Borland
A MICRO FOCUS COMPANY

Silk TrueLog Explorer 9.5

ヘルプ

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

2012-11-20

目次

TrueLog Explorer 9.5

TrueLog Explorer 9.5 へようこそ。

TrueLog Explorer では、テスト スクリプトをカスタマイズしたり、返されたテスト結果を徹底的に分析したりするためのフレームワークを提供することで、 Silk Performer のテスト作業をサポートします。

TrueLog Explorer では、数々のツールやレポートが用意されているため、スクリプトに自動コンテンツ検証を組み込んだり、静的セッション データの解析関数を挿入したり、TrueLog On Error 機能でエラーの根本原因を特定することが簡単にできます。

入門

このセクションでは、TrueLog Explorer、TrueLog ファイル、テストの分析、サンプル アプリケーションについて、概要を説明します。

TrueLog Explorer の概要

TrueLog Explorer では、テスト スクリプトをカスタマイズしたり、返されたテスト結果を徹底的に分析したりするためのフレームワークを提供することで、 Silk Performer のテスト作業をサポートします。

TrueLog Explorer では、数々のツールやレポートが用意されているため、スクリプトに自動コンテンツ検証を組み込んだり、静的セッション データの解析関数を挿入したり、TrueLog On Error 機能でエラーの根本原因を特定することが簡単にできます。

組み込みの負荷テスト方法論

TrueLog Explorer は負荷テスト スクリプトのカスタマイズやエラー分析のベスト プラクティスにもとづいて設計されており、セッション処理のカスタマイズやコンテンツ検証から、ユーザー データのカスタマイズ、最終的にエラー分析へと進めるようになっています。 TrueLog Explorer のワークフロー バーを使ってこの方法論に沿って一歩ずつ進むことで、セッション処理や検証チェックや入力データのパラメータ化などによりテストを強化することができます。 最終的には、ワークフロー バーから TrueLog On Error 分析を実行することができます。

注: このヘルプ システムに含まれる例のほとんどは、特に Web アプリケーションに関するものです。 しかし、そこで説明している概念の多くは、データベース アプリケーションや、Citrix アプリケーション、Oracle Forms アプリケーション、SAPGUI アプリケーション、AJAX 対応の Web アプリケーション、ターミナル エミュレーション画面アプリケーション、XML アプリケーションにも適用できます。 また、その概念の多くは、POP3 や SMTP やカスタム プロトコルなど、UDP や TCP/IP をベースとしたアプリケーションのテストにも適用できます。 その概念を適用する際、TrueLog 分析は利用可能ですが、スクリプトのカスタマイズはできません。

TrueLog Explorer でできること

TrueLog Explorer を使って次の作業を行うことができます。

- **再生エラーの迅速かつ容易な検索**- TrueLog の比較モードを使うと、記録セッションと再生セッションの違いを自動的に検出して、アプリケーションが意図したとおりに機能しているかを判断することができます。 記録セッションも再生セッションも視覚的に表示されるため、両者を並べてその違いを比較したり、ユーザーに対して表示されたのと同じエラーを見ることができます。
- **セッション処理のカスタマイズ**– TrueLog Explorer の解析関数を使うと、スクリプト内の静的セッション ID を動的セッション ID に置き換え、それによって、成功した負荷テスト実行の状態情報を保守する

ことができます。 これらの関数は、Web アプリケーション、データベース アプリケーション、XML アプリケーションで有効です。

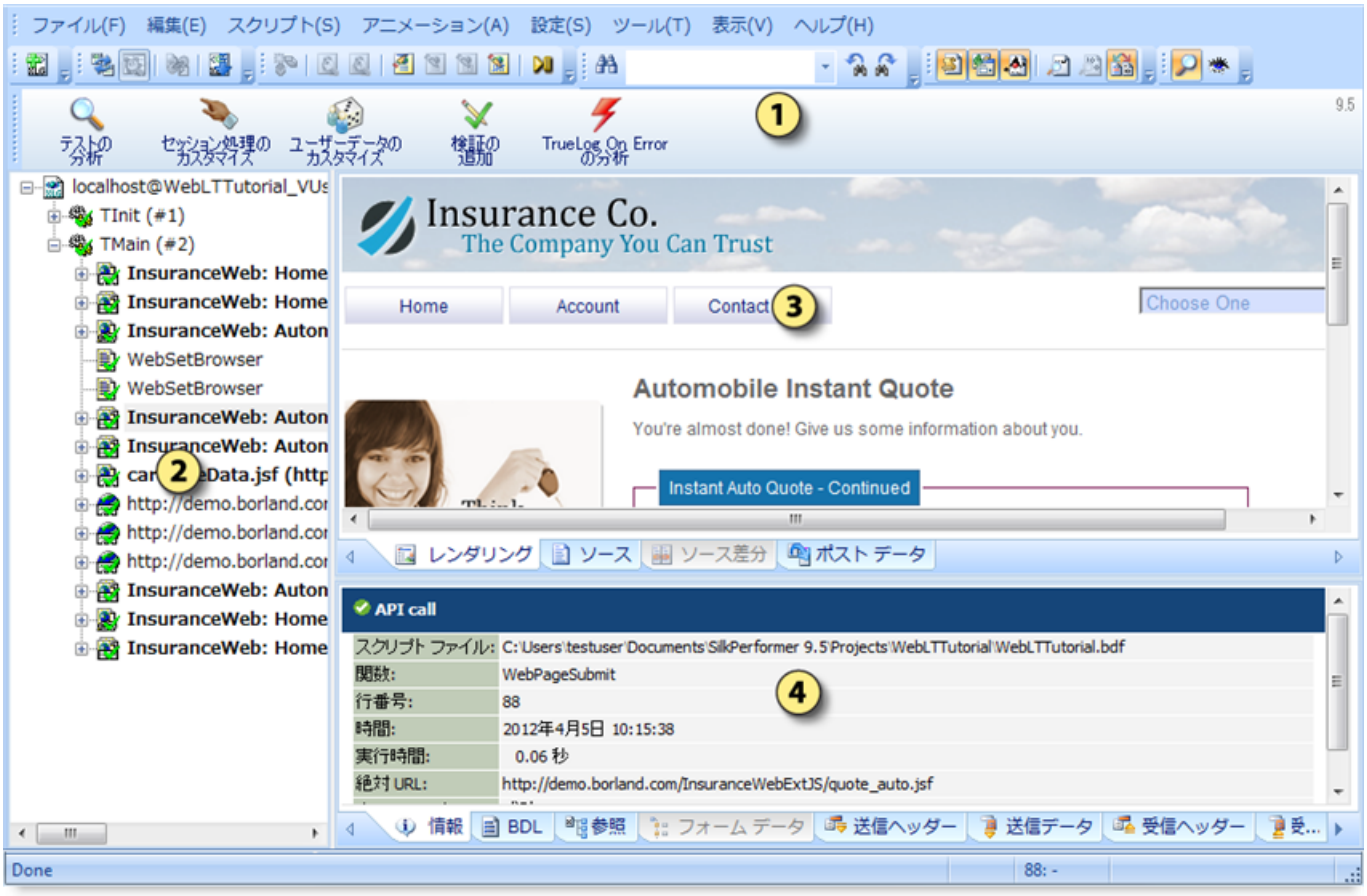
注: Silk Performer の記録技術は、記録されたテスト実行に固有のセッション データを使用するスクリプトを自動生成します。 スクリプト自体には静的セッション情報を含みません。 そのため、ほとんどのアプリケーションでは、通常、セッション処理をカスタマイズする必要がありません。

- **入力データのパラメータ化**- ユーザー データのカスタマイズでは、記録された静的ユーザー入力データを、トランザクションごとに変化するパラメータ化された動的ユーザー データに置き換えることによって、現実に近いテスト スクリプトを作成できます。 このようなデータ駆動型テストを実行するために手動でスクリプトを作成する必要はありません。 この機能は、Web アプリケーション、データベース アプリケーション、XML アプリケーション、Citrix アプリケーション、SAPGUI アプリケーション、ターミナル エミュレーション アプリケーション、Oracle Forms アプリケーションで有効です。
- **テスト スクリプトへの検証の追加**- **検証の追加** ツールを使うと、テスト中にダウンロードされたデータの内容を確認することができ、サーバーの送信したコンテンツをクライアントが受信していることを検証できます。 検証は、システム開発後も、稼働中のパフォーマンス管理に役立ちます。 この機能は、Web アプリケーション、データベース アプリケーション、XML アプリケーション、SAPGUI アプリケーション、Citrix アプリケーション、ターミナル エミュレーション アプリケーション、Oracle Forms アプリケーションで有効です。 TrueLog Explorer は、Citrix サーバーをホストとするアプリケーションのビットマップとウィンドウの検証をサポートしています。
- **TrueLog On Error の分析**- TrueLog On Error ファイルは、テスト中に発生したすべてのエラーのエラー履歴を完全に記録しているため、根本原因の分析に非常に役立ちます。 TrueLog を用いると、現実のコンテンツを詳しく調べてエラー状況を分析することができます。 TrueLog は、エラーが発生したセッションのコンテキストにおけるエラーを表示するため、テスト スクリプトと密接に統合されています。
- **データベース操作の視覚化とカスタマイズ**- SQL クエリなどのデータベース操作のカスタマイズは、HTML 操作のカスタマイズと同様に行うことができます。 結果セット データに検証を適用することも可能です。
- **XML の視覚化とカスタマイズ**– TrueLog Explorer には直感的な XML ビューアが含まれていて、SOAP のリクエストやレスポンスといった XML ベースのデータをすべて詳しく調査することができます。 XML データのカスタマイズや視覚的検証を行うこともできます。
- **Citrix の視覚化とカスタマイズ**- Citrix 対応アプリケーションでは、マウスの移動、マウスのクリック、キーボード入力、およびウィンドウの作成/破棄といった画面上のイベントを、記録、再生、およびカスタマイズすることができます。 検証用に画面同期関数も提供されています。
- **ターミナル エミュレーションの視覚化とカスタマイズ**- VT100+、IBM 3270、および IBM 5250 というターミナル エミュレーション プロトコル用のテスト スクリプトを、記録、再生、およびカスタマイズすることができます。 テスト スクリプトをカスタマイズして、値解析や、ステータス値解析、値検証、ステータス値検証を含めることができます。
- **TCP/IP および UDP テスト結果の分析**- POP3、SMTP、カスタム プロトコルなどの UDP や TCP/IP ベースのプロトコルを使用するアプリケーションについて、記録 TrueLog と再生 TrueLog を比較し、セッションに依存する情報を特定したりエラーを分析することができます。
- **GUI レベルのテスト結果の分析**- リモート デスクトップ セッションにおける GUI レベル (ファット クライアント) アプリケーションのテストの記録 TrueLog と再生 TrueLog。

UI のツアー

TrueLog Explorer のインターフェイスについて紹介します。このインターフェイスをよく理解していると、テスト スクリプトのカスタマイズや検証の作業を簡単に管理、実行できます。

以下では、TrueLog Explorer のインターフェイスの主要セクションを取り上げて説明します。



1 ワークフロー バー
2 API ノード ツリー メニュー
3 コンテンツ ペイン
4 情報ペイン

メイン メニュー

メイン メニューから、TrueLog Explorer の結果分析機能やスクリプト カスタマイズ機能にアクセスすることができます。

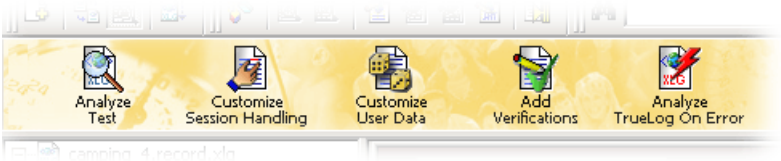
注: 開いている TrueLog のプロトコルのタイプによって、メニュー項目の一部が使用できないことがあります。

ツールバー

ツールバーから、よく使用する機能にアクセスすることができます。

注: 開いている TrueLog のプロトコルのタイプによって、ツールバーの一部のボタンが使用できないことがあります。

ワークフロー バー



ワークフロー バーは、テスト実行結果の分析やスクリプトのカスタマイズに関する主要な操作を行いやすくするためのものです。 ワークフロー バーは、TrueLog Explorer に組み込まれている負荷テスト方法論を元に、5 つの主要な作業をサポートしています。

- テスト実行の分析
- テスト スクリプト内のセッション処理のカスタマイズ
- スクリプトへの検証関数の追加
- TrueLog On Error の分析

API ノード ツリー メニュー

インターフェイスの左側にあるメニュー ツリーでは、ダブル クリックによって TrueLog API ノードを展開したり折りたたんだりすることができます。 ノードをクリックすると、そのコンテンツが [コンテンツ] ペインに、履歴の詳細が **情報** ページに表示されます。

TrueLog に含まれる Web ページの詳細を表示するには、メニュー ツリーでそのページの URL またはリンク記述を選択します。 すると、コンテンツが [コンテンツ] ペインに、プロパティが **情報** ページに表示されます。

[コンテンツ] ペイン

TrueLog Explorer は、受信したすべてのデータに対して複数のビューを提供します。 テスト対象のアプリケーションの種類 (Web、XML、データベース、TCP/IP、UDP、Oracle Forms、SAPGUI、Citrix など) によって、[コンテンツ] ペインの表示オプションは変化します。

レンダリング *(HTML および XML の場合のみ)*

レンダリング ページには、メニュー ツリーで選択した API ノードに対するサーバーからのレスポンスが、視覚的にレンダリングされて表示されます。 HTML レスポンスは、**レンダリング** ページの背後で実行されている Internet Explorer コントロールによってレンダリングされます。 XML レスポンスは、XML データをメニュー ツリー形式で表示する XML コントロールによって表示されます。 HTML でも XML でもないレスポンスはレンダリングされないので、**ソース** ページまたは **受信データ** ページを使って表示する必要があります。 XML コントロール内で要素や属性を右クリックすると、メニュー ツリーを 1 レベル以上展開できます。

SQL コマンド *(データベース アプリケーションの場合のみ)*

SQL コマンド ページには、入力や結果データなど、選択したデータベース呼び出しに関連する SQL 文が表示されます。 他のプロトコルの **レンダリング** ページと同じです。

ソース *(HTML、XML、TCP/IP、UDP の場合のみ)*

ソース データをそのまま表示します。 **ソース** ページには、Web コンテンツを生成するのに使われた HTML コードが表示されます。 ここには、**受信データ** ページと同じ情報が含まれます。 **ソース** ページと **送信データ** ページを使うことで、リクエストの受信データと送信データを同時に表示することができます。

ソース差分 *(HTTP ベースのアプリケーションの場合のみ)*

ソース差分 ページには、選択したノード間の受信データ (ソース) の差分が一覧表示されます。 このページは、サーバー レスポンス内の動的情報を検索するのに役立ちます。 セッション情報を探すには、まずここを検索します。

注: ソース差分 ページが利用可能になるのは 差分モードのときのみです。

ポスト データ *(HTML および XML の場合のみ)*

ポスト データ ページには、出力データ (アプリケーションからサーバーに送られたデータ) がレンダリングされて表示されます。 XML の場合は、XML のメニュー ツリー形式で表示され

ます。 HTML の場合、ポストされたデータ (HTTP-POST コマンドの POST データまたは HTTP-GET コマンドのクエリ文字列) は、そのデータに対応する HTML フォームを含む HTML ページ内に表示されます。 コンテキストを持つ HTML フォーム データの送信 (WebPageSubmit で送信された HTML フォーム データ。WebPageForm 経由の送信は含まない) だけが **ポスト データ** ページに表示されます。 要素は赤で、 属性は青緑で表示されます。

レンダリング ビューのコンテンツを検索するには、レンダリング ビューで右クリックして **検索** を選択します。 グローバル検索オプションでは、レンダリング ビューを検索できません。

注: レンダリング ページと **ポスト データ** ページは、異なる API ノードのコンテンツを同時に表示することがあります。 **レンダリング** ページにその前の Web ページで送信された HTML-FORM の結果が表示されている場合、**ポスト データ** ページには送信された値 (マウスを置くと表示される) を含む Web ページが表示されます。

フォーム コントロール

(Oracle Forms の場合のみ)

フォーム コントロール ページには、Oracle Forms コントロールのプロパティ (状態と内容) が表示されます。 比較モードの場合は、このページで記録と再生のコントロールの状態を比較できます。

画面

(Citrix の場合のみ)

画面 ページには、レンダリングされたアプリケーション GUI、ウィンドウ同期関数、ユーザー入力値が表示されます。

ウィンドウ

(Citrix の場合のみ)

現在の関数のウィンドウ パラメータによって参照されているウィンドウが表示されます。

開始リクエスト

(SAPGUI の場合のみ)

開始リクエスト ページには、サーバーへの各ラウンドトリップの前の SAPGUI アプリケーションの状態が表示されます。

終了リクエスト

(SAPGUI の場合のみ)

終了リクエスト ページには、サーバーへの各ラウンドトリップの後の SAPGUI アプリケーションの状態が表示されます。

ホスト画面

(ターミナル エミュレーション アプリケーションの場合のみ)

ホスト画面 ページには、ターミナル ビュー内で選択した API ノードのサーバーからのレスポンスを視覚的にレンダリングしたものが表示されます。

情報ペイン

情報 ペインでは、BDL スクリプト、HTTP ヘッダー、タイミング統計など、テスト スクリプトおよびテスト実行に関するデータが表示されます。 テスト対象のアプリケーションの種類 (Web、XML、データベース、Oracle Forms、SAPGUI、Citrix、TCP/IP、UDP など) によって、**情報** ペインの表示内容は変化します。

情報

情報 ページには、開いている TrueLog ファイルと選択した API ノードについて、以下のような一般情報が表示されます。

- スクリプト ファイル名
- 関数
- 行番号
- 時間

- 実行時間
- 絶対 URL
- 完了ステータス
- 追加情報 (可能な場合)

BDL

BDL ページには、開いている TrueLog に対応する BDL スクリプトが表示されます。 この BDL スクリプトは、選択されている API ノードの行まで自動的に移動して表示されます。

参照 *(HTML の場合のみ)*

このページは、HTML データを返すノードの場合にのみ有効になります。 **参照** ページには、選択されている HTML の簡単な DOM (Document Object Model) が表示されます。 このページには次の HTML 要素が含まれます。

- ハイパーリンク
- フレーム
- 埋め込みドキュメント
- フォーム

送信ヘッダー *(HTTP の場合のみ)*

送信ヘッダー ページには、アプリケーション (再生 TrueLog の場合は Silk Performer、記録 TrueLog の場合はブラウザ) がサーバーに送信する HTTP ヘッダーがそのまま表示されます。

受信ヘッダー *(HTTP の場合のみ)*

受信ヘッダー ページには、サーバーがアプリケーションに送信する HTTP ヘッダーがそのまま表示されます。

送信データ *(HTTP、Oracle Forms、Citrix、ターミナル エミュレーション、TCP/IP、UDP の場合のみ)*

HTTP の場合、このページには、アプリケーションからサーバーへ HTTP-POST コマンドで送られるデータが表示されます。 TCP/IP と UDP の場合は、アプリケーションからサーバーへ WebTcpipSend 関数で送られるデータが表示されます。 Citrix の場合は、データ文字列が表示されます。 Oracle Forms のログ レベルが 通常 および デバッグ の場合 (再生プロファイル設定で設定可能)、**送信データ** ページには、現在のアクション (ボタンをクリックするなど) で送信される Oracle Forms メッセージが表示されます。

データの表示方法は、16 進数表示とテキスト表示を切り替えることができます (**設定** > **オプション** > **表示** > **データ形式**)。 デフォルトでは、TrueLog Explorer によってそのデータに最適な表示形式が選択されます (画像の場合はバイナリ、HTML ドキュメントの場合はテキストなど)。

受信データ *(HTTP、Oracle Forms、TCP/IP、ターミナル エミュレーション、UDP の場合のみ)*

受信データ ページには、アプリケーションがサーバーから受信したデータが表示されます。 このデータは、サーバーから受信したそのままの形式で表示されます (HTML のレンダリングや XML のメニュー ツリー表示は行われません)。

統計 *(Web ビジネス トランザクション用 - HTTP アプリケーションのみ)*

統計 ページには、Web ページのタイミング統計が表示されます。 ここには、Web ページ コンポーネントや通信要素ごとのタイミング情報が含まれます。 コンポーネントごとの内訳も含めて、正確なレスポンス時間がグラフで表示されます。 これにより、エラーやページ ダウンロードの遅延の根本原因を突き止めることができます。 **統計** ページには、ページ コンポーネントごとに次のデータが表示されます。

- DNS 検索時間：渡されたドメイン/ホスト名から IP アドレスを解決するのに要した時間

- 接続時間：擬似ユーザーがサーバーに接続するのに要した時間
- SSL ハンドシェイク時間：サーバーとの間で暗号鍵を交換するのに要した時間
- データ送信時間：クライアント リクエストの最初のバイトからクライアント リクエストの最後のバイトまでの時間
- サーバー ビジー時間：クライアント リクエストの最後のバイトからサーバー レスポンスの最初のバイトまでの時間
- レスポンス受信時間：サーバー レスポンスの最初のバイトからサーバー レスポンスの最後のバイトまでの時間 (ドキュメントはすべて含むが埋め込みオブジェクトは含まない)
- キャッシュ統計値

統計 *(Web ブラウザ駆動用 - AJAX アプリケーションのみ)*

統計 ページには、Web ページのタイミング統計が表示されます。 ここには、Web ページ コンポーネントや通信要素ごとのタイミング情報が含まれます。 コンポーネントごとの内訳も含めて、正確なレスポンス時間がグラフで表示されます。 これにより、エラーやページ ダウンロードの遅延の根本原因を突き止めることができます。 **統計** ページには、ページ コンポーネントごとに次のデータが表示されます。

- DNS 検索時間：渡されたドメイン/ホスト名から IP アドレスを解決するのに要した時間
- 接続時間：擬似ユーザーがサーバーに接続するのに要した時間
- ネット ラウンドトリップ：クライアント リクエストの最初のバイトからサーバー レスポンスの最後のバイトまでの時間 (ドキュメントはすべて含むが埋め込みオブジェクトは含まない)
- キャッシュ統計値

フォーム データ *(HTTP および Oracle Forms の場合のみ)*

Oracle Forms 画面に含まれる、あるいは Web サーバーに送信される、カスタマイズ可能なすべてのコントロールを表示したり操作するのに便利なツールです。

コントロール *(SAPGUI の場合のみ)*

コントロール ページは、SAPGUI ウィンドウに含まれるカスタマイズ可能なコントロール値を表示したり操作するのに便利なツールです。

ホスト画面情報 *(ターミナル エミュレーション アプリケーションの場合のみ)*

ホスト画面情報 ページは、ホスト画面それぞれについて、ステータス (名前/値のペア) などの追加情報を記録します。 フィールド名を選択すると、上述の **ホスト画面** ウィンドウで対応するフィールドが選択されます。 一般的な画面情報 (カーソル位置、画面の寸法など) も表示されます。

TrueLog Explorer のベスト プラクティスの使用方法

TrueLog Explorer のベスト プラクティスの使用方法は、以下のステップから構成されます。

多くの場合、この一連のステップを 1 度通して実行するだけでは済みません。 これらのタスクは繰り返して行うものです。 たとえば、カスタマイズをし、結果を確認し、さらにカスタマイズをして、そのカスタマイズの結果を確認するなどです。

注: タイマの挿入やその他のスクリプト変更機能を有効にするには、TrueLog が最新でなければ (同期が取れていなければ) なりません。 TrueLog が最新でない場合には、スクリプトの試行を行って同期化してください。

1. **テストの分析** このプロセスでは、以前のテスト実行結果の調査、仮想ユーザー要約レポートの確認、再生エラーの場所の特定、再生セッションと元の記録セッションとの比較を行います。 このプロセスで、カスタマイズしたスクリプトのエラー チェックを行うことができます。

2. **セッション処理のカスタマイズ** 古いセッション データが負荷テスト スクリプトに埋め込まれたまま、後続のテスト実行で再実行すると、セッション処理エラーが発生します。 TrueLog Explorer では、このような静的データを特定し、動的データに置き換えて、テスト実行を成功させるためのプロセスを自動化しています。 手作業でコードを編集する必要はありません。

   注: Silk Performer のスクリプト記録技術では、静的セッション情報を含まない、コンテキストを持つスクリプトが生成されるため、セッション処理のカスタマイズはたいてい必要ありません。

3. **ユーザー データのカスタマイズ** 負荷テスト中に現実の状況をより正確にシミュレートするには、シミュレートするトランザクションごとに、仮想ユーザーがサーバーに対して行うアクションを変更する必要があります。 ユーザー入力データをカスタマイズすることで、フォーム フィールドからのデータ入力といったユーザー タスクをテスト スクリプトでシミュレートする際にデータを取り出すデータ ファイル (名前/アドレス、番号、製品のリストなど) を指定できます。 ランダム関数を使用して、ランダムに生成したデータを入力フィールドに挿入することもできます。

4. **検証関数の追加** セッション情報や仮想ユーザーのデータ入力をテスト スクリプトでどう処理するかのカスタマイズが済んだら、関数をスクリプトに組み込んで、テスト対象のアプリケーションが正確なデータを返しているかどうかを自動的に確認することができます。 このようなコンテンツ検証では、現実の状況でグラフィックやデータなどの要素をクライアントが実際に受信しているかどうかを確認します。 要素を受信していない場合には、検証結果としてエラーが出力されます。

5. **Silk Performer によるカスタマイズの拡張** TrueLog Explorer ではほとんどの BDL スクリプトが自動的に作成されますが、Silk Performer を使って手作業でスクリプトのコードを書くことで、より複雑な検証やカスタマイズを追加することができます。

6. **TrueLog On Error の分析** テストを実行した後、TrueLog On Error ファイルには、テスト中にエラーとなったすべてのトランザクションの完全な履歴が記録されています。 TrueLog On Error を使って実際のコンテンツを詳しく調べ、システムやアプリケーションの障害の根本原因を分析することができます。

TrueLog について

履歴ファイルの概要を説明します。このファイルには、データ/ページ リクエストやサーバー レスポンスについてのきわめて詳細な情報が、テスト中に記録されます。

TrueLog の概要

Silk Performer は、セッションとテスト スクリプトを関連付け、クライアントとサーバーの間で送信されるすべてのデータを記録するため、唯一、再生結果をスクリプトのカスタマイズや検証に取り入れることが可能です。

この TrueLog 技術 は、エラーの再現や根本原因の分析において大きな利点となります。 TrueLog は履歴ファイルであり、このファイルには、データおよびページのリクエストやサーバー レスポンスについてのきわめて詳細な情報が、テスト中に記録されます。 Web テストの場合、この詳細情報にはページの完全なコンテンツが含まれます。 このような情報は、システム障害の原因を特定する際に非常に有益です。

TrueLog を使うと、クライアント/サーバー アプリケーションや Web アプリケーションでエラー状況が発生したときに記録した実際のコンテンツを、詳しく調べることができます。 HTML アプリケーションの場合、TrueLog Explorer では、セッション コンテキストを含め、仮想ユーザーがテスト時に目にするものと同じ、レンダリングされた直感的な HTML インターフェイスを通じて、TrueLog ファイルを分析したり操作することができます。

TrueLog はセッションやテスト スクリプトと密接に結合しているため、リアルタイムでのスクリプト カスタマイズが可能になります。 スクリプトをカスタマイズした後、スクリプトの試験的実行によってすぐにその効果を確認できます。つまり *[スクリプトの試行]* を使って TrueLog を検査し、カスタマイズが成功したかどうかを判断することができます。 カスタマイズがうまく動かない場合には、TrueLog で、エラー

の原因となったセッションでのシステム リクエスト/レスポンスを見ることができます。 このプロセスは、カスタマイズした部分がすべて意図したとおりに動作するまで繰り返すことができます。

TrueLog ファイルと TrueLog On Error ファイルのファイル拡張子は .xlg です。 XLZ ファイルは、ZIP 形式で圧縮された TrueLog ファイルであり、TrueLog Explorer 内では解凍された TrueLog ファイルとして扱われます。 TrueLog を ZIP 圧縮することでダウンロード時間を短縮できます。 Silk Performer は ZIP 圧縮された TrueLog を使用しません。

負荷時の視覚的検証

Web アプリケーションが HTTP エラー コードを使ってエラー情報を送信するのは、一般的ではありません。 通常は、HTML コンテンツにエラー情報を含めて送信します。 このような場合はたいてい、スクリプトがページ上のリンクを呼び出そうとして、ページに元のコンテンツではなくエラー メッセージが含まれていたときに初めて、エラーが検出されます。 このとき一般に、「リンクが見つかりません」というエラーが出力されます。 エラーの発生原因を明らかにし、デバッグ環境でのエラーの再現方法を決定するには、このメッセージではたいてい不十分です。 このような見つけにくいエラーの原因を突き止めるには、負荷がかかったときのエラー履歴を記録し、エラーが発生する前の仮想ユーザーとサーバーのアクションを割り出すしかありません。

TrueLog はこの情報を提供します。シミュレート対象の仮想ユーザーが数千に上り、長い期間の中でまれにしかエラーが発生しない場合でも、それは同じです。

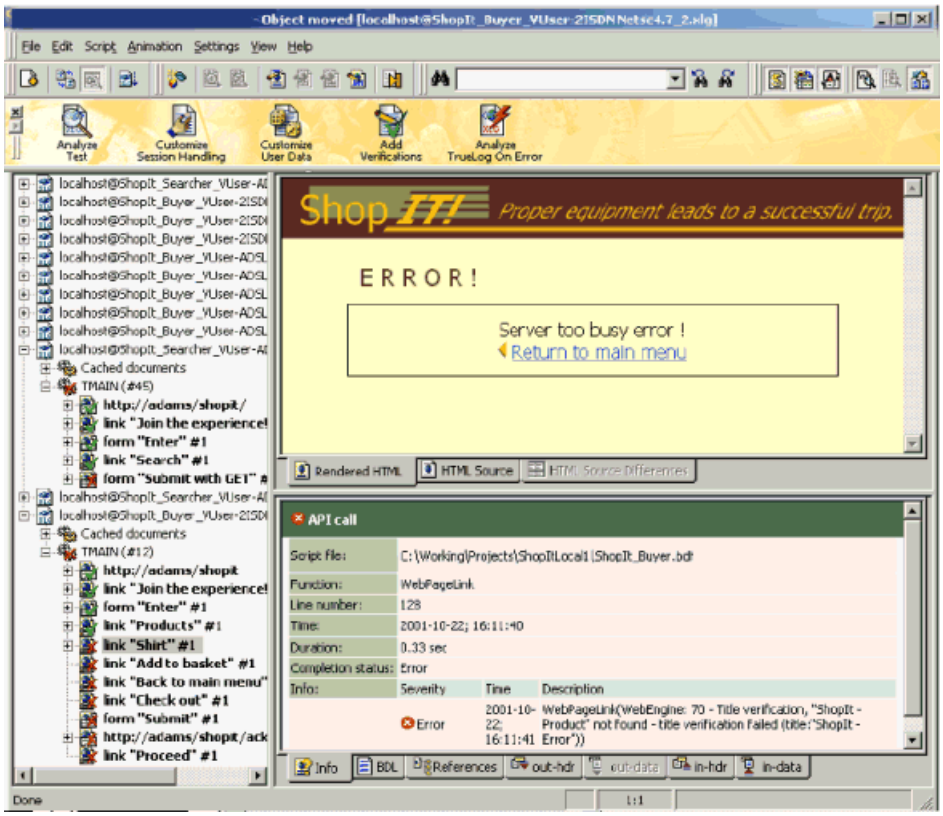
TrueLog が提供する負荷時の視覚的検証によって、アプリケーションの負荷が高いときに一部のユーザーにしか通常は発生しない、見つけにくいアプリケーション レベルのエラーを明らかにすることができます。 ほとんどのアプリケーションでは、アプリケーションが配置された後でしかこの種の負荷がかかることはありません。 Web アプリケーションの HTTP 以外の典型的なエラーには、Web ページのテキストの誤りや値の計算結果の誤り、サーブレット エラー や サーバーがビジー状態です といったアプリケーション関連のメッセージなどがあります。

根本原因分析のためのリクエスト/レスポンス ログ

負荷時の視覚的検証だけでは、アプリケーション レベルのエラーの根本原因まで突き止められないため、あまり価値はありません。 このため、TrueLog では、エラー発生前のものも含め、関連するユーザー操作とサーバー レスポンスをすべて記録しています。

返された Web ページに特定のリンクが含まれていないため リンクが見つかりません というエラーが表示された場合に、仮想ユーザーに対してアプリケーション レベルのエラーが発生する可能性があります。 Web サーバーが完全な Web ページを返せなかったのかもしれませんし、アプリケーション サーバーがページの動的コンテンツをタイミングよく渡せなかったのかもしれません。 TrueLog Explorer を使用して、このようなエラーの発生前に記録された TrueLog 履歴を分析すると、エラーの原因となったクリック パスを視覚的に調べることができます。 エラーの根本原因を特定して根本的な修正を行う上で、この TrueLog 履歴は非常に有益な情報となります。

次の図は、エラーの原因となったトランザクションのクリック パスを示す視覚的ログです。 この TrueLog は、Web サーバーから返されたエラー メッセージが原因でタイトル検証エラーが生じたことを表しています。



TrueLog On Error ファイル

TrueLog ファイルはサイズが大きくなる可能性があるため、負荷テストでエラーが発生したときにのみ TrueLog を生成するよう、Silk Performer を構成することができます。 システムが正しく動いている間は、何も記録されません。 このように対象を限定した TrueLog の生成は *TrueLog On Error* と呼ばれ、生成される TrueLog ファイルはエラー状況に的を絞った小さいものになります。

テスト実行中にエラーが検出された場合、TrueLog On Error ファイルを見ることでそのエラーの原因が明らかになります。 テスト実行後にワークフロー バーで **TrueLog On Error の分析** をクリックし、該当する TrueLog ファイルにアクセスして分析してください。

TrueLog をステップ スルーする

TrueLog のステップ スルー ダイアログ ボックスを使って、記録された TrueLog データを指定した間隔で進むことができます。

1. **TrueLog** メニュー ツリーで TrueLog を選択します。 TrueLog の内容が **レンダリング** ページと **情報** ページに表示されます。
2. **編集** > **TrueLog のステップ スルー** を選択します。
   代替方法:ツールバーの **TrueLog のステップ スルー** をクリックします。
   TrueLog のステップ スルー ダイアログ ボックスが開きます。
3. TrueLog の記録されたデータ ノードの中を進みたい間隔を指定します。
4. **次を検索** と **前を検索** をクリックして、指定した間隔でノード間を移動します。

プロトコル ベースの TrueLog タイプを指定する

TrueLog Explorer で利用できる機能は、開いている TrueLog のプロトコルに応じて変化します。 たとえば、**フォーム データ** タブが有効なのは Web TrueLog が開いているときだけ、**TrueLog のステップ スルー** ダイアログ ボックスの **ラウンドトリップ** オプションが有効なのは SAPGUI TrueLog が開いていると

きだけ、**TrueLog のステップ スルー** ダイアログ ボックスの **安定な画面** オプションが有効なのはターミナル エミュレーション TrueLog が開いているときだけです。

1. **編集** > **TrueLog のタイプ** を選択します。
2. TrueLog Explorer に取り込みたい機能セットを選択します。
   デフォルトのプロトコルは **Web** です。

   注: Oracle Forms は混合型のプロトコルで、多くの場合、スクリプト内に Web 呼び出しを含んでいます。 Web ベース プロトコルの機能セットを選択し、Oracle Forms スクリプトに含まれる HTML に対して、セッション処理をカスタマイズしたり、HTML の値を解析したり、検索を実行したりすることもできます。

テストの分析

Silk Performer Recorder を使って BDL 形式のテスト スクリプトを記録し保存したら、スクリプトの試行を実行してスクリプトをテストする必要があります。 スクリプトの試行は、実際の負荷テストではなく、スクリプトのデバッグが必要かどうかを確認するための試験的なテスト実行です。 スクリプトをカスタマイズするたびにスクリプトの試行を実施することをお勧めします。 **スクリプトの試行** ボタンでスクリプトの試行を開始すると、自動的に TrueLog Explorer が開いて、最も新しいテスト実行の TrueLog が表示されます。

TrueLog Explorer によるテストの分析には、次の 3 つのタスクが含まれます。

- 仮想ユーザー要約レポートの表示
- エラーの検索
- 再生テスト実行と記録テスト実行の比較

視覚的分析

TrueLog Explorer は、テスト対象のアプリケーションが表示する Web コンテンツを視覚的にレンダリングして、仮想ユーザーがテスト中に見ているものを正確に示すことができます。

SQL コマンドや、Oracle Forms コントロール、TCP/IP 操作、UDP 操作は、Web コンテンツのような高度なインターフェイスを持ちませんが、TrueLog Explorer の比較モードを使うことで、データベース、Oracle Forms、Citrix、TCP/IP、UDP の記録 TrueLog を、データベース、Oracle Forms、Citrix、TCP/IP、UDP の 再生 TrueLog と比較することができます。

TrueLog を開く

1. **ファイル** > **TrueLog の追加** または **ファイル** > **TrueLog On Error ファイルの追加** を選択します。 **ファイルを開く** ダイアログ ボックスが開き、指定したファイルの種類が **ファイルの種類** リスト ボックスで選択されます。
2. 該当するテストのディレクトリに移動し、調査対象のファイルを選択します。
3. 省略可能*:* **比較ビューで開く** チェック ボックスをオンにすると、ファイルは比較ビューで開きます。
4. **OK** をクリックします。

> **注:** 複数の TrueLog を TrueLog Explorer で同時に開くこともできます。

テスト実行を分析する

1. 分析または変更したい TrueLog を開きます。
2. ワークフロー バーの **テストの分析** をクリックします。 **ワークフロー - テストの分析** ダイアログ ボックスが開きます。

3. 以下のいずれかのオプションを選択します。

- **仮想ユーザー要約レポートの表示**
- **エラーの検索**
- **テスト実行の比較**

仮想ユーザー要約レポート

スクリプトの試行に関する基本情報やタイミング平均を要約した仮想ユーザー要約レポートについて説明します。

仮想ユーザー要約レポートの概要

仮想ユーザー要約レポートは、個々のスクリプト試行について要約したレポートで、基本情報とタイミング平均を含みます。 それぞれのレポートは 1 人の仮想ユーザーについて作成されます。 データは表形式で表示されます。

注: 仮想ユーザー要約レポートの処理にはかなりの時間がかかるので、デフォルトでは生成されない設定になっています。

アニメーションを指定したスクリプトの試行が終了したときや、メニュー ツリーで TrueLog ファイルのルート ノードをクリックしたときに、自動的に仮想ユーザーレポートが表示されるようにするには、**設定** > **オプション** > **ワークスペース** > **レポート** で **仮想ユーザー レポートを表示する** チェック ボックスをオンにしてください。

仮想ユーザー要約レポートには、以下に関する詳細情報が含まれています。

- 仮想ユーザー
- 検出されたエラー
- テスト スクリプトに定義されているトランザクションごとに追跡したレスポンス時間情報
- ダウンロードされた Web ページごとのページ タイマの測定値
- テスト スクリプトで宣言されている各 Web フォームに関連する測定値。POST、GET、および HEAD の各メソッドを使用したフォーム送信に対するレスポンス時間測定値とスループットを含みます。
- スクリプトで使用された個別のタイマおよびカウンタ (Measure 関数)
- IIOP、Web フォーム、TUXEDO、SAP などに関連する情報

仮想ユーザー要約レポートを表示する

1. 分析または変更したい TrueLog を開きます。
2. ワークフロー バーの **テストの分析** をクリックします。 **ワークフロー - テストの分析** ダイアログ ボックスが開きます。
3. **仮想ユーザー要約レポートの表示** リンクをクリックします。

エラーを検索する

TrueLog Explorer では、スクリプトの試行の後でエラーをすばやく検索できます。 そして、エラーとなったリクエストを調査して、必要なカスタマイズを行うことができます。

注: メニュー ツリーでは、再生エラーを含む API ノードは赤い「X」印が付いて表示されます。

1. 分析または変更したい TrueLog を開きます。
2. ワークフロー バーの **テストの分析** をクリックします。 **ワークフロー - テストの分析** ダイアログ ボックスが開きます。
3. **エラーの検索** リンクをクリックします。 **TrueLog のステップ スルー** ダイアログ ボックスが、**エラー** オプションが選択された状態で表示されます。
4. 一度に 1 つのエラーを検索しながら TrueLog 結果ファイル内を移動するには、**次を検索** をクリックします。

別の方法で TrueLog 内を移動して、スクリプトが意図どおりに動作したかを視覚的に検証することもできます (**ページ全体**、**HTML ドキュメント**、**フォーム送信**、または **API 呼び出し**)。

タイマ関数を挿入する

新しいタイマ セッションを挿入するには、再生 TrueLog を読み込んで、TrueLog Explorer の エクスプローラ モードを有効にしておく必要があります。

注: タイマの挿入やその他のスクリプト変更機能を有効にするには、TrueLog が最新でなければ (同期が取れていなければ) なりません。 TrueLog が最新でない場合には、スクリプトの試行を行って同期化してください。

測定関数 (タイマ) を BDL スクリプトに挿入して、あるトランザクションの実行から次のトランザクションの実行の間に経過した時間を測定することができます。 測定関数の結果は、仮想ユーザーの要約レポートに一緒に表示されます。

注: 元から存在するタイマの MeasureStart 関数と MeasureStop 関数の間に新しいタイマ セッションを挿入することもできます。

1. メニュー ツリーでタイマを開始する API ノードを右クリックし、**新規タイマの開始** を選択します。 **タイマの開始** ダイアログ ボックスが開きます。
2. タイマの名前を入力して **OK** をクリックします。 そのノードの BDL スクリプトに、新しい MeasureStart 関数が挿入されます。
3. タイマを終了する API ノードを右クリックし、**タイマの停止** を選択します。 **タイマの停止** ダイアログ ボックスが開きます。
4. 停止したいタイマを選択し、**OK** をクリックします。 そのノードの BDL スクリプトに、新しい MeasureStop 関数が挿入されます。

注: 1 つのスクリプトに複数のタイマを挿入する場合や、挿入したタイマをすべて停止したかどうかが不確かな場合には、New Timer Session を使って、タイマを挿入する前にこれまでのタイマをすべて停止します。 New Timer Session では、まだ解決されていないすべてのタイマに対して MeasureStop 関数を挿入し、その後、新しいタイマ セッションの MeasureStart 関数を挿入します。

パフォーマンス分析 (HTTP)

テスト実行が正確であることを検証した後、TrueLog Explorer では、**情報** ペインの **統計** タブを使って、"負荷のない" 状態でのアプリケーションのパフォーマンスを分析できます。 **概要** ページには、ページの総レスポンス時間、ドキュメント ダウンロード時間 (サーバー ビジー時間を含む)、埋め込みオブジェクトの受信に要した時間が表示されます。

詳細な Web ページ統計には、個々の Web ページ コンポーネントの正確なレスポンス時間が表示されます。 この詳細統計を使って、エラーやページ ダウンロードの遅延の根本原因を突き止めることができます。

Web ページの詳細な分析結果には、ページ コンポーネントごとに以下のデータが記載されています。

- DNS 検索時間
- 接続時間
- SSL ハンドシェイク時間
- リクエスト送信時間
- サーバー ビジー時間
- レスポンス受信時間
- キャッシュ統計値

パフォーマンス分析 (AJAX)

テスト実行が正確であることを検証した後、TrueLog Explorer では、**情報** ペインの **統計** タブを使って、"負荷のない" 状態でのアプリケーションのパフォーマンスを分析できます。 **概要** ページには、合計ア

クション時間、ドキュメント ダウンロード時間 (サーバー ビジー時間と埋め込みオブジェクトの受信に要した時間を含む) が表示されます。

詳細な Web ページ統計には、個々の Web ページ コンポーネントの正確なレスポンス時間が表示されます。 この詳細統計を使って、エラーやページ ダウンロードの遅延の根本原因を突き止めることができます。

Web ページの詳細な分析結果には、ページ コンポーネントごとに以下のデータが記載されています。

- DNS 検索時間
- 接続時間
- ネット ラウンドトリップ
- キャッシュ統計値

概要ページを表示する

1. 統計値を表示する API ノードを選択します。
2. **統計** タブをクリックします。 **統計** ビューが開きます。

再生 TrueLog と記録 TrueLog

スクリプトの開発プロセス中に生成された TrueLog と、当初生成された TrueLog を比較することで、テスト スクリプトが正確に実行されたかどうかを確かめることができます。

再生 TrueLog と記録 TrueLog の概要

スクリプトの開発プロセス中に生成された TrueLog と、当初生成された TrueLog を比較することで、テスト スクリプトが正確に実行されたかどうかを確かめることができます。

Web アプリケーションのテストの場合、TrueLog Explorer には、テスト時に受信された実際の Web ページが表示されます。 TrueLog Explorer のアニメーション モードでは、ダウンロードされたデータをリアルタイムで監視することができます。 テスト中に受信されたとおりにデータが表示されます。

Web テストの場合、TrueLog Explorer は受信したデータを以下のようなさまざまな形で [コンテンツ] ペインに表示します。

- **レンダリング** タブでレンダリングした HTML として (仮想ユーザーが見るのと同じもの)
- **ソース** タブでネイティブ HTML コードとして
- **ソース差分** タブで再生 TrueLog と記録 TrueLog の違いを示す差分テーブルとして
- **ポスト データ** タブでポストされたデータとして (テキスト入力フィールドなどのユーザー インターフェイス コントロールを含むページ)

注: 再生時のコンテンツを表示しているウィンドウには、左上隅に緑色の三角マークが付いています。 アプリケーションの記録時に元々表示されていたコンテンツを表示しているウィンドウには、左上隅に赤い三角マークが付いています。

ヒント: 分析の結果、セッション処理が有効でないことがわかった場合には、ワークフロー バーの **セッション処理のカスタマイズ** をクリックします。

注: TrueLog の比較には、複数のエントリ ポイントやアプローチが利用可能です。 再生メニュー ツリーで再生 TrueLog を開いておく必要はありませんし、記録メニュー ツリーで記録 TrueLog を開いておく必要もありません。 2 つの TrueLog を相互比較することも可能です。 どのような TrueLog の比較でも手順は同じです。

再生 TrueLog と記録 TrueLog を比較する

1. **ファイル** > **TrueLog の追加** または **ファイル** > **TrueLog On Error ファイルの追加** を選択します。 **ファイルを開く** ダイアログ ボックスが開き、指定したファイルの種類が **ファイルの種類** リスト ボックスで選択されます。

2. 該当するテストのディレクトリに移動し、調査対象のファイルを選択します。
3. 省略可能*:* **比較ビューで開く** チェック ボックスをオンにすると、ファイルは比較ビューで開きます。
4. **OK** をクリックします。
5. ワークフロー バーの **テストの分析** をクリックします。 **ワークフロー - テストの分析** ダイアログ ボックスが開きます。
6. **テスト実行の比較** をクリックします。 対応する記録 TrueLog が **比較** ビューで開き、**ページ全体** オプションが選択された状態で **TrueLog のステップ スルー** ダイアログ ボックスが表示されます。 これを選択していると、TrueLog をノードごとに比較することができます。
7. **次を検索** をクリックして、TrueLog 結果ファイル内を 1 ページずつ移動します。

   別の方法でも TrueLog を移動して、スクリプトが意図どおりに動作したかを視覚的に検証してみてください (**HTML ドキュメント**、**エラー**、**フォーム送信**、または **API 呼び出し**)。

   注: 記録セッションに関するサブペインには赤色の三角マークが、再生セッションに関するサブペインには緑色の三角マークが、それぞれ左上の隅に付いています。

再生 TrueLog と記録 TrueLog を同期させる

比較モードでは、記録値と再生値の差分を識別できるように、再生 TrueLog と記録 TrueLog の間で対応する API ノードを同期することができます。

注: この機能は、TrueLog の自動同期が有効になっている場合は無効になります。

1. 次のいずれかを実行して比較モードを有効にします。
   - **表示** > **比較モード** を選択します。
   - ツールバーの **比較モード** ボタンをクリックします。
2. 対応する一組の記録/再生 TrueLog を開きます。
3. API ノードを右クリックし、**TrueLog の同期** を選択します。 TrueLog Explorer は、対応する TrueLog 内で、選択された API ノードと最も相関がある API ノードを見つけます。

再生 TrueLog と記録 TrueLog を自動的に同期させる

比較モードで、再生 TrueLog と記録 TrueLog の対応する API ノードを自動的に同期させることができます。

1. 次のいずれかを実行して比較モードを有効にします。
   - **表示** > **比較モード** を選択します。
   - ツールバーの **比較モード** ボタンをクリックします。
2. 対応する一組の記録/再生 TrueLog を開きます。
3. 次のいずれかを実行して TrueLog を同期させます。
   - ツールバーの **TrueLog の同期を保つ** をクリックします。
   - **表示** > **TrueLog の同期を保つ** を選択します。

   これで、いずれかの TrueLog で API ノードをクリックしたときに、他方の TrueLog の API ノードのうちで選択した API ノードと最も関連が強いものを TrueLog Explorer が自動的に選択してくれます。

TrueLog On Error

TrueLog On Error ファイルは、負荷テスト時に見つかったエラーの前のトランザクションの完全な履歴です。 このファイルを用いて、実際のコンテンツを詳しく調べ、エラー状況を分析することができます。

TrueLog On Error ファイルには、すべてのクライアント リクエストとサーバー レスポンスの履歴が保存されています。 TrueLog On Error ファイルは、エラーが発生したセッションと関連させてエラーを示し、テスト スクリプトと密接に統合されているため、システムやアプリケーションの障害の根本原因を分析するのに非常に適しています。

TrueLog On Error が記録されるのはエラーが発生した場合だけなので、標準の TrueLog と比べると、消費するメモリや処理量はずっと少なくなります。

通常は、テストを実行した後に、複数の TrueLog On Error ファイルが TrueLog Explorer で開きます (エラーを返した仮想ユーザーごとに TrueLog が 1 つ)。 TrueLog Explorer の **エラーの検索** 機能を使うと、エラーがどの TrueLog に記録されているかに関係なく、発生した順にエラーを特定することができます。これによりエラーの分析作業を単純化できます。 開いている TrueLog をすべて手作業で調べながら次のエラーを順に探す必要はありません。

TrueLog On Error の分析

負荷テストを完了します。

1. Silk Performer で **結果の検討** をクリックします。 **ワークフロー - 結果の検討** ダイアログ ボックスが開きます。
2. **TrueLog Explorer** をクリックします。

   注: TrueLog が見つからない場合、**Silk TrueLog Explorer の起動** オプションは無効になっています。

   TrueLog Explorer が起動し、現在開いているテストで生成された TrueLog On Error ファイルがすべて表示されます。 **エラーの検索** ダイアログ ボックスが開きます。

   注: そのテストに関連する TrueLog ファイルが 100 個を超える場合は、最初の 100 個の TrueLog だけが自動的に開くというダイアログが表示されます。
3. **検索範囲** リスト ボックスから、**開いているすべての TrueLog** (デフォルト) を選択します。

   開いているすべての TrueLog に含まれるエラーが順に検索されます。
4. メニュー ツリーで TrueLog を選択します。
5. **検索範囲** リスト ボックスから、**選択した TrueLog のみ** を選択します。

   選択した TrueLog に含まれるエラーだけが順に検索されます。

   注: TrueLog On Error ファイルは、作られた順に表示されます。
6. このダイアログの 検索対象 の部分では、検索したいエラーの種類 (**[エラー]**、**[警告]**、**[情報]**) を選択します。
7. ダイアログの 検索方法 の部分では、**検索方法** (*[*選択したノードから検索を開始する*]*、*[*最初のエラーから検索を開始する*]*) を指定します。
8. **次を検索** ボタンと **前を検索** ボタンを使用して、エラー間を移動します。

   ヒント: Web アプリケーションのテストでは、ロードされない画像やエラー メッセージなどの形でページ コンテンツにエラーが現れる 1、2 ステップ前に、実際のエラーが発生していることがよくあります。

TrueLog を閉じる

TrueLog の分析が終わったら、TrueLog を閉じてメニュー ツリーから消すことができます。

1. TrueLog のメニュー ツリーで、閉じたい TrueLog を選択します。
2. **ファイル** > **選択した TrueLog を削除** を選択します。

   代替方法: 開いているすべての TrueLog を閉じるには、**ファイル** > **すべての TrueLog を削除** を選択します。

Silk Performer での作業

ここでは、読者が既に Silk Performer の機能をよく知っていることを前提としています。 Silk Performer の使い方や TrueLog Explorer との連携の詳細については、 のヘルプを参照してください。

Silk Performer から TrueLog を検討する

Silk Performer ワークフロー バーの **スクリプトの試行** をクリックしてスクリプトの試行を開始すると、TrueLog Explorer が自動的に起動し、最も新しく生成された TrueLog が表示されます。 その他にも、TrueLog Explorer は以下の方法で Silk Performer から開くことができます。

- Silk Performer のスクリプト エディタでスクリプトを右クリックし **最近のスクリプト試行の TrueLog を検討** を選択すると、最も新しいスクリプト試行の再生 TrueLog を対応するノードで開くことができます。
- Silk Performer の **出力** ペインの **仮想ユーザー** タブで仮想ユーザー出力を右クリックし **TrueLog の検討** を選択すると、現在の実行の再生 TrueLog を対応するノードで開くことができます。
- Silk Performer のプロジェクト メニュー ツリーでスクリプトを右クリックして **記録された TrueLog の検討** を選択すると、記録 TrueLog を開くことができます。
- Silk Performer のプロジェクト メニュー ツリーでスクリプトを右クリックして **最近のスクリプト試行の TrueLog を検討** を選択すると、最も新しいスクリプト試行の再生 TrueLog を開くことができます。
- **結果** > **TrueLog の検討** を選択すると、再生 TrueLog を開くことができます。
- Silk Performer の **監視/仮想ユーザー** ビューで仮想ユーザーを右クリックし **TrueLog の検討** を選択すると、現在の実行の再生 TrueLog を開くことができます。

注: ワークフロー バーで **結果の検討** をクリックすると、TrueLog On Error を分析することができます。

Silk Performer から TrueLog を有効化する

TrueLog Explorer では、TrueLog および TrueLog On Error ファイルの書き出しに Silk Performer を使用しています。 スクリプトの試行 (TrueLog ファイルを利用する最も一般的なシナリオ) の際には、自動的に TrueLog ファイルが有効化されます。 構成は必要ありません。

TrueLog を標準的に生成すると大量のデータが出力されてパフォーマンスが低下する可能性があるため、トランザクションが失敗したときにのみ TrueLog 情報を記録する TrueLog On Error を使うことを検討してください。

以下の表では、TrueLog および TrueLog On Error を有効化するためのオプションを説明します。

オプション	ステップ
Silk Performer の **結果** ツールバー	すべての TrueLog アクティビティを記録するには Silk Performer の **結果** ツールバーの **TrueLog ファイルの生成** を、トランザクションの失敗が起きたときにだけ TrueLog を生成するには **TrueLog On Error ファイルの生成** をクリックします。
Silk Performer メニュー	**設定** > **アクティブ プロファイル** > **結果** > **TrueLog** を選択します。 すべてのアクティビティを記録するには **TrueLog ファイル (.xlg)** チェック ボックスを、トランザクションの失敗が起きたときにだけ TrueLog を生成するには **TrueLog On Error ファイル (.xlg)** チェック ボックスをオンにします。 **OK** をクリックします。
Silk Performer の **ワークロードの設定** ページ	テストの TrueLog On Error を有効化するには、Silk Performer の **ワークロードの設定** ページで **設定** セクションのチェック ボックスをオンにします。

スクリプトの試行

スクリプトの試行は、テスト スクリプトの準備状況を評価するために用いられる Silk Performer のテスト実行です。 スクリプトの試行では、元の記録セッションを再生セッションと比較し、解析が必要なセッション情報を特定したり検証関数をテストしたりします。

TrueLog Explorer からスクリプトの試行を実行する

スクリプト > **スクリプトの試行** を選択します。すると、 Silk Performer がテスト スクリプトを開いて実行します。

Silk Performer からスクリプトの試行を実行する

1. Silk Performer のワークフロー バーで **スクリプトの試行** をクリックします。 **スクリプトの試行** ダイアログ ボックスが開きます。
2. **実行** をクリックします。

注: スクリプトの試行中に TrueLog Explorer を開くには、**スクリプトの試行** ダイアログ ボックスで **TrueLog Explorer によるアニメーション実行** チェック ボックスをオンにします。

Silk Test 用 TrueLog Explorer

TrueLog Explorer の機能テスト ツールである Silk Test, には、異なるバージョンの Silk が付属しています。 TrueLog Explorer に付属の Silk Test はビューア モードに設定されています。つまり、Silk Test TrueLog はテスト ケース内の結果の調査のみをサポートするということです。 ほとんどの Silk Performer ベースの TrueLog で利用可能なスクリプトのカスタマイズは提供されません。

Silk Performer は、Silk Test を利用して、リモート デスクトップ セッションで GUI レベルの (ファットクライアントの) テストを実行します。 詳細については Silk Performer のヘルプを参照してください。

Silk Test と TrueLog Explorer の連携については、Silk Test に付属の TrueLog Explorer ヘルプを参照してください。

パースペクティブ

TrueLog Explorer には、さまざまな TrueLog タイプに対応できるよう、2 つの表示モード、つまりパースペクティブが用意されています。 デフォルトであり、Silk Performer ユーザーがもっともよく使用するビュー パースペクティブは、エクスプローラ パースペクティブです。 エクスプローラ パースペクティブには、TrueLog Explorer が提供するスクリプト カスタマイズ機能がすべて備わっています。 エクスプローラ パースペクティブでは、TrueLog Explorer のワークフロー バーに用意されているすべての機能を利用できます。 デフォルトでは、有人の Silk Performer テストは常にエクスプローラ パースペクティブの TrueLog を生成します。

ビューア パースペクティブでは、すべての TrueLog Explorer ビューで一部のコンテキスト メニュー コマンドしか利用できず、ワークフロー バーは無効です。 このパースペクティブの目的は、スクリプトをカスタマイズしないユーザー向けの簡易ビューを提供することと、スクリプトをカスタマイズできない TrueLog タイプをサポートすることです。 デフォルトでは、無人の Silk Performer テストと Silk Test テストは常にビューア パースペクティブの TrueLog を生成します。

ビューア パースペクティブに切り替える

エクスプローラ パースペクティブとビューア パースペクティブの切り替えは、エクスプローラ パースペクティブを提供している TrueLog タイプで有効になっています。 エクスプローラ パースペクティブがサポートされていない TrueLog タイプの場合は、ビューア パースペクティブ モードのみが使用可能です。

次のいずれかのオプションを選択します。

- **ビューア パースペクティブの有効化** (ツールバーの目のアイコン) をクリックします。
- **表示** > **パースペクティブ** > **ビューア** を選択します。

エクスプローラ パースペクティブに切り替える

エクスプローラ パースペクティブとビューア パースペクティブの切り替えは、エクスプローラ パースペクティブを提供している TrueLog タイプで有効になっています。 エクスプローラ パースペクティブがサポートされていない TrueLog タイプの場合は、ビューア パースペクティブ モードのみが使用可能です。

次のいずれかのオプションを選択します。

- **エクスプローラ パースペクティブの有効化** (ツールバーの拡大鏡のアイコン) をクリックします。
- **表示** > **パースペクティブ** > **エクスプローラ** を選択します。

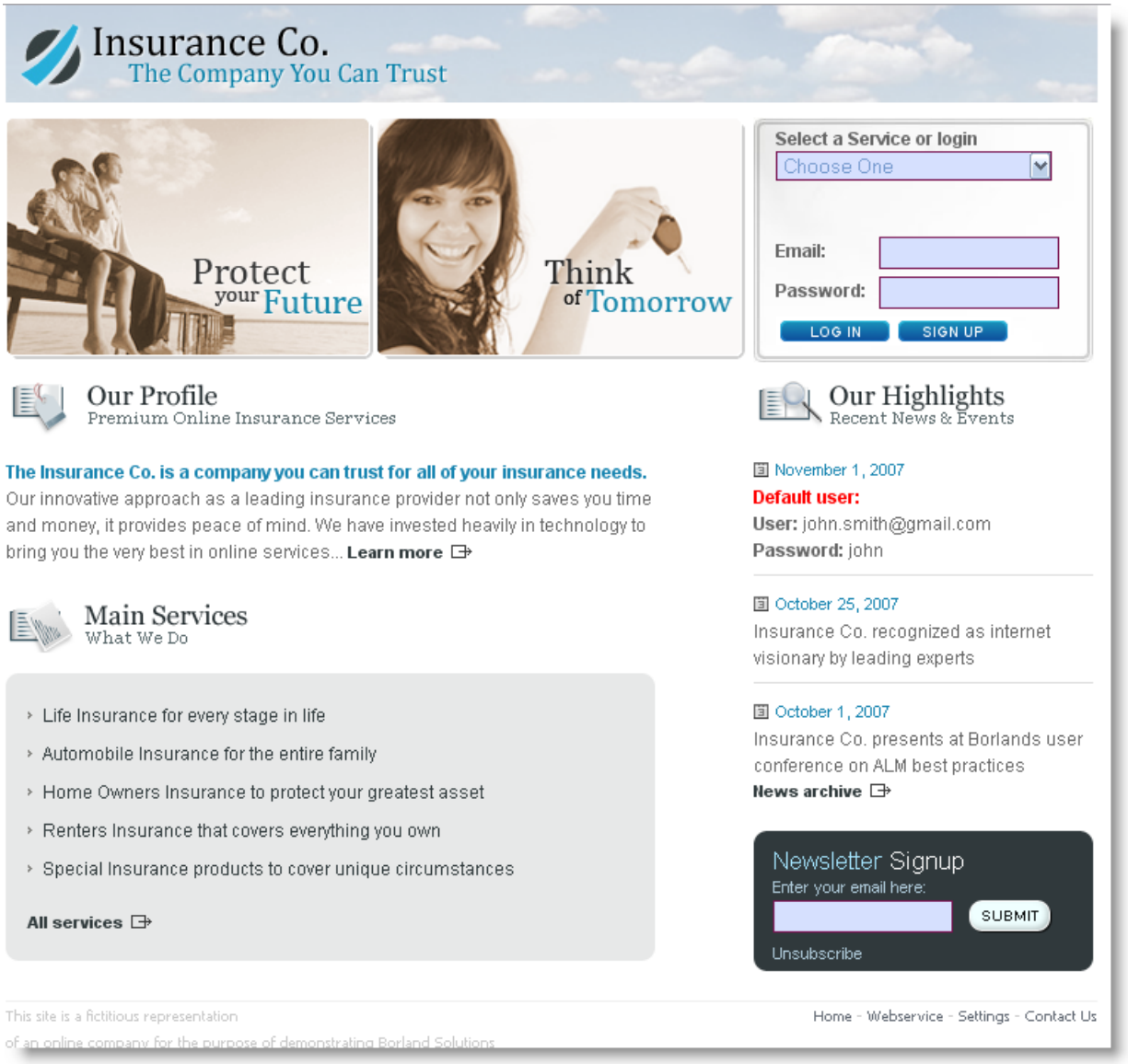
サンプル アプリケーション

このヘルプでは、3 つのサンプル アプリケーション (Web 2.0 アプリケーション、純粋な HTML を使用した典型的な Web アプリケーション、およびデータベース アプリケーション) を参照します。 これらのアプリケーションによって Silk Performer および TrueLog Explorer の機能をよく学んでから、重要なデータに適用してください。

サンプル Web 2.0 アプリケーション

Silk Performer には、Web 2.0 アプリケーションのテストを習得するために使用できる最新のサンプル Web アプリケーションが備えられています。 InsuranceWeb サンプル Web アプリケーションは、ExtJS および JSF フレームワークに基づいて作成され、AJAX 技術を採用し、JSON と XML を介して通信します。

サンプル アプリケーションは、*http://demo.borland.com/InsuranceWebExtJS/* でホストされます。



典型的な Web アプリケーションのサンプル

ShopIt は、擬似的なオンライン ショッピングのためのキャンピング用品カタログを持つ、純粋な HTML を使用した簡単な e コマースの Web サイト をシミュレートするものです。 このアプリケーションを使用して、TrueLog Explorer の純粋な HTML を使用した Web アプリケーション機能を試すことができます。

ShopIt の概要

ShopIt は、擬似的なオンライン ショッピングのためのキャンピング用品カタログを持つ、簡単な e コマースの Web サイト をシミュレートするものです。 このアプリケーションを使用すると、TrueLog Explorer の Web アプリケーション機能を試すことができます。

ShopIt は、セッション エラーや、在庫切れ商品に起因する Web リンクの欠落などのエラーを生成するように作られています。 セッション エラーの原因は JavaScript に埋め込まれたセッション情報であり、この情報は Silk Performer のコンテキスト管理では検出されません。 テスト スクリプトをエラーなしで実行するためには、セッション データがハードコードされたテスト スクリプトをカスタマイズする必要があります。

注: ShopIt の最新情報については、ShopItV60.exe の readme ファイル (セットアップ時にアクセス可能になります) を参照してください。

ShopIt のソフトウェア要件

Active Server Pages を持つ IIS 4 または IIS 5 と Microsoft Internet Explorer (バージョン 5.0 以降) を備えた Windows オペレーティング システムが必要です。

ShopIt V 6.0 をインストールする

Silk Performer のサンプル Web アプリケーションが、ShopIt V 6.0 です。 ShopIt V 6.0 は、シンプルな e コマース Web サイトをシミュレートするもので、オンライン購入のシミュレーションに使用できるキャンピング用品カタログを備えています。 このアプリケーションを使用すると、Silk Performer の Web アプリケーション機能を試すことができます。 ShopIt V 6.0 は、不明な Web リンク (商品の在庫切れが原因) やセッション エラーなどのエラーを生成するように作られています。

ShopIt V 6.0 をインストールする前に リリース ノート を参照し、お使いのシステムで ShopIt V 6.0 の使用がサポートされているか確認してください。

ShopIt V 6.0 セットアップは、次の場所から利用できます。

- Silk Performer のインストール CD ¥ Extras ¥ ShopItV60.exe
- Web パッケージ：抽出された Web パッケージを保存した場所で、 ¥ Extras ¥ ShopItV60.exe を探してください。

1. インストール CD の ¥ Extras フォルダ、またはダウンロードした場所にある、ShopItV60.exe ファイルをダブルクリックします。

   注: ShopIt V 6.0 では、IIS (Internet Information Server) が ShopIt V 6.0 をインストールするコンピュータにインストールされていることが必要です。 IIS 7 では、Role Services ASP と ISAPI Extensions もインストールします。

   InstallShield によるインストールの準備ができると、**Welcome** ページが開きます。
2. **次へ** をクリックします。 **Choose Destination Location** ページが開きます。
3. デフォルトのインストール ディレクトリを変更するには、**Browse** をクリックし、**Choose Folder** ダイアログ ボックスを開きます。

   デフォルトのインストール先が、*Destination Folder* セクションに表示されます。

   ShopIt V 6.0 をインストールするフォルダを指定して、**OK** をクリックして前のダイアログ ボックスに戻ります。
4. **Next** をクリックして、インストール プロセスを続行します。

   Web アプリケーションの仮想ディレクトリの名前をエントリ フィールドに入力します。 これは Web サーバー上に作成されるディレクトリの名前です。 **Next** をクリックし、続行します。

   Specify Virtual Directory ダイアログ ボックスが開きます。
5. セットアップによりファイルがインストールされ、IIS で ShopIt V 6.0 Web アプリケーションが実行されるように設定されます。 完了すると、**Installation Complete** ダイアログ ボックスが開きます。
6. **Installation Complete** ダイアログ ボックスで、**Finish** をクリックします。 ShopIt V 6.0 Web アプリケーションをインストールしたコンピュータで使用する準備ができました。 選択した Web ブラウザで次の URL を入力して、ShopIt V 6.0 にアクセスできます。

   ```
   http://<computer  name>/<virtual  directory  name>/
   ```

   例：

   コンピュータ名が JohnSmith で、仮想ディレクトリのデフォルト値 ShopItV60 を変更していない場合、URL は次のようになります。

   ```
   http://JohnSmith/ShopItV60/
   ```

あるいは、インストールしたコンピュータから ShopIt V 6.0 にアクセスする場合は、次の URL でも動作します。

```
http://localhost/ShopItV60/
```

7. IIS 7 では、仮想ディレクトリを IIS に手動で追加します。
   - エイリアス：ShopItV60
   - 物理パス：ShopIt のディレクトリをインストールします。

サンプル データベース アプリケーション - Customer OCI

サンプル データベース アプリケーションの Customer OCI は、顧客の連絡先情報の追加、編集、削除を行う簡単なアプリケーションをシミュレートするものです。

Customer OCI の概要

サンプル データベース アプリケーションの Customer OCI は、顧客の連絡先情報の追加、編集、削除を行う簡単なアプリケーションをシミュレートするものです。 このアプリケーションで、新規顧客レコードの挿入とデータベース クエリを行うサンプルのデータベース セッションを記録することができます。 その結果として出力される TrueLog を使って、TrueLog Explorer のデータベース機能を学ぶことができます。

Customer OCI には、次の 2 つのバージョンがあります。

- Person PB V6 - OCI7
- Person PB V7 - OCI8

使用する PowerBuilder (バージョン 6 と 7) と OCI (バージョン 7 と 8) のバージョンが異なる点を除けば、どちらも同一です。

注: サンプルの TrueLog はすべて Person PB V7 - OCI8 を使って作成されています。

Customer OCI のソフトウェア要件

Customer OCI を実行するには、以下の要件を満たす必要があります。

- Oracle8 または Oracle9 クライアントがインストールされていること
- Oracle データベースにアクセスできること

Customer OCI にアクセスする

サンプル データベース アプリケーション Customer OCI を使用して、新規顧客レコードの挿入とデータベース クエリを行うサンプルのデータベース セッションを記録します。 Customer OCI は、インストールされた Silk Performer に含まれています。

どちらのバージョンにアクセスするかを決めてから、以下のステップのいずれかを実行してください。

- PersonPB V6 - OCI7 にアクセスするには、**スタート** > **プログラム** > **Silk** > **Silk Performer 9.5** > **サンプル アプリケーション** > **データベース サンプル** > **PersonPB V6 - OCI7 sample application**
- PersonPB V7 - OCI8 にアクセスするには、**スタート** > **プログラム** > **Silk** > **Silk Performer 9.5** > **サンプル アプリケーション** > **データベース サンプル** > **PersonPB V7 - OCI8 sample application**

セッション処理のカスタマイズ

記録 TrueLog に現れることがあるハードコードされたセッション データを解析する方法について説明します。 セッション処理のカスタマイズを行うと、テスト中に状態情報を保持できます。

セッション処理の概要

セッション処理のカスタマイズとは、負荷テスト時にアプリケーションの状態情報が変わらないようにしながら、サーバー レスポンスを操作する処理です。

後続のクライアント リクエストが同じコンピュータから同じユーザー セッションとして送られてきたものかを識別するための情報を、サーバーが実行時に生成し、クライアントへのレスポンスに含めることがよくあります。 たとえば、サーバーが セッション ID と一般に呼ばれる一意の文字列を送信するなどが考えられます。 このようなセッション情報は、テスト スクリプトで更新しなければ、後のテスト実行で問題の原因となる場合があります。

再生テスト実行を元の記録テスト実行と比較して、古いセッション情報が見つかった場合には、今後テストを実行するときのために、その情報を動的変数に置き換える必要があります。 そうしなければ、テスト スクリプトは無効なセッション ID などのセッション情報を渡してしまいます。

TrueLog Explorer は、カスタマイズに関係する差分を黄色で、カスタマイズが必要ない関係のない差分を青で示します。

注: Silk Performer の記録技術では、静的セッション情報を含まない、コンテキストを持つスクリプトが生成されるため、ほとんどのアプリケーションでは、セッション処理のカスタマイズは必要ありません。 そのため、自分のテストを分析して何も問題が検出されなければ、セッション処理のカスタマイズを省略して、ユーザー データのカスタマイズに進むことができます。

注: ここで紹介する例は具体的に Web アプリケーションに関するものですが、データベース アプリケーションや XML アプリケーションにも同じ原則を適用できます。 Oracle Forms、SAPGUI、Citrix、ターミナル エミュレーション、TCP/IP、および UDP の記録/再生 TrueLog は、TrueLog Explorer を使って比較できますが、これらの種類のアプリケーションでセッション処理のカスタマイズを行うことはできません。

セッション処理をカスタマイズする状況の判断

スクリプト内の WebPageUrl 呼び出しが、クエリ文字列の一部にセッション ID を含む URL を使用する場合、このスクリプトが再生されると、それと同じハードコードされた静的なセッション ID がサーバーに送信されます。 しかし、この セッション ID では再生セッションを正しく識別できません。 これは、以前の記録セッションを表しているに過ぎないため、スクリプトを再生するとエラーが発生します。

スクリプト内の静的なセッション ID を、実行時に生成される動的な値に置換しないと、Web アプリケーションでは、セッションが期限切れです。 ログイン画面に戻ってください。などのエラーが発生します。

Silk Performer のセッション処理方法の性質上、ほとんどの Web アプリケーションでは、セッション処理のカスタマイズは必要ありません。 まれに、手動によるセッションのカスタマイズが必要になった場合でも、TrueLog Explorer を利用して、簡単にその作業を実行できます。

Silk Performer では、ハードコードされたセッション ID を手動で処理する必要性が減るように、次の 2 つのセッション ID 処理方法を使用しています。

クッキー管理

Silk Performer は、クッキーを使用してセッション情報を交換するサーバーに対して、動的なセッション ID 値を自動的に処理します。 Silk Performer は、ブラウザのクッキー管理を正確にエミュレートするので、ブラウザと同じように、サーバーにクッキーを送信できます。このため、手動でやり取りをして、状態を維持する必要がありません。

ページ ベースのブラウザ レベル API

記録の際にページレベル API を使用する (これがデフォルト設定) と、WebPageLink や WebPageSubmit などの、コンテキストのある Web API 関数呼び出しを生成するスクリプトが提供されます。 コンテキストを持つ Web API 呼び出しは、HTTP レベルではなく、HTML レベルで動作します。したがって、URL をパラメータとして使用しません。 コンテキストを持つ API 呼び出しに対しては、手動によるセッションの

カスタマイズは必要ありません。 Silk Performer のページレベルの API が使用されるのは、**ワークフロー - プロジェクトの概要設定** ダイアログ ボックスの [アプリケーションの種類] に [Web ビジネス トランザクション (HTML/HTTP)] が選択されている場合です。

注: Silk Performer の低レベルのブラウザベースの API よりも、Silk Performer のページレベルの API を使用することを強くお勧めします。

HTML を動的に生成するためにクライアント側の JavaScript を利用している場合は、Silk Performer の Recorder が HTML のコンテキストを失って、コンテキストのない Web API 呼び出しが生成されることがあります。 コンテキストなしの Web API 呼び出し (WebPageUrl、WebPageForm など) には、パラメータとして URL が含まれます。 このようなまれなケースでは、スクリプトにハードコードされたセッション ID が含まれています。このようなセッション ID は、Web API 呼び出しの URL パラメータ内や、スクリプトの DCLFORM セクションで宣言されているフォーム フィールド内で見つけることができます。

例

カスタマイズする必要がないコンテキストを持つスクリプトの例を以下に示します。

```
transaction TMain
  begin
    WebPageUrl("http://myHost/ShopIt/"); // first call 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:
                                              //
LGIJALLCGEBMIBIMFKOEJIMM2
                                              // recognized as a
hidden
                                              // form field, the
value is
                                              // taken from the
actual
                                              // HTML form field.
    "name"              := "Tester",          // changed
    "New Name Button" := "" <USE_HTML_VAL>; // unchanged value:
"Enter"
```

例

コンテキストなしの関数を含むスクリプトの例 (カスタマイズが必要な静的なセッション データが DCLFORM セクションに含まれている) を以下に示します。

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

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

```
SHOPIT_MAIN_ASP002:
   "SessionID"          := "LGIJALLCGEBMIBIMFKOEJIMM2",
   "name"                 := "Tester",
   "New  Name  Button" := "Enter";
```

セッション処理のカスタマイズ

解析関数に指定した値を使用して、TrueLog Explorer で新しい記録ルールを作成できます。 たとえば、セッション カスタマイズ解析関数から記録ルールを作成すると、そのアプリケーションの記録が可能になり、セッションのカスタマイズの問題を完全に避けることができます。

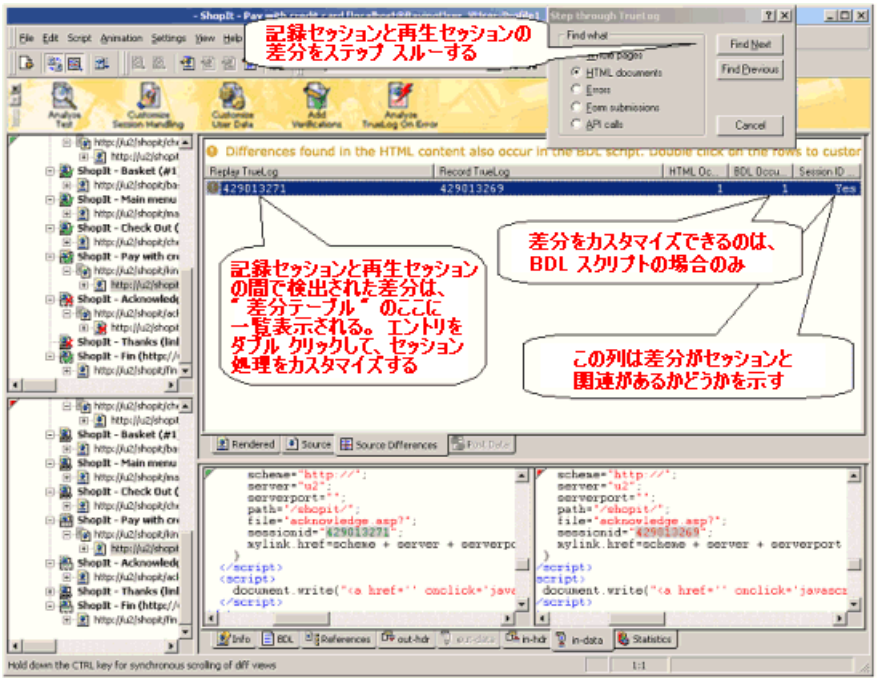
1. メニュー ツリーで TrueLog を選択します。
2. ワークフロー バーの **セッション処理のカスタマイズ** をクリックします。 **ワークフロー - セッション処理のカスタマイズ** ダイアログ ボックスが開きます。
3. **差分の検索** リンクをクリックして、差分テーブルを **ソース差分** ページに表示します。

   この手順で、再生セッション中に記録セッションと異なる (または動的な) 情報をサーバーが返した箇所が明らかになります。 このような情報は、変数に置き換える必要があります。

   注: 対応する記録 TrueLog がまだ開かれていない場合には、対応する記録 TrueLog を開くかどうかを確認するダイアログ ボックスが表示されます。

   a) **OK** をクリックします。
4. **TrueLog のステップ スルー** ダイアログ ボックスを使って、HTML サーバー レスポンスをステップ スルーします。

   記録レスポンスは、対応する再生レスポンスと並んで表示されます。 解析が必要なのは、テスト スクリプトに静的情報が含まれていて、それがサーバーに送り返されていることを示す差分だけです。 たとえば、e コマース サイトの在庫切れが原因で再生セッションと記録セッションの差分が発生する可能性があります。 そのような差分は、セッションに関係がないため、スクリプト カスタマイズの対象とはなりません。



ヒント: ソース差分 ページの列見出しの上にマウス ポインタを移動させると、各列に含まれる要素を説明する、便利なツールチップが表示されます。

5. **ソース差分** ページに一覧表示されたエラーの 1 つをダブルクリックします。 **解析関数の挿入** ダイアログ ボックスが、境界値が既に挿入された状態で表示されます。 境界値を手動で探して入力する必要はありません。
6. 省略可能*:* **記録ルールの作成** をクリックして、解析関数の値に基づいた記録ルールを作成します。 これにより、今後は、セッションのカスタマイズについて再度心配する必要なく、そのアプリケーションを記録できます。
7. **解析関数の挿入** ダイアログ ボックスに戻って、**OK** をクリックします。
8. スクリプトの変更が成功したら、ワークフロー バーの **セッション処理のカスタマイズ** をクリックします。
9. **スクリプトの試行** をクリックして、セッション処理を変更した結果、スクリプトが正しく実行されるかどうかを確認します。 スクリプトの試行が新たに開始されます。
10. その後のテスト実行の結果を分析して、セッション処理のカスタマイズが成功したかどうかを確認します。

学習機能付き Recorder

Silk Performer の学習機能付き Recorder は、クライアント側のスクリプトに強く依存したアプリケーションを適切にサポートします。 このレコーダは、高度なコンテキスト管理技術を備えており、ページ コンテキストを維持する Web スクリプトを生成できます。これは、JavaScript の関数によって実行されるページ リクエストにも対応しています。

ページ コンテキストを維持することは、Web ページを呼び出したリンクやフォーム送信のコンテキスト内で、そのページがリクエストされることを意味します。 このようなコンテキストを持つ Web ページ リクエスト関数は、URL だけでなく、リンクやフォーム名によって Web ページを参照します。 このため、Silk Performer はリンクやフォーム送信を Web サーバーのトラフィックを通じて解析し、コンテンツを呼び出したページを識別します。

解析関数

解析関数は、負荷テストの実行中に状態情報を保持し、実際のユーザー データの入力をシミュレートするために使用できます。 解析関数 (BDL 変数に格納される) は、サーバー レスポンス内のセッション関連の値を読み取ります。 そして、解析した値をパラメータ化されたデータとして使用します。このデータは、ポストされるクエリ文字列や URL データの一部として、サーバーに送り返されます。

注: 解析関数は、指定した入力値が存在することを単にチェックするだけの検証関数とは異なります。

解析関数の概要

解析機能は、一般に、以下のようなタスクのために使用されます。

- スクリプト内の静的なセッション ID を、状態情報を保持する動的なセッション ID に置換する。
- 検証関数だけでは実現できない強力なコンテンツ検証をスクリプト内に作成する。 たとえば、データベース テーブルの 3 行目 2 列目の値が、1 行目 2 列目の値と 2 行目 2 列目の値の和に等しいかどうかを、解析関数によって検証できます。 それには、この 3 つの値を抽出して比較する解析関数をスクリプト内に生成します。
- サーバーから返されたデータに応じて、テスト スクリプトの一部を条件付きで実行する。 たとえば、HTTP リクエストが、<nnn> 項目が見つかりましたという結果を含む HTML ページを返すとします。 <nnn> の値に応じて異なるアクションを実行する必要があります。 そのために、トランザクションを次のように設計します。
  - <nnn> = 0 の場合は終了する。
  - <nnn> = 1 の場合は詳細ページにリンクする。

- <nnn> が 1 より大きい場合は次のページにリンクする。

これを実現するには、この HTML ページから <nnn> の値を抽出し、その値に基づいてスクリプトに記述されたアクションを実行しなければなりません。

TrueLog Explorer を利用すると、**レンダリング** ビューと **ソース** ビューで、ビジュアルに解析関数を挿入できます。 TrueLog Explorer によって、解析関数が自動的にスクリプト内に生成されるので、手動でコードを記述する必要はありません。 TrueLog Explorer では、アプリケーションのライフ サイクル中の任意の時点でコントロール値を解析するため、ユーザーのスクリプトに解析機能を追加するウィザードが提供されます。 解析機能を追加するには、解析するコントロールの **値** 列で右クリックし、関連するコンテキスト メニューを選択します。

HTML コンテンツの解析関数

HTML コンテンツの解析関数は、レンダリングされた HTML コンテンツに適用できます。 これらは、TrueLog Explorer の **レンダリング** ビューで適用されます。 HTML の解析関数を利用すると、Web ブラウザで表示可能な HTML コンテンツを解析できます。

利用可能な HTML コンテンツの解析関数を以下に示します。

- WebParseHtmlBound および WebParseHtmlBoundEx
- WebParseHtmlBoundArray (TrueLog Explorer では未サポート)
- WebParseHtmlTitle
- WebParseTable
- WebParseResponseTag (TrueLog Explorer では未サポート)
- WebParseResponseTagContent (TrueLog Explorer では未サポート)

レスポンス データの解析関数

データ解析関数は、サーバーから返されたレスポンス データに適用できます。 HTML ドキュメントがサーバーから返された場合、それには、ドキュメントの完全なソース コードが含まれています。 これらの解析関数は、TrueLog Explorer の **ソース** ページで適用されます。

利用可能なデータ解析関数を以下に示します。

- WebParseDataBound および WebParseDataBoundEx (従来の WebParseResponseData の代替関数)
- WebParseDataBoundArray (TrueLog Explorer では未サポート)
- WebParseResponseHeader (TrueLog Explorer では未サポート)

注: TrueLog Explorer による視覚的解析は、TCP/IP および UDP ベースのアプリケーションでは現在利用できません。 ただし、WebTcpipRecv 関数からのレスポンス データを解析できる BDL 関数は利用できます。

Web アプリケーションのセッション処理

セッション ID は、さまざまな方法でクライアントに送信されます。 ほとんどの場合、セッション ID は、クッキー、ハイパーリンクの URL、埋め込みオブジェクトの URL、および HTML のフォーム フィールドに含まれています。 セッション ID は、同様に、クッキー、URL、および HTTP のポスト データに含めてサーバーに送り返されます。 以下に例を示します。

例：セッション情報がクッキーに含まれる場合：

クライアントに送信される情報:

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

サーバーに返される情報:

```
Cookie: SessionID=LGIJALLCGEBMIBIMFKOEJIMM
```

例：セッション情報が URL に含まれる場合：

クライアントに送信される情報:

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

サーバーに返される情報:

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

例：セッション情報が、非表示のフォーム フィールドに含まれる場合：

クライアントに送信される情報:

```
<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>
```

サーバーに返される情報:

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

セッション処理のカスタマイズ手順

セッション処理のカスタマイズは、通常、標準的な手順に従います。 どのセッション ID の場合も、以下に示す高レベルの手順に従う必要があります。

1. セッション ID をカスタマイズする必要があるかどうかを決定します。
2. そのセッション ID がサーバーからクライアントへ送信された最初のレスポンス (スクリプト内の Web API 呼び出し) を検索します。
3. そのレスポンス内のセッション ID を抽出して変数に代入します。
4. スクリプト内の、ハードコーディングされているセッション ID の出現箇所すべてをその変数に置換します。

セッション ID の識別

セッション ID は、記録セッションと再生セッションで異なりますが、記録 TrueLog と再生 TrueLog を比較すれば、セッション ID を簡単に識別できます。

セッション ID は、TrueLog Explorer を利用して簡単に見つけることができます。 ワークフロー バーの **セッション処理のカスタマイズ** をクリックすると、開いている再生 TrueLog に関連付けられた記録

TrueLog ファイルが自動的に表示されます。 記録 TrueLog と再生 TrueLog を 1 行ごとに読むことによって、記録スクリプトと再生スクリプト間の HTML レスポンスの違いを簡単に識別できます。

注: ソース差分 ビューに差分リストが自動的に表示されるので、手動で 1 行ごとに読む必要はありません。

記録セッションからのサーバー レスポンス データ内のフレーズの違いが、Web API 呼び出しの URL や、スクリプトの DCLFORM セクションに含まれるフォーム フィールドでも見つかった場合は、セッション ID をカスタマイズする必要があります。

記録セッションから返されたデータと、再生セッションから返されたデータとの差分すべてが、セッションのカスタマイズに関連するとは限りません。 セッションのカスタマイズを検討する必要があるのは、記録セッションと再生セッションとの差分がスクリプトに埋め込まれている場合だけです。 これは、**ソース差分** ビューの黄色の警告メッセージと **BDL 出現数** 列 (0 より大きい値である場合) によって示されます。

ソース差分 ビューでは、カスタマイズが必要だと思われる差分にはオレンジ色の感嘆符が付きます。 カスタマイズが必要ないと思われる差分には、青い疑問符が付きます (HTML 内のそのような文字列の出現箇所を置換することはお勧めしません。そのような文字列用の解析関数を挿入することはできますが、置換は行われません)。 青い感嘆符は、情報コメントのみの場合です。

最初のレスポンス (Web API) を検索する

セッション処理のカスタマイズでは、セッション ID がサーバーからクライアントへ送信された最初のレスポンス (スクリプト内の Web API 呼び出し) を検索することが必要になります。

1. メニュー ツリーから再生 TrueLog を選択します。
2. ツールバーの **比較モード** をクリックします。
3. ツールバーの **差分モード** をクリックします。 **ソース差分** ビューに、TrueLog 間で識別された差分が一覧表示されます。
4. **ソース差分** ビューのリストを右クリックして、**BDL スクリプト内の最初の出現箇所に移動** または **HTML 内の最初の出現箇所に移動** を選択します。

セッション ID を解析および置換する

セッション ID や、セッション ID を返す BDL スクリプトの Web API 呼び出しを識別できたら、そのセッション ID を Web API 呼び出しへのレスポンスから抽出する必要があります。 ハードコードされたセッション ID の出現箇所すべてを、変数で置換する必要があります。 TrueLog Explorer は、この作業をサポートするので、手動でスクリプトをカスタマイズする必要はありません。 TrueLog Explorer が、解析と置換のための適切なコードを自動生成します。

セッション ID を解析および置換する

1. メニュー ツリーから再生 TrueLog を選択します。
2. **セッション処理のカスタマイズ** をクリックします。 **ワークフロー - セッション処理のカスタマイズ** ダイアログ ボックスが開きます。
3. **差分の検索** をクリックします。 TrueLog Explorer を利用して、関連する記録 TrueLog を見つけて表示するかどうかを確認するダイアログ ボックスが表示されます。
4. **はい** をクリックします。 関連する記録 TrueLog が **比較ビュー** に表示され、**TrueLog のステップ スルー** ダイアログ ボックスが開きます。

   注: ソース差分 ビューでは、カスタマイズが必要となり得る記録データと再生データの差分が明らかになります。 **TrueLog のステップスルー** ダイアログ ボックスを使用して、コンテンツの [ページ全体] を走査し、カスタマイズが必要な差分が見つかるまで API ノード間を進みます。
5. **ソース差分** ビューにあるリストをダブルクリックすると、**解析関数の挿入** ダイアログ ボックスが開きます。
6. リストを右クリックして、**セッション処理のカスタマイズ** を選択します。 **解析関数の挿入** ダイアログ ボックスが開きます。

7. **解析関数の挿入** ダイアログ ボックスには、解析関数を調整するためのパラメータが表示されます。 デフォルト設定でも問題はありませんが、以下の設定を調整することもできます。 :
   a) **大文字/小文字を区別する** - 解析関数で大文字と小文字を区別したい場合は、このオプションをオンにします。
   b) **変数名** - 解析関数の結果を受け取るための変数の名前を入力します。
   c) **左境界** - このフィールドには、指定したテキストの左境界 (指定したテキストの直前に位置するテキスト) が表示されます。
   d) **右境界** - このフィールドには、指定したテキストの右境界 (指定したテキストの直後のテキスト) が表示されます。
   e) **スクリプトに出力文を挿入** - Web ページ呼び出しの後に、通知のための Print 文をスクリプトに挿入するには、**Print 文** をオンにします。 これによって、解析関数の結果が Silk Performer の **仮想ユーザー** 出力ウィンドウに出力されます。
   f) (Print 文を実行して **仮想ユーザー 出力** ウィンドウに値を書き込むほかに) デバッグしやすくするために解析値を出力ファイルに書き込む場合は、**Writeln 文** (*write line* 文) をオンにします。 出力ファイルを生成すると負荷テストの時間測定値が変化するので、これらのファイルはデバッグのためだけに使用し、完全な負荷テストでは生成しないでください。
   g) **変数に置き換える** - このチェック ボックスをオンにすると、解析済みテキストのすべての出現回数を指定した変数に置き換えます。
8. 省略可能*:* 指定した値に基づいて記録ルールを作成したい場合は、**記録ルールの作成** をクリックします。 記録ルール作成の詳細については、『Silk Performer ヘルプ』を参照してください。
9. **OK** をクリックします。 これで、選択したセッション ID の処理をカスタマイズするために必要な、解析と置換の文が生成されます。
10. [スクリプトの試行] を実行してカスタマイズを検証し、セッション処理の問題が解決したことを確認します。 緑色のチェックマークが付いた API ノードは、スクリプトの試行の実行中にセッション エラーが発生しなかったことを表します。 赤い X マークは、まだスクリプト再生エラーが存在することを表します。

解析と置換の文の例

以下のサンプル コードは、セッション処理をカスタマイズするために必要な解析と置換の文の例を示しています。

注: 元のセッション ID はコメントアウトされていますが、スクリプト内に残っています。

TrueLog Explorer によって生成された WebParseDataBoundEx 関数が、それに続く WebPageLink 呼び出しから返された HTML ドキュメント内の、境界 name=¥" と ¥" に挟まれた値を抽出します。 次に、実際の値を *sSessionInfo1* 変数に格納します。 解析関数は、レスポンス データが解析される Web API 呼び出しの前に指定する必要があります。 デバッグ用に、TrueLog Explorer は、解析結果を Silk Performer の **仮想ユーザー出力** ウィンドウに出力されます。Print セッション ID の変数は、*sFormSid1* フォーム フィールド変数に代入されます。これは、定数値のセッション ID (858891446) の代わりに、DCLFORM スクリプト セクションで直接使用されます。

例

```
transaction TMain
begin
  ...
  WebParseDataBoundEx(sSessionInfo1, STRING_COMPLETE,
    "name=¥"", 3, "¥"", WEB_FLAG_IGNORE_WHITE_SPACE |
    WEB_FLAG_CASE_SENSITIVE, 1);
  WebPageLink("Check out", "ShopIt - Check Out"); // Link 3
  Print("sSessionInfo1: " + sSessionInfo1);

  ...

  sFormSid1 := sSessionInfo1;
```

```
  WebPageUrl("http://lab3/ShopItV60/kindofpayment.asp",
    "kindofpayment.asp",  SHOPITV60_KINDOFPAYMENT_ASP004);
  ...
end  TMain;

dclform
  ...
  SHOPITV60_KINDOFPAYMENT_ASP004:
    ...
//      "sid"                              :=  "858891446";
    "sid"                              :=  sFormSid1;
  ...
```

手動での差分の選択

ソース差分 ビューを利用する代わりに、**ソース** ビューまたは **受信データ** ビューで、差分を手動で選択すると、フレーズの抽出方法をきめ細かく制御できます。

たとえば、セッション ID が 1 つの HTML ページの複数の場所に埋め込まれている場合に **ソース差分** ビューを使用すると、最初に出現したフレーズのみが抽出されます。 これは、解析に使用した境界が適切でない場合でも同じです。

抽出するフレーズを手動で選択することによって、より正確な境界を選択できます。

手動で差分を選択する

1. メニュー ツリーから再生 TrueLog を選択します。
2. ツールバーの **比較モード** をクリックします。
3. **ソース** ビューまたは **受信データ** ビューのいずれかで、TrueLog 間の差分を識別します。
4. 記録 TrueLog で差分フレーズを右クリックします。 選択したフレーズが、BDL スクリプトに含まれる静的なデータの場合は、コンテキスト メニューが表示されます。
5. **セッション処理のカスタマイズ** を選択します。 **解析関数の挿入** ダイアログ ボックスが開きます。

必要に応じて解析関数を構成してください。

スクリプト内の解析関数

解析関数は、スクリプト内の、レスポンス データの解析/検証を開始する Web API 呼び出しの前の位置に挿入する必要があります。 各 Web API 呼び出しの前には、複数の解析/検証関数を指定できます。

解析/検証関数の順序は関係ありません。 この規則の例外には、フラグ WebParseDataBound を使用する WebVerifyDataBound および WEB_FLAG_SYNCHRON があります。

例

検証関数を利用した BDL スクリプトの例を以下に示します。

```
WebVerifyHtml("Proper equipment leads to a successful trip", 1, ...);
WebPageLink("ShopIt");
```

HTTP 解析ルールの使用

HTTP 解析の例は、Silk Performer の記録ルールの機能の概要を示すために設計されました。

HTTP 解析ルールは、動的に値を変更するために、Recorder が解析関数 WebParseDataBoundEx() を生成するタイミングと、解析結果を置換する場所を指定します。 解析ルールを利用すると、Recorder に作業スクリプトを自動生成させることができます。 また、TrueLog Explorer を使用して、視覚的にセッションをカスタマイズする必要が通常はなくなります。

手動での記録ルールのスクリプト記述に関する詳細は、『*Silk Performer Advanced Concepts Book*』の「*Rule Based Recording*」の章を参照してください。

Web アプリケーションの解析ルールの例

サンプル Web アプリケーション ShopIt は、セッション ID を含むフォーム定義によって、Silk Performer Recorder にコンテキストなしの WebPageUrl() 関数を生成させる意図で作成したものです。 これは、JavaScript で URL を組み立てることによって実現されています。 記録ルールを使用せずに ShopIt を記録すると、ハードコーティングされたセッション ID を含むスクリプトが生成されます。

この HTTP 解析ルールの例では、セッション ID を含むフォーム定義によって、コンテキストなしの WebPageUrl() 関数が生成されています。 [スクリプトの試行] を実行すると、ハードコードされたセッション ID が原因で再生エラーが発生します。

TrueLog Explorer のセッション処理カスタマイズ機能では、次のようにスクリプトを変更することによって、この問題を解決します。

例

```
var
  sFormSid1       : string (100);
  sSessionInfo1   : string (100);

  ...

  WebParseDataBoundEx(sSessionInfo1, STRING_COMPLETE,
    "name=¥"", 5, "¥"", WEB_FLAG_IGNORE_WHITE_SPACE, 1);
  WebPageLink("Check out", "ShopIt - Check Out"); // Link 3
  Print("sSessionInfo1: " + sSessionInfo1);

  ...

  sFormSid1 := sSessionInfo1;
  WebPageUrl(sParsedUrl,
    "Unnamed page", SHOPITV60_KINDOFPAYMENT_ASP004);

  ...

dclform
  ...
  SHOPITV60_KINDOFPAYMENT_ASP004:
    "choice"                    := "CreditCard",
    "price"                     := "115.8",
//      "sid"                       := "858891471";
    "sid"                       := sFormSid1;
```

カスタム Recorder の詳細

カスタマイズが完了して、スクリプトが正しく実行されるようになりました。 しかし、今後記録されるスクリプトでも、毎回カスタマイズを行わなければならないという問題が残ります。

HTTP 解析ルールを利用すると、今後は、この種のカスタマイズを Silk Performer の Recorder に自動的に行わせることができます。つまり、手動の操作なしで、記録スクリプトを自動生成できます。

それには、各セッション ID の解析方法を調査する必要があります。 TrueLog Explorer によって提示されたカスタマイズを見ると、セッション ID が最初に発生した API 呼び出しがわかります。また、各セッション ID を解析するために使用できる境界もわかります。

TrueLog Explorer を使用して、HTML コード内でセッション ID が最初に出現する場所を見つけます。

```
<script LANGUAGE="JavaScript">
  function doProcess(mylink)
```

```
  {
    scheme="http://";
    server="lab3";
    serverport="";
    path="/ShopItV60/";
    file="kindofpayment.asp?";
    name="858891471";
      price="115.8";
    choice="CreditCard";
    mylink.href=scheme + server + serverport + path + file + "choice=" +
      choice + "&price=" + price + "&sid=" + name;
  }
</script>
```

TrueLog Explorer によって識別された左境界 name=" と 右境界 " は、セッション ID を抽出するためには妥当な選択です。

これで、HTTP 解析ルールの初期バージョンを Recorder 用に書き込むことができます。

```
<?xml version="1.0" encoding="UTF-8"?>
<RecordingRuleSet>

  <HttpParsingRule>
    <Name>ShopIt V5.1 Session Id</Name>

    <Search>
      <SearchIn>Body</SearchIn>
      <LB>
        <Str>name=&amp;quot;</Str>
      </LB>
      <RB>
        <Str>&amp;quot;</Str>
      </RB>
    </Search>

    <ScriptGen>
      <VarName>ShopItSessionId</Varname>
    </ScriptGen>

  </HttpParsingRule>

</RecordingRuleSet>
```

ShopIt のセッション ID は、HTTP のレスポンス ヘッダーには存在しないので、レスポンス ボディだけを検索するように指定します (Search¥SearchIn 属性を使用)。

左境界 (Search¥LB¥Str 属性で指定) を検索して、セッション ID を見つけるように設定します。

注: XML では、引用符 (") は、特殊文字列 " を使用してエンコードされます。

セッション ID の終りには、単一引用符が付きます。 これは、Search¥RB¥Str 属性で指定します。 ここで、再び、引用符がエンコードされています。

最後に、解析結果を格納する変数の名前を定義しています。 この名前は、ScriptGen¥VarName 属性を使用して指定します。

このルール ファイルをすべてのプロジェクトでグローバルに利用できるようにするには、Silk Performer の Include ディレクトリに保存します。 ファイル名は任意の長さにできますが、ファイルの拡張子は .xrl にする必要があります。 この記録ルールを 1 つのプロジェクトだけで使用する場合は、このファイルを、特定の Silk Performer プロジェクトの Documents ディレクトリに保存します。

手動での記録ルールのスクリプト記述に関する詳細は、『*Silk Performer Advanced Concepts Book*』の「*Rule Based Recording*」の章を参照してください。

検証を追加する

このセクションでは、HTML ページ タイトル、HTML コンテンツ、HTML テーブル、および HTML ソース コードについての検証チェックを作成する方法の概要を説明します。

注: TCP/IP および UDP ベースのアプリケーションでは、現在、コンテンツの検証を利用できません。

検証の概要

TrueLog Explorer を使用して、HTML ページ タイトル、HTML コンテンツ、HTML テーブル、および HTML ソース コードについての Web コンテンツ検証チェックを作成することができます。 検証したいテキストを HTML ソース コードで、またはレンダリングされた HTML ビューで直接に選択するだけで、必要なすべての検証関数が生成され、自動的に BDL スクリプトに挿入されます。

視覚的データ検証

TrueLog Explorer では、2 つの方法でコンテンツ検証を行うことができます。 簡単なのは、単純なポイント アンド クリック インターフェイスを使ってコンテンツ検証チェックを作成する方法です。

レスポンス データ検証

レスポンス データ検証は、サーバー機能を検証するための強力ですが複雑な方法です。 視覚的データ検証と同様に、レスポンス データ検証もポイント アンド クリック インターフェイスでセットアップします。 視覚的データ検証では、レンダリングされた HTML のビューで作業を行い、アプリケーションがエンド ユーザーに対して表示するコンテンツを検証します。 レスポンス データ検証では、HTML コードや XML コードなど、サーバーから返される、エンド ユーザーには表示されないデータを検証します。

選択された関数に応じて、TrueLog Explorer は、サーバーから返されたレスポンス データ全体に、または境界を指定して選択した部分だけに適用する、レスポンス データ検証関数を生成できます。

検証チェック

テスト スクリプトに検証チェックを埋め込む利点について説明します。

検証チェックを使う理由

アプリケーション エラーが HTTP レスポンスのエラーとして返されることは多くありません。 それよりも、アプリケーションが不適切なデータ値やエラー メッセージを HTML コンテンツに組み込んで返す方が一般的です (サーブレット例外が発生しました あるいは サーバーがビジー状態です といったエラー メッセージなど)。 HTTP ステータス コードをチェックしてもこの種のエラーは発見できないため、標準の HTTP エラー以外のものをチェックする検証関数がテスト スクリプトに組み込まれていない限り、アプリケーション エラーは見過ごされがちです。

検証がテスト スクリプトに組み込まれると、テストは単純な負荷テストから複合的な負荷/機能テストに進化します。 そのようなスクリプトは、大規模な負荷テスト シナリオで使用してもそれほどパフォーマンスは劣化しません。 この機能を使うことで、他の負荷テスト ツールでは検出できない種類のエラーを検出することができます (負荷時にしか発生しないエラーは、標準の負荷テスト スクリプトで検出できないため)。

TrueLog Explorer では、以下の方法でテスト スクリプトに検証機能を追加することができます。

- Silk Performer Recorder を有効化して、記録時に検証チェックを自動生成する

- ユーザーが **レンダリング** ビューでポイント アンド クリック インターフェイスを使って検証チェックを視覚的に適用する。 BDL コードを編集する必要はありません。 TrueLog Explorer が自動的に検証関数をスクリプトに追加してくれます。
- 検証関数のコードを手作業でスクリプトに挿入して直接的に拡張する

記録中に自動的に検証を生成する

Silk Performer スクリプトによる記録中に、自動的に検証を生成できるようにするには:

1. Silk Performer 内から、**設定** > **アクティブ プロファイル** を選択します。 **プロファイル - [<プロファイル名>] - シミュレーション** ダイアログが表示されます。
2. ショートカット リストから **記録** > **Web** を選択し、**検証** タブをクリックします。
3. **記録** セクションで、**タイトル検証を記録する** および **ダイジェスト検証を記録する** チェック ボックスをオンにします。
4. **OK** をクリックします。

TrueLog Explorer の検証チェック

TrueLog Explorer では、コンテンツ検証関数をテスト スクリプトに挿入することで、テスト中にアプリケーション サーバーから返されたコンテンツが正確かどうかを検証することができます。

TrueLog Explorer では、アプリケーションのライフ サイクル中の任意の時点でコントロール値を検証するため、ユーザーのスクリプトに検証機能を追加するウィザードが提供されます。 検証機能を追加するには、検証するコントロールの **値** 列で右クリックし、関連するコンテキスト メニューを選択します。

検証したいコンテンツを特定したら、必要な検証関数をすべて生成し、テスト スクリプトに自動的に挿入することができます。 検証対象のコンテンツ (レンダリングされた HTML、HTML ソース コード、SQL コマンド、Oracle Forms などに含まれるもの) を特定するには、そのコンテンツを選択して右クリックします。

検証は、TrueLog Explorer の **レンダリング** および **ソース** ビューで、以下のいずれかの方法で視覚的に適用することができます。

- **スクリプト** メニュー
- **検証の追加** ダイアログ ボックス
- **レンダリング**、**ソース**、および **フォーム コントロール** の各ビューのコンテキスト メニュー
- ワークフロー バー

再生中に検証チェックを有効にする

スクリプト再生中に有効または無効にできる検証チェックのリストに対する検証設定オプションを確認します。

ヒント: スクリプトのプロファイル設定は、WebSetOption という BDL 関数を使って上書きできます。

1. Silk Performer 内から、**設定** > **アクティブ プロファイル** を選択します。 **プロファイル - [<プロファイル名>] - シミュレーション** ダイアログが表示されます。
2. ショートカット リストから **再生** > **Web** を選択し、**検証** タブをクリックします。
3. **HTML/XML** と **データ** 領域で、再生中に有効化する検証チェック対象のチェック ボックスをオンにします。

コンテンツ検証関数を挿入する

1. 分析または変更したい TrueLog を開きます。
2. 検証したいコンテンツ（テキストや画像など）を含む TrueLog API ノードを選択します。

3. 検証したいコンテンツを **ソース** ページで選択します。

   注: ページ タイトル検証関数およびページ ダイジェスト検証関数の場合は、このステップは不要です。
4. ワークフロー バーで、**検証の追加** をクリックします。 **ワークフロー - 検証の追加** ダイアログ　ボックスが開きます。
5. 有効になっている検証を以下から 1 つ選択します。

   - ページ タイトルの検証
   - 選択したテキストの検証
   - HTML テーブル内で選択したテキストの検証
   - ダイジェストの検証
6. 表示されるダイアログ ボックスに必要事項を入力します。

   検証関数を BDL スクリプトにどう挿入するかを指定します。

   注: [左境界] と [右境界] は、自動的に識別されます。
7. BDL スクリプトに追加したい検証それぞれについて、この作業を繰り返します。
8. **ワークフロー - 検証の追加** ダイアログ ボックスで **はい** をクリックします。 スクリプトの試行が開始します。
9. 検証が正常に合格したことを確認します。

   検証を含む API ノードには、青い "V" 印が表示されます。

以下の作業が終了したときには、負荷テスト スクリプトは、エラーなしで実行できるようになっているはずです。

- アプリケーションがセッション情報やユーザー入力データをどう処理するかのカスタマイズ
- 必要な検証関数の挿入
- 必要であれば、Silk Performer での BDL スクリプトを手動で編集します。

HTML 検証関数

HTML 検証では、レンダリングして表示された Web コンテンツをチェックします。 このチェックでは、Web ブラウザに表示されたテキスト ベースのコンテンツを検証します。 **レンダリング** ビューで以下の HTML 検証関数を適用できます。

- WebVerifyHtmlTitle- ページ タイトルの検証
- WebVerifyHtml- テキスト ベースのコンテンツの検証
- WebVerifyHtmlBound(Ex)- 境界内のコンテンツの検証
- WebVerifyTable- HTML テーブルに含まれるコンテンツの検証
- WebVerifyHtmlDigest- HTML ページのダイジェストの検証

タイトル検証関数を生成する

WebVerifyHtmlTitle 関数は、選択された HTML ページのタイトル (HTML <title> タグのコンテンツ) をチェックします。 タイトル検証を適用する際にテキストを選択する必要はありません。

1. メニュー ツリーで、検証したい Web ページを選択します。
2. 以下の手順の 1 つを実行します。

   - **スクリプト** > **ページ タイトルの検証** を選択します。
   - **検証の追加** をクリックし、**ワークフロー - 検証の追加** ダイアログ ボックスで **ページ タイトルの検証** をクリックします。

   ページ タイトルが **定数値** テキスト ボックスに表示され、**定数値** オプション ボタンが選択されます。

3. 省略可能: 既存または新規のパラメータと照合して検証するには、**パラメータ値** オプション ボタンをクリックします。
   a) 参照 (**...**) ボタンをクリックし、パラメータを選択します。
   b) パラメータが存在しない場合には、**次へ** をクリックして **パラメータ ウィザード** を開き、新しいパラメータを作成します。
4. **ページ タイトル** リスト ボックスで、次のいずれかを選択します。
   - **等しい**
   - **異なる**
   - **含む**
   - **含まない**
5. チェック ボックスを使って、検証で大文字/小文字を区別するか、スクリプト全体のルールとして適用するかを指定します。
6. **深刻度** グループ ボックスで、検証から否定的な結果が返されたときに生成する深刻度を指定します。
7. **OK** をクリックします。 関数がテスト スクリプトに追加されます。

コード例

次のサンプル コード (太字部分) は、タイトル検証用に自動生成されたものです。

```
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");
   ...
```

テキスト検証関数を生成する

WebVerifyHtml 関数は、**レンダリング** ビューで選択されたテキストを検証します。 TrueLog Explorer は、選択されたテキストが HTML ドキュメント内で出現する回数を自動的に検知します。

1. メニュー ツリーで、検証したい Web ページを選択します。
2. **レンダリング** ビューで適切なテキストを選択します。
3. 以下の手順の 1 つを実行します。
   - **スクリプト** > **選択したテキストの検証** を選択します。
   - 選択したテキストを右クリックして、**選択したテキストの検証** を選択します。
   - **検証の追加** をクリックし、**ワークフロー - 検証の追加** ダイアログ ボックスから **HTML ページ内で選択したテキストの検証** を選択します。

   選択したテキストが **定数値** テキスト ボックスに表示され、**定数値** オプション ボタンが選択されます。
4. 省略可能: 既存または新規のパラメータと照合して検証するには、**パラメータ値** オプション ボタンをクリックします。
   a) 参照 (**...**) ボタンをクリックし、パラメータを選択します。
   b) パラメータが存在しない場合には、**次へ** をクリックして **パラメータ ウィザード** を開き、新しいパラメータを作成します。
5. 省略可能: 検証の許容度を上げたい場合には、以下の設定を変更します。

注: このダイアログ ボックスの設定は、現在のページについて検証が成功する値に自動的に設定されています。 検証の許容度を上げたい場合にのみ、これらの設定を変更します ("**ちょうど**" "**2**" 回を "**以上**" "**1**" 回に変える、大文字/小文字を区別しない、など)。

a) **このページでの出現回数** リスト ボックスで、次のいずれかを選択します。

- **ちょうど**
- **以上**
- **以下**

b) **回** テキスト ボックスに数値を入力します。

6. チェック ボックスを使って、検証で大文字/小文字を区別するか、スクリプト全体のルールとして適用するかを指定します。 結果の変数名が **結果の変数名** テキスト ボックスに表示されます。
7. 省略可能: **結果の変数名** を編集します。
8. **境界文字列を必要とする** チェック ボックスがオフになっていることを確認してください。
9. **深刻度** グループ ボックスで、検証から否定的な結果が返されたときに生成する深刻度を指定します。
10. **OK** をクリックします。 関数がテスト スクリプトに追加されます。

コード例

次のサンプル コード (太字部分) は、HTML テキスト検証用に自動生成されたものです。

```
transaction TMain
begin
  ...
  WebVerifyHtml("Hi Tester! ", 1,
WEB_FLAG_IGNORE_WHITE_SPACE |
    WEB_FLAG_EQUAL | WEB_FLAG_CASE_SENSITIVE, NULL,
    SEVERITY_ERROR, nVerifyHtmlResult1);
  WebPageSubmit("Enter", SHOPIT_MAIN_ASP001, "ShopIt - Main menu");
  ...
```

境界内のテキスト検証関数を生成する

WebVerifyHtmlBound(Ex) 関数は、**レンダリング** ビューで選択されたテキストを検証します。 TrueLog Explorer は、選択されたテキストの位置を特定する一意の境界を自動的に検出します。

注: Silk Performer では、 TrueLog Explorer は WebVerifyHtmlBoundEx の代わりに WebVerifyHtmlBound を使用します。

1. メニュー ツリーで、検証したい Web ページを選択します。
2. **レンダリング** ビューで適切なテキストを選択します。
3. 以下の手順の 1 つを実行します。

- **スクリプト** > **選択したテキストの検証** を選択します。
- 選択したテキストを右クリックして、**選択したテキストの検証** を選択します。
- **検証の追加** をクリックし、**ワークフロー - 検証の追加** ダイアログ ボックスから **HTML ページ内で選択したテキストの検証** を選択します。

選択したテキストが **定数値** テキスト ボックスに表示され、**定数値** オプション ボタンが選択されます。

4. 省略可能: 既存または新規のパラメータと照合して検証するには、**パラメータ値** オプション ボタンをクリックします。

a) 参照 (**...**) ボタンをクリックし、パラメータを選択します。

b) パラメータが存在しない場合には、**次へ** をクリックして **パラメータ ウィザード** を開き、新しいパラメータを作成します。

5. チェック ボックスを使って、検証で大文字/小文字を区別するか、スクリプト全体のルールとして適用するかを指定します。 結果の変数名が **結果の変数名** テキスト ボックスに表示されます。

6. 省略可能: **結果の変数名** を編集します。
7. **境界文字列を必要とする** チェック ボックスをオンにします。 これにより、**このページでの出現回数** の設定が無効になります。

   左境界と右境界が、**左境界** と **右境界** のテキスト ボックスに表示されます。 このダイアログ ボックスで境界の文字列を編集することはできません。
8. **深刻度** グループ ボックスで、検証から否定的な結果が返されたときに生成する深刻度を指定します。
9. **OK** をクリックします。 関数がテスト スクリプトに追加されます。

コード例

次のサンプル コード (太字部分) は、境界内の HTML テキスト検証用に自動生成されたものです。

```
transaction  TMain
begin
   ...
   WebVerifyHtmlBound("Name:",  "Address",  "Tester",
      WEB_FLAG_IGNORE_WHITE_SPACE  |
WEB_FLAG_CASE_SENSITIVE,
      NULL,  SEVERITY_ERROR,  bVerifyHtmlBoundSuccess2);
   WebPageSubmit("Submit",  SHOPIT_KINDOFPAYMENT_ASP002”);
   ...
```

HTML テキスト検証関数を生成する

WebVerifyTable 関数は、**レンダリング** ビューの HTML テーブル セルで選択されたテキストを検証します。 TrueLog Explorer は、HTML テーブルと、選択されたセルの行と列を自動的に特定します。

1. メニュー ツリーで、検証したい Web ページを選択します。
2. **レンダリング** ビューで、テーブル セル内の適切なテキストを選択します。
3. 以下のいずれかを行います。

   - **スクリプト** > **HTML テーブル内で選択したテキストの検証** を選択します。
   - **検証の追加** をクリックし、**ワークフロー - 検証の追加** ダイアログ ボックスで **HTML テーブル内で選択したテキストの検証** を選択します。

   選択した行と列が **検証内容** リスト ボックスに表示されます。 必要であれば、別の行や列を指定します。
4. リスト ボックスから次のいずれかを選択します。

   - **等しい**
   - **異なる**
   - **含む**
   - **含まない**

   選択したテキストが **値** テキスト ボックスに表示され、**値** オプション ボタンが選択されます。
5. 省略可能: 既存または新規のパラメータと照合して検証するには、**パラメータ値** オプション ボタンをクリックします。
   a) 参照 (**...**) ボタンをクリックし、パラメータを選択します。
   b) パラメータが存在しない場合には、**次へ** をクリックして **パラメータ ウィザード** を開き、新しいパラメータを作成します。
6. 検証で **大文字/小文字を区別する** かどうかを指定します。
7. **深刻度** グループ ボックスで、検証から否定的な結果が返されたときに生成する深刻度を指定します。
8. **OK** をクリックします。 関数がテスト スクリプトに追加されます。

コード例

次のサンプル コード (太字部分) は、HTML テーブル検証用に自動生成されたものです。

```
transaction  TMain
begin
  ...
  WebVerifyTable(7, 3, 1, "115.8",
    WEB_FLAG_IGNORE_WHITE_SPACE |
    WEB_FLAG_CONTAINS | WEB_FLAG_CASE_SENSITIVE,
    NULL, SEVERITY_ERROR, bVerifyTableSuccess1);
  WebPageLink("Add to basket", "ShopIt - Basket");
  ...
```

HTML ダイジェスト検証関数を生成する

WebVerifyHtmlDigest 関数は、Web ページ ダイジェストを計算し、計算済みのダイジェストに照らして検証します。 ダイジェストとは、レンダリングされた HTML ページ内の特定の文字に関して計算された、文字の出現頻度のテーブルです。 HTML ドキュメントがフレーム セットに含まれている場合、テキストの検証は、ページ全体ではなく、選択されたテキストを含むフレームだけを対象に行われます。

1. メニュー ツリーで、検証したい Web ページを選択します。
2. **レンダリング** ビューを選択します。
3. 以下の手順の 1 つを実行します。
   - **スクリプト** > **ページ ダイジェストの検証** を選択します。
   - **検証の追加** をクリックし、**ワークフロー - 検証の追加** ダイアログ ボックスで **ダイジェストの検証** をクリックします。

   ダイジェストの名前が **定数名** テキスト ボックスに表示されます。
4. 省略可能: **定数名** テキスト ボックスを編集します。
5. **検証対象** グループ ボックスで、検証を適用する対象を指定します。
   - **すべての文字**
   - **印刷可能な文字**
   - **英数字**
6. **深刻度** グループ ボックスで、検証から否定的な結果が返されたときに生成する深刻度を指定します。
7. **OK** をクリックします。 関数がテスト スクリプトに追加されます。

コード例

次のサンプル コード (太字部分) は、HTML ダイジェスト検証用に自動生成されたものです。

```
const
  HTML_DIGEST_1       :=
"¥h01910000000011D063032CCB7D023EFBBD020000000000000000000000000000000001
602010601010703010108010503020201"
"¥h030102010305020101050104030701030102050504020601070501 02";

transaction  TMain
begin
  ...
  WebVerifyHtmlDigest(HTML_DIGEST_1, 79, 0, 0, NULL,
    SEVERITY_ERROR);
  WebPageLink("Check out", "ShopIt - Check Out"); // Link 3
  ...
```

レスポンス データ検証関数

レスポンス データ検証関数は、サーバーから正しいレスポンス データが返されたかどうかを検証します。 Web アプリケーションの場合、データ検証の一環として包括的なソース コード チェックも行われます。

ソース ビューで、コードやテキストなどのデータを選択することによって、レスポンス データ検証関数を視覚的に適用できます。 TrueLog Explorer では、次のデータ検証関数を生成することができます。

- WebVerifyData- 選択したデータの検証
- WebVerifyDataBound(Ex)- 境界内の選択したデータの検証
- WebVerifyDataDigest- データ ダイジェストの検証

HTML データ検証関数を生成する

WebVerifyData 関数は、**ソース** ビューで選択されたコードやテキストなどのデータを検証します。 TrueLog Explorer は、選択されたデータが選択された HTML ドキュメント内で何回出現するかを自動的に検出します。

1. メニュー ツリーで、検証したい Web ページを選択します。
2. **ソース** ビューで該当するデータを選択します。
3. 以下の手順の 1 つを実行します。
   - **スクリプト** > **選択したテキストの検証** を選択します。
   - 選択したテキストを右クリックして、**選択したテキストの検証** を選択します。
   - **検証の追加** をクリックし、**ワークフロー - 検証の追加** ダイアログ ボックスから **HTML ページ内で選択したテキストの検証** を選択します。

   選択したテキストが **定数値** テキスト ボックスに表示され、**定数値** オプション ボタンが選択されます。
4. 省略可能: 既存または新規のパラメータと照合して検証するには、**パラメータ値** オプション ボタンをクリックします。
   a) 参照 (**...**) ボタンをクリックし、パラメータを選択します。
   b) パラメータが存在しない場合には、**次へ** をクリックして **パラメータ ウィザード** を開き、新しいパラメータを作成します。
5. 選択したテキストが出現する頻度を以下のように指定します。
   a) **このページでの出現回数** リスト ボックスで、次のいずれかを選択します。
      - **ちょうど**
      - **以上**
      - **以下**
   b) **回** テキスト ボックスに数値を入力します。
6. チェック ボックスを使って、検証で大文字/小文字を区別するか、スクリプト全体のルールとして適用するかを指定します。 結果の変数名が **結果の変数名** テキスト ボックスに表示されます。
7. 省略可能: **結果の変数名** を編集します。
8. **境界文字列を必要とする** チェック ボックスがオフになっていることを確認してください。
9. **深刻度** グループ ボックスで、検証から否定的な結果が返されたときに生成する深刻度を指定します。
10. **OK** をクリックします。 関数がテスト スクリプトに追加されます。

コード例

次のサンプル コード (太字部分) は、データ検証用に自動生成されたものです。

```
transaction  TMain
begin
  ...
  WebVerifyData("main.asp?from=welcome",  1,
```

```
    WEB_FLAG_IGNORE_WHITE_SPACE |
    WEB_FLAG_EQUAL | WEB_FLAG_CASE_SENSITIVE, 1,
    SEVERITY_ERROR, nVerifyDataResult1);
  WebPageUrl("http://myHost/shopit/", "ShopIt - Greetings");
  ...
```

境界内のデータ検証関数を生成する

WebVerifyDataBound(Ex) 関数は、**ソース** ビューで選択されたコードやテキストなどのデータを検証します。 TrueLog Explorer は、選択されたデータの位置を特定する一意の境界を自動的に検出します。

注: TrueLog Explorer は、WebVerifyDataBound の代わりに WebVerifyDataBoundEx を使用します。

1. メニュー ツリーで、検証したい Web ページを選択します。
2. **ソース** ビューで該当するデータを選択します。
3. 以下の手順の 1 つを実行します。
   - **スクリプト** > **選択したテキストの検証** を選択します。
   - 選択したテキストを右クリックして、**選択したテキストの検証** を選択します。
   - **検証の追加** をクリックし、**ワークフロー - 検証の追加** ダイアログ ボックスから **HTML ページ内で選択したテキストの検証** を選択します。

   選択したテキストが **定数値** テキスト ボックスに表示され、**定数値** オプション ボタンが選択されます。
4. 省略可能: 既存または新規のパラメータと照合して検証するには、**パラメータ値** オプション ボタンをクリックします。
   a) 参照 (**...**) ボタンをクリックし、パラメータを選択します。
   b) パラメータが存在しない場合には、**次へ** をクリックして **パラメータ ウィザード** を開き、新しいパラメータを作成します。
5. チェック ボックスを使って、検証で大文字/小文字を区別するか、スクリプト全体のルールとして適用するかを指定します。 結果の変数名が **結果の変数名** テキスト ボックスに表示されます。
6. 省略可能: **結果の変数名** を編集します。
7. **境界文字列を必要とする** チェック ボックスをオンにします。 これにより、**このページでの出現回数** の設定が無効になります。

   左境界と右境界が、**左境界** と **右境界** のテキスト ボックスに表示されます。 このダイアログ ボックスで境界の文字列を編集することはできません。
8. **深刻度** グループ ボックスで、検証から否定的な結果が返されたときに生成する深刻度を指定します。
9. **OK** をクリックします。 関数がテスト スクリプトに追加されます。

コード例

次のサンプル コード (太字部分) は、境界内のデータ検証用に自動生成されたものです。

```
transaction TMain
begin
  ...
  WebVerifyDataBound("ahref=¥"", "¥"", "main.asp?
from=welcome",
    WEB_FLAG_IGNORE_WHITE_SPACE |
WEB_FLAG_CASE_SENSITIVE, 1,
    SEVERITY_ERROR, bVerifyDataBoundSuccess1);
  WebPageUrl("http://myHost/shopit/", "ShopIt - Greetings");
  ...
```

HTML データ ダイジェスト検証関数を生成する

WebVerifyDataDigest 関数は、Web ページ ダイジェストを計算し、その後、コンテンツを計算済みのダイジェストに照らして検証します。 HTML や XML のソース コードの場合、ダイジェストは、ページのレスポンス ボディに含まれる特定の文字に関して計算した、文字の出現頻度のテーブルから構成されます。

1. メニュー ツリーで、検証したい Web ページを選択します。
2. **ソース** ビューを開きます。
3. 以下の手順の 1 つを実行します。
   - **スクリプト** > **ページ ダイジェストの検証** を選択します。
   - **検証の追加** をクリックし、**ワークフロー - 検証の追加** ダイアログ ボックスで **ダイジェストの検証** をクリックします。

   ダイジェストの名前が **定数名** テキスト ボックスに表示されます。
4. 省略可能: **定数名** テキスト ボックスを編集します。
5. **検証対象** グループ ボックスで、検証を適用する対象を指定します。
   - **すべての文字**
   - **印刷可能な文字**
   - **英数字**
6. **深刻度** グループ ボックスで、検証から否定的な結果が返されたときに生成する深刻度を指定します。
7. **OK** をクリックします。 関数がテスト スクリプトに追加されます。

コード例

次のサンプル コード (太字部分) は、データ ダイジェスト検証用に自動生成されたものです。

```
const
   DATA_DIGEST_1        :=
"¥h019200000000E7FBEFFFAEF7FD2AFEFFFF2B00000000000000000000000000
000000004
F06040072000800020038000600060002"
"¥h0006000A0035004B0024000E00050001000100010001000200010002000
F0059005A0
05800010003000700030002000300010 0";
"¥h12000100010006000200030001001500130002000100010003000100030003
0300A1002
30047003E00BA002100710056009B0001"
"¥h000D0062003C00620079003E000300AE006D00BC0023000E002C0005000
E000200020
0";

transaction  TMain
begin
   ...
   WebVerifyDataDigest(DATA_DIGEST_1, 188, 0, 0, 1,
      SEVERITY_ERROR);
   WebPageUrl("http://lab3/ShopItV60/default.asp", "ShopIt - Greetings");
   ...
```

ユーザー データのカスタマイズ

このセクションでは、ユーザーにより入力されたデータをどのようにパラメータ化すれば現実世界におけるユーザーの活動を負荷テスト時により適切にエミュレートできるかを説明します。

ユーザー入力データ

ビジュアルなユーザー入力データのカスタマイズ機能を使用すると、記録されたユーザー入力データをパラメータ化されたデータに置き換えることでテスト スクリプトをより現実的なものにすることができます。

ユーザー入力データのカスタマイズを行わないと、シミュレートするトランザクションはすべて同一のものになり、現実の環境でよく見られる変数を再現できません。

たとえば、**パラメータ ウィザード** を使用すると、テスト中にフォームに入力されるユーザー入力データをカスタマイズできます。 **パラメータ ウィザード** を使用すると、テスト中にフォーム フィールドに入力される値を指定できます。 そして、記録されたユーザー入力データをランダムなパラメータ化されたユーザー データに置き換えることによって、テスト スクリプトをより現実的なものにすることができます。

ユーザー データのカスタマイズの用途は以下のとおりです。

- 特定のトランザクションを実行するたびに異なる値をサーバーに渡して機能テストを行う
- 指定された確率分布に基づいて送信値を選択することにより、ユーザーの現実の行動をシミュレートすること

TrueLog Explorer ワークフロー

ユーザー データのカスタマイズを最も良く行えるのは以下のいずれかです。

- テスト スクリプトからセッション情報が構文解析された後
- サーバーからの応答が検証される前

ユーザー データのカスタマイズのシナリオ

ユーザー データのカスタマイズが一般に行われる場合のシナリオは 5 つあります。

- HTML フォームのポスト (WebPageSubmit を用いる)
- XML データのポスト (WebUrlPostBin または WebCustomRequestBin を用いる)
- データベース アプリケーション用データのポスト (入力データを Oracle OCI、DB2 CLI、ODBC などのデータベース アプリケーションの SQL コマンドやストアド プロシージャにバインドする (OraSet*、Ora8Set*、OdbcSet*、 を用いる) ことによる)
- Oracle フォーム制御値の変更 (ユーザー入力) のポスト (OraForms*Set 関数を用いる)
- Citrix ユーザー入力アクション (たとえば、マウス イベント、キーボード入力など) のポスト (CitrixKey* 関数や CitrixMouse* 関数を用いる)

注: ユーザー データのカスタマイズは、ターミナル エミュレーション アプリケーションではサポートされていませんが、入力パラメータはターミナル エミュレーションのテスト スクリプトで手動で編集することができます。

HTML の場合、パラメータ化はフォームの宣言部で行われます。 パラメータ化すると、テスト スクリプトの dclform セクションで宣言されているようなハードコードされたフォーム フィールド値は、ランダム変数に置き換わります。

XML アプリケーションおよびデータベース アプリケーションの場合、パラメータ化は上記の関数呼び出しのパラメータで行われ、ハードコードされたパラメータ値はデータ駆動型変数またはランダム変数に置き換わります。

HTML ユーザー データをカスタマイズする

パラメータに基づいて HTML ユーザー データをカスタマイズする方法について説明します。

新しいパラメータを使用して HTML ユーザー データをカスタマイズする

読み進める前に、静的なセッション情報がすべてテスト スクリプトから削除されていることと、最も最近のスクリプトの試行で TrueLog Explorer に表示される TrueLog が生成されていることを確認してください。

HTML ベースのアプリケーションでは、ユーザー データのカスタマイズの目標は、フォーム フィールドに送信される値をカスタマイズすることです。

このタスクでは、ランダム変数に基づいてパラメータを作成する手順を説明します。

1. **ファイル** > **TrueLog の追加** を選択して、TrueLog を TrueLog Explorer にロードします。

   TrueLog は、静的なセッション情報がすべて削除されたスクリプトの試行に基づいている必要があります。
2. ワークフロー バーの **ユーザー データのカスタマイズ** をクリックします。 **ワークフロー - ユーザー データのカスタマイズ** ダイアログ ボックスが開きます。
3. **HTML フォームにおけるユーザー入力データのカスタマイズ** リンクをクリックします。 これにより、TrueLog Explorer で以下のアクションが行われます。

   - メニュー ツリーで、最初の **WebPageSubmit** API 呼び出しノードを選択します。
   - **TrueLog のステップ スルー** ダイアログ ボックスを開きます (**フォーム送信** オプション ボタンが選択されています)。
   - **ポスト データ** ビューが表示されます。

     ポスト データ ビューには、選択されている WebPageSubmit 呼び出しによって送信された HTML フォームが含まれているページが表示されます。 送信されたフォームのコントロールが強調表示され、コントロールが緑色の枠線で囲まれている場合は、送信値が初期値と同じである (しかも、記録中にユーザーによって変更されていない) ことを示します。 赤色の枠線の場合は、送信値が初期値と異なる (しかも、記録中に変更されている) ことを示します。 カーソルをフォーム コントロールの上に置くと、ツールチップに、そのコントロールの名前、初期値、および送信値が表示されます。オレンジ色の線は、下にある **フォーム データ** ビュー内の対応する BDL フォーム フィールド宣言を示します。
4. **TrueLog のステップ スルー** ダイアログ ボックスの **次を検索** または **前を検索** をクリックすると、TrueLog 内のすべての WebPageSubmit 呼び出しに目を通すことができます (これらは、ユーザー データのカスタマイズの候補となる呼び出しです)。

   注: ポスト データ ビューで強調表示されている HTML コントロールは、カスタマイズ可能なフォーム フィールドを表しています。
5. **ポスト データ** ページで、カスタマイズしたいフォーム コントロールを右クリックし、**値のカスタマイズ** を選択します。

   記録されている値を、さまざまな種類の入力データ（ファイルにあらかじめ定義した値や一般的なランダム値を含む）に置き換えることができます。また、記録された入力データをカスタマイズした値に置き換えるためのコードをテスト スクリプトに生成することもできます。

   パラメータ ウィザード が開きます。

   パラメータ ウィザード では、次の 2 つの方法でスクリプトの値を変更できます。

   - スクリプトの dclparam または dclrand のセクションで定義されている既存のパラメータを使用する
   - 新しい定数値、ランダム値、複数列データ ファイル内の値に基づいた、新しいパラメータを作成する

新しいパラメータを作成すると、そのパラメータは既存パラメータ群に追加され、以降のカスタマイズで利用できるようになります。

6. **パラメータを新規に作成する** オプション ボタンをクリックし、それから **次へ** をクリックして新しいパラメータを作成します。 **パラメータの新規作成** ページが開きます。
7. **ランダム変数からパラメータを作成** オプション ボタンをクリックし、それから **次へ** をクリックします。 **ランダム変数** ページが開きます。
8. テスト スクリプトに挿入したいランダム変数の種類をリストボックスから選択し、**次へ** をクリックします。

   選択した変数の種類の簡単な説明が下部のウィンドウに表示されます。

   変数の名前と属性の指定 ページが開きます。
9. 変数の名前を **名前** テキスト ボックスに入力します。
10. 省略可能： **ファイル** グループ ボックスの **新規作成** をクリックして、新しいランダム変数ファイルを作成します。
11. 値を呼び出す順序として **ランダム** または **シーケンシャル** を指定します。

    ファイルからの文字列 というランダム変数タイプでは、指定されたファイルからランダムに選択または順次選択できるデータ文字列が生成されます。
12. **ファイル** グループ ボックスの **名前** リスト ボックスから構成済みのデータ ソースを選択して、**次へ** をクリックします。 **使用法の選択** ページが表示されます。
13. 以下の選択肢のいずれかを選択して、新しいランダム変数の使用法を指定します。

    - **毎回**
    - **トランザクションごと**
    - **テストごと**
14. **終了** をクリックします。 これで、テスト スクリプトは、そのフォーム フィールドについて、記録された値ではなくランダム変数を使用するようになります。 新しいランダム変数関数が **BDL** ページに追加されます。

テスト スクリプトにランダム変数関数を追加した状態でスクリプトの試行を起動し、スクリプトがエラーなしに動作することを確かめます。

既存のパラメータを使用して HTML ユーザー データをカスタマイズする

1. **ファイル** > **TrueLog の追加** を選択して、TrueLog を TrueLog Explorer にロードします。

   TrueLog は、静的なセッション情報がすべて削除されたスクリプトの試行に基づいている必要があります。
2. ワークフロー バーで、**ユーザー データのカスタマイズ** をクリックします。 **ワークフロー - ユーザー データのカスタマイズ** ダイアログ　ボックスが開きます。
3. **HTML フォームにおけるユーザー入力データのカスタマイズ** リンクをクリックします。
4. **TrueLog のステップ スルー** ダイアログで **次を検索** または **前を検索** をクリックして、TrueLog 内のすべての WebPageSubmit 呼び出しを参照します。

   注: これらは、ユーザー データのカスタマイズの候補となる呼び出しです。

   ポスト データ ビューには、選択されている WebPageSubmit 呼び出しによって送信された HTML フォームが含まれているページが表示されます。 カーソルをフォーム コントロールの上に置くと、ツール チップに、そのコントロールの名前、初期値、および送信された値が表示されます。

   注: ポスト データ ビューで強調表示されている HTML コントロールは、カスタマイズ可能なフォーム フィールドを表しています。
5. **ポスト データ** ページで、カスタマイズしたいフォーム コントロールを右クリックし、**値のカスタマイズ** を選択します。

記録されている値を、さまざまな種類の入力データ（ファイルにあらかじめ定義した値や一般的なランダム値を含む）に置き換えることができます。また、記録された入力データをカスタマイズした値に置き換えるためのコードをテスト スクリプトに生成することもできます。

パラメータ ウィザード が開きます。

パラメータ ウィザード では、次の 2 つの方法でスクリプトの値を変更できます。

- スクリプトの dclparam または dclrand のセクションで定義されている既存のパラメータを使用する
- 新しい定数値、ランダム値、複数列データ ファイル内の値に基づいた、新しいパラメータを作成する

新しいパラメータを作成すると、そのパラメータは既存パラメータ群に追加され、以降のカスタマイズで利用できるようになります。

6. **既存のパラメータを使用する** を選択します。
   これにより、テスト スクリプトの dclparam セクションまたは dclrand セクションに定義されている既存のパラメータを指定できます。
7. リスト ボックスから、使用するデータ型を選択します。
   このリストには、テスト スクリプトの dclparam セクションと dclrand セクションに定義されているすべての変数が表示されます。
8. **終了** をクリックします。 テスト スクリプトが新しいパラメータによって変更されます。
9. カスタマイズ対象の次のフォーム フィールドを見つけるには、**TrueLog のステップ スルー** ダイアログの **次を検索** をクリックします。

[フォーム データ] ビュー

ポスト データ ビューおよび **フォーム データ** ビューのどちらからでも、入力データを変更する関数呼び出しごとに入力データをカスタマイズできます。 **フォーム データ** ビューは、カスタマイズ可能なすべてのフォーム フィールドの表示とカスタマイズを行う便利な手段となります。 **フォーム データ** ビューは、BDL スクリプトのフォーム セクションとそれに含まれているすべての名前/値ペアを表したものです。

(**ポスト データ** ビューと同様に) コントロールを右クリックして **値のカスタマイズ** を選択し、**パラメータ ウィザード** を開きます。

ユーザー データが既にカスタマイズされているコントロールは、緑色の輪郭で表示されます。

複数列データ ファイル

複数列データ ファイルを使ったパラメータ化は、データをパラメータ化する有効な方法です。 特定の文字列値の組み合わせを保存したファイルを定義します。 データ ファイルの各列はそれぞれのパラメータに対応します。 複数列データ ファイルを使うと、データ駆動型テスト モデルが利用できるようになり、すべてのユーザー データ入力をただ 1 つのデータ ファイルでカバーできます。

データベース アプリケーションでの作業

ここでは、TrueLog Explorer のスクリプト カスタマイズ機能を共通データベース API に依存するアプリケーションのテストに適用する方法について説明します。

データベース アプリケーションでの作業 - 概要

Silk Performer は、データベースの拡張サポートを提供します。これには、Microsoft ODBC や、Oracle データベース (バージョン 8 および 9) にアクセスするための標準インターフェイスである Oracle Call Interface 8 (OCI8) に対する拡張サポートが含まれます。

Silk Performer のデータベース サポートには、次の API が含まれます。

- Oracle OCI 7.0
- Oracle OCI 8.0
- Microsoft ODBC
- IBM DB2/CLI

データベース アプリケーションに対して、TrueLog Explorer は、Web ベース アプリケーションに利用できるものと同じスクリプト カスタマイズ機能を提供します。

- ユーザー データのカスタマイズ
- 検証関数
- 入力と出力の相関関係のカスタマイズ(Web アプリケーションにおけるセッション処理のカスタマイズに相当する)

注: データベースに対する TrueLog On Error 分析は現在サポートされていません。

この章を読み進む前に、TrueLog Explorer の基本機能を熟知していることが不可欠です。

サンプル データベース アプリケーション - Customer OCI

サンプル データベース アプリケーションの Customer OCI は、顧客の連絡先情報の追加、編集、削除を行う簡単なアプリケーションをシミュレートするものです。

データベース TrueLog の構造

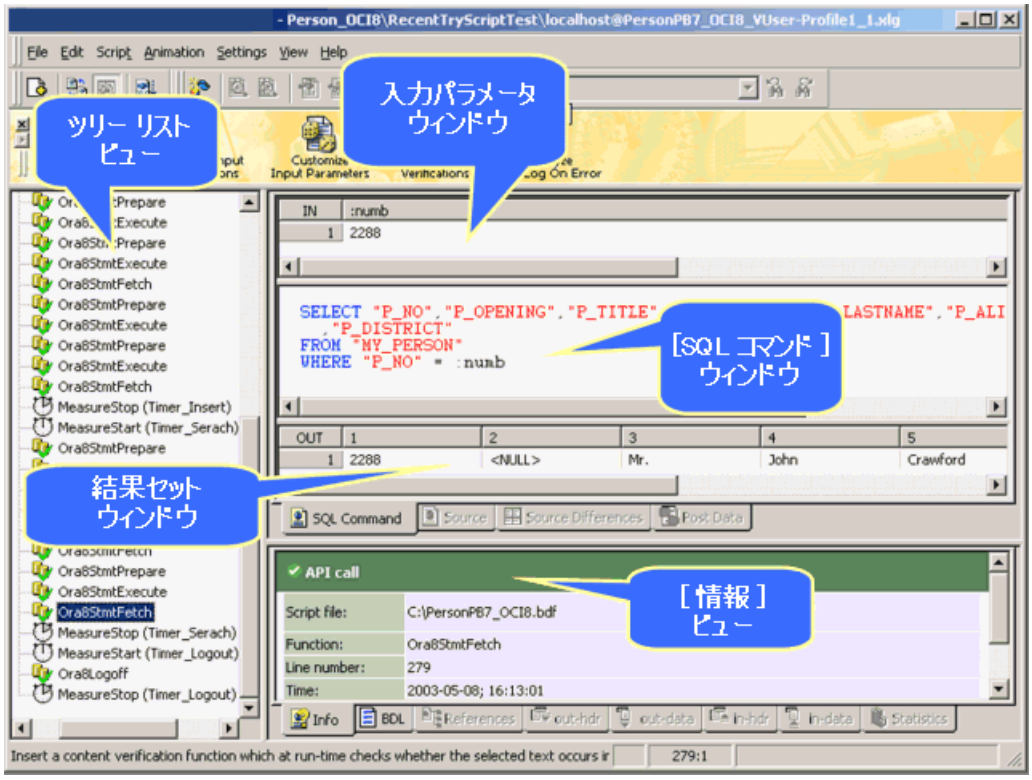
TrueLog Explorer は、Web アプリケーション用の HTTP クライアント リクエストと HTTP/HTML サーバー レスポンスの視覚化をサポートするのと同じ方法で、データベースのリクエストとレスポンスの視覚化をサポートします。 ただし、データベース TrueLog の場合は、HTML TrueLog の場合の GUI の代わりに、読み取り可能なデータベース形式で表されます。

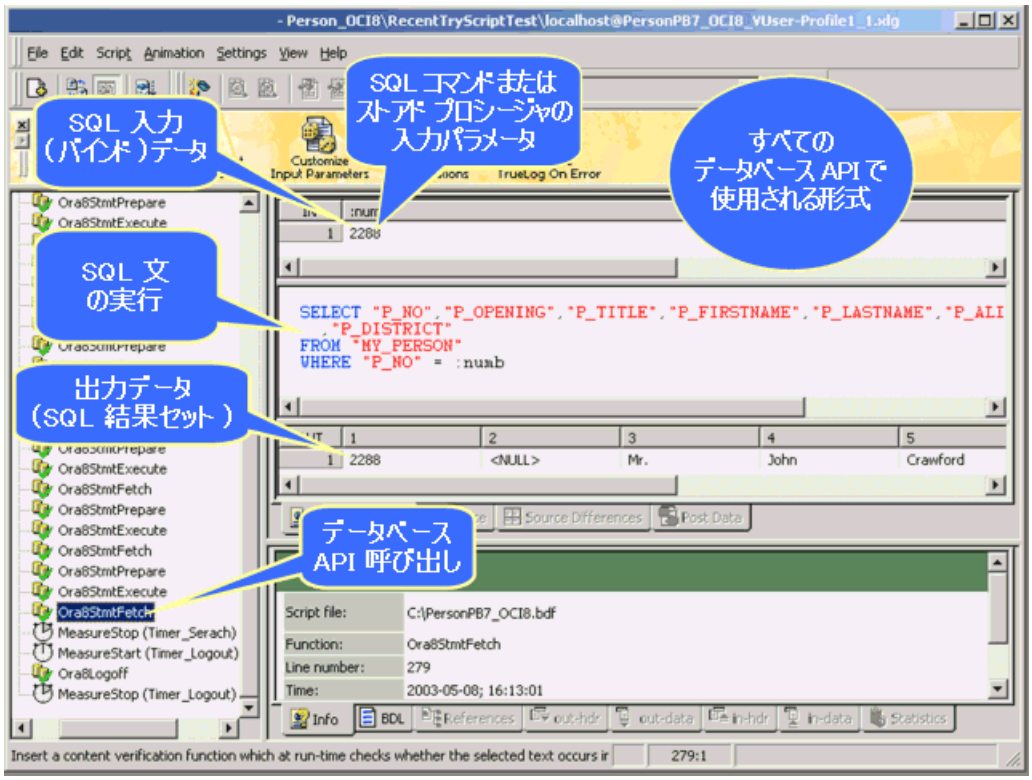
TrueLog Explorer は データベース ブラウザとして機能し、SQL 文およびそれに対応する入力パラメータと出力パラメータを同時に見ることができます。 サポートされる各データベース API は、同じ TrueLog 形式でデータを表します。

データベース TrueLog で提供される 5 つのウィンドウは次のとおりです。

- メニュー ツリー - 関連するすべての API 呼び出し(データを送信または取得する呼び出し)が一覧表示されます。
- **入力パラメータ** - データベースに送信された SQL 入力(バインド)データです。
- **SQL コマンド** - 対応する SQL コマンドです。
- **結果セット** - クライアントが受信した出力 SQL データです。
- **情報** - テスト実行に関するデータです。

データベース TrueLog に対してアクティブになり適用されるビュー タブは、**SQL コマンド**、**情報**、**BDL** のみです。


ツリー リスト
ビュー
入力パラメータ
ウィンドウ
[SQL コマンド]
ウィンドウ
結果セット
ウィンドウ
[情報]
ビュー


SQL 入力
(バインド)データ
SQL コマンドまたは
ストアド プロシージャの
入力パラメータ
すべての
データベース API で
使用される形式
SQL 文
の実行
出力データ
(SQL 結果セット)
データベース
API 呼び出し

相関関係

このセクションでは、データベースの入力パラメータと出力パラメータの相関関係を識別する方法を説明します。

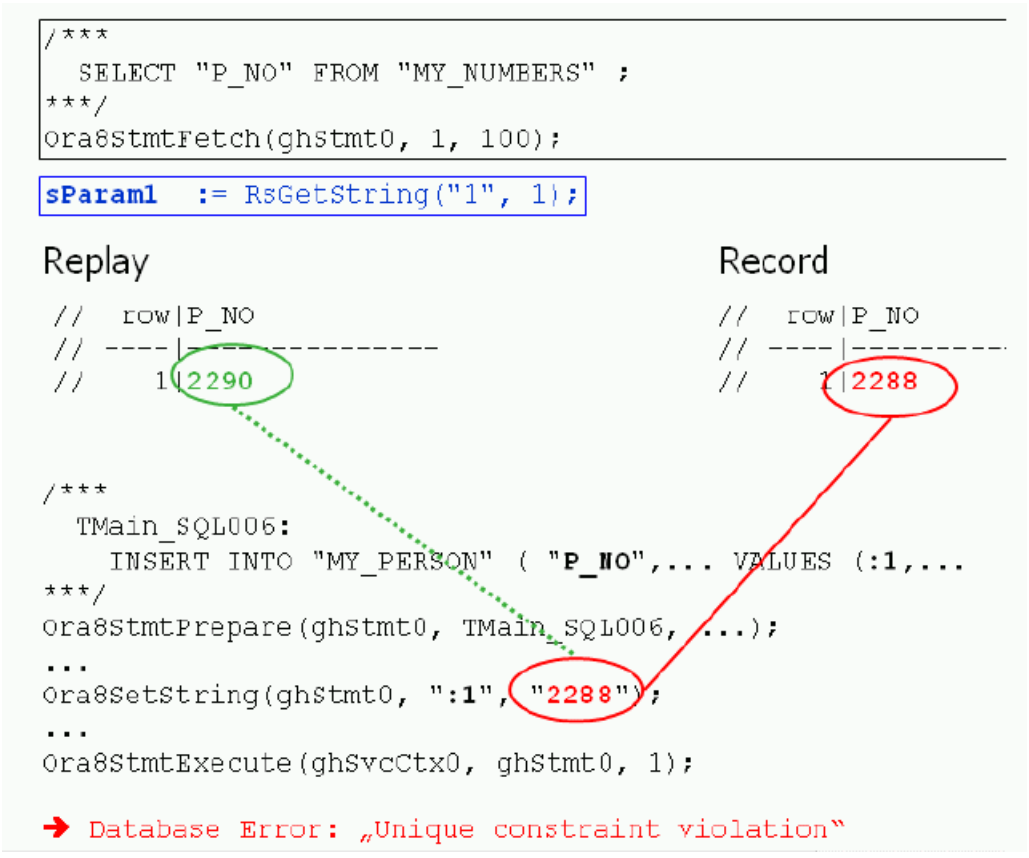
入力と出力の相関関係

TrueLog Explorer の入力と出力の相関関係機能を利用すると、記録 API 呼び出しと再生 API 呼び出しを並べてステップ スルーできます。 また、データベースの入力パラメータと出力パラメータの相関関係を明らかにすることもできます。 Web アプリケーションでのセッション処理のカスタマイズと同様に、入力と出力の相関関係機能は、再生テスト実行と記録セッションを比較して、セッション関連の値を示す可能性のある差分を特定します。 その後、この差分値は、テスト実行スクリプト内の出力値と入力値の相関関係を検索するための基礎として使用されます。 最終的に、TrueLog Explorer を利用すると、テスト スクリプト内で特定された動的な値を変更できます。

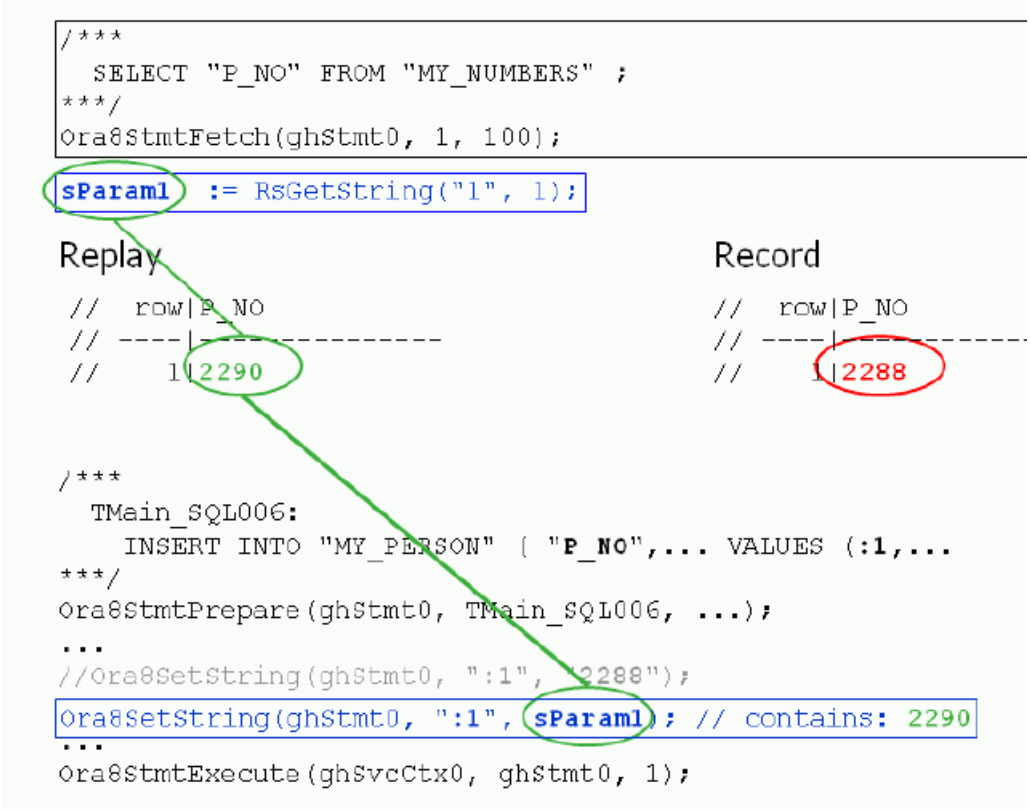
入力と出力の相関関係機能は、TrueLog Explorer でデータベース TrueLog を開いているときのワークフロー バーから利用できます。

データベース エラーの種類は、テスト対象のデータベースの種類に固有です。 一意制約エラーは、通常、同じデータが繰り返しデータベースに送信されると発生します。 このようなエラーは、一般に、ユーザー ID や注文番号のようなデータベース テーブルの主キーとして使用される一意の値が送信された場合に発生します。 このような個人情報が複数回送信されると、エラーが発生します。 この種の重複は、テストスクリプト内で識別し、TrueLog Explorer を使用してカスタマイズできるセッション情報です。

次の 2 つの BDL コード例は、入力と出力の相関関係を示しています。 実行呼び出しの入力パラメータは [2288] です。 この値は、前の呼び出しの出力パラメータ テーブルに出現しているので、相関関係が明かになっています。



出力パラメータの値(2290)は、後のテスト実行で入力パラメータの値を設定する際に使用できるよう、解析された変数(sParam1)に格納されます。



セッション データを変数に置き換える

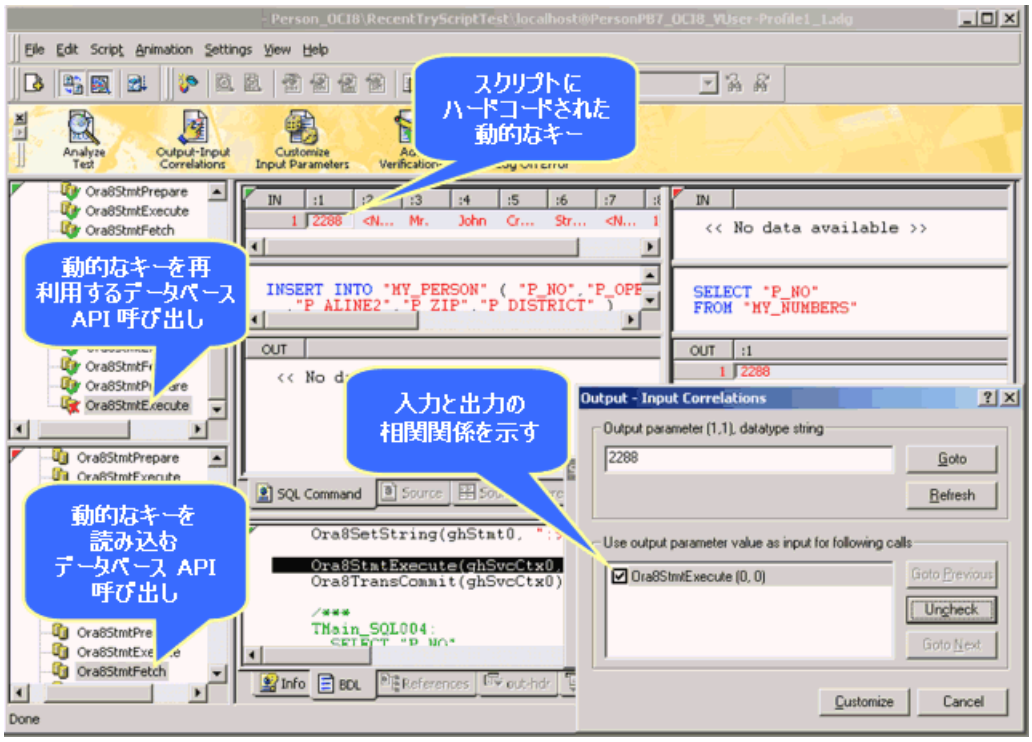
1. **ファイル** > **TrueLog の追加** を選択します。 **ファイルを開く** ダイアログ ボックスが開きます。
2. 再生データベース TrueLog を選択し、**開く** をクリックします。 再生 TrueLog が TrueLog Explorer で開きます。
3. ワークフロー バーの **入力と出力の相関関係** をクリックします。 **ワークフロー - 入力と出力の相関関係** ダイアログ ボックスが表示されます。
4. **差分の検索** をクリックします。 関連する記録 TrueLog を自動的に開くかどうかを尋ねるメッセージ ボックスが表示されます。
5. **OK** をクリックします。 比較モードが有効になります。 記録 TrueLog が開き、**相関関係のステップ スルー** ダイアログ ボックスが表示されます。
6. デフォルトの選択(**異なる結果セット** および **記録 TrueLog の値**)を使用し、**次を検索** をクリックして、記録されている結果セット データと再生された結果セット データの間の最初の差分に進みます。

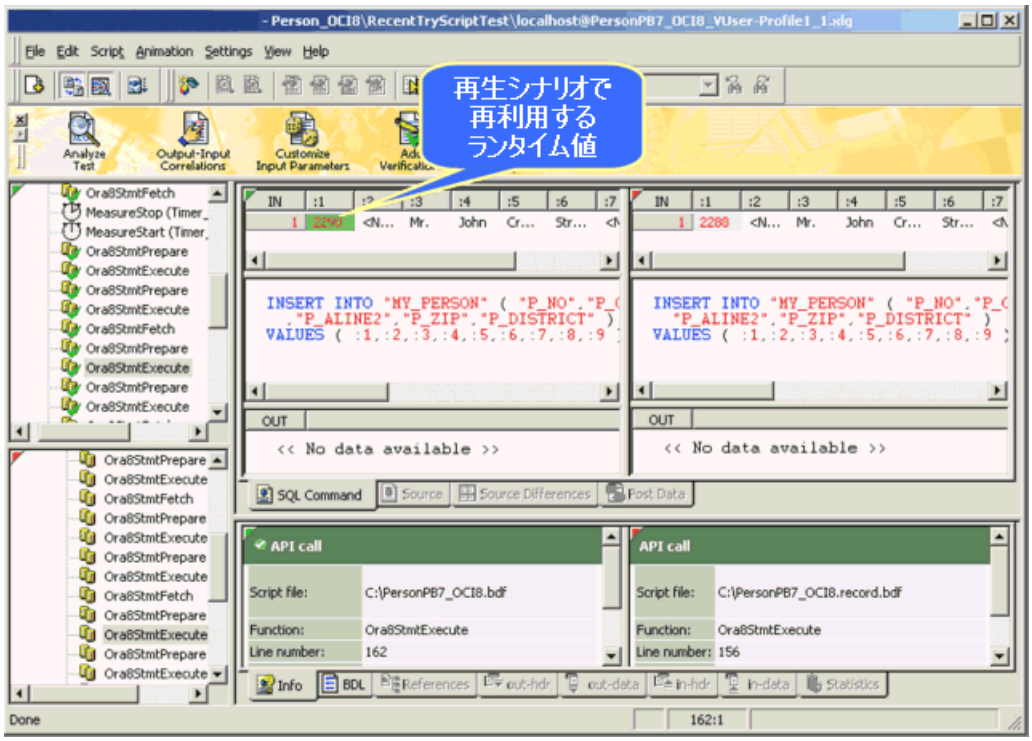
   差分はデータベース サーバーによって返される動的なデータであり、相関関係の候補です。 動的なデータがさらに SQL コマンドまたはストアド プロシージャの入力パラメータとして使用される場合、そのデータは出力データと入力データの相関関係を構成します。
7. **相関関係** をクリックします。 **入力と出力の相関関係** ダイアログ ボックスが開きます。

   動的な値が別の SQL コマンドまたはストアド プロシージャの入力値として出現している場合、そのステートメント/プロシージャ名が **出力パラメータ値を次の呼び出しの入力として使用する** ウィンドウに表示されます。 TrueLog Explorer は、後でこのような呼び出しの入力パラメータ値を出力パラメータ値で置き換えるための変数を作成できます。

   注: 出力パラメータ値を次の呼び出しの入力として使用する ウィンドウに <<相関する値が見つかりません>> と表示される場合は、相関関係は検出されなかったということです。 **キャンセル** をクリックして、**相関関係のステップ スルー** ダイアログ ボックスに戻ります。

8. **チェック ボックス** をクリックして、特定されたステートメントの名前を選択します(複数存在する場合もあります)。
9. **移動** をクリックして、テスト スクリプトのコンテキスト内で入力値を確認します。
10. 入力値をカスタマイズする必要がある場合は、**カスタマイズ** をクリックし、テストスクリプト内の入力パラメータを変数に置き換えます。





スクリプトの試行を起動して、エラーなしでスクリプトが実行されることを確認します。

手動相関

手動相関は、よくわかっているアプリケーションに対してのみ行うことをお勧めします。 手動相関を実行するには、**SQL コマンド** ビューで再生パラメータを右クリックし、**相関関係の検索** を選択します。

手動相関は、**受信データ** ビュー(入力パラメータ)、**送信データ** ビュー(結果セット)、および **SQL コマンド** ビューでのみ使用できます。 **受信データ** ビューで使用すると、TrueLog Explorer は SQL コマンドの前にある相関する値(**送信データ** ビュー内)を検索します。 **送信データ** ビューで使用すると、TrueLog Explorer は SQL コマンドの後にある相関する値(**受信データ** ビュー内)を検索します。 **SQL コマンド** ビューで使用すると、パラメータが SQL コマンド内にハードコーディングされている場合に、TrueLog Explorer は SQL コマンドの前にある相関する値(**送信データ** ビュー内)を検索します。

例：

```
INSERT INTO MY_PERSON (P_NO, P_FIRSTNAME, P_LASTNAME)
VALUES ( 2289, "Michael", "Smith" )
```

データベース解析関数

データベース TrueLog に対して、TrueLog Explorer は単一の解析関数(Parse Element Value)を提供します。 この解析関数は、SQL ウィンドウの **出力データ** ビューのコンテキスト メニューから利用できます。

この関数は、結果セットの 1 つの要素に対する解析文を生成します。 例：

```
Ora8StmtExecute(ghSvcCtx0, ghStmt0);
Ora8StmtFetch(ghStmt0, ORA_FETCH_ALL, 100); // rows fetched: 1
sParam2 := RsGetString("1", 4);
Print("sParam2: " + sParam2);
```

データ型(number、float、または string)に応じて、適切な解析関数が生成されます(RsGetInt、RsGetFloat、または RsGetString)。

入力パラメータのカスタマイズ

効果的な負荷テストのためには、データベース テストを実行するたびに値が送信されるように、パラメータ化された入力データを SQL 文に割り当てることをお勧めします。 そうしないと、テストを実行するたびに、記録されている同じ入力パラメータ値が送信されます。これでは、実際のユーザー アクティビティの実践的なシミュレーションになりません。 TrueLog Explorer を利用すると、記録された入力パラメータ値をカスタマイズされた入力パラメータに置き換えるコードを、テスト スクリプト内にビジュアルに生成できます。

注: Web アプリケーション用のユーザー データのカスタマイズと同じメソッドが使われます。

サンプル データベース アプリケーションである Customer OCI を使用すると、ユーザー入力データをカスタマイズを実験できます。 記録された簡単なビジネス トランザクションを再生するスクリプトをカスタマイズして、元々固定データ フィールドだった所に、ランダムなデータを入力する変数を使用できます。 入力データ値としては、固定値、ランダム値、または構成済みのデータ ファイル(CSV ファイルなど)から取得した値を使用できます。

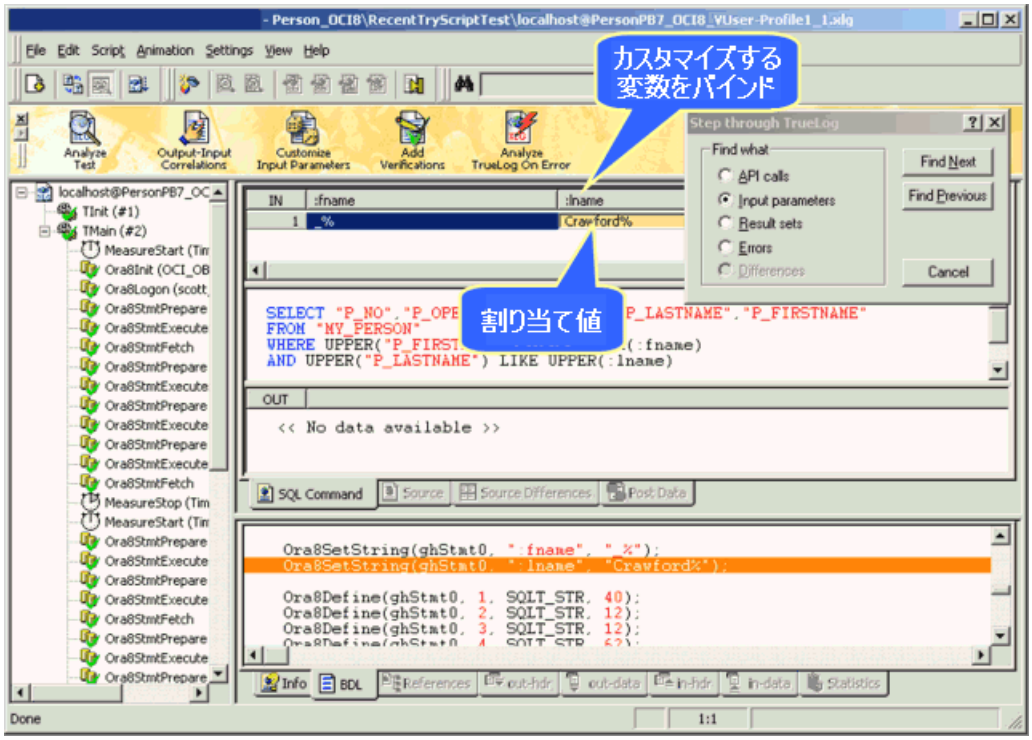
複数列データ ファイルに基づく入力パラメータを作成する

複数列データ ファイルを使ったパラメータ化は、データをパラメータ化する有効な方法です。 特定の文字列値の組み合わせを保存したファイルを定義します。 データ ファイルの各列はそれぞれのパラメータに

対応します。 複数列データ ファイルを使うと、データ駆動型テスト モデルが利用できるようになり、すべてのユーザー データ入力をただ 1 つのデータ ファイルでカバーできます。

1. **ファイル** > **TrueLog の追加** を選択します。 **ファイルを開く** ダイアログ ボックスが開きます。
2. 再生データベース TrueLog を選択し、**開く** をクリックします。 再生 TrueLog が TrueLog Explorer で開きます。
3. ワークフロー バーの **入力パラメータのカスタマイズ** をクリックします。 **ワークフロー - 入力パラメータのカスタマイズ** ダイアログ ボックスが開きます。
4. **入力パラメータのカスタマイズ** をクリックします。 テスト スクリプトで入力パラメータを含む最初の API ノードが選択されて、**TrueLog のステップ スルー** ダイアログが表示されます(その際、**入力パラメータ** オプション ボタンが選択されています)。

   注: 選択されているノードの入力パラメータをカスタマイズしない場合は、**次を検索** をクリックし、入力パラメータを含む次のノードに進みます。



5. **入力パラメータ** ビューまたは **SQL コマンド** ビューで、入力パラメータを右クリックし、**値のカスタマイズ** を選択します。

   スクリプトで値を変更するには 2 つの方法があります。 BDL スクリプトの dclparam セクションまたは dclrand セクションであらかじめ定義されている既存のパラメータを使用できます。 新しいパラメータは、定数値、ランダム変数、または複数列データ ファイルのデータに基づく変数から作成できます。

   パラメータ ウィザード - 値のカスタマイズ ダイアログ ボックスが開きます。
6. **複数列データ ファイルからパラメータを作成** オプション ボタンをクリックし、**次へ** をクリックします。 **パラメータ ウィザード - 複数列データ ファイルからのパラメータ作成** 画面が開きます。
7. 以下の手順の 1 つを実行します。

   - **ファイル名** リスト ボックスから構成済みの複数列データ ファイルを選択します。
   - 新しい複数列 CSV(コンマ区切り値)ファイルを作成するには、**新規作成** ボタンをクリックします。 新規ファイルの名前を入力して **OK** をクリックします。
8. 列名の編集、行の追加/削除、または列全体の追加/削除を行うには、列を右クリックします。

注: 各列は(1 個の入力値に割り当て可能な)個別のパラメータを表します。

9. 必要に応じて、データ フィールドにデータを入力します。
   または、列ヘッダー名を直接選択して、データ フィールドをカスタマイズすることもできます。
10. 必要に応じて **ハンドル名** や **パラメータ名** をカスタマイズします。
11. **次へ** ボタンをクリックしてファイルを保存します。 **パラメータの挿入属性の選択** ページが開きます。
12. **行選択の順序** グループ ボックスで、複数列データ ファイルから値を代入するときの順序を選択します。

    - **ランダム** - FileGetRndRow 関数を使用すると、パラメータがデータ ファイルのランダムな行から割り当てられます。
    - **シーケンシャル (マシン全体)** - FileCSVLoadGlobal 関数と FileGetNextRow 関数を使用すると、マシン全体で、パラメータがデータ ファイルの次のシーケンシャル行から割り当てられます。 マシン上のすべての仮想ユーザーは、現在の行で同じグローバル行ポインタを使用するようになります。
    - **シーケンシャル (テスト全体)** - FileCSVLoadGlobal 関数と FileGetNextUniqueRow 関数を使用すると、テスト全体で、パラメータがデータ ファイルの次のシーケンシャル行から割り当てられます。 すべてのマシン上のすべての仮想ユーザーは、現在の行で同じグローバル行ポインタを使用するようになります。
13. **属性** グループ ボックスで、新しい挿入をどのくらいの頻度で行うかを指定します。
    たとえば、新規ユーザー名の挿入をトランザクションのたびに行うか、テストにつき 1 度だけ行うかを指定します。

    - **トランザクションごと**
    - **テストごと**

    使用可能なすべてのユーザー グループとトランザクションは、**トランザクションごと** ドロップダウン リストから選択できます。
14. **終了** をクリックします。 TrueLog Explorer によってテスト スクリプトが変更され、複数列データ ファイルが保存されます。

カスタマイズされた入力パラメータは、緑で表示されます。

テスト スクリプトにランダム変数関数を追加した状態でスクリプトの試行を起動し、スクリプトがエラーなしに動作することを確かめます。

結果セット データの検証

TrueLog Explorer では、SQL クエリなどのデータベース操作の結果セットに検証を適用できます。 検証を利用すると、テストの実行中に、予想されるテスト結果が返らない場合はエラーを発生させることによって、テスト結果を拡張できます。 出力データを選択して簡単なコマンドを適用することで、必要な検証関数がすべて生成されて、BDL スクリプトに自動的に挿入されます。

TrueLog Explorer には、2 つのデータベース操作検証オプションが用意されています。

- 要素の値の検証
- 結果セットの行数の検証

データベース操作の要素の値を検証する

1. **ファイル** > **TrueLog の追加** を選択します。 **ファイルを開く** ダイアログ ボックスが開きます。
2. 再生データベース TrueLog を選択し、**開く** をクリックします。 再生 TrueLog が TrueLog Explorer で開きます。
3. **編集** > **TrueLog のステップ スルー** を選択します。
   代替方法:ツールバーの **TrueLog のステップ スルー** をクリックします。
   TrueLog のステップ スルー ダイアログ ボックスが開きます。

4. **結果セット** オプション ボタンをクリックします。
5. **次を検索** をクリックして、結果セットを返す最初のデータベース コマンドに進みます。
   たとえば、fetch などの Ora8StmtFetch コマンドを見つけることができます。
   セット内の最初のデータ要素が選択されます。
6. 以下の手順の 1 つを実行します。

   - 検証するデータ要素を右クリックし、**要素値の検証** を選択します。
   - ワークフローバーの **検証の追加** をクリックし、**要素値の検証** をクリックします。

   指定したデータ要素およびその行または列の仕様が **値** のテキスト ボックスに表示されます(その際、**値** オプション ボタンが選択されます)。
7. 省略可能: 既存のパラメータまたは新規のパラメータを検証するには、**パラメータ値** をクリックします。
   a) 参照 (**...**) ボタンをクリックし、パラメータを選択します。
      パラメータが存在しない場合は、**パラメータ ウィザード** が開き、新規パラメータを作成できます。
   b) **OK** をクリックします。
8. **深刻度** グループ ボックスで、検証の結果、否定的な結果が返った場合の深刻度を指定します。
9. 検証で大文字小文字を区別したり、それをスクリプト全体のルールとして適用するには、適切なチェックボックスをオンにします。 結果の変数名が **結果の変数名** テキスト ボックスに表示されます。
10. **OK** をクリックします。 テスト スクリプトに関数が追加されます。

スクリプトの試行を起動して、エラーなしでスクリプトが実行されることを確認します。

データベースの結果セットの行数を検証する

結果セット行数の検証では、返されたデータベース結果セットに、指定した行数が含まれることを確認します。 返された結果セットの行数が、指定した行数と異なる場合は、エラーが発生します。

1. **ファイル** > **TrueLog の追加** を選択します。 **ファイルを開く** ダイアログ ボックスが開きます。
2. 再生データベース TrueLog を選択し、**開く** をクリックします。 再生 TrueLog が TrueLog Explorer で開きます。
3. **編集** > **TrueLog のステップ スルー** を選択します。
   代替方法:ツールバーの **TrueLog のステップ スルー** をクリックします。
   TrueLog のステップ スルー ダイアログ ボックスが開きます。
4. **結果セット** オプション ボタンをクリックします。
5. **次を検索** をクリックして、結果セットを返す最初のデータベース コマンドに進みます。
   たとえば、fetch などの Ora8StmtFetch コマンドを見つけることができます。
   セット内の最初のデータ要素が選択されます。
6. 以下の手順の 1 つを実行します。

   - 検証するデータ要素を右クリックし、**結果セット行数の検証** を選択します。
   - ワークフローバーの **検証の追加** をクリックし、**行数の検証** をクリックします。

   現在のデータベースの行数が **値** のテキスト ボックスに表示されます(その際、**値** オプション ボタンが選択されます)。
7. 省略可能: 既存のパラメータまたは新規のパラメータを検証するには、**パラメータ値** をクリックします。
   a) 参照 (**...**) ボタンをクリックし、パラメータを選択します。
      パラメータが存在しない場合は、**パラメータ ウィザード** が開き、新規パラメータを作成できます。
   b) **OK** をクリックします。
8. 行数が現在の行数と一致する程度を、[ちょうど]、[以上]、[以下] から選択します。
9. **深刻度** グループ ボックスで、検証の結果、否定的な結果が返った場合の深刻度を指定します。

10ΩK をクリックします。 テスト スクリプトに関数が追加されます。

スクリプトの試行を起動して、エラーなしでスクリプトが実行されることを確認します。

XML アプリケーションでの作業

このセクションでは、XML アプリケーションのテスト用に提供されているスクリプト カスタマイズ機能や TrueLog 分析機能について説明します。

XML アプリケーションでの作業 - 概要

TrueLog Explorer では、XML アプリケーションについても、HTML アプリケーションと同様に以下のサポートを行っています。

- セッション処理
- 入力データのカスタマイズ
- 検証関数
- TrueLog On Error の分析

TrueLog Explorer の Web サービス サポートの特徴に、XML TrueLog とのやり取りや分析を行いやすくするための、直感的な XML データ インターフェイスがあります。

TrueLog Explorer の XML API は、Web API と同様の機能を持っています。 クエリや検証のための XML データ アクセス関数は、値が複数回指定されている場合でも値の正確な場所を指定できるという点で、Web 関数より優れています。 Web API では、WebParseDataBound 関数を使って左右の境界を指定できるだけであり、これだけではサーバー データの変更に柔軟に対応しきれません。

この章を読み進む前に、TrueLog Explorer の基本機能を熟知していることが不可欠です。

サンプル Web サービス

本章で使用するサンプル スクリプトは、サンプル Web サービスの SOAP/XML メッセージが基になっています。 このスクリプトは、.NET Explorer で生成し、Silk Performer で使用できるようにエクポートしたものです。

.NET Explorer を使ってサンプル Web サービスを表示するには、次の URL にアクセスしてください。
http://demo.borland.com/BorlandSampleService/BorlandSampleService.asmx?WSDL

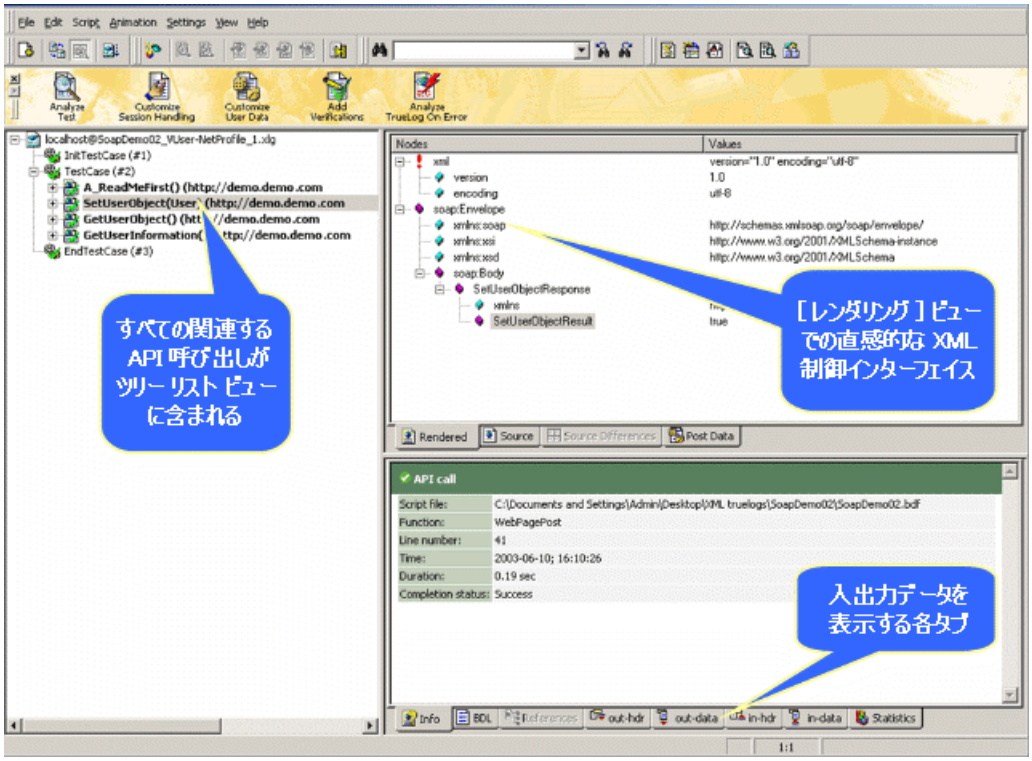
XML TrueLog 構造

XML TrueLog は、読みやすい XML メニュー ツリー形式を使って、XML の要素、属性、値を視覚化します。

XML TrueLog の UI の概要

XML TrueLog は、読みやすい XML メニュー ツリー形式を使って、XML の要素、属性、値を視覚化します。 この XML コントロールは、選択した API 呼び出しのレスポンス データを、**レンダリング** ページの直感的な XML インターフェイスに表示します。 **ソース** ページと **受信データ** ページでは、サーバーから返されたデータがそのまま表示されます。 **送信データ** では、サーバーへ送られたデータがそのまま表示されます。

XML TrueLog の 3 つの主要要素の詳細については、以下のトピックを参照してください。



メニュー ツリー

インターフェイスの左側にあるメニュー ツリーでは、ダブル クリックによって TrueLog API ノードを展開したり折りたたんだりすることができます。 ノードをクリックすると、そのコンテンツが [コンテンツ] ペインに、履歴の詳細が **情報** ページに表示されます。

TrueLog に含まれる Web ページの詳細を表示するには、メニュー ツリーでそのページの URL またはリンク記述を選択します。 すると、コンテンツが [コンテンツ] ペインに、プロパティが **情報** ページに表示されます。

[コンテンツ] ペイン

TrueLog Explorer は、受信したすべてのデータに対して複数のビューを提供します。 テスト対象のアプリケーションの種類 (Web、XML、データベース、TCP/IP、UDP) によって、**コンテンツ** ペインの表示オプションは変化します。

項目	説明
レンダリング (*HTML*、*XML*)	**レンダリング** ページには、メニュー ツリーで選択した API ノードに対するサーバーからのレスポンスが、視覚的にレンダリングされて表示されます。 HTML レスポンスは、**レンダリング** ページの背後で実行されている Internet Explorer コントロールによってレンダリングされます。 XML レスポンスは、XML データをメニュー ツリー形式で表示する XML コントロールによって表示されます。 HTML でも XML でもないレスポンスはレンダリングされないので、**ソース** ページまたは **受信データ** ページを使って表示する必要があります。 XML コントロール内で要素や属性を右クリックすると、メニュー ツリーを 1 レベル以上展開できます。
SQL コマンド	*XML* では利用できません。

項目	説明
ソース (*HTML*、*XML*、*TCP/IP*、*UDP* の場合のみ)	ソース データをそのまま表示します。 **ソース** ページには、Web コンテンツを生成するのに使われた HTML コードが表示されます。 ここには、**受信データ** ページと同じ情報が含まれます。 **ソース** ページと **送信データ** ページを使うことで、リクエストの受信データと送信データを同時に表示することができます。
ソース差分 (*HTTP* アプリケーション)	**ソース差分** ページには、選択したノード間の受信データ (ソース) の差分が一覧表示されます。 このページは、サーバー レスポンス内の動的情報を検索するのに役立ちます。セッション情報を探すには、まずここを検索します。 **差分モード** でのみ利用可能です。
ポスト データ (*HTML*、*XML*)	サーバーから返されたリクエスト データを表示します。ポストされた XML データは、メニュー ツリー形式の XML コントロールに表示されます。 要素は赤で、属性は青緑で表示されます。

情報ペイン

情報 ペインでは、BDL スクリプト、HTTP ヘッダー、タイミング統計など、テスト スクリプトおよびテスト実行に関するデータが表示されます。 テスト対象のアプリケーションの種類 (Web、XML、データベース、Oracle Forms、SAPGUI、Citrix、TCP/IP、UDP など) によって、**情報** ペインの表示内容は変化します。

項目	説明
情報	**情報** ページには、開いている TrueLog ファイルと選択した API ノードについて、以下のような一般情報が表示されます。 • スクリプト ファイル名 • 関数 • 行番号 • 時間 • 実行時間 • 絶対 URL • 完了ステータス • 追加情報 (可能な場合)
BDL	**BDL** ページには、開いている TrueLog に対応する BDL スクリプトが表示されます。 この BDL スクリプトは、選択されている API ノードの行まで自動的に移動して表示されます。
参照	*XML* では利用できません。
送信ヘッダー *(HTTP)*	**送信ヘッダー** ページには、アプリケーション (再生 TrueLog の場合は Silk Performer、記録 TrueLog の場合はブラウザ) がサーバーに送信する HTTP ヘッダーがそのまま表示されます。
受信ヘッダー *(HTTP)*	**受信ヘッダー** ページには、サーバーがアプリケーションに送信する HTTP ヘッダーがそのまま表示されます。
送信データ *(HTTP、TCP/IP、UDP)*	アプリケーションからサーバーへ HTTP-POST コマンドで送られたデータが表示されます。
受信データ *(HTTP、TCP/IP、UDP)*	**受信データ** ページには、アプリケーションがサーバーから受信したデータが表示されます。 このデータは、サーバー

項目	説明
統計 *(HTTP)*	から受信したそのままの形式で表示されます (HTML のレンダリングや XML のメニュー ツリー表示は行われません)。 **統計** ページには、Web ページのタイミング統計が表示されます。 ここには、Web ページ コンポーネントや通信要素ごとのタイミング情報が含まれます。 コンポーネントごとの内訳も含めて、正確なレスポンス時間がグラフで表示されます。 これにより、エラーやページ ダウンロードの遅延の根本原因を突き止めることができます。 **統計** ページには、ページ コンポーネントごとに次のデータが表示されます。 • DNS 検索時間：渡されたドメイン/ホスト名から IP アドレスを解決するのに要した時間 • 接続時間:擬似ユーザーがサーバーに接続するのに要した時間 • ネット ラウンドトリップ：クライアント リクエストの最初のバイトからサーバー レスポンスの最後のバイトまでの時間 (ドキュメントはすべて含むが埋め込みオブジェクトは含まない) • キャッシュ統計値

XML 解析関数

サーバーから返された XML コンテンツに対して、XML 解析関数を構造化された方法で適用することができます。 XML 解析関数は、XML ドキュメント内の特定の要素や属性の値を解析します。

注: XML 解析をお勧めできるのは、検証やカスタマイズを拡張する場合だけです。

セッション処理のカスタマイズにはお勧めしません。要素や属性には、XML ドキュメントの要素を指定するための標準言語である XML Path Language (XPath) を介してアクセスします。 XML 解析関数は、**レンダリング** ページで適用します。 Web API 型の解析は、**ソース** ページから XML に対して実行することができます。

XML 解析関数には次のものがあります。

- WebXmlParseNodeValue
- WebXmlParseNodeAttribute

注: 階層構造になった値は、XML コントロールを使って解析することができません。

XML アプリケーションのセッション処理のカスタマイズ

通常、セッション情報は XML コードに含まれないため、XML セッション処理のカスタマイズが必要になることはほとんどありません。

まれにセッション情報が XML コードに含まれる場合には、Web アプリケーションについて説明したセッション処理のカスタマイズの作業に進んでください。

ユーザー入力データのカスタマイズ

ユーザー入力データのカスタマイズの目的は、送信された XML データの要素や属性の値をカスタマイズすることです。 たとえば、記録された SOAP トラフィックのデータなどです。

HTML では、テスト スクリプトの DCLFORM セクションで宣言された、ハードコードされたフォーム フィールド値は、ランダム変数に置き換えられます。 この場合、form 宣言でパラメータ化が行われます。

XML では、対応する関数のポスト データ パラメータで宣言された、ハード コードされた要素や属性の値は、ランダム変数に置き換えられます。 パラメータ化は post data、WebUrlPostBin、WebCustomRequestBin の各呼び出しの WebPagePost パラメータで行われます (WebPagePost は .NET Explorer によって生成されます。 これは、HTTP-POST コマンド用に使用するページ レベルの API 呼び出しです)。

ユーザー入力データのカスタマイズは、以下の目的で使用することができます。

- 特定のトランザクションを実行するたびに異なる値をサーバーに渡して機能テストを行う
- 所定の確率分布に基づいて送信する値を選択することで、実世界のユーザーの動作をシミュレートする

注: 送信する XML データ要素/属性の値のパラメータ化を設定したい場合は、Web アプリケーションについて説明したユーザー入力データの作業に進んでください。

XML アプリケーションの検証関数

XML 検証関数は、指定された XML 要素/属性 (XPath クエリにより特定される) のテスト実行時の値を検証します。

XML アプリケーションの検証関数

XML 検証関数は、指定された XML 要素/属性 (XPath クエリにより特定される) のテスト実行時の値を検証します。 **レンダリング** ページで右クリックして、XML 検証関数を挿入することができます。

TrueLog Explorer には、次の XML 検証関数があります。

- WebXmlVerifyNodeValue - 選択した要素の値を確認します。 この関数は、XML ツリー コントロールのコンテキスト メニューから 要素値の検証 を選択すると生成されます。
- WebXmlVerifyNodeAttribute - 選択した属性の値を確認します。 この関数は、XML ツリー コントロールのコンテキスト メニューから 属性値の検証 を選択すると生成されます。

再生中の検証チェック

Silk Performer のプロファイル設定で、検証関数を有効化/無効化することができます。 Silk Performer で、**設定** > **アクティブ プロファイル** を選択し、その後 **再生** > **Web** > **検証** > **HTML** > **XML** を選択します。

XML 検証 チェックボックスを使って、WebXmlVerifyNodeValue 関数と WebXmlVerifyNodeAttribute 関数を有効化/無効化します。

スクリプトのプロファイル設定は、WebSetOption という BDL 関数を使って上書きできます。

XML 検証関数を挿入する

すべての解析関数および検証関数は、レスポンス データの解析/検証を開始する Web API 呼び出しの前に指定する必要があります。 各 Web API 呼び出しの前には、複数の解析/検証関数を指定できます。

注: 解析/検証関数をテスト スクリプトに挿入する順番は関係ありません。

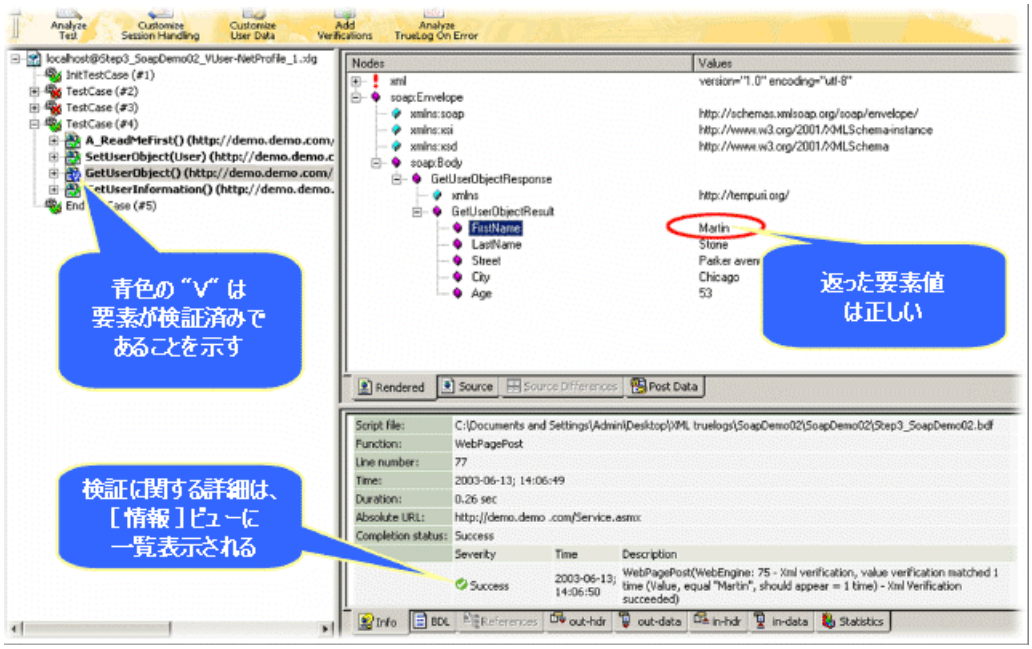
1. **レンダリング** ページで XML の要素または属性を選択して右クリックし、**要素値の検証** を選択します。
   代替方法*:* 属性の検証の場合には、コンテキスト メニューが **属性値の検証** コマンドになります。
   XML 検証関数の挿入 ダイアログが開きます。 選択した要素値が **値** 編集ボックスにあらかじめ設定され、**値** オプション ボタンが自動的に選択されます。
2. パラメータと照らして検証するには、**パラメータ** オプション ボタンを選択します。
   定数値に対する検証だけでなく、既存または新規のパラメータと照らして検証することもできます。
   a) 参照 (**...**) ボタンをクリックします。
   - パラメータが既に存在する場合には、そのパラメータを参照して選択します。

- パラメータが存在しない場合には、参照 (**...**) ボタンをクリックします。 **パラメータ ウィザード** が開いて、新しいパラメータを作成することができます。

3. 選択された要素が出現する頻度を、以下のように指定します。
   - このページでの出現回数 リスト ボックスから、ちょうど、以上、**以下** のいずれかを選択します。
   - **回** フィールドに数値を入力します。

   注: このダイアログの設定は、現在のページに対して正常な検証が保証される値に自動的に設定されています。 検証の許容度を上げたい場合にのみ、これらの設定を変更します (ちょうど 2 回を以上 1 回に変える、あるいは、検証で大文字/小文字を区別しない、など)。
4. 特定の Xpath でのみ要素を検証する場合は、**正確な Xpath クエリ** チェック ボックスをオンにします。
5. 検証をスクリプト全体のルールとして適用するかどうか、大文字/小文字を区別するかどうかを、該当するチェック ボックスで指定します。 **結果の変数名** テキスト ボックスには、結果変数名があらかじめ設定されています。必要に応じてこの名前を編集します。
6. ダイアログの **深刻度** の部分で、検証から否定的な結果が返されたときに生成する深刻度を指定します (エラー、警告、情報、または カスタム)。
7. **OK** をクリックして、関数をテスト スクリプトに追加します。

   XML の属性検証は、属性名も指定しなければならないことを除いて、XML の要素検証と同じです。
8. スクリプトの試行を行って、カスタマイズしたスクリプトが正しく実行されることを確認します。



SAPGUI アプリケーションでの作業

SAPGUI ベースのアプリケーションに対して、TrueLog Explorer では、TrueLog の記録、TrueLog の再生、基本的な TrueLog 解析、データ解析、コンテンツ検証、およびユーザー入力データのカスタマイズが利用できます。

この章を読み進む前に、TrueLog Explorer の基本機能を熟知していることが不可欠です。

SAPGUI TrueLog の構造

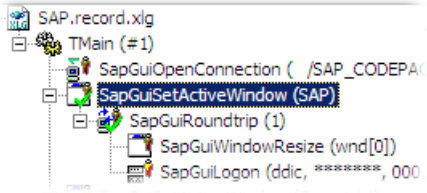
SAPGUI TrueLog は Oracle Forms TrueLog に似た構造を持ちます。 各 SapGuiSetActiveWindow の呼び出しは、新しい上位レベル ノードで発生します。 SapGuiSetActiveWindow の呼び出しでは、記録セッション中に起動される各ウィンドウでスクリプト化されます。 ウィンドウで実行されるすべてのアクショ

ン (コントロールの編集、リスト エントリの選択など) は、仮想 SapGuiRoundtrip ノードによってグループ化されたサブノードとして表示されます。

SAPGUI TrueLog の関数

TrueLog Explorer には、上位レベルの SAPGUI 関数が 2 つあります。

- SapGuiActiveSetWindow- 新しい GUI ウィンドウの生成を示す最上位の API ノードです。 ウィンドウで実行されたすべてのアクションが、SapGuiActiveSetWindow 関数の下にグループ化されます。
- SapGuiRoundTrip- これは仮想ノードです。実際には、サーバーに送信される SapGuiRoundTrip という API 呼び出しはありません。 これらのノードは、サーバー ラウンドトリップの過程で発生するすべてのクライアント側アクションをグループ化するために使用されます。 ラウンドトリップ前とラウンドトリップ後の両方の状態を表示できます。 各 SapGuiSetActiveWindow ノードの下には、複数のラウンドトリップ ノードが含まれる場合があります。



SAPGUI TrueLog をステップ スルーする

テスト後は、通常、複数の TrueLog On Error ファイルが TrueLog Explorer にロードされます(エラーを返した仮想ユーザーごとに 1 つの TrueLog)。 エラーが発生した順に、次々とエラーを移動できます。 この機能によって、開いているすべての TrueLog を手動で確認して次のエラーを検索する必要がなくなるので、エラー分析プロセスが簡単になります。

1. 負荷テストが終了したら、Silk Performer の **結果の検討** をクリックします。

   注: TrueLog On Error ファイルが作成されるのは、Silk Performer の **エラー時に TrueLog を生成する** オプションが有効になっている場合だけです。

   結果の検討 ダイアログ ボックスが表示されます。
2. **TrueLog Explorer** をクリックします。

   注: Silk TrueLog Explorer の起動 ボタンは、テスト時にエラーが検出されなかった場合や Truelog On Error が有効でない場合は、無効になっています。 そのような場合は、直接 Performance Explorer による結果の分析に進みます。

   TrueLog Explorer が起動されて、現在のテストで生成された TrueLog On Error ファイルがロードされ、**TrueLog のステップ スルー** ダイアログが表示されます。
3. **TrueLog のステップ スルー** ダイアログ ボックスで、適切なオプション ボタンを選択します。 以下を選択できます。

   - すべての [API 呼び出し] (各 API ノード)
   - すべての [ラウンドトリップ] (SapGuiRoundtrip 呼び出し)
   - すべての [ウィンドウ] (SapGuiSetActiveWindow ノード)
   - すべての [カスタマイズ可能な呼び出し] (フィールド値の変更をカスタマイズ可能な次のノード)
   - すべてのエラー

   TrueLog On Error ファイルは、記録された順に検索されます。
4. **次を検索** をクリックして、最初の API 呼び出し、ラウンドトリップ、ウィンドウ、カスタマイズ可能な呼び出し、またはエラーに進みます。

エラー メッセージは、[情報] ペインに表示されます。 **TrueLog** メニュー ツリーでは、再生エラーを含む各 API ノードに赤い "X" が付加されています。

SAPGUI テスト スクリプトを分析する

テスト スクリプトの生成が終わったら、スクリプトの試行を実行して、そのスクリプトがエラーなしで実行されるかどうかを確認できます。 スクリプトの試行は、スクリプトが正確に意図したトランザクションを再作成するかどうかを判定します。

スクリプトの試行のデフォルトのオプション設定には、テスト時にダウンロードされたデータのリアルタイム表示や、ログ ファイルおよびレポート ファイルの作成などがあります。

スクリプトの試行では、1 人の仮想ユーザーのみが実行され、トランザクション間の思考時間遅延が発生しないストレス テスト オプションが有効になります。

1. Silk Performer ワークフロー バーで、**スクリプトの試行** をクリックします。 **スクリプトの試行** ダイアログ ボックスが表示されます。
2. スクリプトの試行中にレンダリングされたページのトランザクションを表示するには、**TrueLog Explorer によるアニメーション実行** チェック ボックスをオンにします。
3. **実行** をクリックします。

   注: ここでは、実際の負荷テストではなく、スクリプトのデバッグが必要かどうかを確認するためのテスト実行のみを実行します。

   スクリプトの試行が開始されます。 Silk Performer の **監視** ウィンドウが開き、実行の進捗についての詳細な情報が表示されます。

   TrueLog Explorer が開き、スクリプトの試行の実行中にダウンロードされたデータが表示されます。記録中にアクセスされた主要な SAPGUI ウィンドウは、それぞれ TrueLog メニュー ツリーの上位の SapGuiSetActiveWindow API ノードとして表示されます。 記録されたすべてのサーバー ラウンドトリップとユーザー アクションは、対応する SapGuiSetActiveWindow ノードのサブノードとして表示されます。

再生 TrueLog と記録 TrueLog

このセクションでは、SAP の記録 TrueLog と再生 TrueLog の差分を分析する方法について説明します。

再生 TrueLog と記録 TrueLog を比較する

SAPGUI アプリケーションをテストしたとき、再生 TrueLog と記録 TrueLog の間で、対応するコントロール値に差分が生じることがあります。 それらの一部は、再生エラーの原因となる場合があります。

1. 再生 TrueLog を開き、**テストの分析** をクリックします。 **テストの分析** ダイアログ ボックスが表示されます。
2. **テスト実行の比較** をクリックします。 関連する記録 TrueLog が [比較ビュー] に表示されます。 **TrueLog のステップ スルー** ダイアログ ボックスが表示されます。
3. **TrueLog のステップ スルー** ダイアログ ボックスで、適切なオプション ボタンを選択します。 以下を選択できます。

   - すべての [API 呼び出し] (各 API ノード)
   - すべての [ラウンドトリップ] (SapGuiRoundtrip 呼び出し)
   - すべての [ウィンドウ] (SapGuiSetActiveWindow ノード)
   - すべての [カスタマイズ可能な呼び出し] (フィールド値の変更をカスタマイズ可能な次のノード)
   - すべてのエラー

   TrueLog On Error ファイルは、記録された順に検索されます。
4. **次を検索** をクリックして、最初の API 呼び出し、ラウンドトリップ、ウィンドウ、カスタマイズ可能な呼び出し、またはエラーに進みます。

5. コントロールおよびスクリーンショットの値と状態をビジュアルに比較して、差分があるかどうかを確認します。 見つかった差分に応じて、必要なカスタマイズを行います。
6. **スクリプトの試行** をクリックして、カスタマイズがエラーなしで実行されることを確認します。
7. スクリプトが十分にカスタマイズされ、必要なカスタマイズがすべて追加されるまで、この手順を繰り返します。
8. 次の手順を実行し、比較モードをオフにします。
   - **表示** > **比較モード** を選択します。
   - **比較モード** アイコンをクリックします。

再生 TrueLog と記録 TrueLog を同期させる

比較モードでは、記録値と再生値の差分を識別できるように、再生 TrueLog と記録 TrueLog の間で対応する API ノードを同期することができます。

注: この機能は、TrueLog の自動同期が有効になっている場合は無効になります。

1. 次のいずれかを実行して比較モードを有効にします。
   - **表示** > **比較モード** を選択します。
   - ツールバーの **比較モード** ボタンをクリックします。
2. 対応する一組の記録/再生 TrueLog を開きます。
3. API ノードを右クリックし、**TrueLog の同期** を選択します。 TrueLog Explorer は、対応する TrueLog 内で、選択された API ノードと最も相関がある API ノードを見つけます。

SAPGUI テストスクリプトのカスタマイズ

Silk Performer を利用してテスト スクリプトを生成し、スクリプトの試行を実行したら、TrueLog Explorer では、以下の方法によってスクリプトをカスタマイズできます。

- コンテンツ検証関数 - **検証の追加** ツールを使用して、サーバーが送信したコンテンツが実際に受信されたことを検証することによって、テスト中にダウンロードされたデータを分析できます。 検証は、システム開発後も、稼働中のパフォーマンス管理に役立ちます。
- 解析関数 -TrueLog Explorer では、**ソース** ビューと **コントロール** ページで、SAPGUI 解析関数をビジュアルに挿入できます。 手動でコードを記述する必要はありません。TrueLog Explorer によって、自動的にスクリプト内に解析関数が生成されます。
- パラメータ化されたユーザー データ - ユーザー入力データのカスタマイズを利用すると、静的なユーザー入力データを、トランザクションごとに変更されるパラメータ化された動的ユーザー データに置き換えることによって、テスト スクリプトを現実に近いものにできます。 このようなデータ駆動型テストを作成するために、手動でのスクリプト記述は必要ありません。

入力データを変更する各 SAPGUI 関数呼び出しに対して、戻り値の検証、値の解析、および入力データのカスタマイズができます。 これらの操作は、**ソース** ビュー (コントロール内を右クリック) と **コントロール** メニュー ツリーの両方から実行できます。

フォーム フィールド用のユーザー入力データをカスタマイズする

現実の条件の下では、SAPGUI アプリケーションのユーザーは、予測できないような組み合わせのデータをフォームに投稿します。 効果的な SAPGUI アプリケーション テストの目標は、テスト スクリプトによって、そのような変則的で多様なユーザーの動作をエミュレートすることです。

パラメータ ウィザード を使用すると、テスト中にフォームに入力されるユーザー入力データをカスタマイズできます。 [パラメータ ウィザード] によって、フォーム フィールドに入力されるデータを指定できます。また、記録されたユーザー入力データをランダムなパラメータ化されたユーザー データに置き換えることによって、テスト スクリプトを現実に近いものにできます。

1. **TrueLog** メニュー ツリーで TrueLog を選択します。 TrueLog の内容が **レンダリング** ページと **情報** ページに表示されます。
2. **編集** > **TrueLog のステップ スルー** を選択します。

   代替方法:ツールバーの **TrueLog のステップ スルー** をクリックします。

   TrueLog のステップ スルー ダイアログ ボックスが開きます。
3. **カスタマイズ可能な呼び出し** オプション ボタンをクリックし、**次を検索** をクリックして、TrueLog にある入力のカスタマイズが可能なすべてのフォーム フィールドをステップ スルーします。

   注: カスタマイズ可能なコントロールは、オレンジ色の輪郭で表示されます。 既にカスタマイズされているコントロールは、緑色の輪郭で表示されます。 青色の輪郭で表示されるコントロールは、値の解析または検証は可能ですが、カスタマイズはできません。
4. **ポスト データ** ページで、カスタマイズしたいフォーム コントロールを右クリックし、**値のカスタマイズ** を選択します。

   記録されている値を、さまざまな種類の入力データ（ファイルにあらかじめ定義した値や一般的なランダム値を含む）に置き換えることができます。また、記録された入力データをカスタマイズした値に置き換えるためのコードをテスト スクリプトに生成することもできます。

   パラメータ ウィザード が開きます。

   パラメータ ウィザード では、次の 2 つの方法でスクリプトの値を変更できます。

   - スクリプトの dclparam または dclrand のセクションで定義されている既存のパラメータを使用する
   - 新しい定数値、ランダム値、複数列データ ファイル内の値に基づいた、新しいパラメータを作成する

   新しいパラメータを作成すると、そのパラメータは既存パラメータ群に追加され、以降のカスタマイズで利用できるようになります。
5. **パラメータを新規に作成する** オプション ボタンをクリックし、それから **次へ** をクリックして新しいパラメータを作成します。 **パラメータの新規作成** ページが開きます。
6. **ランダム変数からパラメータを作成** オプション ボタンをクリックし、それから **次へ** をクリックします。 **ランダム変数** ページが開きます。
7. テスト スクリプトに挿入したいランダム変数の種類をリストボックスから選択し、**次へ** をクリックします。

   選択した変数の種類の簡単な説明が下部のウィンドウに表示されます。

   変数の名前と属性の指定 ページが開きます。
8. 変数の名前を **名前** テキスト ボックスに入力します。
9. **終了** をクリックします。 これで、テスト スクリプトは、そのフォーム フィールドについて、記録された値ではなくランダム変数を使用するようになります。 新しいランダム変数関数が **BDL** ページに追加されます。

テスト スクリプトにランダム変数関数を追加した状態でスクリプトの試行を起動し、スクリプトがエラーなしに動作することを確かめます。

複数列データ ファイル

複数列データ ファイルを使ったパラメータ化は、データをパラメータ化する有効な方法です。 特定の文字列値の組み合わせを保存したファイルを定義します。 データ ファイルの各列はそれぞれのパラメータに対応します。 複数列データ ファイルを使うと、データ駆動型テスト モデルが利用できるようになり、すべてのユーザー データ入力をただ 1 つのデータ ファイルでカバーできます。

SAPGUI の [コントロール] メニュー ツリー

入力データを変更する各 SAPGUI 関数呼び出しに対して、**ソース** ビューと情報ペインの **コントロール** メニュー ツリーの両方から、戻り値の確認、値の解析、および入力データのカスタマイズができます。 **コントロール** メニュー ツリーは、フォーム フィールドを表示したりカスタマイズするための便利な手段を提供

します。 **コントロール** メニュー ツリーは、BDL スクリプトのフォーム セクションと、それに含まれるすべての名前/値ペアをビジュアルに表現したものです。

カスタマイズ可能なコントロールは、オレンジ色または青色の輪郭で表示されます。 一般に、オレンジ色の輪郭で表示されるコントロールは、値の解析、値の検証、およびユーザー データのカスタマイズが可能です。 青色の輪郭で表示されるコントロールは、値の解析と値の検証は可能ですが、ユーザー データのカスタマイズはできません。 コントロールを右クリックし、実行したいスクリプト カスタマイズの種類を選択します。

ユーザー データが既にカスタマイズされているコントロールは、緑色の輪郭で表示されます。

コピー コントロール ID 機能

Copy Control ID 機能を使用すると、選択した SAPGUI コントロールのコントロール ID をクリップボードにコピーできます。 この機能によって、Silk Performer 内の特殊な呼び出しの非ビジュアルのスクリプト記述が簡単になります。 コントロール ID をコピーするには、GUI コントロールを右クリックし、コンテキスト メニューから **コントロール ID のコピー** を選択します。 コントロール ID がクリップボードにコピーされます。

注: コントロールは、**ソース** ビューまたは **コントロール** メニュー ツリーから選択できます。

検証および解析関数

ほとんどのコントロール タイプで検証と解析を実行できます。 TrueLog Explorer には、SAPGUI TrueLog 用の標準的な検証および解析ウィザードが備わっています。 ウィザードにアクセスするには、スクリーンショットまたはコントロール ツリーでコントロールを右クリックします。 その後、コンテキスト メニューから、検証および解析ウィザードを起動します。

検証および解析関数は、ツリーで現在選択している API ノードの後にスクリプト化されます。 検証および解析関数をウィンドウの最後の API 呼び出しの後に挿入すると、再生エラーが発生することがあります。 たとえば、現在選択している API ノードが **閉じる** ボタンを押すことである場合、現在のウィンドウとそのウィンドウのすべてのコントロールがこのアクションで破棄されます。 よって、この呼び出しの後にスクリプト化されている検証/解析関数は、再生中に「ハンドラが見つかりません」エラーにより失敗します。 このため、ウィンドウの最後のノードに挿入される検証/解析関数を追加しようとする場合、TrueLog Explorer では確認メッセージが表示されます。

TrueLog Explorer では、コンテンツ チェックを追加して、サーバーから送信されたコンテンツが現実の条件下でクライアントによって本当に受信されるかどうかを簡単に確認できます。 入力データが挿入される任意の SAPGUI 関数呼び出しに対して、戻り値検証関数を挿入できます。 検証関数は、**ソース** ビューまたは **コントロール** メニュー ツリーから挿入できます。

再生テスト実行と記録テスト実行を比較することによって、TrueLog Explorer では、SAPGUI アプリケーションが過負荷状態にあるときに、テキスト、グラフィック、フィールド データ、その他のデータがクライアントによってダウンロードされて表示されるかどうかを視覚的に確認できます。 これによって、他の SAPGUI トラフィック シミュレーション ツールでは検出できないようなエラー、つまり、標準的なテスト スクリプトでは検出されない負荷がかかっている場合にのみ発生するエラーを検出できるようになります。

コンテンツ検証関数は、稼働中のパフォーマンス管理にも利用できるので、システムの配置後も役立ちます。

値の検証を追加する

検証するコントロールを右クリックすることによって、必要な検証関数を生成して BDL スクリプトに挿入できます。 TrueLog Explorer には、SAPGUI アプリケーション用にあらかじめ有効になっている検証関数が 1 つあります。

1. **ソース** ビューまたは **コントロール** メニュー ツリーで、戻り値を検証するコントロール フィールドを選択します。

検証可能なコントロールは、青色の輪郭で表示されます。 オレンジ色の輪郭で表示されるコントロールも、ほとんどのものは検証可能です。

2. コントロール フィールド内を右クリックし、**値の検証** を選択します。 **値検証関数の挿入** ダイアログが表示されます。 このダイアログを使用すると、BDL スクリプトに挿入する検証関数のタイプを指定できます。
3. **検証内容 選択したコントロールの値が以下に** リストボックスから、(必要に応じて) **等しい** または **等しくない** を選択します。
4. **定数値** または **パラメータ値** のどちらに対する検証であるかを指定します。
5. 検証で **大文字/小文字を区別する** かどうかと、空白を無視するかどうかを指定します。
6. **深刻度** グループ ボックスで、検証から否定的な結果が返されたときに生成する深刻度を指定します。
7. **OK** をクリックします。 関数がテスト スクリプトに追加されます。

BDL スクリプトが正常に変更されたら、BDL スクリプトに追加するその他の検証についても、それぞれこの手順を繰り返します。

SAPGUI 解析関数を追加する

入力データが挿入される任意の SAPGUI 関数呼び出しに対して、値解析関数を挿入できます。 値の解析は、正確な再生ができるようにセッション固有の文字列をパラメータ化する必要がある場合に役立ちます。 たとえば、顧客 ID を解析して customer ID フィールドに挿入するパラメータにできます。

ツールバーとタイトル バーは解析できませんが、テキスト フィールド、フィールド ラベル、コンボ ボックス、ボタンなど、ほとんどの一般的なコントロールは解析できます。

1. **ソース** ビューまたは **コントロール** メニュー ツリーで、値を解析するコントロール フィールドを選択します。

   検証可能なコントロールは、青色の輪郭で表示されます。 オレンジ色の輪郭で表示されるコントロールも、ほとんどのものは値の解析が可能です。
2. コントロール フィールド内を右クリックし、**値の解析** を選択します。 **値解析関数の挿入** ダイアログが開き、解析関数を調整する設定が表示されます。
3. 省略可能： **パラメータ名** テキスト ボックスに、解析関数の結果を受け取るパラメータの名前を入力します。
4. 省略可能： 通知用の Print 文をスクリプトに挿入する場合は、**スクリプトに出力文を挿入** 領域で **Print 文** をオンにします。 解析関数の結果が Silk Performer の **仮想ユーザー** 出力ウィンドウに出力されます。
5. 省略可能： (Print 文によって **仮想ユーザー出力** ウィンドウに値を書き込むだけでなく)デバッグに役立つように解析結果を出力ファイルに書き込むには、**スクリプトに出力文を挿入** 領域で **Writeln 文** (write line 文)をオンにします。

   出力ファイルを生成すると負荷テストの時間測定値が変化するので、これらのファイルはデバッグのためだけに使用し、完全な負荷テストでは生成しないでください。
6. **OK** をクリックします。 解析文がテスト スクリプトに挿入されます。

スクリプトによるセッション情報およびユーザー入力データの処理方法のカスタマイズ、必要なすべての検証関数の追加、および Silk Performer を使用した BDL スクリプトの手動編集が完了したら、テスト スクリプトはエラーなしで動作します。

Oracle Forms アプリケーションでの作業

このセクションでは、スクリプトの試行の実行結果に基づいて Oracle Forms テスト スクリプトをカスタマイズする方法について説明します。

Oracle Forms アプリケーションでの作業 - 概要

Oracle Forms (旧称「SQL*Forms」) は、Oracle の iDS (Internet Developer Suite) のコンポーネントの 1 つです。 Oracle の iAS (Internet Application Server) Forms Services を使用してフォームを Web 全体に配置できる、第 4 世代言語の RAD (Rapid Application Development) 環境です。

Oracle Forms は Java テクノロジに基づいているため、Oracle Forms トランザクションを記録および再生するには、Silk Performer のプロファイル設定を使用して、事前に Java 仮想マシンを構成する必要があります。 Oracle Forms 6i 以降の記録中は、Java の Just-In-Time コンパイラを無効する必要もあります。

この章を読み進む前に、TrueLog Explorer の基本機能を熟知していることが不可欠です。

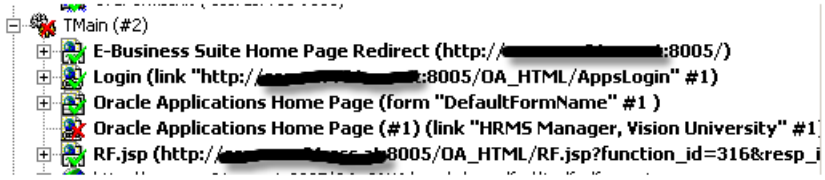
Oracle Applications 12i のセッション情報をカスタマイズする

icx_ticket パラメータが正しくカスタマイズされていることを確認します。

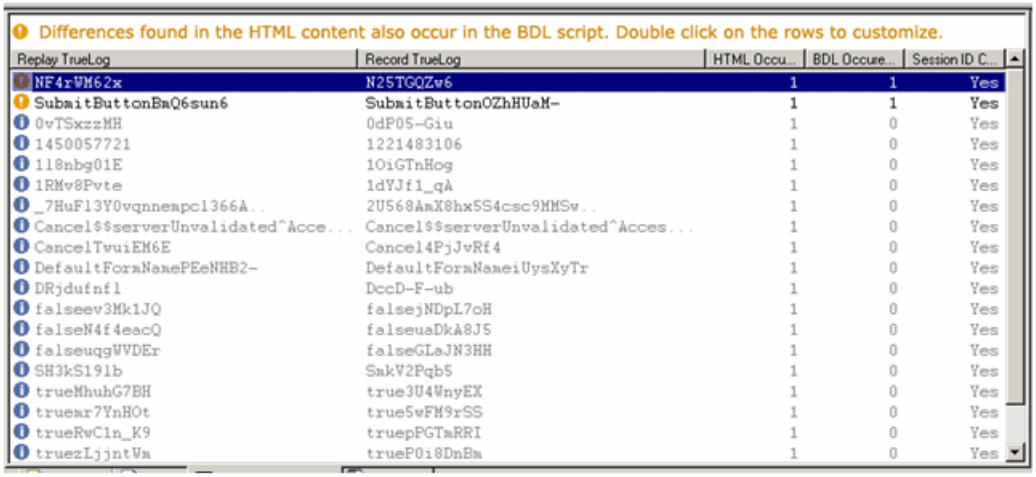
1. **ワークフロー バー** の **スクリプトの試行** ボタンをクリックして、スクリプトの試行を実行します。

   注: TrueLog の生成が、Silk Performer で有効にされている必要があります。

2. スクリプト試行の結果の TrueLog を TrueLog Explorer で開きます。 通常、HTML ページ (ログイン後すぐに表示される) がエラーを表示します。
   エラーの例：



3. **ワークフロー バー** の **テストの分析** ボタンをクリックします。
4. **ワークフロー - テストの分析** ウィザード ページで、**テスト実行の比較** を選択します。
5. **TrueLog のステップ スルー** ダイアログを閉じます。
6. 記録されたアプリケーションのログイン ページに対応する TrueLog ノードを選択します。
7. ノードを右クリックし、コンテキスト メニューから **TrueLog の同期** を選択します。
8. TrueLog Explorer メニュー バーから **編集** > **TrueLog のタイプ** > **Web** を選択します。
9. **ソース差分** タブをクリックします。



10. 記録された TrueLog と再生 TrueLog の差分を示す行を右クリックします。この差分は、スクリプトで発生しており、カスタマイズする必要があるセッション情報 (黄色で強調表示される) を含んでいる可能性があります。
11. Silk Performer でスクリプトの試行を再度実行して、セッション情報のカスタマイズが正常に行われたことを確認します。

Oracle Forms TrueLog の構造

このセクションでは、Oracle Forms TrueLog 構造および **フォーム コントロール** ウィンドウについて説明します。

Oracle Forms TrueLog の構造 - 概要

Oracle Forms アプレットは一般に、Oracle Forms プロトコルと HTTP プロトコルの両方をベースに構築されたアプリケーションに組み込まれています。 このようなアプリケーションを記録するとき、生成される BDL スクリプトには、Oracle Forms と Web 関数呼び出しが含まれます。 Oracle Forms の TrueLog ではこのような混合プロトコルがサポートされます。 **TrueLog** メニュー ツリーには、アクティブな BDL スクリプトに含まれる Oracle Forms と HTTP 関数呼び出しの両方が一覧表示されます。 メニュー ツリーの各 API ノードは、一意の関数呼び出しを表します。

Oracle Forms の TrueLog には **フォーム コントロール** ウィンドウがあり、選択した Oracle Forms アプレット ウィンドウの要素が表形式のデータとして表示されます (他の場合には、このバイナリ プロトコルは読み取り不可)。 Oracle Forms アプレットが埋め込まれた HTML ページを呼び出す HTTP 呼び出しでは、**フォーム コントロール** ウィンドウには完全にレンダリングされた HTML が表示されます。

Oracle Forms TrueLogs には、一意の構造があります。 各 OraFormsSetWindow の呼び出しは、新しい上位レベル ノードで発生します。 OraFormsSetWindow の呼び出しでは、記録セッション中に起動される各ウィンドウで、スクリプト化されます。 ウィンドウで実行されるすべてのアクション (たとえば、テキスト コントロールの編集、リスト エントリの選択など) は、各 OraFormsSetWindow のサブノードに表示されます。

OraFormsInit は、Init トランザクションに含まれています。 OraFormsDestroy は、End トランザクションに含まれています。 メイン トランザクションの最初の Oracle API 呼び出しは OraFormsConnect です。

ノード情報

各 Oracle Forms ノードには、アクティブ化されたウィンドウのすべてのコントロールの現在の状態に関する情報が保存されます。 ウィンドウの最初の外観を表す OraFormsSetWindow ノードを選択する場合、フォームの初期作成時に存在したすべてのコントロールおよび初期値が表示されます。

サブノードを表示する場合、選択したアクションが満たされた後で存在したすべてのコントロールの状態および初期値が表示されます。 たとえば、OraFormsEditSet を選択すると、このアクションが満たされた後のコントロールの状態が表示されます。

ウィンドウが再度有効にされ (つまり、一時的に代替ウィンドウが有効にされ)、再度有効にされたウィンドウの OraFormsSetWindow が選択されると、再度有効にされた後のコントロールのステータスが表示されます (ほとんどの場合は、ウィンドウが再度有効にされたときと同じステータスです)。

Web 呼び出しを処理する

Oracle Forms API 呼び出しに加え、HTTP コンテンツをダウンロードする Web コールは、TrueLogs にも含まれます。 各メイン トランザクションの最初の呼び出しは、テスト中に Oracle Forms アプリケーションの最初のページをダウンロードする Web 呼び出しです。

TrueLog Explorer で利用できる機能は、開いている TrueLog のプロトコルに応じて変化します。 Oracle Forms は、スクリプトに Web 呼び出しが含まれることがある混合プロトコルのため、TrueLog Explorer では、バイナリの Oracle Forms アプレット用のデータをベースにした表と HTML ページ用のレンダリング HTML ビューを切り替えできます。 Web ベースのプロトコルが重要なのは、セッション処理のカスタマイズ、検証関数の挿入、HTML の値の解析、および Oracle Forms スクリプトに含まれる HTML での検索を実行できるからです。

1. **編集** > **TrueLog のタイプ** を選択します。
2. **Web** (デフォルト) を選択します。

[受信データ]/[送信データ] タブ

Silk Performer と Oracle Forms サーバー間で通信されるデータは、[受信データ] タブと [送信データ] タブで追跡できます。 Oracle Forms の再生中、[受信データ] タブには、Silk Performer が Oracle Forms サーバーから受信したデータが表示されます。 [送信データ] タブには、Silk Performer が Oracle Forms サーバーに送信したデータが表示されます。

注: これらのメッセージを利用できるのは、**プロファイル設定** ダイアログでログ オプションを **デバッグ** に設定している場合だけです。

データのカスタマイズ中はメッセージを無視できます (ただし、GET メッセージのカスタマイズ中は無視できません)。 ただし、問題が発生し、カスタマ ケアに連絡する場合には、分析するためにこの情報が必要になります。 記録 TrueLog と再生 TrueLog の間に大きな違いがあることに気付くことがあります。 メッセージはすべて含まれていると思われますが、対応するノードでログに記録されておらず、1 つ前または後のノードでログに記録されている場合があります。

Silk Performer と Oracle Forms サーバー間で通信される各メッセージ ブロック (OraForms 呼び出し) は、以下のメッセージ タイプまたはその組み合わせで構成されます。

項目	説明
Create	新しい UI 要素を作成する必要があることを示します。 作成が必要な各 UI オブジェクトのクラス、プロパティ、および ID が指定されます。 例 : サーバーからクライアントに、指定された種類、名前、ID、および場所に基づいてテキスト ボックスを作成するように指示します。
Destroy	UI 要素を破棄する必要があることを示します。 例 : ウィンドウを閉じるとき、ウィンドウにあるすべてのコントロールは破棄する必要があります。
Update	アプレット内部の要素がどのような状態であるかを示します (たとえば、位置、フォーカス、選択ステータスなど)。

項目	説明
	例：クライアントからサーバーに、マウス カーソルを配置する場所を問い合わせます。 そこでサーバーは、アプレットのインターフェイス内でマウス カーソルの位置を変えることによって、応答します。
Get	クライアントとサーバー間の内部通信を示します。 Get メッセージは UI では表示されません。 例：サーバーがクライアントに、対象のコントロールの ID を問い合わせます。 Get メッセージは一般に最後に Terminal 3 メッセージが付きます。
Terminal	通信のラウンド トリップの終端を示します。 各メッセージ ブロックの終端に Terminal メッセージが付きます。 3 種類の Terminal メッセージを利用できます。 • Terminal 1 • Terminal 2 • Terminal 3

最も一般的なメッセージ ブロックは、クライアントからサーバーに update メッセージと、通信を終了する Terminal 1 メッセージを送信するというものです。 これに対して、サーバーは、通常、Terminal 1 メッセージが最後に付いた destroy メッセージや create メッセージで応答します。

[受信データ]/[送信データ] のメッセージ ブロックの例

マウス クリック関数では、クライアントがサーバーに update メッセージをプロパティとともに送信します。 プロパティの 1 つは、マウス カーソルの位置です。 もう 1 つのプロパティは、マウスがクリックされた状態であることを示すものです。 通信のラウンド トリップは、Terminal メッセージで終了します。 カーソルの位置が変わり、クリックされた状態に変わったこと示すレスポンスをサーバーが返します。

```
MSGTYPE:          UPDATE
CLASS:            0/0
ID:               2824
TITLE:            N/A
RESPONSE:         0
PROPERTIES:
TYPE:   PROP_TYPE_INTEGER
Name:   MENU_MENUUPDATE/367
Value:  2855

MSGTYPE:          DESTROY
CLASS:            0/0
ID:               2855
TITLE:            N/A
RESPONSE:         0
PROPERTIES:

MSGTYPE:          GET
CLASS:            0/0
ID:               1644
TITLE:            N/A
RESPONSE:         0
PROPERTIES:
TYPE:   PROP_TYPE_VOID
Name:   VALUE/131
Value:  null
```

```
MSGTYPE:          TERMINAL
CLASS:            0/0
ID:               0
TITLE:            N/A
RESPONSE:         3
PROPERTIES:
```

Terminal メッセージ

Terminal メッセージは、クライアントとサーバー間の通信のラウンドトリップを終了させるために使用されます。 次の 3 種類の Terminal メッセージがあります。

Terminal 1 メッセージ: これは、一番直接的なメッセージ終了方法です。 Terminal 1 メッセージは常にクライアントが開始します。 たとえば、クライアントがサーバーに「ユーザーがこのボタンを押しました。UI に何を表示しますか」と問い合わせます。そこでサーバーはリクエストの回答を返し、Terminal 1 メッセージで通信を終了します。

Terminal 2 メッセージ: 場合により、クライアントとサーバー間の通信はさらに複雑になります。 たとえば、クライアントが「ユーザーがこのボタンを押しました。UI に何を表示しますか」と問い合わせた場合、回答する前に、サーバーでさらにクライアントからの情報が必要なことがあります。 サーバーは Terminal 2 メッセージで応答します。このとき「オプション ボタン XYZ の状態は」などと問い合わせます。クライアントがこのリクエストに対するスクリプト化レスポンスを準備している場合は、サーバーの問い合わせに対する回答が送信され、Terminal 2 メッセージで通信を終了します。 サーバーはクライアントの最初の問い合わせに対するスクリプト化レスポンスを送信し、Terminal 1 メッセージで通信を終了します。

このようなメッセージ交換は次のとおり説明できます。

1. クライアントからのリクエスト (Terminal 1、[送信データ] タブ)
2. サーバーからのリクエスト (Terminal 2、[受信データ] タブ)
3. クライアントからのレスポンス (Terminal 2、[送信データ] タブ)
4. サーバーからのレスポンス (Terminal 1、[受信データ] タブ)

Terminal 3 メッセージ: クライアントからの値をリクエストする Get 呼び出しをサーバーが自発的に開始することを示します。 これらはクライアントで開始されないリクエストです。 例：サーバーから「キャッシュが空です。オプション ボタン XYZ の値を再度送信してください。」と問い合わせます。クライアントは適切な回答を返信する必要があり、Terminal 3 メッセージで通信を終了します。

Terminal 3 メッセージでは結果として、再生スクリプトが正しく作成されないことがあります。 これらのサーバーのリクエストは予測どおりでないので、クライアントはデフォルトの回答 (null など) のみを送信できます。 サーバーでクライアントからのデフォルトの回答を受け入れる場合、たとえば、Get 呼び出しで特定の回答が本当には必要ない場合は、再生が続行されます (多くの場合に該当)。 ただし、サーバーがデフォルト レスポンスを受け入れ不能とみなす場合は、OraFormsSet 関数と OraFormsOnMessageGet 関数を手動でスクリプト化し、サーバーの Get 呼び出しに対する正しいレスポンスを作成する必要があります。

例：

```
OraFormsSetRectangle("VISIBLERECT", 0, 0, 119, 24,
ORA_SET_TYPE_MESSAGEGET);
OraFormsOnMessageGet("LINE_CUSTOMER_ITEM_DSP_0"); // Requested
Item
OraFormsMouseClick("LINE_CUSTOMER_ITEM_DSP_0", 70, 18, 0); //
Requested  Item
```

Terminal -1 メッセージ:	クライアントとサーバー間の通信でのエラーを示します。 サーバーがリクエストに対して受け入れ不能な回答を受信したので、再生は失敗します。

Oracle Forms テストのエラーを分析する

Oracle Forms の負荷テストを完了します。

テストが終了すると、通常、TrueLog Explorer に複数の TrueLog On Error ファイルが開きます (エラーを返した仮想ユーザーごとに 1 つの TrueLog)。 TrueLog Explorer の **エラーを検索** 機能を利用すると、エラーがどの TrueLog に記録されているかにかかわらず、時間の経過順に 1 つずつ エラーをたどることができます。 これによって、エラー分析プロセスが簡単になります。開いているすべての TrueLog を手動で確認して、次のエラーを検索する必要がなくなるからです。

1. Silk Performer で **結果の検討** ボタンをクリックします。

   注: TrueLog On Error ファイルが作成されるのは、Silk Performer の **エラー時に TrueLog を生成する** オプションが有効になっている場合だけです。

   ワークフロー - 結果の検討 ダイアログ ボックスが開きます。
2. **TrueLog Explorer** をクリックします。

   注: 以下の状況では、**Silk TrueLog Explorer の起動** ボタンは無効になります。

   - テスト時にエラーが検出されなかった
   - Truelog On Error が有効にされていない

   そのような場合は、直接 TrueLog Explorer による結果の分析に進みます。

   TrueLog Explorer が起動され、現在のテストに対して生成された **TrueLog On Error** ファイルが開いて、TrueLog のステップ スルー ダイアログ ボックスが開きます。
3. すべての **フォーム送信** (OraFormsSetWindow 呼び出し)、すべての **API 呼び出し** (各 API ノード)、またはすべての **エラー** 内を移動できます。

   a) ダイアログ ボックスで、適切な **ラジオ** ボタンを選択します。

   TrueLog On Error ファイルは、時間の経過順に検索されます。
4. **次を検索** をクリックして、TrueLog 内を移動します。

   エラー メッセージは、**情報** タブに表示されます。

   TrueLog メニュー ツリーでは、再生エラーが含まれている API ノードには、赤い "X" 印が付いています。

Oracle Forms テスト スクリプトを分析する

テスト スクリプトを生成したら、スクリプトの試行を行うことによって、スクリプトがエラーなしで動作するかどうかを確認できます。 スクリプトの試行によって、記録された操作がスクリプトで正確に再現されるかどうかがわかります。

スクリプトの試行のデフォルトのオプション設定には、テスト時にダウンロードされるデータの実況表示や、ログ ファイルおよびレポート ファイルの作成などがあります。

スクリプトの試行では、1 人の仮想ユーザーのみが実行され、トランザクション間の思考時間遅延が発生しないストレス テスト オプションが有効になります。

1. Silk Performer のワークフロー バーで、**スクリプトの試行** をクリックします。 スクリプトの試行 ダイアログ ボックスが開きます。
2. スクリプトの試行の実行中にページの遷移を表示するには、**TrueLog Explorer によるアニメーション実行** チェック ボックスをオンにします。
3. **実行** をクリックします。

注: ここでは、実際の負荷テストではなく、スクリプトのデバッグが必要かどうかを確認するためのテスト実行のみを実行します。

スクリプトの試行が開始されます。 **監視** ペインが開き、スクリプトの試行の進行状況についての詳細な情報が表示されます。

4. Silk Performer が開き、スクリプトの試行の実行中にダウンロードされたデータが表示されます。

   記録中にアクセスされた主要な **Oracle Forms** ウィンドウは、TrueLog Explorer のメニュー ツリーでそれぞれ上位の API ノードとして表示されます。 記録されたすべてのアクションは、サブノードとして表示されます。
5. スクリプトの試行の実行中にエラーが発生した場合は、TrueLog Explorer を使用して、エラーの検索やスクリプトのカスタマイズを行うことができます。

Oracle Forms の再生 TrueLog と記録 TrueLog を比較する

TrueLog Explorer で TrueLog を開きます。

Oracle Forms アプリケーションをテストしたとき、再生 TrueLog と記録 TrueLog の間で、対応するコントロール値に差分が発生していることがあります。 より深刻なエラーは、記録 TrueLog と再生 TrueLog の間に差分が発生する (たとえば、テスト スクリプトで入力データに対して必要なカスタマイズが行われていないために、重複レコードが入力されたことがメッセージ ボックスで示される) 原因になる可能性もあります。

1. **テストの分析** をクリックします。 **ワークフロー - テストの分析** ダイアログ ボックスが開きます。
2. **テスト実行の比較** をクリックします。 関連付けられた記録 TrueLog と再生 TrueLog が比較ビューに表示されます。 **TrueLog のステップ スルー** ダイアログ ボックスも開きます。
3. 検索する対象を次の中から選択します。
   - [フォーム送信]
   - [API 呼び出し]
   - エラー
4. **次を検索** をクリックします。 記録 TrueLog と再生 TrueLog の対応するフォーム送信、API 呼び出し、またはエラーの最初のセットが表示されます。
5. コントロールの状態を比較して、相違があるかどうかを確認します。 記録と再生の間で変化した名前またはコントロール値は、 TrueLog Explorer で強調表示されます。 これによって、カスタマイズが必要な箇所や検証が役立つ箇所を特定しやすくなります。
6. (検出された差分に基づいて) 必要なカスタマイズを完了します。

   a) **スクリプトの試行** をクリックします。

   カスタマイズしたスクリプトがエラーなしで実行されるかどうかをテストします。
7. 必要なカスタマイズがすべて追加されるまで、この手順を必要な回数繰り返します。
8. **比較モード** をクリックして、比較モードを無効にします。

予期しない Get 呼び出し

サーバー リクエストの大部分は、予測可能であり、記録スクリプトにより正確に処理されます。 スクリプトの再生中に、スクリプトの記録中に送信しなかった追加の Get 呼び出しをサーバーが自発的に送信することがあります。 このような場合は、クライアントはデフォルトの回答のみをサーバーに送信できます。 サーバーがデフォルト回答を受け入れる場合は、エラーなしで再生が続行されます。 ただし、サーバーがデフォルトの回答を受け入れ不能とみなす場合は、手動で OraFormsSet 関数と OraFormsOnMessageGet 関数をスクリプト化し、サーバーの Get 呼び出しに対する正しいレスポンスを作成する必要があります。

たとえば、サーバーが UI コントロールの Get プロパティを問い合わせる VALUE リクエストを自発的に送る場合を考えます。 スクリプトにはこの Get リクエストを処理するスクリプト化関数がないので、クライアントは文字列のデフォルト値 (null) を返信します。 サーバーはこの回答を受け入れません。 クライアントへの接続が異常終了し、再生スクリプトでエラーが発生します。

このような状況に対応するために追加の Get 関数をスクリプトに手動で追加するには、クライアントとサーバー間で交換される受信データと送信データを解析することが必要です。 この解析を実行できるのは、Silk Performer のログ レベルに [デバッグ] を設定した場合のみです。

注: テストを実行する前に、ログ レベルに [デバッグ] を設定しなかった場合は、設定を変更してからテストを再実行し、必要なデバッグ データを生成する必要があります。

注: また、Oracle Forms サーバーでの record=names パラメータは、スクリプトの正確な記録を可能にするために有効にする必要があります。

スクリプト化した OraFormsOnMessageGet 関数では、関連する UI 要素に対する後続の Get 呼び出しに対して、デフォルトの回答とは異なる回答を用意する必要があります。

注: 実際に、スクリプトで回答を用意するのは、OraFormsSet 関数です (OraFormsOnMessageGet 関数呼び出しの前の行にある)。

次の BDL スクリプトの例で示すとおり、Get 呼び出しが処理されるとき、スクリプトを下から上に読む必要があります。

例

```
OraFormsSetRectangle("VISIBLERECT", 0, 0, 119, 24,
ORA_SET_TYPE_MESSAGEGET);
OraFormsOnMessageGet("LINE_CUSTOMER_ITEM_DSP_0"); // Requested
Item
OraFormsMouseClick("LINE_CUSTOMER_ITEM_DSP_0", 70, 18, 0); //
Requested  Item
```

このコード例は次のように解釈できます。クライアントの UI 要素 LINE_CUSTOMER_ITEM_DSP_0 上でマウスがクリックされます。 これは、OraFormsMouseClick 関数で表現されます。 マウスがクリックされると、サーバーから Get 呼び出しを受信することが予測されます。 マウスがクリックされる前に Get 呼び出しに対して準備するには、OraFormsOnMessageGet 関数を挿入します。 ここで、サーバーがプロパティ VISIBLERECT を要求することを予測し、rectangle 関数を使用して、OraFormsOnMessageGet タイプのプロパティ値を OraFormsSetRectangle 関数の前のスタックに配置します。

Silk Performer のログ レベルに [デバッグ] を設定する

予期しない get 呼び出しを受信した場合のスクリプトを手動で記述するには、クライアントとサーバー間で交換される受信データと送信データを解析することが必要です。 この解析を実行できるのは、Silk Performer のログ レベルに [デバッグ] を設定した場合のみです。

1. Silk Performer で、**設定** > **システム** を選択します。
2. **Oracle Forms** グループ ボタンをクリックします。
3. ログ レベル リスト ボックスから **[デバッグ]** を選択します。

Oracle Forms のユーザー入力データのカスタマイズ

Silk Performer でテスト スクリプトを生成してスクリプトの試行を行うと、パラメータ化された入力データを使用して TrueLog Explorer でスクリプトをカスタマイズできます。

ユーザー データのカスタマイズでは、記録された静的ユーザー入力データを、トランザクションごとに変更されるパラメータ化された動的ユーザー データに置き換えることによって、現実に近いテスト スクリプトを作成できます。 このようなデータ駆動型テストの作成では、手動でのスクリプト記述は必要ありません。

複数列データ ファイル

複数列データ ファイルを使ったパラメータ化は、データをパラメータ化する有効な方法です。 特定の文字列値の組み合わせを保存したファイルを定義します。 データ ファイルの各列はそれぞれのパラメータに対応します。 複数列データ ファイルを使うと、データ駆動型テスト モデルが利用できるようになり、すべてのユーザー データ入力をただ 1 つのデータ ファイルでカバーできます。

Oracle Forms のユーザー入力データをカスタマイズする

現実の条件の下では、Web アプリケーションのユーザーは、予測できないような組み合わせのデータをフォームに送信します。 効果的な Web アプリケーション テストの目標の 1 つは、テスト スクリプトによって、そのような変則的で多様なユーザーの動作をエミュレートすることです。

TrueLog Explorer の [パラメータ ウィザード] を使用すると、テスト中にフォームに入力されるユーザー入力データをカスタマイズできます。 [パラメータ ウィザード] によって、フォーム フィールドに入力されるデータを指定できます。また、記録されているユーザー入力データをランダムなパラメータ化されたユーザー データに置き換えることによって、現実に近いテスト スクリプトを作成できます。

1. TrueLog メニュー ツリーからユーザー データ入力を含むノードを選択します。
   背景が黄色のコントロールの入力値をカスタマイズできます。
2. コントロールの **値** 列を右クリックして、**値のカスタマイズ** を選択します。
   [パラメータ ウィザード] では、次のいずれかの方法でスクリプトの値を変更できます。
   - スクリプトの dclparam または dclrand のセクションで定義されている既存のパラメータを使用する
   - 新しい定数値、ランダム値、複数列データ ファイル内の値に基づいた、新しいパラメータを作成する
3. **パラメータを新規作成する** オプション ボタンをクリックします。
   a) **次へ** をクリックします。
   パラメータの新規作成 ダイアログ ボックスが開きます。
4. **ランダム変数からパラメータを作成** オプション ボタンをクリックします。
   a) **次へ** をクリックします。
   パラメータ ウィザード - ランダム変数 が開きます。
5. リスト ボックスから、テスト スクリプトに挿入するランダム変数の種類を選択します。
   強調表示された変数タイプの簡単な説明が下部のペインに表示されます。
   a) **次へ** をクリックします。
   変数の名前と属性の指定 ページが表示されます。 Strings from file というランダム変数タイプでは、指定されたファイルからランダムに選択または順次選択できるデータ文字列が生成されます。
6. 変数の名前を **名前** フィールドに入力します。
   a) 変数が呼び出される順序として次のいずれかを選択します。
   - [ランダム]
   - [シーケンシャル]
7. **ファイル** 領域の 名前 リスト ボックスから、構成済みのデータソースを選択します。
   代替方法:**新規作成** をクリックすると、ランダム変数ファイルを新規作成できます。
8. **テストごと** というランダム変数生成を選択します。
   a) **終了** をクリックします。
   これにより、記録されている値の代わりにランダム変数を指定のフォーム フィールドに使用するようにテスト スクリプトの BDL フォーム宣言が変更されます。 新しいランダム変数関数が [BDL] ビューの下部に表示されます。

テスト スクリプトにランダム変数関数を追加した状態でスクリプトの試行を起動し、スクリプトがエラーなしに動作することを確かめます。

カスタマイズできる入力データ関数

TrueLog Explorer には、記録中にユーザーによって変更されたコントロールの入力値をカスタマイズできるウィザードが用意されています。 入力値は、記録中に最初に入力が行われたのと同じ位置でカスタマイズできます。 次に、カスタマイズできる関数の一覧を示します。

関数	説明
OraFormsEditSet	テキスト コントロールの値をカスタマイズします。
OraFormsRadioSet	ラジオ ボタンを選択するかどうかを指定します。
OraFormsCheckboxSet	チェック ボックスをオンにするかどうかを指定します。
OraFormsListSelect	選択するリスト ボックスの要素を指定します。
OraFormsPopListSelect	選択するポップアップ リスト ボックスの要素を指定します。
OraFormsLogon	[ログオン] ダイアログのログオン資格情報をカスタマイズします。
OraFormsLovFind	[値リスト] ダイアログの検索パターンをカスタマイズします。
OraFormsLovSelect	[値リスト] ダイアログの選択項目をカスタマイズします。
OraFormsEditorDialogOK	[エディタ] ダイアログのテキスト コントロール値をカスタマイズします。

Oracle Forms 用のコンテンツ検証関数

TrueLog Explorer には、Oracle Forms アプリケーション用のあらかじめ有効になっている検証関数があります。 検証するオブジェクトを右クリックして、検証関数を選択します。 それだけで、指定した検証関数が生成されて、BDL スクリプトに自動的に挿入されます。

TrueLog Explorer では、コンテンツ検証関数をテスト スクリプトに挿入することで、テスト中にアプリケーション サーバーから返されたコンテンツが正確かどうかを検証することができます。

TrueLog Explorer では、再生テスト実行と記録テスト実行を比較すること (クライアント/サーバー環境でエンド ユーザーが体験する事柄をテストするという難問に対する、類を見ない効果的なアプローチ) によって、埋め込みオブジェクトや、テキスト、グラフィックス、テーブル データ、SQL レスポンスなどが、実際にクライアントにダウンロードされて表示されるかを視覚的に確認できます (システムの負荷が高い場合)。 これにより、他の Web トラフィック シミュレーション ツールが検知できない種類のエラーを検知できます。つまり、標準的な負荷テスト スクリプトでは検出されない負荷がかかっている場合にのみ発生するエラーを検出できるようになります。

コンテンツ検証は、継続的なパフォーマンス管理に利用できるので、システムを配置した後にも役立ちます。

Oracle Forms 検証関数を挿入する

Oracle Forms のスクリプトの試行をテスト実行します。 TrueLog Explorer で、作成された TrueLog を開きます。

1. TrueLog Explorer で Oracle Forms の TrueLog を開いて、検証可能なデータのある TrueLog ノードを選択します。
2. 検証可能な値を右クリックして、**値の検証** を選択します。 **フォーム コントロールの検証の追加** ダイアログ ボックスが開きます。

3. **現在選択されているフォーム コントロールの値を検証** リンクをクリックします。 **値検証関数の挿入** ダイアログ ボックスが表示されます。
4. ドロップダウン リストから次の値のいずれかを選択します。
   - 等しい
   - 異なる
   - 含む
   - 含まない
5. 検証での大文字/小文字の区別、および空白文字の詳細について指定します。
6. 検証が否定的な結果を返した場合に発生させる深刻度を指定します。
7. パラメータに対して検証を実行するかどうかを指定します。
8. **値検証関数の挿入** ダイアログ ボックスで **OK** をクリックします。 検証関数がテスト スクリプトに追加されます。
9. **検証の追加** ダイアログ ボックスで **Yes** をクリックして、スクリプトの試行を行います。
10. 正常に検証されたことを確認します。
    検証を含む API ノードには、青い "V" 印が表示されます。

Oracle Forms スクリプトへのカスタマイズの適用

以下のステップを完了すれば、Oracle Forms テスト スクリプトはエラーなしで動作します。

- スクリプトがセッション情報およびユーザー入力データを処理する方法をカスタマイズします。
- 必要な検証関数を追加します。
- 必要であれば、Silk Performer での BDL スクリプトを手動で編集します。

Citrix アプリケーションでの作業

Citrix サーバーをホストとするアプリケーションのテストに対する TrueLog 記録、TrueLog 再生、および基本的な TrueLog 分析のほかに、TrueLog Explorer では、ユーザー入力データのカスタマイズも提供されています。 ここで説明する機能は、Citrix NFuse セッションのテストにも適用できます。

この章を読み進む前に、TrueLog Explorer の基本機能を熟知していることが不可欠です。

注: Citrix サーバーをホストとするアプリケーションをテストする際の Silk Performer の使用方法の詳細については、『*Citrix Tutorial*』を参照してください。

Silk Performer Citrix Player

Silk Performer は、再生には TrueLog Explorer ではなく独自の Citrix Player を使用します。 Silk Performer Citrix Player は、スクリプトの試行が開始されたときに開き、記録されたすべてのアクションを完全なアニメーションで再生します。 マウスの移動や操作は、アニメーションのマウス アイコンでシミュレートされます。

Silk Performer Citrix Player には **ログ** ウィンドウがあり、スクリプトの試行のさまざまな側面を詳細に示す以下の 3 つのペインがあります。

[スクリプト] このペインには、実行されたすべての BDL スクリプト関数および現在実行中の BDL 関数が一覧表示されます。

[ウィンドウ] このペインには、現在のセッション中にアクセスされたすべてのクライアント ウィンドウのスタックが表示されます。ウィンドウのキャプション、スタイル、サイズ、および位置も表示されます。 最上位のウィンドウは、ウィンドウ アイコンで示されており、サブウィンドウの上部に表示されます。

[ログ] このペインには、実行された BDL 関数やウィンドウの作成/アクティブ化/破棄を含む、すべての情報メッセージとイベントが一覧表示されます。

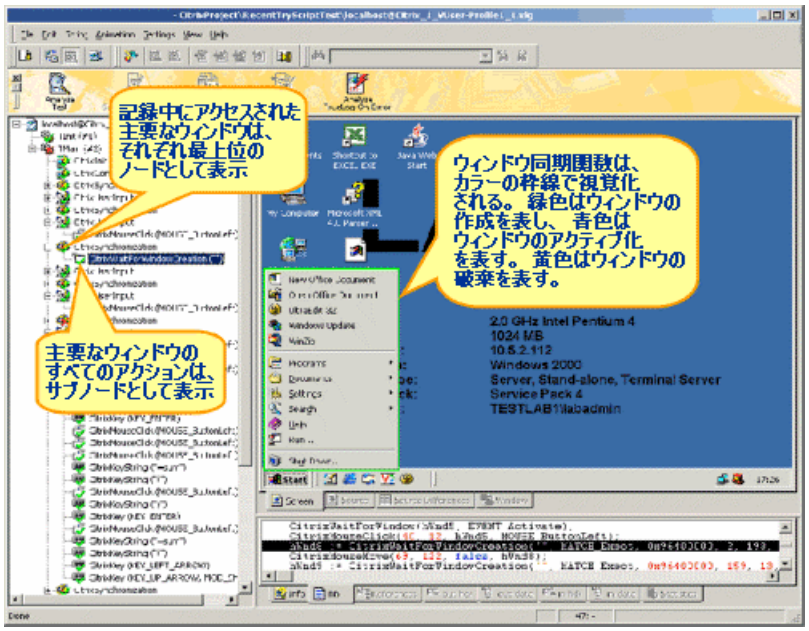
Citrix TrueLog

TrueLog Explorer は、オプションで、スクリプトの試行の実行中に Citrix Player と一緒に開くことができます (それには、**スクリプトの試行** ダイアログ ボックスで **TrueLog Explorer によるアニメーション実行** チェック ボックスをオンにします)。 TrueLog Explorer は、スクリプトの試行の実行中に実際にダウンロードされたデータを表示します。 記録中にアクセスされた主要なウィンドウは、TrueLog Explorer のメニュー ツリーの最上位 API ノードとしてそれぞれ表示されます。 それらのウィンドウに対して記録されたアクションは、サブノードとして表示されます。

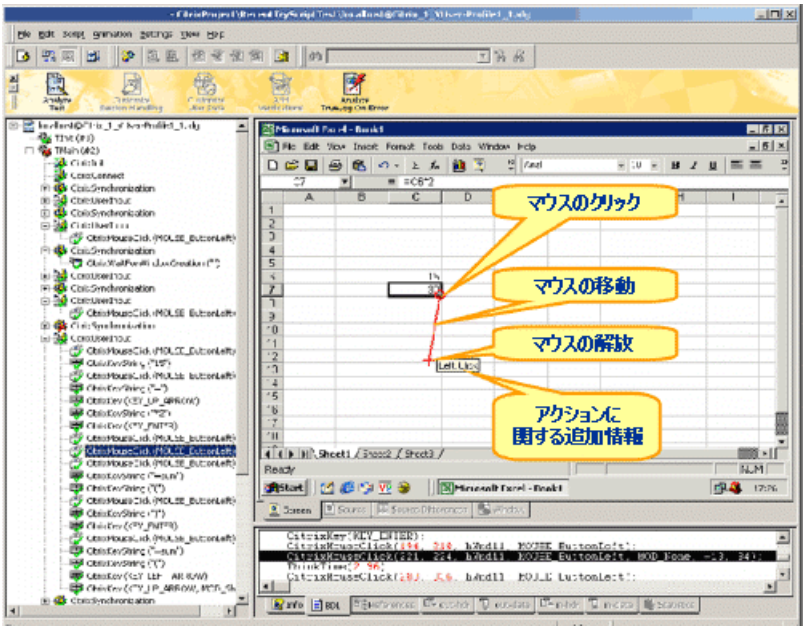
上位の同期ノードを選択すると、最後の同期関数の後に表示されたウィンドウ(再生中にキャプチャされたビットマップ)が表示されます。

ウィンドウ同期関数は、カラーの枠線で視覚化されます。 ウィンドウの作成は、緑色の枠線で示され、 ウィンドウのアクティブ化は、青色の枠線で示されます。 ウィンドウの破棄は、黄色の枠線で示されます。

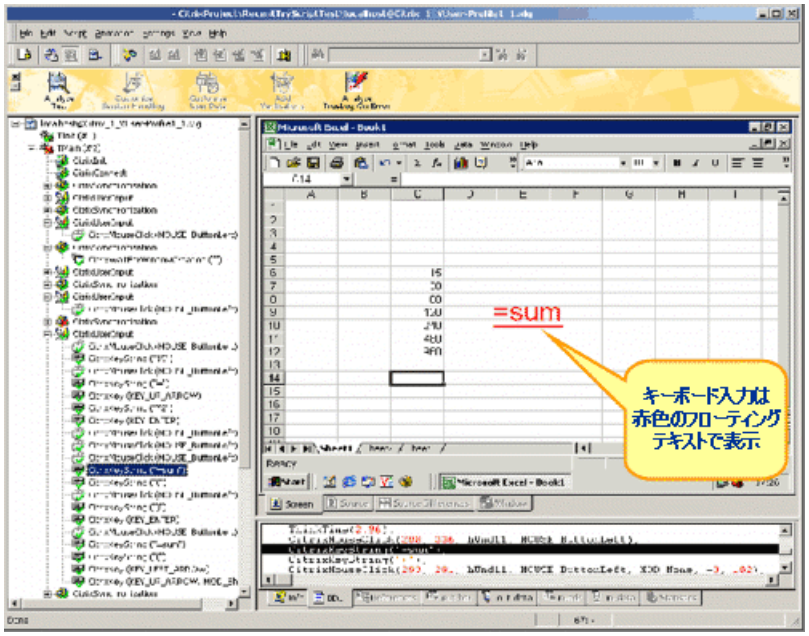
TrueLog は、画面の状態を視覚化することによって、Citrix Player を補完します。 たとえば、Citrix Player の **ログ** ウィンドウに表示されている特定のウィンドウ ID で示されるウィンドウが不明な場合、それに対応する TrueLog 内の対応する同期関数を見つることによって、問題のウィンドウを示すビットマップにアクセスできます。



ユーザー入力ノード (CitrixUserInput とそれに関連する関数) には、キーボード入力とマウス入力が反映されます。 CitrixMouseClick 関数には、2 つの追跡ベクトル パラメータ (X 座標と Y 座標) があります。 赤色の四角は、マウスをクリックした始点を示し、 赤色の十字は、マウスを解放した点を示します。 始点と終点の間の赤色の線は、マウスの経路を示します。 ボタンが押されている間に移動がなかった場合は、赤色の十字だけが表示されます。 オンスクリーン ツールチップには、補足情報(右クリック、左クリック、ダブルクリックなど)が表示されます。



値の文字列 (キーボード入力) は、対象のウィンドウ キャプションが (後続のノードで) 特定されるまで、赤色のフローティング テキストとして画面に視覚化され、文字列が入力される位置を示します。



Citrix スクリプトでの同期の問題

Citrix Player での再生中に、記録中にキャプチャされた値と異なる値が検出されると、ウィンドウがアクティブにならなかったり、画面の同期に失敗することがあります。 同期に関する問題の原因は、常に明白とは限りません。画面の位置が 1 ピクセルだけ変化したことが原因の場合もあります。 このような違いは、ビットマップ表示プログラムを使用して、視覚的に評価することが最良です。

画面の同期の失敗よりもよくあるのは、再生中にウィンドウがアクティブにならないことです。 このような場合は、それ対応するユーザーアクションに関連するスクリーンショットが失敗の手がかりになることがあります。 ユーザー エラーがないのに、ウィンドウがたまにしかアクティブにならないこともあります。 このような場合は、関連する CitrixWaitForWindow 関数を削除する必要があります。

TrueLog Explorer は、エラーが発生するとスクリーンショットをキャプチャし(デフォルトの設定)、そのビットマップをディスクに書き込みます。 TrueLog On Error のビットマップと、Silk Performer によって同期関数と一緒にキャプチャされたビットマップを比較することによって(デフォルトの設定)、記録画面と再生画面を視覚的に比較できます。 デフォルトでは、スクリーンショットはレコーダによってプロジェクト ディレクトリ内のスクリーンショット ディレクトリに書き込まれます。 再生では、現在の結果ディレクトリにスクリーンショットが保存されます。

注: Silk Performer の **画面の同期に失敗したらウィンドウ領域をダンプする** Citrix オプションをアクティブにする必要があります(デフォルト)。

Citrix のユーザー入力データのカスタマイズ

ユーザー入力データのカスタマイズを利用すると、記録された静的ユーザー入力データを、トランザクションごとに変更されるパラメータ化された動的ユーザー データに置き換えることによって、現実に近いテスト スクリプトを作成できます。 このようなデータ駆動型のテストでは、手動によるスクリプト作成は必要ありません。

Citrix ターミナル サービスをホストとするアプリケーションに入力されるユーザー入力は、テスト中に以下の 2 つの方法でカスタマイズできます。

- **パラメータ ウィザード** によって、キーボード イベントで入力される値を指定できます。また、記録されたユーザー入力データをランダムなパラメータ化されたユーザー データに置き換えることによって、現実に近いテスト スクリプトを作成できます。
- 視覚的なカスタマイズによって、記録中にキャプチャされたレンダリング画面を右クリックすることによって、クリック、ドラッグ、リリースなどのマウス イベントをカスタマイズできます。

Citrix のユーザー入力データのカスタマイズ

記録された静的な入力データを、パラメータ化された動的なデータて置き換えることによって、テストスクリプトを現実に近づけるには、**パラメータ ウィザード**を使用します。

1. メニュー ツリーで、ユーザー データ入力を表すノードを選択します(たとえば、キーボードのデータ文字列を表す CitrixKeyString ノードを選択します)。
2. 入力データ文字列(赤色のフローティング文字列)を右クリックし、**ユーザー入力のカスタマイズ** を選択します。
3. **パラメータ ウィザード** が開きます。 **パラメータを新規作成する** を選択し、**次へ** をクリックします。
   [パラメータ ウィザード] では、次のいずれかの方法でスクリプトの値を変更できます。
   - スクリプトの dclparam または dclrand のセクションで定義されている既存のパラメータを使用する
   - 新しい定数値、ランダム値、複数列データ ファイル内の値に基づいた、新しいパラメータを作成する
4. **パラメータの新規作成** ダイアログ ボックスが開きます。 **ランダム変数からパラメータを作成** オプション ボタンを選択し、**次へ** をクリックします。
5. **使用法の選択** ダイアログ ボックスが開きます。 新しいランダム変数を [毎回]、[トランザクションごと]、[テストごと] のどのタイミングで使用するかを指定します。
6. **完了** をクリックして、記録された値の代わりに、指定のフォーム フィールドにランダム変数を使用するように、テスト スクリプトの BDL フォーム宣言を変更します。 新しいランダム変数関数が **BDL** ビューの下部に表示されます。
7. テスト スクリプトにランダム変数関数を追加した状態でスクリプトの試行を起動し、スクリプトがエラーなしに動作することを確かめます。

マウス イベントをカスタマイズする

記録されたマウス イベントの動作は、視覚的にカスタマイズできます。

1. メニュー ツリーで、マウス アクティビティを含む CitrixMouseClick ノードを選択します。 赤色の四角は、マウスをクリックした始点を示し、 赤色の十字は、マウスを解放した点を示します。 始点と終点の間の赤色の線は、マウスの経路を示します。 オンスクリーン ツールチップには、補足情報(右クリック、左クリック、ダブルクリックなど)が表示されます。
2. 画面上の任意の場所を右クリックし、**ユーザー入力のカスタマイズ** を選択します。 **マウス イベントのカスタマイズ** ダイアログ ボックスが開きます。
3. カスタマイズしたマウス移動の始点にしたい画面上の位置をクリックします。
4. **カスタマイズ** をクリックしてカスタマイズを受け入れ、BDL スクリプトを変更します。

マウス イベントのカスタマイズ結果が、記録 TrueLog のビットマップに緑色で示されます。 マウスのカスタマイズ結果は、BDL スクリプトでも緑色のテキストで示されます。 CitrixMouseClick 関数には、2 つの追跡ベクトル パラメータ(X 座標と Y 座標)があります。 このスクリプトが次回実行されたとき、ここで指定した新しい画面座標が使用されます。

Citrix の解析関数と検証関数

Silk Performer の光学文字認識(OCR)サポートを利用すると、キャプチャされたアプリケーション状態のスクリーンショット内のテキスト値が認識されるため、セッションに依存する検証や解析が容易になります。 ほかの TrueLog 形式(Web、データベースなど)の場合と同様に、検証関数や解析関数は、スクリプトの記録後に TrueLog Explorer を使用して追加されます。

注: TrueLog Explorer で、解析関数や検証関数用の OCR を有効にするには、Silk Performer のシステム設定を使用して、事前にフォント データベースを生成する必要があります。

ウィンドウの位置と状態

ウィンドウの位置と状態(最大化/最小化)は、正確な再生を保証するために非常に重要です。TrueLog Explorer では、選択されているテキストが、個々のウィンドウではなくデスクトップに対する相対座標で読み取られるように画面座標がスクリプト記述されるからです。 したがって、再生時に、記録時と異なる位置にウィンドウが表示されると、指定したテキストを OCR 操作で見つけることはできません。 変換領域の絶対位置を指定することができない場合は、ウィンドウに対する相対座標を使用して、手動でスクリプトを更新する必要があります。

検証関数

Silk Performer では、Citrix に対する入力値検証とレスポンス データ検証はサポートされていませんが、Citrix サーバーをホストとするアプリケーションのビットマップとウィンドウの検証はサポートされています。

OCR 検証関数を挿入する

Citrix サーバーをホストとするアプリケーションのビットマップとウィンドウの検証は、セッションに依存するデータ (ログイン名など) の検証には対応していません。 ただし、サード パーティの DLL で提供される OCR サポートを利用すると、Silk Performer は認識可能なテキスト値を変数に格納できるようになり、Citrix 負荷テストでのセッション依存の検証が容易になります。

1. Silk Performer を使用して、Citrix セッションを記録します。
2. **スクリプトの試行** ダイアログ ボックスの **TrueLog Explorer によるアニメーション実行** チェック ボックスをオンにしてスクリプトの試行を実行します。 TrueLog Explorer が開きます。
3. スクリプトの試行が完了したら、テキスト検証の対象となる画面のビットマップ画面キャプチャを含む API ノードを選択します。
4. 画面上でクリックしてカーソルをドラッグし、検証に使用するテキストが含まれている画面領域を選択します。
5. 選択した領域を右クリックして、**テキストの検証** を選択します。
6. **テキスト検証関数の挿入** ダイアログ ボックスが開きます。 選択されたテキストが **定数値** テキスト ボックスにプリロードされ、**定数値** オプション ボタンがデフォルトで選択されます。

ヒント: 定数値の検証に加えて、パラメータ (既存のパラメータまたは新しいパラメータのいずれか) を検証することもできます。 パラメータを検証するには、**パラメータ値** オプション ボタンを選択します。 パラメータが既に存在する場合は、参照 (**...**) ボタンをクリックすると、パラメータを選択できます。 パラメータが存在しない場合は、参照 (**...**) ボタンをクリックすると **パラメータ ウィザード** が起動され、新規パラメータを作成できます。

7. **選択した四角形内のテキストが以下に** リスト ボックスから [等しい] または [等しくない] を選択します。
8. 検証が、[大文字/小文字を区別する] かどうか、[空白を無視する] かどうかを指定します。
9. ダイアログ ボックスの **深刻度** エリアで、検証が否定的な結果を返した場合に発生させる深刻度を指定します([エラー]、[警告]、[情報]、または [カスタム])。 **OK** をクリックします。
10. 確認のためのダイアログ ボックスが開きます。 **OK** をクリックして、Citrix テスト スクリプトに OCR 検証関数を追加します。

OCR 解析関数を挿入する

1. Silk Performer を使用して、Citrix セッションを記録します。
2. **スクリプトの試行** ダイアログ ボックスの **TrueLog Explorer によるアニメーション実行** チェック ボックスをオンにしてスクリプトの試行を実行します。 TrueLog Explorer が開きます。
3. スクリプトの試行が完了したら、テキスト解析の対象となる画面のビットマップ画面キャプチャを含む API ノードを選択します。
4. 画面上をクリックしてカーソルをドラッグし、解析したいテキストを含む画面領域を選択します。
5. 選択した領域を右クリックして、**テキストの解析** を選択します。
6. **解析関数の挿入** ダイアログ ボックスには、解析関数を構成するためのパラメータがあります。 デフォルトの設定で適切な場合がほとんどですが、以下の設定を調整できます。

   パラメータ名 解析関数の結果を受け取るパラメータの名前を入力します。

   スクリプトに出力文を挿入
   - Web ページ呼び出しの後に、通知のための Print 文をスクリプトに挿入するには、**Print 文** をオンにします。 これによって、解析関数の結果が Silk Performer の **仮想ユーザー** 出力ウィンドウに出力されます。
   - (Print 文によって **仮想ユーザー出力** ウィンドウに値を書き込むだけでなく) デバッグに役立つように解析結果を出力ファイルに書き込むには、**Writeln 文** ("write line" 文) をオンにします。 出力ファイルの生成によって時間測定値が変化するので、これらのファイルはデバッグのためだけに使用し、完全な負荷テストでは生成しないでください。

7. **OK** をクリックします。
8. 確認のためのダイアログ ボックスが開きます。 **OK** をクリックして、Citrix スクリプトに OCR 解析関数を追加します。

OCR の検証と解析

このセクションでは、OCR によって生成される文字列を解析および検証する方法について説明します。

OCR による検証と解析の動作

OCR による文字列検証は、CitrixVerifyText API 呼び出しを使用して実現されます。 これらの関数は、スクリプトのカスタマイズ時に TrueLog Explorer によって挿入されます。 CitrixVerifyText 関数は、再生ビットマップ内のテキスト文字列を比較して、それらが同じかどうかを判別します。

CitrixParseText 関数は、テキストを解析するために使用できます。 これらの API 呼び出しは、その他の解析関数(Web、データベースなど)と同様に動作します。

OCR では、さまざまなフォントやテキスト スタイルを認識するためにパターン データベースが使用されます。 フォント データベースは、OCR を実行する前に生成しておく必要があります。

Citrix TrueLog でのみ、検証および解析の API 呼び出しがメニュー ツリーに表示されます。 その他の TrueLog モード(Web、データベースなど)では、新しい API ノードはメニュー ツリーに追加されません。

注: OCR 操作は静的なコンテンツに対して実行する必要があります。 ウィンドウ イベントでの同期を行うと、画面の更新がわずかに遅れることがあります。これによって、タイミングに依存した結果が生じます。 画面があまりに頻繁に変更される場合は、タイミングがずれて、比較のための正しい画像が選択されないことがあります。 したがって、すべての OCR 検証関数や解析関数の前に、CitrixWaitForScreen 関数の呼び出しをするようにスクリプトを記述する必要があります。

OCR を使用するために Silk Performer を構成する

Citrix の解析関数と検証関数用に OCR を有効にするには、Silk Performer のシステム設定を使用して、フォント データベースを生成する必要があります。

OCR は、ビットマップ内のフォントやテキスト スタイルを認識するために、フォント データベース ("パターン" データベース) を使用します。 デフォルトのフォント セットでほとんどのシナリオはカバーできますが、状況によっては、ほかのフォントやフォント スタイルの追加が必要になることがあります。 新しいフォントを追加したり、システムからフォントを削除したときは、毎回、新しいデータベースを生成する必要があります。

データベース内のフォントが多すぎると、処理が遅くなったり、読み取り間違いをする場合があります。このため、テキスト文字列の取得対象となるビットマップで使用されるフォントのみを含めることをお勧めします。

1. Silk Performer で、**設定** > **システム...** を選択し、**Citrix** アイコンを選択します。
2. **OCR** ページで、**追加 >>** または **すべて追加** を使用して、OCR で使用するフォントを **システム フォント** リスト ボックスから **選択済みのフォント** リスト ボックスに移動します。
3. **選択済みのフォント** リスト ボックスから不要なフォントを削除するには、**すべて削除** または **<< 削除** を使用します。
4. **サイズ** テキスト ボックスで、使用するフォント サイズの範囲を指定します(例、[8-20])。
5. **イタリック**、**太字**、および **下線** の各チェック ボックスをオンにして、含めるフォント スタイルを定義します。
6. **フォント データベースの生成** をクリックします。
7. **フォント ベースの構築** ダイアログ ボックスが開きます。 **OK** をクリックして、既存のフォント データベースを新しいデータベースで置き換えることを確認します。
8. **システム設定** ダイアログ ボックスで、**OK** をクリックして、変更を受け入れます。

テキスト同期関数の生成

TrueLog Explorer には、指定したテキストやテキスト パターンが、指定した画面位置に表示されるまで、スクリプトの実行を一時停止する CitrixWaitForText 同期関数が用意されています。 この機能を使用して、画面とユーザー入力を同期させることができます。 たとえば、**OK** を押す前に、指定したダイアログ テキストに "処理中" ではなく "準備完了" と表示されるまで、スクリプトの実行を一時停止する必要があるとします。 処理にかかる時間が不明な場合、このようなアクションを同期させるのは困難です。 この関数のスクリプトでは CitrixWaitForText(... "ready", ...) 関数の前に、CitrixMouseClick(...) が読み取られます。

1. Silk Performer を使用して、Citrix セッションを記録します。
2. **スクリプトの試行** ダイアログ ボックスの **TrueLog Explorer によるアニメーション実行** チェック ボックスをオンにしてスクリプトの試行を実行します。 TrueLog Explorer が起動します。
3. スクリプトの試行が完了したら、特定のテキスト文字列を待つ画面のビットマップ画面キャプチャを含む API ノードを選択します。
4. 画面上をクリックしてカーソルをドラッグし、同期させたいテキストを含む画面領域を選択します。

5. 選択した領域を右クリックして、**テキストの同期** を選択します。
6. **テキスト同期関数の挿入** ダイアログ ボックスが開きます。 スクリプトが待機する時間を指定する定数値を入力するか、パラメータ値を選択します。 **OK** をクリックして、この関数をスクリプトに挿入します。

TCP/IP および UDP ベースのアプリケーションでの作業

TrueLog On Error とテスト スクリプトの分析機能を TCP/IP および UDP プロトコル ベースのアプリケーションに適用する方法について説明します。

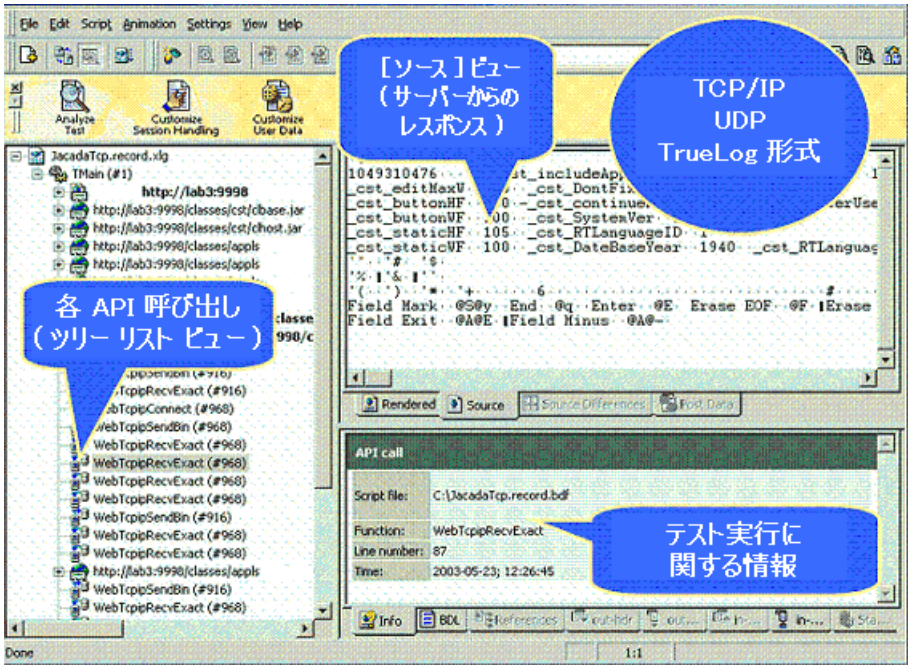
TrueLog Explorer は、TCP/IP および UDP ベースの TrueLog (たとえば、SMTP、POP3、カスタム プロトコルなど) に対する効果的なビューアとしての役割を果たします。 TrueLog Explorer を利用すると、再生 TrueLog と記録 TrueLog を比較できます。 スクリプトのカスタマイズ(セッションの処理、ユーザー入力データのパラメータ化、および検証機能)は TCP/IP と UDP の TrueLog では利用できません。

この章を読み進む前に、TrueLog Explorer の基本機能を熟知していることが不可欠です。

TCP/IP と UDP の TrueLog の構造

TCP/IP と UDP の TrueLog は、HTML TrueLog と同様に構成され、表示されます。 1 つだけ異なるのは、TCP/IP と UDP には HTML のような高度なレンダリング インターフェイスがない点です。

TCP/IP と UDP の TrueLog に対しては、Web API 呼び出しが含まれている場合以外は、**レンダリング** ビューは利用できません。



[コンテンツ] ペイン

TCP/IP および UDP ベースの TrueLog では、TrueLog Explorer の **コンテンツ** ペインに次のデータ ビューがあります。

項目	説明
レンダリング	Web API 呼び出しが含まれている場合以外は、TCP/IP と UDP では利用できません。

項目	説明
ソース	サーバーからアプリケーションに送信されたレスポンスを含みます (WebTcpipRecv 関数など)。
ソース差分	**ソース差分** ページには、選択したノード間の受信データ (ソース) の差分が一覧表示されます。 このページは、サーバー レスポンス内の動的情報を検索するのに役立ちます。セッション情報を探すには、まずここを検索します。 **注: ソース差分** ページが利用可能になるのは 差分モードのときのみです。
ポスト データ	TCP/IP と UDP では利用できません。

情報ペイン

情報 ペインには、TCP/IP と UDP のテスト スクリプトとテスト実行に関連するデータが表示されます。このペインには次のビューがあります。

項目	説明
情報	**情報** ページには、開いている TrueLog ファイルと選択した API ノードについて、以下のような一般情報が表示されます。 • スクリプト ファイル名 • 関数 • 行番号 • 時間 • 実行時間 • 絶対 URL • 完了ステータス • 追加情報 (可能な場合)
BDL	**BDL** ページには、開いている TrueLog に対応する BDL スクリプトが表示されます。 この BDL スクリプトは、選択されている API ノードの行まで自動的に移動して表示されます。
参照	TCP/IP と UDP では利用できません。
送信ヘッダー	TCP/IP と UDP では利用できません。
受信ヘッダー	TCP/IP と UDP では利用できません。
送信データ	TCP/IP と UDP の場合は、アプリケーションからサーバーへ WebTcpipSend 関数で送られるデータが表示されます。
受信データ	サーバーからアプリケーションに送信されたレスポンスを含みます (WebTcpipRecv 関数など)。
統計	TCP/IP と UDP では利用できません。

ASCII 表示オプションと 16 進表示オプションを設定する

自動表示モード を利用すると、最適なデータ表示モードが自動的に選択されますが(たとえば、画像は バイナリ にし、HTML ドキュメントは テキスト にする)、TrueLog Explorer には、TCP/IP 関数と UDP 関数の表示を ASCII 形式と 16 進形式の間で切り替えるオプションがあります。

1. **設定** > **オプション** > **表示** を選択します。
2. ダイアログの **データ形式** の **自動表示モード** チェック ボックスをオフにします。
3. **バイナリ データを表示** チェック ボックスをオンにします。
4. **OK** をクリックします。

TCP/IP と UDP の再生 TrueLog と記録 TrueLog を比較する

TCP/IP と UDP の再生 TrueLog をそれに対応する記録 TrueLog と比較して、プロトコルの動的要素を明らかにする場合は、後のテスト実行のためにこれらの動的要素をカスタマイズする必要があります。さもないと、テスト中にセッション関連のデータが渡されてエラーが発生します。 これらのプロトコルの動的要素をカスタマイズするために、TrueLog Explorer では TCP/IP と UDP の記録ルールが提供されています。

注: 自動セッション処理のカスタマイズは、TCP/IP および UDP ベースのアプリケーションでは利用できません。 動的 BDL 構文解析機能は Silk Performer を利用して手動でコーディングする必要があります。

1. 比較モードで、TCP/IP または UDP の再生 TrueLog とそれに対応する記録 TrueLog を開きます。
2. **ソース** タブをクリックします。

   注: バイナリの場合には差分テーブルは有効になりません。したがって **ソース差分** ビューは利用できません。
3. **TrueLog のステップ スルー** をクリックします。 **TrueLog のステップ スルー** ダイアログ ボックスが開きます。
4. **API 呼び出し** オプション ボタンをクリックします。 記録レスポンスを含む TrueLog API 呼び出しが、それに対応する再生レスポンスと並んで表示されます。 動的要素がテスト スクリプトに含まれていることを示すこのような差分は、Silk Performer の記録ルールによってカスタマイズする必要があります。

ターミナル エミュレーション アプリケーションでの作業

このセクションでは、スクリプトの試行の結果に基づいてターミナル エミュレーション テスト スクリプトをカスタマイズする方法について説明します。

ターミナル エミュレーション アプリケーションでの作業 - 概要

ターミナル エミュレーション アプリケーションの場合、TrueLog Explorer では、TrueLog の記録、TrueLog の再生、TrueLog の基本分析のほか、解析、コンテンツ検証、および入力パラメータのカスタマイズも可能です。 TrueLog 機能は、VT100+、IBM 3270、および IBM 5250 の各プロトコルに対してサポートされています。

この章を読み進む前に、TrueLog Explorer の基本機能を熟知していることが不可欠です。

ユーザー データのカスタマイズ

TrueLog Explorer によるユーザー データのカスタマイズと [パラメータ ウィザード] は、TN3270 と TN5250 の両方のプロトコルのターミナル エミュレーション アプリケーションでサポートされています。

画面上のイベントの同期

記録/再生シナリオで画面上のイベントの同期に使用できる関数はありますが、TrueLog Explorer を通じてそれらの関数を挿入することはできません。

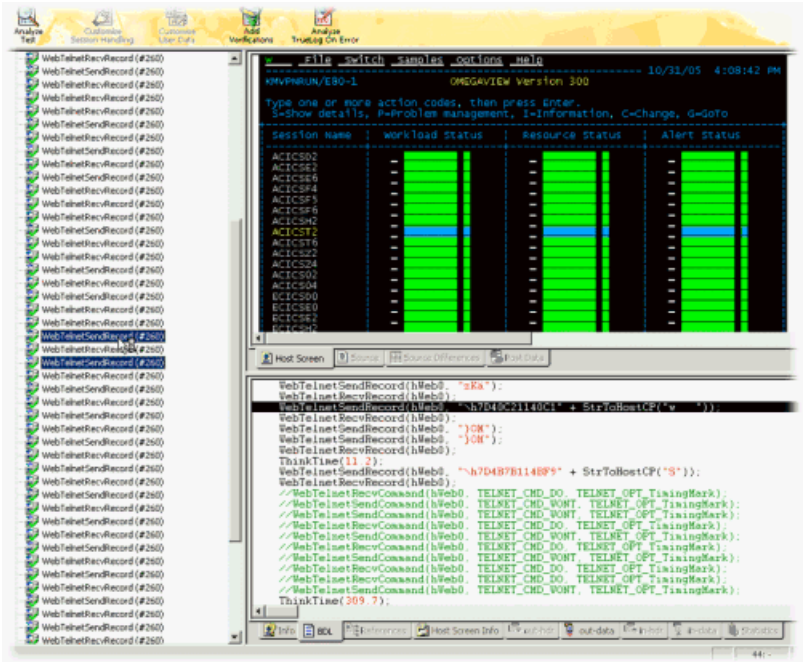
注: ターミナル エミュレーション アプリケーションに対する Silk Performer のサポートの詳細については、Miscellaneous Tutorials (PDF) を参照してください。 このファイルは Silk Performer の **ドキュメント** > **チュートリアル** にあります。

ターミナル エミュレーション TrueLog の構造

TrueLog Explorer では、HTTP クライアント リクエストと HTTP/HTML サーバー レスポンスの視覚化の場合と同じように、ターミナル エミュレーションのリクエストとレスポンスの視覚化をサポートしています。

ターミナル エミュレーション TrueLog で表示される 3 種類のウィンドウを以下に示します。

- **TrueLog メニュー ツリー** - テスト実行に含まれていたすべてのターミナル エミュレーション API 呼び出しが一覧表示されます。
- **[コンテンツ] ペイン** - 各 API ノードでのターミナル エミュレーション アプリケーションの状態が表示されます。
- **[情報] ペイン** - 最新のテスト実行に関するデータが表示されます。 このペインで、アクティブかつターミナル エミュレーション TrueLog に適用されるビュー タブは、**情報**、**BDL**、**受信データ**、**送信データ**、および **ホスト画面情報** です。



ホスト画面表示のカスタマイズ

([コンテンツ] ペイン内の) **ホスト画面** ビューに表示されるテキストのフォントやフォント サイズを変更できます。 テスト対象のアプリケーションで受信された文字が、定義済みのフォント (lucida console) で正しく表示されない場合は、この変更が必要な可能性があります。

フォント タイプ :

フォントを変更するには、Windows レジストリで以下のキーを変更する必要があります。

```
HKEY_CURRENT_USER¥Software¥Silk¥Silk  Performer¥9.5¥SystemSettings¥Layout
¥GreenScreen  Font  Name
```

フォント サイズ :

表示される文字のサイズを変更するには、Windows レジストリで以下のキーを変更する必要があります。

```
HKEY_CURRENT_USER¥Software¥Silk¥Silk  Performer¥9.5¥SystemSettings¥Layout
¥GreenScreen  Font  Size
```

フォントのサイズを入力します (デフォルトは *10*)。

ターミナル エミュレーション TrueLog のステップ スルーを行う

テストが終了すると、通常、TrueLog Explorer に複数の TrueLog On Error ファイルが表示されます (エラーを返した仮想ユーザーごとに 1 つの TrueLog)。 TrueLog Explorer の [エラーを検索] 機能を利用すると、エラーがどの TrueLog に記録されているかにかかわらず、エラーが発生した順番に、1 つずつ エラーをたどることができます。 これによりエラーの分析作業を単純化できます。 開いている TrueLog をすべて手動で確認して、次のエラーを検索する必要はありません。

1. テストが終了したら、Silk Performer で **結果の検討** をクリックします。

   注: TrueLog On Error ファイルが作成されるのは、Silk Performer の **エラー時に TrueLog を生成する** オプションが有効になっている場合だけです。

   ワークフロー - 結果の検討 ダイアログ ボックスが開きます。
2. TrueLog Explorer をクリックします。

   注: TrueLog Explorer は、テスト時にエラーが検出されなかった場合や [エラー時に TrueLog を生成する] が有効でない場合は、無効になっています。 そのような場合は、直接 Performance Explorer による結果の分析に進みます。

   TrueLog Explorer が起動され、現在のテストに対して生成された TrueLog On Error ファイルが開いて、**TrueLog のステップ スルー** ダイアログ ボックスが開きます。
3. **TrueLog のステップ スルー** ダイアログで該当するオプション ボタンをクリックすると、すべての [安定な画面] (WebTelnetRecvCommand 呼び出し)、[API 呼び出し] (各 API ノード)、または [エラー] をたどることができます。

   TrueLog On Error ファイルは、記録された順に検索されます。

   注: 複数の WebTelnetRecvCommand 呼び出しが 1 つのシーケンスにまとめられている場合、**安定な画面** オプションを選択して 次を検索 をクリックすると、そのシーケンスの最後にある WebTelnetRecvCommand 呼び出しにフォーカスが自動的に移動します。
4. **次を検索** をクリックして、最初の安定な画面、API 呼び出し、またはエラーに進みます。

   エラー メッセージは、**情報** タブに表示されます。

   メニュー ツリーでは、再生エラーが含まれている API ノードには、赤い "X" 印が付いています。

開いている再生 TrueLog に対応する元の記録 TrueLog を表示するには、メニュー ツリーで再生 TrueLog の仮想ユーザー ノードを右クリックし、**関連する記録 TrueLog を開く** を選択します。 これにより、各サーバー レスポンスが **ホスト画面** ビューに表示されます。

ターミナル エミュレーション テスト スクリプトを分析する

テスト スクリプトを生成したら、スクリプトの試行を行って、スクリプトがエラーなしで動作するかどうかを確認します。 スクリプトの試行を行うことによって、記録された操作がスクリプトで正確に再現されるかどうかがわかります。

スクリプトの試行のデフォルトのオプション設定には、テストの実行中にダウンロードされるデータの実況表示や、ログ ファイルおよびレポート ファイルの作成などがあります。

スクリプトの試行では、トランザクション間の思考時間による遅延が発生しないように、1 人の仮想ユーザーのみが実行され、ストレス テスト オプションが有効になります。

1. ワークフロー バーで、**スクリプトの試行** をクリックします。 **スクリプトの試行** ダイアログ ボックスが開きます。
2. スクリプトの試行中にレンダリングされたページのトランザクションを表示するには、**TrueLog Explorer によるアニメーション実行** チェック ボックスをオンにします。
3. **実行** をクリックします。

   注: ここでは、実際の負荷テストではなく、スクリプトのデバッグが必要かどうかを確認するためのテスト実行のみを実行します。

   スクリプトの試行が開始されます。 **監視** ウィンドウが開き、実行の進捗についての詳細な情報が表示されます。

   TrueLog Explorer が開き、スクリプトの試行の実行時に実際にダウンロードされたデータが表示されます。 記録時にアクセスされた主要なターミナル エミュレーション ウィンドウはそれぞれ、上位レベルの WebTelnetRecvCommand API ノードとしてメニュー ツリーに一覧表示されます。

再生 TrueLog と記録 TrueLog を比較する

ターミナル エミュレーション アプリケーションをテストする場合、再生 TrueLog と記録 TrueLog の間で、対応するフィールド値に差異が発生することがあります。 それらの一部は、再生エラーの原因となる場合があります。

1. 再生 TrueLog を開いて、ワークフロー バーの **テストの分析** をクリックします。 **ワークフロー - テストの分析** ダイアログ ボックスが開きます。
2. **テスト実行の比較** をクリックします。
3. 関連する記録 TrueLog が **比較ビュー** に表示されます。 **TrueLog のステップ スルー** ダイアログ ボックスも開きます。
4. 検索の間隔 ([安定な画面]、[API 呼び出し]、[エラー] のいずれか) を選択します。
5. **次を検索** をクリックして、記録 TrueLog と再生 TrueLog 内の対応する安定な画面、API 呼び出し、またはエラーの最初のセットに進みます。 コントロールおよびスクリーンショットの値と状態をビジュアルに比較して、差分があるかどうかを確認します。
6. (検出した差分に基づいて) 必要なカスタマイズを完了したら、**スクリプトの試行** をクリックして、カスタマイズ結果がエラーなしで動作することを確認します。
7. その後のテスト実行の結果を分析して、カスタマイズが成功したかどうか、さらなるカスタマイズが必要かどうかを判断します。
8. **比較モード** をクリックして、比較モードを無効にします。
9. スクリプトが十分にカスタマイズされ、必要なカスタマイズがすべて追加されるまで、この手順を繰り返します。

記録 TrueLog と再生 TrueLog を同期させる

比較モードでは、再生 TrueLog と記録 TrueLog の間で、対応する API ノードを同期させることができます。

1. 比較モードを有効にします。
2. 一組の記録/再生 TrueLog をロードします。
3. 同期させる API ノードを右クリックします。
4. **TrueLog の同期** を選択します。 TrueLog Explorer は、対応する TrueLog 内で、選択された API ノードと最も相関がある API ノードを見つけます。

ターミナル エミュレーション TrueLog 関数

6 種類のターミナル エミュレーション関数をテスト スクリプトに挿入できます。

WebTelnetScreenVerifyText	画面上のテキスト (**ホスト画面** ビューで参照できます) を検証します。
WebTelnetScreenGetText	画面上のテキスト (**ホスト画面** ビューで参照できます) を解析します。
WebTelnetScreenVerifyField	テキスト フィールド (**ホスト画面情報** ビューで参照できます) を検証します。
WebTelnetScreenGetField	フィールドのテキスト (**ホスト画面情報** ビューで参照できます) を解析します。
WebTelnetScreenVerifyStatus	フィールドのステータス (両方のビューで参照できます) を検証します。
WebTelnetScreenGetStatus	フィールドのステータス (両方のビューで参照できます) を解析します。

ユーザー入力データをカスタマイズする

注: ターミナル エミュレーションのユーザー入力データのカスタマイズは、TN3270 と TN5250 の両方のプロトコルのスクリプトでサポートされています。

1. ターミナル エミュレーション TrueLog が TrueLog Explorer にロードされた状態で、カスタマイズするユーザー入力データが含まれる WebTelnetSendRecordExecute API 呼び出し (**API ノードのツリー メニュー**) を選択します。
2. **ホスト画面** タブを選択します。
   ホスト画面 タブには、ユーザー入力画面の画面キャプチャが表示されます。
3. **ホスト画面情報** タブを選択します。
   ホスト画面情報 タブには、選択した画面にあるすべてのユーザー入力データ フィールドのリストが表示されます。 各入力フィールドは、画面に表示される順序で番号が付けられ、表示されます。 **ホスト画面情報** タブで入力フィールドのエントリを選択すると、**ホスト画面** タブの画面キャプチャで、対応するフィールドが強調表示されます。

   注: 記録時にデータが入力された入力フィールドだけが強調表示され、カスタマイズが可能です。
4. **ホスト画面情報** タブで、カスタマイズするフィールド エントリを右クリックし、**ユーザー入力のカスタマイズ** を選択します。 **パラメータ ウィザード** が起動します。
5. **パラメータ ウィザード** を使用して、入力データのパラメータをスクリプトに含める方法を定義します。
   パラメータ ウィザード の操作方法については、Silk Performer のヘルプを参照してください。
   パラメータ ウィザード を使用してスクリプトに適用した変更は、[BDL] タブで確認できます。

Try Script の試行を実行して、新しいパラメータによって、ユーザー入力データが適切に挿入されることを確認します。

ターミナル エミュレーション アプリケーションの検証関数

このセクションでは、ターミナル エミュレーションの API 呼び出しにおいて、コンテンツとフィールドを検証する方法について説明します。

ターミナル エミュレーション アプリケーションの検証関数

TrueLog Explorer では、テスト スクリプトにコンテンツ検証を追加することにより、サーバーから送信されたコンテンツが現実の条件下でクライアントによって本当に受信されるかどうかを簡単に確かめることができます。 コンテンツ検証関数は、任意のターミナル エミュレーション API 呼び出しに対して挿入できます。 たとえば、入力データが入力される WebTelnetSendCommand 関数呼び出しの場合は、戻り値検証関数を挿入できます。 **ホスト画面** ビューおよび **ホスト画面情報** ビューのどちらからでも、コンテキスト メニューから検証関数を挿入できます。

TrueLog Explorer では、再生テスト実行と記録テスト実行を比較することによって、ターミナル エミュレーション アプリケーションの負荷が重い場合でも、入力データが実際にクライアントにダウンロードされて表示されるかどうかを確認できます。これは、クライアント/サーバー環境でのエンドユーザーの状況をテストするという課題に対する他に類を見ないほど強力なアプローチです。 これにより、他のターミナル エミュレーション トラフィック シミュレーション ツールでは検出できないようなエラー、つまり、標準的な負荷テスト スクリプトでは検出されない負荷がかかっている場合にのみ発生するエラーを検出できるようになります。

コンテンツの検証は、稼働中のパフォーマンス管理に利用できるので、システムを配置した後にも役立ちます。

ホスト画面 ビューまたは **ホスト画面情報** ビューで検証する入力フィールドを右クリックすることで、必要な検証関数を生成し BDL スクリプトに挿入できます。 TrueLog Explorer には、あらかじめ有効になっている検証関数が 3 つあります (WebTelnetScreenVerifyText、WebTelnetScreenVerifyField、および WebTelnetScreenVerifyStatus)。

コンテンツ検証と フィールド検証の比較

コンテンツ検証では、画面上のテキスト文字列の検証を、その画面上の位置にかかわらず、**ホスト画面** ビューで選択および設定できます。 フィールド検証を利用すると、ターミナル ビュー内の定義済み領域 (フィールド) の検証を設定できます。 アプリケーションで画面全体が単一のフィールドとして定義されていたり、ページがヘッダー フィールド、フッター フィールド、ボディ フィールドなどで構成されている場合があります。 フィールドは、入力領域として使用される場合が多いものの、必ずしも入力を受け取らなければならないわけではありません。 画面上の文字は、必ずしもフィールド内に存在しなければならないわけではありません。

コンテンツ検証関数を挿入する

1. **ホスト画面情報** ビューで、戻り値を検証する入力フィールドを右クリックします。
2. **選択したテキストの検証** を選択します。
3. **テキスト検証関数の挿入** ダイアログ ボックスが開きます。 このダイアログを使用すると、BDL スクリプトに挿入する検証関数のタイプを指定できます。 **検証内容: 列 x、行 x から始まる一連の文字が以下に** リスト ボックスから、[等しい] または [等しくない] を選択します。
4. [定数値] または [パラメータ値] のどちらに対する検証であるかを指定します。
5. 検証で [大文字/小文字を区別する] かどうかと、空白を無視するかどうかを指定します。
6. ダイアログ ボックスの **深刻度** エリアで、検証が否定的な結果を返した場合に発生させる深刻度を指定します([エラー]、[警告]、[情報]、または [カスタム])。

7. **OK** をクリックすると、検証関数がテスト スクリプトに追加されます。 BDL スクリプトが正常に変更されたら、BDL スクリプトに追加する検証ごとに、この手順を繰り返します。

ステータス検証関数を挿入する

ステータス検証関数を使用すると、ターミナル エミュレーションの入力フィールドのステータス値を検証できます。 ステータスは名前と値のペアです (検証が可能なのはステータス名の値です)。

1. **ホスト画面情報** ビューで、入力フィールドを右クリックします。
2. **ステータス値の検証** を選択します。
3. **ステータス値検証関数の挿入** ダイアログ ボックスが開きます。 このダイアログを使用すると、BDL スクリプトに挿入する検証関数のタイプを指定できます。 **検証内容: ステータス** リスト ボックスから、検証するステータスの名前を選択します。
4. [等しい] または [等しくない] を選択して、検証するステータスの状態を定義します。
5. [定数値] または [パラメータ値] のどちらに対する検証であるかを指定します。
6. 検証で [大文字/小文字を区別する] かどうかと、空白を無視するかどうかを指定します。
7. ダイアログ ボックスの **深刻度** エリアで、検証が否定的な結果を返した場合に発生させる深刻度を指定します([エラー]、[警告]、[情報]、または [カスタム])。
8. **OK** をクリックすると、ステータス検証関数がテスト スクリプトに追加されます。 BDL スクリプトが正常に変更されたら、BDL スクリプトに追加するステータス検証ごとに、この手順を繰り返します。

フィールド検証関数を挿入する

ターミナル エミュレーションの入力フィールドにフィールド検証関数を定義すると、フィールド内のテキストを検証できます。

1. **ホスト画面情報** ビューでフィールドを右クリックして、**フィールド テキストの検証** を選択します。
2. **フィールド検証関数の挿入** ダイアログ ボックスが開きます。 このダイアログを使用すると、BDL スクリプトに挿入する検証関数のタイプを指定できます。
3. 検証の演算子として [等しい] (または [等しくない]) を選択します。
4. [定数値] または [パラメータ値] のどちらに対する検証であるかを指定します。
5. 検証で [大文字/小文字を区別する] かどうかと、空白を無視するかどうかを指定します。
6. ダイアログ ボックスの **深刻度** エリアで、検証が否定的な結果を返した場合に発生させる深刻度を指定します([エラー]、[警告]、[情報]、または [カスタム])。
7. **OK** をクリックすると、検証関数がテスト スクリプトに追加されます。 BDL スクリプトが正常に変更されたら、BDL スクリプトに追加するその他の検証ごとに、この手順を繰り返します。

ターミナル エミュレーション アプリケーションの解析関数

このセクションでは、セッション固有の文字列をパラメータ化する必要がある場合に、値の解析を使用して正確な再生を可能にする方法について説明します。

ターミナル エミュレーション アプリケーションの解析関数

ターミナル エミュレーション TrueLog 内の画面上の任意のテキスト、フィールド、またはステータスに対して解析関数を挿入できます。 値の解析は、正確な再生のためにセッション固有の文字列をパラメータ化する必要がある場合 (たとえば、新たに作成された顧客 ID を解析して、請求書の顧客 ID フィールドに挿入されるパラメータにするような場合)、再生を可能にするのに役立ちます。

コンテンツ解析と フィールド解析の比較

コンテンツ解析関数を利用すると、画面上のテキスト文字列の解析を、その画面上の配置にかかわらず、**ホスト画面** ビューで選択および設定することができます。 フィールド解析関数を利用すると、ターミナ

ル ビュー内の定義済み領域 (フィールド) の解析を設定できます。 アプリケーションで画面全体が単一のフィールドとして定義されていたり、ページがヘッダー フィールド、フッター フィールド、ボディ フィールドなどで構成されている場合があります。 フィールドは、入力領域として使用される場合が多いものの、必ずしも入力を受け取らなければならないわけではありません。 画面上の文字は、必ずしもフィールド内に存在しなければならないわけではありません。

コンテンツ解析関数を挿入する

1. **ホスト画面** ビューで、値を解析するテキストを選択します。
2. 右クリックして **選択したテキストの解析** を選択します。
3. **テキスト解析関数の挿入** ダイアログ ボックスには、解析関数を調整するための設定があります。
   デフォルト設定で適切と思われますが、以下の設定を調整することもできます。
   - **パラメータ名**: 解析関数の結果を受け取るパラメータの名前を入力します。
   - **スクリプトに出力文を挿入**: 通知用の Print 文をスクリプトに挿入する場合は、**Print 文**をオンにします。 これによって、解析関数の結果が Silk Performer の **仮想ユーザー** 出力ウィンドウに出力されます。 (Print 文を実行して **仮想ユーザー出力** ウィンドウに値を書き込むほかに) デバッグしやすくするために解析値を出力ファイルに書き込む場合は、**Writeln** 文 をオンにします。 出力ファイルを生成すると負荷テストの時間測定値が変化するので、これらのファイルはデバッグのためだけに使用し、完全な負荷テストでは生成しないでください。
4. **OK** をクリックすると、解析関数がテスト スクリプトに挿入されます。

ステータス解析関数を挿入する

1. **ホスト画面情報** ビューでステータスを右クリックして、**ステータス値の解析** を選択します。

   注: この関数には、**ホスト画面** ビューでフィールドを右クリックすることでもアクセスできます。

2. **ステータス値解析関数の挿入** ダイアログ ボックスには、解析関数を調整するための設定があります。
   デフォルト設定で適切と思われますが、以下の設定を調整することもできます。
   - **これはステータス**:既存のステータス名のリスト ボックスから、挿入する解析関数の対象となるステータスを選択します。
   - **パラメータ名**: 解析関数の結果を受け取るパラメータの名前を入力します。
   - **スクリプトに出力文を挿入**: 通知用の Print 文をスクリプトに挿入する場合は、**Print 文**をオンにします。 これによって、解析関数の結果が Silk Performer の **仮想ユーザー** 出力ウィンドウに出力されます。 (Print 文を実行して **仮想ユーザー出力** ウィンドウに値を書き込むほかに) デバッグしやすくするために解析値を出力ファイルに書き込む場合は、**Writeln** 文 をオンにします。 出力ファイルを生成すると負荷テストの時間測定値が変化するので、これらのファイルはデバッグのためだけに使用し、完全な負荷テストでは生成しないでください。
3. **OK** をクリックすると、解析関数がテスト スクリプトに挿入されます。

フィールド解析関数を挿入する

1. **ホスト画面情報** ビューでフィールドを右クリックして、**フィールド テキストの解析** を選択します。
2. **フィールド解析関数の挿入** ダイアログ ボックスには、解析関数を調整するための設定があります。 デフォルト設定で適切と思われますが、以下の設定を調整することもできます。
   - **パラメータ名**: 解析関数の結果を受け取るパラメータの名前を入力します。
   - **スクリプトに出力文を挿入**: 通知用の Print 文をスクリプトに挿入する場合は、**Print 文**をオンにします。 これによって、解析関数の結果が Silk Performer の **仮想ユーザー** 出力ウィンドウに出力されます。 (Print 文を実行して **仮想ユーザー出力** ウィンドウに値を書き込むほかに) デバッグしやすくするために解析値を出力ファイルに書き込む場合は、**Writeln** 文 をオンにします。 出力ファイルを生成すると負荷テストの時間測定値が変化するので、これらのファイルはデバッグのためだけに使用し、完全な負荷テストでは生成しないでください。

3. **OK** をクリックすると、解析関数がテスト スクリプトに挿入されます。

AJAX 対応 Web アプリケーションでの作業

このセクションでは、JSON データや XML データを "きれいに" 整形する方法と、AJAX 技術を使ったアプリケーションのテストに TrueLog Explorer のスクリプト カスタマイズ機能を適用する方法について説明します。

Silk Performer Recorder は、AJAX (Asynchronous JavaScript and XML) リクエストを利用する Web アプリケーションを記録し、(TrueLog Explorer の助けにより) 再生することができます。 これが可能なのは、HTML レスポンス内の XML や JSON の形式で受信した AJAX 同期リクエスト/レスポンスを Silk Performer が認識しているからです。

AJAX の詳細については、Silk Performer のヘルプを参照してください。

AJAX サポートの概要

TrueLog Explorer および Silk Performer では、AJAX リクエスト内の値にアクセスすることができます。これを利用すると、AJAX レスポンス内での入力データのパラメータ化、検証関数、解析関数、セッション情報のカスタマイズなど、TrueLog Explorer スクリプトのカスタマイズが簡単にできるようになります。

JSON データと XML データの書式整形

JSON および XML は、AJAX アプリケーションや REST 技術などの環境で一般に使われるデータ構造形式です。 TrueLog Explorer は、記録されたスクリプト内に含まれる XML 形式や JSON 形式のバイト ストリームを、書式整形して表示する機能をサポートしています。 JSON 形式のデータのレンダリングを拡張することで、文字列の検証や解析など、TrueLog Explorer のカスタマイズ機能を利用した文字列値のカスタマイズができるようになっています。

JSON ビューの書式整形は比較モードでもサポートされていて、記録セッションと再生セッションを並べて比較し、自動的に差分を検出することができます。

注: TrueLog Explorer には、JSON データを書式整形された JSON レンダリング ビューで表示するか、手を加えないそのままの JSON バイト ストリームとして表示するかを選択するオプションがあります。ただし、検証や解析などのスクリプト カスタマイズ機能は、JSON データをそのまま表示するビューではサポートされません。

書式整形したデータ表示は Silk Performer でも可能で、可読性の向上や B.F. スクリプトのカスタマイズに役立ちます。

TrueLog Explorer で JSON および XML 表示の書式整形を有効化する

TrueLog Explorer が JSON データや XML データを検出できない場合には、**レンダリング** ページで JSON および XML を表示する際の書式整形を有効化します。 TrueLog Explorer が JSON データや XML データを検出すると、**レンダリング** ページは自動的に書式整形されます。

1. TrueLog メニュー ツリーで JSON 形式または XML 形式のデータを含むノード (HTTP Post コマンド ノードなど) を選択します。
2. **レンダリング** タブをクリックします。
3. ツール バーの **自動表示** をクリックします。 JSON データおよび XML データが自動的に書式整形されます。
4. JSON データまたは XML データを右クリックし、**ファイルの種類を指定してレンダリング** > **JSON** または **ファイルの種類を指定してレンダリング** > **XML** を選択してデータを書式整形します。

TrueLog Explorer のカスタマイズ

このセクションでは、TrueLog Explorer のカスタマイズ、表示、および TrueLog 生成の各機能を、特定のニーズに適合させる方法について説明します。 TrueLog Explorer のオプション設定を使用すると、データの表示、差異を検出するための区切り文字の定義、仮想ユーザー レポートの生成などを構成できます。 カスタマイズ機能を使用すると、UI コマンドおよびツールバーの表示をカスタマイズできます。

比較モードを使用すると、一度に 1 API ノードずつ、記録 TrueLog と再生 TrueLog を直接比較できます。 一方、差分モードを使用すると、**ソース差分** ビューを介して、Web、XML、およびデータベース アプリケーションの記録 TrueLog と再生 TrueLog の相違を自動的に識別して一覧表示できます。 また、差分モードでは、**ソース**、**受信データ**、**送信データ**、**受信ヘッダー**、**送信ヘッダー**、および **BDL** の各ビューにおけるテキスト ベースの形式での相違も、異なるテキストを緑と赤で示すことで表示されます。

TrueLog Explorer のアニメーションを有効にすると、ダウンロードされている再生コンテンツ (グラフィックス、テキスト、SQL コマンドなど) をリアルタイムで表示できます。

最後に、TrueLog ファイルの生成には CPU/メモリが大量に使用される場合があるので、Silk Performer を使用して TrueLog の特性およびファイルが生成される基準を調整する方法を説明します。

TrueLog Explorer のオプションの設定

オプション ダイアログ ボックスで設定できる TrueLog Explorer のオプション設定を使用すると、HTML ドキュメントや受信/送信データの表示、TrueLog 間の相違を特定する区切り文字の定義、仮想ユーザー レポートの生成などのアプリケーション設定を構成できます。

表示オプションを設定する

1. **設定** > **オプション** を選択します。 **表示** タブが開きます。
2. HTML ドキュメントをレンダリングする方法を指定します。
   - HTML ドキュメントをレンダリングするときにクライアント側スクリプトの実行を有効にするには、**スクリプトを実行** チェック ボックスをオンにします。
   - HTML ドキュメントをレンダリングするときに画像またはフレームを表示するには、**フレームと画像をロード** チェック ボックスをオンにします。
   - HTML ドキュメントをレンダリングするときにアプレットまたは ActiveX コントロールの実行を有効にするには、**アプレットを実行** チェック ボックスをオンにします。
3. **送信ヘッダー**、**送信データ**、**受信ヘッダー**、および **受信データ** ビューで、受信データおよび送信データを表示する方法を指定します。
   - データに最も適した表示形式 (たとえば、画像の場合はバイナリ形式、HTML ドキュメントの場合はテキスト形式) を TrueLog Explorer に自動的に選択させるには、**自動表示モード** チェック ボックスをオンにします。
   - テキスト書式設定のシンボル (たとえば、復帰改行の場合は ¥r¥n、空白文字の場合は ".") を表示するには、**空白を表示** チェック ボックスをオンにします。

     このオプションは、データをテキスト形式で表示する場合にのみ関係します。
   - データをバイナリ形式で表示するには、**バイナリ データを表示** チェック ボックスをオンにします。

     このオプションは、**自動表示モード** をオンにしていない場合にのみ指定できます。
4. TrueLog メニュー ツリーにフィルタを使用するには、**フィルタを適用** チェック ボックスをオンにします。 たとえば、TrueLog メニュー ツリーに特定のタイプのファイル (画像、スクリプト、スタイル シートなど) を表示しないようにするには、フィルタを使用します。
5. **OK** をクリックします。

区切り文字を挿入する

TrueLog Explorer の一般比較タグを使用すると、区切り文字を定義できます。区切り文字は、TrueLog 間の相違を区切るための文字またはテキストです。 区切り文字は、あるシーケンスを単一の相違として扱うのか、それとも複数の相違として扱うのかを決定します。 区切り文字は、HTML を除くすべてのドキュメント タイプに使用されます。

1. **設定** > **オプション** を選択して、**比較タグ** タブをクリックします。 **比較タグ** ページが開きます。
2. **一般比較タグ** テーブルに、区切り文字 (文字またはテキスト) を入力します。

   ヒント: 変更を破棄してデフォルトの設定に戻すには、**デフォルト** をクリックします。

3. **OK** をクリックします。

HTML 区切り文字を挿入する

HTML 区切り文字は、HTML TrueLog 間の相違を区切るための文字またはテキストです。 区切り文字は、あるシーケンスを単一の相違として扱うのか、それとも複数の相違として扱うのかを決定します。

1. **設定** > **オプション** を選択して、**HTML 比較タグ** タブをクリックします。 **HTML 比較タグ** ページが開きます。
2. **HTML 比較タグ** テーブルに、区切り文字 (文字またはテキスト) を入力します。

   ヒント: 変更を破棄してデフォルトの設定に戻すには、**デフォルト** をクリックします。

3. **OK** をクリックします。

仮想ユーザー要約レポートを有効にする

TrueLog Explorer のワークスペース オプションを使用すると、アニメーション化されたスクリプトの試行の実行の終了時や、メニュー ツリーで TrueLog ファイルのルート ノードを選択したときに、仮想ユーザー レポートを自動的に表示できます。

1. **設定** > **オプション** を選択して、**ワークスペース** タブをクリックします。 **ワークスペース** ページが開きます。
2. **仮想ユーザー レポートを表示する** チェック ボックスをオンにします。

   ヒント: 変更を破棄してデフォルトの設定に戻すには、**デフォルト** をクリックします。

3. **OK** をクリックします。

ツールバーとコマンドのカスタマイズ

このセクションでは、ツールバーとそこに表示されるコマンドをカスタマイズする方法について説明します。

どのツールバーを表示するかを指定する

TrueLog Explorer インターフェイスにどのツールバーを表示するかをカスタマイズできます。

1. **設定** > **カスタマイズ** を選択します。 **ツールバー** ページが開きます。
2. 表示したいツールバーの横のチェック ボックスをオンにします。
3. 省略可能: ツールチップ (マウスを重ねたときに表示される UI コントロールの説明) を有効化したい場合には、**ツールチップを表示** チェック ボックスをオンにします。
4. 省略可能: TrueLog Explorer の "クールな外見" を有効化したい場合には、**クールな外見** チェック ボックスをオンにします。 クールな外見を指定すると、Windows 3.1 形式の影の付いたボタンが影のないボタンに置き換えられます。

5. **OK** をクリックします。
   代替方法:**適用** をクリックして、選択した内容を即座に UI に適用します。

カスタム ツールバーを作成する

1. **設定** > **カスタマイズ** を選択します。 **ツールバー** ページが開きます。
2. **ツールバー** タブをクリックします。
3. **新規作成** をクリックします。 **ツールバーの新規作成** ダイアログ ボックスが開きます。
4. ツールバーの名前を入力して **OK** をクリックします。
5. **OK** をクリックします。

新しいツールバーに表示されるコマンド ボタンをカスタマイズします。

ツールバーのコマンド ボタンをカスタマイズする

特定のツールバーにどのコマンド ボタンを表示するかを定義することができます。

注: メニュー バー ツールバーはカスタマイズできません。

1. **設定** > **カスタマイズ** を選択します。 **ツールバー** ページが開きます。
2. **コマンド** タブをクリックします。
3. **カテゴリ** リスト ボックスで、コマンド ボタンをカスタマイズしたいツールバーを選択します。
4. **ボタン** ボックスに表示されているコマンドをクリックすると、**説明** ボックスでその説明を見ることができます。
5. ツールバーに追加したいコマンドを、**カテゴリ** リスト ボックスに表示されたそのツールバーの名前にドラッグします。
6. **OK** をクリックします。

表示モード

比較モードおよび差分モードを使用すると、TrueLog の相違を表示できます。 比較モードを使用すると、一度に 1 API ノードずつ、記録 TrueLog と再生 TrueLog を直接比較できます。 一方、差分モードを使用すると、**ソース差分** ビューを介して、Web、XML、およびデータベース アプリケーションの記録 TrueLog と再生 TrueLog の相違を自動的に識別して一覧表示できます。 また、差分モードでは、**ソース**、**受信データ**、**送信データ**、**受信ヘッダー**、**送信ヘッダー**、および **BDL** の各ビューにおけるテキスト ベースの形式での相違も、異なるテキストを緑と赤で示すことで表示されます。

比較モード

比較モードでは、再生 TrueLog とアプリケーションの記録中に生成された対応する TrueLog が比較されます。 デフォルト ビューには再生 TrueLog が表示されます (左上隅に緑の三角形が付いています)。 比較ビューには記録 TrueLog が表示されます (左上隅に赤い三角形が付いています)。

対応する記録 TrueLog を開いている場合、または **ファイルを開く** ダイアログ ボックスで **比較ビューで開く** チェック ボックスをオンにした場合、TrueLog は比較ビューで表示されます。

注: ワークフロー - セッション処理のカスタマイズ ダイアログ ボックスから **差分の検索** を実行すると、TrueLog は自動的に比較モードで開かれます。

比較モードを有効にする

再生 TrueLog と対応する記録 TrueLog を比較して再生エラーを分析するには、比較モードを使用します。 **ファイルを開く** ダイアログ ボックスで **比較ビューで開く** チェック ボックスをオンにすると、TrueLog が自動的に比較モードで表示されます。 オフにすると、TrueLog はデフォルト ビューで表示されます。

注: 比較モードでは、記録と再生の **ソース** ビューを、左右ではなく上下に配置できます。 **ソース** ビューを左右に並べて比較するには、**表示** > **左右に並べて比較** を選択します。

1. TrueLog メニュー ツリーで TrueLog を選択します。
2. 以下の手順の 1 つを実行します。
   - **比較モード** をクリックします。
   - **表示** > **比較モード** を選択します。

> **注:** TrueLog の比較には、複数のエントリ ポイントやアプローチが利用可能です。

差分モード

スクリプトの試行の実行中に生成された TrueLog とアプリケーションの記録中に生成された対応する TrueLog を比較することは、負荷テスト スクリプトの問題を特定する有効な手段です。

ソース差分 ページから使用できる**差分テーブル**には、再生 TrueLog とそれに対応する記録 TrueLog の間で検出された差分の一覧が自動的に表示されます。

デフォルトで有効になる差分モードは特別な種類の比較モードであり、**ソース**、**受信データ**、**送信データ**、**受信ヘッダー**、および **送信ヘッダー** の各ビューで、テキストの差分が色付きでマーキングされます (赤と緑およびグレーの背景)。

差分モードを有効にする

差分モードはデフォルトで有効になります。

以下の手順の 1 つを実行します。

- **表示** > **差分モード** を選択します。
- **差分モード** アイコンをクリックします。

差分テーブルを表示する

このタスクを開始する前に、差分モードが有効になっていることを確認してください (差分モードはデフォルトで有効になります)。

1. 対応する記録 TrueLog と再生 TrueLog のペアを開きます。
2. **表示** > **比較モード** を選択して比較モードを有効にします。
3. **ソース差分** タブをクリックします。 再生 TrueLog と対応する記録 TrueLog の間で検出された差分が、自動的に一覧表示されます。

データのアニメーション

アニメーションを有効にすると、仮想ユーザー実行の間にサーバーから受信するコンテンツが、ユーザーが見るのと同じようにリアルタイムで表示されます。 Web アプリケーションのテストの場合、これは、仮想ユーザーと同じように、ページ コンテンツのロード (グラフィックス、テキスト、および埋め込みオブジェクト) を見ることができることを意味します。 アニメーションはデフォルトで有効になっています。

TrueLog アニメーションは、TrueLog On Error を有効にして負荷テストを実行し、個別のユーザーに関するエラーを Silk Performer の **監視** ウィンドウ内の **仮想ユーザー** ビューに表示されるエラー数で検出する場合に役に立ちます。 **仮想ユーザー** ビューのコンテキスト メニューで **TrueLog の検討** をクリックすると、すぐに TrueLog On Error ファイルを見ることができます。 ただし、この機能はコントローラ マシンで実行している仮想ユーザーに対してのみ使用できます。負荷テストの間は、リモート エージェントの TrueLog にはアクセスできません。

アニメーションの一時停止、再開、および停止

アニメーションを一時停止または再開するには、**アニメーション** > **アニメーションの一時停止** を選択します。 代替方法: キーボードで Ctrl+P を押します。

アニメーションを停止するには、**アニメーション** > **アニメーションの中止** を選択します。

スケーラビリティに対する TrueLog の影響

TrueLog の生成は、負荷テストの環境に適合できる場合にのみ役に立ちます。 したがって、メモリとパフォーマンスの要件を最小限に留める必要があります。

TrueLog のスケーラビリティに影響を与える要因として次の 2 つがあります。

- パフォーマンスのダウングレード
- メモリ使用量

パフォーマンスのダウングレード

TrueLog の生成に伴ってパフォーマンスがダウングレードする可能性は、0 から 10% の範囲です。 これは、ランタイムはすべての関連データをメモリに保持し、必要なメモリ割り当てを最低限に維持しようと試みるためです。 TrueLog On Error によるパフォーマンスへの影響はごくわずかです。

メモリ使用量

スケーラビリティに関して、TrueLog と TrueLog On Error の間には大きな違いがあります。 TrueLog 機能は、個々のリクエストとレスポンスを TrueLog ファイルにすぐに書き込みます。 TrueLog ファイルの書式設定のために内部 CPU 使用量は多くなり、TrueLog ファイルへの書き込みのために大量の IO が実行されます。

このため、多くの場合、TrueLog の生成は負荷テストには適していません。

TrueLog はサイズが大きくなることがあるため、テストの間にエラーが発生した場合にのみ TrueLog を生成するように、Silk Performer を構成することができます。 システムが正確に動作している間は何も記録されません。 このような対象を絞った TrueLog の生成は *TrueLog On Error* と呼ばれ、その結果、TrueLog ファイルはより小さい限定的なものになります。

TrueLog On Error の場合のメモリ要件は、利用する BDL スクリプトによって大きく異なります。 したがって、一般的な割合を示すことはできません。特に、完全なトランザクションのログを取得するように TrueLog On Error が構成されている場合は不可能です。 これが当てはまるのは、TrueLog On Error ファイルに対してデフォルトではない **1 トランザクションごと** の TrueLog の保存の設定が選択されている場合だけであることに注意してください。 このような場合、メモリの使用量は、トランザクションに含まれるリクエストされた Web ページの数とサイズに関連します。 デフォルトの設定である **コンテンツ履歴ベース** の場合は、トランザクションの長さによる影響はありません。 通常、コンテンツの履歴は 5 ページ分の Web ページを超えることはなく、保存する必要があるのはこれらのページのみです。

カスタム コンテンツ タイプとファイル拡張子

TrueLog Explorer では、特殊なコンテンツ タイプおよび URL ファイル拡張子のカスタム処理がサポートされます。 カスタム コンテンツ タイプおよびファイル拡張子をいくつかのレジストリ値に追加することによって、製品の動作を変更できます。 関連する設定は、レジストリ (HKEY_CURRENT_USER¥Software¥Silk¥Silk TrueLog Explorer¥9.5¥) にあります。

この中に、次の 4 つのレジストリ値があります。

- MaskContentTypes
- MaskExtensions
- SuppressContentTypes
- SuppressExtensions

これらの値には、特殊な処理を必要とするコンテンツ タイプとファイル拡張子の一覧が含まれます。 Mask* の値はテキスト データに適用され、これらは HTML 表示でプレーン テキストとしてレンダリングできます。 Suppress* の値はバイナリ (テキスト以外の) データに適用され、これらは正しくレンダリングできません。

ファイル ダウンロード ダイアログ ボックスを非表示にする

1. TrueLog Explorer で、ファイル ダウンロード ダイアログ ボックスが開くノードを選択します。
2. **受信ヘッダー** タブをクリックして、内容を調べます。
3. コンテンツ タイプが見慣れないものの場合は (app/fireclick.x-hint.1 など)、以下の手順を実行します。
   a) TrueLog Explorer を閉じます。
   b) 見慣れないコンテンツ タイプを MaskContentTypes の値に追加します。
      複数の文字列はセミコロンで区切ります。
   c) TrueLog Explorer を開始します。
      レジストリの設定は起動時にのみ読み取られます。
   d) ダウンロード ボックスがまだ開く場合は、同じコンテンツ タイプを SuppressContentTypes の値に追加します。 ファイル ダウンロード ダイアログ ボックスは開かなくなります。
4. コンテンツ タイプに問題がない場合は、URL を調べます。
   a) 拡張子が .html、.js、.pl、.chtml、.jar、.class、または .vbs 以外の場合は、TrueLog Explorer を閉じ、**受信データ** ビューを調べて、レスポンス ボディがテキスト データまたはバイナリ データのどちらで構成されているかを確認します。
   b) **受信データ** ビューにテキスト データがある場合は、URL ファイル拡張子を MaskExtensions の値に追加します。
   c) データがバイナリの場合は、そのファイル拡張子を SuppressExtensions の値に追加します。
      複数の拡張子はセミコロンで区切ります。
   d) TrueLog Explorer を開始します。
      レジストリの設定は起動時にのみ読み取られます。

TrueLog 生成の設定

概要

TrueLog は CPU と I/O を大量に使用するので、通常、負荷テストの間は有効にはしません。 一般に、TrueLog はスクリプト開発/検証の間のみ有効にされます。 一方、TrueLog On Error ファイルは、パフォーマンスに対して最適化されており、CPU への影響がわずかなので、負荷テストの間も通常は有効になっています。 TrueLog On Error ファイルは、平均的な数のエラーが予想される場合であっても、大規模なテストに対してアクティブにすることができ、再生パフォーマンスに大きな影響はありません。

ただし、 Silk Performer による TrueLog On Error の生成は、スケーラビリティに対して若干の影響を及ぼします。 Silk Performer の TrueLog 記録レベルを次の方法で調整することにより、この影響を可能な限り低く抑えておくことができます。

- TrueLog の長さ
- TrueLog が記録されるインシデントの深刻度
- 埋め込みオブジェクトを含めるかどうか

以下のセクションで詳細に説明する TrueLog 生成オプションは、Silk Performer の **設定** > **アクティブ プロファイル** > **結果** > **TrueLog** を選択することによって構成できます。 この同じページを使用して、テストでの仮想ユーザーごとの TrueLog On Error ファイル (.xlg) の生成も有効にできます (**TrueLog On Error ファイル (.xlg)** チェック ボックス)。

TrueLog の長さ

TrueLog On Error の追跡は、一度に 1 トランザクションずつ完全なトランザクションを保存するように、または Web アプリケーションの場合はコンテンツ履歴の最新の 5 ページを保存するように構成できます。

1 トランザクションごと - この設定は、関連するすべてのページ (コンテンツ履歴) ではなく、エラーのある個々のトランザクションを追跡します。 メモリの消費は各トランザクションの長さに依存するので、このオプションは、メモリを心配しなくてよい場合、またはトランザクションが短い場合にのみ使用します。

コンテンツ履歴ベース - この設定は、最近のページの履歴のみを保存し、関連のないページはメモリから削除します。 この方法では、メモリの消費量の予測と管理が容易になります。 ただし、消費量がメモリに保存されるページのサイズに依存することに変わりはありません。

深刻度

ディスクへの TrueLog データの書き込み (最初は RAM に保存されます) をトリガするインシデントの深刻度を、**TrueLog On Error の生成対象** の設定で定義できます。

埋め込みオブジェクト

すべての埋め込みオブジェクトを記録する - TrueLog には、すべての埋め込みオブジェクトのレスポンスボディが含まれます。 リクエストおよびレスポンスのヘッダーが履歴に含まれなくなっても、埋め込みオブジェクトのコンテンツをブラウザに表示できます。 これは、キャッシュ ヒットであるためにページがサーバーに画像をリクエストしない場合、またはサーバーが「304 変更なし」エラーで応答した場合は、問題になる可能性があります。 すべての埋め込みオブジェクトを表示するため、Silk Performer は各埋め込みオブジェクトのコンテンツを、プロセス全体を対象とするグローバル キャッシュに保存します。

キャッシュされた埋め込みオブジェクトを除外する - キャッシュされたオブジェクトに対するグローバル ストアが有効になっていないと、一部の画像はロードされません。 この設定を選択すると、パフォーマンスがわずかに向上し、メモリの消費が減ります。

すべての埋め込みオブジェクトを除外する - 埋め込みオブジェクトのボディがまったくログに格納されない場合、埋め込みオブジェクトはロードされません。 この設定を選択すると、メモリの消費は大幅に減少します。 ただし、この方法で提供されるビジュアル情報には大きな欠損が発生します。

TrueLog On Error 情報がコンテンツ履歴に基づいて記録される場合は、仮想ユーザーごとのメモリ使用量に対する影響を、サンプル テストに基づいて予測できます。 次に示す値は、Amazon.com や IBM.com などのよく知られた Web サイトに対して実行したサンプル テストに基づいて計算されています。

設定	メモリ消費の増加
すべての埋め込みオブジェクトを除外する	80%
キャッシュされた埋め込みオブジェクトを除外する	125%
すべての埋め込みオブジェクトを記録する	130%

注: 必要なメモリの量が最低限の設定 ([TrueLog の保存] を [コンテンツ履歴ベース] とし、かつ [すべての埋め込みオブジェクトを除外する] を選択) は、すべての関連データのログ取得には十分です。ただし、含まれるビジュアル情報は大きく損なわれます。

解析関数と検証関数

このトピックスのマトリックスは、さまざまな検証関数と解析関数、Silk Performer Recorder、および TrueLog Explorer の間の相互作用を表しています。

利用可能なすべての BDL 関数の詳細については、『The BDL Function Reference』(英語) を参照してください。

HTML の解析関数と検証関数

HTML 解析関数

HTML コンテンツ	レスポンス データ	用途	説明
WebParseHtmlBound	WebParseDataBound	視覚的解析、スクリプト記述	与えられた境界内の HTML コンテンツ/レスポンス データを解析します。
WebParseHtmlBoundEx	WebParseDataBoundEx	視覚的解析、スクリプト記述	与えられた境界内の HTML コンテンツ/レスポンス データを解析します。
WebParseHtmlBoundArray	WebParseDataBoundArray	スクリプト記述	指定された境界内の HTML コンテンツ/レスポンス データを解析します。右側境界が識別された後でのみ、後続の左側境界が探索されます。
WebParseHtmlTitle	-	スクリプト記述	HTML ドキュメントのタイトルを解析します。
WebParseTable	-	視覚的解析、スクリプト記述	HTML テーブルのセル内のコンテンツを解析します。
-	WebParseResponseTag	スクリプト記述	指定された HTML 属性の値を解析します。 XML レスポンス データにも適用できます。
-	WebParseResponseTagContent	スクリプト記述	指定された HTML 属性のコンテンツを解析します。 XML レスポンス データにも適用できます。
-	WebParseResponseHeader	スクリプト記述	HTTP レスポンス ヘッダーを解析します。
-	WebParseResponseRedirect	スクリプト記述	URL エンコードされたパラメータのリダイレクション レスポンスの URL を解析します。

HTML 検証関数

HTML コンテンツ	レスポンス データ	用途	説明
WebVerifyHtml	WebVerifyData	視覚的検証、スクリプト記述	指定された文字列が HTML コンテンツ/レスポンス データに含まれているかどうかを確認します。
WebVerifyHtmlBound	WebVerifyDataBound	視覚的検証、スクリプト記述	指定された文字列が、与えられた境界内の HTML コンテンツ/レスポンス データ

HTML コンテンツ	レスポンス データ	用途	説明
			に含まれているかどうかを確認します。
WebVerifyHtmlBoundEx	WebVerifyDataBoundEx	視覚的検証、スクリプト記述	指定された文字列が、与えられた境界内の HTML コンテンツ/レスポンス データに含まれているかどうかを確認します。
WebVerifyHtmlTitle	-	視覚的検証、Recorder により自動生成、スクリプト記述	指定された文字列が HTML ドキュメントのタイトルに含まれているかどうかを確認します。
WebVerifyTable	-	視覚的検証、スクリプト記述	指定された文字列が HTML テーブルに含まれているかどうかを確認します。
WebVerifyHtmlDigest	WebVerifyDataDigest	Recorder により自動生成、視覚的検証	レスポンス データが、以前の記録セッション時に Recorder によってキャプチャされたレスポンス データと異なるかどうかを確認します。

XML の解析関数と検証関数

XML 解析関数

XML コンテンツ	レスポンス データ	用途	説明
-	WebXmlParseNodeValue	TrueLog Explorer の視覚的解析、スクリプト記述	レスポンス ドキュメントを解析し、渡された XPath の XML ノード値を返します。
-	WebXmlParseNodeAttribute	TrueLog Explorer の視覚的解析、スクリプト記述	レスポンス ドキュメントを解析し、渡された XPath を持つノードの XML 属性値を返します。

XML 検証関数

XML コンテンツ	レスポンス データ	用途	説明
-	WebXmlVerifyNodeValue	TrueLog Explorer の視覚的検証、スクリプト記述	ノードの値が指定値と一致するかどうかを確認します。
-	WebXmlVerifyNodeAttribute	TrueLog Explorer の視覚的検証、スクリプト記述	ノードの属性値が指定値と一致するかどうかを確認します。

データベースの値取得関数と検証関数

データベースの値取得関数

データベース関数	用途	説明
RsGetInt	視覚的解析、スクリプト記述	内部の結果セットから数値を取得します。データを生成する SQL 文 (SELECT 文) が実行されるたびに、それがどのデータベース API (ORA、ODBC) で生成されたかに関係なく、この結果セットには値が設定されます。
RsGetFloat	視覚的解析、スクリプト記述	内部の結果セットから浮動小数点値を取得します。データを生成する SQL 文 (SELECT 文) が実行されるたびに、それがどのデータベース API (ORA、ODBC) で生成されたかに関係なく、この結果セットには値が設定されます。
RsGetValue	スクリプト記述	内部の結果セットからバイナリ値を取得します。データを生成する SQL 文 (SELECT 文) が実行されるたびに、それがどのデータベース API (ORA、ODBC) で生成されたかに関係なく、この結果セットには値が設定されます。
RsGetString RsGetString2	視覚的解析、スクリプト記述	内部の結果セットから文字列値を取得します。データを生成する SQL 文 (SELECT 文) が実行されるたびに、それがどのデータベース API (ORA、ODBC) で生成されたかに関係なく、この結果セットには値が設定されます。
RsIsNull	スクリプト記述	内部の結果セットの値が NULL かどうかを調べます。データを生成する SQL 文 (SELECT 文) が実行されるたびに、それがどのデータベース API (ORA、ODBC) で生成されたかに関係なく、この結果セットには値が設定されます。
RsRows	スクリプト記述	内部の結果セットの行数を取得します。データを生成する SQL 文 (SELECT 文) が実行されるたびに、それがどのデータベース API (ORA、

データベース関数	用途	説明
		ODBC) で生成されたかに関係なく、この結果セットには値が設定されます。
RsCols	スクリプト記述	内部の結果セットの列数を取得します。データを生成する SQL 文 (SELECT 文) が実行されるたびに、それがどのデータベース API (ORA、ODBC) で生成されたかに関係なく、この結果セットには値が設定されます。

データベース検証関数

データベース関数	用途	説明
RsVerifyRowCount	視覚的検証、スクリプト記述	内部の結果セットに特定の数の行が含まれているかどうかを調べます。データを生成する SQL 文 (SELECT 文) が実行されるたびに、それがどのデータベース API (ORA、ODBC) で生成されたかに関係なく、この結果セットには値が設定されます。
RsVerifyInt	視覚的検証、スクリプト記述	内部の結果セット内の指定行/列にある数値要素を検証します。データを生成する SQL 文 (SELECT 文) が実行されるたびに、それがどのデータベース API (ORA、ODBC) で生成されたかに関係なく、この結果セットには値が設定されます。
RsVerifyFloat	視覚的検証、スクリプト記述	内部の結果セット内の指定行/列にある浮動小数点要素を検証します。データを生成する SQL 文 (SELECT 文) が実行されるたびに、それがどのデータベース API (ORA、ODBC) で生成されたかに関係なく、この結果セットには値が設定されます。
RsVerifyValue	スクリプト記述	内部の結果セット内の指定行/列にあるバイナリ要素を検証します。データを生成する SQL 文 (SELECT 文) が実行されるたびに、それがどのデータベース API (ORA、ODBC) で生成されたかに関係なく、この結果セットには値が設定されます。
RsVerifyString	視覚的検証、スクリプト記述	内部の結果セット内の指定行/列にある文字列要素を検証します。データを生成する SQL 文 (SELECT 文) が実行されるたびに、それがどのデータベース API (ORA、ODBC) で生成されたかに関係なく、この結果セットには値が設定されます。
RsVerifyNull	視覚的検証、スクリプト記述	内部の結果セット内の指定行/列にある要素が NULL かどうかを検証します。データを生成する SQL 文 (SELECT 文) が実行されるたびに、それがどのデータベース API (ORA、

データベース関数	用途	説明
		ODBC) で生成されたかに関係なく、この結果セットには値が設定されます。

Oracle Forms の解析関数と検証関数

Oracle Forms 解析関数

Oracle Forms 関数	用途	説明
OraFormsParseTextValue	TrueLog Explorer の視覚的解析、スクリプト記述	テキスト フィールドまたはテキスト領域コントロールの値を返します。
OraFormsParseCheckValue	TrueLog Explorer の視覚的解析、スクリプト記述	チェック ボックスの状態を返します。
OraFormsParseRadioValue	TrueLog Explorer の視覚的解析、スクリプト記述	オプション ボタンの状態を返します。
OraFormsParseListValue	TrueLog Explorer の視覚的解析、スクリプト記述	コンボ ボックス、リスト ボックス、またはポップアップ リスト ボックスの選択値を返します。
OraFormsParseListIndex	TrueLog Explorer の視覚的解析、スクリプト記述	コンボ ボックス、ポップアップ リスト ボックス、またはリスト ボックスで選択されている項目のインデックスを返します。
OraFormsParseListEntries	TrueLog Explorer の視覚的解析、スクリプト記述	リスト項目、コンボ ボックス、またはポップアップ リスト項目の要素数を返します。
OraFormsParseListEntry	スクリプト記述	指定されたインデックスにあるコンボ ボックス、リスト項目、ポップアップ リスト項目の値を返します。

Oracle Forms 検証関数

Oracle Forms 関数	用途	説明
OraFormsVerifyTextValue	TrueLog Explorer の視覚的検証、スクリプト記述	テキスト フィールドまたはテキスト領域を検証します。
OraFormsVerifyCheckValue	TrueLog Explorer の視覚的検証、スクリプト記述	チェック ボックスがオンかオフかを検証します。
OraFormsVerifyRadioValue	TrueLog Explorer の視覚的検証、スクリプト記述	オプション ボタンの値を検証します。
OraFormsVerifyListValue	TrueLog Explorer の視覚的検証、スクリプト記述	コンボ ボックス、リスト ボックス、またはポップアップ リスト ボックスの選択値を検証します。
OraFormsVerifyListIndex	TrueLog Explorer の視覚的検証、スクリプト記述	リスト コントロール内の選択項目を検証します。
OraFormsVerifyListEntries	スクリプト記述	コンボ ボックス、リスト ボックス、またはポップアップ リスト内のリスト エントリの数を検証します。
OraFormsVerifySelectedTreeItem	スクリプト記述	指定値を持つツリー コントロール内の選択されているツリー ノードを検証します。

Oracle Forms 関数	用途	説明
OraFormsVerifyPropString	スクリプト記述	プロパティ値を検証します。
OraFormsVerifyPropInt	スクリプト記述	指定されたコントロールの整数値プロパティを検証します。
OraFormsVerifyPropByte	スクリプト記述	バイト プロパティを検証します。
OraFormsVerifyPropPoint	スクリプト記述	点プロパティを検証します。
OraFormsVerifyPropRectangle	スクリプト記述	矩形プロパティを検証します。
OraFormsVerifyPropNull	スクリプト記述	プロパティ値が NULL かどうかを検証します。
OraFormsVerifyPropStringArray	スクリプト記述	配列内のすべての文字列を検証します。
OraFormsVerifyPropByteArray	スクリプト記述	バイト配列内のすべての要素を検証します。Oracle Forms のバイト配列型は、主に、サーバーからクライアントに画像を転送するために使用されます。

SAPGUI の解析関数と検証関数

SAPGUI 解析関数

SAPGUI 関数	用途	説明
SapGuiGetStatusBarText	TrueLog Explorer の視覚的解析、スクリプト記述	ステータス バー コントロールの値を返します。
SapGuiGetText	TrueLog Explorer の視覚的解析、スクリプト記述	コントロールの値を返します。
SapGuiGetComboboxEntry	TrueLog Explorer の視覚的解析、スクリプト記述	コンボ ボックス コントロールの選択値を返します。
SapGuiIsCheckboxChecked	TrueLog Explorer の視覚的解析、スクリプト記述	チェック ボックス コントロールの状態を返します。
SapGuiIsRadioButtonSelected	TrueLog Explorer の視覚的解析、スクリプト記述	オプション ボタン コントロールの状態を返します。

SAPGUI 検証関数

SAPGUI 関数	用途	説明
SapGuiVerifyCheckbox	TrueLog Explorer の視覚的検証、スクリプト記述	チェック ボックス コントロールの値を検証します。

SAPGUI 関数	用途	説明
SapGuiVerifyTextValue	TrueLog Explorer の視覚的検証、スクリプト記述	コントロールの値を検証します。
SapGuiVerifyRadioValue	TrueLog Explorer の視覚的検証、スクリプト記述	オプション ボタン コントロールの状態を検証します。
SapGuiVerifyStatusBarText	TrueLog Explorer の視覚的検証、スクリプト記述	ステータス バーのテキストを検証します。
SapGuiVerifyComboboxEntry	TrueLog Explorer の視覚的検証、スクリプト記述	コンボ ボックス コントロールの選択値を検証します。

Citrix の解析関数と検証関数

Citrix 解析関数

Citrix 関数	用途	説明
CitrixParseText	TrueLog Explorer の視覚的解析、スクリプト記述	指定された画面領域に表示されるテキストを返します。

Citrix 検証関数

Citrix 関数	用途	説明
CitrixVerifyText	TrueLog Explorer の視覚的解析、スクリプト記述	指定された画面領域に表示されるテキストを検証します。

ターミナル エミュレーションの解析関数と検証関数

ターミナル エミュレーション解析関数

ターミナル エミュレーション関数	用途	説明
WebTelnetScreenGetText	TrueLog Explorer の視覚的解析、スクリプト記述	アクティブな Telnet 接続の表示画面の内容を照会します。
WebTelnetScreenGetField	TrueLog Explorer の視覚的解析、スクリプト記述	アクティブな Telnet 接続の表示画面のフィールド値を照会します。
WebTelnetScreenGetStatus	TrueLog Explorer の視覚的解析、スクリプト記述	アクティブな Telnet 接続の表示画面のステータス値を照会します。

ターミナル エミュレーション検証関数

ターミナル エミュレーション関数	用途	説明	
WebTelnetScreenVerifyText	TrueLog Explorer の視覚的検証、スクリプト記述	アクティブな Telnet 接続の表示画面の内容を検証します。	
WebTelnetScreenVerifyField	TrueLog Explorer の視覚的検証、スクリプト記述	アクティブな Telnet 接続の表示画面のフィールド内容を検証します。	
WebTelnetScreenVerifyStatus	TrueLog Explorer の視覚的検証、スクリプト記述	アクティブな Telnet 接続の表示画面のステータス値を検証します。	

索引

数字

A

B

C

D

G

H

I

J

M

O

P

R

S

T

U

W

X

あ

こ

さ

し

す

せ

そ

た

ち

つ

て

と

に

ね

の

は

ひ

ふ

へ

ほ

ま

め

も

ゆ

よ

ら

れ

ろ

わ