Citrix Tutorial

SilkPerformer®
2007

Borland®

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

August 2007
PDF

Contents

Introduction 1
Overview 1
SilkPerformer 2
Support for Citrix MetaFrame Terminal Services . 3

Chapter 1 Defining Load Test Projects 7
Overview 7
Prerequisites 8
Defining a Load Test Project 9

Chapter 2 Creating Test Scripts 11
Overview 11
Creating a Load Test Script 12
Screen Synchronization & Verification 21
Verification & Parsing via OCR 24
Trying Out a Generated Script 31

Chapter 3 Customizing User Data 45
Overview 45
Customizing User Data 45
Synchronizing Text 52

Chapter 4 Testing NFuse Sessions 53
Overview 53
Creating a Load Test Script 53

Chapter 5 Citrix Project & System Settings 61
Overview 61
General Citrix Settings 62
Citrix Simulation Settings 63
Citrix Client Settings. 65
Citrix System Settings for OCR 66

Chapter 6 Best Practices 69
Overview 69
Test Preparation 70
Recording Use Cases 74
Troubleshooting Scripts 78
Issues Specific to Citrix MetaFrame 80

Index 85

Introduction

About these tutorials

The *Citrix Tutorial* offers an overview of using Borland SilkPerformer to set up and run load tests of applications that are hosted via Citrix MetaFrame terminal services.

This Introduction contains the following sections:

Section	Page
Overview	1
SilkPerformer	2
Support for Citrix MetaFrame Terminal Services	3

Overview

The *Citrix Tutorial* is designed to ease you into the process of using SilkPerformer to load test applications that are hosted via Citrix MetaFrame terminal services and to get you up and running as quickly as possible. It will help you to take full advantage of SilkPerformer's ease of use and exploit the leading-edge functionality that's embodied in e-business' load-testing tool of choice.

What is Citrix

Citrix facilitates real-time access to shared applications over networks and the Internet. Remote access to Citrix-enabled applications can be over DSL, T1, ISDN, or dial-up. Citrix MetaFrame enables multiple users to run shared applications simultaneously. Communication between Citrix clients and servers consists of exchange of user inputs (keyboard/mouse) and screen shots.

Citrix MetaFrame runs on Windows NT 4.0 (Terminal Server Edition) and Windows 2000, with Terminal Services installed.

> **Note** For technical support and questions regarding Citrix MetaFrame, go to http://support.citrix.com

SilkPerformer

Working with SilkPerformer

For details regarding the following SilkPerformer functionality, please see the *SilkPerformer Web Load Testing Tutorial:*

- User profiles
- Baseline performance
- Monitoring templates
- Workload
- Running and monitoring tests
- Exploring test results

See the *Silk TrueLog Explorer User Guide* for details regarding working with TrueLogs and visual script customization.

SilkPerformer Benefits

SilkPerformer is the industry's most powerful and easiest to use enterprise-class load and stress testing tool. Visual script generation techniques and the ability to test multiple application environments with thousands of concurrent users allow you to thoroughly test your enterprise applications' reliability, performance, and scalability before they're deployed—regardless of their size and complexity. Powerful root cause analysis tools and management reports help you isolate problems and make quick decisions—thereby minimizing test cycles and accelerating your time to market.

Ensure the scalability, performance, and reliability of your enterprise applications. SilkPerformer ensures the quality of your enterprise applications by measuring their performance from the end-user perspective, as well as internally, in a variety of workload scenarios and dynamic load conditions.

Test remote components early in the development cycle. Dramatically reduce the cost of bugs in your multi-tier enterprise application by testing the functionality, interoperability, and performance of remote components early in the development cycle—even before client applications have been built. You can rapidly generate test drivers for Web services, .NET remoting objects, EJB's and Java RMI objects by exploring them via a point & click interface. Alternately, you can reuse unit test drivers written by developers for concurrency tests or you can build new test cases directly in Java and other .NET languages, such as C# and VB.NET, using SilkPerformer's Visual Studio .NET Add-In.

Pinpoint problems easily for quick resolution. SilkPerformer's unrivaled TrueLog™ technology for HTML, XML, SQL, Oracle Forms, Citrix, TCP/IP, and UDP based protocol data provides full visual root-cause analysis from the end-user perspective. TrueLogs visually recreate the data that users provide and receive during load tests—for HTML pages this includes all embedded objects—enabling you to visually analyze the behavior of your application as

errors occur during load tests. In addition detailed response timer statistics help you uncover the root causes of missed Service Level Agreements before your application goes live.

Reusing projects

SilkPerformer's extended workflow simplifies and deepens its integration with Test Manager (Borland's test management solution)

By clicking SilkPerformer's new *Reuse Project* button, test projects can be uploaded to and reused by *SilkCentral Test Manager* (for test automation). See SilkCentral Test Manager documentation for details.

Support for Citrix MetaFrame Terminal Services

SilkPerformer provides record and replay support for the load testing of applications that are hosted via Citrix MetaFrame terminal services.

To ensure accurate simulation of users accessing Citrix MetaFrame terminal services, SilkPerformer uses the Citrix ICA Simulation Client to generate Citrix traffic. The Citrix ICA Simulation Client is a real Citrix ICA client that runs in "headless" mode (i.e., without graphical output). To avoid influencing Citrix MetaFrame server load test results, SilkPerformer load tests Citrix MetaFrame terminal services in an entirely non-intrusive manner. In fact, SilkPerformer doesn't require an installation on the Citrix MetaFrame Presentation server - making test preparation on the server side obsolete.

Generating load solely on the client side also allows SilkPerformer to load test load-balanced Citrix server farms.

SilkPerformer supports all Citrix deployment models, including application sharing, desktop sharing, and NFuse browser-based sharing.

Script-to-screen synchronization technology

To ensure that recorded user actions can be reliably replayed under all load conditions and not fall out of sync with on-screen events - even when server response slows - SilkPerformer provides a unique script-to-screen synchronization technology.

For common desktop applications that use multiple application windows, SilkPerformer's ICA recorder automatically generates synchronization functions based on window events (e.g., application window pop-ups, window caption changes, etc.).

For applications that always display data in a single application window (e.g., browser applications), SilkPerformer allows you to specify synchronization events based on screen regions. During recording you simply mark the screen regions that display the content that virtual users need to wait for and SilkPerformer's recorder scripts such screen synchronization functions automatically.

Synchronization functions also act as content verification functions during load testing, providing you with the ability to verify the response screens of terminal services under real load conditions.

In addition to the automated script generation of the SilkPerformer Citrix Recorder, SilkPerformer also provides a powerful BDL API for effectively customizing Citrix test scripts.

Optical character recognition

SilkPerformer's support for *optical character recognition (OCR)* simplifies session-dependent verifications and parsing by recognizing text values in the screengrabs of captured application states.

As with other TrueLog formats (Web, database, etc), verification and parsing functions are added using TrueLog Explorer after script recording.

See "Verification & Parsing via OCR" for details.

Alternate modes for replay

SilkPerformer provides the following two alternate modes for replaying Citrix test scripts:

- Animated mode (for debugging)
- Thin mode (for load testing up to high user levels)

In animated mode, SilkPerformer runs Citrix test scripts in full animation using SilkPerformer's Citrix Player software. The SilkPerformer Citrix Player is a fully functional "headful" Citrix ICA simulation client that shows all test script user input, such as mouse moves and key strokes, as they occur. This mode also provides the ability to execute Citrix test scripts step-by-step and log all script actions to an output window.

In thin mode, SilkPerformer can execute Citrix test scripts with multiple concurrent users to run full load tests against Citrix terminal server farms. To analyze what happens on the virtual user front-end (e.g., when errors occur), SilkPerformer generates TrueLog On Error files for front-end error analysis (see chapter 1.5).

Script customization

In addition to a powerful BDL API for Citrix that enables programmers to effectively customize Citrix test scripts, SilkPerformer also provides powerful TrueLog technology for Citrix - offering easy visual analysis of script output and visual script customization.

TrueLogs provide complete visual representation of all actions that are generated by test scripts, and the results of those actions. Citrix TrueLogs clearly show all user interactions (e.g., mouse moves and key strokes) as well as screen transitions of recorded and replayed user sessions.

With visual customization you can customize user input data such as mouse and keyboard actions using TrueLog Explorer's intuitive point-and-click interface. Simply select the input values that you wish to customize and then choose customization functions (e.g., change the coordinates of mouse clicks, randomize user input data entered via the keyboard, etc). All customization

functions are then generated and automatically inserted into BDL scripts. No manual scripting is required.

This visual approach to analyzing and customizing Citrix sessions is easier than all script-based approaches to Citrix terminal service testing.

Front-end error analysis

Using SilkPerformer's TrueLog On Error functionality for Citrix, you can visually inspect the actions of specific Citrix users and the responses they receive from Citrix MetaFrame servers that result in error conditions. With Citrix TrueLogs you can visually analyze error conditions from the virtual user's perspective (front-end).

1

Defining Load Test Projects

Introduction

This chapter explains how to define a Citrix load-test project in SilkPerformer.

What you will learn

This chapter contains the following sections:

Section	Page
Overview	7
Prerequisites	8
Defining a Load Test Project	9

Overview

The first step in conducting a Citrix load test is to define the basic settings for your SilkPerformer load-test project. A project is given a name, and optionally, a brief description. The type of application to be tested is *Citrix MetaFrame,* though a range of other choices are available for other projects, encompassing all the major traffic that is encountered in e-business today on the Internet and on the Web, including the most important database and distributed applications.

Though the settings that you specify will be associated with a specific load-testing project, later you'll find that it's easy to switch between projects, to edit projects, and to save projects so that they can later be modified and reused.

A project contains all the resources that are required to complete a load test. These include a workload, one or more profiles and test scripts, all the data files that are accessed from scripts, a specific number of agent computers, and information for server-side monitoring.

Prerequisites

Before defining a Citrix project you must install the Citrix client software and configure how the Citrix MetaFrame server is to handle time-outs.

Installing the Citrix client

You have two options for installing the Citrix client:

- Install Citrix client software from the *SilkPerformer* installation CD
- Download Citrix client software from the Citrix Web site

Procedure To install the Citrix client from the SilkPerformer CD:

1. Insert the *SilkPerformer* installation CD
2. Browse to and select *\Redist\Citrix Client\ica32.msi*
3. Execute the Citrix setup program and edit default options as required

Procedure To download the Citrix client:

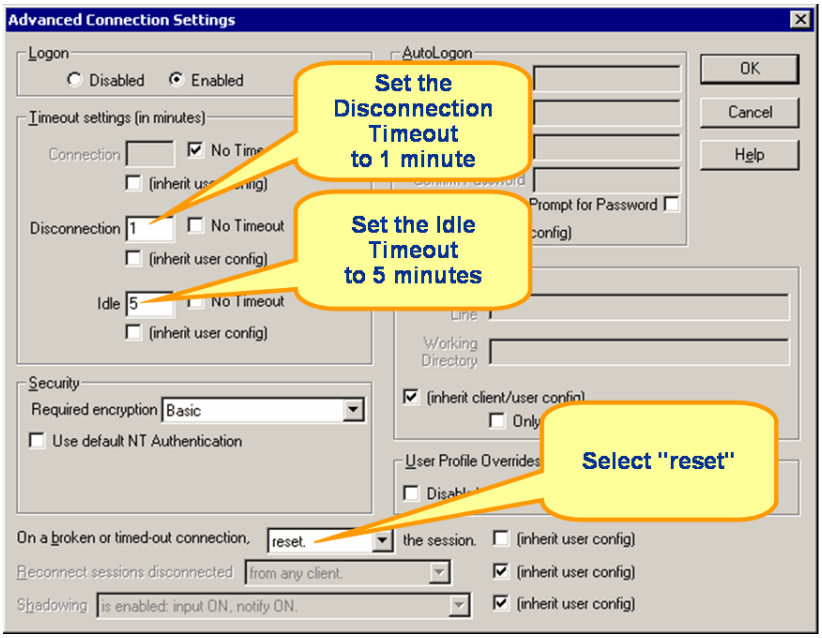
1. Navigate to *http://www.citrix.com*
2. Click *Download.*
3. Click *Clients*
4. Select *Program Neighborhood Version 7.x*
5. Select a language version
6. Click *Get Software*
7. Click *Download here*
8. Execute the Citrix setup program and edit default options as required

Configuring Citrix MetaFrame

Procedure To configure how Citrix MetaFrame server handles time-outs:

1. Go to *Start/Programs/Citrix/MetaFrame XP* and launch *Citrix Connection Configuration*
2. Double-click *ica-tcp* connection.
3. Click *Advanced*. The *Advanced Connection Settings* dialog appears.
4. Set the *Disconnection* timeout to 1 minute.
5. Set the *Idle* timeout to 5 minutes.
6. Select *reset* from the *On a broken or timed-out connection* drop-down list. If this option is not selected sessions will remain open after replay stops due to broken or timed-out connections. This will generate replay

problems when users next log into sessions as scripts will continue from the point where they left off in the previous session. Sessions then need to be ended manually.



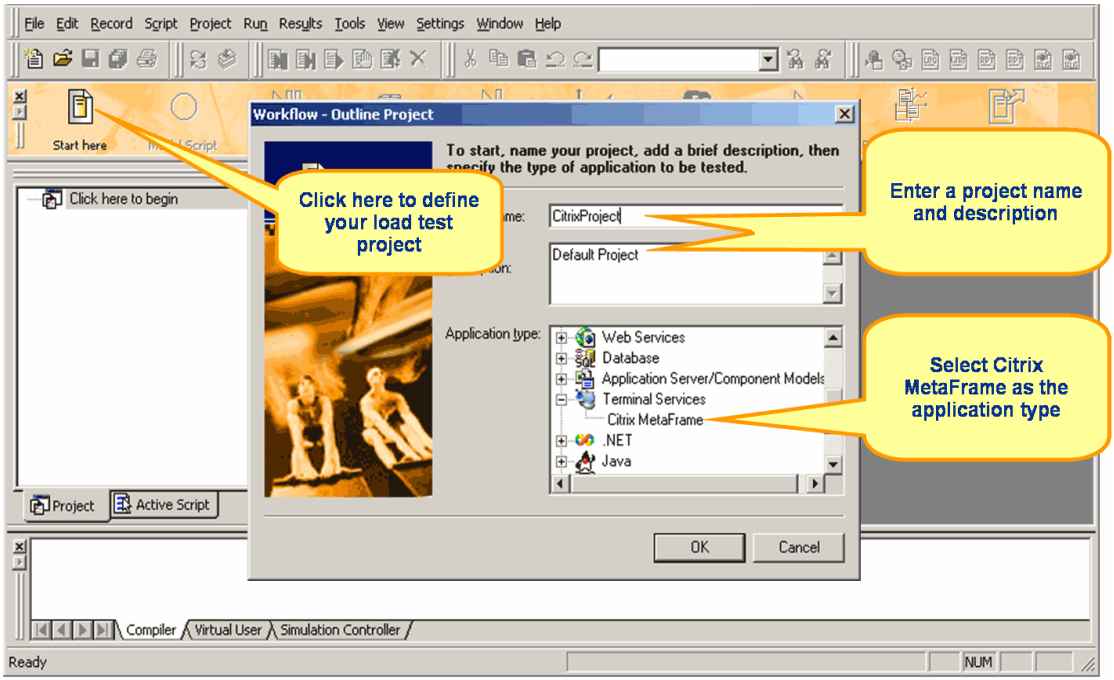
Defining a Load Test Project

The first step in creating a load test project is to define the project—giving the project a name, an optional description, and specifying the application type under test.

Procedure To define a Citrix load test project:

1. Click the *Start here* button on the SilkPerformer Workflow bar.
2. The *Outline Project* dialog opens.
   Enter a project name in the *Project name* field.
3. Enter a description for the project in the *Project description* field.
4. Select *Citrix MetaFrame* in the *Application type* field.

5 Click *OK* to create a project based on your settings.



2

Creating Test Scripts

Introduction

This tutorial explains how to model Citrix load test scripts and try out test scripts via TryScript runs.

What you will learn

This chapter contains the following sections:

Section	Page
Overview	11
Creating a Load Test Script	12
Screen Synchronization & Verification	21
Verification & Parsing via OCR	24
Trying Out a Generated Script	31

Overview

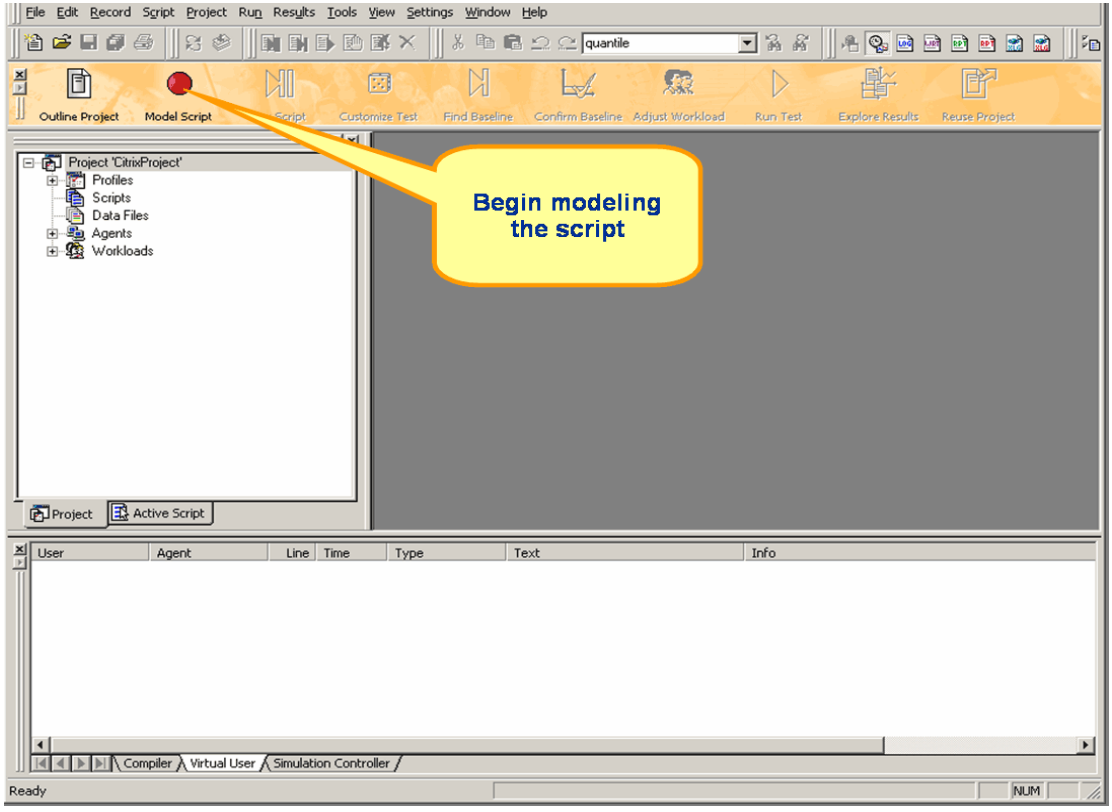
The easiest approach to creating a load test script is to use the SilkPerformer Recorder, SilkPerformer's engine for capturing and recording traffic and generating test scripts.

The SilkPerformer Recorder captures and records the traffic between a client application and the server under test. When recording is complete, the SilkPerformer Recorder automatically generates a test script based on the recorded traffic. Scripts are written in SilkPerformer's scripting language, *Benchmark Description Language (BDL)*.

Creating a Load Test Script

Procedure To model a load test script:

1 Click the *Model Script* button on the SilkPerformer Workflow bar.

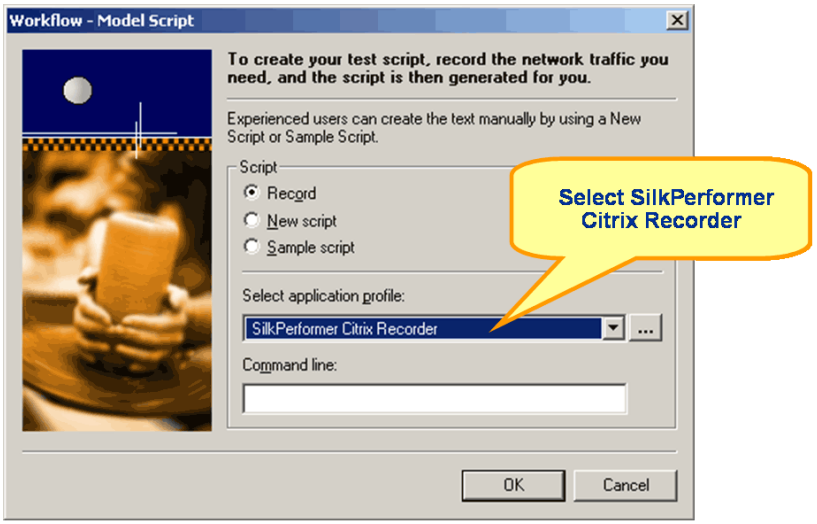


2 The *Model Script* dialog appears. Select *Record* in the *Script* area of the dialog.

3 From the *Select application profile* drop-down list, select *SilkPerformer Citrix Recorder* to record a Citrix application.

Note The *SilkPerformer Citrix Recorder* application profile is appropriate for testing standard Citrix clients that connect to Citrix MetaFrame servers (for NFuse sessions, see "Testing NFuse Sessions").

4 Click *OK*.



5 The SilkPerformer Recorder then opens in minimized form along with the SilkPerformer Citrix Recorder.

Note If correct login credentials are not available at startup, you will be presented with the *Connect* dialog (see below). The *Connect* dialog is also accessible via the *Connect* button in the upper left corner of the SilkPerformer Citrix Recorder.

6 On the *Connect* dialog, enter the name of the Citrix server you will be recording in the *Server* field.

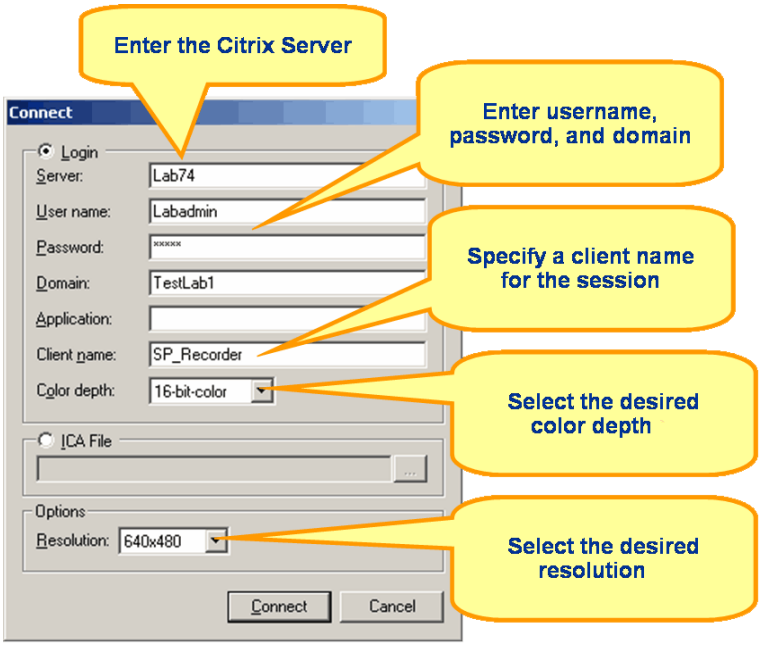
7 Complete the *User name, Password,* and *Domain* fields for the server under test.

8 The *Client name* field enables you to explicitly specify a client name for your session. The default client name for the Citrix recorder is *SP_Recorder* (The default client name for the Citrix Player is *SP_User_x.)* If you specify a different client name for recording, then a *CitrixSetClientName* function will be inserted into the script. In such cases you must customize the client name value, otherwise all users will use the same client name, which can lead to replay problems.

9 Select the desired *Color depth* for recording.

10 Select the desired screen *Resolution* for recording.

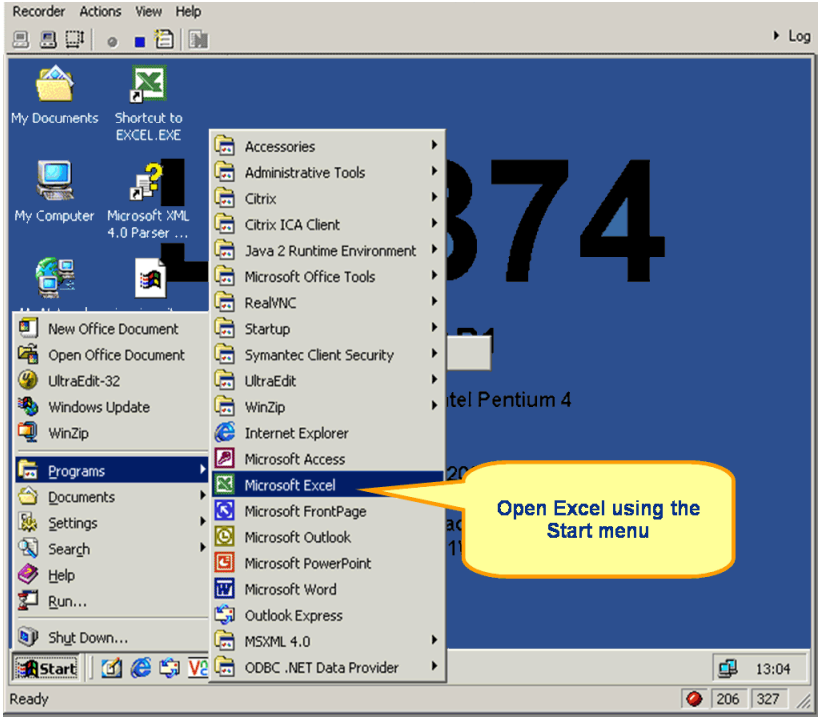
11 Click *Connect* to begin the Citrix session.

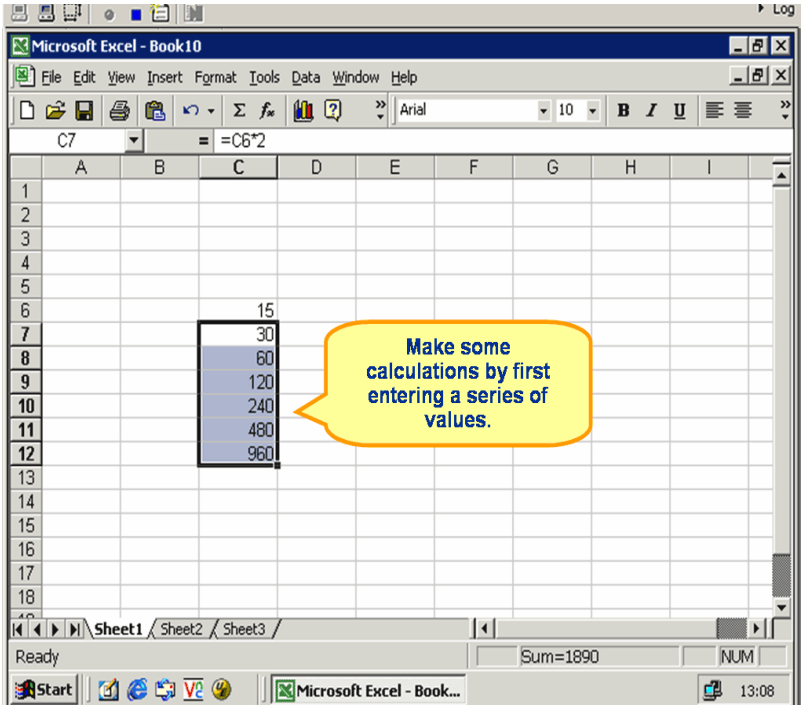


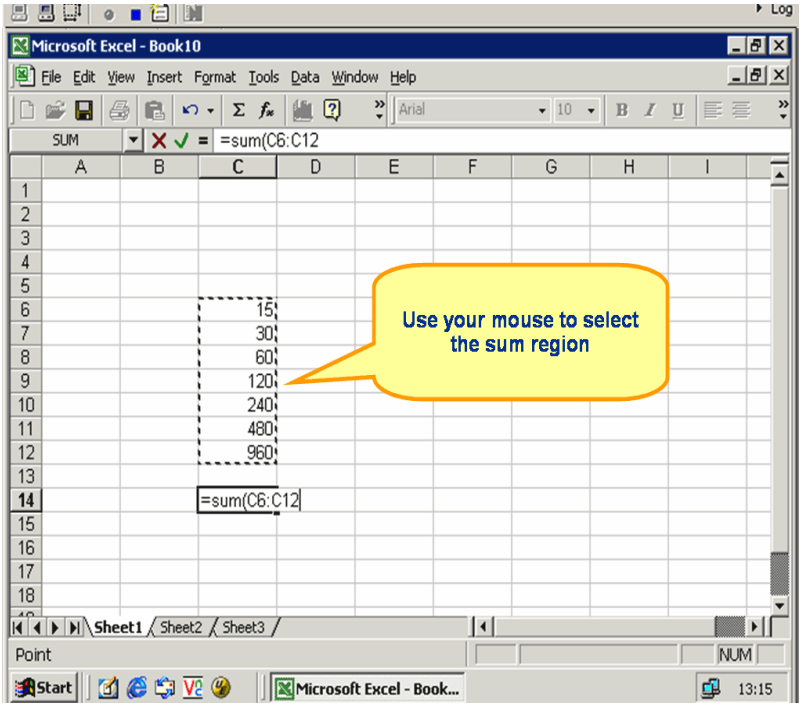
12 Interact with the shared desktop in the Citrix Recorder in the same way that you want your virtual users to act during the load test (i.e., click links, open applications, and enter data). Your actions will be captured and recorded by the Citrix Recorder.

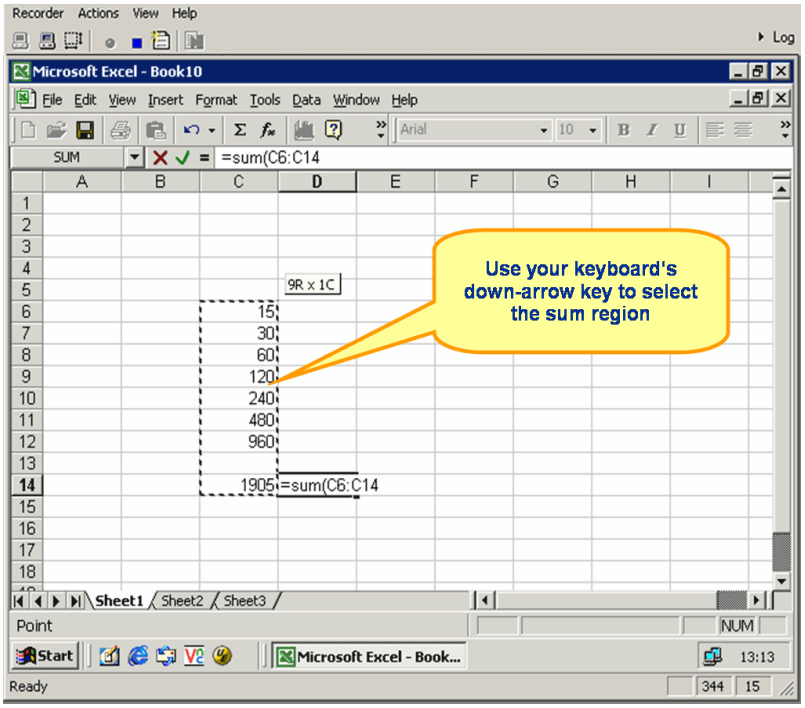
Note The Citrix Recorder supports session sharing, allowing you to start additional published applications in the existing session (use the *Run Application* button on the Citrix Recorder toolbar). Use the *Select Window* button to switch between applications.

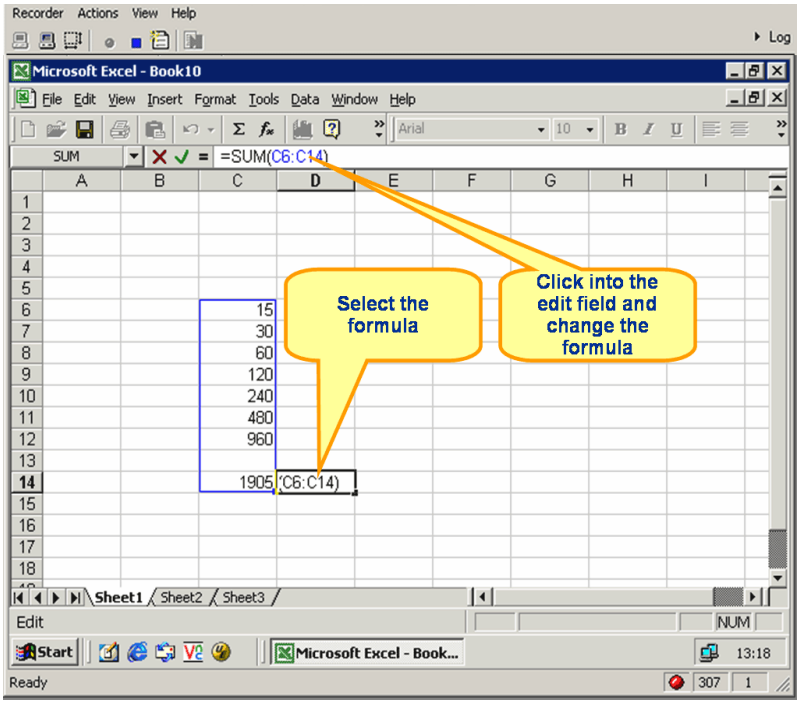
The following example citrix session includes simple Excel calculations in which the mouse and keyboard are used to open Excel, enter new data values, insert an AutoSum formula, select a screen region, edit the AutoSum formula, and close Excel.

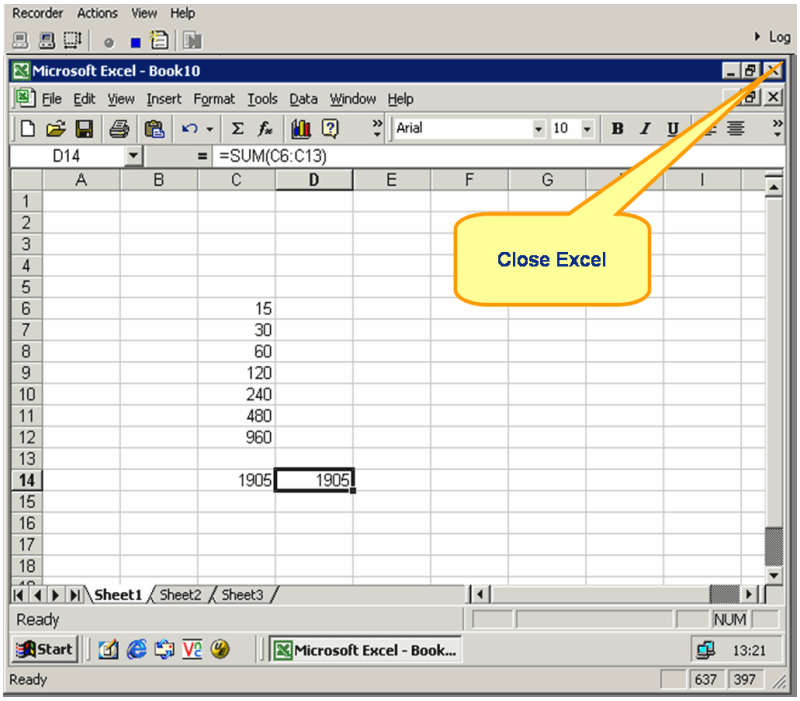


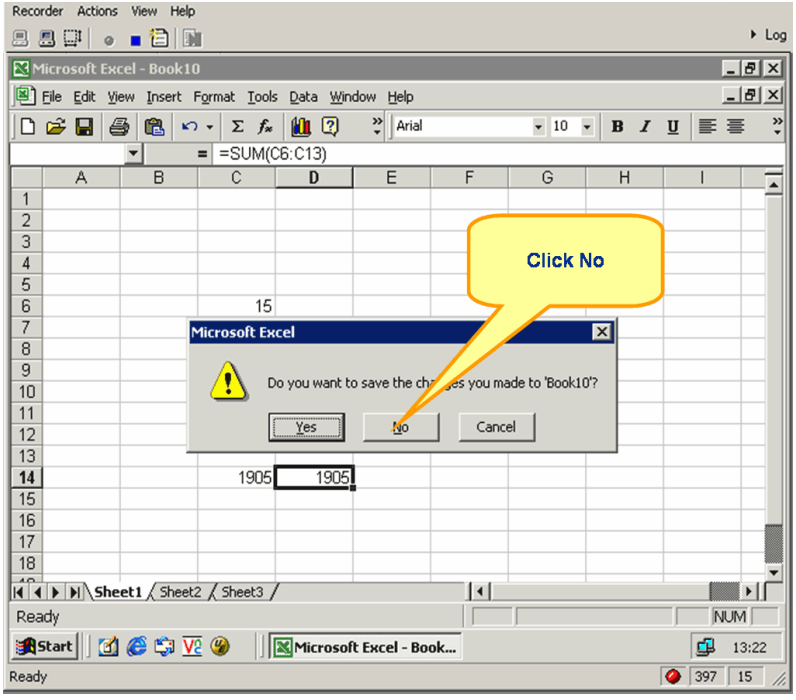

Microsoft Excel - Book10
File Edit View Insert Format Tools Data Window Help
Arial
10
C7
=C6*2
15
30
60
120
240
480
960
Make some calculations by first entering a series of values.
Sheet1
Sheet2
Sheet3
Ready
Sum=1890
NUM
Start
Microsoft Excel - Book...
13:08


Microsoft Excel - Book10
File Edit View Insert Format Tools Data Window Help
SUM
=sum(C6:C12
15
30
60
120
240
480
960
=sum(C6:C12
Use your mouse to select the sum region
Sheet1 Sheet2 Sheet3
Point
NUM
Start
Microsoft Excel - Book...
13:15


Recorder Actions View Help
Log
Microsoft Excel - Book10
File Edit View Insert Format Tools Data Window Help
SUM
=sum(C6:C14
9R x 1C
15
30
60
120
240
480
960
1905
=sum(C6:C14
Use your keyboard's down-arrow key to select the sum region
Sheet1 Sheet2 Sheet3
Point
NUM
Start
Microsoft Excel - Book...
13:13
Ready
344 15


Recorder Actions View Help
Log
Microsoft Excel - Book10
File Edit View Insert Format Tools Data Window Help
SUM
=SUM(C6:C14)
15
30
60
120
240
480
960
1905
'C6:C14)
Select the formula
Click into the edit field and change the formula
Sheet1 Sheet2 Sheet3
Edit
NUM
Start
Microsoft Excel - Book...
13:18
Ready
307 1


Recorder Actions View Help
Log
Microsoft Excel - Book10
File Edit View Insert Format Tools Data Window Help
Arial 10
D14
=SUM(C6:C13)
Close Excel
15
30
60
120
240
480
960
1905
1905
Sheet1 Sheet2 Sheet3
Ready
NUM
Start
Microsoft Excel - Book...
13:21
Ready
637 397


Recorder Actions View Help
Log
Microsoft Excel - Book10
File Edit View Insert Format Tools Data Window Help
Arial 10
=SUM(C6:C13)
Click No
15
Microsoft Excel
Do you want to save the changes you made to 'Book10'?
Yes
No
Cancel
1905
1905
Sheet1 Sheet2 Sheet3
Ready
NUM
Start
Microsoft Excel - Book...
13:22
Ready
397 15

13 When you're done recording your Citrix session, click the *Stop Recording* button on the SilkPerformer Citrix Recorder. The *Generating script* progress window appears.

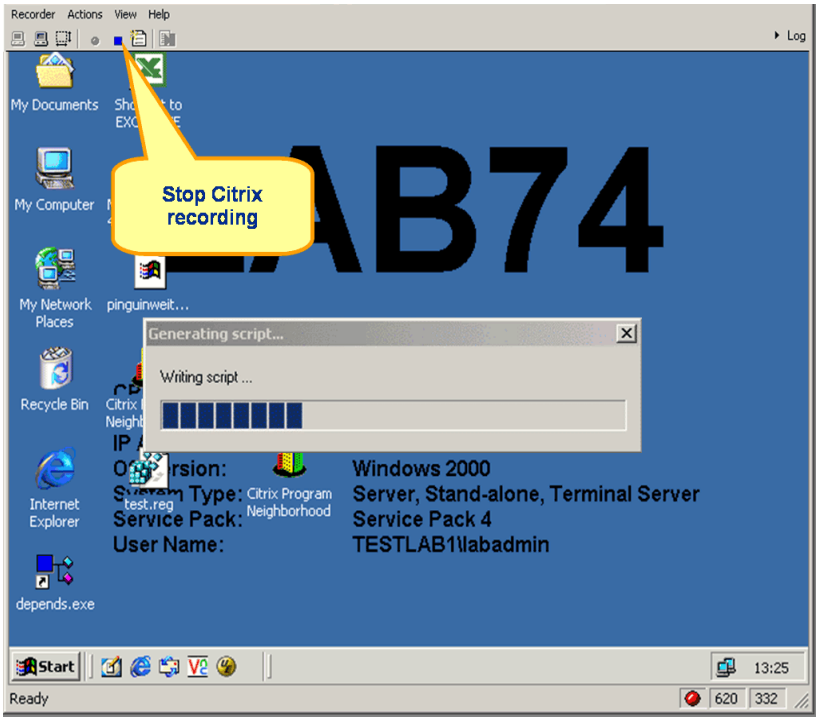
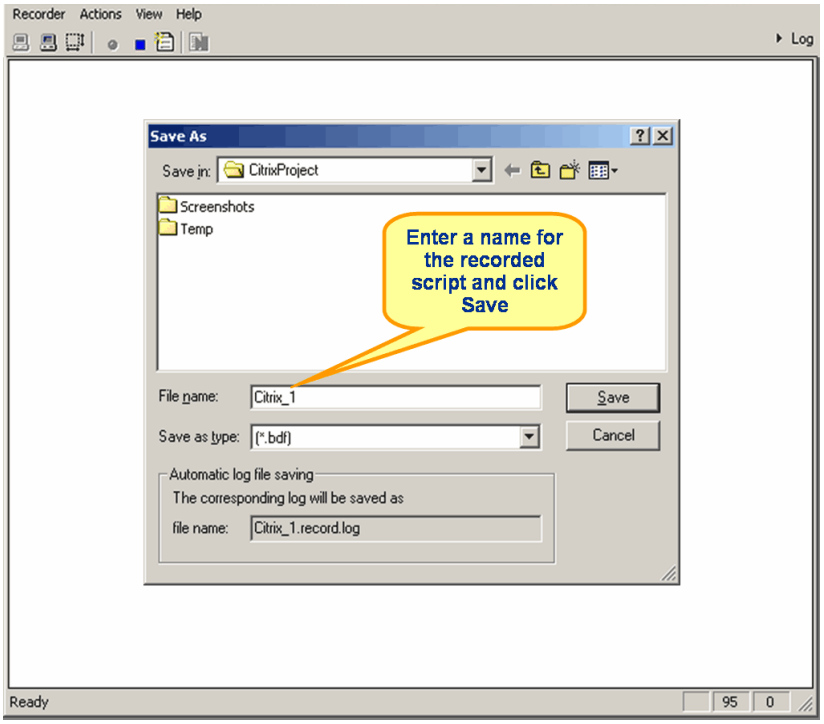


Figure 1:

14 The *Save As* dialog appears. Save the script with a meaningful name.



15 A newly generated load test script that's based on the Citrix interactions you recorded appears in the SilkPerformer script editor window.

Screen Synchronization & Verification

SilkPerformer supports bitmap and window verification for applications that are hosted by Citrix MetaFrame servers. *Screen synchronization* offers a means of verifying replayed Citrix content. Screen synchronizations differ from standard script verifications in that they allow for verification of window and screen region appearances (not input values) and they are inserted during script recording, not during subsequent script customization. Screen synchronizations are particularly useful for synchronizing subsequent user input that is displayed in browsers, or similar interfaces (i.e., Citrix application windows), because such applications don't support "automatic synchronization" through window events such as user input and text display.

> **Note** See "Verification & Parsing via OCR" for information regarding onscreen value verification and parsing. Response data verification is not supported for Citrix.

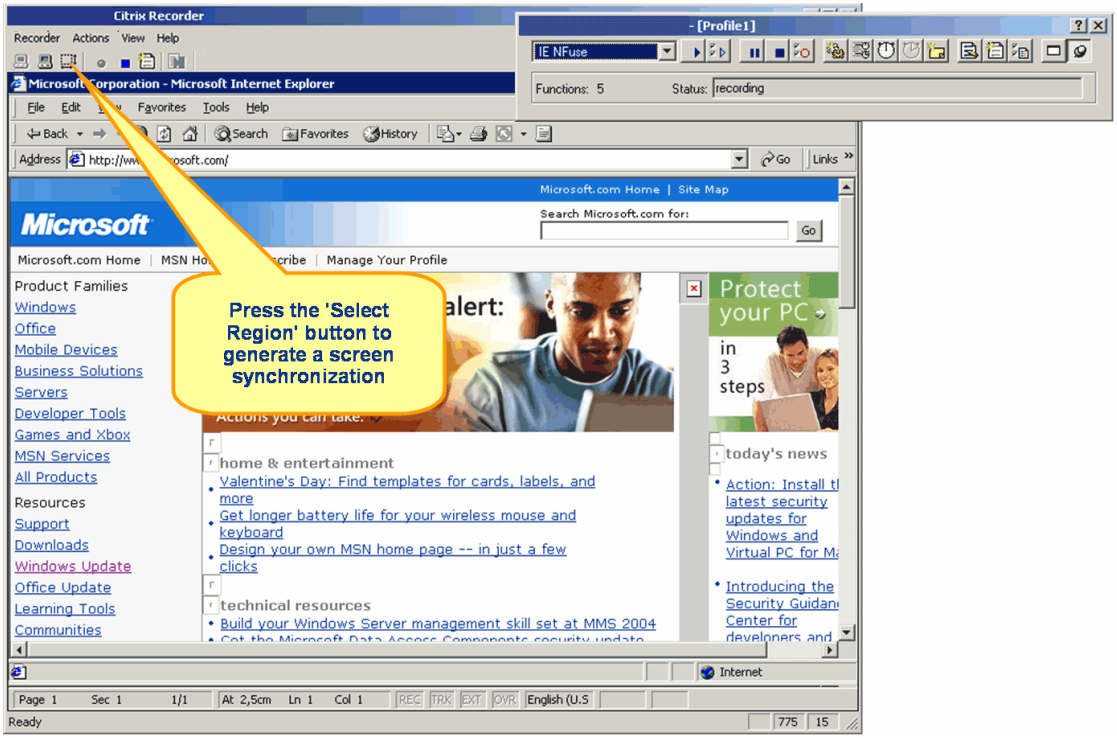
SilkPerformer relies on hash values to verify replayed bitmaps against recorded bitmaps. Hash values are computer-readable values that reflect bitmap specifications such as size, position, resolution, and color depth (see "Understanding screen synchronization" for details)

> **Note** To verify replay screen regions against hash values that are captured at recording it's necessary that the same color depth that is used during recording also be used during replay. Scripts will fail if these specifications are not maintained because changes as small as a single pixel can change hash values and result in replay content appearing to be different from recorded content.
>
> **Note** It's vital that windows that are maximized during recording be maximized during replay. This is because replay cannot change the state of windows (it can only resize and move them). So if a window state changes (e.g., from *Maximized* to *Restored),* then it is likely that some user input in the script caused the change (e.g., clicking the *Restore* button). On replay the user will click at the same position (which is now the *Maximize* button) and consequently a different operation will be executed—the subsequent *CitrixWaitForWindowRestore* function will fail.

Procedure To generate a screen region synchronization during recording:

1 Record a Citix session as described in the "Creating a Load Test Script" section of this chapter.

2 During recording, press the *Select Region* button on the SilkPerformer Citrix Recorder.



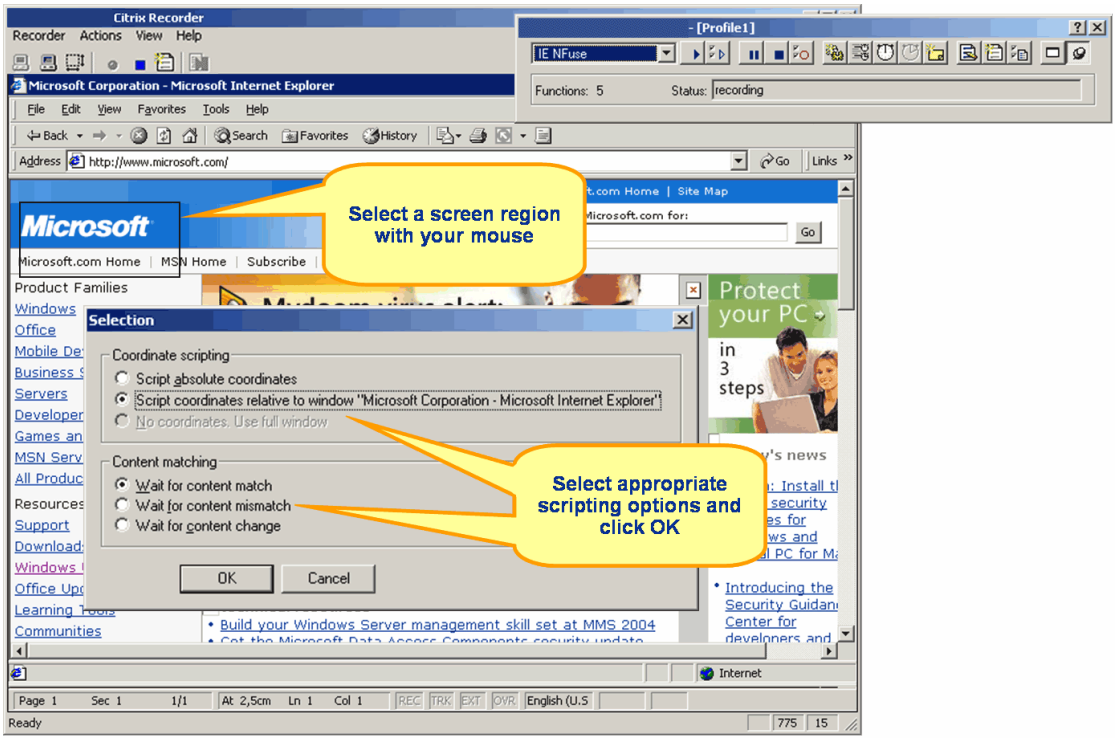
3 Click your mouse onscreen and drag it to select a screen region for which you wish to generate a bitmap synchronization.

Note Because differences as small as a single pixel can cause synchronization processes to fail, it's recommended that you select the minimum screen area required for text verifications. Otherwise unanticipated screen differences (e.g., disabled toolbars) may affect verification results.

4 The *Selection* dialog appears. Specify how you wish to have screen region coordinates scripted *(Script absolute coordinates, Script coordinates relative to window,* or *No coordinates. Use full window)*. When windows are maximized there is effectively no difference between absolute and relative coordinates. When windows are not maximized, relative coordinates are measured from the top-left corner of the Citrix Recorder window.

5 Specify the *Content matching* type that the Citrix player should wait for during replay *(content match, content mismatch,* or *content change).*

6 Click *OK* to add the synchronization to your Citrix test script.



Understanding screen synchronization

Screen synchronization is achieved via *CitrixWaitForScreen* functions, which are not scripted automatically by the recorder. These functions are inserted via the *Screen Region* dialog during recording. *CitrixWaitForScreen* functions compare replay and record bitmaps to determine whether or not they are identical. Hash values, as opposed to actual bitmaps, are used to compare the images. This is to save on resource consumption during replay.

Verification & Parsing via OCR

SilkPerformer's support for *optical character recognition (OCR)* simplifies session-dependent verifications and parsing by recognizing text values in the screengrabs of captured application states.

As with other TrueLog formats (Web, database, etc), verification and parsing functions are added using TrueLog Explorer after script recording.

See the *Silk TrueLog Explorer User Guide* for full details regarding parsing and verification functions.

Window position and state

Window positions and state (maximized/minimized) is quite important for insuring accurate replay as the TrueLog Explorer scripts screen coordinates where selected text is to be read from relative to the desktop, not individual windows. So if a window appears in a different position during replay than it did during recording, OCR operations will not be able to locate the specified text. If it is not possible to specify an absolute position of the conversion-region, the script has to be manually updated by the use of coordinates relative to windows.

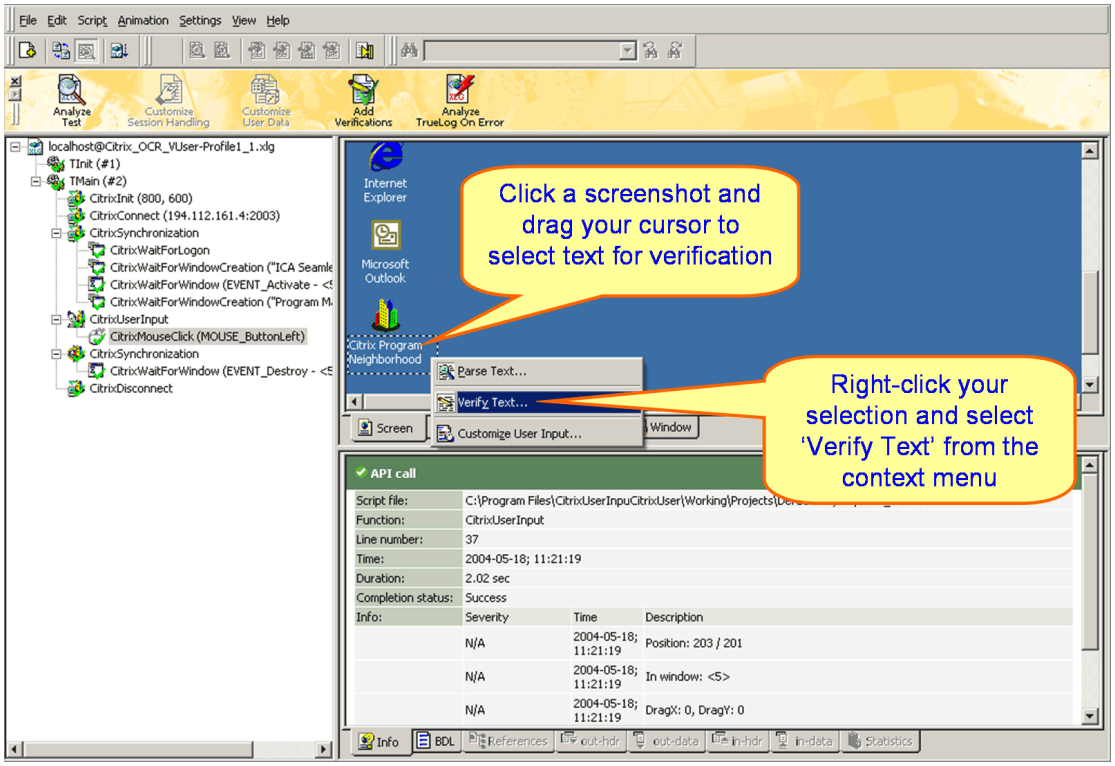
Adding OCR verification functions

Previously SilkPerformer supported only bitmap and window verification for applications hosted by Citrix MetaFrame servers. This type of verification didn't allow for the verification of session-dependent data (e.g., login names). Now, with OCR support offered through a third-party .dll, SilkPerformer enables the storing of recognizable text values in variables—thereby simplifying session-dependent verifications in Citrix load tests.

Procedure To generate an OCR verification function:

1. From SilkPerformer record a Citix session as described in the "Creating a Load Test Script" section of this chapter.
2. Run a TryScript run, with the *Animation* checkbox selected on the *TryScript* dialog. This opens TrueLog Explorer.
3. When the TryScript run is complete, select the API node that includes the bitmap screengrab of the screen on which you'd like to verify text.
4. Click and drag your cursor onscreen to select the screen region that includes the text you would like to use for verification.

5 Right-click in the selected area and select *Verify Text* from the context menu.



6 The *Insert Text Verification Function* dialog appears. The selected text is pre-loaded into the *constant value* edit box and the *constant value* radio button is selected by default.

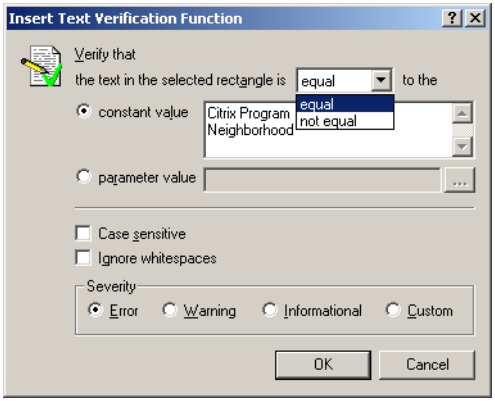
Alternative In addition to being able to verify against a constant value, you can also verify against a parameter—either an existing parameter or a new parameter. To verify against a parameter, select the *parameter* radio button. If a parameter already exists, clicking the *"..."* button will allow you to browse to and select a parameter. If no parameters exist, clicking the *"..."* button will launch the Parameter Wizard—enabling you to create a new parameter. See the *Silk TrueLog Explorer User Guide* for details regarding parameters.

7 From the *Verify that the text in the selected rectangle is* drop-down list, select *equal* or *not equal.*

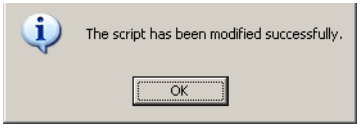
8 Specify whether or not the verification is to be *Case sensitive* or should *Ignore whitespaces.*

9 In the *Severity* portion of the dialog box, specify the severity that is to be raised if the verification returns a negative result *(Error, Warning, Informational,* or *Custom)*.

10 Click *OK*.



11 A confirmation dialog appears. Click *OK* to add the OCR verification function to your Citrix test script.



Adding OCR parsing functions

Procedure To generate an OCR parsing function:

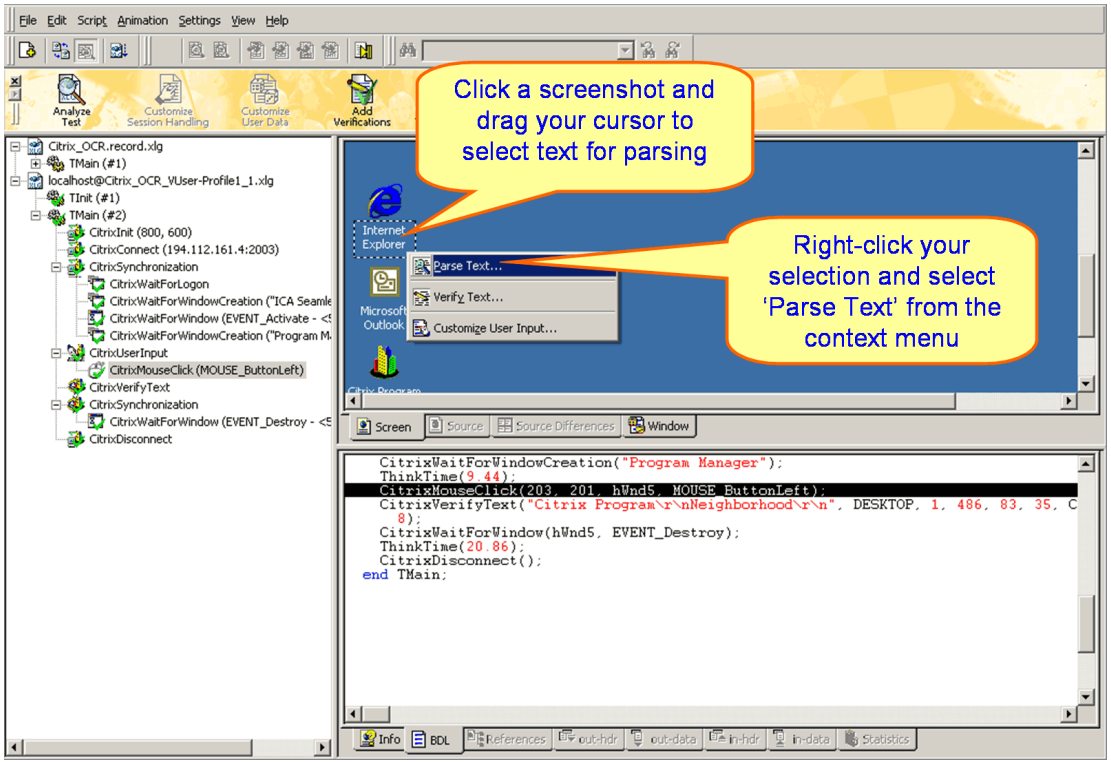
1 From SilkPerformer record a Citix session as described in the "Creating a Load Test Script" section of this chapter.

2 Run a TryScript run, with the *Animation* checkbox selected on the *TryScript* dialog. This opens TrueLog Explorer.

3 When the TryScript run is complete, select the API node that includes the bitmap screengrab of the screen on which you'd like to parse text.

4 Click and drag your cursor onscreen to select the screen region that includes the text you would like to parse.

5 Right-click in the selected area and select *Parse Text* from the context menu.

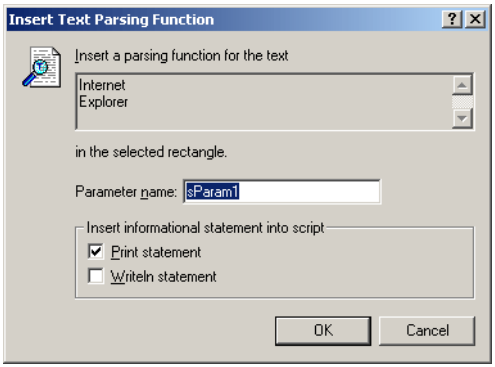


6 The *Insert Parsing Function* dialog offers parameters by which the parsing function can be configured. Though the default settings will likely be correct, you can adjust:

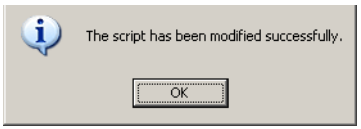
- *Parameter name* - Enter the name of the parameter that is to receive the result of the parsing function.
- *Informational statement insertion* - Select Print statement to insert an informational Print statement into the script after the Web page call. This writes the result of the parsing function to SilkPerformer's Virtual User Output window.

  Select *Writeln statement* (“write line” statement) to write the parsed value to an output file to facilitate debugging (in addition to writing the value to the Virtual User Output window as a *Print* statement does). Because generating output files alters load-test time measurements, these files should only be used for debugging purposes and should not be generated for full load tests.

7 Click *OK*.



8 A confirmation dialog appears. Click *OK* to add the OCR parsing function to your Citrix test script.



Understanding OCR verification & parsing

String verification via optical character recognition (OCR) is achieved using *CitrixVerifyText* API calls. These functions are inserted via TrueLog Explorer during script customization. *CitrixVerifyText* functions compare text strings in replay bitmaps to determine if they are identical.

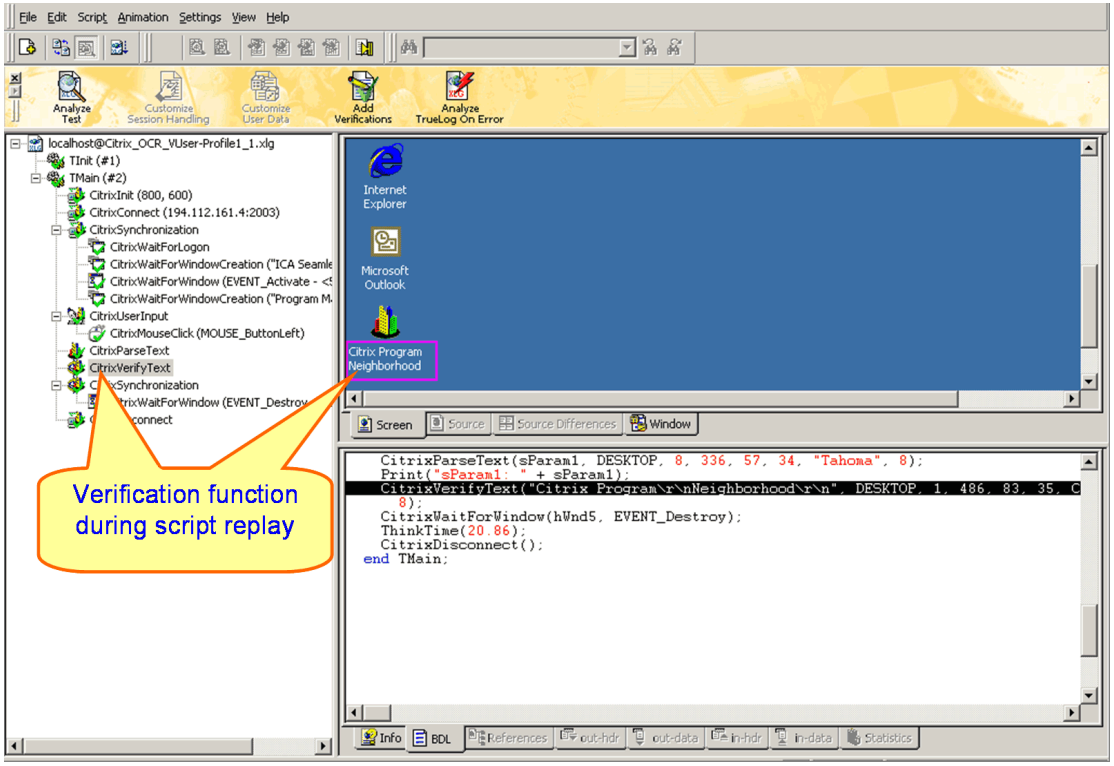
CitrixParseText functions are available for parsing text. These API calls work in the same way that standard parsing functions work (Web, database, etc).

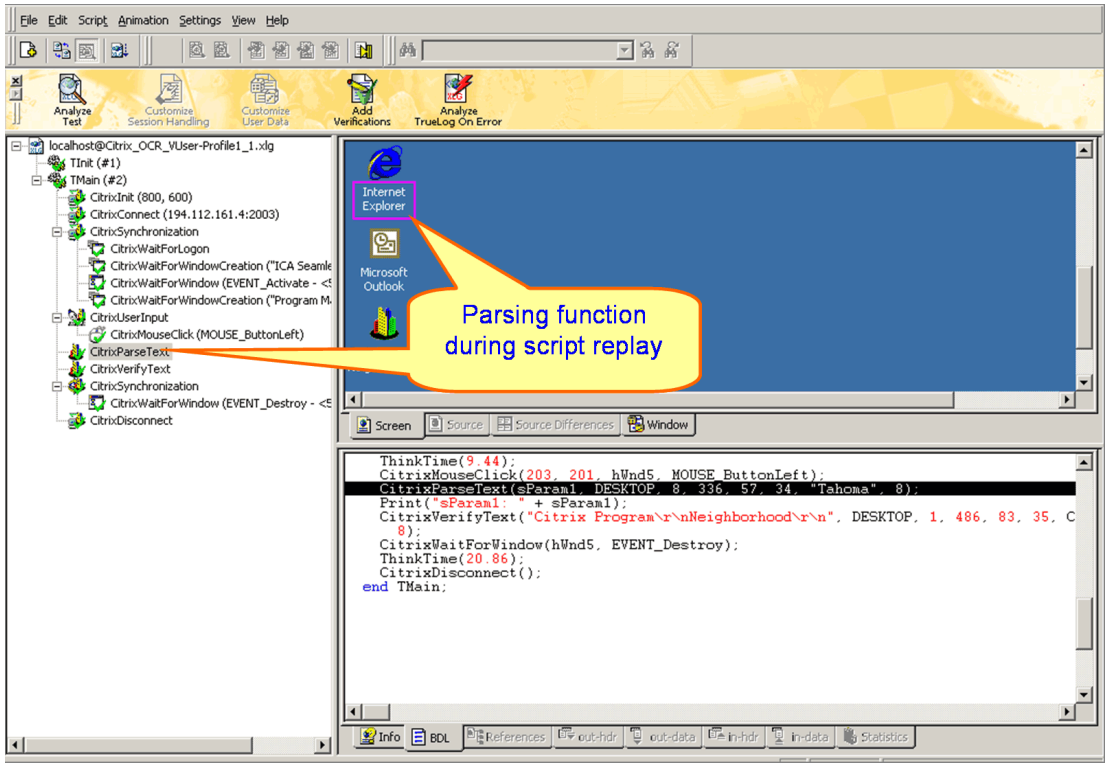
Optical character recognition relies on pattern databases to recognize varying fonts and text styles. Font databases must be generated before OCR can be run. See "Citrix System Settings for OCR" for details.

Only Citrix TrueLogs show verification and parsing API calls in the tree view. With other TrueLog modes (Web, database, etc), new API nodes are not added to the tree view.

> **Note** It is absolutely important to perform OCR operations on stable content, because when used on a frequently changing screen it is only a matter of timing which image to use for the conversion, which results in accidental results. When synchronizing on window events it is possible that the screen refresh is slightly delayed, which again results in timing dependent outcome. Therefore it is a good practice to either script a wait or a CitrixWaitForScreen function call before all OCR verification/ parsing functions.

The following two screen examples show the output of verification and parsing functions after TryScript runs.





Trying Out a Generated Script

Once you've generated a test script you should determine if the script runs without error via a *TryScript run.* A TryScript run will determine if a script accurately recreates the interactions you recorded.

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.

> **Note** The default option settings for Citrix TryScript runs do not include live display of data downloaded during testing (via TrueLog Explorer), though they do include the writing of log files, report files, and replay within the SilkPerformer Citrix Player.

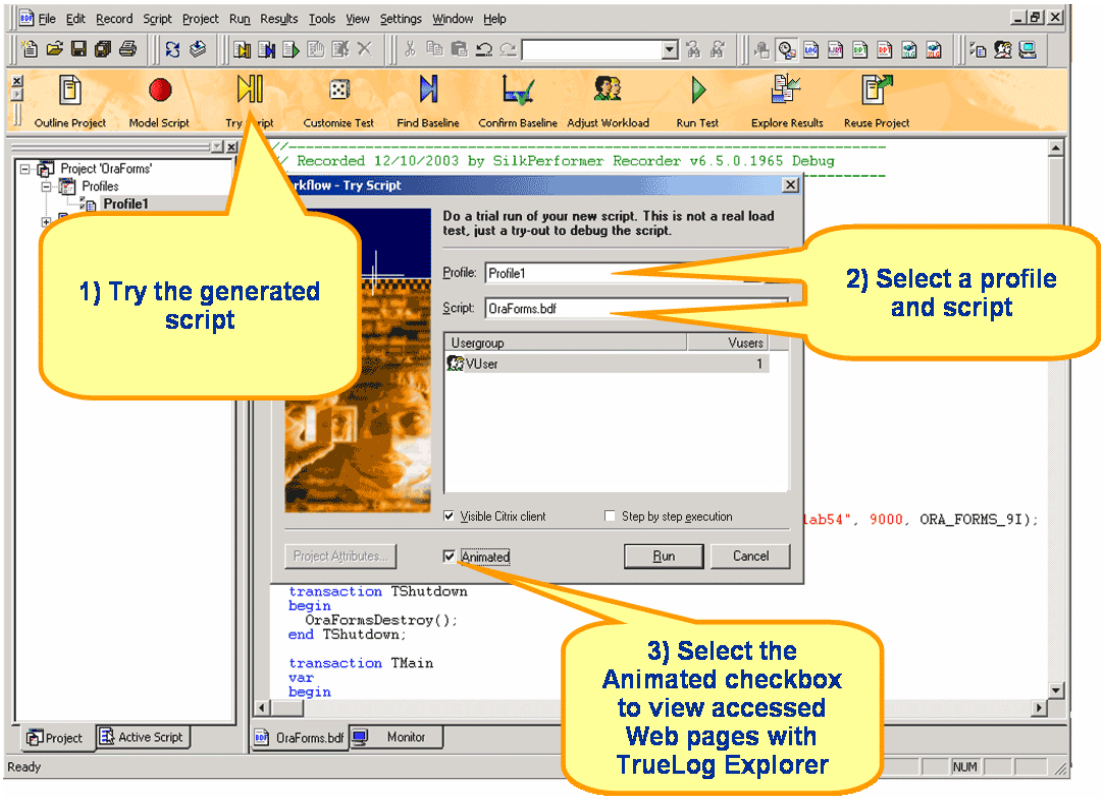
> **Note** If you have configured parsing or verification functions based on Citrix OCR support, you must generate an OCR font database before attempting a TryScript run; otherwise these functions may not operate correctly. See "Citrix System Settings for OCR" for details.

Analyzing a test script

Procedure To try out a load test script:

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 a live display of page content within TrueLog Explorer during replay, select the *Animated* checkbox.

   Note The *Visible Citrix Client* option (selected by default) enables visual replay in the SilkPerformer Citrix Player during TryScript runs.
3. Click *Run*.



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

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

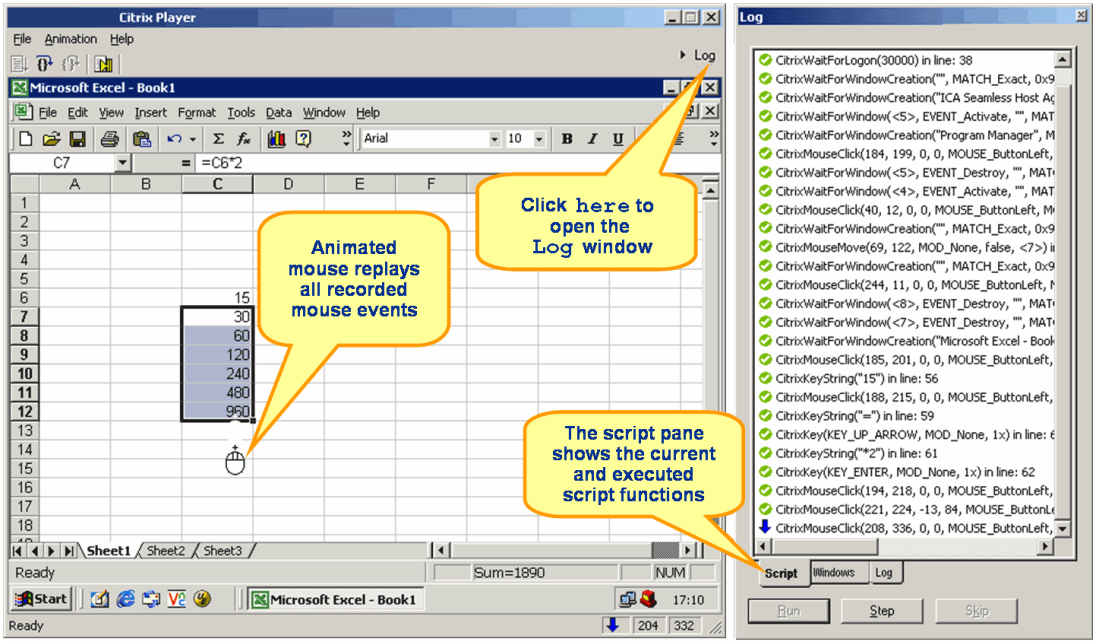
SilkPerformer Citrix Player

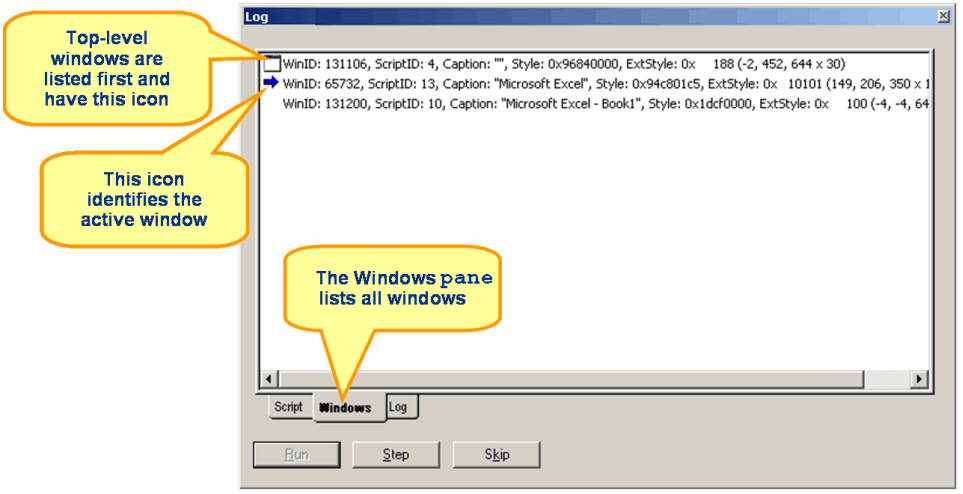
The SilkPerformer Citrix Player opens when TryScript runs begin, replaying all recorded actions in full animation. Mouse movements and operations are simulated with an animated mouse icon.

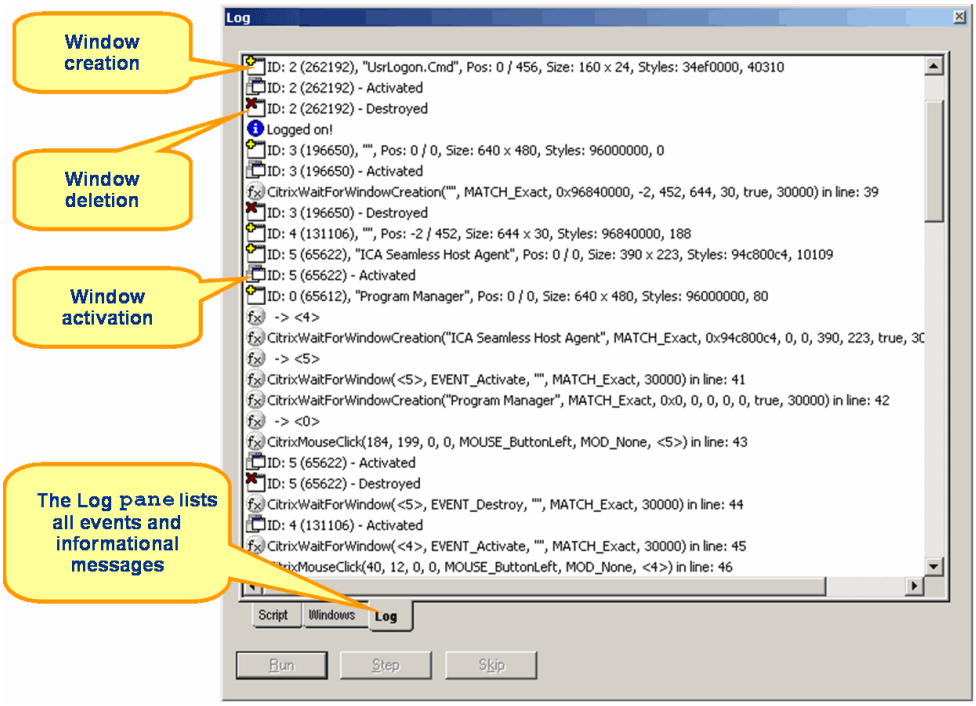
Click the *Toggle Log Window* button in the upper right corner of the player to open the *Log* window, The *Log* window includes three panes that detail different aspects of TryScript runs:

- *Script* - This pane lists all of the executed BDL script functions and the currently executing BDL function.
- *Windows* - This pane includes a stack of all the client windows of the current session—including window captions, styles, sizes, and positions. Top-level windows carry a window icon and are listed above sub-windows.
- *Log* - This pane lists all informational messages and events, including executed BDL functions, and window creation/activation/destruction.

In all panes, active functions and windows are indicated by a right-pointing blue arrow.







Step-by-step replay

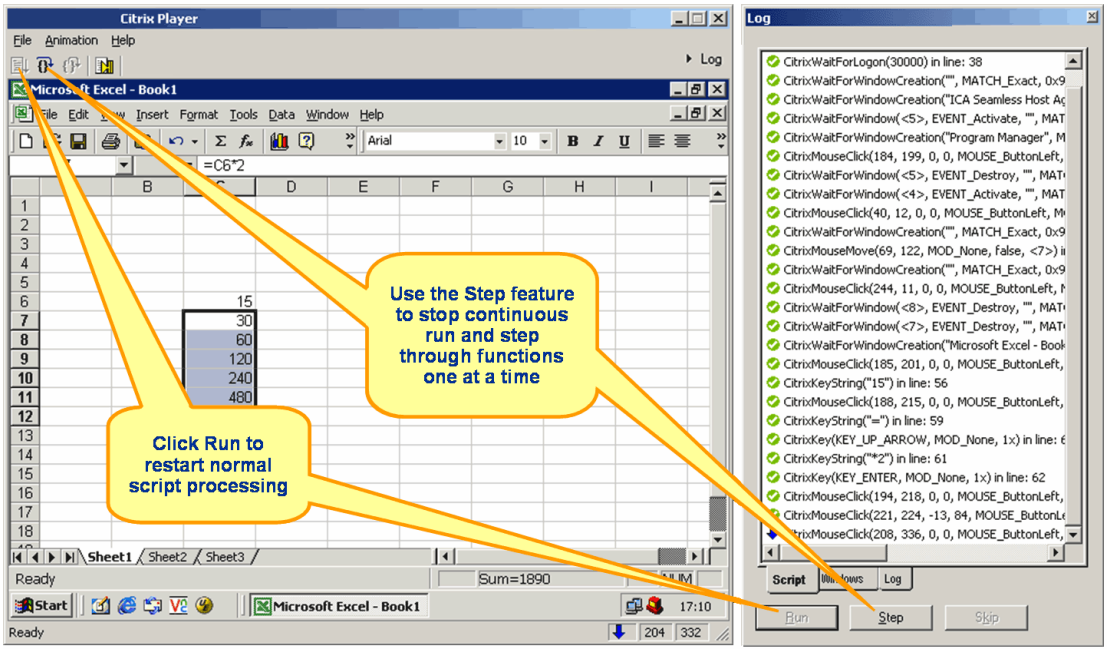
Step-by-step mode allows you to advance through replay one function at a time. This is helpful for close analysis and debugging.

Procedure To advance through replay step-by-step:

1. During a Citrix TryScript run, click the *Toggle Log Window* button in the upper right corner of the SilkPerformer Citrix Player to open the *Log* window.
2. Click the *Step* button.
3. Replay stops at the active function. A right-pointing arrow icon indicates the next function in the script. Click the *Step* button to execute that function.
4. Continue clicking the *Step* button to advance through the script.

   Note You can also enable step-by-step execution by selecting the *Step by step execution* checkbox on the TryScript dialog

5 Click the *Run* button to resume continuous script processing.



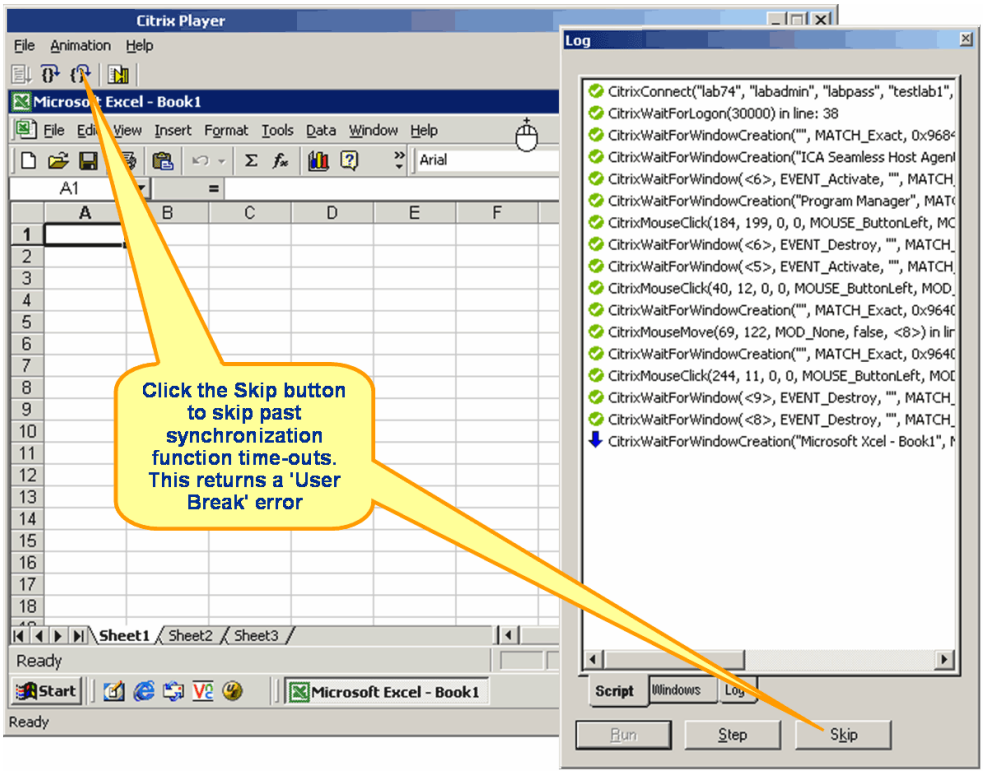
Skipping time-outs

The SilkPerformer Citrix Player waits for all time-outs that are encountered during replay. To avoid waiting the full duration of time-outs (the default time-out is 60 secs.), click the *Skip* button to advance past them. Clicking the *Skip* button generates “user break” errors into scripts.

Procedure To skip past synchronization function time-outs:

1 During a Citrix TryScript run, click the *Toggle Log Window* button in the upper right corner of the SilkPerformer Citrix Player to open the *Log* window.

2 When replay encounters a time-out, click the *Skip* button to advance to the next function. Note that this will generate a "user break" error into the test script.



Select Citrix script functions

Following is a brief description of some common SilkPerformer Citrix functions. See SilkPerformer online help for full details regarding all available Citrix script functions.

- *CitrixWaitForWindowCreation*, used for screen synchronization, is the most important Citrix function.

  The first parameter that synchronizations consider is the window caption. If during replay a window caption is returned that matches the caption of the window that was recorded during replay, then the verification succeeds.

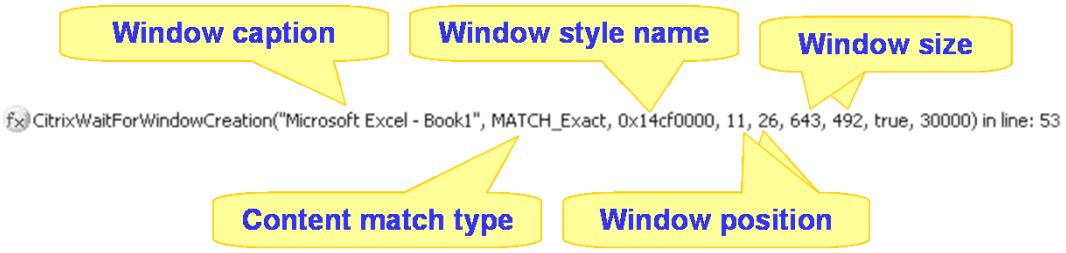
  > **Note** Wildcards (*) can also be specified at the end or beginning of window captions.

  The second parameter is the match parameter. Normally this is an exact, case-sensitive comparison of window caption strings.

The third parameter is the style of the window. If a window caption name is not available, then a portion of the style name is used to identify the window. Styles reflect whether or not windows were maximized during recording.

The fourth parameter is the position of the window. This may be a negative number, as maximized windows have a position of *"-4, -4"*. This parameter can be controlled via the *Force window position* profile setting (see "General Citrix Settings" for details). When this profile setting is enabled windows are automatically moved to the exact positions they held during recording.

The fifth parameter is window size. During replay windows must be resized to the same size they had during recording.



- *CitrixWaitForLogon* waits until a Citrix client logs on successfully or a specified timeout period expires.
- *CitrixWaitForScreen* captures screen regions and checks them against specified conditions (normally hash value captured at recording). If a condition doesn't match and the timeout period expires, the function call fails. CitrixWaitForScreen functions also use screen appearance to synchronize subsequent user actions. Based on provided parameters, this function waits until the image in a specified screen region changes, matches the hash value captured at recording, or does not match the hash value captured at recording.
- *CitrixGetScreen* takes a screenshot of a selected region and writes the screenshot to a file in the result directory. If the file name is omitted, it will be automatically generated by the user ID and hash value of the image.
- *CitrixGetScreenHash* retrieves the hash value of a selected screen.
- *CitrixSetOption* sets particular options, such as network protocol specification, speed screen latency reduction, data compression, image caching, mouse/keyboard timings and event queueing, client disconnect, synchronization time-outs, think times, TrueLog capture, and window position forcing.

- *CitrixWaitForWindow* waits until a specified window event (specified with the *nEvent* parameter) occurs. Such events may be an activation, destruction or caption change for a specified window. If the selected event is a caption change, a matching caption change for the specified window will satisfy the function. Captions can be specified explicitly using the *sCaption* and *nMatch* parameters.
- *CitrixKeyString* is the standard function for entering printable characters.
- *CitrixMouseClick* moves the mouse to a specified position and presses a specified button. Optionally, the mouse can be specified to move while the button is pressed. Key modifiers (such as Ctrl and Alt) can be provided.

Citrix TrueLogs

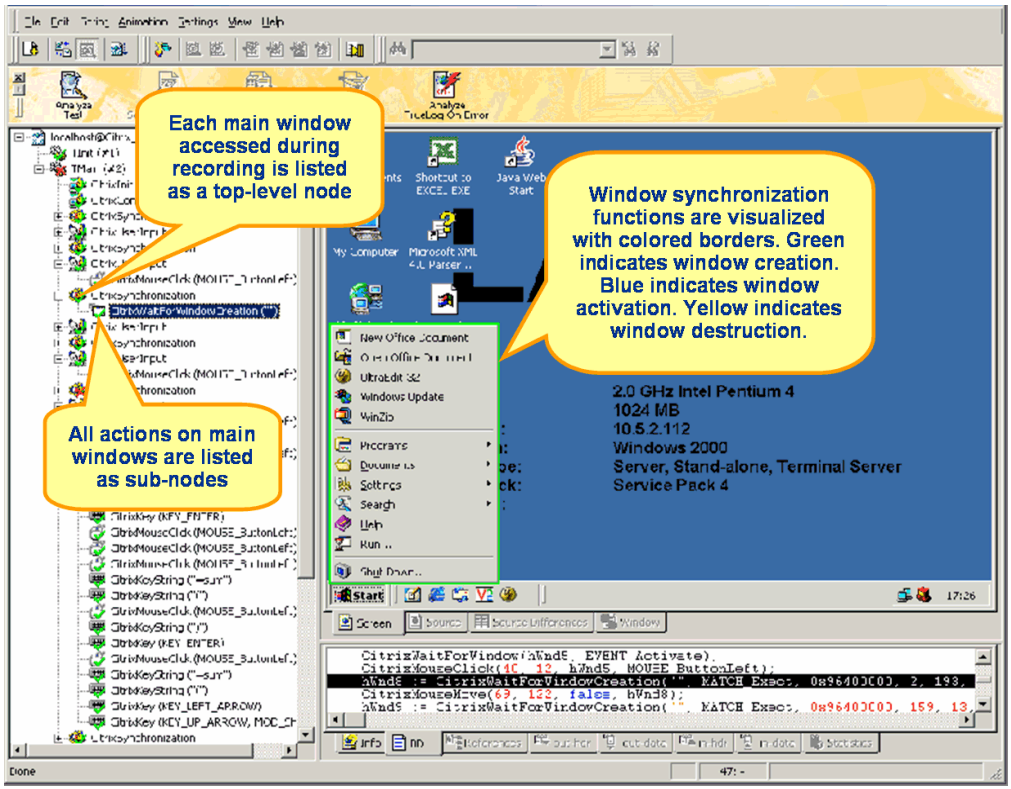
Along with the SilkPerformer Citrix Player, TrueLog Explorer may also be opened with TryScript runs (by selecting the *Animation* checkbox on the *TryScript* dialog). TrueLog Explorer displays the data that is actually downloaded during TryScript runs.

By selecting a high-level synchronization node you see a window as it appeared after the last synchronization function (the bitmap captured during replay).

Window synchronization functions are visualized with colored borders. Window creations are indicated with green borders. Window activations are indicated with blue borders. Window destructions are indicated with yellow borders.

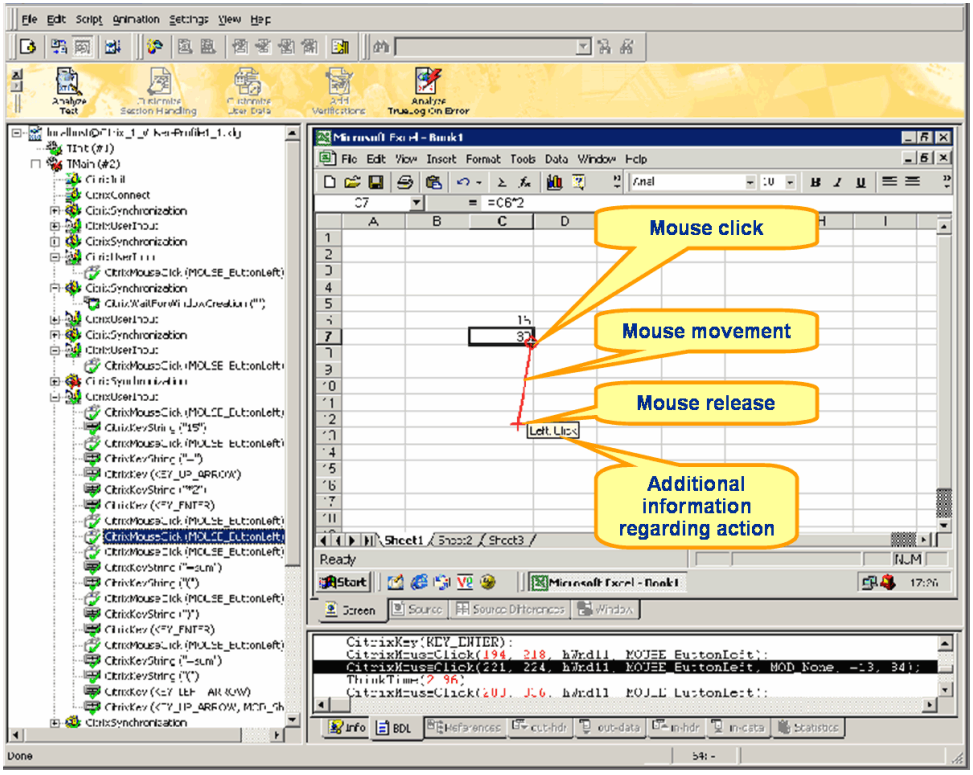
TrueLogs work in complement with the SilkPerformer Citrix Player by visualizing screen states. For example, if you aren't sure which window is indicated by a certain window ID that's listed in the SilkPerformer Citrix Player's *Log* window, you can find the corresponding synchronization function

in the corresponding TrueLog and thereby access a bitmap that shows the window in question.

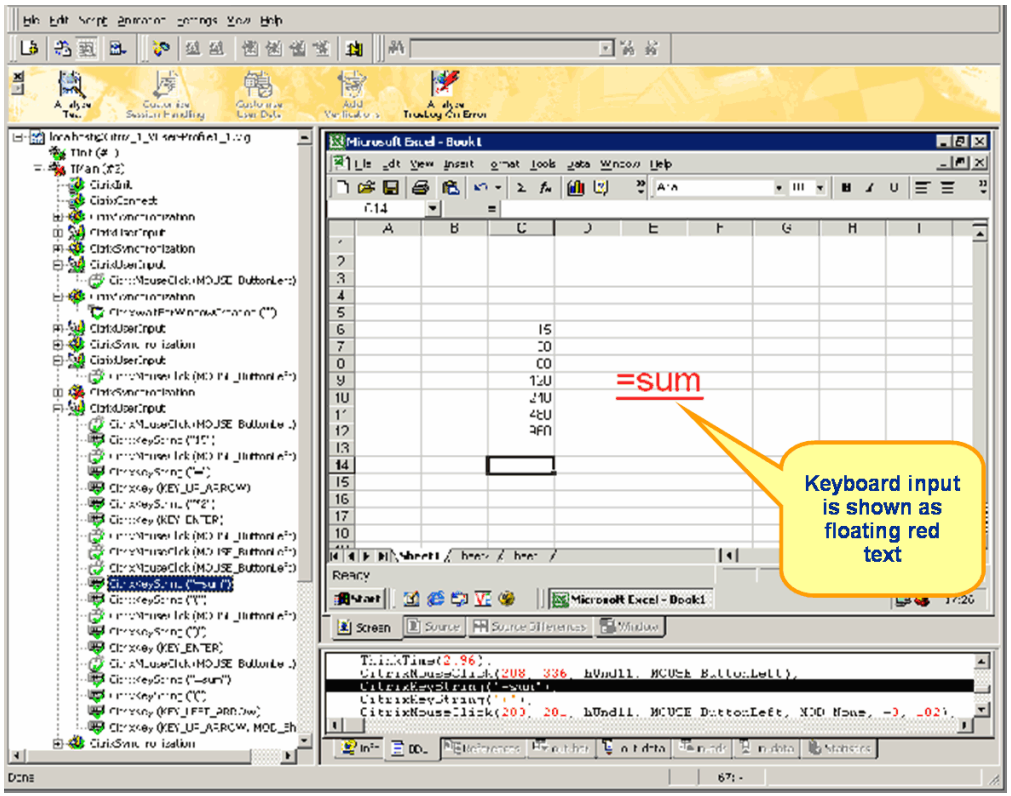


User input nodes *(CitrixUserInput* and related functions) reflect keyboard and mouse input. *CitrixMouseClick* functions offer two track vector parameters (X and Y coordinates). Red squares indicate mouse-click start points. Red cross-marks indicate mouse release points. A red line between a start and end point indicates the path of the mouse. If there is no move while the button is pressed

then only a red cross is displayed. Onscreen tool tips offer additional information (right-click, left-click, double-click, etc).



Value strings (keyboard input) are visualized onscreen as floating red text until target window captions are identified (in subsequent nodes) to indicate where strings are to be input.



Debugging scripts

Windows may fail to be activated and screen synchronizations may fail when the SilkPerformer Citrix replay encounters different values during replay than were captured during recording. Sometimes the causes of synchronization problems are not apparent — they may be due to a change in screen position of only a single pixel.

More common than screen synchronization failures are windows not being activated during replay. In such cases the screenshots associated with the corresponding user actions may explain the fault (sometimes there is no user fault and a window is activated only sporadically. In such cases you must remove the associated *CitrixWaitForWindow* function).

SilkPerformer Citrix replay captures screengrabs when errors occur (the default setting) and writes these bitmaps to disk. By default the recorder writes screenshots to the screenshots directory in the project directory. Replay stores screenshots in the current used result directory. Visual comparison of record and replay screens can be done best by using bitmap viewing programs.

Note SilkPerformer's *Dump window region of unsuccessful screen synchronizations* citrix option must be activated (the default) to have these bitmaps captured and saved.

See the *Silk TrueLog Explorer User Guide* for more details. See "Citrix Project & System Settings" for information regarding Citrix settings.

3

Customizing User Data

Introduction

This tutorial explains how to customize test scripts with parameterized user input data.

What you will learn

This chapter contains the following sections:

Section	Page
Overview	45
Customizing User Data	45
Synchronizing Text	52

Overview

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.

Customizing User Data

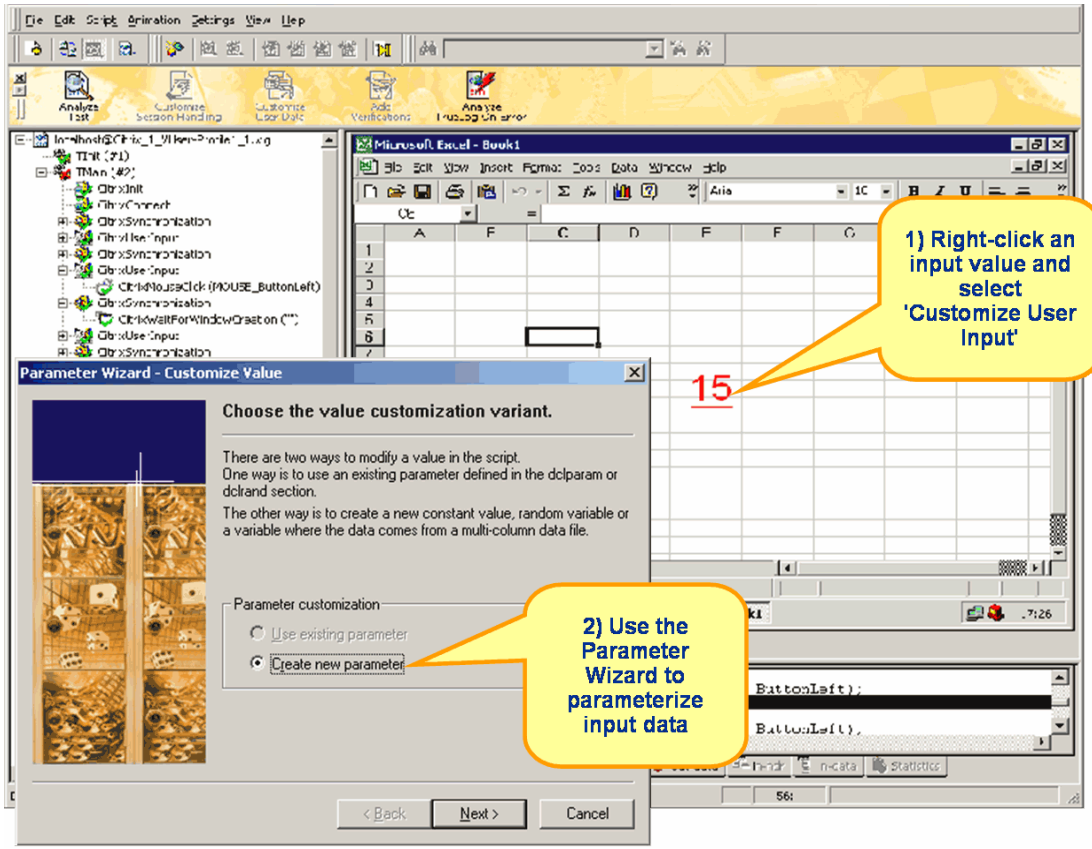
During testing you can customize the user input that's entered into applications that are hosted by Citrix terminal services in two ways:

- The *Parameter Wizard* allows you to specify values to be entered with keyboard events—enabling your test scripts to be more realistic by replacing recorded user input data with randomized, parameterized user data.
- *Visual Customization* allows you to customize mouse events such as clicks, drags, and releases.

Parameter Wizard

Procedure To use the Parameter Wizard to customize user data input:

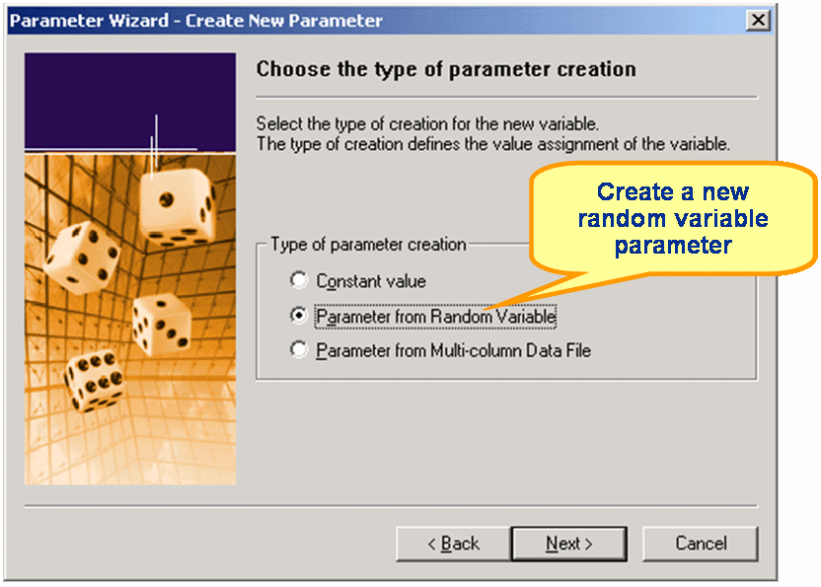
1. Select a node in the tree list view that reflects user data input (e.g., select a *CitrixKeyString* node that specifies a keyboard datastring).
2. Right-click the input datastring (shown as floating red text) and select *Customize User Input* from the context menu.
3. The Parameter Wizard opens. Select *Create new parameter* and click *Next*.



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.

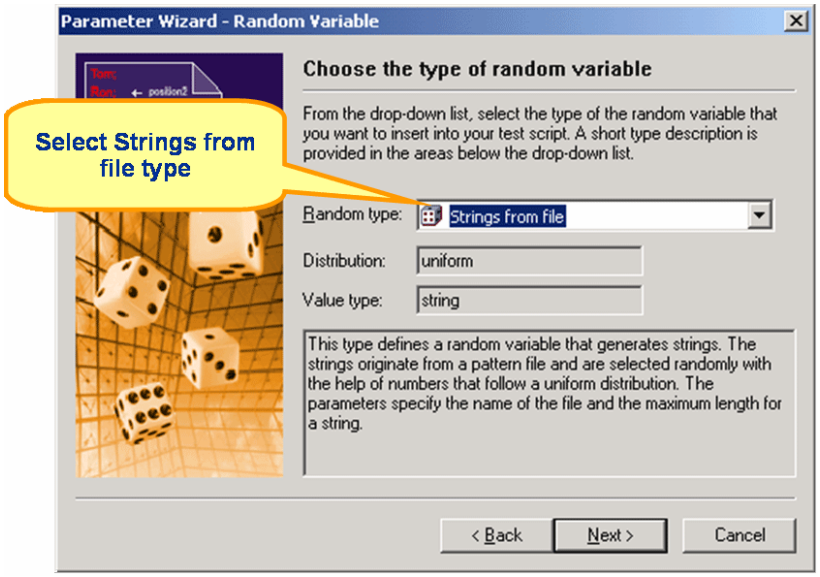
Note This tutorial explains only the process of creating a parameter based on a new random variable. See SilkPerformer documentation for complete details regarding the functionality of the Parameter Wizard.

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



6 The *Random Variable Wizard* appears. From the drop-down list, select the type of random variable (e.g., *Strings from file)* you wish to insert into your test script. A brief description of the highlighted variable type appears in the lower window.

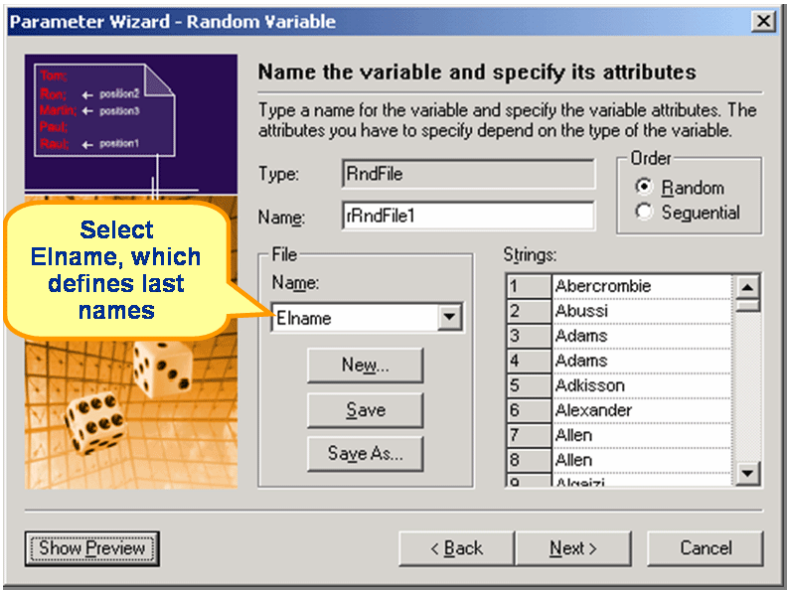
7 Click *Next*.



8 The *Name the variable and specify its attributes* screen appears. The *Strings from file* random variable type generates data strings that can either be selected randomly or sequentially from a specified file.

Enter a name for the variable in the *Name* field. Specify whether the

values should be called in *Random* or *Sequential* order. Then select a preconfigured datasource (e.g., *Elname* which defines last names) from the *File/Name* drop-down list.



Alternative New random variable files can be created by clicking the *New* button.

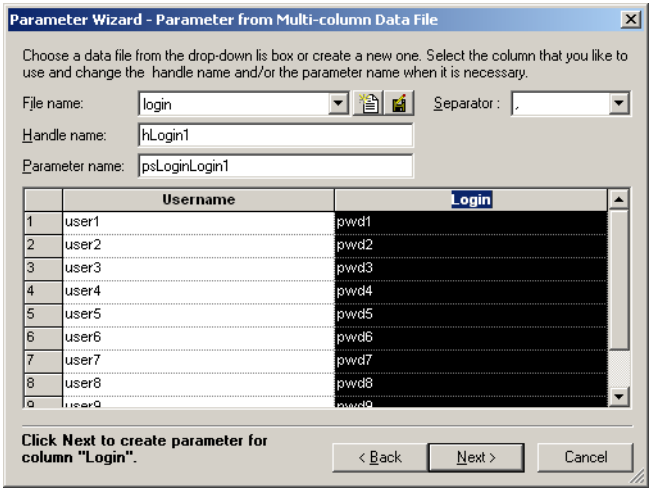
9 The *Choose the kind of usage* dialog appears. Specify whether the new random value should be used *Per usage, Per transaction,* or *Per test.*

10 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.

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

Multi-column data files

Parameterization from multi-column data files is a powerful means of parameterizing data because it defines files in which specific combinations of string values are stored (e.g., usernames/passwords, first names/last names, etc). Each column in a data file corresponds to a specific parameter. Multi-column data files enable a data driven test model and allow you to cover all user data input with a single data file.



Note See SilkPerformer documentation for more information regarding multi-column data files.

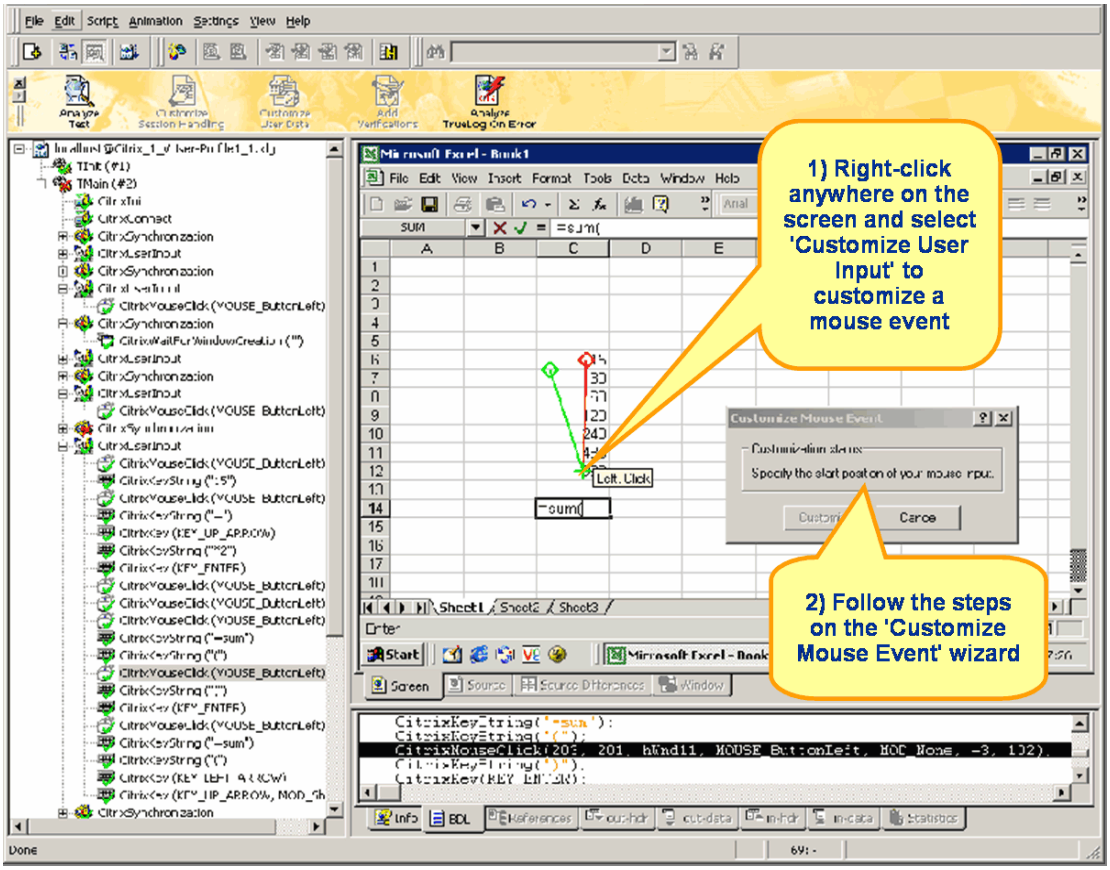
Customizing mouse events

The behavior of recorded mouse events can be visually customized.

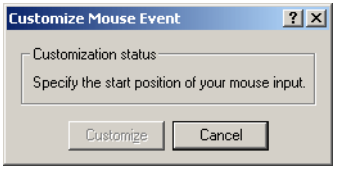
Procedure To customize replay of a recorded mouse event:

1 In the tree view, select a *CitrixMouseClick* node that includes mouse activity. Red squares indicate mouse-click start points. Red cross-marks indicate mouse-release points. A red line between a start and end point indicates the path of the mouse. Onscreen tooltips offer additional information (right-click, left-click, double-click, etc).

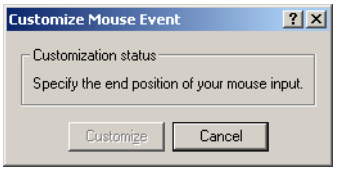
2 Click anywhere on the screen and select *Customize User Input* from the context menu. The *Customize Mouse Event* dialog appears.



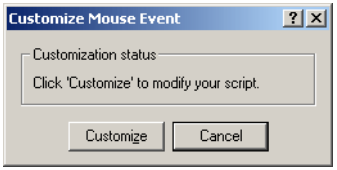
3 Click at the screen position where you want the customized mouse move to begin.



4 Click at the screen position where you want the customized mouse move to end.



5 Click the *Customize* button to accept the customization and modify the BDL script accordingly.



Your mouse event customization now appears in the recorded TrueLog bitmaps in green. The mouse customization also appears in the BDL script in green text. *CitrixMouseClick* functions offer two track vector parameters (X and Y coordinates). The next time this script executes, it will use the new screen coordinates you've specified.

Adding verifications

SilkPerformer supports bitmap and window verification for applications that are hosted by Citrix MetaFrame servers. Value verification and response data verification are not supported for Citrix. See "Screen Synchronization & Verification" for details.

Synchronizing Text

TrueLog Explorer offers a synchronization function that pauses the execution of Citrix functions until specified text or a text pattern appears in a specified location. Such synchronization is vital for the accurate execution of verification functions.

See *Silk TrueLog Explorer User Guide* for more information on synchronizing text.

4

Testing NFuse Sessions

Introduction

This chapter explains how to load test applications that are accessed via Citrix NFuse sessions.

What you will learn

This chapter contains the following sections:

Section	Page
Overview	53
Creating a Load Test Script	53

Overview

NFuse Classic application portal software provides Web access to Java, UNIX, and Windows applications that are hosted via MetaFrame application server software. While MetaFrame offers server-side control of hosted applications, NFuse makes applications accessible through a Web browser interface (MS Internet Explorer, version 5.5. or higher).

> **Note** For technical support and questions regarding Citrix MetaFrame NFuse, go to http://support.citrix.com

Creating a Load Test Script

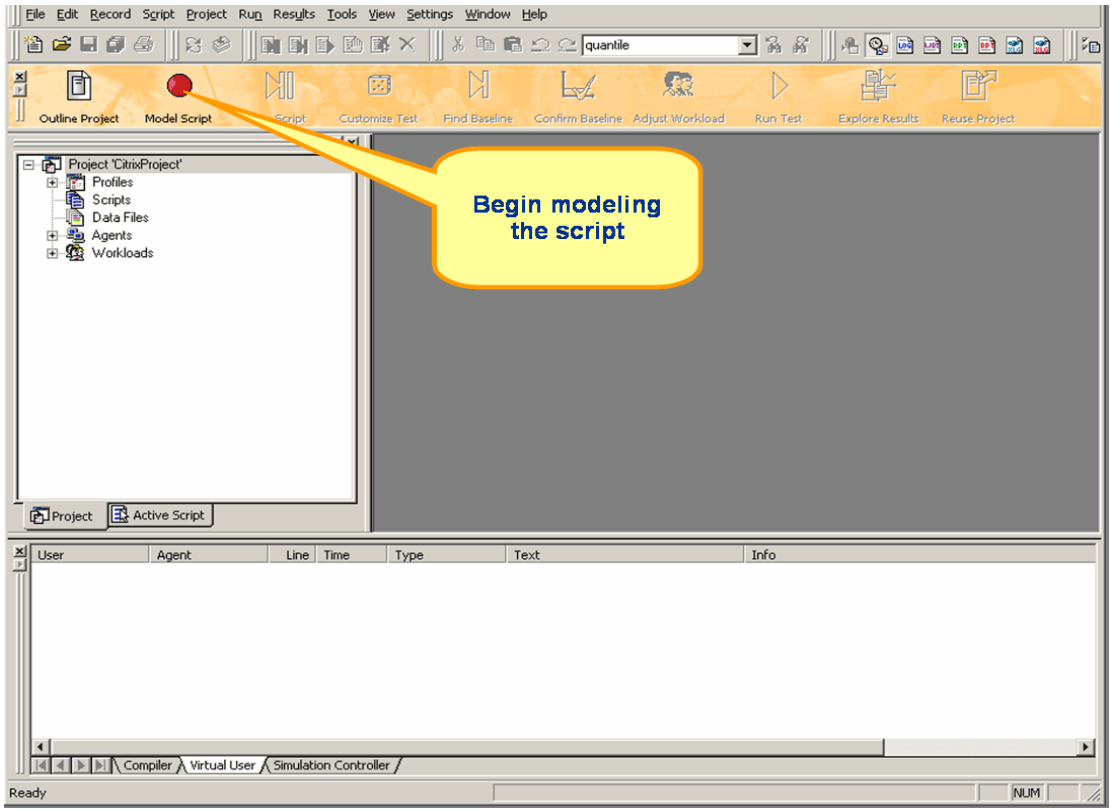
The easiest approach to creating a load test script for an NFuse session is to use the SilkPerformer Recorder, SilkPerformer's engine for capturing and recording traffic and generating test scripts.

The SilkPerformer Recorder captures and records traffic between an NFuse client application (MS Internet Explorer, version 5.5 or higher) and the server under test. When recording is complete, the SilkPerformer Recorder automatically generates a test script based on the recorded traffic. Scripts are

written in SilkPerformer's scripting language, *Benchmark Description Language (BDL)*.

Procedure To model a load test script for an NFuse session

1 Click the *Model Script* button on the SilkPerformer Workflow bar.

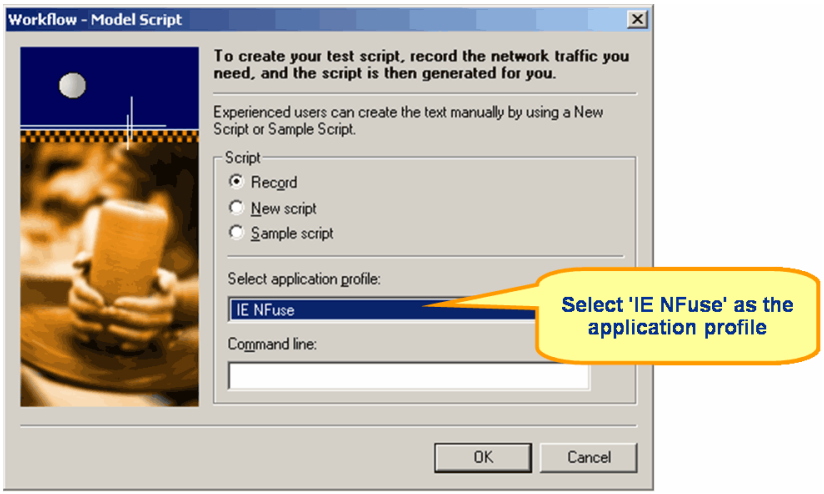


2 The *Model Script* dialog appears. Select *Record* in the *Script* area of the dialog.

3 From the *Select application profile* drop-down list, select *IE NFuse*.

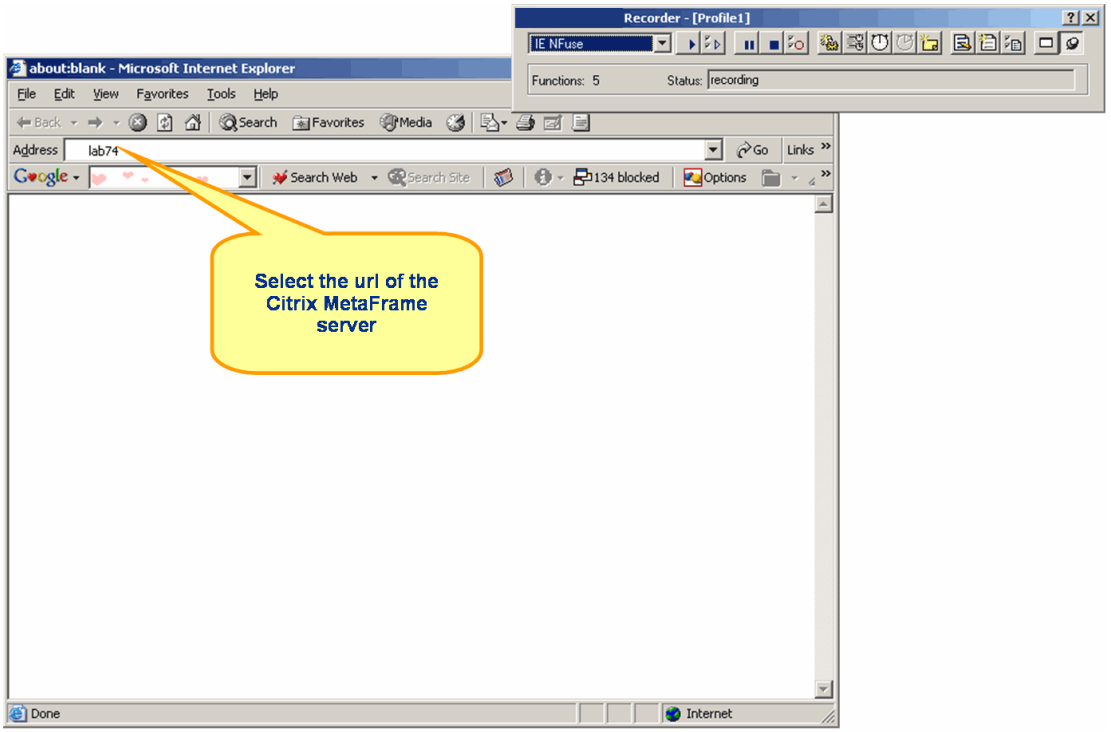
Note The *IE NFuse* application profile is appropriate for testing Citrix NFuse sessions only (for standard Citrix MetaFrame client recording, see "Creating Test Scripts").

4 Click *OK*.



5 The SilkPerformer Recorder then opens in minimized form along with Internet Explorer. Enter the name of the Citrix MetaFrame server in Internet Explorer's *Address* field and click *Enter*.

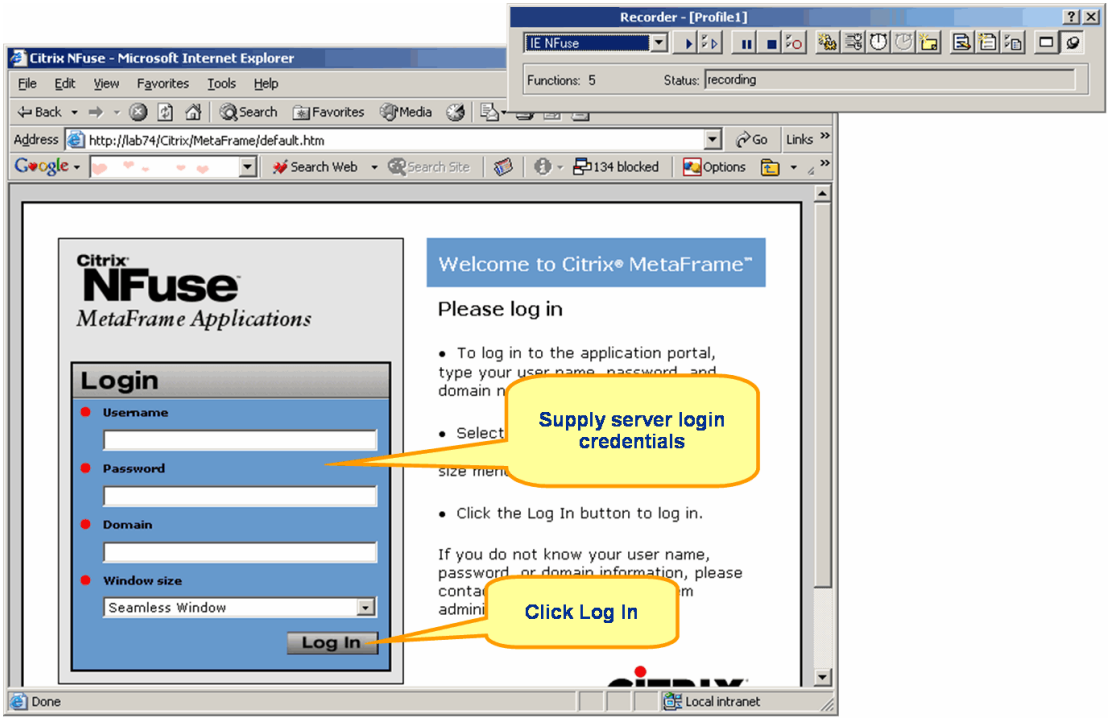
Note To see a report of the actions that occur during recording, maximize the SilkPerformer Recorder dialog by clicking the *Change GUI size* button.



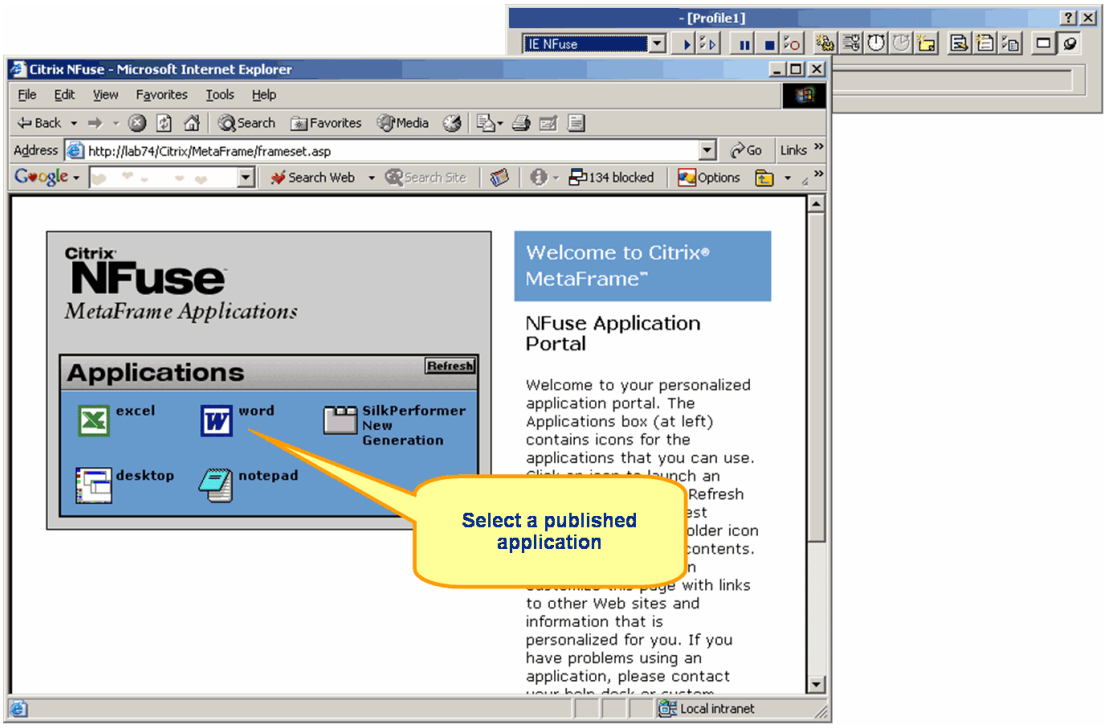
6 To log into the MetaFrame Application Portal, enter your *Username, Password,* and *Domain* into the NFuse login screen. Contact your system administrator if you do not have this information.

7 Specify how you wish to have application windows sized via the *Window size* drop-down list, seamless windows (i.e., application sharing) or entire desktop. If you specify seamless windows, then the size specified on the *Connect* dialog will be used—so it's recommended that you specify a size.

8 Click *Log In.*



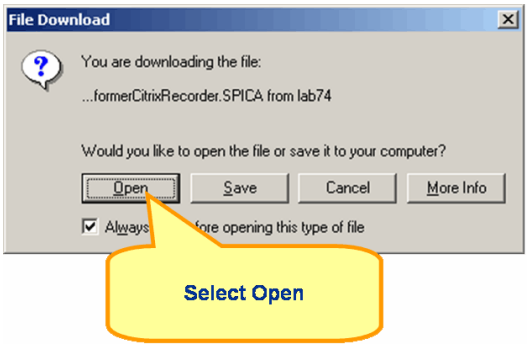
9 The NFuse Application Portal appears. This portal contains the applications that have been published for shared use. Select the hosted application you wish to record.



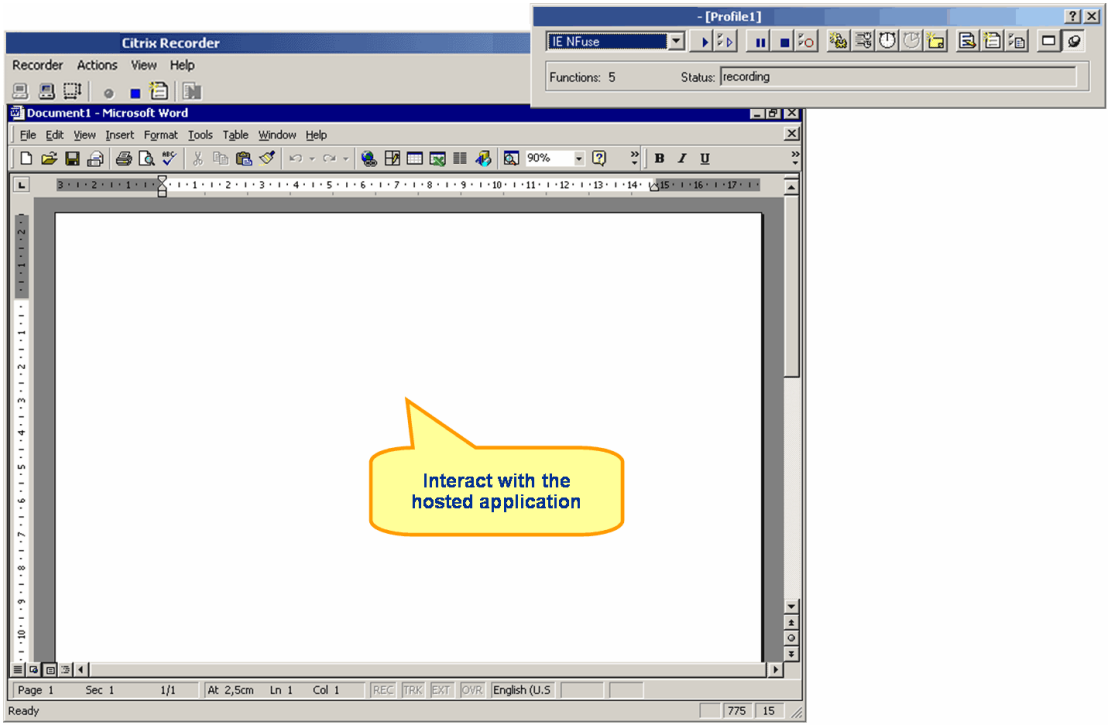
Note The above procedure for opening an NFuse hosted application can be managed automatically by entering the name of the hosted application into the *Application* field of the SilkPerformer Citrix Recorder's *Connect* dialog. The *Connect* dialog is accessible via the *Connect* button in the upper left corner of the SilkPerformer Citrix Recorder. Note however that if you take this approach front-end Web traffic will not be recorded with your test.

10 A dialog appears, asking you what you wish to do with the hosted application's *.spica* file (when the SilkPerformer Recorder intercepts *.ica* files it converts them to *.spica* format). *SPICA* files are SilkPerformer

formatted Citrix configuration files that include hosted application parameters such as host name, encryption level, and protocols. Click *Open* to open the hosted application in SilkPerformer's Citrix Recorder.



11 The hosted application appears in the SilkPerformer Citrix Recorder (MS Word is shown in the example below). Interact with the shared application in the Citrix Recorder in the same way that you want your virtual users to operate during the load test. Your actions will be captured by the Citrix Recorder and generated into a BDL script.



12 When you close the application the Citrix session will disconnect and you can save your recorded script.

NFuse scripts

BDL scripts of recorded NFuse sessions are multi-protocol scripts that include a small number of SilkPerformer Web functions.
The contents of *.spica* files are parsed via *WebParseDataBound* functions. These configuration settings are then referenced as a parameter to *CitrixConnectIcaData* functions. See SilkPerformer Online Help for complete details regarding available Citrix and Web functions.

5

Citrix Project & System Settings

Introduction

This chapter explains the Citrix profile settings that are available with SilkPerformer.

What you will learn

This chapter contains the following sections:

Section	Page
Overview	61
General Citrix Settings	62
Citrix Simulation Settings	63
Citrix Client Settings	65
Citrix System Settings for OCR	66

Overview

Citrix profile settings are project-specific settings related to Citrix synchronization, logging, virtual user simulation, and client options. Citrix settings are specified on a per-project basis.

See *SilkPerformer Online Help* for complete details regarding the use of project profiles and all available profile settings.

> **Note** This chapter focuses on Citrix replay settings. Citrix record options are limited to network protocol, encryption level, the *Log screen before each user action* setting, and the *Use RAM disk* setting.

General Citrix Settings

Procedure To set general Citrix options:

1. In the *Project* tab of the tree-view area of the main *SilkPerformer* window, right-click the *Profiles* node and select *Edit Active Profile.*

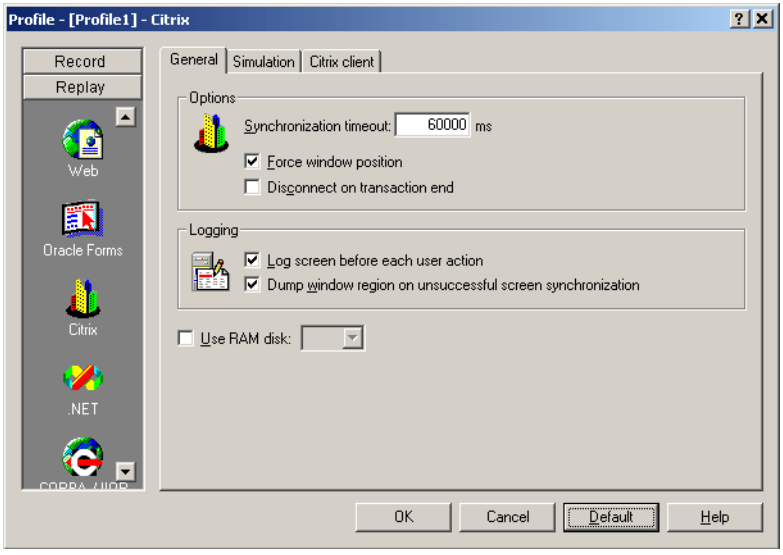
   Alternative Select *Settings/Active Profile* from the SilkPerformer menu bar.

2. The *Profile - [Profile1] - Simulation* dialog opens at the *Simulation* tab *(Replay* category).
3. Scroll down to and select the *Citrix* icon. The Citrix *General* settings tab opens.
4. In the *Options* section of the *General* tab, specify a timeout value in the *Synchronization timeout* field. You may have to increase this value if your Citrix server is slow.
5. The *Force window position* option (enabled by default) automatically moves replayed windows to the coordinates specified in *CitrixWaitForWindowCreation* functions.
6. The *Disconnect on transaction end* option (disabled by default) disconnects the client after each transaction—even init transactions.
7. In the *Logging* section of the *General* tab, the *Log screen before each user action* option (enabled by default) enables onscreen display of user input (datastring values) for the moment before values are actually input into screens (e.g., a user might enter a string value into a spreadsheet cell. The value will not actually be input until the user clicks the *Enter* key. With this option enabled, in the node just preceding the click of the *Enter* key, the string value will appear onscreen as floating red text). This option requires significant processing and disk storage as it dictates that each user action generate a screenshot. With this option disabled you will not see all user input updates.
8. The *Dump window region on unsuccessful screen synchronization* option specifies that screengrabs be generated for all unsuccessful screen synchronizations. These bitmaps, when captured and saved to the current result directory (e.g., *RecentTryScript)* of your SilkPerformer installation, can be compared to corresponding recorded synchronizations to assist in debugging efforts. For example, a difference of a single pixel is enough to cause a screen synchronization error. Such

a difference might best be detectable visually, by comparing recorded screengrabs with screengrabs captured when synchronization errors occur.

9 The grabbing, reading, compressing, and writing of screengrabs for TrueLogs involves significant processing resources and can lead to slow replay. The *Use RAM* disc option (disabled by default) enables faster TrueLog replay via the use of a RAM disk, rather than naming files and writing them to hard disk. Use the drop-down list to select the letter of your RAM disk. Note that this option does not install a RAM disk, it only allows you to specify a previously installed RAM disk.

10 Click *OK* to save your changes, or click *Default* to restore the default settings.



Citrix Simulation Settings

Citrix simulation settings enable you to configure the behavior of Citrix virtual users, including keyboard/mouse input and think time.

Procedure To set Citrix simulation options:

1 In the *Project* tab of the tree-view area of the main SilkPerformer window, right-click the *Profiles* node and select *Edit Active Profile.*

Alternative Select *Settings/Active Profile* from the SilkPerformer menu bar.

2 The *Profile - [Profile1] - Simulation* dialog opens at the *Simulation* tab *(Replay* category).

3 Scroll down to and select the *Citrix* icon.

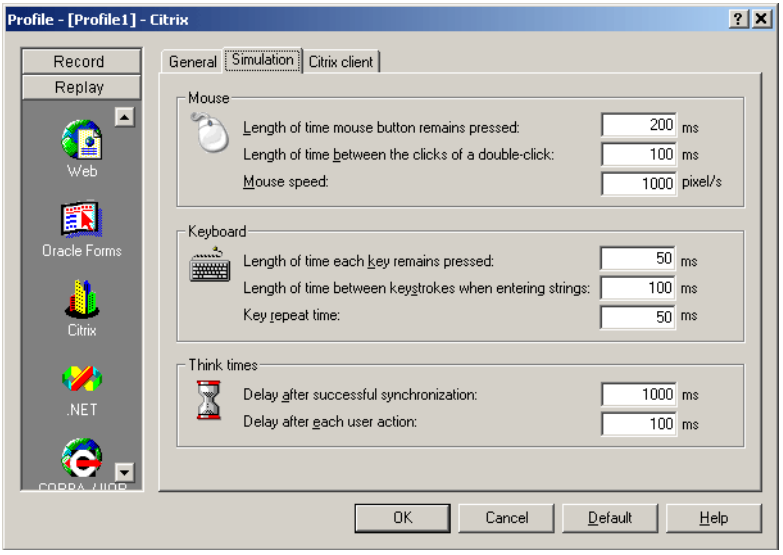
4 Select the *Simulation* tab.

5 In the *Mouse* section of the *Simulation* tab, specify virtual user mouse behavior (in milliseconds) such as the length of time that mouse clicks remain pressed, the length of time between the clicks of a double click, and mouse speed. Note that simulated mouse events move at constant speeds—they do not simply jump across the screen.

6 In the *Keyboard* section of the *Simulation* tab, specify virtual user keyboard behavior (in milliseconds) such as the length of time that keys remain pressed, the length of time between keystrokes when entering strings (i.e., *CitrixKeyString* functions), and key repeat time (i.e., repeat parameters of *CitrixKeyString* functions).

7 In the *Think times* section of the *Simulation* tab, specify virtual user think time behavior (in milliseconds) for delay after successful synchronizations, and delay after each user action. This is a virtual simulation of user reaction time that helps to stabilize replay.

8 Click *OK* to save your changes, or click *Default* to restore the default settings.



Citrix Client Settings

Citrix client settings enable you to configure client options such as network protocol, disk cache, data compression, the queuing of mouse movements, speed screen latency reduction, and encryption level.

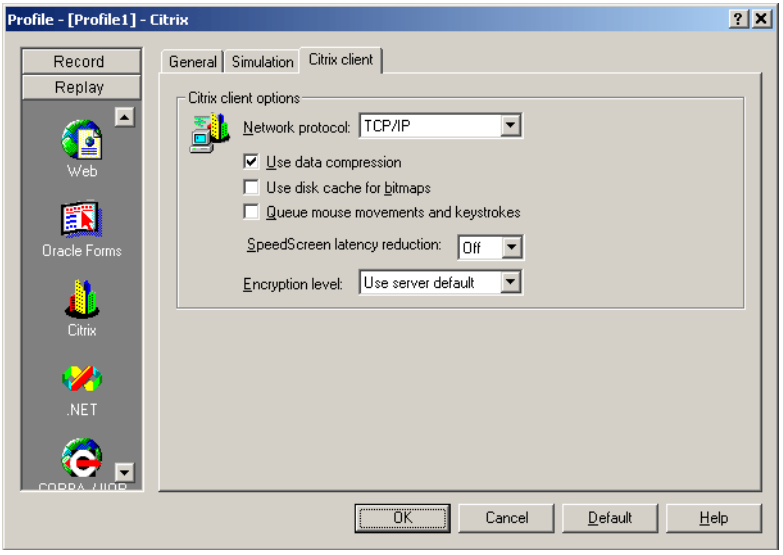
Procedure To set Citrix client options:

1. In the *Project* tab of the tree-view area of the main SilkPerformer window, right-click the *Profiles* node and select *Edit Active Profile.*

   Alternative Select *Settings/Active Profile* from the SilkPerformer menu bar.

2. The *Profile - [Profile1] - Simulation* dialog opens at the *Simulation* tab *(Replay* category).
3. Scroll down to and select the *Citrix* icon.
4. Select the *Citrix client* tab.
5. Select the network protocol upon which your client will run *(TCP/IP* or *TCP/IP + HTTP).* When you specify *TCP/IP + HTTP* load balancing is done with the HTTP protocol, using post commands. When you specify *TCP/IP,* UDP is used. No other network protocols are supported.
6. Select the *Use data compression* checkbox to enable data compression (enabled by default).
7. Select the *Use disk cache for bitmaps* checkbox to enable the caching of bitmaps on your hard disk (disabled by default).
8. Select the *Queue mouse movements and keystrokes* checkbox to queue mouse movements and keystrokes for a specified period of time before they are sent to the server (disabled by default).
9. *SpeedScreen latency reduction* enables local echo of mouse and keyboard actions (disabled by default). Local echo means that you don't have to wait for round trips to the server to see the results of your input. Specify *Off, On,* or *Auto.*
10. Specify an encryption level for the client. Options include, *Use server default, Basic, 128 Bit for Login Only, 40 Bit, 56 Bit,* and *128 Bit.* See SilkPerformer Online Help for details regarding these settings.

11 Click *OK* to save your changes, or click *Default* to restore default settings.



Citrix System Settings for OCR

To enable optical character recognition (OCR) for parsing and verification functions in TrueLog Explorer, you must generate a font database using SilkPerformer's system settings.

Optical character recognition relies on font, or "pattern," databases to recognize fonts and text styles in bitmaps. The default set of fonts covers most scenarios, however in some situations you may wish to add additional fonts or font styles. A new database should be generated whenever new fonts are added or removed from the system.

Including too many fonts in the database can slow down processing and lead to contradictory reading, so it's recommended that you only include those fonts that are used in the bitmaps from which you will be capturing text strings.

Procedure To generate a font database for Citrix optical character recognition:

1 Within SilkPerformer, go to *Settings/System.../* and select the *Citrix* icon.

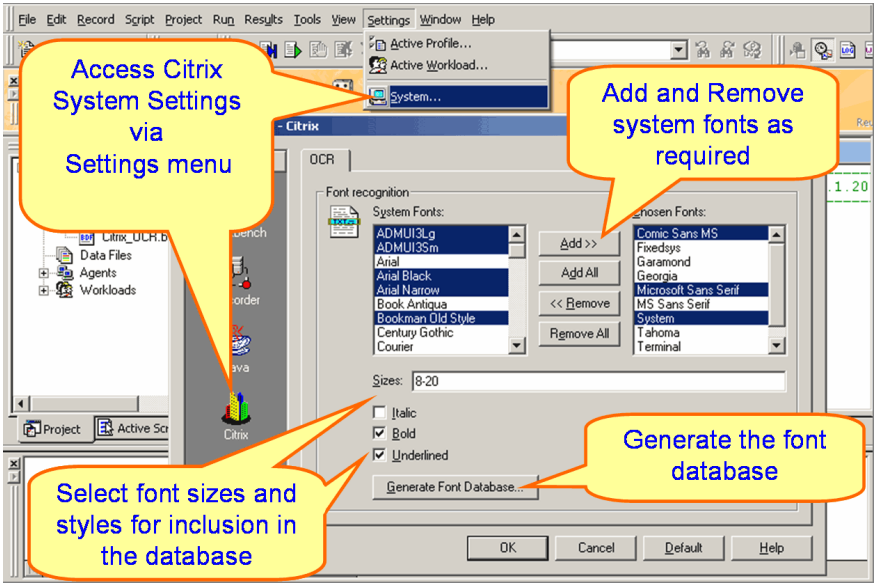
2 On the *OCR* tab, use the *Add >>* and *Add All* buttons to move those fonts that you wish to have used for OCR from the *System Fonts* list box to the *Chosen Fonts* list box.

3 Use the *Remove All* and << *Remove* buttons to delete unnecessary fonts from the *Chosen Fonts* list box.

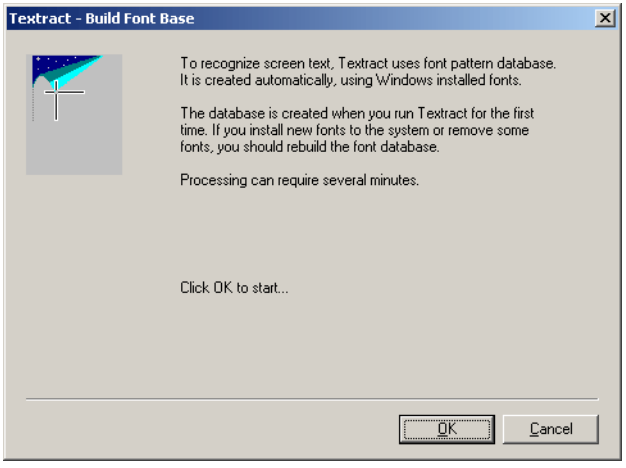
4 In the *Sizes* text field, specify the font size range that should be used (e.g., 8-20).

5 Define which font styles should be included by selecting the *Italic, Bold,* and *Underlined* checkboxes.

6 Click the *Generate Font Database* button.



7 The *Build Font Base* dialog appears. Click *OK* to confirm that you wish to replace the existing font database with a new database.



8 Click *OK* on SilkPerformer's *System Settings* dialog to accept the changes.

6

Best Practices

Introduction

This chapter offers suggestions for successfully load testing GUI applications via Citrix MetaFrame terminal services.

What you will learn

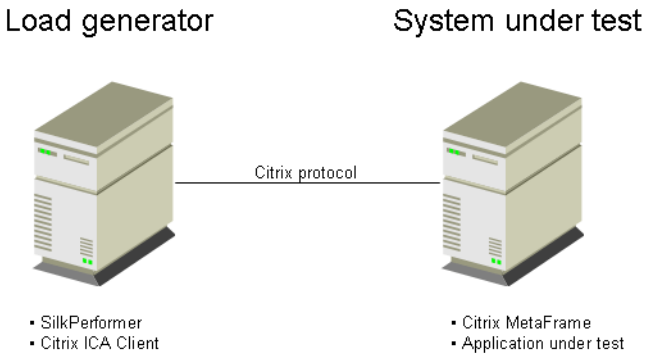
This chapter contains the following sections:

Section	Page
Overview	69
Test Preparation	70
Recording Use Cases	74
Troubleshooting Scripts	78
Issues Specific to Citrix MetaFrame	80

Overview

While GUI-based load testing presents a number of challenges, it can be rewarding as it offers the opportunity to closely simulate a real user experience. GUI-based load testing via SilkPerformer involves applying load to applications by simulating the terminal services protocol.

At least two physical computers are required to perform a load test via Citrix MetaFrame: one computer runs the terminal services environment and the application under test (AUT); the other computer runs SilkPerformer and the Citrix ICA client software.



The *load generator* computer simulates a large number of users by mass producing the terminal services network protocol. The terminal services server (system under test) simulates multiple MS Windows-based desktop sessions simultaneously. This server also hosts the GUI application that is placed under load.

It is recommended that you place an isolated network (LAN) between the load generator and the system under test. This allows for network throughput analysis. An isolated network is also less vulnerable to external influence, thereby reducing errors and misleading network throughput results.

Test Preparation

Before you begin conducting GUI-based load tests, it is recommended that you adhere to standard test preparations to avoid common problems. Suggestions for your preparation are listed below:

1 *User Interface (UI) design*

Take advantage of any opportunity to influence the UI design of the application. A solid, consistent UI allows you to create reusable sub-routines. In turn, you can write less test code—and consequently have less code to maintain.

2 *Test plan*

A thoughtful test plan is a prerequisite of successful GUI-based load testing. In your test plan you should define your goals, objectives, test runs, critical metrics, etc.

3 *Use cases*

Use cases are step-by-step scenarios that you plan to test. Use cases represent typical user-to-application interaction (e.g., starting an application, opening a project, editing settings, etc). Your use cases guide you through the recording process. See "Creating use cases" for more details.

4 *Application expert*

Even when equipped with well documented use cases you may not be able to make sense of all of an application's workflow. It is helpful to have access to someone who has expert knowledge of the application under test (e.g., a business process owner).

5 *Screen resolution*

The higher the screen resolution, the higher the system requirements are per terminal session. It is recommended that you use a screen resolution of no more than 800x600 for record and replay. This approach has the added benefit of allowing for compatibility with older computers (which you then have the opportunity of including in your load tests).

6 *Remote users*

If your application is to be available to remote users, consider simulating a slow network during some of your load tests. Slow network connections can have a major impact on the performance of applications, therefore additional tests with simulated network congestion help improve test results. A good WAN emulation software is developed by Shunra (http://www.shunra.com).

Defining your goals

The first thing to do when planning your load test is to define your goals. While testers often assume that their goals are clear to all involved parties, they often are not. It is important that the quality assurance team and management work toward the same goals.

When writing up goals, first consider your highest-level goals:

1 Discover the baseline performance of your application

2 Optimize/tune your application

3 Determine if the application under test is ready for production

Once you have established your high-level goals, you need to break them down into clear, measurable objectives—covering issues such as how you plan to accomplish your goals, how you plan to measure your goals, and what results will be considered acceptable.

Here is an example of how you might break down your testing goals:

- Goal #1: *Discover and document the baseline performance of your application*
  - Use SilkPerformer to measure the response times of critical application functions.
  - Place timer functions around window/screen synchronizations (*WaitFor* events). These measurements will show up in the final report.

    Critical timers:

    - Measure how much time it takes from when the *OK* button is clicked on the *Login* window until the *Welcome* window appears.

    - Measure how much time it takes from when the *Query results* button is clicked until the populated list is displayed.
- Goal #2: *Optimize/tune the application*
  - Optimize/tune the application for 5 days.
  - Document all changes and their impact on the performance of the application under test.
- Goal #3: *Determine if the application under test is ready for production*
  - The application is ready for production if the following condition is met: All response times are under 5 seconds (SilkPerformer provides metrics for each window and screen sync).

With a list of measurable objectives, your load test results define a definite point at which the application will be ready for production. This also eliminates the risk of endlessly optimizing the application as tuning it to the defined goal is adequate.

Creating use cases

A use case is a typical procedure that a user undertakes when working with the application under test. A use case must use the features of the application that require testing. It is essential that you only test features that are important and working properly. This is not the time to perform functional testing, which should already have been completed. Testing is a long process: the longer the use cases, the more time that will be required for testing.

When stepping through a use case, write down all significant screen events. For example, when entering a formula in Microsoft Excel you should document the changing of cells due to formula processing. Such events will translate into screen synchronizations that will be important during script development. Text synchronizations can be used for text-based screen synchronizations (see “Text synchronization” for detailed information).

It is important that use-case documentation be highly detailed. You need to document each mouse click, key stroke, and expected result. While this may be tedious initially, it makes things easier later when you begin recording your test cases.

The following example—a test that simply locates an existing instance of Microsoft® Word and opens a document—displays the level of detail that a use case should have. The square brackets ("[]") indicate an event.

```
In the "Microsoft Word" window, navigate to the File
menu and select Open….
[The "Open" dialog appears]
Select 'Test.doc.'
Click Open.
[The "Open" dialog closes]
[The "Microsoft Word" window has focus once again]
```

Note that the exact titles of the windows are documented in the above example. This becomes important later during script development as such information is needed to keep the script in sync with the application. Once you have a well documented use case, it is easy to script the application

There are a number of ways to write use cases. You can use XML notation, numbered notation, or another format that you are familiar with.

XML use case sample

```
<Task Name="Open a document">
  In the "Microsoft Word" window, navigate to the
File menu    and select Open….
  [The "Open" dialog appears]
  Select 'Test.doc.'
  Click Open.
  [The "Open" dialog closes]
  [The "Microsoft Word" window has focus once again]
</Task>
```

Numbered use case sample

```
100.01 – Open a document
  In the "Microsoft Word" window, navigate to the
File menu    and select Open….
  [The "Open" dialog appears]
  Select 'Test.doc.'
  Click Open.
  [The "Open" dialog closes]
  [The "Microsoft Word" window has focus once again]
100.02
    …
```

Stable environment

Before you begin recording, make sure that there will be no additional changes to the application's GUI (at least until after your load tests are complete). This requires solid communication with the development team. If the GUI changes after you record your test scripts, your scripts will likely become obsolete.

Also, make sure that you have the ability to back up and restore any databases that are used in your testing. Because changes to database content often lead to differences in GUI content (e.g., more/different items in lists, different query results), databases need to be returned to a baseline state before the start of each load test cycle.

Recording Use Cases

Once you have documented your use cases, recording or coding your load test scripts should be a relatively simple task. With SilkPerformer you have both the record/playback option and the option to code your test scripts manually. Each approach has its advantages and disadvantages. The most efficient way to create test scripts with SilkPerformer is to combine the two approaches, as follows:

1. With SilkPerformer, record the scenario you have outlined in your use case description.
2. Use SilkPerformer's TrueLog Explorer to visually customize your script (this is where the "Text synchronization" feature comes into play).
3. For more complex scripting, edit the generated BDL script manually.

Manually editing a script is also the best option for inserting the content of your use case into your test script (in the form of comments). See the example below, which uses numbered notation format:

```
100.001 In the "Microsoft Word" window, navigate the
menu to File, Open….
100.002 [The "Open" dialog window shows]
100.003 Select Test.doc.
100.004 Click Open.
100.005 [The "Open" dialog window goes away]
100.006[The "Microsoft Word" window has focus again]
```

You can now copy/paste these contents into your BDL script as follows:.

```
// 100.001 In "Microsoft Word" window,
    hwndWordMainWindow := CitrixSearchWindow("*Microsoft
    Word", MATCH_Wildcard);
    CitrixWindowBringToTop(hwndWordMainWindow);

// navigate the menu to File, Open….
    CitrixKey(KEY_Alt);
    CitrixKeyString("f");
    CitrixKeyString("o");

// 100.002 [The "Open" dialog window shows]
    hwndOpenDialog := CitrixWaitForWindowCreation("Open",
    Match_Exact);

// 100.003 Select Test.doc.
    CitrixMouseClick(150, 100, hwndOpenDialog, MOUSE_
    ButtonLeft);

// 100.004 Click Open
```

```
        CitrixMouseClick(300, 200, hwndOpenDialog, MOUSE_
        ButtonLeft);

    // 100.005 [The "Open" dialog window goes away]
        CitrixWaitForWindow(hwndOpenDialog, EVENT_Destroy);

    // 100.006 [The "Microsoft Word" window has focus again]
        CitrixWaitForWindow(hwndWordMainWindow, EVENT_Activate);
```

If the script fails, these comments will help you determine where the script failed in reference to the use case.

Synchronization Options

Window synchronization

Window synchronizations such as *CitrixWaitForWindow()* and *CitrixWaitForWindowCreation()* are well suited to synchronizing with an application. It is important to synchronize with the application so that the script does not wait until the application is ready for additional user input. If you do not use synchronizations, the script will most likely overrun the application. Also, synchronization gives you point-in-time reference as to when tasks are completed—effectively measuring response times.

Text synchronization

Many tasks that can be performed on an application do not show, hide, create, or destroy windows—they simply change a section of the screen. In such instances you should use SilkPerformer's text synchronization functionality.

After recording your script, you can visually add text synchronization points via TrueLog Explorer's *Synchronize Text* option (see the *Silk TrueLog Explorer User Guide* for detailed information). Text synchronization works via built-in OCR (Optical Character Recognition).

Screen synchronization

SilkPerformer offers two types of screen synchronizations: *wait for content change* and *wait for content match*. This form of synchronization requires a window reference and two sets of x/y coordinates. However since unplanned events can lead to displaced window coordinates, you should use text synchronization whenever possible, and only use screen synchronization when there is no text to compare.

Wait for content change waits until the selected portion of the screen changes, while *wait for content match* waits until the selected portion of the screen matches what was defined during recording.

> **Note** Screen synchronization points must be inserted while you record your test case, whereas text synchronization points can be inserted afterward, via TrueLog Explorer.

Recording Tips

When recording a terminal services script, you must be aware that unplanned events can change the position and size of target windows during replay, causing test runs to fail. Here are some general tips that will help you avoid the effort of finding and fixing such replay errors.

Launching the application

Try launching the application via the Windows *Run* command. Launching the application via a desktop icon or mouse click may also work, but consider if you will be moving the application to a different server, or if the GUI will vary between users, in which case a desktop icon might appear in a different location. It is therefore recommended that you type the path to the application's .exe file in the Windows *Run* command window.

Using keyboard shortcuts

Mouse clicks require x/y coordinates, a window reference, and the target window must have focus. While GUI load-testing technologies for tracking mouse clicks have become increasingly reliable, MS Windows remains a dynamically changing environment, so you must build scripts that can tolerate minor inconsistencies. One way to do this is to use keyboard shortcuts. For shortcuts, the only requirement is window focus.

Using the *Alt* key

Almost all menu systems in Windows applications have *Alt* key combinations that can be used to initiate specific menu commands. To record use of an *Alt* key shortcut, first give your application focus, then press the *Alt* key—this gives the menu system focus. Notice that many of the menu items have one of their letters underlined. This indicates which *Alt*-key commands are available for specific menu items. Press appropriate letter keys to gain access to sub-menus until you reach the menu item that you want to initiate. In some cases, a menu item itself will have a key combination. For example, to open a document in Microsoft Word use the *Ctrl+O* combination. This shortcut does not require that the menu have focus.

Maximizing windows

If keyboard shortcuts are not available and you must use the mouse, you can reduce the chance of triggering inaccurate mouse clicks by maximizing the window that you are working with. This places the window in a fixed position each time it is used and, in turn, makes mouse clicks more accurate. As a general rule, if a window allows for maximization, it is recommended that you take advantage of it.

> **Note** *Alt-Space*, *X* is the Windows keyboard combination for maximizing an active window

When a window does not allow for maximization, during the recording session, move the window to the upper left-hand corner of the desktop before clicking. When recording, the recorder may not be able to pick up on the window handle that you have in the foreground, so having the coordinates relative to the desktop is a must.

Keeping the system-under-test clean

To eliminate unexpected detours during terminal services sessions, ensure that you keep your Windows desktop as uncluttered as possible. Imagine, for example, that while your script is replaying, a Windows network message appears—your script will break because the subsequent mouse click will go to the Windows network message window rather than to your intended application window.

Ensuring correct focus

Before clicking a window, ensure that the window you intend to click exists and has focus. Here is a sample BDL function that helps automate this process:

```
function MyCitrixWaitForWindowCreationAndActivation(sCaption:string;
                nMatch     : number optional;
                nStyle     : number optional;
                nX         : number optional;
                nY         : number optional;
                nWidth     : number optional;
                nHeight    : number optional
                ) : number
var
    hwndNewWindow  : number;  // hwnd of the new window created
    nRTT:number;
begin
    //
    // Check if window exists
    //
        hwndNewWindow := CitrixSearchWindow(sCaption, nMatch);
        hwndNewWindow := 0;
    //
    // If window doesn't exist, wait for creation
    //
        if (hwndNewWindow < 1) then
            hwndNewWindow := CitrixWaitForWindowCreation(sCaption, nMatch,
            nStyle, nX, nY, nWidth, nHeight);
        end;
    //
    // Check if window is already active, wait if it's not active
    //
        MyCitrixWaitForWindowActivate(hwndNewWindow);
        MyCitrixWaitForWindowCreationAndActivation:=hwndNewWindow;

end MyCitrixWaitForWindowCreationAndActivation;
```

Using parameters

If you input the same information more than once, the system under test caches the data in memory rather than repeating the same work. For this reason it is important that you vary input/request data so that the work is as realistic as possible. For example, use different user accounts that have different data ranges. This can be accomplished using a CSV (comma separated values) file as input for a load test (see the *Parameter Wizard* section in SilkPerformer's *Online Help* for detailed information).

Adding verifications

Verifications are checks that are added to code to check if the responses that are received from the server are correct. When performing GUI based load testing, verifications are automatically added to your script via window synchronizations, however it is recommended that you add additional verifications to your code.

Adding timers

Custom timers are critically important for tracking an application's response times. Without custom timers, you cannot determine your end user's overall experience with an application. Metrics can help determine which aspects of your application are performing well and which are not.

Before you begin adding custom timers to your test script, identify the critical areas of your application and determine where your *MeasureStart* and *MeasureStop* calls should be placed. For this effort it is good practice to review the log in TrueLog Explorer.

Here is an example of using timers in a SilkPerformer script:

```
//
// Submit a work order.
//
CitrixMouseClick(27, 31, hwndWorkOrder, MOUSE_ButtonLeft);

//
// Start the response time clock.
//
MeasureStart("202.01: Work Order Submission.");

//
// Wait for the "Order Submission Complete" dialog box.
//
MyCitrixWaitForWindowCreationAndActivation(
   "Order Submission Complete",
   MATCH_Exact
   );

//
// Stop the response time clock.
//
MeasureStop("202.01: Work Order Submission ");
```

Troubleshooting Scripts

You may encounter timeout errors or other execution failures during load test execution. Here are some tips for avoiding, finding, and fixing such errors.

Using TrueLog Explorer

Test runs often fail due to the appearance of unexpected dialogs, causing scripts to lose focus on the windows they are working on. It is therefore recommended that you enable the *TrueLog On Error* feature in SilkPerformer before you

execute load tests. Then, if an error occurs, you will be able to visually track it down in TrueLog Explorer.

Clearing terminal server sessions

After a failed test execution, ensure that you reset all terminal server sessions before you execute the load test again, otherwise your script will likely fail.

Handling application errors

During load tests, your application is likely to throw errors due to generated load. Adding event handlers to your script that can handle and report such errors is very helpful.

Here is an example of an event handler that continuously watches for a window with the name *Program Error Intercepted* and executes an *ALT-C* to close any such window that appears. The event handler also generates an error message whenever such an error occurs.

```
dclevent
  handler Handler1 <EVENT_CITRIXINTERRUPT>
  var
    nInterrupt, nWindow : number;
    nStyle              : number;
    sWindowCaption      : string;
    begin
      CitrixGetActInterrupt(nInterrupt, nWindow);
      ErrorAdd(FACILITY_CITRIXENGINE, 47, SEVERITY_
INFORMATIONAL);
      print(string(nWindow));
      CitrixGetWindowCaption(nWindow, sWindowCaption);
      if sWindowCaption = "Program Error Intercepted" then
        CitrixKey(67, MOD_Alt); // 'c'
      end;
      ErrorRemove(FACILITY_CITRIXENGINE, 47);
    end Handler1;
```

Avoiding think times

While it may be tempting to use *Wait()* or *ThinkTime* statements to avoid having scripts overrun applications under test, this practice is not recommended for two reasons:

- When load generation increases, application processing speed may slow considerably. The wait statement may eventually be too short and the problem of overrunning the application will again present itself.
- If you are measuring the response times of window, text, or screen synchronizations, the time in the *Wait()* statement will artificially bloat response times.

The solution is to use synchronizations. See “Synchronization Options” for more information.

Re-recording portions of a script

Sometimes changes in application behavior result in portions of a recorded script becoming obsolete. Re-recording a portion of a use case is an option, but beware that the recorder may not be able to track the handles of the existing windows, resulting in incorrect window handle numbers. These issues can make

the process of integrating newly recorded script with an original script quite tedious. When a use case is small, it is recommended that you re-record the entire use case. Otherwise, follow the process outlined below to integrate a new script with an outdated script:

1. Comment out the section of code that is to be re-recorded.
2. Match the location of the code section requiring replacement with the corresponding section in the use case.
3. Bring a terminal services session up to the corresponding point in the use case.
4. Begin recording the open terminal services session.
5. Perform the actions of the use case that require replacement.
6. Stop the recorder (a script from the recording is then produced).
7. Replace the window handle variables in the newly recorded script with the respective handles of the original script. You may use functions such as *CitrixSearchWindow()* to locate correct window handles.
8. Copy the edited code into the original script.

Issues Specific to Citrix MetaFrame

Depending on how you connect to Citrix terminal services sessions and your licensing setup, you may receive one or both of the following dialog boxes:

- *ICA Seamless Host Agent*
- *Citrix License Warning Notice*

These dialog boxes are informational pop-ups that may or may not appear when you initially log into terminal services sessions. Following are two ways of handling these dialog boxes.

Handling Citrix dialog boxes (solution #1)

This solution creates an interrupt that handles the dialog boxes if they appear:

```
transaction TMain
  var
  begin
    CitrixInit(800, 600);
    CitrixAddInterrupt(INTERRUPT_WindowCreate, "ICA Seamless Host Agent",
MATCH_Exact);
    CitrixConnect("lab74", "labadmin", "labpass", "testlab1", COLOR_16bit);
    CitrixWaitForLogon();
    hWnd4 := CitrixWaitForWindowCreation("", MATCH_Exact, 0x96840000, -2,
572, 804, 30);
    CitrixWaitForWindowCreation("Program Manager");
    CitrixMouseClick(36, 17, hWnd4, MOUSE_ButtonLeft, MOD_None, -1, 0);
```

```
    hWnd11 := CitrixWaitForWindowCreation("", MATCH_Exact, 0x96400000, 2,
313, 163, 263);
    CitrixMouseClick(62, 247, hWnd11, MOUSE_ButtonLeft);
    CitrixWaitForWindow(hWnd11, EVENT_Destroy);
    hWnd12 := CitrixWaitForWindowCreation("Shut Down Windows", MATCH_Exact,
0x94C808CC, 191, 136, 417, 192);
    CitrixWaitForWindow(hWnd12, EVENT_Activate);
    CitrixMouseClick(203, 170, hWnd12, MOUSE_ButtonLeft);
    CitrixWaitForDisconnect();
  end TMain;

dclevent
  handler Handler1 <EVENT_CITRIXINTERRUPT>
  var
    nInterrupt, nWindow : number;
    nStyle              : number;
  begin
    CitrixGetActInterrupt(nInterrupt, nWindow);

    ErrorAdd(FACILITY_CITRIXENGINE, 47, SEVERITY_INFORMATIONAL);
    print(string(nWindow));
    if CitrixGetWindowStyle(nWindow, nStyle) and (nStyle <> 0xB4000000) then
      CitrixWaitForWindow(nWindow, EVENT_Activate);
      CitrixMouseClick(201, 202, nWindow, MOUSE_ButtonLeft);
      CitrixWaitForWindow(nWindow, EVENT_Destroy);
    end;
    ErrorRemove(FACILITY_CITRIXENGINE, 47);

  end Handler1;
```

Handling Citrix dialog boxes (solution #2)

This sample code loops for 30 seconds, waiting for the Citrix dialog boxes to appear. If the dialog boxes appear, this code closes them.

```
function MyCitrixStartup(nMaxWait: number optional): boolean
   var
      hwndICAHandle              : number;
      hwndFoundLicenseWarning    : number;
      nCount                     : number;
   begin
      hwndICAHandle:=-1;
      hwndFoundLicenseWarning:=-1;
      nCount:=0;

      if (nMaxWait = 0) then
         nMaxWait:=10;
       // if no wait time was passed,
       // use 10 tries (seconds) as a default
      end;

      MeasureStart("MyCitrixStartup");
```

```
//
// loop until we've handled the conditions or we've tried
//

   while ((nCount < nMaxWait) and ((hwndICAHandle <=0) or
      (hwndFoundLicenseWarning <=0))) do

//
// Just a little feedback, every 10 tries
//
      if ((nCount MOD 10) =0) then
         print(string(nCount) + ")MyCitrixStartup "
         + " vUser:" + string(GetUserId())
         + " hwndICAHandle=" + string(hwndICAHandle)
         + " hwndFoundLicenseWarning="
         + string(hwndFoundLicenseWarning),
         OPT_DISPLAY_ERRORS , TEXT_GREEN );
      end;
//
// if we haven't handled this window yet
//
      if (hwndICAHandle  <=0) then
         hwndICAHandle := CitrixWaitForWindowCreation
         ("ICA Seamless Host Agent", MATCH_Exact, 0x94C800C4,
         0, 0, 0, 0, false, 1, true);

         if (hwndICAHandle > 0) then
            if (CitrixWindowBringToTop(hwndICAHandle)) then
               CitrixKey(KEY_ENTER); // press ok to close the dialog
               CitrixWaitForWindow(hwndICAHandle, EVENT_Destroy);
               // wait for the close
            end; // end waiting for window to top
         end; // end if we have a valid handle
      end; // if window has not been found yet

      if (hwndFoundLicenseWarning <=0) then
         hwndFoundLicenseWarning := CitrixWaitForWindowCreation
         ("Citrix License Warning Notice", MATCH_Exact, 0x94C800C4,
         0, 0, 0, 0, false, 1, true);

         if (hwndFoundLicenseWarning > 0) then
            if (CitrixWindowBringToTop(hwndFoundLicenseWarning)) then
               CitrixKey(KEY_ENTER);  // Press ok
               CitrixWaitForWindow(hwndFoundLicenseWarning,
                  EVENT_Destroy);
               // wait for it to go away
            end; // end waiting for window to top
         end; // end if we have a valid handle
```

```
            end; // if window has not been found yet

            nCount :=nCount+1;
            Wait 1.0;
        end; // while nCount

        MeasureStop("MyCitrixStartup");
        //
        // return true if we handled any one of these conditions
        //

        MyCitrixStartup2 := (hwndFoundLicenseWarning > 0)
            or (hwndICAHandle > 0) ;

        print("MyCitrixStartup "
            + " vUser:" + string(GetUserId())
            + " Waited " + string(nCount) + " of " + string(nMaxWait)
            + " hwndICAHandle=" + string(hwndICAHandle)
            + " hwndFoundLicenseWarning=" + string(hwndFoundLicenseWarning),
            OPT_DISPLAY_ERRORS , TEXT_GREEN );

end MyCitrixStartup;
```

Index

A

Adding timers 78
Adding verifications 78
Application under test 69
AUT 69

B

Baseline performance 2
BDL 11, 54
Best practices 69

C

Citrix
  Customizing user input 45
  Verification 21, 52
Citrix ICA client 69
CitrixWaitForText 52
Creating use cases 72
CSV
  Comma separated values 77

D

Data files, multi-column 50
dclparam 46
dclrand 46
Defining Your Goals 71

F

Focus 77

H

Handling application errors 79

I

Input data, customization 50

K

Keyboard shortcuts 76

L

Load generator 70
Load test scripts
  Creating 11, 53
  Trying out 31

M

Monitoring templates 2
Monitoring tests 2
Multi-column data files 50

O

OCR 75
Optical Character Recognition 75
Optical character recognition 4, 24, 66

P

Parameter value 26
Parameter Wizard 45
Preparation 70
Print statement 28
Projects
  Defining new 7
  Reusing 3

R

Random Variable Wizard 47
Recorder, SilkPerformer's 11, 53
Recording
  tips 76
Recording use cases 74
Re-recording portions of a script 79

S

Screen synchronization 21
Silk TrueLog Explorer User Guide for SilkPerformer 52
SilkCentral 3
Synchronization
  Screen synchronization 75
  Text synchronization 75

Window synchronization 75
Synchronizing Text 52

T

Test automation 3
Test Manager 3
Test scripts
Customizing 45
Troubleshooting Scripts 78
TrueLog Explorer User Guide 52
TryScript runs 31

U

Use case
Numbered notation 73
User data
Customization 45
Parameterized 45
User profiles 2

V

Variables, random 47
Verification 21

W

Window focus 77
Workload 2
Writeln statement 28