiShell SubType Tree	SE80 - Package SLIS Programs BCALV_TREE*
GUIUserArea	GuiUserArea	
GUIUtils	GuiUtils	GuiUtils
GUIVComponent		
GUIVContainer		
GUIVSwitchTarget		

```
#-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-------------------------------------------------------------------
```

```
#-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-------------------------------------------------------------------
```

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	GUIChart		
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	GUICTextField	ctxt	I/O
22	GUICustomControl	cntl	CUCTR
23	GUIDialogShell	shellcont	
24	GUIEAIViewer2D		
25	GUIEAIViewer3D		

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	GUISapChart		
46	GUIScrollbar		
47	GUIScrollContainer	ssub	SUBSC
48	GUISession	ses	
49	GUISessionInfo		
50	GUIShell	shell	
51	GUISimpleContainer	sub	

52	GUISplit		
53	GUISplitterContainer	splc	SPCTR
54	GUIStage		
55	GUIStatusbar	sbar	
56	GUIStatusbarLink		
57	GUIStatusPane	pane	
58	GUITab	tabp	PUSH
59	GUITableColumn		
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	GUIVSwitchTarget		

```
"-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 tabname, fieldname, as4local, rollname, checktable,
        inttype, intlen
        FROM dd03l
        INTO CORRESPONDING FIELDS OF @ls_dd03l
        WHERE tabname = @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------------------------------------------------------------------
```

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: https://sourceforge.net/projects/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.

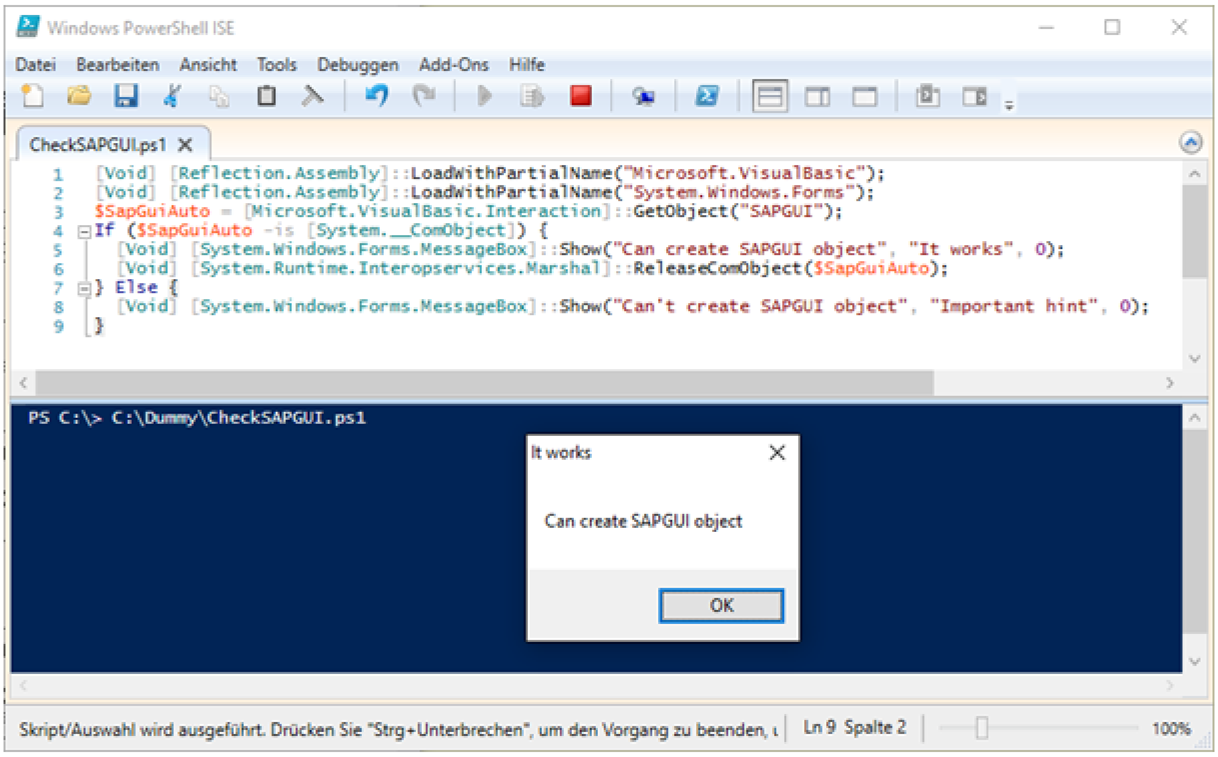
- You can find the web driver specification here
  https://w3c.github.io/webdriver/

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.



```
[Void] [Reflection.Assembly]::LoadWithPartialName("Microsoft.VisualBasic");
[Void] [Reflection.Assembly]::LoadWithPartialName("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);
}
```

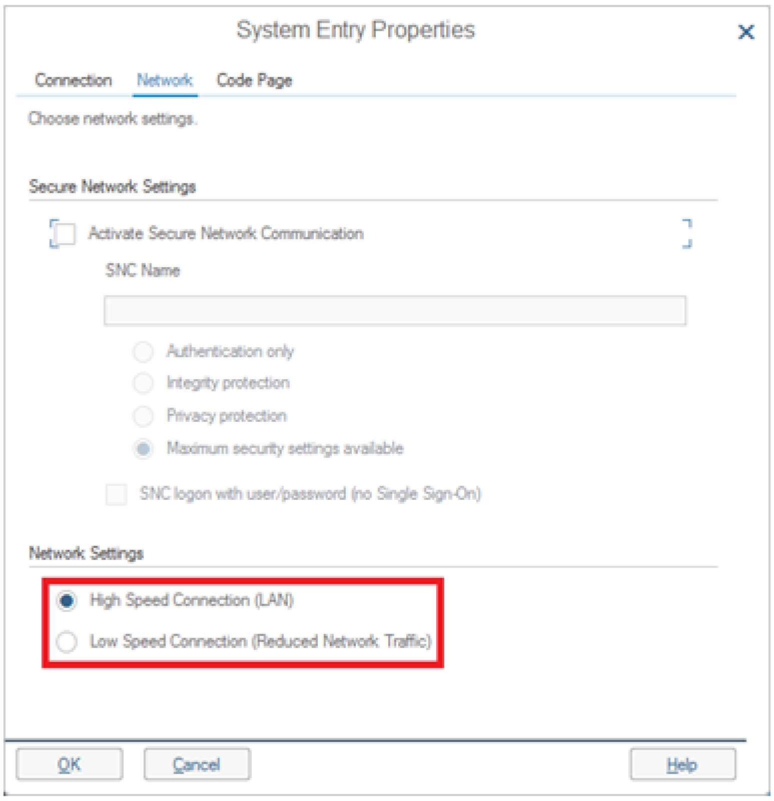
```
PS C:\> C:\Dummy\CheckSAPGUI.ps1
```

Can create SAPGUI object

Skript/Auswahl wird ausgeführt. Drücken Sie "Strg+Unterbrechen", um den Vorgang zu beenden, ι   Ln 9 Spalte 2   100%

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.


System Entry Properties

Connection | Network | Code Page

Choose network settings.

Secure Network Settings

[ ] Activate Secure Network Communication

SNC Name

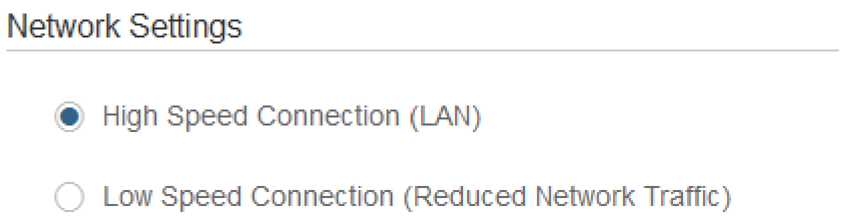
( ) Authentication only
( ) Integrity protection
( ) Privacy protection
(●) Maximum security settings available

[ ] SNC logon with user/password (no Single Sign-On)

Network Settings

(●) High Speed Connection (LAN)
( ) Low Speed Connection (Reduced Network Traffic)

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 loose 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]/shel
l").selectNode "          1"
session.findById("wnd[0]/shellcont/shell/shellcont[3]/shell/shellcont[2]/shel
l").nodeContextMenu "          1"
session.findById("wnd[0]/shellcont/shell/shellcont[3]/shell/shellcont[2]/shel
l").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]/shel
l").selectNode "          1"
session.findById("wnd[0]/shellcont/shell/shellcont[3]/shell/shellcont[2]/shel
l").nodeContextMenu "          1"
session.findById("wnd[0]/shellcont/shell/shellcont[3]/shell/shellcont[2]/shel
l").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
```

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:

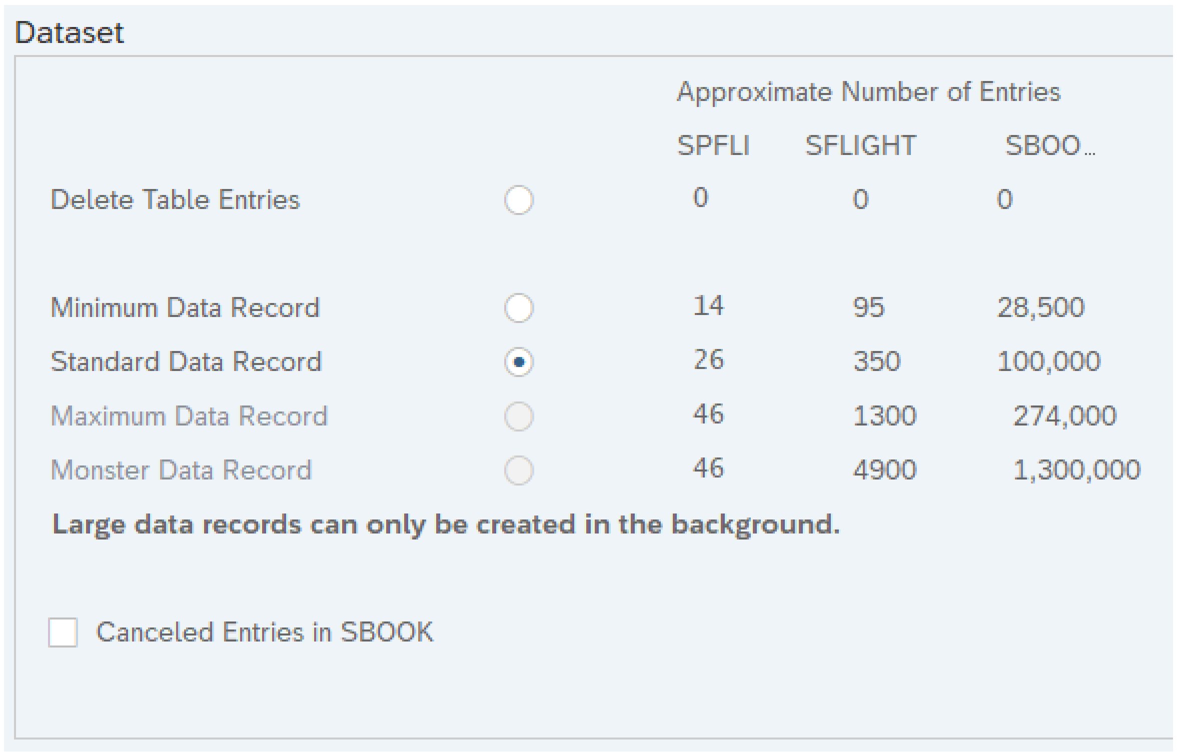
- SWDP_DEMO
- SWDP_DEMO_TUTORIALS

SFLIGHT

Generate Data
Add Records

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	○	0	0	0
Minimum Data Record	○	14	95	28,500
Standard Data Record	●	26	350	100,000
Maximum Data Record	○	46	1300	274,000
Monster Data Record	○	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:

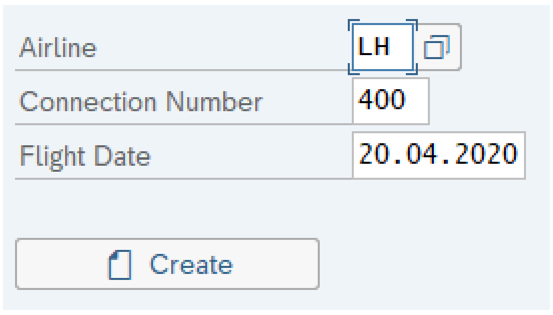
Number	Tablename	Description
1	SCARR	Airline
2	SCURX	Currency
3	SCURR	Exchange rates
4	SGEOCITY	Geographical position of a city
5	SAIRPORT	Airports
6	SCITAIRP	City-Airport assignment
7	SAPLANE	Plane
8	SCPLANE	Cargo plane
9	SCUSTOM	Flight customers
10	STRAVELAG	Travel agency
11	SBUSPART	Airline partner
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

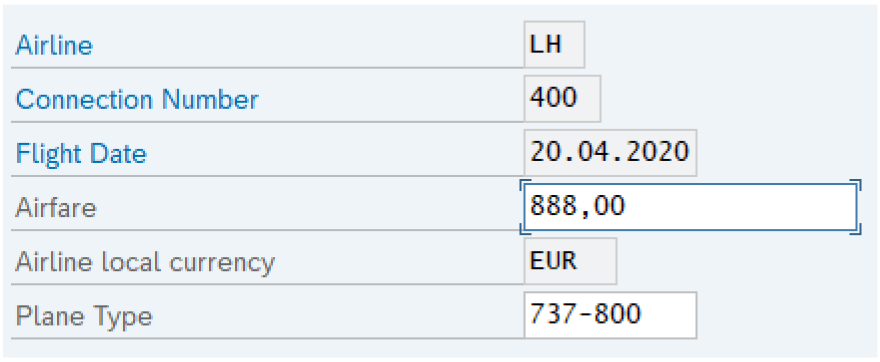
With the transaction code BC_GLOBAL_SFLGH_CREA, which calls the report SAPBC_GLOBAL_SFLIGHT_CREATE, it is possible to add additional records to the SFLIGHT table.

1. Select a flight and press Create.



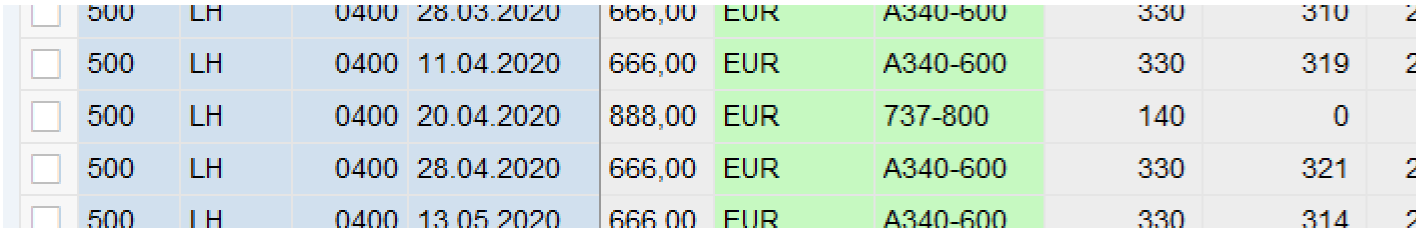
Airline	LH
Connection Number	400
Flight Date	20.04.2020

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.



☐	500	LH	0400	28.03.2020	666,00	EUR	A340-600	330	310	2
☐	500	LH	0400	11.04.2020	666,00	EUR	A340-600	330	319	2
☐	500	LH	0400	20.04.2020	888,00	EUR	737-800	140	0	
☐	500	LH	0400	28.04.2020	666,00	EUR	A340-600	330	321	2
☐	500	LH	0400	13.05.2020	666,00	EUR	A340-600	330	314	2

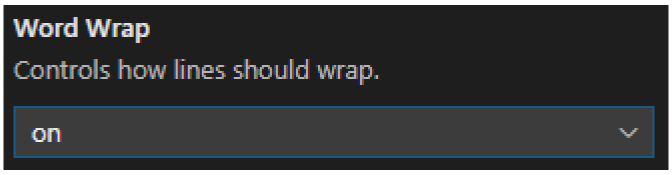
Visual Studio Code

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



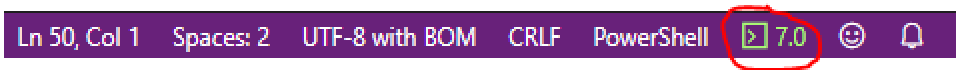
Word Wrap
Controls how lines should wrap.
on

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"
        }
    ]
```

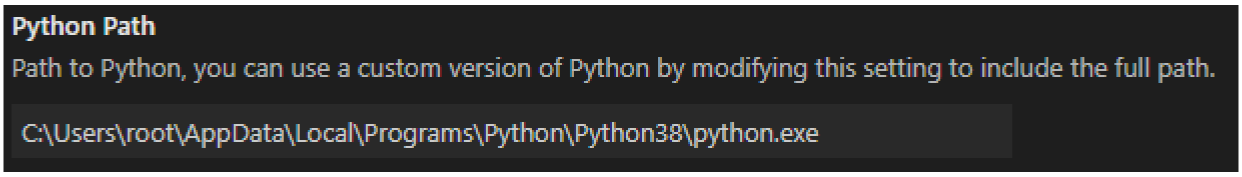
Now you have the possibility to switch between different PowerShell versions with a click in the statusbar of VSCode.



Ln 50, Col 1   Spaces: 2   UTF-8 with BOM   CRLF   PowerShell   7.0

Install Python Visual Studio extension. and install Python for Windows extension.

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



Python Path
Path to Python, you can use a custom version of Python by modifying this setting to include the full path.
C:\Users\root\AppData\Local\Programs\Python\Python38\python.exe

This adds to settings.json:

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

eCATT

```
***********************************************************************
*
* 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
    ).                                                    "#EC NOTEXT
    IF sy-subrc <> 0.

    ENDIF.

    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
        ).                                                "#EC NOTEXT
        IF sy-subrc <> 0.

        ENDIF.

      ENDLOOP.
      ENDIF.

    ENDLOOP.

"-End-------------------------------------------------------------------
```

SAP Demo Reports

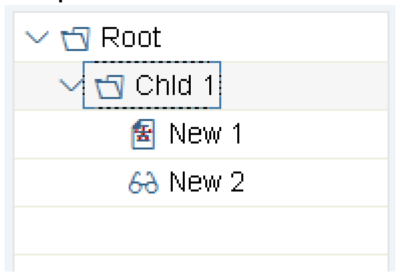
Tree

Tree

Development Class
SEU_TREE_MODEL

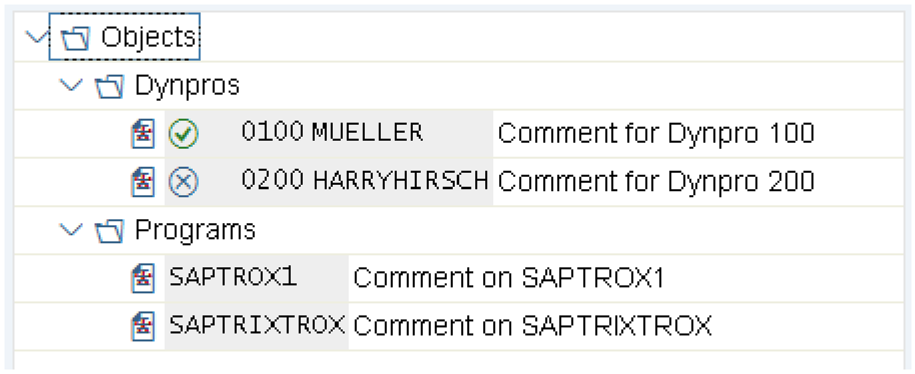
Reports
SAPSIMPLE_TREE_MODEL_DEMO
SAPTLIST_TREE_MODEL_DEMO
SAPTLIST_TREE_CONTROL_DEMO_HDR
SAPCOLUMN_TREE_MODEL_DEMO

Simple Tree



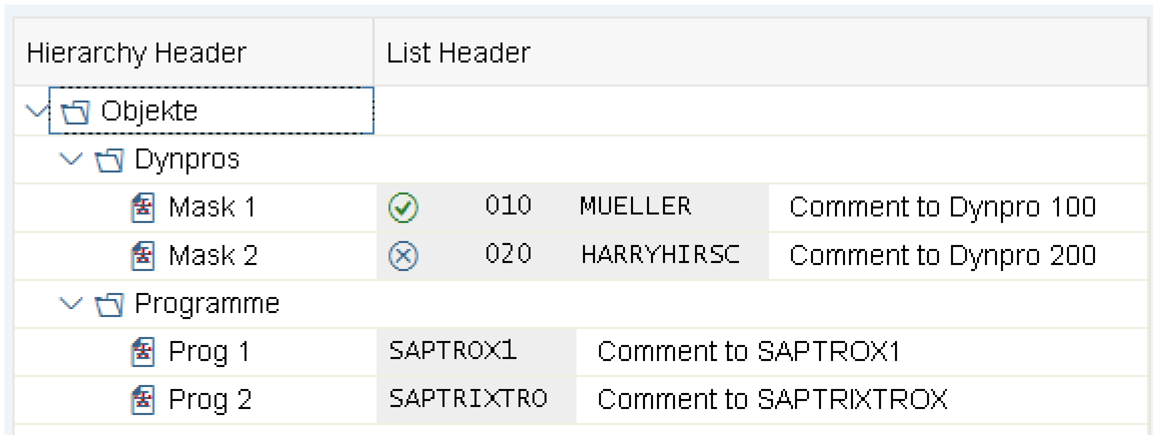
- Root
  - Chld 1
    - New 1
    - New 2

List Tree without header



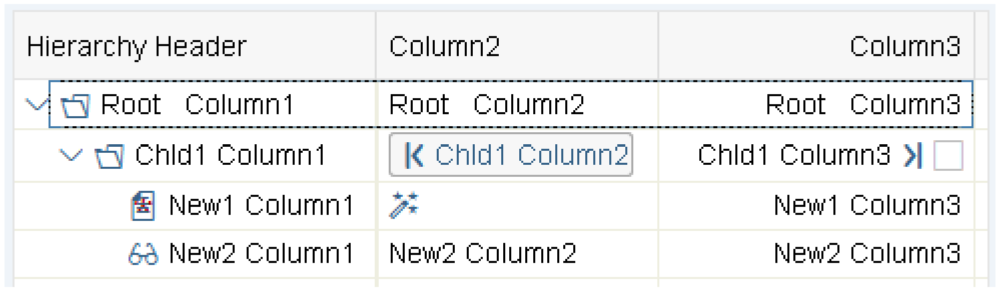
- Objects
  - Dynpros
    - ✓ 0100 MUELLER Comment for Dynpro 100
    - ⊗ 0200 HARRYHIRSCH Comment for Dynpro 200
  - Programs
    - SAPTROX1 Comment on SAPTROX1
    - SAPTRIXTROX Comment on SAPTRIXTROX

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
New1 Column1		New1 Column3
New2 Column1	New2 Column2	New2 Column3

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 conext 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 SAPGUISERVER.

  • 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 SAPGUISERVER 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 ABAP® in Eclipse

  • The correct entry in the running object table (ROT) for the SAP® GUI Scripting inside ABAP® in Eclipse is SAPGUISERVER.

- 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.

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
AutoIt	Path and name of the AutoIt engine.
AutoItHelp	Path and name of the AutoIt 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.

- `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 = 10
WindowPosY = 10
WindowPosWidth = 700
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.

Frames

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

PowerShell
WSH
VBA
AutoIt
Python
JScript

```
#-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 $SapGuiAuto
    Return
  }

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

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



  Free-Object $SapGuiAuto

}

#-Main------------------------------------------------------------------
Main

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

```
#-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) {
    $SapROTWr = New-Object -ComObject "SapROTWr.SapROTWrapper";
    $SapROTWr.GetROTEntry($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--------------------------------------------------------------------
```

```
#-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 activties 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-------------------------------------------------------------------
```

```
#-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:

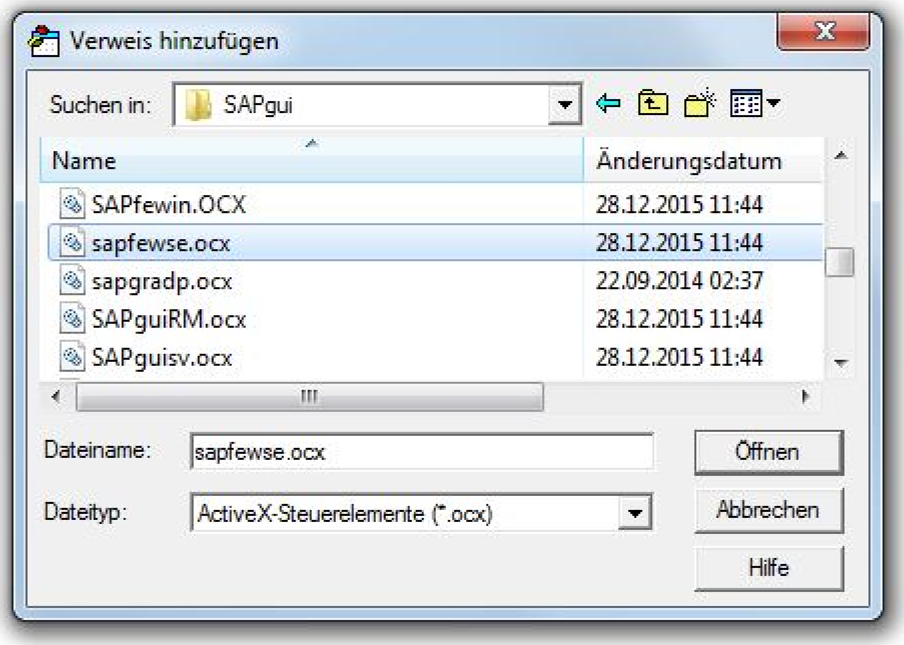
```
#-Begin-----------------------------------------------------------------

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-------------------------------------------------------------------
```

```
'-Begin-----------------------------------------------------------------

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

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

  Dim SapGuiAuto, application, connection, session

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

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

  Set connection = application.Children(0)
  If Not IsObject(connection) Then
    Set SapGuiAuto = Nothing
    Exit Sub
  End If

  Set session = connection.Children(0)
  If Not IsObject(session) Then
    Set SapGuiAuto = Nothing
    Exit Sub
  End If



  Set SapGuiAuto = Nothing

End Sub

'-Main------------------------------------------------------------------
Main

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

```
'-Begin-----------------------------------------------------------------

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

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

  Dim SapGuiAuto As Object
  Dim App As SAPFEWSELib.GuiApplication
  Dim connection As SAPFEWSELib.GuiConnection
  Dim session As SAPFEWSELib.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

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).



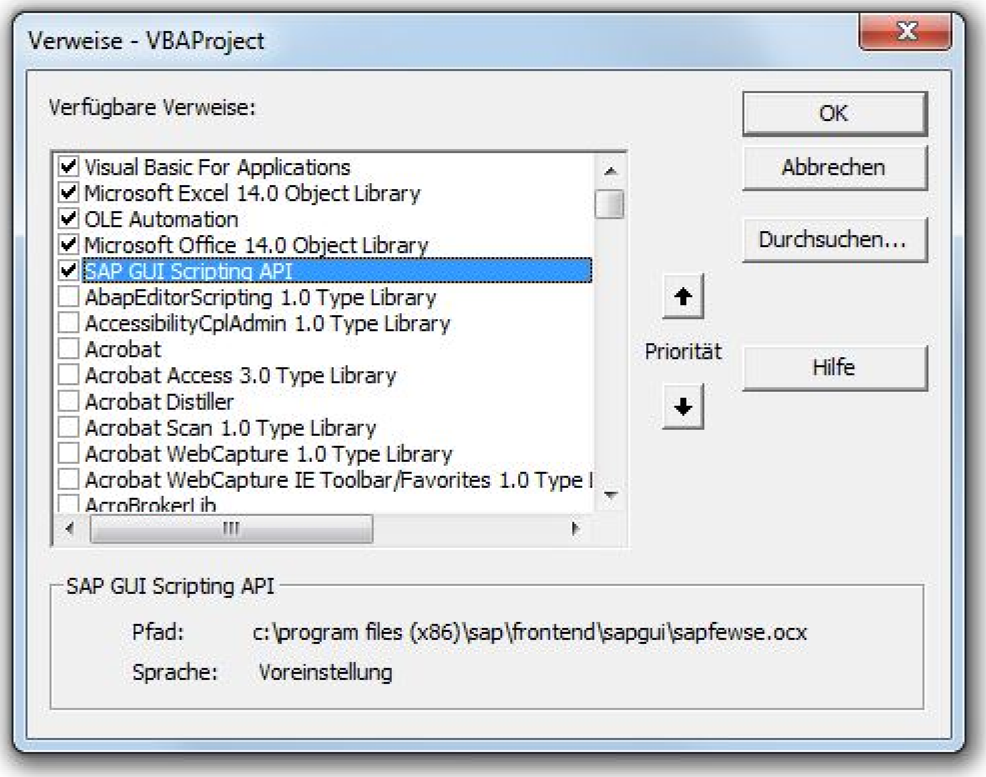
Verweis hinzufügen

Suchen in: SAPgui

Name	Änderungsdatum
SAPfewin.OCX	28.12.2015 11:44
sapfewse.ocx	28.12.2015 11:44
sapgradp.ocx	22.09.2014 02:37
SAPguiRM.ocx	28.12.2015 11:44
SAPguisv.ocx	28.12.2015 11:44

Dateiname: sapfewse.ocx

Dateityp: ActiveX-Steuerelemente (*.ocx)



Verweise - VBAProject

Verfügbare Verweise:

- [✓] Visual Basic For Applications
- [✓] Microsoft Excel 14.0 Object Library
- [✓] OLE Automation
- [✓] Microsoft Office 14.0 Object Library
- [✓] SAP GUI Scripting API
- [ ] AbapEditorScripting 1.0 Type Library
- [ ] AccessibilityCplAdmin 1.0 Type Library
- [ ] Acrobat
- [ ] Acrobat Access 3.0 Type Library
- [ ] Acrobat Distiller
- [ ] Acrobat Scan 1.0 Type Library
- [ ] Acrobat WebCapture 1.0 Type Library
- [ ] Acrobat WebCapture IE Toolbar/Favorites 1.0 Type l
- [ ] AcroBrokerLib

SAP GUI Scripting API

Pfad: c:\program files (x86)\sap\frontend\sapgui\sapfewse.ocx

Sprache: Voreinstellung

```
;-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

  $Connection = $Application.Children(0)
  If Not IsObj($Connection) Then
    Return
  EndIf

  $Session = $Connection.Children(0)
  If Not IsObj($Session) Then
    Return
  EndIf



EndFunc

;-Main------------------------------------------------------------------
Main()

;-End-------------------------------------------------------------------
```

```
#-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

    connection = application.Children(0)
    if not type(connection) == win32com.client.CDispatch:
      application = None
      SapGuiAuto = None
      return

    session = connection.Children(1)
    if not type(session) == win32com.client.CDispatch:
      connection = None
      application = None
      SapGuiAuto = None
      return

    #-Insert your SAP GUI Scripting code here--------------------------



  except:
    print(sys.exc_info()[0])

  finally:
    session = None
    connection = None
    application = None
    SapGuiAuto = None

#-Main------------------------------------------------------------------
if __name__ == "__main__":
  main()

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

```
//-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-------------------------------------------------------------
function Main() {
  var MSScrCtrl = new ActiveXObject("MSScriptControl.ScriptControl");
  MSScrCtrl.AllowUI = 1;
  MSScrCtrl.Language = 'VBScript';
  MSScrCtrl.AddCode(VBSCode);
  MSScrCtrl.Run('SAPGUIScripting');
}

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

//-End------------------------------------------------------------------
```

Examples

Collection of additional examples.

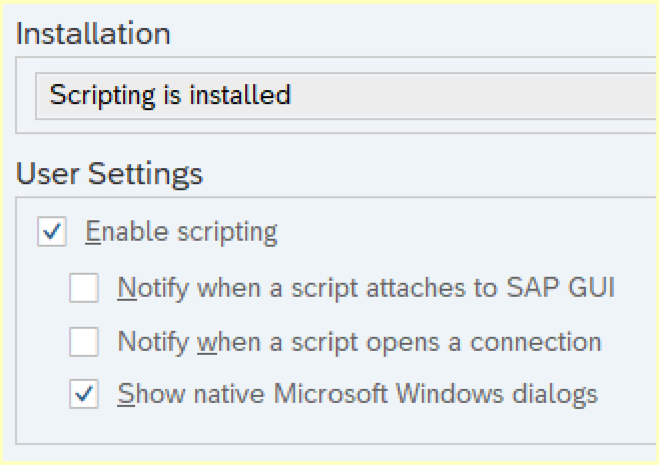
PowerShell
VBA
WSH
AutoIt

PowerShell

```
#-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--------------------------------------------------------------------
```

```
#-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-------------------------------------------------------------------
```

```
#-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---------------------------------------------------------------------
```

```
#-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-------------------------------------------------------------------
```

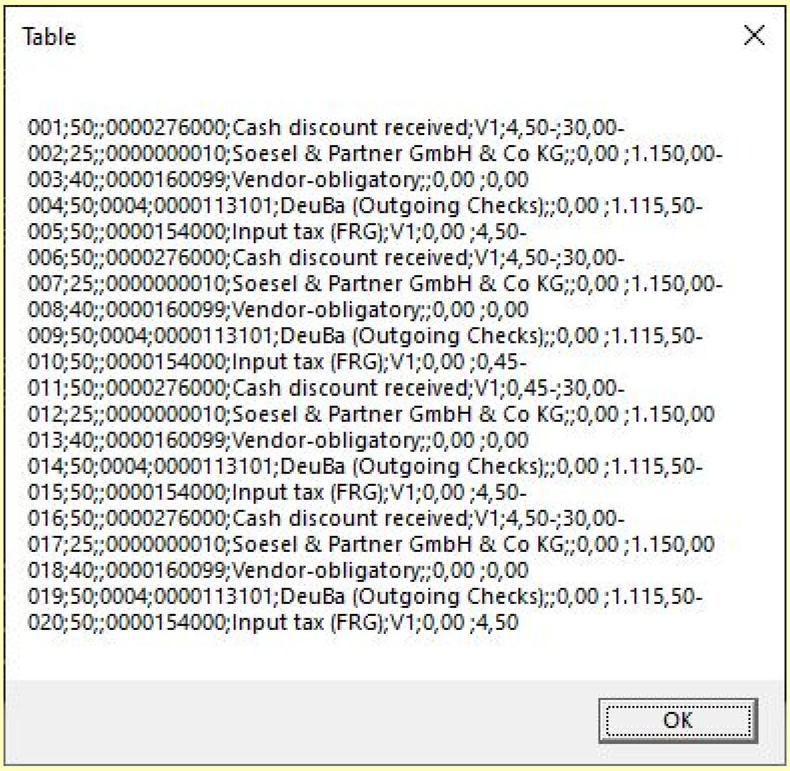
```
#-Begin-----------------------------------------------------------------

#-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 MessagesiIn 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------------------------------------------------------------------
```

Table

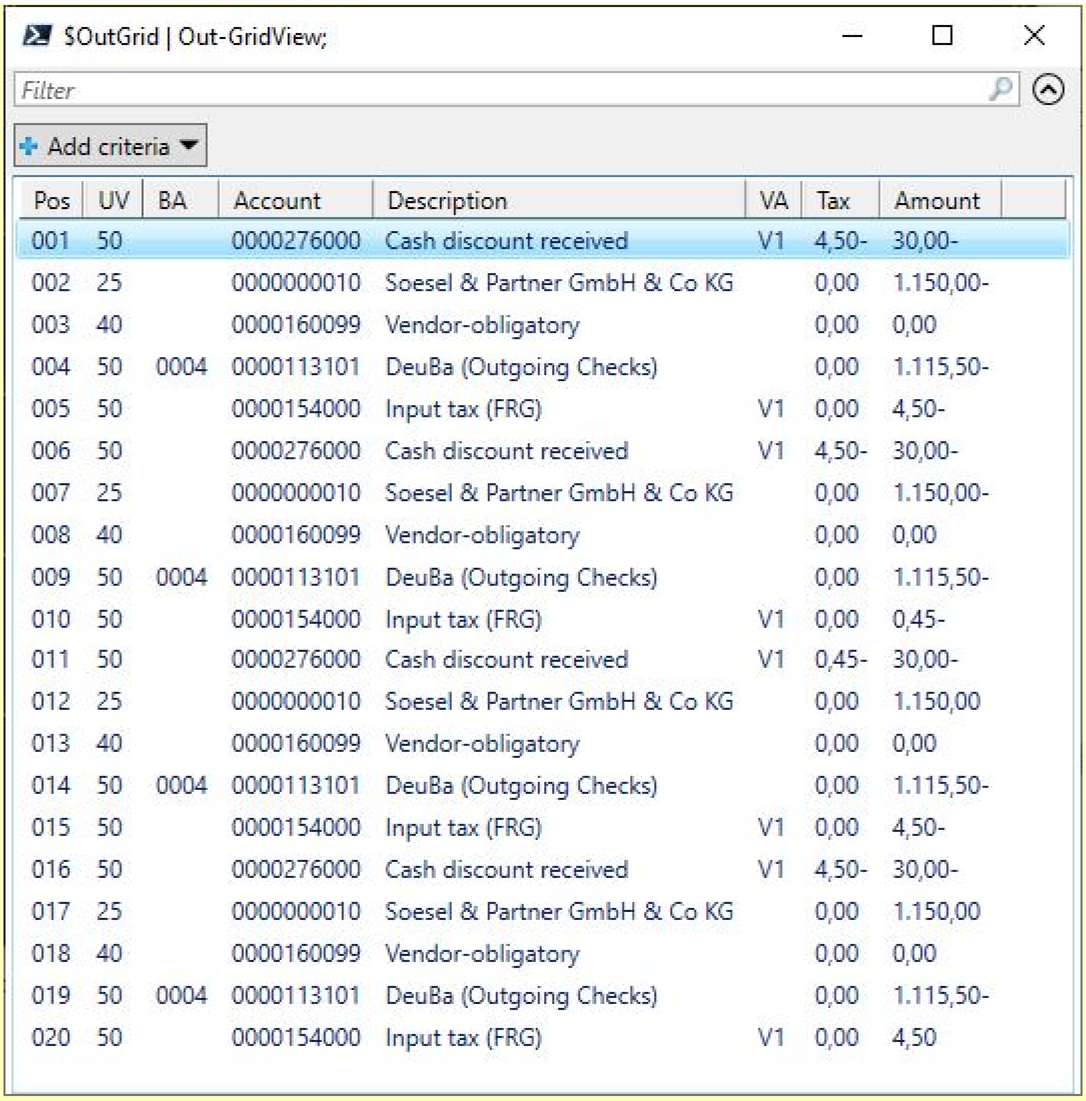
Read TableControl

```
#-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);
```



```
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
```

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


$OutGrid | Out-GridView;

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--------------------------------------------------------------------
```

VBA

You can find information to prepare VBA here.

ClearAllChangeableFields

```
'-Begin-----------------------------------------------------------------


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


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

  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


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

  '-Variables-----------------------------------------------------------
  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-------------------------------------------------------------------
```

WSH

```
'-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-------------------------------------------------------------------
```

```
'-Sub 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
```

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-------------------------------------------------------------------
```

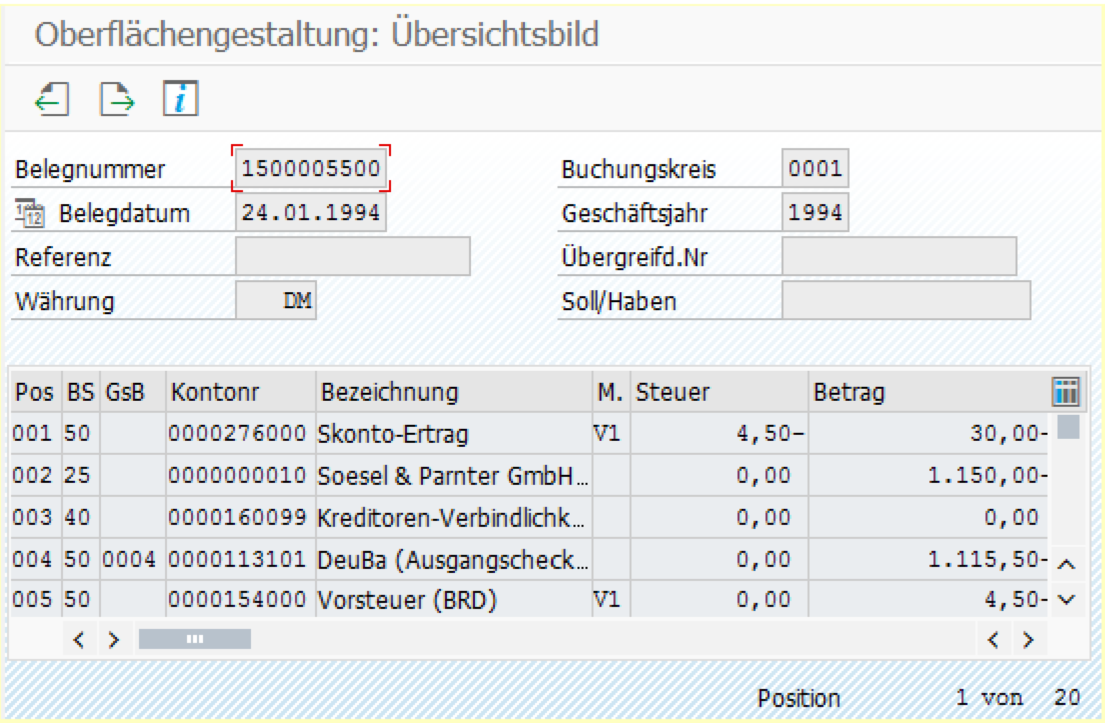
```
'-Begin-----------------------------------------------------------------
'-
'- Example to show how to select a specific session to do SAP GUI
'- Scripting activities inside it. It scans all connections with all
'- sessions to find the correct one.
'-
'- Author: Stefan Schnell
'-
'-----------------------------------------------------------------------


'-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----------------------------------------------------------------
```

```
'-Begin-----------------------------------------------------------------


'-Sum all numbers in a column of a GridView (ALV grid)------------------
Set table =
session.findById("wnd[0]/usr/cntlBCALV_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-------------------------------------------------------------------
```

Table

```
'-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/cntlGRID1/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-------------------------------------------------------------------
```

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

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

'-Sub Main--------------------------------------------------------------
Sub Main

  Dim SapGuiAuto, application, connection, session
  Dim 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

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

  For Row = 0 To RowCount - 1
    Table.verticalScrollbar.position = Row
    Set Table = session.findById("wnd[0]/usr/tblSAPMBIBSTC535")
    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 (Ausgangscheck…		0,00	1.115,50-
005	50		0000154000	Vorsteuer (BRD)	V1	0,00	4,50-

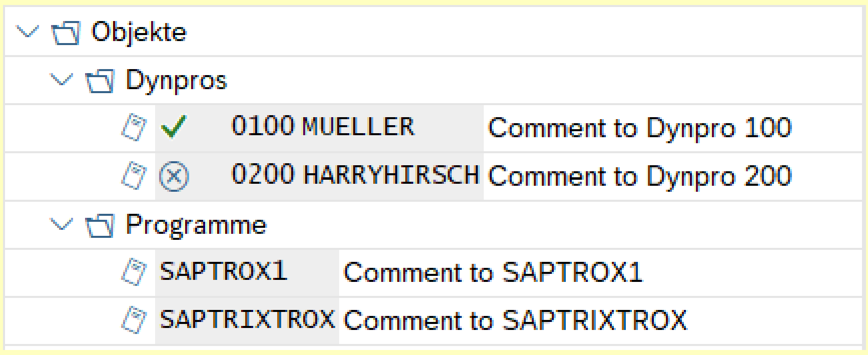
Position 1 von 20

Tree

```
'-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----------------------------------------------------------------------
```

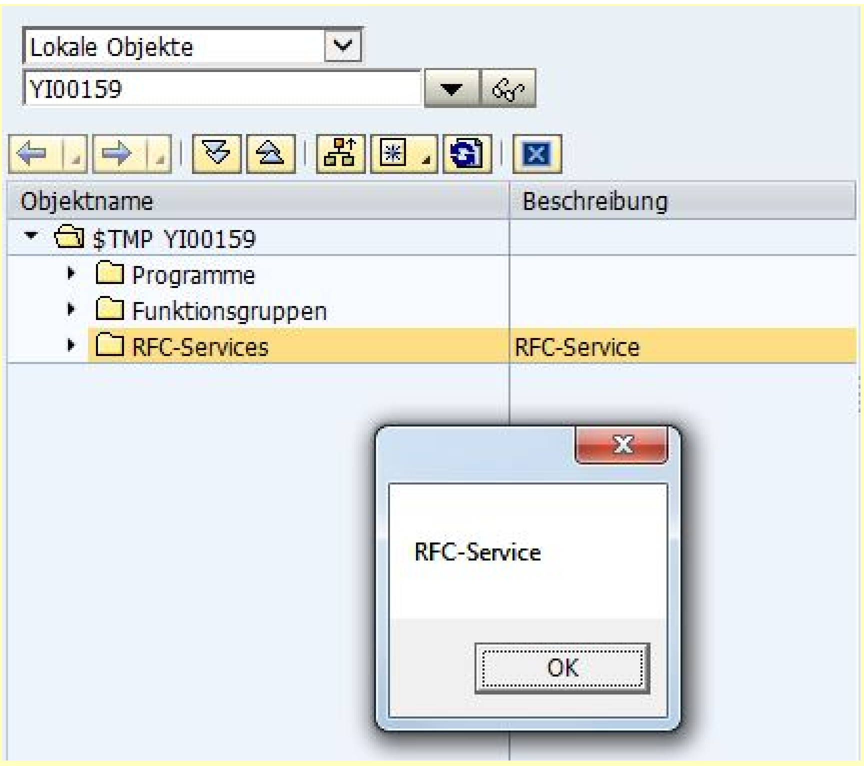
```
'-Begin-----------------------------------------------------------------
'-
'- TAC SESSION_MANAGER or one of the demo reports
'-
'-----------------------------------------------------------------------

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 Tree.GetNodeTextByKey(NodeKey)
Next

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

```
'-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/cntlTREE_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

```
'-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-------------------------------------------------------------------
```

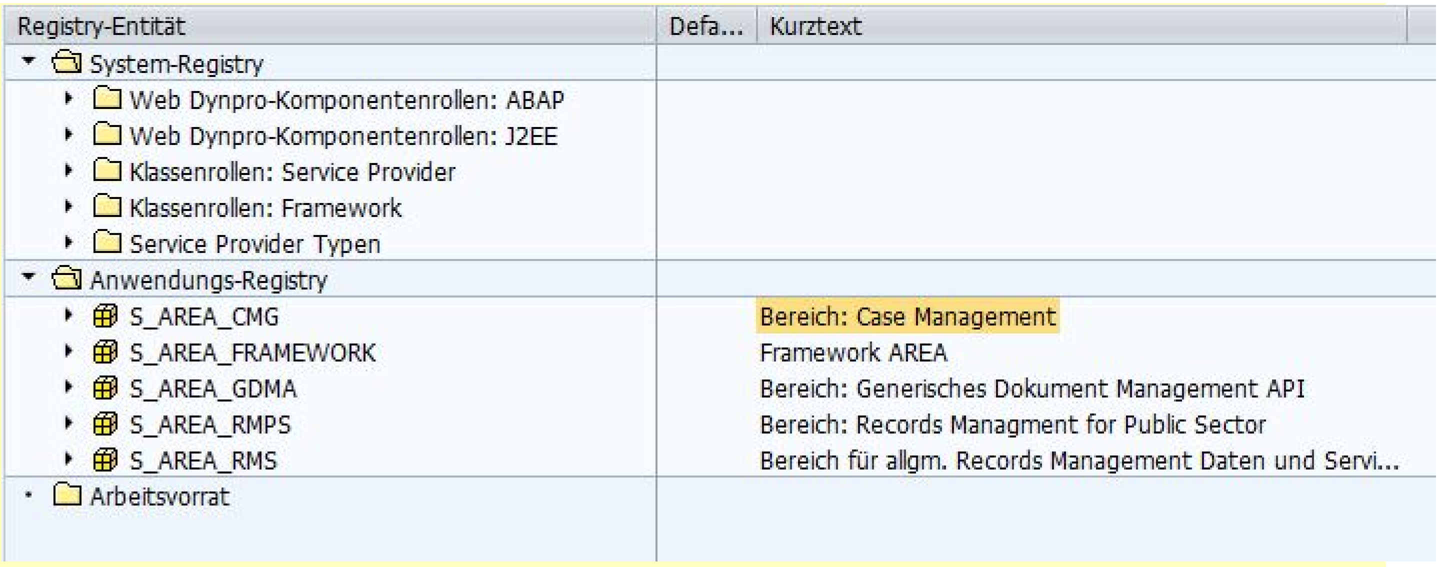

Lokale Objekte

YI00159

Objektname	Beschreibung
$TMP YI00159	
Programme	
Funktionsgruppen	
RFC-Services	RFC-Service

RFC-Service

```
'-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-------------------------------------------------------------------
```



Registry-Entität	Defa...	Kurztext
System-Registry		
Web Dynpro-Komponentenrollen: ABAP		
Web Dynpro-Komponentenrollen: J2EE		
Klassenrollen: Service Provider		
Klassenrollen: Framework		
Service Provider Typen		
Anwendungs-Registry		
S_AREA_CMG		Bereich: Case Management
S_AREA_FRAMEWORK		Framework AREA
S_AREA_GDMA		Bereich: Generisches Dokument Management API
S_AREA_RMPS		Bereich: Records Managment for Public Sector
S_AREA_RMS		Bereich für allgm. Records Management Daten und Servi...
Arbeitsvorrat		

```
'-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
```

```
'-Begin-----------------------------------------------------------------

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

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

  Dim objShell, RegEx, Matches, con_no, ses_no

  Set RegEx = New RegExp
  RegEx.Pattern = "[0-9]"
  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

  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-------------------------------------------------------------------
```

```
'-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----------------------------------------------------------------------
```

AutoIt

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------------------------------------------------------------------
```

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, Scripting Host, PowerShell and Edge are registered trademarks of Microsoft, C# and VB.NET are product names from Microsoft
- AutoIt and AutoItX is property of Jonathan Bennet and the AutoIt team
- Java and JShell are registered trademarks of Oracle
- Jacob is property of Clay Shooter
- Google, Chrome and Android are registered trademarks of Google
- Python is a registered trademark of Python Software Foundation
- UiPath
- Blue Prism

Scintilla

Scripting Tracker uses Scintilla.

AutoItX

Scripting Tracker contains AutoItX. It is allowed to reproduce and distribute an unlimited number of copies of AutoItX either in whole or in part. Scripting Tracker contains the help file of AutoItX and therewith a copy of all copyright and trademark notices.

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.

Contact

WebSite: www.stschnell.de
Support: mail@stschnell.de

Guarantee exclusion