SAPGUI Tutorial

SilkPerformer® 2006 Release 2

Borland®

Borland Software Corporation
20450 Stevens Creek Blvd., Suite 800
Cupertino, California 95014 USA
http://www.borland.com

January 2007
PDF

Contents

Introduction

About this manual

This tutorial provides the information you need to record and customize SAPGUI test scripts.

This chapter contains the following sections:

Overview

SilkPerformer offers recording and replay support for the load testing and functional testing of SAP$^{®}$ systems that use the SAPGUI Scripting interface. SilkPerformer's unique content verification feature enables you to verify application functionality even under real-world load conditions—and thereby intercept application errors that occur only under load.

Test scripts created for functional testing can be reused for load testing purposes, without requiring any changes.

Together with its outstanding support for the load testing of Web applications, SilkPerformer supports load and functional testing of SAP R/3 4.6C, SAP R/3 Enterprise (4.7), and mySAP Business Suite (and higher) via SAPGUI Client 6.2 (and higher) for Windows and HTML, as well as with mySAP Enterprise Portal.

> **Note** This tutorial offers only a brief overview of the functionality that is available with SilkPerformer and TrueLog Explorer. Please see the *SilkPerformer User Guide* and the *TrueLog Explorer User Guide* for full details of available functionality.

Scripting

In addition to a powerful BDL API for SAP that enables programmers to effectively customize SAP test scripts, SilkPerformer 2006 R2 also provides TrueLog technology for SAP—offering easy visual script analysis and customization.

TrueLogs provide complete visual representation of all actions and results that are generated by test scripts. Screenshots are captured during test runs and details regarding all visible GUI controls are logged. Using TrueLog Explorer's intuitive point-and-click interface, you can visually customize all user-input data and create content verification checks for return data. Simply select the input values that you wish to customize, or the result values that you wish to verify, and then choose any appropriate parsing, parameterization, or verification functions. All customization and verification functions are then automatically generated and inserted into your BDL script. No manual scripting is required.

Functional testing

SilkPerformer provides functional and load testing with a single tool. Simply reuse your scripts as both functional and load testing scripts using the same script API.

Front-end analysis

Using SilkPerformer's TrueLog On Error functionality for SAP, you can visually inspect the actions of SAP virtual users and SAP system responses that result in error conditions. In this way, you can visually analyze error conditions from the virtual-user perspective (the front-end).

SAP monitoring

SilkPerformer offers five Performance Explorer monitors that enable you to query SAP server-side performance values.

SAP eCATT

SAP's *eCATT (Extended Computer Aided Test Tool)* facility allows you to create test scripts in SAP using the scripting language of your choice. eCATT allows you to use external test tools such as SilkPerformer while utilizing eCATT as a repository for your test scripts. See "SAP eCATT Integration With SilkPerformer" for details.

Enabling SAP scripting

SAPGUI record/replay technology is based on the SAPGUI Scripting API, which must be enabled on both the server and client side.

The SAPGUI Scripting API is not available in all SAPGUI client versions; therefore you must confirm your patch level. Please refer to *Enable SAP Scripting* in SilkPerformer Online Help for details.

Checking SAP patch level

SAPGUI Scripting is not supported by all versions of SAP. Therefore it is necessary that you confirm that your installation offers this support. You can do this by checking your current patch level. The patch level needs to be at least 44.

Procedure To confirm server patch level:

1 Launch the SAPGUI logon window (Start/Programs/SAP Front End/SAPLogon) and choose the *About SAP Logon* menu item from the window menu.

2 The SAP version information dialog box appears. Confirm that the patch level is at least 44. It is however recommended that you use patch level 50 or higher as some memory related issues have been found with previous patches.

Profile settings

SilkPerformer SAP GUI support is configurable via SilkPerformer profile settings.

Recording settings

The following recording settings can be configured on the profile settings' *Recording* tab:

Script logon as single function
When enabled, the logon procedure is scripted as a *SapGuiLogon* API call. When disabled, multiple API calls (e.g., setting username, setting password, hitting ENTER are scripted).

Script low level functions
Rather than scripting high-level API functions (e.g., SapGuiSetText) low-level API functions are scripted (e.g., SapGuiInvokeMethod, SapGuiSetProperty).

Script timers
Most SAPGUI API functions take an optional timer parameter. When such a parameter is defined, measures are generated during replay. When this option is enabled, the SAPGUI recorder automatically scripts appropriate timer names for each function.

Attach to existing SAP session
When enabled, the SAPGUI recorder attaches to an existing SAPGUI session without recording the *SapGuiOpenConnection* statement.

Record window title verification
When enabled, the SAPGUI recorder scripts *SapGuiSetActiveWindow* with the window title so that the title can later be verified during replay.

Common Settings
The following settings are common to both recording and replay.

Log level
Defines the logging level. For troubleshooting, *Debug* should be used. Otherwise *Normal* should be used. When running large load tests, logging can be *Disabled* to reduce memory consumption.

Capture screenshots
When enabled, screenshots are captured for each new window that is activated. This option is only available when *Show SAP GUI during replay* is enabled during script replay.

Capture screenshots for every action
When enabled, screenshots are captured for each user action that causes a round-trip to the SAP server. This option is only available when *Capture screenshots* is enabled.

Log control information in TrueLog
When enabled, control information for each control on the active window is logged to the TrueLog. This allows you to use TrueLog Explorer's customization feature. This option should be disabled when running load tests as it consumes additional resources.

Log control information on error
When enabled, control information for each control on the active window is logged to the TrueLog when errors occur during replay. This allows you to troubleshoot replay problems by capturing the current state of all controls on the screen when errors occur. It is recommended that you use this option during load tests rather then *Log control information in TrueLog,* which is resource intensive.

Highlight controls (replay only)
With this setting, controls that are accessed during replay by any API call will be highlighted on the SAPGUI Client. This option is only valid when *Show SAP GUI during replay* is enabled.

Replay settings

The following replay settings can be set on the profile settings' *Replay* tab:

Replay timeout
Defines timeout during replay. When there is no response from the server within this timeout period, an error is thrown.

Show SAP GUI during replay
When enabled, the SAP GUI client is shown during replay. This option can only be used for TryScripts. By default, replay for load tests is GUI-less.

Enable client-side scripting
SAPGUI Scripting must be enabled on each client machine via the Options menu of the SAPGUI client application. When running a load test on multiple agents, this setting must be changed manually on each machine before the load test begins. By enabling this option, SilkPerformer changes this setting automatically on each agent before starting load tests.

Use new SAP Visual Design
SAPGUI can be run in one of two visual modes: original design or new design mode. This setting can be changed via the SAP Configuration Tool. By enabling/disabling this option, SilkPerformer performs these changes automatically before starting load tests. This option allows you to compare resource consumption between the old and new visual designs. The measure tab contains settings for replay measurement.

You can either enable all timers for all control types, or select only those timers that are of interest to you. Timers are only created for those method calls that have the optional timer parameter specified.

For a description of these timers, please refer to *SAP Results* in SilkPerformer Online Help.

Client/Server Requirements

On the Server

- Required patch level for SAPGUI support must be installed
- Sapgui/user_scripting:
  - Profile parameter must be set to *True.* This can be changed using the transaction *RZ11*.

On the Client

- SAPGUI Client 620 or 640
- Latest patch level
- SAPGUI Scripting must be installed and enabled
  - To enable SAPGUI Scripting:
    - Start the SAPGUI client.
    - Open the *Options* dialog.
    - Select the *Scripting* tab.
    - Select *Enable Scripting* and deselect the two security check boxes.

Available Functions

SilkPerformer uses a testing interface called SAPGUI Scripting API, which has been introduced by SAP for SAPGUI Windows clients.

To record and replay SAPGUI scripts, some pre-requirements must be met. To prepare your environment for SAP testing, please refer to *Checking your SAP Patch Level* and *Enable SAP Scripting* in SilkPerformer Online Help.

SilkPerformer offers both a low- and high-level API for testing SAP systems. The following high-level function groups are offered:

- Connecting functions
- Control functions
- Parsing functions
- Verifying functions
- Window functions

Please see the *SilkPerformer Online Help* for a complete list of functions and function descriptions.

1

Recording SAPGUI Test Scripts

Introduction

This chapter explains how to generate a SAPGUI test script by recording a SAPGUI application and how to analyze a replayed test script via a TryScript run.

What you will learn

This chapter contains the following sections:

Section	Page
Overview	7
Generating Test Scripts	7
Exploring Recorded Scripts	17

Overview

SilkPerformer offers record and replay support for the load testing and functional testing of SAP systems that use the SAPGUI Scripting interface.

This chapter shows you how to generate a test script by recording a SAPGUI application and then how to analyze the resultant test script by replaying it in a TryScript run.

Generating Test Scripts

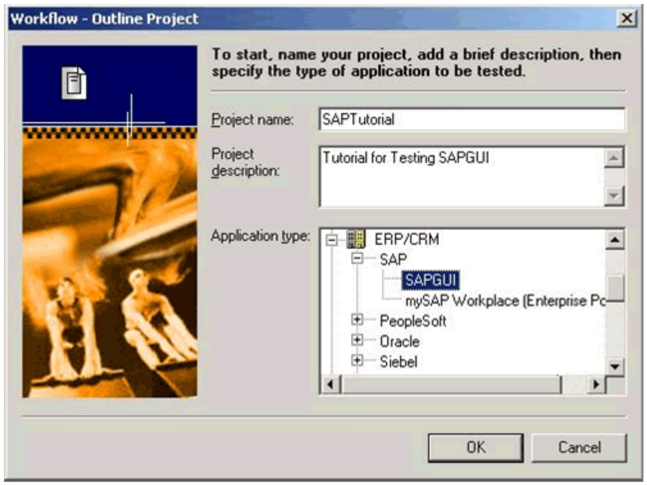
Procedure To generate a test script by recording a SAPGUI application session:

1 Click SilkPerformer's *Outline Project* button to create a new project.

2 Enter a *Project name* and *Project description.*

3 Select *ERP/CRM\SAP\SAPGUI* as the *Application type.*

4 Click *OK.*



Workflow - Outline Project

To start, name your project, add a brief description, then specify the type of application to be tested.

Project name: SAPTutorial

Project description: Tutorial for Testing SAPGUI

Application type: ERP/CRM > SAP > SAPGUI (selected)

5 Click the *Model Script* button on the Workflow Bar.

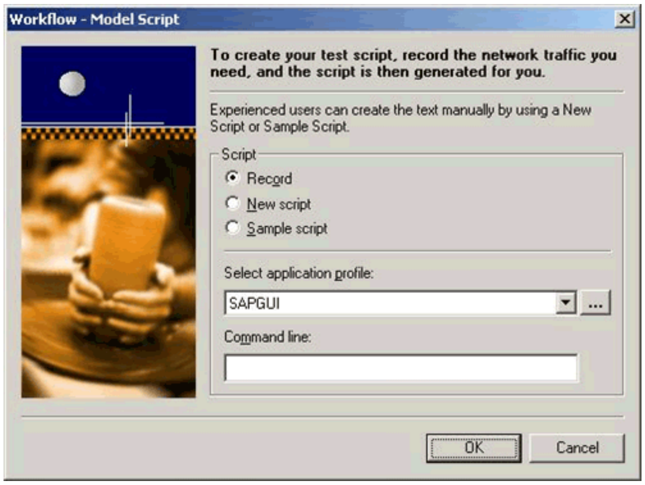
6 If not selected already, select the *Record* radio button to record an application session.

7 The *SAPGUI* Application Profile is preselected. If SAPGUI does not appear in the *Application profile* field, you do not have a SAPGUI client installed on your computer.

Note If you do not have a SAPGUI client installed, install SAPGUI Client 620 or 640 with the latest patch levels before creating a new SilkPerformer project.

If SAPGUI is still not listed in the *Application profile* field, you need to create the application profile for *saplogon.exe*. In SilkPerformer, open the *System Settings* dialog and select the *Recorder* icon on the left. Under the *Application Profiles* tab, *add* a new Application profile called *SAPGUI* and specify your saplogon.exe as executable in the *Application path* field. Specify *Custom Application* for the *Application type* and check the *SAPGUI* protocol checkbox. Click the *OK* button to confirm your settings and close the *System Settings* dialog.

8 Click *OK* to launch *saplogon.exe.*


Workflow - Model Script

To create your test script, record the network traffic you need, and the script is then generated for you.

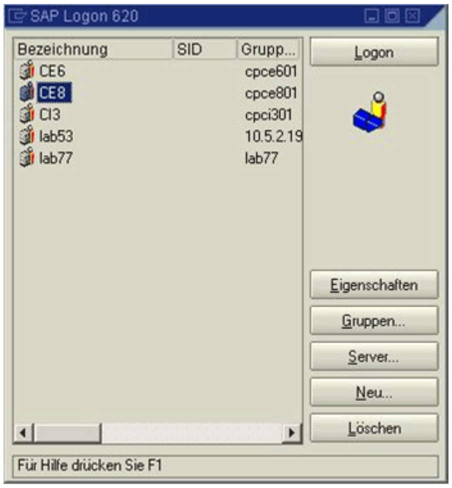
Experienced users can create the text manually by using a New Script or Sample Script.

Script
- [✓ Record]
- New script
- Sample script

Select application profile: SAPGUI

Command line:

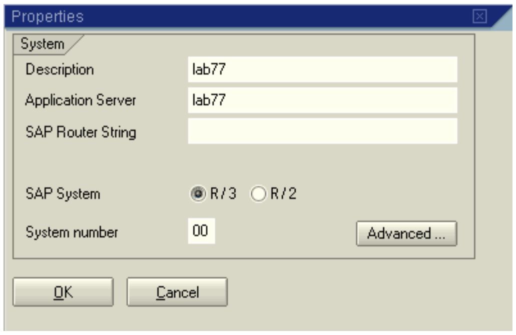
9 Specify the SAP application server that is to be tested (this tutorial illustrates the testing of a SAP calendar application).


SAP Logon 620

Bezeichnung	SID	Grupp...
CE6		cpce601
CE8		cpce801
CI3		cpci301
lab53		10.5.2.19
lab77		lab77

Für Hilfe drücken Sie F1

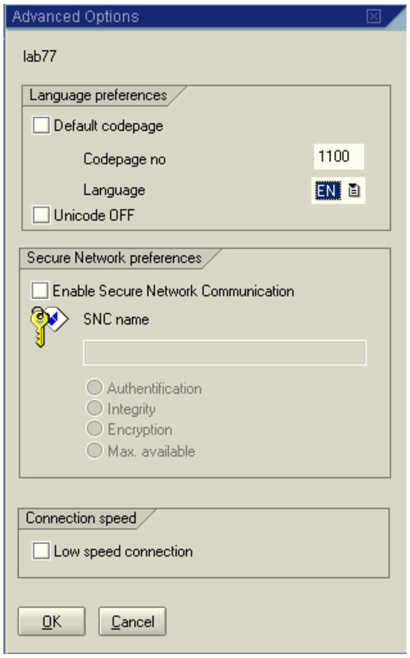
Note On multi-lingual SAP systems it is recommended that you specify the language that is to be used by the SAPGUI client before recording begins. This prevents possible language differences between recording and replay (i.e., different languages may be selected by different load test agents, which will lead to Window Title Verification errors). This change can be made on the Properties dialog of each SAP connection in the SAPGUI login application. Edit the properties of each connection and click the *Advanced* button.


Properties

System

Description	lab77
Application Server	lab77
SAP Router String	
SAP System	[✓ R/3] R/2
System number	00

On the *Advanced Options* dialog, deselect *Default codepage* and select your preferred language. Click *OK* to accept the change.


Advanced Options

lab77

Language preferences

☐ Default codepage

Codepage no	1100
Language	EN

☐ Unicode OFF

Secure Network preferences

☐ Enable Secure Network Communication

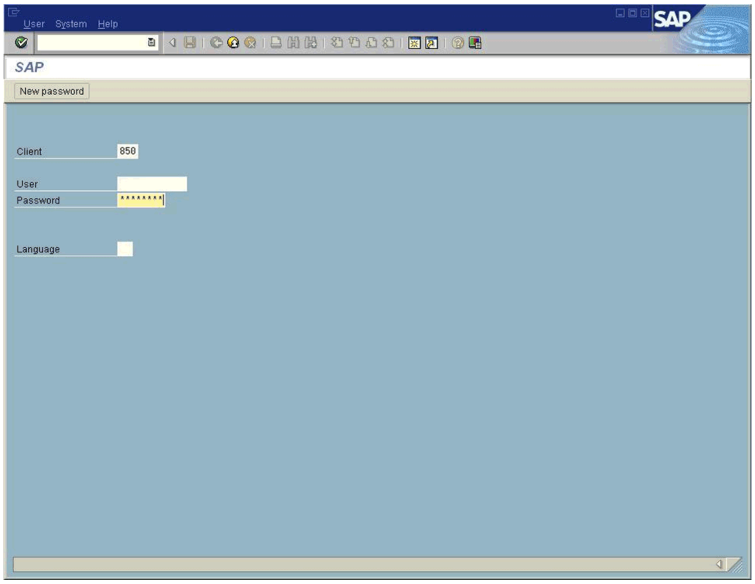
SNC name

○ Authentification
○ Integrity
○ Encryption
○ Max. available

Connection speed

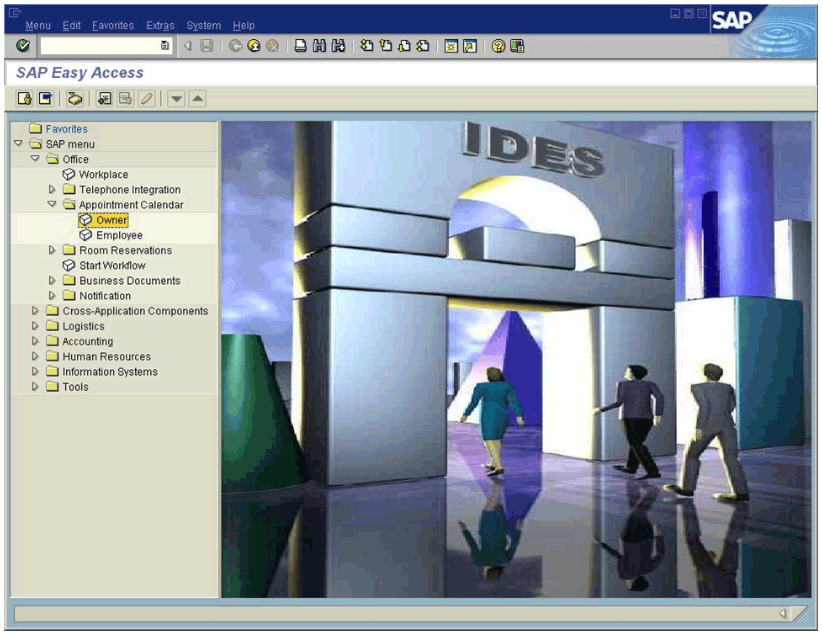
☐ Low speed connection

10 Now simulate the actions of a typical user transaction. Login with username and password. Hit *Enter.*



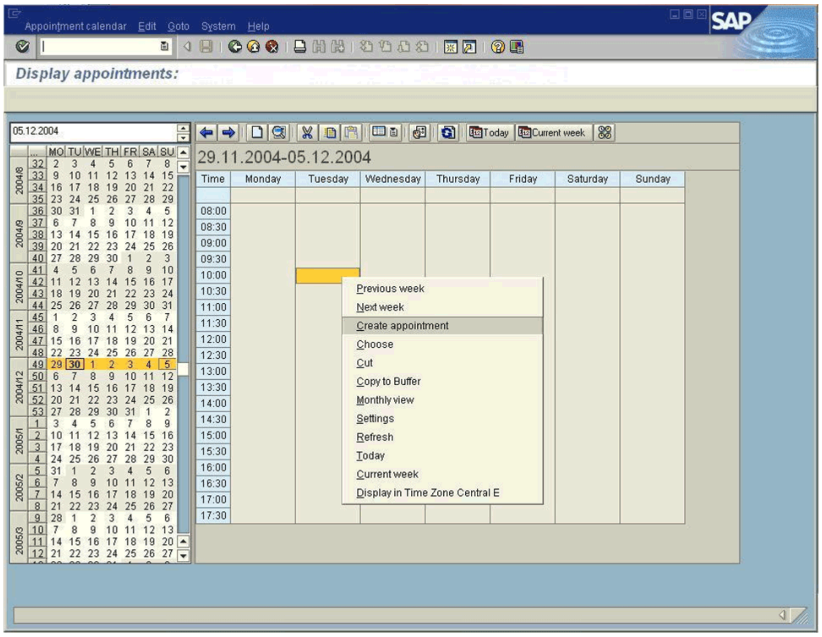
Client 850
User
Password ********
Language

11 Expand the tree and double-click the *Owner* item.

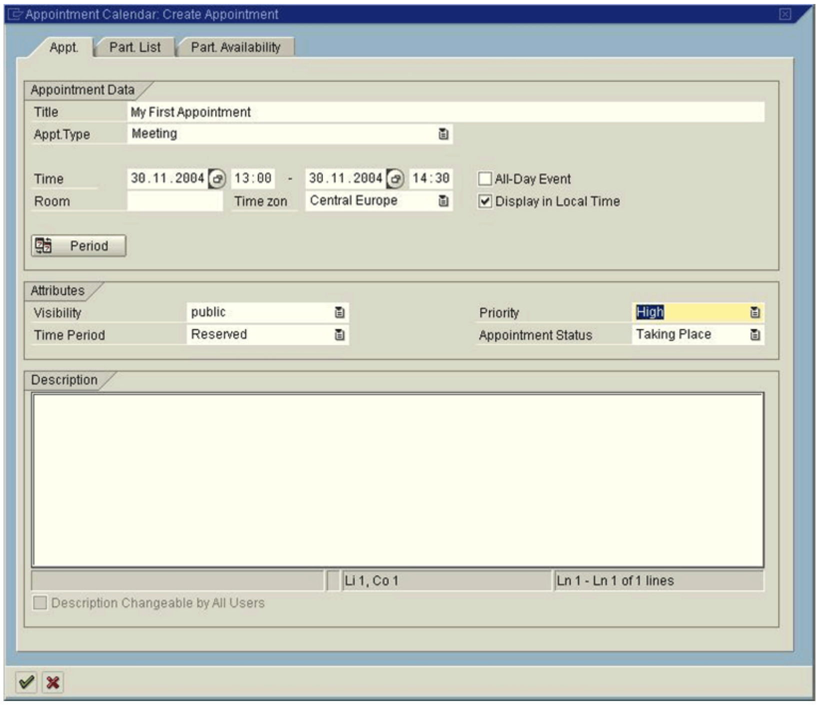


- Favorites
- SAP menu
  - Office
    - Workplace
    - Telephone Integration
    - Appointment Calendar
      - Owner
      - Employee
    - Room Reservations
    - Start Workflow
    - Business Documents
    - Notification
  - Cross-Application Components
  - Logistics
  - Accounting
  - Human Resources
  - Information Systems
  - Tools

12 Right-click a time frame and select *Create Appointment* from the context menu to create a new appointment.



13 Define the appointment by setting *Title, Appointment Type, Time,* and *Priority.* Then click *Save.*



Appointment Data

Title	My First Appointment
Appt.Type	Meeting
Time	30.11.2004 13:00 - 30.11.2004 14:30
Room	
Time zon	Central Europe

All-Day Event [ ]
Display in Local Time [✓]

Attributes

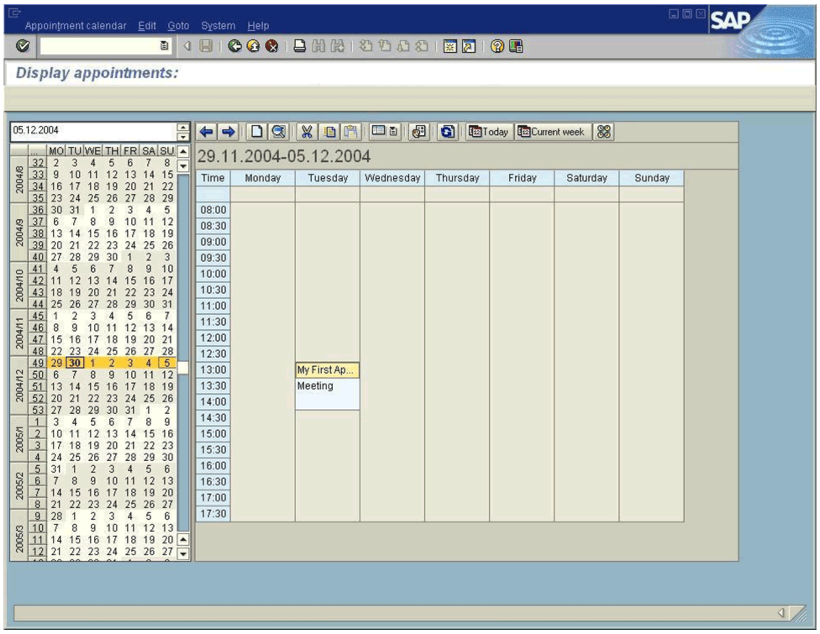
Visibility	public	Priority	High
Time Period	Reserved	Appointment Status	Taking Place

Description

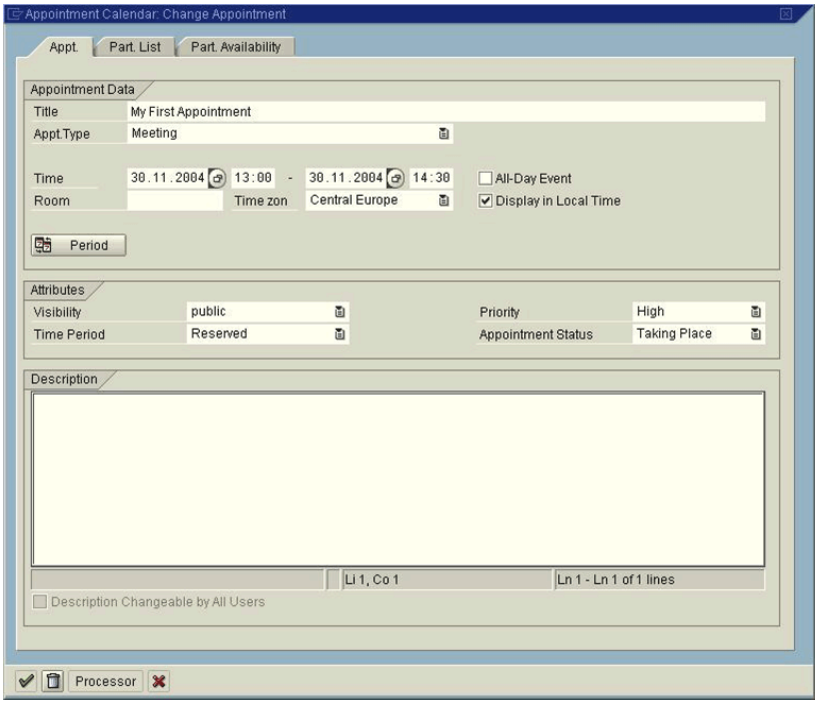
Li 1, Co 1 — Ln 1 - Ln 1 of 1 lines

Description Changeable by All Users [ ]

14 Now edit the appointment you created by double-clicking it.



15 Click the *Delete* button (trash can icon) to delete the appointment. A confirmation dialog appears. Click *Yes* to confirm deletion of the appointment.



Appointment Data

Title	My First Appointment
Appt.Type	Meeting
Time	30.11.2004 13:00 - 30.11.2004 14:30
Room	
Time zon	Central Europe

☐ All-Day Event
☑ Display in Local Time

Attributes

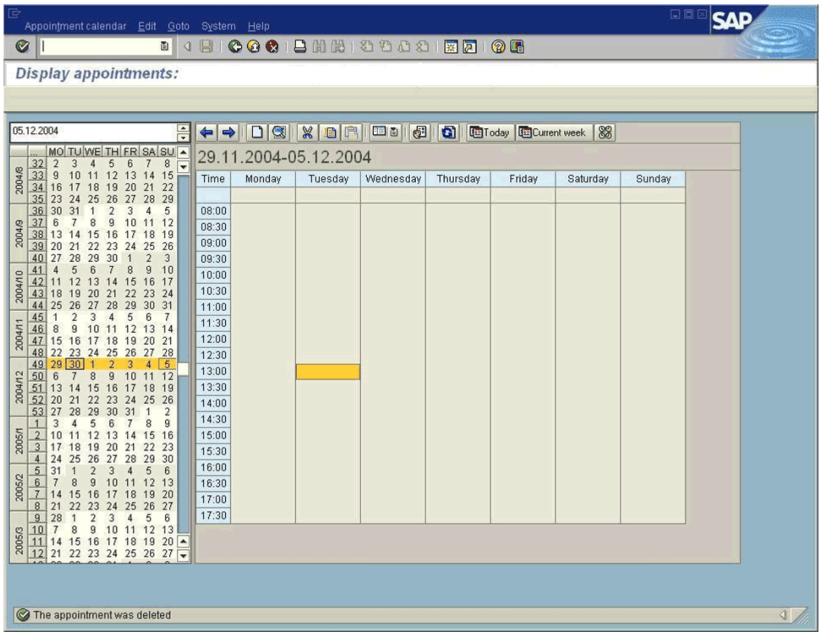
Visibility	public	Priority	High
Time Period	Reserved	Appointment Status	Taking Place

Description

Li 1, Co 1 Ln 1 - Ln 1 of 1 lines

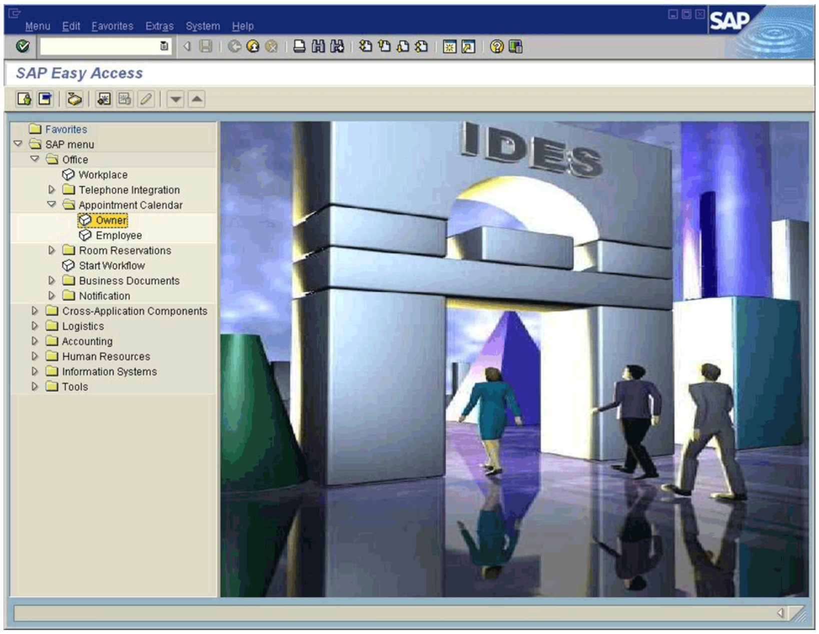
☐ Description Changeable by All Users

16 Click *Exit* (the yellow circular button) to exit the calendar application.



The appointment was deleted

17 Click *Exit* again to exit SAP Workbench. On the following confirmation dialog, confirm that you wish to exit the application.



18 Close the SAPLogon application. This ends your simulated user transaction.

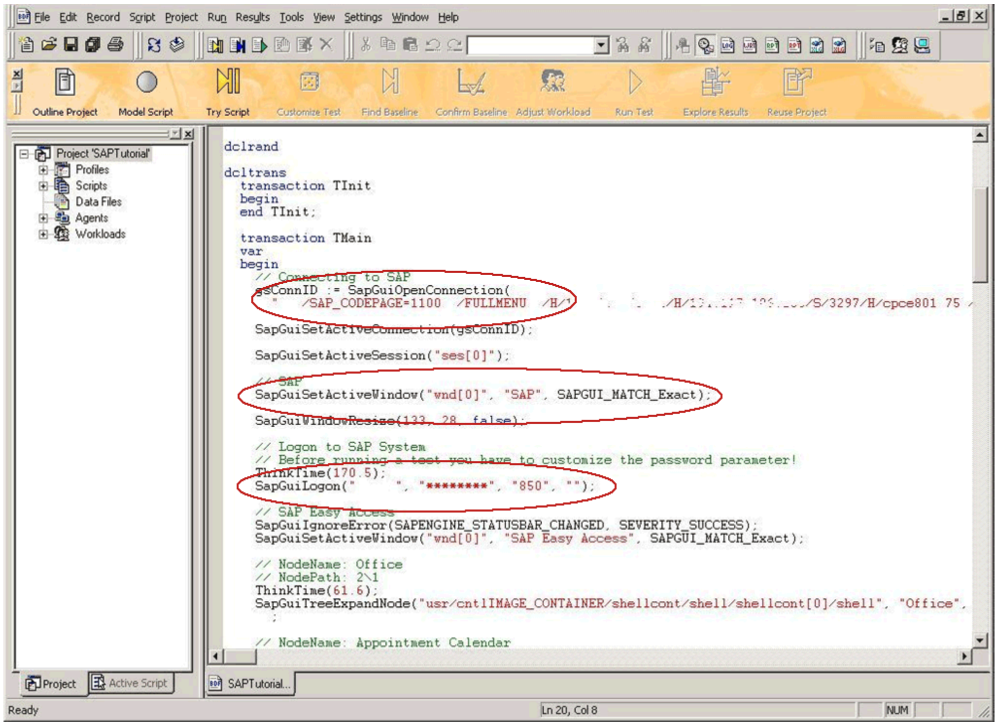
19 Stop SilkPerformer‘s Recorder.

20 Save the recorded script file.

Note If no script has been recorded, review section “Client/Server Requirements”.

Notice three things in the generated BDL script:

- The connection is opened with the full connection string.
- During replay new active windows are verified based on their titles.
- The login string must be customized with a parameter because the password value wasn’t retrieved during recording.



```
dclrand

dcltrans
  transaction TInit
  begin
  end TInit;

  transaction TMain
  var
  begin
    // Connecting to SAP
    gsConnID := SapGuiOpenConnection(
      "   /SAP_CODEPAGE=1100  /FULLMENU  /H/[illegible]/H/[illegible]/S/3297/H/cpce801 75

    SapGuiSetActiveConnection(gsConnID);

    SapGuiSetActiveSession("ses[0]");

    // SAP
    SapGuiSetActiveWindow("wnd[0]", "SAP", SAPGUI_MATCH_Exact);

    SapGuiWindowResize(133, 28, false);

    // Logon to SAP System
    // Before running a test you have to customize the password parameter!
    ThinkTime(170.5);
    SapGuiLogon("      ", "********", "850", "");

    // SAP Easy Access
    SapGuiIgnoreError(SAPENGINE_STATUSBAR_CHANGED, SEVERITY_SUCCESS);
    SapGuiSetActiveWindow("wnd[0]", "SAP Easy Access", SAPGUI_MATCH_Exact);

    // NodeName: Office
    // NodePath: 2\1
    ThinkTime(61.6);
    SapGuiTreeExpandNode("usr/cntlIMAGE_CONTAINER/shellcont/shell/shellcont[0]/shell", "Office",
      ;

    // NodeName: Appointment Calendar
```

Exploring Recorded Scripts

The first step in analyzing and customizing a test script is executing a TryScript run to look for replay errors.

Both recorded and replayed test scripts can be opened in TrueLog Explorer. TrueLog Explorer supports the visualization of SAPGUI requests and responses in the same way it supports the visualization of HTTP client requests and HTTP/HTML server responses. See "Customizing SAPGUI Test Scripts" and the *TrueLog Explorer User Guide* for full details regarding TrueLog Explorer.

Executing TryScripts

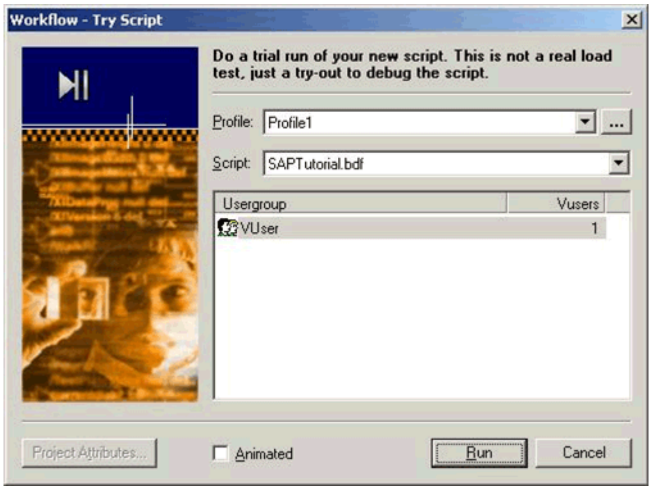
The default option settings for TryScript runs include live display of data downloaded during testing and the writing of log and report files.

With TryScript runs, only a single virtual user is run and the stress test option is enabled so that there is no think-time or delay between transactions.

Procedure To execute a TryScript run:

1. Click the *Try Script* button on the SilkPerformer Workflow bar. The *Try Script* dialog appears. The active profile is selected in the *Profile* drop-down list and the script you created is selected in the *Script* drop-down list. The *VUser* virtual user group is selected in the *Usergroup* area.
2. To view rendered page transitions during a TryScript run, select the *Animated* checkbox.
3. Click *Run*.

   Note You are not running an actual load test here, only a test run to see if your script requires debugging.



Do a trial run of your new script. This is not a real load test, just a try-out to debug the script.

Profile: Profile1

Script: SAPTutorial.bdf

Usergroup	Vusers
VUser	1

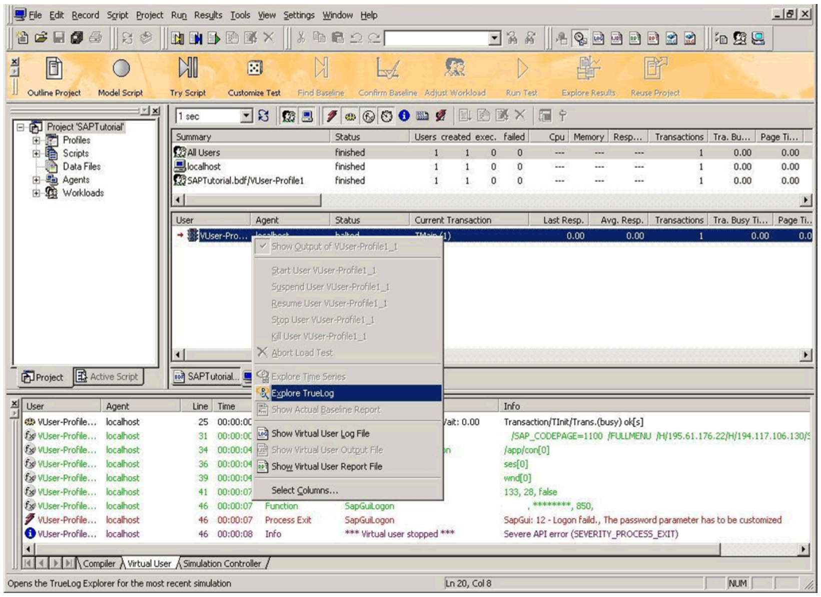
☐ Animated

4. The TryScript run begins. The SilkPerformer Monitor window opens, giving you detailed information about the run's progress.

Note If you selected the *Animated* checkbox on the *TryScript* dialog, TrueLog Explorer will open, showing you the data that is actually downloaded during the TryScript run. Each main SAPGUI window accessed during recording is listed as a high-level *SapGuiSetActiveWindow* API node in TrueLog Explorer's tree view. All recorded server round-trips and user actions are listed as subnodes of corresponding *SapGuiSetActiveWindow* nodes. Animated mode for TryScripts is not really necessary as replay includes the GUI by default—having an additional animated TrueLog might confuse results.

Exploring TrueLogs

Once you have executed a TryScript run, you can explore the TrueLog that was generated by the script run by right-clicking the TryScript user and selecting *Explore TrueLog* from the context menu. This launches TrueLog Explorer loaded with the TrueLog from the recent TryScript run.



Summary	Status	Users	created	exec.	failed	Cpu	Memory	Resp...	Transactions	Tra. Bu...	Page Ti...
All Users	finished	1	1	0	0	---	---	---	1	0.00	0.00
localhost	finished	1	1	0	0	---	---	---	1	0.00	0.00
SAPTutorial.bdf/VUser-Profile1	finished	1	1	0	0	---	---	---	1	0.00	0.00

User	Agent	Status	Current Transaction	Last Resp.	Avg. Resp.	Transactions	Tra. Busy Ti...	Page Ti..
VUser-Pro...	localhost	halted	TMain (1)	0.00	0.00	1	0.00	0.0

User	Agent	Line	Time				Info
VUser-Profile...	localhost	25	00:00:0C			ait: 0.00	Transaction/TInit/Trans.(busy) ok[s]
VUser-Profile...	localhost	31	00:00:0C				/SAP_CODEPAGE=1100 /FULLMENU /H/195.61.176.22/H/194.117.106.130/
VUser-Profile...	localhost	34	00:00:04			n	/app/con[0]
VUser-Profile...	localhost	36	00:00:04				ses[0]
VUser-Profile...	localhost	39	00:00:04				wnd[0]
VUser-Profile...	localhost	41	00:00:07				133, 28, false
VUser-Profile...	localhost	46	00:00:07	Function	SapGuiLogon		, ********, 850,
VUser-Profile...	localhost	46	00:00:07	Process Exit	SapGuiLogon		SapGui: 12 - Logon faild., The password parameter has to be customized
VUser-Profile...	localhost	46	00:00:08	Info	*** Virtual user stopped ***		Severe API error (SEVERITY_PROCESS_EXIT)

Opens the TrueLog Explorer for the most recent simulation	Ln 20, Col 8	NUM

2

Customizing SAPGUI Test Scripts

Introduction

This chapter explains how to customize a SAPGUI load test script based on the results of a TryScript run.

What you will learn

This chapter contains the following sections:

Overview

Once you've recorded a test script and identified session-specific errors through a TryScript run, use TrueLog Explorer to customize the test script so that it can handle session-specific strings (e.g., user IDs, password,).

Note TrueLog Explorer is a powerful test script customization tool that offers much more functionality than is demonstrated in this tutorial. Please see the *TrueLog Explorer User Guide* for details regarding content verifications, content parsing, comparison of record/replay TrueLogs, and much more.

Once you've generated a load test script with SilkPerformer and executed a TryScript run, TrueLog Explorer can help you customize the script by:

- **Adding content verifications -** Using the *Add Verifications* tool, you can gain tremendous insight into data that's downloaded during load tests—enabling you to verify that the content that is to be sent by the server is correct. Verifications remain useful after system deployment for ongoing performance management. See the *TrueLog Explorer User Guide* for details.
- **Adding parsing functions** - TrueLog Explorer allows you to insert SAPGUI parsing functions visually in *Source* screengrab view and on the *Controls* view tab. Manual code writing isn't required—TrueLog Explorer automatically generates parsing functions in scripts. See the *TrueLog Explorer User Guide* for details.
- **Parameterizing input data -** With user data customization you can make your test scripts more realistic by replacing static recorded user input data with dynamic, parameterized user data that changes with each transaction. Manual scripting isn't required to create such "data-driven" tests.

For each SAPGUI function call that changes input data, you can verify return values, parse values, and customize input data. These operations can be executed from both *Source* screengrab view (by right-clicking within a control) and the *Controls* tree view.

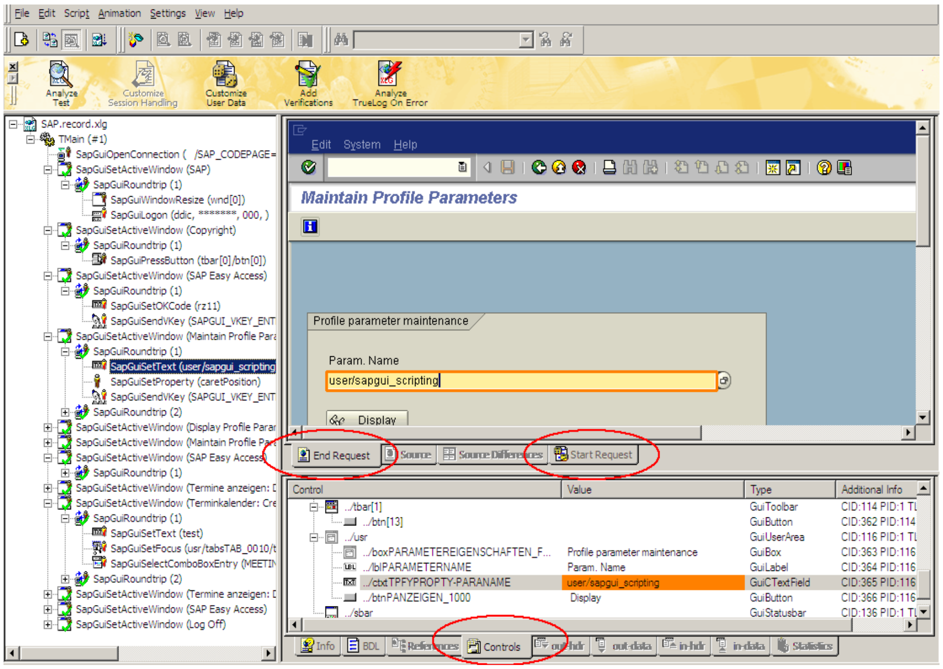
SAPGUI TrueLog Structure

The three windows that are displayed with SAPGUI TrueLogs are:

- **Tree list** (left-hand pane) **-** Lists all SAPGUI API calls that were included in the test run
- **Source window** (upper right-hand pane) - Displays the state of the GUI at each API node. The *End Request* and *Start Request* view tabs enable you to view both the initial and final states of each SAPGUI server request, to see how the server request has affected the GUI display (e.g., the display of a new dialog or error message).

  Note TrueLog screengrabs are captured only during TryScript runs, not load tests.

- **Information window** (lower right-hand pane) **-** Displays data regarding the most recent test run. The view tabs in this pane that are active and applicable to SAPGUI TrueLogs are *Info, BDL,* and *Controls.* The *Controls* tab offers a convenient means of viewing and working with all customizable controls that are included on each GUI screen.



```
SAP.record.xlg
  TMain (#1)
    SapGuiOpenConnection (  /SAP_CODEPAGE=
    SapGuiSetActiveWindow (SAP)
      SapGuiRoundtrip (1)
        SapGuiWindowResize (wnd[0])
        SapGuiLogon (ddic, *******, 000, )
    SapGuiSetActiveWindow (Copyright)
      SapGuiRoundtrip (1)
        SapGuiPressButton (tbar[0]/btn[0])
    SapGuiSetActiveWindow (SAP Easy Access)
      SapGuiRoundtrip (1)
        SapGuiSetOKCode (rz11)
        SapGuiSendVKey (SAPGUI_VKEY_ENT
    SapGuiSetActiveWindow (Maintain Profile Para
      SapGuiRoundtrip (1)
        SapGuiSetText (user/sapgui_scripting
        SapGuiSetProperty (caretPosition)
        SapGuiSendVKey (SAPGUI_VKEY_ENT
      SapGuiRoundtrip (2)
    SapGuiSetActiveWindow (Display Profile Parar
    SapGuiSetActiveWindow (Maintain Profile Pa
    SapGuiSetActiveWindow (SAP Easy Access)
      SapGuiRoundtrip (1)
    SapGuiSetActiveWindow (Termine anzeigen: [
    SapGuiSetActiveWindow (Terminkalender: Cre
      SapGuiRoundtrip (1)
        SapGuiSetText (test)
        SapGuiSetFocus (usr/tabsTAB_0010/t
        SapGuiSelectComboBoxEntry (MEETIN
      SapGuiRoundtrip (2)
    SapGuiSetActiveWindow (Termine anzeigen: [
    SapGuiSetActiveWindow (SAP Easy Access)
    SapGuiSetActiveWindow (Log Off)
```

Maintain Profile Parameters

Profile parameter maintenance

Param. Name: user/sapgui_scripting

Display

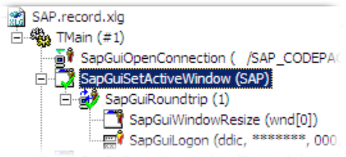
Control	Value	Type	Additional Info
../tbar[1]		GuiToolbar	CID:114 PID:1 Tl
../btn[13]		GuiButton	CID:362 PID:114
../usr		GuiUserArea	CID:116 PID:1 Tl
../boxPARAMETEREIGENSCHAFTEN_F...	Profile parameter maintenance	GuiBox	CID:363 PID:116
../lblPARAMETERNAME	Param. Name	GuiLabel	CID:364 PID:116
../ctxtTPFYPROPTY-PARANAME	user/sapgui_scripting	GuiCTextField	CID:365 PID:116
../btnPANZEIGEN_1000	Display	GuiButton	CID:366 PID:116
../sbar		GuiStatusbar	CID:136 PID:1 Tl

SAPGUI TrueLog functions

Two of the main SAPGUI function types that TrueLog Explorer relies on are:

SapGuiActiveSetWindow - These are top-level API nodes that indicate the generation of new GUI windows. All actions taken on windows are grouped below their corresponding *SapGuiActiveSetWindow* functions.

SapGuiRoundTrip - These are virtual nodes; there are in fact no API calls called SapGuiRoundTrip that are sent to the server. These nodes are used to group all client-side actions that occur in the course of each server round-trip. Both the before and after states of round-trips can be viewed. Multiple round-trip nodes may be included under each *SapGuiSetActiveWindow* node.



```
SAP.record.xlg
  TMain (#1)
    SapGuiOpenConnection (  /SAP_CODEPA
    SapGuiSetActiveWindow (SAP)
      SapGuiRoundtrip (1)
        SapGuiWindowResize (wnd[0])
        SapGuiLogon (ddic, ********, 000
```

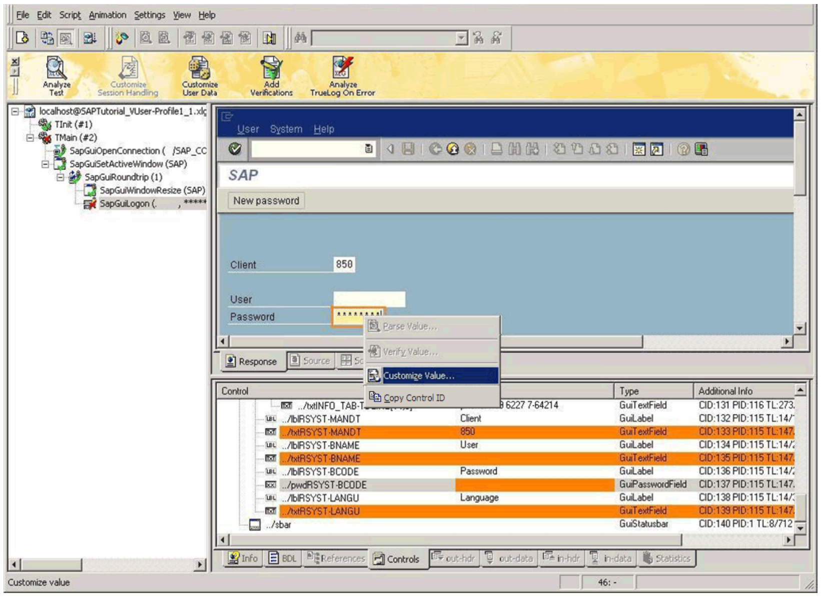
Customizing Input Parameters

In the previous chapter, replay execution was halted when the replay engine checked for “******” in the password field and an error resulted. Until the password string is customized with a variable, the script will not replay correctly.

Procedure To customize an input parameter:

1. Select the failed *SapGuiLogon* method API call in TrueLog Explorer’s tree view.
2. Select the password field in the rendered GUI window.
3. Right-click in the field and select *Customize Value* from the context menu.

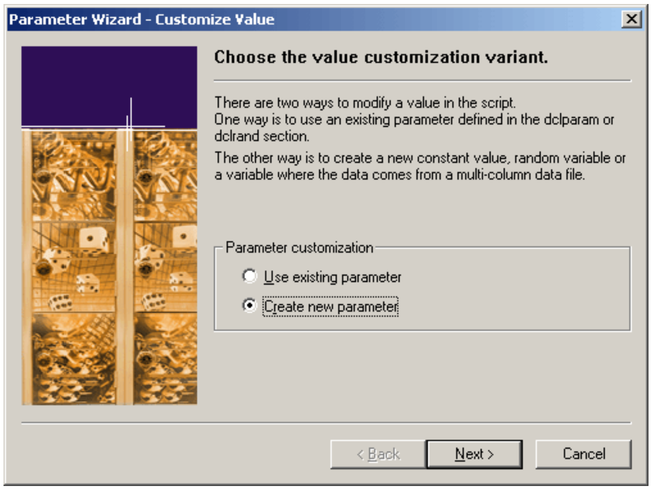
   Note All GUI controls on the window at the selected API node are alternately displayed below on the *Controls* tree window. Fields that are changed by the current call (and can therefore be customized) are highlighted in orange. You can right-click values in the *Controls* window to access the same customization functions that are available above in the rendered GUI window. Most controls can be parsed for their values. Verifications can also be defined for most controls. All available functions are accessible via context menus.



Control		Type	Additional Info
../txtINFO_TAB-T	6227 7-64214	GuiTextField	CID:131 PID:116 TL:273
../lblRSYST-MANDT	Client	GuiLabel	CID:132 PID:115 TL:14/
../txtRSYST-MANDT	850	GuiTextField	CID:133 PID:115 TL:147
../lblRSYST-BNAME	User	GuiLabel	CID:134 PID:115 TL:14/
../txtRSYST-BNAME		GuiTextField	CID:135 PID:115 TL:147
../lblRSYST-BCODE	Password	GuiLabel	CID:136 PID:115 TL:14/
../pwdRSYST-BCODE		GuiPasswordField	CID:137 PID:115 TL:147
../lblRSYST-LANGU	Language	GuiLabel	CID:138 PID:115 TL:14/
../txtRSYST-LANGU		GuiTextField	CID:139 PID:115 TL:147
../sbar		GuiStatusbar	CID:140 PID:1 TL:8/712

4 The *Parameter Wizard* opens. The Parameter Wizard enables you to create a new parameter for the recorded password. To keep this example simple, a constant parameter type will be used. Select the *Create new parameter* radio button and click *Next.*

Note See the *SilkPerformer User Guide* for full details regarding the Parameter Wizard.

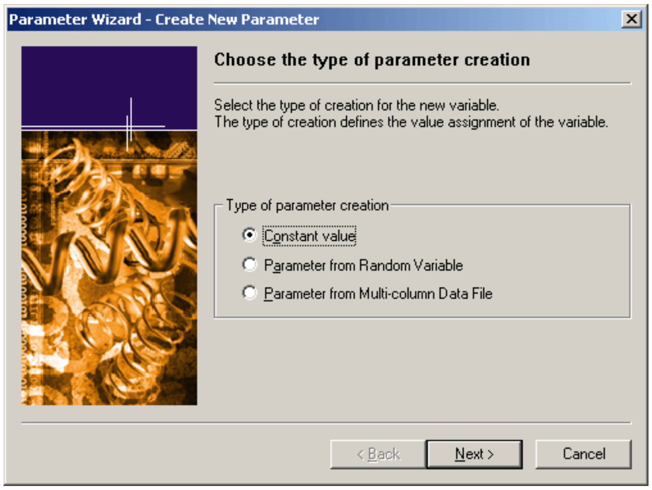

Parameter Wizard - Customize Value

Choose the value customization variant.

There are two ways to modify a value in the script.
One way is to use an existing parameter defined in the dclparam or dclrand section.
The other way is to create a new constant value, random variable or a variable where the data comes from a multi-column data file.

Parameter customization
- ○ Use existing parameter
- ● Create new parameter

5 Select the *Constant value* radio button and click *Next.*

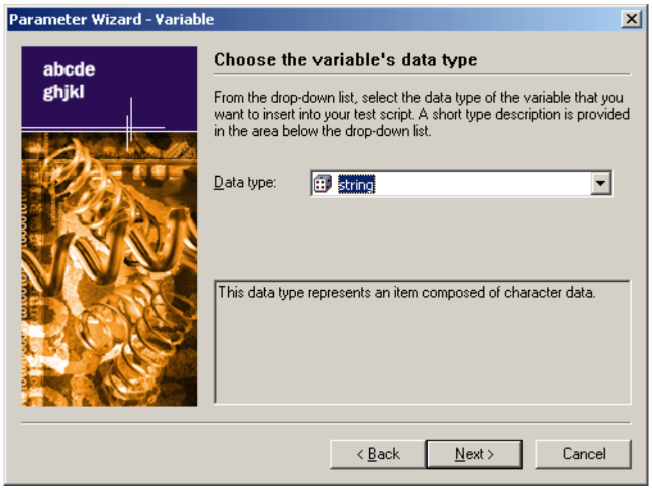

Parameter Wizard - Create New Parameter

Choose the type of parameter creation

Select the type of creation for the new variable.
The type of creation defines the value assignment of the variable.

Type of parameter creation
- ● Constant value
- ○ Parameter from Random Variable
- ○ Parameter from Multi-column Data File

6 The data type to be used is *string*. Click *Next*.


Parameter Wizard - Variable

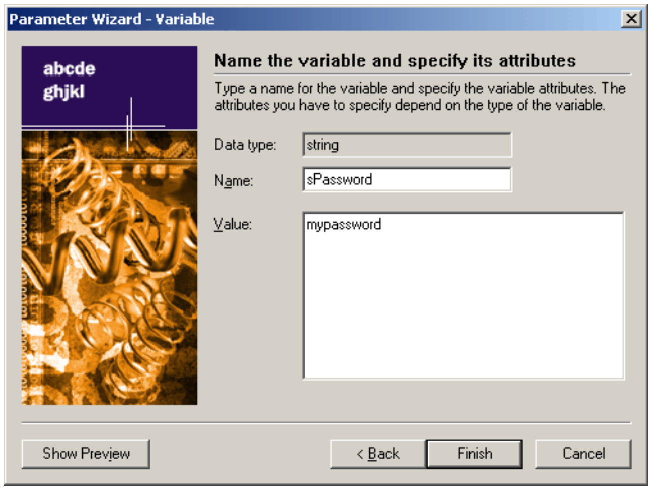
Choose the variable's data type

From the drop-down list, select the data type of the variable that you want to insert into your test script. A short type description is provided in the area below the drop-down list.

Data type: string

This data type represents an item composed of character data.

7 Define a meaningful *Name* for the new parameter and enter your user password as the string *Value*.


Parameter Wizard - Variable

Name the variable and specify its attributes

Type a name for the variable and specify the variable attributes. The attributes you have to specify depend on the type of the variable.

Data type: string

Name: sPassword

Value: mypassword

8 Now execute a new TryScript run. Your password parameter will automatically be inserted into the replayed test script and the script should run without error.

Customizing SAPGUI User Input Data

Under real world conditions, SAPGUI application users submit unpredictable combinations of data into forms. One goal of effective SAPGUI application testing is to emulate such irregular and diverse user behavior using test scripts.

You can customize the user input data that's entered into forms during testing with TrueLog Explorer's Parameter Wizard. The Parameter Wizard lets you specify values to be entered into form fields—enabling your test scripts to be more realistic by replacing recorded user input data with randomized, parameterized user data.

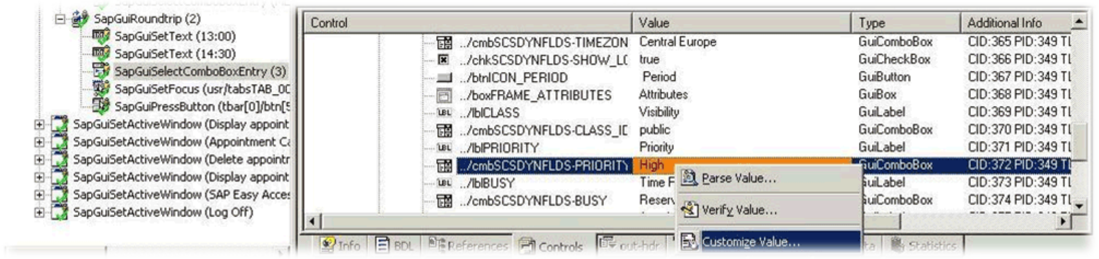
Procedure To customize user input data for a form field:

1 Select the *Step through TrueLog* toolbar button to display the *Step through TrueLog* dialog.

2 Select the *Customizable calls* radio button and click *Find Next* to step through all form fields in the TrueLog that offer input customization.

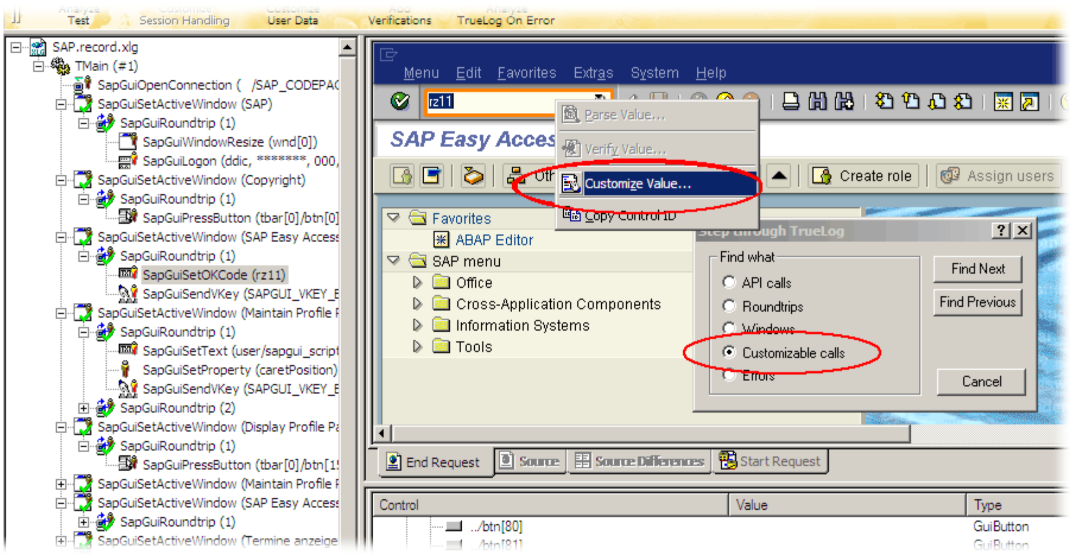
3 When you arrive at a control field that reflects user data input that you wish to customize, right-click in the control and select *Customize Value* from the context menu.

Note For this example, select the *Priority* field as shown in the figure below.


SapGuiRoundtrip (2)
SapGuiSetText (13:00)
SapGuiSetText (14:30)
SapGuiSelectComboBoxEntry (3)
SapGuiSetFocus (usr/tabsTAB_0C
SapGuiPressButton (tbar[0]/btn[5
SapGuiSetActiveWindow (Display appoint
SapGuiSetActiveWindow (Appointment Ca
SapGuiSetActiveWindow (Delete appointr
SapGuiSetActiveWindow (Display appoint
SapGuiSetActiveWindow (SAP Easy Acces
SapGuiSetActiveWindow (Log Off)

Control	Value	Type	Additional Info
../cmbSCSDYNFLDS-TIMEZON	Central Europe	GuiComboBox	CID:365 PID:349 TI
../chkSCSDYNFLDS-SHOW_LC	true	GuiCheckBox	CID:366 PID:349 TI
../btnICON_PERIOD	Period	GuiButton	CID:367 PID:349 TI
../boxFRAME_ATTRIBUTES	Attributes	GuiBox	CID:368 PID:349 TI
../lblCLASS	Visibility	GuiLabel	CID:369 PID:349 TI
../cmbSCSDYNFLDS-CLASS_IC	public	GuiComboBox	CID:370 PID:349 TI
../lblPRIORITY	Priority	GuiLabel	CID:371 PID:349 TI
../cmbSCSDYNFLDS-PRIORITY	High	GuiComboBox	CID:372 PID:349 TI
../lblBUSY	Time F	GuiLabel	CID:373 PID:349 TI
../cmbSCSDYNFLDS-BUSY	Reser	GuiComboBox	CID:374 PID:349 TI

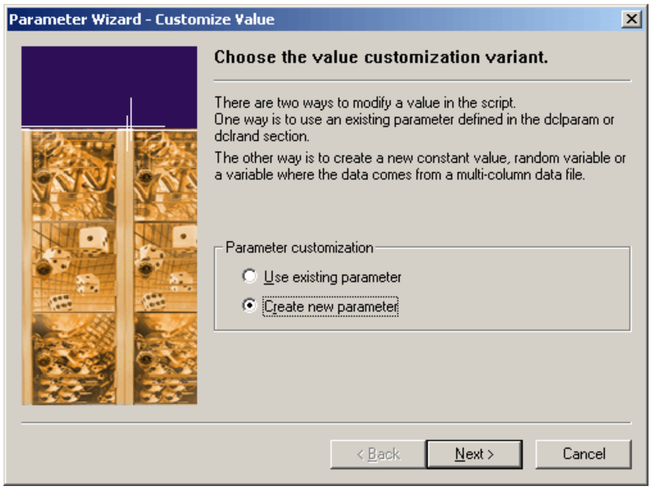
Note Controls that can be customized are outlined in orange. Controls that have already been customized are outlined in green. Controls that are outlined in blue can have their values parsed or verified, but they cannot be customized.


Test
Session Handling
User Data
Verifications
TrueLog On Error
SAP.record.xlg
TMain (#1)
SapGuiOpenConnection (/SAP_CODEPAG
SapGuiSetActiveWindow (SAP)
SapGuiRoundtrip (1)
SapGuiWindowResize (wnd[0])
SapGuiLogon (ddic, ********, 000,
SapGuiSetActiveWindow (Copyright)
SapGuiRoundtrip (1)
SapGuiPressButton (tbar[0]/btn[0]
SapGuiSetActiveWindow (SAP Easy Access
SapGuiRoundtrip (1)
SapGuiSetOKCode (rz11)
SapGuiSendVKey (SAPGUI_VKEY_E
SapGuiSetActiveWindow (Maintain Profile F
SapGuiRoundtrip (1)
SapGuiSetText (user/sapgui_script
SapGuiSetProperty (caretPosition)
SapGuiSendVKey (SAPGUI_VKEY_E
SapGuiRoundtrip (2)
SapGuiSetActiveWindow (Display Profile Pa
SapGuiRoundtrip (1)
SapGuiPressButton (tbar[0]/btn[1
SapGuiSetActiveWindow (Maintain Profile F
SapGuiSetActiveWindow (SAP Easy Access
SapGuiRoundtrip (1)
SapGuiSetActiveWindow (Termine anzeige
Menu Edit Favorites Extras System Help
rz11
SAP Easy Acces
Parse Value...
Verify Value...
Customize Value...
Copy Control ID
Create role
Assign users
Favorites
ABAP Editor
SAP menu
Office
Cross-Application Components
Information Systems
Tools
Step through TrueLog
Find what
API calls
Roundtrips
Windows
Customizable calls
Errors
Find Next
Find Previous
Cancel
End Request
Source
Source Differences
Start Request
Control
Value
Type
../btn[80]
GuiButton

4 With the Parameter Wizard you can modify script values in one of two ways. You can either use an existing parameter that's defined in the *dclparam* or *dclrand* section of your script, or you can create a new parameter (based on either a new constant value, a random variable, or values in a multi-column data file). Once you create a new parameter, that parameter is added to the existing parameters and becomes available for further customizations.

Note This example demonstrates the process of creating a parameter based on a new random variable. See the *SilkPerformer User Guide* for complete details regarding the functionality of the Parameter Wizard.

5 Select the *Create new parameter* radio button and click *Next* to create a new parameter.

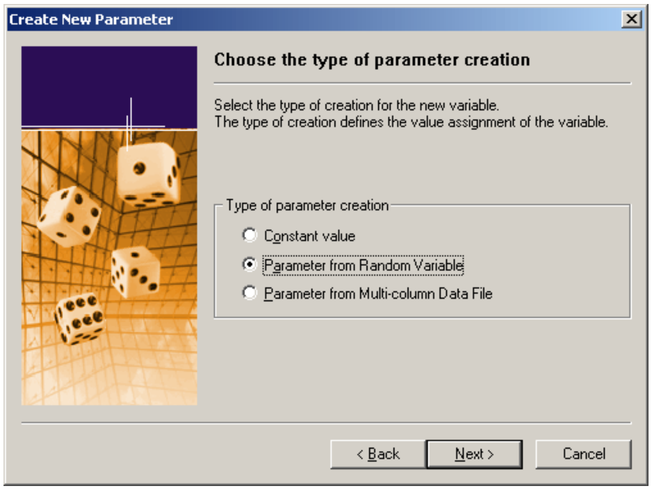

Parameter Wizard - Customize Value

Choose the value customization variant.

There are two ways to modify a value in the script.
One way is to use an existing parameter defined in the dclparam or dclrand section.
The other way is to create a new constant value, random variable or a variable where the data comes from a multi-column data file.

Parameter customization
- ( ) Use existing parameter
- (●) Create new parameter

6 The *Create New Parameter* dialog appears. Select the *Parameter from Random Variable* radio button and click *Next.*


Create New Parameter

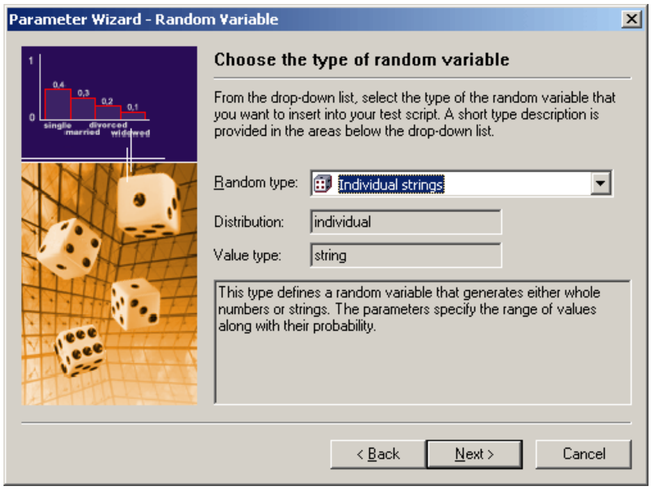
Choose the type of parameter creation

Select the type of creation for the new variable.
The type of creation defines the value assignment of the variable.

Type of parameter creation
- ( ) Constant value
- (●) Parameter from Random Variable
- ( ) Parameter from Multi-column Data File

7 The *Random Variable Wizard* appears with the *Individual strings* random variable type selected. A brief description of the highlighted variable type appears in the lower window.

8 Click *Next.*



Parameter Wizard - Random Variable

Choose the type of random variable

From the drop-down list, select the type of the random variable that you want to insert into your test script. A short type description is provided in the areas below the drop-down list.

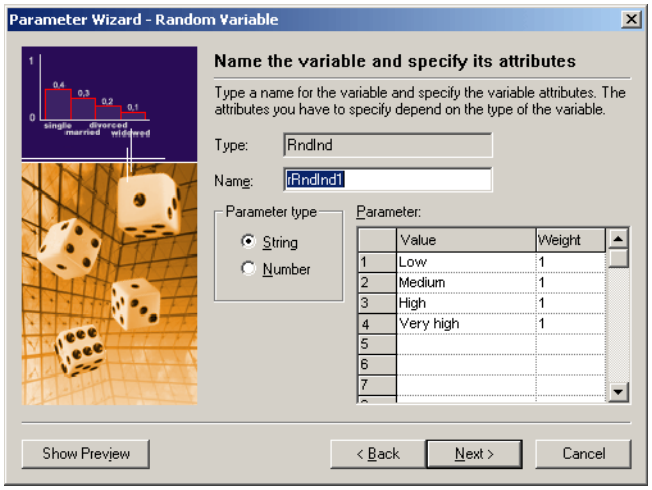
Random type: Individual strings

Distribution: individual

Value type: string

This type defines a random variable that generates either whole numbers or strings. The parameters specify the range of values along with their probability.

9 The *Name the variable and specify its attributes* screen appears. With SAPGUI applications, all available list-box values are pre-loaded with weight values of *1*. Enter a name for the variable in the *Name* field and click *Next.*



Parameter Wizard - Random Variable

Name the variable and specify its attributes

Type a name for the variable and specify the variable attributes. The attributes you have to specify depend on the type of the variable.

Type: RndInd

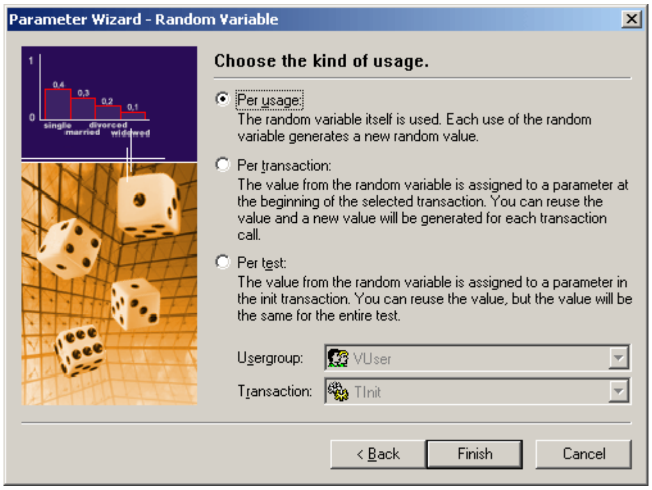
Name: rRndInd1

Parameter type: [✓ String] Number

Parameter:

	Value	Weight
1	Low	1
2	Medium	1
3	High	1
4	Very high	1
5		
6		
7		

10 *Per usage* random value generation is selected by default. Click *Finish*.



Choose the kind of usage.

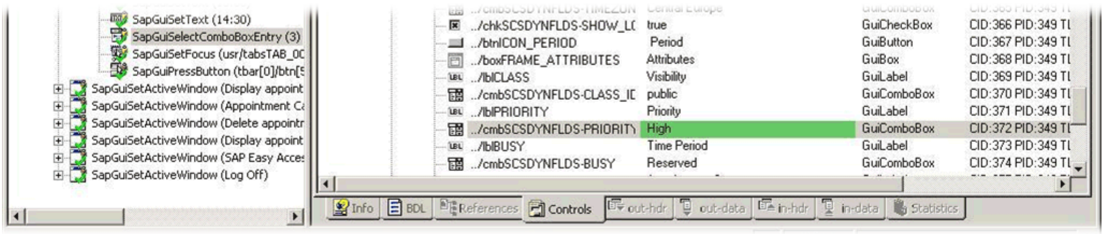
- (●) Per usage:
  The random variable itself is used. Each use of the random variable generates a new random value.
- ( ) Per transaction:
  The value from the random variable is assigned to a parameter at the beginning of the selected transaction. You can reuse the value and a new value will be generated for each transaction call.
- ( ) Per test:
  The value from the random variable is assigned to a parameter in the init transaction. You can reuse the value, but the value will be the same for the entire test.

Usergroup: VUser
Transaction: TInit

11 Click *Finish* to modify the BDL form declaration of your test script so that it uses the random variable for the given form field in place of the recorded value. The new random variable function appears below in *BDL* view.

12 Initiate a TryScript run with the random variable function in your test script to confirm that your script runs without error.

Note Controls that have been customized appear with green highlighting.



SapGuiSetText (14:30)
SapGuiSelectComboBoxEntry (3)
SapGuiSetFocus (usr/tabsTAB_0C
SapGuiPressButton (tbar[0]/btn[5
SapGuiSetActiveWindow (Display appoint
SapGuiSetActiveWindow (Appointment Ca
SapGuiSetActiveWindow (Delete appointn
SapGuiSetActiveWindow (Display appoint
SapGuiSetActiveWindow (SAP Easy Acces
SapGuiSetActiveWindow (Log Off)

../chkSCSDYNFLDS-SHOW_LC	true	GuiCheckBox	CID:366 PID:349 TI
../btnICON_PERIOD	Period	GuiButton	CID:367 PID:349 TI
../boxFRAME_ATTRIBUTES	Attributes	GuiBox	CID:368 PID:349 TI
../lblCLASS	Visibility	GuiLabel	CID:369 PID:349 TI
../cmbSCSDYNFLDS-CLASS_IC	public	GuiComboBox	CID:370 PID:349 TI
../lblPRIORITY	Priority	GuiLabel	CID:371 PID:349 TI
../cmbSCSDYNFLDS-PRIORITY	🟩 High	GuiComboBox	CID:372 PID:349 TI
../lblBUSY	Time Period	GuiLabel	CID:373 PID:349 TI
../cmbSCSDYNFLDS-BUSY	Reserved	GuiComboBox	CID:374 PID:349 TI

Additional customizations

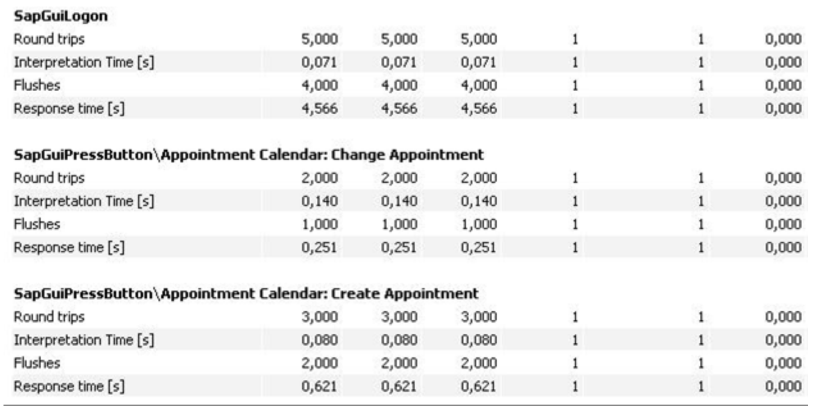
You may find that additional customizations are useful (e.g., randomizing username and appointment-time input parameters for load testing purposes). Customization is possible for nodes that involve changes of text, combo boxes, checkboxes, and radio-button controls. See the *TrueLog Explorer User Guide* for full details regarding available script customizations.

Note It's recommended that you not verify or parse values that occur in the last nodes of round-trips. This is because functions are scripted *after* selected API calls. For example, if you verify a *SAPGuiPressButton* function that closes the current window, the verification function will subsequently attempt to verify a control on a window that has already been closed—and a replay error will occur.

Analyzing Result Files

Each TryScript run generates an Overview Report (see example below).

Depending on measure settings in the active profile, measures are generated for method calls that have the optional timer parameter defined and also force a round-trip to the SAP server. Note that not all API calls force server round-trips.



SapGuiLogon						
Round trips	5,000	5,000	5,000	1	1	0,000
Interpretation Time [s]	0,071	0,071	0,071	1	1	0,000
Flushes	4,000	4,000	4,000	1	1	0,000
Response time [s]	4,566	4,566	4,566	1	1	0,000
SapGuiPressButton\Appointment Calendar: Change Appointment						
Round trips	2,000	2,000	2,000	1	1	0,000
Interpretation Time [s]	0,140	0,140	0,140	1	1	0,000
Flushes	1,000	1,000	1,000	1	1	0,000
Response time [s]	0,251	0,251	0,251	1	1	0,000
SapGuiPressButton\Appointment Calendar: Create Appointment						
Round trips	3,000	3,000	3,000	1	1	0,000
Interpretation Time [s]	0,080	0,080	0,080	1	1	0,000
Flushes	2,000	2,000	2,000	1	1	0,000
Response time [s]	0,621	0,621	0,621	1	1	0,000

Each server round-trip creates the following measures:

Round Trips
Before SAPGUI sends data to the server it locks the user interface. In many cases it will not unlock the interface after data is returned by the server, but instead sends a new request to the server. Controls use this technology to load data they need for visualization. A count of these token switches between SAPGUI and the server is offered with this measure.

Flushes
Counts the number of flushes in the automation queue during server communication.

Interpretation Time [s]
The interpretation time begins after data has arrived from the server. It comprises the parsing of the data and the distribution to the SAPGUI elements.

Response Time [s]
This is the time that is spent on network communication from the moment data is sent to the server to the moment the server response arrives.

Note An overall counter for all round trips is shown in SilkPerformer's Monitor window during load tests. This counter can also be monitored in Performance Explorer as a SilkPerformer Controller/Agent measure.

Further Steps for Load Testing

This tutorial offers only a brief overview of the steps that you may require for your load test scenario. Other steps that you will likely need to address are listed below. See the *SilkPerformer User Guide* for details regarding these additional steps:

- Run a baseline test
- Define your workload
- Setup your monitors
    - New SAPGUI monitor for monitoring SAP servers
- Run your load tests
- Analyze load test results

3

SAP eCATT Integration With SilkPerformer

Introduction

This chapter explains how to make the most of SilkPerformer's integration with SAP® eCATT (Extended Computer Aided Test Tool).

This chapter contains the following sections:

Overview

SAP *eCATT (Extended Computer Aided Test Tool)* has been integrated with SilkPerformer. SAP's eCATT facility allows you to create test scripts in SAP using the scripting language of your choice. eCATT allows you to use external test tools (i.e., SilkPerformer) while utilizing eCATT as a repository for your test scripts. eCATT also serves as a basic test management solution for triggering script executions. Not only can both internal and external scripts be executed individually, they can also be combined and executed in sequence.

eCATT offers *import arguments,* a mechanism for calling scripts with special input values. Scripts can not only receive input values, scripts can also set output values when they are executed—scripts can be executed in sequence, using input values derived from the output values of earlier script executions.

Note For more information regarding eCATT, please consult SAP documentation.

Setting Up Integration

This section includes detailed instructions for each of the steps that must be completed to make use of SilkPerformer's eCATT integration.

Procedure To configure SilkPerformer's eCATT integration:

1. On your SAP server, register SilkPerformer as an external tool for eCATT.
2. On your SAP server, create a new user account.
3. On the client machine where you will be using SilkPerformer in combination with eCATT, install both SilkPerformer and the SAPGUI client.
4. If you access your SAP server via a SAP gateway, you must create a registry key on the client that defines your default SAPGUI connection.
5. Within SilkPerformer system settings, configure SAP eCATT server connection data.
6. Within SilkPerformer system settings, define a SAP eCATT directory for extended SilkPerformer test results.

Registering SilkPerformer in eCATT

SilkPerformer must be registered in the *ECCUST_ET* SAP table. This is done by calling the *SET_EXTERNAL_TOOL* function module, which creates the necessary entries in the *ECCUST_ET* table. You need to method using the following values for the parameters:

Parameter	Value
TOOL_NAME	SilkPerformer
PROG_ID	SAPeCATTPlugIn.BorlandSPeCATT
TOOL_DESC	SilkPerformer for eCATT
TOOL_DATABASE	<blank>
TOOL_RUN_DB	<blank>
TOOL_NO_PWD	'X'
TOOL_NO_DB	'X'

You can call this method using the *SE37* transaction. On the first screen, enter the function module name *SET_EXTERNAL_TOOL.* Then select *Test / Single Test* from the *Function Module* menu.

In the following window, enter the parameter values as described above and press the *Execute* button (F8).

Creating a specific user account

To take advantage of eCATT integration using external tools, a standard user must be generated in your system by your system administrator. This is done by executing the *ECATT_GENERATE_ET_USER* program in SE38 (once per system).

After running the report, the following steps should be executed to activate the newly created user role:

1. In transaction *PFCG,* enter role *SAP_ECET,* and select *Change.* Ignore the subsequent warning that appears.
2. Switch to the *Authorizations* tab and select *Change Authorization Data.*
3. Place your cursor over the top node of the tree display (SAP_ECET) and select *Authorizations / Generate.*
4. Click *Back* to return to the role maintenance screen.
5. Click *Save.*

Installing the client software

On the machine where you plan to use SilkPerformer with the eCATT integration, you must first install your SAPGUI client and afterward apply the SilkPerformer installation. Whenever eCATT initiates the integration between SilkPerformer and eCATT, SilkPerformer installs a COM object on the agent that is called by SAP eCATT.

Setting the registry from behind a SAP gateway

If you are accessing your SAP system via a SAP gateway, you must create a registry key for the communication between SilkPerformer and eCATT. eCATT forwards the connection that is to be used to SilkPerformer, but it is unaware of gateways. Therefore the connection string that is passed from eCATT cannot be used if you are behind a gateway.

You have to create a registry key under *HKLM\Software\Silk.* The key must be a string value with the name *SAPeCATTLogonID*; the value must be the SAP login ID that you use when logging in to your system (i.e., the name of your SAP connection that you specify in *SAPLOGON).*

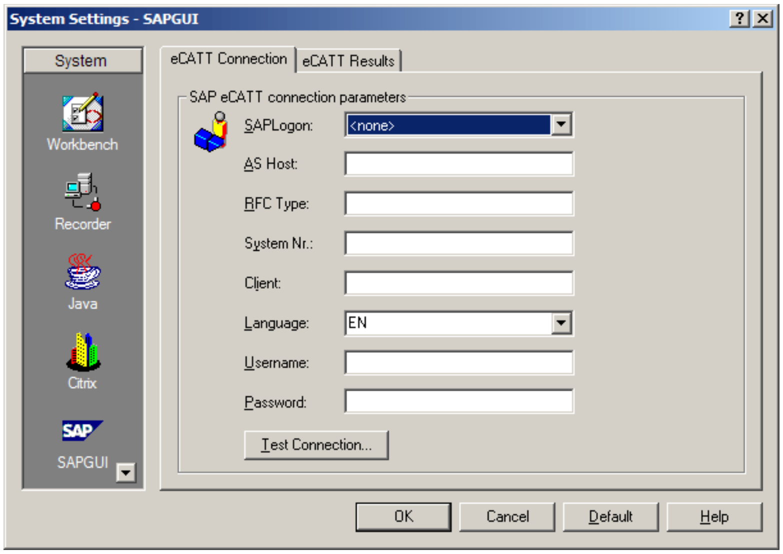
Configuring SAP eCATT connection

Connection details for SilkPerformer's communication with SAP eCATT must be specified in SilkPerformer system settings. There are two options for connecting to SAP—you can either specify a *SAPLOGONID* or you can specify *AS Host, RFC Type,* and *SystemNr* settings. With either option you must specify client, language, username, and password details. Note that when you select a SAPLOGONID the *AS Host, RFC Type,* and *SystemNr* fields are grayed out.

Procedure To specify SAP eCATT connection data:

1. Select the **System** command from SilkPerformer's **Settings** menu.
2. On the System Settings - Workbench dialog, select the **SAPGUI** group icon.
3. The **eCATT Connection** tab is selected by default. From the **SAPLogon** drop box, select your SAP login ID. This box is preconfigured with all available SAP login IDs.
4. In the **AS Host** edit field, enter the combined router/application-server string (e.g., H/195.61.176.22/H/194.117.106.130/S/3297/H/cpce801).
5. In the **RFC Type** edit field, enter either '3' (for R/3) or '2' (for R/2).
6. In the **System NR** edit field, enter the SAP system number.
7. In the **Client** edit field, enter the internal client ID number from the SAP server (i.e., the value that must be entered on the SAP login screen).
8. From the **Language** drop box, select your language preference. The values 'EN' (English) and 'DE' (German) are preconfigured, though you can specify any other language abbreviation string.
9. In the **Username** edit field, enter your SAP eCATT username.
10. In the **Password** edit field, enter your SAP eCATT password.
11. Once you have completed this dialog, click **Test Connection** to confirm that you have specified accurate connection details. If your connection attempt is unsuccessful, please confirm your settings.

12 Click the **OK** button once you have completed configuring SAP eCATT connection settings.



SAP eCATT connection parameters

SAPLogon:	<none>
AS Host:	
RFC Type:	
System Nr.:	
Client:	
Language:	EN
Username:	
Password:	

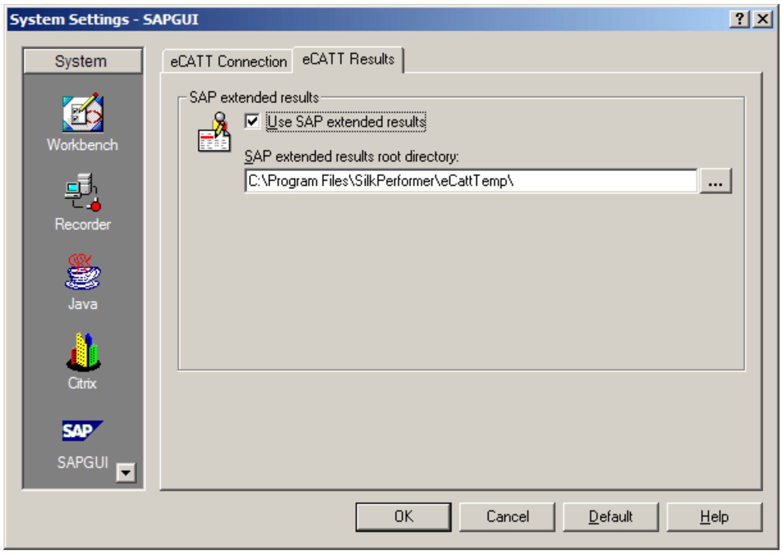
Configuring eCATT extended results

To enable the viewing of SilkPerformer result files from within SAPGUI, you can specify a UNC path to a public file share in which extended SilkPerformer test results can be stored and accessed by users (e.g., \\fileserver\ecattresults). SilkPerformer will use the specified directory to store the results of SilkPerformer test executions initiated via SAP eCATT. Users can easily access test results by clicking a link in the SAP eCATT GUI.

Procedure To configure eCATT extended results:

1. Select the **System** command from SilkPerformer's **Settings** menu.
2. On the System Settings – Workbench dialog, select the **SAPGUI** group icon.
3. Select the **eCATT Results** tab.
4. Select the **Use SAP extended results** checkbox.
5. In the **SAP extended results directory** field, browse to and select the directory that is to be used for SAP extended results.

6 Click **OK** to save your settings.


System Settings - SAPGUI

System | eCATT Connection | eCATT Results

SAP extended results

☑ Use SAP extended results

SAP extended results root directory: C:\Program Files\SilkPerformer\eCattTemp\

Interacting with eCATT from SilkPerformer

SilkPerformer can be run in *eCATT Standalone mode.* In this mode, SilkPerformer can be used to:

- Upload projects to SAP eCATT
- Open projects from SAP eCATT

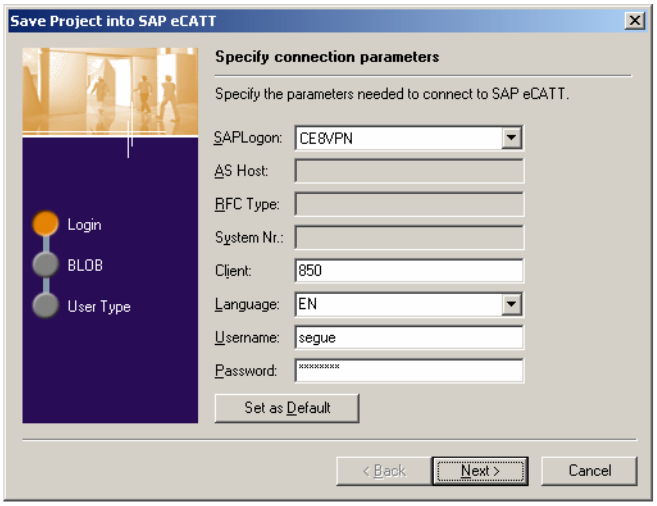
SilkPerformer also allows you to specify import and export arguments that can be used to exchange values between eCATT scripts.

Uploading a project to eCATT

The menu entry *Upload Project into eCATT* from the *File / SAP eCATT* menu allows you to both upload new projects to SAP and update existing projects.

When uploading a project, the project is exported with all of its dependent files (e.g., data files, include files) and uploaded to SAP eCATT to a *BlobId* and *Version* that you define in the *Save Project into SAP eCATT* wizard.

The first step prompts you for the SAP server connection that is to be used for uploading the project:

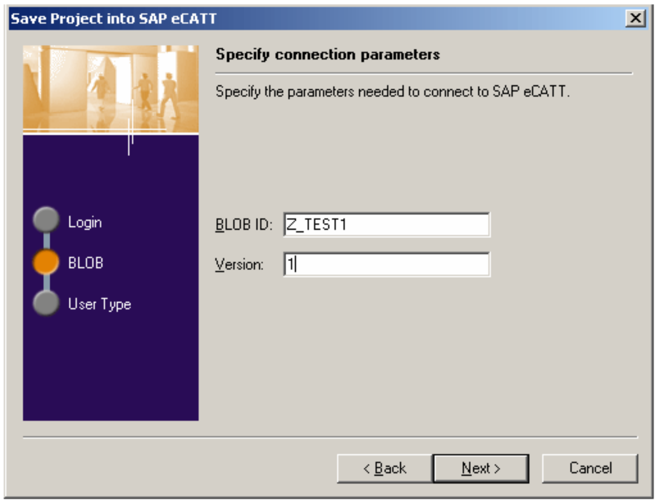


Specify connection parameters

Specify the parameters needed to connect to SAP eCATT.

SAPLogon:	CE8VPN
AS Host:	
RFC Type:	
System Nr.:	
Client:	850
Language:	EN
Username:	segue
Password:	********

The second wizard step prompts you for the *BlobId* and *Version:*

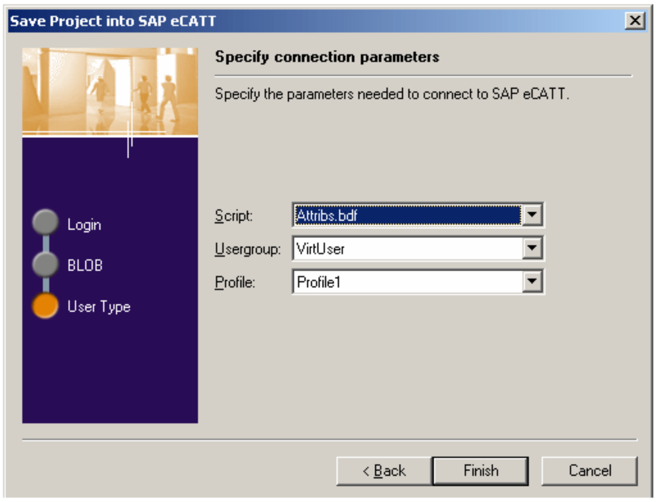


Specify connection parameters

Specify the parameters needed to connect to SAP eCATT.

BLOB ID:	Z_TEST1
Version:	1

In the third step you have to define the user type that is to be the primary user type in the script. As you may have multiple user types configured in your project, you must define the user type that is to be executed by default when eCATT executes scripts:



Specify connection parameters

Specify the parameters needed to connect to SAP eCATT.

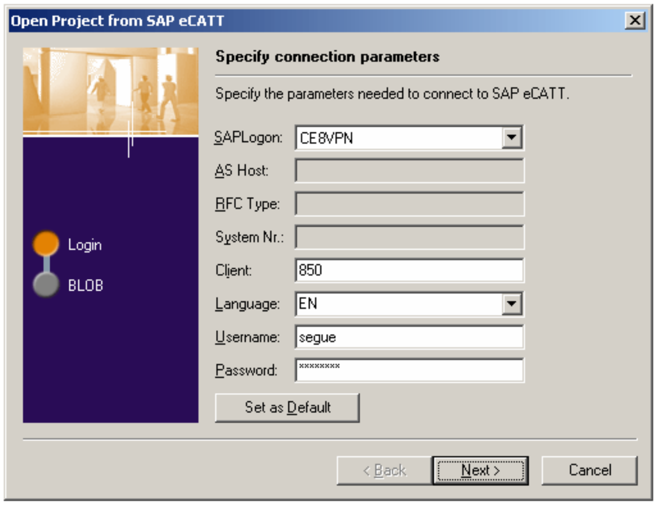
Script: Attribs.bdf
Usergroup: VirtUser
Profile: Profile1

Opening a project from eCATT

The menu entry *Open Project from eCATT* from the *File / SAP eCATT* menu allows you to open an existing project from SAP eCATT.

When opening a project, the project is downloaded to a temporary directory and then imported to SilkPerformer Workbench. A downloaded project is identified by the *BlobId* and *Version* that you define in the *Open Project from SAP eCATT* wizard.

The first step in the wizard prompts you for the SAP server connection that is to be used to download the project:

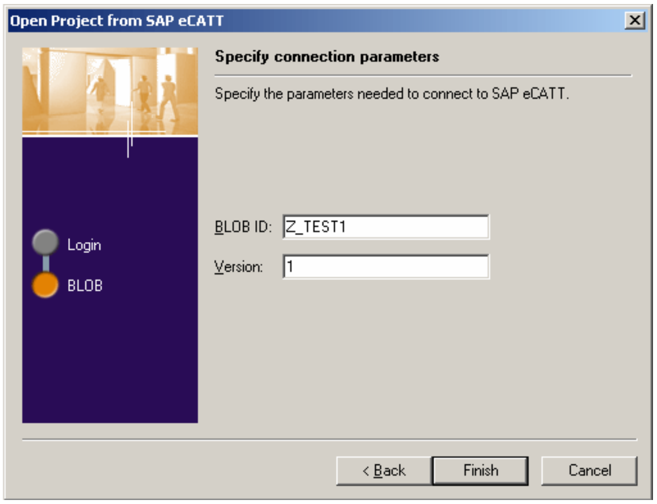


Specify connection parameters

Specify the parameters needed to connect to SAP eCATT.

SAPLogon: CE8VPN
AS Host:
RFC Type:
System Nr.:
Client: 850
Language: EN
Username: segue
Password: ********

The second step prompts you for the *BlobId* and *Version* of the project that is to be downloaded.



Specify connection parameters

Specify the parameters needed to connect to SAP eCATT.

BLOB ID: Z_TEST1

Version: 1

Now you can make changes to the project. If you want to update the project in eCATT after you have completed your changes, simply upload the project using the *Upload Project into SAP eCATT* command on the *SAP eCATT* menu.

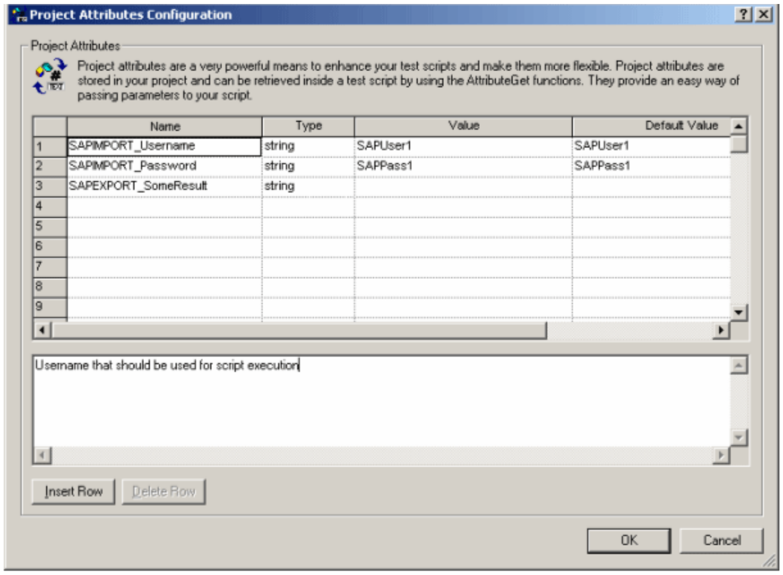
Defining import/export arguments

eCATT offers *import arguments,* a mechanism for calling scripts with special input values. Scripts can not only receive input values, scripts can also set output values when they are executed—scripts can be executed in sequence, using input values derived from the output values of earlier script executions.

To define import and export arguments, SilkPerformer project attributes are used. Project attributes that serve as input arguments must have the prefix *SAPIMPORT_*. Project attributes that serve as output arguments must have the prefix *SAPEXPORT_*.

> **Note** Only project attributes of type *string* are accepted, since SAP only allows string data types.

Following is an example that defines two input arguments and one output argument:



Project attributes are a very powerful means to enhance your test scripts and make them more flexible. Project attributes are stored in your project and can be retrieved inside a test script by using the AttributeGet functions. They provide an easy way of passing parameters to your script.

	Name	Type	Value	Default Value
1	SAPIMPORT_Username	string	SAPUser1	SAPUser1
2	SAPIMPORT_Password	string	SAPPass1	SAPPass1
3	SAPEXPORT_SomeResult	string		
4				
5				
6				
7				
8				
9				

Username that should be used for script execution

In a script you access these input values as follows:

```
AttributeGetString("SAPIMPORT_Username", sUsername);
AttributeGetString("SAPIMPORT_Password", sPassword);
```

The output value is set as follows:

```
AttributeSetString("SAPEXPORT_SomeResult", "thats my
result");
```

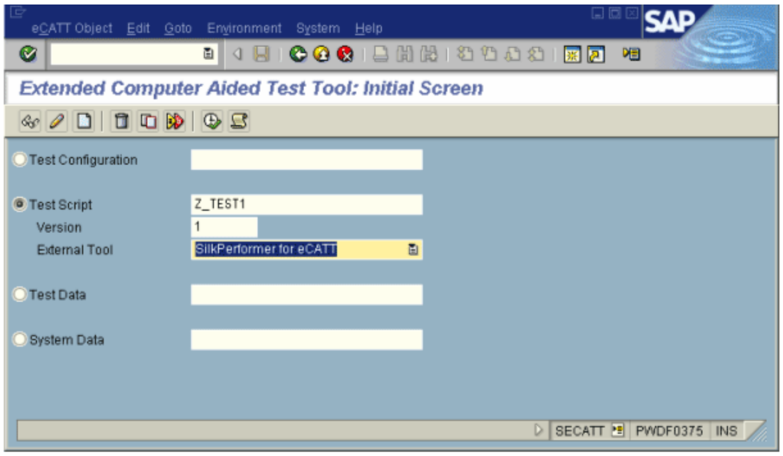
Interacting with SilkPerformer from eCATT

From eCATT it’s possible to utilize SilkPerformer as an external test tool by:

- Creating a new SilkPerformer script
- Editing/Viewing an existing SilkPerformer script
- Executing a script in one of three modes:
    - “Normal” without SilkPerformer Workbench
    - “Debug Mode” with SilkPerformer Workbench and the option of adjusting settings before executing the script
    - “With Surface of External Tool” with SilkPerformer Workbench

Creating a new SilkPerformer script

Within SAP eCATT (SECATT transaction) you can create a new SilkPerformer script by specifying a test script name (blobID), a version, and *SilkPerformer for eCATT* as the external tool, as shown below:



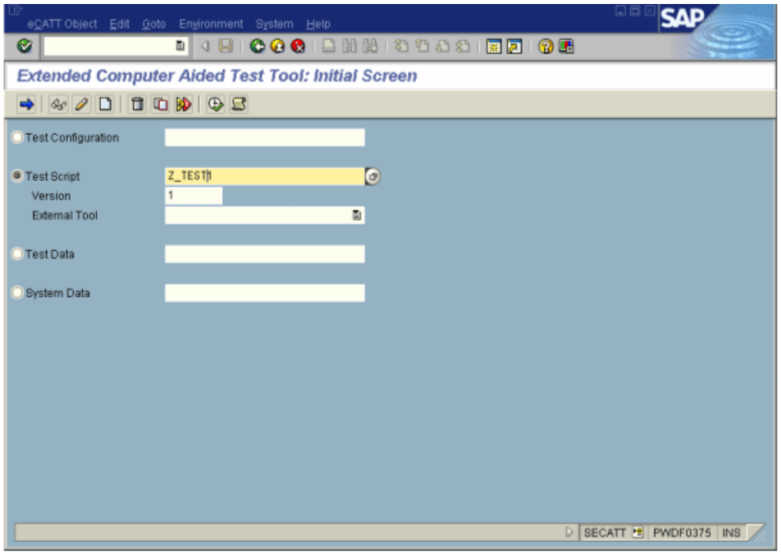
Extended Computer Aided Test Tool: Initial Screen

Test Configuration	
● Test Script	Z_TEST1
Version	1
External Tool	SilkPerformer for eCATT
Test Data	
System Data	

Now click the *Create Object* button (or press F5). This creates an empty SilkPerformer script and stores it in the eCATT repository. On the following screen you must enter all required fields before you can edit the script with SilkPerformer. Once you have completed all required fields, click the *Script* toolbar button. SilkPerformer then opens and downloads the newly created project in Edit mode (for other available options, please see the next chapter).

Editing/viewing an existing SilkPerformer script

Within SAP eCATT (SECATT transaction) you can both edit and view an existing SilkPerformer script by specifying the test script name (blobID) and the version, as shown below:


eCATT Object Edit Goto Environment System Help
SAP
Extended Computer Aided Test Tool: Initial Screen
SECATT
PWDF0375
INS

Test Configuration	
● Test Script	Z_TEST1
Version	1
External Tool	
Test Data	
System Data	

Now you can either click the *Display Object* (F7) button or the *Change Object* (F6) button to view or edit the eCATT script. On the following screen you can click the *Script* button to either view or edit the script in SilkPerformer.

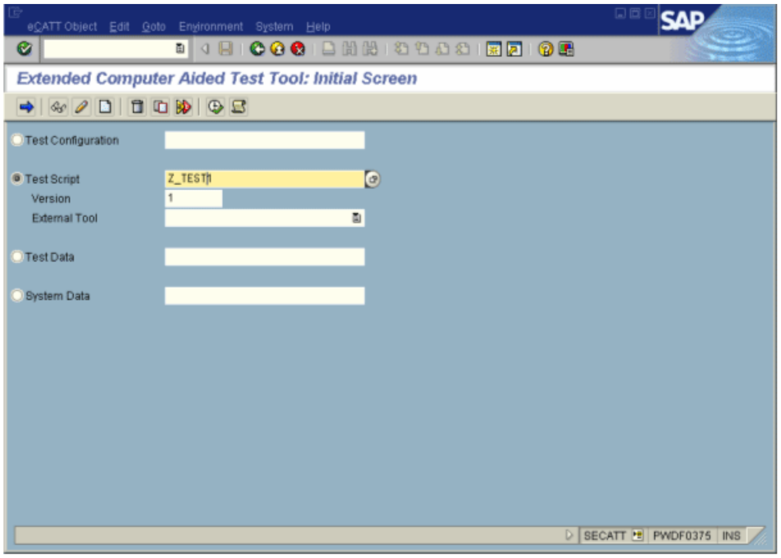
If you are only viewing the script, SilkPerformer downloads the project from SAP eCATT and opens it in Read-Only mode. You can go back to eCATT by selecting *Close Project without Save* from the *SAP eCATT* menu.

If you are opening the script in Edit mode, SilkPerformer downloads the project from SAP eCATT so that you can modify the script. Once you have completed your modifications you have three options for returning to SAP eCATT—select one of the following from the *SAP eCATT* menu:

- *Save Project into SAP eCATT*
  The project is saved to SAP eCATT and you can continue working in SAP eCATT.
- *Save Project into SAP eCATT and Continue*
  The project is saved to SAP eCATT, but remains open in SilkPerformer so that you can perform further modifications.
- *Close Project without Save*
  The project is closed without saving your changes and you can continue working in SAP eCATT.

Executing a SilkPerformer script

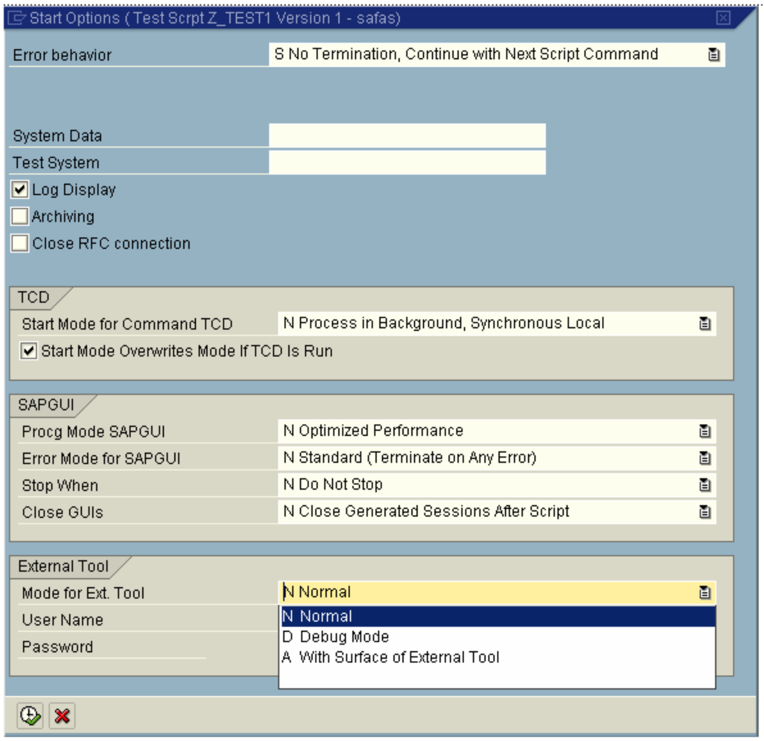
Within SAP eCATT (SECATT transaction) you can execute a SilkPerformer script by specifying a test script name (blobID) and version, as shown below:



Extended Computer Aided Test Tool: Initial Screen

Test Configuration
[✓ Test Script] Z_TEST
Version 1
External Tool
Test Data
System Data

Once you have entered these values, click the *Execute* button (F8) to go to the execution dialog where you can specify start options for the script:



Start Options (Test Scrpt Z_TEST1 Version 1 - safas)

Error behavior	S No Termination, Continue with Next Script Command
System Data	
Test System	

☑ Log Display
☐ Archiving
☐ Close RFC connection

TCD

Start Mode for Command TCD	N Process in Background, Synchronous Local

☑ Start Mode Overwrites Mode If TCD Is Run

SAPGUI

Procg Mode SAPGUI	N Optimized Performance
Error Mode for SAPGUI	N Standard (Terminate on Any Error)
Stop When	N Do Not Stop
Close GUIs	N Close Generated Sessions After Script

External Tool

Mode for Ext. Tool	N Normal
User Name	
Password	

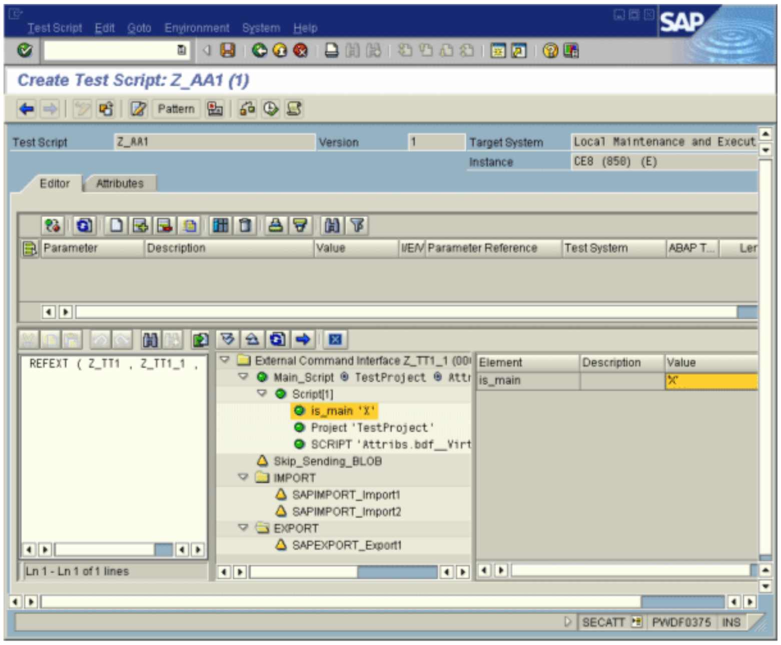
Depending on the *Mode for Ext. Tool* selection, the test will either be executed without SilkPerformer Workbench *(Normal),* with SilkPerformer Workbench *(With Surface of External Tool),* or in the attended debug mode with SilkPerformer Workbench *(Debug Mode).*

In *With Surface of External Tool* mode, SilkPerformer launches and immediately executes the primary user type. After executing, SilkPerformer closes.

In Debug mode, SilkPerformer opens the *Baseline test* dialog and waits for the user to begin the test. The user can perform some modifications to the project before the test is actually executed. When the test is finished, the user can use the *Finish and Return Results* command on the *SAP eCATT* menu to report back the results of the most recently executed test run.

As with other eCATT scripts, SilkPerformer scripts can be executed from other eCATT scripts and executed in sequence with other scripts.

To do this, create a simple eCATT script that calls an external script using the *REFEXT* method. Once you have specified the blobId and version of the script that you want to execute, double-click the second parameter of *REFEXT* and explore the external project (see the example below):



Create Test Script: Z_AA1 (1)

Test Script	Z_AA1	Version	1	Target System	Local Maintenance and Execut
				Instance	CE8 (850) (E)

```
REFEXT ( Z_TT1 , Z_TT1_1 ,
```

Ln 1 - Ln 1 of 1 lines

```
External Command Interface Z_TT1_1 (00
  Main_Script ◉ TestProject ◉ Attr
    Script[1]
      is_main 'X'
      Project 'TestProject'
      SCRIPT 'Attribs.bdf__Virt
  Skip_Sending_BLOB
  IMPORT
    SAPIMPORT_Import1
    SAPIMPORT_Import2
  EXPORT
    SAPEXPORT_Export1
```

Element	Description	Value
is_main		'X'

You will see the various scripts that are there-one should be marked is_main. You will also see all import and export arguments.

Limitations

The following points should be considered when working with SilkPerformer's eCATT integration.

Script name length

When uploading a SilkPerformer project, you must specify the *primary script.* This is the user type that will be executed by default and marked as the main script within eCATT. The name of the user type is a combination of the BDL script name, user group, and profile (e.g., a script called *test1.bdf* that defines a user group called *VUser*, and a profile called *Profile1,* results in an internal representation of the user type as *test1.bdf__VUser__Profile1).*

When SAP triggers SilkPerformer to execute a script, SAP specifies the name of the primary script as shown in the example above, *(test1.bdf__VUser__ Profile1).*

A bug in the current versions of SAP eCATT truncates such passed values to 32 characters. Though SAP will likely address this issue in future patches, it is quite possible that you are running a SAP system with this limitation.

The problem with the 32-character limitation is that during execution in normal mode, SilkPerformer can not find the passed user type because the name has been truncated. Therefore you will receive an error indicating that the script can not be found.

To work around this problem, make sure that the combination of script name, user group, and profile does not exceed 32 characters—28 characters in fact as 4 underscore characters are used to separate the values.

So save your script files with short names, use short names for user groups, and use short names for profiles.

Default values for arguments

During testing efforts with various SAP systems, a problem with default values in eCATT arguments has been identified. With some older patch levels of SAP eCATT, default values are not passed to SilkPerformer when running script executions from external eCATT scripts. If you experience a problem of default values not being passed for arguments, update your SAP eCATT patches.

Recording in edit-mode

When recording a SAPGUI script in SilkPerformer while in the “edit” mode of an eCATT script triggered from within SAP eCATT, you run into the following problem: When you begin a new session during recording you actually have two SAP connections open on your system—the connection that you are recording on and the connection that is still open from eCATT. Therefore you will see two SapGuiOpenConnection calls scripted in your script, and the SapGuiSetActiveConnection contains a connection ID of 1. This is what you will see in your recorded script:

```
    gsConnID := SapGuiOpenConnection(" ecatt connection",
"SapGuiOpenConnection");
    SapGuiOpenConnection("  recorded connection",
"SapGuiOpenConnection");
    SapGuiSetActiveConnection("/app/con[1]");
```

You must remove the first SapGuiOpenConnection entry and change the SapGuiSetActiveConnection to use the return value of the 2nd SapGuiOpenConnection. After this modification, the script should look like this:

```
    gsConnID := SapGuiOpenConnection("  recorded
connection", "SapGuiOpenConnection");
    SapGuiSetActiveConnection(gsConnID);
```

Index