

Borland®

Silk Performer 16.5

Web Load Testing
Tutorial

**Borland Software Corporation
700 King Farm Blvd, Suite 400
Rockville, MD 20850**

Copyright © Micro Focus 2015. All rights reserved. Portions Copyright © 1992-2009 Borland Software Corporation (a Micro Focus company).

MICRO FOCUS, the Micro Focus logo, and Micro Focus product names are trademarks or registered trademarks of Micro Focus IP Development Limited or its subsidiaries or affiliated companies in the United States, United Kingdom, and other countries.

BORLAND, the Borland logo, and Borland product names are trademarks or registered trademarks of Borland Software Corporation or its subsidiaries or affiliated companies in the United States, United Kingdom, and other countries.

All other marks are the property of their respective owners.

2015-09-10

Contents

Web Load Testing Tutorial	4
Web Load Testing Overview	4
Sample Web 2.0 Application	4
Defining a Web Load Test Project	5
Creating a Test Script	6
Recording a Test Script	6
Try Script Runs	6
Trying Out Your Test Script	6
Analyzing Test Scripts	7
Visual Analysis with TrueLog Explorer	7
Viewing a Virtual User Summary Report	8
Finding Errors in a TrueLog	9
Viewing Page Statistics	9
Comparing Record and Replay TrueLogs	9
Customizing Test Scripts	10
User-Input Data Customization	10
Verifications	13
Parsing Functions	16
Session Handling	17
User Profiles	20
Configuring Monitoring	21
Defining Monitoring Options	22
Adjusting Workload	22
Workload Models	22
Defining Workload	23
Running Load Tests	24
Running a Load Test	24
Monitoring Load Tests	24
Using a Graph to Monitor Server Performance	25
Exploring Test Results	26
TrueLog On Error	26
Performance Explorer Overview	27

Web Load Testing Tutorial

This tutorial will assist you in the process of using Silk Performer to load-test Web applications, and get you up and running as quickly as possible. It will help you take full advantage of Silk Performer's ease of use and leading-edge functionality.

Web Load Testing Overview

The fastest and easiest approach to test today's modern Web applications is to test them on the protocol level (HTML/HTTP). This approach generates simple scripts that incorporate advanced functionality. When testing a Web application with built-in AJAX logic and testing on the protocol level has proved to be unsuccessful we recommend using browser-driven Web load testing.

In addition to facilitating testing of today's modern Web applications on the protocol level (HTTP), Silk Performer now enables you to use real Web browsers (Internet Explorer) to generate load. In this way, you can leverage the AJAX logic built into Web applications to precisely simulate complex AJAX behavior during testing. This powerful testing approach provides results that reflect real-world end user browsing experience, including rendering time and protocol-level statistics.

Sample Web 2.0 Application

Silk Performer offers a modern sample Web application that you can use to learn about Web 2.0 application testing. The InsuranceWeb sample Web application is built upon ExtJS and JSF frameworks, uses AJAX technology, and communicates via JSON and XML.

The sample application is hosted at <http://demo.borland.com/InsuranceWebExtJS/>.

Insurance Co.
The Company You Can Trust

Protect your Future

Think of Tomorrow

Select a Service or login
Choose One

Email:

Password:

LOG IN SIGN UP

Our Profile
Premium Online Insurance Services

The Insurance Co. is a company you can trust for all of your insurance needs.
Our innovative approach as a leading insurance provider not only saves you time and money, it provides peace of mind. We have invested heavily in technology to bring you the very best in online services... [Learn more](#)

Main Services
What We Do

- > Life Insurance for every stage in life
- > Automobile Insurance for the entire family
- > Home Owners Insurance to protect your greatest asset
- > Renters Insurance that covers everything you own
- > Special Insurance products to cover unique circumstances

[All services](#)

Our Highlights
Recent News & Events

November 1, 2007
Default user:
User: john.smith@gmail.com
Password: john

October 25, 2007
Insurance Co. recognized as internet visionary by leading experts

October 1, 2007
Insurance Co. presents at Borlands user conference on ALM best practices

[News archive](#)

Newsletter Signup
Enter your email here:

[Unsubscribe](#)

This site is a fictitious representation of an online company for the purpose of demonstrating Borland Solutions

Home - Webservice - Settings - Contact Us

Defining a Web Load Test Project

1. Click **Start here** on the Silk Performer workflow bar.



Note: If another project is already open, choose **File > New Project** from the menu bar and confirm that you want to close your currently open project.

The **Workflow - Outline Project** dialog box opens.

2. In the **Name** text box, enter a name for your project.
3. Enter an optional project description in **Description**.
4. From the **Type** menu tree, select **Web business transaction (HTML/HTTP)**.
5. Click **Next** to create a project based on your settings.

The **Workflow - Model Script** dialog box appears.

Creating a Test Script

The easiest approach to creating a test script is to use the Silk Performer Recorder, the Silk Performer engine for capturing and recording Web traffic and generating test scripts based on the captured traffic.

Recording a Test Script

1. Click **Model Script** on the workflow bar. The **Workflow - Model Script** dialog box appears.
2. Select one of the listed browsers from the **Application Profile** list, depending on the browser you want to use for recording.
3. In the **URL** field, enter the URL that is to be recorded.



Note: The InsuranceWeb sample Web 2.0 application is available at <http://demo.borland.com/InsuranceWebExtJS/>. In the **Select a Service or login** list, the `Auto Quote` and `Agent Lookup` services are available for testing while the other listed services do not provide any functionality.

4. Click **Start recording**. The Silk Performer Recorder dialog opens in minimized form, and the client application starts.
5. To see a report of the actions that happen during recording, maximize the Recorder dialog by clicking the **Change GUI size** button. The maximized Recorder opens at the **Actions** page.
6. Using the client application, conduct the kind of interaction with the target server that you want to simulate in your test. The interaction is captured and recorded by the Recorder. A report of your actions and of the data downloaded appears on the **Actions** page.
7. To end recording, click the **Stop Recording** button.
8. Enter a name for the .bdf file and save it. If a test script can be generated without issues, the script will automatically display in the Workbench. If additional settings are required, the **Capture File** page displays. You can then generate a script out of the capture file.

Try Script Runs

Once you have generated a test script, determine if the script runs without error by executing a Try Script run. A Try Script run determines if a script accurately recreates the actions that you recorded with the Recorder. It also determines if the script contains any context-specific session information that you must parameterize before the script can run error free.

With Try Script 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 the scripted actions.

Trying Out Your Test Script

1. Click the **Try Script** button on the Silk Performer Workflow bar. The **Workflow – Try Script** dialog appears.
2. Choose a script from the **Script** list box.
3. In the **Profile** list box, the currently active profile is selected (this is the default profile if you have not configured an alternate profile).
 - a) To configure simulation settings for the selected profile, click **Settings** to the right of the list box.
 - b) To configure project attributes, select the **Project Attributes** link.
4. In the **Usergroup** list of user groups and virtual users, select the user group from which you want to run a virtual user.

Since this is a Try Script run, only one virtual user will be run.
5. To view the actual data that is downloaded from the Web server during the Try Script in real-time, select the **Animated Run with TrueLog Explorer** check box.

If you are testing anything other than a Web application, you should disable this option.

6. Click **Run**. The Try Script begins.

All recorded think times are ignored during Try Script runs. The **Monitor** window opens, giving you detailed information about the progress of the Try Script run. If you have selected the **Animated** option, TrueLog Explorer opens. Here you can view the actual data that is downloaded during the Try Script run. If any errors occur during the Try Script run, TrueLog Explorer can help you to find the errors quickly and to customize session information. Once you have finished examining and customizing your script with TrueLog Explorer, your script should run without error.

Analyzing Test Scripts

Silk Performer offers several means of evaluating a test script following the execution of a Try Script run.

Visual Analysis with TrueLog Explorer

One of TrueLog Explorer's most powerful features is its ability to visually render Web content that is displayed by applications under test. In effect, it shows you what virtual users see when they interact with an application.

The TrueLog Explorer interface is comprised of the following sections:

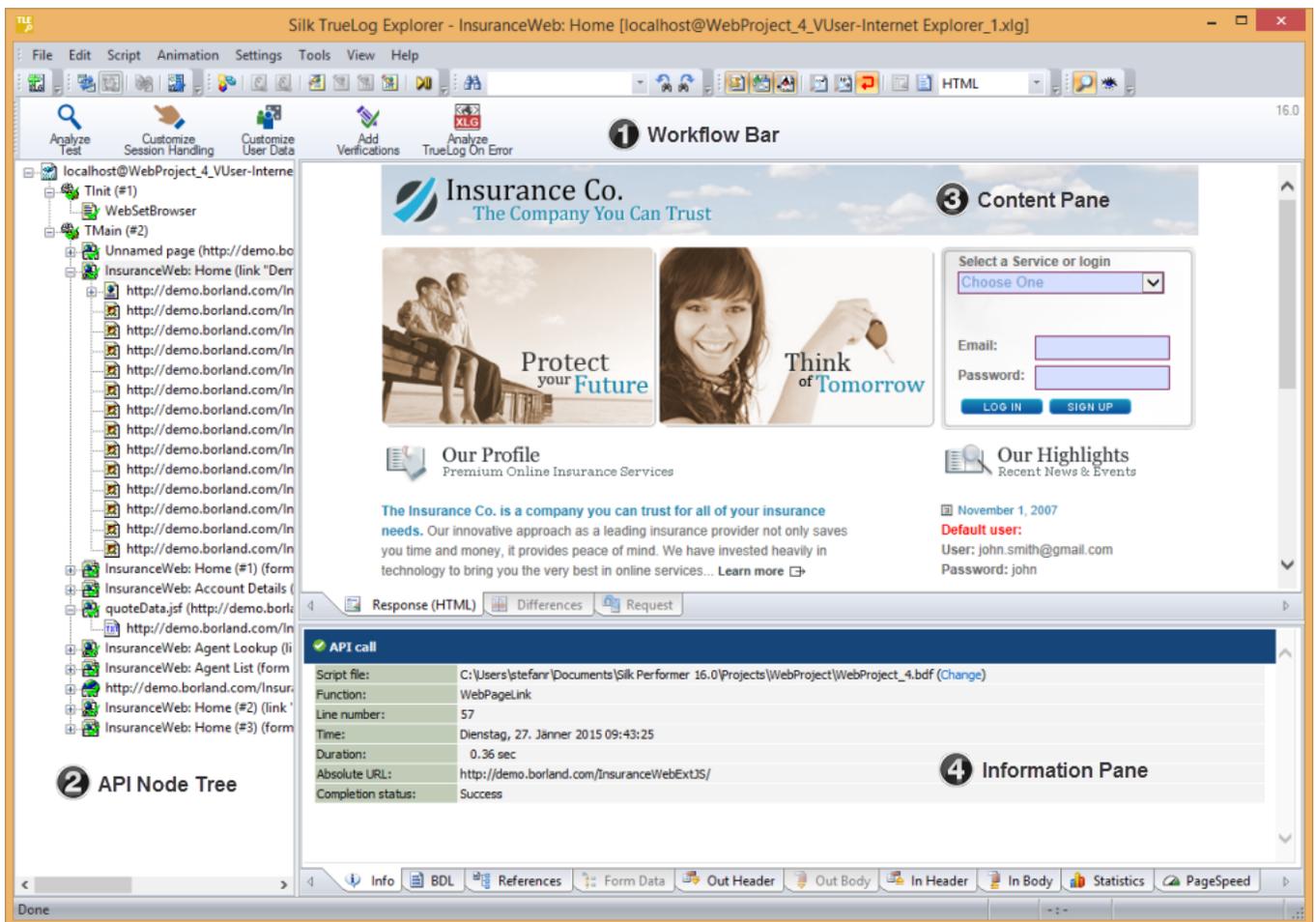
- The **Workflow Bar** acts as your primary interface as you work with TrueLog Explorer. The Workflow Bar reflects TrueLog Explorer's built-in testing methodology by supporting its five primary tasks.
- The **API Node Tree** menu on the left of the interface allows you to expand and collapse TrueLog data downloaded during tests. Each loaded TrueLog file is displayed here along with links to all relevant API nodes. You can click a node to display a screen shot in the **Screen** pane and history details in **Information** view.
- The **Content** pane provides multiple views of all received data.
- The **Information** pane displays data regarding testing scripts and test runs, including general information about the loaded TrueLog file, the selected API node, BDL script, and statistics.



Note: HTTP header data is not currently available.



Note: To launch TrueLog Explorer from Silk Performer, choose **Results > Explore TrueLog**.



Analyzing a Test Run

1. With the TrueLog from a Try Script run loaded into TrueLog Explorer, click the **Analyze Test** button on the Workflow bar.

The **Analyze Test** dialog box displays.

2. Proceed with one of the following options:

- View a virtual user summary report
- Look for errors in the TrueLog
- Compare the replay test run to the recorded test run

Viewing a Virtual User Summary Report

Virtual user summary reports are summary reports of individual Try Script runs that offer basic descriptions and timing averages. Each report tracks a single virtual user. Data is presented in tabular format.



Note: Because virtual user summary reports require significant processing, they are not generated by default.

To enable the automatic display of virtual user reports at the end of animated Try Script runs, or when clicking the root node of a TrueLog file in the menu tree, check the **Display virtual user report** check box at **Settings > Options > Workspace > Reports**.

Virtual user summary reports include details related to:

- Virtual users
- Uncovered errors
- Response time information tracked for each transaction defined in a test script
- Page timer measurements for each downloaded Web page
- Measurements related to each Web form declared in a test script, including response time measurements and throughput rates for form submissions using POST, GET, and HEAD methods.
- Individual timers and counters used in scripts (Measure functions)
- Information covering IIOF, Web forms, TUXEDO, SAP, and others

Finding Errors in a TrueLog

TrueLog Explorer helps you find errors quickly after Try Script runs. Erroneous requests can then be examined and necessary customizations can be made.



Note: When viewed in the menu tree, API nodes that contain replay errors are tagged with red “X” marks.

1. Open the TrueLog you want to analyze or modify.
2. Click **Analyze Test** on the workflow bar. The **Workflow - Analyze Test** dialog box opens.
3. Click the **Find errors** link. The **Step through TrueLog** dialog box appears with the **Errors** option selected.
4. Click **Find Next** to step through TrueLog result files one error at a time.

You can select different increments by which to advance through the TrueLog to visually verify that the script worked as intended (**Whole pages, HTML documents, Form submissions, or API calls**).

Viewing Page Statistics

After verifying the accuracy of a test run, TrueLog Explorer can analyze the performance of the application under “no-load” conditions via the **Statistics** tab under the **Information** pane. The **Overview** page details total page response times, document download times (including server busy times), and time elapsed for receipt of embedded objects.

Detailed Web page statistics show exact response times for individual Web page components. These detailed statistics assist you in pinpointing the root causes of errors and slow page downloads.

Detailed Web page drill-down results include the following data for each page component:

- DNS lookup time
- Connection time
- SSL-handshake time
- Send-request time
- Server-busy time
- response-receive time
- Cache statistics

Viewing an Overview Page

1. From the API Node Tree menu, select the API node for which you would like to view statistics.
2. Click the **Statistics** tab to open **Statistics** view.
3. Select specific components listed in the URL column for detailed analysis and page drill-down.

Comparing Record and Replay TrueLogs

By comparing a TrueLog that has been generated during the script development process alongside the corresponding TrueLog was recorded originally, you can verify that the test script runs accurately.

1. Click the **Analyze Test** button on the Workflow Bar. The **Workflow - Analyze Test** dialog box appears.
2. Click **Compare your test run**.
3. The corresponding recorded TrueLog opens in Compare view and the **Step through TrueLog** dialog box appears with the **Browser Nodes** option selected, allowing you to run a node-by-node comparison of the TrueLogs.
4. Click the **Find Next** button to step through TrueLog result files one page at a time.



Note: Windows displaying content presented during replay have green triangles in their upper left corners. Windows displaying content originally displayed during application recording have red triangles in their upper left corners.

Synchronizing Record and Replay TrueLogs

In compare mode you can synchronize corresponding API nodes between replay and record TrueLogs to identify differences between recorded values and replayed values.



Note: This feature is disabled when automatic synchronization of TrueLogs is enabled.

1. Enable compare mode by doing one of the following:
 - Choose **View > Compare Mode**.
 - Click the **Compare Mode** button on the toolbar.
2. Open a set of corresponding record and replay TrueLogs.
3. Right-click an API node and choose **Synchronize TrueLogs**. TrueLog Explorer locates the API node in the matching TrueLog that best correlates with the selected API node.

Customizing Test Scripts

Once you have generated a test script with Silk Performer and executed a Try Script run, TrueLog Explorer can help you customize the script in the following ways, among other options:

- **Parameterize input data** – With user-data customization, you can make your test scripts more realistic by replacing static, recorded, user-input data with dynamic, parameterized user data that varies with each transaction. Manual scripting is not required to run such data-driven tests. This feature is available for Web, database, XML, Citrix, SAPGUI, terminal emulation, and Oracle Forms applications.
- **Add verifications to test scripts** – With the **Add Verifications** tool, you can gain insight into data that is downloaded during tests, enabling you to verify that content sent by servers is received by clients. Verifications remain useful after system deployment for ongoing performance management. This feature is available for Web, database, XML, SAPGUI, Citrix, terminal emulation, and Oracle Forms applications. TrueLog Explorer supports bitmap and window verification for applications that are hosted by Citrix servers.

User-Input Data Customization

Without user-input data customization, all simulated transactions are identical and do not account for the variables that are typically experienced in real world environments.

For example, you can customize the user-input data that is entered into forms during testing using the **Parameter Wizard**. The **Parameter Wizard** lets you specify the values that are to be entered into form fields during testing. This enables test scripts to be more realistic by replacing recorded user-input data with randomized, parameterized user data.

Customizing HTML User Data With a New Parameter

Before proceeding, ensure that all static session information has been removed from your test script and that the most recent Try Script run produced a TrueLog that is open in TrueLog Explorer.

With HTML-based applications, the goal of user-data customization is to customize values submitted to form fields.

This task explains the process of creating a parameter based on a random variable.

1. Click **Customize User Data** on the workflow bar. The **Workflow - Customize User Data** dialog box opens.
2. Click the **Customize user input data in HTML forms** link. TrueLog Explorer then performs the following actions:
 - Selects the first **WebPageSubmit API call** node in the menu tree.
 - Opens the **Step through TrueLog** dialog box (with the **Form submissions** option button selected).
 - Displays **Post Data** view

Post Data view shows the page that contains the HTML form that was submitted by the selected `WebPageSubmit` call. When your cursor passes over a form control, a tool tip shows the control's name in addition to its initial and submitted values; an orange line indicates the corresponding BDL form field declaration in **Form Data** view below.

3. Click **Find Next** or **Find Previous** on the **Step through TrueLog** dialog box to browse through all `WebPageSubmit` calls in the TrueLog (these are the calls that are candidates for user-data customization).



Note: Highlighted HTML controls in **Post Data** view identify form fields that can be customized.

4. On the **Post Data** page, right-click the form control that you want to customize and choose **Customize Value**.

You can replace the recorded values with various types of input data (including predefined values from files and generic random values) and generate code into your test script that substitutes recorded input data with your customizations.

The **Parameter Wizard** opens.

With the **Parameter Wizard** you can modify script values in two ways:

- Use an existing parameter that is defined in the `dclparam` or `dclrand` sections of your script.
- Create a new parameter based on a new constant value, random variable, or values in a multi-column data file.

After you create a new parameter, that parameter is added to the existing parameters and is available for further customizations.

5. Click the **Create new parameter** option button and then click **Next** to create a new parameter. The **Create New Parameter** page opens.
6. Click the **Parameter from Random Variable** option button and then click **Next**. The **Random Variable** page opens.
7. From the list box, select the type of random variable that you want to insert into your test script and then click **Next**.

A brief description of the selected variable type appears in the lower window.

The **Name the variable and specify its attributes** page opens.

8. Enter a name for the variable in the **Name** text box.
9. Specify whether the values should be called in **Random** or **Sequential** order.

The **Strings from file** random variable type generates data strings that can either be selected randomly or sequentially from a specified file.
10. In the **File** group box, select a preconfigured data source from the **Name** list box and then click **Next**. The **Choose the kind of usage** page displays.

11. Specify the new random value to use by selecting one of the following choices:

- **Per usage**

- **Per transaction**
- **Per test**

12. Click **Finish**. Your test script now uses the random variable for the given form field in place of the recorded value. The new random variable function appears on the **BDL** page.

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

AJAX and Script Customization

The Silk Performer Recorder can record and replay Web applications that utilize AJAX (Asynchronous JavaScript and XML) requests. This is possible because Silk Performer recognizes asynchronous AJAX requests and responses that arrive in the form of either XML or JSON within HTML responses. Silk Performer scripts AJAX requests that it encounters as `WebPageUrl` calls.

Silk Performer and TrueLog Explorer support access to values within AJAX requests. This enables script customizations such as input-data parameterization, verification, parsing, and session-information customization within AJAX responses.



Tip: The InsuranceWeb demo application, available at <http://demo.borland.com/InsuranceWebExtJS/>, offers the functionality to switch between JSON and XML serialization methods for testing purposes. From the **Select a Service or login** list, select `Agent Lookup` for JSON or `Agent Lookup (XML)` for XML.

Pretty-Format JSON and XML Data

JSON and XML are data-structure formats commonly used in AJAX applications, REST techniques, and other environments. Silk Performer supports pretty-formatted viewing of XML and JSON-formatted byte streams in BDF scripts. Enhanced rendering of JSON formatted data enables easier customization of string values via TrueLog Explorer's string customization functions.

When JSON-formatted data is recorded or inserted into a BDF script, Silk Performer displays the raw JSON byte stream. Once displayed in JSON format, XML can easily be customized using Silk Performer's Parameter Wizard.

Silk Performer offers the option of viewing JSON data in either pretty-formatted JSON-rendering view or as a raw JSON byte stream.

Understanding JSON

According to JSON.org (www.json.org), "JSON (JavaScript Object Notation) is a lightweight data-interchange format. It is easy for humans to read and write. It is easy for machines to parse and generate. It is based on a subset of the JavaScript Programming Language..." "JSON is a text format that is completely language independent but uses conventions that are familiar to programmers of the C-family of languages, including C, C++, C#, Java, JavaScript, Perl, Python, and many others. These properties make JSON an ideal data-interchange language."

Enabling Pretty-Formatted JSON and XML Viewing in TrueLog Explorer

Enable pretty-formatted JSON and XML viewing on the **Rendered** page if TrueLog Explorer is unable to detect JSON or XML data. If TrueLog Explorer detects JSON and XML data, it is automatically pretty-formatted on the **Rendered** page.

1. Select a node on the TrueLog menu tree that includes JSON- or XML-formatted data (for example, a `HTTP Post` command node).
2. Click the **Rendered** tab.
3. Click **Auto View** on the toolbar. JSON and XML data is automatically pretty-formatted.
4. Right-click JSON or XML data and choose **Render As > JSON** or **Render As > XML** to pretty-format the data.

Enabling Pretty-Formatted JSON and XML Viewing in Silk Performer

1. Within a BDL script that includes JSON- or XML-formatted data, right-click within the screen data that you want to view.
2. Select your preferred viewing format from the context menu.
 - Select **Format As > JSON** to format the data in enhanced JSON format.
 - Select **Format As > XML** to format the data as raw XML.

You can also select **Format As > Auto Format** from the context menu to have Silk Performer determine the best formatting option for screen data types. By default, JSON and XML data is pretty-formatted in BDF scripts.

3. Right-click formatted data strings to access Silk Performer's standard string customization commands.



Note: Because formatted JSON data is integrated into the Silk Performer code editor, JSON-formatting can be undone/redone using the **Undo/Redo** buttons on the toolbar.



Note: Pretty-formatted JSON and XML data viewing is also available in TrueLog Explorer.

Verifications

Often application errors do not cause erroneous HTTP responses. Instead, the application responds with incorrect data values or with error messages incorporated in the HTML content, such as `A Servlet Exception occurred... or Server Too Busy...`. A simple check of the HTTP status code will not uncover this class of errors. Therefore, application errors are often overseen if you do not build additional checks into your test script. Verification functions help you to check for application errors that are not simple, standard HTTP errors.

With verification built into your test scripts, your test evolves from being a simple load test to a becoming a combined load and functionality test. You can use these scripts without major performance penalties, even during large load test scenarios. They will, therefore, be able to detect a new class of application errors—errors which only occur under load, but which would be overlooked by a simple load test script without additional verification checks.

Silk Performer offers three means of enhancing your test scripts with verification functionality:

- Automatically let the Recorder generate verification functions during the recording session.
- Directly enhance the script by coding verification functions manually.
- Apply verifications visually via TrueLog Explorer without the need to write a single line of BDL code. TrueLog Explorer will automatically generate verification functions into your script.



Tip: Refer to TrueLog Explorer Help for detailed information.

Automatically Generating Verifications During Recording

To enable the automatic generation of verifications during Silk Performer script recording:

1. Within Silk Performer, choose **Settings > Active Profile**. The **Profile - [<profile name>] - Simulation** dialog appears.
2. Select **Record > Web** in the shortcut list, then click the **Verification** tab.
3. In the **Recording** section, check the **Record title verification** and **Record digest verification** check boxes.
4. Click **OK**.

Title Verification

Title verification offers a simple, automatic means of checking to see if applications return correct Web pages. When retrieving HTML documents, the Recorder can automatically generate title verification functions for each page-level Web API call or low-level Web API call.



Note: Title verification does not work when HTML page titles (contents of the <TITLE HTML> tags) are not set, or when they share common names.

Title Verification

```
transaction TMain
begin
    ...
    WebVerifyHtmlTitle("ShopIt - Greetings",
        WEB_FLAG_IGNORE_WHITE_SPACE |
        WEB_FLAG_EQUAL | WEB_FLAG_CASE_SENSITIVE, 1,
        SEVERITY_ERROR, bVerifyTitleSuccess1);
    WebPageUrl("http://myHost/shopit");
```

Digest Verification

Digest verification is useful for verifying the content of relatively static HTML pages. Digest verification is not useful for HTML pages that are dynamic or contain session information in hyperlinks, embedded objects, or hidden form fields—and therefore change with each call.

The function calls `WebVerifyDataDigest` and `WebVerifyHtmlDigest` check to see if data that Silk Performer receives during replay differs from the data the Recorder captured during corresponding record sessions. With these function calls, Silk Performer generates digests that contain information about the occurrence of all characters and compares those results with digests generated by the Recorder. This function enables you to see if Silk Performer captures the same data during replay that it captured during recording. The Recorder automatically generates a digest verification function for each page-level Web API call and low-level Web API call that retrieves a document with a content type specified for digest verification (the default content type is text/html).

Digest Verification

```
const
    gsVerDigest_ShopIt_Greetings := "\h014200000B..."
                                   "\h016000500A..."
                                   "\h030001000A..."
                                   "\h0900410020...";

transaction TMain
begin
    ...
    WebVerifyDataDigest(gsVerDigest_ShopIt_Greetings, 162);
    WebPageUrl("http://myHost/shopit", "ShopIt - Greetings");
    ...
end TMain;
```

Enabling Verification Checks During Replay

See verification settings options for a list of possible verification checks that you can enable or disable during script replay.



Tip: Profile settings in scripts can be overridden using the `WebSetOption` BDL function.

1. Within Silk Performer, choose **Settings > Active Profile** . The **Profile - [<profile name>] - Simulation** dialog appears.
2. Select **Replay > Web** in the shortcut list, then click the **Verification** tab.
3. In the **HTML / XML** and **Data** areas, check which verification checks you want to enable during replay.

Inserting Content-Verification Functions

1. Open the TrueLog you want to analyze or modify.
2. Select a TrueLog API node that includes content that you want to have verified (for example, text or an image).
3. Select the content that is to be verified on the **Source** page.



Note: This step is not required for page-title and page-digest verification functions.

4. Click **Add Verifications** on the workflow bar. The **Workflow - Add Verifications** dialog box opens.
5. Select a pre-enabled verification:

- Verify the page title
- Verify the selected text
- Verify the selected text in an HTML table
- Verify the digest

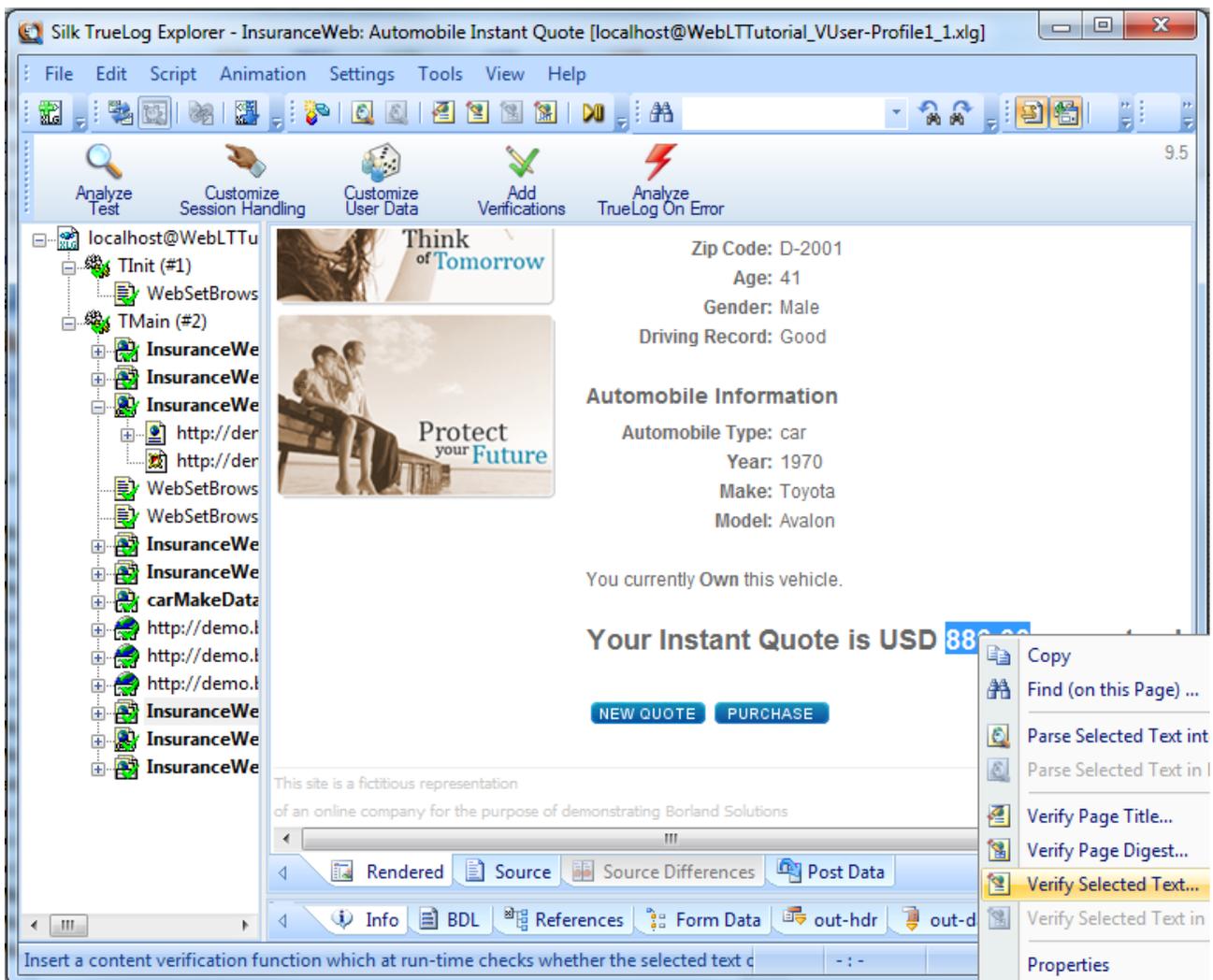
6. Complete the following dialog box.

Specify how verification functions should be inserted into the BDL script.



Note: Left and right boundaries are automatically identified for you.

7. Repeat the process for each verification you want to add to the BDL script.
8. Click **Yes** on the **Workflow - Add Verifications** dialog box. A Try Script run is initiated.
9. Confirm that verifications have passed successfully.
API nodes that include verifications are indicated with blue “V” symbols.



Parsing Functions

Similar to the verification functions, parsing functions are used to parse content returned from a Web server and check if the values meet your testing criteria. Different from the verification functions, which basically check the occurrence of a specified input value, the parsing functions parse the content of a server response and return the parsed value in a BDL variable. Typically, you will use parsing functions for the following tasks:

- Parse session IDs and replace static session IDs in the script with parsed, dynamic session IDs to maintain state information in Web applications. This is one of the main fields of application of parsing functions.
- Build enhanced verifications into your script with parsing functions, which cannot be done with verification functions. For example, if you want to verify that the value of column 2 of row 3 in an HTML table is equal to the sum of the values of column 2 of row 1 and column 2 of row 2. By using parsing functions to parse out the three values and compare them in our script, you can build this enhanced verification check.
- Conditional execution of parts of your testing script which depends on the data returned from the server. For example, an HTTP request returns an HTML page which includes: Results: <nnn> items found. You want to execute different actions depending on the value of <nnn>. Let's say you want to exit the transaction if <nnn> = 0, and you want to go to link **Details** if <nnn> = 1, and you want to go

to link **Next** if `<nnn>` is greater than 0. To accomplish this, you need to parse the value of `<nnn>` from the HTML page and script the actions, depending on the value you have parsed out.

Silk Performer offers two means to enhancing your test scripts with parsing functionality:

- Directly enhance your script by coding verification functions manually.
- Apply parsing functions visually in TrueLog Explorer without the need to write a single line of BDL code. TrueLog Explorer will automatically generate parsing functions in your script.



Tip: Refer to TrueLog Explorer Help for detailed information.

HTML Content Parsing

HTML content parsing functions parse out portions of the rendered, visible HTML content. The HTML content verification and parsing functions allow you to verify and/or parse the textual content that you see in your browser. Apply HTML content parsing functions in the **HTML view** of TrueLog Explorer, which includes the following functions:

- `WebParseHtmlBound`
- `WebParseHtmlTitle`
- `WebParseTable`
- `WebParseResponseTag`
- `WebParseResponseTagContent`

Response Data Parsing

Data parsing functions parse out portions of the response data returned from the server. In cases in which an HTML document is returned from the server, the complete source code of the HTML document is parsed. Apply data verification functions in the **source view** of TrueLog Explorer, which includes the following functions:

- `WebParseDataBound` (this function replaces the deprecated `WebParseResponseData` function)
- `WebParseResponseHeader`

Session Handling

Web applications often use a unique Web session ID so that the Web server is able to handle further client requests.

Refer to TrueLog Explorer Help for detailed information.

Customized Session Handling

Web server applications often generate information at runtime that is necessary in order to identify further client requests. In the response to the browser, the server may include a unique string, commonly known as the Session ID. This string is returned by the browser to the server as a part of each subsequent request, allowing the server to identify the unique Web session of which the request is a part. Generally, Session IDs refer to the method the Web server application uses to identify individual users and to associate this identification with the state of the user session information that the application has previously had with those users.

Session IDs can be sent to the client in a number of ways. Most often you will find them included in cookies, or inside HTML as part of URL's used in hyper links or embedded objects, or in hidden HTML form fields. Session IDs are sent back to the server in cookies, URL's, and HTTP post data.

Session Information Inside Cookies

From the server:

```
Set-Cookie: SessionID=LGIJALLCGEBMIBIMFKOEJIMM; path=/
```

To the server:

```
Cookie: SessionID=LGIJALLCGEBMIBIMFKOEJIMM
```

Session Information in the URL's of HTML Links

From the server:

```
<html>
...
<a href="/ShopIt/acknowledge.asp?
SessionID=LGIJALLCGEBMIBIMFKOEJIMM" >
  Enter Shop
</a>
...
</html>
```

To the server:

```
GET /ShopIt/acknowledge.asp? SessionID =
LGIJALLCGEBMIBIMFKOEJIMM HTTP/1.1
```

Session Information in Hidden Form Fields

From the server:

```
<html>
...
<form action="kindofpayment.asp" method="post" >
  Currently we only accept Credit Cards
  <input type="hidden" name="SessionID"
value="LGIJALLCGEBMIBIMFKOEJIMM">
  <input type="text" name="name" value="Jack " >
  <input type="submit" name="paymentButton" value="Submit">
</form>
...
</html>
```

To the server:

```
POST /ShopIt/kindofpayment.asp HTTP/1.1
...
SessionId=LGIJALLCGEBMIBIMFKOEJIMM&name=Jack&paymentButton=Submi
t
```

When to Use Customized Session Handling

Assuming that a `WebPageUrl` call in your script uses a URL that contains a session ID as part of the query string of the URL. When replaying the script, this hard coded static session ID is sent to the server. Because the session ID does not correctly identify your replayed session—it still identifies the recording session—the replay won't work correctly. By not replacing static session IDs in your script with dynamic values that had been generated at runtime, your Web application will usually generate errors such as `We are sorry, your session has expired. Please return to the login screen and try again.`

The good news is that with Silk Performer, session customization is not often necessary and, even if manual session customization is needed, Silk Performer's TrueLog Explorer will guide you through the process.

Most often when you record a script, it will work without any modifications necessary for the customization of session handling. Silk Performer therefore uses multiple, highly sophisticated methods that prevent the user from manually handling hard coded session IDs:

- **Cookie Management:** When your server uses cookies to exchange session information, Silk Performer will automatically handle dynamic session ID values. Because Silk Performer exactly emulates the cookie management of a browser, it will send cookies the same way a browser sends them to your server. No manual interaction is needed for state management.
- **page-level web API:** Using the Page-level API when recording (which is the default setting) will generate scripts that mainly generate context-full Web API function calls such as `WebPageLink` and `WebPageSubmit`. Context-full Web API calls are working on the HTML level, not the HTTP level, and thus they do not use URL's as parameters. For all context-full API calls, manual session customization is not needed. The page-level API is used if you choose the application type `Web business transaction (HTML/HTTP)` in the **Outline Project** dialog. Therefore it is strongly recommended that you use Silk Performer's page-level API instead of the low-level web API.

Due to the heavy use of client side Java Script for dynamically generating HTML, the Silk Performer Recorder sometimes loses the HTML context and scripts context-less Web API calls. Context-less Web API calls, such as `WebPageUrl` and `WebPageForm`, contain URL's as parameters. In these rare cases, your script may contain hard coded session IDs. You will find them in the URL parameters of Web API calls and in the form fields declared in the `dclform` section of your script.

Context-full script (there is nothing to customize)

```
transaction TMain
  begin
    WebPageUrl("http://lab38/ShopIt/"); // first call is always
context-less
    WebPageLink("Join the experience!");
    WebPageSubmit("Enter", SHOPIT_MAIN_ASP001);
    WebPageLink("Products");
  end TMain;

dclform
SHOPIT_MAIN_ASP001:
  "SessionID" := "" <USE_HTML_VAL>, // hidden value:
  "LGIJALLCGEBMIBIMFKOEJIMM"
  // recognized as a hidden form
  // field, the value is taken from
  // the actual HTML form field.
  "name" := "Jack", // changed
  "New-Name-Button" := "" <USE_HTML_VAL> ; //unchanged
value: "Enter"
```

Script with context-less functions (static session data that needs to be customized is included in the DCLFORM section)

```
transaction TMain
  begin
    WebPageUrl("http://lab38/ShopIt/"); // first call is always
context-less
    WebPageUrl("http://lab38/ShopIt/main.asp", NULL,
SHOPIT_MAIN_ASP001);
    WebPageForm("http://lab38/ShopIt/main.asp",
SHOPIT_MAIN_ASP002);
    WebPageUrl("http://lab38/ShopIt/products.asp");
  end TMain;

dclform
SHOPIT_MAIN_ASP001:
  "from" := "welcome";

SHOPIT_MAIN_ASP002:
  "SessionID" := "LGIJALLCGEBMIBIMFKOEJIMM",
```

```
"name" := "Jack" ,  
"New-Name-Button" := "Enter" ;
```

Customizing Session Handling

The following steps are required to customize session handling.

Refer to TrueLog Explorer Help for detailed information.

1. Identify the session ID that needs to be customized.
2. Search for the first response (Web API call in the script) in which the session ID is sent from the server to the client.
3. Parse out the session ID from the found response of the Web API call into a variable.
4. Replace all occurrences of the hard coded session ID in the script with this variable.

User Profiles

Browser Types

Virtual users in a test can be customized to use any of a wide choice of Web browsers, and of the functionality they embody. The most popular browsers in use today can be emulated. Lesser-known browsers are also available, as are browsers serving mobile device users. Custom browsers can be defined, using different versions of threading technologies and of HTTP.

Bandwidth

Modems are the means that home users typically use to access the Internet. Since a user's modem is sometimes the slowest link in the network communication chain, modems can be simulated in a load test.

Virtual users can be customized to use the bandwidth associated with any of the major connection types in wide use among consumers today. Custom settings can be defined where connections use different bandwidth settings for downstream (from the server to the client) and upstream (from the client to the server) traffic.

Adding a Profile

1. Select **Project > New Profile** . The **New Profile** dialog opens.
2. Enter a **Name** for the new profile and click **OK**. The **Profiles** folder expands in the **Project** menu tree and the new profile is available.

Configuring Browser Settings

1. In Silk Performer, expand the **Profiles** node in the **Project** tree.
2. Right-click the profile that you want to configure and choose **Edit Profile**.



Tip: Alternatively, you can choose **Settings > Active Profile** from the menu bar.

The **Profile - [<profile name>]** dialog box opens, and the **Replay** category is displayed in the shortcut list on the left.

3. In the shortcut list, click the **Web** icon.
4. Click the **Browser** tab.

Use the **Browser type** area to specify browser-specific settings.

5. From the **Browser** list box, select the Web browser you want to use for your simulation.

The selection you make determines the format of the header information included in your HTTP requests and the threading model used for simulation.



Note: For mobile Web application testing (iPhone, iPad, Android, Windows Phone or Blackberry) you can change the user agent string used for recording.

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

Configuring Browser-Simulation Settings

1. In Silk Performer, expand the **Profiles** node in the **Project** tree.
2. Right-click the profile that you want to configure and choose **Edit Profile**.



Tip: Alternatively, you can choose **Settings > Active Profile** from the menu bar.

The **Profile - [<profile name>]** dialog box opens, and the **Replay** category is displayed in the shortcut list on the left.

3. In the shortcut list, click the **Web** icon.
4. Click the **Simulation** tab.

Use the **Simulation** area to set options for realistic simulation of users visiting Web sites.

5. Check the **Simulate user behavior for each transaction** check box to have each virtual users reset their browser emulation after each transaction.

Depending on the additional option you select, Silk Performer either simulates users who visit a Web site for the first time or users who revisit the site. While users who visit a site for the first time have no persistent cookies stored and no documents cached, users who revisit a page typically have closed their browsers between the Web site visits, have documents cached, and persistent cookies set.

Disabling this option lets the virtual user emulate a Web browser that is not closed until the end of the test, thereby reusing cached information throughout multiple transactions.

6. Click the **First time user** option button to generate a realistic simulation of users who visit a Web site for the first time.

Persistent connections will be closed, the Web browser emulation will be reset, and the document cache, the document history, the cookie database, the authentication databases, and the SSL context cache will be cleared after each transaction. In such instances, Silk Performer downloads the complete sites from the server, including all files.

7. Click the **Revisiting user** option button to generate a realistic simulation of users who revisit a Web site.

Persistent connections will be closed, and the document history, the non-persistent cookie database, the authentication database, and the SSL context cache will be cleared after each transaction. In such cases, users do not clear the document cache. For more details, review the `WebSetUserBehavior` function in the BDL Function Reference.

Use the **User tolerance** area to adjust the advanced options of the user tolerance simulation.

8. Click **OK** to save your settings on the **Profile - [<profile name>]** dialog box.

Configuring Monitoring

Before running a test you need to define how Performance Explorer, the Silk Performer server monitoring tool, is to monitor local and remote servers involved in your test. Server monitoring reveals, locates, and assists in resolving server bottlenecks, allowing you to examine the performance of operating systems and application servers.

Three monitoring options are available:

- **Default monitoring** - This option directs Performance Explorer to monitor a recommended set of data sources based on the application type under test. This is equivalent to enabling the **Automatically start monitoring** and **Use default monitoring template** settings for the Performance Explorer workspace (**Settings > Active Profile > Replay > Monitoring > Use default monitoring template**).
- **Custom monitoring** - This option opens Performance Explorer in monitoring mode with the **Data Source Wizard - Select Data Sources** dialog box open, enabling you to manually configure data

sources. Your Performance Explorer monitoring project settings will be saved along with your Silk Performer project settings.

- **No monitoring** - This option enables you to run your test without monitoring of any local or remote servers. With this option the **Automatically start monitoring** setting is disabled (**Settings > Active Profile > Replay > Monitoring > Use default monitoring template**).

Defining Monitoring Options

1. Click **Configure Monitoring** on the workflow bar. The **Workflow - Configure Monitoring** dialog box appears.
2. Select one of the following options and click **Next**:
 - **Default monitoring** - This option directs Performance Explorer to monitor a recommended set of data sources based on the application type under test. This is equivalent to enabling the **Automatically start monitoring** and **Use default monitoring template** settings for the Performance Explorer workspace (**Settings > Active Profile > Replay > Monitoring > Use default monitoring template**).
 - **Custom monitoring** - This option opens Performance Explorer in monitoring mode with the **Data Source Wizard - Select Data Sources** dialog box open, enabling you to manually configure data sources. Your Performance Explorer monitoring project settings will be saved along with your Silk Performer project settings.
 - **No monitoring** - This option enables you to run your test without monitoring of any local or remote servers. With this option the **Automatically start monitoring** setting is disabled (**Settings > Active Profile > Replay > Monitoring > Use default monitoring template**).

(for Default Monitoring and Custom Monitoring only) A confirmation dialog box will notify you if you have logging enabled. Logging may skew your test results.

3. Click **OK** to accept your logging settings or click **Cancel** to adjust your logging options (**Settings > Active Profile > Results > Logging**).
4. *(for Custom Monitoring only)* Performance Explorer starts and the **Data Source Wizard** opens. Complete the steps outlined in the wizard.
5. The **Workflow - Workload Configuration** dialog box appears. Click **OK** to accept your monitoring settings.

Adjusting Workload

Configuring workload is part of the process of conducting a load test. Silk Performer offers different workload models to be used as a basis for your load test. Before configuring workload, you must select the model that best fits your needs.

You can define more than one workload model in your load test project and save them for further usage, but only one workload model can be active at a time. The accepted baseline results are associated with a workload model. If you copy or rename a workload model, the accepted baseline results are copied or renamed accordingly.

Workload Models

Silk Performer provides the following workload models:

- **Increasing** – At the beginning of a load test, Silk Performer does not simulate the total number of users defined. Instead, it simulates only a specified part of them. Step by step, the workload increases until all the users specified in the user list are running.

This workload model is especially useful when you want to find out at which load level your system crashes or does not respond within acceptable response times or error thresholds.

- **Steady State** – In this model, the same number of virtual users is employed throughout the test. Every virtual user executes the transactions defined in the load-testing script. When work is finished, the virtual user starts again with executing the transactions. No delay occurs between transactions, and the test completes when the specified simulation time is reached.

This workload model is especially useful when you want to find out about the behavior of your tested system at a specific load level.

- **Dynamic** – You can manually change the number of virtual users in the test while it runs. After the maximum number of virtual users is set, the number can be increased or decreased within this limit at any time during the test. No simulation time is specified. You must finish the test manually.

This workload model is especially useful when you want to experiment with different load levels and to have the control over the load level during a load test.

- **All Day** – This workload model allows you to define the distribution of your load in a flexible manner. You can assign different numbers of virtual users to any interval of the load test, and each user type can use a different load distribution. Therefore, you can design complex workload scenarios, such as workday workloads and weekly workloads. You can also adjust the load level during a load test for intervals that have not started executing.

This workload model is especially useful when you want to model complex, long lasting workload scenarios in the most realistic way possible.

- **Queuing** – In this model, transactions are scheduled by following a prescribed arrival rate. This rate is a random value based on an average interval that is calculated from the simulation time and the number of transactions per user specified in `dcluser` section of your script. The load test finishes when all of the virtual users have completed their prescribed tasks.



Note: With this model, tests may take longer than the specified simulation time because of the randomized arrival rates. For example, if you specify a simulation time of 3,000 seconds and want to execute 100 transactions, then you observe an average transaction arrival rate of 30 seconds.

This workload model is especially useful when you want to simulate workloads that use queuing mechanisms to handle multiple concurrent requests. Typically, application servers like servlet engines or transaction servers, which are receiving their requests from Web servers and not from end users, can be accurately tested by using the queuing model.

- **Verification** – A verification test run is especially useful when combined with the extended verification functionality. This combination can then be used for regression tests of Web-based applications. A verification test run always runs a single user of a specific user type on a specified agent computer.

This workload is especially useful when you want to automate the verification of Web applications and when you want to start the verification test from the command line interface.

Defining Workload

Prior to the execution of a load test, you must specify the workload model that you want to use and configure its settings. To select a workload model, click **Adjust Workload** in the workflow bar.

1. Click **Adjust Workload** on the workflow bar. The **Workflow - Select and adjust Workload** dialog box appears.
2. Select one of the following workload models by clicking the appropriate option button:
 - Increasing
 - Steady State
 - Dynamic
 - All Day
 - Queuing
 - Verification
3. Click **Next**.

4. Configure specific workload types based on the steps outlined in the Help.

Running Load Tests

In a load test, multiple virtual users are run by means of a test script against a target server to determine the load impact on server performance. A large load test requires an appropriate testing environment on your LAN, including a full complement of agent computers to host the virtual users.

It is essential that you complete the following tasks:

- Set options for the appropriate type of test that is to be run
- Accurately define the required workloads
- Enable the generation of test results to assess server performance

Do not enable complete result-logging during load testing because it might interfere with load-test results. However, the **TrueLog On Error** logging option writes necessary log files to a disk when errors occur, allowing you to inspect replay errors visually.

Running a Load Test

1. Click **Run Test** on the workflow bar. The **Workflow - Workload Configuration** dialog box appears.
2. Click **Run** to start the load test.
3. Click **OK** on the **New Results Files Subdirectory** dialog box.
(*Optional*) To specify a name for the results subdirectory, uncheck the **Automatically generate unique subdirectory** check box and enter a name for the new subdirectory in the **Specify subdirectory for results files** text box.

Monitor test progress and server activity by viewing the Silk Performer tabular monitor view and the Performance Explorer graphical monitor view.

Monitoring Load Tests

Detailed real-time information is available to testers while Silk Performer load tests run. Graphic displays and full textual reporting of activity on both the client side and the server side offer intuitive monitoring of the progress of tests as they occur.

Directly from the workbench on which the test is conducted, a tester can view comprehensive overview information about the agent computers and virtual users in the test. A tester can control the level of detail of the displayed information, from a global view of the progress of all the agent computers in the test to an exhaustive detail of the transactions conducted by each virtual user. Progress information for each agent and each user is available in multiple categories. Run-time details for each user include customizable, color-coded readouts on transactions, timers, functions, and errors as they occur.

In addition, real-time monitoring of the performance of the target server is available in graphical form. Charts display the most relevant performance-related information from a comprehensive collection of the Web servers, application servers, and database servers running on all of the OSes most widely used today. Multiple charts can be open at the same time, and these charts can be juxtaposed to provide the most relevant comparisons and contrasts for the tester. A menu tree editor allows for the combination of elements from any data source in the charts. Response times and other performance information from the client application can be placed in the same chart as performance data from the server. This feature enables a direct visual comparison so that one can determine the influence of server shortcomings on client behavior.

Monitoring Agent Computers During load Testing

Use the **Monitor** window to view progress while a load test runs. The top part of the window displays information about the progress of agent computers and user groups.

Among the comprehensive number of information options, you can view the following information:

- Status of a particular agent
- Percentage of the test that is complete on an agent
- Number of executed transactions

Monitoring a Specific Agent During load Testing

In the top part of the **Monitor** window, select the specific agent to monitor.

The following information about the virtual users running on the selected agent appears in the bottom of the **Monitor** window:

- Status
- Name of the current transaction
- Percentage of work completed
- Number of executed transactions

Monitoring a Specific Virtual User During load Testing

In the bottom part of the **Monitor** window, right-click the virtual user that you want to monitor and choose **Show Output of Vuser**.

In the **Virtual User** window, Silk Performer displays detailed run-time information about the selected user, such as the transactions and functions the user executes and the data the user sends to and receives from the server.



Tip: Right-click the virtual user area and choose **Select Columns** to select the columns you want to view.

Using a Graph to Monitor Server Performance

1. Click **Confirm Baseline** on the workflow bar. The **Workflow - Confirm Baseline** dialog box opens.
2. Click **Define monitoring options** to specify the settings for receiving online performance data. The **Profile Results** dialog box opens.
3. In the **Profile Results** dialog box, check the **Automatically start monitoring** check box to automatically start monitoring while running a load test and then choose one of the following options:
 - Click the **Use default monitoring template** option button.
 - Click the **Use custom monitoring template** option button to create a customized monitoring template.
4. Click **Create/Edit Custom Monitor Template**. Performance Explorer appears.
5. Close all monitor windows that you are not currently using.
6. Click **Monitor Server** on the Performance Explorer workflow bar.
Alternatively, you can choose **Results > Monitor Server** from the menu bar.
The **Data Source Wizard / Select Data Sources** dialog box opens.
7. Perform one of the following steps:
 - If you are certain of the data sources that the server provides, click the **Select from predefined Data Sources** option button to then select them from the list of predefined data sources.
 - If you are uncertain of the data sources that the server provides, click the **Have Data Sources detected** option button to let Performance Explorer scan the server for available data sources.
8. Click **Next**.

In the menu tree, expand the folder that corresponds to the OS on which the server and application are running.

9. Select the server application you want to monitor from the list that appears.
For example, to monitor the OS, select **System**.
10. Click **Next**. The **Connection Parameters** dialog box opens.
11. In the **Connection parameters** area, specify connection parameters such as the host name or IP address of the appropriate server system, the port number, and other data required to connect to the data source.
The specified data depends on the OS running on the computer that you are monitoring.
12. Click **Next**. The **Select Displayed Measures** dialog box opens.
13. Expand the menu tree and select the factors you want to monitor.
14. Click **Finish**. A Monitor graph shows the specified elements in a real-time, color-coded display of the server performance. Beneath the graph is a list of included elements, a color-coding key, and performance information about each element.

Exploring Test Results

Silk Performer offers several approaches to displaying, reporting, and analyzing test results. Defined measurements take place during tests and can be displayed in a variety of graphical and tabular forms. Options include the following:

- **Performance Explorer:** This is the primary tool used for viewing test results. A fully comprehensive array of graphic features displays the results in user-defined graphs with as many elements as are required. The results of different tests can be compared. There are extensive features for server monitoring. A comprehensive HTML based overview report that combines user type statistics with time series test result information is also available.
- **TrueLog On Error:** Silk Performer provides full visual verification under load capabilities for various application types. It allows you to combine extensive content verification checks with full error drill-down analysis during load tests.
- **Virtual User Report files:** When enabled, these files contain the simulation results for each user. Details of the measurements for each individual user are presented in tabular form.
- **Virtual User Output files:** When enabled, these files contain the output of write statements used in test scripts.
- **Baseline Reports:** A detailed XML/XSL-based report that provides you with a summary table, transaction response-time details, timers for all accessed HTML pages, Web forms, and errors that occurred. This information is available for all user types involved in baseline tests.
- **Silk Central Reports:** Silk Performer projects can be integrated into Silk Central (Silk Central) test plans and directly executed from Silk Central. This allows for powerful test-result analysis and reporting. For detailed information on Silk Central reporting, refer to *Silk Central Help*.

TrueLog On Error

With Silk Performer TrueLog technology, you can find errors that usually occur to only a subset of users when your application is under a heavy load. For most applications, this is the type of load that will most likely be experienced once the application is deployed in the real world. Typical errors include incorrect text on a Web page, incorrectly computed and displayed values, or application-related messages, such as `Servlet Error` or `Server Too Busy` errors. These are not system-level errors and are displayed on Web pages with `HTTP 200` status codes.

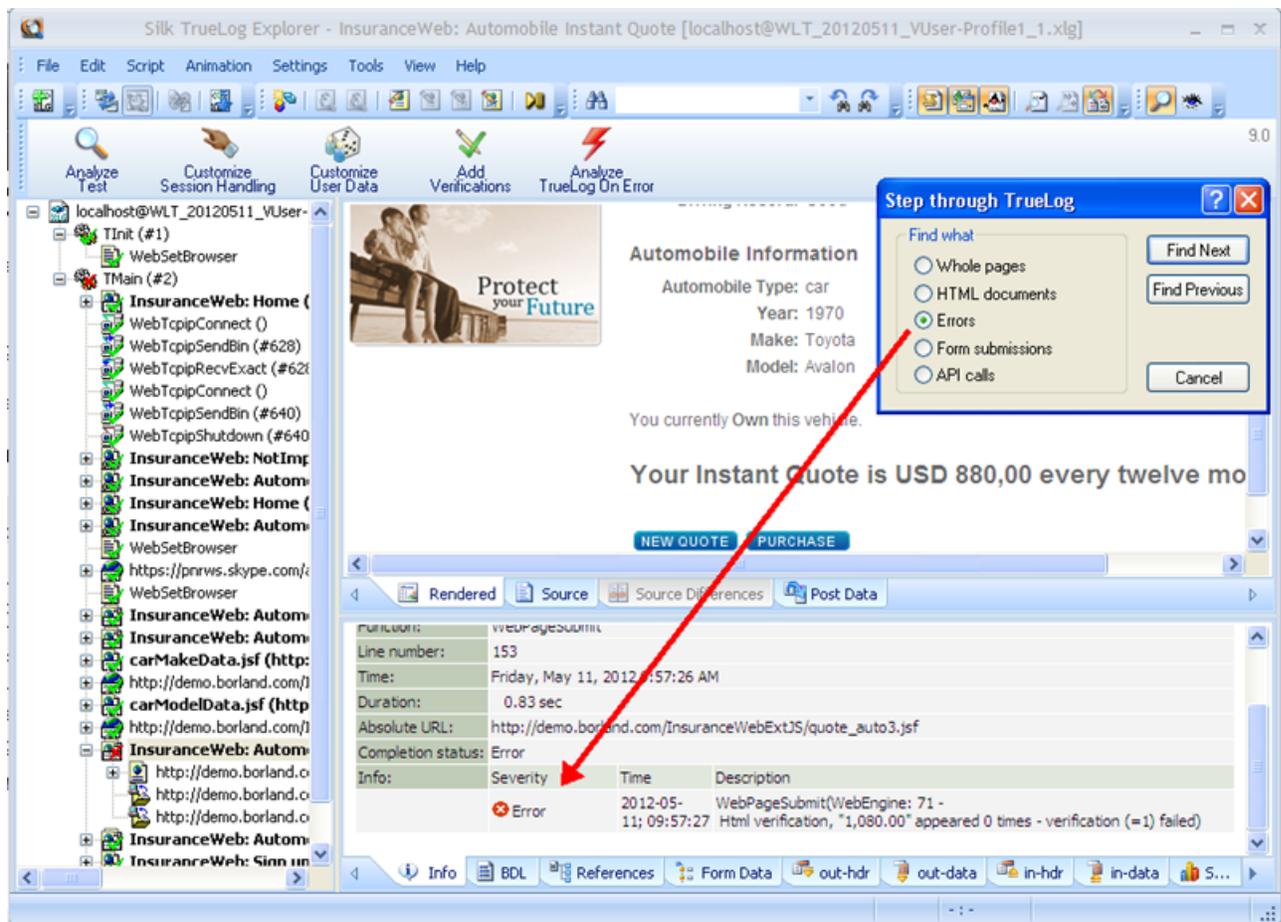
TrueLog Explorer provides a view into Silk Performer verification-under-load capabilities with the following features:

- Visual content verification allows you to visually define the content that is to be verified.
- TrueLog On Error generation and TrueLog On Error analysis allow you to visually analyze errors to identify their root causes.

Viewing Errors in TrueLog Explorer

1. When an error occurs during a load test, you can view the visual content of the TrueLog by clicking the **Explore Results** button on the workflow bar. The **Workflow - Explore Results** dialog opens.
2. Click the Silk TrueLog Explorer link. TrueLog Explorer opens with the **Step Through TrueLog** dialog box active. Select **Errors**.
3. Navigate from one occurrence of an error to the next.

 **Tip:** To display the history of an error, click through the preceding API nodes in the menu tree.



Performance Explorer Overview

Performance Explorer offers two main capabilities:

- Results Analysis
- Real-Time Monitoring

Results Analysis

With the Results Analysis functionality of Performance Explorer, you can analyze the results of a completed load test by creating charts, tables, and reports. With charts, you can visualize the data that was collected during the load test, and reports as well as tables help you to summarize important data and findings. The basis for all charts, tables, and reports is the data in the .tsd files. During each load test, Silk Performer captures a big amount of data and stores it in several time-series data (.tsd) files. When a load test is completed, you can load the data (the .tsd files) in Performance Explorer and visualize and edit them according to your requirements. All charts are fully customizable, and they can contain as many measures

as required. You can open multiple charts using measures from one or multiple tests to show similarities and differences. Performance Explorer provides a variety of templates for charts, tables, and reports. Furthermore, you can place information on client and server performance in a single chart, so that you can directly view the effect of server performance on client behavior. By saving your changes as a template, you can reuse your individual settings.

Real-Time Monitoring

With the Real-Time Monitoring functionality of Performance Explorer, you can monitor a broad range of systems by creating and configuring charts that show the system performance in real-time. It is possible to open multiple charts at the same time, which allows you to watch the performance of two or more systems simultaneously, for example the web server performance and the operating system performance. Adding measures to a chart is easy and intuitive in Performance Explorer: You can drag a single measure or a set of measures from the tree onto the chart.

Overview Report Measurements

The overview report comprises the following sections:

- General information
- Summary tables
- User types
- Custom charts
- Custom tables
- Detailed charts

General information

The general information section includes administrative information in tabular form as well as important load test results in a graphical form.

Administrative information includes the project name, a description of the project, the load test number, a description of the load test, the date of the load test, the duration of the load test, the number of used agent computers, and the number of virtual users that were running.

The charts display the number of active virtual users, response time measurements for transactions, and the number of errors that occur over time. Transaction response times are provided for successfully executed transactions, for failed transactions, and for cancelled transactions.

Additional charts display summary measurements related to the type of load testing project. For example, in the case of Web application testing, response time measurements for Web pages are presented in a graph.

Summary tables

This section contains summary measurements in tabular form, that is, aggregate measurements for all virtual users. The first table provides general information, such as the number of transactions that were executed and the number of errors that occurred. All the following tables provide summary information relevant to the type of application that was tested.

User types

For each user group, this section provides detailed measurements in tabular form. The measurements include transaction response times, individual timers, counters, and response time and throughput measurements related to the type of application that was tested (Web, database, CORBA, or TUXEDO). In addition, errors and warnings for all user groups are listed.

Custom charts

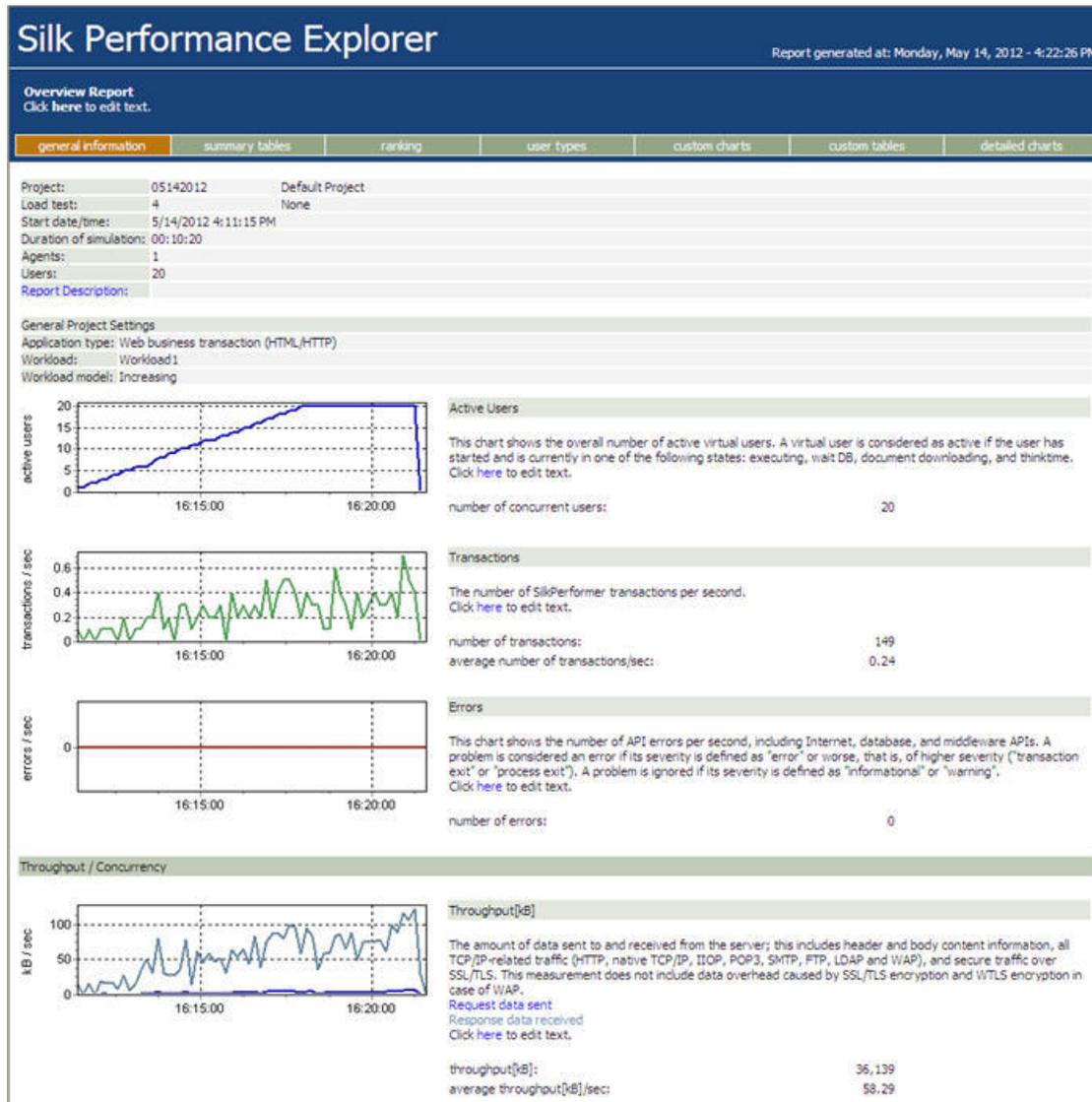
This section contains graphs that you have added manually. You can add charts to and remove charts from this section at any time. You can save your changes as a template to be displayed for every summary report.

Custom tables

This section contains tables that you have added manually. You can add tables to and remove tables from this section at any time. You can save your changes as a template to be displayed for every summary report.

Detailed charts

This section provides enlarged versions of the charts included in the report. Click a reduced version of a chart to jump to the enlarged version, and vice versa.



Viewing Overview Reports

1. Click the **Explore Results** button on the workflow bar. The **Workflow - Explore Results** dialog appears.

2. Click the **Silk Performance Explorer** button or link. If you have selected the **Generate overview report automatically** option in the **Settings/Options/Reporting** dialog, Performance Explorer opens and displays an overview summary report for the most recent load test. Additionally, you can choose to use a previously stored template for the generation of an overview report in this dialog. This setting is a global setting and will be used with Performance Explorer, regardless of the workload project you are using.
3. If the overview report does not appear automatically, click the **Overview Report** button on the workflow bar.
 - a) Navigate to the directory of the load test you are working with and select the corresponding `.tsd` file and then click **Open**.
 - b) Click **Next**.
 - c) Optionally, in the **Template** field, click [...] to navigate to the template file that you want to use.
 - d) Click **Finish**.
 - e) Depending on the load test that you selected, you may be prompted to confirm that you want to load all relevant files into your project.

The overview report opens.

Index

A

- adding
 - profiles 20
- agents
 - monitoring during Web testing 24
- AJAX
 - sample Web 2.0 application 4
- all day workload model 23

B

- browser
 - Web testing 20
- browser-driven Web testing
 - sample Web 2.0 application 4
- browsers
 - emulation 21
 - settings 20
 - simulation 21

C

- connections
 - concurrent 20
- cookies 19
- CORBA 28
- creating a test script
 - for Web testing 6
- customizing
 - user data 10
 - user-input data 10
- customizing test scripts
 - Web testing 10

D

- Data Source Wizard 25
- defining a project
 - for Web testing 5
- defining workload
 - Web testing 23
- digest verification 14
- dynamic workload model 23

G

- graphs
 - Web testing 25

I

- increasing workload model 22

J

- JSON
 - pretty format 12, 13
 - understanding 12

M

- monitoring
 - configuring 21, 22
 - servers 21, 22
- monitoring tests
 - Web testing 24

O

- overview page
 - Web testing 9
- overview reports
 - Web testing 28, 29

P

- page statistics
 - Web testing 9
- parameters
 - random variables 10
- parsing functions
 - HTML context parsing 17
 - overview 16
 - response data parsing 17
- Performance Explorer
 - Web testing 27
- pop-up window support
 - sample Web 2.0 application 4
- profiles
 - adding 20

Q

- queuing workload model 23

R

- recording a test script
 - for Web testing 6
- replay
 - browser-simulation settings 21
- response data parsing 17
- running tests
 - Web testing 24

S

- session handling
 - customizing 20
 - session information 17
 - when to use 18
- settings
 - browsers 20
- simulation
 - browsers 21
- steady state workload model 23

- summary reports
 - Web testing 8
- syncing record/replay TrueLogs 10

T

- test results
 - Web testing 26
- test scripts
 - visual analysis with TrueLog Explorer 7
 - Web testing 7
- title verification 14
- TrueLog
 - analyzing test runs 8
 - comparing Web record and replay 9, 10
 - finding errors in 9
- TrueLog Explorer
 - XML 12
- TrueLog On Error
 - Web testing 26, 27
- Try Script runs
 - for Web testing 6
- Tuxedo 28

U

- user data
 - customizing 10
- user profiles
 - Web testing 20

V

- variables
 - input attributes 10
- verification checks
 - automatically generating during recording 13
 - digest verification 14
 - enabling during replay 14
 - overview 13
 - title verification 14
- verification workload model 23
- verifying content
 - Web testing 15

W

- Web 2.0 testing
 - sample AJAX-based application 4
- Web testing
 - adjusting workload 22
 - analyzing test runs 8
 - analyzing test scripts 7
 - browser types 20
 - comparing record and replay TrueLogs 9, 10
 - content verification 15
 - creating a test script 6
 - customizing test scripts 10
 - customizing user data 10
 - defining a project 5
 - defining a workload 23
 - finding errors in TrueLog 9
 - monitoring a specific agent 25
 - monitoring a virtual user 25
 - monitoring all agent computers 24
 - monitoring tests 24
 - monitoring performance with a graph 25
 - overview page 9
 - overview reports 28, 29
 - page statistics 9
 - Performance Explorer 27
 - recording a test script 6
 - running tests 24
 - summary reports 8
 - syncing record/replay TrueLogs 10
 - test results 26
 - TrueLog On Error 26, 27
 - Try Script runs 6
 - user profiles 20
 - virtual user summary reports 8
 - visual analysis with TrueLog Explorer 7
 - workload models 22
- workload
 - Web testing 22
- workload models
 - Web testing 22

X

- XML
 - pretty format 12, 13