DevTest Solutions

リファレンス

バージョン 8.0

ca technologies

CA への連絡先

テクニカル サポートの詳細については、弊社テクニカル サポートの Web サイト（http://www.ca.com/jp/support/）をご覧ください。

目次

第 1 章: テスト ケースのリファレンス

このセクションには、以下のトピックが含まれています。

アサーションの説明

このセクションでは、DevTest で使用可能な各アサーションについて説明します。

正規表現は、多くのアサーションで比較を行うために使用されます。 正規表現の詳細については、「正規表現」を参照してください。

このセクションには、以下のアサーションの説明が含まれています。

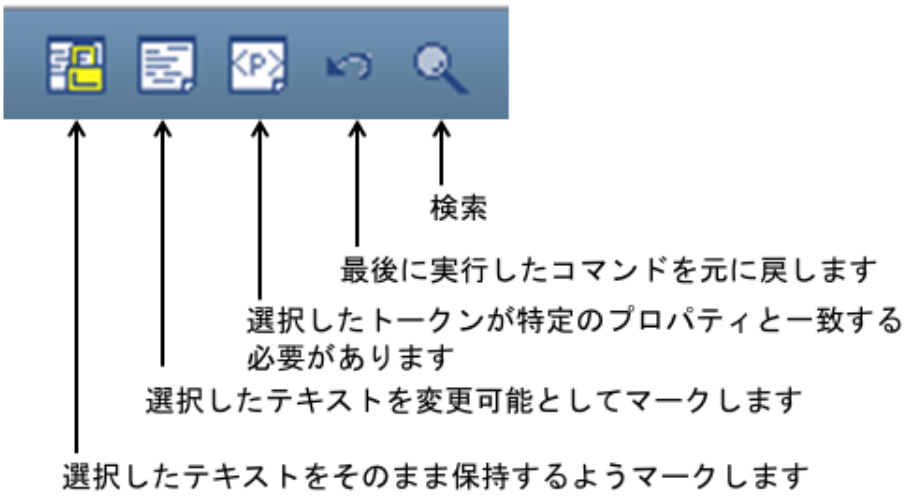
HTTP アサーション

以下のアサーションは、任意のテスト ステップの HTTP アサーションのリストで使用できます。

比較する HTML コンテンツを強調表示

比較する HTML コンテンツを強調表示アサーションでは、比較を、HTML ページのコンテンツに基づいたものにすることができます。 このアサーションは、HTML ページで動作するように設計された「画面のペイント」技術を使用します。たとえば、大きな HTML ドキュメントがある場合、「対象のコンテンツ」の前後のデータを特定できます。 これにより、「対象のコンテンツ」が何と比較されるかが特定されます（通常、データ セットで提供される期待値）。

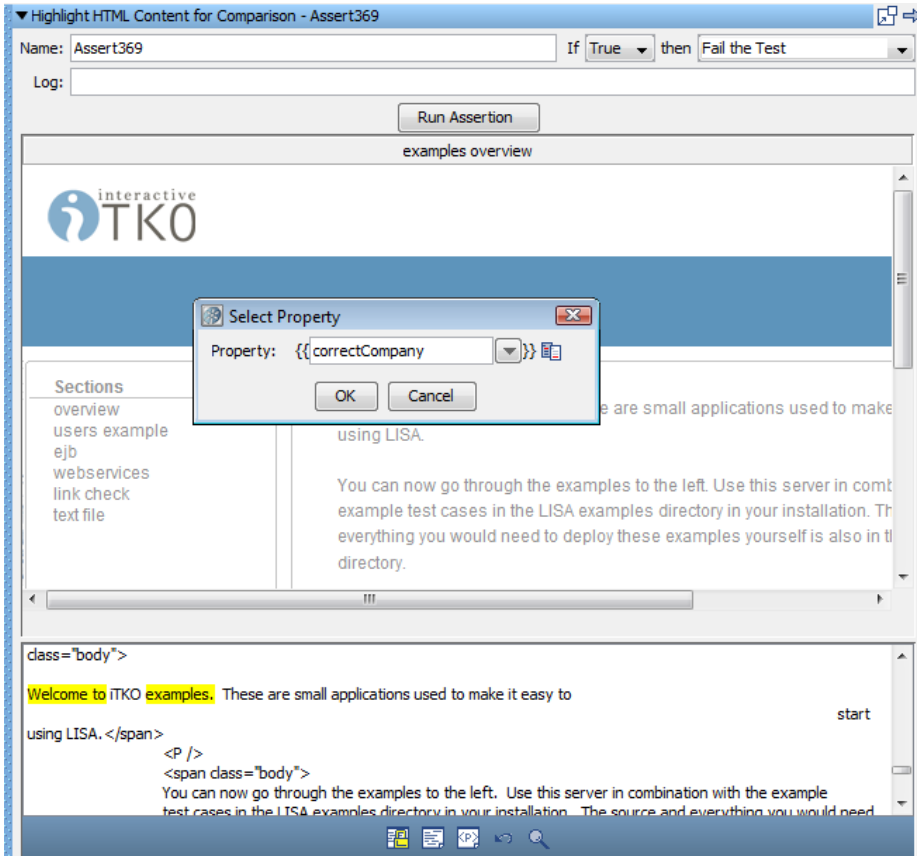
テキストは、エディタの下部のアイコンを使用してマークされます。



この技術について、例を用いて説明します。

以下の例では、「Welcome to ITKO examples」というテキストに表示される会社名（現在は ITKO）が、指定されたプロパティの値と一致するかどうかを確認します。 上記に示したボタンを使用して、テキストを選択し、適切なアイコンをクリックすることにより、テキストをマークしています。

- 黄色の背景色は、そのまま表示される必要があるテキストを示します。
- 白の背景色は、存在する必要がないか変更可能なテキストを示します。
- 赤の背景色は、ダイアログ ボックスに入力されたプロパティに一致する必要があるテキストを識別します。



この画面では、上部パネルのブラウザに HTML が表示され、下部パネルに実際の HTML テキストが表示されています。「Welcome to」と「examples」を必須にします。 それらのテキストの境界を設定し、保持テキスト アイコン をクリックします。 次に、強調表示されたコンテンツの内部の会社名のテキスト「ITKO」を選択し、プロパティ一致アイコン をクリックします。 ダイアログ ボックスにプロパティ名「correctCompany」を入力します。 このプロパティは、境界が設定された 2 つのテキストの間に表示されているテキストと比較されます。 会社名のテキストは、プロパティの名前と置き換えられます。

アサーションを実行するには、［アサーションの実行］ボタンをクリックします。

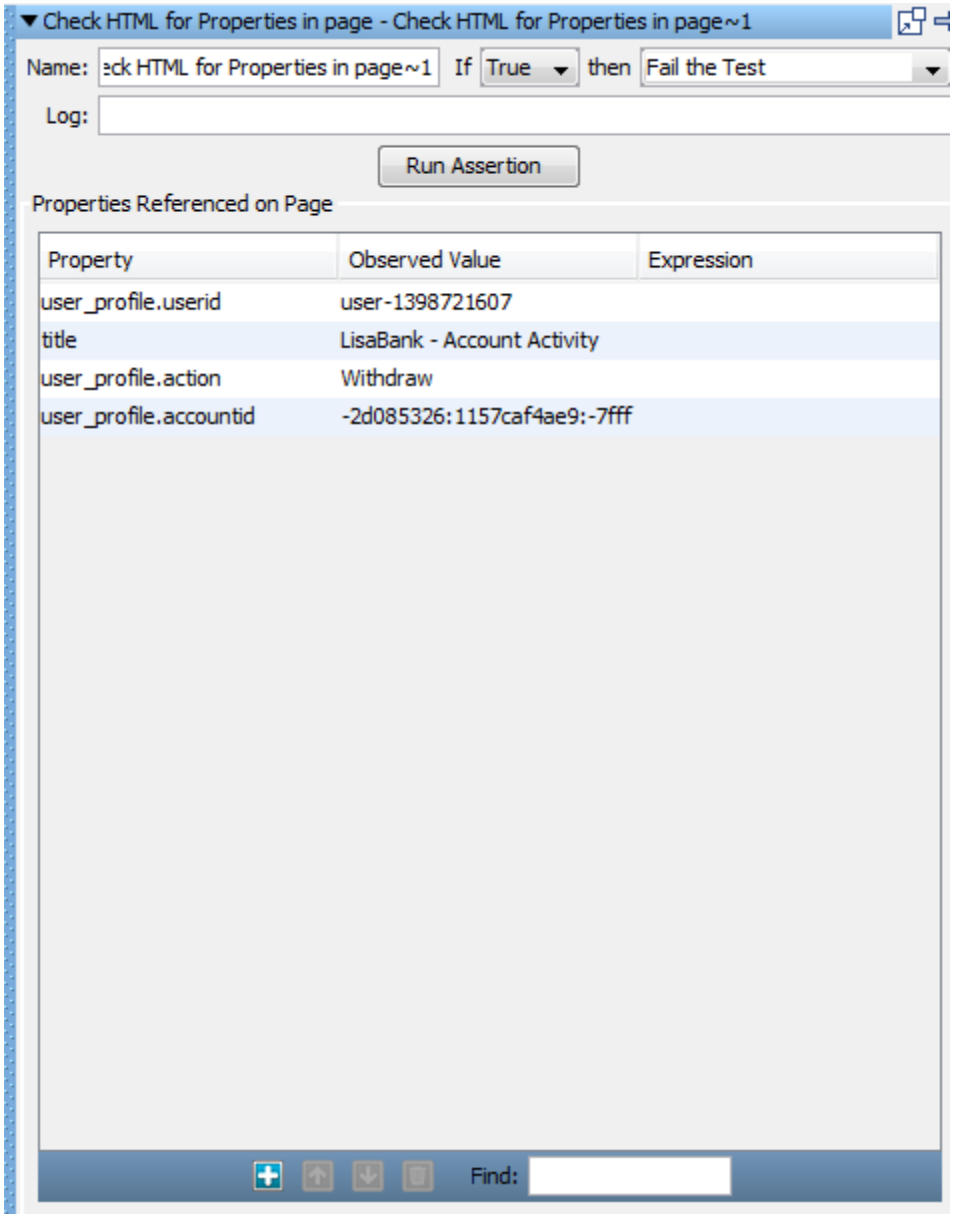
このアサーションが実行されると、correctCompany プロパティの値が、「Welcome to」と「examples」の間に挿入されます。 結果として作成されたテキストが、HTML 応答の対応するテキストと比較されます。「Welcome to correctCompany examples」というテキストの HTML 内での場所は変更でき、変更しても検出できます。

HTML のページ内プロパティの確認

アサーションで使用される可能性があるプロパティ データが Web ページに含まれている場合は、HTML のページ内プロパティの確認アサーションを使用します。 プロパティ データは、以下の項目について Web ページを解析することにより、アサーションで使用可能になります。

- メタ タグ
- タイトル タグ
- 非表示のフォーム フィールド
- 製品が自動的に解析できるその他のタグ（<lisaprop> タグ、DevTest Integration API など）

使用可能なプロパティのテーブルの例を以下に示します。



以下のパラメータを入力します。

名前

アサーションの名前を定義します。

条件:

ドロップダウン リストからアサーションの動作を指定します。

次のステップ

アサーションが起動された場合のリダイレクト先のステップを指定します。

ログ

アサーションが起動された場合に出力するイベント テキストを指定します。

アサーションを実行するには、［アサーションの実行］をクリックします。

注: HTML 結果のタグの解析フィルタをインストールするように促される場合があります。

HTTP ヘッダに式が含まれていることを確認

HTTP ヘッダに式が含まれていることを確認アサーションでは、特定の HTTP 結果ヘッダに、指定した正規表現に一致するフィールドが含まれていることを確認できます。

以下のパラメータを入力します。

名前

アサーションの名前を定義します。

条件:

ドロップダウン リストからアサーションの動作を指定します。

次のステップ

アサーションが起動された場合のリダイレクト先のステップを指定します。

ログ

アサーションが起動された場合に出力するイベント テキストを指定します。

ヘッダ フィールド

ヘッダ フィールドの名前。

正規表現

ヘッダ フィールドに含まれている必要がある正規表現。

アサーションを実行するには、[アサーションの実行]をクリックします。

HTTP 応答コードのチェック

HTTP 応答コードのチェックアサーションでは、HTTP 応答コードが指定した正規表現に一致することを確認できます。

以下のパラメータを入力します。

名前

アサーションの名前を定義します。

条件:

ドロップダウン リストからアサーションの動作を指定します。

次のステップ

アサーションが起動された場合のリダイレクト先のステップを指定します。

ログ

アサーションが起動された場合に出力するイベント テキストを指定します。

正規表現

応答コードに含まれている必要がある正規表現。たとえば、HTTP 応答コードが 400 から 499 の範囲にあることを確認するには、［正規表現］を「4¥d¥d」に設定します。

アサーションを実行するには、［アサーションの実行］をクリックします。

シンプル Web アサーション

シンプル Web アサーションは、Web アプリケーションからリターン コードを読み取ります。

アプリケーションがリターン コードの 404 （ページが見つかりません）、500（サーバ エラー）、またはその他のエラーを返す場合、このアサーションは true を返します。

examples プロジェクトの multi-tier-combo テスト ケースには、このタイプのアサーションがあります。

以下のパラメータを入力します。

名前

アサーションの名前を定義します。

条件:

ドロップダウン リストからアサーションの動作を指定します。

次のステップ

アサーションが起動された場合のリダイレクト先のステップを指定します。

ログ

アサーションが起動された場合に出力するイベント テキストを指定します。

アサーションを実行するには、[アサーションの実行] をクリックします。

Web 応答のリンクを確認

Web 応答のリンクを確認アサーションは、返された Web ページ上のすべてのリンクを確認して、有効なページが含まれていて 404 エラーやその他の HTTP エラーが返されないことを確認します。 このアサーションは、通常、リンクがアプリケーションにわたって正しく動作しており、ページに非アクティブなリンクがないことを確認するために使用されます。

以下のパラメータを入力します。

名前

アサーションの名前を定義します。

条件:

ドロップダウン リストからアサーションの動作を指定します。

次のステップ

アサーションが起動された場合のリダイレクト先のステップを指定します。

ログ

アサーションが起動された場合に出力するイベント テキストを指定します。

リンクに対して以下の条件を確認できます。

同じドメインのリンクのみ確認

返された Web ページの現在のドメインのリンクのみを確認します。

クエリ文字列を含める

返された Web ページにクエリ文字列が存在する場合、それらを確認します。

アンカーを含める(<_a>)

現在の Web ページのすべてのアンカー リンクが確認されます。

イメージを含める

返された Web ページ上のイメージがすべて確認されます。

アセットを含める(<_link> & <_script>)

現在の Web ページのスクリプトおよびリンクが確認されます。

正規表現に一致するリンクをスキップ

スキップするリンクの正規表現を入力します。

データベース アサーション

以下のアサーションは、任意のテスト ステップのデータベース アサーションのリストで使用できます。

結果セット サイズを確認 (P. 20)
結果セットに式が含まれていることを確認 (P. 21)

結果セット サイズを確認

結果セット サイズを確認アサーションは、結果セット内の行数を数え、上限および下限の値の範囲内にそのサイズが収まることを確認します。

このアサーションの例としては、HTML テーブルの行数がデータ セットからの有効な値と一致するかどうかを確認するチェックなどがあります。

以下のパラメータを入力します。

名前

アサーションの名前を定義します。

条件:

ドロップダウン リストからアサーションの動作を指定します。

次のステップ

アサーションが起動された場合のリダイレクト先のステップを指定します。

ログ

アサーションが起動された場合に出力するイベント テキストを指定します。

結果セットが警告を含む

確認された場合、データベースは結果セット内の警告を返します。データベースが結果セットで警告をサポートしているかどうかを判断するには、システム管理者に問い合わせてください。

行数 >=

結果セット内の最小行数。-1 は最小値が設定されていないことを示します。

行数 <_=

結果セット内の最大行数。-1 は最大値が設定されていないことを示します。

アサーションを実行するには、[アサーションの実行] をクリックします。

たとえば、データベース アサーション ステップが 1 つ（1 つだけ）の行を返すようにするには、[行数 >=] フィールドに「1」を設定し、[行数 <=] フィールドに「1」を設定します。

結果セットに式が含まれていることを確認

結果セットに式が含まれていることを確認アサーションは、結果セットの特定の列を確認し、指定された式が少なくとも 1 つの値に一致することを確認します。

以下のパラメータを入力します。

名前

アサーションの名前を定義します。

条件:

ドロップダウン リストからアサーションの動作を指定します。

次のステップ

アサーションが起動された場合のリダイレクト先のステップを指定します。

ログ

アサーションが起動された場合に出力するイベント テキストを指定します。

列

確認するテキストが含まれる列。 この値は列名またはインデックスです。

正規表現

列と比較する正規表現。

アサーションを実行するには、[アサーションの実行] をクリックします。

たとえば、クエリから返された行のうち少なくとも 1 行が「wp」で始まるログインの値を持つことを確認するには、［列］フィールドを「login」に設定し、［正規表現］フィールドを「wp.*」に設定します。

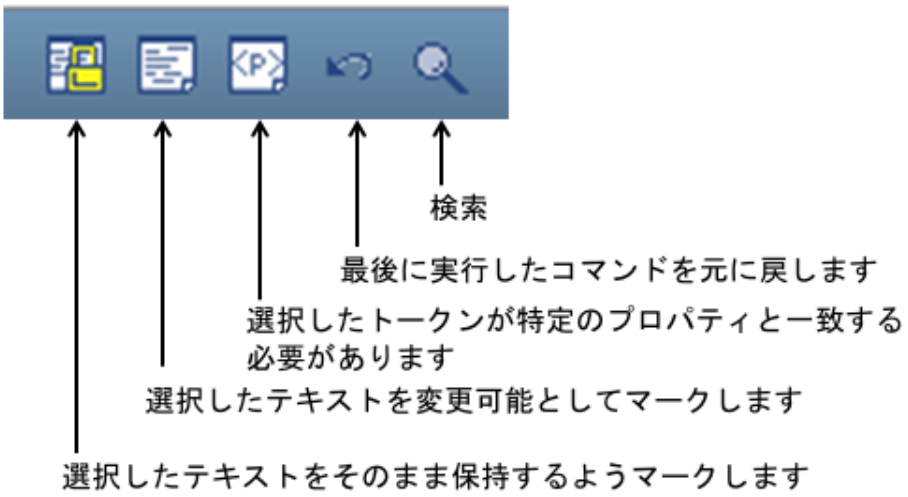
XML アサーション

以下のアサーションは、任意のテスト ステップの XML アサーションのリストで使用できます。

テキスト コンテンツの強調表示による比較

テキスト コンテンツの強調表示による比較アサーションは、HTML ページで動作するよう特別に設計された「画面のペイント」技術を使用します。たとえば、大きな HTML ドキュメントがある場合、ユーザが対象のコンテンツの前後のデータを特定します。 これにより、対象のコンテンツが何と比較されるかが特定されます（通常、このコンテンツはデータ セットで提供される期待値）。

エディタの下部のアイコンを使用してテキストをマークします。

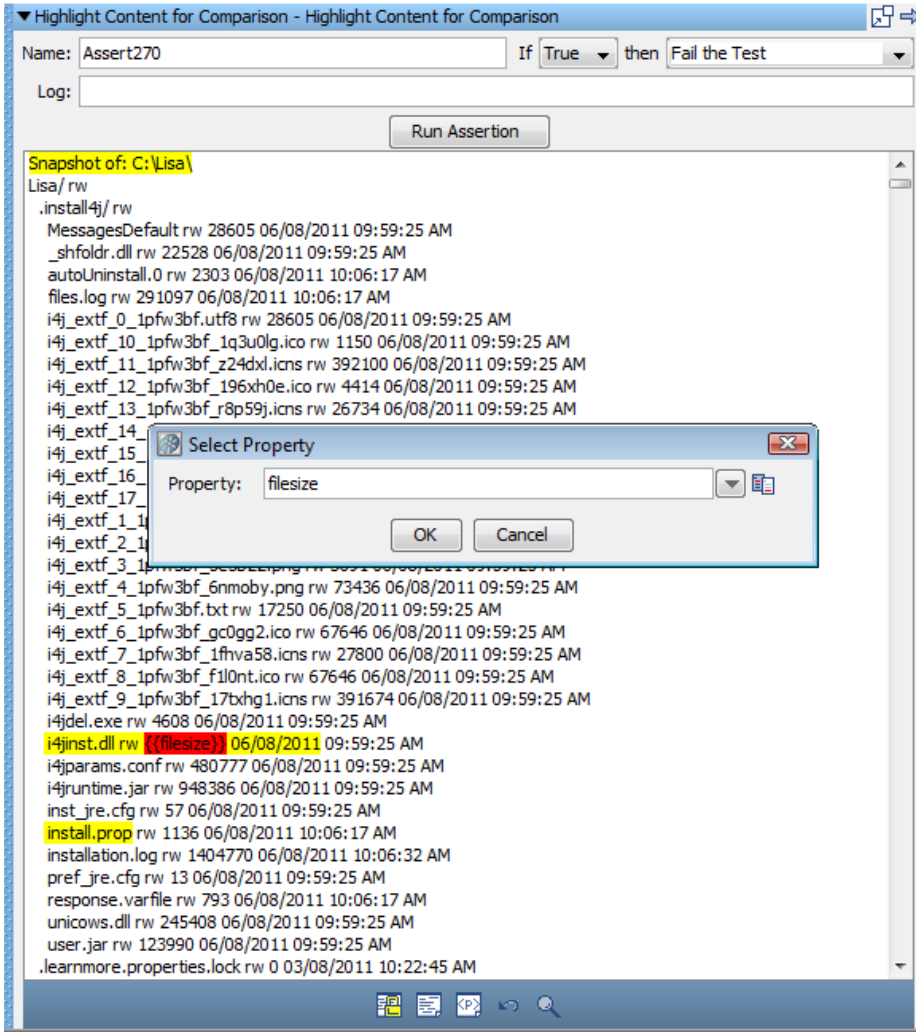


以下の例の目的は、次のとおりです。

- バッファに特定のファイルが含まれていることを確認する。
- いずれかのファイルのサイズをプロパティの値と比較する。

上記の図に示された 3 つのボタンを使用して、テキストを選択し、適切なアイコンをクリックすることにより、テキストをマークしています。

- 黄色の背景色は、そのまま表示される必要があるテキストを示します。
- 白の背景色は、存在する必要がないか変更可能なテキストを示します。
- 赤の背景色は、ダイアログ ボックスに入力されたプロパティに一致する必要があるテキストを識別します。



上記の図に示されたトークンのセットは、以下のように解釈できます。

- バッファは、黄色で示された「Snapshot of: C:¥Lisa¥」というテキストで始まる必要があります。
- 次のトークンでは、多くのファイルがバッファにある場合と、ない場合があります。 次のトークンは「変更可能テキスト」トークンであるため、不一致は重要ではありません。
- 「i4jinst.dll」ファイルおよび「rw」属性が存在する必要があります。

  赤で示された「filesize」は、プロパティ キー filesize と関連付けられた値が式で変換され、その後に比較が実施されることを意味します。
- 「06/08/2011」というテキストが存在する必要があります。
- 「install.prop」ファイルが存在する必要があります。

- それ以後、バッファは任意の量のコンテンツを持つことができます。

マークアップを完了した後、以下のパラメータを入力します。

名前

アサーションの名前を定義します。

条件:

ドロップダウン リストからアサーションの動作を指定します。

次のステップ

アサーションが起動された場合のリダイレクト先のステップを指定します。

ログ

アサーションが起動された場合に出力するイベント テキストを指定します。

アサーションを実行するには、[アサーションの実行]をクリックします。

注: 「保持テキスト」ブロックは、常に「プロパティ一致」ブロックの両側にある必要があります。

結果が文字列を含むことを確認

結果が文字列を含むことを確認アサーションでは、（テキストとして）応答内の文字列を検索できます。

以下のパラメータを入力します。

名前

アサーションの名前を定義します。

条件:

ドロップダウン リストからアサーションの動作を指定します。

次のステップ

アサーションが起動された場合のリダイレクト先のステップを指定します。

ログ

アサーションが起動された場合に出力するイベント テキストを指定します。

含む文字列

ステップ結果で検索する文字列 - この文字列にはプロパティを含めることができます。

ステップ応答時間を確認

ステップ応答時間を確認アサーションでは、応答時間の上限と下限を定義し、応答時間がそれらの範囲内であることをアサートできます。

以下のパラメータを入力します。

名前

アサーションの名前を定義します。

条件:

ドロップダウン リストからアサーションの動作を指定します。

次のステップ

アサーションが起動された場合のリダイレクト先のステップを指定します。

ログ

アサーションが起動された場合に出力するイベント テキストを指定します。

想定最小時間(ミリ秒)

下限をミリ秒で入力します。

想定最大時間(ミリ秒)

上限をミリ秒で入力します。 **-1** に設定されている場合、この値は無視されます。

注: パラメータには、プロパティを含めることができます。

グラフィカル XML 比較

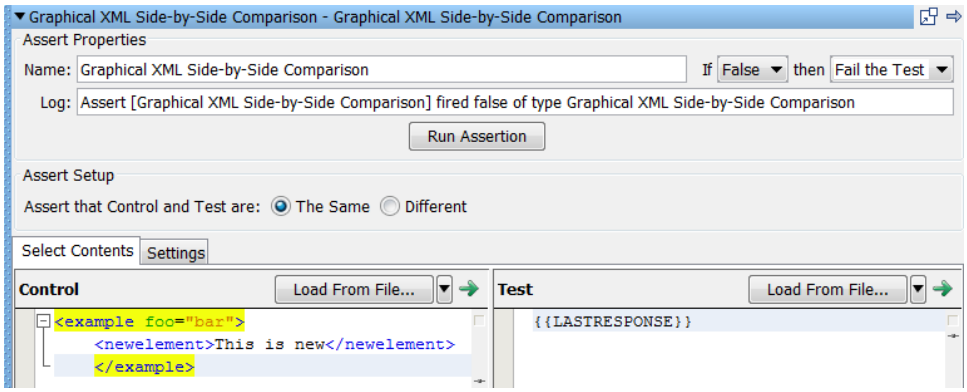
グラフィカル XML 比較アサーションでは、テストから受信したテスト XML 値を XML 制御値と比較できます。 応答が同じでも異なっていても、アサーションは ture を返すことができます。 このアサーションにより、ビジネス プロセスのさまざまなステップで、期待される基準に一致するかどうかを確認するために XML ドキュメントを柔軟に比較することが可能になります。 この方法は「排他的」テストとも言い、変化が予期される一部の値を除いて応答全体を比較します。

アサーション エディタは、XML の左側と右側を互いに比較することにより動作します。左側は、制御コンテンツと呼ばれます。右側は、テスト コンテンツと呼ばれます。 たとえば、制御コンテンツは、テスト中のアプリケーションの Web サービスから返される予期された XML です。テスト コンテンツは、実際のコンテンツです。 デフォルトでは、テスト コンテンツは、空のプロパティ キーで示され、アサーションと関連付けられたテスト ステップの最終応答からロードされます。 そうでない場合は、任意の有効なプロパティ キーが使用でき、テスト コンテンツはそれからロードされます。

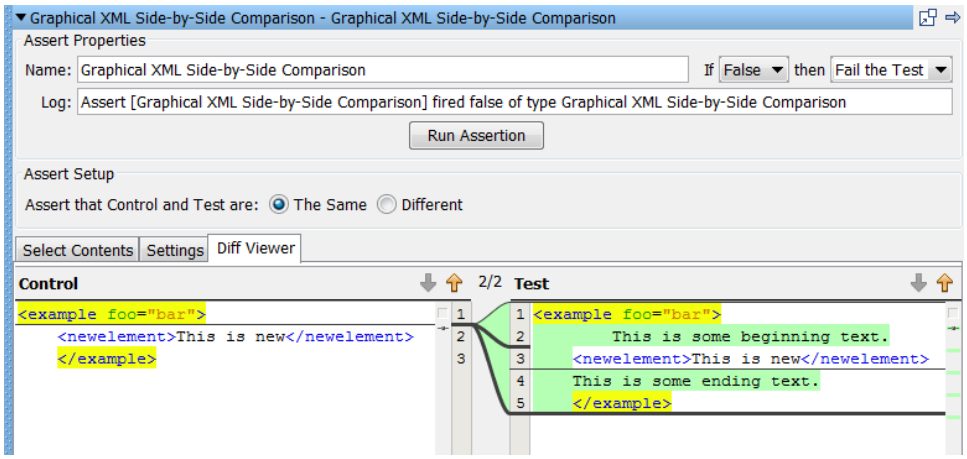
また、テスト ケース オーサリング モードで以下のいずれかのアクションを実行することにより、グラフィカル比較を迅速に実行できます。

- XML をファイルからロードする。
- 制御コンテンツまたはテスト コンテンツに対して XML を手動で入力する。

比較を実行するには、緑の矢印  をクリックします。



比較の実行後、アサーション エディタの［差分ビューア］タブのビジュアライザに結果が表示されます。



実行中の出力

アサーションが実行されると、DevTest は、差分結果をテスト イベントとしてログに記録します。

XML 差分結果が含まれる情報メッセージのイベント ID は、常にログに記録されます。

アサーションが起動されると、XML 差分結果が含まれるアサーションの起動イベント ID がログに記録されます。

差分結果は、オリジナルの UNIX diff ユーティリティと似た形式でレポートされます。 以下に、テキスト差分レポートの例を示します。

```
Assert [Assert1] fired false of type Graphical XML Diff Assertion
XML is [Different]
=====
1,2[ELEMENT_NAME_CHANGED]1,2
<! <test2>
<! </test2>
---
>! <test>
>! </test>
```

各差分は、以下の形式の見出し付きで表示されます。

```
<First Start Line>, <First End Line>'['<Diff Type>']'<Second Start Line>,<Second End Line>
```

最初のコンテンツの差分の末尾には区切り文字「---」が表示され、その後に 2 番目のコンテンツの差分が表示されます。

+ 文字は追加を示し、- 文字は削除を示し、! 文字は 変更を示します。 これらの文字が存在する場合、コンテンツの行に（コンテキスト行の代わりに）実際の変更が発生したことを示します。

XML 比較オプション

以下の比較オプションを差分エンジンで使用できます。

全般

大文字と小文字を区別

比較時に、大文字と小文字を区別するかどうか（デフォルトでは有効）。

空白

前後の空白を削除する

比較時に、エレメント テキストおよび属性値からすべての先頭および末尾の空白が削除されます（デフォルトでは有効）。

Collapse whitespace（空白を整理する）

空白の削除に加えて、テキスト内部の 1 つ以上の連続した空白文字が単一の空白文字に変換されます。

空白を正規化する

1 つ以上の連続した空白文字が単一の空白文字に変換されます。

すべての空白を無視する

すべての空白が比較時に無視されます。

ネームスペース

ネームスペースを無視する

エレメントまたは属性のネームスペース値が無視されます。

ネームスペース プレフィックスを無視する

エレメントまたは属性のネームスペース プレフィックスが無視されます（デフォルトでは有効）。

順序

子エレメントの順序を無視する

XML ドキュメント内の子エレメントの順序を無視します。

属性の順序を無視する

XML ドキュメント内の属性の順序を無視します（デフォルトでは有効）。

ノード タイプ

エレメント テキストを無視する

エレメント テキストをすべて無視します。

属性値を無視する

属性名を比較しますが、属性値を無視します。

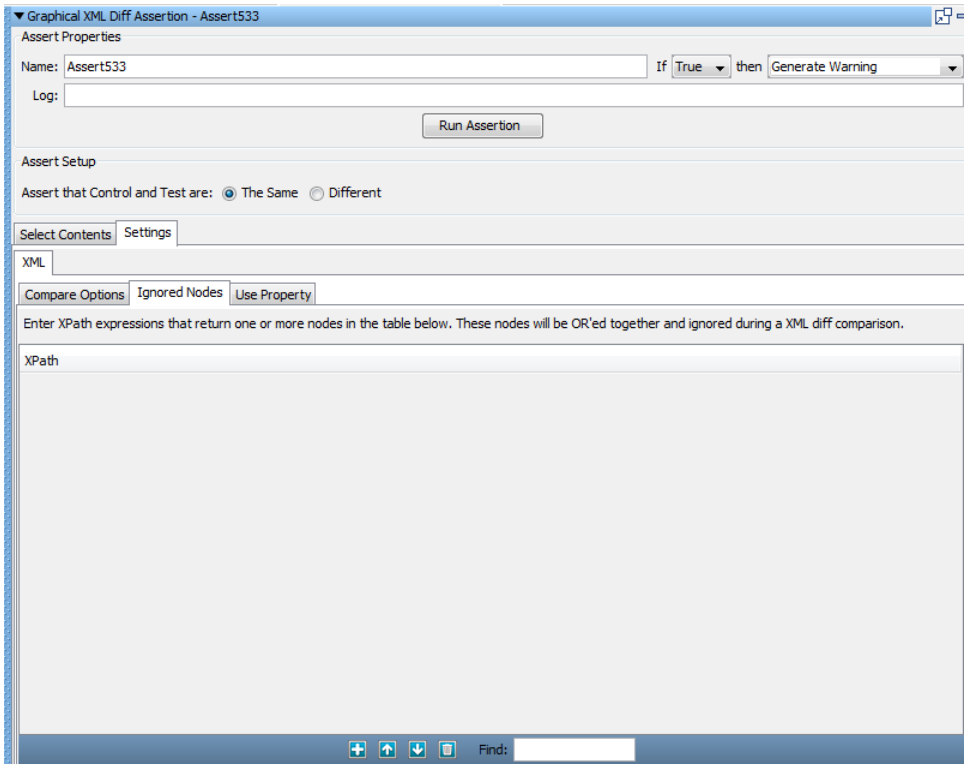
属性を無視する

属性の名前と値を無視します。

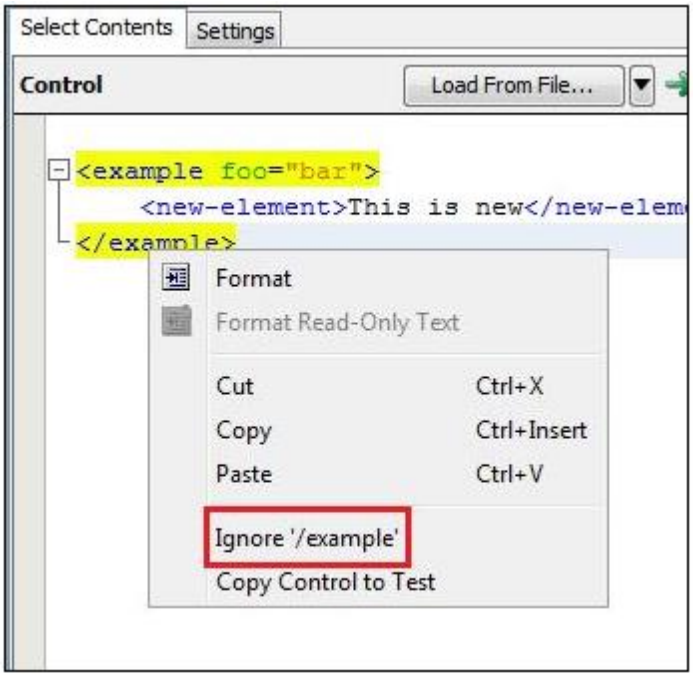
除外ノード

無視されたノードは、左側および右側のドキュメントに対して実行される XPath のリストから作成されます。 ノード セットを返す評価された XPath が集約されます。 差分が発生すると、集約セット内に見つかるノードは無視されます。

無視されたノードの XPath は、ノード セットを返す任意のクエリである場合があります。 たとえば、XPath **//*** は、XML ドキュメント内のすべてのノードを除外します。 **/example/text()** は、XML ドキュメント内の **example** エレメントの子の最初のテキスト ノードを除外します。 **/example/@myattr** は、XML ドキュメント内の **example** エレメントに属する、属性テキスト値を含む **myattr** 属性を無視する XPath です。



右クリック メニュー項目でも、XML ドキュメントの内部で直接ノードを選択でき、その XPath は［除外ノード］リストに追加されます。



XML XPath アサーション

XML XPath アサーションでは、応答で実行される XPath クエリを使用できます。このアサーションが選択されている場合、最終応答はコンテンツ パネルにロードされます。

XPath は、「XML 用の SQL」と考えることができます。XPath は XML の解析を簡単にする強力なクエリ言語です。 XPath アサーションは、結果全体の特定の文字列を単に解析するのではなく、より洗練された方法で Web サービス応答を検証するのに役立ちます。 たとえば、2 番目および 3 番目の項目に「ITKO」が含まれ、それらの行の項目の値が 10 を超えることを確認できます。

応答は XML ドキュメントまたは DOM ツリーとして表示できます。ただし、XPath の選択は DOM ツリー ビューからしか行えません。

以下の方法で XPath クエリを作成できます。

- ［XPath クエリ］テキスト ボックスに手動で XPath 式を入力する。
- DOM ツリーからエレメントを選択し、DevTest によって XPath 式が作成されるようにする。
- DOM ツリーからエレメントを選択し、DevTest が作成する XPath を編集する。 たとえば、プロパティやカウンタ データ セットを使用するよう変更できます。

以下のパラメータを入力します。

名前

アサーションの名前を定義します。

条件:

ドロップダウン リストからアサーションの動作を指定します。

次のステップ

アサーションが起動された場合のリダイレクト先のステップを指定します。

ログ

アサーションが起動された場合に出力するイベント テキストを指定します。

上記の方法のいずれかを使用して、XPath クエリを作成します。

XPath クエリを作成したら、パネルの上部の［アサーションの実行］ボタンをクリックして XPath クエリをテストします。クエリの結果は、［クエリ結果］パネルに表示されます。

上記の例では、<wsdl：part> タグの 4 番目の出現を使用します。

一般的なユース ケースは、Web サービス結果の XPath ノードを選択し、ノードをテキスト値と比較することです。 その後、応答に期待値が含まれていることをアサートできます。 この一般的なユース ケースでは、提供された初期の式の末尾に等価演算子を追加できます。 たとえば、私たちが応答で返された新しいパスワードを選択する場合（BobPass）：

DevTest は以下の XPath 式を作成します。

```
string(/env:Envelope/env:Body/ns2:updatePasswordResponse/[name(
)='return']/[name()='pwd'])=
```

「= 'NewPassword'」を追加すると（以下の例のように）、結果の文字列が、新しいパスワードを設定するために使用したプロパティの値と比較されます。 等価テストが一致しない場合、アサーションは失敗します。

```
string(/env:Envelope/env:Body/ns2:updatePasswordResponse/[name(
)='return']/[name()='pwd'])='NewPassword'
```

この式は、Web サービス応答を確認して、新しいパスワードを探し、それが予期したものと一致することを確認します。

XML 妥当性検証

XML 妥当性検証アサーションでは、XML ドキュメントを検証できます。XML ドキュメントが整形されているかどうかを確認できます。文書型定義（DTD）、または 1 つ以上のスキーマに対して検証できます。 XML フラグメントがある場合、DevTest に XML 宣言タグを追加させることができます。また、DevTest が警告をエラーとして報告するように指定することもできます。 プロパティとして検証する XML を入力します。

以下のパラメータを入力します。

名前

アサーションの名前を定義します。

条件:

ドロップダウン リストからアサーションの動作を指定します。

次のステップ

アサーションが起動された場合のリダイレクト先のステップを指定します。

ログ

アサーションが起動された場合に出力するイベント テキストを指定します。

Source

XML が含まれるプロパティ。 このフィールドが空白のままの場合、最終応答が使用されます。

検証

以下の複数の検証オプションを選択できます。

整形式 XML

XML が整形されていることを確認します。

DTD 準拠

DTD への準拠を確認します。

スキーマ

1 つ以上のスキーマへの準拠を確認します。

XML フラグメント

XML がフラグメントである場合、XML 宣言が XML フラグメントの先頭に追加されます。

警告をエラーとして扱う

警告がエラーとしてレポートされます。

すべてのスキーマの場所を使用

同じネームスペースに対して複数のインポートをしている場合、このオプションは最初のスキーマだけでなく、各スキーマの場所を開きます。

アサーションを実行するには、[アサーションの実行]をクリックします。

[検証]タブ

[検証の実行] ボタンをクリックして、検証を実行できます。検証エラー結果は、検証エラー リストに表示されます。[検証タイプ] オプション ボタンを使用して、エラーを処理する方法を選択できます。

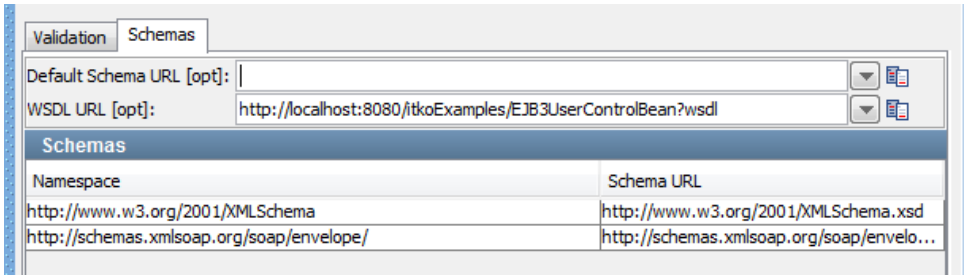
許可されているエラーはありません

検証はすべてのエラーで失敗します。

エラー メッセージ式

検証で無視されるエラーをマークできます。このオプションを選択する場合、無視されるエラーを [検証エラー リスト] で選択できます。

[スキーマ]タブ



検証で使用する各スキーマの情報を入力します。また、デフォルト スキーマを指定できます。

デフォルト スキーマ URL

必要に応じて、スキーマのデフォルト URL を指定します。

WSDL URL

必要に応じて、WSDL の URL を指定します。

JSON アサーション

以下のアサーションは、任意のテスト ステップの JSON アサーションのリストで使用できます。

結果が等しいことを確認 (P. 38)
結果が含むことを確認 (P. 39)
JSON スキーマの確認 (P. 41)

結果が等しいことを確認

結果が等しいことを確認アサーションでは、JSON パス結果が期待値に等しいことを確認できます。

以下のパラメータを入力します。

名前

アサーションの名前を定義します。

条件:

ドロップダウン リストからアサーションの動作を指定します。

次のステップ

アサーションが起動された場合のリダイレクト先のステップを指定します。

ログ

アサーションが起動された場合に出力するイベント テキストを指定します。

JSON パス

JSON ドキュメント内の JSON プロパティのシーケンスから構成される式を指定します。 JSON パスは、送信先 JSON プロパティへのパスを表します。

Expected value

JSON パス結果の期待値を定義します。 配列は順序リストであるため、両方の配列内の要素の順序が同じ場合、2 つの配列は等しいと見なされます。

アサーションの実行

フィルタを実行するには、[アサーションの実行]をクリックします。

結果が含むことを確認

結果が含むことを確認アサーションでは、JSON オブジェクトまたは JSON 配列のいずれかである JSON パス結果に、指定したリストの一部またはすべてが含まれることを確認できます。

以下のパラメータを入力します。

名前

アサーションの名前を定義します。

条件:

ドロップダウン リストからアサーションの動作を指定します。

次のステップ

アサーションが起動された場合のリダイレクト先のステップを指定します。

ログ

アサーションが起動された場合に出力するイベント テキストを指定します。

JSON パス

JSON ドキュメント内の JSON プロパティのシーケンスから構成される式を指定します。 JSON パスは、送信先 JSON プロパティへのパスを表します。

Contains all expected values または Contains any expected values

JSON パスと期待値とを一致させるパラメータを定義します。［Contains all expected values］チェック ボックスをオンにした場合、アサーションが成功するためには、［Expected value］リスト内の値がすべて JSON パス結果に含まれている必要があります。 ［Contains any expected values］チェック ボックスをオンにした場合、アサーションが成功するためには、［Expected value］リスト内の値が少なくとも 1 つ JSON パス結果に含まれている必要があります。

Expected value

JSON パス結果に含まれる必要がある値のリストを定義します。［Expected value］フィールドの値の編集を停止するには、Ctrl キーを押しながら［Enter］をクリックします。 OS X システムの場合は、Command キーを押しながら［Enter］をクリックします。

アサーションの実行

フィルタを実行するには、[アサーションの実行]をクリックします。

JSON スキーマの確認

JSON スキーマの確認アサーションでは、JSON スキーマが有効で、JSON 応答のペイロードがそのスキーマに有効であることを確認できます。

以下のパラメータを入力します。

名前

アサーションの名前を定義します。

条件:

ドロップダウン リストからアサーションの動作を指定します。

次のステップ

アサーションが起動された場合のリダイレクト先のステップを指定します。

ログ

アサーションが起動された場合に出力するイベント テキストを指定します。

Validate Payload Against Schema

JSON スキーマおよびペイロード（応答）の両方を検証するかどうかを示します。 オフにすると、スキーマのみが検証されます。

デフォルト： オン

[コンテンツの選択]タブ

［コンテンツの選択］タブでは、JSON スキーマおよびペイロードのソースを選択して、そのコンテンツを検査できます。

以下のパラメータを入力します。

スキーマ

このアサーション用の JSON スキーマのソースを示します。 ドロップダウン リストから、以下を選択します。

- From Content： ファイル システム上のフラット ファイルから JSON スキーマをロードします。ファイルを選択するには、［Browse File Selection］をクリックします。

- URL から： URL から JSON スキーマをロードします。 ［URL Path］フィールドにその URL を入力します。 ファイル システムを参照するには、［Browse］  をクリックします。 URL を再ロードするには、［Refresh］  をクリックします。
- プロパティから： プロパティから JSON スキーマをロードします。 プロパティ値をリフレッシュするには、［Refresh］  をクリックします。

ペイロード

ペイロードのソース、すなわち JSON テキストを示します。 ドロップダウン リストから、以下を選択します。

- From Content： ファイル システム上のフラット ファイルから JSON スキーマをロードします。ファイルを選択するには、［Browse File Selection］をクリックします。
- URL から： URL から JSON スキーマをロードします。 ［URL Path］フィールドにその URL を入力します。 ファイル システムを参照するには、［参照］  をクリックします。 URL を再ロードするには、［Refresh］  をクリックします。
- プロパティから： プロパティから JSON スキーマをロードします。 プロパティ値をリフレッシュするには、［Refresh］  をクリックします。

［設定］タブ

［設定］タブでは、JSON スキーマの確認アサーションの詳細オプションを設定できます。

以下のパラメータを入力します。

Use Subschema

特定のサブスキーマを検索するように、アプリケーションに指示します。 ［Subschema Filter］フィールドにサブスキーマを入力します。

Referencing Mode

使用する逆参照モードを示します。

解決された URI をすべて逆参照するには、［Canonical］を選択します。スキーマ内の URI を逆参照するには、［Inline］を選択します。

デフォルト： Canonical

アサーションの実行

フィルタを実行するには、［アサーションの実行］をクリックします。

仮想サービス環境アサーション

以下のアサーションは、任意のテスト ステップの仮想サービス環境アサーションのリストで使用できます。

実行モードでのアサート (P. 44)

実行モードでのアサート

実行モードでのアサート アサーションは、現在の実行モードとその参照を確認し、一致した場合に起動します。このアサーションは主に仮想サービス モデルのステップ フローを制御するために使用されます。

以下のパラメータを入力します。

名前

アサーションの名前を定義します。

条件:

ドロップダウン リストからアサーションの動作を指定します。

次のステップ

アサーションが起動された場合のリダイレクト先のステップを指定します。

ログ

アサーションが起動された場合に出力するイベント テキストを指定します。

実行モード

ドロップダウンの使用可能なオプションから実行モードを選択します。実行モードの詳細については、「*CA Service Virtualization の使用*」の「選択したモデルの動作を指定する」を参照してください。

アサーションを実行するには、[アサーションの実行]をクリックします。

モバイル アサーション

以下のアサーションは、テスト ステップのモバイル アサーションのリストで使用できます。

Ensure Same Mobile Screen（モバイル画面が同一であることを確認） (P. 45)
Ensure Screen Element Matches Expression (式に一致する画面エレメントを確認) (P. 47)
Ensure Screen Element is Removed （画面エレメントの削除を確認） (P. 48)

Ensure Same Mobile Screen（モバイル画面が同一であることを確認）

Ensure Same Mobile Screen（モバイル画面が同一であることを確認）アサーションは、モバイル デバイス上の画面エレメントがすべて元のレコーディングに一致することを確認します。 デフォルトでは、画面上のエレメントが異なる場合、テスト ステップは失敗します。

以下のフィールドに入力します。

name

アサーションの名前。 デフォルト値は **Assert same screen** です。

条件:

以下の値のいずれかを選択します。

- **True：** モバイル デバイス上の画面エレメントがすべて元のレコーディングに一致する場合、［次のステップ］フィールドで定義したアクションが実行されます。
- **False：** モバイル デバイス上の画面エレメントが元のレコーディングに一致しない場合、［次のステップ］フィールドで定義したアクションが実行されます。 **False** がデフォルト値です。

次のステップ

選択された［条件:］ステートメントの条件が満たされた場合に実行するアクションを選択します。

- 警告またはエラーを生成する。
- テストを終了、失敗、または中止する。
- 実行される次のテスト ステップを選択する。

ログ

アサーションが実行された場合に、ログ イベント テキストとして表示されるテキスト。

アサーションを実行するには、［アサーションの実行］をクリックします。

Difference Threshold（差異しきい値）

［Difference Threshold （差異しきい値）］では、画面を比較する場合に使用する精度のレベルを定義できます。アプリケーションが変更されていない場合でも、アプリケーションと記録されたスクリーンショットの間に微妙な違いが存在することがあります。 たとえば、テストでは、アプリケーションのテーブルを同じテーブルのスクリーンショットと比較できます。スクリーンショットには強調表示された行が含まれていて、アプリケーションでは行が強調表示されていない場合、ピクセルごとの正確な比較によってこれらのテーブルは不一致として表示されます。

デフォルトの［Difference Threshold （差異しきい値）］は **1000** です。ただし、テスト ケースが正しくない一致を返す場合は、この数値を調整できます。 以下の値を参考に設定します。

- **0 :** 画面上のすべてのピクセルがレコーディングのすべてのピクセルに一致する必要がある完全一致を示します。
- **1000 :** 画面とレコーディングが一致と見なされる程度に近いことを示します。
- **2000 以上 :** 画面とレコーディングがほぼ確実に不一致として分類できることを示します。

Start of Step（**ステップの開始時**）

このチェック ボックスをオンにすると、画面比較はステップの最初に行われます。

End of Step（**ステップの終了時**）

このチェック ボックスをオンにすると、画面比較はステップの最後に行われます。

Consider image pixels（**画像ピクセルを考慮**）

このチェック ボックスをオンにすると、ピクセル比較を使用して画面一致が判定されます。

Consider screen structure（**画面構造を考慮**）

このチェック ボックスをオンにすると、ピクセル比較の代わりに画面の構造を使用して画面一致が判定されます。 たとえば、現在の日時を表示する画面があるとします。 2 日前に実行されたレコーディングの日時は、今日実行するテストに一致しません。 このオプションを使用すると、画面上の特定の値が一致しなくても、同じ画面の構造であることを確認できます。

コンポーネント バインドの検討

Ensure Screen Element Matches Expression (式に一致する画面エレメントを確認)

モバイルの［Ensure Screen Element Matches Expression (式に一致する画面エレメントを確認)］アサーションは、画面エレメントに入力されたデータが指定した正規表現に一致することを確認します。 デフォルトでは、画面エレメントに入力された値が指定した表現に一致しない場合、テストステップは失敗します。

［Ensure Screen Element Matches Expression (式に一致する画面エレメントを確認)］アサーションを選択すると、画面エレメントと一致させる表現を入力するように求められます。 入力すると、名前および表現がタブの［アクション］セクションにキー/値ペアとして表示されます。 これらのフィールドをダブルクリックすると変更できます。

たとえば、正規表現「ame」は名前「Cameron」および「Pamela」に一致します。 画面エレメントに入力された値が大文字で始まり、その後に小文字が 3 つ続くようにする場合、以下の正規表現を入力します。

```
'[A-Z][a-z]{3}'
```

正規表現の詳細については、「アサーションの説明 (P. 9)」を参照してください。

以下のフィールドに入力します。

一致させる正規表現 (値)

指定した画面エレメントに一致させる正規表現。

一致しない場合は、ステップを実行します。

一致しない場合に実行するステップをドロップダウン リストから選択してください。

Ensure Screen Element is Removed（画面エレメントの削除を確認）

Ensure Screen Element is Removed （画面エレメントの削除を確認）アサーションは、次のステップに移る前に、モバイル デバイス上の指定された画面エレメントが削除されたことを確認します。 デフォルトでは、指定されたエレメントが存在する場合、テスト ステップは失敗します。

このアサーションを追加すると、ポップアップ ダイアログ ボックスでエレメントが削除されたことをテストする画面を選択できます。

たとえば、3 つのステップが含まれているテスト ケースがあるとします。最後のステップでは、［戻る］ボタンをクリックする必要があります。［戻る］ボタンが最初のステップに含まれていないことを確認するには、最後のステップの［戻る］ボタンについて Ensure Screen Element is Removed アサーションを追加し、ドロップダウン リストで最初のステップを選択します。

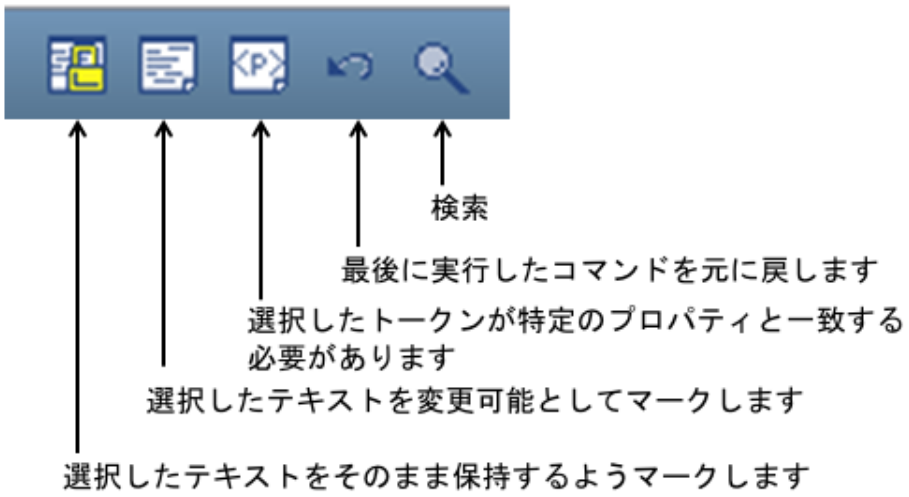
その他のアサーション

以下のアサーションは、任意のテスト ステップのその他のアサーションのリストで使用できます。

テキスト コンテンツの強調表示による比較アサーション

テキスト コンテンツの強調表示による比較アサーションは、HTML ページで動作するよう特別に設計された「画面のペイント」技術を使用します。たとえば、大きな HTML ドキュメントがある場合、ユーザが対象のコンテンツの前後のデータを特定します。 これにより、「対象のコンテンツ」が何と比較されるかが特定されます（通常、データ セットで提供される期待値）。

テキストは、エディタの下部のアイコンを使用してマークされます。



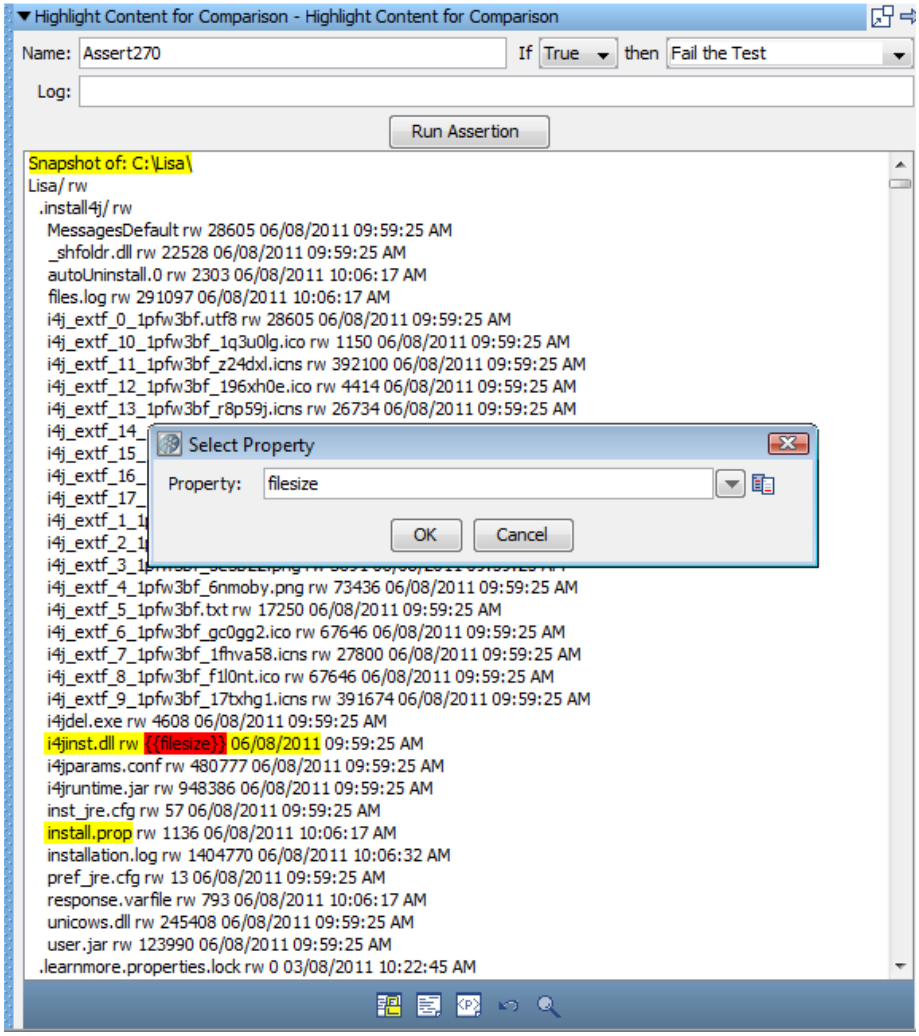
この技術について、例を用いて説明します。

以下の例では、以下の手順を完了します。

- 特定のファイルがバッファに表示されることを確認する。
- いずれかのファイルのサイズをプロパティの値と比較する。

上記の図に示された 3 つのアイコンを使用して、テキストを選択し、適切なアイコンをクリックすることにより、テキストをマークしています。

- 黄色の背景色は、そのまま表示される必要があるテキストを示します。
- 白の背景色は、存在する必要がないか変更可能なテキストを示します。
- 赤の背景色は、ダイアログ ボックスに入力されたプロパティに一致する必要があるテキストを識別します。



ここに示されたトークンのセットは、以下のように解釈できます。

- バッファは、黄色で示された「Snapshot of: C:¥Lisa¥」というテキストで始まる必要があります。
- 次のトークンでバッファ内にあるファイルの数は、変わる可能性があります。 このトークンは「変更可能テキスト」トークンであるため、不一致は重要ではありません。
- 「i4jinst.dll」ファイルおよび「rw」属性が存在する必要があります。
- 赤で示された「filesize」は、プロパティ キー filesize と関連付けられた値が式で変換され、その後に比較が実施されることを意味します。
- 「06/08/2011」というテキストが存在する必要があります。
- 「install.prop」ファイルが存在する必要があります。

- それ以後、バッファは任意の量のコンテンツを持つことができます。

マークアップを完了した後、以下のパラメータを入力します。

名前

アサーションの名前を定義します。

条件:

ドロップダウン リストからアサーションの動作を指定します。

次のステップ

アサーションが起動された場合のリダイレクト先のステップを指定します。

ログ

アサーションが起動された場合に出力するイベント テキストを指定します。

アサーションを実行するには、[アサーションの実行]をクリックします。

注: 「保持テキスト」ブロックは、常に「プロパティ一致」ブロックの両側にある必要があります。

結果が空でないことを確認アサーション

結果が空でないことを確認アサーションは、ステップからの戻り値を確認して、値が返されたことを検証します。応答がない（タイムアウト）か、戻り値の長さがゼロである場合、戻り値は空であると見なされます。［条件］が［True］に設定されている場合、空の応答が受信されない限り、このアサーションはテストに失敗します。 ［条件］が［False］に設定されている場合、空の応答が受信されると、このアサーションはテストに失敗します。

結果が NULL である場合、失敗イベントが発生し、アサーションの evaluate() メソッドが返されます。そのため、アサーションの If/then ロジックには到達しません。

以下のパラメータを入力します。

名前

アサーションの名前を定義します。

条件:

ドロップダウン リストからアサーションの動作を指定します。

次のステップ

アサーションが起動された場合のリダイレクト先のステップを指定します。

ログ

アサーションが起動された場合に出力するイベント テキストを指定します。

アサーションを実行するには、［アサーションの実行］をクリックします。

その他の属性は必要ありません。

注: コンテンツの検証が行われないため、このアサーションは注意して使用してください。

結果が文字列を含むことを確認アサーション

検索対象の値が応答内で見つかった場合、結果が文字列を含むことを確認アサーションは true を返します。 このアサーションは、通常、要求時に提供された一意の ID などの必要な値が応答に含まれることを確認するために使用されます。

詳細については、「結果が文字列を含むことを確認 (P. 26)」を参照してください。

結果が式を含むことを確認アサーション

結果が式を含むことを確認アサーションでは、指定した正規表現がテキストとして結果に存在することを確認できます。

以下のパラメータを入力します。

名前

アサーションの名前を定義します。

条件:

ドロップダウン リストからアサーションの動作を指定します。

次のステップ

アサーションが起動された場合のリダイレクト先のステップを指定します。

ログ

アサーションが起動された場合に出力するイベント テキストを指定します。

正規表現

ステップ結果で検索する正規表現。 たとえば、400 番台の数値が結果のどこかに存在することを確認するには、このパラメータを「4/d/d」に設定します。

アサーションを実行するには、[アサーションの実行] をクリックします。

プロパティと式の一致を確認アサーション

プロパティと式の一致を確認アサーションでは、プロパティの現在値が指定した正規表現に一致することを確認できます。

以下のパラメータを入力します。

名前

アサーションの名前を定義します。

条件:

ドロップダウン リストからアサーションの動作を指定します。

次のステップ

アサーションが起動された場合のリダイレクト先のステップを指定します。

ログ

アサーションが起動された場合に出力するイベント テキストを指定します。

プロパティ キー

確認するプロパティの名前。 プロパティ名を入力するか、プロパティのドロップダウン リストから選択するか、既存の文字列パターンを選択するか、文字列パターンを作成します。

正規表現

プロパティの現在値に一致する必要がある正規表現。

アサーションを実行するには、[アサーションの実行] をクリックします。

ステップ応答時間を確認アサーション

ステップ応答時間を確認アサーションでは、アプリケーションの応答時間の上限と下限のしきい値を定義できます。 パフォーマンスが速すぎるか遅すぎる場合、テスト ケースはこのアサーションの使用により失敗する場合があります。 アプリケーションが応答をすぐに返すときは、トランザクションが正しく処理されていない場合があります。

詳細については、「ステップ応答時間を確認 (P. 27)」を参照してください。

スクリプト アサーション

スクリプト アサーションでは、スクリプトを作成および実行できます。結果はブール値である必要があります。または、false が返されます。

以下のパラメータを入力します。

名前

アサーションの名前を定義します。

条件:

ドロップダウン リストからアサーションの動作を指定します。

次のステップ

アサーションが起動された場合のリダイレクト先のステップを指定します。

ログ

アサーションが起動された場合に出力するイベント テキストを指定します。

アサーションを実行するには、[アサーションの実行] をクリックします。

言語

使用するスクリプト言語を指定します。

値

- Applescript （OS X 用）
- Beanshell
- Freemarker
- Groovy
- JavaScript
- 速度

デフォルト： Beanshell

追加のスクリプト言語を使用するには、「追加のスクリプト言語の有効化」を参照してください。

Copy properties into scope

ステップで使用するためにダウンロードするプロパティを指定できます。

値

- Test state and system properties： テスト ケースおよびシステムのすべてのプロパティ
- Test state properties： テスト ケースに関する情報を提供するプロパティ
- TestExec and logger only： TestExec およびロガーのプロパティのみ

デフォルト： Test state properties

左側のスクリプト エディタへスクリプトを入力します。

スクリプト エディタで利用可能なオブジェクトはすべて右側の［利用可能なオブジェクト］パネルに表示されます。 リストには、データのプリミティブ型（文字列および数値）と、テスト ケースで実行された EJB 応答オブジェクトなどのオブジェクトが含まれます。 ［利用可能なオブジェクト］テーブル内のエントリをダブルクリックすると、その変数名がエディタ領域に貼り付けられます。

スクリプト実行の結果または発生したエラーの説明が表示されるウィンドウを開くには、［テスト］をクリックします。

テスト ケースを保存すると、アサーションに構文エラーがないか確認されます。

考慮事項

- **{{someprop}}** プロパティをスクリプトで使用する場合、スクリプトが実行される前に、実行時のプロパティ値が代入されます。
- 名前に「.」があるプロパティへアクセスする必要がある場合、それらのプロパティは「.」が「_」に置き換えられてスクリプト環境へインポートされます。 したがって、スクリプト内の **{{foo.bar}}** は、**foo_bar** と同じです。
- **testExec** オブジェクトを使用すると、スクリプト ステップまたはアサーションの内部で DevTest ログ イベントを生成できます。 DevTest ログ イベントを生成するには、log4j ロガーの使用に対応するものとして、以下の行をコード化します。 **testExec.log()** メソッドによって、実際の DevTest イベントが発生します。 イベントは ITR で参照できます。

```
testExec.log("Got here");
```

プロパティが等しいことを確認アサーション

プロパティが等しいことを確認アサーションでは、2 つのプロパティの値を比較してそれらが同じであることを確認できます。 通常、このアサーションはデータ セットおよび提供される「期待値」と共に使用され、アプリケーションが正しく機能していることを確認します。

以下のパラメータを入力します。

名前

アサーションの名前を定義します。

条件:

ドロップダウン リストからアサーションの動作を指定します。

次のステップ

アサーションが起動された場合のリダイレクト先のステップを指定します。

ログ

アサーションが起動された場合に出力するイベント テキストを指定します。

第 1 プロパティ

比較での第 1 のプロパティ。 プロパティ名を入力するか、プロパティのドロップダウン リストから選択するか、既存の文字列パターンを選択するか、文字列パターンを作成します。

第 2 プロパティ

比較での第 2 のプロパティ。 プロパティ名を入力するか、プロパティのドロップダウン リストから選択するか、既存の文字列パターンを選択するか、文字列パターンを作成します。

アサーションを実行するには、[アサーションの実行] をクリックします。

呼び出し例外でのアサート アサーション

呼び出し例外でのアサート アサーションでは、Java 例外の発生に対して、テスト フローを変更できます。このアサーションは、応答で特定の Java 例外が返される場合に true をアサートします。

以下のパラメータを入力します。

名前

アサーションの名前を定義します。

ログ

アサーションが起動された場合に出力するイベント テキストを指定します。

アサート

オプション ボタンを使用して、アサーションの動作を選択します。

実行

アサーションが起動された場合のリダイレクト先のステップを選択します。

式

呼び出し例外で検索する式。 正規表現も使用できます。 通常、「.*」という式を使用します。

ファイル監視アサーション

ファイル監視アサーションでは、特定のコンテンツのファイルをモニタし、特定の式の存在（または存在しないこと）に対応できます。 テスト ケースの実行中、このアサーションはバックグラウンドで実行されます。

以下のパラメータを入力します。

名前

アサーションの名前を定義します。

ログ

アサーションが起動された場合に出力するイベント テキストを指定します。

ファイル コンテンツ確認前の遅延時間合計（秒）

このアサーションが含まれるステップの先頭でファイルを確認するまでの待機時間（秒）。

ファイル コンテンツ確認間の待機時間合計（秒）

各確認の間の待機秒数。

ファイル監視が式の監視を停止するまでの時間（秒）

このアサーションが式を確認する秒数の合計。

監視するファイルの URL

監視されているファイルの URL またはパス。

ファイル内で監視する式

応答で監視されている正規表現。

注: 時間は秒単位で、整数である必要があります。 時間のデフォルトは 0 です。

コレクション オブジェクトのコンテンツを確認アサーション

コレクション オブジェクトのコンテンツを確認アサーションでは、コレクションのコンテンツに対する簡単なアサーションを作成できます。 このアサーションは、いくつかの簡単な制約を追加するオプションを使用して、特定のトークンがコレクションにあるかどうかを調べるのに有用な方法です。 たとえば、銀行の Web サービスから返されたデータに口座のリストが含まれている場合、このアサーションは口座 ID が期待値と一致するかどうかを確認できます。

以下のパラメータを入力します。

名前

アサーションの名前を定義します。

条件:

ドロップダウン リストからアサーションの動作を指定します。

次のステップ

アサーションが起動された場合のリダイレクト先のステップを指定します。

ログ

アサーションが起動された場合に出力するイベント テキストを指定します。

チェックするプロパティ（応答全体の場合は空白）

アサーションで使用するオブジェクトを保持するプロパティの名前。 最終応答を使用するには、このフィールドを空白のままにします。 オブジェクトのタイプは、Java コレクションまたは配列である必要があります。

チェックするフィールド（"toString" の場合は空白）

フィールドの名前を入力します。DevTest はその get メソッドをコールします。

検索するトークン（値 1, 値 2）

確認する一連のカンマ区切りのトークン文字列。

完全一致のみ

トークン名は正確に一致する必要があります。

順番が一致する場合のみ

検索するトークン文字列内と同じ順番で見つかる必要がある場合は、このチェック ボックスをオンにします。

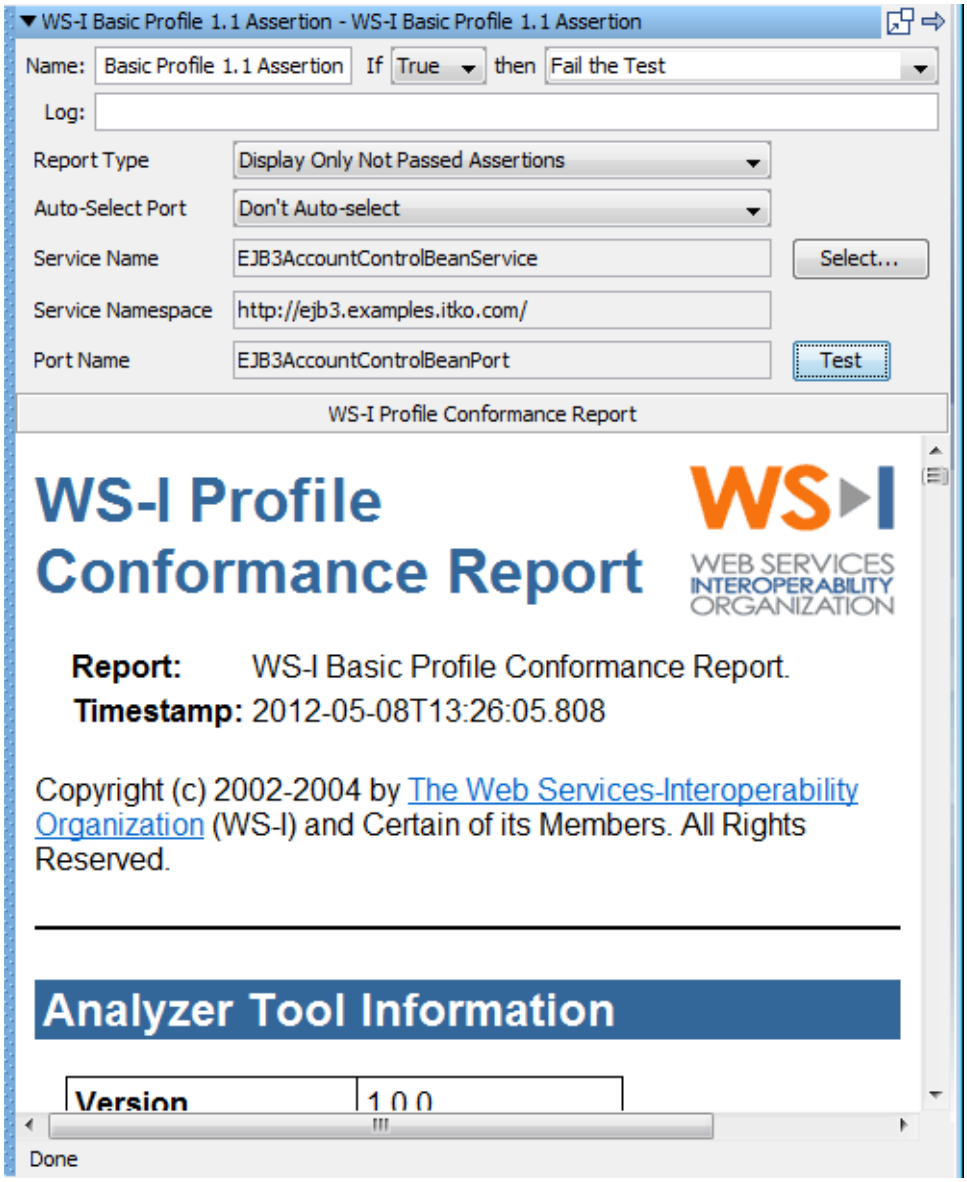
指定したトークンのみを含む（他のオブジェクトは含まない）

見つかるトークンは、検索するトークン文字列内のトークンのみである必要があります。

アサーションを実行するには、[アサーションの実行]をクリックします。

WS-I Basic Profile 1.1 アサーション

WS-I Basic Profile 1.1 アサーションでは、特定の Web サービスに対して WS-I Basic Profile 準拠レポートを取得できます。 このレポートは、WS-I Basic Profile が指定する標準形式で提供されます。



以下のパラメータを入力します。

名前

アサーションの名前を定義します。

ログ

アサーションが起動された場合に出力するイベント テキストを指定します。

レポート タイプ

レポートに含める（WS-I）アサーションのレベルを以下から 1 つ選択します。

- すべてのアサーションを表示
- 情報以外のアサーションを表示
- 失敗したアサーションのみを表示
- 成功しなかったアサーションのみを表示

ポートの自動選択

ポートを選択する方法（特定のポートまたは「自動選択しない」）を決定します。

サービス名

リストからサービス名を選択します。この名前はステップから自動入力できます。

サービス ネームスペース

サービス名に基づいて自動的に入力されます。

ポート名

サービス名に基づいて自動的に入力されます。

アサーションを実行するには、［テスト］をクリックします。

メッセージング VSE ワークフロー アサーション

メッセージング VSE ワークフロー アサーションは、VSE レコーダから自動的に追加されます。このアサーションは、VSE レコーディングを正しく動作させるという特定の目的のために使用されます 使用するときには注意してください。このアサーションが VSE モデルのステップに追加されている場合は、それを削除または編集しないでください。

アサーションを実行するには、[アサーションの実行]をクリックします。

SWIFT メッセージの検証アサーション

SWIFT メッセージの検証アサーションでは、SWIFT メッセージの構文およびセマンティックを検証できます。

このアサーションには、以下のフィールドが含まれます。

名前

アサーションの名前を定義します。

条件:

ドロップダウン リストからアサーションの動作を指定します。

次のステップ

アサーションが起動された場合のリダイレクト先のステップを指定します。

ログ

アサーションが起動された場合に出力するイベント テキストを指定します。

このアサーションには、以下の SWIFT 固有のフィールドも含まれます。

SWIFT メッセージ タイプ

メッセージを検証するメッセージ タイプを指定します。 使用可能なメッセージ タイプは MT、MX、および SEPA です。 デフォルトのメッセージ タイプは MT です。

メッセージ タイプ MT については、DevTest は以下のカテゴリをサポートしています。

MT1nn

顧客支払

MT2nn

銀行間送金

MT3nn

外国為替、金融取引、デリバティブ

MT4nn

督促状および預金票

MT5nn

証券取引

MT7nn

荷為替信用状および保証状

MT9nn

資金管理および顧客状況

メッセージ タイプ SEPA については、DevTest は以下のカテゴリをサポートしています。

- 照会処理（camt.029.001.03）
- 支払取消要求（camt.056.001.01）
- 顧客送金開始（pain.001.001.03）
- 顧客支払状況レポート（pain.002.001.03）
- 金融機関支払状況レポート（pacs.002.001.03S2）
- 払戻（pacs.004.001.02）
- 顧客送金（pacs.008.001.02）

メッセージ タイプ MX については、DevTest は、ISO20022 メッセージの完全なカタログの最新のバージョン（2014 年 2 月時点）をサポートしています。メッセージの一覧については、http://www.iso20022.org/full_catalogue.page を参照してください。

構文のみ検証

アサーションが構文およびセマンティックを検証するかどうかを指定します。

値

- **オン：** アサーションは構文のみを検証します。
- **オフ：** アサーションは構文およびセマンティックの両方を検証します。

デフォルト： オフ

［アサーションの実行］をクリックした場合、検証エラーは［システム メッセージ］ウィンドウに表示されます。

注:

- MT メッセージ内の行が必要なキャリッジ リターン/改行文字（ASCII（16 進）の 0D0A、EBCDIC（16 進）の 0D25）で終了することを確認します。 そうでない場合、以下のエラーがレポートされます。

  "The input Swift message cannot be parsed because of invalid syntax. Please check the message structure. Take notice of block separators,carriage-return line-feed characters and the presence of mandatory blocks."

- MT メッセージの行の末尾に無効なスペースがないことを確認します。 レポートされるエラーは不明瞭な場合があります。

  たとえば、

  :32A:071119EUR50000,

  という行の末尾に無効なスペースがある場合、DevTest は以下のエラーメッセージをレポートします。

  「T43 - The integer part of Rate must contain at least one digit. A decimal comma is mandatory and is included in the maximum length tag:32A.」

- MX メッセージがサポートされているバージョンに準拠していることを確認します。

  たとえば、**camt.052.001.04** はサポートされていますが、古い **camt.052.001.01** はサポートされていません。

アセットの説明

このセクションには、以下のアセットの説明が含まれています。

JDBC 接続アセット

JDBC システムへの接続情報を定義するには、送信先アセットを使用します。 JDBC 接続アセットのアセット クラスは、JDBC 接続アセットと命名されます。

前提条件： データベースの適切な JDBC ドライバが DevTest クラスパスに存在する必要があります。ドライバ JAR ファイルは、ホット デプロイ ディレクトリに配置できます。 Derby クライアント ドライバは DevTest クラスパスに含まれています。したがって、再度それを追加する必要はありません。

パラメータ要件： JDBC ドライバ クラスの名前、データベースの JDBC URL、およびデータベースのユーザ ID とパスワード。 また、SQL クエリを作成するために、データベースのテーブルのスキーマを知っておく必要があります。

事前定義済みの送信先アセットがある場合、ステップ エディタの［送信先］ドロップダウン フィールドからアセットを選択できます。ステップ エディタからアセットを作成するには、アセットの追加  アイコンを選択します。 ステップ エディタからアセットを編集するには、アセットの編集 アイコンを選択します。

アセットを作成する方法

1. このアセットの以下のフィールドを定義します。 接続パラメータにプロパティを使用できます。

   name

   アセットの名前。この名前はステップ エディタの［送信先］フィールドに表示されます。 JDBC システムに対して意味のある名前を使用します。

   説明

   アセットがターゲットにするシステムの詳細について説明する情報。

   JDBC ドライバ

   適切なドライバ クラスの完全なパッケージ名を入力または選択します。 標準のドライバ クラスは、ドロップダウン リストで使用可能です。 また、［参照］ボタンを使用して、ドライバ クラスの DevTest クラス パスを参照できます。

接続文字列

接続文字列はデータベースの標準の JDBC URL です。 URL を入力または選択します。 ドロップダウン リストには、共通データベース マネージャの JDBC URL テンプレートが含まれています。

ユーザ ID

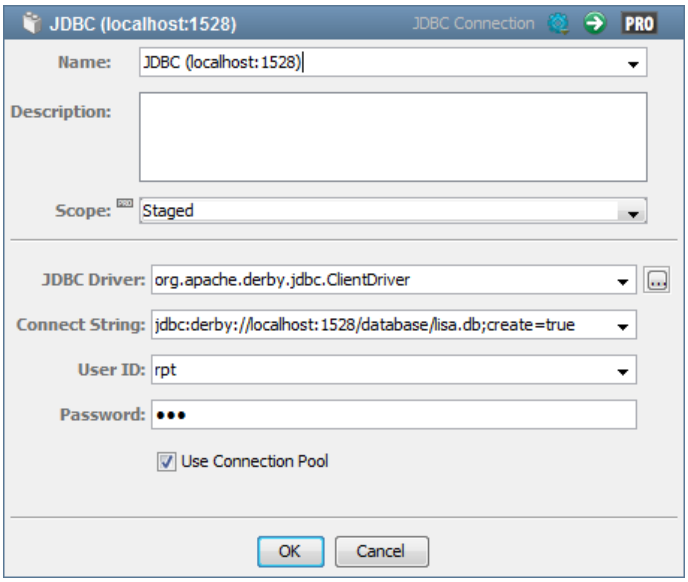
ユーザ ID を入力します（データベースで必要な場合）。

パスワード

パスワードを入力します（データベースで必要な場合）。

接続プールを使用

［接続プールを使用］を選択すると、**lisa.properties** ファイル内の **lisa.jdbc.asset.pool.size** プロパティを使用して接続プールのサイズを設定できます。



2. 詳細モードを表示するには、アセット エディタの右上隅の［PRO］アイコンを選択します。詳細モードでは、アセットのランタイム スコープを指定できます。

JMS クライアント アセット

Java Message Service （JMS）は、Java プログラムがエンタープライズ メッセージング システムと通信することを可能にする仕様です。 元のバージョンは 1.0 です。 最新のバージョンは 1.1 です。

JMS では以下の用語を使用します。

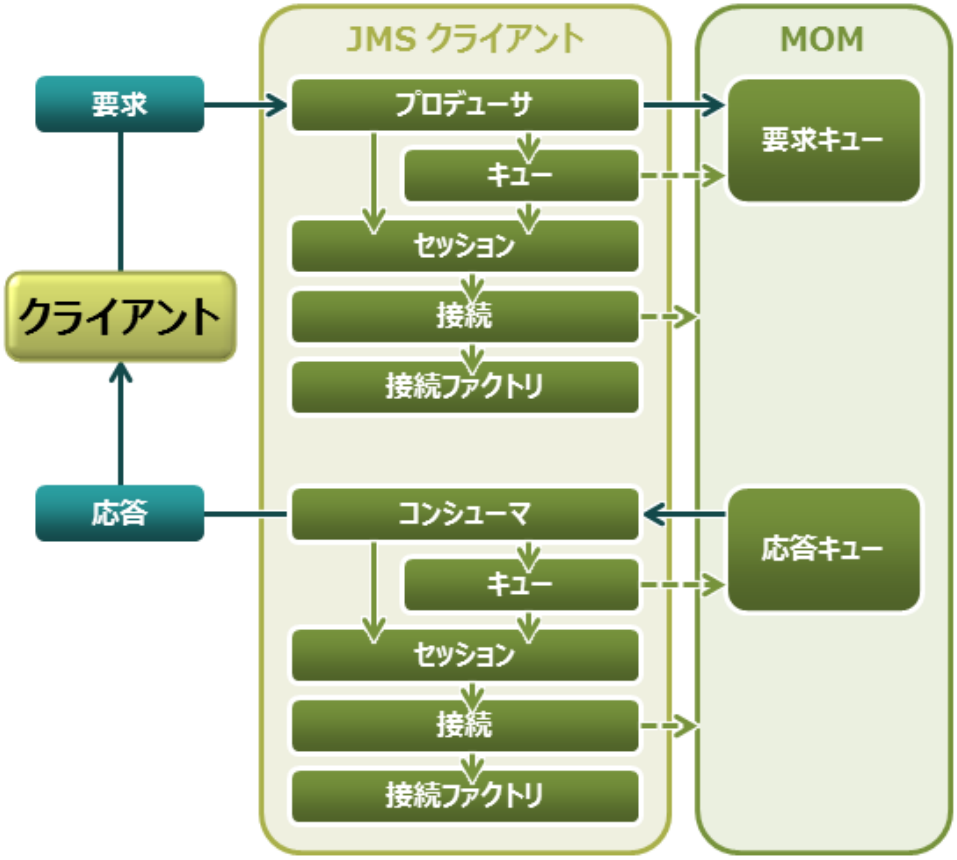
- **JMS クライアント**： JMS を使用してメッセージング システムと通信する Java プログラム。
- **JMS プロバイダ**： JMS を実装するメッセージング システム。

JMS クライアントが使用する以下のタイプのオブジェクトのアセットを作成できます。

- 接続ファクトリ
- 接続
- セッション
- プロデューサ
- コンシューマ
- 送信先

各 JMS クライアント アセットのエディタでは、各パラメータにパラメータの目的を説明するツールヒントがあります。

以下の図は、JMS クライアントがどのように要求メッセージおよび応答メッセージを処理するかを示しています。



接続ファクトリ

接続ファクトリは接続を作成するために使用されます。

接続ファクトリは、どのように初期化されるかという点から特徴付けることができます。

- **汎用接続ファクトリ**： Java Naming and Directory Interface （JNDI）を使用して初期化されます。
- **直接接続ファクトリ**： JMS プロバイダに固有の方法で初期化されます。このタイプの接続ファクトリには、多くの場合大きな、固有のパラメータのセットがあります。

接続ファクトリは、サポートする送信先という点からも特徴付けることができます。

- **キュー接続ファクトリ**：キューのみをサポートします。このタイプは JMS バージョン 1.0 からです。
- **トピック接続ファクトリ**：トピックのみをサポートします。このタイプは JMS バージョン 1.0 からです。
- **接続ファクトリ**：接続ファクトリが上記のタイプのどちらかとして指定されない場合、キューおよびトピックの両方をサポートします。

接続ファクトリ アセットの多くは、これらの 2 つのカテゴリの組み合わせです。 たとえば、TIBCO EMS 用の直接 JMS 1.0 トピック接続ファクトリ アセットは、トピックのみをサポートする直接接続ファクトリです。

接続

接続は、JMS プロバイダとのアクティブな接続を表します。 接続はセッションを作成するために使用されます。

接続は、サポートする送信先という点から特徴付けることができます。

- **キュー接続**： キューのみをサポートします。 このタイプは JMS バージョン 1.0 からです。
- **トピック接続**： トピックのみをサポートします。 このタイプは JMS バージョン 1.0 からです。
- **接続**： 接続が上記のタイプのどちらかとして指定されない場合、キューおよびトピックの両方をサポートします。

接続は、そのセッションが開いている間は開いたままである必要があります。

セッション

セッションはプロデューサおよびコンシューマを作成するために使用されます。

セッションは、サポートする送信先という点から特徴付けることができます。

- **キュー セッション**： キューのみをサポートします。このタイプは JMS バージョン 1.0 からです。

- **トピック セッション**： トピックのみをサポートします。 このタイプは JMS バージョン 1.0 からです。
- **セッション**： セッションが上記のタイプのどちらかとして指定されない場合、キューおよびトピックの両方をサポートします。

セッションは、そのプロデューサおよびコンシューマがアクティブな間は開いたままである必要があります。

プロデューサ

プロデューサは送信先にメッセージを送信するために使用されます。

DevTest のプロデューサのタイプは 1 つだけです。

コンシューマ

コンシューマは送信先からメッセージを受信するために使用されます。

DevTest のコンシューマのタイプは 1 つだけです。 ただし、以下の 2 つの方法のいずれかでコンシューマを使用することができます。

- **同期**： クライアントがメッセージの受信を待機している間、クライアントはほかに何もすることができません。
- **非同期**： クライアントがメッセージの受信を待機している間、クライアントはその他のタスクを実行できます。

送信先

送信先は、メッセージが配置される JMS プラットフォーム上の場所を表します。

送信先はキューおよびトピックに分けられます。

- **キュー**： ポイント ツー ポイント メッセージング モデルをサポートする送信先。 メッセージがキューに送信される場合、1 人の受信者のみがメッセージを受信できます。

- **トピック**： パブリッシュ/サブスクライブ メッセージング モデルをサポートする送信先。 メッセージがトピックに送信される場合、複数の受信者がメッセージを受信できます。

送信先は、その作成方法という点からも特徴付けることができます。

- **JNDI 送信先**： Java Naming and Directory Interface （JNDI）を使用して初期化されます。
- **一時 JNDI 送信先**: JMS セッションを使用して一時的に送信先を作成できます。この送信先は、セッションが開いている限り存在し、セッションが閉じられると削除されます。
- **送信先**： 上記のタイプのどちらかとして指定されない、JMS セッションを通じて取得される静的な送信先。

送信先アセットは、これらの 2 つのカテゴリの組み合わせです。たとえば、JMS JNDI キュー アセットは、JNDI によって初期化されるポイント ツー ポイントの送信先です。

JNDI 初期コンテキスト アセット

Java Naming and Directory Interface （JNDI）は、Java プログラムがネーミングおよびディレクトリ サービスと通信することを可能にする仕様です。

DevTest には、JNDI 初期コンテキスト アセットが含まれます。JMS クライアント アセット (P. 69)は、JNDI 初期コンテキスト アセットを使用して JMS 接続ファクトリおよび送信先を見つけます。

JNDI 初期コンテキスト アセットのエディタでは、各パラメータにパラメータの目的を説明するツールヒントがあります。

SAP JCo 送信先アセット

SAP システムへの接続情報を定義するには、送信先アセットを使用します。1 つのアセット クラスが 3 つの SAP ステップすべてに使用されます。このクラスは、SAP JCo 送信先アセットと命名されます。

事前定義済みの送信先アセットがある場合、ステップ エディタの［送信先］ドロップダウン フィールドからアセットを選択できます。ステップ エディタからアセットを作成するには、アセットの追加  をクリックします。 ステップ エディタからアセットを編集するには、アセットの編集  をクリックします。

アセットを作成する方法

1. このアセットの以下のフィールドを定義します。 SAP 接続パラメータにプロパティを使用できます。

   name

   アセットの名前。この名前はステップ エディタの［送信先］フィールドに表示されます。 SAP システムに対して意味のある名前を使用します。

   説明

   アセットがターゲットにするシステムの詳細について説明する情報。

   サーバ タイプ

   ［アプリケーション サーバ］または［メッセージ サーバ］を選択します。

   ホスト

   SAP システムのホスト名または IP アドレス

   システム番号

   SAP システム番号（アプリケーション サーバ用）

   R/3 名

   R/3 の名前（メッセージ サーバ用）

   グループ

   SAP アプリケーション サーバのグループ（メッセージ サーバ用）

   ユーザ

SAP ユーザ名

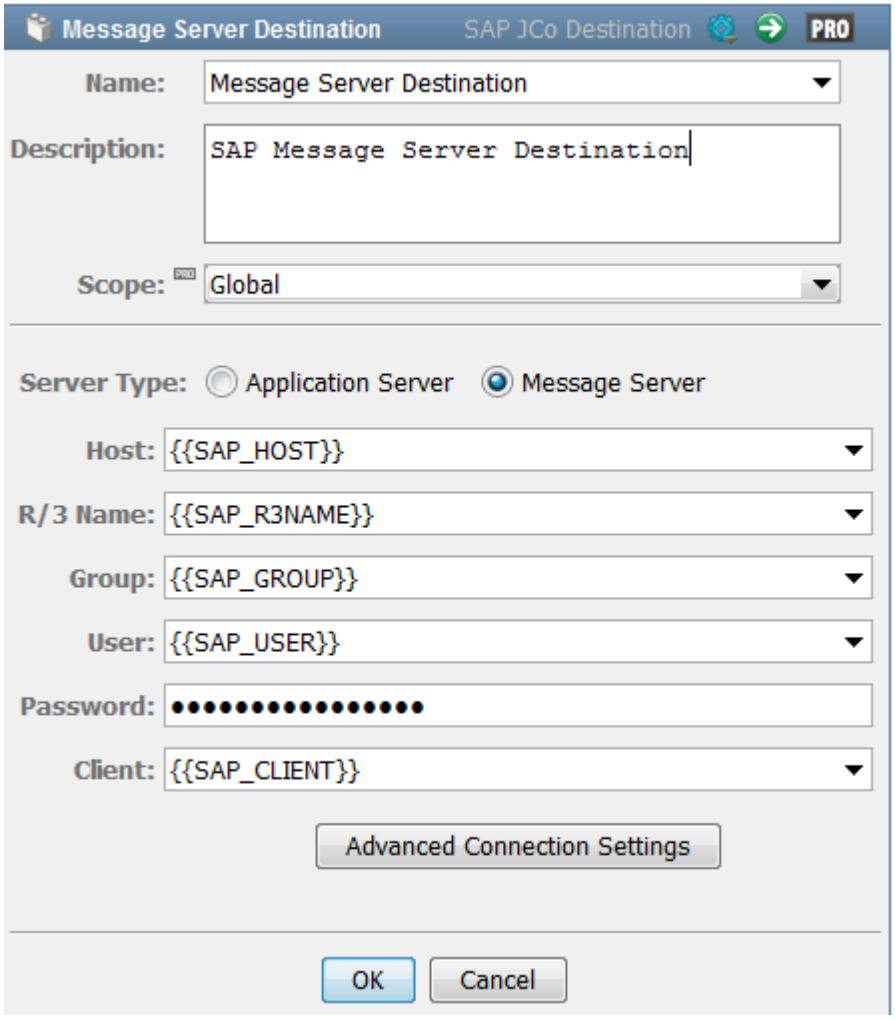
パスワード

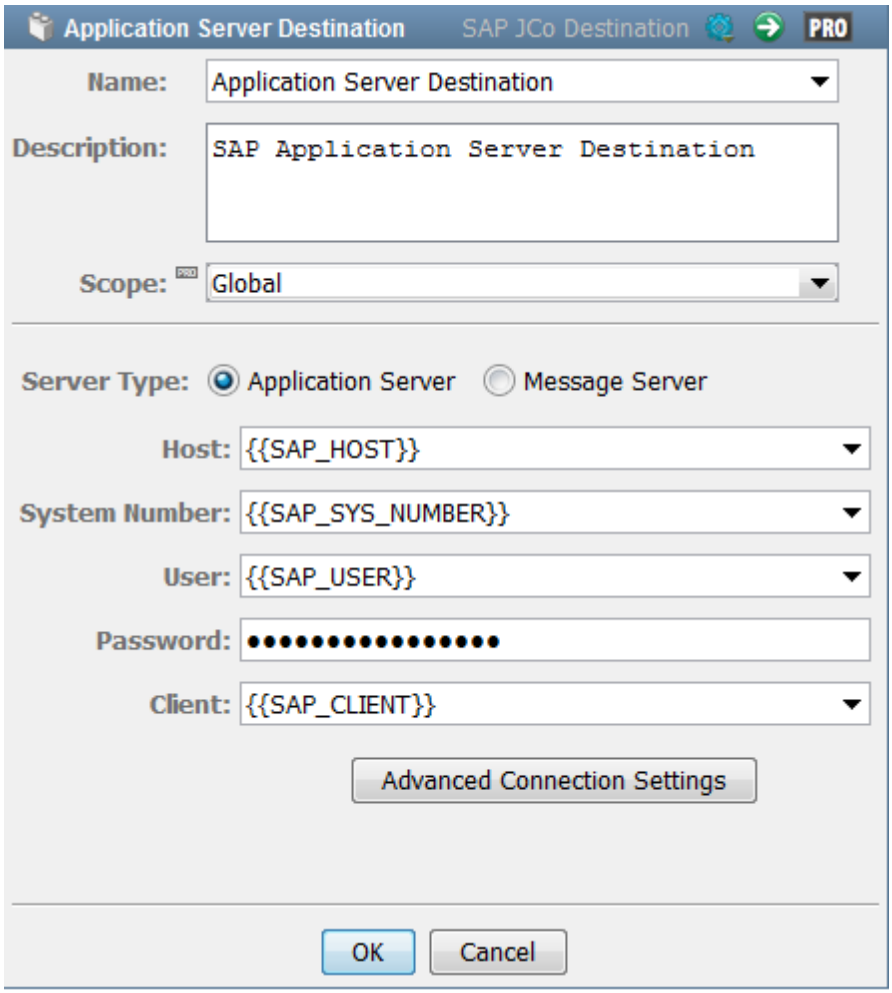
SAP ユーザのパスワード

クライアント

SAP クライアント

以下の図は、アプリケーション サーバおよびメッセージ サーバ両方の送信先アセット定義を示しています。





2. 詳細モードを表示するには、アセット エディタの右上隅の［PRO］を選択します。詳細モードでは、アセットのランタイム スコープを指定できます。
3. 高度な SAP 接続プロパティを選択するには、［高度な接続設定］をクリックします。 設定値を入力すると、設定のデフォルト値を上書きできます。 目的の設定を選択し、［OK］をクリックします。

電子メール接続アセット

SMTP メール サーバへの接続情報を定義するには、接続アセットを使用します。

事前定義済みの接続アセットがある場合、ステップ エディタの［接続］ドロップダウン リストからアセットを選択できます。 ステップ エディタからアセットを作成するには、アセットの追加  をクリックします。

ステップ エディタからアセットを編集するには、［アセットの編集］  をクリックします。

アセットを作成する方法

1. このアセットの以下のフィールドを定義します。 SMTP 接続パラメータにプロパティを使用できます。

   name

   アセットの名前を定義します。 この名前は、ステップ エディタの［接続］フィールドに表示されます。 SMTP メール システムに対して意味のある名前を使用します。

   説明

   アセットがターゲットにするシステムの詳細について説明する情報を指定します。

   サーバ

   SMTP メール サーバの名前を指定します。

   セキュリティ

   通信チャネルを保護するために使用する暗号化のタイプを指定します。

   値

   - **なし：** 通信で暗号化を使用しません。
   - **SSL/TLS：** 通信で SSL/TLS 暗号化を使用します。

   ポート

   （オプション） SMTP メール サーバが接続するポートを指定します。

   デフォルト設定は以下のとおりです。

- **25：**［セキュリティ］に［なし］を指定する場合のデフォルトです。
- **465：**［セキュリティ］に［**SSL/TLS**］を指定する場合のデフォルトです。

認証

電子メール サーバへの接続に認証を使用するかどうかを指定します。

値

- **オフ：** 電子メール サーバでは、認証が不要です。
- **Password Authentication：** 電子メール サーバでは、ユーザ認証およびパスワード認証が必要です。

ユーザ

（オプション）接続を認証するために SMTP メール サーバが使用するユーザ ID を指定します。［認証］が［**オフ**］の場合、このフィールドは無効です。

パスワード

（オプション） SMTP メール サーバが検証するパスワードを指定します。［認証］が［**オフ**］の場合、このフィールドは無効です。

2. 詳細モードを表示するには、アセット エディタの右上隅の［PRO］をクリックします。詳細モードでは、アセットのランタイム スコープを指定できます。

モバイル アセット

［Mobile Session］ダイアログ ボックスを使用して、さまざまなデバイスでモバイル アセットを作成できます。

次の手順に従ってください:

1. アセットを作成する設定ファイルを開きます。
2. アセット ブラウザで、ペインの下部にある［追加］  をクリックします。
3. ［Mobile Session］をクリックします。

   ［Mobile Session］ダイアログ ボックスが表示されます。
4. アセットの名前を入力します。この名前はステップ エディタの［送信先］フィールドに表示されます。 意味のある名前を使用します。
5. アセットを識別するのに役立つ説明を入力します。
6. プラットフォーム（iOS または Android）を選択します。
7. ［アプリケーション］フィールドに、アプリケーション パッケージ ファイル（.apk または .app）のフル パスを入力します。 または、フォルダ アイコンをクリックして、コンピュータ上のファイルを見つけ、選択します。
8. （iOS のみ）［ファミリー］フィールドで、iOS デバイスのタイプ（［iPhone または iPad］、［iPhone のみ］、または［iPad のみ］）を選択します。
9. ターゲットを選択します。
   - エミュレータ（Android）
   - シミュレータ（iOS）
   - 接続デバイス
   - SauceLabs

   この後のフィールドは、選択するターゲットによって異なります。
10. 以下のフィールドを定義します。

    エミュレータの場合（Android のみ）

    エミュレータ アセットは、Android デバイスでのテストに使用されるモバイル エミュレータを指定します。

    AVD

フォルダ アイコンをクリックして、モバイル テストを設定するときに定義した Android AVD を見つけます。

SDK バージョン

テスト ケースで使用する SDK のバージョンを選択します。

シミュレータの場合（iOS のみ）

シミュレータ アセットは、iOS デバイスでのテストに使用されるモバイル シミュレータを指定します。

シミュレータ

フォルダ アイコンをクリックして、コンピュータ上のシミュレータを見つけます。

iOS のバージョン

テスト ケースで使用する iOS のバージョンを選択します。

接続デバイスの場合

接続デバイス アセットは、テスト ケースに使用されるモバイル デバイスを指定します。 このアセットは、ユーザのコンピュータに接続される物理 iOS および Android モバイル デバイスに使用されます。

接続

フォルダ アイコンをクリックして、コンピュータ上の目的のシミュレータを見つけます。

注: デバイスを接続または切断する場合は、［接続されたデバイス］ダイアログ ボックスの［リフレッシュ］をクリックして最新のデバイスを表示してください。

SDK バージョン（Android のみ）

テスト ケースで使用する SDK のバージョンを選択します。

iOS バージョン（iOS のみ）

テスト ケースで使用する iOS のバージョンを選択します。

SauceLabs の場合

SauceLabs は、拡張性の高い iOS および Android シミュレータのクラウド プロバイダです。SauceLabs アセットは、モバイル クラウド テスト用の SauceLabs アカウント情報を指定します。

注: SauceLabs アセットを作成するには、一意のアクセス キーを持つ SauceLabs アカウントが必要です。

ユーザ名

SauceLabs にアクセスするためのユーザ名を定義します。

アクセス キー

SauceLabs にアクセスするためのキーを定義します。

SDK **バージョン**（Android **のみ**）

テスト ケースで使用する SDK のバージョンを選択します。

iOS **バージョン**（iOS **のみ**）

テスト ケースで使用する iOS のバージョンを選択します。

11. 詳細オプションを定義するには、PRO をクリックします。

    これで［スコープ］フィールドを使用できます。詳細については、「ランタイム スコープ」を参照してください。

12. ［アプリケーション］フィールドにアプリケーションを入力すると、［詳細］ウィンドウにアプリケーションの固有情報が表示されます。
13. アプリケーションがネイティブ エレメントとブラウザ エレメントの両方を使用して作成されている場合は、［Mixed Mode］チェック ボックスをオンにします。
14. 保存する前にアセットを検証するには、  をクリックします。
15. ［OK］をクリックします。

    新しいアセットがアセット ブラウザに表示されます。

    注: 複数のモバイル アセットを定義した場合、テスト ケースを記録する前にアセットを 1 つ選択する必要があります。

重要: Android テストには、適切なバージョンの Android SDK ビルド ツールが必要です。zipalign または aapt に関するエラー メッセージが表示される場合は、作業を続行する前に、「Android SDK ビルド ツール」を参照してください。

コンパニオンの説明

このセクションには、以下のコンパニオンの説明が含まれています。

Web ブラウザ シミュレーション コンパニオン

Web ブラウザ シミュレーション コンパニオンでは、さまざまな Web ブラウザをシミュレートできます。 Web ブラウザは User-Agent HTTP ヘッダを使用して、Web サーバに識別情報を提供します。 ステージングされたテストで複数の仮想ユーザを実行する場合、複数のユーザ エージェントをシミュレートするために DevTest を設定します。各ユーザ エージェント文字列に相対的なウェイトを割り当てることで、特定のブラウザがほかのものよりも頻繁に使用されるようにします。

ウェイトを指定するには、デフォルトのブラウザ選択コンパニオン エディタを使用します。

Web ブラウザ選択コンパニオンを設定するには、ブラウザ エージェントおよびウェイトのリストを入力または編集します。

以下のパラメータを入力します。

ユーザ - エージェント

シミュレートするブラウザ。

ウェイト

このブラウザのウェイト。たとえば、3 つのブラウザに対して 25 パーセント、25 パーセント、50 パーセントのウェイトを割り当てると仮定します。 3 つの行に対して 1、1、2 のウェイトを入力し、その他の行に対しては 0 のウェイトを入力します (または余分な行を削除します)。

テスト ケースが実行されると、DevTest はエミュレートするブラウザを 1 つ選択します。DevTest から送信されたすべての HTTP トランザクションには、そのブラウザ用の User-Agent 文字列が含まれます。 選択条件はランダムで、各ブラウザのウェイトは、そのブラウザが選択される「可能性」を表します、

DevTest は、テスト ケースの初期化時にエミュレートするブラウザを選択するだけです。 DevTest が選択するブラウザ エージェント文字列は、テスト ケースの実行中も有効なままです。 ブラウザのディストリビューションをシミュレートするには、スイート内でテスト ケースを複数回実行します。

別のユーザ エージェントを追加するには、［追加］ をクリックします。

行を削除するには、［削除］ をクリックします。

ブラウザ帯域幅シミュレーション コンパニオン

ブラウザ帯域幅シミュレーション コンパニオンでは、仮想ユーザのさまざまな帯域幅をシミュレートできます。 一部のテスト シナリオでは、さまざまなタイプのインターネット接続のシミュレーションが要求されます。

ブラウザ帯域幅シミュレーション コンパニオンを設定する方法

以下のパラメータを入力します。

バイト/秒

接続速度。 たとえば、56K の接続速度をシミュレートするには、［バイト/秒］に「7000」（56,000 ビット / 8 ビット/バイト = 7,000 バイト/秒）と入力します。

ウェイト

この行に与えられるウェイト。 たとえば、3 つの行に 25%、25%、50% のウェイトを割り当てる場合には、3 つの行の［ウェイト］列に、「1」、「1」、「2」と入力します。 示した例では、仮想ユーザの半分は 6,000 バイト/秒で接続し、半分は 100,000 バイト/秒で接続します。

行を追加するには、［追加］ボタンを使用します。

行を削除するには、［削除］ボタンを使用します。

HTTP 接続プール コンパニオン

HTTP 接続プール コンパニオンによって、ターゲット サーバごとの HTTP 接続数を制限できます。 このコンパニオンは、HTTP/HTML 要求、REST、および RAW SOAP 要求テスト ステップにのみ適用されます。

DevTest は、通常、各仮想ユーザあたり 1 つの HTTP 接続を使用します。 たとえば、100 人の仮想ユーザを使用してテストを実行する場合、クライアントとサーバにはそれぞれ開いているソケットが 100 個あります。

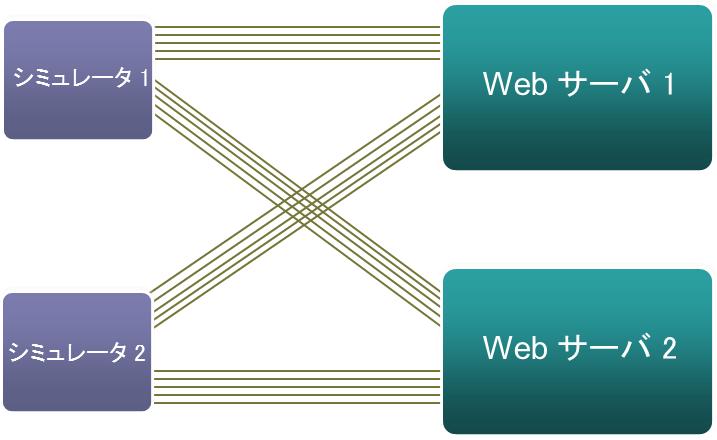
各シミュレータに数千の仮想ユーザで負荷テストを実行する場合、基盤となるオペレーティング システムが使用可能なソケットを使い果たす可能性があります。 このシナリオでは、HTTP 接続プール コンパニオンを使用します。

ConnectionsPerTargetHost パラメータは、一意のエンドポイントそれぞれに割り当てる接続数を指定します。

テスト ケースに、Web サーバ 1 に接続する HTTP ステップと Web サーバ 2 に接続する 2 番目の HTTP ステップがあると仮定します。このテスト ケースには、HTTP 接続プール コンパニオンがあり、ターゲット ホストごとに 5 個の接続が設定されています。 ステージング ドキュメントは、100 人の仮想ユーザを実行するように設定されています。2 台のシミュレータ サーバがあるため、デフォルトでは、それぞれ 50 人の仮想ユーザを取得します。

シミュレータ 1 は、Web サーバ 1 への 5 つの接続と Web サーバ 2 への 5 つの接続を作成します。 シミュレータ 2 も同様に実行します。 各 Web サーバには、現在 10 個のクライアント接続があります。 最初の HTTP ステップでは、仮想ユーザは、Web サーバ 1 への 5 個の接続の 1 個が使用可能になるのを待つ必要があります。 仮想ユーザは、その接続を使用して HTTP コールを作成し、接続はプールへ戻ります。

以下の図に、このシナリオを示します。 シミュレータ 1 は、Web サーバ 1 への 5 個の接続および Web サーバ 2 への 5 個の接続を保持しています。 シミュレータ 2 は、Web サーバ 1 への 5 個の接続および Web サーバ 2 への 5 個の接続を保持しています。


シミュレータ 1
シミュレータ 2
Web サーバ 1
Web サーバ 2

Web プロキシ コンパニオンを使用するための DevTest の設定

Web プロキシ コンパニオンを使用するための DevTest の設定によって、すべての Web テスト ステップにプロキシを設定できます。 環境でプロキシの使用を指定する場合は、このコンパニオンを使用します。 プロキシ情報は組織に固有です。 会社のプロキシ設定に関しては運用チームに問い合わせてください。

Web プロキシ セットアップ コンパニオンを設定するには、Web プロキシ コンパニオン エディタで以下のパラメータを入力します。

Web プロキシ サーバ（ホストおよびポート）

1 番目のフィールドにプロキシ サーバの名前または IP アドレス、2 番目のフィールドにポート番号。

プロキシ サーバを使用しないホストおよびドメイン

プロキシ サーバを使用しないドメインおよびホストの名前。

セキュア Web プロキシ サーバ（SSL プロキシ ホストおよびポート）

1 番目のフィールドに SSL プロキシ サーバの名前または IP アドレス、2 番目のフィールドにポート番号。

セキュアな Web プロキシ サーバを使用しないホストおよびドメイン

セキュアなプロキシ サーバを使用しないドメインおよびホストの名前。

単純なホスト名を除外

localhost や servername.company.com のようなホスト名を除外するために選択します。

プロキシ サーバ認証

プロキシ サーバに対して認証が必要な場合のドメイン名、ユーザ名、およびパスワード。

事前送信

［チャレンジを待機］、［ベーシックを送信］、または［NTLM を送信］を選択します。

DevTest は、local.properties ファイルを使用して、すべてのテスト ケースに Web プロキシを割り当てることもできます。 このファイルは、DevTest ホーム ディレクトリにあります。 **lisa.http.webProxy.host** および **lisa.http.webProxy.port** プロパティを適切に更新し、DevTest を再起動します。 DevTest ホーム ディレクトリに local.properties ファイルがない場合は、既存の _local.properties の名前を local.properties へ変更し、それを使用します。

同期ポイントのセットアップ コンパニオン

同期ポイントの作成コンパニオンでは、テスト ケースまたはテスト スイートで同期ポイントとして使用するテスト ステップを選択できます。同期ポイントでは、全員がこのステップに到達するまで、仮想ユーザが一時停止および待機します。 その後、すべての仮想ユーザはステップを同時に実行するために解放されます。 この機能は、負荷テストで同時テストまたはリソースのピーク使用率を設定する場合に役立ちます。

たとえば、100 人のユーザのテストを設定し、ユーザをすべてアプリケーションにログインさせ、テストを同時に実行させることができます。

同期ポイントは単一のテストに適用されます。または、テスト スイート内のすべてのテストに適用できます。 テスト スイートでは、同期ポイント名はスイート全体で同じである必要があります。ただし、ステップは異なることができます。 シリアル テストは定義によって同期ポイントに同時に到達できないため、複数のテスト シナリオ（およびテスト スイート）がパラレルで実行されるように設定される必要があります。 複数の同期ポイントをテスト ケースまたはテスト スイートで定義できます。

同期ポイントの作成コンパニオンを設定するには、以下のパラメータを入力します。

同期ポイント名

同期ポイントに対して指定する名前。

ステップ

ドロップダウン リストから同期ポイントのステップを選択します。仮想ユーザはこのステップを実行する前に一時停止します。

タイムアウト秒数（なし場合は 0）

同期が発生するための待機秒数。 タイムアウト期間が経過する前に、すべての仮想ユーザがステップに到達する必要があります。

Set Up an Aggregate Step（集約ステップの設定）コンパニオン

Set Up an Aggregate Step （集約ステップの設定）コンパニオンでは、複数の物理テスト ステップをメトリック収集およびレポートのための 1 つの論理ステップとして集約して報告できます。 コンパニオンは、含められたステップを自動的にすべて Quiet に設定します。

集約ステップの設定エディタで、以下のパラメータを入力します。

集約名

集約ステップの名前を定義します。 スペースを使用できます。

開始ステップ

集約の開始（最初の）ステップを指定します。

集約対象

集約に含めるステップ（開始ステップおよび終了ステップを除く）を選択します。

終了ステップ

プルダウン メニューから、集約の終了（最後の）ステップを指定します。

集約ステップは、すべてのレポートに示されます。

1 つのテスト ケースに含めることができる、集約ステップの設定コンパニオンは、1 つのみです。

観察対象システム VSE コンパニオン

LISA 6.0.6 より前では、インバウンド要求の応答時間は、その要求に対して決定された特定の応答の反応時間仕様から求められていました。 主に負荷テストおよびパフォーマンス テストのシナリオでは、ライブ システムの応答時間から応答時間をモデル化する必要がある場合があります。 たとえば、ライブ システムでは、負荷のピーク時にパフォーマンスの低下を示して、名目上の応答時間の 2 倍になる可能性があります。 この場合、VSE にこの応答時間曲線をエミュレートさせることは有用です。 また、たとえば 3 時間のテストを超えて実施される 12 時間の観察対象応答時間メトリックなど、任意の間隔にわたって、この曲線を VSE がカバー（再生）することを可能にすることも有用です。

観察対象システム VSE コンパニオンはこれらの要件をサポートします。仮想サービスでこの動作を提供する場合は、このコンパニオンを追加および設定します。

設定情報

観察対象システム コンパニオンには、以下のパラメータが必要です。

開始日時および終了日時

測定された応答時間を読み取るタイム「ウィンドウ」を定義します。これらのタイムスタンプは包含的です。 このウィンドウの外部にあるデータ セットまたはデータ プロバイダからのタイム スタンプ データは無視されます。

想定実行時間

仮想サービスが応答時間曲線に「適合」する間隔を表す期間を定義します。 前の例では、この値は 3 時間に設定されます。

たとえば、以下の値が使用されます。

- 想定実行時間は 30 分です。
- 開始日時ウィンドウは 30 分です。
- 終了日時ウィンドウは 30 分です。
- バッファ サイズは 10 分です。

最初に 10 分、その後、順に 10 ～ 20 分、20 ～ 30 分、45 ～ 60 分のバッファを取得します。

以下の値を使用します。

- 想定実行時間は 15 分です。

- 開始日時ウィンドウは 30 分です。
- 終了日時ウィンドウは 30 分です。
- バッファ サイズは 10 分です。

最初の 10 分間は最初の 5 分間で再生し、次の 10 分間は次の 5 分間で再生し、その次の 10 分間はその次の 5 分間で再生します。

バッファ サイズ

分単位の期間を定義します。応答時間データが一度に取得される量を制御するために使用できます。 パラメータのデフォルト値は、一度に 1 時間のデータです。

たとえば、想定実行時間が 1 時間、バッファ サイズが 15 分の場合、以下の結果が得られます。

- 最初に 15 分のバッファを取得します。
- 次に、15 ～ 30 分のバッファを取得します。
- 次に、30 ～ 45 分のバッファを取得します。
- 次に、45 ～ 60 分のバッファを取得します。
- 次に、最初の 15 分からデータを再度取得します

観察対象のシステム データ プロバイダ

このパラメータは、データを取得する場所をコンパニオンに指示します。 現在、DevTest データ セット データ プロバイダおよび CA Application Performance Management （Wily）データ プロバイダが提供されています。

どのデータ プロバイダも、以下の 3 つの情報を提供する必要があります。

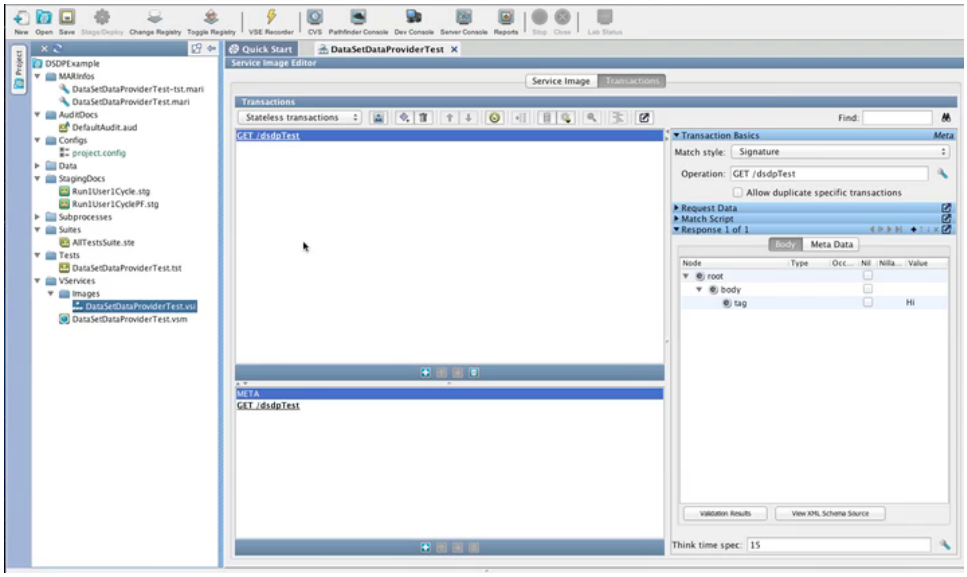
- ID （文字列）
- タイム スタンプ
- タイム スタンプの時点の応答時間

DevTest データ セット データ プロバイダでは、「id」、「timestamp」、「responseTime」というフィールドで、データ セットがこれらの情報を提供する必要があります。 タイムスタンプ値は、実際の Date オブジェクトではない場合、「yyyy-MM-dd HH:mm:ss.SSS」形式の文字列である必要があります。ID を任意の指定したインバウンド要求にマップする方法は、データ プロバイダに固有です。 データ セット プロバイダの場合、それは要求の操作に一致する必要があります。 CA Application Performance Management（Wily）プロバイダは正規表現に基づいた方法を使用します。

データ セット ソースの例

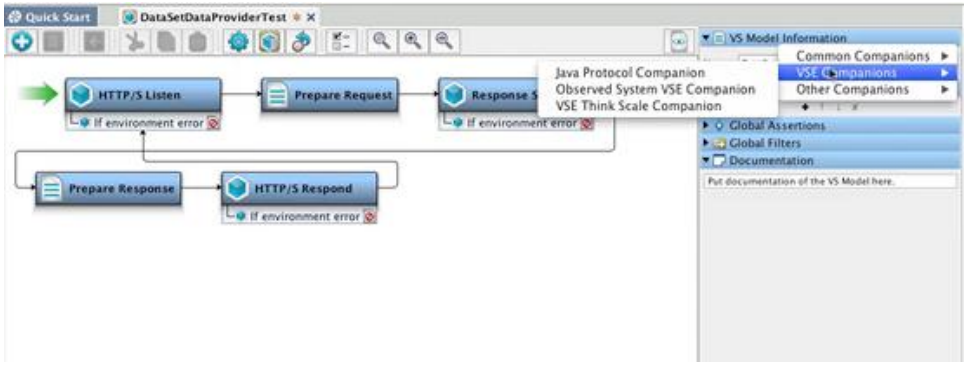
この例では、データ セットが、観察対象システム VSE コンパニオンの入力を提供します。 このコンパニオンが提供するデータの定義により、トランザクション、またはある期間にわたる複数のトランザクションに対して応答時間を変更できます。

以下に示すサービス イメージには、1 つのトランザクションが含まれています。また、［反応時間］は 15 ミリ秒に設定されています。



このサービス イメージと関連する仮想サービス モデルには、少数の単純なステップがあります。

1. コンパニオンを追加するには、コンパニオン パネルの ✚ ［追加］をクリックします。
2. VSE コンパニオン、観察対象システム VSE コンパニオンを選択します。



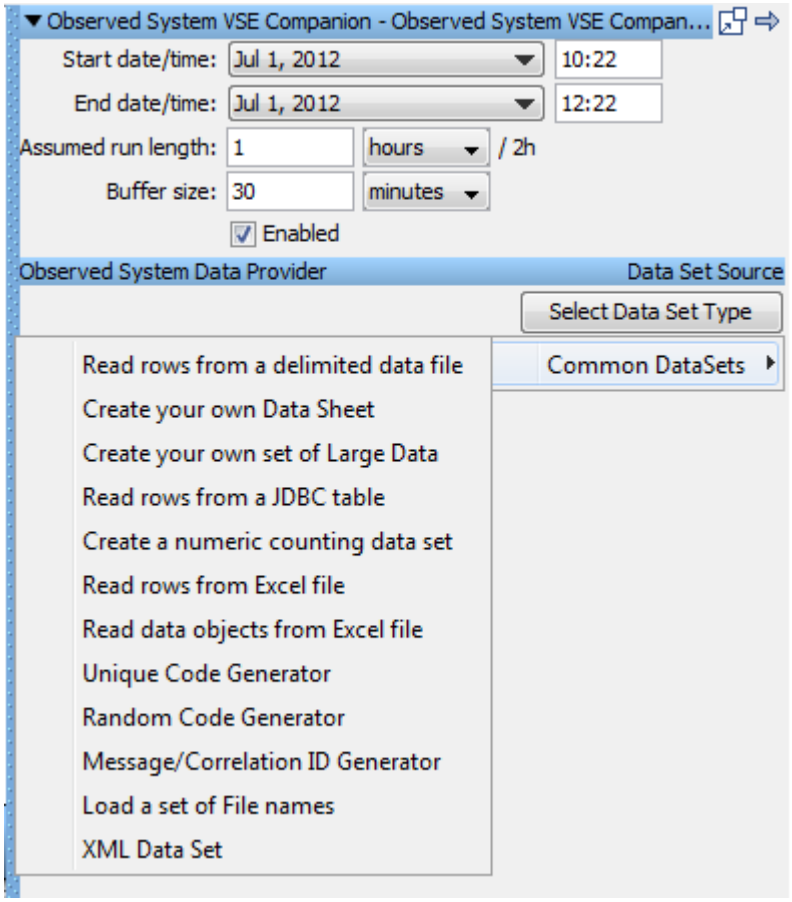
パネルの上部には、コンパニオンのパラメータに関する一般情報が表示されます。

この例では、［開始日時］および［終了日時］には 2 時間のウィンドウが定義されていますが、［想定実行時間］は 1 時間に設定されています。このため、VSE 実行時間のうちの各 1 時間に対して、コンパニオンは、データ セット データ プロバイダからの 2 時間のデータを使用します。 バッファ サイズが 30 分であることは、VSE が 30 分ごとにデータ セットにアクセスしてデータを取得することを意味します。 バッファ サイズはスケーリングの前です。 したがって、30 分のバッファでは、

- VSE は 10:22 と 10:52 の間のデータ セットからデータを取得します。
- VSE は、半分のスケーリング（この例の場合）を実行して、計算を正しく実行することができます。

コンパニオンを一時的に無効にするには、［有効］チェック ボックスをオフにします。

DevTest データ セットおよび Wily 観察対象システム データ プロバイダを含むどのデータ プロバイダでも、観察対象システム コンパニオンをサポートできます。［（ここをクリックして選択）］をクリックし、［データ セット ソース］を選択すると、［データ セット タイプの選択］ボタンが表示されます。［データ セット タイプの選択］をクリックし、［共通データ セット］-［ユーザ定義データ シートの作成］を選択します。



［ユーザ定義データ シートの作成］のテーブルが開かれるときに、あらかじめ以下の項目が列に入力されます。

id

要求の処理時に操作名に対して照合されます。

timestamp

responseTime

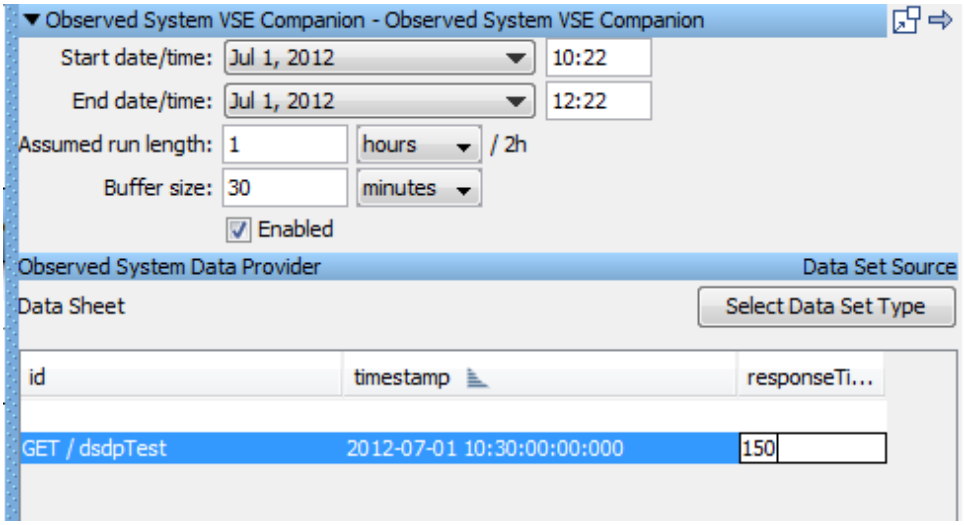
standardDeviation の値が 0 の場合は、responseTime によって、再生時に使用する間隔が定義されます。 ゼロ以外の値については、standardDeviation 列に関して説明する数式を使用します。 この応答時間の計算は、サービス イメージで設定された 15 ミリ秒の反応時間を上書きします。

standardDeviation

応答時間の平均値からの標準偏差を表す統計データを定義します。 再生時に使用される応答時間は、平均応答時間の +/- 3 シグマ内になります。 **x** が responseTime で、**y** が standardDeviation である場合、再生時の応答時間は（**x** -3 **y**）と（**x** +3 **y**）の範囲内のランダムな値です。

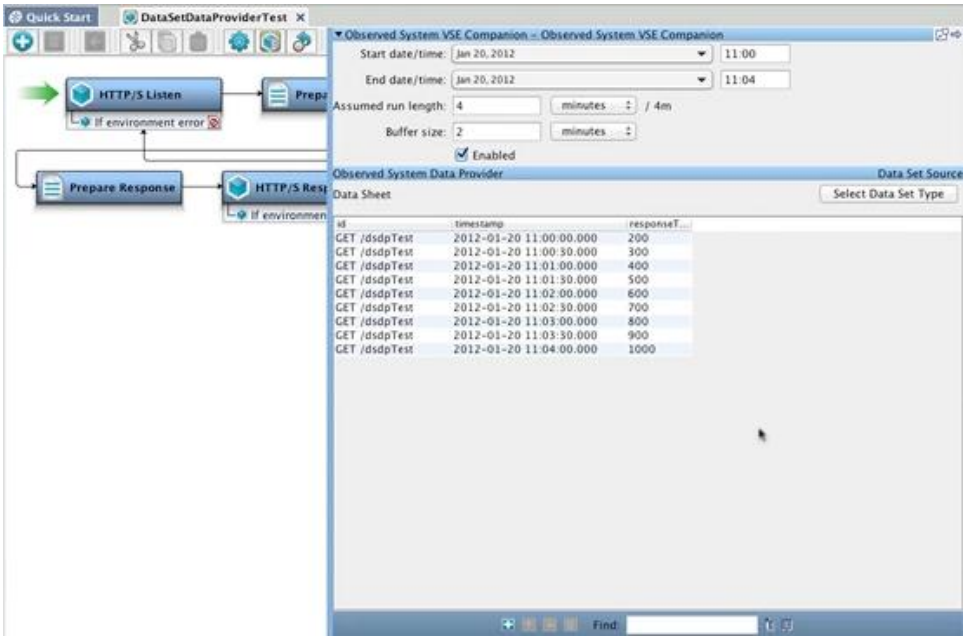
これらの列は、このコンパニオンの情報を提供するために使用する任意のタイプのデータ セット プロバイダに必要です。

この例では、サービス イメージから操作名 GET / dsdpTest をコピーし、［id］フィールドにそれを入力します。［responseTime］フィールドに「150」と入力します。



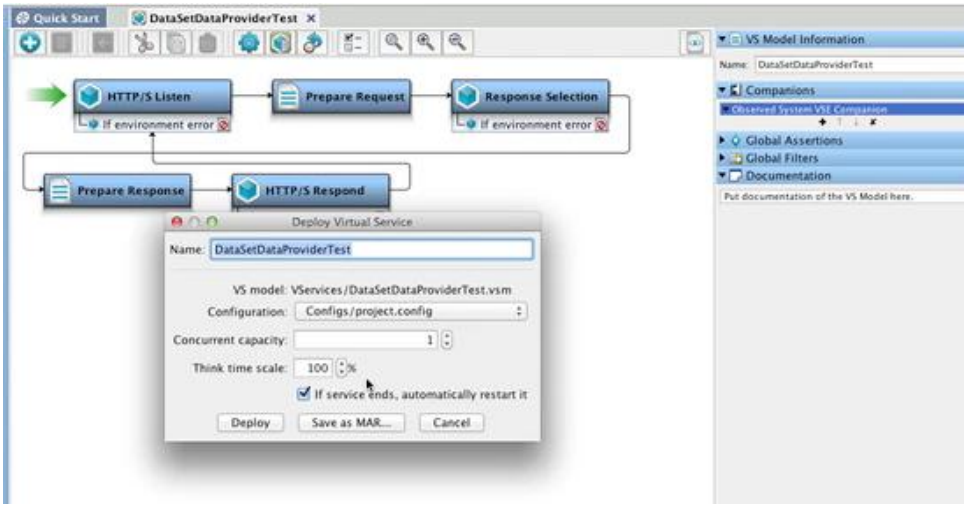
［timestamp］フィールドに、［開始日時］および［終了日時］で定義された時間ウィンドウに入る日時を入力します。

この例を続行するには、あらかじめ入力されたデータ セットのある既存のプロジェクトを使用します。



データ セット応答時間は、30 秒ごとに 100 ミリ秒の増加を定義します。そのため、このコンパニオンは、常に、仮想サービス モデルで設定された 15 ミリ秒の反応時間を上書きする必要があります。

注: 仮想サービスを展開する場合は、［反応時間スケール］に「0%」を入力しないでください。



観察対象のシステム データ プロバイダ（Wily）ソース

観察対象システム VSE コンパニオンは、CA Application Performance Management （Wily）アプリケーションから入力を受信できます。 Wily と連携するコンパニオンを設定するには、パネルの上部の設定情報を入力します。 その後、［（ここをクリックして選択）］をクリックし、［データ セット ソース］に［Wily ソース］を選択します。

Wily 観察対象システム コンパニオンのパラメータは以下のとおりです。

Web サービス URL

Wily Web サービスの URL を指定します。

エージェント正規表現

エージェントを識別する正規表現を指定します。

メトリック正規表現

メトリックを識別する正規表現を指定します。 このメトリック正規表現は、Wily サーバに送信されるメトリック正規表現の中央部分としてラップされます。 サーバに送信されるメトリック正規表現は、次のとおりです。

```
".*WebServices¥|Server¥|<ユーザ入力パターン>:Average Response Time.*"
```

たとえば、メトリックのクエリを実行する場合は、次のようになります。

```
"WebServices¥|Server¥|http_//ejb3¥.examples¥.itko¥.com/¥|(.*)Average Response Time ¥(ms¥)"
```

次のように入力できます。

```
"http_//ejb3¥.examples¥.itko¥.com/"
```

この例は、Web サービスのすべての平均応答時間メトリック を取得します。 特定の操作のメトリックを取得するには、カスタム正規表現を使用します。

デフォルト： ".*"

サービス ユーザ名

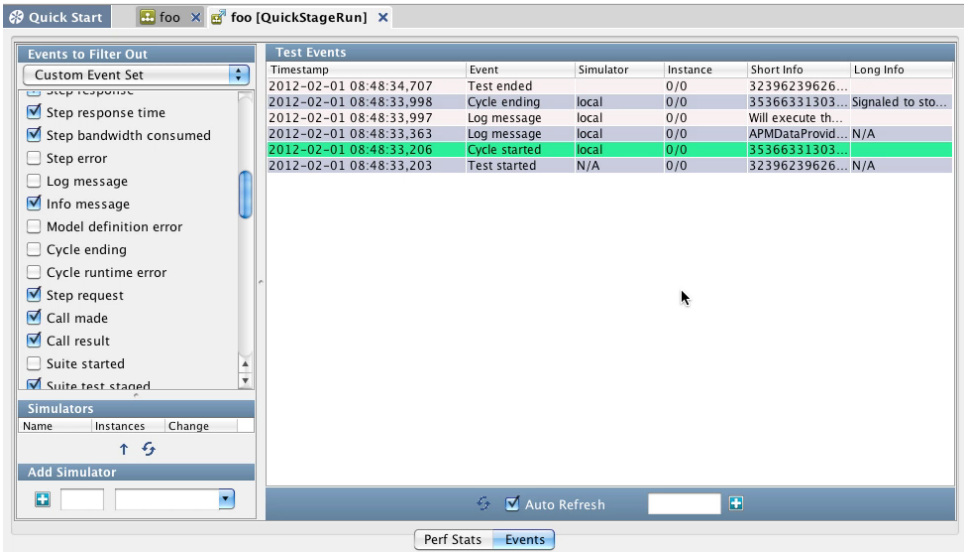
Wily サービスにアクセスするために使用されるユーザ名を指定します。

サービス パスワード

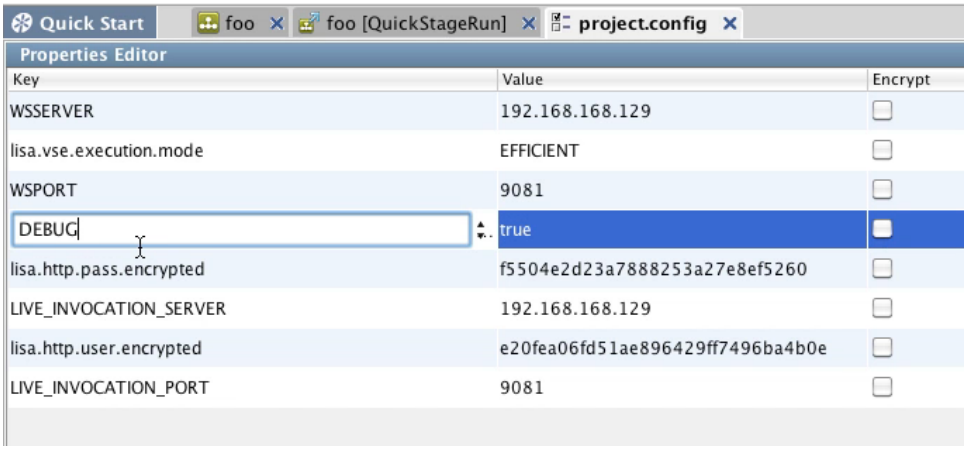
必要に応じて、［サービス ユーザ名］に関連するパスワードを指定します。

コンパニオンの動作をテストするには、クイック テストをステージングします。

この例のテスト ケースには、1 つのステップ（ログ メッセージ出力ステップ）がありました。［Quick Stage Run］（クイック ステージングの実行）ウィンドウの［テスト イベント］タブで出力を表示できます。



このテスト ケースに使用される設定ファイルで debug プロパティを true に設定することは、ログ メッセージが出力に表示されることを意味します。



実行時の優先度

実行時のプロセスは以下のとおりです。

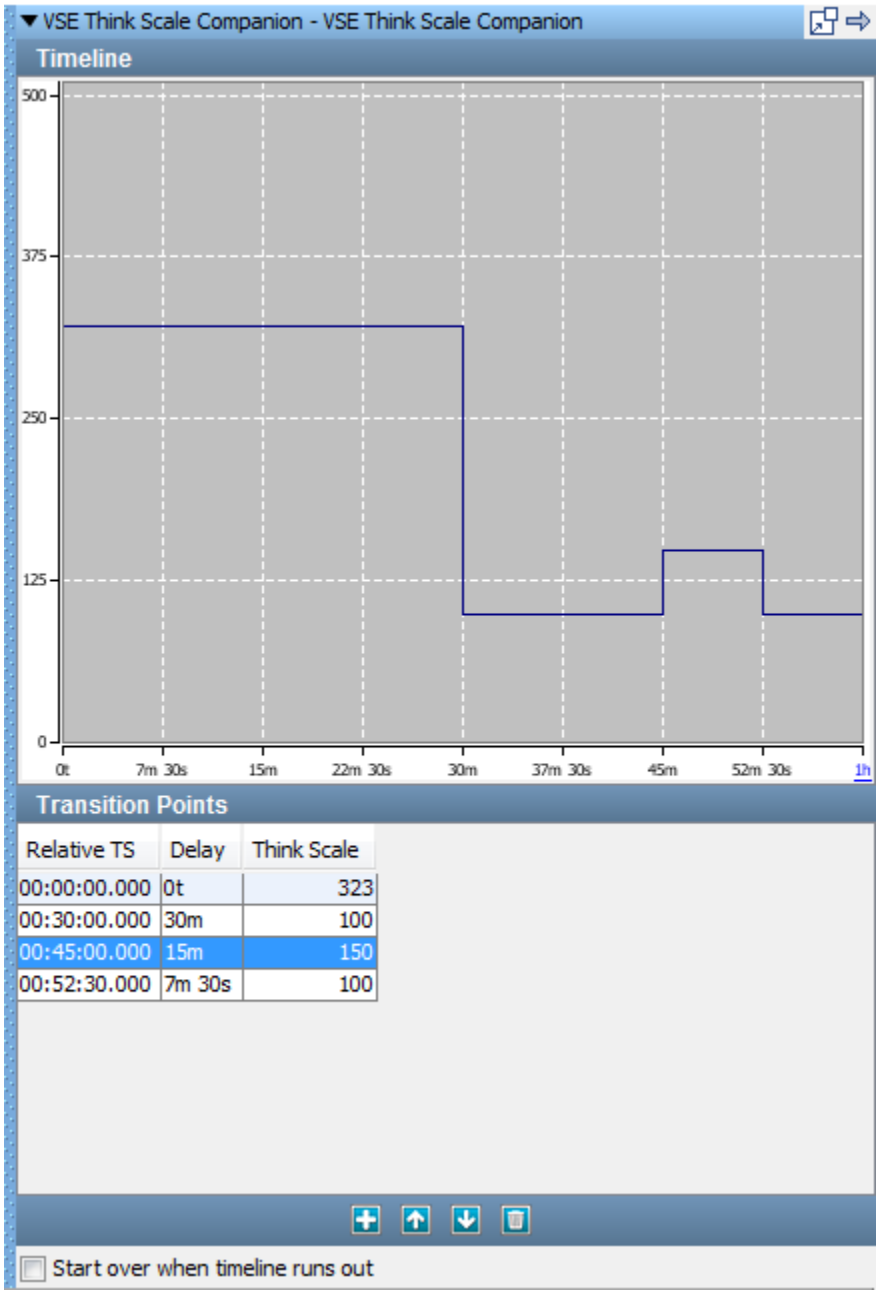
- VSE 応答からの反応時間仕様は、遅延の要因が特定されたときに計算されます。
- 反応時間仕様に状態（二重中かっこの式）が含まれる場合、式は直接評価され、結果は応答の応答時間として使用されます。
- 反応時間仕様に状態が含まれず、仮想サービスに観察対象システム コンパニオンが含まれる場合、コンパニオンは応答の応答時間を決定するように求められます。
- コンパニオンが存在しないか、または応答時間を決定できない場合、反応時間仕様が応答時間として使用されます。

たとえば、想定実行時間が 1 時間、バッファ サイズが 15 分の場合、以下の結果が得られます。

- 最初に 15 分のバッファを取得します。
- 次に、15 ～ 30 分のバッファを取得します。
- 次に、30 ～ 45 分のバッファを取得します。
- 次に、45 ～ 60 分のバッファを取得します。
- 次に、最初の 15 分からデータを再度取得します

VSE 反応時間スケール コンパニオン

VSE 反応時間スケール コンパニオンは仮想サービス モデルに追加できます。 このコンパニオンでは、反応時間スケールの変化のグラフを指定することにより、サービスの反応時間スケールのパーセンテージを時間と共に変化させることができます。



移行ポイントの追加、削除、並び変えを行うには、［追加］、［削除］、［移動］をクリックします。 ［遅延］および［反応時間スケール］のエントリは、直接編集できます。また、タイムライン表示の折れ線をクリックおよびドラッグすることで、目的の遅延およびスケールを表示できます。［移行ポイント］テーブルは、対応して更新されます。

相対 TS

入力した遅延に基づいて計算され、形式は hh:mm:ss.ms です。

遅延

指定された反応時間スケールを適用する前に待機する時間の長さ（分および秒）。

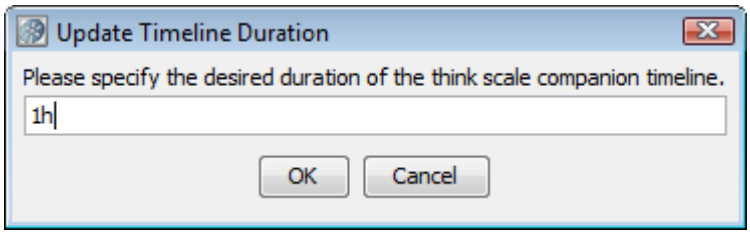
反応時間スケール

反応時間スケールは、応答で反応時間に適用されるパーセンテージです。

タイムラインが終了したらやり直す

タイムラインが終了した場合に、最初の相対 TS から開始します。

水平軸上の右端のティック マークのラベルをクリックすると、コンパニオンのタイムラインの合計を調節できます。 このラベルは、上記のウィンドウでは「1h」というラベルです。 ［タイムライン期間の更新］ダイアログ ボックスが表示されます。 タイムライン期間を移行ポイントよりも短く更新する場合、一部の移行ポイントが削除されるという警告が表示されます。



グラフにカーソルを置くと、ツールヒントで垂直線上に時間が表示され、水平線上に反応時間スケールのパーセンテージが表示されます。

バッチ応答反応時間コンパニオン

仮想サービス モデルが指定された時刻に定義された数の応答を送信するように、バッチ応答反応時間コンパニオンは反応時間スケールを変更します。

このコンパニオンにより、仮想サービス モデルの応答をバッチ処理できます。応答を送信するスケジュールを指定できます。たとえば、毎日 8:15 に 100 の応答を送信できます。 要求はいつでも受信できますが、VSE は 8:15 にのみ応答します。保留中の応答が 100 未満の場合、VSE は応答をすべて送信します。保留中の応答が 100 を超えている場合、VSE は 100 の応答を送信し、次のバッチまで残りを保存します。

VSE は受信要求に応答するだけです。未承諾応答を生成しません。VSE は、12 の要求を受信すると、［Quantity （数量）］が 12 より多くても 12 の応答を送信します。

このコンパニオンは SAP トランスポート プロトコルにのみ役立ちます。

コンパニオン エディタで以下のパラメータを入力します。

有効

コンパニオンを有効にするかどうかを指定します。

時間

いつ応答を送信するかを定義します。

値： 00:00 ～ 23:59

Quantity（**数量**）

送信する応答の最大数を定義します。

［Response Schedule（応答スケジュール）］テーブル内の行を追加、移動、削除するには、［追加］、［上へ］、［下へ］、［削除］をクリックします。

Repeat daily（**毎日繰り返す**）

毎日応答を送信するかどうかを指定します。

Send all at once（**一括送信**）

応答を配信する方法を指定します。

値

- **オン：** 応答を一度にすべて送信するか、指定した間隔で、サイズ制限に達するまで、要求が到達したときに送信します。
- **オフ：** 現在のバッチの時間と次のバッチの時間またはスケジュールされた終了時間までの間に応答を均等に配分します。

Schedule End Time（**スケジュール終了時刻**）

［Send all at once （一括送信）］チェック ボックスがオフの場合、このフィールドは［Response Schedule （応答スケジュール）］テーブル内の最後にスケジュールされた時間の終了時間を定義します。

値： 00:00 ～ 23:59

制限： この値は、［Response Schedule （応答スケジュール）］テーブル内の最後にスケジュールされた時間より後にする必要があります。

例:

［Response Schedule （応答スケジュール）］テーブルに 2 つのエントリが含まれていると仮定します。

- 最初のエントリの値は 08:00 および 10 です。
- 2 番目のエントリの値は 08:10 および 10 です。

また、［Send all at once （一括送信）］チェック ボックスがオフで、［Schedule End Time （スケジュール終了時刻）］フィールドが 08:15 に設定されていると仮定します。

仮想サービス モデルを展開した場合、この設定の結果は以下のようになります。

- 08:00 ～ 08:10 の間は 1 分間に 1 つの応答。
- 08:10 ～ 08:15 の間は 30 秒間に 1 つの応答。

これらの結果では、サービスがその期間中に一致する応答がある十分な要求を受信すると仮定します。

繰り返し期間反応時間コンパニオン

繰り返し期間反応時間コンパニオンは、指定するすべての期間に対して応答をバッチで送信するために応答反応時間を調整します。

このコンパニオンにより、仮想サービス モデルの応答をバッチ処理できます。応答を送信するスケジュールを指定できます。たとえば、1 時間ごとに 100 の応答を送信できます。要求はいつでも受信できますが、VSE はその時間にのみ応答します。保留中の応答が 100 未満の場合、VSE は応答をすべて送信します。保留中の応答が 100 を超えている場合、VSE は 100 の応答を送信し、次のバッチまで残りを保存します。

VSE は受信要求に応答するだけです。未承諾応答を生成しません。VSE は、12 の要求を受信すると、［Quantity（数量）］が 12 より多くても 12 の応答を送信します。

このコンパニオンは SAP トランスポート プロトコルにのみ役立ちます。

コンパニオン エディタで以下のパラメータを入力します。

有効

コンパニオンを有効にするかどうかを指定します。

開始

タイマがいつ開始するかを定義します。

値

- 即時
- On the Quarter Hour（15 分刻み）
- On the Half Hour（30 分刻み）
- On the Hour（1 時間刻み）

間隔

応答の送信間に待機する分数または時間数を定義します。

Quantity（数量）

送信する応答の最大数を定義します。

Repeat daily（毎日繰り返す）

応答の送信を毎日繰り返すかどうかを指定します。

Send all at once（一括送信）

応答を配信する方法を指定します。

値

- **オン：** 応答を一度にすべて送信するか、サイズ制限に達するまで要求が到達したときに送信します。
- **オフ：** 設定された期間に応答を均等に配分します。

例:

［開始］フィールドが［即時］に設定され、［間隔］フィールドが 5 分に設定されていると仮定します。 また、［Quantity（数量）］フィールドが 10 に設定され、［Send all at once（一括送信）］チェック ボックスがオフになっていると仮定します。

仮想サービス モデルを展開した場合、この設定の結果は 30 秒ごとに 1 つの応答になります。 この結果では、サービスがその期間中に一致する応答がある十分な要求を受信すると仮定します。

各テストのサンドボックス クラス ローダの作成コンパニオン

サンドボックス クラス ローダの作成コンパニオンでは、すべてのテスト ランがそれ自体の Java クラス ローダ（JVM）で実行されることを検証できます。 マルチスレッドまたはマルチユーザ アクセスで設計されていないローカル Java オブジェクトをテストする場合、このコンパニオンは非常に有用です。 大半のテストは、このコンパニオンを必要としません。 これは通常、動的 Java 実行ステップでローカル Java オブジェクトをテストする場合にのみ必要です。

クラス ローダ サンドボックスコンパニオンを設定するには、クラス パス サンドボックス コンパニオン エディタを使用します。

- 実行するクラスが hotDeploy ディレクトリにある場合は、［ホット デプロイ パス エントリを追加］チェック ボックスをオンにします。

実行するクラスが hotDeploy ディレクトリにない場合は、［追加］ をクリックして［クラス パス ディレクトリ］リストに行を追加し、適切なクラス パスを追加します。

注: 編集または実行する Java オブジェクトは、クラス パスまたは hotDeploy ディレクトリにある**必要があります**。 これらを LISA lib または bin ディレクトリに配置**しないでください**。 Java VM がクラスをロードする方法が原因で、クラス ローダ サンドボックスが動作しなくなります。

実行する最終ステップの設定コンパニオン

実行する最終ステップの設定コンパニオンでは、テスト中のシステムがテストの結果にかかわらず整合性のある状態にあることを検証できます。通常のテスト フローが回避された場合に、どのステップが常に最後に実行されるかを指定します。

一般的な使用方法は、リソースがテストの最後に解放されるようにすることです。最初のステップの指定は通常必要ではありませんが、最終ステップの指定が重要なシナリオが多くあります。

テスト ケースが終了ステップに到達するか、テスト ケースを終了するように指示された場合でも、最終ステップが実行されます。

最終ステップは、ITR の［実行履歴］リストには表示されません。結果は、実行された最後のステップの［イベント］タブで確認できます。

最終ステップの設定コンパニオン エディタは、最終ステップを設定するために使用されます。

最終ステップの設定コンパニオンを設定するには、以下のパラメータを入力します。

最終ステップ

ドロップダウン リストから実行する最終ステップを選択します。

実行する最終ステップとして指定されるステップは、緑のフラグ アイコンで示されます。

ネガティブ テスト コンパニオン

すべてのステップを失敗させる場合に、ネガティブ テスト コンパニオンは役立ちます。 含まれている任意のテスト ステップが成功した場合に、通常終了するテストケースを失敗させるには、このコンパニオンを使用します。

このコンパニオンには、設定パラメータはありません。

失敗テスト ケース コンパニオン

失敗テスト ケース コンパニオンは、任意のテスト ステップのアサートがエラーになった場合に、正常終了したテストケースを失敗としてマークします。 EVENT_TRANSFAILED イベントのイベント リスナは、このコンパニオンに登録されます。 この使用方法の例には、Web サービス ステップのエラーに対する WSDL 検証アサーションがあります。検証の失敗をレポートする一方で、テスト ケースを続行したい場合が考えられます。

このコンパニオンには、設定パラメータはありません。

XML 差分除外ノード コンパニオン

XML 差分除外ノード コンパニオンでは、［コンパニオン］タブにリスト表示されている 1 つ以上のノードを返す XPath 式を入力できます。これらのノードは OR 演算され、このテスト ケースのすべての XML 差分比較の実行中に無視されます。

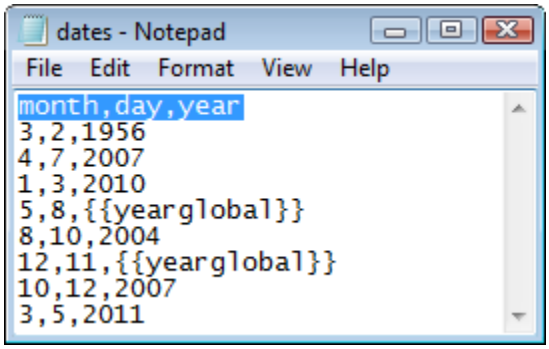
データ セットの説明

このセクションには、以下のデータ セットの説明が含まれています。

区切りデータ ファイルからの読み取りデータ セット

区切りデータ ファイルからの読み取りデータ セットは、テキスト ファイルのコンテンツに基づいてプロパティに値を割り当てます。 これは、DevTest Solutions で最も一般的に使用されるデータ セットのタイプです。テキスト ファイルの最初の行では、データ値が格納されるプロパティの名前を指定します。後続の行では、これらのプロパティで使用されるデータ値がリストされています。テキスト ファイルは、単純なテキスト エディタを使用して作成されます。

以下の例では、カンマ区切りのデータ ファイルを示します。最初の行は、このデータ セットのプロパティ名を示します。



データ セット エディタは、データ セットを定義するために使用されます。

以下のパラメータを入力します。

名前

データ セットの名前を入力します。

ローカル

グローバル データ セットとして機能させるか、またはローカル データ セットとして機能させるかを指定します。デフォルトは [グローバル] です。 ローカル データ セットはシミュレータごとに 1 つ作成されます。グローバル データ セットは 1 回作成され、すべてのシミュレータで共有されます。

ランダム

現在のレコードの次のレコード（シーケンシャル アクセス）が読み取られるか、またはランダムなレコードが読み取られるか。 シーケンシャル読み取りがデフォルトです。

取得する最大レコード

ランダム アクセスの場合に取得するレコード数の上限。 [ランダム] チェック ボックスがオンになっていない場合、このテキスト フィールドは無効です。

データが終了した場合

データ セットの終了に対応するアクションを選択します。 データ セットの最初からもう一度値を読み取るか、または実行するステップを選択できます。

ファイルの場所

テキスト ファイルのフル パス名、または[参照]ボタンを使用してファイルを参照します。

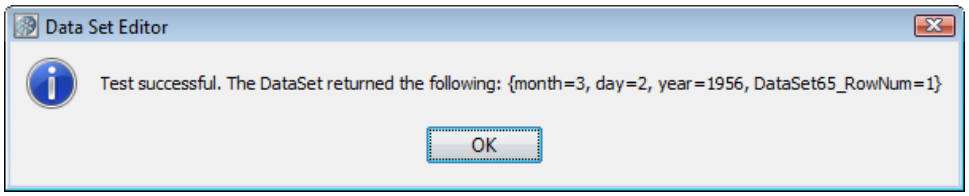
ファイル エンコーディング

デフォルトの UTF-8 エンコーディングを使用するか、またはドロップダウン リストから代わりのエンコーディングを選択します。 [自動検出] を選択して [検出] ボタンをクリックし、選択されたエンコーティング タイプを表示することもできます。

区切り文字

使用する区切り文字。 任意の文字を区切り文字として使用できます。 ドロップダウン リストには、一般的な区切り文字が含まれています。

[テスト アンド キープ] ボタンをクリックして、データをテストおよびロードします。 データ セットを読み取れたことを示す確認メッセージが表示され、データの最初のセットが表示されます。



ユーザ定義データ シートの作成データ セット

データ シート データ セットでは、外部ファイルへの参照を必要とすることなく、DevTest でデータ セット データを生成できます。

データ シートはデータ テーブルから構成されます。 列見出しはプロパティ名を指定し、テーブル行はそれらのプロパティのデータ値を指定します。 テーブル スケルトンは、行数および列名（プロパティ）を指定することにより作成されます。 デフォルトの名前は、**ユーザ定義データ シートの作成**です。 作成される後続のデータ シートには、データ シート名に「~番号」が追加されます。

以下のパラメータを入力します。

名前

データ セットの名前を入力します。

ローカル

グローバル データ セットとして機能させるか、またはローカル データ セットとして機能させるかを指定します。デフォルトは［グローバル］です。 ローカル データ セットはシミュレータごとに 1 つ作成されます。グローバル データ セットは 1 回作成され、すべてのシミュレータで共有されます。

ランダム

現在のレコードの次のレコード（シーケンシャル アクセス）が読み取られるか、またはランダムなレコードが読み取られるか。 シーケンシャル読み取りがデフォルトです。

取得する最大レコード

ランダム アクセスの場合に取得するレコード数の上限。 ［ランダム］チェック ボックスがオンになっていない場合、このテキスト フィールドは無効です。

データが終了した場合

データ セットの終了に対応するアクションを選択します。 データ セットの最初からもう一度値を読み取るか、または実行するステップを選択できます。

行数

行数の初期予測値。 この値は後で変更できます。

列名

列名のカンマ区切りリスト。 これらの名前はプロパティ名でもあります。

［データ シート スケルトンの作成］ボタンをクリックし、テーブルにデータを入力します。

昇順または降順で並べ替えるには、列ラベルをクリックします。 列メニューを表示するには、列ラベルを右クリックします。

暗号化列

列の値をすべて暗号化するには、このオプションを選択します。 暗号化された列にはロック アイコンが表示されます。また、すべての値が並んだアスタリスクとして表示されます。 エクスポートでは、列は <名前>_enc で、データは暗号化されて表示されます。

列名の変更

列の名前を変更するには、このオプションを選択します。

昇順にソート

列の値を昇順でソートするには、このオプションを選択します。 ソートを降順に変更するには、再度クリックします。

ソートのリセット

列の値を元の状態にソートするには、このオプションを選択します。

列のサイズ変更モード

［自動］、［後続の列］、［次の列］、［最後の列］、または［手動］を選択します。

すべての列を最大化

このオプションを選択すると、［列のサイズ変更モード］が［手動］に変更されます。 列はすべて表示されます。また、スクロール バーが使用できます。

テーブルの作成および変更を行うには、下部のツールバーの機能を使用します。

追加

テーブルに行を追加します。

上へ/下へ

行を選択し、テーブル内の上または下へ移動します。

削除

選択した行を削除します。

列の追加

テーブルに列を追加します。

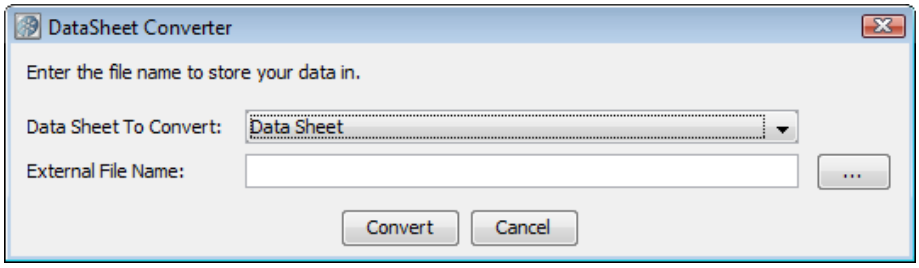
列の削除

列を削除します。 削除する列のセルを選択し、セルが編集のために選択されたことを確認し、このボタンをクリックします。

［テスト アンド キープ］ボタンをクリックして、データをテストおよびロードします。 データ セットを読み取れたことを示す確認メッセージが表示され、データの最初のセットが表示されます。

以下のアクションも実行できます。

- 行をソートするには、列見出しをダブルクリックします。
- 列の名前を変更するには、列の名前を右クリックします
- ツールバーの機能を選択するか、または拡張ビューの起動を行ってセルを編集するには、セルを選択して右クリックします。
- データ セットまたはファイルにデータ シートを変換するには、パネルの下部の［データ シートへの変換］をクリックします。



変換するデータ シート

デフォルトでは、作成またはプルダウン リストから選択したデータ シート。

外部ファイル名

データ シートから作成するファイルの名前およびパス。

注: テーブル内で行を上または下へ移動させると、テストの結果に影響する場合があります。 列の順番は、テストの結果に影響しません。

データ シートを使用するステップを選択します。

ユーザ定義の大規模データ セットの作成データ セット

ユーザ定義の大規模データ セットの作成データ セットでは、任意の大きさのカスタム データ テーブルを定義できます。 データの行および列の数は任意に設定できます。 バッキング ファイル名は、データがすべて格納されるファイルです。

以下のパラメータを入力します。

名前

データ セットの名前を入力します。

ローカル

グローバル データ セットとして機能させるか、またはローカル データ セットとして機能させるかを指定します。デフォルトは［グローバル］です。 ローカル データ セットはシミュレータごとに 1 つ作成されます。グローバル データ セットは 1 回作成され、すべてのシミュレータで共有されます。

ランダム

現在のレコードの次のレコード（シーケンシャル アクセス）が読み取られるか、またはランダムなレコードが読み取られるか。 シーケンシャル読み取りがデフォルトです。

取得する最大レコード

ランダム アクセスの場合に取得するレコード数の上限。 ［ランダム］チェック ボックスがオンになっていない場合、このテキスト フィールドは無効です。

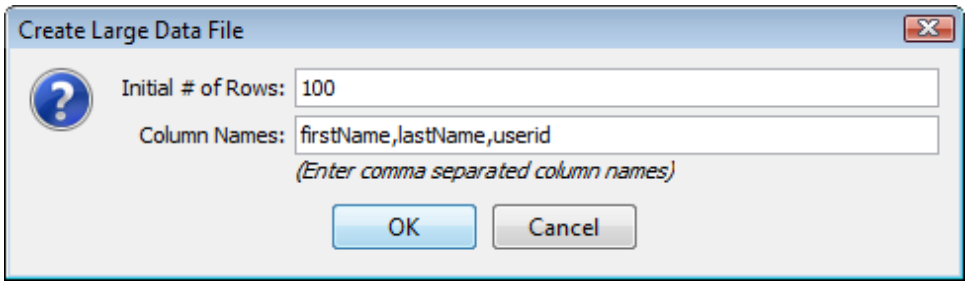
データが終了した場合

データ セットの終了に対応するアクションを選択します。 データ セットの最初からもう一度値を読み取るか、または実行するステップを選択できます。

バッキング ファイル名

データが格納されるファイルの名前。 このファイルは自動的に作成されます。ユーザが提供するデータはこのファイルに挿入されます。

［作成］ボタンは、ファイル作成のために追加のパラメータを指定できるパネルを開きます。

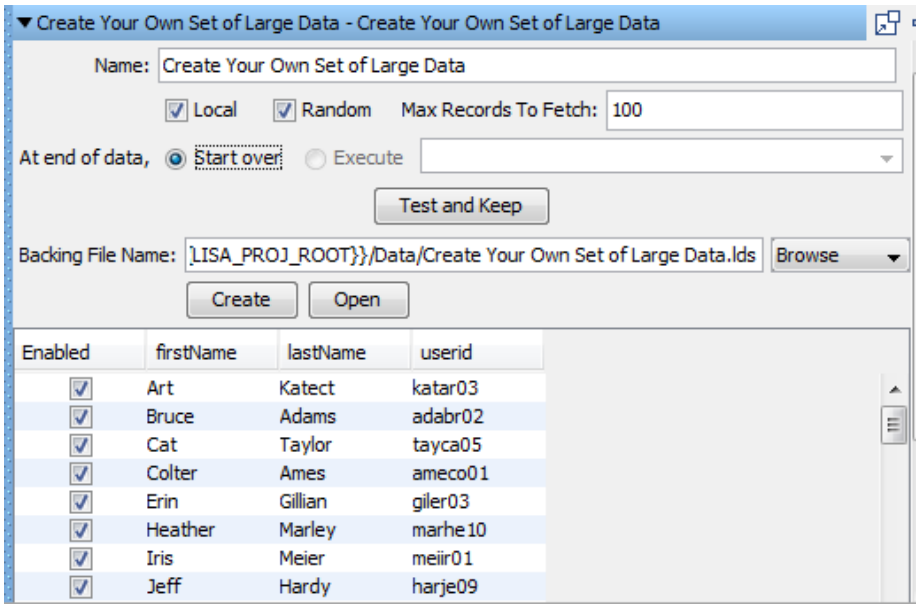


行の初期数

作成する行の初期数を定義します。エディタを使用して行を追加することができます。

列名

データ セットに入力されるカンマ区切りの列名を定義します。



列にデータを入力した後に［テスト アンド キープ］ボタンを押すと、作成されたバッキング ファイルにデータがコピーされます。

JDBC テーブルからの読み取りデータ セット

JDBC テーブルからの読み取りデータ セットは、データベースからソース テスト ケース データを読み取るために使用されます。データは JDBC ドライバ（ユーザが提供する必要がある）を使用して読み取られます。 データのテーブル内の各列は、プロパティとして表されます。このデータ セットは SQL クエリから返された行すべてをループします。

以下のパラメータを入力します。

名前

データ セットの名前を入力します。

ローカル

グローバル データ セットとして機能させるか、またはローカル データ セットとして機能させるかを指定します。デフォルトは［グローバル］です。 ローカル データ セットはシミュレータごとに 1 つ作成されます。グローバル データ セットは 1 回作成され、すべてのシミュレータで共有されます。

ランダム

現在のレコードの次のレコード（シーケンシャル アクセス）が読み取られるか、またはランダムなレコードが読み取られるか。 シーケンシャル読み取りがデフォルトです。

取得する最大レコード

ランダム アクセスの場合に取得するレコード数の上限。 ［ランダム］チェック ボックスがオンになっていない場合、このテキスト フィールドは無効です。

データが終了した場合

データ セットの終了に対応するアクションを選択します。 データ セットの最初からもう一度値を読み取るか、または実行するステップを選択できます。

ドライバ クラス

適切なドライバ クラスの完全なパッケージ名を入力または選択します。標準のドライバ クラスは、プルダウン メニューで使用可能です。

接続文字列

接続文字列はデータベースの標準の JDBC URL です。URL を入力または選択します。プルダウン メニューには、共通データベース マネージャの JDBC URL テンプレートが含まれています。

ユーザ ID

ユーザ ID を入力します（データベースで必要な場合）。

パスワード

パスワードを入力します（データベースで必要な場合）。

SQL クエリ

データ セットを作成するために使用される SQL クエリ。

［テスト アンド キープ］ボタンをクリックして、データをテストおよびロードします。 データ セットを読み取れたことを示す確認ダイアログ ボックスが表示され、データの最初のセットが表示されます。

数値カウント データ セットの作成データ セット

数値カウント データ セットの作成データ セットは、プロパティに数値を割り当てます。 割り当てられた数値は、与えられた値で開始され、データ セットが使用されるごとに既定の上限を超えるまで一定の増分で変化します。 このデータ セットは、「for」ループをシミュレートするため、または何かが発生した回数を設定するために使用されます。 たとえば、同じステップを 100 回コールする場合が考えられます。以下の例では、数値カウント データ セットの作成データ セットを使用して、これを行う方法を示します。

以下のパラメータを入力します。

名前

データ セットの名前を入力します。

ローカル

グローバル データ セットとして機能させるか、またはローカル データ セットとして機能させるかを指定します。デフォルトは［グローバル］です。 ローカル データ セットはシミュレータごとに 1 つ作成されます。グローバル データ セットは 1 回作成され、すべてのシミュレータで共有されます。

ランダム

現在のレコードの次のレコード（シーケンシャル アクセス）が読み取られるか、またはランダムなレコードが読み取られるか。 シーケンシャル読み取りがデフォルトです。

取得する最大レコード

ランダム アクセスの場合に取得するレコード数の上限。 ［ランダム］チェック ボックスがオンになっていない場合、このテキスト フィールドは無効です。

データが終了した場合

データ セットの終了に対応するアクションを選択します。 データ セットの最初からもう一度値を読み取るか、または実行するステップを選択できます。

プロパティ キー

カウンタ値が格納されるプロパティの名前。

開始

初期カウンタ値。

終了

最終カウンタ値。

増分

カウンタのステップ増分。 カウンタ データ セットは、マイナスの増分を割り当てることにより、逆にカウントするために使用できます。

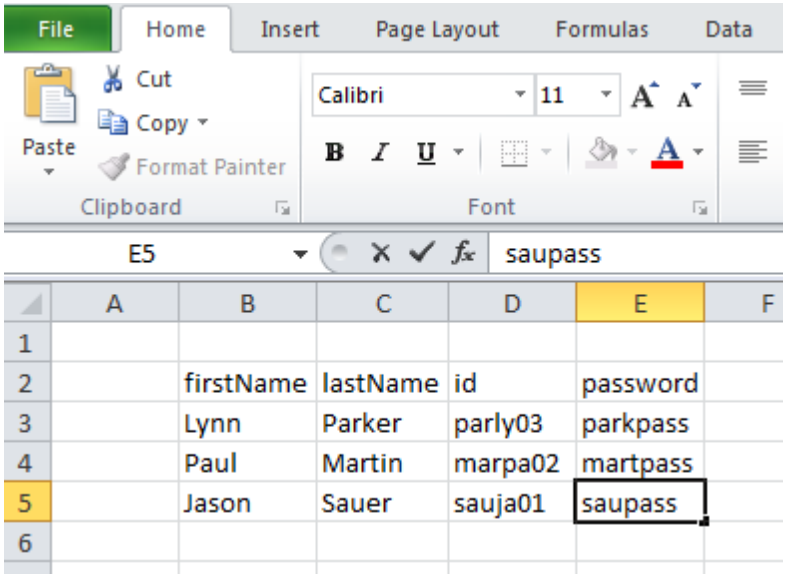
［テスト アンド キープ］ボタンをクリックして、データをテストおよびロードします。 データ セットを読み取れたことを示す確認メッセージが表示され、データの最初のセットが表示されます。

Excel ファイルからの読み取りデータ セット

Excel ファイルからの読み取りデータ セットは、Excel スプレッドシートのコンテンツに基づいてプロパティに値を割り当てます。

Excel スプレッドシートの最初の空白でない行では、データ値が割り当てられるプロパティの名前を指定します。 後続の行では、これらのプロパティに使用されるデータ値をリストします。 空白のセルばかりの最初の行は、データの終端として扱われます。

たとえば、名、姓、ユーザ ID、およびパスワードの組み合わせのセットをテストすることが考えられます。



	A	B	C	D	E	F
1						
2		firstName	lastName	id	password	
3		Lynn	Parker	parly03	parkpass	
4		Paul	Martin	marpa02	martpass	
5		Jason	Sauer	sauja01	saupass	
6						

以下のパラメータを入力します。

名前

データ セットの名前を入力します。

ローカル

グローバル データ セットとして機能させるか、またはローカル データ セットとして機能させるかを指定します。デフォルトは［グローバル］です。 ローカル データ セットはシミュレータごとに 1 つ作成されます。グローバル データ セットは 1 回作成され、すべてのシミュレータで共有されます。

ランダム

現在のレコードの次のレコード（シーケンシャル アクセス）が読み取られるか、またはランダムなレコードが読み取られるか。 シーケンシャル読み取りがデフォルトです。

取得する最大レコード

ランダム アクセスの場合に取得するレコード数の上限。 ［ランダム］チェック ボックスがオンになっていない場合、このテキスト フィールドは無効です。

データが終了した場合

データ セットの終了に対応するアクションを選択します。 データ セットの最初からもう一度値を読み取るか、または実行するステップを選択できます。

ファイルの場所

Excel ファイルのフル パス名を入力するか、または［参照］ボタンを使用してファイルを参照します。パス名にはプロパティを使用できます（例： LISA_HOME）。

シート名

Excel スプレッドシートのシート名を入力します。

［テスト アンド キープ］ボタンをクリックして、データをテストおよびロードします。 データ セットを読み取れたことを示す確認メッセージが表示され、データの最初のセットが表示されます。

データ セットを使用するステップを選択します。

［XLS ファイルを開く］ボタンをクリックすると、Excel スプレッドシートを開いて編集できます。

注: Excel ファイル データ セットおよび Excel DTO データ セットは、Excel 2007 以降のスプレッドシートを使用できます。Excel DTO データ セットは、現在も XLS 形式を使用して作成されています。

Excel ファイルからの DTO の読み取りデータ セット

Excel ファイルからの DTO の読み取りデータ セットでは、テスト ステップで Java データ転送オブジェクト（DTO）をパラメータ化できます。データ セットは、Excel スプレッドシートを使用してこれらのパラメータのデータ値を指定する簡単な方法を提供します。

Excel ファイルからの DTO の読み取りデータ セットは、DTO のプロパティに値を割り当て、オブジェクトをプロパティに格納します。 DTO がパラメータとして必要な場合は、いつでもこのプロパティを使用できます。データ セット内のデータには、数値または文字列のような単純なデータ型、あるいは DTO、配列、およびコレクションなどの複雑なデータ型を使用できます。 Excel で表されるデータは、必要に応じて適切なデータ型に自動的に変換されます。このデータ セットを使用する場合の複雑な点は、最初の Excel スプレッドシート作成です。 幸い、この作成は完了しています。 DTO のパッケージ名に応じて、オブジェクトを表す 1 つ以上の Excel シートを使用してテンプレートが作成されます。プリミティブ、文字列、プリミティブの配列、および単純な個別の DTO の配列などのデータ型は、単一のシートで表すことができます。 オブジェクトの配列などのより複雑なデータ型は、完全な DTO を表すためにより多くの Excel シートを必要とします。

多くの場合、Web サービス エンドポイントは複雑な DTO を想定しています。 Excel データ セットは、Web サービスへのパラメータとして使用するオブジェクトの作成を容易にします。 Web サービスが最初に参照されると、WSDL の名前および URL が与えられます。 **com.lisa.wsgen.SERVICENAME.OBJECTNAME** 形式の Java DTO クラスが自動的に生成され、クラスパスで使用可能になります。 DTO クラス ブラウザで生成されたクラスを参照し、Excel ファイルを生成して、テンプレートに簡単に記入することができます。 以下の例を参照してください。

データ セットの作成は 2 段階のプロセスです。 まず、DevTest に Excel でテンプレートを作成させます。 次に、Excel スプレッドシートを開き、作成されたすべてのシート内のデータ フィールドに入力します。

テンプレートを作成する方法

1. データ セット エディタで以下のパラメータを入力します。

name

データ セットの名前。 この名前は、現在の DTO オブジェクトを格納するために使用されるプロパティになります。

ローカル

グローバル データ セットとして機能させるか、またはローカル データ セットとして機能させるかを指定します。デフォルトは［グローバル］です。 ローカル データ セットはシミュレータごとに 1 つ作成されます。グローバル データ セットは 1 回作成され、すべてのシミュレータで共有されます。

ランダム

現在のレコードの次のレコード（シーケンシャル アクセス）が読み取られるか、またはランダムなレコードが読み取られるか。 シーケンシャル読み取りがデフォルトです。

取得する最大レコード

ランダム アクセスの場合に取得するレコード数の上限。 ［ランダム］チェック ボックスがオンになっていない場合、このテキスト フィールドは無効です。

データが終了した場合

データ セットの終了に対応するアクションを選択します。 データ セットの最初からもう一度値を読み取るか、または実行するステップを選択できます。

ファイル

完全修飾パス名、または参照プルダウン メニューを使用して Excel ファイルを参照します。

DTO クラス名

完全なパッケージ名、または DTO オブジェクトを参照します。クラス ファイルは、テスト マネージャ上にある必要があります。 クラスは、hotDeploy ディレクトリにコピーしてテスト マネージャ上に配置します。

［詳細設定］では、以下を指定できます。

フラット化された子プロパティ表記を生成中に使用する

Excel DTO データ セットの生成中に、フラット化された子プロパティの上書きを選択します。 オフにした場合、子プロパティは固有のワークシートによる参照として生成されます。

フラット化されたプロパティに対して新しい空のセルのセマンティックを使用する

フラット化されたプロパティに対する空のセルのセマンティックとは、DTO スプレッドシートで空のセルが解釈される方法を意味します。 たとえば、フラット化されたプロパティ

`{ "prop1.subprop1"、"prop1.subprop2" }` は、

subprop1 および subprop2 の両方に空のセル値がある場合、新しいセマンティックでは「prop1」への参照は NULL に設定されます。 古いセマンティックでは、prop1 は NULL ではありませんが、subprop1 および subprop2 への参照は両方とも NULL です。 新しいセマンティックはデフォルトで使用されます。これは特に、Nil 不可型を使用する WSDL による Web サービスでは使用される必要があります。 Nil 不可型の場合に使用しないと、DTO スプレッドシートを読み取るときに自動的に作成されるプロパティが含まれているので、中間の NULL でない参照が、スキーマに従って無効な XML を作成する場合があります（セル値が空であるため）。

1. ［テンプレートの生成］ボタンをクリックします。 DevTest がテンプレートを作成し、システム メッセージによって、ファイルがいつ作成されたかがレポートされます。
2. ［XLS ファイルを開く］ボタンをクリックします。

   スプレッドシートには、オブジェクトを構築するために必要なものがすべて含まれます。 次のセクションでは、データを追加する方法について説明します。
3. XLS ファイルを閉じます。
4. ［テスト アンド キープ］ボタンをクリックして、データをテストおよびロードします。データ セットを読み取れたことを示す確認ウィンドウが表示され、データの最初のセットが表示されます。

Excel スプレッドシートの作成

説明をわかりやすくするため、実際の DTO オブジェクト com.itko.example.dto.Customer を使用します。 このクラスは DevTest のサンプルに含まれており、［参照］ボタンを使用してテスト マネージャ を参照すると見つけることができます。

Customer DTO には以下のプロパティがあります。

プロパティ名	タイプ
balance	Double
id	int
name	String
poAddr	アドレス

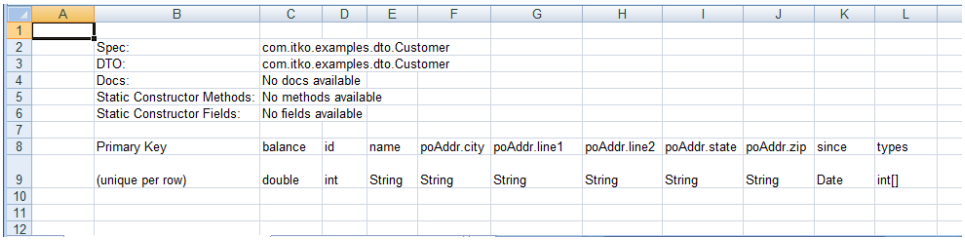
since	日付
types	int[]
locations	Address[]

Address DTO には以下のプロパティがあります。

プロパティ名	タイプ
city	String
line1	String
line2	String
状態	String
zip	String

最初の 6 つの Customer DTO プロパティが 1 つの Excel スプレッドシートに表示されます。 ただし、locations プロパティ（Address オブジェクトの配列）は別の Excel スプレッドシートを必要とします。

DevTest は 1 番目のスプレッドシートの上部に、DTO 仕様（Customer）および現在の DTO オブジェクト（Customer）をリストします。 使用可能な場合、スプレッドシートには Java ドキュメントの場所がリストされます。以下の図は、プロパティ名を指定する行に続けてデータ型を指定する行があるデータ シートを示しています。 1 番目のフィールド（列）は、DTO プロパティではなく特別なフィールド（プライマリ キー）で、各行に対する一意の値を保持します。

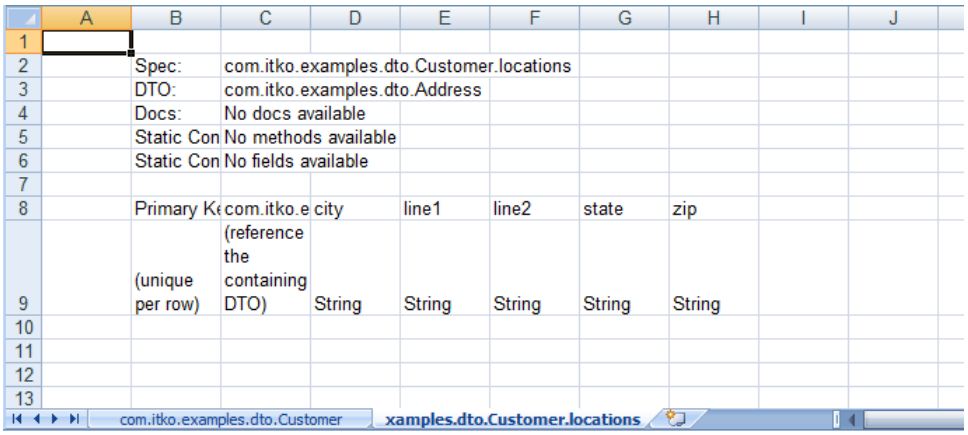


	A	B	C	D	E	F	G	H	I	J	K	L
1												
2		Spec:	com.itko.examples.dto.Customer									
3		DTO:	com.itko.examples.dto.Customer									
4		Docs:	No docs available									
5		Static Constructor Methods:	No methods available									
6		Static Constructor Fields:	No fields available									
7												
8		Primary Key	balance	id	name	poAddr.city	poAddr.line1	poAddr.line2	poAddr.state	poAddr.zip	since	types
9		(unique per row)	double	int	String	String	String	String	String	String	Date	int[]
10												
11												
12												

データ シートには、以下の項目があります。

- 行にはそれぞれ、単一の Customer DTO のデータが含まれます。
- 型が Address の poAddr プロパティは、「フラット化」されています。また、そのプロパティがこのシートにリストされています。 これらのプロパティは、先頭に poAddr が付けられ、それに続いてアドレス オブジェクトのプロパティ名が付けられます（例： poAddr.city）。
- 型が date の since プロパティには、現在の日付が事前に入力されます。このエントリは、mm/dd/yyyy の所定の日付形式で表示されます。 日付はすべて、Excel テンプレートではこの形式であることが必要です。
- 型が int[] の types プロパティは、カンマ区切りリストとして配列エレメントを含めることができる単一のセルです。 このリストは、プリミティブまたは文字列の配列にのみ使用できます。

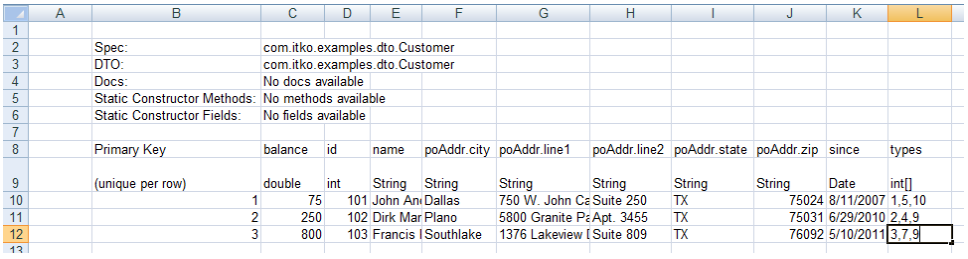
型が Address[] の location プロパティは、このシートに表示されません。location プロパティは、オブジェクトの配列であるため、Excel ファイルの 2 番目のシートに表示されます。このシートには、各行に Address オブジェクトのデータが含まれます。このシートには、「Primary Key」と「reference the containing DTO」という 2 つの特別なフィールドがあり、後者は、このシート内の行をプライマリ シート内の行にリンクするために使用されます。



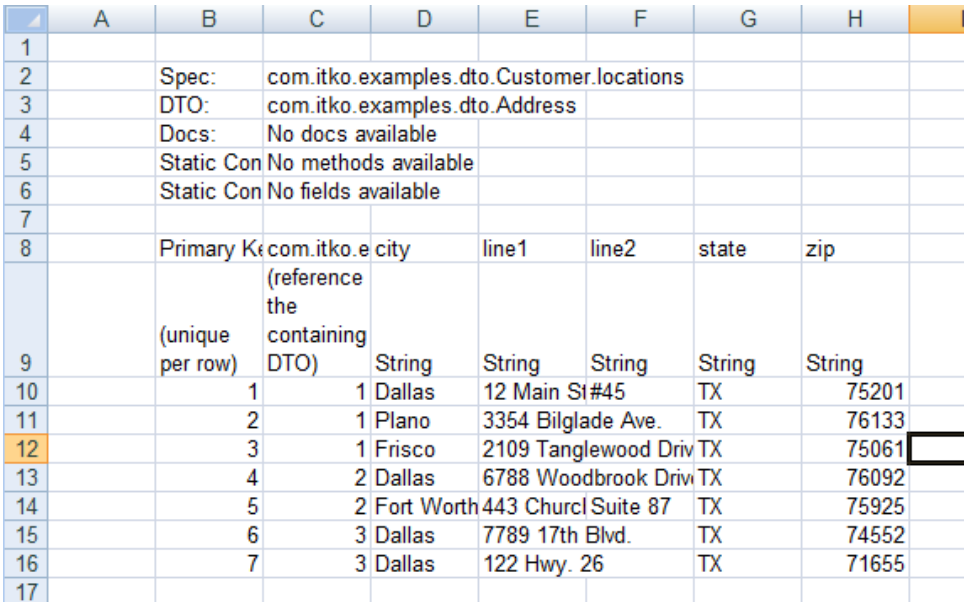
	A	B	C	D	E	F	G	H	I	J
1										
2		Spec:	com.itko.examples.dto.Customer.locations							
3		DTO:	com.itko.examples.dto.Address							
4		Docs:	No docs available							
5		Static Con	No methods available							
6		Static Con	No fields available							
7										
8		Primary Ke	com.itko.e	city	line1	line2	state	zip		
9		(unique per row)	(reference the containing DTO)	String	String	String	String	String		
10										
11										
12										
13										

com.itko.examples.dto.Customer | xamples.dto.Customer.locations

各 Customer オブジェクトに複数の場所があるため、locations シート内の複数の行が、Customer シートの単一の行で指定されたオブジェクトに属します。これは、親 Customer オブジェクトのプライマリ キーを Customer に属する各場所の「reference the containing DTO」フィールドにリストすることにより、2 番目のシートに明示されます。 これは、データベースのプライマリ/外部キーの関係と似ています。

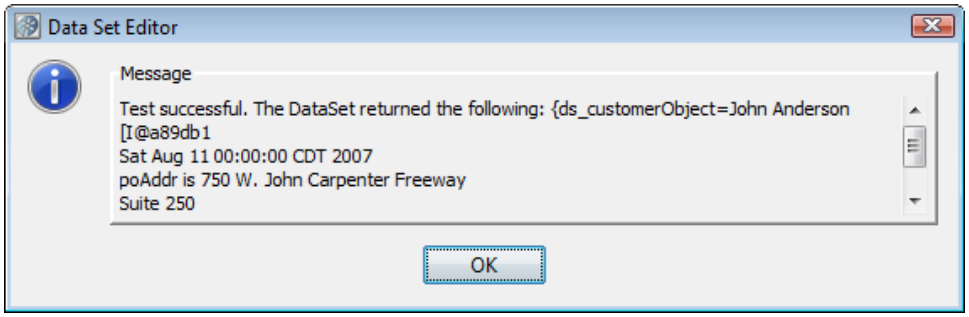


	A	B	C	D	E	F	G	H	I	J	K	L
1												
2		Spec:	com.itko.examples.dto.Customer									
3		DTO:	com.itko.examples.dto.Customer									
4		Docs:	No docs available									
5		Static Constructor Methods:	No methods available									
6		Static Constructor Fields:	No fields available									
7												
8		Primary Key	balance	id	name	poAddr.city	poAddr.line1	poAddr.line2	poAddr.state	poAddr.zip	since	types
9		(unique per row)	double	int	String	String	String	String	String	String	Date	int[]
10		1	75	101	John An	Dallas	750 W. John Ca	Suite 250	TX	75024	8/11/2007	1,5,10
11		2	250	102	Dirk Mar	Plano	5800 Granite P	Apt. 3455	TX	75031	6/29/2010	2,4,9
12		3	800	103	Francis I	Southlake	1376 Lakeview I	Suite 809	TX	76092	5/10/2011	3,7,9



	A	B	C	D	E	F	G	H
1								
2		Spec:	com.itko.examples.dto.Customer.locations					
3		DTO:	com.itko.examples.dto.Address					
4		Docs:	No docs available					
5		Static Con	No methods available					
6		Static Con	No fields available					
7								
8		Primary Ke	com.itko.e	city	line1	line2	state	zip
9		(unique per row)	(reference the containing DTO)	String	String	String	String	String
10		1	1	Dallas	12 Main St	#45	TX	75201
11		2	1	Plano	3354 Bilglade Ave.		TX	76133
12		3	1	Frisco	2109 Tanglewood Driv		TX	75061
13		4	2	Dallas	6788 Woodbrook Driv		TX	76092
14		5	2	Fort Worth	443 Churcl	Suite 87	TX	75925
15		6	3	Dallas	7789 17th Blvd.		TX	74552
16		7	3	Dallas	122 Hwy. 26		TX	71655
17								

スプレッドシートを保存し、DevTest で［テスト アンド キープ］をクリックすると、メッセージに 1 番目の Customer DTO が表示されます。



DTO の複雑さに応じて、Excel ワークブックにさらに複数の Excel シートがあります。 ただし、そのプロセスは上記の例と同じです。

プロパティとしてこの Customer DTO を使用する方法については、「*CA Application Test の使用*」の「テスト ステップ」の Java オブジェクトのテストについてのセクションを参照してください。

一意コード ジェネレータ データ セット

一意コード ジェネレータ データ セットは、DevTest がそれをコールするたびに、一意のトークン（またはコード）を提供します。 トークンは数値または英数字で、それにユーザ定義のプレフィックスを先頭に付けることができます。 一意コード ジェネレータは、通常、新しいユーザ、アカウントなどを作成するためにテストで使用されます。 このジェネレータを使用することで、生成されたトークンまたはコードに、システム内の既存の値が使用されていないことを保証します。

以下のパラメータを入力します。

名前

データ セットの名前を入力します。

ローカル

グローバル データ セットとして機能させるか、またはローカル データ セットとして機能させるかを指定します。デフォルトは［グローバル］です。 ローカル データ セットはシミュレータごとに 1 つ作成されます。グローバル データ セットは 1 回作成され、すべてのシミュレータで共有されます。

ランダム

現在のレコードの次のレコード（シーケンシャル アクセス）が読み取られるか、またはランダムなレコードが読み取られるか。 シーケンシャル読み取りがデフォルトです。

取得する最大レコード

ランダム アクセスの場合に取得するレコード数の上限。 ［ランダム］チェック ボックスがオンになっていない場合、このテキスト フィールドは無効です。

データが終了した場合

データ セットの終了に対応するアクションを選択します。 データ セットの最初からもう一度値を読み取るか、または実行するステップを選択できます。

タイプ

返されるトークンのタイプ。 数値または英数字です。

プレフィックス（オプション）

先頭に付けられるプレフィックス（オプション）。

［テスト アンド キープ］ボタンをクリックして、データをテストおよびロードします。 データ セットが読み取り可能であることを確認するダイアログ ボックスが表示されます。 このダイアログ ボックスには、データの最初のセットが表示されます。

このデータ セットは常にトークンを返します。データ セット エディタの［データが終了した場合］セクションは、このデータ セット タイプの場合、意味を持ちません。

ランダム コード ジェネレータ データ セット

ランダム コード ジェネレータ データ セットは、テスト ケースで使用される数値または英数字データをランダムに生成します。 このデータ セットは一意コード ジェネレータ データ セットに似ていますが、結果長さをユーザが設定することができます。 このデータ セットは、電話番号や社会保障番号など、特定のタイプの一意の値を作成するために使用できます。

以下のパラメータを入力します。

名前

データ セットの名前を入力します。

ローカル

グローバル データ セットとして機能させるか、またはローカル データ セットとして機能させるかを指定します。デフォルトは［グローバル］です。 ローカル データ セットはシミュレータごとに 1 つ作成されます。グローバル データ セットは 1 回作成され、すべてのシミュレータで共有されます。

ランダム

現在のレコードの次のレコード（シーケンシャル アクセス）が読み取られるか、またはランダムなレコードが読み取られるか。 シーケンシャル読み取りがデフォルトです。

取得する最大レコード

ランダム アクセスの場合に取得するレコード数の上限。 ［ランダム］チェック ボックスがオンになっていない場合、このテキスト フィールドは無効です。

データが終了した場合

データ セットの終了に対応するアクションを選択します。 データ セットの最初からもう一度値を読み取るか、または実行するステップを選択できます。

プレフィックス（オプション）

生成されたデータにプレフィックスを追加します。 このフィールドはオプションです。

タイプ

このフィールドでは、英数字または数値タイプのいずれかのデータ セットの生成を許可します。

長さ

このフィールドは、生成されるランダム データの長さをここで設定された値に制限します。

［テスト アンド キープ］をクリックして、データをテストおよびロードします。 データ セットを読み取れたことを示す確認メッセージが表示され、データの最初のセットが表示されます。

メッセージ/相関 ID ジェネレータ データ セット

メッセージ/相関 ID ジェネレータ データ セットは、メッセージング専用の一意コード ジェネレータです。 このデータ セットは、24 バイトの一意のコードを生成します。 このデータ セットは IBM MQ シリーズの相関 ID 専用に設計されていますが、任意の JMS プロバイダに対しても使用できます。このデータ セットは、2 つの特別なプロパティ（**lisa.jms.correlation.id** および **lisa.mq.correlation.id**）を作成または更新します。メッセージング ステップはこれらのプロパティを認識し、メッセージ相関 ID を適切に設定します。

以下のパラメータを入力します。

名前

データ セットの名前を入力します。

ローカル

グローバル データ セットとして機能させるか、またはローカル データ セットとして機能させるかを指定します。デフォルトは［グローバル］です。 ローカル データ セットはシミュレータごとに 1 つ作成されます。グローバル データ セットは 1 回作成され、すべてのシミュレータで共有されます。

ランダム

現在のレコードの次のレコード（シーケンシャル アクセス）が読み取られるか、またはランダムなレコードが読み取られるか。 シーケンシャル読み取りがデフォルトです。

取得する最大レコード

ランダム アクセスの場合に取得するレコード数の上限。 ［ランダム］チェック ボックスがオンになっていない場合、このテキスト フィールドは無効です。

データが終了した場合

データ セットの終了に対応するアクションを選択します。 データ セットの最初からもう一度値を読み取るか、または実行するステップを選択できます。

設定するプロパティ

設定する DevTest プロパティ。デフォルトは **lisa.jms.correlation.id** です。

ファイル名セットのロード データ セット

ファイル名セットのロード データ セットは、ファイル システムから、フィルタされたファイル名のセットに基づいてプロパティに値を割り当てます。

ファイル名のセットには、特定のディレクトリ内のすべてのファイル、または「ファイル パターン」でフィルタされたセットが含まれます。 セットにサブディレクトリを再帰的に含めるオプションもあります。 ファイルは、大文字と小文字を区別し、アルファベット順、上位のディレクトリからサブディレクトリの順に（深さを優先して）返されます。 このデータ セットが使用されるごとに、データ セットの次のファイル名（フル パス名）が返されます。 このファイル名は、データ セットと同じ名前のプロパティに格納されます。

以下のパラメータを入力します。

名前

データ セットの名前を入力します。

ローカル

グローバル データ セットとして機能させるか、またはローカル データ セットとして機能させるかを指定します。デフォルトは［グローバル］です。 ローカル データ セットはシミュレータごとに 1 つ作成されます。グローバル データ セットは 1 回作成され、すべてのシミュレータで共有されます。

ランダム

現在のレコードの次のレコード（シーケンシャル アクセス）が読み取られるか、またはランダムなレコードが読み取られるか。 シーケンシャル読み取りがデフォルトです。

取得する最大レコード

ランダム アクセスの場合に取得するレコード数の上限。 ［ランダム］チェック ボックスがオンになっていない場合、このテキスト フィールドは無効です。

データが終了した場合

データ セットの終了に対応するアクションを選択します。 データ セットの最初からもう一度値を読み取るか、または実行するステップを選択できます。

ディレクトリの位置

スキャンするディレクトリのフル パス名、または、［参照］ボタンを使用して参照できます。

ファイル パターン

フィルタ パターン文字列。必要に応じて、ワイルドカードとしてアスタリスク（*）を使用します。

サブディレクトリからのファイルを含める

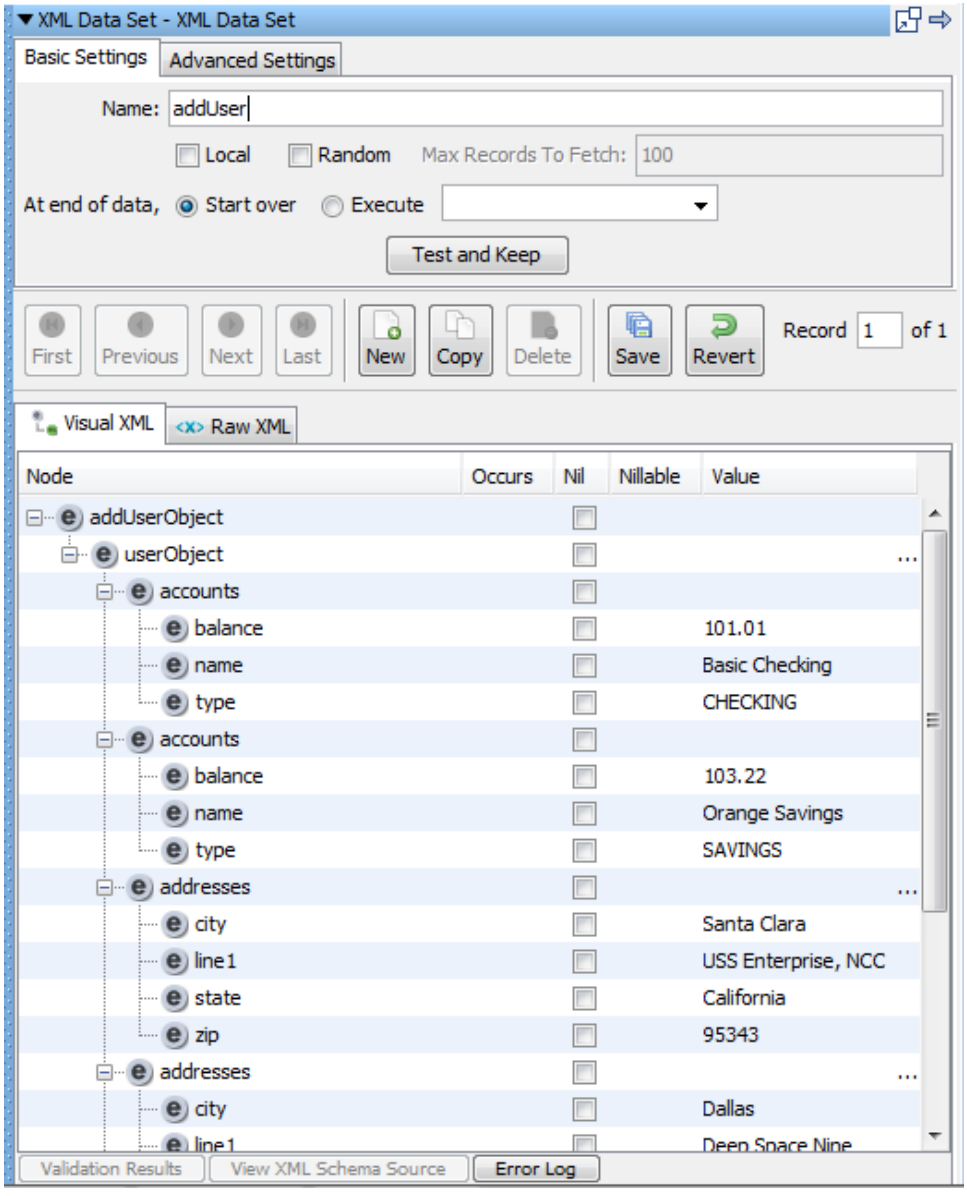
サブディレクトリ内のファイルをリストする場合、このオプションを選択します。

［テスト アンド キープ］をクリックして、データをテストおよびロードします。データ セットを読み取れたことを示す確認ダイアログ ボックスが表示され、データの最初のセットが表示されます。

注: 大量のデータの場合、予測より長くかかることがあります。操作を停止するには、［キャンセル］ボタンを使用します。

XML データ セット

DevTest のその他のデータ セットと同様に、XML データ セットでは、ユーザ指定のコンテンツを個別のレコードに追加できます。ただし、XML データ セットは XML コンテンツの処理に特化しています。



設計時には、［テスト アンド キープ］のクリックにより、XML データ セットの最初のレコードがテスト状態のプロパティの値に入力されます。 プロパティ名はデータ セットの名前と同じです。 たとえば、データ セットが **DataSet1** という名前である場合、テスト ケースに入力されるプロパティは「{{DataSet1}}」です。実行時に、すべてのレコードへのシーケンシャルなアクセスにより、データ駆動型テスト ケースを作成できます。

［基本設定］タブ

name

このデータ セットの名前。 この値は、テスト ステップでプロパティの名前として使用されます。 たとえば、**DataSet1** という名前の XML データ セットは、{{DataSet1}} プロパティに入力されます。

ローカル

グローバル データ セットとして機能させるか、またはローカル データ セットとして機能させるかを指定します。デフォルトは［グローバル］です。 ローカル データ セットはシミュレータごとに 1 つ作成されます。グローバル データ セットは 1 回作成され、すべてのシミュレータで共有されます。

ランダム

現在のレコードの次のレコード（シーケンシャル アクセス）が読み取られるか、またはランダムなレコードが読み取られるか。 シーケンシャル読み取りがデフォルトです。

取得する最大レコード

ランダム アクセスの場合に取得するレコード数の上限。 ［ランダム］チェック ボックスがオンになっていない場合、このテキスト フィールドは無効です。

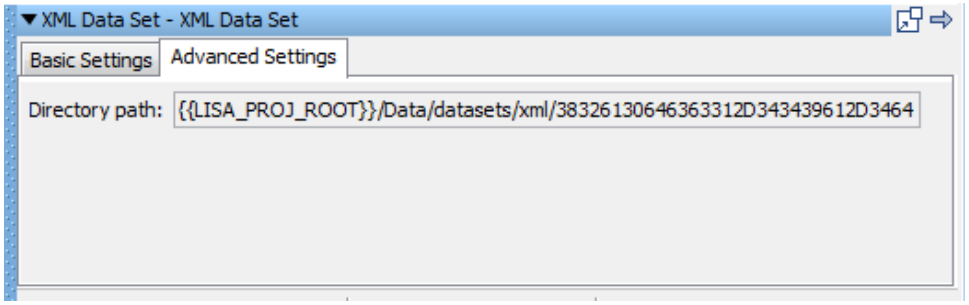
データが終了した場合

データ セットの終了に対応するアクションを選択します。 データ セットの最初からもう一度値を読み取るか、または実行するステップを選択できます。

テスト アンド キープ

設計時に、テスト状態でデータ セットと関連付けられるプロパティに、最初のレコードの値を入力します。

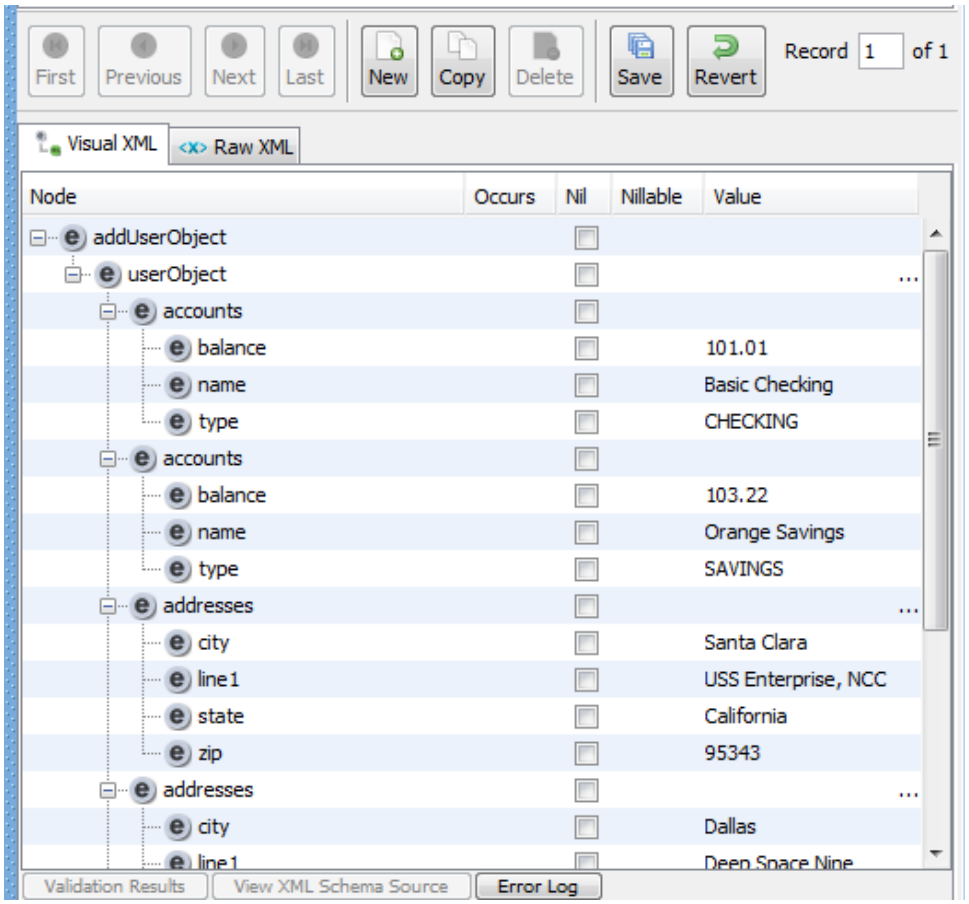
［詳細設定］タブ



ディレクトリ パス

この読み取り専用値は、レコードが保存されるファイル システムのディレクトリを表します。レコードとディレクトリ パス内のファイルとのマッピングは、1 対 1 のマッピングです。 XML データ セットがプロジェクト内で作成されている場合、ディレクトリ パスの先頭は、「{{LISA_PROJ_ROOT}}/Data/datasets/xml/」または「{{LISA_RELATIVE_PROJ_ROOT}}/Data/datasets/xml/」になります。 プロジェクトが使用されない場合、ディレクトリ パスの先頭は「{{LISA_HOME}}/datasets/xml/」になります。

レコード編集パネル



アクション ボタン

最初

XML データ セットの最初のレコードに移動します。

前へ

前のレコードに移動します。

次へ

次のレコードに移動します。

最後

最後のレコードに移動します。

新規

レコードを作成します。

コピー

データ セットの最後に現在のレコードのコピーを作成し、そのレコードに移動します。

削除

現材のレコードを削除します。

保存

保留中のすべての変更を保存します。

元に戻す

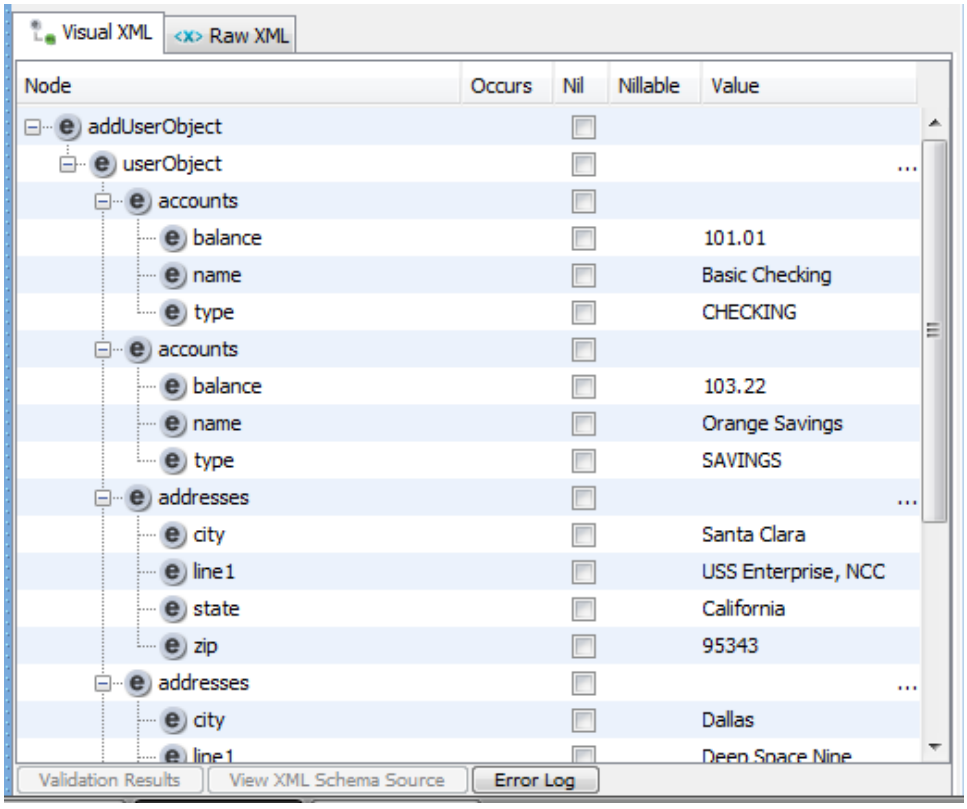
保留中のすべての変更を元に戻します。

レコード番号セレクタ

レコード X / X

特定のレコードにジャンプするには、レコード番号を入力します。

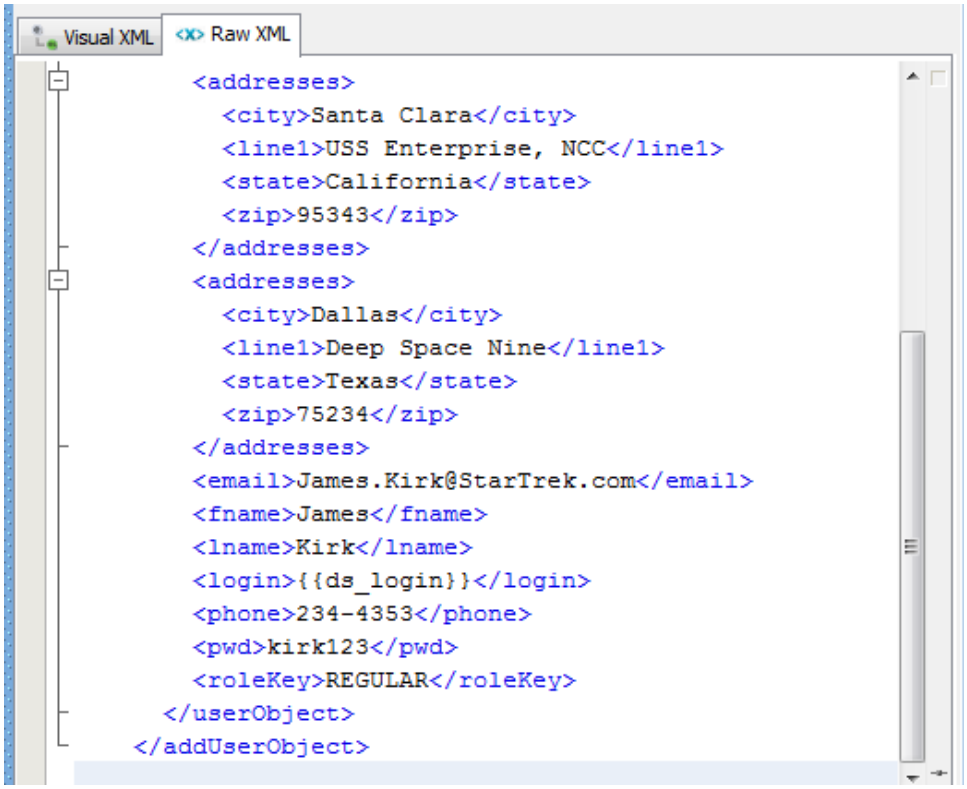
[ビジュアル XML]タブ



ビジュアル XML エディタ

［ノード］および［値］列にマウス カーソルを置くと、ツールヒントが表示されます。テーブル内の値が長すぎて完全に表示できない場合、ツールヒントが役立ちます。

［RAW XML］タブ



このタブには、レコードに含まれる XML の RAW テキスト ビューが含まれます。

左の境界をダブルクリックすると、エディタの上部のツールバー、行番号バー、および下部の編集情報バーの表示/非表示が切り替わります。

フィルタの説明

このセクションでは、使用可能な各フィルタについて説明します。

正規表現は、複数のフィルタで比較を行うために使用されます。 正規表現の詳細については、「正規表現」を参照してください。

このセクションには、以下のフィルタの説明が含まれています。

ユーティリティ フィルタ

以下のフィルタは、任意のテスト ステップのユーティリティ フィルタのリストで使用できます。

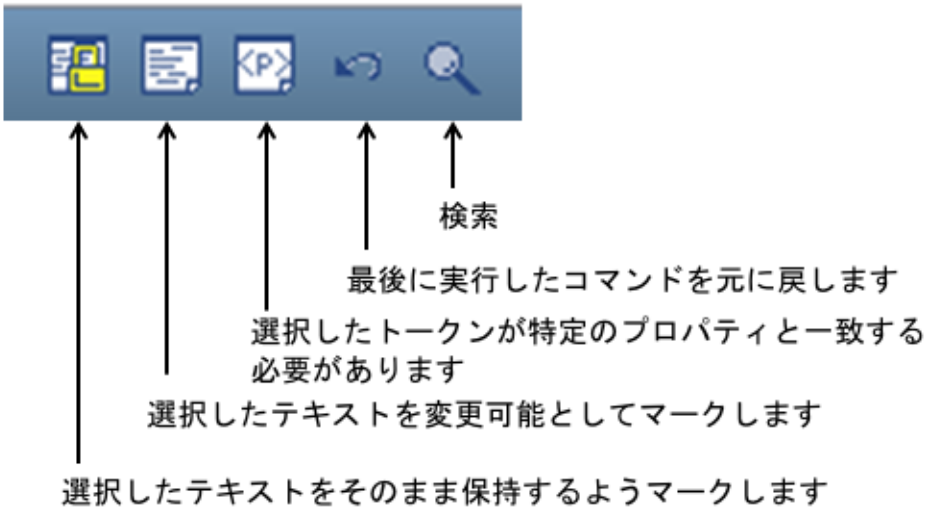
関連値に基づくプロパティの作成

関連値に基づくプロパティの作成フィルタでは、テキスト コンテンツを読み取ってフィルタを実行し、DevTest のプロパティに格納する情報を絞り込むことができます。 このフィルタはテキストに対して、またテキストとして扱われる XML や HTML に対して使用できます。 このフィルタでは、「画面のペイント」技術を使用します。

「画面のペイント」は、HTML のどの部分をプロパティとして解析するかを定義する優れた柔軟性をもたらします。 以下のいずれかの方法でテキストをマークします。

- 応答にそのまま表示される必要があるテキスト: 保持テキスト ブロック。
- 応答にそのまま表示される必要のないテキスト: 変更可能テキスト ブロック。
- プロパティに格納されているテキスト： プロパティ一致ブロック。

テキストは、エディタの下部のアイコンを使用してマークされます。

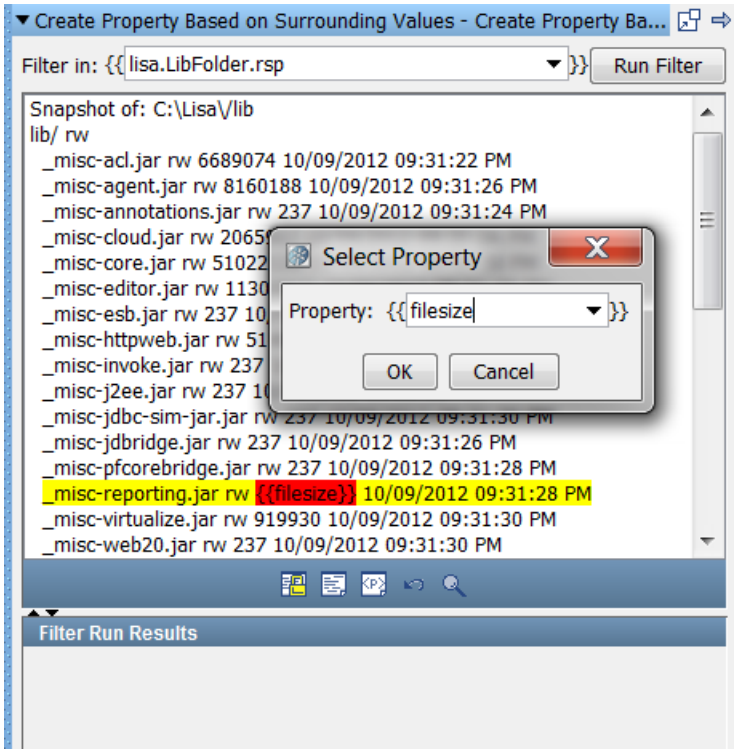


以下の例の目的は、特定のファイルのサイズをプロパティに格納することです。

テキストはエディタ アイコンを使用してマークします。テキストを選択して、適切なアイコンをクリックします。

- 黄色の背景色は、そのまま表示される必要があるテキストを表しています。 これは保持テキスト ブロックであり、保持テキスト アイコン を使用してマークされます。
- 赤の背景色は、ダイアログ ボックスに入力されたプロパティに格納されているテキストを表しています。 これはプロパティ一致ブロックであり、プロパティ一致アイコン を使用してマークされます。

注: プロパティ一致ブロックは保持テキスト ブロックによって常に囲まれている必要があります。



この画面は、テキスト バッファの内容を表しています。 この目的は、_misc-reporting.jar ファイルのサイズを解析することです。 ファイル サイズは、**_misc-reporting.jar** の後に表示されている数値です。

ファイル サイズの周囲に境界が設定されており、保持テキスト がクリックされています。 選択された内容の内側の実際のファイル サイズ テキストが選択され、プロパティ一致 がクリックされています。

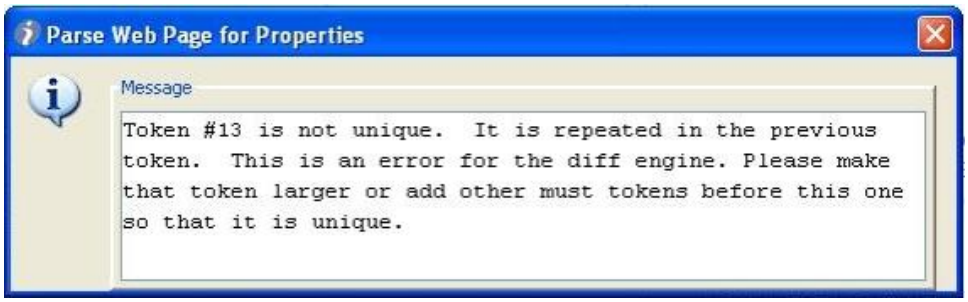
その後、プロパティ名がダイアログ ボックスに入力されています。 ファイル サイズの実際の値がプロパティの名前で置き換えられています。

このフィルタが実行されると、**filesize** プロパティに _misc-reporting.jar ファイルのサイズが割り当てられます。

このテキスト バッファに対してこのプロセスを繰り返して、プロパティを必要な数だけ定義することができます。

一意ではないトークンの処理

以下のエラー メッセージが表示される場合、選択したトークンは一意ではありません。選択内容が前のトークンと重複しています。



ほとんどの場合、この問題を解決するには、別のトークンを作成して前のトークンも保持テキスト トークンにします。 この方法が有効でない場合は、2 つの重複したトークンのもう一方の保持テキスト ブロックを配置し直すことで、エラーを回避します。

DevTest は 2 つの重複したトークンを相対的な場所に基づいて識別できるため、このソリューションは有効です。

ステップ応答の格納

ステップ応答の格納フィルタでは、最終応答を将来使用するためにプロパティとして保存できます。

以下のパラメータを入力します。

フィルタ

フィルタを適用する場所。 このフィルタに対するこの値を変更することはできません。

プロパティ

最終応答を格納するプロパティの名前。

フィルタ実行結果

フィルタの実行結果として設定されるプロパティおよび値を表示します。

フィルタを実行

フィルタを実行するには、［フィルタを実行］ボタンをクリックします。 結果は［フィルタ実行結果］セクションに表示されます。

最終応答（Last Response）プロパティの上書

"最終応答（Last Response）" プロパティの上書フィルタでは、最終応答の現在の値を既存のプロパティの値に置き換えることができます。 たとえば、EJB を実行するものの、EJB に対してメソッド コールを何度か行った後に最終的に値を受け取ると想定します。（EJB オブジェクトの代わりに）メソッド コールの結果を最終応答として使用すると、テスト ケースが改善される場合があります。 フィルタを使用してそのコールからの戻り値をプロパティとして保存し、その後このフィルタでそのプロパティを使用することができます。

以下のパラメータを入力します。

フィルタ

ステップの最終応答と見なすプロパティの名前。プロパティがプルダウン メニューにない場合は入力できます。プロパティは存在している必要があります。

XML に変換

応答を有効な XML に変換する場合は、このオプションを選択します。

フィルタ実行結果

フィルタの実行結果として設定されるプロパティおよび値を表示します。

フィルタを実行

フィルタを実行するには、［フィルタを実行］ボタンをクリックします。 結果は［フィルタ実行結果］セクションに表示されます。

ファイルへのプロパティ値の保存

ファイルへのプロパティ値の保存フィルタでは、既存のプロパティの値をファイル システム内のファイルに保存できます。

以下のパラメータを入力します。

フィルタ

ファイルに値を書き込むプロパティの名前。

場所

値を書き込むファイルのパス名。ファイルを参照できます。場所にプロパティを使用することもできます。

追加モード

既存のファイルに情報を追加する場合は、このチェック ボックスをオンにします。

フィルタ実行結果

フィルタの実行結果として設定されるプロパティおよび値を表示します。

フィルタを実行

フィルタを実行するには、［フィルタを実行］ボタンをクリックします。 結果は［フィルタ実行結果］セクションに表示されます。

引数文字列としてのプロパティ値の解析

引数文字列としてのプロパティ値の解析フィルタでは、特定の属性のテキストをプロパティに格納できます。 このフィルタは、フィルタされた値を情報として解析する 2 つ目のフィルタとして使用すると便利です。

以下のパラメータを入力します。

フィルタ

解析する既存のプロパティの名前。 たとえば、「lisa.deleteUser.cookies.rsp」プロパティを解析して SESSIONID 属性の値を返すには、「lisa.deleteUser.cookies.rsp」と入力します。

IsURL

プロパティ値が URL である場合は、このチェック ボックスをオンにします。

属性

取得する属性。 この例では、JSESSIONID 属性です。

プロパティ

属性のテキストを格納するプロパティの名前。 この例では、sessionID です。

デフォルト（見つからない場合）

属性が見つからない場合に使用するデフォルト値です。

フィルタ実行結果

フィルタの実行結果として設定されるプロパティおよび値を表示します。

フィルタを実行

フィルタを実行するには、［フィルタを実行］ボタンをクリックします。 結果は［フィルタ実行結果］セクションに表示されます。

キーから別のキーへのプロパティ値の保存

キーから別のキーへのプロパティ値の保存フィルタでは、通常の Java ルールが適用される参照によって、あるキーから別のキーに値をコピーします。

このフィルタは、単なるプロパティのコピーによる BeanShell のオーバーヘッドをシンプルに最適化します。

以下のパラメータを入力します。

フィルタ

このフィールドは、ほかのプロパティにコピーされるプロパティの内容を受け入れます。 このフィールドは入力ソース プロパティです。

コピー先プロパティ

このフィールドは、入力プロパティの内容がコピーされるプロパティ名です。

フィルタを実行

テスト ケースが完了するまで待機することなく、テスト ステップの開発中にただちにフィルタをテストできます。

前処理

ステップの前にフィルタを実行します。

後処理

ステップの後にフィルタを実行します。

タイム スタンプ フィルタ

タイム スタンプ フィルタは、プロパティに現在の時刻および日付を割り当てて、後のステップでこのプロパティを使用できるようにするために使用します。

以下のパラメータを入力します。

フィルタ

既存のステップの名前。

日付パターン

表示する日付パターンを選択します。

オフセット

現在の日付を基準にして、適切な（将来または過去の）日付にオフセットするために使用します。

前処理

選択すると、ステップを実行する前にタイム スタンプを生成します。

前処理用プロパティ

前処理のタイム スタンプを格納するプロパティ。

後処理

選択すると、ステップを実行した後にタイム スタンプを生成します。

後処理用プロパティ

後処理のタイム スタンプを格納するプロパティ。

フィルタ実行結果

フィルタの実行結果として設定されるプロパティおよび値を表示します。

フィルタを実行

フィルタを実行するには、［フィルタを実行］ボタンをクリックします。 結果は［フィルタ実行結果］セクションに表示されます。

データベース フィルタ

以下のフィルタは、任意のテスト ステップのデータベース フィルタのリストで使用できます。

JDBC 結果セットから値を抽出

JDBC 結果セットから値を抽出フィルタでは、特定の JDBC 結果セット値のテキストをプロパティに格納できます。 このフィルタは、フィルタ リストから手動フィルタとして作成する方法か、結果セット応答に対して組み込みのフィルタ コマンドを使用する方法の 2 つの方法で作成できます。

フィルタを手動で作成する方法

以下のパラメータを入力します。

フィルタ

ステップの最終応答と見なすプロパティの名前。プロパティがプルダウン メニューにない場合は入力できます。プロパティは存在している必要があります。

列(1 以上の整数または名前)

列（フィールド）のインデックスまたは名前。

行(0 以上の整数)

値を取得する行。 このフィールドはゼロ ベースのインデックスです。

プロパティ

行と列が交差する箇所のセルの値が格納されるプロパティの名前。

フィルタ実行結果

フィルタの実行結果として設定されるプロパティおよび値を表示します。

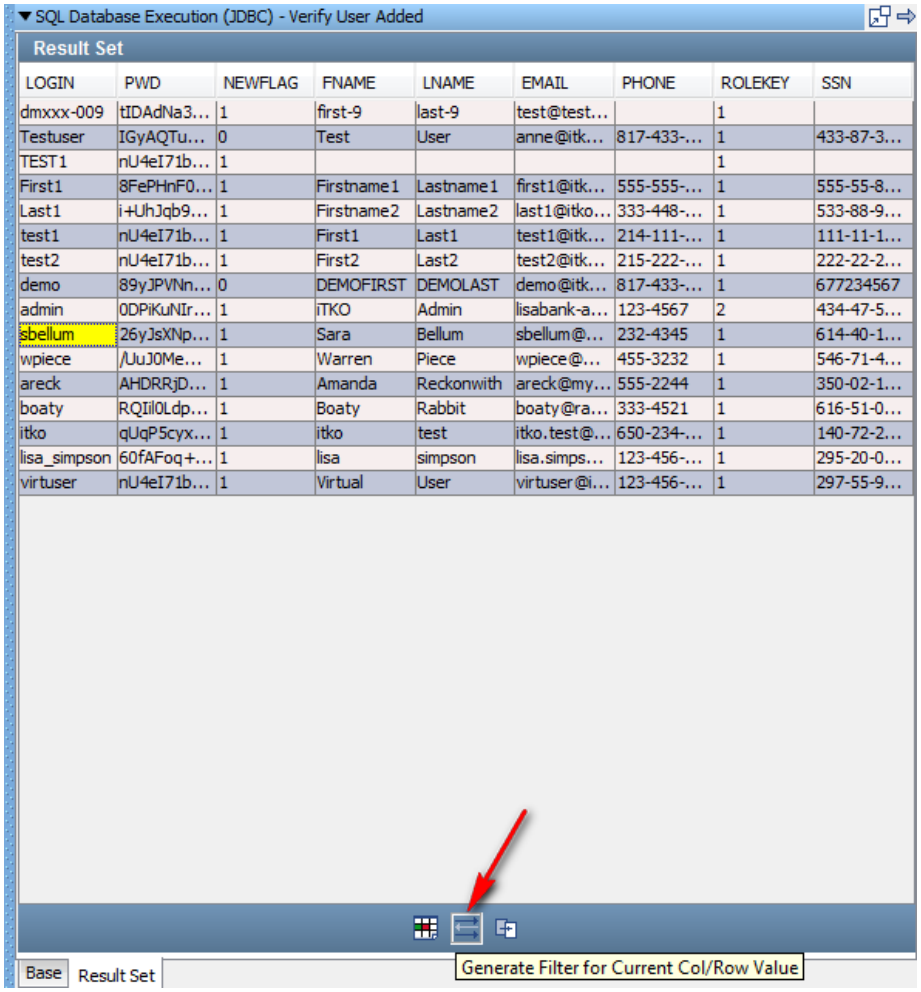
フィルタを実行

フィルタを実行するには、［フィルタを実行］ボタンをクリックします。 結果は［フィルタ実行結果］セクションに表示されます。

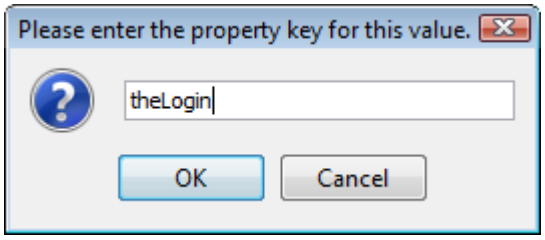
結果セット応答からフィルタを作成する方法

次の手順に従ってください:

1. 結果セットが含まれるステップ応答を表示します。
2. 結果セットから、フィルタに格納するセルの値を選択します。



3. 上記の例で矢印で示されている  ［現在の列/行の値用にフィルタを生成］をクリックします。
4. ダイアログ ボックスにプロパティ キーを入力します。



5. ［OK］をクリックします。

theLogin プロパティに値 **sbellum** を格納するフィルタが作成されます。

シンプル結果セット

シンプル結果セット フィルタは、結果セット応答内の行の数をカウントするために使用します。

詳細については、「*CA Application Test の使用*」の「JDBC 結果セットのサイズ (P. 155)」を参照してください。

JDBC 結果セットのサイズ

JDBC 結果セットのサイズ フィルタでは、各 JDBC ベースのステップで返された結果セットが、指定した条件に一致するかどうかを確認できます。このフィルタは、最も一般的なデータベース エラーを自動的に処理する簡単なフィルタです。

このフィルタは JDBC ステップ以外のステップには影響せず、テスト ケースでグローバル フィルタとして頻繁に使用されます。

以下のパラメータを入力します。

結果セットが警告を含む

一部のデータベースからは、結果セットに警告が返されます。使用するデータベースでこの機能がサポートされており、このフィルタの エラー時ステップに対して警告を発したい場合には、［結果セットが警告を含む］チェック ボックスをオンにします。

行数 >=

結果セット内の最小行数。結果セットに含まれる行の数がこの値より少ない場合、フィルタは次のステップにエラー時ステップで指定された値を設定します。

行数 <_=

結果セット内の最大行数。結果セットに含まれる行の数がこの値より多い場合は、フィルタは次のステップにエラー時ステップで指定された値を設定します。

エラー時

このフィルタに対する条件が満たされない場合に実行するステップ。

注: エラーの有無に基づいて次のステップを選択できるため、このフィルタは一般的なグローバル アサーションの目的に適っています。

結果セットのサイズをプロパティに設定

結果セットのサイズをプロパティに設定フィルタでは、結果セットの数を指定したプロパティに格納できます。

以下のパラメータを入力します。

フィルタ

ステップの最終応答と見なすプロパティの名前。プロパティがプルダウン メニューにない場合は入力できます。プロパティは存在している必要があります。

行数を格納するプロパティ

行数を格納するためのユーザが指定するプロパティ名。デフォルトのプロパティ名は PROP です。

フィルタ実行結果

フィルタの実行結果として設定されるプロパティおよび値を表示します。

フィルタを実行

フィルタを実行するには、［フィルタを実行］ボタンをクリックします。 結果は［フィルタ実行結果］セクションに表示されます。

結果セット行の別の値に対する値の取得

結果セット行の別の値に対する値の取得フィルタでは、特定の値を持つ結果セット内の列（フィールド）を検索できます。値が見つかった場合は、同じ行の別の列（フィールド）の値がプロパティに設定されます。

このフィルタは、フィルタ リストから手動フィルタとして作成する方法か、結果セット応答に対して組み込みのフィルタ コマンドを使用する方法で作成できます。

フィルタを手動で作成する方法

以下のパラメータを入力します。

フィルタ

ステップの最終応答と見なすプロパティの名前。プロパティがプルダウン メニューにない場合は入力できます。プロパティは存在している必要があります。

検索テキスト（正規表現）

検索文字列。

検索列（1 以上の整数または名前）

検索する列のインデックスまたは名前。

値列（1 以上の整数または名前）

プロパティ値を抽出する列のインデックスまたは名前。

プロパティ

値を格納するプロパティの名前。

フィルタ実行結果

フィルタの実行結果として設定されるプロパティおよび値を表示します。

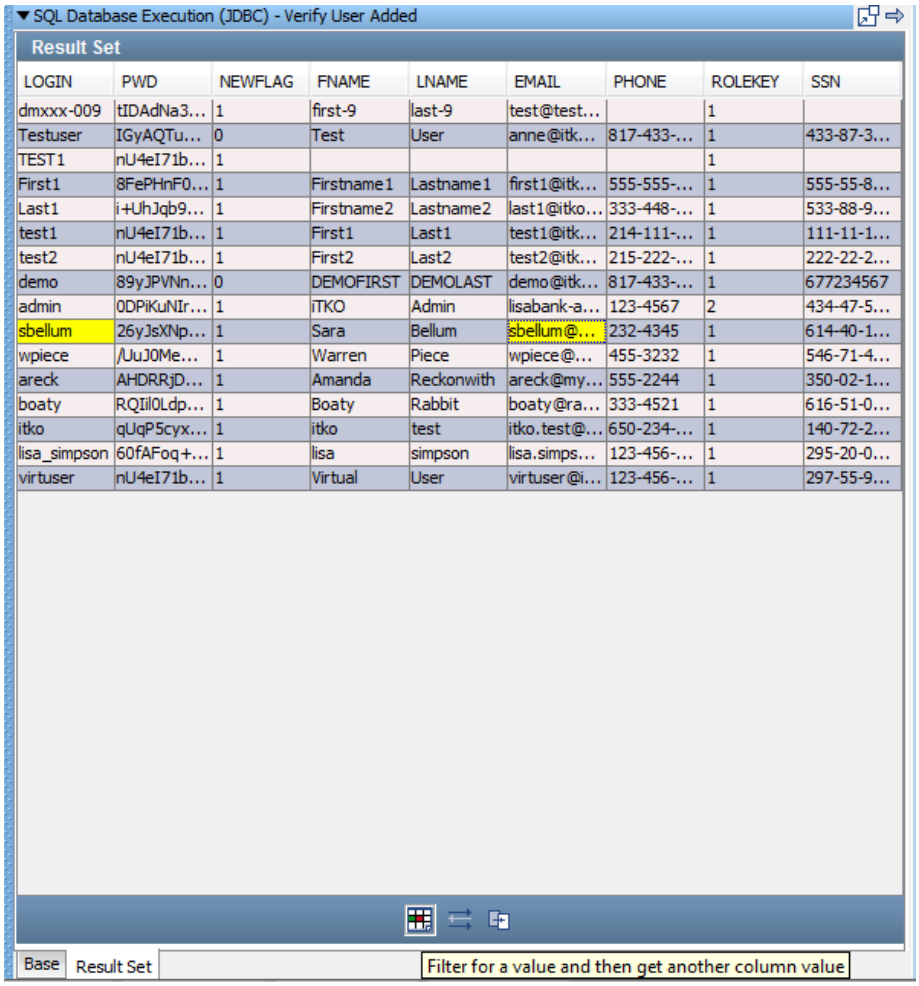
フィルタを実行

フィルタを実行するには、［フィルタを実行］ボタンをクリックします。結果は［フィルタ実行結果］セクションに表示されます。

結果セット応答からフィルタを作成する方法

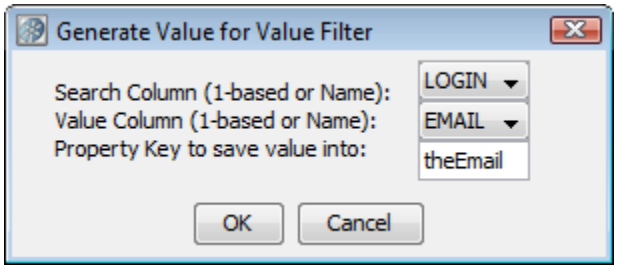
1. 結果セットが含まれるステップ応答を表示します。

2. 結果セットから、Ctrl キーを使用して、別々の列にある 2 つの値を選択します。



3. ［フィルタ］ アイコンを使用して、指定の列を、指定の値で検索し、別の列の値を取得しますフィルタを選択します。

4. 表示されるダイアログ ボックスで、検索と値の列を選択または再割り当てし、プロパティ キーを入力します。



5. ［OK］をクリックします。

作成されるフィルタは、前の例で手動で作成したものと同じです。

この例では、**sbellum** を検索し、見つかった場合はその行の EMAIL 列の値が **theEmail** プロパティに設定されます。

Messaging/ESB フィルタ

以下のフィルタは、任意のテスト ステップのメッセージング/ESB フィルタのリストで使用できます。

メッセージからのペイロードとプロパティの抽出 (P. 160)
MQ メッセージを VSE 要求に変換 (P. 161)
JMS メッセージを VSE 要求に変換 (P. 162)

メッセージからのペイロードとプロパティの抽出

DevTest では、メッセージからのペイロードとプロパティの抽出フィルタを使用して、メッセージのいくつかの内部プロパティをテスト ステップ内のプロパティに自動的に抽出します。 また、ペイロードをプロパティに自動的に抽出することも選択できます。 これはメッセージからデータを迅速に取得する方法です。

さまざまなメッセージング プラットフォームによってさまざまな制限があり、実行時には警告と見なされる場合があります。

プロパティ名はデフォルトで **lisa.stepName.message** です。また、プレフィックスを指定することもできます。 ペイロードの正確な名前を指定できます。

以下のパラメータを入力します。

フィルタ

ステップの最終応答と見なすプロパティの名前。プロパティがプルダウン メニューにない場合は入力できます。プロパティは存在している必要があります。

ペイロードの取得

ペイロードが必要な場合は、このオプションを選択します。

ペイロードを保存するプロパティ キー

ペイロードとして使用するプロパティ キーを入力または選択します。

抽出された詳細のプレフィックス

結果内のプロパティ名に付けられるプレフィックスを入力します。

メッセージ ID の取得

メッセージ ID を取得する場合に選択します。

適切な場合に相関 ID を取得

相関 ID を取得する場合に選択します。

追加拡張プロパティ

追加拡張プロパティをすべて取得する場合に選択します。

フィルタを実行

フィルタを実行するには、［フィルタを実行］ボタンをクリックします。 結果は［フィルタ実行結果］セクションに表示されます。

MQ メッセージを VSE 要求に変換

MQ メッセージを VSE 要求に変換フィルタは、VSE レコーダから自動的に追加されます。 このフィルタは、レコーディングを適切に機能させるという目的に役立ちますが、慎重に使用する必要があります。VSE モデル内のステップに追加した場合、通常は削除および編集しないようにします。

以下のパラメータを入力します。

フィルタ

ステップの最終応答と見なすプロパティの名前。プロパティがプルダウン メニューにない場合は入力できます。プロパティは存在している必要があります。

オブジェクト フォーム

オブジェクト フォームを取得する場合に選択します。

相関 ID の追跡

相関 ID を追跡する場合に選択します。

メッセージ ID の追跡

メッセージ ID を追跡する場合に選択します。

トランザクション トラッキング タイプ

［シーケンシャル］、［相関 ID］、［メッセージ ID］、または［メッセージ ID と相関 ID］から、適切なトラッキング タイプを選択します。

フィルタを実行

フィルタを実行するには、［フィルタを実行］ボタンをクリックします。 結果は［フィルタ実行結果］セクションに表示されます。

JMS メッセージを VSE 要求に変換

JMS メッセージを VSE 要求に変換フィルタは、LISA VSE レコーダから自動的に追加されます。 このフィルタは、レコーディングを適切に機能させるという目的に役立ちますが、慎重に使用する必要があります。VSE モデル内のステップに追加した場合、通常は削除および編集しないようにします。

以下のパラメータを入力します。

フィルタ

ステップの最終応答と見なすプロパティの名前。プロパティがプルダウン メニューにない場合は入力できます。プロパティは存在している必要があります。

オブジェクト フォーム

オブジェクト フォームを取得する場合に選択します。

相関 ID の追跡

相関 ID を追跡する場合に選択します。

メッセージ ID の追跡

メッセージ ID を追跡する場合に選択します。

トランザクション トラッキング タイプ

［シーケンシャル］、［相関 ID］、［メッセージ ID］、または［メッセージ ID と相関 ID］から、適切なトラッキング タイプを選択します。

フィルタを実行

フィルタを実行するには、［フィルタを実行］ボタンをクリックします。 結果は［フィルタ実行結果］セクションに表示されます。

HTTP/HTML フィルタ

以下のフィルタは、任意のテスト ステップの HTTP/HTML フィルタのリストで使用できます。

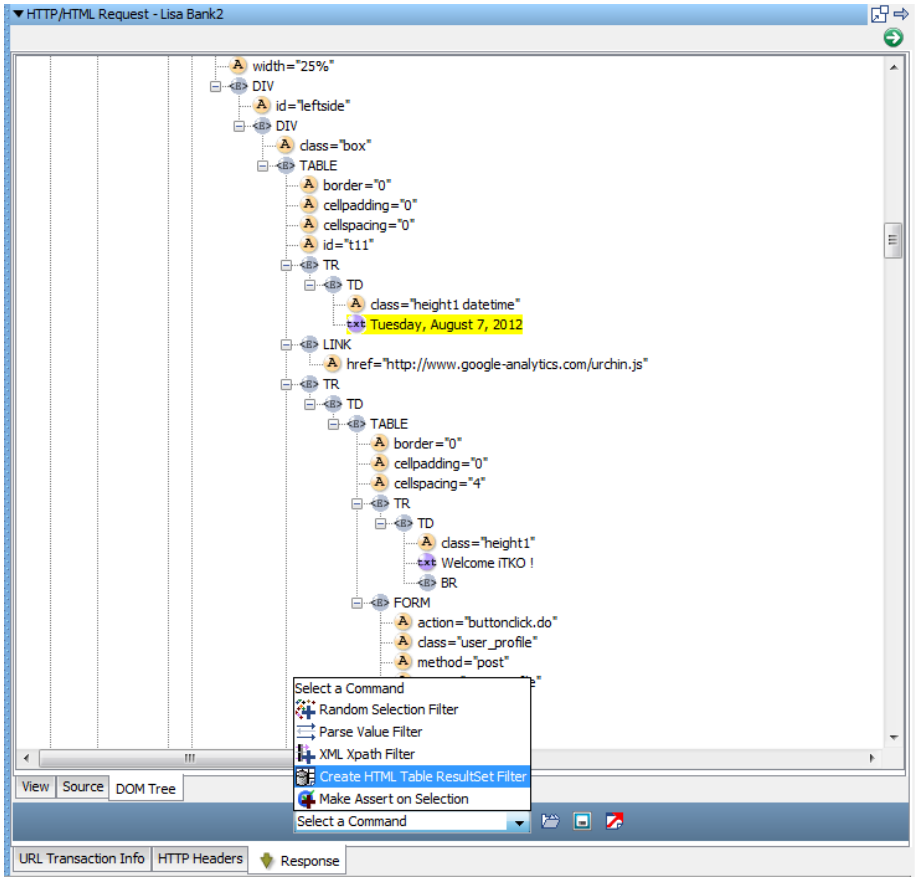
HTML テーブル行からの結果セットの作成

HTML テーブル行からの結果セットの作成フィルタでは、HTML 応答によって返される HTML テーブルから結果セット（JDBC 結果セットなど）を作成できます。 HTML テーブルの列および行は選択でき、これらから結果セットが作成されます。 その後、結果セットはデータベース ステップの場合と同じ方法でアサーションを生成するために使用できます。

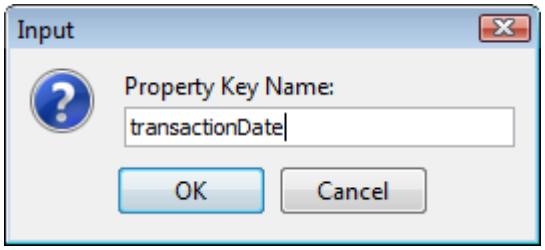
このフィルタは、フィルタ リストから選択してパラメータを指定することで作成できますが、ステップで使用可能なフィルタ コマンドのいずれかを使用して HTTP/HTML 要求ステップの応答から直接作成する方がはるかに簡単です。 ここではこの方法を使用します。 ここで作成されたパラメータ、つまりこのフィルタを手動で作成する場合に計算が必要となるパラメータについては、このセクションの後半で説明します。

テーブルでフィルタを作成する方法

1. テーブルが含まれる Web ページを記録します。
2. 適切な HTML ステップに移動し、DOM ツリーからそれを確認します。
3. テーブルに設定する値を選択します。Ctrl キーを使用して複数のフィールドを選択します。 結果セットで使用するテーブル内の各列から例とする値を 1 つ選択します。

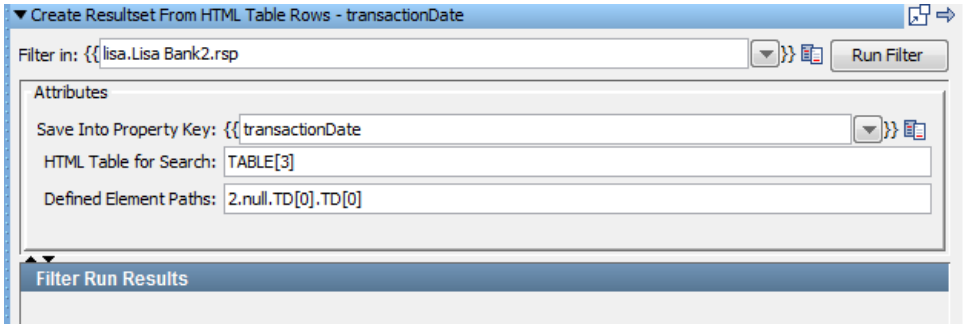


4. その値が強調表示されたら、［HTML テーブル結果セット フィルタの作成］を選択します。
5. ウィンドウにプロパティ名を入力します。

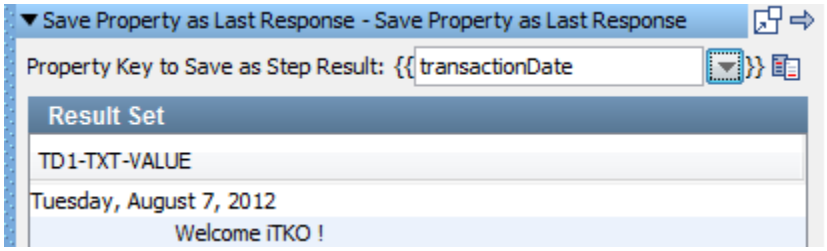


プロパティがテスト ケースで使用できるようになります。

プロパティが現在のステップに追加されます。 以下の図は、このステップのために計算されたパラメータを示しています。 これらは、このフィルタを作成するために手動で指定しなければならなかったパラメータです。



このフィルタの結果を表示するために、［プロパティを最終応答として保存］タイプのステップと、フィルタによって作成されたプロパティを追加しました。 結果セット パネルに結果が表示されます。



既存のテスト ケースを編集するときには、［指定した時点までテスト ケースを再生］コマンドを使用して、フィルタからプロパティを生成するテスト ケースを再生する必要がある場合があります。 ［指定した時点までテスト ケースを再生］コマンドは、ツールバーにある［再生］  アイコンを使用してアクティブ化できます。また、このステップの結果セット ウィンドウの下部にある組み込みのフィルタおよびアサーションを使用することもできます。

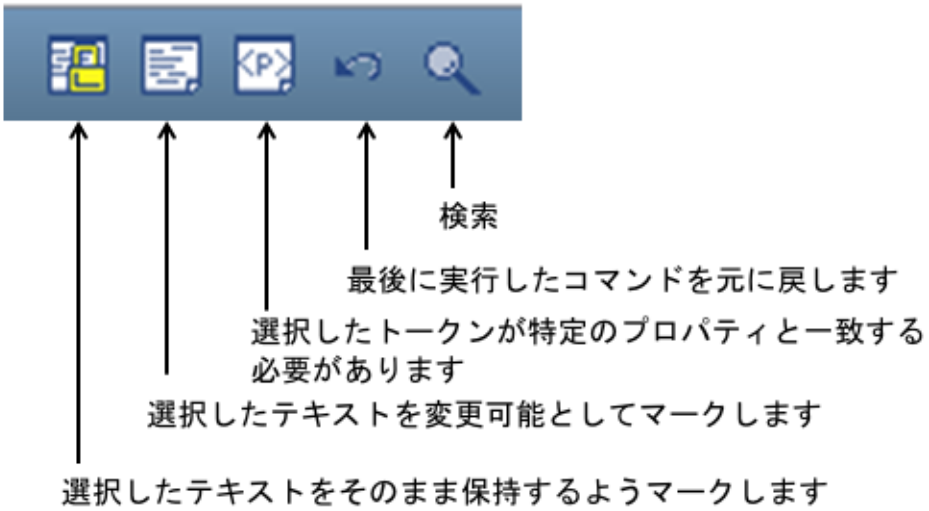
Web ページのプロパティの解析

Web ページのプロパティの解析フィルタでは、表示された Web ページを確認して、HTML コンテンツからプロパティを作成することができます。このフィルタでは、「画面のペイント」技術を使用します。

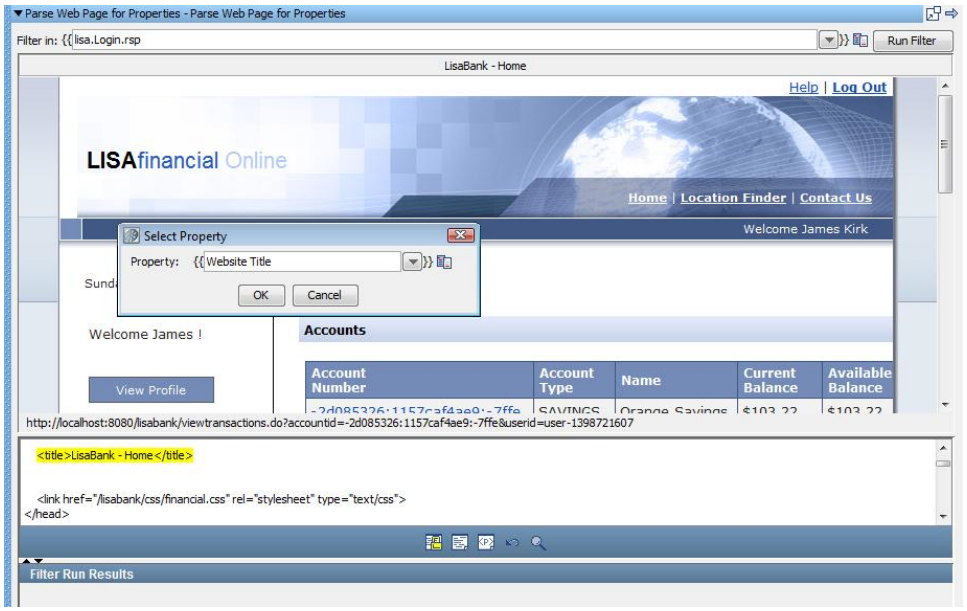
「画面のペイント」は、HTML のどの部分をプロパティとして解析するかを定義する優れた柔軟性をもたらします。 以下のいずれかの方法でテキストをマークします。

- 応答にそのまま表示される必要があるテキスト: 保持テキスト ブロック。
- 応答にそのまま表示される必要のないテキスト: 変更可能テキスト ブロック。
- プロパティに格納されているテキスト： プロパティ一致ブロック。

テキストは、エディタの下部のアイコンを使用してマークされます。



以下の例では、Web サイトのタイトル「LisaBank - Home」がユーザによって変わるため、プロパティとして格納する必要があると想定しています。



このウィンドウでは、上部パネルのブラウザに HTML が表示され、下部パネルに実際の HTML テキストが表示されています。

Web サイト上のフィールドからプロパティを作成方法

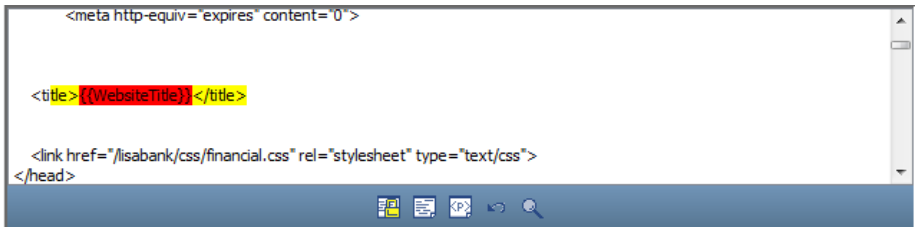
1. 下部パネルの HTML テキストで示されているフィールドに移動します。
2. テキストを選択し、保持テキスト アイコン をクリックします。

   この例では、選択されているフィールドは Web サイトのタイトルです。黄色い強調表示は、そのまま表示される必要があるテキストを示します。
3. 強調表示されたコンテンツの内部の名前のテキストを選択し、プロパティ一致アイコン をクリックします。

   この例では、Web サイト名のテキスト「LisaBank - Home」が選択されています。
4. プロパティ一致アイコン をクリックします。

   ［プロパティの選択］ダイアログ ボックスが表示されます。
5. ダイアログ ボックスにプロパティ名を入力します。

   下部パネルの HTML テキストで、Web サイト名のテキストがプロパティの名前に置き換わります。赤の背景色は、ダイアログ ボックスに入力されたプロパティに格納されているテキストを表しています。



注: すべてのプロパティ一致ブロックは保持テキスト ブロックによって常に囲まれている必要があります。

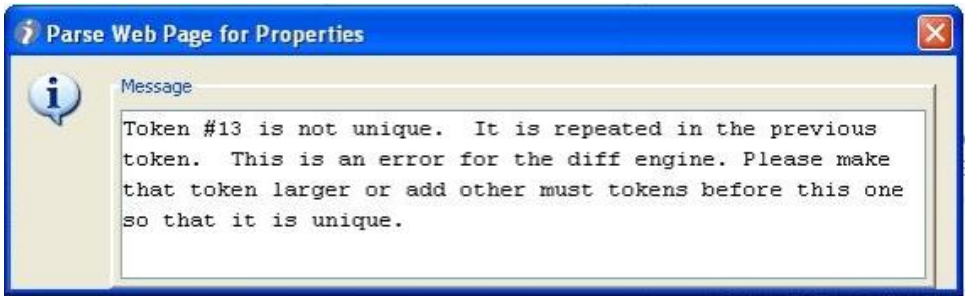
Web ブラウザ内のコンテンツを選択することにより、Web ページ ビューから単純にこのアクションを繰り返し実行できます。 Web ブラウザで選択したい領域をクリックし、次に HTML パネルで選択する方が簡単な場合もあります。

このフィルタが実行されると、**Website Title** プロパティに HTML ページに表示されている現在の値が割り当てられます。 Web サイトのタイトルは、テキスト バッファでその場所を変更することができますが、引き続きプロパティ用に検出および解析できます。

さらにプロパティを定義するには、このテキスト バッファでこのプロセスを繰り返します。

一意ではないトークンの処理

以下のエラー メッセージが表示される場合、選択したトークンは一意ではありません。選択内容が前のトークンと重複しています。



ほとんどの場合、この問題を解決するには、別のトークンを作成して前のトークンも保持テキスト トークンにします。 この方法が有効でない場合は、2 つの重複したトークンのもう一方の保持テキスト ブロックを配置し直すことで、エラーを回避します。

DevTest は 2 つの重複したトークンを相対的な場所に基づいて識別できるため、このソリューションは有効です。

HTML/XML 結果の特定のタグまたは属性値の解析

このフィルタは、フィルタ リストから手動で作成するか、または結果セット応答に対して組み込みのフィルタ コマンドを使用して作成できます。

フィルタを手動で作成する方法

以下のパラメータを入力します。

フィルタ

ステップの最終応答と見なすプロパティの名前。プロパティがプルダウン メニューにない場合は入力できます。プロパティは存在している必要があります。

タグ

HTML タグの名前。 画像タグの場合は「IMG」と入力します。

タグ数

応答の先頭からのタグの数。 最初の画像タグの場合は「1」と入力します。

属性

フィルタする属性の名前。 ソース属性の場合は「src」と入力します。

プロパティ

値を格納するプロパティ。

デフォルト（見つからない場合）

属性値が見つからない場合に使用する値。

URL エンコード

選択すると、プロパティ値が URL エンコードされます。

フィルタ実行結果

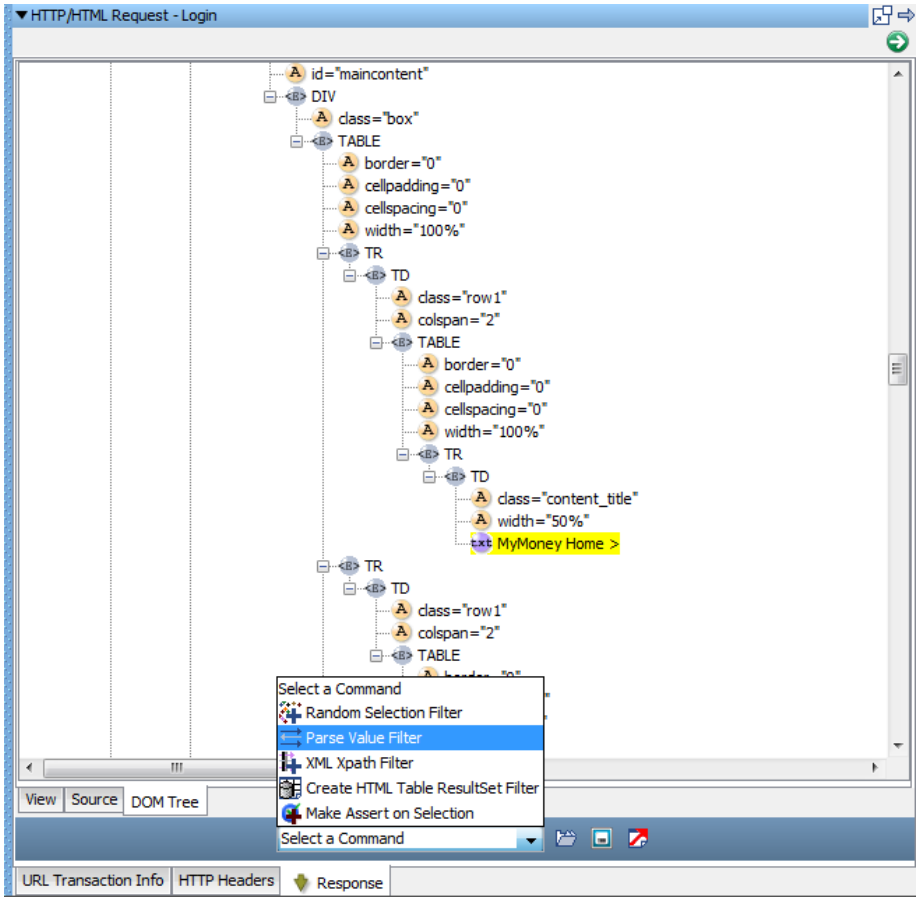
フィルタの実行結果として設定されるプロパティおよび値を表示します。

フィルタを実行

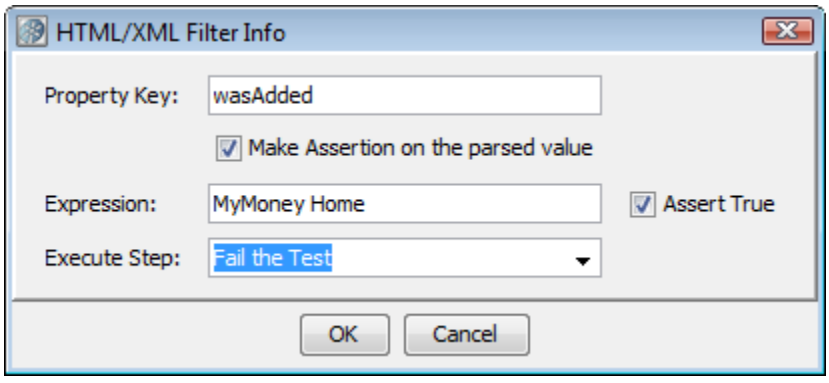
フィルタを実行するには、［フィルタを実行］ボタンをクリックします。 結果は［フィルタ実行結果］セクションに表示されます。

HTTP/HTML 要求ステップ応答ページからフィルタを作成する方法

1. HTML 応答が含まれるステップ応答を表示します。



2. DOM ツリー ビューから、プロパティにその値を格納する属性を選択します。
3. その値が強調表示されたら、［解析値フィルタ］を選択します。
4. ウィンドウにプロパティ名を入力します。



ここでアサーションも追加できます。

HTML 結果の特定の値の解析と値の解析

HTML 結果の特定のタグまたは属性値の解析と値の解析フィルタは、以下のフィルタを組み合わせたものです。

- HTML 結果の属性値の解析
- 引数文字列としてのプロパティ値の解析

このフィルタは、Web ページの特定の属性を検索し、さらにその属性を解析するように設計されています。 属性が URL であり、名前と値のペアでない場合は、その情報を処理するための機能があります。

この例では、フィルタは 7 番目のアンカー タグの「href」属性を検索します。 これは URL です。 フィルタは「cmd」パラメータを取り、**cmdlistusers_KEY** にその値を格納します。

以下のパラメータを入力します。

フィルタ

ステップの最終応答と見なすプロパティの名前。プロパティがプルダウン メニューにない場合は入力できます。プロパティは存在している必要があります。

タグ

HTML タグの名前。 アンカー タグの場合は「a」と入力します。

タグ数

応答の先頭からのタグの数。 7 番目のアンカー タグとして「7」を入力します。

属性

フィルタする属性の名前。 href 属性の場合は「href」と入力します。

IsURL

属性値が URL の場合は、このチェック ボックスをオンにします。

解析する引数

値を解析する引数の名前。この例では「cmd」です。

プロパティ

値を格納するプロパティ。

この例では「cmdlistusers_KEY」です。

URL エンコード

選択すると、プロパティ値が URL エンコードされます。

デフォルト（見つからない場合）

属性値が見つからない場合に使用する値。

フィルタ実行結果

フィルタの実行結果として設定されるプロパティおよび値を表示します。

フィルタを実行

フィルタを実行するには、［フィルタを実行］ボタンをクリックします。 結果は［フィルタ実行結果］セクションに表示されます。

HTML 結果のタグの子テキストの解析

HTML 結果のタグの子テキストの解析フィルタでは、タグの子テキストのテキストをプロパティに格納できます。

以下のパラメータを入力します。

フィルタ

ステップの最終応答と見なすプロパティの名前。プロパティがプルダウン メニューにない場合は入力できます。プロパティは存在している必要があります。

タグ

タグのタイプ。 たとえば h1 タグの場合は「h1」と入力します。

タグ数

タグが出現する数。 3 番目の h1 タグの子テキストの場合は「3」と入力します。

プロパティ

値を格納するプロパティ。

フィルタ実行結果

フィルタの実行結果として設定されるプロパティおよび値を表示します。

フィルタを実行

フィルタを実行するには、［フィルタを実行］ボタンをクリックします。 結果は［フィルタ実行結果］セクションに表示されます。

HTML 結果の HTTP ヘッダ値の解析

HTML 結果の HTTP ヘッダ値の解析フィルタでは、返された HTTP ヘッダ キーの値をプロパティに格納できます。

このフィルタの一般的な使用方法は、SERVER_NAME という名前のプロパティに HTTP ヘッダ Server を保存することです。

以下のパラメータを入力します。

フィルタ

ステップの最終応答と見なすプロパティの名前。プロパティがプルダウン メニューにない場合は入力できます。プロパティは存在している必要があります。

HTTP ヘッダ キー

HTTP ヘッダの名前（例： 「Server」）。

プロパティ

値を格納するプロパティ。

フィルタ実行結果

フィルタの実行結果として設定されるプロパティおよび値を表示します。

フィルタを実行

フィルタを実行するには、［フィルタを実行］ボタンをクリックします。 結果は［フィルタ実行結果］セクションに表示されます。

HTML 結果の属性値の解析

HTML 結果の属性値の解析フィルタでは、特定の属性のテキストをプロパティに格納できます。 属性は、スクリプト コードを含めて、結果のどのような場所にも出現します。

以下のパラメータを入力します。

フィルタ

ステップの最終応答と見なすプロパティの名前。プロパティがプルダウン メニューにない場合は入力できます。プロパティは存在している必要があります。

属性

取得する属性のタイプ。 たとえば、アンカー タグの URL の場合は「href」と入力します。

カウント

タグが出現する数。 たとえば、ページの 3 番目のアンカー タグの URL の場合は「3」と入力します。

プロパティ

値を格納するプロパティ。 この例では「anchor3」になります。

フィルタ実行結果

フィルタの実行結果として設定されるプロパティおよび値を表示します。

フィルタを実行

フィルタを実行するには、［フィルタを実行］ボタンをクリックします。 結果は［フィルタ実行結果］セクションに表示されます。

HTML 結果の DevTest タグの解析

HTML 結果の DevTest タグの解析フィルタでは、開発者が Web アプリケーションをテスト可能にできます。 テストの有効化の詳細については、「*SDK の使用*」を参照してください。

このフィルタでは、Web ページに「LISAPROP」タグを挿入できます。 LISAPROP タグには、name と value という 2 つの属性があります。LISAPROP タグは Web ページには表示されません。 それらの機能は、テスターに Web ページに関する重要な情報を提供する用途のみです。 LISAPROP の例を以下に示します。

```
<LISAPROP name="FIRST_USER" value="sbellum">.
```

テスターがこのタイプのフィルタをインストールした場合、**FIRST_USER** プロパティには **sbellum** という値が自動的に割り当てられます。このフィルタによって、テスターがこの値を解析する必要がなくなります。 このタイプのフィルタは、開発者によるテストを簡単にするのに役立ちます。

Web ページには適切な検証に必要な情報が含まれていないか、または情報を解析するのが難しいことがよくあります。 情報がある場合でも、生成される HTML の微妙な変化のために解析が正しくなくなることもあります。 このフィルタでは、Web テストの解析に関する問題の多くを解決できます。

パラメータは不要です。

ランダム HTML 属性の選択

ランダム HTML 属性の選択フィルタでは、セットからランダムに選択したテキストをプロパティに格納できます。 属性は、スクリプト コードを含めて、結果のどのような場所にも出現します。

以下のパラメータを入力します。

フィルタ

ステップの最終応答と見なすプロパティの名前。プロパティがプルダウン メニューにない場合は入力できます。プロパティは存在している必要があります。

外部タグ

選択するリストが含まれる外部のエレメント。 たとえば、ドロップダウン リストを選択するには、テキスト「select」を入力します。

タグ数

外部タグが出現する数。 たとえば、2 番目のドロップダウン リストを選択するには、テキスト「2」を入力します。

内部タグ

属性をランダムに選択するタグ。ドロップダウン リスト内のランダムなアイテムを選択するには、テキスト「option」を入力します。

フィルタ属性

選択リストに表示されるべきでない属性名を指定するオプションのフィールド。

フィルタ値

選択リストに表示されるべきでない属性値を指定するオプションのフィールド。

属性

そこからテキストを取得する属性。 このフィールドを空白にすると、内部タグの子テキストが返されます。

プロパティ キー

属性のテキストを格納するプロパティ。

フィルタ実行結果

フィルタの実行結果として設定されるプロパティおよび値を表示します。

フィルタを実行

フィルタを実行するには、［フィルタを実行］ボタンをクリックします。 結果は［フィルタ実行結果］セクションに表示されます。

HTML を属性リストに解析

HTML を属性リストに解析フィルタでは、1 セットの属性のテキストをリストとしてプロパティに格納できます。

以下のパラメータを入力します。

フィルタ

ステップの最終応答と見なすプロパティの名前。プロパティがプルダウン メニューにない場合は入力できます。プロパティは存在している必要があります。

外部タグ

解析するタグのリストが含まれる外部のエレメント。 たとえば、テーブルにすべてのアンカー タグからのリンクをすべて格納するには、「table」と入力します。

外部タグ数

外部タグが出現する数。2 番目のテーブルの場合は「2」と入力します。

内部タグ

そこから値を取得するタグ。テーブル内のすべてのアンカー タグの場合は「a」と入力します。

フィルタ属性

選択リストに表示されてはならない属性名を指定するオプションのフィールド。

フィルタ値

選択リストに表示されてはならない属性値を指定するオプションのフィールド。

属性

そこからテキストを取得する内部タグの属性。 このフィールドを空白にすると、内部タグの子テキストが返されます。 テーブルにすべてのアンカー タグからのリンクをすべて格納するには、「href」と入力します。

プロパティ キー

属性のテキストを格納するプロパティの名前。

フィルタ実行結果

フィルタの実行結果として設定されるプロパティおよび値を表示します。

フィルタを実行

フィルタを実行するには、［フィルタを実行］ボタンをクリックします。 結果は［フィルタ実行結果］セクションに表示されます。

HTTP ヘッダ Cookie の解析

HTTP ヘッダ Cookie の解析フィルタでは、HTTP ヘッダの Cookie の値を解析し、特定のプレフィックスから始まるプロパティにそれらを格納することができます。

以下のパラメータを入力します。

フィルタ

ステップの最終応答と見なすプロパティの名前。プロパティがプルダウン メニューにない場合は入力できます。プロパティは存在している必要があります。

プロパティ プレフィックス

使用するプロパティ名を指定するために Cookie 名の前に付加されるテキスト文字列。 このため、これらのプロパティのフル ネームは、返された Cookie の名前によって異なります。 Cookie 名は、対話型テストラン（ITR）の［プロパティ］タブで識別できます。

フィルタ実行結果

フィルタの実行結果として設定されるプロパティおよび値を表示します。

フィルタを実行

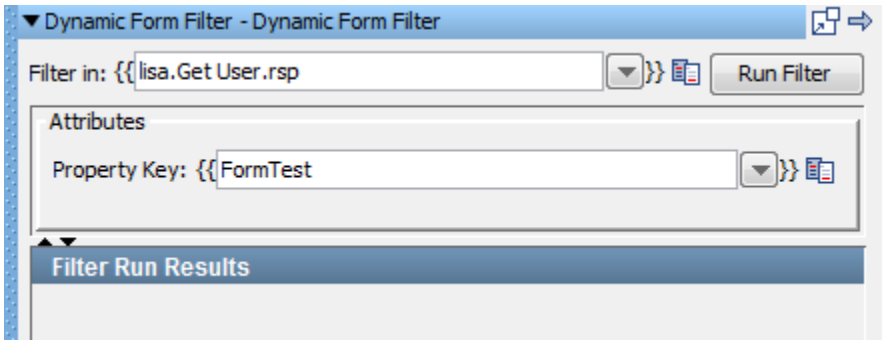
フィルタを実行するには、［フィルタを実行］ボタンをクリックします。 結果は［フィルタ実行結果］セクションに表示されます。

動的フォーム フィルタ

動的フォーム フィルタでは、HTML 応答で動的に生成されたフォームを識別し、それらを一連のプロパティに解析します。入力したプロパティ キーは、各フォーム内の各フォーム エレメントのプロパティ名の一部になります。 この動作は、以下の例を活用すると理解しやすくなります。

以下のような動的に生成されるフォームを 2 つ持つ HTML ページをテストできます。

```
<form name="F001" action="index.jsp"> <input type="text" name="0001A"
value="default" /> <input type="text" name="0001B" value="" /></form>
<form name="F002" action="orders.jsp"> <input type="text" name="0002A"
value=Key"" /> <input type="text" name="0002B" value="" /></form>
```



フィルタ パネルでプロパティ キー FormTest を使用して、以下のキー値ペアを作成します。

キー	値
FormTest.Form1.text1.name	0001A
FormTest.Form1.text1.value	default
FormTest.Form1.text2.name	0001B
FormTest.Form1.text2.value	
FormTest.Form2.text1.name	0002A
FormTest.Form2.text1.value	
FormTest.Form2.text2.name	0002B
FormTest.Form2.text2.value	

タグ/属性値の検索による HTML 結果の解析

タグ/属性値の検索による HTML 結果の解析フィルタでは、そのタグ内の別の属性の名前と値を検索することによって、タグ属性の値をフィルタできます。複数のタグが条件に一致する場合は、いずれかを指定できます。

以下のパラメータを入力します。

フィルタ

ステップの最終応答と見なすプロパティの名前。プロパティがプルダウン メニューにない場合は入力できます。プロパティは存在している必要があります。

タグ

検索するタグの名前。

検索条件属性

検索する属性。

検索条件値式

検索する属性の式。

タグ数

検索条件を満たすタグから使用する特定のタグ。

属性

その値を必要とする属性。

プロパティ

値を格納するプロパティ。

デフォルト（見つからない場合）

属性値が見つからない場合に使用する値。

URL エンコード

選択すると、プロパティ値が URL エンコードされます。

フィルタ実行結果

フィルタの実行結果として設定されるプロパティおよび値を表示します。

フィルタを実行

フィルタを実行するには、［フィルタを実行］ボタンをクリックします。 結果は［フィルタ実行結果］セクションに表示されます。

XML フィルタ

以下のフィルタは、任意のテスト ステップの XML フィルタのリストで使用できます。

XML のテキストの解析 (P. 185)
XML タグからの属性の読み取り (P. 187)
XML 結果の DevTest タグの解析 (P. 189)
ランダム XML 属性の選択 (P. 190)
XML XPath フィルタ (P. 191)

XML のテキストの解析

XML のテキストの解析フィルタでは、タグの子テキストのテキストをプロパティに格納できます。 XML のテキストの解析フィルタを定義するには、フィルタのタイプを設定し、属性を 3 つ設定します。

このフィルタは、フィルタ リストから手動フィルタとして作成する方法か、XML 応答に対して組み込みのフィルタ コマンドを使用する方法の 2 つの方法で作成できます。

フィルタを手動で作成する方法

以下のパラメータを入力します。

フィルタ

ステップの最終応答と見なすプロパティの名前。プロパティがプルダウン メニューにない場合は入力できます。プロパティは存在している必要があります。 このフィルタに合わせてこの値を編集できます。

タグ

タグのタイプ。 たとえば、multiRef タグの子テキストを対象とする場合は、「multiRef」と入力します。

タグ数

タグが出現する数。 たとえば、最初の multiRef タグの子テキストを対象とする場合は、「1」と入力します。

プロパティ

値を格納するプロパティ。

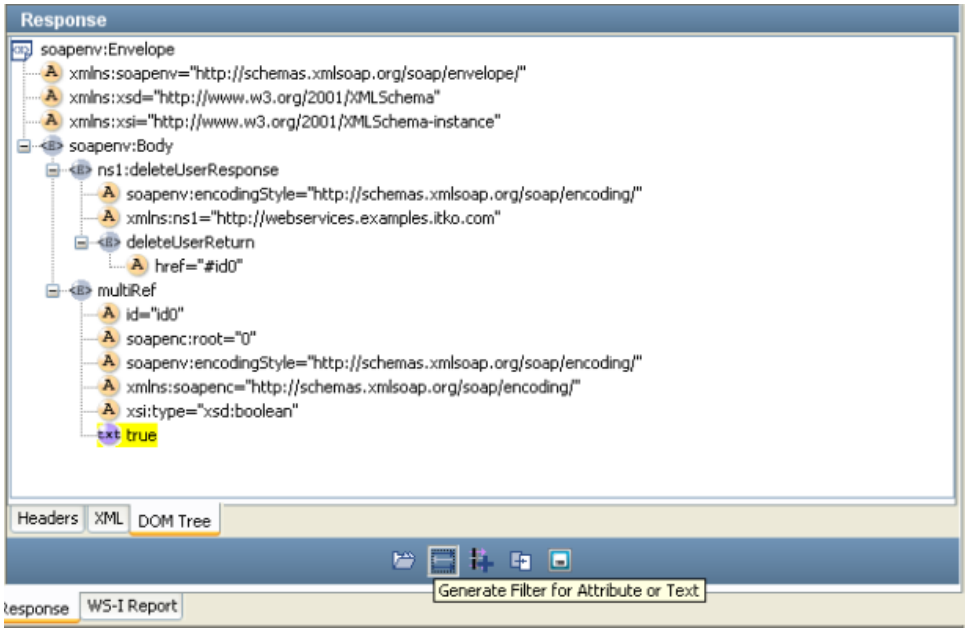
フィルタ実行結果

フィルタの実行結果として設定されるプロパティおよび値を表示します。

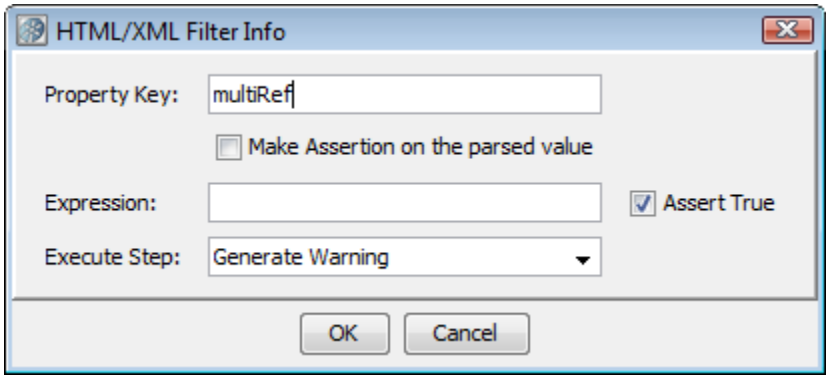
フィルタを実行

フィルタを実行するには、［フィルタを実行］ボタンをクリックします。 結果は［フィルタ実行結果］セクションに表示されます。

応答ページからフィルタを直接作成する方法



1. DOM ツリー ビューから、プロパティにその値を格納する属性を選択します。
2. 選択した後、［属性またはテキスト用のフィルタの生成］を選択します。
3. ダイアログ ボックスにプロパティ名を入力します。



ここでアサーションも追加できます。

XML タグからの属性の読み取り

XML タグからの属性の読み取りフィルタでは、特定の属性のテキストをプロパティに格納できます。 属性は、結果のどのような場所にも出現します。

このフィルタは、フィルタ リストから手動フィルタとして作成する方法か、XML 応答に対して組み込みのフィルタ コマンドを使用する方法の 2 つの方法で作成できます。

フィルタを手動で作成する方法

以下のパラメータを入力します。

フィルタ

ステップの最終応答と見なすプロパティの名前。プロパティがプルダウン メニューにない場合は入力できます。プロパティは存在している必要があります。

タグ

XML タグの名前（例： 「target」）。

タグ数

応答の先頭からのタグの数。最初のタグの場合は「1」と入力します。

属性

フィルタする属性の名前。href 属性の場合は「href」と入力します。

プロパティ

値を格納するプロパティ。

デフォルト（見つからない場合）

属性値が見つからない場合に使用する値。

URL エンコード

選択すると、プロパティ値が URL エンコードされます。

フィルタ実行結果

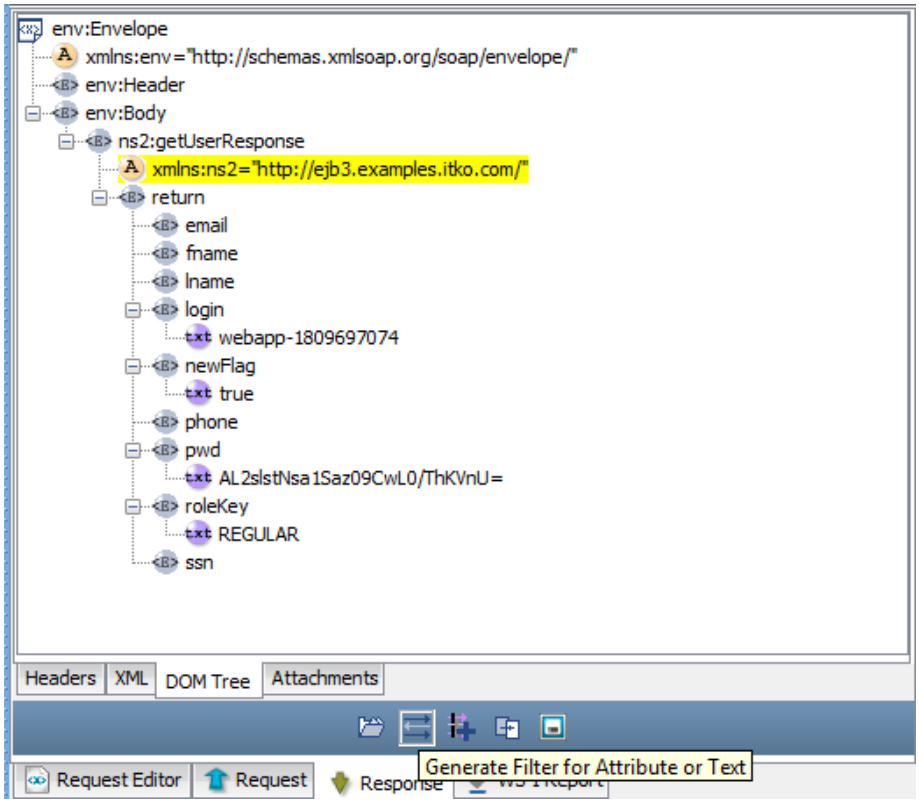
フィルタの実行結果として設定されるプロパティおよび値を表示します。

フィルタを実行

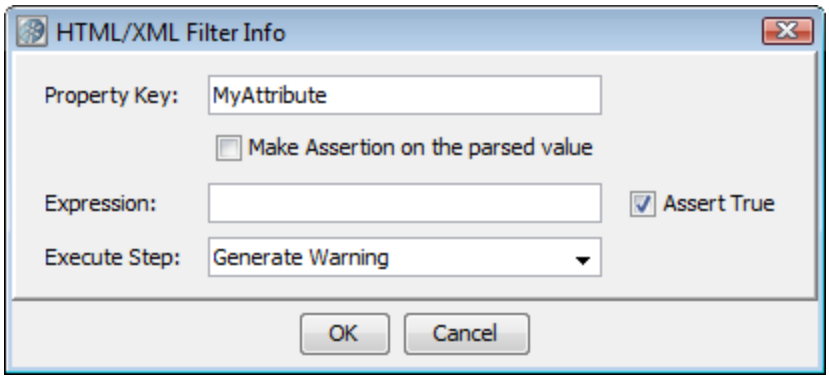
フィルタを実行するには、［フィルタを実行］ボタンをクリックします。 結果は［フィルタ実行結果］セクションに表示されます。

応答ページからフィルタを作成する方法

1. XML が含まれるステップ応答を表示します。



2. DOM ツリー ビューから、プロパティにその値を格納する属性を選択します。
3. 強調表示されたら、［属性またはテキスト用のフィルタの生成］を選択します。
4. ウィンドウにプロパティ名を入力します。



ここでアサーションも追加できます。 このステップには、プロパティ値式アサーションを追加できます。

XML 結果の DevTest タグの解析

XML 結果の DevTest タグの解析フィルタでは、開発者が XML アプリケーションをテスト可能にできます。 テストの有効化の詳細については、「*SDK の使用*」を参照してください。

このフィルタでは、XML ページに「LISAPROP」タグを挿入できます。 LISAPROP タグには、name と value という 2 つの属性があります。LISAPROP タグの機能は、テスターに XML に関する重要な情報を提供する用途のみです。 LISAPROP の例を以下に示します。

```
<LISAPROP name="FIRST_USER" value="sbellum">.
```

テスターがこのタイプのフィルタをインストールした場合、「FIRST_USER」プロパティには **sbellum** という値が自動的に割り当てられます。 このフィルタによって、テスターがこの値を解析する必要がなくなります。 このタイプのフィルタは、開発者によるテストを簡単にするのに役立ちます。

XML には適切な検証に必要な情報が含まれていないか、または情報を解析するのが難しいことがあります。情報がある場合でも、生成される XML の微妙な変化のために解析が正しくなくなることもあります。 この LISAPROP フィルタでは、解析に関する問題の多くを解決できます。

パラメータは不要です。

ランダム XML 属性の選択

ランダム XML 属性の選択フィルタでは、セットからランダムに選択したテキストをプロパティに格納できます。 属性は、結果のどのような場所にも出現します。 このフィルタは、ランダム HTML 属性の選択 (P. 178)とまったく同様に機能します。

XML XPath フィルタ

XML XPath フィルタでは、プロパティまたは最終応答に対して実行する XPath クエリを使用して、それをプロパティに格納できます。 このフィルタが選択されている場合、最終応答はコンテンツ パネルにロードされます。

応答は XML ドキュメントまたは DOM ツリーとして表示できます。ただし、XPath の選択は DOM ツリーからしか行えません。

以下のいずれかの方法を使用して、XPath クエリを作成します。

- ［XPath クエリ］テキスト ボックスに手動で XPath 式を入力する。
- DOM ツリーからエレメントを選択し、DevTest によって XPath 式が作成されるようにする。
- DOM ツリーからエレメントを選択し、作成する XPath を編集する。 たとえば、プロパティやカウンタ データ セットを使用するよう変更できます。

以下のパラメータを入力します。

フィルタ

ステップの最終応答と見なすプロパティの名前。プロパティがプルダウン メニューにない場合は入力できます。プロパティは存在している必要があります。

保存先プロパティ

XPath クエリの結果を格納するプロパティ。

上記の方法のいずれかを使用して、XPath クエリを作成します。

XPath クエリを作成した後、［フィルタを実行］をクリックしてテストします。 クエリの結果が［フィルタ実行結果］ペインに表示されます。

lisa.xml.xpath.computeXPath.alwaysUseLocalName プロパティは、XPath の作成の際に、XPath local-name() 関数が必ず使用されるかどうかを制御します。 デフォルト値は false です。これは、local-name() 関数が必要な場合にのみ使用されることを意味します。 XML ノードのネームスペースに関係なく動作する XPath を作成するには、値 を true に設定します。

JSON フィルタ

以下のフィルタは、任意のテスト ステップの JSON フィルタのリストで使用できます。

JSON パス フィルタ (P. 192)

JSON パス フィルタ

JSON パス フィルタでは、JSON オブジェクトから JSON プロパティ値を抽出してプロパティに保存することができます。 フィルタをクリックすると、そのエディタが開きます。

以下のパラメータを入力します。

フィルタ

ステップでフィルタするプロパティの名前を指定します。プロパティがプルダウン メニューにない場合は入力できます。プロパティは必ず必要です。このフィルタに合わせてこの値を編集できます。

JSON パス

JSON ドキュメント内の JSON プロパティのシーケンスから構成される式を指定します。 JSON パスは、送信先 JSON プロパティへのパスを表します。

「?()」式で表される配列フィルタは、特定の配列要素を選択するための選択基準として配列に適用できます。 たとえば、**?(@.age > 20)** を使用して、経過期間が 20 を超えている配列メンバを選択することができます。 別の例の **?(@.name in ('Mary', 'John'))** では、名前が Mary または John のいずれかである配列メンバが選択されます。

以下の表は、JSON データ型で利用可能な、サポートされるフィルタ演算子のリストを示しています。

演算子	String	数値	ブール値
== （等しい）	サポート対象	サポート対象	サポート対象
!= （等しくない）	サポート対象	サポート対象	サポート対象
>=		サポート対象	
<=		サポート対象	
>		サポート対象	
<		サポート対象	
in	サポート対象	サポート対象	

not in	サポート対象	サポート対象

Save Value to Property

JSON パスのプロパティ値が保存されるプロパティの名前を指定します。

Save Length to Property

JSON パスの属性値内のコンポーネントの数が保存されるプロパティの名前を定義します。 プロパティ値が配列の場合、コンポーネントの数は配列内の要素数です。 プロパティ値が JSON オブジェクトの場合、コンポーネントの数は JSON オブジェクト内のプロパティの数です。 文字列や数値などの単純なデータ型の場合、コンポーネントの数は 1 です。

フィルタ実行結果

フィルタの実行結果として設定されるプロパティおよび値を表示します。

フィルタを実行

フィルタを実行するには、［フィルタを実行］ボタンをクリックします。 結果は［フィルタ実行結果］セクションに表示されます。

Java フィルタ

以下のフィルタは、任意のテスト ステップの Java フィルタのリストで使用できます。

Java の "最終応答（Last Response）" プロパティの上書フィルタ (P. 194)
Java のステップ応答の格納フィルタ (P. 195)
Java のファイルへのプロパティ値の保存フィルタ (P. 196)

Java の "最終応答(Last Response)" プロパティの上書フィルタ

最終応答と呼ばれる特別なプロパティには、前のステップからの応答が含まれます。 たとえば、前の応答が HTTP ステップだった場合、最終応答は返された Web ページです。

最終応答をデフォルト値以外の値にしたい場合は、"最終応答（Last Response）" プロパティの上書フィルタを使用します。このフィルタでは、最終応答の現在値を既存のプロパティの値に置き換えることができます。

以下のパラメータを入力します。

フィルタ

ステップの最終応答と見なすプロパティの名前。プロパティがプルダウン メニューにない場合は入力できます。プロパティは存在している必要があります。

XML に変換

応答を有効な XML に変換する場合は、このオプションを選択します。

フィルタ実行結果

フィルタの実行結果として設定されるプロパティおよび値を表示します。

フィルタを実行

フィルタを実行するには、［フィルタを実行］ボタンをクリックします。 結果は［フィルタ実行結果］セクションに表示されます。

詳細については、「最終応答（Last Response）プロパティの上書 (P. 147)」を参照してください。

Java のステップ応答の格納フィルタ

ステップ応答の格納フィルタでは、最終応答を将来使用するためにプロパティとして保存できます。

以下のパラメータを入力します。

フィルタ

フィルタを適用する応答を入力します。 このフィルタに対するこの値を変更することはできません。

プロパティ

値を格納するプロパティ。

フィルタ実行結果

フィルタの実行結果として設定されるプロパティおよび値を表示します。

フィルタを実行

フィルタを実行するには、［フィルタを実行］ボタンをクリックします。 結果は［フィルタ実行結果］セクションに表示されます。

Java のファイルへのプロパティ値の保存フィルタ

ファイルへのプロパティ値の保存フィルタでは、既存のプロパティの値をファイル システム内のファイルに保存できます。

以下のパラメータを入力します。

フィルタ

ファイルに値を書き込むプロパティの名前。

場所

値を書き込むファイルのパス名。ファイルを参照できます。場所にプロパティを使用することもできます。

追加モード

既存のファイルに情報を追加する場合は、このチェック ボックスをオンにします。

フィルタ実行結果

フィルタの実行結果として設定されるプロパティおよび値を表示します。

フィルタを実行

フィルタを実行するには、［フィルタを実行］ボタンをクリックします。 結果は［フィルタ実行結果］セクションに表示されます。

VSE フィルタ

以下の 1 つのフィルタは、任意のテスト ステップの VSE フィルタのリストで使用できます。

データ プロトコル フィルタ (P. 197)

データ プロトコル フィルタ

データ プロトコル フィルタは、仮想モデルのプロトコル固有のリスン ステップで使用します。このフィルタは、フィルタとして動作するデータ プロトコルに必要なラッパーを提供します。これは、LISA 仮想サービス環境のランタイム側で機能するものに適した方法です。

フィルタをクリックすると、そのエディタが開きます。

以下のパラメータを入力します。

フィルタ

フィルタを適用する応答を入力します。上記の図では、**lisa.Get User.rsp** と示されています。これは、フィルタが Get User （ユーザの取得）の応答に適用されることを意味しています。 このフィルタに対するこの値を変更することはできません。

データ プロトコル

ドロップダウン リストから使用する適切なデータ プロトコルを選択します。

プロセス要求

プロセス要求を確認する場合に選択します。

プロセス応答

プロセス応答を確認する場合に選択します。

フィルタを実行

フィルタを実行するには、［フィルタを実行］ボタンをクリックします。 結果は［フィルタ実行結果］セクションに表示されます。

CAI のフィルタ

以下のフィルタは、任意のテスト ステップの CAI フィルタのリストで使用できます。

CA Continuous Application Insight の統合 (P. 198)
webMethods Integration Server の統合 (P. 199)

CA Continuous Application Insight の統合

CA CAI の DevTest 統合サポート フィルタは、DevTest がサポートするすべてのテクノロジに対して CAI を有効にする共通のフィルタです。このフィルタは、CAI アプリケーションから追加の情報を収集します。

現在、DevTest では、Web サービス、JMS、サーブレット、EJB、および Java オブジェクトとの統合をサポートしています。

以下のパラメータを入力します。

次の最大ビルド時間(ミリ秒)を超えた場合はエラー

ビルド時間をミリ秒で入力します。 ビルド時間が指定された間隔を超えると、エラーが生成されます。

トランザクション エラー時のステップ

フィルタを実行するよう設定した後、トランザクション エラーが発生したときにリダイレクトするステップを選択します。

CAI 警告時のステップ

フィルタを実行するよう設定した後、CAI 警告が発生したときにリダイレクトするステップを選択します。

コンポーネント コンテンツをレポート

コンポーネント コンテンツのレポートを生成します。

要求の開始および終了時にサーバ上でガベージ コレクションを強制実行

要求の開始および終了時に、サーバ上でガベージ コレクションを強制的に実行します。

サーバ側例外が記録されたらテスト失敗

サーバ側で例外がスローされた場合に、テスト ケースを失敗にします。

テスト イベントでキャプチャする Log4J レベル

テスト イベントでキャプチャされる Log4J レベルを選択します。

一時的に変更する Log4J ロガー(空白にした場合、ルート ロガー)

ロガーの名前を入力します。

webMethods Integration Server の統合

webMethods Integration Server の DevTest 統合サポート フィルタでは、CAI が有効にされた webMethods Integration Server から多くの情報を収集します。

以下のパラメータを入力します。

次の最大ビルド時間(ミリ秒)を超えた場合はエラー

ビルド時間をミリ秒で入力します。 ビルド時間が指定された間隔を超えると、エラーが生成されます。

トランザクション エラー時のステップ

フィルタを実行するよう設定した後、トランザクション エラーが発生したときにリダイレクトするステップを選択します。

CAI 警告時のステップ

フィルタを実行するよう設定した後、CAI 警告が発生したときにリダイレクトするステップを選択します。

Copybook フィルタ

以下のフィルタは、任意のテスト ステップの Copybook フィルタのリストで使用できます。

Copybook フィルタ (P. 200)

Copybook フィルタ

Copybook フィルタは、実行時に Copybook ペイロードを XML に変換します。これらのペイロードを XML で表示することにより、データを確認してメンテナンスすることができます。

フィルタをクリックすると、そのエディタが開きます。

以下のパラメータを入力します。

フィルタ

ステップでフィルタするプロパティの名前を指定します。プロパティがプルダウン メニューにない場合は入力できます。プロパティは必ず必要です。このフィルタに合わせてこの値を編集できます。

Copybook definition file

Copybook ファイル定義を格納する場所を指定します。

エンコーディング

有効な Java 文字セットを選択します。この値は、ペイロード内のバイトを出力 XML で使用されるテキストに変換するために使用されます。

デフォルト： UTF-8

Copybook **パーサ開始列**

Copybook は、多くの場合、行番号で各行を開始します。このパラメータは、Copybook ファイル定義を解析する場合、パーサがどの列から開始するかを定義します。

値： 包含的なゼロベース インデックス。ただし、「通常」の排他的な 1 ベース インデックスと見なすことができます。

デフォルト： 6

例： この値に 6 を設定した場合、パーサは、行の最初の 6 文字をスキップして、7 文字目から開始します。

Copybook **パーサ終了列**

場合によっては、Copybook の各行の末尾にその他の参照データが含まれることがあります。その場合、パーサは、どの列で停止するかを知る必要があります。ファイル内の行の末尾に「余分な」データがない場合、この数値を、ファイル内の最長行の長さより大きく設定します。この数値が行の長さを超えている場合、パーサは行の末尾で停止します。

値： 排他的なゼロベース インデックス。ただし、「通常」の包含的な 1 ベース インデックスと見なすことができます。

例： この値に 72 を設定した場合、パーサは、行内の 72 番目の文字を読み取った後に停止します。

フィルタ実行結果

フィルタの実行結果として設定されるプロパティおよび値を表示します。

フィルタを実行

フィルタを実行するには、［フィルタを実行］ボタンをクリックします。 結果は［フィルタ実行結果］セクションに表示されます。

パフォーマンス上の理由で、Copybook フィルタは、Copybook 定義をメモリ内に 86400 秒間キャッシュします。 この期間が終了すると、DevTest は変換された Copybook 定義をキャッシュから削除します。 ファイルが再度必要な場合、DevTest はそれを読み取って再変換します。

テスト ステップの説明

このセクションには、以下のテスト ステップの説明が含まれています。

テスト ステップ情報

いくつかの標準のテスト ステップを以下に示します。

テストを中止する

次の手順に従ってください:

1. その後にテスト ケースを中止させるテスト ステップを右クリックします。
2. ［次のステップ］-［テストを中止する］を選択します。

   テストを中止するステップはテスト ケースを終了し、中止したとしてステップをマークします。

テストを終了する

次の手順に従ってください:

1. その後にテスト ケースを終了させるテスト ステップを右クリックします。
2. ［次のステップ］-［テストを終了する］を選択します。

   このステップはテストを完了し、正常に終了したとしてテストをマークします。

テストを失敗させる

次の手順に従ってください:

1. その後にテスト ケースを失敗させるテスト ステップを右クリックします。
2. ［次のステップ］-［テストを失敗させる］を選択します。

   テストを失敗させるステップはテスト ケースを失敗させ、失敗したとしてテストをマークします。

Web / Web サービス ステップ

以下のステップが使用できます。

HTTP/HTML 要求ステップ

このステップは、HTTP(S) 要求を送受信する従来の Web アプリケーションをテストするために使用されます。 要求には GET パラメータおよび POST パラメータを含めることができます。また、必要に応じて、応答として埋め込みイメージを含めることができます。 また、Web サイト プロキシ レコーダを使用して、HTTP ステップを記録できます。

HTTP/HTML 要求 ステップには、「HTTP(s)（「GET」または「POST」）（URL のリーフ）」という命名規則を使用したデフォルトの名前があります。 例は **HTTP GET rejectCard.jsp** です。 ステップ名は、いつでも変更できます。

設計時に手動で HTTP/HTML ステップを実行できます（WS ステップと同様）。 ［アクション］-［ここから再生］を選択すると、ステップ エディタで応答値を表示できるようにステップ応答が保存されます。

テスト ケースにこのステップを追加した場合、ステップ エディタには以下のタブが含まれます。

- ［URL トランザクション情報］タブ（P. 205）
- ［HTTP ヘッダ］タブ（P. 209）
- ［Response］タブ（P. 209）
- ［SSL］タブ（P. 210）

［URL トランザクション情報］タブ

［URL トランザクション情報］タブで URL の構築に使用する情報を指定します。

以下のオプションのいずれかを使用して、URL トランザクション情報を設定できます。

- URL を分割して指定
- プロパティの使用

URL を分割して指定

URL を必要な部分に分割して指定するには、［URL を分割して指定］オプション（デフォルト）を選択します。

プロトコル

Web サーバとの通信に使用されるプロトコル。 デフォルトは **http** です。

ホスト名

Web サーバのホスト名。 SERVER プロパティを使用するか、またはアプリケーション サーバのホスト名または IP アドレスを入力します。 ホスト名は、www.mycompany.com などのドメイン名、または 123.4.5.6 などの IP アドレスが可能です。 ローカル Web サーバの場合は、ホスト名 **localhost** または IP アドレス **127.0.0.1** を使用します。

ポート

（オプション）必要に応じて、PORT プロパティまたは Web サーバにアクセスするために使用される Web サーバのポートを使用します。たとえば、Apache Tomcat Web サーバにアクセスするために必要なポートは、デフォルトでは 8080 です。

パス

アクセスするファイルのパス。 たとえば、アクセスする URL が http://localhost:8080/mysite/index.jsp である場合は、［パス］フィールドに「mysite/index.jsp」と入力します。

ユーザ

アプリケーション サーバにユーザ ID が必要な場合に入力します。

パスワード

アプリケーション サーバにパスワードが必要な場合に入力します。

エンコーディング

［エンコーディング］ドロップダウンは、次のプロパティによって制御されます：**lisa.supported.html.request.encodings**=ISO-8859-1, UTF-8, Shift_JIS, EUC-JP, Windows-31J。

サポートするエンコーディングを含めるためにカンマ区切りリストを変更できます。 また、基盤となる JVM は、このリストのエンコーディングをすべてサポートする必要があります。 このリストでサポートされていないエンコーディングを Web ページが使用する場合、ドロップダウンのエントリは空白です。 その状態で保存すると、DevTest は、エンコーディングを DevTest のデフォルト（lisa.properties 内の file.encoding キー）に置き換えます。 また、HTTP/HTML 要求ステップを作成するときにエンコーディングが選択されていない場合は、DevTest のデフォルトのエンコーディングが使用されます。

URL パラメータ

GET （URL）要求パラメータ： これらの要求パラメータは、URL の一部として渡されます。そのため、Web ブラウザのアドレス バーに表示されます。

POST パラメータ

POST 要求パラメータ。 この要求パラメータは、ページ要求のボディの一部として渡されます。 これらは、Web ブラウザのアドレス バーに表示されません。

フォーム エンコーディング

ステップの実行中、パラメータは送信時に URL エンコードされます。 使用される MIME タイプは application/form-urlencode です。

すべての既知の状態

テスト ケース プロパティ、データ セット、およびフィルタなどの既知のプロパティがすべてリスト表示されます。

参照されているファイルをダウンロード（img、link、script、embed）

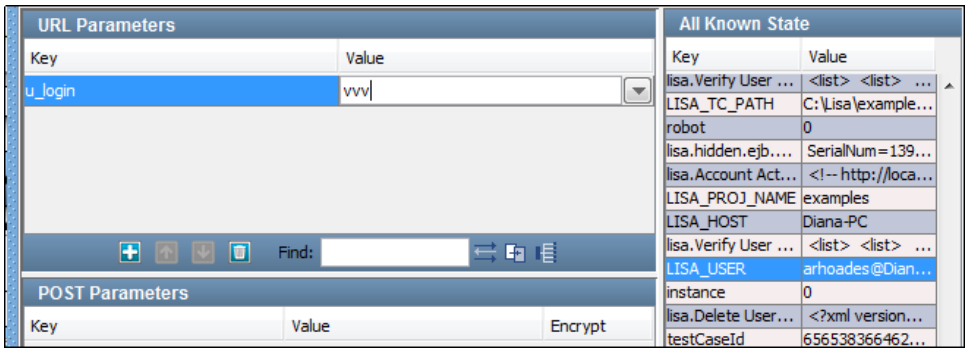
このエレメントが選択されている場合、ステップはテスト環境に Web ページ イメージをダウンロードします。 このボックスをオンにしない場合、イメージはダウンロードされません。

以下の表では、［URL パラメータ］および［POST パラメータ］セクションで使用可能なツールバーの一部であるその他の機能について説明します。

フィールド	アイコン	説明
追加		要求パラメータを追加します
稼働		パラメータのリストで既存のパラメータを上へ移動させます
ダウン		パラメータのリストで既存のパラメータを下へ移動させます
削除		既存のパラメータを削除します
検索		テキストを検索します
フィルタの自動生成		参照するステップからこのパラメータを動的に生成します。 実行時にこのプロパティを自動入力するための新しいフィルタを作成します。 フィルタの詳細については、「フィルタ」を参照してください。
選択された［すべての既知の状態］のプロパティを適用		パラメータに状態を適用します。状態の適用の詳細については、以下の「すべての既知の状態」を参照してください。
［すべての既知の状態］プロパティを自動適用		パターンによって可能なすべてのプロパティにすべての状態を適用します。 状態の適用の詳細については、以下の「すべての既知の状態」を参照してください。

すべての既知の状態

テスト ケース プロパティ、データ セット、およびフィルタなどのすべての既知のプロパティは、［すべての既知の状態］パネルに表示されます。



URL 要求 パラメータにプロパティの値を割り当てることができます。

たとえば、**LISA_USER** データ セット キーの値を上記の例の **u_login request** パラメータに割り当てるには、以下の手順に従います。

1. ［URL パラメータ］ペインで、**u_login** キーを選択します。
2. ［すべての既知の状態］パネルで、**LISA_USER** キーを選択します。
3. ［選択された［すべての既知の状態］のプロパティを現在のパラメータに適用します］ をクリックします。

   変更を確認する警告メッセージが表示されます。
4. ［OK］をクリックします。

   新しいプロパティが［URL パラメータ］ペインに表示されます。

   URL パラメータ キーのすべての名前が［すべての既知の状態］キーの名前と同じである場合、［すべてに適用する］をクリックして、関連するパラメータに迅速にすべてのプロパティを割り当てることができます。

［プロパティの使用］オプション

［プロパティの使用］オプション ボタンが選択されている場合、以下のパラメータを指定できます。

プロパティ キー

接続情報が含まれるプロパティを指定します。

参照されているファイルをダウンロード（img、link、script、embed）

このエレメントが選択されている場合、ステップはテスト環境に Web ページ イメージをダウンロードします。 このボックスをオンにしない場合、イメージはダウンロードされません。

［HTTP ヘッダ］タブ

［HTTP ヘッダ］タブでは、任意のカスタム HTTP ヘッダを作成します。

- 上部の［カスタム HTTP ヘッダ（現在のみ）］セクションは、この要求でのみサーバに送信されるヘッダ用です。
- 下部の［カスタム HTTP ヘッダ（永続）］セクションは、このトランザクションおよびテストのその他のすべてのトランザクションで送信されるヘッダ用です。

いずれかのセクションで要求パラメータを作成するには、［追加］ をクリックして目的の値のキーおよび値を変更します。

［応答］タブ

［応答］タブには、このテストが記録されたときにサーバが返す HTTP 応答が表示されます。 以下のものが表示されます。

- 応答のソース。
- 応答の DOM ツリー。

[SSL]タブ

［SSL］タブでは、単一または複数の SSL 証明書に情報を入力できます。

単一の SSL 証明書の使用

以下の情報を入力します。

SSL キーストア ファイル

クライアント アイデンティティ証明書が格納されるキーストア ファイルの名前。 このファイルは JKS または PKCS 形式です。

SSL キーストア パスワード

キーストア ファイルのパスワード。

SSL キー エイリアス

サーバの秘密鍵を格納および取得するために使用されるエイリアスを定義するキーストア属性。

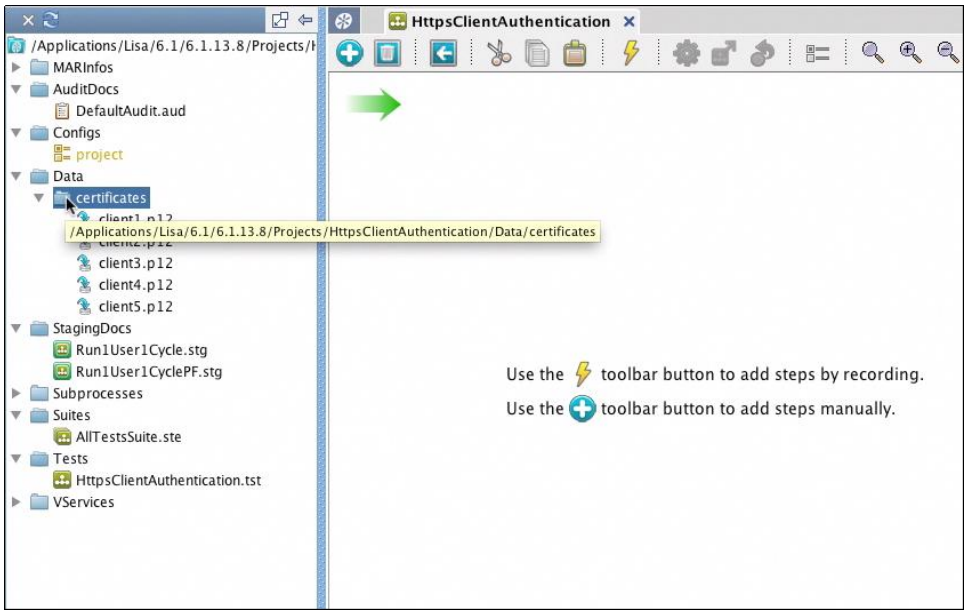
SSL キー パスワード

JKS キーストアを使用する場合に、キーにキーストアとは異なるパスワードがあるときのキー エントリ用のオプションのパスワード。

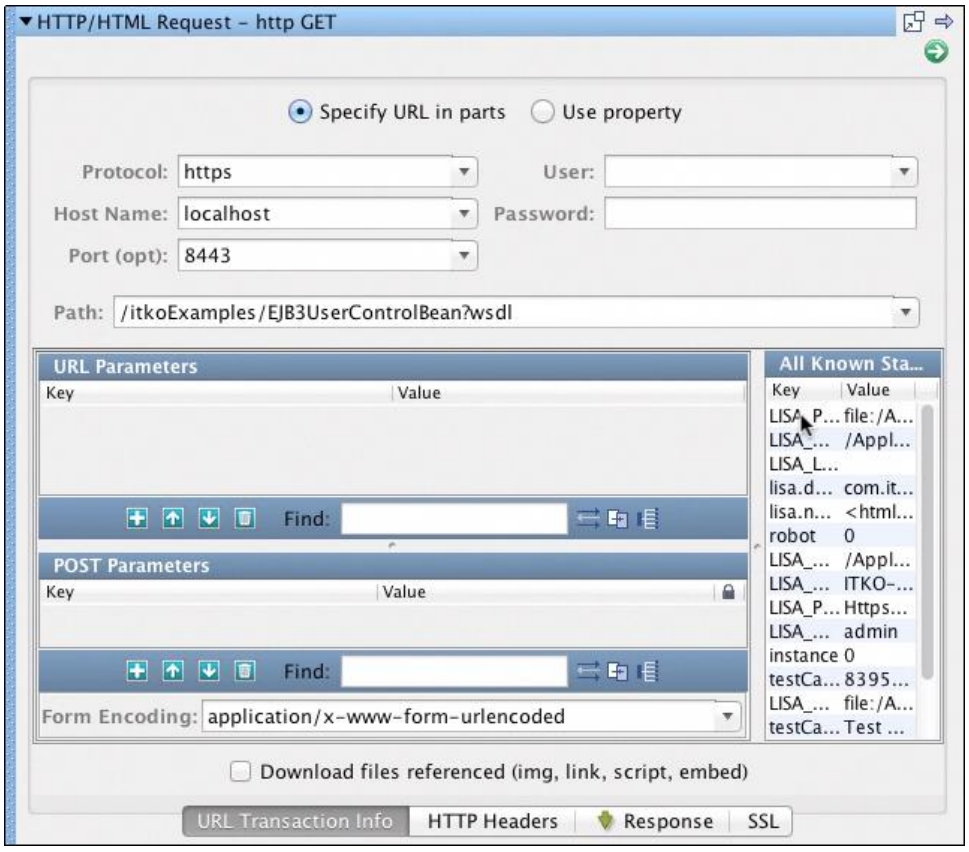
複数の SSL 証明書の使用

1 つのステップで複数の証明書を使用するには、データ セットに SSL 情報を格納します。 以下の例は、複数の証明書に関する情報を格納するためにデータ シートを使用する方法を示しています。

この例では、デモ サーバはバックグラウンドで実行されており、それはクライアント側認証を必要とするように設定されています。 デモ サーバは、5 つの証明書を受理します。それらの証明書のキーストア ファイルは、プロジェクト構造の Data フォルダの「**certificates**」に配置されています。これらのキーストア ファイルは pkcs12 キーストアです。

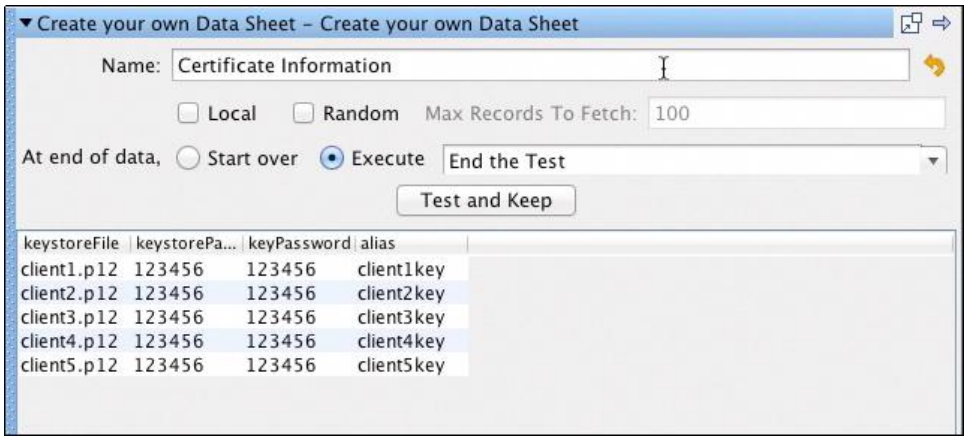


次に、HTTPS 要求ステップを追加し、https をプロトコルとして指定します。また、ポートおよびパスを入力します。

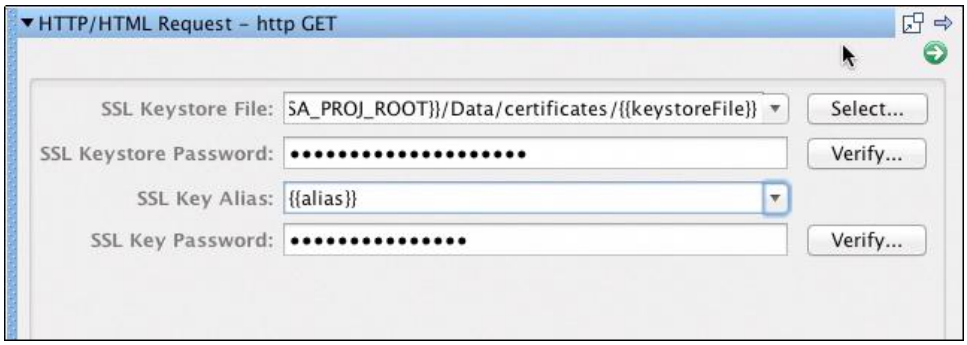


このプロセスがすべての証明書で動作することを確認し、テストを 1 回のみ実行するために、このテスト ステップ用にデータ セットを作成します。データ セットを作成するために、［ユーザ定義データ シートの作成］データ セットを使用します。 データ セットには **Certificate Information** という名前を付け、データ セットの各行を 1 回実行して終了するように設定します。証明書にはそれぞれ 5 つの証明書および 4 つのパラメータがあります。情報を入力したら、［データ シート スケルトンの作成］ボタンを使用して、エディタを開きます。

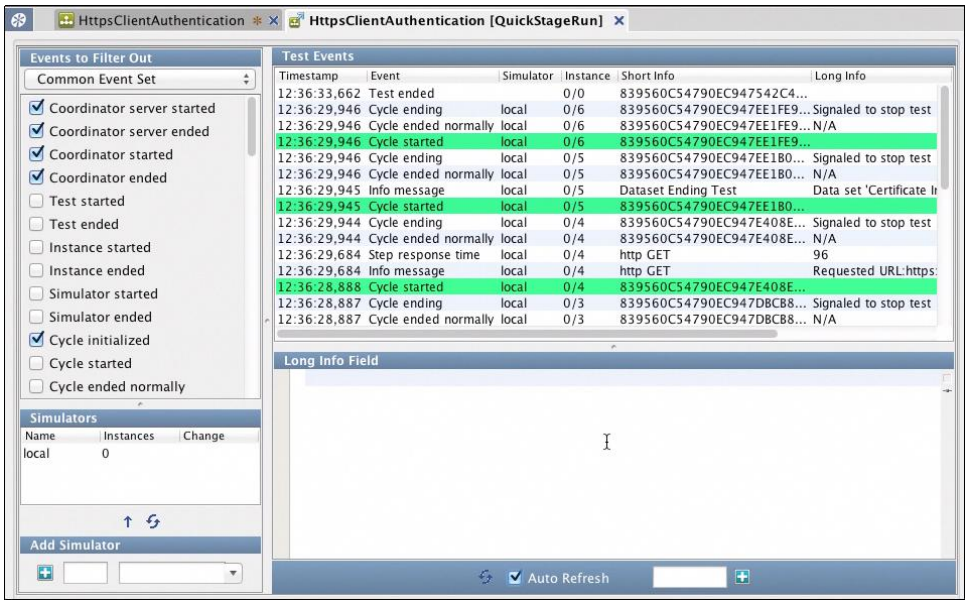
keystoreFile については、後で相対パスを使用できるように、キーストアの短いファイル名を使用します。 パスワード列を暗号化するには、列ラベルを右クリックして［暗号化］を選択します。



HTTP ステップに戻り、［SSL］タブを選択します。 変数にプロパティの置換を使用します。 パスワード フィールドに、「**{{keystorePassword}}**」および「**{{keyPassword}}**」を入力します。



テストは［クイック テストのステージング］を使用してステージングされ、すべてのテストが正常に完了しています。5 つのすべての証明書がデモ サーバに対してテストされ、クライアント認証はそれらのすべてに対して正常に動作しました。



REST ステップ

REST アプリケーションをテストする場合、REST ステップを使用します。

このステップは、GET および POST のパラメータを含む HTTP(S) 要求を送受信するために使用されます。

パラメータを表示するには、右の垂直バーの［テスト状態］ボタンをクリックします。［テスト状態］領域がスライドして開きます。このリストは、ドッキング、固定、または非表示にできます。

HTTP 応答を表示するには、［応答］ボタンをクリックします。

［応答］パネルの下部には、応答に追加できるフィルタおよびアサーションが表示されます。

デモ サーバには、LISA Bank ユーザ データベースから情報を取得する JSON サービスを呼び出すサンプルが含まれています。 URL は http://localhost:8080/rest-example/ です。

REST ステップのサンプル テスト ケースは、examples プロジェクトにあります。 テスト ケースの名前は **rest-example.tst** です。

Web サービス実行（XML）ステップ

Web サービス実行（XML）ステップは、HTTP POST または JMS メッセージを使用して、SOAP ベースの Web サービスの操作を実行するように設計されています。

WSDL へのアクセスは必要ではありません。推奨されてはいますが、オプションの設定情報です。 WSDL が設定されている場合、サービスに送信する SOAP メッセージを作成するプロセスで役立ちます。このステップでは、RAW SOAP メッセージ（XML）を直接操作できます。 この機能は柔軟性および強力な機能を提供しますが、Web サービスがどのように動作するかの詳細が公開されます。

一般的に、エディタの上部は、SOAP メッセージの送信方法および送信先に使用されます。 下部はメッセージのコンテンツに使用されます。

Web サービス実行（XML）ステップには、「Web サービス webServiceOperation 名」という命名規則を使用したデフォルトの名前があります。デフォルトのステップ名を別のステップが使用する場合、DevTest は、このステップ名に番号を追加して一意にします。 ステップ名は、いつでも変更できます。

テスト ステップを開くと 2 つのタブが表示され、それぞれのタブには複数のサブタブが表示されます。

［PRO］ PRO アイコンは、基本オプションと詳細オプションとの間で切り替わります。［PRO］が選択されている場合は、一部のタブおよびオプションのみが使用可能です。

［接続］タブ

［接続］タブには、接続のフィールドがあります。 このタブには、上部および下部のバーにサブタブがあります。

- 上部バー - ［ビジュアル XML］、［RAW XML］、［ヘッダ］、［添付］の表示用。
- 下部バー - ［要求］および［応答］用。
  - 基本設定
  - 設計時の実行

基本設定

接続

WSDL URL

［WSDL URL］はオプションですが、推奨フィールドです（薄い灰色で表示されます）。

［WSDL URL］は URL （file:/、http:/、または https:/ のいずれか）である必要があります。 追加オプションのメニュー  から、以下を実行できます。

- ファイル システムのローカルの WSDL または WSDL バンドル ファイルを参照する。
- （高度な UDDI アクセス ポイント ルックアップを行う） UDDI レジストリを検索する。
- レガシー WS ステップからマイグレートするために、hotDeploy から WSDL を選択する。
- WSDL バンドルを作成および使用する。［アクション］メニューから WSDL バンドルを作成することもできますが、［オプション］メニューからの場合、DevTest は、WSDL バンドルの結果ファイル URL を使用して WSDL URL を自動的に入力します。 または、すでに WSDL URL を入力している場合は、WSDL バンドル ダイアログ ボックスに WSDL URL が事前に入力されます。

まだ WSDL バンドルではない WSDL URL を入力すると、DevTest は、WSDL バンドルを作成し、プロジェクトの Data/wsdls ディレクトリにローカルに保存します。 この操作により、WSDL がローカルにキャッシュされ、迅速にアクセスできるようになります。 DevTest は、WSDL を解析し、そのスキーマを、サンプル SOAP メッセージを作成するために使用します。 ビジュアル XML エディタも、WSDL を使用して、SOAP メッセージの手動での編集を支援します。 DevTest は、WSDL の処理時は常に、まずキャッシュされた WSDL バンドルをロードしようとします。 外部の WSDL が変更されている場合にローカルの WSDL キャッシュを強制的に更新するには、

［WSDL キャッシュのリフレッシュ］  ボタンを使用します。 WSDL バンドルは、いつでも手動で Data/wsdls へドロップできます。 ステップは、「ライブ」WSDL URL を処理しようとするときに、キャッシュされたバンドルを代わりに使用します。

［サービス］、［ポート］、［操作］

WSDL URL が入力されると、WSDL が処理され、［サービス］、［ポート］、および［操作］の選択内容が入力されます。 これらの、オプションですが推奨されているフィールドは、サンプル SOAP 要求メッセージを作成するために使用できます。ポートを選択すると、WSDL の定義に一致するようにエンドポイント URL も更新されます。 WSDL URL を変更すると、その項目がリフレッシュされます。エンドポイント URL と SOAP メッセージが変更されていない場合は、それらも、選択された新しい WSDL、サービス、ポート、および操作に対応させるために更新されます。

エンドポイントが変更され、WSDL のエンドポイントに一致しなくなった場合、［警告］ボタンがフィールドの隣に表示されます。このボタンのツールヒントは、入力された値と WSDL 定義との違いを示します。このボタンをクリックすると、WSDL 定義に一致するようにフィールドが更新されます。

SOAP 要求メッセージがデフォルトに一致しなくなっている場合は、自動的に更新されません。 ［操作］フィールドの隣の［メッセージの構築］  ボタンを使用すると、SOAP 要求メッセージを強制的に更新できます。

操作

ここでは以下のオプションが使用できます。

- 空の SOAP 要求メッセージの構築
- 完全な SOAP 要求メッセージの構築

ポート

このフィールドは、サービスが使用可能なサーバ ポートを示します。

エラー時

このフィールドは、実行中にエラーが発生した場合にどのアクションが実行されるかを示します。

エンドポイント

アイデンティティ プロバイダの SAML クエリ API の URL。

構築オプション

サンプル SOAP メッセージを作成する場合、さまざまな構築オプションが使用され、さまざまな状況で実行することが決定されます。

- 値に文字列パターンを使用する：選択した場合、ハードコードされたリテラル値ではなく、DevTest 文字列パターンを使用して、エレメント値に入力します。
- デフォルト リテラル値： 文字列パターンを使用しない場合、このリテラル値をすべての文字列値に使用します。
- すべての選択を構築：デフォルトでは、XML スキーマで選択されている最初のエレメントのみが生成されます。すべての選択エレメントを作成するには、このオプションを選択します。

  注: 複数の選択エレメントを含める場合、SOAP 要求は有効ではありません。ただし、これは可能な選択のサンプルを提供しており、最初の選択を使用しない場合のメッセージの作成をより簡単にします。
- ［最大エレメント数］： サンプル メッセージを作成する場合に含めるエレメントの最大数を定義します。
- ［最大タイプ数］： サンプル メッセージを作成する場合に含める複雑なスキーマ タイプの最大数を定義します。
- コメントの挿入：デフォルトでは、スキーマに関するコメントが生成されます。 たとえば、エレメントがオプションの場合、代わりの選択肢（Nil 可エレメント）が生成されます。 ビジュアル XML エディタには、これらのコメントは表示されません。これらは、RAW エディタに表示されます。

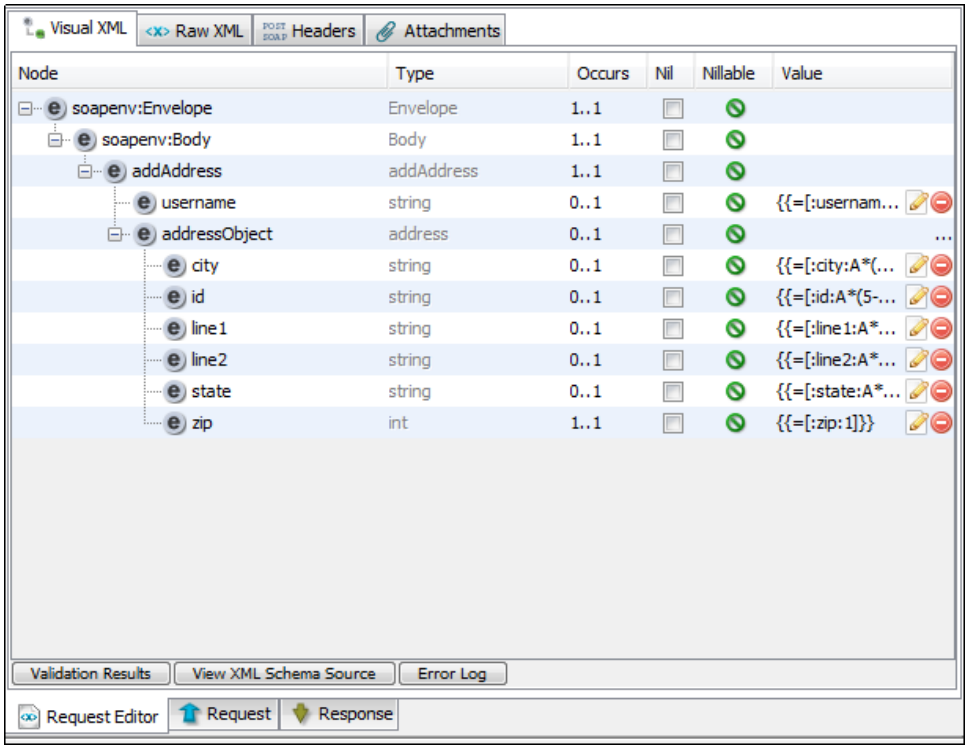
Web サービス実行タブ

ここでは、Web サービス実行エディタで使用可能なタブについて説明します。

［ビジュアル XML］タブ (P. 219)
［RAW XML］タブ (P. 223)
［ヘッダ］タブ (P. 224)
［添付］タブ (P. 227)
［アドレシング］タブ (P. 230)
［セキュリティ］タブ (P. 230)

[ビジュアル XML]タブ

ビジュアル XML エディタ（VXE）は、XML ドキュメント用のグラフィカルなエディタです。 SOAP メッセージは、SOAP 仕様（SOAP スキーマ）に準拠した XML ドキュメントです。 このエディタを使用して、SOAP メッセージを作成および編集できます。



このテーブルは、SOAP エンベロープやボディ エレメントを含む XML ドキュメントを示しています。

タイプ

各エレメントのタイプを示します。

出現数

予期されるエレメント数を示します。 最初の数は、エレメントが出現できる最小回数を示します。 ゼロは、それがオプションのエレメントで削除できることを意味します。 2 番目の数は、エレメントが出現できる最大回数を示します。 無限は、その名前のエレメントの出現回数が無制限であることを意味します。

Nil

エレメント値を **Nil にする**か、または **Nil を解除する**ことができます。 このチェック ボックスをオンにすると、エレメントの子または値も削除されますが、属性はすべてそのままです。 このチェック ボックスをオフにすると、WSDL スキーマの定義に従って、予期されるすべての子および属性が入力されます。

Nil 可

エレメントが Nil 可かどうかを示します。赤いアイコンは、エレメントが Nil 不可であるが、**Nil** に設定されることを示します。 緑のアイコンは、エレメントが Nil 不可であり、**Nil** に設定されないことを示します。

値

［値］列には、エレメント値を直接入力できます。 エレメント タイプが既知のタイプである場合、専用の編集ボタンが表示されます。

表示または非表示にする列を選択するには、列見出しを右クリックします。

エディタを右クリックすると、コンテキスト メニューにドキュメントの操作用のオプションが表示されます。 このメニューには以下のオプションがあります。

スキーマ属性/エレメントの追加

有効な属性またはエレメントのリストから選択できます。 スキーマが存在し、エレメントまたは属性がエディタで選択されている場合、子のエレメントおよび属性を選択して追加できます。 20 を超えるスキーマ オブジェクトが使用可能な場合、スキーマ オブジェクトを選択するダイアログ ボックスを使用できます。 このダイアログ ボックスには、検索テキスト フィールドが含まれます。 このフィールドにより、多数のスキーマ オブジェクトがある場合に、特定のスキーマ オブジェクトを見つけることが容易になります。

エレメントの追加

ドキュメントにエレメントを追加します。

属性の追加

ドキュメントに属性を追加します。

テキストの追加

エレメント/属性の削除

選択したエレメントまたは属性を削除します。

上/下に移動

選択したノードを上または下に移動します。

添付ファイルへの変換

標準の参照添付ファイルを作成するショートカット メソッド。 詳細については、「［添付］タブ (P. 227)」を参照してください。

XOP 添付ファイルへの変換

標準の参照 XOP Include 添付ファイルを作成するショートカット メソッド。 詳細については、「［添付］タブ (P. 227)」を参照してください。

XML データ セットの作成

XML データ セットを生成するショートカット メソッド。典型的なユース ケースは、完全な SOAP メッセージを作成することです。次に、データ セットの最初のレコードとする XML ドキュメントのセクションを選択します。 XML データ セットを作成すると、選択した XML エレメント ツリーを使用して最初のレコードが自動的に生成されます。 新しいデータ セット プロパティは、エディタに値として設定されます。

テキスト ノードを非表示

デフォルトでは、DevTest は通常、冗長なテキスト ノード（空白など）を非表示にします。 ただし、XML エレメントがさまざまな組み合わせのエレメントおよびテキストをサポートする混合タイプ エレメントの場合など、それらを表示することが役立つ場合もあります。

ネームスペース ノードを非表示

デフォルトでは、DevTest はネームスペース宣言およびネームスペース プレフィックス宣言を非表示にします。 プレフィックス値を確認する場合やプレフィクスまたはネームスペースのスコープを変更する場合は、表示できます。

検証

XML を検証し、その結果をウィンドウの下部の［検証結果］ペインに表示します。

XML の検証結果の表示/非表示を切り替えるには、［検証結果］をクリックします。

また、キーボード ショートカットを一部のタスクに使用することもできます。

- Windows で選択したエレメントまたは属性を削除するには、Ctrl + BackSpace キーを押します。
- Windows で選択したノードを上に移動させるには、Ctrl + 上方向キーを押します。
- Windows で選択したノードを下に移動させるには、Ctrl + 下方向キーを押します。

属性を右クリックすると、一般的な右クリック メニューのいくつかの機能を持ったメニューが表示されます。

［タイプ］フィールドの編集

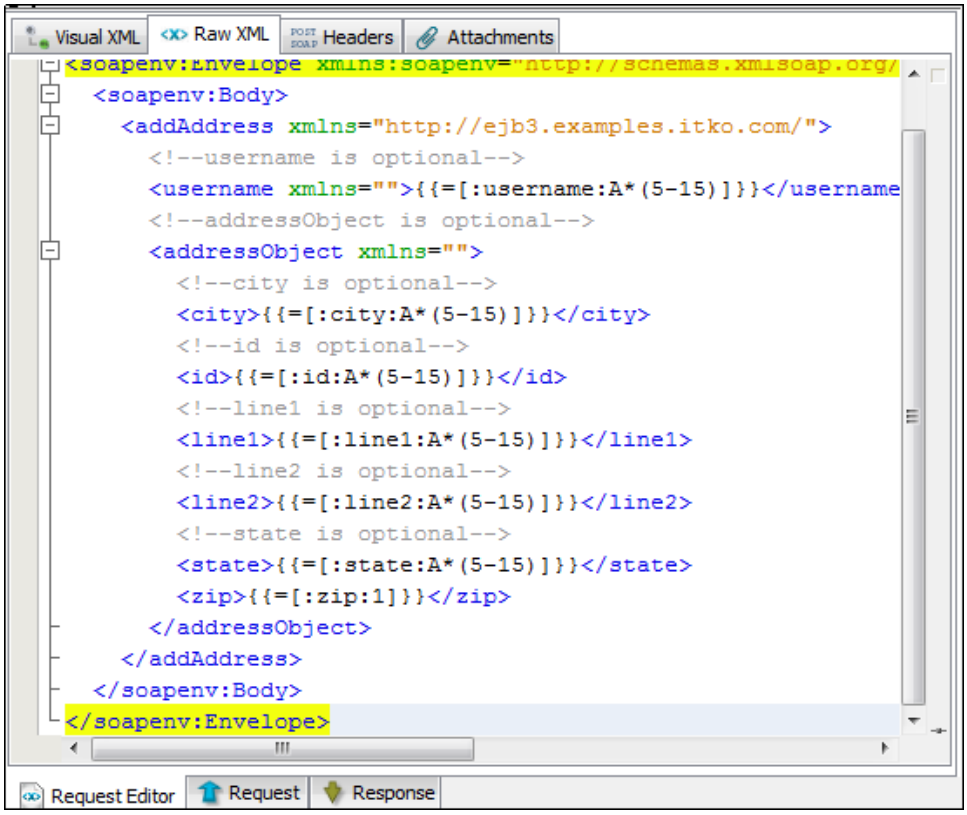
［タイプ］列には、XML スキーマ タイプ（ローカル名）が表示されます。この列には、ネームスペースを使用して完全修飾名（qName）を表示するツールヒントがあります。この列は、XSD の派生型に対して編集できます。派生型を持った基本型のみを編集できます。

列が編集可能な場合、使用可能な派生型および基本型のリストがコンボボックスに表示されます。 型を選択できます。関連するエレメントは、選択した型と関連付けられます。

型を変更すると、エレメントからすべての子のエレメントおよび属性が削除され、エレメントの nil 属性が nil=true に設定されます。

[RAW XML]タブ

RAW XML エディタは、XML 対応で、RAW XML SOAP メッセージを手動で編集できるテキスト ベースのエディタです。 ビジュアル XML エディタ (P. 219)に切り替えた場合（およびその逆の場合）、行なわれたすべての変更が表示されます。



RAW XML エディタには、RAW XML を整形するための右クリック メニューがあります。

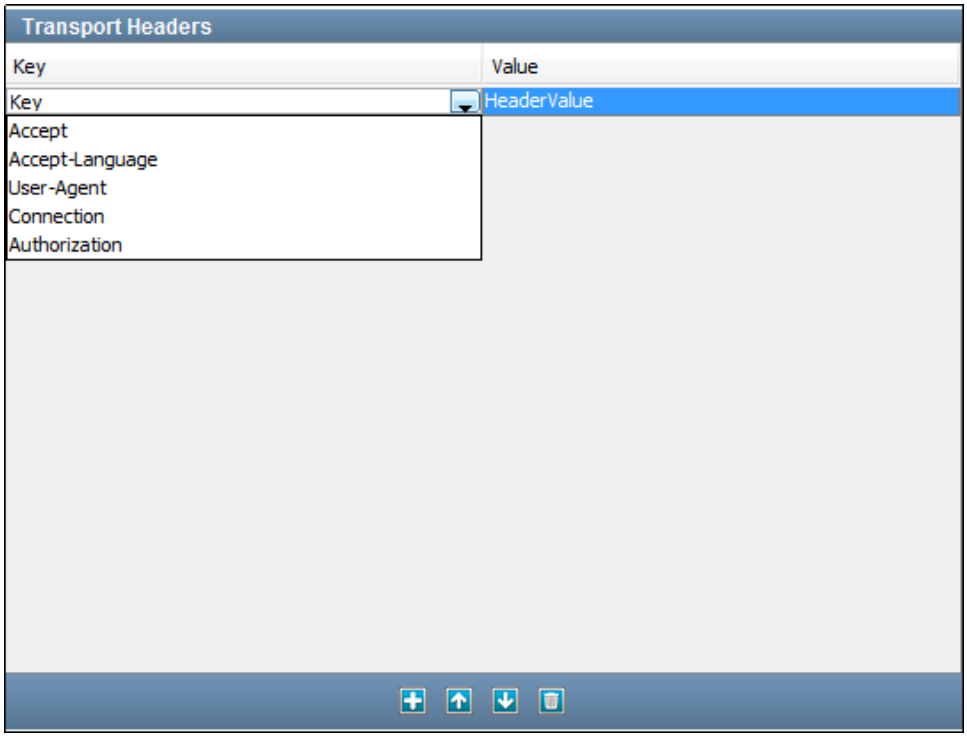
注: ［空白の削除］オプションは、ドキュメント内の各行の先頭および末尾から空白およびすべての ASCII 制御文字を削除します。

編集したドキュメントが無効な XML の場合、ビジュアル XML エディタはエラー メッセージを表示する場合があります。

RAW XML エディタで変更を修正すると、ビジュアル XML エディタは再び動作します。

[ヘッダ]タブ

［ヘッダ］タブでは、SOAP メッセージで転送されるヘッダ（HTTP ヘッダや JMS プロパティなど）を挿入できます。



ヘッダ行を追加してドロップダウン リストからヘッダを選択するには、プラス記号をクリックします。

Accept

Accept 要求ヘッダ フィールドは、応答で許可するメディア タイプを指定するために使用されます。 Accept ヘッダは、要求のタイプのセットを特に限定して指定するために使用します。 たとえば、インライン イメージを要求する場合などです。

このフィールドには、この要求に対する応答で許可される表現スキームのセミコロン区切りリストが含まれます。

```
Accept          = "Accept" ":"
                  #( media-range [ accept-params ] )
```

Accept - Language

Accept - Language ヘッダ フィールドは Accept に似ていますが、応答に希望する言語の値をリストします。 指定していない言語での応答は不正ではありません。

```
Accept-Language = "Accept-Language" ":"

 1#( language-range [ ";" "q" "=" qvalue ] )

   language-range  = ( ( 1*8ALPHA *( "-" 1*8ALPHA ) ) | "*" )
```

User - Agent

User-Agent 要求ヘッダ フィールドには、要求の送信元のユーザ エージェントに関する情報が含まれます。

［ヘッダ］タブは、統計、プロトコル違反の追跡、および特定のユーザ エージェントの制限を回避するように応答を調整するためのユーザ エージェントの自動認識に使用します。 ユーザ エージェントは、このフィールドを要求に含める必要があります。

規則では、アプリケーションを識別するために、製品トークンが重要度の順にリストされます。

```
User-Agent     = "User-Agent" ":" 1*( product | comment
```

Connection 汎用ヘッダ フィールドにより、送信者は、その特定の接続で適切なオプションを指定できます。 また、その後の接続でプロキシによって通信 *しない*ことを指定できます。

```
Connection = "Connection" ":" 1#(connection-token)
connection-token  = token
```

認証

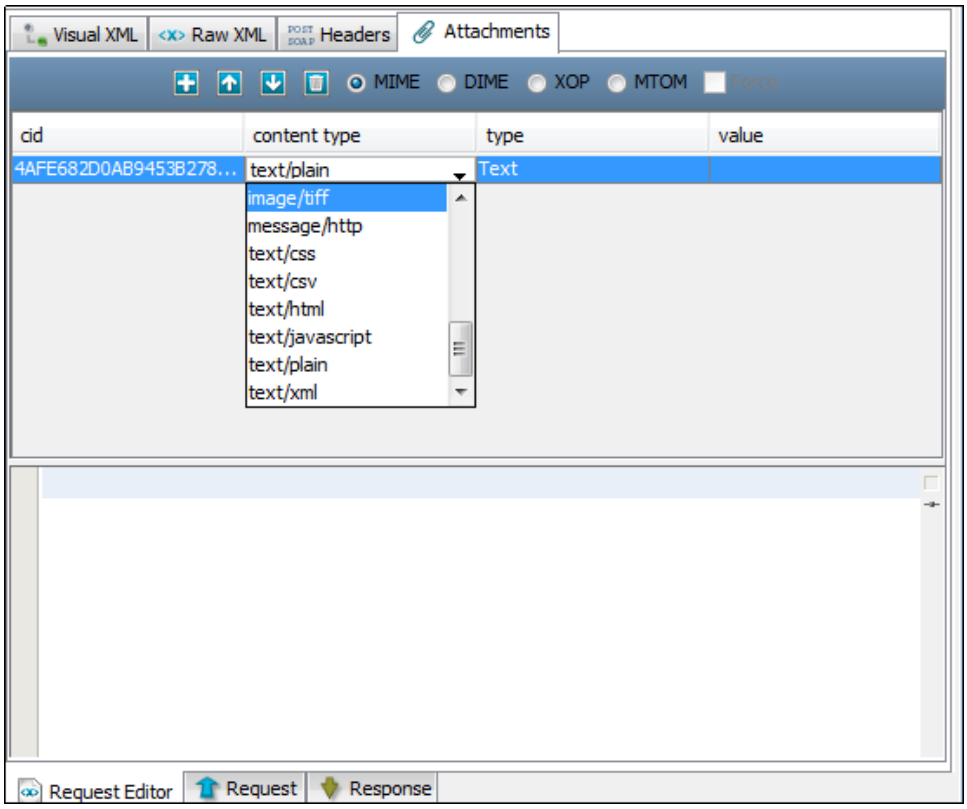
ユーザ エージェント自体をサーバで認証する（必須ではないが、通常、401 応答を受信した後に）には、ユーザ エージェントが、要求に Authorization 要求ヘッダ フィールドを含めます。Authorization フィールド値は、要求されているリソース領域に対するユーザ エージェントの認証情報から構成されます。

```
Authorization  = "Authorization" ":" credential
```

DevTest は、［ヘッダ］タブでのプロパティの使用をサポートしていません。UI は設計時にユーザ名とパスワードを取得し、Authorization ヘッダの値を計算します。プロパティを使用するには、local.properties または設定ファイルのいずれかで **lisa.http.user** および **lisa.http.pass** のプロパティを設定します。DevTest は、実行時にそれらのプロパティを使用します。

［添付］タブ

［添付］タブでは、添付ファイル データを編集できます。



参照添付ファイル

*参照添付ファイル*は、SOAP メッセージで参照されます。 VXE で参照添付ファイルを使用するには、参照添付ファイルにするエレメントを右クリックし、適切なアクションを選択します。

- ［添付ファイルに変換］（MIME および DIME）
- ［XOP 添付ファイルに変換］（MTOM/XOP Include 形式）

この操作は必要なエレメントおよび属性を自動的に作成し、添付ファイルとエレメントを一致させるために使用されるコンテンツ ID を設定します。その後、以下の手順に従います。

- ［添付］タブに切り替えます。
- コンテンツ ID を持った新しい添付ファイルを生成します。

- デフォルトのコンテンツ タイプおよび添付ファイル タイプを選択します。
- VXE の任意の既存のエレメント データを使用して値を入力します。

参照されていない添付ファイル

参照されていない（匿名）添付ファイルを使用する場合は、［添付］タブを使用して添付ファイルを手動で追加します。

［追加］、［上へ］、［下へ］、［削除］アイコンを使用して、テーブルで添付ファイルを追加、削除、再配置します。

MIME DIME XOP MTOM

このフィールドは、MIME、DIME、XOP、または MTOM 標準を使用して、添付ファイルの送信方法を制御します。 XOP は、SOAP バージョンに基づくさまざまなコンテンツ ヘッダを送信します。

MTOM を選択した場合、XOP 標準を使用して、base64binary スキーマ タイプが自動的に最適化されます。 添付ファイルを手動で追加する必要はありません。 エレメントが添付ファイルとしてすでに設定されている場合、操作は必要ありません。 手動で追加した追加の添付ファイルも送信されます。

［強制］を選択した場合（ドキュメントに base64binary エレメントが含まれていなくても）そのドキュメントは添付ファイル（Microsoft MTOM 方式）として整形および送信されます。

自動的にエレメントを最適化する場合の制限は、VXE が base64binary スキーマ タイプ（またはその拡張/制限）として エレメントを解釈する必要があることです。 データ セットまたはプロパティを使用する場合、拡張されたプロパティまたは最初のデータ セット エントリには、最適化するすべてのエレメントが含まれている必要があります。 最適化する必要があるエレメントが VXE に表示されていない場合、そのエレメントは最適化されません。

cid

エレメントと添付ファイル データをリンクさせるために SOAP メッセージの **href** 属性で使用できるコンテンツ ID。

コンテンツ タイプ

添付ファイル データを処理するためにサーバで使用される MIME エンコーディング方式。

タイプ/値

値データを編集および解釈する方法を決定する DevTest 添付ファイルのタイプ。 各タイプには専用のエディタがあります。

タイプ エディタ

XML

添付ファイル値を編集するための XML 対応テキスト エディタ。

テキスト

添付ファイル値を編集するためのテキスト ベースのエディタ。

Base64 エンコード

添付ファイル値を編集するためのテキスト ベースのエディタ、およびデコードされたバイナリ データを表示するバイト ビューア。

16 進エンコード

添付ファイル値を編集するためのテキスト ベースのエディタ、およびデコードされたバイナリ データを表示するバイト ビューア。

URL/テキスト

添付ファイル値を編集するための URL フィールド、および URL からのデータ ロードの結果を表示するテキスト データ ビューア。

URL/XML

添付ファイル値を編集するための URL フィールド、および URL からのデータ ロードの結果を表示する XML 対応テキスト データ ビューア。

URL/バイナリ

添付ファイル値を編集するための URL フィールド、および URL からのデータ ロードの結果を表示するバイナリ データ ビューア。

プロパティ

添付ファイル値を編集するためのプロパティ フィールド。 プロパティ値が文字列の場合は、添付ファイルをテキストとして送信します。それ以外の場合は、バイナリ データとして送信します。

プロパティ/URL

添付ファイル値を編集するためのプロパティ フィールド。 プロパティ値は URL であると想定されます。URL コンテンツは、添付ファイル データとしてロードおよび送信されます。

[アドレシング]タブ

［アドレシング］タブについては、「詳細設定 (P. 240)」を参照してください。

[セキュリティ]タブ

［セキュリティ］タブの詳細については、「詳細設定 (P. 240)」を参照してください。

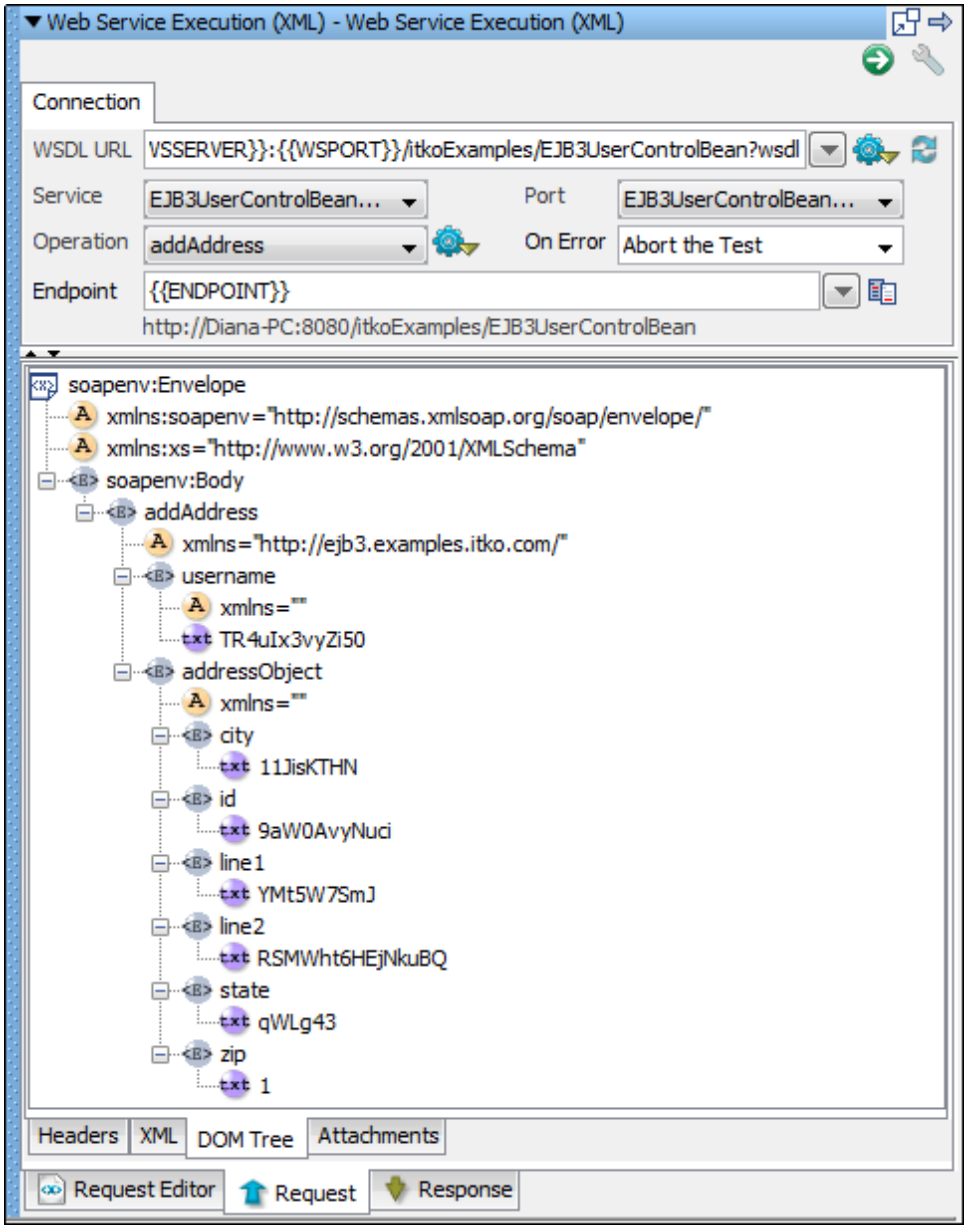
設計時の実行

接続情報の設定と SOAP 要求メッセージの作成を完了したので、設計時にステップを実行してテストできます。

右上隅の［実行］  をクリックして、Web サービス操作を実行します。実行後、［要求］および［応答］タブが入力され、［応答］タブに自動的に切り替わります。

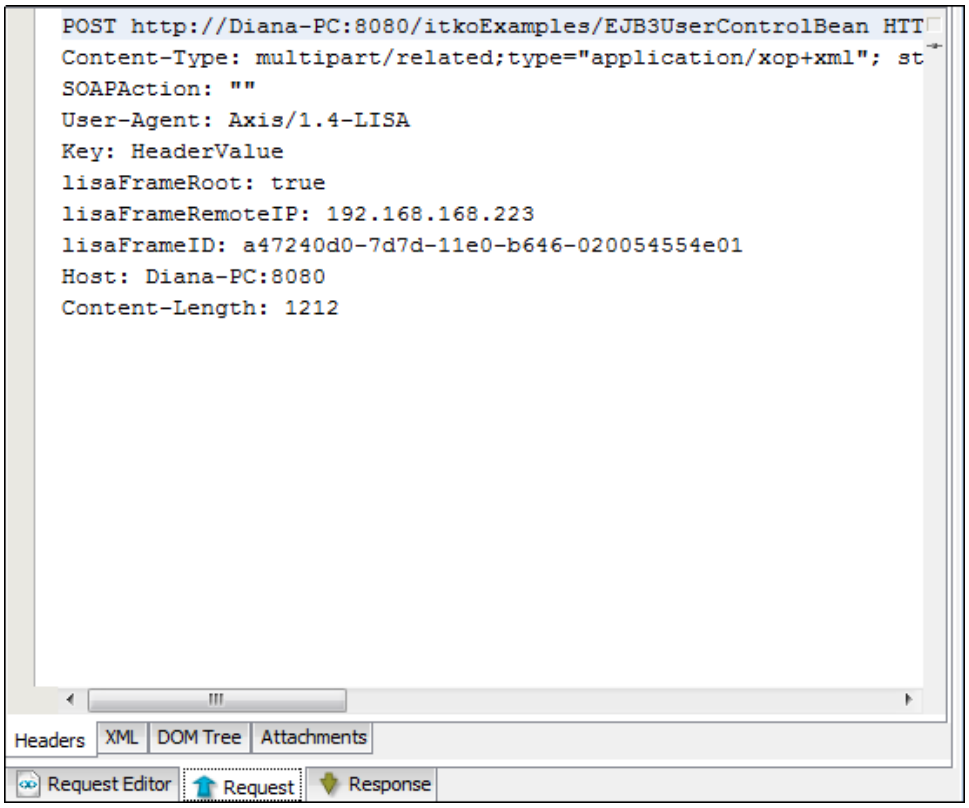
[Request]タブ

［要求］タブには、何らかの後処理（DevTest プロパティの置換など）の後に送信された結果としての要求データが表示されます。 メッセージに添付ファイルが含まれている場合、RAW MIME または DIME でエンコードされたメッセージではなく、処理されたメッセージおよび添付ファイルが表示されます。RAW メッセージを表示するには、TCPMon などのツールを使用します。



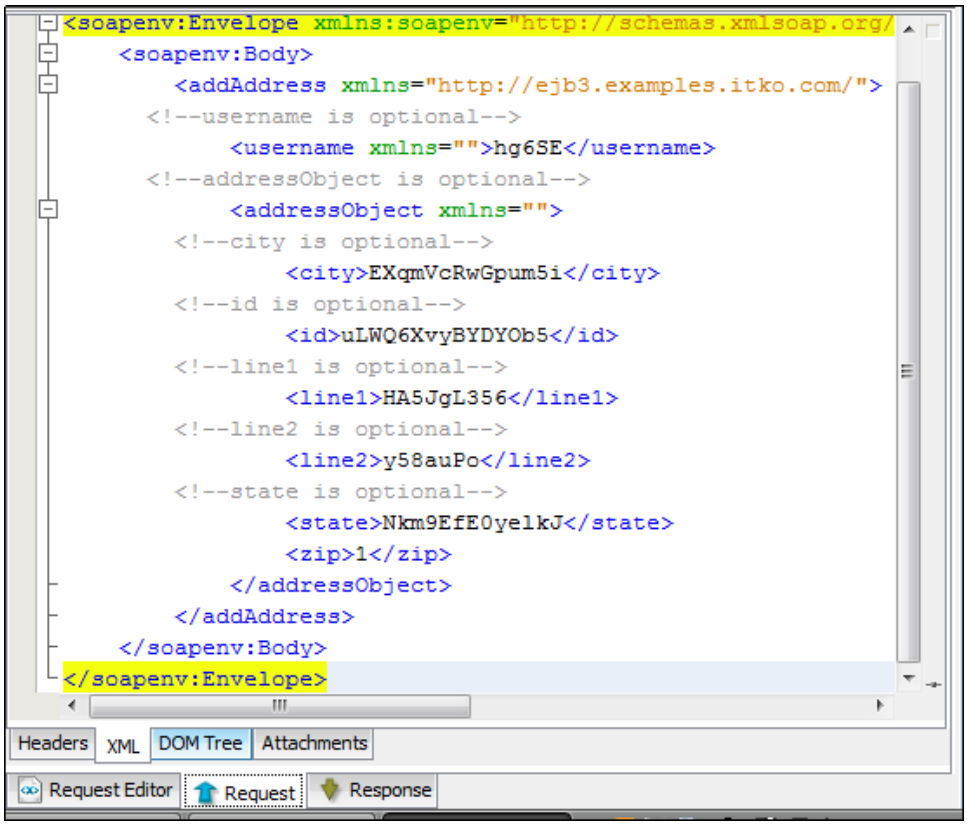
［ヘッダ］タブ

［ヘッダ］タブには、要求で送信されたトランスポート ヘッダが表示されます。



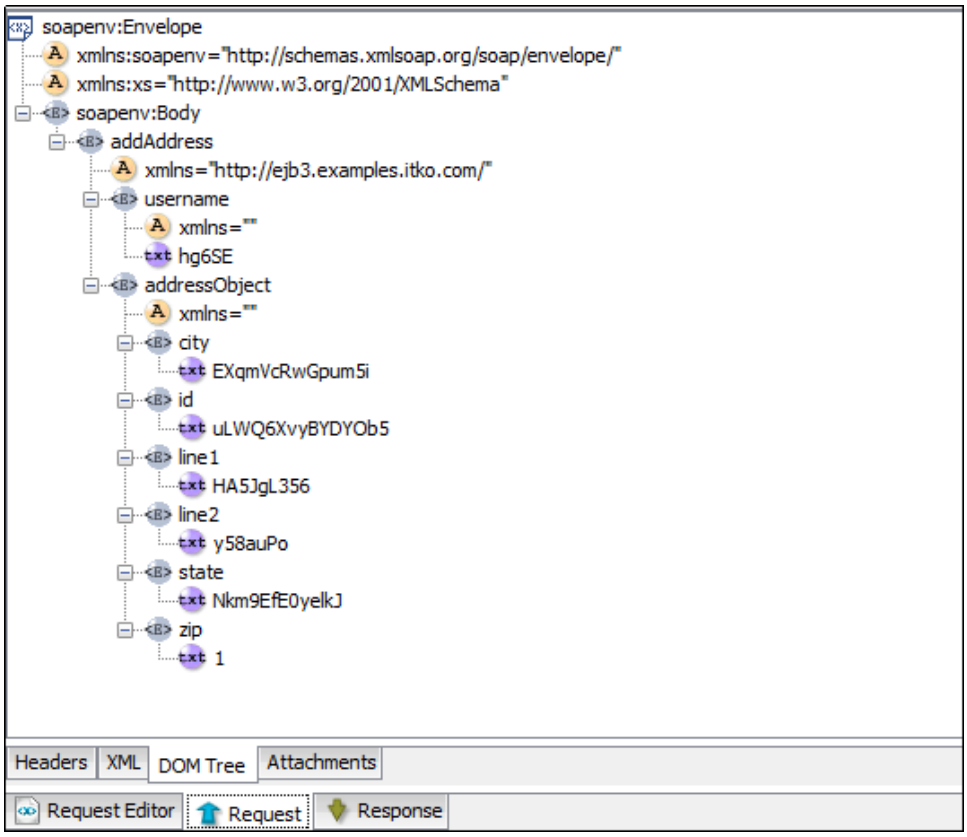
［XML］タブ

［XML］タブには、送信された RAW SOAP メッセージが高度な処理後に表示されます。



［DOM ツリー］タブ

［DOM ツリー］タブには、SOAP メッセージの DOM ツリーが表示されます。

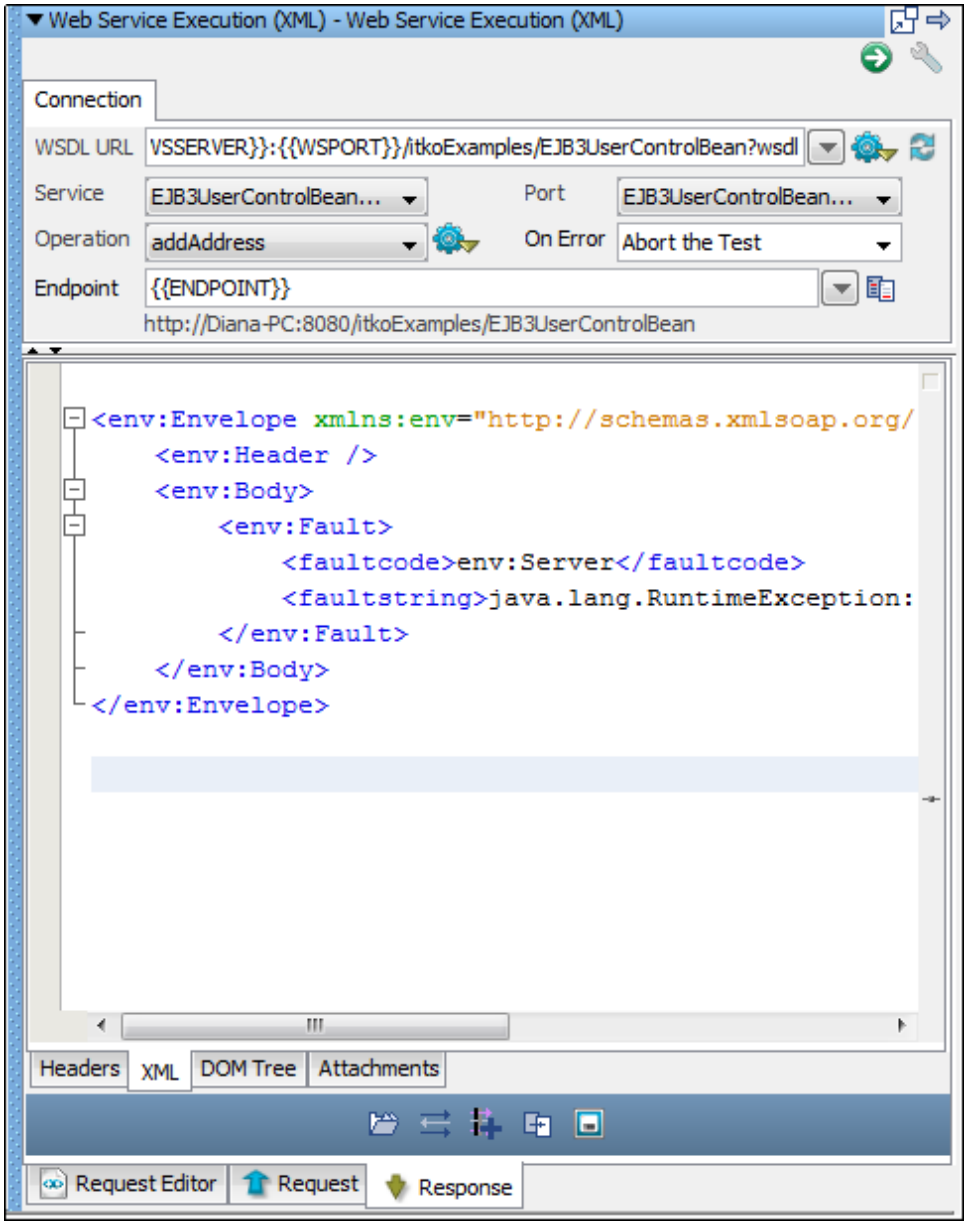


［添付］**タブ**

［添付］タブには、送信されたすべての添付ファイルが表示されます。

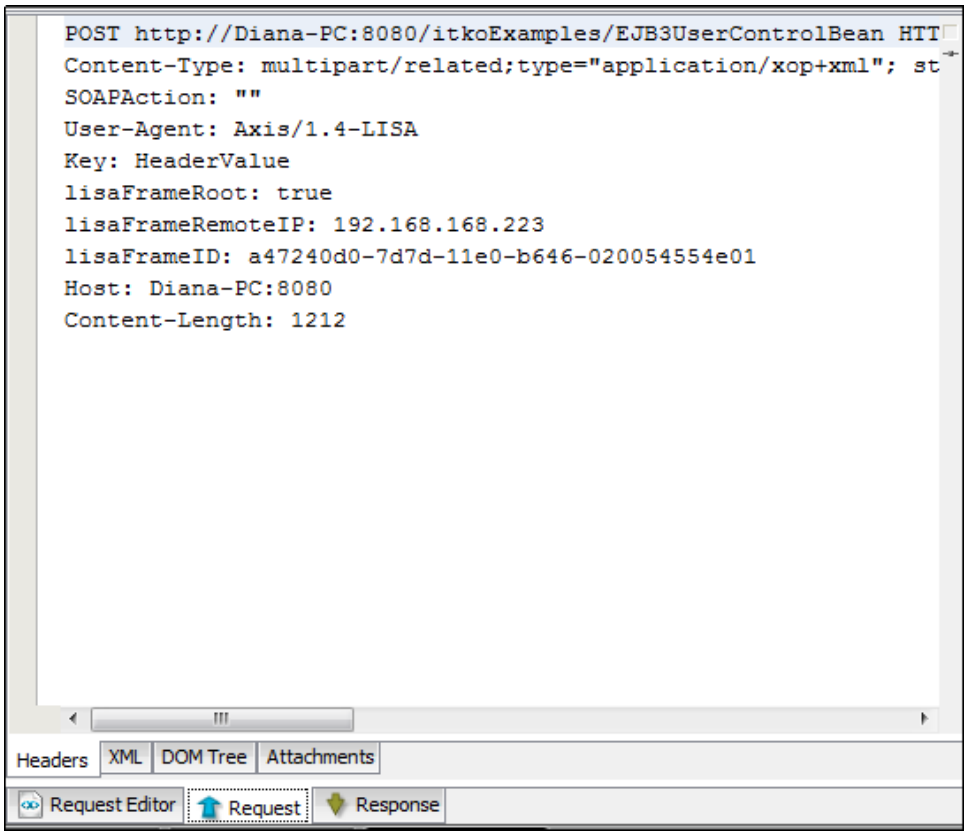
[Response]タブ

［応答］タブには、結果として受信した応答データが表示されます。メッセージに添付ファイルが含まれている場合、RAW MIME または DIME でエンコードされたメッセージではなく、処理されたメッセージおよび添付ファイルが表示されます。RAW メッセージを表示するには、TCPMon などのツールを使用します。 高度な後処理オプションが設定されている場合（以下のウィンドウを参照）、SOAP 応答メッセージは処理されてから表示されます。



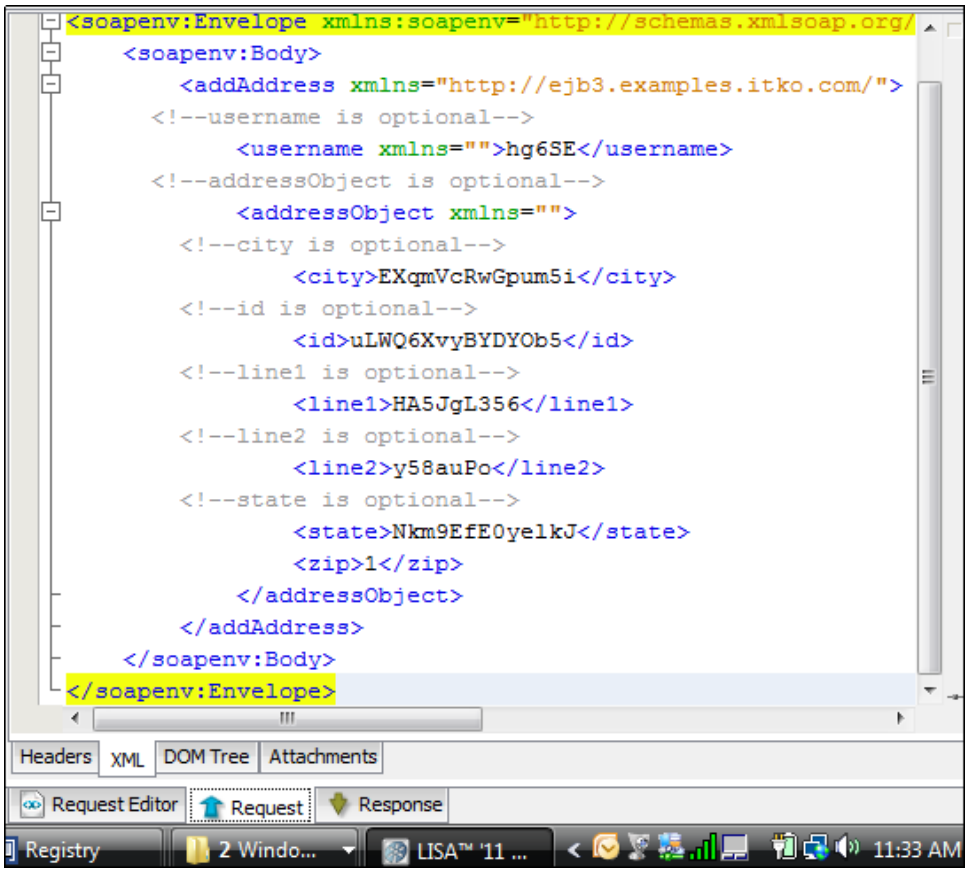
［ヘッダ］タブ

［ヘッダ］タブには、応答で受信したトランスポート ヘッダが表示されます。



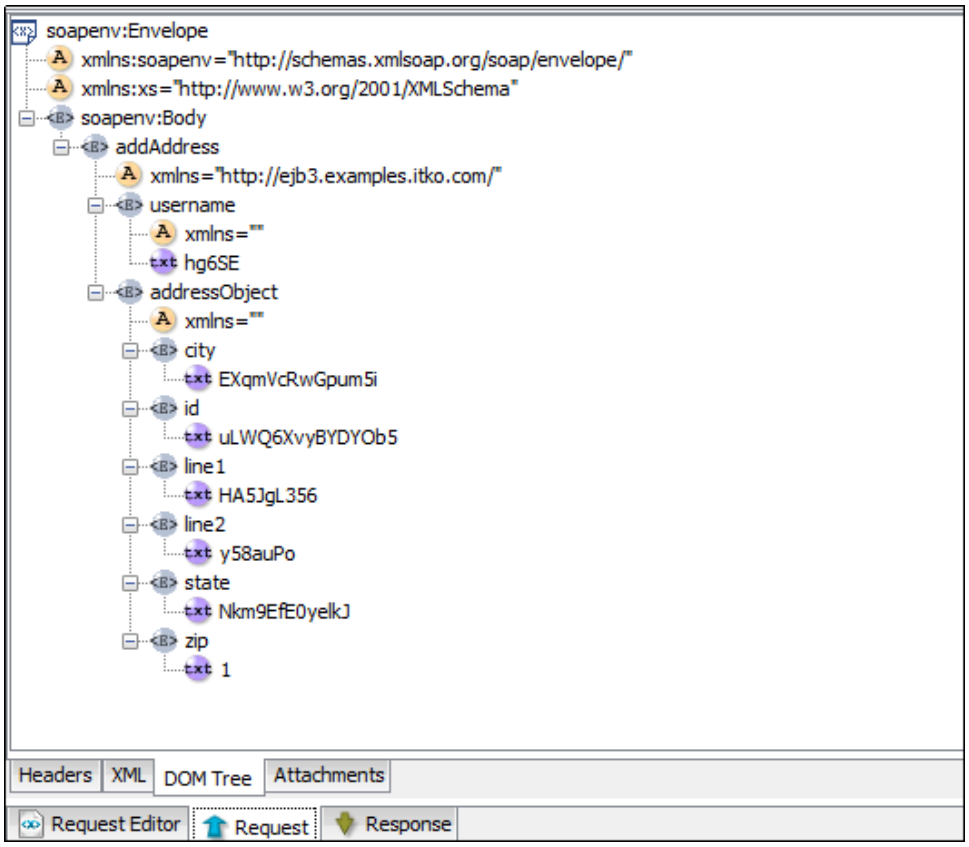
［XML］タブ

［XML］タブには、受信した RAW SOAP メッセージが高度な処理後に表示されます。



［DOM ツリー］タブ

［DOM ツリー］タブには、SOAP メッセージの DOM ツリーが表示されます。 このタブから、結果の SOAP メッセージに対してフィルタおよびアサーションを迅速に追加できます。



［添付］**タブ**

［添付］タブには、受信したすべての添付ファイルが表示されます。 受信した添付ファイルには、自動生成された DevTest プロパティを使用してアクセスできます（後でステップ、フィルタ、またはアサーションで使用するため）。 各添付ファイルに対して、以下のプロパティが設定されます。

lisa.<step name>.rsp.attachment.<cid>

添付ファイル値（バイトまたは文字列）

lisa.<step name>.rsp.attachment.contenttype.<cid>

コンテンツ タイプ（MIME タイプ： text/plain など）

lisa.<step name>.rsp.attachment.<index>

添付ファイル値（バイトまたは文字列）

lisa.<step name>.rsp.attachment.contenttype.<index>

コンテンツ タイプ（MIME タイプ： text/plain など）

lisa.<step name>.rsp.attachment.contentid.<index>

コンテンツ ID （cid）

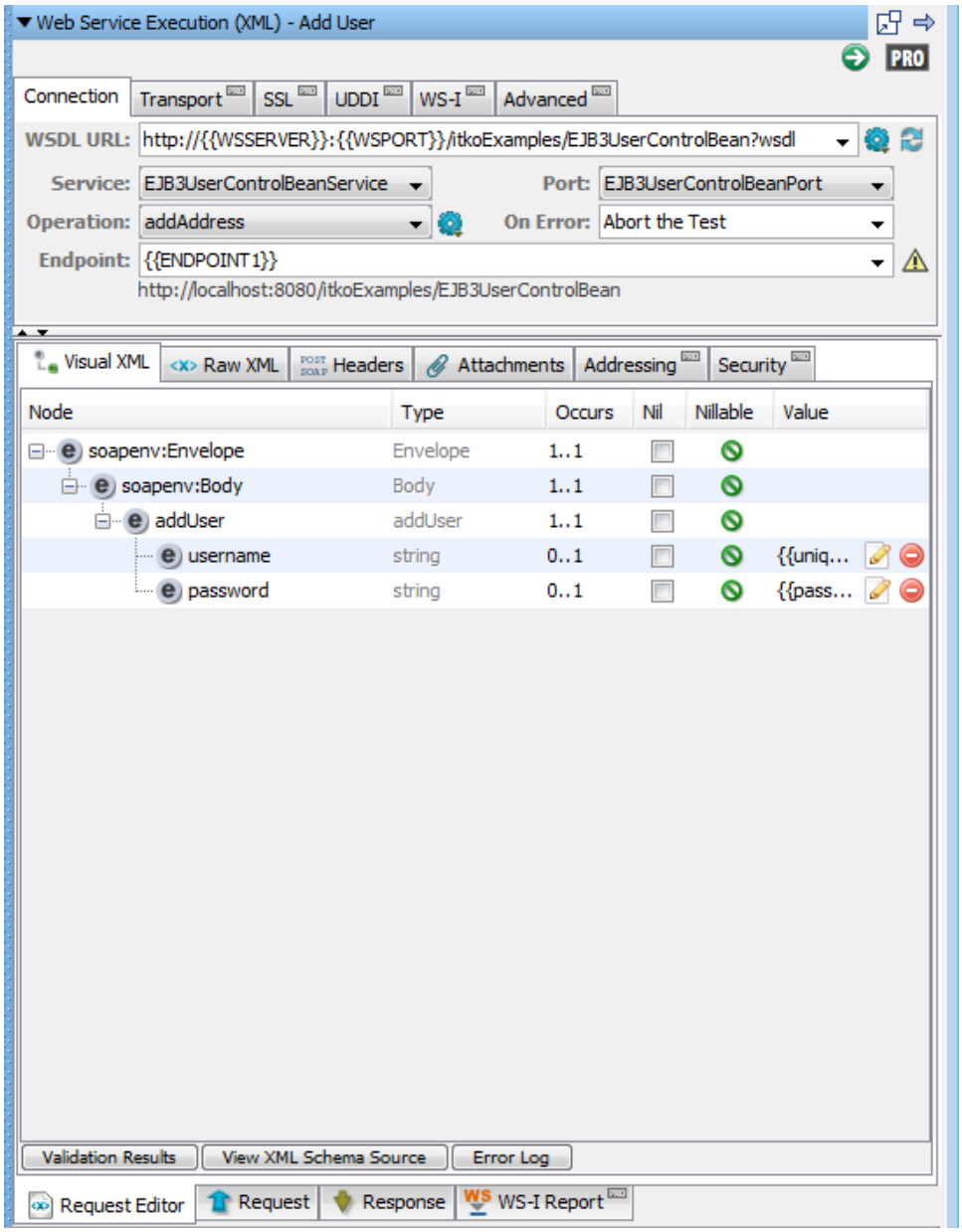
パラメータ **step name** は実行されているステップ名です。

パラメータ **cid** は SOAP メッセージで通常参照されるコンテンツ ID です。

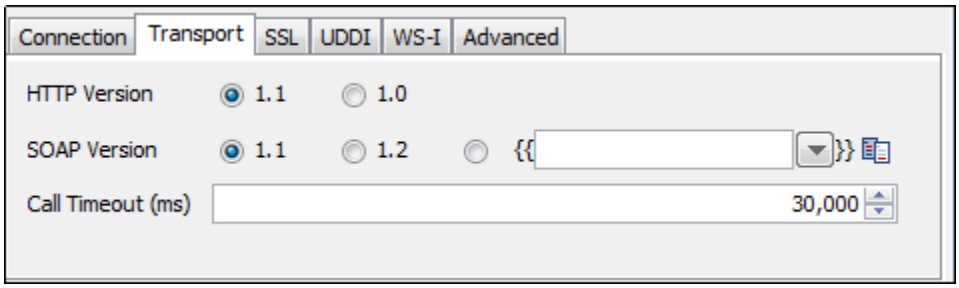
パラメータ **index** は 0 から開始され、応答の添付ファイル リスト内の添付ファイルごとに増加されます。

詳細設定

［PRO］ PRO アイコンをクリックすると、詳細設定タブが開きます。 パネルの上部には 5 つの新しいタブが表示され、下部には 2 つの新しいタブが表示されます。



[トランスポート]タブ



HTTP バージョン

このパラメータは、操作要求を送信するときにどの HTTP プロトコルが使用されるかを制御します。 デフォルトは 1.1 です。

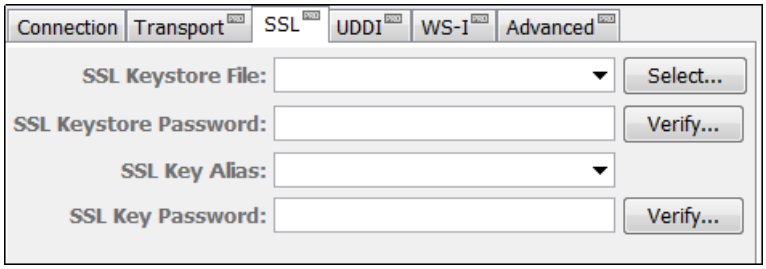
SOAP バージョン

SOAP バージョンは WSDL 定義に基づいて自動入力されます。 [SOAP バージョン] は、多くのトランスポート ヘッダ (SOAPAction や contentType など) の生成を制御します。

コール タイムアウト (ミリ秒)

このパラメータは、操作を実行しようとするときに待機する方法を定義します。 タイムアウトに達すると、例外がスローされ、エラー時の処理が発生します。

[SSL]タブ



SSL キーストア ファイル

クライアント アイデンティティ証明書が格納されるキーストア ファイルの名前。 このファイルは JKS または PKCS 形式です。

SSL キーストア パスワード

キーストア ファイルのパスワード。

SSL キー エイリアス

サーバの秘密鍵を格納および取得するために使用されるエイリアスを定義するキーストア属性。

SSL キー パスワード

JKS キーストアを使用する場合に、キーにキーストアとは異なるパスワードがあるときのキー エントリ用のオプションのパスワード。

複数の SSL 証明書を使用する例については、「HTTP/HTML 要求ステップ (P. 204)」を参照してください。

local.properties ファイルで SSL のグローバル証明書プロパティを指定します。

グローバル証明書の場合（Web サーバ、RAW SOAP、および Web サービス ステップ）

ssl.client.cert.path

キーストアへのフル パス。

ssl.client.cert.pass

キーストアのパスワード（DevTest が実行されると、このパスワードは自動的に暗号化されます）。

ssl.client.key.pass

JKS キーストアを使用する場合に、キーにキーストアとは異なるパスワードがあるときのキー エントリ用のオプションのパスワード DevTest が実行されると、このパスワードは AES （Advanced Encryption Standard）を使用して自動的に暗号化されます。

注: このオプションは、WS テスト ステップでは使用できません。 このオプションを設定するには、**local.properties** ファイルを使用します。

Web サービス ステップのみの証明書の場合（RAW SOAP ステップ以外）

ws.ssl.client.cert.path

キーストアへのフル パス。

ws.ssl.client.cert.pass

キーストアのパスワード。 DevTest が実行されると、このパスワードは自動的に暗号化されます。

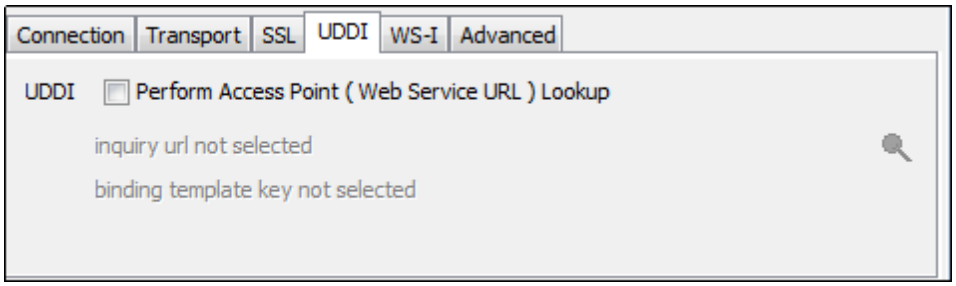
ws.ssl.client.key.pass

JKS キーストアを使用する場合に、キーにキーストアとは異なるパスワードがあるときのキー エントリ用のオプションのパスワード DevTest が実行されると、このパスワードは AES （Advanced Encryption Standard）を使用して自動的に暗号化されます。

注: このオプションは、WS テスト ステップでは使用できません。 このオプションを設定するには、**local.properties** ファイルを使用します。

注: **local.properties** と［全般］タブで重複している値がある場合、［全般］タブの値が使用されます。

[UDDI]タブ

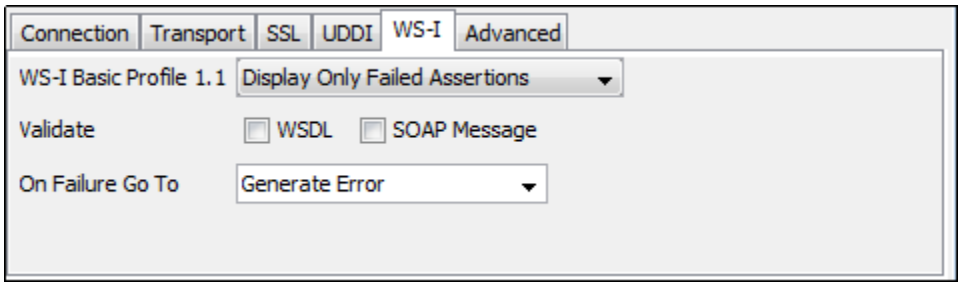


アクセス ポイント(Web サービス URL)ルックアップの実行

照会 URL およびバインディング テンプレートを選択します。 ルックアップを実行するには、［UDDI サーバの検索］ボタンを使用して正しいバインディングに移動します。

注: ステップを作成する場合、Web サービスの WSDL URL を指定するときに UDDI 検索機能を使用すると、これらの値が自動的に入力されます。照会 URL が指定されていて、バインディング テンプレートが指定されていない場合、モデル検索を実行している可能性があります。 バインディング テンプレートを見つけるには、より高いレベルの階層で検索を実行します。 次に、TModel をドリルダウンして、特定のバインディング テンプレートを見つけます。

[WS-I]タブ



WS-I Basic Profile 1.1

このプルダウン メニューでは、4 つの異なる検証レベルを選択できます。

- すべてのアサーションを表示
- 情報以外のアサーションを表示
- 失敗したアサーションのみを表示
- 成功しなかったアサーションのみを表示

検証

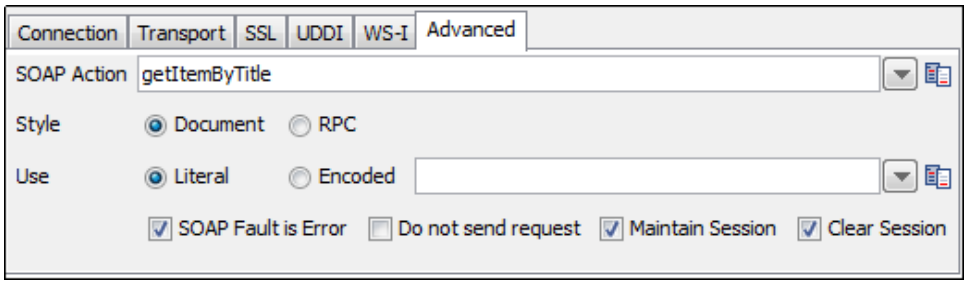
WSDL、SOAP メッセージ、またはその両方を検証するために選択します。

失敗時の移動先

エラー時にリダイレクトするステップを選択します。

注: 検証のエラーはよく発生しますが、通常、テストの結果には影響しません。 テストを完了できるよう、続行する次のステップを設定することをお勧めします。

[詳細]タブ



SOAP アクション

このフィールドは、WSDL 操作の定義に基づいて自動入力されます。このフィールドは、SOAP 1.1 要求 メッセージの SOAPAction トランスポート ヘッダの値として使用されます。 このフィールドを手動で変更する必要はほとんどありません。

スタイル

このフィールドは、WSDL 操作の定義に基づいて自動入力されます。このフィールドは、サンプル SOAP メッセージの生成方法を決定するために使用されます。 このフィールドを手動で変更する必要はほとんどありません。

使用

このフィールドは、WSDL 操作の定義に基づいて自動入力されます。このフィールドは、サンプル SOAP メッセージの生成方法を決定するために使用されます。 [エンコード] が選択されている場合、隣のフィールドのエンコード URI も編集できます。 これらのフィールドを手動で変更する必要はほとんどありません。

SOAP フォールトはエラー

SOAP フォールトが返された場合に、エラー時の処理を実行します。

要求を送信しない

選択した場合、ステップ実行は通常のすべての SOAP メッセージ処理を実行しますが、生成された SOAP メッセージを送信しません。代わりに、応答を、送信された要求メッセージとして設定します。

これは、HTTP/S 以外の転送を使用して SOAP 要求を送信するための推奨方法です。 たとえば、JMS クライアントを使用する場合、要求を作成するだけで送信しないように WS ステップを設定できます。 次に、汎用 JMS ステップを追加して実際に要求を送信し、応答を受信するようにできます。

セッションを保持

オンにすると、呼び出し全体で Cookie を保持します。

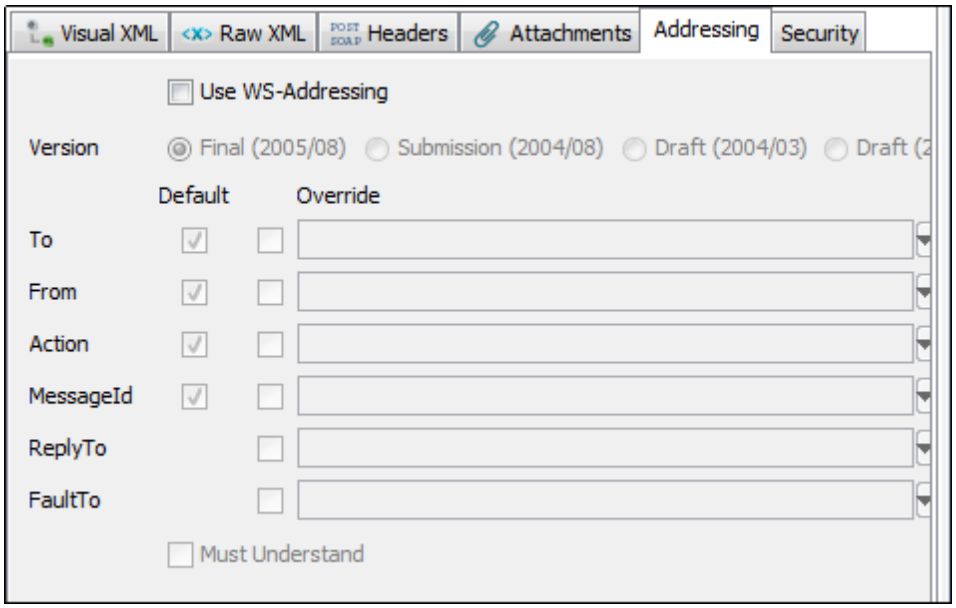
セッションをクリア

オンにすると、呼び出し全体で Cookie をクリアします。 セッション Cookie のクリアは、新しいセッションが作成されたように機能します。古いセッション Cookie は使用されず、応答の新しい Cookie もセッションに設定されませんが、後続のステップが引き続き使用できるようにセッションはクリアされません。

要求エディタ タブ

［アドレシング］タブ

要求で WS-Addressing ヘッダを送信できます。 WSDL は、ユーザがそれを設定する必要があるように、WS-Addressing 情報が必要かどうかを指定しません。



［アドレシング］タブをクリックし、以下を指定します。

WS-Addressing を使用

WS-Addressing を使用する場合はオンにします。

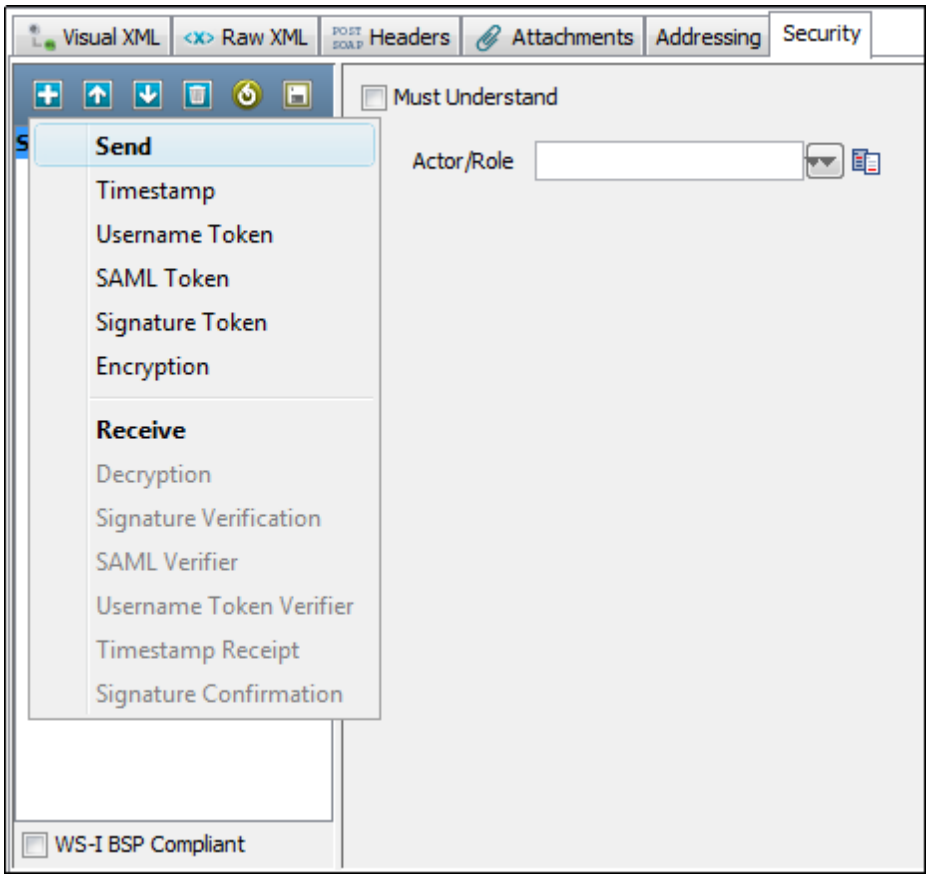
Version

適切なバージョンを選択します。 一部の Web サービス プラットフォーム（.NET など）は古いドラフト仕様を使用するため、WS-Addressing 仕様のいくつかのバージョンがオプションとしてリストされています。お使いの Web サービス プラットフォームが使用しているものを特定してください。

DevTest は入力可能な値をすべて入力します。 その後、デフォルト値を使用するか、それを上書きするかを選択できます。 エレメントの［デフォルト］チェック ボックスをオフにすることにより、デフォルト エレメントの一部を送信しないように選択できます。

Web サービスが WSAddressing ヘッダを解釈できるようにするには、［サーバによるヘッダの解釈を必須とする］チェック ボックスをオンにします。

［セキュリティ］タブ



［セキュリティ］タブをクリックし、［送信］をクリックします。

サーバによるヘッダの解釈を必須とする

サーバに WS-Security ヘッダを処理させる場合はオンにします。

アクター/ロール

必要に応じて名前を入力します。ほとんどの Web サービスは、複数のアクター/ロールを使用しません。

［追加］ をクリックし、追加するセキュリティ対策タイプを選択します。 セキュリティ対策タイプの設定パネルが表示されます。

応答へのセキュリティ検証の追加も同様です。

［セキュリティ］タブをクリックします。［追加］ をクリックし、［受信］を選択します。

- アクター/ロール名を入力します（必要な場合）。
- ［追加］ をクリックし、セキュリティ対策タイプを選択します。そのセキュリティ対策タイプの設定パネルが表示されます。

注: Web サービスの実行に必要な数のセキュリティ タイプを追加できます。

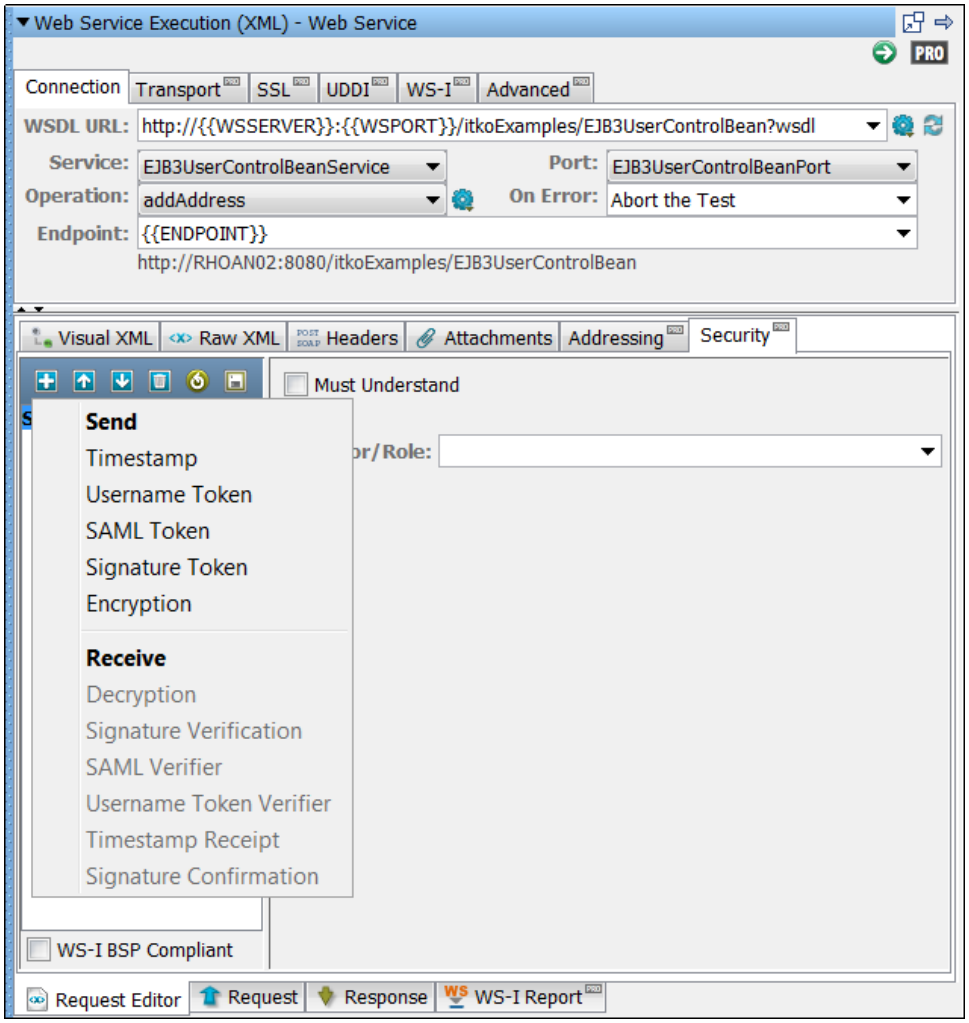
キーストアをセキュリティ対策タイプ設定で使用する場合、正しい形式、パスワード、エイリアス、およびエイリアス パスワードが使用されていることをキーストア設定で確認できます。 シグネチャ、暗号化/復号化、および SAML アサーション トークン用のエディタで［検証］ボタンをクリックすると、検証レポートを作成するキーストア検証が呼び出されます。

WS-Security 設定で必要なエイリアス名が不明な場合は、キーストア検証を使用して、キーストア内のエイリアスをすべてリストします。 ［キーストア エイリアス］および［エイリアス パスワード］ボックスを空にしたまま、［確認］をクリックします。キーストア検証については、WS-Security についてのこのセクションの最後で説明しています。

注: ［ロード］ および［保存］ アイコンを使用して、.wss ファイルのセキュリティ設定情報をロードおよび保存できます。 この機能を使用すると、同じサービスに接続する新しいステップを迅速かつ容易に作成できます。

セキュリティの例

このセクションでは、WS-security の例を実行するのに必要なパラメータについて説明します。



XML の暗号化 - 復号化

暗号化

［暗号化の使用］チェック ボックスをオンにします。

キーストア ファイル

キーストア ファイルの場所。

キーストア パスワード

キーストアのパスワードを入力します。

キーストア エイリアス

公開鍵のエイリアスを入力します。

エイリアス パスワード

空白のままにするか、または PKCS #12 ファイルのキーストア パスワードと同じにします。

キー ID タイプ

プルダウン メニューから適切なキー ID タイプを選択します。

アルゴリズム

Triple DES、AES 128、AED 192、または AES 256 を選択します。

トランスポート

PKCS#1 （RSA Encryption Standard v1.5）または RSA-OAEP （Optimal Asymmetric Encryption Padding with RSA Encryption）を選択します。

デフォルトの動作では、SOAP ボディのコンテンツのみ暗号化します。

部分のみを暗号化

暗号化するさまざまなパーツを指定する場合。 暗号化するパーツを指定するには、［選択］ボタンをクリックします。

タイプ

以下の値のいずれかを選択します。

- エレメント : エレメントおよびコンテンツを暗号化する場合に選択します。
- コンテンツ : コンテンツのみを暗号化する場合に選択します。

ネームスペース URL

エレメントの値を入力します。

エレメント

エレメントの名前を入力します。

［追加］ボタンをクリックして、目的のエレメント数までこの処理を繰り返すことができます。

ボディ エレメントを含める場合は、手動で追加します。 バイナリ セキュリティ トークンを一部として含める場合は、エレメント名として **Token** を使用します。

復号化

キーストア ファイル

キーストア ファイルの場所。

キーストア パスワード

キーストアのパスワードを入力します。

キーストア エイリアス

公開鍵のエイリアスを入力します。

エイリアス パスワード

空白のままにするか、または PKCS #12 ファイルのキーストア パスワードと同じにします。

XML シグネチャトークン - 署名検証

シグネチャ トークン

［シグネチャの追加］チェック ボックスをオンにします。

キーストア ファイル

キーストア ファイルの場所。

キーストア パスワード

キーストアのパスワードを入力します。

キーストア エイリアス

秘密鍵のエイリアスを入力します。

エイリアス パスワード

空白のままにするか、または PKCS #12 ファイルのキーストア パスワードと同じにします。

キー ID タイプ

プルダウン メニューから適切なキー ID タイプを選択します。

アルゴリズム

［DSA with SHA-1］を選択します。

ダイジェスト アルゴリズム

値： SHA-1、SHA-256、SHA-384、SHA-512、RIPEMD-160、または MD5（非推奨）。

デフォルトの動作では、SOAP ボディのコンテンツのみ署名します。

パーツのみに署名

署名するさまざまなパーツを指定する場合、署名するパーツを指定するには、［選択］ボタンをクリックします。

タイプ

値

- エレメント： エレメントおよびコンテンツを暗号化します。
- コンテンツ： コンテンツのみを暗号化します。

ネームスペース URL

エレメントの値を入力します。

エレメント

エレメントの名前を入力します。

［追加］をクリックして、このプロセスを繰り返します。

ボディ エレメントを含めるには、手動で追加します。 バイナリ セキュリティ トークンを一部として含める場合は、エレメント名として **Token** を使用します。

署名検証

署名検証を設定するために必要なパラメータは、署名に必要なパラメータのサブセットです。

キーストア ファイル

キーストア ファイルの場所。

キーストア パスワード

キーストアのパスワードを入力します。

キーストア エイリアス

公開鍵のエイリアスを入力します。

エイリアス パスワード

空白のままにするか、または PKCS #12 ファイルのキーストア パスワードと同じにします。

タイムスタンプ - タイムスタンプ受信

タイムスタンプ

［タイムスタンプの追加］チェック ボックスをオンにします。

有効期間（秒）

メッセージの有効期間を秒単位で入力します。 Expires エレメントを含めない場合は、「0」を入力します。

注: 一部の Web サービス（特に WSE 2.0 を使用する .NET 1.x/2.0）は標準のタイムスタンプ形式に準拠しておらず、ミリ秒を使用できません。 これらの Web サービスについては、［タイムスタンプでミリ秒を使用する］チェック ボックスをオフにします。

タイムスタンプ受信

タイムスタンプ受信に必要なパラメータは、タイムスタンプに必要なパラメータのスーパー セットです。 追加のパラメータは以下のとおりです。

期限切れを許可しない

期限切れタイム スタンプを許可しない場合は、このパラメータを選択します。

ユーザ名トークン - ユーザ名トークン検証

ユーザ名 トークン

［ユーザ名 トークンの追加］チェック ボックスをオンにします。

ユーザ名

ユーザ名を入力します。

パスワード

適切なパスワードを入力します。

パスワード タイプ

ドロップダウン リスト（［テキスト］、［ダイジェスト］、［なし］）からパスワード タイプを選択します。［なし］は通常、［シグネチャの追加］オプションと一緒に使用されます。

ノンスの追加

ノンスが必要な場合に選択します。リプレイ攻撃から保護するために使用します。

作成日時の追加

タイムスタンプが必要な場合に選択します。

タイム スタンプでミリ秒を使用する

ミリ秒の精度を使用するには、このチェック ボックスをオンにします。一部の Web サービス（特に WSE 2.0 を使用する .NET 1.x/2.0）は標準のタイムスタンプ形式に準拠しておらず、ミリ秒を使用できません。

シグネチャの追加

ユーザ名とパスワードの組み合わせをキーとして使用して作成されるシグネチャを追加するために選択します。

パーツのみに署名

署名するさまざまなパーツを指定する場合、署名するパーツを指定するには、［選択］ボタンをクリックします。

タイプ

以下の値のいずれかを選択します。

- エレメント：エレメントおよびコンテンツを暗号化する場合に選択します。

コンテンツ： コンテンツのみを暗号化する場合に選択します。

ネームスペース URL

エレメントの値を入力します。

エレメント

エレメントの名前を入力します。

目的のエレメント数まで繰り返し［追加］をクリックします。

ボディ エレメントを含める場合は、手動で追加します。 バイナリ セキュリティ トークンを一部として含める場合は、エレメント名として **Token** を使用します。

ユーザ名 トークン検証

［ユーザ名 トークンの検証］チェック ボックスをオンにします。

ユーザ名

ユーザ名を入力します。

パスワード

適切なパスワードを入力します。

タイム スタンプでミリ秒を使用する

ミリ秒の精度を使用するには、このチェック ボックスをオンにします。一部の Web サービス（特に WSE 2.0 を使用する .NET 1.x/2.0）は標準のタイムスタンプ形式に準拠しておらず、ミリ秒を使用できません。

シグネチャの検証

シグネチャの検証が必要な場合は、このチェック ボックスをオンにします。

SAML アサーション トークン - SAML アサーション受信

SAML アサーション トークン

［SAML トークンの追加］チェック ボックスをオンにします。

以下のいずれかのオプションを実行します。

- ［ステップ結果から］チェック ボックスをオンにします。結果が XML SAML アサーションであるステップ（SAML クエリ ステップや手動で XML を入力したテキストの解析ステップなど）を選択します。
- ［プロパティから］チェック ボックスをオンにし、XML SAML アサーションが含まれるプロパティを入力します。

SOAP 要求を送信する場合のように、［検証］をクリックして、DevTest に SAML アサーション XML を解析させ、SAML アサーション オブジェクトを構築させます。 検証は、手動で作成された SAML アサーションが有効な SAML アサーションであることを確認するために役立ちます。 また、検証は、アサーションと関連付けられたすべてのシグネチャも検証します。ただし、検証に使用する公開証明書を設定しないと、DevTest がアサーションを検証できない場合があります。

送信者がアサーションに署名（SAML アサーションのベアラ/作成者でなく、送信者が信頼性を保証）する必要がある場合は、［署名付き送信者保証］チェック ボックスをオンにします。 オンにする場合は、以下の情報が必要です。

キーストア ファイル

キーストア ファイルの場所。

キーストア パスワード

キーストアのパスワードを入力します。

キーストア エイリアス

秘密鍵のエイリアスを入力します。

エイリアス パスワード

空白のままにするか、または PKCS #12 ファイルのキーストア パスワードと同じにします。

キー ID タイプ

プルダウン メニューから適切なキー ID タイプを選択します。

アルゴリズム

［DSA with SHA-1］を選択します。

ダイジェスト アルゴリズム

値： SHA-1、SHA-256、SHA-384、SHA-512、RIPEMD-160、または MD5（非推奨）。

デフォルトの動作では、SOAP ボディのコンテンツのみ署名します。

パーツのみに署名

署名するさまざまなパーツを指定する場合、署名するパーツを指定するには、［選択］ボタンをクリックします。

タイプ

以下の値のいずれかを選択します。

- エレメント：エレメントおよびコンテンツを暗号化する場合に選択します。
- コンテンツ： コンテンツのみを暗号化する場合に選択します。

ネームスペース URL

エレメントの値を入力します。

エレメント

エレメントの名前を入力します。

このプロセスを繰り返すには、［追加］をクリックします。 ボディ エレメントを含めるには、手動で追加します。

バイナリ セキュリティ トークンを一部として含める場合は、エレメント名として **Token** を使用します。

SAML アサーション受信

応答をスキャンして SAML アサーション受信ヘッダを確認するには、［SAML アサーションの処理］チェック ボックスをオンにします。 このオプションを選択し、SAML アサーション受信ヘッダがない場合、例外が発生します。

シグネチャ確認

応答をスキャンしてシグネチャ確認ヘッダを確認するには、［シグネチャ確認］チェック ボックスをオンにします。 このオプションを選択し、確認ヘッダがない場合、例外が発生します。

Using the Keystore Verifier （キーストア検証の使用）

正しい形式、パスワード、エイリアス、およびエイリアス パスワードを使用していることを確認するためにキーストア設定を検証できます。 検証レポートを作成するには、SSL、シグネチャ、暗号化/復号化、および SAML 設定用のエディタで［検証］ボタンをクリックします。

SSL 検証は、キーストア パスワードのみ検証します。 SSL 検証は、キーストア パスワードを使用してキーストア内の少なくとも 1 つのキーをロードできることも確認します。

WS-Security 検証は、キーストアのパスワード、エイリアス、およびエイリアス パスワードを検証します。正しい検証は緑のエントリで示されます。見つかった検証エラーは赤で表示されます。 警告はオレンジで表示されます。

注: この検証では、キーストア パラメータのみが検証されます。 証明書セットの不一致やアルゴリズムの選択の誤りなど、Web サービスに関する問題が引き続き存在する可能性があります。 これらの問題は個別に検証する必要があります。

Alias Search （エイリアスの検索）

WS-Security 設定に必要なエイリアス名が不明な場合は、キーストア検証を使用します。 キーストア検証は、キーストア内のすべてのエイリアスをリスト表示します。 ［キーストア エイリアス］および［エイリアス パスワード］ボックスを空白のままにして、［検証］ボタンをクリックします。

エイリアスには青い背景があります。

［キーストア エイリアス］および［エイリアス パスワード］ボックスを空白のままにしておいたため、検証は失敗します。

WS-I レポート

オブジェクト エディタ ウィンドウの［実行］ボタンをクリックすると、検証が実行されます。（DevTest インストール ディレクトリ内の reports ディレクトリに）レポートが生成されて保存されます。 ウィンドウの下部の［WS-I レポート］タブをクリックすると、レポートを表示できます。

注: このレポートの形式は標準です。これは、Web-Services-Interoperability Organization （WS-I）によって策定されています。

WSDL 検証

WSDL 検証ステップでは、WSDL をロードし、WSDL を検証するために 1 つ以上のアサーションを追加できます。 このステップは、ユーザに静的な WSDL ファイルをロードさせ、それに対してアサーションを実行するという点で少し異なります。 ほとんどのステップでは、アサーションは応答に対して実行されます。

以下のリストでは、最も役立つアサーションについて説明します。

- **XML 差分アサーション：** WSDL を元の WSDL の制御コピーと比較して、変更されていないか確認します。
- **XML バリデータ：** スキーマまたは DTD を使用して、有効な XML を確認します。
- **WS-I Basic Profile 1.1 アサーション:** WS-I Basic Profile への準拠を確認します。

これらのアサーションについては、「アサーションの説明 (P. 9)」で説明しています。

前提条件： 上記の 3 つのアサーションに対する理解。

パラメータ要件： 検証する WSDL の場所。

WSDL 検証ステップを設定するには、［WSDL URL］フィールドに WSDL を入力し、［ロード］をクリックします。

WSDL がエディタに表示されます。 XML または DOM ビューで表示できます。

これで、アサーションを追加することができます。

Web-RAW SOAP 要求

RAW SOAP 要求ステップでは、RAW SOAP 要求（RAW XML）を送信して、Web サービスをテストできます。 WSDL がないレガシー SOAP コールまたは Web サービスをテストするために、このステップを使用できます。 また、このステップでは、正しくないタイプのデータに対する Web サービスの反応をテストできます。 たとえば、数値が必要な場合に文字列を送信することは、Web サービス実行ステップでは許可されません。 RAW SOAP 要求の別の用途は、負荷の高いテスト時のオーバーヘッドを低減することです。 要求を作成するために オブジェクトを XML にマーシャリングするため、通常の Web サービス ステップにはいくつかの追加のオーバーヘッドがあります。 このステップは、SOAP XML 応答のマーシャリングを解除してオブジェクトに戻します。 RAW SOAP ステップは、このオーバーヘッドを回避し、RAW SOAP XML のみを処理します。 処理が少ないため、より速く実行されます。

SOAP 要求をエディタに入力または貼り付けることができます。 また、要求をファイルから読み取り、DevTest プロパティを使用してパラメータ化できます。

動的な WS-Addressing または WS-Security ヘッダはサポートされていません。 これらのタイプのヘッダが必要な場合は、SOAP 要求の一部として入力領域にそれらを静的に入力します。 要求に WS-Security シグネチャ トークンなどの項目が含まれる場合、署名されたエレメントはパラメータ化できません。または、シグネチャは有効ではありません。

注: このステップは SOAP コールに制限されていません。 XML またはテキストの POST も実行できます。

RAW SOAP 要求を作成する方法

以下のパラメータを入力します。

SOAP **サーバ** URL

Web サービス エンドポイントの URL を入力します。 URL は、単に WSSERVER および PORT プロパティに置換されるのではなく、単一のプロパティに変換されます。

SOAP **アクション**

コールされているメソッドの WSDL の <soap: operation> タグに示されているように、SOAP アクションを入力します。 このアクションは SOAP 1.1 に必要です。SOAP 1.2 では、通常、空白のままにしておく必要があります。

コンテンツ タイプ

コンテンツ タイプを選択します。 SOAP 1.1 では text/HTML、SOAP 1.2 では application/SOAP+XML を使用します。

［詳細］ボタン

クリックして、すべてのカスタム HTTP ヘッダを追加します。

応答を破棄

オンにすると応答を破棄し、それを短い有効な静的 SOAP テキストに置換します。 この機能は、サイズの大きな応答の処理により負荷ジェネレータ コンピュータのスケーラビリティが制限される場合の負荷テストを対象としています。

エディタに SOAP 要求を入力または貼り付けるか、あるいは［ファイルから要求を読み取り］をクリックして SOAP 要求が含まれるファイルを参照します。

ここで、プロパティを使用して要求をパラメータ化できます。

コールを実行するには、［テスト］をクリックします。

応答を確認するには、［確認］タブをクリックします。

これで、フィルタおよびアサーションを追加することができます。

Base64 エンコーダ

Base64 エンコーダ ステップは、Base-64 エンコーディング アルゴリズムを使用してファイルをエンコードするために使用します。 結果は、テストケースの別の場所で使用するためにプロパティに格納できます。

Base64 エンコーダ ステップは、ファイルを入力として受け取り、Base64 エンコーディングを使用してそのファイルをエンコードします。 エンコードされたファイルは、プロパティに格納できます。 ファイルをエンコードするには、［ロード］をクリックします。エディタに表示される Base64 にエンコードされたテキストは、読み取り専用です。

以下のパラメータを入力します。

ファイル

フル パスおよびパス名を入力するか、またはエンコードするファイルを参照します。

プロパティ キー[オプション]

エンコードされたファイルを格納するプロパティの名前。

ロード

クリックすると、ファイルをロードしてエンコーディングをテストします。 必要に応じて、指定されたプロパティにそれを格納します。

環境エラーの場合

環境エラーが発生した場合に実行するアクション。

ファイルがエンコードされた後、フィルタおよびアサーションを追加できます。 有効なオプションは以下のとおりです。

- ランダム選択フィルタ
- 解析値フィルタ
- XML Xpath フィルタ
- HTML テーブル結果セット フィルタの作成
- 選択範囲にアサートを作成

フィルタおよびアサーションを追加した場合、［ロード］をクリックしてファイルをロードするか、［保存］ をクリックしてエディタのコンテンツを新しいファイルに保存します。

マルチパート MIME ステップ

マルチパート MIME （Multipurpose Internet Mail Extensions）ステップを使用すると、DevTest は、ファイルからデータをロードし、エンコードして、HTTP 要求の POST パラメータとして使用されるプロパティに保存することができます。エンコードされたドキュメントは、HTTP/HTML 要求ステップで事前に定義されているプロパティに格納されます。

マルチパート MIME 形式のサブミット要求が記録された場合、アップロードされたファイルのコンテンツが記録されます。 後の再生では、フォームの「file upload」の部分と同じコンテンツが再度サブミットされます。マルチパート MIME ステップは、テスト ケースが再生される場合に、どのファイルがアップロードされるかを変更するために使用します。

前提条件： HTTP パラメータが含まれる HTTP/HTML 要求ステップが存在する必要があります。このステップは、マルチパート MIME ステップの前にある必要があります。

以下のパラメータを入力します。

ステップ

HTTP/HTML 要求ステップの名前を選択するか、またはプルダウン メニューから、エンコードされたドキュメントが含まれるプロパティを受信するステップを選択します。

パラメータ

プルダウン メニューから選択されたプロパティ（［ステップ］フィールドで指定したステップでリスト表示される）の名前を選択します。

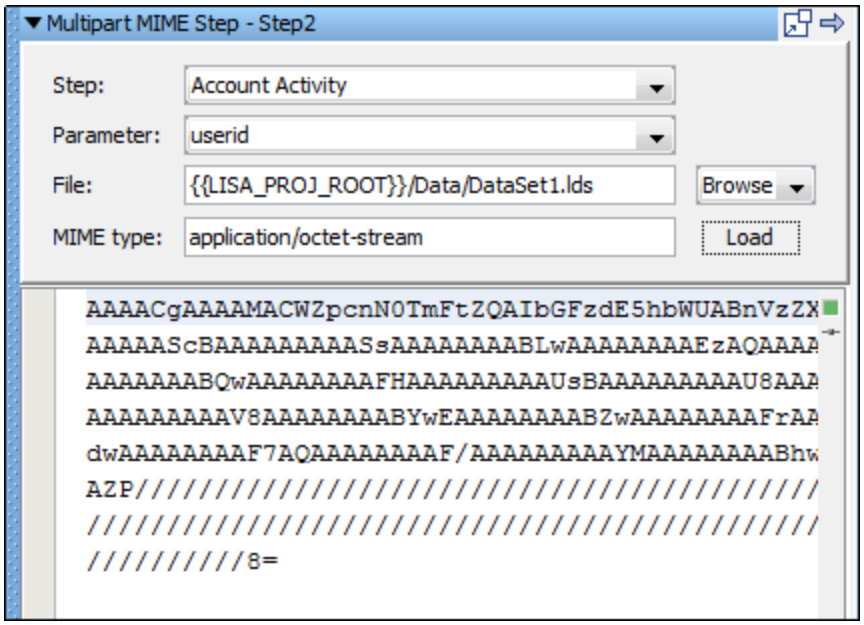
ファイル

エンコードするドキュメントのパス名を入力するか、または参照します。

MIME タイプ

サーバが想定する MIME タイプを入力します。

［ロード］をクリックして、ファイルをエンコードします。



エディタのコンテンツをファイルに保存するには、［保存］ をクリックします。

上記の図の例では、口座のアクティビティ ステップには、**DataSet1.lds** ファイルのエンコードされたバージョンを含む POST パラメータ ユーザ ID があります。

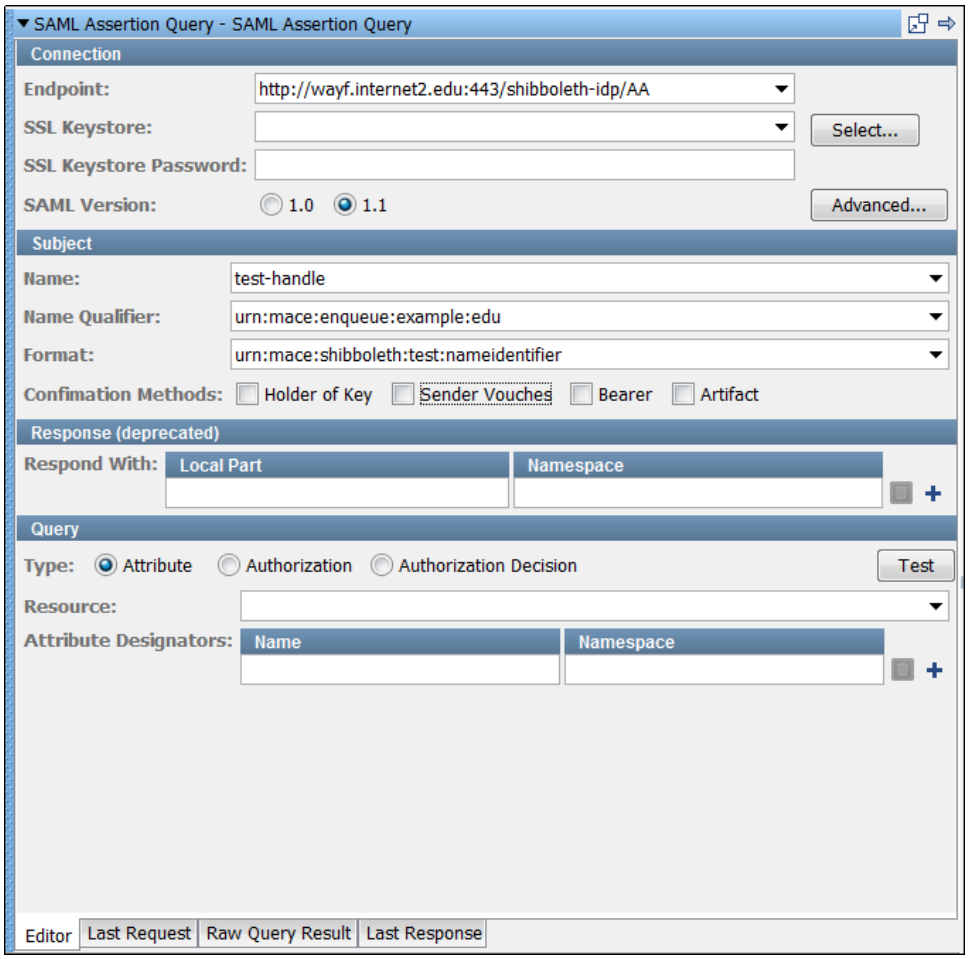
SAML アサーション クエリ

SAML アサーション クエリ ステップでは、WS-Security SAML 1.x アサーション トークンを使用する Web サービス実行ステップで後で使用される SAML アサーションを、アイデンティティ プロバイダから取得できます。

前提条件： 実行する必要がある SAML アサーション クエリのタイプについての大まかな理解。 この情報は、ID セキュリティとして SAML アサーションを使用するシステムの開発者、またはアイデンティティ プロバイダ管理者のいずれかから取得できます。

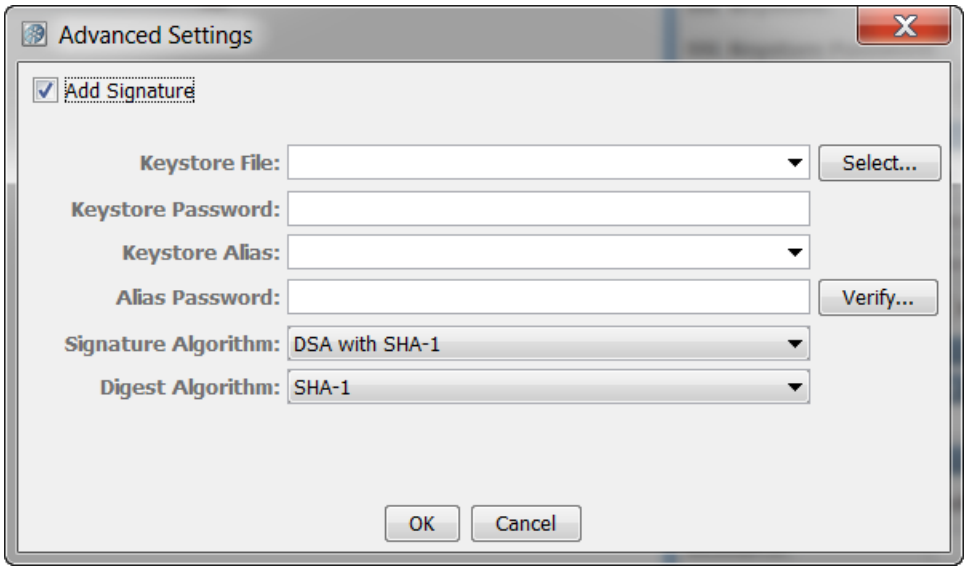
パラメータ要件： 最小要件として、以下の情報が必要です。

- ID プロバイダの SAML クエリ インターフェース（エンドポイント）の URL。
- サブジェクト情報（SAML アサーションを取得する人/もの）。
- 実行するクエリのタイプ。
- クエリのタイプに応じて、いくつかの追加情報。



SAML アサーション クエリ エディタには 4 つのタブがあります。［エディタ］タブでは、クエリ情報を設定できます。クエリを設定した後、エディタの［クエリ］セクションの［テスト］ボタンを使用して、クエリをテストできます。クエリをテストした後、［最終要求］タブで送信された RAW 要求を表示できます。［RAW クエリ結果］タブでは、RAW SOAP 応答を表示できます。 たとえば、複数のアサーションが返された場合、ここでアサーションを参照できます。 ［最終応答］タブでは、ステップ応答を表示できます。 たとえば、このタブには、Web サービスの WS-Security トークンで使用されたものが表示されます。

［詳細］ボタンを使用すると、より多くの情報を入力できます。



［シグネチャの追加］チェック ボックスをオンにします。

キーストア ファイル

キーストア ファイルの場所。

キーストア パスワード

キーストアのパスワードを入力します。

キーストア エイリアス

秘密鍵のエイリアスを入力します。

エイリアス パスワード

空白のままにするか、または PKCS #12 ファイルのキーストア パスワードと同じにします。

シグネチャ アルゴリズム

［DSA with SHA-1］を選択します。

ダイジェスト アルゴリズム

［SHA-1］、［SHA-256］、［SHA-384］、［SHA-512］、［RIPEMD-160］または［MD5 （非推奨）］を選択します。

接続

この情報は、SAML クエリ API サーバの場所および接続方法を示しています。

エンドポイント

アイデンティティ プロバイダの SAML クエリ API の URL。

SSL キーストア

エンドポイントに接続するためにクライアント側の識別証明書を使用するには、［選択］ボタンを使用して、キーストア ファイルを選択します。 または、プルダウン リストから事前に入力された項目を選択するか、手動で入力します。

SSL キーストア パスワード

SSL キーストアのパスワード（使用する場合）。

SAML バージョン

アイデンティティ プロバイダにクエリを実行するために使用する SAML バージョン。

サブジェクト

この情報は、誰または何に対して SAML アサーションを要求するかを示します。 サブジェクトは、現在の認可/権限に関するアサーションを提供するユーザ、ユーザ グループ、またはその他のエンティティです。

名前

エンティティの名前（ユーザ名など）。

名前修飾子

名前を修飾するために使用するグループまたはカテゴリ（ドメインなど）。

形式

このフィールドは、名前が送信される形式を示します（たとえば、ユーザ名に対するものとしてフル ネーム）。

確認方法

クエリに含める確認方法のタイプを選択します。 クエリは、指定されたタイプの少なくとも 1 つを含むアサーションのみを返します。 タイプを選択しない場合、確認方法に関係なくアサーションを返します。

応答（非推奨）

この情報は、SAML アサーションの一部としてどのアサーション ステートメントを返すかを示します。 SAML 1.1 では非推奨です。

ローカル パート

エレメント名（AuthenticationStatement、AuthorizationDecisionStatement、AttributeStatement など）。

ネームスペース

エレメントのネームスペース（urn:oasis:names:tc:SAML:1.0:assertion など）。

［追加］ をクリックすると、返されるセットに XML エレメントを追加します。

［削除］ をクリックすると、すでに追加されているエレメントを削除します。

クエリ

実行するクエリのタイプの説明。 クエリのタイプは以下のとおりです。

- 属性
- 認証
- 認可の決定

属性

属性クエリは、属性ステートメントのセットで応答します。たとえば、それは、サブジェクトがどのグループのメンバかを示します。

リソース

クエリを特定のリソース（特定の Web サービス、ドメイン、ファイルなど）に制限するには、リソース名を指定します。

属性指示子

名前およびネームスペース（XML エレメントなど）で各属性を識別します。 返される各属性タイプを指定することにより、返される属性ステートメントのセットをフィルタできます

例：

```
Name = urn:mace:dir:attribute-def:eduPersonScopedAffiliation、
Namespace = urn:mace:shibboleth:1.0:attributeNamespace:uri)
```

認証

特定のサブジェクトの SAML アサーションに関連する認証ステートメントを要求するために使用します。

認証方法

このパラメータは、クエリを、特定の認証方法である認証ステートメントに制限します。

例：

```
urn:oasis:names:tc:SAML:1.0:am:X509-PKI、
urn:oasis:names:tc:SAML:1.0:am:PGP、
urn:oasis:names:tc:SAML:1.0:am:password
```

事前定義済みの認証方法のセットがプルダウン リストから使用可能です。

認可の決定

特定の証拠に対してサブジェクトが実行する特定のアクションの SAML アサーションを要求するために使用します。

リソース

クエリを特定のリソース（特定の Web サービス、ドメイン、ファイルなど）に制限するには、リソース名を指定します。

アクション

名前（データ）およびネームスペース（XML エレメントなど）で指定して、認可に要求して実行する少なくとも 1 つのアクション（ログイン、表示、編集など）を指定します。

証拠（アサーション）

（オプション）アイデンティティ プロバイダへのアドバイスとして認可の決定クエリで含める 1 つ以上の SAML アサーションを指定します。 SAML アサーション XML を保持するプロパティを指定します。 前のステップ（別の SAML アサーション クエリまたはテキストの解析ステップなど）の応答を使用するには、**lisa.<ステップ名>.rsp** を使用します。

証拠（参照 ID）

必要に応じて、アサーション参照 ID を指定します。

Java/J2EE ステップ

以下のステップが使用できます。

動的 Java 実行 (P. 275)
RMI サーバ実行 (P. 279)
Enterprise JavaBean 実行 (P. 283)

動的 Java 実行

動的 Java 実行ステップでは、Java オブジェクトをインスタンス化および操作できます。 DevTest クラスパスのすべての Java クラス（JRE クラスパスのクラスも含む）を使用できます。 hotDeploy ディレクトリにコピーすることにより、ユーザ クラスをクラスパスに配置できます。 テスト中のクラスは、複合オブジェクト エディタにロードされます。このエディタでは、Java コードを記述せずにクラスを操作できます。

この例では、Java の日付インスタンス（**java.util.Date** クラス）を使用します。

1. 動的 Java 実行エディタでは、以下のパラメータを入力します。

   JVM を使用

   ［ローカル］を選択します。

   注: ［リモート］オプション ボタンをクリックすると、コンテナ内テスト（ICT）を使用して、Java オブジェクトをリモートで実行できます。 ただし、このモードは、使用する前にいくつかの追加の設定を必要とします。 詳細については、「*SDK の使用*」を参照してください。

   ローカル JVM 設定

   以下のいずれかのオプションを選択します。

   - クラスの新規オブジェクトを作成： このオプション ボタンを選択し、インスタンス化する Java クラスを入力、選択、または参照します。この値は、Java クラスのパッケージを含む完全修飾クラス名（例： **com.example.MyClass**）である必要があります。
   - プロパティからロード： このオプション ボタンをクリックし、値としてシリアル化されたオブジェクトを持つプロパティの名前を入力します。

   環境エラーの場合

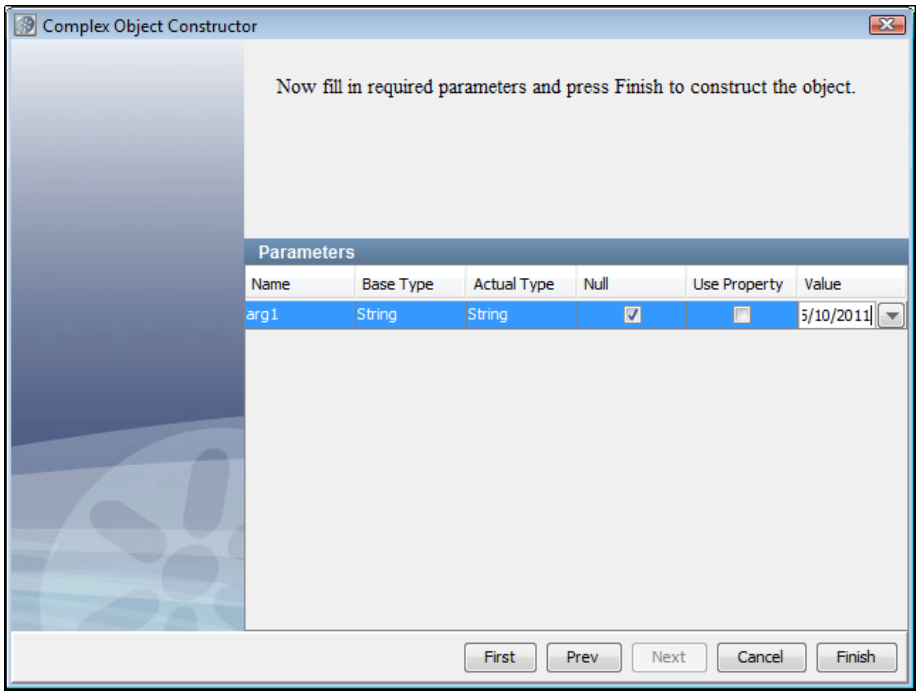
   オブジェクトの作成時に環境エラーが発生した場合にリダイレクトするステップを選択します。

   注: Java オブジェクトが個別のクラスローダによってロードされることを要求する場合は、**クラス ローダ サンドボックス コンパニオン**を追加します。

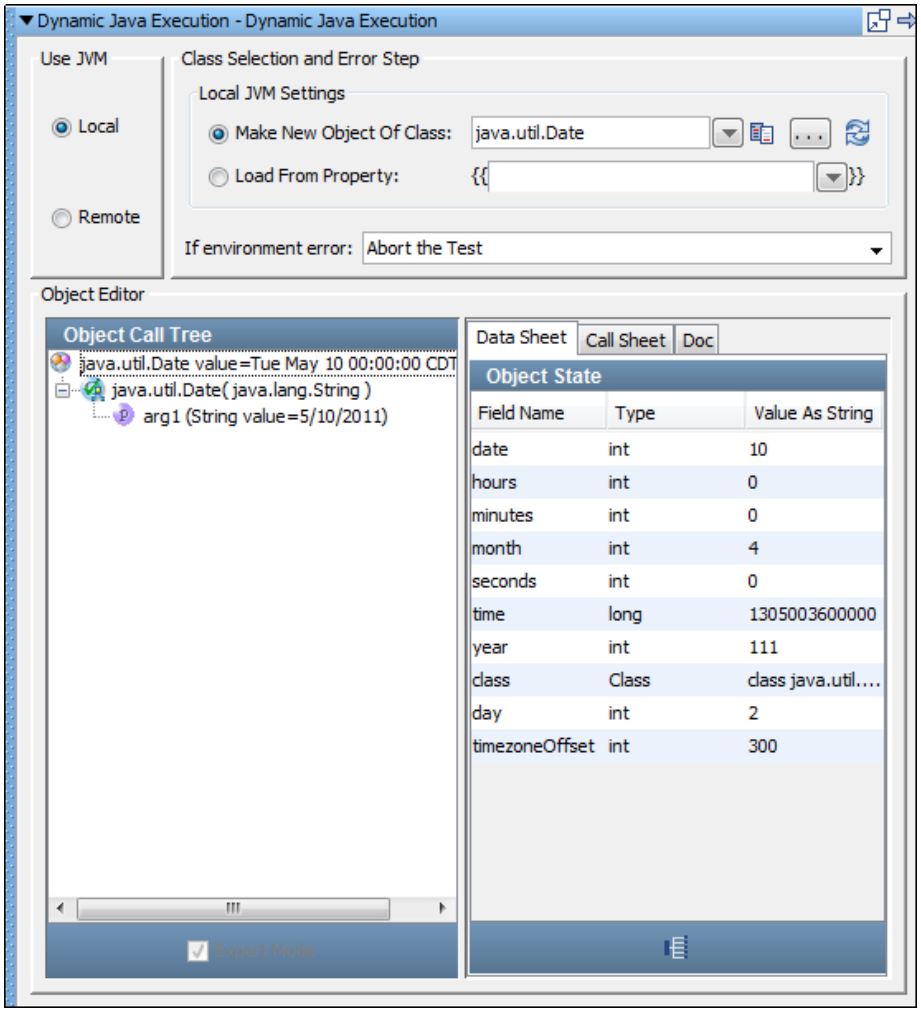
2. ［オブジェクトの構築/ロード］をクリックします。

［複合オブジェクト コンストラクタ］ウィンドウが開き、オブジェクトで使用可能なコンストラクタがリスト表示されます。

3. コンストラクタを選択して、［次へ］をクリックします。
4. コンストラクタが必要とする入力パラメータを入力します。



5. この例では、日付（2011/5/10）の文字列表現を入力します。 値、プロパティ、または NULL を入力できます。 DevTest はオブジェクトを構築し、複合オブジェクト エディタにロードします。



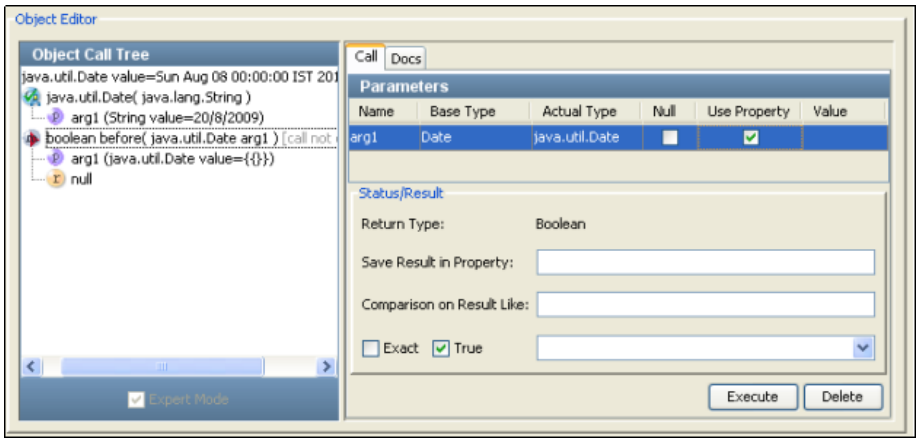
6. これで、複合オブジェクト エディタを使用して、オブジェクトを操作し、メソッドを実行できます。

   複合オブジェクト エディタの使用方法の詳細については、「*CA Application Test の使用*」の「複合オブジェクト エディタ（COE）」を参照してください。

7. 以下のいずれかの方法で、フィルタおよびアサーションを追加できます。

   - インライン フィルタ/アサーション フォーム（複合オブジェクト エディタの一部）を使用する。
   - テスト ケース ツリーのテスト ステップの下でフィルタを手動で選択する。

   たとえば、以下の図は、Date クラスに対して before メソッドを実行する前のウィンドウです。



［ステータス/結果］セクションでは、［結果を保存するプロパティ］テキスト ボックスでインライン フィルタを追加できます。また、［結果の比較 - Like］テキスト ボックスでインライン アサーションを追加できます。

動的 Java 実行ステップには、「動的 Java 実行」という命名規則を使用したデフォルトの名前があります。 ステップ名は、いつでも変更できます。

RMI サーバ実行

RMI サーバ実行ステップでは、以下のアクションを実行できます。

- RMI（リモート メソッド呼び出し）によってリモート Java オブジェクトへの参照を取得する。
- Java オブジェクトをコールする。

前提条件 : 複合オブジェクト エディタについての知識。また、リモート オブジェクトのインターフェースおよびスタブ クラスをホット デプロイ ディレクトリにコピーする必要があります。 これらのアクションは、リモート オブジェクトへの接続および操作に必要です。これらのクラスは、リモート オブジェクト開発者から入手してください。

パラメータ要件 : RMI サーバに接続する方法（通常はホスト名およびポート）と呼び出すオブジェクトの RMI 名を知っている必要があります。

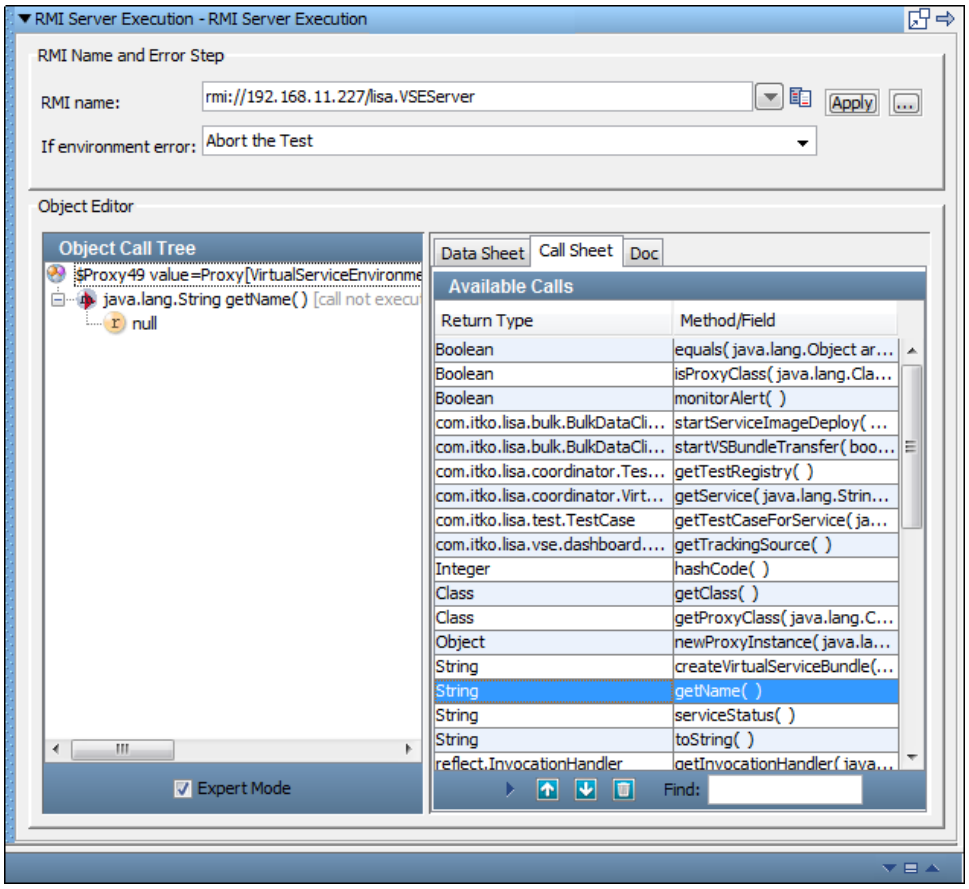
RMI サーバ ステップ エディタでは、以下のパラメータを入力できます。

RMI Name

以下のアクションのいずれかを完了します。

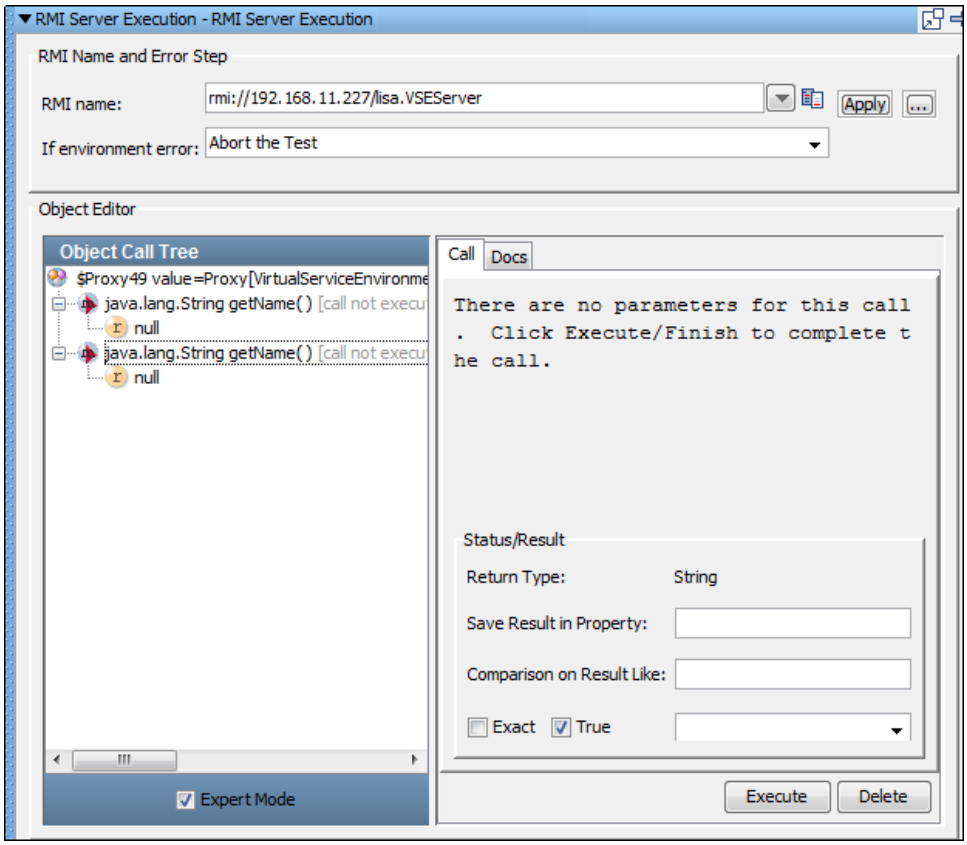
- オブジェクトの完全 RMI 名（上記のとおり）を入力または選択する。
- RMI サーバ名を入力し、［適用］をクリックして利用可能なオブジェクトのリストを開く。

DevTest はオブジェクトを構築し、複合オブジェクト エディタにロードします。

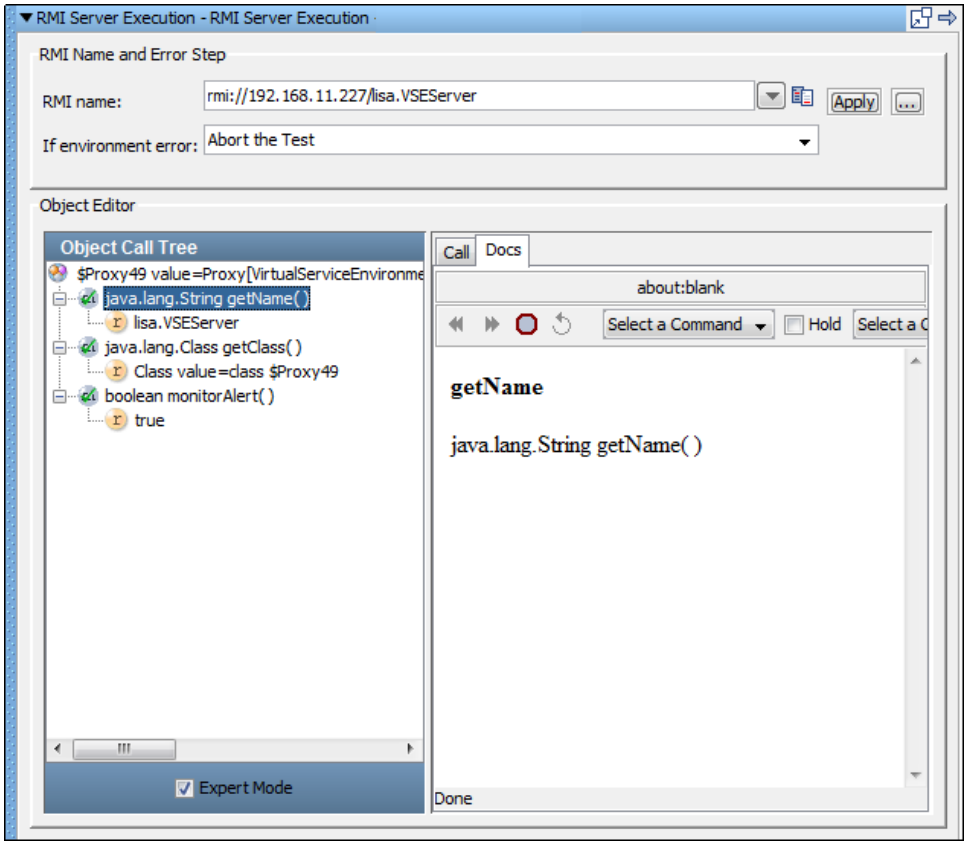


これで、複合オブジェクト エディタを使用して、オブジェクトを操作し、メソッドを実行できます。

上記の図では、getName メソッドが選択されています。 それをダブルクリックすると、オブジェクト コール ツリーに追加されます。 次に、ダイアログ ボックスの［実行］をクリックし、必要なメソッド引数と、結果の処理方法に関する情報を使用して実行できます。



上記の図では、メソッドがまだ実行されていないため、戻り値として NULL が表示されています。 ［実行］をクリックすると実行され、正しい戻り値型が表示されます。 以下の図は、オブジェクト コール ツリーがいくつかのメソッドの実行結果をどのように追跡するかを示しています。



また、以下のいずれかの方法でも、フィルタおよびアサーションを追加できます。

- インライン フィルタ/アサーション フォームを使用する。
- テスト ケース ツリーのテスト ステップの下でフィルタを手動で選択する。

［ステータス/結果］セクションが表示されます。このセクションの［結果を保存するプロパティ］テキスト ボックスでインライン フィルタを追加できます。［結果の比較 - Like］テキスト ボックスでインライン アサーションを確認できます。

注: 複数のネットワーク カードがある場合、RMI 名に localhost を使用するとエラーが発生する場合があります。 IP アドレスまたは IP アドレスに対応するホスト名を使用する必要がある場合があります。

RMI サーバ実行ステップには、「RMI サーバ実行 - 操作名」という命名規則を使用したデフォルトの名前があります。 デフォルトのステップ名を別のステップが使用する場合、DevTest は、このステップ名に番号を追加して一意にします。 ステップ名は、いつでも変更できます。

Enterprise JavaBean 実行

Enterprise JavaBean 実行ステップでは、J2EE アプリケーション サーバで実行される Enterprise JavaBean （EJB）への参照を取得し、コールを実行することができます。

EJB のテストは、Java オブジェクトのテストと同様です。 DevTest は、ホーム EJB インターフェースを使用して動的に EJB に接続し、EJB オブジェクトのインスタンスを作成します。 ホーム インターフェースを必要としないため、このプロセスは EJB と少し異なります。 テスト中の EJB は、複合オブジェクト エディタにロードされます。このエディタでは、Java コードを記述せずに EJB を操作できます。

前提条件： 複合オブジェクト エディタについての知識。 アプリケーション クライアント JAR およびクライアント EJB JAR は、DevTest クラスパスに存在する必要があります。 これらの JAR ファイルは両方とも、hotDeploy ディレクトリにコピーされます。 EJB のサンプルをすぐに実行できるように、hotDeploy ディレクトリには、JBoss アプリケーション サーバ用の jboss-all-client.jar ファイルおよびサンプル JAR ファイルが含まれます。

パラメータ要件

- サーバ接続情報（JNDI 接続）とユーザ ID およびパスワード（必要な場合）。
- EJB ホーム インターフェースのグローバル JNDI ルックアップ名。

EJB デプロイヤは、この情報を提供する必要があります。

SIBC を使用した DevTest Solutions による WebSphere への接続

IBM は、ダウンロードして Sun JVM と一緒に使用できる EJB および JMS クライアントを提供しています。 このクライアントは、ここから入手できます。

次の手順に従ってください:

1. このファイルをダウンロードし、そのインストーラを実行します。
2. 以下のコマンドを実行してください。

   ```
   java -jar sibc_install-o0902.06.jar jms_jndi_sun <出力ディレクトリ>
   ```

3. <出力ディレクトリ> から以下のファイルを取得します。
   - lib¥sibc.jms.jar
   - lib¥sibc.jndi.jar
   - lib¥sibc.orb.jar
4. 上記の 3 つの JAR ファイルを参照するために、LISA_PRE_CLASSPATH 環境変数を作成します。

   例： LISA_PRE_CLASSPATH=C:¥sibc.jms.jar;C:¥sibc.jndi.jar;C:¥sibc.orb.jar;
5. **local.properties** を編集し、以下の行を追加します。

   ```
   com.ibm.CORBA.ORBInit=com.ibm.ws.sib.client.ORB
   ```

6. JMS ステップで、以下の設定を使用します。
   - **JNDI ファクトリ クラス：** com.ibm.websphere.naming.WsnInitialContextFactory
   - **JNDI URL：** iiop://SERVER:PORT

例

この例では、ITKO サンプル サーバ（JBoss サーバ）を使用します。 ローカルのデモ サーバを使用するために、ホスト名として localhost を使用します。

1. 以下のパラメータを入力します。

   アプリケーション サーバを選択

   リストからアプリケーション サーバを選択します。 アプリケーション サーバがリストにない場合は、［その他/ユーザ指定］オプションを選択します。

   エディタの下部セクションは、選択内容によって異なります。 上記の図は、JBoss の設定パネルを示しています。

2. JBoss パネルでは、以下のパラメータを入力します。

   ホスト名または IP アドレス

   アプリケーション サーバのホスト名と IP アドレスを入力します。

   ポート番号

   ポート番号を入力します。

   ユーザ

   アプリケーション サーバにユーザ ID が必要な場合に入力します。

   パスワード

   アプリケーション サーバにパスワードが必要な場合に入力します。

3. ［次へ］をクリックします。

［その他/ユーザ指定］**ウィンドウ**

1. 以下のパラメータを入力します。

   JNDI ファクトリ

   アプリケーション サーバの完全修飾 JNDI ファクトリ クラス名を入力または選択します。

   JNDI サーバ URL

   JNDI サーバ名を入力または選択します。

   ユーザ

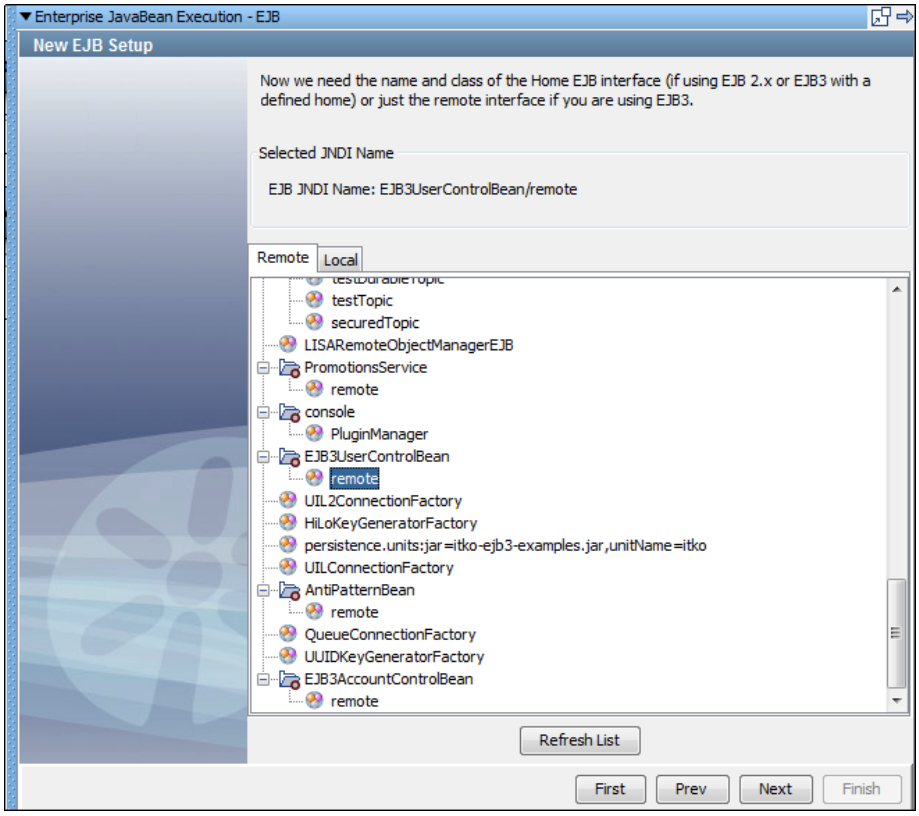
アプリケーション サーバにユーザ ID が必要な場合に入力します。

パスワード

アプリケーション サーバにパスワードが必要な場合に入力します。

2. ［次へ］をクリックします。

［新規 EJB セットアップ］ウィンドウが開き、アプリケーション サーバに登録されている JNDI 名がすべてリスト表示されます。

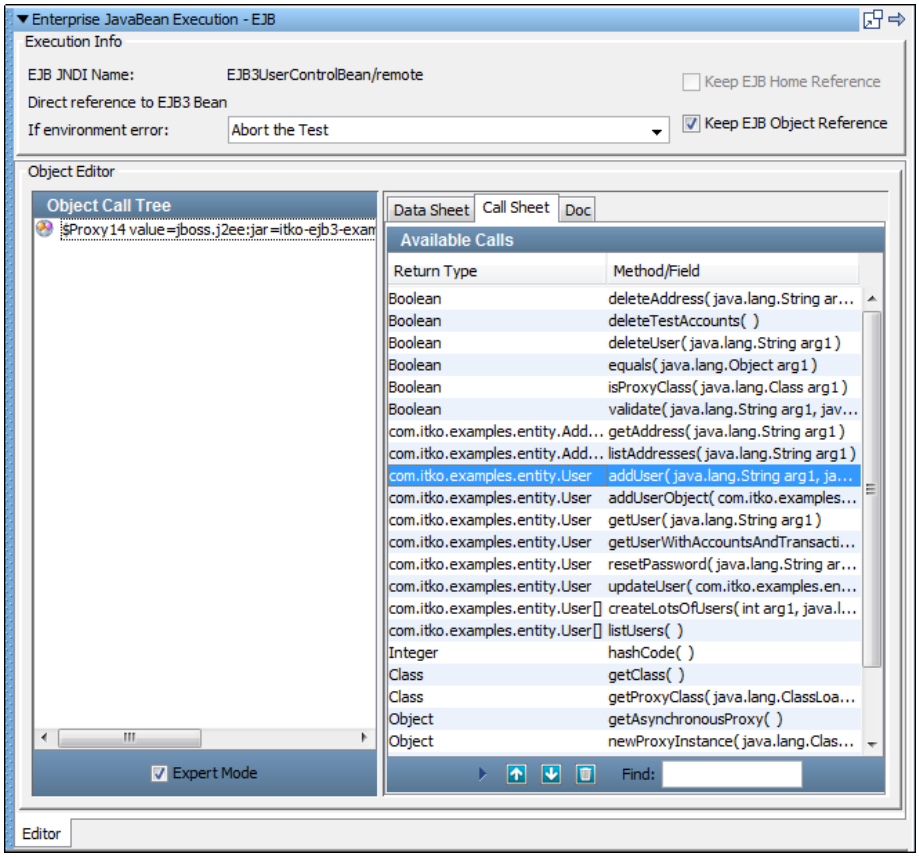


3. 適切な EJB ホーム インターフェースの名前を選択します。

この例では、JNDI 名は **com.itko.examples.ejb.UserControlBean** です。EJB3 仕様では、ステートフル ビーンおよびステートレス ビーンを JNDI ツリーに直接バインドでき、ホーム インターフェースを必要としません。 その場合、ビーンは直接選択でき、DevTest がインスタンスを作成する必要はありません。

4. ［次へ］をクリックします。

オブジェクトが構築され、複合オブジェクト エディタにロードされます。



［実行情報］領域に、現在の EJB 情報が表示されます。 この EJB を再利用する場合は、以下のチェック ボックスをオンにすることにより EJB オブジェクトおよび EJB ホームへの参照を保持できます。

- EJB ホーム参照の保持
- EJB オブジェクト参照の保持

ビーンがホーム インターフェースのない EJB 3 ビーンである場合、［EJB ホーム参照の保持］チェック ボックスは無効です。 ［例外の場合］を例外が発生した場合にリダイレクトするステップに設定します。

EJB ステップには、「EJB Java メソッド 動的 Java 実行」という命名規則を使用したデフォルトの名前があります。 ステップが保存または入力されるまでは、上記に示すように、デフォルトのステップ名は *EJB* です。 別のステップもデフォルトのステップ名を使用する場合、番号がステップ名に追加されます。 ステップ名は、いつでも変更できます。

5. これで、複合オブジェクト エディタを使用して、オブジェクトを操作し、メソッドを実行できます。

   使用方法は、RMI サーバ実行 (P. 279)ステップと同じです。

その他のトランザクション ステップ

以下のステップが使用できます。

SQL データベース実行（JDBC） (P. 289)
SQL データベース実行（JDBC/アセット） (P. 294)
CORBA 実行 (P. 297)

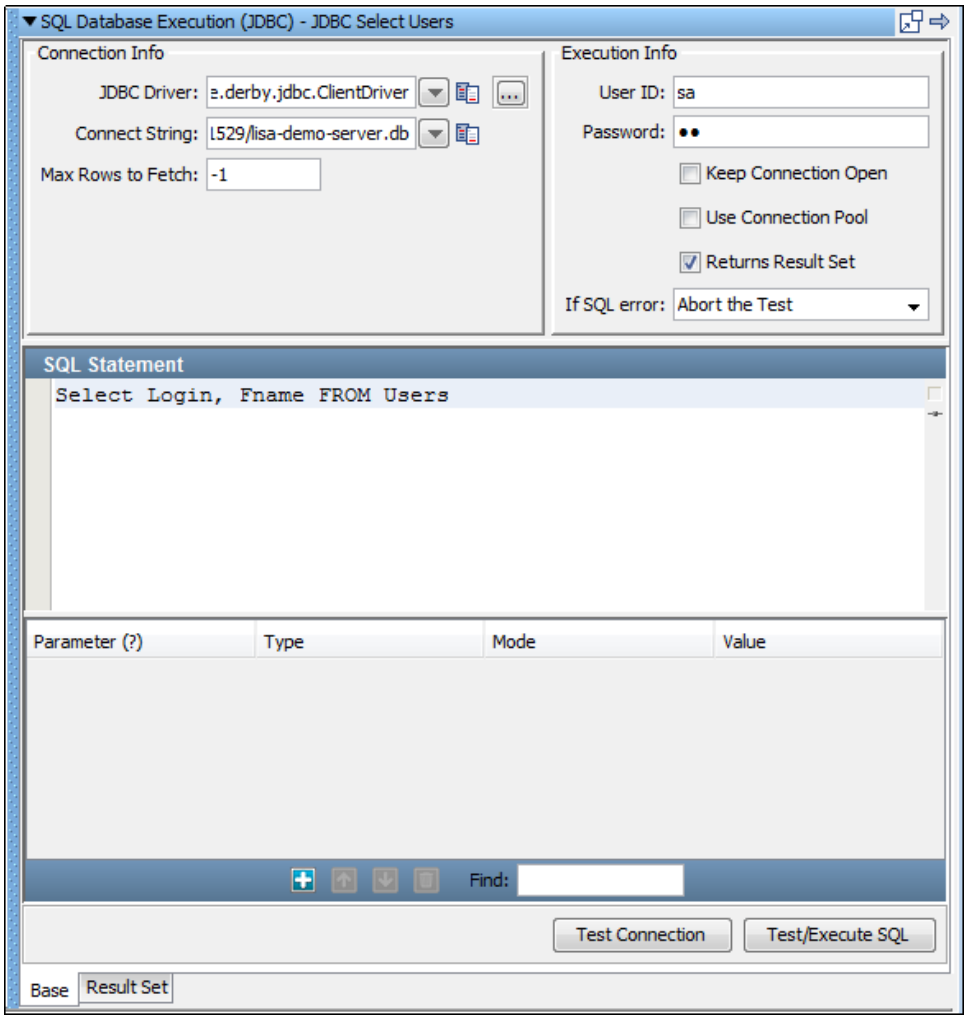
SQL データベース実行（JDBC）

SQL データベース実行ステップでは、JDBC （Java Database Connectivity）を使用してデータベースに接続し、データベースに対して SQL クエリを実行することができます。

SQL 構文が完全にサポートされています。ただし、SQL は検証されません。SQL はデータベースに送信され、そこで検証されます。 SQL エラーが発生すると、応答にキャプチャされます。 エラーに対してアサートを行えます。 SQL が、使用しているデータベース マネージャで有効であることを確認してください。

前提条件： データベースの適切な JDBC ドライバが DevTest クラスパスに存在する必要があります。ドライバ JAR ファイルは、ホット デプロイ ディレクトリに配置できます。 DevTest クラスパスには Derby クライアント ドライバが含まれています。したがって、再度それを追加する必要はありません。

パラメータ要件： JDBC ドライバ クラスの名前、データベースの JDBC URL、およびデータベースのユーザ ID とパスワード。 また、SQL クエリを作成するために、データベースのテーブルのスキーマを知っておく必要があります。



1. SQL データベース ステップ エディタでは、以下のパラメータを入力します。

   接続情報

   JDBC ドライバ

   適切なドライバ クラスの完全なパッケージ名を入力または選択します。標準のドライバ クラスは、ドロップダウン リストで使用可能です。また、［参照］ボタンを使用して、ドライバ クラスの DevTest クラス パスを参照できます。

   接続文字列

接続文字列はデータベースの標準の JDBC URL です。 URL を入力または選択します。一般的なデータベース マネージャの JDBC URL テンプレートは、ドロップダウン リストで使用可能です。

取得する最大行数

結果セットで返される最大行数を入力します。 このフィールドは必須です。 行数に制限を設けない場合は、「-1」と入力します。

実行情報

ユーザ ID

ユーザ ID を入力します（データベースで必要な場合）。

パスワード

パスワードを入力します（データベースで必要な場合）。

接続を維持

このオプションをオンにすると、ステップが実行して最初に開かれたデータベース接続がキャッシュされます。 そのステップに対してガベージ コレクションが発生すると、データベース接続が閉じられます。 ［接続を維持］がオフの場合、接続はステップが実行されるたびに閉じられます。

結果セットを返す

クエリで結果セットを返す場合（SELECT タイプのクエリの場合）は、このチェック ボックスをオンにします。 UPDATE、INSERT、または DELETE の場合はオフにします。このチェック ボックスを誤って設定すると、クエリでエラーが発生します。

SQL エラーの場合

エラーが発生した場合にリダイレクトするステップを選択します。

ローカルのデモ サーバと通信するには、以下を使用します。

JDBC ドライバ

org.apache.derby.jdbc.ClientDriver

接続文字列

jdbc:derby://localhost:1527/reports/lisa-reports.db

ユーザ ID

sa

パスワード

sa

2. ユーザ ID やパスワード（必要な場合）などのデータベース接続情報を入力した後、［テスト接続］をクリックして接続をテストします。

   情報が正しい場合は、成功メッセージ ウィンドウが表示されます。情報が正しくない場合は、エラー メッセージが表示されます。

3. これで、下部ウィンドウに SQL ステートメントを入力することができます。

   SQL ではプロパティを使用できます。 DevTest は SQL 文字列をデータベースへ渡す前にパラメータを置換します。

   JDBC ステップは、ストアド プロシージャ コールをサポートしています。ストアド プロシージャが入力および戻り値として使用する引数として、基本データ型（文字列、数値、日付、ブール値）がサポートされています。 パラメータを追加するには、［追加］ をクリックします。 ［パラメータ］列の数値は編集できません。 行を追加、削除、移動すると、［パラメータ］列の数値が自動的に再設定されます。

   また、JDBC ステップでは、JDBC のプリペアード ステートメントも使用できます。SQL ステートメントでは、疑問符を使用できます。また、{{プロパティ}} を指定して追加できます。 この機能により、引数の型とパラメータ値の一重引用符のエスケープを気にしなくてすみます。{{col1}} および {{col2}} への参照を持つステートメント「insert into MYTABLE(COL1,COL2) values (?, ?)」を考えると、理解しやすくなります。型およびエスケープ文字は自動的に変換されます。

   ステートメントに NULL を含めるには、**{{<<NULL>>}}** 構文を使用します。

4. SQL クエリを作成したら、［SQL をテスト/実行］をクリックして、クエリを実行します。

   メッセージは結果のステータスを示します。

5. ［OK］をクリックします。

   結果は、［結果セット］タブに表示されます。

6. これで、結果セットに対してフィルタおよびアサーションを作成できます。

   ［結果セット］タブの下部のアイコンは、以下のフィルタおよびアサーションへの容易なアクセスを提供します。

   結果セット行の別の値に対する値の取得

検索フィールドのセル（値フィールドのフィルタ）を選択し、プロパティ名を入力します。 検索フィールドのセルの値が見つかった場合、その行の［値］フィールドの値が入力されるプロパティの値として設定されます。

結果の値を解析フィルタ

選択したセルの値が入力されるプロパティの値として設定されます。

結果セットのコンテンツ アサーション

選択したフィールド（列）の値が入力した正規表現と比較されます。

これらおよびその他の結果セットの適切なフィルタおよびアサーションの詳細については、「フィルタの説明 (P. 141)」および「アサーションの説明 (P. 9)」を参照してください。

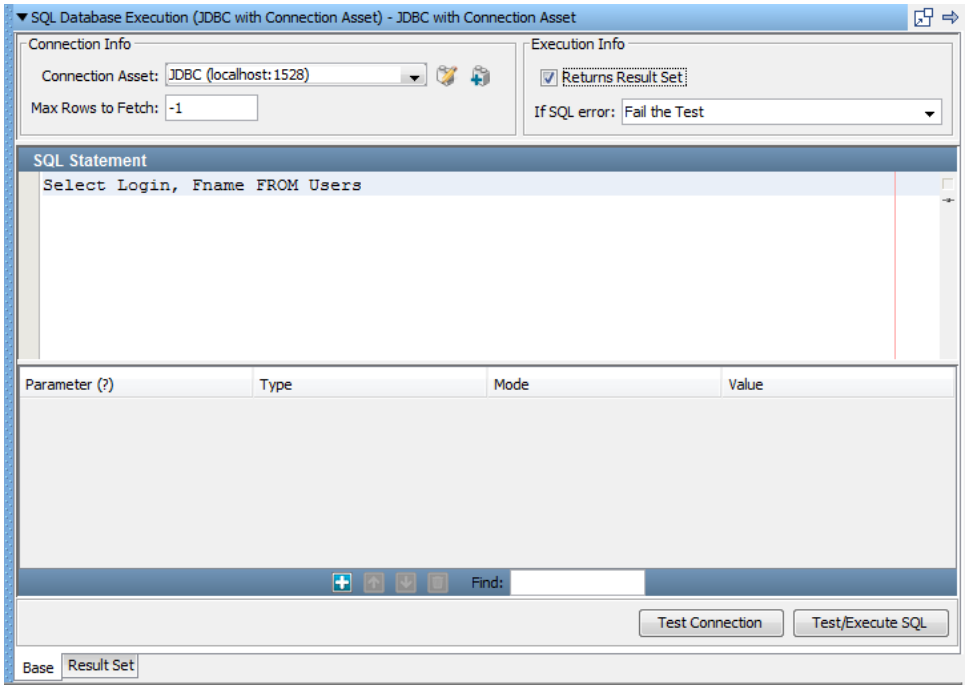
SQL データベース実行ステップには、「JDBC SQL 関数 テーブル名」という命名規則を使用したデフォルトの名前があります。 ステップのデフォルト名でサポートされている関数は、*select*、*insert*、*delete*、*update*、および *perform* です。 ステップ名は、いつでも変更できます。

SQL データベース実行（JDBC/アセット）

SQL データベース実行ステップでは、JDBC （Java Database Connectivity）を使用してデータベースに接続し、データベースに対して SQL クエリを実行することができます。 JDBC 接続アセット (P. 67)が定義されている場合、このステップを使用します。

注: レガシー SQL データベース実行（JDBC）ステップがある場合、接続情報を手動で作成する代わりに、既存のステップからエクスポートできます。詳細については、「テスト ステップからのアセットの作成」を参照してください。

SQL 構文が完全にサポートされています。ただし、SQL は検証されません。SQL はデータベースに送信され、そこで検証されます。 SQL エラーが発生すると、応答にキャプチャされます。 エラーに対してアサートを行えます。 SQL が、使用しているデータベース マネージャで有効であることを確認してください。



1. SQL データベース ステップ エディタでは、以下のパラメータを入力します。

   接続情報

   接続アセット

JDBC 接続用の接続パラメータが含まれる接続アセット (P. 67)を選択します。

取得する最大行数

結果セットで返される最大行数を入力します。 このフィールドは必須です。 行数に制限を設けない場合は、「-1」と入力します。

実行情報

結果セットを返す

クエリで結果セットを返す場合（SELECT タイプのクエリの場合）は、このチェック ボックスをオンにします。 UPDATE、INSERT、または DELETE の場合はオフにします。このチェック ボックスを誤って設定すると、クエリでエラーが発生します。

SQL エラーの場合

エラーが発生した場合にリダイレクトするステップを選択します。

2. ユーザ ID やパスワード（必要な場合）などのデータベース接続情報を入力した後、［テスト接続］をクリックして接続をテストします。

   情報が正しい場合は、成功メッセージ ウィンドウが表示されます。情報が正しくない場合は、エラー メッセージが表示されます。

3. これで、下部ウィンドウに SQL ステートメントを入力することができます。

   SQL ではプロパティを使用できます。 DevTest は SQL 文字列をデータベースへ渡す前にパラメータを置換します。

   JDBC ステップは、ストアド プロシージャ コールをサポートしています。ストアド プロシージャが入力および戻り値として使用する引数として、基本データ型（文字列、数値、日付、ブール値）がサポートされています。 パラメータを追加するには、［追加］ をクリックします。 ［パラメータ］列の数値は編集できません。 行を追加、削除、移動すると、［パラメータ］列の数値が自動的に再設定されます。

また、JDBC ステップでは、JDBC のプリペアード ステートメントも使用できます。SQL ステートメントでは、疑問符を使用できます。また、{{プロパティ}} を指定して追加できます。 この機能により、引数の型とパラメータ値の一重引用符のエスケープを気にしなくてすみます。{{col1}} および {{col2}} への参照を持つステートメント「insert into MYTABLE(COL1,COL2) values (?, ?)」を考えると、理解しやすくなります。型およびエスケープ文字は自動的に変換されます。

ステートメントに NULL を含めるには、**{{<<NULL>>}}** 構文を使用します。

4. SQL クエリを作成したら、［SQL をテスト/実行］をクリックして、クエリを実行します。

   メッセージは結果のステータスを示します。

5. ［OK］をクリックします。

   結果は、［結果セット］タブに表示されます。

6. これで、結果セットに対してフィルタおよびアサーションを作成できます。

   ［結果セット］タブの下部の 3 つのアイコンは、以下のフィルタおよびアサーションへの容易なアクセスを提供します。

   結果セット行の別の値に対する値の取得

   検索フィールドのセル（値フィールドのフィルタ）を選択し、プロパティ名を入力します。 検索フィールドのセルの値が見つかった場合、その行の［値］フィールドの値が入力されるプロパティの値として設定されます。

   結果の値を解析フィルタ

   選択したセルの値が入力されるプロパティの値として設定されます。

   結果セットのコンテンツ アサーション

   選択したフィールド（列）の値が入力した正規表現と比較されます。

   これらおよびその他の結果セットの適切なフィルタおよびアサーションの詳細については、「フィルタの説明 (P. 141)」および「アサーションの説明 (P. 9)」を参照してください。

SQL データベース実行ステップには、「JDBC/接続アセット SQL 関数テーブル名」という命名規則を使用したデフォルトの名前があります。 ステップのデフォルト名でサポートされている関数は、*select*、*insert*、*delete*、*update*、および *perform* です。 ステップ名は、いつでも変更できます。

CORBA 実行

CORBA 実行ステップは、Java RMI-IIOP ライブラリを使用して CORBA コールを行うために使用します。 適切なスケルトン クラスを提供する必要があります。

次の手順に従ってください:

1. 実行を開始する前に、**corbaserver.jar** ファイルを DevTest lib ディレクトリにコピーします。 この JAR は、CORBA サーバの lib ディレクトリにあります。
2. ステップ エディタを開くには、CORBA ステップを選択します。
3. フィールドに必要なデータを入力します。

   オブジェクト IOR

   このフィールドには、オブジェクトまたはネーム サービスの RAW IOR 文字列が含まれます。 この文字列は、**nameserver.sh** バッチ ファイルを実行した出力（生成された IOR）から取得できます。

   注: nameserver の出力からコピーして［オブジェクト IOR］フィールドに貼り付けると、スペースが含まれる場合があります。 この文字列にはスペースが含まれないようにしてください。

   クラス名

   IOR はこのオブジェクト クラスを参照します。

   また、［IOR 構築］ダイアログ ボックスも使用できます。 このダイアログ ボックスが開いている場合、個別のパーツに入力されているすべての IOR 文字列を解析します。奇妙な外観のキーは無視します。IOR はバイト形式でキーを格納するため、一部のバイトを正しく表示できません。 RAW IOR の解析されたバージョンを使用する場合は、フィールドを編集しないでください。
4. ［オブジェクトの構築/ロード］をクリックします。

   動的オブジェクト エディタに、オブジェクトのコール シートが表示されます。
5. コールするメソッドを選択し、実行します。

デフォルトの CORBA 実行ステップは、「*CORBA クラス名*」という命名規則を使用します。 ステップ名は、いつでも変更できます。

ユーティリティ ステップ

以下のステップが使用できます。

プロパティを最終応答として保存

プロパティを最終応答として保存ステップでは、最終応答としてプロパティの値を保存できます。

既存のプロパティのプロパティ名を入力するか、またはプルダウン メニューから選択します。 プロパティの値は、最終応答（およびステップ応答）としてロードされます。 その後、この値は、最終応答としてすぐにアクセスできるか、または後で **lisa.thisStepname.rsp** プロパティ（**thisStepName** は現在のステップの名前）を使用してテスト ケースでアクセスできます。

テスト ケースの各ステップには、関連付けられた応答があります。 そのステップが実行されると、応答は **LASTRESPONSE** および **lisa.thisStepName.rsp** という 2 つのプロパティに自動的に保存されます。 このステップを使用すると、ステップの実際の応答ではなく、プロパティ値にフィルタおよびアサーションを適用できます。

ログ メッセージの出力

「ログ メッセージの出力」ステップでは、ログにテキスト メッセージを書き込むことができます。 このステップは、実行されるテスト ケースの追跡およびログに役立ちます。 ログ メッセージは、通常、テキストとプロパティの組み合わせです。

ログ メッセージをエディタで入力します。 このログ メッセージは、このステップが対話型テスト ラン （ITR）で実行された場合に表示されるか、またはテスト ラン中にログに記録されます。

区切りファイルへの書き込み

区切りファイルへの書き込みステップでは、複数のプロパティの現在値を CSV ファイルに保存できます。このステップは通常、プロパティが複数のテスト ケースで共有されている場合、またはテストが失敗した場合のデバッグに使用します。 既存のプロパティまたは新しいプロパティを対象にできます。 既存のプロパティを別のプロパティ名で保存できます。

以下のパラメータを入力します。

ファイル名

ファイルのパス名を入力するか、または［参照］ボタンを使用してファイルを参照します。

エンコーディング

デフォルトの UTF-8 エンコーディングを使用するか、またはドロップダウン リストから代わりのエンコーディングを選択します。

BOM を含める

選択したエンコーディングがバイト オーダー マークを含んでいる場合、このチェック ボックスをオンにして、ファイルの先頭に BOM を含めることができます。

区切り文字

ファイルに使用する区切り文字を入力します。

行末

ドロップダウンからファイルの正しい行末文字を選択します。

書き込むプロパティ

［追加］ をクリックして、行を追加します。 次に、保存するプロパティ（ヘッダ）および対応する値（値）を入力します。 プロパティのリストには、既存または新しいプロパティを含めることができます。 既存のプロパティ（ヘッダ）を指定する場合、新しい値を指定してそれらの現在値を上書きできます。または、既存の値を使用することもできます。

上記の例の場合

- 最初の 2 つのプロパティは、変更なしで保存されています。
- 3 番目のプロパティは、新しい名前（ヘッダ）と日付で保存されています。

- 4 番目のプロパティは、その値として正規表現が使用されています

プロパティは、ファイルの最初の行が保存されているプロパティ名のセットである CSV ファイルに保存されます。 2 番目の行には、それぞれのプロパティに対応する値が含まれます。

注: このステップは、テスト ケース間でデータを渡すために使用します。 たとえば、最初のテストでは、銀行にカスタマを追加して新しい口座番号のリストを返すと仮定します。 それらの口座をファイルに書き込むことができます。 2 番目のテストは、口座を取得して預け入れを行うことができます。 2 番目のテストでは、データ セットを使用して、最初のテストで作成されたファイルを読み取ります。

区切りファイルに書き込む場合は、以下の点に注意してください。 Data ディレクトリは、両方のテストが再実行をサポートしている場合にのみ、CSV ファイルの場所として使用できます。 2 番目のテストが実行できるように、最初のテストを再実行して **lads** フォルダに CSV を作成する必要があります。 テスト ケースまたはスイートが実行されている間、**lads** ディレクトリには一時的にファイルが保存されます。

最初のテストを実行せずに 2 番目のテストを実行するには、project または MAR 以外の共通の場所にデータ セットを置く必要があります。

区切りファイルへの書き込みステップは、実行ごとに 1 つ（1 つのみ）のデータ行を書き込みます。 例外は、最初の実行でターゲット ファイルが存在しない場合です。 ファイルが存在しない場合、選択されたエンコーディングのバイト オーダー マークに続いてキーが含まれる行が書き込まれます。

区切りファイルへの書き込みステップのデフォルト名は、「ファイル < ファイル> へのプロパティの書き込み」です。 デフォルトのステップ名を別のステップが使用する場合、DevTest は、このステップ名に番号を追加して一意にします。 ステップ名は、いつでも変更できます。

ファイルからのプロパティの読み取り

ファイルからのプロパティの読み取りステップは、外部ファイルからプロパティを読み取るために使用します。プロパティは 2 つの方法で読み取ることができます。

- 名前/値ペア
- XML タグ

以下のパラメータを入力します。

ファイル名

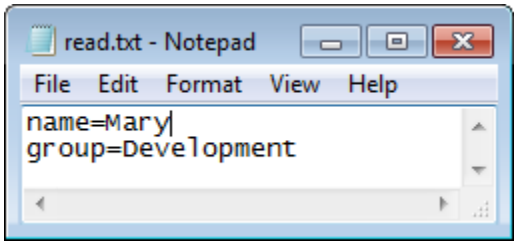
ファイルのパス名を入力するか、または［参照］ボタンを使用してファイルを参照します。

ファイル エンコーディング

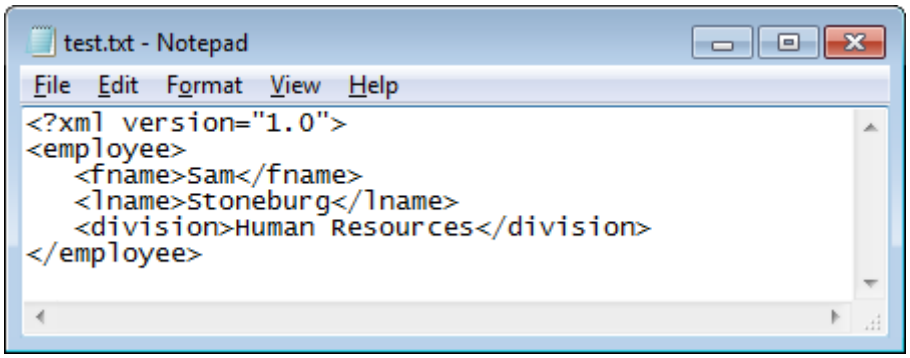
デフォルトの UTF-8 エンコーディングを使用するか、またはドロップダウン リストから代わりのエンコーディングを選択します。［自動検出］を選択して［検出］をクリックし、DevTest にエンコーディング タイプを選択させます。

ファイル タイプ

ファイルのタイプ（名前/値ペアまたは XML タグ）を選択します。



上記の図は、*name* がプロパティで *Mary* が name プロパティの値である名前/値ペア タイプのプロパティ ファイルを示しています。



上記の図は、*fname* がプロパティで *Sam* が fname プロパティの値である XML タグ タイプのプロパティ ファイルを示しています。

ファイルからのプロパティの読み取りステップには、「<*ファイル名*>のプロパティ リーダ（<*ファイル名*> は入力または選択されたファイルのリーフ名）」という命名規則を使用したデフォルトの名前があります。デフォルトのステップ名を別のステップが使用する場合、DevTest は、このステップ名に番号を追加して一意にします。 ステップ名は、いつでも変更できます。

何も実行しないステップ

何も実行しないステップはパラメータを取りません。また、それ自体には何の機能もありません。 ただし、このステップにはアサーションを追加できるため、特定の状況で役立ちます。

たとえば、スクリプト アサーションを使用して、迅速なカスタム アサーションの追加、数値または日付の比較、または複数の数値比較テストが行えます。

応答としてのテキストの解析

応答としてのテキストの解析ステップでは、最終応答として保存できるファイルからテキスト コンテンツを入力できます。 コンテンツはプロパティに格納できます。 エディタにテキストを入力または貼り付けるか、ファイルからロードできます。

以下のパラメータを入力します。

プロパティ キー

コンテンツを格納するプロパティの名前（オプション）。

ファイルからロード

クリックしてファイルを参照します。 そうでない場合は、テキストをエディタに入力するか貼り付けます。

これで、コンテンツのパラメータ化、フィルタ、およびコンテンツへのアサーションの追加を行えます。

結果のテキストを表示するには、［テスト］をクリックします。

監査ステップ

監査ステップでは、現在のテスト ステップ、リモート テスト、または仮想サービスに対して監査ドキュメントを適用できます。

このステップにより、実行中に作成されるイベントによってモデルを検証できます。

ステップをクリックすると、そのエディタが表示されます。

モード

このステップは 2 つのモードがあります。

- モニタの開始
- 監査文書の適用

モニタの開始モード

モニタの開始モードを選択する場合

監査ドキュメント

監査ドキュメントを入力または参照します。

ターゲット

監査ドキュメントの対象を選択します。

- このモデル： 現在のモデルに適用されます。
- テスト ラン： テスト ランに適用されます。
- 仮想サービス： 仮想モデルに適用されます。

環境エラーの場合

環境エラーが発生した場合に実行するステップまたは実行するアクションを選択します。

監査文書の適用モード

監査文書の適用モードを選択する場合

監査が失敗した場合

監査が失敗した場合に実行されるステップを選択します。

環境エラーの場合

環境エラーが発生した場合に実行するステップまたは実行するアクションを選択します。

Base64 エンコーダ ステップ

このステップは、Base 64 エンコーディング アルゴリズムを使用してファイルをエンコードするために使用します。

結果は、テスト ラン全体で使用されるプロパティ ファイルに格納できます。

ステップをクリックすると、そのエディタが表示されます。

以下のパラメータを入力します。

ファイル

エンコードするファイルの名前を入力するか、またはターゲットの場所を参照します。

プロパティ キー

エンコードされたファイルを格納するプロパティの名前。 このフィールドはオプションです。

環境エラーの場合

環境エラーが発生した場合に実行するステップまたは実行するアクションを選択します。

［ロード］をクリックして、エディタにファイルをロードします。

ここでは、必要に応じて、下部の［コマンド］メニューからフィルタまたはアサーション（または両方）を適用できます。

チェックサム ステップ

チェックサム ステップは、ファイルのチェックサムを計算し、プロパティにその値を保存できます。

以下のパラメータを入力します。

ファイル

チェックサムを計算するファイルのパス名を入力するか、または参照します。

プロパティ キー

チェックサム値を格納するプロパティの名前を入力します。

［ロード］をクリックします。

チェックサムが応答として表示されます。プロパティ名が入力されている場合、値はそのプロパティにあります。

これで、チェックサム値のフィルタとアサーションの追加を行えます。

XML をエレメント オブジェクトに変換

XML をエレメント オブジェクトに変換ステップは、RAW XML を以下のいずれかのタイプのオブジェクトに変換します。

- メッセージ エレメント配列
- メッセージ エレメント
- DOM エレメント

このステップは、任意のタイプを取る Web サービス API があるときに厳密な処理を使用する場合に役立ちます。 このタイプの WSDL エレメントは、入力パラメータとしてメッセージ エレメント配列を必要とします。 前のステップ（ファイルからの読み取りや応答としてのテキストの解析など）からの RAW XML をキャプチャし、プロパティに格納できます。 そのプロパティは、このステップの入力パラメータになります。

前提条件： XML は、プロパティにすでに格納されている必要があります。

以下のパラメータを入力します。

プロパティから XML をロード

XML が含まれるプロパティを入力します。このプロパティは、ユーザ定義のプロパティまたはビルトインの DevTest プロパティです。

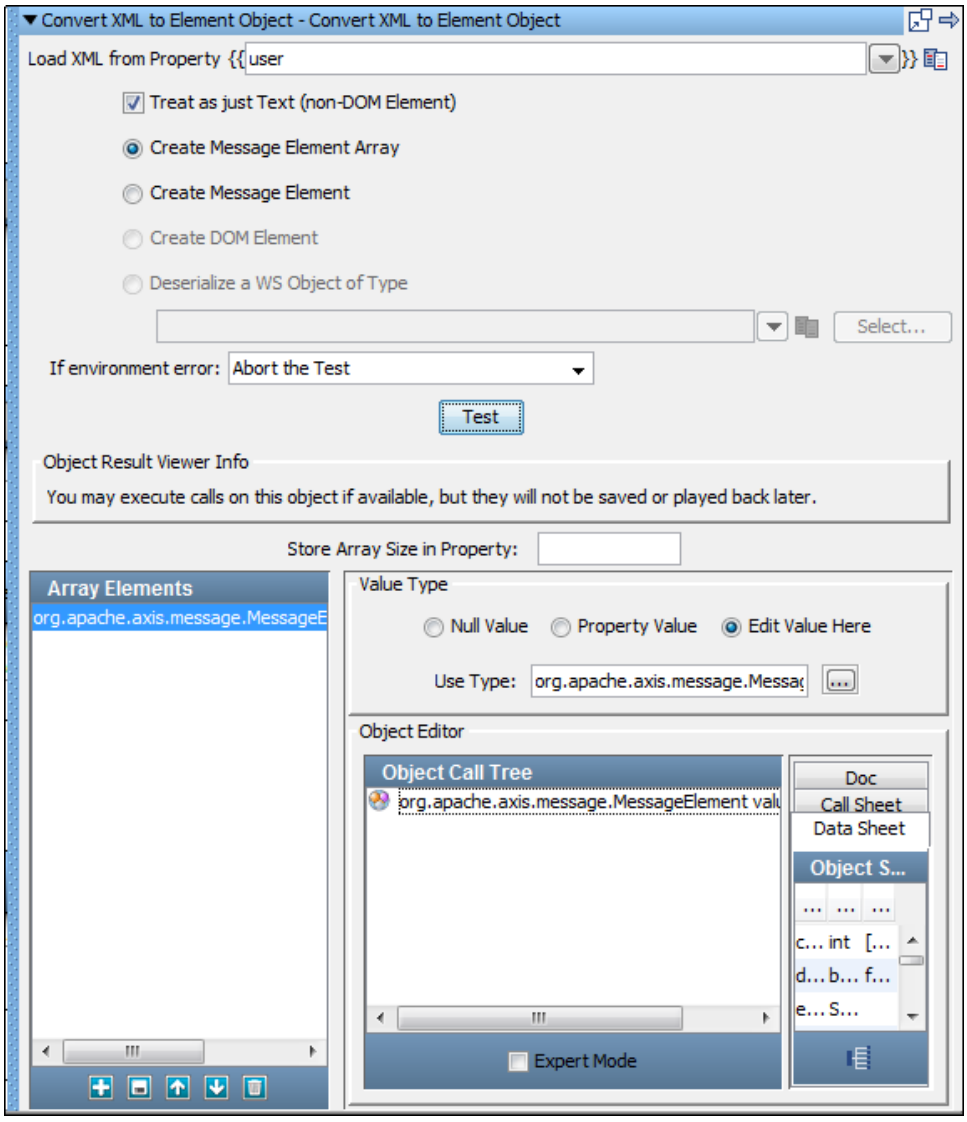
テキストとして処理

入力として、XML ではなくプレーン テキストを使用する必要がある場合に、このチェック ボックスをオンにします。 これをオンにすることにより、メッセージ エレメントにはプレーン テキストが含まれます。

各オプション ボタンをクリックして、利用可能なタイプからオブジェクトのタイプを選択します。

変換を実行するには、［テスト］をクリックします。

［テキストとして処理］チェック ボックスをオンにすると、以下の図のような結果が表示されます。［メッセージ エレメントの作成］、［DOM エレメントの作成］、および［WS オブジェクトのデシリアライズ］オプションは淡色表示されます。



このオブジェクトが別のステップでパラメータとして必要な場合、このステップの応答を使用します。 フィルタ内の応答を保存するには、ステップ応答の格納フィルタを使用します。 または、**lisa.<ステップ名>.rsp** として参照できます。

応答用の文字列比較ルックアップ

応答用の文字列比較ルックアップ ステップは、仮想サービスへの受信要求を検査し、適切な応答を決定するために使用します。 部分テキスト一致、正規表現などを使用して、受信要求に一致させることができます。

仮想 Web サービス HTTP レコーダを使用した場合、このステップは自動的に記録され、仮想サービスに追加されます。

ステップ エディタを開くには、ステップをクリックします。

以下のパラメータを入力します。

一致対象テキスト

条件に一致させるテキストを入力します。 この値は通常、LASTRESPONSE などのプロパティ参照です。

一致対象範囲

範囲の開始および終了を入力します。

一致が見つからない場合

一致が見つからない場合に実行するステップを選択します。

環境エラーの場合

環境エラーが発生した場合に実行するステップまたは実行するアクションを選択します。

応答をテスト ケース ファイルに圧縮形式で格納します

デフォルトで選択されます。 このオプションは、テスト ケース ファイルに応答を圧縮します。

ケース応答エントリ

このテーブルでは、［追加］、［移動］、［削除］をクリックして、エントリを追加、移動、削除できます。 テーブルの列を以下に示します。

有効

エントリを追加する場合にデフォルトでオンになります。エントリを無視するにはオフにします。

name

ケース応答エントリの一意の名前を入力します。

遅延仕様

遅延仕様の範囲を入力します。デフォルトは 1000-10000 です。これは、1,000 から 10,000 ミリ秒のランダムに選択された遅延時間が使用されることを示します。（構文は、反応時間仕様と同じ形式です。）

条件

この領域には、エントリが［一致対象テキスト］フィールドに一致する場合のこのステップの応答が表示されます。条件を編集するには、［条件］列領域で適切な行を選択し、条件リストに別の設定を入力します。 続く行を更新するには、［入力］をクリックします。

比較タイプ

リストから比較タイプのオプションを選択します。

- 文字列内の検索（デフォルト）
- 正規表現
- 前方一致
- 後方一致
- 完全一致

応答

エントリのステップ応答を更新できます。

条件

エントリの条件文字列を更新します。

応答

この領域には、エントリが［一致対象テキスト］フィールドに一致する場合のこのステップの応答が表示されます。 応答を編集するには、［ケース応答エントリ］領域で適切な行を選択し、［応答］リストに異なる設定を入力します。 上記の行でそれを更新するには、［入力］をクリックします。

次のステップ用の文字列比較ルックアップ

このステップは、受信要求を検査し、適切な次のステップを決定するために使用します。

部分テキスト一致、正規表現などを使用して、受信要求に一致させることができます。 各一致条件では、一致が検出された場合の転送先ステップ名を指定します。

JDBC データベース トラフィック レコーダを使用した場合、このステップは自動的に記録され、仮想サービスと照合されます。

ステップ エディタを開くには、ステップをクリックします。

以下のパラメータを入力します。

一致対象テキスト

条件に一致させるテキストを入力します。 この値は通常、LASTRESPONSE などのプロパティ参照です。

一致対象範囲

範囲の開始および終了を入力します。

一致が見つからない場合

一致が見つからない場合に実行するステップを選択します。

環境エラーの場合

環境エラーが発生した場合に実行するステップまたは実行するアクションを選択します。

次のステップのエントリ

エントリを追加するには、［追加］をクリックします。 エントリを移動または削除するには、［移動］および［削除］を使用します。 エントリを検索するには、［検索］フィールドにテキストを入力します。

次のステップのエントリの列は、以下のとおりです。

有効

エントリを追加する場合にデフォルトでオンになります。エントリを無視するにはオフにします。

name

次のステップのエントリの一意の名前を入力します。

遅延仕様

遅延仕様の範囲を入力します。デフォルトは 1000-10000 です。これは、1,000 から 10,000 ミリ秒のランダムに選択された遅延時間が使用されることを示します。構文は、反応時間仕様と同じ形式です。

条件

この領域は、［一致対象テキスト］フィールドと比較する条件を定義します。

比較タイプ

以下の 5 つのオプションから選択します。

- 文字列内の検索（デフォルト）
- 正規表現
- 前方一致
- 後方一致
- 完全一致

次のステップ

ステップ名。

条件

エントリの条件文字列を更新します。

電子メールの送信

テスト ケースから電子メール通知を送信するには、電子メールの送信ステップを使用します。

以下のパラメータを入力します。

接続

SMTP メール サーバへの接続パラメータが含まれるアセット (P. 77)を指定します。 ドロップダウン リストから接続を選択します。アセットを追加するには、［新規アセットの追加］をクリックします。

送信元

電子メール送信者を示します。 電子メール アドレスまたはプロパティを入力します。

終了

電子メール受信者を示します。 複数の受信者に送信するには、電子メール アドレスの間にカンマを使用します。 プロパティ、またはプロパティのリストを使用することもできます。

例：

```
{{Ops_email}},{{QA_email}},InfoTechnology@company.com
```

Cc

第 2 の電子メール受信者を示します。 複数の受信者に送信するには、電子メール アドレスの間にカンマを使用します。プロパティ、またはプロパティのリストを使用することもできます。

Bcc

第 3 の電子メール受信者を示します。 複数の受信者に送信するには、電子メール アドレスの間にカンマを使用します。 第 3 の受信者が、その他の受信者によって特定されることはありません。 プロパティ、またはプロパティのリストを使用することもできます。

サブジェクト

電子メールの件名を指定します。 ［件名］フィールドには、プロパティ（複数可）を使用できます。

メッセージ

電子メールの本文を指定します。HTML がサポートされます。電子メール本文でプロパティとしてバッファ済みイメージを使用すると、イメージはメッセージに埋め込まれます。[メッセージ]フィールドには、プロパティ（複数可）を使用できます。 Selenium 統合ステップの場合、**selenium.last.screenshot** プロパティを使用して、Selenium 統合ステップのスクリーンショットを電子メール メッセージに埋め込むことができます。

添付

電子メールへの添付ファイルのリストが表示されます。  をクリックして、電子メールに添付するファイルの名前とパスを参照します。

環境エラーの場合

環境エラーが発生した場合に実行するステップまたは実行するアクションを選択します。

電子メール接続をテストするには、［テスト メールの送信］をクリックします。

電子メールの送信ステップには、「*電子メールの送信ステップ*」という命名規則を使用したデフォルトの名前があります。 デフォルトのステップ名を別のステップが使用する場合、DevTest は、このステップ名に番号を追加して一意にします。 ステップ名は、いつでも変更できます。

外部サブプロセス ステップ

以下のステップが使用できます。

外部コマンドの実行 (P. 315)
ファイル システム スナップショット (P. 318)
サブプロセスの実行 (P. 319)
JUnit テスト ケース スイート (P. 321)
ファイルの読み取り（ディスク、URL、またはクラスパス） (P. 322)
外部 - FTP ステップ (P. 324)

外部コマンドの実行

外部コマンドの実行ステップを使用すると、外部プログラム（オペレーティング システム スクリプト、オペレーティング システム コマンド、実行可能ファイルなど）を実行し、そのコンテンツをフィルタまたはアサーション作成のためにキャプチャできます。

外部プログラムの構文は、オペレーティング システムによって異なります。

外部コマンドの実行エディタでは、以下のパラメータを入力します。

ディレクトリから実行

外部コマンドが実行される場合に、カレント ディレクトリと見なされるディレクトリ。 ディレクトリがテストを実行しているシステムに存在しない場合、DevTest は、ディレクトリを作成します（ファイル システムの権限に従う）。 ディレクトリが存在せず、作成できない場合、ステップは失敗します。

タイム アウト（秒）

タイムアウトの場合に実行されるよう定義されたステップに転送するまでの待機時間。

タイムアウトの場合

タイムアウト値までに外部コマンドの実行が完了しない場合に実行されるステップ。

環境エラーの場合

環境エラーが発生した場合に実行するステップまたは実行するアクションを選択します。

出力エンコーディング

デフォルトの UTF-8 エンコーディングを使用するか、またはドロップダウン リストから代わりのエンコーディングを選択します。［自動検出］を選択し、エンコーディング タイプを選択させることもできます。

プロパティを許可

このチェック ボックスは、プロパティが以下の 4 つのパラメータを許可するかどうかを決定します。 このオプションにより、コマンド エディタ インターフェースの外観が変更されます。

［プロパティを許可］チェック ボックスをオフにしたまま、5 つのチェック ボックスが使用可能です。 ここでの選択は、パラメータを選択するかしないかだけです。

完了まで待機

このチェック ボックスをオンにすると、ステップは実行が完了するまで待機するので、結果をフィルタまたはアサートできます。このチェック ボックスをオフにした場合、フィルタおよびアサーションは実行されます。ただし、実行したコマンドの結果を待たずに実行されます。

テスト終了時に強制終了

［完了まで待機］チェック ボックスがオフの場合、このチェック ボックスをオンにすると、テスト ケースが完了した後にプロセスが強制終了されます。 このパラメータは、テスト ケースの実行中にプロセスを実行して、その後にプロセスをシャット ダウンします。 プロパティには、起動されたコマンドのプロセス ID が含まれます。

プロセス生成

コマンドを実行するオペレーティング システムに、プロセスを作成します。 このパラメータは以下の場合に役立ちます。

- バックグラウンドで長時間実行されるプロセスが必要な場合
- 環境変数の新しいセットが設定され、DevTest によって設定されたものが環境にないことを確認する必要がある場合

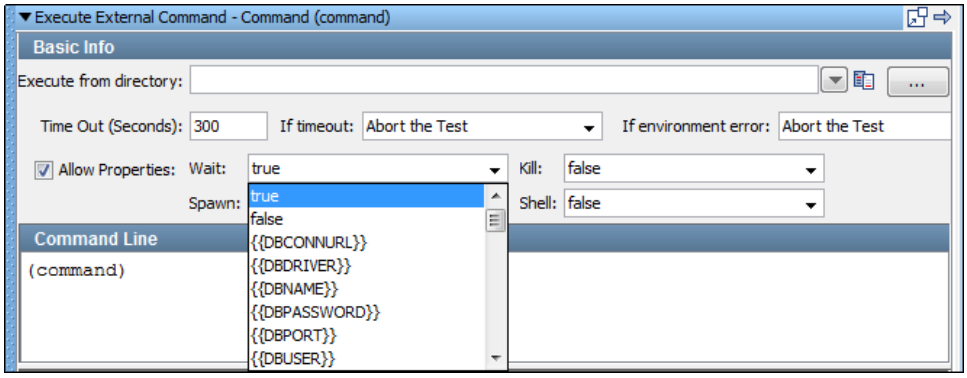
シェルで実行

システム シェルのコマンド ラインのコンテンツを実行します。出力ストリームをファイルまたはその他のコマンドにリダイレクト（パイプ）するようなシェル プロセス機能を使用する必要がある場合、このオプションが必要です。 システムに応じて、このオプションは dir や ls などのシステム コマンドを実行するために必要な場合があります。 Windows オペレーティング システムの場合は、このチェック ボックスをオンにする必要があります。

環境変数に追加

環境変数がステップで定義される場合、DevTest は、これらの変数のみが定義される環境を作成する代わりに、変数を既存の環境に追加します。

［プロパティを許可］チェック ボックスをオンにすると、上記に示したものと同じ機能を持つプルダウン メニューが表示されますが、各パラメータはプロパティになります。



コマンド ライン

外部コマンドは、通常、シェル スクリプトまたはバッチ ファイルとして記述されている単一のコマンドです。 ［シェルで実行］オプションもオンにした場合、複数のコマンドを実行できます。 コマンド文字列は、実行しているオペレーティング システムで有効である必要があります。

環境変数

既存の環境変数を新しい環境変数で上書きすることができます 何も入力しない場合は、既存の環境変数がコマンドで使用されます。環境変数を定義する場合、DevTest を起動するために使用した環境変数の代わりに新しい変数のセットが使用されます。

終了コード

テストの結果が、完了したプロセスの終了コードに基づいたものになります。 プロセスがこのコードで終了する場合、実行するステップに対応する終了コードのカンマ区切り文字列を入力します。

コマンドをテストするには、［実行］をクリックします。

これで、コンテンツのフィルタおよびコンテンツへのアサーションの追加を行えます。

外部コマンドの実行ステップには、「コマンド コマンドの最初の単語」という命名規則を使用したデフォルトの名前があります。 デフォルトのステップ名を別のステップが使用する場合、DevTest は、このステップ名に番号を追加して一意にします。ステップ名は、いつでも変更できます。

ファイル システム スナップショット

ファイル システム スナップショット ステップでは、オペレーティング システムに依存しない形式でディレクトリ内のファイルをリストできます。

単一のファイル、ディレクトリ内のすべてのファイル、またはディレクトリ ツリーのすべてのファイルをリストできます。

ステップ エディタを開くには、ステップをクリックします。

以下のパラメータを入力します。

ディレクトリから実行

パス名を入力するか、ファイルまたはディレクトリを参照します。

サブディレクトリ再帰

サブディレクトリを含む完全なディレクトリ ツリーが必要な場合は選択します。

ファイル サイズを含める

ファイル サイズをリストする場合に選択します。

日/時間を含める

最終変更日をリストする場合に選択します。

環境エラーの場合

環境エラーが発生した場合に実行するステップまたは実行するアクションを選択します。

ファイル システムのスキャンを実行および開始するには、［今すぐ実行］をクリックします。

これで、コンテンツのフィルタおよびコンテンツへのアサーションの追加を行えます。

サブプロセスの実行

サブプロセスの実行ステップでは、単一のステップとしてサブプロセス テスト ケースをコールすることができます。

このステップは、サブプロセスをコールして出力を受け取るために使用します。 このステップは、通常、特定の機能が多数のテスト ケースで実行される場合に使用します。 たとえば、特定の検証が常に同じように動作する場合があります。そのため、サブプロセスが検証を実行するために作成され、さまざまなテスト ケースに追加されます。

詳細については、「サブプロセスの作成」を参照してください。

以下のパラメータを入力します。

サブプロセス

プルダウン メニューからサブプロセスを選択します。

プロパティの完全展開

パラメータにネストされたプロパティがある場合、サブプロセスに送信する前にプロパティを完全に展開します

HTTP Cookie を送信

選択すると、サブプロセスに Cookie が転送されます。

HTTP Cookie を取得

サブプロセスから HTTP Cookie を取得します。

環境エラーの場合

環境エラーが発生した場合に実行するステップまたは実行するアクションを選択します。

サブプロセスの［ドキュメント］領域に入力されているドキュメントが表示されます。

［サブプロセスのパラメータ］パネルには、サブプロセスが必要とするパラメータのリストが含まれます。 これらのキーおよび値は、現在のテスト ケースに存在する必要があります。必要に応じて、［値］列を編集し、正しい値を入力します。

［結果プロパティ］パネルは、サブプロセスで生成されるプロパティをすべてリスト表示します。 サブプロセスから返されるプロパティを選択します。 これらのプロパティはテスト ケースで使用されます。 戻り値は複数使用できます。

このステップが実行される場合、単一のステップとして実行されているように表示されます。 対話型テスト ラン（ITR）ユーティリティは、ステップの実行中に起動されたイベントを表示します。 イベントの短い名前は、現在のステップの名前とイベントが起動されたサブプロセス ステップの名前の組み合わせです。

アサーションがサブプロセスでトリガされた場合、該当するイベントが対話型テスト ラン（ITR）ユーティリティの［テスト イベント］タブに表示されます。

サブプロセスの実行ステップには、「サブプロセス サブプロセス名」という命名規則を使用したデフォルトの名前があります。 ステップ名は、いつでも変更できます。

JUnit テスト ケース スイート

JUnit テスト ケース/スイートの実行ステップでは、DevTest ステップで JUnit テスト ケースまたは JUnit テスト スイートを実行できます。 JUnit テストが成功すると、テスト ステップが成功します。 失敗の後、別のテスト ステップにリダイレクトできます。

前提条件： JUnit テストがクラスパスに存在する必要があります。 それをホット デプロイ ディレクトリにドロップします。

以下のパラメータを入力します。

テスト クラス

JUnit テスト クラスまたはテスト スイート クラスのパッケージ名を入力します。 ［参照］ボタンを使用して、クラスパスを参照できます。 この方法で、クラスがクラスパスに存在することも確認します。

環境エラーの場合

JUnit テストが失敗した場合にリダイレクトするステップを選択します。

クラス ファイルをロードするには、［ロード］をクリックします。 クラス ツリーが左側のパネルに表示されます。

JUnit テストを実行するには、［実行］をクリックします。 標準の JUnit 結果が右側のパネルに表示されます。

JUnit テスト ケース スイート ステップには、「JUnit テスト ケース スイート テスト クラス名」という命名規則を使用したデフォルトの名前があります。 デフォルトのステップ名を別のステップが使用する場合、DevTest は、このステップ名に番号を追加して一意にします。 ステップ名は、いつでも変更できます。

ファイルの読み取り（ディスク、URL、またはクラスパス）

ファイルの読み取りステップは、ファイル システム、URL、またはクラスパスからファイルを読み取ります。

ファイルは、テスト用のソース データとして使用されます。 このステップは、ファイル名セットのロード データ セットと組み合わせることで、テスト用のソース データを提供できます。

テキスト ファイルまたはバイナリ ファイルを読み取ることができます。ファイルのコンテンツは、必要に応じてプロパティに格納できます。

以下のパラメータを入力します。

ファイル

パス名、URL、またはクラスパスを入力するか、［参照］ボタンを使用してファイルを参照します。

ファイル エンコーディング

デフォルトの UTF-8 エンコーディングを使用するか、またはドロップダウン リストから代わりのエンコーディングを選択します。［自動検出］を選択して［検出］ボタンをクリックし、<ldtF> にエンコーティング タイプを選択させます。

プロパティ キー

ファイルのコンテンツを格納するプロパティ名を入力します（オプション）。

byte[] としてロード

バイトの配列としてコンテンツをロードするには、このチェックボックスをオンにします。この機能は、Web サービス実行パラメータで binaryData 型として使用されるファイルを入力する場合に役立ちます。

環境エラーの場合

ファイルの読み取りテストが失敗した場合にリダイレクトするステップを選択します。

文字として表示

このチェック ボックスをオンにしない場合、コンテンツは 16 進エンコード バイトとして表示されます。 このチェック ボックスは、［byte[] としてロード］チェック ボックスがオンになっている場合にのみ表示されます。

ファイルをロードおよび表示するには、［ロード］をクリックします。これで、コンテンツのフィルタおよびコンテンツへのアサーションの追加を行えます。

注: バイナリ ファイルをロードする際にバイトとしてロードすることを選択しない場合、DevTest はデータを文字（大部分は判読不能）に変換します。また、ステップ応答は文字列です。

ファイルの読み取りステップには、「読み取りファイル <ファイル名>」という命名規則を使用したデフォルトの名前があります。 デフォルトのステップ名を別のステップが使用する場合、DevTest は、このステップ名に番号を追加して一意にします。ステップ名は、いつでも変更できます。

外部 - FTP ステップ

FTP ステップでは、FTP プロトコルを使用してファイルを送受信できます。FTP 情報、ユーザ名、およびユーザ パスワードを入力した後、ファイルをアップロードまたはダウンロードできます。

必要に応じて、FTP ステップ エディタ ウィンドウを左にドラッグして、［今すぐ実行］ボタンを表示することができます。

以下のパラメータを入力します。

ホスト

FTP サーバのホスト名を入力します（プロトコルなし）。

ポート

FTP サーバ アクセス用のポートを入力します。 ポートはオプションです。デフォルトのポートは 21 です。

ユーザ

FTP サーバ アクセス用のユーザ ID を入力します。

パスワード

FTP サーバ アクセス用のパスワードを入力します。

方向

データ フローが、アップロードかダウンロードかを示します。

モード

パッシブ FTP またはアクティブ FTP を指定します。あるいは、パッシブまたはアクティブを示すプロパティを入力します。

転送タイプ

ファイル転送タイプ（バイナリまたは ASCII）を選択します。あるいは、バイナリまたは ASCII を示すプロパティを入力します。

ホスト パス

ソース ファイルのパス（FTP サーバまたはローカル コンピュータのいずれか）を入力します。

ローカル パス

送信先ファイルのパス（FTP サーバまたはローカル コンピュータのいずれか）を入力します。

注: ［ホスト パス］はリモート コンピュータ上のファイルのパスです。 ［ローカル パス］は、常に、DevTest を実行しているローカル コンピュータ上のファイルのパスです。 ファイルをアップロードする場合は、［ローカル パス］から［ホスト パス］にファイルをアップロードします。ファイルをダウンロードする場合は、［ホスト パス］から［ローカル パス］にファイルをダウンロードします。

環境エラーの場合

環境エラーが発生した場合に実行するステップまたは実行するアクションを選択します。

送信/受信アクションを開始するには、［今すぐ実行］をクリックします。

ダウンロードからの応答はファイル自体です。 成功したアップロードからの応答は「success」という文字列です。

注: 同じ名前のファイルが存在する場合、警告メッセージなしで上書きされます。

FTP ステップには、「FTP 操作（put または get） ホスト名」（例： FTP get ftp.download.com）という命名規則を使用したデフォルトの名前があります。 ステップ名は、いつでも変更できます。

JMS メッセージング ステップ

以下のステップが使用できます。

JMS メッセージング（JNDI） (P. 326)
JMS メッセージング - メッセージ コンシューマ (P. 337)
JMS 送信/受信ステップ (P. 341)

JMS メッセージング（JNDI）

JMS メッセージング （JNDI）ステップでは、トピックおよびキューに対してメッセージを送受信できます。また、既存のメッセージを受信、変更、転送できます。 使用可能なキューおよびトピックのリストは、JNDI を使用して参照できます。DevTest が読み取ることができるクライアント ライブラリを提供します。

空、テキスト、オブジェクト、バイト、メッセージ、マップ済み（拡張）など、一般的なメッセージ タイプがすべてサポートされています。

JMS メッセージング （JNDI）ステップは、メッセージング要件にかかわらず、単一のエディタを使用して設定されます。入力オプションはメッセージング要件によって異なります。 エディタは有効な設定のみを許可します。そのため、一部の機能を有効にすると、その他の機能が非アクティブになる場合があります。

JMS メッセージング（JNDI）ステップには、「JMS パブリッシュ キュー名 パブリッシュ」という命名規則を使用したデフォルトの名前があります。パブリッシュ キュー名がない場合、デフォルトのステップ名は「JMS サブスクライブ キュー名 サブスクライブ」です。 デフォルトのステップ名を別のステップが使用する場合、DevTest は、このステップ名に番号を追加して一意にします。 ステップ名は、いつでも変更できます。

前提条件： このアプリケーションと一緒に DevTest を使用するには、1 つ以上のファイルを DevTest で使用可能にする必要があります。詳細については、「*管理*」の「サードパーティ ファイル要件」を参照してください。

パラメータ要件： このステップには、テスト中のアプリケーションで使用される接続パラメータ、およびキューまたはトピック名が必要です。環境に応じて、その他のパラメータが必要となる場合があります。 これらのパラメータは、アプリケーション開発者から入手します。 ほとんどの場合、これらの必須パラメータの一部を取得するために、サーバ リソースを参照できます。

examples プロジェクトの **jms.tst** テスト ケースには、このセクションで説明するステップが示されています。

jms.tst テスト ケースは、JMS のパブリッシュ/サブスクライブ ステップを使用して、メッセージの送信および一時キューのリスンを行います。サーバ上の メッセージ駆動型 Bean （MDB）は、メッセージを処理し、一時キューのメッセージをドロップします。 メッセージ タイプはテキストです。 メッセージは、XML エレメントにプロパティを動的に挿入することにより作成される XML ペイロードです。 プロパティは **order_data** データセットから読み取られます。応答メッセージを受信すると、JMS メッセージからの XML がプロパティに格納されます。 次のステップは、順番（ID）を検証するアサーションを実行します。 この確認で true をアサートした後、既存のメッセージ オブジェクトは変更され、メッセージは別の JMS 送信先に送信されます。

jms.tst テスト ケースは、メッセージが複数のメッセージング サービス バックボーンを介して送受信される場合に、メッセージをリスンおよびインターセプトする方法を示します。 お使いのコンピュータ上のデモ サーバに対してこのテスト ケースを実行できます。 アプリケーション バックエンドはそこで使用可能です。

JMS メッセージング（JNDI）ステップ エディタには以下のタブが含まれます。

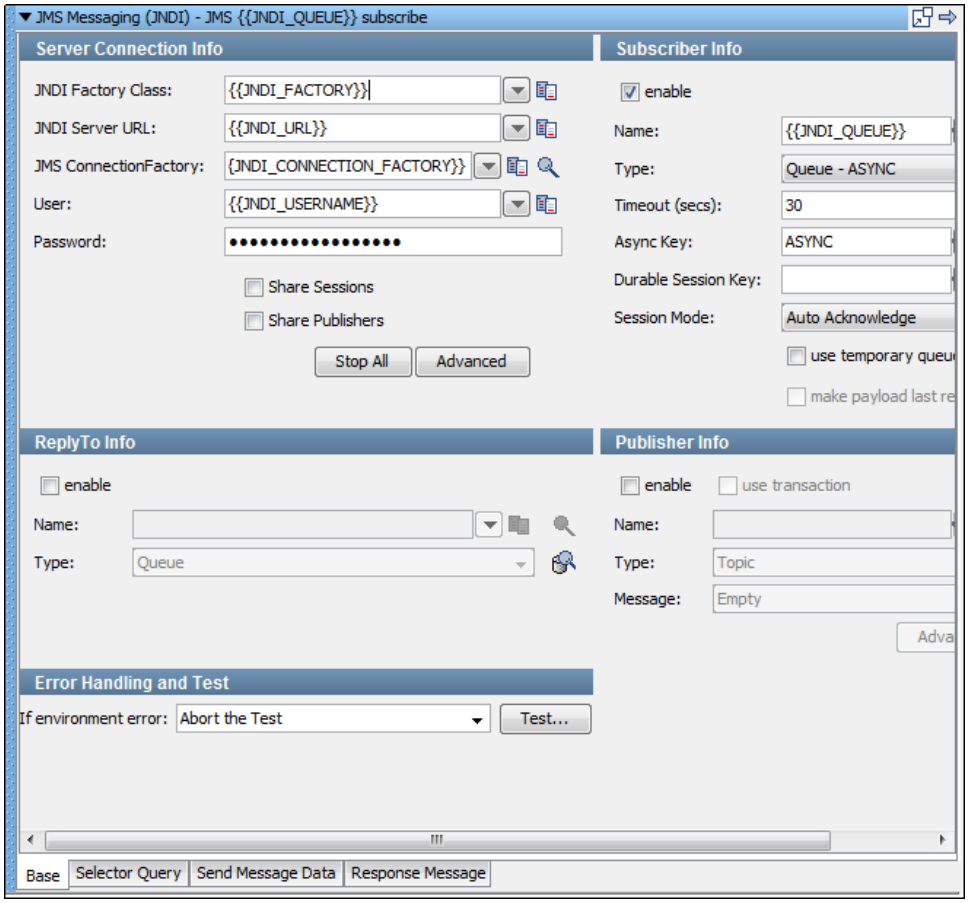
- ［ベース］タブでは、接続およびメッセージング パラメータを定義します。
- ［セレクタ クエリ］タブでは、キュー上のメッセージをリスンする場合に実行されるセレクタ クエリを指定できます。
- ［メッセージ データの送信］タブでは、メッセージ コンテンツを作成します。
- ［応答メッセージ］タブでは、応答メッセージを POST します。

[ベース]タブ

［ベース］タブでは、接続およびメッセージング パラメータを定義します。

以下の図は、［ベース］タブを示しています。 このタブは、以下のセクションで構成されています。

- サーバ接続情報
- サブスクライバ情報
- ReplyTo 情報
- パブリッシャ情報
- エラー処理およびテスト



［サブスクライバ情報］、［パブリッシャ情報］、および［返信先情報］を有効および無効にするには、各セクションの左上隅にある［有効］チェック ボックスを使用します。 このオプションにより、ステップをパブリッシュ ステップ、サブスクライブ ステップ、またはその両方に設定できます。 また、オンにすると、ステップに JMS の返信先コンポーネントを含めることもできます。

テスト ステップの設定が完了したら、［エラー処理およびテスト］セクションの［テスト］をクリックして設定をテストします。

サーバ接続情報

［ベース］タブの［サーバ接続情報］セクションに JNDI 情報を入力します。

テスト中のアプリケーションの変更を容易にするには、これらの値を設定内のプロパティでパラメータ化します。 上記の図は、この方法の例を示しています。

テスト中のシステムに対して、以下のパラメータを使用できます。

JNDI ファクトリ クラス

JNDI プロバイダのコンテキスト ファクトリの完全修飾クラス名。

JNDI サーバ URL

JNDI サーバに接続するための URL。 URL の形式は、使用されている特定の JNDI プロバイダによって異なります。

JMS 接続ファクトリ

［検索］ を使用して、サーバ上の利用可能なリソースを参照します。JMS 仕様に従って、このステップの実行に使用する接続ファクトリを選択または入力します。

プルダウン メニューには、これらの値の一般的な例またはテンプレートが含まれます。

ユーザおよびパスワードはオプションとなる場合があります。

ユーザ

JNDI プロバイダに接続し、接続ファクトリのハンドルを取得するためのユーザ名。

パスワード

JNDI プロバイダに接続し、接続ファクトリのハンドルを取得するためのパスワード。

セッションの共有およびパブリッシャの共有

テスト ケース全体で JMS セッションおよびパブリッシャを共有するには、これらのチェック ボックスを使用します。 この方法はオーバーヘッドを低減できますが、通常、JMS クライアントがリソースを解放するため、現実的なシミュレーションとなるとは限りません。 ［パブリッシャの共有］チェック ボックスをオンにすると、［セッションの共有］チェック ボックスもオンになります。セッションを共有しなければ、パブリッシャを共有できません。これらのパラメータの詳細については、「*Deliberate Delays in VSE*」ナレッジ ベース記事を参照してください。

すべて停止

設計時にすべてのリスナを停止します。 一部のリスナは切り離すことができますが、引き続きメッセージを処理します。 メッセージを処理する場合は、テスト ケースを作成することは困難です。

詳細

接続情報と一緒に送信されるカスタム プロパティを追加でき、第 2 レベルの認証を設定できるパネルを表示します。

注: ［サーバ接続情報］セクションの［ユーザ］および［パスワード］フィールドは、JNDI プロバイダに接続して接続ファクトリのハンドルを取得するためのものです。 ［第 2 レベル認証］タブの［ユーザ］および［パスワード］フィールドは、実際の JMS 接続のハンドルを取得するためのものです。

パブリッシャ情報

メッセージを送信する機能を設定するには、［有効］チェック ボックスをオンにします。

メッセージを送信する場合にコミットを実行するには、［トランザクションの使用］チェック ボックスをオンにします。

以下のパラメータを入力します。

名前

トピックまたはキューの名前。［検索］アイコン を使用して、トピックまたはキュー名の JNDI サーバを参照します。

タイプ

トピックまたはキューを使用する場合に選択します。 どのメッセージがキュー（のみ）で処理されるのを待機しているかを確認するには、このフィールドの右にある［検索］アイコン を使用します。

メッセージ

送信するメッセージのタイプを選択します。 サポートされているタイプは、［なし］、［テキスト］、［オブジェクト］、［バイト］、［メッセージ］、および［マップ済み（拡張）］です。

詳細

メッセージ ヘッダの編集とメッセージ プロパティの追加を行えるパネルを表示します。

サブスクライバ情報

メッセージを受信する機能を設定するには、［有効］チェック ボックスをオンにします。

以下のパラメータを入力します。

名前

トピックまたはキューの名前。［検索］アイコン を使用して、トピックまたはキュー名の JNDI サーバを参照します。

タイプ

トピックまたはキューを使用するかどうか、および同期または非同期モードでリスンするかどうかを選択します。 非同期モードでは、［非同期キー］フィールドにもエントリが必要です。 どのメッセージがキューで消費されるのを待機しているかを確認するには、このフィールドの右にある［検索］アイコン を使用します。

タイムアウト（秒）

メッセージの待機中に中断するまでの時間。 タイムアウトを指定しない場合は 0 とします。

非同期キー

非同期メッセージを識別するために必要な値。 このフィールドは非同期モードにのみ必要です。このフィールドは、非同期メッセージを取得するために後続のメッセージ コンシューマ ステップで使用されます。

持続セッション キー

ここで名前を入力することによって、持続セッションを要求します。 また、そのセッションのキーも指定します。 持続セッションでは、ログアウトした後に再度ログインしても、トピックからのメッセージをすべて受信できます。

セッション モード

使用できるオプションは以下のとおりです。

- 自動確認応答 : 受信した JMS メッセージを JMS クライアント ライブラリがすぐに確認します。
- クライアント確認応答 : JMS クライアントは明示的に JMS メッセージを確認する必要があります。
- トランザクションの使用 : JMS セッションはトランザクションで動作します。 確認応答モードは無視されます。
- 自動（重複 OK） : JMS クライアント ライブラリは不定期で自動的に確認します。 結果として、JMS プロバイダが配信を再試行する前に自動確認応答が到着しない場合、重複したメッセージを受信する可能性があります。

［自動確認応答］、［クライアント確認応答］、［自動確認応答（重複を許可）］には、実質的な違いはありません。 ［クライアント確認応答］では、受信した各メッセージは受信時にすぐ確認されます。 違いは、JMS クライアント ライブラリに実行させる代わりに、確認応答コールが明示的に作成されるという点だけです。 ［自動確認応答（重複を許可）］では、高負荷の場合以外は、［自動確認応答］と動作は同じです。

［トランザクションの使用］オプションは、厳密には確認応答モード設定ではありません。このオプションは、以下の 2 つの理由でリストに含まれています。

- JMS セッションがトランザクションで動作している場合、確認応答モードは無視されます。 メッセージは、セッションのトランザクションをコミットすることによって確認されます。

- ［トランザクションの使用］モードは、現在も JMS からのメッセージの配信の保証を制御する方法です。 メッセージが受信され、セッション トランザクションがコミットされない場合、確認されなかった場合と同じように、メッセージが再送信されます。

  一時キュー/トピックを使用する

  JMS プロバイダに一時キュー/トピックを設定させる場合は、このチェック ボックスをオンにします。一時キュー/トピックが使用される場合、DevTest によって、一時キュー/トピックに送信するメッセージの **JMS ReplyTo** パラメータが自動的に設定されます。返信を送信できるように、一時キュー/トピック機能とパブリッシャを常に一緒に使用する必要があります。一時キュー/トピックを使用する場合、［返信先］セクションは無効です。

  ペイロードを最終応答にする

  ペイロード応答を最終応答にするには、このチェック ボックスをオンにします。

ReplyTo 情報

送信先キュー/トピックを設定するには、［有効］チェック ボックスをオンにします。

アプリケーションが送信先を必要とする場合、このセクションで設定されます。

以下のパラメータを入力します。

名前

トピックまたはキューの名前。［検索］アイコン  を使用して、トピックまたはキュー名の JNDI サーバを参照します。

タイプ

トピックまたはキューを使用する場合に選択します。 どのメッセージがキュー（のみ）で処理されるのを待機しているかを確認するには、このフィールドの右にある［検索］アイコン  を使用します。

エラー処理およびテスト

エラーが発生した場合、［エラー処理およびテスト］セクションはステップにリダイレクトします。

環境エラーの場合

環境エラーが発生した場合に実行するステップまたは実行するアクションを選択します。

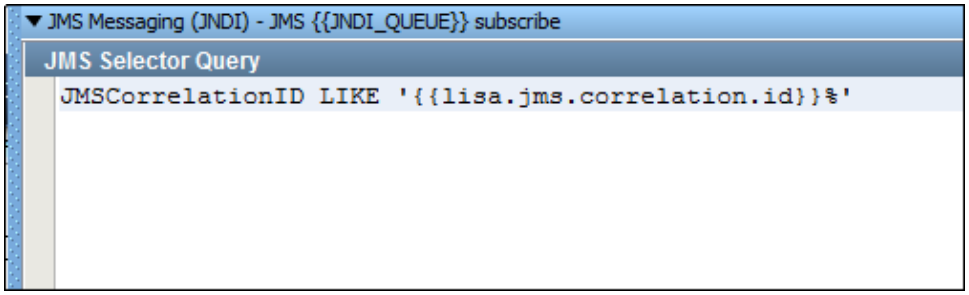
ステップ設定をテストするには、［テスト］をクリックします。

［セレクタ クエリ］タブ

［セレクタ クエリ］タブでは、JMS セレクタ クエリを入力できます。 構文は SQL に準拠しています。 このクエリは SQL92 のサブセットです。

JMS セレクタ クエリは、パブリッシュされたメッセージに対する応答であるキュー上のメッセージをリスンする場合に指定できます。

以下の図は、元のメッセージで送信された **lisa.jms.correlation.id** プロパティに一致する **JMSCorrelationID** を検索するクエリを示しています。

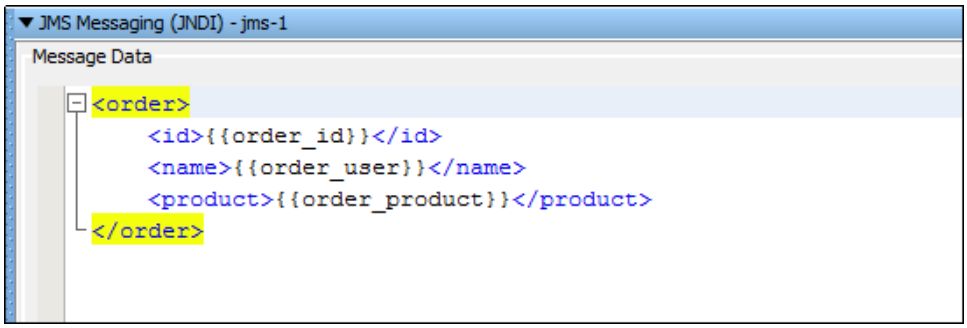


[メッセージ データの送信]タブ

［メッセージ データの送信］タブは、ステップがパブリッシュのために設定されている場合にメッセージを作成する場所です。

テキストを入力できます。または、タブの右下隅の［ファイルからメッセージを読み取り］ボタンを使用して、ファイルからテキストを読み取ることができます。 また、プロパティにテキストを格納できます。その場合、エディタにプロパティ（例：**{{プロパティ名}}**）を配置します。

以下の図は、プロパティを持った XML フラグメントを示しています。 プロパティを使用すると、テストの実行時にメッセージを動的に作成することができます。

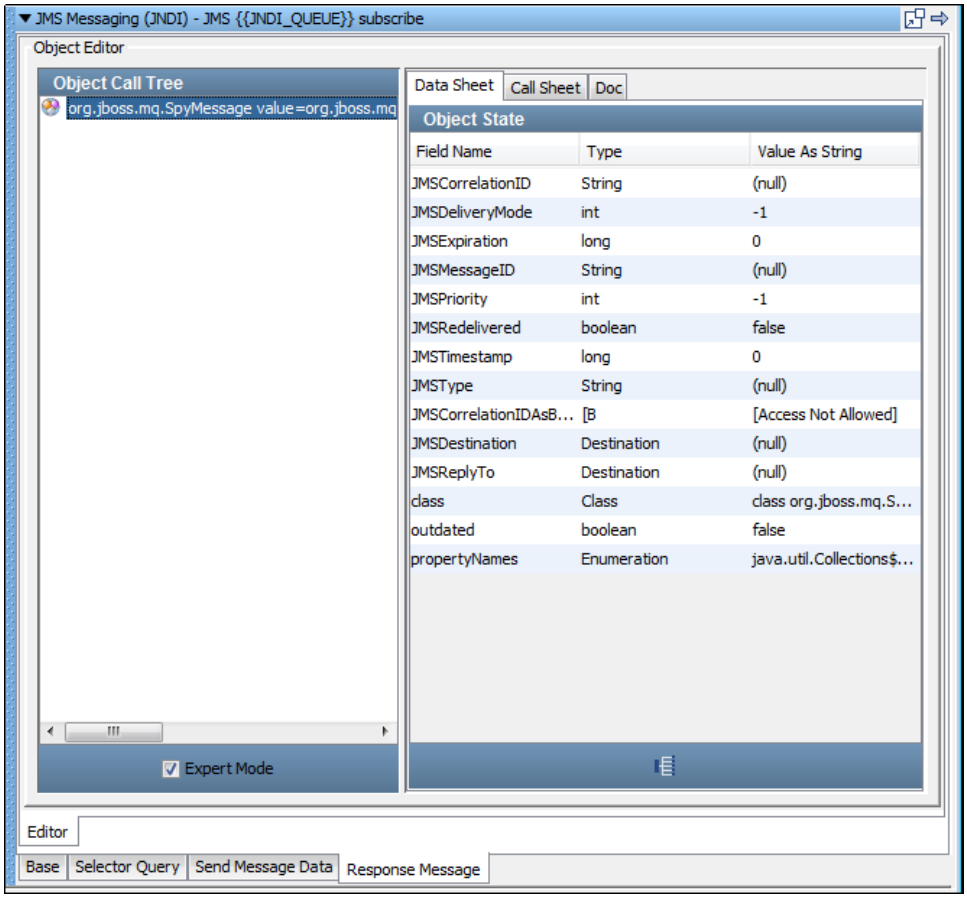


[応答メッセージ]タブ

ステップがサブスクライブのために設定されている場合、［ベース］タブの［テスト］をクリックすると、応答が［応答メッセージ］タブに表示されます。

このタブには、返されるオブジェクトの複合オブジェクト エディタが表示されます。 返されるオブジェクトは、アプリケーション サーバのタイプによって異なります。メッセージ自体に加えて、返されるすべての JMS パラメータにアクセスできます。 オブジェクトは、オブジェクトをその他の Java オブジェクトのように操作できる複合オブジェクト エディタにロードされます。

以下の図は、JBoss オブジェクトからのテキスト応答を示しています。



JMS メッセージング - メッセージ コンシューマ

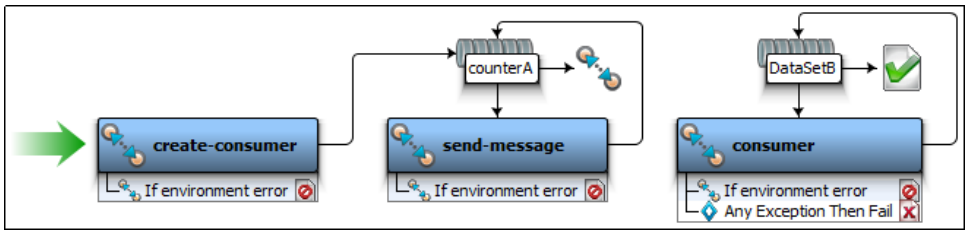
メッセージ コンシューマ ステップでは、テスト ケースで非同期メッセージを処理できます。 このステップは既知のキューまたはトピックに接続でき、このサブスクライバに対してメッセージを POST できます。 一意のキーでユーザを識別します。 キューまたはトピックをすでにサブスクライブしており、メッセージがその送信先にプッシュされている必要があります。

前提条件： サンプル テスト ケースを実行するには、デモ サーバが実行されている必要があります。

パラメータ要件： テスト中のアプリケーションで使用されているキューまたはトピックについての知識。

メッセージ コンシューマ ステップには、「サブスクライブ キュー名でリスン」という命名規則を使用したデフォルトの名前があります。 キュー名を入力する前のデフォルトのステップ名は、「非同期 JMS」です。デフォルトのステップ名を別のステップが使用する場合、DevTest は、このステップ名に番号を追加して一意にします。 ステップ名は、いつでも変更できます。

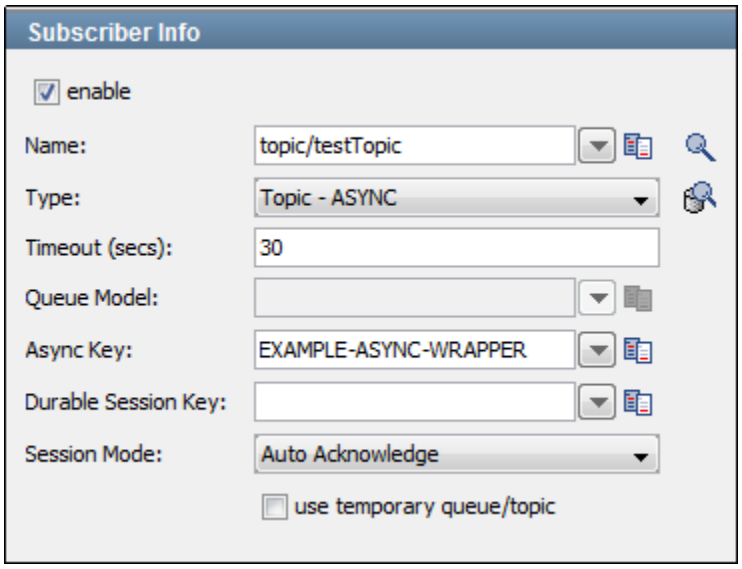
このセクションで説明されていることを示す、examples プロジェクトの **async-consumer-jms.tst** テスト ケースを確認してください。 以下の図は、**async-consumer-jms.tst** テスト ケースを示しています。



create-consumer ステップは、非同期キー（EXAMPLE-ASYNC-WRAPPER）を使用して、非同期メッセージ（topic/testTopic）をサブスクライブします。**send-message** ステップは、キュー（queue/C）に対してメッセージをパブリッシュします。 パブリッシュされるメッセージの数は、データ セット（counterA）を使用して制御されます。 メッセージ コンシューマ ステップには、create-consumer ステップによってサブスクライブされたメッセージを処理する非同期キュー（EXAMPLE-ASYNC-WRAPPER）があります。 処理されるメッセージの数は、データ セット（DataSetB）を使用して制御されます。

注: create-consumer および consumer ステップで指定されている非同期キーは、一致する必要があります。

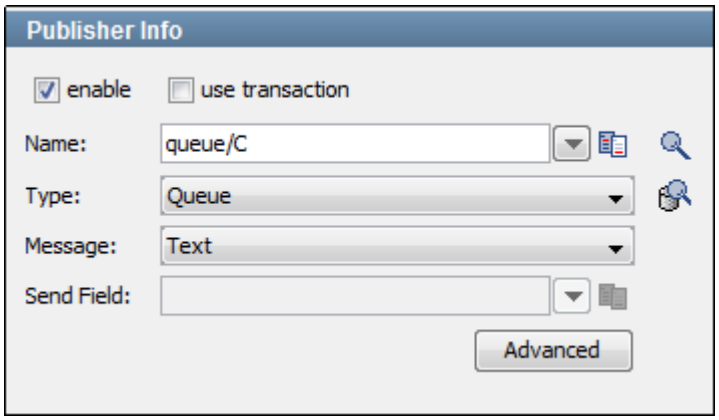
以下の図は、このステップのサブスクライバ セクションの例を示しています。



メッセージをリスン（サブスクライブ）する機能を設定して有効にするには、［有効］チェック ボックスをオンにします。

非同期トピックが［タイプ］フィールドで指定され、［非同期キー］パラメータが定義されていることに注目してください。 このキーは、このステップの入力として必要です。

以下の図は、このステップのパブリッシャ セクションの例を示しています。



メッセージを送信（パブリッシュ）する機能を設定するには、［有効］チェック ボックスをオンにします。 メッセージを送信する場合にコミットを実行するには、［トランザクションの使用］チェック ボックスをオンにします。

以下のパラメータを入力します。

名前

トピックまたはキューの名前。［検索］アイコン を使用して、トピックまたはキュー名の JNDI サーバを参照します。

タイプ

トピックまたはキューを使用する場合に選択します。 どのメッセージがキュー（のみ）で処理されるのを待機しているかを確認するには、このフィールドの右にある［検索］アイコン を使用します。

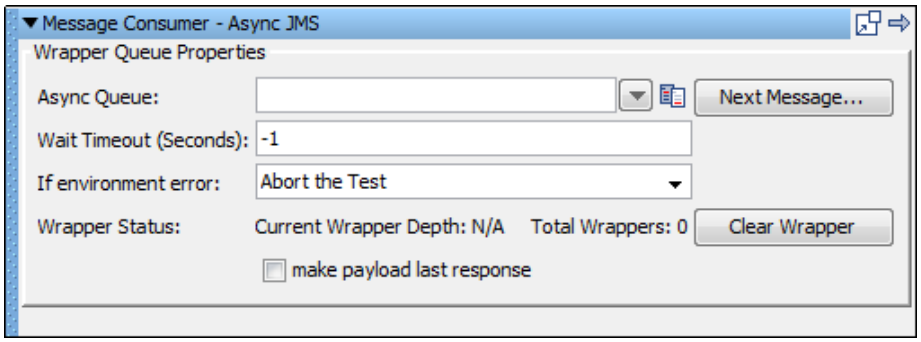
メッセージ

送信するメッセージのタイプを選択します。 サポートされているタイプは、［なし］、［テキスト］、［オブジェクト］、［バイト］、［メッセージ］、および［マップ済み（拡張）］です。

詳細

メッセージ ヘッダの編集とメッセージ プロパティの追加を行えるパネルを表示します。

数式 1: JMS メッセージング - メッセージ コンシューマ ステップ



以下のパラメータを入力します。

非同期キュー

前のサブスクライバ ステップ（EXAMPLE-ASYNC-WRAPPER）で指定された［非同期キー］パラメータを入力または選択します。 これらの名前は一致する必要があります。

タイムアウトの待機時間（秒）

次のメッセージを待機する間隔を秒単位で入力します。

環境エラーの場合

環境エラーが発生した場合に実行するステップまたは実行するアクションを選択します。

ラッパー ステータス

ラッパー ステータスには、以下の 2 つの出力ステータス値があります。

- 現在のラッパーの深さ : 現在のラッパーで読み取る残りのメッセージの数。
- 合計ラッパー数 : ラッパーの数（送信先）。

ペイロードを最終応答にする

- ステップでペイロードを最終応答にするには、このオプションをオンにします。

複数のメッセージが待機している場合、［次のメッセージ］をクリックすることにより読み取ることができます。 複合オブジェクト エディタにメッセージが表示されます。

これで、このオブジェクトを操作することができます。

ラッパーは、非同期トピックおよびキューからの応答を保持するための FIFO リストです。 ラッパーは、アプリケーションが後で処理する応答を配置するべき場所を提供します。 メッセージは、（このメッセージ コンシューマ ステップでの）後続の処理のためにこのリストで待機します。

JMS 送信/受信ステップ

JMS 送信/受信ステップでは、任意の JMS 互換のメッセージ サーバに接続して、以下のアクションを実行できます。

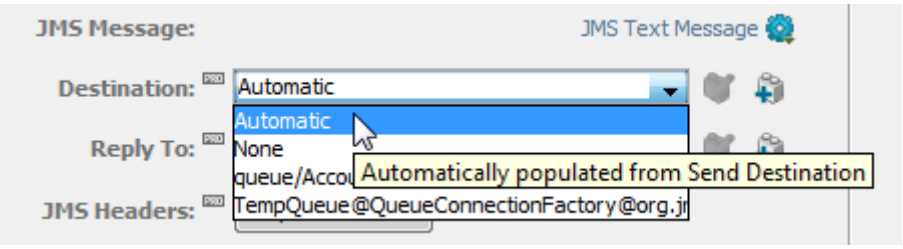
- 要求メッセージの送信
- 応答メッセージの受信

ステップ エディタには、基本パラメータおよび詳細パラメータがあります。 詳細パラメータを表示するには、エディタの上部の［PRO］をクリックします。

各パラメータには、パラメータの目的を説明するツールヒントがあります。

一部のパラメータには、オプションとして［自動］という値が含まれます。この値は、実際の設定が別のパラメータから取得されることを示します。ドロップダウン矢印をクリックし、［自動］という値の上にマウス ポインタを置くと、ツールヒントに別のパラメータの名前が表示されます。

以下の図では、ツールヒントに［送信先］パラメータの値が［送信先］パラメータから自動的に入力されることが示されています。



一部のパラメータでは、値としてプロパティを入力できるようにエディタを変更できます。

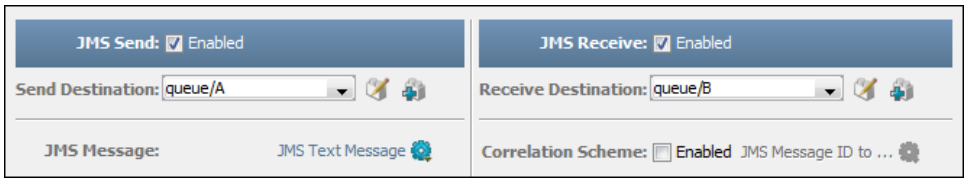
値の個別のセットを提供するパラメータを使用すると、値を直接入力できるようにエディタを変更できます。たとえば、［JMS 配信モード］パラメータには、［永続性］と［非永続性］の 2 つの値があります。 JMS API では、これらの値は数字 2 および 1 にそれぞれマップされます。値を直接入力するには、直接入力エディタに切り替えます。 直接入力エディタを使用すると、値の正式なリストにない値を指定できます。

送信と受信

JMS 送信/受信ステップには、JMS 送信操作と JMS 受信操作を設定するための個別の領域があります。

ステップ エディタで 2 つの送信先 (P. 69)を指定する必要があります。1 つは送信操作用、もう 1 つは受信操作用です。送信先は同じでも異なっていてもかまいません。 リストにはアクティブな設定から JMS 送信先アセットが入力されます。

以下の図は、送信操作と受信操作の送信先が異なる例を示しています。



応答を受信せずにメッセージを送信するには、JMS 受信操作を無効にします。最初にメッセージを送信しないでメッセージを待機するには、JMS 送信操作を無効にします。

両方の操作を無効にすると、ステップでは何も行われません。

デフォルトでは、受信操作には同期コンシューマ (P. 69)が使用されます。詳細パラメータには、非同期コンシューマを指定するチェック ボックスが含まれます。

アセットにはそれぞれランタイム スコープがあります。 JMS 送信/受信ステップでは、送信操作および受信操作の最小ランタイム スコープを指定できます。 スコープ パラメータは詳細パラメータです。

JMS メッセージ

JMS 送信/受信ステップでは、送信または受信されるメッセージを設定できます。

以下のメッセージ タイプをサポートしています。

- テキスト
- バイト
- ストリーム
- マップ
- 空

［コンテンツ］領域でメッセージのペイロードを設定します。

［コンテンツ］領域で使用可能なエディタはメッセージ タイプによって異なります。

テキスト メッセージには、以下のエディタがあります。

- プレーン テキスト
- JSON
- EDI
- XML

バイト メッセージには、以下のエディタがあります。

- バイナリ
- グラフィック イメージ

ストリーム メッセージには、オブジェクトを追加できるエディタがあります。

マップ メッセージには、キー/値ペアを追加できるエディタがあります。

空メッセージには、エディタはありません。

メッセージ タイプを変更する場合、ステップは既存のペイロードを変換できるかどうかを確認します。 変換できない場合、ステップは既存のペイロードを破棄します。

JMS では、メッセージにヘッダのセットおよびオプションのカスタム プロパティのセットが含まれます。 ステップ エディタでこれらのヘッダおよびプロパティを設定できます。

JMS 相関スキーム

同時に実行されている 2 つのクライアントを持ったメッセージ サービスについて考えてみましょう。 クライアントはそれぞれ同じ要求キューに要求メッセージを送信できます。 サービスは任意の順に要求を処理し、応答キューに応答メッセージを送信できます。 各クライアントはどのように応答を受信し、ほかのクライアント宛の応答は受信しないのでしょうか。

最も一般的な方法は、要求にいくつかのタイプの識別子を組み込み、応答にその識別子をコピーすることです。 この識別子は、*相関 ID* と呼ばれます。 クライアントはそれぞれ一意の相関 ID を使用します。 各クライアントがそのクライアント用の相関 ID が含まれる応答のみを受信できるようにするために使用されるメカニズムもあります。

JMS 送信/受信ステップで相関を有効にできます。 以下の相関スキームが使用可能です。

- JMS 相関 ID
- JMS メッセージ ID から相関 ID
- JMS ペイロード

相関スキームは、以下のことに基づいて異なります。

- 相関 ID によって応答を正しいクライアントにルーティングする方法
- 要求および応答メッセージ内で相関 ID が配置されている場所

JMS 相関 ID

ほとんどのメッセージング プラットフォームには、相関 ID フィールドがあります。このスキームでは、クライアントは一意の ID を生成し、相関 ID フィールドにその ID が含まれる要求メッセージを送信します。 サービスは、応答メッセージを送信する前に、相関 ID を要求メッセージから応答メッセージにコピーします。 クライアントは、元の要求と同じ相関 ID が含まれる応答メッセージをリスンします。

デフォルトでは、相関 ID は自動的に生成されます。 値を手動で指定するには、［手動値］フィールドを使用します。

［ID の再利用］リストは、新しいトランザクションそれぞれに新しい相関 ID を生成するか、テスト期間で同じ相関 ID を使用するかを示します。

JMS メッセージ ID から相関 ID

このスキームは、JMS 相関 ID スキームに類似しています。

ほとんどのメッセージング プラットフォームには、メッセージの一意の識別子が含まれるメッセージ ID フィールドもあります。 メッセージ ID は自動的に生成されます。

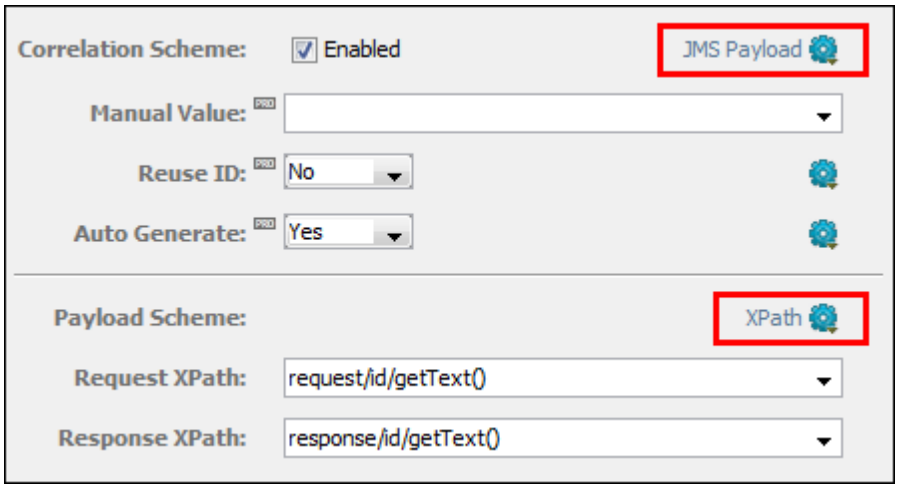
サービスは、応答メッセージを送信する前に、要求メッセージのメッセージ ID を応答メッセージの相関 ID にコピーします。 クライアントは、元の要求のメッセージ ID に一致する相関 ID が含まれる応答メッセージをリスンします。

メッセージ ID を手動で指定することはできません。 メッセージ ID を再利用することはできません。

JMS ペイロード

このスキームは、要求および応答メッセージのペイロードに埋め込まれている相関 ID に基づいています。 このスキームには、相関 ID を埋め込む方法を制御する*ペイロード スキーム*が関連付けられています。 デフォルトのペイロード スキームでは、メッセージから相関 ID を取得するための XPath 式を定義できます。

以下の図は、JMS ペイロード相関スキームと XPath ペイロード スキームの例を示しています。



［ID の再利用］リストは、新しいトランザクションそれぞれに新しい相関 ID を生成するか、テスト期間で同じ相関 ID を使用するかを示します。

JMS ペイロード相関スキームを同じキュー上のその他の 2 つの相関スキームと混在させることはできません。1 つのリスナが JMS ペイロード相関スキームを使用してキューでリスンしている場合、そのキュー上のリスナはすべて JMS ペイロードを使用している必要があります。 ただし、JMS ペイロード相関スキームのペイロード スキームとして、JMS 相関 ID および JMS メッセージ ID から相関 ID を使用できます。

JMS 送信/受信ステップのテスト

エディタで JMS 送信/受信ステップの機能を確認できます。

実行ログは、バックグラウンドで実行されるアクティビティの概要を示します。 たとえば、実行ログ内の以下の行は、さまざまな JMS クライアント アセット (P. 69)の作成を示しています。

```
Creating JMS Connection
Starting JMS Connection
Creating JMS Session
Performing JNDI lookup with name: queue/B
Creating JMS Consumer on Queue B
Performing JNDI lookup with name: queue/A
Creating JMS Producer
```

以下の手順に従います。

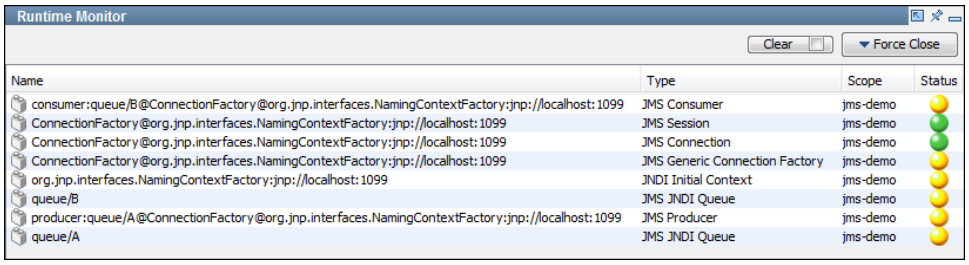
1. ステップ エディタで、緑色の［実行］ボタンをクリックします。
2. ステップを実行している間にキャンセルするには、［キャンセル］をクリックします。

   操作が完了すると、［応答］タブに受信した応答が表示されます。
3. ステップのアクティビティを表示するには、［実行ログ］をクリックします。
4. アセット インスタンスをモニタする (P. 348)には、［ランタイム モニタ］をクリックします。
5. 送信された要求を表示するには、［要求］タブをクリックします。

キャッシュされたアセット インスタンスのモニタおよびクローズ

JMS 送信/受信ステップをテスト (P. 347)する場合、［応答］タブのランタイム モニタによってアセット インスタンスをモニタできます。 アセットは手動で閉じることもできます。

以下の図は、ランタイム モニタの例を示しています。



デフォルトでは、ランタイム モニタは閉じられたアセットを自動的に削除します。 以下の手順で、この動作を無効にします。

以下の手順に従います。

1. ［応答］タブで、［ランタイム モニタ］をクリックします。
2. ［クリア］ボタン内に表示されるチェック ボックスをオンにします。
3. ステップを再実行します。

   ランタイム モニタにステップの実行中に作成されたアセットが表示されます。
4. 以下の情報を確認します。
   a. 名前： アセットの名前。
   b. タイプ： アセットのタイプ。
   c. スコープ： アセットのランタイム スコープに対応するテスト ステップ、テスト ケース インスタンス、テスト ケース、または DevTest コンポーネントの名前。 この値には、スコープを示すツールヒントがあります。
   d. ステータス：緑色はアセットがアクティブであることを示します。黄色はアセットがアイドル状態であることを示します。 灰色はアセットが閉じられていることを示します。
5. アセットの詳細を表示するには、ステータス アイコンをクリックします。

6. アセットをすぐに閉じるには、ステータス アイコンをクリックし、［閉じる］または［強制的にクローズ］を選択します。
7. アセットのセットを閉じるには、［強制的にクローズ］をクリックします。

チュートリアル - JMS メッセージの送信および受信

このチュートリアルでは、JMS 送信/受信テスト ステップの使用方法の概要を示します。

前提条件

- レジストリが実行されている。
- デモ サーバが実行されている。

手順 1 - JMS および JNDI アセットの作成

この手順では、サンプル テスト ケースのテスト ステップから JMS および JNDI アセットを作成します。

次の手順に従ってください:

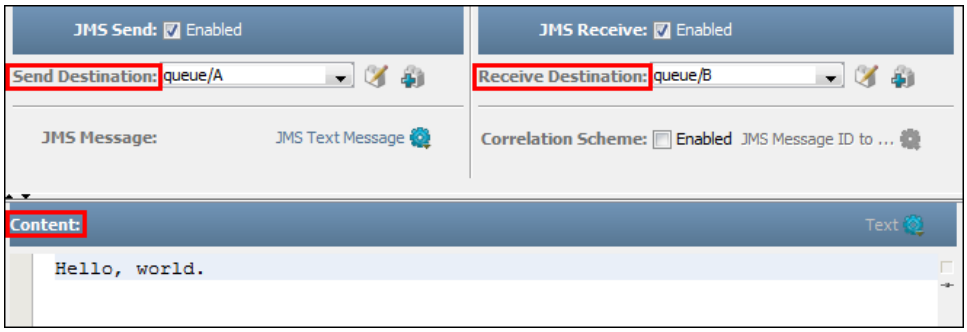
1. DevTest ワークステーション に移動し、examples プロジェクトを開きます。
2. Tests フォルダで、**jms.tst** テスト ケースを開きます。
3. 最初の JMS メッセージング (JNDI) ステップを選択します。このステップの名前は **jms-1** です。
4. モデル ツールバーの［選択されたステップからアセットを生成］ボタンをクリックします。
5. 2 番目の JMS メッセージング （JNDI）ステップを選択します。 このステップの名前は **send-msg-post-update** です。
6. モデル ツールバーの［選択されたステップからアセットを生成］ボタンをクリックします。
7. Configs フォルダで、プロジェクト設定を開きます。
8. JMS および JNDI アセットが生成されたことを確認します。次の手順では、2 つの JMS 送信先アセットを選択します。

手順 2 - JMS 送信/受信ステップの設定

この手順では、以下のアクションを実行します。

- 新しいテスト ケースに JMS 送信/受信ステップを追加する。
- キューに JMS のテキスト メッセージを送信し、別のキューから応答を受信するステップを設定する。

以下の図は、ステップ エディタの一部を示しています。 変更するフィールドが強調表示されます。



［送信先］フィールドは、メッセージが送信されるキューを指定します。

［受信先］フィールドは、応答が受信されるキューを指定します。

［コンテンツ］領域は、メッセージのテキストを指定します。

次の手順に従ってください:

1. examples プロジェクトにテスト ケースを作成します。
2. JMS 送信/受信ステップを追加します。

   ステップ エディタが表示されます。
3. ［送信先］リストで、**queue/A** を選択します。
4. ［受信先］リストで、**queue/B** を選択します。
5. ［コンテンツ］領域で、文を入力します。
6. テスト ケースを保存します。

手順 3 - JMS および JNDI アセットの確認

この手順では、JMS 送信/受信ステップで使用する JMS 送信先アセットの 1 つを確認します。 また、JMS 送信先アセットが使用する JNDI コンテキスト アセットも確認します。

次の手順に従ってください:

1. ［送信先］リストの右側に表示される［選択されたアセットの編集］ボタンをクリックします。

   JMS 送信先アセットのエディタが表示されます。

2. パラメータを確認します。ただし、変更は行いません。 パラメータ名の上にマウス ポインタを置くと、ツールヒントが表示されます。

3. ［JNDI コンテキスト］リストの右側に表示される［選択されたアセットの編集］ボタンをクリックします。

   JNDI コンテキスト アセットのエディタが表示されます。

4. パラメータを確認します。ただし、変更は行いません。

5. ［キャンセル］をクリックして JMS 送信先アセットのエディタに戻ります。

6. ［キャンセル］をクリックしてステップ エディタに戻ります。

手順 4 - JMS 送信/受信ステップのテスト

この手順では、JMS 送信/受信ステップが正しく設定されていることを検証します。

次の手順に従ってください:

1. ステップ エディタで、緑色の［実行］ボタンをクリックします。

2. ［要求］タブおよび［応答］タブに同じ文が含まれていることを確認します。

3. ステップ エディタを閉じます。

BEA ステップ

以下のステップが使用できます。

WebLogic JMS（JNDI）

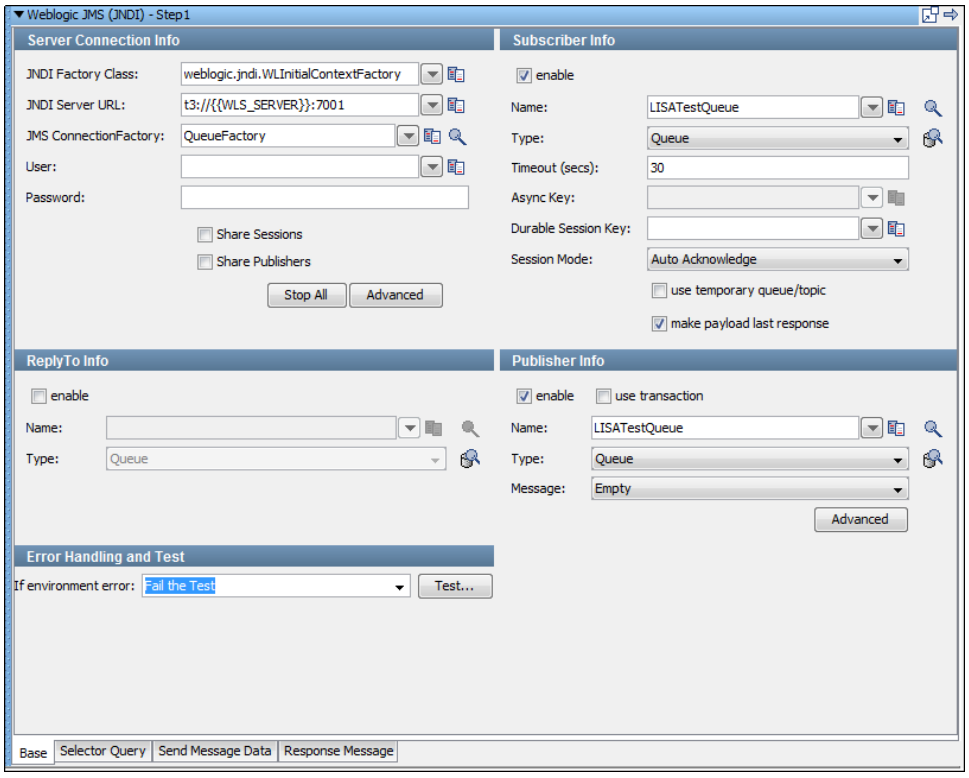
WebLogic JMS （JNDI）ステップでは、トピックおよびキューに対してメッセージを送受信できます。 また、既存のメッセージを受信、変更、転送できます。

WebLogic JMS （JNDI）では、空、テキスト、オブジェクト、バイト、メッセージ、マップ済み（拡張）など、一般的なメッセージ タイプがすべてサポートされています。

WebLogic JMS （JNDI）ステップは、メッセージング要件にかかわらず、単一のエディタを使用して設定されます。 入力オプションはメッセージング要件によって異なります。エディタは有効な設定のみを許可します。特定の機能を有効にすると、その他の機能が非アクティブになる場合があります。

前提条件： このアプリケーションと一緒に DevTest を使用するには、1 つ以上のファイルを DevTest で使用可能にする必要があります。詳細については、「*管理*」の「サードパーティ ファイル要件」を参照してください。

パラメータ要件： テスト中のアプリケーションで使用される接続パラメータ、およびサブジェクト名が必要です。 以下のセクションでは、必要なパラメータについて説明しています。 環境に応じて、その他のパラメータが必要となる場合があります。これらのパラメータは、アプリケーション開発者から入手します。



WebLogic JMS（JNDI）ステップ エディタには以下のタブが含まれます。

［ベース］タブでは、接続およびメッセージング パラメータを定義します。

［セレクタ クエリ］タブでは、キュー上のメッセージをリスンする場合に実行されるセレクタ クエリを指定できます。

［メッセージ データの送信］タブでは、メッセージ コンテンツを作成します。

［応答メッセージ］タブでは、応答メッセージを POST します。

[ベース]タブ

［ベース］タブは、以下のセクションで構成されています。

- サーバ接続情報
- サブスクライバ情報
- パブリッシャ情報
- ReplyTo 情報
- エラー処理およびテスト

［サブスクライバ情報］、［パブリッシャ情報］、および［返信先情報］を有効および無効にするには、各セクションの［有効］チェック ボックスを使用します。このチェック ボックスを使用して、ステップをパブリッシュ ステップ、サブスクライブ ステップ、またはその両方に設定できます。 また、ステップに JMS の返信先コンポーネントを含めることもできます。

テスト ステップの設定が完了したら、［エラー処理およびテスト］セクションの［テスト］をクリックして設定をテストします。

サーバ接続情報

JNDI 情報を入力します。

これらの値は、設定のプロパティでパラメータ化されます。 これらのプロパティにより、テスト中のアプリケーションの変更が容易になります。デフォルトでは、JNDI サーバ URL の WLS_SERVER プロパティが使用されます。このプロパティを使用する場合は、設定に追加する必要があります。

テスト中のシステムに対して、5 つのパラメータを使用できます。 プルダウン メニューには、これらの値の一般的な例またはテンプレートが含まれます。

JNDI ファクトリ クラス

JNDI プロバイダのコンテキスト ファクトリの完全修飾クラス名。

JNDI サーバ URL

JNDI サーバに接続するための URL。 URL の形式は、使用されている特定の JNDI プロバイダによって異なります。

JMS 接続ファクトリ

［検索］  を使用して、サーバ上の利用可能なリソースを参照します。JMS 仕様に従って、このステップの実行に使用する接続ファクトリを選択または入力します。

ユーザ

JNDI プロバイダに接続し、接続ファクトリのハンドルを取得するためのユーザ名。

パスワード

JNDI プロバイダに接続し、接続ファクトリのハンドルを取得するためのパスワード。

セッションの共有およびパブリッシャの共有

テスト ケース全体で JMS セッションおよびパブリッシャを共有するには、これらのチェック ボックスを使用します。 この方法はオーバーヘッドを低減できますが、通常、JMS クライアントがリソースを解放するため、現実的なシミュレーションとなるとは限りません。 ［パブリッシャの共有］チェック ボックスをオンにすると、［セッションの共有］チェック ボックスもオンになります。セッションを共有しなければ、パブリッシャを共有できません。これらのパラメータの詳細については、「*Deliberate Delays in VSE*」ナレッジ ベース記事を参照してください。

すべて停止

設計時にすべてのリスナを停止します。 一部のリスナは切り離すことができますが、引き続きメッセージを処理します。 メッセージを処理する場合は、テスト ケースを作成することは困難です。

詳細

接続情報と一緒に送信されるカスタム プロパティを追加でき、第 2 レベルの認証を設定できるパネルを表示します。

パブリッシャ情報

メッセージを送信（パブリッシュ）する機能を設定するには、［有効］チェック ボックスをオンにします。 メッセージを送信する場合にコミットを実行するには、［トランザクションの使用］チェック ボックスをオンにします。

以下のパラメータを入力します。

名前

トピックまたはキューの名前。［検索］アイコン を使用して、トピックまたはキュー名の JNDI サーバを参照します。

タイプ

トピックまたはキューを使用する場合に選択します。 どのメッセージがキュー（のみ）で処理されるのを待機しているかを確認するには、このフィールドの右にある［検索］アイコン を使用します。

メッセージ

送信するメッセージのタイプを選択します。 サポートされているタイプは、［なし］、［テキスト］、［オブジェクト］、［バイト］、［メッセージ］、および［マップ済み（拡張）］です。

詳細

メッセージ ヘッダの編集とメッセージ プロパティの追加を行えるパネルを表示します。

サブスクライバ情報

メッセージを受信（サブスクライブ）する機能を設定して有効にするには、［有効］チェック ボックスをオンにします。

以下のパラメータを入力します。

名前

トピックまたはキューの名前。［検索］アイコン を使用して、トピックまたはキュー名の JNDI サーバを参照します。

タイプ

トピックまたはキューを使用するかどうか、および同期または非同期モードでリスンするかどうかを選択します。 非同期モードでは、［非同期キー］フィールドにもエントリが必要です。 このフィールドの右側にある［検索］ を使用して、どのメッセージがキュー（のみ）で処理されるのを待機しているかを確認します。

タイムアウト（秒）

アプリケーションがメッセージの待機を中断するまでの待機期間（このフィールドを空白のままにすると、タイムアウトは設定されません）。

非同期キー

非同期メッセージを識別する値。 この値は非同期モードにのみ必要です。この値は、非同期メッセージを取得するために後続のメッセージ コンシューマ ステップで使用されます。

持続セッション キー

ここで名前を入力することによって、持続セッションを要求します。 また、そのセッションのキーも指定します。 持続セッションでは、ログアウトした後に再度ログインしても、トピックからのメッセージをすべて受信できます。

トランザクションの使用

メッセージを受信する場合にコミットを実行するには、［トランザクションの使用］チェック ボックスをオンにします。

一時キュー/トピックを使用する

JMS プロバイダに一時キュー/トピックを設定させるには、［一時キュー/トピックを使用する］チェック ボックスをオンにします。一時キュー/トピックが使用される場合、一時キュー/トピックに送信するメッセージの JMS **ReplyTo** パラメータが自動的に設定されます。 返信を送信できるように、一時キュー/トピック機能とパブリッシャを常に一緒に使用する必要があります。 一時キュー/トピックを使用する場合、［返信先］セクションは無効です。

ペイロードを最終応答にする

ペイロードをこのステップの応答にするには、［ペイロードを最終応答にする］チェック ボックスをオンにします。

ReplyTo 情報

送信先キューまたはトピックを設定するには、［有効］チェック ボックスをオンにします。

アプリケーションが送信先を必要とする場合、このセクションで設定されます。

以下のパラメータを入力します。

名前

トピックまたはキューの名前。［検索］アイコン を使用して、トピックまたはキュー名の JNDI サーバを参照します。

タイプ

トピックまたはキューを使用する場合に選択します。 どのメッセージがキュー（のみ）で処理されるのを待機しているかを確認するには、このフィールドの右にある［検索］アイコン を使用します。

エラー処理およびテスト

例外が発生した場合、［エラー処理およびテスト］セクションはステップにリダイレクトします。

環境エラーの場合

環境エラーが発生した場合に実行するステップまたは実行するアクションを選択します。

ステップ設定をテストするには、［テスト］をクリックします。

［セレクタ クエリ］タブ

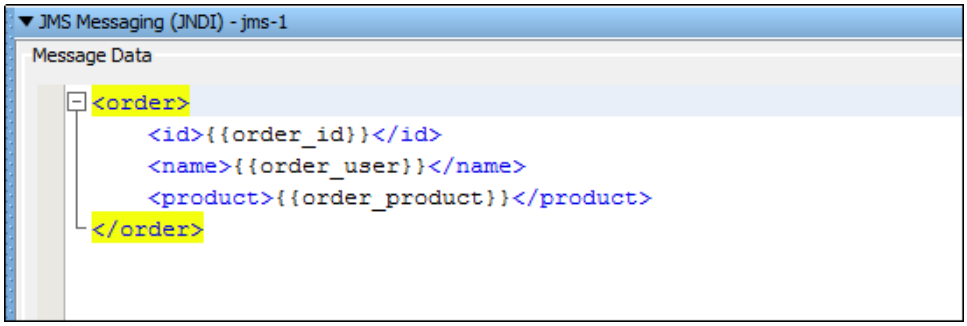
このエディタで JMS セレクタ クエリを入力できます。 構文は SQL に準拠しています。このクエリは SQL92 のサブセットです。 JMS セレクタ クエリは、パブリッシュされたメッセージに対する応答であるキュー上のメッセージをリスンする場合に指定できます。 上記の図は、元のメッセージで送信されたプロパティで設定されたものと一致する **JMSCorrelationID** に関する特定のクエリを示しています。

ビルトイン メカニズムでは、メッセージを送信する前にテスト作成者が **JMSCorrelationID** を設定することができます。**lisa.jms.correlation.id** プロパティを設定することにより、メッセージが送信される前に相関 ID を設定できます。

ゼロ以外の値が検出されます。また、メッセージが送信される前に、メッセージの **JMSCorrelationID** プロパティが設定されます。

[メッセージ データの送信]タブ

ステップがパブリッシュのために設定されている場合、このタブでメッセージを作成できます。 以下の例の［メッセージ データの送信］タブビューは、テキスト メッセージを示しています。



この例は、XML フラグメントと使用されているプロパティを示しています。テキストを入力するか、［ファイルからメッセージを読み取り］をクリックしてファイルから読み取るか、プロパティにフラグメントを格納できます。プロパティにテキストを格納する場合は、単にエディタにプロパティ（**LISA_PROP** など）を配置します。

プロパティがメッセージの XML で使用されていることで、テストの実行時にメッセージを動的に作成できることに注目してください。

[応答メッセージ]タブ

ステップがサブスクライブのために設定されている場合、応答はここに表示されます。 詳細については、「JMS メッセージング （JNDI） (P. 326)」を参照してください。

メッセージ コンシューマ

このステップの詳細については、「JMS メッセージング - メッセージ コンシューマ (P. 337)」を参照してください。

ファイルの読み取り

このステップの詳細については、「ファイルの読み取り（ディスク、URL、またはクラスパス） (P. 322)」を参照してください。

Web サービス実行（XML）

このステップの詳細については、「Web サービス実行（XML）ステップ (P. 215)」を参照してください。

RAW SOAP 要求

このステップの詳細については、「Web-Raw SOAP 要求 (P. 263)」を参照してください。

FTP ステップ

このステップの詳細については、「外部 - FTP ステップ (P. 324)」を参照してください。

Sun JCAPS ステップ

以下のステップが含まれます。

JCAPS メッセージング（ネイティブ） (P. 362)
JCAPS メッセージング（JNDI） (P. 365)

JCAPS メッセージング（ネイティブ）

JCAPS メッセージング（ネイティブ）ステップでは、トピックおよびキューからメッセージを送受信できます。 また、既存のメッセージを受信、変更、転送できます。

JCAPS メッセージング（ネイティブ）では、空、テキスト、オブジェクト、バイト、メッセージ、マップ済み（拡張）など、一般的なメッセージ タイプがすべてサポートされています。

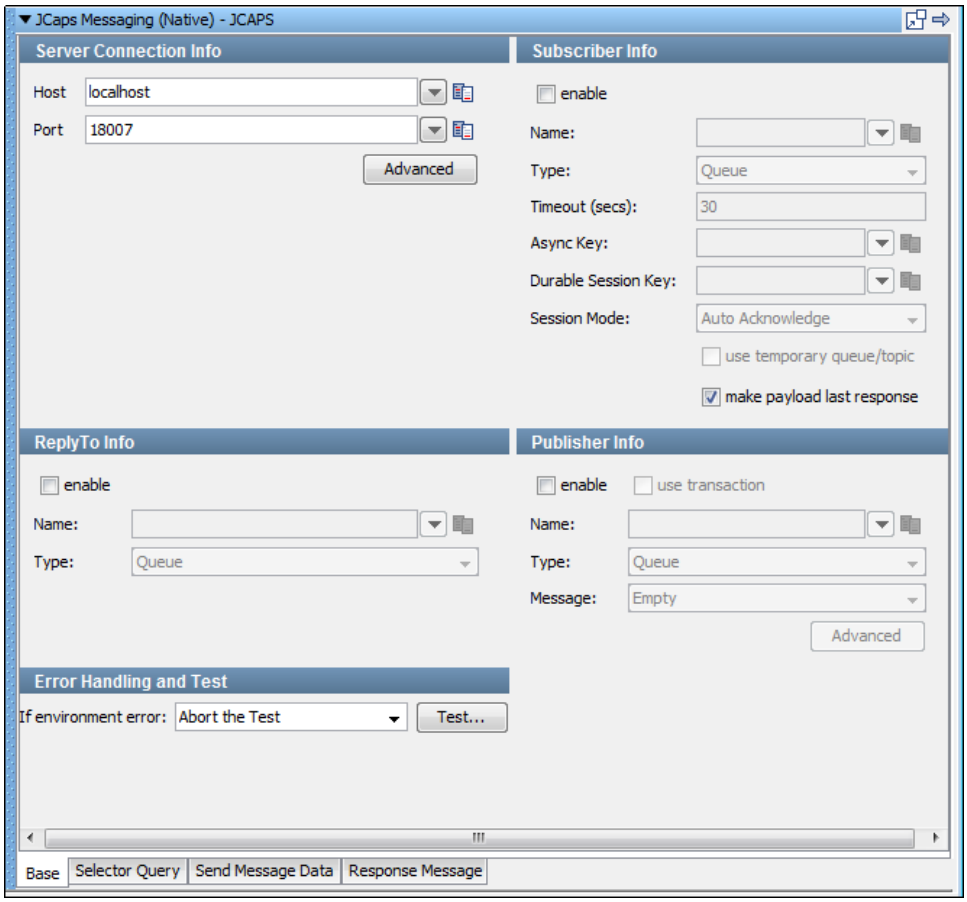
JCAPS メッセージング（ネイティブ）ステップは、メッセージング要件にかかわらず、単一のエディタを使用して設定されます。 入力オプションはメッセージング要件によって異なります。 エディタは有効な設定のみを許可します。そのため、特定の機能を有効にすると、その他の機能が非アクティブになる場合があります。

JCAPS メッセージング（ネイティブ）ステップには、「JCAPS キュー名 パブリッシュ」という命名規則を使用したデフォルトの名前があります。パブリッシュ キュー名がない場合、デフォルトのステップ名は「JCAPS キュー名 サブスクライブ」です。 デフォルトのステップ名を別のステップが使用する場合、DevTest は、このステップ名に番号を追加して一意にします。 ステップ名は、いつでも変更できます。

前提条件： このアプリケーションと一緒に DevTest を使用するには、1 つ以上のファイルを DevTest で使用可能にする必要があります。詳細については、「*管理*」の「サードパーティ ファイル要件」を参照してください。

パラメータ要件： テスト中のアプリケーションで使用される接続パラメータ、およびサブジェクト名が必要です。 以下のセクションでは、必要なパラメータについて説明しています。 環境に応じて、その他のパラメータが必要となる場合があります。これらのパラメータは、アプリケーション開発者から入手します。

JCAPS メッセージング（ネイティブ）ステップ エディタは、このステップを設定するために使用されます。



JCAPS メッセージング（ネイティブ）ステップ エディタには以下のタブが含まれます。

- ［ベース］タブでは、接続およびメッセージング パラメータを定義します。
- ［セレクタ クエリ］タブでは、キュー上のメッセージをリスンする場合に実行されるセレクタ クエリを指定できます。
- ［メッセージ データの送信］タブでは、メッセージ コンテンツを作成します。
- ［応答メッセージ］タブでは、応答メッセージを POST します。

[ベース]タブ

［ベース］タブは、以下のセクションで構成されています。

- サーバ接続情報
- サブスクライバ情報
- 返信先情報
- パブリッシャ情報
- エラー処理およびテスト

各セクションの左上隅にある［有効］チェック ボックスを使用して、［サブスクライバ情報］、［パブリッシャ情報］、および［返信先情報］セクションを有効および無効にできます。このチェック ボックスを使用して、ステップをパブリッシュ ステップ、サブスクライブ ステップ、またはその両方に設定できます。 また、オンにすると、ステップに JMS の返信先コンポーネントを含めることもできます。

テスト ステップの設定が完了したら、［エラー処理およびテスト］セクションの［テスト］をクリックして設定をテストします。

サーバ接続情報

［サーバ接続情報］セクションには、テスト中のシステムで使用可能な 2 つのパラメータが表示されます。

ホスト

JMS サーバの名前。

ポート

JMS サーバが実行されているポート番号。

［詳細］ボタンは、接続情報と一緒に送信されるカスタム プロパティを追加できるパネルを表示します。

その他のすべてのタブについては、「JMS メッセージング（JNDI） (P. 326)」で詳しく説明しています。

JCAPS メッセージング（JNDI）

JCAPS メッセージング（JNDI）ステップでは、トピックおよびキューからメッセージを送受信できます。 また、既存のメッセージを受信、変更、転送できます。

JCAPS メッセージング（JNDI）では、空、テキスト、オブジェクト、バイト、メッセージ、マップ済み（拡張）など、一般的なメッセージ タイプがすべてサポートされています。

JCAPS メッセージング（JNDI）ステップは、メッセージング要件にかかわらず、単一のエディタを使用して設定されます。 入力オプションはメッセージング要件によって異なります。 エディタは有効な設定のみを許可します。そのため、一部の機能を有効にすると、その他の機能が非アクティブになる場合があります。

前提条件： このアプリケーションと一緒に DevTest を使用するには、1 つ以上のファイルを DevTest で使用可能にする必要があります。詳細については、「*管理*」の「サードパーティ ファイル要件」を参照してください。

パラメータ要件： このステップには、テスト中のアプリケーションで使用される接続パラメータ、およびサブジェクト名が必要です。 環境に応じて、その他のパラメータが必要となる場合があります。 これらのパラメータは、アプリケーション開発者から入手します。

JCAPS メッセージングのパラメータおよびフィールドの詳細については、「JMS メッセージング （JNDI） (P. 326)」を参照してください。

デフォルトのステップ名： JCAPS メッセージング（JNDI）ステップのデフォルト名では、「*JCAPS キュー名 パブリッシュ*」という命名規則が使用されます。 パブリッシュ キュー名がない場合、デフォルトのステップ名は「JCAPS キュー名 サブスクライブ」です。デフォルトのステップ名を別のステップが使用する場合、DevTest は、このステップ名に番号を追加して一意にします。 ステップ名は、いつでも変更できます。

Oracle ステップ

以下のステップが使用できます。

Oracle OC4J （JNDI） (P. 366)
Oracle AQ ステップ (P. 367)

Oracle OC4J（JNDI）

Oracle OC4J（JNDI）ステップでは、トピックおよびキューに対してメッセージを送受信できます。 また、既存のメッセージを受信、変更、転送できます。 Oracle OC4J （JNDI）では、空、テキスト、オブジェクト、バイト、メッセージ、マップ済み（拡張）など、一般的なメッセージ タイプがすべてサポートされています。

Oracle OC4J （JNDI）ステップは、メッセージング要件にかかわらず、単一のエディタを使用して設定されます。 入力オプションはメッセージング要件によって異なります。 エディタは有効な設定のみを許可します。そのため、一部の機能を有効にすると、その他の機能が非アクティブになる場合があります。

前提条件： このアプリケーションと一緒に DevTest を使用するには、1 つ以上のファイルを DevTest で使用可能にする必要があります。詳細については、「*管理*」の「サードパーティ ファイル要件」を参照してください。

パラメータ要件： テスト中のアプリケーションで使用される接続パラメータ、およびサブジェクト名が必要です。 環境に応じて、その他のパラメータが必要となる場合があります。 これらのパラメータは、アプリケーション開発者から入手します。

デフォルトでは、DevTest は JNDI サーバ URL の OC4J_SERVER プロパティを使用します。 このプロパティを使用する場合は、設定に追加する必要があります。

このステップのパラメータおよびフィールドの詳細については、「JMS メッセージング（JNDI） (P. 326)」を参照してください。

Oracle AQ ステップ

Oracle AQ は、IBM WebSphere MQ、webMethods Broker、TIBCO EMS などのようなメッセージング プロバイダです。 Oracle AQ は、これらのその他のメッセージング プロバイダのように DevTest に適合しています。

Oracle AQ ステップは、メッセージの送信、メッセージの受信、メッセージの非同期での受信、またはそれら 3 つの複数の組み合わせを可能にするためにテスト ケースに追加されます。

AQ には、2 つの個別のステップ タイプ（JMS および JPUB）があります。これらの機能は同じですが、Oracle AQ メッセージング プロバイダと通信する 2 つの異なる方法を表わしています。

どちらの Oracle AQ ステップにも、メッセージング ステップの標準設定セクションがあります。

- 接続情報を設定するための［接続］セクション。
- このステップがメッセージを受信する場所、および受信するメッセージのタイプを設定するための［サブスクライバ］セクション（オプション）。
- このステップがメッセージを送信する場所、および送信するメッセージのタイプを設定するための［パブリッシャ］セクション（オプション）。 送信するメッセージのコンテンツは個別のタブで設定します。
- このステップは、1 つのメッセージを送信/受信（または両方）します。このステップは、複数のメッセージを送受信するために、同一のテスト ケース内のループで複数回実行される場合があります。
- このステップが非同期サブスクライバとして使用される場合は、テスト ケースで 1 回のみ実行されます。 プロバイダから受信した各メッセージを処理するために、追加のコンシューマ ステップが実行されます。

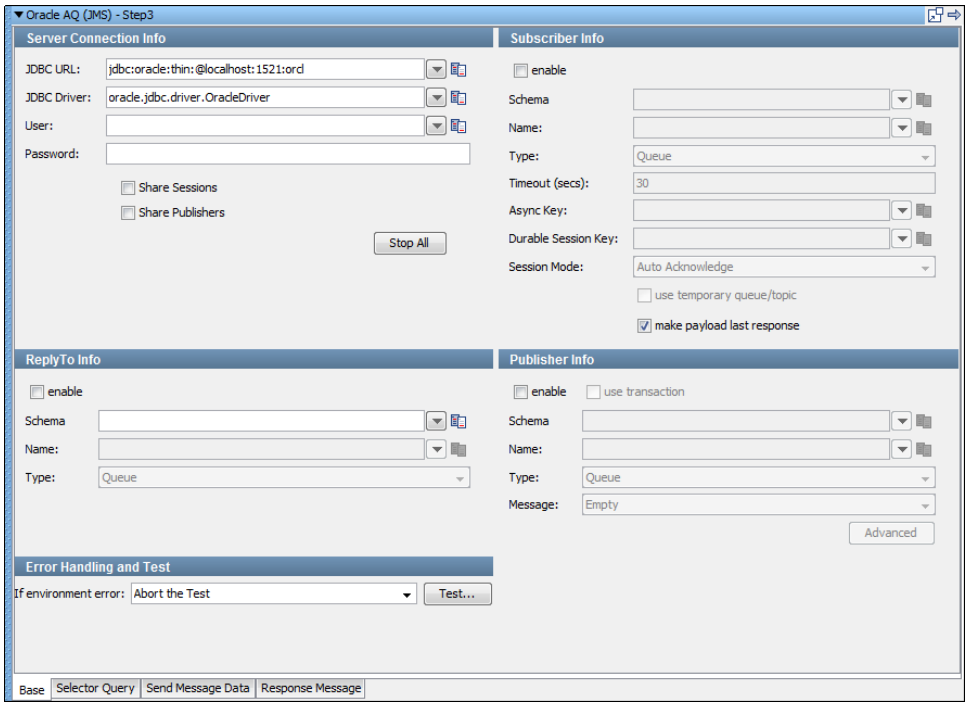
AQ を使用する 2 つの方法を以下に示します。

- JMS（P. 368）
- JPUB（P. 375）

Oracle AQ（JMS）

Oracle Advanced Queuing （AQ）は、Oracle データベース自体に組み込まれているメッセージング プロバイダです。 AQ は、デフォルトの JMS プロバイダとして、Oracle Enterprise Service Bus などの多くの Oracle 製品で使用されます。

AQ を使用する 2 つの方法のうちの 1 つは JMS です。



JMS ライブラリを使用することにより、その他の通常の JMS プロバイダのように動作します。相違点は以下のとおりです。

- JMS 接続は JNDI によって作成されません。 接続は JDBC 接続を使用して作成されます。これには、JDBC URL、ドライバ クラス名、ユーザ名、およびパスワードの入力が含まれます。
- キューおよびトピックは、データベース内のスキーマに関連付けられます。キューに対して送受信を行うには、キュー名とキュー スキーマの両方を指定します。
- それぞれのキューまたはトピックは、特定のタイプの JMS メッセージに制限されます。たとえば、キューが通常 JMS テキスト メッセージを転送する場合、JMS オブジェクト メッセージまたは JMS バイト メッセージを転送するためにその同じキューを使用できません。

- Oracle DB での JMS キューおよびトピックの設定には、ストアド プロシージャの実行が含まれています。

エディタの下部の 4 つのタブが使用できます。

- ［ベース］タブでは、接続およびメッセージ パラメータを定義します。
- ［セレクタ クエリ］タブでは、キュー上のメッセージをリスンする場合に実行されるセレクタ クエリを指定できます。
- ［メッセージ データの送信］タブでは、メッセージ コンテンツを作成します。
- ［応答メッセージ］タブでは、応答メッセージを POST します。

基本情報タブ

上記の例で示されている［ベース］タブ ビューには、5 つの主要セクションがあります。

- サーバ接続情報
- サブスクライバ情報
- パブリッシャ情報
- ReplyTo 情報
- エラー処理およびテスト

［サーバ接続情報］および［エラー処理およびテスト］セクションは常にアクティブです。 各セクションの左上隅にある［有効］チェック ボックスを使用して、［サブスクライバ情報］、［パブリッシャ情報］、および［返信先情報］を有効または無効にします。 これらのチェック ボックスを使用して、ステップをパブリッシュ ステップ、サブスクライブ ステップ、またはその両方に設定できます。 また、オンにすると、ステップに返信先コンポーネントを含めることもできます。

テスト ステップの設定が完了したら、［エラー処理およびテスト］セクションの［テスト］をクリックして設定をテストします。

サーバ接続情報

ここでは、JDBC 関連の情報を入力します。

これらの値を設定内のプロパティでパラメータ化して、テスト中のアプリケーションの変更を容易にします。

DevTest は、デフォルトでは、JDBC ドライバの場所の **oracle.jdbc.driver.OracleDriver** を使用します。

テスト中のシステムに対して、以下のパラメータを使用できます。 プルダウン メニューには、これらの値の一般的な例またはテンプレートが含まれます。

JDBC URL

このフィールドには、デフォルト値が入力されます。

JDBC **サーバ**

このフィールドには、デフォルト値が入力されます。

ユーザ

ユーザ名を入力します。

パスワード

パスワードを入力します。

セッションの共有およびパブリッシャの共有

テスト ケース全体で JMS セッションおよびパブリッシャを共有するには、これらのチェック ボックスを使用します。 この方法はオーバーヘッドを低減できますが、通常、JMS クライアントがリソースを解放するため、現実的なシミュレーションとなるとは限りません。 ［パブリッシャの共有］チェック ボックスをオンにすると、［セッションの共有］チェック ボックスもオンになります。セッションを共有しなければ、パブリッシャを共有できません。これらのパラメータの詳細については、「*Deliberate Delays in VSE*」ナレッジ ベース記事を参照してください。

すべて停止

設計時にすべてのリスナを停止します。 一部のリスナは切り離すことができますが、引き続きメッセージを処理します。 メッセージを処理する場合は、テスト ケースを作成することは困難です。

パブリッシャ情報

メッセージを送信（パブリッシュ）する機能を設定するには、［有効］チェック ボックスをオンにします。

メッセージを送信する場合にコミットを実行するには、［トランザクションの使用］チェック ボックスをオンにします。

以下のパラメータを入力します。

スキーマ

使用するスキーマの名前を入力します。

名前

使用するトピックまたはキューの名前を入力します。

タイプ

トピックまたはキューを使用する場合に選択します。

メッセージ

送信するメッセージのタイプを選択します。 サポートされているタイプは、［なし］、［テキスト］、［オブジェクト］、［バイト］、［メッセージ］、および［マップ済み（拡張）］です。

詳細

メッセージ ヘッダの編集とメッセージ プロパティの追加を行えるパネルを表示します。

サブスクライバ情報

メッセージを受信（サブスクライブ）する機能を設定して有効にするには、［有効］チェック ボックスをオンにします。

以下のパラメータを入力します。

スキーマ

使用するスキーマの名前を入力します。

名前

使用するトピックまたはキューの名前を入力します。

タイプ

トピックまたはキューを使用するかどうか、および同期または非同期モードでリスンするかどうかを選択します。 非同期モードでは、［非同期キー］フィールドにもエントリが必要です。 このフィールドの右側にある［検索］アイコンを使用して、どのメッセージがキュー（のみ）で処理されるのを待機しているかを確認できます。

タイムアウト（秒）

DevTest がメッセージの待機を中断するまでの秒数を示します。タイムアウトを設定しない場合は、このフィールドは空白にしておきます。

非同期キー

非同期メッセージを識別するために必要な値を入力します。 この値は非同期モードにのみ必要です。 この値は、非同期メッセージを取得するために後続のメッセージ コンシューマ ステップで使用されます。

持続セッション キー

ここで名前を入力することによって、持続セッションを要求します。 また、そのセッションのキーも指定します。 持続セッションでは、ログアウトした後に再度ログインしても、トピックからのメッセージをすべて受信できます。

セッション モード

ドロップダウン リストをクリックして、使用可能なオプションから適切なモードを選択します。オプションは次のとおりです:［自動確認応答］、［クライアント確認応答］、［トランザクションの使用］、［自動（重複 OK）］。

ReplyTo 情報

アプリケーションが送信先を必要とする場合、このセクションで設定されます。

送信先キュー/トピックを設定するには、［有効］チェック ボックスをオンにします。

以下のパラメータを入力します。

スキーマ

使用するスキーマの名前を入力します。

名前

使用するトピックまたはキューの名前を入力します。

タイプ

トピックまたはキューを使用する場合に選択します。

エラー処理およびテスト

エラーが発生した場合、［エラー処理およびテスト］セクションはステップにリダイレクトします。

環境エラーの場合

環境エラーが発生した場合に実行するステップまたは実行するアクションを選択します。

ステップ設定をテストするには、［テスト］をクリックします。

Oracle AQ（JPUB）

AQ を使用する 2 つの方法を以下に示します。

- JMS
- JPUB

Oracle AQ JMS API は、より低いレベルの AQ API の上に構築されるレイヤです。 この低いレベルの API は扱いがはるかに難しく、また、ほとんど JMS のようには動作しません。

最も大きな違いはメッセージ形式です。低いレベルの AQ メッセージにはペイロードが含まれます。これは、データベースで定義されている任意の型になります。 varchar 型または clob 型も使用できますが、通常はユーザ定義の構造化データベースの型です。 AQ JMS キューと同様、AQ の低いレベルのキューはそれぞれ 1 つのペイロード型のみを処理できます。

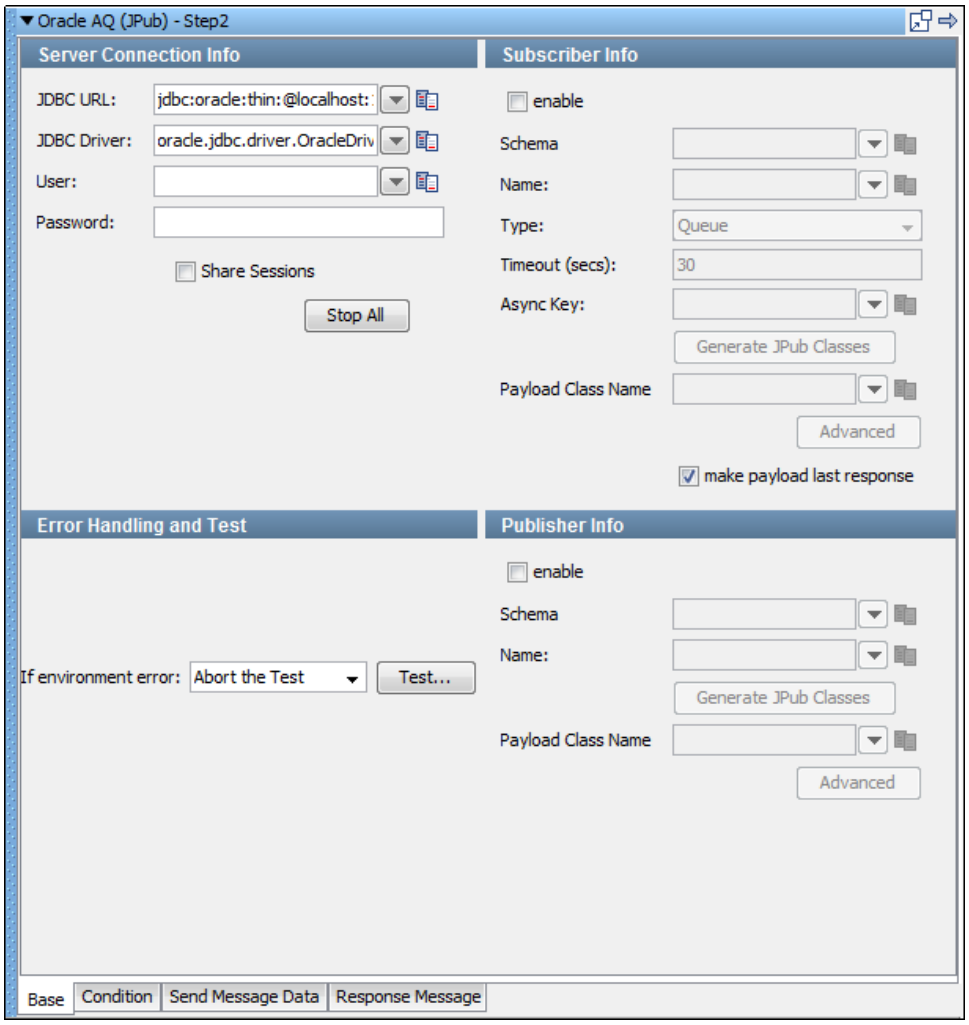
Oracle は、これらのユーザ定義の構造型を処理できる Java オブジェクトを生成可能な JPUB という名前のユーティリティを提供しています。JPUB は、Axis が Web サービスを使用する Java オブジェクトを生成するのと同じ方法で動作します。 Oracle AQ JPUB という名前の低いレベルの AQ ステップは、自動的にこのユーティリティを使用し、キュー情報に基づくクライアント クラスを生成できます。 その後、ユーザは標準の COE を使用して、ペイロード オブジェクトに入力します。

キューまたはトピックの相違点はありません。 クライアントは以下を実行できます。

- AQ キューから次のメッセージを削除し、それを本質的にキューとする。
- AQ キューから次のメッセージを削除せずに読み取り、それを本質的にトピックとする。

低いレベルの AQ キューの設定は、ストアド プロシージャによって再度実行されます。 AQ キューを作成する前に、データベースにユーザ定義の構造型を作成することができる追加の手順があります。 技術的には、低いレベルの API を使用して、AQ JMS キューと通信できます。 JMS キューには、標準の JMS メッセージのように構造化された特定のペイロード型があります。 ただし、低いレベルの AQ キュー（JMS ペイロード型を使用しないキュー）との通信には AQ JMS API を使用できません。

テスト ケースに Oracle AQ （JPUB）ステップを追加するには、ステップをクリックしてエディタを開きます。



エディタの下部の 4 つのタブが使用できます。

- ［ベース］タブでは、接続およびメッセージ パラメータを定義します。
- ［条件］タブでは、実行される条件を指定できます。
- ［メッセージ データの送信］タブでは、メッセージ コンテンツを作成します。
- ［応答メッセージ］タブでは、応答メッセージを POST します。

基本情報タブ

［ベース］タブ ビューはデフォルト ビューで、上記の図に示されています。 これは、4 つの主なセクションで構成されています。

- サーバ接続情報
- サブスクライバ情報
- パブリッシャ情報
- エラー処理およびテスト

［サーバ接続情報］および［エラー処理およびテスト］セクションは常にアクティブです。 各セクションの左上隅にある［有効］チェック ボックスを使用して、［サブスクライバ情報］および［パブリッシャ情報］セクションを有効および無効にできます。 これらのチェック ボックスを使用して、ステップをパブリッシュ ステップ、サブスクライブ ステップ、またはその両方に設定できます。

テスト ステップの設定が完了したら、［エラー処理およびテスト］セクションの［テスト］をクリックして設定をテストします。

サーバ接続情報

これらの値を設定内のプロパティでパラメータ化して、テスト中のアプリケーションの変更を容易にする必要があります。デフォルトでは、JDBC ドライバの場所の **oracle.jdbc.driver.OracleDriver** が使用されます。

JDBC URL

このフィールドには、デフォルト値が入力されます。

JDBC **サーバ**

このフィールドには、デフォルト値が入力されます。

ユーザ

ユーザ名を入力します。

パスワード

パスワードを入力します。

セッションの共有およびパブリッシャの共有

テスト ケース全体で JMS セッションおよびパブリッシャを共有するには、これらのチェック ボックスを使用します。 この方法はオーバーヘッドを低減できますが、通常、JMS クライアントがリソースを解放するため、現実的なシミュレーションとなるとは限りません。 ［パブリッシャの共有］チェック ボックスをオンにすると、［セッションの共有］チェック ボックスもオンになります。セッションを共有しなければ、パブリッシャを共有できません。これらのパラメータの詳細については、「*Deliberate Delays in VSE*」ナレッジ ベース記事を参照してください。

すべて停止

設計時にすべてのリスナを停止します。 一部のリスナは切り離すことができますが、引き続きメッセージを処理します。 メッセージを処理する場合は、テスト ケースを作成することは困難です。

パブリッシャ情報

メッセージを送信（パブリッシュ）する機能を設定するには、［有効］チェック ボックスをオンにします。

以下のパラメータを入力します。

スキーマ

使用するスキーマの名前を入力します。

名前

使用するトピックまたはキューの名前を入力します。

JPub クラスの生成

クリックすると、JPub クラスを生成します。

ペイロード クラス名

ペイロード クラス名を入力します。

［詳細］ボタン

［詳細設定（パブリッシュ）］ダイアログ ボックスをクリックして開き、［相関］に入力または選択して［OK］をクリックします。

サブスクライバ情報

メッセージを受信（サブスクライブ）する機能を設定して有効にするには、［有効］チェック ボックスをオンにします。

以下のパラメータを入力します。

スキーマ

使用するスキーマの名前を入力します。

名前

使用するトピックまたはキューの名前を入力します。

タイプ

トピックまたはキューを使用するかどうか、および同期または非同期モードでリスンするかどうかを選択します。 非同期モードでは、［非同期キー］フィールドにもエントリが必要です。 このフィールドの右側にある［検索］アイコンを使用して、どのメッセージがキュー（のみ）で処理されるのを待機しているかを確認できます。

タイムアウト（秒）

DevTest がメッセージの待機を中断するまでの秒数を示します。タイムアウトを設定しない場合は、このフィールドは空白にしておきます。

非同期キー

非同期メッセージを識別するために必要な値を入力します。 この値は非同期モードにのみ必要です。 この値は、非同期メッセージを取得するために後続のメッセージ コンシューマ ステップで使用されます。

JPub クラスの生成

クリックすると、JPub クラスを生成します。

ペイロード クラス名

ペイロード クラス名を入力します。

詳細

サブスクライバの詳細設定ダイアログ ボックスをクリックして開きます。

［詳細］ダイアログ ボックスでは、［コンシューマ名］、［相関］、および［メッセージ ID］を入力できます。

エラー処理およびテスト

エラーが発生した場合、［エラー処理およびテスト］セクションはステップにリダイレクトします。

環境エラーの場合

環境エラーが発生した場合に実行するステップまたは実行するアクションを選択します。

ステップ設定をテストするには、［テスト］をクリックします。

TIBCO ステップ

以下のステップが使用できます。

TIBCO Rendezvous メッセージング (P. 381)
TIBCO EMS メッセージング (P. 388)
TIBCO ダイレクト JMS (P. 389)

TIBCO Rendezvous メッセージング

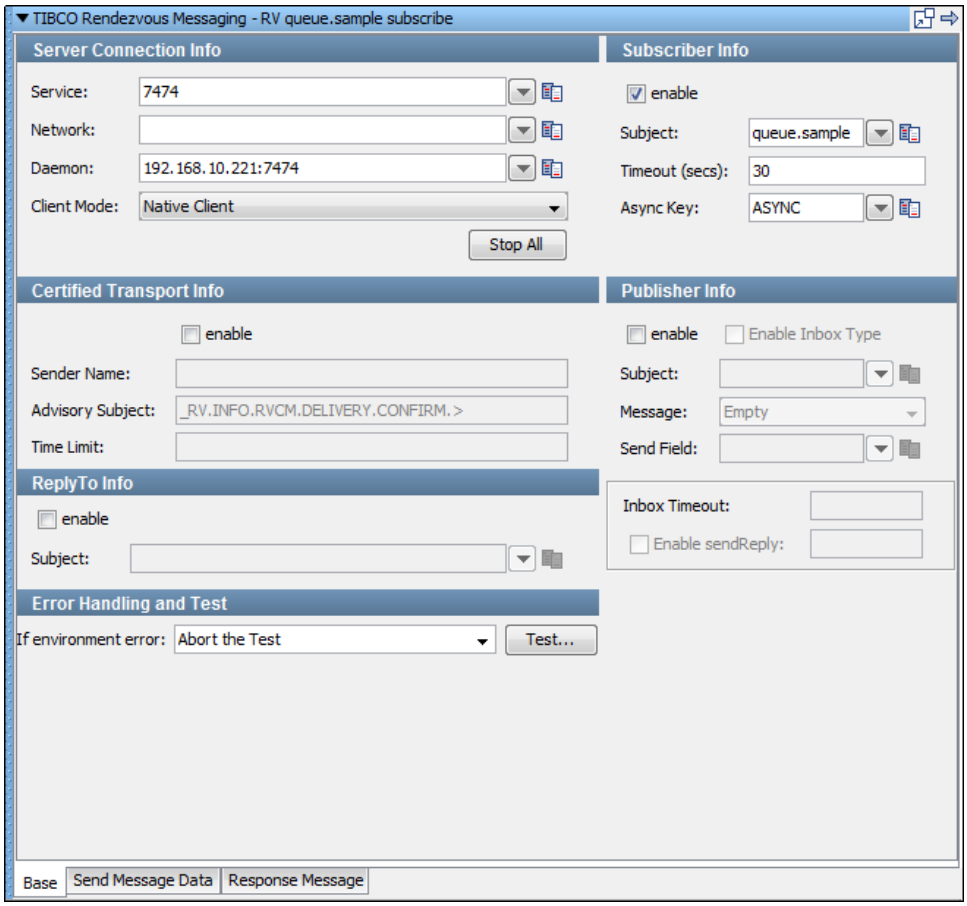
TIBCO Rendezvous メッセージング ステップでは、ネイティブ Rendezvous プロトコルを使用して、Rendezvous「サブジェクト」からメッセージを送受信できます。 また、既存のメッセージを受信、変更、転送できます。

TIBCO Rendezvous メッセージング ステップは、メッセージング要件にかかわらず、単一のエディタを使用して設定されます。入力オプションはメッセージング要件によって異なります。 エディタは有効な設定のみを許可します。そのため、特定の機能を有効にすると、その他の機能が非アクティブになる場合があります。

TIBCO Rendezvous メッセージング ステップには、「RV キュー名 パブリッシュ」という命名規則を使用したデフォルトの名前があります。パブリッシュ キュー名がない場合、デフォルトのステップ名は「RV キュー名 サブスクライブ」です。 デフォルトのステップ名を使用する別のステップがある場合、番号がステップ名に追加されます。 ステップ名は、いつでも変更できます。

前提条件： このアプリケーションと一緒に DevTest を使用するには、1 つ以上のファイルを DevTest で使用可能にする必要があります。詳細については、「*管理*」の「サードパーティ ファイル要件」を参照してください。

パラメータ要件： テスト中のアプリケーションで使用される接続パラメータ、およびサブジェクト名が必要です。 以下のセクションでは、必要なパラメータについて説明しています。 環境に応じて、その他のパラメータが必要となる場合があります。これらのパラメータは、アプリケーション開発者から入手します。



TIBCO Rendezvous メッセージング ステップ エディタには以下のタブが含まれます。

- ［ベース］タブでは、接続およびメッセージング パラメータを定義します。
- ［メッセージ データの送信］タブでは、メッセージ コンテンツを作成します。
- ［応答メッセージ］タブでは、応答メッセージを POST します。

[ベース]タブ

［ベース］タブは、以下のセクションで構成されています。

- サーバ接続情報
- サブスクライバ情報
- 認定済みトランスポート情報
- パブリッシャ情報
- 返信先情報
- エラー処理およびテスト

［**サブスクライバ情報**］、［パブリッシャ情報］、および［返信先情報］を有効および無効にするには、各セクションの左上隅にある[有効]チェック ボックスを使用します。 これらのチェック ボックスは、ステップをパブリッシュ ステップ、サブスクライブ ステップ、またはその両方に設定するために使用します。 また、オンにすると、ステップに JMS の返信先コンポーネントを含めることもできます。

テスト ステップの設定が完了したら、［エラー処理およびテスト］セクションの［テスト］をクリックして設定をテストします。

サーバ接続情報

［サーバ接続情報］領域の Rendezvous 情報に固有の接続情報を入力します。

テスト中のシステムに対して、以下のパラメータを使用できます。

サービス、ネットワーク、およびデーモン

これらのパラメータは、通信する RV ネットワークへの接続を有効にします。

クライアント モード

Rendezvous ネイティブ クライアント モードまたは Java クライアント モードのいずれかを選択します。 通常、より用途の広いクライアント モードを使用します。

これらの値を設定内のプロパティでパラメータ化して、テスト中の別のシステムの変更を容易にする必要があります。

パブリッシャ情報

メッセージを送信する機能を設定するには、［有効］チェック ボックスをオンにします。

以下のパラメータを入力します。

サブジェクト

使用するサブジェクトの名前。 独自のサブジェクトを定義できます。 有効なサブジェクト名は、**queue.sample** の形式です。 無効なサブジェクト名は、 **queue…..My_Samples** （NULL エレメント）または **.My.Queue.** （3 つの NULL エレメント）のような形式です。

メッセージ

送信するメッセージのタイプを選択します。 サポートされているタイプは、［なし］、［テキスト］、［オブジェクト］、［バイト］、［メッセージ］、および［マップ済み（拡張）］です。

［送信］フィールド

RV メッセージは、実際はフィールドと値のマップです。 このフィールドは、単一のフィールドのメッセージを迅速に作成するために使用します。ここで値を入力すると、メッセージの送信データがこの名前を持ったフィールドの値に入力されます。この値は、マップ済み（拡張）タイプのメッセージによって上書きされます。これは、このタイプのメッセージが単一のメッセージに複数のフィールドおよび値を入力するためです、

インボックス タイプの有効化

［インボックス タイムアウト］および［sendReply の有効化］フィールドを有効にするには、このチェック ボックスをオンにします。

sendReply の有効化

［インボックス タイムアウト］を指定、またはパブリッシャの sendReply 機能を有効にするには、［インボックス タイプの有効化］を選択します。

認定済みトランスポート情報

トランスポート情報を提供するには、［有効］チェック ボックスをオンにします。

送信者名

CM トランスポートに対応する名前。

アドバイザリの件名

Rendezvous ソフトウェアは、**_RV.class.SYSTEM.name** というテンプレートを使用して、システム アドバイザリ メッセージの件名を作成します。 Rendezvous 認定済みメッセージ配信ソフトウェアは、**_RV.class.RVCM.category.condition.subject** および **_RV.class.RVCM.category.condition.name** というテンプレートを使用して、アドバイザリ メッセージの件名を作成します。 分散キュー ソフトウェアは、**_RV.class.RVCM.category.role.condition.name** というテンプレートを使用して、アドバイザリ メッセージの件名を作成します。 Rendezvous フォールト トレランス ソフトウェアは、**_RV.class.RVFT.name.group** というテンプレートを使用して、アドバイザリ メッセージの件名を作成します。

時間制限

メッセージが存在する時間制限。

サブスクライバ情報

［**有効**］チェックボックスをオンにすると、サブスクライバ機能がオンになり、メッセージを受信する機能を設定できます。

以下のパラメータを入力します。

サブジェクト

使用するサブジェクトの名前。 独自のサブジェクトを定義できます。

タイムアウト（秒）

DevTest がメッセージの待機を中断するまでの秒数を示します。タイムアウトを設定しない場合は、このフィールドは空白にしておきます。

非同期キー

非同期メッセージを識別するために必要な値を入力します。 この値は非同期モードにのみ必要です。 この値は、非同期メッセージを取得するために後続のメッセージ コンシューマ ステップで使用されます。

返信先情報

送信先サブジェクトを設定するには、［**有効**］チェック ボックスをオンにします。

アプリケーションが送信先を必要とする場合、このセクションで設定されます。

以下のパラメータを設定します。

サブジェクト

使用するサブジェクトの名前。

エラー処理およびテスト

エラーが発生した場合、［エラー処理およびテスト］セクションはステップにリダイレクトします。

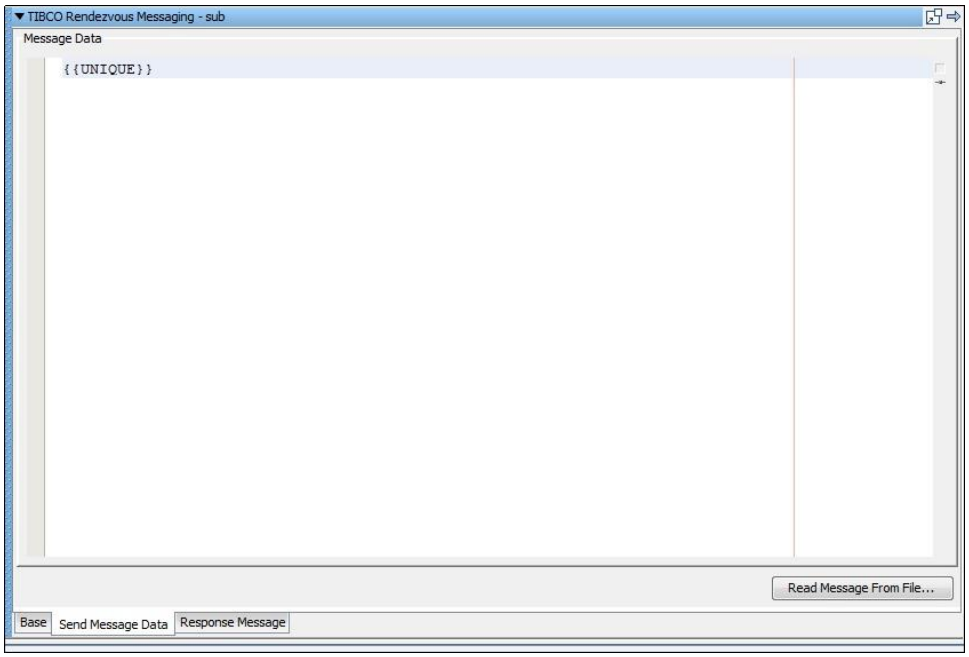
環境エラーの場合

環境エラーが発生した場合に実行するステップまたは実行するアクションを選択します。

ステップ設定をテストするには、［テスト］をクリックします。

メッセージ データの送信

ステップがパブリッシュのために設定されている場合、このタブでメッセージを作成できます。 以下の例の［メッセージ データの送信］タブビューは、テキスト メッセージを示しています。



この例は、XML フラグメントと使用されているプロパティを示しています。テキストを入力するか、［ファイルからメッセージを読み取り］をクリックしてファイルから読み取ることができます。 また、プロパティにテキストを格納できます。その場合、エディタにプロパティ（例： LISA_PROP）を配置します。

プロパティがメッセージの XML で使用されていることで、テストの実行時にメッセージを動的に作成できることに注目してください。

［応答メッセージ］タブ

ステップがサブスクライブのために設定されている場合、応答が表示されます。 詳細については、「JMS メッセージング（JNDI） (P. 326)」を参照してください。

TIBCO EMS メッセージング

TIBCO EMS メッセージング ステップでは、トピックおよびキューからメッセージを送受信できます。 また、既存のメッセージを受信、変更、転送できます。

空、テキスト、オブジェクト、バイト、メッセージ、マップ済み（拡張）など、一般的なメッセージ タイプがすべてサポートされています。

TIBCO EMS メッセージング ステップのデフォルト名では、「*EMS キュー名 パブリッシュ*」という命名規則が使用されます。 パブリッシュ キュー名がない場合、デフォルトのステップ名は「EMS キュー名 サブスクライブ」です。 デフォルトのステップ名を別のステップが使用する場合、DevTest は、このステップ名に番号を追加して一意にします。 ステップ名は、いつでも変更できます。

前提条件： このアプリケーションと一緒に DevTest を使用するには、1 つ以上のファイルを DevTest で使用可能にする必要があります。詳細については、「*管理*」の「サードパーティ ファイル要件」を参照してください。

パラメータ要件： このステップには、テスト中のアプリケーションで使用される接続パラメータ、およびサブジェクト名が必要です。 デフォルトは、JNDI サーバ URL の TIBCO_SERVER プロパティです。 このプロパティを使用する場合は、設定に追加する必要があります。 環境に応じて、その他のパラメータが必要となる場合があります。 これらは、アプリケーション開発者から入手します。

パラメータおよびフィールドの詳細については、「JMS メッセージング（JNDI） (P. 326)」を参照してください。

TIBCO ダイレクト JMS

TIBCO ダイレクト JMS ステップでは、JNDI ライブラリを使用せずに、トピックおよびキューからメッセージを送受信できます。また、既存のメッセージを受信、変更、転送できます。

空、テキスト、オブジェクト、バイト、メッセージ、マップ済み（拡張）など、一般的なメッセージ タイプがすべてサポートされています。

TIBCO ダイレクト JMS ステップは、メッセージング要件にかかわらず、単一のエディタを使用して設定されます。 入力オプションはメッセージング要件によって異なります。エディタは有効な設定のみを許可します。一部の機能を有効にすると、その他の機能を使用できなくなる場合があります。

TIBCO ダイレクト JMS ステップのデフォルト名では、「*EMS キュー名 パブリッシュ*」という命名規則が使用されます。 パブリッシュ キュー名がない場合、デフォルトのステップ名は「EMS キュー名 サブスクライブ」です。別のステップもデフォルトのステップ名を使用する場合、DevTest は、番号をステップ名に追加します。ステップ名は、いつでも変更できます。

前提条件： このアプリケーションと一緒に DevTest を使用するには、1 つ以上のファイルを DevTest で使用可能にする必要があります。詳細については、「*管理*」の「サードパーティ ファイル要件」を参照してください。

パラメータ要件： このステップには、テスト中のアプリケーションで使用される接続パラメータ、およびサブジェクト名が必要です。 デフォルトでは、DevTest は JNDI サーバ URL の TIBCO_SERVER プロパティを使用します。 このプロパティを使用する場合は、設定に追加する必要があります。環境に応じて、その他のパラメータが必要となる場合があります。これらは、アプリケーション開発者から入手します。

パラメータおよびフィールドの詳細については、「JMS メッセージング（JNDI） (P. 326)」を参照してください。

Sonic ステップ

以下のステップが使用できます。

SonicMQ メッセージング（ネイティブ） (P. 390)
SonicMQ メッセージング（JNDI） (P. 391)

SonicMQ メッセージング（ネイティブ）

SonicMQ メッセージング（ネイティブ）ステップでは、ネイティブ Sonic プロトコルを使用して、トピックおよびキューからメッセージを送受信できます。 また、既存のメッセージを受信、変更、転送できます。

SonicMQ メッセージング（ネイティブ）では、空、テキスト、オブジェクト、バイト、メッセージ、マップ済み（拡張）など、一般的なメッセージタイプがすべてサポートされています。

SonicMQ メッセージング（ネイティブ）ステップは、メッセージング要件にかかわらず、単一のエディタを使用して設定されます。 入力オプションはメッセージング要件によって異なります。 エディタは有効な設定のみを許可します。そのため、特定の機能を有効にすると、その他の機能が非アクティブになる場合があります。

前提条件： このアプリケーションと一緒に DevTest を使用するには、1 つ以上のファイルを DevTest で使用可能にする必要があります。詳細については、「*管理*」の「サードパーティ ファイル要件」を参照してください。

パラメータ要件： テスト中のアプリケーションで使用される接続パラメータ、およびサブジェクト名が必要です。

テスト中のシステムに対して、4 つのパラメータを使用できます。

- ブローカ ホスト
- ブローカ ポート
- ユーザ
- パスワード

環境に応じて、その他のパラメータが必要となる場合があります。 これらのパラメータは、アプリケーション開発者から入手します。

デフォルトのステップ名： SonicMQ メッセージング（ネイティブ）ステップには、「Sonic キュー名 パブリッシュ」という命名規則を使用したデフォルトの名前があります。 パブリッシュ キュー名がない場合、デフォルトのステップ名は「Sonic キュー名 サブスクライブ」です。 デフォルトのステップ名を別のステップが使用する場合、DevTest は、このステップ名に番号を追加して一意にします。 ステップ名は、いつでも変更できます。

パラメータおよびフィールドの詳細については、「JMS メッセージング（JNDI） (P. 326)」を参照してください。

SonicMQ メッセージング（JNDI）

SonicMQ メッセージング（JNDI）では、空、テキスト、オブジェクト、バイト、メッセージ、マップ済み（拡張）など、一般的なメッセージ タイプがすべてサポートされています。

SonicMQ メッセージング（JNDI）ステップでは、トピックおよびキューからメッセージを送受信できます。また、既存のメッセージを受信、変更、転送できます。

SonicMQ メッセージング（JNDI）ステップは、メッセージング要件にかかわらず、単一のエディタを使用して設定されます。入力オプションはメッセージング要件によって異なります。 エディタは有効な設定のみを許可します。そのため、一部の機能を有効にすると、その他の機能が非アクティブになる場合があります。

前提条件： このアプリケーションと一緒に DevTest を使用するには、1 つ以上のファイルを DevTest で使用可能にする必要があります。詳細については、「*管理*」の「サードパーティ ファイル要件」を参照してください。

パラメータ要件： テスト中のアプリケーションで使用される接続パラメータ、およびサブジェクト名が必要です。 デフォルトでは、DevTest は JNDI サーバ URL の SONICMQ_SERVER プロパティを使用します。 このプロパティを使用する場合は、設定に追加する必要があります。 環境に応じて、その他のパラメータが必要となる場合があります。これらのパラメータは、アプリケーション開発者から入手します。

デフォルトのステップ名： SonicMQ メッセージング（JNDI）ステップには、「Sonic キュー名 パブリッシュ」という命名規則を使用したデフォルトの名前があります。パブリッシュ キュー名がない場合、デフォルトのステップ名は「Sonic キュー名 サブスクライブ」です。 デフォルトのステップ名を別のステップが使用する場合、DevTest は、このステップ名に番号を追加して一意にします。 ステップ名は、いつでも変更できます。

パラメータおよびフィールドの詳細については、「JMS メッセージング（JNDI） (P. 326)」を参照してください。

webMethods ステップ

以下のステップが使用できます。

webMethods ブローカ (P. 393)
webMethods Integration Server サービス (P. 399)

webMethods ブローカ

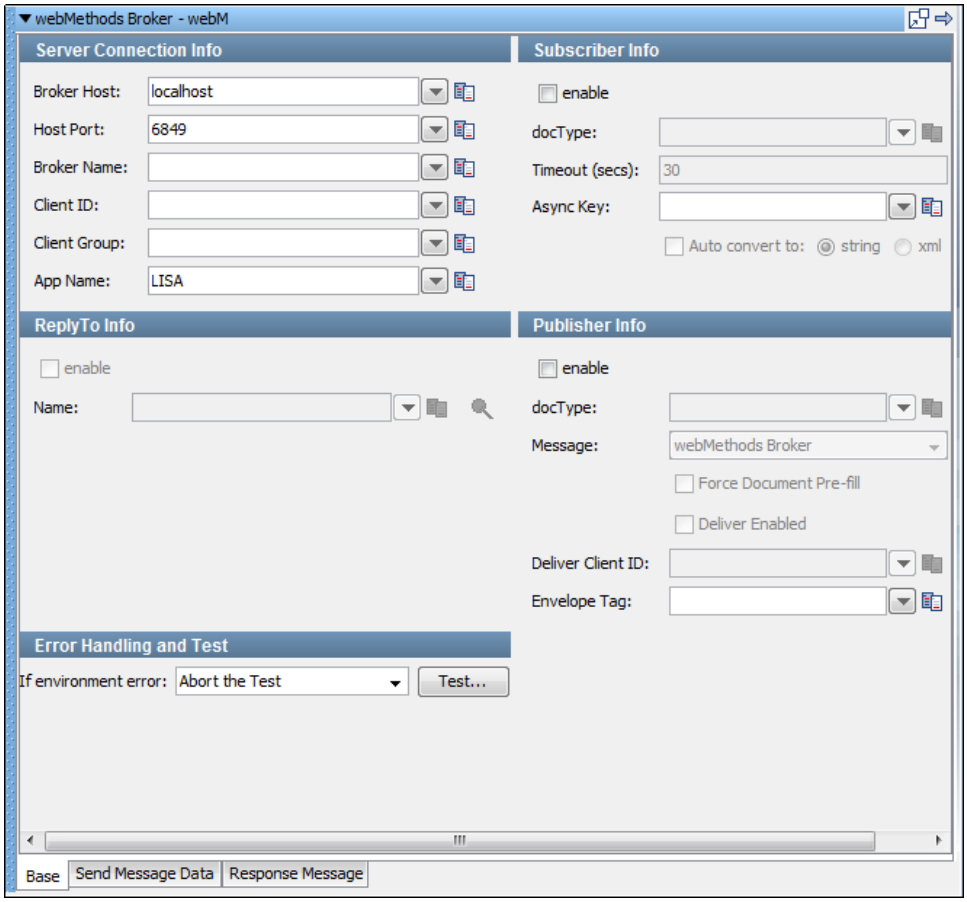
webMethods ブローカでは、ブローカ イベントを作成するマップ済み（拡張）メッセージがサポートされています。

webMethods ブローカ ステップでは、ブローカからメッセージを送受信できます。 また、既存のブローカ イベント/メッセージを受信、変更、転送できます。

webMethods ブローカ ステップは、メッセージング要件にかかわらず、単一のエディタを使用して設定されます。 入力オプションはメッセージング要件によって異なります。 ステップ エディタは有効な設定のみを許可します。そのため、特定の機能を有効にすると、その他の機能が非アクティブになる場合があります。

前提条件： このアプリケーションと一緒に DevTest を使用するには、1 つ以上のファイルを DevTest で使用可能にする必要があります。詳細については、「*管理*」の「サードパーティ ファイル要件」を参照してください。

パラメータ要件： テスト中のアプリケーションで使用される接続パラメータ、およびサブジェクト名が必要です。 以下のセクションでは、必要なパラメータについて説明しています。 環境に応じて、その他のパラメータが必要となる場合があります。これらのパラメータは、アプリケーション開発者から入手します。



webMethods ブローカのメッセージング ステップ エディタには、ページの下部に以下の 3 つのタブがあります。

- ［ベース］タブでは、接続およびメッセージ パラメータを定義します。
- ［メッセージ データの送信］タブでは、メッセージ コンテンツを作成します。
- ［応答メッセージ］タブでは、応答メッセージを POST します。

[ベース]タブ

上記の図で示されている［ベース］タブは、以下のセクションで構成されています。

- サーバ接続情報
- サブスクライバ情報
- パブリッシャ情報
- ReplyTo 情報
- エラー処理およびテスト

［サーバ接続情報］および［エラー処理およびテスト］セクションは常にアクティブです。 ［サブスクライバ情報］、［パブリッシャ情報］、および［返信先情報］を有効または無効にするには、各セクションの左上隅にある［有効］チェック ボックスを使用します。 このチェック ボックスを使用して、ステップをパブリッシュ ステップ、サブスクライブ ステップ、またはその両方に設定できます。 また、ステップに**返信先**コンポーネントを含めることもできます。

テスト ステップの設定が完了したら、［エラー処理およびテスト］セクションの［テスト］をクリックして設定をテストします。

サーバ接続情報

［サーバ接続情報］セクションでは、webMethods ブローカに固有の接続情報を入力します。

テスト中のシステムに対して、4 つのパラメータを使用できます。

ブローカ ホスト

ホスト ポート

ブローカ名

クライアント ID

クライアント グループ

この値は、使用するブローカの送信先を参照できるクライアント グループです。

アプリケーション名

ここでブローカを使用して、アプリケーションを指定します。 このパラメータはオプションであり、多くの場合、デバッグのためにサーバ ログで使用されます。 デフォルトは「DevTest」です。 このパラメータを使用することをお勧めしますが、その他のものをアプリケーション ロジックで使用する必要がある場合、それは可能です。

テスト中のシステムの変更を容易にするには、これらの値を設定のプロパティでパラメータ化します。

パブリッシャ情報

メッセージを送信（パブリッシュ）する機能を設定するには、［有効］チェック ボックスをオンにします。

以下のパラメータを入力します。

ドキュメント タイプ

使用するドキュメント タイプの名前を入力します。

メッセージ

プルダウン メニューから送信するメッセージのタイプを選択します。 サポートされているメッセージは、webMethods ブローカ、オブジェクト、メッセージ、およびマップ済み（拡張）です。

ドキュメントの事前入力を強制

選択したドキュメント タイプは検査され、メッセージは必要なフィールドと共に事前にロードされます。 このチェック ボックスを使用すると、フィールドを変更し、欠落したフィールドを再追加するかどうかを決定できます。DevTest は既存のフィールドを同じ名前で上書きしません。 このプロパティは設計時のみ有効で、テストの実行時には何も行いません。

配信有効

オンにすると、［配信クライアント ID］フィールドが有効になります。

配信クライアント ID

接続用のブローカ クライアント ID 値が NULL の場合、ブローカは識別子を自動的に生成します。 値が別の接続によってすでに使用中の場合、エラーを返すことができます。

エンベロープ タグ

このパラメータにより、ブローカ イベント メッセージに env.tag プロパティを設定できます。

サブスクライバ情報

メッセージを受信 (サブスクライブ) する機能を設定して有効にするには、［有効］チェック ボックスをオンにします。

以下のパラメータを入力します。

ドキュメント タイプ

使用するドキュメント タイプの名前を入力します。

タイムアウト(秒)

DevTest がメッセージの待機を中断するまでの秒数を示します。タイムアウトを設定しない場合は、このフィールドは空白にしておきます。

非同期キー

非同期メッセージを識別するために必要な値を入力します。 この値は非同期モードにのみ必要です。 この値は、非同期メッセージを取得するために後続のメッセージ コンシューマ ステップで使用されます。

自動変換先

ペイロードの文字列表現を返すには、ペイロード オブジェクトに対して toString() 関数を呼び出す文字列を入力します。「xml」は、ペイロードを XML 形式で返します。

ReplyTo 情報

送信先キュー/トピックを設定するには、［有効］チェック ボックスをオンにします。

アプリケーションが送信先を必要とする場合、このセクションで設定されます。

以下のパラメータを入力します。

名前

トピックまたはキューの名前。［検索］アイコン を使用して、トピックまたはキュー名の JNDI サーバを参照します。

エラー処理およびテスト

エラーが発生した場合、［エラー処理およびテスト］オプションはステップにリダイレクトします。

環境エラーの場合

環境エラーが発生した場合に実行するステップまたは実行するアクションを選択します。

ステップ設定をテストするには、［テスト］をクリックします。

[メッセージ データの送信]タブ

ステップがパブリッシュのために設定されている場合、このタブでメッセージを作成できます。

[応答メッセージ]タブ

ステップがサブスクライブのために設定されている場合、応答はここに表示されます。 詳細については、「JMS メッセージング （JNDI） (P. 326)」を参照してください。

デフォルトのステップ名

webMethods ブローカ ステップには、「webM キュー名 パブリッシュ」という命名規則を使用したデフォルトの名前があります。 パブリッシュ キュー名がない場合、デフォルトのステップ名は「webM キュー名 サブスクライブ」です。 デフォルトのステップ名を別のステップが使用する場合、DevTest は、このステップ名に番号を追加して一意にします。 ステップ名は、いつでも変更できます。

webMethods Integration Server サービス

webMethods Integration Server サービス ステップでは、ネイティブ Java API を介して Integration Server サービスを実行できます。 これは、HTTP トランスポートによって公開されていないサービスで動作するように IData オブジェクトを使用して実行されます。

前提条件： このアプリケーションと一緒に DevTest を使用するには、1 つ以上のファイルを DevTest で使用可能にする必要があります。詳細については、「*管理*」の「サードパーティ ファイル要件」を参照してください。

パラメータ要件： テスト中のアプリケーションで使用される接続パラメータ、およびサブジェクト名が必要です。 以下のセクションでは、必要なパラメータについて説明しています。 環境に応じて、その他のパラメータが必要となる場合があります。 これらは、アプリケーション開発者から入手します。

[ベース]タブ: [サーバ接続情報]

以下のパラメータを入力します。

ホスト

ホスト名。

ユーザ

ユーザ ID。

パスワード

パスワード。

パッケージ

サービスが存在するパッケージ。

サービス

コールする実際のサービスの名前。

入力タイプ

[プロパティ]、[IData オブジェクト]、[IData の事前入力を強制]から入力タイプを選択します。

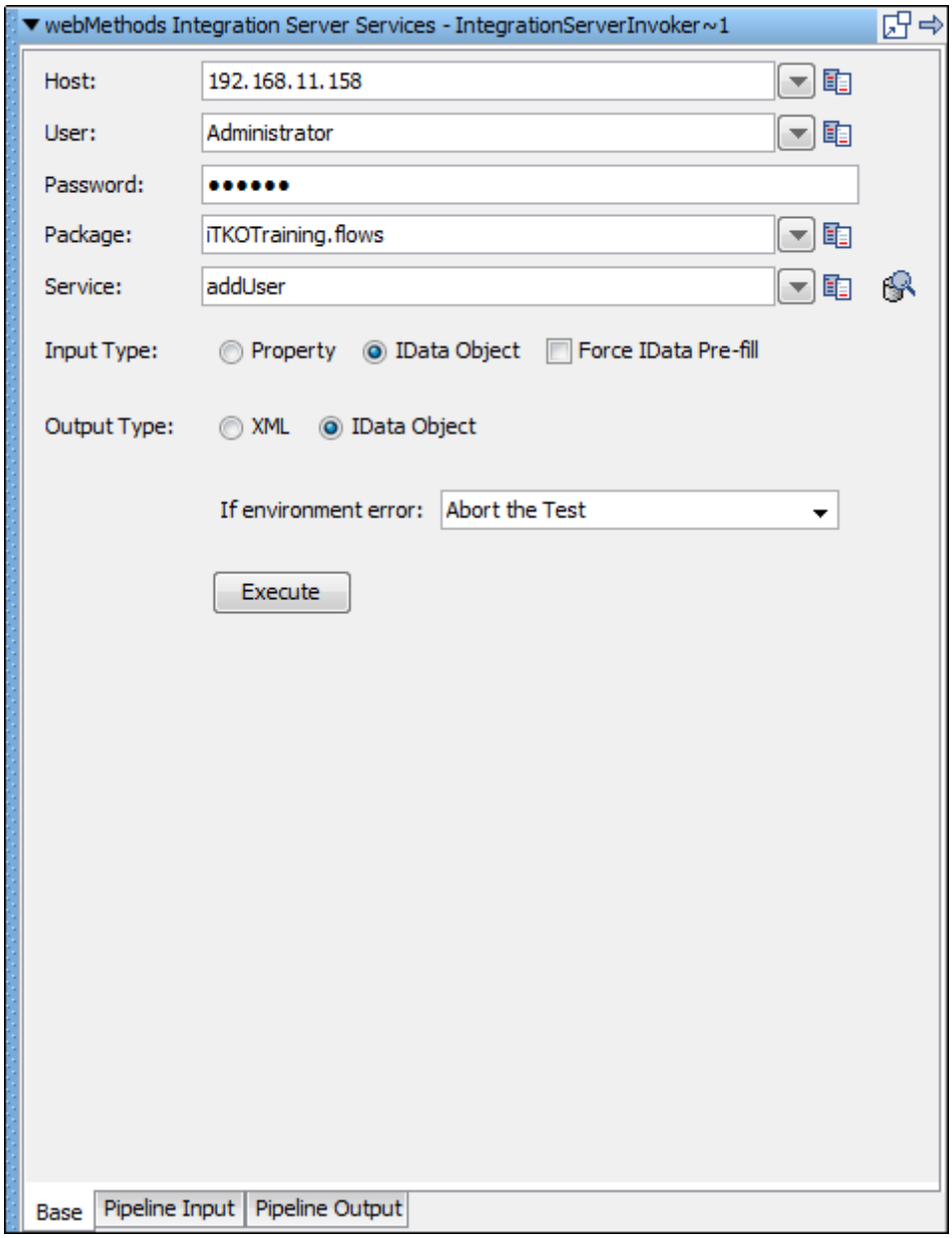
出力タイプ

[XML]または[IData オブジェクト]から出力タイプを選択します。

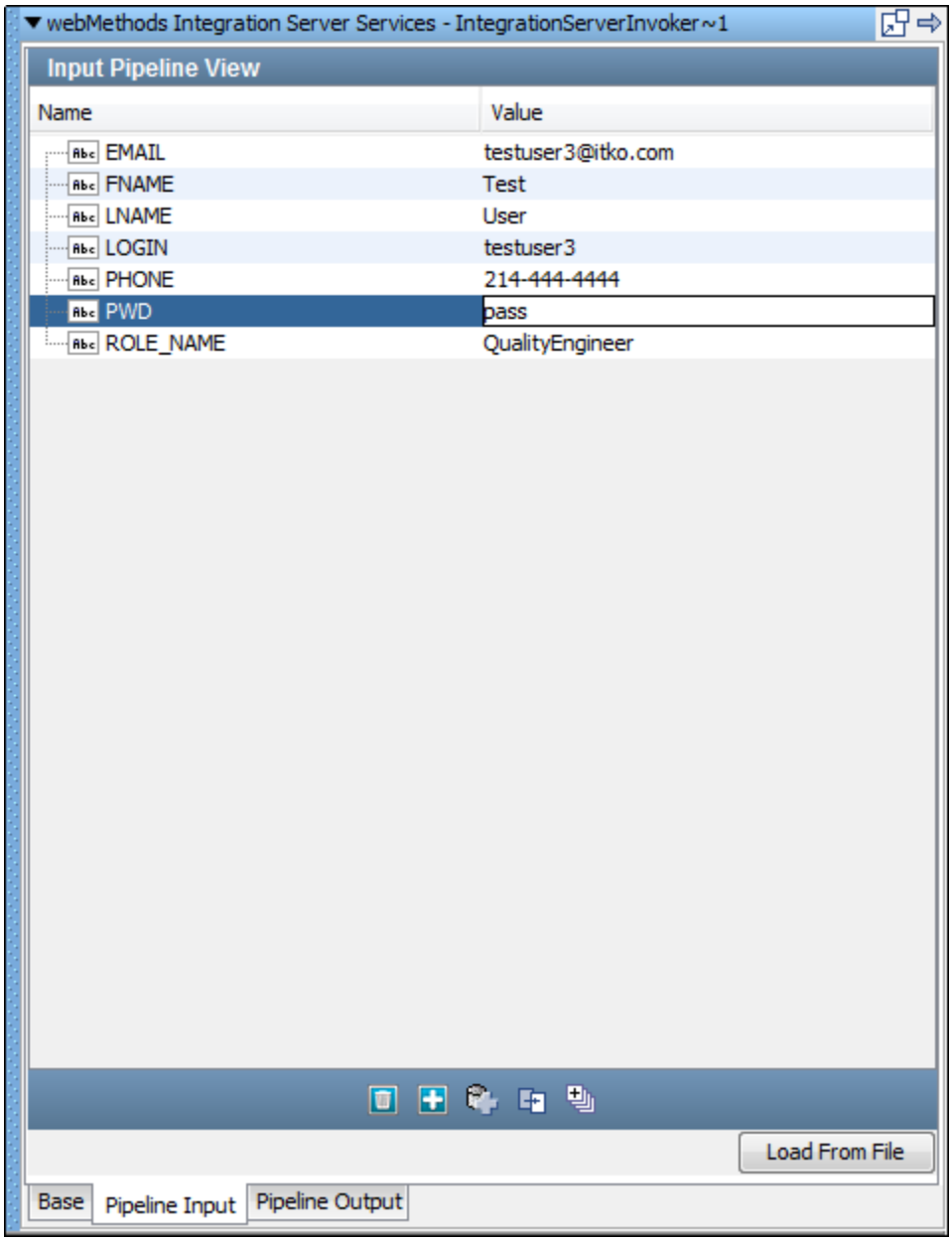
環境エラーの場合

環境エラーが発生した場合に実行するステップまたは実行するアクションを選択します。

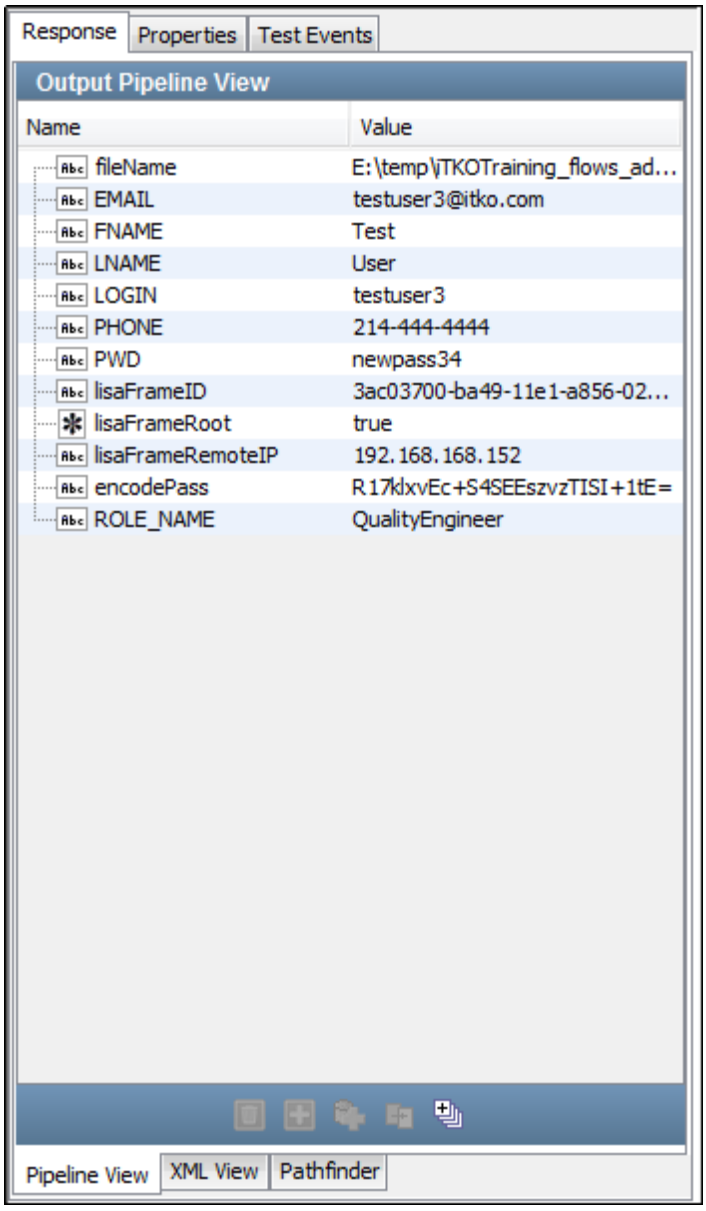
接続するには、[実行]をクリックします。 オブジェクトの応答が表示されます。 応答（これ自体が IData オブジェクト）からペイロードまたはその他のプロパティを取得するには、これを Java 実行ステップにエクスポートします。 このタスクを完了するには、DevTest で Java ステップを作成し、プロパティからロードして、最終応答のステップ名パターンを指定します。 lisa.<ステップ名>.rsp プロパティを使用します。


webMethods Integration Server Services - IntegrationServerInvoker~1
Host:
192.168.11.158
User:
Administrator
Password:
Package:
iTKOTraining.flows
Service:
addUser
Input Type:
Property
IData Object
Force IData Pre-fill
Output Type:
XML
IData Object
If environment error:
Abort the Test
Execute
Base
Pipeline Input
Pipeline Output

[パイプライン入力]タブ


webMethods Integration Server Services - IntegrationServerInvoker~1
Input Pipeline View
Name
Value
EMAIL
testuser3@itko.com
FNAME
Test
LNAME
User
LOGIN
testuser3
PHONE
214-444-4444
PWD
pass
ROLE_NAME
QualityEngineer
Load From File
Base
Pipeline Input
Pipeline Output

[パイプライン出力]タブ



デフォルトのステップ名

webMethods Integration Server サービス ステップには、「IntegrationServerInvoker サービス名@ホスト名」という命名規則を使用したデフォルトの名前があります。 デフォルトのステップ名を別のステップが使用する場合、DevTest は、このステップ名に番号を追加して一意にします。 ステップ名は、いつでも変更できます。

IBM ステップ

以下のステップが使用できます。

IBM WebSphere MQ ステップ (P. 405)

IBM WebSphere MQ ステップ

IBM WebSphere MQ ステップでは、トピックおよびキューに対してメッセージを送受信できます。 また、既存のメッセージを受信、変更、転送できます。

空、テキスト、オブジェクト、バイト、メッセージ、マップ済み（拡張）など、一般的なメッセージ タイプがすべてサポートされています。

IBM WebSphere MQ ステップは、メッセージング要件にかかわらず、単一のエディタを使用して設定されます。 入力オプションはメッセージング要件によって異なります。 エディタは有効な設定のみを許可します。そのため、特定の機能を有効にすると、その他の機能が非アクティブになる場合があります。

IBM WebSphere MQ ステップには、「MQ キュー名 パブリッシュ」という命名規則を使用したデフォルトの名前があります。 パブリッシュ キュー名がない場合、デフォルトのステップ名は「MQ キュー名 サブスクライブ」です。 デフォルトのステップ名を別のステップが使用する場合、DevTest は、このステップ名に番号を追加して一意にします。 ステップ名は、いつでも変更できます。

前提条件： このアプリケーションと一緒に DevTest を使用するには、1 つ以上のファイルを DevTest で使用可能にする必要があります。詳細については、「*管理*」の「サードパーティ ファイル要件」を参照してください。

パラメータ要件: テスト中のシステム用の接続パラメータが必要です。以下のセクションでは、必要なパラメータについて説明しています。

IBM WebSphere MQ ステップ エディタには以下のタブが含まれます。

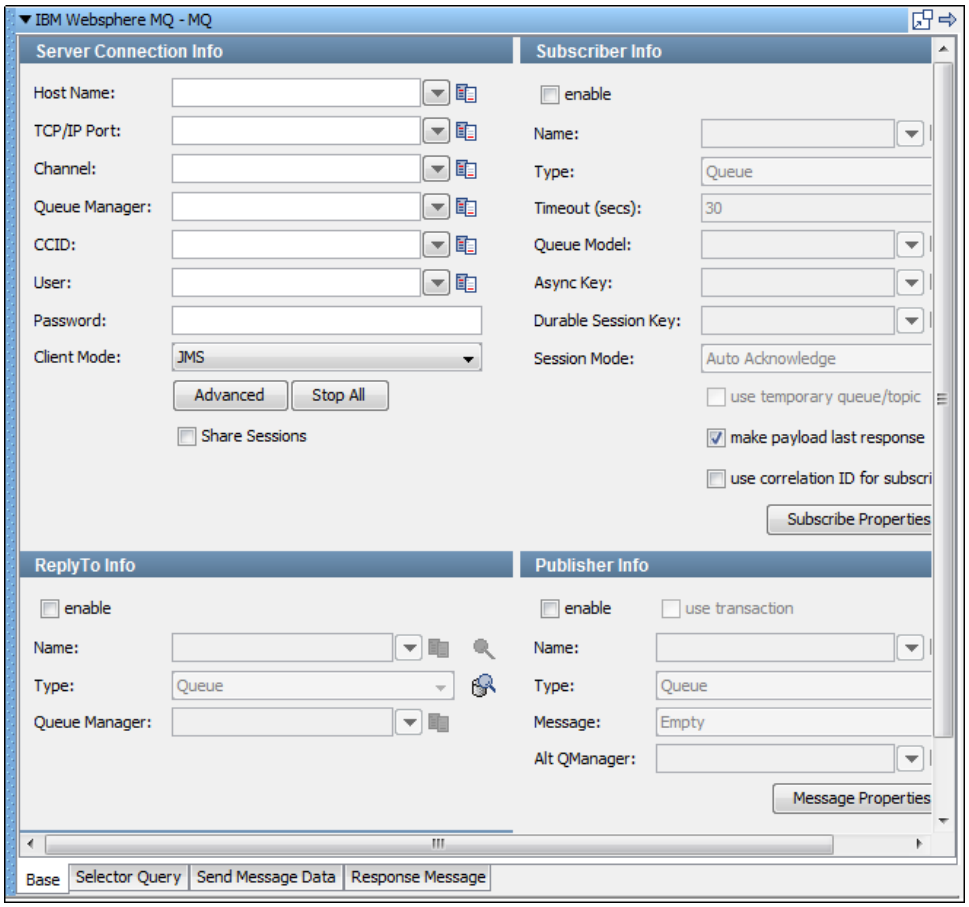
- ［ベース］タブでは、接続およびメッセージング パラメータを定義します。
- ［セレクタ クエリ］タブでは、キュー上のメッセージをリスンする場合に実行されるセレクタ クエリを指定できます。
- ［メッセージ データの送信］タブでは、メッセージ コンテンツを作成します。
- ［応答メッセージ］タブでは、応答メッセージを POST します。

注: このトピックでは、ベース タブについて説明します。その他のタブの詳細については、「JMS メッセージング（JNDI） (P. 326)」を参照してください。

IBM WebSphere MQ: [ベース]タブ

以下の図は、［ベース］タブを示しています。 このタブは、以下のセクションで構成されています。

- サーバ接続情報
- サブスクライバ情報
- ReplyTo 情報
- パブリッシャ情報
- エラー処理およびテスト



［サブスクライバ情報］、［パブリッシャ情報］、および［返信先情報］を有効および無効にするには、各セクションの左上隅にある[有効]チェック ボックスを使用します。 このチェック ボックスを使用して、ステップをパブリッシュ ステップ、サブスクライブ ステップ、またはその両方に設定できます。 また、ステップに返信先コンポーネントを含めることもできます。

テスト ステップの設定が完了したら、［エラー処理およびテスト］セクションの［テスト］をクリックして設定をテストします。

サーバ接続情報

WebSphere MQ に接続するには、以下の情報を入力します。

ホスト名

ホストの名前。

TCP/IP ポート

クライアント接続用のポート。

チャネル

ルーティングおよびメッセージ バスでの管理に使用される接続プロパティ。

キュー マネージャ

ルーティングおよび管理に使用される接続プロパティ。

CCID

接続のオプションであり、クライアント（DevTest）とサーバの間で文字変換を行う必要がある場合のみ適用されます。

ユーザ名

ログイン ユーザ名（該当する場合）。

パスワード

ログイン パスワード（該当する場合）。

クライアント モード

WebSphere MQ サーバと通信する方法を選択できます。

- JMS： JMS 仕様に基づく Pure Java 実装。 この実装には、MQ の代わりに JMS トランスポート プロトコルを使用することをお勧めします。
- ネイティブ クライアント: IBM 固有の API を使用する Pure Java 実装。

- バインディング: WebSphere MQ クライアント インストールからネイティブ ライブラリへのアクセスを必要とします。 これらのライブラリが DevTest アプリケーション ランタイムによってアクセス可能であることを確認します。ほとんどの場合、これらのライブラリを PATH 環境変数で使用可能にしておくことで動作します。

セッションの共有

オンにすると、MQ ネイティブ モードでのすべての共有（接続を含む）を指定します。

パブリッシャ情報

メッセージを送信（パブリッシュ）する機能を設定するには、［有効］チェック ボックスをオンにします。

メッセージを送信する場合にコミットを実行するには、［トランザクションの使用］チェック ボックスをオンにします。

以下のパラメータを入力します。

名前

トピックまたはキューの名前。［検索］アイコン  を使用して、トピックまたはキュー名の JNDI サーバを参照します。

タイプ

トピックまたはキューを使用する場合に選択します。 どのメッセージがキュー（のみ）で処理されるのを待機しているかを確認するには、このフィールドの右にある［検索］アイコン  を使用します。

メッセージ

送信するメッセージのタイプを選択します。 サポートされているタイプは、［なし］、［テキスト］、［オブジェクト］、［バイト］、［メッセージ］、および［マップ済み（拡張）］です。

代替キュー マネージャ

接続するキュー マネージャと異なる場合に、パブリッシュ キューがホストされるキュー マネージャ。

メッセージ プロパティ

メッセージにパブリッシュ プロパティを設定できます。 プロパティのリストは、クライアント モードによって異なります。 詳細については、WebSphere MQ のドキュメントおよび「*JMS and MQ Message Properties*」ナレッジ ベース記事を参照してください。 クライアント モードが JMS の場合、カスタム メッセージ プロパティを追加できます。

サブスクライバ情報

メッセージを受信 (サブスクライブ) する機能を設定して有効にするには、［有効］チェック ボックスをオンにします。

以下のパラメータを入力します。

名前

トピックまたはキューの名前。［検索］アイコン  を使用して、トピックまたはキュー名の JNDI サーバを参照します。

タイプ

トピックまたはキューを使用するかどうか、および同期または非同期モードでリスンするかどうかを選択します。 非同期モードでは、［非同期キー］フィールドにもエントリが必要です。 このフィールドの右側にある［検索］アイコンを使用して、どのメッセージがキュー（のみ）で処理されるのを待機しているかを確認できます。

タイムアウト(秒)

DevTest がメッセージの待機を中断するまでの秒数を示します。タイムアウトを設定しない場合は、このフィールドは空白にしておきます。

キュー モデル

MQ は、一時送信先を作成するためにこの値を必要とします。 キュー モデルは MQ サーバで設定され、［一時キュー/トピックを使用する］がオンの場合にのみアクティブになります。この場合、［返信先情報］セクションは無効です。

非同期キー

非同期メッセージを識別するために必要な値を入力します。 この値は非同期モードにのみ必要です。 この値は、非同期メッセージを取得するために後続のメッセージ コンシューマ ステップで使用されます。

持続セッション キー

ここで名前を入力することによって、持続セッションを要求します。 また、そのセッションのキーも指定します。 持続セッションでは、ログアウトした後に再度ログインしても、トピックからのメッセージをすべて受信できます。

セッション モード

ドロップダウン リストをクリックして、使用可能なオプションから適切なモードを選択します。オプションは次のとおりです:［自動確認応答］、［クライアント確認応答］、［トランザクションの使用］、［自動（重複 OK）］。

- 自動確認応答 : セッションは、クライアントによって受信されたメッセージを自動的に確認します。
- クライアント確認応答 : このオプションは、メッセージをプログラム的に確認するようにクライアントに指示します。
- トランザクションの使用 : メッセージを受信した場合に、トランザクションを使用してコミットを実行します。
- 自動（重複 OK） : このオプションは、メッセージの配信を確認するようにセッションに指示します。

一時キュー/トピックを使用する

JMS プロバイダに一時キュー/トピックを設定させるには、［一時キュー/トピックを使用する］チェック ボックスをオンにします。一時キュー/トピックが使用される場合、一時キュー/トピックに送信するメッセージの JMS ReplyTo パラメータが自動的に設定されます。 返信を送信できるように、一時キュー/トピック機能とパブリッシャを常に一緒に使用する必要があります。 一時キュー/トピックを使用する場合、［返信先］セクションは無効です。

ペイロードを最終応答にする

ペイロードを最終応答にするには、このオプションを選択します。

サブスクライブに相関 ID を使用

［サブスクライブ プロパティ］の値を使用して、受信メッセージのフィルタリングを有効にします。

サブスクライブ プロパティ

メッセージにサブスクライブ プロパティを設定できます。 詳細については、WebSphere MQ のドキュメントおよび「JMS and MQ Message Properties」ナレッジ ベース記事を参照してください。

ReplyTo 情報

送信先キュー/トピックを設定するには、［有効］チェック ボックスをオンにします。

アプリケーションが送信先を必要とする場合、このセクションで設定されます。

以下のパラメータを入力します。

名前

トピックまたはキューの名前。［検索］アイコン  を使用して、トピックまたはキュー名の JNDI サーバを参照します。

タイプ

トピックまたはキューを使用する場合に選択します。 どのメッセージがキュー（のみ）で処理されるのを待機しているかを確認するには、このフィールドの右にある［検索］アイコン  を使用します。

キュー マネージャ

（このステップで）返信先をパブリッシャと異なるキュー マネージャに設定することができます。

エラー処理およびテスト

エラーが発生した場合、［エラー処理およびテスト］セクションはステップにリダイレクトします。

環境エラーの場合

環境エラーが発生した場合に実行するステップまたは実行するアクションを選択します。

ステップ設定をテストするには、［テスト］をクリックします。

SAP ステップ

以下のステップが使用できます。

SAP RFC 実行 (P. 413)
SAP IDoc 送信側システム (P. 416)
SAP IDoc ステータス取得 (P. 419)

SAP RFC 実行

SAP RFC 実行ステップでは、SAP システムに接続して RFC （リモート ファンクション コール）を実行できます。

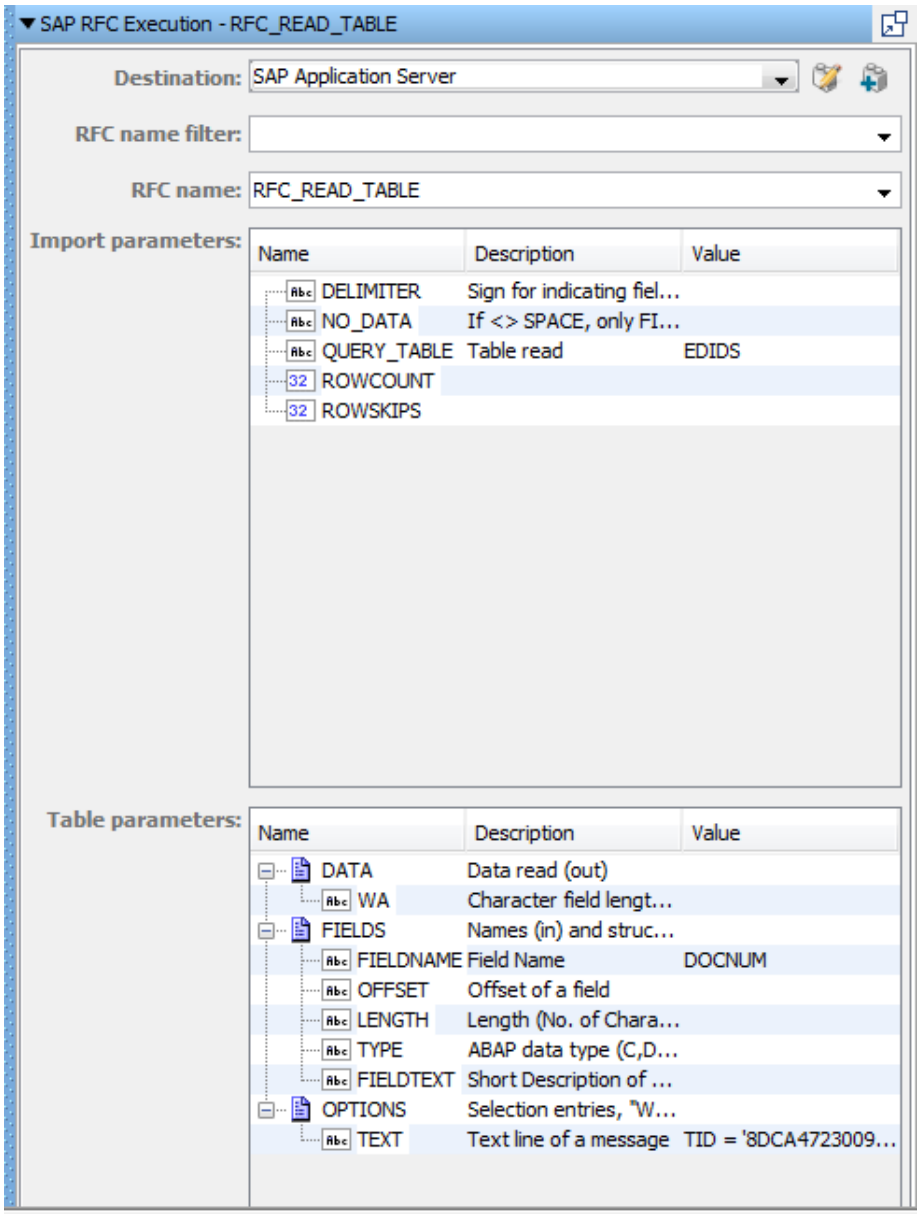
前提条件： このアプリケーションと一緒に DevTest を使用するには、1 つ以上のファイルを DevTest で使用可能にする必要があります。詳細については、「*管理*」の「サードパーティ ファイル要件」を参照してください。

SAP RFC 実行ステップを作成する方法

1. アプリケーション サーバまたはメッセージ サーバのいずれかの接続情報が含まれる送信先アセット (P. 74)を選択します。
2. メッセージ サーバを使用している場合

   Windows で、C:¥Windows¥System32¥drivers¥etc¥services の *sapmsCR2 3600/tcp* ファイルの末尾に以下の行を追加します。

   Linux で、/etc/services の *sapmsCR2 3600/tcp* の末尾に以下の行を追加します。



3. SAP RFC 実行ステップ エディタで以下のパラメータを入力します。RFC 入力パラメータにプロパティを使用できます。

   RFC 名フィルタ

   RFC 関数名をフィルタするには、フィルタ値を入力します。フィルタ値には、ワイルドカード文字 (*) を含めることができます。[RFC 名] フィールドのドロップダウン矢印をクリックすると、フィルタによって絞り込まれた RFC 名を取得するために、DevTest が SAP システムに要求を送信し、フィールドに入力します。

RFC 関数名を選択または入力すると、DevTest は RFC 関数の入力パラメータを取得するために SAP システムに要求を送信します。 各パラメータのキーおよび説明が SAP システムから返されます。 説明はほとんどのパラメータで使用可能ですが、すべてのパラメータに説明があるとは限りません。 RFC 関数を実行する前にパラメータ値を入力できます。

RFC 名

Field Description

インポート パラメータ

キー

パラメータの名前

説明

パラメータの説明

値

パラメータの値

値列の各テーブル セルは、使用可能なプロパティのドロップダウン リストです。

テーブル パラメータ

キー

パラメータの名前

説明

パラメータの説明

値

パラメータの値

値列の各テーブル セルは選択可能なプロパティを示すドロップダウンです。

環境エラーの場合

環境エラーが発生した場合に実行するステップまたは実行するアクションを選択します。

4. ［テスト］をクリックし、［応答］タブに入力します。このタブには、XML および DOM ツリーの出力用のサブタブが含まれます、

SAP IDoc 送信側システム

SAP IDoc 送信側システム ステップでは、SAP システムに接続して SAP IDoc を送信できます。

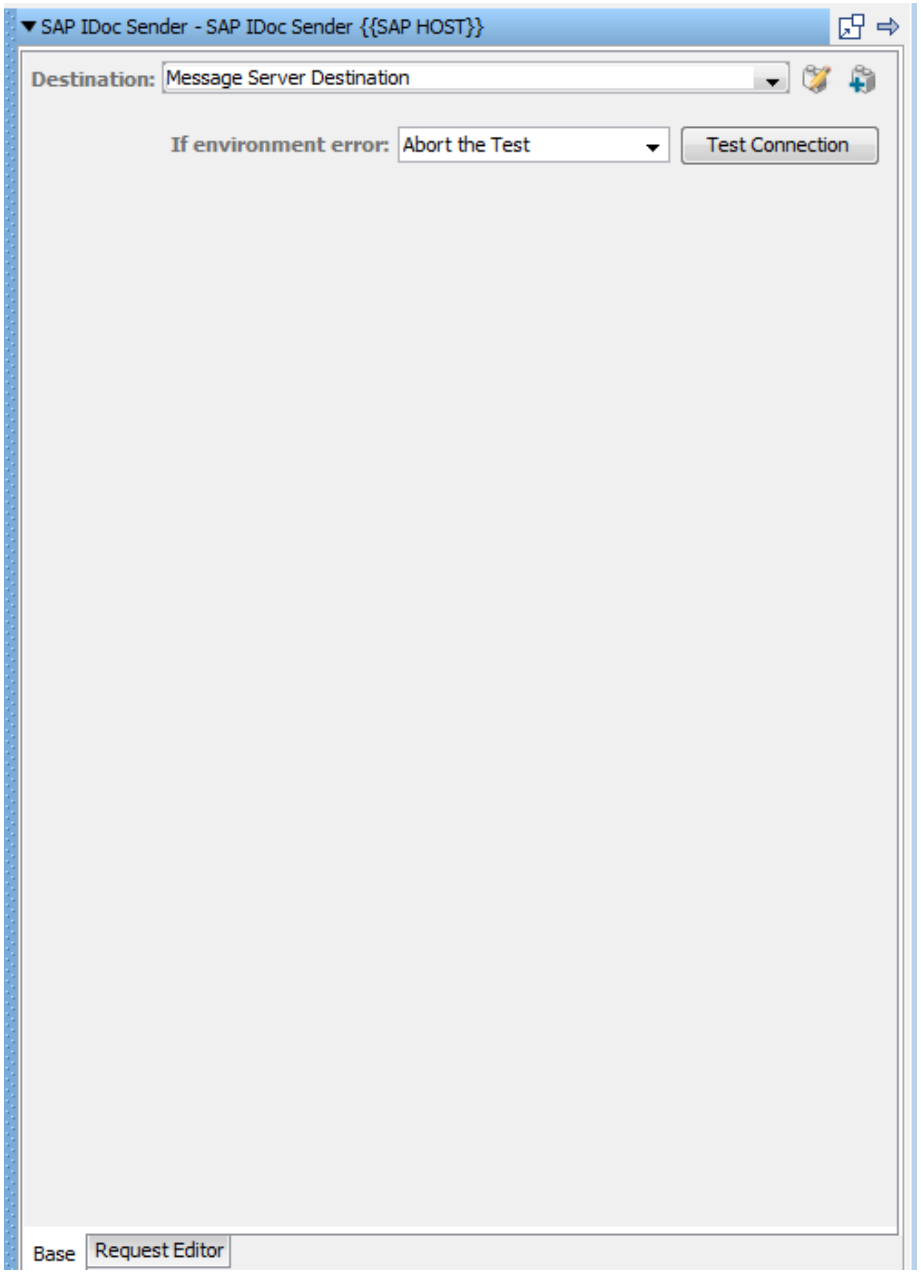
前提条件： このアプリケーションと一緒に DevTest を使用するには、1 つ以上のファイルを DevTest で使用可能にする必要があります。詳細については、「*管理*」の「サードパーティ ファイル要件」を参照してください。

SAP IDoc 送信側システム ステップを作成する方法

1. アプリケーション サーバまたはメッセージ サーバのいずれかの接続情報が含まれる送信先アセット (P. 74) を選択します。
2. メッセージ サーバを使用している場合

   Windows で、C:¥Windows¥System32¥drivers¥etc¥services の *sapmsCR2 3600/tcp* ファイルの末尾に以下の行を追加します。

   Linux で、/etc/services の *sapmsCR2 3600/tcp* の末尾に以下の行を追加します。


SAP IDoc Sender - SAP IDoc Sender {{SAP HOST}}
Destination:
Message Server Destination
If environment error:
Abort the Test
Test Connection
Base
Request Editor

3. SAP IDoc 送信側システム ステップ エディタで以下のパラメータを入力します。

   環境エラーの場合

   環境エラーが発生した場合に実行するステップまたは実行するアクションを選択します。

4. SAP サーバへの接続をテストするには、［テスト接続］をクリックします。

5. ファイル ロケータを表示するには、［ファイルから IDoc を読み取る］をクリックします。 IDoc の場所に移動し、それを選択します。

   SAP IDoc では、XML および RAW テキスト形式がサポートされています。

SAP IDoc ステータス取得

SAP IDoc ステータス取得ステップでは、SAP システムに接続して、完了または指定された間隔の終了まで、SAP IDoc のステータスを定期的にポーリングできます。

前提条件： このアプリケーションと一緒に DevTest を使用するには、1 つ以上のファイルを DevTest で使用可能にする必要があります。詳細については、「*管理*」の「サードパーティ ファイル要件」を参照してください。

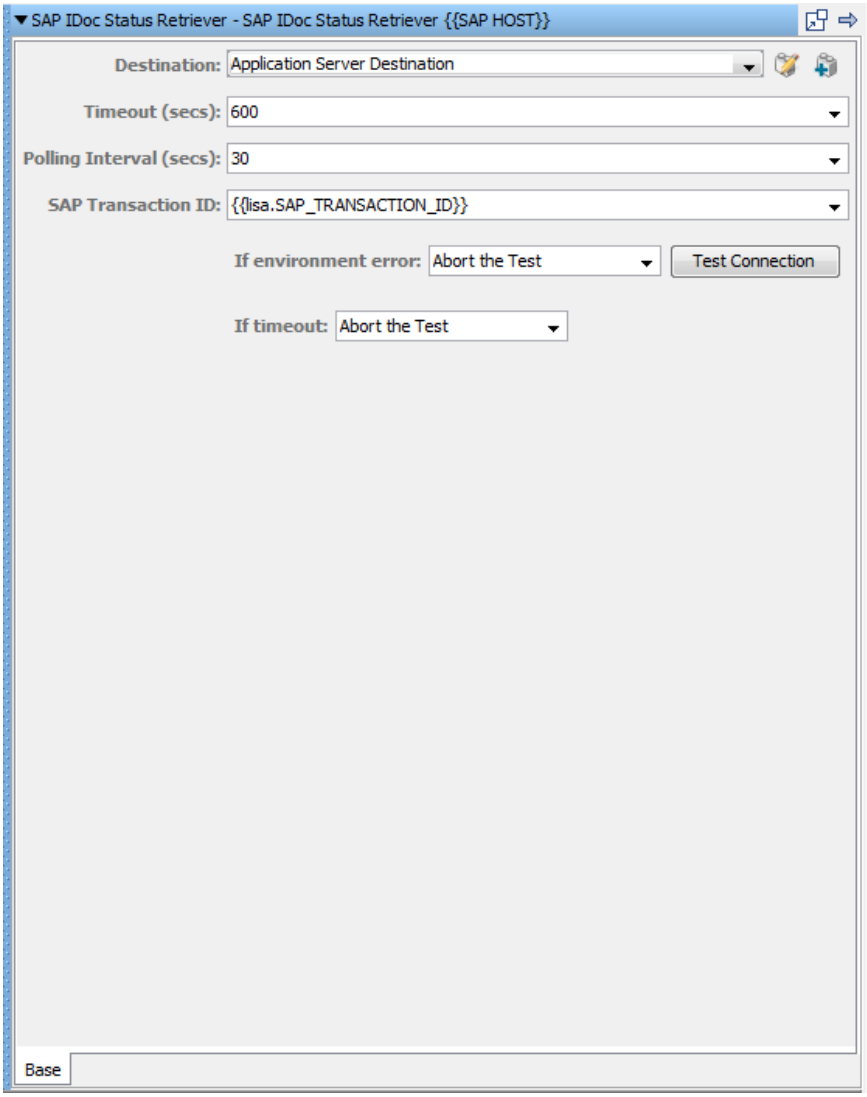
SAP IDoc ステータス取得ステップを作成する方法

1. アプリケーション サーバまたはメッセージ サーバのいずれかの接続情報が含まれる送信先アセット (P. 74) を選択します。
2. メッセージ サーバを使用している場合

   Windows で、C:¥Windows¥System32¥drivers¥etc¥services の *sapmsCR2 3600/tcp* ファイルの末尾に以下の行を追加します。

   Linux で、/etc/services の *sapmsCR2 3600/tcp* の末尾に以下の行を追加します。

図 1: SAP IDoc ステータス取得ステップの［ベース］タブ


SAP IDoc Status Retriever - SAP IDoc Status Retriever {{SAP HOST}}
Destination:
Application Server Destination
Timeout (secs):
600
Polling Interval (secs):
30
SAP Transaction ID:
{{lisa.SAP_TRANSACTION_ID}}
If environment error:
Abort the Test
Test Connection
If timeout:
Abort the Test
Base

3. SAP IDoc ステータス取得ステップ エディタで以下のパラメータを入力します。 SAP 入力パラメータにプロパティを使用できます。

   タイムアウト（秒）

   SAP IDoc ステータスのポーリング操作の期間

   ポーリング間隔（秒）

   SAP IDoc ステータスのポーリング操作の頻度

   たとえば、デフォルト値を使用すると、SAP IDoc ステータス取得ステップは 30 秒ごとに 合計 10 分間、IDoc ステータスをポーリングします。 IDoc が期間内に完了した場合、SAP IDoc ステータス取得ステップは 10 分間を待たずにすぐに停止します。

   SAP トランザクション ID

   英数字のトランザクション ID または DevTest プロパティ。SAP IDoc 送信側システム ステップをこの IDoc ステータス取得ステップの前に使用する場合は、デフォルトのプロパティ {{lisa.SAP_TRANSACTION_ID}} を使用します。

   環境エラーの場合

   環境エラーが発生した場合に実行するステップまたは実行するアクションを選択します。

   タイムアウトの場合

   タイムアウトの場合に実行するステップまたは実行するアクションを選択します。

4. SAP サーバへの接続を検証するには、［テスト接続］をクリックします。

Selenium 統合ステップ

Selenium ステップでは、Web ベースのユーザ インターフェースのテスト スクリプトを Selenium Builder から CA Application Test にインポートできます。エクスポートされた JSON テスト スクリプトを使用して、ユーザ インターフェースをテストするためのテスト ステップを作成できます。

Selenium ステップを作成するには、以下のタスクを完了します。

- Selenium Builder のレコーディングの作成およびエクスポート（P. 424）
- Selenium Builder の JSON の CA Application Test へのインポート

注: Selenium 統合テストのレコーディングをサポートするブラウザは、Firefox に制限されています。 これらのステップを CA Application Test にインポートした後、Google Chrome、Mozilla Firefox 24 以降、または Internet Explorer 8.0 以降でテスト ケースを実行できます。

また、JSON スクリプトに CA Application Test Selenium テスト ケースをエクスポートすることもできます。 詳細については、以下を参照してください。

- JSON スクリプトへの Selenium テスト ケースのエクスポート（P. 429）

Selenium Web ドライバの詳細オプションの設定

ユーザの要件を満たす、Selenium Web ドライバの詳細オプションを設定できます。

次の手順に従ってください：

1. LISA_HOME ディレクトリにあるサンプル プロジェクトの Data ディレクトリで、**selenium-capabilities.conf** を見つけます。

   このファイルには、Selenium Web ドライバを設定するためのサンプル パラメータが含まれます。これらのパラメータに関する Selenium からの最新情報については、ファイルに含まれている URL を参照してください。

   ファイルはプレーン テキスト ファイルです。 メモ帳（Windows）や vi（UNIX および Linux）などの、任意のテキスト エディタを使用し、ファイルを表示して編集します。

2. 更新が必要なパラメータすべての値を入力します。

3. Selenium ステップのパラメータを更新するには、local.properties ファイルまたはアクティブなプロジェクト設定ファイルで、**selenium.WebDriver.DesiredCapabilities.filePath** プロパティを使用して **selenium-capabilities.conf** の場所を指定します。

   デフォルトの場所を使用するには、selenium-capabilities.conf ファイルをプロジェクトの Data ディレクトリにコピーします。

Selenium Builder のレコーディングの作成およびエクスポート

Selenium Builder は、Web ベース ユーザのインターフェースでアクションを記録し、Selenium テストを作成できる Firefox アドオンです。 CA Application Test で Selenium テスト ステップを作成する最初の手順は、Selenium Builder テスト スクリプトを作成することです。

Selenium Builder がインストールされていない場合は、http://sebuilder.github.io/se-builder/ からダウンロードしてインストールします。

注: 自動アップグレード オプションをオフにすることを推奨します。

a. Firefox のメイン メニューの［Firefox］-［Add Ons］をクリックします。

   アドオン マネージャが開きます。

b. ［Extensions］をクリックし、［Selenium Builder］をダブルクリックします。

c. ［Automatic Updates］の［Off］オプションを選択します。

次の手順に従ってください:

1. Firefox ブラウザを開きます。
2. Firefox のメイン メニューの ［Firefox］-［Web Developer］-［Launch Selenium Builder］をクリックします。

   Selenium Builder ページが表示されます。

   注: アドオン バーを有効にしておく場合、Firefox ブラウザ ウィンドウの右下隅にある  をクリックします。
3. テストする Web アプリケーションの URL を［**Start recording at**］フィールドに入力します。
4. ［Selenium 2］をクリックします。

   注: ［Selenium 1］は、JSON にエクスポートするために必要な機能をサポートしていません。
5. Web アプリケーションに移動し、テストするアクションを実行します。
6. 終了したら、Selenium Builder ページに戻り、［Stop Recording］をクリックします。

注: Selenium Builder テストの作成の詳細については、http://sebuilder.github.io/se-builder/ にある Selenium Builder のドキュメントを参照してください。

7. ［File］-［Export］をクリックします。

   ［Choose Export Format］ダイアログ ボックスが表示されます。

8. ［Save as JSON］をクリックします。
9. JSON ファイルを保存するディレクトリを参照します。
10. JSON ファイルの名前を入力し、［Save］をクリックします。
11. スクリプトが有効であることを検証するには、Selenium Builder でスクリプトを実行します。
12. ［Run］-［Run test locally］をクリックします。

    結果と、スクリプト エラーがあれば表示されます。

Selenium Builder の JSON の CA Application Test へのインポート

Selenium 統合テスト ステップを使用して、Selenium Builder で作成した JSON テスト スクリプト (P. 424)をインポートし、その JSON ファイルに基づいてテスト ステップまたはスクリプトを作成することができます。

次の手順に従ってください:

1. 既存のテスト ケースを開くか、または新しいテスト ケースを作成します。
2. 以下のいずれかの操作を実行します。
   - ［Selenium］  をクリックします。
   - ［ステップの追加］  をクリックしてから、［Selenium］-［Selenium Import/Export］をクリックします。

   ［Selenium 統合］ページの［インポート］タブが表示されます。
3. ［入力 JSON スクリプト］フィールドの隣の［参照］をクリックし、インポートする JSON テスト スクリプトの場所を参照します。
4. 以下の出力オプションのいずれかを選択します。

   Selenium スクリプト

   1 つのテスト ステップで完全な JSON テスト スクリプトをインポートします。 このオプションは、Selenium Builder スイートもサポートしています。

   Selenium ステップ

   JSON スクリプトを分割し、スクリプト内の各アクションに対して個別のテスト ステップを作成します。このオプションは、Selenium Builder スイートをサポートしていません。
5. ［作成］をクリックします。

   1 つの新しいステップ（Selenium ステップ）または複数の新しいステップ（Selenium スクリプト）がテスト ケースに表示されます。

   ［Selenium 統合インポート］ダイアログ ボックスが表示されます。このページは、インポート中に発生した警告またはエラーをユーザに通知します。また、インポート プロセスによって生成されたステップの数も示します。
6. ［閉じる］をクリックします。

7. 各ステップをダブルクリックして、右側のパネルのエレメント ツリーの［Selenium スクリプト］または［Selenium ステップ］タブに JSON スクリプトを表示します。
8. 各ステップの以下のフィールドに入力します。

Alert Behavior（**オプション**）

Selenium ステップでは、「インライン」アラート処理を提供できます。 ［Alert Behavior］領域内のフィールドは、**unexpectedAlertBehaviour** パラメータ（**selenium.WebDriver.DesiredCapabilities.filePath** プロパティによって名前が付けられたファイル内で定義）の値と同時に機能します。

これらのフィールドでは、Web アプリケーションに表示されたモーダルなアラート ダイアログ ボックスに、テスト ステップがどのように反応するかを指定できます。

Alert Action（**オプション**）

モーダルなアラート ダイアログ ボックスに対応して実行するアクションを定義します。

値： Accept、Dismiss、Answer

入力テキスト（オプション）

［Alert Action］が［Answer］の場合に入力するテキストを定義します。 ステップでは、このテキスト ボックスにユーザが入力したテキストが入力されてから、［OK］がクリックされます。

ステップの失敗時

テスト ケースの特定のステップが失敗した場合に実行するアクションを定義します。 実行するステップ（移動先）またはステップが失敗した場合に実行するアクションを選択します。 詳細については、「次のステップの設定」または「警告およびエラーの生成」を参照してください。

注: JSON スクリプトは、ユーザが入力する値に対してプロパティによる変数の置換をサポートしています。 また、機密データを公開しないようにするために、{{password_enc}} などの暗号化された値を持ったプロパティを使用できます。

以下の Selenium Builder ステップがテスト ケースにマップされます。

ストア

名前/値ペアはテスト ケースの標準プロパティになります。ほかのプロパティと区別するために、名前の先頭に「selenium」が付けられます。たとえば、以下の JSON 定義は **selenium.window_title** という名前の新しいプロパティになります。ステップが実行された後、値が入力されます。

```
{
  "type": "storeTitle",
  "variable": "window_title"
},
```

検証

Selenium Builder の検証ステップは、ユーザ インターフェース エレメントを検証するために使用されます。検証が失敗すると、ステップの状態は「エラー」に設定されますが、テスト ケース実行フローは次のステップに進みます。関連する DevTest エラー イベント（赤色）が失敗に対して作成されます。

Assertion（アサーション）

Selenium Builder のアサーション ステップは、ユーザ インターフェース エレメントを検証します。 検証が失敗すると、ステップの状態は「失敗」に設定され、テスト ケース実行フローは停止します。 関連する DevTest エラー イベント（赤色）が失敗に対して作成されます。

スクリーンショットの保存

スクリーンショットを保存するためのフル パスが、スクリプトの［スクリーンショットの保存］ステップに含まれない場合、DevTest は $LISA_HOME¥tmp¥selenium ディレクトリにファイルを作成しようとします。また、ファイル名に変数を使用することもできます。以下に例を示します。

c:¥testcase1¥snapshot1-{{LISA_TEST_RUN_ID}}.png

or

c:¥{{testCase}}¥{{LISA_TEST_RUN_ID}}¥snapshot1.png

ターゲット ファイルの親ディレクトリの一部が存在しない場合、ディレクトリのその部分が自動的に作成されます。 ターゲット ファイルがすでに存在する場合、新しいデータを書き込む前に削除されます。

JSON スクリプトへの Selenium テスト ステップが含まれるテスト ケースのエクスポート

Selenium ステップが含まれるテスト ケースを、ファイル システム上に格納できる JSON スクリプトにエクスポートすることができます。

Selenium テスト ケースをエクスポートする前に、DevTest プラグインを Firefox Selenium Builder の plugins ディレクトリにインストールする必要があります。

テスト ケースをエクスポートした後、Selenium Builder を使用して Web アプリケーションを再記録することができます。 その後、新しく生成された JSON を Selenium Builder から DevTest ワークステーション に再インポートし、変更を既存のテスト ケースにマージすることができます。

エクスポートされた JSON スクリプトを、［**Open a script exported by CA Application Test**］メニュー項目を使用して Selenium Builder にロードします。このメニュー項目は、DevTest プラグインをインストールした場合にのみ利用できます。

Selenium Builder に JSON スクリプトをロードすると、Selenium Builder の標準機能を使用して、以下のような更新を実行できます。

- ステップの編集
- ステップの追加
- ステップの削除
- ステップの実行順序の変更

DevTest プラグインをインストールする方法

1. DevTest ワークステーション の [LISA_HOME]¥addons¥sebuilder-plugin ディレクトリから [Firefox profile]¥SeBuilder¥plugins ディレクトリに、**lisa** ディレクトリおよびそのコンテンツ全体をコピーします。 Firefox プロファイルを見つけるには、以下の手順に従います。
   a. Firefox 3.6 以降の場合は、Firefox で、［ヘルプ］-［トラブルシューティング情報］を選択します。
   b. ［アプリケーション基本情報］で、以下の手順に従います。
      - Windows および Linux では、Firefox のバージョンに応じて、［フォルダを開く］（Windows）、［ディレクトリを開く］（Linux）、または［Open Containing Folder］をクリックします。

- OS x では、［Finder で表示］をクリックします。

  注: Firefox メニュー バーには［ファイル］、［編集］、［表示］、［履歴］、［ブックマーク］、［ツール］、および［ヘルプ］メニュー項目があります。 Windows では、メニュー バーが非表示の場合があります。 非表示のメニュー バーを一時的に表示するには、Alt キーを押します。

2. **lisa** ディレクトリを SeBuilder/plugins ディレクトリにコピーします。

Selenium テスト ケースをエクスポートする方法

1. 既存のテスト ケースを開きます。
2. 以下のいずれかの操作を実行します。

   - ［Selenium］  をクリックします。
   - ［ステップの追加］  をクリックしてから、［Selenium］-［Selenium Import/Export］をクリックします。

   ［Selenium 統合］ページが表示されます。
3. ［エクスポート］タブをクリックします。
4. ［Output JSON Script］フィールドの隣の［参照］をクリックし、エクスポートする JSON テスト スクリプトの場所を参照するか、またはパスおよびファイル名を入力します。
5. ［保存］をクリックします。

LISA 仮想サービス環境ステップ

CA Service Virtualization テスト ステップについては、「*CA Service Virtualization の使用*」の「VSM の編集」を参照してください。

CAI ステップ

以下のステップが含まれます。

トランザクション フレームの実行 (P. 431)

トランザクション フレームの実行

トランザクション フレームの実行ステップでは、DevTest Java エージェントでトランザクション フレームを実行することができます。

このステップはテスト中のシステムの機能を検証する場合に役立ちますが、パブリック アクセス ポイントはありません。また、このステップは、自動的にベースラインに含めることができます。

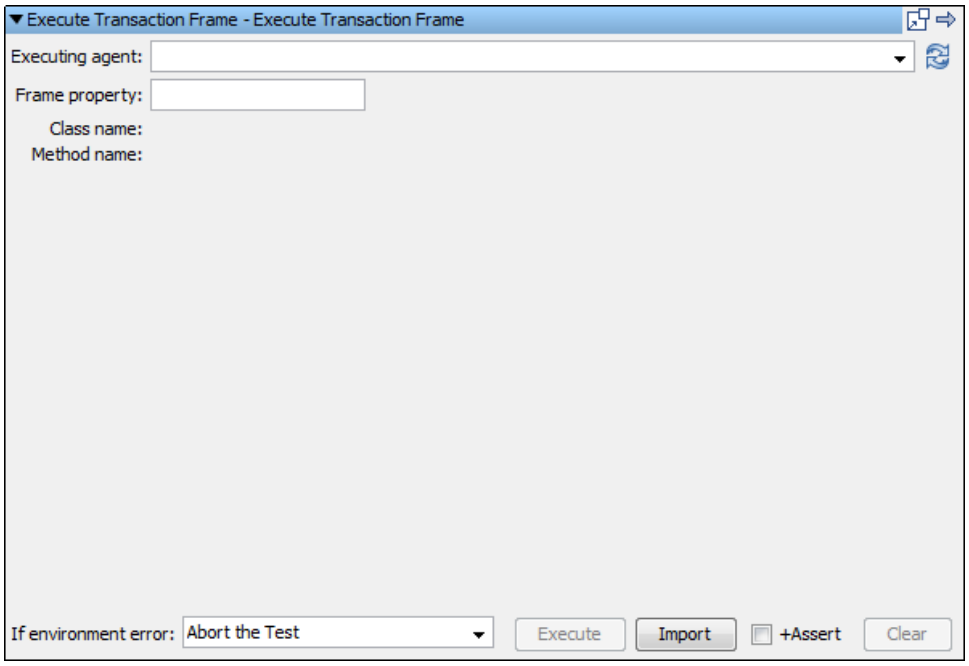
このステップは、エージェントがキャプチャできる任意のタイプのトランザクション フレームで使用できます。

テスト ケースにステップを手動で追加する場合は、以下のいずれかを実行します。

- トランザクションのインポート
- フレームを検索するプロパティの指定

この両方を行うと、DevTest は、まずプロパティを確認し、プロパティがフレームを参照していない場合にのみ、インポートされたトランザクションを実際のフレームまたはその XML 表現のいずれかとして使用します。フレームは、XML ファイルまたは ZIP ファイルからインポートできます。フレームは、CAI コンソールからのエクスポート によって取得できます。

以下の図は、このステップの初期ビューを示しています。 フレームを指定するには、［インポート］をクリックします。インポート プロセスで、インポートされたフレームの応答に対するアサーションを作成し、それをステップに自動的に追加するには、［インポート］をクリックする前に［アサート］ チェック ボックスをオンにします。

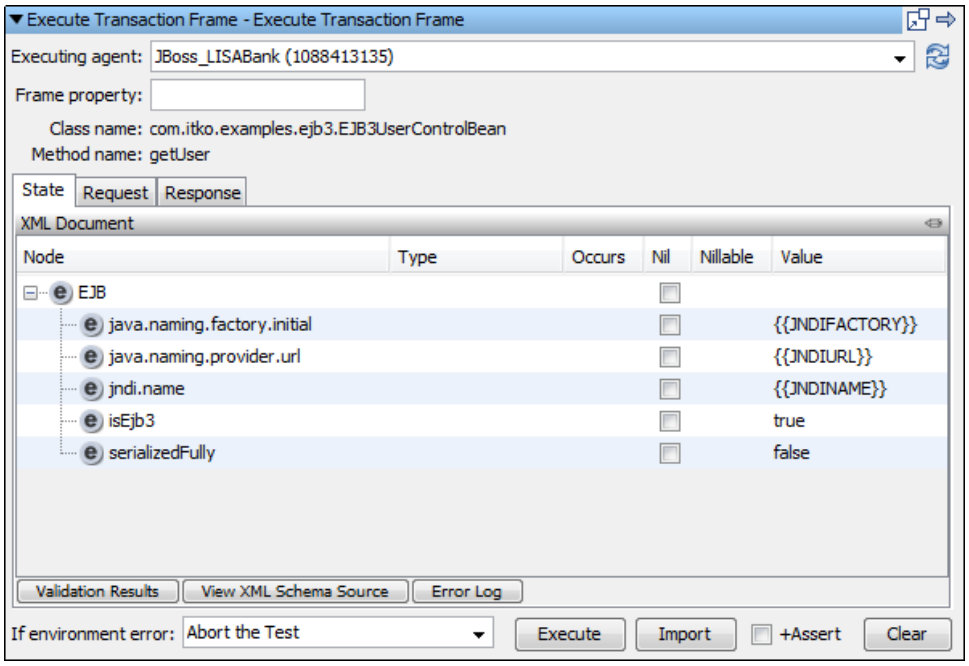


インポートが成功すると、3 つのタブが追加されます。

- 状態
- 要求
- 応答

これで、フレームを設定および呼び出すことができます。

以下の図は、フレームがインポートされた後のステップを示しています。



上部には、以下の項目があります。

実行エージェント

フレームを実行するエージェント。 ドロップダウン リスト内のエージェントの色は、それがアクティブかどうかを示します。エージェントがアクティブな場合にのみ、このエディタからの実行は動作します。 コンテンツを更新するには、ドロップダウン リストの右の［更新］ボタンをクリックします。 プロパティも指定できます。

フレーム プロパティ

ステップが実行された場合に、実行するフレームが取得されるプロパティ。 このフィールドは、CAI コンソールが作成する統合ベースライン テストに対して自動的に入力されます。

クラス名

メソッドのクラスの名前。

メソッド名

エージェントがインターセプトしたメソッドの名前。

［状態］タブには、基盤となるプロトコルが必要とするメタデータが含まれます。 たとえば、EJB フレームの状態には JNDI ルックアップ情報が含まれます。 実行の前に状態を変更できます。

［要求］タブには、フレームのメソッドに対する入力が含まれます。 実行の前に要求を変更できます。

［状態］および［要求］タブでは、見出しバーは、表示されているペイロードのタイプを示します。 見出しバーの右のアイコンをクリックして適切なタイプを選択することにより、ペイロード タイプを変更できます。

［応答］タブは、フレームの呼び出しの予期される応答を実際の応答と比較します。その形式は、グラフィカル XML 比較アサーションと同じです。

下部には、以下の項目があります。

環境エラーの場合

環境エラーが発生した場合に実行するステップまたは実行するアクションを選択します。

実行

現在選択されているエージェントに対してフレームを呼び出します。 このボタンを使用すると、エディタでステップが正しく設定されていることをテストできます。

インポート

フレームをインポートします。

アサート

応答を検証するステップにアサーションを追加します。 アサーションは、いつフレームをインポートするかです。

クリア

インポートされたフレームを削除します。

モバイル ステップ

モバイル テスト ケースを記録すると、モバイル テスト ステップが自動的に作成されます。 テスト ステップは、レコーディング中にキャプチャされるモバイル アクションまたはジェスチャを表します。 各テスト ステップは、アプリケーションの特定の画面上で実行したアクションを表します。

記録されたテスト ケースが完了すると、手動でのアクションの順序変更、追加アクションの挿入、またはアサーションの追加など、さまざまな方法でテスト ケースを変更できます。

このセクションには、以下のトピックが含まれます。

モバイル テスト ステップの変更 (P. 436)

モバイル テスト ステップの変更

次の手順に従ってください:

1. 更新するモバイル テスト ケースを開きます。
2. 各ステップの詳細を表示するには、以下を実行します。
   a. 確認するテスト ステップをクリックします。
   b. 右側のエレメント ツリーの［モバイル テスト ステップ］をクリックして、ステップの詳細を展開します。

      ［モバイル テスト ステップ］タブが開き、テスト アプリケーションのスクリーンショットが表示されます。タブの上部の［アクション］セクションには、テスト ステップで実行される個別のアクションが表示されます。
   c. 特定のアクションと関連付けられたスクリーンショットを表示するには、［アクション］セクション内のアクションをクリックします。
3. 手動で新しいアクションを追加するには、［アクション］セクションの［追加］（+）をクリックし、アクションのキー/値のペアを入力します。
4. アクションの順序を変更するには、移動させるアクションをクリックし、上矢印および下矢印を使用してアクションを移動させます。
5. アクションを削除するには、アクションをクリックし、［削除］をクリックします。
6. 記録されたアクションのスクリーンショットを右クリックして、以下のエレメントまたは画面アクションを追加することもできます。

   戻る

   アプリケーションの前の画面に戻るコマンドを挿入します。Android のみ。

   Get element（**エレメントを取得**）

   選択されたエレメントに対して取得を実行します。 選択されたエレメントの名前が［Get element （エレメントを取得）］オプションの隣に表示されます。

   Tap element（**エレメントをタップ**）

   選択されたエレメントに対してタップ アクションを実行します。選択されたエレメントの名前が［Tap element （エレメントをタップ）］オプションの隣に表示されます。

Send keys（**キーを送信**）

キーボードを呼び出し、テキスト エレメントの値を設定するために指定されたキー操作をシミュレートします。

注: 一部のインスタンスでは、アプリケーションは入力された値を表示しません。 たとえば、パスワード フィールドは入力された値を表示しない可能性があります。

Set value（**値を設定**）

選択されたエレメントの値を指定します。 選択されたエレメントの名前が［Set value（値を設定）］オプションの隣に表示されます。

注: スライダの値を設定するために、このアクションを使用することもできます。 たとえば、値を 0.8 に設定すると、スライダが 80% 右に移動します。 スライダの値を設定する場合は、0.1 単位で指定します。 0.25 の値を設定すると認識されず、スライダは移動しません。

Long Tap screen（**画面をロング タップ**）

特定のエレメントではなく、画面上でロング タップします。

Long Tap element（**エレメントをロング タップ**）

選択されたエレメント上でロング タップします。 選択されたエレメントの名前が［Long Tap element（エレメントをロング タップ）］オプションの隣に表示されます。

Wait（**待機**）

テストに一時停止を挿入します。 ステップ レベルで反応時間のような待機を指定できます。 たとえば、**1s-10s** は 1 ～ 10 秒の間のランダムな一時停止を挿入します。 **100z** は、100 ミリ秒の一時停止を挿入します。

Comment（**コメント**）

特定のアクション用のコメントを挿入します。 コメントはアクションを実行しません。説明用です。

Script（**スクリプト**）

任意の beanshell スクリプトを挿入できます。 通常、このスクリプトはデバイスやシミュレータと対話しません。このアクションは、カスタム アサーションを挿入する方法を提供します。

Change Orientation（**向きを変更**）

デバイスの向きを変更します。

Shake（**シェイク**）

デバイスに対してシェイク ジェスチャを実行します。

Go to background（**バックグラウンドに移動**）

ホーム ボタンを押すアクションを実行します。 このアクションは、アプリケーションを 10 秒間バックグラウンドにした後に戻します。 このアクションは、通常、アプリケーションにメモリを解放させ、CPU に負荷がかかる作業を停止させるため、テストに役立ちます。 iOS のみ。

Assertion（**アサーション**）

詳細については、「アサーションのモバイル テスト ステップへの追加」を参照してください。

7. ［保存］をクリックします。

カスタム拡張ステップ

以下のステップが使用できます。

カスタム テスト ステップの実行 (P. 438)
JavaScript ステップ（非推奨） (P. 439)
スクリプト実行（JSR-223）ステップ (P. 441)

カスタム テスト ステップの実行

カスタム テスト ステップは、ユーザのチームがソフトウェア開発キット（SDK）を使用して作成したテスト ステップを実行します。このステップについては、「*SDK の使用*」で説明しています。

JavaScript ステップ（非推奨）

注: JavaScript ステップは非推奨となっています。 代わりに、スクリプト実行（JSR-223） (P. 441)ステップを使用します。

JavaScript ステップは、いくつかの関数またはプロシージャを実行する Java スクリプトを作成および実行する柔軟性を提供します。スクリプトは BeanShell インタープリタを使用して実行されます。 テスト ケースのすべてのプロパティ（ビルトイン オブジェクトを含む）にアクセスできます。

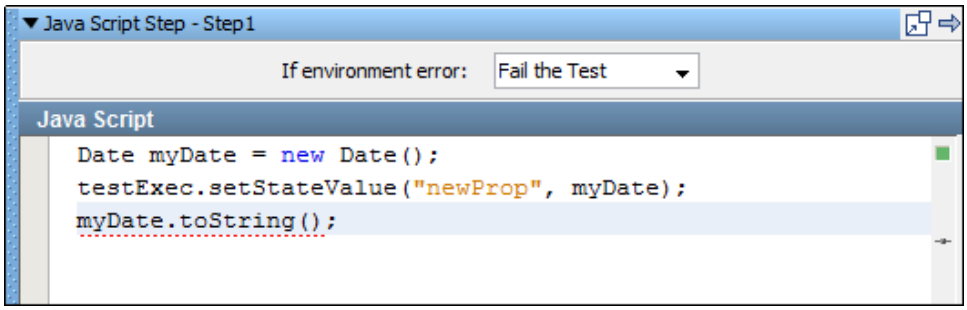
前提条件： BeanShell についての知識。 BeanShell の詳細については、http://www.beanshell.org/ を参照してください。

このステップには、スクリプト エディタがあります。 ［利用可能なオブジェクト］リストの項目をダブルクリックすると、その変数名がエディタに貼り付けられます。

スクリプトで公開される最後の値は、このステップの応答として保存されます。

以下の図は、スクリプト エディタを示しています。このスクリプトには、以下のステートメントが含まれます。

- 新しい Date オブジェクトが作成され、現在の日時に初期化されます。
- このオブジェクトは、公開されている DevTest オブジェクトの 1 つである testExec を使用して、新しいプロパティ **newProp** に保存されます。詳細については、「*SDK の使用*」を参照してください。
- Date オブジェクトの **toString()** 値がステップの応答として設定されます。



［テスト］ボタンを使用すると、スクリプトをテストできます。 スクリプトの実行結果または発生したエラーを説明するエラー メッセージが表示されます。

DevTest プロパティ名の構文は柔軟で、スペースを含めることができます。有効な Java 識別子でないプロパティ名は、このステップでの使用のために変換されます。 無効な文字はアンダースコア（_）で置換されます。

{{exampleprop}} プロパティをスクリプトで使用する場合、DevTest は、スクリプトが実行される前に、実行時の実際のプロパティ値をこのプロパティで置換します。

名前に「.」が含まれるプロパティは、スクリプト環境へのインポート時に「.」が「_」で置換されます。したがって、スクリプト内の **{{foo.bar}}** は、**foo_bar** と同じです。

スクリプト ステップまたはアサーションの内部にログ イベントを作成できます。 **testExec** オブジェクトが役立ちます。 ログ イベントを生成するには、log4j ロガーを使用する代わりに、以下をコード化します。 **testExec.log()** オブジェクトによって実際のイベントが発生し、ITR でそれを参照できます。

```
testExec.log("Got here");
```

実行されたテスト ケースのスクリプト インスタンスは 1 つのみです。 同じテスト ケースまたはサブプロセスに複数の Java スクリプト ステップがある場合、DevTest は、同じインスタンスを使用してテスト ケース全体を実行します。

デフォルトでは、変数はインスタンスに対してグローバルです。 この動作はサブプロセスに拡張されます。

変数のスコープをローカルにする場合は、中かっこで囲んでコードを記述します。 以下に例を示します。

```
{
     String var= "local";
     return var;
}
```

サブプロセスのパラメータ名は、グローバル変数として扱われます。

スクリプト実行（JSR-223）ステップ

注: JavaScript ステップは非推奨となっています。 代わりに、スクリプト実行（JSR-223）ステップを使用します。

スクリプト実行（JSR-223）ステップでは、いくつかの関数またはプロシージャを実行するスクリプトを記述して実行できます。 以下のものから選択できます。

- Applescript （OS X 用）
- Beanshell
- Freemarker
- Groovy
- JavaScript
- 速度

追加のスクリプト言語を使用するには、「追加のスクリプト言語の有効化」を参照してください。

テスト ケースのすべてのプロパティ（ビルトイン オブジェクトを含む）にアクセスできます。

以下のパラメータを入力します。

環境エラーの場合

環境エラーが発生した場合に実行するステップまたは実行するアクションを選択します。

Script Language

使用するスクリプト言語を指定します。

値

- Applescript （OS X 用）
- Beanshell
- Freemarker
- Groovy
- JavaScript
- 速度

デフォルト： Beanshell

Copy properties into scope

ステップで使用するためにダウンロードするプロパティを指定できます。

値

- Test state and system properties： テスト ケースおよびシステムのすべてのプロパティ
- Test state properties： テスト ケースに関する情報を提供するプロパティ
- TestExec and logger only： TestExec およびロガーのプロパティのみ

デフォルト： Test state properties

スクリプト実行の結果または発生したエラーの説明が表示されるウィンドウを開くには、［テスト］をクリックします。

{{式}} を使用するどの場所でも、以下の構文を使用してスクリプト言語を指定できます。

```
{{=%language%}}
```

例は、サンプル プロジェクトの**スクリプト** テスト ケースにあります。

デフォルトのスクリプト言語は、**lisa.scripting.default.language** プロパティを使用して設定します。

第 2 章: テストドキュメントのリファレンス

このセクションには、以下のトピックが含まれています。

Events

以下の表では、標準のイベントについて説明します。 この表は、イベント名順に記載されています。

イベント名	説明	概要	詳細
中止	このイベントは、「完了できない」状態のテストを終了させます。 これは、失敗タイプの終了イベントです。	イベントのアボート	
アサートの評価	埋め込み以外のアサーションが起動しない場合、このイベントが生成されます。 これらのイベントは、結果を実行しなかったアサーションです。	アサーションの名前	
アサーションの起動	埋め込み以外のアサーションが起動された場合、このイベントが生成されます。 起動は、それが ture で、その結果に従うことを意味します。	アサーションの名前	アサーションのログ メッセージ、またはログ メッセージのセットがない場合は DevTest が生成したメッセージ
コールの生成	Web サービスまたは EJB などのオブジェクト コールを実行するステップは、オブジェクトに対する各コールをレポートするためにこのイベントを使用します。	ステップの名前	文字列としてのコール（例： **void setName(java.lang.String name[Basic Checking])**）

コール結果	Web サービスまたは EJB などのオブジェクト コールを実行するステップは、コールから取得した応答をレポートするためにこのイベントを使用します。	ステップの名前	文字列としてのコールの応答。 応答がオブジェクトの場合、オブジェクトの XML ビューが表示されます。
負荷テストの設定		負荷テストの設定	テスト ランは負荷テストとして設定されています。 個別のイベントの選択は無効になっています。
コーディネータの終了	コーディネータが削除されました。	コーディネータ サーバの名前	
コーディネータ サーバの終了	コーディネータ サーバが終了されました。	コーディネータ サーバの名前	
コーディネータ サーバの開始	コーディネータ サーバが作成されました。	コーディネータ サーバの名前	
コーディネータの開始	新しいコーディネータが作成されました。	コーディネータ サーバの名前	
サイクルの正常終了	テストの実行は成功した状態で完了しました。	テスト ケースの名前	
サイクル終了	インスタンスは、テストを停止するように指示されました。		
サイクルの失敗	テストの実行が失敗しました。テスト ケースでは例外は発生していません。ただし、テスト ケースまたはテスト中のシステムのいずれかのロジック エラーにより、テスト ケースが予期されたとおりに完了しませんでした。	テスト ケースの名前	

サイクル履歴	実行されるすべてのモデルは、これらのイベントの 1 つを生成します。 これは、その実行のすべての詳細の最後のトレースです。	サイクルの履歴	
サイクルの初期化	テストが初期化（ロード）されました。	テスト ケースの名前	
サイクル ランタイム エラー	異常な DevTest エラーが発生しました。 たとえば、テストの実行中に、コーディネータがシミュレータへの接続を失った場合などです。	場合によって異なる。通常、エラーが発生したテスト エレメント（ステップ、データ セット、またはフィルタ）の名前	通常、エラーについて説明するメッセージ
サイクルの開始	このテスト インスタンスおよびサイクルはたった今開始されました。	テスト ケースの名前	
データ セットの読み取り	データ セットは、使用される値を示すイベントを生成します。	データ セットの読み取り	
HTTP パフォーマンス	このイベントは、パフォーマンス統計をキャプチャするために、DevTest が実行するすべての HTTP トランザクションに対して生成されます。	HTTP パフォーマンス統計	
情報メッセージ	基本的なログ データ。 たとえば、HTTP/HTML 要求ステップは、Short フィールド（ステップ名）と Long フィールド（サーバに送信される URL）と一緒にこのメッセージを送信します。	通常、このメッセージが生成されたステップの名前	通常、DevTest が生成したメッセージ
インスタンスの終了	シミュレータのインスタンスが終了したときに送信されます。	シミュレータの名前	
インスタンスの開始	シミュレータがインスタンスを作成したときに送信されます。	シミュレータの名前	

ログ メッセージ	基本的なログ データ。 イベントをフィルタする場合にオーバーヘッドを最小化するためにオフにできます。	ログに送信されたメッセージ	
メトリック アラート	メトリックは収集されており、その値をレポートしています。	メトリックの短い名前（例： **DevTest: Avg Response Time**）	収集されたメトリックの値
メトリックの開始	収集されるメトリックは、それらのイベントの値をリアルタイムで生成します。	メトリックの開始	
メトリック値	収集されるメトリックは、それらのイベントの値をリアルタイムで生成します。	メトリック値	
モデル定義エラー	テストの実行中にテスト ケース エラーが検出されました。 たとえば、名前が存在しないプロパティで構成されている場合などです。	場合によって異なる。 通常、エラーが発生したテスト エレメント（ステップ、データ セット、またはフィルタ）の名前	通常、エラーについて説明するメッセージ
Pathfinder	テスト中のシステムで DevTest 統合が有効になりました。 このイベントには、キャプチャされた DevTest 統合 XML データが含まれます。	ステップの名前	キャプチャされた DevTest 統合データの XML 表現
プロパティの削除	プロパティが削除されました。	プロパティ キー	「<削除済み>」という単語
プロパティの設定	プロパティが設定されました。	プロパティ キー	プロパティ値
シミュレータの終了	シミュレータが終了したときに送信されます。	シミュレータの名前	
シミュレータの開始	シミュレータが開始されたときに送信されます。	シミュレータの名前	
ステップ帯域幅の消費	ある程度の量のデータがテスト中のシステムのステップ実行に対して送受信されました。	ステップの名前	実際の読み取り/受信バイト数

ステップ エラー	テスト中のシステムでエラーが発生しました。 たとえば、Web サーバから応答がなかった場合などです。 このイベントは単一のステップ用です。EVENT_TESTFAILED はテストケース全体を示します。	ステップの名前	失敗の原因の特定に役立つメッセージ（使用可能な場合）
ステップ履歴	すべてのステップには、その詳細で **com.itko.lisa.test.NodeExecHistory** タイプを起動する履歴イベントがあります。	ステップ履歴	
ステップ要求	このイベントをサポートするステップは、テスト中のシステムに対して要求が送信されたことをレポートするためにこのイベントを使用します。	ステップの名前	文字列としての要求データ
ステップ要求帯域幅			
ステップ応答	ステップがテスト中のシステムに対して完了しました。	ステップの名前	文字列としての応答データ
ステップ応答帯域幅			
ステップ応答時間	ステップをテスト中のシステムに対して実行するためにかかった時間。	ステップの名前	ステップを実行するための時間（ミリ秒）
ステップの開始	ステップが実行されています。	ステップの名前	
ステップ ターゲット	すべてのステップにはターゲット（Web 要求の URL や EJB の JNDI 名など）があります。		
ステップ警告	警告が記録されました。 たとえば、現在値が見つからなかったためにフィルタがデフォルト値を取得した場合などです。	ステップ警告	

サブプロセスの完了	サブプロセスの実行が完了しました。		
サブプロセスの実行	サブプロセスが開始されました。		
スイートの中止	セットアップ テスト（スイート ドキュメントで定義）が失敗した場合、スイートはスイートで定義されているテストを実行しません。 このイベントは、この状況が発生したことを示します。	スイートの名前	
スイートの終了	スイートの一部として実行されるテストがすべて完了しました。	スイートの名前	
スイート履歴	実行されるすべてのスイートは、これらのイベントの 1 つを生成します。 これは、その実行のすべての詳細の最後のトレースです。	スイートの履歴	
スイートのセットアップ/後処理	スイートにセットアップまたは後処理テストが定義されており、そのテストの実行が開始されました。	スイートの名前	テストの名前、テストのパス、およびその他の情報
スイートの開始	スイートが実行のためにステージングされています。	スイートの名前	
スイート テストの失敗	スイートの一部として実行されるテストが失敗して終了しました。	スイートの名前	
スイート テストの正常終了	スイートの一部として実行されるテストが正常に終了しました。	スイートの名前	
スイート テストのステージング	テストがスイートの一部として実行されるためにステージングされました。	スイートの名前	テストの名前、テストのパス、およびその他の情報
テストの終了	コーディネータがテストを停止するときに送信されます。	テスト ケースの名前	

テスト非アクティブ	スイート内のテストが非アクティブとしてマークされています。	テストが非アクティブのイベント	
テストの開始	コーディネータがテストを開始するときに送信されます。	テスト ケースの名前	
VS ログ メッセージ	VSE の内部ログ	VS ログ メッセージ	
VS トランザクションの一致なし	仮想サービスで、少なくとも 1 つの記録された応答にトランザクション要求が一致しませんでした。	VS トランザクションの一致なし	
VS サービスの終了	仮想サービスが停止されました。	VS サービスの停止	
VS サービスの開始	仮想サービスが開始されました。	VS サービスの開始	
VS トランザクションの完了	仮想サービスがトランザクション要求の処理を完了しました。		
VS トランザクションの一致	仮想サービスで、少なくとも 1 つの記録された応答にトランザクション要求が一致しました。	VS トランザクションの一致	
VSE サーバのリセット	サーバのサービス リセット要求が行われました。	VSE サーバのリセット	
VSE サーバのシャットダウン	仮想サービス環境のシャットダウンが要求されました。	VSE サーバのシャットダウン	
VSE サーバの停止	サーバのサービス停止要求が行われました。	VSE サーバの停止	

メトリック

このセクションでは、使用可能なメトリックのタイプについて説明します。

DevTest 包括テスト メトリック

包括テスト メトリックは、名前が示すように、テスト ケースに関するすべての基本情報を収集し、6 つのサブメトリックを提供します。

以下のサブメトリックが収集されます。 応答時間メトリックはミリ秒単位でレポートされます。

- **インスタンス**（仮想ユーザ数）
- **平均応答時間**
- **最大応答時間**
- **最小応答時間**
- **最終応答時間**
- **ステップ数/秒**

注: 仮想ユーザ数（インスタンス）、平均応答時間、およびステップ数/秒サブメトリックは、ステージング ドキュメントのメトリック リストにデフォルトで追加されます。

DevTest テスト イベント メトリック

テスト イベント メトリックは、DevTest イベントに関するメトリックを提供します。 これらのメトリックには、カウンタとゲージの両方が含まれます。

イベント メトリックでは、イベントの短い説明フィールドと一致させる正規表現を含めることにより、特定のタイプのイベントをフィルタできます。

テスト イベント メトリック カテゴリには、8 つのサブメトリックがあります。

- **コーディネータ サーバの開始**
- **コーディネータ サーバの終了**
- **コーディネータの開始**
- **コーディネータの終了**
- **テストの開始**
- **テストの終了**
- **インスタンスの開始**
- **インスタンスの終了**

テスト イベント メトリックを追加する方法

メトリックを選択に加えて、以下を指定できます。

- キー表現一致： 短い説明フィールドにこの表現がある場合にのみ、選択したイベントをサンプリングする表現を入力します。 このフィールドを空白のままにするか、または「*」を入力すると、このタイプのすべてのイベントがレポートされます。
- メトリックがカウンタである: このチェック ボックスをオンにすると、カウント値が経時的に記録されます。このチェック ボックスをオフにすると、絶対値が記録されます（メトリックはゲージとして機能します）。

テスト イベント メトリックは、ワークフロー エンジンで特定のイベントの発生を追跡することを目的としています。［カウンタ］チェック ボックスをオンにすると、メトリック コレクタは選択したイベントが発生した回数を追跡します。 カウントはテストの実行と共に増加します。ただし、DevTest ワークステーション のグラフおよびレポート コンソールには、サンプリング期間に収集された値が表示されます。 これは、テスト イベント メトリックのデフォルトのオプションです。

［カウンタ］チェック ボックスをオフにすると、メトリック コレクタはイベントの詳細な説明を数値として解析します。 サンプリング期間中、コレクタはこれらの解析値の累計を保持します。 レポートされるメトリックは、解析された数値の単純平均です。 累計は、サンプリング期間ごとにリセットされます。 イベントの詳細な説明に数値が含まれていない場合、メトリック値は 0 になります。

デフォルトのサンプリング期間は 1 秒です。これは、ステージング ドキュメントで変更できます。

- このメトリックをメトリック リストに追加するには、［OK］をクリックします。

SNMP メトリック

SNMP メトリックは、SNMP （Simple Network Management Protocol）を使用してシステムのパフォーマンスをモニタします。

SNMP サポートの設定

SNMP メトリックを収集するには、SNMP を設定する必要があります。UNIX および Windows での SNMP の設定の詳細については、「*インストール*」の「SNMP のインストールおよび設定」を参照してください。

SNMP メトリックを追加する方法

1. ダイアログ ボックスから［SNMP メトリック］を選択し、［OK］をクリックします。

   ［SNMP メトリックの追加］ダイアログ ボックスが表示されます。

   MIB グループ ツリーに、DevTest ワークステーション の標準の SNMP メトリックがすべて表示されます。

   MIB は、特定のドメインのメトリック セットの事前定義済みデータベースです。

   ホスト

   ［ホスト］フィールドは、コーディネータ サーバをホストするサーバに関するシステム情報を提供します。 ホスト メトリックには、CPU 使用率を示す **hrProcessorLoad** や、このホストが最後に初期化されてからの時間を示す **hrSystemUptime** などが含まれます。

   サーバ O/S

   稼働時間、日付、ユーザ数など、システムに関する情報を提供します。

   BEA WebLogic

BEA WebLogic を実行しているサーバの JDBC、JMS、JVM、ソケット、サーブレット、および Web アプリケーションに関する情報を提供します。 BEA WebLogic メトリックには、JVM ヒープの現在の空きメモリ容量をバイト単位で示す **jvmRuntime-HeapFreeCurrent** や、このコンポーネントの現在のオープン セッションの合計数を示す **webAppComponentRuntimeOpenSessionsCurrentCount** などが含まれます。

RDBMS

一般的なリレーショナル データベース管理システムを実行しているサーバに関する情報を提供します。 RDBMS メトリックには、RDBMS が最後に再起動されてから完了した物理ページの読み取り数を示す **rdbmsSrvInfoPageReads** や、RDBMS が最後に再起動されてから完了した単一のページ書き込み数を示す **rdbmsSrvInfoPageWrites** などが含まれます。

Oracle

Oracle を実行しているサーバに関する情報を提供します。 Oracle メトリックには、ユーザのコミット数を示す **oraDbSysUserCommits** や、データがロール バックされた回数を示す **oraDbSysUserRollbacks** などが含まれます。

Microsoft SQL Server

Microsoft SQL Server を実行しているサーバに関する情報を提供します。Microsoft SQL Server メトリックには、要求されたデータ ページが（ディスクから読み込まれるのではなく）データ キャッシュで見つかった回数のパーセンテージを示す **mssqlSrvInfoCacheHitRatio** や、ユーザ接続のオープン数を示す **mssqlSrvInfoUserConnections** などが含まれます。

2. メトリックを選択するために、これらの選択肢を参照します。
3. 左側のパネルで目的のメトリックをクリックします。

   目的のメトリックに必要なパラメータが、右側のパネルの［MIB オブジェクト］フォームに入力されます。 メトリックの説明がテキスト ボックスに表示されます。 その他の情報は無視するか、またはそのまま使用できます。
4. メトリックを収集するホスト コンピュータ（ホスト）のドメイン名または IP アドレスを入力します。 このメトリックを追加するには、［OK］をクリックします。
5. 目的の SNMP メトリックをすべて追加するまで、この手順を繰り返します。

MIB グループ ツリーにない SNMP メトリックを追加するには、［**MIB オブジェクト**］フォームに手動でデータ（OID）を入力します。 OID は特定のメトリックの一意の識別子で、ツリー構造の命名規則が使用されます。ITKO のドメイン ルート OID は、「.1.3.6.1.4.1.12841.1.1」です。すべての SNMP メトリックは、この OID から始まります。

［MIB からロード］ボタンを使用すると、MIB のファイル システムを参照できます。

［SNMP Walk］ボタンを使用すると、SNMP ツリーを参照できます。

注: すべての SNMP MIB がサポートされています。 ［SNMP メトリックの追加］ダイアログ ボックスに表示される MIB のセットは、サポートされている MIB のサンプルです。［SNMP メトリックの追加］ダイアログ ボックスには、MIB 内のオブジェクト ID（OID）がわかりやすく表示されます。ただし、［SNMP メトリックの追加］ダイアログ ボックスに表示される OID だけでなく、有効な OID はすべて動作します。 IETF および IANA のすべての標準 MIB のセットが提供されています。 これらの MIB は、**LISA_HOME¥snmp¥ietf** および **LISA_HOME¥snmp¥iana** ディレクトリに格納されています。

JMX メトリック

JMX メトリックは、JMX （Java Management Extension） API を使用してメトリックを提供します。

容易に設定できるよう、以下の JMX コネクタが用意されています。

- 任意の JSR 160 RMI 接続
- JBoss 3.2-4.0
- JSE 5 コネクタ
- Oracle AS （OCJ4）
- Tomcat 5.0.28
- WebLogic 6.1-8.1
- WebLogic 9.x
- WebSphere 5.x[WebSphere 5.x]
- ITKO JMX エージェント

これらのアプリケーションはそれぞれ、少し異なる接続パラメータを必要とします。 サーバに必要な値については、サーバ管理者に問い合わせてください。 プロバイダによって、提供されるメトリックは少し異なります。 その他の JMX 機能を使用する場合は、RMI ステップとして呼び出すことができます。

数値属性のみがサポートされています。

以下の例では、JBoss を使用します。

次の手順に従ってください:

1. ダイアログ ボックスから［JMX 属性リーダ］を選択します。
2. ［OK］をクリックします。

   ［JMX エージェントの選択および設定］ダイアログ ボックスが表示されます。
3. JMX コネクタのリストから **JBoss** を選択します。
4. お使いのインストールの［サーバ ネーミング URL］および［エージェント RMI 名］を入力します。
5. 残りのフィールドに入力する値は、JBoss サーバが JMX データにアクセスするために認証を必要とするかどうかによって異なります。

- JBoss サーバが JMX へのアクセスに認証を必要としない場合は、デフォルトのコンテキスト ファクトリ「org.jnp.interfaces.NamingContextFactory」を使用します。
- JBoss サーバが認証を必要とする場合

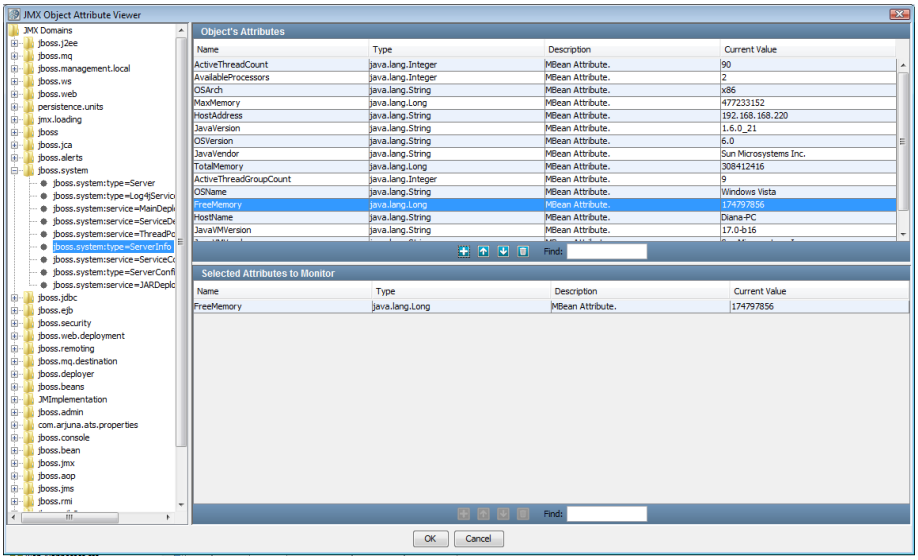
  a. コンテキスト ファクトリを「org.jboss.security.jndi.JndiLoginInitialContextFactory」に変更します。

  b. 認証に必要なユーザ名およびパスワード（プリンシパルおよび認証情報）を入力します。

  c. これらの認証情報を認証する JAAS ログイン モジュールを入力します。

6. ［OK］をクリックします。

［JMX オブジェクト属性ビューア］が開きます。



左側が JBoss の JMX ドメインの階層です。JMX メトリックは、オブジェクト - 属性モデルを使用します。 このモデルでは、ドメイン オブジェクトは特定のアプリケーション サーバ（「system」など）によってパブリッシュされます。また、属性はオブジェクト内の名前/値ペアです。

このツリーでオブジェクトを選択すると、そのオブジェクトの基本属性もツリーに表示されます。 基本属性を選択すると、属性名の残り部分が右上の［オブジェクト属性］パネルにリスト表示されます。

上記の例では、ドメイン オブジェクトに jboss.system、基本属性に ServerInfo、属性名の残り部分に FreeMemory を選択しています。ServerInfo の属性は、［オブジェクト属性］パネルに表示されます。

7. これらの属性（メトリック）のいずれかを選択するには、メトリックを選択して［追加］ をクリックします。

   このメトリックは、ウィンドウの下部の［モニタ対象として選択された属性］パネルの選択されたメトリックのリストに追加されます。

   このパネルから属性を削除するには、［削除］ をクリックします。

8. 選択したすべてのメトリックがリスト（下部パネル）に表示されるまで、このプロセスを繰り返します。

9. ［OK］をクリックして、メインのメトリック パネルに戻ります。

   JMX メトリックがメトリック リストに表示されていることに注目してください。

   アプリケーション サーバによっては、そのアプリケーション サーバとの JMX 通信を有効にするために、ベンダー固有の JAR が必要になる場合があります。

10. 同様に、JMX コネクタのリストから **JSE 5 コネクタ** を選択することもできます。

11. ［OK］をクリックして、JMX エージェントに接続します。

Tomcat 用の JMX メトリックの有効化

次の手順に従ってください:

1. **catalina.bat** ファイルを変更して、**CATALINA_OPTS** を追加します。

   DevTest Solutions にパッケージされている組み込みの Jakarta-Tomcat を使用することもできます。

2. JConsole を使用して Tomcat に接続します。
3. DevTest を起動してステージング ドキュメントを開き、［メトリック］タブに移動します。
4. ［追加］ をクリックします。
5. ［JMX 属性リーダ］を選択して、［OK］をクリックします。
6. ［Tomcat 5.0.28］を選択します。

   ユーザ名とパスワード以外のパラメータは事前に入力されています。

7. ［OK］をクリックして、JMX コンソールに接続します。
8. Catalina に固有のいくつかのオブジェクトを選択してリストに追加します。
9. ［OK］をクリックします。

   それらの属性がメトリック リストに表示されます。

TIBCO Hawk メトリック

TIBCO Hawk は、分散アプリケーションおよびオペレーティング システムをモニタおよび管理するためのツールです。 その他のモニタリング ソリューションとは異なり、TIBCO Hawk ソフトウェアは通信に TIBCO メッセージング ソフトウェアを使用し、その利点の多くを継承しています。これらの利点には、柔軟なアーキテクチャ、エンタープライズ全体にわたるスケーラビリティ、およびロケーションの透過性を備えた設定が容易な製品コンポーネントが含まれます。

DevTest には、テストのコンテキストで分散アプリケーションおよびオペレーティング システムのメトリックをモニタするための TIBCO Hawk との統合が標準で用意されています。TIBCO Hawk では、TIBCO BusinessWorks プロセスのアーカイブ用のコンテナ内メトリックが提供されています。TIBCO Hawk を使用することにより、DevTest は、展開されている TIBCO BusinessWorks プロセスのすべてのアクティビティのメトリックをモニタできます。 この統合は、ボトルネックの発生箇所を把握するために、プロセス内部のピアリングを促進します。

前提条件： このアプリケーションと一緒に DevTest を使用するには、1 つ以上のファイルを DevTest で使用可能にする必要があります。詳細については、「*管理*」の「サードパーティ ファイル要件」を参照してください。

次の手順に従ってください:

1. ステージング ドキュメントを作成します。
2. ステージング ドキュメントの［メトリック］タブで、［追加］ をクリックします。
3. ドロップダウン リストから［TIBCO Hawk］を選択します。

   以下のパラメータを使用できます。

   - **Transport：** ［**Rendezvous**］または［**EMS**］を選択します。
   - **Hawk Domain：** TIBCO Hawk ドメインを入力します。
   - **Rendezvous Service**
   - **Rendezvous Network**
   - **Rendezvous Daemon**
   - **EMS Url**
   - **EMS User**
   - **EMS Password**

4. ［OK］をクリックします。

   Hawk オブジェクト属性ビューアが開きます。

5. ［Process Archive］を選択して展開します。

6. **GetProcessDefinitions** メソッドは、［Process Archive］で定義されているプロセス定義を取得します。

7. **GetActivities** メソッドのパラメータとして［Process Definition Name］を使用します。

   TIBCO Hawk には、プロセス アクティビティのメトリックが数多く用意されています。

8. ［Process Definition Name］と［Activity Name］を使用してフィルタを設定します。

9. メソッド戻り**値**を選択し、［OK］をクリックします。

   アクティビティ メトリックが、このステージング ドキュメントを使用してステージングされたテスト ケースに対してモニタされるようになります。

TIBCO Hawk は TIBCO Hawk API を介してテスト ケースで呼び出すこともできます。

注: TIBCO Hawk が初期化されて、メトリックをレポートできる状態になるには数分かかります。 この状態になったときに、TIBCO は DevTest に通知します。 TIBCO Hawk メトリックを収集するステージング ドキュメントを使用する場合のタイムアウトを回避するには、**lisa.net.timeout.ms** をある程度高い値（少なくとも 2 ～ 3 分）に設定することが重要です。

Windows Perfmon メトリック

Perfmon メトリックは、Microsoft Windows Perfmon を使用して Windows オペレーティング システムのシステム パフォーマンスをモニタするメトリックを提供します。これらのメトリックは SNMP メトリックに似ています。

Perfmon の設定

Perfmon メトリックを収集するには、Windows コンピュータで Perfmon を設定する必要があります。

Perfmon の設定の詳細については、「*インストール*」の「パフォーマンス モニタ（Perfmon）のインストール」を参照してください。

注: リモート コンピュータの Perfmon メトリックを収集している場合は、リモート レジストリ サービスが実行されていることを確認してください。このサービスは、デフォルトでは Windows 7 または Windows 8 上で実行されません。Windows サービス マネージャからリモート レジストリ サービスを手動で開始する必要があります。

次の手順に従ってください:

1. ［メトリック］ダイアログ ボックスから［Windows Perfmon メトリック］を選択し、［OK］をクリックします。

   ［Windows Perfmon マシン名］ダイアログ ボックスが表示されます。

2. 以下のフィールドに入力します。

   マシン名

   Windows インストールのマシン名を入力します。マシン名は、［マイ コンピュータ］を右クリックして［プロパティ］を選択し、［コンピュータ名］タブに移動することによって確認できます。

   ユーザ名

   リモート コンピュータのユーザ名を入力します。

   パスワード

   リモート コンピュータのパスワードを入力します。

   ドメイン

   ドメイン名を入力します。 ドメインはオプションです。

3. ［OK］をクリックして続行します。

   有効なログイン認証情報を入力すると、使用可能なすべてのメトリック タイプを示すウィンドウが表示されます。 Windows Perfmon メトリック ウィンドウが表示されます。

   Perfmon メトリック ウィンドウには 3 つのパネルがあります。

   - 左側のパネルには、選択されたパフォーマンス カテゴリが表示されます。
   - 右上のパネルには、［カウンタの説明］セクションで選択されているカテゴリの説明が表示されます。
   - 右下のパネルには、左側のパネルで選択されているメトリックが表示されます。

4. ［パフォーマンス カテゴリ］リストからメトリックを選択します。

   このリストには、モニタ可能なカテゴリがすべて表示されます。

   - .NET CLR Remoting/ LocksandThreads
   - .NET CLR Data/Networking
   - Job Objects/Job Object Details
   - Performance/RSVP Service
   - Memory/Print Queue
   - ICMP/Process
   - Outlook/Logical disk
   - IP/Server/Cache

   カテゴリを選択すると、左側のパネルに追加されます。

5. 左側のパネルで目的のメトリックをダブルクリックして、右側のパネルの[モニタ対象として選択されたカウンタ]テーブルに追加します。
6. 目的の Perfmon メトリックをすべて追加するまで繰り返します。
7. ［OK］をクリックします。

SSH 経由の UNIX メトリック

このメトリックは、コマンドライン メトリックを収集するために使用されます。 このメトリックは、認証情報、ホスト名、および収集対象として選択されたメトリックなどの入力を収集します。

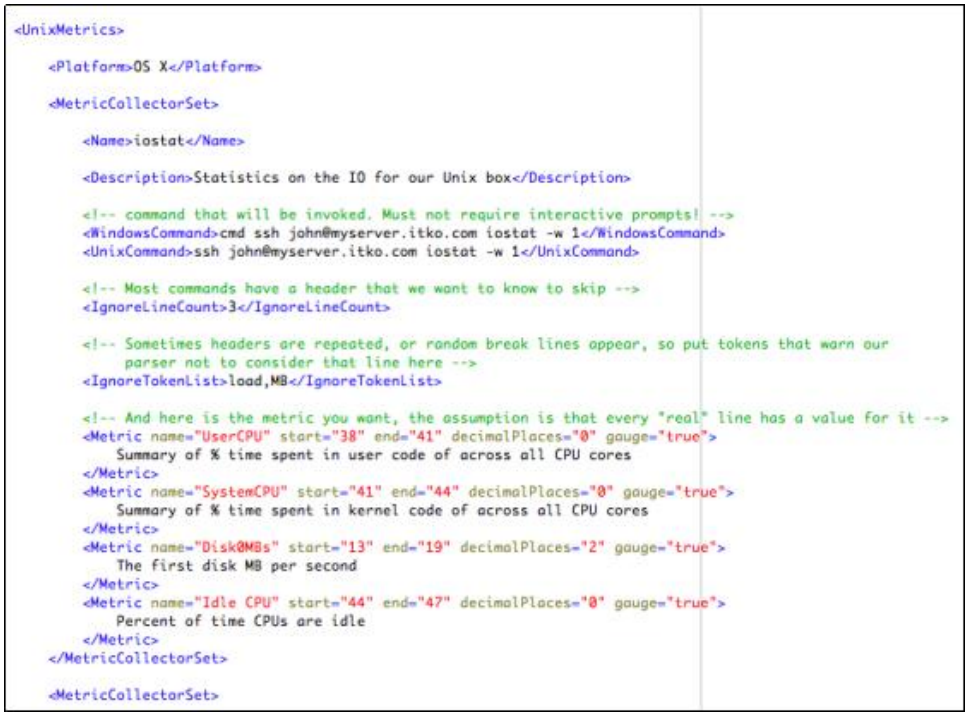
このメトリック データは、デフォルトでは **LISA_HOME/umetrics** ディレクトリにある XML ファイルに格納されます。 ただし、**local.properties** の **stats.unix.xml.folder** キーを更新することによって、新しい場所を定義できます。

このディレクトリの各ファイルには、オペレーティング システムに基づいて一意のファイル名が付けられています。

- Linux.xml
- osx.xml
- Unix.xml
- windows.xml

この XML ファイルは、特定のオペレーティング システムで収集されるコマンドおよびメトリックに関する情報を提供するために使用されます。 たとえば、Linux オペレーティング システムのコマンドおよびメトリックを収集するには、linux.xml ファイルを **LISA_HOME/umetrics** フォルダに配置する必要があります。

以下の図は、CPU および disk0 メトリックを収集する **iostat** の OSX コマンド パーサの例です。



```
<UnixMetrics>

    <Platform>OS X</Platform>

    <MetricCollectorSet>

        <Name>iostat</Name>

        <Description>Statistics on the IO for our Unix box</Description>

        <!-- command that will be invoked. Must not require interactive prompts! -->
        <WindowsCommand>cmd ssh john@myserver.itko.com iostat -w 1</WindowsCommand>
        <UnixCommand>ssh john@myserver.itko.com iostat -w 1</UnixCommand>

        <!-- Most commands have a header that we want to know to skip -->
        <IgnoreLineCount>3</IgnoreLineCount>

        <!-- Sometimes headers are repeated, or random break lines appear, so put tokens that warn our
             parser not to consider that line here -->
        <IgnoreTokenList>load,MB</IgnoreTokenList>

        <!-- And here is the metric you want, the assumption is that every "real" line has a value for it -->
        <Metric name="UserCPU" start="38" end="41" decimalPlaces="0" gauge="true">
            Summary of % time spent in user code of across all CPU cores
        </Metric>
        <Metric name="SystemCPU" start="41" end="44" decimalPlaces="0" gauge="true">
            Summary of % time spent in kernel code of across all CPU cores
        </Metric>
        <Metric name="Disk0MBs" start="13" end="19" decimalPlaces="2" gauge="true">
            The first disk MB per second
        </Metric>
        <Metric name="Idle CPU" start="44" end="47" decimalPlaces="0" gauge="true">
            Percent of time CPUs are idle
        </Metric>
    </MetricCollectorSet>

    <MetricCollectorSet>
```

［リモート マシン詳細の指定］ウィンドウでは、オペレーティング システム（Linux、OS X、または Solaris）を入力します。

リモート コンピュータの詳細を入力します。

- ホスト
- ユーザ

認証タイプを選択し、以下の情報を入力します。

- パスワード
- 秘密鍵
- 暗号化された秘密キー

秘密鍵または暗号化された秘密鍵を使用する場合、秘密鍵ファイルは PEM 形式でプロジェクト下の Data フォルダにある必要があります。［参照］を使用して、ファイルを選択します。

選択するメトリックのセットは、オペレーティング システムによって少し異なります。 収集するメトリックを選択するには、左側の列のチェック ボックスを使用します。

用語集

アサーション

*アサーション*は、1 つのステップとそのすべてのフィルタが実行された後に実行されるエレメントです。 アサーションにより、ステップの実行結果が予測と一致することが検証されます。 アサーションは、通常、テスト ケースまたは仮想サービス モデルのフローを変更するために使用されます。 グローバル アサーションは、テスト ケースまたは仮想サービス モデルの各ステップに適用されます。詳細については、「*CA Application Test の使用*」の「アサーション」を参照してください。

アセット

*アセット*は、1 つの論理的な単位にグループ化される設定プロパティのセットです。 詳細については、「*CA Application Test の使用*」の「アセット」を参照してください。

一致許容差

*一致許容差*は、CA Service Virtualization が受信要求をサービス イメージ内の要求と比較する方法を制御する設定です。 オプションは、EXACT、SIGNATURE、および OPERATION です。 詳細については、「*CA Service Virtualization の使用*」の「一致許容差」を参照してください。

イベント

*イベント*は、発生したアクションに関するメッセージです。 テスト ケースまたは仮想サービス モデル レベルでイベントを設定できます。 詳細については、「*CA Application Test の使用*」の「イベントについて」を参照してください。

会話ツリー

*会話ツリー*は、仮想サービス イメージにおいてステートフル トランザクションの会話パスを表すリンクされたノードのセットです。 各ノードは、withdrawMoney などの操作名でラベル付けされます。 getNewToken、getAccount、withdrawMoney、deleteToken は、金融機関システムの会話パスの一例です。 詳細については、「*CA Service Virtualization の使用*」を参照してください。

仮想サービス モデル（VSM）

*仮想サービス モデル*は、実際のサービス プロバイダなしでサービス要求を受信および応答します。 詳細については、「*CA Service Virtualization の使用*」の「仮想サービス モデル（VSM）」を参照してください。

監査ドキュメント

*監査ドキュメント*では、1 つのテスト、またはスイート内の 1 つのテスト セットに対する成功条件を設定できます。 詳細については、「*CA Application Test の使用*」の「監査ドキュメントの作成」を参照してください。

クイック テスト

*クイック テスト*機能を使用すると、最小のセットアップでテスト ケースを実行できます。 詳細については、「*CA Application Test の使用*」の「クイック テストのステージング」を参照してください。

グループ

グループ、または仮想サービス グループは、VSE コンソールでまとめてモニタできるように、同じグループ タグでタグ付けされている仮想サービスのコレクションです。

継続的検証サービス（CVS）ダッシュボード

*継続的検証サービス（CVS）ダッシュボード*では、長期間にわたって定期的に実行するテスト ケースおよびテスト スイートをスケジュールできます。 詳細については、「*CA Application Test の使用*」の「継続的検証サービス（CVS）」を参照してください。

コーディネータ

*コーディネータ*はテスト ランの情報をドキュメントとして受け取り、1 つ以上のシミュレータ サーバで実行されるテストをコーディネートします。詳細については、「*CA Application Test の使用*」の「コーディネータ サーバ」を参照してください。

コンパニオン

*コンパニオン*は、すべてのテスト ケースの実行の前後に実行されるエレメントです。 コンパニオンは、単一のテスト ステップではなく、テスト ケース全体に適用されるフィルタとして理解できます。 コンパニオンはテスト ケース内で（テスト ケースに対して）グローバルな動作を設定するために使用されます。 詳細については、「*CA Application Test の使用*」の「コンパニオン」を参照してください。

サービス イメージ（SI）

*サービス イメージ*は、CA Service Virtualization で記録されたトランザクションの正規化バージョンです。 各トランザクションは、ステートフル（会話型）またはステートレスです。 サービス イメージを作成する方法の 1 つは、仮想サービス イメージ レコーダを使用することです。 サービス イメージは、プロジェクトに格納されます。 サービス イメージは、*仮想サービス イメージ*（VSI）とも呼ばれます。詳細については、「*CA Service Virtualization の使用*」の「サービス イメージ」を参照してください。

サブプロセス

*サブプロセス*は、別のテスト ケースによってコールされるテスト ケースです。 詳細については、「*CA Application Test の使用*」の「サブプロセスの作成」を参照してください。

シミュレータ

*シミュレータ*は、コーディネータ サーバの管理下でテストを実行します。詳細については、「*CA Application Test の使用*」の「シミュレータ サーバ」を参照してください。

ステージング ドキュメント

*ステージング ドキュメント*には、テスト ケースを実行する方法に関する情報が含まれます。詳細については、「*CA Application Test* の使用」の「ステージング ドキュメントの作成」を参照してください。

設定

*設定*は、プロパティの名前付きのコレクションであり、通常はテスト中のシステムの環境に固有の値を指定します。 ハードコードされた環境データをなくすことにより、設定を変更するだけで、異なる環境内のテスト ケースまたは仮想サービス モデルを実行できます。 プロジェクトのデフォルト設定の名前は project.config です。 プロジェクトは多数の設定を持つことができますが、一度にアクティブになるのは 1 つの設定のみです。詳細については、「*CA Application Test の使用*」の「設定」を参照してください。

対話型テスト ラン（ITR）

*対話型テスト ラン（ITR）*ユーティリティを使用すると、テスト ケースまたは仮想サービス モデルをステップごとに実行できます。 テスト ケースまたは仮想サービス モデルを実行時に変更し、結果を確認できます。 詳細については、「*CA Application Test の使用*」の「対話型テスト ラン（ITR）ユーティリティの使用」を参照してください。

ディセンシタイズ

*ディセンシタイズ*は、機密データをユーザ定義の代替データに変換するために使用されます。 クレジット カード番号や社会保障番号は機密データの例です。 詳細については、「*CA Service Virtualization の使用*」の「データのディセンシタイズ」を参照してください。

データ セット

*データ セット*は、実行時にテスト ケースまたは仮想サービス モデルにプロパティを設定するために使用できる値のコレクションです。 データ セットによって、テスト ケースまたは仮想サービス モデルに外部のテスト データを使用することができます。データ セットは、DevTest の内部または外部（たとえば、ファイルやデータベース テーブル）に作成できます。 詳細については、「*CA Application Test の使用*」の「データ セット」を参照してください。

データ プロトコル

*データ プロトコル*は、データ ハンドラとも呼ばれます。 CA Service Virtualization では、データ プロトコルは、要求の解析処理を行います。一部のトランスポート プロトコルは、要求を作成するジョブの委任先のデータ プロトコルを許可（または要求）します。 結果として、プロトコルは要求ペイロードを認識する必要が生じます。 詳細については、「*CA Service Virtualization の使用*」の「データ プロトコルの使用」を参照してください。

テスト ケース

*テスト ケース*は、テスト中のシステムのビジネス コンポーネントをテストする方法の仕様です。 各テスト ケースには、1 つ以上のテスト ステップが含まれます。 詳細については、「*CA Application Test の使用*」の「テスト ケースの作成」を参照してください。

テスト スイート

*テスト スイート*は、順番に実行されるようにスケジュールされたテスト ケース、その他のテスト スイート、またはその両方のグループです。 スイート ドキュメントは、スイートのコンテンツ、生成するレポート、および収集するメトリックを指定します。詳細については、「*CA Application Test の使用*」の「テスト スイートの作成」を参照してください。

テスト ステップ

*テスト ステップ*は、実行される単一のテスト アクションを表すテスト ケース ワークフローのエレメントです。 テスト ステップの例としては、Web サービス、Java Bean、JDBC、JMS メッセージングなどがあります。 テスト ステップには、フィルタ、アサーション、データ セットなどの DevTest エレメントを含めることができます。 詳細については、「*CA Application Test の使用*」の「テスト ステップの作成」を参照してください。

トランザクション フレーム

*トランザクション フレーム*は、DevTest Java エージェントまたは CAI Agent Light がインターセプトしたメソッド コールに関するデータをカプセル化します。詳細については、「*CA Continuous Application Insight の使用*」の「ビジネス トランザクションおよびトランザクション フレーム」を参照してください。

ナビゲーション許容差

*ナビゲーション許容差*は、CA Service Virtualization が会話ツリーを検索して次のトランザクションを見つける方法を制御する設定です。 オプションは、CLOSE、WIDE、および LOOSE です。 詳細については、「*CA Service Virtualization の使用*」の「ナビゲーション許容差」を参照してください。

ネットワーク グラフ

ネットワーク グラフは、DevTest クラウド マネージャおよび関連するラボをグラフで表示するサーバ コンソールの領域です。詳細については、「*CA Application Test の使用*」の「ラボの開始」を参照してください。

ノード

DevTest の内部では、テスト ステップは*ノード*とも呼ばれます。これが、一部のイベントがイベント ID 内にノードを持つ理由です。

パス

*パス*には、Java エージェント がキャプチャしたトランザクションに関する情報が含まれます。詳細については、「*CA Continuous Application Insight の使用*」を参照してください。

パス グラフ

*パス グラフ*には、パスおよびそのフレームのグラフ表示が含まれています。 詳細については、「*CA Continuous Application Insight の使用*」の「パス グラフ」を参照してください。

反応時間

*反応時間*は、テスト ステップを実行する前にテスト ケースが待機する時間です。詳細については、「*CA Application Test の使用*」の「テスト ステップの追加 - 例」および「ステージング ドキュメント エディタ - ［ベース］タブ」を参照してください。

フィルタ

*フィルタ*は、ステップの前後に実行されるエレメントです。フィルタは、結果のデータを処理、またはプロパティに値を格納する機会を提供します。グローバル フィルタは、テスト ケースまたは仮想サービス モデルの各ステップに適用されます。 詳細については、「*CA Application Test の使用*」の「フィルタ」を参照してください。

プロジェクト

*プロジェクト*は、関連する DevTest ファイルのコレクションです。 ファイルには、テスト ケース、スイート、仮想サービス モデル、サービス イメージ、設定、監査ドキュメント、ステージング ドキュメント、データ セット、モニタ、および MAR 情報ファイルなどが含まれます。 詳細については、「*CA Application Test の使用*」の「プロジェクト パネル」を参照してください。

プロパティ

*プロパティ*は、ランタイム変数として使用できるキー/値ペアです。 プロパティには、さまざまなタイプのデータを格納できます。 一般的なプロパティには、LISA_HOME、LISA_PROJ_ROOT、LISA_PROJ_NAME などがあります。 設定は、プロパティの名前付きのコレクションです。 詳細については、「*CA Application Test の使用*」の「プロパティ」を参照してください。

マジック ストリング

*マジック ストリング*は、サービス イメージの作成中に生成される文字列です。 マジック ストリングは、仮想サービス モデルによって応答内で意味のある文字列値が提供されることを確認するために使用されます。 {{=request_fname;/chris/}} は、マジック ストリングの一例です。 詳細については、「*CA Service Virtualization の使用*」の「マジック ストリングとマジック デート」を参照してください。

マジック デート

レコーディング中、日付パーサは要求および応答をスキャンします。 日付表示形式の広範な定義に一致する値は、*マジック デート*に変換されます。 マジック デートは、仮想サービス モデルによって応答内で意味のある日付値が提供されることを確認するために使用されます。 {{=doDateDeltaFromCurrent("yyyy-MM-dd","10");/*2012-08-14*/} は、マジック デートの一例です。詳細については、「*CA Service Virtualization の使用*」の「マジック ストリングとマジック デート」を参照してください。

メトリック

*メトリック*により、テストおよびテスト中のシステムのパフォーマンス/機能面に定量的手法および測定単位を適用できます。 詳細については、「*CA Application Test の使用*」の「メトリックの生成」を参照してください。

モデル アーカイブ（MAR）

*モデル アーカイブ（MAR）*は、DevTest Solutions における主要な展開アーティファクトです。 MAR ファイルには、プライマリ アセット、プライマリ アセットを実行するために必要なすべてのセカンダリ ファイル、情報ファイル、および監査ファイルが含まれます。 詳細については、「*CA Application Test の使用*」の「モデル アーカイブ（MAR）の操作」を参照してください。

モデル アーカイブ（MAR）情報

*モデル アーカイブ（MAR）情報*ファイルは、MAR を作成するために必要な情報が含まれるファイルです。詳細については、「*CA Application Test の使用*」の「モデル アーカイブ（MAR）の操作」を参照してください。

ラボ

*ラボ*は、1 つ以上のラボ メンバの論理コンテナです。 詳細については、「*CA Application Test の使用*」の「ラボとラボ メンバ」を参照してください。

レジストリ

*レジストリ*は、すべての DevTest サーバおよび DevTest ワークステーション コンポーネントの登録を一元的に行うための場所です。 詳細については、「*CA Application Test* の使用」の「レジストリ」を参照してください。

仮想サービス環境（VSE）

*仮想サービス環境（VSE）*は、仮想サービス モデルを展開して実行するために使用する DevTest サーバ アプリケーションです。 VSE は CA Service Virtualization とも呼ばれます。 詳細については、「*CA Service Virtualization の使用*」を参照してください。