Borland
A MICRO FOCUS COMPANY

Silk Performer 9.5

Web Load Testing Tutorial

Micro Focus
575 Anton Blvd., Suite 510
Costa Mesa, CA 92626

2012-11-14

Contents

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

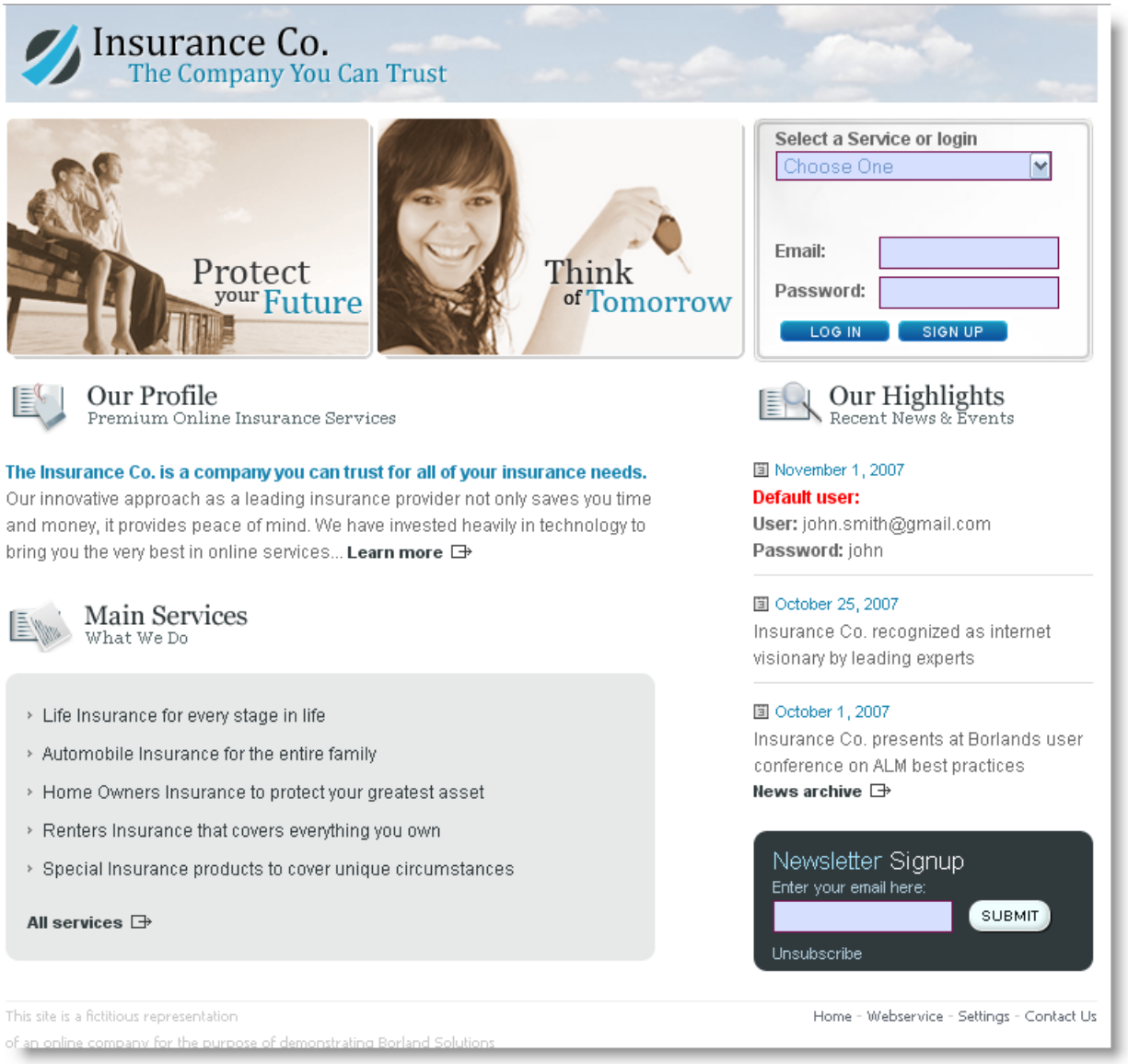
The fastest and easiest approach to test today's modern Web applications is to test them on the protocol level (HTML/HTTP). This approach generates simple scripts that incorporate advanced functionality. When testing a Web application with built-in AJAX logic and testing on the protocol level has proved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



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. The test script is generated from the recorded traffic.
8. In the ensuing dialog, name your test script and specify where to save it. In the Silk Performer Recorder message that appears, confirm that you want to close the Recorder.

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

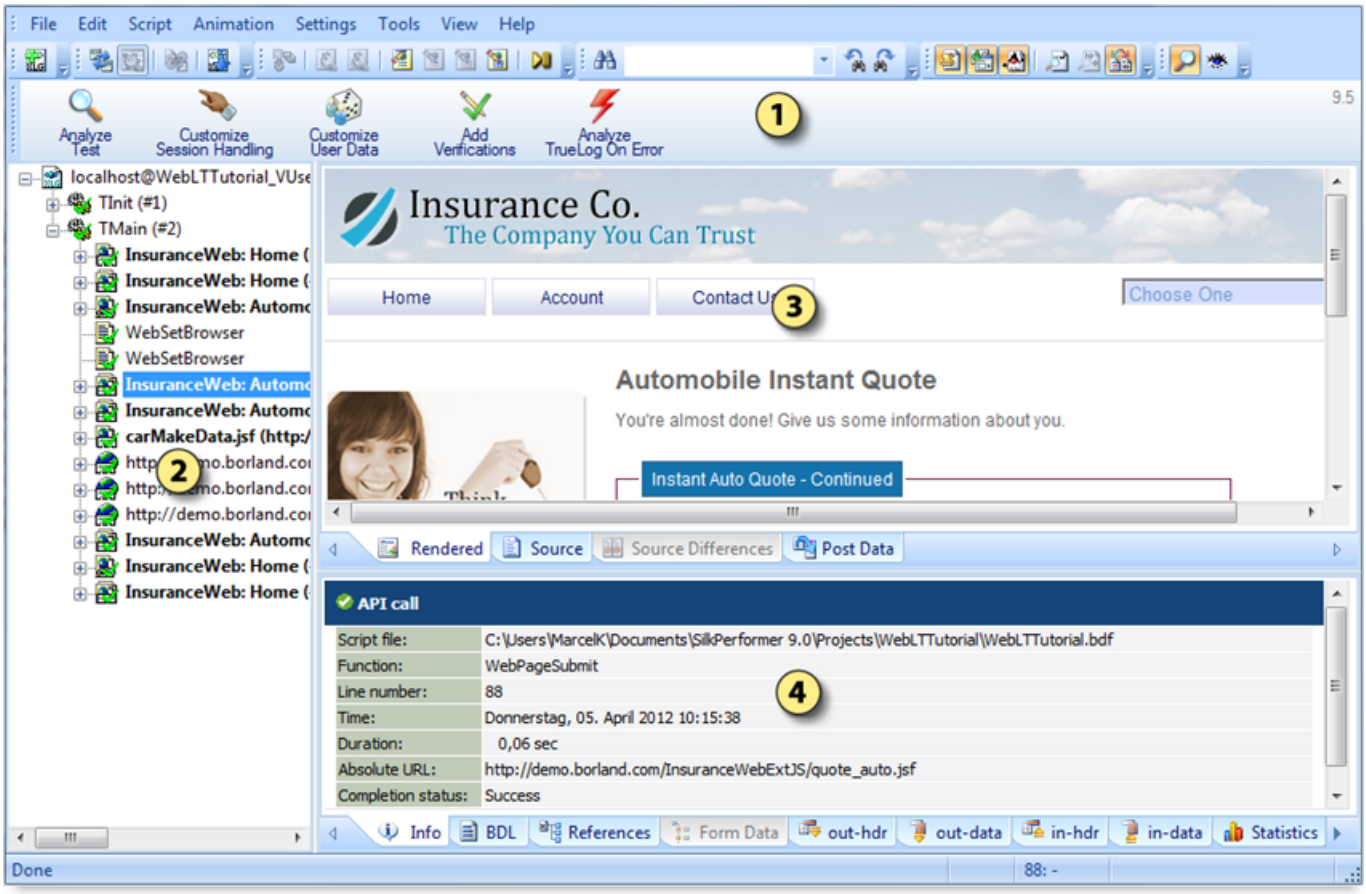
One of TrueLog Explorer's most powerful features is its ability to visually render Web content that is displayed by applications under test. In effect, it shows you what virtual users see when they interact with an application.

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**.



1 **Workflow bar**

2 **API node tree menu**

3 **Content pane**

4 **Information pane**

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 IIOP, 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. The submitted form's controls are highlighted; a green border around a control indicates that the submitted value is the same as the initial value (and was not changed by the user during recording). A red border indicates that the submitted value differs from the initial value (and was altered during recording). 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. Click **Settings** at the bottom right-hand corner of the page to access this feature.

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.

Content Verification

TrueLog Explorer enables you to insert content-verification functions into test scripts to verify the accuracy of content that is returned by application servers during testing.

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.

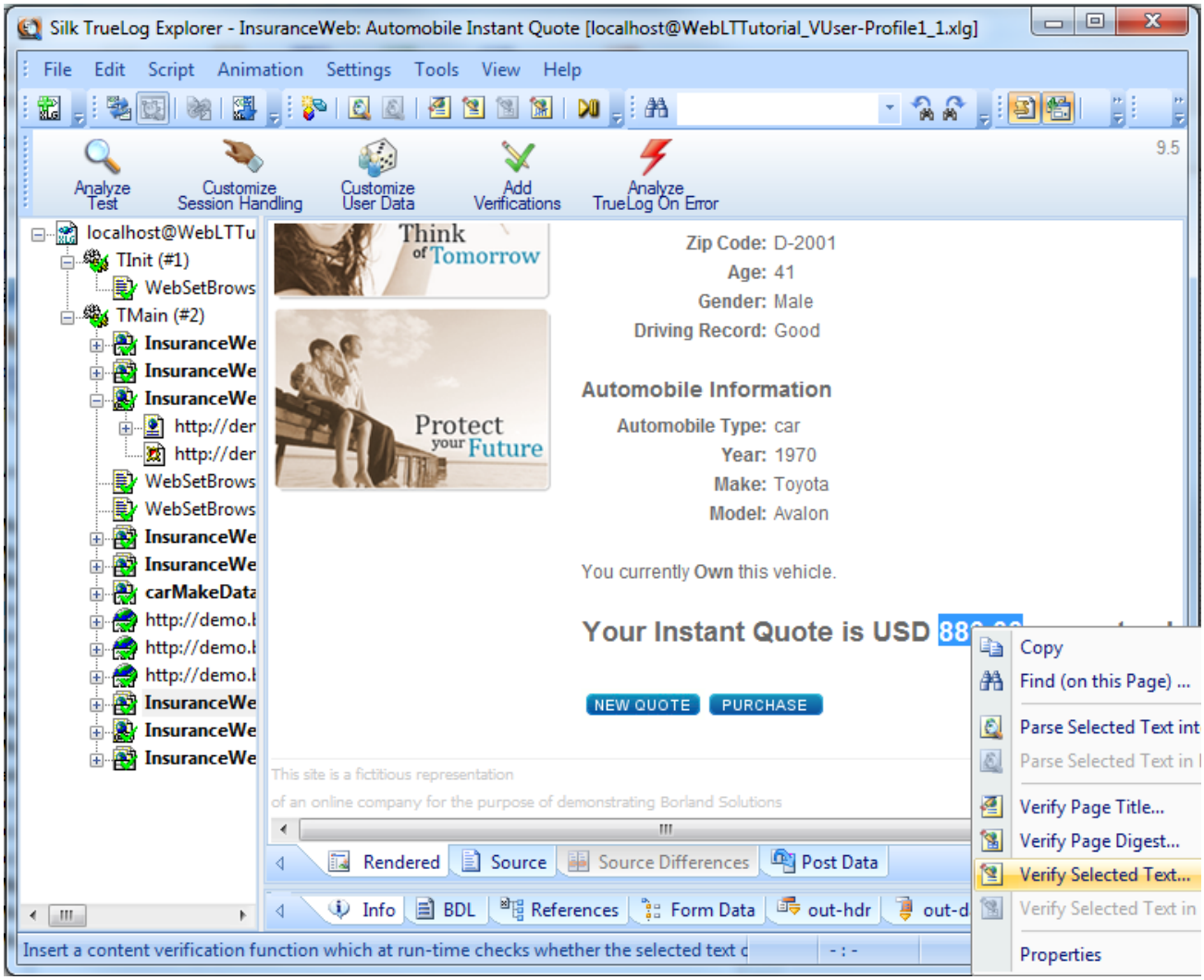
Note: This step is not required for page-title and page-digest verification functions.

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.



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 menu 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 menu 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.

Identifying Baseline Performance

A baseline performance test enables you to determine your application's ideal performance baseline.

Note: The workflow bar displays the simplified workflow by default. If you want to create a performance baseline to set response time thresholds, enable the full workflow bar by right-clicking on the **Simple Workflow** area on the right-side of the workflow bar and selecting **Show Full Workflow Bar**.

With a baseline test, a customized test is run with just one user per user type, and the results from this unstressed performance of the application form the basis for calculating the number of concurrent users per user type and for setting the appropriate boundaries for the HTML page response times and transaction response times. Additionally, the required bandwidth for running the load test is calculated from the baseline results. The measurements typical of a real load test are used, and report and output files are generated.

A secondary function of baseline tests is to serve as a trial run of your test script. A trial run can help you verify the following:

- Customizations have not introduced new errors into the script
- The script can accurately and fully reproduce the interaction that is intended between the client application and the server

Note: Baseline tests ignore the **Random thinking time** option. This enables you to compare the results of different baseline tests more effectively because the think times remain constant between tests.

For baseline runs, the following settings are set automatically:

- A **Baseline report file** is automatically created.
- The **Stop virtual users after simulation time (Queuing Workload)** option is enabled.
- The **Random thinking time** option is disabled.
- The **Loadtest description** text box is set to `Baseline Test`.
- The **Display All Errors Of All Users** option in the Monitor window is enabled.
- The **Virtual user output files (.wrt)** option is enabled.
- The **Virtual user report files (.rpt)** option is enabled.

Running a Baseline Test

1. Click **Find Baseline** on the workflow bar. The **Workflow - Find Baseline** dialog box opens.
2. Using the check boxes on the left side of the dialog box, check the **User Type** that you want to run in the actual baseline test.

   A *user type* is a combination of a script, a usergroup, and a profile. One virtual user from each user type is executed.

3. *(Optional)* Click **Add** to create a new user type. The **Add User Type** dialog box opens. The active profile is selected as the default profile in the **Profile** list box unless you have configured another profile. Select a combination of **Script**, **Usergroup**, and **Profile** objects to add to the new user type and click **OK**.
4. *(Optional)* Click **Settings** to configure simulation settings.
5. Click **Run** to perform the baseline test.

Validating Baseline Test Results

You must confirm that the test baseline established by a baseline test actually reflects the desired performance of the application under test. Complete this task by inspecting the results of a baseline test in a baseline report. If the results are satisfactory, the baseline can be accepted and can serve as the basis for calculating the number of concurrent virtual users and the thresholds of specific timers in future tests. The baseline results are used for comparison with the results of actual load tests.

Baseline confirmation also provides an opportunity to inspect the measurements generated during a baseline test to ensure that the final load test contains all required results.

Server monitoring is set up for baseline confirmation so that live server-monitoring can occur during load tests and that server-side results information can be generated for analysis and correlation after the test.

Understanding Baseline Reports

A baseline report consists of the following sections:

- General information
- User types
    - Summary tables
    - Transaction response times
    - HTML page timers
    - Web form measurements
    - Accept Results button

General Information

The **General Information** section includes administrative information in tabular form.

Administrative information includes the Silk Performer Version information, the project name, a description of the project, the date and time of the baseline test, the workload definition, the workload model, and the number of errors that occurred.

User Types

For each user type involved in the baseline test run, a separate section is available with details on the measured response times. The following information appears in the summary line:

- Number of virtual users (one for a baseline test)
- Test duration
- Session time
- Session busy time
- Average page time
- The following numbers:
    - Transactions that were executed successfully
    - Cancelled transactions
    - Failed transactions
    - Errors

The session time consists of the execution time of all transactions defined in the `dcluser` section of a test script, without the initial and end transactions. The session busy time is calculated as the session time minus think times, as the following example shows.

```
dcluser
  user
    Vuser1
  transactions
    TInit  : begin;
    T1     : 1;
    T2     : 3;
    Tend   : end;
```

The session time is the average response time of T1 + 3 * average response time of T2. The session busy time is the same without any think time.

If you are satisfied with the results, you can accept them as baseline results and save them for further processing, such as the calculation of the number of concurrent virtual users and the network bandwidth needed for the load test.

For each user type, detailed results are available in the following sections:

- **Summary tables** – Contains aggregate measurements of all virtual users in tabular form. The first table provides general information, such as the number of transactions that were executed and the number of errors that occurred. The remaining tables provide summary information that is relevant to the tested application type.
- **Transactions** – Displays the following aggregate measurements in tabular form for all transactions of a specific user type:
  - Transaction response time – Measured from the beginning to the end of the transaction, including all think times.
  - Transaction busy time – The transaction response time without any think time.
- **Page timers** – Displays the following summary measurements in tabular form for every accessed Web page:
  - Download time of the entire page
  - HTML-document download time
  - Server busy time
  - Amount of data downloaded with the page
  - Amount of data downloaded for embedded objects
- **Web forms** – Displays the following measurements in tabular form for every used Web form:
  - Roundtrip time
  - Server busy time
  - Number of HTTP hits
  - Amount of request data that was sent
  - Amount of response data that was received

If the results of an inspected load test are acceptable for use as a baseline, you can store them for further calculations and processing by clicking **Accept Results**. The results of tests with all user types involved are stored in the file `<projectdir>\BaselineResults\baselineReport_<workloadname>.brp`. You must define a separate baseline for every workload that is defined in your load test project. If you create a copy of a workload, the accepted baseline results are also copied.

Confirming Baseline Test Results

1. Click **Confirm Baseline** on the workflow bar. The **Workflow - Confirm Baseline** dialog box appears.
2. Click **Baseline Report**.

   The Baseline report appears, showing the results of the previous test run.

3. Accept the results of the test as your baseline by clicking **Accept Baseline** in the report.
4. Click **Yes** to confirm.

   If the baseline test did not identify the correct baseline performance for your application or if any errors occurred during the baseline test, click **No**. Then click **Customize Test** to refine your test script.

Accepted Baseline Report

Use the results from this report to generate thresholds for the following values:

- Accessed Web pages
- Transaction response times
- Custom timers

The overview report displays these thresholds at the end of a load test. With this information, you can easily verify whether the response times of a test are acceptable.

You can also use the Accepted Baseline report to calculate the number of concurrent virtual users and to view the network bandwidth needed for a test. Silk Performer stores Accepted Baseline reports in the file `<projectdir>\BaselineResults\baselineReport_<workloadname>.brp`.

If you define multiple workloads in your load test project, you must define a separate baseline for each workload. If you create a copy of a workload, the accepted baseline results are also copied.

Viewing these reports can be useful for verifying the accepted results.

Viewing the Accepted Baseline Report

1. Click **Confirm Baseline** on the workflow bar. The **Workflow - Confirm Baseline** dialog box appears.
2. Click **Accepted Baseline Report**. A report showing all accepted baseline results appears.

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 workload model – 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 workload model – 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 workload model – 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, and you must bring the test to an end 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 workload model – 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 workload model – 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 workload model – 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.

1. Click **Adjust Workload** on the workflow bar. The **Workflow - Select Workload Model** dialog box appears.
2. Select one of the following workload models by clicking the appropriate option button:
   - Increasing
   - Steady State
   - Dynamic
   - All Day
   - Queueing
   - 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 deterimine 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*.

Accessing Results Files

1. Click the **Explore Results** button on the workflow bar. The **Workflow - Explore Results** dialog appears.
2. To view a graphic display of the results from your load test, click **Silk Performance Explorer**.
3. If the previous load test produced errors and the **TrueLog On Error** option was enabled on the **Workflow - Workload Configuration** dialog box when the test was ran, you can inspect the errors by clicking **Silk TrueLog Explorer** and use TrueLog Explorer to analyze the errors.
4. To view a report of the results from your load test, click **Baseline Report**.

   In such **Summary reports**, you can accept the results of any previous test as your baseline.
5. To view accepted baseline results, click **Accepted Baseline Report**.
6. *(if enabled)* To view virtual user report files, click the **View Virtual User Report Files** link.

   This report contains the simulation results for each virtual user.

   Note: To enable virtual user report data and this results link, click the **Generate Virtual User Report Files** button before running your test.
7. *(if enabled)* To view virtual user output files, click the **View Virtual User Output Files** link.

   Output files contain the output of write statements used in the test script.

   Note: To enable virtual user output data and this results link, click the **Generate Virtual User Output Files** button before running your test.
8. Click **Next** to go to the **Workflow - Reuse Project** dialog box.

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.

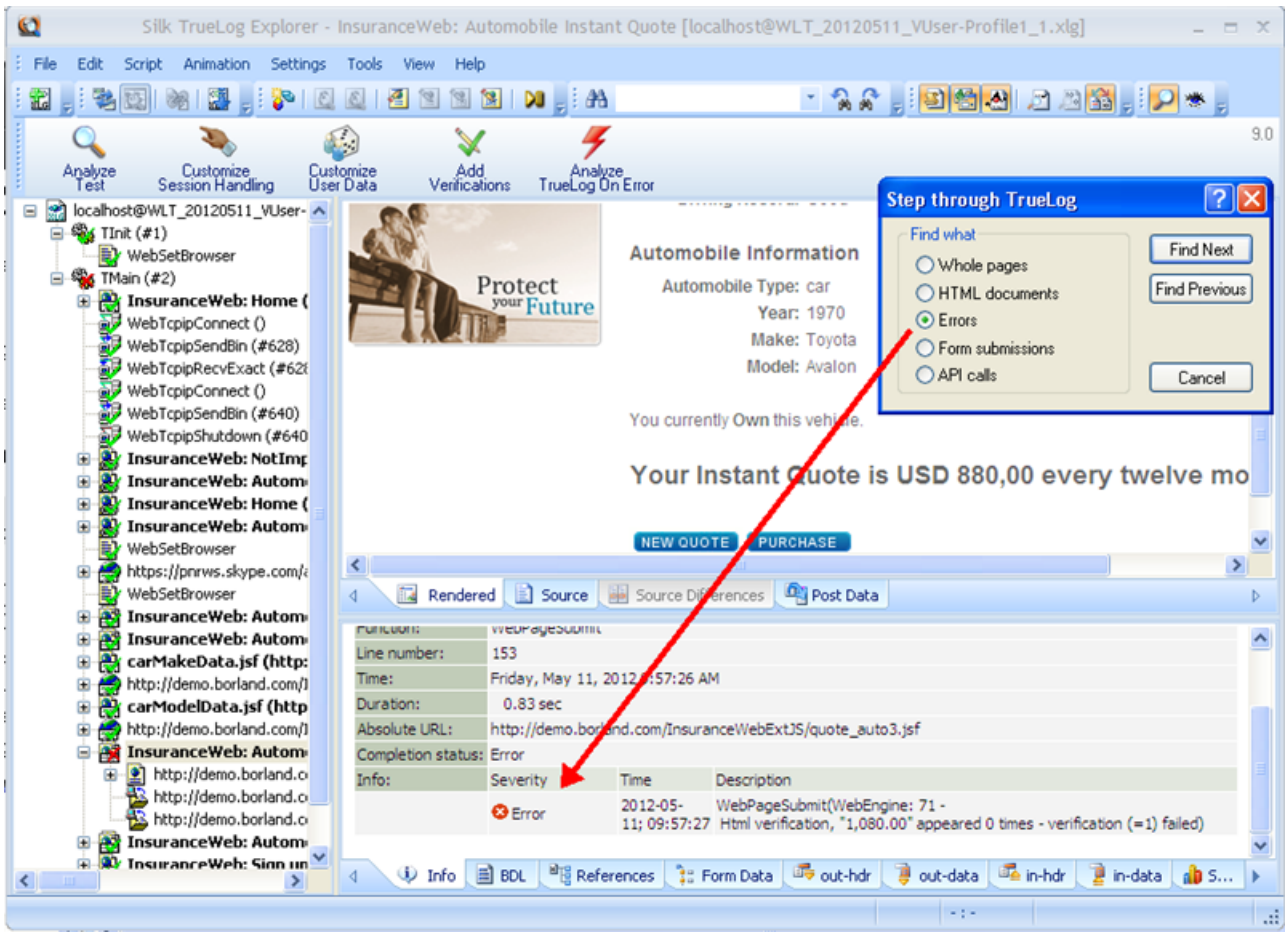
TrueLog Explorer provides a view into Silk Performer verification-under-load capabilities with the following features:

- 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

With Performance Explorer real-time monitoring, live charts provide a customizable display of the most relevant performance information from the target server. Monitoring is available for a comprehensive collection of Web servers, application servers, and database servers. You can open multiple charts at the same time allowing you to watch a graphic display of Web server performance and operating system performance simultaneously. A menu tree editor with drag functionality allows you to combine elements from any data source into charts.

After a test, you can chart the performance of the target server from both the client side and the server side. Response-time measurements display the client perspective, while throughput data displays the perspective from the server side. Charts and graphs are fully customizable, and they can contain as many or as few of the measurements taken during the test as you require. You can open multiple charts, using information from one or multiple tests, at once to facilitate contrast and compare operations. Templates for the most typical test scenarios, such as Web, Database, IIOP, are provided. You can populate these default charts easily and quickly with the data you need. Drag functionality enables chart elements to be combined from any data source. You can place information on client response times and server performance in a single chart so that you can directly view the effect of server performance on client behavior.

Using the advanced drag functionality of the menu tree editor, you can combine information elements from any data source in any number of selected charts. You can even add and remove measurements from different sources to produce permanent or temporary charts and reports that suit your needs. You can save modified settings with each project to ensure that Performance Explorer always opens with your preferred view.

Performance Explorer also offers a comprehensive XML/XSL-based overview report that combines user group data from the baseline report file (`baseline.brp`) with time-series, test-result information. You can customize the overview report to your needs by changing the provided texts and by inserting charts of your interest. Saving your changes as a template allows the reuse of your individual settings.

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

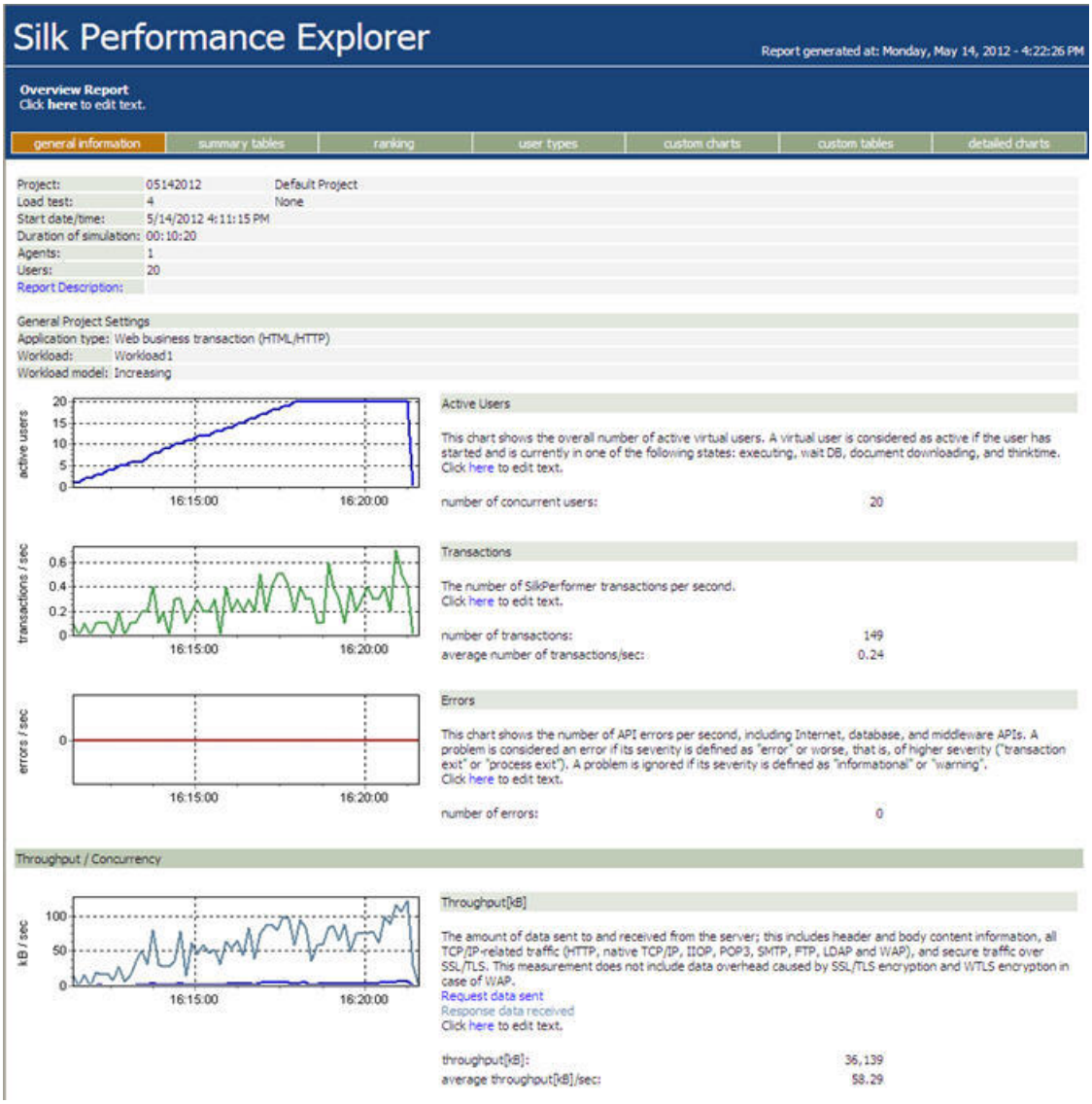
This section contains graphs that you have added manually. You can add charts to and remove charts from this section at any time. You can save your changes as a template to be displayed for every summary report.

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

T

U

V

W

X