Silk Performer 18.5

SAPGUI Tutorial

2017-07-10

Contents

Introduction

Overview

Silk Performer offers recording and replay support for the load testing and functional testing of SAP® systems that use the SAPGUI Scripting interface. Silk Performer'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, Silk Performer supports load and functional testing of SAP R/3 4.6C, SAP R/3 Enterprise (4.7), and mySAP Business Suite (and higher) through 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 Silk Performer and TrueLog Explorer. Please see the *Silk Performer Help* and the *TrueLog Explorer Help* 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, Silk Performer 18.5 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

Silk Performer 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 Silk Performer'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

Silk Performer 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 Silk Performer while utilizing eCATT as a repository for your test scripts. See "SAP eCATT Integration With Silk Performer" 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 Silk Performer 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. Make sure that you have the latest SAPGUI patch level.

Procedure To confirm the SAPGUI patch level:

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

2 The SAPGUI version information dialog box opens and displays the current patch level.

Profile settings Silk Performer SAPGUI support is configurable through Silk Performer profile settings.

> **Note** Make sure that in replay profile settings the *Log control information in TrueLog* option is not checked. When this option is turned on during load tests, each virtual user builds TrueLog with information for all controls on every window in transactions. Depending on your transactions and the number of controls that are on-screen, you may experience heavy performance impact with this setting enabled.

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, for example setting username, setting password, and hitting ENTER, are scripted.

Script low level functions
Rather than scripting high-level API functions, for example SapGuiSetText, low-level API functions are scripted, for example SapGuiInvokeMethod and 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, a transaction-exit error is thrown and the affected VUser is restarted.

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 through 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, Silk Performer 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 through the SAP Configuration Tool. By enabling/disabling this option, Silk Performer 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 Silk Performer 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 6 or 7
- Latest patch level
- SAPGUI Scripting must be installed and enabled
    - To enable SAPGUI Scripting:
        - o Start the SAPGUI client.
        - o Open the *Options* dialog.
        - o Select the *Scripting* tab.
        - o Select *Enable Scripting* and uncheck the two security check boxes.

Available Functions

Silk Performer 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 Silk Performer Online Help.

Silk Performer offers both a low- and high-level API for testing SAP systems. Please see the *Silk Performer 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:

Overview

Silk Performer 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 Silk Performer's *Outline Project* button to create a new project.

2   Enter a project *Name* and enter an optional project description in *Description.*

3   Select *ERP/CRM\SAP\SAPGUI* as the application *Type.*

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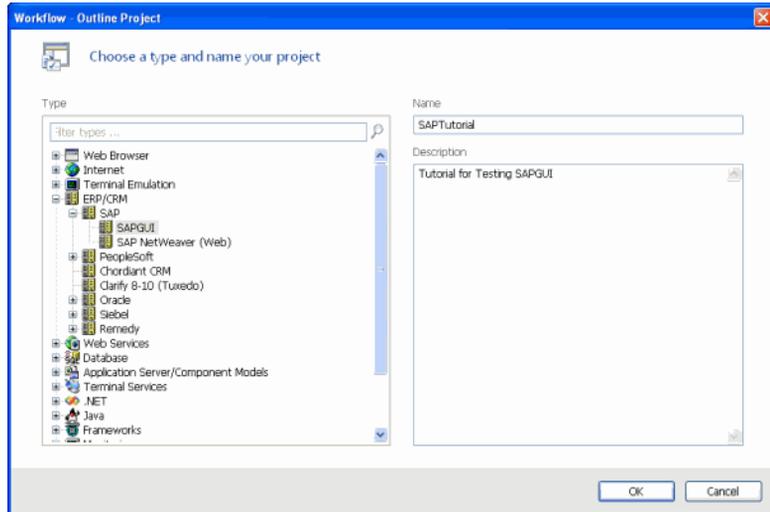
5   Click the *Model Script* button on the Workflow Bar.

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

If SAPGUI is still not listed in the *Application Profile* field, you need to create the application profile for *saplogon.exe*. In Silk Performer, open the *System Settings* dialog box 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 check box. Click *OK* to confirm your settings and close the *System Settings* dialog box.

7   Click *Start recording* to launch *saplogon.exe.*

8 Specify the SAP application server that is to be tested. This tutorial illustrates the testing of a SAP calendar application.


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, for example 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 box of each SAP connection in the SAPGUI login application.
Edit the properties of each connection and click *Advanced*.
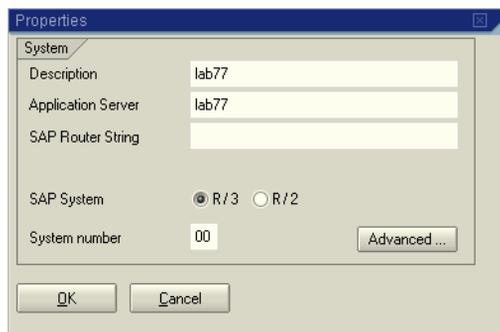
On the *Advanced Options* dialog box, deselect *Default codepage* and select your preferred language. Click *OK* to accept the change.
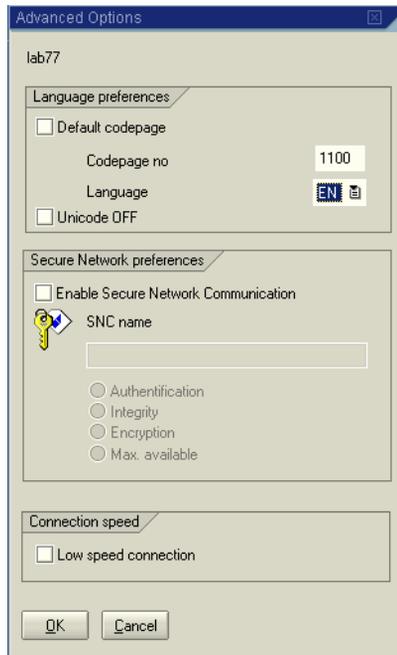

9   Now simulate the actions of a typical user transaction. Login with username and password. Hit *Enter*.
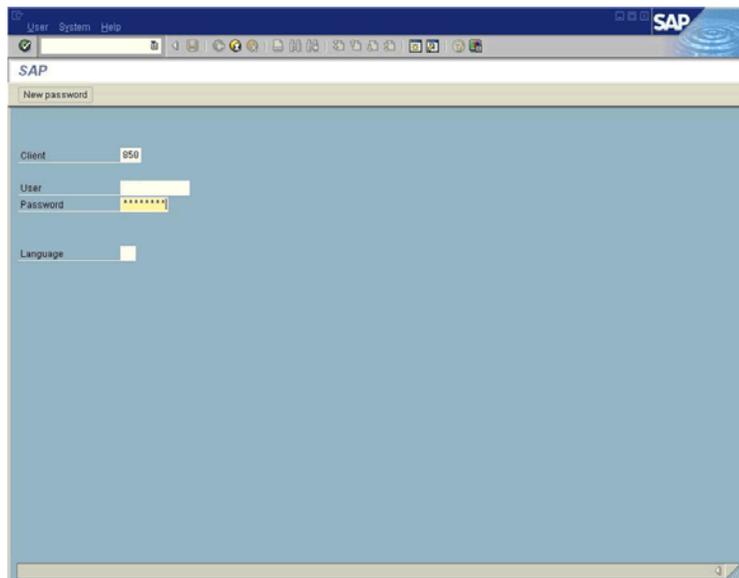
10 Expand the tree and double-click the *Owner* item.
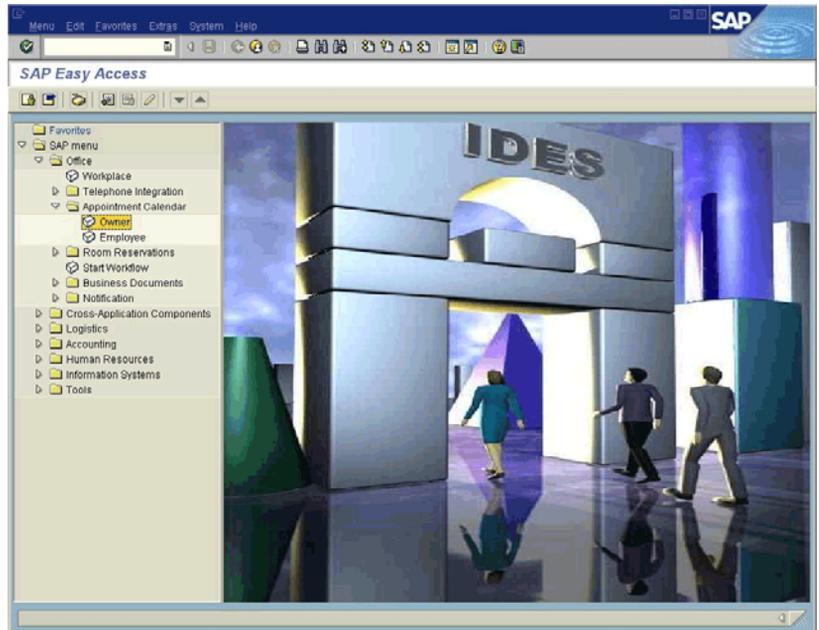
11 Right-click a time frame and select *Create Appointment* to create a new appointment.
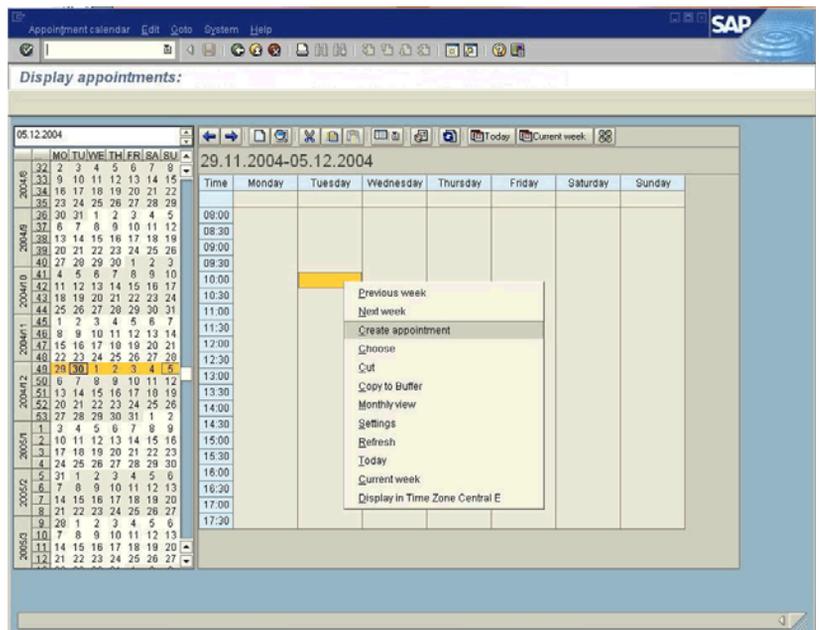
12 Define the appointment by setting *Title, Appointment Type, Time,* and *Priority.* Then click *Save.*

13 Now edit the appointment you created by double-clicking it.
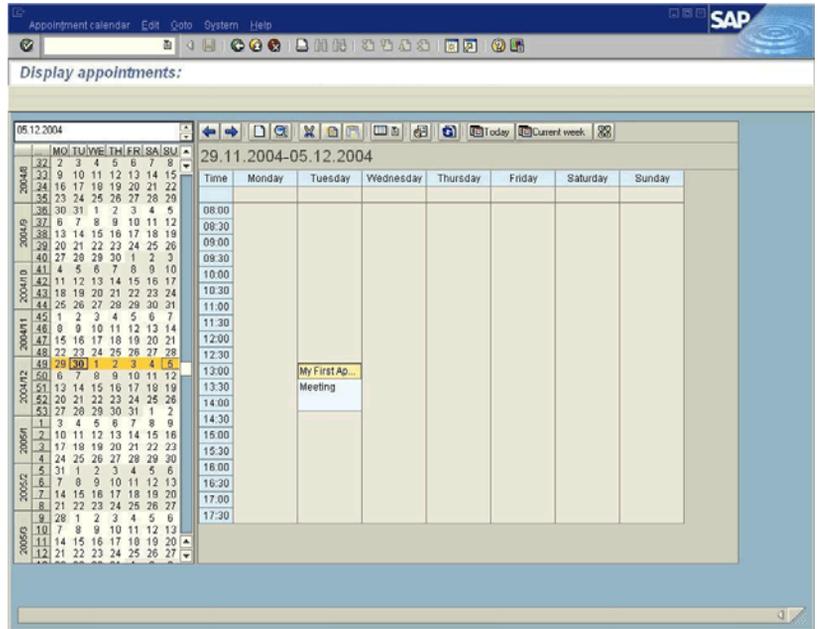
14 Click the *Delete* button (trash can icon) to delete the appointment. A confirmation dialog box opens. Click *Yes* to confirm deletion of the appointment.
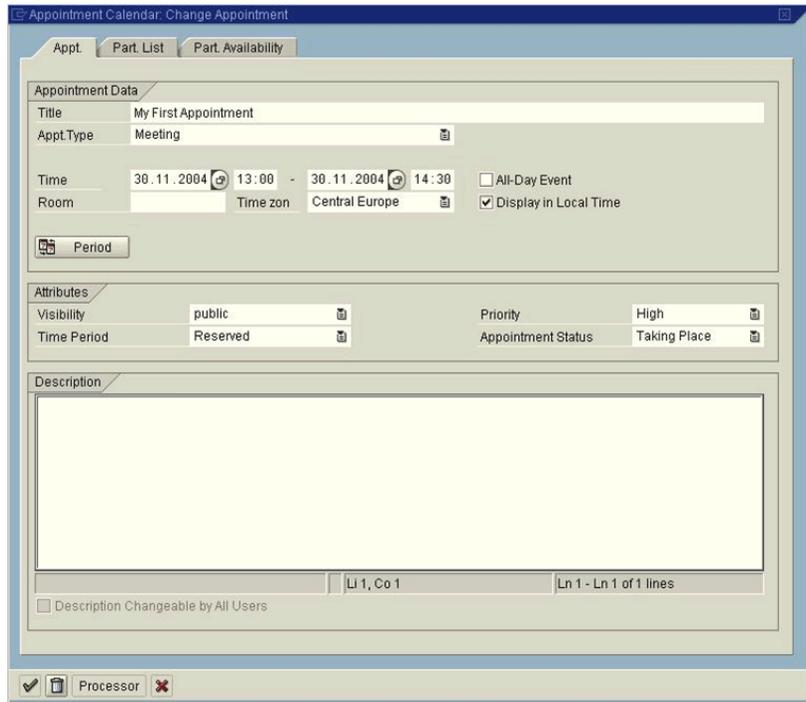
15 Click *Exit* (the yellow circular button) to exit the calendar application.
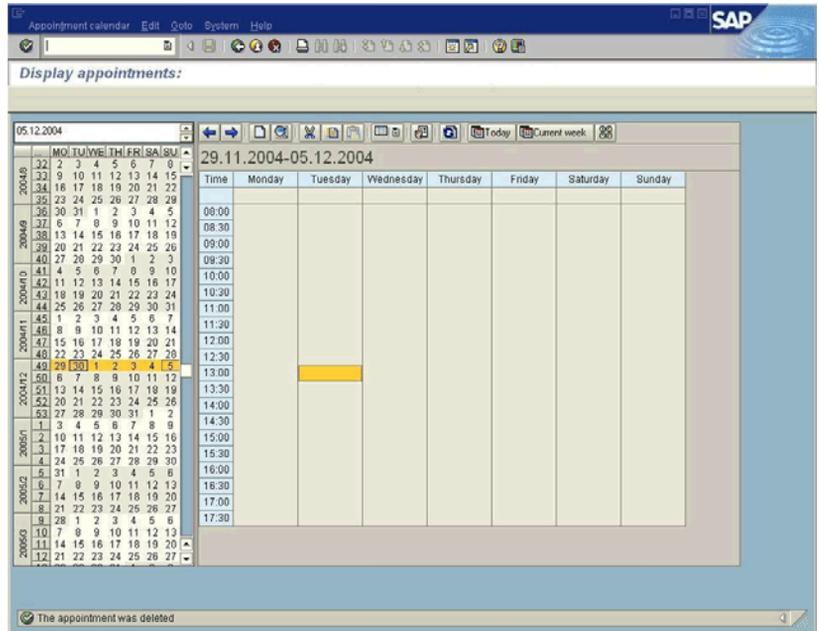
16 Click *Exit* again to exit SAP Workbench. On the following confirmation dialog box, confirm that you wish to exit the application.
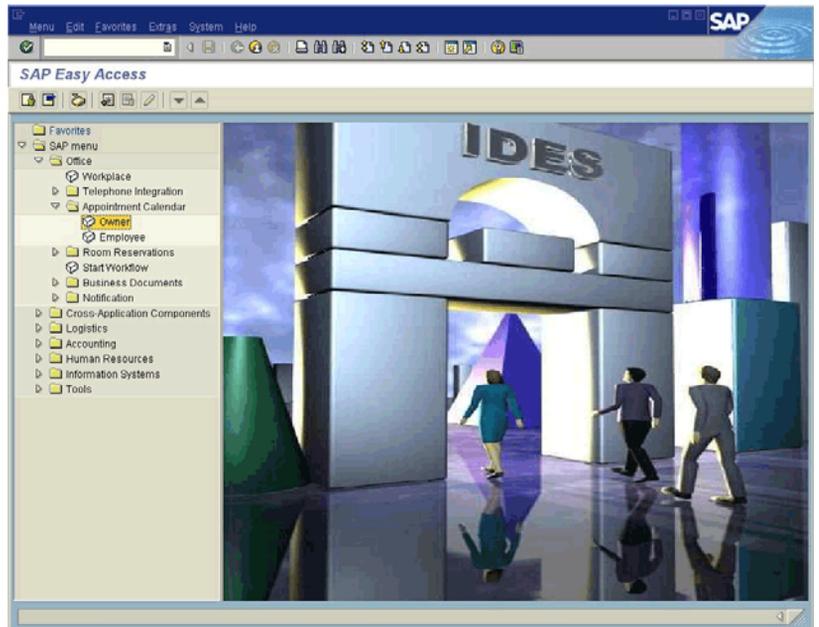
17 Close the SAPLogon application. This ends your simulated user transaction.

18 Stop Silk Performer's Recorder.

19 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.

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 *Try Script* on the Silk Performer Workflow bar. The *Try Script* dialog box opens.

2   To view rendered page transitions during a TryScript run, select the *Animated Run with TrueLog Explorer* check box.

3   Click *Run.*

   Note You are not running an actual load test here, only a test run to see if your script requires debugging.
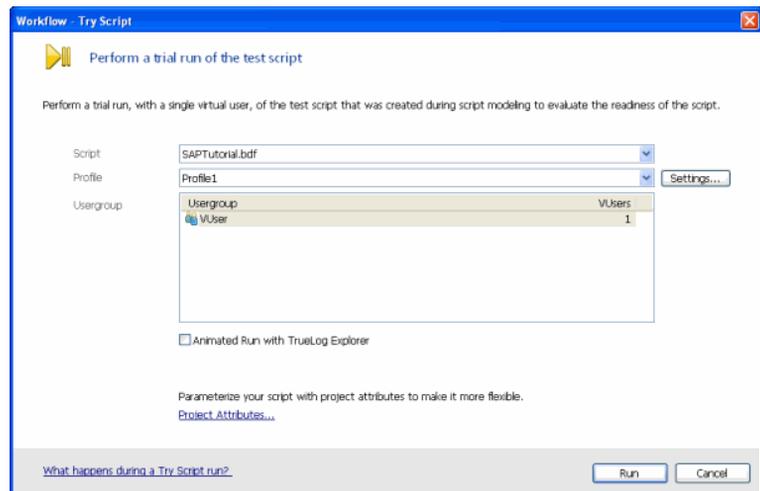
4   The TryScript run begins. The Silk Performer Monitor window opens, giving you detailed information about the run's progress.

Note If you checked the *Animated* checkbox on the *TryScript* dialog box, 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.

How SilkPerformer Handles SAPGUI Replay

For SAPGUI script replay, SilkPerformer uses an architectural de-coupling of the SilkPerformer virtual user and the SAPGUI ActiveX control. This means that when there is a crash or failure of the SAPGUI process, SilkPerformer virtual user measurements are retained. This is achieved by running a separate SAPGUI replay process, called *PerfSapGuiReplay.exe,* for each virtual user.

Upon a SAPGUI API timeout or crash, the affected virtual user process automatically restarts in the background. SilkPerformer thereby recognizes and reports potential instabilities of the SAPGUI client. This minimizes test failures when problems occur within SAPGUI itself and allows for more reliable testing.

Note The *SapGuiRestart( )* function enables you to force a restart of the SAPGUI engine when an unstable or inconsistent state is detected.

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. This launches TrueLog Explorer loaded with the TrueLog from the recent TryScript run.
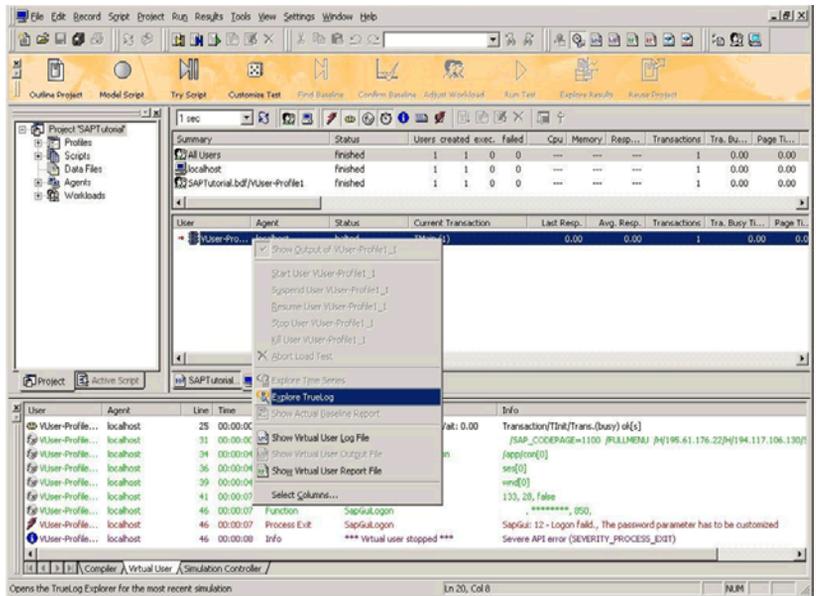
2

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, for example user IDs, password, and others.

> **Note** TrueLog Explorer is a powerful test script customization tool that offers much more functionality than is demonstrated in this tutorial. Refer to 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 Silk Performer 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. Refer to 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. Refer to 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, for example the display of a new dialog box 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.
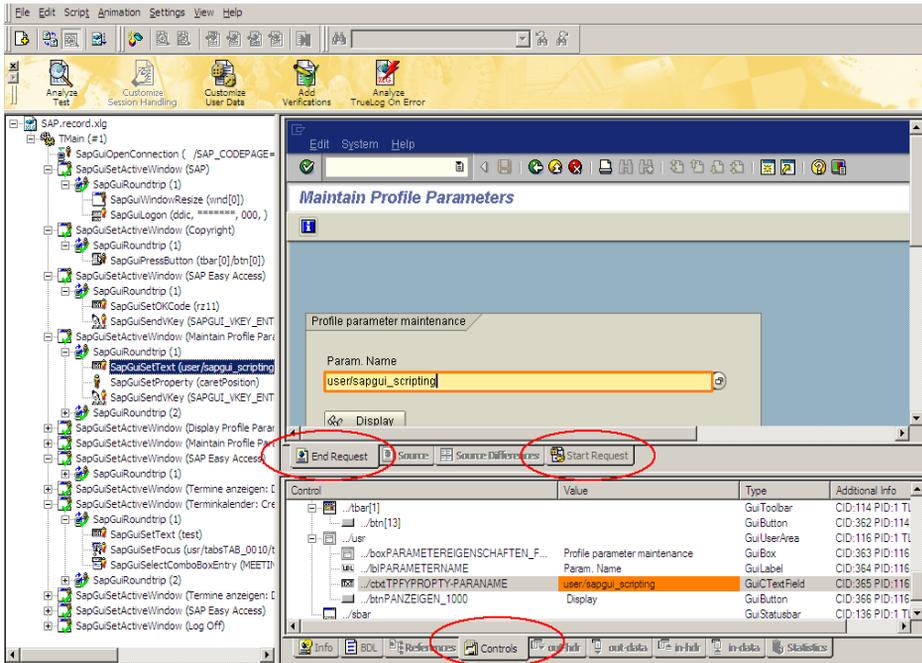
SAPGUI TrueLog functions

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

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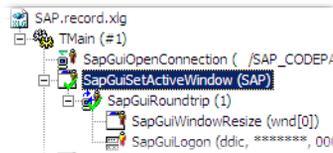
Customizing Input Parameters

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*.

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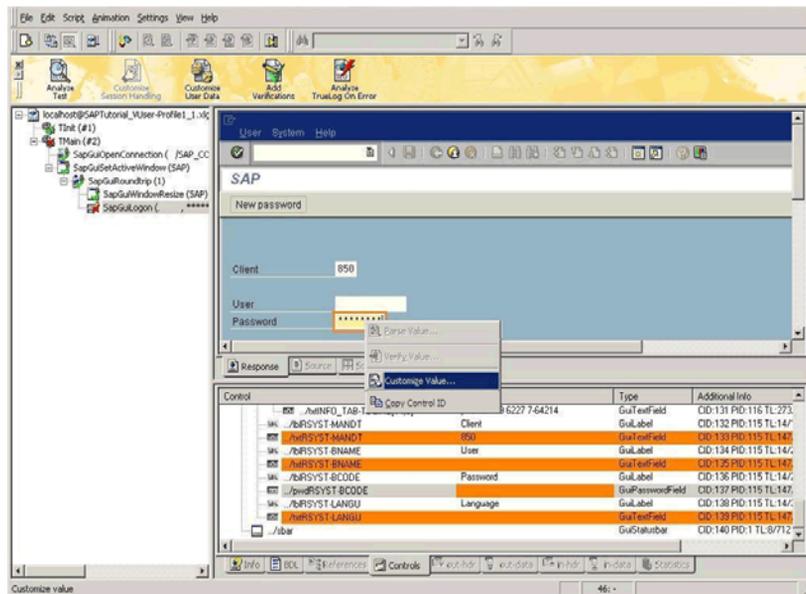
4   The *Parameter Wizard* opens. The Parameter Wizard enables you to
create a new parameter for the recorded password. To keep this example
simple, a constant parameter type will be used. Click the *Create new
parameter* option button and click *Next.*

Note Refer to the *Silk Performer User Guide* for full details
regarding the Parameter Wizard.
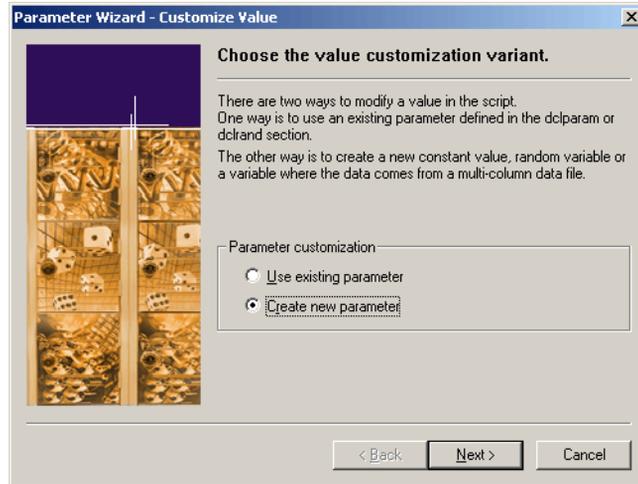
5   Click the *Constant value* option button and click *Next.*
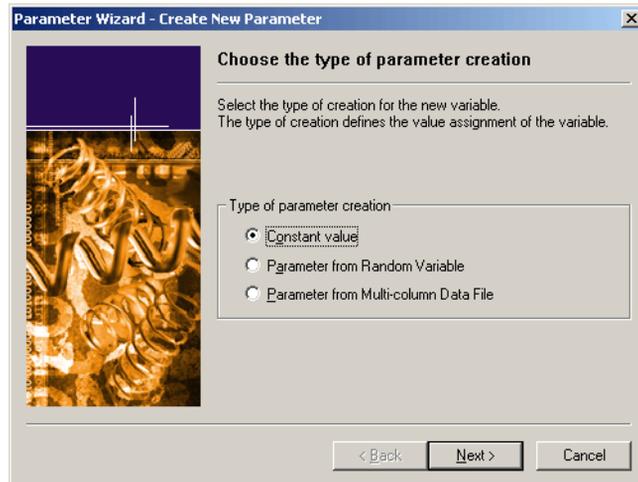
6 The data type to be used is *string.* Click *Next.*
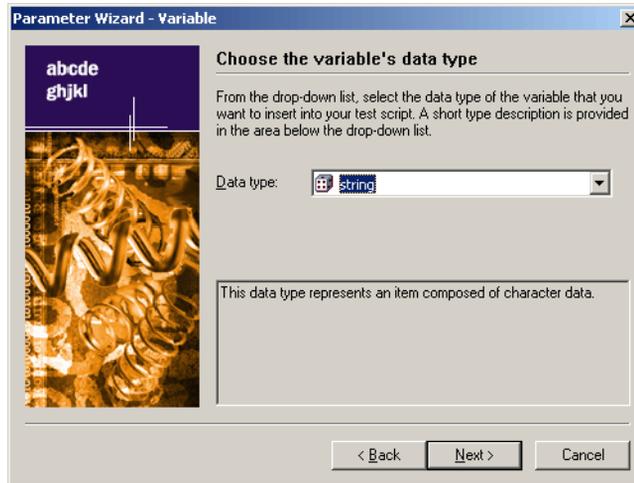
7 Define a meaningful *Name* for the new parameter and enter your user password as the string *Value*.
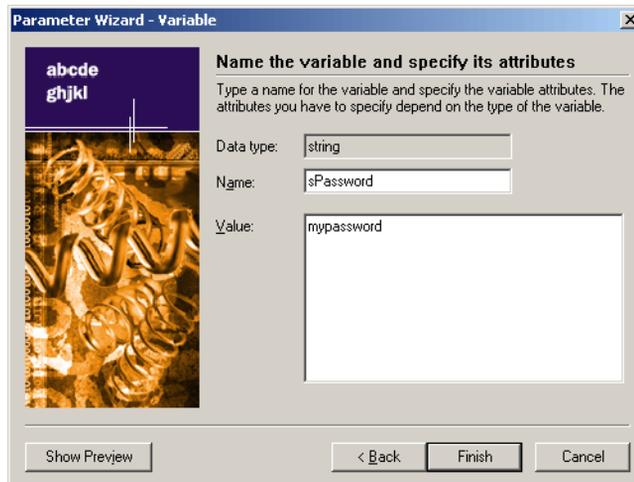
8 Now execute a new TryScript run. Your password parameter will automatically be inserted into the replayed test script and the script should run without error.

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 box.

2 Click the *Customizable calls* option 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*.

> **Note** For this example, select the *Priority* field as shown in the figure below.
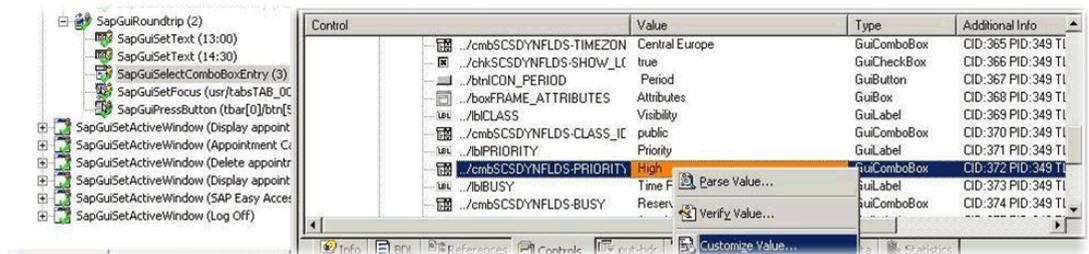
Note Controls that can be customized are outlined in orange.
Controls that have already been customized are outlined in green.
Controls that are outlined in blue can have their values parsed or
verified, but they cannot be customized.
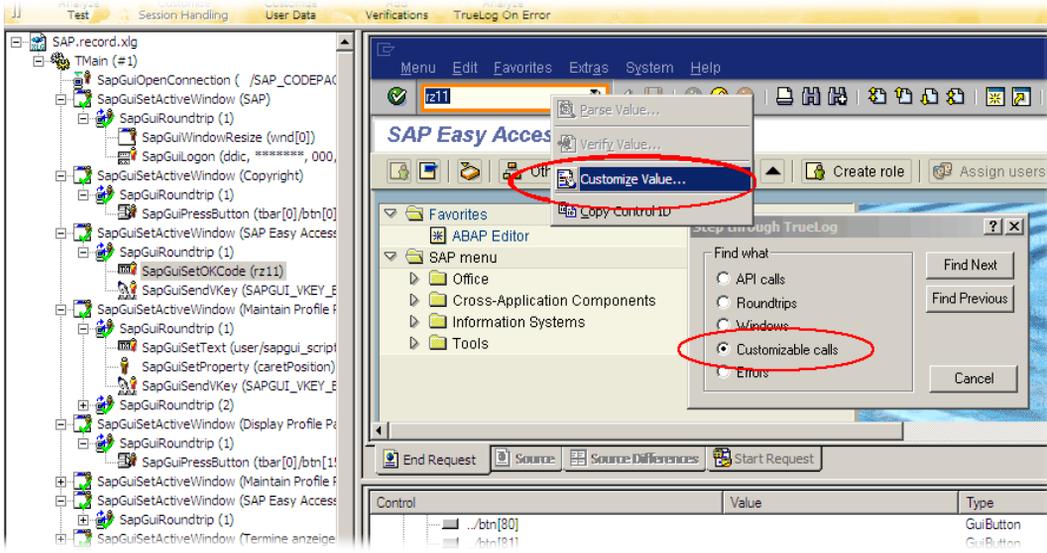
4 With the Parameter Wizard you can modify script values in one of two
ways. You can either use an existing parameter that's defined in the
dclparam or *dclrand* section of your script, or you can create a new
parameter, based on either a new constant value, a random variable, or
values in a multi-column 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. Refer to the *Silk Performer User
Guide* for complete details regarding the functionality of the
Parameter Wizard.

5 Click the *Create new parameter* option button and click *Next* to create a new parameter.
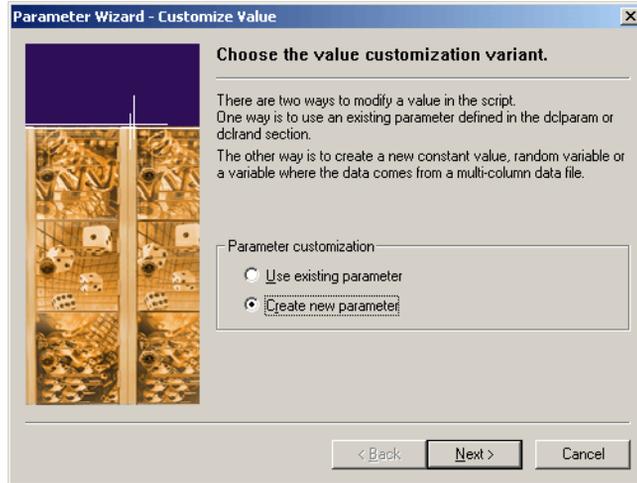
6 The *Create New Parameter* dialog box opens. Select the *Parameter from Random Variable* option button and click *Next*.

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

8   Click *Next*.
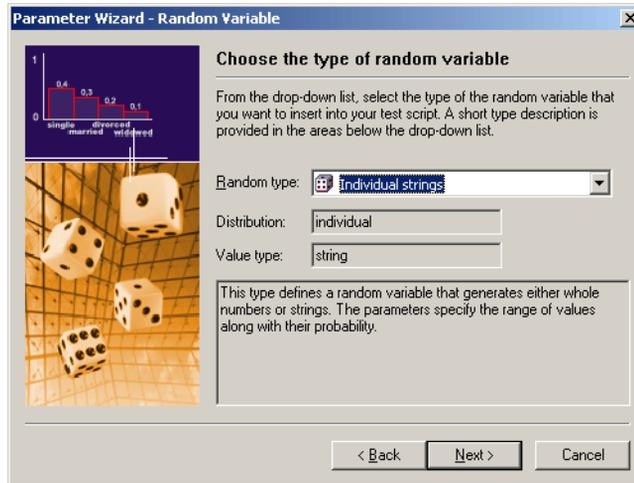
9   The *Name the variable and specify its attributes* screen opens. With SAPGUI applications, all available list box values are pre-loaded with weight values of *1*. Enter a name for the variable in the *Name* text box and click *Next*.
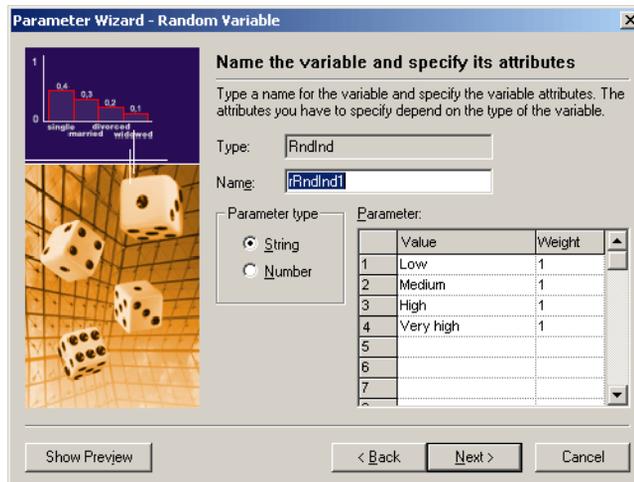
10 *Per usage* random value generation is selected by default. Click *Finish.*

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 displays 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 display with green
highlighting.
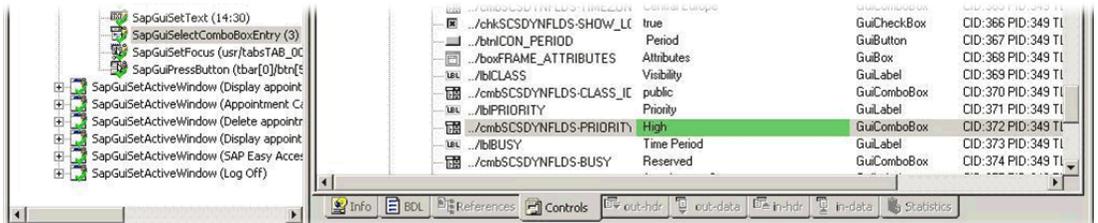
**Additional
customizations**

You may find that additional customizations are useful, for example
randomizing username and appointment-time input parameters for load testing
purposes. Customization is possible for nodes that involve changes of text,
combo boxes, check boxes, and option-button controls. Refer to 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 Silk Performer's Monitor window during load tests. This counter can also be monitored in Performance Explorer as a Silk Performer 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. Refer to the *Silk Performer 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

Best Practices for Testing SAPGUI

Introduction

This chapter explains best practices for load testing SAPGUI with Silk Performer.

This chapter contains the following sections:

Prerequisites for SAPGUI Load Tests

The following issues need to be considered before you begin load testing SAPGUI:

Issues with agents

To run huge load tests you need to consider your Agent setup. Please see the "Agents" section for details.

SAPGUI scripting	Ensure that SAPGUI scripting is installed on all agents and the controller. Also ensure that SAPGUI scripting is enabled on the servers.
Test data	For data-driven testing, it's important to use accurate data as input. See "Customizing Input Data" to learn what should be considered.
Test cases	Test cases that are to be tested must be well documented.
	Determine if varying input data in input controls will result in the display of different screens or change the availability of any onscreen controls.
Determining your goals	The most critical measure of an SAP system is the number of dialog steps that can be executed during a specified timeframe, for example 100,000 dialog steps in an hour).
	From the SAPGUI perspective, a dialog step is a transition from one screen to the next. Certain dialog steps are simple. Others are complex and cause additional server load. Therefore it is necessary to have a good mix of test scripts that cover most of the common transactions that are used within companies.
	The dialog steps can be monitored with different ST transactions within SAP. Our SAP monitor, which uses ST03N, can also be used.

SAPGUI Client Versions

It is recommended that you always use the latest version of the SAPGUI client, with the latest patch.

The latest patches can be downloaded from http://service.sap.com/swdc (username and password required). Navigate to *SAP Support Packages / Entry by Application Group / Frontend Components.*

Testing Logon Sequences

The logon sequence is a resource-intensive task that should not be tested. The logon sequence should be extracted and executed within the TInit transaction. The TMain transaction should only contain those transaction steps that are to be tested, ending again at the screen that follows logon. The logout sequence should be moved to the TEnd transaction. To return to the initial screen, use the "/n" transaction code.

Here is a sample customized script for reference:

```
dcluser
  user
    VUser
  transactions
```

```
    TInit           : begin;
    TMain           : 1;
    TEnd            : end;

var
  gsConnID : string;

dclrand

dcltrans
  transaction TInit
  begin
    // Connecting to SAP
    gsConnID := SapGuiOpenConnection(
      "   /SAP_CODEPAGE=1100  /FULLMENU  /H/111.111.111.111/S/3299/H/
222.222.222.222/S/3297/H/cpce801 75 /3",
      "SapGuiOpenConnection");

    SapGuiSetActiveConnection(gsConnID);
    SapGuiSetActiveSession("ses[0]");
    // SAP
    SapGuiSetActiveWindow("wnd[0]", "SAP", SAPGUI_MATCH_ExactNoCase);
    SapGuiWindowResize(175, 28, false, "SapGuiWindowResize");

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

    // SAP Easy Access
    SapGuiIgnoreError(SAPENGINE_STATUSBAR_CHANGED, SEVERITY_SUCCESS);
  end TInit;

  transaction TMain
  var
  begin
    // start with the SapGuiSetActiveWindow
    SapGuiSetActiveWindow("wnd[0]", "SAP Easy Access", SAPGUI_MATCH_
ExactNoCase);

    // now lets do the transaction specific tasks
    // .....


    // end the end you need to make sure that the last calls brings you
back to the SAP Easy Access window
    // so that the next TMain iteration can successfully call the
SapGuiSetActiveWindow

    SapGuiSetOKCode("tbar[0]/okcd", "/n"); // this can be used to switch
back to the SAP Easy Access window
  end TMain;

  transaction TEnd
  begin
    SapGuiPressButton("tbar[0]/btn[15]", "SapGuiPressButton\\btn[15]");
    // Log Off
    SapGuiSetActiveWindow("wnd[1]", "Log Off", SAPGUI_MATCH_
ExactNoCase);
    // Yes
   SapGuiPressButton("usr/btnSPOP-OPTION1", "SapGuiPressButton\\Yes");
  end TEnd;
```

Customizing Input Data

Data-driven testing is required for this type of load test. This means that you need CSV files that contain input data for virtual users. For example, CSV files may contain material numbers or document IDs that are used by virtual users in transactions.

Inputting different types of materials may result in the display of different screens. So, when virtual users pick certain materials, verifications may fail because resulting screen contain different controls with different information.

To solve this problem, do one of the following:

*   Use input data that contains materials, or documents, that are similar and will return the same screens.

*   Adjust your script to check for the type of screen that is displayed, that is, the controls that are available. You can use SapGuiVerifyObjectAvailability here to verify if a control is available. Multiple recordings should be performed in which different types of materials are accessed. Then you will see what controls to expect in different situations and, based on whether or not the controls appear, configure the verifications.

Prerequisites for Using Test Manager

Customizing scripts If you are going to upload your SAP scripts to SilkCentral Test Manager, you should perform some customizations in your script to make your uploaded project easily reusable for different projects and test definitions within Test Manager.

The following values in scripts should be customized to use project attributes:

*   Connection strings

*   Usernames

*   Passwords

*   Client numbers

*   Languages

If you create these five project attributes and customize your script so that the SapGuiOpenConnection and SapGuiLogon methods use the values of the attributes, your script can easily be customized with different values for the attributes from Test Manager.

Here is a sample script that shows you how your SAP logon sequence should look:

```
var
  gsConnID : string;
  sServer:    string;
  sUsername : string;
  sPassword : string;
  sClientNum : string;
  sLanguage : string;


dclrand

dcltrans
  transaction TInit
  begin
    AttributeGetString("SAPServer", sServer);
    AttributeGetString("SAPUser", sUsername);
    AttributeGetString("SAPPass", sPassword);
    AttributeGetString("ClientNum", sClientNum);
    AttributeGetString("Language", sLanguage);

    // Connecting to SAP
    gsConnID := SapGuiOpenConnection(sServer, "SapGuiOpenConnection");

    SapGuiSetActiveConnection(gsConnID);

    SapGuiSetActiveSession("ses[0]");

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

    SapGuiWindowAction(SAPGUI_WND_MAXIMIZE, "SapGuiWindowAction\\SAPGUI_
WND_MAXIMIZE");

    // Logon to SAP System
    // Before running a test you have to customize the password
parameter!
    ThinkTime(5.9);
    SapGuiLogon(sUsername, sPassword, sClientNum, sLanguage,
"SapGuiLogon");

    // SAP Easy Access
    SapGuiIgnoreError(SAPENGINE_STATUSBAR_CHANGED, SEVERITY_SUCCESS);
  end TInit;
```

Now you need to define these project attributes via Silk Performer's Project Attributes Configuration dialog.

Defining verification loads

If you are going to upload a functional SAP test to Test Manager you should define a verification workload in your Silk Performer project. Remember that when you upload a Silk Performer project to Test Manager, Test Manager actually executes the workload of the uploaded project. The verification workload is designed for one-user script executions. Therefore you need to change your workload to be a verification workload and specify which script should be executed by it.

Procedure To Specify "verification" workload and select a script for execution:

1 Click "Adjust Workload" on the workflow toolbar.

2 Select "Verification" and click "Next". The Verification Workload Configuration dialog box opens.

3 Select the profile and script to be executed by the workload and specify how you want TrueLog to be generated.

4 Click "OK".

5 Select the "Save Project" command from the File menu to save your workload configurations.

Before uploading the project to Test Manager, you should run the verification test at least once in Silk Performer.

Using Silk Performer projects in Test Manager

After you have uploaded a Silk Performer project to Test Manager, the project will be executed with the default values of the project attributes that you have defined. From Test Manager you can now change those project attributes for the test definition that you created while uploading the project, or you can create new test definitions that reference the uploaded project and then specify different attribute values for those test definitions.

When creating a test definition in Test Manager, browse for the uploaded project in your current source control profile. On the "Parameters" tab, specify different values for the Silk Performer project attributes (connection string, username, password, etc).

This feature enables you to create SAP scripts for common tasks that can be reused to test servers in different environments.

Accessing Low-Level Properties

The functions SapGuiInvokeMethod, SapGuiSetProperty, and SapGuiGetProperty can be used to access the low level properties of each control on the current screen. This makes it possible to, for example, verify whether or not a text control is read-only, or determine the background color of a label.

The SAPGUI scripting API that is used to perform SAPGUI testing is a large COM library that allows Silk Performer to access controls and perform actions. The same COM library can be used with the above mentioned API calls. To access the list of methods and properties that individual controls offer, you must inspect the type library of the SAPGUI scripting API.

You need a tool that allows inspection of type libraries, such as the Ole32View tool that comes with Visual Studio. You need to open the sapfewse.ocx file,

which can be found in the SAPGUI installation directory under
\FrontEnd\SapGui.

To, for example, get the name of a control where you know the control ID, you
would use the following call:

SapGuiGetProperty("/usr/lbl[1,2]", "Name", sOutValue);

Print("The control has the following name:" + sOutValue);

Most properties return a simple type, such as a string, number, or Boolean.
Some properties return another object. An example is the "Parent" property that
returns the parent control of the current control. Whenever a property returns
another control, this control is temporarily held in cache and can be accessed
with the constant SAPGUI_ACTIVE_OBJECT. Here is an example call for
retrieving the name of the parent property:

SapGuiGetProperty("/usr/lbl[1,2]", "Parent");

SapGuiGetProperty(SAPGUI_ACTIVE_OBJECT, "Name", sOutValue);

Print("The parent control has the following name:" + sOutValue);

Properties that are of the type Boolean are also returned in string representation
because SapGuiGetProperty only returns string values. As SAPGUI scripting is
a COM library, you get the string representation of the two possible values,
VARIANT_TRUE and VARIANT_FALSE. VARIANT_TRUE is "-1" and
VARIANT_FALSE is "0".

Here is an example that verifies if a Boolean property is true or false:

```
SapGuiGetProperty("/usr/txtName", "Changeable", sOutValue);
If(sOutValue = "-1") then
  Print("The text control is changeable!!");
End;
```

Properties overview   Everything in SAP is a component and therefore has the following properties
(ComClass GuiComponent):

- *Name* - Name of the control.

- *Type* - Type of the control as text (for example, GuiButton or
  GuiTextField).

- *TypeAsNumber* - All types have internal numbers (for example,
  30=GuiLabel, 31=GuiTextField).

- *ContainerType* - Boolean property that defines if the control is a
  container. Containers contain other controls as children (for example, a
  toolbar is a container that contains toolbar buttons).

- *ID* - This is the unique ID of the control.

- *Parent* - If this control is contained within a container, this will return the
  parent control.

Visual components such as controls have additional properties (ComClass GuiVComponent):

- *Text* - The main text of a control. For example, the text in a text control or the text on a button.

- *Left, Top, Width, Height, ScreenLeft, and ScreenTop* - Number values that return information about screen coordinates and coordinates within parent containers.

- *Changeable and Modified* - Boolean parameters that indicate the current state of a control, for example whether or not the control is changeable or read-only or whether or not the control as been modified.

Each control type can have additional properties that can be seen in the COM type library using Ole32View.

Handling Unexpected Windows

If you know that at a point in a script a dialog box may be generated, a dialog box will not display in each iteration, you must handle the additional dialog box. You can either use an error handler or you can use the API to query if a certain dialog box has been activated.

For example, if you enter values in a text field and press the Execute button, depending on the entered value, a dialog box may open on which you have to press an additional button to continue. Then the transaction continues with a subsequent screen.

You can use SapGuiVerifyWindowAvailability if a window with a certain ID or title is currently available. Make sure that you define the severity parameter as "informational". Otherwise, the method will throw an error in cases where the window does not display. Here is an example:

```
// here is our first screen - we enter some value and hit enter
SapGuiSetActiveWindow("wnd[0]", "First screen");
SapGuiSetText("usr/txt1", "some text");
SapGuiSendVKey(SAPGUI_VKEY_ENTER);

// now we check for a specific popup
if(SapGuiVerifyWindowAvailability("wnd[1]", "Some Popup Window", false,
SEVERITY_INFORMATIONAL)) then
  SapGuiSetActiveWindow("wnd[1]", "Some Popup Window");

  SapGuiPressButton("usr/btnPOP-OPTION1");
end;

// we go on with our next screen that we expect in both cases
SapGuiSetActiveWindow("wnd[0]", "Next screen");
```

If there is a situation where different dialog boxes can open and you have to handle each dialog box individually, you can just verify for the window ID and

then check the window title. Here is an example:

```
 // now we check for a new window
 if(SapGuiVerifyWindowAvailability("wnd[1]", null, false, SEVERITY_
INFORMATIONAL)) then
   SapGuiSetActiveWindow("wnd[1]");

   // now - check what the window title is and depending on that do some
action
   sWindowTitle := SapGuiGetActiveWindowTitle();
   if(sWindowTitle = "Some Alert") then
     SapGuiPressButton("usr/btnPOP-OPTION1");
   end;
   if(sWindowTitle = "Some other alert") then
     SapGuiPressButton("usr/btnPOP-OPTION2");
   end;
 end;
```

Handling windows that have dynamic titles

It's common to see window titles that contain dynamic values (for example, "Change Material 1110 – (Finished Product)") when you execute MM02 transactions to change products. In this example, the material number is part of the window title.

By default, the recorder scripts the following method call when this window opens:

```
SapGuiSetActiveWindow("wnd[0]", "Change Material 1110 (Finished
Product)", SAPGUI_MATCH_ExactNoCase);
```

If you customize your script so that you choose a random material number, then your script will throw an error as the window title verification will fail. So if your virtual user, for example, picks material number "1111", the resulting title will be "Change Material 1111 (Finished Product)". This will cause an error as the verification is performed on the recorded title.

SapGuiSetActiveWindow not only allows you to verify against a constant value, it also allows you to verify against wildcard expressions and regular expressions. To solve the above described example problem, you could use the following change to SapGuiSetActiveWindow:

```
SapGuiSetActiveWindow("wnd[0]", "Change Material *", SAPGUI_MATCH_
WildCard);
```

Or, if you don't want to perform a title verification, you can leave the last two parameters. They are optional. No verification will then be performed.

Known SAP Issues

The following section describes workarounds for issues that may be related to SAP internal issues.

Tables with single rows

When there is only one row in a table, the row count is returned as '2' ('rowcount=2'). It seems that there is a second empty row added to these tables. Tables with more than one row entry return the correct number of rows.

Selecting the second entry in this example would result in an error as the entry is not valid. Therefore, before selecting an entry, you should confirm that the entry is not empty.

SAP API calls only work in first TMain iteration

The following two method calls have been known to cause problems:

*   SapGuiGridGetRowCount
*   SapGuiGridSelectCell

These methods have been known to fail in the second iteration of TMain, and also in loops. The internal COM interface may not be up-to-date. To update the internal COM reference to the control, call the SapGuiGetProperty method. This method updates the internal COM pointer. You need to call SapGuiGetProperty on the control just before you see a call that fails in a second iteration. It is recommended that you use the Name property as every control has this property.

Here is an example:

```
SapGuiGetProperty("/usr/somecontrol", "name");
SapGuiGridGetRowCount("/usr/somecontrol", nCount);
```

Settings for Large Load Tests

When running large load tests, you should evaluate the following issues:

Agents

Installed Versions

Ensure that all your agents have the same version as the installed SAPGUI client. The SAPGUI scripting API needs to be installed on all agents. The API needs to be enabled and warning message boxes need to be disabled.

Number of Users on an Agent

To run large load tests you need more agents, rather than stronger agents. 30 virtual users can be simulated on a 1GHZ/1GB machine.

A 1GHz machine would normally be able to handle 100 users. This is not the case however. GDI resources begin to run out at that point. GDI is the Graphical Device Interface in Windows. When virtual users are simulated, each virtual user actually uses the SAPGUI client and therefore requires GDI resources for each control on the individual screens. So, theoretically it is possible to run more than 30 users, but tests and feedback have shown that Windows resource limitations appear beyond that point. Depending on the transactions that are executed (for example, number of controls on a screen) it may be possible to simulate more than 30 users.

Replay profile settings   Make sure that in replay profile settings the *Log control information in TrueLog* option is not checked. When this option is turned on during load tests, each virtual user builds TrueLog with information for all controls on every window in transactions. Depending on your transactions and the number of controls that are on-screen, you may experience heavy performance impact with this setting enabled. Therefore, this option should be turned off during load tests. Only the Log control information on error option should be selected. This option logs control information for the current window when errors occur. Ensure that you turn this option on when running Try-Scripts as you want to have this control information in Try-Script Trueogs.

Testing login sequences   Logon sequences are resource intensive and should not be tested in large load tests.

Server-side changes   When running a large load test you may have to change certain server-side parameters that allow additional users from other machines connecting to the servers. Please consult SAP documentation related to the rdisp/rfc_max_own_used_wp parameter.

Visit the following link for detailed instructions:

http://help.sap.com/saphelp_erp2004/helpdata/en/c3/ce70a5cf0ab1499bfbf464e6e92de0/frameset.htm

SAP gateways   If your SAP environment uses a SAP gateway machine to connect to your SAP servers, you may run into the problem that your gateway no longer accepts new connections. This is caused by SAP gateways do not always recognizing when existing connections are shut down. Sudden aborting of load tests and ungraceful system shut-downs do not always result in gateway connections to servers being closed. If too many connections are left open, you may receive a "cannot open more client connections" error. To resolve this problem, reboot your gateway service.

4

SAP eCATT Integration With Silk Performer

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

This chapter contains the following sections:

Section	Page
Overview	47
Setting Up Integration	48
Interacting with eCATT from Silk Performer	52
Interacting with Silk Performer from eCATT	56
Limitations	61

Overview

SAP *eCATT (Extended Computer Aided Test Tool)* has been integrated with Silk Performer. 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., Silk Performer) 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 Silk Performer's eCATT integration.

Procedure To configure Silk Performer's eCATT integration:

1   On your SAP server, register Silk Performer 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 Silk Performer in combination with eCATT, install both Silk Performer 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 Silk Performer system settings, configure SAP eCATT server connection data.

6   Within Silk Performer system settings, define a SAP eCATT directory for extended Silk Performer test results.

Registering Silk Performer in eCATT

Silk Performer 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	Silk Performer
PROG_ID	SAPeCATTPlugIn.BorlandSPeCATT
TOOL_DESC	Silk Performer 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 Silk Performer with the eCATT integration, you must first install your SAPGUI client and afterward apply the Silk Performer installation. Whenever eCATT initiates the integration between Silk Performer and eCATT, Silk Performer 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 Silk Performer and eCATT. eCATT forwards the connection that is to be used to Silk Performer, 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).*

> **Note** On 64-bit machines the key is *HKLM\Software\Wow6432Node\Silk*

Configuring SAP eCATT connection

Connection details for Silk Performer's communication with SAP eCATT must be specified in Silk Performer 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 Silk Performer'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.
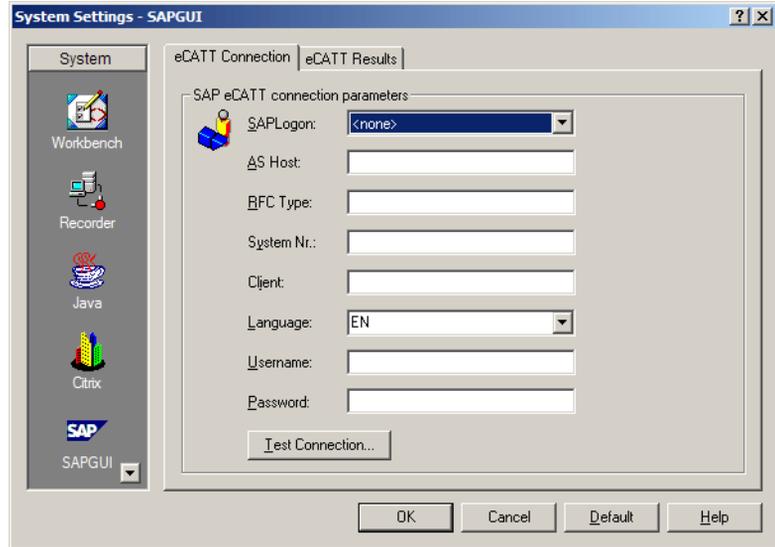
Configuring eCATT extended results

To enable the viewing of Silk Performer result files from within SAPGUI, you can specify a UNC path to a public file share in which extended Silk Performer test results can be stored and accessed by users (e.g., \\fileserver\ecattresults). Silk Performer will use the specified directory to store the results of Silk Performer 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 Silk Performer'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.
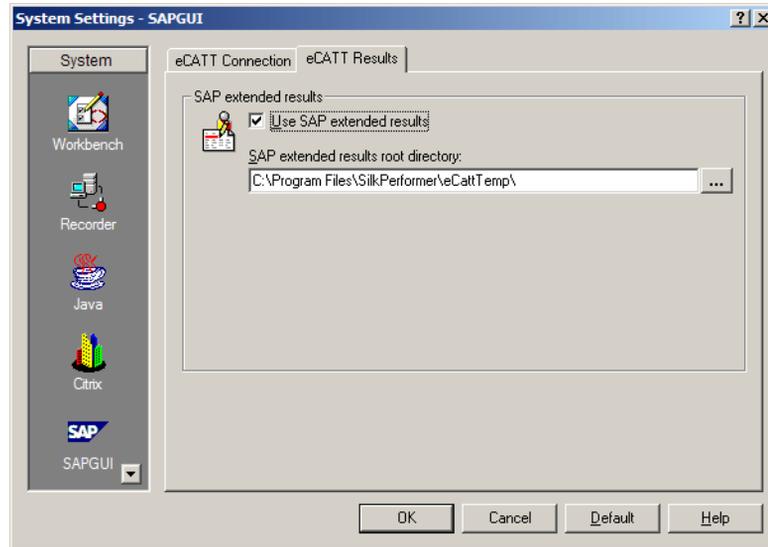
Interacting with eCATT from Silk Performer

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

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

Silk Performer 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:
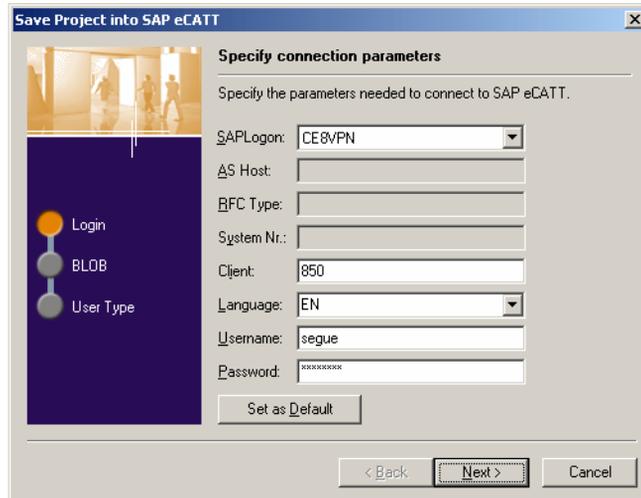
The second wizard step prompts you for the *BlobId* and *Version:*
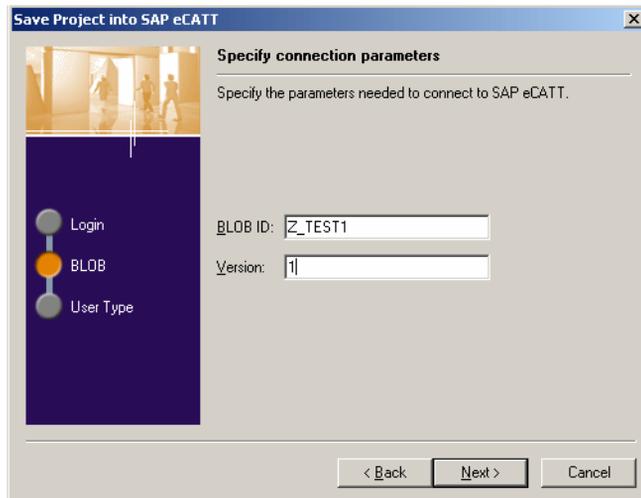
In the third step you have to define the user type that is to be the primary user type in the script. As you may have multiple user types configured in your

project, you must define the user type that is to be executed by default when eCATT executes scripts:
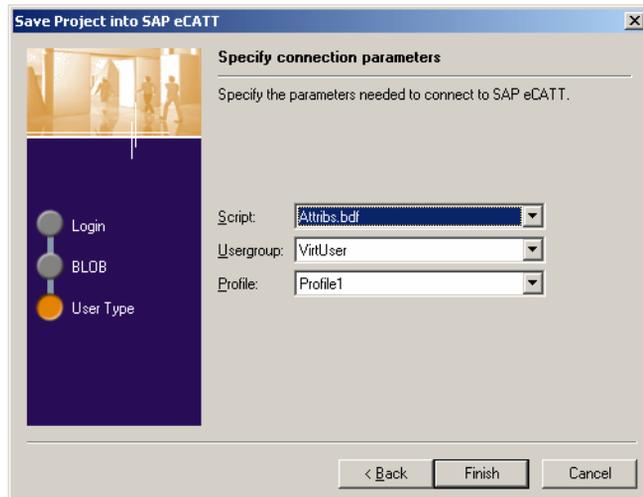
Opening a project from eCATT

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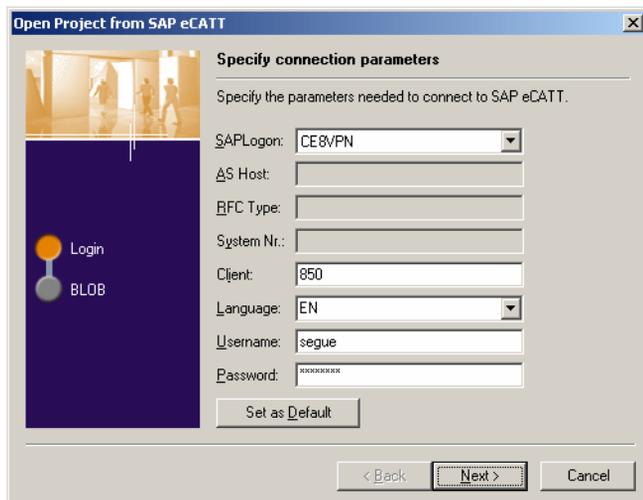
The second step prompts you for the *BlobId* and *Version* of the project that is to be downloaded.

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, Silk Performer 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:
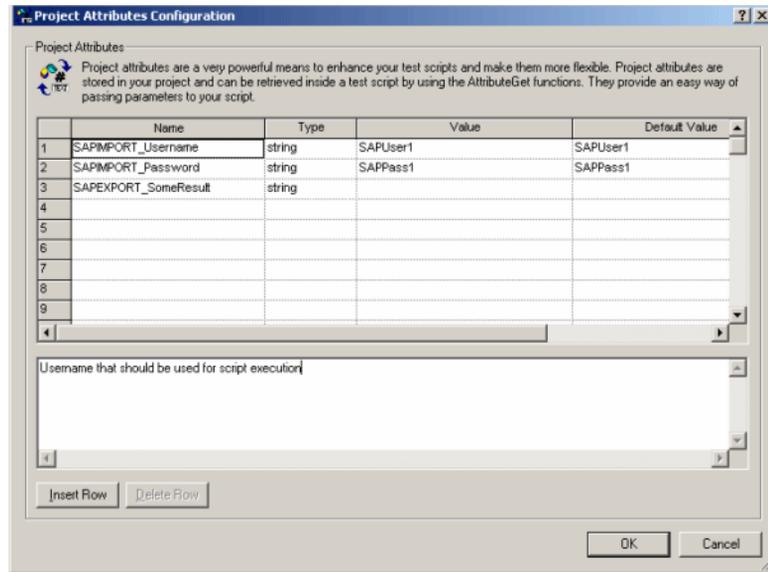
In a script you access these input values as follows:

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

The output value is set as follows:

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

Interacting with Silk Performer from eCATT

From eCATT it's possible to utilize Silk Performer as an external test tool by:

• Creating a new Silk Performer script

• Editing/Viewing an existing Silk Performer script

• Executing a script in one of three modes:

   - "Normal" without Silk Performer Workbench

   - "Debug Mode" with Silk Performer Workbench and the option of adjusting settings before executing the script

   - "With Surface of External Tool" with Silk Performer Workbench

Creating a new Silk Performer script

Within SAP eCATT (SECATT transaction) you can create a new Silk Performer script by specifying a test script name (blobID), a version, and *Silk Performer for eCATT* as the external tool, as shown below:
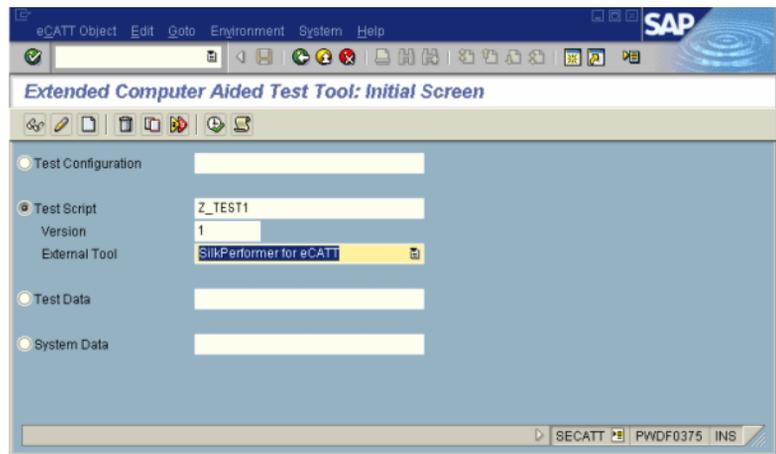
Now click the *Create Object* button (or press F5). This creates an empty Silk Performer script and stores it in the eCATT repository. On the following screen you must enter all required fields before you can edit the script with Silk Performer. Once you have completed all required fields, click the *Script* toolbar button. Silk Performer then opens and downloads the newly created project in Edit mode (for other available options, please see the next chapter).

Editing/viewing an existing Silk Performer script

Within SAP eCATT (SECATT transaction) you can both edit and view an existing Silk Performer script by specifying the test script name (blobID) and the version, as shown below:
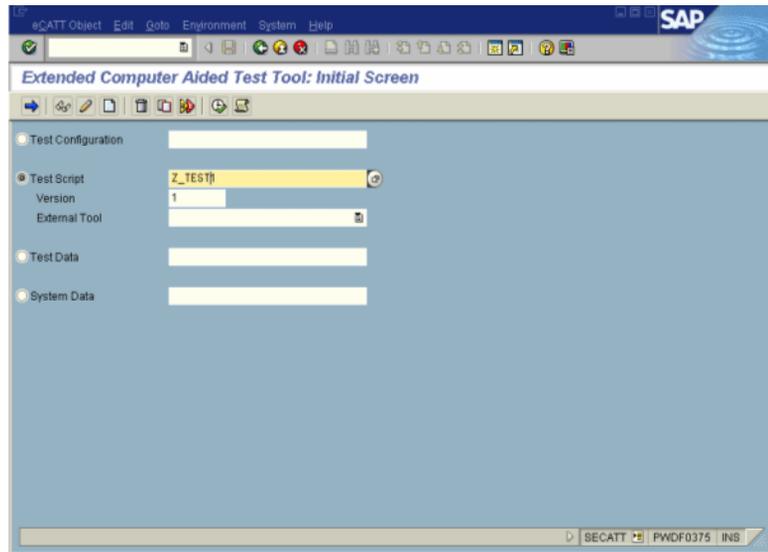
Now you can either click the *Display Object* (F7) button or the *Change Object* (F6) button to view or edit the eCATT script. On the following screen you can click the *Script* button to either view or edit the script in Silk Performer.

If you are only viewing the script, Silk Performer 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, Silk Performer 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 Silk Performer 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 Silk Performer script

Within SAP eCATT (SECATT transaction) you can execute a Silk Performer script by specifying a test script name (blobID) and version, as shown below:
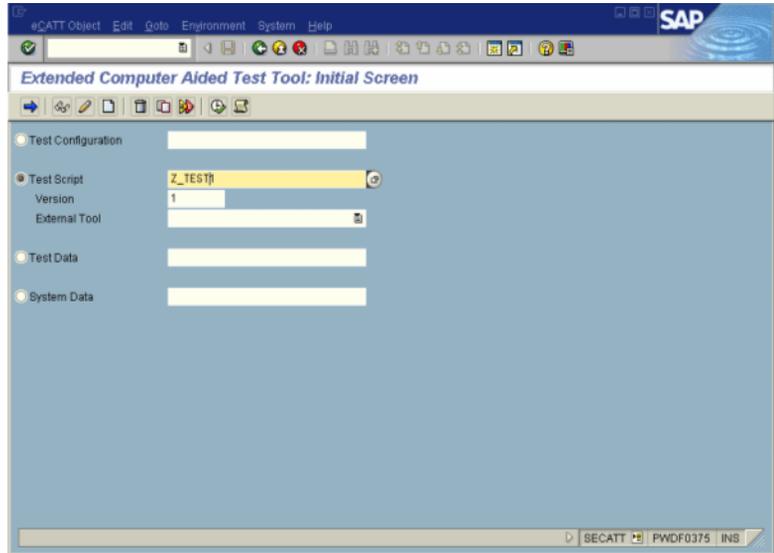
Once you have entered these values, click the *Execute* button (F8) to go to the execution dialog where you can specify start options for the script:
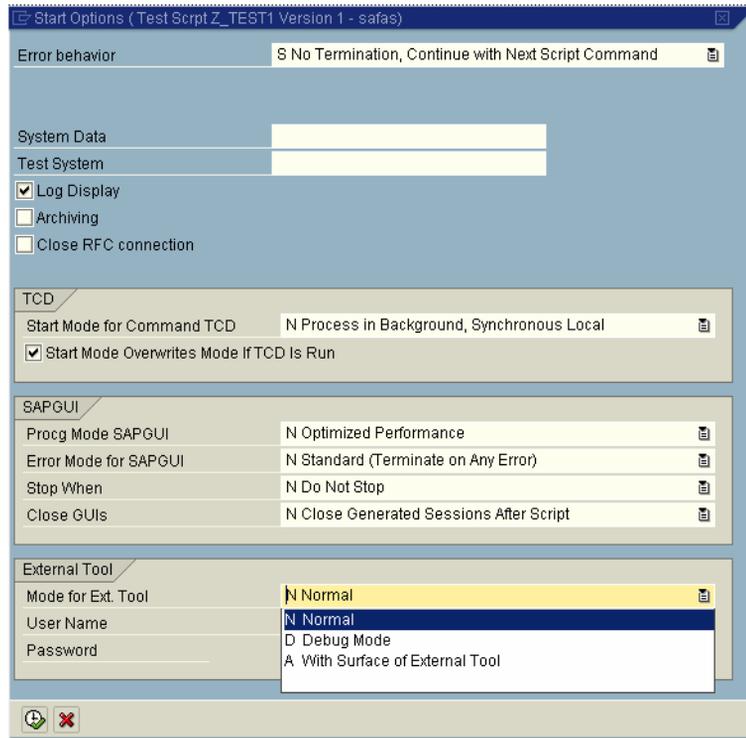

Depending on the *Mode for Ext. Tool* selection, the test will either be executed without Silk Performer Workbench *(Normal),* with Silk Performer Workbench *(With Surface of External Tool),* or in the attended debug mode with Silk Performer Workbench *(Debug Mode).*

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

In Debug mode, Silk Performer 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, Silk Performer 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):
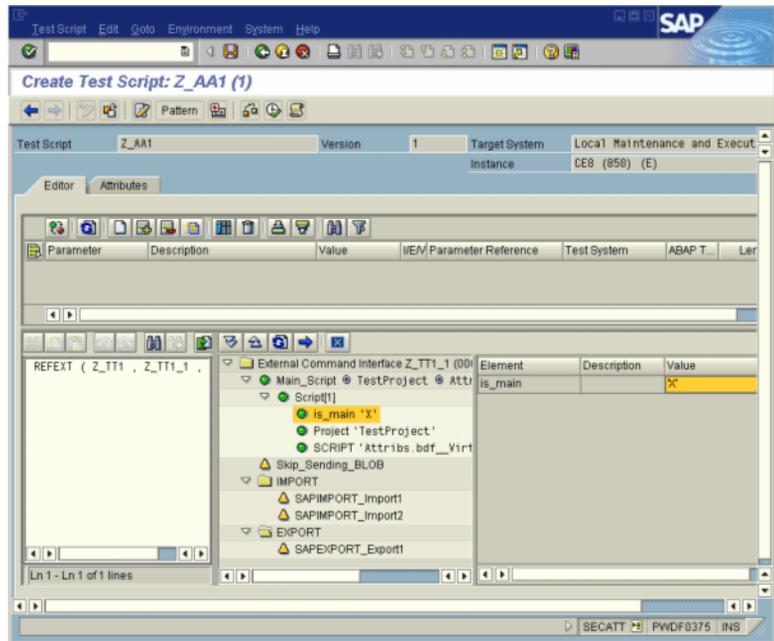
You will see the various scripts that are there-one should be marked is_main. You will also see all import and export arguments.

Limitations

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

Script name length

When uploading a Silk Performer 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 Silk Performer 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, Silk Performer 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 Silk Performer 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 Silk Performer 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

U

V