HP Virtual User Generator

Windows オペレーティング・システム用

ソフトウェア・バージョン：9.50

ユーザーズ・ガイド
第 II 巻 - プロトコル

製造部品番号：T7182-99016
ドキュメント発行日：2009 年 1 月（英語版）
ソフトウェア・リリース日：2009 年 1 月（英語版）

invent

利用条件

保証

HP の製品およびサービスの保証は，かかる製品およびサービスに付属する明示的な保証の声明において定められている保証に限ります。本ドキュメントの内容は，追加の保証を構成するものではありません。HP は，本ドキュメントに技術的な間違いまたは編集上の間違い，あるいは欠落があった場合でも責任を負わないものとします。

本ドキュメントに含まれる情報は，事前の予告なく変更されることがあります。

制限事項

本コンピュータ・ソフトウェアは，機密性があります。これらを所有，使用，または複製するには，HP からの有効なライセンスが必要です。FAR 12.211 および 12.212 に従って，商用コンピュータ ソフトウェア，コンピュータ ソフトウェアのドキュメント，および商用アイテムの技術データは，HP の標準商用ライセンス条件に基づいて米国政府にライセンスされています。

サードパーティ Web サイト

HP は，補足情報の検索に役立つ外部サードパーティ Web サイトへのリンクを提供します。サイトの内容と利用の可否は予告なしに変更される場合があります。HP は，サイトの内容または利用の可否について，いかなる表明も保証も行いません。

著作権

商標

Adobe® および Acrobat® は，Adobe Systems Incorporated の商標です。

Intel®，Pentium® および Intel® Xeon™ は，米国およびその他の国における Intel Corporation またはその子会社の商標または登録商標です。

Java™ は，Sun Microsystems, Inc. の米国商標です。

Microsoft®，Windows®，Windows NT® および Windows XP® は，Microsoft Corporation の米国登録商標です。

Oracle® は，カリフォルニア州レッドウッド市の Oracle Corporation の米国登録商標です。

Unix® は，The Open Group の登録商標です。

SlickEdit® は，SlickEdit Inc. の登録商標です。

文書の更新

本書のタイトル・ページには，次の識別情報が含まれています。

- ソフトウェアのバージョンを示すソフトウェア・バージョン番号
- ドキュメントが更新されるたびに更新されるドキュメント発行日
- 本バージョンのソフトウェアをリリースした日付を示す，ソフトウェア・リリース日付

最新のアップデートまたはドキュメントの最新版を使用していることを確認するには，**http://h20230.www2.hp.com/selfsolve/manuals** を参照します。

このサイトでは，HP Passport に登録してサインインする必要があります。HP Passport ID の登録は，以下の Web サイトにアクセスしてください。

http://h20229.www2.hp.com/passport-registration.html

または，HP Passport のログイン・ページの［**New users - please register**］リンクをクリックしてください。

適切な製品サポート・サービスに登録すると，更新情報や最新情報も入手できます。詳細については HP の営業担当にお問い合わせください。

サポート

HP ソフトウェアのサポート Web サイトは，次の場所にあります。
http://www.hp.com/go/hpsoftwaresupport

HP ソフトウェアのオンライン・サポートは，インタラクティブな技術サポート・ツールにアクセスするための効率的な手段を提供します。サポート・サイトを利用することで，次のようなことができるメリットがあります。

- 関心のある内容の技術情報の検索
- サポート・ケースおよび機能強化要求の提出および追跡
- ソフトウェア・パッチのダウンロード
- サポート契約の管理
- HP サポートの連絡先の表示
- 利用可能なサービスに関する情報の確認
- ほかのソフトウェア顧客との議論の開始
- ソフトウェアのトレーニングに関する調査と登録

ほとんどのサポート・エリアは，HP Passport ユーザとしての登録およびサインインが必要です。また多くは，サポート契約も必要です。アクセス・レベルの詳細情報については，**http://h20230.www2.hp.com/new_access_levels.jsp** を参照してください。

HP Passport ID の登録は，次の場所で行います。
http://h20229.www2.hp.com/passport-registration.html

目次

第 1 章

プロトコルについて

VuGen はさまざまな種類のアプリケーションとプロトコルをサポートしており，ユーザのアクションを正確にエミュレートしたスクリプトを作成します。

注 :『Virtual User Generator ユーザーズ・ガイド』のオンライン版は 1 冊，印刷版は『**第 1 巻 ― VuGen の使用**』および『**第 2 巻 ― プロトコル**』の 2 冊から成ります。

本書の使用方法

本書『**プロトコル**』は，『HP Virtual User Generator ユーザーズ・ガイド』の第 2 巻です。第 1 巻の『VuGen の使用』は，Vugen の使用およびテストの作成について説明します。第 2 巻では，個々のプロトコルに応じた設定とガイドラインを説明します。たとえば，第 2 巻には，プロトコル固有の記録オプションや実行環境の設定について説明し，『**第 1 巻 ― VuGen の使用**』ではすべてのあるいはほとんどのプロトコルに共通の設定について示しています。

記録する Vuser タイプを決めるとき，アプリケーションで複数のプロトコル（たとえば，Web と FTP，または Web と Web サービスなど）を使用していることに気づくかもしれません。VuGen は複数のプロトコルを使用するスクリプトの記録をサポートしています。詳細については，『**第 1 巻 ― VuGen の使用**』を参照してください。

サポートされるプロトコルをアルファベット順に並べたリストを見るには，［**ファイル**］＞［**新規作成**］を選択し，［**カテゴリ**］リスト・ボックスで［**すべてのプロトコル**］を選択します。

LoadRunner で使用する GUI Vuser スクリプトの作成方法については，『**WinRunner ユーザーズ・ガイド**』または QuickTest を参照してください。

Vuser の種類

VuGen は，システムのエミュレートを可能にするさまざまな Vuser テクノロジを備えています。それぞれのテクノロジは特定のアーキテクチャに対応しており，アーキテクチャごとに専用の Vuser スクリプトが作成されます。たとえば，Web Vuser スクリプトは，Web ブラウザおよび FTP Vuser を操作しているユーザが FTP セッションをエミュレートする場合に使用します。さまざまな Vuser テクノロジを単独で使用したり，組み合わせて使用したりすることによって，効果的なテスト，または Business Process Monitor プロファイルを作成できます。

Vuser のタイプは次のカテゴリに分類されます。

- ［**すべてのプロトコル**］：サポートされている全プロトコルのアルファベット順リスト。
- ［**アプリケーションの導入ソリューション**］：Citrix および Microsoft Remote Desktop Protocol（RDP）プロトコル用。
- ［**クライアント / サーバ**］：DB2 CLI，DNS，Informix，Microsoft .NET，MS SQL，ODBC，Oracle（2-Tier），Sybase CTlib，Sybase DBlib，Windows Sockets プロトコル用。
- ［**ユーザ定義**］：C テンプレート，Java テンプレート，Javascript，VB Script，VBNet，および VB テンプレート・タイプ・スクリプト用。
- ［**分散コンポーネント**］：COM/DCOM，および Microsoft .NET プロトコル用。
- ［**e ビジネス**］：AJAX（Click and Script），AMF，Flex，FTP，LDAP，Microsoft .NET，Web（Click and Script），Web（HTTP/HTML），および Web サービス・プロトコル用。
- ［**Enterprise Java Beans**］：EJB テスト用。
- ［**ERP/CRM**］：Oracle NCA，Oracle Web Applications 11i，Peoplesoft Enterprise，Peoplesoft-Tuxedo，SAP-Web，SAPGUI，SAP （Click and Script），および Siebel Web プロトコル用。
- ［**Java**］：Java Record/Replay プロトコル用。
- ［**レガシ**］：ターミナル・エミュレーション（RTE）用。

- ［**メール サービス**］: Internet Messaging（IMAP），MS Exchange（MAPI），Post Office Protocol（POP3），Simple Mail Protocol（SMTP）プロトコル用。
- ［**ミドルウェア**］: Tuxedo プロトコル用。
- ［**ストリーミング**］: MediaPlayer（MMS）と RealPlayer プロトコル用。
- ［**ワイヤレス**］: マルチメディア・メッセージング・サービス（MM）および WAP プロトコル用。

第 2 章

Web サービス プロトコル

VuGen を使用して，Web サービスを対象としたテストを作成します。Web サービス・テスト・スクリプトを作成したら，スクリプト・ビューまたはツリー・ビューで表示したり変更できます。

本章の内容

Web サービス・スクリプトの作成について

SOA システムは Web サービスに基づいて，インターネットをまたいでさまざまなプラットフォームの上で広く実行できる自己完結型のアプリケーションです。サービスは，XML（Extensible Markup Language）と SOAP（Simple Object Access Protocol）を使って作成されます。Web サービスは，新しいアプリケーションの開発，デプロイメントを短期間で実現するビルディング・ブロックとして機能します。

VuGen を使用して，SOA 環境をテストするためのテスト・スクリプトを作成します。テスト生成ウィザードでスクリプトを自動的に生成するか，スクリプトを手作業で作成できます。

スクリプトを手作業で作成するには，空のスクリプトを作成することから始めます。その後で，セッションを記録するか，ネットワーク・トラフィックを分析するか，第 4 章「Web サービス－スクリプト内容の追加」で説明しているように手作業で Web サービスに呼び出しを挿入して，スクリプトに内容を追加します。

手動スクリプトの場合は，VuGen を使って次のスクリプトのいずれかを作成します。

- ［**シングル プロトコル スクリプト**］: Web サービスに SOAP 要求を送信して，SOAP トラフィックをエミュレートするスクリプト。
- ［**マルチ プロトコル スクリプト**］: シングル・スクリプトで複数のプロトコルをエミュレートするスクリプト。たとえば，環境に Web サービスと Web ページにアクセスするクライアントが含まれている場合は，Web サービスと Web（Click および Script）プロトコルを選択します。

Web サービス仮想ユーザ・スクリプトの概要

本項では，Web サービス /SOA 仮想ユーザ・スクリプトを作成する工程の概略を説明します。

テスト・スクリプトを作成するには，次の手順を実行します。

1 新しい Web サービス・スクリプトを作成します。

SOA テスト・ジェネレータを使って新しいスクリプトを作成するか，新しい単一または複数のプロトコル・スクリプト，すなわち Business Process Testing コンポーネントを手動で作成します。

2 スクリプトに内容を追加します。

スクリプトに内容を追加します（SOA テスト・ジェネレータは除きます）。詳細については，第 4 章「Web サービス－スクリプト内容の追加」

3 プロパティ，値，およびチェックポイントを設定します。

ステップ・プロパティをカスタマイズしたり，引数値を挿入したり，チェックポイントを設定してスクリプトを拡張します。詳細については，第 6 章「Web サービス呼び出しのプロパティ」

4 スクリプトをパラメータ化します。

パラメータ化によって，定数値を変数に置き換え，各反復に新しい値を代入できます。値をパラメータ化するには，ステップをダブルクリックしてそのプロパティを開き，値ボックスの横にある **ABC** アイコンをクリックします。より複雑なタイプの要素の場合には，『**第 1 巻 － VuGen の使用**』の第 14 章「ファイル，テーブル，および XML パラメータ・タイプ」で説明しているように XML パラメータ・タイプを使用できます。

5 スクリプトをトランザクションで拡張します。

アプリケーションのパフォーマンスをチェックするために，一連のアクションをトランザクションとして定義します。たとえば，サービスでアドレスを更新するのにかかる時間をチェックする場合は，それらのアクションをトランザクションとしてマークします。詳細については，『**第 1 巻 － VuGen の使用**』の第 6 章「Vuser スクリプトの拡張」を参照してください。

6 実行環境を設定します。

実行環境を設定することによって，実行中のスクリプトの動作を制御します。この設定には，Web サービス固有の設定（クライアントのエミュレーション）と一般設定（実行論理，ペースの設定，ログ，思考遅延時間）が含まれます。

Web サービス固有の設定については，124 ページ「Web サービス JMS の実行環境の設定」および『**第 1 巻 － VuGen の使用**』の「実行環境の設定」を参照してください。

7 スクリプトが機能するか確認します。

VuGen でスクリプトを再生して，スクリプトが正しく実行されることを確認します。

スクリプト再生の詳細については，『**第 1 巻 － VuGen の使用**』の「スタンドアロン・モードでの Vuser スクリプトの実行」を参照してください。

8 スクリプトを保存します。

ファイル・システムまたは Quality Center リポジトリにスクリプトを保存します。Quality Center にスクリプトを保存すると，スクリプトをテスト・セットに関連付け，Quality Center から直接機能テストおよび回帰テストを実行できます。Quality Center およびスクリプトとの統合に関する詳細は，76 ページ「Service Test Management の操作」を参照してください。

スクリプトを用意したら，いつでもテストに使用できます。詳細については，75 ページ「スクリプトの使い方」を参照してください。

Quality Center を使用して，不具合や要件を追跡しながらすべてのテストを管理します。詳細については，www.hp.com を参照するか，担当営業にお問い合わせください。

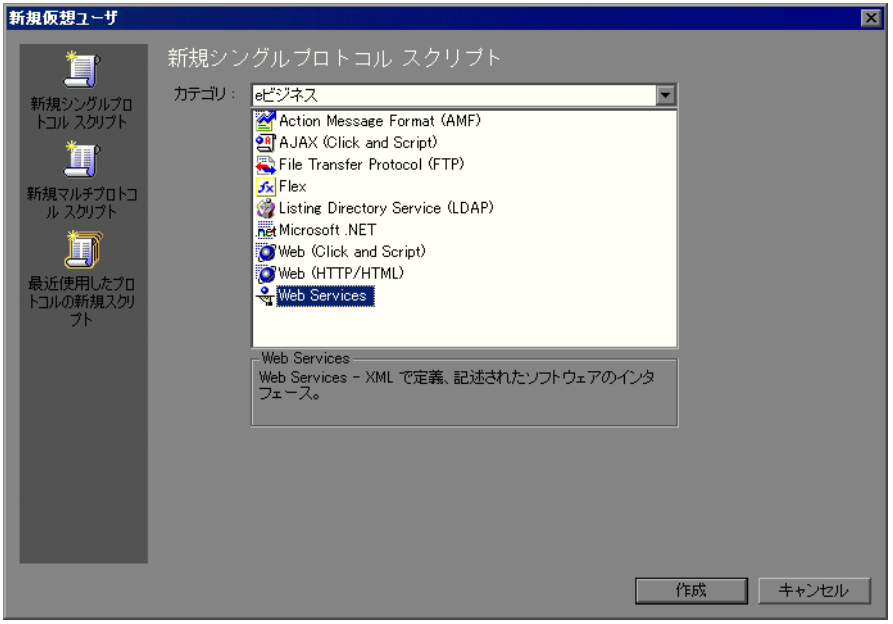
空の Web サービス・スクリプトの作成

手動の Web サービス呼び出しでスクリプトを作成するには，まず空のスクリプトを作成する必要があります。空のスクリプトがベースになり，Web サービス呼び出しやほかの SOA 関連ステップを追加します。

空のスクリプトを作成するには，次の手順を実行します。

1 空のスクリプトを作成します。［新規スクリプト］ボタンをクリックするか，［**ファイル**］＞［**新規作成**］を選択して，［新規仮想ユーザ］ダイアログ・ボックスを開きます。

2 左の表示枠で［**新規シングル プロトコル スクリプト**］をクリックします。［**e- ビジネス**］カテゴリから［**Web サービス**］プロトコルを選択します。［**OK**］をクリックします。



Web サービスや Web など，いくつかの異なるプロトコルを記録する必要がある場合は，左の表示枠で［**新規マルチ プロトコル スクリプト**］をクリックして，目的のプロトコルを指定します。［**OK**］をクリックします。

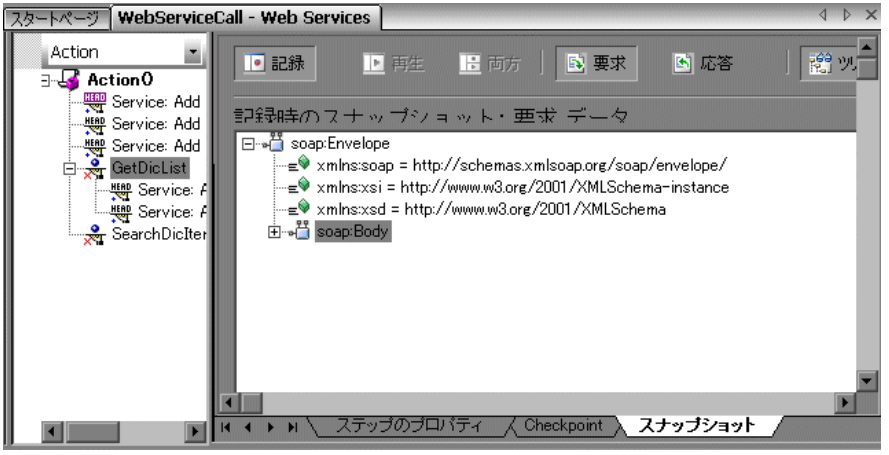
スクリプトの表示と編集

手作業でも自動でも［VuGen］ウィンドウで作成したスクリプトはすべて，表示して編集できます。

スクリプトはツリー・ビューまたはスクリプト・ビューのどちらでも表示できます。ツリー・ビューには，スクリプトの手順がグラフィカル・インタフェースで表示されます。一方，スクリプト・ビューには，サービスをエミュレートする実際の **web_service_call** 関数など，すべての手順が表示されます。スクリプト・ビューは，スクリプト内で柔軟性を必要とする上級ユーザに適しています。

ツリー・ビュー

ツリー・ビューには，スクリプトの各ステップが視覚的に表示されます。



ステップを選択すると，VuGen ではいくつかのタブにそのステップに関する情報が表示されます。

- ［**ステップのプロパティ**］：Web サービス呼び出しのプロパティと引数値。このタブでは，既存のステップのプロパティを変更できます。詳細については，95 ページ「Web サービス呼び出しのプロパティについて」を参照してください。
- ［**CheckPoint**］：ステップに定義されているチェックポイントのリスト。詳細については，124 ページ「チェックポイント」を参照してください。
- ［**スナップショット**］：SOAP 要求と記録および再生に対する応答のスナップショット。詳細については，92 ページ「Web サービスの SOAP スナップショットの表示」を参照してください。

これらのタブの詳細については，第 6 章「Web サービス呼び出しのプロパティ」を参照してください。

ツリー・ビューでスクリプトを表示するには，次の手順を実行してください。

1 ［**ツリー**］ボタンをクリックするか，［**表示**］＞［**ツリー ビュー**］を選択します。

2 左上のボックスで，表示するスクリプトのステップ，すなわち **vuser_init**，**Action**，または **vuser_end** が含まれているセクションを選択します。新しいアクションを指定するには，［**アクション**］＞［**新規アクションを作成**］を選択します。

3 左の表示枠で，表示または変更するステップまたはサブノードを選択します。

4 プロパティを表示したり変更するには，右の表示枠で［**ステップのプロパティ**］タブを選択します。

5 ステップの SOAP ヘッダと本文を表示するには，［**スナップショット**］タブを選択します。特定の再生反復を表示するには，［**表示**］＞［**スナップショット**］＞［**反復の選択**］を選択します。

6 さらに Web サービス・ステップを追加するには，［**サービス呼び出しの追加**］ボタンをクリックします。詳細については，68 ページ「新しい Web サービス呼び出しの追加」を参照してください。

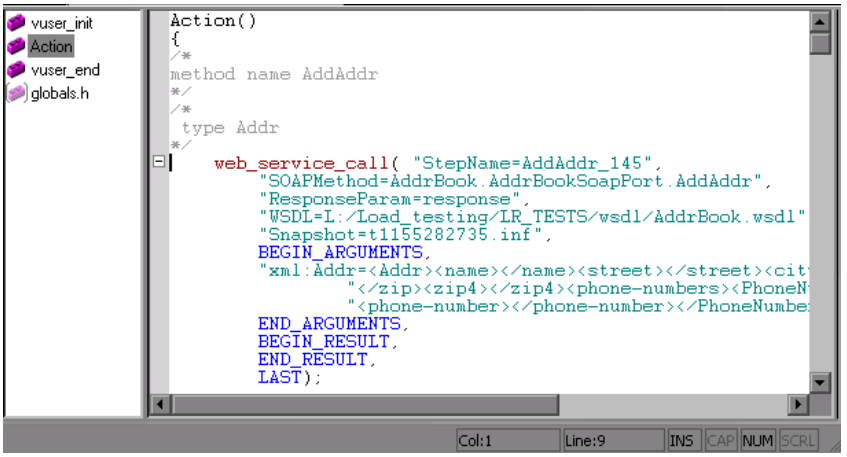
7 JMS キュー機能や SAML セキュリティなどの高度な機能を挿入するには，［**挿入**］＞［**新規ステップ**］を選択し，該当するステップを選択します。詳細については，第 9 章「Web サービスーセキュリティ」を参照してください。

8 引数値をパラメータに置き換えるには，［**ステップのプロパティ**］タブに移動します。スクリプト内で値を置き換えるノードを選択し，［**値**］ボックスの右側にある［**ABC**］アイコンをクリックします。

9 チェックポイントを設定するには，［**CheckPoint**］タブをクリックします。詳細については，124 ページ「チェックポイント」を参照してください。

スクリプト・ビュー

スクリプト・ビューには，スクリプト内に生成された実際の関数が表示されます。**web_service_call** 関数ごとに展開したり折りたたんだりして，目的の関数だけを表示できます。



ツリー・ビューでスクリプトを表示するには，次の手順を実行してください。

1 ［**スクリプト**］ボタンをクリックするか，［**表示**］＞［**スクリプト ビュー**］を選択します。

2 左の表示枠で，表示するスクリプトのステップ，すなわち **vuser_init**，**Action**，または **vuser_end** が含まれているセクションを選択します。新しいセクションを指定するには，［**アクション**］＞［**新規アクションを作成**］を選択します。

3 カーソルの位置でさらに Web サービス・ステップを追加するには，［**サービス呼び出しの追加**］ボタンをクリックします。詳細については，第 6 章「Web サービス呼び出しのプロパティ」を参照してください。

4 JMS キュー機能や SAML セキュリティなどの高度な機能を挿入するには，［**挿入**］＞［**新規ステップ**］を選択し，該当するステップを選択します。詳細については，第 9 章「Web サービスーセキュリティ」を参照してください。

5 引数の値をパラメータに置き換えるには，スクリプト内で置き換える値を選択し，右クリックして表示されるメニューから［**パラメータで置換**］を選択します。

関数の詳細については，『**Online Function Reference**』（英語版）（［**ヘルプ**］＞［**関数リファレンス**］）を参照するか，関数を選択して F1 キーをクリックします。

スクリプトのパラメータ化

VuGen では，すべての引数値のパラメータ化をサポートしています。パラメータ化により，元の値を外部の値に置き換えることができます。これは異なる値でサービスをテストしたり，ステップ間で情報を渡すのに便利です。パラメータ化については，『**第 1 巻 － VuGen の使用**』の「パラメータの作成」を参照してください。

引数が単純で配列型でない場合，引数を単純パラメータに置き換えることができます。たとえば，足し算を行うサービスをテストする場合，各入力引数をパラメータに置き換え，値をファイルまたはテーブルに格納できます。

しかし，引数が多数の値を持つ複雑な構造である場合，XML タイプのパラメータを使用して構造全体を 1 つのパラメータに置き換えることができます。XML タイプのパラメータの値セットをいくつか作成して，反復ごとに異なる値セットを割り当てることもできます。詳細については，『**第 1 巻 － VuGen の使用**』の「パラメータ・タイプについて」。

パラメータを使用すると，ある操作からの出力値を後の操作の入力として渡すことができます。詳細については，127 ページ「Web サービスの出力パラメータの使用方法」を参照してください。

定数値をパラメータに置換するには，次の手順を実行します。

1 ［**ステップのプロパティ**］タブに切り替えて，値をパラメータ化する親または子要素を選択します。

2 ［**入力引数**］ノードで，パラメータ化する引数を選択します。右側の表示枠で，［**値**］ボックスの［**ABC**］アイコンをクリックします。［パラメータの選択または作成］ダイアログ・ボックスが開きます。

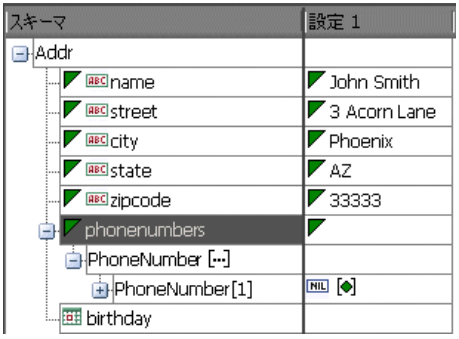
3 パラメータの名前と種類を指定します。

4 ［**プロパティ**］をクリックしてパラメータのタイプ（ファイル，XML など）を設定し，値を割り当てます。

詳細については，『**第 1 巻 ― VuGen の使用**』の「パラメータの作成」を参照してください。

XML の編集

Service Test には，WSDL または XML スキーマに従って XML 構造を表示するエディタが用意されています。グリッド状の表示では，XML を適切な階層で表示し，各要素の値を設定できます。左の列にはスキーマが表示され，その他の列には生成される XML とそのプロパティが表示されます。



XML エディタをスタンドアロンエディタとして使用し，XML コードを書式化できますが，このエディタにはパラメータ化や XML 検証など，多くの Service Test 機能も統合されています。

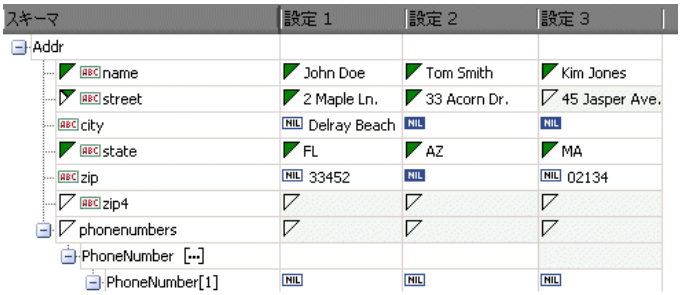
要素の値を設定するには，XML を手作業で編集するか，XML ファイルを値と一緒にインポートします。

このエディタでは，オプション要素がセルの左上隅の小さな三角形で示されます。塗りつぶされた三角形は，インクルードされている要素を示します。オプション要素を除外するには，小さな三角形をクリックしてクリアします。

要素を除外すると，エディタが動的に機能して，XML コードから要素全体が取り除かれます。要素を再インクルードすると，エディタによって XML に戻されます。

複数値セット

値セットは一連の値が含まれている配列です。パラメータには複数値セットを作成し，さまざまな反復やテスト実行に使用できます。



ある値セットには表示されても，別のセットには表示されないオプション要素を使用できます。これによって，各反復に送信する値を変えることができます。特定の配列要素をインクルードできる反復もあれば，それらを除外する反復もあります。

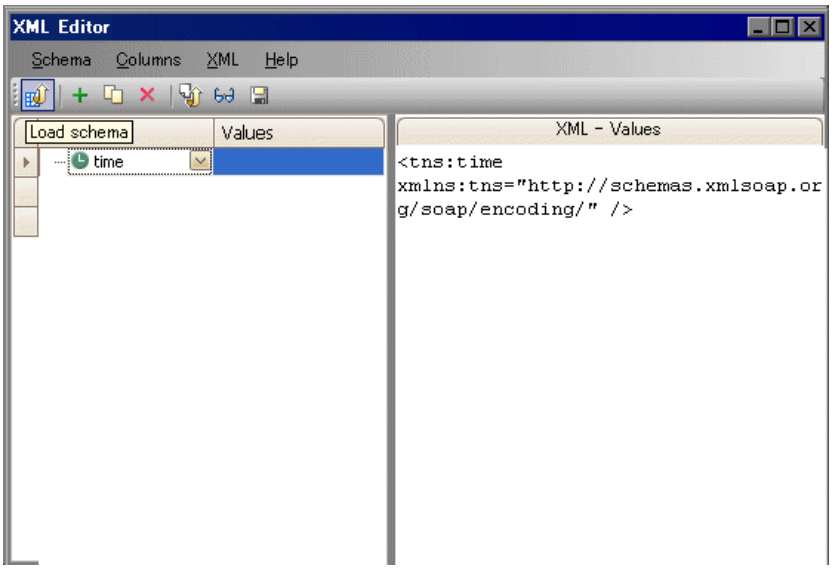
値セットを使用するときは，パラメータごとの配列要素の数を一定にする必要はありません。この例外は Choice，Derived，および< any >タイプであり，要素の数がすべての値セットに固定されています。

値セットのカラムをサイズ変更するには，カラムのタイトルで区切り線をクリックして，マウスを目的の幅にドラッグします。

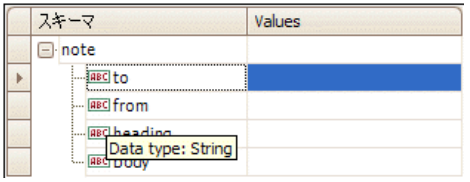
XML エディタを開くには，次の手順を実行します。

1 ［**SOA ツール**］ > ［**XML エディタ**］を選択します。エディタが開きます。

2 スキーマをロードするには，［**Schema**］ > ［**Load**］を選択します。XML ファイルをロードするには，［**XML**］ > ［**Load**］を選択します。

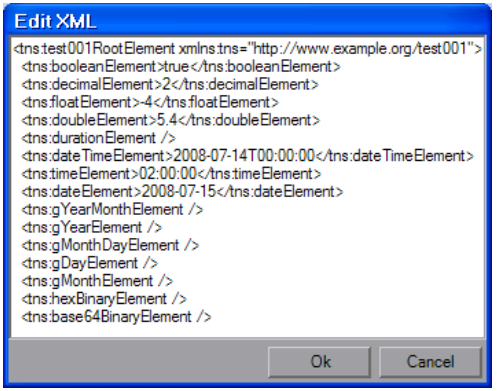


3 ［**Values**］カラムに手作業で入力するか，XML ファイルから値をロードします。［**Load XML from file into the selected column** ］ボタンをクリックします。マウスを要素のアイコン上に移動すると，そのデータ型がわかります。



4 ［**Columns**］ > ［**Add**］を選択するか，［カラムの追加］ボタンをクリックして，別の値セットを追加します。手作業で，または前のステップで説明したように XML をロードして値を追加します。

5 実際の XML コードを編集するには，カラムを選択し，［**Edit XML from the selected column**］ボタンをクリックします。編集が終了したら，［**OK**］をクリックします。



6 カラムを複製するには，カラムを選択して［**Columns**］＞［**Duplicate**］を選択するか，［**Duplicate Column**］ボタンをクリックします。

7 カラムを削除するには，カラムを選択して［**Columns**］＞［**Remove**］を選択するか，［**Duplicate Column**］ボタンをクリックします。

8 カラムを XML ファイルに保存するには，カラムを選択して［**XML**］＞［**Save**］を選択するか，［**Save XML from the selected column**］ボタンをクリックします。

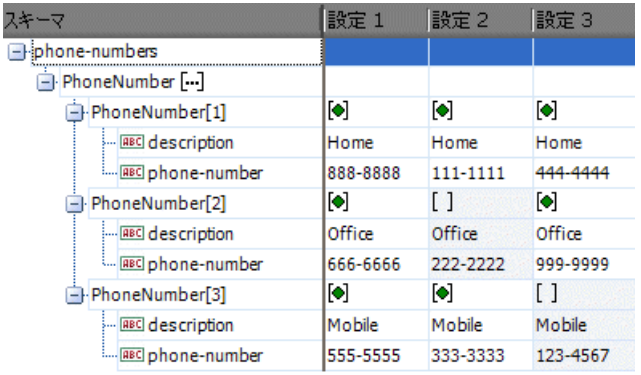
配列

配列要素を操作でき，その構造全体を複製できます。

VuGen では，個々の配列要素を包含または除外することもできます。配列要素を除外すると，VuGen によって SOAP 要求からその要素が除外されます。

この機能によって，一定の値セットでは要素を除外し，ほかのセットでは包含して，XML の内容を動的に制御できます。配列要素を除外すると，その子孫もすべて自動的に除外されます。

次の例では，いくつかの値セットで配列要素が除外されています。



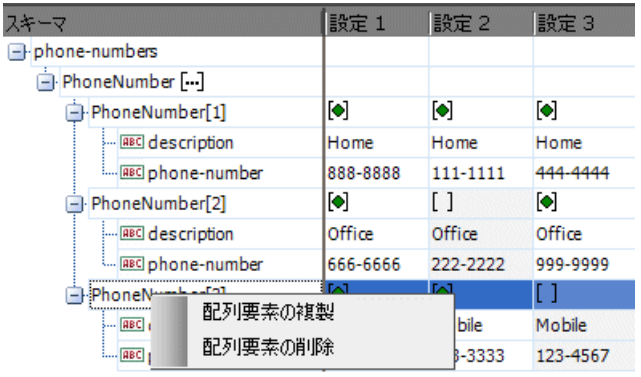
配列要素を包含または除外するには，大括弧内にある緑色の菱形をクリックします。

- 緑色の菱形が表示されている場合は，要素が包含されています。
- 緑色の菱形がない場合は，要素とその子孫がすべて除外されています。

配列の複製

グリッド内にある配列は複製できます。VuGen によって，配列と同一構造がスキーマカラムとその値セットに追加されます。

配列を複製するには，親ノードを右クリックし，［**配列要素の複製**］を選択します。

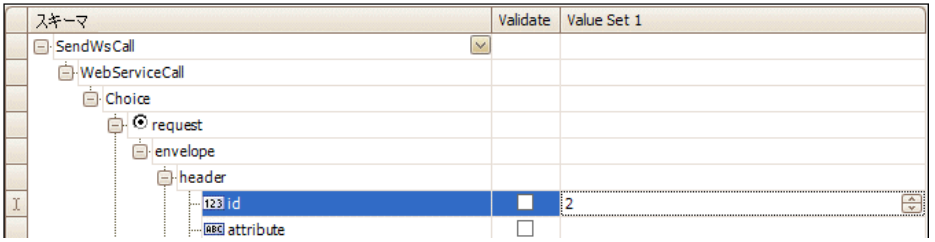


データ型

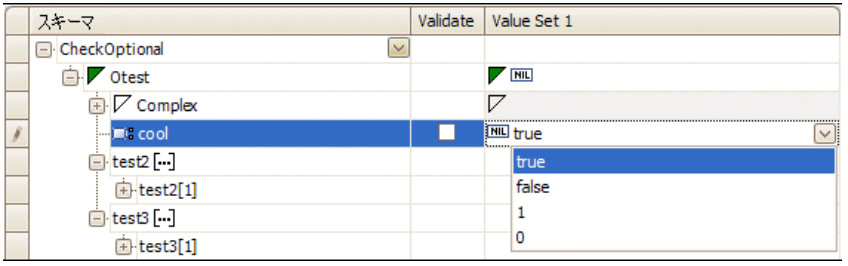
XML エディタでは，要素とサブ要素の配列を操作し，値を割り当てることができます。

実際のデータ型を決めるには，要素名（**123**，**ABC**，**B64** など）の横にあるアイコンの上にマウスを移動します。マウスオーバー・ポップアップにデータ型が表示されます。

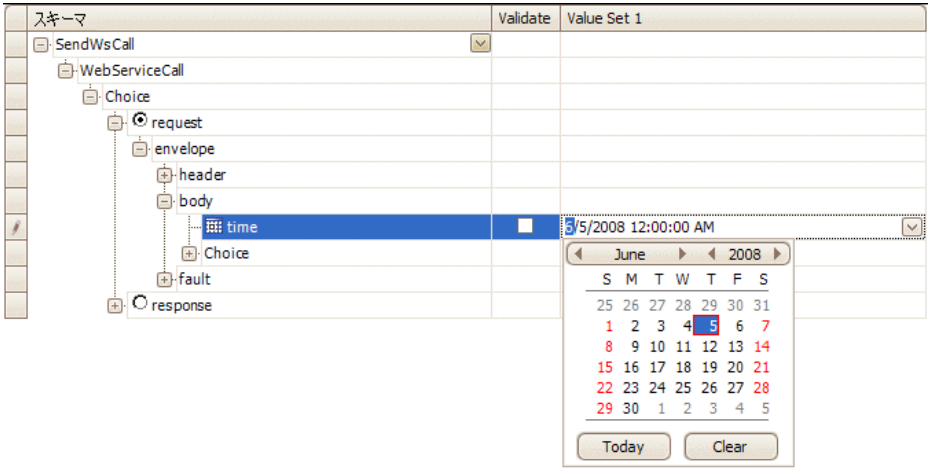
グリッドによって要素のデータ型が認識され，値を設定するためのインタフェースが提供されます。たとえば，**Int** 型の場合は，値セルに番号スクロール・コントロールが含まれています。



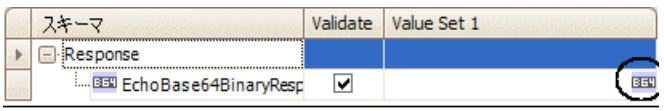
ブール型の場合は，値セルにリスト・ボックスと値（**true**，**false**，**1**，または **0**）が含まれています。



Date 要素の場合は，カレンダーを開いて日付を選択できます。



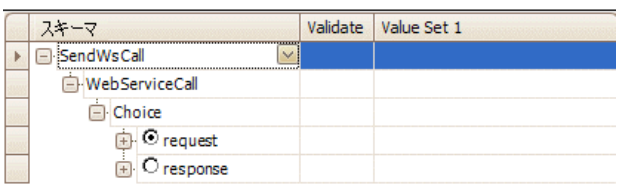
このスキーマは，値ボックスの右に **Base64** 要素と **B64** ボタンを示します。このボタンをクリックすると，［Process Base 64 Data］ダイアログ・ボックスが開きます。詳細については，110 ページ「Base 64 エンコーディング」を参照してください。



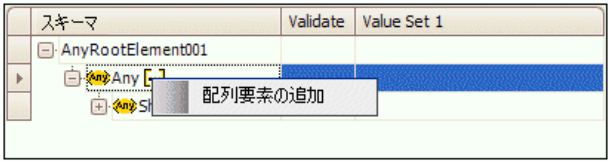
Any 型要素

XML エディタを使用すると，＜ **any** ＞型要素を操作できます。

単純で配列でない Any 要素の場合は，グリッド表示に要素が表示されます。

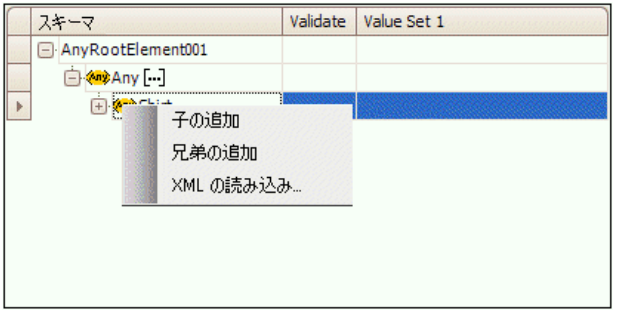


Any 要素の配列の場合は，要素，その名前，および値を操作できます。右クリック・メニューを使用して，要素を追加します。



Any 配列要素の操作

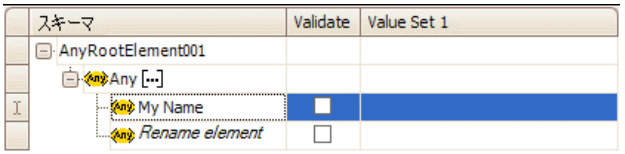
Any 配列要素を操作するには，右クリック・メニューを開きます。



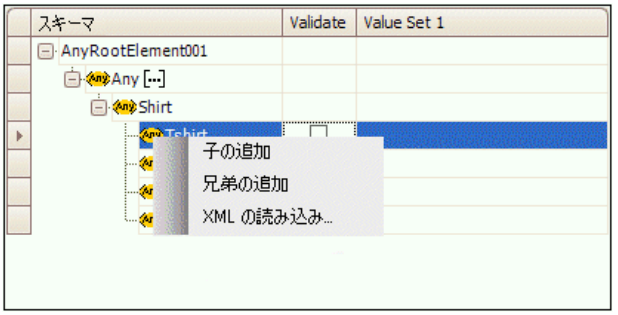
右クリック・メニューは，カーソルの位置に応じて，以下のオプションの一部または全部を提供します。

- ［**配列要素の削除**］：選択配列要素を削除します。
- ［**子の追加**］：選択された要素にサブ要素を追加します。
- ［**XML のロード**］：XML ファイルから要素値をロードします。
- ［**XML の保存**］：配列を XML ファイルとして保存します。
- ［**XML のコピー**］：選択した要素の完全な XML をクリップボードにコピーします。

Any 要素に名前を付けるには，**Rename Element** テキストをクリックし，名前を入力します。

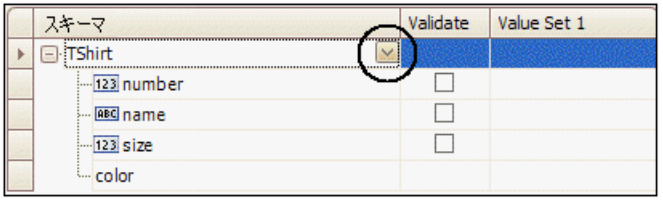


子要素を操作しているときは，右クリック・メニューを使って，兄弟要素を追加します。



複数のルート

Web サービスのスキーマに複数のルート要素がある場合は，要素の 1 つを選択できます。ルート名の横にある小さなボタンをクリックすると，ルート・ドロップダウン・ボックスが開きます。



XML エディタの統合

スタンドアロン・エディタとして役立つだけでなく，XML エディタは Service Test コンポーネントのいくつか（パラメータ化，XML 検証，サービス・エミュレーションなど）に統合されています。

パラメータ化では，『**第 1 巻 ー VuGen の使用**』ので説明しているように，エディタを利用してパラメータ要素と値を表示します。

第 3 章

Web サービス－管理

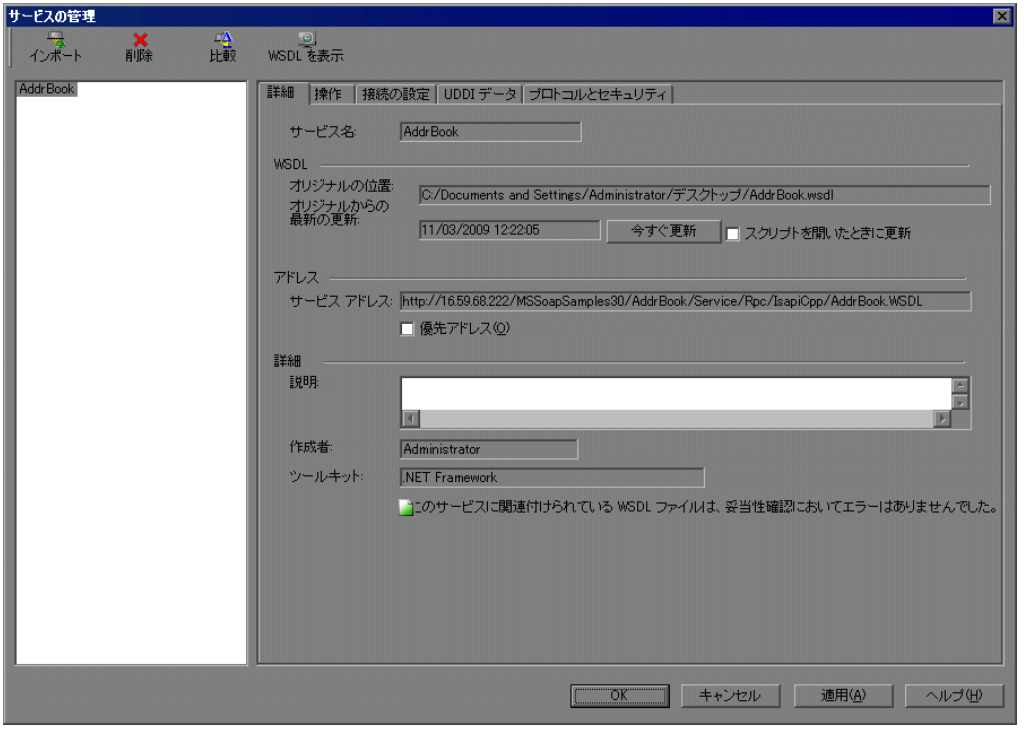
VuGen には，サービス・エントリに関連付けられている WSDL ファイルを検証，管理するユーティリティがあります。

本章の内容

以下の情報は，**Web** サービスと **SOA Vuser** スクリプトのみを対象とします。

Web サービス仮想ユーザ・スクリプトの管理について

［サービスの管理］ウィンドウでは，現在のスクリプト用のサービス・エントリのリストを管理できます。それぞれのサービス・エントリのプロパティを確認および設定できます。



サービス・エントリをリストに追加するには，WSDL ファイルをインポートします。WSDL をリストに追加すると，VuGen によって作業用のコピーが作成され，スクリプトと一緒に保存されます。これは共有可能なコピーではありません。したがって，作成するスクリプトごとに，必要な WSDL ファイルをインポートする必要があります。

ローカルに保存された WSDL のコピーを Internet Explorer で表示するには，［**WSDL の表示**］ボタンをクリックします。

注：WSDL ファイルに対する検証と変更は，すべて作業用コピーを対象に実行されます。インポートされた WSDL ファイルを新しいバージョンに置き換える場合は，43 ページ「詳細」で説明しているように，［**今すぐ更新**］オプションを使います。

［サービスの管理］ウィンドウを開くには，［**SOA ツール**］＞［**サービスの管理**］を選択するか，［サービスの管理］ツールバー・ボタンをクリックします。

［サービスの管理］ウィンドウには次のインタフェースが用意されています。

- サービス・プロパティの表示と設定
- サービスのインポート
- サービスの削除
- WSDL ファイルの比較
- WSDL ファイルの表示

サービス・プロパティの表示と設定

［サービスの管理］ウィンドウでは，インポートした WSDL に関する情報を表示したり変更できます。このウィンドウには，以下のタブで選択したサービス・エントリに関する詳細が表示されます。

- 詳細
- 操作
- 接続の設定
- UDDI データ

詳細

［サービスの管理］ウィンドウの［**詳細**］タブには，サービスに関する情報（［WSDL］，［アドレス］，［詳細］）が表示されます。［詳細］領域には，サービスに関する注釈またはメモを追加できます。

WSDL

WSDL 領域には，WSDL の場所に関する情報と，WSDL が最後にインポートされた日付が表示されます。

- ［**オリジナルの位置**］：WSDL ファイルの元のソース（読み取り専用）。
- ［**サービス名**］：Web サービスの名前。
- ［**オリジナルからの最新の更新**］：（Quality Center からインポート**しない**サービスの場合のみ）ローカル・コピーが元のソースからの WSDL ファイルで更新された最終日付。
  - WSDL の作業コピーを手作業で更新するには，［**今すぐ更新**］をクリックします。VuGen によって，既存の WSDL がバックアップされ，前述の場所から更新されます。
  - スクリプトを開くたびに WSDL を更新するよう VuGen に指示するには，［**スクリプトを開いたときに更新**］を選択します。
  - QC からのサービスの最終日付
- ［**QC からの最終更新**］：（Quality Center からインポートしたサービスの場合）サービスが Quality Center から更新された最終日付。

アドレス

- ［**サービス アドレス**］：要求の送信先となるエンドポイント・アドレスです。

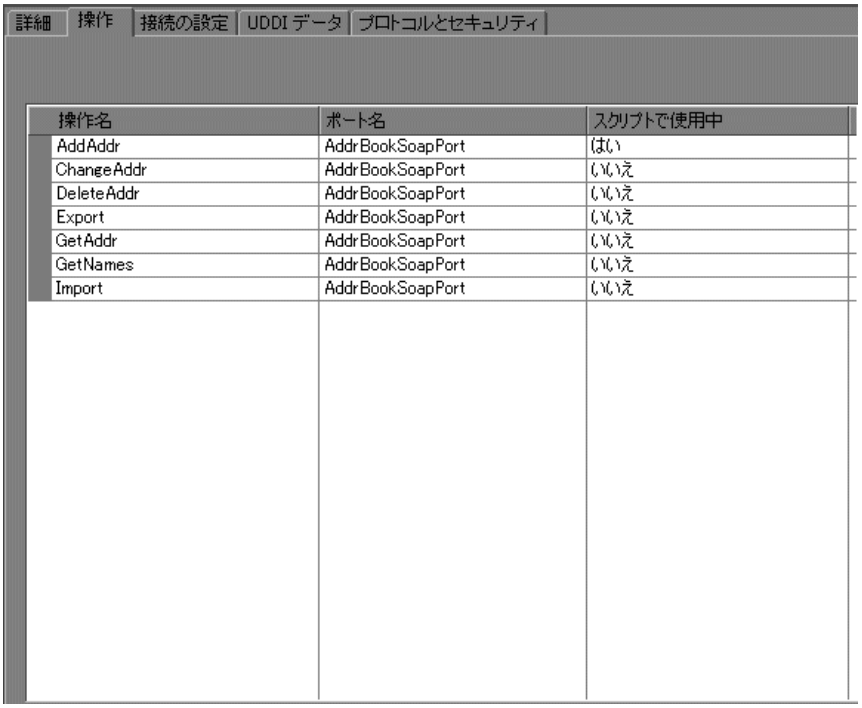
  WSDL ファイルに指定されているエンドポイントをオーバーライドするには，［**優先アドレス**］を選択し，［**サービス アドレス**］ボックスに別のアドレスを指定します。

詳細

- ［**詳細**］：標準の設定では WSDL ファイルから取得される Web サービスの説明です。このテキスト領域は編集可能です。
- ［**作成者**］：最初にサービスをインポートしたユーザの名前です（読み取り専用）。
- ［**ツールキット**］：スクリプトと関連するツールキット。これは最初の WSDL ファイルをインポートする前に設定します。

操作

インポートした各サービスで複数の操作を定義できます。［**操作**］タブには，左の表示枠で選択したサービスに使用される操作が表示されます。



操作のリストを並べ替えるには，関連するカラムをクリックします。たとえば，操作を名前でリストアップするには，［**操作名**］カラムをクリックします。操作を降順でリストアップするには，カラム名を再度クリックします。小さな矢印は，ソートされたカラムであることを示しています。上向き矢印は昇順であることを示し，下向き矢印は降順であることを示しています。

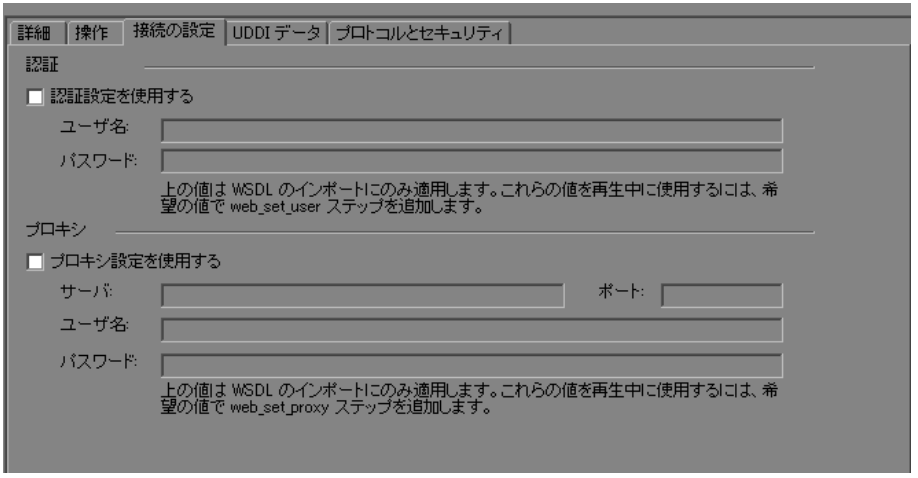
接続の設定

一部の場合には，認証を要求するセキュア・サイトに WSDL があります。また場合によっては，プロキシ・サーバを通じて WSDL にアクセスします。

VuGen では，セキュリティを使用する WSDL とプロキシ・サーバを通じてアクセスする WSDL のインポートをサポートしています。以下のセキュリティと認証方法をサポートしています。

- SSL
- 基本認証と NTLM 認証
- .NET および汎用ツールキット用 Kerberos

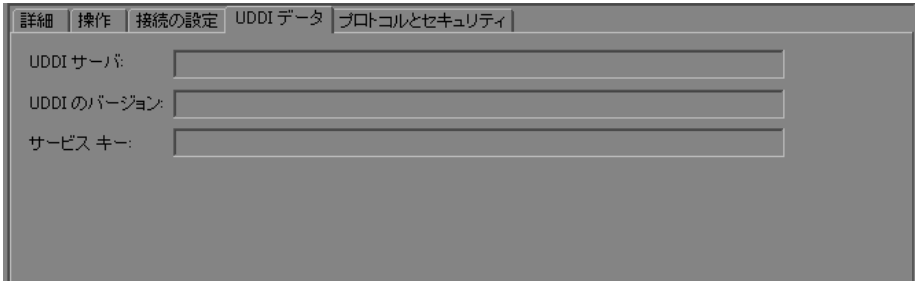
WSDL をインポートするときは，認証またはプロキシ情報を入力することをお勧めします。しかし，設定が変更されている場合は，サービス・マネージャの［**接続の設定**］タブで変更できます。



WSDL をインポートするときの接続情報の設定に関する詳細は，49 ページ「接続オプション」を参照してください。

UDDI データ

UDDI レジストリからインポートしたサービスごとに UDDI サーバの詳細を表示できます。



UDDI サーバの URL，UDDI のバージョン，およびサービス・キーの情報を読み取り専用で表示します。

UDDI からのインポートについては，50 ページ「UDDI サーバ上のサービスの指定」を参照してください。

プロトコルとセキュリティ設定

［プロトコルとセキュリティ］タブには，スクリプトに適用されるセキュリティ・シナリオの詳細が表示されます。シナリオを選択しなかった場合は，標準設定の＜**シナリオなし**＞が使用されます。

このタブに情報が読み取り専用で表示されているときに，［**データの編集**］ボタンをクリックして，セキュリティ設定を変更します。詳細については，168 ページ「セキュリティ・シナリオの設定」を参照してください。

サービスのインポート

VuGen では，Web サービス呼び出しステップを使用する高レベルのテストを作成するためにサービスをインポートできます。通常は，WSDL ファイルをインポートすることでスクリプトの作成を開始します。

ファイルをインポートするときは，以下の情報を指定します。

- **ソース**：WSDL のソース：URL，ファイル，UDDI，または Quality Center
- **場所**：WSDL のパスまたは URL，手作業または参照で入力
- **ツールキット**：スクリプトのすべてのサービスと永久的に関連しているツールキット（スクリプトに最初に追加されたサービスでのみ利用可能）
- **接続オプション**：認証またはプロキシ・サーバ情報（オプション）

インポート時に VuGen によって WSDL ファイルに問題があることが検出されると，警告が表示され，レポートを開くよう要求されます。レポートにはエラーが一覧表示され，それらのエラーの説明が示されます。

ソース

WSDL を指定するときに，以下のソースを指示できます。

- ［**URL**］：サービスの完全な URL。
- ［**ファイル**］：WSDL ファイルの完全パスと名前。
- ［**UDDI**］：Universal Description, Discovery and Integration の略。サービスの共通リポジトリです。詳細については，50 ページ「UDDI サーバ上のサービスの指定」を参照してください。
- ［**Quality Center**］：Quality Center リポジトリに格納されたサービスです。詳細については，51 ページ「Quality Center からのサービスの選択」を参照してください。

VuGen では，セキュアな URL または UDDI がサポートされています。認証またはプロキシ・サーバ経由のアクセスが要求されます。詳細については，50 ページ「UDDI サーバ上のサービスの指定」を参照してください。

場所

場所入力ボックスで，WSDL のパスまたは URL を指定します。

URL または UDDI オプションの場合は，短縮版ではなく，完全な URL を挿入してください。標準設定のブラウザを開くには，テキスト・ボックスの右側にある［**参照**］ボタンをクリックします。

ファイルの場合は，テキスト・ボックスの右にある［**参照**］ボタンをクリックして，ファイル・システム上で WSDL を探します。

Quality Center の場合は，［**Quality Center への接続**］ボタンをクリックして，サーバの URL を指定し，接続を開始します。詳細については，51 ページ「Quality Center からのサービスの選択」を参照してください。

ツールキットの選択

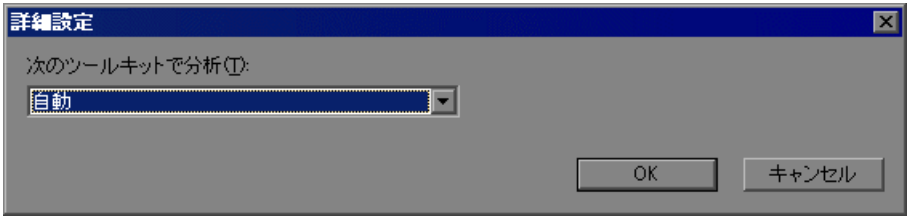
ツールキットを選択すると，VuGen にエミュレーションではなく，実際のツールキットを使って，実際のクライアント・トラフィックを送信するよう指示します。選択したツールキットは，スクリプトに永続的に関連付けられ，以降のすべての記録，インポート，および再生で使用されます。

VuGen では，.NET Framework と WSE 2 バージョン SP3 および Axis/Java ベースの Web Services Framework ツールキットをサポートしています。VuGen は実際の .NET または Axis ツールキットを使用して，スクリプトをインポート，記録，および再生します。

標準設定では，VuGen は自動検出によって最適なツールキットを判断します。

ツールキットを選択するには，次の手順を実行します（新規スクリプトの場合のみ）。

1 ［サービスのインポート］ダイアログ・ボックスで，［**詳細設定**］をクリックします。



2 ツールキットを選択するか，標準の**自動**検出を使用します。

接続オプション

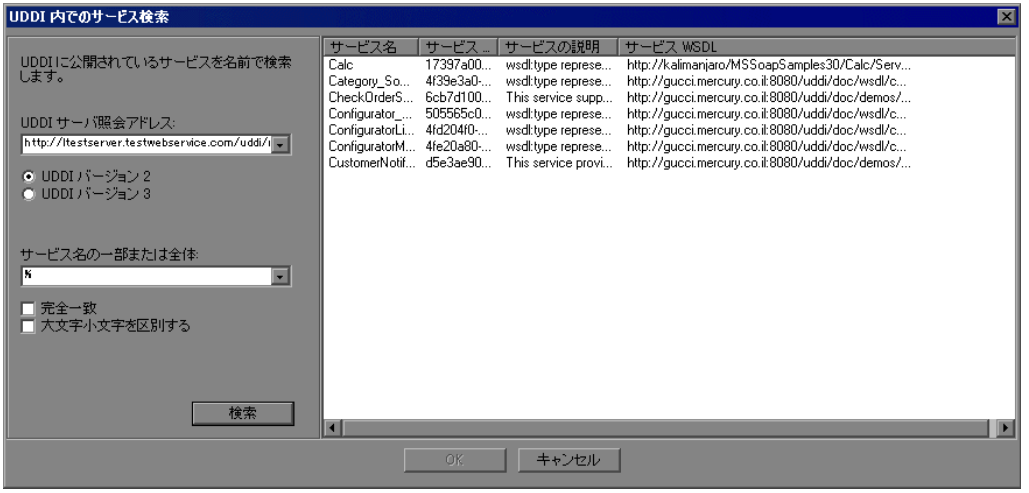
URL または UDDI から WSDL ファイルをインポートする場合，そのファイルがセキュアな場所にあれば，認証を要求されることがあります。場合によって，WSDL へのアクセスはプロキシ・サーバ経由になります。［接続の設定］ボタンを使うと，この情報を指定できます。詳細については，52 ページ「WSDL 接続オプションの指定」を参照してください。

UDDI サーバ上のサービスの指定

サービス・プロバイダは公開された Web サービスを登録および分類し，検索機能を提供します。UDDI ビジネス・レジストリは，WSDL で記述された Web サービス用のサービス・ブローカの例です。

Web サービス・クライアントは，UDDI のようなブローカ・サービスを使用して，必要な WSDL ベースのサービスを検索できます。該当サービスが見つかったら，サーバにバインドしてサービス・プロバイダを呼び出します。

［UDDI 内でのサービス検索］ダイアログ・ボックスを開くには，［**参照**］ボタンをクリックします。



UDDI サーバ上のサービスを検索するには，次の手順を実行します。

1 ［**UDDI サーバ照会アドレス**］ボックスに UDDI サーバの URL を入力します。

2 UDDI バージョンを指定します。

3 サービスの名前または名前の一部を指定します。該当する場合，［**完全一致**］または［**大文字小文字を区別する**］を選択して検索対象を絞り込みます。ワイルドカード検索実行するには，パーセント（%）文字を使います。

4 ［**検索**］をクリックします。VuGen には，マッチング結果がすべて表示されます。

5 リスト内でサービスをダブルクリックしてインポートします。

Quality Center からのサービスの選択

Service Test Management アドオンを装備した HP Quality Center は VuGen と統合されています。この統合によって，サービス・エントリとテストを Quality Center に保管できます。また，テスト要件とテスト計画に従ってサービスを作成して編成することもできます。

Quality Center からサービスを指定するには，次の手順を実行します。

1 ［サービスのインポート］ダイアログ・ボックスで，［**Quality Center**］を選択します。

2 Quality Center プロジェクトにまだ接続していなければ，［Quality Center への接続］をクリックして［接続］ダイアログ・ボックスを開きます。Quality Center に接続する方法の詳細については，『**第 1 巻 － VuGen の使用**』の「Quality Center を使ったスクリプト管理」を参照してください。

3 ［**参照**］ボタンをクリックして，Quality Center に保存されているサービス・エントリのリストを表示します。

4 リスト内でサービスをダブルクリックしてインポートします。

WSDL 接続オプションの指定

VuGen では，認証を使用する WSDL とプロキシ・サーバを通じてアクセスする WSDL のインポートをサポートしています。

セキュリティまたはプロキシ情報を入力すると，その情報は WSDL と関連付けられ，［サービスの管理］ダイアログ・ボックスの［**接続の設定**］タブに表示されます。［**最新の状態を維持する**］オプションを有効にして自動同期化を可能にすると，Service Test は認証またはプロキシ・サーバ設定を使用して，WSDL のソースにアクセスします。

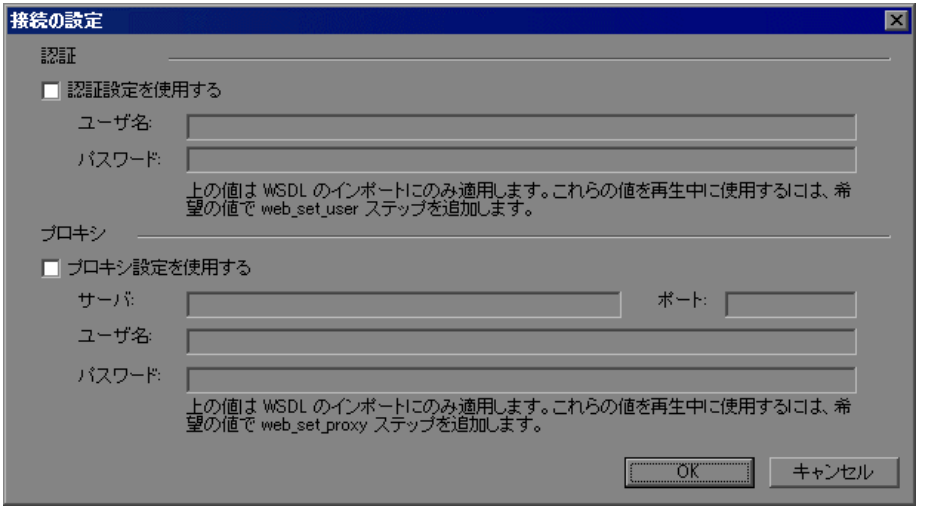
これらの接続オプションは，WSDL のインポートにのみ適用されます。再生中に認証を使ってサーバにアクセスするには，**web_set_user** または **web_set_proxy** ステップを使用します。詳細については，『**Online Function Reference**』（英語版）（［**ヘルプ**］＞［**関数リファレンス**］）を参照してください。

インポートするために認証またはプロキシ情報を指定するには，次の手順を実行します。

1 新しい Web サービス呼び出し，記録，またはトラフィック分析のいずれでも，［サービスのインポート］ダイアログ・ボックスを通常どおり開きます。

2 ［**URL**］または［**UDDI**］オプションを選択し，インポートするサービスの URL を指定します。

3 ［サービスのインポート］ダイアログ・ボックスで，［**接続の設定**］ボタンをクリックしてボックスを開きます。



4 ［接続の設定］ダイアログ・ボックスで，目的のオプション（**認証設定を使用する**，**プロキシ設定を使用する**，または両方）を選択します。

5 認証の詳細，プロキシ・サーバの場合はサーバの名前とポートを指定します。必要な証明書を指定する前にセキュア・サービスをインポートしようとすると，VuGen から情報を入力するよう要求されます。

6 セキュリティ設定を更新または変更するには，サービスの管理を開いて，左の表示枠で適切なサービスを選択します。［**接続の設定** ］タブをクリックします。必要なフィールドを編集，［**OK**］をクリックします。

サービスの削除

必要のなくなったサービス・エントリは，［サービスの管理］ダイアログ・ボックスから削除できます。サービスが更新されている場合，ソースから WSDL を同期できます。サービスを削除してインポートしなおす必要はありません。

サービスを削除する前に，そのサービスがスクリプトに必要ないことを確認します。特定のサービスに基づいてスクリプトを作成した後で，そのサービスを削除しようとすると，VuGen からその削除がスクリプトに影響を与える可能性があると警告されます。削除を行うと元に戻すことはできません。

サービスを削除するには，サービスのリストからサービスを選択し，［**削除**］ボタンをクリックします。

WSDL ファイルの比較

WSDL ファイルをインポートすると，VuGen によって作業用のコピーが作成され，スクリプトと一緒に保存されます。その結果，リソースが節約され，環境の拡張性と安定性が向上します。

ただし，スクリプトを実行するまでに，元の WSDL ファイルが変更される可能性があります。スクリプトを実行すると，テスト結果が不正確になり，スクリプトが機能しなくなるおそれがあります。したがって，以前に作成した Web サービス・スクリプトを再生するときは，その前に WSDL ファイルを対象とする比較テストを実行してください。

VuGen は比較ツールを備えています。このツールは，ローカルにコピーされた作業用の WSDL ファイルと，ファイル・システムまたは Web サーバ上にある元のファイルとを比較します。

違いが大きい場合は，［サービスの管理］ダイアログ・ボックスの［**同期化**］オプションを使用して，元のコピーから WSDL を更新できます。

また，VuGen には，任意の 2 つの XML ファイルを比較できる汎用ユーティリティも用意されています。詳細については，58 ページ「XML ファイルの比較」を参照してください。

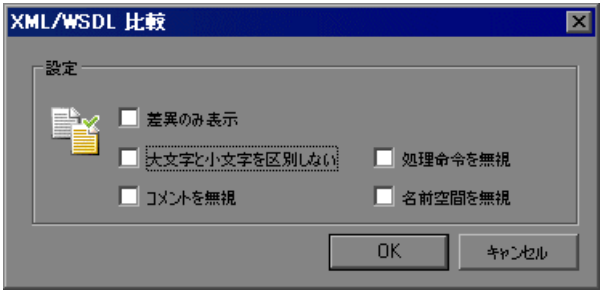
WSDL/XML 比較オプションの設定

VuGen では，WSDL ドキュメントのローカル・コピーとグローバル・コピー，または XML ファイルの改訂を比較する際に，次の比較オプションが提供されます。

- ［**差異のみ表示**］：相違のある行だけを表示します。ドキュメント全体は表示されません。
- ［**大文字と小文字を区別しない**］：テキストどうしの大文字小文字の違いを無視します。
- ［**コメントを無視**］：テキスト内のすべてのコメントを無視します。
- ［**処理命令を無視**］：処理命令のあるすべてのテキストを無視します。
- ［**名前空間を無視**］：すべての名前空間の相違を無視します。

比較オプションを設定するには，次の手順を実行します。

1 比較設定を設定します。［**SOA ツール**］＞［**SOA 設定**］＞［**XML/WSDL 比較**］を選択します。［XML/WSDL 比較］ダイアログ・ボックスが開きます。目的とするオプションを選択します。



2 ［**OK**］をクリックします。

注：比較オプションの設定は，［サービスの管理］ウィンドウ内での WSDL 比較と，［**SOA ツール**］メニューからアクセスする XML 比較の両方に適用されます。

比較レポート

VuGen は，ファイル間の相違を比較レポートに一覧表示します。

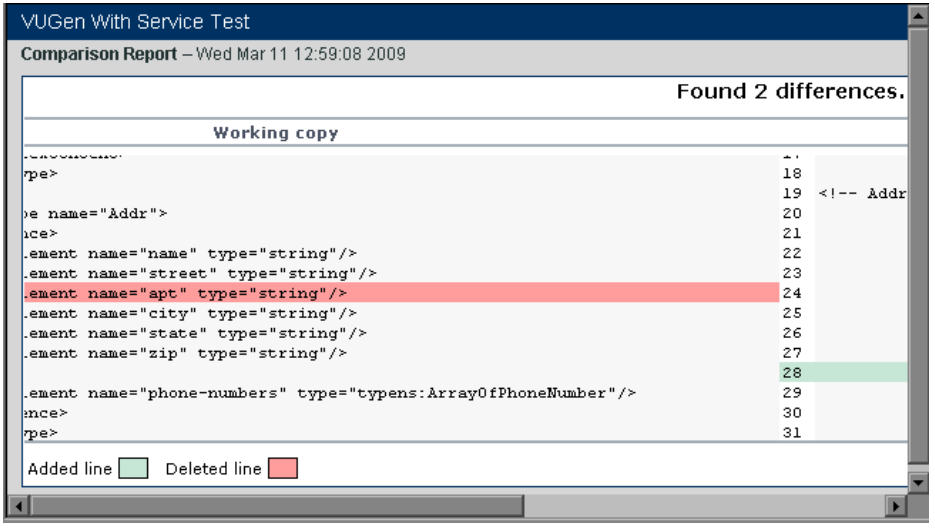
WSDL 比較レポートには，［**Working Copy**］と［**Original File**］の 2 つのカラムがあります。［Working Copy］はスクリプトと一緒に格納されている WSDL です。一方，［Original File］は元の場所（ネットワーク・ファイル・パスまたは URL）にある WSDL です。

XML 比較レポートには，各カラムに XML ファイルのパスが表示されます。

比較レポートでは，2 つのファイル間の相違を示すために次の凡例を使用します。

- **黄**：既存の要素の変更（双方のバージョンに表示）
- **緑**：新しい要素の追加（元のファイル・コピーに表示）
- **桃**：要素の削除（作業用コピーに表示）

次の例では，24 行目が元のコピーから削除され，28 行目が追加されています。

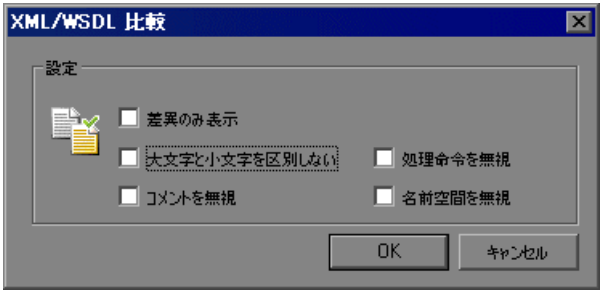


WSDL 比較の実行

ファイル比較を実行したら，変更を無視するか（存在する場合），WSDL ファイルをロードしなおすかを決定します。

WSDL ファイルを比較するには，次の手順を実行します。

1 比較設定を設定します。［**SOA ツール**］＞［**SOA 設定**］＞［**XML/WSDL 比較**］を選択します。［XML/WSDL 比較］ダイアログ・ボックスが開きます。目的とするオプションを選択します。



2 ［サービスの管理］ウィンドウを開きます。［**SOA Tools**］＞［**サービスの管理**］を選択するか，［**サービスの管理**］ツールバー・ボタンをクリックします。

3 比較を実行する対象となるサービスを選択します。比較の対象にできるサービスは一度に 1 つだけです。

4 ［**比較**］をクリックします。［WSDL 比較レポート］が開きます。

5 ファイルを下にスクロールして相違点を探します。

2 つのファイルの間で相違が見つかり，WSDL ファイルの VuGen 側の作業用コピーを更新する場合は，左側の表示枠にあるツリーで WSDL ファイルをクリックします。右クリックして表示されるメニューから［**グローバル コピーからファイルを更新**］を選択します。これで，現在のバージョンの WSDL がスクリプトの WSDL ディレクトリにコピーされます。

6 ［WSDL 比較レポート］ウィンドウを閉じるには，［**ファイル**］＞［**終了**］を選択します。

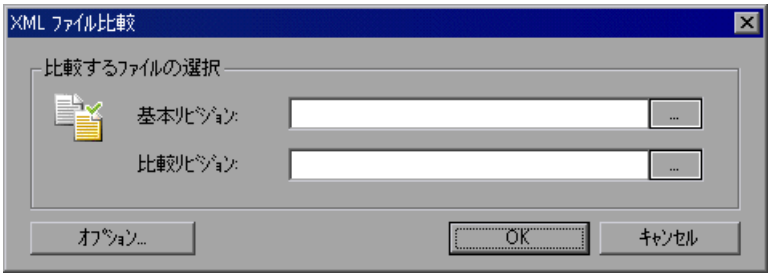
XML ファイルの比較

VuGen は，2 つの XML ファイルを比較できるユーティリティを備えています。

大文字小文字の区別やコメントなど，無視してよい相違点を指定できます。比較オプションの詳細については，55 ページ「WSDL/XML 比較オプションの設定」を参照してください。

2 つの XML ファイルを比較するには，次の手順を実行します。

1 ［**SOA ツール**］＞［**XML ファイルの比較**］を選択します。［XML ファイル比較］ダイアログ・ボックスが開きます。



2 ［**基本リビジョン**］ボックスの右にある参照ボタンをクリックして，元の XML ファイルを探します。

3 ［**比較リビジョン**］ボックスの右にある参照ボタンをクリックして，新しい XML ファイルを探します。

4 ［**OK**］をクリックします。VuGen によって［XML 比較レポート］ウィンドウが開かれます。

比較レポートの詳細については，56 ページ「比較レポート」を参照してください。

WSDL ファイルの表示

［サービスの管理］ウィンドウでは，標準設定のブラウザに WSDL を表示できます。

WSDL を表示するには，次の手順を実行します。

1 左側の表示枠で WSDL を選択します。

2 ［サービスの管理］ウィンドウで［**WSDL の表示**］ボタンをクリックします。

第 4 章

Web サービス－スクリプト内容の追加

VuGen を使用して，記録，手作業による呼び出しの追加，またはサーバ・トラフィックの分析によって Web サービスをテストするスクリプトを作成します。

本章の内容

以下の情報は，**Web** サービスと **SOA Vuser** スクリプトのみを対象とします。

Web サービス・スクリプトへの内容の追加について

Web サービス・スクリプトでは，Web サービス・クライアントをエミュレートして環境をテストできます。

空の Web サービス・スクリプトを作成した後で，24 ページ「空の Web サービス・スクリプトの作成」で説明しているように，記録，手作業による Web サービスの挿入，SOAP のインポート，またはサーバ・トラフィックの分析によって内容を追加します。

スクリプトの作成方法	参照先
記録	「Web サービス・スクリプトの記録」
Web サービス呼び出しの手作業での挿入	68 ページ「新しい Web サービス呼び出しの追加」
SOAP 要求のインポート	71 ページ「SOAP 要求のインポート」
サーバ・トラフィックの分析	第 5 章「Web サービス－サーバ・トラフィック・スクリプト」

Web サービス・スクリプトの記録

Web サービス・セッションを記録して，一般的なビジネス・プロセスのイベントをキャプチャします。Web サービスと対話するクライアントをすでに作成している場合は，クライアントが実行するアクションをすべて記録できます。結果として作成されるスクリプトは，Web サービス・クライアントの操作をエミュレートします。記録した後で，さらに Web サービス呼び出しを追加したり，その他の拡張を行うことができます。

記録のためのサービスの指定

アプリケーションを記録するときは，Web サービス WSDL ファイルを使用するようにも，または WSDL ファイルを使用しないようにも，記録できます。可能であれば，WSDL で記録することをお勧めします。

WSDL ファイルを含める場合，VuGen では使用するメソッドを選択してそれらの引数の値を入力することでスクリプトを作成できます。VuGen では，WSDL に変更があると簡単に更新できる説明的なスクリプトが生成されます。

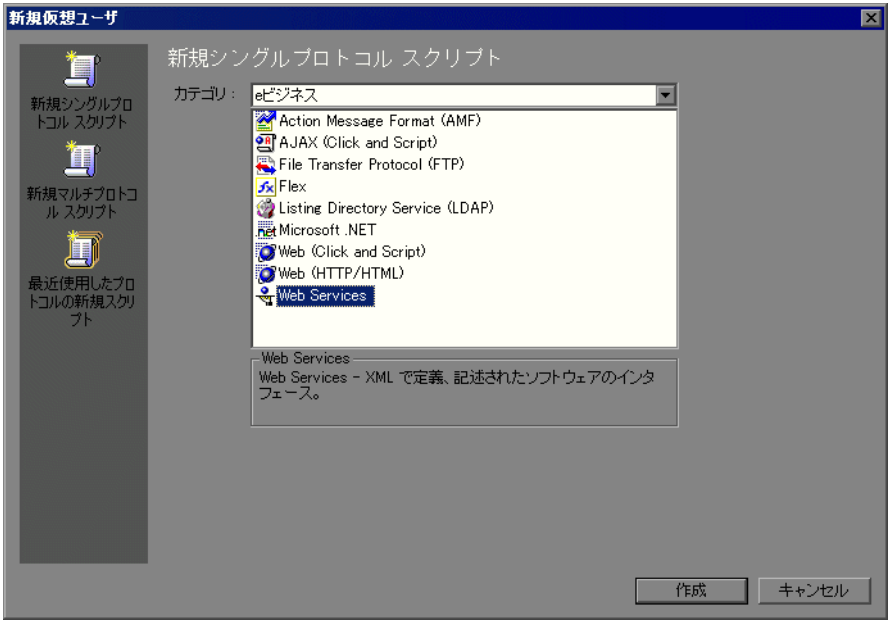
WSDL ファイルを指定しない場合（非推奨），VuGen では Web サービス呼び出しステップではなく，SOAP 要求を使用してテストが作成されます。サービスをインポートしないでスクリプトを作成すると，VuGen では **soap_request** ステップが作成されます。ただし，その引数を保持するのは困難です。

記録によって Web サービス・スクリプトを作成するには，次の手順を実行します。

1 **空のスクリプトを作成します。**

［**ファイル**］＞［**新規作成**］を選択して，［新規仮想ユーザ］ダイアログ・ボックスを開きます。

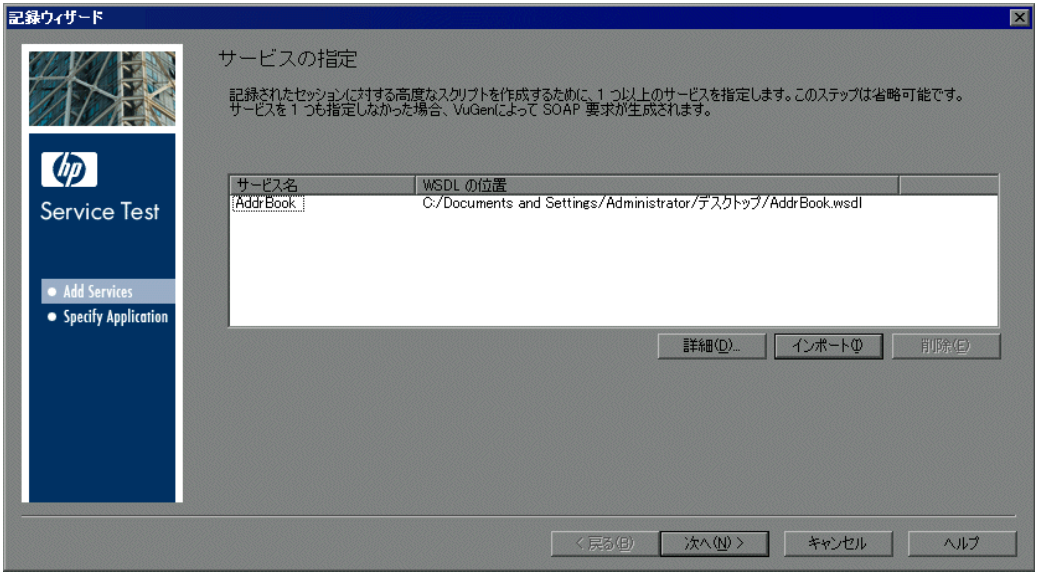
シングル・プロトコル・スクリプトの場合は，左の表示枠で［**新規シングル プロトコル スクリプト**］をクリックします。［e- ビジネス］カテゴリから［**Web サービス**］プロトコルを選択します。［**OK**］をクリックします。



Web サービスや Web など，いくつかの異なるプロトコルを記録する必要がある場合は，左の表示枠で［**新規マルチ プロトコル スクリプト**］をクリックして，目的のプロトコルを指定します。［**OK**］をクリックします。

2 記録プロセスを開始します。

［**記録開始**］ボタンをクリックするか，Ctrl+R キーを押して，［サービスの指定］画面を開きます。



3 リストにサービスを追加します。

高レベルの Web サービス・スクリプトを作成するには，［**インポート**］ボタンを使用して 1 つ以上のサービスを追加します。記録された Web サービスの WSDL が使用できる場合は，インポートすることをお勧めします。サービスをインポートしないでスクリプトを作成すると，VuGen によって **soap_request** ステップが作成されます。［サービスのインポート］ダイアログ・ボックスが開きます。

a WSDL のソースと場所を選択します。

b ツールキットを選択します。選択したツールキットは，スクリプトと永続的に関連付けられます。詳細については，47 ページ「サービスのインポート」を参照してください。

c［**インポート**］をクリックします。

インポートする WSDL ごとに，前述の手順を繰り返します。

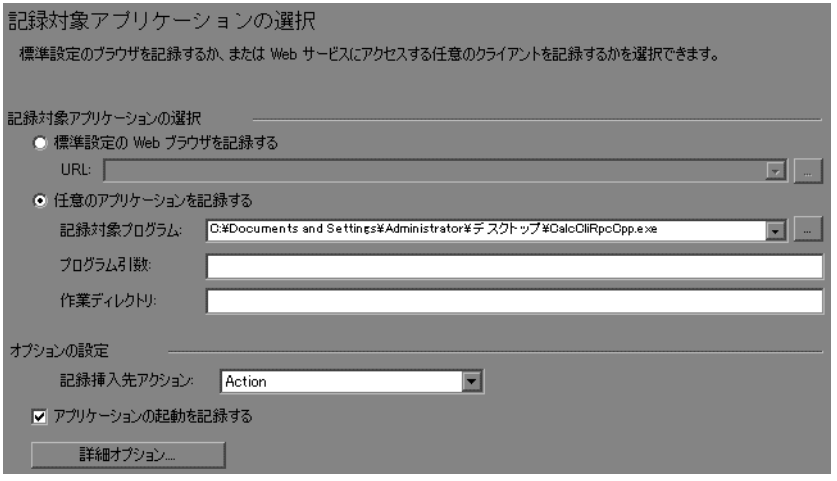
4 Web サービスの詳細を入力します。

［**詳細**］をクリックして，追加する Web サービスの詳細を入力します。詳細については，第 6 章「Web サービス呼び出しのプロパティ」を参照してください。

次の手順では，記録対象アプリケーションを選択します。

記録対象アプリケーションの選択

この画面で，記録対象アプリケーションを指定します。ブラウザ・セッションまたはクライアント・アプリケーションを記録できます。



1 記録の詳細を指定します。

- ［**標準設定の Web ブラウザを記録する**］：指定した URL で始まる，標準設定の Web ブラウザのアクションを記録します。Web ベースの UI で Web サービスにアクセスする場合は，このオプションを選択します。
- ［**任意のアプリケーションを記録する**］：クライアント・アプリケーションのアクションを記録します。［**記録対象プログラム**］ボックスで，パスを指定または参照します。関連する引数や作業用ディレクトリをすべて指定します。

2 オプションを設定します。

- ［**記録挿入先アクション**］：コードを生成して挿入する対象となるアクション。繰り返す必要のない起動処理がある場合は，それらを **vuser_init** セクションに置きます。記録中に **Action** などの別のセクションに切り替えることができます。
- ［**アプリケーションの起動を記録する**］：アプリケーションの起動処理をスクリプトの一部として記録します。特定の位置から，起動処理を含めずに記録を開始する場合は，このチェック・ボックスをクリアします。
- ［**詳細オプション**］：［記録オプション］ダイアログ・ボックスが開き，スクリプトの生成方法をカスタマイズできます。詳細については，スクリプト生成オプションの設定に関するセクションを参照してください。

3［**完了**］をクリックします。

VuGen によってアプリケーションが開かれ，記録が始められます。対象アプリケーション内で必要なアクションを実行し，その後フローティング・ツールバーの［**停止**］ボタンを押します。WSDL をインポートしなかった場合，VuGen によって **web_service_call** 関数または **soap_request** 関数が生成されます。

記録後，追加サービス呼び出しとパラメータ化でスクリプトを拡張できます。完成したスクリプトを VuGen で実行して，その機能をチェックします。完成したスクリプトを VuGen で実行して，その機能をチェックします。

詳細については，22 ページ「Web サービス仮想ユーザ・スクリプトの概要」を参照してください。

ワークフローの表示

トラフィックを記録または分析してスクリプト内容を追加するときに，VuGen のワークフローはスクリプトを準備する手順をガイドしてくれます。［**タスク**］表示枠をクリックすると，スクリプト作成の手順についての説明を読むことができ，記録に関する情報を表示できます。また，再生を検証することもできます。［**戻る**］および［**次**］ボタンを使用して，画面間を移動できます。

ワークフロー画面が表示されない場合は，ツールバーで［**タスク**］ボタンをクリックして，［タスク］表示枠を開き，タスクを選択します。

スクリプトの作成

［スクリプトの作成］ウィンドウには，Web サービス・スクリプトの作成のいくつかのガイドラインが含まれます。Web サービス・ウィザードを起動するためのリンクも含まれます。

- ［**はじめに**］：スクリプトの作成を開始する前に知っておくべき事項について説明します。
- ［**スクリプトの作成について**］：スクリプト作成の段階について説明します。
- ［**アクション**］：スクリプトのセクションとそれらの重要性について説明します。

2 つのアクションに関するリンクがあります。

- ［**記録開始**］：［記録開始］ダイアログ・ボックスを開きます。記録対象アプリケーションに関する情報をここに指定します。
- ［**トラフィックの分析**］：ネットワーク経由でキャプチャされたトラフィックを分析し，サーバをエミュレートするスクリプトを作成します。

作成サマリ

スクリプトを作成したら，［作成サマリ］画面に記録とスクリプトの作成に関する情報が表示されます。

- ［**プロトコル**］：スクリプトの作成中に使用されたプロトコルの一覧が表示されます。
- ［**アクション**］：アクションの記録先またはインポート先となったセクションを示します。

また，スクリプトを変更するための次のリンクも含まれます。

- ［**web_service_call ステートメントの追加**］：サービス，操作，および引数の値を指定して，スクリプトに **web_service_call** 関数を手作業で追加できます。
- ［**サービスの管理**］：サービス・リポジトリ・ウィンドウが開いて，スクリプトとそのプロパティに利用できるすべてのサービスの一覧が表示されます。
- ［**XML ファイル比較**］：2 つのバージョンの XML ファイルを比較できます。これは，WSDL ファイルを比較して，ファイルを最初にスクリプトにインポートしてい以来，変更があったかどうかを調べるのに役立ちます。

ワークフローのその他の詳細については，第 4 章「VuGen ワークフローの表示」を参照してください。

新しい Web サービス呼び出しの追加

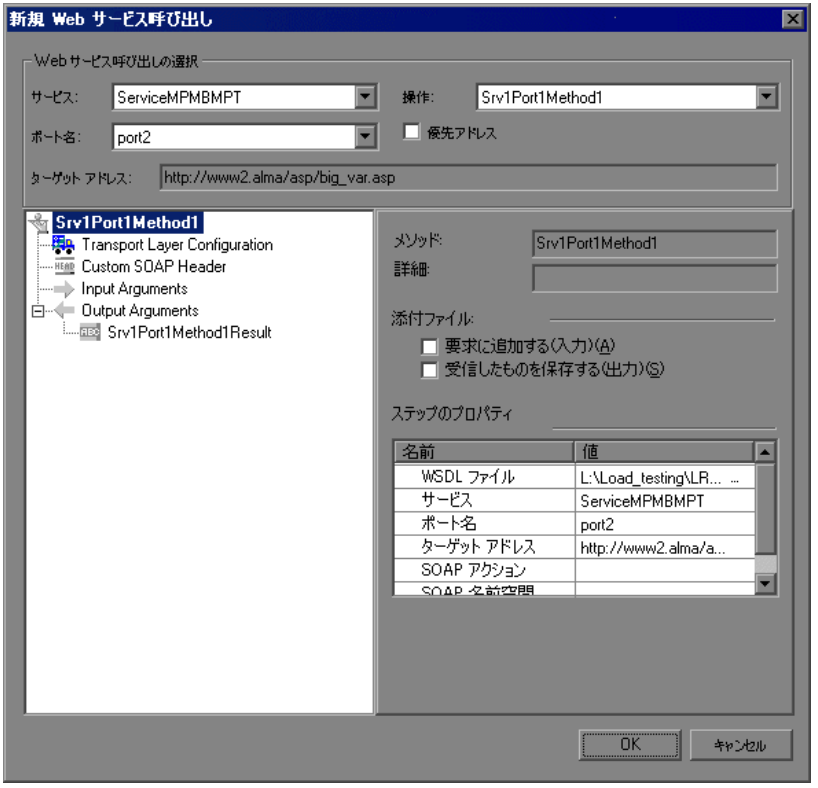
新しい Web サービス呼び出し関数は，ツリー・ビューおよびスクリプト・ビューの両方で追加できます。

Web サービス呼び出しを追加するには，次の手順を実行します。

1 スクリプト内の追加先の場所でカーソルをクリックします。

2 ［**サービス呼び出しの追加**］ボタンをクリックします。［新規 Web サービス呼び出し］ダイアログ・ボックスが開きます。目的の**サービス**を選択します。この新規のスクリプトで，まだ WSDL をインポートしていなければ，この時点でインポートします。詳細については，47 ページ「サービスのインポート」を参照してください。

3 [**操作**] を選択します。複数のポートを使用するサービスの場合は，操作の [**ポート名**] を選択します。これによって，別々のポートで実行する個別の操作を識別できます。



4 アクティブなポートに標準設定以外の宛先アドレスを指定するには，[**優先アドレス**] を選択してアドレスを入力します。

5 サービスにサンプル入力値を与えるには，最高レベル・ノード（サービス名）をクリックして，[**入力引数用の自動値を生成する**] を選択します。VuGen によって，サンプル値が追加され，各引数の前に自動値の使用を示す矢印が配置されます。

すべての入力引数にサンプル値を与えるには，[**入力引数**] ノードを選択して，[**生成**] をクリックします。

6 操作の入力引数をパラメータ化する方法の詳細については，『**第 1 巻 － VuGen の使用**』の第 14 章「ファイル，テーブル，および XML パラメータ・タイプ」を参照してください。

7 ［**トランスポート層の設定**］ノードを選択して，SOAP メッセージ用の JMS トランスポート（Axis ツールキットのみ），非同期メッセージング，WS Addressing などの詳細オプションを指定します。詳細については，第 11 章「Web サービス－トランスポート層とカスタマイズ」を参照してください。

8 各ノードをクリックして引数値に対する設定を指定します。詳細については，第 6 章「Web サービス呼び出しのプロパティ」を参照してください。

9 引数に添付ファイルを追加するには，または出力引数のパラメータを指定するには，操作のメイン・ノードを選択し，適切な選択を行います。詳細については，115 ページ「添付ファイル」を参照してください。

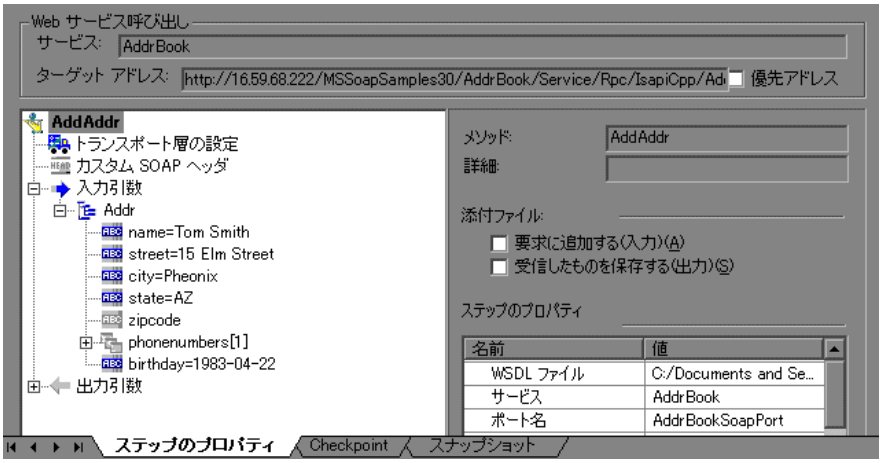
SOAP 要求のインポート

VuGen では，SOAP ファイルから Web サービス呼び出しを作成できます。SOAP 要求ファイルがある場合は，スクリプトに直接ロードできます。VuGen によって，SOAP 要求全体（セキュリティ・ヘッダを除く）と引数値が XML 要素に定義されているとおりインポートされます。SOAP をインポートすることにより，標準設定の Web サービス呼び出しにおいて引数値を手動で設定する必要はありません。

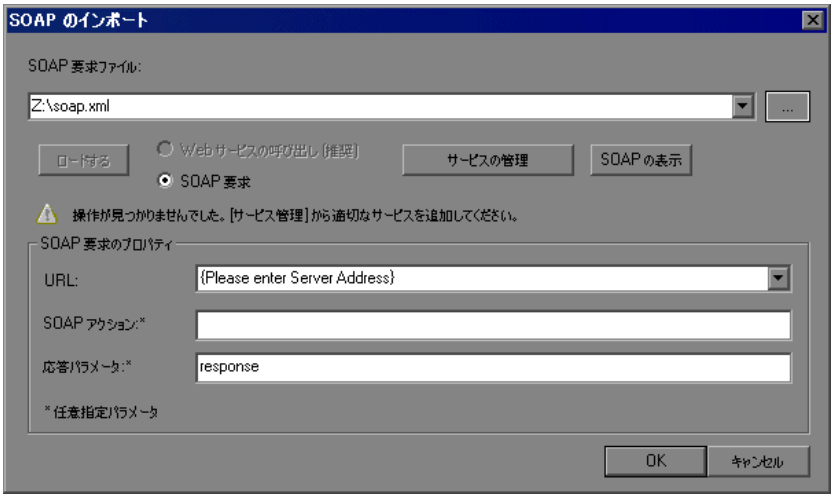
たとえば，次の要素が含まれる SOAP 要求があるとします。

```
- <soap:Body soap:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  - <q1:AddAddr xmlns:q1="http://tempuri.org/AddrBook/message/">
              <Addr href="#id1" />
    </q1:AddAddr>
   - <q2:Addr id="id1" xsi:type="q2:Addr" xmlns:q2="http://tempuri.org/AddrBook/type/">

          <name xsi:type="xsd:string">Tom Smith</name>
               <street xsi:type="xsd:string">15 Elm Street</street>
               <city xsi:type="xsd:string">Pheonix</city>
              <state xsi:type="xsd:string">AZ</state>
              <zip-code xsi:type="xsd:string">97432</zip-code>
              <phone-numbers href="#id2" />
              <birthday xsi:type="xsd:date">1983-04-22</birthday>
    </q2:Addr>
...
```

SOAP 要求をインポートすると，VuGen によって，すべての値が Web サービス呼び出しにインポートされます。



SOAP 要求に基づいて新しい Web サービス呼び出しを作成するには，まず WSDL ファイルをインポートする必要があります。WSDL が利用できなかったり，SOAP トラフィックを直接送信する必要がある場合は，SOAP 要求ステップを作成できます。サーバの URL，SOAP アクション，および応答パラメータを指定します。



『**Online Function Reference**』（英語版）（［**ヘルプ**］＞［**関数リファレンス**］）で説明しているとおり，スクリプト・ビューでは，SOAP 要求ステップが **soap_request** 関数として表示されます。

SOAP 要求をインポートするには，次の手順を実行します。

1 ［**SOAP のインポート**］ボタンをクリックするか，［**SOA ツール**］＞［**SOAP のインポート**］を選択します。

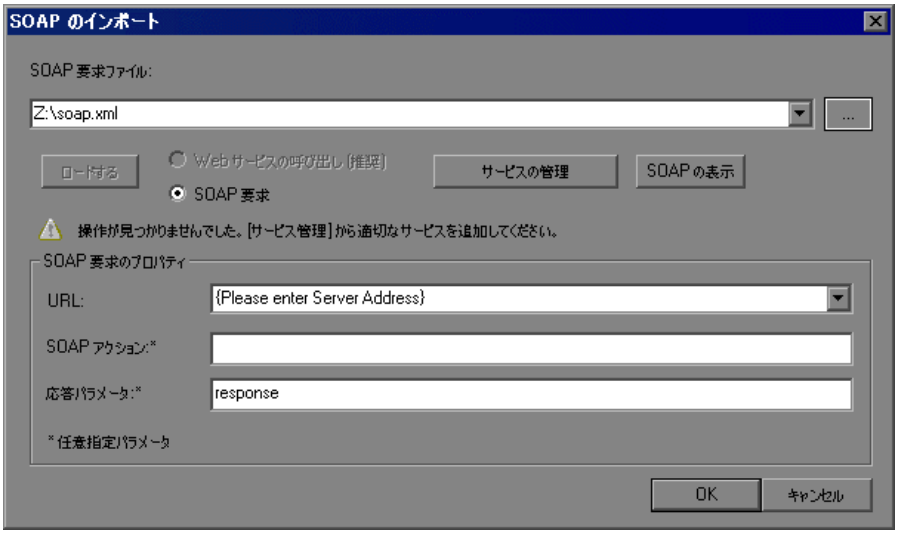
2 SOAP 要求を示す XML ファイルを参照します。

3 生成するステップのタイプ，すなわち［**Web サービス呼び出しの作成**］または［**SOAP 要求の作成**］を選択します。Web サービス呼び出しを作成するには，SOAP 要求ファイルで操作を説明する少なくとも 1 つの WSDL をまずインポートする必要があります。WSDL をインポートするには，［**サービスの管理**］をクリックして，［**インポート**］ボタンをクリックします。SOAP をロードする前に表示するには，［**SOAP の表示**］をクリックします。

4 ［**ロードする**］をクリックします。VuGen によって，XML 要素の値がロードされます。

Web サービス呼び出しでは，95 ページ「Web サービス呼び出しのプロパティについて」で説明しているように，サービス呼び出しのプロパティを設定します。

SOAP 要求の場合は，URL とほかの関連パラメータを提供します。



5 Web サービス呼び出しでは，同じ操作（メソッド）名のサービスが複数ある場合，インポートする SOAP トラフィックのサービスを選択する必要があります。追加プロパティについては，95 ページ「Web サービス呼び出しのプロパティについて」を参照してください。

6 ［**OK**］をクリックして，スクリプト内で新しいステップを生成します。

7 チェックポイントを設定して，ステップを再生します。詳細については，124 ページ「チェックポイント」を参照してください。

スクリプトの使い方

スクリプトを作成したら，次のいずれかの方法で管理できます。

- **Service Test Management**：Quality Center でサービスをインポート，保管，定義できるようにして SOA テストを管理できる HP Quality Center 用のアドオン。このセクションには要件管理，テスト計画，テスト・ラボ，および不具合管理が含まれています。詳細については，76 ページ「Service Test Management の操作」を参照してください。

完成したスクリプトを使って，複数の方法でシステムをテストできます。

- **機能テスト**：スクリプトを実行して，Web サービスが機能するか確認します。また，Web サービスによって期待値が生成されたかどうかも確認できます。詳細については，第 7 章「Web サービス ー 再生の準備」を参照してください。
- **負荷テスト**：スクリプトを LoadRunner Controller シナリオに組み込んで，負荷環境下でそのパフォーマンスをテストします。詳細については，『**HP LoadRunner Controller**』または『**Performance Center**』のドキュメントを参照してください。
- **運用テスト**：ビジネス・プロセス・モニタ・プロファイルで，Web サービスのパフォーマンスを経過時間ごとにチェックします。詳細については，『**HP Business Availability Center**』のドキュメントを参照してください。

Quality Center でのデータ管理

Quality Center を使用する場合は，Quality Center テスト・セットの各インスタンスにさまざまなパラメータ値を割り当てることができます。これによって，異なるデータを使用して同じテストを実行できます。

あるテスト・インスタンスのパラメータ値を変更しても，ほかのテスト・インスタンスには影響がありません。いつでも，元のパラメータ値に復元できます。

詳細については，『**HP Quality Center ユーザーズ・ガイド**』を参照してください。

Service Test Management の操作

HP Quality Center は Web ベースのテスト管理アプリケーションです。このセクションには要件管理，テスト計画，テスト・ラボ，および不具合管理が含まれています。

Quality Center 用の **Service Test Management** アドオンによって，Quality Center でサービスをインポート，保管，定義できるようにして SOA テストを管理できます。

Service Test Management 統合によって，次のことができます。

- **Web サービスの保管**：Web サービスは，**Service Test** で使用するために，Quality Center に保管して整理できます。
- **サービス・テストのスクリプトの作成**：Quality Center に保管した Service Test Management に基づいてスクリプトを作成でき，サービスとスクリプトのライフサイクルを通じて最新の WSDL を維持できます。
- **ビジネス・プロセス・テストの作成**：Quality Center では，**Service Test Management** で定義したサービスに基づいて BPT（ビジネス・プロセス・テスト）を作成できます。

Service Test Management には，テスト要件とテスト計画を作成するために，HP の Systinet レジストリも統合されています。サービスをインポートすると，**Service Test Management** によってサービスの変更内容が確認され，実行するのに必要なテスト・ケースが自動的に生成されます。

Quality Center 用の **Service Test Management** アドインを使用すると，組織全体のグループが次の方法で品質プロセスに貢献できます。

- ビジネス・アナリストはアプリケーション要件とテスト目標を定義します。
- テスト・マネージャとプロジェクト・リーダーはテスト計画を策定し，テスト・ケースを作成します。
- テスト・マネージャは QA テスト要件と，SOA サービスおよび環境用のテスト・アセットを自動的に作成します。
- テスト自動化エンジニアは自動化スクリプトを作成し，リポジトリに保管します。
- QA テスト担当者は手動および自動化テストを実行し，実行結果を報告して，不具合を入力します。
- 開発者はデータベースに記録された不具合を確認して修正します。

- プロジェクト・マネージャは分析のために，テストおよびリソース・データをさまざまなレポートまたはネイティブな Microsoft Excel にエクスポートできます。
- 製品マネージャはアプリケーションをリリースできるかどうかを決定します。
- QA アナリストはテスト・アセット・ドキュメントを Microsoft Word 形式で自動作成できます。

統合の詳細については，『**HP Service Test Management User Guide**』（英語版）を参照してください。

第 5 章

Web サービス－サーバ・トラフィック・スクリプト

VuGen を使用すると，サーバのトラフィックをキャプチャしたファイルを分析して，Web サービスをテストするスクリプトを作成できます。

本章の内容

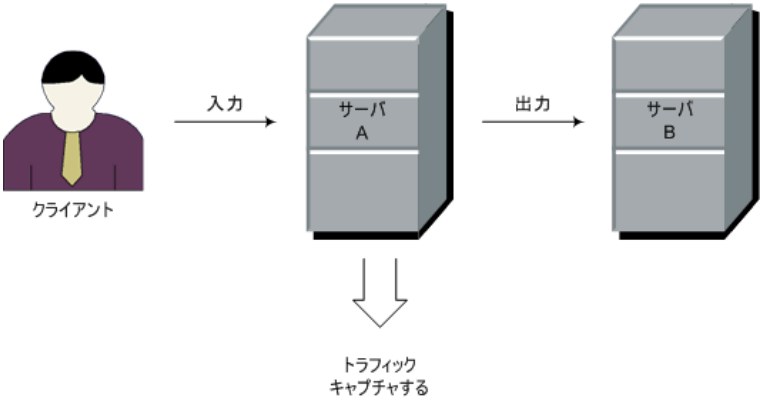
以下の情報は，**Web** サービスと **SOA Vuser** スクリプトのみを対象とします。

サーバ・トラフィック・スクリプトの作成について

エンタープライズ・システムや複合システムをテストする場合，クライアント側からパフォーマンスを測定することに重点が置かれます。通常は，VuGen によりアプリケーションまたはブラウザでユーザが実行するアクションが記録され，サーバに対するクライアントのアクションや要求をエミュレートするスクリプトが生成されます。

テスト環境によっては，サーバに対する要求を取得するためのクライアント・アプリケーションを記録することができない場合があります。これはたとえば，サーバがクライアントとして動作していたり，クライアント・アプリケーションを利用できなかったりする状況の場合にあり得ます。このような場合は，VuGen の**トラフィック分析**機能を使用してスクリプトを作成できます。

トラフィック分析機能では，サーバ・ネットワーク・トラフィックが格納されているキャプチャ・ファイルを調査することで，サーバとの間で送受信された要求をエミュレートするスクリプトを作成します。サーバ・トラフィックを分析してスクリプトを作成する手順の詳細については，次項の「サーバ・トラフィック・スクリプトの概要」で説明します。



エミュレートするスクリプトには，**受信 トラフィック**と**送信トラフィック**の 2 つのタイプがあります。

受信トラフィック・スクリプトは，サーバに要求を送信したい場合，たとえばセキュリティ上の制約があるために，クライアント・アプリケーションを利用できないような状況をエミュレートします。この場合，最も正確な解決方法は，サーバがクライアント側から**受信**するトラフィックに基づいてスクリプトを生成することです。

受信サーバ・ネットワーク・トラフィックを指定するには，サーバの IP アドレスとアプリケーションが対象としているポート番号を指定します。VuGen により，サーバが受信するすべてのトラフィックが調査され，関係するメッセージが抽出されて，スクリプトが作成されます。前出の図で，クライアントが利用できない状況では，受信スクリプトを作成して，**Server A** が受信する要求をエミュレートします。

送信トラフィック・スクリプトは，別のサーバに対してクライアントとして動作するサーバをエミュレートします。いくつかの内部サーバを持ったアプリケーション・サーバでは，サーバ・マシン間の通信のエミュレーションが必要なことがあります。たとえば前出の図の **Server A** と **Server B** の間です。この場合の解決方法は，特定のサーバから**送信**されたトラフィックに基づいてスクリプトを生成することです。

送信トラフィック・スクリプトを作成するには，送信トラフィックをエミュレートするサーバの IP アドレスを指定します。VuGen によりそのサーバから送信されるトラフィックが抽出されます。前出の図では，送信スクリプトを使用して **Server A** により **Server B** に送信される要求をエミュレートできます。

サーバ・トラフィック・スクリプトの概要

次の項では，サーバ・トラフィックを分析するスクリプトの作成プロセスを説明します。

1 キャプチャ・ファイルを作成します。

VuGen では，キャプチャ・ファイルを使用してサーバ・トラフィックを分析し，エミュレートします。詳細については，83 ページ「キャプチャ・ファイルの生成」を参照してください。

2 新しい Web サービス・スクリプトを作成します。

VuGen のメイン・インタフェースを使用して，新しい Web サービス・スクリプトを作成します。詳細については，第 4 章「Web サービス－スクリプト内容の追加」を参照してください。

3 サービスを指定します（任意ですが推奨）。

高レベルのスクリプトを作成するには，テストの対象となる Web サービスが記述されている WSDL をインポートします。

4 トラフィック情報を指定します。

［**トラフィックの分析**］ボタンをクリックします。トラフィック・ファイルの場所と，スクリプトが受信トラフィック向けか送信トラフィック向けかを指定します。詳細については，87 ページ「トラフィック情報の指定」を参照してください。

5 トラフィック・フィルタの記録オプションを指定します。

フィルタ・オプションを使用することで，スクリプトに含めるホストと除外するホストを指定できます。詳細については，88 ページ「受信フィルタまたは送信フィルタの選択」を参照してください。

6 SSL 証明書情報を指定します。

SSL を設定することで，スクリプトを生成するために HTTPS 経由のセキュアなトラフィックを分析できます。詳細については，90 ページ「SSL 証明書の指定」を参照してください。

キャプチャ・ファイルの生成

キャプチャ・ファイルは，ネットワークを通過したすべての TCP トラフィックのログを含んだトレース・ファイルです。スニッファ・アプリケーションを使用して，すべてのネットワーク・トラフィックのダンプを取得します。スニッファはネットワーク上のすべてのイベントをキャプチャし，キャプチャ・ファイルに保存します。

より小さく，処理しやすいスクリプトを生成するには，アプリケーションでアクションを実行する間にのみネットワーク・トラフィックをキャプチャするようにします。

注：キャプチャ・ファイルにはループバック・ネットワーク・トラフィックは含まれません。

キャプチャ・ファイルを取得するには，コマンド・ライン・ユーティリティまたは既存のキャプチャ・ツールを使用します。

キャプチャ・ファイル・コマンド・ライン・ユーティリティ

VuGen のコマンド・ライン・ユーティリティ **lrtcpdump** は，製品の **bin** ディレクトリにあります。プラットフォームごとに次の個別のユーティリティがあります：**lrtcpdump.exe**（Windows），**lrtcpdump.hp9**，**lrtcpdump.ibm**，**lrtcpdump.linux**，および **lrtcpdump.solv4**。

キャプチャ・ツールを呼び出すには，次のように入力します。

lrtcpdump -i ＜**インタフェース**＞ -f ＜**ファイル**＞

ここで＜**インタフェース**＞は，トラフィックをキャプチャするネットワーク・カードの名前です。＜**ファイル**＞は，情報を格納するキャプチャ・ファイルの名前です。コマンド・ライン・オプション（**i** または **f**）と値の間にスペースを置かないでください。

Windows プラットフォームでキャプチャ・ファイルを作成するには，次の手順を実行します。

1 ［**スタート**］＞［**実行**］をクリックします。**cmd** と入力して［**OK**］をクリックし，コマンド・ウィンドウを開きます。

2 製品の **bin** ディレクトリにある **lrtcpdump.exe** プログラムをドラッグしてドロップするか，そのフル・パスを入力します。

3 次の構文を使用して，キャプチャ・ファイルのファイル名を渡します。
lrtcpdump -f ＜**ファイル**＞

例：**lrtcpdump -fmydump.cap**

4 **lrtcpdump** により，ネットワーク・カードの選択を求められます。インタフェース・カードが複数ある場合は，それらが一覧表示されます。インタフェース・カードの番号（1，2，3 など）を入力し，**Enter** キーを押します。

5 アプリケーション内で，標準的な操作を実行します。

6 コマンドライン・ウィンドウに戻り，**Enter** キーを押してキャプチャ・セッションを終了します。

Unix プラットフォームでキャプチャ・ファイルを作成するには，次の手順を実行します。

1 製品の **bin** ディレクトリで，プラットフォームに適した **lrtcpdump** ユーティリティを探します。該当する lrtcpdump ユーティリティを UNIX マシンからアクセスできるフォルダにコピーします。たとえば HP プラットフォームの場合，**lrtcpdump.hp9** をコピーします。FTP を使用する場合，必ずバイナリ転送モードを使用してください。

2 ユーティリティを実行するために，root ユーザに切り替えます。

3 実行パーミッションを次のように与えます。**chmod 755 lrtcpdump.** ＜**プラットフォーム**＞

4 UNIX プラットフォームでは，インタフェース・カードが複数ある場合，**lrtcpdump** はアルファベット順で最初のものを使用します。全インタフェースの一覧を取得するには，**ifconfig** コマンドを使用します。

5 インタフェースとファイル名を指定し，完全な構文でユーティリティを実行します。
例：**lrtcpdump.hp9 -ietho -fmydump.cap**
ネットワーク・トラフィックのキャプチャが始まります。

6 アプリケーション内で，標準的な操作を実行します。

7 **lrtcpdump** を実行しているウィンドウに戻り，画面の指示に従ってキャプチャ・セッションを終了します。

8 VuGen を実行するマシンからアクセスできるネットワーク上の場所にキャプチャ・ファイルを格納します。

既存のキャプチャ・ツール

ほとんどの UNIX オペレーティング・システムは，キャプチャ・ツールを備えています。ほかにも，Ethereal/tcpdump などダウンロード可能なキャプチャ・ツールが数多くあります。

外部ツールを使用する場合，すべてのパケット・データがキャプチャされ，切り捨てられているものがないことを確認します。

注：一部のユーティリティには引数を追加する必要があります。たとえば，**tcpdump** には，データを切り捨てずにパケットをキャプチャするために引数 **-s 0** が必要です。

サーバ・トラフィックに基づく基本スクリプトの作成

記録して作成するスクリプトと同じように，サーバ・トラフィックからスクリプトを作成します。

スクリプトに Web サービスを任意で指定することもできます。サービスを指定すると，VuGen によって **web_service_call** 関数を含んだスクリプトが作成されます。サービスを指定しないと，VuGen によって **soap_request** 関数を含んだスクリプトが作成されます。

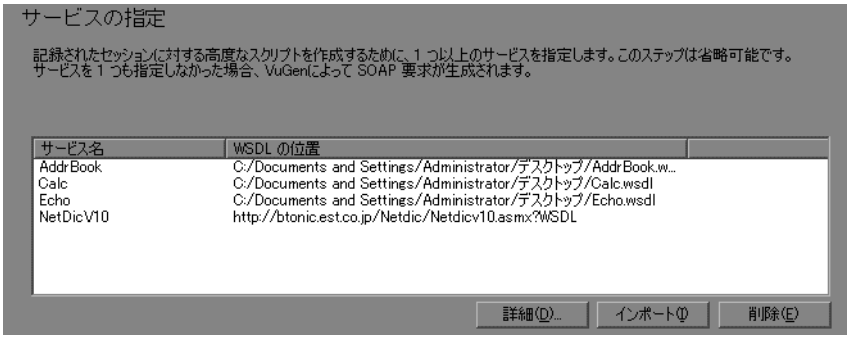
サーバ・トラフィック・スクリプトを作成するには，次の手順を実行します。

1 ［**ファイル**］＞［**新規作成**］を選択し，左側の表示枠で［**新規シングル プロトコル スクリプト**］をクリックします。

2 ［**Web サービス**］プロトコルを選択して［**OK**］をクリックします。

3 ［**トラフィックの分析**］ボタンをクリックするか，［**仮想ユーザ**］＞［**トラフィックの分析**］を選択します。ウィザードが起動します。

4 リストに 1 つ以上のサービスを追加します。この手順は任意です。



- ➤ 新規サービスを追加するには，［**インポート**］をクリックします。［サービスのインポート］ダイアログ・ボックスで，WSDL の場所を指定します。URL，ファイル，UDDI サーバ（Systinet など），または Quality Center 内の場所を指定できます。［サービスのインポート］ダイアログ・ボックスで，サービスを分析するためのツールキットも選択します。選択したツールキットはスクリプトに永続的に関連付けられます。変更はできません。詳細については，47 ページ「サービスのインポート」を参照してください。
- ➤ サービスの設定または詳細の表示を行うには，［**詳細**］をクリックして［サービスの管理］ウィンドウを開きます。サービス管理の詳細については，第 3 章「Web サービス－管理」を参照してください。
- ➤ リストに表示されているサービスを削除するには，対象サービスを選択して［**削除**］をクリックします。

5 ウィザード画面の最下部にある［**次**］をクリックして，トラフィック・ファイルの情報を指定します。詳細については，次項を参照してください。

6 トラフィックの情報を入力して［**完了**］をクリックすると，スクリプトが生成されます。

トラフィック情報の指定

トラフィック・ファイルにはすべてのネットワーク・トラフィックのダンプが含まれています。ウィザードを使用して，トラフィック・ファイルの場所と，スクリプトが受信トラフィックと送信トラフィックのどちらをエミュレートするのかを指定します。

- ［**キャプチャ ファイル**］：トラフィック・ファイルの名前とパスです。通常，拡張子は .cap です。
- ［**受信トラフィック**］の［**サーバ**］と［**ポート**］：受信トラフィックを調査する対象となるサーバの IP アドレスとポートです。
- ［**送信トラフィック**］の［**サーバ**］：送信トラフィックを調査する対象となるサーバの IP アドレスです。
- ［**記録先アクション**］：スクリプトを作成して挿入する対象となるセクションです。反復を使用する場合は，**Actions** セクションを指定します。
- ［**フィルタ オプション**］：スクリプトに含める IP アドレスまたは除外する IP アドレスを指定できるフィルタ・インタフェースが開きます。詳細については，次項を参照してください。
- ［**SSL 設定**］：SSL 証明書を追加して，必要な資格情報を使用してセキュア・サーバからのトラフィックを分析できるようにします。詳細については，90 ページ「SSL 証明書の指定」を参照してください。

受信フィルタまたは送信フィルタの選択

サーバの IP アドレスとポートを指定することで，サーバが受信する要求またはサーバが送信する要求を絞り込むフィルタを提供できます。

また，キャプチャ・ファイルを VuGen にロードする前に，外部ツールでフィルタ処理できます。この場合，追加のフィルタリングは必要ありません。

要求にフィルタを適用するには，関連するホストの IP アドレスを選択します。包含フィルタまたは除外フィルタを適用できます。つまり，リスト内の IP アドレスのみを含めることも，リスト内の IP アドレスを除くすべての IP アドレスを含めることもできます。

トラフィック・ファイルにフィルタを適用するには，次の手順を実行します。

1 トラフィック・フィルタの記録オプションを開きます。［トラフィック情報を指定してください］ステップで［**フィルタ オプション**］をクリックするか，［**ツール**］＞［**記録オプション**］を選択します。［**トラフィック分析**：**トラフィック フィルタ**］ノードを選択します。

2［**リスト内の全 IP アドレスを含める**］または［**リスト内の全 IP アドレスを除外する**］のいずれかのフィルタ・オプションを選択します。

3 スクリプトのタイプ（**受信トラフィック**または**送信トラフィック**）に対応するタブを選択します。

4 ホストをリストに追加します。

このリストにホストを追加するには，［**追加**］ボタンをクリックします。リストに追加するサーバの IP アドレスを指定します。受信トラフィックの場合，含めるかまたは除外するサーバのポートを指定します。［**OK**］をクリックして設定を受け入れます。

エントリを削除するには，［**削除**］をクリックします。

スクリプトが作成されたら，フィルタを変更してスクリプトを再生成できます。キャプチャ・ファイルを分析しなおす必要はありません。

SSL 証明書の指定

セキュア・サーバからのトラフィックを分析するには，サーバの秘密鍵を含む証明書を指定する必要があります。

トラフィックが SSL により暗号化されている場合，復号するために証明書ファイルとパスワードを指定する必要があります。複数のサーバからのトラフィックをスクリプトに反映する場合は，SSL を使用する IP アドレスごとに個別の証明書とパスワードを指定する必要があります。

SSL 証明書を指定するには，次の手順を実行します。

1 ［トラフィック情報を指定してください］画面で，［**SSL 設定**］をクリックします。

2 証明書をリストに追加します。

このリストに証明書を追加するには，［**追加**］ボタンをクリックします。IP アドレス，ポート，証明書ファイル（拡張子は **.pem**）のパス，および証明書のパスワードを指定します。**pem** ファイルに秘密鍵が含まれていることを確認します。証明書の取得方法がわからない場合は，システム管理者に問い合わせてください。

リストからエントリを削除するには，［**削除**］ボタンをクリックします。

3 追加する証明書ごとに，前述の手順を繰り返します。

4 ［**OK**］をクリックし，ダイアログ・ボックスを閉じます。

第 6 章

Web サービス呼び出しのプロパティ

Web サービス呼び出しビューを使用して，スナップショットを表示したり，プロパティを設定したり，Web サービス呼び出しにチェックポイントを追加します。

本章の内容

以下の情報は，**Web** サービスと **SOA Vuser** スクリプトのみを対象とします。

Web サービス呼び出しビューについて

Web サービス呼び出しビューを使用すると，スナップショットを表示したり，プロパティを設定したり，スクリプトの各 Web サービス呼び出しにチェックポイントを定義できます。

Web サービス呼び出しビューを開くには，ツリー・ビューを表示している必要があります。［VuGen］ウィンドウで［**表示**］＞［**ツリー ビュー**］を選択して，ツリー・ビューを開きます。

スナップショットによって，各ステップの SOAP 構造を表示できます。記録，再生，要求，および応答のスナップショットを表示できます。詳細については，次を参照してください。

［プロパティ］タブでは，Web サービス呼び出しの設定を行います。引数値，パラメータ化，トランスポート層，およびセキュリティに関してプロパティを設定できます。詳細については，95 ページ「Web サービス呼び出しのプロパティについて」を参照してください。

チェックポイントでは，Web サービスの結果を検証できます。期待値をチェックし，出力を表示して，それらが一致しているか確認できます。詳細については，124 ページ「チェックポイント」を参照してください。

Web サービスの SOAP スナップショットの表示

VuGen のスナップショット・ビューアを使用して，記録中または再生中に発生した SOAP 要求および SOAP 応答を調べることができます。再生スナップショットを表示するには，少なくとも 1 回はセッションを再生する必要があります。

SOAP スナップショットを表示する方法はいくつかあります。

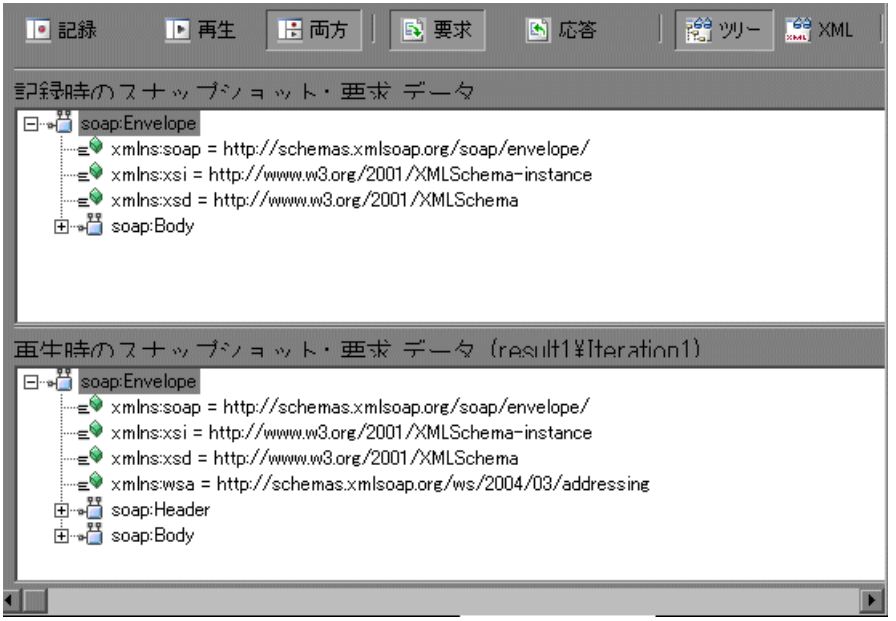
- 記録または再生，あるいはその両方
- 要求データまたは応答データ
- ツリー・ビューまたは XML ビュー

［スナップショット］ウィンドウのボタンを使用してビューを制御します。

ツリー・ビューでは，ノードを展開して引数の値を表示できます（割り当てられている場合）。

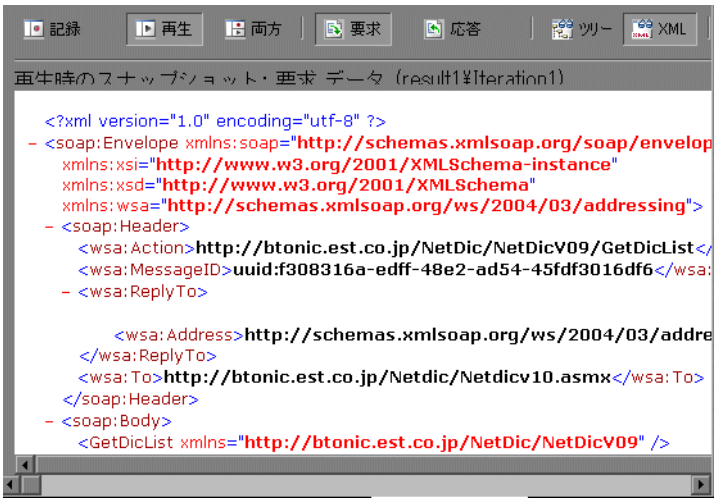
要求ビューには，Web サービス・セッション中にサーバに送られた値が表示されます。**応答**ビューには，サーバから返された結果の値が表示されます。

次の例では，［スナップショット］ウィンドウに，**要求**データの**記録**および**再生**スナップショットが**ツリー・**ビューで表示されています。



ノードの詳細については，ノードを選択し，右クリック・メニューから［**ノードのプロパティ**］を選択します。

XML ビューでは，SOAP メッセージ全体を XML 形式で表示できます。



再生反復の指定

複数の反復があるスクリプトを再生した場合は，スナップショットに表示する反復を指定できます。また，スクリプト・フォルダ以外の場所に保存したテスト結果からもスナップショットを表示できます。標準設定では，スナップショット・ビューには最後の反復が表示されます。

表示する反復を指定するには，次の手順を実行します。

1 ［**表示**］＞［**スナップショット**］＞［**反復の選択**］を選択します。

2 目的とする反復を選択して，［**OK**］をクリックします。

3 別のフォルダから結果を表示するには，［**フォルダを選択**］を選択して，テスト結果の場所を参照します。

Web サービス呼び出しのプロパティについて

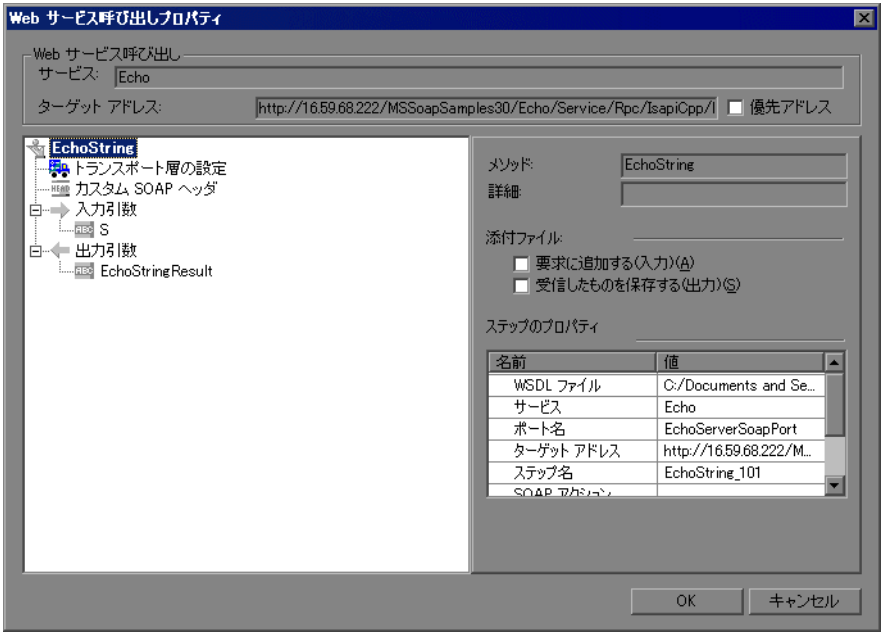
VuGen は，各 Web サービス呼び出しのプロパティを表示して変更するインタフェースを提供します。

プロパティはサービス内で各メソッドの動作を記述します。サービスの各メソッドごとに宛先アドレス，引数値，パラメータ化，およびトランスポート層オプションを設定できます。

次の方法のいずれかで，ツリー・ビュー（[**表示**] > [**ツリー ビュー**]）からステップのプロパティを表示できます。

- 左の表示枠でステップをダブルクリックして，[Web サービス呼び出しのプロパティ] ダイアログ・ボックスを開きます。
- 右の表示枠で [**ステップのプロパティ**] タブを選択します。

プロパティ・ビューには，各サービスの操作がツリー階層で表示されます。ツリーのノードは，トランスポート層の設定，SOAP ヘッダ，入力引数，および出力引数を表します。



標準設定では，スクリプトによって WSDL から宛先アドレスが取り出されます。各操作ごとに，このアドレスをオーバーライドできます。［**優先アドレス**］オプションを選択し，目的とする［**ターゲット アドレス**］を指定します。

選択したツリー・ノードのレベルに応じて，右側の表示枠の内容が変わります。次の表に，各ノードの内容を示します。

選択対象 ...	**右側の表示枠の表示**
メソッド名または操作名	➤ 入出力添付ファイルを含めるチェック・ボックス ➤ ステップのプロパティ
［トランスポート層の設定］	詳細トランスポート・オプション： ➤ **非同期**または **WSA** サポートを使用した **HTTP/S トランスポート** ➤ 応答/要求キュー情報を使用した **JMS トランスポート**・サポート
［SOAP ヘッダ］	現在の要素に SOAP ヘッダの値を指定するエディット・ボックス。詳細については，118 ページ「SOAP ヘッダ」を参照してください。
入力引数ノード	➤ 各メソッドの［**名前**］とその［**値**］。 ➤［**全て含む**］，［**リセット**］，および［**生成**］ボタン。
引数単体	➤［**名前**］：引数の名前（読み取り専用）。 ➤［**種類**］：WSDL で定義された引数タイプ。WSDL に派生型が含まれていると，このボックスはドロップ・ダウン・リストになります。詳細については，105 ページ「派生型」を参照してください。 ➤［**呼び出しに引数を含める**］：オプション・パラメータを Web サービス呼び出しに含めます。詳細については，106 ページ「オプション・パラメータを使った作業」を参照してください。 ➤［**なし**］：Nillable 属性を True に設定します。 ➤［**値**］：引数の値です。base64 バイナリ・タイプの引数としては，**Get from file** または **Base64 Encoded text** です。 ➤［**この引数の自動値を生成**］：このノードの自動値を挿入します。

複雑な入力引数	➤［**XML**］:［**編集**］ボタン，［**インポート**］ボタン，および［**エクスポート**］ボタンを有効にします。XML を編集することで，引数値を手作業で挿入できます。XML 構造全体を 1 つの XML 型パラメータで置き換えるには，［**ABC**］アイコンをクリックします。 **注**：このインポート操作では，標準の SOAP ファイルではなく，以前エクスポートした XML ファイルを処理します。SOAP をインポートするには，71 ページ「SOAP 要求のインポート」を参照してください。 ➤［**この引数の自動値を生成**］：この複合型ノードのすべての引数に対して，自動値を挿入します。 ➤［**追加**］/［**削除**］：配列に要素を追加，または配列から要素を削除します。
［出力引数］	➤ 出力引数リスト ➤ 異常系テスト・オプション，第 13 章「Web サービス ー 異常系テスト」を参照。
［入力添付ファイル］	SOAP 要求をエンコードする［**添付ファイル形式**］：VuGen ツールキットの場合は DIME または MIME（224 ページ「MIME 添付ファイルの包含」を参照），.Net の場合は DIME のみ，Axis の場合は MIME のみです。

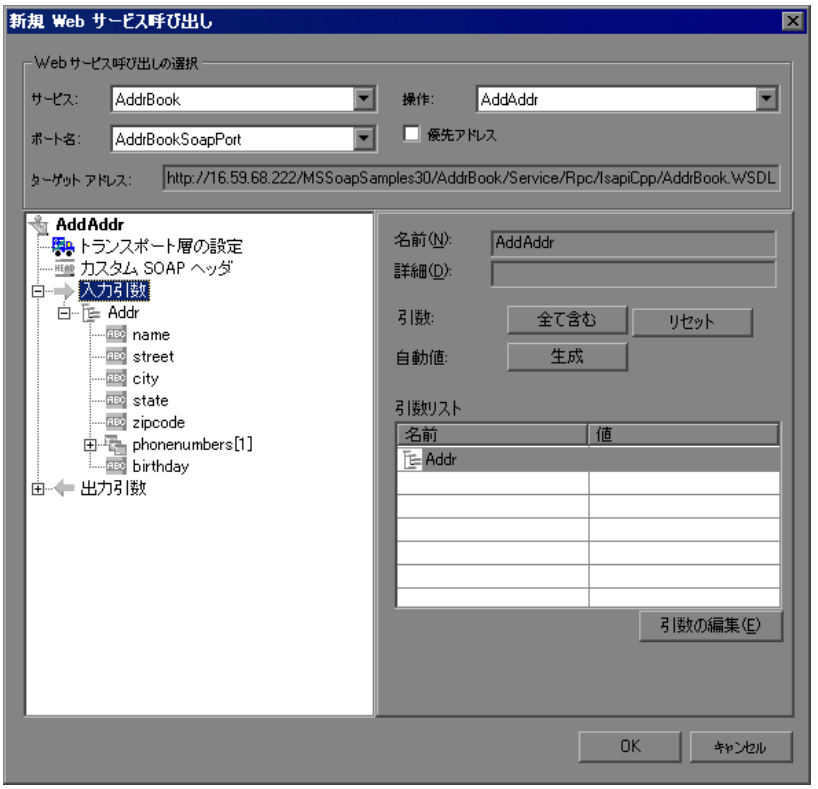
［添付］	➤［**データ取得先**］：添付するファイルの名前，またはデータが含まれるパラメータの名前。 ➤［**Content Type**］：添付ファイルのコンテント・タイプです。タイプを自動的に検出するように VuGen に指示するか，ドロップダウン・リストからタイプを選択するか，値を手作業で入力します。 ➤［**Content ID**］：添付ファイルの ID 属性です。値を自動的に生成するように VuGen を指示するか，独自の ID を指定します。 ➤［**添付ファイルを削除**］：添付ファイルを削除するボタンです。
出力添付ファイル	➤［**すべての添付ファイルを保存する**］：すべての添付ファイルをパラメータに保存します。 ➤［**内容**］：添付ファイルを格納するパラメータのプレフィックスの名前。 ➤［**Content Type**］：添付ファイルのコンテント・タイプ属性（読み取り専用）。 ➤［**Content ID**］：添付ファイルの ID 属性（読み取り専用）。 ➤［**添付ファイルをインデックスとして保存する**］：1 から始まる番号に基づいて添付ファイルを保存します。追加の添付ファイル・インデックスを挿入するには，［**追加**］をクリックします。

次の要素の引数値を設定できます （手作業で編集した引数は青で表示されます）。

- 入力引数値
- 出力引数
- 配列
- 添付ファイル
- SOAP ヘッダ

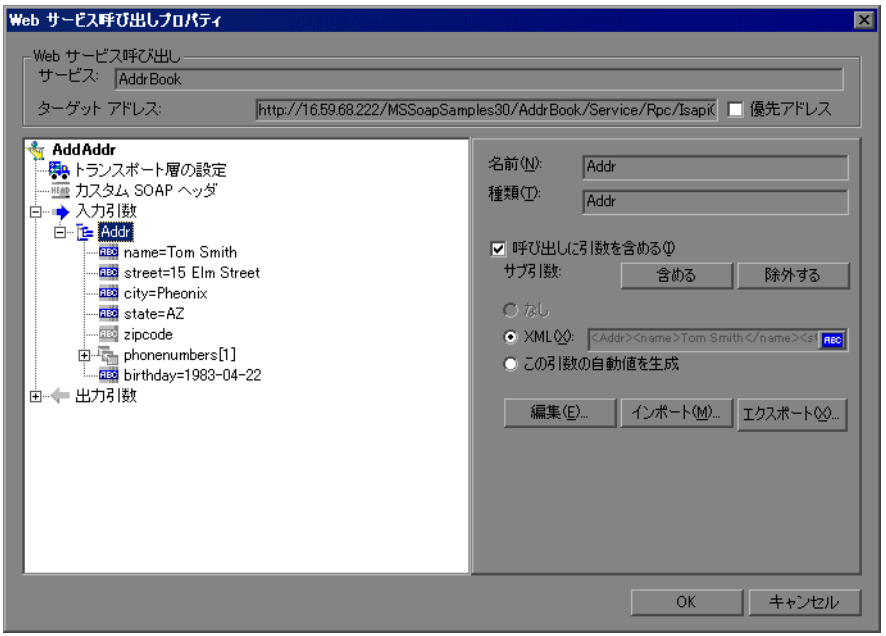
入力引数値

［入力引数］ノードでは，入力引数のすべての値を定義でき，オプション要素を制御できます。オプション要素の詳細については，106 ページ「オプション・パラメータを使った作業」を参照してください。



- ［**全て含む**］: すべてのオプション要素を含めます。操作の要素がすべて含まれます。
- ［**リセット**］: すべてのオプション要素を除外し，必須の要素のみを含めます。
- ［**生成**］: すべてのオプション要素を含め，すべての操作要素の自動値を生成します。
- ［**引数の編集**］: 選択した引数のノードを開き，その値を設定できます。

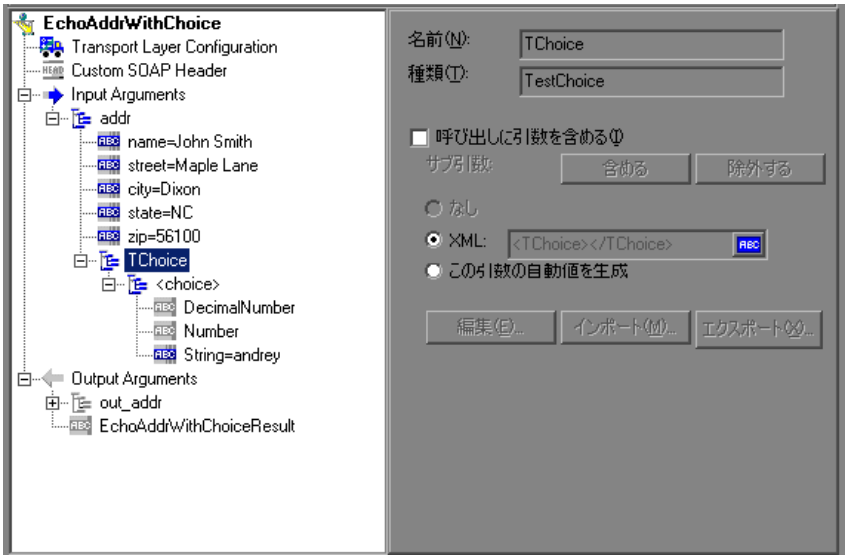
個々の引数ノードでは，各入力引数の値を定義できます。



- ［**XML/ 値**］：ノードに手作業で指定した値。引数が配列の場合は，XML 構造を指定できます。配列でなければ，通常の値を指定します。引数に対するパラメータを作成するには，［**XML/ 値**］ボックスの右隅にある［**ABC**］アイコンをクリックして［パラメータの選択または作成］ダイアログ・ボックスを開きます。
- ［**この引数の自動値を生成**］：この引数の値を VuGen に自動生成させる場合は，このオプションを選択するか，ツリー階層で引数を選択して，右クリック・メニューから［**Generate Auto-values**］を選択します。

Choice 要素

WSDL で Choice 要素を定義すると，それらを表示して，プロパティ・ビューでその値を設定できます。



choice 要素の値を設定するには，親要素を選択し，［**呼び出しに引数を含める**］オプションを有効にして，値を設定します。

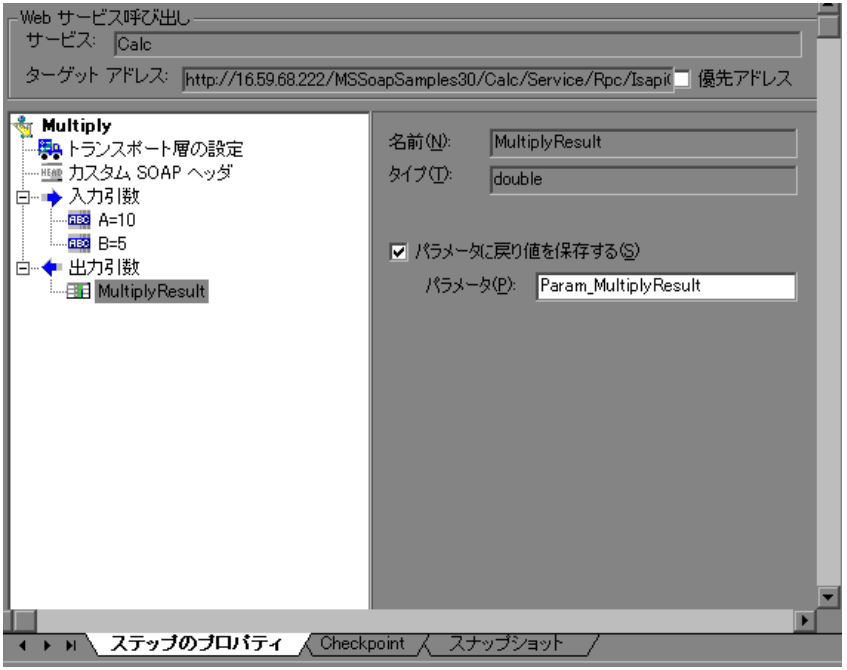
引数をパラメータ化するには，［**ABC**］アイコンをクリックします。［パラメータのプロパティ］ダイアログ・ボックスで，choice 引数に値を設定します。choice 要素の 1 つに値を設定すれば十分です。複数の反復を実行する場合は，割り当て方法（順次，一意，またはランダム）に従って，スクリプトで同じ choice 要素の値を使用します。たとえば，choice 要素が **Decimal Number**，**String**，および **Number** であり，**Number** に値を設定した場合，Vuser は必ず **Number** 要素を使用します。

Choice のサポートは，入出力引数，パラメータ化，チェックポイント，およびサービス・エミュレーションに用意されています。

オプション引数の使用については，106 ページ「オプション・パラメータを使った作業」を参照してください。

出力引数

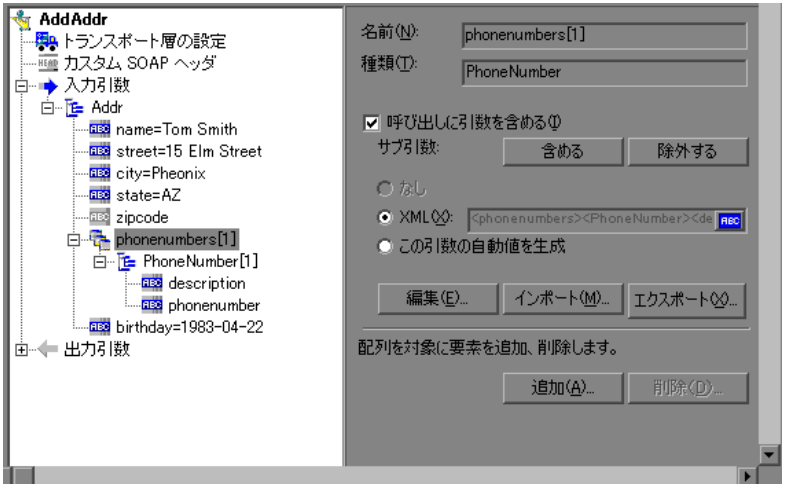
出力引数値を表示したり，出力引数値をパラメータまたは配列に保存したりできます。



➤［**パラメータに戻り値を保存する**］：テキスト・ボックスに指定した名前のパラメータに戻り値を保存します。

配列

入力引数または出力引数に配列を使用するには，左側の表示枠でその配列を選択します。

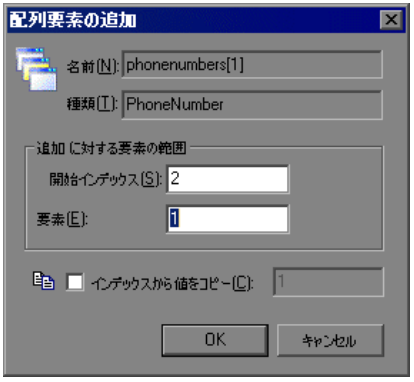


- ［**XML**］：配列要素の値が含まれる XML ファイルのパス。XML を XML 型パラメータで置き換えるには，［**ABC**］アイコンをクリックします。XML パラメータ化は入力引数として配列をサポートしています。XML パラメータには，必要に応じて配列要素の数を定義します。パラメータに配列を保存すると，パラメータごとの配列要素の数が一定になります。各反復で異なる数の配列要素を使用して，複数の反復を実行する場合は，それぞれの反復に目的とする数の配列要素を含めて，別々のパラメータを定義する必要があります。XML パラメータの詳細については，『**第 1 巻 ― VuGen の使用**』の「XML パラメータのプロパティの設定」を参照してください。
- ［**編集**］/［**インポート**］/［**エクスポート**］：複合型および配列を変更するには，要素および引数を選択して［**編集**］をクリックします。［**エクスポート**］をクリックして，選択したエントリを別の XML ファイルにエクスポートするか，［**インポート**］をクリックして，以前エクスポートした XML ファイルをロードします。**注**：この［**インポート**］操作では，標準の SOAP ファイルではなく，以前エクスポートした XML ファイルを処理します。SOAP をインポートするには，71 ページ「SOAP 要求のインポート」を参照してください。

- ［**追加**］：［配列要素の追加］ダイアログ・ボックスが開き，単純型または複合型の配列要素を追加できます。
- ［**削除**］：配列要素を削除します。削除する要素の開始添字と個数を指定します。

配列要素の追加

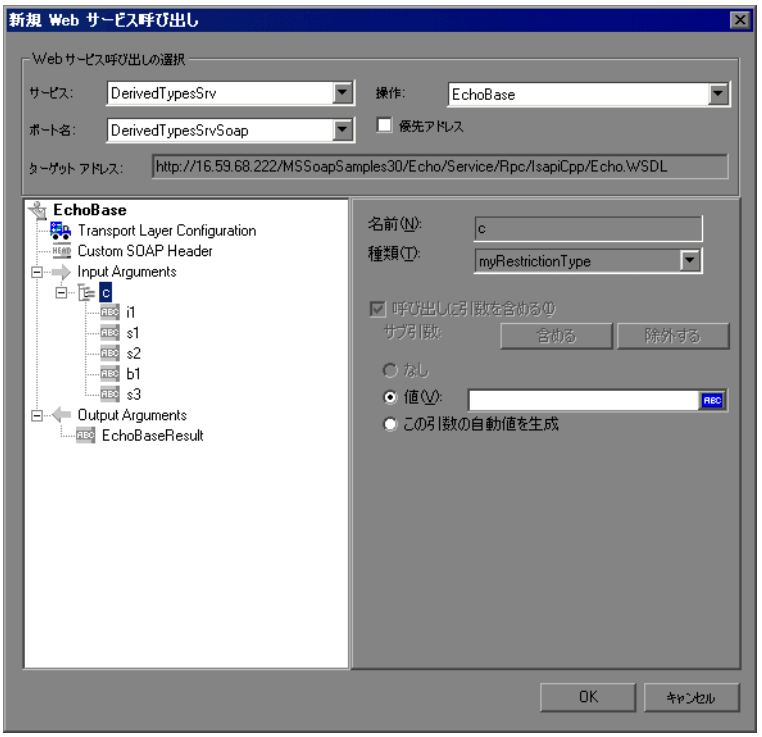
配列要素セクションの［**追加**］をクリックすると，次のように［配列要素の追加］ダイアログ・ボックスが開きます。



- ［**開始インデックス**］：追加される最初の要素の添字。
- ［**要素**］：追加する要素の個数。
- ［**インデックスから値をコピー**］：既存の配列要素の値を新しい要素に割り当てます。使用する値を持つ要素の配列添字を指定します。

派生型

VuGen では，派生型を使用した WSDL をサポートしています。Web サービス呼び出しにプロパティを設定すると，基底型または派生型を使用する引数を設定できます。



目的のタイプを選択すると，VuGen によって新しい型を反映して引数ツリー・ノードが更新されます。

抽象型

抽象型は，プログラマが宣言する宣言型です。要素または型を**抽象型**として宣言すると，インスタンス・ドキュメントでは使用できません。代わりに，要素の置換グループの数を，XML スキーマで設定して，インスタンス・ドキュメントに表示する必要があります。このような場合には，当該要素のすべてのインスタンスで，**xsi:type** を使用して，抽象型ではない派生型を指示する必要があります。

VuGen が抽象型に直面しても，抽象型クラスを作成できないため，再生に失敗します。この場合，VuGen は［**型**］ボックスの下に警告メッセージを表示して，抽象型を派生型に置き換えるよう指示します。

オプション・パラメータを使った作業

WSDL ファイルにオプション・パラメータが含まれている場合は，それらを SOAP 要求に含めるかどうかを指示できます。

WSDL ファイルでは，オプション・パラメータを次の属性のいずれかで定義します。

minoccurs='0'

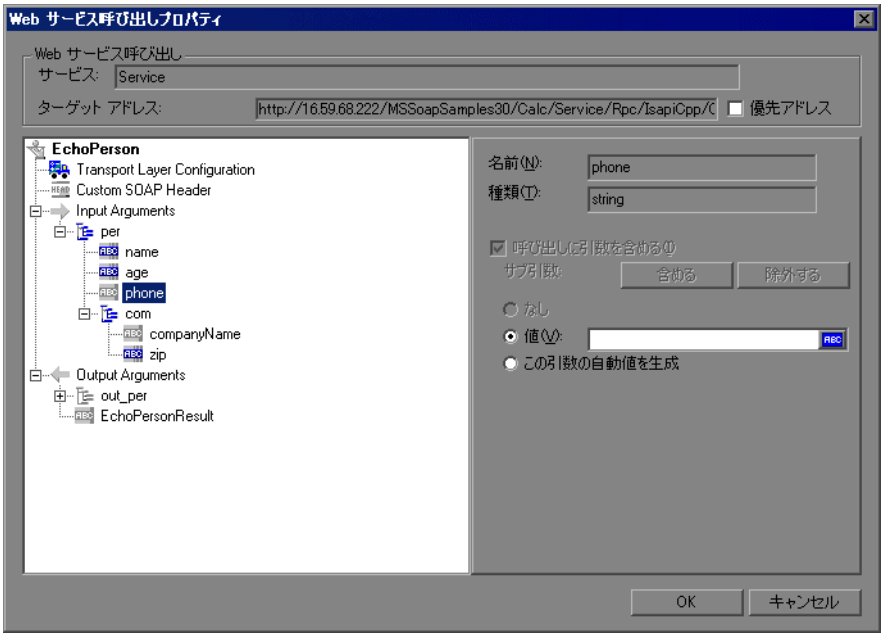
nillable='true'

minoccurs = 0 は省略可能なオプション要素であることを明示します。Nillable は，nillable 属性を真または 1 に設定すると，通常の内容がなくても，要素が存在できることを意味します。標準設定では，**minoccurs** および **maxoccurs** 属性は 1 に設定されます。

次の例では，**name** が必須，**age** がオプション，**phone** が nillable です。

```
<s:element minOccurs="1" name="name" type="s:string" />
<s:element minOccurs="0" name="age" type="s:int" />
<s:element minOccurs="1" name="phone" nillable="true" type="s:string" />
```

サービス呼び出しに引数値を設定すると，VuGen がオプションを有効 / 無効にして要素の型を指示します。



次の表に，使用可能なオプションを示します。

パラメータのタイプ	[なし]（Nil）ラジオ・ボタン	[呼び出しに引数を含める]
必須	無効	無効
MinOccurs=0	無効	有効
Nillable	有効	無効

サービス呼び出しに特定のオプション引数を含めるには，ノードをクリックして，［**呼び出しに引数を含める**］を選択します。含まれているすべての引数のノードは青になります。含まれていない引数はグレーになります。

親レベルの要素を含めると，その下にある必須および nillable の子要素がすべて自動的に含まれます。子要素であれば，親要素およびそのレベルにあるほかの必須または nillable 要素がすべて自動的に含まれます。親要素に［**Generate auto-value**］を指定すると，VuGen は，親の下に含まれている子要素に値を提供します。

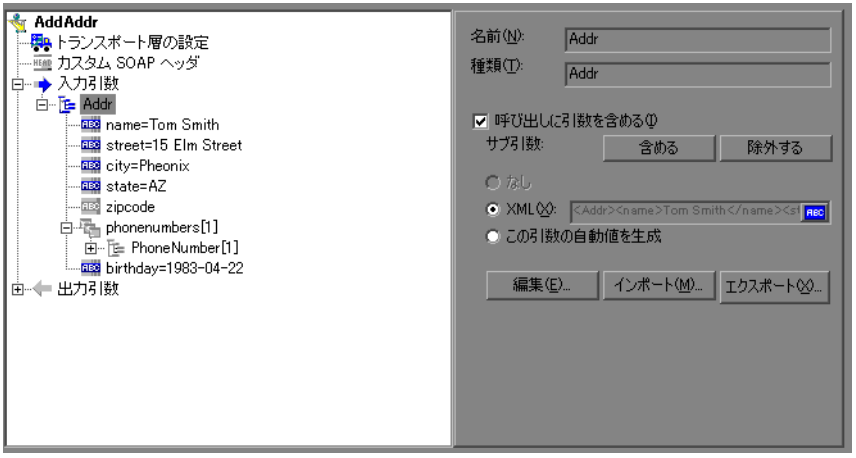
注：VuGen は，ツールキットの実装によって，要素が必須であるかオプションであるか解釈します。これは WSDL ファイルでの要素の属性と必ずしも一致しません。

サブ要素を含めるには，次の手順を実行します。

1 特定のサブ要素を含めるには，左側の表示枠で選択して，［**呼び出しに引数を含める**］オプションを選択します。

2 親要素のサブ要素をすべて含めるには，［**呼び出しに引数を含める**］を親要素に適用し，その下にある［**含める**］ボタンをクリックします。

3 すべてのサブ要素を除外するには，親要素を選択して，［**除外する**］ボタンをクリックします。省略可能な引数がすべて除外されます。

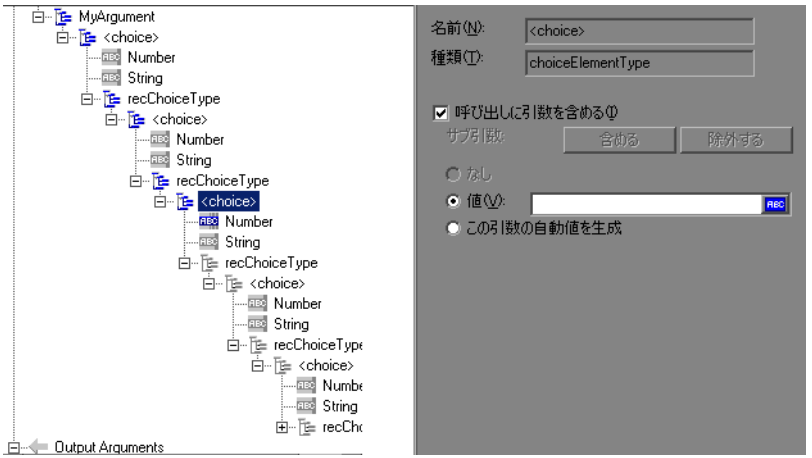


繰り返し要素

［プロパティ］ダイアログ・ボックスを使うと，Web サービス呼び出しに含める繰り返し要素のレベルを制御できます。

特定のレベル以下を除外するには，含めたい最低の親ノードを強調表示して，［**呼び出しに引数を含める**］を選択します。VuGen によって，選択したノード，その必須の子，その親ノードのすべてが含まれます。

次の例では，Choice 引数の 3 つのレベルが含まれ，残りは含まれていません。含まれていないノードはグレー表示になります。



Choice オプション要素

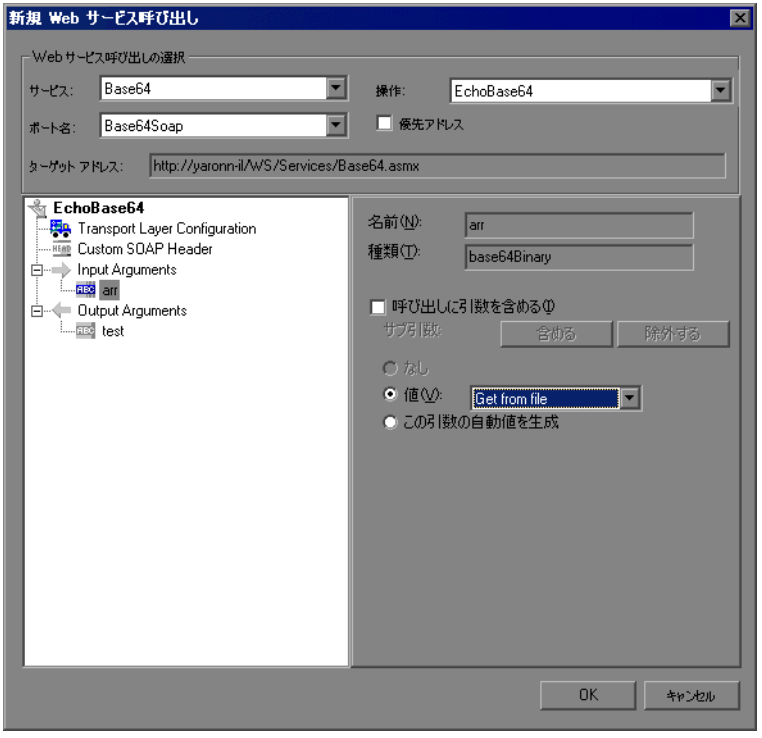
WSDL の Choice 要素では一連の要素を定義し，その 1 つだけが SOAP メッセージに表示されます。場合によっては，Choice 要素の 1 つがオプションで，その他はオプションではありません。**Service Test** では，Choice 要素を選択して，そのオプション要素が SOAP エンベロープに表示されないようにできます。ツリー・ビューでは，Choice 要素を選択して，［**呼び出しに引数を含める**］オプションをクリアします。スクリプト・ビューでは，Choice 引数を定義している行を削除します。

Base 64 エンコーディング

Base 64 エンコーディングは，バイナリ・データを ASCII テキストとして表示するのに使用するエンコーディング方式です。SOAP エンベロープはプレーンテキストであるため，このエンコーディングを使用すると，バイナリ・データを SOAP エンベロープ内でテキストとして表示できます。

VuGen が **base64Binary** タイプの WSDL 要素を検出すると，エンコードされた値を指定できます。値は次の 2 つの方法で指定できます。

- ［**Get from file**］：ファイル名を参照します。
- ［**Embed encoded text**］：エンコードするテキストを指定します。

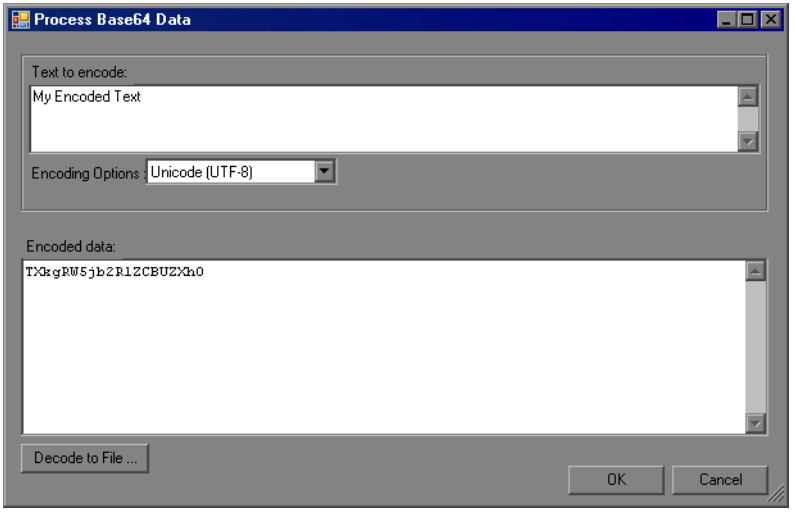


base64Binary 値を指定するには，次の手順を実行します。

1 ［**値**］オプションを選択します。

2 ファイルを指定するには，［**Get from file**］を選択し，下の［参照］ボタンを使ってファイルを見つけます。

3 テキストを指定するには，［**Embed encoded text**］オプションを選択し，下の［参照］ボタンをクリックします。［Process Base64 Data］ダイアログ・ボックスが開きます。



［**Text to encode**］ボックスにテキストを入力します。

標準設定の UTF-8 方式以外のエンコーディングを使用するには，［**Encoding Options**］リストから選択します。

［**Encode**］をクリックします。

4 ［**OK**］をクリックします。VuGen によって，Web サービス呼び出しがスクリプトに追加されます。これで，ステップとそのプロパティをツリー・ビューに表示できます。

ファイルから値を参照した場合は，Web サービス呼び出しにファイル名が含まれます。

```
"xml:arr="
    "<arr base64Mode=¥"file¥">C:¥¥Load_testing¥¥TEcho.xml</arr>",
```

［**Base 64 Encoding Text**］オプションを使って実際のテキストを挿入した場合は，スクリプトの Web サービス呼び出しにエンコードされたテキストが含まれます。

```
"xml:arr="
    "<arr base64Mode=¥"encoded¥">YWJjZGVmZw==</arr>",
```

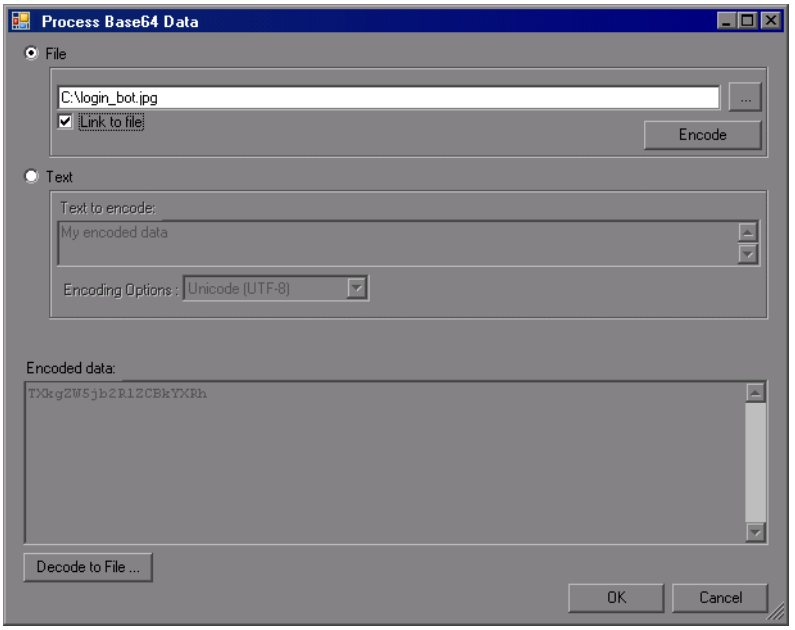
Base64Binary データのパラメータ値の設定

Base64 引数値をパラメータに設定するには，ファイル形式の新しいパラメータ，すなわち Complex 引数の XML タイプを作成します。詳細については，『**第 1 巻 ― VuGen の使用**』の「XML パラメータのプロパティの設定」を参照してください。

base64Binary 値が含まれた複合型引数の場合，VuGen では，パラメータ，チェックポイント，またはエミュレーション値を設定するために base64Binary を処理できます。値を指定するときは，**ファイル**から値を取得するか，手作業で**テキスト**を指定してエンコーディングを適用します。

ファイルから値を取得する場合は，次のオプションのいずれかを指定します。

- ［**Link to file**］：値が含まれているファイルを参照します。
- ［**Do not Link to a file**］：指定したファイルの内容を使用します。VuGen によって，その内容がスクリプト・フォルダにコピーされます。このオプションを使用するには，［**Link to file** ］チェック・ボックスをクリアします。



ヒント：スクリプトのパフォーマンスが向上するため，通常はファイルにリンクすることをお勧めします。テキストが 10KB を超える場合は，ファイルにリンクする必要があります。

［Process Base64］ダイアログ・ボックスを開くには，次の手順を実行します。

1 複合型引数の新しいパラメータを作成します。

2 グリッド・ビューで（XML パラメータまたはチェックポイント），値ボックスの右にある［参照］ボタンをクリックします。［Process Base64］ダイアログ・ボックスが開きます。

3 値のソースとして，［**ファイル**］または［**テキスト**］を指定します。

4 ソースとして［**ファイル**］を選択した場合は，ファイルにリンクするかどうかを指定します。

5 暗号化された値（再生中に取得した値など）をデコードするには，［**Decode to File**］をクリックします。詳細については，次を参照してください。

この項は，VuGen で引数値のグリッド・ビュー（パラメータ・リストとチェックポイント）を使用するすべての場所に適用されます。

ファイルへのデコード

VuGen では，エンコードされたテキストをファイルにデコードできます。これは特に，サーバから戻された base64 エンコード値（画像など）の正しさをチェックするのに役立ちます。

次の手順では，記録画像と再生画像が互いに一致するかチェックする方法について説明します。

デコードを使って画像を検証するには，次の手順を実行します。

1 新規の Web サービス呼び出しを作成します。

2 ［**Get from file**］オプションを使って，Base64 引数の値を設定します。画像ファイルを指定します。スクリプトの作成を続けます。

3 スクリプトを保存して再生します。

4 ［チェックポイント］タブに切り替え，再生値をロードします。

5 Base 64 引数の再生値をクリックし，そのプロパティを開きます。

6 ［**Decode to file**］をクリックします。ファイルに保存するファイル名を指定します。元のファイルと同じ拡張子を使用します。

7 デコードされた画像と元の画像を比較して，一致しているか確認します。

添付ファイル

画像などのバイナリ・ファイルを SOAP を介して送信する場合は，データを XML 形式にシリアル化する必要があります。しかし，シリアル化およびシリアル化解除は，非常に大きなオーバーヘッドを伴う可能性があります。そのため，大きなバイナリ・ファイルは，添付ファイル・メカニズムを使用して送信するのが一般的です。これにより，バイナリ・ファイルはそのままの状態で，解析のオーバーヘッドが減少します。

添付ファイルを使用すると，元のデータは SOAP エンベロープとは独立に送信されます。これによって，データを XML にシリアル化する必要がなくなり，データ転送がより効率的になります。

SOAP メッセージとともにバイナリ・データを送信するのに使用されるフォーマットは MIME（Multipurpose Internet Mail Extensions）仕様と，より効率的な新しいメカニズムである DIME（Direct Internet Message Encapsulation）仕様です。VuGen は，すべてのツールキットで DIME をサポートしていますが，Axis ツールキットでは MIME しかサポートしていません。.NET ツールキットで MIME 添付ファイルを使用するには，224 ページ「MIME 添付ファイルの包含」を参照してください。

VuGen では，SOAP メッセージと添付ファイルの送受信をサポートしています。入力（要求）添付ファイルの送信または出力（応答）添付ファイルの保存が行えます。

添付ファイルを追加または保存するには，左側の表示枠で，添付ファイルを関連付ける対象となる操作またはメソッドを選択します。入力添付ファイルおよび出力添付ファイルの両方を追加できます。

入力添付ファイル

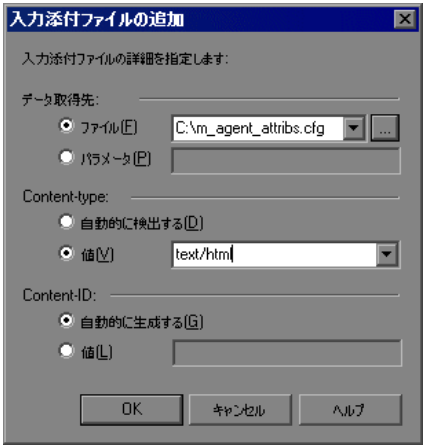
入力添付ファイルが要求メッセージに追加されます。

添付ファイルを要求に追加するには，次の手順を実行します。

1 添付ファイルを追加する対象となる操作を左側の表示枠で選択します。

2 右の表示枠で，［**要求に追加する（入力）**］を選択します。VuGen から添付ファイルに関する情報を入力するよう要求されます。添付ファイルがメソッドのツリー構造に追加されます。

［入力添付ファイルの追加］ダイアログ・ボックスが表示されます。



次の情報を指定します。

- ［**データ取得先**］: データの場所。バイナリ・データが含まれるファイルまたはパラメータを指定できます。
  - ［**ファイル**］: 次の 2 つの方法でファイルの場所を指定できます。
    - 絶対パス：ファイルのフル・パス。このファイルは，スクリプトを実行しているすべてのマシンからアクセス可能でなければなりません。
    - 相対パス（推奨）: ファイル名。この方法を使用した場合，VuGen は，再生時にスクリプトのフォルダ内で添付ファイルを検索します。添付ファイルをスクリプトのフォルダに追加するには，［**ファイル**］＞［**ファイルをスクリプトに追加**］を選択し，ファイル名を指定します。
  - ［**パラメータ**］: データが含まれているパラメータの名前を指定します。
- ［**Content-type**］: データが含まれているファイルの内容タイプ。［**自動的に検出する**］オプションを指定すると，VuGen によってコンテント・タイプが自動的に判断されます。また，［**値**］ボックスの一般的なコンテント・タイプのリストから text/html や image/gif，あるいはほかのコンテント・タイプを選択できます。
- ［**Content-ID**］: 内容の ID。標準設定では，VuGen によって自動的に生成され，添付ファイルの一意の識別子として使用されます。必要に応じて，［**値**］ボックスに別の ID を指定できます。

出力添付ファイル

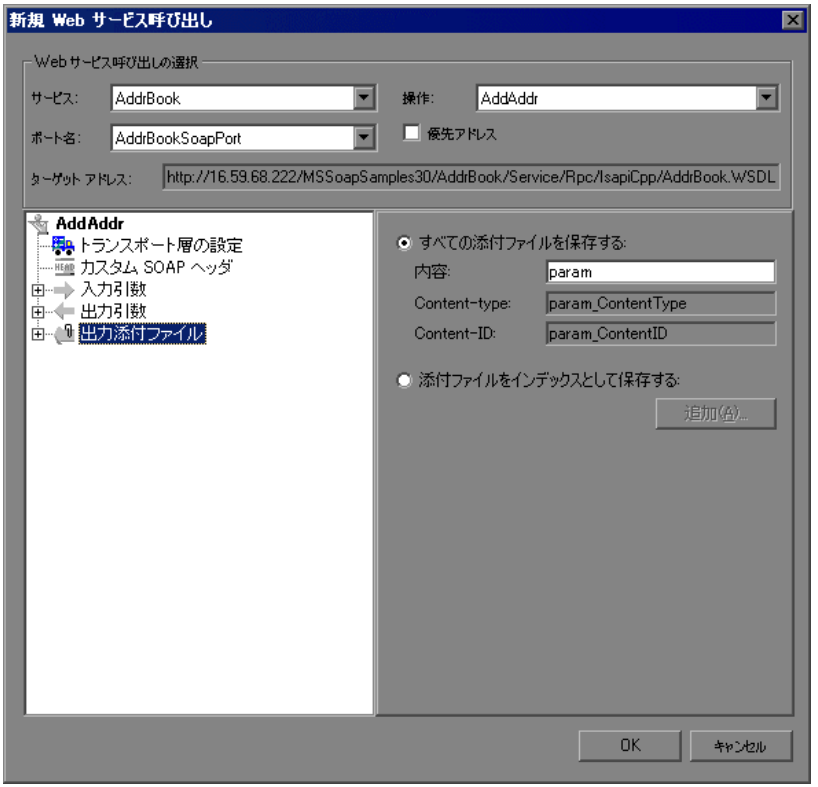
出力添付ファイルが応答メッセージに追加されます。

応答を添付ファイルとして保存するには，次の手順を実行します。

1 応答を保存する対象となる操作を左側の表示枠で選択します。

2 右の表示枠で，[**受信したものを保存する（出力）**] を選択します。VuGen によって，出力添付ファイル・ノードが左側の表示枠のメソッドのツリー構造に追加されます。

3 使用するオプションを選択します。[**すべての添付ファイルを保存する**]，または 1 から始まるインデックス番号に基づいて [**添付ファイルをインデックスとして保存する**] を選択します。



［**すべての添付ファイルを保存する**］を指定すると，VuGen により，指定したパラメータ名に基づいて各添付ファイルに次の 3 つのパラメータが作成されます。添付ファイル・データを含んでいるパラメータ，添付ファイルのコンテント・タイプを含んでいるパラメータ，添付ファイルの一意の ID を含んでいるパラメータ。

たとえば，［内容］フィールドに MyParam という名前を指定すると，最初の添付ファイルのパラメータ名は次のようになります。

```
MyParam_1
MyParam_1_ContentType
MyParam_1_ContentID
```

［**添付ファイルをインデックスとして保存する**］を指定する場合は，添付ファイルを格納するパラメータのインデックス番号と名前を指定します。［**内容**］で指定したパラメータ名は，ContentType および ContentID パラメータのプレフィックスとして使用されます。

入力または出力の添付ファイルのプロパティを編集するには，左側の表示枠で添付ファイルをクリックし，右側の表示枠で必要な情報を入力します。

SOAP ヘッダ

このビューは，メソッドのツリー・ビューで［**SOAP ヘッダ**］を選択したときに使用できます。右側の表示枠で，SOAP ヘッダを使用するかどうかを指定できます。SOAP ヘッダを使用するには，［**SOAP ヘッダを使用する**］を選択します。各要素について SOAP ヘッダを個別に指定する必要があります。SOAP ヘッダの XML コードをインポートするか，［XML の編集］オプションを使用して自分で作成できます。詳細については，119 ページ「XML ツリーの編集」を参照してください。

XML を使った作業

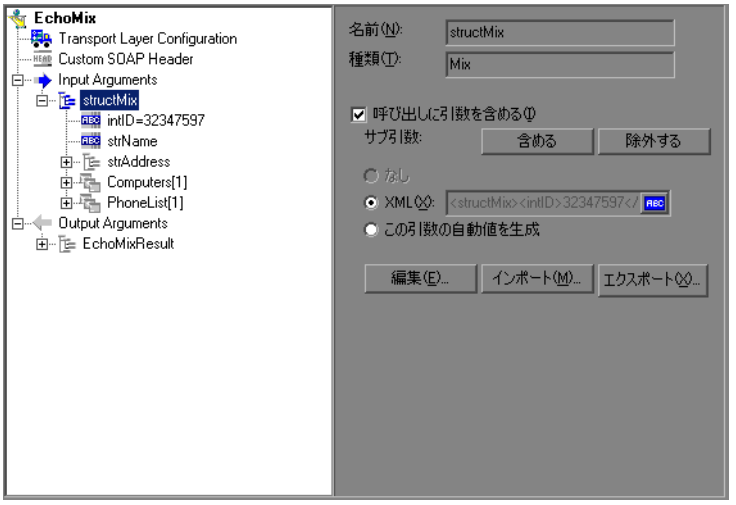
Web サービスでは，XML の表示と編集ができます。

以降の各項では，次の項目について説明します。

- XML ツリーの編集
- SOAP 応答の保存とコピー

XML ツリーの編集

VuGen の XML エディタを使用すると，複合型（構造体やオブジェクトなど）および配列の XML 表現を表示および編集できます。



XML 要素の値の入力は，面倒で間違いを起こしやすい作業です。VuGen が提供するインタフェースを使用すると，情報の入力，保存，および復元の作業が簡単になります。データを手作業で入力した後，「**エクスポート**」オプションを使用してデータを XML ファイルに保存できます。以降のテストではこのファイルをインポートするだけで済み，値を再び入力し直す必要がなくなります。

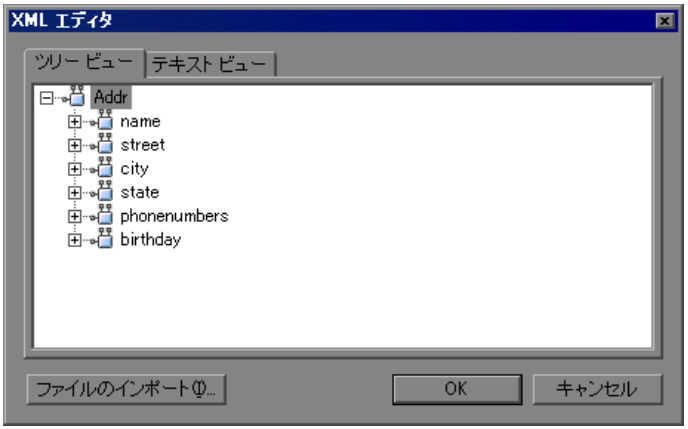
XML 文字列を編集するには，次の手順で実行します。

1 要素を変更したい Web サービス呼び出しを選択し，［**ステップのプロパティ**］タブをクリックします。

2 メソッドのツリー階層で，複合型または配列引数をクリックします。最も右側の表示枠に，XML コードが単独の文字列として表示されます。［**XML**］オプションを選択します。

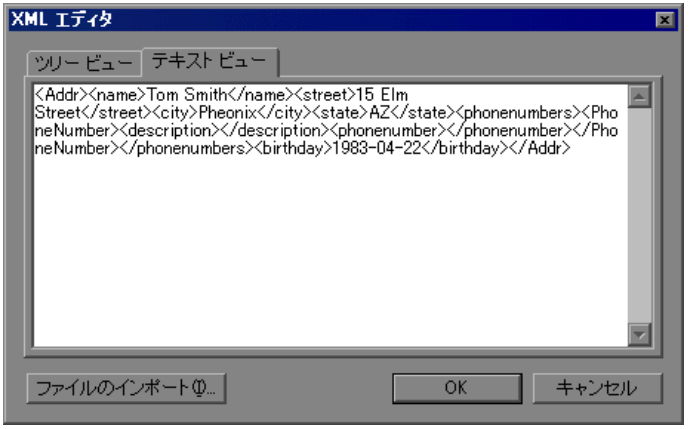
3 そのエントリに対応する XML コードを編集するには，［**編集**］をクリックします。VuGen では，XML 要素自体の変更ではなく，要素の値および配列要素の数に対する変更のみ保存されることを示す警告が発行されることがあります。

4 ［**OK**］をクリックします。［XML エディタ］ダイアログ・ボックスが開きます。



5 XML ツリー・ビューでノードをダブルクリックしてプロパティ・ダイアログ・ボックスを開きます。必要に応じて値を編集します。［**OK**］をクリックして，新しい値を保存します。

6 コードをテキスト・モードで編集するには，［**テキスト ビュー**］タブをクリックします。XML コードを手作業で編集します。［**OK**］をクリックして，変更内容を保存します。



7 以前に保存した XML ファイルをインポートするには，［**インポート**］をクリックし，ファイルの場所を指定します。ファイルを［XML エディタ］ダイアログ・ボックスで編集します。

8 XML データをファイルに保存してほかのテストで使用できるようにするには，［**エクスポート**］をクリックし，場所を指定します。

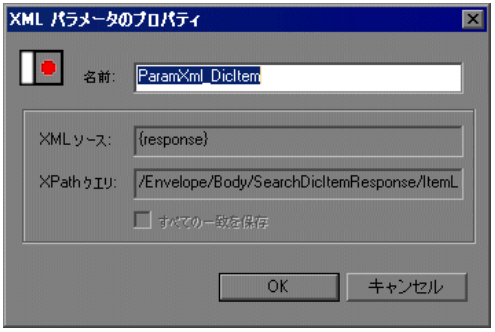
SOAP 応答の保存とコピー

入力引数の値を XML タイプのパラメータとして保存できほか，SOAP 応答をパラメータに保存したり，エディタで使用するためにコピーしたりできます。

SOAP 応答をパラメータに保存するには，次の手順を実行します。

1 ［**スナップショット**］タブに切り替えて，値をパラメータ化する対象の親または子要素を選択します。

2 右クリック・メニューから［**パラメータに XML を保存**］を選択します。［XML パラメータのプロパティ］ダイアログ・ボックスに，選択した XML 要素のプロパティが表示されます。



3 XML パラメータ名を指定して，［**OK**］をクリックします。

別のエディタ内で使用するために XML 構造をコピーするには，右クリックメニューから［**XML のコピー**］を選択します。

第 7 章

Web サービス ― 再生の準備

Web サービス・スクリプトを作成したら，環境を正確にエミュレートできるように再生の準備を行います。再生後に，テスト結果を表示してサービスが期待どおりに実行されたかどうかを確認します。

本章の内容

以降の情報は，Web サービスおよび SOA Vuser スクリプトのみを対象とします。

Web サービス・スクリプトの再生準備について

Web サービス呼び出しを使ってスクリプトを作成したら，再生の準備を行います。

ユーザ定義のエラーやログ・メッセージ，またはトランザクションを使ってスクリプトを拡張できます。詳細については，『**第 1 巻 ― VuGen の使用**』の第 6 章「Vuser スクリプトの拡張」を参照してください。

さらに，JMS 関数 **jms_<suffix>**，または XML 関数 **lr_xml_<suffix>** を使用して，スクリプトを強化することもできます。詳細については，『**Online Function Reference**』（英語版）（［**ヘルプ**］＞［**関数リファレンス**］）を参照してください。

実際のユーザをより正確にエミュレートするために，実行環境の設定を行えます。これらの設定には，一般的な実行環境の設定（反復，ログ，思考遅延時間，および一般的な情報）が含まれています。Web サービス固有の設置は JMS トランスポート・メソッドに関連しています。124 ページ「Web サービス JMS の実行環境の設定」を参照してください。

一般の実行環境の設定については，『**第 1 巻 ー VuGen の使用**』の「実行環境の設定」を参照してください。

スクリプトを再生する前に，サーバから正しい応答を受け取っていることを確認するために，チェックポイントを設定できます。詳細については，次を参照してください。

チェックポイント

機能テストで最も重要なタスクの 1 つは，サーバからの応答をチェックして，テストによって操作が正しく実行されたことを確認することです。Web サービスでは，応答には，それぞれが複数のデータ項目を含んだ引数が複数含まれていることがあります。［**Checkpoint**］タブは，テストに必要な応答値を一元的に定義するために使用します。

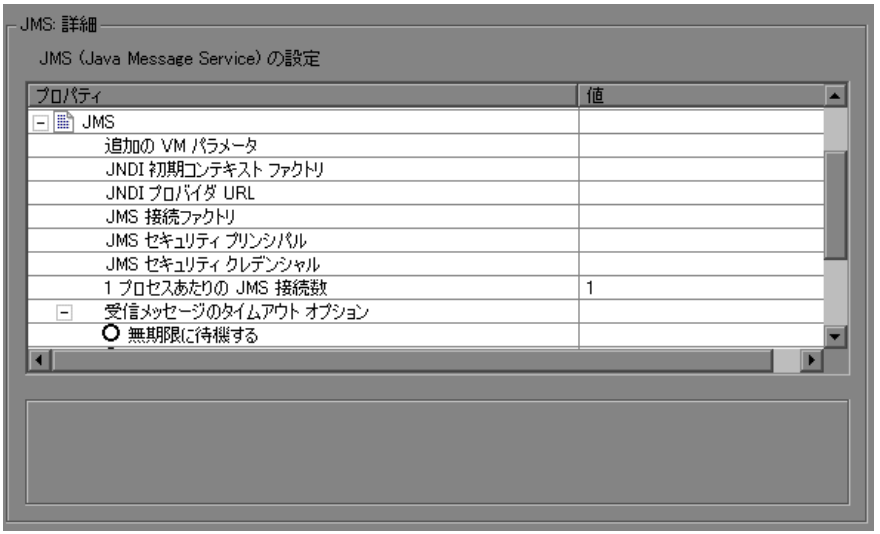
チェックポイント検証は，Service Test ライセンスを持つ HP Service Test または HP LoadRunner でのみ利用できます。詳細については，HP のサポートにお問い合わせください。

Web サービス JMS の実行環境の設定

Web サービス呼び出しのトランスポートとして JMS を使用するには，いくつかのリソースの割り当ておよび設定が必要です。これらのリソースには，JVM，JNDI 初期化パラメータ，JMS リソース，およびタイムアウト値が含まれます。トランスポート・レベルの設定の詳細については，第 11 章「Web サービス ー トランスポート層とカスタマイズ」を参照してください。

VuGen では，実行環境の設定でこれらのリソースのいくつかを設定できます。

VM（仮想マシン），JMS 接続，およびメッセージ・タイムアウトの領域のオプションを設定できます。



VM

- ［**外部 VM を使用する**］: 標準以外の VM（仮想マシン）を選択できます。このオプションを無効にすると，Vuser は VuGen で提供される JVM を使用します。
- ［**JVM ホーム**］: 外部 JVM の場所です。JDK_HOME によって定義されている JDK のホーム・ディレクトリを指すようにします。VuGen では，JDK 1.4 以降をサポートしています。
- ［**クラスパス**］: ほかの必要なサポート・クラスとともにベンダによって実装された JMS クラスです。JMS 実装ベンダによって決定されます。

JMS

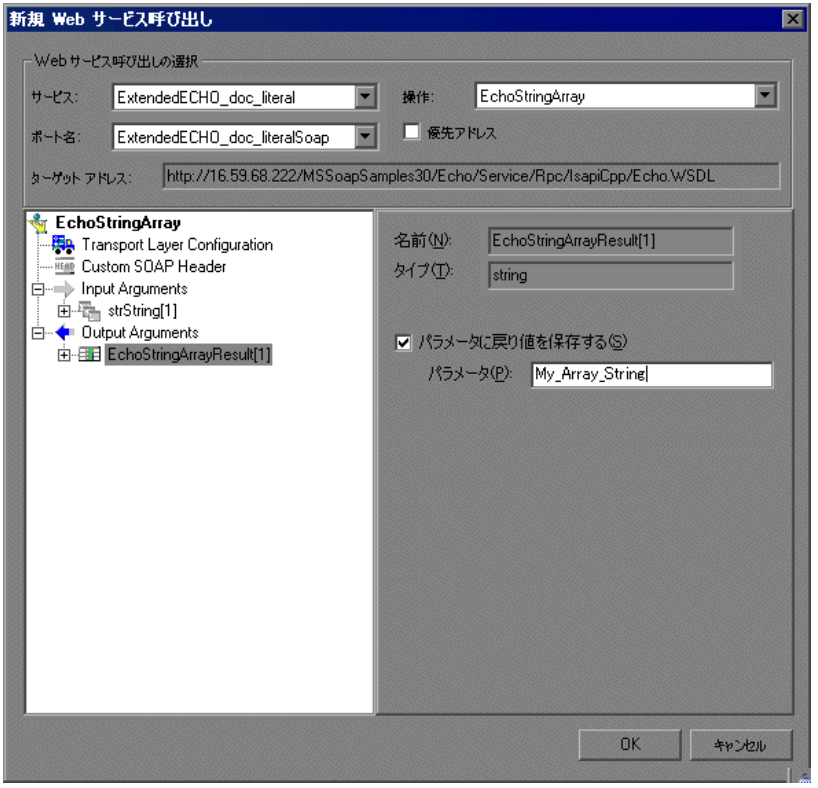
- ［**追加の VM パラメータ**］: Xbootclasspath や JVM ドキュメントで指定されているパラメータなど，JVM に送る追加のパラメータです。
- ［**JNDI 初期コンテキスト ファクトリ**］: 初期コンテキストを作成するファクトリ・クラスの完全指定クラス名です。リストからコンテキスト・ファクトリを選択するか，独自のコンテキスト・ファクトリを入力してください。

- ［**JNDI プロバイダ URL**］：サービス・プロバイダの URL の文字列です。次に例を示します。
  Weblogic - t3://myserver:myport
  Websphere - iiop://myserver:myport
- ［**JMS 接続ファクトリ**］：JMS 接続ファクトリの JNDI 名です。1 つのスクリプトに指定できる接続ファクトリは 1 つだけです。
- ［**JMS セキュリティ プリンシパル**］：認証方式におけるプリンシパル（ユーザなど）の識別情報です。
- ［**JMS セキュリティ クレデンシャル**］：認証方式におけるプリンシパルの資格情報です。
- ［**1 プロセスあたりの JMS 接続数**］：**mdrv** プロセス，または Vuser ごとの JMS 接続の数です。接続を共有するすべての Vuser は，同じメッセージを受信します。標準設定は 1 Vuser，最大値は 50 Vuser です。プロセスごとの接続数が少ないほどパフォーマンスは向上します。
- ［**受信メッセージ タイムアウト オプション**］：受信メッセージのタイムアウトです。標準設定は［**待機しない**］です。
  - ［**無期限に待機する**］：メッセージを必要なだけ待機してから次に進みます。
  - ［**待機しない**］：受信メッセージを待機せず，すぐにスクリプトに制御を戻します。キューにメッセージがなければ操作は失敗となります。
  - ［**タイムアウトを指定する（秒単位）**］：メッセージのタイムアウト値を手作業で指定します。タイムアウトするまでにメッセージが到着しなければ操作は失敗となります（標準設定）。
  - ［**ユーザ定義タイムアウト**］：タイムアウトするまでにメッセージを待機する時間を指定します（秒単位）。標準設定は 20 秒です。
- ［**自動生成セレクタ**］：要求の相関 ID を持つ応答メッセージに対してセレクタを生成します（標準設定は［**なし**］です）。サーバに送信される各 JMS メッセージは，固有の ID を持っています。VuGen でメッセージ ID を含むセレクタを自動的に作成する場合は，このオプションを有効にします。

Web サービスの出力パラメータの使用方法

場合によっては，ある Web サービス呼び出しの結果を別のサービスの入力として使用する必要がある場合があります。このような操作を行うには，出力パラメータに結果を保存して，必要なときに参照します。

次の例では，出力引数はパラメータ **My_Array_String** に保存されます。

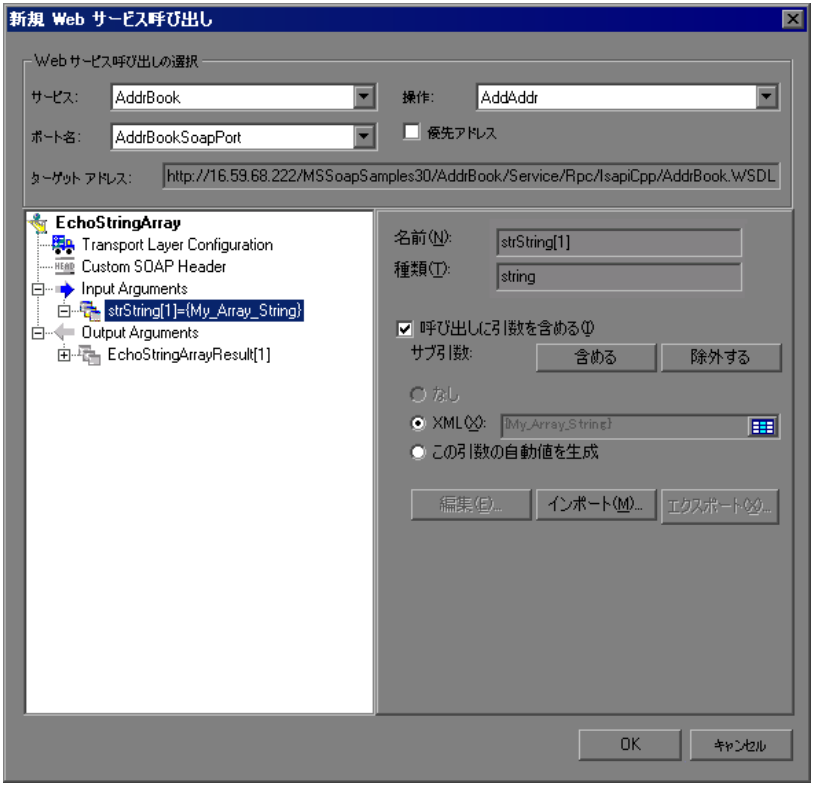


このスクリプトでは，保存した出力パラメータが結果の引数として示されます。

```
web_service_call( "StepName=EchoStringArray_101",
"SOAPMethod=ExtendedECHO_doc_literal.ExtendedECHO_doc_literalSoap.EchoStri
ngArray",
        "ResponseParam=response",
        "Service=ExtendedECHO_doc_literal",
        "Snapshot=t1169994766.inf",
        BEGIN_ARGUMENTS,
        "xml:strString=<strString><string></string></strString>",
        END_ARGUMENTS,
        BEGIN_RESULT,
        "EchoStringArrayResult/*[1]=My_Array_String",
        END_RESULT,
        LAST);
```

結果をパラメータに保存する方法の詳細については，130 ページ「出力パラメータの保存」を参照してください。

出力パラメータを保存した後は，パラメータ置換，または評価や値の出力など，その他の参照に利用できるようになります。次の例では，保存した出力パラメータ **My_Array_String** は，後の Web サービス呼び出しに使用する入力引数として使用されます。



保存されている出力パラメータの使用方法の詳細については，131 ページ「保存したパラメータを入力に使用する方法」を参照してください

出力パラメータの保存

複数の結果の引数は，［プロパティ］ダイアログ・ボックスを使って，またはスクリプト・コード内で手作業で編集することによって，パラメータに保存できます。

また，結果のパラメータを XML アクションから保存して，入力引数として使用することもできます。たとえば，**lr_xml_insert** の結果のパラメータを保存した場合，保存したパラメータを後の Web サービス呼び出しで参照できます。詳細については，『**Online Function Reference**』（英語版）を参照してください。

出力パラメータを保存するには，次の手順を実行します。

1 ツリー・ビューでスクリプトを表示します。

ツリー・ビューが表示されていることを確認します。表示されていない場合は，［**表示**］＞［**ツリー ビュー**］を選択します。

2 サービスを確認します。

［**サービスの管理**］ボタンをクリックして少なくとも 1 つのサービスがインポートされていることを確認します。新しいサービスをインポートするには，［サービスの管理］ダイアログ・ボックスで［**インポート**］をクリックします。

3 ステップのプロパティを表示します。

新しい Web サービス呼び出しについては，［**サービス呼び出しの追加**］をクリックします。

既存の Web サービス呼び出しについては，ステップをダブルクリックするか，または右側の表示枠で［**プロパティ**］タブをクリックします。

4 出力引数を選択します。

左側の表示枠で，値をパラメータに保存する出力引数を選択します。

5 出力パラメータの保存を有効にします。

右側の表示枠で，［**パラメータに戻り値を保存する**］を選択します。標準設定の名前を受け入れるか，またはユーザ定義の名前を指定します。

保存したパラメータを入力に使用する方法

保存した出力パラメータは，後の Web サービス呼び出しで使用できます。

また，XML アクションから保存した結果のパラメータを入力として使用することもできます。たとえば，**lr_xml_insert** の結果のパラメータを保存した場合，後の Web サービス呼び出しで参照できます。詳細については，『**Online Function Reference**』（英語版）を参照してください。

保存したパラメータを入力に使用するには，次の手順を実行します。

1 ツリー・ビューにステップのプロパティを表示します。

新しい Web サービス呼び出しについては，［**サービス呼び出しの追加**］をクリックします。

既存の Web サービス呼び出しについては，ステップをダブルクリックするか，または右側の表示枠で［**プロパティ**］タブをクリックします。

2 入力引数を選択します。

左側の表示枠で，以前保存した出力パラメータに値を置き換える入力引数を選択します。

3 パラメータ選択ダイアログ・ボックスを開きます。

右側の表示枠で，［**値**］を選択して［**値**］ボックスの横にある［ABC］アイコンをクリックします。［パラメータの選択または作成］ボックスが開きます。

4 出力パラメータを選択します。

必要な出力パラメータをドロップダウン・リストから選択し，［**OK**］をクリックします。

スクリプト・ビューで入力パラメータを指定するには，置換する値を選択し，右クリック・メニューで［**既存のパラメータを使用**］を選択します。利用できるパラメータのいずれかを選択します。

注 : スクリプト・ビューで出力パラメータ名を変更した場合，ツリー・ビューに切り替えるまでパラメータ・リスト内で更新されません。

特別な場合の処理

この項では，特別な場合においてスクリプトを実行するためのガイドラインを提供します。

Any タイプの XSD

XSD スキーマを使用する Web サービスで，**Any** タイプの要素 **<xsd:element name="<Any_element>" type="xsd:anyType" />** を含むものについては，スクリプトが次のモデルに準拠していることを確認してください。

```
BEGIN_ARGUMENTS,
                    "xml:Any_element="
                        "<Any_element>"
                            "<string>the string to send</string>"
                        "</Any_element>",
END_ARGUMENTS,
```

実際の SOAP はやや異なる可能性がありますが，スクリプトが前述のモデルに準拠しているかぎり適切に実行できます。

また，< any > タイプに対して complex タイプの要素を送信することもできます。次に例を示します。

```
"xml:Any_element="
                        "<Any_element>"
                            "<myComplexTypeName>"
                                "<property1>123</property1>"
                                "<property2>456</property2>"
                            "</myComplexTypeName>"
                        "</Any_element>",
```

第 8 章

Web サービス－データベース統合

VuGen を使用すると，ライブ・データベース・サービスと統合して，テストのデータを取得できます。

本章の内容

データベース統合について

Web サービスをテストするときは，正確で最新のデータを使用することが重要です。過去のデータのスナップショットを使用すると，有効または妥当なものでなくなるおそれがあります。

データベース統合によって，テスト中にデータベースの値にアクセスでき，データを最新のものにできます。

データベース統合は次のシナリオで役立ちます。

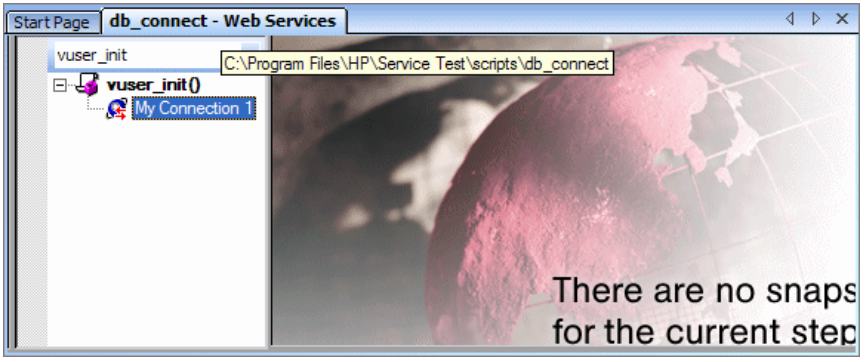
- SQL クエリから取得したデータの使用
- Web サービス呼び出し後のデータベース値の検証
- データベース経由の戻り値のチェック

Service Test はすべてのデータベース相互作用情報を保存し，［テスト結果］レポートに表示します。詳細については，『**第 1 巻 － VuGen の使用**』の第 10 章「テスト結果の表示」を参照してください。

データベースへの接続

データベースに接続するには，スクリプトに接続ステップを追加します。ツリー・ビューで［ステップの追加］ダイアログ・ボックス（［**挿入**］＞［**新規ステップ**］）を使用して，データベース接続ステップを追加します。組み込みの接続文字列ジェネレータによって，データベースと証明書に固有のデータベース接続文字列の作成が示されます。ステップを挿入する前に，接続をテストすることもできます。

反復を使用したスクリプトを実行するとき，仮想ユーザはスクリプトの **Action** セクションを繰り返すだけです。**Action** セクションにデータベース接続ステップを含めると，テストで反復ごとにそのステップが繰り返されます。Vuser は **vuser_init** または **vuser_end** セクションではなく，スクリプトの **Action** セクションを繰り返すだけです。そのため，データベース接続ステップは **vuser_init** セクションに，切断ステップ **lr_db_disconnect** は **vuser_end** セクションに配置することをお勧めします。



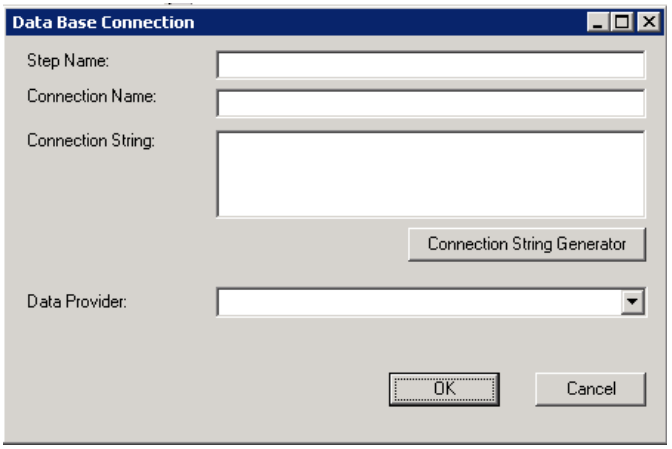
1 回のクエリを実行して，データをスクロールする必要がある場合は，同様に **vuser_init** セクションに **Database: Execute SQL Query** ステップを配置してください。

ツリー・ビューでデータベース接続ステップを追加するには，次の手順を実行します。

1 ［**表示**］＞［**ツリー ビュー**］を選択して，ツリー・ビューに入ります（まだ表示されていない場合）。

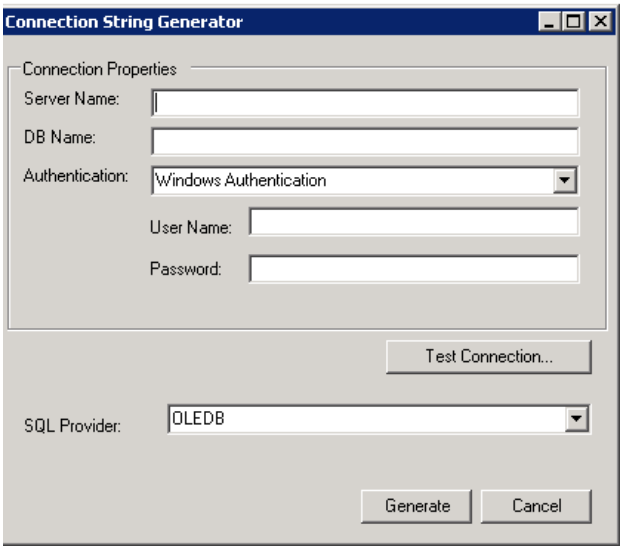
2 目的とするオプション，**vuser_init** または **Action** を選択します。接続ステップは **vuser_init** セクションに配置することをお勧めします。詳細については，下記を参照してください。

3 ［**挿入**］＞［**新規ステップ**］を選択します。**Database: Connect** ステップを選択します。［Data Base Connection］ダイアログ・ボックスが表示されます。



4 ［**Step Name**］，［**Connection Name**］，および［**Data Provider**］，OLEDB または SQL を指定します。

5［**Connection String Generator**］をクリックすると，使用環境固有のデータベース接続文字列が生成されます。



6 接続プロパティを指示します。

- ［**Server Name**］
- ［**DB Name**］
- ［**Authentication**］方式：Windows 認証またはユーザ / パスワード
- ［**Username**］および［**Password**］

7［**Test Connection**］をクリックして，指定した情報が正しいことを確認します。

8［**SQL Provider**］，OLEDB または SQL を選択し，［**Generate**］をクリックします。

lr_db_connect の必要な構文に関する詳細は，『**Online Function Reference**』（英語版）（［**ヘルプ**］＞［**関数リファレンス**］）を参照してください。

SQL クエリから取得したデータの使用

このシナリオでは，テストによってデータベースからデータをフェッチして，Web サービスの呼び出しなど，その後のスクリプトで使用します。スクリプトによって各テスト実行中にデータを取得するので，データは最新で適切なものになります。

次の表に，スクリプトの一般的なフローを示します。

ステップ	API 関数
データベースに接続する	lr_db_connect
SQL クエリを実行する	lr_db_executeSQLStatement
データを取得して保存する	lr_db_getvalue to ＜ param_name ＞
Web サービスを呼び出す	web_service_call with { ＜ param_name ＞ }
データベースから切断する	lr_db_disconnect

次の 2 つの方法で結果を反復できます。

- 各反復中に結果を単純パラメータに保存する
- VuGen に組み込まれている反復を使って，データをスクロールする

詳細については，『**Online Function Reference**』（英語版）（［**ヘルプ**］＞［**関数リファレンス**］）を参照してください。

次の例では，**vuser_init** セクションがデータベースに接続し，データベース・クエリを実行します。

```
vuser_init()
{
lr_db_connect("StepName=myStep",
    "ConnectionString=Initial Catalog=MyDB;Data Source=mylab.net;user id =sa
;password = 12345;" ,
    "ConnectionName=MyConnection",
    "ConnectionType=SQL",
    LAST);

lr_db_executeSQLStatement("StepName=MyStep",
        "ConnectionName=MyConnection",
        "SQLQuery=SELECT * FROM Addresses",
        "DatasetName=ds1",
        LAST);

    return 0;
}
```

テストの最後に，**vuser_end** セクションでデータベースから切断します。

```
vuser_end()
{

    lr_db_connect("StepName=myStep",
        "ConnectionString=Initial Catalog=MyDB;Data Source=LAB1.devlab.net;user id
=sa ;password = soarnd1314;" ,
        "ConnectionName=MyConnection",
        "ConnectionType=SQL",
        LAST);

    return 0;
}
```

Action セクションには，繰り返すステップを含めます。**Row** 引数の使い方に注意してください。データベースの最初の呼び出しでは，最初の行に **Row=next** を指定します。同じ行の別の値を取得するには，**current** を使用します。

```
Action()
{
    lr_db_getvalue("StepName=MyStep",
        "DatasetName=ds1",
        "Column=Name",
        "Row=next",
        "OutParam=nameParam",
        LAST);

    lr_db_getvalue("StepName=MyStep",
        "DatasetName=ds1",
        "Column=city",
        "Row=current",
        "OutParam=cityParam",
        LAST);

/* Web サービス呼び出しでデータベースから取得する値を使用する */
    web_service_call( "StepName=EchoAddr_101",
        "SOAPMethod=SanityService|SanityServiceSoap|EchoAddr",
        "ResponseParam=response",
        "Service=SanityService",
        "ExpectedResponse=SoapResult",
        "Snapshot=t1227168459.inf",
        BEGIN_ARGUMENTS,
        "xml:addr="
            "<addr>"
                "<name>{nameParam}</name>"
                "<street></street>"
                "<city>{cityParam}</city>"
                "<state></state>"
                "<zip></zip>"
            "</addr>",
        END_ARGUMENTS,
        BEGIN_RESULT,
        END_RESULT,
        LAST);

return 0;
}
```

Web サービス呼び出し後のデータベース値の検証

このシナリオでは，バックエンドでデータベースを変更する Web サービスをテストで実行します。このシナリオの目標は，データベースに結果としてもたらされる値が正しいか検証することです。

次の表に，スクリプトの一般的なフローを示します。

ステップ	API 関数
データベースに接続する	lr_db_connect (in **vuser_init** section)
Web サービスを呼び出す	web_service_call
SQL クエリを実行する	lr_db_executeSQLStatement
データを取得して保存する	lr_db_getvalue to ＜ param_name ＞
データをチェックする	lr_checkpoint
データベースから切断する	lr_db_disconnect (in **vuser_end** section)

詳細については，『**Online Function Reference**』（英語版）（［**ヘルプ**］＞［**関数リファレンス**］）を参照してください。

次の例で，データをチェックするこのプロセスを説明します。

```
Action()
{
/* バックエンドでデータベースを変更する Web サービス呼び出し。*/
    web_service_call( "StepName=addAddr_102",
        "SOAPMethod=Axis2AddrBookService|Axis2AddrBookPort|addAddr",
        "ResponseParam=response",
        "Service=Axis2AddrBookService",
        "ExpectedResponse=SoapResult",
        "Snapshot=t1227169681.inf",
        BEGIN_ARGUMENTS,
        "xml:arg0="
            "<arg0>"
                "<name>{Customers}</name>"
                "<city>{City}</city>"
            "</arg0>",
        END_ARGUMENTS,
        LAST);

/* Web サービスで変更した顧客名でデータベースを照会する */
    lr_db_executeSQLStatement("StepName=MyStep",
        "ConnectionName=MyConnection",
        "SQLQuery=SELECT * FROM Addresses WHERE name = '{Customers}' ",
        "DatasetName=ds1",
        LAST);

/* データベース・クエリで検索した値を取得する。*/
    lr_db_getvalue("StepName=MyStep",
        "DatasetName=ds1",
        "Column=Name",
        "Row=current",
        "OutParam=CustomerName",
        LAST);

/* 実際の値とデータベースに保管された期待値を比較する。*/
    lr_checkpoint("StepName=validateCustomer",
        "ActualValue={Customers}",
        "ExpectedValue={CustomerName}",
        "Compare=Equals",
        "StopOnValidationError=false",
        LAST);

return 0;
}
```

データベース経由の戻り値のチェック

このシナリオでは，XML 応答を返す Web サービス呼び出しをユーザが実行します。このシナリオの目標は，期待値に対して，Web サービス呼び出しの応答を検証することです。期待値はデータベースに保管されます。スクリプトでデータベースから期待値をフェッチして，実際の応答と比較します。

次の表に，スクリプトの一般的なフローを示します。

ステップ	API 関数
データベースに接続する	lr_db_connect (in **vuser_init** section)
Web サービスを呼び出す	web_service_call with **Result=<result_param>**
SQL クエリを実行する	lr_db_executeSQLStatement
期待データを取得する	lr_db_getvalue to ＜ param_name ＞
データを検証する	soa_xml_validate with an XPATH checkpoints
データベースから切断する	lr_db_disconnect (in **vuser_end** section)

XML 検証ツールを使用して，応答データのチェックポイントを作成できます。検証ステップを作成するときは，**lr_db_getvalue** で取得したデータベース・パラメータを使用します。

次の例で，Web サービス呼び出しで返されるデータの一般的な検証を説明します。検証ステップで実際の期待値を比較します。

```
Action()
{
    web_service_call( "StepName=GetAddr_102",
        "SOAPMethod=AddrBook|AddrBookSoapPort|GetAddr",
        "ResponseParam=response",
        "Service=AddrBook",
        "ExpectedResponse=SoapResult",
        "Snapshot=t1227172583.inf",
        BEGIN_ARGUMENTS,
        "Name=abcde",
        END_ARGUMENTS,
        BEGIN_RESULT,
        END_RESULT,
        LAST);

    lr_db_executeSQLStatement("StepName=MyStep",
        "ConnectionName=MyConnection",
        "SQLQuery=SELECT * FROM Addresses WHERE name = 'abcde' ",
        "DatasetName=ds1",
        LAST);

    lr_db_getvalue("StepName=MyStep",
        "DatasetName=ds1",
        "Column=Name",
        "Row=current",
        "OutParam=CustomerName",
        LAST);

    soa_xml_validate ("StepName=XmlValidation_1146894916",
        "Snapshot=t623713af7a594db2b5fef43da68ad59d.inf",
        "XML={GetAddrAllArgsParam}",
        "StopOnValidationError=0",
        BEGIN_CHECKPOINTS,
            CHECKPOINT,"XPATH=/*[local-name(.)='GetAddr'][1]/*[local-
name(.)='Result'][1]/*[local-name(.)='name'][1]","Value_Equals={CustomerName}",
        END_CHECKPOINTS,
        LAST);
    return 0;
}
```

詳細については，『**Online Function Reference**』（英語版）（［**ヘルプ**］＞［**関数リファレンス**］）を参照してください。

データセットに対するアクション実行

VuGen では，SQL クエリで返されるデータセットにアクションを実行できます。

lr_db_dataset_action 関数は，データセットに対して次のアクションを実行します。

- ＜**リセット**＞：データセットの最初のレコードにカーソルをセットします。
- ＜**削除**＞：データセットに割り当てられたメモリを解放します。
- ＜**プリント**＞：データセット全体の内容を再生ログとほかのテスト・レポート・サマリに出力します。

lr_db_getvalue によってバイナリ・データを取得すると，**Print** アクションを使って，その内容を出力することはできません。

この関数の構文と使い方については，『**Online Function Reference**』（英語版）（［**ヘルプ**］＞［**関数リファレンス**］）を参照してください。

第 9 章

Web サービス－セキュリティ

上級ユーザはトランスポート層のプロパティとセキュリティ・ポリシーを設定し，Web サービス呼び出しの動作を定義するユーザ・ハンドラを作成して，Web サービス呼び出しをカスタマイズできます。

本章の内容

以下の情報は，Web サービスと SOA Vuser スクリプトのみを対象とします。

Web サービス・テストへのセキュリティ追加について

Web サービス・アプリケーションを作成する場合，安全で拡張が容易なアプリケーションを作成するのは簡単なことではありません。Secure Sockets Layer（SSL）などの安全なトランスポートを介してメッセージを送信すれば，Web サービスのセキュリティを保護できますが，これはポイントツーポイント通信に限られます。

メッセージを安全に送信できるようにするために，VuGen では，セキュリティ・トークン（WS-Security）や SAML など複数のセキュリティ・メカニズムをサポートしています。

トークンの詳細については，次を参照してください。SAML の詳細については，153 ページ「SAML オプションの設定」を参照してください。

WS-Security のカスタマイズと WCF を使用したテストに関する詳細は，第 10 章「Web サービスー高度なセキュリティと WS 仕様」を参照してください。

注：WSDL がセキュアな場所にある場合は，［サービスの管理］ダイアログ・ボックスでセキュリティ情報を提供する必要があります。詳細については，52 ページ「WSDL 接続オプションの指定」を参照してください。

Service Test では，Web サービス呼び出しのセキュリティを設定する 2 つのモデル，**レガシ**および**シナリオ**をサポートしています。本章では，**web_service_set_security** を使用したレガシ・セキュリティ・モデルについて説明します。シナリオ・モデルについては，第 10 章「Web サービスー高度なセキュリティと WS 仕様」を参照してください。

次の表に，各モデルの使用に関する注意事項を示します。

レガシ・モデル	シナリオ・ベース・モデル
レガシ・モデルを使用しているスクリプトを使用する	WCF サービスをテストする
.NET 2.0，Axis，またはほかの古いツールキットなどのフレームワークで作成されたサービスをテストする	Axis2 や Metro（WSIT）など，新しいフレームワークで作成されたサービスをテストする
WS-Security トークンに対する低レベル制御が必要になる	サービスで WS-SecureConversation や WS-Trust など，高度な仕様を使用する
新しいモデルを使用するのが難しいので，レガシのニーズに適した機能を見つける	レガシ・モデルを使用するのが難しいので，新しいモデルのより適切な機能を見つける

Web サービス呼び出しへのセキュリティ追加 ー 一般ワークフロー

次の項では，Web サービス呼び出しにセキュリティを追加する一般ワークフローについて説明します。

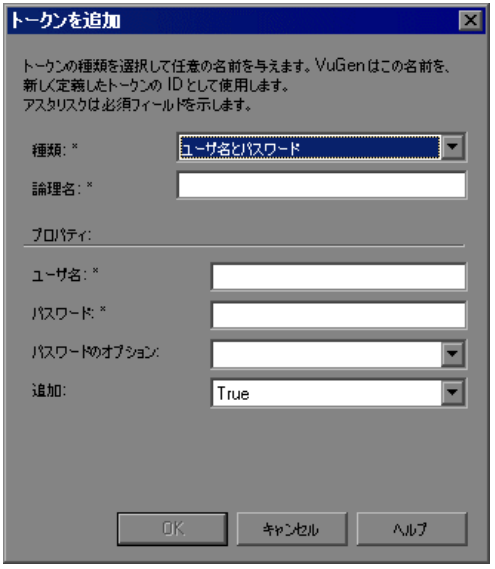
Web サービス・セキュリティを追加するには，次の手順を実行します。

1 セキュリティ設定を追加するポイントにカーソルを置きます。ほとんどの場合は，セキュリティ範囲をスクリプト全体に適用できるように，vuser_init セクションに配置することをお勧めします。特定の呼び出しにセキュリティを設定したいだけなら，目的とする位置にセキュリティを配置します。

2 [**挿入**] > [**新規ステップ**] を選択し，[ステップの追加] ダイアログ・ボックスを開きます。

3 [**Web Services Set Security**] を選択して，[**OK**] をクリックします。[セキュリティ プロパティの設定] ダイアログ・ボックスが表示されます。

4［**追加**］をクリックして，新しいトークンを追加します。［トークンを追加］ダイアログ・ボックスが表示されます。



5 トークンのタイプを選択します。一般的なトークンはユーザ名と X.509 証明書です。トークンのタイプの詳細については，149 ページ「セキュリティ・トークンおよび暗号化」を参照してください。

［**論理名**］ボックスに，VuGen によってトークンの識別に使用されるトークンの任意の名前を指定します。

User Name と Password タイプのトークンの場合には，［**ユーザ名**］ボックスと［**パスワード**］ボックスに必要な情報を入力します。

SOAP エンベロープ・ヘッダに明示的にトークンを送信するには，［**True**］を選択します。SOAP エンベロープ・ヘッダからトークンを除外するには，［**False**］を選択します。

6 メッセージ・パケットの有効期間を指定するには，［**存続期間**］を選択して，時間（秒）を指定します。

7 メッセージ署名と暗号化が必要な場合は，［**追加**］をクリックして追加します。署名および暗号化については，暗号化 / 署名トークンとして事前に定義されたトークンを指定する必要があります。SOAP で特定の XPath を暗号化したり署名する方法については，157 ページ「web_service_set_security の使用例」を参照してください。

8 スクリプト内の特定ポイントでセキュリティ設定を中止するには，目的のポイントに **Web Service Cancel Security** ステップを追加します。

9 ［**OK**］をクリックします。カーソルの位置に Web Services Set Security ステップが挿入されます。

157 ページ「web_service_set_security の使用例」で，一般的なセキュリティ設定のいくつかについて説明します。

セキュリティ・トークンおよび暗号化

WS-Security 仕様では，SOAP メッセージ自体にセキュリティ信用証明を含めることができます。これは，クライアントに対して，送信者と受信者の両方によって信頼されている発行元からセキュリティ信用証明を取得するように指示することで実現します。SOAP メッセージの送信者が要求を送信する際，セキュリティ・**トークン**と呼ばれるセキュリティ信用証明が SOAP メッセージに含められます。Web サーバは，この SOAP 要求を受信したとき，送信者の信頼性を検証するために改めて要求を送信する必要がありません。サーバは，Web サービスによるアプリケーションの実行を認める前に，信用証明が信頼できるかどうか検証します。信用証明の発行元へ戻る必要がないため，アプリケーションのスケーラビリティが大幅に向上します。

Web サービスをさらに安全なものにするため，SOAP メッセージにはデジタル署名または暗号化を使用するのが一般的です。SOAP メッセージにデジタル署名を使用することによって，送信中にメッセージが改ざんされていないことが証明されます。SOAP メッセージを暗号化すると，意図された受信者以外の誰かがメッセージの内容を読むことが難しくなり，Web サービスの保護に役立ちます。

Web サービスのセキュリティ・メカニズムでは，セキュリティ・トークンがメッセージに関連付けられます。このメカニズムでは，さまざまな認証要件に対応するために，いくつかのセキュリティ・トークン形式をサポートしています。たとえば，クライアントは，身分証明書またはセキュリティ証明書の提示が必要になる場合があります。

WS-Security をサポートするために，VuGen ではスクリプトのセキュリティ・トークンを作成できるようになってます。複数のトークンを作成し，そのプロパティを設定できます。トークンを作成したら，そのトークンを使用して，SOAP メッセージに署名したり，SOAP メッセージを暗号化したりします。

場合によっては，トークンを明示的には送信しません。つまり，SOAP エンベロープ・ヘッダにトークン自体を含めずに，署名または暗号化の目的でトークンを使用します。**Add** オプションを使用すれば，実際のトークンを送信するかどうかを明示的に指定できます。

使用可能なトークンは，**ユーザ名とパスワード**，**X.509 証明書**，**Kerberos チケット**，**Kerberos2 チケット**，**セキュリティ・コンテキスト・トークン**，および**派生トークン**です。指定する必要がある情報は，トークンに応じて異なります。

- **ユーザ名と パスワード**：**ユーザ名とパスワード**トークンには，認証のためのユーザ識別情報（**ユーザ名**および**パスワード**）が含まれています。

  また，認証のためにパスワードをサーバに送信する方法を示す **パスワードのオプション**（**SendPlainText**，**SendNone**，**SendHashed**）も指定できます。

- **X.509 証明書**：このセキュリティ・トークンは，X.509 証明書に基づくトークンです。証明書を取得するには，ベリサイン社などの認証局から証明書を購入するか，独自に証明書サービスを構築して証明書を発行します。ほとんどの Windows サーバでは，証明書の作成を可能にする公開鍵基盤（PKI）をサポートしています。作成後，認証局で証明書に署名をしてもらうか，無署名の証明書を使用します。

  Vuser スクリプトに X.509 トークンを追加する場合は，**論理名**，**ストア名**，**キー識別子のタイプ**，**キー識別子の値**，および**格納場所**引数を指定します。

- **Kerberos チケット /Kerberos2 チケット**（Windows 2003 または XP SP1 以降）：Kerberos プロトコルは，オープンで安全ではないネットワーク上でユーザおよびサービスを相互に認証するのに使用されます。共有秘密鍵を使用して，ユーザの信用証明を暗号化し，署名します。そして，KDC（Kerberos Key Distribution Center）と呼ばれる第三者機関が，この信用証明の真正性を確認します。真正性の確認後，ユーザは，ネットワークの 1 つ以上のサービスにアクセスするためにサービス・チケットを要求できます。このチケットには，ユーザの暗号化され，かつ真正性が確認された識別情報が含まれます。チケットは，現在のユーザの信用証明を使用して取得されます。

  VuGen では，Kerberos と Kerberos2 両方のセキュリティ・トークンに基づいたトークンをサポートしています。Kereberos と Kerberos2 のトークンの主な違いは，Kerberos2 では Security Support Provider Interface（SSPI）を使用していて，クライアントのアイデンティティを獲得するのに高い特権を必要としないという点です。また，Kerberos2 セキュリティ・トークンは，Web ファームで運用されている Web サービスに送信される SOAP メッセージのセキュリティを保護するのにも使用できます。

Vuser スクリプトに Kerberos トークンを追加する場合は，トークンの **論理名**，および Web サービス・マシンの**ホスト**名および**ドメイン**名を指定します。

- **セキュリティ・コンテキスト・トークン**：このトークンは，有効期限が切れるまで繰り返し使用できるセキュリティ・トークンです。SOAP メッセージの送信者は，セキュリティ・コンテキスト・トークンを使用して，SOAP メッセージの送信者とターゲット Web サービスの間の一連の SOAP メッセージ（会話と呼ばれる）に署名をしたり，このメッセージを暗号化したりできます。このタイプのトークンの主な利点は次のとおりです。
  - セキュリティ・コンテキスト・トークンが期限切れになっていない限り，SOAP メッセージの送信者は，同じセキュリティ・コンテキスト・トークンを使用して，ターゲット Web サービスに送信する SOAP メッセージに署名をしたり，このメッセージを暗号化したりできます。
  - セキュリティ・コンテキスト・トークンは，対称鍵に基づいています。これは，SOAP メッセージのデジタル署名または暗号化において，非対称鍵以上にセキュリティ・コンテキスト・トークンを有効なものにします。
  - セキュリティ・コンテキスト・トークンは，SOAP メッセージを送信することによって，あるセキュリティ・トークン・サービスから別のセキュリティ・トークン・サービスに要求できます。

  Vuser スクリプトに**セキュリティ・コンテキスト**・トークンを追加する場合は，**論理名**，**基本トークン**，**発行者トークン**，**エンドポイント URI**，および**追加の適用先**引数の値を指定します。

- **派生トークン**：派生トークンは，派生がサポートされない X.509 を除く，既存の別のトークンに基づくトークンです。**論理名**および**派生元**トークンを指定する必要があります。元のトークンが削除されると，派生トークンは使用できなくなります。派生タイプのトークンは，再帰的に使用することはできません。

トークンの設定に関する詳細は，『**Online Function Reference**』（英語版）（［**ヘルプ**］＞［**関数リファレンス**］）を参照してください。

セキュリティ・ポリシーの追加

スクリプトのセクションにセキュリティ・ポリシーを追加するには，関連ステップを **Web Service Set Security** および **Web Service Cancel Security** ステップで囲みます。

Web Services Set Security ステップをスクリプトに追加すると，VuGen によって，セキュリティ・プロパティで定義したトークン，メッセージ署名，および暗号化を持つ引数を含んだ **web_service_set_security** 関数が追加されます。

```
web_service_set_security(
  SECURITY_TOKEN, "Type=USERNAME", "TokenName=mytoekn1",
  "UserName=bob", "Password=123", "PasswordOptions=SendNone",
  "Add=True", LAST);
```

トークンのタイプ，**論理名**，**基本トークン**，**発行者トークン**，**派生元**引数では，パラメータ化がサポートされていません。

メッセージ署名および暗号化データを使った作業

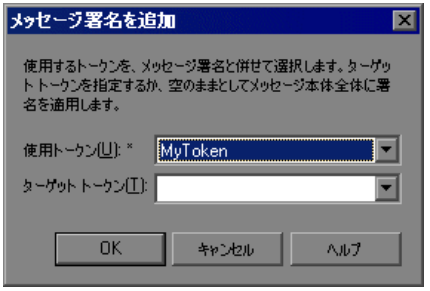
SOAP メッセージにセキュリティ・トークンを追加すると，セキュリティ・トークンは，XML 要素の形式で SOAP メッセージの WS-Security SOAP ヘッダに追加されます。

しかし，このメッセージは無防備なので，更なるセキュリティが必要です。これが特に該当するのは，パスワードを含め，信用証明情報がロールベースのセキュリティの場合のように平文で送信される場合です。

こうしたデータのセキュリティを保護するのに使用される 2 つの方法が，デジタル署名および暗号化です。

- **デジタル署名**：デジタル署名は，署名されてからメッセージが改ざんされていないことを確認するために，メッセージ受信者によって使用されます。デジタル署名は通常，XML の形式で SOAP メッセージ内にあります。受信者は，署名を調べて，有効であることを確認します。WSE などの特定の環境では，SOAP 受信者のコンピュータで自動的に署名が検証されます。
- **暗号化**：XML デジタル署名は，署名されてからメッセージが改ざんされていないことを検証するためのメカニズムを提供しますが，SOAP メッセージは暗号化しません。このメッセージは，依然として XML 形式の平文です。メッセージが無防備とならないようにセキュリティを保護するには，メッセージを暗号化して，侵入者がユーザのパスワードを取得するのを困難にします。

VuGen では，暗号化およびメッセージ署名に関する情報を指定できます。



なお，メッセージ署名および暗号化の引数では，パラメータ化はサポートされません。スクリプトへのメッセージ署名および暗号化の追加の詳細については，次を参照してください。

SAML オプションの設定

VuGen では，Web サービス用の SAML（Security Assertion Markup Language）をサポートしています。SAML とは，インターネット経由で，セキュリティに関連した「**アサーション**」と呼ばれる情報をビジネス・パートナーの間で交換するための XML 標準です。アサーションには，属性ステートメント，認証，決定ステートメント，および認可決定ステートメントを含めることができます。

SAML は，STS（セキュリティ・トークン・サービス）によって発行されたセキュリティ・トークンを用いて仲介された認証を使用します。STS は，クライアントおよび Web サービスから信頼されており，相互運用が可能なセキュリティ・トークンを提供します。SAML トークンは，プラットフォーム間での相互運用性，および同一のセキュリティ・ドメイン内に存在しないクライアントとサービスの間で情報を交換する手段を提供するため，Web サービス・セキュリティにとって重要です。

SAML 設定は，スクリプト全体，またはスクリプトの一部に対して設定できます。SAML セキュリティを設定するには，**Web Services Set Security SAML** ステップを追加します。SAML セキュリティを削除するには，**Web Services Cancel Security SAML** ステップを挿入します。

注 : SAML セキュリティおよび標準 Web サービス（**Web Service Set Security** ステップ）・セキュリティを同じステップに適用することはできません。Web サービス・セキュリティを中止するには，**Web Service Cancel Security** ステップを挿入します。

SAML アサーションの署名

VuGen は符号なし SAML アサーションに署名するメソッドを提供します。入力として，符号なしアサーション，証明書ファイル，およびオプション・パスワードを提供します。VuGen は出力として符号付き SAML アサーションを提供します。

このメソッドをスクリプトに追加するには，［ステップの追加］ダイアログ・ボックス（［**挿入**］＞［**新規ステップ**］）を使います。VuGen によって，**ws_sign_saml_assertion** がスクリプトに追加されます。構文については，『**Online Function Reference**』（英語版）（［**ヘルプ**］＞［**関数リファレンス**］）を参照してください。

ポリシー・ファイル

SAML ポリシー・ファイルは，WSE 3.0 標準に準拠し，SAML セキュリティの属性値を定義します。標準設定では，VuGen はインストールの **dat** フォルダにある **samlPolicy.config** ファイルを使用します。

SAML セキュリティ情報を入力する際には，プロパティ・ダイアログ・ボックスに手作業で入力することも，すべてのセキュリティ情報を含んでいるポリシー・ファイルを参照することもできます。samlPolicy.config に基づいた独自のポリシー・ファイルを作成することもできます。

ポリシー・ファイルに変更を加えて，ユーザ名や証明書情報などのセキュリティ・パラメータの値を含めることができます。スクリプトに SAML セキュリティ・ステップを追加するとき，セキュリティ引数の値を明示的に指定すると，その値はポリシー・ファイルの値に優先します。

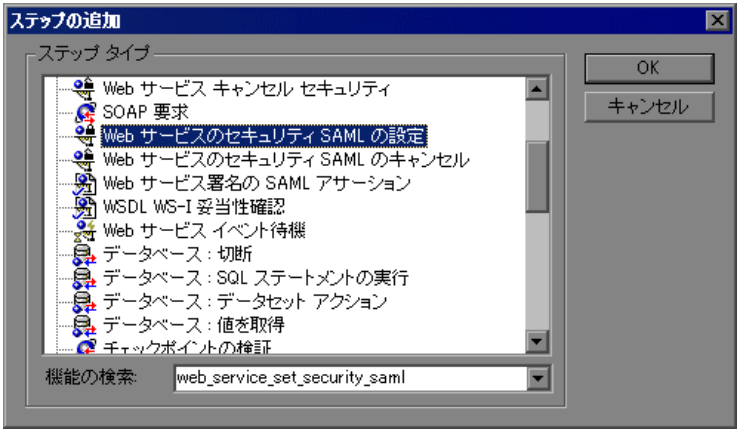
標準設定のポリシー・ファイルに変更を加える場合は，新しいポリシー・ファイルをスクリプト・フォルダにコピーすることをお勧めします。スクリプトを別のマシンで実行したり，LoadRunner Controller から呼び出したりしても，ポリシー・ファイルがスクリプトとともに残るように，カスタムのポリシー・ファイルには必ず **.config** 拡張子を付けて保存します。

SAML ポリシー・ファイルの詳細については，MSDN Web サイトの SAML STS の例を参照してください。SAML Federation の動作をエミュレートする場合は，**samlFederationPolicy.config** ファイルをデータ・フォルダからスクリプト・フォルダにコピーして，これをポリシー・ファイルとして指定します。

SAML セキュリティを追加するには，次の手順を実行します。

1 スクリプトの適切な場所をクリックします。スクリプト全体にセキュリティを適用するには，スクリプトの先頭にカーソルを置きます。

2 ［**挿入**］＞［**新規ステップ**］を選択し，［ステップの追加］ダイアログ・ボックスを開きます。



3 SAML セキュリティを追加するには，［**Web サービスのセキュリティ SAML の設定**］を選択します。

必要な情報を入力します。このダイアログ・ボックスに値を入力すると，それらの値はポリシー・ファイルの値に優先します。Issuer URL（**STS URL** とも呼ばれます）を指定する必要があります。

別のポリシー・ファイルを使用するには，使用するポリシー・ファイルを［**Policy File**］ボックスに指定します。フル・パスを指定するか，スクリプトの場所を起点にして，ファイルの相対パスを指定します。

4 SAML セキュリティを削除するには，**Web Services Cancel Security SAML** ステップを選択します。セキュリティが，指定のポイント以降取り消されます。

これらの関数の詳細については，『**Online Function Reference**』（英語版）（［**ヘルプ**］＞［**関数リファレンス**］を選択するか，関数の上で **F1** キーを押します）を参照してください。

web_service_set_security の使用例

この項では，いくつかの一般的なセキュリティ・シナリオについて説明します。

Username トークンを使用した認証

次の例では，メッセージ・レベルの username/password トークン（username トークン）の送信について説明します。ここでは，ユーザ名は John であり，パスワードは 1234 です。

```
    web_service_set_security(
        SECURITY_TOKEN, "Type=USERNAME","LogicalName=myToken",
"UserName=John", "Password=1234", "PasswordOptions=SendPlainText",
"Add=True",
        LAST);
```

X.509 Certificate を使用した特定要素の署名

メッセージで，特定の要素のみに署名できます。次の例では，XPATH 表現を使用している特定の要素に署名します。

```
    web_service_set_security(
        SECURITY_TOKEN, "Type=X509","LogicalName=myCert", "StoreName=My",
"IDType=SubjectName", "IDValue=CN=myCert", "StoreLocation=CurrentUser",
"Add=True",
        MESSAGE_SIGNATURE, "UseToken=myCert", "TargetPath=//*[local-
name(.)='someElement' and namespace-uri(.)='http://myNamespace']",
        LAST);
```

X.509 Certificate を使用した署名

次の例に，デジタル署名のために X.509 証明書を使用するスクリプトを示します。

```
    web_service_set_security(
        SECURITY_TOKEN, "Type=X509","LogicalName=myCert", "StoreName=My",
"IDType=SubjectName", "IDValue=CN=myCert", "StoreLocation=CurrentUser",
"Add=True",
        MESSAGE_SIGNATURE, "UseToken=myCert",
        LAST);
```

証明書は Windows 証明書ストアにインストールする必要があります。前述の例では，実際のストア名，ストア位置，および証明書のサブジェクト名を設定する必要があります。

証明書を使用した暗号化

次の例では，サービスの X.509 証明書を使ってメッセージを暗号化します。

```
    web_service_set_security(
        SECURITY_TOKEN, "Type=X509","LogicalName=serviceCert",
"StoreName=My", "IDType=SubjectName", "IDValue=CN=serviceCert",
"StoreLocation=CurrentUser", "Add=False",
        ENCRYPTED_DATA, "UseToken=serviceCert",
        LAST);
```

X.509 証明書の詳細を指定したら，メッセージで特定の XPATH を暗号化できます。

Subject Key Identifier を生成したいので，Add 値を **False** に設定します。詳細については，161 ページ「SubjectKeyIdentifier の使い方」を参照してください。

Username トークンおよび暗号化と X.509 Certificate を使用した認証

次の例では，username トークンをサービスに送信し，サーバの X.509 証明書を使って暗号化します。

```
web_service_set_security(
        SECURITY_TOKEN, "Type=X509","LogicalName=serviceCert",
"StoreName=My", "IDType=SubjectName", "IDValue=CN=serviceCert",
"StoreLocation=CurrentUser", "Add=True",
        SECURITY_TOKEN, "Type=USERNAME","LogicalName=myUser",
"UserName=John", "Password=1234", "PasswordOptions=SendPlainText",
"Add=True",
        ENCRYPTED_DATA, "UseToken=serviceCert", "TargetToken=myUser",
        LAST);
```

UseToken および **TargetToken** プロパティは，使用するトークンと暗号化するトークンを示します。それらの値は，トークンの **LogicalName** プロパティを参照します。

メッセージの暗号化と署名

この例では，秘密鍵を使ってメッセージに署名し，サービスの公開鍵を使って暗号化する方法を示します。

```
web_service_set_security(
        SECURITY_TOKEN, "Type=X509","LogicalName=myCert", "StoreName=My",
"IDType=SubjectName", "IDValue=CN=myCert", "StoreLocation=CurrentUser",
"Add=True",
        SECURITY_TOKEN, "Type=X509","LogicalName=serverToken",
"StoreName=My", "IDType=SubjectName", "IDValue=CN=serverCert",
"StoreLocation=CurrentUser", "Add=False",
        MESSAGE_SIGNATURE, "UseToken=myCert",
        ENCRYPTED_DATA, "UseToken=serverCert",
    LAST);
```

ハッシュを使用した X.509 証明書の参照

場合により，サブジェクト名を使用した証明書を参照できない可能性があります。この例では，固有のハッシュを使用してい証明書を参照しする方法を示します。

```
web_service_set_security(
   SECURITY_TOKEN, "Type=X509","LogicalName=serviceCert", "StoreName=My",
   "IDType=Base64KeyID", "IDValue=pOl0+1iuotKLlO91nhjDg5reEw0=",
   "StoreLocation=CurrentUser", "Add=False",
   ENCRYPTED_DATA, "UseToken=serviceCert",
 LAST);
```

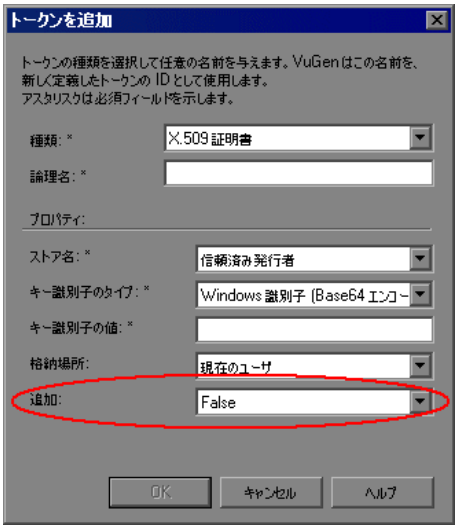
セキュリティのカスタマイズ

以下の項では，Web サービス・セキュリティに共通する特殊な場合を設定する方法について説明します。

SubjectKeyIdentifier の使い方

標準設定では，Service Test は定義された X.509 トークンをすべて SOAP エンベロープに追加し，バイナリ・トークンとして参照します。また，メッセージからトークンを除外し，**SubjectKeyIdentifier** を使用してトークンを参照することもできます。これは暗号化に使用するトークンでは一般的です。

これを実行するために，UI でトークンを追加する場合は，Add オプションとして **False** を選択します。



あるいは，この設定をスクリプトで行うことができます。

```
SECURITY_TOKEN, "Type=X509","LogicalName=myToken", "StoreName=My",
"IDType=SubjectName", "IDValue=CN=myCert", "StoreLocation=CurrentUser",
"Add=False",
```

SKI（Subject Key Identifier）を使用する場合は，163 ページ「WS-Security のカスタマイズ」で説明しているように，**useRFC3280** 設定も変更する必要があります。

ユーザ名のカスタマイズ

新しい username トークンを追加すると，エディタに **web_service_set_security** が次のように表示されます。

```
    web_service_set_security(
        SECURITY_TOKEN, "Type=USERNAME","LogicalName=myToken",
"UserName=john", "Password=1234", "PasswordOptions=SendPlainText", "Add=True",
        LAST);
```

カスタマイズに使用でき，ユーザ・インタフェースに使用できない 2 つの付加的な設定があります。

名前	意味	指定可能な値
IsNonceIncluded	username トークンに nonce を含める	True（標準設定）または False
TimestampFormat	username トークンにタイムスタンプを含める。その場合，形式は任意	➤ **None**：タイムスタンプなし ➤ **Full**：<timestamp> 要素と <created> および <expired> 内部要素 ➤ **Created**：（標準設定）<created> 要素のみ

これらのオプションは次のようにして **web_service_set_security** に追加します。

```
web_service_set_security(
        SECURITY_TOKEN, "Type=USERNAME","LogicalName=myToken",
"UserName=John", "Password=1234",  "PasswordOptions=SendPlainText",
"IsNonceIncluded=true", "TimestampFormat=Full", "Add=True",
        LAST);
```

暗号化のカスタマイズ

暗号化をカスタマイズするには，要素を暗号化する方法（要素全体またはその内容を暗号化する）を指示します。これはユーザ名などのトークンを暗号化するときに一般的です。

次の設定で，正確な暗号化タイプを決定できます。

名前	意味	指定可能な値
EncryptionType	暗号化する対象	➤ 要素 ➤ 内容（標準設定）

このオプションは次のようにして **web_service_set_security** に追加します。

```
web_service_set_security(
...
ENCRYPTED_DATA, "UseToken=myToken", "TargetToken=myOtherToken",
"EncryptionType=Element",
LAST);
```

WS-Security のカスタマイズ

Service Test で暗号化に使用するアルゴリズム変更したり，ほかの低レベル・セキュリティの詳細を変更する必要がある場合もあります。

これらの項目のいずれかを変更するには，テキスト・エディタで **%Service Test%/bin/mmdrv.exe.config** ファイルを開きます。
このファイルに <**microsoft.web.services2**> 要素が含まれていない場合は，下記のように追加します。

```
<configuration>
...
    <microsoft.web.services2>
        <security>
        <x509 storeLocation="CurrentUser" allowTestRoot="true" useRFC3280="true" />
            <binarySecurityTokenManager valueType="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509v3">
            <sessionKeyAlgorithm name="TripleDES" />
            <keyAlgorithm name="RSA15" />
        </binarySecurityTokenManager>
        </security>
    </microsoft.web.services2>
...
<configuration>
```

必要に応じて，要素値を設定します。

名前	意味	指定可能な値
verifyTrusy	送受信する x.509 証明書の有効性をチェックするかどうか	➤ True（標準設定） ➤ False
sessionKeyAlgorithm	セッション対象鍵でメッセージを暗号化するのに使用するアルゴリズム	➤ AES128 ➤ AES192 ➤ AES256 ➤ TripleDES
keyAlgorithm	公開鍵でセッション鍵を暗号化するのに使用するアルゴリズム	➤ RSA15 ➤ RSAOAEP
useRFC3280	相互運用可能だが，Windows 固有ではない subject key identifier を生成するかどうか	➤ True ➤ False（標準設定）

第 10 章

Web サービス－高度なセキュリティと WS 仕様

Service Test では，スクリプトをカスタマイズして付加的な WS standards をサポートできます。これらには，WCF およびほかの WS - ＜ **spec_name** ＞ 仕様に実装されている WS standards が含まれています。

本章の内容

以下の情報は，Web サービスと SOA Vuser スクリプトのみを対象とします。

高度なセキュリティと WS 仕様

Service Test では，高度なセキュリティと WS-Specifications を利用する Web サービスをテストできます。このようなサービスは，WCF（Windows Communication Foundation），Metro（WSIT），および Axis2 など，さまざまなプラットフォームで作成できます。WCF サービスについては，Service Test でも独自の標準とトランスポートをサポートしています。

このサポート有効にするには，セキュリティ・シナリオを設定します。それぞれのシナリオは，Web サービス呼び出しとともに用いられる一般的な環境を表します。Service Test には，よく使用される組み込みセキュリティ・シナリオがいくつか用意されています。シナリオの設定は各サービスに個別に適用されます。

組み込みシナリオの場合，ユーザ・インタフェースで必要な識別情報を提供できます。セキュリティ，トランスポート，プロキシ，その他の詳細設定をカスタマイズできます。

使用環境に対応するシナリオがない場合は，汎用のカスタム・シナリオを使用できます。

シナリオの選択に関する「入門」ガイドとして，192 ページ「ヒントとガイドライン」を参照してください。

セキュリティ・モデルの選択

Service Test では，Web サービス呼び出しのセキュリティを設定する 2 つのモデル，**レガシ**および**シナリオ**をサポートしています。本章では，シナリオ・セキュリティ・モデルについて説明します。レガシ・モデルでは，**Web Service Set Security** ステップ，すなわち **web_service_set_security** 関数を手作業で追加します。レガシ・モデルについては，第 9 章「Web サービスーセキュリティ」を参照してください。

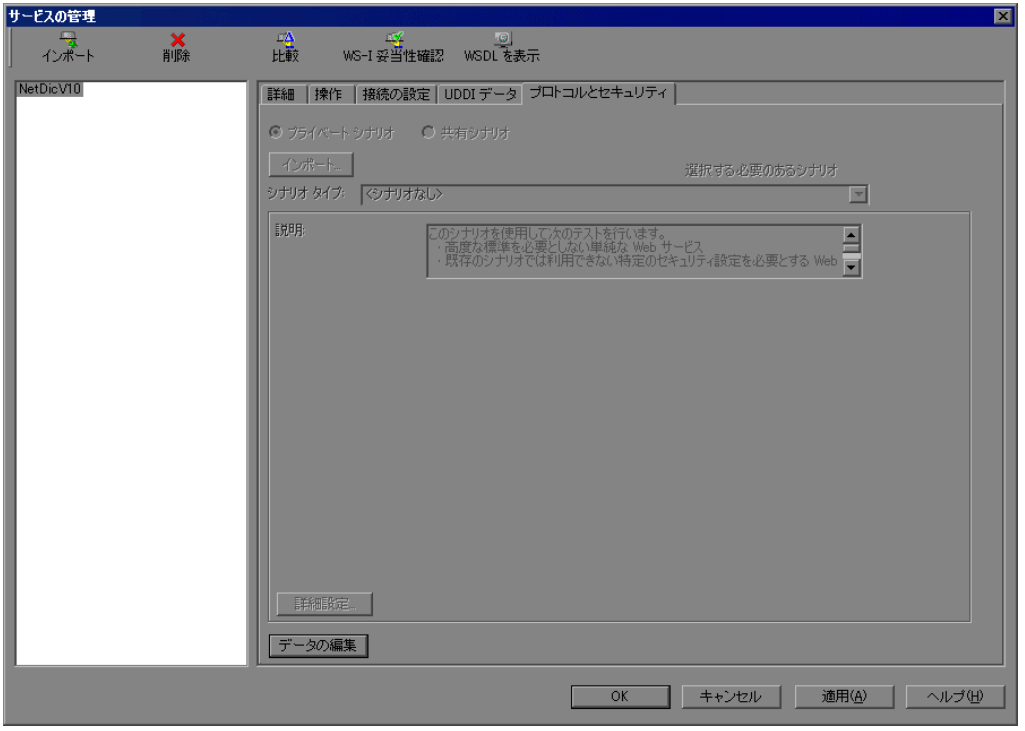
次の表に，各モデルの使用に関する注意事項を示します。

レガシ・モデル	シナリオ・ベース・モデル
レガシ・モデルを使用しているスクリプトを使用する	WCF サービスをテストする
.NET 2.0，Axis，またはほかの古いツールキットなどのフレームワークで作成されたサービスをテストする	Axis2 や Metro（WSIT）など，新しいフレームワークで作成されたサービスをテストする
WS-Security トークンに対する低レベル制御が必要になる	サービスで WS-SecureConversation や WS-Trust など，高度な仕様を使用する
新しいモデルを使用するのが難しいので，レガシ関数の適切な機能を見つける	レガシ・モデルを使用するのが難しいので，新しいモデルのより適切な機能を見つける

セキュリティ・シナリオの設定

スクリプトごとに，サービス・レベルにセキュリティ・シナリオを割り当てます。すなわち，スクリプトの各サービスごとに異なるセキュリティ・シナリオを割り当てます。

［サービスの管理］ウィンドウには，個別サービスのセキュリティ・シナリオを作成，編集するインタフェースがあります。このインタフェースには，［**プロトコルとセキュリティ**］タブからアクセスします。



セキュリティ・シナリオ・エディタを使用すると，スクリプトから独立してシナリオを作成することもできます。シナリオは個人的利用または他者との共同作業のために，共有の場所に保存できます。

サービスのセキュリティ・シナリオ設定

特定のサービスにセキュリティ・シナリオを割り当てるには，［サービスの管理］ウィンドウを使います。［**プロトコルとセキュリティ**］タブには，個別サービスのセキュリティ・シナリオを作成，表示するインタフェースがあります。

シナリオは，次の 3 つの方法で選択できます。

- **プライベート・シナリオ**：組み込みシナリオのいずれかを選択し，Web サービスに合わせてカスタマイズして，新しいシナリオを作成します。
- **インポート・シナリオ**：以前作成したシナリオを利用します。このシナリオは編集可能であり，誰かが元のシナリオを変更しても，影響はありません。
- **共有シナリオ**：遠隔地またはファイル・システムから，別のユーザが設定したセキュリティ・シナリオをロードします。このシナリオの設定は，［サービスの管理］ウィンドウから編集できません。誰かがシナリオを編集すると，使用環境に影響を及ぼします。通常，このオプションは，製品をしばらく使用して，シナリオ・ファイルを保存した後で使います。

特定サービスのセキュリティ・シナリオを作成するには，次の手順を実行します。

1 ［**サービスの管理**］をクリックします。左側の表示枠で，セキュリティ・シナリオを設定するサービスを選択します。必要ならば，47 ページ「サービスのインポート」で説明しているように，サービスをインポートします。

2 ［**プロトコルとセキュリティ**］タブを選択し，［**データの編集**］ボタンをクリックします。セキュリティ・シナリオ・データ・ダイアログ・ボックスが表示されます。

3 シナリオをロードします。セキュリティ・シナリオを使い始めるときは，まず最初のメソッドを使用します。シナリオをいくつか作成したら，ほかのメソッドを使用できます。

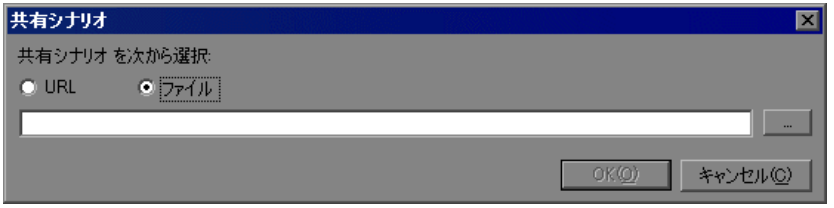
a 現在のサービスの組み込みセキュリティ・シナリオを選択するには，［**プライベート シナリオ**］を選択します。

［シナリオ タイプ］ボックスで，シナリオを選択します。シナリオ・タイプについては，173 ページ「シナリオ・タイプ」を参照してください。

シナリオに必要な値を指定します。詳細については，175 ページ「シナリオ情報の指定」を参照してください。

b 変更できる既存のシナリオを使用するには，［**プライベート シナリオ**］を選択します。［**インポート**］をクリックします。［共有シナリオ］ダイアログ・ボックスで，保管されたシナリオを選択します。必要ならば，175 ページ「シナリオ情報の指定」で説明しているように設定を変更します。

c 変更するオプションのない既存のシナリオを使用するには，［**共有シナリオ**］を選択します。［参照］ボタンを使って，［共有シナリオ］ダイアログ・ボックスを開き，保管されたシナリオを選択します。



［**OK**］をクリックします。これで，サービスがセキュリティ・シナリオと一緒にロードされます。誰かがソースのシナリオ・ファイルを変更すると，スクリプトに影響を及ぼします。

4 ［**詳細設定**］をクリックして，プロキシ，エンコーディング，その他の詳細設定（オプション）を行います。ほとんどのシナリオでは，標準設定が最適です。これらの設定については，182 ページ「詳細設定」を参照してください。

5 ［**OK**］をクリックしてこのダイアログ・ボックスを閉じ，スクリプトを保存します。

コラボレーション用セキュリティ・シナリオの保存

ユーザはセキュリティ・シナリオをカスタマイズでき，他者に利用できるようにします。セキュリティ・シナリオ・エディタを使うと，設定をカスタマイズして，シナリオ・ファイルに保存できます。

新しいセキュリティ・シナリオを作成するには，次の手順を実行します。

1 ［**SOA ツール**］＞［**セキュリティ シナリオ エディタ**］を選択します。

2 ［**シナリオ タイプ**］を選択して，関連情報を入力します。

3 有効な場合は，182 ページ「詳細設定」で説明しているように，［**詳細設定**］をクリックして，プロキシ，エンコーディング，その他の設定を行います。［**OK**］をクリックし，ダイアログ・ボックスを閉じます。

4 新しいシナリオの場合，［**名前を付けて保存**］をクリックし，シナリオ・ファイルのファイル名とパスを指定します。

保管された既存のシナリオを編集するには，次の手順を実行します。

1 ［**SOA ツール**］＞［**セキュリティ シナリオ エディタ**］を選択します。

2 ［**開く**］ボタンをクリックして，既存のシナリオ・ファイルを参照します。

3 必要に応じて，シナリオ設定を変更します。

4 ［**保存**］または［**名前を付けて保存**］ボタンをクリックします。

5 有効な場合は，［**詳細設定**］をクリックして，プロキシ，エンコーディング，その他の詳細設定を行います。詳細については，182 ページ「詳細設定」を参照してください。

6 ［**OK**］をクリックして，セキュリティ・シナリオ・エディタを閉じます。

シナリオ・タイプ

シナリオには，Web サービスの設定が記述されています。エンコーディング，識別属性，プロキシなどの情報が含まれています。Service Test には，各シナリオの設定が行えるセキュリティ・シナリオ・エディタが用意されています。

サービスに最適なシナリオを決めるには，下の表を参照してください。選択するシナリオがわからない場合は，サービスをテストできる **Custom Binding** シナリオの使用をお勧めします。詳細については，178 ページ「カスタム・バインド」を参照してください。

次の表に，組み込みシナリオを表示し，それらを使用する場合について説明します。

シナリオ名	使用する場合
< シナリオなし >	➤ 標準設定 : 特別なセキュリティ / プロトコル設定の必要がない場合は，この値をそのままにします。詳細な標準が必要ない単純な Web サービス。 ➤ 第 9 章「Web サービス－セキュリティ」で説明しているレガシ・セキュリティ・モデルを使用するスクリプト ➤ 既存シナリオのいずれかで利用できず，特定のセキュリティ設定を必要とする Web サービス ➤ 組み込みシナリオを選択し，再生中に問題に直面すると，シナリオが必要とされず，問題がほかにある可能性があります。値を＜**シナリオなし**＞にリセットします。
プレーンな SOAP	➤ 詳細な標準を必要としない Web サービス ➤ WS-Addressing バージョンを指定する必要がある Web サービス
MTOM	➤ MTOM が有効な Web サービス ➤ WS-Addressing バージョンを指定する必要がある Web サービス

次の表に，WCF を利用する Web サービスのシナリオを示します。WSHttpBinding ベースのシナリオは，クライアントが認証される方法に従ってサーバに分割されます。たとえば，クライアントがユーザ名とパスワードをサーバに提示する場合は，［**ユーザ名 (メッセージ保護)**］シナリオを選択します。ユーザ・インタフェースでは，必要に応じてユーザ名または証明書の形式で識別情報を提供できます。

WCF シナリオ名	使用する場合
WSHttpBinding － 認証なし	➤ クライアントが暗号化のためにサーバの X.509 証明書を使用する ➤ クライアントが認証されない ➤ 通信でセキュアな対話や MTOM など，詳細な標準を利用できる
WSHttpBinding － Windows 認証	➤ クライアントとサーバが Windows 認証を使用する ➤ セキュリティが Kerberos または SPNEGO ネゴシエーションに基づいている ➤ 通信でセキュアな対話や MTOM など，詳細な標準を利用できる
wsHttpBInding － 証明書	➤ クライアントが暗号化のためにサーバの X.509 証明書を使用する ➤ クライアントが署名に X.509 証明書を使用する ➤ 通信でセキュアな対話や MTOM など，詳細な標準を利用できる
WSHttpBinding － ユーザ名 (メッセージ保護)	➤ クライアントが暗号化のためにサーバの X.509 証明書を使用する ➤ クライアントをユーザ名とパスワードで認証する ➤ 通信でセキュアな対話や MTOM など，詳細な標準を利用できる
WSHttpBinding － ユーザ名 (トランスポート保護)	➤ SSL を有効にする ➤ クライアントをユーザ名とパスワードで認証する ➤ 通信でセキュアな対話や MTOM など，詳細な標準を利用できる
WSFederationHttpBinding	➤ クライアントが定義済みのシナリオを使って，STS に対して認証を受ける ➤ クライアントが STS から得たトークンを使用し，サーバに対して認証を受ける
カスタム・スタンバイ	➤ WS-* standards を使用する Web サービス ➤ 任意の設定の WCF サービス

シナリオ情報の指定

この項では，各セキュリティ・シナリオに必要な値について説明します。

< シナリオなし >

セキュアなシナリオを使用しないので，任意の値を指定する必要はありません。

プレーンな SOAP

このタイプのシナリオに関して，サービスで WS-Addressing 使用する場合は，バージョンを指定します。

MTOM

MTOM タイプのシナリオに関して，サービスで WS-Addressing 使用する場合は，バージョンを指定します。

認証なし

このシナリオでは，クライアントがサーバの証明書を使って，メッセージを暗号化します。クライアントの認証はありません。

次の設定から 1 つだけ指定します。

- **サービス資格情報をネゴシエーションする**：Web サービスの証明書をサーバとネゴシエートします。
- **サービス証明書を指定する**：サービス証明書を参照します。詳細については，180 ページ「証明書の選択」を参照してください。このオプションを選択すると，［**サービス資格情報をネゴシエーションする**］オプションは使用できません。

DNS 情報を提供します。

- **期待されるサーバの DNS**：DNS に関するサーバの期待識別属性。これには **localhost**，IP アドレス，またはサーバ名を設定できます。また，証明書が発行された一般名でもかまいません。

Windows 認証

この WCF シナリオでは Windows 認証を使用します。

SPN または **UPN** 識別属性に関して，サーバの期待識別属性を宣言します。カスタマイズされておらず，標準設定を使用する WCF サービスをテストする場合は，このタイプのシナリオを使用します。

証明書認証

この WCF WSHttpBinding シナリオでは，クライアントはサーバの X.509 証明書を使ってメッセージと署名用の証明書を暗号化します。

次の設定から 1 つだけ指定します。

- **サービス資格情報をネゴシエーションする**：Web サービスの証明書をサーバとネゴシエートします。
- **サービス証明書を指定する**：サービス証明書を参照します。詳細については，180 ページ「証明書の選択」を参照してください。このオプションを選択すると，［**サービス資格情報をネゴシエーションする**］オプションは使用できません。

DNS 情報を提供します。

- **期待されるサーバの DNS**：DNS に関するサーバの期待識別属性。これには **localhost**，IP アドレス，またはサーバ名を設定できます。また，証明書が発行された一般名でもかまいません。

ユーザ名（メッセージ保護）

この WCF WSHttpBinding シナリオでは，クライアントはサーバの X.509 証明書を使ってメッセージを暗号化し，認証に使用するユーザ名とパスワードを送信します。

次の設定を指定します。

- **ユーザ名・パスワード**：クライアントのユーザ名およびパスワード証明書。

次の設定から 1 つだけ指定します。

- **サービス資格情報をネゴシエーションする**：Web サービスの証明書をサーバとネゴシエートします。

- **サービス証明書を指定する**：サービス証明書を参照します。詳細については，180 ページ「証明書の選択」を参照してください。このオプションを選択すると，［**サービス資格情報をネゴシエーションする**］オプションは使用できません。

DNS 情報を提供します。

- **期待されるサーバの DNS**：DNS に関するサーバの期待識別属性。これには **localhost**，IP アドレス，またはサーバ名を設定できます。また，証明書が発行された一般名でもかまいません。

ユーザ名（トランスポート保護）

この WCF WSHttpBinding s シナリオで SSL を有効にし，メッセージ・レベルでユーザ名とパスワードを使ってクライアントを認証します。

次の設定を指定します。

- **ユーザ名・パスワード**：クライアントのユーザ名およびパスワード証明書。

フェデレーション

WSFederationHttpBinding シナリオでは，クライアントは STS（セキュリティ・トークン・サービス）から認証を受けて，トークンを取得します。クライアントはこのトークンを使用して，アプリケーション・サーバから認証を受けます。

そのため，2 つのバインドが必要とされます。1 つは STS に対するバインドで，もう 1 つはアプリケーション・サーバに対するバインドです。

まず，セキュリティ・シナリオ・エディタを使って，STS バインドを定義します。詳細については，171 ページ「コラボレーション用セキュリティ・シナリオの保存」を参照してください。アプリケーション・サーバに対するバインドを設定するときは，［**参照ファイル**］ボックスでそのファイルを指定します。

フェデレーション・シナリオでは，次のサーバ情報を指定します。

- **トランスポート**：HTTP または HTTPS
- **エンコード**：テキストまたは MTOM

Federation シナリオでは，次のセキュリティ情報を指定します。

- **認証モード**：IssuedToken，IssuedTokenForCertificate，IssuedTokenForSslNegotiated，IssuedTokenOverTransport，または SecureConversation
- **ブートストラップ・ポリシー**：IssuedToken，IssuedTokenForCertificate，IssuedTokenForSslNegotiated，または IssuedTokenOverTransport

Federation シナリオでは，次の識別情報を指定します。

- **サーバ証明書**：サーバ証明書を参照します。詳細については，180 ページ「証明書の選択」を参照してください。
- **期待される DNS**：DNS に関するサーバの期待識別属性。これには **localhost**，IP アドレス，またはサーバ名を設定できます。

Federation シナリオでは，次の STS（セキュリティ・トークン・サービス）情報を指定します。

- **発行者アドレス**：STS の発行者のアドレス。これには **localhost**，IP アドレス，またはサーバ名を設定できます。
- **参照ファイル**：STS（セキュリティ・トークン・サービス）にコンタクトするバインドを参照するファイル

カスタム・バインド

カスタム・バインド・シナリオでは，最高レベルのカスタマイズができます。このシナリオは WCF **customBinding** に基づいているため，WS - ＜ **spec_name** ＞ 仕様を使用する Java など，ほかのプラットフォームのサービスと一緒に，ほとんどの WCF サービスをテストできます。

カスタム・バインド・シナリオを利用して，定義済みのセキュリティ・シナリオに適合しないカスタム・シナリオを設定します。

カスタム・バインド・シナリオでは，次のサーバ情報を指定します。

- **トランスポート**：HTTP，HTTPS，TCP，または NamedPipe
- **エンコード**：Text，MTOM，または WCF Binary

次のセキュリティ情報を指定します。

- **認証モード**：None，AnonymousForCertificate，AnonymousForSslNegotiated，CertificateOverTransport，Kerberos，KerberosOverTransport，MutualCertificate，MutualSslNegotiated，SecureConversation，SspiNegotiated，UserNameForCertificate，UserNameForSslNegotiated，UserNameOverTransport，または SspiNegotiatedOverTransport
- **ブートストラップ・ポリシー**：SecureConversation タイプの認証では，次のブートストラップ・ポリシーを指定します。AnonymousForCertificate，AnonymousForSslNegotiated，CertificateOverTransport，Kerberos，KerberosOverTransport，MutualCertificate，MutualSslNegotiated，SspiNegotiated，UserNameForCertificate，UserNameForSslNegotiated，UserNameOverTransport，または SspiNegotiatedOverTransport
- **ネット・セキュリティ**：ネットワーク・セキュリティ。［なし］，［Windows ストリーム セキュリティ］，または［SSL ストリーム セキュリティ］を選択します。HTTP トランスポートを使用するサービスでは，標準設定値，**なし**のままにします。HTTP で SSL を有効にするには，HTTPS トランスポートを選択します。

  Web サービスで信頼できるメッセージングを使用する場合は，そのオプションを有効にし，［**要求済み**］または［**未要求**］を選択します。

  識別属性

  セキュリティ設定では，クライアントもしくはサーバ，または両方の識別詳細を指定する必要があります。

  クライアントの識別詳細には，ユーザ名 / パスワードや **X.509** 証明書などがあります。

  識別に情報に関しては，サービスの必要に応じて，1 つ以上の認証詳細を設定します。

  ユーザ名，**パスワード**，**サーバ証明書**，または **Client 証明書**などです。証明書の選択については，180 ページ「証明書の選択」を参照してください。

  一部のシナリオでは，DNS，SPN，または UPN 識別属性に関して，サーバの期待識別属性を宣言する必要があります。

- **DNS**：サーバの名前を設定するか，localhost を使用します。
- **SPN**：SPN 識別属性は，domain¥machine 形式で設定します。

➤ **UPN**：UPN 識別属性は，user@domain 形式で設定します。

基本値を設定したら，182 ページ「詳細設定」で説明しているように，詳細属性を設定できます。

証明書の選択

この項では，証明書の選択方法について説明します。

証明書は，ファイルまたは Windows ストアから選択できます。

ファイルに保管された証明書

この場合，証明書はファイルに保管されています。

ファイルから証明書を選択するには，次の手順を実行します。

1 ［クライアント証明書］または［サーバ証明書］ボックスの横にある［参照］ボタンをクリックします。

2 ［**ファイル**］を選択します。［ファイル］ボックスの右にある［参照］ボタンをクリックして，証明書ファイルを見つけます。

3 必要に応じて，秘密鍵のパスワードを指定します。

4 ［**選択**］をクリックします。これで，アプリケーションが証明書ファイルを参照します。

Windows ストアからの証明書

この場合，証明書は Windows ストアにあります。ダイアログ・ボックスに，証明書を検索するメカニズムが表示されます。

Windows ストアから証明書を選択するには，次の手順を実行します。

1 ［クライアント証明書］または［サーバ証明書］ボックスの横にある［参照］ボタンをクリックします。

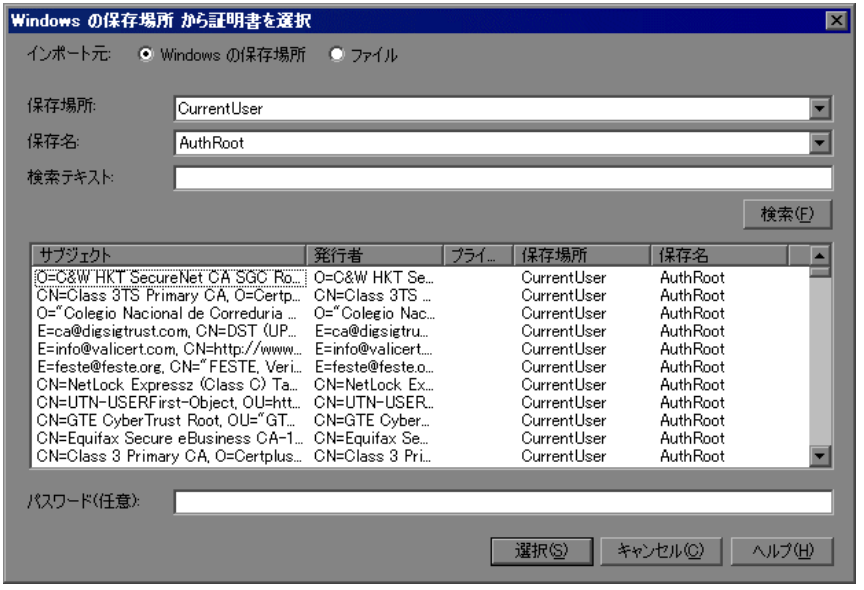
2 ［**Windows の保存場所**］を選択します。［ファイル］ボックスの右にある［参照］ボタンをクリックして，証明書ファイルを見つけます。

3 ストアの場所を選択します。**All**，**CurrentUser**，または **LocalMachine**（オプション）です。

4 ドロップダウンメニューから［**保存名**］ボタンを選択します（オプション）。

5 すべての証明書を検索するには，［**検索テキスト**］ボックスを空のままにします。特定の証明書を検索するには，証明書名のサブ文字列を指定します。

6 ［**検索**］をクリックすると，ストアにある証明書のリストが生成されます。



7 必要に応じて，秘密鍵のパスワードを指定します。

8 リストで証明書を選択し，［**選択**］をクリックします。これで，アプリケーションが選択した証明書を参照します。

詳細設定

この項では，詳細な シナリオ設定について説明します。これらの設定を使用すると，［Encoding］，［Advanced Standards］，［Security］，または［HTTP and Proxy］の領域でセキュリティ・シナリオをカスタマイズできます。

すべての設定がすべてのシナリオに関連しており，その一部はシナリオに応じて無効または非表示にできます。

エンコード

［エンコード］タブでは，メッセージに使用するエンコーディングのタイプ，［**Text**］，［**MTOM**］，または［**Binary**］を指定できます。標準設定は［**Text**］エンコーディングです。

これらの各エンコーディング方式に関しては，WS-Addressing のバージョンを選択できます。

- None
- WSA 1.0
- WSA 04/08

高度な標準

このタブでは，Reliable Messaging など，詳細な WS- standards を設定できます。

サービスで［**WS-ReliableMessaging**］仕様を実装する場合は，［**信頼できるメッセージ**］オプションを有効にして，次のオプションを設定します。

- **信頼できるメッセージが要求される**：信頼できるセッションを整理するかどうかを指定します。
- **信頼できるメッセージングのバージョン**：WSReliableMessagingFebruary2005 または WSReliableMessaging11

セキュリティ

高度なセキュリティ設定は，**WS-Security** 仕様に対応しています。

WCF WSHttpBinding をベースにしたセキュリティ・シナリオでは，次の設定を指示できます。

- **セキュア セッションを有効にする**：WS-SecureConversation 標準を使用するセキュリティ・コンテキストを設定します。
- **サービス資格情報をネゴシエーションする**：サービスのセキュリティをネゴシエートする WCF プロパティ・ネゴシエーションを可能にします。

WSHttpBinding，**Custom Binding**，または **WSFederationHttpBinding** WCF タイプのシナリオでは，標準のアルゴリズム・スイートと保護レベルを設定できます。

属性	意味	指定可能な値
既定のアルゴリズム スイート	対称 / 非対称暗号化に使用するアルゴリズム。 これらの値は WCF の SecurityAlgorithmSuite 設定からもたらされる。	➤ Basic128 ➤ Basic128Rsa15 ➤ Basic128Sha256 ➤ Basic128Sha256Rsa15 ➤ Basic192 ➤ Basic192Rsa15 ➤ Basic192Sha256 ➤ Basic192Sha256Rsa15 ➤ Basic256 ➤ Basic256Rsa15 ➤ Basic256Sha256 ➤ Basic256Sha256Rsa15 ➤ TripleDes ➤ TripleDesRsa15 ➤ TripleDesSha256 ➤ TripleDesSha256Rsa15
保護レベル	SOAP Body を暗号化 / 署名する	None，Sign，および EncryptAndSign（標準設定）

カスタム・バインディング または **WSFederationHttpBinding** WCF タイプのシナリオでは，セキュリティ設定をさらに詳細にカスタマイズできます。次の表に，そのオプションと値を示します。

属性	意味	指定可能な値
メッセージ保護の順序	署名と暗号化の順序	➤ SignBeforeEncrypt ➤ SignBeforeEncrypt-AndEncryptSignature ➤ EncryptBeforeSign
メッセージ・セキュリティのバージョン	WS-Security のセキュリティ・バージョン	現行バージョンのリスト
セキュリティ・ヘッダのレイアウト	メッセージヘッダのレイアウト	➤ Strict ➤ Lax ➤ LaxTimeStampFirst ➤ LaxTimeStampLast
キー・エントロピー・モード	セキュリティ鍵のエントロピー・モード	➤ Client Entropy ➤ Security Entropy ➤ Combined Entropy

次のオプションを有効または無効にできます。

- **派生キーを要求する**：派生鍵が必要かどうかを指定します。
- **セキュリティ・コンテキストの取り消しを要求**：このオプションを無効にすると，**WS-SecureConversation** セッション（有効な場合）でステートフル・セキュリティ・トークンが使用されます。
- **タイムスタンプを含める**：ヘッダのタイムスタンプが含まれます。
- **応答時にシリアル化された署名トークンを許可**：応答でシリアル化トークンを送信できるようにします。
- **署名の確認を要求**：応答で署名確認を送信するようサーバに指示します。

X.509 証明書では，次の項目の値を指定できます。

属性	意味	指定可能な値
X509 包含モード	X509 証明書を含める場合	➤ Always to Recipient ➤ Never ➤ Once ➤ AlwaysToInitiator
X509 リファレンス スタイル	証明書を参照する方法	➤ Internal ➤ External
X509 は派生キーを要求する	X509 証明書で派生鍵が必要な場合	➤ Enable - Yes ➤ Disable - No
X509 キー識別子句のタイプ	X509 キーを識別するのに使用する文節のタイプ	➤ 任意 ➤ Thumbprint ➤ IssuerSerial ➤ SubjectKeyIdentifier ➤ RawDataKeyIdentifier

HTTP & プロキシ

このタブでは，テストに関する HTTP および Proxy 情報を設定します。

HTTP (S) トランスポート

次の表で，HTTP(S) Transport オプションについて説明します。

オプション	意味	指定可能な値
転送モード	要求 / 応答の転送方式	Buffered，Streamed，StreamedRequest，StreamedResponse
Cookie を許可する	クッキーを有効にする	有効 / 無効
Keep-Alive を有効にする	キープアライブ接続を有効にする	有効 / 無効
認証スキーム	HTTP 認証方法	None，Digest，Negotiate，NTLM，IntegratedWindowsAuthentication，Basic，Anonymous

オプション	意味	指定可能な値
領域	認証スキームの範囲	任意の URL
クライアント証明書を要求する	SSL トランスポートの場合は，証明書が必要になる	有効 / 無効

プロキシ情報

Web サービスのトランスポートでプロキシ・サーバを使用する場合は，［**セキュリティ**］タブで詳細を指定できます。次の表で，プロキシ・オプションについて説明します。

オプション	意味	指定可能な値
規定の Web プロキシを使う	マシンの標準プロキシ設定を使用する	有効 / 無効
ローカルではプロキシは使わない	サービスがローカル・マシンにある場合にプロキシを無視する	有効 / 無効
プロキシ アドレス	プロキシ・サーバ	任意の URL
プロキシ認証スキーム	プロキシの HTTP 認証方式	None，Digest，Negotiate，NTLM，IntegratedWindowsAuthentication，Basic，Anonymous

セキュリティ・モデルのカスタマイズ

組み込みシナリオとシナリオ・エディタによって，高度な標準を使用するほとんどの Web サービスをテストできます。

ただし，設定ファイルを手作業で変更する必要があります。

設定ファイルを変更するには，次の手順で行います。

1. 169 ページ「サービスのセキュリティ・シナリオ設定」で説明しているように，シナリオを選択するかセットアップします。
2. スクリプトのルート・ディレクトリを開きます。スクリプト・ビューで，スクリプト内をクリックし，右クリック・メニューから［**スクリプト ディレクトリを開く**］を選択します。
3. 内部フォルダ，**% スクリプト・ルート %/WSDL/@config** に移動します。このフォルダには，1 つ以上の .stss ファイルがあります。
4. 関連する .stss ファイルを開きます。ディレクトリに .stss ファイルが 1 つしかない場合は，テキストエディタで開きます。複数のファイルがある場合は，**configurationIndex.xml** を開き，現在設定しているサービスに一致する .stss ファイルを確定します。
5. 以下の項で説明するように，設定ファイルを変更します。ファイルを変更した後で，［サービスの管理］の［**プロトコルとセキュリティ**］タブから設定を再び更新しないでください。そうすると，変更が無効になります。

設定ファイルを使って実行できる機能は次のとおりです。

応答バッファ容量の増加

MTOM を使用する場合によくあるように，大きな応答が予想される場合は，再生によって次のエラーが発生する可能性があります。

エラー：着信メッセージの最大メッセージ容量割り当て（65536）を超えました。割り当てを増加するには，該当するバインド要素で MaxReceivedMessageSize プロパティを使用します。

これを解決するには，**httpTransport** 要素に **maxReceivedMessageSize** 属性を追加し，大容量を使用できるように設定します。たとえば，次のように設定します。

```
<protocols scenario="customBinding xmlns="http://hp/ServiceTest/config">"
    <customization>
        <mtomMessageEncoding />
            <httpTransport maxReceivedMessageSize="6000000" />
    </customization>
</protocols>
```

Windows 資格情報のカスタマイズ

セキュリティ・シナリオの一部では，Windows 資格情報を使用します。これは WCF WsHttpBinding SPNEGO および Kerberos を利用するサービスでは一般的です。標準設定では，Service Test は現在ログインしているユーザをクライアント識別属性として使用します。別のユーザの識別属性を使用するよう Service Test に指示するには，該当する設定ファイルを変更します。いずれかの要素が設定ファイルにある場合は，新しい情報で更新します。要素を複製しないでください。

```
<protocols ...>
    <identities>
        <client>
            <windowsCredentials>
                <username>myUser</username>
                <password>myPassword</password>
                <domain>myDomain</domain>
            </windowsCredentials>
        </client>
    </identities>
    ...
</protocols>
```

論理アドレス（clientVia Behavior）の使用

この項では，物理アドレスではなく，**clientVia** 動作をエミュレートする論理アドレスを指定する方法について説明します。この場合は，メッセージを中間サービスに送信し，そのサービスによって実際のサーバに送信されます。メッセージをデバッグ・プロキシに送信するときも，これを行うことができます。

このような場合は，メッセージを実際に送信する物理アドレスと，メッセージが対象とする論理アドレスを区別するのが便利です。論理アドレスは，最終サーバの物理アドレスでも任意の名前でもかまいません。論理アドレスは SOAP メッセージに次のように表示されます。

<wsa:Action>http://myLogicalAddress<wsa:Action>

論理アドレスの設定

論理アドレスは，Service Test ユーザインタフェースで決定します。標準設定では，WSDL で指定したアドレスです。このアドレスは，［サービスの管理］ダイアログ・ボックスからオーバーライドできます。

物理アドレスの設定

物理アドレスを設定するには，設定ファイルを変更する必要があります。**protocols** 要素の下で，**behaviors** 要素を変更します。必ず論理アドレスを正しいアドレスに変更してください。

```
<protocols scenario="customBinding" xmlns="http://hp/ServiceTest/config">
...
<behaviors>
    <endpointBehaviors>
        <behavior>
            <clientVia viaUri="http://MyLogicalAddress" />
        </behavior>
    </endpointBehaviors>
</behaviors>
</protocols>
```

注：前述の動作が（標準動作の場合と同様に）設定ファイルにないと，論理アドレスが物理アドレスにもなり，要求が物理アドレスに送信されます。

セキュリティ要素のパラメータ化

新しいセキュリティ・モデルを利用するスクリプトは，通常の方法でパラメータ化できます。セキュリティ要素も独立してパラメータ化できます。たとえば，ユーザ名ベースのセキュリティ・シナリオでは，それぞれの Vuser または反復ごとに異なるユーザ名を使用させることができます。

各セキュリティ要素ごとに個別にパラメータを作成するには，次の手順を実行します。

1 シナリオ・エディタを開きます。［**SOA ツール**］＞［**セキュリティ シナリオ エディタ**］を選択します。

2 各 Vuser ごとのシナリオを設定して保存します。**user1**，**user2** などの名前を使用し，新しいフォルダ，**% スクリプト・ルート %/WSDL/referencedConfig** に保存することをお勧めします。

3 ［パラメータ リスト］ウィンドウを開きます。［**仮想ユーザ**］＞［**パラメータ リスト**］を選択します。

4 新しいパラメータ，＜**サービス名**＞ **_shared_config** を作成します。＜サービス名＞をテストするサービスの名前（大文字 / 小文字を区別）に置き換えます。サービスの正確な名前を決めるには，［**サービスの管理**］をクリックして，サービスのリストを確認します。

5 パラメータ・ダイアログ・ボックスで，.stss 拡張子の付いたセキュリティ・シナリオのファイル名をパラメータ値として追加します。スクリプト・ルート・フォルダに対する相対パスを使用できます。［**行の追加**］をクリックして，複数の値を追加します。

6 ［**サービスの管理**］をクリックし，［**プロトコルとセキュリティ**］タブを選択します。［**データの編集**］をクリックします。

7 ［**共有シナリオ**］を選択します。［参照］ボタンをクリックして，テスト・ボックスにパラメータ名，＜**サービス名**＞ **_shared_config** を入力します。

反復によるユーザのシミュレート

多くのセキュリティ・シナリオでは，サーバを使用してセッションを構築しています。たとえば，**WS-SecureConversation** を使用するすべてのシナリオでは，サーバ・セッションを構築します。このセッションは，最初の操作を実行するときに構築され，スクリプトが終了すると終わります。標準設定では，Service Test は各反復の終了後にセッションを閉じ，次の反復が始まると再開します。つまり，すべての反復では新しいセッションと Vuser をシミュレートします。

多数の反復を使用するときは，これが望ましい効果にならないこともあります。元のセッションをアクティブにしたまま，各反復で新しいセッションを開かないようにしてください。LoadRunner Controller による負荷テストのときや，実行環境設定で多数の反復を設定するときも同様です。

この動作を無効にできますが，そうすると最初の反復だけで新しいセッションが構築され，その後の反復では開いているセッションが使用されます。これで，同じセッションを使用してアクションを繰り返し実行するユーザがシミュレートされます。

利用するシミュレーション・モードを決めるには，シミュレートするものに最適なモードを選択します。たとえば，ほとんどのアクションが単一セッションで同じユーザによって繰り返し実行される負荷テストをシミュレートする場合は，前述の設定を使用します。よくわからない場合は，標準設定のままにします。

すべての反復で同じセッションを使用する環境を設定するには，次の手順を実行します。

1 スクリプトのルート・ディレクトリを開きます。スクリプト・ビューで，スクリプト内をクリックし，右クリック・メニューから［**スクリプト ディレクトリを開く**］を選択します。

2 テキスト・エディタで **default.cfg** ファイルを開きます。

3 ［**WebServices**］セクションで，toolkit の下に 1 行を追加します。

```
[WebServices]
Toolkit=.Net
SimulateNewUserInNewIteration=0
```

Axis ツールキットを使用する場合，またはほかの設定を行う場合は，ファイル内容が異なることがあります。

4 ファイルを保存して閉じます。

ヒントとガイドライン

この項では，Service Test for WCF，一般的なセキュリティ，および高度な標準テストの使用について要約します。

WCF

WCF サービスをテストする方法

［**サービスの管理**］をクリックし，［**プロトコルとセキュリティ**］タブを選択します。［**データの編集**］をクリックします。

WCF ノードを展開し，バインドに従って適切なシナリオを選択します。適切なバインドが見つからない場合は，任意のバインドをテストできるので，**customBinding** シナリオを選択します。

WSHttpBinding を使用する WCF サービスをテストする方法

WSHttpBinding は WCF で最も人気のあるバインドの 1 つです。このバインドを使用するには，［**サービスの管理**］をクリックして，［**プロトコルとセキュリティ**］タブを選択します。［**データの編集**］をクリックします。

［**WCF**］＞［**クライアント認証のタイプ別**］ノードを展開して，バインドで使用するクライアント資格情報を選択します。この値は，WCF の WSHttpBinding の **MessageClientCredentialType** プロパティに対応しています。

新しい WCF サービスでは，**Windows** 認証が標準設定値です。サービスに WCF の標準設定を使用する場合は，このオプションを使用します。その他のオプションは ユーザ名，証明書，またはなしです。ユーザ名はメッセージ・レベルでもトランスポート・レベルでもかまいません（WCF の **TransportWithMessageCredential** に相当）。

一部のシナリオでは，WCF 独自のネゴシエーション・メカニズムを利用して，サービス証明書を取得するかどうかを指定してください。

詳細なシナリオ・プロパティを使用して，セキュア・セッションの使用を制御します。

CustomBinding を使用する WCF サービスをテストする方法

173 ページ「シナリオ・タイプ」で説明しているように，**WCF** ＞ **カスタム・バインディング**のシナリオ・タイプを選択します。そのときに，トランスポート・メソッド，エンコーディング，セキュリティ，および信頼できるメッセージングなど，多くのバインド要素をカスタマイズできます。

netTcp または namedPipe トランスポートを使用する WCF サービスをテストする方法

173 ページ「シナリオ・タイプ」で説明しているように，**WCF** ＞ **カスタム・バインディング**のシナリオ・タイプを選択します。トランスポートを **TCP** または **NamedPipe** に設定します。

STS（セキュリティ・トークン・サービス）を使用する フェデレーション・シナリオをテストする方法

このシナリオでは，STS およびサービスの通信プロパティを定義する必要があります。さらに，Microsoft の WSE3 と互換性のあるフェデレーション・シナリオは，**web_service_set_security_saml** 関数を使ってテストできます。詳細については，『**Online Function Reference**』（英語版）（［**ヘルプ**］＞［**関数リファレンス**］）または第 9 章「Web サービス－セキュリティ」を参照してください。

WCF ＞ **WSFederationHttpBinding** シナリオを選択します。このシナリオでは，STS およびアプリケーション・サーバの通信プロパティを定義する必要があります。

アプリケーション・サーバの通信プロパティを定義するには，［サービスの管理］ダイアログ・ボックスの［**プロトコルとセキュリティ**］タブを使用します。

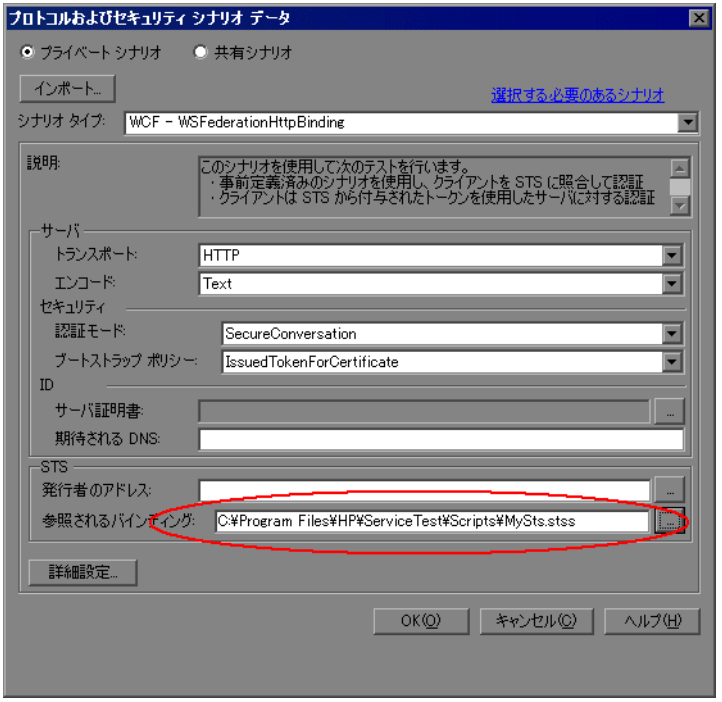
STS との通信を設定するには，次の手順を実行します。

1 スタンドアロンのセキュリティ・シナリオ・エディタを開きます。［**SOA ツール**］＞［**セキュリティ シナリオ エディタ**］を選択します。

2 ［**新規作成**］ボタンをクリックします。STS との通信を設定します。

3 ［**名前を付けて保存**］をクリックし，ファイル名を指定します。

4［サービスの管理］ダイアログ・ボックスを開き，［**プロトコルとセキュリティ**］タブを選択します。［**データの編集**］をクリックします。**STS** セクションで，前のステップで作成したシナリオを参照します。



一般セキュリティ

SSL を使用する Web サービスをテストする方法

セキュアなサイトをテストするのに，特別な設定は必要ありません。サービスの URL が **https** から始まる場合は，SSL が自動的に使用されます。SSL に加えて，メッセージ・レベルのセキュリティ（ユーザ名など）を使用する場合は，レガシまたはシナリオ・ベースのモデルを利用して，メッセージごとにセキュリティを設定する必要があります。シナリオ・ベースのモデルを利用する場合は，WSHttpBinding シナリオで HTTPS トランスポートまたはトランスポート証明書モードを選択して，SSL を使用するように設定する必要があります。

HTTP レベルで Windows 認証を必要とする Web サービスをテストする方法

web_set_user 関数を使用します。追加標準が必要な場合は，シナリオ・ベースのモデルと一緒に，またはその代わりに，レガシ・セキュリティ・ベースのモデルを利用します。

WS-Security を使用する Web サービスをテストする方法

この項で説明しているシナリオ・ベースのセキュリティ，または **web_service_set_security** を使用するレガシ・セキュリティを使用します。

WS-Security トークンの低レベル詳細を設定する方法

ほとんどの場合，182 ページ「詳細設定」で説明しているように，低レベル詳細を設定できます。WS-Security トークンに対するごく低レベルの制御が必要な場合は，レガシ・セキュリティ・モデルを使用します。詳細については，第 9 章「Web サービス－セキュリティ」を参照してください。

STS（セキュリティ・トークン・サービス）を使用する Federation シナリオをテストする方法

このシナリオでは，STS およびサービスの通信プロパティを定義する必要があります。さらに，Microsoft の WSE3 と互換性のある Federation シナリオは，**web_service_set_security_saml** 関数を使ってテストできます。詳細については，『**Online Function Reference**』（英語版）（［**ヘルプ**］＞［**関数リファレンス**］）または第 9 章「Web サービス－セキュリティ」を参照してください。

WCF ＞ **WSFederationHttpBinding** シナリオを選択します。このシナリオでは，STS およびアプリケーション・サーバの通信プロパティを定義する必要があります。

アプリケーション・サーバの通信プロパティを定義するには，［サービスの管理］ダイアログ・ボックスの［**プロトコルとセキュリティ**］タブを使用します。

STS との通信を設定するには，次の手順を実行します。

1 スタンドアロンのセキュリティ・シナリオ・エディタを開きます。［**SOA ツール**］＞［**セキュリティ シナリオ エディタ**］を選択します。

2［**新規作成**］ボタンをクリックします。STS との通信を設定します。

3［**名前を付けて保存**］をクリックし，ファイル名を指定します。

4［サービスの管理］ダイアログ・ボックスを開き，［**プロトコルとセキュリティ**］タブを選択します。［**データの編集**］をクリックします。**STS** セクションで，前のステップで作成したシナリオを参照します。

一般的なプロトコル

詳細な設定を使用しないことを指示する方法

シナリオ・タイプとして，＜**シナリオなし**＞を選択します。

シナリオを選択し，再生中にエラーを受信した場合は，詳細なシナリオを必要としない可能性があります。＜**シナリオなし**＞を選択し，既存の選択を中止して，スクリプトを再実行します。

MTOM を使用する Web サービスをテストする方法

［**MTOM**］シナリオを選択します。追加セキュリティが必要な場合は，ほかのシナリオのいずれかを選択します。［詳細設定］ダイアログ・ボックスで，エンコーディングを MTOM に設定します。詳細については，182 ページ「詳細設定」を参照してください。

サービスの WS-Addressing バージョンを変更する方法

標準設定では，.NET ツールキットは WS-Addressing 2004/03 を使用しますが，Axis ツールキットはどんなアドレッシングも使用しません。この動作をオーバーライドするには，［**プレーンな SOAP**］シナリオを選択し，WS-Addressing バージョンを選択します。サポートされているほかのバージョンは 2004/08，1.0，および None です。サービスでセキュリティなどの追加標準が必要な場合は，適切なシナリオを使用し，［詳細設定］ウィンドウの［**エンコード**］タブからアドレッシング・バージョンを設定します。詳細については，182 ページ「詳細設定」を参照してください。

第 11 章

Web サービス－トランスポート層とカスタマイズ

トランスポート層のプロパティを設定し，Web サービス呼び出しの動作を定義するユーザ・ハンドラを作成して，Web サービス呼び出しをカスタマイズできます。

本章の内容

以下の情報は，Web サービスと SOA Vuser スクリプトのみを対象とします。

Web サービス・トランスポート層のテストについて

Web サービスは，さまざまなトランスポート層で送信できます。トランスポート層は，サーバとメッセージをやり取りするのに使用するプロトコルです。

VuGen では，サービスのトランスポート層を設定できます。HTTP/HTTPS および JMS（Java Message Service）トランスポート層を完全サポートしています。

Service Test ソリューションによって，同期および非同期メッセージングをエミュレートできます。HTTP/HTTPS トランスポートを使用する場合は，WS-Addressing も使用できます。

ユーザ・ハンドラを使用すると，SOAP 要求および応答を処理して，カスタム動作に割り当てることができます。詳細については，第 12 章「Web サービス－ユーザ・ハンドラとカスタマイズ」を参照してください。

トランスポート層の設定

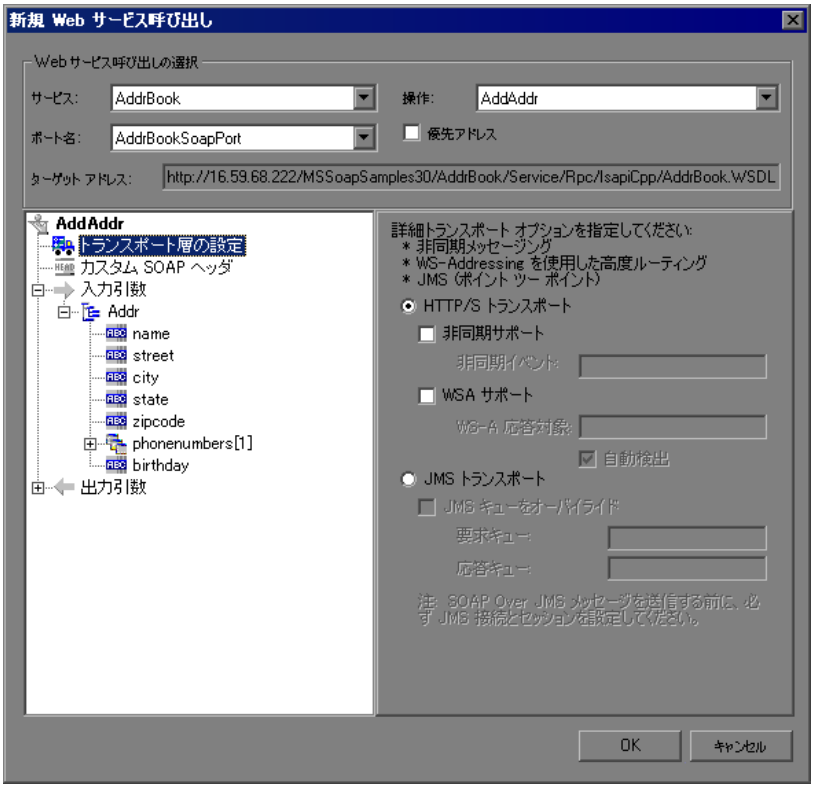
VuGen では，サービスのトランスポート層を設定できます。トランスポート層では，サーバとメッセージをやり取りする方法，つまり HTTP/HTTPS または JMS（Java Message Service）を指定できます。

HTTP/HTTPS については，200 ページ「HTTP/HTTPS でのメッセージ送信」を参照してください。JMS トランスポートの使用法の詳細については，201 ページ「JMS について」を参照してください。

トランスポート層を選択するには，次の手順を実行します。

1 新規の Web サービス呼び出しを作成します。既存の呼び出しの場合は，ツリー・ビューを開いて，ステップを選択し，［**ステップのプロパティ**］タブを選択します。

2［**トランスポート層の設定**］ノードを選択します。



3 目的のトランスポート・モード，すなわち［**HTTP/S トランスポート**］または［**JMS トランスポート**］を選択します。

詳細なオプションについては，「HTTP/HTTPS でのメッセージ送信」または 202 ページ「トランスポート層としての JMS の使用」を参照してください。

HTTP/HTTPS でのメッセージ送信

HTTP は Web クライアント（通常はブラウザ）から Web サーバに要求を送信するのに使用します。また，サーバからクライアントに Web コンテンツを返すのにも使用します。

HTTPS は，クライアントとサーバ間のセキュアな通信を処理します。通常は，クレジット・カードのトランザクションなど，極秘データを処理します。

一般的な要求と応答のメカニズムは同期です。同期メッセージングでは，再生エンジンはサーバが応答を送信するまでスクリプト実行をブロックします。非同期モードでは，再生エンジンは前のメッセージに対するサーバの応答を待機せずにスクリプトを実行します。

HTTP/HTTPS トランスポートを使用する場合，非同期呼び出しとともに WS-Addressing を使用できます。詳細については，205 ページ「HTTP/HTTPS を使用した非同期呼び出しの送信」を参照してください。

また，Service Test では，Web サービス呼び出しの非同期メッセージングをエミュレートできます。詳細については，213 ページ「JMS を使用した非同期呼び出しの送信」を参照してください。

JMS について

JMS は Java クライアント間でメッセージ（テキストまたは Java オブジェクト）を送信するための J2EE 標準です。

通信には次の 2 つのシナリオがあります。

ピアツーピア：**ポイントツーポイント**とも呼ばれています。JMS は，メッセージ・キューをメッセージのターゲットとして定義することで，ポイントツーポイントのメッセージングを実装します。複数の送信者が 1 つのメッセージ・キューにメッセージを送信し，受信者はそのキューからメッセージを取得します。

パブリッシュ－サブスクライブ：各メッセージが指定されたトピックを通じて，1 人の発行者から多数の予約購読者に送信されます。予約購読者は，予約購読後に送信されるメッセージを受信するだけです。

VuGen では，JMS メッセージをキューと送受信できるようにして，ポイントツーポイント通信をサポートしています。

JMS トランスポートを使用してメッセージを送信するには，トランスポートを記述する次の項目を設定する必要があります。

- ［**JNDI 初期コンテキスト ファクトリ**］：JMS 接続ファクトリや JMS キューなどの JMS リソースの検索に使用する初期コンテキストを作成するファクトリ・クラスのクラス名。
- ［**JNDI プロバイダ URL**］：JMS 接続ファクトリや JMS キューなどの JMS リソースの検索に使用するサービス・プロバイダの URL。
- ［**JMS 接続ファクトリ**］：JMS 接続ファクトリの JNDI 名です。

  また，受信メッセージのタイムアウトおよびプロセスごとの JMS 接続数を設定できます。

  これらの設定はすべて JMS 実行環境の設定を使用して設定できます。詳細については，124 ページ「Web サービス JMS の実行環境の設定」を参照してください。

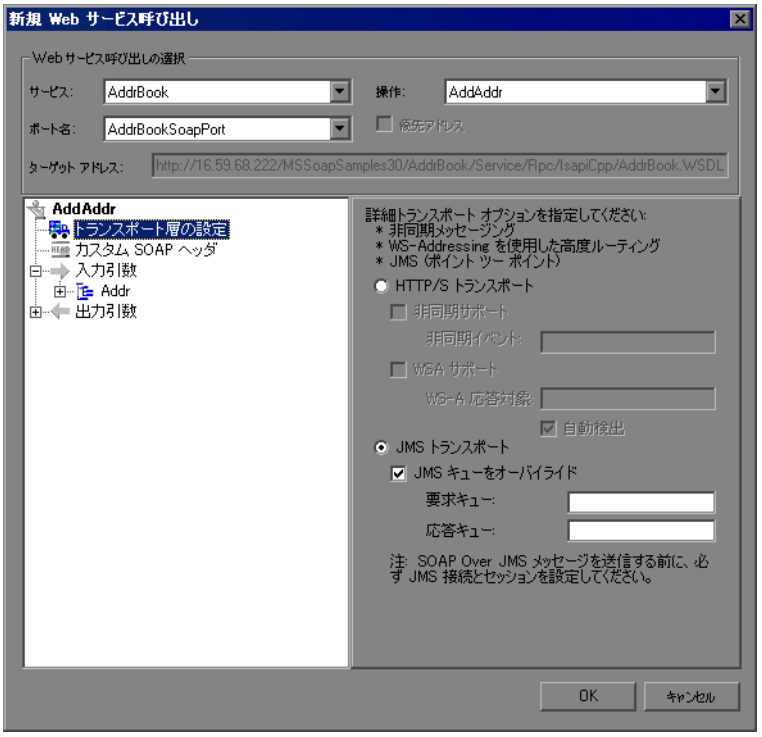
トランスポート層としての JMS の使用

Web サービスに JMS トランスポートを使用すると，一般の HTTP トランスポートではなく JMS トランスポートを使用して SOAP メッセージを送信できます。

JMS トランスポート層でメッセージを送信するには，次の手順を実行します。

1 新規の Web サービス呼び出しを作成します。既存の呼び出しの場合は，ツリー・ビューを開いて，ステップを選択し，［**ステップのプロパティ**］タブを選択します。

2 ［**トランスポート層の設定**］ノードを選択し，［**JMS トランスポート**］を選択します。



3 124 ページ「Web サービス JMS の実行環境の設定」で説明しているように，スクリプトを実行する前に JMS 実行環境の設定を行います。

同期 JMS メッセージを送信する方法については，前述の手順どおりです。JMS での非同期メッセージのエミュレートについては，213 ページ「JMS を使用した非同期呼び出しの送信」を参照してください。

また，Web サービス呼び出しを作成しなくても，JMS でメッセージを送信できます。たとえば，VuGen を使用すると，次のことができます。

- 標準の Web プロトコルを使用して SOAP メッセージを記録し，JMS トランスポート関数によってキューに送信します。
- 指定のキューから一般的な JMS メッセージを手作業で送受信します。

詳細については，「JMS スクリプト関数」を参照してください。

JMS スクリプト関数

VuGen では，API 関数を使用して JMS トランスポートを実装します。各関数の先頭には，**jms** というプレフィックスが付きます。

関数名	説明
jms_receive_message_queue	キューからメッセージを受信します。
jms_send_message_queue	メッセージをキューに送信します。
jms_send_receive_message_queue	指定されたキューにメッセージを送信し，指定されたキューからメッセージを受信します。
jms_set_general_property	ユーザ・コンテキストで一般的なプロパティを設定します。
jms_set_message_property	次に送信されるメッセージの JMS ヘッダまたはプロパティを設定します。

JMS ステップ / 関数は，手作業でスクリプトを作成するときにのみ使用できます。クライアントとサーバ間で送信される JMS メッセージは記録できません。

JMS 関数の詳細については，『**Online Function Reference**』（英語版）（［**ヘルプ**］＞［**関数リファレンス**］を選択するか，関数の上で **F1** キーを押します）を参照してください。

JMS メッセージ・タイプ

それぞれの JMS メッセージは次の要素で構成されています。

- **ヘッダ**：標準属性（相関 ID，優先度，有効期限）が含まれています。
- **プロパティ**：カスタム属性。
- **本文**：テキストまたはバイナリ情報。

JMS は複数のメッセージ本文形式を使用して送信できます。一般的な形式は **TextMessage** と **BytesMessage** の 2 つです。

Service Test はメッセージのコンテント・タイプに基づいて，目的の形式を解決しようとします。コンテント・タイプが **text/*** の場合は，TextMessage 形式でメッセージが送信されます。それ以外の場合は，**BytesMessage** 形式で送信されます。

標準動作をオーバーライドするには，メッセージを送信する前に，**jms_set_general_property** 関数を使用します。JMS_MESSAGE_TYPE プロパティは，TextMessage，BytesMessage，または Default に設定します。たとえば，次のように設定します。

```
jms_set_general_property("step1","JMS_MESSAGE_TYPE","BytesMessage");
```

詳細については，『**Online Function Reference**』（英語版）を参照してください。

非同期メッセージの送信

VuGen を使用すると，同期および非同期メッセージングをエミュレートできます。

同期メッセージングでは，再生エンジンはサーバが応答を送信するまでスクリプト実行をブロックします。非同期モードでは，再生エンジンは前のメッセージに対するサーバの応答を待機せずにスクリプトを実行します。

HTTP/HTTPS を使用した非同期呼び出しの送信

次の項では，HTTP/HTTPS で非同期呼び出しを使用する方法について説明します。**イベント待機**ステップを使用して，以前の非同期要求の応答を待ってから続行するよう Vuser に指示します。リスナは，サーバが応答するまでサービスの実行をブロックします。

Web サービス イベント待機ステップを追加する場合には，次を実行します。

- **Quantifier**：Quantifier は，**ALL** イベントが応答を受信するのを待つか，**ANY**（それらの 1 つだけ）にするかを Vuser に指示します。**ANY** は応答を受信する最初のイベント名を返します。**ALL** はイベント名を 1 つだけ返します。
- **タイムアウト**：タイムアウト時間。単位はミリ秒。指定したタイムアウト時間内にイベントが応答を受信しない場合は，**web_service_wait_for_event** が NULL を返します。
- **Events**：待機させる非同期イベントをすべてリストアップします。

非同期メッセージングを指定してスクリプトを実行すると，再生ログにはイベントおよび入力引数 / 出力引数に関する情報が記録されます。

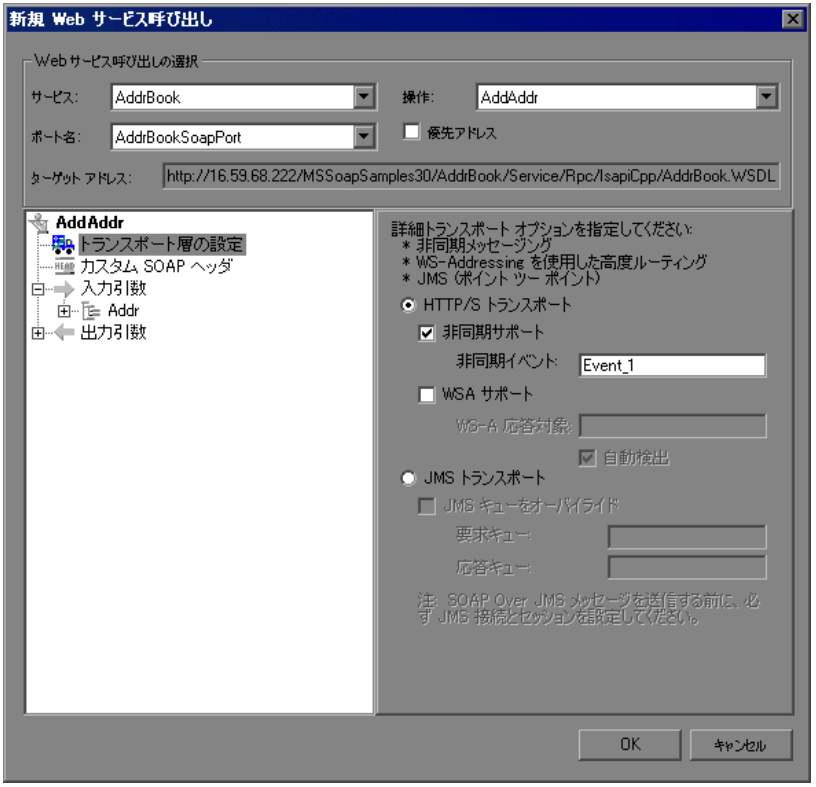
Web サービス イベント待機ステップまたは **web_service_wait_for_event** 関数の詳細については，『**Online Function Reference**』（英語版）（［**ヘルプ**］＞［**関数リファレンス**］を選択するか，関数上で **F1** キーを押します）を参照してください。

非同期メッセージを設定するときに，サービスが WS-Addressing を使用してイベントを検出すると応答する場所を設定できます。詳細については，209 ページ「WS-Addressing」を参照してください。

トランスポート・プロトコルとして HTTP/S を使って非同期メッセージを送信するには，次の手順を実行します。

1 ツリー・ビューで，［**ステップのプロパティ**］タブを選択し，［**トランスポート層の設定**］ノードを選択します。

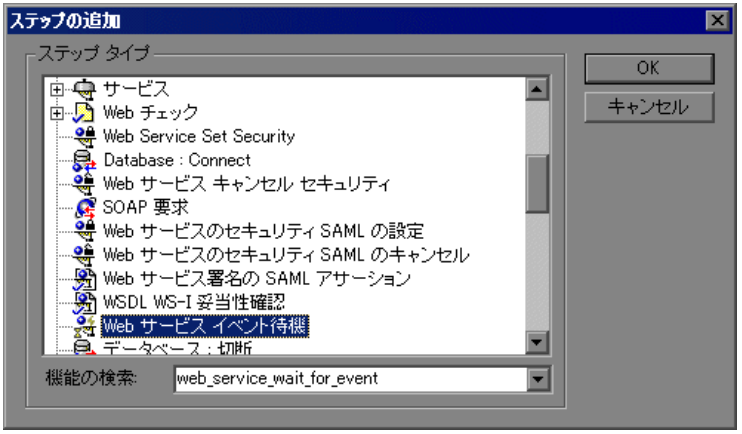
2［**HTTP/S トランスポート**］を選択し，［**非同期サポート**］オプションを選択します。



3［**非同期イベント**］ボックスにイベント名を入力します。

4［**OK**］クリックすると，Web サービス呼び出しが生成されます。

5 **イベント待機**ステップを追加します。［**挿入**］＞［**新規ステップ**］を選択し，［**Web サービス イベント待機**］を選択します。



6 ステップ名，数量子，およびタイムアウトを指定します。［**追加**］をクリックして，前のステップで定義したイベントの名前を挿入します。

スクリプト・ビューでは，VuGen から，追加パラメータとして**非同期サポート**を使用した非同期メッセージングを指示されます。

```
web_service_call( "StepName=EchoString_101",
        "SOAPMethod=EchoRpcEncoded.EchoSoap.EchoString",
        "ResponseParam=response1",
        "Service=ExtendedECHO_rpc_encoded",
        "AsyncEvent=Event_1",
        "Snapshot=t1157371707.inf",
        BEGIN_ARGUMENTS,
        "sec=7",
        "strString=mytext",
        END_ARGUMENTS,
        BEGIN_RESULT,
        "EchoStringResult=first_call",
        END_RESULT,
        LAST);
```

AsyncEvent フラグは，前の非同期サービス要求の応答を待つよう Vuser に指示します。

同期メッセージの場合は，標準の Web サービス呼び出しを作成し，［**Async Support**］オプションを有効にしないでください。

WS-Addressing

WS-Addressing は，Web サービスがアドレス指定情報を伝えられるようにする仕様です。この仕様では，メッセージにおけるエンドツーエンドのエンドポイントを識別できるように，Web サービス・エンドポイントを識別します。これにより，エンドポイント・マネージャ，ファイアウォール，およびゲートウェイなど追加の処理ノードを持つネットワークを経由してメッセージを送信できます。WS-Addressing は，同期トランスポートおよび非同期トランスポートの両方を伝送される Web サービス・メッセージをサポートしています。

WS-Addressing 仕様では，サービスの応答先となる **WSAReplyTo** アドレスが要求されます。

オプションの **WSAAction** 引数によって，トランスポート層でメッセージを送信できないインスタンスの SOAP アクションを定義できます。

次の例に，VuGen によってバックグラウンドに実装された，WS-Addressing を使用する一般的な SOAP メッセージを示します。

```
<S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
            xmlns:wsa="http://www.w3.org/2004/12/addressing">
    <S:Header>
        <wsa:MessageID>
            http://example.com/SomeUniqueMessageIdString
        </wsa:MessageID>
        <wsa:ReplyTo>
        <wsa:Address>http://myClient.example/someClientUser</wsa:Address>
        </wsa:ReplyTo>
<wsa:Address>http://myserver.example/DemoErrorHandler</wsa:Address>
        </wsa:FaultTo>
        <wsa:To>http://myserver.example/DemoServiceURI</wsa:To>
        <wsa:Action>http://myserver.example/DoAction</wsa:Action>
    </S:Header>
    <S:Body>
        <!-- Body of SOAP request message -->
    </S:Body>
</S:Envelope>
```

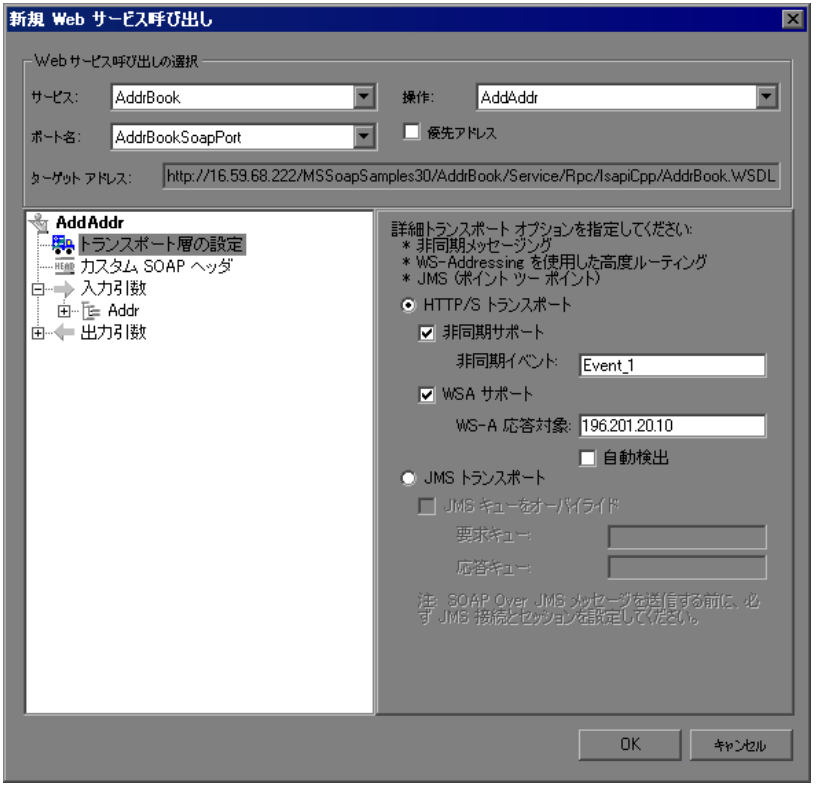
次の例では，サーバが Event_1 を検出すると，インタフェース 212.199.95.138 に応答します。

```
web_service_call( "StepName=Add_101",
        "SOAPMethod=Calc.CalcSoap.Add",
        "ResponseParam=response",
        "AsyncEvent=Event_1",
            "WSAReplyTo=212.199.95.138",
        "WSDL=http://lab1/WebServices/CalcWS/Calc.asmx?wsdl",
        "UseWSDLCopy=1",
        "Snapshot=t1153825715.inf",
        BEGIN_ARGUMENTS,
            "first=1",
            "second=2",
        END_ARGUMENTS,
        BEGIN_RESULT,
            "AddResult=Param_AddResult1",
        END_RESULT,
        LAST);
```

WS-Addressing 呼び出しは，非同期および同期モードで発行できます。同期モードで WS-Addressing を使用するには，［トランスポート層］オプションの［**非同期イベント**］ボックスを空にしておきます。スクリプト・ビューで，**AsyncEvent** 引数を削除します。こうすることで，再生エンジンは完全な応答がサーバから受信されるまでスクリプトの実行をブロックします。

非同期メッセージおよび WS-Addressing のサポートを追加するには，次の手順を実行します。

1 新規の Web サービス呼び出しの場合は，［**トランスポート層の設定**］ノードを選択します。既存の Web サービス呼び出しの場合，ツリー・ビューでステップを選択し，［**ステップのプロパティ**］タブを選択します。［**トランスポート層の設定**］ノードを選択します。



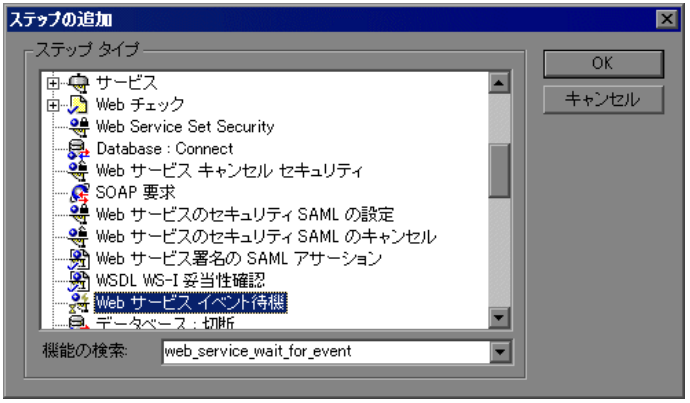
2 Web サービス呼び出しを非同期メッセージとしてマークするには，次の手順を実行します。

- ➤［**非同期サポート**］オプションを選択します。これは，HTTP/S タイプのトランスポートの場合にのみ有効になります。
- ➤［**非同期イベント**］ボックスにイベント名を指定します。任意の名前を指定できます。

3［**WSA サポート**］を選択します。［**WS-A 応答対象**］ボックスに，IP アドレスまたは現在のホストを使用する **autodetect** を入力します。autodetect は，複数の異なるマシンで同じスクリプトを実行する場合に便利です。これにより，イベントの発生時にサーバは指定された場所に応答します。

4［**OK**］をクリックして設定を保存します。

5 Vuser にイベントを待つよう指示します。［**挿入**］＞［**新規ステップ**］を選択し，Web サービス呼び出しステップの後に **Web サービス イベント待機**ステップを追加します。



6 ステップ名，数量子，およびタイムアウトを指定します。イベント名を追加するには，［**追加**］をクリックします。これにより，Web サービスは，指定されたイベントを待ってから応答します。

7 ［**編集**］，［**上へ移動**］，および［**下へ移動**］ボタンを使用してイベントを操作します。

8 ［**OK**］をクリックします。

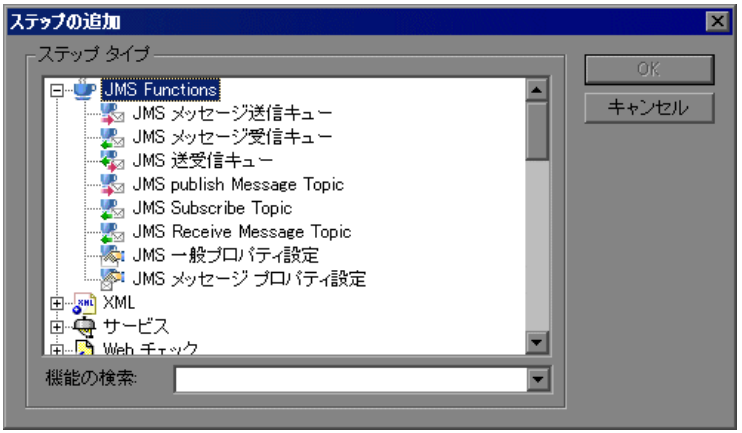
JMS を使用した非同期呼び出しの送信

この項では，JMS を使用して非同期メッセージを送信する方法について説明します。

これを実行するには，Web サービス呼び出しではなく，JMS を使用して要求を送信するか，応答を受信します。**JMS メッセージ送信キュー**でメッセージをキューに送信します。**JMS メッセージ受信キュー**でキューからメッセージを受信します。

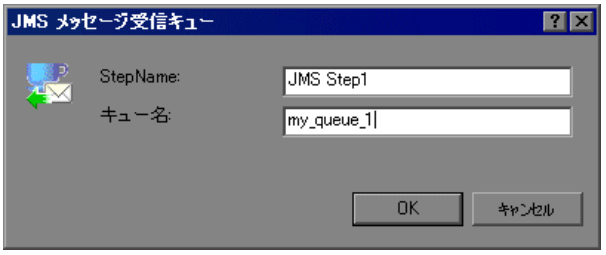
JMS を使用して非同期メッセージを送受信するには，次の手順を実行します。

1 スクリプト内で目的の位置をクリックします。［**挿入**］＞［**新規ステップ**］を選択して，［**JMS 関数**］ノードを展開します。



2 JMS 関数を選択して，メッセージ・キュー情報，すなわち **JMS メッセージ送信キュー**または **JMS メッセージ受信キュー**を設定します。

3 ［**OK**］をクリックして，JMS 関数のプロパティを開きます。



4 キュー名を指定して，［**OK**］をクリックすると，JMS 関数が生成されます。

JMS 同期メッセージの場合は，標準の Web サービス呼び出しを作成し，JMS トランスポートを選択して，必要ならばキュー情報を指定します。

これらの関数の詳細については，『**Online Function Reference**』（英語版）（［**ヘルプ**］＞［**関数リファレンス**］を選択するか，関数の上で **F1** キーを押します）を参照してください。

第 12 章

Web サービス－ユーザ・ハンドラとカスタマイズ

Service Test では，ユーザ・ハンドラと設定ファイルを使用してスクリプトをカスタマイズできます。

本章の内容

以下の情報は，Web サービスと SOA Vuser スクリプトのみを対象とします。

Web サービス・スクリプト動作のカスタマイズについて

VuGen には，スクリプトの動作方法をカスタマイズできる高度な機能がいくつか用意されています。これらの機能がユーザ・ハンドラと設定ファイルです。

ユーザ・ハンドラを使用すると，SOAP 要求および応答を処理して，カスタム動作に割り当てることができます。詳細については，次を参照してください。

設定ファイルでは，セキュリティ情報や WSE 設定などの詳細設定をカスタマイズできます。詳細については，226 ページ「カスタム設定ファイルの使い方」を参照してください。

ユーザ・ハンドラの設定

ユーザ・ハンドラは，次の操作を実行できるオープン API です。

- ➤ 要求 / 応答 SOAP エンベロープを取得，設定する
- ➤ トランスポート層をオーバーライドする
- ➤ 要求 / 応答コンテント・タイプを取得，設定する
- ➤ LoadRunner パラメータの値を取得，設定する
- ➤ スクリプトから設定引数を取得する
- ➤ 実行ログにメッセージを発行する
- ➤ 実行を失敗する

ユーザ・ハンドラはスクリプトで直接設定するか，DLL を通じて実装できます。ハンドラはローカルまたはグローバルに適用できます。詳細については，次の項を参照してください。

- ❑ スクリプトでハンドラ関数を定義
- ❑ DLL としてカスタムのユーザ・ハンドラを作成
- ❑ ユーザ・ハンドラの設定
- ❑ ユーザ・ハンドラの実装

サンプルのユーザ・ハンドラについては，222 ページ「ユーザ・ハンドラの例」を参照してください。

スクリプトでハンドラ関数を定義

ユーザ・ハンドラの基本実装については，Vuser スクリプト内でユーザ・ハンドラ関数を定義します。

```
int MyScriptFunction(const char* pArgs, int isRequest)
{
...
}
```

pArgs 引数には，**web_service_call** 関数の **UserHandlerArgs** 引数で指定した文字列が含まれています。詳細については，『**Online Function Reference**』（英語版）（［**ヘルプ**］＞［**関数リファレンス**］）を参照してください。

isRequest 引数は，Request (1) または Response (0) SOAP エンベロープの処理中にこの関数を呼び出すかどうかを指定します。

SOAP エンベロープの内容は，要求および応答のどちらでも，**SoapEnvelopeParam** というパラメータに渡されます。この関数で SOAP エンベロープを処理したら，必ず同じパラメータで保管します。

ハンドラ関数を呼び出すには，関連する Web サービス呼び出しステップで **UserHandlerFunction** 引数の値として関数名を指定します。

```
web_service_call(
...
"UserHandlerFunction=MyScriptFunction",
"UserHandlerArgs=<handler arguments>",
LAST);
```

VuGen は以下のハンドラ関数のリターン・コードを認識します。

リターン・コード		説明
LR_HANDLER_SUCCEEDED	0	ハンドラは成功しましたが，SOAP エンベロープは変わりませんでした。
LR_HANDLER_FAILED	1	ハンドラが失敗し，その後の処理が停止されました。
LR_HANDLER_SUCCEEDED_AND_MODIFIED	2	ハンドラが成功し，更新された SOAP エンベロープが **SoapEnvelopeParam** に保管されています。

次の例では，スクリプト・ハンドラで送信エンベロープを操作します。

```
// この関数は SOAP エンベロープを処理してから，サーバに送信する。
int MyScriptFunction(const char* pArgs, int isRequest)
{
    if (isRequest == 1) {

        // 送信される要求を取得する
        char* str = lr_eval_string("{SoapEnvelopeParam}");

        // 文字列を操作する ...

        // 新しい要求内容を指定する
        lr_save_string(str, "SoapEnvelopeParam");

        return LR_HANDLER_SUCCEEDED_AND_MODIFIED;
    }
    return LR_HANDLER_SUCCEEDED;
}
Action()
{
    //web_service_call にハンドラを使用するよう指示する
    web_service_call( "StepName=EchoAddr_102",
        "SOAPMethod=SpecialCases.SpecialCasesSoap.EchoAddr",
        "ResponseParam=response",
        "userHandlerFunction=MyScriptFunction",
        "Service=SpecialCases",
        "Snapshot=t1174304648.inf",
        BEGIN_ARGUMENTS,
        "xml:addr="
            "<addr>"
                "<name>abcde</name>"
                "<street>abcde</street>"
                "<city>abcde</city>"
                "<state>abcde</state>"
                "<zip>abcde</zip>"
            "</addr>",
        END_ARGUMENTS,
        BEGIN_RESULT,
        END_RESULT,
        LAST);

    return 0;
```

DLL としてカスタムのユーザ・ハンドラを作成

Visual Studio とハンドラ API を通じて DLL を作成することで，ユーザ・ハンドラを定義することもできます。**LoadRunner/include** フォルダにある API ヘッダ・ファイル，**LrWsHandlerAPI.h** には，多くのインライン・コメントおよび説明が含まれています。

VuGen には，ハンドラ作成のテンプレートとして使用できるサンプルの Visual Studio プロジェクトが用意されています。このサンプルは，要求および応答のエンベロープを取得し，それをパラメータに保存します。このサンプルは，**LoadRunner/samples/WebServices/SampleWsHandler** フォルダにあります。このサンプルを使用するには，Visual Studio で開き，必要に応じて変更します。要求 / 応答をパラメータに保存する必要がなければ，サンプルのそのセクションを削除できます。

サンプルを編集したら，サンプルを保存し，DLL をコンパイルします。プロジェクトをコンパイルすると，Visual Studio によって **<user_handler_name>.DLL** ファイルが **LoadRunner/bin** フォルダに置かれます。プロジェクトを別の場所からコンパイルする場合，またはマシン間で DLL をコピーする場合は，必ず bin フォルダに置いてください。

ユーザ・ハンドラの設定

DLL ユーザ・ハンドラは，グローバルまたはローカルで宣言できます。

グローバルに，すなわちスクリプトのすべての要求でハンドラを使用するには，スクリプトのフォルダにある **default.cfg** ファイルに次のセクションを追加します。

```
[UserHandler]
Function=<name>
Args=<arguments>
Order=<BeforeSecurity/AfterSecurity/AfterAttachments>
```

- **Name**：DLL の名前。
- **Args**：ハンドラの設定引数のリスト。ハンドラの引数を取得するには，**GetArguments** メソッドを使用します。

➤ **Order**：Vuser が要求のユーザ・ハンドラを処理する順序，すなわち **Before Security**（セキュリティの前），**After Security**（セキュリティの後），**After Attachments**（添付ファイルの後）。この引数を使ってトランスポート層をオーバーライドするには，**Replace Transport** という値を入力します。

注：web_service_call 関数の **UserHandlerFunction** プロパティを設定すると，.cfg ファイルの定義がオーバーライドされます。

標準設定では，ユーザ・ハンドラはセキュリティの前に処理されます。要求メッセージの場合，Vuser は，セキュリティ・ハンドラの後で添付ファイル・ハンドラを処理します。応答メッセージの場合，Vuser は逆の順序でハンドラを処理します。一般的な場合には，順序が問題にならないため，任意の値が容認されます。

トランスポート層をオーバーライドするには，**Order=Replace Transport** を指定し，新しいトランスポート・ハンドラを指定します。トランスポート・ハンドラを別個の DLL として実装すると，**HandleRequest** 関数が呼び出され，**HandleResponse** 関数は無視されます。

ローカルに，すなわち特定の要求でハンドラを使用するには，**web_service_call** 関数に次の引数を追加します。

```
UserHandlerName=<name1>
UserHandlerArgs=<args1>
UserHandlerOrder=<BeforeSecurity/AfterSecurity/AfterAttachments/Replace
                  Transport>
```

注：スクリプトを別のマシンにコピーしても，ハンドラ情報は保持されます。これは，ハンドラ情報がスクリプトのフォルダに定義されているためです。スクリプトの特定のステップでローカルに定義されているユーザ・ハンドラは，グローバルなハンドラ設定（スクリプトの **default.cfg** ファイルで定義）に優先します。

ユーザ・ハンドラ DLL は，呼び出されるスクリプトを実行するすべての Load Generator マシンからアクセスできなければなりません。たとえば，ユーザ・ハンドラ DLL は **LoadRunner/bin** フォルダにコピーできます。

ユーザ・ハンドラの実装

ユーザ・ハンドラを実装するには，エントリ関数の **HandleRequest** または **HandleResponse** を使用します。2 つの関数には，単一のパラメータ，**context** があり，そのパラメータはハンドラで設定できます。プロパティを取得するには，Get 関数群を使用します。また，再生フレームワークからハンドラへ，あるいはハンドラ間で情報を渡すには，Set 関数群を使用します。

- **GetEnvelope**：エンベロープの内容を取得します。
  例：const char * pEnvelope = context->GetEnvelope();
- **GetEnvelopeLength**：エンベロープの長さを取得します。
- **SetEnvelope**：エンベロープの内容と長さを設定します。
  例：
  string str("MySoapEnvelope...");context->SetEnvelope(str.c_str(), str.length());
- **SetContentType**：HTTP ヘッダのコンテント・タイプに新しい値を設定します。
- **LogMessage**：再生ログへメッセージを発行します。
- **GetArguments**：DLL に渡すため，現在のハンドラに定義されている設定引数を取得します。
- **GetProperty**：ユーザ定義のプロパティ値を取得します。
- **SetProperty**：ユーザ定義のプロパティ値を設定します。

詳細については，**LoadRunner/include** フォルダにある **LrWsHandlerAPI.h** ファイルのコメントを参照してください。

ユーザ・ハンドラの例

次の項では，いくつかの一般的な問題に対するユーザ・ハンドラを作成する方法について説明します。

- .NET フィルタ
- トランスポート層のオーバーライド
- MIME 添付ファイルの包含

.NET フィルタ

Microsoft の Web Service Enhancements (WSE) 2.0 に精通していれば，.NET フィルタを作成して，受信または送信 SOAP メッセージに登録できます。.NET フィルタは，Microsoft.Web.Services2.SoapInputFilter または Microsoft.Web.Services2.SoapOutputFilter から派生するクラスです。このクラスの **ProcessMessage** 関数をオーバーライドすることで，エンベロープの本文およびヘッダを検査，変更できます。

.NET フィルタは，ユーザ・ハンドラ・メカニズムを利用してメッセージに適用できます。

スクリプト全体でフィルタをグローバルに定義するには，以下の行をスクリプトの default.cfg ファイルに追加します。

```
[UserHandler]
Function=LrWsSoapFilterLoader
Args=<Filters InputFilterClass="class name" InputFilterLib="lib name"
OutputFilterClass="class name" OutputFilterLib="lib name" />
Order=BeforeSecurity/AfterSecurity/AfterAttachments
```

InputFilterClass パラメータはクラスの名前を示し，**InputFilterLib** はクラスがあるアセンブリの名前を示します。たとえば，次のように設定します。

```
web_service_call(
    ...
    "UserHandlerName=LrWsSoapFilterLoader",
    "UserHandlerArgs=<Filters
InputFilterClass=¥"MyFilterNamespace.MyFilterClassName¥"
InputFilterLib=¥"MyAssemblyName¥" />",
    BEGIN_ARGUMENTS,
    ...
    END_ARGUMENTS,
    ...
    );
```

SoapOutputFilter を使用して，送信 **web_service_call** 要求を検査し，SoapInputFilter を使用して，サーバから要求を検査します。フィルタが SoapInputFilter から派生したものであれば，**InputFilterClass** および **InputFilterLib** を使用します。あるいは，フィルタが SoapOutputFilter から派生したものであれば，**OutputFilterClass** および **OutputFilterLib** を使用します。

特定のステップにフィルタを定義するには，**web_service_call** 関数に次の引数を追加します。

```
UserHandlerName= LrWsSoapFilterLoader

UserHandlerArgs=<Filters InputFilterClass=¥"class name¥" InputFilterLib=¥"lib name¥"
OutputFilterClass=¥"class name¥" OutputFilterLib=¥"lib name¥" />

UserHandlerOrder=BeforeSecurity/AfterSecurity/AfterAttachments
```

トランスポート層のオーバーライド

トランスポート層をオーバーライドするユーザ・ハンドラ関数を作成します。この場合は，VuGen によって，SOAP 要求が HTTP トランスポート上で自動的に送信されることはありません。代わりに，カスタム・ハンドラで指示されたトランスポート・メソッドに従います。

応答を受信したら，このコマンドを使用して応答エンベロープを設定できます。

```
lr_save_string(someResponseEnvelopeStr, "SoapEnvelopeParam");
```

別のトランスポート層を適用するには，**UserHandlerOrder** 引数の値として **ReplaceTransport** を指定し，ハンドラ関数でトランスポート層を定義します。

```
web_service_call(
...
"UserHandlerFunction=<Transport HandlerFunction>",
"UserHandlerArgs=<handler arguments>",
"UserHandlerOrder=ReplaceTransport"
...
LAST);
```

MIME 添付ファイルの包含

.NET ツールキットに基づいて Web サービス・スクリプトを使用するとき，インフラストラクチャでは MIME 添付ファイル がサポートされていません。ハンドラ・メカニズムを利用すると，.NET スクリプトに MIME 添付ファイル機能を追加できます。

以下の項では，.NET ツールキットで MIME 添付ファイルを送受信する方法について説明します。MIME 添付ファイルは同じ操作で送受信できます。

MIME 添付ファイルの送信

MIME 添付ファイルを送信するには，太字のコードを **web_service_call** に追加します。

```
web_service_call( "StepName=EchoComplex_101",
    "SOAPMethod=SimpleService|SimpleServiceSoap|EchoComplex",
    "ResponseParam=response",
    "Service=SimpleService",
    "UserHandlerName=LrWsAttachmentsHandler",
    "UserHandlerArgs=ATTACHMENT_ADD; ATTACHMENTS_FORMAT_MIME;
    ContentType=text/plain; FileName=C:¥¥temp¥¥results.discomap",
    "ExpectedResponse=SoapResult",
    "Snapshot=t1208947811.inf",
    BEGIN_ARGUMENTS,
    "xml:cls="
    "<cls>"
    "<i>123456789</i>"
    "<s>abcde</s>"
    "</cls>",
    END_ARGUMENTS,
    BEGIN_RESULT,
    END_RESULT,
    LAST);
```

FileName および **ContentType** パラメータを変更して，送信するファイルとそのコンテント・タイプを指定します。

MIME 添付ファイルの受信

MIME 添付ファイルを受信するには，次のコードを **web_service_call** に追加します。

```
"UserHandlerName=LrWsAttachmentsHandler",
"UserHandlerArgs=ATTACHMENT_SAVE_ALL;ParamNamePrefix=attach;"
```

MIME 添付ファイルの送受信

同じ **web_service_call** の MIME 添付ファイルを送受信するには，次のコードを追加します。

```
"UserHandlerName=LrWsAttachmentsHandler",
"UserHandlerArgs=ATTACHMENT_SAVE_ALL;ParamNamePrefix=attach;
ATTACHMENT_ADD; ATTACHMENTS_FORMAT_MIME; ContentType=text/plain;
FileName=C:¥¥temp¥¥results.discomap",
```

カスタム設定ファイルの使い方

設定ファイルでは，セキュリティ情報や WSE 設定などの詳細設定をカスタマイズできます。

標準の .NET 設定ファイル，**mmdrv.exe.config** は **LoadRunner/bin** フォルダにあります。

アプリケーションに独自の設定ファイル **app.config** がある場合，このファイルをいくつかの方法で実装できます。

- app.config を **mmdrv.exe.config** という名前で保存して，既存の設定ファイルを上書きします。これにより，マシン上のすべてのスクリプトに設定情報が適用されます。
- **app.config** をスクリプトのフォルダに保存します。**app.config** の設定は **mmdrv.exe.config** の設定に優先します。また，app.config をスクリプトのフォルダに保存すれば，常にスクリプトに関連付けられることになり，ほかのマシンに個別にコピーする必要がなくなります。

フィルタが SOAP 入力から導かれたものであれば，**Input** プレフィックスの付いたフィルタを使用し，フィルタが SOAP 出力から導かれたものであれば，**Output** プレフィックスの付いたフィルタを使用します。

さらに，設定ファイルには，セキュリティ情報も含まれます。テスト用の無署名の証明書を許可するかどうか設定できます。

標準設定では，テストを円滑にするために無署名の証明書が使用できます。無署名の証明書を不許可にするには，**mmdrv.exe.config** ファイルの **allowTestRoot** フラグを false に変更します。

```
<security>
  <x509 storeLocation="currentuser" alllowTestRoot="false"
```

第 13 章

Web サービス ― 異常系テスト

VuGen では，正常系テストや異常系テストなどのさまざまなテスト方法を使って Web サービスをテストできます。

本章の内容

以降の情報は，Web サービス /SOA Vuser スクリプトのみを対象とします。

テスト方法の適用について

Web サービスの機能テストを実行する場合は，さまざまな方法でテストに取り組む必要があります。最も一般的なテストは**正常系テスト**と呼ばれるもので，設計された内容をサービスが実行しているか検査します。

さらに，**異常系テスト**を行い，設計されていないタスクをアプリケーションが実行していないことを確認する必要があります。この場合は，アプリケーションが適切なエラー（SOAP エラー）を発行していることを確認する必要があります。

たとえば，入力データを受け入れるフォームについて考えてみましょう。Web サービスが名前とそれ以外の入力データを適切に受け取っているか検査するには，正常系テストを適用します。アプリケーションが不正な文字（電話番号欄の文字など）を検出しているか確認するには，異常系テストを適用します。

サービスがサーバに要求を送信すると，サーバは次のいずれかの方法で応答します。

- **SOAP 結果**：要求に対する SOAP 応答。
- **SOAP エラー**：SOAP 要求が無効であることを示す応答。異常系テストは SOAP エラーにのみ適用されます。
- **HTTP エラー**：「ページが見つかりません」などの HTTP エラーや Web サービスに関係のない HTTP エラー。

VuGen は，標準的な SOAP 結果応答や SOAP エラー応答を検査することはできますが，HTTP エラーでは常に失敗します。たとえば，Web サービスがアクセスを試みた Web ページが見つからなかった場合は，結果として 404 HTTP エラーが発生し，SOAP が有効であっても再生は失敗します。

テスト設定について

VuGen で Web サービス呼び出しや SOAP 要求を作成する際には，再生時に実行するテストのタイプを指定できます。

テストのタイプ	説明
正常系テスト	SOAP 結果応答を受け入れ，SOAP エラーで失敗します。
異常系テスト	SOAP エラーを受け入れ，SOAP 結果応答で失敗します。
すべて	SOAP 結果応答と SOAP エラー応答の両方を受け入れます。

標準設定では，VuGen は正常系テストのみ実行し，SOAP 結果応答を受け取った場合にテストが成功となります。異常系テストのみ実行する，あるいは，どの SOAP 応答も受け入れるように VuGen を指定することもできます。異常系テストのみ有効で，サーバが標準的な SOAP 結果応答を発行した場合，ステップのステータスは**失敗**となります。

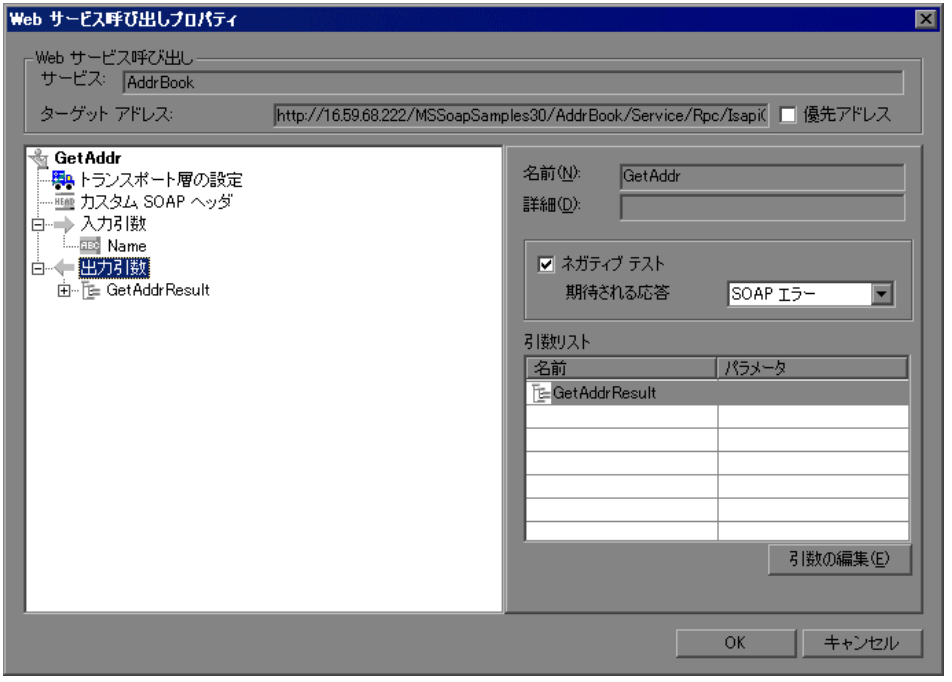
VuGen は，どの SOAP 応答（SOAP 結果および SOAP エラー）も受け入れるように指定できます。これは，要求の送信のみ必要で，SOAP の検査は別の関数を使って後で行うというテスト環境で役に立ちます。このテスト・モードでは，SOAP 結果または SOAP エラーを伴うステップには**成功**ステータスが発行されます。

再生ログ・レポートおよびテスト結果レポートでは，再生のステータスを確認できます。テスト結果レポートでは，失敗したステップは赤で**失敗**と表示されます。

Quality Center または HP Service Test Manager を使用している場合，アプリケーションは［期待される応答］の設定に基づいてテストのステータスを表示します。

テスト方法の定義

スクリプトを再生する前に，［Web サービス呼び出しプロパティ］ダイアログ・ボックスでテスト方法（正常系テスト，異常系テスト，すべての SOAP）を指定します。



テスト方法を選択するには，次の手順を実行します。

1 応答をテストするステップを選択します。右クリックして表示されるメニューから［プロパティ］を開きます。

2［**出力引数**］ノードを選択します。

3 期待される応答を選択します。

- 正常系テストのみ実行するには，［**ネガティブ テスト**］チェック・ボックスをクリアします。
- 異常系テストのみ実行するには，［**ネガティブ テスト**］チェック・ボックスを選択し，［**期待される応答**］で［**SOAP エラー**］を選択します。
- どのタイプの SOAP 応答も受け入れるには，［**ネガティブ テスト**］チェック・ボックスを選択し，［**期待される応答**］で［**任意の SOAP**］を選択します。

スクリプト・ビューでは，テスト方法が **ExpectedResponse** 引数で示されます。設定される値は，**SoapResult**，**SoapFault**，または **AnySoap** です。次のスクリプト例では，**SoapFault** 値で指定された異常系テストが実行されます。

```
web_service_call("StepName=AddAddr_101",
        "SOAPMethod=AddrBook|AddrBookSoapPort|AddAddr",
        "ResponseParam=response",
        "Service=AddrBook",
        "ExpectedResponse=SoapFault",
        "Snapshot=t1189409011.inf",
        BEGIN_ARGUMENTS,
        END_ARGUMENTS,
        BEGIN_RESULT,
        END_RESULT,
        LAST);
```

SOAP エラー値の評価

SOAP エラーが発生したスクリプトを再生すると，VuGen は，そのエラーを **response** というパラメータに保存します。返された SOAP エラーの値を確認するには，**lr_xml_find** を使用して **response** 出力パラメータを評価します。

次の例では，**lr_xml_find** が **VersionMismatch** SOAP エラーを検査し，出力メッセージを発行しています。

```
lr_xml_find("XML={response}",
            "FastQuery=/Envelope/Body/Fault/faultString ",
            "Value=VersionMismatch",
            LAST);

if (soap_fault_cnt >0)
    lr_output_message{"A Version Mismatch SOAP Fault occurred")
```

lr_xml_find の詳細については，『**Online Function Reference**』（英語版）（［**ヘルプ**］ ＞［**関数リファレンス**］）を参照してください。

第 14 章

Java プロトコル ー 記録

VuGen では，Java で書かれた，CORBA，RMI，EJB，JMS または Jacada などのプロトコルを使うアプリケーションまたはアプレットを記録できます。VuGen のナビゲーション・ツールを使用して，スクリプトに任意のメソッドを追加することもできます。

本章の内容

Java 言語 Vuser スクリプトの記録について

VuGen を使用して，Java アプリケーションまたはアプレットを記録できます。記録を行うと，VuGen によって，ピュア Java を Vuser API Java 固有の関数で拡張したスクリプトが生成されます。記録後，JDK ライブラリまたはユーザ定義クラスを使った標準 Java コードで，そのスクリプトの拡張や変更ができます。

スクリプトを用意したら，VuGen を使ってスタンドアロン・モードで実行します。Sun の標準 Java コンパイラ，**javac.exe** によってエラーのないことが確認された後，スクリプトがコンパイルされます。スクリプトが正常に機能することを確認したら，スクリプトを LoadRunner シナリオまたは Business Process Monitor プロファイルに組み込みます。

スクリプトを記録と手作業で拡張する場合，Java Vuser スクリプトに関連したすべてのガイドラインと制限が適用されます。また，スクリプトで使用するクラスは，Vuser を実行するマシンに存在している必要があり，**classpath** 環境変数に指定されている必要があります。関数の構文とシステムの設定で留意すべき点については，第 27 章「Java Vuser スクリプトのプログラミング」を参照してください。

CORBA セッションの記録を開始する前に，アプリケーションまたはアプレットが記録用のマシンで正しく動作することを確認します。

VuGen を実行するマシンに，Sun の JDK を正しくインストールしておく必要があります（JRE だけでは不十分です）。スクリプトを記録する前に，インストールを完了させておく必要があります。**classpath** および **path** 環境変数が，JDK のインストール時の指示に従って設定されていることを確認します。

注：記録中に VuGen でアプレットまたはアプリケーションをロードすると，VuGen を使わずにそれらをロードする場合よりも，多少時間がかかります。

VuGen には，Web 用に作成された Vuser スクリプトを Java に変換するツールがあります。詳細については，568 ページ「Web Vuser スクリプトの Java への変換」を参照してください。

記録後，JDK ライブラリまたはユーザ定義クラスを使った標準 Java コードで，そのスクリプトの拡張や変更ができます。

スクリプトを用意したら，VuGen を使ってスタンドアロン・モードで実行します。Sun の標準 Java コンパイラ，**javac.exe** によってエラーのないことが確認された後，スクリプトがコンパイルされます。

完成したスクリプトを自分の環境（LoadRunner シナリオ，Performance Center 負荷テスト，または Business Process Monitor プロファイル）に組み込みます。詳細については，**HP LoadRunner Controller**，**HP Performance Center**，または **HP Business Availibility Center** のドキュメントを参照してください。

Java Vuser スクリプトの概要

次の手順は，Java 言語 Vuser スクリプトの記録方法の概要です。

1 記録するマシンの設定が正しいことを確認します。

記録を開始する前に，マシンが Java を扱えるように正しく設定されていることを確認します。詳細については，第 27 章「Java Vuser スクリプトのプログラミング」と Readme ファイルを参照してください。

2 Vuser スクリプトを新規作成します。

Java Record/Replay Vuser タイプを選択します。

3 Java プロトコルを指定します。

記録オプションからプロトコルを選択します。

4 スクリプトの記録パラメータとオプションを設定します。

作業ディレクトリやパスなど，アプレットまたはアプリケーションに対するパラメータを指定します。JVM，シリアル化，相関，レコーダ，デバッグなどの記録オプションを設定することもできます。詳細については，『**第 1 巻 ー VuGen の使用**』の第 18 章「Java 記録オプション」を参照してください。

5 典型的なユーザ・アクションを記録します。

スクリプトの記録を開始します。アプレットまたはアプリケーションで一般的な操作をします。VuGen によってアクションが記録され，Vuser スクリプトが生成されます。

6 Vuser スクリプトを拡張します。

Vuser API 固有の関数を追加して，Vuser スクリプトを拡張します。詳細については，第 27 章「Java Vuser スクリプトのプログラミング」を参照してください。組み込みの Java Function Navigator（Java 関数ナビゲータ）を使用できます。詳細については，255 ページ「Java メソッドの表示」を参照してください。

7 Vuser スクリプトをパラメータ化します。

記録された定数をパラメータと置き換えます。文字列の全体または一部をパラメータ化できます。複数の引数を持つ関数には，複数のパラメータを定義できます。詳細については，『**第 1 巻 ― VuGen の使用**』の「VuGen パラメータをつ使った作業」を参照してください。

8 スクリプトの実行環境の設定を行います。

Vuser スクリプトの実行環境の設定を行います。実行環境の設定では，スクリプト実行時の環境を定義します。Java 固有の実行環境の設定については，『**第 1 巻 ― VuGen の使用**』の第 24 章「選択したプロトコルの実行環境の設定」を参照してください。

9 Vuser スクリプトを保存して実行します。

VuGen からスクリプトを実行して，実行時の情報を実行ログで確認します。詳細については，『**第 1 巻 ― VuGen の使用**』の「スタンドアロン・モードでの Vuser スクリプトの実行」を参照してください。

記録手順の詳細については，各 Vuser タイプの章を参照してください。

Java イベントの記録

Vuser を実行するマシンに，Sun が提供している JDK が正しくインストールされていることを確認してください（JRE だけでは十分ではありません）。**classpath** および **path** 環境変数が JDK のインストール手順に従って設定されていることを確認してください。Vuser スクリプトを再生する前に，JDK および関連 Java クラスに応じて環境が正しく設定されていることを確認してください。

次に示すのは，Java セッションを記録する基本的な手順です。

記録を開始するには，次の手順を実行します。

1 ［**ファイル**］＞［**新規作成**］を選択し，［**Java**］カテゴリから［**JAVA Record Replay**］を選択します。［記録開始］ダイアログ・ボックスが開きます。

2 ［**アプリケーションの種類**］ボックスで，以下の選択肢から適切な値を選択します。

- ［**Java アプレット**］: Sun のアプレットビューアを使って Java アプレットを記録します。
- ［**Java アプリケーション**］: Java アプリケーションを記録します。
- ［**Netscape**］または［**IExplore**］: ブラウザ内のアプレットを記録します。
- ［**Executable/Batch**］: バッチ・ファイルまたは実行ファイル名から起動されるアプレットまたはアプリケーションを記録します。

➤［**リスナ**］：記録の前に設定を初期化してアプリケーションを実行するバッチ・ファイルを待機するよう VuGen に指示します。このモードでは，JDK 1.2.x 以上を使って，システム変数 _JAVA_OPTIONS に値 **--Xrunjdkhook** を設定しなければなりません（JDK 1.1.x の場合，環境変数 **_classload_hook=JDKhook** を定義します。JDK 1.6 の場合は，_JAVA_OPTIONS を **-agentlib:jdhook** に設定します）。

3 次の表に従って，追加パラメータを指定します。

アプリケーションの種類	設定するフィールド
Java アプレット	［アプレット パス］，［作業ディレクトリ］
Java アプリケーション	［アプリケーション メイン］，［作業ディレクトリ］，［アプリケーション パラメータ］
IExplore	［IExplore パス］，［URL］
Netscape	［Netscape パス］，［URL］
Executable/Batch	［実行可能 / バッチ］，［作業ディレクトリ］
リスナ	なし

注：作業ディレクトリを指定する必要があるのは，アプリケーションに作業ディレクトリの場所を指定する必要がある（たとえば，プロパティ・ファイルの読み込みや，ログ・ファイルの作成をする）場合だけです。

4［**オプション**］をクリックして［記録オプション］ダイアログ・ボックスを開きます。Java プロトコル（CORBA，RMI，JMS，または Jacada）を選択し，それ以外の記録プロパティを設定します。詳細については，『**第 1 巻 ― VuGen の使用**』の第 18 章「Java 記録オプション」を参照してください。

5［アクション内に記録］ボックスで，記録を開始するメソッドを選択します。vuser_init，Actions，および vuser_end の各セクションに対応する **init**，**action**，および **end** の 3 つのセクションがあります。次の表に，各メソッドに何を含め，どのタイミングで呼び出されるかを示します。

Actions クラス内のメソッド	記録対象アクション	エミュレーション内容	実行のタイミング
init	**vuser_init**	サーバへのログイン	初期化時
action	**Actions**	クライアントの動作	実行中
end	**vuser_end**	ログオフ処理	終了時または停止時

注：必ず **vuser_init** セクションで **org.omg.CORBA.ORB** 関数をインポートすることで，この関数が反復のたびに呼び出されないようにします。

6［**OK**］をクリックして記録を開始します。VuGen によってアプリケーションが開始されます。VuGen は最小化され，進行状況バーとフローティング・ツールバーが開きます。進行状況バーには，そのときロードされているクラスの名前が表示されます。これは，Java 記録機能が動作中であることを示します。

7 アプリケーション内で，記録したい標準的な操作を行います。記録中にメソッドを切り替えるには，フローティング・ツールバーを使います。

8 標準的なユーザ・アクションを記録した後で，フローティング・ツールバーで **vuser_end** メソッドを選択します。

ログオフ手順を実行します。VuGen によって手順がスクリプトの **vuser_end** メソッドに記録されます。

9 記録ツールバーの［**記録停止**］をクリックします。VuGen スクリプト・エディタに，記録されたすべてのステートメントが表示されます。

［**保存**］をクリックし，スクリプトに名前を付けます。

CORBA の記録

CORBA セッションを記録する場合は，［記録オプション］で次のオプションを設定する必要があります。

- ［JNDI］
- ［DLL フックを使用して LoadRunner サポートをアタッチする］

CORBA アプリケーション・ベンダ・クラスの使用

JDK1.2 以降と Corba アプリケーションを組み合わせて実行すると，ベンダ固有の Corba クラスではなく，JDK 内部の Corba クラスがロードする場合があります。仮想マシンがベンダ固有の Corba クラスを必ず使用するようにするには，コマンド・ラインで次の java.exe パラメータを指定します。

Visigenic 3.4
-Dorg.omg.CORBA.ORBClass=com.visigenic.vbroker.orb.ORB
-Dorg.omg.CORBA.ORBSingletonClass=com.visigenic.vbroker.orb.
ORBSingleton

Visigenic 4.0
-Dorg.omg.CORBA.ORBClass=com.inprise.vbroker.orb.ORB
-Dorg.omg.CORBA.ORBSingletonClass=com.inprise.vbroker.orb.ORBSingleton

OrbixWeb 3.x
-Dorg.omg.CORBA.ORBClass=IE.Iona.OrbixWeb.CORBA.ORB
-Dorg.omg.CORBA.ORBSingletonClass=IE.Iona.OrbixWeb.CORBA.
singletonORB

OrbixWeb 2000
-Dorg.omg.CORBA.ORBClass=com.iona.corba.art.artimpl.ORBImpl
-Dorg.omg.CORBA.ORBSingletonClass=com.iona.corba.art.artimpl.
ORBSingleton

RMI over IIOP の記録

IIOP（**Internet Inter-ORB Protocol**）技術は，CORBA のソリューションをインターネット上で実装することを目的に開発されました。HTTP とは異なり，IIOP では，ブラウザとサーバが配列などの複雑なオブジェクトを交換できます。HTTP は，テキストの送信のみをサポートしています。

「**RMI-IIOP**」技術によって，これまで RMI または CORBA クライアントからのみアクセス可能だったサービスに，1 つのクライアントからアクセスできるようになります。この技術は，RMI で使用される JRMP プロトコルと，CORBA で使用される IIOP を組み合わせたものです。**RMI-IIOP** によって，CORBA クライアントは，EJB（**Enterprise Java Beans**）や，その他の J2EE 標準の新しい技術にアクセスすることができます。

VuGen では **RMI-IIOP** プロトコルを使用する Vuser の記録と再生を完全サポートします。記録する内容によりますが，VuGen の RMI レコーダを使用して実際のユーザを適切にエミュレートするスクリプトを作成できます。

- **ピュア RMI クライアント**：リモート呼び出しのためのネイティブの JRMP プロトコルを使用するクライアントの記録
- **RMI-IIOP クライアント**：（CORBA サーバに対応するために）JRMP の代わりに IIOP プロトコルを使用してコンパイルされたクライアント・アプリケーションの記録

RMI の記録

RMI セッションを記録する前に，記録するマシンでアプリケーションまたはアプレットが正しく機能することを確認してください。

記録を行う前に，使用する環境が正しく設定されていることを確認します。使用するクラスがクラスパスに含まれており，JDK の完全インストールが済んでいることを確認します。必要な環境設定の詳細については，第 27 章「Java Vuser スクリプトのプログラミング」を参照してください。

Jacada Vuser の記録

Jacada Interface Server は，メインフレーム・アプリケーションのためのインタフェース・レイヤを提供します。このレイヤは，ユーザ・インタフェースとアプリケーション・ロジックを分けることで，規格や技術の変更が組織に影響しないようにします。Jacada サーバでは，単色の端末画面のアプリケーションで作業するのではなく，環境をユーザ・フレンドリなインタフェースに変換します。

VuGen では，Jacada の Java シンクライアントを記録します。HTML シンクライアントを使用した Jacada サーバとの通信を記録するには，Web HTTP/HTML タイプの Vuser を使用します。詳細については，第 37 章「Web（HTTP/HTML, Click and Script）プロトコル」を参照してください。

また，再生前に，Jacada サーバから **clbase.jar** ファイルをダウンロードする必要があります。Java Vuser によって使用されるすべてのクラスが，マシンの CLASSPATH 環境変数，または，実行環境設定の［**Classpath**］パネルで設定されている **Classpath Entries** に含まれている必要があります。

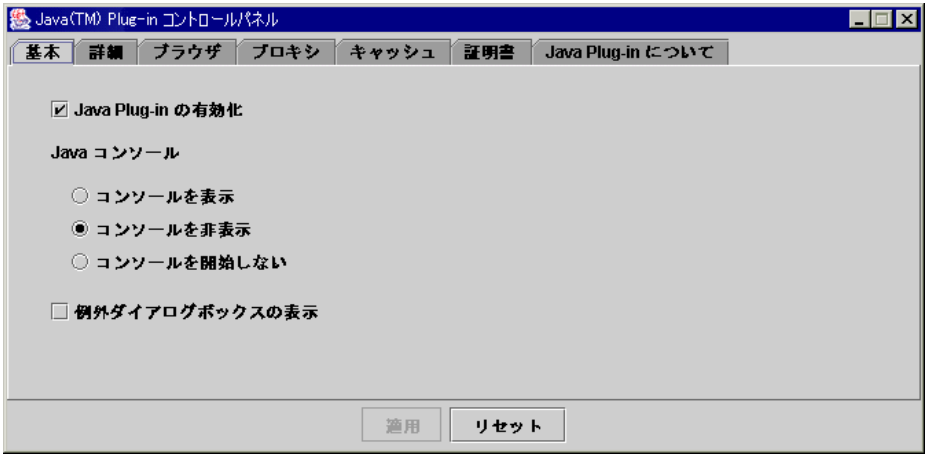
Jacada サーバは，再生中に，記録されたスクリプトにおける順序とは異なる順序でレガシー・システムの画面を戻すことがあります。これにより，再生時に例外が発生する可能性があります。この例外の扱い方については，サポート窓口までお問い合わせください。

Windows XP および Windows 2000 サーバでの記録

Windows XP および Windows 2000 サーバで記録を行う場合，Java プラグインが VuGen のレコーダとの互換性がないことがあります。正常に動作させるには，Java プラグインのインストール後，スクリプトの記録を開始する前に次の手順を実行します。

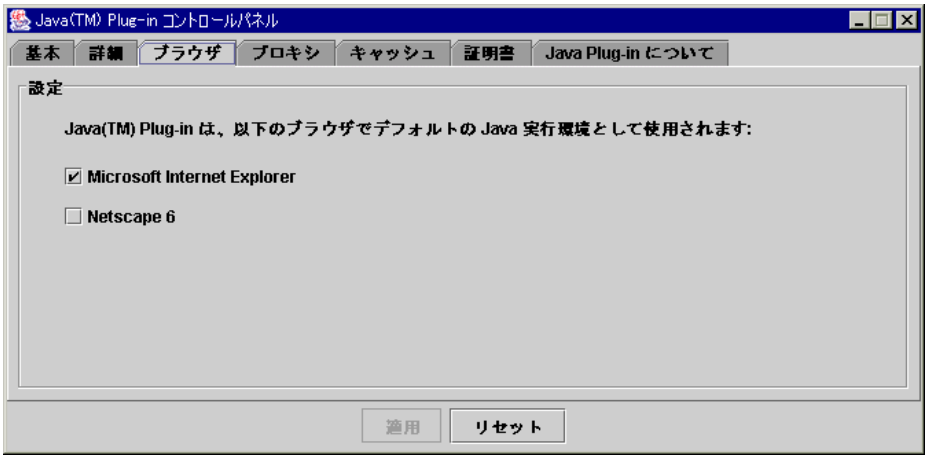
CORBA セッションまたは RMI セッションの記録用にマシンを設定するには，次の手順を実行します。

1 ［コントロール パネル］から［Java Plug-in］を開きます。［**スタート**］＞［**設定**］＞［**コントロール パネル**］を選択して，［**Java Plug-in**］コンポーネントを開きます。［基本］タブが開きます。



2 ［**Java Plug-In の有効化**］チェック・ボックスをクリアして，［**適用**］をクリックします。その後，［**Java Plug-In の有効化**］チェック・ボックスを選択しなおして，［**適用**］をクリックします。

3［ブラウザ］タブを開きます。



4［**Microsoft Internet Explorer**］チェック・ボックスをクリアして，［**適用**］をクリックします。その後，［**Microsoft Internet Explorer**］チェック・ボックスを選択しなおして，［**適用**］をクリックします。

第 15 章

Java － Vuser スクリプトの管理

VuGen では，Java で書かれたアプリケーションまたはアプレットを記録できます。記録したスクリプトは，そのまま実行したり，標準の Java ライブラリ関数および Vuser API Java 固有の関数を使って拡張したりすることができます。

本章の内容

Java Vuser スクリプトについて

セッションを記録すると，VuGen によってサーバに対するすべての呼び出しが記録され，関数を含んだスクリプトが生成されます。これらの関数は，アプリケーションまたはアプレットにおけるユーザのすべてのアクションを表します。スクリプトにはプロパティの設定やネーミング・サービスの初期化（JNDI）など，適切な再生に必要な追加コードも含まれています。

記録されたスクリプトは，次の 3 つのセクションで構成されています。

- インポート
- コード
- 変数

インポート・セクションはスクリプトの先頭にあります。ここにはスクリプトのコンパイルに必要なすべてのパッケージへの参照が含まれます。**コード**・セクションには，Actions クラスと，**init**，**actions**，および **end** メソッド内に記録されたコードが含まれます。**end** メソッドの後の**変数**セクションには，コードで使用される変数のすべての型宣言が含まれます。

記録終了後，スクリプト内の関数に変更を加えたり，Java または LoadRunner の関数を追加してスクリプトを拡張したりできます。Java Vuser をスレッドとして実行する場合，スクリプトに追加する Java コードはスレッドセーフでなければなりません。関数の構文の詳細については，『**Online Function Reference**』（英語版）（［**ヘルプ**］＞［**関数リファレンス**］）を参照してください。さらに，スクリプトを別のパッケージの一部として実行できるように変更することも可能です。詳細については，446 ページ「パッケージの一部としてのスクリプトのコンパイルと実行」を参照してください。

CORBA を使った作業

通常，CORBA 固有のスクリプトには，明確なパターンがあります。最初のセクションには，ORB の初期化処理と設定が含まれています。次のセクションでは，CORBA オブジェクトの場所を指定します。その次のセクションは，CORBA オブジェクトでのサーバ呼び出しで構成されます。最後のセクションには，ORB を閉じるシャットダウンの処理が含まれています。必ずこのパターンに従わなければならないわけではありません。また，これらのセクションは 1 つのスクリプト内に複数現れることがあります。

次に示すスクリプトのコードでは，ORB インスタンスを初期化し，バインドの処理を実行して CORBA オブジェクトを取得しています。VuGen で必要なクラスをすべてインポートしている点に注目してください。

```
import org.omg.CORBA.*;
import org.omg.CORBA.ORB.*;
import lrapi.lr;

public class Actions {

    // Public 関数：init
    public int init() throws Throwable {

    // Orb インスタンスを初期化する ...
    MApplet mapplet = new MApplet("http://chaos/classes/", null);
    orb = org.omg.CORBA.ORB.init(mapplet, null);

    // サーバへのバインド
    grid = grid_dsi.gridHelper.bind("gridDSI", "chaos");
    return lr.PASS;
}
```

org.omg.CORBA.ORB 関数は，ORB への接続を行います。そのため，この関数は 1 回だけ呼び出します。複数の反復を実行するときは，この関数を **init** セクションに置きます。

次に示すコードでは，CORBA オブジェクト（grid）を対象に実行したアクションが記録されています。

```
// public 関数：action
public int action() throws Throwable {

    grid.width();
    grid.height();
    grid.set(2, 4, 10);
    grid.get(2, 4);

    return lr.PASS;
}
```

セッションの最後に，VuGen によって ORB のシャットダウンが記録されました。記録されたコード全般に渡って使用された変数は，**end** メソッドの末尾から Actions クラスの終了の括弧（ } ）までの間に現れます。

```
// public 関数：end
    public int end() throws Throwable {

        if ( lr.get_vuser_id() == -1 )
            orb.shutdown();

        return lr.PASS;
}

// 変数セクション
    org.omg.CORBA.ORB orb;
    grid_dsi.grid grid;
}
```

ORB シャットダウン・ステートメントは本製品用にカスタマイズされています。このカスタマイズは，1 つの Vuser のシャットダウンによってほかのすべての Vuser がシャットダウンされないようにするものです。

RMI を使った作業

本項では，RMI 特有の Java Vuser スクリプトの要素について説明します。RMI は CORBA のような構造要素はなく，Serializable Java オブジェクトを使用します。最初のセクションでは，ネーミング・レジストリの初期化と設定を行います。次のセクションは，Java オブジェクト（Remote および Serializable）が見つかり，キャストされた場合に生成されます。その次のセクションには，Java オブジェクトを対象とするサーバ呼び出しが含まれます。RMI は CORBA とは異なり，専用のシャットダウン・セクションがありません。スクリプトの中でオブジェクトが何度も現れることがあります。

次に示すコードでは，ネーミング・レジストリを検索しています。この処理の後に，特定の Java オブジェクトを取得するための検索・取得処理が行われます。オブジェクトを取得したら，それを使用して **set_sum**，**increment**，**get_sum** などの呼び出しを実行できます。このコード例は，VuGen で必要とされるすべての RMI クラスをどのようにインポートするかを示します。

```
Import java.rmi.*;
Import java.rmi.registry.*;

:
:

// public 関数：action
public int action() throws Throwable {

    _registry = LocateRegistry.getRegistry("localhost",1099);

    counter = (Counter)_registry.lookup("Counter1");

    counter.set_sum(0);
    counter.increment();
    counter.increment();
    counter.get_sum();

    return lr.PASS;
    }
:
```

RMI Java を記録する際，スクリプトにはすべての関連オブジェクトのシリアル化を解除する **lr.deserialize** への複数の呼び出しが含まれていることがあります。**lr.deserialize** 呼び出しは，次の呼び出しに渡されるオブジェクトが，それ以前の呼び出しの戻り値と相関できない場合に生成されます。この場合，VuGen によってオブジェクトの状態が記録され，再生時に **lr.deserialize** 呼び出しを使って，オブジェクトの値が提示されます。シリアライズの解除は，VuGen によってオブジェクトがパラメータとして呼び出しに渡される前に行われます。詳細については，276 ページ「シリアル化メカニズムの使用」を参照してください。

Jacada を使った作業

Jacada が使用された Java Vuser スクリプトの Actions メソッドには，2 つの主要な部分，プロパティと本体があります。プロパティ部分では，サーバのプロパティを取得します。その後 VuGen によって，システム・プロパティが設定され，Jacada サーバへの接続が行われます。

```
// システム・プロパティを設定する
    _properties = new Properties(System.getProperties());
    _properties.put("com.ms.applet.enable.logging", "true");
    System.setProperties(_properties);

    _jacadavirtualuser = new cst.client.manager.JacadaVirtualUser();

    lr.think_time(4);
    _jacadavirtualuser.connectUsingPorts("localhost", 1100, "LOADTEST", "", "", "");
    ...
```

スクリプトの本体には，ユーザ・アクションのほか，checkFieldValue メソッドと checkTableCell メソッドの例外処理ブロックが含まれています。

```
      l...
/*
try {
     _jacadavirtualuser.checkFieldValue(23, "S44452BA");
     }catch( java.lang.Exception e ) {
      lr.log_message(e.getMessage());
     }
*/   l...
/*
try {
     _jacadavirtualuser.checkTableCell(41, 0, 0, "");
     }catch( java.lang.Exception e ) {
      lr.log_message(e.getMessage());
     }
*/   l...
```

checkField メソッドには，フィールド ID 番号と期待値の 2 つの引数があります。**checkTableCell** メソッドには，テーブル ID，行，カラム，期待値の 4 つの引数があります。期待値と戻り値の間に不一致がある場合は，例外が生成されます。

標準設定では，try-catch 例外処理ブロックはコメントになっています。これをスクリプトで使用するには，コメントの印を削除してください。

記録されたスクリプトには，任意の Java Vuser API 関数を追加できます。これらの関数の一覧とスクリプトへの追加方法については，第 27 章「Java Vuser スクリプトのプログラミング」を参照してください。

パッケージの一部としてのスクリプトの実行

この項は，Jacada タイプのスクリプトには適用されません。

Java Vuser スクリプトを作成または記録するときに，メソッドまたはクラスが保護されているクラスのメソッドを使用する必要が生じる場合があります。そのようなスクリプトをコンパイルすると，メソッドにアクセスできないことを示すコンパイル・エラーを受け取ることになります。

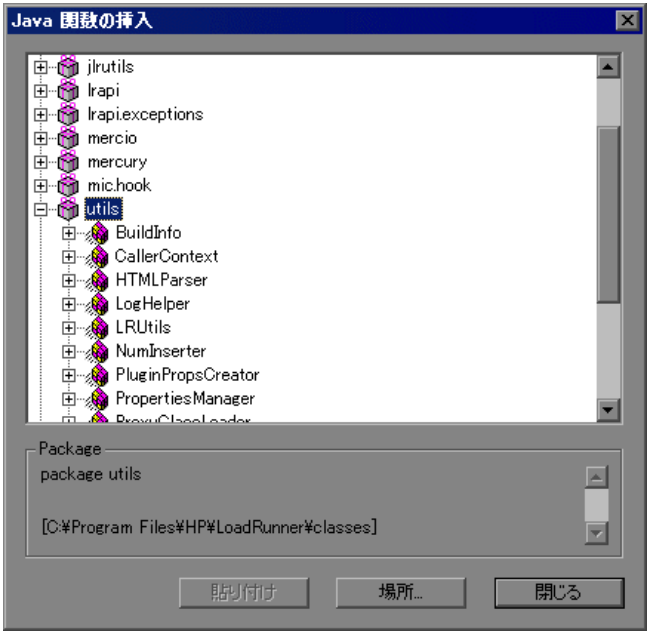
保護されているメソッドを Vuser で使用するには，必要なメソッドのパッケージにその Vuser を追加します。スクリプトの先頭に次の行を追加します。

```
package a.b.c;
```

ここで，**a.b.c** はディレクトリ階層を表します。VuGen はユーザ・ディレクトリにディレクトリ階層 a/b/c を作成し，そこで **Actions.java** ファイルをコンパイルして，パッケージの一部にします。**package** ステートメントは記録されません。手作業で挿入する必要があります。

Java メソッドの表示

VuGen では，アプリケーションのパッケージに含まれているすべての Java クラスとメソッドを参照できるナビゲータが利用できます。



クラスまたはメソッドをスクリプトに挿入するには，クラスまたはメソッドを選択してスクリプトに貼り付けます。詳細な手順については，257 ページ「手作業による Java メソッドの挿入」を参照してください。

ダイアログ・ボックスの下部には，Java オブジェクトの詳細，プロトタイプ，戻り値，およびパスが表示されます。次の例に示す詳細情報は，deserialize メソッドが，文字列と整数の 2 つのパラメータを取る public な静的クラス・メソッドであることを示します。このメソッドは java.lang.object を返し，例外をスローします。

```
public static synchronized java.lang.Object deserialize (java.lang.String, int) throws
Exception
```

次の表に，各種の Java オブジェクトを表すアイコンの説明を示します。

アイコン	項目	例
	パッケージ	java.util
	クラス	public class Hashtable extends java.util.Dictionary implements java.lang.Cloneable, java.io.Serializable
	インタフェース・クラス (灰色のアイコン)	public interface Enumeration
	メソッド	public synchronized java.util.Enumeration keys ()
	静的メソッド (黄色のアイコン)	public static synchronized java.util.TimeZone getTimeZone
	コンストラクタ・メソッド	public void Hashtable ()

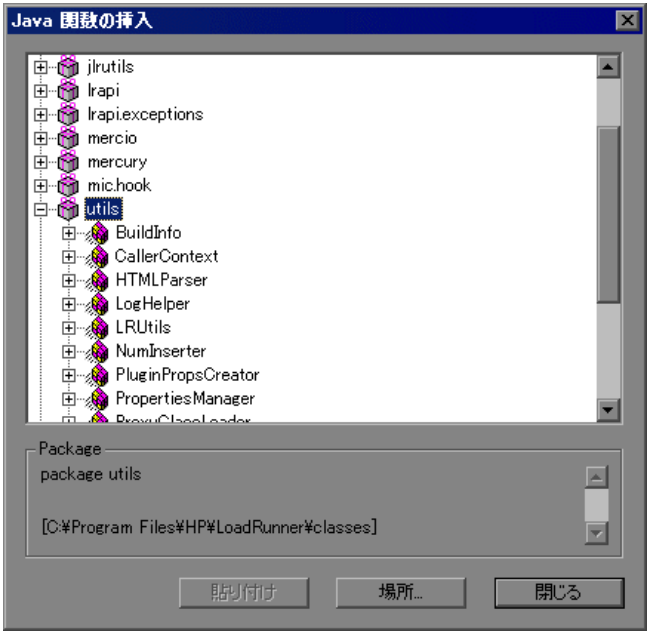
手作業による Java メソッドの挿入

Java 関数ナビゲータを使用して Java 関数の表示やスクリプトへの追加をします。本項の情報は，EJB テストおよび Java 記録 / 再生 Vuser に適用されます。設定ファイルを変更して，関数の生成についての設定をカスタマイズできます。詳細については，259 ページ「スクリプト生成の設定」を参照してください。

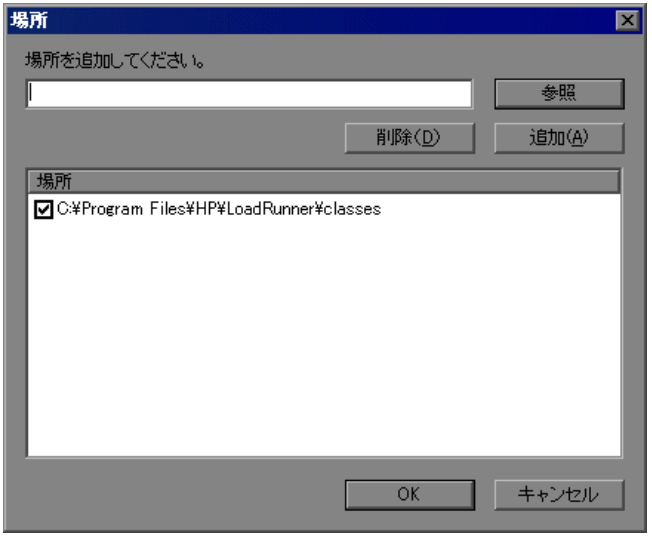
Java 関数を挿入するには，次の手順を実行します。

1 スクリプトの中で，挿入を行う場所をクリックします。関数の貼り付けを行うと，VuGen によってカーソルの位置に関数が貼り付けられます。

2 ［**挿入**］＞［**Java 関数の挿入**］を選択します。［Java 関数の挿入］ダイアログ・ボックスが表示されます。



3 [**場所**] をクリックします。[場所] ダイアログ・ボックスが表示されます。標準設定では，CLASSPATH 環境変数に定義されているパスの一覧が表示されます。



4 [**参照**] をクリックして別のパスまたはアーカイブをリストに追加します。パスを追加するには，[**参照**] ＞ [**フォルダ**] を選択します。アーカイブ（**jar** または **zip**）を追加するには，[**参照**] ＞ [**ファイル**] を選択します。フォルダまたはファイルを選択すると，VuGen によって [**場所を追加してください。**] ボックスにその名前が挿入されます。

5 [**追加**] をクリックして，リストに項目を追加します。

6 追加するパスまたはアーカイブごとに手順 4 と 5 を繰り返します。

7 リスト項目の左にあるチェック・ボックスを選択するかクリアします。選択した項目のメンバのリストが，Java クラス・ナビゲータに表示されます。

8 [**OK**] をクリックして [場所] ダイアログ・ボックスを閉じると，使用可能なパッケージが表示されます。

9 ナビゲータの各項目の左にあるプラスとマイナスの記号をクリックして，ツリーの分岐の表示と非表示を切り替えます。

10 オブジェクトを選択し，[**貼り付け**] をクリックします。VuGen によってスクリプトのカーソルの位置にオブジェクトが挿入されます。1 つのクラスのすべてのメソッドをスクリプトに貼り付けるには，そのクラスを選択して [**貼り付け**] をクリックします。

11 使用するすべてのメソッドとクラスについて，手順 10 を繰り返します。

12 メソッドのパラメータを変更します。スクリプト生成の設定で **DefaultValues** を **true** に設定しておくと，VuGen によって挿入される標準設定値を使用できます。**DefaultValues** を **false** に設定すると，スクリプトに挿入するすべてのメソッドについてパラメータを追加する必要があります。

次に戻り値を変更します。たとえば，スクリプトで「(String)=LavaVersion.getVersionId();」というステートメントが生成された場合，(String) を文字列型変数に置き換えます。

13 import ステートメントや Vuser API Java 関数（第 27 章「Java Vuser スクリプトのプログラミング」で説明）など，必要な任意のステートメントをスクリプトに追加します。

14 スクリプトを保存して，VuGen から実行します。

スクリプト生成の設定

スクリプトの次の要素について，ナビゲータによるメソッドの追加方法をカスタマイズできます。

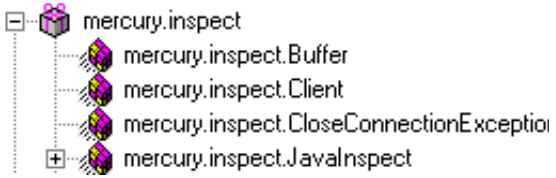
- クラス名のパス
- 自動トランザクション
- 標準のパラメータ値
- クラスの貼り付け

設定を表示するには，VuGen の dat ディレクトリにある **jquery.ini** ファイルを開きます。

```
[Display]
FullClassName=False

[Insert]
AutoTransaction=False
DefaultValues=True
CleanClassPaste=False
```

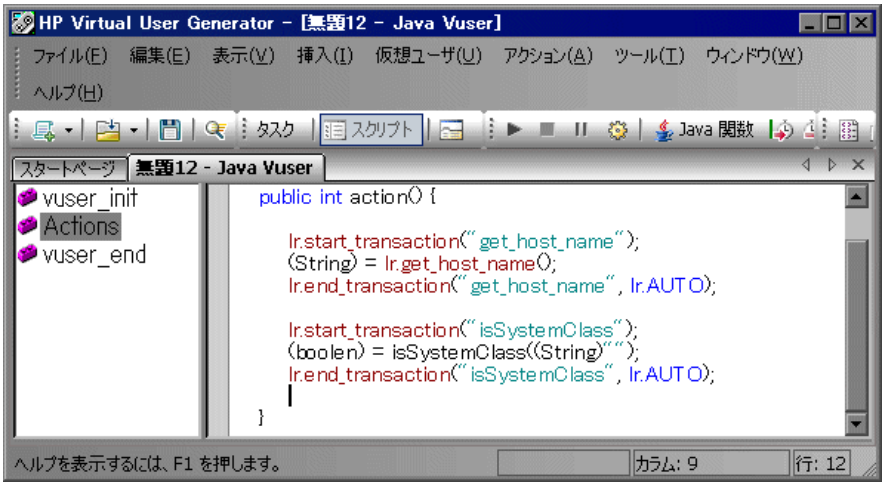
クラス名のパス

FullClassName オプションを有効にすると，Java 関数ナビゲータにパッケージとクラスの完全な名前が表示されます。このオプションは関数がスクリプトにどのように追加されるかには影響しません。ナビゲータでのクラスの表示が変わるだけです。標準設定では，このオプションは false に設定されています。パッケージに多数のクラスが含まれていてパッケージとクラスの名前が同時に見えない場合は，このオプションを有効にします。

FullClassName が有効	FullClassName が無効
	

自動トランザクション

AutoTransaction を有効にすると，スクリプトに貼り付けるすべてのメソッドについて Vuser トランザクションが作成されます。このオプションを有効にすると，VuGen によってすべての Java メソッドが自動的に **lr.start_transaction** 関数と **lr.end_transaction** 関数で囲まれます。これにより，各メソッドのパフォーマンスを個別に追跡できます。標準設定ではこのオプションは無効です。



標準のパラメータ値

DefaultValues を有効にすると，スクリプトに貼り付けるすべてのメソッドが標準設定の値を持ちます。標準設定ではこのオプションは有効で，すべてのオブジェクトについて null が挿入されます。このオプションを無効にした場合は，スクリプトのすべての関数のパラメータ値を手作業で挿入する必要があります。次の表に，DefaultValues フラグが有効になっている場合と無効になっている場合を示します。

DefaultValues が有効	DefaultValues が無効

lr.message((String)"");
lr.think_time((int)0);
lr.enable_redirection((boolean)false);
lr.save_data((byte[])null, (String)"");
	lr.message((String));
	lr.think_time((int));
	lr.enable_redirection((boolean));
	lr.save_data((byte[]), (String));

クラスの貼り付け

CleanClassPaste を有効にすると，クラスがエラーなくコンパイルされるように貼り付けられます。つまり，コンストラクタからインスタンスが返され，パラメータに標準の値が設定され，import ステートメントを必要としません。このオプションを使うと，多くの場合，ほかの修正をせずにスクリプトを実行できます。このオプションが無効の場合（標準設定），パラメータを手作業で定義し，import ステートメントを含める必要があります。この設定が適用されるのは，1 つのメソッドでなく，クラス全体をスクリプトに貼り付ける場合だけです。

次のコードは，CleanClassPaste オプションを有効にしてスクリプトに toString メソッドを貼り付けた場合を示します。

```
_class.toString();
    // 戻り値 :java.lang.String
```

CleanClassPaste オプションが無効の場合，同じメソッドが次のように貼り付けられます。

```
(String) = toString();
```

次のコードは，CleanClassPaste オプションを有効にしてスクリプトに **NumInserter** コンストラクタ・メソッドを貼り付けた場合を示します。

```
utils.NumInserter _numinserter = new utils.NumInserter
            ((java.lang.String)"", (java.lang.String)"", (java.lang.String)""...;);
// 戻り値 :void
```

CleanClassPaste オプションを無効にすると，同じメソッドが次のように貼り付けられます。

```
new utils.NumInserter((String)"", (String)"", (String)"",...);
```

Java ユーザ定義フィルタ

Java アプリケーションをテストする目的は，クライアント要求に対するサーバの反応を確認することです。負荷テストの場合には，多数のユーザによる負荷にサーバがどのように応答するかを確認します。VuGen の Java Vuser を使用して，サーバと通信するクライアントをエミュレートするスクリプトを作成します。

VuGen には，よく使用されるメソッドにフック・プロパティを定義するフィルタ・ファイルがあります。RMI，CORBA，JMS，JACADA の各プロトコル用のフィルタ定義が用意されています。また，後述の手順に従ってユーザ定義フィルタを作成することもできます。

メソッドを記録する場合，記録されるメソッドから直接または間接的に呼び出されるメソッドは記録されません。

VuGen は，メソッドを記録するために，メソッドの引数とともに，メソッドが呼び出されたオブジェクトを認識する必要があります。次の条件が満たされるのであれば，記録される別のメソッドによってオブジェクトが返される場合，VuGen はそのオブジェクトを認識します。

- そのオブジェクトの構築メソッドがフックされています。
- プリミティブまたは組み込みオブジェクトです。
- シリアル化可能なインタフェースをサポートしています。

ユーザ定義フィルタを作成して，不要なメソッドを除外できます。Java アプリケーションを記録すると，スクリプトにはローカル・ユーティリティや GUI インタフェースの呼び出しなど，サーバに影響を与えないメソッドの呼び出しが含まれる場合があります。通常，こうした呼び出しはテストの目的には関係ないので，フィルタによって除外するのが適切です。

RMI，CORBA，JMS，JACADA の各プロトコルの組み込みフィルタは，テストの目的にかなうサーバ関連のトラフィックのみを記録するように作成されています。ただし，状況によっては，Java アプリケーションの呼び出しをキャプチャしたり不要な呼び出しを除外したりするために，ユーザ定義のカスタム・フィルタが必要になることがあります。ユーザ定義 Java プロトコル，標準設定プロトコルの独自の機能強化および拡張，データ抽象化では，ユーザ定義のフィルタ定義が必要になります。

フィルタ設定についてのガイドライン

記録に何を含めるかを判断できるように，テストを作成する前にアプリケーションについて理解を深め，アプリケーションの主要なクラスやメソッドを確認することをお勧めします。

アプリケーションのクラスに精通していない場合は，アプリケーションによって呼び出されたすべてのメソッドをログに記録するスタック・トレース付きで記録することが可能です。スタック・トレース付きで記録するには，ログ・レベルを**詳細**に設定します。詳細については，265 ページ「効果的なフィルタの定義」を参照してください。

必要なメソッドとクラスを確認したら，**user.hooks** ファイルを更新してそれらを包含するようにします。スクリプトを準備するときに，最適なフィルタにするためにフィルタを数回カスタマイズしなければならない場合があります。フィルタが最適であれば，スクリプトに無関係の多数の呼び出し取り込まずに必要なメソッドが記録されます。

注：フロー制御やメッセージ・ステートメントなど，スクリプトにコードを手作業で追加する場合は，必ず VuGen 内での実行が可能なスクリプトが得られてからにしてください。その理由は，フィルタの変更後にスクリプトを再記録すると，手作業で行ったすべての変更が上書きされてしまうためです。

包含または除外する要素の指定

ユーザ定義フィルタを作成するとき，基本のフィルタとして適切な組み込みフィルタを選択するところから始めることをお勧めします。その後，次のどちらかの方法を使用してフィルタをカスタマイズできます。

- **トップ・ダウン方式**：この方式では，関係するパッケージを包含し，クライアント / サーバの動作に関与しない個別のクラスを除外します。アプリケーションに精通しており，GUI 要素が関与しないクライアント / サーバの動作をすべて実装する明確なレイヤを特定できる場合には，この方法をお勧めします。
- **ボトム・アップ方式**：標準設定のフィルタを使用し，個別のメソッドまたはクラスを包含するようにしてフィルタを調整していく方式です。明確なレイヤを特定できない場合，またはアプリケーションに精通していない場合は，この方式を使用します。すべての AUT パッケージを追加して，その後余分なコンポーネントを 1 つずつ削除するようなことはしないでください。

次の項では，要素を包含または除外する際のガイドラインを示します。

- クラスを包含した結果，スクリプトに多くの無関係のメソッド呼び出しが含まれた場合は，フィルタに変更を加えて無関係のメソッドが除外されるか試してみます。
- スクリプトにクライアント／サーバの動作にかかわらない呼び出しが含まれた場合，フィルタからそのメソッドを除外します。
- 記録中，VuGen はこれまで遭遇したことのない構造の引数など，未知の入力引数を検出する場合があります。この引数がシリアライズをサポートしている場合，VuGen は引数を特別な形式で引数をファイルに保存することでその引数をシリアライズします。再生中，VuGen は引数をシリアル化解除することでそれを復元します。
- VuGen は，フィルタによって包含されなかった引数として渡されたオブジェクトをシリアライズします。オブジェクトの構造と動作を追跡するために，オブジェクトをシリアライズされた形式で使用するのではなく，フィルタに含めることをお勧めします。スクリプト内のシリアライズされたオブジェクトを特定するには，スクリプトで **lr.deserialize()** メソッドへの呼び出しを検索します。詳細については，276 ページ「シリアル化メカニズムの使用」を参照してください。
- GUI 要素が関与する動作はすべて除外します。
- スクリプトのコンパイルに必要なユーティリティ用のクラスを包含するようにします。

効果的なフィルタの定義

スクリプトを準備するときに，最適なフィルタにするためにフィルタを数回カスタマイズしなければならない場合があります。フィルタが最適であれば，スクリプトに無関係の多数の呼び出し取り込まずに必要なメソッドが記録されます。

効果的なフィルタを定義するには，次の手順を実行します。

1 製品の **classes** ディレクトリにある **user.hooks** ファイルを変更することによって，いずれかの組み込みフィルタに基づいた新しいフィルタを作成します。

2 ［記録オプション］を開き（Ctrl+F7），［**ログ オプション**］ノードを選択します。［ログ レベル］を［**詳細**］に設定します。

3 アプリケーションを記録します。［**記録開始**］（Ctrl+R）をクリックすると記録が始まり，［**停止**］（Ctrl+F5）をクリックすると終了します。

4 スクリプトのステップを表示します。ステップを見てビジネス・ロジックを特定し相関を適用できる場合には，カスタム・フィルタを作成する必要はありません。しかし，スクリプトが非常に長かったり保守や相関が困難だったりする場合には，スクリプトのフィルタをカスタマイズする必要があります。

5 1 つ以上のクライアント・サーバ呼び出しをキャプチャまたはラップする呼び出しの中で，高水準のメソッドの識別を試みます。そのためには，AUT ソース・ファイルを開くか（使用可能な場合），スクリプトのスタック・トレースを表示します。

6 関係するメソッドを包含するようにフィルタを設定します。詳細については，264 ページ「包含または除外する要素の指定」を参照してください。

7 アプリケーションを記録しなおします。フィルタを変更したら必ずアプリケーションを記録しなおしてください。

8 容易に保守と相関が行える簡単なスクリプトになるまで，手順 4 から 7 を繰り返します。

9 スクリプトを相関させます。テストを正しく実行するために，相関を挿入して値をキャプチャし，スクリプトの以降の場所で使用する必要がある場合があります。組み込みの相関メカニズムの詳細については，第 16 章「Java ー 相関」を参照してください。

注：VuGen レコーダが壊れることがあるため，ほかのどの .hooks ファイルも変更しないでください。

標準設定レコーダへのユーザ定義フックの追加作業は複雑であり，機能とパフォーマンスの両方で影響があるため，入念に検討する必要があります。

フック定義を誤ると，不適切なスクリプト，記録速度の低下，アプリケーションのフリーズを招くおそれがあります。

フック・ファイルの構造

次の項では，標準的なフック・ファイルの構造について説明します。

```
[Hook-Name]
class           = MyPackage.MyClass
method          = MyMethod
signature       = ()V
ignore_cl       =
ignore_mtd      =
ignore_tree     =
cb_class        = mercury.ProtocolSupport
cb_mtd          =
general_cb      = true
deep_mode       = soft | hard
make_methods_public= true | false
lock            = true | false
```

フック・ファイルは，各セクションがフック定義を表す .ini ファイルとして構成されます。一部のエントリでは正規表現がサポートされています。正規表現が使用されたエントリは '!' で始める必要があります。

Hook-Name

フック・ファイルのこのセクションの名前を示します。Hook-Name は，すべてのフック・ファイルで一意でなければなりません。完全修飾クラス名およびメソッドを指定することをお勧めします。次に例を示します。

[javax.jms.Queue.getQueueName]

クラス

完全修飾クラス名。正規表現を使用して，同じパッケージ，パッケージ全体，または複数のパッケージの複数のクラス，あるいは名前が一致した任意のクラスを含めることができます。次に例を示します。

Class = !javax¥.jms¥.*

Method

含めるメソッドの単純名。正規表現を使用して，クラスの複数のメソッドを含めることができます。次に例を示します。

Method = getQueueName

Signature

メソッドの標準的な Java 内部型シグネチャ。メソッドのシグネチャを指定するには，javap -s class-name というコマンドを実行します（class name はクラスの完全修飾名）。正規表現を使用して，名前が同じで引数が異なる複数のメソッドを含めることができます。次に例を示します。

Signature = !.*

ignore_cl

このフックに一致するクラスの中から無視する特定のクラス。カンマで区切ったクラス名のリストを指定することもできます。リストの各項目には正規表現を含めることができます。リストの項目に正規表現を含める場合は，クラス名の前に '!' を付けてください。次に例を示します。

Ignore_cl = !com.hp.jms.Queue,!com¥.hp¥..*

ignore_mtd

無視する特定のメソッド。ロードされたクラス・メソッドがこのフック定義に一致した場合，そのメソッドはフックされません。メソッド名は，後にシグネチャ（前述）が続く単純メソッド名でなければなりません。複数のメソッドを無視するには，カンマ区切りリストで指定します。正規表現を使用するには，メソッド名の前に '!' を付けてください。次に例を示します。

Ignore_cl = open, close

ignore_tree

無視する特定のツリー。無視するツリーの式とクラス名が一致した場合，そのクラスを継承したクラスは，このフック定義に一致していたとしてもフックされません。複数のツリーを無視するには，カンマ区切りリストで指定します。正規表現を使用するには，クラス名の前に '!' を付けてください。このオプションは，「深い」と定義されているフックにのみ使用できます。

cb_class

フックされたメソッドからの呼び出しを取得するコールバック・クラス。常に **mercury.ProtocolSupport** に設定する必要があります。

cb_mtd

フックされたメソッドからの呼び出しを取得するコールバック・クラスのメソッド。省略した場合は，標準設定の **general_rec_func** が使用されます。呼び出しのサブツリーをロックするだけでよい場合は，**general_func** を使用します。

general_cb

一般的なコールバック・メソッド。この値は，常に **true** に設定する必要があります。

Deep_mode

深いモードでは，フックが記述されているクラスまたはインタフェースを継承あるいは実装するクラスおよびインタフェースが参照されます。継承されたクラスは，フックの種類（**Hard**，**Soft**，**Off**）に基づいてフックされます。

- **Hard**：現在のクラスと，そのクラスを継承したクラスをフックします。正規表現が存在する場合は，このフック定義のクラスを継承したすべてのクラスと照合されます。インタフェースの継承は，クラスの継承と同様に扱われます。
- **Soft**：継承クラスでメソッドがオーバライドされる場合にのみ，現在のクラスと，そのクラスを継承したクラスをフックします。フックがインタフェースを記述している場合，クラスがこのインタフェースを実装していれば，そのメソッドはフックされます。メソッドは，そのクラスを直接継承するクラスに存在する場合もフックされます。ただし，フックがインタフェースを記述し，そのインタフェースを継承する別のインタフェースをクラスが実装した場合，そのクラスはフックされません。

注：正規表現は継承されませんが，実際のメソッドに変換されます。

- **Off**：フック定義に記述されているクラスと直接継承クラスのみフックされます。フックがインタフェースを記述している場合は，そのインタフェースを直接実装するクラスのみフックされます。

make_methods_public

フック定義と一致するメソッドが public メソッドに変換されます。これは，ユーザ定義フックに役立ちます。また，public 以外のメソッドからの呼び出しのサブツリーをロックする場合にも便利です。

これは記録時にのみ適用されます。再生時，メソッドは元のアクセス・フラグを使用します。public 以外のメソッドの場合は java.lang.VerifyError をスローします。

Lock

true に設定した場合は，サブツリーがロックされ，元のメソッドから発生したメソッドの呼び出しが行われないようになります。

false に設定した場合は，サブツリーのロックが解除され，現在のメソッドから発生したメソッドが記録されて（フックされている場合），コールバックが起動されます。

第 16 章

Java ー 相関

VuGen の相関機能を使うと，あるステートメントの結果を別のステートメントへの入力として使用して，Java Vuser 関数同士をリンクさせることができます。

本章の内容

Java スクリプトの相関について

Java コードを含む Vuser スクリプトには，多くの場合，動的データが含まれています。Java Vuser スクリプトを記録すると，動的データはスクリプトに記録されますが，再生時には再使用することができません。Vuser の実行中にエラーが発生した場合は，スクリプト内でエラーが発生した場所を調べます。多くの場合は，相関を行って，あるステートメントの結果を別のステートメントへの入力として使用できるようにすることで，問題を解決できます。

VuGen の Java レコーダは，生成されたスクリプト内のステートメントを自動的に相関させようとします。相関の対象は，Java オブジェクトに限ります。CORBA レコーダが記録中に Java のプリミティブ（byte，character，boolean，integer，float，double，short，long）を見つけると，引数の値が変数に割り当てられていない状態でスクリプトに示されます。VuGen では，オブジェクト，オブジェクトの配列，プリミティブの配列がすべて自動的に相関されます。Java の配列と文字列もオブジェクトとみなされます。

VuGen では，複数の相関レベル（標準，詳細，文字列）を使用します。［記録オプション］から相関を有効にしたり無効にしたりできます。前述の方法でスクリプトを処理できない場合は，シリアル化というもう 1 つの方法を使用してスクリプトを処理できます。詳細については，276 ページ「シリアル化メカニズムの使用」を参照してください。

標準的な相関

標準的な相関は，記録中にオブジェクトの配列，ベクトル，コンテナ構成要素を除く単純なオブジェクトを対象に実行される自動相関を指します。

記録されたアプリケーションが，オブジェクトを返すメソッドを起動すると，VuGen の相関メカニズムによってこれらのオブジェクトが記録されます。スクリプトを実行すると，生成されたオブジェクトと記録されたオブジェクトが比較されます。オブジェクトが一致すると，同じオブジェクトが使用されます。次の例は，2 つの CORBA オブジェクト my_bank と my_account を示しています。最初のオブジェクト my_bank が呼び出され，2 つ目のオブジェクト my_account が相関され，コードの最後の行でパラメータとして渡されます。

```
public class Actions {

    // public 関数：init
    public int init() throws Throwable {

    Bank my_bank = bankHelper.bind("bank", "shunra");
    Account my_account = accountHelper.bind("account","shunra");

    my_bank.remove_account(my_account);
    }
:
}
```

詳細相関

詳細相関または「**深い**」相関は，オブジェクトの配列や CORBA コンテナ構成要素などの複雑なオブジェクトを対象に，記録中に実行される自動相関を指します。

詳細相関メカニズムは，CORBA の構成要素（構造体，共用体，シーケンス，配列，ホルダ，「any」）をコンテナとして処理します。これによって，コンテナ，追加のオブジェクト，またはほかの各種コンテナの内部メンバを参照できます。オブジェクトは，呼び出されるかパラメータとして渡されると，コンテナの内部メンバとも比較されます。

次の例では，VuGen によって配列の要素を参照する詳細相関が実行されています。remove_account オブジェクトが my_account オブジェクトをパラメータとして受け取ります。相関メカニズムによって記録中に，返された配列 my_accounts が検索され，その 6 番目の要素をパラメータとして渡すことが検出されています。

```
public class Actions {

    // public 関数：init
    public int init() throws Throwable {

        my_banks[] = bankHelper.bind("banks", "shunra");
        my_accounts[] = accountHelper.bind("accounts","shunra");

        my_banks[2].remove_account(my_accounts[6]);
    }
:
}
```

次のコードは，詳細相関の別の例を示します。スクリプトによって address 型の引数を受け取った send_letter オブジェクトが呼び出されています。相関メカニズムによって，my_accounts 配列の 6 番目の要素の内部メンバ address が取得されます。

```
public class Actions {

    // public 関数：init
    public int init() throws Throwable {

        my_banks = bankHelper.bind("bank", "shunra");
    my_accounts = accountHelper.bind("account", "shunra");

        my_banks[2].send_letter(my_accounts[6].address);
    }
:
}
```

文字列の相関

文字列の相関は，記録された値を実際の文字列または変数として表すことを指します。文字列の相関を無効にすると（標準設定），記録された実際の文字列の値が，スクリプト内で明示されます。文字列の相関を有効にすると，各文字列の代わりに変数が作成され，スクリプトの以降の部分でこの変数を使用できるようになります。

次のコードでは，文字列の相関が有効になっており，get_id メソッドから返された値を，スクリプトのそれ以降の部分で使用できるように文字列（string）型変数に格納します。

```
public class Actions {

    // public 関数：init
    public int init() throws Throwable {

        my_bank = bankHelper.bind("bank", "shunra");
        my_account1 = accountHelper.bind("account1", "shunra");
        my_account2 = accountHelper.bind("account2", "shunra");

        string = my_account1.get_id();
        string2 = my_account2.get_id();
        my_bank.transfer_money(string, string2);
    }
:
}
```

相関メソッドの設定は，［記録オプション］の［**相関オプション**］タブで行います。

- ［**文字列を相関する**］：記録中にスクリプト内の文字列を相関させます。このオプションを無効にすると，実際に記録された値はスクリプトに引用符付きで含まれます。このオプションが無効になると，ほかのすべての相関オプションが無視されます（標準設定では無効）。
- ［**文字列配列を相関する**］：記録中に，文字列配列内の文字列を相関させます。このオプションを無効にすると，配列内の文字列は相関されず，実際の値がスクリプトに挿入されます（標準設定では有効）。
- ［**詳細相関**］：配列，CORBA コンテナの構成要素および配列といった複雑なオブジェクトを対象に相関させます。このタイプの相関は，「深い」相関としても知られています（標準設定では有効）。

- ［**相関レベル**］：複雑な相関のレベルの深さ（検索する内部コンテナの数）を指定します。
- ［**コレクションタイプを相関する**］：JDK 1.2 以上の Collection クラスに含まれるオブジェクトを相関させます（標準設定では無効）。

シリアル化メカニズムの使用

RMI および CORBA（一部）では，クライアントの AUT によって，java.io.serializable インタフェースに基づく Java オブジェクトの新しいインスタンスが作成されます。サーバの呼び出しに対して，このインスタンスがパラメータとして渡されます。次のコードでは，インスタンス p が作成され，パラメータとして渡されています。

```
// AUT コード：
java.awt.Point p = new java.awt.Point(3,7);
map.set_point(p);
:
```

前のどの呼び出しからもオブジェクトが返されなかったので，自動相関メカニズムはここでは適用されません。この場合，VuGen ではシリアル化メカニズムが実行され，パラメータとして渡されるオブジェクトが格納されます。この情報はユーザ・ディレクトリのバイナリ・データ・ファイルに保存されます。その他のパラメータは，新しいバイナリ・データ・ファイルとして保存され，順番に番号が付けられます。VuGen によって次のコードが生成されます。

```
public class Actions {

    // public 関数：init
    public int init() throws Throwable {
        java.awt.Point p = (java.awt.Point)lr.deserialize(0, false);
        map.set_point(p);
    }
:
}
```

lr.deserialize に渡された整数は，Vuser ディレクトリのバイナリ・データ・ファイルの番号を表します。

記録された値をパラメータ化するには，public メソッドの setLocation メソッドを使用します。詳細については，JDK の関数リファレンスを参照してください。次の例では，setLocation メソッドを使って，オブジェクト p の値を設定しています。

```
public class Actions {

    // public 関数：init
    public int init() throws Throwable {
        java.awt.Point p = (java.awt.Point)lr.deserialize(0, false);
        p.setLocation(2,9);
        map.set_point(p);
    }
:
:
}
```

インスタンスによっては，public メソッドの **setLocation** を適用できないものもあります。代わりに，クラスのアクセッサ・メソッドである get または set メソッドを使う API を使用できます。AUT のクラスに get および set メソッドがないか，プライベート・メソッドを使っている場合や，クラスの API に慣れていない場合は，VuGen に組み込まれているシリアル化メカニズムを使用できます。このメカニズムによって，オブジェクトを ASCII 表現に展開しスクリプトを手作業でパラメータ化できます。このメカニズムを有効にするには，［記録オプション］ダイアログ・ボックスで設定します。詳細については，『**第 1 巻ー VuGen の使用**』の第 18 章「Java 記録オプション」を参照してください。

VuGen によって，データがデシリアル化されます。つまり複雑なデータ構造を一連の文字列として表示する **lr.deserialize** メソッドを生成します。データ構造体をコンポーネントに分解すると，パラメータ化が簡単になります。
lr.deserialize メソッドは文字列と整数の 2 つの引数を受け取ります。文字列は，再生中に置換されるパラメータの値です。整数は，ロードするバイナリ・ファイルのインデックス番号です。

スクリプトのオブジェクトを展開しないように，［シリアル化オブジェクトの展開］チェック・ボックスをクリアすると，lr.deserialize メソッドに引数を渡すことによって，シリアル化メカニズムを制御できます。最初の引数は整数で，ロードするバイナリ・ファイルの数を示しています。2 つ目の整数はブール値です。

true	VuGen のシリアル化メカニズムを使用します。
false	標準の Java シリアル化メカニズムを使用します。

次のコードは，シリアル化メカニズムが有効になっている状態で生成されたスクリプトを示します。

```
public class Actions {

    // public 関数：init
    public int init() throws Throwable {
        _string = "java.awt.Point __CURRENT_OBJECT = {" +
            "int x = "#5#" +
            "int y = "#8#" +
        "}";
        java.awt.Point p = (java.awt.Point)lr.deserialize(_string,0);
        map.set_point(p);
    }
:
}
```

文字列値は，区切り文字の間に置かれます。区切り文字を変更するには，［記録オプション］の［**シリアル化オプション**］パネルで行います。区切り文字を使用するのは，再生時の文字列の解析処理を高速化するためです。

文字列を変更するときには，次のルールを守る必要があります。

- 行の順序は変更できません。パーサは，メンバ名ではなく，値を 1 つずつ読み取ります。
- 2 つの区切り文字の間にある値だけを変更できます。
- オブジェクト参照は変更できません。オブジェクト参照は，内部の一貫性を保つためだけに示されています。
- 「_NULL_」が値（Java の null 定数）として示されていることがあります。これは文字列型の値にのみ置換できます。

- オブジェクトはスクリプト内の任意の場所でシリアル化解除できます。たとえば，**init** メソッド内のすべてのオブジェクトをシリアル化解除し，**Action** メソッドの値を使用することができます。
- オブジェクトの内部的な一貫性を保ちます。たとえば，ベクトル・オブジェクトに要素数を示すメンバ（**element count**）があり，要素を追加した場合は，要素数を変更する必要があります。

  次のコードでは，ベクトルに，2 つの要素が含まれています。

```
public class Actions {

    // public 関数：init
    public int init() throws Throwable {
        _string = "java.util.Vector CURRENTOBJECT = {" +
            "int capacityIncrement = "#0#" +
            "int elementCount = #2#" +
            "java/lang/Object elementData[] = {" +
                "elementData[0] = #First Element#" +
                "elementData[1] = #Second Element#" +
                "elementData[2] = _NULL_" +
                  ....
                "elementData[9] = _NULL_" +

            "}" +
        "}";
        _vector = (java.util.Vector)lr.deserialize(_string,0);
        map.set_vector(_vector);
    }
:
}
```

次の例では，ベクトルの要素の 1 つを「_NULL_」から「Third element」に変更しています。新しい要素の追加に伴って，「elementCount」メンバを「3」に変更しています。

```
public class Actions {

    // public 関数：init
    public int init() throws Throwable {
        _string = "java.util.Vector CURRENTOBJECT = {" +
            "int capacityIncrement = "#0#" +
            "int elementCount = #3#" +
            "java/lang/Object elementData[] = {" +
                "elementData[0] = #First Element#" +
                "elementData[1] = #Second Element#" +
                "elementData[2] = #Third Element#" +
                 ....
                "elementData[9] = _NULL_" +
            "}" +
        "}";
        _vector = (java.util.Vector)lr.deserialize(_string,0);
        map.set_vector(_vector);
    }
:
}
```

オブジェクトを ASCII 表現に展開するシリアル化メカニズムは複雑なため，記録中に大きなオブジェクトを開くと，スクリプトの生成に時間がかかることがあります。この時間を短縮するために，シリアル化メカニズムのパフォーマンスを向上させるフラグを指定できます。

スクリプトに **lr.deserialize** を追加するときは，**action** メソッドではなく，**init** メソッドに追加することをお勧めします。VuGen によって文字列が一度だけシリアル化解除されればよいので，パフォーマンスが向上します。lr.deserialize が **action** メソッドにあると，VuGen では反復処理が行われるたびに文字列のシリアル化解除が行われることになります。

［記録オプション］の［**シリアル化オプション**］パネルでは，次のオプションを設定できます。

- ［シリアル化区切り文字］
- ［未展開のシリアル化オブジェクト］
- ［配列を展開する］
- ［配列エントリの制限］
- ［シリアル化オブジェクトを無視する］

記録オプションの詳細については，『**第 1 巻 ー VuGen の使用**』の第 18 章「Java 記録オプション」を参照してください。

第 17 章

Enterprise Java Beans（EJB）プロトコル

VuGen を使用して，アプリケーション・サーバで Enterprise Java Beans（EJB）オブジェクトをテストするためのスクリプトを作成できます。

本章の内容

EJB テストについて

VuGen には，Java アプリケーションをテストするためのスクリプトを作成する，いくつかのツールがあります。Vuser スクリプトを記録して生成するには，Java Record/Replay Vuser を使用します。スクリプトをプログラミングによって作成する場合には，ユーザ定義の Java Vuser タイプを使用します。

EJB テスト用 Vuser と標準 Java Vuser との違いは，記録やプログラミングを行わずに，VuGen が自動的に EJB の機能をテストまたはチューニングするスクリプトを作成する点です。スクリプトを生成する前に，アプリケーション・サーバに関する JNDI のプロパティなどの情報を指定します。VuGen の EJB Detector は，アプリケーション・サーバ内を検索し，どの EJB が使用可能かを判断します。テストまたはチューニングする EJB を選択すると，VuGen は各 EJB メソッドをエミュレートするスクリプトを生成します。

メソッドごとにトランザクションを作成して，各メソッドのパフォーマンスの測定と問題の特定ができるようにします。さらに，各メソッドは例外処理のために **try / catch** ブロックに入れられます。

EJB テスト・スクリプトを作成するには，EJB Detector がアプリケーション・サーバのホスト上にインストールされてアクティブになっていなければなりません。Detector については，この後の項で説明します。

VuGen には，スクリプトにメソッドを挿入するユーティリティも組み込まれています。このユーティリティを使用して，すべての利用可能なパッケージの表示，使用するメソッドの選択，およびそれらのスクリプトへの挿入ができます。詳細については，299 ページ「EJB Vuser スクリプトの実行」を参照してください。

EJB Detector での作業

EJB Detector は独立したエージェントで，EJB を検索するマシンごとにインストールする必要があります。このエージェントは，マシン上の EJB を検出します。EJB Detector をインストールする前に，マシンに有効な JDK 環境が設定されていることを確認します。

EJB Detector のインストール

EJB Detector は，アプリケーション・サーバ・マシンまたはクライアント・マシンにインストールして実行できます。クライアント・マシンで EJB Detector を実行するには，アプリケーション・サーバ・マシンにマウントされたドライブが必要です。

EJB Detector エージェントをインストールするには，次の手順を実行します。

1 アプリケーション・サーバ・マシンまたはクライアント・マシンに EJB Detector のホーム・ディレクトリを作成します（クライアント・マシンに作成する場合は，ファイル・システムを前述のようにマウントします）。

2 EJB Detector フォルダで圧縮ファイル ＜ **LoadRunner のインストール・フォルダ** ＞ **¥ejbcomponent¥ejbdetector.jar** を解凍します。

EJB Detector の実行

VuGen で EJB スクリプトの生成を開始する前に，EJB Detector を実行しておく必要があります。EJB Detector はアプリケーション・サーバまたはクライアント・マシンで実行できます（クライアント・マシンの場合は，EJB Detector のクライアント・マシンからアプリケーション・サーバへのマウントを行い，検索ルート・ディレクトリ内にマウント・ディレクトリを指定し，生成されたスクリプトをローカル・マシンではなくマウントされたマシンに接続するように変更します）。

EJB Detector は，コマンド・ラインまたはバッチ・ファイルから実行できます。

EJB Detector のコマンド・ラインから実行するには，次の手順を実行します。

1 コマンド・ラインで EJB Detector を実行する前に，CLASSPATH 環境変数に DETECTOR_HOME¥classes と DETECTOR_HOME¥classes¥xerces.jar を 追加します。

2 EJB1.0（Weblogic 4.x，WebSphere 3.x）で作業する場合は，テスト対象の EJB クラスを，次のベンダー EJB クラスとともに CLASSPATH に追加します。

WebLogic 4.x の場合：＜ WebLogic のディレクトリ＞ ¥lib¥weblogicaux.jar

WebSphere 3.x の場合：＜ WebSphere のディレクトリ＞ ¥lib¥ujc.jar

3 対象の EJB で使用しているクラス・ディレクトリまたは .jar ファイルがほかにもある場合，それらも CLASSPATH に追加します。

4 コマンド・ラインから EJB Detector を実行するには，次の文字列を使用します。

```
java EJBDetector [ 検索ルート・ディレクトリ ] [ リッスン・ポート ]
```

検索ルート・ディレクトリ	EJB を検索する 1 つ以上のディレクトリまたはファイル（セミコロンで区切って指定します）。次のガイドラインに従ってください。 **BEA WebLogic Servers 4.x および 5.x**：アプリケーション・サーバのルート・ディレクトリを指定します。 **BEA WebLogic Servers 6.x**：ドメイン・フォルダの完全パスを指定します。 **WebSphere Servers 3.x**：デプロイ済み EJB フォルダの完全パスを指定します。 **WebSphere Servers 4.0**：アプリケーション・サーバのルート・ディレクトリを指定します。 **Oracle OC4J**：アプリケーション・サーバのルート・ディレクトリを指定します。 **Sun J2EE Server**：**.ear** ファイルまたは複数の **.ear** ファイルがあるディレクトリへの完全パスを指定します。 指定しない場合は，クラスパスが検索されます。
リッスン・ポート	EJB Detector のリッスン・ポートです。標準設定のポートは 2001 です。ポート番号を変更したら，［**EJB スクリプトの生成**］ダイアログ・ボックスの［**ホスト**］の［**名前**］ボックスでも指定しなおす必要があります。 たとえば，ホストが「metal」で，標準設定のポートを使用している場合，「**metal**」と指定します。別のポート，たとえばポート 2002 を使用している場合は，「**metal:2002**」と指定します。

EJB Detector のバッチ・ファイルから実行するには，次の手順を実行します。

バッチ・ファイル **EJB_Detector.cmd** を使用して，EJB Detector を起動できます。このファイルは，圧縮ファイル **ejbdetector.jar** を解凍すると，インストールされている EJB Detector のルート・ディレクトリに置かれます。

1 EJB Detector の ルート・ディレクトリで env.cmd を開き，環境に応じて次の変数を変更します。

JAVA_HOME	インストールされている JDK のルート・ディレクトリ。
DETECTOR_INS_DIR	インストールされた Detector のルート・ディレクトリ。
APP_SERVER_DRIVE	アプリケーション・サーバのインストールをホストするドライブ。
APP_SERVER_ROOT	次のガイドラインに従ってください。 **BEA WebLogic Servers 4.x および 5.x**：アプリケーション・サーバのルート・ディレクトリを指定します。 **BEA WebLogic Servers 6.x**：ドメイン・フォルダの完全パスを指定します。 **WebSphere Servers 3.x**：デプロイ済み EJB フォルダの完全パスを指定します。 **WebSphere Servers 4.0**：アプリケーション・サーバのルート・ディレクトリを指定します。 **Oracle OC4J**：アプリケーション・サーバのルート・ディレクトリを指定します。 **Sun J2EE Server**：**.ear** ファイルまたは複数の **.ear** ファイルがあるディレクトリへの完全パスを指定します。
EJB_DIR_LIST（任意）	デプロイ可能な ear/.jar ファイル，および任意の追加クラス・ディレクトリまたは .jar ファイルを含んでいるか，テスト対象の EJB で使用されている，セミコロン（;）で区切られたディレクトリとファイルのリスト。

2 env.cmd を保存します。

3 EJB1.0（Weblogic 4.x，WebSphere 3.x）で作業する場合は，テスト対象の EJB のクラスとともに，次のベンダー EJB クラスを env ファイル の CLASSPATH に追加します。

- WebLogic 4.x の場合：< WebLogic のディレクトリ > ¥lib¥weblogicaux.jar
- WebSphere 3.x の場合：< WebSphere のディレクトリ > ¥lib¥ujc.jar

4 バッチ・ファイル EJB_Detector.cmd または EJB_Detector.sh（Unix プラットフォームの場合）を実行して，EJB を含むデプロイ可能なアプリケーションに関する情報を収集します。たとえば，次のように指定します。

```
C:¥>EJB_Detector [listen_port]
```

「listen_port」は，EJB Detector が受信する要求を読み取るポート番号を指定するための任意の引数です（標準設定では「2001」）。

EJB Detector の出力およびログ・ファイル

EJB Detector の出力を調べて，アクティブな EJB をすべて検出したかどうかを確認できます。出力ログには，EJB で検査されたパスが表示されます。検索が終わると，見つかった EJB の名前と場所のリストが表示されます。次に例を示します。

```
Checking EJB Entry: f:/weblogic/myserver/ejb_basic_beanManaged.jar...
Checking EJB Entry: f:/weblogic/myserver/ejb_basic_statefulSession.jar...
Checking EJB Entry: f:/weblogic/myserver/ejb_basic_statelessSession.jar...
------------------------ Found 3 EJBs --------------------------
** PATH: f:/weblogic/myserver/ejb_basic_beanManaged.jar
- BEAN: examples.ejb.basic.beanManaged.AccountBean
** PATH: f:/weblogic/myserver/ejb_basic_statefulSession.jar
- BEAN: examples.ejb.basic.statefulSession.TraderBean
** PATH: f:/weblogic/myserver/ejb_basic_statelessSession.jar
- BEAN: examples.ejb.basic.statelessSession.TraderBean
```

EJB が検出されなかった場合（「Found 0 EJBs」と表示されます），EJB jar ファイルが「Checking EJB Entry:...」行に表示されているかどうかを確認してください。リストに表示されていない場合は，「**検索ルート・ディレクトリ**」のパスが正しいかどうか確認します。検査が行われているのにもかかわらず EJB が検出されない場合は，EJB jar ファイルがデプロイ可能か（アプリケーション・サーバにデプロイできるか）確認します。デプロイ可能な jar ファイルには，Home Interface, Remote Interface, Bean の実装，Deployment Descriptor ファイル（xml ファイルまたは .ser ファイル）およびその他のベンダー固有のファイルが含まれています。

それでも問題が生じる場合は，DETECTOR_HOME¥classes ディレクトリにある「**detector.properties**」ファイルにデバッグ・プロパティを設定し，さらに詳しいデバッグ情報を取得します。

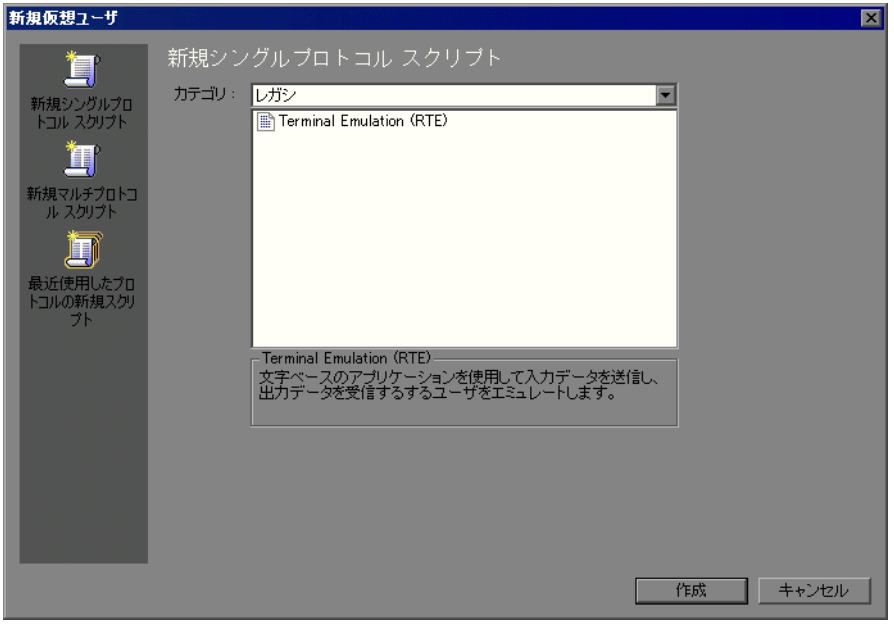
EJB が検出されると，HTTP サーバは初期化されて，VuGen の EJB テスト Vuser からの要求を待機します。この通信過程に問題がある場合は，DETECTOR_HOME¥classes ディレクトリにある **webserver.properties** ファイルで webserver.enableLog プロパティを有効にします。

これにより，**webserver.log** ファイルに詳しいデバッグ情報や，その他の潜在的に重要なエラー・メッセージが出力されるようになります。

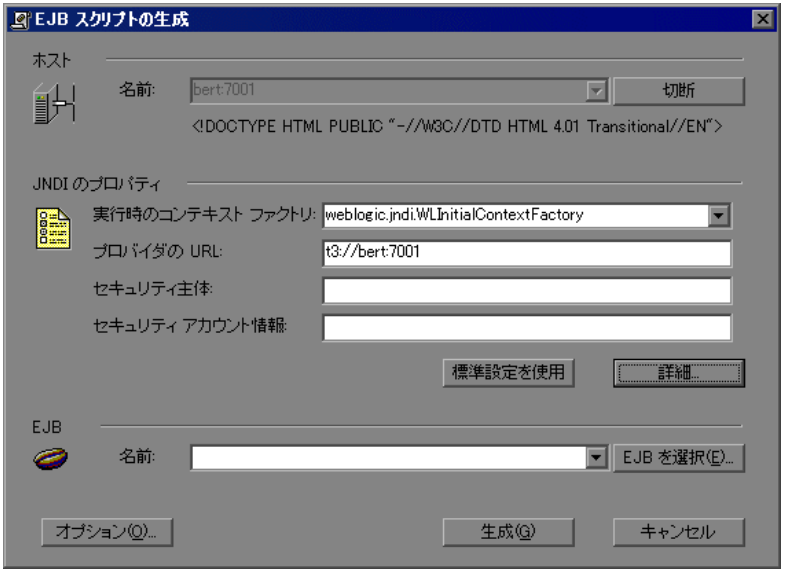
EJB テストの Vuser の作成

EJB Vuser スクリプトを作成するには，次の手順を実行します。

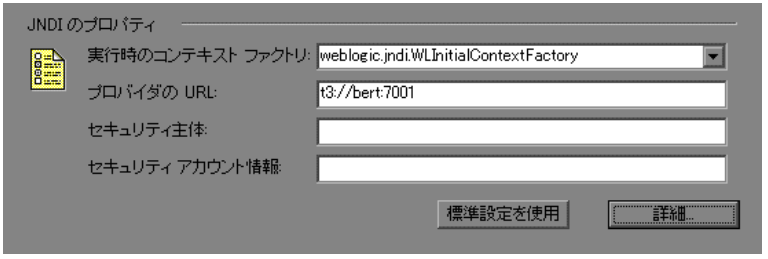
1 ［**ファイル**］＞［**新規作成**］を選択するか，［**新規作成**］ボタンをクリックします。［新規仮想ユーザ］ダイアログ・ボックスが表示されます。



2 ［**Enterprise Java Beans**］カテゴリから［**Enterprise Java Beans（EJB）**］を選択し，［**OK**］をクリックします。VuGen が空の Java Vuser スクリプトを開き，［EJB スクリプトの生成］ダイアログ・ボックスが表示されます。



3 VuGen EJB Detector がインストールされているマシンを指定します。接続するためには Detector が稼動していなければなりません。［**接続**］をクリックします。［**JNDI のプロパティ**］セクションが有効になります。



4 EJB Detector は，標準設定の JNDI プロパティを自動的に検出します。これらのプロパティは，編集ボックスで手作業で変更することもできます。変更可能なプロパティは，［**実行時のコンテキスト ファクトリ**］および［**プロバイダの URL**］の文字列です。

アプリケーション・サーバが認証を必要とする場合は，［**セキュリティ主体**］ボックスにユーザ名，［**セキュリティ アカウント情報**］にパスワードを入力します。

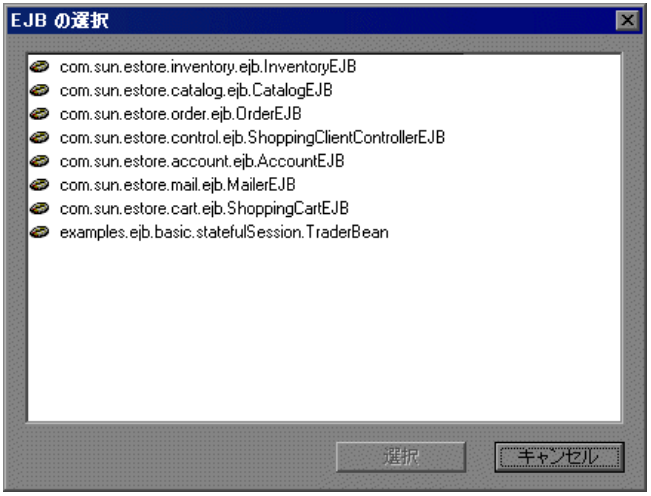
次に JNDI の必須プロパティの 2 つの標準設定値を示します。

タイプ	実行時のコンテキスト・ファクトリ	プロバイダの URL
WebLogic	weblogic.jndi.WLInitialContextFactory	t3://<appserver_host>:7001
WebSphere 3.x	com.ibm.ejs.ns.jndi.CNInitialContextFactory	iiop://<appserver_host>:900
WebSphere 4.x	com.ibm.websphere.naming.WsnInitialContextFactory	iiop://<appserver_host>:900
Sun J2EE	com.sun.enterprise.naming.SerialInitContextFactory	なし
Oracle	com.evermind.server.AppplicationClientInitialContextFactory	ormi:// ＜ appserver_host ＞ / ＜ application_name ＞（＜ oc4j ＞ /config/server.xml 形式の EJB アプリケーション名）

5 JNDI の詳細プロパティを設定するには，［**詳細**］をクリックして［JNDI の詳細プロパティ］ダイアログ・ボックスを開きます。

必要に応じて，[**オブジェクト ファクトリ**]，[**状態ファクトリ**]，[**URL パッケージの前置記号**]，[**セキュリティ プロトコル**]，[**セキュリティ認証**] プロパティを指定します。[**OK**] をクリックします。

6 ダイアログ・ボックス内の [**EJB**] セクションで [**EJB を選択**] をクリックし，テストを作成する EJB を選択します。ダイアログ・ボックスが開き，現在アプリケーション・サーバからアクセス可能なすべての EJB のリストが表示されます。



7 テスト対象の EJB を強調表示し，[**選択**] をクリックします。

8 [EJB スクリプトの生成] ダイアログ・ボックスで，[**生成**] をクリックします。VuGen は，Java Vuser 関数を含むスクリプトを作成します。スクリプトには，アプリケーション・サーバに接続し，EJB のメソッドを実行するためのコードが含まれます。

9 スクリプトを保存します。

既存のスクリプト内には追加 EJB のテストコードを生成できません。別の EJB を作成するには，新規のスクリプトを開き，前述のステップ 2 ～ 9 を繰り返します。

EJB Vuser スクリプトについて

VuGen は，Vuser スクリプトの作成時に指定された JNDI（Java Naming and Directory Interface）のプロパティに基づいて，EJB をテストするスクリプトを生成します。JNDI は，Java プログラムを DNS および LDAP などのネーム・サービスやディレクトリ・サービスに接続するときに使用される Sun のプログラミング・インタフェースです。

各 EJB Vuser スクリプトは，次の 3 つの主要部分からなります。

- JNDI による EJB Home の検索
- インスタンスの作成
- EJB メソッドの起動

JNDI による EJB Home の検索

スクリプトの最初のセクションには，JNDI のプロパティを取得するためのコードが含まれています。このコードは指定されたコンテキスト・ファクトリおよびプロバイダの URL を使用して，アプリケーション・サーバに接続し，指定された EJB を検索し，EJB Home を見つけます。

次の例では，JNDI Context Factory は weblogic.jndi.WLInitialContextFactory，プロバイダの URL は t3://dod:7001, 選択された EJB の JNDI 名は carmel.CarmelHome です。

```
public class Actions
{

    public int init() {
        CarmelHome _carmelhome = null;
        try {
            // JNDI Initial Context を取得する
            java.util.Properties p = new java.util.Properties();
            p.put(javax.naming.Context.INITIAL_CONTEXT_FACTORY,
"weblogic.jndi.WLInitialContextFactory");
            p.put(javax.naming.Context.PROVIDER_URL, "t3://dod:7001");
            javax.naming.InitialContext _context = new javax.naming.InitialContext(p);

            // JNDI コンテキストで Home Interface を検索し，絞り込む
            Object homeobj = _context.lookup("carmel.CarmelHome");
            _carmelhome =
(CarmelHome)javax.rmi.PortableRemoteObject.narrow(homeobj, CarmelHome.class);

        } catch (javax.naming.NamingException e) {
            e.printStackTrace();
        }
```

注：スクリプトがアプリケーション・サーバではなくクライアントで実行されている EJB Detector で生成されている場合は，プロバイダの URL を手作業で変更する必要があります。たとえば，次の行では，プロバイダは EJB Detector のホスト名として dod を指定しています。

p.put(javax.naming.Context.PROVIDER_URL, "t3://dod:7001")

記録されたホスト名をアプリケーション・サーバ名で置き換えます。

次に例を示します。

p.put(javax.naming.Context.PROVIDER_URL, "t3://bealogic:7001")

［EJB スクリプトの生成］ダイアログ・ボックスの［**JDNI のプロパティ**］セクションで，記録を開始する前にプロバイダの URL を指定しておくと，手作業で変更する必要がなくなります。

インスタンスの作成

EJB メソッドを実行する前に，スクリプトは，EJB の Bean インスタンスを作成します。インスタンスの作成は，トランザクションとしてマークされるので，スクリプトの実行後に分析できます。さらに，インスタンスの作成プロセスは，例外処理のために **try / catch** ブロックに入れられます。

セッション Beans では，EJB Home「作成」メソッドを使用して新しい EJB インスタンスを作成します。

次の例では，スクリプトは，Carmel EJB のインスタンスを作成します。

```
// Bean インスタンスの作成
Carmel _carmel = null;
try {
    lr.start_transaction("create");
    _carmel = _carmelhome.create();
    lr.end_transaction("create", lr.AUTO);
} catch (Throwable t) {
    lr.end_transaction("create", lr.FAIL);
    t.printStackTrace();
}
```

エンティティ Beans では，findByPrimaryKey メソッドを使用して既存のデータベースで EJB インスタンスを検索します。見つからなかった場合は，create メソッドを使用してその中に作成します。

次の例では，スクリプトは，アカウント EJB のインスタンスを検索し，見つからない場合には作成します。

```
// Bean インスタンスの検索
try {
    com.ibm.ejs.doc.account.AccountKey _accountkey = new
com.ibm.ejs.doc.account.AccountKey();
    _accountkey.accountId = (long)0;

    lr.start_transaction("findByPrimaryKey");
    _account = _accounthome.findByPrimaryKey(_accountkey);
    lr.end_transaction("findByPrimaryKey", lr.AUTO);
} catch (Throwable thr) {

    lr.end_transaction("findByPrimaryKey", lr.FAIL);
    lr.message("Couldn't locate the EJB object using a primary key. Attempting to
manually create the object... ["+thr+"]");

    // Bean インスタンスの作成
    try {
        lr.start_transaction("create");
        _account = _accounthome.create((com.ibm.ejs.doc.account.AccountKey)null);
        lr.end_transaction("create", lr.AUTO);
    } catch (Throwable t) {
        lr.end_transaction("create", lr.FAIL);
        t.printStackTrace();
    }

}
```

エンティティ Bean によって提供されるほかの find... メソッドを使用して EJB インスタンスを検出することもできます。次に例を示します。

```
// データベース内の「John」を示す,
// すべての Email EJB インスタンスのリストの取得
Enumeration enum = home.findByName( "John" );
    while ( enum.hasMoreElements() ) {
        Email addr = (Email)enum.nextElement();
        ...
}
```

EJB メソッドの起動

スクリプトの最後の部分には，実際の EJB メソッドが含まれています。各メソッドはトランザクションとしてマークされるので，スクリプト実行後にメソッドを分析できます。さらに，各メソッドは例外処理のために try / catch ブロックに入れられます。例外があった場合には，トランザクションは「failed」とマークされ，スクリプトは次のメソッドから続行されます。VuGen は EJB メソッドごとに個別のブロックを作成します。

```
// ------- メソッド ------------

        int _int1 = 0;
        try {
            lr.start_transaction("buyTomatoes");
            _int1 = _carmel.buyTomatoes((int)0);
            //lr.value_check(_int1, 0, lr.EQUALS);
            lr.end_transaction("buyTomatoes", lr.AUTO);
        } catch (Throwable t) {
            lr.end_transaction("buyTomatoes", lr.FAIL);
            t.printStackTrace();
        }
```

VuGen はそれらのメソッドの標準値を挿入します。たとえば，整数の場合は 0，文字列の場合は空の文字列（""），複雑な Java オブジェクトの場合は NULL となります。必要に応じて，生成されたスクリプト内の標準値を変更します。

```
            _int1 = _carmel.buyTomatoes((int)0);
```

次の例では，パラメータ化を使用した，複雑なタイプの標準値の変更方法を示しています。

```
Detail details = new Details( < city > , < street > , < zip > , < phone > );
JobProfile job = new  JobProfile( < department > , < position > , < job_type > );
Employee employee=new Employee( < first > , < last > , details, job, < salary > );
_int1 = _empbook.addEmployee((Employee)employee);
```

簡単な値や文字列を返すメソッドの場合には，VuGen はコメントアウトされたメソッド lr.value_check を挿入します。このメソッドを使用して，EJB メソッドの期待値を指定できます。この検証メソッドを使用するときには，コメントの記号（//）を削除し，期待値を指定します。たとえば，carmel.buyTomatoes メソッドは整数を返します。

```
_int1 = _carmel.buyTomatoes((int)0);
//lr.value_check(_int1, 0, lr.EQUALS);
```

メソッドが 500 という値を返すようにするには，コードを次のように変更します。

```
_int1 = _carmel.buyTomatoes((int)0);
lr.value_check(_int1, 500, lr.EQUALS);
```

メソッドが特定の値を返さなかったかどうかを検査する場合は，コードを次のように変更します。

```
_int1 = _carmel.buyTomatoes((int)0);
lr.value_check(_int1, 10, lr.NOT_EQUALS);
```

期待値が検出されない場合は，例外処理が行われ，その情報は出力ウィンドウのログに記録されます。

```
System.err: java.lang.Exception: lr.value_check failed.[Expected:500 Actual:5000]
```

EJB Vuser スクリプトは，標準 Java 規約をすべてサポートしています。たとえば，テキストの前に 2 つのスラッシュ「//」を付けることによって，コメントを挿入できます。

Java Vuser スクリプトは，スケーラブルなマルチ・スレッド・アプリケーションとして稼動します。スクリプトにユーザ定義のクラスを含める場合は，コードをスレッドセーフにする必要があります。スレッドセーフでないコードは，結果が不正確になることがあります。スレッドセーフでないコードの場合は，Java Vuser をプロセスとして実行します。つまり，プロセスごとに個別の Java 仮想マシンを作成します。その結果，スクリプトの拡張性は低くなります。

EJB Vuser スクリプトの実行

EJB テストのスクリプトを生成した後，必要な変更を行えば，スクリプト実行の準備が完了します。EJB スクリプトでは，機能テストと負荷テストの 2 種類のテストを実行できます。機能テストでは，EJB が実際の環境下で正しく機能することを検証します。負荷テストでは，一度に多数のユーザを実行したときの EJB のパフォーマンスを評価します。

機能テストを行うには，次の手順を実行します。

1 自動的に生成された標準値を変更します。

2 **lr.value_check** メソッドを使用して値の検査を挿入します。これらのメソッドは，スクリプト内にコメントとして生成されます。詳細については，297 ページ「EJB メソッドの起動」を参照してください。

3 追加のメソッドを挿入し，標準値を変更します。詳細については，第 14 章「Java プロトコル ― 記録」の Java 関数の挿入に関する項を参照してください。

4 スクリプトの一般的な実行環境を設定します。詳細については，『**第 1 巻 ― VuGen の使用**』の「実行環境の設定」を参照してください。

5 Java VM の実行環境を設定します （追加のクラスパス，VM パラメータをすべて指定します)。必ずアプリケーション・サーバ EJB クラスを入れます。テストされている実際の EJB クラスは Vuser ディレクトリに保存され，再生中に自動的に取り出されます。追加のクラスパスの指定，JavaVM 実行環境の設定の詳細については，『**第 1 巻 ― VuGen の使用**』の第 24 章「Java および EJB の実行環境の設定」を参照してください。

6 **Websphere 3.x** ユーザの場合：

IBM JDK 1.2 以上を使用した場合：

- クラスパスに< WS > ¥lib¥ujc.jar を追加します。

Sun JDK 1.2.x を使用する場合：

- ファイル< JDK > ¥jre¥lib¥ext¥iiimp.jar を削除します。
- 次のファイルを, < WS > ¥jdk¥jre¥lib¥ext フォルダから< JDK > ¥jre¥lib¥ext フォルダにコピーします。
- ujc.jar を＜WS＞¥lib フォルダから＜JDK＞¥jre¥lib¥ext フォルダにコピーします。

➤ ファイル＜WS＞¥jdk¥jre¥bin¥ioser12.dll を＜JDK＞¥jre¥bin フォルダにコピーします。

ここで＜ WS ＞は WebSphere インストールの ホーム・ディレクトリ，＜ JDK ＞は JDK インストールのホーム・ディレクトリです。

このオプションを無効にするには，［**-Xbootclasspath VM パラメータを使用する**］チェック・ボックスをクリアします。

7 **WebSphere 4.0** ユーザの場合：

マシンに IBM JDK1.3 の Java 環境が正しく設定されていることを確認します。［実行環境設定］ダイアログ・ボックスを開いて，［Java VM］ノードを選択します。次のエントリを［Additional Classpath］に追加します。

```
＜ WS ＞ ¥lib¥webshpere.jar;
＜ WS ＞ ¥lib¥j2ee.jar;
```

＜ WS ＞は，WebSphere インストールのホーム・ディレクトリです。

このオプションを無効にするには，［**-Xbootclasspath VM パラメータを使用する**］チェック・ボックスをクリアします。

注：アプリケーション・サーバが UNIX マシンにインストールされている場合，または WebSphere 3.0.x を使用している場合は，必要なファイルを取得するために，クライアントに IBM JDK 1.2.x をインストールします。

---

8 **Oracle OC4J** ユーザの場合：

マシンに JDK1.2 またはそれ以上（JDK1.3 推奨）の Java 環境が正しく設定されていることを確認します。［実行環境設定］ダイアログ・ボックスを開いて，［Java VM］ノードを選択します。次のエントリを［Additional Classpath］に追加します。

```
＜ OC4J ＞ ¥orion.jar; ＜ OC4J ＞ ¥ejb.jar; ＜ OC4J ＞ ¥jndi.jar; ; ＜ OC4J ＞ ¥xalan.jar;
＜ OC4J ＞ ¥crimson.jar
```

＜ OC4J ＞は，アプリケーション・サーバのインストールのホーム・ディレクトリです。

9 Sun J2EE ユーザの場合：

マシンに JDK1.2 またはそれ以上の Java 環境が正しく設定されていることを確認します。［実行環境設定］ダイアログ・ボックスを開いて，［Java VM］ノードを選択します。次のエントリを［Additional Classpath］に追加します。

```
< J2EE > ¥j2ee.jar; < AppClientJar >
```

< J2EE >にはアプリケーション・サーバをインストールするホーム・フォルダを指定します。< AppClientJar >は，デプロイメント工程の間に sdk ツールによって自動的に生成されるアプリケーション・クライアントの jar ファイルへのパスを指定します。

10 WebLogic 4.x ー 5.x ユーザの場合：

マシンに Java 環境（パスおよびクラスパス）が正しく設定されていることを確認します。［実行環境設定］ダイアログ・ボックスを開いて，［Java VM］ノードを選択します。次の 2 つのエントリを［Additional Classpath］セクションに追加します。

```
< WL > ¥classes; < WL > ¥lib¥weblogicaux.jar
```

< WL > は，WebLogic インストールのホーム・ディレクトリです。

11 WebLogic 6.x および **7.0** ユーザの場合：

マシンに Java 環境（パスおよびクラスパス）が正しく設定されていることを確認します。WebLogic 6.1 を使用する場合は，JDK 1.3 をインストールする必要があります。［実行環境設定］ダイアログ・ボックスを開いて，［Java VM］ノードを選択します。次のエントリを［Additional Classpath］に追加します。

```
< WL > /lib/weblogic.jar;   // Weblogic 6.x
< WL > /server/lib/weblogic.jar    // Weblogic 7.x
```

< WL > は，WebLogic インストールのホーム・ディレクトリです。

このオプションを無効にするには，［**-Xbootclasspath VM パラメータを使用する**］チェック・ボックスをクリアします。

12 スクリプトを実行します。［**実行**］ボタンをクリックするか，［**仮想ユーザ**］＞［**実行**］を選択します。［実行ログ］ノードを開き，実行時のエラーを表示します。実行ログは，スクリプトのフォルダ内の **mdrv.log** ファイルに保存されます。Java コンパイラ（Sun の javac）によってスクリプトに誤りがないか調べられた後，コンパイルが実行されます。

スクリプトを作成した後，スクリプトを自分の環境（LoadRunner シナリオ，Performance Center 負荷テスト，または Business Process Monitor プロファイル）に組み込みます。詳細については，**HP LoadRunner Controller**，**HP Performance Center**，または **HP Business Availibility Center** のドキュメントを参照してください。

第 18 章

Citrix プロトコル

VuGen では Citrix ICA プロトコルを使ってサーバとやり取りを行う Citrix クライアントのアクションを記録できます。記録の結果として作成されるスクリプトを，「Citrix Vuser スクリプト」と呼びます。

オプションの **Citrix Agent** は，組み込み同期化を提供する直感的なスクリプトの作成を支援します。詳細については，第 19 章「Citrix － Citrix Presentation Server エージェント」を参照してください。スクリプトの作成に役立つヒントについては，323 ページ「Citrix Vuser スクリプトの再生とトラブルシューティングのヒント」を参照してください。

本章の内容

Citrix Vuser スクリプトの作成について

Citrix Vuser スクリプトは，Citrix クライアントとサーバ間の Citrix ICA プロトコルの通信をエミュレートします。VuGen は，通信している間のすべての活動を記録し，Vuser スクリプトを作成します。

リモート・サーバに対してアクションを実行すると，VuGen によってこれらのアクションを表す関数が生成されます。各関数の先頭には，**ctrx** というプレフィックスが付きます。これらの関数は，マウスやキーボードのアナログ動作をエミュレートします。また，**ctrx** 関数では，特定のウィンドウが開くまで待機することで，アクションの再生を同期させることもできます。

VuGen では Citrix Nfuse セッションの記録も可能です。Citrix NFuse を使用する場合，クライアントはインストールされますが，インタフェースはクライアントのインタフェースではなくブラウザとなります。Nfuse セッションを記録するには，Citrix と Web の Vuser 用のマルチ・プロトコル・スクリプトの記録を実行する必要があります（『**第 1 巻 ー VuGen の使用**』の「VuGen を使った記録」を参照）。マルチ・プロトコル・モードでは，VuGen は記録中に Citrix と Web プロトコルの両方から関数を生成します。

次に示す例では，**ctrx_mouse_click** によってマウスの左クリックをシミュレートしています。

```
ctrx_mouse_click(44, 318, LEFT_BUTTON, 0, CTRX_LAST);
```

構文およびパラメータの詳細については，『**Online Function Reference**』（英語版）（［**ヘルプ**］＞［**関数リファレンス**］）を参照してください。

記録されたスクリプトは，VuGen のメイン・ウィンドウで表示および編集ができます。VuGen では，セッション中に記録された API 呼び出しが表示されるので，アクションを追うことができます。

Citrix Vuser スクリプトの開発の概要

本項では，VuGen を使って Citrix ICA Vuser スクリプトを開発する工程の概要を説明します。このほか，323 ページ「Citrix Vuser スクリプトの再生とトラブルシューティングのヒント」も参照してください。

Citrix ICA スクリプトを開発するには，次の手順を実行します。

1 クライアントとサーバの設定が正しいことを確認します。

これらの設定の詳細については，307 ページ「クライアントとサーバのセットアップ」を参照してください。

2 VuGen を使ってアクションを記録します。

VuGen を起動して，新しい Vuser スクリプトを作成します。記録中にビットマップとテキストの同期化を挿入します。詳細については，313 ページ「再生の同期化」を参照してください。

記録全般については，『**第 1 巻 － VuGen の使用**』の「VuGen を使った記録」を参照してください。

3 Vuser スクリプトを拡張します。

トランザクション，ランデブー・ポイント，および制御フロー構造をスクリプトに挿入することによって，Vuser スクリプトを拡張します。

詳細については，『**第 1 巻 － VuGen の使用**』の「Vuser スクリプトの拡張」を参照してください。

4 パラメータを定義します（任意）。

Vuser スクリプトに記録された固定値に対してパラメータを定義します。固定値をパラメータで置き換えることにより，同じビジネス・プロセスを，異なる値で何度も繰り返すことができます。

詳細については，『**第 1 巻 － VuGen の使用**』の「パラメータの作成」を参照してください。

5 Citrix の表示オプションを設定します。

Citrix Vuser を再生する際の表示オプションを設定します。これらのオプションを設定すると，再生中に Citrix クライアントを表示したり，エラー発生時のスナップショットを開いたりできます。詳細については，311 ページ「Citrix 表示の設定」を参照してください。

6 実行環境を設定します。

実行環境を設定することによって，スクリプト実行中の Vuser の動作を制御します。この設定には，ペースの設定，ログ，思考遅延時間，接続情報が含まれます。

Citrix 固有の実行環境の設定方法の詳細については，『**第 1 巻 － VuGen の使用**』の第 24 章「選択したプロトコルの実行環境の設定」を参照してください。一般的な実行環境の設定の詳細については，『**第 1 巻 － VuGen の使用**』の「実行環境の設定」を参照してください。

7 VuGen で Vuser スクリプトを保存して実行します。

VuGen で生成した Vuser スクリプトを保存して実行し，正しく動作することを確認します。記録中，VuGen によって一連の設定ファイル，データ・ファイル，ソース・コード・ファイルが生成されます。これらのファイルには，Vuser の実行時の情報および設定情報が含まれます。VuGen は，これらのファイルをスクリプトと一緒に保存します。

Vuser スクリプトをスタンドアロン・テストとして実行する方法の詳細については，323 ページ「Citrix Vuser スクリプトの再生とトラブルシューティングのヒント」および『**第 1 巻 － VuGen の使用**』の「スタンドアロン・モードでの Vuser スクリプトの実行」を参照してください。

スクリプトを作成した後，スクリプトを自分の環境（LoadRunner シナリオ，Performance Center 負荷テスト，または Business Process Monitor プロファイル）に組み込みます。詳細については，**HP LoadRunner Controller**，**HP Performance Center**，または **HP Business Availibility Center** のドキュメントを参照してください。

クライアントとサーバのセットアップ

スクリプトを作成する前に，サポートされている Cirix クライアントがマシンにインストールされており，サーバが正しく設定されていることを確認します。本項では，次の項目について説明します。

- クライアントのバージョン
- サーバの設定

クライアントのバージョン

スクリプトを実行するには，各 Load Generator マシンに Citrix クライアントがインストールされている必要があります。クライアントがインストールされていない場合は，Citrix の Web サイト（www.citrix.com）の **download** セクションからダウンロードできます。

VuGen は，バージョン 8.00，バージョン 6.30.1060 以前，および Citirx Web クライアントを除くすべての Citrix クライアントをサポートします。

サーバの設定

VuGen で記録を行うには，次の項目について Citrix サーバを設定する必要があります。

MetaFrame サーバのインストール

MetaFrame サーバ（3，4，または 4.5）がインストールされていることを確認します。サーバのバージョンを調べるには，サーバのコンソール・ツールバーにある［**Citrix コネクション構成ツール**］を選択して，［**ヘルプ**］＞［**バージョン情報**］を選択します。

セッションを終了するように MetaFrame サーバを設定

Citrix サーバがセッションを完全に終了するように設定します。Citrix クライアントが接続を終了するとき，標準設定では，次回そのクライアントが新しい接続を開いたときのためにセッションを保存するようにサーバが設定されています。したがって，同じクライアントで新たに接続すると，前回接続を解除したときと同じ作業領域が表示されます。新しいテスト実行のたびに初期状態の作業領域を使用できるようにすることが望まれます。

テストのたびに初期状態の作業領域を確保するには，以前のセッションを保存しないよう Citrix サーバを設定する必要があります。その代わりに，クライアントがタイムアウトまたは接続断になるたびにクライアントから接続を解除することによって，接続をリセットするようにします。

サーバを設定するには，次の手順を実行します。

1 ［Citrix コネクション設定］ダイアログ・ボックスを開きます。［**プログラム**］＞［**Citrix**］＞［**管理ツール**］＞［**CCitrix コネクション設定ツール**］を選択します。

2 ica-tcp 接続名を選択し，［**コネクション**］＞［**編集**］を選択します。あるいは，接続をダブルクリックします。［コネクションの編集］ダイアログ・ボックスが開きます。

3 ［**詳細設定**］ボタンをクリックします。［コネクションの詳細設定］ダイアログ・ボックスが開きます。

4 このダイアログの一番下のセクションにある［**接続が切断またはタイムアウトしたときの処理**］リスト・ボックス横の［**(アカウントの設定を使用)**］チェック・ボックスをクリアします。リスト・ボックスのエントリを「**リセットする**」に変更します。

5 ［**OK**］をクリックします。

記録に関するヒント

スクリプトを記録するときは，効果的なスクリプトを作成できるように必ず次のガイドラインに従ってください。

シングル・プロトコル・スクリプトとマルチ・プロトコル・スクリプト

スクリプトを新規に作成するときは，シングル・プロトコルかマルチ・プロトコルのスクリプトを作成できます。簡単な Citrix ICA セッションを記録する場合は，シングル・プロトコル・スクリプトを使用します。ただし，Nfuse Web アクセス・セッションの記録時には，Citrix ICA と Web（HTML/HTTP）を対象とするマルチ・プロトコル・スクリプトを作成する必要があります。これによって，両方のプロトコルを記録できるようになります。詳細については，『**第 1 巻 ー VuGen の使用**』の「VuGen を使った記録」を参照してください。

適切なセクションに記録する

接続処理は「**vuser_init**」セクションに，終了処理は「**vuser_end**」セクションに記録します。これにより，接続もしくは接続解除時に反復が実行されないようになります。セクションへの記録の詳細については，『**第 1 巻 ー VuGen の使用**』の「VuGen を使った記録」を参照してください。

初期状態のセッションを実行する

セッションを記録するときは必ず，接続操作から始まって，クリーンアップで終わる完全なビジネス・プロセスを実行するようにします。ビジネス・プロセス全体を始めから再実行できるところでセッションを終了するようにします。クライアントやアプリケーションのウィンドウを開いたままにしないようにします。

明示的にクリックする

サブメニューを開くときは，メニューの自動展開に頼らずに，各オプションを明示的にクリックします。たとえば，［**スタート**］＞［**プログラム**］＞［**Microsoft Word**］の場合は，「**プログラム**」という語をクリックします。

ウィンドウのサイズを変更しない

VuGen は，セッションの記録時におけるウィンドウのサイズ変更をサポートしていますが，記録中はウィンドウを動かしたり，ウィンドウの大きさを変えたりしないことをお勧めします。ウィンドウのサイズまたは位置を変更するには，スクリプトのツリー・ビューの中で該当する**ウィンドウで同期**ステップをダブルクリックし，ウィンドウの座標を変更します。

解像度の設定に矛盾がないことを確かめる

ビットマップの同期化を確実に成功させるために，解像度の設定が一致していることを確認します。記録用マシンの ICA クライアントの設定，記録オプション，および実行環境の設定を確認します。Load Generator で，ICA クライアントの設定を調べ，それらがすべての Load Generator マシンおよび記録用マシンと一致することを確認します。解像度間に不一致があると，必要な調整を行うためにサーバのトラフィックが増大します。

手動の同期ポイントを追加する

記録中に，イベント（たとえばアプリケーションの開始など）を待機する場合には，**ビットマップで同期**や**テキストで同期**などの同期化ポイントを手作業で追加することをお勧めします。詳細については，313 ページ「再生の同期化」を参照してください。

クライアントの更新を無効にする

Citrix クライアントの更新を有効にするかどうか尋ねられた場合には，クライアントの更新を無効にします。これにより，VuGen とまだテストされていない最新の Citrix クライアント間の前方互換性の問題が回避されます。

ウィンドウの形式

ビットマップで同期ステップの場合は，ウィンドウを XP 形式ではなく「クラシック」のウィンドウ形式で記録します。

ウィンドウの形式を「クラシック」に変更するには，次の手順を実行します。

1 デスクトップをクリックします。

2 右クリックして表示されるメニューから［**プロパティ**］を選択します。

3 ［テーマ］タブを選択します。

4 ［テーマ］ドロップ・ダウン・リストから，［**Windows クラシック**］を選択します。

5 ［**OK**］をクリックします。

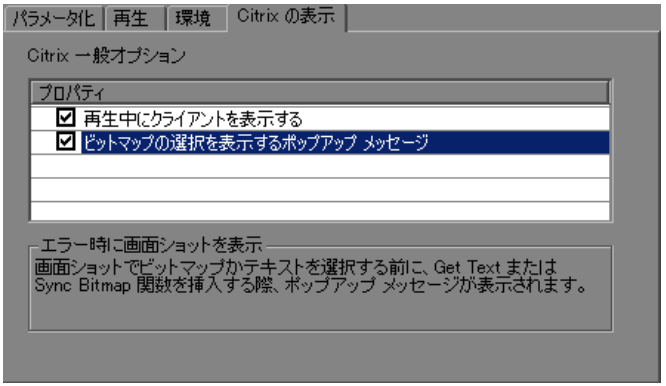
Citrix 表示の設定

Citrix Vuser スクリプトを実行する前に，再生中に使用される表示オプションをいくつか設定できます。これらのオプションを設定すると，サーバの負荷が大きくなりますが，セッションのデバッグと分析には役立ちます。

Citrix 表示オプションを設定するには，次の手順を実行します。

1 ［一般オプション］ダイアログ・ボックスを開きます。VuGen メイン・ウィンドウで，［**ツール**］＞［**一般オプション**］を選択します。

2 ［**Citrix の表示**］タブを選択します。



3 Vuser スクリプトの再生中に Citrix クライアントを表示するには，［**再生中にクライアントを表示する**］を選択します。

4 スナップショットの中で対話形式で作業を開始するときにポップアップ・メッセージを発行するには，［**ビットマップの選択を表示するポップアップ メッセージ**］を選択します。ビットマップまたはテキストを選択する前に，右クリック・メニュー・オプションで［**Insert Sync Bitmap**］または［**テキスト取得を挿入**］を選択すると，このメッセージが表示されます。

5 ［**OK**］をクリックします。

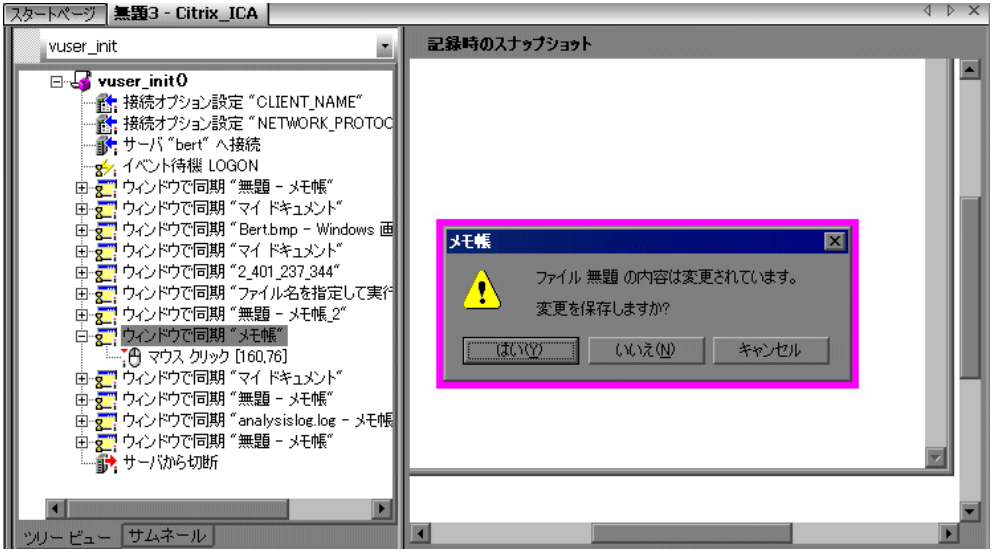
Citrix Vuser スクリプトの表示と変更

Vuser スクリプトの内容は，VuGen のスクリプト・ビューまたはツリー・ビューで表示できます。スクリプトの表示の詳細については，『**第 1 巻 ― VuGen の使用**』の「VuGen の紹介」を参照してください。

ツリー・ビューでは，Citrix Vuser のスナップショットを表示できます。各ステップには，スナップショットが関連付けられています。スナップショットでは，クライアント・ウィンドウが表示されるほか，アクションの実行対象オブジェクトも強調表示されます。

- **マウス・**ステップでは，ユーザがクリックした場所がピンク色の小さな四角形で示されます。
- **ビットマップで同期**では，ビットマップ領域がピンク色のボックスで囲まれます。

ウィンドウで同期では，ウィンドウ全体がピンク色のボックスで囲まれます。次の例では，スナップショットに**ウィンドウで同期**ステップが示されています。操作が実行されたウィンドウを正確に示すボックスによって，メモ帳の確認ボックスが囲まれています。



VuGen は，スナップショットをスクリプトの data/snapshots ディレクトリにビットマップ・ファイルとして保存します。スナップショット・ファイルの名前は関数の引数を調べることでわかります。

```
ctrx_sync_on_window("ICA Administrator Toolbar", ACTIVATE, 768, 0, 33,
                    573, "snapshot12", CTRX_LAST);
```

記録後，スクリプト・ビューかツリー・ビューのどちらかで，手作業でステップをスクリプトに追加できます。さまざまなスクリプト・ビューの詳細については，『**第 1 巻 － VuGen の使用**』の「VuGen の紹介」を参照してください。

新しい関数を手作業で追加するだけでなく，Citrix Vuser のために，新しいステップをスナップショットから直接，対話形式で追加できます。右クリック・メニューを使用して，ビットマップの同期化を追加できます。Citrix エージェントが Citrix サーバ・マシンにインストールされている場合は，右クリック・メニューを使用してテキストとオブジェクトの同期化を追加することもできます。詳細については，第 19 章「Citrix － Citrix Presentation Server エージェント」を参照してください。

関数を対話形式で挿入するには，次の手順を実行します。

1 ツリー・ビューの中のステップをクリックします。スナップショットが表示されていることを確認します。

2 右クリックしてコマンドのいずれか 1 つを選択します。ダイアログ・ボックスが開き，使用可能なプロパティが表示されます。

3 必要なプロパティを変更して［**OK**］をクリックします。VuGen によってステップがスクリプトに挿入されます。

再生の同期化

スクリプトの実行中，再生が正常に行われるようにするために，しばしばアクションを同期化する必要があります。同期化とは，スクリプトの中でイベントのタイミングを調整し，ウィンドウやオブジェクトが使用可能になるまで待つからアクションを実行することを指します。たとえば，ウィンドウ内のボタンを押す前に特定のウィンドウがすでに開いているかどうかを知りたい場合があります。

VuGen は再生中のアクションの同期化を行う関数を自動的に生成します。同期化関数は手作業で追加することもできます。

自動同期化

記録中，Vuser スクリプトの再生の同期化を支援するステップが VuGen によって自動的に生成されます。

- **ウィンドウで同期**
- **オブジェクト情報で同期**
- **テキストで同期**

ウィンドウで同期

ウィンドウで同期ステップは，指定されたイベントが発生するまで，Vuser に再生の再開を待機するよう指示するステップです。指定できるイベントは **Create** または **Active** です。Create イベントを指定した場合は，ウィンドウが作成されるまで待機します。Active イベントを指定した場合は，ウィンドウが作成されてアクティブになるまで（フォーカスを得るまで）待機します。VuGen は通常 CREATE イベントを待機する関数を生成します。しかし，次の命令がキーボード・イベントである場合，VuGen は ACTIVE イベントを待機する関数を生成します。

スクリプト・ビューでは，**ウィンドウで同期**ステップに対応する関数呼び出しは **ctrx_sync_on_window** です。

オブジェクト情報で同期

オブジェクト情報で同期ステップは，指定されたオブジェクト・プロパティが発生するまで，Vuser に再生の再開を待機するよう指示するステップです。使用可能な属性は，**Enabled**，**Visible**，**Focused**，**Text**，**Checked**，**Lines**，および **Item** です。Enabled，Visible，Focused，および Checked の各属性は，**true** または **false** を受け取ることができるブール値です。その他の属性は，テキストまたは数値のオブジェクト値を必要とします。

このステップの主な目的は，オブジェクトを対象とするアクションを実行する前に，当該オブジェクトにフォーカスが当たるのを待機することです。

Citrix エージェントがインストールされていて，記録オプションの［Use Citrix Agent Input in Code Generation］オプションが有効になっている場合，VuGen は自動的にオブジェクト情報で同期ステップを生成します。標準設定では，この記録オプションは有効になっています。詳細については，『**第 1 巻 － VuGen の使用**』の第 16 章「選択したプロトコルの記録オプション」を参照してください。

```
ctrx_sync_on_obj_info("Run=snapshot9", 120, 144, TEXT, "OK",
                                        CTRX_LAST);
```

テキストで同期

テキストの同期化ステップ，**テキストで同期**は，指定された位置にテキスト文字列が現れるまで，Vuser に続行を待機するよう指示します。**テキストで同期**の再生時，Vuser は，ステップのプロパティに指定されている変更可能な座標の矩形領域の中でテキストを検索します。

エージェントがインストールされている場合は（第 19 章「Citrix － Citrix Presentation Server エージェント」を参照），マウスのクリックまたはダブルクリックそれぞれの前にテキストの同期化ステップを自動的に生成するように VuGen を設定できます。標準設定では，自動的なテキストの同期化は無効になっています。この記録オプションを有効にする方法の詳細については，『**第 1 巻 － VuGen の使用**』の第 16 章「選択したプロトコルの記録オプション」を参照してください。

このオプションを無効にしてスクリプトを記録した場合でも，このオプションを有効にしてスクリプトを再生成すると，VuGen によってスクリプト全体にテキストの同期化呼び出しが挿入されます。詳細については，『**第 1 巻 － VuGen の使用**』の第 16 章「選択したプロトコルの記録オプション」を参照してください。

記録中および記録後，個々のステップに手作業でテキストの同期化を追加できます。詳細については，316 ページ「手作業での同期化」を参照してください。

スクリプト・ビューでは，**テキストで同期**ステップに対応する関数呼び出しは **ctrx_sync_on_text_ex** です。

次のコードの抜粋は，HP Citrix エージェントがインストールされていて，テキストの同期化が有効になっている Citrix の記録中に記録された ctrx_sync_on_text_ex 関数を示しています。

```
ctrx_sync_on_window ("ICA Seamless Host Agent", ACTIVATE, 0, 0,391,224,
"snapshot1", CTRX_LAST);
ctrx_sync_on_text_ex (196, 198, 44, 14, "OK", "ICA Seamless Host
Agent=snapshot2", CTRX_LAST);
ctrx_obj_mouse_click ("<class=Button text=OK>", 196, 198, LEFT_BUTTON,
0, "ICA Seamless Host Agent=snapshot2", CTRX_LAST);
```

この関数の詳細については，『**Online Function Reference**』（英語版）（［**ヘルプ**］＞［**関数リファレンス**］）を参照してください。

手作業での同期化

自動同期化に加えて，記録中および記録後，手作業で同期化を追加できます。通常，この機能は，実際のウィンドウは変更されていないのに，ウィンドウ内部のオブジェクトが変更された場合に使用します。ウィンドウは変更されていないので，VuGen は**ウィンドウで同期**ステップを検出も記録もしていません。

たとえば，特定のグラフィック・イメージがブラウザ・ウィンドウに出現するまで再生を待機する場合は，手作業で同期化を挿入します。また，複数のタブを持つ大きなウィンドウを記録している場合，新しいタブの内容が開くのを待機する同期化ステップを挿入できます。

次の項では，ビットマップを対象とする同期化について説明します。手作業による**テキストで同期**の追加の詳細については，333 ページ「テキストの取得」を参照してください。

記録中の手作業による同期化の追加

記録中に同期化を追加するには，フローティング・ツールバーを使用します。**ビットマップで同期**関数を使用すると，再生を再開する前にフォーカスを得る必要のあるクライアント・ウィンドウの領域を指定できます。

同期化するビットマップ領域をマークするには，次の手順を実行します。

1 ツールバーの［**ビットマップ同期を挿入**］ボタンをクリックします。

2 対象とするビットマップを囲むように矩形領域を指定します。ツリー・ビューの現在のステップの後にビットマップで同期ステップが生成されます。スクリプト・ビューでは，選択した座標を引数として ctrx_sync_on_bitmap 関数が生成されます。

```
ctrx_sync_on_bitmap(93, 227, 78, 52,
                    "66de3122a58baade89e63698d1c0d5dfa", CTRX_LAST);
```

再生中，Vuser は指定された座標でビットマップを探し，それが使用可能になるまで待って，テストを再開します。

記録後の手作業による同期化の追加

記録セッションの後に同期化を追加することもできます。同期化ステップを追加するには，［スナップショット］ウィンドウで右クリックし，次の同期化オプションを選択します。

- **ビットマップで同期**：ビットマップが表示されるまで待機します。
- **オブジェクト情報で同期**：オブジェクトの属性が，指定の値になるまで待機します（エージェントがインストールされている場合のみ）。
- **テキストで同期**：指定したテキストが表示されるまで待機します（エージェントがインストールされている場合のみ）。

記録中，**ビットマップで同期**ステップに生成されたビットマップは，スクリプトの **data¥snapshots** ディレクトリに保存されます。同期化が失敗した場合は，VuGen によって新しいビットマップが生成され，同期化が失敗した原因を調べることができます。VuGen は，両方のビットマップを［画像同期の失敗］ダイアログ・ボックスに表示します。詳細については，320 ページ「画像同期の失敗」を参照してください。

ビットマップ名は sync_bitmap_ ＜ hash_value ＞ .bmp の形式となります。このビットマップは，スクリプトの出力ディレクトリに格納されます。シナリオまたはプロファイルの場合には，出力ファイルの書き込み先に格納されます。

追加の同期化

前記に加えて，同期化に間接的に影響するその他のいくつかのステップを追加することもできます。

- 待ち時間の設定
- ウィンドウの表示 / 非表示の確認
- ビットマップ変更の待機

待ち時間の設定

待機時間設定ステップは，ほかの Citrix 同期化関数の待機時間を設定します。この設定は，同一スクリプト内でこの関数以降のすべての関数に適用されます。たとえば，**ウィンドウで同期**ステップがタイムアウトする場合，標準設定のタイムアウトである 60 秒を 180 秒に延ばすことができます。

このステップを挿入するには，［**挿入**］＞［**新規ステップ**］＞［**待機時間設定**］を選択します。

ウィンドウの表示 / 非表示の確認

ウィンドウ有無ステップは，Citrix クライアントでウィンドウが表示されているかどうかを調べます。フロー制御ステートメントを追加することにより，この関数を使用して，常に開いているとはかぎらない警告ダイアログ・ボックスなどのウィンドウを調べることができます。次の例では，ctrx_win_exist によってブラウザが起動されたかどうかを調べます。2 番目の引数は，ブラウザ・ウィンドウが開くまでの待機時間を示します。指定した時間開かなかった場合は，ブラウザによって自身のアイコンがダブルクリックされます。

```
if (!ctrx_win_exist("Welcome",6, CTRX_LAST))
   ctrx_mouse_double_click(34, 325, LEFT_BUTTON, 0, CTRX_LAST)
```

このステップを挿入するには，［**挿入**］＞［**新規ステップ**］＞［**ウィンドウ有無**］を選択します。

このステップのもう一つの有用な用途は，ウィンドウが閉じているかどうかを確認することです。ウィンドウが閉じるのを待機する必要がある場合は，**ウィンドウ設定解除**や **ctrx_unset_window** などの同期化ステップを使用するべきです。

これらの関数の詳細については，『**Online Function Reference**』（英語版）（［**ヘルプ**］＞［**関数リファレンス**］）を参照してください。

ビットマップ変更の待機

領域内に表示されるデータまたは画像がどのようなものかわからなくても，変更が行われることはわかっている場合があります。これをエミュレートするには，**ビットマップ変更時に同期化**ステップまたはその対応する関数 **ctrx_sync_on_bitmap_change** を使用します。スナップショットを右クリックし，右クリック・メニューから［**ビットマップ同期を挿入**］を選択します。ステップまたは関数がカーソルの位置に挿入されます。

この関数の構文は次のとおりです。

```
ctrx_sync_on_bitmap (x 座標 , y 座標 , 幅 , 高さ , ハッシュ値 , CTRX_LAST);
ctrx_sync_on_bitmap_change (x 座標 , y 座標 , 幅 , 高さ ,
                            [ 初期待機時間 ,] [ タイムアウト ,]
                            [ 初期ビットマップ値 ,] CTRX_LAST);
```

ctrx_sync_on_bitmap_change では，次のオプションの引数が使用できます。

- 初期待機時間の値 -- 変更の有無の確認を開始する時間。
- タイムアウト -- 失敗する前に変更が発生するまで待機する最大の秒数。
- 初期ビットマップ値 -- ビットマップの初期ハッシュ値。Vuser は，ハッシュ値が指定した初期ビットマップ値と異なる値になるまで待機します。

次の例では，記録された関数に変更を加えて，300 秒の初期待機時間と 400 秒のタイムアウトを割り当てています。

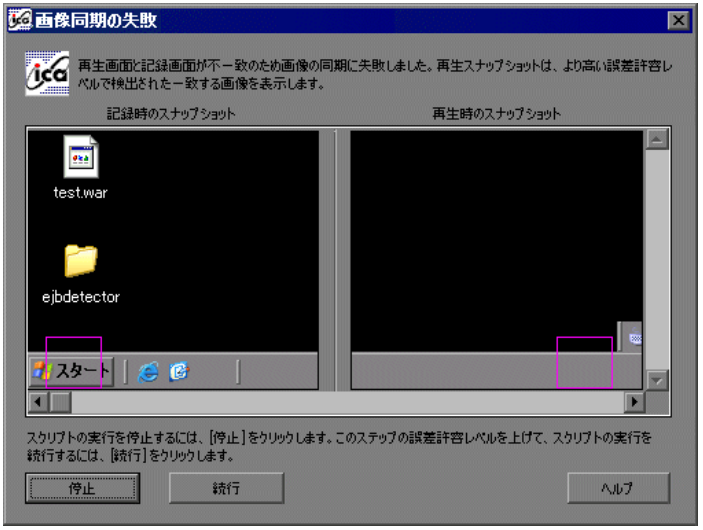
```
ctrx_sync_on_bitmap_change(93, 227, 78, 52,
                    300,400, "66de3122a58baade89e63698d1c0d5dfa",CTRX_LAST);
```

注 :［**ビットマップで同期**］を使用している場合は，Controller，Load Generator マシン，および画面の設定が同じであることを確認します。設定が同じでないと，VuGen が再生中に正しいビットマップを見つけることができない場合があります。クライアントの設定方法の詳細については，『**第 1 巻 ― VuGen の使用**』の第 16 章「選択したプロトコルの記録オプション」を参照してください。

画像同期の失敗

スクリプトの再生中に記録のスナップショットと再生のスナップショットの間に不一致があった場合，［画像同期の失敗］ダイアログ・ボックスが開きます。

不一致をエラーとしてマークするか，それとも変更を採用するか，指定できます。



このダイアログ・ボックスが開いたら，次のどちらかのボタンをクリックして続行します。

- ［**停止**］：スナップショット間の不一致をエラーとみなします。このエラーは，ほかのすべてのエラーと同じように処理され，実行を停止させます。特定の関数に［エラー時も処理を継続する］を指定できます。詳細については，326 ページ「エラー時も処理を継続する」を参照してください。
- ［**続行**］：不一致を受け入れ，元のスナップショットと新しいスナップショットの両方を，将来の再生時に画面間を比較するための基準として使用します。再生時にどちらか一方のビットマップが返された場合，Vuser は失敗しません。

ICA ファイルについて

Citrix ICA クライアント・ファイルは，Citrix クライアントを通じてアクセスされるアプリケーションの設定情報が含まれているテキスト・ファイルです。これらのファイルには，.ica 拡張子が付き，次の形式に準拠している必要があります。

```
[WFClient]
Version=
TcpBrowserAddress=

[ApplicationServers]
AppName1=

[AppName1]
Address=
InitialProgram=#
ClientAudio=
AudioBandwidthLimit=
Compress=
DesiredHRES=
DesiredVRES=
DesiredColor=
TransportDriver=
WinStationDriver=

Username=
Domain=
ClearPassword=
```

注：［記録オプション］を使って ICA ファイルをロードすると，VuGen によってファイルがスクリプトと一緒に保存されるので，ICA ファイルを各 Load Generator マシンにコピーする手間が省けます。

次の例は，Citrix クライアントを通じて Microsoft Word をリモート・マシン上で使うための ICA ファイルのサンプルです。

```
[WFClient]
Version=2
TcpBrowserAddress=235.119.93.56

[ApplicationServers]
Word=

[Word]
Address=Word
InitialProgram=#Word
ClientAudio=On
AudioBandwidthLimit=2
Compress=On
DesiredHRES=800
DesiredVRES=600
DesiredColor=2
TransportDriver=TCP/IP

WinStationDriver=ICA 3.0

Username=test
Domain=user_lab
ClearPassword=test
```

詳細については，Citrix の Web サイト www.citrix.com を参照してください。

Citrix 関数の使用方法

Citrix 記録セッション中，VuGen によってクライアントとリモート・サーバ間のやり取りをエミュレートする関数が生成されます。生成された関数には，**ctrx** というプレフィックスが付きます。どの関数も記録セッションの後，Vuser スクリプトを手作業で編集して追加できます。たとえば，**ctrx_obj_mouse_click** は，特定のオブジェクトのマウス・クリックをエミュレートします。

ctrx 関数の詳細については，『**Online Function Reference**』（英語版）（［**ヘルプ**］＞［**関数リファレンス**］）を参照してください。

ウィンドウ名を指定する関数では，ワイルドカード記号であるアスタリスク（*）を使用できます。ワイルドカードは文字列の先頭と末尾を含めて任意の場所に指定できます。

Citrix Vuser スクリプトの再生とトラブルシューティングのヒント

ここでは，Citrix Vuser の次の項目に関するガイドラインやヒントを示します。

- 再生に関するヒント
- デバッグに関するヒント

記録のヒントについては，309 ページ「記録に関するヒント」を参照してください。

再生に関するヒント

ワイルドカード

ウィンドウ名の定義には，ワイルドカード（*）を使用できます。これは，サフィックスまたはプレフィックスによって，再生中にウィンドウ名が変わる可能性がある場合に特に役立ちます。

次の例では，**Microsoft Internet Explorer** ウィンドウのタイトルをワイルドカードで置き換えています。

```
ctrx_mouse_click(573, 61, LEFT_BUTTON, 0,
"Welcome to MSN.com - Microsoft Internet Explorer");
ctrx_mouse_click(573, 61, LEFT_BUTTON, 0,
"* - Microsoft Internet Explorer");
```

詳細については，『**Online Function Reference**』（英語版）（［**ヘルプ**］＞［**関数リファレンス**］）を参照してください。

初期化クォータを設定する

接続中に複数の Vuser によって過負荷になるのを防ぐため，Vuser の初期化クォータを（サーバの性能に応じて）4 から 10 に設定するか，スケジューラを使用して Vuser のランプアップによる初期化を実施します。

思考遅延時間を有効にする

最良の結果を得るには，実行環境の設定で思考遅延時間を無効にしないようにします。思考遅延時間は，特に安定するまでに時間を要する **ctrx_sync_on_window** 関数や **ctrx_sync_on_bitmap** 関数の前には重要です。

スクリプトの再生成

VuGen は，記録中，スクリプトとともにすべてのエージェント情報を保存します。標準設定では，VuGen はこの情報をスクリプトにも含めます（**テキストで同期**ステップを除く）。テキストの同期化の問題が発生した場合は，スクリプトを再生成して，テキストの同期化ステップを含めることができます。

また，記録オプションでエージェント情報の生成を無効にした場合でも，スクリプトを再生成して，この情報を含めることができます。

スクリプトの再生成は，手作業で変更を加えたスクリプトにも役立ちます。スクリプトを再生成すると，VuGen は手作業による変更をすべて廃棄し，最初に記録されたバージョンに戻します。

スクリプトを再生成するには，［**ツール**］＞［**スクリプトの再生成**］を選択し，必要なオプションを選択します。スクリプトの再生成の詳細については，『**第 1 巻 － VuGen の使用**』の「Vuser スクリプトの再生成」を参照してください。

マシン間で一貫性を保つ

別のマシンでスクリプトを再生する場合には，記録マシンと再生マシンの間で，Citrix クライアントのウィンドウ・サイズ（解像度），ウィンドウの色設定，システム・フォント，その他の標準オプションの設定が同じであることを確認します。これらの設定はビットマップのハッシュ値に影響し，不一致があった場合は，再生が失敗する可能性があります。Citrix クライアントの設定を表示するには，Citrix プログラム・グループから項目を選択し，［**Application Set Settings**］を選択するか，右クリック・メニューから［**Custom Connection Settings**］を選択します。［Default Options］タブを選択します。

Load Generator マシンごとの Vuser 数の増加

Citrix Vuser を稼動する Load Generator マシンでは，マシンで使用可能なグラフィック・リソース（GDI － Graphics Device Interface）により，実行できる Vuser の数が制限される場合があります。マシンごとに実行できる Vuser の数を増やすには，マシンでターミナル・サーバ・セッションを開き，追加の Load Generator として動作させることができます。

GDI カウントはオペレーティング・システムによって異なります。LoadRunner を使用している負荷の重いマシンの実際の GDI（Graphics Device Interface）カウントは，約 7,500 です。Windows 2000 マシンで使用可能な GDI の最大数は，16,384 個です。

ターミナル・サーバ・セッションの作成方法の詳細については，『**HP LoadRunner Controller**』のターミナル・サービスに関する項を参照してください。

注：標準設定では，ターミナル・サーバでのセッションには，256 色のカラー・セットが使用されます。ターミナル・セッションを負荷テスト用に使用しようとしている場合は，必ず 256 色のカラー・セットを備えたマシンに記録してください。

デバッグに関するヒント

クライアントを 1 つだけインストールする

Citrix セッションに任意のアクションを記録するのに失敗した場合は，お使いのマシンに Citrix クライアントが 1 つしかインストールされていないことを確認してください。インストールされているクライアントが 1 つだけかどうかを調べるには，コントロールパネルから［プログラムの追加と削除］ダイアログ・ボックスを開き，Citrix ICA クライアントのエントリが 1 つだけあることを確認します。

ブレークポイントを追加する

問題が生じているコードの行を知るには，VuGen でスクリプトにブレークポイントを追加します。

スクリプトを同期化する

再生が失敗した場合，スクリプトに同期化関数を挿入して，対象ウィンドウがフォーカスを得るまで，待機時間を延ばす必要があるかもしれません。**lr_think_time** 関数を使って遅延時間を手作業で追加することもできますが，313 ページ「再生の同期化」で説明している同期化関数を使用することをお勧めします。

エラー時も処理を継続する

一致するウィンドウが見つからないなどのエラーが発生した後も実行を継続するように Vuser を設定できます。［エラー時も処理を継続する］は個々のステップに指定します。

これは，2 つのウィンドウのいずれかが開くことを知っているけれども，どちらが開くかわからない場合に特に役立ちます。つまり，どちらのウィンドウも正しいけれども，一方のみが開きます。

［エラー時も処理を継続する］を指定するには，次の手順を実行します。

ツリー・ビューでステップを右クリックし，［**プロパティ**］を選択します。［**Continue on Error**］ボックスで［**CONTINUE_ON_ERROR**］オプションを選択します。

スクリプト・ビューの中で関数を探し，**CTRX_LAST** の前に最後の引数として **CONTINUE_ON_ERROR** を追加します。

このオプションは次の関数には使用できません：**ctrx_key**，**ctrx_key_down**，**ctrx_key_up**，**ctrx_type**，**ctrx_set_waiting_time**，**ctrx_save_bitmap**，**ctrx_execute_on_window**，**ctrx_set_exception**。

拡張ログ

［拡張ログ］でほかの再生情報を参照できます。拡張ログを有効にするには，実行環境の設定（F4 ショートカット・キー）の［**ログ**］パネルを使います。この情報は［記録ログ］タブ，またはスクリプト・ディレクトリの output.txt ファイルで参照できます。

スナップショット・ビットマップ

エラーの発生時，VuGen はスクリプトの**出力**ディレクトリに画面のスナップショットを保存します。このビットマップを見て，エラーが起きた原因を確認できます。

記録中，**ctrx_sync_on_bitmap** 関数に生成されたビットマップはスクリプトの **data** ディレクトリに保存されます。ビットマップ名は ＜**ハッシュ値**＞ .bmp の形式となります。再生中に同期化に失敗した場合には，生成されたビットマップはスクリプトの出力ディレクトリに書き込まれます。あるいは，シナリオで実行している場合には，出力ファイルが書き込まれる場所に書き込まれます。新しいビットマップを確認して，同期化が失敗した理由を調べることができます。

Vuser の表示

シナリオ実行時に Vuser を表示するには，Vuser コマンド・ライン・ボックスに -lr_citrix_vuser_view と入力します。Controller で，［グループ情報］ダイアログ・ボックスを開き［**詳細表示**］をクリックして，ダイアログ・ボックスを拡張します。この操作は，テストのスケーラビリティに影響するので，問題のある Vuser の振る舞いを調査する場合にだけ行うようにします。

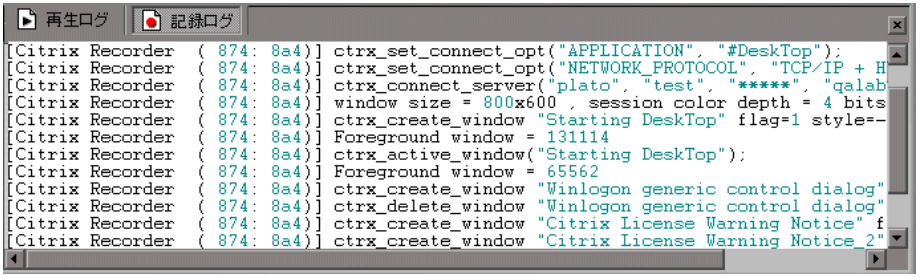
スクリプトのスケーラビリティに対する影響を減らすには，コマンド・ラインの最後に Vuser の ID を追加して，個々の Vuser の詳細を表示します。-lr_citrix_vuser_view ＜ **VuserID** ＞

複数の Vuser を開くには，コマンド・ラインの後に ID のカンマ区切りリストを置きます。スペースは使用できませんが，カンマとダッシュは使用できます。たとえば，1,3-5,7 と指定すると，Vuser 1，3，4，5，および 7 が表示されます。Vuser 2，6 は表示されず，ID が 7 より大きい Vuser も表示されません。

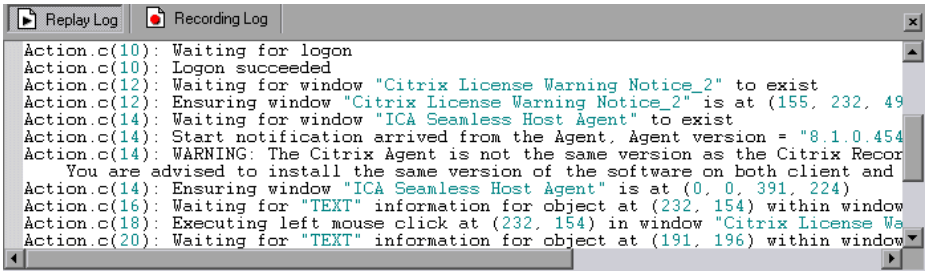
記録ログおよび再生ログの表示

記録に関する詳細情報を調べるには，［出力］ウィンドウに記録ログおよび再生ログを表示します。［出力］ウィンドウを開くには，［**表示**］＞［**出力ウィンドウ**］を選択します。

記録ログを表示するには，［**記録ログ**］タブを選択します。記録によって生成されたすべての関数，およびその時に発行された警告メッセージとエラーに関する詳細ログが表示されます。



再生ログを表示するには，［**記録ログ**］タブを選択します。VuGen によって実行されたすべてのアクション，および再生中に発行された警告とエラーに関する説明が表示されます。



ログ・メッセージに対応する，スクリプト内のステップに直接移動するには，再生ログの中で対象メッセージをダブルクリックします。

再生ログの情報の範囲は，ログの実行環境の設定によって変わります。詳細については，『**第 1 巻 － VuGen の使用**』の「実行環境の設定」を参照してください。

第 19 章

Citrix ― Citrix Presentation Server エージェント

Citrix Presentation Server エージェントは，Citrix サーバに任意でインストールできるユーティリティです。このユーティリティには，スクリプトを拡張できるいくつかの重要な利点があります。

本章の内容

Citrix Presentation Server エージェントについて

Citrix Presentation Server エージェントは，Citrix サーバに任意でインストールできるユーティリティです。このユーティリティにより，通常の Citrix 機能が強化されます。このユーティリティは，製品のインストール・ディスクに含まれており，任意の Citrix サーバにインストールできます。

このエージェントには次の利点があります。

- 直感的かつ可読性の高いスクリプト
- 組み込み同期化

- 関連オブジェクトの詳細な情報
- クライアント・ウィンドウ内で対話的に作業することが可能

Citrix Presentation Server エージェントの機能の使用

Citrix Presentation Server エージェントにより，通常の Citrix 機能が強化されます。次の項では，その機能強化について説明し，詳細とスクリプトのサンプルを示します。

オブジェクトの詳細な記録

Citrix Presentation Server エージェントがインストールされると，VuGen は，アクションに関する一般的な情報の代わりにアクティブ・オブジェクトの特定の情報を記録します。たとえば，VuGen は，エージェントなしで生成される**マウス・クリック**および**マウス・ダブル・クリック**の代わりに**オブジェクト・マウス・クリック**および**オブジェクト・マウス・ダブル・クリック**・ステップを生成します。

次の例は，エージェントをインストールした場合とインストールしない場合とで記録された同じマウスクリック操作を示しています。エージェントを使用する場合，VuGen はクリック，ダブルクリック，解放などのすべてのマウス操作に対応する ctrx_obj_xxx 関数を生成します。

```
/* エージェントをインストールしない場合 */ctrx_mouse_click(573, 61,
LEFT_BUTTON, 0, test3.txt - Notepad);

/* エージェントをインストールした場合 */
ctrx_obj_mouse_click("<text=test3.txt - Notepad class=Notepad>" 573,
        61, LEFT_BUTTON, 0, test3.txt - Notepad=snapshot21, CTRX_LAST);
```

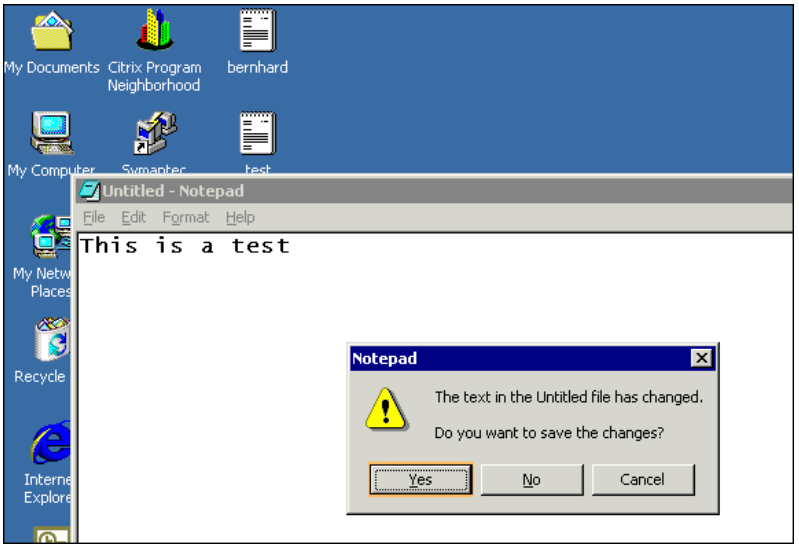
上の例で，**ctrx_obj_mouse_click** 関数の 1 番目の引数には，ウィンドウのタイトルとクラスのテキスト，つまりメモ帳が格納されています。エージェントは個々のオブジェクトに関する追加情報を提供しますが，Vuser はウィンドウ名とオブジェクトの座標によってのみ，オブジェクトにアクセスします。

アクティブ・オブジェクトの認識

エージェントをインストールすると，クライアント・ウィンドウでどのようなオブジェクトが VuGen によって検出されたのかを調べることができます。これには，現在のウィンドウ内の編集ボックスやボタン，項目リストなど，Windows の基本的なオブジェクトがすべて含まれます。

どのオブジェクトが検出されたのかを調べるには，スナップショットの上でマウスを動かします。マウスがオブジェクト上を通過すると，検出されたオブジェクトの境界が強調表示されます。

次の例では，検出されたオブジェクトの 1 つは［**Yes**］ボタンです。



拡張された右クリック・メニュー

スナップショット内をクリックした後，右クリック・メニューを使用して，いくつかの関数をスクリプトに挿入できます。エージェントがインストールされていない場合は，**マウスのクリックを挿入**，**マウスのダブルクリックを挿入**，**ビットマップ同期を挿入**，および**ビットマップ値取得を挿入**に限られます。256 色表示を使用している場合は，右クリック・メニューから **ビットマップ同期を挿入**ステップおよび**ビットマップ値取得を挿入**ステップを使用することはできません。

Citrix Presentation Server エージェントがインストールされている場合は，フォーカスを得ているウィンドウの右クリック・メニューから次の追加オプションを使用できます。

- **オブジェクト情報取得**と**オブジェクト情報同期**：オブジェクトの状態に関する情報として，ENABLED，FOCUSED，VISIBLE，TEXT，CHECKED，および LINES を提供します。
- **オブジェクト情報同期を挿入**：特定の状態になるまで待ち，それから処理を続けるよう VuGen に指示します。このオプションは，**ctrx_sync_on_obj_info** 関数として生成されます。
- **オブジェクト情報取得を挿入**：任意のオブジェクトのプロパティの現在の状態を取得します。このオプションは，**ctrx_get_obj_info** 関数として生成されます。
- **テキスト同期を挿入**と**テキスト取得**：詳細については，333 ページ「テキストの取得」の項を参照してください。

これらのコマンドは対話形式であり，スクリプトに挿入するときはスナップショット内のオブジェクトまたはテキスト領域をマークします。

次の例では，ctrx_sync_on_obj_info 関数は，［フォント］ダイアログ・ボックスがフォーカスを得るまで待機することによって同期化を実現します。

```
ctrx_sync_on_obj_info("Font", 31, 59, FOCUSED, "TRUE", CTRX_LAST);
```

VuGen のオブジェクト検出機能を使用して，特定のオブジェクトに対してスナップショットの中から対話形式でアクションを実行できます。

エージェントの機能を使用して関数を対話形式で挿入するには，次の手順を実行します。

1 ツリー・ビューの中で，新しいステップを挿入する場所をクリックします。スナップショットが表示されていることを確認します。

2 スナップショットの中をクリックします。

3 ビットマップをマークするには，ビットマップを右クリックして［**ビットマップ同期を挿入**］を選択します。

カーソルをドラッグして対象領域をマークする必要があることを示すメッセージが表示されます。［**OK**］をクリックし，選択するビットマップを対角方向にドラッグします。

マウスのボタンを放すと，スクリプトで現在選択されているステップの後に，ステップが挿入されます。

4 ほかのすべてのステップの場合は，スナップショット・オブジェクトの上でマウスを移動し，どの項目がアクティブなのかを決定します。マウスがオブジェクト上を通過すると，アクティブなオブジェクトの境界が強調表示されます。

［挿入］コマンドの 1 つを右クリックして選択します。ダイアログ・ボックスが開き，ステップのプロパティが表示されます。

必要なプロパティを設定して［**OK**］をクリックします。VuGen によってステップがスクリプトに挿入されます。

テキストの取得

エージェントをインストールすると，標準のテキストをバッファに保存できます。VuGen では純粋なテキストのみ保存できます。画像の形式でグラフィカルに表現されているテキストは保存できません。

テキストは，記録中または記録後に，**テキストで同期**ステップを使用して保存します。

文字列を取得するには，次の手順を実行します。

1 記録中：ツールバーにある［**テキスト同期を挿入**］ボタンをクリックします。

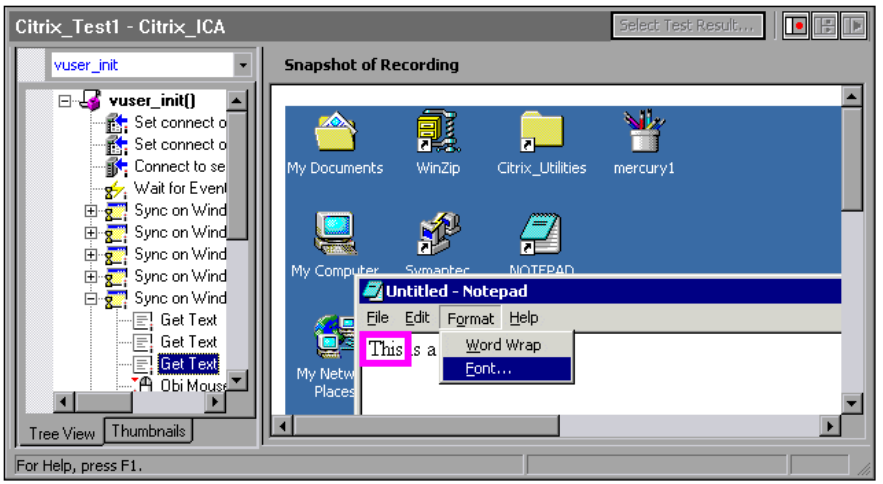
記録後：スナップショットの右クリック・メニューから［**テキスト同期を挿入**］を選択します。［ビットマップの選択］ダイアログが開き，同期化関数または情報提供関数を挿入するため領域をマークする必要があることを示します。

2 キャプチャするテキストの角をクリックし，マウスを対角方向にドラッグして保存するテキストをマークし，マウス・ボタンを放します。

3 記録中にステップを追加した場合は，現在の位置に**テキスト同期を挿入**ステップが置かれ，テキストがバッファに保存されます。

記録後にステップを追加した場合は，［Sync On Text Ex］ダイアログ・ボックスが表示され，手作業でテキストを指定できます。

保存されたテキストはピンク色のボックスでマークされます。次のスナップショットでは，**テキスト同期を挿入**ステップによってテキスト **This** を取得しています。



データ実行防止（DEP）と Citrix パフォーマンス

DEP は，Windows XP Service Pack 2 以降に含まれているセキュリティ機能です。DEP は，Citrix Presentation Server エージェントの一部の機能に干渉する可能性があります。エージェントが適切に実行されるように，DEP を無効にしておくことをお勧めします。

DEP を無効にするには，次の手順を実行します。

1 デスクトップで［**マイ コンピュータ**］を右クリックし，［**プロパティ**］を選択します。あるいは，［**スタート**］＞［**コントロール パネル**］＞［**システム**］をクリックします。［**システムのプロパティ**］ダイアログ・ボックスが開きます。

2［**詳細設定**］タブをクリックします。

3［**起動と回復**］表示枠の［**設定**］をクリックします。

4［**起動システム**］にある［**編集**］をクリックします。

5 **/noexecute** 要素を探し，値を **AlwaysOff** に変更します。

6 ファイルを保存し，マシンを再起動します。

Citrix Presentation Server エージェントのインストール

Citrix Presentation Server エージェントのインストール・ファイルは，LoadRunner インストール・ディスクの **Additional Components¥Agent for Citrix Presentation Server** フォルダにあります。

Citrix エージェントは，Load Generator マシンではなく，Citrix サーバ • マシンにのみインストールされなければなりません。

エージェントをアップグレードする場合は，新しいバージョンをインストールする前に，前のバージョンをアンインストールしてください。

Citrix Presentation Server エージェントをインストールするには，次の手順を実行します。

1 サーバへのソフトウェアのインストールに管理者権限が必要な場合は，サーバに管理者としてログインします。

2 リモート・デスクトップ接続（RDP）を使用して，Windows 2003 が実行されているマシンにエージェントをインストールする場合は，インストールを開始する前に，対象マシンで次のコマンドを実行します。

```
Change user /install
```

3 製品のインストール・ディスクの **Additional Components¥Agent for Citrix Presentation Server¥Win32** または **Win64** ディレクトリで，インストール・ファイル **Setup.exe** を探します。

4 インストール・ウィザードに従ってインストールを完了します。

注：インストール後，LoadRunner から Citrix セッションが呼び出されると Citrix エージェントがアクティブになります。LoadRunner なしで Citrix セッションを開始してもアクティブにはなりません。

エージェントを無効にするには，アンインストールする必要があります。

Citrix Presentation Server エージェントをアンインストールするには，次の手順を実行します。

1 サーバからのソフトウェアの削除に管理者権限が必要な場合は，サーバに管理者としてログインします。

2 サーバ・マシンのコントロール・パネルから［**プログラムの追加と削除**］を開きます。「**HP Software Agent for Citrix Presentation Server 32 または 64**」を選択し，［**変更と削除**］をクリックします。

Citrix エージェントの効果と記憶容量

エージェントのインストールされた Citrix Vuser を実行すると，各 Vuser は **ctrxagent.exe** の独自の手順を実行します。その結果，サーバ・マシンで実行できる Vuser の数がわずかに減少します（約 7%)。

Citrix Vuser ごとの記憶容量（各 Vuser がそれぞれ **ctrxagent.exe** プロセスを実行した場合）は約 4.35 MB です。25 Vuser を実行するには，110 MB のメモリが必要です。

サンプル・スクリプト

次のスクリプトは，エージェントを含む実際の Citris ICA セッションを示します。

```
vuser_init()
{
   ctrx_set_connect_opt (NETWORK_PROTOCOL, "TCP/IP + HTTP");
   ctrx_connect_server ("Plato", "test", lr_decrypt("428c4445a14409b9"), "QAlab");
   ctrx_wait_for_event ("LOGON");
   ctrx_sync_on_window ("ICA Seamless Host Agent", ACTIVATE, 0, 0,391,224,
"snapshot1", CTRX_LAST);
   ctrx_sync_on_text (196, 198, "OK", TEXT, "ICA Seamless Host Agent=snapshot2",
CTRX_LAST);
   ctrx_obj_mouse_click ("<class=Button text=OK>", 196, 198, LEFT_BUTTON, 0, "ICA
Seamless Host Agent=snapshot2", CTRX_LAST);
   lr_think_time(73);
return 0;
}
```

第 20 章

リモート・デスクトップ・プロトコル（RDP）

VuGen では，Microsoft リモート・デスクトップ・プロトコル（RDP）を使ってサーバとやり取りを行うクライアントのアクションを記録できます。記録の結果として作成されるスクリプトを，「RDP Vuser スクリプト」と呼びます。

本章の内容

Microsoft リモート・デスクトップ・プロトコル（RDP）Vuser スクリプトについて

ユーザは，Microsoft リモート・デスクトップ・プロトコル（RDP）によってリモート・コンピュータに接続できるようになります。たとえば，特定のビジネス・アプリケーションや画像端末で作業するために，RDP を使用して強力な中央サーバに接続できます。これにより，ユーザは，スタンドアロン PC で作業しているのと同じルック・アンド・フィールが得られます。

注：RDP バージョン 5.1 以降には，さまざまなオプションを設定できる［**エクスペリエンス**］タブがあります。VuGen の記録では，このタブはサポートされません。オプションはすべて ON に設定されます。

記録に関するヒント

スクリプトを記録するときは，効果的なスクリプトを作成できるように必ず次のガイドラインに従ってください。

シングル・プロトコル・スクリプトとマルチ・プロトコル・スクリプト

スクリプトを新規に作成するときは，シングル・プロトコルかマルチ・プロトコルのスクリプトを作成できます。たとえば，RDP トラフィックと Web 応答の両方を記録するには，両方のプロトコルが記録されるように，RDP と Web のマルチ・プロトコル・スクリプトを作成します。詳細については，『**第 1 巻 － VuGen の使用**』の「VuGen を使った記録」を参照してください。

適切なセクションに記録する

接続処理は「**vuser_init**」セクションに，終了処理は「**vuser_end**」セクションに記録します。これにより，接続もしくは接続解除時に反復が実行されないようになります。セクションへの記録の詳細については，『**第 1 巻 － VuGen の使用**』の第 5 章「VuGen を使った記録」を参照してください。

初期状態のセッションを実行する

セッションを記録するときは必ず，接続操作から始まって，クリーンアップで終わる完全なビジネス・プロセスを実行するようにします。ビジネス・プロセス全体を始めから再実行できるところでセッションを終了するようにします。クライアントやアプリケーションのウィンドウを開いたままにしないようにします。

また，切断されたセッションを終了するようにターミナル・サーバを設定する必要があります。［**管理ツール**］＞［**ターミナル サービスの構成**］＞［**接続プロパティ**］＞［**セッション**］＞［**ユーザー設定より優先にする**］を選択して，切断されたセッションを終了するようにサーバを設定します。

明示的にクリックする

サブメニューを開くときは，メニューの自動展開に頼らずに，各オプションを明示的にクリックします。たとえば，［**スタート**］＞［**プログラム**］＞［**Microsoft Word**］の場合は，「**プログラム**」という語をクリックします。

RDP Vuser の記録

記録を行う前に，前述のように必要な記録オプションを設定します。

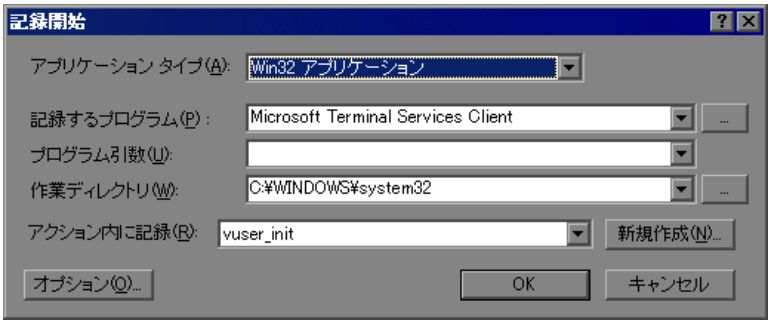
注：RDP スクリプトを記録するときは必ず，サーバからログオフする動作を記録するようにしてください。これにより，スクリプトの実行時，新しいセッションから開始されるようになります。

記録中のどの時点でも，同期化対象リモート・デスクトップ領域をマークできます。画像は **snapshot_xxx.png** という名前でディスクに格納されます。**xxx** は，1 から始まり，1 画像につき 1 ずつ増えていく順次インデックスです。このファイルには，マウス・クリックなど，ほかのステップの画像が含まれます。コード生成時には，このポイントでスクリプトにマッチング関数が現れます。

RDP Vuser スクリプトを作成するには，次の手順を実行します。

1 ［**ファイル**］＞［**新規作成**］を選択するか，［**新規作成**］ボタンをクリックします。［新規仮想ユーザ］ダイアログ・ボックスが開きます。

2 ［**Microsoft Remote Desktop Control Protocol (RDP)**］を選択します。［記録開始］ダイアログ・ボックスが開きます。



3 ［**オプション**］をクリックして，記録オプションを設定します。詳細については，『**第 1 巻 － VuGen の使用**』の 289 ページ「RDP 記録オプション」を参照してください。

4 ［RDP］の［**ログイン**］ノードを選択します。セッション・オプション（［**RDP クライアント アプリケーションの実行**］，［**カスタム接続ファイルを使用**］，［**標準設定接続ファイルを使用**］）の中から 1 つ選択します。

5 ［RDP］の［**コードの生成**］ノードを選択し，必要なオプションを有効にします。

6 記録中に同期化対象画面領域を選択するには，記録ツールバーの［**画像による同期**］ボタンをクリックし，同期化対象領域を指定します。

注：画像の同期化は，スクリプトの記録後に追加することもできます。画像のスナップショットを右クリックし，メニューから［**画像による同期を挿入**］を選択します。

---

7 記録を停止し，スクリプトを保存します。

RDP Vuser スクリプトの実行

RDP スクリプトを実行する前に，実行環境を設定してスクリプトの動作をカスタマイズします。思考遅延時間，反復，ペース，ログなど，すべてのプロトコルの一般的な実行環境を設定できます。詳細については，『**第 1 巻 － VuGen の使用**』の「実行環境の設定」を参照してください。

RDP 固有の設定については，前述を参照してください。

RDP スクリプトを実行するには，次の手順を実行します。

1 ［実行環境設定］ダイアログ・ボックスを開きます。ツールバーにある［**実行環境の設定**］ボタンをクリックするか，［**仮想ユーザ**］＞［**実行環境の設定**］を選択します。

2 ［**設定**］ノードを選択します。使用する設定を選択します。

3 ［**同期化**］ノードを選択します。使用する設定を選択します。

4 ［**OK**］をクリックして実行環境の設定を適用して，ダイアログ・ボックスを閉じます。

5 ［実行］ボタンをクリックするか，［**仮想ユーザ**］＞［**実行**］を選択します。

スクリプトを作成した後，スクリプトを自分の環境（LoadRunner シナリオ，Performance Center 負荷テスト，または Business Process Monitor プロファイル）に組み込みます。詳細については，**HP LoadRunner Controller**，**HP Performance Center**，または **HP Business Availibility Center** のドキュメントを参照してください。

クリップボード・データを使った作業

VuGen では，RDP セッション中のクリップボードのテキスト内容をコピーして貼り付けるできます。テキスト内容は，ローカルでコピーしてリモートで貼り付けることができます。あるいは逆に，リモート・マシンからコピーしてローカルで貼り付けることもできます。テキストのコピーでサポートされている形式は，TEXT，LOCALE，および UNICODE です。

クリップボード・データが提供または保存されると，VuGen は別個の関数を生成します。

次の例では，ローカル・マシンでのコピー操作とリモート・マシンでの貼り付け操作を示します。

```
// ローカル・マシンのクリップボードの新規データが利用可能であることをリモート・デスクトップに通知する // データは 3 つの形式（TEXT，UNICODE，LOCALE）で提供可能
rdp_notify_new_clipboard_data(
"StepDescription=Send local clipboard formats 1",
"Snapshot=snapshot1.inf",
"FormatsList=RDP_CF_TEXT|RDP_CF_UNICODE|RDP_CF_LOCALE",
RDP_LAST );

rdp_key(
"StepDescription=Key Press 2",
"Snapshot=snapshot_9.inf",
"KeyValue=V",
"KeyModifier=CONTROL_KEY",
RDP_LAST );

// 要求があったら，クリップボード・データをリモート・デスクトップに提供する
rdp_send_clipboard_data(
"StepDescription=Set Remote Desktop clipboard 1",
"Snapshot=snapshot1.inf",
"Timeout=20",
REQUEST_RESPONSE, "Format=RDP_CF_UNICODE", "Text=text for clipboard",
RDP_LAST);
```

次の例では，リモート・マシンでのコピー操作とローカル・マシンでの貼り付け操作を示します。

```
rdp_key(
"StepDescription=Key Press 2",
"Snapshot=snapshot_9.inf",
"KeyValue=C",
"KeyModifier=CONTROL_KEY",
RDP_LAST);

// リモート・デスクトップの UNICODE テキストを要求し，// そのテキストをパラメータに保存する
rdp_receive_clipboard_data(
"StepDescription=Get Remote Desktop clipboard 1",
"Snapshot=snapshot1.inf",
"ClipboardDataFormat=RDP_CF_UNICODE",
"ParamToSaveData=MyParam",
RDP_LAST);
```

通常，リモート・デスクトップのクリップボード・データは UNICODE 形式で保存されます。リモート・デスクトップが TEXT 形式または LOCALE 形式のデータを要求した場合，**rdp_send_clipboard_data** 関数は，自動的に MyParam の内容を UNICODE から要求された形式に変換し，リモート・デスクトップに送信します。再生ログには，この変換が情報メッセージ付きで示されます。変換が不可能な場合，ステップは失敗します。

rdp 関数の詳細については，『**Online Function Reference**』（英語版）（［**ヘルプ**］＞［**関数リファレンス**］）を参照してください。

クリップボード・パラメータの相関

記録セッション中，クライアントが，受信したのと同じデータをサーバに送信した場合，VuGen は，コード生成時に，送信されたデータをパラメータで置き換えます。この相関は，受信データと送信データの形式が一致している場合にのみ行われます。

次の例は，**MyParam** という同じパラメータがデータの受信と送信の両方にどのように使用されるかを示しています。

```
// サーバからデータを受信する
rdp_receive_clipboard_data("StepDescription=Get Remote Desktop clipboard 1",
"Snapshot=snapshot_9.inf",
"Timeout=0",
"ClipboardDataFormat=RDP_CF_UNICODETEXT",
"ParamToSaveData=MyParam",
RDP_LAST);
...
// サーバにデータを送信する
rdp_send_clipboard_data("StepDescription=Get Remote Desktop clipboard 1",
"Snapshot=snapshot_9.inf",
"Timeout=10",
REQUEST_RESPONSE, "Format=RDP_CF_UNICODETEXT", "Text={MyParam}",
RDP_LAST);
```

再生の同期化

RDP セッションはリモートで実行されます。キーボードとマウスの処理はすべてサーバで行われます。対応するのはサーバなのです。たとえば，デスクトップ上のアプリケーションがダブルクリックされたとき，ダブルクリックが発生したことを認識し，アプリケーションをロードして表示しなければならないのはサーバです。

RDP クライアントは，サーバに接続する際，次の 2 つのことを行います。

- アクションの座標をサーバに送信します。
  例：画面の (100, 100) の座標でマウスの左ボタンがクリックされた。
- アクション発生後の画面の現在のステータスを示す画像をサーバから受信します。

  RDP クライアントは（したがって LoadRunner も），画面にウィンドウ，ボタン，アイコン，もしくはそれ以外のオブジェクトが含まれているかどうかわかりません。わかるのは，画面に画像が含まれているということと，ユーザがアクションを実行した座標だけです。サーバがアクションを適切に解釈できるように，スクリプト内に同期化ポイントを設定します。このポイントにより，サーバの画像と格納されている画像が一致するまで，スクリプトに続行を待機させることができます。

画像の同期化ポイントをスクリプトに追加するには，次の手順を実行します。

1 ツリー・ビューでスクリプトを表示します。［**表示**］＞［**ツリー ビュー**］を選択します。

2 同期化ポイントを追加する操作を選択します。

3 画像のスナップショットを右クリックし，メニューから［**画像による同期を挿入**］を選択します。カーソルが十字形に変わります。

4 同期化する画面上の領域をマークします。左クリックしてドラッグし，領域をボックスで囲みます。マウス・ボタンを放すと，［画像による同期］ダイアログ・ボックスが開きます。

5 ［**OK**］をクリックします。選択したステップの前に新しい Sync on Image ステップが追加されます。このステップを選択すると，同期化のために選択した領域がピンク色のボックスで囲まれているスナップショットが表示されます。

次にスクリプトを再生したとき，スクリプトは，サーバによって返された画像と選択した画像が一致するまで待機します。

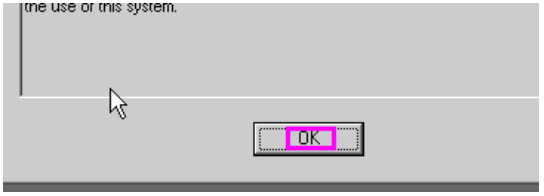
画像の同期化に関するヒント

画像の同期化を効果的に行うために，次のガイドラインを参考にしてください。

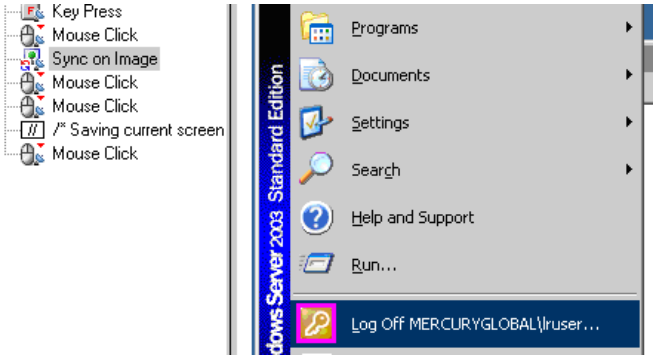
最小の有効領域を同期化する

画像を同期化する場合は，画像の必要な部分だけを同期化するようにしてください。画像内の余分な細部は，再生時に再現されないことがあり，同期化の失敗を招く可能性があります。

たとえば，ボタンの画像を同期化する場合はテキストのみ選択し，テキストの周りの点線は，再生時に表示されないことがあるため選択しないようにします。



強調表示された領域を同期化する場合は，画像の中で強調表示の影響を受けない部分のみキャプチャするようにしてください。次の例では，ボタン全体ではなく，ログオフ・アイコンを同期化しています。これは，強調表示は再生時に表示されないことがあり，色も配色パターンによって変わる可能性があるためです。



各ユーザ・アクションの前で同期化する

各マウス操作の前で同期化する必要があります。また，マウス操作に続く最初の **rdp_key**/**rdp_type** 操作の前で同期化することをお勧めします。

画像同期の失敗

スクリプトの再生中に記録のスナップショットと再生のスナップショットの間に不一致があった場合，［画像同期の失敗］ダイアログ・ボックスが開きます。［画像同期の失敗］ダイアログ・ボックスにはいくつかの種類があります。

- スナップショットを追加する
- 誤差許容レベルを上げる
- 誤差許容レベルを下げる
- スナップショットなし

注：このダイアログ・ボックスから許容レベルを上げた場合，または下げた場合，変更されるのはそのステップのレベルだけです。スクリプト全体の許容レベルを変更するには，［**画像の同期の標準設定の許容レベル**］の設定を変更します。詳細については，『**第 1 巻 ― VuGen の使用**』の第 24 章「選択したプロトコルの実行環境の設定」を参照してください。

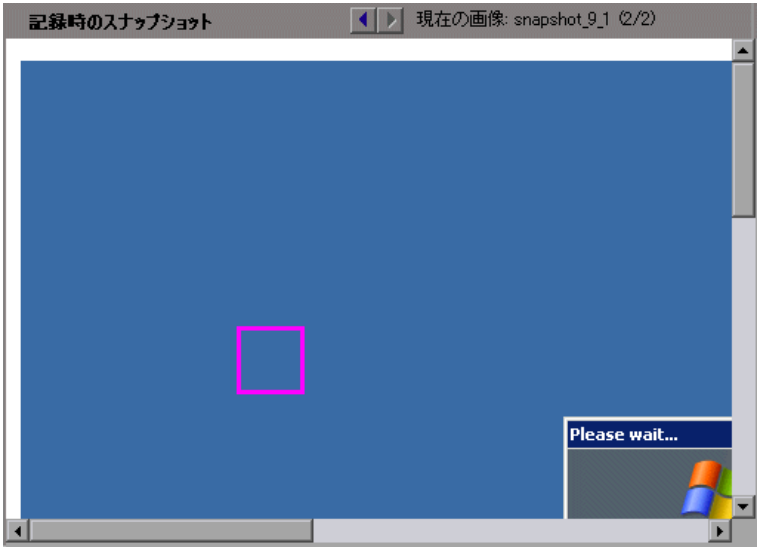
スナップショットを追加する

再生画像と記録画像が大きく違っていて，許容レベルの変更が役に立たない場合は，［画像同期の失敗 ― スナップショットを追加する］ダイアログ・ボックスが開きます。不一致をエラーとしてマークするか，それとも変更を採用するか，指定できます。

このダイアログ・ボックスが開いたら，次のどちらかのボタンをクリックして続行します。

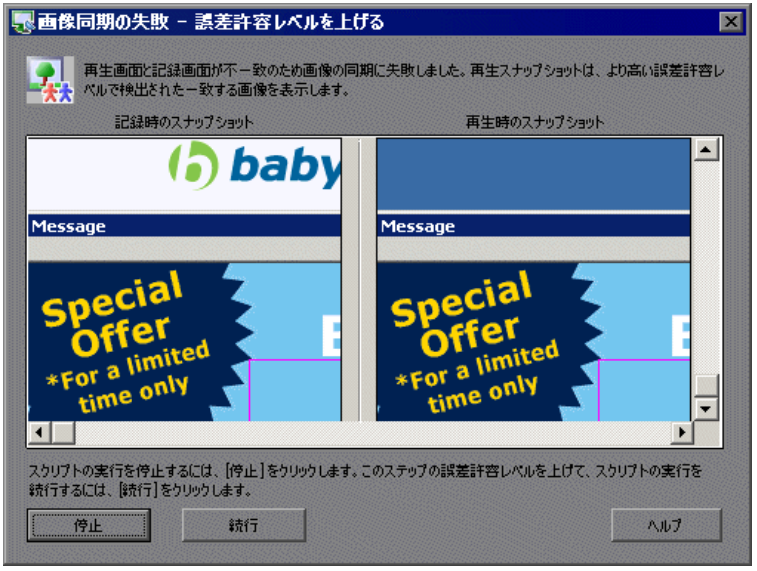
- ［**停止**］：スナップショット間の不一致をエラーとみなします。このエラーは，ほかのすべてのエラーと同じように処理され，実行を停止させます。
- ［**続行**］：不一致を受け入れ，元のスナップショットと新しいスナップショットの両方を，将来の再生時に画面間を比較するための基準として使用します。再生時にどちらか一方のビットマップが返された場合，Vuser は失敗しません。

元のスナップショットに新しいスナップショットを追加すると，そのスナップショットが現在のステップに追加されます。記録時のスナップショットの上の矢印をクリックすると，元のスナップショットと追加されたスナップショットの両方を表示できます。再生時のスナップショットには，1 つの画像，つまり再生中に見つかった画像だけが表示されます。



誤差許容レベルを上げる

スクリプトの再生で，要求された正確な画像を見つけられなかった場合は，［画像同期の失敗 ー 誤差許容レベルを上げる］ダイアログ・ボックスが開きます。しかし，画像の同期化の許容レベルが厳しくなければ，画像を発見できるかもしれません。



このダイアログ・ボックスが開いたら，次のどちらかのボタンをクリックして続行します。

- ［**停止**］：スナップショット間の不一致をエラーとみなします。このエラーは，ほかのすべてのエラーと同じように処理され，実行を停止させます。
- ［**続行**］：不一致を受け入れ，記録画像と再生中に表示された画像の不一致の度合いが大きくても容認されるように許容レベルを上げます。

誤差許容レベルを下げる

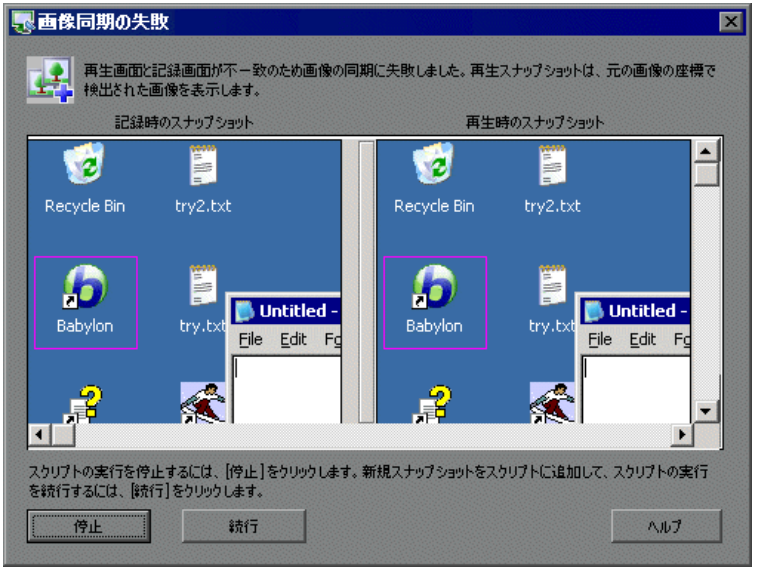
スクリプトの再生で，**NotAppear** 条件または **Change** 条件を満たすことができなかった場合は，［画像同期の失敗 ― 誤差許容レベルを下げる］ダイアログ・ボックスが開きます。VuGen は，検出することを期待されていなかった画像の一致を検出しました。許容レベルが下がれば，記録画像と再生画像が一致することはなく，**NotAppear** 条件または **Change** 条件が満たされ，結果的に再生が成功することになります。

このダイアログ・ボックスが開いたら，次のどちらかのボタンをクリックして続行します。

- ［**停止**］：スナップショット間の不一致をエラーとみなします。このエラーは，ほかのすべてのエラーと同じように処理され，実行を停止させます。
- ［**続行**］：不一致を受け入れ，記録画像と再生中に表示された画像の不一致の度合いが小さくても容認されるように許容レベルを下げます。

スナップショットなし

スクリプトの再生で，**NotAppear** や **Change** などの同期化条件を満たすことができなかった場合は，［画像同期の失敗］ダイアログ・ボックスが開きます。VuGen は，元の座標で，スクリプトに追加された可能性のある別の画像を見つけられませんでした。不一致に関係なく再生を続行するように VuGen を指定できます。この方法は，欠落している画像がビジネス・プロセスにとって重要ではないことがわかっている場合に最適です。



このダイアログ・ボックスが開いたら，次のどちらかのボタンをクリックして続行します。

- ［**停止**］：スナップショット間の不一致をエラーとみなします。このエラーは，ほかのすべてのエラーと同じように処理され，実行を停止させます。
- ［**続行**］：不一致を受け入れ，スクリプトの変更は行いません。不一致に関係なく，スクリプトの実行を続けます。

座標の移動

スクリプトの再生中に，記録オブジェクトが画面上の異なる座標に表示されることがあります。オブジェクトは同じですが，配置が変わります。たとえば，ウィンドウが記録中には (100, 100) の座標で開き，再生中には (200, 250) の座標で開くといった具合です。

この場合は，調整しなくても，同期化ポイントが自動的に新しい座標を見つけます。横軸には 100 ピクセルの差があり，縦軸には 150 ピクセルの差があることを自動的に認識します。

座標に依存する後続のすべてのマウス操作では，変更された座標が使用されます。したがって，(130, 130) で記録されたマウス・クリックは，(230, 280) = (130 + 100, 130 + 150) で再生されます。

座標の移動の制御は，**rdp_sync_on_image** ステップの **AddOffsetToInput** パラメータで行います。このパラメータを変更し，後続の操作のために再生時の位置の差が記録座標に追加されるように，あるいは追加されないようにすることができます。このパラメータを変更しなかった場合は，実行環境の設定の標準設定から値が取得されます。

rdp_mouse_click や **rdp_mouse_drag** などの操作に対応するパラメータは **Origin** です。このパラメータによって，記録された「初期状態」の値からのみ座標を取得するか，それとも，最後の同期化ポイントによって追加された差を考慮するかが決まります。明示的に指定しなかった場合は，実行環境の設定からこのパラメータの値が取得されます。

第 21 章

RDP ― Microsoft Terminal Server エージェント

Microsoft Terminal Server エージェントは，RDP サーバに任意でインストールできるユーティリティです。このユーティリティには，スクリプトを拡張できるいくつかの重要な利点があります。

本章の内容

Microsoft Terminal Server エージェントについて

Microsoft Terminal Server エージェントは，RDP サーバに任意でインストールできるユーティリティです。このユーティリティにより，通常の RDP 機能が強化されます。このユーティリティは，製品 DVD に含まれており，任意の RDP サーバにインストールできます。

このエージェントには次の利点があります。

- 直感的かつ可読性の高いスクリプト
- 組み込み同期化
- 関連オブジェクトの詳細な情報

Microsoft Terminal Server エージェントの機能の使用

Microsoft Terminal Server エージェントにより，通常の RDP 機能が強化されます。次の項では，その機能強化について説明し，詳細とスクリプトのサンプルを示します。

オブジェクトの詳細な記録

Microsoft Terminal Server エージェントがインストールされると，VuGen は，アクションに関する一般的な情報の代わりに，使用されているオブジェクトに関する特定の情報を記録できるようになります。たとえば，VuGen は，エージェントなしで生成される**マウス・クリック・**ステップおよび**マウス・ダブル・クリック・**ステップの代わりに，**オブジェクトのマウス・クリックを同期**ステップおよび**オブジェクトのマウス・ダブルクリックを同期**ステップを生成します。

次の例は，エージェントをインストールした場合とインストールしない場合とで記録されたマウスのダブルクリック操作を示しています。エージェントを使用する場合，VuGen はすべてのマウス操作に対応する sync_object 関数を生成します。

```
rdp_sync_object_mouse_double_click("StepDescription=Mouse Double Click on
Synchronized Object 1",
        "Snapshot=snapshot_12.inf",
        "WindowTitle=RDP2",
        "Attribute=TEXT",
        "Value=button1",
        "MouseX=100",
        "MouseY=71",
        "MouseButton=LEFT_BUTTON",
        RDP_LAST);

rdp_mouse_double_click("StepDescription=Mouse Double Click 1",
        "Snapshot=snapshot_2.inf",
        "MouseX=268",
        "MouseY=592",
        "MouseButton=LEFT_BUTTON",
        "Origin=Default",
        RDP_LAST);
```

拡張された右クリック・メニュー

スナップショット内をクリックした後，右クリック・メニューを使用して，いくつかの関数をスクリプトに挿入できます。エージェントが動作していない場合は，**マウス・クリック**・ステップ，**マウス・ダブル・クリック**・ステップ，**画像による同期**ステップしか挿入できません。

エージェントがインストールされていれば，RDP エージェントが関係する起こりうるすべてのステップを挿入できます。そのステップのうちの一部に関する説明を次に示します。

- **オブジェクト情報の取得**と**オブジェクト情報と同期**：オブジェクトの状態に関する情報を提供し，特定のオブジェクト・プロパティ（ENABLED，FOCUSED，CONTROL_ID，ITEM_TEXT，TEXT，CHECKED，LINES など）で同期化します。

次の例では，rdp_sync_on_object_info 関数は［インターネット オプション］ダイアログ・ボックスにフォーカスに入るまで待機することで同期化を行います。

```
rdp_sync_on_object_info("StepDescription=Sync on Object Info 0",
        "Snapshot=snapshot_30.inf",
        "WindowTitle=Internet Options",
        "ObjectX=172",
        "ObjectY=155",
        "Attribute=FOCUSED",
        "Value={valueParam}",
        "Timeout=10",
        "FailStepIfNotFound=No",
        RDP_LAST);
```

Microsoft Terminal Server エージェントの使用に関するヒント

次の項では，Microsoft Terminal Server エージェントを使用しながら RDP スクリプトを記録する場合のヒントとベスト・プラクティスについて説明します。

- マウスを使ってアプリケーション・メニュー（［ファイル］や［編集］など）を開いた場合，同期化ステップが失敗することがあります。この問題を回避するために，記録時にはキーボードを使ってメニュー項目を選択することをお勧めします。

- **sync_on_object_mouse_click** ステップを手作業で追加した場合，座標は，画面全体に関係する絶対座標となります。同期化ポイントを作成するには，ウィンドウ内の必要なクリック位置のオフセット（相対座標）を計算し，それに応じて，同期化が正常に再生されるように絶対座標を変更する必要があります。
- 同期化オブジェクトが再生中に正しい位置と時間で存在し，別のウィンドウ（ポップアップ・ウィンドウなど）で覆われている場合，同期化ステップは成功し，同期化ポイントを覆っているウィンドウでクリックが実行されます。したがって，スクリプト・フローに支障が出ることになります。
- 再生中に AUT ウィンドウを前面に戻す場合は，タイトル・バーをクリックするか，キーボード（ALT+TAB）を使用します。AUT ウィンドウを前面に戻すためにウィンドウ内部をクリックすると，RDP セッションが途中で終了することがあります。

Microsoft Terminal Server エージェントのインストール

Microsoft Terminal Server エージェントのインストール・ファイルは，製品のインストール・ディスクの **Additional Components¥Agent for Microsoft Terminal Server** ディレクトリにあります。

RDP エージェントは，Load Generator マシンではなく，RDP サーバ • マシンにのみインストールされなければなりません。

エージェントをアップグレードする場合は，新しいバージョンをインストールする前に，前のバージョンをアンインストールしてください。

Microsoft Terminal Server エージェントをインストールするには，次の手順を実行します。

1 サーバへのソフトウェアのインストールに管理者権限が必要な場合は，サーバに管理者としてログインします。

2 リモート・デスクトップ接続（RDP）を使用して，Windows 2003 が実行されているマシンにエージェントをインストールする場合は，インストールを開始する前に，対象マシンで次のコマンドを実行します。

```
Change user /install
```

3 LoadRunner DVD の **Additional Components¥Agent for Microsoft Terminal Server** ディレクトリで，インストール・ファイル **Setup.exe** を探します。

4 インストール・ウィザードに従ってインストールを完了します。

注：エージェントを使用するには，Vuser スクリプトを記録する前に記録オプションを設定しておく必要があります。［記録開始］ダイアログ・ボックスの［オプション］をクリックします。［Code Generation-Agent］ノードの［**Use RDP Agent**］にチェック・マークを付けます。

Microsoft Terminal Server エージェントをアンインストールするには，次の手順を実行します。

1 サーバからのソフトウェアの削除に管理者権限が必要な場合は，サーバに管理者としてログインします。

2 サーバ・マシンのコントロール・パネルから［**プログラムの追加と削除**］を開きます。「**HP Software Agent for Microsoft Terminal Server**」を選択し，［**変更と削除**］をクリックします。

効果と記憶容量

エージェントのインストールされた RDP Vuser を実行すると，Vuser は **lrrdpagent.exe** の独自の手順を実行します。その結果，サーバ・マシンで実行できる Vuser の数がわずかに減少します。

第 22 章

データベース・プロトコル

VuGen を使用して，データベース・クライアント・アプリケーションとサーバの間の通信を記録することができます。記録の結果として作成されるスクリプトを，「データベース Vuser スクリプト」と呼びます。

本章の内容

データベース Vuser スクリプトの作成について

サーバと通信を行っているデータベース・アプリケーションを記録すると，VuGen によってデータベース Vuser スクリプトが生成されます。VuGen では，次のデータベース・タイプがサポートされています。
CtLib，DbLib，Informix，Oracle，ODBC，DB2-CLI。
生成されたスクリプトには，データベース操作を表す LRD 関数が含まれています。LRD 関数の名前には **lrd** というプレフィックスが付いており，1 つまたは複数のデータベース機能に対応します。たとえば，**lrd_fetch** 関数はフェッチ操作を表します。

記録されたセッションを実行すると，Vuser スクリプトとデータベース・サーバとの間で直接通信が行われ，実際のユーザと同じ操作が実行されます。Vuser の動作を設定して（実行環境の設定），操作の反復回数と反復間隔を指定できます。詳細については，『**第 1 巻 － VuGen の使用**』の「実行環境の設定」を参照してください。

VuGen を使って，記録された定数をパラメータで置き換えることによって，スクリプトをパラメータ化できます。詳細については，『**第 1 巻 － VuGen の使用**』の「パラメータの作成」を参照してください。

さらに，スクリプトのクエリやほかのデータベース・ステートメントを相関させて，特定のクエリの結果を別のクエリに結び付けることができます。詳細については，『**第 1 巻 － VuGen の使用**』の「ステートメントの相関」を参照してください。

トラブルシューティング情報とスクリプト作成のヒントについては，デバッグに関するヒントを参照してください。

データベース Vuser の紹介

全国のカスタマ・サービス担当者がアクセスする顧客情報のデータベースがあるとします。この場合には，データベース Vuser を使って，データベース・サーバが多数の情報の問い合わせに対応するという状況をエミュレートします。データベース Vuser では，次のことが可能です。

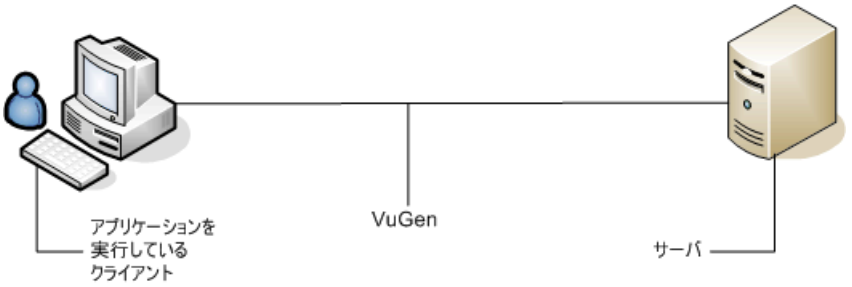
- サーバへの接続
- SQL クエリの発行
- 情報の検索と処理
- サーバからの切断

まず，利用可能な Load Generator に数百の DB Vuser を振り分けます。各 Vuser はサーバ API を使ってデータベースにアクセスします。これにより，多数のユーザによる負荷をかけた状態でのサーバのパフォーマンスを測定できます。

サーバ API への呼び出しが含まれるプログラムをデータベース（DB）Vuser スクリプトといいます。データベース Vuser スクリプトによって，クライアント・アプリケーションと，そのすべての操作がエミュレートされます。Vuser によってスクリプトが実行されると，クライアント / サーバ・システムにユーザ負荷をかけた状態がエミュレートされます。Vuser によって生成されるパフォーマンス・データは，レポートやグラフを使って分析できます。

データベース Vuser 技術について

VuGen では，データベース・クライアントとサーバの間のやり取りをすべて記録することによって，データベース Vuser スクリプトを作成します。VuGen で，データベースのクライアント側を監視し，データベース・サーバとの間で送受信されるすべての要求を追跡します。



VuGen を使って作成するほかのすべての Vuser と同様に，データベース Vuser も，サーバと通信を行うときにクライアント・ソフトウェアに依存しません。その代わりに，各データベース Vuser ではサーバ API 関数を直接呼び出すスクリプトが実行されます。

データベース Vuser スクリプトは，Windows 環境で VuGen を使って作成します。しかし，作成したスクリプトは，Windows および UNIX のどちらの環境でも Vuser に割り当てることができます。スクリプトの記録の詳細については，『**第 1 巻 ー VuGen の使用**』の「VuGen を使った記録」を参照してください。

UNIX 環境では，VuGen のテンプレートを土台にしてスクリプトのプログラミングを行うことにより，DB Vuser スクリプトを作成できます。UNIX での DB Vuser スクリプトのプログラミングの詳細については，『**第 1 巻 ー VuGen の使用**』の付録 31「UNIX プラットフォームでのスクリプトのプログラミング」を参照してください。

データベース Vuser スクリプトの概要

本項では，VuGen を使ったデータベース Vuser スクリプトの作成プロセスの概要を説明します。

データベース Vuser スクリプトを作成するには，次の手順を実行します。

1 VuGen を使用して基本スクリプトを記録します。

VuGen を起動して，新しい Vuser スクリプトを作成します。Vuser のタイプ（「**クライアント / サーバ**」または「**ERP/CRM**」プロトコル・タイプ）を指定します。記録対象のアプリケーションを選び，記録オプションを設定します。選択したアプリケーションの典型的な操作を記録します。

詳細については，『**第 1 巻 － VuGen の使用**』の「VuGen を使った記録」を参照してください。

2 スクリプトを拡張します。

トランザクション，ランデブー・ポイント，および制御フロー構造をスクリプトに挿入することによって，Vuser スクリプトを拡張します。

詳細については，『**第 1 巻 － VuGen の使用**』の「Vuser スクリプトの拡張」を参照してください。

3 パラメータを定義します（任意）。

スクリプトに記録された固定値の代わりにパラメータを定義します。固定値をパラメータで置き換えることで，同じクエリを異なる値を使って反復実行できます。

詳細については，『**第 1 巻 － VuGen の使用**』の「パラメータの作成」を参照してください。

4 クエリを相関させます（任意）。

データベース・ステートメントを相関させることによって，クエリの結果を以降のほかのクエリで使用できるようになります。この機能は，ユーザに制約が課せられるデータベースで作業をするときに有用です。

詳細については，『**第 1 巻 － VuGen の使用**』の「ステートメントの相関」を参照してください。

5 実行環境を設定します。

実行環境を設定することによって，スクリプト実行時の Vuser スクリプトの振る舞いを制御します。設定には，ループ，ログ，タイミング情報などがあります。

詳細については，『**第 1 巻 ー VuGen の使用**』の「実行環境の設定」を参照してください。

6 **VuGen でスクリプトを実行します。**

VuGen でスクリプトを保存して実行し，正しく動作することを確認します。

詳細については，『**第 1 巻 ー VuGen の使用**』の「スタンドアロン・モードでの Vuser スクリプトの実行」を参照してください。

スクリプトを作成した後，スクリプトを自分の環境（LoadRunner シナリオ，Performance Center 負荷テスト，または Business Process Monitor プロファイル）に組み込みます。詳細については，**HP LoadRunner Controller**，**HP Performance Center**，または **HP Business Availibility Center** のドキュメントを参照してください。

LRD 関数の使用法

データベース・クライアントとサーバの間の通信をエミュレートするために作成された関数を LRD Vuser 関数といいます。LRD Vuser 関数には，**lrd** というプレフィックスが付きます。VuGen では，データベース・セッション（CtLib，DbLib，Informix，Oracle（2-Tier），および ODBC）の間に，この項で示すほとんどの LRD 関数を自動的に記録します。また，手作業でスクリプトに任意の関数をプログラミングすることもできます。

LRD 関数は，いくつかのカテゴリ（アクセス管理関数，環境関数，検索・取得処理，ステートメント処理，ステートメント相関，変数処理，Siebel，Oracle 8）に分類されます。

LRD 関数の構文と使用例については，『**Online Function Reference**』（英語版）（［**ヘルプ**］＞［**関数リファレンス**］）を参照してください。

データベース Vuser スクリプトについて

データベース・セッションを記録した後，VuGen に組み込まれているエディタを使って，記録されたコードを表示できます。スクリプトをスクロールして，アプリケーションによって生成された SQL ステートメントを確認したり，サーバから返されたデータを調べたりできます。

VuGen ウィンドウには，記録されたデータベース・セッションに関する次の情報が表示されます。

- 記録された関数の並び
- データベース・クエリによって返されたデータを表示するグリッド
- クエリ中に取り出された行の数

関数の並び

VuGen ウィンドウに Vuser スクリプトを表示すると，VuGen によって記録した操作のシーケンスを見ることができます。たとえば，標準的な Oracle データベース・セッションでは，次のような関数の並びが記録されます。

lrd_init	環境を初期化します。
lrd_open_connection	データベース・サーバに接続します。
lrd_open_cursor	データベース・カーソルを開きます。
lrd_stmt	SQL ステートメントとカーソルを関連付けます。
lrd_bind_col	ホスト変数をカラムにバインドします。
lrd_exec	SQL ステートメントを実行します。
lrd_fetch	結果セットから次の記録を取り出します。
lr_commit	データベース・トランザクションをコミットします。
lr_close_cursor	カーソルを閉じます。
lrd_close_connection	データベース・サーバから接続を解除します。
lrd_end	環境をクリーンアップします。

次に示すスクリプトは，Oracle サーバへの接続を開いて，ローカル設定を要求するクエリを実行したオペレータのアクションを VuGen で記録したものです。

```
lrd_init(&InitInfo, DBTypeVersion);
lrd_open_connection(&Con1, LRD_DBTYPE_ORACLE, "s1", "tiger", "hp1", "", 0, 0, 0);
lrd_open_cursor(&Csr1, Con1, 0);
lrd_stmt(Csr1, "select parameter, value    from v$nls_parameters "
     "    where (upper(parameter) in ('NLS_SORT','NLS_CURRENCY',"
     "'NLS_ISO_CURRENCY', 'NLS_DATE_LANGUAGE',"
     "'NLS_TERRITORY'))", -1, 0 /*Non deferred*/, 1 /*Dflt Ora Ver*/, 0);
lrd_bind_col(Csr1, 1, &D1, 0, 0);
lrd_bind_col(Csr1, 2, &D2, 0, 0);
lrd_exec(Csr1, 0, 0, 0, 0, 0);
lrd_fetch(Csr1, 7, 7, 0, PrintRow2, 0);
. . . . . lrd_close_cursor(&Csr1, 0);
lrd_commit(0, Con1, 0);
lrd_close_connection(&Con1, 0, 0);
lrd_end(0);
```

行情報

VuGen によって，SQL クエリごとに lrd_fetch 関数が生成されます。

```
lrd_fetch(Csr1, -4, 1, 0, PrintRow7, 0);
```

関数の 2 番目のパラメータは，取り出される行の数を示します。正数，負数のどちらも指定できます。

正数の値

正数の値は記録中に取り出された行の数を示し，すべての行が取り出されていないことを示します （オペレータがクエリ完了前にクエリを取り消した場合など)。

次の例では，データベース・クエリの実行時に 4 行を取り出していますが，すべてのデータは取り出していません。

```
lrd_fetch(Csr1, 4, 1, 0, PrintRow7, 0);
```

実行時には，正数によって示される数の行がスクリプトによって取り出されます（行が存在することが前提)。

負数の値

行の値が負数の場合，記録時にすべての行が取り出されたことを示します。負数の絶対値が，取り出された行の数です。

次の例では，結果セットの 4 行すべてを取り出しています。

```
lrd_fetch(Csr1, -4, 1, 0, PrintRow7, 0);
```

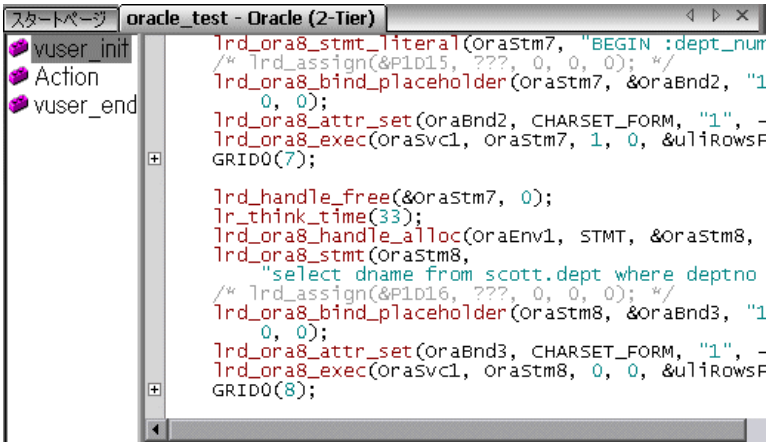
負数の値を含む **lrd_fetch** ステートメントを実行すると，実行時にテーブルから取り出せる行がすべて取り出されます。必ずしも記録時の行数と同じではありません。上の例では，テーブルの 4 行すべてが記録時に取り出されています。しかし，スクリプト実行時に，それ以上の行数がある場合は，それらがすべて取り出されます。

lrd_fetch の詳細については，『**Online Function Reference**』（英語版）（［**ヘルプ**］＞［**関数リファレンス**］）を参照してください。

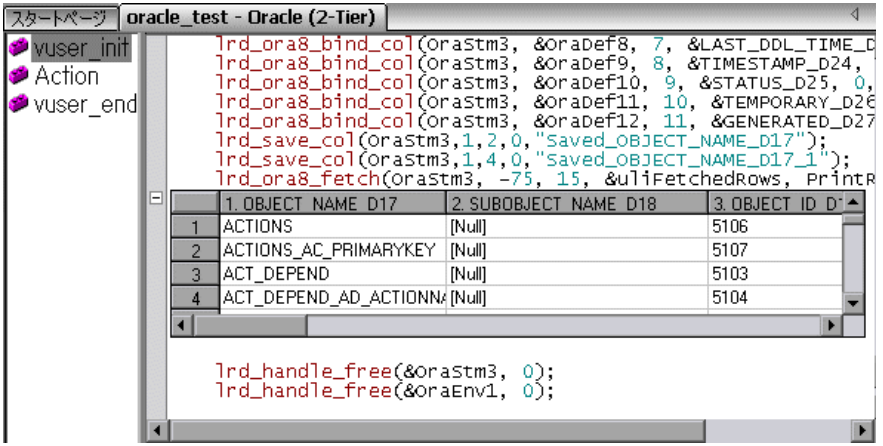
グリッドを使った作業

記録セッション中にクエリから返されたデータはグリッドに表示されます。グリッドを参照することで，アプリケーションによって SQL ステートメントがどのように生成されたかを確認したり，クライアント / サーバ・システムの効率を把握したりできます。

データ・グリッドは **GRID** ステートメントで表されます。データ・グリッドを開くには，GRID ステートメントの横の余白にあるアイコンをクリックします。



次の例では，ウィンドウにフライト予約データベースを対象に実行されたクエリが表示されています。クエリでは，フライト番号，空港名コード，出発地，曜日，およびその他のフライト関連情報が取得されています。



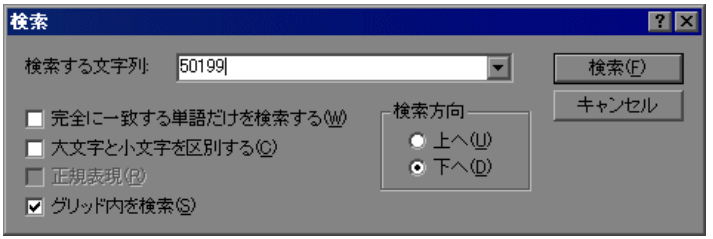
データ値が非常に長い場合は，その一部のみグリッドに表示されます。この切り捨ては，表示されているグリッドでのみ生じ，データには影響を与えません。

ウィンドウの表示枠は，幅を調整することが可能です。また，スクロール・バーを使って，最高 200 行までスクロールできます。この値を変更するには，マシンのオペレーティング・システム・フォルダ（C:¥WINNT など）にある **vugen.ini** ファイルを開いて次のエントリを変更します。

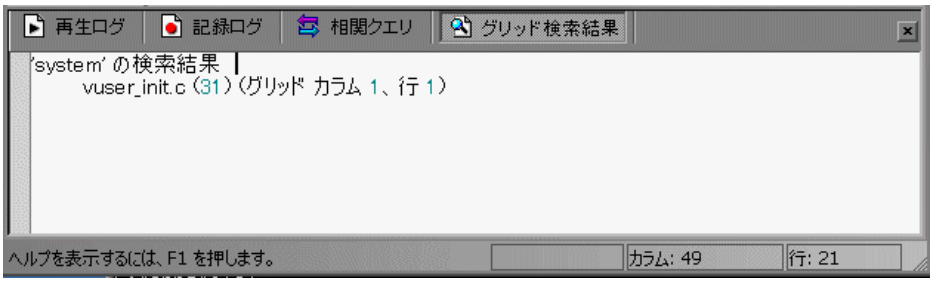
```
[general]
max_line_at_grid=200
```

値を相関させる，あるいはデータをファイルに保存するには，セルをクリックし，右クリック・メニューの［**相関を作成**］または［**ファイルに保存**］を選択します。

グリッド全体でデータ（見えない部分も含む）を検索するには，［検索］ダイアログ・ボックスの［**グリッド内を検索**］を選択します。



出力ウィンドウの［**グリッド検索結果**］タブに結果が表示されます。



データベース・プロトコルのトラブルシューティング

次の項では，データベース・プロトコルのトラブルシューティングについて説明します。

Oracle NCA/11i スクリプトの記録時に IE がクラッシュした

これは，互換性のない dll ファイルが原因で発生することがあります。

互換性のない dll を置き換えるには，次の手順を実行します。

1 **C:¥program files¥oracle¥JInitiator_1.3.1.18¥bin¥hotspot** ディレクトリを開きます。

2 **jvm.dll** ファイルをバックアップします。

3 **jvm.dll** ファイルを削除し，違うバージョンのファイルで置き換えます。

エラー・コードの分析

Vuser が LRD 関数を実行すると，関数によってリターン・コードが生成されます。リターン・コード「0」は，関数が成功したことを示します。たとえば，リターン・コード「0」は，結果セットに利用可能な行が残っていることを示します。エラーが発生した場合，リターン・コードはエラーの種類を表します。たとえば，リターン・コード「2014」は，初期化中にエラーが発生したことを表します。

リターン・コードは 4 種類に分類され，それぞれ数値の範囲が決まっています。

リターン・コードの種類	範囲
情報	0 ～ 999
警告	1000 ～ 1999
エラー	2000 ～ 2999
内部エラー	5000 ～ 5999

リターン・コードの詳細については，『**Online Function Reference**』（英語版）（［**ヘルプ**］ > ［**関数リファレンス**］）を参照してください。

LRD 関数のリターン・コードを調べて，関数が成功したかどうかを確認できます。以下のスクリプトでは，lrd_fetch 関数のリターン・コードを評価しています。

```
static int rc;
rc=lrd_fetch(Csr15, -13, 0, 0, PrintRow4, 0);
if (rc==0)
    lr_output_message("The function succeeded");
else
    lr_output_message("The function returned an error code:%d",rc);
```

エラー処理

データベース Vuser スクリプトの実行時に，データベース Vuser にエラーをどのように処理させるかを制御できます。標準設定では，スクリプト実行時にエラーが発生すると，スクリプトの実行が終了します。Vuser の標準の振る舞いを変更するには，エラーが発生しても処理を継続するように設定します。この振る舞いは次の方法で適用できます。

- **エラー処理のグローバル変更**：スクリプト全体またはスクリプトのセグメントにエラー処理を適用します。
- **エラー処理のローカル変更**：特定の関数にのみエラー処理を適用します。

エラー処理のグローバル変更

LRD_ON_ERROR_CONTINUE または LRD_ON_ERROR_EXIT ステートメントを発行することによって，Vuser のエラー処理の方法を変更できます。標準設定では，データベース関連であれパラメータ関連であれ，エラーが生じると Vuser によるスクリプトの実行が中止されます。この標準設定の振る舞いを変更するには，スクリプトに次の行を挿入します。

```
LRD_ON_ERROR_CONTINUE;
```

以降，Vuser でエラーが発生してもスクリプトの実行が継続されます。

また，スクリプトの特定のセグメントだけでエラーが発生する場合に，Vuser にスクリプトの実行を継続するように指定することもできます。たとえば，以下のコードは，lrd_stmt や lrd_exec の関数内でエラーが発生した場合は，スクリプトの実行を継続するように Vuser に指示します。

```
LRD_ON_ERROR_CONTINUE;
lrd_stmt(Csr1, "select..."...);
lrd_exec(...);
LRD_ON_ERROR_EXIT;
```

LRD_ON_ERROR_CONTINUE ステートメントを使うときは，重要かつ重大なエラーを見逃してしまう可能性があるので注意が必要です。

エラー処理のローカル変更

選択した関数の重要度を変更してエラー処理を設定できます。**lrd_stmt** や **lrd_exec** といったデータベース処理を実行する関数は，重要度を使用します。重要度は関数の最後のパラメータである **miDBErrorSeverity** で示します。このパラメータは，データベース・エラー（エラー・コード 2009）が発生した場合に，Vuser にスクリプトの実行を継続させるかどうかを指示するものです。標準設定の「0」は，エラー発生時に Vuser によるスクリプトの実行を中止することを示します。

たとえば，存在しないテーブルを参照した場合，次のデータベース・ステートメントが失敗し，スクリプトの実行が終了します。

```
lrd_stmt(Csr1, "insert into EMP values ('Smith',301)¥n", -1, 1 /*Deferred*/,
        1 /*Dflt Ora Ver*/, 0);
```

データベース処理でエラーが発生した場合でも，Vuser にスクリプトの実行を継続するように指示するには，ステートメントの重要度パラメータを「0」から「1」に変更します。

```
lrd_stmt(Csr1, "insert into EMP values ('Smith',301)¥n", -1, 1 /*Deferred*/,
        1 /*Dflt Ora Ver*/, 1);
```

重要度を「1」に設定した場合，データベース・エラーが発生すると，警告が表示されます。特定の関数に重要度レベルを設定すると，その重要度レベルはその関数にだけ適用されることに注意してください。

CtLib 結果セット・エラー

CtLib の記録時には，アプリケーションによってステートメントが実行された後，利用可能な結果セットがすべて取り出されます。返された結果セットに取り出し可能なデータが含まれている場合は，アプリケーションでそのデータを対象にバインドおよび取り出しの操作が行われます。次に例を示します。

```
lrd_stmt(Csr15, "select * from all_types", -1, 148, -99999, 0);
lrd_exec(Csr15, 0, 0, 0, 0, 0);
lrd_result_set(Csr15, 1 /*Succeed*/, 4040 /*Row*/, 0);
lrd_bind_col(Csr15, 1, &tinyint_D41, 0, 0);
...
lrd_fetch(Csr15, -9, 0, 0, PrintRow3, 0);
```

結果セットに取り出し可能なデータが含まれていない場合，バインドと取り出しの操作は行えません。

スクリプトをパラメータ化すると，パラメータによっては結果データが取り出せなくなることがあります。新しいデータを取り出せない場合，特定のステートメントに対するバインドと取り出しの操作を記録した CtLib セッションは，実行できないことがあります。**lrd_bind_col** 関数または **lrd_fetch** 関数を実行しようとすると，エラーが発生し（LRDRET_E_NO_FETCHABLE_DATA：エラー・コード 2064），Vuser によるスクリプトの実行が終了します。

このタイプのエラーが発生したときに，Vuser にスクリプト実行の継続を指示することにより，実行を優先させることができます。次の行をスクリプトに挿入します。

```
LRD_ON_FETCHABLE_SET_ERR_CONT;
```

エラーの発生時にスクリプトの実行が終了する標準設定のモードに戻すには，次の行をスクリプトに入力します。

```
LRD_ON_FETCHABLE_SET_ERR_EXIT;
```

このステートメントを使うときは，重要かつ重大なエラーを見逃してしまう可能性があるので注意が必要です。

第 23 章

データベース ― スクリプトの相関

データベース・セッションの記録後に，スクリプト内の 1 つ以上のクエリを相関させる必要が生じることがあります。これによって，データベース・セッション中に取得された値を，セッション内の以降の処理で使用できるようになります。

本章の内容

データベース Vuser スクリプトの相関について

スクリプト実行時にエラーが発生した場合は，スクリプトの中でエラーが発生した場所を調べます。多くの場合，クエリを相関させることによって問題を解決できます。クエリの相関とは，実行時の値をパラメータに保存することです。保存した値は，同じスクリプト内の以降の場所で使用します。つまり，相関とは，あるステートメントの結果を別のステートメントへの入力として使用することです。

多くのクエリは，その入力が前のクエリの結果に依存します。この振る舞いをエミュレートするには，VuGen の相関機能を使用します。

スクリプトでの相関候補の検索

VuGen には，スクリプトを修正して確実に再生できるようにするための相関ユーティリティがあります。相関ユーティリティは，次の処理を行います。

- 相関候補を検索する。
- 適切な相関関数を挿入して，結果をパラメータに保存する。
- ステートメントの値をパラメータで置き換える。

自動相関は，スクリプト全体またはスクリプト内の特定の場所を対象に実行できます。

本項では，相関させる必要のあるステートメントを見つける方法について説明します。すでに相関させたい値がわかっている場合は，次の項に進んで，特定の値を相関させる方法を参照してください。

自動相関で検出されたスクリプトの検索と相関を行うには，次の手順を実行します。

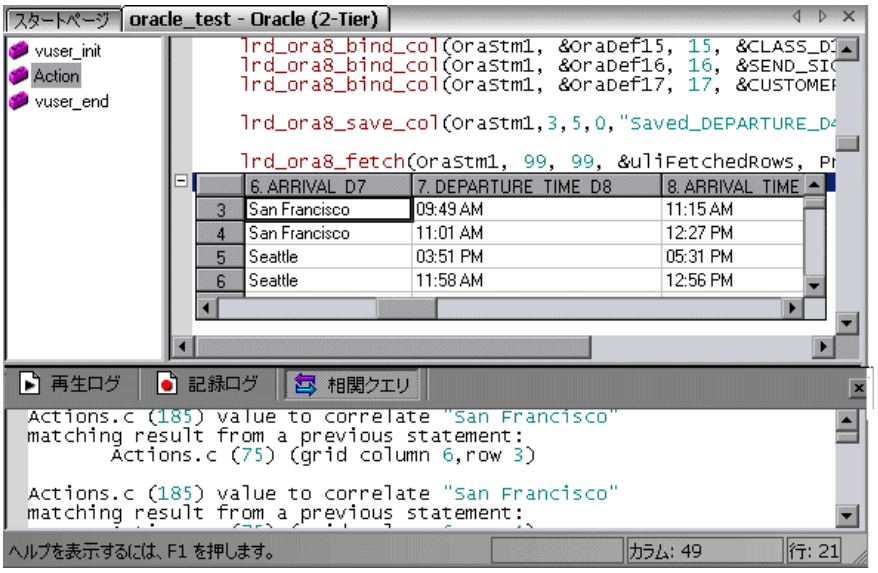
1 ［出力］ウィンドウを開きます。

［**表示**］＞［**出力ウィンドウ**］を選択して，ウィンドウの下部に出力タブを表示します。［再生ログ］タブでエラーがないか確認します。多くの場合，これらのエラーは相関によって修正できます。

2 ［**仮想ユーザ**］＞［**相関を検索**］を選択します。

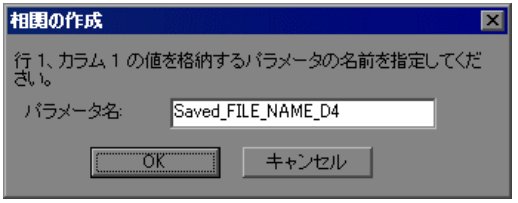
VuGen によってスクリプト全体を対象とする検索が行われ，相関候補の値がすべて［**相関クエリ**］タブに表示されます。

次の例では，**lrd_ora8_fetch** 関数の中で，相関させる必要のある値が検出されています。



3 ［相関クエリ］タブで，相関させるクエリ結果をダブルクリックします。「**(grid column x, row y)**」という語をクリックします。グリッド内の値の位置にカーソルが移動します。

4 右クリック・メニューから［**相関を作成**］を選択します。結果の値を保存するパラメータの名前を入力するように求められます。



5 名前を指定するか，標準設定の名前をそのまま使用します。［**OK**］をクリックして作業を続けます。VuGen によって，パラメータに結果を保存する適切な相関ステートメント（**lrd_save_value**，**lrd_save_col**，**lrd_save_ret_param**，**lrd_ora8_save_col** のいずれか）が挿入されます。

6［**はい**］をクリックして相関を確定します。

スクリプト内に現れる値をすべて検索するかどうかを尋ねるメッセージ・ボックスが開きます。

- 選択したステートメント内の値だけを置き換える場合は，［**いいえ**］をクリックします。
- 次の候補を検索して置き換える場合は［**はい**］をクリックします。

7［検索と置換］ダイアログ・ボックスが開きます。オリジナルのステートメントも含め，すべての置き換えを確認します。

8［検索と置換］ダイアログ・ボックスを閉じます。ステートメントの値がパラメータへの参照に置き換えられます。相関を取り消すと，直前のステップで作成されたステートメントは消去されます。

既知の値の相関

相関させる必要がある値がわかっている場合は，次の手続きを行います。

特定の値を相関させるには，次の手順を実行します。

1 相関させる値を含むクエリを使って，スクリプト内のステートメントを検索します。これは，通常 **lrd_assign**，**lrd_assign_bind**，**lrd_stmt** のいずれかの関数の引数です。引用符を含めずに値を選択します。

2 右クリック・メニューから［**相関を検索（カーソル位置）**］を選択します。選択した値を対象に相関が検索されます。

3 出力ウィンドウの［**相関クエリ**］タブで，相関させる結果をダブルクリックします。「**(grid column x, row y)**」という語をクリックします。グリッド内の値の位置にカーソルが移動します。

4 グリッド内で，相関させる値をクリックし，右クリック・メニューから［**相関を作成**］を選択します。結果の値を保存するパラメータの名前を入力するように求められます。

5 名前を指定するか，標準設定の名前をそのまま使用します。［**OK**］をクリックして作業を続けます。VuGen によって，パラメータに結果を保存する適切な相関ステートメント（**lrd_save_value**，**lrd_save_col**，**lrd_save_ret_param**，**lrd_ora8_save_col** のいずれか）が挿入されます。

6 ［**はい**］をクリックして相関を確定します。

スクリプト内に現れる値をすべて検索するかどうかを尋ねるメッセージ・ボックスが開きます。

- 選択したステートメント内の値だけを置き換える場合は，［**いいえ**］をクリックします。
- 次の候補を検索して置き換える場合は［**はい**］をクリックします。

7 ［検索と置換］ダイアログ・ボックスが開きます。オリジナルのステートメントも含め，すべての置き換えを確認します。

8 ［検索と置換］ダイアログ・ボックスを閉じます。ステートメントの値がパラメータへの参照に置き換えられます。相関を取り消すと，直前のステップで作成されたステートメントは消去されます。

注：lrd_stmt 関数の値を相関させている場合は，データ型 date，time，binary（RAW，VARRAW）はサポートされません。

データベース Vuser 相関関数

データベース Vuser スクリプト（DbLib，CtLib，Oracle，Informix など）を使用するときに，VuGen の自動相関機能を使用して，スクリプトに適切な関数を挿入できます。相関関数は次のとおりです。

- **lrd_save_col** 関数では，表示枠内に表示されるクエリ結果がパラメータに保存されます。この関数はデータの取り出しの前に配置されます。lrd_save_col 関数によって，それ以降に **lrd_fetch** によって取り出された値が指定されたパラメータに代入されます （Oracle 8 以降は **lrd_ora8_save_col**）。
- **lrd_save_value** 関数では，プレースホルダ記述子の現在値がパラメータに保存されます。出力プレースホルダを設定するデータベース関数（Oracle の特定のストアド・プロシージャなど）と一緒に使用します。
- **lrd_save_ret_param** 関数では，ストアド・プロシージャの戻り値がパラメータに保存されます。この関数は主に，戻り値を生成する DbLib 内のデータベース・プロシージャと組み合わせて使用します。

注：VuGen では，保存された値が無効または NULL（行が返されない）の場合，相関が適用されません。

これらの関数と引数の詳細については，『**Online Function Reference**』（英語版）（［**ヘルプ**］ ＞ ［**関数リファレンス**］）を参照してください。

第 24 章

DNS プロトコル

VuGen では，DNS サーバに直接アクセスしてネットワークの動作状態をエミュレートできます。

本章の内容

DNS Vuser スクリプトの作成について

DNS プロトコルは，低レベルのプロトコルで，DNS サーバに対して行うユーザのアクションをエミュレートします。

DNS プロトコルは，Domain Name Server にアクセスし，IP アドレスでホスト名を解決するユーザをエミュレートします。このプロトコルでは，再生機能のみサポートされているため，手作業でスクリプトに関数を追加します。

DNS プロトコルのスクリプトを作成するには，［**ファイル**］＞［**新規作成**］を選択して［新規 Virtual ユーザ］ダイアログ・ボックスを開きます。［クライアント / サーバ］カテゴリから［Domain Name Resolution（DNS）］を選択します。DNS プロトコルについてはスクリプトの記録ができないため，適切な DNS 関数，Vuser API 関数，C 関数を使ってスクリプトをプログラミングします。これらの関数の詳細については，『**Online Function Reference**』（英語版）（［**ヘルプ**］＞［**関数リファレンス**］）を参照してください。

Vuser スクリプトを作成したら，Windows または UNIX プラットフォームのいずれかでシナリオに組み込みます。Vuser スクリプトをシナリオに組み込む方法の詳細については，『**HP LoadRunner Controller ユーザーズ・ガイド**』を参照してください。

DNS 関数を使った作業

DNS Vuser スクリプト関数ではドメイン名解決サーバ（DNS）を往復するクエリが記録されます。DNS 関数はどれもプレフィックス **dns** で始まります。これらの関数の構文情報の詳細については，『**Online Function Reference**』（英語版）（［**ヘルプ**］＞［**関数リファレンス**］）を参照してください。

関数名	説明
ms_dns_query	ホストの IP アドレス解決をします。
ms_dns_nextresult	**ms_dns_query** 関数によって返される IP アドレス・リスト内の次の IP アドレスに進みます。

次の例では，クエリを DNS サーバに送信し，その結果をログ・ファイルに出力しています。

```
Actions()
{
int   rescnt = 0;
    char results = NULL;
    results = (char *) ms_dns_query("transaction",
                                    "URL=dns:// ＜ Dns サーバ＞ ",
                                    "QueryHost= ＜ホスト名＞ ",
                                    LAST);

    // ホスト名の全 IP アドレスを表示 ..
    while (*results) {
        rescnt++;
        lr_log_message(lr_eval_string("(%d) IP of ＜ Hostname ＞ is %s"),
            rescnt, results);
        results = (char *) ms_dns_nextresult(results);
    }
    return 1;
}
```

第 25 章

Windows Sockets（WinSock）プロトコル

VuGen を使って，Windows Sockets プロトコルでやり取りするクライアント・アプリケーションとサーバ間の通信を記録することができます。その結果生成されるスクリプトを，Windows Sockets Vuser スクリプトと呼びます。

本章の内容

Windows Sockets Vuser スクリプトの記録について

Windows Sockets プロトコルは，アプリケーションの低レベルのコードを分析するのに適したプロトコルです。たとえば，ネットワークの検査を行うために，Windows Sockets（WinSock）スクリプトを使って，バッファによって送受信される実際のデータを見ることができます。また，WinSock プロトコルはほかの低レベルの通信セッションを記録するためにも使用できます。さらに，ほかのタイプの Vuser でサポートされていないアプリケーションの記録と再生もできます。

Windows Sockets プロトコルを使用するアプリケーションを記録すると，記録されたアクションを表す関数が生成されます。各関数には，**lrs** というプレフィックスが付きます。LRS 関数は，ソケット，データ・バッファ，および Windows Sockets 環境に対応します。VuGen を使って，アプリケーションの Winsock.dll または Wsock32.dll に対する API 呼び出しを記録します。

たとえば，**telnet** アプリケーションのアクションを記録して，スクリプトを作成できます。

次に示す例では，lrs_send を使って指定したソケットにデータを送信しています。

```
lrs_send("socket22", "buf44", LrsLastArg);
```

記録されたスクリプトは，VuGen のメイン・ウィンドウで表示および編集ができます。このウィンドウには，セッション中に記録された Windows Sockets API 呼び出しが表示されるので，記録時のネットワークの動作状況を追うことができます。

VuGen では，WinSock スクリプトを次の 2 つの方法で表示できます。

- スクリプトをアイコン形式で表示します。これは標準設定のビューで，「**ツリー・ビュー**」といいます。
- スクリプトをテキスト形式で表示して Windows Sockets API 呼び出しを示します。これは，「**スクリプト・ビュー**」と呼びます。

VuGen では，スクリプトをツリー・ビューとスクリプト・ビューの両方で表示して編集できます。詳細については，『**第 1 巻 － VuGen の使用**』の「VuGen の紹介」を参照してください。

スクリプトを作成したら，記録したデータをスナップショットか未処理のデータ・ファイルで表示できます。詳細については，393 ページ「Windows Sockets データを使った作業」を参照してください。

Windows Sockets Vuser スクリプトの開発の概要

本項では，VuGen を使って Windows Sockets Vuser スクリプトを開発する工程の概略を説明します。

Windows Sockets スクリプトを開発するには，次の手順を実行します。

1 VuGen を使ってアクションを記録します。

VuGen を起動し，Windows Sockets タイプを指定して，新しい Vuser スクリプトを作成します。記録対象のアプリケーションを選び，記録オプションを設定します。選択したアプリケーションの典型的な操作を記録します。

詳細については，『**第 1 巻 － VuGen の使用**』の「VuGen を使った記録」を参照してください。

2 Vuser スクリプトを拡張します。

トランザクション，ランデブー・ポイント，および制御フロー構造をスクリプトに挿入することによって，Vuser スクリプトを拡張します。

詳細については，『**第 1 巻 － VuGen の使用**』の「Vuser スクリプトの拡張」を参照してください。

3 パラメータを定義します（任意）。

Vuser スクリプトに記録された固定値に対してパラメータを定義します。固定値をパラメータで置き換えることにより，同じビジネス・プロセスを，異なる値で何度も繰り返すことができます。

詳細については，『**第 1 巻 － VuGen の使用**』の「パラメータの作成」を参照してください。

4 ステートメントを相関させます（任意）。

ステートメントの相関によって，あるビジネス・プロセスの結果を以降の別のビジネス・プロセスで使用できるようになります。

詳細については，『**第 1 巻 － VuGen の使用**』の「ステートメントの相関」を参照してください。

5 実行環境を設定します。

実行環境を設定することによって，スクリプト実行時の Vuser の動作を制御します。設定には，ループ，ログ，タイミング情報などがあります。

詳細については，「実行環境の設定」と『**第 1 巻 － VuGen の使用**』の「ネットワーク実行環境の設定」を参照してください。

6 VuGen から Vuser スクリプトを実行します。

VuGen で生成した Vuser スクリプトを保存して実行し，正しく動作することを確認します。

詳細については，『**第 1 巻 － VuGen の使用**』の「スタンドアロン・モードでの Vuser スクリプトの実行」を参照してください。

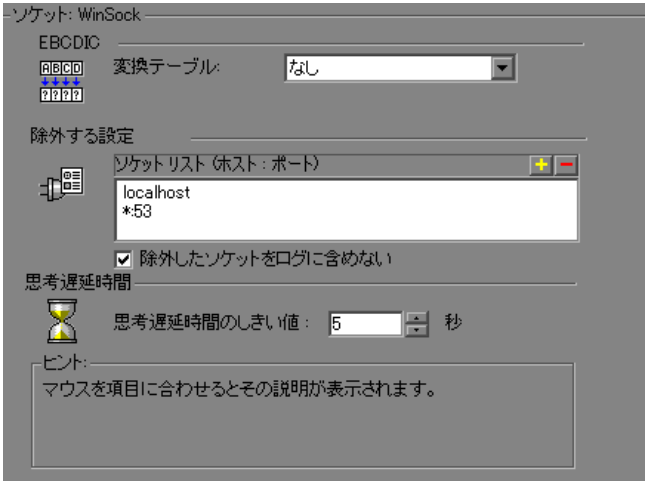
スクリプトを作成した後，スクリプトを自分の環境（LoadRunner シナリオ，Performance Center 負荷テスト，または Business Process Monitor プロファイル）に組み込みます。詳細については，**HP LoadRunner Controller**，**HP Performance Center**，または **HP Business Availibility Center** のドキュメントを参照してください。

WinSock 記録オプションの設定

WinSock Vuser に対しては，以下の記録オプションを使用できます。

- 変換テーブルの設定
- ソケットの除外
- 思考遅延時間のしきい値の設定

［記録オプション］ダイアログ・ボックスを開くには，［**ツール**］＞［**記録オプション**］を選択するか，［記録開始］ダイアログ・ボックスで［**オプション**］ボタンをクリックします。WinSock 用のオプションが表示されます。



変換テーブルの設定

EBCDIC 形式のデータを表示するには，［記録オプション］で変換テーブルを指定します。

［変換テーブル］で記録セッションの形式を指定できます。この設定は，メインフレーム・マシンや AS/400 サーバで実行しているユーザに適用されます。サーバ・マシンおよびクライアント・マシンは，システムにインストールされている変換テーブルに基づいて，データの形式を判断します。リスト・ボックスから変換オプションを選択します。

リスト・ボックスの項目の最初の 4 桁はサーバの形式を表します。後半の 4 桁はクライアントの形式を表します。上の例で選択した変換テーブルは 002501b5 です。サーバの形式が 0025，クライアントの形式が 01b5 で，サーバからクライアントへの送信を表しています。クライアントからサーバへの送信の場合，形式を逆転した項目「01b50025」を選択することで，クライアントの 01b5 形式をサーバの 0025 形式に変換する必要があることを指定します。

変換テーブルは，VuGen のインストール先ディレクトリの **ebcdic** ディレクトリにあります。システムで別の変換テーブルを使用している場合，テーブルを **ebcdic** ディレクトリにコピーします。

注：データが ASCII 形式の場合，変換の必要はありません。［**None**］オプション（標準設定）を選択します。変換テーブルを選択した場合は，VuGen は ASCII データを変換します。

Solaris マシンで作業しているときは，Vuser スクリプトを実行するすべてのマシンで次の環境変数を設定する必要があります。

```
setenv LRSDRV_SERVER_FORMAT 0025
setenv LRSDRV_CLIENT_FORMAT 04e4
```

ソケットの除外

VuGen ではソケット除外機能を使用して，記録セッションから特定のソケットを除外できます。スクリプトから特定のソケットを対象とするすべてのアクションをから除外するには，［除外する設定］の［ソケット リスト］からそのソケットのアドレスを選択します。ソケットをこのリストに追加するには，ボックスの右上角にあるプラス記号をクリックし，ソケットのアドレスを次のいずれかの形式で入力します。

値	意味
ホスト : ポート	指定したホストの指定したポートだけを除外します。
ホスト	指定したホストのすべてのポートを除外します。
: ポート	ローカル・ホストの指定したポートを除外します。
*: ポート	すべてのホストの指定したポートを除外します。

複数のホストとポートを除外するには，それらをリストに追加します。除外対象リストからソケットを削除するには，ソケットのアドレスを選択し，ボックスの右上角にあるマイナス記号をクリックします。ローカル・ホストや DNS ポート（53）など，テスト対象サーバの負荷に影響しないホストとポートを除外することをお勧めします。これらは標準設定では除外されています。

標準設定では，［除外する設定］の［ソケット リスト］で除外されたソケットのアクションがログに記録されることはありません。除外されたソケットのアクションをログに記録するには，［除外したソケットをログに含めない］チェック・ボックスをクリアします。除外されたソケットについてのログ記録を有効にすると，ログ・ファイルでは，アクションは「Exclude」という単語の後に記録されます。

```
Exclude :/* recv():15 bytes were received from socket 116 using flags 0 */
```

思考遅延時間のしきい値の設定

VuGen では記録中にオペレータの思考遅延時間が自動的に挿入されます。思考遅延時間のしきい値を設定して，それよりも低い場合には思考遅延時間を無視するようにできます。記録された思考遅延時間がしきい値を越えていると，VuGen によって LRS 関数の前に **lr_think_time** ステートメントが挿入されます。記録された思考遅延時間がしきい値を越えている場合，**lr_think_time** ステートメントは生成されません。

思考遅延時間のしきい値を設定するには，［**思考遅延時間のしきい値**］ボックスに必要な値（秒単位）を入力します。標準設定の値は 5 秒です。

LRS 関数の使用

Windows Sockets プロトコルを使ったクライアントとサーバの間の通信をエミュレートするために開発された関数を，LRS Vuser 関数と呼びます。LRS Vuser 関数には，**lrs** というプレフィックスが付きます。VuGen では，Windows Sockets セッション中に，この項で説明する LRS 関数のほとんどが自動的に記録されます。また，手作業で任意の関数をプログラミングして，スクリプトに挿入することもできます。

LRS 関数は，いくつかのカテゴリ（ソケット，バッファ，環境，ステートメント相関，変換，タイムアウト）に分類されます。

LRS 関数の詳細については，『**Online Function Reference**』（英語版）（［**ヘルプ**］＞［**関数リファレンス**］）を参照してください。

セッションを記録した後に，記録されたコードを VuGen に組み込まれているエディタで表示できます。スクリプトをスクロールし，アプリケーションによって生成された関数を表示し，転送されたデータを調べることができます。メイン・ウィンドウにスクリプトを表示すると，VuGen が動作状況を記録したシーケンスを確認できます。

次は，一般的なセッションで記録される関数の並びを示します。

lrs_startup	Windows Sockets DLL を初期化します。
lrs_create_socket	ソケットを初期化します。
lrs_send	データグラムで，またはストリーム・ソケットへデータを送信します。
lrs_receive	データグラムまたはストリーム・ソケットからデータを受信します。
lrs_disable_socket	ソケットの処理を無効にします。
lrs_close_socket	開いているソケットを閉じます。
lrs_cleanup	WinSock DLL の使用を終了します。

VuGen では，Windows 上で Windows Sockets プロトコルを使用するアプリケーションの記録と再生がサポートされます。UNIX プラットフォームでは再生だけがサポートされます。

Windows Sockets データを使った作業

VuGen を使用してアプリケーションを記録すると，データを含む複数のデータ・バッファができます。

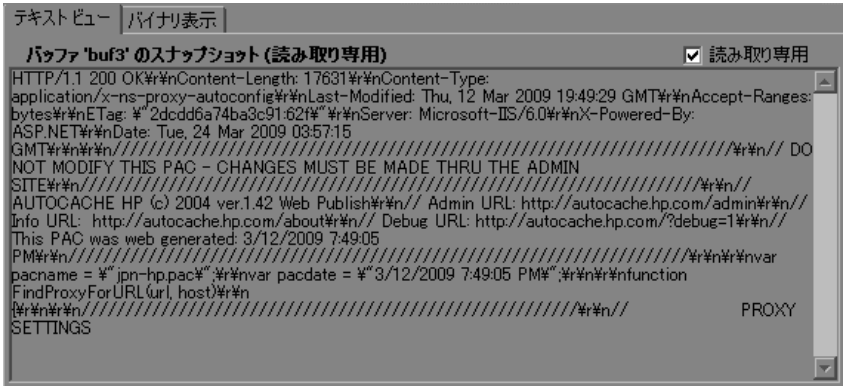
WinSocket スクリプトをツリー・ビューで表示すると，VuGen によって，データ・バッファ内の移動とデータの修正が可能なスナップショット・ウィンドウが表示されます。

スクリプト・ビューで作業しているときには，**data.ws** ファイルの未処理のデータを表示できます。詳細については，406 ページ「スクリプト・ビューでの Windows Sockets データの表示」を参照してください。

スナップショット・ウィンドウでのデータの表示

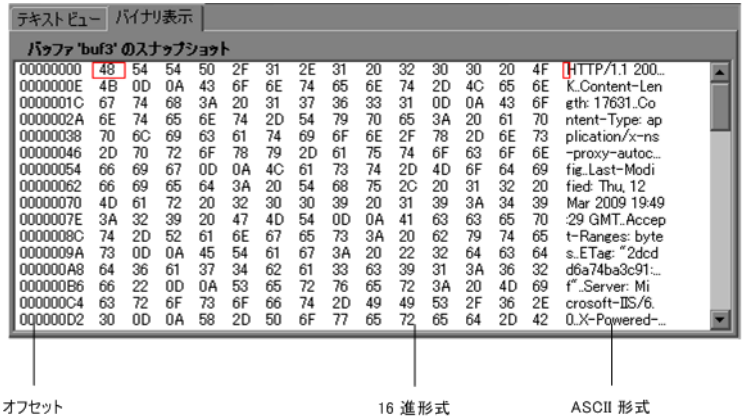
ツリー・ビューに Windows Sockets スクリプトを表示すると，編集が可能な［バッファのスナップショット］ウィンドウにデータが表示されます。スナップショットを［**テキスト ビュー**］または［**バイナリ表示**］に表示できます。

［テキスト ビュー］には，バッファのスナップショットの内容がテキストとして表示されます。



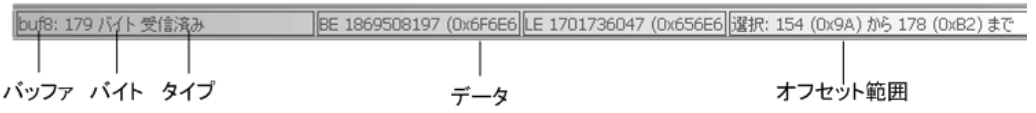
標準設定では，バッファ・データは読み取り専用として保存されます。バッファの内容を変更する場合は，バッファのテキスト・ビューで［**読み取り専用**］ボックスをクリアします。VuGen から，ブックマークとパラメータに影響がある可能性があるという警告が発行されます。

［バイナリ表示］にはデータが 16 進形式で表示されます。左のカラムには，行の最初の文字のオフセットが表示されます。中央のカラムには，データの 16 進値が表示されます。右のカラムには，データが ASCII 形式で表示されます。



バッファのスナップショットの下のステータス・バーには，データとバッファについての情報が提供されます。

- **バッファ番号**：選択されたバッファのバッファ番号。
- **合計バイト数**：バッファの合計バイト数。
- **バッファのタイプ**：バッファのタイプ（「受信済み」または「送信済み」）。
- **データ**：選択したデータの値をリトル・エンディアン順（バッファ内とは逆）の 10 進および 16 進で表示。
- **オフセット**：バッファの先頭からの選択（テキスト・ビュー内でのカーソル位置）のオフセット。複数のバイトを選択した場合は，選択範囲が示されます。



ステータス・バーには元のデータが変更されたかどうかも示されます。

データ内の移動

ツリー・ビューには，データ内を移動して，特定の値の識別と分析を行うためのいくつかのツールがあります。

- バッファ・ナビゲータ
- オフセットに移動
- ブックマーク

バッファ・ナビゲータ

標準設定では，VuGen の左の表示枠にすべてのステップおよびバッファが表示されます。［バッファ ナビゲータ］は，送信および受信バッファ・ステップ（**lrs_send**，**lrs_receive**，**lrs_receive_ex**，および **lrs_length_receive**）だけが表示されるフローティング・ウィンドウです。さらに，フィルタを適用して送信バッファまたは受信バッファの一方だけを表示できます。

［バッファ ナビゲータ］でバッファを選択すると，バッファの内容が［バッファのスナップショット］ウィンドウに表示されます。

記録後にバッファの名前を変えると，ステップをクリックしても，バッファの内容は［バッファのスナップショット］ウィンドウに表示されません。名前を変えたバッファのデータを表示するには，［バッファ ナビゲータ］を使って，新しいバッファ名を選択します。VuGen によって，選択したバッファのパラメータ作成が無効になることを示す警告メッセージが表示されます。

［バッファ ナビゲータ］を開くには，［**表示**］＞［**バッファ ナビゲータ**］を選択します。［バッファ ナビゲータ］を閉じるには，［バッファ ナビゲータ］ダイアログ・ボックスの右上角の［X］をクリックします。

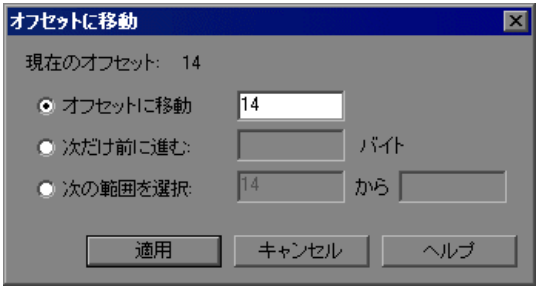
左の表示枠のツリー・ビューで，バッファ・ステップをクリックすることによっても，バッファ間を移動できます。［バッファ ナビゲータ］の利点は，それがフィルタ機能のあるフローティング・ウィンドウであることです。

オフセットに移動

オフセットを指定して，データ・バッファ内を移動できます。バッファ内におけるデータの絶対位置またはカーソルの現在位置に対する相対位置を指定できます。また，このダイアログ・ボックスで先頭と末尾のオフセットを指定することによって，ある範囲のデータを選択することもできます。

特定のオフセットへ移動するには，次の手順を実行します。

1 ［バッファのスナップショット］ウィンドウ内をクリックします。次に，右クリック・メニューから［**オフセットに移動**］を選択します。［オフセットに移動］ダイアログ・ボックスが表示されます。



2 バッファ内の特定のオフセット（絶対位置）に移動するには，［**オフセットに移動する**］をクリックして，オフセット値を指定します。

3 カーソルの相対位置へジャンプするには，［**次だけ前に進む**］を選択して，移動するバイト数を指定します。バッファ内を前進する場合は，正の数値を入力します。後退する場合は，負の数値を入力します。

4 バッファ内で，ある範囲のデータを選択するには，［**次の範囲を選択**］を選択して，先頭と末尾のオフセットを指定します。

ブックマーク

バッファ内の場所にブックマークとして印を付けることができます。それぞれのブックマークにわかりやすい名前を付けます。ブックマークをクリックすると，直接その場所に移動します。ブックマークは，バッファのスナップショットの下にある出力ウィンドウの［**ブックマーク**］タブに表示されます。

ブックマークは，［テキスト ビュー］でも［バイナリ表示］でも使用できます。［テキスト ビュー］で特定のデータの位置を見つけ，その位置をブックマークとして保存し，［バイナリ表示］の中でそのブックマークに直接ジャンプできます。

ブックマークは 1 バイトにも複数バイトにも付けられます。リスト中のブックマークをクリックすると，対応する場所が選択された状態で［バッファのスナップショット］ウィンドウに表示されます。初期設定では，そのデータがテキスト・ビューでは青で強調表示され，バイナリ・ビューではブックマーク・ブロックが赤で表示されます。また，バイナリ・ビューでブックマークにカーソルを置くと，ポップアップ・テキスト・ボックスが開いてブックマークの名前が表示されます。

永久ブックマークと標準ブックマークを作成できます。永久ブックマークは，バッファの［バイナリ表示］内で常に印が付けられています（青いボックスで囲まれています)。バッファ内の異なる位置を指している場合でも，このブックマークは選択された状態で青いボックスの中に表示されています。カーソルの位置は赤でマークされます。一方，標準ブックマークには常に印が付けられているわけではありません。標準ブックマークは，そこにジャンプしたときには赤く印が付きますが，バッファ内でカーソルを移動すると選択されていない状態になります。標準設定のブックマークは永久ブックマークです。標準設定のブックマークは永久ブックマークです。

ブックマークを処理するには，次の手順を実行します。

1 ブックマークを作成するには［バッファのスナップショット］（［テキストビュー］または［バイナリ表示］）で 1 つ以上のバイトを選択し，右クリックで表示されるメニューから［**新規ブックマーク**］を選択します。

2 ブックマーク・リストを表示するには，［**表示**］＞［**出力ウィンドウ**］を選択し，［**ブックマーク**］タブを選択します。

3 ブックマークに名前を割り当てるには，ブックマーク・リストのブックマークをクリックして，タイトルを編集します。

4 ブックマークの場所を変更するには，［**ブックマーク**］タブでブックマークを選択してから［バッファのスナップショット］で新しいデータを選択します。［**ブックマーク**］タブの［**変更**］をクリックします。

5 永久ブックマークを標準ブックマークに変更する場合（永久ブックマークは，カーソルを新しい場所に移動しても常にマークされた状態を維持します）は，ブックマークを選択してマウスを右クリックし，［**永久ブックマーク**］の横にあるチェックをクリアします。

6 リストに永久ブックマークだけを表示するには，［**ブックマーク**］タブで［**永久ブックマークのみ表示する**］チェック・ボックスを選択します。

7 特定のバッファのブックマークを表示するには，そのバッファからブックマークを選択し，［**フィルタ**］ボックスの［**選択バッファのみ**］を選択します。

8 ブックマークを削除するには，［**ブックマーク**］タブでブックマークを選択して［**削除**］をクリックします。

バッファ・データの修正

ツリー・ビューには，既存のデータを削除，変更，追加することによってデータを修正するためのいくつかのツールがあります。

- データの挿入
- データの編集
- データのパラメータ化

データの挿入

データ・バッファに数値を挿入できます。数値はシングルバイト，ダブルバイト，または 4 バイト値として挿入できます。

データ・バッファに数値を挿入するには，次の手順を実行します。

1 バッファ内をクリックします。

2 右クリックして表示されるメニューから［**詳細設定**］＞［**数字の挿入**］＞［**指定 ...**］を選択します。

3 挿入する ASCII 値を［**値**］ボックスに入力します。

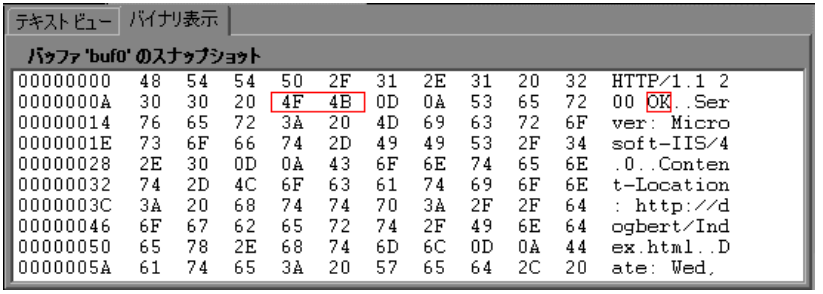
4 挿入するデータのサイズとして，1 バイト，2 バイト，または 4 バイトを［**サイズ**］ボックスから選択します。

5［**OK**］をクリックして完了します。VuGen によってデータの 16 進表現がバッファに挿入されます。

データの編集

バッファ・データに対して，標準的な編集操作（コピー，貼り付け，切り取り，削除，元に戻す）が行えます。［バイナリ表示］では，挿入する実際のデータを指定できます。VuGen では，データの形式（1，2，または 4 バイト，および 16 または 10 進値など）を指定できます。バイナリ・データをコピーし，数値としてバッファに挿入できます。［バイナリ表示］の右カラムには，10 進数または 16 進数を表示できます。

次の例では「**OK**」という語が選択されています。



データの次行の先頭で単純なコピー（CTRL+C）と貼り付け（CTRL+V）操作を実行すると，実際のテキストが挿入されます。

```
00000014  4F  4B  76  65  72  3A  20  4D  69  63  OKver: Mic
```

［**詳細設定**］＞［**数値としてコピー**］＞［**10 進法**］を選択してからデータを貼り付けると，選択された文字の ASCII コードの 10 進値が挿入されます。

```
00000014  31  39  32  37  39  76  65  72  3A  20  19279ver:
```

［**詳細設定**］＞［**数値としてコピー**］＞［**16 進法**］を選択してからデータを貼り付けると，選択された文字の ASCII コードの 16 進値が挿入されます。

```
00000014  30  78  34  42  34  46  76  65  72  3A  0x4B4Fver:
```

元戻しバッファに，選択したバッファに対して行われたすべての変更が保持されます。この情報はファイルに保存されるので，ファイルを閉じても利用できます。ほかの人に変更を取り消されないようにしたい場合は，元戻しバッファを空にします。元戻しバッファを空にするには，右クリックして表示されるメニューで［**詳細設定**］＞［**元戻しバッファを空にする**］を選択します。

バイナリ・ビューでバッファ・データを編集するには，次の手順を実行します。

1 バッファ・データをコピーするには，次の手順を実行します。

- ➤ 文字としてコピーする場合は，1 つ以上のバイトを選択し，CTRL+C キーを押します。
- ➤ 10 進数としてコピーする場合は，右クリックして表示されるメニューから［**詳細設定**］＞［**数値としてコピー**］＞［**10 進法**］を選択します。

➤ 16 進数としてコピーする場合は，右クリックして表示されるメニューから［**詳細設定**］＞［**数値としてコピー**］＞［**16 進法**］を選択します。

2 データを貼り付けるには，次の手順を実行します。

➤ 単一バイト（クリップボードのデータのサイズを単一バイトと仮定した場合）として貼り付ける場合は，バッファ内の望みの場所をクリックして CTRL+V キーを押します。

➤ 短い形式（2 バイト）として貼り付ける場合は，右クリックして表示されるメニューから［**詳細設定**］＞［**数字の挿入**］＞［**短形式で貼り付け（2 バイト）**］を選択します。

➤ 長い形式（4 バイト）として貼り付ける場合は，右クリックして表示されるメニューから［**詳細設定**］＞［**数字の挿入**］＞［**長形式で貼り付け（4 バイト）**］を選択します。

3 データを削除するには，［テキスト ビュー］または［バイナリ表示］で削除するデータを選び，右クリックして表示されるメニューから［**削除**］を選択します。

データのパラメータ化

ツリー・ビューの［バッファのスナップショット］ビューでデータを直接パラメータ化できます。パラメータ化する範囲を指定して境界を指定できます。パラメータ化文字列の境界を指定しなければ，VuGen によってスクリプトに **lrs_save_param** 関数が挿入されます。境界を指定すると，境界引数を指定できる **lrs_save_searched_string** 関数がスクリプトに挿入されます。

lrs_save_param 関数と **lrs_save_searched_string** 関数によってデータの相関が行われます。つまり，受信したデータが格納され，そのテスト内の以降の任意の場所で使用されます。相関では受信データが格納されるので，受信バッファにだけ適用され，送信バッファには適用されません。お勧めする手順は，受信バッファ内でパラメータ化する動的データの文字列を選択することです。同じパラメータを以降の送信バッファで使用します。

このタイプの相関を，単純なパラメータ化と混同しないでください。単純なパラメータ化（［**挿入**］＞［**新規パラメータ**］）は送信バッファ内のデータにだけ適用されます。パラメータを設定し，それに複数の値を割り当てます。VuGen によって，テストの実行ごと，または反復ごとに異なる値が割り当てられたパラメータが使用されます。詳細については，『**第 1 巻 ― VuGen の使用**』の「パラメータの作成」を参照してください。

次の各項では，受信バッファのデータの相関について説明します。

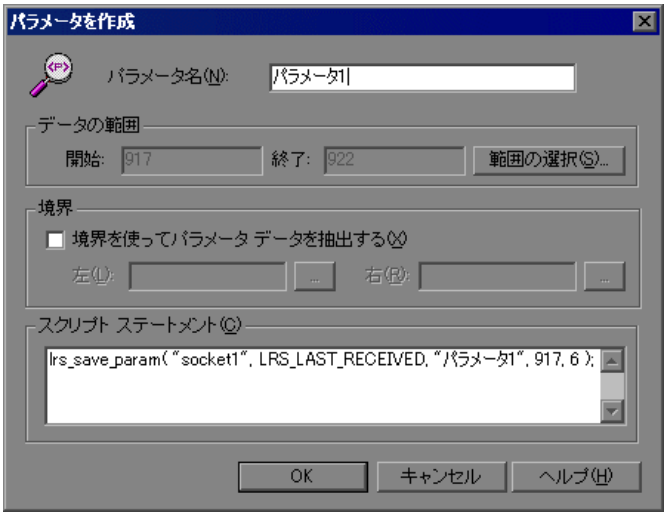
パラメータ作成後，VuGen によって文字列をパラメータに置き換えたすべての場所が表示されます。パラメータ作成についての情報，つまりパラメータが作成されたバッファやバッファ内のオフセットなどの情報も表示されます。［バッファのスナップショット］ビューの下の出力ウィンドウの［**パラメータ**］タブに，パラメータのすべての出現箇所のリストが表示されます。

VuGen でパラメータを操作できます。

- **絞り込み**：パラメータ名を指定してパラメータ置換の絞り込み表示ができます。
- **ソースへの移動**：置換を選択して［ソースに移動］をクリックすると，バッファ内の置換されたパラメータの場所に移動します。
- **パラメータの削除**：任意のパラメータを削除できます。パラメータを削除すると，データが元の値に置き換わり，スクリプトからパラメータ化関数が削除されます。
- **名前**：各置換に名前を付けることができます。
- **置換を元に戻す**：リストに表示された 1 つまたは複数の置換を元に戻すことができます。

スナップショット・ウィンドウでデータのパラメータ化を行うには，次の手順を実行します。

1 パラメータ化するデータを選択して，右クリックすると表示されるメニューから［**パラメータの作成**］を選択します（受信バッファにだけ有効）。ダイアログ・ボックスが開きます。



2 ［**パラメータ名**］ボックスにパラメータの名前を指定します。

3 パラメータ化するデータ範囲を選択します。標準設定では，バッファで選択されたデータ範囲が使用されます。ダイアログ・ボックスに表示された範囲とは異なる範囲を選択するには，［**範囲の選択**］をクリックします。選択されている範囲を示す小さなダイアログ・ボックスが表示されます。

［バッファのスナップショット］ウィンドウで範囲を選択して［**完了**］をクリックします。

4 パラメータ・データは定数ではないが，境界が一定の場合は，左右の境界を指定できます。

境界を指定するには，次の手順を実行します。

- ［**境界を使ってパラメータ データを抽出する**］チェック・ボックスを選択します。［**スクリプト ステートメント**］表示枠の関数が **lrs_save_param** から **lrs_save_searched_string** に変更されます。［**完了**］をクリックします。
- ［**境界**］セクションの［**左**］ボックスの隣にある参照ボタンをクリックします。バッファ内の選択を示す小さいダイアログ・ボックスが表示されます。バッファ内の境界を選択して［**完了**］をクリックします。右の境界についてもこの手順を繰り返します。

5 ［**スクリプト ステートメント**］セクションの引数に必要な修正を加えます。たとえば，**lrs_save_param** 関数に **_ex** を追加してエンコード・タイプを指定できます。これらの関数の詳細については，『**Online Function Reference**』（英語版）（［**ヘルプ**］＞［**関数リファレンス**］）を参照してください。

6 ［**OK**］をクリックしてパラメータを作成します。パラメータの置換前には確認を求められます。［**はい**］をクリックします。［**パラメータ**］タブにすべての置換を表示できます。

7 バッファ内のパラメータの元の場所に移動するには，パラメータを選択して［**ソースに移動**］をクリックします。

8 選択された置換のバッファの場所に移動するには，パラメータを選択して［**移動**］をクリックします。

9 パラメータ全体を削除するには，［**フィルタ**］ボックスのパラメータを選択して［**パラメータを削除**］をクリックします。

10 置換を取り消すには，［**パラメータ**］タブでパラメータを選択して［**元に戻す**］をクリックします。表示されているパラメータの置換をすべて取り消すには，［**パラメータ**］タブでパラメータを選択して［**すべて元に戻す**］をクリックします。

11 特定の置換を取り消すと，その出現について［パラメータ化済み］カラムに［**いいえ**］が表示されます。取り消した出現に対してパラメータ化ルールを適用するには，その出現を選択し，右クリックして表示されるメニューから［**パラメータで置換**］を選択します。

12 パラメータ全体を削除してすべての置換を元に戻すには，［**フィルタ**］ボックスのパラメータを選択して［**パラメータを削除**］をクリックします。

13 ［**仮想ユーザ**］＞［**パラメータ リスト**］を選択してデータをパラメータに割り当てます。

バッファ名の修正

バッファ名を修正するには，**data.ws** ファイルのスクリプト・ビューを使います。記録後にバッファ名を修正すると，Vuser スクリプトの再生に影響が及びます。バッファ・ナビゲータを使うと，名前を変更したバッファの内容をスクリプト・ビューまたはツリー・ビューに表示できます。

バッファで作成したブックマークが使えなくなっている場合は，定義されたバッファ内のブックマークを削除するよう求められます。

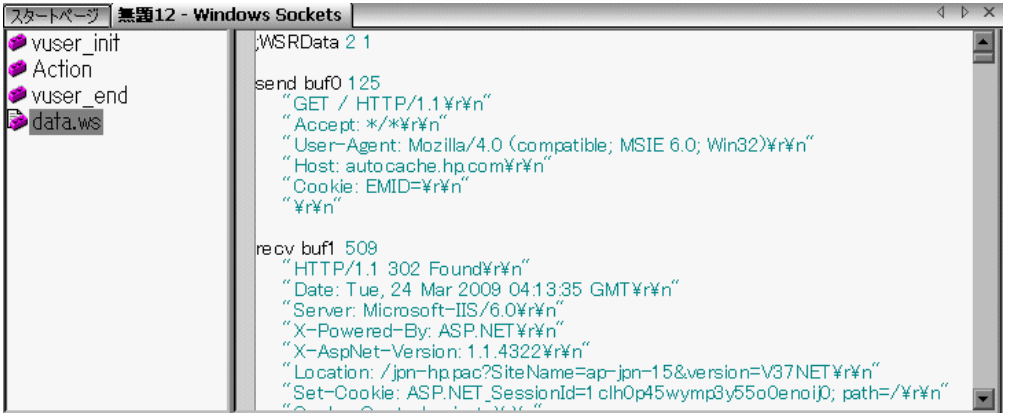
バッファで作成したパラメータが使えなくなっている場合，VuGen によって，パラメータが定義されたバッファ内のパラメータを削除するよう求められます。パラメータを削除すると，ほかのバッファも含め，すべての置換が元に戻ります。

名前を変更したバッファをバッファ・ナビゲータに表示すると，パラメータの作成がそのバッファ内で無効になることが VuGen によって警告されます。

スクリプト・ビューでの Windows Sockets データの表示

VuGen を使用して Windows Sockets Vuser スクリプトを作成すると，アクションはスクリプトの 3 つのセクション **vuser_init**，**Actions**，**vuser_end** に記録されます。Vuser スクリプトに加えて，記録セッション中に転送または受け取ったデータを含む **data.ws** というデータ・ファイルも作成されます。VuGen のメイン・ウィンドウの［データ ファイル］ボックスで **data.ws** を選択すれば，データ・ファイルの内容を表示できます。

データ・ファイルを表示するオプションは，Windows Sockets スクリプトでは標準で使用できます。データはスクリプト・ビューにのみ表示できます。



lrs_receive や **lrs_send** などのいくつかの LRS 関数は，サーバとクライアントの間で送信される実際のデータを処理します。受け取った，または転送されたデータはデータ・バッファに保管されますが，データ・バッファは非常に大きくなることがあります。Vuser スクリプトの可読性を高めるために，実際のデータは C ファイルではなく外部ファイルに保管されます。データ転送が発生すると，データは外部ファイルから一次バッファにコピーされます。

外部ファイルである **data.ws** には，すべての一時バッファの内容が含まれています。バッファの内容は連続したレコードとして保管されます。レコードはデータが送信されたか受信されたかを示す識別子とバッファ記述子によってマークされます。LRS 関数では，バッファ記述子を使ってデータにアクセスします。

記述子の形式は次のいずれかです。

```
recv buf ＜インデックス＞＜受信したバイト数＞
send buf ＜インデックス＞
```

バッファ・インデックスは 0（ゼロ）で始まり，以降のバッファは送受信に関係なくすべて順番に（1，2，3...）番号が付けられます。

次に示す例では，**lrs_receive** 関数が Vuser セッション中に記録されています。

```
lrs_receive("socket1", "buf4", LrsLastArg)
```

この例では，socket1 で受信したデータが **lrs_receive** によって処理されています。データは 5 番目の受信記録（buf4）に保管されています。バッファ・インデックスは 0（ゼロ）から始まります。**data.ws** ファイルの対応するセクションは，バッファとその内容を示します。

```
recv buf4 39
  "¥xff¥xfb¥x01¥xff¥xfb¥x03¥xff¥xfd¥x01"
  "¥r¥n"
  "¥r¥n"
  "SunOS UNIX (sunny)¥r¥n"
  "¥r"
  "¥x0"
  "¥r¥n"
  "¥r"
  "¥x0"
```

データ・ファイルの形式について

データ・ファイル **data.ws** の形式は以下のとおりです。

- ファイル・ヘッダ
- バッファとその内容のリスト

ファイル・ヘッダにはデータ・ファイル形式の内部バージョン番号が含まれます。現在のバージョンは 2 です。バージョン 1 の形式のデータ・ファイルからデータにアクセスしようとすると，エラーが発生します。

```
;WSRData 2 1
```

各レコードの前には，データの受信と送信を区別する識別子が付き，バッファ・インデックスと受信したバイト数（**lrs_receive** の場合のみ）が続きます。バッファ・インデックスはバッファを識別する番号です。

次に例を示します。

```
recv buf5   25
```

これは，バッファに受信したデータが含まれていることを表します。バッファ・インデックスが「5」なので，この受信操作は 6 番目のデータ転送（バッファ・インデックスは 0 から始まります）で，受信したデータは 25 バイトです。

データが ASCII 形式の場合，ソケットによって転送された実際の ASCII データが記述子の後に続きます。

データが EBCDIC 形式の場合，変換テーブルを通じて変換する必要があります。変換テーブルの設定については，389 ページ「WinSock 記録オプションの設定」を参照してください。EBCDIC データでも，変換後の ASCII 値が印字文字の場合は，ASCII 文字で表示されます。ASCII 値が非印字文字と対応している場合，VuGen によって元の EBCDIC 値が表示されます。

```
recv buf6 39
  "¥xff¥xfb¥x01¥xff¥xfb¥x03¥xff¥xfd¥x01"
  "¥r¥n"
  "SunOS UNIX (sunny)¥r¥n"
```

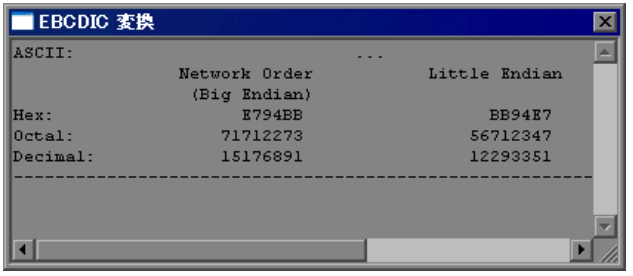
次の例は通常のデータ・ファイルのヘッダ，記述子およびデータを示します。

```
;WSRData 2 1

send buf0
    "¥xff¥xfd¥x01¥xff¥xfd¥x03¥xff¥xfb¥x03¥xff¥xfb¥x18"

recv buf1 15
    "¥xff¥xfd¥x18¥xff¥xfd¥x1f¥xff¥xfd"
    "#"
    "¥xff¥xfd"
    """
    "¥xff¥xfd"
    "$"

send buf2
    "¥xff¥xfb¥x18"
```

バッファ・データの 16 進形式での表示

VuGen には，データのオフセットのほかに，データのセグメントを 16 進形式と ASCII 形式で表示できるユーティリティがあります。

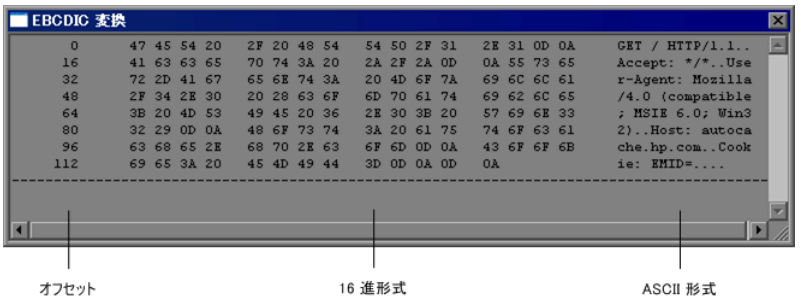
ビューア・ウィンドウでデータを表示するには，データを選択して F7 キーを押します。選択されたテキストが 4 文字以下の場合，VuGen によってそのデータが「**短い形式**」の 16 進，10 進，および 8 進で表示されます。



conv_frm.dat ファイルでは短い形式をカスタマイズできます。詳細については，412 ページ「表示形式の設定」を参照してください。

選択されたテキストが 4 文字を超える場合，VuGen では複数のカラムにデータが「**長い形式**」で表示されます。**conv_frm.dat** ファイルを変更すれば，長い形式をカスタマイズできます。詳細については，412 ページ「表示形式の設定」を参照してください。

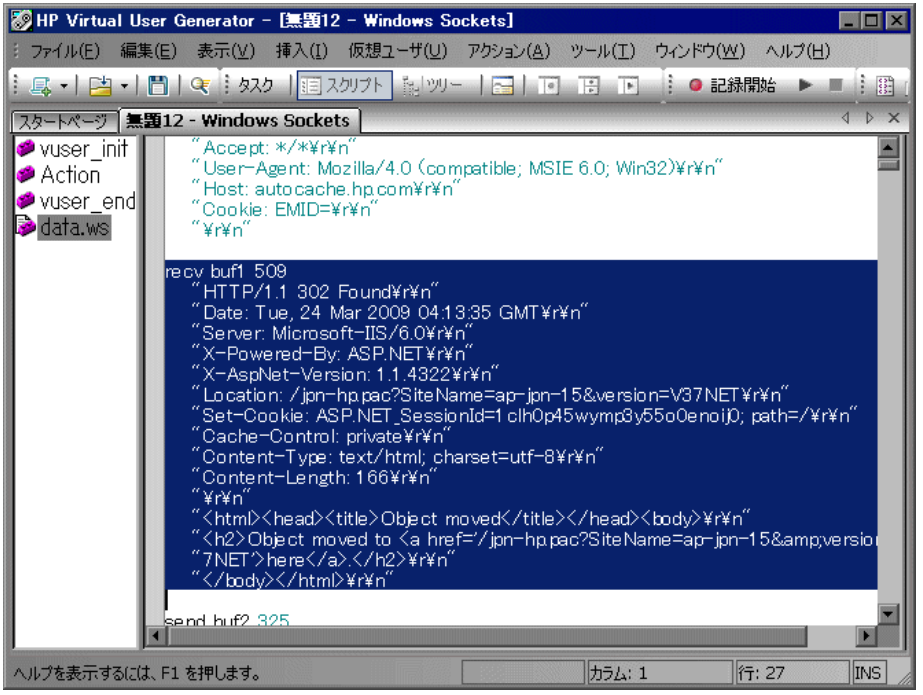
標準設定では，最初のカラムに，マークされた領域の先頭からの文字オフセットが表示されます。2 番目のカラムには，データが 16 進形式で表示されます。3 番目のカラムには，ASCII 形式でデータが表示されます。EBCDIC データを表示するときは，ASCII 形式の非印字文字（「/n」など）はすべてドットで表されます。



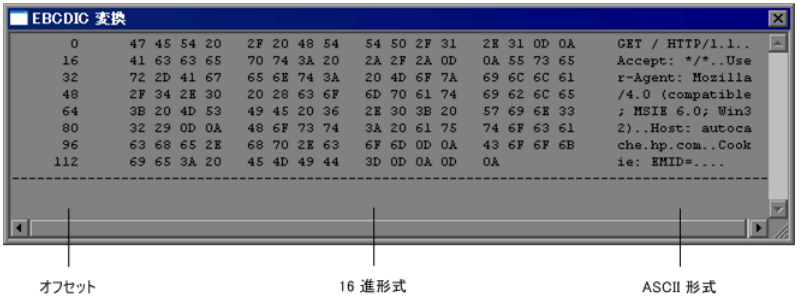
F7 キーを押すと表示されるビューア・ユーティリティは，特にパラメータ化を行う際に役立ちます。このユーティリティを使うことで，パラメータに保存するデータのオフセットを決定できます。

特定の文字のオフセットを決定するには，次の手順で行います。

1 **data.ws** を表示して，バッファの最初からデータを選択します。



2 F7 キーを押してデータと文字のオフセットを表示します。4 文字以上選択すると，データが長い形式で表示されます。



3 ASCII データの中で相関させる値を検索します。この例では，31 番目の文字で始まり，最後の文字が 2 行目にある 13546 番（UNIX セッション中のプロセス ID）を相関させます。

4 **lrs_save_param_ex** 関数のオフセット値を使って，プロセス ID の値を相関させます。詳細については，『**第 1 巻 ー VuGen の使用**』の「ステートメントの相関」を参照してください。

表示形式の設定

VuGen のビューア・ウィンドウ（F7 キー）でのバッファ・データの表示方法を指定できます。＜ **LoarRunner ディレクトリ**＞ **¥dat** ディレクトリの **conv_frm.dat** ファイルには，次の表示パラメータが含まれています。

- **LongBufferFormat**：5 文字以上を表示するときに使われる形式です。オフセットには nn，16 進データには XX，ASCII データには aa を使います。
- **LongBufferHeader**：長いバッファ形式で表示する場合に，各バッファの先頭に表示するヘッダです。
- **LongBufferFooter**：長いバッファ形式で表示する場合に，各バッファの末尾に表示するフッタです。
- **ShortBufferFormat**：4 文字以下を表示するときに使われる形式です。標準的なエスケープ・シーケンスと変換文字を使用できます。

  サポートされているエスケープ・シーケンス文字は次のとおりです。

¥a	**ベル（アラート）**
¥b	バックスペース
¥f	改ページ
¥n	改行
¥r	復帰
¥t	水平タブ
¥v	垂直タブ
¥'	引用符
¥"	二重引用符

¥a	ベル（アラート）
¥¥	円記号
¥?	疑問符
¥ooo	ASCII 文字（8 進数）

サポートされている変換文字は次のとおりです。

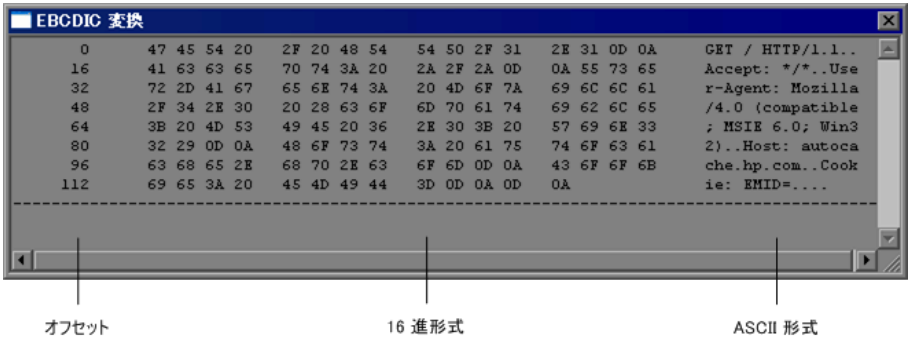
%a	ASCII 表記
%BX	ビッグ・エンディアン（ネットワーク順序）16 進数
%BO	ビッグ・エンディアン（ネットワーク順序）8 進数
%BD	ビッグ・エンディアン（ネットワーク順序）10 進数
%LX	リトル・エンディアン 16 進数
%LO	リトル・エンディアン 8 進数
%LD	リトル・エンディアン 10 進数

- **AnyBufferHeader**：各バッファの先頭に表示するヘッダ。
- **AnyBufferFooter**：各バッファの末尾に表示するフッタ。
- **NonPrintableChar**：非印字 ASCII 文字を表す文字。
- **PrintAllAscii**：非印字 ASCII 文字を強制的に出力するには，1 に設定します。

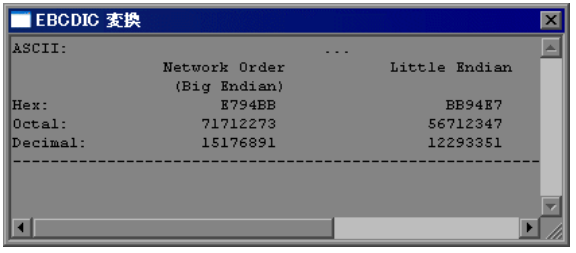
標準設定では，長い形式と短い形式が設定され，非印字文字はドットとして出力するように指定されています。

```
[BufferFormats]
LongBufferFormat=nnnnnnnn     XX XX XX XX   XX XX XX XX   XX XX XX XX   XX XX
XX XX     aaaaaaaaaaaaaaaa¥r¥n
LongBufferHeader=
LongBufferFooter=
ShortBufferFormat=ASCII:¥t¥t¥t%a¥r¥n¥t¥tNetwork Order¥t¥tLittle Endian¥r¥n¥t¥t (Big
Endian)¥r¥nHex:¥t¥t%BX¥t¥t%LX¥r¥nOctal:¥t¥t%BO¥t¥t%LO¥r¥nDecimal:¥t%BD¥t¥t%LD¥r¥n
AnyBufferHeader=
AnyBufferFooter=----------------------------------------------------------------------¥r¥n
NonPrintableChar=.
PrintAllAscii=0
```

標準設定の LongBufferFormat は次のように表示されます。



標準の ShortBufferFormat は次のように表示されます。



デバッグに関するヒント

VuGen では，いくつかの方法でスクリプトのデバッグを行えます。スクリプト実行に関する詳細メッセージを示すさまざまな出力ログとウィンドウを表示できます。

特に Windows Sockets Vuser スクリプトについては，「バッファの不一致」に関する追加情報が提供されます。バッファの不一致とは，受け取ったバッファ（再生中に生成されたもの）のサイズと期待バッファ（記録中に生成されたもの）の差を示します。ただし，受け取ったバッファと期待バッファが同じサイズの場合は，内容が異なっていても不一致のメッセージは発行されません。この情報は，システムや Vuser スクリプトでの問題を見つけるのに役立ちます。

バッファの不一致情報は実行ログに記録できます。実行ログが表示されていないときは，［**表示**］＞［**出力ウィンドウ**］を選択して，実行ログを表示します。

バッファの不一致は必ずしも問題を示しているわけではありません。たとえば，バッファに以前のログイン時刻などの重要でないデータが含まれている場合，このような不一致は無視できます。

```
Mismatch (expected 54 bytes, 58 bytes actually received)
The expected buffer is:
=================
¥r¥n Last login: Wed Sep 2 10:30:18 from acme.hplab.c¥r¥n
=================
The received buffer is:
=================
¥r¥n Last login: Thu Sep 10 11:19:50 from dolphin.hplab.c¥r¥n
```

ただし，期待バッファと受信バッファの間でサイズが著しく異なる場合は，システムに問題があることを示します。対応するバッファでデータの不一致を確認してください。

重要な不一致かどうかを判断できるように，アプリケーションを完全に理解しておく必要があります。

WinSock スクリプトの手作業による相関

VuGen には，Vuser スクリプトを相関させるためのユーザ・インタフェースがあります。相関は，動的なデータを利用する場合に必要です。WinSock Vuser スクリプトでよくある問題の 1 つに，ポート番号が動的に割り当てられる「動的ポート」があります。特定のアプリケーションが常に同じポートを使用していると，ほかのアプリケーションは次の使用可能なポートを使用します。スクリプトを再生しようとしたときに，記録されたポートが使用できない場合，テストは失敗します。この問題に対処するには，相関を行う必要があります。実際の実行時の値を保存し，それらの値をスクリプト内で使用します。

動的な値をパラメータに保存する相関関数を使用して，Vuser スクリプトを手作業で相関させることができます。**lrs_save_param** と **lrs_save_param_ex** 関数では，受信バッファ内のデータのオフセットに基づいて，データをパラメータに保存できます。高度な相関関数 **lrs_save_searched_string** を使えば，データの境界と，一致パターンの何回目の出現をパラメータに保存するかを指定できます。次の例では，**lrs_save_param_ex** を使った相関について説明します。ほかの相関関数の使用法については，『**Online Function Reference**』（英語版）を参照してください。

WinSock Vuser のステートメントを相関させるには，次の手順を実行します。

1 スクリプト内の，バッファの内容を保存する場所に **lrs_save_param_ex** ステートメントを挿入します。ユーザ・バッファ，静的バッファ，または受信バッファを保存できます。

```
lrs_save_param_ex (socket, type, buffer, offset, length, encoding, parameter);
```

2 パラメータを参照します。

VuGen のメイン・ウィンドウの［データ ファイル］ボックスで **data.ws** ファイルを選択して，バッファの内容を表示します。保存したバッファの内容で置換するデータを探します。置換する値のすべてのインスタンスを山括弧（＜＞）で囲まれたパラメータ名で置き換えます。

次の例では，ユーザが telnet セッションを実行しています。ユーザは ps コマンドを使ってプロセス ID（PID）を調べ，その PID を使ってアプリケーションを強制終了しています。

```
frodo:/u/jay>ps
    PID TTY      TIME CMD
14602 pts/180:00 clock
14569 pts/180:03 tcsh

frodo:/u/jay>kill 14602
[3] Exit 1          clock
frodo:/u/jay>
```

テストの実行時の PID は異なる（UNIX は各実行に固有の PID を割り当てます）ので，記録された PID を強制終了しても意味がありません。この問題に対処するには，**lrs_save_param_ex** を使って，現在の PID をパラメータに保存します。そして定数をパラメータで置き換えます。

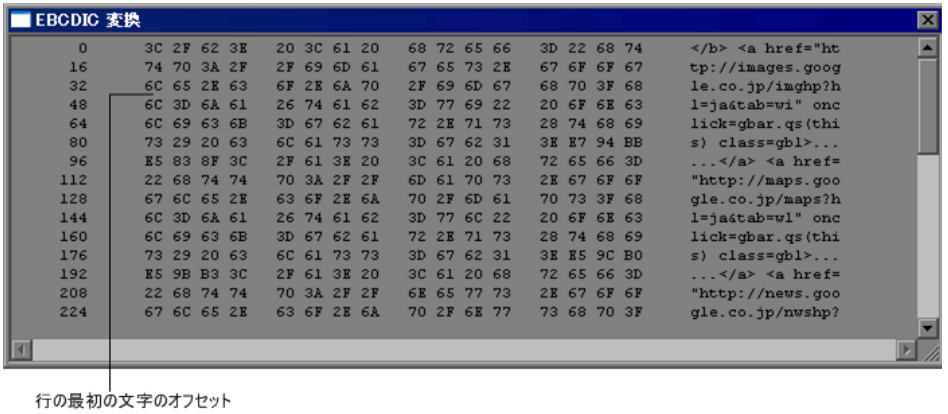
3 data.ws ファイル内で，データを受け取ったバッファを調べます（この例では buf47）。

```
recv buf47 98
  "¥r"
  "¥x00"
  "¥r¥n"
  "   PID TTYTIME CMD¥r¥n"
  " 14602 pts/180:00 clock¥r¥n"
  " 14569 pts/180:02 tcsh¥r¥n"
  "frodo:/u/jay>"
.
.
.
send buf58
  "kill 14602"
```

4 Actions セクションで，buf47 によって使用されるソケットを調べます。この例では socket1 です。

```
lrs_receive("socket1", "buf47", LrsLastArg);
```

5 保存するデータ文字列のオフセットと長さを調べます。バッファ全体を強調表示して F7 キーを押します。**PID** のオフセットは 11 で，長さは 5 バイトです。これらのデータの表示方法の詳細については，408 ページ「データ・ファイルの形式について」を参照してください。



6 Actions セクションで，当該バッファの lrs_receive 関数の後に **lrs_save_param_ex** 関数を挿入します。この例では，バッファは **buf47** です。PID は param1 という名前のパラメータに保存されます。**lr_output_message** を使って出力にパラメータを送信します。

```
lrs_receive("socket1", "buf79", LrsLastArg);
lrs_save_param("socket1", "user", buf47, 11, 5, ascii, param1);
lr_output_message ("param1: %s", lr_eval_string("<param1>"));
lr_think_time(10);
lrs_send("socket1", "buf80", LrsLastArg);
```

7 data.ws ファイル内で，パラメータで置換するデータを調べます（この例では PID）。

```
send buf58
 "kill 14602"
```

8 値を山括弧で囲まれたパラメータで置換します。

```
send buf58
 "kill <param1>"
```

第 26 章

VuGen エディタでのスクリプトのプログラミング

セッションを記録する方法に加えて，ユーザ定義の Vuser スクリプトを作成することができます。Vuser API 関数と標準の C，Java，VB，VBScript，JavaScript コードの両方を使用できます。

本章の内容

ユーザ定義の Vuser スクリプトの作成について

VuGen では，実際のセッションを記録せずに，独自の関数をスクリプトにプログラミングできます。Vuser API または標準のプログラミング関数を使用できます。Vuser API 関数では，Vuser についての情報を収集できます。たとえば，Vuser 関数を使って，サーバ・パフォーマンスの測定，サーバ負荷の制御，デバッグ・コードの追加，テストに含まれる Vuser または監視対象の Vuser についてのランタイム情報の取得などができます。

本章では，対象アプリケーションのライブラリやクラスと連携して動作する Vuser スクリプトを VuGen エディタでプログラミングする方法について説明します。

また，Visual C および Visual Basic 環境からプログラミングを行って Vuser スクリプトを開発することもできます。これらの環境では，開発アプリケーションに Vuser API 関数のライブラリをインポートして Vuser スクリプトを作成します。

ユーザ定義のスクリプトを作成するには，まずスクリプトのスケルトンを作成します。スクリプトのスケルトンには，スクリプトの 3 つの主要セクション，**init**，**actions**，および **end** が含まれています。これらのセクションは最初は空なので，手作業で関数を挿入します。

空のスクリプトは，次のプログラミング言語で作成できます。

- C
- Java
- Visual Basic
- VBScript
- JavaScript

注： JavaScript と VBScript Vuser を使う場合スクリプトで使用する COM オブジェクトは完全なオートメーション対応である必要があります。これによって，あるアプリケーションが別のアプリケーションにあるオブジェクトを操作したり，オブジェクトを外部から操作できるように公開したりできます。

C Vuser

C Vuser スクリプトでは，標準 ANSI 規格の C 言語のコードを配置できます。空の C Vuser スクリプトを作成するには，［新規仮想ユーザ］ダイアログ・ボックスの［ユーザ定義］カテゴリから［C Vuser］を選択します。VuGen によって，空のスクリプトが作成されます。

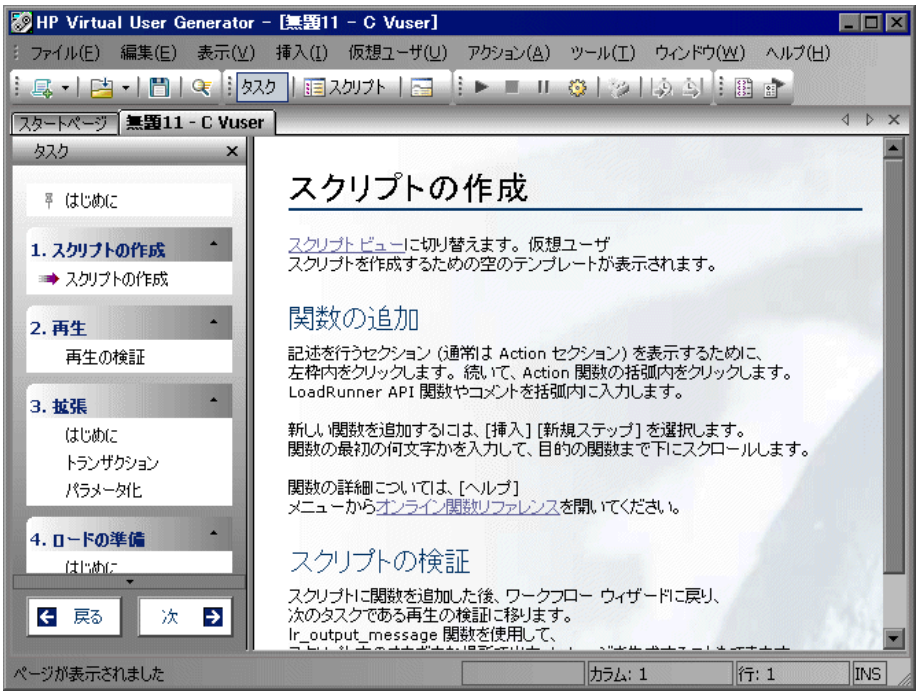
```
Action1()
{

    return 0;
}
```

C 言語の関数を使用するタイプの Vuser スクリプトであれば，どのスクリプトでも C Vuser 関数を使用できます。

よく使用される C 言語の関数の構文や使用例については『**Online Function Reference**』（英語版）（［**ヘルプ**］＞［**関数リファレンス**］）の C 言語リファレンスを参照することもできます。

C Vuser スクリプトのワークフロー・ウィザード

ワークフロー・ウィザードは，スクリプトの作成の手順を導いてくれます。［タスク］表示枠のリンクをクリックすると，スクリプト作成の手順についての説明を読むことができ，再生に関する情報を表示できます。［**戻る**］および［**次**］ボタンを使用して，画面間を移動できます。



ワークフロー・ウィザードが見えない場合は，［タスク］表示枠が開いていることを確認してください（［タスク］表示枠は，ツールバーの［**タスク**］ボタンを使用して表示 / 非表示を切り替えます。）次に最初のリンクである［**はじめに**］をクリックします。

ウィザードの詳細については，『**第 1 巻 ― VuGen の使用**』の第 4 章「VuGen ワークフローの表示」を参照してください。

スクリプトの作成

［スクリプトの作成］ウィンドウには，Web Services スクリプトの作成のいくつかのガイドラインが含まれます。

- ［**関数の追加**］：関数の入力方法と入力場所を示します。
- ［**スクリプトの検証**］：関数の追加後のスクリプトの検証方法を示します。

C 言語の関数の使用についてのガイドライン

制御フローや構文など，標準 ANSI-C の規約はすべて，C Vuser スクリプトにも適用されます。ほかの C 言語のプログラムと同じように，スクリプトでもコメント文や条件文が利用できます。ANSI-C の規約に従って変数を宣言して定義できます。

Vuser スクリプトの実行に使用される C インタプリタは，標準の ANSI C 言語を受け付けます。ANSI-C の Microsoft 拡張はサポートしていません。

C 言語関数を Vuser スクリプトに追加するときは，以下の制限に注意してください。

- Vuser スクリプトでは，関数のアドレスをコールバックとしてライブラリ関数に渡すことはできません。
- **stdargs**，**longjmp**，および **alloca** 関数は Vuser スクリプトではサポートされていません。
- Vuser スクリプトには，構造体引数や戻り値の型はありません。構造体へのポインタはサポートされています。
- Vuser スクリプト内のリテラル文字列は読み取り専用です。リテラル文字列に書き込もうとするとアクセス違反が起こります。
- int を返さない C 関数は型変換を行う必要があります。次に例を示します。
  extern char * strtok();

libc 関数の呼び出し

Vuser スクリプトで，**libc** 関数を呼び出すことができます。ただし，Vuser スクリプトの実行で使われているインタプリタが ANSI C の Microsoft 社による拡張をサポートしていないため，Microsoft 社のインクルード・ファイルを使うことはできません。自分自身のプロトタイプを書くか，HP のカスタマ・サポートに連絡して，**libc** 関数のプロトタイプを含む ANSI 準拠のインクルード・ファイルを入手してください。

リンク・モード

Vuser スクリプトの実行に使用する C インタプリタでは，使用前に関数を定義しておくかぎりは実行開始時に定義する必要がないようにするために，「遅延」リンク・モードを使用します。次に例を示します。

```
lr_load_dll("mydll.dll");
    myfun();  /* mydll.dll で定義済み。
                 myfun.dll がロードされたらすぐに直接呼び出せる。*/
```

Java Vuser

Java Vuser スクリプトでは，標準 Java コードが使用できます。空の Java Vuser スクリプトを作成するには，［新規仮想ユーザ］ダイアログ・ボックスの［ユーザ定義］カテゴリから［Java Vuser］を選択します。VuGen によって，空の Java スクリプトが作成されます。

```
import lrapi.lr;

public class Actions
{

    public int init() {
        return 0;
    }



    public int action() {
        return 0;
    }


    public int end() {
        return 0;
    }
}
```

Java Vuser タイプでは，**Actions** クラスの編集しかできないことに注意してください。Actions クラスの中には，**init**，**action** および **end** の 3 つのメソッドがあります。**init** メソッドに初期化コードを，**action** メソッドにビジネス・プロセスを，**end** メソッドにクリーンアップ・コードを記述します。

VB Vuser

空の Visual Basic Vuser スクリプトを作成して，Visual Basic コードを書くことができます。このスクリプト・タイプでは，Visual Basic アプリケーションを VuGen に取り入れることができます。空の VB Vuser スクリプトを作成するには，［新規仮想ユーザ］ダイアログ・ボックスの［ユーザ定義］カテゴリから［VB Vuser］を選択します。VuGen によって，空の VB スクリプトが作成されます。

```
Public Function Actions() As Long

    '"TO DO: Place your action code here



    Actions = lr.PASS
End Function
```

VuGen によって，**vuser_init**，**action**，および **vuser_end** の 3 つのセクションが作成されます。これらのセクションにはそれぞれ **Init**，**Actions**，および **Terminate** の VB 関数が含まれています。コードは，TO DO コメントの指示に従って，これらの関数の中に配置します。

VuGen から表示できる追加のセクションは，Vuser と VB アプリケーションのオブジェクトおよび変数グローバル宣言が含まれる **global.vba** ファイルです。

VB Vuser スクリプトの再生エラー

VB Vuser スクリプトの再生を試みたときにエラー番号 -25210 のエラーが発生した場合は，一部の DLL ファイルに問題が存在する可能性があります。

解決方法

1 **c:¥Program Files¥Common Files¥Microsoft Shared¥vba¥vba6** ディレクトリを開きます。

2 **VBE6.dll** ファイルと **VBE6EXT.OLB** ファイルを探します。

3 ファイルを右クリックし，プロパティをクリックして，各ファイルのバージョンを確認します。

4 **VBE6.dll** ファイルまたは **VBE6EXT.OLB** ファイルのバージョンが 6.04.9972 ～ 6.05.1024 である場合は，両方のファイルを置き換える必要があります。どちらのファイルのバージョンもこの範囲外の場合は，HP ソフトウェア・サポートまでお問い合わせください。

5 バージョンが 6.04.9972 または 6.05.1024 の **VBE6.dll** ファイルを置き換えます。

6 バージョンが 6.04.9969 または 6.05.1024 の **VBE6EXT.OLB** ファイルを置き換えます。

VBScript Vuser

空の VBScript Vuser スクリプトを作成して，VBScript コードを配置できます。このスクリプト・タイプでは，VBScript アプリケーションを VuGen に取り入れることができます。空の VBScript Vuser スクリプトを作成するには，［新規仮想ユーザ］ダイアログ・ボックスの［ユーザ定義］カテゴリから［VB Script Vuser］を選択します。VuGen によって，空の VBScript Vuser スクリプトが作成されます。

```
Public Function Actions()

    '"TO DO: Place your action code here



    Actions = lr.PASS
End Function
```

VuGen によって，**vuser_init**，**action**，および **vuser_end** の 3 つのセクションが作成されます。これらのセクションにはそれぞれ **Init**，**Actions**，および **Terminate** の VBScript 関数が含まれています。コードは，TO DO コメントの指示に従って，これらの関数の中に配置します。

VuGen から表示できる追加のセクションは，Vuser API 関数と VB スクリプトのオブジェクトを作成する **global.vbs** ファイルです。たとえば，次のコードは標準のオブジェクト **lr** を作成します。

```
Set lr = CreateObject("LoadRunner.LrApi")
```

JavaScript Vuser

空の JavaScript Vuser スクリプトを作成して，JavaScript コードを配置できます。このスクリプト・タイプでは，既存の JavaScript アプリケーションを VuGen に取り入れることができます。空の JavaScript Vuser スクリプトを作成するには，［新規仮想ユーザ］ダイアログ・ボックスの［ユーザ定義］カテゴリから［Javascript Vuser］を選択します。

```
function Actions()
{
    //"TO DO: Place your business process/action code here



    return(lr.PASS);
}
```

VuGen によって，**vuser_init**，**action**，および **vuser_end** の 3 つのセクションが作成されます。これらのセクションにはそれぞれ **Init**，**Actions**，および **Terminate** の JavaScript 関数が含まれています。コードは，TO DO コメントの指示に従って，これらの関数の中に配置します。

VuGen から表示できる追加のセクションは，Vuser API 関数と Javascript のオブジェクトを作成する **global.js** ファイルです。たとえば，次のコードは標準のオブジェクト **lr** を作成します。

```
var lr = new ActiveXObject("LoadRunner.LrApi")
```

第 27 章

Java Vuser スクリプトのプログラミング

VuGen では Java タイプのユーザがプロトコル・レベルでサポートされています。本章では，プログラミングによって Java Vuser スクリプトを作成する方法について説明します。記録によって Java Vuser スクリプトを作成する方法については，Java プロトコルに関する章を参照してください。

本章の内容

Java Vuser スクリプトのプログラミングについて

Java コードを使って Vuser スクリプトを作成するには，**Java** タイプの Vuser を使用します。この Vuser タイプでは，プロトコル・レベルで Java がサポートされています。Vuser スクリプトは Java コンパイラによってコンパイルされ，Java の標準規約をすべてサポートします。たとえば，テキストの前に 2 つのスラッシュ「//」を付けることによって，コメントを挿入できます。

第 14 章「Java プロトコル ― 記録」では，**Java Record Replay** Vuser を使って記録によってスクリプトを作成する方法を説明しています。プログラミングによって Java コードのスクリプトを作成するには，次の各項を参照してください。

Java 互換の Vuser スクリプトを作成する第一歩は，**Java Vuser** タイプの新しい Vuser スクリプトのテンプレートを作成することです。次に，任意の Java コードをスクリプト・テンプレートにプログラミングするか，貼り付けます。Java Vuser 関数を追加して，スクリプトを拡張したり，反復時にさまざまな値を使用できるように引数をパラメータ化したりできます。

Java Vuser スクリプトは，スケーラブルなマルチ・スレッド・アプリケーションとして稼動します。スクリプトにユーザ定義のクラスを含める場合は，コードをスレッドセーフにする必要があります。コードをスレッドセーフにしないと，結果が不正確になることがあります。スレッドセーフでないコードの場合は，Java Vuser をプロセスとして実行します。つまり，プロセスごとに個別の Java 仮想マシンを作成します。その結果，スクリプトの拡張性は低くなります。

スクリプトを作成したら，VuGen でスタンドアロン・テストとして実行します。Java コンパイラ（Sun の javac）によってスクリプトに誤りがないか調べられた後，コンパイルが実行されます。

スクリプトを作成した後，スクリプトを自分の環境（LoadRunner シナリオ，Performance Center 負荷テスト，または Business Process Monitor プロファイル）に組み込みます。詳細については，**HP LoadRunner Controller**，**HP Performance Center**，または **HP Business Availibility Center** のドキュメントを参照してください。

Java Vuser の作成

Java 互換の Vuser スクリプトを作成する第一歩は，Java Vuser テンプレートを作成することです。

Java Vuser スクリプトを作成するには，次の手順を実行します。

1 VuGen を開きます。

2 ［**ファイル**］＞［**新規作成**］を選択するか，［**新規作成**］ボタンをクリックします。［新規仮想ユーザ］ダイアログ・ボックスが開きます。

3 Vuser のタイプを選択するリストから［**ユーザ定義**］＞［**Java Vuser**］を選択し，［**OK**］をクリックします。空の Java Vuser スクリプトが表示されます。

4 左側の表示枠の **Actions** セクションをクリックして，**Actions** クラスを表示します。

Java Vuser スクリプトの編集

空のテンプレートを生成したら，任意の Java コードを挿入できます。このタイプの Vuser スクリプトを使用する場合は，すべてのコードを Actions クラスに配置します。Actions クラスを表示するには，左側の表示枠の［**Actions**］をクリックします。その内容が右側の表示枠に表示されます。

```
import lrapi.*;
public class Actions
{
    public int init() {
        return 0;
    }

    public int action() {
        return 0;
    }

    public int end() {
        return 0;
    }
}
```

Actions クラスには，init，action，end の 3 つのメソッドが含まれています。次の表に，各メソッドに何を含める必要があり，各メソッドがどのタイミングで呼び出されるかを示します。

スクリプトのメソッド	エミュレーション内容	実行のタイミング
init	サーバへのログイン	Vuser を初期化（ロード）するとき
action	クライアントの動作	Vuser が「実行」状態のとき
end	ログオフ処理	Vuser を終了または停止するとき

Init メソッド

すべてのログイン手続きと一度だけ行う設定を init メソッドに置きます。init メソッドは，Vuser がスクリプトの実行を開始するときに 1 回だけ実行されます。次のサンプル init メソッドではアプレットを初期化しています。

```
import org.omg.CORBA.*;
import org.omg.CORBA.ORB.*;
import lrapi.lr;

// Public function:init
    public int init() throws Throwable {

    // Orb インスタンスを初期化する ..
    MApplet mapplet = new MApplet("http://chaos/classes/", null);
    orb = org.omg.CORBA.ORB.init(mapplet, null);
...
```

action メソッド

Vuser で行うすべての操作を action メソッドに配置します。action メソッドは，実行環境の設定で指定した反復回数に従って実行されます。反復の設定の詳細については，『**第 1 巻 ー VuGen の使用**』の「実行環境の設定」を参照してください。次のサンプル action メソッドでは，Vuser ID を取り出して出力しています。

```
public int action() {
    lr.message("vuser: " + lr.get_vuser_id() + " xxx");
        return 0; v
}
```

end メソッド

end メソッドには，サーバからのログオフや環境の後始末など，スクリプトの終了時に Vuser に実行させるコードを配置します。

end メソッドは，Vuser がスクリプトの実行を終了するときに 1 回だけ実行されます。次の例に示す end メソッドでは「End」というメッセージを実行ログに出力しています。

```
public int end() {
    lr.message("End");
    return 0;
}
```

Java Vuser API 関数

VuGen には，Java Vuser スクリプト専用の Java API があります。これらの関数はすべて lrapi.lr クラスの静的メソッドです。

Java API 関数は，トランザクション，コマンド・ライン解析，情報，文字列，メッセージ，およびランタイム関数のカテゴリに分類されます。

個々の関数の詳細については，『**Online Function Reference**』（英語版）（［**ヘルプ**］＞［**関数リファレンス**］）を参照してください。Java Vuser スクリプトを新規作成すると，**import lrapi.*** がスクリプトに自動的に挿入されます。

Java クラスを追加して使うには，次の例に示すようにそれらをスクリプトの先頭でインポートします。

インポートするクラスのディレクトリや関連 jar ファイルをクラスパスに忘れずに追加します。また，追加するクラスがスレッドセーフかつスケーラブルであることを確認します。

```
import java.io.*;
import lrapi.*;

public class Actions
{
...
}
```

Java Vuser 関数を使った作業

Java Vuser 関数を使って次の作業を行うことにより，スクリプトを拡張できます。

- トランザクションの挿入
- ランデブー・ポイントの挿入
- Vuser 情報の取得
- 出力メッセージの発行
- ユーザの思考時間のエミュレート
- コマンド・ライン引数の処理

トランザクションの挿入

サーバのパフォーマンスを測定するには，トランザクションを定義します。各トランザクションは，Vuser からの特定の要求に対するサーバの応答時間を測定します。これらの要求は，単純な場合と複雑な場合があります。LoadRunner を使用している場合は，シナリオの実行中および実行後に，オンライン・モニタとグラフを使ってトランザクションごとのパフォーマンスを分析できます。

またトランザクションのステータスとして，lr.PASS および lr.FAIL を指定することもできます。トランザクションが正常終了したかどうか Vuser に判定させることができます。また，条件ループを使って自分で判定することもできます。たとえば，コードの中で，リターン・コードが特定の値になっているかどうかを調べることができます。リターン・コードが正しい値であれば，lr.PASS ステータスを発行します。リターン・コードの値が正しくなければ，lr.FAIL ステータスを発行します。

トランザクションの開始と終了を示すには，次の手順を実行します。

1 **lr.start_transaction** 関数は，スクリプト内で処理の実行時間の計測を開始する場所に挿入します。

2 **lr.end_transaction** 関数は，スクリプト内で処理の実行時間の計測を終了する場所に挿入します。**lr.start_transaction** 関数で指定したトランザクション名を指定します。

3 トランザクションのステータス（lr.PASS または lr.FAIL）を指定します。

```
public int action() {

for(int i=0;i<10;i++)
    {
      lr.message("action()"+i);
      lr.start_transaction("trans1");
      lr.think_time(2);
      lr.end_transaction("trans1",lr.PASS);
    }
      return 0;
}
```

ランデブー・ポイントの挿入

次の項は，HP Business Availability Center には適用されません。

クライアント / サーバ・システムを対象に多数のユーザによる負荷をエミュレートするには，「ランデブー・ポイント」を作成して，多数の Vuser が同時にタスクを実行するように同期させなければなりません。ある Vuser がランデブー・ポイントに到達すると，Controller によって，ほかのすべての Vuser がランデブー・ポイントに到着するまでその Vuser は待機させられます。

Vuser スクリプトにランデブー関数を挿入することによって，この「待ち合わせ場所」を指定します。

ランデブー・ポイントを挿入するには，次の手順を実行します。

➤ Vuser を待機させる場所に lr.rendezvous 関数を挿入します。

```
public int action() {

    for(int i=0;i<10;i++)
    {
      lr.rendezvous("rendz1");
      lr.message("action()"+i);
      lr.think_time(2);
    }
    return 0;
}
```

Vuser 情報の取得

次の関数を Vuser スクリプトに追加して，Vuser 情報を取得できます。

lr.get_attrib_string	Vuser ID や Load Generator の名前などのコマンド・ライン引数値または実行時情報を含む文字列を返します。
lr.get_group_name	Vuser グループの名前を返します。
lr.get_host_name	Vuser スクリプトを実行している Load Generator の名前を返します。
lr.get_master_host_name	LoadRunner Controller または Business Process Monitor を実行しているマシンの名前を返します。
lr.get_scenario_id	現在のシナリオの ID を返します（LoadRunner のみ）。
lr.get_vuser_id	現在の Vuser の ID を返します（LoadRunner のみ）。

次の例では，lr_get_host_name 関数を使って Vuser を実行しているコンピュータの名前を取得しています。

```
String my_host = lr.get_host_name( );
```

前述の関数の詳細については，『**Online Function Reference**』（英語版）（［**ヘルプ**］ ＞ ［**関数リファレンス**］）を参照してください。

出力メッセージの発行

シナリオを実行すると，Controller の［出力］ウィンドウに，スクリプトの実行に関するメッセージが表示されます。エラーおよび通知メッセージを Controller に送るためのステートメントを Vuser スクリプトに挿入することができます。Controller はこれらのメッセージを［出力］ウィンドウに表示します。たとえば，クライアント・アプリケーションの現在のステータスを示すメッセージを挿入することができます。また，これらのメッセージをファイルに保存することもできます。

注：トランザクション内からメッセージを送ってはなりません。トランザクション内からメッセージを送ると，トランザクションの実行時間が長くなり，実際のトランザクションの結果に偏りが生じる可能性があります。

Vuser スクリプトの中で，次のメッセージ関数を使用できます。

lr.debug_message	デバッグ・メッセージを［出力］ウィンドウに送ります。
lr.log_message	Vuser ログ・ファイルにメッセージを送信します。
lr.message	メッセージを［出力］ウィンドウに送ります。
lr.output_message	場所の情報を含むメッセージをログ・ファイルと［出力］ウィンドウに送ります。

次の例では，**lr.message** を使用してループ回数を示すメッセージを［出力］ウィンドウに送っています。

```
for(int i=0;i<10;i++)
    {
      lr.message("action()"+i);
      lr.think_time(2);
    }
```

メッセージ関数の詳細については，『**Online Function Reference**』（英語版）（［**ヘルプ**］＞［**関数リファレンス**］）を参照してください。

Vuser が Java の標準出力ストリームと標準エラー・ストリームを VuGen の実行ログにリダイレクトするように指定できます。これは，Vuser スクリプトに既存の Java コードを貼り付けるときや，**System.out** と **System.err** の呼び出しが含まれる既成のクラスを使用するときに，特に役に立ちます。実行ログでは，標準出力メッセージは青，標準エラーは赤で示されます。

lr.enable_redirection を使って，特定のメッセージを標準出力および標準エラーへリダイレクトする方法を次の例に示します。

```
lr.enable_redirection(true);

System.out.println("This is an informatory message..."); // リダイレクトされる
System.err.println("This is an error message..."); // リダイレクトされる

lr.enable_redirection(false);

System.out.println("This is an informatory message..."); // リダイレクトされない
System.err.println("This is an error message..."); // リダイレクトされない
```

注：lr.enable_redirection 関数を **true** に設定すると，この設定が既存のすべてのリダイレクト設定に優先します。以前のリダイレクト設定を復元するには，この関数を **false** に設定します。

この関数の詳細については『**Online Function Reference**』（英語版）（［**ヘルプ**］＞［**関数リファレンス**］）を参照してください。

ユーザの思考時間のエミュレート

連続する操作の間のユーザの待ち時間を「思考遅延時間」といいます。Vuser は，lr_think_time 関数を使ってユーザの思考遅延時間をエミュレートします。次の例では，Vuser がループの中で 2 秒待機しています。

```
for(int i=0;i<10;i++)
    {
      lr.message("action()"+i);
      lr.think_time(2);
    }
```

思考遅延時間の設定では，スクリプト中の値をそのまま使うことも，それらの値の倍数を使うこともできます。Vuser による思考遅延時間関数の処理方法を設定するには，［実行環境設定］ダイアログ・ボックスを開きます。詳細については，『**第 1 巻 － VuGen の使用**』の「実行環境の設定」を参照してください。

lr.think_time 関数の詳細については，『**Online Function Reference**』（英語版）（［**ヘルプ**］ ＞ ［**関数リファレンス**］）を参照してください。

コマンド・ライン引数の処理

スクリプトを実行するときにコマンド・ライン引数を指定することで，実行時に Vuser スクリプトに値を渡すことができます。Controller または Business Process Monitor で，スクリプトのパスとファイル名に続けてコマンド・ライン・オプションを挿入します。コマンド・ライン引数を読み取って，値を Vuser スクリプトに渡すことができる関数として，次の 3 つがあります。

lr.get_attrib_double	double float 型の引数を取得します。
lr.get_attrib_long	long int 型の引数を取得します。
lr.get_attrib_string	文字列を取得します。

コマンド・ラインの形式は次の 2 つのどちらかを使用します。スクリプト名の後ろに，引数とその値を 2 つ 1 組で指定します。

```
スクリプト名　- 引数 引数値　- 引数 引数値
```

```
スクリプト名　- 引数 引数値　- 引数 引数値
```

次の例は，pc4 というマシンで script1 を 5 回繰り返すためのコマンド・ライン文字列を示します。

```
script1  -host pc4  -loop 5
```

コマンド・ライン解析関数の詳細については，『**Online Function Reference**』（英語版）（［**ヘルプ**］ ＞ ［**関数リファレンス**］）を参照してください。コマンド・ライン・オプションの挿入方法の詳細については，**HP LoadRunner Controller**，**HP Performance Center**，または **HP Business Availibility Center** のドキュメントを参照してください。

Java 環境の設定

Java Vuser スクリプトを実行する前に，Vuser を実行するすべてのマシンで環境変数 PATH と CLASSPATH が正しく設定されていることを確認してください。

- スクリプトをコンパイルして再生するためには，JDK 1.1，1.2，または 1.3 のコンポーネントの完全インストールを実行しておく必要があります。JRE をインストールするだけでは不十分です。1 つのマシンに複数の JDK または JRE がインストールされているのは望ましくありません。可能ならば，不要なバージョンはすべてアンインストールします。
- 環境変数 **PATH** には，**JDK¥bin** のパスがなければなりません。
- JDK 1.1.x の場合は，環境変数 **CLASSPATH** に **classes.zip** のパス（**JDK/lib** サブディレクトリ）およびすべての VuGen クラス（**classes** サブディレクトリ）が含まれていなければなりません。
- Java Vuser によって使用されるすべてのクラスが，マシンの **CLASSPATH** 環境変数，または，実行環境設定の［**Classpath**］パネルで設定されている **Classpath Entries** に含まれている必要があります。

Java Vuser スクリプトの実行

Java Vuser スクリプトは，コンパイル後に実行される点が C Vuser スクリプトとは異なります。C Vuser スクリプトは，インタプリタによって解釈されます。VuGen ではインストールされている JDK から **javac** コンパイラが探し出され，スクリプト内の Java コードがコンパイルされます。このとき，VuGen ウィンドウの最下部に「**コンパイルしています ...**」というステータス・メッセージが表示されます。コンパイル中に発生したエラーは，実行ログに表示されます。スクリプトの中でエラーが発生したコードの位置に移動するには，エラーの行番号が示されているエラー・メッセージをダブルクリックします。エラーを修正し，スクリプトを再実行します。

コンパイルが完了すると，ステータス・メッセージが「**コンパイルしています ...**」から「**実行しています ...**」に変わり，スクリプトの実行が始まります。スクリプトに変更を加えていなければ，次にスクリプトを実行したときに，VuGen によるコンパイルが行われずにスクリプトが実行されます。スクリプトをさらに詳細にデバッグする場合は，ステップ実行オプションを使って，ブレークポイントを設定したり，表示しながらの実行を指定したりできます。

注：スクリプト内から JNDI の拡張機能を呼び出している場合には，Vuser を**スレッド**として実行しようとするときに問題が生じることがあります。これは，JNDI ではスレッドごとに独自のコンテキスト・クラス・ローダが必要とされているために発生します。スレッドとして実行するには，次の行を **init** セクションの先頭に追加して，Vuser が実行時に固有のコンテキスト・クラス・ローダを使用するように指定します。

```
DummyClassLoader.setContextClassLoader();
```

パッケージの一部としてのスクリプトのコンパイルと実行

Java Vuser スクリプトを作成するときに，クラスまたはメソッドが保護されている（protected）ほかのクラスのメソッドを使用しなければならないことがあります。このタイプのスクリプトをコンパイルしようとすると，コンパイルの段階で，メソッドにアクセスできないことを示すエラーが報告されます。スクリプトの中で確実にこれらのメソッドにアクセスできるようにするには，標準的な Java プログラムで行うのと同様の方法で，これらのメソッドを含むパッケージ名＜ package_name ＞をスクリプトの先頭に挿入します。以下の例では，スクリプトの中で特定のパスにある just.do.it パッケージを定義しています。

```
package my.test;

import lrapi.*;
public class Actions
{
    :
}
```

前述の例では，Vuser ディレクトリの下に **my/test** というディレクトリ階層が自動的に作成され，**Actions.java** ファイルが **my/test/Actions.java** にコピーされます。これにより，関連パッケージを使ったコンパイルが可能になります。package ステートメントは，Java の場合と同様にスクリプトの（コメントを除く）1 行目になければなりません。

プログラミングに関するヒント

Java Vuser スクリプトのプログラミングを行うときは，既成のコードのセグメントをスクリプトに貼り付けるか，既成のクラスをインポートして，そのメソッドを呼び出せるようにします。スケーラビリティを考慮して Vuser を Controller の管理下でスレッドとして実行する必要がある場合は，インポートするコードがすべてスレッドセーフであることを確認する必要があります。

多くの場合，コードがスレッドセーフであることを検出するのは困難です。VuGen および Controller では，Vuser の数が限られていれば，Java Vuser は問題なく実行されるかもしれません。しかし，ユーザ数が増えると問題が発生します。スレッドセーフでないコードは，通常は静的クラス・メンバを使用していることに起因します。次に例を示します。

```
import lrapi.*;
public class Actions
{
    private static int iteration_counter = 0;

    public int init() {
        return 0;
    }

    public int action() {
        iteration_counter++;
        return 0;
    }

    public int end() {
        lr.message("Number of Vuser iterations:"+iteration_counter);
        return 0;
    }
}
```

1 個の Vuser を実行すると，**iteration_counter** メンバは実行された反復の回数を正確に示します。1 台の仮想マシンで複数の Vuser を複数のスレッドとして実行すると，静的クラス・メンバの **iteration_counter** がすべてのスレッドで共有されるため，反復回数のカウントが不正確になります。これは，すべての Vuser の反復回数の合計がカウントされるからです。

スレッドセーフでないことがわかっているコードをスクリプトにインポートしたい場合は，それらの Vuser をプロセスとして実行できます。Vuser をスレッドまたはプロセスとして実行する方法の詳細については，『**第 1 巻 ー VuGen の使用**』の「実行環境の設定」を参照してください。

基本的な Java Vuser スクリプトを実行する場合，スクリプトは通常 1 個のスレッド（メイン・スレッド）で構成されています。Java Vuser API にアクセスできるのは，メイン・スレッドだけです。Java Vuser が 2 次的なワーカー・スレッドを生成したときに Java API を使用すると，予期しない結果が生じます。したがって，Java Vuser API はメイン・スレッドの中だけで使用することをお勧めします。この制限は，**lr.enable_redirection** 関数に影響を及ぼします。

次の例で，LR API を使用してもよい場所と使用してはいけない場所を示します。実行ログの最初のログ・メッセージは，flag の値が false であることを示します。その後，仮想マシンによって新規スレッド set_thread が生成されます。このスレッドは実行され，flag が true に設定されますが，lr.message への呼び出しにもかかわらず，ログにはメッセージが発行されません。最後のログ・メッセージは，スレッド内のコードが実行され，flag が true に設定されたことを示します。

```
boolean flag = false;

public int action() {
    lr.message("Flag value:"+flag);
    Thread set_thread = new Thread(new Runnable();{
        public void run() {
            lr.message("LR-API NOT working!");
            try { Thread.sleep(1000); } catch(Exception e) {}
            flag = true;
        }
    });
    set_thread.start();
    try { Thread.sleep(3000); } catch(Exception e) {}
    lr.message("Flag value:"+flag);
    return 0;
}
```

第 28 章

COM プロトコル

Windows アプリケーションの多くは，COM ベースの関数を直接呼び出すか，またはライブラリ呼び出しを介して使用します。VuGen を使って，COM ベースのクライアントによる COM サーバへのアクセスをエミュレートするスクリプトを記録できます。その結果生成されるスクリプトを，COM Vuser スクリプトと呼びます。Visual Basic Add-in を使って，COM Vuser スクリプトを作成することもできます。Visual Basic Add-in の詳細については，仮想ユーザ・ガイドの付録を参照してください。

第 29 章「COM の概要と相関」では，COM Vuser スクリプトの操作方法について説明します。

本章の内容

COM Vuser スクリプトの記録について

COM クライアント・アプリケーションを記録すると，VuGen によって COM クライアントとサーバの動作状況を表す関数が生成されます。記録されたスクリプトには，インタフェースの宣言，API 呼び出し，メソッドのインスタンス呼び出しが含まれています。各 COM 関数には，**lrc** というプレフィックスが付いています。

記録されたスクリプトは，VuGen のメイン・ウィンドウで表示および編集ができます。セッション中に記録された COM の API 呼び出しとメソッド呼び出しはウィンドウに表示されるので，アプリケーションの COM/DCOM 呼び出しを目で追うことができます。

Vuser スクリプトの作成に使用するプログラミング言語（C または Visual Basic）を指定できます。詳細については，『**第 1 巻 ー VuGen の使用**』の第 19 章「スクリプト生成オプションの設定」を参照してください。

COM の概要

この節では，COM 技術の概要を説明します。これらは COM Vuser スクリプトを使用する前に最低限知っておくべき情報です。詳細については，『Microsoft Developer Network（MSDN）』などのドキュメントを参照してください。

COM（Component Object Model）は，再利用可能なソフトウェア・コンポーネント（「プラグイン」）を開発するための技術です。DCOM（Distributed COM）は，リモート・コンピュータ上の COM コンポーネントを使用できるようにする技術です。MTS（Microsoft Transaction Server），Visual Basic，エクスプローラはすべて，COM/DCOM 技術を使用します。したがって，テスト対象のアプリケーションも，気が付かないところで COM 技術を間接的に使用していることがあります。多くの場合，アプリケーションによって実行された COM 呼び出しの一部（全部ではない）を Vuser スクリプトに加えなければならないでしょう。

オブジェクト，インタフェース，タイプ・ライブラリ

COM オブジェクトはバイナリ・コードのモジュールです。各 COM オブジェクトには，クライアント・プログラムとの通信を可能にする 1 つまたは複数のインタフェースが実装されています。Vuser スクリプト内の COM 呼び出しを追跡するには，これらのインタフェースを理解しておく必要があります。COM インタフェースのメソッドおよびパラメータにアクセスするための参照として使用されるタイプ・ライブラリには，COM オブジェクトとインタフェースに関する記述が含まれています。各 COM クラス，インタフェース，タイプ・ライブラリは，GUID（Global Unique Identifier）によって識別されます。

COM インタフェース

COM インタフェースは，相互に関連するメソッドの集合を提供します。たとえば，**Clock** オブジェクトには，**Clock**，**Alarm**，**Timer** のインタフェースがあるかもしれません。各インタフェースには，1 つまたは複数のメソッドがあります。たとえば，**Alarm** インタフェースには，**AlarmOn** メソッドおよび **AlarmOff** メソッドがあるかもしれません。

また，インタフェースは，1 つまたは複数のプロパティを持つことができます。メソッド呼び出しによる方法と，プロパティの値の設定または取得による方法の両方で同じ機能を実行できることがあります。たとえば，**Alarm Status** プロパティを **On** に設定した場合の結果は，**AlarmOn** メソッドを呼び出した場合の結果と同じです。

COM オブジェクトは，多数のインタフェースをサポートすることができます。コンポーネントには必ず **IUnknown** インタフェースが実装されており，ほかのインタフェースを調べるために使用できます。多くのコンポーネントは，**IDispatch** インタフェースも実装しています。IDispatch インタフェースは，オブジェクトのほかのインタフェースとメソッドをすべて公開して，スクリプト言語による COM オートメーションの実装を可能にします。

COM のクラスのコンテキストと場所の透過性

COM オブジェクトは，クライアント・アプリケーションを実行しているマシン，またはリモート・サーバ上で実行できます。アプリケーションによって作成される COM オブジェクトは，ローカル・ライブラリ，ローカル・プロセス，リモート・マシン（「リモート・オブジェクト・プロキシ」）のいずれかにあります。「コンテキスト」と呼ばれる COM オブジェクトのありかは，アプリケーションにとっては透過的です。大半のユーザは，Vuser を使ってリモート・サーバの負荷を検査します。このため，リモート・オブジェクト・プロキシがアクセスするオブジェクトが，通常，負荷テストの対象として最も意味があります。

COM のデータ型

COM は，セーフ配列，BSTR 文字列およびバリアントなど，いくつかの特殊なデータ型も提供します。これらのデータ型は，デバッグやパラメータ化のような作業で使用する必要があるかもしれません。

COM Vuser を使った作業の開始

本項では，COM Vuser スクリプトの作成するプロセスについて説明します。

COM Vuser スクリプトを作成するには，次の手順を実行します。

1 **VuGen を使用して基本スクリプトを記録します。**

VuGen を起動し，新しい Vuser スクリプトを作成します。Vuser のタイプとして COM を指定します。記録対象アプリケーションを選び，記録オプションを設定します。スクリプト記録オプションの設定については，『**第 1 巻 － VuGen の使用**』の「スクリプト生成オプションの設定」を参照してください。COM 固有のオプションとフィルタの設定方法については，456 ページ「COM 記録オプションの設定」を参照してください。アプリケーションを使って，一般的な操作を記録します。

記録方法の詳細については，『**第 1 巻 － VuGen の使用**』の「VuGen を使った記録」を参照してください。

2 **オブジェクト・フィルタを適切に設定し直します。**

生成されたログ・ファイルを使って，フィルタで記録対象のオブジェクトを選択し直します。詳細については，「記録対象の COM オブジェクトの選択」の項を参照してください。

3 スクリプトを拡張します。

トランザクション，ランデブー・ポイント，および制御フロー構造をスクリプトに挿入することによって，Vuser スクリプトを拡張します。

詳細については，『**第 1 巻 ― VuGen の使用**』の「Vuser スクリプトの拡張」を参照してください。

4 パラメータを定義します（任意）。

スクリプトに記録された固定値の代わりにパラメータを定義します。固定値をパラメータで置き換えることにより，同じビジネス・プロセスを，異なる値で何度も繰り返すことができます。

詳細については，『**第 1 巻 ― VuGen の使用**』の「パラメータの作成」を参照してください。

5 実行環境を設定します。

実行環境を設定することによって，スクリプト実行時の Vuser の動作を制御します。設定には，ループ，ログ，タイミング情報などがあります。

詳細については，『**第 1 巻 ― VuGen の使用**』の「実行環境の設定」を参照してください。

6 VuGen でスクリプトを実行します。

VuGen でスクリプトを保存して実行し，正しく動作することを確認します。

詳細については，『**第 1 巻 ― VuGen の使用**』の「スタンドアロン・モードでの Vuser スクリプトの実行」を参照してください。

スクリプトを作成した後，スクリプトを自分の環境（LoadRunner シナリオ，Performance Center 負荷テスト，または Business Process Monitor プロファイル）に組み込みます。詳細については，**HP LoadRunner Controller**，**HP Performance Center**，または **HP Business Availibility Center** のドキュメントを参照してください。

記録対象の COM オブジェクトの選択

テスト対象のアプリケーションは，多数の COM オブジェクトを使用する可能性があります。しかし，実際に負荷を生むのがごく少数の場合，負荷テストで重要となるのは，そのほんの一握りのオブジェクトだけかもしれません。したがって，COM アプリケーションを記録する前に，負荷テスト用に記録するオブジェクトを選択する必要があります。VuGen では，ローカル・マシンまたはネットワーク上のほかのコンピュータから読み込めるタイプ・ライブラリのオブジェクトを参照できます。

使用するオブジェクトの決定

テストに含める COM オブジェクトを指定する方法はいくつかあります。テスト対象のソフトウェアによって使用されるリモート・オブジェクトを調べてみます。どのオブジェクトを選択するべきかわからない場合は，標準のフィルタを使用してみます。フィルタの［環境］ノードの下には，リモート・サーバ上で負荷を生成する可能性のある 3 つのオブジェクト（ADO，RDS，Remote）への呼び出しが含まれています。

実際の呼び出しを調べて，フィルタを適切に設定し直すこともできます。テストを記録した後，ファイルを保存し，VuGen によって作成された **data** ディレクトリにある **lrc_debug_list_** ＜ **nnn** ＞ **.log**（**nnn** はプロセス番号）ファイルを調べます。このログ・ファイルには，各 COM オブジェクトが記録用フィルタに含まれていたかどうかに関係なく，記録されたアプリケーションによって呼び出された COM オブジェクトの一覧が含まれています。サーバに対する負荷を生成する呼び出しだけを，記録に含める必要があります。

たとえば，以下は Visual Basic ライブラリのローカルの COM の例です。

```
Class JetES {039EA4C0-E696-11D0-878A-00A0C91EC756}
was loaded from type library "JET Expression Service Type Library"
({2358C810-62BA-11D1-B3DB-00600832C573} ver 4.0)
```

これは，サーバ上で負荷を生成しないため，追加してはなりません。

同様に，次に示す OLE DB および Microsoft Windows Common Controls はローカル・オブジェクトなので，そのクラスとライブラリは，サーバへの負荷を生成しません。したがって，これらも記録する必要はありません。

```
Class DataLinks {2206CDB2-19C1-11D1-89E0-00C04FD7A829}
was loaded from type library "Microsoft OLE DB Service Component 1.0 Type Library"
({2206CEB0-19C1-11D1-89E0-00C04FD7A829} ver 1.0)

Class DataObject {2334D2B2-713E-11CF-8AE5-00AA00C00905}
was loaded from type library "Microsoft Windows Common Controls 6.0 (SP3)"
({831FDD16-0C5C-11D2-A9FC-0000F8754DA1} ver 2.0)
```

ただし，たとえば，次の一覧は，サーバで負荷を生成するため記録する必要があるクラスを示します。

```
Class Order {B4CC7A90-1067-11D4-9939-00105ACECF9A}
was loaded from type library "FRS"
({B4CC7A8C-1067-11D4-9939-00105ACECF9A} ver 1.0)
```

たとえば VuGen とともにインストールされる flight_sample で使用される **FRS** ライブラリのクラスは，サーバの処理能力を使用するので記録する必要があります。

COM オブジェクトが別の COM オブジェクトを呼び出す場合，すべての呼び出しがタイプ情報ログ・ファイルにリストアップされます。たとえば，アプリケーションが **FRS** クラス関数を呼び出すたびに，**FRS** ライブラリは **ADO**（**ActiveX Data Object**）ライブラリを呼び出します。このような呼び出しの連鎖に含まれるいくつかの関数がフィルタに表示されている場合，VuGen によって連鎖を開始する最初の呼び出しだけが記録されます。**FRS** および **ADO** 呼び出しの両方を選択した場合は，**FRS** 呼び出しだけが記録されます。

また，フィルタで **ADO** ライブラリだけを選択した場合は，**ADO** ライブラリへの呼び出しが記録されます。多くの場合，連鎖における最初のリモート・オブジェクトへの呼び出しを記録するのが簡単です。ただし，場合によっては，アプリケーションが複数の異なる COM オブジェクトのメソッドを使用します。それらのメソッドがいずれもサーバに負荷をかける単独のオブジェクトを使用する場合は，最終的な共通オブジェクトだけを記録することもできます。

選択可能なオブジェクト

VuGen では，オブジェクトのタイプ・ライブラリを読み込むことができれば，そのオブジェクトを記録できます。タイプ・ライブラリがシステムにインストールされていない場合や VuGen でタイプ・ライブラリが見つからない場合には，対応する COM オブジェクトは［記録オプション］ダイアログ・ボックスに表示されません。これらの COM オブジェクトがアプリケーションによって使用されている場合，VuGen ではこれらのオブジェクトを識別できず，ファイル内に **INoTypeInfo** として示されます。

除外できるインタフェース

［記録オプション］ダイアログ・ボックスでは，タイプ・ライブラリのリストに含まれているすべてのインタフェースが，オブジェクトごとに表示されます。このダイアログ・ボックスで，各インタフェースを記録に含めるか除外するかを指定できます。ただし，**ADO**，**RDS**，**Remote のオブジェクト**は，フィルタに 1 つのグループとして含めることができます。その場合，フィルタには，これらの環境またはインタフェースの個々のオブジェクトまたはインタフェースは表示されません。また，タイプ・ライブラリから記録対象にインクルードしたオブジェクトのインタフェースが，タイプ・ライブラリにないために［記録オプション］ダイアログ・ボックスに表示されない場合があります。その場合は，VuGen のスクリプトを生成した後で，スクリプトに含まれるこれらのインタフェースを特定して，VuGen によって生成された interfaces.h ファイルでそれらのインタフェースの GUID 番号を確認します。この情報を使って，以下に説明する方法で，インタフェースを除外できます。

COM 記録オプションの設定

COM の［記録オプション］ダイアログ・ボックスを使って，フィルタ・オプションと COM のスクリプト編集オプションを設定します。レジストリ，ファイル・システム，Microsoft トランザクション・サーバ（MTS）のタイプ・ライブラリを探すには，オンライン・ブラウザを使用します。

詳細については，次を参照してください。

- オブジェクトのフィルタリング
- フィルタの設定
- COM スクリプト編集オプションの設定

オブジェクトのフィルタリング

フィルタ・オプションを使って，VuGen で記録する COM オブジェクトを指定できます。オブジェクトは環境およびライブラリの中から選択できます。

フィルタ・オプションで，標準フィルタの設定または代替フィルタの作成を行います。環境およびタイプ・ライブラリを指定することで，記録セッションを絞り込めます。

DCOM プロファイル

- ［**Default filter**］: COM Vuser スクリプトを記録するときに標準で使用されるフィルタ。
- ［**新規フィルタ**］: 標準の環境設定に基づく新規のフィルタ。このフィルタの設定を使って記録するには，あらかじめフィルタに名前を付ける必要があります。

DCOM リスナーの設定

［DCOM リスナーの設定］には，タイプ・ライブラリのツリー階層が表示されます。ツリーの分岐を開いて，タイプ・ライブラリで使用できるクラスをすべて表示できます。クラス・ツリーの分岐を開いて，そのクラスによってサポートされているインタフェースをすべて表示できます。

タイプ・ライブラリを除外するには，ライブラリ名の横のチェック・ボックスをクリアします。これによって，そのタイプ・ライブラリに含まれているすべてのクラスが除外されます。ツリーの分岐を開き，各クラスまたはインタフェースの横のチェック・ボックスをクリアすることによって，それらの項目を個別に除外できます。

クラスの種類ごとに，異なるインタフェースを実装できます。除外されていないほかのクラスによって実装されているインタフェースを除外しようとすると，このインタフェースを実装しているすべてのクラスのインタフェースも除外するかどうかを尋ねるダイアログ・ボックスが表示されます。

インタフェースの横のチェック・ボックスをクリアすると，［除外インタフェース］ダイアログ・ボックスでそのインタフェースを選択したときと同じ結果が得られます。

- ［**環境**］: 記録対象の環境。［ADO objects］，［RDS objects］，および［Remote objects］があります。記録しないオブジェクトをクリアします。
- ［**タイプ ライブラリ**］: タイプ・ライブラリ。記録する COM オブジェクトを表す **.tlb** または **.dll** ファイルです。すべての COM オブジェクトには，オブジェクトを表すタイプ・ライブラリがあります。タイプ・ライブラリは，レジストリ，Microsoft Transaction Server，またはファイル・システムから選択できます。

［**タイプ ライブラリ**］: ダイアログ・ボックスの下側に，各タイプ・ライブラリの次の情報が表示されます。

- ［**TypLib**］: タイプ・ライブラリ（tlb ファイル）の名前。
- ［**Path**］: タイプ・ライブラリのパス。
- ［**Guid**］: タイプ・ライブラリの GUID（Global Unique Identifier）。

フィルタの設定

この項では，フィルタの設定方法について説明します。

記録対象の COM オブジェクトを選択するには，次の手順を実行します。

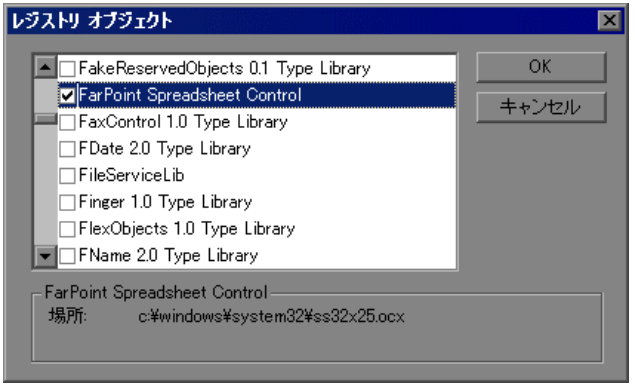
1 メイン・メニューの［**ツール**］＞［**記録オプション**］を選択するか，［記録開始］ダイアログ・ボックスの［**オプション**］をクリックします。記録オプションのツリーが表示されたダイアログ・ボックスが開きます。［COM/DCOM］の［**フィルタ**］ノードを選択します。

［環境］サブツリーをクリックすると，**ADO** オブジェクト，**RDS** オブジェクト，および **Remote** オブジェクトが一覧表示されます。フィルタには，［**Type Libraries**］ツリー（最初は空）も含まれています。タイプ・ライブラリは，次の手順で追加できます。

標準設定では，［環境］のすべての項目が選択されており，それらのオブジェクトへの呼び出しがフィルタに含められています（記録対象になります）。これらのオブジェクトをフィルタから除外するには，［**ADO objects**］，［**RDS objects**］，［**Remote objects**］の横にあるチェック・ボックスをクリアします。

2 別の COM タイプ・ライブラリを追加するには，［**追加**］をクリックし，［レジストリの参照］，［ファイル システムの参照］，［MTS の参照］のいずれかを選択します。

3 ［**レジストリの参照**］を選択すると，ローカル・コンピュータのレジストリに登録されているタイプ・ライブラリのリストが表示されます。



使用するライブラリ（1 つまたは複数）の横のチェック・ボックスを選択し，［**OK**］をクリックします。

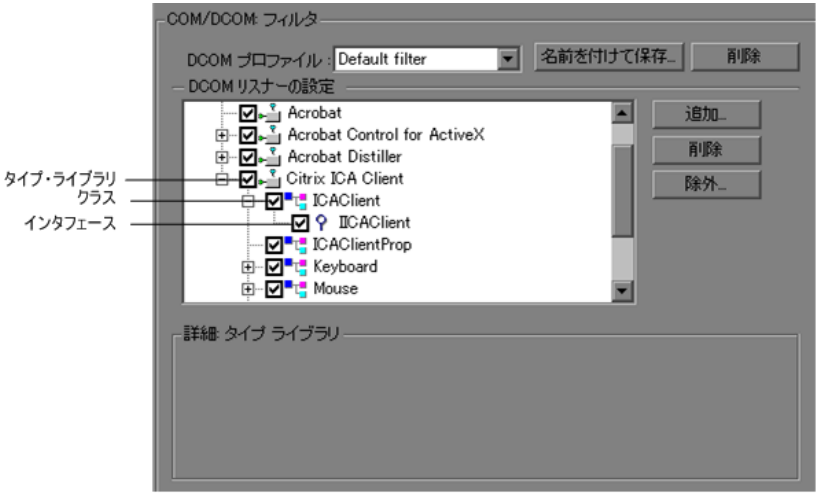
4 ファイル・システムからタイプ・ライブラリを追加するには，［**追加**］をクリックして［**ファイル システムの参照**］を選択します。

使用するファイルを選択して，［**OK**］をクリックします。

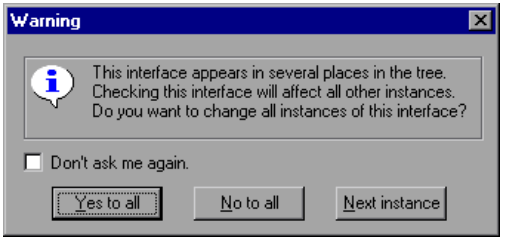
5 タイプ・ライブラリが［タイプ ライブラリ］リストに表示されたら，そのツリーの分岐を開いて，タイプ・ライブラリ内の使用可能なクラスをすべて表示できます。クラス・ツリーの分岐を開いて，そのクラスによってサポートされているインタフェースをすべて表示できます。

タイプ・ライブラリを除外するには，ライブラリ名の横のチェック・ボックスをクリアします。これによって，そのタイプ・ライブラリに含まれているすべてのクラスが除外されます。ツリーの分岐を開き，各クラスまたはインタフェースの横のチェック・ボックスをクリアすることによって，それらの項目を個別に除外できます。

インタフェースの横のチェック・ボックスをクリアすると，[除外インタフェース] ダイアログ・ボックスでそのインタフェースを選択したときと同じ結果が得られます。

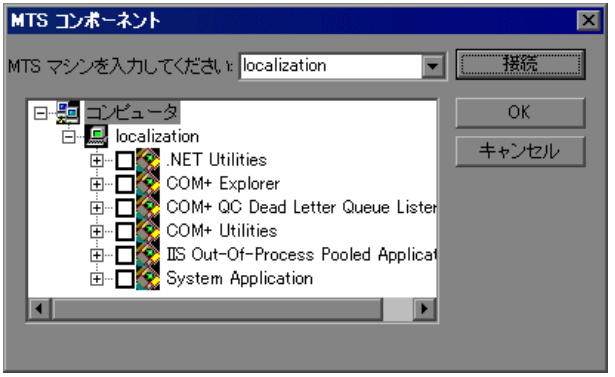


6 クラスの種類ごとに，異なるインタフェースを実装できます。除外されていないほかのクラスにも実装されているインタフェースを除外しようとすると，VuGen によって次の警告が表示されます。



［**Don't Ask me again**］をチェック・ボックスを選択してこのダイアログ・ボックスを閉じると，それ以後，このフィルタを使用し 1 つのオブジェクトに含まれるインタフェースのステータスを変更するたびに，ほかのクラスに含まれているそのインタフェースのステータスもすべて自動的に変更されます。ほかのクラスにあるそのインタフェースのステータスもすべて変更するには，［**Yes to all**］をクリックします。その他のクラスにあるそのインタフェースのステータスを変更しないときには，［**No to all**］をクリックします。そのインタフェースを使用する次のクラスを表示するには，［**Next Instance**］をクリックします。

7 Microsoft Transaction Server のコンポーネントを追加するには，［**追加**］をクリックして［**MTS の参照**］を選択します。MTS サーバの名前を入力するための［MTS コンポーネント］ダイアログ・ボックスが表示されます。

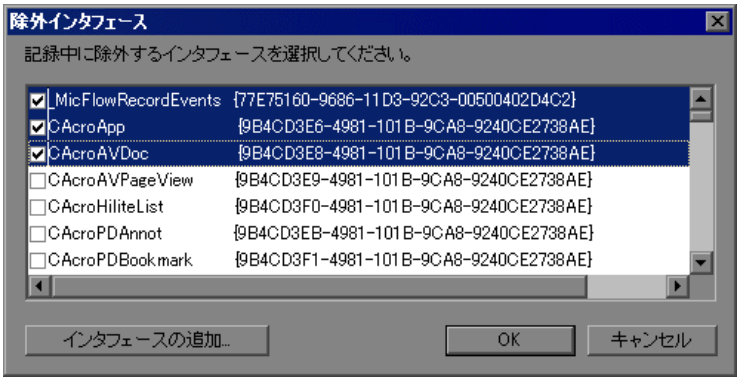


MTS サーバの名前を入力して［**接続**］をクリックします。MTS のコンポーネントを記録するには，マシンに MTS クライアントをインストールしておく必要があります。

使用可能なパッケージのリストから MTS コンポーネントのパッケージを 1 つまたは複数選択して，［**追加**］をクリックします。パッケージが［タイプ ライブラリ］のリストに表示されたら，パッケージから特定のコンポーネントを選択します。

8 ツリー表示で各インタフェースの記録を無効にしたり有効にしたりする以外に，［記録オプション］ダイアログ・ボックスの［**除外**］をクリックして，フィルタにインタフェースを含めたり，フィルタから除外したりできます（どのオブジェクトのインタフェースかに関係なく変更できます）。

タイプ・ライブラリのツリー階層で，クラスとインタフェースの横のチェック・ボックスをクリアして，対応する項目を除外することもできます。



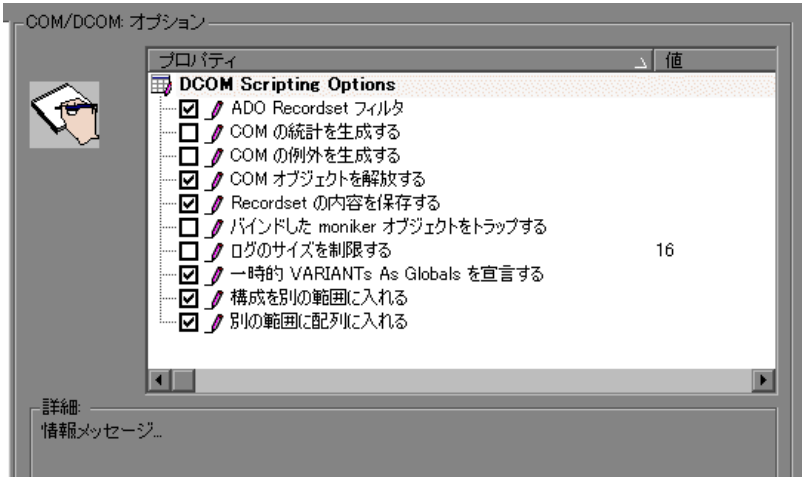
［除外インタフェース］ダイアログ・ボックスでチェックの印が付いているインタフェースが除外されます。表示されていないインタフェースを追加することもできます。［除外インタフェース］ダイアログ・ボックスで［**インタフェースの追加 ...**］をクリックし，GUID 番号（インタフェース ID）とインタフェース名を入力します。VuGen によって作成され，VuGen 画面の左側のカラムの選択ツリーに表示されている interfaces.h ファイルから，GUID をコピーすることもできます。［**インタフェースの追加 ...**］は，スクリプトによって不必要に呼び出されたものの，フィルタに表示されないインタフェースを除外するために使います。

9 フィルタを変更した場合は，［**OK**］をクリックして保存してから，ダイアログ・ボックスを閉じます。新しいフィルタを保存するとき，または既存のフィルタを新しい名前で保存するときは，［**名前を付けて保存**］をクリックします。保存したフィルタは以降の記録で選択できます。標準設定は，［**Default filter**］で表示できます。

COM スクリプト編集オプションの設定

COM の記録セッションの追加オプションを，オブジェクトの処理，ログの生成，VARIANT 定義に関して設定できます。

DCOM スクリプト編集オプションはすべてのプログラミング言語に適用されます。これらのオプションにより，DCOM メソッドおよびインタフェースの処理に関するスクリプトのオプションを設定できます。



- ［**ADO Recordset フィルタ**］: 複数のレコードセットの処理を，1 行の fetch ステートメントにまとめます（標準設定では有効）。
- ［**一時的 VARIANTs As Globals を宣言する**］: 一時 VARIANT をローカル変数としてではなく，グローバル変数として定義します（標準設定では有効）。
- ［**別の範囲に配列を入れる**］: 各配列を独立のスコープに入力します（標準設定では有効）。
- ［**構成を別の範囲に入れる**］: 各構造体を独立のスコープに入力します（標準設定では有効）。
- ［**COM の例外を生成する**］: 記録中に例外が生成された COM 関数およびメソッドを生成します（標準設定では無効）。
- ［**COM の統計を生成する**］: 記録時のパフォーマンスの統計とサマリ情報を生成します（標準設定では無効）。
- ［**ログのサイズを制限する**］: COM 呼び出しごとのセーフ配列のログに出力される要素の数を 16 に制限します（標準設定では有効）。
- ［**COM オブジェクトを解放する**］: 使用されなくなった COM オブジェクトの解放を記録します（標準設定では有効）。

➤［**Recordset の内容を保存する**］：レコードセットの内容を，後に VuGen で表示できるように記録中にグリッドとして保存します（標準設定では有効）。

➤［**バインドした moniker オブジェクトをトラップする**］：バインドされているモニカ・オブジェクトをすべてトラップします（標準設定では無効）。

COM スクリプトのオプションを設定するには，次の手順を実行します。

1 メイン・メニューの［**ツール**］＞［**記録オプション**］を選択するか，［記録開始］ダイアログ・ボックスの［**オプション**］をクリックします。記録オプションのツリーが表示されます。［COM/DCOM］オプションの［**オプション**］ノードを選択します。

2 オプション名の隣のチェック・ボックスをクリックしてオプションを選択します。

3［**OK**］をクリックして設定を保存し，終了します。

第 29 章

COM の概要と相関

本章では，VuGen が COM クライアントの通信を記録して生成するスクリプト，その関数呼び出し，および使用例について詳しく説明します。COM Vuser スクリプトを使った作業を開始する際に知っておくべき基本的な情報については，第 28 章「COM プロトコル」を参照してください。

本章の内容

COM Vuser スクリプトについて

VuGen では，COM Vuser スクリプトごとに次のものが作成されます。

- interfaces.h ファイルに，インタフェース・ポインタとその他の宣言
- vuser_init，Actions，vuser_end の各セクションに，記録可能な関数呼び出し
- user.h ファイルに，下位レベルの呼び出しに変換された Vuser スクリプトのコード

COM クライアントの通信を記録すると，COM API 関数およびインタフェース・メソッドへの呼び出しを含むスクリプトが作成されます。このスクリプトに，COM タイプの変換関数を手作業でプログラミングすることもできます。

スクリプトを記録すると，VuGen 画面の左側のツリーでこれらのファイルを選択して表示できるようになります。

各 VuGen COM 関数呼び出しには，**lrc** というプレフィックスが付いています（**lrc_CoCreateInstance** や **lrc_long** など）。lrc 関数は，インスタンスの作成，IDispatch インタフェース呼び出し，文字列からの型変換，バリアントへの代入，新規バリアントの作成，パラメータ化，バリアントからの取り出し，配列型，ADO レコードセット，バイト配列，VB コレクションのサポート，およびデバッグ関数のカテゴリに分類されます。

lrc 関数の構文と例については，『**Online Function Reference**』（英語版）（［**ヘルプ**］＞［**関数リファレンス**］）を参照してください。

Vuser スクリプトの作成に使用するプログラミング言語（C または Visual Basic）を指定できます。詳細については，『**第 1 巻 － VuGen の使用**』の「スクリプト生成オプションの設定」を参照してください。

VuGen COM スクリプトの構造について

VuGen COM スクリプトは，COM インタフェースの要件を満たすために特別な方法で構成されています。

インタフェース・メソッド

インタフェース・メソッドへの呼び出しの名前と構文の形式は，次のとおりです。

```
lrc_ ＜インタフェース名＞ _ ＜メソッド名＞ (instance, ...);
```

instance は常に，最初に渡されるパラメータです。

一般的に，インタフェース関数に関するドキュメントは各 COM コンポーネントのベンダから提供されます。

インタフェース・ポインタ

interfaces ヘッダ・ファイルは，スクリプト内で使用されるインタフェース・ポインタとその他の変数を定義できます。各インタフェースには，そのインタフェースを一意に識別するインタフェース ID（IID）が割り当てられています。

インタフェースの定義の形式は，次のとおりです。

```
＜インタフェース・タイプ＞ * ＜インタフェース名＞ = 0; //"{ ＜インタフェース・タイプの IID ＞ }"
```

次の例では，インタフェース・タイプは IDispatch，インタフェースのインスタンスの名前は IDispatch_0，IDispatch タイプの IID は long 型の文字列です。

```
IDispatch* IDispatch_0= 0;//"{00020400-0000-0000-C000-000000000046}"
```

Vuser スクリプトのステートメント

COM Vuser スクリプトは，オブジェクトのインスタンスの作成，インタフェース・ポインタの取得，インタフェース・メソッドの呼び出しを行うコードで構成されます。各ユーザ・アクションは，1 つまたは複数の COM 呼び出しを生成します。COM 呼び出しはそれぞれ，VuGen によってステートメント・グループとして記述されます。グループは，それぞれが独立したスコープとして括弧で囲まれます。いくつかのステートメントによって，値の代入と型変換を行うことによって，main の呼び出しに備えます。オブジェクトの作成に必要な呼び出しのグループの例を次に示します。

```
{
GUID pClsid = lrc_GUID("student.student.1");
IUnknown * pUnkOuter = (IUnknown*)NULL;
unsigned long dwClsContext = lrc_ulong("7");
GUID riid = IID_IUnknown;
lrc_CoCreateInstance(&pClsid, pUnkOuter, dwClsContext, &riid, (void**)&IUnknown_0,
CHECK_HRES);
}
```

エラーの検査

各 COM メソッドまたは API 呼び出しは，エラーの値を返します。VuGen によって，最初の記録時に呼び出しが成功したかどうかに応じて，再生時にエラーを検査するかどうかを示すフラグが設定されます。フラグは関数呼び出しの最後の引数として指定されます。フラグの値は次のいずれかです。

CHECK_HRES	記録時に関数が正しく実行されたため，再生時にエラーを検査する必要がある場合は，この値が挿入されます。
DONT_CHECK_HRES **DONT_CHECK_HRES**	記録時に関数が失敗したため，再生時にエラーを検査する必要がない場合は，この値が挿入されます。

VuGen COM スクリプトの検証

本項では，VuGen が COM クライアント・アプリケーションをエミュレートする仕組みを，例を使って説明します。

クエリの結果はグリッドに表示されます。グリッドをスクロールすれば，最高 200 レコードまで見ることができます。詳細については，370 ページ「グリッドを使った作業」を参照してください。

COM スクリプトが行う基本的な処理

基本的な処理として，次のことを行います。

- オブジェクトのインスタンスの作成
- インタフェース・ポインタの取得
- インタフェース・メソッドの呼び出し

それぞれの処理は，独立のスコープで実行されます。

オブジェクトのインスタンスの作成

アプリケーションは，COM オブジェクトを使用するために，最初にインスタンスを作成し，そのオブジェクトのインタフェースへのポインタを取得します。

VuGen では，次の手順でオブジェクトのインスタンスが作成されます。

1 VuGen によって lrc_GUID が呼び出され，そのオブジェクト用の一意の ProgID が取得されて pClsid に格納されます。

```
GUID pClsid = lrc_GUID("student.student.1");
```

pClsid は，ProgID の「**student.student.1**」から変換された，オブジェクトの一意のグローバル CLSID です。

2 IUnknown インタフェース・ポインタが，集約オブジェクトへのポインタである場合，VuGen によってそのオブジェクトへのポインタが取得されます。取得できない場合は，VuGen によってポインタが NULL に設定されます。

```
IUnknown * pUnkOuter = (IUnknown*)NULL;
```

3 VuGen によって，作成するオブジェクトのコンテキストが設定されます。

```
unsigned long dwClsContext = lrc_ulong("7");
```

dwClsContext には，オブジェクトのコンテキストが含まれます（プロセス，ローカル，リモート，またはこれらのうちの複数の場所）。

4 VuGen によって，要求されたインタフェース ID を保持する変数が設定されます。この例では，IUnknown です。

```
GUID riid = IID_IUnknown;
```

riid には，**IUnknown** インタフェースのインタフェース ID が含まれます。

5 入力パラメータが用意された後に，lrc_CoCreateInstance への呼び出しにより，用意されたパラメータを使ってオブジェクトが作成されます。IUnknown インタフェースへのポインタがパラメータ IUnknown_0 に割り当てられます。このポインタは，次の呼び出しに必要です。

```
lrc_CoCreateInstance(&pClsid, pUnkOuter, dwClsContext, &riid, (void**)&IUnknown_0,
CHECK_HRES);
```

前述のとおり入力パラメータが用意されています。呼び出しが成功しているため，VuGen によって **CHECK_HRES** 値が挿入され，ユーザのシミュレーションの実行時にエラーの検査を行うように設定されます。呼び出しの結果，**IUnknown_0** に IUnknown インタフェースへのポインタが返されます。IUnknown_0 は，以降の呼び出しで使用されます。

インタフェースの取得

オブジェクトを作成した時点で VuGen がアクセスできるのは **IUnknown** インタフェースだけです。VuGen は，オブジェクトと通信するために **IUnknown** インタフェースを使用します。これは，**IUnknown** 標準インタフェースの **QueryInterface** メソッドを使って行われます。VuGen のメソッド呼び出しの最初のパラメータは，インタフェースのインスタンスです。この例では，最初のパラメータは事前に **CoCreateInstance** によって設定された **IUnknown_0** ポインタです。**QueryInterface** 呼び出しには，取得すべきインタフェースの ID が入力項目として必要です。QueryInterface 呼び出しによって，指定した ID に対応するインタフェースへのポインタが返されます。

インタフェースを取得するには，次の手順を実行します。

1 最初に VuGen によって Istudent インタフェースの ID のパラメータである riid が設定されます。

```
GUID riid = IID_Istudent;
```

2 QueryInterface を呼び出すと，Istudent オブジェクトに Istudent インタフェースがあれば，出力パラメータ Istudent_0 にそのインタフェースへのポインタが割り当てられます。

```
lrc_IUnknown_QueryInterface(IUnknown_0, &riid, (void**)&Istudent_0,
CHECK_HRES);
```

インタフェースを使ったデータの設定

インタフェースのメソッドを使って，データを設定する例を以下に示します。たとえば，アプリケーションでユーザが名前を入力するように求められるとします。この場合，名前を設定するためのメソッドが起動されます。VuGen によって 2 つのステートメントが記録されます。1 つは，名前の文字列を組み立てるために使用され，もう 1 つは名前のプロパティを設定します。

この関数呼び出し全体を組み立てるには，次の手順を実行します。

1 最初に，VuGen によって変数（Prop Value）に，入力された文字列と同じ値が設定されます。パラメータのタイプは，COM ファイルで使用される文字列型である BSTR です。

```
BSTR PropValue = lrc_BSTR("John Smith");
```

後で，「John Smith」をパラメータで置き換えることによって，この呼び出しをパラメータ化すると便利です。これによって，Vuser スクリプトを実行するたびに，異なる名前を使用できるようになります。

2 次に，VuGen は名前を入力するための，Istudent インタフェースの Put_Name メソッドを呼び出します。

```
lrc_Istudent_put_name(Istudent_0, PropValue, CHECK_HRES);
```

インタフェースを使ったデータの返却

値を格納して，以降の呼び出しで入力項目として使用できるようにパラメータ化を行いたい場合は，データを入力するだけでは不十分で，アプリケーションからデータを返さなければなりません。

アプリケーションがデータを取得したときに VuGen によって何が行われるかを次の例に示します。

1 プロパティの値を格納する適切なタイプの変数（この例では BSTR）を作成します。

```
BSTR pVal;
```

2 前述の手順で作成した変数 **pVal** に，プロパティの値（この例では名前）を保存します。この例では，**Istudent** の get_name メソッドを使用します。

```
lrc_Istudent_get_name(Istudent_0, &pVal, CHECK_HRES);
```

3 次に，VuGen によってこれらの値を保存するためのステートメントが生成されます。

```
//lrc_save_BSTR("param-name",pVal);
```

このステートメントは，コメントアウトされています。コメントを表す記号（//）を削除して，< param-name > を変数に変更できます。変数には，この値を格納することを表すような名前を付けることができます。VuGen によって，直前の呼び出しによって返された **pVal** の値を保存するために，この変数が使用されます。以降，パラメータ化した入力項目として，ほかのメソッドへの呼び出しでこの変数を使用できます。

IDispatch インタフェース

ほとんどの COM オブジェクトには，固有のインタフェースがあります。また多くは，汎用インタフェース **IDispatch** も実装しています。このインタフェースは VuGen によって特別な方法で変換されます。IDispatch は，IDispatch 以外の COM オブジェクト・インタフェースおよびメソッドをすべて公開する「特別なインタフェース」です。VuGen スクリプトからの **IDispatch:Invoke** メソッドへの呼び出しは，**lrc_Disp** 関数を使って実装されます。これらの呼び出しは，ほかのインタフェースへの呼び出しとは異なる形式で構成されます。

IDispatch インタフェースの **Invoke** メソッドは，メソッドの実行，プロパティ値の取得，プロパティの値またはプロパティの参照の値の設定を行うことができます。標準の **IDispatch:Invoke** メソッドでは，これらのさまざまな用途は **wflags** パラメータで示されます。VuGen では，これらはメソッドの呼び出し，またはプロパティの設定や取得を行う別々のプロシージャ呼び出しとして実装されます。

たとえば，GetAgentsArray メソッドを呼び出すための IDispatch への呼び出しは，次のようになります。

```
retValue = lrc_DispMethod1((IDispatch*)IDispatch_0, "GetAgentsArray", /*locale*/1033,
LAST_ARG, CHECK_HRES);
```

前述の呼び出しのパラメータは，次のとおりです。

IDispatch_0	以前に実行された **IUnknown:Queryinterface** メソッドへの呼び出しによって返された **IDispatch** インタフェースへのポインタです。
GetAgentsArray	呼び出すメソッドの名前です。VuGen の実際の動作では，この名前からメソッドの ID が取得されます。
1033	言語ロケールです。

LAST_ARG	引数がこれ以上ないことを **IDispatch** インタフェースに知らせるためのフラグです。
CHECK_HRES	記録時に呼び出しが成功したため，HRES の検査を行うことを示すフラグです。

さらに，OPTIONAL_ARGS という別のパラメータが存在することもあります。これは，VuGen によって標準パラメータ以外に追加引数が送られていることを示します。追加引数はそれぞれ，ID または名前と，その値の組み合わせで構成されています。たとえば，lrc_DispMethod への次の呼び出しは，任意の引数「#3」および「var3」を渡します。

```
{
        GUID riid = IID_IDispatch;
        lrc_IOptional_QueryInterface(IOptional_0, &riid, (void**)&IOptional_0,
CHECK_HRES);
    }
    {
        VARIANT P1 = lrc_variant_short("47");
        VARIANT P2 = lrc_variant_short("37");
        VARIANT P3 = lrc_variant_date("3/19/1901");
        VARIANT var3 = lrc_variant_scode("4");
        lrc_DispMethod((IDispatch*)IOptional_0, "in_out_optional_args", /*locale*/1024,
&P1, &P2, OPTIONAL_ARGS, "#3", &P3, "var3", &var3, LAST_ARG, CHECK_HRES);
}
```

IDispatch インタフェースを使用するさまざまな **lrc_Disp** メソッドの詳細については，『**Online Function Reference**』（英語版）に記載されています。

型変換とデータ抽出

上の例で示したように，COM パラメータの多くはバリアントとして定義されます。これらの値を抽出するために，COM 関数を基に作成したいくつかの変換関数が VuGen によって使用されます。変換関数の一覧については『**Online Function Reference**』（英語版）を参照してください。**lrc_DispMethod1** 呼び出しを使って，名前の文字列の配列を取得する方法については，すでに説明しました（次の例を参照）。

```
VARIANT retValue = lrc_variant_empty();
retValue = lrc_DispMethod1((IDispatch*)IDispatch_0, "GetAgentsArray", /*locale*/1033,
LAST_ARG, CHECK_HRES);
```

次の例では，VuGen で文字列の配列として読み取られるバリアントである **retValue** から文字列を取り出す方法を示します。

まず，VuGen によってバリアントから BSTR 配列が抽出されます。

```
BstrArray array0 = 0;
array0 = lrc_GetBstrArrayFromVariant(&retValue);
```

array0 内にすべての値が含まれている状態で，後でパラメータ化の際に使用できるように，VuGen によって配列の要素を抽出するためのコードが生成されます。次に例を示します。

```
//GetElementFrom1DBstrArray(array0, 0); // 値：Alex
//GetElementFrom1DBstrArray(array0, 1); // 値：Amanda
....
```

VuGen には，さまざまなタイプの変換関数や，バリアントから従来の型を抽出するための関数があります。これらの詳細については，『**Online Function Reference**』（英語版）（［**ヘルプ**］＞［**関数リファレンス**］）を参照してください。

スクリプト内での相関候補の検索

VuGen には，スクリプトを修正して確実な再生を支援する相関ユーティリティがあります。相関ユーティリティは，次の処理を行います。

- 相関候補を探す。
- 適切な相関関数を挿入して，結果をパラメータに保存する。
- ステートメントの値をパラメータで置き換える。

自動相関は，スクリプト全体またはスクリプト内の特定の場所を対象に実行できます。

本項では，相関させる必要のあるステートメントを見つける方法について説明します。すでに相関させたい値がわかっている場合は，次の項に進んで，特定の値を相関させる方法を参照してください。

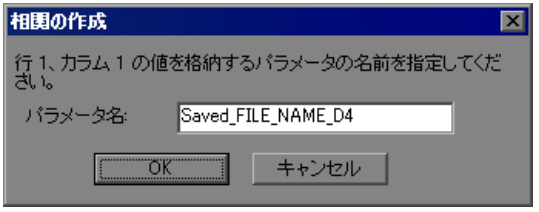
自動相関で検出されたスクリプトの検索と相関を行うには，次の手順を実行します。

1 ［**表示**］＞［**出力ウィンドウ**］を選択して，ウィンドウの下部に出力タブを表示します。［再生ログ］タブでエラーがないか確認します。多くの場合，これらのエラーは相関によって修正できます。

2 ［**仮想ユーザ**］＞［**相関を検索**］を選択します。VuGen によってスクリプト全体を対象とする検索が行われ，相関候補の値がすべて［**相関クエリー**］タブに表示されます。

3 値を相関させます。［相関クエリー］タブで，相関させるクエリ結果をダブルクリックします。この例では，メッセージの「grid column x, row x」となっている行が該当します。VuGen によって，スクリプト内の値の位置にカーソルが移動します。

4 グリッドの中で，右クリック・メニューから［**相関を作成**］を選択します。結果の値を保存するパラメータの名前を入力するように求められます。



5 名前を指定するか，標準設定の名前をそのまま使用します。［**OK**］をクリックして作業を続けます。VuGen によって，パラメータに結果を保存する相関ステートメント（**lrc_save_** ＜タイプ＞）が挿入されます。

6 ［**はい**］をクリックして相関を確定します。

7 スクリプト内に現れる値をすべて検索するかどうかを尋ねるメッセージ・ボックスが開きます。選択したステートメント内の値だけを置き換える場合は，［**いいえ**］をクリックします。次の候補を検索して置き換える場合は［**はい**］をクリックします。

8 ［検索と置換］ダイアログ・ボックスが開きます。オリジナルのステートメントも含め，すべての置き換えを確認します。

9 ［検索と置換］ダイアログ・ボックスを閉じます。ステートメントの値がパラメータへの参照に置き換えられます。相関を取り消すと，直前のステップで作成されたステートメントは消去されます。

既知の値の相関

相関させるべき値がわかっている場合は次の手続きを行います。

特定の値を相関させるには，次の手順を実行します。

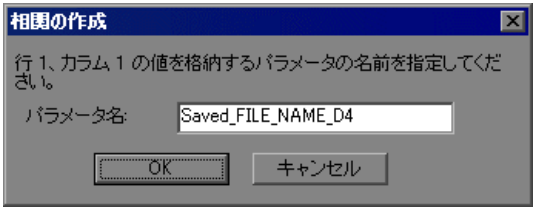
1 相関させる引数を見つけ（通常は **lrc_variant_** ステートメント内にあります），値を選択します（引用符は除きます）。

2 ［**仮想ユーザ**］＞［**アクション内で相関をスキャン**］を選択します。

VuGen によって値が検索され，この値と一致するスクリプト内の結果がすべて表示されます。相関値は［相関クエリー］タブに一覧表示されます。

3 ［相関クエリー］タブで，相関させるクエリ結果をダブルクリックします。この例では，メッセージの「grid column x, row x」となっている行が該当します。VuGen によって，スクリプト内の値の位置にカーソルが移動します。

4 表示枠の中で，相関させる値を選び，［**仮想ユーザ**］＞［**相関を作成**］を選択します。結果の値を保存するパラメータの名前を入力するように求められます。



5 名前を指定するか，標準設定の名前をそのまま使用します。［**OK**］をクリックして作業を続けます。VuGen によって，パラメータに結果を保存する相関ステートメント（lrc_save_ ＜タイプ＞）が挿入されます。

```
lrc_save_rs_param (Recordset20_0, 1, 1, 0, "Saved_AGENT_NAME");
```

6 ［**はい**］をクリックして相関を確定します。

7 スクリプト内に現れる値をすべて検索するかどうかを尋ねるメッセージ・ボックスが開きます。選択したステートメント内の値だけを置き換える場合は，［**いいえ**］をクリックします。次の候補を検索して置き換える場合は［**はい**］をクリックします。

8 ［検索と置換］ダイアログ・ボックスが開きます。オリジナルのステートメントも含め，すべての置き換えを確認します。

第 30 章

AJAX（Click and Script）プロトコル

VuGen では，AJAX（Asynchronous JavaScript and XML）対応のアプリケーションをエミュレートするスクリプトを作成できます。

本章の内容

AJAX（Click and Script）Vuser スクリプトの作成について

AJAX（Asynchronous JavaScript and XML）は，対話型の Web アプリケーションを作成する手法です。AJAX では，Web ページで，ページ全体をリロードする代わりにサーバと小さなデータ・パケットが交換されます。これにより，データ要求時のユーザの待ち時間が短くなります。また，対話機能や使いやすさが向上します。

AJAX を使用して，開発者は Java スクリプトと非同期サーバ要求を使用する高速の Web ページを作成できます。要求は，ユーザ・アクション，タイマー・イベント，またはほかのあらかじめ定義されたトリガから生成できます。

AJAX コントロールともよばれる AJAX コンポーネントは，AJAX テクニックを使用する GUI ベースのコントロールで，トリガが発生すると要求をサーバに送信します。

たとえば，一般的な AJAX コントロールは，コンポーネントをリスト内の目的の位置にドラッグできる **Reorder List** コントロールです。VuGen の AJAX 実装のサポートは，Microsoft の ASP.NET AJAX Control Toolkit（以前の Atlas）に基づいています。

AJAX 関数に対してサポートされているフレームワークは次のとおりです。

- Atlas 1.0.10920.0/ASP.NET AJAX：すべてのコントロール。
- Scriptaculous 1.8：Autocomplete，Reorder List，および Slider。

VuGen では，エンジン・レベルで次のフレームワークがサポートされています。これは，VuGen は標準の Web（Click and Script）ステップを作成しますが，次の特定の AJAX 関数は作成しないことを示します。

- Prototype 1.6
- Google Web Toolkit（GWT）1.4

AJAX（Click and Script）セッションの記録

AJAX 対応のアプリケーションをエミュレートする Vuser スクリプトを作成するには，［**e ビジネス**］カテゴリから［**AJAX (Click and Script)**］プロトコル・タイプを選択します。記録を開始するには，［**記録**］ボタンをクリックし，リストの記録など AJAX の影響を受ける標準的なアクションを Web ページ上で実行します。スクリプトの作成および記録の詳細については，『**第 1 巻 － VuGen の使用**』の「VuGen を使った記録」を参照してください。

イベントに関する記録オプションを設定できます。詳細については，『**第 1 巻 － VuGen の使用**』の第 17 章「Click and Script 記録」を参照してください。

スクリプトを作成した後，スクリプトを自分の環境（LoadRunner シナリオ，Performance Center 負荷テスト，または Business Process Monitor プロファイル）に組み込みます。詳細については，**HP LoadRunner Controller**，**HP Performance Center**，または **HP Business Availibility Center** のドキュメントを参照してください。

AJAX（Click and Script）プロトコルについて

VuGen はコントロール・ハンドラ・レイヤを使用して，GUI コントロールに対する操作に影響を与えます。記録中，いずれかのサポートされている AJAX コントロールが出現した場合，VuGen は **ajax_xxx** というプレフィックスを持つ関数を生成します。

次の例では，ユーザは Accordion コントロールで項目番号 1（index=1）を選択しています。VuGen は **ajax_accordion** 関数を生成しています。

```
web_browser("Accordion.aspx",
                DESCRIPTION,
                ACTION,
                "Navigate=http://labm1app08/AJAX/Accordion/Accordion.aspx",
                LAST);

lr_think_time(5);

ajax_accordion("Accordion",
                DESCRIPTION,
                "Framework=atlas",
                "ID=ctl00_SampleContent_MyAccordion",
                ACTION,
                "UserAction=SelectIndex",
                "Index=1",
                LAST);

web_edit_field("free_text_2",
        "Snapshot=t18.inf",
        DESCRIPTION,
        "Type=text",
        "Name=free_text",
        ACTION,
        "SetValue=FILE_PATH",
        LAST);
```

注：AJAX セッションを記録すると，VuGen はサポートされている AJAX コントロール以外のオブジェクトに対して標準 Web（Click and Script）関数を生成します。上の例では，文字列 FILE_PATH がエディット・ボックスに入力されました。

第 31 章

AMF プロトコル

VuGen では，AMF 形式を使用する Flash Remoting をエミュレートする Vuser を作成できます。

本章の内容

AMF Vuser スクリプトの作成について

多くのクライアント・アプリケーションは，RPC（リモート・プロシージャ・コール）を使用してサーバと通信します。しかし，RPC では，インターネットを介して作業する場合，互換性およびセキュリティ上の問題が生じます。たいていの場合，ファイアウォールやプロキシ・サーバは，このタイプのトラフィックをブロックします。

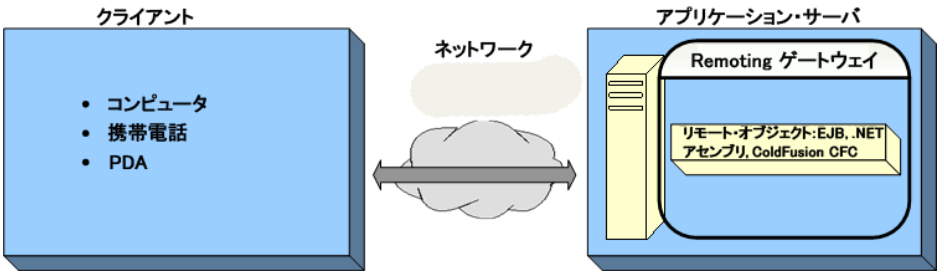
HTTP は，すべてのインターネット・ブラウザおよびサーバでサポートされます。したがって，インターネット経由で作業する場合，HTTP は，クライアント・アプリケーションとサーバ間の通信における好ましい方法であるといえます。

SOAP（XML ベースの形式）は，HTTP を使用してアプリケーション間の通信を行うための安全な方法を提供します。しかし，メッセージがテキスト・ベースであるため，Flash ファイルなどの大きなメッセージやほかの RIA（Rich Internet Application）を使用して作業する場合，SOAP では非効率的です。

この非効率性を解決するため，Macromedia は，HTTP を使用してバイナリ形式で通信する独自のプロトコル，すなわち AMF（Action Messaging Format）を開発しました。バイナリ形式の AMF データ・セットは，SOAP のテキスト・ベースの XML よりもサイズが大幅に小さくなります。

サーバに AMF メッセージを送信する代表的なクライアント・アプリケーションの例としては，パーソナル・コンピュータ上で Flash クリップを再生する Flash Player があります。Flash Player は，ゲートウェイを経由してアプリケーション・サーバにネイティブな Flash オブジェクトを送信します。ゲートウェイ（**Flash Remoting** ゲートウェイとも呼ばれます）は，Java（ColdFusion を含む）または .NET サーバにインストールされているサーバ側オブジェクトです。ゲートウェイは，Flash Player とサーバの間で要求を処理するブローカとして機能します。Flash オブジェクトをサーバのネイティブ・オブジェクトに変換し，それらをサーバ側の適切なサービスに渡します。

結果が返されると，ゲートウェイはその結果をシリアル化してネイティブ Flash オブジェクトに変換し，AMF を使用して Flash クライアントに送ります。



AMF トラフィックをエミュレートする Vuser スクリプトを作成するには，［**e ビジネス**］カテゴリから［**AMF**］プロトコル・タイプを選択します。記録を開始するには，［**記録**］ボタンをクリックし，Web サーバに対する標準的なアクションを実行します。スクリプトの作成および記録の詳細については，『**第 1 巻 ー VuGen の使用**』の「VuGen を使った記録」を参照してください。

スクリプトを作成した後，スクリプトを自分の環境（LoadRunner シナリオ，Performance Center 負荷テスト，または Business Process Monitor プロファイル）に組み込みます。詳細については，**HP LoadRunner Controller**，**HP Performance Center**，または **HP Business Availibility Center** のドキュメントを参照してください。

AMF の用語について

次の表に，AMF に関係する，よく使用される用語の定義を示します。

用語 / 略語	説明
ActionScript	Flash ムービーおよびアプリケーションの制御に使用されるスクリプト・プログラミング言語です。この構文は JavaScript に似ています。
AMF	Flash Remoting に使用される独自のバイナリ通信プロトコルです。
Flash Remoting	Flash Remoting では，AMF 形式を使用して，Flash Player とアプリケーション・サーバ間でデータを交換できます。
Flex	RIA（Rich Internet Application）を生成するためのアプリケーション・サーバです。
SOAP	通常 HTTP を使用し，コンピュータ・ネットワーク経由で XML ベースのメッセージを交換するための標準です。

AMF 関数を使った作業

Flash Remoting セッションの記録時に，VuGen はユーザのアクションをエミュレートする AMF Vuser スクリプト関数を生成します。すべての AMF 関数の先頭には，**amf** というプレフィックスが付きます。

次の例では，**amf_define_header_set** 関数がヘッダ・セットを定義しています。**amf_call** 関数はゲートウェイにアクセスし，サーバにメッセージを送信しています。

```
amf_define_header_set("Id=amf_header_set",
    HEADER,
    "Name=amf_server_debug",
    "MustUnderstand=true",
    "Data=<object><boolean key=¥"coldfusion¥">true</boolean>
            <boolean key=¥"""amfheaders¥">false</boolean>...
    LAST);

amf_call("flashgateway.samples.FlashJavaBean.testDocument",
    "Gateway=http://testlab:8200/flashservices/gateway",
    "AMFHeaderSetId=amf_header_set",
    "Snapshot=t3.inf",
    MESSAGE,
    "Method=flashgateway.samples.FlashJavaBean.testDocument",
    "TargetObjectId=/1",
    BEGIN_ARGUMENTS,
    "<xmlString><![CDATA[<TEST message=¥"test¥"><INSIDETEST
    /></TEST>]]></"
    xmlString>",
    END_ARGUMENTS,
    LAST);
```

これらの関数の構文情報の詳細については，『**Online Function Reference**』（英語版）（［**ヘルプ**］ > ［**関数リファレンス**］）を参照してください。

AMF スクリプトの相関

たいていの場合，Flash アプリケーションには，スクリプトを実行するたびに変わる動的なデータが含まれています。たとえば，あるサーバでは，現在の日時で構成されるリンクを使用しています。あるいは，実行のたびにオブジェクト名が変わるかもしれません。

Vuser スクリプトを記録するとき，VuGen はデータと引数値のセットを記録します。しかし，スクリプトの再生時に，サーバがこの引数を拒否し，エラーを発行することがあります。このエラーは，データが古くなってサーバで受け入れられなくなった動的なデータが原因であると考えられます。

これを回避するには，スクリプトの相関を行います。

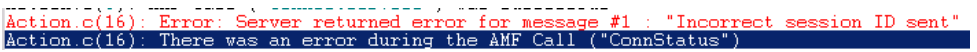
- 必要な値を取り出す準備として，サーバからの応答を保存します。
- サーバの応答から必要な値を取り出します。
- 値をパラメータに保存します。
- それらのパラメータを AMF 要求への入力として使用します。

このようなエラーは必ずしもはっきりわかるわけではなく，Vuser ログ・ファイルを注意深く調べないと検出できないことがあります。Vuser の実行時にエラーが発生した場合は，スクリプト内でのエラーの発生箇所を調べてください。多くの場合，相関によって，1 つのステートメントの結果を別のステートメントの入力項目として使用することで，問題を解決できます。

相関を実行するには，次の手順を実行します。

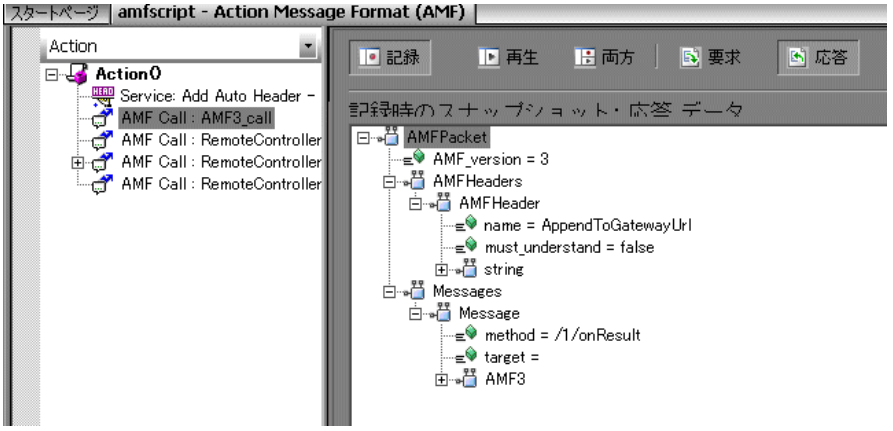
1 動的な値が原因で失敗し，相関が必要となったスクリプト内のステップを探します。

［再生ログ］を利用して問題のステップを探します。



2 先行するステップの中にある，正しい値を持っているサーバ応答を探します。

先行するステップをツリー・ビューの中で調べて，［**サーバ応答**］タブ内で値を探します。



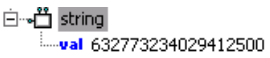
3 サーバ応答全体をパラメータに保存します。

値を取り出す前に，サーバ応答全体をパラメータに次のように保存します。

- 対象の値を含んでいるサーバ応答に対応するステップ・ノード（[アクション］表示枠内）を右クリックし，［**プロパティ**］を選択します。
- ［AMF Call Properties］ダイアログの中で，**応答パラメータ**の名前を入力します。詳細については，491 ページ「[AMF Call Properties]」を参照してください。
- ［**OK**］をクリックして，新しいパラメータ名を保存します。

4 元のサーバ応答値をパラメータに保存します。

- ［サーバ応答］の XML ツリーの中で，値の上のノード（たとえば，string）を右クリックして［**パラメータに XML を保存**］を選択します。



- ［XML パラメータのプロパティ］ダイアログの中で，**Name** にパラメータ名を入力します。以降のステップでは，この名前を使用することになります。

➤ ［**OK**］をクリックします。これで，スクリプトに **lr_xml_get_values** という新しい関数が追加されます。

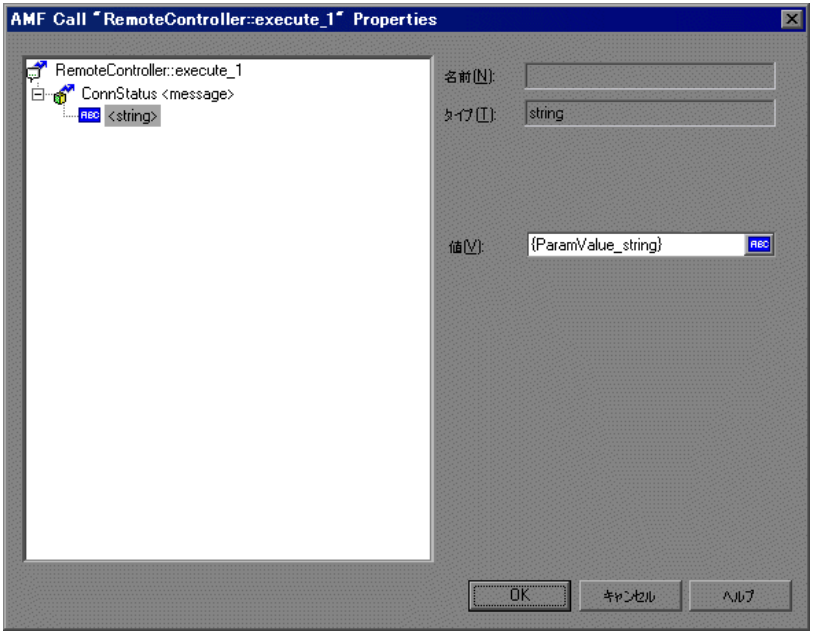
5 パラメータを以降の呼び出しに挿入します。

VuGen の編集ビューで，失敗した呼び出しを開始点に，オブジェクトに対する以降のすべての呼び出しに含まれている値を，定義したパラメータで置き換えます。

➤ 失敗している呼び出しに対応するステップ・ノード（［アクション］表示枠内）を右クリックし，［**プロパティ**］を選択します。

➤ 相関を必要としていた引数を探します。

➤ ［**値**］ボックスに，**{ParamValue_string}** のようにパラメータ名を中括弧で囲んで入力します。



➤ ［**OK**］をクリックします。

6 スクリプトを実行します。

引数値が，保存したパラメータ値で適切に置き換えられていることを確認します。

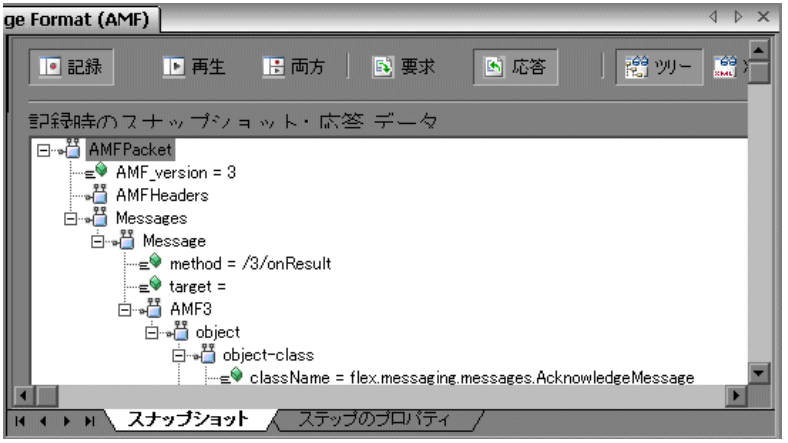
次の例では，**lr_xml_get_values** を使用して応答から値を取得し，ParamValue_string というパラメータを作成しています。この ParamValue_string パラメータは，次の **amf_call** 関数で使用しています。

```
amf_call(
    "ConnectService",
    "Gateway=http://labm1app08/AMF/EchoAMF/gateway.aspx",
    "Snapshot=t6.inf",
    "ResponseParameter=resp",
    MESSAGE,
    "Method=EchoAMF.SpecialCases.ConnectService",
    "TargetObjectId=/1",
    BEGIN_ARGUMENTS,
    END_ARGUMENTS,

    LAST);

lr_xml_get_values("XML={resp}",
    "FastQuery=/AMFPacket/Messages/Message/string",
    "ValueParam=ParamValue_string",
    LAST);

amf_call(
    "ConnStatus",
    "Gateway=http://labm1app08/AMF/EchoAMF/gateway.aspx",
    "Snapshot=t7.inf",
    MESSAGE,
    "Method=EchoAMF.SpecialCases.ConnStatus",
    "TargetObjectId=/2",
    BEGIN_ARGUMENTS,
    "<string>{ParamValue_string}</string>",
    END_ARGUMENTS,

    LAST);
```

Web Vuser の相関の詳細については，第 42 章「Web（HTTP/HTML）の相関ルール」を参照してください。

AMF データの表示

AMF データを XML 形式で表示するには，該当するステップをツリー・ビューの中で選択します。VuGen の右表示枠に，クライアントの要求とサーバの応答が XML 階層で表示されます。実際の XML を表示するには，［**XML**］ボタンをクリックします。

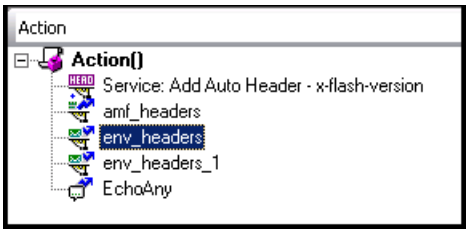


メッセージのすべての要素を表示するには，XML のすべてのノードを確実に展開します。XML 引数値をグリッドに表示するには，［**ノード値をグリッドに表示する**］を選択します。

VuGen には，XPath によって XML を検索するユーティリティが用意されています。また，クエリの作成を支援するクエリ・ビルダを使用することもできます。詳細については，505 ページ「XML ツリーに対するクエリ」を参照してください。

AMF スクリプトについて

次の例では，AMF スクリプトに AMF 呼び出し，AMF ヘッダ，および AMF エンベロープが含まれています。各ノードの詳細を表示するには，ノードを選択し，右クリック・メニューから［プロパティ］を選択します。

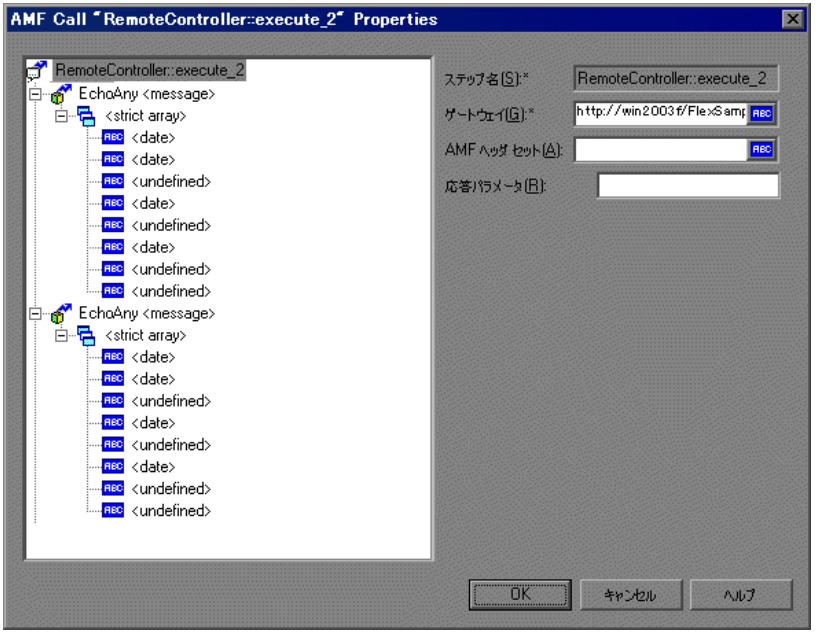


次の各項では，AMF ノードのプロパティについて説明します。

- ［AMF Call Properties］
- AMF ヘッダ・セット・プロパティ
- AMF エンベロープ・ヘッダ・セット・プロパティ

[AMF Call Properties]

[AMF Call Properties] ダイアログ・ボックスには，AMF 呼び出し，1 つ以上のメッセージ，各メッセージの引数を含む，AMF 要求の詳細が表示されます。次の例の中では，AMF 呼び出しは **EchoAny** で，**amf_header** ヘッダ・セットに関連付けられています。

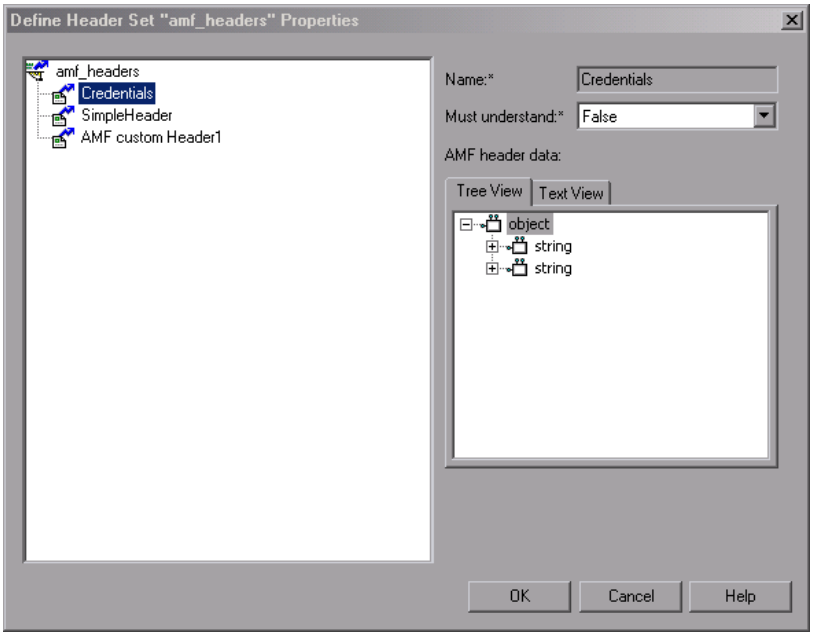


各 AMF 呼び出しには 3 つのレベルがあります。

- 「AMF Call」のレベルには，ステップ名，ゲートウェイ，AMF ヘッダ・セット，および応答パラメータが含まれます。
- 「AMF Message」のレベルには，メソッド，ターゲット・オブジェクト ID，およびエンベロープ・ヘッダ・セットが含まれます。
- 「AMF Argument」のレベルには，名前，タイプ，および値が含まれます。

AMF ヘッダ・セット・プロパティ

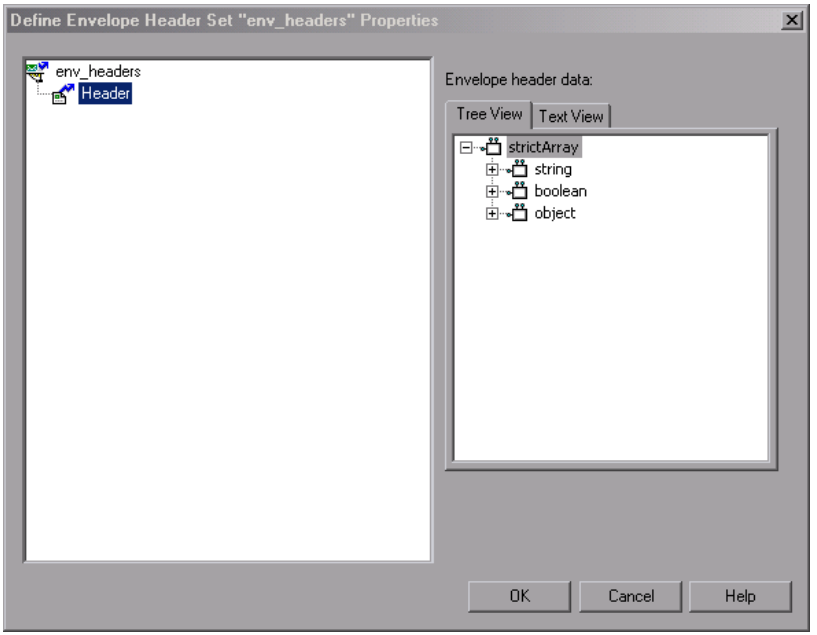
［AMF Header Set Properties］ダイアログ・ボックスには，1 つ以上の AMF ヘッダ名と，関連するヘッダ・データが XML 形式から成る AMF ヘッダ・セットの定義が表示されます。次の例では，ヘッダ・セットに 3 つのヘッダがあります。



それぞれのヘッダには，サーバがヘッダを解釈できなかったときに呼び出しの処理を継続するかどうかを示す **Must understand** 引数があります。MustUnderstand が True の場合，対象ヘッダは必須であり，解釈ができなかった場合には処理が中止されます。

AMF エンベロープ・ヘッダ・セット・プロパティ

［AMF Envelope Header Set Properties］ダイアログ・ボックスには，AMF エンベロープ・ヘッダ・セットと，各ヘッダに対応するデータが表示されます。



第 32 章

FTP プロトコル

VuGen では，FTP（File Transfer Protocol）サーバに直接アクセスすることによってネットワークの動作状況をエミュレートできます。

本章の内容

FTP Vuser スクリプトの作成について

FTP プロトコルは，FTP サーバに対して作業しているユーザのアクションをエミュレートする，下層プロトコルです。

FTP に関して，ユーザが FTP サーバにログインし，ファイルを転送してログアウトするのをエミュレートします。スクリプトを作成するには，FTP セッションを記録するか FTP 関数を手作業で入力します。

FTP セッションを記録するときに，VuGen は，メール・クライアントのアクションをエミュレートする関数を生成します。FTP，HTTP，およびメール・プロトコルなど，複数のプロトコルを介して通信を行う場合は，それらを全部記録できます。マルチ・プロトコルの指定の詳細については，『**第 1 巻 ― VuGen の使用**』の「VuGen を使った記録」を参照してください。

FTP プロトコルのスクリプトを作成するには，［e ビジネス］カテゴリで［File Transfer Protocol（FTP)］プロトコル・タイプを選択します。記録を開始するには，［**記録**］ボタンをクリックして，FTP サーバを対象に一般的なアクションを実行し記録します。スクリプトの作成と記録の詳細については，『**第 1 巻 ― VuGen の使用**』の「VuGen を使った記録」を参照してください。

スクリプトを作成した後，スクリプトを自分の環境（LoadRunner シナリオ，Performance Center 負荷テスト，または Business Process Monitor プロファイル）に組み込みます。詳細については，**HP LoadRunner Controller**，**HP Performance Center**，または **HP Business Availibility Center** のドキュメントを参照してください。

FTP 関数の処理

FTP セッションの記録中，VuGen は FTP 関数を生成します。各 FTP 関数は，**ftp** プレフィックスで始まります。

大部分の FTP 関数は，グローバル・セッションに使う関数と特定のメール・セッションの場所を示す関数が対になっています。すべてのセッションにアクションを適用するには，ex サフィックスのないバージョンを使用します。特定のセッションにアクションを適用するには，ex サフィックスのセッション識別子を持つバージョンを使用します。たとえば，**ftp_logon** はグローバルに FTP サーバにログオンしますが，**ftp_logon_ex** を使うと特定のセッションの FTP サーバにログオンします。

これらの関数の構文情報の詳細については，『**Online Function Reference**』（英語版）（［**ヘルプ**］＞［**関数リファレンス**］）を参照してください。

Vuser スクリプトの作成に使用するプログラミング言語が指定できます。詳細については，『**第 1 巻 ― VuGen の使用**』の「スクリプト生成オプションの設定」を参照してください。

次の例では，ftp_delete 関数を使って FTP サーバから test.txt ファイルを削除しています。

```
ftp_logon("FTP",
                    "URL=ftp://user:pwd@ftp.merc-int.com",
                    "LocalAddr=ca_server:21",
                    LAST);

ftp_delete("Ftp_Delete",
        "PATH=/pub/for_jon/test.txt", ENDITEM ,
        LAST);

ftp_logout();
```

第 33 章

Flex プロトコル

VuGen では，Flex 2 SDK で提供されているプロトコル・スイートをエミュレートする Vuser を作成できます。

本章の内容

Flex Vuser スクリプトの作成について

Flex は，Flash Player に基づいて RIA（Rich Internet Applications）を構築するためのフレームワークを開発者に提供するテクノロジの集まりです。

RIA は，標準の Web ページよりも動的な制御をユーザに提供する，軽量なオンライン・プログラムです。AJAX で構築された Web アプリケーションのように，Flex アプリケーションではユーザがアクションを起こすたびに新しい Web ページをロードする必要がないため，高い応答性を実現します。ただし，AJAX を使った作業とは異なり，Flex は JavaScript や CSS などのブラウザ実装には依存しません。フレームワークは Adobe のクロス・プラットフォーム対応の Flash Player で動作します。

Flex 2 アプリケーションは，多くの MXML ファイルおよび ActionScript ファイルで構成されます。これらは Flash player で再生できる単一の SWF ムービー・ファイルにコンパイルされ，クライアントのブラウザにインストールされます。

Flex 2 は，RPC，Data Management，およびリアルタイム・メッセージなど，さまざまなクライアント / サーバ間の通信方式をサポートします。Flex 2 は，HTTP，AMF，SOAP など，いくつかのデータ形式をサポートします。

VuGen では，Flex 2 RPC サービスとの通信をエミュレートする Vuser スクリプトを作成できます。VuGen の **Flex** タイプでは，AMF3 または HTTP データを使って作業する Flex アプリケーションをエミュレートするスクリプトを作成できます。SOAP データを使って作業する Flex アプリケーションの場合は，**Web サービス** Vuser タイプを使用します。

次の項では，AMF3 または HTTP 通信を使用するアプリケーション向けのスクリプトの作成方法について説明します。Web サービス Vuser スクリプトの詳細については，第 2 章「Web サービス プロトコル」を参照してください。

スクリプトを作成するには，［**e ビジネス**］カテゴリから［**Flex**］プロトコル・タイプを選択します。記録を開始するには，［**記録**］ボタンをクリックし，Flex アプリケーション内で標準的なアクションを実行します。スクリプトの作成および記録の詳細については，『**第 1 巻 ー VuGen の使用**』の「VuGen を使った記録」を参照してください。

スクリプトを作成した後，スクリプトを自分の環境（LoadRunner シナリオ，Performance Center 負荷テスト，または Business Process Monitor プロファイル）に組み込みます。詳細については，**HP LoadRunner Controller**，**HP Performance Center**，または **HP Business Availibility Center** のドキュメントを参照してください。

Flex 関数の処理

Flex アプリケーションを記録すると，VuGen はアプリケーションをエミュレートする Flex Vuser スクリプト関数を生成します。次の関数は Flex リモート・ステップの一部を示します。

関数名	説明
flex_login	パスワード保護された Flex アプリケーションにログオンします。
flex_logout	パスワード保護された Flex アプリケーションからログオフします。

flex_ping	Flex アプリケーションが使用できるかどうかを確認します。
flex_remoting_call	サーバ側リモート・オブジェクト（RPC）の 1 つ以上のメソッドを呼び出します。
flex_web_request	HTTP によってサポートされている任意のメソッドで HTTP 要求を送信します。
flex_amf_call	AMF 要求を送信します。
flex_amf_define_header_set	AMF のヘッダ・セットを定義します。
flex_amf_define_envlope_header_set	エンベロープ・ヘッダ・セットを定義します。

次の例では，**flex_ping** によってサービスの可用性を確認し，**flex_remoting_call** 関数によってサービスをリモートで起動しています。

```
flex_ping("1",
    "URL=http://testlab1/weborb30/console/weborb.aspx",
    "Snapshot=t6.inf",
    LAST);

flex_remoting_call("getProductEdition::GenericDestination",
    "URL=http://testlab1/weborb30/console/weborb.aspx",
    "Snapshot=t7.inf",
    INVOCATION,
    "Target=/2",
    "Operation=getProductEdition",
    "Destination=GenericDestination",
    "DSEndpoint=my-amf",
    "Source=Weborb.Management.LicenseService",
    "Argument=<arguments/>",
    LAST);
```

すべての Flex 関数の構文についての詳細については，『**Online Function Reference**』（英語版）（［**ヘルプ**］ > ［**関数リファレンス**］）を参照してください。

Flex スクリプトの相関

たいていの場合，Flash アプリケーションには，スクリプトを実行するたびに変わる動的なデータが含まれています。たとえば，実行のたびにオブジェクト名が変わるかもしれません。

Vuser スクリプトを記録するとき，VuGen はデータと引数値のセットを記録します。しかし，スクリプトの再生時に，サーバがこの引数を拒否し，エラーを発行することがあります。このエラーは，データが古くなってサーバで受け入れられなくなった動的なデータが原因であると考えられます。

これを回避するには，スクリプトの相関を行います。

- 必要な値を取り出す準備として，サーバからの応答を保存します。
- サーバの応答から必要な値を取り出します。
- 値をパラメータに保存します。
- それらのパラメータを Flex 要求への入力として使用します。

このようなエラーは必ずしもはっきりわかるわけではなく，Vuser ログ・ファイルを注意深く調べないと検出できないことがあります。Vuser の実行時にエラーが発生した場合は，スクリプト内でのエラーの発生箇所を調べてください。多くの場合，相関によって，1 つのステートメントの結果を別のステートメントの入力項目として使用することで，問題を解決できます。

相関を実行するには，次の手順を実行します。

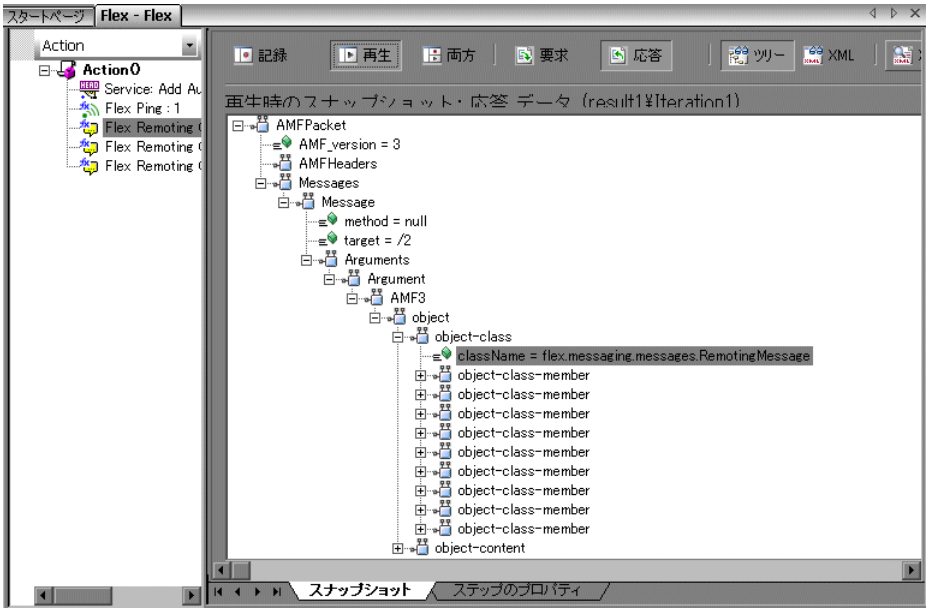
1 動的な値が原因で失敗し，相関が必要となったスクリプト内のステップを探します。

［再生ログ］を利用して問題のステップを探します。

```
Action.c(16): Error Server returned error for message #1 : "Incorrect session ID sent"/
Action.c(16): There was an error during the Flex Call ("ConnStatus")
```

2 先行するステップの中にある，正しい値を持っているサーバ応答を探します。

［再生ログ］内でエラーをダブルクリックすると，エラーが発生しているステップに移動できます。先行するステップをツリー・ビューの中で調べて，［**サーバ応答**］タブ内で値を探します。



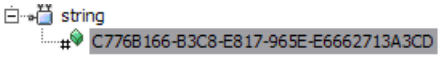
3 サーバ応答全体をパラメータに保存します。

値を取り出す前に，サーバ応答全体をパラメータに次のように保存します。

- 対象の値を含んでいるサーバ応答に対応するステップ・ノード（［アクション］表示枠内）を右クリックし，［**プロパティ**］を選択します。
- ［Flex Call Properties］ダイアログの中で，**応答パラメータ**の名前を入力します。
- ［**OK**］をクリックして，新しいパラメータ名を保存します。

4 元のサーバ応答値をパラメータに保存します。

➤［**再生時のスナップショット**：**応答データ**］で，値の上のノード（たとえば，string）を右クリックして［**パラメータに値を保存**］を選択します。



➤［XML パラメータのプロパティ］ダイアログの中で，**Name** にパラメータ名を入力します。以降のステップでは，この名前を使用することになります。

➤［**OK**］をクリックします。これで，スクリプトに **lr_xml_get_values** という新しい関数が追加されます。

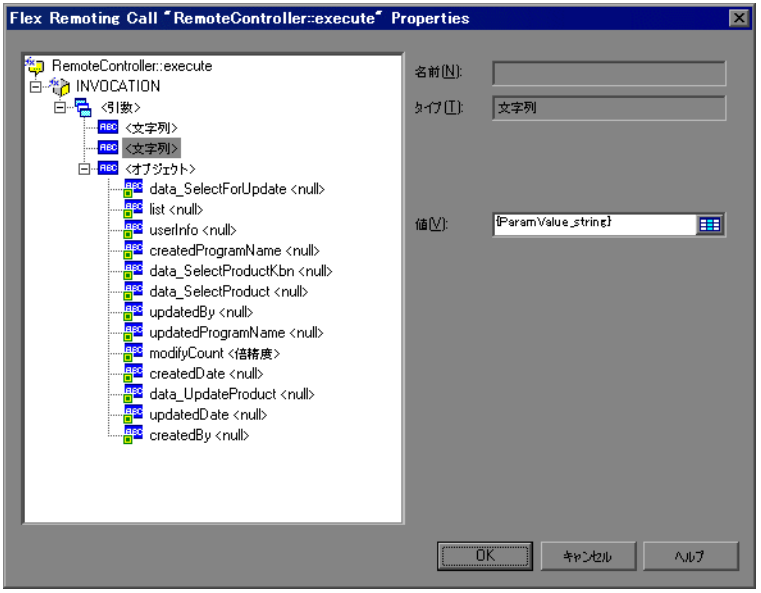
5 パラメータを以降の呼び出しに挿入します。

VuGen の編集ビューで，失敗した呼び出しを開始点に，オブジェクトに対する以降のすべての呼び出しに含まれている値を，定義したパラメータで置き換えます。

➤ 失敗している呼び出しに対応するステップ・ノード（[アクション］表示枠内）を右クリックし，［**プロパティ**］を選択します。

➤ 相関を必要としていた引数を探します。

➤［**値**］ボックスに，**{ParamValue_string}** のようにパラメータ名を中括弧で囲んで入力します。



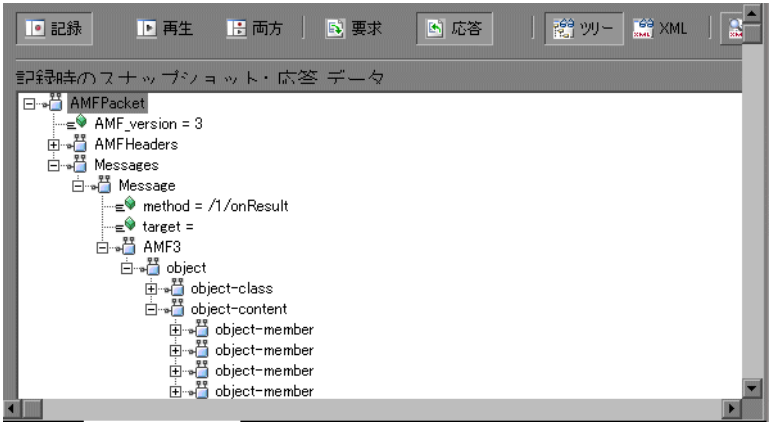
［**OK**］をクリックします。

6 スクリプトを実行します。

引数値が，保存したパラメータ値で適切に置き換えられていることを確認します。

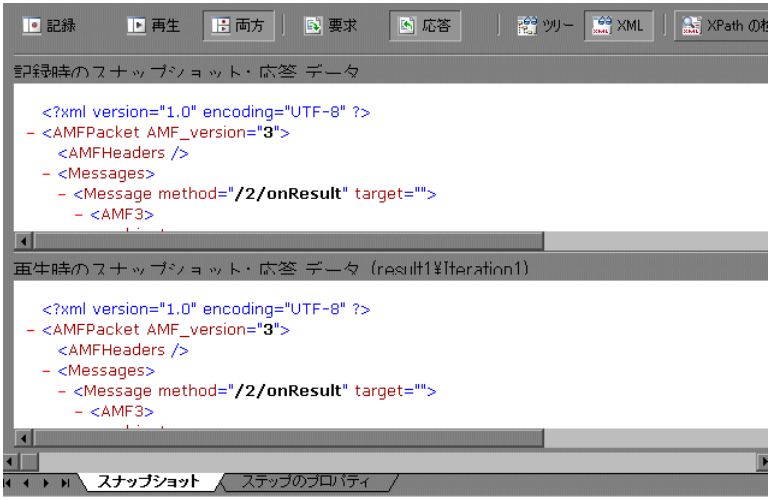
Flex データの表示

Flex データを表示するには，該当するステップをツリー・ビューの中で選択します。VuGen の右表示枠に，データのスナップショットが表示されます。



メッセージのすべての要素を表示するには，目的のノードを展開します。記録および再生時の要求データまたは応答データを表示するには，スナップショット内の該当のボタンをクリックします。

データを XML 構造で表示するには，スナップショット内の［**XML**］ボタンをクリックします。



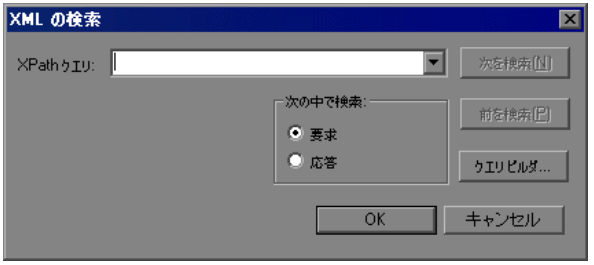
XML ツリーに対するクエリ

VuGen は，XML に対するクエリを作成して実行することが可能なクエリ・ビルダを備えています。

VuGen は展開可能なツリーに XML コードを表示します。XML ドキュメントに対するクエリを実行し，特定の名前空間 URI，値，または属性を検索できます。すべてのクエリで大文字と小文字は区別されます。

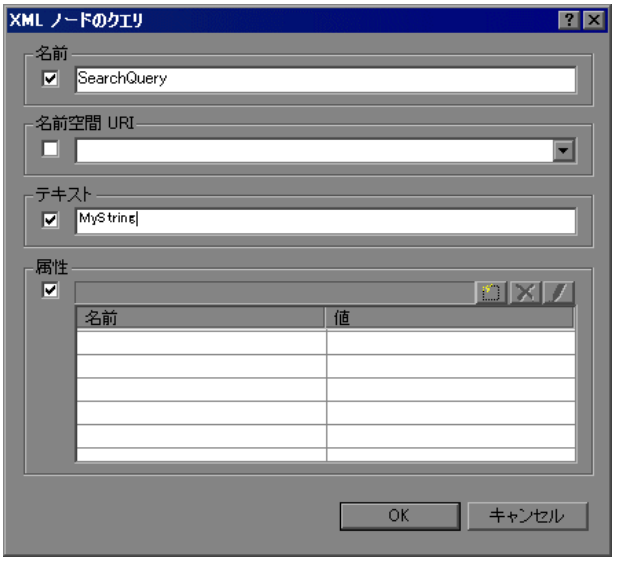
クエリを行うには，次の手順を実行します。

1 ［スナップショット］タブで，検索するノードを選択します。［**XPath の検索**］ボタンをクリックします。［XPath の検索］ダイアログ・ボックスが開きます。



2 ［要求］または［応答］を選択します。XPath クエリを入力し，［**OK**］をクリックします。クエリを作成するには，［**クエリ ビルダ**］ボタンをクリックします。［XML ノードのクエリ］ダイアログ・ボックスが開きます。

3 1 つまたは複数の項目を検索対象として有効にします。



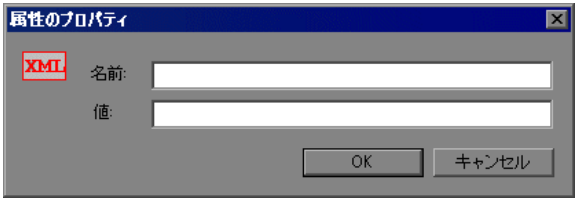
4 [**名前**] セクションを有効にして，ノードまたは要素の名前を検索します。

5 [**名前空間 URI**] セクションを有効にして，名前空間を検索します。

6 [**テキスト**] セクションを有効にして，[名前] ボックスに表示されている要素の値を検索します。

7 [**属性**] セクションを有効にして，属性を検索します。

8 該当するボックスに検索テキストを入力します。属性を追加するには，[**追加**] ボタンをクリックします。[属性のプロパティ] ボックスが開きます。属性の名前と値を入力します。[**OK**] をクリックします。

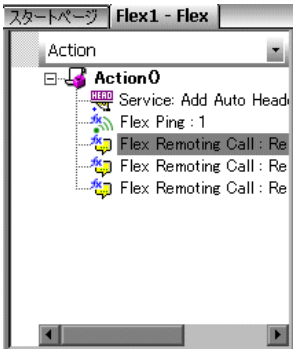


9 ［XML ノードのクエリ］ダイアログ・ボックスの中で［**OK**］をクリックします。VuGen によって，クエリ・テキストが［XPath の検索］ボックスに挿入されます。

10 ［**次を検索**］をクリックして検索を開始します。

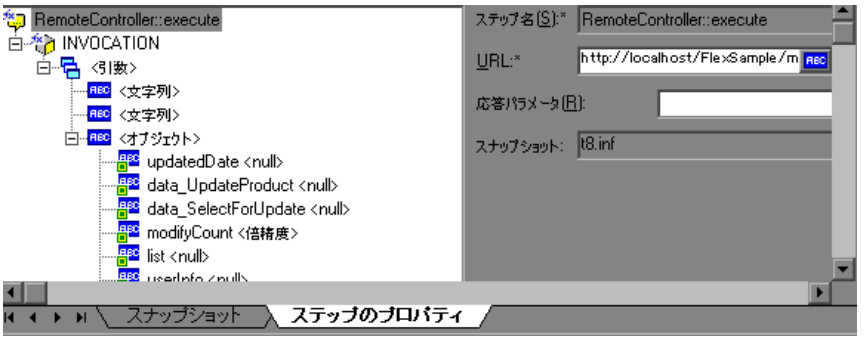
Flex ステップのプロパティの設定

Flex Vuser スクリプトには一連の Flex 呼び出しが含まれています。ツリー・ビューで各ステップを選択すると，その情報を表示できます。



（ほとんどの Flex ステップでは）右側の表示枠に［スナップショット］と［ステップのプロパティ］の 2 つのタブがあります。［スナップショット］タブにはデータが表示され，［ステップのプロパティ］には各ステップのプロパティが表示されます。

次の例は，**Flex Remoting Call** ステップのプロパティを示しています。



それぞれのノードを選択すると，右側の表示枠にノード固有のプロパティが表示されます。

Flex RTMP を使った作業

Flex では，RTMP（Real Time Messaging Protocol）に関するスクリプトの記録と再生ができます。RTMP シミュレーションを可能にするために，Flex プロトコルの記録オプションを設定する必要があります。

RTMP を有効にするには，次の手順を実行します。

1 ［**ツール**］ > ［**記録オプション**］を選択するか，［記録開始］ダイアログ・ボックスで［**オプション**］ボタンをクリックして，［記録オプション］ダイアログ・ボックスを開きます。

2 ［**ネットワーク**］ > ［**ポート マッピング**］ノードで，［**オプション**］をクリックします。

3 ［**バッファ サイズしきい値の送受信**］を 1500 に設定します。

第 34 章

LDAP プロトコル

VuGen で LDAP サーバとの通信をエミュレートできます。

本章の内容

LDAP Vuser スクリプトの作成について

LDAP（Lightweight Directory Access Protocol）は，ディレクトリ・データベースにアクセスするのに使用するプロトコルです。LDAP ディレクトリは，多くの LDAP エントリで構成されています。各 LDAP エントリは，DN（識別名）と呼ばれる名前と属性の集合です。DN の詳細については，512 ページ「識別名エントリの定義」を参照してください。

LDAP ディレクトリ・エントリは，政治的，地理的，組織的な境界を反映した階層構造で配置されています。国を表すエントリは，ツリーの一番上に現れます。その下には州や全国的な組織名を表すエントリが表示されます。さらにその下には，個人や組織，プリンタ，ドキュメントなどのエントリが表示されます。

VuGen では LDAP サーバとの通信を記録できます。VuGen によってユーザのアクションをエミュレートする関数を使ったスクリプトが生成されます。このスクリプトには，LDAP サーバへのログインとログアウト，エントリの追加と削除，およびエントリの照会が記述されます。

LDAP プロトコルのスクリプトを作成するには，［e ビジネス］カテゴリで［Lightweight Directory Access Protocol（LDAP）］プロトコル・タイプを選択します。記録を開始するには，［**仮想ユーザ**］＞［**記録開始**］を選択し，LDAP サーバを対象に一般的な操作を行います。記録の手順については，『**第 1 巻 － VuGen の使用**』の「VuGen を使った記録」を参照してください。

スクリプトを作成した後，スクリプトを自分の環境（LoadRunner シナリオ，Performance Center 負荷テスト，または Business Process Monitor プロファイル）に組み込みます。詳細については，**HP LoadRunner Controller**，**HP Performance Center**，または **HP Business Availibility Center** のドキュメントを参照してください。

LDAP 関数の処理

Vuser スクリプトの作成に使用するプログラミング言語が指定できます。詳細については，『**第 1 巻 － VuGen の使用**』の「スクリプト生成オプションの設定」を参照してください。

LDAP Vuser スクリプト関数は，LDAP プロトコルをエミュレートします。LDAP 関数は，**mldap** というプレフィックスが付いています。

LDAP 関数はすべて，グローバル・セッション用の関数と局所的な特定のセッションを指定できる関数の対になっています。すべてのセッションにアクションを適用するには，**ex** サフィックスのないバージョンを使用します。特定のセッションにアクションを適用するには，ex サフィックスのセッション識別子を持つバージョンを使用します。たとえば，**mldap_logon** はグローバルに LDAP サーバにログオンしますが，**mldap_logon_ex** は特定セッションの LDAP サーバにログオンします。

これらの関数の構文情報の詳細については，『**Online Function Reference**』（英語版）（［**ヘルプ**］＞［**関数リファレンス**］）を参照してください。

次の例では，ユーザが LDAP サーバ「ldap1」にログオンしています。このユーザはエントリを追加し，OU 属性を「Sales」から「Marketing」に変更しています。

```
Actions()
{
        // LDAP サーバにログオン
        mldap_logon("Login",
                "URL=ldap://johnsmith:tiger@ldap1:80",
                LAST);

        // Sally R. Jones にエントリを追加
        mldap_add("LDAP Add",
          "DN=cn=Sally R. Jones,OU=Sales, DC=com",
                "Name=givenName", "Value=Sally", ENDITEM,
                "Name=initials", "Value=R", ENDITEM,
                "Name=sn", "Value=Jones", ENDITEM,
                "Name=objectClass", "Value=contact", ENDITEM,
                LAST);

        // Sally の OU を「Marketing」に変更
        mldap_rename("LDAP Rename",
                "DN=CN=Sally R. Jones,OU=Sales,DC=com",
                "NewDN=OU=Marketing",
                LAST);

        // LDAP サーバからログアウト
        mldap_logoff();
    return 0;
}
```

識別名エントリの定義

LDAP API では，オブジェクトが**識別名**（DN）によって参照されます。DN は，カンマで区切られた一連の「**相対識別名**」（RDN）です。

RDN は，属性と関連する値を attribute=value という形式で表したものです。属性名では大文字と小文字は区別されません。最も一般的な RDN 属性の型を次の表に示します。

文字列	属性の型
DC	domainComponent
CN	commonName
OU	organizationalUnitName
O	organizationName
STREET	streetAddress
L	localityName
ST	stateOrProvinceName
C	countryName
UID	userid

次に識別名の例を示します。

DN=CN=John Smith,OU=Accounting,DC=Fabrikam,DC=COM
DN=CN=Tracy White,CN=admin,DC=corp,DC=Fabrikam,DC=COM

次に属性値に使用できない予約文字を示します。

文字	説明
	文字列の先頭にスペースまたは # 文字は指定できません。
	文字列の末尾にスペース文字は指定できません。
,	カンマ
+	プラス記号
"	二重引用符
¥	バックスラッシュまたは円記号
<	左山括弧
>	右山括弧
;	セミコロン

予約語を属性値の一部として使用するには，その前にエスケープ文字であるバックスラッシュまたは円記号（¥）を付けます。属性値に等号（=）や非 UTF-8 文字などほかの予約文字が含まれる場合は，その文字を 16 進形式でエンコードする必要があります（バックスラッシュまたは円記号の後ろに 16 進数が 2 桁）。

次にエスケープ文字を含む DN の例を示します。最初の例は，カンマの埋め込まれた部門名で，2 番目の例は，キャリッジ・リターンを含む値です。

```
DN=CN=Bitwise,OU=Docs¥, Support,DC=Fabrikam,DC=COM
DN=CN=Before¥0DAfter,OU=Test,DC=North America,DC=Fabrikam,DC=COM
```

接続オプションの指定

mldap_logon[_ex] 関数を使用して，LDAP サーバにログインする方法を制御します。

LDAP サーバの URL を指定する場合，接続方法と使用する資格情報を指定します。

サーバの URL を指定する場合には，次の形式で指定します。

```
ldap[s][ ユーザ名 :[ パスワード ]@][ サーバ [: ポート ]]
```

次の表に，LDAP サーバへの接続の例をいくつか示します。

構文	説明
ldap://a:b@server.com:389	サーバ（ポート 389）に接続し，ユーザ名「a」，パスワード「b」でバインドします。
ldap://:@server.com	サーバ（標準の非保護ポート 389）に接続し，NULL のユーザ名とパスワードで匿名バインドします。
ldaps://a:@server.com	サーバ（標準の保護ポート 636）に接続し，ユーザ名「a」，パスワードなしでバインドします。
ldap://@server.com, ldap://server.com	バインドせずにサーバに接続します。
ldap://a:b@	ユーザ名「a」，パスワード「b」でバインドします。再接続せずに既存のセッションでバインドを実行します。
ldap://:@	NULL のユーザ名とパスワードで匿名バインドします（再接続せずに既存のセッションでバインドを実行）。

次のオプションの引数を使用して，LDAP モードまたは SSL 証明書を指定することもできます。

- **Mode**：LDAP 呼び出しモード（**同期**または**非同期**）です。
- **Timeout**：LDAP サーバを検索する最大時間（秒単位）です。
- **Version**：LDAP プロトコルのバージョンで，バージョン 1，2，または 3 です。
- **SSLCertDir**：SSL 証明書データベース・ファイル（cert8.db）へのパスです。
- **SSLKeysDir**：SSL キー・データベース・ファイル（key3.db）へのパスです。

- **SSLKeyNickname**：キー・データベース・ファイル内の SSL キーのニックネームです。
- **SSLKeyCertNickname**：証明書データベース・ファイル内の SSL キーの証明書のニックネームです。
- **SSLSecModule**：SSL セキュリティ・モジュール・ファイル（secmod.db）へのパスです。
- **StartTLS**：接続を TLS（SSL）モードに切り替えるには，StartTLS 拡張の特定のコマンドを発行する必要があります。

これらの引数の詳細については，『**Online Function Reference**』（英語版）（［**ヘルプ**］＞［**関数リファレンス**］）を参照してください。

第 35 章

Microsoft .NET プロトコル

VuGen は，.NET Framework 環境で作成されたアプリケーションを記録します。

本章の内容

スクリプトを記録する前に，第 36 章「Microsoft .NET フィルタ」を読むことをお勧めします。このガイドラインは，アプリケーションを正確にエミュレートする最適なスクリプトの作成に役立ちます。

Microsoft .NET Vuser スクリプトの記録について

Microsoft .NET Framework は，ASP.NET，Windows Forms，Web サービス，分散アプリケーション，またはこれらのモデルを組み合わせたアプリケーションなど，さまざまな種類のアプリケーションを構築する際の強固な基盤を提供します。

VuGen ではアプリケーション・レベルのプロトコルとして .NET をサポートしています。VuGen を使用して，.NET Framework で作成された Microsoft .NET クライアント・アプリケーションのユーザをエミュレートする Vuser スクリプトを作成できます。VuGen はクライアントのすべてのアクションをメソッドおよびクラスとして記録し，C# または VB .NET 言語のスクリプトを作成します。

標準設定では，VuGen 環境は .NET Remoting，ADO.NET，Enterprise Services，および WCF（Windows Communication Foundation) のアプリケーションで構成されます。これ以外のクライアント / サーバ動作に基づいて作成されたアプリケーションを記録する場合に VuGen を設定する方法については，カスタマ・サポートにお問い合わせください。

.NET および前述の環境の詳細については，MSDN Web サイト（http://msdn2.microsoft.com）を参照してください。

制限事項

VuGen で Microsoft .NET アプリケーションを記録するには，次の制限事項があります。

- Microsoft .NET スクリプトは，VuGen でシングル・プロトコルの記録のみサポートします。
- パブリック・フィールドに直接アクセスすることはできません。AUT はメソッドまたはプロパティを介してフィールドにアクセスする必要があります。
- VuGen はアプリケーションの静的フィールドは記録しません。クラス内のメソッドだけが記録されます。
- マルチ・スレッドをサポートするかどうかは，クライアント・アプリケーションに依存します。記録されたアプリケーションがマルチ・スレッドをサポートする場合は，Vuser スクリプトもマルチ・スレッドをサポートします。
- 場合によっては，スクリプトを修正しないと反復を複数回実行できないことがあります。前の反復ですでに初期化されたオブジェクトの再初期化はできません。したがって，反復を複数回実行するには，各反復の終了時に開いている接続またはリモート・チャネルを必ずすべて閉じます。

- IIS でホストされた MSMQ および Enterprise Services に基づいた Enterprise Services 通信の記録はサポートされていません。
- VuGen は，クライアント・アプリケーションによってホストされた WCF サービスの記録を部分的にサポートします。
- ユーザ定義のプロキシを使用した Remoting 呼び出しの記録はサポートされていません。
- 標準設定の ADO.NET フィルタを使用する場合，ADO.NET オブジェクトの **ExtendedProperties** プロパティの記録はサポートされていません。
- Framework 2.0 と互換性のない .NET Framework 1.1 で作成されたアプリケーションは記録されません。Framework 1.1 アプリケーションに互換性があるかどうかを確認するには，次の XML タグをアプリケーションの .config ファイルに追加します。

```
<configuration>
   <startup>
      <supportedRuntime version="v2.0.50727"/>
   </startup>
</configuration>
```

VuGen を介さずにアプリケーションを直接起動し，アプリケーションの動作を確認します。アプリケーションが正しく動作すれば，VuGen を使用してこのアプリケーションを記録できます。VuGen を使用してこの AUT を記録する前に，前述のタグは削除してください。この方法の詳細については，MSDN Knowledge Base を参照してください。

Microsoft .NET Vuser を使った作業の概要

本項では，Microsoft .NET Vuser スクリプトを作成するプロセスについて説明します。スクリプトを実行するマシンとすべての Load Generator に AUT（テスト対象アプリケーション）がインストールされている必要があります。

基本となる .NET Vuser スクリプトを作成するには，次の手順を実行します。

1 VuGen を使ってアプリケーションを記録します。

VuGen を起動し，新しい Vuser スクリプトを作成します。Vuser のタイプとして **Microsoft .NET** を指定します。記録対象アプリケーションを選び，記録オプションを設定します。詳細については，『**第 1 巻 ー VuGen の使用**』の第 16 章「選択したプロトコルの記録オプション」を参照してください。

一般的な記録方法の詳細については，『**第 1 巻 ー VuGen の使用**』の「VuGen を使った記録」を参照してください。

2 スクリプトをデバッグし，フィルタを設定します。

記録されたスクリプトを調べ，必要なメソッドが記録されているかどうかを確認します。必要があれば，フィルタを変更してスクリプトを記録しなおします。

フィルタ設定の詳細については，543 ページ「フィルタ設定についてのガイドライン」を参照してください。

3 スクリプトを相関させます。

スクリプト内の値を相関させ，スクリプトの以降の場所で使用できるように値をパラメータとして保存します。

詳細については，525 ページ「Microsoft .NET スクリプトの相関」を参照してください。

4 VuGen でスクリプトを実行します。

VuGen でスクリプトを保存して実行し，正しく動作することを確認します。再生が正常に行われるようにするために，スクリプトを Load Generator マシンまたは Controller 経由でリモート実行する前に，VuGen でコンパイルして実行します。これは，スクリプトの設定またはスクリプトそのもののいずれを変更した場合でも行います。また，スクリプトを別の名前で保存した場合は，スクリプトを Load Generator で実行する前に VuGen で再生します。

詳細については，『**第 1 巻 ー VuGen の使用**』の「スタンドアロン・モードでの Vuser スクリプトの実行」を参照してください。

5 実行環境を設定します。

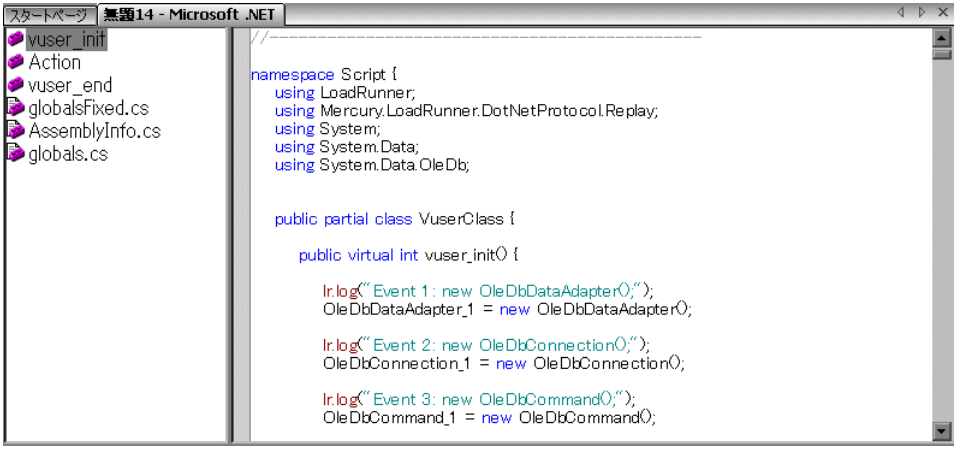
実行環境を設定することによって，スクリプト実行時の Vuser の動作を制御します。設定には，ループ，ログ，タイミング情報などがあります。

詳細については，『**第 1 巻 ― VuGen の使用**』の第 24 章「選択したプロトコルの実行環境の設定」を参照してください。

スクリプトを作成した後，スクリプトを自分の環境（LoadRunner シナリオ，Performance Center 負荷テスト，または Business Process Monitor プロファイル）に組み込みます。詳細については，**HP LoadRunner Controller**，**HP Performance Center**，または **HP Business Availibility Center** のドキュメントを参照してください。

VuGen と Visual Studio でのスクリプトの表示

記録後，VuGen のスクリプト・ビューでスクリプトを表示できます。



VuGen は発生したアクションを記録し，C# または VB コードを生成します。標準設定では，VuGen はすべてのステップを **lr.log** 呼び出しで囲みます。

スクリプトの再生時には，VuGen はまずスクリプトをコンパイルし，すべての呼び出しが有効で構文が正しいことを確かめます。VuGen はスクリプトを「**Script.dll**」という DLL ファイルとしてコンパイルし，スクリプトの **bin** フォルダに保存します。DLL ファイルには，Init，Actions，End という 3 つの関数が含まれます。

スクリプトを実行せずにコンパイルして構文を確認できます。VuGen からスクリプトを直接コンパイルするには，Shift + F5 キーを押すか，［**仮想ユーザ**］＞［**コンパイル**］を選択します。コンパイル・エラーが検出されると，［出力］ウィンドウにエラーが表示されます。エラーをダブルクリックすれば，スクリプトの問題が生じている行に移動できます。

VuGen からスクリプトを直接実行するには，F5 キーを押すか，［**仮想ユーザ**］＞［**実行**］を選択します。ブレークポイントおよびステップ単位の再生は，Microsoft .NET Vuser の場合には VuGen の エディタ・ウィンドウでサポートされていません。スクリプトをデバッグし，ブレークポイントを使用したりステップ単位で実行したりするには，次に説明に従って Visual Studio .NET の中で実行します。

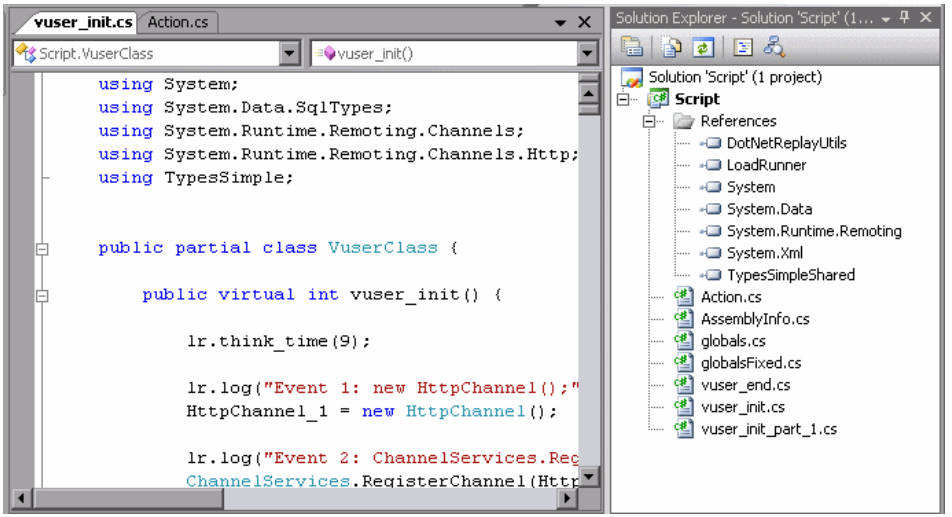
Visual Studio によるスクリプトの表示

Visual Studio は，スクリプトを表示，編集，およびデバッグするための付加的なツールを備えています。ブレークポイントの追加，変数値の表示，アセンブリ参照の追加，Visual Studio の IntelliSense を使用してのスクリプト編集ができます。また，デバッグのためにステップ単位でスクリプトを実行できます。

スクリプトの保存時に，スクリプトのフォルダに **Script.sln** という Visual Studio 2005 ソリューション・ファイルが作成されます。このソリューション・ファイルを Visual Studio .NET で開き，ソリューション・エクスプローラにすべてのコンポーネントを表示できます。

Visual Studio 2005 でソリューションを開くには，［**仮想ユーザ**］＞［**Visual Studio 2005 で開く**］を選択するか，VuGen のツールバーの［Visual Studio］ボタンをクリックします。

ソリューション・エクスプローラの中で **vuser_init.cs** など該当のセクションをダブルクリックし，スクリプトの内容を表示します。



VuGen では，記録中に必要だったすべての参照は自動的にロードされます。ソリューション・エクスプローラを使用して，コンパイル時および再生時に使用する参照を追加できます。［**参照**］ノードを選択して，右クリック・メニューから［**参照の追加**］を選択します。

ソリューション・エクスプローラで **globals.cs** または **globals.vb** をクリックして，スクリプトで定義および使用された変数の一覧を表示します。

データ・セットとグリッドの表示

データ・セット，データ・テーブル，またはデータの読み込みアクションを返すメソッドを記録すると，データを表示するグリッドが生成されます。

データの読み込みを処理する場合，VuGen は各**読み取り**操作から取得したデータを収集し，再生ヘルパ関数 **DoDataRead** に変換します。

たとえば，次のようなアプリケーション・コードを記録したとします。

```
SqlDataReader reader = command.ExecuteReader();
    while( reader.Read() )
    {
        // カラム 1 にある文字列を取得するなど，値を読み取る
        string str = reader.GetString(1)
    }
```

VuGen はその後，スクリプトに次の行を生成します。

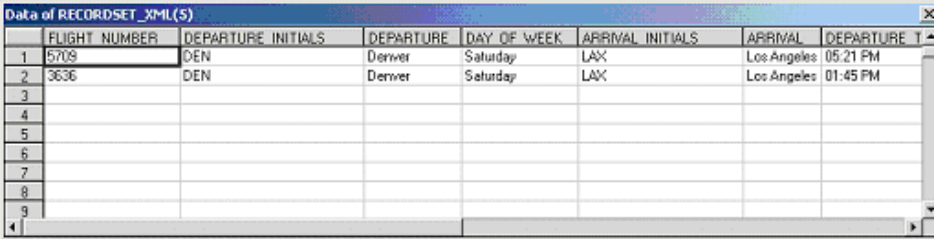
```
SqlDataReader_1 = SqlCommand_1.ExecuteReader();
LrReplayUtils.DoDataRead(SqlDataReader_1, out valueTable_1, true, 27);
```

ここで 2 つのパラメータは，記録時に，アプリケーションが取得可能な 27 のレコードをすべて読み込んだことを示します。したがって，再生時にスクリプトはすべての取得可能なレコードを読み込みます。

さらに，VuGen により，**読み取り**操作で取得したすべての情報を含むデータ・グリッドが生成されます。

再生時に，取得された実際の値を含んだ出力データ・テーブルを相関や検証に使用できます。**DoDataRead** 関数の詳細については，『**Online Function Reference**』（英語版）（［**ヘルプ**］＞［**関数リファレンス**］）を参照してください。

標準設定では，スクリプトにはグリッドが表示されています。グリッドの表示を無効にし，グリッドを閉じた状態で表示させるには，［**表示**］＞［**データ グリッド**］を選択します。



データ・セットは XML ファイルに格納されます。XML ファイルはスクリプトの data/datasets フォルダにあります。データ・ファイルは，20.xml のように＜**インデックス名**＞ .xml ファイルの形式で表されます。1 つのファイルにいくつかのデータ・テーブルが含まれている可能性があるため，**datasets.grd** ファイルを参照してください。このファイルではデータを含む XML を特定するために，スクリプトのインデックスをファイルのインデックスに割り当てます。

グリッドの詳細については，370 ページ「グリッドを使った作業」を参照してください。

Microsoft .NET スクリプトの相関

セッションを記録し終えたら，スクリプト内の 1 つまたは複数の値を**相関**する必要がある場合があります。値の相関とは，スクリプトの再生中に値をキャプチャし，これをパラメータとして保存することです。このパラメータは，スクリプトのそれ以降の場所で使用できます。

VuGen では自動的に基本の相関が行われます。オブジェクトが関数呼び出しから返され，その後スクリプト内で呼び出されるか，またはそのオブジェクトが別のメソッドに渡される場合，VuGen は同じオブジェクト・インスタンスを使用します。

コーディングするか VuGen の組み込み相関ツールを使用してパラメータを手作業で保存することにより，スクリプト内の値をさらに相関できます。

VuGen 内で値を相関させるには，データ・セット内で値を探し出し，値を右クリック・メニューを使用してパラメータに保存します。

ADO.NET 環境の値を相関させるには，次の手順を実行します。

1 スクリプト内でデータ・セットを見つけます。

スクリプトにグリッドを表示し，返されたデータ・セットを表示します。グリッドが表示されない場合は，［**表示**］＞［**データ グリッド**］を選択するか，適切な **DATASET_XML** ステートメントを展開します。次に例を示します。

	CustomerID	CompanyName	ContactName	ContactTitle	Address
1	ABC	ABC Company	John Smith	Owner	One My Way
2	ALFKI	Alfreds Futterkiste	Maria Anders	Sales Representative	Obere Str. 57
3	ANATR	Ana Trujillo Emparec	Ana Trujillo	Owner	Avda. de la C
4	ANTON	Antonio Moreno Tac	Antonio Moreno	Owner	Mataderos 2
5	AROUT	Around the Horn	Thomas Hardy	Sales Representative	120 Hanover

2 値を探します。

相関させる値を検索します。グリッド内で値を検索するには，［検索］ダイアログ・ボックスを開き（Ctrl+F），［**グリッド内を検索**］オプションを選択します。

3 相関を作成します。

相関させる値をグリッドの中でクリックし，右クリック・メニューから［**相関の作成**］を選択します。［相関の作成］ダイアログ・ボックスが開きます。

4 パラメータ名を指定します。

先に定義した変数と同一のパラメータ名を指定します。［**OK**］をクリックします。すべての候補を検索するかどうかを尋ねられます。［**OK**］をクリックします。

VuGen は各データ・セットの前に lr.save_string 関数を追加します。次に例を示します。

```
lr.save_string("MyCustomerID",
CustomerAndOrdersDataSet_3.Tables["Customers"].Rows[0]["CompanyName"].ToStri
ng());
```

5 スクリプトの以降の場所でパラメータを参照します。

パラメータで置換する値を選択して，右クリック・メニューから［**パラメータで置換**］を選択します。保存した変数名を［パラメータ名］ボックスに入力します。［OK］をクリックします。文字列の値を評価する lr.eval_string 関数を使用してすべての値をパラメータで置換するかどうかを尋ねられます。

```
lr.message("The customer ID is""+ lr.eval_string("{MyCustomerID}") + '');
```

ほかのプロトコルとは異なり，スクリプトにはアプリケーションまたはフレームワーク・メソッドに対する直接の呼出しが含まれます。したがって，文字列値を {paramName} で置換することはできません。代わりにパラメータの値を評価する **lr.eval_string** を使用する必要があります。

この方法は ADO.NET 環境に対して適用できます。プリミティブな値については，出力パラメータ値を含んだスクリプトを生成し，出力パラメータを調べて相関させる必要があります。

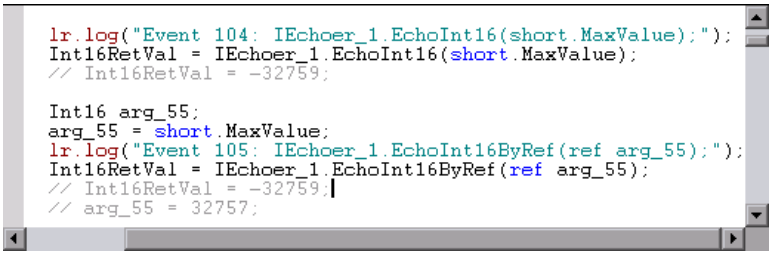
出力パラメータと相関させるには，次の手順を実行します。

1 ［**ツール**］ > ［**記録オプション**］を選択し，［**一般 : スクリプト**］ノードを選択します。

2 ［**出力パラメータ値を挿入する**］オプションを有効にします。［**OK**］をクリックして［記録オプション］を閉じます。

3 ［**ツール**］ > ［**スクリプトの再生成**］を選択して，スクリプトを再生成します。

4 コメントになっている出力プリミティブ値を検索して相関させます。



相関関数の詳細については，『**Online Function Reference**』（英語版）（［**ヘルプ**］ > ［**関数リファレンス**］）を参照してください。

アプリケーションのセキュリティと権限の設定

アプリケーションの記録中に発行されるセキュリティ例外は，記録を行うマシンにアプリケーションを記録するための十分な権限がないという，権限の不足によるものが一般的です。これは，アプリケーションがローカル・マシンになく，インターネットやネットワーク上にある場合によく起こります。

この問題を解決するには，記録マシンからアプリケーションおよびスクリプトに Full Trust 権限でアクセスできるようにする必要があります。

1 つの解決方法は，アプリケーションをローカル・マシンにコピーし，スクリプトをローカル・マシンに保存するというものです。ユーザは，ローカル・アプリケーションおよびフォルダには標準設定で Full Trust 権限を持つからです。

もう 1 つの解決方法は，各アプリケーション・フォルダおよびスクリプト・フォルダに Full Trust 権限を付与する新しいコード・グループを作成するというものです。

この手順は，使用しているマシンに Visual Studio がインストールされているかどうかによって異なります。

Visual Studio がインストールされていない場合，Full Trust 権限を特定のフォルダに付与するには，次の手順を実行します。

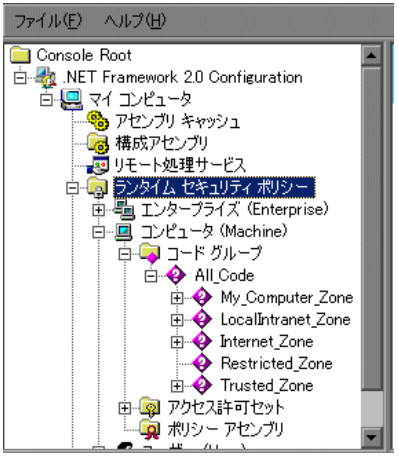
1 コマンド・プロンプトから，caspol.exe アプリケーションを実行します。

2 必要な権限を設定します。

Visual Studio がインストールされている場合，Full Trust 権限を特定のフォルダに付与するには，次の手順を実行します。

1 ［.NET Configuration］の設定を開きます。［**スタート**］＞［**プログラム**］＞［**管理ツール**］＞［**Microsoft .NET Framework 2.0 Configuration**］を選択します。［.NET Configuration］ウィンドウが開きます。

2［**ランタイム セキュリティ ポリシー**］ノードを展開して，マシンのコード・グループを表示します。

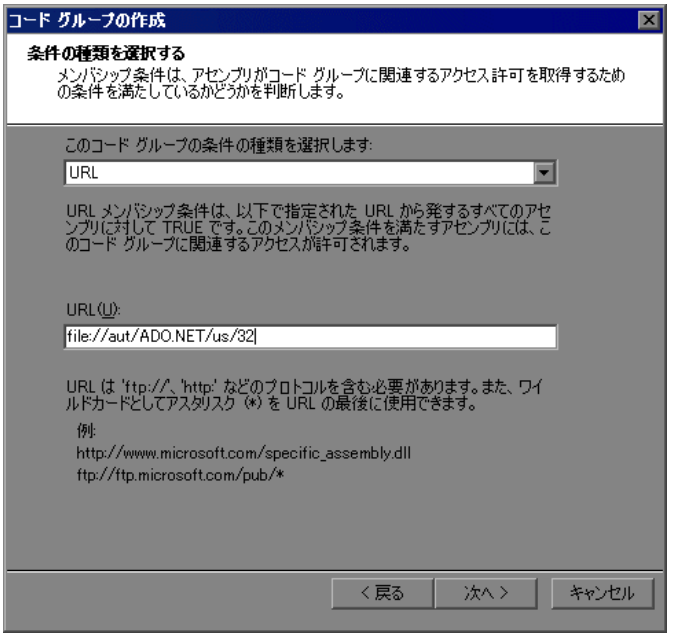


3［**All_Code**］ノードを選択します。

4［**操作**］＞［**新規作成**］を選択します。［コード グループの作成］ダイアログ・ボックスが表示されます。

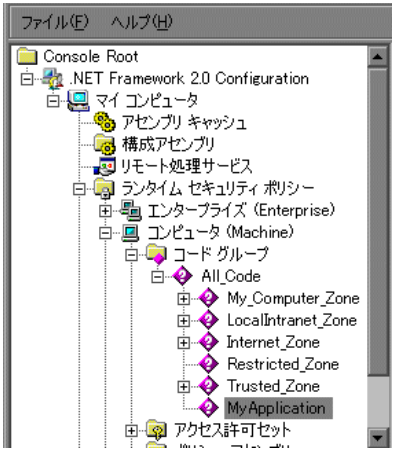
5 アプリケーションまたはスクリプトに新規コード・グループの名前を入力します。［**次へ**］をクリックします。

6 条件の種類に［**URL**］を選択します。［URL］ボックスで，アプリケーションまたはスクリプトのフル・パスを指定し（file://... 形式で），［**次へ**］をクリックします。



7 アクセス許可セットに［**FullTrust**］を選択します。［**次へ**］ボタンをクリックします。

8［ウィザードの完了］ダイアログ・ボックスで［**完了**］をクリックします。設定ツールにより，既存のグループの一覧に新しいコード・グループが追加されます。



9 記録するすべての .NET アプリケーションに前述の手順を繰り返します。

10 Vuser スクリプト・フォルダに，前述の手順を繰り返します。

注：スクリプト・フォルダに，テストに参加しているすべての Load Generator マシンに対する **FullTrust** 権限があることを確認してください。

WCF 双方向通信の記録

WCF（Windows Communication Foundation）のプログラミング・モデルは，Web サービス，.NET Remoting，Distributed Transactions，および Message Queues を，分散コンピューティングのための単一のサービス指向プログラミング・モデルに統合します。

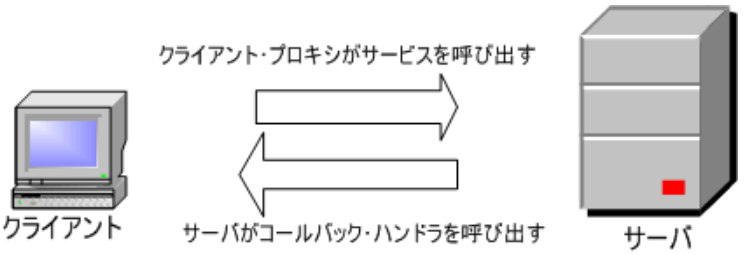
WCF はプロキシ・オブジェクトを作成し，サービスにデータを提供します。また，サービスによって返されたデータを呼び出し元が期待する形式にまとめます。

WCF 環境の一般的なサポートに加え，VuGen は，WCF の双方向通信を使用するアプリケーション専用のサポートを行います。双方向通信では，クライアントのプロキシはサービスに接続し，サービスはクライアントのマシンに対してコールバック・ハンドラを呼び出します。コールバック・ハンドラは，サーバによって定義されたコールバックのインタフェースを実装しています。サーバは同期方式で応答する必要がありません。サーバは応答やコールバック・ハンドラの呼び出し時期を独立して決定できます。

クライアントとサーバ間の通信は次のとおりです。

- サーバはサービス契約とコールバックのインタフェースを定義します。
- クライアントは，サーバによって定義されたコールバック・インタフェースを実装します。

➤ サーバは，必要に応じてクライアント内のコールバック・ハンドラを呼び出します。



双方向通信を記録して再生しようとすると，スクリプトが元のコールバック・メソッドを呼び出すときに問題が生じる可能性があります。標準設定では，コールバック・ハンドラはフィルタに含まれていません。コールバック・ハンドラを含めるようにフィルタをカスタマイズできます。ただし，コールバックの多くは GUI 更新などローカルな操作なので，標準的な再生をしても負荷テストの意味がありません。効果的な負荷テストを行うために，サーバによって呼び出される元のコールバック・メソッドを再生することはできません。

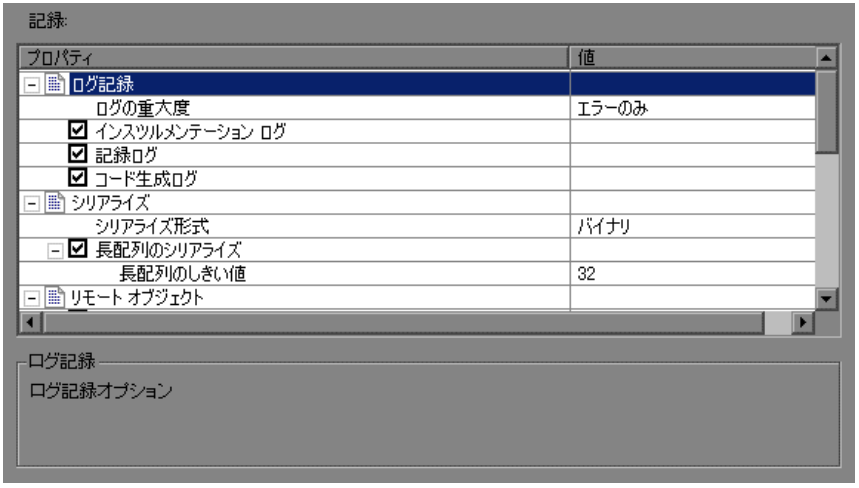
VuGen は，元のコールバック・ハンドラをダミー実装に置きかえてこれを解決します。この実装は，ユーザのアプリケーション向けにさらにカスタマイズできる一般的なアクション・セットを実行します。

VuGen に対して元のコールバックを置き換えるよう指定するには，［**ダミー コールバック ハンドラを生成**］記録オプションをアクティブにします。詳細については，次項を参照してください。

WCF 記録オプションの設定

WCF 双方向通信向けの VuGen の記録オプションでは，負荷テストを効果的に行うためのスクリプトを生成できます。次の項目について記録オプションの設定が可能です。

- ダミー・コールバック実装の生成
- デュアル HTTP バインディングの記録



ダミー・コールバック実装の生成

［**ダミー コールバック ハンドラを生成**］記録オプションは，VuGen に対して，双方向通信での元のコールバックをダミー・コールバックに置き換えるよう指定します。

ダミー・コールバック実装は，次のアクションを実行します。

- **引数の格納**：再生中にサーバがハンドラを呼び出すと，メソッド引数がメモリ・マップ内のキー値に保存されます。
- **再生の同期化**：次の応答が到着するまでスクリプトの実行を停止します。VuGen は再生中にコールバックが発生した場所に同期ポイントを配置します。これは次のようにスクリプト内で警告として表されます。

```
#warning:  Code Generation Warning
        // ここで次の応答を待つ
        // 記録中の元のコールバック ...
```

同期化の一環として，スクリプトは GetNextResponse を呼び出して格納された値を取得します。

```
Vuser<Callback_Name>.GetNextResponse();
```

ダミー・コールバックの記録オプションの有効化

標準設定では，このオプションは有効になっています。

この記録オプションを有効にするには，次の手順を実行します。

1 ［**ツール**］ > ［**記録オプション**］を選択します。

2 ［**Microsoft .NET: 記録**］ノードを選択します。

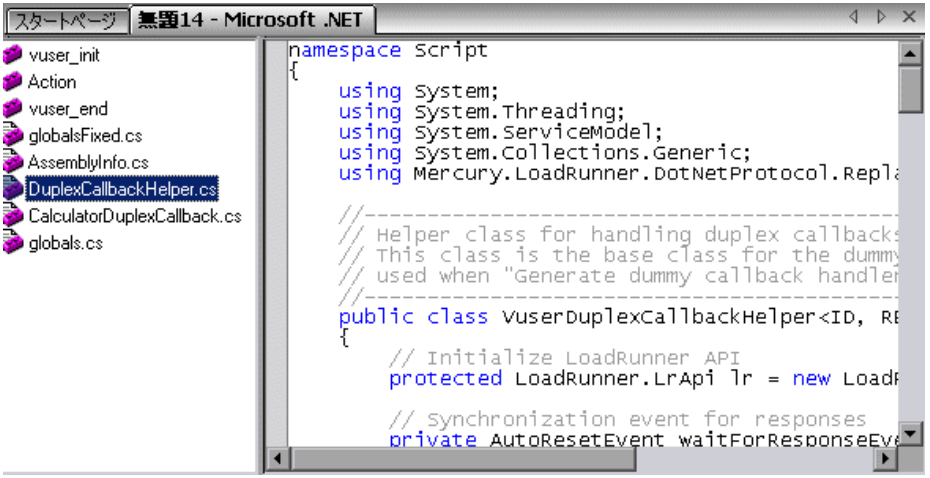
3 ［**ダミー コールバック ハンドラを生成する**］を選択します。

VuGen における双方向コールバックの実装

双方向通信ソリューションの一部として，VuGen は次の 2 つのサポート・ファイルを生成します。

- DuplexCallbackHelper.< 言語 >
- **Callback_Name.< 言語 >**

次の例は，双方向通信を使用する Calculator アプリケーション向けに生成されたファイルを示しています。



Helper ファイルは，双方向コールバック・ハンドラを使った作業のための汎用テンプレートとして使用されます。これはコールバック実装の基本クラスとなるものです。

2 つめの **Callback_Name** ファイルには，コールバック実装が含まれています。コールバック実装クラスの名前は **Vuser<xxxx>** です。ここで **xxxx** はコールバック・インタフェース名で，Helper ファイルで定義された **VuserDuplexCallbackHelper** クラスから継承します。VuGen はインタフェースごとに個別の実装ファイルを作成します。

このファイルは次の 2 つの主要なタスクを実行します。

- **応答の設定**：サーバからのデータをマップ内に格納します。取得を容易にするシーケンシャル ID をデータに付けて保存します。このメソッドは，コールバック・インタフェースの実装から呼び出されます。次のサンプル・コードは，**Result** という名前のコールバック・メソッドのダミー実装を示しています。メソッドの引数はオブジェクト配列としてマップに格納されています。

```
// ---------------------------------------------------------------------
        public virtual void Result(string operation, double result) {
            // ここにコールバック実装を追加し応答データを設定
            SetResponse(responseIndex++, new object[] {
                    operation,
                    result});
}
```

- **応答の取得**：次の応答の到着を待ちます。GetNextResponse の実装は，シーケンシャル・インデックスを使用してマップに格納された次の応答を取得するか，次の応答が到着するまで待ちます。

  スクリプトは，記録中に元のコールバック・ハンドラが呼び出された場所で GetNextResponse を呼び出します。このポイントで，スクリプトは警告を表します。

```
    // ここで次の応答を待つ
    // 記録中の元のコールバック
```

スクリプトでのコールバックの置き換え

ダミー・コールバック・オプションを有効にすると（標準設定は有効），VuGen は元の双方向コールバック・ハンドラをダミー実装に置き換えます。ダミー実装は「Vuser ＜**コールバック名**＞」と呼ばれます。元のコールバック・ハンドラの場所で，スクリプトは元のコールバック・ハンドラが置き換えられたことを示す警告を出力します。

デュアル HTTP バインディングの記録

アプリケーションでデュアル HTTP バインディングを使用している場合，HTTP は本来双方向プロトコルではないので，フレームワークはコールバックに渡される応答データの受信に標準ポートを使用します。アプリケーションの複数のインスタンスを実行しようとすると，同じポート番号を使用して実行できない可能性があります。VuGen には，元のクライアント・ベースのアドレスのポート番号を一意のポートに置き換えるオプションが用意されています。

［**一意のクライアント ベース アドレスを生成**］記録オプションを有効にすると，VuGen はアプリケーションが使用する通信のタイプを検査します。デュアル HTTP 通信 **WSDualHttpBinding** が検出されると，VuGen は Helper ファイル内の **FindPort** ユーティリティ（LrReplayUtils で提供）を実行し，コールバックのインスタンスごとに一意のポートを検索します。

標準設定では，このオプションは有効になっています。これは，前述のオプション［**ダミー コールバック ハンドラを生成**］を有効にしている場合にのみ適用されます。

このオプションを有効にすると，VuGen はスクリプト内に次のコードを生成します。

```
#warning:  Code Generation Warning
    // 元のクライアント・ベース・アドレスを一意のポート番号でオーバーライド
    DualProxyHelper.SetUniqueClientBaseAddress<XXXX>(YYYYY);
```

ダミー実装のカスタマイズ

環境に合わせて実装ファイルを変更できます。

推奨するカスタマイズは次のとおりです。

- タイムアウト
- キー識別子
- 戻り値
- 応答順序の取得
- ポートの検索

タイムアウト

コールバックが次の応答を待つ標準設定のタイムアウトは 60,000 ミリ秒，つまり 1 分です。特定のタイムアウトを使用するには，**GetNextResponse** への呼び出しを，次に示すような引数としてタイムアウトを取得するオーバーロード・メソッドに置き換えます。このメソッドは，コールバック実装ファイル <**Callback_Name**> の左側の表示枠に一覧表示されている **DuplexCallbackHelper** ファイルの下に反映されます。

```
// 次の応答を取得する
        // このメソッドはサーバからの応答を受信するまで
        // または指定されたタイムアウトを超えるまで待つ
        public virtual object GetNextResponse(int millisecondsTimeout) {
            return base.GetResponse(requestIndex++, millisecondsTimeout);
        }
```

すべてのコールバックの標準設定のしきい値を変更するには，**DuplexCallbackHelper** ファイルを変更します。

```
// レスポンスを待つ間の標準設定のタイムアウトしきい値
        protected int millisecondsTimeoutThreshold = 60000;
```

キー識別子

多くのアプリケーションは，要求や応答に相互に接続するキー識別子をデータに割り当てます。これにより，組み込みのインクリメンタル・インデックスの代わりに ID を使用して，マップからデータを取得できます。インデックスの代わりにキー識別子を使用するには，最初のベース・テンプレート・パラメータ **named ID** をキー識別子のタイプに置き換えて <**Callback_Name**> ファイルを変更します。たとえば，キー識別子が文字列の場合，次のように最初のテンプレート引数を **int** から **string** に変更します。

```
public class VuserXXX : VuserDuplexCallbackHelper<string, object>
```

また，GetNextResponse() の実装を削除し，ベース・クラスで定義された GetResponse(ID) への呼び出しに置き換えることもできます。

戻り値

標準設定では，VuGen は**片方向**通信をサポートしているので，実装コールバックは呼び出されたときに，値を返したり出力パラメータを更新したりしません。

```
public virtual void Result(string operation, double result) {
        // ここにコールバック実装を追加し応答データを設定
```

アプリケーションでコールバックが値を返す必要がある場合，ここに実装を挿入します。

応答順序の取得

VuGen の実装では，ブロッキング・メソッドが各応答を待ちます。これは，記録中にイベントが発生した順序（サーバがデータを返した順序）を反映します。この動作を変更して，応答を待たずに実行したり，ビジネス・プロセスの完了後にのみブロッキングを実装するようにできます。

ポートの検索

Helper ファイルの **FindPort** メソッドは，さまざまな実装で使用できる便利なユーティリティです。Helper クラスはスクリプトの複数のインスタンスを実行するために，このメソッドを使用して一意のポートを検索します。ほかのユーザ定義の実装にもこのユーティリティ・メソッドを利用できます。

クライアント・アプリケーションによってホストされたサーバの記録

システム内の通信がクライアントによってホストされたサーバである場合，双方向通信に対する VuGen の標準のソリューションは効果がありません。クライアントによってホストされたサーバ環境では，クライアントはサービスを起動しますが Framework 経由で通信しないので，本来の双方向通信ではありません。たとえばキューイングでは，クライアントはメッセージをサービスに送信し，応答キューを開き，応答を収集します。

クライアントによってホストされたサーバをエミュレートするには，前述のソリューションで説明したパターンを使用します。元の応答キューをダミー・コールバックに置き換え，必要に応じて同期を実行します。詳細については，HP サポートに問い合わせてください。

第 36 章

Microsoft .NET フィルタ

VuGen には，カスタマイズ可能ないくつかの組み込みフィルタが用意されています。

本章の内容

Microsoft .NET フィルタについて

記録用のフィルタには，記録およびスクリプトの生成時に包含または除外するアセンブリ，インタフェース，名前空間，クラス，またはメソッドを指定します。

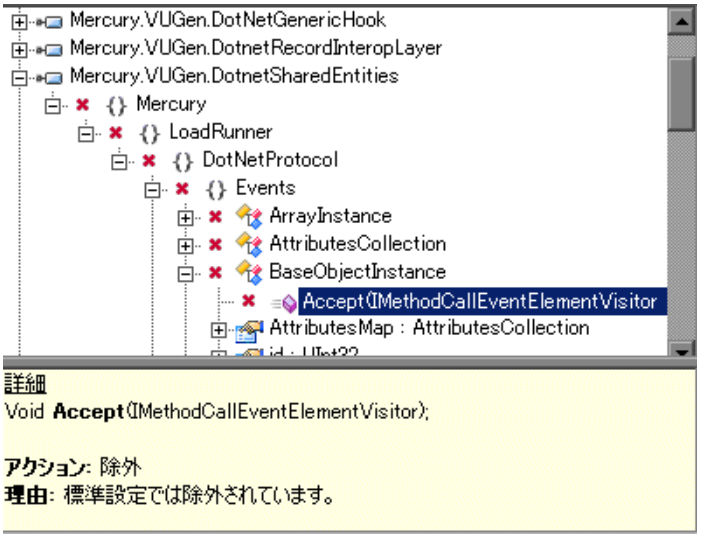
標準設定では，VuGen は .NET Remoting，ADO.NET，Enterprise Services，および WCF（Windows Communication Foundation) 用に組み込みシステム・フィルタを備えています。これらのフィルタは，標準の ADO.NET，Remoting，Enterprise Services，および WCF に対応するインタフェースを包含するように作成されています。VuGen ではカスタム・フィルタも作成できます。

カスタム・フィルタには次のいくつかの利点があります。

- **リモート**：.NET Remoting を対象とした作業を行う場合，リモート・メソッドに渡される引数を記録できるクラスを包含することが重要です。
- **欠落オブジェクト**：記録したスクリプトにアプリケーション内の特定のオブジェクトが記録されていない場合，フィルタを使用して欠落しているインタフェース，クラス，またはメソッドを包含することができます。

- **デバッグ**：エラーを返されてもその原因がわからない場合，問題の原因となった操作を絞り込むために，フィルタを使用してメソッド，クラス，またはインタフェースを除外できます。
- **保守性**：高水準のスクリプトを記録することで，スクリプトの保守と相関がしやすくなります。

フィルタ・マネージャを使用して既存のカスタム・フィルタを操作できます。フィルタ・マネージャには，アセンブリ，名前空間，クラス，メソッド，およびプロパティが色分けされたツリー階層として表示されます。



下の表示枠には，アセンブリ，名前空間，クラス，メソッド，プロパティ，またはイベントの説明が表示されます。また，それぞれの要素が包含されるのか除外されるのかがわかるほか，包含または除外の理由も示されます。

次の項では，フィルタをカスタマイズするべき状況とカスタマイズする方法について説明します。

- フィルタ設定についてのガイドライン
- 記録用フィルタの設定
- フィルタ・マネージャを使った作業

フィルタ設定についてのガイドライン

.NET アプリケーションをテストする目的は，クライアントからの要求に対するサーバの反応を確認することです。負荷テストの場合には，多数のユーザによる負荷にサーバがどのように応答するかを確認します。

.NET アプリケーションを記録すると，スクリプトにはローカル・ユーティリティや GUI インタフェースの呼び出しなど，サーバに影響を与えないメソッドの呼び出しが含まれる場合があります。通常，こうした呼び出しはテストの目的には関係ないので，フィルタによって除外するのが適切です。

組み込みのフィルタである .NET Remoting，ADO.NET，Enterprise Services，および WCF は，テストの目的にかなうサーバ関連のトラフィックのみを記録するように作成されています。ただし，状況によっては，.NET アプリケーションの呼び出しをキャプチャしたり不要な呼び出しを除外したりするために，ユーザ定義のカスタム・フィルタが必要になることがあります。フィルタ・マネージャを使用すれば，カスタム・フィルタを作成して，関係のない呼び出しの除外やサーバに関係する呼び出しのキャプチャができます。

記録に何を含めるかを判断できるように，テストを作成する前にアプリケーションについて理解を深め，アプリケーションの主要なクラスやメソッドを確認することをお勧めします。

アプリケーションのクラスに精通していない場合，フィルタにメソッドを包含するために，**Visual Studio** または**スタック・トレース**を使用して，アプリケーションに呼び出されるメソッドを調べることができます。VuGen では，アプリケーションによって呼び出されたすべてのメソッドをログに記録するスタック・トレース付きで記録することが可能です。

必要なメソッドとクラスを確認したら，フィルタ・マネージャを使用してそれらを包含するようにします。スクリプトを準備するときに，最適なフィルタにするためにフィルタを数回カスタマイズしなければならない場合があります。フィルタが最適であれば，スクリプトに無関係の多数の呼び出し取り込まずに必要なメソッドが記録されます。

ヒント：VuGen 内でスクリプトがまずはコンパイル（Shift+F5）できるようになるまで，つまり，正しい構文のテストとなるように，フィルタを可能なかぎり調整します。次に，VuGen でのスクリプトの実行が可能になるまでフィルタにカスタマイズを加えます。

フロー制御やメッセージ・ステートメントなど，スクリプトにコードを手作業で追加する場合は，必ず VuGen 内での実行が可能なスクリプトが得られてからにしてください。これは，スクリプトを再記録または再生成すると，手作業による変更がすべて失われるためです。

包含または除外する要素の指定

カスタム・フィルタを作成するとき，基本のフィルタとして適切な組み込みフィルタを選択するところから始めることをお勧めします。その後，次のどちらかの方法を使用してフィルタをカスタマイズできます。

- **トップ・ダウン方式**：この方式では，関係する名前空間を包含し，クライアント / サーバの動作に関与しない個別のクラスを除外します。アプリケーションに精通しており，MyDataAccessLayer.dll のような GUI 要素が関与しないクライアント / サーバの動作をすべて実装する明確なアセンブリを特定できる場合には，この方法をお勧めします。
- **ボトム・アップ方式**：標準設定のフィルタを使用し，個別のメソッドまたはクラスを包含するようにしてフィルタを調整していく方式です。明確なレイヤを特定できない場合，またはアプリケーションに精通していない場合は，この方式を使用します。すべての AUT アセンブリを追加して，その後余分なコンポーネントを 1 つずつ削除するようなことはしないでください。

次の項では，要素を包含または除外する際のガイドラインを示します。

- クラスを包含した結果，スクリプトに多くの無関係のメソッド呼び出しが含まれた場合は，フィルタに変更を加えて無関係のメソッドが除外されるか試してみます。
- スクリプトにクライアント/サーバの動作にかかわらない呼び出しが含まれた場合，フィルタからそのメソッドを除外します。
- 記録中，VuGen はこれまで遭遇したことのない構造の引数など，未知の入力引数を検出する場合があります。この引数がシリアライズをサポートしている場合，VuGen は引数を特別な形式で引数をファイルに保存することでその引数を**シリアライズ**します。再生中，VuGen は引数を**シリアル化解除**することでそれを復元します。
- VuGen は，フィルタによって包含されなかった引数として渡されたオブジェクトをシリアライズします。オブジェクトの構造と動作を追跡するために，オブジェクトをシリアライズされた形式で使用するのではなく，フィルタに含めることをお勧めします。スクリプト内のシリアライズされたオブジェクトを特定するには **LrReplayUtils.GetSerializedObject** メソッド，WCF 環境の場合は

LrReplayUtils.GetSerializedDataContract メソッドへの呼び出しを検索します。VuGen は，シリアライズされたオブジェクトを，**Serialization_1.xml**，**Serialization_2.xml** などのインデックス付きで，スクリプトの **¥data¥SerializedObjects** ディレクトリに XML ファイルとして格納します。

- 標準設定では，メソッドに対してルールが指定されていなければ，メソッドは除外されます。ただし Remoting 環境が有効な場合，標準設定では，明示的に包含していなくてもすべてのリモート呼び出しが追加されます。標準設定の動作を変更するには，ユーザ定義ルールを追加して，リモート・サーバを対象とする特定の呼び出しを除外できます。
- Remoting 呼び出しで渡された，フィルタによる包含対象になっていない種類の引数は，シリアル化メカニズムによって処理されます。引数がシリアライズされないようにするには，引数の構造と動作を記録するために，その種類の引数を明示的に包含するようにします。
- GUI 要素が関与する動作はすべて除外します。
- スクリプトのコンパイルに必要なユーティリティ用のアセンブリを包含するようにします。

要素を包含または除外する方法の詳細については，553 ページ「要素の包含および除外」を参照してください。

効果的なフィルタの定義

スクリプトを準備するときに，最適なフィルタにするためにフィルタを数回カスタマイズしなければならない場合があります。フィルタが最適であれば，スクリプトに無関係の多数の呼び出し取り込まずに必要なメソッドが記録されます。

効果的なフィルタを定義するには，次の手順を実行します。

1. いずれかの組み込みフィルタに基づいて新しいフィルタを作成します。不要なフィルタによって記録速度が低下する可能性があります。そのため AUT（テスト対象アプリケーション）が ADO.NET，Remoting，WCF，または Enterprise Services を使用しないことがわかっている場合は，そのオプションをクリアします。
2. 記録とコード生成の両方に対して，［**スタック トレース**］オプションを「True」に設定します。［記録オプション］を開き（CTRL+F7），［**記録**］ノードを選択します。［**デバッグ オプション : スタック トレース**］と［**コードの生成 : スタック トレースを表示する**］を有効にします。

3 アプリケーションを記録します。［**記録開始**］（CTRL+R）をクリックすると記録が始まり，［停止］（CTRL+F5）をクリックすると終了します。

4 スクリプトのステップを表示します。ステップを見てビジネス・ロジックを特定し相関を適用できる場合には，カスタム・フィルタを作成する必要はありません。しかし，スクリプトが非常に長かったり保守や相関が困難だったりする場合には，スクリプトのフィルタをカスタマイズする必要があります。

5 1 つ以上のクライアント・サーバ呼び出しをキャプチャまたはラップする呼び出しの中で，高水準のメソッドの識別を試みます。そのためには，Visual Studio で AUT ソース・ファイルを開くか（使用可能な場合），スクリプトのスタック・トレースを表示します。

6 関係するメソッドを包含するようにフィルタを設定します。あらかじめそれらのアセンブリを包含する必要がある場合があります。フィルタに要素を含めたり，フィルタから要素を除外したりする際のヒントについては，544 ページ「包含または除外する要素の指定」を参照してください。

7 アプリケーションを記録しなおします。フィルタを変更したら必ずアプリケーションを記録しなおしてください。

8 容易に保守と相関が行える簡単なスクリプトになるまで，手順 4 から 7 を繰り返します。

9 最適なスクリプトを作成したら，［**スタック トレース**］オプションを無効にしてスクリプトを再生成します。［記録オプション］を開き（CTRL+F7），［**記録**］ノードを選択します。［**デバッグ オプション : スタック トレース**］と［**コードの生成 : スタック トレースを表示する**］を無効にします。これにより，以降の記録のパフォーマンスが向上します。

10 スクリプトを相関させます。テストを正しく実行するために，相関を挿入して値をキャプチャし，スクリプトの以降の場所で使用する必要がある場合があります。組み込みの相関メカニズムの詳細については，525 ページ「Microsoft .NET スクリプトの相関」を参照してください。

記録用フィルタの設定

組み込みフィルタの .NET Remoting，ADO.NET，Enterprise Services，および WCF（Windows Communication Foundation）は，これらの環境の標準インタフェースを含むように設計されています。フィルタの利点の詳細については，543 ページ「フィルタ設定についてのガイドライン」を参照してください。

記録にフィルタを適用する場合，最初のステップは適切なフィルタを選択することです。［**フィルタ**］記録オプションを使用して，1 つ以上の環境フィルタを使用するか新しいフィルタを作成できます。

- ［**新規フィルタ**］：新しいフィルタを作成することを示します。
- ［**環境フィルタ**］：使用可能な環境フィルタ（.NET Remoting，ADO.NET，Enterprise Services，および WCF（Windows Communication Foundation of Framework 3.0））の一覧が表示されます。
- ［**カスタム フィルタ**］：現在のマシンで以前に作成したフィルタを表示します。

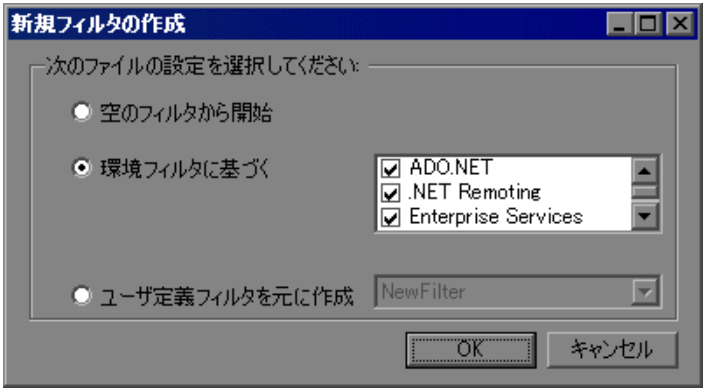
フィルタを作成したら，フィルタ・マネージャを使用してフィルタのプロパティを変更できます。詳細については，553 ページ「要素の包含および除外」を参照してください。

フィルタを指定するには，次の手順を実行します。

1 ［フィルタ］記録オプションを開きます。［**ツール**］＞［**記録オプション**］を選択し，［**Microsoft .NET : フィルタ**］ノードを選択します。

2 フィルタ・オプションとして，［**新規フィルタ**］，［**環境フィルタ**］または［**カスタム フィルタ**］を選択します。

3 新規フィルタの場合は［**作成**］をクリックします。［新規フィルタの作成］ダイアログ・ボックスが開きます。



4 いずれかのフィルタ・オプションを選択します。環境フィルタに基づく新しいフィルタを作成するには，環境フィルタの横にあるチェック・ボックスを 1 つ以上選択します。

5 ［**OK**］をクリックします。フィルタ・マネージャが開きます。

既存のフィルタの場合，フィルタ・マネージャを開くには，記録オプションのメイン・ダイアログ・ボックスで［**フィルタ マネージャ**］ボタンをクリックします。

フィルタに必要な変更を加えて保存します。詳細については，次を参照してください。

フィルタ・マネージャを使った作業

フィルタ・マネージャでは，環境フィルタ以外のフィルタの表示および変更ができます。環境フィルタは読み取り専用のため，表示はできますが変更はできません。

フィルタの管理および操作を行うには，フィルタ・マネージャ・ツールバーを使用します。

フィルタの管理

- ［**新規作成**］：［新規フィルタの作成］ダイアログ・ボックスを開きます。ここで空のフィルタを作成するか，既存のフィルタに基づく新しいフィルタを作成します。
- ［**保存**］：フィルタに加えた変更を保存します。
- ［**削除**］：選択したカスタム・フィルタを削除します。フィルタ・マネージャにより確認を求められます。

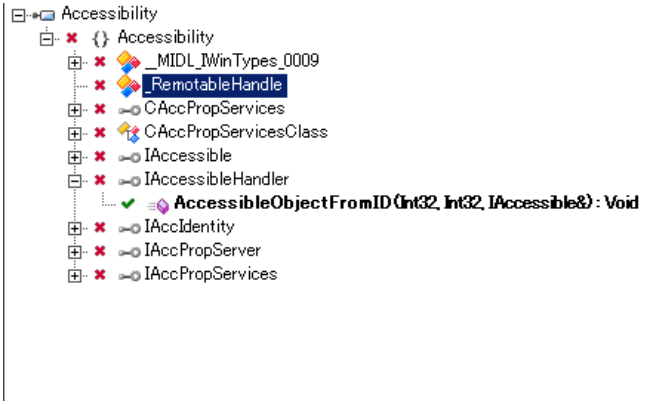
フィルタの操作

- ［**参照の追加**］：［参照の追加］ダイアログ・ボックスを開き，.NET Framework コンポーネントまたは Public Assemblies フォルダ内のアセンブリのリストを表示します。また，コンピュータを参照して，リストに表示されていないコンポーネントを検索できます。詳細については，552 ページ「参照の追加」を参照してください。
- ［**参照を削除**］：フィルタ・マネージャの中で選択されているアセンブリと，アセンブリに関連付けられているすべての要素を削除します。フィルタ・マネージャにより確認を求められます。
- ［**包含**］，［**除外**］，および［**リセット**］：アセンブリ，名前空間，クラス，またはメソッドを，包含または除外します。包含または除外のルールを標準設定の状態にリセットすることもできます。詳細については，553 ページ「要素の包含および除外」を参照してください。
- ［**戻る**］および［**進む**］：ユーザが訪れた前のツリー・ノードまたは次のツリー・ノードに移動します。

➤ [**影響のログ**]：選択したフィルタ用の Impact ログを開きます。Impact ログには，直前のアクションの影響を受けたツリー・ノードが表示されます。詳細については，555 ページ「Impact ログの表示」を参照してください。

さらに，標準の Windows キーの組み合わせと右クリック・メニューを使用して，フィルタのコピー，貼り付け，および名前変更ができます。

フィルタ・マネージャ・ツリーでは，記号を使用して要素とそのステータスを示します。



➤ 要素アイコンは，要素の種類（アセンブリ，名前空間，クラス，メソッド，構造体，プロパティ，イベント，またはインタフェース）を表します。

➤ 要素アイコンの横のチェック・マークまたは X は，要素が包含または除外されているかどうかを示します。

➤ 太字の要素は，明示的に包含または除外されていることを示します。これは，ユーザの手作業によって，または環境フィルタの定義済みのルールによって，包含または除外された結果です。太字のノードをリセットすると，元の太字ではない状態に戻ります。

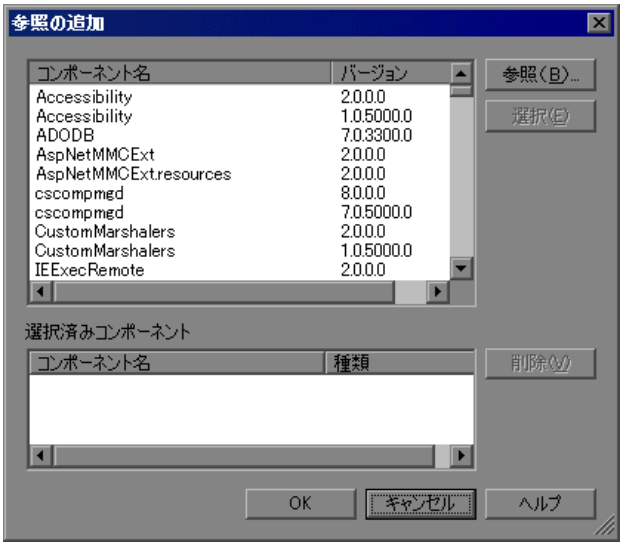
次の表に，各種の要素を表すアイコンを示します。

	アセンブリ		インタフェース
	ロードできなかったアセンブリ		メソッド
	一部がロードされたアセンブリ		静的メソッド
	クラス	{}	名前空間
	コンストラクタ		プロパティ
	静的コンストラクタ		静的プロパティ
	イベント		構造体
	静的イベント		

参照の追加

新しいフィルタを作成する場合，フィルタの動作を指定するために，フィルタにアセンブリ参照を追加する必要があります。また，参照を既存のカスタム・フィルタに追加することもできます。

［参照の追加］ダイアログ・ボックスを開くと，GAC（グローバル・アセンブリ・キャッシュ）のすべての public アセンブリが一覧表示されます。［**参照**］ボタンを使用すると，一覧表示されていない参照を追加できます。



下の表示枠の中で，［**種類**］カラムを見ることで参照を区別できます。GAC 内に存在する参照は種類が「**.NET**」で，GAC 内に存在しない参照は種類が「**File**」になります。

参照のリストには，同じアセンブリの異なるバージョンが含まれている場合があります。その場合は，テストに最適なバージョンを選択します。

フィルタ・マネージャを使用して参照を追加するには，次の手順を実行します。

1 ツールバーの［**参照の追加**］ボタンをクリックします。［参照の追加］ダイアログ・ボックスが開きます。

2 一覧表示されている項目のいずれかを追加するには，［**選択**］をクリックします。複数のコンポーネントを選択するには CTRL キーを押したまま，コンポーネントをクリックしていきます。下の表示枠に選択した参照が表示されます。

3 リストに表示されていないアセンブリを追加するには，［**参照**］をクリックしてファイル・システムまたはネットワーク上で参照を検索します。

4 フィルタに追加するすべての参照に対して前記の手順を繰り返します。

5 下の表示枠に表示された参照のリストを確認します。リストから項目を削除するには，下の表示枠で削除する項目を選択して［**削除**］をクリックします。

6 参照のリストの作成が完了したら，［**OK**］をクリックしてダイアログ・ボックスを閉じ，参照をフィルタに追加します。フィルタ・マネージャのツリーで，要素のリストの最後に参照が追加されます。選択した参照のいずれかが有効なアセンブリでない場合，エラー・メッセージが発行されます。

7 フィルタ・マネージャのツリーから参照を削除するには，親アセンブリ・ノードを選択して右クリック・メニューから［**参照の削除**］を選択するか，ツールバーの［参照の削除］ボタンをクリックします。

参照を追加したら，フィルタ・マネージャで表示して，正しいノードがすべて包含または除外されていることを確認してください。必要に応じて，特定の名前空間，クラス，またはメソッドを包含または除外することができます。詳細については，次の「要素の包含および除外」を参照してください。

要素の包含および除外

フィルタに参照を追加したら，フィルタ・マネージャのツリーでフィルタのすべてのノードを表示できます。特定の名前空間，クラス，またはメソッドを除外できます。あるいは，標準設定または別のルールによって除外されているものを包含するようにできます。フィルタ・マネージャの下部の表示枠の説明には，要素が包含または除外されている理由が示されます。

フィルタ・マネージャのツールバーには，要素を包含または除外するための次のボタンが表示されます。

➤ ［**包含**］：選択した要素を包含し，チェック・マークを付けます。親ノードを手作業で追加すると，ほかにルールが設定されていなければ，フィルタ・マネージャによりそのノードの下位の子要素が包含されます。たとえばクラスを追加すると，ユーザが明示的に除外したメソッドを除き，クラスのすべてのメソッドが包含されます。

➤ [**除外**]：選択した要素を除外し，X を付けます。ほかのルールによって包含されていないかぎり，子要素も除外されます。標準設定では，**クラス**を除外するとフィルタ・マネージャはそのクラスに **Exclude** 属性を適用しますが，削除されたクラスのメソッド内の動作を記録エンジンが記録することは可能です。ただし，**メソッド**を除外するとフィルタ・マネージャは **Totally Exclude** を適用し，除外されたクラスのメソッド内のすべての動作を記録エンジンが記録できないようにします。上級ユーザは，フィルタ・ファイル内でこれらの設定を変更できます。詳細については，556 ページ「フィルタ・ファイルに関する詳細情報」を参照してください。

➤ [**リセット**]：手作業で設定した包含または除外のルールを削除します。この場合，対象要素はほかの親要素の影響を受けます。

包含または除外のルールのプロパティは，次のとおりです。

- ➤ ルールは階層構造です。包含または除外するルールをクラスに追加すると，派生クラスはほかに指定がある場合を除き，同じルールに従います。
- ➤ クラスを対象とするルールは，クラスの public メソッド，派生クラス，および内部クラスにのみ影響します。
- ➤ 名前空間を対象とするルールは，すべてのクラスとその public メソッドに影響します。
- ➤ アセンブリを追加または削除しても，必ずしもアセンブリに含まれるクラスが影響を受けるとは限りません。アセンブリを削除しても，フィルタの階層構造に起因してアセンブリのメソッドが記録されることもあります。
- ➤ フィルタ作成時に考慮すべきこととして，**.cctor()** や **Dispose(bool)** などのいくつかのメソッドは，標準の階層ルールに従わないことが挙げられます。

注：親ノードをリセットしても，子ノードに適用した手作業の包含または除外のルールに優先しません。たとえば，メソッドを手作業で**除外**し，その後そのメソッドのクラス（標準設定ですべてのサブ・ノードを**包含**するクラス）をリセットしても，メソッドは除外されたままです。

プロパティおよびイベントは表示専用であり，フィルタ・マネージャを使用して包含または除外することはできません。また，システムに関連するいくつかの要素は保護されており，変更できません。

フィルタに要素を含めたり，フィルタから要素を除外したりする際のヒントについては，544 ページ「包含または除外する要素の指定」を参照してください。

アセンブリを追加または削除するには，552 ページ「参照の追加」の説明に従って［**参照の追加**］ボタンおよび［**参照の削除**］ボタンを使用します。次の項では，名前空間，クラス，およびメソッドを包含する方法について説明します。

要素を包含または除外するには，次の手順を実行します。

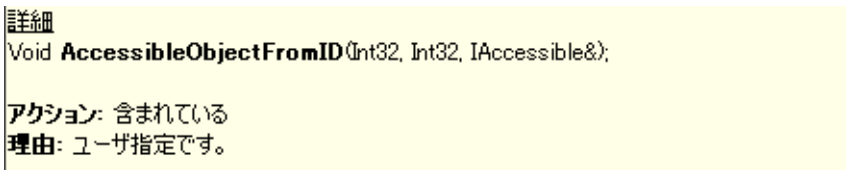
1 ツリー階層を展開し，名前空間，クラス，またはメソッドを選択します。

2 特定の要素を包含するには，その要素を選択して［**包含**］ボタンをクリックするか，右クリック・メニューで［包含］コマンドを使用します。

3 特定の要素を除外するには，［**除外**］ボタンをクリックするか，右クリック・メニューで［除外］コマンドを使用します。

4 要素を標準設定に戻すには，［**リセット**］ボタンをクリックするか，右クリック・メニューから［**リセット**］を選択します。

変更が適用されたことを確認するには，コンポーネントを選択して下の表示枠を確認します。

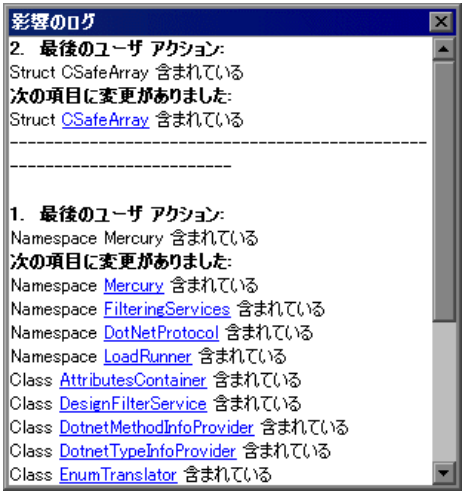


Impact ログの表示

Impact ログには，最後に加えた変更と，変更がフィルタに与えた影響が示されます。ユーザ・アクションが，最後に加えた変更を先頭に降順で一覧表示されます。

ログには，手作業で包含または除外されたことで影響を受けた要素ごとに，どのような影響を受けたかが示されます。フィルタ・マネージャ内の要素へのリンクも含まれます。

Impact ログを表示するには，フィルタ・マネージャのツールバーの［影響のログ］ボタンをクリックするか，［フィルタ マネージャ］ウィンドウで［**アクション**］＞［**影響のログ**］を選択します。



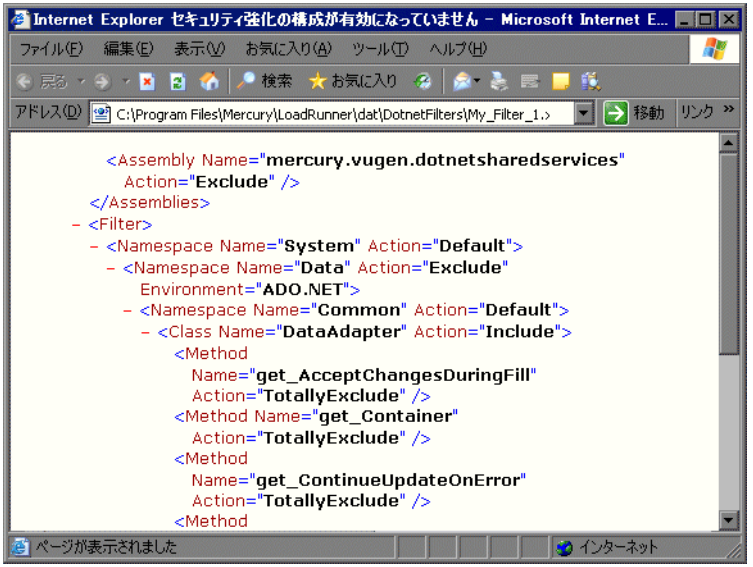
フィルタ・ファイルに関する詳細情報

フィルタ・マネージャのツリー階層には，public クラスと public メソッドのみが表示されます。public 以外のクラスやデリゲートは表示されません。

public 以外のクラスまたはメソッドを追加するには，フィルタの定義ファイルに手作業で入力します。

フィルタ定義ファイル「＜**フィルタ名**＞ **.xml**」は，インストール先の dat/DotnetFilters フォルダにあります。各要素に対して指定可能なアクション・プロパティは，**Include**，**Exclude**，または **Totally Exclude** です。詳細については，553 ページ「要素の包含および除外」を参照してください。

標準設定では，**クラス**を除外するとフィルタ・マネージャは **Exclude** を適用し，クラスを除外しますが，除外されたクラスによって生成された動作は包含されます。しかし，**メソッド**を除外すると **Totally Exclude** が適用され，すべての参照メソッドが除外されます。



たとえば，関数 A が関数 B を呼び出すとします。関数 A に **Excluded** が適用されている場合，サービスが関数 A を呼び出すと，スクリプトには関数 B の呼び出しが含まれます。しかし，関数 A に **Totally Excluded** が適用されている場合，スクリプトには関数 B の呼び出しは含まれません。関数 B は，関数 A からではなく直接呼び出された場合にのみ記録されます。

記録中 VuGen は，設定されたフィルタのバックアップ・コピーを，スクリプトの **data** フォルダに **RecordingFilterFile.xml** という名前で保存します。このファイルは，最後に記録を行ってからフィルタに変更を加えており，環境を復元する必要がある場合に役立ちます。

第 37 章

Web（HTTP/HTML，Click and Script）プロトコル

VuGen を使用して，クライアント・ブラウザ操作時のユーザ・アクションに基づく Web Vuser スクリプトを作成します。

本章の内容

Web レベル Vuser スクリプトの作成について

VuGen を使用して，Web Vuser スクリプトを作成できます。標準的なユーザ操作を実行し Web サイトをナビゲートしている間に，VuGen によってユーザのアクションが記録され Vuser スクリプトが生成されます。生成されたスクリプトを実行すると，Vuser によってインターネットにアクセスするユーザがエミュレートされます。

Vuser スクリプトを作成した後，VuGen を使用して，スクリプトをスタンドアロン・モードで実行します。実行に成功すれば，Vuser スクリプトをシナリオに組み込むことができます。Vuser スクリプトをシナリオに組み込む方法の詳細については，『**HP LoadRunner Controller ユーザーズ・ガイド**』を参照してください。

Vuser タイプによっては，記録されたスクリプトとイベントに関する情報が記載された，Microsoft Word 形式のビジネス・プロセス・レポートを作成できます。詳細については，『**第 1 巻 ― VuGen の使用**』の「ビジネス・プロセス・レポートの作成」を参照してください。

Web Vuser の紹介

会社の製品情報を表示する Web サイトがあったとします。このサイトには，見込み顧客がアクセスします。このサイトでは，多数のユーザ（200 ユーザなど）が同時にサイトにアクセスしたときでも，顧客の問い合わせに対する応答時間が必ず指定値（20 秒など）未満になるようにしたいとします。そのために，Vuser を使って，Web サーバが同時に複数の要求に対してサービスを提供する状況をエミュレートします。このとき各 Vuser は次のような操作を行うものと考えられます。

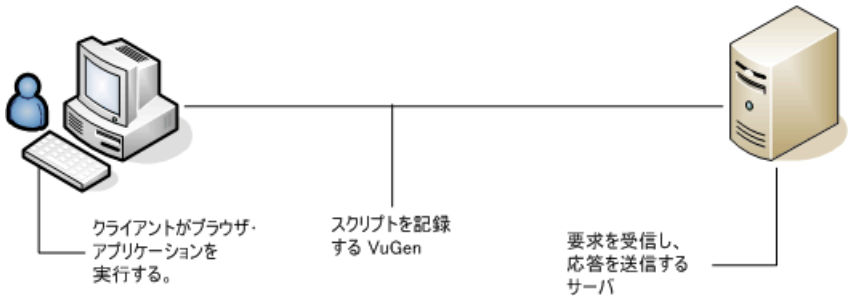
- ホーム・ページのロード
- 製品情報が掲載されているページへの移動
- クエリの送信
- サーバからの応答の待機

利用可能な複数のテスト用マシンに，数百の Vuser を分散配置できます。各 Vuser では API を通じてサーバにアクセスします。これにより，多数のユーザによる負荷の下でサーバのパフォーマンスを測定できます。

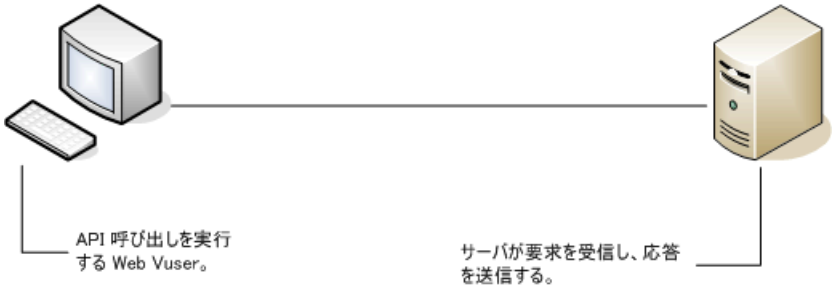
サーバ API の呼び出しを含むプログラムを Vuser スクリプトと呼びます。Vuser スクリプトでは，ブラウザ・アプリケーションと，ブラウザが実行するすべてのアクションをエミュレートします。Controller を使用して，1 つのスクリプトを複数の Vuser に割り当てることができます。これらの Vuser によってスクリプトが実行され，Web サーバのユーザ負荷がエミュレートされます。

Web Vuser 技術について

VuGen では，ブラウザと Web サーバの間のやり取りを記録することによって，Web Vuser スクリプトを作成します。VuGen でシステムのクライアント側（ブラウザ）を監視し，サーバとの間で送受信されるすべての要求を追跡します。



記録された Vuser スクリプトを実行するとき，Vuser はサーバと直接通信し，クライアント・ソフトウェアに依存しません。Vuser スクリプトでは，クライアント・ソフトウェアを使わず，API 関数を使って Web サーバへの呼び出しを直接実行します。



Web Vuser のタイプの選択

新しい Web Vuser スクリプトを作成する場合，次のいずれかのタイプの Web Vuser を選択できます。

- Web（Click and Script）
- Web（HTTP/HTTPS）

Web（Click and Script）

Web（Click and Script）Vuser は，ユーザ・アクション GUI レベルで Web セッションを記録するためのソリューションです。VuGen は，Web インタフェースを対象としたアクションを直感的に表現する GUI レベルのスクリプトを作成します。たとえば，ボタンをクリックして情報を送信すると **web_button** 関数が生成され，エディット・ボックスにテキストを入力すると **web_edit_field** 関数が生成されます。

Web（Click and Script）Vuser は，JavaScript などのクライアント側の非 HTML コードをサポートします。VuGen は，Web ページでのユーザ・アクションを正確にエミュレートする直感的なスクリプトを作成し，必要な Javascript コードを実行します。

Web（Click and Script）Vuser は大部分の相関を自動的に処理するため，スクリプト作成時間が短縮されます。ほとんどの場合，相関のルールを定義したり，記録後に手作業で相関を実行したりする必要はありません。

また，Web（Click and Script）Vuser では，スクリプトについてまとめた詳細なビジネス・プロセス・レポートを生成することもできます。

たとえば，ボタンをクリックしてデータを送信すると，VuGen によって **web_button** 関数が生成されます ボタンが画像の場合は，**web_image_submit** が生成されます。次の例では，ユーザは［login］ボタンをクリックしています。

```
...
web_image_submit("Login",
        "Snapshot=t4.inf",
        DESCRIPTION,
        "Alt=Login",
        "Name=login",
        "FrameName=navbar",
        ACTION,
        "ClickCoordinates=31,6",
        LAST);}
```

次の項では，ユーザが「Manage Assets」分岐の下にある Asset ExpressAdd プロセスに移動していることを示します。ユーザは，対象となる分岐のテキスト・リンクをクリックすることで移動します。**web_text_link** 関数が生成されます。

```
web_text_link("Manage Assets_2",
        DESCRIPTION,
        "Text=Manage Assets",
        "Ordinal=2",
        "FrameName=main",
        ACTION,
        "UserAction=Click",
        LAST);

web_text_link("Use",
        DESCRIPTION,
        "Text=Use",
        "FrameName=main",
        ACTION,
        "UserAction=Click",
        LAST);

web_text_link("Asset ExpressAdd",
        DESCRIPTION,
        "Text=Asset ExpressAdd",
        "FrameName=main",
        ACTION,
        "UserAction=Click",
        LAST);
```

次の例では，**web_list** によってリスト項目の選択がエミュレートされています。

```
...
web_list("Year",
        DESCRIPTION,
        "Name=Year",
        "FrameName=CalFrame",
        ACTION,
        "Select=2000",
        LAST);
```

画像マップに関連付けられている画像をクリックすると，VuGen によって **web_map_area** 関数が生成されます。

```
web_map_area("map2_2",
        DESCRIPTION,
        "MapName=map2",
        "Ordinal=20",
        "FrameName=CalFrame",
        ACTION,
        "UserAction=Click",
        LAST);
```

注：Web（Click and Script）Vuser は，アプレットおよび VBScript をサポートしません。テスト対象 Web サイトにこれらの項目が含まれている場合は，Web（HTTP/HTML）ユーザを使用します。

Web（HTTP/HTTPS）

Web（HTTP/HTML）スクリプトを記録すると，VuGen は，ブラウザとサーバ間の HTTP トラフィックを記録します。スクリプトには，記録されたトラフィックに関する詳細情報が含まれます。

Web（HTTP/HTML）Vuser には，2 つの記録レベルがあります。**HTML ベースのスクリプト**と **URL ベースのスクリプト**の 2 つです。これらのレベルにより，Vuser スクリプトの生成時に記録する情報および使用する関数を指定できます。記録レベルの選択の詳細については，『**第 1 巻 ー VuGen の使用**』の第 20 章「Web Vuser の記録オプションの設定」を参照してください。

ヒント：JavaScript を使用するアプリケーションを含むほとんどのアプリケーションに対しては，Web（Click and Script）Vuser を使用します。アプレットまたは VBScript を使用するブラウザ・アプリケーション，または非ブラウザ・アプリケーションの場合は，Web（HTTP/HTML）Vuser を使用します。

Web Vuser スクリプト入門

本項では，Web Vuser スクリプトを作成する工程の概要を説明します。

Web Vuser スクリプトを作成するには，次の手順を実行します。

1 VuGen を使って新しいスクリプトを作成します。

VuGen の［**スタート ページ**］タブをクリックし，［**ファイル**］＞［**新規作成**］をクリックします。シングル・プロトコル・モードまたはマルチ・プロトコル・モードで，［**e ビジネス**］カテゴリから Web（Click and Script）または Web（HTTP/HTML）Vuser スクリプトを選択します。

新規スクリプトの作成の詳細については，『**第 1 巻 ― VuGen の使用**』の「VuGen を使った記録」を参照してください。

2 記録オプションを設定します。

記録オプションを設定します。記録オプションの設定については，『**第 1 巻 ― VuGen の使用**』の第 16 章「選択したプロトコルの記録オプション」を参照してください。

記録レベルの選択の詳細については，『**第 1 巻 ― VuGen の使用**』の第 20 章「Web Vuser の記録オプションの設定」を参照してください。

Web（Click and Script）固有のオプションの詳細については，『**第 1 巻 ― VuGen の使用**』の第 17 章「Click and Script 記録」を参照してください。

3 ブラウザ・セッションを記録します。

Web サイトをナビゲートしている間のアクションが記録されます。

新規スクリプトの作成の詳細については，『**第 1 巻 ― VuGen の使用**』の「VuGen を使った記録」を参照してください。

4 記録した Vuser スクリプトの機能を拡張します。

トランザクション，ランデブー・ポイント，チェック，サービス・ステップを挿入して，Vuser スクリプトを拡張します。

詳細については，第 40 章「Web（HTTP/HTML，Click and Script）のテキストと画像の検証」，第 41 章「Web とワイヤレス Vuser スクリプトの変更」，および第 42 章「Web（HTTP/HTML）の相関ルール」を参照してください。

5 パラメータを定義します（任意）。

スクリプトに記録された固定値の代わりにパラメータを定義します。固定値をパラメータで置き換えれば，その値を毎回変えて，同じ Vuser のアクションを何度でも繰り返せます。

詳細については，『**第 1 巻 ー VuGen の使用**』の「パラメータの作成」を参照してください。

6 実行環境を設定します。

実行環境を設定することによって，スクリプト実行中の Vuser の動作を制御します。この設定には，一般的な実行環境の設定（反復，ログ，思考遅延時間，一般情報）と Web 関連の設定（プロキシ，ネットワーク，HTTP の詳細）が含まれます。

詳細については，『**第 1 巻 ー VuGen の使用**』の「実行環境の設定」を参照してください。

7 相関を実行します。

Web（HTTP/HTML）スクリプトの場合，Vuser スクリプトの相関を探し出し，Vuser のメカニズムの 1 つを使って，その相関を実現します。

自動相関の設定方法の詳細については，第 42 章「Web（HTTP/HTML）の相関ルール」を参照してください。記録後の相関の詳細については，第 43 章「Web（HTTP/HTML）ー 記録後の相関」を参照してください。

8 VuGen で Vuser スクリプトを実行してデバッグします。

VuGen から Vuser スクリプトを実行して，スクリプトが正しく実行されることを確認します。

詳細については，『**第 1 巻 ー VuGen の使用**』の「スタンドアロン・モードでの Vuser スクリプトの実行」と『**第 1 巻 ー VuGen の使用**』の「テスト結果の表示」を参照してください。

スクリプトを作成した後，スクリプトを自分の環境（LoadRunner シナリオ，Performance Center 負荷テスト，または Business Process Monitor プロファイル）に組み込みます。詳細については，**HP LoadRunner Controller**，**HP Performance Center**，または **HP Business Availibility Center** のドキュメントを参照してください。

Web セッションの記録

Web セッションを記録するとき，VuGen によって Web ブラウザで実行されるすべてのアクションが監視されます。ハイパーリンク・ジャンプ（ハイパーテキストとハイパーグラフィックの両方）やフォーム送信などもアクションに含まれます。記録中，VuGen によって記録対象のアクションが Web Vuser スクリプトに保存されます。

作成する各 Vuser スクリプトには，少なくとも次の 3 つのセクションがあります。**vuser_init**，1 つ以上の **Actions**，および **vuser_end** です。VuGen で記録中に，記録対象の関数を挿入する対象となるスクリプトのセクションを選択できます。通常，vuser_init と vuser_end セクションは，サーバのログオンとログオフ手続きの記録に使います。これらのセクションは Vuser スクリプトを何度も反復するときに，反復されない部分です。

したがって，ブラウザ・セッションを完全な形で反復するためには，Web セッションを Actions セクションに記録する必要があります。

Web Vuser スクリプトの Java への変換

VuGen には，Web Vuser 用に作成したスクリプトを Java Vuser に変換するユーティリティがあります。これにより，Web および Java の Vuser の両方を混合した Vuser スクリプトを作成できます。

Web Vuser スクリプトを Java Vuser スクリプトに変換するには，次の手順を実行します。

1 空の Java Vuser スクリプトを作成して保存します。

2 空の Web Vuser スクリプトを作成して保存します。

3 標準の HTML/HTTP 記録オプションで Web セッションを記録します。

4 Vuser スクリプトを再生します。正常に再生できたら，スクリプト全体を切り取り，テキスト・ドキュメントに貼り付け，テキストとして **.txt** ファイルに保存します。テキスト・ファイルでパラメータの括弧を Web 形式の「**"{ }"**」から Java 形式の「**"< >"**」に変更します。

5 DOS コマンド・ウィンドウを開いてお使いの製品の **dat** ディレクトリに移動します。

6 次のコマンドを入力します。

＜アプリケーションのインストール先＞ ¥bin¥sed -f web_to_java.sed filename > outputfilename

ここで **filename** には先に保存したテキスト・ファイルのフル・パスとファイル名を指定し，**outputfilename** には出力ファイルのフル・パスとファイル名を指定します。

7 出力ファイルを開いて，ファイルの内容を Vuser スクリプトの Action 部分の適切な場所にコピーします。内容を空のユーザ定義 Java テンプレート（Java Vuser タイプ）に貼り付ける場合は，public int action() を含む行を次のように変更します。

public int action() throws Throwable

記録を行うことによって作成する Java Vuser（RMI および CORBA）では，この変更は自動的に行われます。

通常の Java スクリプトと同様に，Vuser スクリプトのパラメータ化と相関を行った後，スクリプトを実行して動作を確認します。

プッシュ技術に対するサポート

Vuser スクリプトはステップを順番に実行します。前のステップが完了するまで次のステップを開始できません。またステップは，要求されたすべてのデータのダウンロードが終了するまで完了できません。Web サイトの中には，ダウンロードの完了を目的としないプッシュ技術を使用するものがあります。データは最新化されるとそのつど定期的にダウンロードされます。

Vuser スクリプトに，プッシュ技術を使用するリソースにアクセスするステップが含まれている場合，Vuser スクリプトはタイムアウトが発生するまでそのステップで停止します。サイトは設計したとおりに機能しているので，これは「偽」のタイムアウトです。

熟練したユーザであれば，ステップの条件を完了に変更してこの問題に対処できます。詳細については，『**Online Function Reference**』（英語版）（［**ヘルプ**］＞［**関数リファレンス**］）を参照してください。

第 38 章

Web（Click and Script）のヒント

記録，再生，および拡張のヒントは，Web（Click and Script）スクリプト開発のガイドラインを示します。

本章の内容

以降の情報は，Web（Click and Script），AJAX（Click and Script），SAP（Click and Script），Oracle Web Applications 11i，および PeopleSoft Enterprise の各プロトコルを対象とします。

記録に関する問題

次の項に，記録に関して最もよく起こる問題をリストアップします。

Firefox がサポートされていない

Web（Click and Script）でサポートされているのは Internet Explorer のみです。Firefox でのブラウザの動作を記録するには，Web（HTTP/HTML）プロトコルを使用します。

アプリケーションの記録時の動作が記録していないときと異なる

アプリケーションの動作が，記録をしているときとしていないときで異なる場合，その記録の問題が Web（Click and Script）に固有の問題かどうかを確認する必要があります。たとえば，Web ページが読み込まれない，内容の一部が欠落している，ポップアップ・ウィンドウが表示されないなどの症状があります。

新しい Web（HTTP/HTML）スクリプトを作成して，記録をしなおします。

Web（HTTP/HTML）にして記録に失敗したら，ソケット・レベルの記録を無効にすることをお勧めします（574 ページ「ソケット・レベルの記録を無効にする」を参照）。

この問題は，イベント・リスナーが原因の可能性があります。［**Web イベント設定**］記録オプションの設定を色々変えてみて該当するイベント・リスナーを無効にして，セッションを Web（Click and Script）ユーザとして記録しなおします。

イベント・リスナーを無効にするには，次の手順を実行します。

➤ ［記録オプション］ダイアログ・ボックスを開きます。［**ツール**］＞［**記録オプション**］を選び，［**GUI プロパティ：Web イベント設定**］ノードを選択します。

➤ ［**ユーザ定義設定**］をクリックして［**Web オブジェクト**］ノードを展開します。オブジェクトを選択します。

➤ ［**記録**］カラムのリストの中で，該当する Web オブジェクトについて［**無効**］を選択します。それでも記録が正常に行われない場合は，無効にしたリスナーを有効にして，別のリスナーを無効にしてみます。記録が正常に行われるまで，これらの手順を繰り返します。

動的メニューの操作が記録されない

動的メニューとは，選択する場所に応じて動的に変化するメニューのことです。動的メニューの操作が記録されなかった場合は，［高］のイベント設定モードで再度記録してみてください。

設定レベルを［高］に設定するには，次の手順を実行します。

➤ ［記録オプション］ダイアログ・ボックスを開きます。［**ツール**］＞［**記録オプション**］を選び，［**GUI プロパティ：Web イベント設定**］ノードを選択します。

➤ スライダを［**高**］の位置まで移動します。

一部のユーザ・アクションが記録されない

ブラウザ内で Java アプレットが動作していないか確認してください。動作していない場合は，Web（HTTP/HTML）プロトコルを利用してスクリプトを記録してください。

記録に関するヒント

キーボードではなく，マウスを使用する

オブジェクトをマウスでクリックするほうが，キーボードを使用するよりも望ましいです。記録中は，ブラウザの表示枠内にある GUI オブジェクトだけを使用してください。ブラウザのアイコン，コントロール，［停止］ボタン，または［**表示**］＞［**最新の情報に更新**］などのメニュー項目は使用しないでください。ただし，［最新の情報に更新］，［ホーム］，［戻る］，［進む］の各ボタンおよびアドレス・バーは使用できます。

既存のスクリプトに上書きして記録しない

既存のスクリプトではなく，新規の作成したスクリプトに記録するのが最善です。

ショートカット・メニューを使用しない

記録中はショートカット・メニューの使用を避けます。ショートカット・メニューとは，右クリック・メニューなど，グラフィカル・ユーザ・インタフェースの項目をクリックしたときに現れるメニューのことです。

記録時に別のブラウザで作業をしない

記録時には，VuGen によって開かれたブラウザ・ウィンドウ以外のブラウザ・ウィンドウでは作業をしないようにします。

ダウンロードは待機する

すべてのダウンロードが完了するまで待ってから，ボタンをクリックしたり，テキスト・フィールドに入力を行ったりするなどのアクションを起こすようにします。

ページの読み込みを待機する

記録時には，ページの読み込みが完全に終わるのを待ってから次のステップを実行するのが最善です。ページ全体の読み込みが完了するまで待たなかった場合は，スクリプトを記録しなおしてください。

開始ページに移動する

アクションの最後のページに，反復の最初に利用可能であったリンクやボタンが含まれていない場合，次の反復は失敗します。たとえば，最初のページに「**Book A Flight**」というテキスト・リンクがある場合，ビジネス・プロセスの最後に同じリンクが表示されているように，記録の最後に，適切なページに移動するようにします。

イベント設定レベルは高くする

［**高**］のイベント設定レベルを使用してビジネス・プロセスを記録しなおします。イベント設定レベルの変更の詳細については，572 ページ「動的メニューの操作が記録されない」を参照してください。

ソケット・レベルの記録を無効にする

ソケット・レベルのメッセージをキャプチャすると，アプリケーションが中断することがあります。ほとんどの記録では，ソケット・レベル・データは必要ありません。ソケット・レベル・データが記録されないようにするには，記録オプションでこのオプションを無効にします。詳細については，Click and Script を使った記録に関する項を参照してください。

［描画関連のプロパティ値の記録］を有効にする

アプリケーションのクライアント側のスクリプトがスタイル設定アクティビティを多用する場合，スクリプトの記録を開始する前に［**描画関連のプロパティ値の記録**］を有効にします。たとえば，**offsetTop** などの追加 DOM プロパティを記録するには，このオプションを有効にします。このオプションを有効にすると，記録速度が低下することがあります

［描画関連のプロパティ値の記録］を有効にするには，次の手順を実行します。

➤ ［記録オプション］ダイアログ・ボックスを開きます。［**ツール**］＞［**記録オプション**］を選び，［**GUI プロパティ：詳細**］ノードを選択します。

再生に関する問題

GUI オブジェクトが見つからない

エラーは 2 回目の反復の先頭で起きますか？

このエラーが 2 回目の反復の先頭で起きる場合，その原因はおそらく最初の反復のときに存在していた開始ページが，2 回目の反復のときに存在しなかったことです。アクションの最後のページに，反復の最初に利用可能であったリンクやボタンが含まれていない場合，次の反復は失敗します。たとえば，最初のページに「**Book A Flight**」というテキスト・リンクがある場合，ビジネス・プロセスの最後に同じリンクが表示されているように，適切なページに移動するようにします。

非 ASCII 文字を含んだテキスト・リンクですか？

非 ASCII 文字のときにこの問題が生じる場合，VuGen に対してデータを適切な文字セットに変換するように設定します。

Windows マシン上でデータ変換を有効にするには，次の手順を実行します。

1 ［実行環境設定］ダイアログ・ボックスを開きます。［**仮想ユーザ**］＞［**実行環境の設定**］を選び，［**インターネット プロトコル : プリファレンス**］ノードを選択します。

2 ［**オプション**］をクリックして［詳細オプション］ダイアログ・ボックスを開きます。

3 ［Web (Click and Script)］＞［一般オプション］で［**HTTP による文字セットの変換**］を探して，それを［**Yes**］に設定します。

UNIX マシン向けに UTF-8 変換を有効にするには，次の手順を実行します。

1 ［実行環境設定］ダイアログ・ボックスを開きます。［**仮想ユーザ**］＞［**実行環境の設定**］を選び，［**インターネット プロトコル : プリファレンス**］ノードを選択します。

2 ［**オプション**］をクリックして［詳細オプション］ダイアログ・ボックスを開きます。

3 ［一般オプション］で［**UTF-8 から、または UTF-8 への変換**］オプションを探して，それを［**Yes**］に設定します。

あるいは，リンクが見つからなかった場合に表示される代替候補のリストを確認します。表示されている ¥xA0 のような 16 進表現のエスケープ・シーケンスや非標準の形式のテキストをそのまま入力します。

アプリケーションの中で当該アクション・シーケンスを 2 回実行することはできますか？

データベースから特定のユーザを削除するなど，プロセスによっては実行できるのは一度だけというものもあります。この場合，1 回目の反復以降，アクションはもはや有効ではなくなるので，再生は失敗します。対象ビジネス・プロセスを，記録しなおさずに，同じデータを使ってアプリケーションの中で 2 回以上繰り返して実行できることを確認します。

画像の「Id」,「Name」,「Alt」の各プロパティは空でしたか？

ツリー・ビューの中で，直前の画像ステップをダブル・クリックしてプロパティを表示します。「**Id**」,「**Name**」,「**Alt**」の各プロパティが空の場合，「**Src**」プロパティにファイル名を指定するなど，画像を識別するためのほかの情報を指定します。

あるいは，［**Ordinal**］引数を追加して，ページ内での画像の出現番号を指定します。［**Ordinal**］引数は，ほかの識別引数で一意に識別できない場合に，ページ上の各画像を一意に識別します。詳細については，『**Online Function Reference**』（英語版）（［**ヘルプ**］＞［**関数リファレンス**］）を参照してください。

ステップの記述が変わりましたか？

［出力］ウィンドウの［再生ログ］を確認して，問題のステップのオブジェクト一覧を探します。場合によっては，オブジェクト記述が実行ごとに変わっていることがあります。

解決の方法はいくつかあります。

- 新しい値が安定している場合には，［スクリプト ビュー］を開いて，ステップの DESCRIPTION 引数の値を手作業で変更します。
- 記述が実行のたびに変わる場合は，DESCRIPTION 引数の中で正規表現を使用します。詳細については，『**Online Function Reference**』（英語版）（［**ヘルプ**］＞［**関数リファレンス**］）を参照してください。
- あるいは，Name など，問題となっているオブジェクト記述プロパティを，Ordinal プロパティに置き換えます。詳細については，『**Online Function Reference**』（英語版）（［**ヘルプ**］＞［**関数リファレンス**］）を参照してください。

記録時にページの読み込みは完了しましたか？

記録時には，ページの読み込みが完全に終わるのを待ってから次のステップを実行するのが最善です。ページ全体の読み込みが完了するまで待たなかった場合は，スクリプトを記録しなおしてください。

再生に関するヒント

問題のトラブルシューティングに次のヒントを役立ててください。

並べ替えをしない

記録されたスクリプト内のステートメントは並べ替えないようにします。また，ある Action から別の Action へコードのセグメントをコピーすることも推奨されません。

非 ASCII 文字は変換する

リンクに非 ASCII 文字が含まれている場合，VuGen に対してデータを UTF-8 形式との間で変換を行うように設定します。

UTF-8 変換を有効にするには，次に手順を実行します。

➤［記録オプション］ダイアログ・ボックスを開きます。［**仮想ユーザ**］＞［**実行環境の設定**］を選び，［**インターネット プロトコル：プリファレンス**］ノードを選択します。

➤［**オプション**］をクリックして［詳細オプション］ダイアログ・ボックスを開きます。

➤［**UTF-8 から、または UTF-8 への変換**］オプションを探して，それを［**Yes**］に設定します。

あるいは，リンクが見つからなかった場合に表示される代替候補のリストを確認します。表示されている ¥xA0 のような 16 進表現のエスケープ・シーケンスや非標準の形式のテキストをそのまま入力します。

同じアクション・シーケンスを 2 回実行してみる

データベースから特定のユーザを削除するなど，プロセスによっては実行できるのは一度だけというものもあります。この場合，1 回目の反復以降，アクションはもはや有効ではなくなるので，再生は失敗します。対象ビジネス・プロセスを，記録しなおさずに，同じデータを使ってアプリケーションの中で 2 回以上繰り返して実行できることを確認します。

一意の画像プロパティを設定するようにする

ツリー・ビューの中で，直前の画像ステップをダブル・クリックしてプロパティを表示します。「**Id**」，「**Name**」，「**Alt**」の各プロパティが空の場合，「**Src**」プロパティにファイル名を指定するなど，画像を識別するためのほかの情報を指定します。

あるいは，［**Ordinal**］引数を追加して，ページ内での画像の出現番号を指定します。［**Ordinal**］引数は，ほかの識別引数で一意に識別できない場合に，ページ上の各画像を一意に識別します。詳細については，『**Online Function Reference**』（英語版）（［**ヘルプ**］＞［**関数リファレンス**］）を参照してください。

ステップの記述を確認する

「**GUI Object is not found**」というエラーが生じる場合，［出力］ウィンドウの［再生ログ］を確認して，問題のステップのオブジェクト一覧を探します。場合によっては，オブジェクト記述が実行ごとに変わっていることがあります。

解決の方法はいくつかあります。

- 新しい値が安定している場合には，［スクリプト ビュー］を開いて，ステップの DESCRIPTION 引数の値を手作業で変更します。
- 記述が実行のたびに変わる場合は，DESCRIPTION 引数の中で正規表現を使用します。詳細については，『**Online Function Reference**』（英語版）（［**ヘルプ**］＞［**関数リファレンス**］）を参照してください。
- あるいは，Name など，問題となっているオブジェクト記述プロパティを，Ordinal プロパティに置き換えます。詳細については，『**Online Function Reference**』（英語版）（［**ヘルプ**］＞［**関数リファレンス**］）を参照してください。

その他の問題

JavaScript でメモリ不足のエラー

実行環境の設定で JavaScript 用のメモリを増やします。

JavaScript 用のメモリ容量を増やすには，次の手順を実行します。

1 ［記録オプション］ダイアログ・ボックスを開きます。［**仮想ユーザ**］＞［**実行環境の設定**］を選び，［**インターネット プロトコル : プリファレンス**］ノードを選択します。

2 ［**オプション**］をクリックして［詳細オプション］ダイアログ・ボックスを開きます。

3 ［**メモリ管理 : JavaScript ランタイム メモリのサイズ (KB)**］オプションと［**メモリ管理 : JavaScript スタック メモリのサイズ (KB)**］オプションを探します。

4 メモリ・サイズを 512 以上の値に増やします。

VuGen に JavaScript エラーが表示される

VuGen の［再生ログ］に JavaScript エラーが表示される場合，Internet Explorer のスクリプト・エラーを有効にすることで，JavaScript のコードそのものにエラーがないことを確認します。

スクリプトのエラーを表示させるには，次の手順を実行します。

1 Internet Explorer（IE）を開きます。［**ツール**］＞［**インターネット オプション**］を選択し，［**詳細設定**］タブを選択します。

2 ［**ブラウズ**］セクション内の［**スクリプト エラーごとに通知を表示する**］を有効にします。

3 アプリケーションを IE の中で再度実行します。IE にスクリプト・エラーが表示された場合，JavaScript アプリケーションに問題があることになります。アプリケーションを修正することが不可能な場合には，該当する再生エラーは無視することができます。

パラメータ化後に生じる問題

値をパラメータ化した後に問題に遭遇した場合は，値がアプリケーションに対して有効であることを確認してください。パラメータに使用されている値を使用してビジネス・プロセスを実行し，アプリケーションがその値を受け入れることを確認します。

スタイル設定アクションを利用するアプリケーションに関する問題

アプリケーションのクライアント側のスクリプトがスタイル設定アクティビティを多用する場合，［**描画関連のプロパティ値の記録**］を有効にしてから，スクリプトを記録しなおすべきです。これにより，追加の DOM オブジェクトの記録が可能となります。

［描画関連のプロパティ値の記録］を有効にするには，次の手順を実行します。

1 ［記録オプション］ダイアログ・ボックスを開きます。［**ツール**］＞［**記録オプション**］を選び，［**GUI プロパティ：詳細**］ノードを選択します。

2 ［**描画関連のプロパティ値の記録**］を有効にします。スクリプトを記録しなおします。

その他のヒント

問題のトラブルシューティングに次の追加のヒントを役立ててください。

警告を検索する

［再生ログ］の中で警告などを検索します。

応答を確認する

Web_reg_find を使用して，直前のステップの応答が正しいことを確認してください。詳細については，『**Online Function Reference**』（英語版）（［**ヘルプ**］＞［**関数リファレンス**］）を参照してください。

代替ナビゲーションを使用する

問題が生じているステップや Java アプレットを使用しているステップでは，「**代替ナビゲーション**」を使用して Web（Click and Script）ステップを HTTP レベルのステップで置き換えます。HTTP レベルのステップでは，手作業による相関が必要なる場合があります。代替ナビゲーションを行うには，ツリー・ビューでステップを選択するか，スクリプト・ビューでテキストを選択し，右クリック・メニューから［**代替ナビゲーションに置き換え**］を選択します。

Kerberos プロトコルを使った作業

認証に Kerberos プロトコルを使用している場合，認証セッションを正しく行うために VuGen をカスタマイズする必要があります。上級ユーザであれば，自分でカスタマイズを実行できます。

Kerberos プロトコルを正しく動作させるには，krb5.ini ファイルを作成し，それを使用可能なディレクトリに格納します。krb5.ini の完全パス名を KRB5_CONFIG 環境変数に保存します。

krb5.ini ファイルには，各ドメインに関する詳細情報（KDS および AS アドレス）とトラスト・チェーンが含まれている必要があります。

詳細については，HP ソフトウェア・サポートに問い合わせてください。

Web（Click and Script）Vuser スクリプトの拡張

次の項では，スクリプトの作成に役立ついくつかの拡張機能について説明します。

以降に示す機能の大部分は API 関数に対する拡張機能です。関数とその引数の詳細については，『**Online Function Reference**』（英語版）（［**ヘルプ**］＞［**関数リファレンス**］）を参照するか，任意の関数で F1 キーを押します。

条件ステップの追加

web_xxxx という形式の名前の Web（Click and Script）関数群では，再生時の条件アクションを指定できます。たとえば，要素を調べ，要素が見つかったときにのみアクションを実行する必要がある場合などに条件が役立ちます。

たとえば，インターネット検索を実行し，［次へ］をクリックしてすべての結果ページに移動したいとします。結果ページが何ページになるかわからないので，次のページがあることを示す［次へ］ボタンがあるかどうかを，ステップを失敗させずに調べる必要があります。次のコードでは，通知を伴う検証ステップを追加しています。［次へ］ボタンが見つかった場合は，そのボタンをクリックします。

```
While (web_text_link("Next",
DESCRIPTION,
    "Text=Next",
    VERIFICATION,
    "NotFound=Notify",
    ACTION,
    "UserAction=Click",
    LAST) == LR_PASS);
```

VERIFICATION セクションの構文と使用方法の詳細については，『**Online Function Reference**』（英語版）（［**ヘルプ**］＞［**関数リファレンス**］）を参照してください。

ページ・タイトルの確認

web_browser ステップでは，タイトル検証記録オプションを使用して，正しいページがダウンロードされていることを確認できます。Vuser が，ステップごと，あるいは新しい最上位ウィンドウにナビゲーションするごとに，自動的にこのチェックを実行するようにできます。

さらに，完全一致検索および正規表現検索の両方を使用して，スクリプトの任意の位置にタイトル検証を手作業で追加することもできます。

```
web_browser("test_step",
DESCRIPTION,
...
VERIFICATION,
    "BrowserTitle=Title",
    ACTION,]
,
LAST);
```

詳細については，『**Online Function Reference**』（英語版）（［**ヘルプ**］＞［**関数リファレンス**］）を参照してください。

タイトル検証オプションは，記録オプション内で直接設定できます。詳細については，Click and Script を使った記録に関する項を参照してください。

テキスト・チェック検証

テキスト・チェックポイントを使用すれば，テキスト文字列が Web ページまたはアプリケーションの適切な場所に表示されているかどうかを検証し，その結果に基づいてアクションを実行できます。完全一致検索または正規表現検索を使用して，テキスト文字列が存在すること（**ContainsText**），あるいは，テキスト文字列が存在しないこと（**DoesNotContainText**）を確認できます。

たとえば，Web ページに「Flight departing from New York to San Francisco」という文が表示されるとします。「New York」という単語が「Flight departing from」と「to San Francisco」の間に表示されることを検査するテキスト・チェックポイントを作成できます（この例では，正規表現条件を使用する必要があります）。

これらのチェックポイントを実装するには，ステップの VERIFICATION セクションにテキスト・チェック関連の引数を追加します。Vuser は，再生時に，ブラウザの HTML ドキュメントと子フレームの innerText を検索します。**NotFound** 引数は，オブジェクトが見つからなかったため，あるいはテキスト検証が失敗したために検証が失敗した場合に行うアクションを指定します（Error，Warning，または Notify）。

テキスト検証は，スクリプトの既存ステップに手作業で追加できます。テキスト検証は，要素を生成したステップの後に置くようにします。

テキスト検証の引数は次の Action 関数で有効です：**web_browser**，**web_element**，**web_list**，**web_text_link**，**web_table**，**web_text_area**。

注：同じタイプのテキスト検証は，ステップごとに 1 回だけ使用できます（たとえば，**ContainsText** を 2 回使用することはできません）。複数のテキストについて検証する必要がある場合は，検証を複数のステップに分けます。ただし，同じステップの中でも異なる検証であれば同時に使用できます（たとえば，**ContainsText** と **DoesNotContainText**）。この場合，ステップが成功するためには，すべての条件が満たされる必要があります。

次の例では，www.acme.com からフランス版の Web サイト acme.com/fr に誘導されなかったかどうかを検証引数によって確認しています。

```
web_browser("www.acme.com",
        ACTION,
        "Navigate=http://www.acme.com/",
    LAST);

web_browser("Verify",
        VERIFICATION,
        "ContainsText=Go to Acme France",
        "DoesNotContainText=acme.com in English",
    LAST);
```

パラメータへの JavaScript 値の保存

EvalJavaScript 引数は，Web ページの JavaScript を評価すること可能にします。

たとえば，ページ・タイトルと同じ名前のリンクをクリックしたいとします。次の例では，ドキュメントのタイトルを評価し，そのタイトルを次の web_text_link 関数で使用しています。

```
web_browser("GetTitle",
    ACTION,
    "EvalJavaScript=document.title;",
    "EvalJavaScriptResultParam=title",
    LAST);

web_text_link("Link",
    DESCRIPTION,
    "Text={title}",
    LAST);
```

ユーザ定義の記述を使った作業

何らかのグループに属するリンクを無作為にクリックしたいとします。たとえば，**hp.com** で国を無作為に選択したいとしましょう。この種の操作は，通常の記述の照合ではできません。しかし，ユーザ定義記述引数を使用すれば，グループのすべてのリンクに共通の属性を使用してグループを特定できます。

ユーザ定義記述引数を使用して，対象要素に対して事前定義されていない属性も含め，要素の属性を指定します。Vuser は，再生時に，DESCRIPTION セクションに指定されている属性を検索します。再生は，DESCRIPTION セクションの未知の引数について失敗することはありません。

たとえば，次のハイパーリンクを探したいとします。

Yahoo。この場合次のコードを使用します。

```
web_text_link("yahoo",
    DESCRIPTION,
    "Text=yahoo",
    "my_attribute=bar",
    LAST);
```

次の例では，関係するすべてのリンクが newmerc-left-ct という同じクラス名を持っているため，次のコードによって無作為にリンクをクリックできます。

```
web_text_link("Click",
            DESCRIPTION,
            "Class=newmerc-left-ct",
            "Ordinal=random",
            LAST);
```

次の関数ではユーザ定義記述引数はサポートされません：**web_browser**，**web_map_area**，**web_radio_group**，**web_reg_dialog**。

第 39 章

Web（HTTP/HTML，Click and Script）関数

VuGen を使用して，Web サイトでのユーザ・アクションが記述された Web Vuser スクリプトを作成できます。各スクリプトには，実行された各アクションに直接対応する関数が含まれます。

本章の内容

以降の情報は，Web（Click and Script），Web（HTTP/HTML），Oracle Web Applications 11i，および PeopleSoft Enterprise を対象とします。

Web Vuser 関数について

ブラウザと Web サーバの間のインターネット通信をエミュレートするために開発された関数を，Web Vuser 関数といいます。各 Web Vuser 関数の名前には，**web** というプレフィックスが付きます。関数の中には，スクリプトの記録時に生成されるものと，手作業でスクリプトに挿入しなければならないものがあります。

Web 関数は次のように分類されます。

- アクション関数
- 認証関数
- チェック関数
- 接続定義関数
- 同時実行グループ関数
- クッキー関数
- 相関関数
- フィルタ関数
- ヘッダ関数
- プロキシ・サーバ関数
- 再生関数
- その他の関数

Web（Click and Script）Vuser は，ユーザ・アクションをエミュレートするのに前述以外の関数を使用します。

アクション関数でない関数のほとんどは，Web（Click and Script）Vuser スクリプトの中で使用できます。ただし，**web_concurrent_start** 関数および **web_concurrent_end** 関数は，Web（HTTP/HTML）Vuser スクリプトに固有のものです。

Web プロトコル関数の詳細な情報や例については，『**Online Function Reference**』（英語版）（［**ヘルプ**］＞［**関数リファレンス**］）を参照してください。

一般 Vuser 関数をスクリプトに追加する方法の詳細については，『**第 1 巻 － VuGen の使用**』の「Vuser スクリプトの拡張」を参照してください。

関数の追加と編集

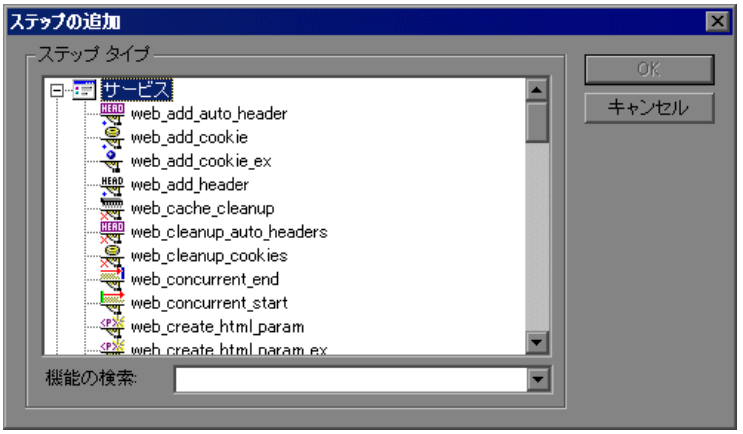
多くの Web Vuser 関数は記録セッション中に作成されます。また，ツリー・ビューとスクリプト・ビューの両方において，記録後に Web Vuser 関数を手作業で追加して編集できます。

スクリプトに追加する新しいステップを選択すると，VuGen によって，次のタイプにステップが分類されます。

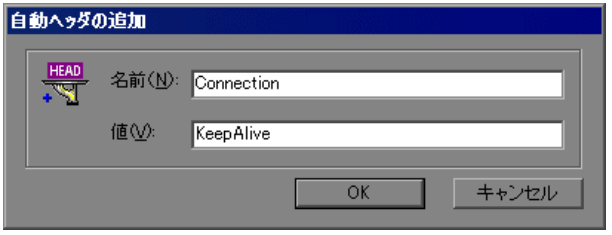
アイコンの種類	説明
サービス	**サービス**・ステップは，Web アプリケーション・コンテキストをまったく変更しないステップを表します。サービス・ステップはコンテキストを変更するのではなく，各種プロキシの設定，認証情報の送信，ユーザ定義のヘッダの発行などのカスタマイズ作業を実行します。
URL	［**URL**］ステップは，ブラウザに URL を入力したりブックマークを使用したりして，特定の Web ページにアクセスすると生成されます。各［URL］アイコンは，Vuser スクリプト内の **web_url** 関数を表します。ターゲット・ページの URL の最後の部分が，［URL］アイコンの標準ラベルとなります。
リンク	記録中にハイパーテキスト・リンクをクリックすると，［**リンク**］ステップが追加されます。各［リンク］ステップは，Vuser スクリプトの **web_link** 関数を表します。ハイパーテキスト・リンクのテキスト文字列が，このステップの標準のラベルとなります。
画像	記録中にハイパーグラフィック・リンクをクリックすると，［**画像**］ステップが Vuser スクリプトに追加されます。各［画像］ステップは，Vuser スクリプトの **web_image** 関数を表します。HTML コードの画像に ALT 属性がある場合，アイコンの標準のラベルとしてこの属性の値が使用されます。HTML コードの画像に ALT 属性がない場合は，アイコンのラベルとして SRC 属性の最後の部分が使用されます。
フォームを送信 / データを送信	記録中にフォームを送信すると，［**フォームを送信**］ステップまたは［**データを送信**］ステップが追加されます。フォームの処理に使用される実行プログラムの名前が，ステップの標準のラベルとなります。
ユーザ定義要求	VuGen で標準的なアクション（URL，リンク，画像，フォームの送信など）として認識されないアクションを記録する場合は，［**カスタム要求**］ステップが Vuser スクリプトに追加されます。非標準 HTTP アプリケーションに適用されます。

Vuser スクリプトに新しい関数を追加するには，次の手順を実行します。

1 ［**挿入**］＞［**新規ステップ**］を選択します。［ステップの追加］ダイアログ・ボックスが開きます。



2 使用する関数を選択して，［**OK**］をクリックします。ほとんどの Web Vuser 関数は，［**サービス**］カテゴリの下にあります。選んだ関数の［プロパティ］ダイアログ・ボックスが開きます。このダイアログ・ボックスでは，関数の引数を指定できます。



3 プロパティを指定して［**OK**］をクリックします。関数が引数と一緒にカーソルの位置に挿入されます。

［プロパティ］ダイアログ・ボックスを開いて，引数の値を変更することによって既存のステップを編集できます。これは，ツリー・ビューをサポートするプロトコルの場合にのみ有効です（WAP には使用できません）。

既存のステップを編集するには，次の手順を実行します。

1 ツリー・ビューで，右クリック・メニューから［**プロパティ**］を選択します。選んだ関数の［プロパティ］ダイアログ・ボックスが開きます。

2 必要に応じて引数の値を変更し，［**OK**］をクリックします。

トランザクション，ランデブー，コメント，ログ関数などの一般的な Vuser 関数は，記録中に手作業で追加できます。詳細については，『**第 1 巻 － VuGen の使用**』の「Vuser スクリプトの拡張」を参照してください。

全般的な Web（Click and Script）API に関する注意事項

本項では Web（Click and Script）関数の全般的な注意事項について説明します。等号の前に「/RE」を付けてテキスト文字列の先頭に置くことで，ほとんどのオブジェクト記述に対して正規表現を指定できます。詳細については，オンライン関数リファレンス（［ヘルプ］ ＞［関数リファレンス］）を参照してください。次に例を示します。

```
web_text_link("Manage Assets",
        DESCRIPTION,
        "Text/RE=(Manage Assets)|(Configure Assets)",
        ACTION,
        "UserAction=Click",
        LAST);
```

序数

Ordinal 属性は，同じ記述を持つオブジェクトが複数登場する場合に，それぞれを識別するために使用される，1 から始まるインデックス番号です。次の例では，記録されている 2 つの **web_text_link** 関数の引数がまったく同じです。序数だけが異なります。序数値の 2 は，2 番目の出現であることを示しています。

```
web_text_link("Manage Assets",
        DESCRIPTION,
        "Text=Manage Assets",
        "FrameName=main",
        ACTION,
        "UserAction=Click",
        LAST);

web_text_link("Manage Assets_2",
        DESCRIPTION,
        "Text=Manage Assets",
        "Ordinal=2",
        "FrameName=main",
        ACTION,
        "UserAction=Click",
        LAST);
```

空の文字列

引数を指定しないことと，引数を空の文字列として指定することとは，別の意味になります。引数を指定しないときは，標準設定値が使用されるか，または引数が無視されます。引数を列挙したものの，値として空の文字列を割り当てたときは，空の文字列に一致するものか，文字列がまったくないものとして一致するものが検索されます。たとえば，id 引数を省略すると，HTML 要素の id プロパティを無視するよう VuGen に指示することになります。"ID=" と指定した場合は，id プロパティを持たない HTML 要素か，空の ID を持つ HTML 要素が検索されます。

```
web_text_link("Manage Assets_2",
        DESCRIPTION,
        "Text=Manage Assets",
        "Id=",
        "FrameName=main",
        ACTION,
        "UserAction=Click",
        LAST);
```

キャッシュ・データの使用

ブラウザのキャッシュにデータを保存しておいて，スクリプト内の後の時点で読み込むことができます。

この機能をスクリプト内に実装する場合は，**web_dump_cache** および **web_load_cache** 関数を手動で追加します。

キャッシュへの情報のダンプ

キャッシュにデータを転送することを，情報のダンプと呼びます。**web_dump_cache** 関数を実行して，**FileName** 引数で指定した場所にキャッシュ・ファイルを生成します。キャッシュ・ファイルを生成するためには，この関数を 1 回のみ実行する必要があります。

次の例では，web_dump_cache 関数は，スクリプトを実行する VuserName パラメータごとに C:¥temp にキャッシュ・ファイルを生成します。

```
web_dump_cache("paycheckcache","FileName=c:¥¥temp¥¥{Vuser
        Name}paycheck", "Replace=yes", LAST)
```

1 個の Vuser を 10 回実行すると，VuGen は次の形式で 10 個のキャッシュ・ファイルを生成します。プレフィックスは VuserName の値になります。

```
Ku001paycheck.cache
Ku002paycheck.cache
Ku003paycheck.cache
...
```

1 番目と 2 番目の引数（この例では paycheckcache と paycheck）を変更して，現在のトランザクション名を反映させることができます。すべてのリソースをロードした後，スクリプトの最後にこの関数を置きます。

キャッシュからの情報のロード

web_load_cache 関数は，**FileName** 引数で指定された場所にあるキャッシュ・ファイルをロードします。**web_load_cache** 関数にはキャッシュ・ファイルが必須です。したがって，この関数は **web_dump_cache** 関数を実行した後にのみ実行できます。

web_load_cache 関数が **C:¥temp** から **paycheck** キャッシュ・ファイルをロードする例を次に示します。

```
web_load_cache("ActionLoad","FileName=c:¥¥temp¥¥{VuserName}paycheck",LAST)
```

キャッシュ関数の挿入

次の手順では，キャッシュ関数を使用する方法について説明します。

キャッシュ関数を使用するには，次の手順を実行します。

1. **web_dump_cache** 関数をスクリプトに挿入します。
2. スクリプトを最低 1 回実行します。
3. **web_load_cache** 関数をスクリプトの Vuser アクションの前に挿入します。
4. **web_dump_cache** 関数をコメントにします。
5. スクリプトを実行し，保存します。

キャッシングの例

給与明細を参照している PeopleSoft Enterprise Vuser の例を次に示します。

```
Actions()
{
//   web_add_cookie("storedCookieCheck=true; domain=pbntas05; path=/");

web_load_cache("ActionLoad","FileName=c:¥¥temp¥¥{VuserName}paycheck",LAST);

    web_browser("signon.html",
        DESCRIPTION,
        ACTION,
        "Navigate=http://pbntas05:8200/ps/signon.html",
        LAST);
    lr_think_time(35);

    web_edit_field("userid",
        "Snapshot=t1.inf",
        DESCRIPTION, "Type=text",
        "Name=userid",
        ACTION,
        "SetValue={VuserName}",
        LAST);
```

```
    web_edit_field("pwd",
        "Snapshot=t2.inf",
        DESCRIPTION,
        "Type=password",
        "Name=pwd",
        ACTION,
        "SetValue=HCRUSA_KU0007",
        LAST);

    lr_start_transaction("login");
        web_button("Sign In",
            "Snapshot=t3.inf",
            DESCRIPTION,
            "Type=submit",
            "Tag=INPUT",
            "Value=Sign In",
            LAST);
    lr_end_transaction("login", LR_AUTO);

    web_image_link("CO_EMPLOYEE_SELF_SERVICE",
        "Snapshot=t4.inf",
        DESCRIPTION,
        "Alt=",
        "Name=CO_EMPLOYEE_SELF_SERVICE",
        "Ordinal=1",
        ACTION,
        "ClickCoordinate=10,10",
        LAST); ...

    web_text_link("Sign out",
        "Snapshot=t7.inf",
        DESCRIPTION,
        "Text=Sign out",
        "FrameName=UniversalHeader",
        ACTION,
        "UserAction=Click",
        LAST);

/*web_dump_cache("paycheck","FileName=c:¥¥{VuserName}paycheck",
"Replace=yes", LAST);*/
    return 0;
}
```

第 40 章

Web（HTTP/HTML，Click and Script）のテキストと画像の検証

Web Vuser スクリプトにチェックを追加できます。これを使って，Vuser スクリプトを実行したときにサーバが正しい Web ページを返すかどうかを判定します。

本章の内容

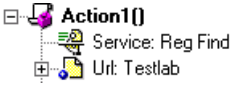
負荷下の検証について

VuGen を使って Web Vuser スクリプトにチェックを追加できます。Web チェックでは，Web ページに特定のオブジェクトがあるかどうかを検証します。オブジェクトは，テキスト文字列または画像です。

Web チェックによって，多数の Vuser がアクセスしているときに Web サイトが正しく機能するか，つまりサーバが正しい Web ページを返すかどうかを確認できます。これが特に重要なのは，サイトに多数のユーザの負荷がかかるときです。大きな負荷がかかった状態では，サーバが間違ったページを返す可能性が高くなるからです。

たとえば，世界の主要都市の気温に関する情報を表示する Web サイトがあるとします。VuGen を使って，その Web サイトにアクセスする Vuser スクリプトを作成します。

Vuser はサイトにアクセスし，この Web ページのテキストをチェックします。たとえば，ページ内に「**Temperature**」という単語が表示されれば，チェックは合格です。サーバから正しいページが返されなかったために，「**Temperature**」という単語が表示されなければ，チェックは不合格になります。テキスト・チェックのステップは URL ステップの前に出現します。これは，VuGen によって次のステップに必要な検索情報が「登録」される，つまり，事前に準備されるためです。Vuser スクリプトを実行すると，以降の Web ページを対象とするチェックが実行されます。



スクリプトを記録したときや，単独の Vuser でスクリプトを実行したときには，サーバから正しいページを返されていたとしても，多数の Vuser でサーバに負荷をかけた場合には，正しいページが返されないことがあります。サーバが過負荷になり，そのために無意味な，あるいは不正な HTML コードが返されることがあります。また，過負荷になったサーバが「**500 Server Error**」ページを返すこともあります。どちらの場合も，サーバから正しいページが返されたかどうかを判定するチェックを挿入できます。

注：Web チェックによって Vuser の処理が増えるので，Load Generator ごとの Vuser 数を減らす必要が生じることがあります。Web チェックは，実際にサーバから不正確なページが返されたことがある場合に限って使うことをお勧めします。

Web チェックは，Vuser スクリプトの記録中または記録後に定義できます。

VuGen では，それぞれが異なるチェックの種類を示す数種類の Web チェック・アイコンを使用しています。

Web チェック・アイコン	説明
テキスト 	テキスト・チェック。次のアクション関数ステップ内で（**web_reg_find**），またはビジネス・プロセス・ステップ全体の中で（**web_global_verification**），特定の文字列を検索します。
テキスト 	テキスト・チェック。ダウンロードした HTML ページ内で，**web_find** ステップを使用して特定の文字列を検索します。詳細については，603 ページ「テキスト・チェック関数について」を参照してください。
画像 	画像チェック。Web ページ内で，特定の画像を検索します。詳細については，603 ページ「テキスト・チェック関数について」を参照してください。

本章では，VuGen を使ってツリー・ビューに Web チェックを追加する方法を説明します。テキスト・ベースのスクリプト・ビューでのスクリプトへのチェックの追加については，『**Online Function Reference**』（英語版）（［**ヘルプ**］＞［**関数リファレンス**］）を参照してください。

テキスト・チェックの追加

VuGen では，Web ページのテキスト文字列を検索するチェックを追加できます。テキスト・チェックは記録中または記録後に追加できます。

テキスト・チェックを作成するときに，チェックの名前，チェックの範囲，チェックするテキスト，および検索条件を定義します。

記録中にテキスト・チェックを追加するには，次の手順を実行します。

1 アプリケーションまたは Web ブラウザのウィンドウで，対象となるテキストを選択します。

2 記録ツールバーの［**テキスト チェックを挿入**］ボタンをクリックします。VuGen によって **web_reg_find** 関数がスクリプトに追加されます。

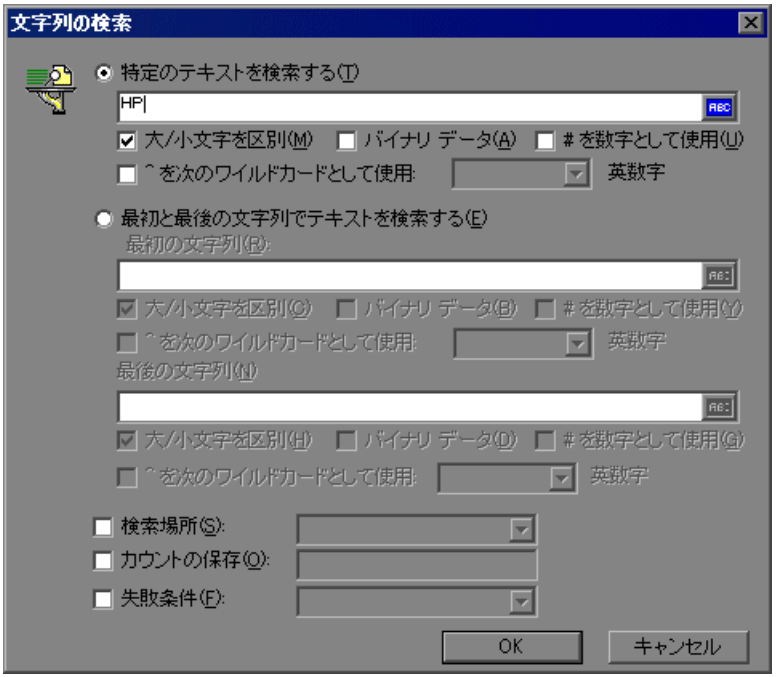
記録後のテキスト・チェックを追加するには，次の手順を実行します。

1 テキスト・チェックの対象となるステップのスナップショットに移動します。

2 スナップショットの中で，検証対象テキストを選択します。

3 右クリックして表示されるポップアップ・メニューから［**テキスト チェックを追加 (web_reg_find)**］を選択します。［文字列の検索］ダイアログ・ボックスが開きます。

注：プロトコルによっては，スナップショットからではなく［サーバの応答］タブからテキスト・チェックを追加しなければならないというメッセージが発行されます。［**サーバの応答**］タブをクリックし，［**HTML ドキュメント**］タブを選択します。本文のノードを拡張し，次に示すとおり続行します。



web_reg_find では次の属性を使用できます。

- ［**特定のテキストを検索する**］：検索するテキスト文字列です。この属性は空でない，NULL 終端文字列である必要があります。
- ［**最初と最後の文字列でテキストを検索する**］
  - ［**最初の文字列**］：検索するテキスト文字列のプレフィックスです。
  - ［**最後の文字列**］：検索するテキスト文字列のサフィックスです。

検索方法（特定のテキストあるいは開始または終了文字列）を指定したら，検索オプションを指定できます。

- 大文字と小文字を区別して検索するには，［**大 / 小文字を区別**］を選択します。バイナリ・データを指定するには，［**バイナリ データ**］を選択します。任意の数字が一致するように指定するには，テキスト文字列にハッシュ（#）を使用して［**# を数字として使用**］を選択します。
- ［**^ を次のワイルドカードとして使用**］：英数字（すべての文字，大文字，または小文字）に対してワイルドカードを使った検索を行えます。ワイルドカードは，**^** を使って指定します。

- ［**検索場所**］：テキストを検索する対象です。指定可能な値は，**Headers**，**Body**，または **All** です。標準設定は **Body** です。
- ［**カウントの保存**］：検出された一致の数です。この属性を使用するには，［**カウントの保存**］を選択して一致数を保存するパラメータ名を指定します。変数は NULL で終了する ASCII 値になります。
- ［**失敗条件**］：文字列が検出されない場合の処理メソッドです。指定可能な値は，**Found** および **Not Found** です。**Found** は，テキストが検出されたときにエラーが発生することを示します（「Error」など）。**Not Found** は，テキストが検出されないときにエラーが発生することを示します。

テキスト・チェックのプロパティを作成後に表示または変更するには，［**ツリー ビュー**］タブをクリックして「**Services: Reg Find**」ステップをダブルクリックします。［テキストの検索］ダイアログ・ボックスでは，すべてのステップの属性を表示または変更できます。

テキスト・チェック関数について

テキスト・チェックを追加すると，VuGen によって **web_reg_find** 関数がスクリプトに追加されます。この関数によって，HTML ページ上でのテキスト文字列の検索が「登録」されます。登録とは，直ちに検索を実行しないで，**web_url** など，次の Action 関数の実行後にだけチェックを実行することを意味します。同時実行関数グループで作業する場合，**web_reg_find** 関数はグループ化の後にだけ実行されます。

次の例では，**web_reg_find** 関数がテキスト文字列「Welcome」を検索します。文字列が検出されない場合は，次のアクション関数が失敗し，スクリプトの実行が停止します。

```
web_reg_find("Text=Welcome", "Fail=Found", LAST);
web_url("Step", "URL=...", LAST);
```

web_reg_find 関数以外に，ほかの関数を使用して HTML ページ内のテキストを検索できます。

ほかにもいくつかの関数を使用してテキストを検索できます。

- web_find
- web_global_verification

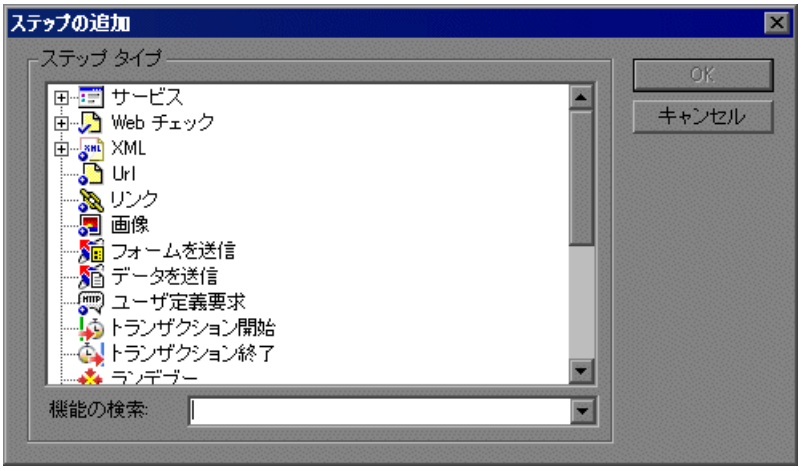
web_find 関数は下位互換性を保つためにだけ提供されているもので，HTML ベースのスクリプトに限定されている点が **web_reg_find** 関数とは異なります（［**記録オプション**］＞［**記録**］タブを参照）。この関数にはインスタンスなどの追加の属性もあり，テキストの出現回数を決めることができます。標準のテキスト検索を実行する場合には，**web_reg_find** 関数のほうが便利です。

web_global_verification 関数によって，ビジネス・プロセス全体のデータを検索できます。次のアクション関数のみに適用される **web_reg_find** とは対照的に，この関数は **web_url** など，後続の**すべての**アクション関数に適用されます。標準設定では，検索範囲は「NORESOURCE」で，ヘッダとリソースを除いた HTML 本体のみを検索します。

web_global_verification 関数は，HTTP ステータス・コードに含まれないアプリケーション・レベルのエラーの検出に最適です。この関数は HTML ベースのスクリプトだけに制限されません。詳細については，［**記録オプション**］＞［**記録**］タブを参照してください。

Vuser スクリプトへ関数を追加するには，次の手順を実行します。

1 VuGen のメイン・ウィンドウで，テキスト・チェックを追加する位置をクリックします。［**挿入**］＞［**新規ステップ**］を選択します。



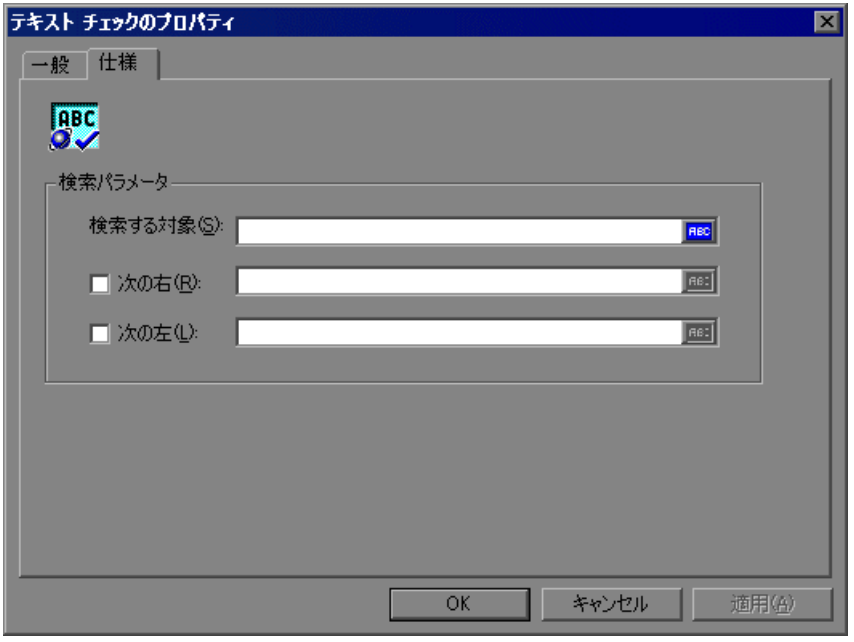
2 **web_find** 関数の場合は，［**Web チェック**］ノードを展開して［**テキスト チェック**］を選択します。**web_global_verification** 関数の場合は，［**サービス**］ノードを展開して関数名を選択します。［プロパティ］ダイアログ・ボックスが開きます。

3 これらの関数のプロパティを設定します（以降の説明を参照）。

4 ［**OK**］をクリックします。VuGen によって新しい関数がスクリプトに挿入されます。

web_find のプロパティの設定

web_find 関数では次のプロパティを設定できます。



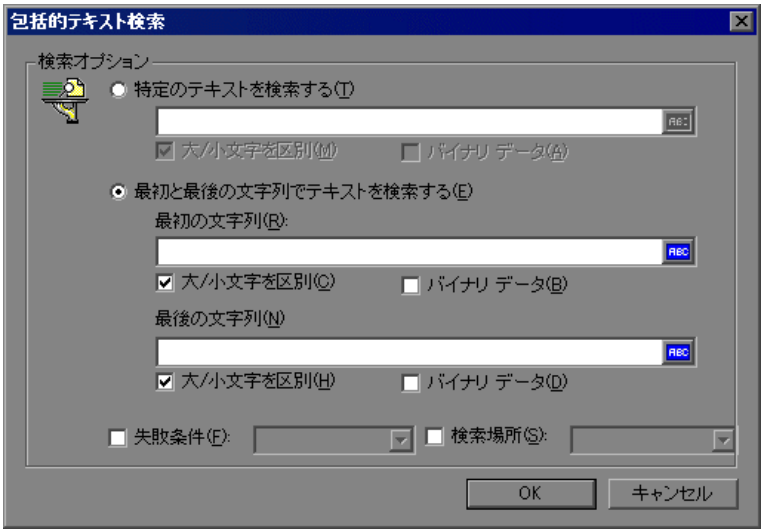
- [**検索する対象**]：確認対象文字列。[ABC] アイコンは，[**検索する対象**] ボックスの文字列がパラメータを割り当てられていないことを示します。パラメータの指定については，『**第 1 巻 ― VuGen の使用**』の「パラメータの作成」を参照してください。
- [**次の右**] または [**次の左**]：隣接するテキストを基準とする検索文字列の位置。該当するフィールドにテキストを入力します。たとえば，「support@hp.com」という文字列が「e-mail:」という単語の右に表示されることを検証する際，[**次の右**] を選択して，[**次の右**] ボックスに「e-mail:」と入力します。
- [**ステップ名**]：テキスト・チェックの名前。[**一般**] タブをクリックし，テキスト・チェックの名前をボックスに入力します。後で識別しやすいようにわかりやすい名前を付けます。

注：スクリプト実行時の Vuser による Web チェックの実行は，チェックが有効になっていて，スクリプトが HTML モードで実行されているときに限られます。チェックを有効にするには，［実行環境設定］ダイアログ・ボックスの［**プリファレンス**］タブで［**画像とテキスト チェックを有効にする**］オプションを選択します。詳細については，『**第 1 巻 ー VuGen の使用**』の「実行環境の設定」を参照してください。

---

web_global_verification のプロパティの設定

web_find 関数では次のプロパティを設定できます。



- ［**特定のテキストを検索する**］：存在の有無を確認する対象となる文字列。［ABC］アイコンは，［**検索する対象**］ボックスの文字列がパラメータを割り当てられていないことを示します。パラメータの指定については，『**第 1 巻 ー VuGen の使用**』の「パラメータの作成」を参照してください。
- ［**最初と最後の文字列でテキストを検索する**］：境界。テキストを囲む「**開始**」および「**終了**」文字列とも呼ばれます。［**大 / 小文字を区別**］するのか，または［**バイナリ データ**］を検索するのかに応じて，該当するオプションを選択して指定します。

➤［**失敗条件**］：条件を満たす場合にスクリプトを失敗させます。失敗条件として，テキストが［**Found**］なのか（見つかったのか），または［**Not found**］なのか（見つからなかったのか）を指定することもできます。必要な動作を［**失敗条件**］ボックスで選択します。

テキスト・フラグ

登録された検索 **web_reg_find** を使用して検索テキストを指定するときは，フラグを追加して検索範囲を制御できます。

/IC は，大文字小文字を区別しないよう指定します。

/BIN は，バイナリ・データを指定します。

/DIG は，シャープ記号（#）を 1 桁の数値を表すワイルドカードとして解釈するよう指定します。DIG フラグではリテラルのシャープ記号は一致しません。

/ALNUM ＜大文字小文字＞は，キャレット記号（^）を 1 文字の US-ASCII 英数字を表すワイルドカードとして解釈するよう指定します。これには 3 つの構文があります。**ALNUMIC** は，大文字小文字を区別しません。**ALNUMLC** は，小文字にのみ一致します。**ALNUMUC** は，大文字にのみ一致します。ALNUM フラグではリテラルのキャレットは一致しません。

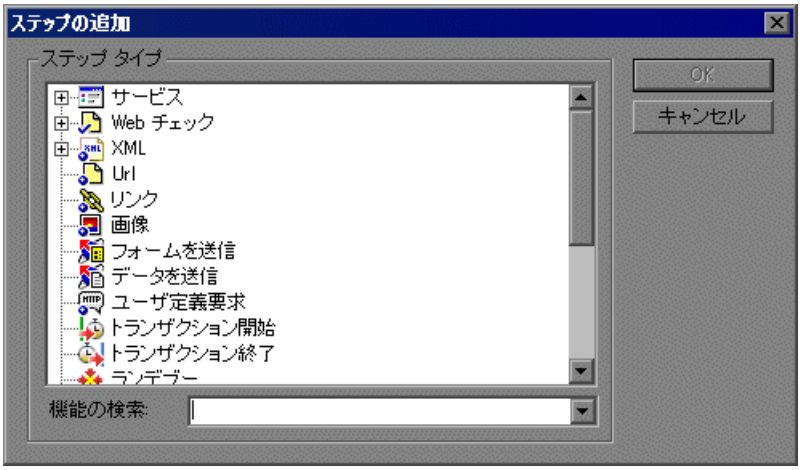
フラグを使用するには，属性 **TEXT** を入力し，その直後にスラッシュとフラグ名を入力します。たとえば，大文字小文字を区別せずに文字列を検索するには，"Text/IC= **検索対象テキスト** " と入力します。

画像チェックの追加

VuGen を使って Web ページ内の画像を検索するユーザ定義チェックを追加できます。画像は ALT 属性と SRC 属性のどちらかまたは両方によって識別できます。

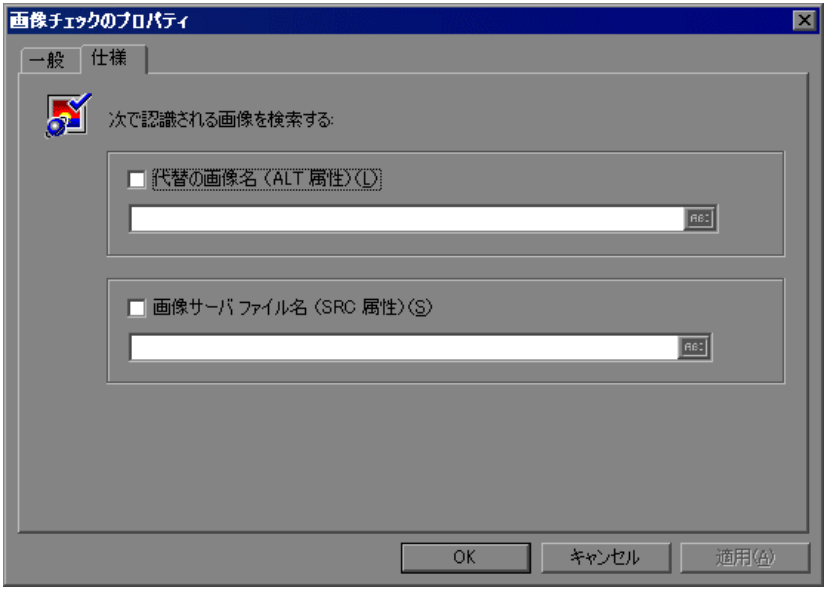
画像チェックを追加するには，次の手順を実行します。

1 VuGen のメイン・ウィンドウで，チェックを実行する Web ページに対応するステップを右クリックします。ポップアップ・メニューから［**後に挿入**］を選択します。［ステップの追加］ダイアログ・ボックスが開きます。



2［**ステップ タイプ**］ツリーで［**Web チェック**］を展開します。

3［**画像チェック**］を選択して［**OK**］をクリックします。［画像チェックのプロパティ］ダイアログ・ボックスが表示されます。［**仕様**］タブが表示されていることを確認します。



4 画像を識別するメソッドを選択します。

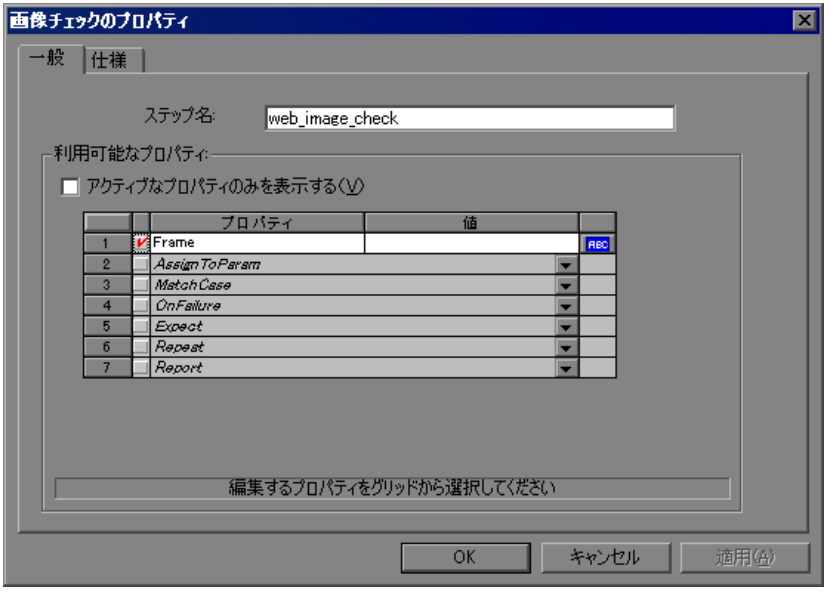
- ［**代替の画像名 (ALT 属性)**］: **ALT** 属性を使って画像を識別します。テキスト・ボックスに ALT 属性のテキスト値を入力します。スクリプトを実行すると，Vuser は指定された ALT 属性を持つ画像を検索します。
- ［**画像サーバ ファイル名 (SRC 属性)**］: SRC 属性を使って画像を識別します。テキスト・ボックスに SRC 属性のテキスト値を入力します。スクリプトを実行すると，Vuser は指定された SRC 属性を持つ画像を検索します。

［ABC］アイコンは，ALT 属性または SRC 属性にパラメータが指定されていないことを示します。パラメータの指定については，『**第 1 巻 ― VuGen の使用**』の「パラメータの作成」を参照してください。

注 :［ALT 属性］チェック・ボックスと［SRC 属性］チェック・ボックスを両方選択した場合，Vuser は指定された ALT 属性と指定された SRC 属性の両方を持つ画像を検索します。

---

5 画像チェックに名前を付けるには，［**一般**］タブをクリックします。［**ステップ名**］ボックスに，画像チェックの名前を入力します。後で識別しやすいようにわかりやすい名前を付けます。

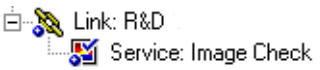


6 プロパティ・テーブルにチェックを定義する追加プロパティが表示されます。

［**アクティブなプロパティのみを表示する**］チェック・ボックスをクリアし，アクティブなプロパティと非アクティブなプロパティの両方を表示します。プロパティを有効にするには，プロパティ名の左のセルをクリックします。［**値**］カラムでプロパティの値を指定します。

プロパティ値の指定の詳細については，611 ページ「追加プロパティの定義」を参照してください。

7 ［**OK**］をクリックして設定を適用します。新しい［画像チェック］アイコンが，Vuser スクリプト内の関連ステップに追加されます。



追加プロパティの定義

Vuser スクリプトに挿入する各 Web チェックの追加のプロパティを指定できます。チェック・プロパティ・ダイアログ・ボックスの［**一般**］タブにあるプロパティ・テーブルで追加オプションを設定します。以降の説明は **web_find** および **web_image_check** 関数のみが対象であり，**web_reg_find** 関数は対象外です。

追加プロパティを設定するには，次の手順を実行します。

1 プロパティを編集する Web チェックを右クリックして，ポップアップ・メニューから［**プロパティ**］を選択します。該当するチェックのプロパティのダイアログ・ボックスが表示されます。［**一般**］タブが表示されていることを確認します。

2［**アクティブなプロパティのみを表示する**］チェック・ボックスをクリアして，利用可能なすべてのプロパティを表示します。

3 プロパティを有効にするには，プロパティ名の左のセルをクリックします。プロパティの横に赤いチェック・マークが表示されます。

4［**値**］カラムでプロパティの値を指定します。

- ［**Frame**］：検査オブジェクトがあるフレーム名を入力します。
- ［**AssignToParm**］：［**YES**］を選択すると，パラメータへの代入が有効になります。［**NO**］を選択するとこの機能は無効になります。標準設定の値は［**NO**］です。
- ［**MatchCase**］：［**YES**］を選択すると，大文字と小文字を区別して検索します。［**NO**］を選択すると，大文字と小文字を区別しないで検索します。標準設定の値は［**NO**］です。
- ［**OnFailure**］：［**Abort**］を選択すると，検査が失敗した場合に Vuser スクリプトをすべて中止します。VuGen は，実行環境の設定で指定されているエラー処理方法にかかわらず，Vuser スクリプトを中止します。［**続行**］を選択すると，チェックに失敗したスクリプトを中止するかどうかは，実行環境の設定で定義されているエラー処理方法に従います。

  標準設定の値は［**続行**］です。エラー処理方法の定義の詳細については，『**第 1 巻 － VuGen の使用**』の「実行環境の設定」を参照してください。
- ［**Expect**］：［**NotFound**］を選択すると，Vuser が指定した検査オブジェクトを見つけられない場合に，検査に合格したことになります。［**Found**］を選択すると，Vuser が指定したチェック対象オブジェクトを見つけた場合に，チェックに合格したことになります。標準設定の値は［Found］です。

- ［**Repeat**］:［**YES**］を選択すると，指定したチェック対象オブジェクトの複数の出現を検索します。［**NO**］を選択すると，指定したチェック対象オブジェクトが 1 つ見つかった時点で直ちにチェックが終了します。Vuser スクリプトは次のステップから実行を続けます。このオプションは，検査オブジェクトが複数含まれる可能性のある Web ページを検索するときに役立ちます。標準設定の値は［YES］です。
- ［**Report**］:［**Always**］を選択すると，実行ログでチェック結果の詳細を常に表示できます。［**Failure**］を選択すると，チェックに不合格だった場合にのみチェック結果の詳細を表示します。［**Success**］を選択すると，チェックに合格した場合にのみチェック結果の詳細を表示します。標準設定の値は［**Always**］です。

［ABC］アイコンは，プロパティの値がパラメータを割り当てられていないことを示します。アイコンをクリックしてパラメータを割り当てます。詳細については，『**第 1 巻 ー VuGen の使用**』の「パラメータの作成」を参照してください。

第 41 章

Web とワイヤレス Vuser スクリプトの変更

Web またはワイヤレス Vuser スクリプトの記録が終わったら，VuGen を使って，記録したスクリプトを変更できます。新規ステップの追加のほか，既存のステップの編集または削除ができます。

本章の内容

Web とワイヤレス Vuser スクリプトの変更

セッションの記録後，ステップのプロパティを編集したり，ステップを追加，削除したりして，記録したスクリプトを VuGen で変更できます。

変更は，アイコン・ベースのツリー・ビューまたはテキスト・ベースのスクリプト・ビューのどちらでも行えます。2 つの表示モードの詳細については，第 37 章「Web（HTTP/HTML，Click and Script）プロトコル」を参照してください。

本章では，VuGen を使用してツリー・ビューでスクリプトを変更する方法について説明します。テキスト・ベースのスクリプト・ビューでのスクリプトの変更については，『**Online Function Reference**』（英語版）（［**ヘルプ**］＞［**関数リファレンス**］）を参照してください。

バイナリ・データの追加

バイナリ・コード・データを HTTP 要求の本体に含めるには，次の形式に従います。

¥x[char1][char2]

これは，[char1][char2] によって表される 16 進値です。

たとえば，¥x24 は 10 進数で表せば 16*2+4=36 となり，対応する文字は $ 記号です。同様に，¥x2B は + 記号です。

16 進法のシーケンスとして 1 桁のシーケンスは使用しないでください。たとえば，¥x2 は有効なシーケンスではありませんが，¥x02 は有効なシーケンスです。

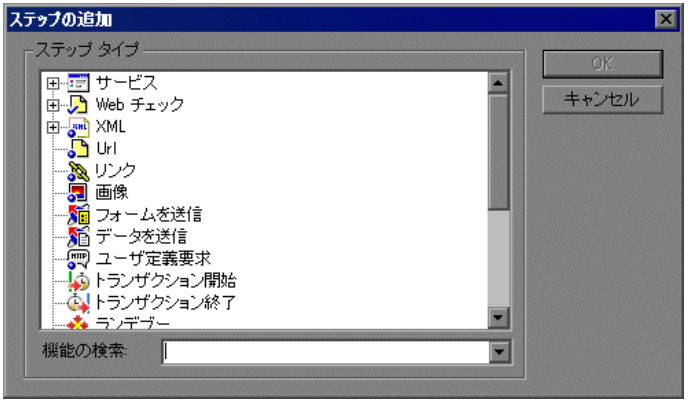
Vuser スクリプトへのステップの追加

記録セッション中に VuGen が記録するステップに加え，記録されたスクリプトにステップを追加することもできます。

Vuser スクリプトにステップを追加するには，次の手順を実行します。

1 スクリプトのツリー・ビューで，新しいステップを追加する位置の前または後のステップを選択します。

2 [**挿入**] > [**新規ステップ**] を選んで，選択したステップの後にステップを挿入するか，右クリックして表示されるメニューから [**前に挿入**] または [**後に挿入**] を選択します。[ステップの追加] ダイアログ・ボックスが開きます。



3 [**ステップ タイプ**] ツリーまたは [**関数の検索**] リストから，追加するステップの種類を選択します。

4 [**OK**] をクリックします。追加するステップに関する情報を入力するダイアログ・ボックスが開きます。このダイアログ・ボックスは，追加するステップの種類に応じて異なります。

このダイアログ・ボックスの使い方の詳細については，以下に示す各項を参照してください。

追加項目	参照先
Vuser API 関数	『**第 1 巻 ― VuGen の使用**』の「Vuser スクリプトの拡張」
サービス・ステップ	637 ページ「サービス・ステップの変更」
Web チェック	638 ページ「Web チェックの変更（Web のみ）」
トランザクション	634 ページ「トランザクションの変更」
ランデブー・ポイント	635 ページ「ランデブー・ポイントの変更」
「思考遅延時間」ステップ	636 ページ「思考遅延時間の変更」
「URL」ステップ	617 ページ「「URL」ステップの変更」
「リンク」ステップ	620 ページ「「ハイパーテキスト・リンク」ステップの変更（Web のみ）」
「画像」ステップ	621 ページ「「画像」ステップの変更（Web のみ）」
「フォームを送信」ステップ	623 ページ「「フォームを送信」ステップの変更（Web のみ）」
「データを送信」ステップ	627 ページ「「データを送信」ステップの変更」
「ユーザ定義要求」ステップ	631 ページ「「ユーザ定義要求」ステップの変更」
「ユーザ定義」ステップ	『**第 1 巻 ― VuGen の使用**』の「Vuser スクリプトの拡張」

Vuser スクリプトからのステップの削除

セッションの記録後に，VuGen を使って Vuser スクリプトからステップを削除できます。

Vuser スクリプトからステップを削除するには，次の手順を実行します。

1 Vuser スクリプトのツリー・ビューで，削除するステップを右クリックして，ポップアップ・メニューから［**削除**］を選択します。

2［**OK**］をクリックして，このステップの削除を確認します。

ステップがスクリプトから削除されます。

アクション・ステップの変更

アクション・ステップは，記録中のユーザ・アクションを表します。つまり，新しい URL へのジャンプまたは Web コンテキストの変更などを表します。

Vuser スクリプトのツリー・ビューにアクション・アイコンとして表示されているアクション・ステップは，記録中に自動的にスクリプトに追加されます。記録後，記録されたアクション・ステップを変更できます。

本項の内容

- 「URL」ステップの変更
- 「ハイパーテキスト・リンク」ステップの変更（Web のみ）
- 「画像」ステップの変更（Web のみ）
- 「フォームを送信」ステップの変更（Web のみ）
- 「データを送信」ステップの変更
- 「ユーザ定義要求」ステップの変更

「URL」ステップの変更

URL を入力したり，特定の Web ページにアクセスするブックマークを使用したりすると，Vuser スクリプトに「URL」ステップが追加されます。

変更できるプロパティは，ステップ名，URL のアドレス，ターゲット・フレーム，記録モードです。

標準では，VuGen は記録時のモードに基づいて「URL」ステップを実行します。記録モードには，**HTML**，または **HTTP**（リソースを含まないモード）があります。記録モードの詳細については，『**第 1 巻 － VuGen の使用**』の 380 ページ「記録レベルについて」を参照してください。

再生モードの設定

Vuser によって，スクリプトを記録時のモードとは異なるモードで実行するには，［URL ステップのプロパティ］ダイアログ・ボックスでモードの設定を変更します。再生モードを変更するには，［**記録モード**］チェック・ボックスを選択します。以下に示す再生モードを使用できます。

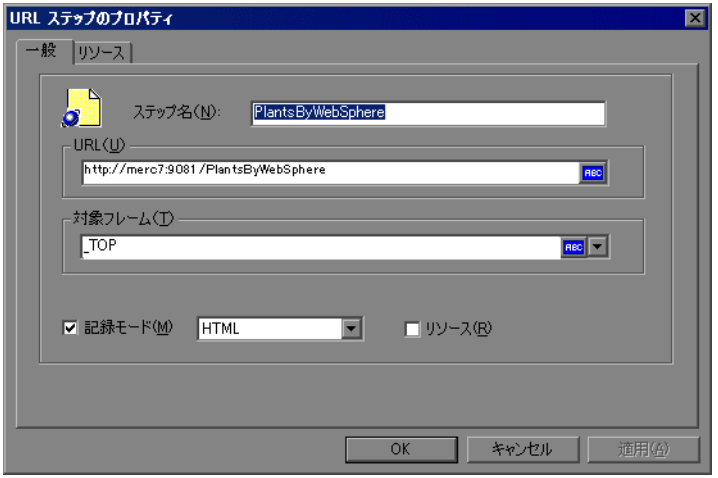
- **HTML**：自動的にすべてのリソースと画像をダウンロードし，以降のステップに必要な HTTP 情報を格納します。このモードは，Web のリンクが含まれるスクリプトに適しています。
- **HTTP**：再生時に，このステップのためにリソースをダウンロードしません。関数によって明示的に指定されたリソースだけをダウンロードします。

特定のステップをリソースとしてみなさないようにすることもできます。たとえば，省略する必要のある特定の画像を表すステップがある場合，その種類のリソースを含めないように設定できます。詳細については，『**第 1 巻 － VuGen の使用**』の第 20 章「Web Vuser の記録オプションの設定」を参照してください。

「URL」ステップのプロパティを変更するには，次の手順を実行します。

1 Vuser スクリプトのツリー・ビューで，編集する「URL」ステップを選択します。「URL」ステップは「URL」アイコンで表示されます。

2 VuGen ツールバーの［**プロパティ**］ボタンをクリックします。［URL ステップのプロパティ］ダイアログ・ボックスが開きます。



3 ステップ名を変更するには，［**ステップ名**］ボックスに新しい名前を入力します。記録中は，URL の最後の部分が標準名となります。

4 ［**URL**］ボックスに，「URL」ステップがアクセスした Web ページの Web アドレス（URL）を入力します。［**ABC**］アイコンは，この URL にパラメータが割り当てられていないことを示します。パラメータの指定については，『**第 1 巻 ー VuGen の使用**』の「パラメータの作成」を参照してください。

5 ［**対象フレーム**］リストで，次の値から 1 つを選択します。

- ［**SELF**］: 最後の（変更のあった）フレームを置き換えます。
- ［**PARENT**］: 最後の（変更のあった）フレームの親の内容を置き換えます。
- ［**TOP**］: ページ全体を置き換えます。
- ［**BLANK**］: 新規ウィンドウを開きます。

6 再生モードを変更するには，［**記録モード**］チェック・ボックスを選択します。

次の中から使用するモードを選択します：HTML または HTTP。

7 項目をリソースとしてダウンロードしないようにするには，［**リソース**］チェック・ボックスをクリアします。

8 ［**OK**］をクリックして，［URL ステップのプロパティ］ダイアログ・ボックスを閉じます。

「ハイパーテキスト・リンク」ステップの変更（Web のみ）

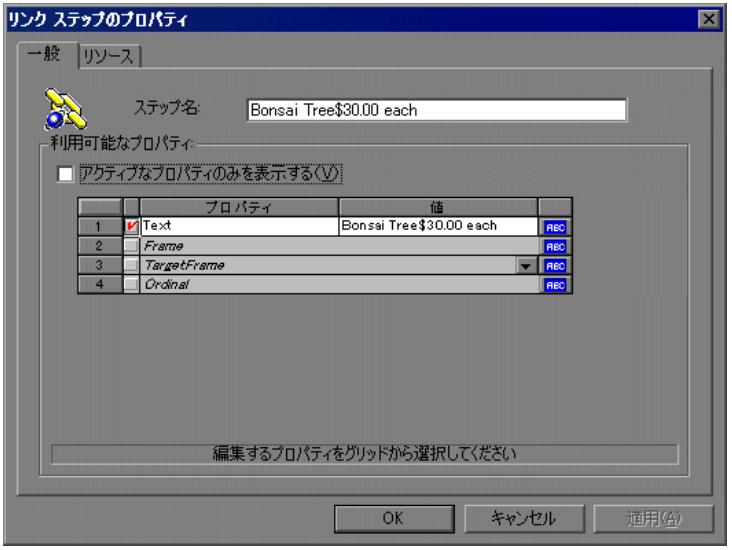
ハイパーテキスト・リンク・ステップは，ハイパーテキスト・リンクをクリックすると，Vuser スクリプトに追加されます。このステップは，**HTML ベースのスクリプト**・モードで記録するためのオプションを選択した場合にのみ記録されます。詳細については，『**第 1 巻 ― VuGen の使用**』の第 20 章「Web Vuser の記録オプションの設定」を参照してください。

変更できるプロパティは，ステップ名，ハイパーテキスト・リンクの識別方法，配置場所です。

ハイパーテキスト・リンク・ステップのプロパティを変更するには，次の手順を実行します。

1 Vuser スクリプトのツリー・ビューで，編集するハイパーテキスト・リンク・ステップを選択します。ハイパーテキスト・リンク・ステップは［**Hypertext Link**］アイコンで表示されます。

2 右クリックして表示されるメニューから［**プロパティ**］を選択します。［リンク ステップのプロパティ］ダイアログ・ボックスが開きます。



3 ステップ名を変更するには，［**ステップ名**］ボックスに新しい名前を入力します。記録中は，ハイパーテキスト・リンクのテキスト文字列が標準名となります。

4 プロパティ・テーブルに，リンクを識別するプロパティが表示されます。

［**アクティブなプロパティのみを表示する**］チェック・ボックスをクリアし，アクティブなプロパティと非アクティブなプロパティの両方を表示します。プロパティを有効にするには，プロパティ名の左のセルをクリックします。［**値**］カラムでプロパティの値を割り当てます。

- ［**Text**］：ハイパーテキスト・リンクの正確な文字列。
- ［**Frame**］：リンクが属しているフレームの名前。
- ［**TargetFrame**］：次のターゲット・フレーム。
  - ［**TOP**］：ページ全体を置き換えます。
  - ［**BLANK**］：新規ウィンドウを開きます。
  - ［**PARENT**］：最後の（変更のあった）フレームの親の内容を置き換えます。
  - ［**SELF**］：最後の（変更のあった）フレームを置き換えます。
- ［**Ordinal**］：ほかのすべてのプロパティ属性が，同じ Web ページ上にあるほかのリンク（1 つまたは複数）と同じときに，リンクを一意に識別する番号。詳細については，『**Online Function Reference**』（英語版）を参照してください。

［ABC］アイコンは，プロパティの値にパラメータが割り当てられていないことを示します。パラメータの指定については，『**第 1 巻 － VuGen の使用**』の「パラメータの作成」を参照してください。

5 ［**OK**］をクリックして，［リンク ステップのプロパティ］ダイアログ・ボックスを閉じます。

「画像」ステップの変更（Web のみ）

ハイパーグラフィック・リンクをクリックすると，「画像」ステップが Vuser スクリプトに追加されます。このステップは，HTML（コンテキスト・センシティブ）モードで記録するためのオプションを選択した場合にのみ記録されます。詳細については，『**第 1 巻 － VuGen の使用**』の第 20 章「Web Vuser の記録オプションの設定」を参照してください。

変更できるプロパティは，ステップ名，ハイパーグラフィック・リンクの識別方法，配置場所です。

「画像」ステップのプロパティを変更するには，次の手順を実行します。

1 Vuser スクリプトのツリー・ビューで，編集する「画像」ステップを選択します。「画像」ステップは［**画像**］アイコンによって示されます。

2 右クリックして表示されるメニューから［**プロパティ**］を選択します。［画像ステップのプロパティ］ダイアログ・ボックスが開きます。

3 ステップ名を変更するには，［**ステップ名**］ボックスに新しい名前を入力します。記録時の標準の名前は，画像の ALT 属性の値です。画像に ALT 属性がない場合は，SRC 属性の最後の部分が標準名として使用されます。

4 プロパティ・テーブルに，リンクを識別するプロパティが表示されます。

［**アクティブなプロパティのみを表示する**］チェック・ボックスをクリアし，アクティブなプロパティと非アクティブなプロパティの両方を表示します。プロパティを有効にするには，プロパティ名の左のセルをクリックします。［**値**］カラムでプロパティの値を割り当てます。

- ［**ALT**］：画像の ALT 属性。
- ［**SRC**］：画像の SRC 属性。
- ［**MapName**］：画像に対応するマップ名。クライアント側イメージ・マップにのみ適用されます。

- ［**AreaAlt**］：クリックする領域の ALT 属性。クライアント側イメージ・マップにのみ適用されます。
- ［**AreaOrdinal**］：クリックする領域のシリアル番号。クライアント側イメージ・マップにのみ適用されます。
- ［**Frame**］：画像が配置されているフレームの名前。
- ［**TargetFrame**］：次のターゲット・フレーム。
  - ［**_TOP**］：ページ全体を置き換えます。
  - ［**_BLANK**］：新規ウィンドウを開きます。
  - ［**_PARENT**］：最後の（変更のあった）フレームの親の内容を置き換えます。
  - ［**_SELF**］：最後の（変更のあった）フレームを置き換えます。
- ［**Ordinal**］：ほかのすべてのプロパティ属性が，同じ Web ページ上にあるほかの画像（1 つまたは複数）と同じときに，画像を一意に識別する番号。詳細については，『**Online Function Reference**』（英語版）を参照してください。
- ［**XCoord**］，［**YCoord**］：画像上でマウスをクリックした場所の座標。

［ABC］アイコンは，プロパティの値にパラメータが割り当てられていないことを示します。パラメータの指定については，『**第 1 巻 － VuGen の使用**』の「パラメータの作成」を参照してください。

5 ［**OK**］をクリックして［画像ステップのプロパティ］ダイアログ・ボックスを閉じます。

「フォームを送信」ステップの変更（Web のみ）

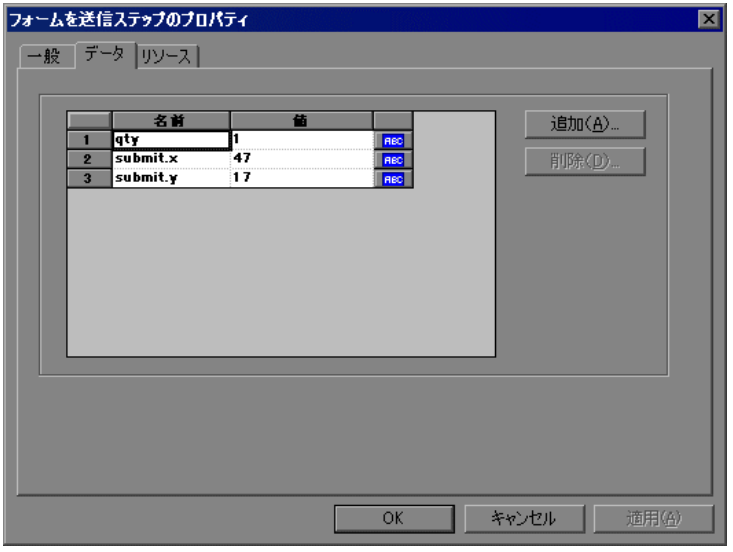
フォームを送信すると，「フォームを送信」ステップが Vuser スクリプトに追加されます。このステップは，HTML（コンテキスト・センシティブ）モードで記録するためのオプションを選択した場合にのみ記録されます。詳細については，『**第 1 巻 － VuGen の使用**』の第 20 章「Web Vuser の記録オプションの設定」を参照してください。

変更できるプロパティは，ステップ名，フォームの場所，フォーム送信の識別方法，フォーム・データ，ステップのリソースです。

「フォームを送信」ステップのプロパティを変更するには，次の手順を実行します。

1 Vuser スクリプトのツリー・ビューで，編集する「フォームを送信」ステップを選択します。「フォームを送信」ステップは，［**フォームを送信**］アイコンで示されます。

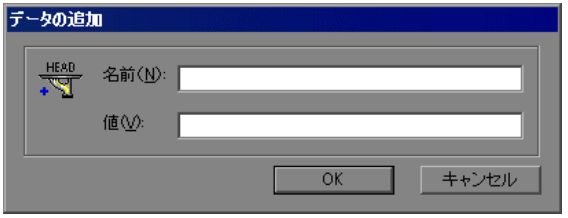
2 右クリックして表示されるメニューから［**プロパティ**］を選択します。［フォームを送信ステップのプロパティ］ダイアログ・ボックスが開きます。［**データ**］タブをクリックします。



- ［**名前**］カラムには，フォームのすべてのデータ引数が一覧表示されます。
- ［**値**］カラムには，データ引数に対応する入力値が表示されます。
- タイプを示す［値］の右のカラムには，アイコンが表示されます。最初は，すべての値が定数かパラメータ化されていない値なので，アイコンは［ABC］です。『**第 1 巻 ― VuGen の使用**』の「パラメータの作成」, で説明されているとおり，データ値にパラメータを割り当てると，［ABC］アイコンはテーブル・アイコンに変わります。

3 データ引数を編集するには，データ引数をダブル・クリックしてセル内でカーソルをアクティブにし，ボックス内に新しい値を入力します。

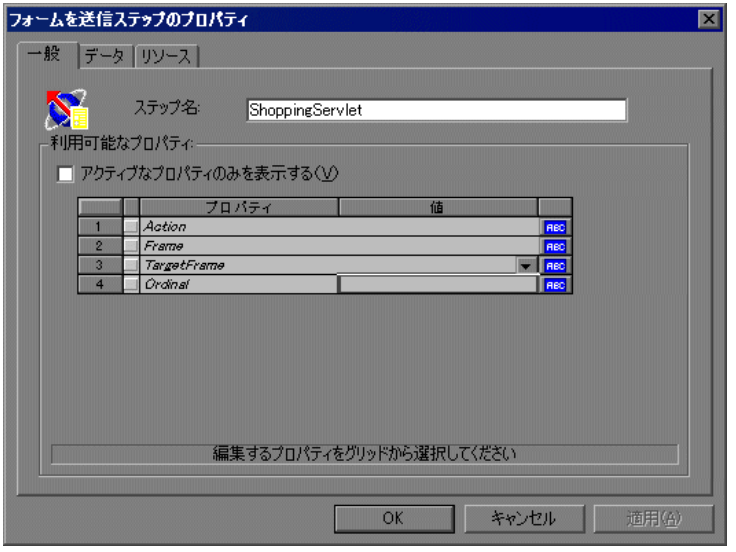
4 フォームの送信に新規データ引数を追加するには，［**追加**］をクリックします。［データの追加］ダイアログ・ボックスが開きます。



5 データ引数の［**名前**］と［**値**］を入力して，［**OK**］をクリックします。

6 引数を削除するには，引数を選択して［**削除**］をクリックします。

7 ［フォームを送信］ステップの名前を変更するには，［**一般**］タブをクリックします。



8 ステップ名を変更するには，［**ステップ名**］ボックスに新しい名前を入力します。記録時の標準の名前は，フォームの処理に使用される実行プログラム名です。

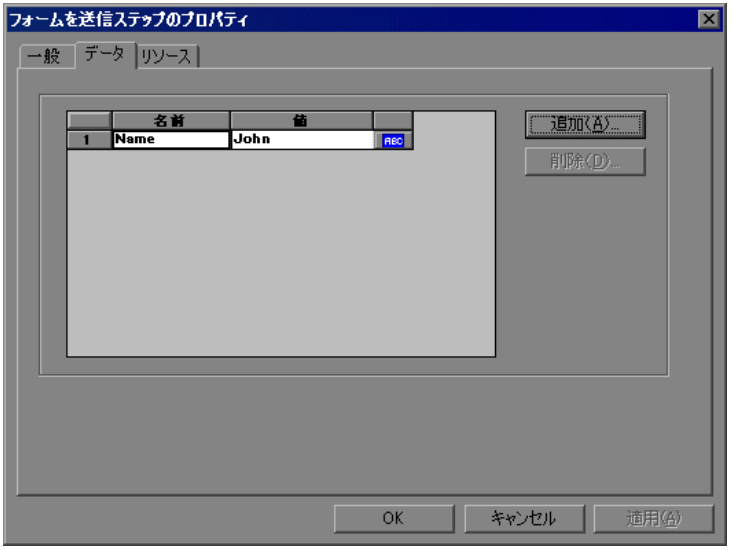
9 プロパティ・テーブルに，フォーム送信のプロパティが表示されます。

［**アクティブなプロパティのみを表示する**］オプションをクリアし，アクティブなプロパティと非アクティブなプロパティの両方を表示します。プロパティを有効にするには，プロパティ名の左のセルをクリックします。［**値**］カラムでプロパティの値を割り当てます。

- ［**Action**］：フォーム・アクションの実行に使用されるアドレス。
- ［**Frame**］：送信フォームが配置されているフレームの名前。
- ［**TargetFrame**］：次のターゲット・フレーム。
  - ［**_TOP**］：ページ全体を置き換えます。
  - ［**_BLANK**］：新規ウィンドウを開きます。
  - ［**_PARENT**］：最後の（変更のあった）フレームの親の内容を置き換えます。
  - ［**_SELF**］：最後の（変更のあった）フレームを置き換えます。
- ［**Ordinal**］：ほかのすべてのプロパティ属性が，同じ Web ページ上にあるほかのフォーム（1 つまたは複数）と同じときに，フォームを一意に識別する番号。詳細については，『**Online Function Reference**』（英語版）（［**ヘルプ**］＞［**関数リファレンス**］）を参照してください。

［ABC］アイコンは，「フォームを送信」ステップのプロパティの値にパラメータが割り当てられていないことを示します。パラメータの指定については，『**第 1 巻 ― VuGen の使用**』の「パラメータの作成」を参照してください。

10 ステップのリソースを指定するには，［**リソース**］タブをクリックします。［**追加**］をクリックして，リソースの URL と参照先ページを追加します。



11 ［**OK**］をクリックして［フォームを送信ステップのプロパティ］ダイアログ・ボックスを閉じます。

「データを送信」ステップの変更

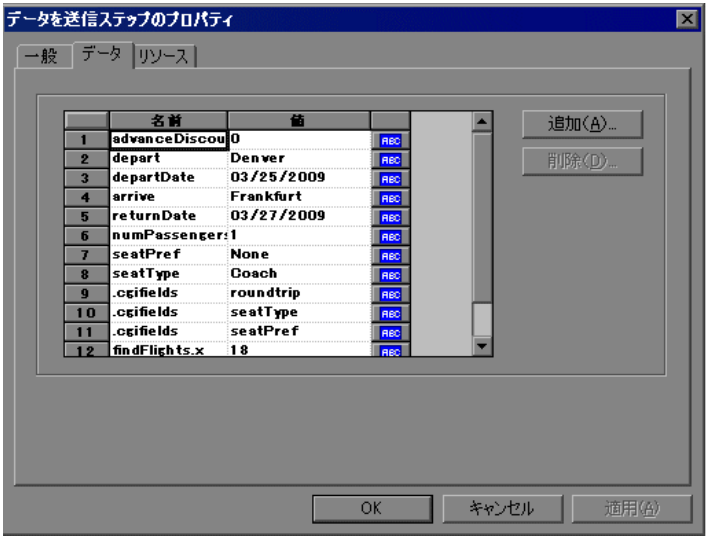
「データを送信」ステップは，Web サイトにデータを送信して処理することを表します。「フォームを送信」ステップとは異なり，この要求を実行するときにフォームのコンテキストは必要ありません。

変更できるプロパティはステップ名，メソッド，アクション，対象フレームおよびフォームのデータ項目です。

「データを送信」ステップのプロパティを変更するには，次の手順を実行します。

1 Vuser スクリプトのツリー・ビューで，編集する「データを送信」ステップを選択します。「データを送信」ステップは，［**データを送信**］アイコンによって示されます。

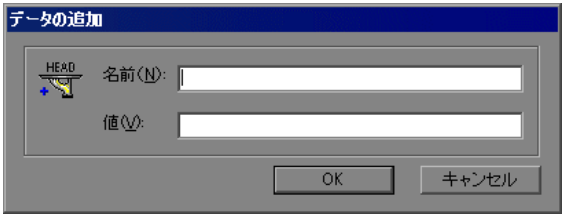
2 右クリックして表示されるメニューから［**プロパティ**］を選択します。［データを送信ステップのプロパティ］ダイアログ・ボックスが開きます。［**データ**］タブをクリックします。



- ［**名前**］カラムには，フォームのすべてのデータ引数が一覧表示されます。これにはすべての隠しフィールドが含まれます。
- ［**値**］カラムには，データ引数に対応する入力値が表示されます。
- タイプを示す［値］の右のカラムには，アイコンが表示されます。最初は，すべての値が定数かパラメータ化されていない値なので，アイコンは［ABC］です。『**第 1 巻 ― VuGen の使用**』の「パラメータの作成」, で説明されているとおり，データ値にパラメータを割り当てると，［**ABC**］アイコンはテーブル・アイコンに変わります。

3 データ引数を編集するには，データ引数をダブルクリックしてセル内でカーソルをアクティブにし，新しい値を入力します。次に，新しい値を入力します。

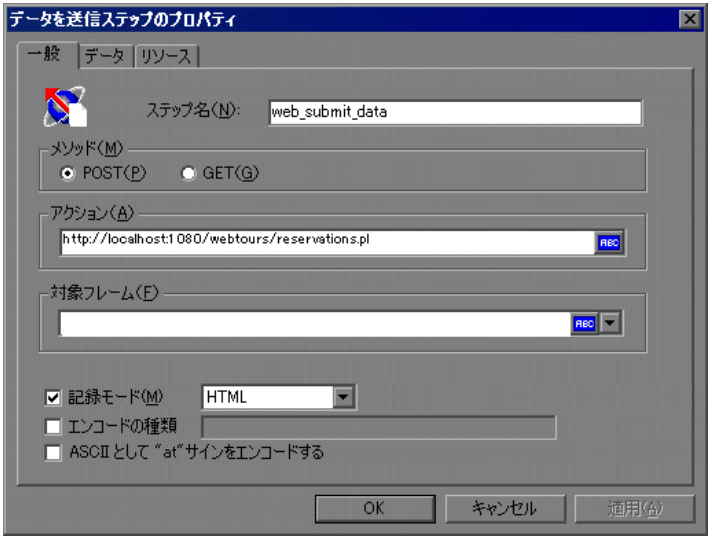
4 新規データを追加するには，［**追加**］をクリックします。［データの追加］ダイアログ・ボックスが開きます。



5 データ引数の［**名前**］と［**値**］を入力して，［**OK**］をクリックします。

6 引数を削除するには，引数を選択して［**削除**］をクリックします。

7「データを送信」ステップの名前を変更するには，［**一般**］タブをクリックします。



8 ステップ名を変更するには，［**ステップ名**］ボックスに新しい名前を入力します。

9［**メソッド**］ボックスで，［**POST**］か［**GET**］をクリックします。標準のメソッドは［**POST**］です。

10 ［**アクション**］ボックスに，データ送信時に使用するアドレスを入力します。［ABC］アイコンは，このアクションにパラメータが割り当てられていないことを示します。パラメータの指定については，『**第 1 巻 － VuGen の使用**』の「パラメータの作成」を参照してください。

11 次の［**対象フレーム**］から 1 つを選択します。

- ［**TOP**］: ページ全体を置き換えます。
- ［**BLANK**］: 新規ウィンドウを開きます。
- ［**PARENT**］: 最後の（変更のあった）フレームの親の内容を置き換えます。
- ［**SELF**］: 最後の（変更のあった）フレームを置き換えます。

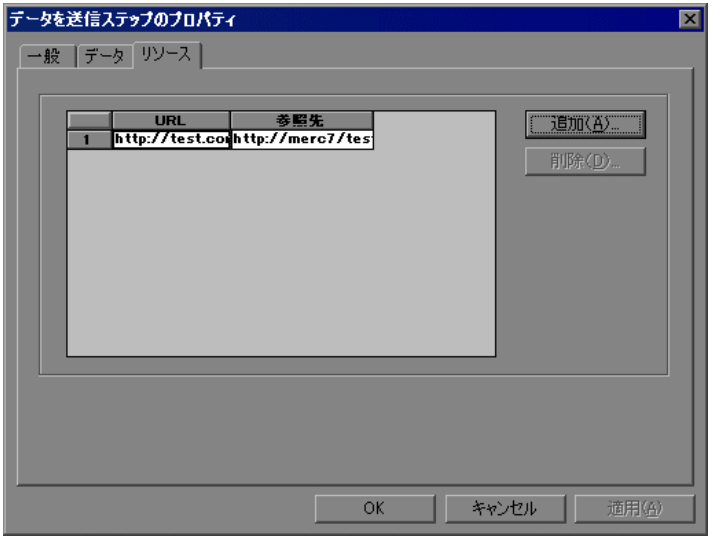
12 再生モードをカスタマイズするには，［**記録モード**］オプションを選択します。次の中から使用するモードを選択します : HTML または HTTP。詳細については，618 ページ「再生モードの設定」を参照してください。

13 **multipart/www-urlencoded** など，エンコーディング・タイプを指定するには，［**エンコードの種類**］チェック・ボックスを選択して，エンコーディング方式を指定します。

14 URL 中の「@」をエンコードするには，［**ASCII として "at" サインをエンコードする**］を選択します。

15 ［**OK**］をクリックして，［データを送信ステップのプロパティ］ダイアログ・ボックスを閉じます。

16 ステップのリソースを指定するには，［**リソース**］タブをクリックします。［**追加**］をクリックして，リソースの URL と参照先ページを追加します。



「ユーザ定義要求」ステップの変更

「ユーザ定義要求」とは，HTTP がサポートするいずれかの方法を使用した，URL に対するユーザ定義の HTTP 要求です。「ユーザ定義要求」ステップには，コンテキストはありません。

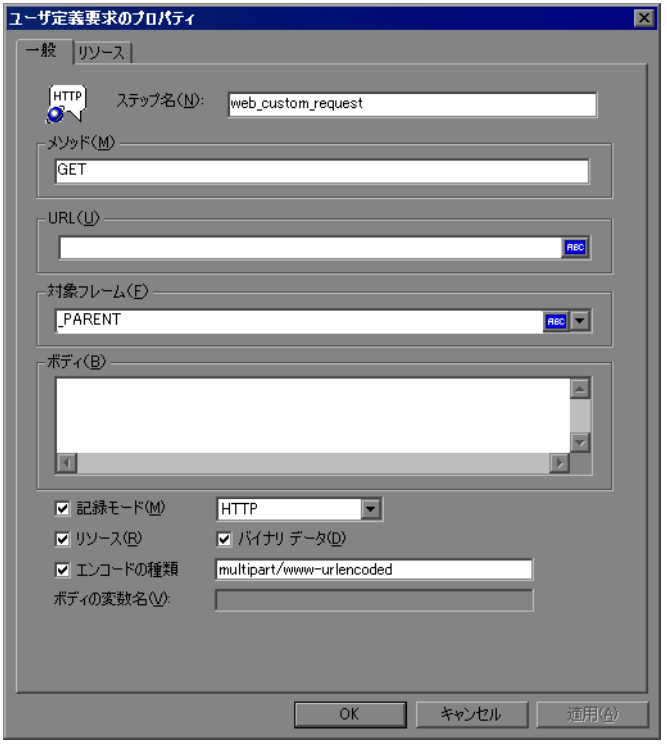
変更できるプロパティは，ステップ名，メソッド，URL，対象フレームおよび本体です。

VuGen には，ユーザ定義要求の本体を C 形式に変換する機能があります。たとえば，XML ツリーまたは大量のデータをユーザ定義要求の本体領域にコピーすることで，現在の関数で使用できるように，文字列を C 形式に変換できます。これにより，必要なエスケープ・シーケンス文字が挿入され，文字列の改行が削除されます。

［ユーザ定義要求］ステップのプロパティを変更するには，次の手順を実行します。

1 Vuser スクリプトのツリー・ビューで，編集する「ユーザ定義要求」ステップを選択します。「ユーザ定義要求」ステップは，［**ユーザ定義要求**］アイコンで表示されます。

2 右クリックして表示されるメニューから［**プロパティ**］を選択します。［ユーザ定義要求のプロパティ］ダイアログ・ボックスが開きます。



3 ステップ名を変更するには，［**ステップ名**］ボックスに新しい名前を入力します。記録中は，URL の最後の部分が標準名となります。

4 ［**メソッド**］ボックスに，HTTP によってサポートされているメソッドを入力します。たとえば，GET，POST，HEAD です。

5 ［**URL**］ボックスに，要求する URL を入力します。

6 次の［**対象フレーム**］から 1 つを選択します。

- ［**TOP**］：ページ全体を置き換えます。
- ［**BLANK**］：新規ウィンドウを開きます。
- ［**PARENT**］：最後の（変更のあった）フレームの親の内容を置き換えます。
- ［**SELF**］：最後の（変更のあった）フレームを置き換えます。

7 ［**ボディ**］属性ボックスに，要求の本体を入力するかテキストを貼り付けます。［**バイナリ データ**］チェック・ボックスを選択すると，テキストは ASCII ではなく 2 進数として処理されます。バイナリ・データの使い方の詳細については，『**Online Function Reference**』（英語版）（［**ヘルプ**］＞［**関数リファレンス**］）を参照してください。

VuGen によって，長さが 100K を超える Body 属性は，［Body variable name］ボックス内の変数に置き換えられます。この変数は，**include** フォルダにある **lrw_custom_body.h** ファイルで定義します。

8 ［**ボディ**］ボックスに貼り付けた文字列には，テキストを選択し，右クリックして表示されるメニューから［**C フォーマットに変換**］を選択します。

9 再生モードをカスタマイズするには，［**記録モード**］オプションを選択します。次の中から使用するモードを選択します：HTML または HTTP。詳細については，618 ページ「再生モードの設定」を参照してください。

10 項目をリソースとしてダウンロードしないようにするには，［**リソース**］オプションをクリアします。

11 **multipart/www-urlencoded** など，エンコーディング・タイプを指定するには，［**エンコードの種類**］を選択して，エンコーディング方式を指定します。

12 ［**OK**］をクリックして，［ユーザ定義要求のプロパティ］ダイアログ・ボックスを閉じます。

制御ステップの変更

制御ステップは，負荷テスト中に使用するアクション・ステップ以外のステップを表します。制御ステップには，トランザクション，ランデブー・ポイント，思考遅延時間があります。

記録中または記録後に，Vuser スクリプトのツリー・ビューに制御アイコンで表示される制御ステップをスクリプトに追加します。

本項の内容

- トランザクションの変更
- ランデブー・ポイントの変更
- 思考遅延時間の変更

トランザクションの変更

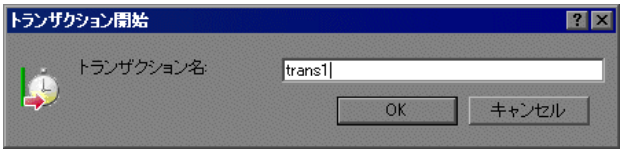
トランザクションとは，サーバの応答時間を測定するタスクや一連のアクションです。

変更できるプロパティは，トランザクション名（[トランザクション開始］と［トランザクション終了]），とそのステータス（[トランザクション終了］のみ）です。

「トランザクション開始」制御ステップを変更するには，次の手順を実行します。

1 Vuser スクリプトのツリー・ビューで，編集する「トランザクション開始」制御ステップを選択します。「トランザクション開始」制御ステップは，［**トランザクション開始**］アイコンによって示されます。

2 右クリックして表示されるメニューから［**プロパティ**］を選択します。［トランザクション開始］ダイアログ・ボックスが開きます。



3 トランザクション名を変更するには，［**トランザクション名**］ボックスに新しい名前を入力して［**OK**］をクリックします。

「トランザクション終了」制御ステップを変更するには，次の手順を実行します。

1 Vuser スクリプトのツリー・ビューで，編集する「トランザクション終了」制御ステップを選択します。「トランザクション終了」制御ステップは，［**トランザクション終了**］アイコンによって示されます。

2 右クリックして表示されるメニューから［**プロパティ**］を選択します。［トランザクション終了］ダイアログ・ボックスが開きます。

3 終了するトランザクションの名前を，［**トランザクション名**］リストから選択します。

4 トランザクションのステータスを［**トランザクション ステータス**］リストから選択します。

- ［**LR_PASS**］:「成功」のリターン・コードを返します。
- ［**LR_FAIL**］:「失敗」のリターン・コードを返します。
- ［**LR_STOP**］:「停止」のリターン・コードを返します。
- ［**LR_AUTO**］: 検出されたステータスを自動的に返します。

詳細については，『**Online Function Reference**』（英語版）（［**ヘルプ**］ > ［**関数リファレンス**］）を参照してください。

5 ［**OK**］をクリックして，［トランザクション終了］ダイアログ・ボックスを閉じます。

ランデブー・ポイントの変更

ランデブー・ポイントを使用して，複数の Vuser が同時にタスクを実行するように同期させることができます。

変更できるプロパティは，ランデブー・ポイントの名前です。

ランデブー・ポイントを変更するには，次の手順を実行します。

1 Vuser スクリプトのツリー・ビューで，編集するランデブー・ポイントを選択します。ランデブー・ポイントは，［**ランデブー**］アイコンによって表されます。

2 右クリックして表示されるメニューから［**プロパティ**］を選択します。［ランデブー］ダイアログ・ボックスが開きます。

3 ランデブーの名前を変更するには，［**ランデブー名**］ボックスに新しい名前を入力し，［**OK**］をクリックします。

思考遅延時間の変更

思考遅延時間は，アクション間で実際のユーザが待機する時間をエミュレートします。記録中，アクション間の時間があらかじめ定義したしきい値を超えると，VuGen によってそれぞれのユーザ・アクションの後に，思考遅延時間が自動的に Vuser スクリプトに追加されます。

変更できるプロパティは思考遅延時間（秒単位）です。

思考遅延時間を変更するには，次の手順を実行します。

1 Vuser スクリプトのツリー・ビューで，編集する「思考遅延時間」ステップを選択します。「思考遅延時間」ステップは，［**思考遅延時間**］アイコンによって示されます。

2 右クリックして表示されるメニューから［**プロパティ**］を選択します。［思考遅延時間］ダイアログ・ボックスが開きます。

3 ［**思考遅延時間**］ボックスに思考遅延時間を入力し，［**OK**］をクリックします。

注：Web Vuser スクリプトを実行するときは，記録されたとおりに思考遅延時間を再生するか，記録された思考遅延時間を無視するかを，Vuser に対して設定できます。詳細については，『**第 1 巻 ー VuGen の使用**』の「実行環境の設定」を参照してください。

サービス・ステップの変更

「サービス」ステップは，プロキシの設定，認証情報の送信，カスタム・ヘッダの発行などの，カスタマイズのためのタスクを実行する関数です。「サービス」ステップは，Web サイトのコンテキストを一切変更しません。

記録中または記録後に「サービス」ステップをスクリプトに追加できます。

「サービス」ステップのプロパティを変更するには，次の手順を実行します。

1 Vuser スクリプトのツリー・ビューで，編集する「サービス」ステップを選択します。「サービス」ステップはサービス・アイコンによって示されます。

2 右クリックして表示されるメニューから［**プロパティ**］を選択します。適切なサービス・ステップ・プロパティのダイアログ・ボックスが開きます。このダイアログ・ボックスは，変更するサービス・ステップの種類により異なります。選択したサービス・ステップについての説明が，ダイアログ・ボックスのタイトル・バーに表示されます。

注：一部のサービス・ステップ関数には，引数がありません。この場合，プロパティのメニュー項目は無効です。

3 サービス・ステップに必要な引数を入力または選択します。各関数の詳細については，『**Online Function Reference**』（英語版）（［**ヘルプ**］＞［**関数リファレンス**］）を参照してください。

4 ［**OK**］をクリックして，サービス・ステップのプロパティ・ダイアログ・ボックスを閉じます。

Web チェックの変更（Web のみ）

Web チェックは Web ページで特定のオブジェクトの存在を確認する機能です。オブジェクトは，テキスト文字列または画像です。

記録中または記録後に Web チェックをスクリプトに追加できます。

Web チェックのプロパティを変更するには，次の手順を実行します。

1 Vuser スクリプトのツリー・ビューで，編集する Web チェックを選択します。Web チェックは Web チェック・アイコンで表示されます。

アイコン	説明
	画像チェック・アイコン
	テキスト・チェック・アイコン

2 右クリックして表示されるメニューから［**プロパティ**］を選択します。該当する Web チェックのプロパティ・ダイアログ・ボックスが開きます。このダイアログ・ボックスは，変更するチェックの種類により異なります。

3 チェックに必要なプロパティを入力または選択します。詳細については，第 40 章「Web（HTTP/HTML，Click and Script）のテキストと画像の検証」を参照してください。

4 ［**OK**］をクリックして，チェックのプロパティ・ダイアログ・ボックスを閉じます。

第 42 章

Web（HTTP/HTML）の相関ルール

VuGen の相関機能を使うと，1 つのステートメントの結果を別のステートメントの入力項目として使用して，Vuser 関数どうしを結び付けることができます。

本章の内容

相関ステートメントについて

HTML ページには，動的データが含まれていることがよくあります。動的データとは，ユーザがサイトにアクセスするたびに変わるデータです。たとえば，Web サーバによっては，現在の日時を含むリンクを使用するものもあります。

Web Vuser スクリプトを記録すると，動的データがスクリプトに記録されることがあります。そのスクリプトによって，記録された値が Web サーバに送信されても，その値は有効なものではなくなっています。これらの変数は Web サーバによって拒否され，エラーが発行されます。このようなエラーは必ずしもはっきりわかるわけではなく，Vuser ログ・ファイルを注意深く調べないと検出できないことがあります。

Vuser の実行中にエラーが発生した場合は，スクリプト内でエラーが発生した場所を調べます。多くの場合，相関によって，1 つのステートメントの結果を別のステートメントの入力項目として使用することで，問題を解決できます。

HTML ページ内の動的データは，次の形式になっていることがあります。

- 対応する Web ページにアクセスするたびに変わる URL
- フォーム・フィールド（非表示の場合あり）の値
- JavaScript のクッキー

ケース 1

「Buy me now!」というテキストのハイパーテキスト・リンクを含む Web ページがあるとします。HTTP データを含むスクリプトを記録すると，その URL は VuGen によって次のように記録されます。

```
"http://host//cgi-bin/purchase.cgi?date=170308&ID=1234"
```

日付「170308」と ID「1234」は記録中に動的に生成されるので，新たなブラウザ・セッションを実行するたびに日付と ID が再生成されます。スクリプトを実行すると，「Buy me now!」のリンクは，記録時の URL ではなく，別の URL に関連付けられています。このため，Web サーバはその URL を取得できません。

ケース 2

名前と顧客 ID をフォームに入力し，フォームを送信するとします。

このフォームを送信すると，一意のシリアル番号がユーザのデータと一緒にサーバに送られます。このシリアル番号は HTML コード内の隠しフィールドに含まれていますが，VuGen によってスクリプトに記録されます。シリアル番号はブラウザ・セッションごとに変わるので，Vuser は記録されたスクリプトを正しく再生できませんでした。

ステートメントを相関させることで，上の 2 つのケースにおける問題を解決できます。記録されたスクリプトの動的データを，1 つまたは複数のパラメータで置き換えます。スクリプトを実行すると，各パラメータに値が割り当てられます。

相関の方法について

本章では，組み込みルールまたはユーザ定義のルールを使った自動相関について説明します。ステートメントを手作業で相関する場合や，ワイヤレスの Vuser スクリプトの相関を実行する場合は，663 ページ「手作業による相関」を参照してください。

ブラウザ・セッションを記録するときには，まず HTML モードで記録してみます。このモードを使用すると，相関が必要な箇所が少なくなります。各種の記録モードの詳細については，『**第 1 巻 － VuGen の使用**』の第 20 章「Web Vuser の記録オプションの設定」を参照してください。

記録中または記録後に，スクリプト内のステートメントを相関させるように VuGen を設定できます。本章では，記録時のソリューションとして，スクリプト内のステートメントを自動的に相関させます。また，VuGen のスナップショット相関機能を使って，記録後にスクリプトを相関させることもできます。記録後の相関の詳細については，第 43 章「Web（HTTP/HTML）－ 記録後の相関」を参照してください。

VuGen の相関ルールの使用

VuGen の相関エンジンを使えば，記録セッション中に，次のいずれかのメカニズムを使用して動的なデータを自動的に相関できます。

- 組み込み相関
- ユーザ定義ルールによる相関

詳細については，645 ページ「一致条件の追加」および 646 ページ「詳細相関ルール」を参照してください。

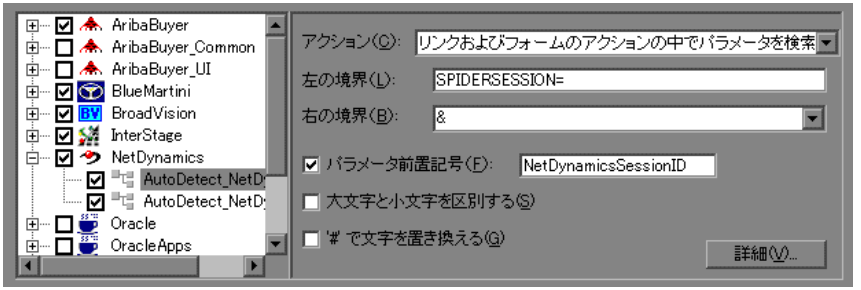
組み込み相関

組み込み相関では，サポートされているアプリケーション・サーバを対象に動的データを検出し，相関させることができます。ほとんどのサーバには，リンクおよび参照の作成時に必ず使用される，明確な構文ルール，つまり「コンテキスト」があります。

たとえば，BroadVision サーバで作成されるセッション ID は，次に示すように，必ず特定の区切り文字の間に挟まれています。左側は「BV_SessionID=」，右側は「&」です。

```
BV_SessionID=@@@@@1303778278.0969956817@@@@&
```

サポートされているアプリケーション・サーバを対象にセッションを記録するときは，VuGen に組み込まれている既存ルールの 1 つを使用できます。各アプリケーション・サーバには 1 つ以上のルールを適用できます。また，ルールの横にあるチェック・ボックスを設定またはクリアして，特定のルールを有効または無効にできます。ルールの定義は VuGen の右側の表示枠に表示されます。



サポート対象外のアプリケーション・サーバのセッションを記録するとき，コンテキストが不明で相関のルールを決められない場合には，VuGen のスナップショット比較方式を使用することができます。この方式では，記録後に相関処理の手順を導いてくれます。詳細については，第 43 章「Web（HTTP/HTML）ー 記録後の相関」を参照してください。

ユーザ定義ルールによる相関

アプリケーションに固有のルールがあり，それらを明確に定義できる場合は，［記録オプション］タブで新規ルールを定義できます。

ユーザ定義のルールによる相関を行う場合は，セッションを記録する前に，相関のルールを定義する必要があります。相関のルールは，［記録オプション］ダイアログ・ボックスで作成します。ルールには，相関させる動的データの境界などの情報や，バイナリ，大文字と小文字の一致，インスタンス番号など，一致に関する仕様が含まれます。

VuGen に対して，どの場所で条件を検索するかを指定します。

- 本体テキスト全体
- リンク / フォーム・アクション
- クッキー・ヘッダ
- フォーム・フィールド値
- クッキー挿入関数

標準設定では，ルール用に保存できる文字列の最大サイズは 4096 文字です。必要ならば，Windows インストール・ディレクトリにある **CorrelationSettings.xml** ファイルの **MaxParamLen** 属性の値を増やすことによって，この値を変更できます。

本体テキスト全体

［**本体テキスト全体の中でパラメータを検索**］オプション を選択すると，レコーダはリンク，フォーム・アクションまたはクッキーだけではなく，本体の全体を探します。テキストは，指定した境界を使って検索されます。

リンク / フォーム・アクション

［**リンクおよびフォームのアクションの中でパラメータを検索**］方式では，VuGen はリンクおよびフォームのアクションの中でパラメータ化するテキストを検索します。この方式は，コンテキストのルールがわかっているアプリケーション・サーバ向けです。左の境界，右の境界，代替の右境界，左の境界のインスタンス（左の境界の出現）を現在のリンクで定義します。

たとえば，文字列「sessionid=」の 2 回目の出現と「@」の間にあるテキストをパラメータに置き換えるとします。［**左の境界**］ボックスに，左の境界として「**sessionid=**」を指定し，［**右の境界**］ボックスに，右の境界として「**@**」を指定します。2 回目の出現を検索しているので，［**左の境界インスタンス**］ボックスで［**2**］を指定します。

右の境界の文字列が一定でない場合は，［**代替の右境界**］ボックスに，代わりに使用する右の境界を指定できます。指定した右の境界を一意に特定できないときに，この値が使用されます。

たとえば，Web ページに次の形式のリンクが含まれていたとします。

"SessionID=122@page.htm"
"Page.htm@SessionID=122&test.htm"

右の境界が一定でないので（「@」の場合と「&」の場合がある），右の境界として 1 つを指定するだけでは不十分です。この場合，代わりに使用する右の境界として「&」を指定します。

左と右の境界は，文字列を一意に特定するものでなければなりません。境界に動的データを含めることはできません。また，ドロップダウン・メニューの選択肢として用意されている［**文字列の最後**］または［**改行文字**］を，右の境界として指定することもできます。

このオプションでは，左右の境界はスクリプトに含まれている文字列内に必ず含まれている必要があります。サーバが境界を返すだけでは十分ではありません。この制限はほかのアクション・タイプにはありません。

クッキー・ヘッダ

［**クッキー ヘッダーの中でパラメータを検索**］方式は，前述のルールと似ていますが，値がリンクまたはフォームのアクションからではなく，クッキーのテキストから（記録ログに示されるとおりに）抽出される点が異なります。

フォーム・フィールド値

［**フォーム フィールド値のパラメータ化**］方式を指定すると，名前付きのフォーム・フィールド値がパラメータとして保存されます。この方式は，パラメータを作成し，それをスクリプト内のフォームのアクション・ステップの前に配置します。このオプションでは，フィールド名を指定する必要があります。

クッキー挿入関数

［**web_reg_add_cookie 関数を入力する際使用するテキスト**］方式は，バッファの中で特定の文字列を検出すると **web_reg_add_cookie** 関数を挿入します。指定したプレフィックスを持つクッキーを検出したときにのみ関数が追加されます。このオプションでは，検索するテキストとクッキーのプレフィックスを指定する必要があります。

一致条件の追加

前述のルール以外に，文字列で次の項目を指定することによって，相関検索の種類を制御することもできます。

- ［**パラメータ前置記号**］：このルールに基づいて自動的に生成されたすべてのパラメータに，プレフィックスを使用します。プレフィックスによって，既存のユーザ・パラメータへの上書きを防ぐことができます。さらに，プレフィックスによって，スクリプト内のパラメータが見分けやすくなります。たとえば，組み込みのルールの 1 つである Siebel Web は Siebel_row_id というプレフィックスを検索します。
- ［**大文字と小文字を区別する**］：境界を検索するときに大文字と小文字を区別します。
- ［**'#' で数字を置き換える**］：数値をすべてハッシュ記号（#）で置き換えます。ハッシュ記号はワイルドカードとして機能し，任意の数字を含むテキスト文字列を検索できます。たとえば，このオプションを有効にし，左の境界として「**HP###** 」を指定した場合は，「**HP193**」や「**HP284**」が有効な一致文字列として検索されます。

コメントの追加

わかりやすいコメントをスクリプト内の相関ステップに挿入するように VuGen を設定できます。このオプションを有効にするには，［**スクリプトにコメントを追加**］オプションを選択します。

詳細相関ルール

VuGen では，次の詳細相関ルールも指定できます。

- ［**常に新規のパラメータを作成する**］：パラメータに置換された値が，前のインスタンスから変わっていない場合も，このルールに応じて新しいパラメータを作成します。Web サーバがページごとに異なる値を割り当てる場合には，このオプションを設定します。たとえば，NetDynamics サーバは，不正行為を最小限に抑えるために，ページごとにセッション ID を変更する場合などがあります。
- ［**完全一致のみパラメータで置換する**］：境界と境界の間のテキストが（最初のスナップショットから）見つかった値と完全に一致した場合のみ，記録された値をパラメータで置き換えます。見つかった文字列の前または後に別の文字がある場合，パラメータの置き換えは実行されません。

  たとえば，フォーム送信の際，VuGen によって境界 aaa と bbb の間の 1234 という文字群が記録され，aaa1234bbb となったとします。以後このフォームを送信する際，VuGen は 1234 という文字列を見つけると（つまり，Name=1234 の場合），記録された値をパラメータで置き換える処理のみ実行します。ほかの値が入力された場合，その値に最初の文字列（たとえば，Name=12345）が含まれている場合でも，VuGen は，その値をパラメータで置き換えずに，12345 という値を使用します。
- ［**逆方向検索**］：文字列の最後から左境界を検索します。
- ［**左の境界インスタンス**］：一致として考えられる，文字列内（ボディではなく）の左境界の一致件数です。
- ［**オフセット**］：一致した値の部分文字列の開始位置を指定するオフセットです。部分文字列がパラメータに保存されます。標準設定は，一致した文字列の最初の部分です。必ず負数以外を指定してください。
- ［**長さ**］：パラメータに保存する，一致した文字列の部分文字列の，オフセットからの長さです。このオプションを無効にすると，標準の値が使用され，指定したオフセットから一致文字列の末尾までの文字列が保存されます。
- ［**代替の右境界**］：あらかじめ指定されている右境界が見つからなかった場合の代替条件です。テキスト，**文字列の末尾**，または**改行文字**を指定できます。

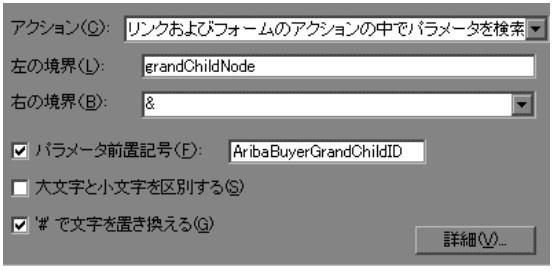
相関のルールの設定

［相関］記録オプションを使用して，ルールを追加，変更または削除できます。アプリケーション・サーバの環境を対象に自動的に作成されたルールを編集することもできます。

記録前に記録オプションを使用してルールを作成するだけでなく，記録後にルールを作成することもできます。スクリプトを実行した後，ルールを対象に相関をスキャンします（CTRL キーを押しながら F8 キーを押します）。相関結果の 1 つを選択し，そのプロパティに基づくルールを作成します。詳細については，659 ページ「相関の検索の実行」を参照してください。

相関のルールを定義するには，次の手順を実行します。

1 既存のルールをクリックするか，左側の表示枠の［**新規ルール**］をクリックします。右側の表示枠に相関ルールが表示されます。



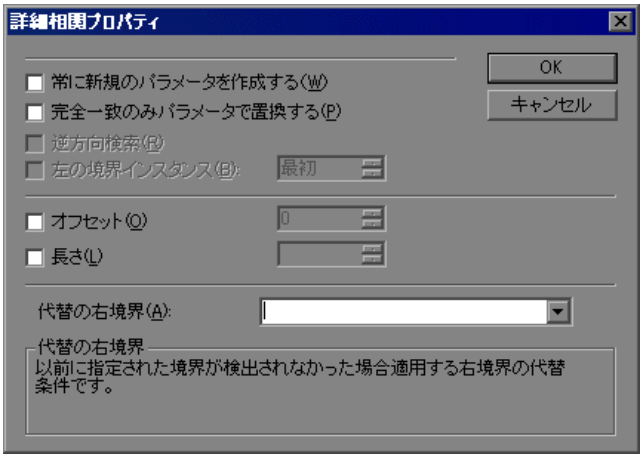
2 アクションの種類をリンクまたはフォーム・アクション，クッキー，すべてのボディ，フォーム・フィールド，または web_reg_add_cookie から選択します。

3 最初の 3 種類では，［**左の境界**］と［**右の境界**］ボックスでデータの境界を指定します。

4 フォーム・フィールド・タイプのアクションの場合は，フィールド名を指定します。

5 ［**大文字と小文字を区別する**］および［**パラメータ前置記号**］のいずれか一方，または両方のオプションを指定します。パラメータのプレフィックスを指定します。すべての数字を # 記号に変換するには，［**'#' で文字を置き換える**］を選択します。

6 詳細ルールを設定するには，［相関］ノードの［**詳細**］をクリックします。［詳細相関プロパティ］ダイアログ・ボックスが開きます。



- ［**常に新規のパラメータを作成する**］を選択すると，パラメータによって置き換えられる値が前のインスタンスから変わっていない場合も，このルールに応じて新しいパラメータを作成します。
- ［**完全一致のみパラメータで置換する**］を選択すると，テキストが検出された値に正確に一致する場合にだけ，その値をパラメータで置換します。
- ［**逆方向検索**］を選択すると，逆方向に検索します。
- ［**左の境界インスタンス**］ボックスを選択して，必要なインスタンスを指定します。
- ［**オフセット**］を選択して，一致した文字列内の文字列のオフセットを指定します。
- ［**長さ**］を選択して，一致した文字列の何文字までパラメータに保存するかを指定します。このオプションは［**オフセット**］オプションと組み合わせて使用することができます。
- ［**代替の右境界**］ボックスで別の右境界を指定するか，ドロップダウン・メニューで［**文字列の最後**］または［**改行文字**］を選択します。

7［**テスト**］をクリックして，定義したルールをテストします。詳細については，649 ページ「ルールのテスト」を参照してください。

8［**OK**］をクリックしてルールを保存し，ダイアログ・ボックスを閉じます。

ルールのテスト

本項の内容は，コンテキストが知られているサーバを対象に作成したユーザ定義のルールを対象とします。［相関］パネルで新しいルールを定義したら，セッションを記録する前に，サンプルの文字列に対してルールを適用することによってテストを実行できます。［トークン置換テスト パッド］を使用してルールをテストします。テスト・パッドを使用するには，次の手順を実行します。

1 左側の表示枠でルールを選び，［**テスト**］をクリックします。［トークン置換テスト パッド］ダイアログ・ボックスが開きます。

2［**置換用のソース テキスト**］ボックスにテキストを入力します。

3［**テスト**］をクリックします。

置換が行われた場合，パラメータ化されたソース・テキストは［**置換結果**］ボックスで確認でき，その置換に適用されたルールのリストは［**適用されたルール**］ボックスで確認できます。

第 43 章

Web（HTTP/HTML）― 記録後の相関

記録中に相関を行わなかった場合，VuGen に組み込まれている Web 相関メカニズムを使って，記録セッション後に Vuser スクリプトを相関させることができます。

本章の内容

以降の情報は，Web，ワイヤレス，SAP-Web，および Siebel Web Vuser スクリプトを対象とします。

ステートメントの相関について

VuGen には，Web Vuser スクリプトを対象にしたいくつかの相関メカニズムが用意されています。第 42 章「Web（HTTP/HTML）の相関ルール」で説明した自動による方法は，記録中に動的な値を検出するので，それらの値を直ちに相関させることができます。自動相関を無効にした場合や，自動による方法で差異のすべてが検出されなかった場合は，本章で説明する VuGen の組み込み相関メカニズムを使って差異を検出し，値を相関させることができます。部分的に相関されたスクリプトに対しても，このメカニズムを使用できます。

相関メカニズムは，スナップショットを使用してスクリプトの実行結果を追跡します。スナップショットは，Web ページを視覚的に表したものです。VuGen は，記録中および再生中に Web ページのスナップショットをキャプチャします。記録時のスナップショットと再生時のスナップショットを比較することによって，スクリプトの実行を成功させるために相関させる必要のある値を特定します。記録時および再生時のスナップショットの詳細については，『**第 1 巻ー VuGen の使用**』の第 2 章「VuGen の紹介」を参照してください。

Web 相関メカニズムには，スナップショット間のテキストまたはバイナリの差異を表示できる，比較ユーティリティが組み込まれています。比較の後，差異を 1 つずつ相関させることも，一度にすべてを相関させることもできます。

VuGen の相関メカニズムでは十分でない場合や，こうしたメカニズムをサポートしないプロトコル（ワイヤレスや手作業による相関など）に対しては，手作業による相関を使用します。詳細については，663 ページ「手作業による相関」を参照してください。

相関結果タブの表示

［相関結果］タブには，記録時のスナップショットと再生時のスナップショットの差異が表示されます。

スクリプト内を検索して相関を見つけるように指示すると，［出力］ウィンドウが開き，［**相関結果**］タブに記録時のスナップショットと再生時のスナップショットの差異が表示されます。

［**次の差異を表示**］リスト・ボックスから対象を選択することによって，スクリプト内のすべての差異を表示したり，現在のステップまたはアクションの差異だけを表示したりできます。

相関され差異が発見された項目には，［**相関済み**］カラムにチェック・マークが付きます。その右にある［**記録時の文字列**］と［**再生時の文字列**］の 2 つのカラムには，スナップショット間のテキストの差異が表示されます。その右の［**次で最初に発生**］カラムには，相関が最初に検出されたアクションが表示されます。

スナップショット間の差異を検出したら，相関を選択して［**相関**］をクリックすることにより，差異を一度に 1 つずつ相関させます。また，［**相関の削除**］ボタンを使用して特定の相関を取り消すこともできます。検出された相関のいずれかをその後の記録で発生させるには，新しい相関ルールを作成します。ルールを作成することにより，記録中に差異を認識し，自動的に相関させることができます。詳細については，654 ページ「ルールの作成」を参照してください。

このメカニズムを使って値を相関させると，VuGen は web_reg_save_param 関数と，パラメータを対象に相関が行われたことを示すコメントをスクリプトに挿入します。コメントには，元の値も示されます。

```
// [WCSPARAM WCSParam_Diff1 14 reserveFlights] Parameter {WCSParam_Diff1}
created by Correlation Studio
    web_reg_save_param( "WCSParam_Diff1", "LB= NAME=¥"", "RB=¥"", "Ord=5",
"Search=Body", "RelFrameId=1", LAST );
    web_submit_form("reservations.pl",
        "Snapshot=t4.inf",
        ITEMDATA,
        "Name=depart", "Value=Denver", ENDITEM,
        "Name=departDate", "Value=06/25/2004", ENDITEM,
        "Name=arrive", "Value=Los Angeles", ENDITEM,
        "Name=returnDate", "Value=06/26/2004", ENDITEM,
        "Name=numPassengers", "Value=1", ENDITEM,
        "Name=roundtrip", "Value=<OFF>", ENDITEM,
        "Name=seatPref", "Value=None", ENDITEM,
        "Name=seatType", "Value=Coach", ENDITEM,
        "Name=findFlights.x", "Value=44", ENDITEM,
        "Name=findFlights.y", "Value=12", ENDITEM,
        LAST);
    lr_think_time(12);
```

ルールの作成

相関結果のリストから直接ルールを作成できます。ルールを作成することにより，記録中に差異を認識し，自動的に相関させることができます。

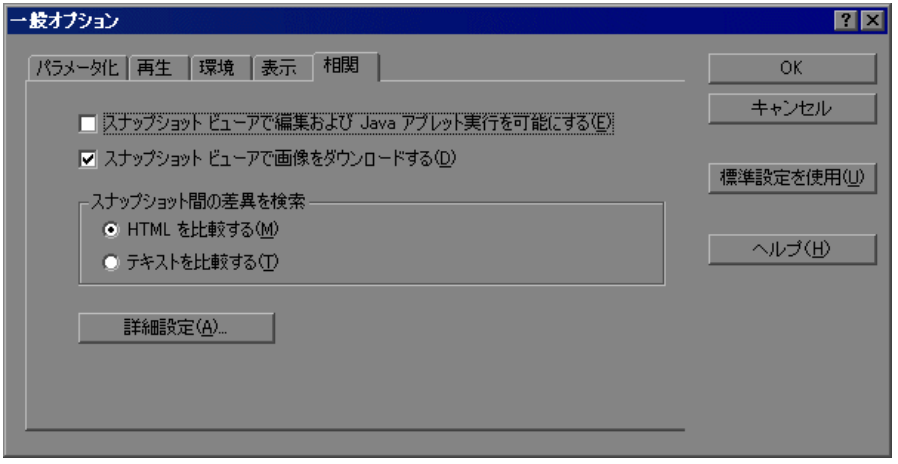
検出された相関からルールを作成するには，次の手順を実行します。

相関を選択し，［**ルールの作成**］をクリックします。相関を選択し，右クリックして表示されるメニューから［**相関ルールを作成**］を選択してルールを作成することもできます。

このルールが**相関**ルールのリストに追加されます。このルールは，［記録オプション］の［相関］ノードに表示されます。次の例では，ルールが CSRule_1 として追加されています。

VuGen の相関の設定

［一般オプション］でグローバルな相関の設定を行います。これらのオプションを設定すると，Vuser は後で使用できるように，再生中に相関情報を保存します。スナップ・ショットを比較する際に，HTML またはテキストのどちらを比較するかを指定できます。［詳細相関］ダイアログ・ボックスでは，区切り文字として扱う文字も指定できます。



- ［**スナップショット ビューアで編集および Java アプレット実行を可能にする**］：スナップショット・ウィンドウでアプレットと JavaScript を実行できるようにします。多量のリソースを使用するため，このオプションは標準設定では無効になっています。
- ［**スナップショット ビューアで画像をダウンロードする**］：画像をスナップショット・ビューアに表示するよう VuGen に指示します。ビューアでの画像の表示が遅い場合は，このオプションを無効にすることもできます。標準設定では，このオプションは有効になっています。
- ［**スナップショット間の差異を検索**］：比較の方法を選択します。
  - ［**HTML を比較する**］：HTML コードの差異のみを表示します。
  - ［**テキストを比較する**］：すべてのテキスト，HTML，およびバイナリの差異を表示します。

注：ほとんどの場合，標準の HTML を比較する方法を使用することをお勧めします。スクリプトに HTML タグ以外のタグが含まれている場合は，テキストを比較する方法を使用できます。

➤［**詳細設定**］:［詳細相関］ダイアログ・ボックスを開きます。

［詳細相関］ダイアログ・ボックス

このダイアログ・ボックスでは，区切り文字として扱う文字を指定できます。

➤［**区切り文字として扱う文字**］: 1 つまたは複数の標準設定以外の区切り文字を指定します。

➤［**追加区切り文字**］: 復帰，復帰改行，タブ文字など，標準設定の区切り文字を指定できます。設定を変更するには，区切り文字の横のチェック・ボックスをクリアします。

➤［**X 文字以下の差異を無視する**］: 相関を行うしきい値を指定できます。VuGen は記録時のデータを再生時のデータと比較する検索・取得処理で差異を検出します。差異の文字数がしきい値以上でないかぎり相関は行われません。標準設定の長さは 4 文字です。

➤［**大きな相関に警告を発行**］: サイズが 10 KB 以上の文字列を相関させようとしたときに警告を表示します。

相関オプションの設定

セッションの記録を開始する前に，相関オプションを設定します。

相関のオプションを設定するには，次の手順を実行します。

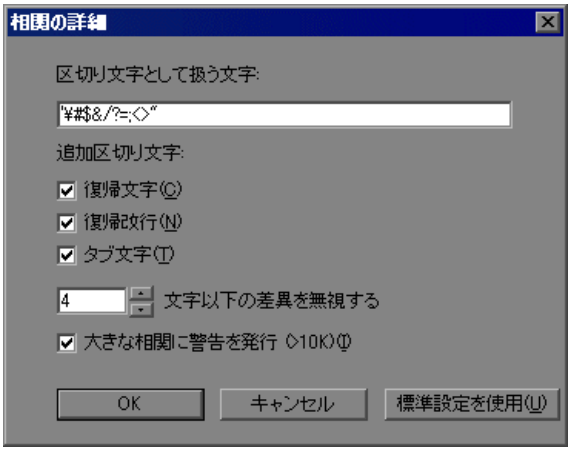
1 ［**オプション**］＞［**一般**］を選択し，［**相関**］タブを選択します。

2 ［**スナップショット ビューアで編集および Java アプレット実行を可能にする**］を選択して，スナップショット・ウィンドウでアプレットと Java スクリプトを実行できるようにします。

3 画像をスナップショット・ビューアに表示するよう VuGen に指示するには，［**スナップショット ビューアで画像をダウンロードする**］オプションを選択します。

4 比較の方法として，［**HTML を比較する**］または［**テキストを比較する**］（HTML の要素以外に対してのみ）を選択します。

5 区切り文字を設定するには，［**詳細設定**］をクリックして，［相関の詳細］ダイアログ・ボックスを開きます。



6 ［**区切り文字として扱う文字**］ボックスで，区切り文字として扱うすべての文字を指定します。

7 ［**追加区切り文字**］セクションで必要なオプションを選択して，標準で使用する 1 つまたは複数の区切り文字を指定します。

8 ［**X 文字以下の差異を無視する**］ボックスで，相関のしきい値を指定します。VuGen は記録時のデータを再生時のデータと比較する検索・取得処理で差異を検出します。差異の文字数がしきい値以上でないかぎり相関は行われません。

9 ［**大きな相関に警告を発行**］には，このオプションのチェック・ボックスを選択します。

10 ［**OK**］をクリックして詳細相関設定を受け入れ，ダイアログ・ボックスを閉じます。

11 ［一般オプション］ダイアログ・ボックスの［**OK**］をクリックし，［相関］の設定を受け入れてダイアログ・ボックスを閉じます。

相関の検索の実行

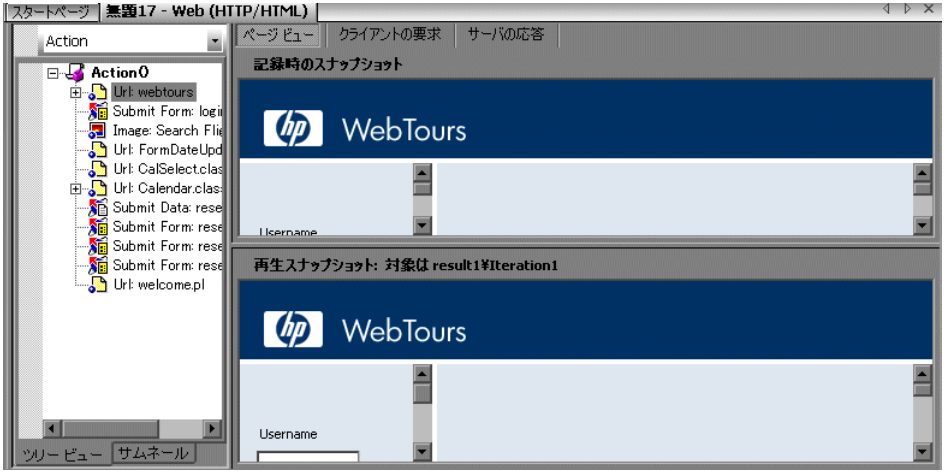
VuGen のスナップショット・ウィンドウを使って，スクリプト内のどの値が動的で相関が必要かを判断します。次の項では，スクリプト内を自動的に検索して差異を見つける方法と，VuGen を使って必要な相関を行う方法について説明します。

スクリプト内を検索して相関を実行するには，次の手順を実行します。

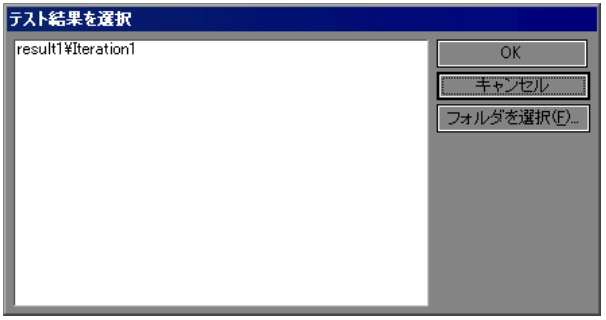
1 スクリプトを開き，ツリー・ビューで表示します（[**表示**] > [**ツリービュー**]）。スナップショットを表示します（[**表示**] > [**スナップショット**] > [**スナップショットを表示**]）。

2 左側の表示枠のツリー・ビューでスクリプトのステップを選択します。スナップショットが右側の表示枠に表示されます。

3 記録時のスナップショットと最初の再生時のスナップショットを両方とも表示するには，[**表示**] > [**スナップショット**] > [**記録と再生済み**] をクリックします。

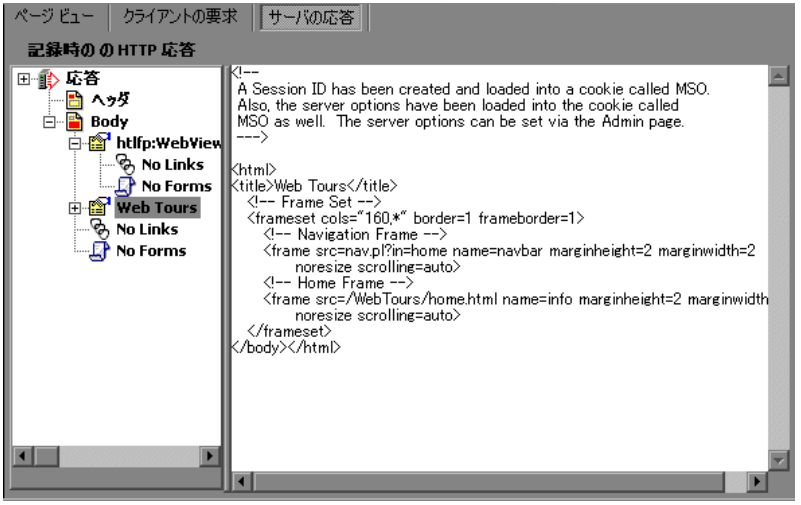


4 2 回目以降の再生時のスナップショットを使用するには，［**表示**］＞［**スナップショット**］＞［**反復の選択**］をクリックします。［テスト結果を選択］ダイアログ・ボックスが開き，スナップショット・ファイルが格納されているフォルダが表示されます。通常，スクリプトのフォルダの下には **result** フォルダと **Iteration** フォルダが表示されます。



5 スクリプトのサブ・フォルダ以外のフォルダにあるスナップショット・ファイルを選択するには，［**フォルダを選択**］をクリックします。目的のフォルダの場所を見つけ，［**OK**］をクリックします。

6 HTML コードを表示するには，［**サーバの応答**］タブをクリックします。「Body」の分岐を開きます。



ページ・ビューに戻るには，［**ページ ビュー**］タブをクリックします。

7［**仮想ユーザ**］＞［**相関を検索**］を選択するか，［**相関の検索**］ボタンをクリックします。VuGen はスクリプト内を検索し，相関が必要な動的データを見つけ，それらを［相関結果］タブに表示します。

8［**次の差異を表示**］リスト・ボックスで，すべての差異を表示するか，フィルタ方法を選択します。［**すべてのアクション**］，［**現在のアクション**］，または［**現在のステップのみ**］を選択できます。

相関させる差異の決定

差異のリストを生成したら，相関させる差異を決定する必要があります。相関させる必要がない差異を誤って相関させると，再生に悪影響を与える可能性があります。

相関を必要とする可能性が最も高いのは，次の文字列です。

- **ログイン文字列** ― セッション ID やタイム・スタンプなどの動的なデータを含むログイン文字列。
- **日付やタイム・スタンプ** ― 日付，タイム・スタンプ，またはその他のユーザ・アカウント情報を使った文字列。
- **共通のプレフィックス** ― 文字列の前に付く **SessionID** や **CustomerID** などの共通のプレフィックス。

相関させる必要がある差異かどうかが不明な場合は，その差異だけを相関させてスクリプトを実行し，問題が解決したかどうかを再生ログで確認します。

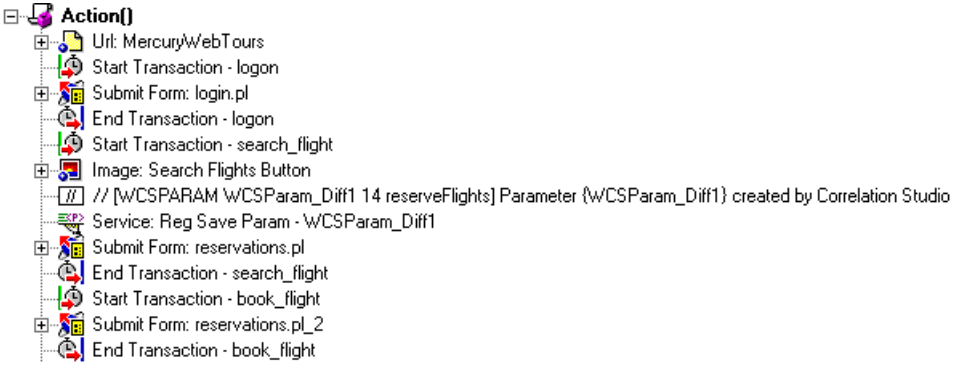
また，記録された文字列と再生された文字列の一部だけが異なる場合も，差異を相関させる必要があります。たとえば，SessionID 文字列のプレフィックスとサフィックスが同じでも，中間の文字が異なる場合は相関させる必要があります。

差異を相関させる必要があることを確認したら，それを相関させるように VuGen を設定します。

差異を相関させるには，次の手順を実行します。

1 ［相関］タブに差異を表示し，相関させる差異を選択します。一度に相関させる差異は 1 つだけにすることをお勧めします。

2 ［**相関**］をクリックします。相関された差異の横に緑のチェック・マークが付けられ，スクリプトに **web_reg_save_param** 関数が挿入されます。



相関させるすべての差異について，この手順を繰り返します。

3 検出された相関からルールを作成するには，相関を選択して［**ルールの作成**］をクリックします。これは，右クリックして表示されるメニューからも選択できます。VuGen は，ルールが作成されたことを確認するメッセージを発行します。

このルールを表示するには，［記録オプション］（CTRL + F7）を開き，［**相関**］ノードを選択します。［**Correlation Studio**］エントリを拡張し，ルールを選択します。

4 相関を取り消すには，差異を選択して［**相関を削除**］をクリックします。

5 ［**ファイル**］＞［**上書き保存**］を選択して，スクリプトへの変更を保存します。

手作業による相関

Web Vuser の場合，VuGen の自動相関またはルールに基づいた相関では，通常はスクリプトを正常に実行できるよう，スクリプトの動的関数を相関させます。VuGen のスナップショットの比較機能を使って，記録セッションの後に相関を行うこともできます。

自動相関が適用されなかったワイヤレス Vuser やその他の Vuser スクリプトについては，手作業でスクリプトを相関させることができます。スクリプトを手作業で相関させるには，コード相関関数を追加します。パラメータにデータを動的に保存する関数は，**web_reg_save_param** です。

スクリプトを実行すると，**web_reg_save_param** 関数はそれ以降にアクセスする HTML ページ内を検索します。左と右の境界を指定すると，VuGen によってこれらの境界の範囲内でテキストが検索されます。テキストが見つかると，テキストはパラメータに保存されます。

関数の構文は次のとおりです。

```
int web_reg_save_param (const char *mpszParamName, ＜属性のリスト＞ , LAST);
```

使用できる属性を次の表に示します。属性値の文字列（Search=all など）では，大文字と小文字は区別されません。

NotFound	境界が見つからず，空の文字列が生成されたときの処理方法。標準設定の「ERROR」では，境界が見つからない場合にエラーが発行されます。「EMPTY」に設定した場合，エラー・メッセージは発行されず，スクリプトの実行が継続されます。スクリプトに対して［エラーでも処理を継続する］を有効にしている場合は，NOTFOUND を「ERROR」に設定していても，境界が見つからない場合にスクリプトが継続されます。ただし，エラー・メッセージは詳細ログ・ファイルに記録されます。
LB	パラメータまたは動的データの左の境界。このパラメータは，空でない，NULL で終わる文字列でなければなりません。境界のパラメータでは，大文字と小文字が区別されます。大文字と小文字の区別をしないようにするには，境界の後に「/IC」を追加します。バイナリ・データを指定するには，境界の後に「/BIN」を指定します。
RB	パラメータまたは動的データの右の境界。このパラメータは，空でない，NULL で終わる文字列でなければなりません。境界のパラメータでは，大文字と小文字が区別されます。大文字と小文字の区別をしないようにするには，境界の後に「/IC」を追加します。バイナリ・データを指定するには，境界の後に「/BIN」を指定します。
RelFrameID	要求された URL を基準にした HTML ページの階層レベル。値は，ALL か数値です。
Search	検索の範囲（区切り文字で区切られたデータを検索する場所）。値は，Headers（ヘッダだけを検索），Body（ヘッダでなく本体のデータだけを検索），ALL（本体とヘッダを検索）のいずれかです。標準の値は ALL です。
ORD	このパラメータは省略可能で，何回目に出現する検索内容かを序数で示します。標準の序数は 1 です。「All」を指定すると，パラメータの値が配列に保存されます。

SaveOffset	見つかった値において，パラメータに保存する部分の文字列の開始位置を示すオフセットです。標準設定は 0 です。オフセットの値は負でない数字でなくてはなりません。
Savelen	パラメータに保存する部分文字列の長さ。部分文字列は，見つかった値においてオフセット位置から始まります。標準設定は -1 で，文字列の末尾までを保存することを示します。
Convert	データに適用する変換方式。 HTML_TO_URL：HTML でエンコードされたデータを URL でエンコードされたデータ形式に変換します。 HTML_TO_ TEXT：HTML でエンコードされたデータを通常のテキストに変換します。

スクリプトを手作業で相関させるには，次の手順を実行します。

1 動的データを含むステートメントと，データの境界を示すパターンを探します。詳細については，668 ページ「動的文字列の境界の定義」を参照してください。

2 スクリプトの中で，動的データを独自のパラメータ名に置き換えます。詳細については，下記を参照してください。

3 スクリプト内の動的データが含まれるステートメントの前に **web_reg_save_param** 関数を追加します。詳細については，666 ページ「相関関数の追加」または『**Online Function Reference**』（英語版）（［**ヘルプ**］＞［**関数リファレンス**］）を参照してください。

動的データのパラメータへの置き換え

記録されたステートメントから実際の動的データを特定します。次に，スクリプト全体からその動的データを検索し，パラメータに置き換えます。パラメータに名前を付け，{param_name} のように中括弧で囲みます。1 つのスクリプトには，最大 64 個のパラメータを設定できます。

動的データをパラメータに置き換えるには，次の手順を実行します。

VuGen のメイン・ウィンドウで［**編集**］＞［**置換**］を選択して，［検索と置換］ダイアログ・ボックスを表示します。動的データをスクリプト全体から検索して，パラメータに置き換えます。

相関関数の追加

スクリプトの動的データを保存するには，**web_reg_save_param** ステートメントを挿入します。この関数は，再生中に動的データの実行時の値を保存するパラメータを作成するように VuGen に指示します。

スクリプトを実行すると，**web_reg_save_param** 関数はそれ以降にアクセスする HTML ページ内を検索します。左の境界，任意の文字列，右の境界が並んでいる文字列を検索します。文字列が見つかると，VuGen は左と右の境界の間にある文字列を，関数の引数に指定したパラメータに代入します。指定された数の出現箇所が見つかると，**web_reg_save_param** はそれ以上 HTML ページを検索しません。Vuser はスクリプト内の次のステップから実行を続けます。

Web Vuser の相関の例

次のように，スクリプトに動的なセッション ID が含まれているとします。

```
web_url("FirstTimeVisitors",
    "URL=/exec/obidos/subst/help/first-time-visitors.html/002-8481703-
        4784428>Buy books for a penny ",
    "TargetFrame=",
    "RecContentType=text/html",
    "SupportFrames=0",
     LAST);
```

前述のステートメントの前に，次のように **web_reg_save_param** ステートメントを挿入します。

```
web_req_save_param ("user_access_number", "NOTFOUND=ERROR", "LB=first-
time-visitors.html/","RB=>Buy books for a penny" , "ORD=6", LAST );
```

相関ステートメントを実装した後，変更後のスクリプトは次のようになります。user_access_number は動的なデータを表すパラメータの名前です。

```
web_url("FirstTImeVisitors",
    "URL=/exec/obidos/subst/help/first-time-"
        "visitors.html/{user_access_number}Buy books for a penny",
    "TargetFrame=",
    "RecContentType=text/html",
    "SupportFrames=0",
     LAST);
```

注：各相関関数は，以降の HTTP 要求について，動的データを一度だけ取得します。スクリプト内の後方にある別の HTTP 要求によって新しい動的データが生成される場合は，相関関数をもう 1 つ挿入しなければなりません。

ワイヤレス Vuser の相関の例

次のように，スクリプトに動的なセッション ID が含まれていると仮定します。

```
web_url("login.po;sk=IuZSuuRlHUMnpF-wpK8PzEpy(1YOSBSMy)",
    "URL=http://room33.com/portal/login.po;sk=IuZSuuRlHUMnpF-
        wpK8PzEpy(1YOSBSMy)",
    "Resource=0",
    "RecContentType=text/vnd.wap.wml",
    "Mode=HTML",
    LAST);
```

web_reg_save_param ステートメントを前述のステートメントの前に挿入し，動的な値をパラメータに置き換えます。次の例では，web_reg_save_param 関数を使って，ログイン ID 文字列を SK という変数に保存しています。RB/BIN 属性に従ってバイナリ・データを保存し，左の境界を「sk=」として設定しています。

```
web_reg_save_param(
    "SK",
"LB=sk=",
"RB/BIN=#login¥¥x00¥¥x01¥¥x03",
"Ord=1",
LAST);


web_url("login.po;sk={SK}",
    "URL=http://room33.com/portal/login.po;sk={SK}",
    "Resource=0",
    "RecContentType=text/vnd.wap.wml",
    "Mode=HTML",
    LAST);
```

動的文字列の境界の定義

動的データの特定と指定は，次のガイドラインに従って行います。

- 動的データの場所は，記録されたスクリプト内ではなく，必ず HTML コード内で探すようにします。
- 動的データのすぐ左側にある文字列を特定します。この文字列は動的データの左の境界を示します。
- 動的データのすぐ右側にある文字列を特定します。この文字列は動的データの右の境界を示します。
- **web_reg_save_param** は，指定された境界の間（境界を含まない）にある文字を検索し，左の境界の 1 バイト後から，右の境界の 1 バイト前までの情報を保存します。**web_reg_save_param** は，境界文字の重複をサポートしません。たとえば，入力バッファが {a{b{c} で，「{」が左の境界，「}」が右の境界の場合，検索に一致するのは「c」で，それ以外に一致するものはありません。この場合，左右の境界は検出されましたが，境界の重複は認識されないので，「c」が唯一の一致項目となります。

  標準では，境界文字列は最大 256 文字です。最大文字数を増やすには，スクリプトに **web_set_max_html_param_len** 関数を挿入します。たとえば，次の関数は最大文字数を 1024 文字に増やします。

  ```
  web_set_max_html_param_len("1024");
  ```

第 44 章

Web（HTTP/HTML）― XML ページの処理

VuGen の Web Vuser は XML コードを含む Web ページをサポートします。

本章の内容

XML ページのテストについて

VuGen は，Web ページに含まれる XML コードの記録および再生をサポートしています。

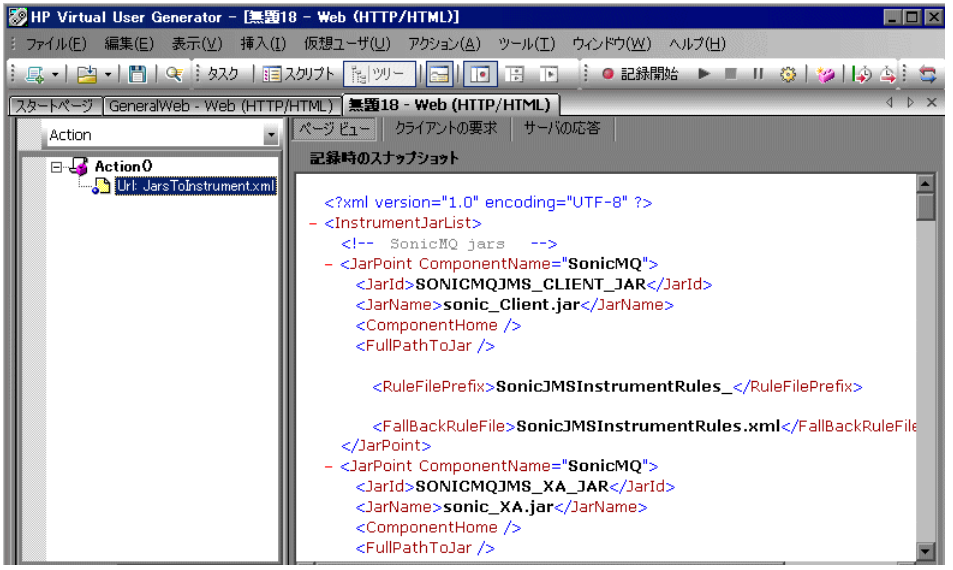
XML コードは，スクリプトに通常の URL ステップまたはユーザ定義の要求として現れます。VuGen は XTML を検出し，ユーザがそれぞれの文書型定義（DTD）およびそのエンティティと属性を参照できるようにします。MIME タイプが **RecContentType** 属性に表示された場合，あるいは再生中にサーバによって返された MIME タイプが **xml**（**application/xml** または **text/xml**）で終了した場合に XML を解釈します。DTD は色分けされているので，各要素を識別できます。DTD のツリー・ビューの分岐の展開と折りたたみも可能です。

DTD を展開する場合には，属性の値をパラメータ化できます。また標準の相関関数を使って相関できるように値を保存できます。相関関数の詳細については，『**Online Function Reference**』（英語版）（［**ヘルプ**］＞［**関数リファレンス**］）を参照してください。

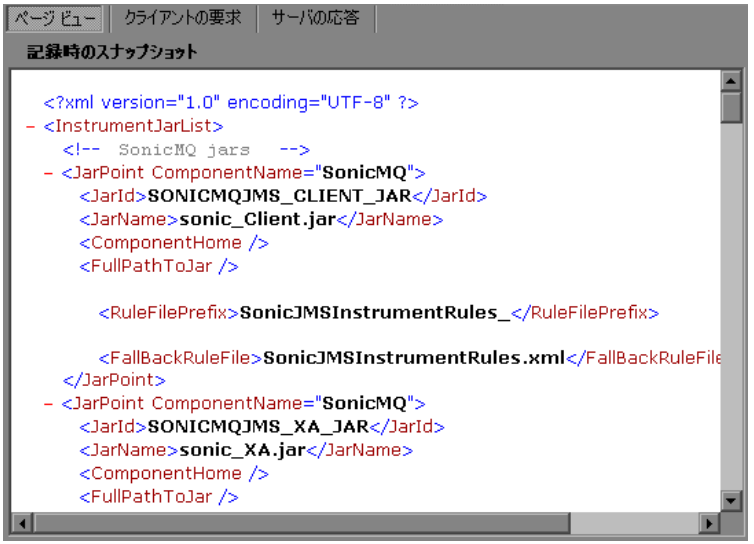
注：HTML ページに埋め込まれた断片的な XML である **XML アイランド**を含む DTD は表示できません。VuGen は，全体が XML であるページのみを表示します。

XML の URL ステップとしての表示

XML コードを含むページをテストする 1 つの方法は，VuGen で記録することです。まず，XML ページを通常の Web ページと同じ方法で記録します。VuGen は，DTD およびすべての XML 要素を記録します。XML ページのスナップショットは作成されませんが，XML ステップごとに，XML コードが［サーバの応答］タブのスナップショット・フレームに表示されます。



VuGen が作成した展開可能な DTD 階層は，色分けされてスナップショット表示枠に表示されます。項目の分岐を展開するにはプラス（+）記号をクリックし，折りたたむにはマイナス（-）記号をクリックします。XML タグは茶色，値は黒で表示されます。



定数値をパラメータで置き換えるには，値を選択して，右クリックし［**パラメータで置換**］を選択します。パラメータ化の標準手続きに従います。詳細については，『**第 1 巻 － VuGen の使用**』の「パラメータの作成」を参照してください。

XML ページの［サーバ応答］と［クライアントの要求］もタブをクリックして表示できます。次の例は，XML ページのサーバ応答を示しています。XML ツリーのすべての分岐は展開することも折りたたむことも可能です。

次の例は，XML ページのヘッダに対するクライアントの要求を示しています。

XML をユーザ定義の要求として挿入

XML コードをユーザ定義の要求として挿入して，XML ページをテストすることもできます。このモードでは，DTD の要素が［**ユーザ定義要求プロパティ**］ボックス内にテキスト形式または XML 形式で表示されます。

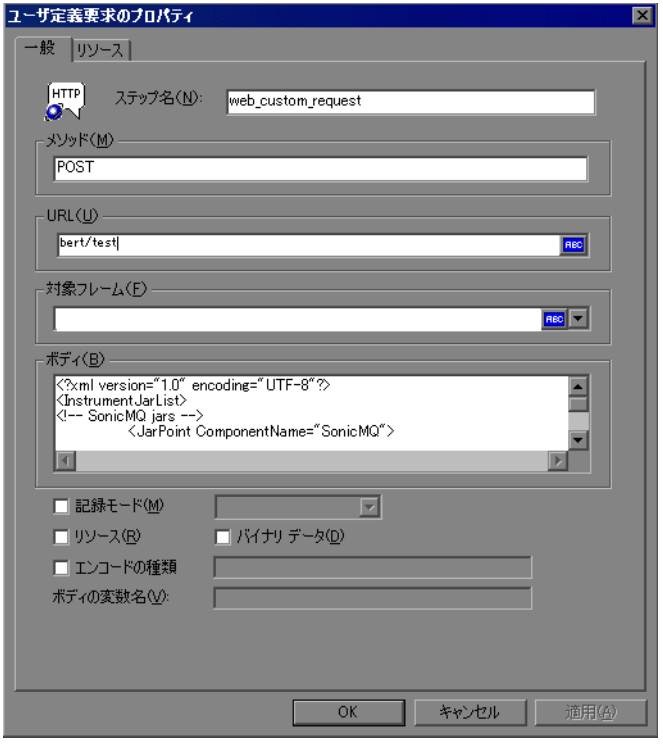
XML コードをユーザ定義の要求として追加するには，次の手順を実行します。

1 ツリー・ビュー・モードでスクリプトを表示し，希望する場所にカーソルを置き，［**挿入**］＞［**新規ステップ**］を選択します。［ステップの追加］ダイアログ・ボックスが開きます。

2 リストの下部までスクロールして，［**ユーザ定義要求**］を選択します。［**OK**］をクリックします。［ユーザ定義要求のプロパティ］ダイアログ・ボックスが開きます。

3 ステップ名，メソッド（GET または POST），URL，対象フレーム（任意）を入力します。

4 XML コードをブラウザまたはエディタからコピーし，［ユーザ定義要求のプロパティ］ボックスの［**ボディ**］セクションに貼り付けます。



5 ［**記録モード**］，［**リソース**］，［**バイナリ データ**］などの再生に関するオプションを選択します。詳細については，第 41 章「Web とワイヤレス Vuser スクリプトの変更」を参照してください。

6 ［**OK**］をクリックします。ユーザ定義の要求のステップがスクリプトに挿入されます。

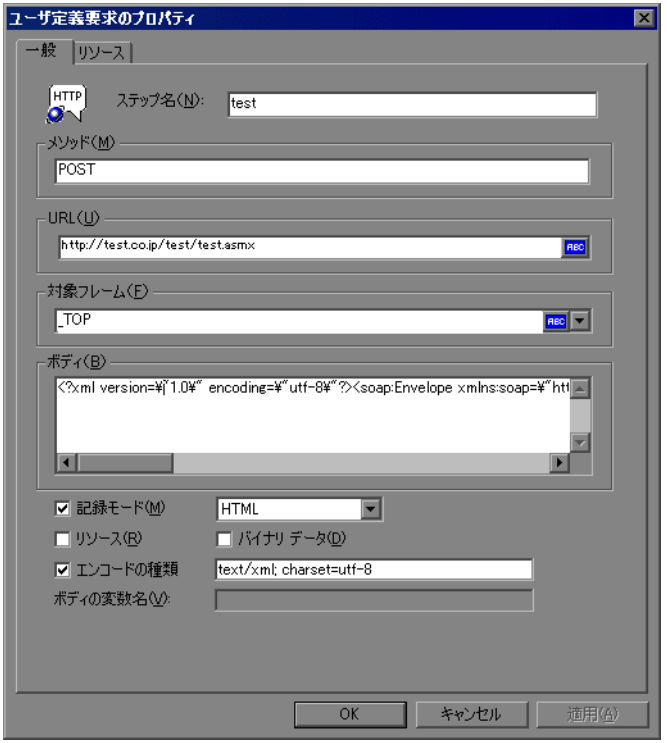
XML ユーザ定義要求のステップの表示

ユーザ定義の要求のステップとして挿入された XML コードは，いつでも表示したり変更したりすることができます。VuGen のビューアを使って，DTD の階層を表示し，必要に応じて要素の分岐を展開したり折りたたんだりできます。

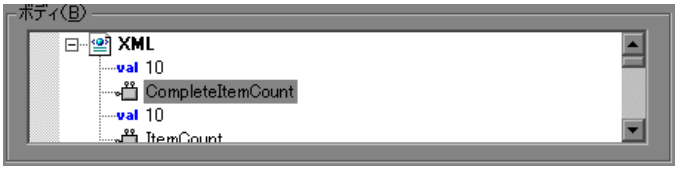
ユーザ定義の要求のステップの XML コードを表示するには，次の手順を実行します。

1 スクリプトをツリー・ビューで表示し，XML コードを表示するステップを選択します。

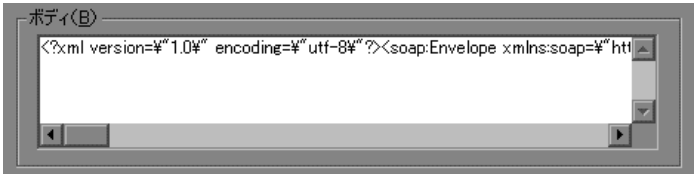
2 右クリックして表示されるメニューから［**プロパティ**］を選択します。［ユーザ定義要求のプロパティ］ダイアログ・ボックスが開きます。



ダイアログ・ボックスの最下部に XML コードが表示されます。**RecContentType** 属性が「**text/xml**」に設定されていると，標準設定では，コードが XML 形式の階層として表示されます。このモードのときには，XML コードの編集はできません。

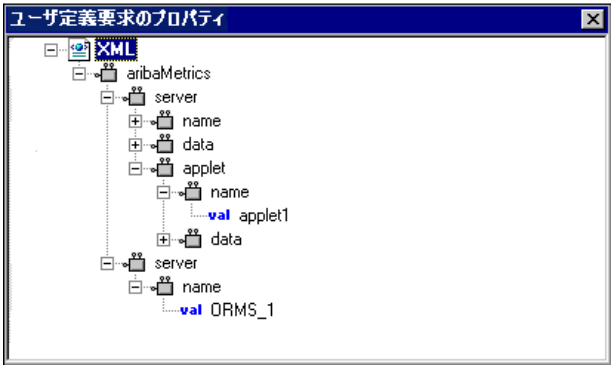


RecContentType 属性が「**text/xml**」以外に設定されているときには，コードはテキスト形式で表示されます。このモードのときには，XML コードは編集可能です。



3 テキストと XML の表示を切り替える場合は，右クリックして表示されるメニューから［**XML として表示**］または［**テキストとして表示**］を選択します。

4 XML 表示になっている場合，ウィンドウを大きくしてコードを表示できます。右クリックして表示されるメニューから［**拡張表示**］を選択します。ダイアログ・ボックスの表示に戻るには，右クリックして表示されるメニューから［**標準表示**］を選択します。



第 45 章

Oracle NCA プロトコル

VuGen を使って，Oracle NCA ユーザをエミュレートするスクリプトを作成できます。まず，VuGen で典型的な NCA のビジネス・プロセスを記録します。次に，システムと対話するユーザをエミュレートするスクリプトを実行します。

本章の内容

Oracle NCA Vuser スクリプトの作成について

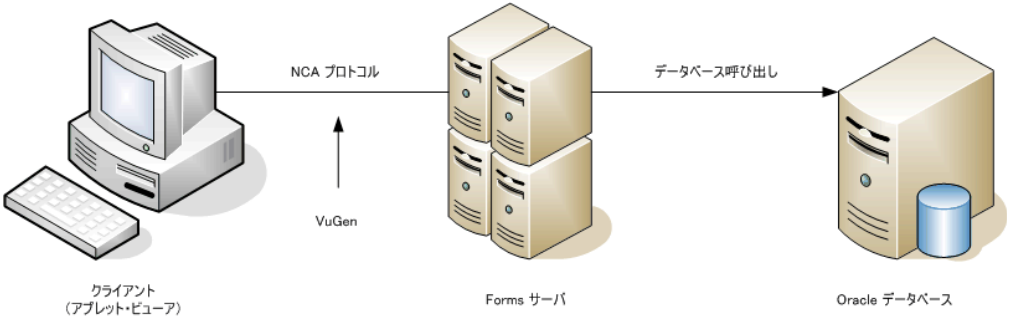
Oracle NCA は，Java ベースのデータベース・プロトコルです。データベース・クライアントであるアプレット・ビューアは，ブラウザを使用して起動します。NCA データベースに対するアクションは，アプレット・ビューアを使用して実行します。

アプレット・ビューアによりクライアント・ソフトウェアが不要となり，アプレット・ビューアをサポートするあらゆるプラットフォームでデータベース・アクションを実行できるようになります。Oracle NCA クライアントをエミュレートするために特別に設計された Vuser の種類があります。

NCA の環境は 3 層構造の環境です。ユーザはまずブラウザから Web サーバへ http 呼び出しを送信します。この呼び出しは，Oracle Applications アプレットを起動するスタートアップ HTML ページにアクセスします。このアプレットはクライアント・マシンでローカルに実行します。以降の呼び出しはすべて，独自の NCA プロトコルを使ってクライアントと Forms サーバの間で通信されます。

クライアント（アプレット・ビューア）は，データベース・サーバに情報を送信するアプリケーション・サーバ（Oracle Forms サーバ）と通信します。

VuGen は，クライアントと Forms サーバ（アプリケーション・サーバ）間の NCA 通信を記録し，再生します。



シングル・プロトコル・スクリプトを作成した場合であっても，Oracle NCA セッションが記録される際には，VuGen によってすべての NCA アクションおよび Web アクションが記録されます。テストにおいて Web 関数が必要であることが事前にわかっている場合は，最初から Oracle NCA プロトコルと Web プロトコルを対象としたマルチ・プロトコル・スクリプトを作成します。

最初に Oracle NCA プロトコルを対象としたシングル・プロトコル・スクリプトを作成し，後でテストのために Web 関数が必要となった場合は，セッションを再記録しなくても VuGen でスクリプトを再生成して Web 関数を追加できます。これは，［スクリプトの再生成］ダイアログ・ボックスの［プロトコル］ノードで指定します。詳細については，『**第 1 巻 ― VuGen の使用**』の「VuGen を使った記録」を参照してください。

Oracle NCA Vuser の開発の概要

次の手順は，Oracle NCA Vuser スクリプトの作成方法の概要を示します。

1 記録するマシンが正しく設定されていることを確認します。

VuGen を起動する前に，Oracle NCA アプレット・ビューアが動作するようにマシンが設定されていることを確認します。また，VuGen がお使いの Oracle Forms のバージョンをサポートしていることを確認する必要があります。詳細については，680 ページ「記録作業のガイドライン」と Readme ファイルを参照してください。

2 スケルトン Oracle NCA Vuser スクリプトを作成します。

VuGen を使用して，スケルトン Oracle NCA Vuser スクリプトを作成します。詳細については，『**第 1 巻 ― VuGen の使用**』の「VuGen を使った記録」を参照してください。

3 典型的なユーザ・アクションを記録します。

記録を開始し，アプレット・ビューアを使って典型的なアクションとビジネス・プロセスを実行します。VuGen によってアクションが記録され，Vuser スクリプトが生成されます。詳細については，『**第 1 巻 ― VuGen の使用**』の「VuGen を使った記録」を参照してください。

4 Vuser スクリプトを拡張します。

［挿入］メニューを使って，トランザクション，ランデブー・ポイント，コメント，メッセージなどを追加して，Vuser スクリプトを拡張します。詳細については，『**第 1 巻 ― VuGen の使用**』の「Vuser スクリプトの拡張」を参照してください。

5 スクリプトをパラメータ化します。

記録された定数をパラメータと置き換えます。詳細については，『**第 1 巻 ― VuGen の使用**』の「パラメータの作成」を参照してください。

6 スクリプトの実行環境プロパティを設定します。

Vuser スクリプトの実行環境の設定を行います。実行環境設定は，スクリプト実行のいくつかの特性を規定します。詳細については，『**第 1 巻 ― VuGen の使用**』の「実行環境の設定」を参照してください。

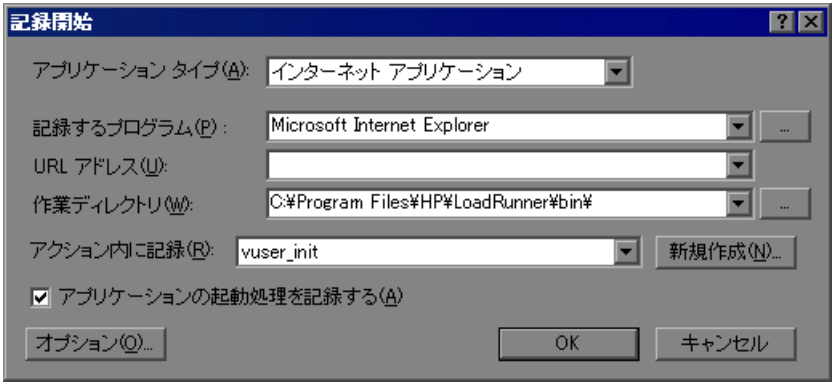
7 Vuser スクリプトを保存して実行します。

VuGen からスクリプトを実行して，実行時の情報を実行ログで確認します。詳細については，『**第 1 巻 ― VuGen の使用**』の「スタンドアロン・モードでの Vuser スクリプトの実行」を参照してください。

記録作業のガイドライン

Oracle NCA Vuser スクリプトを記録する際には，次のガイドラインに従います。

➤ Oracle NCA セッションを記録するときに VuGen が使用するブラウザを指定します。［記録開始］ダイアログ・ボックスの［**記録するプログラム**］リストで，使用するブラウザを選択します。このリストには，使用可能なブラウザがすべて含まれています。



➤ 記録を開始する前にすべてのブラウザを閉じます。

➤ vuser_init セクションにログイン・プロシージャを記録します。Actions セクションに典型的なビジネス・プロセスを記録します。スクリプトを実行するときに，特定のビジネス・プロセスを複数回反復するように指定できます。詳細については，『**第 1 巻 ― VuGen の使用**』の第 5 章「VuGen を使った記録」を参照してください。

```
vuser_init()
{
    nca_set_connect_opt(SCALE_INFO, 11, 18);
    nca_connect_server("labm1orcl05.devlab.ad", "9000",
"module=/opt/applvis/visappl/fnd/11.5.0/forms/US/FNDSCSGN
userid=APPLSYSPUB/PUB@VIS fndnam=APPS record=names  ");
    nca_set_window("Oracle Applications");
    nca_edit_set("SIGNON_USERNAME_0", "OPERATIONS");
    nca_obj_type("SIGNON_USERNAME_0", '¥t', 0);
    nca_edit_set("SIGNON_PASSWORD_0", lr_decrypt("4768d647f4f1840f2e46d5"));
    nca_button_press("SIGNON_CONNECT_BUTTON_0");
return 0;
}
```

- Netscape の制限により，使用しているマシンで別の Netscape ブラウザがすでに動いている場合は，Netscape 内で Oracle NCA セッションを起動することができません。
- VuGen では，マルチ・プロトコル・モードで Forms Listener Servlet を使用することで Oracle Forms アプリケーションを記録できます。アプリケーション・サーバが **Forms Listener Servlet** を使用して各クライアントの実行時プロセスを作成します。**Forms Server Runtime** という実行時プロセスは，クライアントとの永続的な接続を維持し，サーバとの間で情報をやり取りします。

  再生時に Forms 4.5 をサポートするには，**mdrv.dat** ファイルに以下を設定します。

```
[Oracle_NCA]
ExtPriorityType=protocol
WINNT_EXT_LIBS=ncarp110.dll
WIN95_EXT_LIBS=ncarp110.dll
LINUX_EXT_LIBS=liboranca.so
SOLARIS_EXT_LIBS=liboranca.so
HPUX_EXT_LIBS=liboranca.sl
AIX_EXT_LIBS=liboranca.so
LibCfgFunc=oracle_gui_configure
UtilityExt=lrun_api
```

  Forms 4.5 以降のサポートに戻すには，最初の値に戻します。

名前によるオブジェクトの記録の有効化

Oracle NCA スクリプトを記録するときには，標準オブジェクト ID ではなくオブジェクト名を使用してセッションを記録する必要があります。オブジェクト ID はサーバによって動的に生成され，記録時と再生時とでは異なるため，オブジェクト ID を使用してスクリプトを記録すると，再生は失敗します。**nca_connect_server** ステートメントを調べれば，スクリプトがオブジェクト名で記録されているかどうかを確認できます。

```
nca_connect_server("199.35.107.119","9002"/*version=11i*/,"module=/d1/oracle/visappl/fnd/11.5.0/forms/US/FNDSCSGN userid=APPLSYSPUB/PUB@VIS fndnam=apps record=names ");
```

nca_connect_server 関数に **record=names** 引数が含まれていなければ，オブジェクト ID が記録されているということです。オブジェクト名を記録するように VuGen を設定するには，次のいずれかを変更します。

- スタートアップ HTML ファイル
- 記録する URL
- Forms 設定ファイル

すべてのオブジェクトの開発者名を取得する機能は，Oracle Forms6i Patch 9（Oracle Forms Version: 6.0.8.18.3）で初めて導入されました。そのため，Oracle Forms 6i Patch 9 のリリース前に作成された Test Starter Kit スクリプトでは，編集フィールドを除き，オブジェクトの物理的記述に開発者名は含まれません。

スタートアップ HTML ファイル

スタートアップ HTML ファイルにアクセスできる場合は，そのスタートアップ・ファイル，つまり Oracle NCA アプリケーションを起動したときにロードされるファイルに **record=names** フラグを設定すれば，オブジェクト ID ではなくオブジェクト名が記録されます。

アプレット・ビューアの起動時に呼び出されるスタートアップ・ファイルを編集します。次の行を変更します。

```
< PARAM name="serverArgs ..... fndnam=APPS" >
```

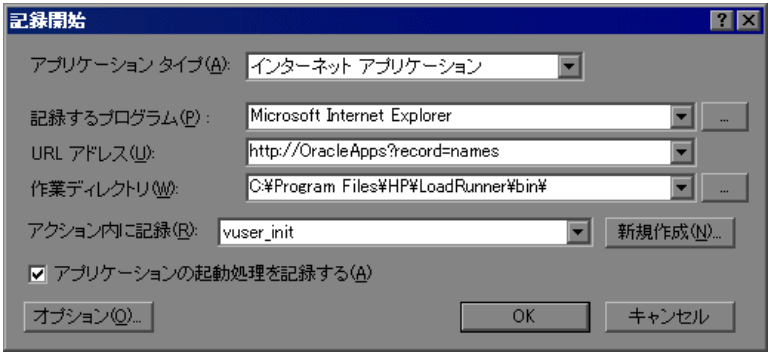
次に，Oracle キー「record=names」を次の形式で追加します。

```
<PARAM name="serverArgs ... fndnam=APPS record=names">
```

記録する URL

スタートアップ HTML ファイルにアクセスできない場合は，記録する URL を変更することで，Oracle NCA がオブジェクト ID ではなくオブジェクト名を記録するようにできます。次の方法は，スタートアップ HTML ファイルがロード時にほかのファイルを参照しない場合にのみ有効です。

この方法では，［記録開始］ダイアログ・ボックスの URL の後，つまり記録する URL 名の後に「**?record=names**」を追加します。これにより，VuGen はセッションの中でオブジェクト名を記録できるようになります。



Forms 設定ファイル

Forms Web CGI 設定ファイル **formsweb.cfg** を参照するスタートアップ HTML ファイルがアプリケーションにある場合（よく行われる参照です）は，スタートアップ・ファイルに **record=names** を追加すると，問題が生じる場合があります。

この場合は，設定ファイルに変更を加える必要があります。

設定ファイルに変更を加えてオブジェクト名を記録できるようにするには，次の手順を実行します。

1 Forms Web CGI 設定ファイルに移動します。

2 このファイルに新しいパラメータを定義します（以下に示す Web CGI 設定ファイルの例を参照）。

```
serverApp=forecast
serverPort=9001
serverHost=easgdev1.dats.ml.com
connectMode=socket
archive=f60web.jar
archive_ie=f60all.cab
xrecord=names
```

3 スタートアップ HTML ファイルを開き，PARAM NAME="serverArgs" を探します。

4 record=%xrecord% のように，変数名を ServerArgs パラメータに引数として追加します。

```
<PARAM NAME="serverArgs" VALUE="module=%form% userid=%userid%
%otherParams% record=%xrecord%">
```

5 または，Oracle Forms のインストール・ディレクトリにある **basejini.htm** ファイルを編集します。このファイルは，JInitiator スタイルのタグを使用して Forms アプレットを読み込む Web 上のフォームを実行するための標準の HTML ファイルです。basejinin.hmt ファイルで，パラメータ定義に次の行を追加します。

```
<PARAM NAME="recordFileName" VALUE="%recordFileName%">
```

<EMBED> タグに次の行を追加します。

```
serverApp="%serverApp%"
logo="%logo%"
imageBase="%imageBase%"
formsMessageListener="%formsMessageListener%"
recordFileName="%recordFileName%"
```

サーブレット設定ファイル **formsweb.cfg** ではなく basejini.htm ファイルを編集することの難点は，Oracle Forms を再インストールすると，このファイルが置き換えられてしまう点です。これを避けるには，**basejini.htm** ファイルのコピーを作成し，そのコピーを別の場所に保管しておきます。サーブレット設定ファイルで，**baseHTMLJinitiator** パラメータを編集して新しいファイルを指すようにします。

Personal Home Page からの Oracle Applications の使用

Personal Home Page にログインして Oracle Forms 6i を起動する場合，ユーザ・レベルでいくつかのシステム・プロファイル・オプションを設定する必要があります。これらの変数は，全ユーザに適用されるサイト・レベルではなく，ユーザ・レベルで設定することをお勧めします。

「ICX: Forms Launcher」プロファイルを設定するには，次の手順を実行します。

1 アプリケーションにサインオンし，［System Administrator］の権限を選択します。

2 ［Navigator］メニューから［**Profile/System**］を選択します。

3 ［**Find System Profile Values**］フォームで次の操作をします。

   a ［**Display:Site**］オプションを選択します。

   b **Users** = ＜ログオン・ユーザ名＞（operations，mfg など）

   c **Enter Profile** =%ICX%Launch%

   d ［**検索**］をクリックします。

4 「**ICX:Forms Launcher**」プロファイルに対する User の値を更新します。

   ➤ URL に対してパラメータが渡されていない場合，ユーザ指定値の後ろに次の文字列を追加します。**?**play=&record=names

   ➤ URL に対してパラメータが渡されていれば，ユーザ指定値の後ろに次の文字列を追加します。**&**play=&record=names

5 トランザクションを保存します。

6 Oracle Forms セッションからログアウトします。

7 Personal Home Page セッションからログアウトします。

8 **Personal Home Page** から，自分の名前を使って再度サインオンします。

「ICX: Forms Launcher」プロファイル・オプションをユーザ・レベルで更新できない場合には，［**Application Developer**］権限を開き，ICX_FORMS_LAUNCHER プロファイルに対する［**Updatable**］オプションを選択します。

URL に渡す最初のパラメータは，疑問符（?）で始めなければなりません。その後に続くパラメータはすべてアンパサンド（&）を付けて渡します。ほとんどの場合，URL にパラメータが含まれています。含まれているかどうかは，疑問符を検索することで調べることができます。

Oracle NCA Vuser 関数の使用

VuGen は典型的な NCA ビジネス・プロセスを記録して，Oracle NCA に固有の関数を生成します。これらの関数ではプレフィックスに **nca** を使用します。

NCA 関数は，次のカテゴリに分類されます。ボタン・オブジェクト，接続，コンボ・ボックス・オブジェクト，編集および編集ボックス・オブジェクト，Flexfield ウィンドウ，Java オブジェクト，リスト・オブジェクト，メニュー・オブジェクト，メッセージ・オブジェクト，オブジェクト，応答オブジェクト，スクロール・オブジェクト，セッション，タブ・オブジェクト，ツリー・オブジェクト，およびウィンドウ・オブジェクトの関数。

また，手作業で任意の関数をプログラミングして，スクリプトに挿入することもできます。テキスト・ビューで，Intellisense と自動補完機能を利用して，手作業で新しい関数を追加できます。ツリー・ビューで［**挿入**］＞［**新規ステップ**］を選択し，必要なステップを選択します。

Oracle NCA Vuser 関数の詳細については，『**Online Function Reference**』（英語版）（［**ヘルプ**］＞［**関数リファレンス**］）を参照してください。

C Vuser 関数（**lr_output_message** や **lr_rendezvous** など）を使ってスクリプトをさらに拡張できます。これらの関数の使用法については，『**第 1 巻 ― VuGen の使用**』の「Vuser スクリプトの拡張」を参照してください。

Oracle NCA Vuser について

Oracle NCA Vuser スクリプトの作成時，VuGen は，クライアントとアプリケーション・サーバ間の NCA 通信をすべて記録します。記録中，VuGen はコンテキスト・センシティブ関数を生成します。コンテキスト・センシティブ関数は，データベースに対するアクションを GUI オブジェクト（ウィンドウ，リスト，ボタンなど）を基準にして表します。記録の実行中，VuGen によってコンテキスト・センシティブ関数が Vuser スクリプトに挿入されます。

記録を終えた後で，スクリプト内の関数に変更を加えたり，関数を追加してスクリプトを拡張したりできます。Vuser スクリプトの拡張については，『**第 1 巻 ー VuGen の使用**』の「Vuser スクリプトの拡張」を参照してください。使用可能な Oracle NCA Vuser 関数の一覧については，686 ページ「Oracle NCA Vuser 関数の使用」を参照してください。これらの関数の詳細については，『**Online Function Reference**』（英語版）（［**ヘルプ**］＞［**関数リファレンス**］）を参照してください。

次のコード例では，ユーザがリストから項目を選択して（**nca_list_activate_item**），ボタンを押し（**nca_button_press**），リストの値を取得（**nca_lov_retrieve_items**）しました。そして，エディット・フィールド内をクリック（**nca_edit_click**）しています。オブジェクトの論理名は，これらの関数のパラメータとなっています。

```
...
nca_lov_select_item("Responsibilities","General Ledger, Vision Operations");
nca_list_activate_item("FNDSCSGN.NAVIGATOR.LIST.0","+ Journals");
nca_list_activate_item("FNDSCSGN.NAVIGATOR.LIST.0","       Enter");
nca_button_press("GLXJEENT.TOOLBAR.LIST.0");
nca_lov_find_value("Batches","");
nca_lov_retrieve_items("Batches",1,9);
nca_lov_select_item("Batches","AR 1020 Receivables 2537: A 1020");
nca_edit_click("GLXJEENT.FOLDER_QF.BATCH_NAME.0");
...
```

Oracle Configurator アプリケーションを対象に実行するテストなど特定のテストでは，ある関数によって返される情報がセッション全体で必要になります。VuGen は，スクリプトに **web_reg_save_param** 関数を挿入することによって，動的な情報を自動的にパラメータに保存します。次の例では，接続情報が NCAJServSessionID という名前のパラメータに保存されています。

```
web_reg_save_param ("NCAJServSessionId", "LB=¥r¥n¥r¥n", "RB=¥r",
        LAST);
web_url("f60servlet",
        "URL=http://usscifforms05.sfb.na/servlet/f60servlet¥?config
        =mult", LAST);
```

前述の例では，右の境界は ¥r です。実際の右の境界は，システムによって異なる場合があります。

Oracle NCA アプリケーションのテスト

次の項では，セキュア Oracle NCA アプリケーションおよびサーブレットをテストするためのヒントをいくつか取り上げます。

セキュア Oracle NCA アプリケーションのテスト

- 記録するプロトコルを選択するとき，プロトコル・リストから **Oracle NCA** だけ選択すればよく，**Web プロトコル**を選択する必要はありません。VuGen は，内部でセキュリティ情報を記録するため，明示的な Web 関数は不要です。
- ［ポートの割り当て］記録オプションで，ポート 443 の既存のエントリを削除して，Oracle サーバ名の新しいエントリを作成します。

  ［**サービス ID**］：HTTP
  ［**対象サーバ**］：Oracle Forms サーバの IP アドレスまたはロング・ホスト名
  ［**ポート**］：443
  ［**接続の種類**］：SSL［**SSL バージョン**］：使用している SSL のバージョン。不明な場合は「SSL 2/3」を選択します。

  詳細については，『**第 1 巻 ー VuGen の使用**』の「ポートの割り当て設定」を参照してください。
- **nca_connect_server** コマンド実行中に NCA HTTPS スクリプトを再生するときに問題が生じた場合は，スクリプトの先頭に次の関数を挿入します。

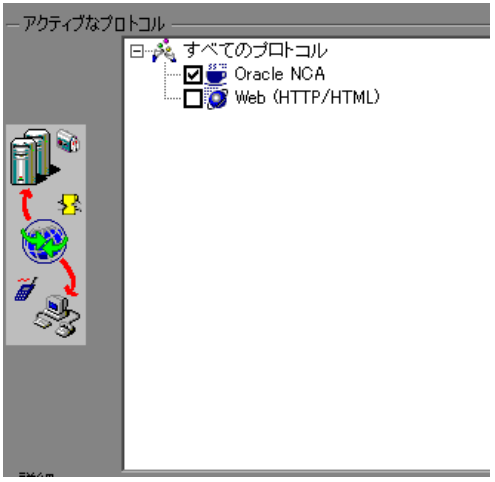
```
web_set_sockets_option("SSL_VERSION","3");
```

サーブレットおよびその他の Oracle NCA アプリケーションのテスト

NCA セッションの中には，サーブレットを使用するものがあります。

- Forms Listener サーブレット
- NCA および HTTP 通信の両方を使用するアプリケーションまたはモジュール（Oracle Configurator など）
- NCA アプリケーションの初期化（アプレット，jar ファイル，gif ファイルのダウンロード）

サーブレットを記録するときは，Oracle NCA 関数と Web 関数の両方を記録する必要があります。そのためには，最初にマルチ・プロトコル・スクリプトを作成します。また，Oracle NCA を対象としたシングル・プロトコル・スクリプトを作成した場合は，［記録オプション］で［**一般：プロトコル**］ノードを開き，Web プロトコルを有効にします。これで，記録が開始できます。



アプリケーションでサーブレットが使用されているかどうか不確かな場合は，script ディレクトリの **default.cfg** ファイルを確認します。次のエントリを探します。

UseServletMode=

値が 1 または 2 ならば，サーブレットが使用されています。Oracle NCA に加えて HTTP の記録も有効にする必要があります。

すでにスクリプトが記録されている場合は，Web 関数を含めるようにコードを自動的に再生成できます。再度記録をする必要はありません。［**ツール**］＞［**スクリプトの再生成**］を選択し，［プロトコル］セクションで Web プロトコルを選択します。

記録モードの指定

Oracle NCA スクリプトを記録するとき，VuGen は自動的に正しい接続モード，つまり HTTP モードかソケット・モードかを判断します。通常は VuGen によってシステムの構成が自動的に検出されるため，記録の設定を変更する必要はありません。標準のポート割り当てがほかのアプリケーションによって予約されているシステムの場合，記録モードに応じて［ポートの割り当て］の設定を変更しなければならないことがあります。

記録モードは，次のいずれかの方法で判断できます。

➤ NCA アプリケーションを使用しているときに，Java コンソールを開きます。

```
proxyHost=null
proxyPort=0
connectMode=HTTP
Forms Applet version is : 60812
```

connectMode エントリに，**HTTP**，**HTTPS**，または **socket** が表示されます。

➤ NCA セッションの記録後に，Vuser ディレクトリの **default.cfg** ファイルを開き，**UseHttpConnectMode** エントリの値を確認します。

```
[HttpConnectMode]
UseHttpConnectMode= 2
// 0 = socket 1 = http 2 = https
```

［サーバ エントリ］ダイアログ・ボックスで新しいポート割り当てを定義する場合，HTTP または HTTPS モードのときは［**サービス ID**］として「**HTTP**」を選択します。ソケット・モードのときは，［**サービス ID**］として「NCA」を選択します。

ポート割り当ての設定の詳細については，『**第 1 巻 ― VuGen の使用**』の「ポートの割り当て設定」を参照してください。

Oracle DB の追跡情報の記録

スクリプトをデバッグするには，Oracle DB ブレークダウン・グラフを使用できます。このグラフのデータを収集するには，スクリプトを実行する前に追跡メカニズムを有効にします。

追跡メカニズムを手動で有効にするには，**nca_set_custom_dbtrace** 関数を使用します。詳細については，『**Online Function Reference**』（英語版）（［**ヘルプ**］＞［**関数リファレンス**］）を参照してください。

ロード・バランシングに向けた Oracle NCA ステートメントの相関

VuGen は，複数のアプリケーション・サーバを対象とするロード・バランシングをサポートしています。HTTP の戻り値を **nca_connect_server** パラメータと相関させます。以降，Vuser はテスト実行時に，対応するサーバに接続してロード・バランシングを適用します。

ロード・バランシングに向けてステートメントを相関させるには，次の手順を実行します。

1 マルチ・プロトコル・スクリプトを記録します。

Oracle NCA および Web プロトコルのマルチ・プロトコル・スクリプトを記録します。必要なアクションを実行し，スクリプトを保存します。

2 ホストのパラメータと引数を定義します。

パラメータ化用に 2 つの変数 **serverHost** および **serverArgs** を定義します。

```
web_set_max_html_param_len("512");
web_reg_save_param("serverHost", "NOTFOUND=ERROR",
    "LB=<PARAM name=¥"serverHost¥" value=¥"","RB=¥">", LAST);
web_reg_save_param("serverArgs", "NOTFOUND=ERROR",
    "LB=<PARAM name=¥"serverArgs¥" value=¥"","RB=¥">", LAST);
```

3 値を serverHost および serverArgs に割り当てます。

```
web_url("step_name", "URL=http://server1.acme.com/test.htm", LAST);
```

4 次の **nca_connect_server** ステートメントを変更します。

```
nca_connect_server("199.203.78.170",
                   9000"/*version=107*/,
"module=e:¥¥appsnca...fndnam=apps ");
```

を次のように変更します。

```
    nca_connect_server("{ serverHost }", "9000"/*version=107*/,
"{serverArgs}");
```

スクリプトは次のようになるはずです。

```
web_set_max_html_param_len("512");
web_reg_save_param("serverHost", "NOTFOUND=ERROR",
    "LB=<PARAM name=¥"serverHost¥" value=¥"","RB=¥">", LAST);
web_reg_save_param("serverArgs", "NOTFOUND=ERROR",
    "LB=<PARAM name=¥"serverArgs¥" value=¥"","RB=¥">", LAST);
    web_url("step_name", "URL=http://server1.acme/test.htm", LAST);
nca_connect_server("{serverHost}","9000"/*version=107*/,"{serverArgs}");
```

その他に推奨される相関

Oracle NCA セッションを記録するとき，動的な値，つまり記録セッションおよび再生セッションごとに変化する値が VuGen によって記録されます。よく使用される動的な 2 つの引数が，icx_ticket と JServSessionIdroot です。

icx_ticket

icx_ticket 変数は，**web_url** 関数および **nca_connect_server** 関数で送信する情報の一部です。

```
web_url("fnd_icx_launch.runforms",
"URL=http://ABC-
123:8002/pls/VIS/fnd_icx_launch.runforms¥?ICX_TICKET=5843A55058947ED3&RES
P_APP=AR&RESP_KEY=RECEIVABLES_MANAGER&SECGRP_KEY=STANDARD",
LAST);
```

この **icx_ticket** の値は記録ごとに異なります。この変数には，クライアントによって送信されたクッキー情報が格納されます。記録を相関させるには，記録された **icx_ticket** 値の最初の出現の前に **web_reg_save_param** を追加します。

```
web_reg_save_param("icx_ticket", "LB=TICKET=", "RB=&RES", LAST);

...

web_url("fnd_icx_launch.runforms",
"URL=http://ABC-
123:8002/pls/VIS/fnd_icx_launch.runforms¥?ICX_TICKET={icx_ticket}&RESP_APP=
AR&RESP_KEY=RECEIVABLES_MANAGER&SECGRP_KEY=STANDARD", LAST);
```

注：web_reg_save_param の左右の境界は，アプリケーションの設定によって異なる場合があります。

JServSessionIdroot

JServSessionIdroot 値は，セッション ID を格納するためにアプリケーションによって設定されるクッキーです。ほとんどの場合，この値は VuGen によって自動的に相関され，**web_reg_save_param** 関数が挿入されます。この関数が自動的に追加されなかった場合は，手作業で追加し，値をすべてパラメータ名で置き換えます。

相関させる必要がある値を特定するには，実行ログを開き（[**表示**] > [**出力ウィンドウ**]），応答の本体を探します。

```
vuser_init.c(8):    Set-Cookie: JServSessionIdroot=my1sanw2n1.JS4; path=/¥r¥n
vuser_init.c(8):    Content-Length: 79¥r¥n
vuser_init.c(8):    Content-Type: text/plain¥r¥n
vuser_init.c(8):    ¥r¥n
vuser_init.c(8): 81-byte response body for "http://ABC-
123/servlet/oracle.forms.servlet.ListenerServlet?ifcmd=getinfo&ifhost=mercury&ifip=123.45
.789.12" (RelFrameId=1)
vuser_init.c(8):
/servlet/oracle.forms.servlet.ListenerServlet?JServSessionIdroot=my1sanw2n1.JS4¥r¥n
```

この動的な値を相関させるには，最初の出現の前に **web_reg_save_param** 関数を挿入し，スクリプト全体にわたって変数値をパラメータ名で置き換えます。この例では，左右の境界は ¥r と ¥n ですが，使用する環境での正確な境界を知るために，個別の環境を確認する必要があります。

```
web_reg_save_param("NCAJServSessionId","LB=¥r¥n¥r¥n","RB=¥r","ORD=1",LAST);

web_url("f60servlet",
    "URL= http://ABC-"123/servlet/oracle.forms.servlet.ListenerServlet?ifcmd=getinfo&"
"ifhost=mercury&ifip=123.45.789.12", LAST);

web_url("oracle.forms.servlet.ListenerSer",
    "URL=http://ABC-123{NCAJServSessionId}?ifcmd=getinfo&"
"ifhost=mercury&ifip=123.45.789.12", LAST);
```

プラグマ・モードでの記録

Oracle NCA Vuser のクライアント側では，サーバに対して **Pragma** という名前の追加ヘッダを送信するように設定できます。このヘッダは，次のように振る舞うカウンタです。NCA ハンドシェイクの最初のメッセージは 1 という値を持っています。

ハンドシェイクに続くメッセージは，3 からカウントが始まります。カウンタの値は，クライアントによって送信されるメッセージごとに 1 つ増加します。

サーバから受信したメッセージの種類が plain¥text で，メッセージの本体が ifError:#/#00 で始まる場合，クライアントはサーバに 0 バイトのメッセージを送信し，Pragma 値はマイナスに変更されます。クライアントがサーバからの情報の受信に成功すると，マイナス記号は元に戻ります。

Pragma ヘッダの記録は，マルチ・プロトコル・モード（Oracle NCA および Web）だけでサポートされます。プラグマ・モードは，スクリプトの default.cfg ファイル内で特定できます。プラグマ・モードで操作すると，UseServletMode は 2 に設定されます。

```
[HttpConnectMode]
UseHttpConnectMode=1
RelativeURL= < NCAJServSessionId >
UseServletMode=2
```

プラグマに関係する実行環境の設定の詳細については，『**第 1 巻 ー VuGen の使用**』の第 24 章「選択したプロトコルの実行環境の設定」を参照してください。

プラグマ・モードかどうかを知るには，WinSock レベルの記録を実行し，バッファの内容を確認します。最初の例では，バッファにカウンタとして Pragma 値が含まれています。

```
send buf108
    "POST /ss2servlet/oracle.forms.servlet.ListenerServlet?JServSessionIdss2ser"
    "vlet=gk5q79uqy1 HTTP/1.1¥r¥n"
    "Pragma: 1¥r¥n"
    ...
send buf110
    "POST /ss2servlet/oracle.forms.servlet.ListenerServlet?JServSessionIdss2ser"
    "vlet=gk5q79uqy1 HTTP/1.1¥r¥n"
    "Pragma: 3¥r¥n"
    ...
```

次の例では，バッファにエラー・インジケータとして Pragma 値が含まれています。

```
recv buf129 281
    "HTTP/1.1 200 OK¥r¥n"
    "Date: Tue, 21 May 2002 00:03:48 GMT¥r¥n"
    "Server: Oracle HTTP Server Powered by Apache/1.3.19 (Unix) mod_fastcgi/2.2"
    ".10 mod_perl/1.25 mod_oprocmgr/1.0¥r¥n"
    "Content-Length: 13¥r¥n"
    "Content-Type: text/plain¥r¥n"
    "¥r¥n"
    "ifError:8/100"

send buf130
    "POST /ss2servlet/oracle.forms.servlet.ListenerServlet?JServSessionIdss2ser"
    "vlet=gk5q79uqy1 HTTP/1.1¥r¥n"
    "Pragma: -12¥r¥n"
    ...
```

第 46 章

SAPGUI プロトコル

成長を続けている ERP（Enterprise Resource Planning）の分野において，SAP は企業が自社のすべてのビジネス・プロセスを管理できるようにするソリューションを提供しています。HP は，SAP ソリューションのモジュールを機能テスト・レベルと負荷テスト・レベルの両方でテストするためのツールを提供しています。本章では，SAPGUI for Windows クライアント（SAP GUI）をテストするためのソリューションについて説明します。mySAP Workplace および Portal クライアントのためのソリューションをテストする方法の詳細については，第 49 章「SAP-Web プロトコル」を参照してください。

本章の内容

SAPGUI Vuser スクリプトの作成について

本章では，SAPGUI for Windows クライアント（SAP GUI）をテストするためのソリューションについて説明します。クライアント上でのみ運用する SAPGUI ユーザをテストするには，SAPGUI Vuser タイプを使用します。Web ブラウザも使用する SAPGUI ユーザをテストするには，SAP（Click and Script）プロトコルを使用します。

セッションを記録する前に，モジュールとクライアント・インタフェースが VuGen によってサポートされていることを確認します。次の表に，SAP ビジネス・アプリケーションおよび関連ツールの SAP のクライアント・モジュールを示します。

SAP モジュール	VuGen サポート
SAP Web クライアントまたは mySAP.com	SAP-Web Vuser タイプを使用します。
SAPGUI for Windows	Windows ベースのクライアント。SAPGUI Vuser によってエミュレートします。また，APO モジュールの記録もサポートします（APO 3.0 のパッチ・レベル 24 が必要です）。
SAPGUI for Windows と Web ブラウザ	SAP（Click and Script）プロトコルを使用します。
SAPGUI for Java	このクライアントはサポートされていません。

バージョン 6.20 以降：

- **機能テストの場合**：mySAP.com クライアント用の QuickTest Professional アドインを使用します。
- **負荷テストの場合**：SAPGUI または SAP（Click and Script）プロトコルを使用して，VuGen にスクリプトを作成して，Controller でシナリオを実行します。

まず，VuGen で典型的なビジネス・プロセスを記録します。VuGen では，SAP ビジネス・プロセス中の SAPGUI for Windows クライアントのアクティビティを記録し，Vuser スクリプトを生成できます。SAPGUI for Windows クライアント内でアクションを実行すると，このアクティビティを説明する関数が生成されます。各関数の先頭には，**sapgui** というプレフィックスが付きます。

SAPGUI Vuser のための環境の確認

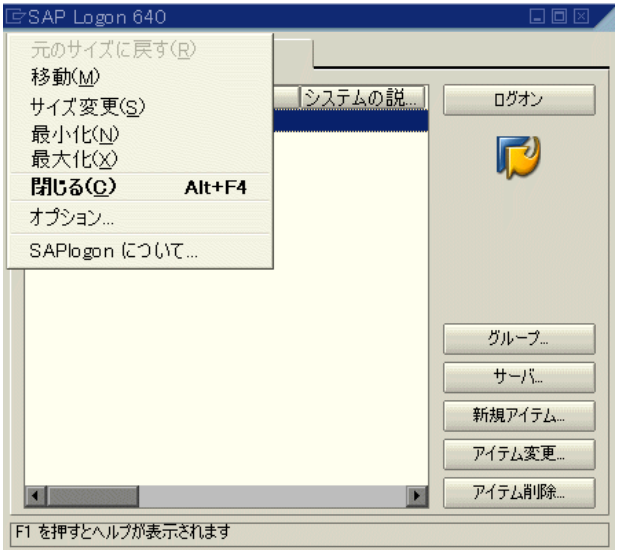
SAPGUI Vuser を記録するためのシステムの確認および設定の基本的な手順については，「パッチ・レベルの確認」および「スクリプティングの有効化」を参照してください。環境が適切に設定されれば，通常の SAP セッションを記録して VuGen で再生できます。

パッチ・レベルの確認

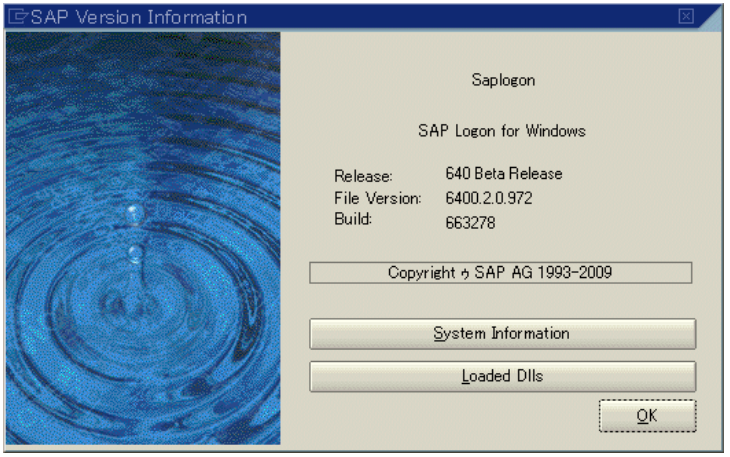
SAPGUI for Windows クライアントのパッチ・レベルは，バージョン情報ダイアログ・ボックスから確認できます。サポートされている最も低いパッチ・レベルは，バージョン 6.20 のパッチ 32 です。

パッチ・レベルを確認するには，次の手順を実行します。

1 SAPGUI ログオン・ウィンドウを起動します。［SAP Logon］ダイアログ・ボックスの左上隅をクリックし，メニューから［**SAPlogon について ...**］を選択します。



2［SAP Version Information］ダイアログ・ボックスが開きます。パッチ・レベルのエントリが 32 以上であることを確認します。



スクリプティングの有効化

VuGen による SAPGUI for Windows クライアントに対するサポート機能では，SAP の Scripting API を利用しています。この API により，Vuser は SAPGUI クライアントと対話したり，通知を受け取ったり，操作を実行したりできるようになります。

Scripting API が利用できるのは，SAP Kernel の最近のバージョンだけです。スクリプティングをサポートするバージョンのカーネルでは，オプションは標準で無効になっています。VuGen を使用するには，まず SAP サーバが Scripting API をサポートしていることを確認し，サーバとクライアントの両方で Scripting API を有効にする必要があります。詳細およびパッチのダウンロードについては，『SAP OSS note #480149』を参照してください。

VuGen には，システムでスクリプティングがサポートされているかどうかを確認するユーティリティが付属しています。ユーティリティの **VerifyScript.exe** は，DVD の **Additional Components¥SAP_Tools¥VerifySAPGUI** フォルダ内に収録されています。詳細については，このユーティリティに付属の **VerifyScripting.htm** ファイルを参照してください。

次の項では，スクリプティングを有効にする方法について説明します。

- 設定の確認
- SAP Application Server でのスクリプティングの有効化
- SAPGUI 6.20 Client でのスクリプティングの有効化

設定の確認

スクリプティングを有効にするには，まず正しいバージョンのカーネルがインストールされていることを確認し，必要に応じてアップデートします。

SAP Application Server のバージョン別に必要な最低限のカーネル・パッチ・レベルを次の表で確認します。必要に応じて，最新のパッチをダウンロードしてインストールします。

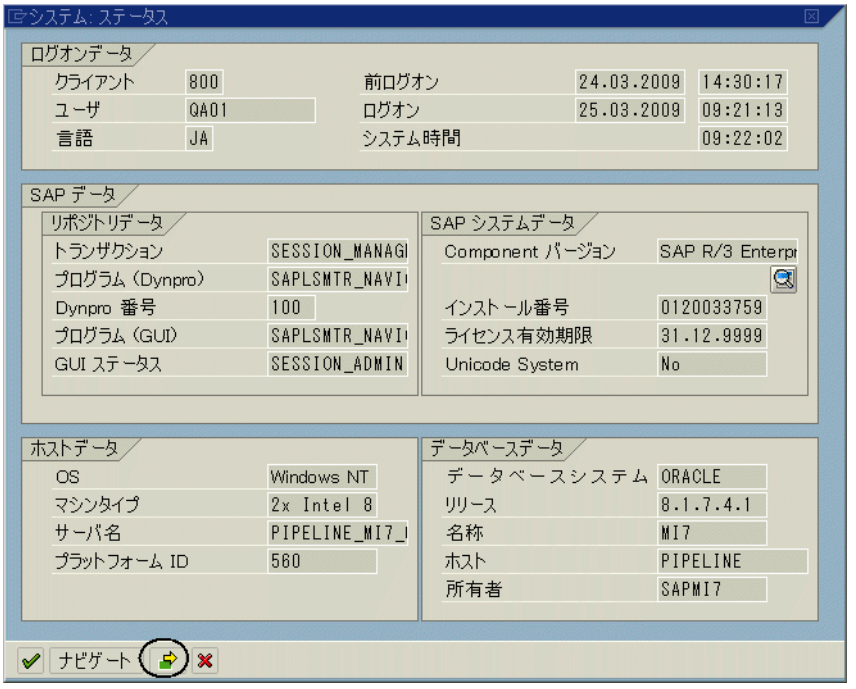
ソフトウェア・コンポーネント	リリース	パッケージ名	カーネル・パッチ・レベル
SAP_APPL	31I	SAPKH31I96	Kernel 3.1I レベル 650
SAP_APPL	40B	SAPKH40B71	Kernel 4.0B レベル 903
SAP_APPL	45B	SAPKH45B49	Kernel 4.5B レベル 753
SAP_BASIS	46B	SAPKB46B37	Kernel 4.6D レベル 948
SAP_BASIS	46C	SAPKB46C29	Kernel 4.6D レベル 948
SAP_BASIS	46D	SAPKB46D17	Kernel 4.6D レベル 948
SAP_BASIS	610	SAPKB61012	Kernel 6.10 レベル 360

カーネル・パッチ・レベルを確認するには，次の手順を実行します。

1 SAP システムにログインします。

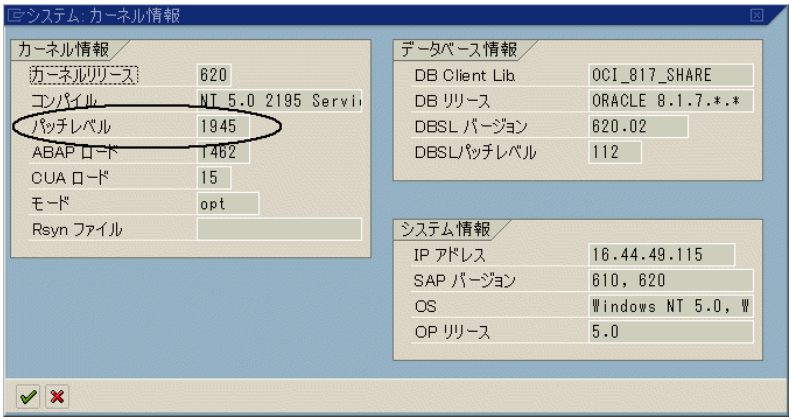
2 ［**システム**］ > ［**ステータス**］を選択します。

3 黄色い矢印の付いた［**他のカーネル情報**］ボタンをクリックします。



4［**カーネル情報**］セクションで，［**Sup. Pkg. lvl**］の値を確認します。

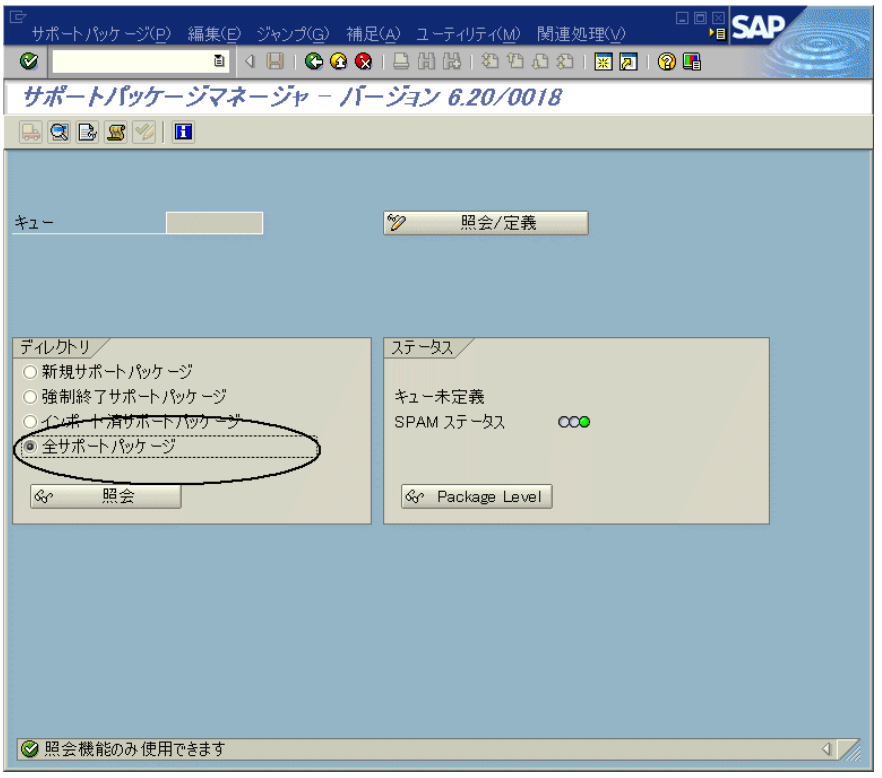
レベルが 948 より低い場合は，最新のバージョンのカーネルをダウンロードして，既存のカーネルをアップグレードする必要があります。このアップグレード方法の詳細については『SAP OSS note #480149』を参照してください。



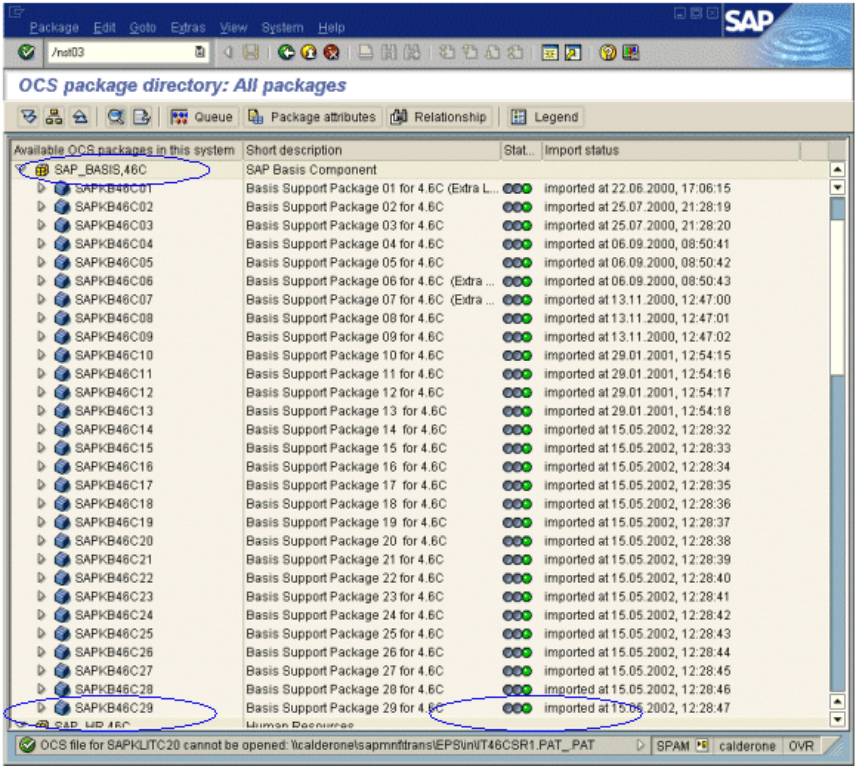
R/3 サポート・パッケージを確認するには，次の手順を実行します。

1 SAP システムにログインし，SPAM トランザクションを実行します。

2 ［**ディレクトリ**］セクションで，［**全サポートパッケージ**］を選択し，［**照会**］ボタンをクリックします。



3 SAP_BASIS, 4.6C に SAPKB46C29 がインストールされていることを確認します。インストールされていれば，［ステータス］カラムに緑色の丸が表示されます。



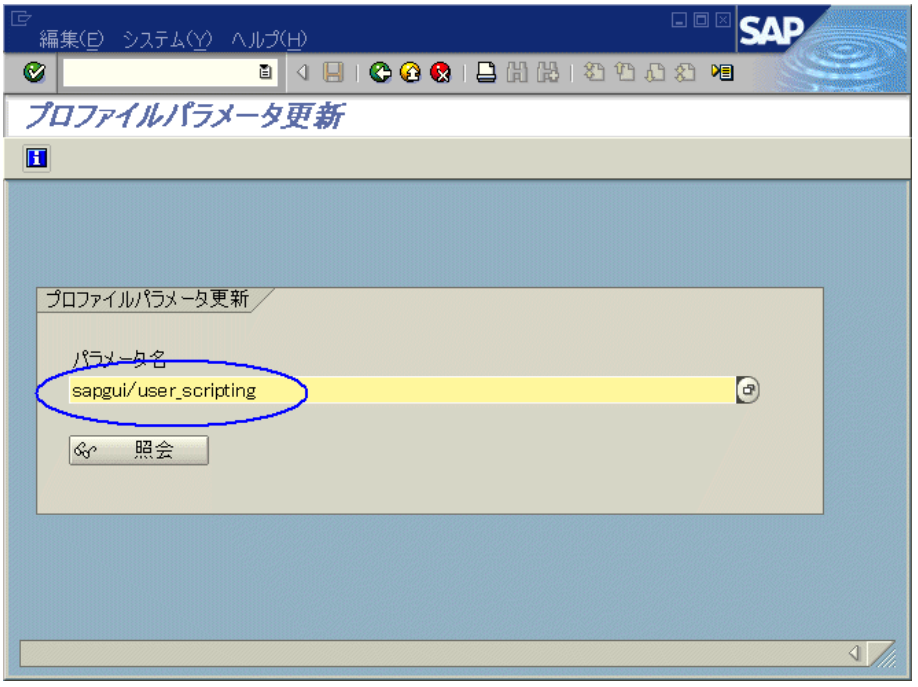
OCS パッケージがインストールされていない場合は，www.sap.com Web サイトからダウンロードしてインストールします。詳細については，『SAP OSS note ＃ 480149』を参照してください。

SAP Application Server でのスクリプティングの有効化

スクリプティングを有効にするには，管理者権限のあるユーザがアプリケーション・サーバで **sapgui/user_scripting** プロファイル・パラメータを **TRUE** に設定します。すべてのユーザに対してスクリプティングを有効にするには，すべてのアプリケーション・サーバでこのパラメータを設定します。特定のユーザ・グループに対してスクリプティングを有効にするには，必要なアクセス制限のかかったアプリケーション・サーバでパラメータを設定します。

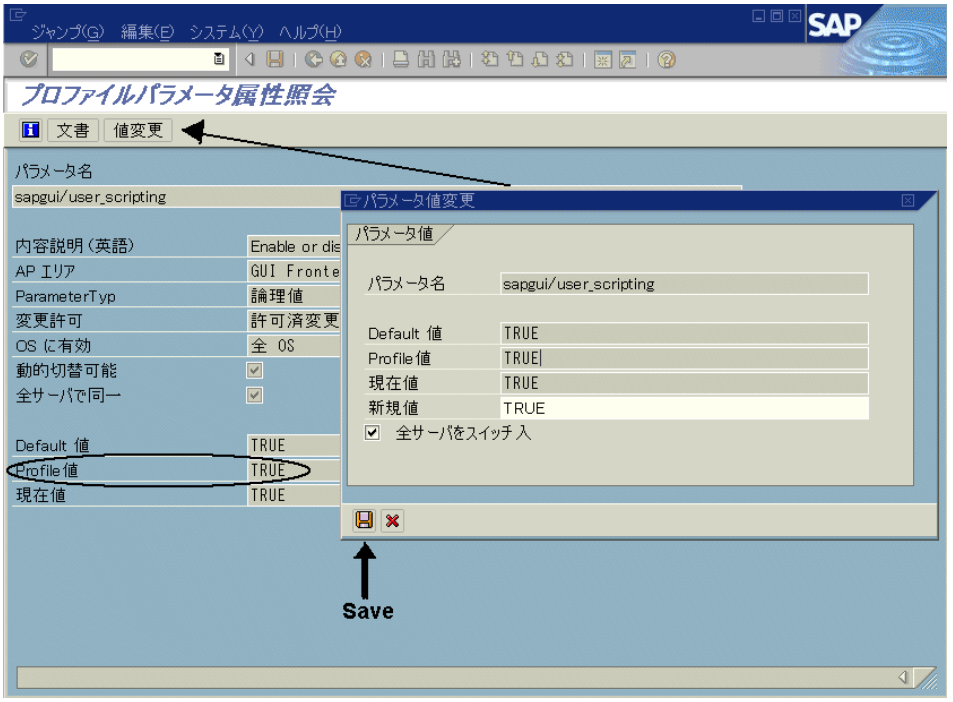
プロファイル・パラメータを変更するには，次の手順を実行します。

1 トランザクション **rz11** を開きます。パラメータ名 **sapgui/user_scripting** を指定し，［**照会**］ボタンをクリックします。［プロファイルパラメータ属性照会］ウィンドウが開きます。



ステータス・バーに「**パラメータ名が識別されません**」というメッセージが表示された場合は，最新の Support Package が見当たらないことを示しています。アプリケーション・サーバの SAP BASIS とカーネルのバージョンに対応する Support Package をインポートします。詳細については，701 ページ「設定の確認」を参照してください。

2 **Profile 値**が FALSE の場合は，値を変更する必要があります。ツールバーの［**値変更**］ボタンをクリックします。［パラメータ値変更］ウィンドウが開きます。［**Profile 値**］ボックスに TRUE と入力し，［**保存**］ボタンをクリックします。



変更を保存するとウィンドウが閉じ，**Profile 値**が TRUE に設定されます。

3 アプリケーション・サーバを再起動します。この変更はシステムにログオンしたときにのみ有効になります。

更新された **Profile 値**がサーバの再起動後にも変更されていない場合は，アプリケーション・サーバのカーネルが古くなっています。必要なカーネル・パッチをインポートします。詳細については，701 ページ「設定の確認」に記載されています。

Profile Value は，次のバージョンのカーネルでは，トランザクション rz11 を使用して動的に有効化できます。アプリケーション・サーバを再起動する必要はありません。

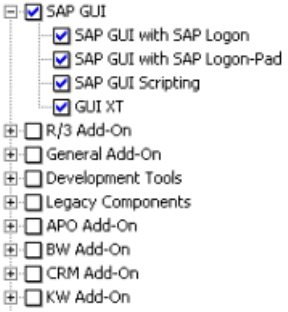
リリース	カーネル・バージョン	パッチ・レベル
4.6B，4.6C，4.6D	4.6D	972
6.10	6.10	391
6.20	すべてのバージョン	すべてのレベル

SAPGUI 6.20 Client でのスクリプティングの有効化

VuGen でスクリプトを実行できるようにするには，SAPGUI クライアントでもスクリプティングを有効にする必要があります。また，接続が確立されたときやスクリプトが GUI プロセスにアタッチされたときなどに表示される特定のメッセージが表示されないようにクライアントを設定する必要もあります。

VuGen で使用できるように SAPGUI クライアントを設定するには，次の手順を実行します。

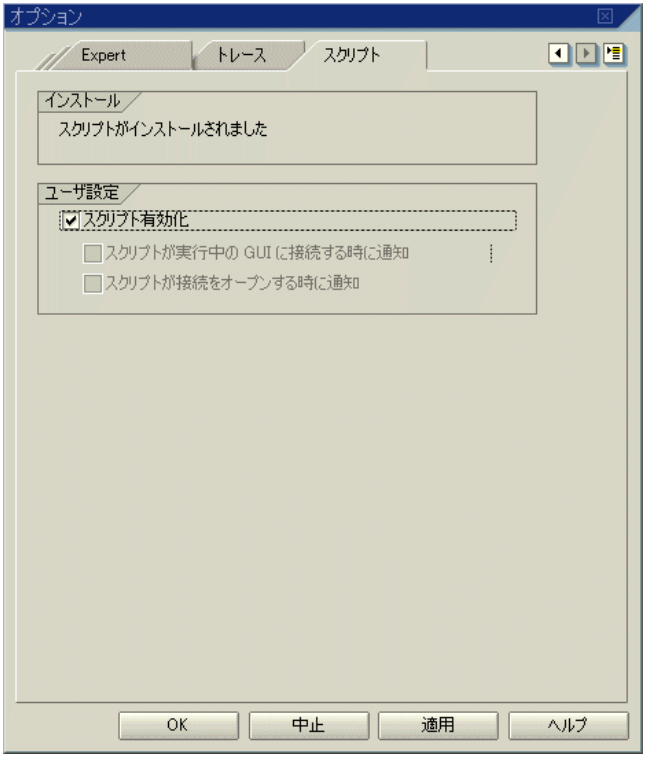
➤ **インストール中**：SAPGUI クライアントのインストール中に，［**SAP GUI Scripting**］オプションを有効にします。



➤ **インストール後**：警告メッセージが表示されないようにします。SAPGUI クライアントで［オプション］ダイアログ・ボックスを開きます。［**スクリプト**］タブを選択し，次のオプションをクリアします。

1 **［スクリプトが実行中の GUI に接続するときに通知］**

2 **［スクリプトが接続をオープンにする時に通知］**



また，次のレジストリ・キーの中で **WarnOnAttach** と **WarnOnConnection** の値を 0 に設定することによっても，これらのメッセージが表示されないようにできます。

HKCU¥SOFTWARE¥SAP¥SAPGUI Front¥SAP Frontend Server¥Security

SAPGUI Vuser スクリプトの作成

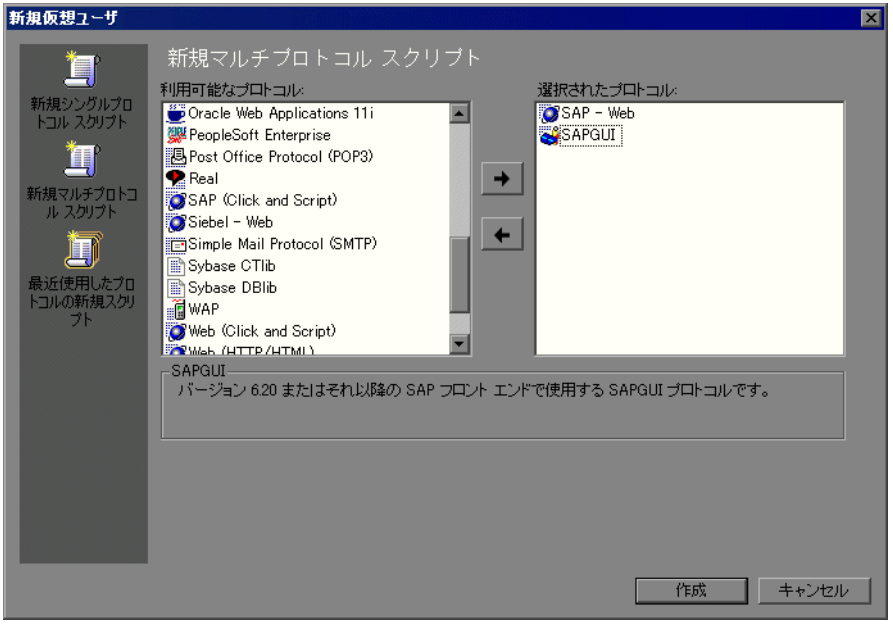
SAPGUI Vuser スクリプト作成の第一歩は，Vuser とスクリプトのタイプを選択することです。SAP の Vuser タイプ **SAPGUI** は，**ERP/CRM** カテゴリの下にあります。シングル・プロトコルとマルチ・プロトコルのどちらの Vuser スクリプトでも作成できます。

SAPGUI Vuser スクリプトを作成するには，次の手順を実行します。

1 VuGen を起動し，［**ファイル**］＞［**新規作成**］を選択します。

2 標準的な SAPGUI クライアント・セッション（ブラウザのコントロールなし）を記録するには，**SAPGUI** タイプの Vuser を使用して，シングル・プロトコルの Vuser スクリプトを作成します。

3 ブラウザのコントロールを使用する SAPGUI セッションを記録するには，マルチ・プロトコルの Vuser スクリプトを作成します。**SAPGUI** および **SAP-Web** の両方の Vuser タイプを指定します。これで，ブラウザのコントロールが存在するときに VuGen で Web 固有の機能を記録できるようになります。



4 ［**OK**］をクリックして Vuser スクリプトを開きます。

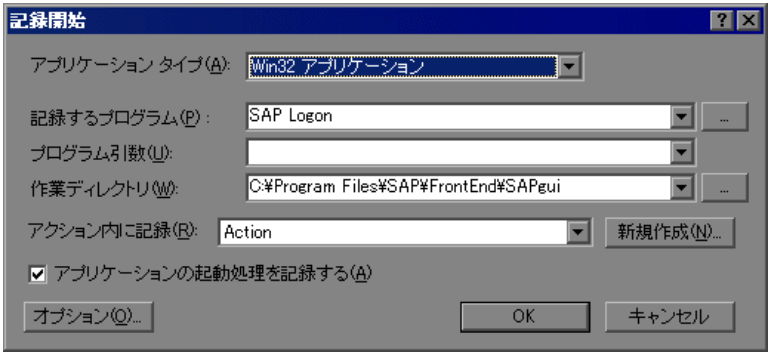
SAPGUI Vuser スクリプトの記録

空のスクリプトの作成後，記録オプションを設定し，SAPGUI セッションを記録します。VuGen はクライアント内のアクションに対応するスクリプトを生成します。

SAPGUI スクリプトを開始するには，次の手順を実行します。

1 ［記録開始］ダイアログ・ボックスが開かない場合は，［**記録開始**］ボタンをクリックします。［記録開始］ダイアログ・ボックスが開きます。

2 VuGen によって関連情報が検出され入力されます。



- ［**記録するプログラム**］: VuGen は SAP クライアントのインストール先から saplogon.exe ファイルを見つけます。
- ［**作業ディレクトリ**］: 作業ディレクトリを指定する必要があるアプリケーションの場合は，ここで指定します。必要となる情報は，Vuser スクリプトのタイプによって異なります。
- ［**アクション内に記録**］: 記録の挿入先セクションを選択します。最初に選択できるセクションは vuser_init，Action1，および vuser_end です。

3 ［**OK**］をクリックして記録を開始します。

カーソル位置での記録

VuGen では，既存のスクリプトにアクションを記録することもできます。いくつかの理由から，既存のスクリプトに記録する場合があります。

- 記録中に操作を誤った場合。
- 操作は正しくても，ポップアップ・ウィンドウの処理などの追加情報が必要な場合。たとえば，SAP サーバは記録セッション中に適用されなかったインベントリ警告を出すことがあります。

これは「カーソル位置での記録」と呼ばれる，新しいアクションの挿入または既存のアクションの置換を行えるようにする機能です。カーソルでの記録を開始すると，VuGen は次の 2 つのオプションの入力を求めるプロンプトを表示します。

- ［アクションにステップを挿入する］: 既存のステップを一切上書きせずに，新しく記録されたステップを挿入します。新しいセグメントは追加されたセクションの始まりと終わりを表すコメントで囲まれます。このオプションは，記録中には存在しなかった，ときおり現れるポップアップ・ウィンドウの処理に適しています。

```
// Recording at the cursor ー 開始
    sapgui_select_active_connection("con[0]");
    sapgui_select_active_session("ses[0]");
    sapgui_select_active_window("wnd[0]");
//Recording at the cursor ー 終了
```

- ［**スクリプトの残りを上書きする**］: カーソル位置から後のすべてのステップを置換します。このオプションは現在のアクションの残りの部分を上書きし，ほかのすべてのアクションを削除します。**vuser_init** セクションと **vuser_end** セクションは影響を受けません。

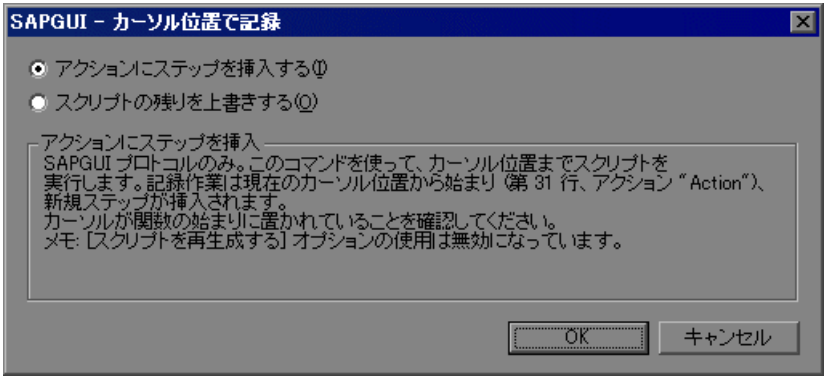
カーソルでの記録オプションのどちらかを選択すると，VuGen はスクリプトを先頭からカーソルの位置まで再生します。次に［記録］フローティング・ツールバーを表示して，記録を開始します。**カーソル位置での記録**機能を使う場合には，**スクリプトの再生成**ツールが無効になります。

注： カーソル位置で記録するには，既存の関数の直前で VuGen エディタの左余白をクリックする必要があります。

カーソル位置で記録するには，次の手順を実行します。

1 スクリプト・ビューを開き（[**表示**] > [**スクリプト ビュー**]），既存の関数の横の左余白をクリックします。

2 [**カーソル位置で記録**] ボタンをクリックします。選択を求めるプロンプトが表示されます。



3 [**アクションにステップを挿入する**] または [**スクリプトの残りを上書きする**] を選択します。[**OK**] をクリックします。VuGen はスクリプトをカーソルの位置まで再生します。

4 [記録] フローティング・ツールバーが開くのを待ちます。SAPGUI クライアント内でアクションを開始し，必要に応じてセクションとアクションを切り替えます。

5 記録セッションを終了するには，[**停止**] ボタンをクリックします。

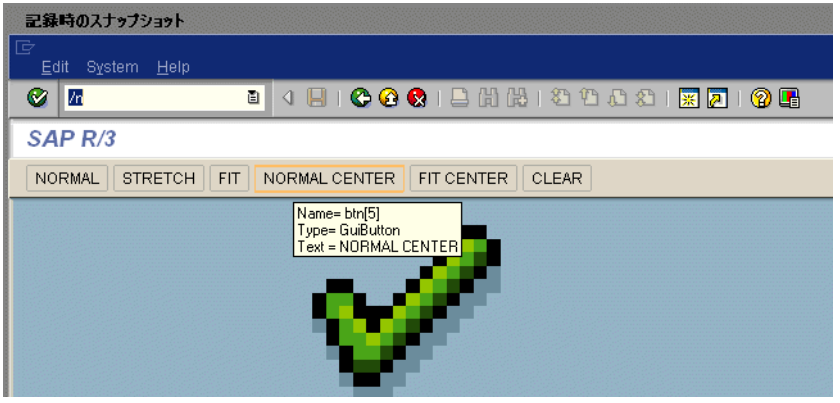
SAPGUI スクリプトのステップの対話的挿入

記録後，スクリプト・ビューかツリー・ビューのどちらかで，手作業でステップをスクリプトに追加できます。異なるビューからステップを追加する方法の詳細については，『**第 1 巻 ー VuGen の使用**』の「VuGen の紹介」を参照してください。

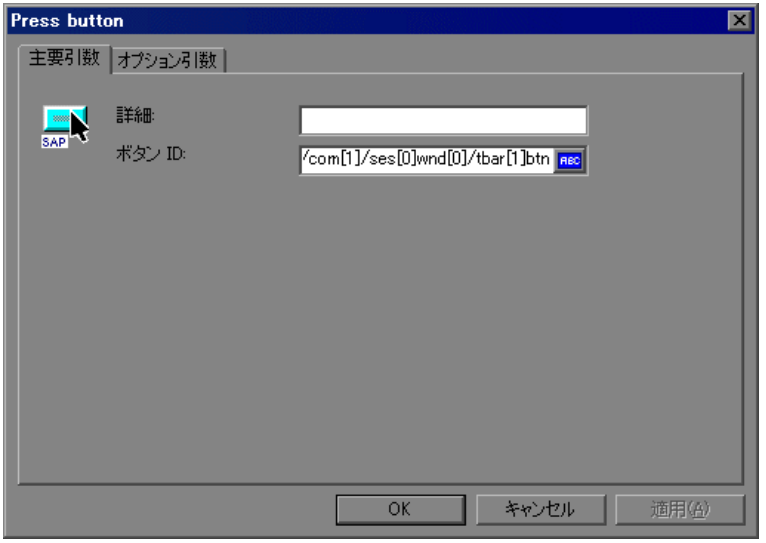
新しい関数を手作業で追加するだけでなく，Citrix Vuser のために，新しいステップをスナップショットから直接，対話形式で追加できます。右クリック・メニューを使用して，ビットマップ関連またはテキスト関連のステップを追加できます。

スナップショットの中からステップを追加する場合には，VuGen は Active Screen 機能を使い，SAPGUI クライアント・ウィンドウ内の各オブジェクトの ID を調べます（SAPGUI 一般記録オプションの中で Active Screen スナップショットを無効にしていない場合）。

VuGen によってどのオブジェクトが認識されているかを調べるには，スナップショット上でマウスを動かします。VuGen はオブジェクトの周囲にボックスを描画し，オブジェクトの Control ID を持つツール・チップを表示します。次の例では，選択されたアクティブ・オブジェクトは NORMAL CENTER ボタンです。



認識されたオブジェクト上にマウスがある間にステップを追加すると，VuGen は自動的にそのオブジェクトの Control ID を［Properties］ダイアログ・ボックスの関連フィールドに挿入します。たとえば，上に示すように，NORMAL CENTER ボタンのために **Press Button** ステップを挿入した場合，［プロパティ］ダイアログ・ボックスには次の ID が表示されます。



特定のオブジェクトにステップを対話的に挿入するには，次の手順を実行します。

1 ［バッファのスナップショット］ウィンドウ内をクリックします。

2 関数を追加するオブジェクト上にマウスを移動します。VuGen がそのオブジェクトを認識し，ボックスで囲んでいることを確認します。

3 右クリックで表示されるメニューから［**新規ステップの挿入**］を選択します。［ステップの追加］ボックスが開きます。

4 メニューからステップを選択します。ステップの「プロパティ」ダイアログ・ボックスが開き，関連するオブジェクトの Control ID が表示されます。

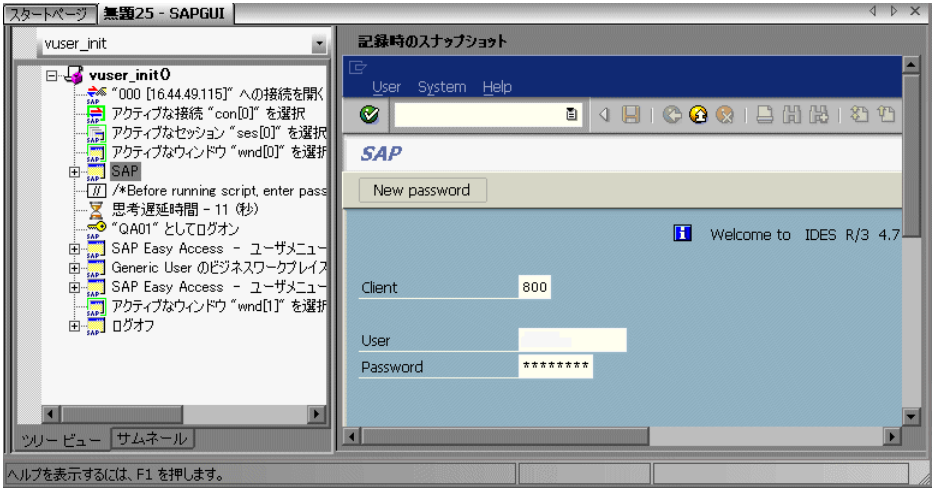
5 ［**詳細**］ボックスにオブジェクトの名前を入力します。［**OK**］をクリックします。選択したステップの後に新しいステップが挿入されます。

6 特定の場所に貼り付けるオブジェクトの Control ID を取得するには，右クリック・メニューから［**コントロール ID のコピー**］を選択します。Control ID がクリップボードに置かれます。［スクリプト］ビューから［プロパティ］ボックスまたは直接コードに貼り付けることができます。

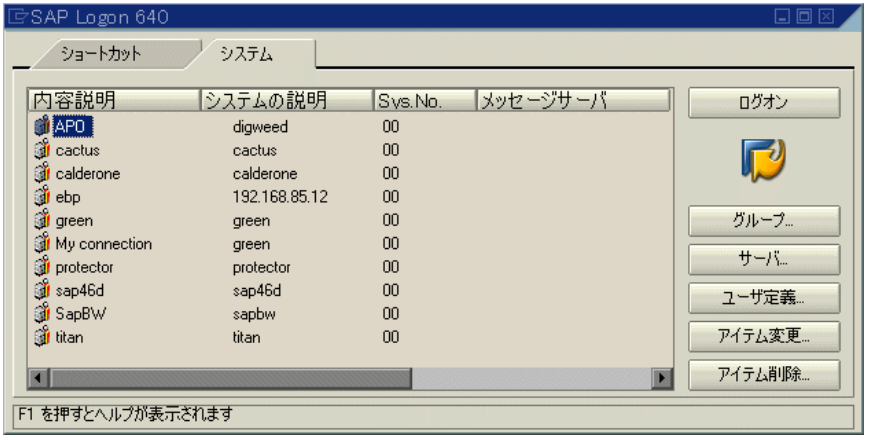
SAPGUI Vuser スクリプトについて

通常 SAPGUI Vuser スクリプトには，ビジネス・プロセスを構成する複数の SAP トランザクションが含まれています。ビジネス・プロセスは，ユーザ・アクションをエミュレートする関数で構成されます。ツリー・ビューを開くと，各ユーザ・アクションが Vuser スクリプトのステップとして表示されます。

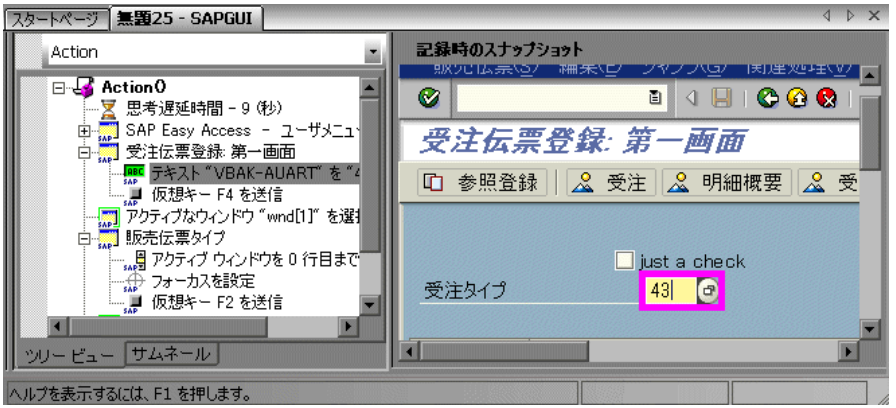
次の例は，SAPGUI クライアントの典型的な記録を示しています。最初のセクションである **vuser_init** には，接続の開始とログインが含まれます。



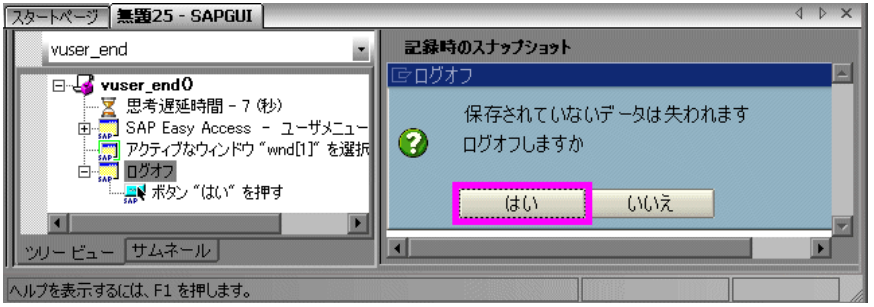
［接続を開く］ステップは，［SAP Logon］の［**詳細**］リストにある接続名の 1 つを使用します。指定された名前がリストにない場合，Vuser はその名前のサーバを検索します。



次のセクションでは，関数によって，メニューの選択やチェック・ボックスの設定など，一般的なユーザ操作がエミュレートされます。



最後のセクション **vuser_end** は，ログオフ手順を示しています。



SAPGUI と Web の両方に対してマルチ・プロトコルのスクリプトを記録しているときは，両方のプロトコルに対するステップが VuGen によって生成されます。スクリプト・ビューでは，**sapgui** と **web** の両方の関数を表示できます。

次の例は，SAPGUI クライアントによって Web コントロールが開かれるマルチ・プロトコル記録を示しています。**sapgui** 関数から **web** 関数に切り替わることに注意してください。

```
sapgui_tree_double_click_item("Use as general WWW browser, REPTITLE",
    "shellcont/shell",
    "000732",
    "REPTITLE",
    BEGIN_OPTIONAL,
        "AdditionalInfo=sapgui1020",
    END_OPTIONAL);

...
sapgui_set_text("",
    "http:¥¥¥¥yahoo.com",
    "usr/txtEDURL",
    BEGIN_OPTIONAL,
        "AdditionalInfo=sapgui1021",
    END_OPTIONAL);

...
web_add_cookie("B=7pt5cisv1p3m2&b=2; DOMAIN=www.yahoo.com");

web_url("yahoo.com",
        "URL=http://yahoo.com/",
        "Resource=0",
        "RecContentType=text/html",
        "Referer=",
        "Snapshot=t1.inf",
        "Mode=HTML",
        EXTRARES,
"URL=http://srd.yahoo.com/hpt1/ni=17/ct=lan/sss=1043752588/t1=1043752575385/d1
=1251/d2=1312/d3=1642/d4=4757/0.4097009487287739/*1",
"Referer=http://www.yahoo.com/", ENDITEM,
        LAST);
```

SAPGUI Vuser スクリプトの拡張

記録された Vuser スクリプトを確認し終わったら，次の方法でそれを拡張します。

- **トランザクション**：トランザクション，ランデブー・ポイント，および制御フロー構造を，スクリプトに挿入します。詳細については，『**第 1 巻 － VuGen の使用**』の「Vuser スクリプトの拡張」を参照してください。
- **検証**：SAPGUI の検証関数を挿入し，SAPGUI オブジェクトの現在のステータスを検証します。詳細については，「検証関数の追加」を参照してください。
- **情報の取得**：SAPGUI の関数を挿入し，SAPGUI オブジェクトの現在の値を検証します。情報は **sapgui_get_xxx** 関数を使用して取得します。詳細については，720 ページ「情報の取得」を参照してください。

パラメータの定義（任意）：Vuser スクリプトに記録された固定値に対してパラメータを定義します。固定値をパラメータで置き換えることにより，同じビジネス・プロセスを，異なる値で何度も繰り返すことができます。詳細については，『**第 1 巻 － VuGen の使用**』の「パラメータの作成」を参照してください。

検証関数の追加

オプションの，または動的なウィンドウやフレームで作業をしているときに，検証関数を使用して，ウィンドウやオブジェクトが使用可能かどうかを調べることができます。オプションのウィンドウは，SAP セッション中に常に表示されるとは限らないウィンドウです。この関数により，オプションのウィンドウが開いたり例外が発生したりした場合でも，Vuser スクリプトの実行を続けることができます。

最初の例では，ウィンドウが使用可能かどうかを確認しています。ウィンドウが使用可能な場合は，Vuser へ実行を継続する前にそのウィンドウを閉じます。

```
if (!sapgui_is_object_available("wnd[1]"))
        sapgui_call_method("{ButtonID}",
            "press",
            LAST,
            AdditionalInfo=info1011");
sapgui_press_button(.....)
```

次の例は，ME51N トランザクション内の動的なオブジェクトを示しています。［Document overview］フレームはオプションであり，［**Document overview on/off**］ボタンによって開いたり閉じたりできます。

コードでは［Document overview］ボタンのテキストを調べています。ボタンのテキストが Document overview on であれば，そのボタンをクリックして［Document overview］フレームを閉じます。

```
if(sapgui_is_object_available("tbar[1]/btn[9]"))
{
    sapgui_get_text("Document overview on/off button",
        "tbar[1]/btn[9]",
        "paramButtonText",
        LAST);

    if(0 == strcmp("Document overview off", lr_eval_string("{paramButtonText}")))
        sapgui_press_button("Document overview off",
            "tbar[1]/btn[9]",
            BEGIN_OPTIONAL,
                "AdditionalInfo=sapgui1013",
            END_OPTIONAL);
}
```

情報の取得

SAPGUI Vuser で作業しているときに，**sapgui_get_** ＜ **xxx** ＞関数を使用して SAPGUI オブジェクトの現在の値を取得できます。この値は，別のビジネス・プロセスの入力として使用したり，出力ログに表示したりできます。

ステータス・バー情報の取得

次の例では，ステータス・バー・メッセージの一部を保存して注文番号を取得する方法を示します。

ステータス・バーから注文番号を取得するには，次の手順を実行します。

1 ステータス・バー・テキストを確認する位置に移動して，［**挿入**］＞［**新規ステップ**］を選択します。**sapgui_status_bar_get_type** 関数を選択します。この関数は，Vuser がステータス・バーからテキストを正常に取得できるかどうかを確かめます。

2 前のステートメントが正常に実行されたかどうかを確かめる **if** ステートメントを挿入します。正常に実行された場合は，**sapgui_status_bar_get_param** を使用して引数の値を保存します。

この sapgui_status_bar_get_param 関数は，注文番号をユーザ定義のパラメータに保存します。ここでは，注文番号はステータス・バー文字列の 2 番目のインデックスです。

```
sapgui_press_button("Save (Ctrl+S)",
    "tbar[0]/btn[11]",
    BEGIN_OPTIONAL,
        "AdditionalInfo=sapgui1038",
    END_OPTIONAL);

sapgui_status_bar_get_type("Status");
if(0==strcmp(lr_eval_string("{Status}"),"Success"))
  sapgui_status_bar_get_param("2", " Order_Number ");
```

テストの実行中，Execution ログには次のように値とパラメータ名が示されます。

```
Action.c(240): Pressed button " Save (Ctrl+S)"
Action.c(248): The type of the status bar is "Success"
Action.c(251): The value of parameter 2 in the status bar is "33232"
```

日付情報の保存

日付を使用するスクリプトを作成すると，正しく動作しないことがあります。たとえば，スクリプトを 6 月 2 日に記録し，それを 6 月 3 日に再生した場合，日付フィールドが正しくなりません。そのため，テキスト実行中に日付をパラメータに保存し，保存した値をほかの日付フィールドへの入力として使用する必要があります。スクリプト実行中の現在の日付または時刻を保存するには，**lr_save_datetime** 関数を使用します。この関数を，日付情報を必要とする関数の前に挿入します。日付の形式はロケールに固有です。**lr_save_datetime** 関数の中ではロケールに応じた形式を使用します。たとえば，＜月＞.＜日＞.＜年＞の形式にする場合は，「%m.%d.%Y」と指定します。

次の例では，lr_save_datetime で現在の日付を保存します。この値を **sapgui_set_text** 関数で使い，2 日後の配送日を設定します。

```
lr_save_datetime("%d.%m.%Y", DATE_NOW + (2 * ONE_DAY),
    "paramDateTodayPlus2");
sapgui_set_text("Req. deliv.date",
        "{paramDateTodayPlus2}","usr/ctxtRV45A-KETDAT",
        BEGIN_OPTIONAL,
            "AdditionalInfo=sapgui1025",
        END_OPTIONAL);
```

第 47 章

SAPGUI ー スクリプトの再生

記録および手作業による機能強化を通して SAPGUI スクリプトを作成したら，VuGen の中で再生してその機能をテストします。

本章の内容

次の情報は **SAPGUI** および **SAP**（**Click and Script**）のプロトコルに適用されます。

SAPGUI Vuser スクリプトの再生について

本章では，SAPGUI Vuser スクリプトの実行について説明します。実行環境を設定して，テストまたは監視セッションの間の Vuser の振る舞いを制御できます。

また本章には，SAPGUI Vuser を使用した作業のためのいくつかのガイドラインのほか，一般的な問題を解決するためのトラブルシューティングの項があります。

SAPGUI Vuser スクリプトは，**sapgui** というプレフィックスで始まる SAPGUI 関数を使用して，一般的なビジネス・プロセスをエミュレートします。

再生中，この関数は SAPGUI オブジェクトでのユーザ・アクティビティをエミュレートします。

たとえば，sapgui_select_radio_button によって「Blue」ラジオ・ボタンが選択されます。

```
sapgui_select_radio_button("Blue",
    "usr/radRB7",
    BEGIN_OPTIONAL,
        "AdditionalInfo=sapgui1027",
    END_OPTIONAL);
```

SAPGUI のオプションのウィンドウの再生

SAPGUI Vuser スクリプトの処理中に，SAPGUI クライアントにオプションのウィンドウ（記録時には表示されるのに，再生時には表示されないウィンドウ）が表示されることがあります。記録したスクリプトをそのまま再生しようとしても，存在しないウィンドウを見つけようとして失敗することになります。

VuGen のオプションのウィンドウのメカニズムは，そのウィンドウの存在の確認後にのみ，アクションを実行します。Vuser は，［**アクティブ ウィンドウを選択**］ステップに示されたウィンドウが存在するかどうかを検査します。ウィンドウが再生時に見つかれば，スクリプトに記録されているとおりにアクションを実行します。存在しない場合には，Vuser は次の［**アクティブ ウィンドウを選択**］ステップまでのすべてのウィンドウのアクションを無視します。
SAPGUI ステップ（**sapgui** プレフィックスで始まるステップ）のみが無視されます。

この機能を使用するには，ツリー・ビューで適切な［アクティブ ウィンドウを選択］ステップを選択し，右クリック・メニューから［**ウィンドウのみでステップを実行する**］を選びます。

この機能を無効にし，これらのステップが常に実行されるようにするには，Vuser がウィンドウを見つけるかどうかにかかわらず，右クリック・メニューから［**このウィンドウでは常にステップを実行する**］を選択します。

SAPGUI の関数

SAPGUI の記録セッション中，SAPGUI クライアントでのユーザの作業をエミュレートする関数が生成されます。SAPGUI for Windows クライアントを記録すると，**sapgui** というプレフィックスを持つ関数が生成されます。本項ではすべての **sapgui** 関数について説明します。

SAP Workplace または Portal などの Web インタフェースを使用して SAP セッションを記録するとき，または SAPGUI クライアントから Web コントロールを開く場合は，**web** というプレフィックスを持つ関数が生成されます。

ほとんどの関数は記録されますが，任意の関数をスクリプトに手作業で挿入することもできます。**sapgui_get** で始まるデータ取得関数と，**sapgui_is** で始まる検証用の関数は，記録されません。

sapgui 関数には，接続およびセッション関数，メソッドおよびプロパティ関数，検証およびデータ取得関数，およびオブジェクト関数など，複数のカテゴリがあります。オブジェクト関数は，SAPGUI オブジェクト（カレンダー関数，グリッド関数，APO グリッド関数，ステータス・バー関数，テーブル関数，ツリー関数，ウィンドウ関数，および一般オブジェクト関数）内でアクションを実行するものです。

sapgui および **web** 関数の詳細については，［編集］メニューから［**関数構文の自動表示**］機能を使用するか，または『**Online Function Reference**』（英語版）を参照してください（［**ヘルプ**］ > ［**関数リファレンス**］）。

SAPGUI Vuser スクリプトに関するヒント

以下の各項では，SAPGUI Vuser の記録に関するヒント，再生に関するヒント，およびシナリオ内での再生に関するヒントについて説明します。また，SAP のサポート・サイトからも直接情報を参照できます。

記録に関するヒント

本項では，SAPGUI Vuser スクリプトの記録に関するヒントについて説明します。

- アクションを適切なセクションに記録するようにします。ログイン手順は **vuser_init** セクションに，反復実行するアクションは **Actions** セクションに，ログオフ手順は **vuser_end** セクションに記録します。
- SAPGUI クライアントに Web コントロールが含まれているマルチ・プロトコル・スクリプトを記録する場合は，記録を開始する前に SAPLogon アプリケーションを終了します。
- F1 に対応したモーダル・ダイアログ・ボックスを使用します。F1 ヘルプをモーダル・ダイアログ・ボックスで開くように SAPGUI に指定します。［**ヘルプ**］＞［**設定**］を選択します。［**F1 Help**］タブをクリックし，［照会］セクションの［**Modal ダイアログボックス**］オプションを選択します。

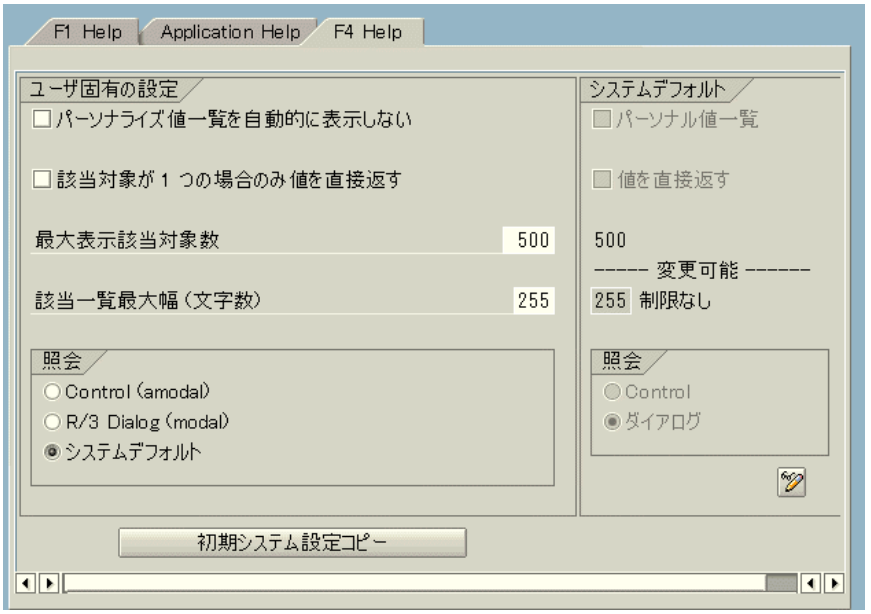
- F4 に対応したモーダル・ダイアログ・ボックスを使用します。F4 ヘルプをモーダル・ダイアログ・ボックスで開くように SAPGUI に指定します。

SAP 管理者は次の手順を実行する必要があります。

F4 ヘルプをモーダル・ダイアログ・ボックスで開くには，次の手順を実行します。

1 サーバからすべてのユーザがログオフしていることを確認してください。

2 ［**ヘルプ**］＞［**設定**］を選択します。［**F4 Help**］タブをクリックします。



3 ［照会］セクション（左下）で，［システムデフォルト］を選択します。

4 ［システムデフォルト］セクションの［照会］部分（右下）で，［**ダイアログ**］を選択します。

5 変更を保存します。［**初期システム設定コピー**］をクリックするか，CTRL キーを押しながら S キーを同時に押します。

6 ステータス・バーに「**データが保存されました。**」というメッセージが表示されているかどうか確認します。

7 セッションを終了します。

8 SAP Management Console を使って，サービスを再起動します。

再生に関するヒント

スクリプトをスタンドアロン・モードで再生する前に，このガイドラインを読んでください。

- 記録時に生成された **sapgui_logon** 関数の暗号化されたパスワードを，実際のパスワードに置き換えます。次に示すような関数の 2 つ目の引数（ユーザ名の次）がパスワードです：sapgui_logon("user", "pswd", "800", "EN"); セキュリティ向上のため，コード内のパスワードは暗号化できるようになっています。パスワード・テキスト（***** ではなく実際のテキスト）を選び，右クリック・メニューで［**文字列を暗号化**］を選択します。**lr_decrypt** 関数（sapgui_logon("user", lr_decrypt("3ea037b758"), "800", "EN");）がパスワードの位置に挿入されます。
- 初めてスクリプトを実行する場合は，再生時に SAPGUI ユーザ・インタフェースを表示するように VuGen を設定し，その UI を使って実行されている操作を確認できるようにします。再生中にユーザ・インタフェースを表示するには，実行環境の設定を開き（F4），［**SAPGUI : 一般**］ノードで［**再生時に SAP クライアントを表示する**］オプションを選択します。複数の Vuser を実行すると UI の表示に多量のシステム・リソースが使用されるので，負荷シナリオの実行中はこのオプションを無効にします。

シナリオ内での再生に関するヒント

以下の各項では，Controller や Load Generator マシンでスクリプトを実行する際の設定に関するヒントについて説明します。

Controller の設定

LoadRunner シナリオを対象に作業をしているとき，負荷テストの構成でスクリプトを実行する場合は次の値を設定します。

- **ランプアップ**：（適切なログオンを保証するために）1 つずつスケジューラで設定します。
- **思考遅延時間**：実行環境に乱数の思考遅延時間を設定します。
- **Load Generator ごとのユーザ数**：512 MB のメモリを搭載したマシンでは，［ロード・ジェネレータ］ダイアログ・ボックスで 50 Vuser を設定します。

Load Generator の設定

スクリプトをシナリオで実行するときは，Load Generator マシンでエージェント・モードを確認し，ターミナル・セッションを設定します。

➤ **エージェント・モード**：プロセス・モードで LoadRunner（または Performance Center）Remote Agent が実行されていることを確認します。サービス・モードはサポートされていません。

これを調べるには，Windows のタスクバーにあるエージェントのアイコン上にマウスを移動し，説明を読みます。「LoadRunner Agent Service」と説明に表示された場合は，サービスとしてエージェントが実行されています。

プロセスとしてエージェントを再起動するには，次の手順を実行します。

1 エージェントを停止します。LoadRunner Agent のアイコンを右クリックし，［**閉じる**］を選択します。

2 **magentproc.exe** を実行します。これは，LoadRunner インストール先の **launch_service¥bin** ディレクトリにあります。

3 次回マシンを起動したときに正しいエージェントが起動されるようにするには，Agent Service の開始方法を［自動］から［手動］に変更します。**magentproc.exe** へのショートカットを Windows の［スタートアップ］フォルダに追加します。

ターミナル・セッション：SAPGUI Vuser を実行するマシンは，使用できるグラフィック・リソースによっては，実行できる Vuser の数が限られる可能性があります。各マシンの Vuser 数を増やすには，Load Generator マシンで追加の Terminal Server セッションを開始します。［**スタート**］＞［**プログラム**］＞［＜**製品名**＞］＞［**詳細設定**］から［**エージェント設定**］を選択し，［ターミナル サービスを有効にする］オプションを選択します。Load Generator マシンのプロパティで，ターミナル・セッションの数を指定します。詳細については，『**HP LoadRunner Controller ユーザーズ・ガイド**』の「ターミナル・サービスの設定」を参照してください。

注：LoadRunner エージェントがターミナル・セッションで実行されているときに，ターミナル・セッションのウィンドウが最小化されていると，エラー発生時にスナップショットがキャプチャされません。

SAPGUI Vuser スクリプトのトラブルシューティング

質問 1: スクリプトは記録できました。しかし再生できません。なぜでしょうか？

回答：プロセス・モードで LoadRunner Remote Agent が実行されていることを確認します。サービス・モードはサポートされていません。詳細については，728 ページ「再生に関するヒント」を参照してください。

質問 2：特定の SAPGUI コントロールが記録されないのはなぜですか？

回答：一部の SAPGUI コントロールはそのメニューまたはツールバーのコンテキストの中でのみサポートされます。問題のあるタスクについて，メニュー・オプションやショートカット・メニュー，ツールバーなどのさまざまな手段を使用して実行を試みます。

質問 3：VuGen でスクリプトの記録や再生ができないのはなぜですか？

回答：

a SAPGUI 6.20 の最新のパッチがインストールされていることを確認します。最低でもパッチ・レベル 32 である必要があります。

b スクリプティングが有効になっていることを確認します。699 ページ「SAPGUI Vuser のための環境の確認」を参照してください。

c SAPGUI for Windows クライアントで通知が無効にされていることを確認します。［ローカルレイアウトのカスタマイジング］ボタンをクリックするか，Alt キーを押しながら F12 キーを押します。［**オプション**］をクリックして［スクリプト］タブを選択します。両方の［**通知**］オプションをクリアします。

質問 4：スクリプトを実行しようとしたときに表示されるエラー・ポップアップ・メッセージはどのような意味ですか？

回答：SAP アプリケーションの中にはユーザごとに直前のレイアウトを格納するものがあります（どのフレームが表示か非表示か，など）。スクリプトの記録後に，格納されているレイアウトが変更された場合，再生に問題が生じることがあります。たとえば，ME52N トランザクションでは，「**Document overview Off/On**」ボタンによって，表示されるフレームの数が変わります。

この問題が発生した場合は，次のようにします。

1 再生を開始する前に，トランザクションの記録中と同じ位置に移動します。スナップショット・ビューアを使用すると，トランザクションが記録されたレイアウトを表示できます。

2 スクリプトにステートメントを追加して，再生中にトランザクションが望みのレイアウトになるようにします。たとえば，オプションのフレームによって再生が妨げられる場合は，フレームが開いているかどうかを確かめる検証関数を挿入します。フレームが開いている場合は，ボタンをクリックして閉じます。検証の例については，719 ページ「検証関数の追加」を参照してください。

質問 5：スクリプトをリモート・マシンで実行するときにシングル・サインオンのメカニズムを使用できますか？

回答：できません。VuGen ではシングル・サインオンの接続メカニズムはサポートされていません。SAPGUI クライアントで，［拡張オプション］を開き，［**安全なネットワーク通信有効化**］機能をクリアします。接続の設定を変更した後，スクリプトを記録し直す必要があります。

質問 6：VuGen ですべての SAP オブジェクトを記録できますか？

回答：SAPGUI スクリプティングでサポートされていないオブジェクトについては，記録はできません。これらのオブジェクトの詳細については記録ログを参照してください。

質問 7：すべてのビジネス・プロセスがサポートされていますか？

回答：VuGen では，Microsoft Office のモジュール・コントロールを起動するビジネス・プロセス，あるいは GuiXT の使用を必要とするビジネス・プロセスはサポートされていません。**GuiXT** は SAPGUI for Windows クライアントの［オプション］メニューから無効にできます。

その他の参考資料

LoadRunner

ダイアログ・ボックスに関するオンライン・ヘルプを参照するには，ダイアログ・ボックスの中で F1 キーを押します。［**ヘルプ**］＞［**目次と索引**］を選択してヘルプを手作業で開くこともできます。［目次］タブで，「**SAPGUI Vuser スクリプト**」という項目を探し，該当するサブ項目をクリックします。

関数に関するオンライン・ヘルプを参照するには，関数またはステップの中をクリックし，F1 キーを押して『**Online Function Reference**』（英語版）を開きます。

SAP

詳細については，SAP の Web サイト（www.sap.com）または，次のいずれかの場所を参照してください。

➤ **SAP に関する注意事項** ー https://websmp103.sap-ag.de/notes

Note #480149: New profile parameter for user scripting on the front end
（フロント・エンドのユーザ・スクリプティングのための新しいプロファイル・パラメータ）

Note #587202: Drag & Drop is a known limitation of the SAPGUI interface
（ドラッグアンドドロップは，SAPGUI インタフェースの既知の制限）

➤ **SAP のパッチ** ー https://websmp104.sap-ag.de/patches

SAP GUI for Windows ー SAPGUI 6.20 パッチ（パッチ・レベル 32 以上）

第 48 章

SAP（Click and Script）プロトコル

VuGen では，Web を通じて SAP アプリケーションをエミュレートするスクリプトを作成できます。

本章の内容

SAP（Click and Script）Vuser スクリプトの作成について

VuGen では，SAP 用にカスタマイズされた特別なテスト・オブジェクトおよびメソッドを使用して，SAP Enterprise portal7 および SAP ITS 6.20/6.40 環境用のテスト・スクリプトを作成できます。オブジェクトは，SAP に対する HP QuickTest サポートに基づく API です。

テストまたはコンポーネントを SAP アプリケーションに記録すると，VuGen によって，実行した操作が記録されます。VuGen は，フレーム，テーブル・コントロール，iViews，およびポータルなど，特殊な SAP ウィンドウ・オブジェクトを認識します。

VuGen では，SAP コントロール（ボタン，チェック・ボックス，ドロップダウン・メニュー，エディット・フィールド，iview，リスト，メニュー，ナビゲーション・バー，OK コード，ポータル，ラジオ・グループ，ステータス・バー，タブ・ストリップ，テーブル，およびツリー・ビュー）の記録をサポートしています。

SAP（Click and Script）のセッションの記録

SAP Web アプリケーションをエミュレートする Vuser スクリプトを作成するには，**ERP** カテゴリから **SAP（Click and Script）**プロトコル・タイプを選択します。記録を開始するには，［**記録**］ボタンをクリックし，SAP Web アプリケーションで標準的なアクションを実行します。サイン・イン情報は **vuser_init** セクションに，サイン・オフ・プロセスは **vuser_end** セクションに記録する必要があります。スクリプトの作成および記録の詳細については，『**第 1 巻 ― VuGen の使用**』の「VuGen を使った記録」を参照してください。

イベントに関連する記録オプションを設定できます。詳細については，『**第 1 巻 ― VuGen の使用**』の第 17 章「Click and Script 記録」を参照してください。

下位レベルのスクリプトが必要な場合，またはサポートされていない SAP コントロールで記録を実行する必要がある場合は，ERP カテゴリで **SAP-Web** プロトコルを使用します。

スクリプトを作成した後，スクリプトを自分の環境（LoadRunner シナリオ，Performance Center 負荷テスト，または Business Process Monitor プロファイル）に組み込みます。詳細については，**HP LoadRunner Controller**，**HP Performance Center**，または **HP Business Availibility Center** のドキュメントを参照してください。

SAP（Click and Script）スクリプトについて

VuGen では，コントロール・ハンドラ層を使用して GUI コントロールに対する操作の効果を作成します。記録中にサポートされている SAP オブジェクトのいずれかに遭遇した場合，VuGen によって，**sap_xxx** というプレフィックスを持つ関数が生成されます。

次の例では，ユーザは［**User Profile**］タブを選択しました。VuGen によって，**sap_portal** 関数が生成されています。

```
web_browser("Close_2",
        "Snapshot=t7.inf",
        DESCRIPTION,
        "Ordinal=2",
        ACTION,
        "UserAction=Close",
        LAST);

lr_think_time(7);

web_text_link("Personalize",
        "Snapshot=t8.inf",
        DESCRIPTION,
        "Text=Personalize",
        ACTION,
        "UserAction=Click",
        LAST);

    lr_think_time(6);

sap_portal("Sap Portal_2",
        "Snapshot=t9.inf",
        DESCRIPTION,
        "BrowserOrdinal=2",
        ACTION,
        "DetailedNavigation=User Profile",
        LAST);
```

注：SAP（Click and Script）セッションを記録すると，VuGen は，SAP 固有でないオブジェクトに対して標準の Web（Click and Script）関数を生成します。Web プロトコルを明示的に指定する必要はありません。前述の例では，ユーザが［**Personalize**］ボタンをクリックしたときに，VuGen によって **web_text_link** 関数が生成されます。

第 49 章

SAP-Web プロトコル

VuGen の SAP-Web Vuser を使用して，SAP Workplace および SAP Portal クライアントでの作業を記録することができます。

本章の内容

SAP-Web Vuser スクリプトの作成について

まず，VuGen で典型的な SAP ビジネス・プロセスを記録します。VuGen では，ビジネス・プロセス中の SAP Workpalce または SAP Portal のアクティビティを記録し，Vuser スクリプトを生成できます。ブラウザ内でアクションを実行すると，このアクティビティを説明する関数が生成されます。各関数の先頭には，**web** というプレフィックスが付きます。

再生中，この関数は SAP Workplace または SAP Portal クライアントでのユーザ・アクティビティをエミュレートします。たとえば，次の web_url では PageBuilder を開いています。

```
web_url("PageBuilder[myPage]",
"URL=http://sonata.hplab.com/hrnp$30001/sonata.hplab.co.il:80/Action/PageBuilder[myPage]?pageName=com.sapportals.pct.home.mynews",
        "Resource=0",
        "RecContentType=text/html",
        "Referer=http://sonata.hplab.co.il/sapportal",
        "Snapshot=t2.inf",
        "Mode=HTML",
        EXTRARES,
        "Url=/irj/services/laf/themes/portal/sap_mango_polarwind/.., ENDITEM,
        LAST);
```

SAP-Web Vuser スクリプトの作成

SAP-Web Vuser スクリプト作成の第一歩は，Vuser とスクリプトのタイプを選択することです。**SAP-Web** Vuser は，［**ERP/CRM**］カテゴリの下にあります。シングル・プロトコルとマルチ・プロトコルのどちらの Vuser スクリプトでも作成できます。また，シングル・プロトコルの SAP（Click and Script）Vuser タイプを使用できます。SAP（Click and Script）Vuser は，より高いレベル（より直観的なスクリプト）を生成します。

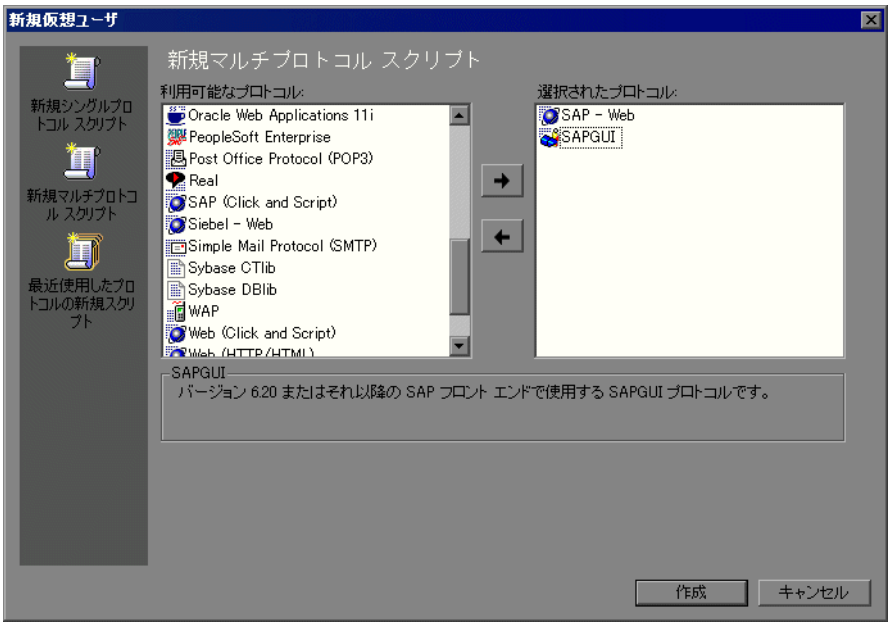
SAP-Web Vuser を作成するは，次の手順を実行します。

1 VuGen を起動し，［**ファイル**］＞［**新規作成**］を選択します。

2 SAPGUI コントロールが含まれない SAP Workplace または SAP Portal クライアントでのセッションを記録するには，**SAP-Web** Vuser タイプを使ってシングル・プロトコル Vuser スクリプトを作成します。

3 SAPGUI コントロールを使用するセッションを記録するには，次のどちらかを選択します。

- シングル・プロトコル Vuser スクリプト（**SAP**（**Click and Script**）プロトコル）
- マルチ・プロトコル Vuser スクリプト（**SAP-Web** および **SAPGUI** Vuser タイプ）



SAP-Web Vuser スクリプトについて

通常 SAP-Web Vuser スクリプトには，ビジネス・プロセスを構成する複数の SAP トランザクションが含まれています。ビジネス・プロセスは，ユーザのアクションをエミュレートする関数で構成されます。これらの関数の詳細については，『**Online Function Reference**』（英語版）（［**ヘルプ**］＞［**関数リファレンス**］）の「Web 関数」を参照してください。

SAP Portal クライアントにおける典型的な記録の例を以下に示します。

```
vuser_init()
{
    web_reg_find("Text=SAP Portals Enterprise Portal 5.0",
        LAST);

    web_set_user("junior{UserNumber}",
        lr_decrypt("3ed4cfe457afe04e"),
        "sonata.hplab.com:80");

    web_url("sapportal",
        "URL=http://sonata.hplab.com/sapportal",
        "Resource=0",
        "RecContentType=text/html",
        "Snapshot=t1.inf",
        "Mode=HTML",
        EXTRARES,

"Url=/SAPPortal/IE/Media/sap_mango_polarwind/images/header/branding_image.jpg",
"Referer=http://sonata.hplab.com/hrnp$30001/sonata.hplab.coml:80/Action/26011[header]", ENDITEM,
        "Url=/SAPPortal/IE/Media/sap_mango_polarwind/images/header/logo.gif",
"Referer=http://sonata.hplab.com/hrnp$30001/sonata.hplab.com:80/Action/26011[header]", ENDITEM,
...
        LAST);
```

次のセクションは，SAP Portal クライアントが SAP コントロールを開くマルチ・プロトコル記録を示します。web_xxx 関数から sapgui_xxx 関数に切り替わっている点に注目してください。

```
web_url("dummy",

"URL=http://sonata.hplab.com:1000/hrnp$30000/sonata.hplab.com:1000/Action/dummy?PASS_PARAMS=YES&dummyComp=dummy&Tcode=VA01&draggable=0&CompFName=VA01&Style=sap_mango_polarwind",
        "Resource=0",
        "RecContentType=text/html",
        "Referer=http://sonata.hplab.com/sapportal",
        "Snapshot=t9.inf",
        "Mode=HTML",
        LAST);

    sapgui_open_connection_ex(" /H/Protector/S/3200 /WP",
        "",
        "con[0]");

    sapgui_select_active_connection("con[0]");

    sapgui_select_active_session("ses[0]");

    /* スクリプト実行時に，ログオン関数でアスタリスクの代わりにパスワードを入力 */

    sapgui_logon("JUNIOR{UserNumber}",
        "ides",
        "800",
        "EN",
        BEGIN_OPTIONAL,
            "AdditionalInfo=sapgui102",
        END_OPTIONAL);
```

SAP-Web Vuser スクリプトの再生

SAP-Web Vuser スクリプトの作成と拡張を終えたら，その実行環境の設定を行い，VuGen から実行してその機能を確かめます。

実行環境を設定することによって，再生時の Vuser の動作を制御します。これらの設定は Vuser スクリプトを実行する前に行います。実行環境の一般設定と Web 関連の設定が可能です。

一般設定には，実行論理，ペース設定，ログ，思考遅延時間，パフォーマンスの設定が含まれます。一般的な実行環境の設定の詳細については，『**第 1 巻 － VuGen の使用**』の「実行環境の設定」を参照してください。SAP-Web 固有の設定については，『**第 1 巻 － VuGen の使用**』の第 23 章「ネットワーク実行環境の設定」を参照してください。

実行環境を設定したら，Vuser スクリプトを保存し，VuGen でスタンドアロンのテストとして実行し，正しく動作することを確認します。詳細については，『**第 1 巻 － VuGen の使用**』の「スタンドアロン・モードでの Vuser スクリプトの実行」を参照してください。

Vuser スクリプトが正常に機能することを確認したら，スクリプトを自分の環境（LoadRunner シナリオ，Performance Center 負荷テスト，または Business Process Monitor プロファイル）に組み込みます。詳細については，**HP LoadRunner Controller**，**HP Performance Center**，または **HP Business Availibility Center** のドキュメントを参照してください。

第 50 章

Siebel-Web プロトコル

VuGen を使って，Siebel Web 環境のアクティビティを記録し，Vuser スクリプトを生成できます。スクリプトを実行すると，Vuser によって Siebel 環境内のアクションがエミュレートされます。

本章の内容

Siebel-Web Vuser スクリプトの作成について

Siebel-Web プロトコルは標準の Web Vuser に似ていますが，Siebel CRM（Customer Relationship Management）アプリケーションを扱えるように，標準設定にいくつかの変更が加えられています。

Siebel セッションの一般的なアクティビティを記録します。VuGen はアクションを記録し，アクションをエミュレートする関数（**web_** というプレフィックスが付きます）を生成します。

以降の各項では，記録された Siebel-Web Vuser スクリプトで作業を行う際のヒントについて説明し，相関に必要なパラメータの例を示します。

Siebel-Web セッションの記録

Siebel-Web セッションを記録するときは，次のガイドラインに従います。

Siebel-Web Vuser スクリプトを記録するには，次の手順を実行します。

1 Siebel-Web タイプの Vuser スクリプトを ERP カテゴリから作成します。

2 次の記録オプションを設定します。

➤［記録］ノード：［**HTML ベースのスクリプト**］

［HTML 詳細設定］－［スクリプト タイプ］オプション：［**明示的な URL のみを含むスクリプト**］

［HTML 詳細設定］－［生成された HTML 以外の要素］：［**記録しない**］

➤［詳細］ノード：［**各アクションごとにコンテキストをリセットする**］オプションをクリア

3 **vuser_init** セクションにログイン手続きを記録します。

4 ビジネス・プロセスを **Action1** に記録します。

5 **vuser_end** セクションにログアウト手続きを記録します。

6 実行環境の設定で，［ブラウザのエミュレーション］ノードの［**反復ごとに新規ユーザをシミュレートする**］オプションをクリアします。

記録オプションと Web 関連の実行環境の設定方法の詳細については，『**第 1 巻 － VuGen の使用**』の 331 ページ「Web，ワイヤレス，Oracle NCA 記録オプション」および『**第 1 巻 － VuGen の使用**』の第 23 章「ネットワーク実行環境の設定」を参照してください。

Siebel-Web スクリプトの相関

Siebel セッションのテスト・スクリプトを作成する場合は，スクリプトに相関を使用する必要が生じる可能性が大いにあります。相関は，VuGen が記録時および再生時に動的な値をパラメータに保存して，スクリプトの以降の場所で使用できるようにするためのメカニズムです。記録したスクリプトを相関なしでそのまま再生すると，スクリプトを実行するたびに引数の値が変化するため，スクリプトの再生に失敗します。このような変数の例として，**SWECount** や **SWEBMC** があります。

相関を使用すると，VuGen は記録時と再生時の両方で動的な変数を保存し，スクリプト内の適切な箇所でこの変数を使用します。記録時に相関を適用するように VuGen を設定するには，次のいずれかの方法を使用します。

- Siebel 相関ライブラリ

  Siebel 相関ライブラリを使用すると，動的な値の大部分が自動的に相関され，大幅な変更を加えなくても再生できるスクリプトが作成されます。この相関方法を使用することをお勧めします。

- VuGen のネイティブ Siebel 相関

  ネイティブ組み込みルールを低レベルで適用すると，スクリプトのデバッグや相関のより詳しい理解が可能になります。

Siebel 相関ライブラリ

相関を簡単に使用できるように，Siebel Application Server バージョン 7.7 の一部として相関ライブラリ・ファイルがリリースされています。このライブラリは，Siebel でのみ使用できます。ライブラリ・ファイル **ssdtcorr.dll** は，Windows では siebsrvr¥bin フォルダの下に，UNIX では siebsrvr/lib の下にあります。

ライブラリ・ファイル **ssdtcorr.dll** は，Load Generator または Controller が存在するすべてのマシンで使用できるようにする必要があります。このライブラリのサポートには，VuGen 8.0 以降が必要です。

このライブラリを使って相関を有効にするには，次の手順を実行します。

1. ライブラリの DLL ファイルを製品のインストール先の bin ディレクトリにコピーします。
2. **Siebel-Web** Vuser タイプを使用して，マルチ・プロトコル・スクリプトを開きます。

3 記録オプションで UTF-8 のサポートを有効にします。詳細については,『**第 1 巻 ー VuGen の使用**』の第 16 章「選択したプロトコルの記録オプション」を参照してください。

4 記録オプションの［相関］ノードを開き，［**インポート**］をクリックします。¥dat¥webrulesdefaultsetting ディレクトリからルール・ファイル **WebSiebel77Correlation.cor** をインポートします。警告が表示された場合は，［**Override**］をクリックします。詳細については,『**第 1 巻 ー VuGen の使用**』の第 16 章「選択したプロトコルの記録オプション」を参照してください。

標準設定の相関に戻すには，Siebel のルールをすべて削除し，［**標準設定値を使用**］をクリックします。

Siebel 相関ライブラリを使用するときは，SWE カウント・ルール（左の境界に **SWEC** という文字列が含まれているルール）が無効になっていないことを確認します。詳細については，750 ページ「ルールの有効化と無効化」を参照してください。

VuGen のネイティブ Siebel 相関

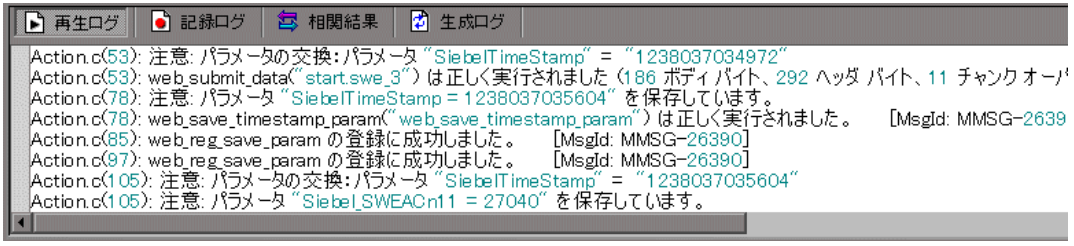
VuGen の Siebel サーバ用ネイティブ組み込みルールは，Siebel サーバの変数と文字列を検出し，それらをスクリプトの以降の場所で使用できるように保存します。

これらのルールは，相関ルールの一覧で参照できます（642 ページ「VuGen の相関ルールの使用」を参照）。これらのルールは，Siebel サーバの文字列に固有の境界条件を示します。

VuGen は，境界条件を使って一致する候補を検出すると，境界と境界の間にある値をパラメータに保存します。保存される値は，単純な変数または public 関数です。

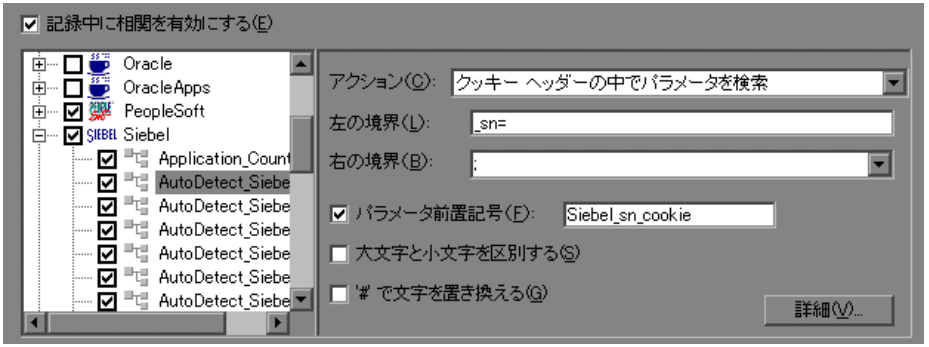
相関記録オプションに独自の境界条件を入力するか（第 42 章「Web（HTTP/HTML）の相関ルール」を参照），または記録後に［相関結果］タブを使用して（659 ページ「相関の検索の実行」を参照），独自のルールを作成することもできます。

［再生ログ］タブには，VuGen がパラメータを登録，保存，および使用したタイミングが示されます。この情報を表示するには，拡張ログを有効にする必要があります。詳細については，『**第 1 巻 － VuGen の使用**』の第 22 章「実行環境設定のログの設定」を参照してください。



単純な変数の相関

次の例では，左の境界条件が _sn= になっています。左の境界に _sn= が，右の境界に ; が現れるたびに，**Siebel_sn_cookie** というプレフィックスの付いたパラメータが作成されます。



次の例では，_sn 境界が検出されています。パラメータが Siebel_sn_cookie6 に保存され，web_url 関数で使用されています。

```
/* ソースからパラメータを登録します
web_reg_save_param("Siebel_sn_cookie6",
"LB/IC=_sn=",
"RB/IC=;",
"Ord=1",
"Search=headers",
"RelFrameId=1",
LAST);

...

web_url("start.swe_3",
"URL=http://cannon.hplab.com/callcenter_enu/start.swe?SWECmd=GotoPostedAction
&SWEDIC=true&_sn={Siebel_sn_cookie6}&SWEC={Siebel_SWECount}&SWEFrame
=top._sweclient&SWECS=true",
"TargetFrame=",
"Resource=0",
"RecContentType=text/html",
"Referer=http://cannon.hplab.com/callcenter_enu/start.swe?SWECmd=GetCachedFra
me&_sn={Siebel_sn_cookie6}&SWEC={Siebel_SWECount}&SWEFrame=top._swe",
"Snapshot=t4.inf",
"Mode=HTML",
LAST);
```

関数の相関

特定のインスタンスでは，境界の一致が関数となります。関数は通常，実行時の値を格納する配列を使用します。これらの値を相関するために，VuGen はその配列を解析し，個々の引数を次の形式で個別のパラメータに保存します。

＜パラメータ名＞ = ＜記録された値＞ (表示名)

表示名は，Siebel アプリケーションで値の隣に表示されるテキストです。

VuGen は，すべてのパラメータ定義を含むコメント・ブロックを挿入します。

```
/* ソース・タスク ID 159 からのパラメータの登録
    // {Siebel_Star_Array_Op33_7} = ""
    // {Siebel_Star_Array_Op33_6} = "1-231"
    // {Siebel_Star_Array_Op33_2} = ""
    // {Siebel_Star_Array_Op33_8} = "Opportunity"
    // {Siebel_Star_Array_Op33_5} = "06/26/2003 19:55:23"
    // {Siebel_Star_Array_Op33_4} = "06/26/2003 19:55:23"
    // {Siebel_Star_Array_Op33_3} = ""
    // {Siebel_Star_Array_Op33_1} = "test camp"
    // {Siebel_Star_Array_Op33_9} = ""
    // {Siebel_Star_Array_Op33_rowid} = "1-6F"
    // */
```

また，関数が見つかると，VuGen は web_reg_save_param に新規パラメータ AutoCorrelationFunction を生成します。また，パラメータのプレフィックスを特定し，それをパラメータ名として使用します。次の例では，プレフィックスは Siebel_Star_Array_Op33 です。

```
web_reg_save_param("Siebel_Star_Array_Op33",
    "LB/IC=`v`",
    "RB/IC=`",
    "Ord=1",
    "Search=Body",
    "RelFrameId=1",
    "AutoCorrelationFunction=flCorrelationCallbackParseStarArray",
    LAST);
```

VuGen は，パラメータをスクリプトの以降の場所で使用します。次の例では，web_submit_data でパラメータが呼び出されています。

```
web_submit_data("start.swe_14",
    "Action=http://cannon.hplab.com/callcenter_enu/start.swe",
    "Method=POST",
    "RecContentType=text/html",
    "Referer=",
    "Snapshot=t15.inf",
    "Mode=HTML",
    ITEMDATA,
    "Name=SWECLK", "Value=1", ENDITEM,
    "Name=SWEField", "Value=s_2_1_13_0", ENDITEM,
    "Name=SWER", "Value=0", ENDITEM,

    "Name=SWESP", "Value=false", ENDITEM,
    "Name=s_2_2_29_0", "Value={Siebel_Star_Array_Op33_1}", ENDITEM,
    "Name=s_2_2_30_0", "Value={Siebel_Star_Array_Op33_2}", ENDITEM,
    "Name=s_2_2_36_0", "Value={Siebel_Star_Array_Op33_3}", ENDITEM,
...
```

VuGen は，パラメータとして保存された配列要素を使用して，再生中に public 関数にコールバックを行います。

注：SWEC パラメータの相関は，相関ルールを通じて行われません。組み込み検出方式により自動的に行われます。詳細については，751 ページ「SWEC 相関」を参照してください。

ルールの有効化と無効化

通常の状況では，ルールを無効にする必要はありません。しかし，場合によっては，適用しないルールを無効にすることもできます。たとえば，英語専用のアプリケーションをテストするときは，日本語コンテンツのチェック・ルールを無効にします。

ルールを無効にするもう 1 つの理由は，Controller でエラー条件を明示的に生成する必要がある場合です。記録オプションでルールのプロパティを表示して，アプリケーションに必要な条件を特定します。

ルールを無効にするには，次の手順を実行します。

1 相関記録オプションを開きます。［**ツール**］＞［**記録オプション**］を選択し，［相関］ノードをクリックします。

2［**記録中に相関を有効にする**］オプションを選択します。サポートされているサーバがダイアログ・ボックスに表示されます。

3［Siebel］の下のルールを展開し，プロパティを表示します。

4 無効にする各ルールの横にあるチェック・ボックスをクリアします。

SWEC 相関

SWEC は，Siebel サーバによって使用されるパラメータで，ユーザのクリック数を表します。SWEC パラメータは通常，URL ステートメントまたは POST ステートメントの引数として現れます。次に例を示します。

```
GET /callcenter_enu/start.swe?SWECmd=GetCachedFrame&_sn=2-
mOTFXHWBAAGb5Xzv9Ls2Z45QvxGQnOnPVtX6vnfUU_&SWEC=1&SWEFrame=top
_swe._sweapp HTTP/1.1
```

あるいは

```
POST /callcenter_enu/start.swe HTTP/1.1
...
¥r¥n¥r¥n
SWERPC=1&SWEC=0&_sn=2-
mOTFXHWBAAGb5Xzv9Ls2Z45QvxGQnOnPVtX6vnfUU_&SWECmd=InvokeMethod
...
```

VuGen は，関連する各ステップの前にカウンタの数を増やして SWEC の変更を処理します。VuGen は SWEC の現在の値を別の変数（Siebel_SWECount_var）に保存します。各ステップの前に，VuGen はカウンタの値を VuGen パラメータ（Siebel_SWECount）に保存します。

次の例では，web_submit_data は SWEC パラメータ Siebel_SWECount. の動的な値を使用しています。

```
Siebel_SWECount_var += 1;

lr_save_int(Siebel_SWECount_var, "Siebel_SWECount");

web_submit_data("start.swe_8",
    "Action=http://cannon.hplab.com/callcenter_enu/start.swe",
    "Method=POST",
    "TargetFrame=",
    "RecContentType=text/html",
    "Referer=",
    "Snapshot=t9.inf",
    "Mode=HTML",
    "EncodeAtSign=YES",
    ITEMDATA,
    "Name=SWERPC", "Value=1", ENDITEM,
    "Name=SWEC", "Value={Siebel_SWECount}", ENDITEM,
    "Name=SWECmd", "Value=InvokeMethod", ENDITEM,
    "Name=SWEService", "Value=SWE Command Manager", ENDITEM,
    "Name=SWEMethod", "Value=BatchCanInvoke", ENDITEM,
    "Name=SWEIPS",...
    LAST);
```

SWEC パラメータは参照 URL にも使用されます。ただし，参照 URL の値は要求される URL の値とは異なります。VuGen はこれを自動的に処理します。

SWECount，ROWID，および SWET パラメータの相関

本項では，次のような特殊なパラメータの相関についてヒントを示します。

- SWECount
- 行 ID 長
- SWETS（タイム・スタンプ）

SWECount

SWECount パラメータの値は，通常 1 桁または 2 桁の小さい数値です。記録されたどの値をパラメータで置換するべきか決めるのが困難なことがしばしばあります。

web_submit_data 関数では，VuGen は SWEC フィールドの数値だけを置換します。

URL では，文字列「SWEC=」または「SWEC`」の後に現れる場合にかぎり，この値を置換します。

すべての SWECount 相関のパラメータ名は同じです。

行 ID 長

場合によっては，**rowid** の前に，その長さが 16 進形式でエンコードされて置かれます。この長さは変更される可能性があるため，その値を相関させる必要があります。

たとえば，xxx6_1-4ABCyyy という文字列は長さの値と RowID で構成されています。ここで 6 は長さ，1-4ABC は RowID です。

文字列を相関させるパラメータを，詳細相関を使用して

```
xxx{rowid_Length}_{rowid}yyy
```

と定義した場合，VuGen は文字列の前に次の関数を生成します。

```
web_save_param_length("rowid", LAST);
```

この関数は **rowid** の値を取得し，その長さをパラメータ **rowid_length** に 16 進形式で保存します。

SWETS（タイム・スタンプ）

スクリプト内の SWETS の値は，1970 年 1 月 1 日午前 0 時を基点として経過したミリ秒数です。

VuGen は，スクリプト内にある空でないすべてのタイム・スタンプを，パラメータ {SiebelTimeStamp} に置き換えます。このパラメータに値を保存する前に，VuGen は次の関数を生成します。

```
web_save_timestamp_param("SiebelTimeStamp", LAST);
```

この関数によって，現在のタイム・スタンプが **SiebelTimeStamp** パラメータに保存されます。

Siebel-Web Vuser スクリプトのトラブルシューティング

本項では，スクリプトを作成するときに発生する可能性のあるエラーとブレークダウン診断ツールについて説明します。

- 一般的なエラー
- ブレークダウン情報の記録

一般的なエラー

Siebel-Web Vuser スクリプトの作成中に，次のエラーが 1 つまたは複数発生する場合があります。

- 「戻る」または「更新」のエラー
- 同一の値
- 「No Content」HTTP 応答
- コンテキストの復元
- レコードの検索不能
- ファイルの終わり
- 検索カテゴリの取得不能

「戻る」または「更新」のエラー

「戻る」または「更新」に関連するエラー・メッセージでは，通常は次のようなテキストが表示されます。

We are unable to process your request. This is most likely because you used the browser BACK or REFRESH button to get to this point.

原因：次の原因が考えられます。

- SWEC が現在の要求と正しく相関されていなかった。
- SWETS が現在の要求と正しく相関されていなかった。
- SWEC が更新されないまま，要求が 2 回 Siebel サーバに送信された。
- 前の要求によってブラウザがダウンロードを行うためのフレームが開かれている必要があった。しかし，おそらくは記録後に SWEMethod が変更されたために，このフレームがサーバに作成されていなかった。

同一の値

同一の値のエラーに対しては，通常は次のような Web ページの応答が表示されます。

@0`0`3`3``0`UC`1`Status`Error`SWEC`10`0`1`Errors`0`2`0`Level0`0`ErrMsg`The same values for 'Name' already exist. If you would like to enter a new record, please make sure that the field values are unique.`ErrCode`28591`

原因：要求内の値の 1 つ（上の例では Name フィールドの値）がデータベース・テーブルの別の行の値と重複していることが考えられます。この値は，ユーザごとに繰り返し使用するたびに，一意な値に置き換える必要があります。解決方法として，行 ID が必ず一意となるようにパラメータに置き換えることをお勧めします。

「No Content」HTTP 応答

「No Content」HTTP 応答タイプのエラーでは，通常は次のような HTTP 応答があります。

HTTP/1.1 204 No Content
Server: Microsoft-IIS/5.0
Date: Fri, 31 Jan 2003 21:52:30 GMT
Content-Language: en
Cache-Control: no-cache

原因：行 ID がまったく相関されていないか，または正しく相関されていないことが考えられます。

コンテキストの復元

コンテキストの復元タイプのエラーでは，通常は次のような Web ページ応答があります。

@0`0`3`3``0`UC`1`Status`Error`SWEC`9`0`1`Errors`0`2`0`Level0`0`ErrMsg`An error happened during restoring the context for requested location`ErrCode`27631`

原因：行 ID が相関されていないか，または相関が正しくないことが考えられます。

レコードの検索不能

レコードの検索不能タイプのエラーでは，通常は次のような Web ページ応答があります。

@0`0`3`3``0`UC`1`Status`Error`SWEC`23`0`2`Errors`0`2`0`Level0`0`ErrMsg`Cannot locate record within view: Contact Detail - Opportunities View applet: Opportunity List Applet.`ErrCode`27573`

原因：Web ページのレコードの行 ID が入力名 SWERowId に含まれていないことが考えられます。入力名はパラメータ化しておく必要があります。パラメータのソース値でその場所が変更されている可能性があります。

ファイルの終わり

ファイルの終わりタイプのエラーでは，通常は次のような Web ページ応答があります。

@0`0`3`3``0`UC`1`Status`Error`SWEC`28`0`1`Errors`0`2`0`Level0`0`ErrMsg`An end of file error has occurred. Please continue or ask your systems administrator to check your application configuration if the problem persists.`ErrCode`28601`

原因：Web ページのレコードの行 ID が入力名 SWERowId に含まれていないことが考えられます。入力名はパラメータ化しておく必要があります。パラメータのソース値でその場所が変更されている可能性があります。

検索カテゴリの取得不能

検索カテゴリの取得不能タイプのエラーでは，通常は次のような Web ページ応答があります。

原因：検索フレームがサーバからダウンロードされなかったことが考えられます。この問題は，前の要求で検索フレームを正しく作成するようサーバに要求していなかったときに発生します。

ブレークダウン情報の記録

VuGen には，テストのトランザクション・コンポーネントを理解するための診断ツールとして，「**トランザクション・ブレークダウン**」が用意されています。トランザクション・ブレークダウン情報を使用して，ボトルネックとなっている場所と解決の必要がある問題を特定できます。

トランザクション・ブレークダウンのスクリプトを準備する場合は，テスト 1 時間当たり 1 秒の割合で各トランザクションの最後に思考遅延時間を追加することをお勧めします。思考遅延時間の入力の詳細については『**第 1 巻 － VuGen の使用**』の「Vuser スクリプトの拡張」を参照してください。

トランザクション・ブレークダウン情報を記録するためには，お使いのスクリプト内のパラメータ化されたスクリプトを変更する必要があります。

トランザクション・ブレークダウンのスクリプトを準備するには，次の手順を実行します。

1 Session ID のスクリプト・パラメータ化置換を特定します。

```
/* ソース・タスク ID 15 からのパラメータの登録
    // {Siebel_sn_body4} = "28eMu9uzkn.YGFFevN1FdrCfCCOc8c_"
    // */
    web_reg_save_param("Siebel_sn_body4",
        "LB/IC=_sn=",
        "RB/IC=&",
        "Ord=1",
        "Search=Body",
        "RelFrameId=1",
        LAST);
```

2 次の **web_submit_data** 関数を，**lr_start_transaction** 関数と **lr_end_transaction** 関数で囲み，トランザクションとしてマークします。

3 トランザクションの末尾の前に，**lr_transaction_instance_add_info** への呼び出しを追加します。ここで，最初のパラメータ 0 は必須で，セッション ID には SSQLBD プレフィックスが付きます。

```
lr_start_transaction("LoginSQLSync");
    web_submit_data("start.swe_2",
        "Action=http://design/callcenter_enu/start.swe",
        "Method=POST",
        "RecContentType=text/html",
        "Referer=http://design/callcenter_enu/start.swe",
        "Snapshot=t2.inf",
        "Mode=HTML",
        ITEMDATA,
        "Name=SWEUserName", "Value=wrun", ENDITEM,
        "Name=SWEPassword", "Value=wrun", ENDITEM,
        "Name=SWERememberUser", "Value=Yes", ENDITEM,
        "Name=SWENeedContext", "Value=false", ENDITEM,
        "Name=SWEFo", "Value=SWEEntryForm", ENDITEM,
        "Name=SWETS", "Value={SiebelTimeStamp}", ENDITEM,
        "Name=SWECmd", "Value=ExecuteLogin", ENDITEM,
        "Name=SWEBID", "Value=-1", ENDITEM,
        "Name=SWEC", "Value=0", ENDITEM,
        LAST);

lr_transaction_instance_add_info(0,lr_eval_string("SSQLBD:{Siebel_sn_body4}"));
lr_end_transaction("LoginSQLSync", LR_AUTO);
```

注：セッション ID の衝突を避けるためには，各セッションの終わりに Vuser がデータベースから必ずログオフするようにしてください。

第 51 章

RTE プロトコル

RTE Vuser は，Windows 環境でターミナル・エミュレータを操作します。本章では，Windows ベースの RTE Vuser スクリプトを作成する方法について説明します。

本章の内容

RTE Vuser スクリプトの作成について

RTE Vuser は，クライアント / サーバ・システムの負荷テストを行うために，ターミナル・エミュレータを操作します。

VuGen を使用してターミナル・エミュレータ・セッションを記録することにより，実際のユーザのアクションを表現します。記録したスクリプトは，トランザクション関数や同期化機能によって拡張できます。

本章では，Windows ベースの RTE Vuser スクリプトを作成する方法について説明します。

RTE Vuser の紹介

RTE Vuser は，文字入力データをターミナル・エミュレータに入力し，それらのデータをサーバに送信した後，サーバから応答があるまで待機します。たとえば，あるメンテナンス会社の顧客情報を管理するサーバがあるとします。フィールド・サービス担当者は，修理を行うたびにターミナル・エミュレータを使ってモデム経由でサーバのデータベースにアクセスします。サービス担当者は顧客の情報にアクセスし，自分が行った修理の詳細を記録します。

RTE Vuser を使用して，この事例をエミュレートできます。RTE Vuser は次の操作を実行します。

1 コマンド・ラインで「**60**」と入力して，アプリケーション・プログラムを開きます。

2 フィールド・サービス担当者の番号として「**F296**」と入力します。

3 顧客番号として「**NY270**」と入力します。

4 画面上に 「詳細」という単語が表示されるまで待機します。「詳細」の表示は，画面上に顧客の詳細がすべて表示されたことを示します。

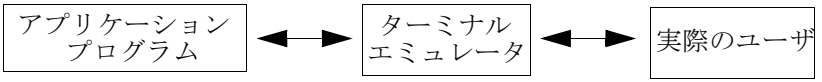
5 最新の修理の詳細として「**ガスケット P249 を交換，主要サービスを実施**」と入力します。

6「**Q**」と入力してアプリケーション・プログラムを閉じます。

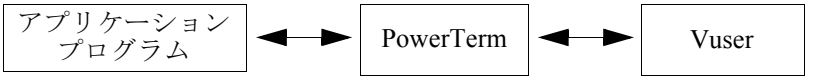
RTE Vuser スクリプトを作成するには，VuGen を使用します。スクリプト・ジェネレータは，実際のユーザがターミナル・エミュレータで行ったアクションを記録します。ターミナル・ウィンドウでのキーボード入力を記録し，対応するステートメントを生成し，それらを Vuser スクリプトに挿入します。記録の実行中，スクリプト・ジェネレータはスクリプトに同期化関数を自動的に挿入します。詳細については，第 53 章「RTE ― 同期」を参照してください。

RTE Vuser 技術について

RTE Vuser は，実際のユーザのアクションをエミュレートします。実際のユーザは，ターミナルまたはターミナル・エミュレータを使用してアプリケーション・プログラムを操作します。



RTE Vuser 環境では，Vuser が実際のユーザの代わりに操作を行います。Vuser は PowerTerm というターミナル・エミュレータを操作します。



PowerTerm は，標準的なターミナル・エミュレータと同じように動作し，IBM 3270，IBM 5250，VT100，VT220，および VT420-7 などの一般的なプロトコルをサポートします。

RTE Vuser スクリプトの概要

本項では，VuGen を使った RTE Vuser スクリプトの作成プロセスの概要を説明します。

RTE Vuser スクリプトを作成するには，次の手順を実行します。

1 VuGen を使用して基本スクリプトを記録します。

仮想ユーザ・ジェネレータ（VuGen）を使用して，ターミナル・エミュレータで行われる操作を記録します。ターミナル・ウィンドウでのキーボード入力を記録し，対応するステートメントを生成して Vuser スクリプトに挿入します。

詳細については，第 52 章「RTE ― 記録」を参照してください。

2 スクリプトを拡張します。

Vuser スクリプトに，トランザクション，ランデブー・ポイント，同期化関数，および制御フロー構造を挿入して，スクリプトを拡張します。

詳細については，『**第 1 巻 ― VuGen の使用**』の「Vuser スクリプトの拡張」を参照してください。

3 パラメータを定義します（任意）。

スクリプトに記録された固定値の代わりにパラメータを定義します。固定値をパラメータで置き換えることにより，同じビジネス・プロセスを，異なる値で何度も繰り返すことができます。

詳細については，『**第 1 巻 ー VuGen の使用**』の「パラメータの作成」を参照してください。

4 実行環境を設定します。

実行環境を設定することによって，スクリプト実行時の Vuser の動作を制御します。設定には，ループ，ログ，タイミング情報などがあります。

詳細については，『**第 1 巻 ー VuGen の使用**』の「実行環境の設定」を参照してください。

5 VuGen でスクリプトを実行します。

VuGen からスクリプトを実行して，スクリプトが正しく実行されることを確認します。標準出力を表示して，プログラムがサーバと正常に通信していることを確認します。

詳細については，『**第 1 巻 ー VuGen の使用**』の「スタンドアロン・モードでの Vuser スクリプトの実行」を参照してください。

スクリプトを作成した後，スクリプトを自分の環境（LoadRunner シナリオ，Performance Center 負荷テスト，または Business Process Monitor プロファイル）に組み込みます。詳細については，**HP LoadRunner Controller**，**HP Performance Center**，または **HP Business Availibility Center** のドキュメントを参照してください。

TE 関数の使用

ターミナルとサーバの通信をエミュレートするために開発された関数を，TE Vuser 関数と呼びます。TE Vuser 関数の名前には，**TE** というプレフィックスが付きます。VuGen は，RTE セッション中に，本項に列挙する TE 関数のほとんどを自動的に記録します。スクリプトに任意の関数を手作業でプログラミングすることもできます。

TE 関数は，ターミナル・エミュレータ接続，テキスト取得，カーソル，システム変数，エラー・コード，タイピング，および同期の関数というカテゴリに分類されます。

また，手作業で任意の関数をプログラミングして，スクリプトに挿入することもできます。テキスト・ビューで，Intellisense と自動補完機能を利用して，手作業で新しい関数を追加できます。ツリー・ビューで［**挿入**］＞［**新規ステップ**］を選択し，必要なステップを選択します。

TE 関数の構文と使用例については，『**Online Function Reference**』（英語版）（［**ヘルプ**］＞［**関数リファレンス**］）を参照してください。

Ericom ターミナル・エミュレーションを使った作業

VuGen では，Ericom ターミナル・エミュレータを使った記録と再生をサポートしています。

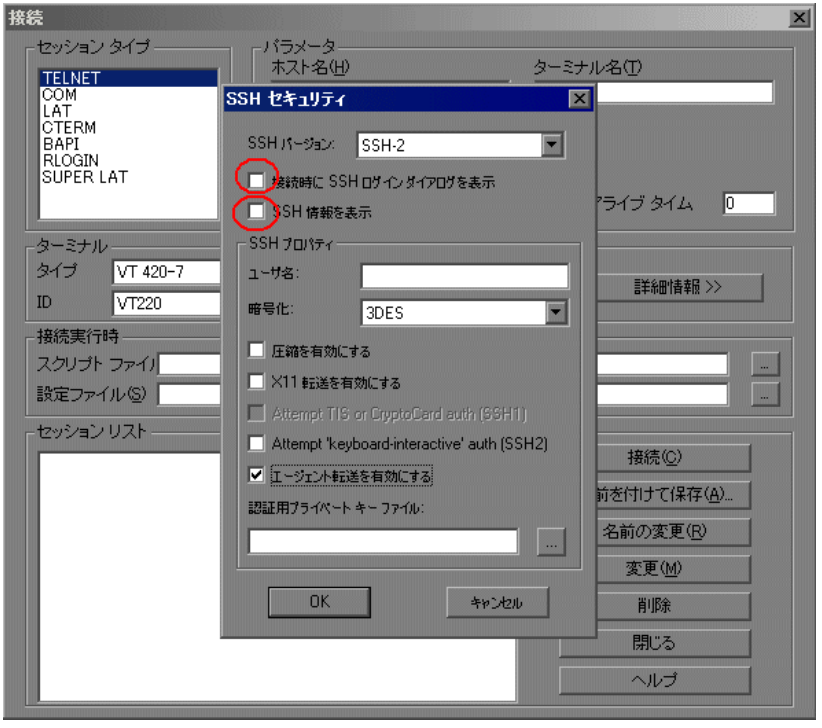
Ericom サポートでは，記録と再生中にエスケープ・シーケンスを扱えます。Ericom PowerTerm を使うと，PC キーをユーザ定義エスケープ・シーケンスに割り当てられます。割り当ての詳細については，PowerTerm のヘルプを参照してください。

Ericom VT セッションの記録中にユーザが割り当てられたキーを押すと，VuGen によって，標準の **TE_type** の代わりに **TE_send_text** 関数が生成されます。これにより，スクリプトは単一ステップでユーザ定義エスケープ・シーケンスを扱うことができます。詳細については，『**Online Function Reference**』（英語版）（［**ヘルプ**］＞［**関数リファレンス**］）で **TE_send_text** 関数を参照してください。

Ericom での SSL と SSH のサポート

VuGen では，RTE Ericom ライブラリに対する SSL/SSH での記録と再生をサポートしています。SSL または SSH を使って作業するには，［接続］ダイアログ・ボックスの［**セキュリティ**］セクションで種類を選択します。

SSH セキュリティを使って作業する場合は，標準設定で，VuGen によって詳細な情報を要求するポップアップ・ダイアログ・ボックスが表示されます。ポップアップが表示されないようにするために，［**表示**］オプションを無効にすることをお勧めします。ポップアップを有効にすると，再生に影響する場合があります。詳細なセキュリティ・オプションにアクセスするには，［**詳細**］ボタンをクリックします。



ターミナル・キーの PC キーボード・キーへの割り当て

ターミナル・エミュレータを使用する場合は，ターミナル・キーボードの代わりに PC キーボードを使用します。ターミナル・キーボードには，PC キーボードにはないキーが多数あります。このようなキーの例として，IBM 5250 キーボードの HELP キー，AUTOR キー，PUSH キーなどがあります。ターミナル・エミュレータや関連するアプリケーション・プログラムを正常に動作させるには，特定のターミナル・キーを PC キーボードのキーに割り当てることが必要な場合があります。

ターミナル・キーを PC キーボードのキーに割り当てるには，次の手順を実行します。

1 ターミナル・エミュレータで，［**オプション**］＞［**キーボードの割り当て**］を選択するか，［**キーボードの割り当て**］ボタンをクリックします。［キーボード割り当て］ダイアログ・ボックスが開きます。

2 ツールバーの［**キーボードの割り当て**］ボタンをクリックします。ターミナル・キーを PC キーに割り当てるには，上側のターミナル・キーボードのキーを下側の PC キーボードのキーにドラッグします。

上側のキーボードで Shift キー，Ctrl キー，あるいはその両方をクリックすると，追加のキー機能が表示されます。追加のキー機能は，これらのキーのいずれかを最初に選択しないかぎり表示されません。キーが表示された後は，上側のターミナル・キーボードの必要なキーを下側の PC キーボードのキーにドラッグできます。

定義を取り消すには，PC キーの定義をごみ箱にドラッグします。これで，PC キーの機能が標準設定に戻ります。

標準の割り当てに戻すには，［**標準設定**］をクリックします。

第 52 章

RTE ー 記録

VuGen を使用して，Windows ベースのリモート・ターミナル・エミュレーション（RTE）Vuser スクリプトを記録できます。

本章の内容

RTE Vuser スクリプトの記録について

VuGen を使用して，Windows ベースの RTE Vuser スクリプトを記録できます。VuGen は PowerTerm ターミナル・エミュレータを使用して，広範囲なターミナル・タイプをエミュレートします。PowerTerm は，一般的なターミナル接続と，その後の一般的なビジネス・プロセスの実行に使用します。その後に，ログオフ手順を実行します。ターミナル・エミュレータ内で一般的なユーザ・アクションを実行している間，VuGen は対応するステートメントを生成し，それらを Vuser スクリプトに挿入します。記録中に，スクリプトを表示および編集することができます。

RTE Vuser スクリプトを記録する前に，記録オプションが正しく設定されていることを確認してください。記録オプションを使用することで，Vuser スクリプトの記録中に VuGen によって特定の関数が生成される方法を制御できます。記録オプションは，以降のすべての記録セッションの間適用されます。

RTE Vuser スクリプトの新規作成

ユーザのアクションを Vuser スクリプトに記録する前に，Vuser スクリプトを開く必要があります。既存のスクリプトを開くか，新しいスクリプトを作成します。新しい Vuser スクリプトを作成するには VuGen を使用します。

新しい RTE Vuser スクリプトを作成するには，次の手順を実行します。

1 製品の開始メニューから［**Virtual User Generator**］を選択します。VuGen のウィンドウが開きます。

2 ［**新規作成**］ボタンをクリックします。［**新規仮想ユーザ**］ダイアログ・ボックスが開きます。

3 ［**レガシ**］プロトコル・タイプを選択し，［**Terminal Emulator (RTE)**］を選択します。［**OK**］をクリックします。VuGen によって空の RTE スクリプトが生成され，カーソルの位置が **vuser_init** セクション内で記録を開始する位置に表示されます。

ターミナルの設定と接続の手順の記録

スケルトンの Vuser スクリプトを作成した後，ターミナルの設定と接続の手順をスクリプトに記録します。VuGen は，RTE Vuser スクリプトを記録するときに PowerTerm ターミナル・エミュレータを使用します。

ターミナルの設定と接続の手順を記録するには，次の手順を実行します。

1 既存の RTE Vuser スクリプトを開くか，新しい RTE Vuser スクリプトを作成します。

2 ［**セクション**］ボックスで，記録されたステートメントの挿入先となるセクションを選択します。使用できるセクションは **vuser_init**，**Action**，および **vuser_end** です。

注：ターミナルの設定と接続の手順は，必ず **vuser_init** セクションに記録するようにします。**vuser_init** セクションは，Vuser スクリプトの反復を複数回実行するときに繰り返されません。繰り返されるのは **Actions** セクションだけです。反復回数の設定の詳細については，『**第 1 巻 ― VuGen の使用**』の「実行環境の設定」を参照してください。

---

3 Vuser スクリプト内で，記録を開始する位置にカーソルを置きます。

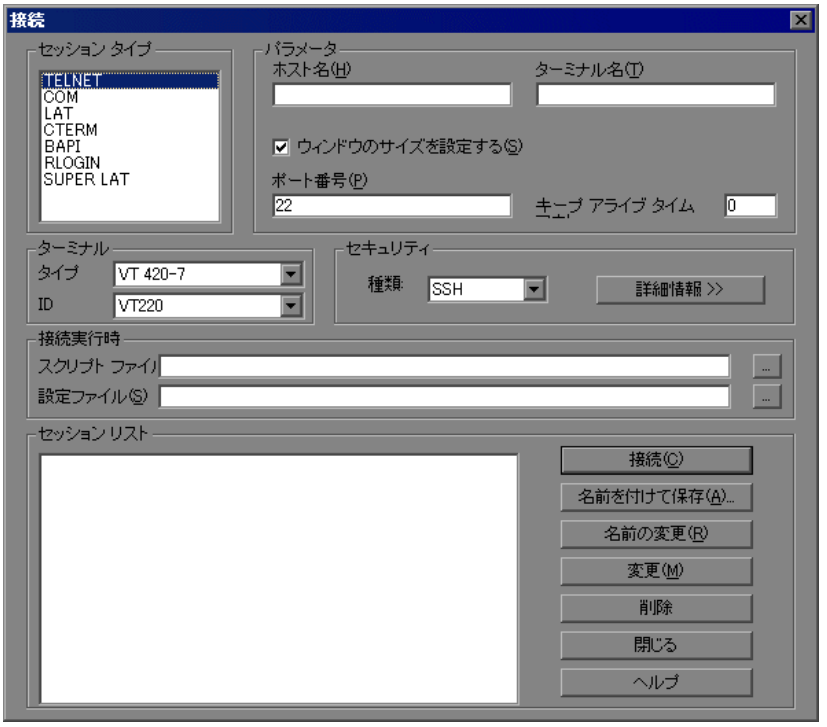
4 ［**記録**］ボタンをクリックします。PowerTerm のメイン・ウィンドウが開きます。

5 PowerTerm のメニュー・バーから［**ターミナル**］＞［**設定**］を選択し，［ターミナルの設定］ダイアログ・ボックスを表示します。

6 エミュレーションのタイプを VT ターミナル・タイプおよび IBM ターミナル・タイプから選択し，［**OK**］をクリックします。

注： AS/400 マシンまたは IBM メインフレームに接続する場合は，IBM ターミナル・タイプを選択します。UNIX ワークステーションに接続する場合は，VT ターミナル・タイプを選択します。

7［**通信**］＞［**接続**］を選択し，［接続］ダイアログ・ボックスを表示します。



8［**セッション タイプ**］で，使用する通信のタイプを選択します。

9［**パラメータ**］で，必要なオプションを指定します。使用できるパラメータは，選択したセッションのタイプによって異なります。パラメータの詳細については，［**ヘルプ**］をクリックしてください。

ヒント：将来再利用できるようにパラメータ・セットを保存するには，［**名前を付けて保存**］をクリックします。保存したパラメータ・セットが［セッション リスト］ボックスに表示されます。

10［**接続**］をクリックします。PowerTerm は指定されたシステムに接続し，VuGen は **TE_connect** 関数をスクリプトの挿入ポイントに挿入します。**TE_connect** ステートメントの形式は次のとおりです。

```
/* *** ターミナル・タイプは VT 100 */
TE_connect(
    "comm-type = telnet;"
    "host-name = alfa;"
    "telnet-port = 992;"
    "terminal-id = ;"
    "set-window-size = true;"
    "security-type = ssl;"
    "ssl-type = tls1;"
    "terminal-type = vt100;"
    "terminal-model = vt100;"
    "login-command-file = ;"
    "terminal-setup-file = ;"
    , 60000);
if (TE_errno != TE_SUCCESS)
    return -1;
```

挿入された **TE_connect** ステートメントの後には if ステートメントが続きます。これは，再生中に **TE_connect** 関数の実行が成功したかどうかを調べます。

注：Vuser スクリプトの中でサーバへの接続（**TE_connect**）を複数記録しないでください。

以上で，ターミナルの設定と接続の手順が完了します。これで，Vuser スクリプトへの一般的なユーザ・アクションの記録を開始する準備が整います。次項ではこの方法について説明します。

一般的なユーザ・アクションの記録

設定手順を記録した後，一般的なユーザ・アクションまたはビジネス・プロセスを実行します。これらのプロセスは Vuser スクリプトの **Actions** セクションに記録します。Vuser スクリプトの反復を複数回実行するときに繰り返されるのは，スクリプトの **Actions** セクションだけです。反復の設定の詳細については，『**第 1 巻 ー VuGen の使用**』の「実行環境の設定」を参照してください。

セッションを記録する際，VuGen によって記録されるのはテキストのストローク（打鍵内容）であり，テキストではありません。したがって，コマンドを直接入力する代わりに PowerTerm ウィンドウにコピーして貼り付けることはお勧めできません。

ユーザ・アクションを記録するには，次の手順を実行します。

1 既存の RTE Vuser スクリプトを開き，［**セクション**］ボックスで［**アクション**］をクリックします。

2 引き続き，ターミナル・エミュレータで一般的なユーザ・アクションを実行します。入力中，VuGen は対応するステートメントを生成し，それらを Vuser スクリプトに挿入します。必要ならば，記録されたステートメントをスクリプトの記録中に編集できます。

注：標準設定では，VuGen は連続するキーストロークの合間に，対応する **TE_type** 関数が生成されるまで最大 5 秒待ちます。この待ち時間を変更する場合は，『**第 1 巻 ー VuGen の使用**』の第 16 章「選択したプロトコルの記録オプション」を参照してください。

一般的なユーザ・アクションの記録が完了したら，引き続きログオフ手順を記録します。次項ではこの方法について説明します。

ログオフ手順の記録

Vuser のログオフ・プロセスは Vuser スクリプトの **vuser_end** セクションに記録します。**vuser_end** セクションは，スクリプトの反復を多数回実行するときに繰り返されません。反復の設定の詳細については，『**第 1 巻 ー VuGen の使用**』の「実行環境の設定」を参照してください。

ログオフ手順を記録するには，次の手順を実行します。

1 前の項で説明した方法で，一般的なユーザ・アクションの実行と記録を必ず済ませておきます。

2 VuGen のメイン・ウィンドウの［**セクション**］ボックスで，**vuser_end** をクリックします。

3 ログオフ手順を実行します。VuGen によって手順がスクリプトの **vuser_end** セクションに記録されます。

4［記録］ツールバーで，［**記録停止**］をクリックします。VuGen のメイン・ウィンドウに，記録されたすべてのステートメントが表示されます。

5［**上書き保存**］ボタンをクリックして，記録されたセッションを保存します。［名前を付けて保存］ダイアログ・ボックスが表示されます（新規 Vuser スクリプトの場合のみ)。スクリプト名を指定します。記録されたスクリプトは，VuGen のメイン・ウィンドウで手作業で編集できます。

ターミナル・エミュレータへの入力

2 つの TE Vuser 関数を使用すると，Vuser が PowerTerm ターミナル・エミュレータに文字を「入力」できるようになります。

- **TE_type** は，文字をターミナル・エミュレータに送ります。記録中，ターミナル・ウィンドウへのキーボード入力に対して，**TE_type** 関数が VuGen によって自動的に生成されます。詳細については，775 ページ「TE_type 関数の使用」を参照してください。
- **TE_typing_style** は，Vuser の入力速度を決めます。**TE_typing_style** 関数を Vuser スクリプトに挿入することによって，入力のスタイルを手作業で定義できます。詳細については，777 ページ「入力スタイルの設定」を参照してください。または，実行環境の設定から入力スタイルを設定することもできます。詳細については『**第 1 巻 ― VuGen の使用**』の，第 24 章「選択したプロトコルの実行環境の設定」を参照してください。

注：RTE Vuser スクリプトを記録するときは，マウスを使用してターミナル・エミュレータ・ウィンドウ内のカーソルの位置を変更しないでください。これらのカーソル移動は記録されません。

TE_type 関数の使用

スクリプトを記録すると，VuGen によってすべてのキーボード入力が記録され，対応する **TE_type** 関数が生成されます。実行中は，**TE_type** 関数によって，整形された文字列がターミナル・エミュレータに送られます。

キーボード入力は通常のテキスト文字列として定義されます（空白も含まれます）。次に例を示します。

```
TE_type ("hello, world");
```

2 文字以上の入力キー名は，文字 **k** で始まる識別子によって表され，大なり記号と小なり記号（＜＞）で囲まれます。

たとえば，次の関数は，Return キーと，その後の Control キーと y キーの入力を表しています。

```
TE_type("<kReturn><kControl-y>");
```

その他の例としては，<kF1>，<kUp>，<kF10>，<kHelp>，<kTab> などがあります。

キー名を調べるには，キーの操作を記録した後，記録されたステートメントで名前を調べます。

注：TE_type ステートメントを（記録するのではなく）プログラムするときは，『**Online Function Reference**』（英語版）（［**ヘルプ**］＞［**関数リファレンス**］）に記載されているキー定義を使用してください。

TE_type のタイムアウト値の設定

システムが X SYSTEM モード（または入力禁止モード）にある間，Vuser が **TE_type** ステートメントを送信しようとした場合，その Vuser は X SYSTEM モードが終了するまで入力を待たされます。システムが TE_XSYSTEM_TIMEOUT ミリ秒を超えて X SYSTEM モードを保持した場合は，**TE_type** 関数が TE_TIMEOUT エラーを返します。

TE_XSYSTEM_TIMEOUT の値は **TE_setvar** を使用して設定できます。TE_XSYSTEM_TIMEOUT の標準値は 30 秒です。

Vuser に対する事前入力の許可

場合によっては，システムが X SYSTEM モード（または入力禁止モード）にある間も，Vuser に対してキーストロークの送信を許可したいことがあります。たとえば，Vuser に対して Break キーの押下を許可することができます。システムが X SYSTEM モードにある間も Vuser に対してキーストロークの送信を許可するには，TE_ALLOW_TYPEAHEAD 変数を使用します。

事前入力を禁止するには，TE_ALLOW_TYPEAHEAD を 0 に設定し，事前入力を許可するには，0 以外の値に設定します。TE_ALLOW_TYPEAHEAD の値を設定するには **TE_setvar** を使用します。標準設定では，TE_ALLOW_TYPEAHEAD は 0 に設定されており，X SYSTEM モード中はキーストロークが送られません。

TE_type 関数とその規約の詳細については，『**Online Function Reference**』（英語版）（［**ヘルプ**］＞［**関数リファレンス**］）を参照してください。

入力スタイルの設定

RTE Vuser に対しては，FAST と HUMAN の 2 つの入力スタイルを設定できます。FAST スタイルでは，Vuser からターミナル・エミュレータへの入力ができるだけ高速に実行されます。HUMAN スタイルでは，文字を入力するたびに Vuser が一時停止します。このため，人間のユーザによるキーボードでの入力が，Vuser によって，より正確にエミュレートされます。

入力スタイルは **TE_typing_style** 関数を使用して設定します。**TE_typing_style** 関数の構文は次のとおりです。

int **TE_typing_style (**char ***style);**

ここで，**style** には FAST または HUMAN を指定できます。標準設定の入力スタイルは HUMAN です。HUMAN の入力スタイルを選択する場合は，次の形式になります。

HUMAN, **delay** [,**first_delay**]

delay には，キーストローク間の間隔（ミリ秒）を指定します。省略可能なパラメータ first_delay には，文字列の 1 文字目を入力する前の待ち時間（ミリ秒）を指定します。次に例を示します。

```
TE_typing_style ("HUMAN, 100, 500");
TE_type ("ABC");
```

前述の場合は，Vuser は文字 A を入力する前に 0.5 秒待ちます。次に，文字「B」を入力する前に 0.1 秒待ち，その次に文字「C」を入力する前にさらに 0.1 秒待ちます。

TE_typing_style 関数とその規約の詳細については，『**Online Function Reference**』（英語版）（［**ヘルプ**］ ＞ ［**関数リファレンス**］）を参照してください。

TE_typing_style 関数を使用する以外に，実行環境の設定を使用して入力スタイルを設定することもできます。詳細については，第 24 章「RTE の実行環境の設定」を参照してください。

一意のデバイス名の生成

APPC などの一部のプロトコルでは，システムにログオンするターミナルごとに一意のデバイス名が必要になります。実行環境の設定を使用して，**TE_connect** 関数から Vuser ごとに 8 文字の一意のデバイス名を生成し，その名前を使用して接続するよう指定することができます。この方法では一意性の条件は満たされますが，システムによっては，デバイス名が特定の形式に従う必要があるなど，さらに別の条件が必要になる場合があります。詳細については，『**第 1 巻 ー VuGen の使用**』の第 22 章「実行環境の設定」を参照してください。

TE_connect 関数が Vuser のシステムへの接続の際に使用するデバイス名について，その形式を定義するには，**RteGenerateDeviceName** 関数を Vuser スクリプトに追加します。この関数のプロトタイプは次のとおりです。

```
void RteGenerateDeviceName(char buf[32])
```

デバイス名は **buf** に書き込まれます。

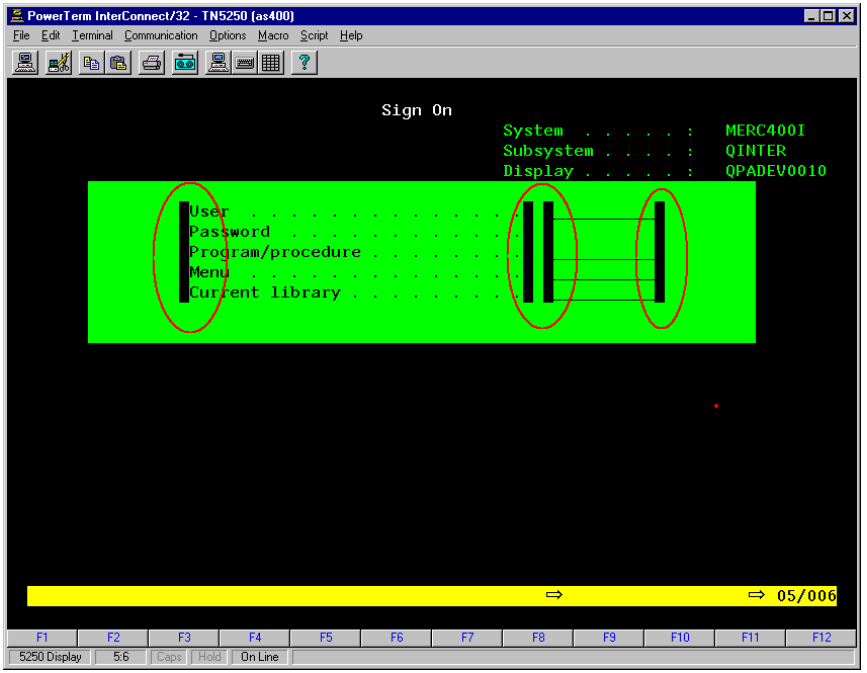
RteGenerateDeviceName 関数が Vuser スクリプト内に存在する場合は，新しいデバイス名が必要になるたびに，Vuser からこの関数が呼び出されます。**RteGenerateDeviceName** 関数がスクリプト内で定義されておらず，一意のデバイス名が必要になった場合は，**TE_connect** 関数によって必要な名前が生成されます。

次の例では，RteGenerateDeviceName 関数によって「TERMx」という形式の一意のデバイス名が生成されます。最初の名前が TERM0 となり，以降 TERM1，TERM2，という具合になります。

```
RteGenerateDeviceName(char buf[32])
{
    static int n=0;
    sprintf(buf, "TERM%d", n);
    n=n+1;
}
```

フィールド区分文字

一部のターミナル・エミュレータでは，各フィールドの先頭と末尾を示すために区分文字が使用されます。これらの区分文字は画面上では空白として現れ，目に見えません。次に示すターミナル・エミュレータでは，フィールド区分文字を見えるようにするために画面の中央部分の色を反転させています。区分文字を楕円で囲んで示してあります。



TE_wait_text，**TE_get_text**，および **TE_find_text** 関数は，画面の指定の部分にある文字を識別することによって動作します。指定の部分の中にフィールド区分文字がある場合は，その文字を空白または ASCII 文字として識別できます。識別の方法を指定するには，TE_FIELD_CHARS システム変数を使用します。TE_FIELD_CHARS は 0 または 1 に設定できます。

- 0 の場合，フィールド区分文字の位置にある文字は空白として返されます。
- 1 の場合，フィールド区分文字の位置にある文字は ASCII コード（ASCII 0 または ASCII 1）として返されます。

標準設定では，TE_FIELD_CHARS は 0 に設定されています。

TE_FIELD_CHARS の値の取得および設定には，**TE_getvar** および **TE_setvar** 関数を使用します。

第 53 章

RTE ー 同期

RTE Vuser スクリプトで同期化関数を使用すると，Vuser がターミナル・エミュレータに送信する入力をサーバからの応答と同期させることができます。

本章の内容

Vuser スクリプトの同期化について

テスト対象のシステムによっては，Vuser がターミナル・エミュレータに送信する入力をサーバからの応答と同期させることが必要な場合があります。入力を同期化するには，次のアクションを実行する前にスクリプトの実行を一時停止し，システムからの合図を待つように Vuser を設定します。たとえば，実際のユーザが次の一連のキーストロークを銀行アプリケーションに送信する場合を考えます。

1 「1」と入力して，銀行アプリケーションのメニューから「金融情報」を選択します。

2 「必要な情報を選択してください。」というメッセージが表示されたら，「3」と入力して，メニューから「ダウ・ジョーンズ工業平均株価」を選択します。

3 詳細なレポートが画面に表示されたら，「5」と入力して，銀行アプリケーションを終了します。

この例では，実際のユーザが各ステップで入力する前に画面上の合図を待っているので，銀行アプリケーションへの入力は同期化されています。この合図は，特定のメッセージが画面上に表示されること，または画面上のすべての情報が安定した状態になることのいずれかです。

Vuser の入力は，TE 同期化関数 **TE_wait_sync**，**TE_wait_text**，**TE_wait_silent**，および **TE_wait_cursor** を使って同じ方法で同期化できます。これらの関数は，ターミナル・ウィンドウに入力する実際のユーザをエミュレートし，次のコマンドを入力する前にサーバの応答を待ちます。

TE_wait_sync 関数は，ブロック・モード（IBM）ターミナルを同期化する場合にのみ使用されます。ほかの TE 同期化関数は，キャラクタ・モード（VT）ターミナルを同期化する場合に使用されます。

RTE Vuser スクリプトを記録すると，**TE_wait_sync**，**TE_wait_text**，および **TE_wait_cursor** の各ステートメントが自動的に生成され，スクリプトに挿入されます。挿入される同期化関数を指定するには，VuGen の記録オプションを使用します。

注：Vuser スクリプトの **Vuser_end** セクションには，同期化ステートメントを挿入しないでください。Vuser の実行はいつでも中止できるため，**Vuser_end** セクションがいつ実行されるかは予測できません。

ブロック・モード（IBM）ターミナルの同期化

TE_wait_sync 関数は，ブロック・モード（IBM）ターミナルを操作する RTE Vuser を同期化するために使用されます。ブロック・モード・ターミナルでは，システムが入力禁止モードになっていることを示すために「X SYSTEM」というメッセージが表示されます。システムが入力禁止モードになっているときは，ターミナル・エミュレータがサーバからデータが転送されるのを待っているため，入力できません。

ブロック・モード・ターミナルでスクリプトを記録すると，標準設定では「X SYSTEM」メッセージが表示されるたびに **TE_wait_sync** 関数が生成され，スクリプトに挿入されます。**TE_wait_sync** 関数を自動的に挿入するかどうかを指定するには，VuGen の記録オプションを使用します。

Vuser スクリプトを実行すると，**TE_wait_sync** 関数はシステムが X SYSTEM モードになっているかどうかを確認します。システムが X SYSTEM モードになっている場合，**TE_wait_sync** 関数はスクリプトの実行を一時停止します。「X SYSTEM」メッセージが画面から削除されると，スクリプトの実行が再開されます。

注：TE_wait_sync 関数は，IBM ブロック・モード・ターミナル（5250 および 3270）エミュレータでのみ使用できます。

一般的に，**TE_wait_sync** 関数は，すべてのブロック・モード・ターミナル・エミュレータに対して適切な同期化機能を提供します。ただし，特定の状況で **TE_wait_sync** 関数がうまく機能しない場合は，**TE_wait_text** 関数を挿入することで同期化機能を拡張できます。**TE_wait_text** 関数の詳細については，787 ページ「画面上のテキスト表示の待機」，および『**Online Function Reference**』（英語版）（［**ヘルプ**］＞［**関数リファレンス**］）参照してください。

TE_wait_sync 関数の構文は次のとおりです。

```
TE_wait_sync();
```

次のスクリプト・セグメントでは，Vuser が「QUSER」というユーザ名と「HPLAB」というパスワードでログオンします。Vuser は，Enter を押してサーバにログイン情報を送信します。サーバから応答を待っている間は，ターミナル・エミュレータに X SYSTEM メッセージが表示されます。

TE_wait_sync ステートメントによって，Vuser はスクリプトの次の行を実行する前に，サーバがログイン要求に応答するまで（X SYSTEM メッセージが削除されるまで）待機します。

```
TE_type("QUSER");
lr_think_time(2);
TE_type("<kTab>HPLAB");
lr_think_time(3);
TE_type("<kEnter>");
TE_wait_sync();
....
```

X SYSTEM メッセージの表示に合わせて **TE_wait_sync** 関数がスクリプトの実行を一時停止している間，Vuser はシステムを監視し続け，X SYSTEM メッセージが消えるのを待ちます。X SYSTEM メッセージが消えないまま同期化のタイムアウトに達すると，**TE_wait_sync** 関数はエラー・コードを返します。標準のタイムアウトは 60 秒です。

TE_wait_sync による同期化のタイムアウトを設定するには，次の手順を実行します。

1 ［**仮想ユーザ**］＞［**実行環境の設定**］を選択します。［実行環境設定］ダイアログ・ボックスが表示されます。

2 ［実行環境設定］ツリーの［**RTE:RTE**］ノードを選択します。

3 ［**X- システムの同期化**］の［**タイムアウト**］ボックスに，値（秒単位）を入力します。

4 ［**OK**］をクリックして，［実行環境設定］ダイアログ・ボックスを閉じます。

Vuser は，**TE_wait_sync** 関数を実行した後，ターミナルが X SYSTEM モードに入らなくなるまで待機します。ターミナルが X SYSTEM モードから戻ると，Vuser はターミナルが完全に安定した（システムが X SYSTEM に戻らない）ことを確認するため，少しの間システムを監視し続けます。この確認ができた場合にのみ，**TE_wait_sync** 関数が終了し，Vuser がスクリプトの実行を再開できるようになります。Vuser が X SYSTEM モードから戻った後のシステムを監視し続ける時間を，安定時間と呼びます。標準の安定時間は 1000 ミリ秒です。

システムに次のような動作が見られる場合は，安定時間を増やす必要があります。

システムが X SYSTEM モードから戻るときに，一部のシステムでは X SYSTEM モードとの間を短時間に何度か「行き来」してからでないとシステムが安定しません。システムが 1 秒以上 X SYSTEM モードにならず，その後で X SYSTEM モードに戻った場合，**TE_wait_sync** 関数はシステムが安定したとみなします。Vuser がシステムに情報を入力しようとすると，システムはキーボード・ロック・モードに移行します。

逆に，システムが X SYSTEM から戻るときにモード間の行き来がない場合は，安定時間を標準値の 1 秒より短く設定できます。

TE_wait_sync 関数の安定時間を変更するには，次の手順を実行します。

1 ［**仮想ユーザ**］＞［**実行環境の設定**］を選択します。［実行環境設定］ダイアログ・ボックスが表示されます。

2 ［**RTE:RTE**］ノードを選択します。

3 ［**X- システムの同期化**］の［**安定時間**］ボックスに，値（ミリ秒単位）を入力します。

4 ［**OK**］をクリックして，［実行環境設定］ダイアログ・ボックスを閉じます。

TE_wait_sync 関数の詳細については，『**Online Function Reference**』（英語版）（［**ヘルプ**］＞［**関数リファレンス**］）を参照してください。

システムが X SYSTEM モードに移行するたびにその継続時間を記録するように VuGen を設定できます。その場合は，次のスクリプト・セグメントに示すように，各 TE_wait_sync 関数の後に TE_wait_sync_transaction 関数が挿入されます。

```
TE_wait_sync();
TE_wait_sync_transaction("syncTrans1");
```

TE_wait_sync_transaction 関数は，「default」という名前のトランザクションを作成します。これによって，ターミナル・エミュレータがシナリオ実行の間にサーバからの応答を待った時間を分析できます。
TE_wait_sync_transaction ステートメントを生成して挿入するかどうかを指定するには，記録オプションを使用します。

TE_wait_sync_transaction ステートメントを挿入するように VuGen を設定するには，次の手順を実行します。

1 ［**仮想ユーザ**］＞［**記録オプション**］を選択します。［記録オプション］ダイアログ・ボックスが表示されます。

2 ［**X- システム用自動トランザクションを作成する**］オプションを選択して，［**OK**］をクリックします。

キャラクタ・モード（VT）ターミナルの同期化

キャラクタ・モード（VT）ターミナルでは，3 種類の同期化を使用できます。選択できる同期化の種類は，次の要因で決まります。

- ターミナル・エミュレータで実行するアプリケーションの設計
- 同期化する具体的なアクション

特定位置でのカーソル表示の待機

VT タイプのターミナルで推奨される同期化方法は，カーソル同期化です。カーソル同期化は，スクロール形式や TTY 形式のアプリケーションよりも，全画面形式やフォーム形式のアプリケーションで特に有効です。

カーソル同期化では，**TE_wait_cursor** 関数を使用します。RTE Vuser スクリプトを実行すると，**TE_wait_cursor** 関数は画面上の指定された位置にカーソルが表示されるまでスクリプトの実行を一時停止するように Vuser に指示します。指定された位置にカーソルが表示されることは，アプリケーションがターミナル・エミュレータから次の入力を受け付ける準備ができたことを意味します。

TE_wait_cursor 関数の構文は次のとおりです。

int **TE_wait_cursor (**int **col**, int **row**, int **stable**, int **timeout);**

スクリプトの実行中，**TE_wait_cursor** 関数はカーソルが **col** と **row** で指定された位置に移動するまで待機します。

stable パラメータには，カーソルが指定された位置で安定している時間（ミリ秒）を指定します。VuGen を使ってスクリプトを記録する場合，**stable** は標準で 100 ミリ秒です。クライアント・アプリケーションが **timeout** パラメータで指定された時間内に安定しない場合，この関数は TIMEOUT を返します。VuGen を使ってスクリプトを記録する場合，**timeout** は標準で TIMEOUT の値（90 秒）に設定されます。**stable** パラメータの値と **timeout** パラメータの値は，どちらも記録されたスクリプトを直接編集することによって変更できます。

次のステートメントは，カーソルが安定した状態で 3 秒間待機します。カーソルが 10 秒以内に安定しない場合，この関数は TIMEOUT を返します。

```
TE_wait_cursor(10, 24, 3000, 10);
```

TE_wait_cursor 関数の詳細については，『**Online Function Reference**』（英語版）（［**ヘルプ**］＞［**関数リファレンス**］）を参照してください。

スクリプトの記録中に TE_wait_cursor ステートメントを自動的に生成してスクリプトに挿入するように VuGen を設定できます。VuGen によって自動的に生成された TE_wait_cursor ステートメントの例を次に示します。

```
TE_wait_cursor(7, 20, 100, 90);
```

記録中に TE_wait_cursor ステートメントを自動的に生成してスクリプトに挿入するように VuGen を設定するには，次の手順を実行します。

1 ［**仮想ユーザ**］＞［**記録オプション**］を選択します。［記録オプション］ダイアログ・ボックスが表示されます。

2 ［**自動同期化コマンドを作成**］の［**カーソル**］チェック・ボックスを選択して，［**OK**］をクリックします。

画面上のテキスト表示の待機

VT ターミナル・エミュレータ上で RTE Vuser を同期化する場合は，テキスト同期化を使用します。テキスト同期化では，**TE_wait_text** 関数を使用します。スクリプトの実行中，**TE_wait_text** 関数はスクリプトの実行を一時停止し，スクリプトの実行を再開する前に特定の文字列がターミナル・ウィンドウに表示されるのを待ちます。テキスト同期化は，カーソルが画面上のあらかじめ定義された領域に常に表示されるとは限らないアプリケーションで有効です。

注：テキスト同期化は，キャラクタ・モード（VT）ターミナルでの使用を目的としていますが，IBM ブロック・モード・ターミナルでも使用できます。ただし，ブロック・モード・ターミナルでは自動的なテキスト同期化を使用しないでください。

TE_wait_text 関数の構文は次のとおりです。

```
int TE_wait_text (char *pattern, int timeout, int col1, int row1, int col2, int row2,
                  int *retcol, int *retrow, char *match);
```

この関数は，col1，row1，col2，row2 で定義される四角形の内部に **pattern** に一致するテキストが表示されるまで待機します。パターンに一致するテキストは **match** に返され，実際の行位置と列位置は **retcol** と **retrow** に返されます。**pattern** が表示されないまま **timeout** に達すると，この関数はエラー・コードを返します。**pattern** には正規表現を含めることができます。正規表現の使用方法の詳細については，『**Online Function Reference**』（英語版）を参照してください。**pattern** と **timeout** 以外のパラメータは，すべて省略可能です。

pattern に空の文字列を指定すると，この関数は四角形の内部に任意のテキストが表示されたときにタイムアウトを待ちます。テキストが表示されなかったときは，すぐに戻ります。

pattern が表示されなかった場合，この関数はエミュレータが安定する（再描画を終了する，または新しい文字を表示しなくなる）のを，TE_SILENT_SEC システム変数と TE_SILENT_MILLI システム変数で定義された時間だけ待ちます。これは，結果的にターミナルを安定させ，実際のユーザをエミュレートすることになります。

ターミナルが TE_SILENT_TIMEOUT で定義された時間内に安定しない場合は，スクリプトの実行が再開されます。この関数は成功を示す 0 を返すと同時に，テキストの表示後にターミナルが安定しなかったことを示すために TE_errno 変数を設定します。

TE_SILENT システム変数の値を変更および取得するには，**TE_getvar** 関数および **TE_setvar** 関数を使用します。詳細については，『**Online Function Reference**』（英語版）（［**ヘルプ**］＞［**関数リファレンス**］）を参照してください。

次の例では，Vuser が自分の名前を入力し，アプリケーションの応答を待ちます。

```
/* TE_wait_text の変数を宣言する */
int ret_row;
int ret_col;
char ret_text [80];

/* ユーザ名を入力する */
TE_type ("John");

/* 窓口の応答を待つ */
TE_wait_text ("Enter secret code:", 30, 29, 13, 1, 13, &ret_col, &ret_row,ret_text);
```

スクリプトの記録中に **TE_wait_text** ステートメントを自動的に生成してスクリプトに挿入するように VuGen を設定できます。

記録中に TE_wait_text ステートメントを自動的に生成してスクリプトに挿入するように VuGen を設定するには，次の手順を実行します。

1 ［**仮想ユーザ**］＞［**記録オプション**］を選択します。［記録オプション］ダイアログ・ボックスが表示されます。

2 ［**自動同期化コマンドを作成**］の［**プロンプト**］チェック・ボックスを選択して，［**OK**］をクリックします。

VuGen によって自動的に生成された TE_wait_text ステートメントの例を次に示します。この関数は，「keys」という文字列が画面上の任意の場所に表示されるのを最大 20 秒間待ちます。VuGen が TE_wait_text 関数を生成するときは，省略可能なパラメータはすべて省略されます。

```
TE_wait_text("keys", 20);
```

ターミナルの安定の待機

カーソル同期化もテキスト同期化もうまく機能しない場合は，「安定同期化」を使用してスクリプトを同期化できます。「安定同期化」では，Vuser はターミナル・エミュレータが安定するのを指定された時間だけ待ちます。エミュレータは，指定された期間内にサーバからの入力を受信しなかったときに安定したとみなされます。

注：安定同期化は，カーソル同期化とテキスト同期化がどちらもうまく機能しない場合にのみ使用します。

ターミナルが安定するまで待機するようにスクリプトを設定するには，**TE_wait_silent** 関数を使用します。ターミナルが安定している期間を指定します。ターミナルが指定された期間だけ安定していれば，**TE_wait_silent** 関数は，アプリケーションがターミナル画面へのテキストの表示を終了し，画面が安定したとみなします。

関数の構文は次のとおりです。

```
int TE_wait_silent (int sec, int milli, int timeout);
```

TE_wait_silent 関数は，ターミナル・エミュレータが安定するのを **sec**（秒）と **milli**（ミリ秒）で指定された期間だけ待ちます。エミュレータは，サーバからの入力を受信しなかったときに安定したとみなされます。**timeout** で指定された時間内にエミュレータが安定しない（文字の受信が終了しない）場合，この関数はエラーを返します。

たとえば，次のステートメントは画面が安定した状態で 3 秒間待機します。10 秒経過しても画面が安定しない場合，この関数はエラーを返します。

```
TE_wait_silent (3, 0, 10);
```

詳細については，『**Online Function Reference**』（英語版）（［**ヘルプ**］＞［**関数リファレンス**］）を参照してください。

第 54 章

RTE ― ターミナル画面からのテキストの読み取り

RTE Vuser は，ターミナル・エミュレータのユーザ・インタフェースからテキストを読み取り，そのテキストを使ってさまざまなタスクを実行できます。

本章の内容

ターミナル画面からのテキストの読み取りについて

RTE Vuser がターミナル画面からテキストを読み取るために使用できる Vuser 関数が複数用意されています。これらの関数は **TE_find_text** と **TE_get_text_line** で，ターミナル・エミュレータが正常に応答していることを確認するため，またはスクリプトのロジックを拡張するために使用できます。

TE_find_text と **TE_get_text_line** は，記録後，RTE Vuser スクリプトに手作業で直接挿入できます。

画面上のテキストの検索

TE_find_text 関数は，画面上のテキスト行を検索します。関数の構文は次のとおりです。

int **TE_find_text (**char ***pattern**, int **col1**, int **row1**, int **col2**, int **row2**,
int ***retcol**, int ***retrow**, char ***match);**

この関数は，**col1**，**row1**，**col2**，**row2** で定義される四角形の内部で，**pattern** に一致するテキストを検索します。パターンに一致するテキストは **match** に返され，実際の行位置と列位置は **retcol** と **retrow** に返されます。検索は左上隅から行われます。複数の文字列が **pattern** に一致した場合は，左上隅に最も近い文字列が返されます。

pattern には正規表現を含めることができます。正規表現の使用方法の詳細については，『**Online Function Reference**』（英語版）を参照してください。

TE_find_text ステートメントは，Vuser スクリプトに手作業で入力する必要があります。**TE_find_text** 関数の構文の詳細については，『**Online Function Reference**』（英語版）（［**ヘルプ**］＞［**関数リファレンス**］）を参照してください。

画面からのテキストの読み取り

TE_get_text_line 関数は，画面上の指定された領域からテキスト行を読み取ります。関数の構文は次のとおりです。

char ***TE_get_text_line (**int **col**, int **row**, int **width**, char ***text);**

この関数は，テキスト行をターミナル画面から **text** バッファにコピーします。行の最初の文字は **col** と **row** で定義します。行の最後の文字の列座標は **width** で定義します。画面から読み取られたテキストは，**text** バッファに返されます。行内にタブや空白が含まれる場合は，それらに相当する数の空白が返されます。

また，**TE_get_cursor_position** 関数を使ってターミナル画面上のカーソルの現在位置を取得することもできます。**TE_get_line_attribute** 関数は，テキスト行内の文字の書式設定（太字，下線など）を返します。

TE_get_text_line ステートメントは，Vuser スクリプトに手作業で入力する必要があります。**TE_get_text_line** 関数の構文の詳細については，『**Online Function Reference**』（英語版）（［**ヘルプ**］ > ［**関数リファレンス**］）を参照してください。

第 55 章

メール・サービス・プロトコル

VuGen では，プロトコル・レベルでいくつかのメール・サービスをテストできます。VuGen は，メール送信，およびメール・サーバに対して実行されるほとんどの標準的な操作をエミュレートします。

本章の内容

メール・サービス Vuser スクリプトの作成について

メール・サービス・プロトコルは，メールの表示や送信など，電子メール・クライアントで作業しているユーザをエミュレートします。サポートされているメール・サービスは次のとおりです。

- IMAP（Internet Messaging）
- MAPI（MS Exchange）
- POP3（Post Office Protocol）
- SMTP（Simple Mail Transfer Protocol）

メール・サービス Vuser スクリプトでは，記録と再生の両方がサポートされています。ただし，MAPI では再生だけがサポートされています。

メール・プロトコルの 1 つを使用してアプリケーションを記録するとき，VuGen によってメール・クライアントのアクションをエミュレートする関数が生成されます。Vuser スクリプトの作成に使用するプログラミング言語（C または Visual Basic）を指定できます。詳細については，『**第 1 巻 － VuGen の使用**』の「スクリプト生成オプションの設定」を参照してください。通信に複数のプロトコルが使われている場合は，両方を記録できます。複数のメール・プロトコルを同時に，または 1 つのメール・プロトコルを HTTP または WinSock と一緒に記録できます。マルチ・プロトコルの指定の詳細については，『**第 1 巻 － VuGen の使用**』の「VuGen を使った記録」を参照してください。

メール・サービス関数はすべて，対で用意されています。一方がグローバル・セッション用で，もう一方で特定のメール・セッションを指定します。たとえば，**imap_logon** はグローバルに IMAP サーバにログオンしますが，**imap_logon_ex** は特定のセッションのために IMAP サーバにログオンします。

メール・サービス Vuser スクリプトの概要

本項では，VuGen を使用したメール・サービス Vuser スクリプトの開発プロセスの概要を説明します。

メール・サービス Vuser スクリプトを作成するには，次の手順を実行します。

1 **VuGen を使って基本スクリプトを作成します。**

VuGen を起動して，1 つまたは複数のメール・プロトコルを対象とする Vuser スクリプトを新規作成します。

2 **VuGen を使用して基本スクリプトを記録します。（MAPI を除く）。**

記録するアプリケーションを選択します。アプリケーションを対象に標準的な操作を実行します。詳細については，『**第 1 巻 － VuGen の使用**』の「VuGen を使った記録」を参照してください。

MAPI では，記録はサポートされていません。その代わりに空の MAPI スクリプトを作成し，**mapi** 関数を手作業で挿入します。使用例は，『**Online Function Reference**』（英語版）（［**ヘルプ**］＞［**関数リファレンス**］）を参照してください。

3 スクリプトを拡張します。

スクリプトに，トランザクション，ランデブー・ポイント，および制御フロー構造を挿入して，スクリプトを拡張します。

詳細については，『**第 1 巻 － VuGen の使用**』の「Vuser スクリプトの拡張」を参照してください。

4 パラメータを定義します（任意）。

スクリプトに記録された固定値の代わりにパラメータを定義します。固定値をパラメータで置き換えることにより，同じビジネス・プロセスを，異なる値で何度も繰り返すことができます。

詳細については，『**第 1 巻 － VuGen の使用**』の「パラメータの作成」を参照してください。

5 ステートメントを相関させます（任意）。

ステートメントの相関によって，あるビジネス・プロセスの結果を以降の別のビジネス・プロセスで使用できるようになります。

詳細については，『**第 1 巻 － VuGen の使用**』の「ステートメントの相関」を参照してください。

6 実行環境を設定します。

実行環境を設定することによって，スクリプト実行時の Vuser の動作を制御します。設定には，ループ，ログ，タイミング情報などがあります。

詳細については，『**第 1 巻 － VuGen の使用**』の「実行環境の設定」を参照してください。

7 VuGen でスクリプトを実行します。

VuGen でスクリプトを保存して実行し，正しく動作することを確認します。

詳細については，『**第 1 巻 － VuGen の使用**』の「スタンドアロン・モードでの Vuser スクリプトの実行」を参照してください。

スクリプトを作成した後，スクリプトを自分の環境（LoadRunner シナリオ，Performance Center 負荷テスト，または Business Process Monitor プロファイル）に組み込みます。詳細については，**HP LoadRunner Controller**，**HP Performance Center**，または **HP Business Availibility Center** のドキュメントを参照してください。

IMAP スクリプトについて

IMAP Vuser スクリプト関数は，Internet Mail Application Protocol（IMAP）を記録します。IMAP 関数の名前には，**imap** というプレフィックスが付いています。

これらの関数の構文情報の詳細については，『**Online Function Reference**』（英語版）（［**ヘルプ**］＞［**関数リファレンス**］）を参照してください。

次の例では，**imap_create** 関数を使って，Products，Solutions，および FAQs という新しいメールボックスを作成します。

```
Actions()
{
    imap_logon("ImapLogon",
            "URL=imap://johnd:letmein@exchange.mycompany.com",
            LAST);

    imap_create("CreateMailboxes",
            "Mailbox=Products",
            "Mailbox=Solutions",
            "Mailbox=FAQs",
            LAST);

    imap_logout( );

    return 1;
}
```

MAPI スクリプトについて

MAPI Vuser スクリプト関数は，MS Exchage サーバを対象とする操作を記録します。MAPI 関数の名前には，**mapi** というプレフィックスが付いています。

注：MAPI スクリプトを実行するには，スクリプトを実行するマシンでメール・プロファイルを定義する必要があります。たとえば，Outlook Express をインストールして標準設定のメール・クライアントとして設定し，メール・アカウントを作成します。あるいは，Microsoft Outlook をインストールして標準設定のメール・クライアントとして設定し，メール・アカウントを作成してメール・プロファイルを作成します。Microsoft Outlook でメール・プロファイルを作成するには，［**設定**］＞［**コントロール パネル**］＞［**メール**］＞［**プロファイルの表示**］を選択し，メール・プロファイルを追加します。

これらの関数の構文情報の詳細については，『**Online Function Reference**』（英語版）（［**ヘルプ**］＞［**関数リファレンス**］）を参照してください。

次の例では，**mapi_send_mail** 関数を使用して，MS Exchange サーバを通じて付せんを送信します。

```
Actions()
{
    mapi_logon("Logon",
            "ProfileName=John Smith",
            "ProfilePass=Tiger",
            LAST);
    // 付せんメッセージを送信
    mapi_send_mail("SendMail",
            "To=user1@techno.merc-int.com",
            "Cc=user0002t@techno.merc-int.com",
            "Subject= < GROUP > : < VUID > @ < DATE > ",
            "Type=Ipm.StickyNote",
            "Body=Please update your profile today.",
            LAST);

    mapi_logout( );
    return 1;
}
```

POP3 スクリプトについて

POP3 Vuser スクリプト関数は，POP3（Post Office Protocol）を使用してアクションをエミュレートします。各 POP3 関数は **pop3** というプレフィックスが付いています。

これらの関数の構文情報の詳細については，『**Online Function Reference**』（英語版）（［**ヘルプ**］＞［**関数リファレンス**］）を参照してください。

次の例では，**pop3_retrieve** 関数を使って POP3 サーバから 5 つのメッセージを取得しています。

```
Actions()
{
pop3_logon("Login","
        URL=pop3://user0004t:my_pwd@techno.merc-int.com",
        LAST);

// サーバ上のメッセージをすべて表示し，値を取得する
totalMessages = pop3_list("POP3", LAST);

// 取得した値を表示する（pop3_list 関数を使って表示することもできる）
lr_log_message("There are %d total messages on the server.¥r¥n¥r¥n", totalMessages);

// 5 つのメッセージをサーバから削除せずに取得する
pop3_retrieve("POP3", "RetrieveList=1:5", "DeleteMail=false", LAST);
pop3_logoff();
    return 1;
}
```

SMTP スクリプトについて

SMTP Vuser スクリプト関数は，SMTP トラフィックをエミュレートします。SMTP 関数の名前には，**smtp** というプレフィックスが付いています。

これらの関数の構文情報の詳細については，『**Online Function Reference**』（英語版）（［**ヘルプ**］＞［**関数リファレンス**］）を参照してください。

次の例では，**smtp_send_mail** 関数を使って，SMTP メール・サーバ techno を通じてメール・メッセージを送信しています。

```
Actions()
{
    smtp_logon("Logon",
        "URL=smtp://user0001t@techno.merc-int.com",
            "CommonName=Smtp Test User 0001",
        NULL);

    smtp_send_mail("SendMail",
                    "To=user0002t@merc-int.com",
                    "Subject=MIC Smtp: Sample Test",
                    "MAILOPTIONS",
                    "X-Priority: 3",
                    "X-MSMail-Priority: Medium",
                    "X-Mailer: Microsoft Outlook Express 5.50.400¥r¥n",
                    "X-MimeOLE: By Microsoft MimeOLE V5.50.00¥r¥n",
                    "MAILDATA",
                    "MessageText="
                        "Content-Type: text/plain;¥r¥n"
                        "¥tcharset=¥"iso-8859-1¥"¥r¥n"
                        "Test,¥r¥n"
                        "MessageBlob=16384",
                NULL);

    smtp_logout();

    return 1;
}
```

第 56 章

Tuxedo プロトコル

VuGen を使用して，Tuxedo クライアント・アプリケーションと Tuxedo アプリケーション・サーバの間の通信を記録します。その結果生成されるスクリプトを Tuxedo Vuser スクリプトといいます。

本章の内容

Tuxedo Vuser スクリプトについて

Tuxedo アプリケーションを記録すると，VuGen は，記録されたアクションを記述する LRT 関数を生成します。これらの関数は，Tuxedo クライアントとサーバの間の通信をエミュレートします。各 LRT 関数は，プレフィックス **lrt** で始まります。

プレフィックス **lrt** に加え，**tp**，**tx**，**F** といった補助プレフィックスを使用する関数もあります。これらの補助プレフィックスは，実際の Tuxedo 関数と同様に，関数の型を示します。補助プレフィックス **tp** は，Tuxedo クライアントの tp セッションを示します。たとえば，**lrt_tpcall** は，サービス要求を送信して応答を待ちます。補助プレフィックス **tx** は，グローバルな tx セッションを表します。たとえば，**lrt_tx_begin** はグローバルなトランザクションを開始します。補助プレフィックス **F** は，FML バッファに関連する関数を表します。たとえば，**lrt_Finitialize** は既存のバッファを初期化します。

補助プレフィックスのない関数は，標準 C 関数をエミュレートします。たとえば，**lrt_strcpy** は，C 関数 **strcpy** と同じように文字列をコピーします。

記録されたスクリプトは，VuGen のメイン・ウィンドウで表示および編集ができます。セッション中に記録された LRT 関数が VuGen ウィンドウに表示されるので，ネットワークの動作状況を視覚的に追跡できます。

記録前の作業

記録を行う前に，Tuxedo ディレクトリ %TUXDIR%¥bin がパスに含まれていることを確認します。

VuGen の再起動後に環境変数を変更している場合，VuGen は，環境変数の現在の値ではなく元の値を記録することがあります。

不整合を防ぐため，Tuxedo アプリケーションを記録する前には VuGen を再起動します。

Tuxedo Vuser スクリプトの概要

本項では，VuGen を使った Tuxedo Vuser スクリプトの作成プロセスの概要を説明します。

Tuxedo Vuser スクリプトの作成は，次の手順を実行します。

1 **VuGen を使用して基本スクリプトを記録します。**

VuGen を起動して，新しい Vuser スクリプトを作成します。Vuser のタイプとして，**Tuxedo 6**（Tuxedo Version 6.x の記録用）または **Tuxedo** （Tuxedo Version 7.x の記録用）を指定します。記録するアプリケーションを選択します。選択したアプリケーションの典型的な操作を記録します。

詳細については，『**第 1 巻 ― VuGen の使用**』の「VuGen を使った記録」を参照してください。

2 **スクリプトを拡張します。**

スクリプトに，トランザクション，ランデブー・ポイント，および制御フロー構造を挿入して，スクリプトを拡張します。

詳細については，『**第 1 巻 ― VuGen の使用**』の「Vuser スクリプトの拡張」を参照してください。

3 **パラメータを定義します（任意）。**

スクリプトに記録された固定値の代わりにパラメータを定義します。固定値をパラメータで置き換えることにより，同じビジネス・プロセスを，異なる値で何度も繰り返すことができます。

詳細については，『**第 1 巻 ― VuGen の使用**』の「パラメータの作成」を参照してください。

4 **ステートメントを相関させます（任意）。**

ステートメントの相関によって，あるビジネス・プロセスの結果を以降の別のビジネス・プロセスで使用できるようになります。

詳細については，『**第 1 巻 ― VuGen の使用**』の「ステートメントの相関」を参照してください。

5 **実行環境を設定します。**

実行環境を設定することによって，スクリプト実行時の Vuser の動作を制御します。設定には，ループ，ログ，タイミング情報などがあります。

詳細については，『**第 1 巻 ー VuGen の使用**』の「実行環境の設定」を参照してください。

6 **VuGen でスクリプトを実行します。**

VuGen でスクリプトを保存して実行し，正しく動作することを確認します。

詳細については，『**第 1 巻 ー VuGen の使用**』の「スタンドアロン・モードでの Vuser スクリプトの実行」を参照してください。

スクリプトを作成した後，スクリプトを自分の環境（LoadRunner シナリオ，Performance Center 負荷テスト，または Business Process Monitor プロファイル）に組み込みます。詳細については，**HP LoadRunner Controller**，**HP Performance Center**，または **HP Business Availibility Center** のドキュメントを参照してください。

Tuxedo Vuser スクリプトについて

VuGen は，一般的な Tuxedo セッションを記録して Tuxedo 固有の関数を生成します。これらの関数ではプレフィックスに **lrt** を使用します。たとえば，**lrt_tpacall** はサーバ要求を送信します。

LRT 関数は，バッファ操作，クライアント / サーバ・セッション，通信，環境変数，エラー処理，トランザクション処理，および相関関数というカテゴリに分類されます。

スクリプトに任意の関数を手作業でプログラミングすることもできます。LRT 関数の構文と例については，『**Online Function Reference**』（英語版）（［**ヘルプ**］＞［**関数リファレンス**］）を参照してください。

セッション記録後，記録されたコードを VuGen の組み込みエディタに表示できます。スクリプトをスクロールして，アプリケーションで生成された Tuxedo ステートメントの表示や，サーバによって返されたデータの検査ができます。VuGen ウィンドウには，記録された Tuxedo セッションに関する重要な情報が表示されます。メイン・ウィンドウにスクリプトを表示すると，VuGen が動作状況を記録したシーケンスを確認できます。

次の例では，VuGen がクライアントのアクションを Tuxedo の銀行アプリケーションに記録しています。クライアントは，銀行の口座を開き，必要な詳細のすべてを指定するアクションを実行しました。クライアントが開始残高としてゼロを指定したところで，セッションは中止されています。

```
lrt_abort_on_error();
lr_think_time(65);
tpresult_int = lrt_tpbegin(30, 0);
data_0 = lrt_tpalloc("FML", "", 512);
lrt_Finitialize((FBFR*)data_0);

/* データ・バッファ data_0 に新規口座情報を入力する */
lrt_Fadd_fld((FBFR*)data_0, "name=BRANCH_ID", "value=8",
LRT_END_OF_PARMS);
lrt_Fadd_fld((FBFR*)data_0, "name=ACCT_TYPE", "value=C",
LRT_END_OF_PARMS);
lrt_Fadd_fld((FBFR*)data_0, "name=MID_INIT", "value=Q", LRT_END_OF_PARMS);
lrt_Fadd_fld((FBFR*)data_0,"name=PHONE","value=123-456-7890",
            LRT_END_OF_PARMS);

lrt_Fadd_fld((FBFR*)data_0, "name=ADDRESS", "value=1 Broadway
            New York, NY 10000", LRT_END_OF_PARMS);
lrt_Fadd_fld((FBFR*)data_0,"name=SSN","value=111111111", LRT_END_OF_PARMS);

lrt_Fadd_fld((FBFR*)data_0,"name=LAST_NAME",
"value=Doe",LRT_END_OF_PARMS);

lrt_Fadd_fld((FBFR*)data_0,"name=FIRST_NAME",
"value=BJ",LRT_END_OF_PARMS);

lrt_Fadd_fld((FBFR*)data_0, "name=SAMOUNT",
"value=0.00",LRT_END_OF_PARMS);

/* 新規口座を開く */
tpresult_int = lrt_tpcall("OPEN_ACCT", data_0, 0, &data_0, &olen_2, 0);
lrt_tpabort(0);
lrt_tpcommit(0);
lrt_tpfree(data_0);
lrt_tpterm();
```

Tuxedo スクリプトでのパラメータの使用

『**第 1 巻 ― VuGen の使用**』の「パラメータの作成」に記載した方法で，パラメータを Tuxedo スクリプトで定義できます。Tuxedo スクリプトには，"name=..." または "value=..." タイプの文字列が含まれます。パラメータの定義は，文字列内の等号（=）以降の部分のみを対象にできます。次に例を示します。

```
lrt_Fadd_fld((FBFR*)data_0,"name=PHONE","value={parameter_1}",
            LRT_END_OF_PARMS);
```

注：一般に，**lrt_save_parm** を使用して文字配列の一部をパラメータに保存することをお勧めします。文字配列内の特定の文字列の位置に関する情報を保存する場合は，**lrt_save_searched_string** を使用します。PeopleSoft Vuser の場合には，**lrt_save_searched_string** を使用することをお勧めします。Peoplesoft サーバから返される応答バッファは，記録時と再生時でサイズが異なることが多いためです。

Tuxedo スクリプトの実行

Tuxedo アプリケーションの記録または実行の際に問題が発生した場合は，Tuxedo アプリケーションが VuGen なしで動作し，環境変数が正しく定義されていることを確認します。詳細については，次の「Tuxedo バッファ・データの表示」を参照してください。Tuxedo 変数の設定または変更をした後は，VuGen とアプリケーションを再起動して，変更を有効にする必要があります。アプリケーションが 16 ビットの場合は，NTVDM プロセスを強制終了する必要もあります。

実行時に問題が発生した場合は，サーバ側の Tuxedo ログ・ファイルでエラー・メッセージを確認します。標準設定では，このファイルは，環境変数 APPDIR で示されるディレクトリにあります。ファイル名は，ULOG.mmddyy の形式です（mmddyy は，現在の月，日，年を示してします)。1999 年 3 月 12 日のファイルは ULOG.031299 となります。このファイルの標準設定の場所は，サーバ上の環境変数 ULOGPFX を設定することで変更できます。ログ・ファイルはクライアント側にもあります。ULOGPFX 変数で場所が変更されていなければ，カレント・ディレクトリに置かれます。

Tuxedo バッファ・データの表示

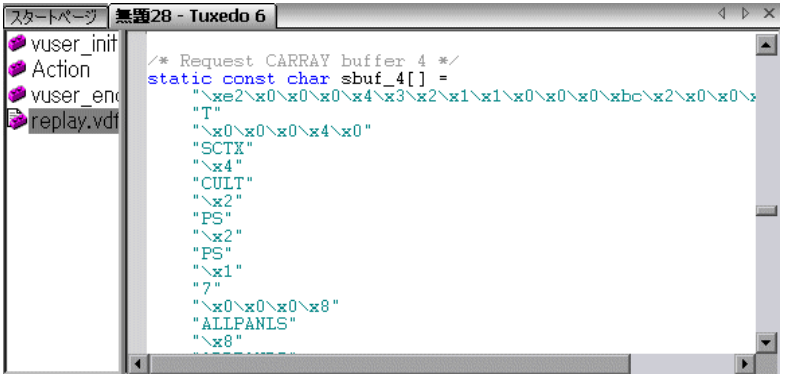
VuGen を使用して Tuxedo Vuser スクリプトを作成すると，アクションはスクリプトの 3 つのセクション **vuser_init**，**Actions**，**vuser_end** に記録されます。

受け取った，または転送されたデータはデータ・バッファに保管されますが，データ・バッファは非常に大きくなることがあります。スクリプトの可読性を高めるために，実際のデータは C ファイルではなく外部ファイルに保管されます。データ転送が発生すると，データは外部ファイルから一次バッファにコピーされます。

この外部ファイルは **replay.vdf** と呼ばれ，すべての一時バッファの内容を含んでいます。バッファの内容は連続したレコードとして保管されます。レコードはデータが送信されたか受信されたかを示す識別子とバッファ記述子によってマークされます。LRT 関数は，バッファ記述子を使用してデータにアクセスします。

VuGen を使用して，左側の表示枠のツリー・ビュー内で **replay.vdf** ファイルを選択することで，データ・ファイルの内容を表示できます。

データ・ファイルを表示するオプションは，Tuxedo スクリプトでは標準で使用できます。



Tuxedo Vuser の環境設定の定義

次の項では，Windows および UNIX プラットフォームで動作する Tuxedo Vuser のシステム変数の設定について説明します。システム変数は，NT の場合は［コントロール パネル］の［システム］ダイアログ・ボックスで，UNIX の場合は .cshrc または .login ファイルで定義します。

TUXDIR	Tuxedo ソースのルート・ディレクトリ
FLDTBLDIR	FML バッファ情報を含むディレクトリのリスト。Windows では，ディレクトリの名前をセミコロンで区切ります。UNIX プラットフォームでは，ディレクトリの名前をコロンで区切ります。
FIELDTBLS	FML バッファ情報を含むファイルのリスト。Windows と UNIX のどちらのプラットフォームでも，ファイル名はカンマで区切ります。

次に例を示します。

```
SET FLDTBLDIR=%TUXDIR%¥udataobj;%TUXDIR%¥APPS¥WS (PC)
SET FIELDTBLS=bankflds,usysflds (PC)
setenv FLDTBLDIR $TUXDIR/udataobj:$TUXDIR/apps/bankapp (Unix)
setenv FIELDTBLS bank.flds,Usysflds (Unix)
```

実行中に，Tuxedo/WS ワークステーションの拡張を使用して，Tuxedo クライアントの次のシステム変数を定義する必要があります。

WSNADDR	ワークステーション・リスナ・プロセスのネットワーク・アドレスを指定します。これによって，クライアント・アプリケーションが Tuxedo にアクセスできるようになります。WSNADDR ステートメントで複数のアドレスを定義するには，各アドレスをカンマで区切る必要があります。
WSDEVICE	ネットワークにアクセスするデバイスを指定します。一部のネットワーク・プロトコルについては，この変数を定義する必要はありません。

次に例を示します。

```
SET WSNADDR=0x0002ffffc7cb4e4a (PC)
setenv WSNADDR 0x0002ffffc7cb4e4a (Unix)
setenv WSDEVICE /dev/tcp (Unix)
```

Tuxedo アプリケーションのデバッグ

一般に，Tuxedo 6.x 以前を使用するアプリケーションを記録する場合は **Tuxedo 6** を，Tuxedo 7.1 以降を使用するアプリケーションを記録する場合は **Tuxedo** を使用します。

Tuxedo アプリケーションの記録または再生の際に問題が発生した場合，またはスクリプトが lrt_tpinitialize への呼び出しを行っていない場合は，アプリケーションでどの DLL が使用されているかをカスタマー・サポートに問い合わせてください。

アプリケーションが **libwsc.dll** ではなく **wtuxws32.dll** を使用している場合は，カスタマー・サポートから記録を行えるようにするパッチを入手します。

Tuxedo スクリプトの相関

VuGen は，Tuxedo アプリケーションで記録される Vuser スクリプトの相関をサポートしています。相関ステートメントでバッファの一部を保存し，それを後続のステートメントで使用することにより，ステートメント間をリンクすることができます。

ステートメントを相関させるには，記録されたスクリプトを VuGen エディタ内で次の LRT 関数の 1 つを使用して変更する必要があります。

- **lrt_save[32]_fld_val** は，FML または FML32 バッファの現在の値（「name= < NAME >」または「id= < ID >」形式の文字列）をパラメータに保存します。
- **lrt_save_parm** は，文字配列の一部（STRING バッファや CARRAY バッファなど）をパラメータに保存します。
- **lrt_save_searched_string** は，バッファ内で文字列を検索し，その文字列に関連するバッファの一部をパラメータに保存します。

これらの関数の構文の詳細については，『**Online Function Reference**』（英語版）を参照してください。

FML および FML32 バッファの相関

lrt_save_fld_val または **lrt_save32_fld_val** を使って，FML バッファ，FML32 バッファの内容を保存します。

lrt_save_fld_val を使用したステートメントを相関するには，次の手順を実行します。

1 スクリプト内の，現在の FML（または FML32）バッファの内容を保存する場所に，**lrt_save_fld_val** ステートメントを挿入します。

lrt_save_fld_val (fbfr, "name", occurrence, "**param_name**");

2 パラメータを参照します。

保存したバッファの内容で記録されている値を置き換える **lrt** ステートメントを見つけます。記録されている値のすべてのインスタンスを，中括弧で囲まれたパラメータ名で置換します。

次の例では，銀行口座を開き，口座番号を **account_id** パラメータに格納します。

```
/* data_0 バッファに新規口座情報を入力する */
data_0 = lrt_tpalloc("FML", "", 512);
lrt_Finitialize((FBFR*)data_0);
lrt_Fadd_fld((FBFR*)data_0, "name=BRANCH_ID", "value=1",
LRT_END_OF_PARMS);
lrt_Fadd_fld((FBFR*)data_0, "name=ACCT_TYPE", "value=S",
LRT_END_OF_PARMS);
...

LRT_END_OF_PARMS);
lrt_Fadd_fld((FBFR*)data_0, "name=LAST_NAME", "value=Doe", ...);
lrt_Fadd_fld((FBFR*)data_0, "name=FIRST_NAME", "value=John", ...);
lrt_Fadd_fld((FBFR*)data_0, "name=SAMOUNT", "value=234.12", ...);

/* 新規口座を開き，新規口座番号を保存する */
tpresult_int = lrt_tpcall("OPEN_ACCT", data_0, 0,&data_0, &olen_2, 0);
lrt_abort_on_error();
lrt_save_fld_val((FBFR*)data_0, "name=ACCOUNT_ID", 0, "account_id");

/* 最初のクエリの結果を預金バッファに入力する */
lrt_Finitialize((FBFR*)data_0);
lrt_Fadd_fld((FBFR*)data_0, "name=ACCOUNT_ID", "value={account_id}",
LRT_END_OF_PARMS);
lrt_Fadd_fld((FBFR*)data_0, "name=SAMOUNT", "value=200.11", ...);
```

前述の例では，アカウント ID が ACCOUNT_ID というフィールド名で表されています。記録時にフィールドがフィールド名ではなく ID 番号で表されるシステムもあります。

フィールド ID による相関は，次のように行います。

```
lrt_save_fld_val((FBFR*)data_0, "id=8302", 0, "account_id");
```

文字列の相関

lrt_save_parm または **lrt_save_searched_string** を使って，キャラクタ文字列を相関させます。

- 一般に，**lrt_save_parm** を使用して文字配列の一部をパラメータに保存することをお勧めします。
- 文字配列内の特定の文字列の位置に関する情報を保存する場合は，**lrt_save_searched_string** を使用します。PeopleSoft Vuser の場合には，**lrt_save_searched_string** を使用することをお勧めします。Peoplesoft サーバから返される応答バッファは，記録時と再生時でサイズが異なることが多いためです。

相関させる値の決定

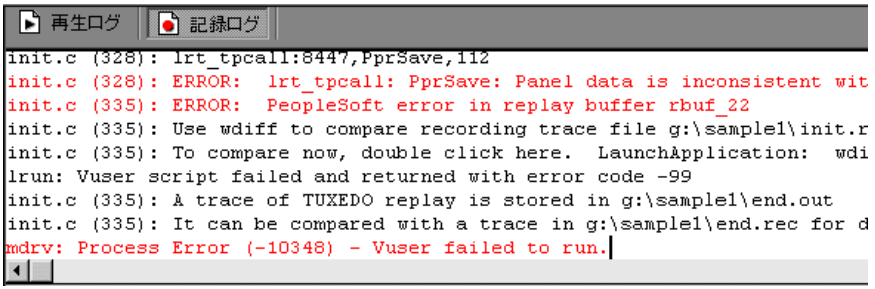
CARRAY バッファで作業を行っている場合，VuGen は Wdiff ユーティリティを使って比較できるログ・ファイルを生成します（記録中であれば拡張子は **.rec** に，再生中であれば拡張子は **.out** になります）。記録ログと再生ログの相違を調べて，CARRAY バッファのどの部分を相関させる必要があるか判断できます。

ログ・ファイルを比較するには，次の手順を実行します。

1 ［**表示**］＞［**出力**］を選択して，スクリプトの実行ログと記録ログを表示します。

2 ［再生ログ］タブを調べます。

エラー・メッセージの後には，**Use wdiff to compare** という句で始まるステートメントが付いています。



3 実行ログのステートメントをダブルクリックして，**Wdiff** ユーティリティを起動します。

WDiff が開き，記録ファイルと再生ファイルの間の相違が黄色で強調表示されます。Wdiff ユーティリティの詳細については，『**第 1 巻 ― VuGen の使用**』の「ステートメントの相関」を参照してください。

lrt_save_parm を使用したステートメントを相関するには，次の手順を実行します。

相関させる値を決定したら，**lrt_save_parm** を使って，文字配列の一部分（STRING または CARRAY バッファなど）をパラメータに保存できます。

1 スクリプト内の，現在のバッファの内容を保存する場所に **lrt_save_parm** ステートメントを挿入します。

lrt_save_parm (buffer, offset, length, "**param_name**");

2 **replay.vdf** ファイル内で，保存されたバッファの内容で置き換えたいデータを見つけます。

VuGen のメイン・ウィンドウの［データ ファイル］ボックスで **replay.vdf** ファイルを選択して，バッファの内容を表示します。

3 値のすべてのインスタンスを中括弧で囲まれたパラメータ名で置換します。

次の例では，CARRAY バッファの従業員 ID を，後で使用できるように保存する必要があります。記録された値は，出力されているとおり "G001" です。

```
lrt_tpcall:227, PprLoad, 1782
Reply Buffer received.
...
123 "G001"
126 "..."
134 "Claudia"
```

オフセット 123 を使用して，「PprLoad」と 227 バイトを送信する要求バッファの直後に **lrt_save_parm** を挿入します。

```
/* CARRAY buffer 57 を要求する */
    lrt_memcpy(data_0, buf_143, 227);
    tpresult_int = lrt_tpcall("PprLoad",
        data_0, 227, &data_1, &olen, TPSIGRSTRT);
    lrt_save_parm(data_1, 123, 9, "empid");
```

replay.vdf ファイル内で，記録された値 "G001" をパラメータ **empid** に置き換えます。

```
char buf_143[] =   "¥xf5¥x0¥x0¥x0¥x4¥x3¥x2¥x1¥x1¥x0¥x0¥x0¥xbc¥x2¥x0¥x0¥x0¥x0¥x0¥x0"
    "X"
"¥x89¥x0¥x0¥x0¥xb¥x0"
    "SPprLoadReq"
    "¥xff¥x0¥x10¥x0¥x0¥x4¥x3¥x6"
    "{empid}" // G001
    "¥x7"
    "Claudia"
    "¥xe"
    "LAST_NAME_SRCH"
...
```

この関数は，FML バッファ内で文字配列の一部を保存するときにも使用できます。次の例では，電話番号が文字配列で，市外局番は最初の 3 文字です。はじめに，**lrt_save_fld_val** ステートメントが電話番号をパラメータ **phone_num** に保存します。**lrt_save_parm** ステートメントは，**lr_eval_string** を使って電話番号を文字配列に変換し，市外局番を **area_code** という名前のパラメータに保存します。

```
lrt_save_fld_val((FBFR*)data_0, "name=PHONE", 0, "phone_num");
lrt_save_parm(lr_eval_string("{phone_num}"), 0, 3, "area_code");
lr_log_message("The area code is %s¥n", lr_eval_string("{area_code}"));
```

lrt_save_searched_string を使用したステートメントを相関するには，次の手順を実行します。

lrt_save_searched_string を使用して，バッファ内で文字列を検索し，文字列に関連するバッファの一部をパラメータに保存します。

1 スクリプト内の現在のバッファの一部を保存する場所に **lrt_save_searched_string** ステートメントを挿入します。

```
lrt_save_searched_string (buffer, buf_size, occurrence, string, offset,
                          length, "param_name");
```

offset は文字列の先頭からのオフセットです。

2 **replay.vdf** ファイル内で，保存されたバッファの内容で置き換えたいデータを見つけます。

VuGen のメイン・ウィンドウの［データ ファイル］ボックスで **replay.vdf** ファイルを選択して，バッファの内容を表示します。

3 値のすべてのインスタンスを中括弧で囲まれたパラメータ名で置換します。

次の例では，Certificate を後で使用できるようにパラメータに保存しています。**lrt_save_searched_string** 関数は，指定された olen バッファの 16 バイトをパラメータ **cert1** に保存します。保存される文字列のバッファ内の位置は，文字列 "SCertRep" の最初の出現から 9 バイトです。

このアプリケーションは，バッファのヘッダ情報が記録環境によって異なる場合に役立ちます。

署名は「SCertRep」の最初の出現より 9 バイト後に来ますが，この文字列より前の情報の長さは不定です。

```
/* CARRAY buffer 1 を要求する */
lrt_memcpy(data_0, sbuf_1, 41);
lrt_display_buffer("sbuf_1", data_0, 41, 41);
data_1 = lrt_tpalloc("CARRAY", "", 8192);
tpresult_int = lrt_tpcall("GetCertificate",
    data_0,
    41,
    &data_1,
    &olen,
    TPSIGRSTRT);

/* CARRAY buffer 1 を再生する */
lrt_display_buffer("rbuf_1", data_1, olen, 51);
lrt_abort_on_error();

lrt_save_searched_string(data_1, olen, 0, "SCertRep", 9, 16, "cert1");
```

第 57 章

Real Player および Media Player のプロトコル

インターネットで音声 / 映像コンテンツを配信するストリーミング・メディアが急成長しています。ストリーミング・メディアの基本的な考え方は，音声 / 映像コンテンツを配信するときに，エンド・ユーザに先にファイル全体をダウンロードする手間をかけさせないようにしようというものです。ストリーミングは，サーバからコンテンツをストリームとして連続して送出させ，それをクライアントが受け取るごとに表示する仕組みになっています。

RealPlayer は，ストリーミング・コンテンツを表示するアプリケーションです。

VuGen を使って，RealPlayer プロトコルでやり取りするクライアント・アプリケーションとサーバ間の通信を記録することができます。記録の結果として作成されるスクリプトを，「Real Vuser スクリプト」と呼びます。

本章の内容

注：Media Player（MMS）プロトコルをマルチメディア・メッセージング・サービス（MMS）プロトコルと混同しないでください。詳細については，834 ページ「MMS（マルチメディア・メッセージング・サービス）Vuser スクリプト」を参照してください。

ストリーミング・データ仮想ユーザ・スクリプトの記録について

ストリーミング・データ・プロトコルにより，メディアまたはストリーミング・データ・ファイルを再生するユーザをエミュレートできます。

ストリーミング・データ・プロトコルを使用してアプリケーションを記録すると，VuGen は記録時のアクションを記述する関数を生成します。RealPlayer セッションの場合，VuGen はプレフィックス **lreal** の付いた関数を生成します。Media Player セッションの場合，VuGen はプレフィックス **mms** の付いた関数を使用します。なお，Media Player 用の mms 関数では記録はサポートされておらず，再生のみ可能です。

ストリーミング・データ Vuser スクリプトの概要

本項では，VuGen を使用して RealPlayer および Media Player のストリーミング・データ Vuser スクリプトを作成する工程の概要を説明します。

RealPlayer または Media Player Vuser スクリプトを作成するには，次の手順を実行します。

1 **VuGen を使用して基本スクリプトを記録します （Real のみ）**

VuGen を起動して，新しい Vuser スクリプトを作成します。記録するアプリケーションを選択し，アプリケーションを対象とする一般的な操作を記録します。詳細については，『**第 1 巻 ― VuGen の使用**』の「VuGen を使った記録」を参照してください。

2 **スクリプトを拡張します。**

スクリプトに，トランザクション，ランデブー・ポイント，および制御フロー構造を挿入して，スクリプトを拡張します。

詳細については，『**第 1 巻 ― VuGen の使用**』の「Vuser スクリプトの拡張」を参照してください。

3 **パラメータを定義します（任意）。**

スクリプトに記録された固定値の代わりにパラメータを定義します。固定値をパラメータで置き換えることにより，同じビジネス・プロセスを，異なる値で何度も繰り返すことができます。

詳細については，『**第 1 巻 ― VuGen の使用**』の「パラメータの作成」を参照してください。

4 ステートメントを相関させます（任意）。

ステートメントの相関によって，あるビジネス・プロセスの結果を以降の別のビジネス・プロセスで使用できるようになります。

詳細については，『**第 1 巻 ― VuGen の使用**』の「ステートメントの相関」を参照してください。

5 実行環境を設定します。

実行環境を設定することによって，スクリプト実行時の Vuser の動作を制御します。設定には，ループ，ログ，タイミング情報などがあります。

詳細については，『**第 1 巻 ― VuGen の使用**』の「実行環境の設定」を参照してください。

6 VuGen でスクリプトを実行します。

VuGen でスクリプトを保存して実行し，正しく動作することを確認します。

詳細については，『**第 1 巻 ― VuGen の使用**』の「スタンドアロン・モードでの Vuser スクリプトの実行」を参照してください。

スクリプトを作成した後，スクリプトを自分の環境（LoadRunner シナリオ，Performance Center 負荷テスト，または Business Process Monitor プロファイル）に組み込みます。詳細については，**HP LoadRunner Controller**，**HP Performance Center**，または **HP Business Availibility Center** のドキュメントを参照してください。

RealPlayer LREAL 関数の使用

RealPlayer プロトコルを使用してクライアントとサーバ間の通信をエミュレートするために作成された関数を，Real Player 関数と呼びます。各 Real Player 関数には，プレフィックス **lreal** が付いています。

VuGen は，Real Player セッション中，本項に列挙されている LREAL 関数のほぼすべてを自動的に記録します。スクリプトに任意の関数を手作業でプログラミングすることもできます。

lreal_play 関数の形式の例を次に示します。

```
int lreal_play (int miplayerID, long mulTimeToPlay);
```

クリップを最後まで再生するには，**mulTimeToPlay** 値に任意の負の値を使用します。クリップの再生を指定した時間だけ継続する場合は，時間をミリ秒単位で指定します。**miplayerID** は，RealPlayer インスタンスの一意の ID を示します。

LREAL 関数の詳細については，『**Online Function Reference**』（英語版）（［**ヘルプ**］＞［**関数リファレンス**］）を参照してください。

Media Player MMS 関数の使用

Media Player の MMS プロトコルを使用してクライアントとサーバ間の通信をエミュレートするために作成された関数を，MMS 仮想ユーザ関数と呼びます。各 MMS 仮想ユーザ関数には，プレフィックス **mms** が付いています。

重要：Media Player 関数を再生するには，**wmload.asf** というファイルを Windows Media サーバ・マシンに配置する必要があります。VuGen マシンは，**mms://** ＜**サーバ名**＞ **/wmload.asf** を使用してアクセスできる必要があります。この ASF ファイルは，**wmload.asf** と名前を変更した任意のメディア・ファイルでかまいません。

MMS 関数はすべて，グローバル・セッションと特定のセッション用にペアになっています。たとえば，**mms_close** は Media Player をグローバルに閉じ，**mms_close_ex** は特定のセッションでの Media Player を閉じます。

典型的な関数である **mms_play** は，次の形式で使用します。

```
int mms_play (char message, ＜属性のリスト＞ , LAST);
```

次の例では，**mms_play** 関数は，**asf** ファイルを継続時間を変えて再生します。

```
// 10 秒再生する
mms_play("Welcome","URL=mms://server/welcome.asf","duration=10",LAST);

// 5 秒待機後，クリップを最後まで再生する
mms_play ("Welcome","URL=mms://server/welcome.asf",
"duration=-1",
"starttime=5",LAST);
```

これらの関数の構文情報の詳細については，『**Online Function Reference**』（英語版）（［**ヘルプ**］＞［**関数リファレンス**］）を参照してください。

第 58 章

ワイヤレス・プロトコル

VuGen を使用して標準的なワイヤレス・セッションを記録することによって，ワイヤレス Vuser スクリプトを生成できます。スクリプトを実行すると，生成された Vuser によって，ツールキットまたは携帯電話と Web サーバ（または WAP 用ゲートウェイ）との間のユーザ・アクションがエミュレートされます。

本章の内容

WAP プロトコルについて

WAP（Wireless Application Protocol）は，モバイル・ユーザがワイヤレス・デバイスを使って瞬時に情報およびサービスにアクセスすることを可能にする，世界標準のオープンな規格です。

WAP プロトコルは，ワイヤレス・モバイル・ターミナル用に最適化された WML と呼ばれる新しい標準言語を使い，マイクロ・ブラウザによるシン・クライアントを規定しています。WML とは，XML を必要最小限まで簡素化したドキュメント記述言語です。

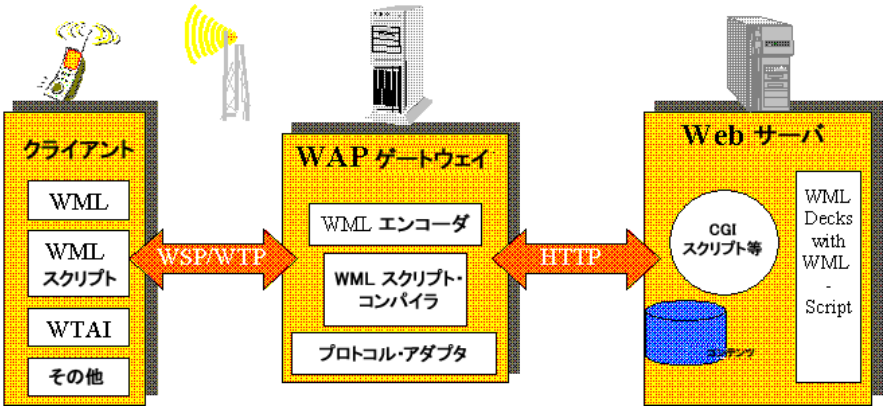
WAP ではさらに，次の条件を満たすプロキシ・サーバを規定しています。

- ワイヤレス・ネットワークと有線のインターネットの間のゲートウェイとして機能する。
- プロトコル変換機能がある。
- ワイヤレス受信のためにデータ転送を最適化する。

WAP アーキテクチャは WWW と非常によく似ています。すべてのコンテンツがインターネットの標準形式に似た形式で記述されます。コンテンツは WWW の領域では標準プロトコルで，ワイヤレスの領域（Wireless Session Protocol）では HTTP に似た最適化されたプロトコルを使って送信されます。WAP コンテンツはすべて WWW で標準的に使われている URL を使って指定します。

WAP では，オーサリングやパブリッシングの方法など，多くの部分が WWW 規格を使用しています。その一方で，ワイヤレス・デバイスおよびネットワークの特徴に合わせて，いくつかの WWW 規格が強化されています。「呼制御」および「メッセージング」などのモバイル・ネットワーク・サービスをサポートするための拡張機能が追加されています。WAP は，モバイル・ターミナルのメモリ容量や CPU 処理能力の制約を考慮しています。また，帯域幅の狭いネットワークおよびレイテンシ時間の長いネットワークにも対応します。

WAP では，モバイル・クライアントとの間で送受信されるデータのエンコードとデコードを行うゲートウェイが存在することが前提となっています。クライアントに配信されるコンテンツをエンコードする目的は，クライアントにワイヤレスで送信されるデータのサイズを最小化することと，クライアントがデータを処理する際の負荷を軽減することです。このようなゲートウェイの機能は発信元サーバに追加することも可能ですが，次の図に示すように専用ゲートウェイに置くこともできます。



WAP ツールキット

Nokia，Ericsson，Phone.com などの主要通信企業は，WAP アプリケーションおよびサービスの開発を支援する「ツールキット」を開発しています。この WAP ツールキットは，モバイル・ターミナル用のインターネット・サービスおよびコンテンツの開発環境を提供します。開発者は，WAP ツールキットを使用して，PC ベースの電話シミュレータによるアプリケーションの開発，テスト，デバッグ，および実行ができます。また，ツールキットから HTTP 接続または WAP ゲートウェイを経由して WAP サイトをブラウズすることができます。

携帯電話からは，WSP プロトコルを使ってゲートウェイと通信します。一方，ツールキットはゲートウェイと通信することも，サーバと直接通信することもできます。VuGen は，ツールキットで設定されている通信モード（WSP または HTTP）を自動的に検出します。ゲートウェイへのトラフィックを調べたい場合は，WSP モードで記録します。サーバおよびコンテンツ・プロバイダを検査したい場合は，HTTP モードでツールキット・セッションを記録して，ゲートウェイはバイパスすることができます。

VuGen は，ユーザ・セッションをエミュレートするためにユーザ定義の API 関数を使用します。ほとんどの関数は HTTP プロトコルを使用した標準の Web プロトコル関数です。いくつかの WAP 関数では，WAP Vuser 固有のアクションをエミュレートします。サポートされている関数の一覧については，830 ページ「ワイヤレス Vuser 関数の使用法」を参照してください。

ワイヤレス Vuser スクリプトの作成の概要

ワイヤレス Vuser スクリプトは，ワイヤレス・ブラウザを使用するユーザをエミュレートします。PC ベースの電話シミュレータ（ツールキット）を使用してユーザのブラウズ操作を記録します。その後，利用可能なテスト用マシンに，数百の Vuser を分散配置し，各 Vuser から API を通じてサーバにアクセスします。これにより，多数のユーザによる負荷の下でサーバのパフォーマンスを測定できます。

本項では，VuGen を使ったワイヤレス Vuser スクリプトの開発プロセスの概要を説明します。

ワイヤレス・スクリプトを作成するには，次の手順を実行します。

1 VuGen を使って新しいスクリプトを作成します。

［**ファイル**］＞［**新規作成**］を選択するか，［**新規作成**］ボタンをクリックして，シングル・プロトコル・モードまたはマルチ・プロトコル・モードで新しいスクリプトを作成します。

新規スクリプトの作成の詳細については，『**第 1 巻 ― VuGen の使用**』の「VuGen を使った記録」を参照してください。

2 VuGen を使ってアクションを記録します。

ツールキット・セッションでのアクションを記録します。VuGen は，自動的にツールキットの設定を検出し，記録時にその設定を使用します。

記録方法の詳細については，『**第 1 巻 ― VuGen の使用**』の「VuGen を使った記録」を参照してください。

3 Vuser スクリプトを拡張します。

トランザクション，ランデブー・ポイント，および制御フロー構造をスクリプトに挿入することによって，Vuser スクリプトを拡張します。

詳細については，『**第 1 巻 ― VuGen の使用**』の「Vuser スクリプトの拡張」を参照してください。

4 パラメータを定義します（任意）。

Vuser スクリプトに記録された固定値に対してパラメータを定義します。固定値をパラメータで置き換えることにより，同じビジネス・プロセスを，異なる値で何度も繰り返すことができます。詳細については，『**第 1 巻 － VuGen の使用**』の「パラメータの作成」を参照してください。

5 実行環境を設定します。

実行環境を設定することによって，スクリプト実行時の Vuser の動作を制御します。これらの設定には，実行論理，ペース設定，ログ，思考遅延時間，パフォーマンスの設定，ゲートウェイの設定が含まれます。

一般的な実行環境の設定の詳細については，『**第 1 巻 － VuGen の使用**』の「実行環境の設定」を参照してください。

インターネット・プロトコルの一般的な実行環境設定の詳細については，『**第 1 巻 － VuGen の使用**』の第 23 章「ネットワーク実行環境の設定」を参照してください。

実行環境の設定の WAP 専用の設定の詳細については，『**第 1 巻 － VuGen の使用**』の第 24 章「選択したプロトコルの実行環境の設定」を参照してください。

6 相関を実行します。

スクリプトを検査して，相関の必要な動的な値があるかどうか確認します。ワイヤレス・プロトコルでは，**web_reg_save_param** 関数を追加して手作業による相関を実行します。

詳細については，663 ページ「手作業による相関」を参照してください。

7 VuGen で Vuser スクリプトを保存して実行します。

VuGen で生成した Vuser スクリプトを保存して実行し，正しく動作することを確認します。記録中，VuGen によって一連の設定ファイル，データ・ファイル，ソース・コード・ファイルが生成されます。これらのファイルには，Vuser の実行時の情報および設定情報が含まれます。VuGen は，これらのファイルをスクリプトと一緒に保存します。

Vuser スクリプトをスタンドアロン・テストとして実行する方法の詳細については，『**第 1 巻 － VuGen の使用**』の「スタンドアロン・モードでの Vuser スクリプトの実行」を参照してください。

スクリプトを作成した後，スクリプトを自分の環境（LoadRunner シナリオ，Performance Center 負荷テスト，または Business Process Monitor プロファイル）に組み込みます。詳細については，**HP LoadRunner Controller**，**HP Performance Center**，または **HP Business Availibility Center** のドキュメントを参照してください。

ワイヤレス Vuser 関数の使用法

ワイヤレス・デバイスと Web サーバ（または WAP 用ゲートウェイ）の間の通信をエミュレートするために開発された関数を Vuser 関数といいます。関数の中には，スクリプトの記録時に生成されるものと，手作業でスクリプトに挿入しなければならないものがあります。また記録の後で，Vuser メッセージ関数とユーザ定義の C 関数を，Vuser スクリプトに追加することもできます。

一般的な Vuser 関数の名前には，**lr** というプレフィックスが付いています。標準的な HTTP のアクションを表す関数の名前には，**web** というプレフィックスが付いています。

WAP 固有の関数の名前には，**wap** というプレフィックスが付いています。たとえば，**wap_connect** は WAP ゲートウェイへの接続を行います。WAP 関数は，ベアラ，接続，書式設定，ゲートウェイ，プッシュ，および Radius 関数というカテゴリに分類されます。

WAP セッション中に RAS または NAS サーバをエミュレートする関数の名前には，**radius** というプレフィックスが付いています。たとえば，**radius_account** は RADIUS サーバに対してユーザを認証します。

すべての VuGen 関数の完全なリストについては，『**Online Function Reference**』（英語版）（［**ヘルプ**］＞［**関数リファレンス**］）を参照してください。

WSP（Wireless Session Protocol）モードでスクリプトを実行する WAP Vuser では，次の関数がサポートされています。

アクション関数	**web_custom_request**，**web_submit_data**，および **web_url**
認証関数	すべて **--web_set_user**，**web_set_certificate**[_ex]
クッキー関数	すべて **--web_add_cookie**，**web_cleanup_cookie**，**web_remove_cookie**

ヘッダ関数	すべて **--web_add_auto_header**，**web_add_header**，**web_cleanup_auto_headers**，**web_save_header**
相関関数	すべて **--web_create_html_param**[_ex]，**web_reg_save_param**，**web_set_max_html_param_len**

プッシュのサポート

通常のクライアント / サーバ・モデルでは，クライアントはサーバに情報またはサービスを要求します。サーバが応答して，クライアントに情報を送信したりサービスを提供したりします。これを**プル**技術といい，クライアントがサーバから情報を取得します。

これに対するものとして，「**プッシュ**」技術があります。WAP のプッシュ・フレームワークは，ユーザによるアクションがなくても，情報をデバイスに送信します。この技術もクライアント / サーバ・モデルに基づいていますが，サーバがコンテンツを送る前にクライアントからの明示的な要求はありません。

WAP でプッシュ操作を実行するときには，「**プッシュ・イニシエータ**（PI）」がクライアントにコンテンツを送信します。ただし，プッシュ・イニシエータ・プロトコルは WAP クライアントと完全互換ではありません。プッシュ・イニシエータはインターネット上にあり，WAP クライアントは WAP ドメインにあるためです。したがって，プッシュ・イニシエータと WAP クライアントの間に，仲介機能を果たす変換ゲートウェイを挿入する必要があります。変換ゲートウェイは，「**プッシュ・プロキシ・ゲートウェイ**（PPG）」といいます。

インターネット側のアクセス・プロトコルは，「**プッシュ・アクセス・プロトコル**（PAP）」といいます。

WAP 側のプロトコルは，「**プッシュ OTA**（Over-The-Air）」プロトコルといいます。

プッシュ・イニシエータは，インターネット上で PAP インターネット・プロトコルを使用して，プッシュ・プロキシ・ゲートウェイ（PPG）にアクセスします。PAP は，HTTP などの一般的なインターネット・プロトコルに埋め込める XML メッセージを使用します。PPG はプッシュされたコンテンツを WAP ドメインに転送します。コンテンツはその後，OTA プロトコルを使用し，モバイル・ネットワークを経由して，目的のクライアントまで送信されます。OTA プロトコルは，WSP サービスに基づいています。

PPG は基本的なプロキシ・ゲートウェイ・サービスを提供するほか，プッシュ・イニシエータにプッシュ操作の最終ステータスを通知することができます。また，双方向のモバイル・ネットワークにおいては，クライアントがコンテンツを受け入れるか拒否するまで待機することができます。

プッシュ・サービスのタイプ

プッシュ・サービスのタイプには，SL と SI があります。

- **SL** ― サービス・ロード（SL）コンテント・タイプでは，モバイル・クライアント上のユーザ・エージェントがサービスをロードして実行できます。たとえば，WML デッキなどを実行できます。SL には，ユーザの介入なしに適宜ユーザ・エージェントによってロードされるサービスを示す URI が含まれています。
- **SI** ― サービス通知（SI）。このコンテント・タイプでは，エンド・ユーザに非同期で通知を送信できます。たとえば，新規メールの到着，株価の変動，ニュースのヘッドライン，広告などの通知が考えられます。

  最も基本的な形式の SI には，ショート・メッセージとサービスを示す URI が含まれています。メッセージは，エンド・ユーザの受信時に提示され，ユーザは URI が示すサービスをすぐに開始するか，後で処理するために SI を延期するかを選択できます。SI を延期すると，クライアントによってサービスが保存され，エンド・ユーザは後でそのサービスを開始できます。

VuGen でのプッシュのサポート

VuGen でのプッシュのサポートは，次の 3 つに分類できます。

- クライアント側でのプッシュのサポート ― プッシュ型メッセージを受信する機能です。
- WAP HTTP Vuser に対するプッシュのサポート ― プッシュ・イニシエータをエミュレートします。
- プッシュ型メッセージ（SI および SL）フォーマット・サービス ― プッシュ型メッセージをフォーマットします。

クライアント側におけるプッシュのサポート

VuGen は，クライアント側では，すべての再生モード（CO および CL）について，SL および SI の両方のプッシュ・サービスをサポートしています。**wap_wait_for_push** 関数は，Vuser にプッシュ・メッセージの到着まで待機するように指示します。この関数のタイムアウトは，実行環境の設定で指定します。

プッシュ・メッセージが到着すると，Vuser によってメッセージが解析され，タイプの識別と属性の取得が行われます。解析が正常に行われると，プル・トランザクションが生成された後に実行され，該当データが取得されます。［実行環境設定］でプル・イベントを無効にすると，Vuser はメッセージを取得しません。詳細については，『**第 1 巻 － VuGen の使用**』の第 24 章「選択したプロトコルの実行環境の設定」を参照してください。

プッシュ・イニシエータのエミュレート

WAP HTTP Vuser のプッシュ機能のサポートにより，PPG の負荷テストが可能です。プッシュのサポートにより，Vuser は，**Push Access Protocol**（PAP）をサポートするプッシュ・イニシエータとして機能できます。PAP では，次の PI と PPG の間の一連の操作が定義されています。

- プッシュ要求を送信する
- プッシュ要求を取り消す
- プッシュ要求のステータスを調べるクエリを送信する
- ワイヤレス・デバイスの機能のステータスを調べるクエリを送信する
- PPG から PI へ結果通知メッセージを発行する

前述の操作はすべて，要求と応答から成ります。つまり，発行されたすべてのメッセージに対して，応答が PI に返されます。PI の操作は，VuGen でサポートされている通常の HTTP POST メソッドに基づいています。現バージョンでは，最初の 2 つの操作だけが **wap_push_submit** および **wap_push_cancel 関数によってサポートされています。**

web_submit_data 関数を使用して，Web サーバにデータを送信できます。ただし，この関数では長く複雑な構造のデータを送信することは困難です。この種のデータの送信を可能にするため，またより直観的に理解できる API 関数を提供するために，XML メッセージ・データを適切にフォーマットする新しい API 関数 **wap_format_si_msg** と **wap_format_sl_msg** が追加されました。これらの関数の詳細については，『**Online Function Reference**』（英語版）を参照してください。

MMS（マルチメディア・メッセージング・サービス）Vuser スクリプト

MMS（マルチメディア・メッセージング・サービス）は，SMS プロトコルの拡張機能です。SMS メッセージにはテキストのみ含められるのに対し，MMS では，MMS 対応の送受話器との間でさまざまなコンテンツを持つメッセージを送受信できます。テキスト，音声，電子メール・メッセージ，画像，ビデオ・クリップ，そしてストリーミング・データの形式のコンテンツが使用できます。また，携帯電話から電子メール・アドレスへマルチメディア・メッセージを送信することもできます。

一般に，MMS メッセージには添付ファイルの集合が含まれます。SMS メッセージ・サイズは 160 バイトに制限されていますが，MMS メッセージ・サイズは数 MB でも可能です。このような大きいサイズのメッセージが送信できるように，MMS では通常，第 3 世代（3G）通信網が必要です。

MMS メッセージを受信するために，携帯電話は SMS を介して MMS 通知を受信します。SMS メッセージは，SMPP，UCP，CIMD2 などのさまざまな SMS プロトコルを介して受信できます。SMS メッセージには，MMSC サーバのデータベースに格納されている MMS メッセージへの一意のパスが含まれます。携帯電話は，このパスを使用して，SMSC からメッセージをダウンロードします。VuGen の現在のバージョンでは，SMPP インタフェースを介した MMS 通知の受信をサポートしています。

マルチメディア・メッセージング・サービス Vuser スクリプトは，OMA（Open Mobile Alliance）で定義されている MMS プロトコルのバージョン 1.0 および 1.1 をサポートしています。MMS Vuser を使用することで，HTTP プロトコルを利用して，または WAP プロトコルを利用して WAP ゲートウェイを経由で，直接 MMSC サーバに MMS メッセージを送信できます。

マルチメディア・メッセージング・サービス関数は，MMS メッセージの送信と受信をエミュレートします。各関数の名前には，**mm** というプレフィックスが付いています。これらの関数の構文情報の詳細については，『**Online Function Reference**』（英語版）（［**ヘルプ**］＞［**関数リファレンス**］）を参照してください。

Controller での MMS シナリオの実行

MMS（マルチメディア・メッセージング・サービス）シナリオには，コマンド・ライン設定が必要です。

MMS コマンド・ラインの設定を行うには，次の手順を実行します。

1 ［シナリオのスケジュール］画面で［**詳細**］をクリックします。［Group Information］ダイアログ・ボックスが表示されます。

2 ［コマンド ライン］ボックスが表示されていない場合は，［**詳細表示**］ボタンをクリックします。

3 ［コマンド ライン］のテキストの最後に -usingwininet yes を追加します。

4 ［**OK**］をクリックして，コマンド・ライン・スイッチを適用します。

索引