HP LoadRunner

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 : 11.00 パッチ 02

Virtual User Generator ユーザーズ・ガイド

ドキュメント・リリース日 : 2011 年 2 月（英語版）
ドキュメント・リリース日 : 2011 年 2 月（英語版）

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。本文書の内容は，追加の保証を構成するものではありません。HP は，本文書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。

本文書に含まれる情報は，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。

著作権

商標

Java は，Oracle および / またはその系列会社の登録商標です。

Microsoft® および Windows® 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。

文書の更新

本書の表紙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。
- 文書が更新されるごとに変更されるドキュメント・リリース日。
- 当該ソフトウェア・バージョンのリリース日を示す，ソフトウェア・リリース日。

最新の更新を確認する，あるいは使用している文書が最新版であるかどうかを確認するには，次の URL を参照してください。

http://h20230.www2.hp.com/selfsolve/manuals

このサイトを使用するには HP Passport に登録してサインインする必要があります。HP Passport ID を登録するには，次の URL を参照してください。

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

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

適切な製品のサポート・サービスに登録すれば，最新版または新版を入手できます。詳細については，HP の担当窓口にお問い合わせください。

サポート

HP Software のサポート Web サイトは次のとおりです。

http://www.hp.com/go/hpsoftwaresupport

この Web サイトでは，連絡先情報や，HP Software が提供する製品，サービスおよびサポートの詳細を提供しています。

HP Software のオンライン・サポートでは，セルフソルブ機能を提供しています。ビジネス管理に必要な対話型技術サポート・ツールにアクセスするための迅速かつ効率的な方法を提供します。弊社サポートの大切なお客様として，サポート Web サイトを使用して次のことが行えます。

- 興味のあるナレッジ文書の検索
- サポート事例と向上のためのリクエストの送信および追跡
- 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 サポート契約の検索
- 利用可能なサービスに関する情報の確認
- ソフトウェアを利用しているほかのお客様との討論への参加
- ソフトウェア・トレーニングの検索と登録

サポート領域のほとんどでは HP Passport ユーザとして登録しサインインする必要があります。また多くでサポート契約も必要です。HP Passport ID を登録するには，次の URL を参照してください。

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

アクセス・レベルの詳細については，次の URL を参照してください。

http://h20230.www2.hp.com/new_access_levels.jsp

目次

目次

LoadRunner VuGen へようこそ

HP Virtual User Generator（*VuGen*）へようこそ。VuGen は，HP の仮想ユーザ・スクリプト作成ツールです。VuGen を使用して，一般的なビジネス・プロセスを実行するユーザを記録することで，仮想ユーザ・スクリプトを作成します。このスクリプトを使用して，実際の状況をエミュレートできます。

VuGen で作成したスクリプトは，HP LoadRunner，HP Performance Center，および HP Business Availability Center など，ほかの製品と組み合わせて使用します。

HP LoadRunner は，パフォーマンス・テストのためのツールです。このツールを使用して，アプリケーション全体に負荷をかけ，クライアント，ネットワークおよびサーバの潜在的なボトルネックの切り分けと特定を行います。

HP Performance Center では，LoadRunner の機能が企業レベルで実装されます。

HP Business Availability Center を使用すると，稼動中のビジネス・アプリケーションとシステムの管理と可用性を容易に最適化できます。

本書の構成

本書は，次の部で構成されています。

第 I 部　VuGen を使った作業

Virtual User Generator のインタフェース，およびスクリプトの記録と再生について説明します。また，標準の実行環境の設定，データ・パラメータの使用方法，スクリプトのカスタマイズ方法について説明します。

第 II 部　プロトコル

個々のプロトコルおよびプロトコル・グループに関連する情報を提供します。

第 III 部　上級ユーザのために

一般的なデバッグのヒント，VuGen によって生成されるファイル，Visual C および Visual Basic でスクリプトをプログラミングする方法など，上級ユーザ向けの情報を提供します。

対象読者

このガイドは次のユーザを対象とします。

- スクリプト開発者
- 機能テスト担当者
- 読み込みテスト担当者

このドキュメントは，読者に，エンタープライズ・アプリケーションについての適切な知識があることを前提とします。

文書ライブラリ・ガイド

文書ライブラリは，次のガイドと参照先で構成されます。これらは，オンライン，PDF 形式，またはその両方で入手できます。PDF は Adobe Reader を使用して参照や印刷を行うことができます。Adobe Reader は，Adobe の Web サイト（http://www.adobe.com/jp）からダウンロードできます。

「**この文書ライブラリの使用**」では，文書ライブラリの使用方法と文書ライブラリがどのように編成されているかについて説明します。

ドキュメントへのアクセス方法

このドキュメントには次のようにしてアクセスできます。

- ［**スタート**］メニューから［**スタート**］>［**LoadRunner**］>［**Documentation**］をクリックし，ドキュメントを選択します。
- ［**ヘルプ**］メニューから，［**文書ライブラリ**］をクリックして 1 つに統合されたヘルプを開きます。

最初にお読みいただくドキュメント

- 『**最初にお読みください**』: LoadRunner に関する最新のお知らせと情報を提供します。『最初にお読みください』には，［**スタート**］メニューからアクセスします。
- 『**HP LoadRunner クイック・スタート**』: LoadRunner の使用に関する簡潔かつ順を追った概要を提供します。［スタート］メニューからクイック・スタートにアクセスするには，［**スタート**］>［**LoadRunner**］>［**Quick Start**］をクリックします。
- 『**HP LoadRunner チュートリアル**』: 自分のペースで進められる印刷可能なガイドです。負荷テストのプロセスを示し，LoadRunner のテスト環境に慣れていただくことを目的としています。［スタート］メニューからチュートリアルにアクセスするには，［**スタート**］>［**LoadRunner**］>［**Tutorial**］をクリックします。

LoadRunner ガイド

- **『HP Virtual User Generator ユーザーズ・ガイド』**: VuGen を使用してスクリプトを作成する方法について説明します。印刷版は，第 1 巻「*VuGen の使用*」と第 2 巻「*プロトコル*」の 2 巻に分冊されていますが，オンライン版は 1 つにまとめられています。このユーザーズ・ガイドは，必要に応じてオンラインの『HP LoadRunner オンライン関数リファレンス』と合わせて参照してください。
- **『HP LoadRunner Controller ユーザーズ・ガイド』**: Windows 環境で LoadRunner Controller を使用して LoadRunner シナリオを作成し実行する方法について説明します。また，シナリオで生成されたデータを監視するために，サーバ・モニタ環境をセットアップし LoadRunner モニタを設定する方法について説明します。
- **『HP LoadRunner Analysis ユーザーズ・ガイド』**: LoadRunner Analysis グラフとレポートを使用してシナリオの実行後にシステム・パフォーマンスを分析する方法について説明します。
- **『HP LoadRunner インストール・ガイド』**: LoadRunner および LoadRunner の追加コンポーネント（LoadRunner サンプルなど）のインストール方法について説明します。

LoadRunner 参照先

- **『LoadRunner Function Reference』（英語版）**: 仮想ユーザ・スクリプトの作成時に使用できる LoadRunner のすべての関数を，その使用例とともにオンラインで参照できます。
- **『Analysis API Reference』（英語版）**: この Analysis API セットは，Analysis セッションの無人作成や，Controller で実行されたテストの結果からのユーザ定義によるデータ抽出に使用できます。この参照先へは，Analysis の［ヘルプ］メニューからアクセスできます。
- **『トラブルシューティング』**:［出力メッセージ］ダイアログ・ボックス（［**Controller**］>［**表示**］>［**出力メッセージを表示**］）に状況に応じたエラーの詳細が表示されます。［ヘルプ］カラムのアイコンをクリックしてトラブルシューティング・ガイドを開きます。このガイドには，Controller 接続および Web プロトコル・エラーのわかりやすい説明とトラブルシューティングのヒントが含まれます。また，Winsock，SAPGUI，Citrix プロトコルに関する一般的なトラブルシューティングのヒントが含まれます。

文書ライブラリの検索とナビゲーション

文書ライブラリ では次の機能を使用できます。

オプション	説明
	**検索とナビゲーション。**ナビゲーション表示枠が表示されます。このボタンはナビゲーション表示枠が閉じている場合にのみ表示されます。 ナビゲーション表示枠には次のタブが含まれます。 ➤ **［目次］タブ：**トピックが階層ツリー形式で編成されており，特定のガイドまたはトピックに直接移動できます。 ➤ **［索引］タブ：**トピックの詳細なリストとともに，各トピックが説明されているページ番号が表示されます。索引エントリをダブルクリックすると，対応するトピックが表示されます。選択した索引エントリが複数の文書にある場合，右の表示枠に選択可能な場所のリストが表示されます。これにより，コンテキストを選択できます。 ➤ **［検索］タブ：**特定のトピックまたはキーワードを検索できます。結果は，ランク順に返されます。範囲ドロップダウン・リストから値を選択することで，検索を特定のガイドに限定できます。 **注：**検索では，引用符（"）が使用されているかどうかに関係なく，語句全体ではなく語句内の個々の単語が調べられます。 ➤ **［お気に入り］タブ：**特定のトピックをすばやく参照できるようにブックマークできます。 ［**お気に入り**］タブは，ヘルプの Java 実装を使用する場合にのみ使用できます。ブラウザで Java がサポートされない場合，JavaScript 実装が自動的に使用され，［お気に入り］タブは表示されません。
	**［内容の表示］：**ナビゲーション表示枠に［**目次**］タブが表示され，現在表示されているページに対応するエントリが強調表示されます。 このボタンは，ナビゲーション表示枠が開いている場合にのみ表示されます。
	**［前］と［次］：**現在表示されているガイドの前のページまたは次のページに移動します。

オプション	説明
	**文書のフィードバックを HP に送信する。**HP はフィードバックを歓迎します。任意のトピックでこのボタンを使用すると，参照ページが含まれた当社宛の電子メールが開きます。ご意見，ご要望，および発見したエラーに関する情報をお送りください。
	**[印刷]**：現在表示されているページを印刷します。ガイド全体を印刷するには，文書ライブラリのホーム・ページから印刷用のリンクにアクセスしてください。
戻る	前に表示していたページに戻るには，使用しているブラウザの**戻る**機能を使用できます。ほとんどのブラウザで，右クリックして表示されるショートカット・メニューから**戻る**機能を選択できます。
この文書ライブラリの使用	各コンテンツ・ページの左下にあります。 このセクションを開きます。
用語集	各コンテンツ・ページの左下にあります。 用語の定義と頭字語を含む用語集が開きます。

トピックの種類

注：このセクションが適用されるのは，LoadRunner Controller，VuGen，Analysis の各ユーザーズ・ガイドのみです。

上記の LoadRunner ガイドの目次はトピック別に整理されています。主なトピックの種類として，**概念**，**タスク**，および**参照**の 3 つが使用されます。トピックの種類はアイコンで視覚的に区別されます。

トピックの種類	説明	使用方法
概念 	背景，説明的な情報，または概念的な情報。	機能についての一般的な情報を学習します。
タスク 	**手順に従ったタスク。**アプリケーションを使った作業を行って目標を達成するためのステップ・バイ・ステップのガイダンス。 タスクの手順は，番号付けされている場合と番号付けされていない場合があります。 ➤ **番号付けされた手順。**順序どおりに各手順に従うことで実行するタスク。 ➤ **番号付けされていない手順。**任意の順序で実行できる自己完結型の一連の操作。	➤ タスクのワークフロー全体について学習します。 ➤ 番号付けされたタスク内の各手順に従って，タスクを完了します。 ➤ 番号付けされていないタスク内の各手順を完了して，独立した操作を実行します。
	事例シナリオ・タスク。特定の状況におけるタスクの実行方法を示す例。	現実に近いシナリオでタスクをどのように実行できるかを学習します。

トピックの種類	説明	使用方法
参照先 	**一般参照。**参照指向の資料の詳細なリストと説明。	特定のコンテキストに関連する，参照情報の特定の部分を調べます。
	ユーザ・インタフェースの参照。特定のユーザ・インタフェースを詳細に説明する専門的な参照トピック。通常，製品の［ヘルプ］メニューから［**Help on this page**］を選択すると，ユーザ・インタフェースのトピックが開きます。	ウィンドウ，ダイアログ・ボックス，ウィザードなど，1 つ以上の特定のユーザ・インタフェース要素の入力項目や使用方法に関する特定の情報を調べます。
トラブルシューティングと制限事項 	**トラブルシューティングと制限事項。**一般的に発生する問題とその解決策の説明，および機能または製品領域の制限事項の一覧が含まれる専門的な参照トピック。	機能を使って作業を行う前に，またはソフトウェアで有用性の問題に直面した場合に，重要な問題に関する認識を高めます。

その他のオンライン・リソース

［**トラブルシューティングとナレッジ ベース**］：セルフソルブ技術情報を検索できる HP Software のサポート Web サイトのトラブルシューティング・ページにアクセスできます。［**ヘルプ**］>［**トラブルシューティングとナレッジ ベース**］を選択します。この Web サイトの URL は http://h20230.www2.hp.com/troubleshooting.jsp です。

［**HP Software のサポート**］：HP Software のサポート Web サイトにアクセスできます。このサイトでは，セルフ・ソルブ技術情報を参照できます。また，ユーザ・ディスカッション・フォーラムにおける新情報送信，既存情報の検索，サポート・リクエストの送信，パッチや最新版ドキュメントのダウンロードなど，さまざまなサービスをご利用いただけます。［**ヘルプ**］>［**HP Software のサポート**］を選択します。この Web サイトの URL は www.hp.com/go/hpsoftwaresupport です。

サポート領域のほとんどでは HP Passport ユーザとして登録しサインインする必要があります。また多くでサポート契約も必要です。

アクセス・レベルの詳細については，次の URL を参照してください。

http://h20230.www2.hp.com/new_access_levels.jsp

HP Passport ユーザ ID を登録するには，次の URL を参照してください。

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

［**HP Software の Web サイト**］からは HP Software の Web サイトにアクセスできます。このサイトでは，HP Software 製品に関する最新情報をご覧になれます。たとえば，新しいソフトウェアのリリース，セミナー，展示会，カスタマー・サポートなどの情報が含まれます。［**ヘルプ**］>［**HP Software の Web サイト**］を選択します。この Web サイトの URL は www.hp.com/go/software です。

HP Software は，製品ドキュメントを新しい情報で継続的に更新しています。

最新の更新を確認する，あるいは使用している文書が最新版であるかどうかを確認するには，HP Software 製品マニュアル Web サイト（http://h20230.www2.hp.com/selfsolve/manuals）を参照してください。

第 I 部

VuGen を使った作業

1

概要

本章の内容

概念

VuGen の概要

環境のテストや監視を実行する際は，システムにおけるユーザの振る舞いを正確にエミュレートする必要があります。HP のテスト・ツールは，複数のユーザが同時に作業している環境や，複数のユーザがシステムに同時にアクセスしている環境をエミュレートします。

そのような環境をエミュレートするために，実際のユーザの代わりとして*仮想ユーザ*を使用します。仮想ユーザが実行するアクションは*仮想ユーザ・スクリプト*によって表現されます。仮想ユーザ・スクリプトを作成するための中心的なツールとなるのが Virtual User Generator，*VuGen* です。

VuGen では，仮想ユーザ・スクリプトの記録だけでなく実行も可能です。VuGen でのスクリプト実行はデバッグに役立ちます。スクリプトを実行することによって，仮想ユーザ・スクリプトが大規模なテストの一部としてどのように動作するかを確認できます。

仮想ユーザ・スクリプトの記録時に，記録セッション中に実行されたアクションを定義するさまざまな関数が VuGen によって生成されます。VuGen では，これらの関数が VuGen エディタに挿入されることによって，基本となる仮想ユーザ・スクリプトが作成されます。

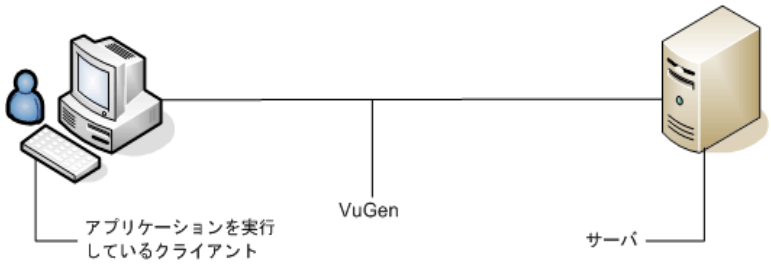
VuGen で仮想ユーザ・スクリプトを記録できるのは Windows プラットフォームのみです。しかし，記録した仮想ユーザ・スクリプトは，Windows および UNIX の両プラットフォームで実行できます。

仮想ユーザの概要

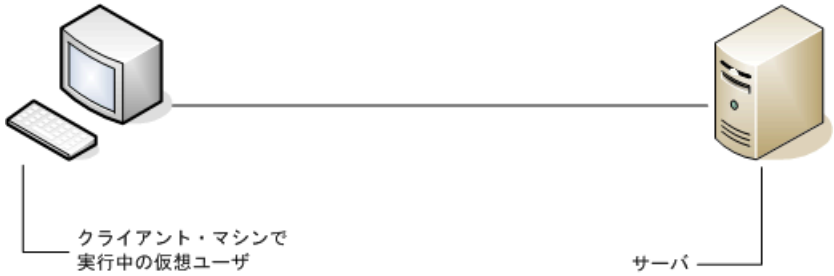
仮想ユーザは実際のユーザのアクションを，一般的なアプリケーションの標準的なビジネス・プロセスを実行することでエミュレートします。仮想ユーザが記録セッション中に実行するアクションは*仮想ユーザ・スクリプト*によって表現されます。

HP のツールの中で，仮想ユーザ・スクリプトを作成するための中心的なツールとなるのが Virtual User Generator，*VuGen* です。VuGen では，クライアント・アプリケーションで典型的なビジネス・プロセスを実行しているユーザの操作を記録することによって，仮想ユーザ・スクリプトを作成します。VuGen は記録セッション中に実行されたアクションを記録しますが，記録されるのはクライアントとサーバの間のやり取りだけです。アプリケーションの API 関数からサーバへの呼び出しを手動でプログラミングする代わりに，VuGen は，実際に起こった状況を正確にモデル化しエミュレートする関数を，自動的に生成します。

記録時には，VuGen がデータベースのクライアント側を監視し，ユーザと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。



再生時には，仮想ユーザ・スクリプトがサーバ API への呼び出しを実行して，サーバと直接やり取りします。仮想ユーザからサーバと直接通信するときは，システム・リソースがクライアント・インタフェースで必要になりません。このため，1 台のワークステーションで多数の仮想ユーザを同時に実行でき，数台のテスト用マシンだけで大きなサーバ負荷をエミュレートすることが可能になります。



また，仮想ユーザ・スクリプトはクライアント・ソフトウェアに依存しないため，クライアント・ソフトウェアのユーザ・インタフェースが完成していなくても，仮想ユーザを使ってサーバのパフォーマンスを検査できます。

VuGen を使うと，スクリプトをスタンドアロン・テストとして実行できます。スクリプトを VuGen から実行することで，デバッグの際に仮想ユーザの振る舞いを調べ，どのような機能拡張が必要なのかを判断できるので便利です。

VuGen ではさまざまなタイプの仮想ユーザを記録でき，それぞれのタイプが特定の負荷テスト環境またはトポロジに対応しています。その結果，特定のタイプの仮想ユーザ・スクリプトが得られます。たとえば，Web 仮想ユーザ・スクリプトは，Web ブラウザを操作しているユーザをエミュレートする場合に使用します。FTP 仮想ユーザは，FTP セッションのエミュレートに使用します。さまざまな仮想ユーザ・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，効果的な負荷テスト，または HP Business Process Monitor 設定を作成できます。

仮想ユーザの実行中は，システムの応答に関する情報を収集します。その後で，それらの情報を Analysis ツールを使って表示できます。たとえば，銀行の ATM から現金を引き出す 100 の仮想ユーザを同時に実行して，そのときのサーバの動作を観察することができます。

VuGen で仮想ユーザ・スクリプトを記録できるのは Windows プラットフォームのみです。しかし，記録した仮想ユーザ・スクリプトは，Windows および UNIX の両プラットフォームで実行できます。

［タスク］表示枠 の概要

［タスク］表示枠には機能スクリプトの作成に必要なタスクの一覧が表示されます。一覧でタスクをクリックすると，ウィザードでそのステップが開きます。現在のタスクは矢印で示されます。

タスクのリストは，［**記録**］（C 仮想ユーザおよび Web サービス仮想ユーザでのスクリプト作成），［**再生**］，［**拡張**］，［**負荷の準備**］，［**完了**］の 5 つのパートに分かれています。

Web，Web Services，記録不可能なプロトコル（ユーザ定義 C 仮想ユーザ）では，最初のタスクが若干異なります。

スクリプトのセクション

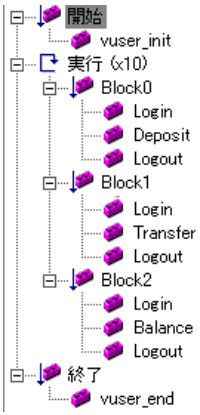
どの仮想ユーザ・スクリプトにも，*vuser_init*，*実行（Action）*，および *vuser_end* の 3 つのセクションがあります。仮想ユーザがスクリプトの実行時に「*実行*」セクションを繰り返し実行するように指定できます。この繰り返しを「*反復*」といいます。

反復を複数回実行する場合，仮想ユーザ・スクリプトの *vuser_init* セクションと *vuser_end* セクションは繰り返されません。

複数のアクションが含まれるスクリプトを実行するときに，アクションの実行方法を指定し，仮想ユーザがどのようにアクションを実行するかを設定できます。

- **[アクション ブロック]**：アクション・ブロックは，スクリプト内のアクションのグループです。アクションをグループ化して個別のアクション・ブロックを作成し，同じアクションを複数のブロックに追加できます。アクション・ブロックまたは各アクションを順番（Sequential）に実行するか，ランダム（Random）に実行するかを設定できます。標準設定の Sequential モードでは，仮想ユーザは，反復のツリー・ビューに表示されている順番でブロックまたはアクションが実行されます。

  次の例では，*Block0* は預け入れ，*Block1* は振り替えを実行し，*Block2* は残高要求を送信します。*Login* と *Logout* のアクションは，3 つのブロックに共通です。



- **[順番]**：スクリプト内のアクションの順序を設定できます。アクションを順番に実行するか，ランダムに実行するかを指定することもできます。
- **[反復]**：「*実行*」セクション全体の反復回数を設定する以外に，各アクションまたはアクション・ブロックの反復回数を設定することもできます。これはたとえば，製品を探すのに多数のクエリを実行するけれども，購入する場合は 1 回だけというような商用サイトでの操作のエミュレーションに役立ちます。
- **[重み付け]**：アクションをランダムに実行するアクション・ブロックには，*重み付け*（ブロック内の各アクションの割合）を設定できます。

  ユーザ・インタフェースの詳細については，452 ページの「[一般] > [実行論理] ノード」を参照してください。

HP LoadRunner ライセンス

LoadRunner の購入時に，設定とニーズに合うようにカスタム・ライセンスを受け取ります。この情報には，LoadRunner Launcher を使っていつでもアクセスできます。ライセンス・キー情報を確認するには，［**スタート**］>［**すべてのプログラム**］>［**HP LoadRunner**］を選択し，LoadRunner Launcher を起動します。Launcher の［**設定**］メニューから［**LoadRunner ライセンス**］を選択します。

Service Test のライセンスを購入した場合は，ライセンス・マネージャからインストールします（［**スタート**］>［**すべてのプログラム**］>［**HP LoadRunner**］>［**Service Test**］>［**Service Test License Manager**］）。ライセンスには，シート・ライセンス（ホスト・ロック）とコンカレント・ライセンス（サイト・ライセンス）があります。詳細については，*HP LoadRunner インストール・ガイド*を参照してください。

HP Service Test は，スタンドアロン製品としても，HP LoadRunner の Virtual User Generator，VuGen の拡張としても利用できます。Service Test には，VuGen のすべての機能に加えて，SOA および Web サービスのテストの領域の機能が追加されています。

HP Service Test は機能テスト・ツールであるため，VuGen より機能的なテスト機能が提供されます。これに加えて，Service Test には，WSDL，XML，および SOAP を使って作業できる複数のユーティリティが含まれています。

HP Service Test で作成したスクリプトはすべて，HP LoadRunner および HP Performance Center で実行できます。この機能は，Service Test と LoadRunner のバージョンが同じである場合のみ利用できます。

VuGen コードの概要

仮想ユーザ・スクリプトを記録すると，VuGen は仮想ユーザ関数を生成してスクリプトに挿入します。仮想ユーザ関数には，次の 3 つのタイプがあります。

- 一般仮想ユーザ関数
- プロトコル固有の仮想ユーザ関数
- 標準 ANSI C 関数

*一般*仮想ユーザ関数と*プロトコル固有の*関数によって LoadRunner API が構成されます。仮想ユーザは，この API によってサーバと直接通信できるようになります。VuGen では新しいスクリプトを作成するときに，サポートしている全プロトコルのリストが表示されます。全仮想ユーザ関数の構文については，*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。

一般仮想ユーザ関数

一般仮想ユーザ関数は，関数名に **lr** というプレフィックスが付いているので LR 関数とも呼ばれます。LR 関数はどのタイプの仮想ユーザ・スクリプトでも使えます。LR 関数を使って次のようなことができます。

- 仮想ユーザ，仮想ユーザ・グループ，およびホストに関する実行情報の取得 ¥¥gollum¥cd¥00966¥FM¥。
- 仮想ユーザ・スクリプトへのトランザクションと同期ポイントの追加。たとえば，**lr_start_transaction**（Java では **lr.start_transaction**）関数はトランザクションの開始を，**lr_end_transaction**（Java では **lr.end_transaction**）関数はトランザクションの終了を示します。詳細については，149 ページの「負荷テスト用スクリプトの準備」を参照してください。
- エラーや警告を示すメッセージの出力への送信。

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。

プロトコル固有の仮想ユーザ関数

仮想ユーザ関数に加えて，VuGen は記録中にプロトコル固有の関数を生成して，仮想ユーザ・スクリプトに挿入します。

プロトコル固有の関数は，記録対象となっている仮想ユーザのタイプに固有のものです。VuGen は，たとえばデータベース・スクリプトには LRD 関数を，Tuxedo スクリプトには LRT 関数を，Windows Sockets スクリプトには LRS 関数を挿入します。

標準設定では，VuGen の自動スクリプト・ジェネレータは，ほとんどのプロトコルで C 言語の仮想ユーザ・スクリプトを生成し，Java タイプ・プロトコルには Java の仮想ユーザ・スクリプトを生成します。VuGen に，Visual Basic や Javascript でコードを生成させることができます。詳細については，358 ページの「[一般］の［スクリプト］ノード」を参照してください。

フロー制御や構文も含め，言語のすべての標準規則がスクリプトに追加できます。ほかのプログラミング言語と同様，スクリプトにコメントや条件ステートメントを追加することもできます。

次の Web 仮想ユーザ・スクリプト（抜粋）は，VuGen によって記録および生成され，スクリプトに挿入される関数の例です。

```
#include "as_web.h"


Action1()
{

    web_add_cookie("nav=140; DOMAIN=dogbert");

    web_url("dogbert",
        "URL=http://dogbert/",
        "RecContentType=text/html",
        LAST);

    web_image("Library",
        "Alt=Library",
        LAST);

    web_link("1 Book Search:",
        "Text=1 Book Search:",
        LAST);

    lr_start_transaction("Purchase_Order");
…
```

仮想ユーザ・スクリプトでの C 関数の使用の詳細については，*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。Java スクリプトの変更の詳細については，第 22 章，「Java プロトコル - 手作業によるスクリプトのプログラミング」。を参照してください。

注：C 言語で記述された仮想ユーザ・スクリプトの実行に使用される C インタプリタは，ANSI C 言語だけをサポートします。Microsoft 社による ANSI C への拡張はサポートしていません。

標準 ANSI C 関数

標準 ANSI C 関数を追加して仮想ユーザ・スクリプトを拡張できます。ANSI C 関数を使って，仮想ユーザ・スクリプトにコメント文，フロー制御ステートメント，条件文などを追加できます。標準 ANSI C 関数は任意のタイプの仮想ユーザ・スクリプトに追加できます。詳細については，1207 ページの「C 言語の関数の使用についてのガイドライン」を参照してください。

VuGen コードのツール

VuGen の API 関数に関する情報を得るには，次のように複数の方法があります。

- オンライン関数リファレンス
- 単語の補完
- 関数構文の表示
- ヘッダ・ファイル

さらに，標準の検索機能を使用して（[**編集**] > [**検索**]）スクリプト内の関数を見つけたり，またはファイル内の検索機能を使用して，スクリプト内のファイルをすべて検索したりできます。

オンライン関数リファレンス

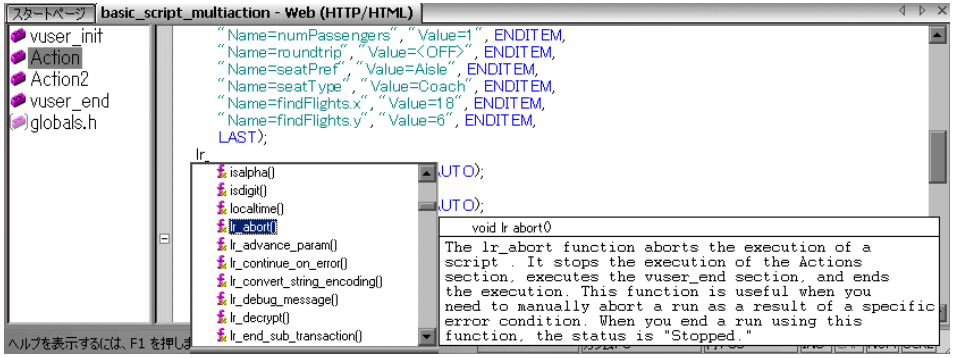
*オンライン関数リファレンス*には，すべての VuGen 関数についての詳しい構文情報が含まれています。関数の使用例も含まれています。関数名の検索が可能なほか，カテゴリまたはアルファベット順のリストから参照できます。

*オンライン関数リファレンス*を開くには，VuGen インタフェースから［**ヘルプ**］>［**関数リファレンス**］を選択します。次に，プロトコルおよび必要なカテゴリを選択します。

スクリプトにすでに含まれている特定の関数についての情報を取得するには，VuGen エディタのカーソルをその関数に移動して，F1 キーを押します。

単語の補完

VuGen エディタには，*ステートメントの補完* 拡張機能の一部として *単語の補完* 機能が組み込まれています。関数のタイプ入力を始めたときに最初のアンダーバーを入力すると，その関数のプレフィックスに一致する使用可能なすべての関数の構文と説明を示すリスト・ボックスが開きます。

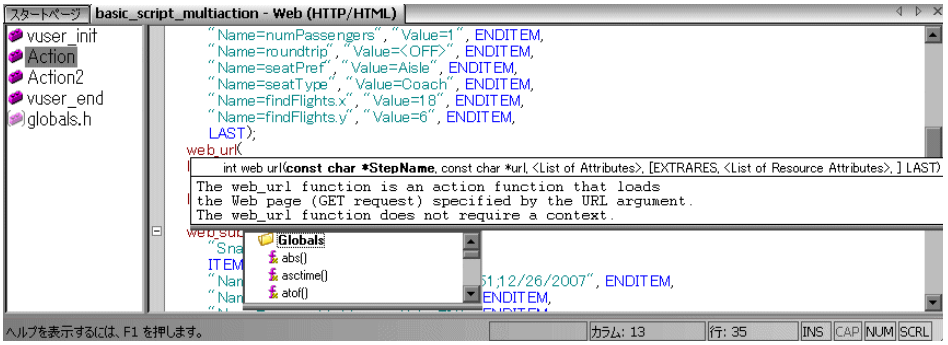


表示された関数を使用するには，その項目を選択するか，目的の項目までスクロールしてから選択します。選択した関数がカーソルの位置に挿入されます。リスト・ボックスを閉じるには，Esc キーを押します。

標準設定では，単語の補完機能は VuGen 全体で有効になっています。単語の補完機能を無効にするには，［**ツール**］>［**一般オプション**］から［**環境**］タブを選択します。［**単語の自動補完**］オプションの横にあるチェック・ボックスをクリアします。単語の補完機能を VuGen 全体で無効にしている場合でも，Ctrl キーを押しながらスペース・キーを同時に押すか，エディタで入力しながら［**編集**］>［**単語入力候補**］を選択すれば，自動補完のリスト・ボックスを表示できます。

関数構文の表示

VuGen のステートメントの自動補完には，このほかにも［**関数の構文を表示**］機能があります。関数の開始括弧を入力すると，関数の構文，引数，プロトタイプ，および簡単な説明が自動的に表示されます。



標準設定では，**関数構文の自動表示**機能が VuGen 全体で有効になっています。この機能を無効にするには，［**ツール**］>［**一般オプション**］から［**環境**］タブを選択します。［**関数構文の自動表示**］チェック・ボックスをクリアします。

［**関数構文の自動表示**］オプションを VuGen 全体で無効にしている場合でも，エディタ画面で開始括弧を入力した後に，Ctrl キーと Shift キーを押しながらスペース・キーを押すか，［**編集**］>［**関数の構文を表示**］を選択すれば，構文を表示させることができます。

ヘッダ・ファイル

Java 以外の関数のプロトタイプはすべてライブラリ・ヘッダ・ファイルに含まれています。ヘッダ・ファイルは，製品のインストール先ディレクトリの下の *include* ディレクトリにあります。ヘッダ・ファイルには，詳しい構文情報と戻り値が含まれています。また，定数の定義，適用範囲，その他の詳細情報が含まれており，関数リファレンスにはない情報もあります。

ほとんどの場合，ヘッダ・ファイル名はプロトコルのプレフィックスに対応しています。たとえば，**lrd** というプレフィックスで始まるデータベース関数のリストは，**lrd.h** ファイルにあります。

次の表に，使用頻度の高いプロトコルと，対応するヘッダ・ファイルを示します。

プロトコル	ファイル
AJAX（Click and Script）	**web_ajax.h**
Citrix	**ctrxfuncs.h**
COM/DCOM	**lrc.h**
データベース	**lrd.h**
FTP	**mic_ftp.h**
一般的な C 関数	**lrun.h**
IMAP	**mic_imap.h**
LDAP	**mic_mldap.h**
MAPI	**mic_mapi.h**
Oracle NCA	**orafuncs.h**
POP3	**mic_pop3.h**
RDP	**lrrdp.h**
SAPGUI	**as_sapgui.h**
SAP（Click and Script）	**sap_api.h**
Siebel	**lrdsiebel.h**
SMTP	**mic_smtp.h**
ターミナル・エミュレータ	**lrrte.h**
WAP	**as_wap.h**
Web（HTML¥HTTP）	**as_web.h**
Web（Click and Script）	**web_api.h**
Web サービス	**wssoap.h**
Windows Sockets	**lrs.h**

HP Service Test 機能

LoadRunner の Service Test の機能をすべて有効にするには，LoadRunner マシンに Service Test ライセンスをインストールする必要があります。LoadRunner マシンに Service Test ライセンスをインストールするには，**<LoadRunner インストール・フォルダ >¥bin¥LicMan.exe** を実行します。

次の機能は，HP Service Test（スタンドアロン）または LoadRunner と Service Test でのみ利用できます。

機能テスト・ユーティリティ

- **XML 妥当性検証**：XML が正しく書式化されていて，スキーマに準拠し，期待値が使用されているかどうかが確認されます。
- **チェックポイント**：応答を期待値と比較できます。
- **ネガティブ・テスト**：SOAP 障害を期待する応答として定義する機能で，エラー・ケースをテストできます。
- **WS-I 妥当性確認**：WSDL および SOAP が WS-I に準拠しているかどうかを確認する機能です。
- **テスト生成ウィザード**：アスペクト・ベースのテストを作成するのに役立つウィザードです。

HP Application Lifecycle Management との統合

- **ALM からのリモート実行**：Service Test スクリプトを，ALM から直接起動できます。パラメータおよび実行環境の設定，テストの実行，および ALM 内からの結果の検査を行えます。

その他のツール

- **サービス・エミュレーション**：クライアントとサーバの両方をシミュレートするエミュレーション・サービスを作成するツールです。
- **コマンド・ライン呼び出し**：コマンド・ラインからスクリプトを呼び出す機能です。

マルチ・プロトコル・スクリプト

シングル・プロトコル・モードで記録する場合，VuGen は指定したプロトコルだけを記録します。マルチ・プロトコル・モードで記録する場合，VuGen はアクションを複数のプロトコルで記録します。マルチ・プロトコル・スクリプトは，COM，FTP，IMAP，Oracle NCA，POP3，RealPlayer，Window Sockets（raw），SMTP，および Web のプロトコルをサポートします。

仮想ユーザ・タイプ間のもう 1 つの違いは，マルチ・アクションのサポートです。ほとんどのプロトコルは，複数の action セクションをサポートします。現在，Oracle NCA，Web，RTE，General（C 仮想ユーザ），WAP，i モード，VoiceXML のプロトコルがマルチ・アクションをサポートしています。

大部分の仮想ユーザ・タイプでは，記録するたびに新しい仮想ユーザ・スクリプトが作成されます。既存のスクリプトに記録することはできません。しかし，Java，Web，WAP，i モード，Oracle NCA，または RTE の仮想ユーザ・スクリプトを記録する場合は，既存のスクリプトに記録することもできます。

VuGen はさまざまなプロトコルをサポートしているため，以降で説明する記録手順のいくつかは，特定のプロトコルにのみ適用されます。

すべての Java 言語仮想ユーザ（CORBA，RMI，Jacada，および EJB）については，683 ページの「Java プロトコル」を参照してください。

SOA（サービス指向アーキテクチャ）システムでは，デプロイメントの前にアプリケーションとサービスの安定性をテストすることは重要です。**LoadRunner VuGen** を使用すると，基本的な Web サービス・スクリプトを作成できます。**HP LoadRunner with Service Test** および HP の SOA テスト・ツールには，SOA 環境に対する包括的なテスト・ソリューションを作成するのに役立つ追加機構が含まれています。**Service Test** の詳細については，HP の担当窓口にお問い合わせください。

オンライン・リソース

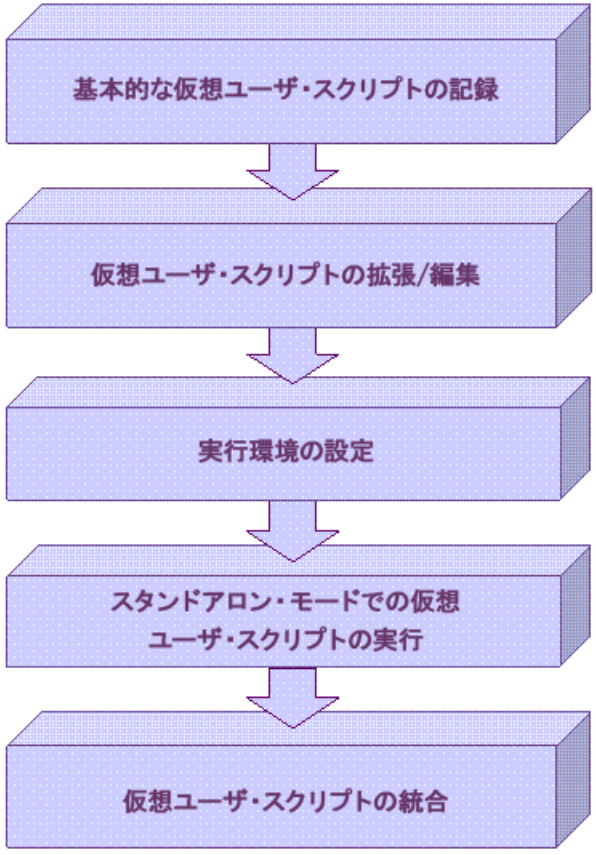
VuGen には，次のオンライン・リソースがあります。

- **『最初にお読みください』**：VuGen の最新のお知らせと情報が提供されます（[**スタート**] メニュー）。
- **オンライン文書**：全マニュアルが PDF 形式で提供されます。オンライン文は Adobe Acrobat Reader を使って読んだり，印刷したりできます（www.adobe.co.jp を参照）。VuGen オンライン文書のアップデートについては，HP のカスタマ・サポート Web サイトをご覧ください（[**ヘルプ**] メニュー）。
- **オンライン関数リファレンス**：仮想ユーザ・スクリプトの作成時に使用できる LoadRunner のすべての API 関数を，その使用例とともにオンラインで参照できます。『オンライン関数リファレンス』のアップデートについては，HP のカスタマ・サポート Web サイトをご覧ください（[**ヘルプ**] メニュー）。
- **コンテキスト・センシティブ・ヘルプ**：VuGen の使用中に生じた疑問をすぐに解決できます。このヘルプは，各ダイアログ・ボックスの説明と，標準的な作業の手順を示します。ウィンドウ上またはウィンドウ内をクリックし，F1 キーを押すと，このヘルプが表示されます。
- **オンライン技術サポート**：普段お使いの Web ブラウザで，HP のカスタマ・サポート Web サイトを開きます。このサイトでは，ナレッジ・ベースの閲覧や記事の投稿，ユーザ・ディスカッション・フォーラムへの参加と検索，サポート要求の送信，パッチやアップデートされたマニュアルのダウンロードなどが行えます。この Web サイトの URL は http://support.hp.com です。
- **サポート情報**：HP のホーム・ページとカスタマ・サポート Web サイトがあるほか，情報やご要望をお寄せいただくための電子メール・アドレスや，世界各地の HP の営業所が掲載されています（[**ヘルプ**] メニュー）。
- **HP の Web サイト**：普段お使いの Web ブラウザで，HP のホーム・ページ（http://www.hp.com）を開きます。このサイトでは，ナレッジ・ベースの閲覧や記事の投稿，ユーザ・ディスカッション・フォーラムへの参加と検索，サポート要求の送信，パッチやアップデートされたマニュアルのダウンロードなどが行えます。
- **LoadRunner チュートリアル**（PDF 形式）：ダウンロード可能な LoadRunner のチュートリアルを提供します。Web アプリケーション用の負荷テスト・スクリプトを作成するための一連の手順を紹介します。

タスク

仮想ユーザ・スクリプトの作成方法 - ワークフロー

次の図は，仮想ユーザ・スクリプトの作成プロセスの概略です。



仮想ユーザ・スクリプトの作成プロセスは次のとおりです。

1. VuGen を使って基本となるスクリプトを記録します。Windows ベースの GUI アプリケーションをテストする場合，またはアプレットや Flash などの複合的な Web 環境をテストする場合は，HP の GUI ベースのツールである WinRunner や QuickTest Professional を使用しなければならないことがあります。
2. 制御フロー・ステートメントやその他の LoadRunner API 関数をスクリプトに追加して，基本となるスクリプトを拡張します。
3. 実行環境を設定します。実効環境の設定には，反復，ログ，タイミングの情報などがあり，これらの設定によってスクリプト実行中における仮想ユーザの振る舞いが決まります。
4. スクリプトの機能を確認し，スタンドアロン・モードで実行します。
5. スクリプトが正常に機能することを確認した後，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）に統合します。詳細については，HP LoadRunner Controller，HP Performance Center，または HP Business Availibility Center の各ユーザーズ・ガイドを参照してください。

ビジネス・プロセス・レポートの作成方法

スクリプト作成の最終段階で，ビジネス・プロセスについて記述されたレポートを作成できます。VuGen によって，スクリプトの情報が Microsoft Word ドキュメントにエクスポートされます。

事前に作成したテンプレートまたは VuGen に用意されているテンプレートを使用して，テスト実行に関するサマリ情報が記載されたレポートを作成できます。

VuGen では，レポートに含める情報のタイプを指定することによって，レポートの内容をカスタマイズできます。

注： ビジネス・プロセス・レポートは，AJAX（Click and Script），Citrix_ICA，Oracle NCA，Oracle Web Applications 11i，PeopleSoft Enterprise，RDP，SAP（Click and Script），SAPGUI，SAP - Web，Web（Click and Script），Web（HTTP/HTML），および Web サービス・プロトコルでのみ使用できます。

このタスクでは，次の手順を実行します。

➤ 55 ページの「ビジネス・プロセス・レポートの作成」

➤ 55 ページの「詳細オプションの設定」

1 ビジネス・プロセス・レポートの作成

［**ファイル**］>［**ビジネス プロセス レポートを作成**］を選択し，ダイアログ・ボックスで設定を行います。ユーザ・インタフェースの詳細については，64 ページの「［ビジネス プロセス レポート］ダイアログ・ボックス」を参照してください。

2 詳細オプションの設定

目次，スナップショット，および Microsoft Word テンプレートなどのレポートの詳細オプションを変更するには，［**詳細**］ボタンをクリックします。ユーザ・インタフェースの詳細については，65 ページの「［詳細］ダイアログ・ボックス」を参照してください。

スクリプトを横に並べて比較する方法

仮想ユーザ・スクリプトは，比較ツールを使用すると横に並べて表示して比較できます。

仮想ユーザ・スクリプトを比較するには，次の手順を実行します。

1 比較対象の最初の仮想ユーザ・スクリプトを開きます。

2［**ツール**］>［**スクリプトと比較**］を選択します。

3 2 つ目の仮想ユーザ・スクリプトを選択します。仮想ユーザ・スクリプトが，新しいウィンドウに横に並んで表示されます。差異は黄色で強調表示されます。

注：比較ツールは，［一般オプション］>［環境］タブで変更できます。詳細については，73 ページの「［一般オプション］ダイアログ・ボックス」を参照してください。

リファレンス

仮想ユーザのタイプ

仮想ユーザのタイプ，説明，およびカテゴリを次の表に示します。

プロトコル	説明	プロトコルのカテゴリ
AJAX（Click and Script）	Asynchronous JavaScript and XML の略語。AJAX では非同期 HTTP 要求を使用するため，Web ページはページ全体を要求する代わりに，わずかな情報を要求できます。	E- ビジネス
Ajax TruClient	Web ブラウザ内のユーザ・アクティビティをエミュレートする最新の JavaScript ベース・アプリケーション（Ajax を含む）向けの高度なプロトコルです。スクリプトは，Mozilla Firefox で対話的に開発できます。	E- ビジネス
Action Message Format（AMF）	Flash Remoting バイナリ・データを HTTP 経由で Flash アプリケーションとアプリケーション・サーバの間でやり取りできるようにする Macromedia 社独自のプロトコルである Action Message Format。	E- ビジネス
C 仮想ユーザ	標準の C ライブラリを使用する一般仮想ユーザ。	ユーザ定義
Citrix_ICA	ユーザが指定のアプリケーションを外部のマシンで実行できるようにするリモート・アクセス・ツール。	アプリケーションの導入ソリューション
COM/DCOM	COM（Component Object Model）- 再利用可能なソフトウェア・コンポーネントを開発するための技術。	分散コンポーネント
DB2 CLI	IBM による DB2 データベース・ファミリへの呼び出しレベル SQL インタフェース。	クライアント / サーバ

プロトコル	説明	プロトコルのカテゴリ
Domain Name Resolution（DNS）	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，手動でスクリプトに関数を追加します。	クライアント / サーバ
EJB テスト	EJB（Enterprise Java Beans）- Java サーバ・コンポーネントの開発とデプロイメントを行うためのアーキテクチャ。	Enterprise Java Beans
Flex	Flex は，企業内および Web を介して Rich Internet Applications（RIA）を作成するためのアプリケーション開発ソリューションです。	E- ビジネス
File Transfer Protocol（FTP）	FTP（File Transfer Protocol）- ある場所から別の場所にネットワークを経由してファイルを転送するシステム。 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。	E- ビジネス
i モード	携帯電話システムでインターネットにアクセスするための，NTT DoCoMo の技術。	
Informix	標準のクライアント・サーバ・アーキテクチャを使用した IBM Informix データベース。	クライアント / サーバ
IMAP（Internet Messaging）	IMAP（Internet Message Application）- クライアントがメール・サーバから電子メールを読み取ることを可能にするプロトコル。	メール・サービス

プロトコル	説明	プロトコルのカテゴリ
Java over HTTP	Java ベースのアプリケーションとアプレットを記録するように設計されています。Web 関数を使用して Java 言語のスクリプトが生成されます。このプロトコルは，HTTP 上で Java リモート呼び出しの記録と再生ができるという点で，ほかの Java プロトコルと区別されます。	
Java Record Replay		
Java テンプレート	プロトコル・レベルのサポートを備えた Java プログラミング言語。	ユーザ定義
Javascript 仮想ユーザ	インターネット・アプリケーションの開発に使用されるスクリプティング言語。	ユーザ定義
Listing Directory Service（LDAP）	電子メール・アプリケーションでサーバから連絡先情報を参照できるように設計されたインターネット・プロトコル。	E- ビジネス
Media Player（MMS）	Microsoft の MMS プロトコルを使用したメディア・サーバからのデータ・ストリーミング。 **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。	ストリーミング
Microsoft .NET	Microsoft.NET クライアント - サーバ・テクノロジの記録をサポートします。	クライアント / サーバ，分散コンポーネント，E- ビジネス
Microsoft リモート・デスクトップ・プロトコル（RDP）	Microsoft Remote Desktop Connection を使用して外部のマシンでアプリケーションを実行するためのリモート・アクセス・ツール。	アプリケーションの導入ソリューション
MAPI（MS Exchange）	MAPI（Messaging Application Programming Interface）- アプリケーションで電子メール・メッセージの送受信を可能にするためのプロトコル。	メール・サービス

プロトコル	説明	プロトコルのカテゴリ
MS SQL Server	Dblib インタフェースを使用する Microsoft の SQL Server。	クライアント / サーバ
マルチメディア・メッセージング・サービス（MMS）	モバイル・デバイス間での MMS メッセージの送信に使用されるメッセージング・サービス。	ワイヤレス
ODBC	ODBC（Open Database Connectivity）- データベース・アクセスのための共通のインタフェースを提供するプロトコル。	クライアント / サーバ
Oracle（2 層）	標準の 2 層クライアント・サーバ・アーキテクチャを使用した Oracle データベース。	クライアント / サーバ
Oracle NCA	Java クライアント，Web サーバ，データベースで構成される，Oracle の 3 層アーキテクチャ・データベース。	ERP/CRM
Oracle Web Applications 11i	Web 経由でアクションを実行する Oracle Applications のインタフェース。この仮想ユーザ・タイプは，Mercury API と JavaScript の両方のレベルでアクションを検出します。	ERP/CRM
Peoplesoft Enterprise	PeopleSoft 8 エンタープライズ・ツールをベースとした ERP（Enterprise Resource Planning）システム。	ERP/CRM
Peoplesoft-Tuxedo	Tuxedo トランザクション・プロセッシング・モニタをベースとし，自動相関を含む，ERP（Enterprise Resource Planning）システム。	ERP/CRM
POP3（Post Office Protocol）	1 台のコンピュータでメール・サーバから電子メールを取得できるようにするためのプロトコル。	メール・サービス
Real	メディア・サーバからストリーミング・データを転送するために使用されるプロトコル。	ストリーミング
SAP（Click and Script）	GUI またはユーザ・アクション・レベルでのブラウザと SAP サーバとの間の通信のエミュレーション。	ERP/CRM

プロトコル	説明	プロトコルのカテゴリ
SAPGUI	SAPGUI for Windows クライアントを使用して主要なビジネス・プロセスおよび管理プロセスを統合する ERP（Enterprise Resource Planning）システム。	ERP/CRM
SAP-Web SAPWeb	SAP Portal または Workplace クライアントを使用して重要なビジネス・プロセスおよび管理プロセスを統合する ERP（Enterprise Resource Planning）システム。	ERP/CRM
Siebel-Web	CRM（Customer Relationship Management）アプリケーション。	ERP/CRM
Silverlight	トランスポート・レベルでユーザ・アクティビティをエミュレートする Silverlight ベース・アプリケーション用のプロトコル。アプリケーションで使用される WSDL ファイルを自動的にインポートおよび設定することで，高レベルのスクリプトを生成できます。	E- ビジネス
Simple Mail Protocol（SMTP）	SMTP（Simple Mail Transfer Protocol）- メールを特定のマシンに配布するためのシステム。	メール・サービス
Sybase Ctlib	Ctlib インタフェースを通じて呼び出されるクライアント・サーバ・アーキテクチャ・データベース。	クライアント / サーバ
Sybase Dblib	Dblib インタフェースを通じて呼び出されるクライアント・サーバ・アーキテクチャ・データベース。	クライアント / サーバ
ターミナル・エミュレーション（RTE）	文字ベースのアプリケーションに対して入力の送信と出力の受信を行うユーザのエミュレーション。	レガシ
Tuxedo	Tuxedo トランザクション・プロセッシング・モニタ。	ミドルウェア
VB スクリプト仮想ユーザ	VBscript（Visual Basic Scripting Edition 言語）- Web ブラウザに表示されるドキュメントのプログラミングに使用されるスクリプティング言語。	ユーザ定義
Visual Basic テンプレート	Visual Basic 言語で記述された仮想ユーザ・スクリプト。	ユーザ定義

プロトコル	説明	プロトコルのカテゴリ
WAP	WAP（Wireless Application Protocol）- モバイル・デバイスとコンテンツ・プロバイダとの間で Web ベースのワイヤレス通信を行うために使用されるプロトコル。	ワイヤレス
Web（Click and Script）	GUI またはユーザ・アクション・レベルでのブラウザと Web サーバとの間の通信のエミュレーション。	E- ビジネス
Web（HTTP/HTML）	HTTP または HTML レベルでのブラウザと Web サーバとの間の通信のエミュレーション。	E- ビジネス
Web サービス / SOA	アプリケーションどうしが World Wide Web 経由で互いに通信するためのプログラミング・インタフェース。	E- ビジネス
Windows Sockets	Windows プラットフォーム用の標準のネットワーク・プログラミング・インタフェース。	クライアント / サーバ

注： さまざまなプロトコルを実行するには，グローバル・ライセンスか希望のプロトコルのライセンスを取得する必要があります。詳細については，LoadRunner Launcher（［**スタート**］>［**すべてのプログラム**］>［**HP LoadRunner**］>［**LoadRunner**］）で［**設定**］>［**LoadRunner ライセンス**］を選択します。

キーボードのショートカット

Virtual User Generator で使用できるキーボード・ショートカットのリストを次に示します。

ALT+F8	現在のスナップショットを比較（Web 仮想ユーザのみ）
ALT+INS	新規ステップの作成
CTRL+A	すべて選択
CTRL+C	コピー
CTRL+F	検索
CTRL+G	指定行へ移動
CTRL+H	置換
CTRL+N	新規作成
CTRL+O	開く
CTRL+P	印刷
CTRL+S	保存
CTRL+V	貼り付け
CTRL+X	切り取り
CTRL+Y	やり直し
CTRL+Z	元に戻す
CTRL+F7	記録オプション
CTRL+F8	相関を検索
CTRL+SHIFT + SPACE	関数の構文を表示（ステートメントの自動補完）
CTRL+SPACE	Complete ウィザード（関数名を完成）
F1	ヘルプ
F3	次を検索（下方）

SHIFT+F3	次を検索（上方）
F4	実行環境の設定
F5	仮想ユーザを実行
F6	表示枠間を移動
F7	EBCDIC 変換ダイアログを表示（WinSock スクリプトの場合）
F9	ブレークポイントの切り替え
F10	仮想ユーザをステップごとに実行

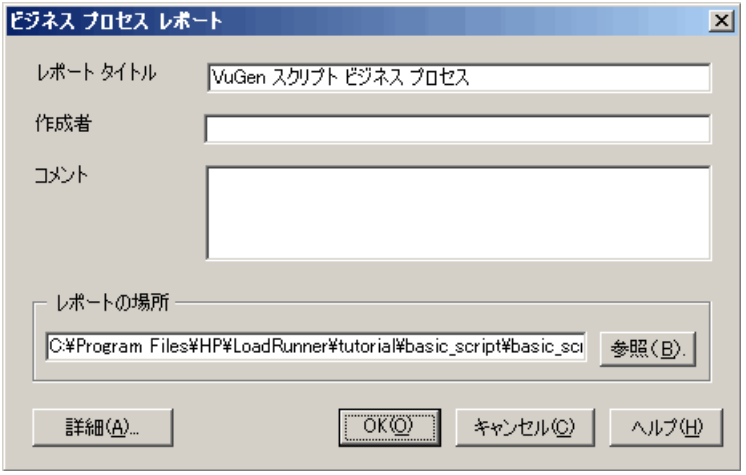
メイン・ユーザ・インタフェース

このセクションの内容

➤［スナップショット］表示枠（98 ページ）

［ビジネス プロセス レポート］ダイアログ・ボックス

このダイアログ・ボックスでは，ビジネス・プロセス・レポートを作成できます。



利用方法	**［ファイル］>［ビジネス プロセス レポートを作成］**
関連タスク	54 ページの「ビジネス・プロセス・レポートの作成方法」

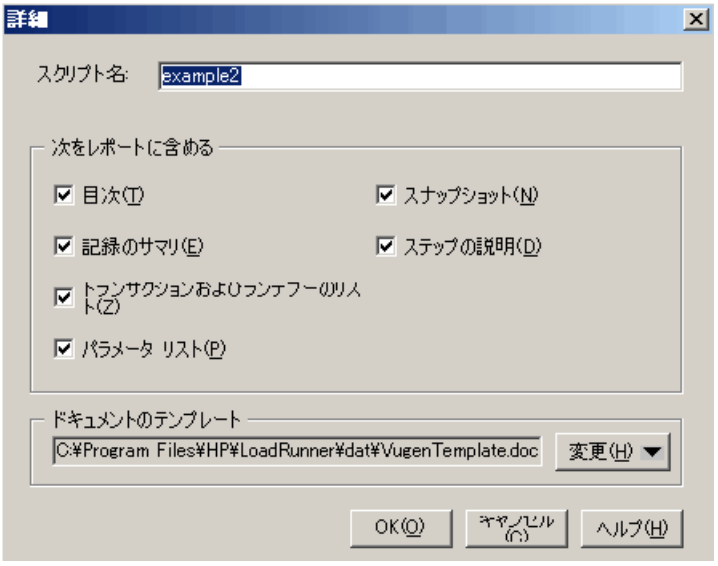
ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
詳細(A)...	［詳細］ダイアログ・ボックスが開きます。
作成者	自分の名前。
コメント	レポートに表示される追加のコメント。

UI 要素	説明
レポートの場所	レポートの場所。 **標準設定値**：script ディレクトリ。
レポート タイトル	レポートのタイトル。

［詳細］ダイアログ・ボックス

このダイアログ・ボックスでは，ビジネス・プロセス・レポートを作成するときの詳細オプションを指定できます。



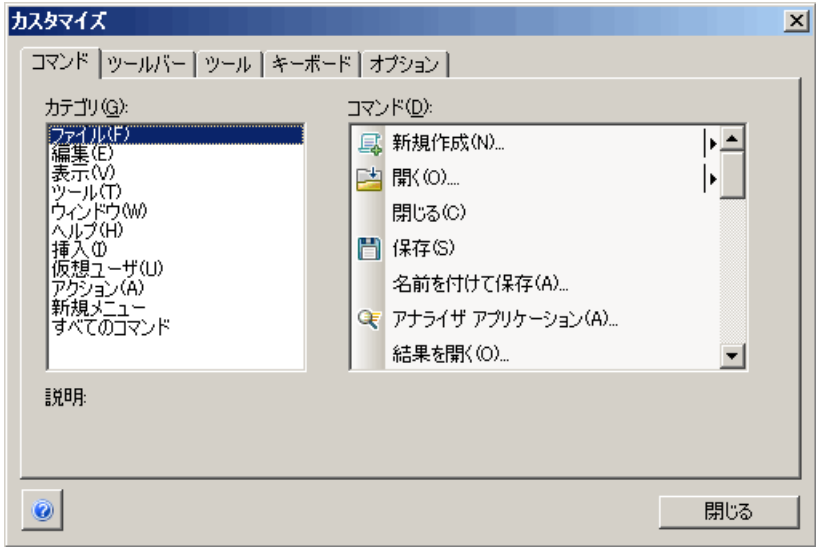
利用方法	**［ファイル］＞［ビジネス プロセス レポートを作成］＞［詳細］**
関連タスク	54 ページの「ビジネス・プロセス・レポートの作成方法」

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
ドキュメントのテンプレート	レポートに使用するテンプレートのパスおよびファイル名。標準設定のテンプレートは，LoadRunner の **dat** フォルダにあります。 レポート・テンプレートを変更するには，［**変更**］を選択し，.doc 拡張子を持つ新しいテンプレートを指定します。新しいテンプレートを作成する場合は，新しいテンプレートの基礎として既存のテンプレートを使用することをお勧めします。これにより，必要なブックマークおよびスタイルが，新しいテンプレートでも維持されるようになります。
パラメータ リスト	スクリプトに対して定義されているすべてのパラメータのリスト。このリストは，［パラメータ リスト］ダイアログ・ボックス（［**仮想ユーザ**］> ［**パラメータ リスト**］）に表示されるパラメータに対応します。 **標準設定値**：有効。
記録のサマリ	［タスク］リストで［記録のサマリ］リンクをクリックすると表示される，記録セッションのサマリ。 **標準設定値**：有効。
スクリプト名	スクリプトの .usr ファイルの名前。
スナップショット	ステップの名前と説明の横に表示される，記録されたステップの実際のスナップショット。 **注**：Oracle NCA および Web サービス・レポートには，スナップショットは含まれていません。 **標準設定値**：有効。
ステップの説明	ツリー・ビューに表示される各ステップの簡単な説明。 **標準設定値**：有効。
目次	レポート上のほかのすべての内容のページ番号を示す目次。特定のオプションを無効にすると，そのオプションに対応する項目は目次に表示されません。 **標準設定値**：有効。
トランザクションおよびランデブーのリスト	スクリプトに定義されているすべてのトランザクションおよびランデブーの包括的なリスト。 **標準設定値**：有効。

［カスタマイズ］ダイアログ・ボックス

このダイアログ・ボックスでは，VuGen のメイン・ツールバーとメイン・メニューをカスタマイズできます。



利用方法	［ツール］>［カスタマイズ］

［コマンド］タブ

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
［カテゴリ］表示枠	異なるコマンドを含む各カテゴリのリスト。カテゴリを選択すると，そのコマンドが［コマンド］表示枠に表示されます。
［コマンド］表示枠	コマンドのリスト。コマンドをクリックするとその説明が表示されます。コマンドをツールバーに追加するには，［コマンド］表示枠のコマンドをツールバーの目的の場所にドラッグします。

[ツールバー] タブ

ユーザ・インタフェース要素の説明は次のとおりです。

表示するツールバーごとにチェック・ボックスを選択します。

UI 要素	説明
すべてリセット(A)	すべてのツールバーの表示設定が標準設定に戻ります。標準設定に戻る前に，VuGen は警告を表示します。
リセット(R)	左側の表示枠で選択したツールバーの表示設定が標準設定に戻ります。
新規作成(N)	表示されているツールバーのリストにカスタム・ツールバーが追加されます。このツールバーを，VuGen ウィンドウの必要な位置にドラッグします。その後，[コマンド] タブを使用してアイコンをツールバーに追加します。
テキスト ラベルを表示	選択したツールバーのテキスト・ラベルが表示されます。たとえば，[記録] ツールバーでこのオプションを有効にすると，[停止] ボタンに「停止」という文字が表示され，[実行] ボタンに「実行」という文字が表示されます。

[ツール] タブ

[**ツール**] タブでは，VuGen ツールバーにユーザ定義コマンドを追加できます。通常，コマンドは，実行可能ファイル，バッチ・ファイル，または *.com* ファイルです。また，VuGen の中から特定のドキュメントや PDF ファイルを開くように指定することもできます。1 つまたは複数のツールを [ツール] タブの [ツール] リストで定義できます。これらのツールは [コマンド] タブの [**ツール**] メニューの下に一覧表示されます。必要なコマンドを [ツール] メニューから選択し，それを必要なツールバーにドラッグします。たとえば，Windows の電卓 *calc.exe* を VuGen から開くことができるように指定できます。

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
引数	指定したファイルと一緒に実行される実行時の引数。
コマンド	実行可能ファイルまたはバッチ・ファイルの名前。実行可能でないファイルを指定する場合は，ファイル拡張子がマシンにインストールされているプログラムに関連付けられていることを確かめます。
初期ディレクトリ	カスタム・ツールを実行する初期ディレクトリ。
[メニューの内容] リスト	追加された項目のリスト。[**新規作成**]，[**削除**]，[**項目を上に移動**]，および [**項目を下に移動**] ボタンを使用してこのリストを管理します。

[キーボード] タブ

[キーボード] タブでは，キーボード・ショートカットをメニュー・コマンドに割り当てることができます。ショートカットは，標準コマンドとユーザ定義コマンドの両方に割り当てることができます。

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
	[**新規ショートカット キーを押す**] のショートカットを該当のコマンドに割り当てます。
	選択したショートカットが [**現在のキー設定**] リストから削除されます。
カテゴリ	コマンド・カテゴリのリスト。
コマンド	コマンドのリスト。
現在のキー設定	選択したコマンドに割り当てられているショートカット・キーのリスト。
新規ショートカット キーを押す	該当のフィールドにカーソルを置き，任意のキーを押します。[**割り当て**] を押すと，そのエントリがコマンドに割り当てられます。

[オプション] タブ

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
使用履歴データをリセットする(R)	カスタム・コマンドがすべて削除され，メニューとツールバーに表示されるコマンドのセットが標準設定に戻ります。明示的なカスタマイズは元に戻しません。
個人設定のメニューとツールバー	[**最近使用したコマンドからメニューに表示する**]：最近使用したコマンドがメニューの先頭に表示されます（標準設定で有効）。 ➤ [**少し時間が経過したら，すべてのメニューを表示する**]：展開可能なメニューの場合，少し時間をおいてからメニュー全体を表示します。
ツールバー	すべてのツールバーに適用されるいくつかの表示オプションを次に示します。 ➤ [**ツールバーにヒントを表示する**]：カーソルをツールバー・ボタンの上に移動したときにヒントを表示します（標準設定では有効）。 ➤ [**ヒントにショートカット キーを表示する**]：ツールヒントにキーボード・ショートカットを表示します（標準設定では無効）。 ➤ [**大きいアイコン**]：ツールバーに大きなボタンを表示します（標準設定では無効）。

[検索] ダイアログ・ボックス

このダイアログ・ボックスを使用して，VuGen コード内の文字列の検索ができます。

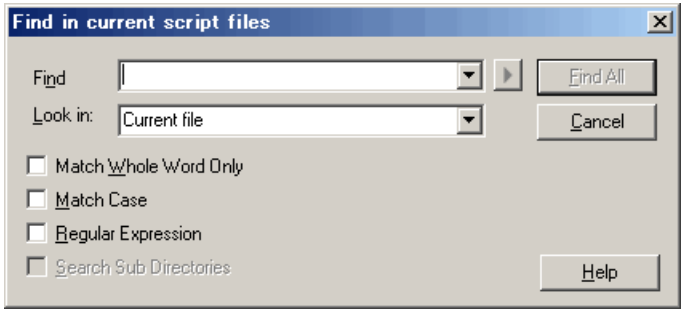
利用方法	［編集］>［検索］

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
検索する文字列	検索する単語を指定します。
完全に一致する単語だけを検索する	別の単語の一部として出現する箇所を除外し，単語として完全に一致する箇所のみを検索します。
大文字と小文字を区別する	検索の際に大文字と小文字の違いを区別します。
正規表現	検索文字列が正規表現であることを示します。
検索方向	検索する方向（［**上へ**］または［**下へ**］）を選択します。
検索(F)	［**検索する文字列**］ボックスのテキストが次に出現する箇所を検索します。

［現在のスクリプト ファイルで検索］ダイアログ・ボックス

このダイアログ・ボックスを使用して，VuGen コード内の文字列の検索と置き換えができます。



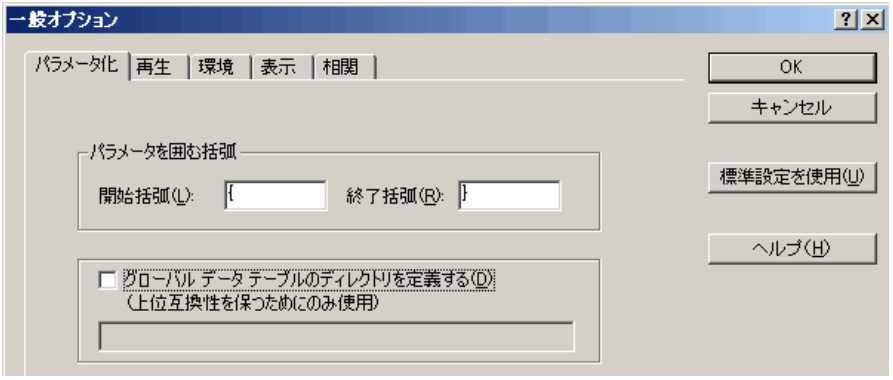
利用方法	［編集］>［現在のスクリプト ファイルで検索］

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
検索する文字列	検索する単語を指定します。
次で検索	**現在のファイル**またはすべての**アクション表示枠のファイル**，左側表示枠のファイル・リストに表示されているすべてのファイルが該当します。
完全に一致する単語だけを検索する	別の単語の一部として出現する箇所を除外し，単語として完全に一致する箇所のみを検索します。
大文字と小文字を区別する	検索の際に大文字と小文字の違いを区別します。
正規表現	検索文字列が正規表現であることを示します。
検索方向	検索する方向（[**上へ**] または［**下へ**]）を選択します。
Search Sub Directories	スクリプトのすべてのサブディレクトリが検索されます。
	［**検索**］ボックスのテキストに一致する項目がすべて検索されます。

［一般オプション］ダイアログ・ボックス

このダイアログ・ボックスでは，表示，パラメータ，および相関オプションなどのさまざまな一般オプションを設定できます。



利用方法	［ツール］ > ［一般オプション］

［パラメータ化］タブ

UI 要素	説明
パラメータを囲む括弧	パラメータを仮想ユーザ・スクリプトに挿入すると，VuGen によってパラメータ名の前後にパラメータ用の括弧が追加されます。1 文字または複数の文字で構成される文字列を指定して，パラメータの括弧のスタイルを変更できます。スペース以外のすべての文字を使用できます。
グローバル データ テーブルのディレクトリを定義する	このオプションは，VuGen の前のバージョンとの互換性を保つためにだけ提供されています。4.51 またはそれ以前のバージョンでは，新しいデータ・テーブルを作成するときに，ローカルかグローバルかを指定していました。ローカルのテーブルは現在の仮想ユーザ・スクリプトのディレクトリに保存され，スクリプトを実行している仮想ユーザだけが使用できます。グローバルなテーブルはすべての仮想ユーザ・スクリプトで使用できます。グローバル・ディレクトリはローカル・ドライブ上にもネットワーク・ドライブ上にも置けます。スクリプトを実行するすべてのマシンから，グローバル・ディレクトリを利用できることを確認してください。グローバル・テーブルの場所は，このダイアログ・ボックスを使っていつでも変更できます。 VuGen の新しいバージョンでは，［パラメータのプロパティ］ダイアログ・ボックスまたは［パラメータ リスト］ダイアログ・ボックスの中でデータ・テーブルの場所を指定します。VuGen は，標準設定のスクリプト・ディレクトリや，ネットワーク上の別のディレクトリなど，指定した任意の場所のデータを取得できます。詳細については，1140 ページの「データ・ファイル」を参照してください。 このオプションを有効にするには，［**グローバル データ テーブルのディレクトリを定義する**］チェック・ボックスを選択し，グローバル・データ・テーブルがあるディレクトリを指定します。 標準設定では，［グローバル データ テーブルのディレクトリを定義する］オプションは無効になっています。

［再生］タブ

仮想ユーザ・スクリプトは表示実行モードまたは非表示実行モードで実行できます。表示実行モードで実行すると，VuGen によって仮想ユーザ・スクリプト内の現在実行されている行が強調表示されます。各ステップの様子がさらによく見えるように，このモードの遅延を設定することができます。非表示実行モードで実行すると，VuGen によって仮想ユーザ・スクリプトが実行されますが，現在実行されている行は強調表示されません。

UI 要素	説明
再生後	再生後の VuGen の動作を指示します。 ➤ **［再生の前のビュー］**：再生前のビューに戻ります。標準で選択されています。 ➤ **［再生のサマリ］**：ワークフロー・ウィザードの［再生のサマリ］ウィンドウに直接移動します。 ➤ **［テスト結果］**：［テスト結果］が開きます（再生後に［**表示**］>［**テスト結果**］を選択しても開くことができます）。
デバッグ	➤ **［表示実行遅延］**：コマンドを実行する遅延間隔をミリ秒単位で指定します。標準の遅延時間は，0 です。 ➤ **［Action のセクションのみで関数を表示実行する］**：Action セクションの内容だけ（init や end セクションの内容を除く）が表示実行モードで実行されます。標準設定では有効になっています。
結果ディレクトリ	**［結果ディレクトリの指定を求める］**：VuGen からスクリプトを実行する前に結果ディレクトリを指定するよう求められます。標準設定では無効になっています。このオプションを選択していない場合は，VuGen によって *result1* という名前が自動的にディレクトリに付けられます。スクリプトの以降の実行結果は，別の結果ファイルを指定しない限り，前回の結果ファイルに自動的に上書きされます。結果は必ずスクリプトのサブディレクトリに格納されます。

［環境］タブ

UI 要素	説明
自動回復	自動回復オプションを使用することにより，クラッシュまたは停電が発生したときにスクリプトの設定を復元できます。自動回復オプションを使用するには，［自動回復情報を次の間隔で保存する］チェック・ボックスを選択し，保存の実行間隔を分単位で指定します。
比較ツール	2 つのスクリプトを比較するときに使用する比較ツールを選択できます。VuGen には，標準の比較ツールが含まれています。2 つのスクリプトを横に並べて表示するには，［**ツール**］>［**スクリプトと比較**］を選択します。
エディタ	エディタのオプションで，単語の自動補完や関数構文の自動表示を行う VuGen のステートメントの補完機能を設定できます。 ➤ **［関数構文の自動表示］**：関数の開始括弧を入力すると，関数の構文，引数およびプロトタイプが自動的に表示されます。標準設定では，このオプションは有効になっています。これを無効にしても，エディタで開始括弧を入力した後に，Ctrl キーと Shift キーを押しながらスペース・キーを押すか，［**編集**］>［**関数の構文を表示**］を選択すれば，ローカルで有効にできます。 ➤ **［単語の自動補完］**：関数の最初のアンダーバーを入力すると，その関数のプレフィックスに一致する使用可能なすべての関数の構文と説明を示すリストが開きます。標準設定では，このオプションは有効になっています。これを無効にしても，エディタで入力しているときに，Ctrl キーを押しながらスペース・キーを押すか，［**編集**］>［**単語入力候補**］を選択すれば，ローカルで有効にできます。 ➤ フォントの選択(F)... **:**［フォントの選択］ダイアログ・ボックスが開きます。使用するフォント，スタイル，サイズを選択します。使用できるフォントは，固定幅のフォント（Courier，Lucida Console，FixedSys など）だけです。

［表示］タブ

このタブには，Web 仮想ユーザ・スクリプトにのみ適用されるオプションがあります。これらのオプションを使用したデバッグ方法の詳細については，135 ページの「Web 仮想ユーザのデバッグ機能」を参照してください。

UI 要素	説明
スクリプトの実行中にレポートを作成する	結果サマリ・レポートを生成するように仮想ユーザに指示します。標準設定では，このオプションは有効になっています。スクリプトの実行後にレポートを開くには，［**表示**］>［**テスト結果**］を選択します。
再生時に実行時ビューアを表示する	実行時ビューアが有効になります。スクリプトの実行終了時に実行時ビューアを最小化するには，［**ウィンドウを自動整列する**］オプションを選択します。標準設定ではこのオプションは無効です。

［相関］タブ

これらのオプションを設定すると，仮想ユーザ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，実行する比較のタイプと区切り文字として扱う文字を指定できます。

UI 要素	説明
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 が実行できるようになります。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。
差異が X 文字以下の場合は無視する	相関を行う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。 **標準設定値**：4 文字。

UI 要素	説明
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が表示されます。
次を使ってスナップショット間の違いを検索する	比較メソッドを選択します。 ➤ **[HTML 比較]**：HTML コードの差異のみが表示されます。 ➤ **[テキスト比較]**：すべてのテキスト，HTML，バイナリの差異が表示されます。 **注**：通常であれば，標準設定のメソッドである［HTML 比較］を使用して作業することをお勧めします。スクリプトに HTML 以外のタグが含まれている場合，［テキスト比較］メソッドを使用できます。

［Citrix の表示］タブ

（Citrix 仮想ユーザ・スクリプトのみ）

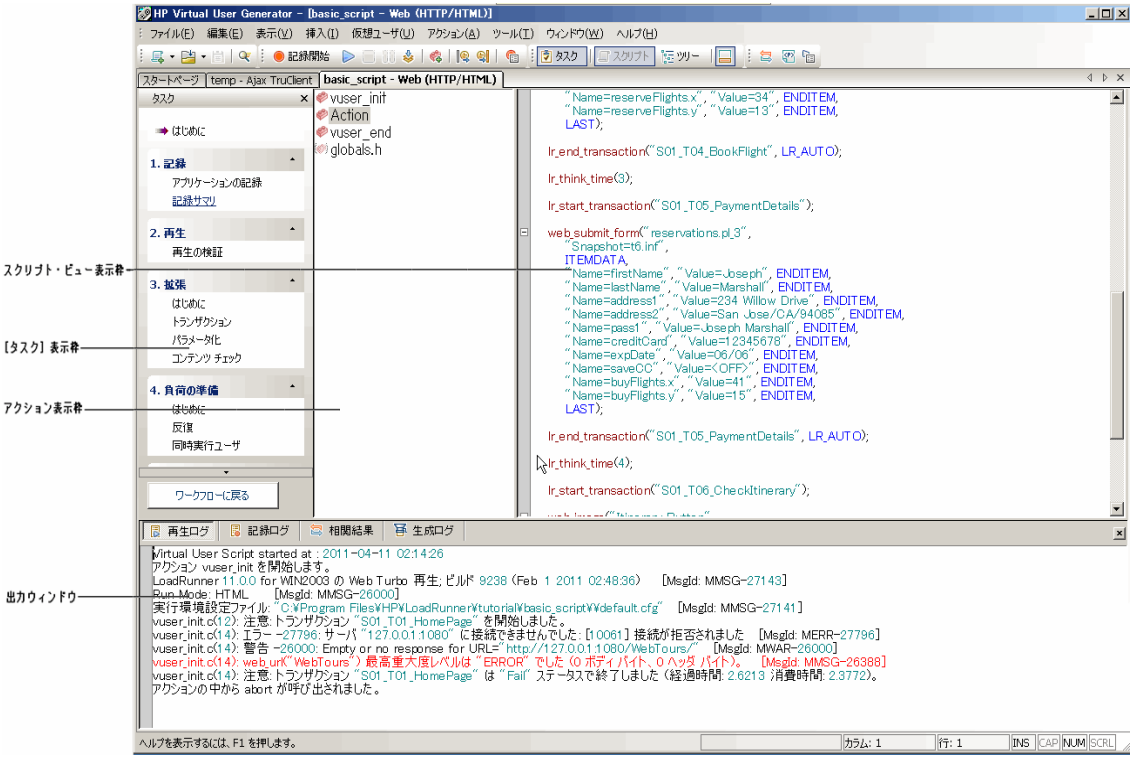
Citrix 仮想ユーザ・スクリプトを実行する前に，再生中に使用される表示オプションをいくつか設定できます。これらのオプションを設定すると，サーバの負荷が大きくなりますが，セッションのデバッグと分析には役立ちます。

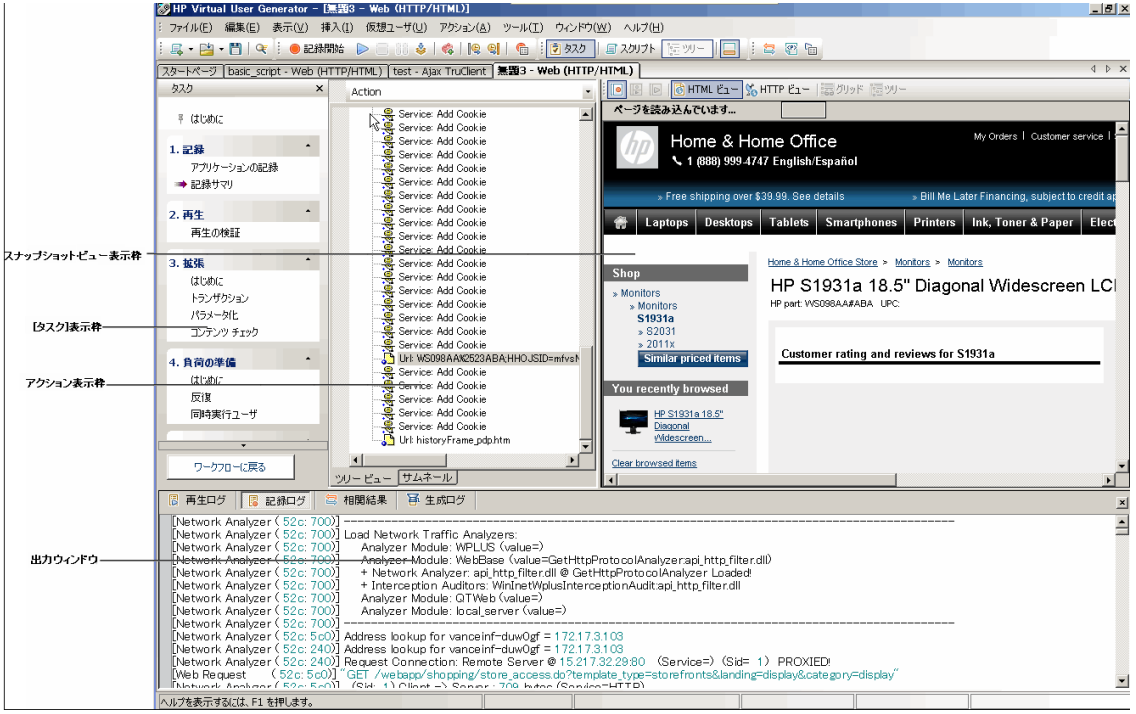
UI 要素	説明
再生中にクライアントを表示する	仮想ユーザ・スクリプトの再生中に Citrix クライアントが表示されます。
ビットマップの選択を表示するポップアップメッセージ	スナップショットの中で対話形式で作業を開始するときにポップアップ・メッセージが表示されます。ビットマップまたはテキストを選択する前に，右クリック・メニュー・オプションで［**ビットマップ同期を挿入**］または［**テキスト取得を挿入**］を選択すると，このメッセージが表示されます。

VuGen のメイン・ユーザ・インタフェース

VuGen には，スクリプトの内容を調べるためのビューが複数用意されています。1 つ目はテキスト形式のスクリプト・ビュー，2 つ目はアイコン形式のツリー・ビュー，3 つ目はアイコン形式のサムネイル・ビューです。

スクリプト・ビューとツリー・ビューは，ほとんどの仮想ユーザのタイプで利用できます。多くのプロトコルでは，サムネイル・ビューもサポートされています。




スナップショットビュー表示枠
[タスク]表示枠
アクション表示枠
出力ウィンドウ
HP Virtual User Generator - [無題3 - Web (HTTP/HTML)]
ファイル(F) 編集(E) 表示(V) 挿入(I) 仮想ユーザ(U) アクション(A) ツール(T) ウィンドウ(W) ヘルプ(H)
記録開始
タスク
スクリプト
ツリー
スタートページ
basic_script - Web (HTTP/HTML)
test - Ajax TruClient
無題3 - Web (HTTP/HTML)
タスク
はじめに
1. 記録
アプリケーションの記録
記録サマリ
2. 再生
再生の検証
3. 拡張
はじめに
トランザクション
パラメータ化
コンテンツ チェック
4. 負荷の準備
はじめに
反復
同時実行ユーザ
ワークフローに戻る
Action
Service: Add Cookie
Url: historyFrame_pdp.htm
ツリー ビュー
サムネール
HTML ビュー
HTTP ビュー
グリッド
ツリー
ページを読み込んでいます...
Home & Home Office
1 (888) 999 4747 English/Español
My Orders | Customer service
Free shipping over $39.99. See details
Bill Me Later Financing, subject to credit ap
Laptops
Desktops
Tablets
Smartphones
Printers
Ink, Toner & Paper
Shop
Monitors
S1931a
S2031
2011x
Similar priced items
You recently browsed
HP S1931a 18.5" Diagonal Widescreen...
Clear browsed items
Home & Home Office Store > Monitors > Monitors
HP S1931a 18.5" Diagonal Widescreen LCI
HP part: WS098AA#ABA UPC:
Customer rating and reviews for S1931a
再生ログ
記録ログ
相関結果
生成ログ
[Network Analyzer (52c: 700)] ----------
[Network Analyzer (52c: 700)] Load Network Traffic Analyzers:
[Network Analyzer (52c: 700)] Analyzer Module: WPLUS (value=)
[Network Analyzer (52c: 700)] Analyzer Module: WebBase (value=GetHttpProtocolAnalyzer:api_http_filter.dll)
[Network Analyzer (52c: 700)] + Network Analyzer: api_http_filter.dll @ GetHttpProtocolAnalyzer Loaded!
[Network Analyzer (52c: 700)] + Interception Auditors: WinInetWplusInterceptionAudit:api_http_filter.dll
[Network Analyzer (52c: 700)] Analyzer Module: QTWeb (value=)
[Network Analyzer (52c: 700)] Analyzer Module: local_server (value=)
[Network Analyzer (52c: 700)] ----------
[Network Analyzer (52c: 5c0)] Address lookup for vanceinf-duw0gf = 172.17.3.103
[Network Analyzer (52c: 240)] Address lookup for vanceinf-duw0gf = 172.17.3.103
[Network Analyzer (52c: 240)] Request Connection: Remote Server @ 15.217.32.29:80 (Service=) (Sid= 1) PROXIED!
[Web Request (52c: 5c0)] "GET /webapp/shopping/store_access.do?template_type=storefronts&landing=display&category=display"
ヘルプを表示するには、F1 を押します。

ユーザ・インタフェース要素の説明は次のとおりです。

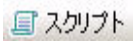
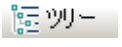
UI 要素	説明
[ツリー ビュー] タブ（ツリー・ビューのみ）	仮想ユーザ・スクリプトがアイコン形式で表示され，各ステップが異なるアイコンで表されます。 ツリーの上にあるドロップダウン・リストを使用して，表示するスクリプトのセクションを選択します。 ステップをドラッグして目的の位置に移動できます。また，ツリー階層の中で，既存のステップの間にステップを追加することもできます。
[サムネイル] タブ（ツリー・ビューのみ）	Web，SAPGUI，Citrix など一部の仮想ユーザ・タイプでは，スナップショットをサムネイルで表示できます。 標準設定では，スクリプト内の主要なステップのみがサムネイル・ビューに表示されます。すべてのサムネイルを表示するには，［**表示**］>［**すべてのサムネイルを表示**］を選択します。スクリプト内の全ステップのサムネイルが表示されます。 反復が複数の場合，最後の反復の再生のサムネイルが表示されます。特定の反復のサムネイルを表示するには，［**表示**］>［**スナップショット**］>［**反復の選択**］を選択し，目的の反復を選びます。
[アクション] 表示枠	スクリプトが主要なセクションに分かれています。セクションをクリックすると，その内容がほかの表示枠に表示されます。標準設定では，この表示枠はスクリプト・ビューで表示されます。この表示枠の表示形式を変更するには，［**表示**］>［**アクション**］を選択し，目的のオプションを選択します。

UI 要素	説明
［スクリプト］表示枠（スクリプト・ビューのみ）	［スクリプト］表示枠では，スクリプトに記録または挿入された実際の API 関数を表示できます。上級ユーザは，このビューで C や仮想ユーザ API の関数，あるいはフロー制御ステートメントを追加することで，スクリプトを編集できます。 ［**挿入**］>［**新規ステップ**］コマンドを使用してスクリプトにステップを追加できます。または，単語の補完機能や関数構文の表示機能を使用して，関数を手動で入力することもできます。詳細については，46 ページの「VuGen コードのツール」を参照してください。 スクリプト・ビューの表示中に仮想ユーザ・スクリプトを変更すると，それに応じて仮想ユーザ・スクリプトのツリー・ビューも変更されます。テキスト形式のスクリプトに行われた変更を解釈できなかった場合は，スクリプト・ビューがツリー・ビューやサムネイル・ビューに変換されません。
［スナップショット］表示枠	スナップショットは，現在のステップを視覚的に表したものです。プロトコルのタイプに応じて，選択したステップに関するさまざまなデータがスナップショットに表示されます。 各種プロトコルの［スナップショット］表示枠に関するユーザ・インタフェース情報の詳細については，98 ページの「［スナップショット］表示枠」を参照してください。 VuGen は，記録中に基本となるスナップショットをキャプチャし，再生中にも新たな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，スクリプトを実行するために相関させる必要がある動的な値を見つけることができます。 スクリプトを再生するたびに，VuGen はスクリプトの結果ディレクトリ（**Iteration1**，**Iteration2**，など）に新たな再生時のスナップショットを保存します。 標準では，VuGen は記録時のスナップショットと，最初に再生されたときのスナップショットを比較します。別のスナップショットを選択して比較することもできます。特定の再生時のスナップショットを選択するには，［**表示**］>［**スナップショット**］>［**反復の選択**］を選択します。結果セットを選択して，［**OK**］をクリックします。 次のツールバー・ボタンを使用して，さまざまなスナップショット・ウィンドウを表示または非表示にすることができます。
	**記録時のスナップショットのみを表示。**

UI 要素	説明
	**記録時と再生時の両方のスナップショットを表示。**
	**再生時のスナップショットのみを表示。**
出力ウィンドウ	メイン・ウィンドウの下部にある表示枠で，再生時および記録時に収集したデータが表示されます。
［タスク］表示枠	機能スクリプトの作成に必要なタスクの一覧が表示されます。一覧でタスクをクリックすると，ウィザードでそのステップが開きます。現在のタスクは矢印で示されます。詳細については，40 ページの「［タスク］表示枠 の概要」を参照してください。
［詳細］ツールバー	
	**新規パラメータの挿入。**
	**［新規ステップを挿入］**：新規ステップがスクリプトに挿入されます（ツリー・ビュー）。タスクの詳細については，163 ページの「ステップをスクリプトに挿入する方法」を参照してください。
	**［テスト結果を表示］**：［テスト結果］ウィンドウを開きます。詳細については，182 ページの「［テスト結果］ウィンドウ」を参照してください。
［デバッグ ツール］ツールバー	
	**ステップ。**
	**ブレークポイントの設定 / 解除。**
	**ブレークポイントの有効化 / 無効化。**
［編集］ツールバー	
	**切り取り。**
	**コピー。**

UI 要素	説明
	**貼り付け。**
	**元に戻す。**
	**選択範囲をコメント アウトする。**
	**選択範囲のコメント アウトを解除する。**
	**インデントを深くする。**
	**インデントを浅くする。**
［記録］ツールバー	
	**［アプリケーションの記録開始］**：記録セッションが初期化されます。
	**［実行］**：スクリプトが実行 / 再生されます。
	**［停止］**：スクリプトが停止します。
	**［一時停止］**：実行中のスクリプトが一時停止します。
	**［コンパイル］**：スクリプトがコンパイルされます。
	**［新規アクションの作成］**：スクリプトに新規アクションのセクションが作成されます。
	**［トランザクション開始マーカを挿入］**：指定した場所にトランザクション開始マーカが挿入されます。
	**［トランザクション終了マーカを挿入］**：指定した場所にトランザクション終了マーカが挿入されます。
	**［記録オプションの編集］**：［記録オプション］ダイアログ・ボックスが開きます。詳細については，313 ページの「記録オプション」を参照してください。

UI 要素	説明
	**[ランデブーを挿入]**：このオプションは，記録する場合にフローティング・ツールバーでのみ使用できます。ランデブー・ポイントの詳細については，152 ページの「ランデブー・ポイント」を参照してください。
	**[コメントを挿入]**：スクリプトにコメントが挿入されます。このオプションは，記録する場合にフローティング・ツールバーでのみ使用できます。
	**[テキスト チェックを挿入]**：このオプションは，Web（HTTP/HTML）または関連するスクリプトを記録する場合にフローティング・ツールバーでのみ使用できます。このボタンをクリックすると，記録中にテキスト・チェックがスクリプトに挿入されます。テキスト・チェックの詳細については，906 ページの「テキスト・チェックと画像チェックを追加する方法」を参照してください。
	**[ビットマップ同期を挿入]**：このオプションは，Citrix および RDP のスクリプトを記録する場合にフローティング・ツールバーでのみ使用できます。これにより，指定したスナップショットの領域に基づいて同期化ステップを挿入できます。
	**[テキスト同期を挿入]**：このオプションは，Citrix および RDP のスクリプトを記録する場合にフローティング・ツールバーでのみ使用できます。指定したテキストの選択部分に基づいて同期化ステップが挿入されます。
	**[画像による同期を挿入]**：このオプションは，RDP のスクリプトを記録する場合にフローティング・ツールバーでのみ使用できます。指定した画像に基づいて同期化ステップが挿入されます。
[標準] ツールバー	
	**[新規作成]**：仮想ユーザ・スクリプトを新規作成します。
	**[開く]**：既存の仮想ユーザ・スクリプトが開きます。
	**[保存]**：現在のスクリプトが保存されます。
	**[プロトコル アドバイザ]**：[プロトコル アドバイザ] が開きます。詳細については，101 ページの「プロトコル・アドバイザ」を参照してください。
[ツリービュー] ツールバー	

UI 要素	説明
	**［プロパティ］**：
	**後に挿入。**
	**前に挿入。**
	**ステップの削除。**
［ツール］ツールバー	
	**［Controller のシナリオを作成］**：VuGen アプリケーションで Controller の基本的なシナリオを作成します。詳細については，155 ページの「VuGen から Controller のシナリオを作成する方法」を参照してください。
	**［HP ALM］**：HP Application Lifecycle Management への接続が開き，ALM で保存したスクリプトを保存および操作できるようになります。詳細については，251 ページの「Application Lifecycle Management を使った作業」を参照してください。
［表示］ツールバー	
	**［タスクの表示］**：［タスク］表示枠が表示されます。詳細については，前述の［タスク］表示枠の説明を参照してください。
	**［スクリプトを表示］**：スクリプト・ビューに切り替わります。スクリプト・ビューの UI 要素の詳細については，前述の説明を参照してください。
	**［ツリーを表示］**：ツリー・ビューに切り替わります。ツリー・ビューの UI 要素の詳細については，前述の説明を参照してください。
	**［出力ウィンドウの表示 / 非表示］**：出力ウィンドウが表示されます。詳細については，前述の出力ウィンドウの説明を参照してください。
［仮想ユーザ］ツールバー	

UI 要素	説明
	**[パラメータ リストを開く]**：[パラメータ リスト] ダイアログ・ボックスが開きます。詳細については，308 ページの「[パラメータ リスト] ダイアログ・ボックス」を参照してください。
	**[実行環境設定の編集]**：[実行環境設定] ダイアログ・ボックスが開きます。詳細については，419 ページの「実行環境の設定」を参照してください。

出力ウィンドウ

このページのダイアログ・ボックスには，記録フェーズまたは再生フェーズで収集された情報が含まれているログが表示されます。

利用方法	[表示] > [出力ウィンドウ]

このウィンドウには，次のタブがあります。

- 88 ページの「[出力ウィンドウ] - [相関結果] タブ」
- 90 ページの「[出力ウィンドウ] - [生成ログ] タブ」
- 92 ページの「[出力ウィンドウ] - [パラメータ] タブ（Winsock のみ）」
- 94 ページの「[出力ウィンドウ] - [記録ログ] タブ」
- 95 ページの「[出力ウィンドウ] - [再生ログ] タブ」
- 96 ページの「[出力ウィンドウ] - [実行時データ] タブ」

［出力ウィンドウ］-［相関結果］タブ

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。また，相関を作成および管理することもできます。

利用方法	**［表示］>［出力ウィンドウ］**
関連タスク	136 ページの「仮想ユーザ・スクリプトを再生する方法」

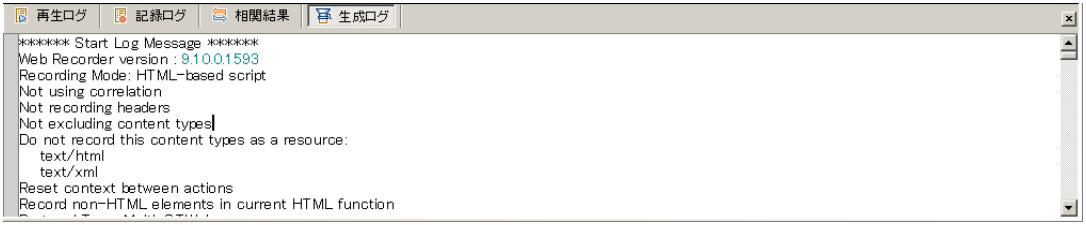
ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
相関	選択した文字列で相関が作成されます。これにより，**web_reg_save_param_*** 関数が追加され，元の値がコメントとしてスクリプトに保存されます。Web ステップ内の元の値の適切な出現箇所がパラメータで置き換えられます。
再生	スクリプトが再生されます。

UI 要素	説明
< 差異のリスト >	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。 ➤ **[相関済み]**：項目が相関済みかどうかが示されます。相関済みの項目にはチェックが付きます。 ➤ **[記録時のスナップショットのテキスト]**：記録フェーズの相関ストリング。 ➤ **[再生時のスナップショットのテキスト]**：再生フェーズの相関ストリング。 ➤ **[次で最初に発生]**：相関が最初に検出されたスクリプトのセクション。
相関オプション	［一般オプション］ダイアログ・ボックスの［相関］タブにあるいくつかの詳細相関オプションを設定できます。ユーザ・インタフェースの詳細については，73 ページの「[一般オプション］ダイアログ・ボックス」を参照してください。
次の差異を表示	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。

［出力ウィンドウ］-［生成ログ］タブ

レコーダのバージョンや記録オプションの値など，コード生成に使用されたスクリプトの設定の概要が表示されます。



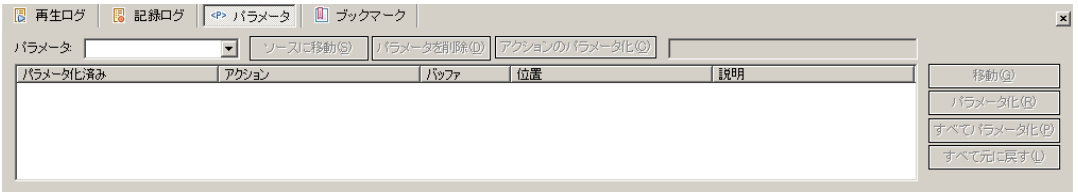
利用方法	**［表示］>［出力ウィンドウ］**
関連タスク	136 ページの「仮想ユーザ・スクリプトを再生する方法」

［出力ウィンドウ］-［対話式再生ログ］タブ

TruClient スクリプトの場合にのみ，実行時の仮想ユーザのアクションを表すメッセージと発生したエラーが表示されます。

利用方法	**［表示］>［出力ウィンドウ］**
関連タスク	516 ページの「Ajax TruClient スクリプトのデバッグ方法」

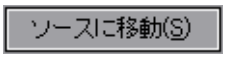
［出力ウィンドウ］-［パラメータ］タブ（Winsock のみ）



利用方法	**［表示］>［出力ウィンドウ］**
関連タスク	136 ページの「仮想ユーザ・スクリプトを再生する方法」

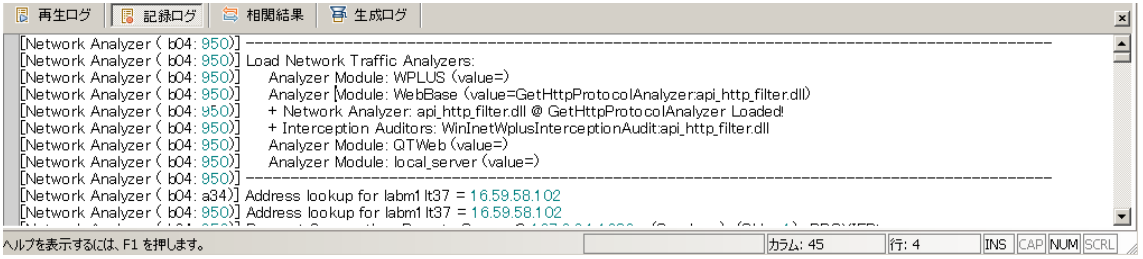
ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
パラメータ	ドロップダウン・メニューから目的のパラメータを選択します。
<パラメータ・インスタンス・リスト>	選択したパラメータのインスタンスと，各インスタンスの次の詳細が表示されます。 ➤ **［パラメータ化済み］**：該当のインスタンスがパラメータ化済みかどうかが示されます。値を変更するには，インスタンスを右クリックして［**置換を元に戻す**］または［**パラメータで置換**］を選択します。 ➤ **［アクション］**：インスタンスが存在するスクリプトのセクション。 ➤ **［バッファ］**：該当のインスタンスが存在するバッファ。 ➤ **［位置］**：バッファ内のインスタンスの場所。 ➤ **［説明］**：インスタンスのステータスの説明。この説明が変更されるのは，［パラメータ化］カラムの値が変更されたときです。

UI 要素	説明
	選択したパラメータがスクリプトから削除されます。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。
	選択したパラメータのインスタンスに移動します。
	パラメータのソース・データに移動します。 注：これは，パラメータ・インスタンス・リストの［アクション］カラムに値がないと機能しません。
	複数のセクション（init，action，および end など）のスクリプトを記録した場合にパラメータを作成すると，現在選択されているスクリプトのセクションでのみインスタンスが検索されます。ほかのセクションのパラメータのインスタンスを検索するには，セクションを選択して［**アクションのパラメータ化**］ボタンをクリックします。
	選択したパラメータのインスタンスがすべてパラメータ化されます。
	選択したパラメータの置換が元に戻ります。
	選択したパラメータの置換がすべて元に戻ります。

［出力ウィンドウ］-［記録ログ］タブ

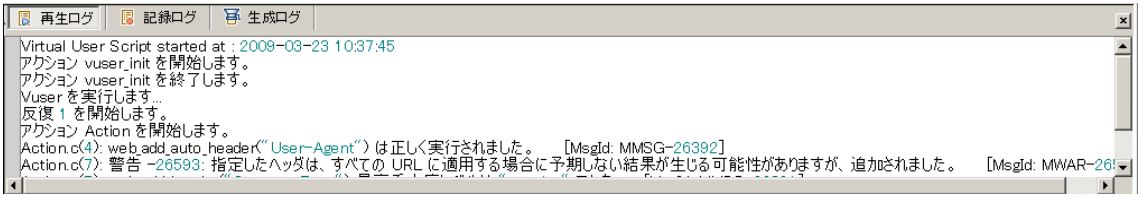
記録中に発行されたメッセージが表示されます。使用しているプロトコルによっては，［記録オプション］で該当のログの詳細レベルを設定できます。



利用方法	**［表示］>［出力ウィンドウ］**
関連タスク	136 ページの「仮想ユーザ・スクリプトを再生する方法」

［出力ウィンドウ］-［再生ログ］タブ

実行時の仮想ユーザのアクションを表すメッセージと発生したエラーが表示されます。



利用方法	**［表示］>［出力ウィンドウ］**
関連タスク	136 ページの「仮想ユーザ・スクリプトを再生する方法」

［再生ログ］では，各種メッセージを識別しやすいようにテキストがさまざまに色分けされています。

アクション名で始まる行をダブルクリックすると，スクリプト内の対応するステップにカーソルが移動します。

［**実行環境の設定**］>［**一般**］>［**ログ**］ノードで，再生ログに送信される情報量（詳細レベル）を設定できます。詳細については，442 ページの「［一般］>［ログ］ノード」を参照してください。

［出力ウィンドウ］-［実行時データ］タブ

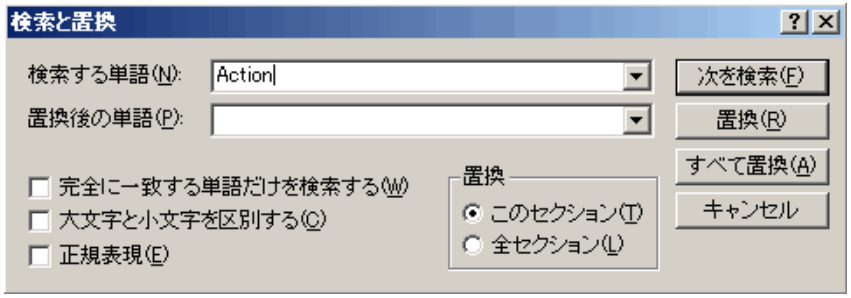
［実行時データ］タブは，再生中にのみアクセスできます。

利用方法	**［表示］>［出力ウィンドウ］**
関連タスク	136 ページの「仮想ユーザ・スクリプトを再生する方法」

- **［一般］**：現在の反復数，現在再生されているステップのアクション名，スクリプト内での行番号（スクリプト・ビュー）が表示されます。
- **［パラメータ］**：スクリプトに定義されているすべてのパラメータと，選択した更新方法（順次，一意など）に基づくそれらのパラメータの置換値が表示されます。これらの情報はパラメータがスクリプトの中で使用されていない場合にも表示されます。パラメータの詳細については，263 ページの「パラメータ」を参照してください。

［検索と置換］ダイアログ・ボックス

このダイアログ・ボックスを使用して，VuGen コード内の文字列の検索と置き換えができます。



利用方法	**［編集］>［置換］**

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
検索する文字列	検索する単語を指定します。
置換後の単語	［検索する文字列］ボックスで指定したテキストと置き換えるテキストを入力します。
完全に一致する単語だけを検索する	別の単語の一部として出現する箇所を除外し，単語として完全に一致する箇所のみを検索します。
大文字と小文字を区別する	検索の際に大文字と小文字の違いを区別します。
正規表現	検索文字列が正規表現であることを示します。
置換	➤ **［このセクション］**：選択したセクション内の出現箇所が置換されます。 ➤ **［全セクション］**：スクリプト全体の出現箇所が置換されます。
次を検索(F)	［検索する文字列］ボックスのテキストが次に出現する箇所を検索します。

UI 要素	説明
置換(R)	［検索する文字列］ボックスのテキストが検索され，強調表示された出現箇所が［置換後の単語］ボックスのテキストで置き換えられます。
すべて置換(A)	［検索する文字列］ボックスのテキストが検索され，上記の［このセクション］または［全セクション］の選択に合わせて［置換後の単語］ボックスのテキストで置き換えられます。

［スナップショット］表示枠

このセクションには，選択したスクリプトのステップに関するデータが表示されます。

利用方法	［ツリー ビュー］の右側にある表示枠
重要情報	この表示枠は，選択したステップの種類に応じて異なります。

Web（HTTP/HTML）プロトコル

UI 要素	説明
HTML ビュー	HTML 形式でステップ・データが表示されます。
HTTP ビュー	HTTP 形式でステップ・データが表示されます。これにより，ユーザはステップに関する詳細な情報（要求データ，応答データ，Cookie，およびヘッダなど）を表示できます。詳細については，904 ページの「Web スナップショット」を参照してください。
グリッド	（HTTP ビューのみ）リスト形式で HTTP フロー・データが表示されます。
ツリー	（HTTP ビューのみ）ツリー構造で HTTP フロー・データが表示されます。
	記録フェーズのスナップショット・データが表示されます。

UI 要素	説明
	記録フェーズおよび再生フェーズの両方のスナップショット・データが表示されます。
	再生フェーズのスナップショット・データが表示されます。

Flex および AMF プロトコル

［スナップショット］タブには，さまざまな形式でステップ・データが表示されます。記録フェーズまたは再生フェーズのステップ・データを表示できます。要求データまたは応答データを表示できます。ツリー・ビューまたは xml 形式でデータを表示できます。

トラブルシューティングと制限事項

このセクションでは，VuGen のメイン・ユーザ・インタフェースのトラブルシューティングと制限事項について説明します。

スナップショットのトラブルシューティング

スナップショットのないステップが見つかった場合は，次のガイドラインに従ってスナップショットが生成されない原因を特定します。すべてのステップがスナップショットに関連付けられているわけではありません。スナップショットがあるのは，画面操作があるステップかブラウザ・ウィンドウの内容を表示しているステップ（Web の場合）のみです。

いくつかのプロトコルでは，記録オプションにより，記録中にスナップショットのキャプチャを無効にできます。

選択したステップの**記録**時のスナップショットがない場合は，次のような原因が考えられます。

- スクリプトが VuGen 6.02 以前のバージョンで記録された。
- スナップショットが特定の種類のステップについて生成されていない。
- インポートされたアクションに，スナップショットが含まれていない。

選択したステップの**再生**時のスナップショットがない場合は，次のような原因が考えられます。

- スクリプトが VuGen 6.02 以前のバージョンで記録された。

- インポートされたアクションに，スナップショットが含まれていない。
- 仮想ユーザ・ファイルが読み取り専用のディレクトリに格納されているため，VuGen が再生時のスナップショットを保存できなかった。
- ステップがリソースへのナビゲーションを表している。

2

プロトコル・アドバイザ

本章の内容

概念

タスク

リファレンス

概念

プロトコル・アドバイザの概要

プロトコル・アドバイザは，仮想ユーザ・スクリプトを記録するために適切なプロトコルを決定するのに役立ちます。プロトコル・アドバイザは，アプリケーションで異なるプロトコルの要素をスキャンし，検出したプロトコルのリストを表示します。これらのプロトコルは，アプリケーションに適したプロトコルを検出するガイドラインおよび起点として使用します。

ほとんどの場合，プロトコルの組み合わせとともに，複数のプロトコルが提案されて表示されます。次の項では，提案されたプロトコルのリストを使用する方法に関するガイドラインを提供します。

タスク

プロトコル・アドバイザの使用方法

このタスクでは，プロトコル・アドバイザを使用して，アプリケーションを記録するための最適なプロトコルを検出するための一般的なワークフローについて説明します。

このタスクでは，次の手順を実行します。

- 103 ページの「プロトコル・アドバイザを開始する」
- 103 ページの「一般的なビジネス・プロセスを実行する」
- 103 ページの「結果を保存する」
- 104 ページの「プロトコルを選択し，新しい仮想ユーザ・スクリプトを作成する」
- 104 ページの「拡張，デバッグ，再生の検証を行う」
- 105 ページの「再生に成功しない場合は，別のプロトコルを選択して繰り返します」

1 プロトコル・アドバイザを開始する

開始ページで［**ファイル**］>［**プロトコル アドバイザ**］>［**アナライザ アプリケーション**］を選択し，ダイアログ・ボックスを設定します。

2 一般的なビジネス・プロセスを実行する

アプリケーションで一般的なビジネス・プロセスを実行します。包括的な結果が得られるように，さまざまなビジネス・プロセスを実行してください。［**分析を停止**］をクリックして分析を終了し，結果を表示します。

3 結果を保存する

［**ファイル**］>［**プロトコル アドバイザ**］>［**結果の保存**］を選択します。名前を入力してディレクトリを選択します。

4 プロトコルを選択し，新しい仮想ユーザ・スクリプトを作成する

次の優先順位に従っていずれかのプロトコルを選択し，そのプロトコルを使用して仮想ユーザ・スクリプトを作成します。

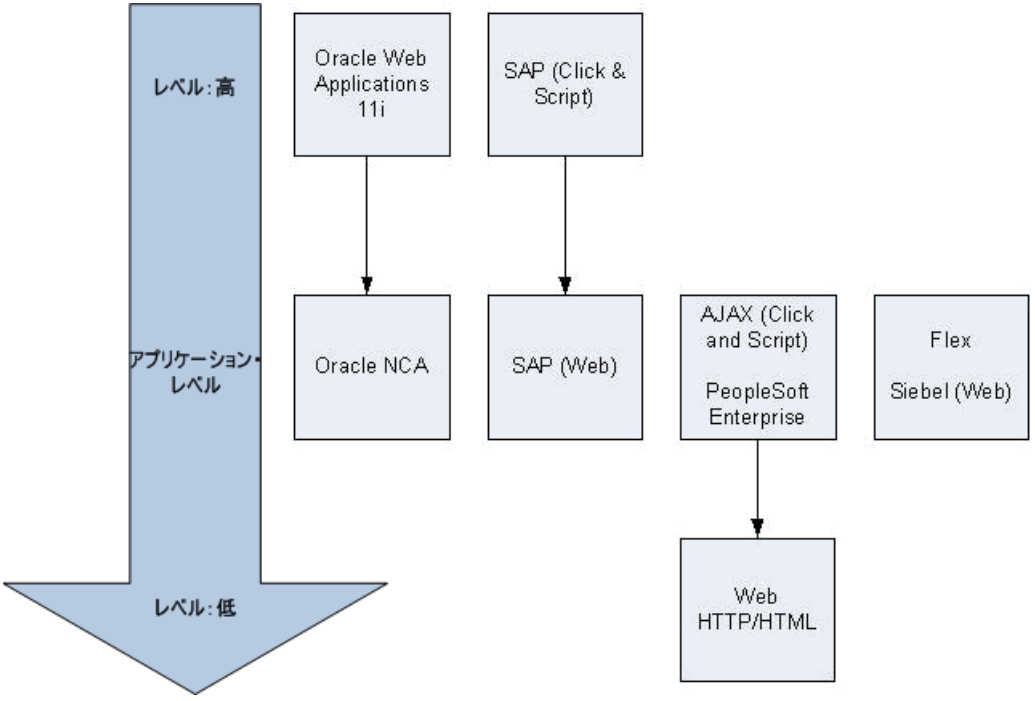
- 複数のプロトコル / プロトコルの組み合わせ
- 最上位レベルのアプリケーション・プロトコル（手順 105 ページの 6 の図を参照）。
- リストで一番上に表示されているプロトコル

5 拡張，デバッグ，再生の検証を行う

再生に成功するまでスクリプトを拡張してデバッグします。仮想ユーザ・スクリプトを拡張およびデバッグした後に，再生に成功しない場合は，次の手順に進みます。

6 再生に成功しない場合は，別のプロトコルを選択して繰り返します

- **Web ベースでないすべてのプロトコル**：[プロトコル アドバイザ結果] ページに戻り，リストで次のプロトコルを選択して，前の手順を繰り返します。ほかにプロトコルがない場合は，HP のカスタマ・サポートに連絡してください。
- **Web ベースのプロトコル**：次の図は，アプリケーション・レベルに基づいて配置された Web ベースのプロトコルの例を示しています。矢印は，新しいプロトコルを試す順序を示しています。たとえば，最初に使用したプロトコルが Oracle Web アプリケーション 11i だった場合，このプロトコルの再生に失敗したら，次に Oracle NCA プロトコルを試します。



リファレンス

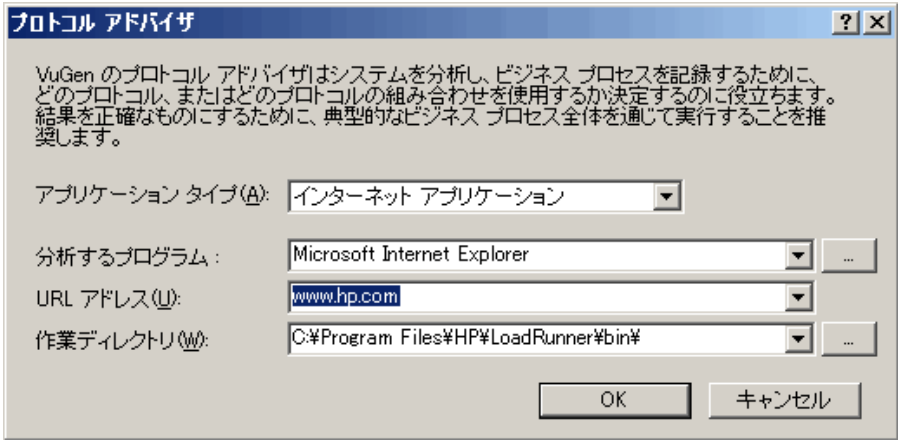
プロトコル・アドバイザのユーザ・インタフェース

このセクションの内容

➤［プロトコル アドバイザ］ダイアログ・ボックス（106 ページ）

［プロトコル アドバイザ］ダイアログ・ボックス

このダイアログ・ボックスでは，プロトコル・アドバイザを使用してアプリケーションを開き，記録に使用する適切な VuGen プロトコルを決定できます。



利用方法	**［ファイル］>［プロトコル アドバイザ］>［アナライザ アプリケーション］**
重要情報	ダイアログ・ボックス内のオプションは，［**アプリケーション タイプ**］フィールドでの選択によって異なります。
関連タスク	103 ページの「プロトコル・アドバイザの使用方法」

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
アプリケーション タイプ	アプリケーションのタイプ：Win32 アプリケーションまたはインターネット・アプリケーション。
プログラム引数（Win32 アプリケーションのみ）	前述で指定した実行ファイルのコマンド・ライン引数を指定します。たとえば，*plus32.exe* にコマンド・ライン・オプション peter@neptune を指定した場合，*plus32.exe* を起動すると，ユーザ *Peter* がサーバ *Neptune* に接続されます。このオプションが表示されるのは，Win32 アプリケーションの場合のみです。
分析するプログラム	分析するブラウザ，インターネット・アプリケーション，または Win32 アプリケーションを選択します。
URL アドレス（インターネット・アプリケーションのみ）	開始する URL アドレス。このオプションが表示されるのは，インターネット・アプリケーションの場合のみです。
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。

トラブルシューティングと制限事項

このセクションでは，プロトコル・アドバイザのトラブルシューティングと制限事項について説明します。

- この機能は，LoadRunner でサポートされているプロトコルのみを検出します。
- 一部の Web プロトコルは URL に基づいて検出されます。たとえば，URL に SAP のようなキーワードが含まれている場合があります。このため，異なる URL または同じ URL を使う異なるアプリケーションを使用する場合は，結果が異なる場合があります。
- 次のプロトコルは頻繁に検出されますが，必ずしも使用に適していません。ほかにプロトコルが検出されなかった場合以外は使用しないでください。
  - COM/DCOM
  - Java
  - .NET
  - WinSocket
  - LDAP

3

記録

本章の内容

概念

タスク

リファレンス

概念

認証情報の提供

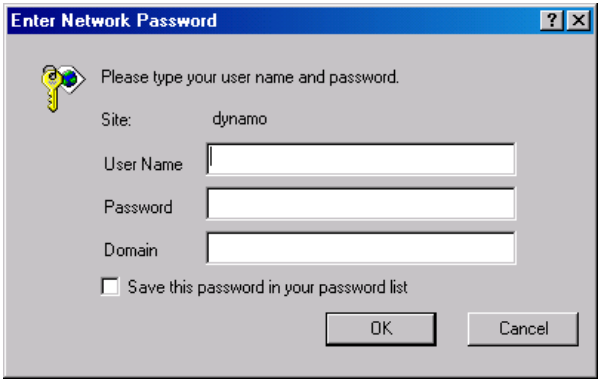
次のセクションの内容は，マルチ・プロトコル・スクリプトを対象としています。

NTLM 認証を使用する Web セッションを記録する際，サーバでユーザ名やパスワードなどの詳細情報を入力しなければならない場合があります。

最初に，IE（Internet Explorer）は，現在のユーザの NT 認証情報の使用を試みます。

➤ IE からこの情報を使用してログインに成功し，スクリプトを記録した場合は，記録の最後にパスワードを入力するよう VuGen から求められます。ユーザ名とドメイン情報は VuGen が自動的に取得します。必要ならば，［Web レコーダ NTLM 認証］ダイアログ・ボックスでユーザ名を編集することもできます。

➤ IE から現在のユーザの情報を使用してログインできない場合は，標準のブラウザ認証ダイアログ・ボックスを使用してユーザ名とパスワードを入力するよう IE から求められます。



web_set_user 関数の生成

NTLM 認証を実行すると，VuGen によって **web_set_user** 関数がスクリプトに追加されます。

➤ 認証が成功した場合は，ユーザ名，暗号化されたパスワード，およびホストが設定された **web_set_user** 関数が，VuGen によって生成されます。

```
web_set_user("domain1¥¥dashwood",
        lr_decrypt("4042e3e7c8bbbcfde0f737f91f"),
       "sussex:8080");
```

➤ ［Web レコーダ NTLM 認証］ダイアログ・ボックスで情報を入力せずにキャンセルした場合も，手作業で編集できるように，VuGen によって **web_set_user** 関数が生成されます。

```
web_set_user("domain1¥¥dashwood,
        "NTLM パスワードをここに入力 ",
        "sussex:8080");
```

注：パスワードを手作業で入力した場合は，スクリプト内にそのまま出現するため，セキュリティ上問題があります。パスワードを暗号化するには，パスワードを右クリックして［**文字列を暗号化**］を選択します。VuGen によって文字列が暗号化され，再生時にパスワードの復号に使用する **lr_decrypt** 関数が生成されます。文字列の暗号化の詳細については，150 ページの「テキストの暗号化」を参照してください。

スクリプト・ディレクトリ・ファイル

記録中，VuGen は一連の設定ファイル，データ・ファイル，ソースコード・ファイルを作成します。これらのファイルには，仮想ユーザ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。スクリプト・フォルダを開くには，スクリプト・ビュー（［**表示**］>［**スクリプト ビュー**］）に切り替えて，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。

仮想ユーザ・スクリプト・テンプレート

ユーザ定義テンプレートにより，特有の構成のスクリプトをテンプレートとして保存できます。作成した後は，スクリプトを作成する際の原形としてこのテンプレートを使用できます。

テンプレートでは，次のファイルとデータがサポートされています。

- 実行環境の設定
- パラメータ
- その他のファイル
- アクション
- スナップショット

［記録］および［一般］オプションは一般的な設定であり，特定のスクリプトに関連していないため，サポートされていません。

注意と制限事項

- 特定のプロトコルに対するスクリプトを設定し，そのスクリプトをテンプレートとして保存すると，そのテンプレートに基づくほかのスクリプトは，そのプロトコルでのみ機能します。たとえば，Web（Click and Script）プロトコルに対するスクリプトを設定し，そのスクリプトからテンプレートを作成した場合，そのテンプレートは Web（Click and Script）でしか使用できません。
- テンプレートを作成した後は，そのテンプレートを VuGen で直接編集することはできません。変更を加えるには，テンプレートを開いて別の名前を付けて再び保存するか，既存のテンプレートを上書きします。
- テンプレートから元のスクリプトを再生成すると，手作業で加えた変更はすべて失われます。

スクリプトのセクション

各仮想ユーザ・スクリプトには，少なくとも vuser_init，Actions（1 つ以上），vuser_end の 3 つのセクションが含まれます。記録前または記録中に，VuGen によって記録される関数の挿入先となるスクリプトのセクションを選択できます。次の表に，各セクションに何が記録され，各セクションがどのタイミングで呼び出されるかを示します。

スクリプトのセクション	何を記録するときに使われるか	実行のタイミング
vuser_init	サーバへのログイン	仮想ユーザを初期化（ロード）するとき
アクション	クライアントの動作	仮想ユーザが「**実行**」状態のとき
vuser_end	ログオフの手順	仮想ユーザを終了または停止するとき

仮想ユーザ・スクリプトの反復を複数回実行するときは，スクリプトの *Actions* セクションだけが繰り返されます。*vuser_init* セクションと *vuser_end* セクションは繰り返されません。反復の設定の詳細については，452 ページの「[一般] > [実行論理] ノード」を参照してください。

各スクリプト・セクションの内容の表示と編集には，VuGen スクリプト・エディタを使用します。一度に表示できるのは，1 つのセクションの内容だけです。セクションを表示するには，左側の表示枠でセクションの名前を選択して強調表示します。

Java クラスを使用する仮想ユーザ・スクリプトの場合には，すべてのコードを Actions クラスに置きます。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。これらのメソッドは，ほかのプロトコルの場合に作成されるスクリプトの各セクションに対応します。つまり，初期化ルーチンは init メソッドに，クライアントの動作は action メソッドに，ログオフの手順は end メソッドに，それぞれ挿入します。

詳細については，713 ページの「Java プロトコル - 手作業によるスクリプトのプログラミング」を参照してください。

```
public class Actions {
    public int init() {
        return 0;}
    public int action() {
        return 0;}
    public int end() {
        return 0;}
}
```

注：Oracle DB のトランザクション・ブレークダウンは，vuser_init セクションで記録されたアクションには使用できません。

タスク

仮想ユーザ・スクリプトを作成または表示する方法

このタスクでは，新しい仮想ユーザ・スクリプトを作成する方法や，既存の仮想ユーザ・スクリプトを開く方法について説明します。

注：スクリプトに *init*，*run*，*end* という名前は付けないでください。これらの名前は VuGen によって使用されます。

このタスクでは，次の手順を実行します。

- 116 ページの「プロトコル・アドバイザを使用したスクリプト・タイプの選択」
- 116 ページの「新しいシングル・プロトコル・スクリプトの作成」
- 116 ページの「新しいマルチ・プロトコル・スクリプトの作成」
- 116 ページの「テンプレートからのスクリプトの作成または表示」
- 116 ページの「標準のスクリプトの表示」
- 116 ページの「.zip スクリプトの表示または操作」
- 116 ページの「Application Lifecycle Management に保存されたスクリプトの表示」

プロトコル・アドバイザを使用したスクリプト・タイプの選択

プロトコル・アドバイザを使用して，アプリケーションに基づいて関連するタイプの仮想ユーザ・スクリプトのリストを生成できます。詳細については，101 ページの「プロトコル・アドバイザ」を参照してください。

新しいシングル・プロトコル・スクリプトの作成

［**ファイル**］>［**新規作成**］>［**新規シングルプロトコル スクリプト**］を選択し，プロトコルのタイプを選択します。ユーザ・インタフェースの詳細については，125 ページの「［新規仮想ユーザ］ダイアログ・ボックス」を参照してください。

新しいマルチ・プロトコル・スクリプトの作成

［**ファイル**］>［**新規作成**］>［**新規マルチプロトコル スクリプト**］を選択し，［**利用可能なプロトコル**］リストから［**選択されたプロトコル**］リストに目的のプロトコルを移動します。ユーザ・インタフェースの詳細については，125 ページの「［新規仮想ユーザ］ダイアログ・ボックス」を参照してください。

テンプレートからのスクリプトの作成または表示

タスクの詳細については，121 ページの「仮想ユーザ・スクリプト・テンプレートを作成および表示する方法」を参照してください。

標準のスクリプトの表示

ローカル・マシンまたはネットワーク・ドライブに格納されているスクリプトを開くには，［**ファイル**］>［**開く**］を選択します。

.zip スクリプトの表示または操作

.zip 形式のスクリプトを解凍または操作できます。タスクの詳細については，117 ページの「.zip ファイルでの作業の方法」を参照してください。

Application Lifecycle Management に保存されたスクリプトの表示

スクリプトを HP ALM に保存して VuGen で変更できます。詳細については，251 ページの「Application Lifecycle Management を使った作業」を参照してください。

.zip ファイルでの作業の方法

VuGen では，.zip ファイルでの作業をいくつかの方法で実行できます。.zip ファイルでの作業には，ディスク容量が節約され，スクリプトが移動しやすくなる，などの利点があります。多数のファイルをマシン間でコピーせずに，1 つの .zip ファイルをコピーするだけで済みます。

このタスクでは，次の手順を実行します。

- 117 ページの「.zip ファイルからのインポート」
- 117 ページの「.zip ファイルでの作業」
- 117 ページの「.zip 形式でのスクリプトの保存」
- 118 ページの「スクリプトの圧縮および電子メール送信」

.zip ファイルからのインポート

.zip ファイルに格納されているスクリプトを開くには，［**ファイル**］>［**ZIP 操作**］>［**Zip ファイルからインポート**］を選択します。.zip ファイルを選択すると，圧縮解除されたファイルを格納する場所を指定するよう求められます。

.zip ファイルでの作業

.zip ファイルから作業を行い，その間スクリプト・ファイルの展開や保存をしないようにするには，［**ファイル**］>［**ZIP 操作**］>［**Zip ファイルで作業**］を選択します。スクリプトを変更して保存すると，変更内容が .zip ファイルに直接格納されます。

.zip 形式でのスクリプトの保存

スクリプトのディレクトリ全体を .zip ファイルとして保存するには，［**ファイル**］>［**ZIP 操作**］>［**Zip ファイルにエクスポート**］を選択します。

すべてのファイルを保存するか，実行可能ファイルだけを保存するかを指定できます。また，圧縮率を指定することもできます。圧縮率が高いほど，VuGen によるアーカイブ作成に時間がかかります（［**最高**］の圧縮オプションが最も低速）。

スクリプトの圧縮および電子メール送信

.zip ファイルを作成して電子メールの添付ファイルとして送信するには，［**ファイル**］>［**ZIP 操作**］>［**Zip して電子メールで送信**］を選択します。［**ファイルの圧縮**］ダイアログ・ボックスで［**OK**］をクリックすると，設定に従ってファイルが圧縮され，.zip ファイルを添付ファイルとして持つ電子メール作成フォームが開きます。

スクリプトからコードをインポートする方法

複数のアクションをサポートする仮想ユーザ・タイプの場合，別の仮想ユーザ・スクリプトから現在のスクリプトにアクションをインポートできます。インポートできるのは，同じタイプの仮想ユーザのアクションだけです。インポートされたアクションに関連付けられているパラメータは，スクリプトに組み込まれます。

現在のスクリプトへアクションをインポートするには，次の手順で行います。

1 ［**アクション**］>［**仮想ユーザにアクションをインポート**］を選択するか，［タスク］表示枠を右クリックして［**仮想ユーザにアクションをインポート**］を右クリック・メニューから選択します。［アクションのインポート］ダイアログ・ボックスが表示されます。

2 ［**参照**］をクリックして仮想ユーザ・スクリプトを選択します。［**インポートするアクション**］セクションに，スクリプトのアクションのリストが表示されます。

3 アクションを強調表示して［**OK**］をクリックします。スクリプトにアクションが表示されます。

仮想ユーザ・スクリプトの記録方法

このタスクでは，仮想ユーザ・スクリプトの記録方法について説明します。

このタスクでは，次の手順を実行します。

➤ 119 ページの「記録オプションの設定（任意）」

➤ 119 ページの「記録セッションの初期化」

➤ 119 ページの「アプリケーションのビジネス・プロセスの実行」

1 記録オプションの設定（任意）

記録オプションには，スクリプトを作成する記録ステージおよび生成ステージでスクリプトに影響するさまざまなオプションがあります。概念とユーザ・インタフェースの詳細については，313 ページの「記録オプション」を参照してください。

2 記録セッションの初期化

新しいスクリプトの作成時に自動的に発生します。手作業で記録を開始するには，ツールバーの［**記録開始**］ボタンをクリックし，［記録開始］ダイアログ・ボックスを設定します。ユーザ・インタフェースの詳細については，126 ページの「［記録開始］ダイアログ・ボックス」を参照してください。

3 アプリケーションのビジネス・プロセスの実行

VuGen によってアプリケーションやフローティング・ツールバーが自動的に開き，アクションの記録が開始されます。記録するビジネス・プロセスを実行します。ツールバーでは，トランザクション，ランデブー・ポイント，コメントを挿入できます。また，アプリケーションの記録時に記録するスクリプトのセクションを決定できます。ユーザ・インタフェースの詳細については，79 ページの「VuGen のメイン・ユーザ・インタフェース」を参照してください。

完了したら，ツールバーの［**停止**］ボタンをクリックします。

仮想ユーザ・スクリプトの再生成方法

最初に記録したスクリプトに戻す必要がある場合は，スクリプトを再生成します。この機能は，デバッグや，破損したスクリプトの修復に非常に役立ちます。スクリプトを再生成すると，記録されたアクションに手作業で追加した拡張機能はすべて削除されます。スクリプトにパラメータを追加した場合は，VuGen によって元の値に戻されます。ただし，パラメータ・リストは削除されないため，それまでに作成したパラメータは再挿入できます。再生成で復元されるのは記録されたアクションだけです。手作業で追加されたアクションは復元されません。

このタスクでは，仮想ユーザ・スクリプトの再生成方法について説明します。

このタスクでは，次の手順を実行します。

➤ 120 ページの「再生成の初期化」

➤ 120 ページの「[オプションの再生成］の変更（任意）」

1 再生成の初期化

［**ツール**］>［**スクリプトの再生成**］を選択します。手作業で行ったすべての変更が上書きされることを示す警告が表示されます。

2［オプションの再生成］の変更（任意）

［**オプション**］をクリックして［**オプションの再生成**］ダイアログ・ボックスを開きます。

マルチ・プロトコル・スクリプトの場合，［**一般**］>［**プロトコル**］ノードで，スクリプトの再生成時に記録するプロトコルを変更できます。ユーザ・インタフェースの詳細については，353 ページの「[一般］の［プロトコル］ノード」を参照してください。

スクリプト・オプションを変更するには，［**一般**］>［**スクリプト**］ノードを選択し，適切なチェック・ボックスを選択またはクリアします。ユーザ・インタフェースの詳細については，358 ページの「[一般］の［スクリプト］ノード」を参照してください。

仮想ユーザ・スクリプト・テンプレートを作成および表示する方法

このタスクでは，仮想ユーザ・スクリプト・テンプレートを作成および表示する方法について説明します。

このタスクでは，次の手順を実行します。

- 121 ページの「仮想ユーザ・スクリプト・テンプレートの作成」
- 121 ページの「仮想ユーザ・スクリプト・テンプレートの表示」

仮想ユーザ・スクリプト・テンプレートの作成

VuGen でスクリプトを開き，［**ファイル**］>［**ユーザ定義テンプレート**］>［**テンプレートとして保存**］を選択し，テンプレートの名前を入力します。

仮想ユーザ・スクリプト・テンプレートの表示

［**ファイル**］>［**ユーザ定義テンプレート**］>［**テンプレートからスクリプトを作成**］を選択し，テンプレート・ファイルを選択します。

注意と制限事項

- 特定のプロトコルに対するスクリプトを設定し，そのスクリプトをテンプレートとして保存すると，そのテンプレートに基づくほかのスクリプトは，そのプロトコルでのみ機能します。たとえば，Web（Click and Script）プロトコルに対するスクリプトを設定し，そのスクリプトからテンプレートを作成した場合，そのテンプレートは Web（Click and Script）でしか使用できません。
- テンプレートを作成した後は，そのテンプレートを VuGen で直接編集することはできません。変更を加えるには，テンプレートを開いて別の名前を付けて再び保存するか，既存のテンプレートを上書きします。
- テンプレートから元のスクリプトを再生成すると，手作業で加えた変更はすべて失われます。

リファレンス

記録中に生成されるファイル

記録されたテストに **vuser** という名前を付け，それを **c:¥tmp** の下に格納したとします。記録後に生成される，特に重要なファイルの一覧を次に示します。

ファイル名	詳細
vuser.usr	仮想ユーザに関する情報（タイプ，テスト対象アプリケーション，アクション・ファイルなど）が含まれます。
vuser.bak	最後に保存した Vuser.usr の 1 つ前のコピー。
default.cfg	VuGen アプリケーションで定義されたすべての実行環境の設定（思考遅延時間，反復，ログ，Web）の一覧が含まれます。
vuser.asc	記録されている API 呼び出し。
vuser.grd	データベース・スクリプトのグリッドのカラム・ヘッダが含まれています。
default.usp	スクリプトの実行ロジック（Actions セクションの実行方法など）が含まれます。
init.c	VuGen メイン・ウィンドウに表示される Vuser_init 関数とまったく同じもの。
run.c	VuGen メイン・ウィンドウに表示される Action 関数とまったく同じもの。
end.c	VuGen メイン・ウィンドウに表示される Vuser_end 関数とまったく同じもの。

ファイル名	詳細
vdf.h	スクリプトで使用される C 変数定義のヘッダ・ファイル。
¥Data	Data ディレクトリには，主にバックアップ用として使用されるすべての記録データが格納されます。データはこのディレクトリに格納されると，編集したり使用したりできなくなります。たとえば，**Vuser.c** は，**run.c** のコピーです。

Vuser.usr ファイルの例

```
[General]
Type=Oracle_NCA
DefaultCfg=default.cfg
AppName=C:¥PROGRA~1¥Netscape¥COMMUN~1¥Program¥netscape.exe
BuildTarget=
ParamRightBrace=>
ParamLeftBrace=<
NewFunctionHeader=0
MajorVersion=5
MinorVersion=0
ParameterFile=nca_test3.prm
GlobalParameterFile=
[Transactions]
Connect=
[Actions]
vuser_init=init.c
Actions=run.c
vuser_end=end.c
```

default.cfg ファイルの例

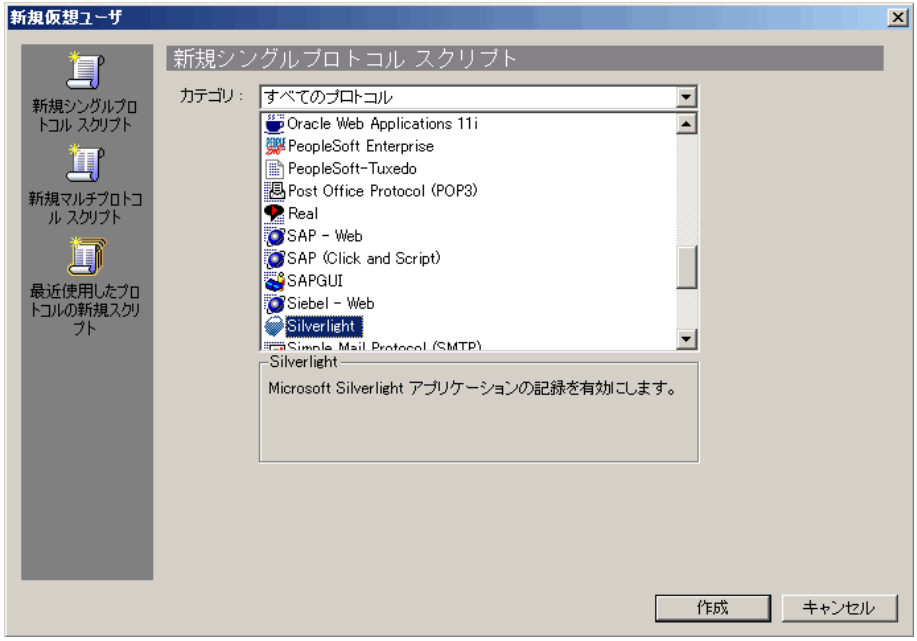
```
[General]
XlBridgeTimeout=120

[ThinkTime]
Options=NOTHINK
Factor=1
LimitFlag=0
Limit=1

[Iterations]
NumOfIterations=1
IterationPace=IterationASAP
StartEvery=60
RandomMin=60
RandomMax=90

[Log]
LogOptions=LogBrief
MsgClassData=0
MsgClassParameters=0
MsgClassFull=0
```

 ［記録］のユーザ・インタフェース

このセクションの内容

［新規仮想ユーザ］ダイアログ・ボックス

このダイアログ・ボックスでは，新しい仮想ユーザ・スクリプトを作成できます。



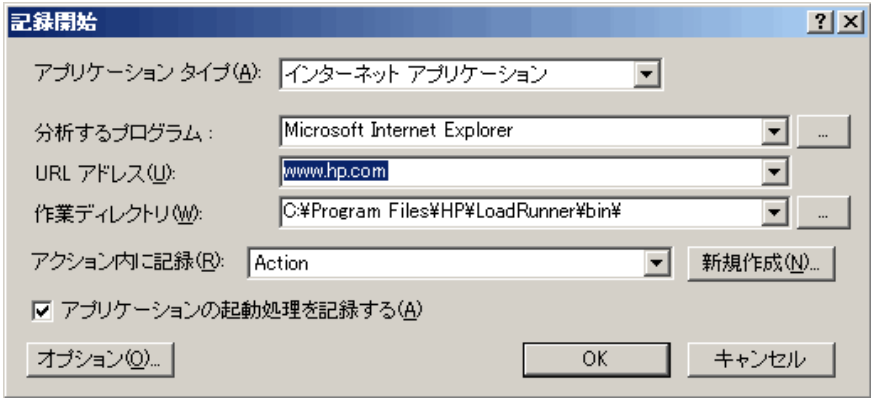
利用方法	**［ファイル］>［新規作成］**
関連タスク	➤ 115 ページの「仮想ユーザ・スクリプトを作成または表示する方法」 ➤ 118 ページの「仮想ユーザ・スクリプトの記録方法」

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
新規シングルプロトコル スクリプト	新しいシングル・プロトコル・スクリプトを作成します。 ➤ **［カテゴリ］ドロップダウン・リスト**：プロトコルのカテゴリのリスト。すべてのプロトコルを表示するには，［**すべてのプロトコル**］を選択します。 ➤ **プロトコル・リスト**：特定のカテゴリのプロトコルのリスト。
新規マルチプロトコル スクリプト	新しいマルチ・プロトコル・スクリプトを作成します。矢印を使用して，［**利用可能なプロトコル**］リストから［**選択されたプロトコル**］リストにプロトコルを移動します。［**作成**］を選択すると，［選択されたプロトコル］リストのプロトコルに基づいて新しいマルチ・プロトコル・スクリプトが作成されます。
最近使用したプロトコルの新規スクリプト	最近使用したプロトコルを使用して新しいスクリプトを作成します。

［記録開始］ダイアログ・ボックス

このダイアログ・ボックスを使用して，スクリプトの記録を開始するのに必要な基本的な詳細を指定できます。



利用方法	**記録を開始すると自動的に開きます。**
重要情報	このダイアログ・ボックスは動的で，選択するオプションや使用するプロトコルに応じて異なります。
関連タスク	118 ページの「仮想ユーザ・スクリプトの記録方法」

ユーザ・インタフェース要素の説明は次のとおりです。

すべてのプロトコル（Java 以外）

UI 要素	説明
	［記録オプション］ダイアログ・ボックスが開きます。ユーザ・インタフェースの詳細については，313 ページの「記録オプション」を参照してください。
アプリケーション タイプ	アプリケーションのタイプ：Win32 アプリケーションまたはインターネット・アプリケーション。たとえば，Web および Oracle NCA スクリプトはインターネット・アプリケーションを記録し，Windows Sockets 仮想ユーザは Win32 アプリケーションを記録します。
プログラム引数（Win32 アプリケーションのみ）	前述で指定した実行ファイルのコマンド・ライン引数を指定します。たとえば，*plus32.exe* にコマンド・ライン・オプション peter@neptune を指定した場合，*plus32.exe* を起動すると，ユーザ *Peter* がサーバ *Neptune* に接続されます。このオプションが表示されるのは，Win32 アプリケーションの場合のみです。
記録するプログラム	記録するブラウザ，インターネット・アプリケーション，または Win32 アプリケーションを選択します。
アクション内に記録	記録先のセクション。

UI 要素	説明
アプリケーションの起動処理を記録する	次の場合は，起動を記録しないことをお勧めします。 ➤ マルチ・アクションの場合。起動が必要なアクションは 1 つだけです。 ➤ アプリケーションで特定の位置まで移動し，その位置から記録を開始する場合。 ➤ 既存のスクリプトに記録する場合。
URL アドレス（インターネット・アプリケーションのみ）	開始する URL アドレス。このオプションが表示されるのは，インターネット・アプリケーションの場合のみです。
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。

Java プロトコル

UI 要素	説明
	［記録オプション］ダイアログ・ボックスが開きます。ユーザ・インタフェースの詳細については，313 ページの「記録オプション」を参照してください。
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。
アクション内に記録	記録先のセクション。
URL	記録を開始する URL。
アプリケーション パラメータ	アプリケーションに必要な追加パラメータ。
アプリケーション メイン クラス	このオプションが表示されるのは，Java アプリケーション・タイプのアプリケーションの場合のみです。
IExplore パス	このオプションが表示されるのは，IExplore タイプのアプリケーションの場合のみです。
実行可能 ¥ バッチ	このオプションが表示されるのは，実行可能 ¥ バッチ・タイプのアプリケーションの場合のみです。

UI 要素	説明
Netscape パス	このオプションが表示されるのは，Netscape タイプのアプリケーションの場合のみです。
アプリケーション タイプ	➤［**Java アプレット**］: Sun のアプレット・ビューアを使って Java アプレットが記録されます。 ➤［**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）。

コード生成通知

このダイアログ・ボックスには，コード生成時に生成される通知および推奨アクションに関する詳細が表示されます。このダイアログ・ボックスは次のプロトコルに適用されます。

- Flex
- Silverlight
- Java over HTTP

利用方法	**［ツール］>［Generation Notifications］**
重要情報	Flex，Silverlight，または Java over HTTP スクリプトのコード生成段階中にエラーが発生した場合，［コード生成通知］ダイアログ・ボックスが自動的に開きます。このダイアログ・ボックスには，各通知および推奨アクションに関する詳細が表示されます。推奨アクションに従い，スクリプトを再生成します。
関連タスク	➤ 679 ページの「外部 Java シリアライザを使用してシリアル化する方法」 ➤ 680 ページの「LoadRunner シリアライザを使用してスクリプトをシリアル化する方法」
関連項目	675 ページの「Flex スクリプトの外部オブジェクト」

ユーザ・インタフェース要素の説明は次のとおりです（ラベルのない要素は山括弧で囲んで示します）。

UI 要素	説明
< 通知リスト >	通知リストには，次のデータ・カラムがあります。 ➤ アクション名 ➤ ステップ名 ➤ 説明 ➤ 推奨操作
< 通知の詳細 >	通知の詳細には，次の 2 つのコンポーネントが含まれています。 ➤ **［詳細］:** スクリプト生成時に生成された例外とその関連メッセージが表示されます。 ➤ **［推奨操作の詳細］:** 例外を修正するアクションを指定します。
生成オプション	［一般］の［記録］ノードを開きます
スクリプトの再生成	仮想ユーザ・スクリプトを再生成します。

4

仮想ユーザ・スクリプトの再生とデバッグ

本章の内容

概念

タスク

リファレンス

概念

再生の概要

スクリプトを作成したら，VuGen インタフェースから直接，スタンドアロン・モードで再生します。この再生では基本的な機能をテストし，パラメータ化が必要な動的な値など，対処が必要なエラーまたは問題を発見するのに役立てます。

再生が正常に実行できたら，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 設定）に統合します。詳細については，関連するユーザ・ガイドを参照してください。

デバッグ機能

VuGen には，仮想ユーザ・スクリプトのデバッグに役立つ次の機能があります。これらの機能は，VBScript および VB アプリケーション・タイプの仮想ユーザでは使用できません。こらの機能には，デバッグ・ツールバーからアクセスします。このツールバーを表示するには，ツールバー領域を右クリックして［**デバッグ**］を選択します。

ステップ実行コマンド

ステップ実行コマンドは，スクリプトを 1 回に 1 行ずつ実行します。これによりスクリプトの実行を追うことができます。スクリプトをステップごとに実行するには，［**仮想ユーザ**］>［**ステップごとに実行**］を選択し，スクリプトの実行が完了するまで［**ステップ**］ボタンをクリックします。

ブレークポイント

ブレークポイントは，スクリプトの特定の場所で実行を一時停止します。これにより，スクリプトの実行中に，あらかじめ定義しておいたポイントで，スクリプトによるアプリケーションへの影響を調査できます。タスクの詳細については，140 ページの「ブレークポイントを設定したスクリプトをデバッグする方法」を参照してください。

ブックマーク

スクリプト・ビューで作業をしているときに，スクリプト内のさまざまな位置にブックマークを置くことができます。ブックマーク間を移動することで，コードの分析やデバッグを実行できます。タスクの詳細については，142 ページの「ブックマークを使用する方法」を参照してください。

移動コマンド

ブックマークを使用せずにスクリプト内を移動するには，［移動］コマンドを使用できます。［**編集**］>［**指定行へ移動**］を選択し，スクリプトの行番号を指定します。この移動方法はツリー・ビューでもサポートされています。

特定のステップまたは関数の再生ログ・メッセージを調べるには，VuGen で該当するステップを選択し，［**編集**］>［**再生ログのステップに移動**］を選択します。出力ウィンドウの［再生ログ］タブ内の対応するステップの位置に，カーソルが置かれます。

追加のデバッグ情報

一般的なデバッグに関するヒント

VuGen は，通常のテキスト・エディタとして使用できます。VuGen で，任意のテキスト・ファイルを開いて編集できます。再生中，下の出力ウィンドウにエラー・メッセージが表示されている場合は，その上でダブルクリックすると，VuGen は問題の原因となっているテキスト行にカーソルを移動します。また，エラー・コードにカーソルを置いて F1 キーを押すと，オンライン・ヘルプのエラー・コードの説明が表示されます。

C 関数を使用した追跡

C インタプリタの追跡オプション（バージョン 230 以上）を使用して，仮想ユーザ・スクリプトをデバッグできます。ci_set_debug ステートメントを使って，スクリプト内の特定の位置で，追跡とデバッグのオン / オフを切り替えることができます。

```
ci_set_debug(ci_this_context, int debug, int trace);
```

たとえば，スクリプトに次のステートメントを追加できます。

```
ci_set_debug(ci_this_context, 1, 1) /* 追跡とデバッグをオンにする */
ci_set_debug(ci_this_context, 0, 0) /* 追跡とデバッグをオフにする */
```

付加的な C 言語のキーワード

VuGen で C スクリプトを実行すると，VuGen のパーサは組み込み C インタプリタを使ってスクリプト内の関数を解析します。標準パーサのライブラリに含まれていないキーワードを追加できます。標準設定では，インストール中に *size_t* や *DWORD* といった一般的な C++ キーワードが追加されます。リストを編集して，環境に合ったキーワードを追加します。

キーワードを追加するには，次の手順で行います。

1. vugen_extra_keywords.ini ファイルを開きます。このファイルはお使いのコンピュータの <Windows> または <Windows>/System ディレクトリにあります。
2. EXTRA_KEYWORDS_C セクションに，C インタプリタ用のキーワードを追加します。

   このファイルの形式は次のとおりです。

```
[EXTRA_KEYWORDS_C]
FILE=
size_t=
WORD=
DWORD=
LPCSTR=
```

再生出力の検証

再生出力を見ます（VuGen から，あるいは VuGen ドライバの出力を表示する **output.txt** ファイルから）。また，より詳細な再生出力を得るために，VuGen の実行環境の設定オプションで，ログの記録を拡充することもできます。

Web 仮想ユーザのデバッグ機能

VuGen には，Web 仮想ユーザ・スクリプトのデバッグに役立つ付加的なツールとして，実行時ビューア（オンライン・ブラウザ）と結果サマリ・レポートがあります。

- Web 仮想ユーザ・スクリプトの実行時に実行時ビューアを表示するように VuGen を設定することができます。実行時ビューアは，VuGen 向けに開発されました。仮想ユーザ・スクリプトの記録に使用するブラウザとは無関係です。実行時ビューアには，仮想ユーザによってアクセスされる各 Web ページが表示されます。これにより仮想ユーザが正しい Web ページにアクセスしているかどうかを確認できるので，Web 仮想ユーザ・スクリプトのデバッグ時に役立ちます。
- Web 仮想ユーザに対して，スクリプトの実行時に結果サマリ・レポートを生成させるかどうかを指定できます。結果サマリ・レポートとは，Web 仮想ユーザ・スクリプト内の各ステップの成功や失敗についてまとめたもので，各ステップで返された Web ページも確認できます。結果サマリ・レポートを使った作業の詳細については，［**表示**］>［**テスト結果**］を選択し，F1 キーを押して，オンライン・ヘルプを参照してください。

  ユーザ・インタフェースの詳細については，77 ページの「[表示] タブ」を参照してください。

注：仮想ユーザに結果サマリ・レポートを生成させると，トランザクションの処理時間が増える場合があります。仮想ユーザから結果サマリ・レポートを生成できるのは，VuGen から実行した場合だけです。スクリプトを Controller，または Business Process Monitor から実行した場合は，仮想ユーザからレポートは生成されません。

タスク

仮想ユーザ・スクリプトを再生する方法

このタスクでは，仮想ユーザ・スクリプトを再生する方法について説明します。

このタスクでは，次の手順を実行します。

- 136 ページの「再生オプションを設定する」
- 136 ページの「実行環境を設定する」
- 136 ページの「スクリプトを再生する」
- 136 ページの「ログで詳細情報を確認する」

1 再生オプションを設定する

［一般オプション］ダイアログ・ボックスの［再生］および［表示］タブで必要なオプションを指定します。ユーザ・インタフェースの詳細については，73 ページの「［一般オプション］ダイアログ・ボックス」を参照してください。

2 実行環境を設定する

［実行環境設定］ダイアログ・ボックスには，実行または再生時のスクリプトの動作に影響する設定があります。ユーザ・インタフェースの詳細については，419 ページの「実行環境の設定」を参照してください。

3 スクリプトを再生する

［**仮想ユーザ**］>［**実行**］を選択して，スクリプトを再生します。

4 ログで詳細情報を確認する

記録および再生段階のスクリプトの動作に関する詳細情報を出力ウィンドウで確認できます。ユーザ・インタフェースの詳細については，87 ページの「出力ウィンドウ」を参照してください。

コマンド・プロンプトから仮想ユーザ・スクリプトを実行する方法

このタスクでは，VuGen のユーザ・インタフェースを使わずに，コマンド・プロンプトや Windows の［ファイル名を指定して実行］ダイアログ・ボックスから仮想ユーザ・スクリプトをテストする方法について説明します。

コマンド・ラインや［ファイル名を指定して実行］ダイアログ・ボックスからスクリプトを実行するには，次の手順で行います。

1 ［**スタート**］>［**すべてのプログラム**］>［**アクセサリ**］>［**コマンド プロンプト**］を選択すると，［**コマンド プロンプト**］ウィンドウを開きます。または，［**スタート**］>［**ファイル名を指定して実行**］を選択して，［ファイル名を指定して実行］ダイアログ・ボックスを開きます。

2 次のとおりに入力して，**Enter** キーを押します。

```
<installation_dir>/bin/mdrv.exe -usr <script_name> -vugen_win 0
```

script_name は，*.usr* スクリプト・ファイルのフル・パスです。たとえば，**c:¥temp¥mytest¥mytest.usr** などになります。

mdrv プログラムによって，ユーザ・インタフェースなしでスクリプトの 1 つのインスタンスが実行されます。出力ファイルで実行時の情報を確認します。

3 次の形式を使用して，スクリプトに渡す引数を指定できます。

```
スクリプト名　- 引数 引数値　- 引数 引数値
```

4 次の例に示すように，Load Generator およびスクリプトの実行回数を指定できます。

```
script1  -host pc4  -loop 5
```

コマンド・ライン解析関数またコマンド・ラインでの引数の使い方の詳細については，*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。

UNIX コマンド・ラインから仮想ユーザ・スクリプトを実行する方法

VuGen を使って UNIX ベースの仮想ユーザを作成するときには，記録されたスクリプトを UNIX プラットフォームで実行できることを確認する必要があります。このタスクでは，この確認方法および UNIX コマンドから仮想ユーザ・スクリプトを実行する方法について説明します。

1 VuGen でスクリプトが再生されるかを確認する

UNIX でスクリプトを実行する前に，VuGen でスクリプトを再生して，スクリプトが Windows で動作することを確認します。スクリプトの編集とデバッグは VuGen で行う方が簡単なので，この確認をすることをお勧めします。タスクの詳細については，136 ページの「仮想ユーザ・スクリプトを再生する方法」を参照してください。

2 スクリプト・ファイルを UNIX サーバにコピーする

スクリプト・ファイルを UNIX サーバに転送します。

3 verify_generator を使用して，UNIX マシン上で仮想ユーザの設定を確認する

すべての仮想ユーザを 1 台のホストで実行する場合は，次のように入力します。

```
verify_generator
```

verify_generator は，設定が正しければ「**OK**」を返します。「**Failed**」が返された場合は，正しく設定し直す方法を示します。

この検証ユーティリティの詳細については，次を入力してください。

```
verify_generator [-v]
```

検証ユーティリティは，通信パラメータと，すべての種類の仮想ユーザの互換性について，ローカル・ホストを検査します。仮想ユーザ環境で次の項目が検査されます。

- 最低 128 のファイル記述子を持っていること。
- **.rhost** ファイルの権限（**-rw-r--r--**）が正しいこと。
- rsh を使ったホスト通信が可能なこと。通信が不可能な場合は，**.rhosts** 内のホスト名が検査されます。
- **M_LROOT** が定義されていること

- **.cshrc** で正しい **M_LROOT** が定義されていること。
- ホーム・ディレクトリに **.cshrc** が存在すること。
- 現在のユーザが **.cshrc** の所有者であること。
- LoadRunner が **$M_LROOT** にインストールされていること。
- 実行可能ファイルが実行可能な権限を持っていること。
- **PATH** に **$M_LROOT/bin** および **/usr/bin** が含まれていること。
- **rstatd** デーモンが存在し，起動していること。

4 スクリプトを実行する

仮想ユーザ・スクリプトのディレクトリから，**run_db_vuser** シェル・スクリプトを使って，スクリプトをスタンドアロン・モードで実行します。

```
run_db_vuser.sh < コマンド > script_name.usr
```

run_db_vuser シェル・スクリプトには，次のコマンド・ライン・オプションがあります。

コマンド	説明
--help	使用可能なオプションが表示されます（このオプションの先頭には，ダッシュを 2 つ付ける必要があります）。
-cpp_only	スクリプトを対象に cpp のみ（プリプロセス）が実行されます。
-cci_only	スクリプトを対象に cci のみ（プリコンパイル）を実行して，拡張子が .ci のファイルが作成されます。cpp が成功しないと，cci は実行できません。
-driver < *ドライバのパス* >	指定したドライバ・プログラムが使用されます。各データベースに固有のドライバ・プログラムが，/bin ディレクトリにあります。たとえば，/bin ディレクトリにある **CtLib** のドライバは **mdrv** です。このオプションを使って，外部ドライバを指定することができます。
-exec_only	仮想ユーザの .ci ファイルが実行されます。このオプションは，有効な .ci ファイルが存在するときにのみ使用できます。
-ci <*ci ファイル名* >	指定した .ci ファイルが実行されます。
-out 出力パス	指定したディレクトリに結果が格納されます。

標準設定では，*run_db_vuser.sh* は冗長モードで **cpp**，**cci**，**execute** を実行します。<VuGen のインストール先 >¥bin ディレクトリにあるドライバを使用し，仮想ユーザ・スクリプト・ディレクトリ内の出力ファイルに結果が保存されます。必ず *.usr* ファイルを指定する必要があります。カレント・ディレクトリがスクリプト・ディレクトリでない場合は，*.usr* ファイルのフル・パスを指定します。

たとえば，次のコマンド・ラインは仮想ユーザ・スクリプト test1 を実行し，出力ファイルをディレクトリ results1 に格納します。結果ディレクトリは，自動的に作成されないので，既存のディレクトリでなければなりません。

```
run_db_vuser.sh -out /u/joe/results1  test1.usr
```

ブレークポイントを設定したスクリプトをデバッグする方法

次の手順では，ブレークポイントを使って作業する方法について説明します。概念の詳細については，132 ページの「デバッグ機能」を参照してください。

- 141 ページの「ブレークポイントを追加する」
- 141 ページの「ブレークポイントを有効または無効にする」
- 141 ページの「ブレークポイントを管理する」
- 141 ページの「ブレークポイントを設定したスクリプトを実行する」

ブレークポイントを追加する

［**挿入**］>［**ブレークポイントの設定 / 解除**］を選択するか，デバッグ・ツールバーの［**ブレークポイントの設定 / 解除**］ボタンをクリックします。ブレークポイントの記号（●）がスクリプトの左余白に表示されます。

ブレークポイントを有効または無効にする

ブレークポイントを無効にするには，ブレークポイントの記号のある行にカーソルを合わせて，デバッグ・ツールバーの［**ブレークポイントの有効化 / 無効化**］ボタンをクリックします。ブレークポイント記号の中の白い点が表示され，そのブレークポイントが無効であることを示します。あるブレークポイントを無効にすると，その次のブレークポイントでスクリプトの実行が一時停止します。ブレークポイントを有効にするにはもう一度ボタンをクリックします。

ブレークポイントを削除するには，ブレークポイントの記号のある行にカーソルを合わせて，［**ブレークポイントの設定 / 解除**］ボタンをクリックするか，F9 キーを押します。

ブレークポイントを管理する

［ブレークポイント］ダイアログ・ボックスを使用すると，1 つのユーザ・インタフェースからすべてのブレークポイントの追加，削除，有効化，無効化，および条件の設定を実行できます。ユーザ・インタフェースの詳細については，146 ページの「［ブレークポイント］ダイアログ・ボックス」を参照してください。

ブレークポイントを設定したスクリプトを実行する

通常どおり，スクリプトの実行を開始します。VuGen がブレークポイントに到達すると，スクリプトの実行が停止されます。ブレークポイントまでのスクリプト実行の影響を調査して，必要な変更を加え，そのブレークポイントからスクリプトの実行を再開します。

実行を再開するには，［**仮想ユーザ**］>［**実行**］を選択します。再開すると，別のブレークポイントに到達するか，スクリプトが終了するまで，スクリプトの実行が続けられます。

ブックマークを使用する方法

スクリプト・ビューで作業をしているときに，スクリプト内のさまざまな位置にブックマークを置くことができます。ブックマーク間を移動することで，コードの分析やデバッグを実行できます。次の手順では，ブックマークを使って作業する方法について説明します。

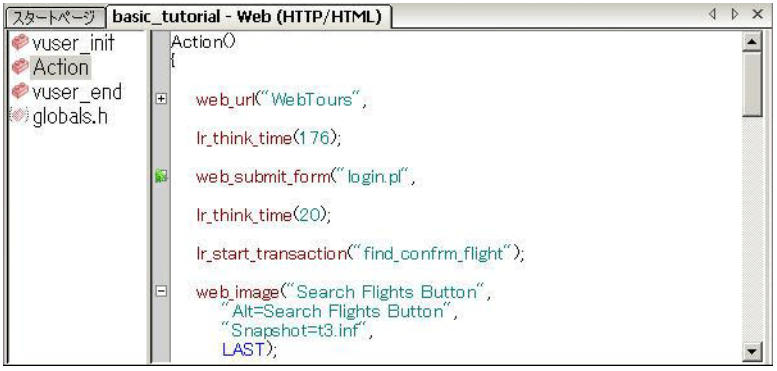
➤ 142 ページの「ブックマークを作成する」

➤ 143 ページの「ブックマークを削除する」

➤ 143 ページの「ブックマーク間を移動する」

ブックマークを作成する

必要な位置にカーソルを置いて Ctrl + F2 キーを押します。アイコンがスクリプトの左余白に置かれます。



ブックマークを削除する

ブックマークを削除するには，必要な位置にカーソルを置いて Ctrl + F2 キーを押します。ブックマーク・アイコンが左余白から削除されます。

ブックマーク間を移動する

- 次のブックマークに移動するには，**F2** キーを押します。
- 前のブックマークに戻るには，**Shift + F2** キーを押します。

［**編集**］>［**ブックマーク**］メニュー項目を通じても，ブックマークの作成や，ブックマーク間の移動が可能です。

ブックマーク間の移動は現在のアクションの中でのみ実行できます。別のアクションのブックマークに移動するには，そのアクションを左側の表示枠で選択した後，F2 キーを押します。

リファレンス

再生中に生成されるファイル

本項では，仮想ユーザを再生したときに何が起きるかを説明します。

1 プリプロセッサ用のコマンド・ライン・パラメータを含む **options.txt** ファイルが作成されます。

2 関連するすべての .c および .h ファイルに対する「includes」を含む **Vuser.c** ファイルが作成されます。

3 開発用ファイルからマクロ定義およびプリコンパイラ指示子などを「挿入する」ために C のプリプロセッサ **cpp.exe** が呼び出されます。

次のコマンド・ラインが使用されます。

```
cpp -foptions.txt
```

4 **pre_cci.c** ファイルが作成されます。これも C ファイルです（**pre_cci.c** は，**options.txt** ファイルで定義されます）。このプロセスのあらゆる出力を含む **logfile.log** ファイル（このファイルも **options.txt** で定義されます）が作成されます。logfile.log ファイルは，プリプロセス処理の段階で何も問題がなければ，空のはずです。このファイルが空でなければ，コンパイルの次の段階で致命的なエラーが発生し，ほぼ確実に失敗します。

5 仮想ユーザ・ドライバ・プログラムによって実行時に解釈される，プラットフォームに依存する疑似バイナリ・ファイル（.ci）を作成するために，C コンパイラ **cci.exe** が起動されます。cci は **pre_cci.c** ファイルを入力として受け取ります。

6 **pre_cci.ci** ファイルが次のように作成されます。

```
cci   -errout  c:¥tmp¥Vuser¥logfile.log   -c  pre_cci.c
```

7 **logfile.log** ファイルは，コンパイル時の出力を格納するログ・ファイルです。

8 **pre_cci.ci** ファイルの名前はここで **Vuser.ci** に変わります。

コンパイル時には警告とエラーの両方が生成される可能性があり，ドライバはこのプロセスの結果を知らないので，ドライバはまず **logfile.log** ファイルにエントリがあるか調べます。logfile.log ファイルにエントリがある場合，ドライバは，**Vuser.ci** ファイルが生成されているかどうかを調べます。ファイル・サイズがゼロでなければ，cci がコンパイルに成功したということです。ファイル・サイズがゼロであれば，コンパイルは失敗しており，エラー・メッセージが表示されます。

9 関連するドライバが実行され，入力データとして **.usr** ファイルと **Vuser.ci** ファイルが使われます。次に例を示します。

```
mdrv.exe  -usr c:¥tmp¥Vuser¥Vuser.usr    -out c:¥tmp¥Vuser
-file c:¥tmp¥Vuser¥Vuser.ci
```

.usr ファイルは，ドライバ・プログラムにどのデータベースが使用されているのかを知らせるために必要です。この段階で，実行のためにロードする必要があるライブラリがわかります。

10 実行中に出力されたすべてのメッセージを含む **output.txt** が（「out」変数によって定義されたパス内に）作成されます。これは，VuGen の実行時の出力ウィンドウおよび VuGen のメイン・ウィンドウの下部の表示枠に表示されるものとまったく同じです。

options.txt ファイルの例

```
-DCCI
-D_IDA_XL
-DWINNT
-Ic:¥tmp¥Vuser（仮想ユーザ・インクルード・ファイルの名前と場所）
-IE:¥LRUN45B2¥include（インクルード・ファイルの名前と場所）
-ec:¥tmp¥Vuser¥logfile.log（出力ログ・ファイルの名前と場所）
 c:¥tmp¥Vuser¥VUSER.c( 処理されるファイルの名前と場所 )
```

Vuser.c ファイルの例

```
#include "E:¥LRUN45B2¥include¥lrun.h"
#include "c:¥tmp¥web¥init.c"
#include "c:¥tmp¥web¥run.c"
#include "c:¥tmp¥web¥end.c"
```

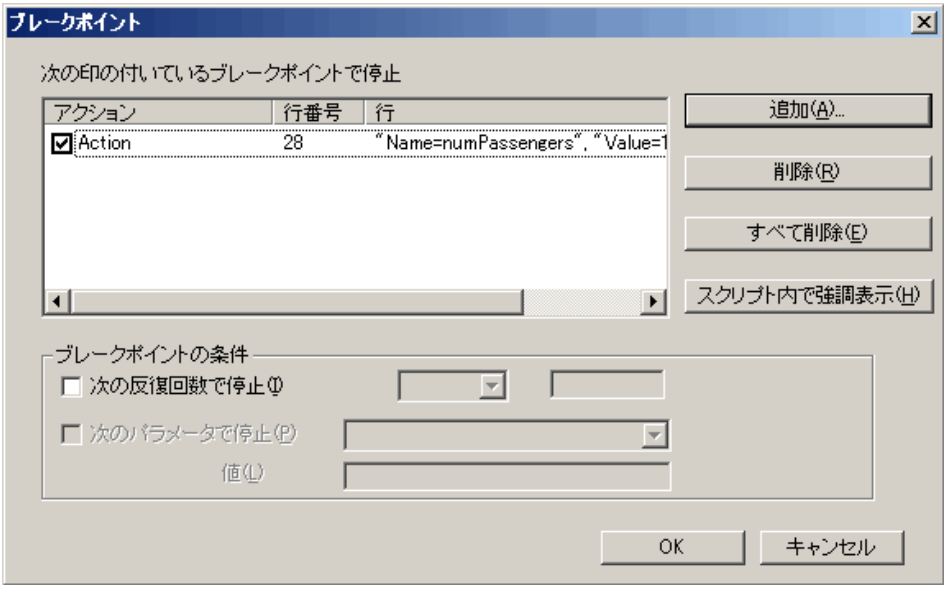
再生のユーザ・インタフェース

このセクションの内容

➤［ブレークポイント］ダイアログ・ボックス（146 ページ）

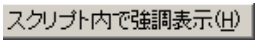
［ブレークポイント］ダイアログ・ボックス

このダイアログ・ボックスでブレークポイントを管理できます。



利用方法	**［編集］>［ブレークポイント］**
関連タスク	140 ページの「ブレークポイントを設定したスクリプトをデバッグする方法」

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
	指定したセクションおよび行番号に新しいブレークポイントが追加されます。
	選択したブレークポイントが削除されます。
	すべてのブレークポイントが削除されます。
	ブレークポイントを強調表示したスクリプトが表示されます。一度に強調表示できるブレークポイントは 1 つだけです。
< ブレークポイント・グリッド >	スクリプト内の現在のブレークポイントとその位置のリスト。ブレークポイントを有効にするには，そのブレークポイントの横のチェックボックスを選択します。ブレークポイントを無効にするには，チェックボックスをクリアします。
ブレークポイントの条件	➤ **[次の反復回数で停止]**：指定した反復回数の後にスクリプトが一時停止されます。 ➤ **[次のパラメータで停止]**：パラメータ **X** が指定した値の場合，スクリプトが一時停止されます。

5

負荷テスト用スクリプトの準備

本章の内容

概念

タスク

リファレンス

概念

パスワードのエンコード

パスワードを暗号化し，その結果生成された文字列をスクリプト内の引数またはパラメータ値として使用できます。たとえば，ユーザがパスワードを入力しなければならないフォームが Web サイトにあるとします。異なるパスワードにサイトがどのように応答するかをテストしたいが，同時にパスワードの安全性も保護したいとします。［**Password Encoder**］を使えばパスワードを暗号化し，テーブルに値を安全な形式で入力できます。

パスワードをエンコードするには，［**スタート**］>［**すべてのプログラム**］>［**LoadRunner**］>［**Tools**］>［**Password Encoder**］を選択します。

タスクの詳細については，155 ページの「パスワードを暗号化する方法」を参照してください。

ユーザ・インタフェースの詳細については，170 ページの「Password Encoder」を参照してください。

テキストの暗号化

スクリプト内のテキストを暗号化して，パスワードなどの機密性の高いテキスト文字列を保護できます。暗号化は，ユーザ・インタフェースから自動的に行うことができます。また，プログラムを使って手動でもできます。文字列は，いつでも復号して元の値に戻せます。暗号化した文字列は，スクリプト中に暗号化された文字列として表れます。VuGen は 32 ビットの暗号化をサポートしています。

スクリプトで暗号化された文字列を使用するには，**lr_decrypt** 関数を使用して復号する必要があります。

```
lr_start_transaction(lr_decrypt("3c29f4486a595750"));
```

タスクの詳細については，154 ページの「テキストを暗号化 / 復号する方法」を参照してください。

トランザクションの概要

*トランザクション*を定義することによって，サーバのパフォーマンスを評価できます。各トランザクションは，サーバが特定の仮想ユーザ要求に応答するまでにかかる時間を測定します。これらの要求は，1 つのクエリに対する応答を待つような簡単な処理の場合や，いくつかのクエリを発行してレポートを作成するといった複雑な処理の場合があります。

トランザクションを測定するために，タスクの開始とタスクの終了を示す仮想ユーザ関数を挿入します。スクリプトには，任意の数のトランザクションを異なる名前で挿入することができます。

LoadRunner に対して，コントローラは各トランザクションの実行に要する時間を測定します。テスト実行の後，アナリシスのグラフとレポートを使用して，トランザクションごとのサーバ・パフォーマンスを分析します。

スクリプトを作成する前に，測定の対象となるビジネス・プロセスを決めておきます。次に，それぞれのビジネス・プロセスまたはサブプロセスをトランザクションとして指定します。

トランザクション名に "_" または "@" の記号を含めることはできません。これらの記号を含めると，Analysis のクロス結果グラフを開くときにエラーが発生します。

トランザクションは記録中または記録後に作成できます。タスクの詳細については，156 ページの「トランザクションを挿入する方法」を参照してください。

ランデブー・ポイント

負荷テストを実行する際には，システムに高いユーザ負荷がかかっている状態をエミュレートする必要があります。そのためには，複数の仮想ユーザを同期させ，まったく同じ瞬間にタスクを実行させます。複数の仮想ユーザを同時に動作させるには，「*ランデブー・ポイント*」を作成します。ある仮想ユーザがランデブー・ポイントに到達すると，ほかのすべての仮想ユーザがランデブー・ポイントに到着するまで，その仮想ユーザは待機させられます。指定した数の仮想ユーザが到着すると，仮想ユーザが解放されます。

仮想ユーザ・スクリプトにランデブー・ポイントを挿入することによって，待機場所を指定します。仮想ユーザは，スクリプトを実行してランデブー・ポイントに到達すると，スクリプトの実行を一時停止し，コントローラからの再開許可を待ちます。仮想ユーザは，ランデブー・ポイントから解放されると，スクリプト内の次のタスクを実行します。

タスクの詳細については，を参照 160 ページの「ランデブー・ポイントを挿入する」してください。

注：ランデブー・ポイントは *Action* セクションでのみ有効です。*init* または *end* セクションでは無効になります。

思考遅延時間

連続するアクションの合間のユーザの待ち時間を「*思考遅延時間*」といいます。仮想ユーザは，**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できます。仮想ユーザ・スクリプトを記録すると，VuGen によって実際の思考遅延時間が記録され，適切な **lr_think_time** ステートメントが仮想ユーザ・スクリプトに挿入されます。記録された **lr_think_time** ステートメントは後で編集できます。また，手動でも **lr_think_time** ステートメントを仮想ユーザ・スクリプトに追加できます。

注：思考遅延時間のステップを挿入するには，［**挿入**］>［**新規ステップ**］>［**思考遅延時間**］を選択します。タスクの詳細については，164 ページの「思考遅延時間ステップを挿入する」を参照してください。Java 仮想ユーザ・スクリプトを記録する場合，**lr_think_time** ステートメントは仮想ユーザ・スクリプトに挿入されません。

実行環境の設定を使用して，仮想ユーザ・スクリプトを実行するときの **lr_think_time** ステートメントの処理方法を設定できます。ユーザ・インタフェースの詳細については，453 ページの「［一般］>［思考遅延時間］ノード」を参照してください。

タスク

テキストを暗号化 / 復号する方法

このタスクでは，コード内で文字列を暗号化および復号する方法について説明します。参考情報については，150 ページの「テキストの暗号化」を参照してください。

文字列を暗号化するには，次の手順を実行します。

1 ツリー・ビューを表示できるプロトコルの場合は，スクリプト・ビューでスクリプトを表示します。[**表示**] > [**スクリプト ビュー**] を選択します。

2 暗号化するテキストを選択します。

3 右クリック・メニューから [**文字列を暗号化（***対象文字列***）**] を選択します。

暗号化された文字列を元に戻すには，次の手順を実行します。

1 ツリー・ビューを表示できるプロトコルの場合は，スクリプト・ビューでスクリプトを表示します。[**表示**] > [**スクリプト ビュー**] を選択します。

2 元に戻す文字列を選択します。

3 右クリック・メニューから [**暗号化文字列を復元（***対象文字列***）**] を選択します。

lr_decrypt 関数の詳細については，*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。

パスワードを暗号化する方法

このタスクでは，パスワードを暗号化する方法について説明します。パスワードを暗号化し，その結果生成された文字列をスクリプト内の引数またはパラメータ値として使用できます。たとえば，ユーザがパスワードを入力しなければならないフォームが Web サイトにあるとします。異なるパスワードにサイトがどのように応答するかをテストしたいが，同時にパスワードの安全性も保護したいとします。

パスワードを暗号化するには，次の手順を実行します。

1 Windows メニューの［**スタート**］>［**すべてのプログラム**］>［**LoadRunner**］>［**Tools**］>［**Password Encoder**］を選択します。［Password Encoder］ダイアログ・ボックスが開きます。

2［**パスワード**］ボックスにパスワードを入力します。

3［**生成**］をクリックします。［Password Encoder］によってパスワードが暗号化され，暗号化された値が［**エンコード文字列**］フィールドに表示されます。

4［**コピー**］ボタンを使用して，暗号化された値をコピーし，データ・テーブルに貼り付けます。

VuGen から Controller のシナリオを作成する方法

注：次の項の内容は，LoadRunner のみを対象としています。スクリプトをビジネス・プロセス・プロファイルに統合する方法の詳細については，*HP Business Availability Center* のマニュアルを参照してください。

通常は，LoadRunner Controller でシナリオを作成します。現在のスクリプトを使用して，基本的なシナリオを VuGen で作成することもできます。

このタイプのシナリオを作成するには，［**ツール**］>［**Controller のシナリオを作成**］を選択し，ダイアログ・ボックスを完了します。ユーザ・インタフェースの詳細については，168 ページの「［シナリオの作成］ダイアログ・ボックス」を参照してください。

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。

トランザクションを挿入する方法

注：

- トランザクションにトランザクションが含まれる「入れ子」のトランザクションを作成できます。トランザクションを入れ子にする場合は，含まれる側のトランザクションを，含む側のトランザクションをよりも前に閉じます。そうしないとトランザクションが正しく解析されません。ただし，トランザクションは 1 つの **action** セクション内に含める必要があります。
- トランザクション名は一意である必要があり，先頭は文字または数字にする必要があります。トランザクション名には文字または数字を使用できます。. , : # / ¥ " < は使用しないでください。

次の手順では，トランザクションを挿入するための各種方法について説明します。参考情報については，151 ページの「トランザクションの概要」を参照してください。

- 158 ページの「記録中にトランザクションを挿入する」
- 156 ページの「スクリプト・ビューまたはツリー・ビューでトランザクションを挿入する」
- 157 ページの「トランザクション・エディタを使用して，トランザクションを挿入する」

スクリプト・ビューまたはツリー・ビューでトランザクションを挿入する

- 記録の後にトランザクションの開始をマークするには，［**挿入**］>［**トランザクション開始**］を選択し，トランザクション名を入力します。
- トランザクションの終了をマークするには，［**挿入**］>［**トランザクション終了**］を選択します。VuGen によって **lr_end_transaction** ステートメントが仮想ユーザ・スクリプトに挿入されます。

トランザクション・エディタを使用して，トランザクションを挿入する

注：このオプションはサムネイルを持つスクリプトにのみ適用されます。

- 標準設定では，トランザクション・リストには，スクリプトの主要なステップに対するサーバの応答を測定する，エラーの生じないトランザクションのみが表示されます。主要でないステップ，クライアント側のトランザクション，またはエラーの生じたトランザクションは表示されません。したがって，トランザクション・リストの上部に［**トランザクション（2 / 4)**］のようなキャプションが表示される場合があります。すべてのトランザクションを表示するには，158 ページの「トランザクションを表示する方法」を参照してください。
- 複数ステップのトランザクションをマークするには，タスク表示枠から［**トランザクション**］を選択し，［**新規トランザクション**］を選択します。トランザクションの開始とするサムネイルの前の場所にカーソルをドラッグし，クリックします。トランザクションの終了とするサムネイルの前の場所にカーソルを置き，クリックします。タイトルをクリックして，トランザクションの名前を入力します。
- 単一ステップのトランザクションをマークするには，サムネイルをクリックして右クリック・メニューから［**新規シングル ステップ トランザクション**］を選択します。新しいトランザクションの名前を入力するダイアログ・ボックスが表示されます。後からトランザクションを拡張するには，トランザクションの括弧をドラッグしてトランザクションに追加のステップを含めます。
- トランザクションの開始点を変更するには，トランザクションの左大括弧を新しい場所にドラッグします。トランザクションの終了点を変更するには，トランザクションの右大括弧を新しい場所にドラッグします。
- トランザクションの追加，名前の変更，削除には，右クリック・メニューを使用します。

ユーザ・インタフェースの詳細については，171 ページの「トランザクション リスト」を参照してください。

記録中にトランザクションを挿入する

➤ トランザクションの開始をマークするには，スクリプトの記録中に記録ツールバーの［**トランザクション開始マーカを挿入**］ボタンをクリックし，トランザクション名を入力します。［**OK**］をクリックすると，VuGen によって **lr_start_transaction** ステートメントが仮想ユーザ・スクリプトに挿入されます。

➤ トランザクションの終了をマークするには，記録ツールバーの［**トランザクション終了マーカを挿入**］ボタンをクリックし，閉じるトランザクションを選択します。［**OK**］をクリックすると，VuGen によって **lr_end_transaction** ステートメントが仮想ユーザ・スクリプトに挿入されます。

トランザクションを表示する方法

次の手順では，タスク表示枠で各種トランザクションを表示する方法について説明します。参考情報については，151 ページの「トランザクションの概要」を参照してください。

➤ 158 ページの「非表示のトランザクションを表示する」

➤ 159 ページの「エラーの生じたトランザクションを表示する」

➤ 159 ページの「主要でないステップのトランザクションを表示する」

非表示のトランザクションを表示する

主要でないトランザクションやクライアント側のトランザクションといった非表示のトランザクションを表示するには，トランザクション・リスト下部の［**非表示のトランザクションを表示**］の隣のボタンをクリックします。非表示のトランザクションが灰色で表示されます。非表示にするには，再度ボタンをクリックします。

エラーの生じたトランザクションを表示する

エラーの生じたトランザクションはサーバ・ステップを測定しないトランザクションか，使用できない名前のついたトランザクションです。エラーの生じたトランザクションを表示するには，［**エラーのあるトランザクションを表示**］ボタンをクリックします。VuGen により，エラーの生じたトランザクションが赤で表示されます。非表示にするには，再度ボタンをクリックします。

主要でないステップのトランザクションを表示する

主要でないステップのトランザクションを表示するには，すべてのサムネイルを表示する必要があります。［**表示**］>［**すべてのサムネイルを表示**］を選択します。トランザクション・エディタに，スクリプト内のすべてのステップのサムネイルとトランザクションが表示されます。

 負荷テスト用のスクリプトを準備する方法

このタスクでは，負荷テスト用のスクリプトを準備するために，スクリプトのデバッグ後に行うことができる追加の処理について説明します。このタスクのすべての項目はオプションです。

このタスクでは，次の手順を実行します。

- 160 ページの「ステップを挿入する」
- 160 ページの「ステップを編集する」
- 160 ページの「トランザクションを挿入する」
- 160 ページの「ランデブー・ポイントを挿入する」
- 161 ページの「コメントを挿入する」
- 161 ページの「VuGen 関数を挿入する」
- 161 ページの「ログ・メッセージを挿入する」
- 162 ページの「ファイルをスクリプトに追加する」
- 162 ページの「スクリプトを同期する（RTE 仮想ユーザのみ）」
- 163 ページの「[テスト結果］ウィンドウ・オプションを設定する」
- 163 ページの「スクリプトの再生とデバッグを行う」

ステップを挿入する

思考遅延時間のステップ，デバッグ・メッセージ，出力メッセージなど，さまざまなステップをスクリプトに挿入できます。タスクの詳細については，163 ページの「ステップをスクリプトに挿入する方法」を参照してください。

ステップを編集する

ツリー・ビューでステップを右クリックし，［**プロパティ**］を選択することで，個々のステップを変更できます。

トランザクションを挿入する

トランザクションをスクリプトに挿入するには，［**挿入**］>［**トランザクション開始**］メニュー項目と［**挿入**］>［**トランザクション終了**］メニュー項目を使用します。タスクの詳細については，156 ページの「トランザクションを挿入する方法」を参照してください。

ランデブー・ポイントを挿入する

*ランデブー・ポイント*を作成することで，複数の仮想ユーザがタスクを同時に実行するように同期できます。ある仮想ユーザがランデブー・ポイントに到達すると，ほかのすべての仮想ユーザがランデブー・ポイントに到着するまで，その仮想ユーザは待機させられます。指定した数の仮想ユーザが到着すると，仮想ユーザが解放されます。

ランデブー・ポイントは，次のいずれかの方法で挿入できます。

- 記録中にランデブー・ポイントを挿入するには，記録ツールバーの［**ランデブー**］ボタンをクリックし，ダイアログ・ボックスで名前（大文字と小文字は区別されない）を入力します。
- 記録後にランデブー・ポイントを挿入するには，［**挿入**］>［**ランデブー**］を選択し，ダイアログ・ボックスで名前（大文字と小文字は区別されない）を入力します。

ランデブー・ポイントを挿入すると，VuGen によって **lr_rendezvous** 関数が仮想ユーザ・スクリプトに挿入されます。たとえば，次の関数は rendezvous1 という名前のランデブー・ポイントを定義します。

```
lr_rendezvous("rendezvous1");
```

概念の詳細については，152 ページの「ランデブー・ポイント」を参照してください。

コメントを挿入する

VuGen では仮想ユーザの実行アクション間にコメントを挿入できます。コメントを挿入して，動作内容を説明したり，特定の操作に関する情報を記入したりできます。たとえば，データベースの動作を記録している場合は，「これは最初のクエリです」のように最初のクエリであることを示すコメントを挿入できます。

コメントは，次のいずれかの方法で挿入できます。

- 記録中にコメントを挿入するには，記録ツールバーの［**コメントを挿入**］ボタンをクリックし，［コメントを挿入］ダイアログ・ボックスで必要なコメントを入力します。
- 記録後にコメントを挿入するには，［**挿入**］>［**コメント**］を選択し，［コメントを挿入］ダイアログ・ボックスで必要なコメントを入力します。

次のスクリプトの抜粋は，仮想ユーザ・スクリプトに挿入されたコメントを示します。

```
/*
* これはコメントです
*/
```

VuGen 関数を挿入する

この時点で VuGen 関数を挿入できます。役に立つ関数のリストについては，165 ページの「役に立つ VuGen 関数」を参照してください。

ログ・メッセージを挿入する

VuGen を使って **lr_log_message** 関数を生成し，仮想ユーザ・スクリプトに挿入できます。たとえば，データベースを対象としたアクションを記録しているときに，最初のクエリを示すために「これは最初のクエリです」というメッセージを挿入できます。

ログ・メッセージを挿入するには，［**挿入**］>［**ログ メッセージ**］を選択し，メッセージを入力します。

ファイルをスクリプトに追加する

ファイルをスクリプト・ディレクトリに追加し，スクリプトの実行時に使用できるようにすることが可能です。ファイルがテキスト・ベースの場合，VuGen のエディタに表示して編集できます。

ファイルをスクリプトに追加するには，［**ファイル**］>［**ファイルをスクリプトに追加**］を選択します。

スクリプトを同期する（RTE 仮想ユーザのみ）

同期化関数を追加して，仮想ユーザ・スクリプトの実行をアプリケーションの出力と同期させることができます。同期化は，RTE 仮想ユーザ・スクリプトにのみ適用されます。

次の同期化関数を使用できます。

関数	説明
TE_wait_cursor	カーソルがターミナル・ウィンドウ内の指定した位置に出現するまで待ちます。
TE_wait_silent	クライアント・アプリケーションが指定した秒数の間無動作になるまで待ちます。
TE_wait_sync	システムが X-SYSTEM または入力抑制モードから制御が戻るまで待ちます。
TE_wait_text	文字列が指定した位置に出現するまで待ちます。
TE_wait_sync_transaction	システムが最新の X SYSTEM モードを維持していた時間を記録します。

RTE 仮想ユーザ・スクリプトにおける同期化の詳細については，823 ページの「RTE 同期の概要」を参照してください。

［テスト結果］ウィンドウ・オプションを設定する

仮想ユーザ・スクリプトの実行結果をまとめたレポートは，スクリプトのデバッグ作業に利用できます。VuGen は，仮想ユーザ・スクリプトの実行中にレポートを生成し，スクリプトの実行が完了したらレポートを表示します。

標準設定では，実行が終了すると，VuGen によってテスト結果が生成されて自動的に表示されます。

VuGen によって結果が生成されないようにするには，［**ツール**］>［**一般オプション**］を選択するか，［**表示**］タブを選択して［**スクリプトの実行中にレポートを作成する**］オプションをクリアします。

スクリプトの実行後に結果を開くかどうかを指定するには，［**ツール**］>［**一般オプション**］を選択し，［**再生**］タブを選択して［**再生後**］セクションでビューを選択します。

スクリプトの再生とデバッグを行う

詳細については，131 ページの「仮想ユーザ・スクリプトの再生とデバッグ」を参照してください。

ステップをスクリプトに挿入する方法

次の手順では，各種ステップを仮想ユーザ・スクリプトに追加する方法について説明します。

➤ 164 ページの「思考遅延時間ステップを挿入する」

➤ 164 ページの「デバッグ・メッセージを挿入する」

➤ 164 ページの「挿入エラーと出力メッセージ」

思考遅延時間ステップを挿入する

連続するアクションの合間のユーザの待ち時間を「*思考遅延時間*」といいます。仮想ユーザは，**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できます。仮想ユーザ・スクリプトを記録すると，VuGen によって実際の思考遅延時間が記録され，適切な **lr_think_time** ステートメントが仮想ユーザ・スクリプトに挿入されます。記録された **lr_think_time** ステートメントは後で編集できます。また，手動でも **lr_think_time** ステートメントを仮想ユーザ・スクリプトに追加できます。

思考遅延時間ステップを追加するには，［**挿入**］>［**新規ステップ**］>［**思考遅延時間**］を選択し，必要な思考遅延時間を秒単位で指定します。

デバッグ・メッセージを挿入する

VuGen のユーザ・インタフェースを使って，デバッグ・メッセージまたはエラー・メッセージを追加できます。デバッグ・メッセージの場合には，テキスト・メッセージのレベルを指定できます。対象のメッセージは，指定したレベルがメッセージ・クラスのレベルに一致する場合にだけ発行されます。メッセージ・クラスは，**lr_set_debug_message** を使用して設定します。

デバッグ・メッセージを挿入するには，［**挿入**］>［**新規ステップ**］>［**デバッグ メッセージ**］を選択し，ダイアログ・ボックスを完了します。ユーザ・インタフェースの詳細については，169 ページの「［デバッグ メッセージ］ダイアログ・ボックス」を参照してください。

挿入エラーと出力メッセージ

Web，Winsock，および Oracle NCA など，スクリプトのツリー・ビューの表示が可能なプロトコルの場合，VuGen の中でエラーまたは出力メッセージを追加できます。この関数の一般的な使用法としては，条件文を挿入して，エラー状態が検出されたときにメッセージを発行するという方法があります。

エラー・メッセージまたは出力メッセージを挿入するには，［**挿入**］>［**新規ステップ**］>［**エラー メッセージ**］または［**出力メッセージ**］を選択し，メッセージを入力します。**lr_error_message** 関数または **lr_output_message** 関数がスクリプトの現在の位置に挿入されます。

リファレンス

役に立つ VuGen 関数

このセクションでは，負荷テスト用のスクリプトをデバッグまたは準備するときに，スクリプトに追加できる便利な VuGen 関数について説明します。

仮想ユーザ情報の取得

次の関数を仮想ユーザ・スクリプトに追加して，仮想ユーザ情報を取得できます。

関数	説明
lr_get_attrib_string	コマンド・ライン・パラメータ文字列を返します。
lr_get_host_name	仮想ユーザ・スクリプトを実行しているマシンの名前を返します。
lr_get_master_host_name	コントローラを実行しているマシンの名前を返します。HP Business Service Management で作業する場合は適用対象外です。
lr_whoami	スクリプトを実行している仮想ユーザの名前を返します。HP Business Service Management で作業する場合は適用対象外です。

次の例では，**lr_get_host_name** 関数を使用して，仮想ユーザを実行しているコンピュータの名前を取得しています。

```
my_host = lr_get_host_name( );
```

前述の関数の詳細については，*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。

出力へのメッセージの送信

仮想ユーザ・スクリプトの中でメッセージ・タイプの関数を使用すると，カスタマイズしたエラー・メッセージや通知メッセージを出力やログ・ファイル，およびテスト・レポートのサマリに送信できます。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。LoadRunner 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。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。

HP Business Availability Center で作業をしているときは，メッセージ・タイプの関数を使用して，エラー・メッセージや通知メッセージを Web サイトや Business Process Monitor ログ・ファイルに送信できます。たとえば，Web ベース・アプリケーションの現在の状態を表示するメッセージを挿入できます。

注：トランザクションの中からメッセージを送信することは避けてください。トランザクションの実行時間が長くなり，トランザクションの結果が不正確になることがあります。

次のメッセージ関数を仮想ユーザ・スクリプトの中で使用できます。

関数	説明
lr_debug_message	デバッグ・メッセージを出力ウィンドウまたは Business Process Monitor ログ・ファイルに送ります。
lr_error_message	エラー・メッセージを出力ウィンドウ，テスト結果レポート，または Business Process Monitor ログ・ファイルに送ります。
lr_get_debug_message	現在のメッセージ・クラスを取得します。
lr_log_message	出力メッセージを仮想ユーザ・スクリプトのディレクトリにあるログ・ファイル *output.txt* に直接送信します。この関数は，出力メッセージによって TCP/IP のトラフィックが妨げられるのを防ぐので便利です。
lr_output_message	メッセージを出力ウィンドウ，テスト結果レポート，または Business Process Monitor ログ・ファイルに送ります。

関数	説明
lr_set_debug_message	出力メッセージのメッセージ・クラスを設定します。
lr_vuser_status_message	メッセージを［Controller］の仮想ユーザ・ステータス領域に送ります。HP Business Service Management で作業する場合は適用対象外です。
lr_message	メッセージを，仮想ユーザ・ログと，出力ウィンドウまたは Business Process Monitor ログ・ファイルに送ります。

lr_message 関数，**lr_output_message** 関数，**lr_log_message** 関数の動作は，［実行環境設定］の［ログ］設定内のスクリプトのデバッグ・レベルの影響を受けません。これらの関数は常にメッセージを送信します。

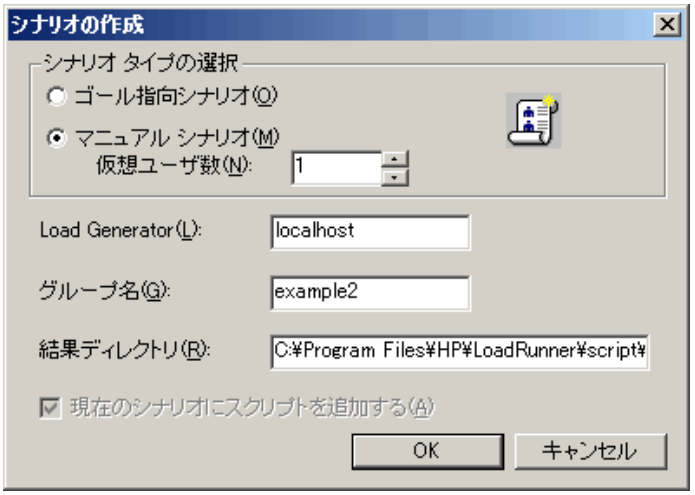
lr_output_message および **lr_error_message** 関数を使用すると，［テスト結果］サマリ・レポートに，意味のあるメッセージを送信することもできます。詳細については，第 6 章，「テスト結果の表示」。を参照してください。

負荷テスト用スクリプトの準備のユーザ・インタフェース

このセクションの内容

［シナリオの作成］ダイアログ・ボックス

このダイアログ・ボックスでは，VuGen 内から基本的な Controller シナリオを作成できます。



利用方法	**［ツール］>［Controller のシナリオを作成］**
関連タスク	155 ページの「VuGen から Controller のシナリオを作成する方法」

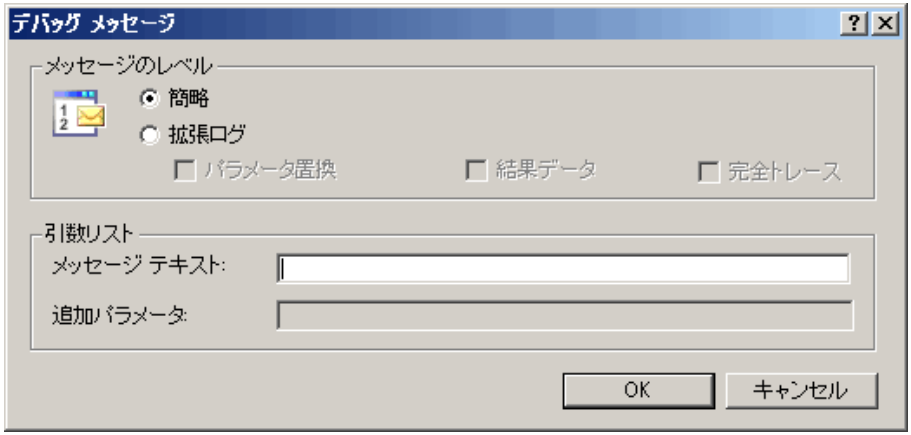
ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
現在のシナリオにスクリプトを追加する	Controller ですでにシナリオを開いていて，このシナリオにスクリプトを追加する場合は，このチェック・ボックスを選択します。チェック・ボックスをクリアすると，LoadRunner によって，指定した数の仮想ユーザを含んだ新しいシナリオが作成されます。
グループ名	マニュアル・シナリオの場合，共通の特性を持つユーザをグループに編成します。仮想ユーザの新しいグループ名を指定します。
Load Generator	シナリオを実行するマシンの名前。

UI 要素	説明
結果ディレクトリ	結果用の場所を入力します。
スクリプト名	ゴール指向シナリオの場合，スクリプト名を指定します。
シナリオ タイプの選択	➤ **[ゴール指向シナリオ]**：LoadRunner は，指定されたゴールに基づいてシナリオを自動的に作成します。 ➤ **[マニュアル シナリオ]**：シナリオは，実行する仮想ユーザの数を指定することで手動で作成されます。

[デバッグ メッセージ] ダイアログ・ボックス

このダイアログ・ボックスでは，デバッグ・メッセージをスクリプトに挿入できます。



利用方法	**[挿入] > [新規ステップ] > [デバッグ メッセージ]**
関連タスク	163 ページの「ステップをスクリプトに挿入する方法」

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
引数リスト	[**メッセージ テキスト**] フィールドにメッセージを入力します。
メッセージのレベル	メッセージのレベル：[**簡略**] または [**拡張ログ**]。[拡張ログ] を選択した場合には，ログに記録する情報の種類を，[**パラメータ置換**]，[**結果データ**]，[**完全トレース**] の中から指定します。

Password Encoder

このダイアログ・ボックスでは，エンコードされたパスワードを生成できます。

利用方法	**[スタート] > [すべてのプログラム] > LoadRunner > [Tools] > [Password Encoder]**
関連タスク	155 ページの「パスワードを暗号化する方法」
関連項目	150 ページの「パスワードのエンコード」

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
コピー	エンコード文字列フィールドから結果をコピーして，パラメータのリストを含むデータ・テーブルに貼り付けます。
エンコード文字列	エンコードされた結果がここに表示されます。
生成	これをクリックすると，エンコードされたパスワードが生成されます。
パスワード	エンコードするパスワードをここに入力します。

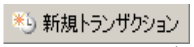
トランザクション リスト

このリストでは，トランザクションを管理できます。

利用方法	**［タスク］表示枠 > ［トランザクション］**
重要情報	このリストはサムネイルを持つスクリプトにのみ適用されます。
関連タスク	➤ 156 ページの「トランザクションを挿入する方法」 ➤ 158 ページの「トランザクションを表示する方法」

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
< サムネイル・リスト >	括弧でラベル付けされてマークされたトランザクションを含むサムネイルが表示されます。

UI 要素	説明
アクション	このドロップダウン・リストを使用して，スクリプト内の関連するセクションを選択します。
トランザクション リスト	➤ **[アクション]**：このドロップダウン・リストを使用して，スクリプト内の関連するセクションを選択します。 ➤ **[トランザクション リスト]**：名前とトランザクション数が表示されます。表示されたトランザクション数がトランザクションの合計数よりも少ない場合（例：2/4），非表示のトランザクションがあります。標準設定では，トランザクション・リストには，スクリプトの主要なステップに対するサーバの応答を測定する，エラーの生じないトランザクションのみが表示されます。主要でないステップ，クライアント側のトランザクション，またはエラーの生じたトランザクションは表示されません。したがって，トランザクション・リストの上部に［**トランザクション（2 / 4）**］のようなキャプションが表示される場合があります。すべてのトランザクションを表示するには，158 ページの「トランザクションを表示する方法」を参照してください。 ➤  ：新しいトランザクションを挿入します。タスクの詳細については，157 ページの「トランザクション・エディタを使用して，トランザクションを挿入する」を参照してください。

6

テスト結果の表示

本章の内容

概念

- テスト結果の概要（174 ページ）
- テスト結果表示のカスタマイズ（175 ページ）
- ［テスト結果］ウィンドウから Application Lifecycle Management への接続（176 ページ）

タスク

- カスタム情報をレポートに送信する方法（177 ページ）
- ［テスト結果］ウィンドウの外観を設定する方法（177 ページ）
- 特定の実行のテスト結果を開く方法（178 ページ）
- テスト結果内のステップを検索する方法（179 ページ）
- 不具合を ALM に送信する方法（179 ページ）

リファレンス

- テスト結果のユーザ・インタフェースの表示（181 ページ）

概念

テスト結果の概要

仮想ユーザ・スクリプトの実行結果をまとめたレポートは，スクリプトのデバッグ作業に利用できます。VuGen は，仮想ユーザ・スクリプトの実行中にレポートを生成し，スクリプトの実行が完了したらレポートを表示します。

［テスト結果］ウィンドウには，次を含むテスト実行のあらゆる側面が表示されます。

- 高レベルの結果概要レポート（テストの成功 / 失敗のステータス）
- すべてのテスト実行に使用されたデータ
- アプリケーション・エラーの発生場所を正確に示す，ステップの展開可能なツリー
- エラーが発生したスクリプト内の具体的な場所
- 特定のステップにおけるアプリケーションの状態の静止画像
- 特定のステップまたはテスト全体のアプリケーションの状態のムービー・クリップ
- テストの各段階で，成功または失敗した各ステップとチェックポイントの詳細な説明

テスト結果表示のカスタマイズ

各結果セットは，単一の **.xml** ファイル（**results.xml**）に保存されます。この **.xml** ファイルには，ディスプレイのそれぞれのテスト結果ノードに関する情報が保存されます。ノードの情報は，［テスト結果］ウィンドウの右上の表示枠に表示される **.htm** ファイルを動的に作成するのに使用されます。

実行結果ツリーの各ノードは **results.xml** ファイルの要素です。また，テスト結果に表示される異なるタイプの情報を表す要素もあります。**.xml** ファイルからテスト結果情報を取得し，XSL を使用して必要な情報をカスタマイズされた形式で表示できます（［テスト結果］ウィンドウから印刷する場合でも，テスト結果を独自にカスタマイズした結果ビューアで表示する場合でも，またはテスト結果を HTML ファイルにエクスポートする場合でも可能）。

XSL には，どのテスト結果情報を表示するか，それをどこでどのように表示，印刷，またはエクスポートするかを示すツールがあります。XSL エディタを使用すると，結果フォルダ内の **.css** ファイルおよび **.xsl** ファイルを変更し，レポートの外観を変更できます（フォント，色など）。

たとえば，**results.xml** ファイルで，ある要素タグにはアクション名が，もう 1 つの要素タグには実行が開始される時間についての情報が含まれているとします。XSL を使用すれば，カスタマイズしたエディタに，アクション名をページの特定の位置に緑色の太字で表示し，時間の情報はまったく表示しないように指示できます。

［テスト結果］ウィンドウから Application Lifecycle Management への接続

［テスト結果］ウィンドウから Application Lifecycle Management へ手動で不具合を送信するには，Application Lifecycle Management に接続する必要があります。

接続プロセスには次の 2 つの段階があります。まず，ローカルまたはリモートの Application Lifecycle Management サーバに接続します。このサーバによって，テスト結果と Application Lifecycle Management プロジェクトの間の接続が処理されます。

次に，ログインしてアクセスするプロジェクトを選択します。プロジェクトには，テストと，テスト対象アプリケーションの実行セッション情報が格納されます。Application Lifecycle Management プロジェクトはパスワードで保護されているため，ユーザ名とパスワードを指定する必要があります。

ALM プロジェクトへの接続の詳細については，253 ページの「ALM プロジェクトのスクリプトを使って作業する方法」を参照してください。

タスク

カスタム情報をレポートに送信する方法

Web サービス仮想ユーザのために，レポートに自動的に送信される情報に加えて，メッセージ関数 **lr_output_message** または **lr_error_message** を使用すると，レポートに情報を送信できます。

タスクの詳細については，163 ページの「ステップをスクリプトに挿入する方法」を参照してください。

［テスト結果］ウィンドウの外観を設定する方法

標準設定では，［テスト結果］ウィンドウは，Microsoft Office 2003 テーマを使用する QuickTest ウィンドウと同じルック・アンド・フィールを採用しています。必要に応じて，［テスト結果］ウィンドウのルック・アンド・フィールを変更できます。

これらの設定を変更するには，［**表示**］>［**ウィンドウのテーマ**］を選択し，テーマを選択します。

注：［テスト結果］ウィンドウに Microsoft Windows XP のテーマを適用するには，ご利用のコンピュータが Windows XP のテーマを使用するように設定されている必要があります。

特定の実行のテスト結果を開く方法

このタスクでは，特定の実行のテスト結果ウィンドウを開く方法について説明します。

1 ［テスト結果］ウィンドウ内で［**ファイル**］>［**開く**］を選択します。

2 そのファイルのテスト結果を表示するには，スクリプト・ファイルを選択し，目的のテスト結果ファイルを選択します。標準設定では，テストの結果ファイルは **< スクリプト >¥< 結果名 >.xml** の形式で保存されます。スクリプトが Application Lifecycle Management に保存されている場合は，次を参照してください。

3 結果セットを選択し，［**開く**］をクリックします。

注：以前のバージョンの結果ファイルは，**.qtp** という拡張子付きで保存されます。標準設定では，［結果ファイルを選択］ダイアログ・ボックスでは拡張子が **.xml** の結果ファイルのみが表示されます。［結果ファイルを選択］ダイアログ・ボックスで拡張子が **.qtp** の結果ファイルが表示されるようにするには，［**ファイルの種類**］ボックスで「**テスト結果（*.qtp）**」を選択します。

Application Lifecycle Management に保存されているスクリプトを選択するには，次の手順で行います。

1 ［**Application Lifecycle Management への接続**］ボタンをクリックし，Application Lifecycle Management プロジェクトに接続します。

2 ［テスト結果を開く］ダイアログ・ボックスで，QuickTest テストの結果ファイルが含まれているフォルダのパスを入力するか，参照ボタンをクリックして［ALM プロジェクトからテストを開く］ダイアログ・ボックスを開きます。

3 ［**Test Type**］リストで［**DB 仮想ユーザ**］を選択します。

4 テスト結果を表示するスクリプトを選択し，［**OK**］をクリックします。

5 ［テスト結果を開く］ダイアログ・ボックスで，表示するテスト結果セットを強調表示し，［**開く**］をクリックします。［テスト結果］ウィンドウに選択したテスト結果が表示されます。

テスト結果内のステップを検索する方法

このタスクでは，特定のタイプのステップのテスト結果を検索する方法について説明します。

1 ［テスト結果］ウィンドウ内で［**ツール**］>［**検索**］を選択します。

2 検索するステップのタイプを選択します。複数のオプションを選択できます。

3 ［**上へ**］または［**下へ**］を選択し，検索方向を指定します。

4 ［**次を検索**］を選択して，選択したステップのテキストが次に出現する箇所を検索します。

不具合を ALM に送信する方法

結果の表示中は，検出された不具合を，［テスト結果］ウィンドウから Application Lifecycle Management プロジェクトに直接送信できます。

不具合を Application Lifecycle Management に直接送信するには，次の手順で行います。

1 Application Lifecycle Management クライアントがコンピュータにインストールされていることを確認します（ブラウザに Application Lifecycle Management サーバの URL を入力し，ログイン画面が表示されることを確認します）。

2 ［**ツール**］>［**Application Lifecycle Management への接続**］を選択するか，［**Application Lifecycle Management への接続**］ボタンをクリックして Application Lifecycle Management プロジェクトに接続します。プロジェクトへの接続の詳細については，251 ページの「Application Lifecycle Management を使った作業」を参照してください。

3 ［**ツール**］>［**不具合の追加**］を選択するか，［**不具合の追加**］ボタンをクリックして，指定した Application Lifecycle Management プロジェクトの［不具合の追加］ダイアログ・ボックスを開きます。

4 必要に応じて不具合の情報を変更できます。説明に含まれる基本的な情報は次のとおりです。

Operating system :	Windows 2000
Test path :	[QualityCenter] Components\YE\ComponentWithDefect

5 ［**送信**］をクリックし，Application Lifecycle Management プロジェクトに不具合情報を追加します。

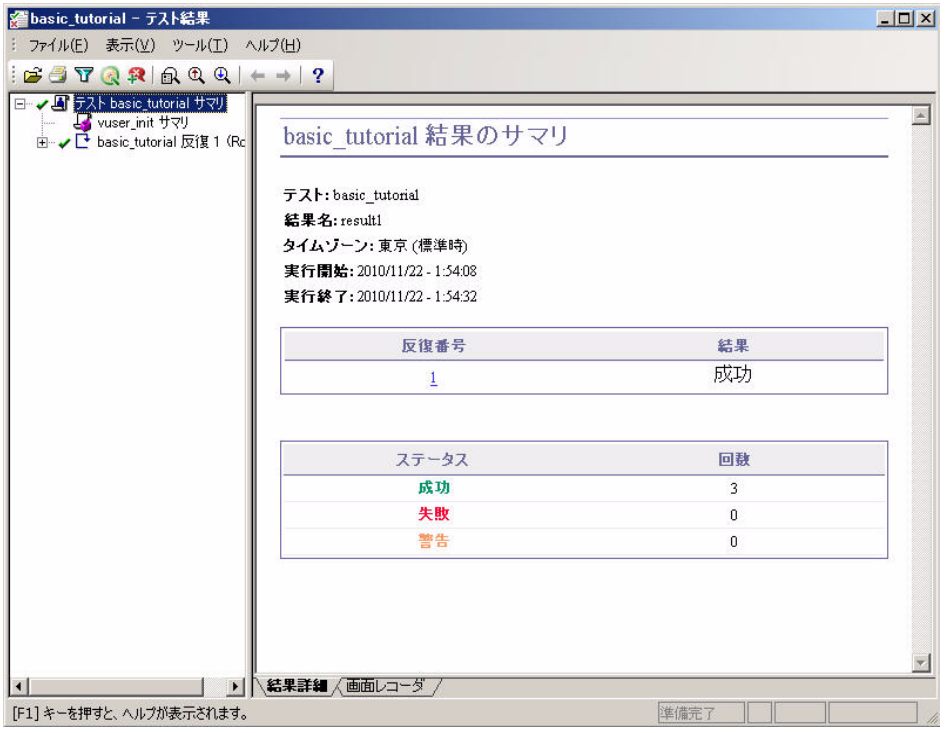
リファレンス

テスト結果のユーザ・インタフェースの表示

このセクションの内容

［テスト結果］ウィンドウ

このウィンドウには，スクリプト実行の結果をまとめたレポートが表示されます。



利用方法	**スクリプトの実行後に自動的に開きます。**
重要情報	➤ 表示設定は，［**ツール**］>［**一般オプション**］>［**再生**］>［**再生後の表示**］で設定できます。 ➤［テスト結果］ウィンドウには，300 レベルまでの結果をツリー階層に表示できます。結果に 300 レベル以上の入れ子が含まれている場合は，**results.xml** ファイルを手動で開くことによって，レポート全体を表示できます。

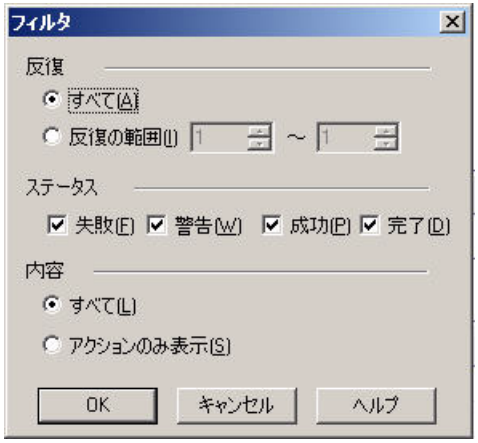
ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
レポート ツリー	ウィンドウの左側の表示枠にテスト結果が視覚的に表示されたものです。ツリーに表示される詳細のレベルを変更するには，実行結果ツリーの分岐を折りたたむか，展開します。 ステップの横にあるアイコンは次の情報を示します。 成功したステップを示します。 失敗したステップを示します。 警告を示します。そのステップは成功しなかったが，テストが失敗する原因にはならなかったことを意味します。 予期せず失敗したステップを示します。 ［**表示**］メニューで，または * をクリックして，すべてのノードを展開および折りたたむことができます。
［結果詳細］表示枠（結果サマリ）	レポート・ツリーで選択した部分に応じて変化するテスト実行の詳細が表示されます。ツリーの最上位ノードを選択すると，［結果詳細］タブに，結果のサマリが表示されます。ツリー内の分岐またはステップを選択すると，［結果詳細］タブに，該当するステップの詳細が表示されます。また，［結果詳細］タブには，強調したステップに関するアプリケーションの静止画像を表示することもできます。 ➤ チェックポイントを含む反復，アクション，およびステップは，［テスト結果］ウィンドウの右側に「**成功**」または「**失敗**」と表示され，ツリー・ウィンドウではアイコンで見分けることができます。 ➤ チェックポイントは含まれないが，実行が成功した反復，アクション，ステップは，［テスト結果］ウィンドウの右側に「**完了**」と表示されます。
［画面レコード］タブ	テスト結果に関連付けられているムービーが表示されます。テスト結果に関連付けられているムービーがない場合は，［画面レコーダ］タブに，「**結果に関連付けられたムービーはありません。**」というメッセージが表示されます。

UI 要素	説明
	特定の実行のテスト結果が開かれます。詳細については，178 ページの「特定の実行のテスト結果を開く方法」を参照してください。
	［印刷］ダイアログ・ボックスが開き，テスト結果を印刷できます。詳細については，186 ページの「［印刷］ダイアログ・ボックス」を参照してください。
	［フィルタ］ダイアログ・ボックスが開き，テスト結果をフィルタできます。詳細については，185 ページの「［フィルタ］ダイアログ・ボックス」を参照してください。
	［文字列検索］ダイアログ・ボックスが開き，特定のタイプのステップのテスト結果を検索できます。詳細については，179 ページの「テスト結果内のステップを検索する方法」を参照してください。
	［文字列検索］ダイアログ・ボックスの条件に一致する前のステップのテスト結果が検索されます。
	［文字列検索］ダイアログ・ボックスで指定した条件に一致する次のステップのテスト結果が検索されます。
	前のノードが選択されます。
	次のノードが選択されます。
	製品情報が開きます。

［フィルタ］ダイアログ・ボックス

このページでは，テスト結果ウィンドウのテスト結果をフィルタできます。



利用方法	**［テスト結果］ウィンドウ > ［表示］ > ［フィルタ］**

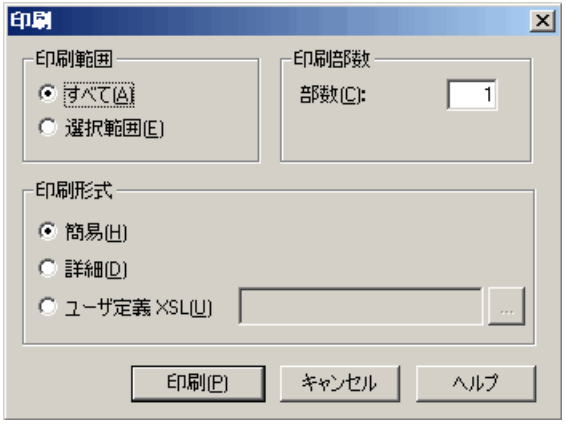
ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
反復	➤ **［すべて］**：すべての反復のテスト結果が表示されます。 ➤ **［反復の範囲 X ～ Y］**：指定した範囲のテスト反復のテスト結果が表示されます。

UI 要素	説明
ステータス	➤ **[失敗]**：失敗したステップの結果が表示されます。 ➤ **[警告]**：ステータスが**警告**のステップ（成功はしなかったがスクリプトが失敗する原因にはならなかったステップ）に関する結果が表示されます。 ➤ **[成功]**：成功したステップの結果が表示されます。 ➤ **[完了]**：ステータスが**完了**のステップ（ステップの実行に成功したが，成功，失敗，警告のステータスを受け取らなかったステップ）に関する結果が表示されます。
内容	➤ **[すべて]**：テストのすべてのノードからすべてのステップが表示されます。 ➤ **[アクションのみ表示]**：テスト内のアクション・ノードが表示されます（アクション・ノードの特定のステップではありません）。

［印刷］ダイアログ・ボックス

このダイアログ・ボックスでテスト結果を印刷できます。



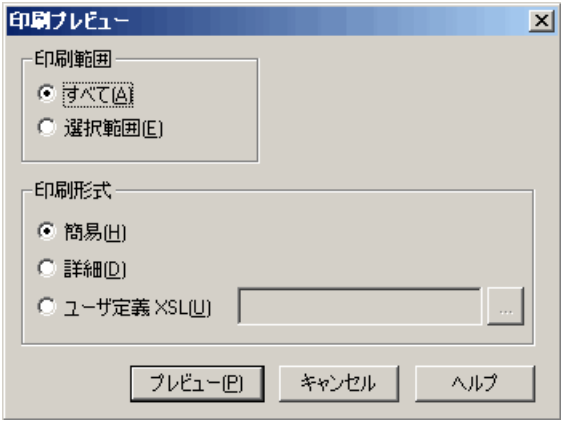
利用方法	［テスト結果］ウィンドウ > ［ファイル］ > ［印刷］

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
印刷範囲	➤ **[すべて]**：スクリプト全体の結果を印刷します。 ➤ **[選択範囲]**：実行結果ツリー内で選択した分岐の結果を印刷します。
印刷部数	印刷する部数。
印刷形式	➤ **[簡易]**：実行結果ツリーの各項目のサマリ行（使用可能な場合）を印刷します。このオプションは，印刷範囲を［**すべて**］に設定している場合にのみ使用できます。 ➤ **[詳細]**：実行結果ツリーの各項目または選択した分岐に対して利用できるすべての情報を印刷します。 ➤ **[ユーザ定義 XSL]**：カスタマイズした **.xsl** ファイルの参照と選択ができます。印刷するレポートに含める情報やその表示形式を指定するユーザ定義の **.xsl** ファイルを作成できます。詳細については，175 ページの「テスト結果表示のカスタマイズ」を参照してください。

［印刷プレビュー］ダイアログ・ボックス

このダイアログ・ボックスでテスト結果の印刷プレビューを表示できます。



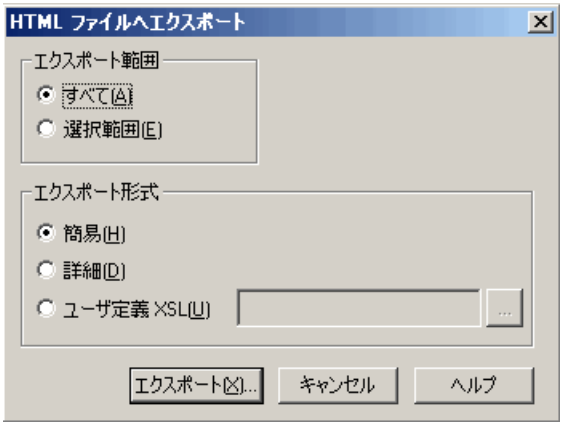
利用方法	［テスト結果］ウィンドウ > ［**ファイル**］> ［**印刷プレビュー**］
重要情報	プレビューに表示されない情報がある場合は（たとえば，チェックポイント名が長すぎてディスプレイに表示されないなど），［印刷プレビュー］ウィンドウの［**ページ設定**］ボタンをクリックして，ページの向きを［**縦**］から［**横**］に変更します。

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
印刷範囲	➤ **［すべて］**：スクリプト全体の結果を印刷します。 ➤ **［選択範囲］**：実行結果ツリー内で選択した分岐の結果を印刷します。
印刷形式	➤ **［簡易］**：実行結果ツリーの各項目のサマリ行（使用可能な場合）を印刷します。このオプションは，印刷範囲を［**すべて**］に設定している場合にのみ使用できます。 ➤ **［詳細］**：実行結果ツリーの各項目または選択した分岐に対して利用できるすべての情報を印刷します。 ➤ **［ユーザ定義 XSL］**：カスタマイズした **.xsl** ファイルの参照と選択ができます。印刷するレポートに含める情報やその表示形式を指定するユーザ定義の **.xsl** ファイルを作成できます。詳細については，175 ページの「テスト結果表示のカスタマイズ」を参照してください。

［HTML ファイルへエクスポート］ダイアログ・ボックス

このダイアログ・ボックスでテスト結果を HTML ファイルへエクスポートできます。テスト結果ビューア環境を利用できない場合でも，これにより結果を表示できます。たとえば，結果を含むファイルを電子メールでサードパーティに送信できます。エクスポートするレポートのタイプを選択できます。また，ユーザ定義のレポートの作成やエクスポートも行えます。



利用方法	［テスト結果］ウィンドウ > **［ファイル］** > **［HTML ファイルへエクスポート］**

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
エクスポート範囲	➤ **[すべて]**：スクリプト全体の結果がエクスポートされます。 ➤ **[選択範囲]**：結果ツリーで選択した分岐の結果情報がエクスポートされます。
エクスポート形式	➤ **[簡易]**：実行結果ツリーの各項目のサマリ行（使用可能な場合）を印刷します。このオプションは，印刷範囲を［**すべて**］に設定している場合にのみ使用できます。 ➤ **[詳細]**：実行結果ツリーの各項目または選択した分岐に対して利用できるすべての情報を印刷します。 ➤ **[ユーザ定義 XSL]**：カスタマイズした **.xsl** ファイルの参照と選択ができます。印刷するレポートに含める情報やその表示形式を指定するユーザ定義の **.xsl** ファイルを作成できます。詳細については，175 ページの「テスト結果表示のカスタマイズ」を参照してください。

7

相関

本章の内容

概念

タスク

リファレンス

概念

相関の概要

相関は，記録されたスクリプトに動的な値（セッション ID など）が含まれていて，再生できない場合に使用されます。これを解決するには，動的な値を変数にします。これにより，スクリプトで再生を正常に行うことができます。

たとえば，多くのアプリケーションや Web サイトでは現在の日時に基づいてセッションが識別されます。スクリプトを再生しようとすると，現在の時刻が記録された時刻と異なるので失敗します。データを相関させると，動的なデータを保持し，これをシナリオ実行の全体を通して使用できます。

相関が作成されると，VuGen によって動的な値をパラメータに抽出する関数が追加されます。元の値の適切な出現箇所はパラメータで置き換えられます。

相関対パラメータ化

パラメータ化は，スクリプトをより現実的なものにするために値を変数に変換する場合に使用されます。たとえば，ある Web サイトのフォームに記入している場合，特定のフィールドの入力値を変化させる必要がある場合があります。

相関は，記録されたスクリプトに動的な値（セッション ID など）が含まれていて，再生できない場合に使用されます。これを解決するには，動的な値を変数にします。これにより，スクリプトで再生を正常に行うことができます。

自動相関

相関は，特定の条件で記録中に自動的に実行できます。記録する前に相関させる必要がある値がわかっている場合は，記録中にそれらの値を自動的に相関させる相関ルールを作成できます。また，VuGen にはサポートされているアプリケーション・サーバ用にあらかじめ定義された相関ルールがいくつか用意されています。ルールは，［記録オプション］ダイアログ・ボックスの［HTTP 相関］ノードで有効または無効にできます。ユーザ・インタフェースの詳細については，373 ページの「HTTP の［相関］ノード」を参照してください。

相関させる値の決定

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 / 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の資格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。

Java スクリプトの相関

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。

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 function: 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 function: 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 function: 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 function: 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);
  }
:
}
```

Java スクリプトの相関 - シリアル化

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 function: 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，仮想ユーザ・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 function: 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，406 ページの「［記録のプロパティ］の［シリアル化オプション］ノード」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，［未展開のシリアル化オブジェクト］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 function: 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。標準設定の区切り文字は「#」です。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 function: 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 function: 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。

Winsock スクリプトの相関

VuGen には，仮想ユーザ・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 仮想ユーザ・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。

VuGen では，**lrs_save_param** および **lrs_save_searched_string** 関数の相関 Winsock スクリプトが使用されます。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，受信バッファ内で相関させ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，239 ページの「スクリプトを相関させる方法 - Winsock（ツリー・ビュー)」を参照してください。

Wdiff ユーティリティ

Wdiff ユーティリティを使用することにより，記録したスクリプトと結果を比較し，相関させる値を判断することができます。

WDiff ユーティリティを効果的に使うには，同じ操作を 2 度記録し，スクリプト（または Tuxedo，WinSock，Jolt の場合にはデータ・ファイル）を比較します。WDiff に，違いが黄色で示されます。ただし，すべての相違部分が相関すべき値というわけではありません。たとえば，実行の時刻を示す受信バッファは相関を必要とはしません。

タスクの詳細については，207 ページの「相関が必要な値を検索する方法」を参照してください。

保存したパラメータの変更

パラメータに値を保存したら，実際にスクリプトの中で使う前に，パラメータを変更する必要がある場合があります。パラメータを使って算術演算を行う場合は，C 関数の **atoi** または **atol** を使って値を文字列から整数に変更する必要があります。値を整数に変換したら，スクリプトの中で新しい変数を使用するには，その整数をもう一度文字列に戻す必要があります。

次の WinSock の例では，オフセット 67 の位置にあるデータをパラメータ **param1** に保存しています。次に，**atol** を使って文字列を long int 型に変換します。**param1** の値に 1 を加算した後で，sprintf を使って値を文字列に戻し，新しい文字列 **new_param1** として保存します。パラメータの値は **lr_output_message** を使って表示しています。この新しい値は，以降でスクリプトの中で使用することができます。

```
lrs_receive("socket2", "buf47", LrsLastArg);lrs_save_param("socket2",
                NULL, "param1", 67, 5);
lr_output_message ("param1: %s", lr_eval_string("<param1>"));
sprintf(new_param1, "value=%ld", atol(lr_eval_string("<param1>")) + 1);
lr_output_message("ID Number:"%s" lr_eval_string("new_param1"));
```

タスク

スクリプトを相関させる方法 - Web（HTTP/HTML）

このタスクでは，Web（HTTP/HTML）プロトコル・スクリプトを相関させるさまざまな方法について説明します。

このタスクでは，次の手順を実行します。

- 206 ページの「相関を検索」
- 207 ページの「手作業による相関」
- 207 ページの「自動相関ルール」

相関を検索

1 ［**仮想ユーザ**］>［**相関を検索**］を選択します。これにより，Web ステップの記録時と再生時のスナップショット・データで不一致がないかどうかスクリプト全体が検索されます。この検索は，すべてのサーバ応答（ヘッダを含む）で行われます。

2 不一致が検出されると，出力ウィンドウの［**相関結果**］タブに表示されます。ユーザ・インタフェースの詳細については，88 ページの「[出力ウィンドウ］-［相関結果］タブ」を参照してください。

3 どの値を相関させるかを判断するには，194 ページの「相関させる値の決定」を参照してください。

4 不一致が相関されると，VuGen によって **web_reg_save_param_*** が追加され，スクリプトのコメント内に元の値が保存されます。Web ステップ内の元の値の適切な出現箇所がパラメータで置き換えられます。

5 スクリプトを再生します。相関によるエラーがほかにもある場合は，この手順を繰り返してエラーを解決します。

相関の検索で相関によるエラーがすべて解決されない場合は，次の手作業による相関手順を使用して解決を試みてください。

手作業による相関

相関の検索でスクリプトの相関によるエラーがすべて解決されない場合は，次の手順を実行して手作業でスクリプトを相関させることができます。

1 **手作業による相関が必要な値を検索します。**相関が必要な値を手作業で検索する方法は複数あります。詳細については，207 ページの「相関が必要な値を検索する方法」を参照してください。

2 **値を相関させます。**

次のいずれかの方法を選択します。

- **スナップショットから相関させます。**相関させる値を強調表示して右クリックし，［**相関の作成**］を選択します。
- **相関関数を手作業で追加します。**関連する相関関数をスクリプトに手作業で挿入します。詳細については，209 ページの「Web スクリプトを手作業で相関させる方法」を参照してください。

自動相関ルール

相関ルールを定義することで特定の文字列を自動的に相関させるように VuGen を設定できます。詳細については，373 ページの「HTTP の［相関］ノード」を参照してください。相関ルールが適用されると，VuGen によって **web_reg_save_param_*** 関数が追加されます。

相関が必要な値を検索する方法

次の手順では，相関が必要な値を検索するさまざまな方法について説明します。

- 207 ページの「スクリプトの比較による検索」
- 208 ページの「再生ログの検索」

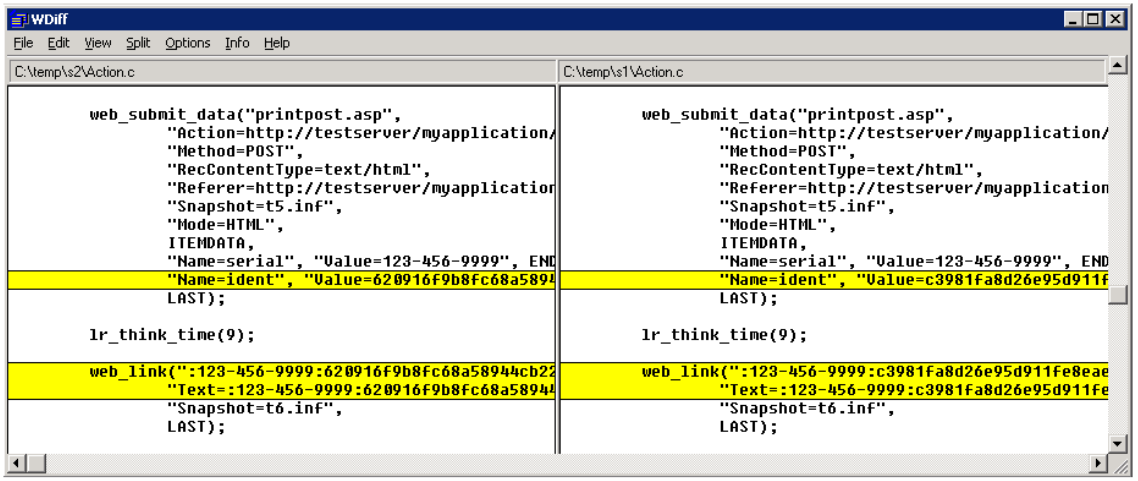
スクリプトの比較による検索

1 スクリプトを記録し，保存します。

2 新しいスクリプトを作成し，まったく同じ操作を記録します。スクリプトを保存します。

3 スクリプトを比較するには，［**ツール**］>［**スクリプトと比較**］を選択します。詳細については，55 ページの「スクリプトを横に並べて比較する方法」を参照してください。

4 スクリプト内の差異が強調表示されます。差異を確認して，相関が必要なものを判断します。



注：*WDiff* が標準設定のユーティリティですが，ユーザ定義の比較ツールを指定することもできます。詳細については，55 ページの「スクリプトを横に並べて比較する方法」を参照してください。

再生ログの検索

1 スクリプト・ビューで，ハッシュ文字列，ランダム文字列，セッション ID などの相関が必要な可能性がある文字列を検索します。

2 生成ログで，文字列の初出を検索します（つまりサーバからの応答です）。

3 同じ応答について拡張再生ログを検索します。この応答に同じ境界内で元の疑わしい文字列と異なる文字列が含まれているかどうかを確認します。含まれている場合，この文字列には相関が必要です。

Web スクリプトを手作業で相関させる方法

このタスクでは，コードを変更することによって，手作業で Web スクリプトを相関させる方法について説明します。

このタスクでは，次の手順を実行します。

- 209 ページの「文字列とその詳細を特定する」
- 210 ページの「web_reg_save_param_* 関数を追加する」
- 211 ページの「データをパラメータで置き換える」

1 文字列とその詳細を特定する

動的データを含むステートメントと，データの場所を示すパターンを探します。これらのパターンは，境界または xpath です。

a 境界を使用したパターンの特定

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。

- HTTP 応答内の動的データの場所を分析します。
- 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。
- 右と左の境界は，文字列をより適切に特定するために，できる限り一意であることが必要です。
- **web_reg_save_param_ex** 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_ex** 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。

これらの文字数制限は，左または右の境界が空の場合は適用されません。

b xpath を使用したパターンの特定

スナップショット表示枠を使用して，目的の文字列の xpath を手作業で検索します。

標準設定では，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。

これらの文字数制限は，左または右の境界が空の場合は適用されません。

2 web_reg_save_param_* 関数を追加する

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_ex** または **web_reg_save_param_xpath** 関数を追加します。

a web_reg_save_param_ex

この関数では，左境界，続いて文字列および右境界について Web ステップ内のサーバ応答が検索され，関数の引数で指定されたパラメータに文字列が保存されます。指定された数の出現箇所が見つかると，**web_reg_save_param_ex** はそれ以上応答を検索しません。詳細については，*HP LoadRunner オンライン関数リファレンス*を参照してください。

b web_reg_save_param_xpath

この関数では，指定した xpath について Web ステップ内のサーバ応答が検索されます。指定した xpath にある文字列は，関数の引数で指定されたパラメータに保存されます。詳細については，*HP LoadRunner オンライン関数リファレンス*を参照してください。

3 データをパラメータで置き換える

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。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。

スクリプトを相関させる方法 - Oracle NCA

次の手順では，相関が必要な可能性がある Oracle NCA スクリプトのさまざまな項目について説明します。

- 211 ページの「ロード・バランシングのためにステートメントを相関させる」
- 213 ページの「icx_ticket 変数を相関させる」
- 214 ページの「JServSessionIdroot 値を相関させる」

ロード・バランシングのためにステートメントを相関させる

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，仮想ユーザはテスト実行時に，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}");
```

icx_ticket 変数を相関させる

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&RESP_APP=AR&RES
P_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 値を相関させる

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);
```

スクリプトを相関させる方法 - データベース・プロトコル

次の手順では，データベース・プロトコルのいずれかを使用してスクリプトを相関させる方法について説明します。

➤ 215 ページの「相関が必要なステートメントを判断する」

➤ 217 ページの「既知の値を相関させる」

相関が必要なステートメントを判断する

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。

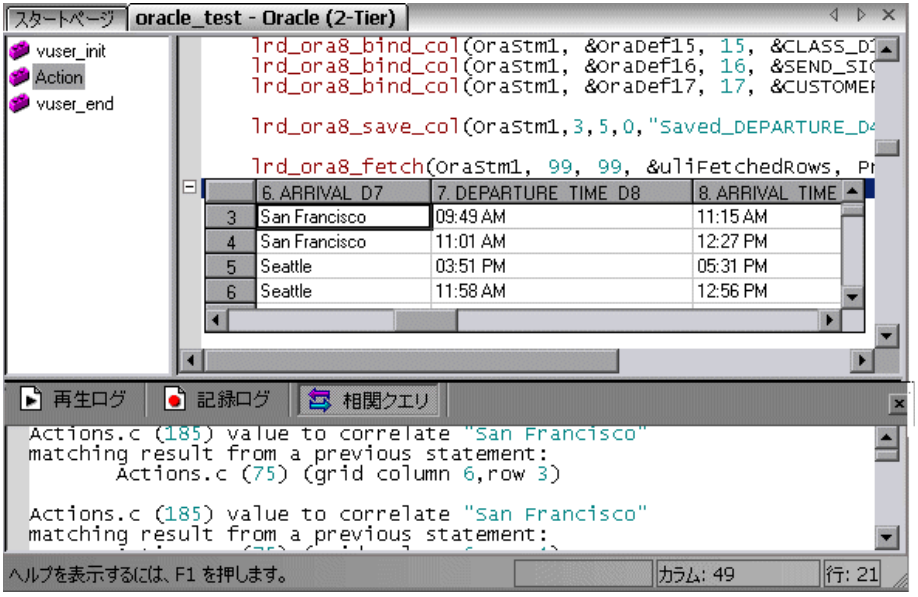
1 出力ウィンドウを開きます。

［**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。

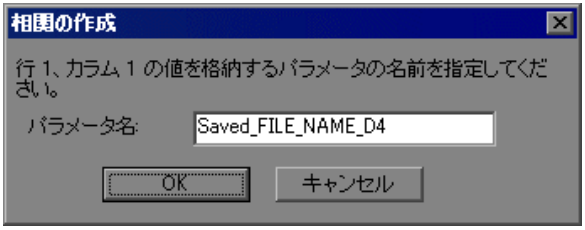
VuGen によってスクリプト全体を対象とする検索が行われ，相関候補の値がすべて［**相関クエリ**］タブに表示されます。

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。

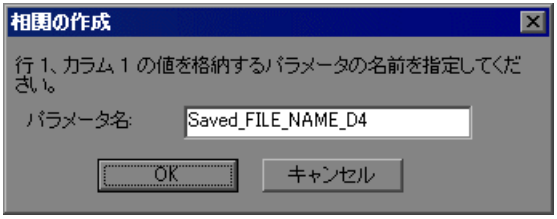
既知の値を相関させる

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** および **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。

スクリプトを相関させる方法 - Microsoft .NET

このタスクでは，Microsoft .NET プロトコル・スクリプトを相関させる方法について説明します。

このタスクでは，次の手順を実行します。

➤ 219 ページの「ADO.net 環境を使用してスクリプトを相関させる」

➤ 220 ページの「出力パラメータと相関させる」

ADO.net 環境を使用してスクリプトを相関させる

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。

	stm_name	stm_dataver	stm_password1	stm_password2	stm_timestamp
1	AdoNetDB	3.30	<Null>	<Null>	<Null>

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 によって各データ・セットの前に **lr.save_string** 関数が追加されます。次に例を示します。

```
lr.save_string("MyCustomerID",
CustomerAndOrdersDataSet_3.Tables["Customers"].Rows[0]["CompanyName"].ToS
tring());
```

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。

出力パラメータと相関させる

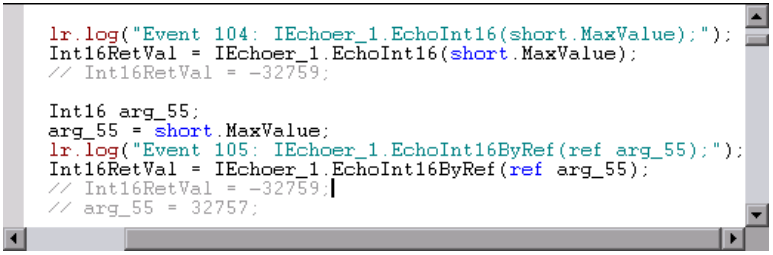
プリミティブな値については，出力パラメータ値を含んだスクリプトを生成し，出力パラメータを調べて相関させる必要があります。

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，*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。

スクリプトを相関させる方法 - Flex および AMF プロトコル

このタスクでは，flex および AMF プロトコル・スクリプトを相関させる方法について説明します。

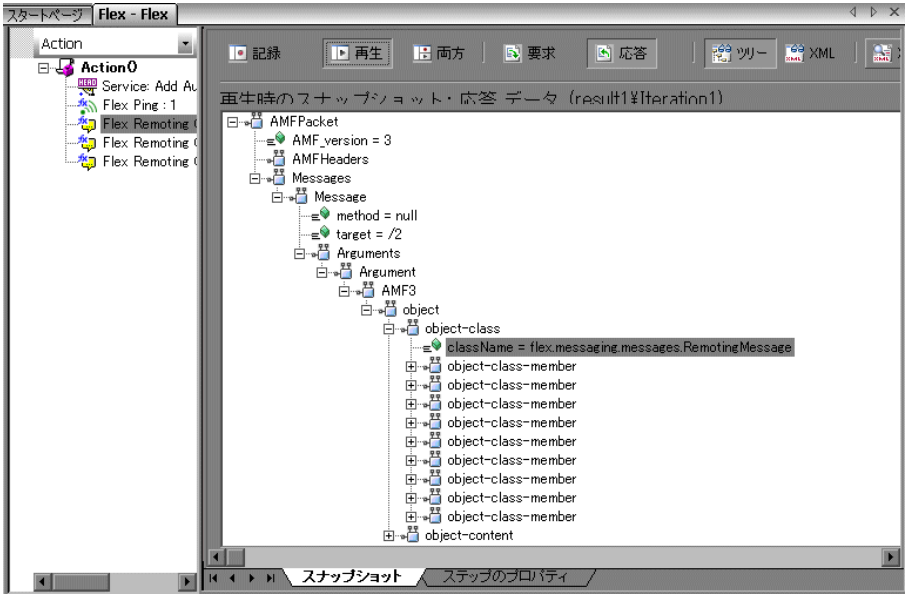
1 動的な値が原因で失敗し，相関が必要となったスクリプト内のステップを探します。

［再生ログ］を利用して問題のステップを探します。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，仮想ユーザ・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。

```
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。



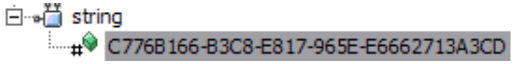
3 サーバ応答全体をパラメータに保存します。

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（または AMF）Call Properties］ダイアログ・ボックスで，［**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。

- ［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。



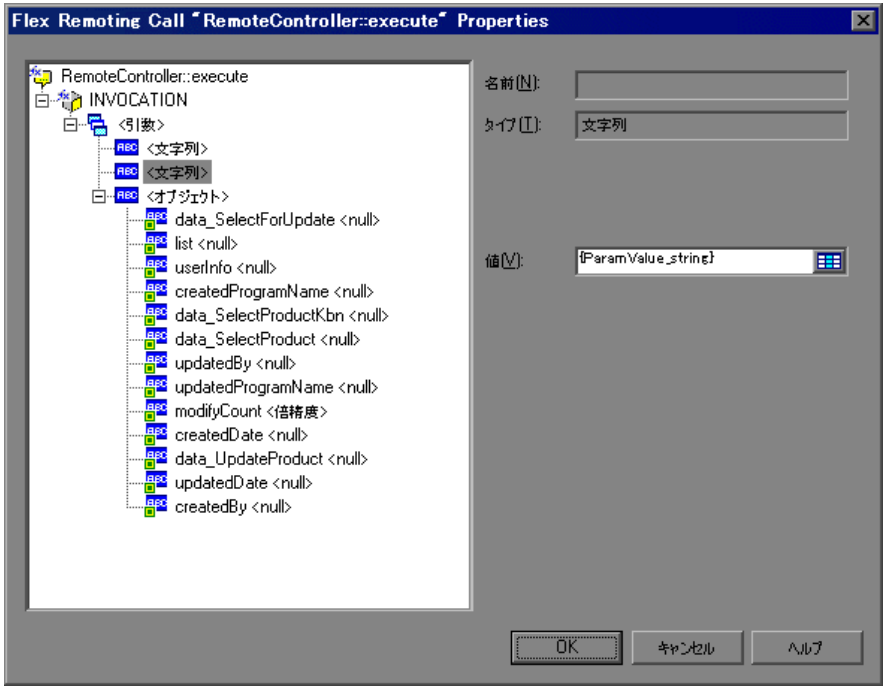
- ［XML パラメータのプロパティ］ダイアログの中で，**Name** にパラメータ名を入力します。以降のステップでは，この名前を使用することになります。
- ［**OK**］をクリックします。これで，スクリプトに **lr_xml_get_values** という新しい関数が追加されます。

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。

スクリプトを相関させる方法 - Siebel プロトコル

次の手順では，Siebel Web 仮想ユーザ・スクリプトを相関させる方法について説明します。

➤ 224 ページの「相関ライブラリ」

➤ 224 ページの「相関ルール」

➤ 230 ページの「SWECount パラメータを相関させる」

➤ 230 ページの「ROWID パラメータを相関させる」

➤ 231 ページの「SWET（タイムスタンプ）パラメータを相関させる」

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** 仮想ユーザ・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**］>［**HTTP プロパティ**］>［**詳細**］ノードで UTF-8 サポートを有効にします。
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，［**上書き**］をクリックします。

   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。

相関ルール

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，Siebel サーバの文字列に固有の境界条件を示します。

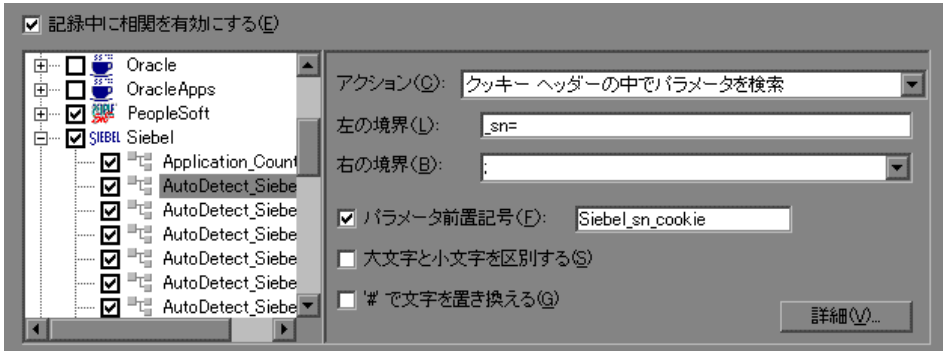
VuGen は，境界条件を使って一致する候補を検出すると，境界と境界の間にある値をパラメータに保存します。保存される値は，単純な変数または public 関数です。

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。

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}&S
WEC={Siebel_SWECount}&SWEFrame=top._sweclient&SWECS=true",
"TargetFrame=",
"Resource=0",
"RecContentType=text/html",
"Referer=http://cannon.hplab.com/callcenter_enu/
start.swe?SWECmd=GetCachedFrame&_sn={Siebel_sn_cookie6}&SWEC={Siebel_S
WECount}&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 によって **web_reg_save_param** に新規パラメータ **AutoCorrelationFunction** が生成されます。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，228 ページの「SWEC 相関」を参照してください。

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-mOTFXHWBAAGb5Xzv9Ls2Z45QvxG
QnOnPVtX6vnfUU_&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-mOTFXHWBAAGb5Xzv9Ls2Z45QvxGQnOnPVtX6vnf
UU_&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 パラメータを相関させる

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。

ROWID パラメータを相関させる

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。

SWET（タイムスタンプ）パラメータを相関させる

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。

スクリプトを相関させる方法 - COM プロトコル

次の手順では，COM プロトコル・スクリプトを相関させる方法について説明します。

➤ 231 ページの「相関が必要な値を見つける」

➤ 232 ページの「既知の値を相関させる」

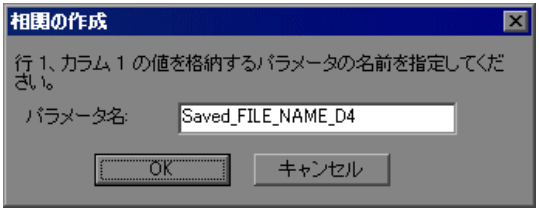
相関が必要な値を見つける

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**］タブに表示されます。

3 値を相関させます。［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。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。

既知の値を相関させる

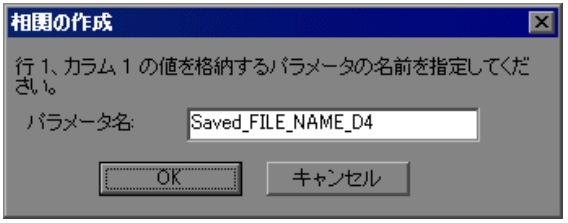
1 相関させる引数を見つけ（通常は **lrc_variant_** ステートメント内にあります），値を選択します（引用符は除きます）。

2 ［**仮想ユーザ**］＞［**相関を検索（カーソル位置）**］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］タブに一覧表示されます。

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。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。

スクリプトを相関させる方法 - Tuxedo プロトコル

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.** 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.** 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.** 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，*オンライン関数リファレンス*を参照してください。

➤ 234 ページの「相関が必要な値を判断する」

➤ 235 ページの「FML および FML32 バッファに対して相関する」

➤ 236 ページの「バッファの場所に基づいて相関させる」

➤ 238 ページの「区切り文字に基づいて相関させる」

相関が必要な値を判断する

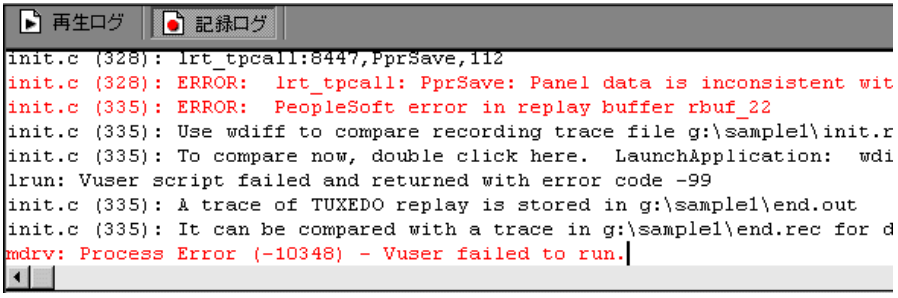
CARRAY バッファで作業を行っている場合，VuGen は Wdiff ユーティリティを使って比較できるログ・ファイルを生成します（記録中であれば拡張子は **.rec** に，再生中であれば拡張子は **.out** になります）。記録ログと再生ログの相違を調べて，CARRAY バッファのどの部分を相関させる必要があるか判断できます。

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。

詳細については，205 ページの「Wdiff ユーティリティ」を参照してください。

FML および FML32 バッファに対して相関する

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。

例：

lrt_save_fld_val (fbfr, "name", occurrence, "param_name");

2 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。

例：

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");
```

バッファの場所に基づいて相関させる

このタスクでは，**lrt_save_parm** 関数を使用して Texedo スクリプトの文字列を相関させる方法について説明します。この関数では，バッファ内の文字列の場所に基づいて相関が作成されます。

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。

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（[スクリプト ビュー] に標準で表示される）の［アクション］表示枠で **replay.vdf** 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。

例：

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}"));
```

区切り文字に基づいて相関させる

このタスクでは，**lrt_save_searched_string** 関数を使用して Texedo スクリプトの文字列を相関させる方法について説明します。この関数では，バッファ内の区切り文字の場所に基づいて相関が作成されます（例：最初の { の直後の文字列を相関させる）。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，この関数は PeopleSoft スクリプトに使用することをお勧めします。

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。

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（[スクリプト ビュー] に標準で表示される）の［アクション］表示枠で **replay.vdf** 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。

例：

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，バッファのヘッダー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");
```

スクリプトを相関させる方法 - Winsock（ツリー・ビュー）

このタスクでは，ツリー・ビューから Winsock スクリプト内に相関を作成する方法について説明します。

このタスクでは，次の手順を実行します。

- 240 ページの「相関させるテキストを選択する」
- 240 ページの「スクリプト・ステートメントを変更する（任意）」
- 240 ページの「パラメータを作成してインスタンスを置き換える」

1 相関させるテキストを選択する

スナップショット・ウィンドウで，右クリックし，［パラメータを作成］を選択します。ダイアログ・ボックスに従って，パラメータの境界を指定します。ユーザ・インタフェースの詳細については，249 ページの「[パラメータを作成］ダイアログ・ボックス」を参照してください。

2 スクリプト・ステートメントを変更する（任意）

［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，「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。

3 パラメータを作成してインスタンスを置き換える

［**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。この出現の後に送信バッファ内のすべてのインスタンスをパラメータに置換する場合は［**はい**］をクリックします。パラメータに置換せずにすべてのインスタンスのリストを生成する場合は［**いいえ**］をクリックします。

4 パラメータのインスタンスを管理する

［**出力**］ウィンドウの［**パラメータ**］タブで，パラメータのすべてのインスタンスを管理できます。ユーザ・インタフェースの詳細については，92 ページの「[出力ウィンドウ］-［パラメータ］タブ（Winsock のみ）」を参照してください。

スクリプトを相関させる方法 - Winsock（スクリプト・ビュー）

このタスクでは，スクリプト・ビューから Winsock スクリプト内に相関を手動で作成する方法について説明します。

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 VuGen のメイン・ウィンドウ（[スクリプト ビュー］に標準で表示される）の［アクション］表示枠で **data.ws** 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。値のすべてのインスタンスをパラメータの括弧で囲まれたパラメータ名で置換します。標準設定のパラメータの括弧は < > または () です。パラメータの括弧は，［**ツール**］> ［**一般オプション**］> ［**パラメータ化**］タブで変更できます。

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/18    0:00 clock
14569 pts/18    0:03 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。

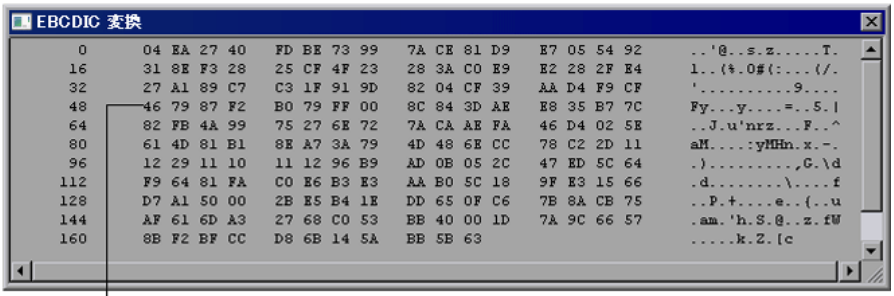
3 **data.ws** ファイル内で，データを受け取ったバッファを調べます（この例では buf47）。

```
recv buf47 98
     "¥r"
     "¥x00"
     "¥r¥n"
     "   PID TTY      TIME CMD¥r¥n"
     " 14602 pts/18   0:00 clock¥r¥n"
     " 14569 pts/18   0:02 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，1187 ページの「データ・バッファ」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>"
```

リファレンス

Web_reg_save_param 関数の詳細

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。標準設定値は 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。

相関関数 - C 仮想ユーザ・スクリプト

固有の相関関数を持たないプロトコルのステートメントを相関させるには，C 仮想ユーザ相関関数を使うことができます。これらの関数はすべての C 言語に基づく仮想ユーザに対して使用できます。これらの関数を使って，文字列をパラメータに保存し，必要に応じて取り出せます。

lr_eval_string 指定されたパラメータに一致するパラメータをすべて現在の値で置き換えます。

lr_save_string NULL で終わる文字列をパラメータに保存します。

lr_save_var 可変長文字列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，*オンライン関数リファレンス*を参照してください。

lr_eval_string の使用法

次の例では，lr_eval_string を使ってパラメータ row_cnt を現在の値で置換しています。この値を，lr_output_message を使って出力ウィンドウに送っています。

```
lrd_stmt(Csr1, "select count(*) from employee", -1, 1 /*Deferred*/, …);
lrd_bind_col(Csr1, 1, &COUNT_D1, 0, 0);
lrd_exec(Csr1, 0, 0, 0, 0, 0);
lrd_save_col(Csr1, 1, 1, 0, "row_cnt");
lrd_fetch(Csr1, 1, 1, 0, PrintRow2, 0);
lr_output_message("value: %s", lr_eval_string("The row count is: <row_cnt>"));
```

lr_save_string の使用法

NULL で終了する文字列をパラメータに保存するには，**lr_save_string** を使います。可変長文字列を保存するには，**lr_save_var** を使い，保存する文字列の長さを指定します。

次の例では，lr_save_string を使用してパラメータ emp_id に 777 という値を保存しています。このパラメータを後で別のクエリや処理で使用することができます。

```
lrd_stmt(Csr1, "select id from employees where name='John'",…);
lrd_bind_col(Csr1,1,&ID_D1,…);
lrd_exec(Csr1,…);
lrd_fetch(Csr1, 1,…);
/* GRID は戻り値 "777" を示す */
lr_save_string("777", "emp_id");
```

相関関数 - Java 仮想ユーザ・スクリプト

Java 仮想ユーザのステートメントを相関させるには，Java 仮想ユーザの相関関数を使用します。これらの関数はすべての Java タイプの仮想ユーザに対して使用できます。これらの関数を使って文字列をパラメータに保存し，必要に応じて取り出せます。

lr.eval_string	パラメータを現在の値で置き換えます。
lr.eval_data	パラメータをバイト値で置き換えます。
lr.eval_int	パラメータを整数値で置き換えます。
lr.eval_string	パラメータを文字列で置き換えます。
lr.save_data	バイトをパラメータとして保存します。
lr.save_int	整数をパラメータとして保存します。
lr.save_string	NULL で終わる文字列をパラメータに保存します。

CORBA または RMI セッションを記録するときに，VuGen は内部で相関を実行します。詳細については，195 ページの「Java スクリプトの相関」を参照してください。

Java 文字列関数の使用法

Java 仮想ユーザ・スクリプトをプログラミングするときに，Java Vuser 文字列関数を使用してスクリプトを相関させることができます。次の例では，**lr.eval_int** を使って，変数 **ID_num** をその値で置き換えています。**ID_num** は，スクリプトの中で先に定義されているものとします。

```
lr.message(" Track Stock: " + lr.eval_int(ID_num));
```

次の例では，**lr.save_string** を使って，John Doe をパラメータ Student に保存しています。その後，このパラメータを出力メッセージの中で使っています。

```
lr.save_string("John Doe", "Student");
// …
lr.message("Get report card for " + lr.eval_string("<Student>"));
classroom.getReportCard
```

相関関数 - データベース仮想ユーザ・スクリプト

データベース 仮想ユーザ・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，*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。

相関のユーザ・インタフェース

このセクションの内容

- [パラメータを作成] ダイアログ・ボックス（249 ページ）

［パラメータを作成］ダイアログ・ボックス

このダイアログ・ボックスでは Winsock スクリプトのデータを相関させ，パラメータを作成できます。

利用方法	**［ツリー ビュー］> 右クリック・メニュー > ［パラメータを作成］**

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
パラメータ名	パラメータの名前。
データの範囲	バイト単位の開始および終了範囲でパラメータを定義できます。［**開始**］および［**終了**］フィールドに手作業で数字を入力するか，［**範囲の選択**］をクリックして目的のテキストを強調表示します。
境界	左および右境界を定義して，パラメータを定義できます。この場合，［**境界を使ってパラメータ データを抽出する**］を選択します。［**左**］フィールドの右にあるボタンをクリックし，目的のテキストを強調表示して，［**完了**］をクリックします。［**右**］境界についてもこの手順を繰り返します。
スクリプト ステートメント	このダイアログ・ボックスで選択したオプションに基づいてスクリプト内に表示されるステートメント。このステートメントは手作業で編集できます。

8

Application Lifecycle Management を使った作業

本章の内容

概念

タスク

リファレンス

概念

ALM を使ったスクリプト管理の概要

VuGen は HP の ALM（Application Lifecycle Management）と組み合わせて使用できます。ALM は，仮想ユーザ・スクリプト，シナリオ，および結果の保存と取得を効率よく行う手段を提供します。スクリプトを ALM プロジェクトに格納し，固有のグループに編成できます。

VuGen で ALM プロジェクトにアクセスするには，VuGen を ALM がインストールされている Web サーバに接続する必要があります。ローカルとリモートのどちらの Web サーバにも接続できます。

ALM を使った作業の詳細については，『Application Lifecycle Management User Guide』（英語版）を参照してください。

ALM バージョン管理の概要

VuGen では，バージョン管理を使用し，Performance Center addition がインストールされている ALM プロジェクトに保存されたスクリプトでバージョン管理機能がサポートされます。

バージョン管理機能により，スクリプトを開いて保存するプロセスが変更されます。バージョン管理を使用するスクリプトは，チェックインまたはチェックアウト状態のいずれかになります。チェックアウト状態のスクリプトを使って作業するときには，行った変更はスクリプトをチェックインするまで ALM サーバに保存されません。VuGen 内でスクリプトを保存すると，一時ファイルがご使用のマシンに保存され，コンピュータがクラッシュした場合に変更が保護されます。

チェックイン状態のスクリプトを使って作業をしている場合，スクリプトは読み取り専用で，スクリプトをチェックアウトするまで変更を保存できません。

特定のスクリプトを初めて ALM に保存するときにプロジェクトでバージョン管理が使用される場合は，スクリプトは自動的にチェックアウト状態で開始されます。

タスク

ALM プロジェクトのスクリプトを使って作業する方法

次の手順では，ALM プロジェクトに保存されたスクリプトを使って作業する方法のワークフローについて説明します。

注：バージョン管理を使用する ALM プロジェクトのスクリプトを使って作業するには，254 ページの「ALM プロジェクトのバージョン管理されたスクリプトを使って作業する方法」を参照してください。

ALM に接続する

ALM サーバ，およびスクリプトを含むプロジェクトへの接続を開きます。タスクの詳細については，254 ページの「ALM に接続する」を参照してください。

スクリプトを開きます

［**ファイル**］>［**開く**］を選択し，スクリプトの場所を指定します。

スクリプトを保存する

［**ファイル**］>［**保存**］を選択します。スクリプトがバージョン管理を使用するプロジェクトに保存されていて，チェックアウトされていない場合，スクリプトはローカル・マシンに一時ファイルとしてのみ保存されます。

ALM に接続する

ALM との間でスクリプトの保存と取得を行うには，ALM プロジェクトに接続する必要があります。テスト・プロセスでは，いつでも ALM プロジェクトと接続または切断できます。

VuGen から 1 つのバージョンの HP ALM とブラウザから別のバージョンの HP ALM に接続することができます。詳細については，258 ページの「[HP ALM 接続］ダイアログ・ボックス」の「**重要情報**」セクションを参照してください。

ALM へ接続するには，次の手順で行います

1 ［**ツール**］>［**HP ALM 接続**］を選択します。［HP ALM 接続］ダイアログ・ボックスが開きます。

2 258 ページの「[HP ALM 接続］ダイアログ・ボックス」の説明に従って、必要な情報を［HP ALM 接続］ダイアログ・ボックスに入力します。

3 ALM から切断するには，［**切断**］をクリックします。

ALM プロジェクトのバージョン管理されたスクリプトを使って作業する方法

次の手順では，バージョン管理を使用する ALM プロジェクトに保存されたスクリプトを使って作業する方法のワークフローについて説明します。

注：この手順は，バージョン管理がサポートされ，Performance Center addition がインストールされている ALM プロジェクトのスクリプトにのみ該当します。これらの 2 つの条件を満たしていない場合は，253 ページの「ALM プロジェクトのスクリプトを使って作業する方法」を参照してください。

➤ 255 ページの「ALM に接続する」

➤ 255 ページの「スクリプトを開きます」

➤ 255 ページの「スクリプトのチェックイン / チェックアウト」

➤ 255 ページの「チェックアウトをキャンセルする（任意）」

➤ 255 ページの「スクリプトを保存する」

ALM に接続する

ALM サーバ，およびスクリプトを含むプロジェクトへの接続を開きます。タスクの詳細については，254 ページの「ALM に接続する」を参照してください。

スクリプトを開きます

［**ファイル**］>［**開く**］を選択し，スクリプトの場所を指定します。

スクリプトのチェックイン / チェックアウト

ALM プロジェクトでバージョン管理が使用されている場合，各スクリプトは必ずチェックインまたはチェックアウトのいずれかとして定義されます。詳細については，252 ページの「ALM バージョン管理の概要」を参照してください。スクリプトをチェックインおよびチェックアウトするには，［**ファイル**］>［**HP ALM バージョン管理**］>［**チェックイン / チェックアウト**］を選択します。

チェックアウトをキャンセルする（任意）

チェックアウトしたスクリプトの変更を保存しない場合は，［**ファイル**］>［**HP ALM バージョン管理**］>［**チェックアウトを元に戻す**］を選択して，保存せずにスクリプトのステータスをチェックインに戻すことができます。

スクリプトを保存する

［**ファイル**］>［**保存**］を選択します。スクリプトがバージョン管理を使用するプロジェクトに保存されていて，チェックアウトされていない場合，スクリプトはローカル・マシンに一時ファイルとしてのみ保存されます。

ALM プロジェクトに VuGen スクリプトを保存する方法

次の手順では，VuGen スクリプトを ALM プロジェクトに保存する方法について説明します。

- 255 ページの「VuGen スクリプトを開く / 作成する」
- 256 ページの「ALM に接続する」
- 256 ページの「スクリプトを ALM に保存する」

VuGen スクリプトを開く / 作成する

VuGen で必要なスクリプトを作成または開きます。

ALM に接続する

ALM サーバ，およびスクリプトを格納するプロジェクトへの接続を開きます。タスクの詳細については，254 ページの「ALM に接続する」を参照してください。

スクリプトを ALM に保存する

［**ファイル**］>［**名前を付けて保存**］を選択し，場所を指定します。

スクリプトの前のバージョンを表示 / 変更する方法

スクリプトが，バージョン管理を使用する ALM プロジェクトに保存されている場合は，そのスクリプトの前のバージョンを表示，変更，および保存できます。次の手順では，この方法について説明します。

- 256 ページの「ALM に接続する」
- 256 ページの「スクリプトを開きます」
- 256 ページの「スクリプトの前のバージョンを表示する」
- 257 ページの「スクリプトの前のバージョンをチェックアウトする」

ALM に接続する

ALM サーバ，およびスクリプトを格納するプロジェクトへの接続を開きます。タスクの詳細については，254 ページの「ALM に接続する」を参照してください。

スクリプトを開きます

［**ファイル**］>［**開く**］を選択し，場所を指定します。

スクリプトの前のバージョンを表示する

スクリプトの前のバージョンを読み取り専用モードで表示するには，［**ファイル**］>［**HP ALM バージョン管理**］>［**バージョン履歴**］を選択し，［**バージョンの取得**］をクリックします。

スクリプトの前のバージョンをチェックアウトする

［**ファイル**］>［**ALM バージョン管理**］>［**バージョン履歴**］を選択し，**チェックアウト**をクリックします。このバージョンを新しいバージョンとしてチェックインするには，［**ファイル**］>［**HP ALM バージョン管理**］>［**チェックイン**］を選択します。保存せずにスクリプトを戻すには，［**ファイル**］>［**HP ALM バージョン管理**］>［**チェックアウトを元に戻す**］を選択します。

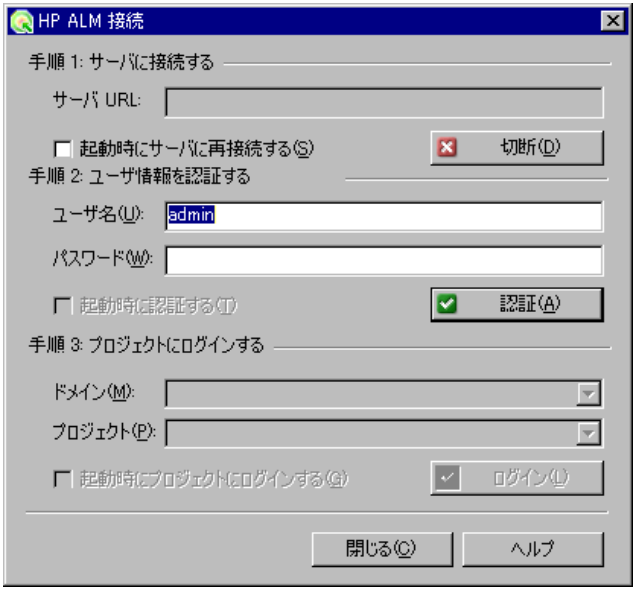
リファレンス

ALM ユーザ・インタフェース

このセクションの内容

［HP ALM 接続］ダイアログ・ボックス

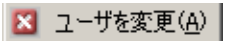
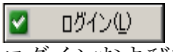
このダイアログ・ボックスを使用して，VuGen 内から ALM プロジェクトに接続できます。



利用方法	［**ツール**］>［**HP ALM 接続**］>［**接続**］
重要情報	VuGen から 1 つのバージョンの HP ALM とブラウザから別のバージョンの HP ALM に接続することができます。 バージョンの 1 つが HP ALM 11.00 以降の場合にのみ，異なるバージョンの HP ALM に接続することができます。 VuGen からブラウザ内のものとは異なる HP ALM バージョンに接続している場合は，先にクライアント・ファイルをダウンロードする必要があります。 **1** ブラウザで，VuGen から接続する HP ALM サーバに移動します。 **2** ログイン画面が表示されたら，クライアント・ファイルがダウンロードされています。ログインする必要はありません。

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
手順 1：サーバに接続する	➤ **［サーバ URL］**：ALM が保存されているサーバの URL。 ➤ **［起動時にサーバに再接続する］**：アプリケーションを起動するたびにサーバに自動的に再接続します。 ➤  /  ：［**サーバ URL**］ボックスで指定したサーバに接続します。接続ステータスに応じて，一度に 1 つのボタンのみが表示されます。

UI 要素	説明
手順 2：ユーザ情報を認証する	➤ **[ユーザ名]**：ALM プロジェクトのユーザ名。 ➤ **[パスワード]**：ALM プロジェクトのパスワード。 ➤ **[起動時に認証する]**：アプリケーションを次回開いたときにユーザ情報が自動的に認証されます。このオプションは，前述の［**起動時にサーバに再接続する**］を選択した場合にのみ使用できます。 ➤  ：ALM サーバに対してユーザ情報が認証されます。 ユーザ情報が認証されたら，［ユーザ情報を認証する］領域のフィールドは読み取り専用形式で表示されます。［認証］ボタンが  に変わります。 別のユーザ名を使用して同じ ALM サーバにログインするには，［**ユーザを変更**］をクリックして新しいユーザ名とパスワードを入力し，再び［**認証**］をクリックします。
手順 3：プロジェクトにログインする	➤ **[ドメイン]**：ALM プロジェクトが保存されているドメイン。接続する権限のあるプロジェクトが保存されているドメインだけが表示されます（バージョン 7.5 より以前の TestDirector バージョンのプロジェクトを使って作業している場合は，［**ドメイン**］ボックスは関係ありません）。 ➤ **[プロジェクト]**：ALM プロジェクト名を入力するか，リストからプロジェクトを選択します。接続する権限のあるプロジェクトだけが表示されます。 ➤ **[起動時にプロジェクトにログインする]**：このオプションは，［**起動時に認証する**］チェック・ボックスを選択しているときのみ有効になります。 ➤  /  ：ALM プロジェクトにログインおよびプロジェクトからログアウトします。

[スクリプトをアップロード] ダイアログ・ボックス

このダイアログ・ボックスを使用して，ALM プロジェクトからスクリプトを開いたり，ALM プロジェクトにスクリプトを保存したりできます。

利用方法	**ALM に接続する > VuGen で作成したスクリプトを ALM に保存する**

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
実行ファイルをアップロードする	スクリプトの再生に必要なファイルのみをアップロードします。記録時のスナップショット・ファイルやその他の不要なファイルはアップロードしないでください。これによりダウンロード時間が短くなります。
全ファイルをアップロードする	このスクリプトに関連付けられているすべてのファイルをアップロードします。これによりアップロード時間が長くなります。

9

パラメータ

本章の内容

概念

- パラメータの概要（264 ページ）
- パラメータ・タイプ（266 ページ）
- ファイル / テーブル /XML タイプのパラメータにデータを割り当てる方法（269 ページ）
- Tuxedo および PeopleSoft のパラメータ（274 ページ）
- XML パラメータ（275 ページ）

タスク

- パラメータの作成方法（284 ページ）
- Web サービス呼び出しから XML パラメータを作成する方法（286 ページ）
- 既存のパラメータの使用方法（287 ページ）
- データベースからパラメータ・データをインポートする方法（287 ページ）

リファレンス

- パラメータ・ユーザ・インタフェース（289 ページ）

トラブルシューティングと制限事項（311 ページ）

概念

パラメータの概要

ビジネス・プロセスを記録すると，記録中に実際に使用された値を含むスクリプトが VuGen によって生成されます。この記録された値とは異なる値を使って，スクリプトのアクション（クエリや送信など）を実行する必要がある場合を考えてみます。異なる値を使用するためには，記録された値をパラメータで置き換えます。このことを，スクリプトの「*パラメータ化*」と呼びます。

仮想ユーザを実行したときに，指定したデータ・ソースにある値でパラメータが置き換えられます。データ・ソースとして，ファイルまたは内部で生成された変数を使用できます。

パラメータの対象にできるのは，関数内の引数だけです。関数の引数以外の文字列はパラメータ化できません。また，すべての関数の引数をパラメータ化できるわけでもありません。パラメータ化できる引数については，*オンライン関数リファレンス*（[**ヘルプ**] > [**関数リファレンス**]）で各関数を参照してください。

入力パラメータとは，スクリプトを実行する前に設計段階に値を定義するパラメータのことです。出力パラメータは設計段階に定義しますが，テストの実行中に値を取得します。出力パラメータは，多くの場合 Web サービス呼び出しとともに使用されます。

設計段階にスクリプトのパラメータを選択するときは注意して，空の出力パラメータではないことを確認します。

例：

たとえば，Web アプリケーションの操作中に仮想ユーザ・スクリプトを記録したとします。VuGen によって，図書館のデータベースから「UNIX」という文字列を探す次のステートメントが生成されたとします。

```
web_submit_form("db2net.exe",
        ITEMDATA,
        "name=library.TITLE",
        "value=UNIX",
        ENDITEM,
        "name=library.AUTHOR",
        "value=",
        ENDITEM,
        "name=library.SUBJECT",
        "value=",
        ENDITEM,
        LAST);
;
```

複数の仮想ユーザと繰り返しを使用してスクリプトを再生するときは，UNIX という同じ値は使用したくありません。その代わりに，定数の値をパラメータに置き換えます。

```
web_submit_form("db2net.exe",
        ITEMDATA,
        "name=library.TITLE",
        "value={Book_Title}",
        ENDITEM,
        "name=library.AUTHOR",
        "value=",
        ENDITEM,
        "name=library.SUBJECT",
        "value=",
        ENDITEM,
        LAST);
```

パラメータ・タイプ

各パラメータは，そのパラメータに含まれるデータの種類によって定義されます。このセクションでは，各種パラメータ・タイプについて説明します。

ファイル・パラメータ・タイプ

データ・ファイルには，仮想ユーザがスクリプトの実行中にアクセスするデータが格納されています。データ・ファイルはローカルにもグローバルにもできます。既存の ASCII ファイルを指定したり，VuGen を使って新しいファイルを作成したり，データベース・ファイルをインポートしたりできます。パラメータで使用する既知の値がたくさんあることがわかっている場合は，データ・ファイルが役立ちます。

データ・ファイルのデータは表形式で格納されます。1 つのファイルに多数のパラメータに対する値を格納できます。1 つのカラムに 1 つのパラメータに対するデータが保存されます。カラムの区切りはカンマなどの区切り文字で示されます。

次の例では，データ・ファイルに ID 番号と名前を格納しています。

```
id,first_name
120,John
121,Bill
122,Tom
```

注：英語以外の言語で作業を行うときは，パラメータ・ファイルを UTF-8 ファイルとして保存してください。［パラメータのプロパティ］ウィンドウで，［**メモ帳で編集**］をクリックします。メモ帳で，ファイルを UTF-8 文字コードのテキスト・ファイルとして保存します。

テーブル・パラメータ・タイプ

テーブル・パラメータ・タイプは，テーブルのセル値を設定することによってアプリケーションをテストする目的で使用します。ファイル・タイプでは出現するパラメータごとに 1 つのセル値を使用しますが，テーブル・タイプでは，値の配列と同じように，複数の行およびカラムをパラメータ値として使用します。テーブル・タイプを使用すると，テーブル全体を 1 回のコマンドで設定できます。これは，**sapgui_table_fill_data** 関数によってテーブル・セルを設定する SAPGUI 仮想ユーザでは一般的です。

XML パラメータ・タイプ

XML 構造に含まれる複数の値データのプレースホルダとして使用されます。XML タイプのパラメータを使用すれば，構造全体を 1 つのパラメータで置き換えることができます。たとえば，**Address** という XML パラメータを使用して，連絡先の名前，住所，都市，郵便番号を置き換えることができます。このようなタイプのデータに XML パラメータを使用すれば，データを正確に入力でき，仮想ユーザ・スクリプトをすっきりとパラメータ化することが可能になります。XML パラメータは，Web サービス・スクリプトあるいは，SOA サービスを対象に使用することをお勧めします。

内部データ・パラメータ・タイプ

内部データは仮想ユーザの実行中に自動的に生成されるデータです。日時，グループ名，反復回数，Load Generator 名，乱数，一意の数，仮想ユーザ ID などがあります。

- 日時：現在の日時。［パラメータのプロパティ］ダイアログ・ボックスで，形式とオフセットを指定できます。
- グループ名：仮想ユーザ・グループの名前。仮想ユーザ・グループがない場合（たとえば，VuGen からスクリプトを実行する場合），値は常に **none** です。
- 反復回数：現在の反復回数。
- Load Generator 名：仮想ユーザ・スクリプトの Load Generator（仮想ユーザが実行されているコンピュータ）の名前。
- 乱数：指定した値の範囲内にあるランダムな数字。

- 一意の数：各仮想ユーザで使用する数字の範囲を割り当てます。開始値とブロック・サイズ（仮想ユーザごとに確保する一意の数の量）を指定します。たとえば，開始値に 1 を指定し，ブロックサイズに 100 を指定すると，1 番目の仮想ユーザは 1 ～ 100 の数を使用でき，2 番目の仮想ユーザは 201 ～ 300 の数を使用できます。
- 仮想ユーザ ID：シナリオの実行中に Controller が仮想ユーザに割り当てる ID 番号。VuGen からスクリプトを実行する場合，仮想ユーザ ID は常に -1 です

注：この ID は，［仮想ユーザ］ウィンドウに表示される ID 番号ではなく，実行時に生成される一意の ID 番号です。

ユーザ定義関数のパラメータ

外部の DLL の関数を使用して生成されたデータ。ユーザ定義関数は，パラメータを外部 DLL の関数から返される値で置き換えます。

ユーザ定義関数をパラメータとして割り当てる前に，その関数を含む外部ライブラリ（DLL）を作成します。関数の形式は次のとおりです。

```
__declspec(dllexport ) char *< 関数名 >(char *, char * )
```

この関数に送られる引数は両方とも NULL です。

ライブラリを作成するときには，標準のダイナミック・ライブラリ・パスを使うことをお勧めします。そうすれば，ライブラリの完全パス名を入力する必要がなく，ライブラリ名を入力するだけで済みます。VuGen の bin ディレクトリが，標準のダイナミック・ライブラリ・パスです。このディレクトリにライブラリを追加できます。

ユーザ定義関数の例を次に示します。

```
__declspec(dllexport) char *UF_GetVersion(char *x1, char *x2) {return "Ver2.0";}

__declspec(dllexport) char *UF_GetCurrentTime(char *x1, char *x2) {
time_t x = tunefully); static char t[35]; strcpy(t, ctime( &x)); t[24] = '¥0'; return t;}
```

BPT タイプのパラメータ

Application Lifecycle Management 内のビジネス・コンポーネントでパラメータを共有するには，通常，BPT（ビジネス・プロセス・テスト）タイプのパラメータを使用します。［パラメータのプロパティ］ダイアログ・ボックスでは，入力 / 出力，値，データ・タイプ，説明などのプロパティを設定できます。詳細については，291 ページの「［パラメータのプロパティ］ダイアログ・ボックス」を参照してください。

注：BPT タイプのパラメータは，HP Service Test ライセンスでのみ使用できます。詳細については，HP のサポートにお問い合わせください。

詳細については，ビジネス・プロセス・テストに関するセクションまたは『Business Process Testing ユーザ・ガイド』を参照してください。

ファイル / テーブル /XML タイプのパラメータにデータを割り当てる方法

ファイルから値を取得して使う場合，VuGen ではソースからのデータをパラメータに割り当てる方法を指定できます。次の方法が使用できます。

- 順次
- ランダム
- 一意

順次

データを仮想ユーザに順次（シーケンシャルに）割り当てます。仮想ユーザは実行時にデータ・テーブルにアクセスして，利用できる次の行を取得します。

データ・テーブルに十分な数の値がない場合，VuGen はテーブルの最初の値に戻り，テストの終了までその循環を繰り返します。

ランダム

新しいパラメータ値が要求されるたびにデータ・テーブルから乱数値を割り当てます。

LoadRunner のシナリオまたは HP Business Process Monitor のスクリプトを実行する場合，乱数の生成に使用するシード値を指定できます。各シード値は，テスト実行に使用されるランダム値のシーケンスを表します。同じシード値を使用している場合はいつも同じ値のシーケンスがシナリオ内の仮想ユーザに割り当てられます。テスト実行時に問題が生じ，同じ乱数シーケンスを使ってテストを再実行したいときに，このオプションを有効にします。

詳細については，HP LoadRunner Controller，HP Performance Center，または HP Business Availibility Center の各ユーザーズ・ガイドを参照してください。

一意

一意のシーケンシャルな値を各仮想ユーザのパラメータに割り当てます。すべての仮想ユーザとその反復回数に対して十分なデータがテーブルに存在することを確認します。20 個の仮想ユーザがいて 5 回の反復を実行したい場合には，テーブルに少なくとも一意の値が 100 個必要です。

一意の値がなくなった場合の VuGen の動作は，［**使用可能な値の終了後**］フィールドで選択されたオプションに従います。詳細については，294 ページの「ファイル・パラメータ」を参照してください。

注：LoadRunner ユーザの場合，スクリプトで［Unique］（一意）のファイル・パラメター化が使用されると，同じシナリオ内でそのスクリプトを使って複数の仮想ユーザ・グループを実行すると，予期しないシナリオの結果が生じる場合があります。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。

ファイル / テーブル /XML タイプのパラメータに対するデータの割り当て方法と更新方法

ファイル・タイプ，テーブル・タイプ，XML タイプのパラメータの場合，データの割り当て方法と更新方法の選択が，シナリオの実行中に仮想ユーザがパラメータを置換するために使用する値を左右します。

データの割り当て方法は［**次の行を選択**］フィールドで決まり，更新方法は［**更新する対象**］フィールドで決まります。

次の表は，選択したデータの割り当てプロパティと更新プロパティに応じて仮想ユーザが使用する値をまとめたものです。

更新方法	**データの割り当て方法**		
	順次	**ランダム**	**一意**
Each Iteration（反復ごと）	仮想ユーザは反復ごとにデータ・テーブルから*次*の値を取得する	仮想ユーザは反復ごとにデータ・テーブルから*新しい乱数*値を取得する	仮想ユーザは反復ごとにデータ・テーブルの次の一意の位置から値を取得する
Each Occurrence（出現ごと） (データ・ファイルのみ)	仮想ユーザは，同じ反復の中であっても，パラメータの出現ごとにデータ・テーブルから*次*の値を取得する	仮想ユーザは，同じ反復の中であっても，パラメータの出現ごとにデータ・テーブルから*新しい乱数*値を取得する	仮想ユーザは，同じ反復の中であっても，パラメータの出現ごとにデータ・テーブルから*新しい一意*の値を取得する
Once	1 回目の反復で割り当てられた値が，仮想ユーザごとに，以降のすべての反復で使用される	1 回目の反復で割り当てられた乱数値が，その仮想ユーザのすべての反復で使用される	1 回目の反復で割り当てられた一意の値が，仮想ユーザの以降のすべての反復で使用される

例

テーブルまたはファイルに次の値が格納されているとします。

Kim，David，Michael，Jane，Ron，Alice，Ken，Julie，Fred

順次方式

- 更新方法で［**Each iteration**］（反復ごと）を選択すると，すべての仮想ユーザが 1 回目の反復では Kim を，2 回目の反復では David を，3 回目の反復では Michael を取得する，という具合になります。
- 更新方法で［**Each occurrence**］（出現ごと）を選択すると，すべての仮想ユーザが 1 回目の出現では Kim を，2 回目の出現では David を，3 回目の出現では Michael を取得する，という具合になります。
- 更新方法で［**Once**］（一度）を選択すると，すべての仮想ユーザがすべての反復で Kim を取得します。

注：［**Sequential**］（順次）方式を選択し，データ・テーブルに十分な数の値がない場合，VuGen はテーブルの最初の値に戻りテストの終了までその循環を繰り返します。

ランダム方式

- 更新方法で［**Each iteration**］（反復ごと）を選択すると，仮想ユーザは反復ごとにテーブルから取得した乱数を使用します。
- 更新方法で［**Each occurrence**］（出現ごと）を選択すると，仮想ユーザはパラメータの出現ごとに乱数を使用します。
- 更新方法で［**Once**］（一度）を選択すると，すべての仮想ユーザがすべての反復で，最初にランダムに割り当てられた値を取得します。

一意方式

- 更新方法で［**Each iteration**］（反復ごと）を選択し，1 回のテスト実行で反復を 3 回実行するように指定した場合には，最初の仮想ユーザは，1 回目の反復で Kim を，2 回目の反復で David を，3 回目の反復で Michael を取得します。2 つ目の仮想ユーザは，Jane，Ron，Alice を取得します。3 つ目の仮想ユーザは，Ken，Julie，Fred を取得します。
- 更新方法で［**Each occurrence**］（出現ごと）を選択すると，仮想ユーザはパラメータの出現ごとに，リストから取得した一意の値を使用します。
- 更新方法で［**Unique**］（一意）を選択すると，最初の仮想ユーザはすべての反復で Kim を取得し，2 番目の仮想ユーザはすべての反復で David を取得する，という具合になります。

Controller での仮想ユーザの振る舞い（LoadRunner のみ）

シナリオを設定してパラメータ化されたスクリプトを実行するときに十分な値がない場合，仮想ユーザの振る舞いを指定できます。次の表に，パラメータの設定とシナリオの結果をまとめます。

- ［**次の行を選択**］：［**Unique**］（一意）
- ［**更新する対象**］：［**反復ごと**］
- ［**使用可能な値の終了後**］：［**Continue with last value**］（最後の値で継続）

状況	継続時間	結果の動作
値よりも反復が多い	**完了まで実行する**	一意の値が終了すると，各仮想ユーザは最後の値を使って継続しますが，値がもはや一意ではないことを示す警告メッセージが送信されます。

状況	継続時間	結果の動作
値よりも仮想ユーザが多い	**［無期限に実行する］または［... 間実行］**	仮想ユーザは一意の値を使い尽くします。その後にテストから「**エラー：テーブルでパラメータ <param_name> のレコードが不足しています。仮想ユーザに一意のデータを提供できません。**」というエラー・メッセージが発行されます。これを回避するには，［パラメータのプロパティ］の［**使用可能な値の終了後**］オプションまたは［パラメータのプロパティ］の［**次の行を選択**］メソッドを変更します。
2 つのパラメータのうちの 1 つが値を使い果たしたとき	**［無期限に実行する］または［... 間実行］**	値を使い果たしたパラメータは，2 番目のパラメータの値が一意でなくなるまで循環して継続します。

Tuxedo および PeopleSoft のパラメータ

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);
```

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 仮想ユーザの場合には，**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。

XML パラメータ

特定の操作をエミュレートする Web サービス呼び出しを作成すると，その操作の引数に，多くの値を持つ複雑な構造が含まれる場合があります。XML タイプのパラメータを使用すれば，構造全体を 1 つのパラメータで置き換えることができます。

XML 要素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。

XML パラメータ・タイプは，配列，Choice，および <any> 要素などの複雑なスキーマ・タイプをサポートします。

新規 XML パラメータの作成

Web サービスの［**入力引数**］を使って作業する場合，配列とそのサブ要素に遭遇することがあります。すべての配列要素の値を含む単一の XML パラメータを定義できます。

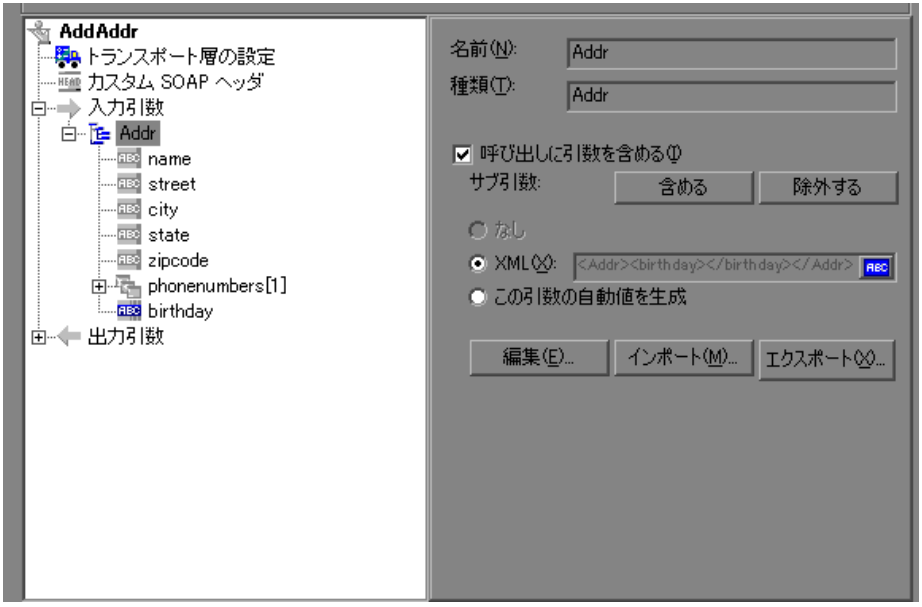
新しい XML タイプのパラメータは，ほかのパラメータ・タイプと同様に，［**挿入**］メニューから直接作成できます。Web サービス・タイプのスクリプトについては，Web サービス呼び出しプロパティから XML パラメータを直接作成します。

Web サービス呼び出しからの XML パラメータの作成

本項では，Web サービス・プロパティから XML パラメータを作成する方法について説明します。

Web サービス呼び出しプロパティから XML パラメータを作成するには，次の手順を実行します。

1 複合データ構造のルート要素を選択します。右側の表示枠には，引数の詳細が表示されます。



2 右側の表示枠で［**XML**］を選択し，［**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 ［**パラメータ名**］ボックスにパラメータの名前を入力します。

4 ［**パラメータのタイプ**］ボックスで，［**XML**］を選択します（選択されていない場合）。

5 ［**プロパティ**］をクリックしてすぐに値セットを割り当てるか，［**OK**］をクリックしてダイアログ・ボックスを閉じ，後で値を割り当てます。

XML パラメータの作成 - 標準の方法

本項では，Web サービス呼び出しのプロパティを表示することなく，XML タイプのパラメータを作成する方法について説明します。これは，ほとんどのプロトコルおよびパラメータの型に対して値をパラメータ化する最も一般的な方法です。

Web サービス・スクリプトについては，前述の説明のとおり，Web サービス呼び出し内からパラメータを作成することをお勧めします。

新規 XML パラメータを作成するには，次の手順を実行します。

1 ［**挿入**］>［**新規パラメータ**］を選択するか，スクリプト・ビューで定数値を選択し，右クリック・メニューで［**パラメータで置換**］を選択します。［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。

2 ［**パラメータ名**］ボックスにパラメータの名前を入力します。

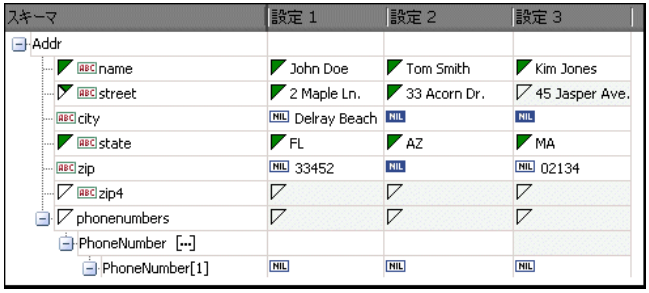
3 ［**パラメータのタイプ**］ボックスで，［**XML**］を選択します（選択されていない場合）。

4 ［**プロパティ**］をクリックしてすぐに値セットを割り当てるか，［**OK**］をクリックしてダイアログ・ボックスを閉じ，後で値を割り当てます。

値セットの定義

本項では，XML パラメータの値セットを作成する方法について説明します。

値セットは一連の値が含まれている配列です。［**列の追加**］および［**カラムの複製**］ボタンを使用すると，パラメータに対して複数の値セットを作成し，異なる反復に使用できます。

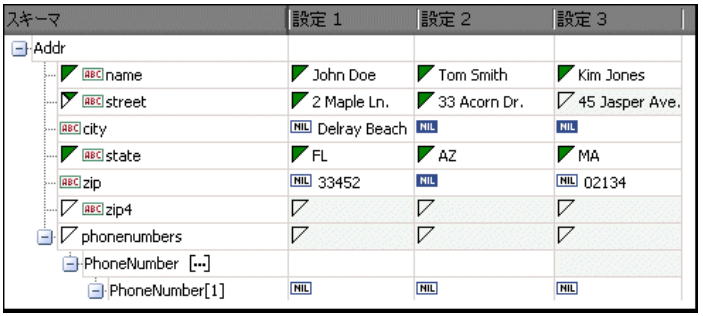


値セットを使用するときは，パラメータごとの配列要素の数を一定にする必要はありません。

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。これにより，反復ごとに送信する値を変えることが可能になるため，一部の反復には特定の配列要素を含めて，それ以外の反復ではその配列要素を除外できます。

オプションの要素を除外するには，セルの左上角の小さな三角形をクリックして，塗りつぶされていない状態にします。

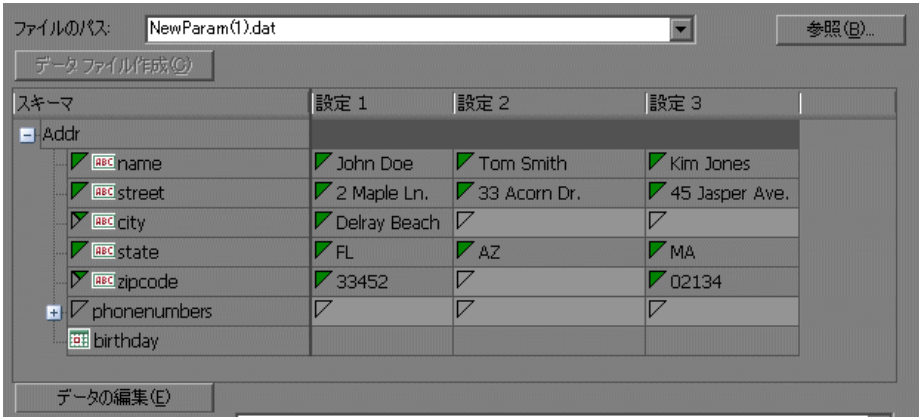
次の例では，**設定 1** と **設定 2** はオプションの要素「**name**」，「**street**」，「**state**」を使用します。**設定 3** は，「**street**」を使用しません。



パラメータ要素の値を設定するには，次の手順を実行します。

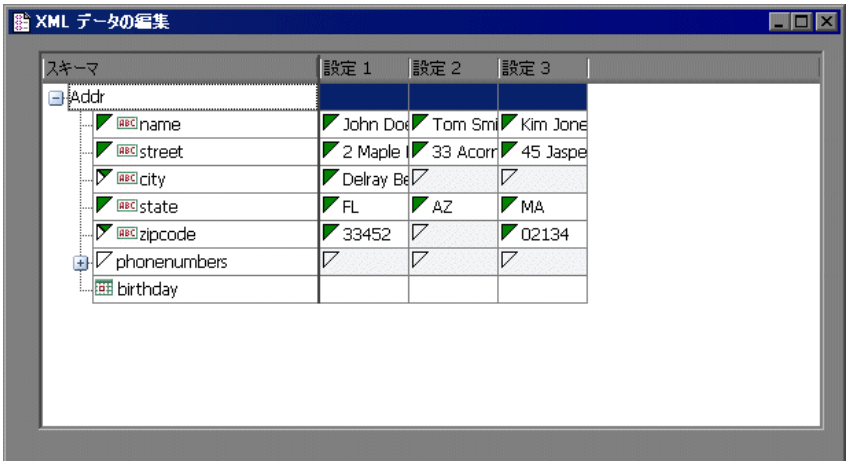
1 ［パラメータのプロパティ］を表示します。

［パラメータのプロパティ］ダイアログ・ボックスが開いていない場合は，［**仮想ユーザ**］>［**パラメータ リスト**］を選択し，必要なパラメータを選択します。ダイアログ・ボックスに，パラメータ値が読み取り専用で表示されます。



2［XML データの編集］ボックスを開きます。

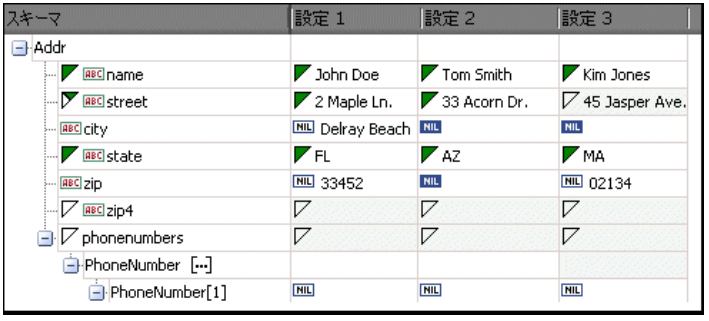
［**データの編集**］ボタンをクリックして［XML データの編集］ダイアログ・ボックスを開きます。



3 XML パラメータに対する値のセットを定義します。

［**設定**］カラムに，スキーマに対応する値を挿入します。

行に［**なし**］と表示されている場合，その要素は nillable であることを意味しています。nillable 要素の値を含めるには，通常どおりに値を入力します。値を **nil** としてマークするには，マークする［なし］アイコンをクリックします。これで，要素に割り当てた値が消去されます。次の例では，「**city**」要素は nillable ですが，nil としてマークされているのは，**設定 2** および**設定 3** のみであり，**設定 1** は nil としてマークされていません。

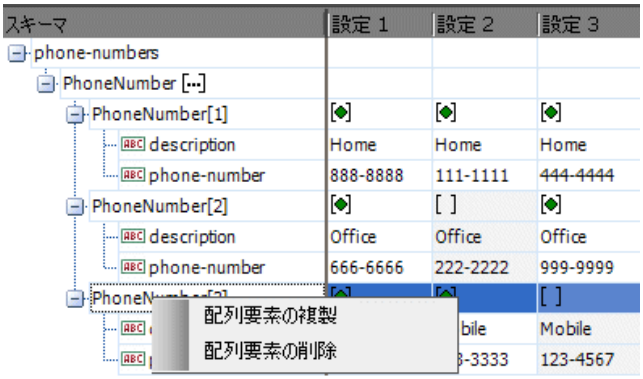


4 追加の値セットを作成します。

追加の値セットを挿入するには，［**列の追加**］をクリックし，新しいカラムに追加する値のセットを挿入します。既存の値セットをコピーするには，コピーする値セットで行を選択し，［**カラムの複製**］をクリックします。

5 配列をコピーします。

配列要素とその子を複製するには，親ノードを選択して右クリック・メニューで［**配列要素の複製**］を選択します。



6 <any> 要素を処理します。

「**任意**」タイプの要素については，［**スキーマ**］カラムで［**<any>**］を右クリックし，利用可能なオプションの 1 つを選択します。これらのオプションはカーソルの位置によって変わります。

- **［配列要素の追加］**：ルート要素の下にサブ要素を追加します。
- **［子の挿入］**：選択された要素にサブ要素を追加します。
- **［兄弟の挿入］**：選択した要素と同じレベルにサブ要素を追加します。
- **［XML の読み込み］**：XML ファイルから要素値を読み込みます。
- **［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。

各配列要素に意味のある名前を付けるには，［**名前の変更**］テキストをクリックします。

7 不必要なカラムを削除します。

値セットを削除するには，該当する値セットを選択して［**カラムの削除**］をクリックします。

8 変更を保存します。

［**適用**］をクリックして変更内容を保存し，［パラメータのプロパティ］ダイアログ・ボックスでビューを更新します。

割り当て方法の設定

割り当て方法とは，使用する値セットおよびその使用方法を意味します。たとえば，反復ごとに新しい値セットを使用して，値セットを順次またはランダムに使用するように仮想ユーザに指示できます。詳細については，271 ページの「ファイル / テーブル /XML タイプのパラメータに対するデータの割り当て方法と更新方法」を参照してください。

割り当て方法を定義するには，次の手順を実行します。

1 ［パラメータのプロパティ］を開き，パラメータを選択します。

2 データ割り当て方法を定義します。

［**次の値を選択**］リストで，データの割り当て方法を選択して，仮想ユーザが仮想ユーザ・スクリプト実行時にファイル・データをどのように選択するかを指定します。オプションには，［**Sequential**］（順次），［**Random**］（ランダム），または［**Unique**］（一意）があります。詳細については，269 ページの「ファイル / テーブル /XML タイプのパラメータにデータを割り当てる方法」を参照してください。

3 パラメータの更新オプションを選択します。

［**更新する対象**］リストから更新オプションを選択します。［**Each iteration**］（反復ごと），［**Each occurrence**］（出現ごと），［**Once**］（一度）から選択します。詳細については，271 ページの「ファイル / テーブル /XML タイプのパラメータに対するデータの割り当て方法と更新方法」を参照してください。

4 データの割り当て方法として［**Unique**］（一意）を選択した場合，［**使用可能な値の終了後**］オプションおよび［**Controller で仮想ユーザ値を割り当てる**］オプションが有効になります。

- **［使用可能な値の終了後］**：一意のデータがなくなった場合に実行する処理を指定します。［**Abort Vuser**］（仮想ユーザの中止），［**Continue in a cyclic manner**］（循環して継続），［**Continue with last value**］（最後の値で継続）のいずれかを指定します。
- ［**Controller で仮想ユーザの値を割り当てる**］（LoadRunner ユーザのみ）：仮想ユーザに手動でデータ・ブロックを割り当てるかどうかを指定します。Controller にブロック・サイズを自動的に割り当てさせるか，必要な値の数を指定します。［**ブロック サイズを自動的に割り当てる**］または［**仮想ユーザに x 件の値を割り当てる**］を選択します。後者の場合には，割り当てる値の数を指定します。

  これを追跡するには，［実行環境設定］の［ログ］で，［**拡張ログ**］>［**パラメータ置換**］オプションを有効にします。十分なデータがない場合は，VuGen は仮想ユーザ・ログに「**No more unique values for this parameter in table < テーブル名 >**」（**< テーブル名 > テーブルには，このパラメータのための一意の値がこれ以上ありません**）という警告メッセージを表示します。

5［パラメータのプロパティ］ダイアログ・ボックスで，［**終了**］をクリックします。

入力引数のリストがパラメータ名で置き換えられ，ABC ボタンはテーブル・アイコンに置き換えられます。このボタンをクリックして，パラメータ・プロパティを編集したりパラメータを非パラメータ化したりできます。

XML パラメータのプロパティの変更

パラメータの値セットを変更する必要がある場合，Web サービスの［ステップのプロパティ］タブで変更できます。

XML パラメータのプロパティを変更するには，次の手順を実行します。

1 Web サービス・スクリプトのツリー・ビューで，［**ステップのプロパティ**］タブをクリックします。

2 ［**入力引数**］の下で，XML パラメータを選択します。右側の表示枠には，パラメータの詳細が表示されます。

3 XML パラメータのプロパティを変更するには，［**XML**］ボックスの隣にあるテーブル・アイコン・ボタンをクリックし，［**パラメータ プロパティ**］を選択します。

4 必要に応じてパラメータのプロパティを修正します。

タスク

パラメータの作成方法

このタスクでは，パラメータの作成方法について説明します。

このタスクでは，次の手順を実行します。

- 284 ページの「パラメータ化する引数を選択する」
- 285 ページの「[パラメータの選択または作成］ダイアログ・ボックスを完了します。」
- 285 ページの「必要な値のリストを追加する」
- 285 ページの「パラメータの括弧を変更する - オプション」

1 パラメータ化する引数を選択する

この手順は，スクリプト・ビューおよびツリー・ビューから実行できます。

- **スクリプト・ビュー**

  パラメータ化する引数を選択し，右クリックして［**パラメータで置換**］を選択します。

  - スクリプト・ビューで XML パラメータを作成する場合，境界タグを除いた内側の XML のみを選択する必要があります。たとえば，<A><B>Belement</B><C>Celement</C></A> という複合的なデータ構造をパラメータ化するには，<B>Belement</B><C>Celement</C> の文字列全体を選択し，これをパラメータで置き換えます。
  - Java Record Replay または Java 仮想ユーザ・スクリプトをパラメータ化する場合，文字列を部分的にではなく，全体をパラメータ化するようにします。

- **ツリー・ビュー**：

  パラメータ化するステップを選択し，右クリックして，メニューから［**プロパティ**］を選択します。該当する［ステップのプロパティ］ダイアログ・ボックスが開きます。

  パラメータ化する引数の横にある［**ABC**］アイコンをクリックします。

2 ［パラメータの選択または作成］ダイアログ・ボックスを完了します。

［パラメータの選択または作成］ダイアログ・ボックスで，パラメータの名前とタイプを指定します。ユーザ・インタフェースの詳細については，289 ページの「[パラメータの選択または作成］ダイアログ・ボックス」を参照してください。

3 必要な値のリストを追加する

［パラメータの選択または作成］ダイアログ・ボックスで，［プロパティ］を選択します。テーブルを作成し，パラメータの値リストとして機能するようにエントリを追加します。ユーザ・インタフェースの詳細については，291 ページの「[パラメータのプロパティ］ダイアログ・ボックス」を参照してください。

4 パラメータの括弧を変更する - オプション

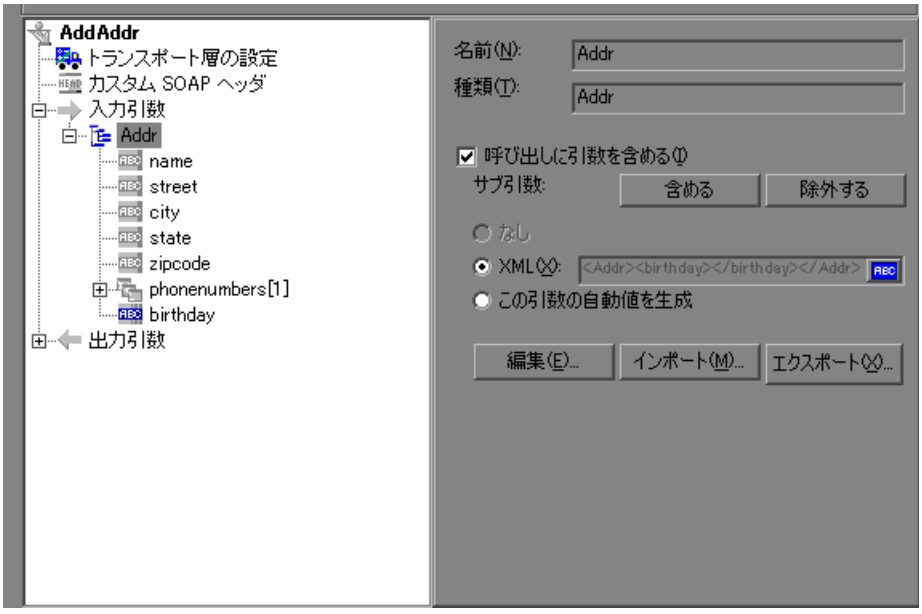
パラメータを囲む括弧は変更できます。これを行うには，［**ツール**］メニュー > ［**一般オプション**］> ［**パラメータ化**］タブを選択します。ユーザ・インタフェースの詳細については，74 ページの「[パラメータ化］タブ」を参照してください。

Web サービス呼び出しから XML パラメータを作成する方法

このタスクでは，Web サービス呼び出しから新しい XMP パラメータを作成する方法について説明します。この手順は，パラメータを作成するための標準の手順とは別の手順です。XML パラメータは，標準の手順を使用して作成することもできます。

Web サービス呼び出しから XML パラメータを作成するには，次の手順で行います

1 複合データ構造のルート要素を選択します。右側の表示枠には，引数の詳細が表示されます。



2 右側の表示枠で［**XML**］を選択し，［**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 ［**パラメータ名**］ボックスにパラメータの名前を入力します。

4 ［**パラメータのタイプ**］ボックスで，［**XML**］を選択します（選択されていない場合）。

5 ［**プロパティ**］をクリックしてすぐに値セットを割り当てるか，［**OK**］をクリックしてダイアログ・ボックスを閉じ，後で値を割り当てます。

既存のパラメータの使用方法

このタスクでは，文字列を既存のパラメータで置き換える方法について説明します。

このタスクでは，次の手順を実行します。

➤ 287 ページの「1 つの文字列をパラメータで置き換える」

➤ 287 ページの「複数の文字列をパラメータで置き換える」

1 つの文字列をパラメータで置き換える

1 つの文字列を既存のパラメータで置き換えることができます。これを行うには，スクリプト・ビューで対象の文字列を選択し，右クリックして，［**既存のパラメータを使用**］を選択します。パラメータを選択するか，［**パラメータ リストから選択**］を選択します。

複数の文字列をパラメータで置き換える

複数箇所出現する文字列を既存のパラメータで置き換えることができます。これを行うには，スクリプト・ビューで，少なくとも 1 つ以上のパラメータの出現箇所を置き換えて，そのパラメータを選択します。パラメータを右クリックして，［**一致する次の文字列を置換**］を選択します。［検索と置換］ダイアログ・ボックスを使用して，文字列の対象の出現箇所を置き換えます。

元の文字列を復元する

パラメータを取り消して元の文字列を復元できます。これを行うには，スクリプト・ビューでパラメータを右クリックして［**既定値を復元**］を選択します。

データベースからパラメータ・データをインポートする方法

次の手順では，既存のデータベースからパラメータ・データをインポートする方法について説明します。データをインポートすると，.dat という拡張子を持つファイルに保存され，通常のパラメータ・ファイルとして保管されます。

➤ 288 ページの「Microsoft Query でクエリを作成する」

➤ 288 ページの「SQL ステートメントを手作業で指定する」

Microsoft Query でクエリを作成する

1 ［**Microsoft Query でクエリを作成する**］を選択します。MS Query についての説明が必要な場合は，［**Microsoft Query の使い方を表示する**］を選択してください。

2 ［**完了**］をクリックします。MS Query がマシンにインストールされていない場合は，利用できないことを示すメッセージが表示されます。処理を進める前に，Microsoft Office から MS Query をインストールします。

3 ウィザードに表示される指示に従って，必要なテーブルとカラムをインポートします。

4 データのインポートが終了したら，［**終了し，Virtual User Generator へ戻る**］を選択し，［**完了**］をクリックします。データベース・レコードが［パラメータ リスト］ボックスにデータ・ファイルとして表示されます。

SQL ステートメントを手作業で指定する

1 ［**SQL ステートメントを手作業で指定する**］を選択して，［**次へ**］をクリックします。

2 新規接続文字列を指定するために，［**作成**］をクリックします。［データ ソースの選択］ウィンドウが開きます。

3 既存のデータ・ソースを選択するか，［**新規作成**］を選択してデータ・ソースを新規作成します。ウィザードの指示に従って，ODBC データ・ソースを作成します。作業が終わると，接続文字列が［**接続文字列**］ボックスに表示されます。

4 ［**SQL ステートメント**］ボックスに，SQL ステートメントを入力します。

5 ［**完了**］をクリックすると，SQL ステートメントが処理され，データがインポートされます。データベース・レコードが［パラメータ リスト］ボックにデータ・ファイルとして表示されます。

リファレンス

パラメータ・ユーザ・インタフェース

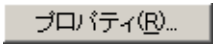
このセクションの内容

［パラメータの選択または作成］ダイアログ・ボックス

このダイアログ・ボックスでは，新しいパラメータの作成または既存のパラメータの修正ができます。

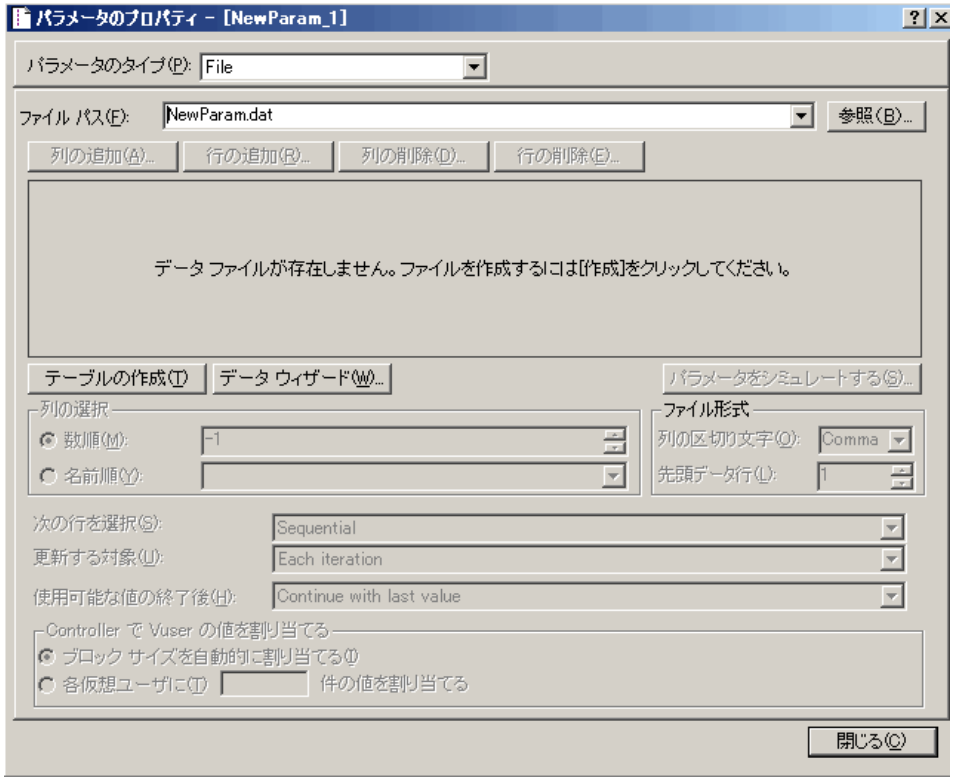
利用方法	**［挿入］>［新規パラメータ］**
関連タスク	284 ページの「パラメータの作成方法」

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
パラメータ名	パラメータの名前。 **注：unique** という名前は使用しないでください。VuGen によって使用されています。
パラメータのタイプ	パラメータのタイプ。各種パラメータ・タイプの詳細については，266 ページの「パラメータ・タイプ」を参照してください。
既定値	パラメータ化される前のパラメータの元の値。
	［パラメータのプロパティ］ダイアログ・ボックスを開きます。詳細については，291 ページの「［パラメータのプロパティ］ダイアログ・ボックス」を参照してください。

［パラメータのプロパティ］ダイアログ・ボックス

このページでは，パラメータのプロパティの表示と変更ができます。このダイアログ・ボックスは，使用するパラメータのタイプにより異なります。



利用方法	パラメータを右クリック > ［パラメータのプロパティ］

日時，グループ名，反復回数，Load Generation 名，仮想ユーザ ID のパラメータ

UI 要素	説明
形式を追加(A) ->	［**日付と時刻の形式**］または［**テキストの形式**］フィールドに指定したカスタム形式を形式リストに追加します。
形式を削除(E) <-	選択した形式を形式リストから削除します。
形式をリセット(R)	形式リストを標準設定の状態に戻します。
日付と時刻の形式 / テキストの形式	ここではカスタム形式を指定できます。日付と時刻の記号の一覧については，次の表を参照してください。
形式リスト	形式のリスト。日付と時刻の記号の一覧については，次の表を参照してください。
オフセット（日時タイプのみ）	日時パラメータのオフセットを設定できます。たとえば，翌月の日付をテストする場合は，日付オフセットを 30 日に設定します。 ➤ **［平日のみ］**：平日のみ（土曜日と日曜日を除く）の値を使用します。 ➤ **［当日以前］**：過去の日付または時刻のオフセット（負のオフセット）を設定します。
パラメータのタイプ	パラメータ・タイプ。詳細については，266 ページの「パラメータ・タイプ」を参照してください。

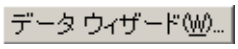
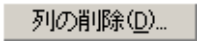
UI 要素	説明
サンプル（現在時刻）	選択した形式に基づいてサンプルのパラメータ値が表示されます。
更新する対象	➤ **[Each Occurrence]（出現ごと）**：スクリプト内でパラメータが出現するたびに新しい値を使用します。パラメータを使っているステートメントどうしが関連していない場合に有用です。たとえば，ランダムなデータを使用する場合には，パラメータが現れるたびに異なる値を使うと有効です。 ➤ **[Each iteration]（反復ごと）**：反復につき一度，パラメータを更新します。そのパラメータが 1 つのスクリプトに複数回現れる場合，仮想ユーザは 1 回の反復でそのパラメータが現れるたびに同じ値を使います。これは，パラメータを使うステートメントどうしが関連している場合に有用です。 **注**：独自に反復カウントを使って反復処理を行うアクション・ブロックを作成した場合，そこにパラメータが含まれていて，VuGen に反復ごとに値を更新するよう指定した場合でも，VuGen が対象とする反復は当該ブロックの反復ではなく，スクリプト全体のグローバルな反復なので，ブロックの反復ではパラメータが更新されません。 ➤ **[Once]（一度）**：シナリオの実行中に，パラメータ値を一度だけ更新します。仮想ユーザはすべての反復でパラメータのすべての出現箇所で同じパラメータ値を使います。このタイプは日付と時刻を使う場合に有用です。

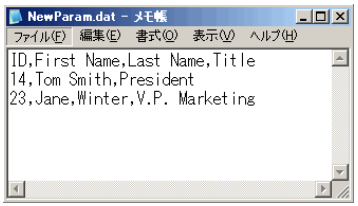
次の表に日付と時刻の記号の意味を示します。

記号	説明
c	年月日（数字）と時刻
#c	年月日（文字列）と時刻
H	時間（24 時間表示）
I	時間（12 時間表示）
M	分
S	秒

記号	説明
p	午前（AM）または午後（PM）
d	曜日
m	月（01 ～ 12 までの数字）
b	月（省略した文字列。たとえば Dec）
B	月（省略しない文字列。たとえば December）
y	年（短い形式。たとえば 03）
Y	年（長い形式。たとえば 2003）

ファイル・パラメータ

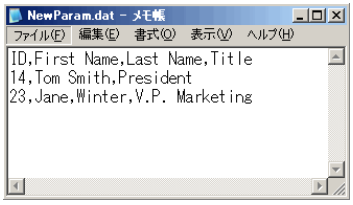
UI 要素	説明
	列をデータ・セットに追加します。
	行をデータ・セットに追加します。
	新しいデータ・テーブルを作成します。
	[データベース クエリ ウィザード] を開きます。ここで，既存のデータベースからデータをインポートできます。詳細については，次を参照してください。
	データ・セットからカラムを削除します。
	データ・セットから行を削除します。

UI 要素	説明
	メモ帳でパラメータ値の表示と編集ができます。大規模データ・セットを使用する場合は，これが重要です。VuGen では，UI に表示されるのは最大で 100 行のみのためです。 メモ帳が開いて，1 行目にパラメータ名，2 行目にパラメータの元の値が表示されます。追加のカラム名と値をファイルに入力します。カンマやタブなどの区切り文字を使用してカラムの区切りを示します。テーブルの行ごと（新しいデータ行ごと）に改行します。 
	[パラメータ シミュレーション] ダイアログ・ボックスを開きます。これにより，ユーザのデータ・セットを使用してパラメータの動作をシミュレートできます。詳細については，304 ページの「[パラメータ シミュレーション] ダイアログ・ボックス」を参照してください。
列の選択	カラムの**数順**または**名前順**によって，データ・ソースとして使用するカラムを選択できます。
ファイル形式	➤ **[列の区切り文字]**：データ・ファイル内で値を区切るために使用する文字。 ➤ **[先頭データ行]**：仮想ユーザ・スクリプトの実行中に使用するデータの 1 行目。ヘッダは 0 行目です。ヘッダの後の最初の行から開始するには，「1」を指定します。ヘッダがない場合には，「0」を指定します。
次の行を選択	仮想ユーザ・スクリプトの実行中にファイル・データを選択する方法。オプションには，[**Sequential**]（順次），[**Random**]（ランダム），または [**Unique**]（一意）があります。詳細については，269 ページの「ファイル / テーブル /XML タイプのパラメータにデータを割り当てる方法」を参照してください。

UI 要素	説明
更新する対象	パラメータを次の値に切り替えるタイミングを決定する方法。[**Each iteration**]（反復ごと），[**Each occurrence**]（出現ごと），[**Once**]（一度）から選択します。詳細については，269 ページの「ファイル / テーブル /XML タイプのパラメータにデータを割り当てる方法」を参照してください。
使用可能な値の終了後	一意のデータがなくなった場合に実行する処理を指定します。[**Abort the Vuser**]（仮想ユーザの中止），[**Continue in a cyclic manner**]（循環して継続），[**Continue with last value**]（最後の値で継続）のいずれかを指定します。
Controller で仮想ユーザの値を割り当てる	(LoadRunner のみ)。仮想ユーザに手動でデータ・ブロックを割り当てるかどうかを指定します。Controller にブロック・サイズを自動的に割り当てさせるか，必要な値の数を指定します。[**ブロック サイズを自動的に割り当てる**] または [**仮想ユーザに x 件の値を割り当てる**] を選択します。後者の場合には，割り当てる値の数を指定します。 これを追跡するには，[実行環境設定] の [ログ] で，[**拡張ログ**] > [**パラメータ置換**] オプションを有効にします。十分なデータがない場合は，VuGen は仮想ユーザ・ログに「No more unique values for this parameter in table < *テーブル名* >」(< *テーブル名* > テーブルには，このパラメータのための一意の値がこれ以上ありません）という警告メッセージを表示します。
ファイル パス	使用するパラメータのデータを含む .dat ファイルを選択します。また，[**テーブルの作成**] ボタンを使用して新しいデータ・セットを作成することもできます。

テーブル・パラメータ

UI 要素	説明
	列をデータ・セットに追加します。
	行をデータ・セットに追加します。
	新しいデータ・テーブルを作成します。

UI 要素	説明
データ ウィザード(W)...	[データベース クエリ ウィザード] を開きます。ここで，既存のデータベースからデータをインポートできます。詳細については，次を参照してください。
列の削除(D)...	データ・セットからカラムを削除します。
行の削除(E)...	データ・セットから行を削除します。
メモ帳で編集(E)...	メモ帳でパラメータ値の表示と編集ができます。大規模データ・セットを使用する場合は，これが重要です。VuGen では，UI に表示されるのは最大で 100 行のみのためです。 メモ帳が開いて，1 行目にパラメータ名，2 行目にパラメータの元の値が表示されます。追加のカラム名と値をファイルに入力します。カンマやタブなどの区切り文字を使用してカラムの区切りを示します。テーブルの行ごと（新しいデータ行ごと）に改行します。 
Controller で仮想ユーザの値を割り当てる	(LoadRunner のみ)。仮想ユーザに手動でデータ・ブロックを割り当てるかどうかを指定します。Controller にブロック・サイズを自動的に割り当てさせるか，必要な値の数を指定します。[**ブロック サイズを自動的に割り当てる**] または [**仮想ユーザに x 件の値を割り当てる**] を選択します。後者の場合には，割り当てる値の数を指定します。 これを追跡するには，[実行環境設定] の [ログ] で，[**拡張ログ**] > [**パラメータ置換**] オプションを有効にします。十分なデータがない場合は，VuGen は仮想ユーザ・ログに「No more unique values for this parameter in table < *テーブル名* >」(< *テーブル名* > テーブルには，このパラメータのための一意の値がこれ以上ありません）という警告メッセージを表示します。

UI 要素	説明
カラム	どのカラムを使用するのかを指定します。または，［**すべて選択する**］を選択します。 1 つまたは複数のカラムを番号で指定するには，［**番号順に表示する**］を選択し，カラム番号をカンマまたはダッシュで区切って入力します。カラム番号とはデータを含んでいるカラムのインデックスです。たとえば，パラメータのデータがテーブルの最初のカラムにある場合は，1 を選択します。 ［**列の区切り文字**］ボックスで，カラムの区切り文字を選択します。区切り文字とは，テーブルのカラムを区切るために使用する文字です。カンマ（Comma），タブ（Tab），またはスペース（Space）を使用できます。
ファイル パス	使用するパラメータのデータを含む .dat ファイルを選択します。また，［**テーブルの作成**］ボタンを使用して新しいデータ・セットを作成することもできます。
ログ表示に使用する行区切り文字	この区切り文字は出力ログで行を区別するのに使用されます。パラメータ置換ログを有効にしている場合は，置換された値が再生ログに送信されます。再生ログ内の行区切り文字は，新しい行を示します。
行	➤ **［反復ごとの行数］**：反復ごとに使用する行の数。これは［**更新する対象**］フィールドを［**Each iteration**］（反復ごと）に設定したときのみ有効です。［**更新する対象**］が［**Once**］（一度）に設定されている場合は，すべての反復で同じ行が使用されます。 ➤ **［第 1 行目のデータ］**：スクリプトの実行中に使用するデータの 1 行目。ヘッダの後の最初の行から開始するには，「1」を入力します。 ➤ テーブル詳細(L)：使用可能なデータ行数など，テーブルに関する情報を表示します。
次の行を選択	仮想ユーザ・スクリプトの実行中にファイル・データを選択する方法。オプションには，［**Sequential**］（順次），［**Random**］（ランダム），または［**Unique**］（一意）があります。詳細については，269 ページの「ファイル / テーブル /XML タイプのパラメータにデータを割り当てる方法」を参照してください。

UI 要素	説明
更新する対象	パラメータを次の値に切り替えるタイミングを決定する方法。[**Each iteration**]（反復ごと），[**Each occurrence**]（出現ごと），[**Once**]（一度）から選択します。詳細については，269 ページの「ファイル / テーブル /XML タイプのパラメータにデータを割り当てる方法」を参照してください。
行数が不足する場合の対処法	テーブルに反復のための十分な行がない場合に VuGen が実行する処理を指定します。 **例**：設定するテーブルの行数は 3 ですが，データには 2 行しかありません。2 つの行のみ設定する場合は，[**パラメータは必要未満の行数を受け取ります**]を選択します。[**次の行を選択**]ボックスで指定した方法に従って次の行を反復処理して取得する場合は，[**「次の行を選択」の動作を使用する**]を選択します。
使用可能な値の終了後	一意のデータがなくなった場合に実行する処理を指定します。[**Abort the Vuser**]（仮想ユーザの中止），[**Continue in a cyclic manner**]（循環して継続），[**Continue with last value**]（最後の値で継続）のいずれかを指定します。

BPT パラメータ

BPT パラメータ・タイプは，HP Service Test ライセンスでのみ使用できます。詳細については，HP のサポートにお問い合わせください。

乱数パラメータ

UI 要素	説明
数字の形式	パラメータに含める最小桁数を指定します。**%01d** は 1 桁の数字を表し，**%02d** は 2 桁の数字を表します。
乱数の範囲	乱数値の最小と最大の範囲。

UI 要素	説明
サンプル値	選択した形式に基づいてサンプルのパラメータ値が表示されます。
更新する対象	➤ **[Each Occurrence]（出現ごと）**：スクリプト内でパラメータが出現するたびに新しい値を使用します。パラメータを使っているステートメントどうしが関連していない場合に有用です。たとえば，ランダムなデータを使用する場合には，パラメータが現れるたびに異なる値を使うと有効です。 ➤ **[Each iteration]（反復ごと）**：反復につき一度，パラメータを更新します。そのパラメータが 1 つのスクリプトに複数回現れる場合，仮想ユーザは 1 回の反復でそのパラメータが現れるたびに同じ値を使います。これは，パラメータを使うステートメントどうしが関連している場合に有用です。 **注**：独自に反復カウントを使って反復処理を行うアクション・ブロックを作成した場合，そこにパラメータが含まれていて，VuGen に反復ごとに値を更新するよう指定した場合でも，VuGen が対象とする反復は当該ブロックの反復ではなく，スクリプト全体のグローバルな反復なので，ブロックの反復ではパラメータが更新されません。 ➤ **[Once]（一度）**：シナリオの実行中に，パラメータ値を一度だけ更新します。仮想ユーザはすべての反復でパラメータのすべての出現箇所で同じパラメータ値を使います。このタイプは日付と時刻を使う場合に有用です。

一意の数のパラメータ

注：Controller でシナリオのスケジュールを設定する際，［**使用可能な値の終了後**］オプションは，スケジュール・ビルダの［継続時間］タブの［**(HH:MM:SS) の間実行する**］オプションに対してのみ適用されます。［**完了まで実行する**］オプションに対しては無視されるので注意してください。

UI 要素	説明
数字の形式	パラメータに含める最小桁数を指定します。**%01d** は 1 桁の数字を表し，**%02d** は 2 桁の数字を表します。
数字の範囲	➤ **［開始］**：開始の値。 ➤ **［仮想ユーザごとのブロック サイズ］**：各仮想ユーザに割り当てる一意の数の量。たとえば，開始値に 1 を指定し，ブロックサイズに 100 を指定すると，1 番目の仮想ユーザは 1 ～ 100 の値を使用でき，2 番目の仮想ユーザは 101 ～ 200 の値を使用できます。
サンプル値	選択した形式に基づいてサンプルのパラメータ値が表示されます。

UI 要素	説明
更新する対象	➤ **[Each Occurrence]（出現ごと）**：スクリプト内でパラメータが出現するたびに新しい値を使用します。パラメータを使っているステートメントどうしが関連していない場合に有用です。たとえば，ランダムなデータを使用する場合には，パラメータが現れるたびに異なる値を使うと有効です。 ➤ **[Each iteration]（反復ごと）**：反復につき一度，パラメータを更新します。そのパラメータが 1 つのスクリプトに複数回現れる場合，仮想ユーザは 1 回の反復でそのパラメータが現れるたびに同じ値を使います。これは，パラメータを使うステートメントどうしが関連している場合に有用です。 **注**：独自に反復カウントを使って反復処理を行うアクション・ブロックを作成した場合，そこにパラメータが含まれていて，VuGen に反復ごとに値を更新するよう指定した場合でも，VuGen が対象とする反復は当該ブロックの反復ではなく，スクリプト全体のグローバルな反復なので，ブロックの反復ではパラメータが更新されません。 ➤ **[Once]（一度）**：シナリオの実行中に，パラメータ値を一度だけ更新します。仮想ユーザはすべての反復でパラメータのすべての出現箇所で同じパラメータ値を使います。このタイプは日付と時刻を使う場合に有用です。
使用可能な値の終了後	仮想ユーザの値の範囲に達した場合に実行する処理を決定します。値の範囲は，開始値とブロック・サイズで決まります。 **[Abort Vuser]（仮想ユーザの中止）**：仮想ユーザ・スクリプトを終了します。 **[Continue in a cyclical manner]（循環して継続）**：この仮想ユーザの一意の数を，割り当てられた範囲の先頭から再度開始します。たとえば，1 ～ 100 の範囲の仮想ユーザの値が 100 に達すると，値は 1 から再び開始されます。 **[Continue with last value]（最後の値で継続）**：このパラメーターの以降のすべての出現箇所で，この値に割り当てられた最後の値を使用します。たとえば，1 ～ 100 の範囲の仮想ユーザの値が 100 に達すると，スクリプトの最後まで 100 の値が継続します。

ユーザ定義関数のパラメータ

UI 要素	説明
関数名	関数の名前。DLL ファイルに作成した関数名をそのまま使用します。
ライブラリ名	関連するライブラリ・ファイルの場所。
更新する対象	➤ **[Each Occurrence]（出現ごと）**：スクリプト内でパラメータが出現するたびに新しい値を使用します。パラメータを使っているステートメントどうしが関連していない場合に有用です。たとえば，ランダムなデータを使用する場合には，パラメータが現れるたびに異なる値を使うと有効です。 ➤ **[Each iteration]（反復ごと）**：反復につき一度，パラメータを更新します。そのパラメータが 1 つのスクリプトに複数回現れる場合，仮想ユーザは 1 回の反復でそのパラメータが現れるたびに同じ値を使います。これは，パラメータを使うステートメントどうしが関連している場合に有用です。 **注**：独自に反復カウントを使って反復処理を行うアクション・ブロックを作成した場合，そこにパラメータが含まれていて，VuGen に反復ごとに値を更新するよう指定した場合でも，VuGen が対象とする反復は当該ブロックの反復ではなく，スクリプト全体のグローバルな反復なので，ブロックの反復ではパラメータが更新されません。 ➤ **[Once]（一度）**：シナリオの実行中に，パラメータ値を一度だけ更新します。仮想ユーザはすべての反復でパラメータのすべての出現箇所で同じパラメータ値を使います。このタイプは日付と時刻を使う場合に有用です。

XML パラメータ

Web サービス の XML パラメータの詳細については，275 ページの「XML パラメータ」を参照してください。

［パラメータ シミュレーション］ダイアログ・ボックス

このダイアログ・ボックスでは，ファイル・パラメータの動作のシミュレーションを表示できます。

利用方法	**［パラメータ リスト］> パラメータを選択する > ［パラメータをシミュレートする］**
重要情報	➤ この機能はファイル・タイプのパラメータに対してのみ関連します。 ➤ すべての種類のパラメータ置換をシミュレートできるわけではありません。［**次の行を選択**］に［**同じ行 ...**］を選択するか，［**更新する対象**］に［**出現ごと**］を選択すると，［パラメータ シミュレーション］ダイアログ・ボックスは開きません。 ➤ VuGen では，最大で 256 回の反復および 256 個の仮想ユーザをシミュレートできます。 ➤ ［**無期限に実行する**］は Controller のスケジューラの［**実スケジュール**］に準拠しています。 ➤ ［パラメータ リスト］ダイアログの［**次の行を選択**］で［**Unique**］（一意）を選択した場合，各仮想ユーザに一意の範囲の行が割り当てられます。シミュレータはこの範囲の行からその仮想ユーザの値を代入します。 この設定を使う場合，［Controller で仮想ユーザの値を割り当てる］セクションの標準設定として［**ブロック サイズを自動的に割り当てる**］が選択されます。この場合，シミュレーションを実行すると，範囲割り当てはシナリオ実行モードの選択に従って実施されます。 標準設定の選択内容を［**仮想ユーザに X 件の値を割り当てる**］に変更すると，仮想ユーザは指定した数値を割り当てられ，シナリオ実行モードの選択は無視されます。

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
仮想ユーザ	シミュレーションで実行する仮想ユーザの数。
シナリオ実行モード	➤ **[完了まで実行する]**：実行する反復回数を入力するか，［**反復回数を実行環境の設定から取得する**］を選択します。 ➤ **[無期限に実行する]**：コントローラで無期限に実行するオプションをシミュレートします。VuGen は，実際には，指定された反復回数をシミュレートするだけです。
	パラメータ・シミュレーションを実行します。各パラメータ置換の値が表示されます。

例：

次の例では，［パラメータ リスト］ダイアログ・ボックスで次のように設定します。

➤［**新規パラメータの値**］：値 1 から値 7

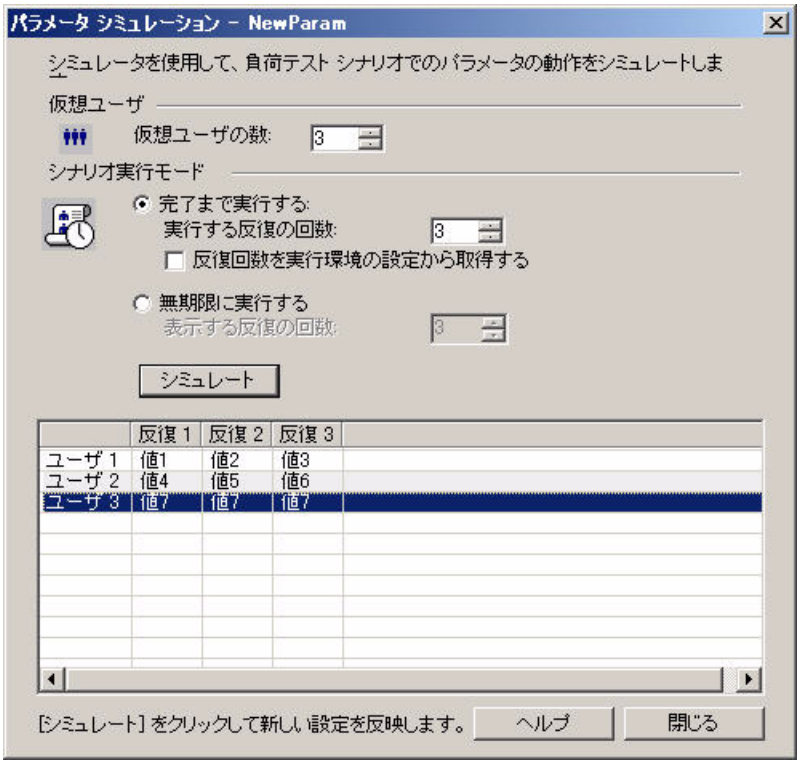
➤［**次の行を選択**］：一意

➤［**使用可能な値の終了後**］：［Continue with last value］（最後の値で継続）：

➤［**Controller で仮想ユーザの値を割り当てる**］：［ブロック サイズを自動的に割り当てる］

[シナリオ実行モード]：[完了まで実行する]

次の例では，ユーザは 3 つの仮想ユーザを選択し，シナリオ実行モードを［**完了まで実行する**］に設定して 3 回の反復を選択しました。



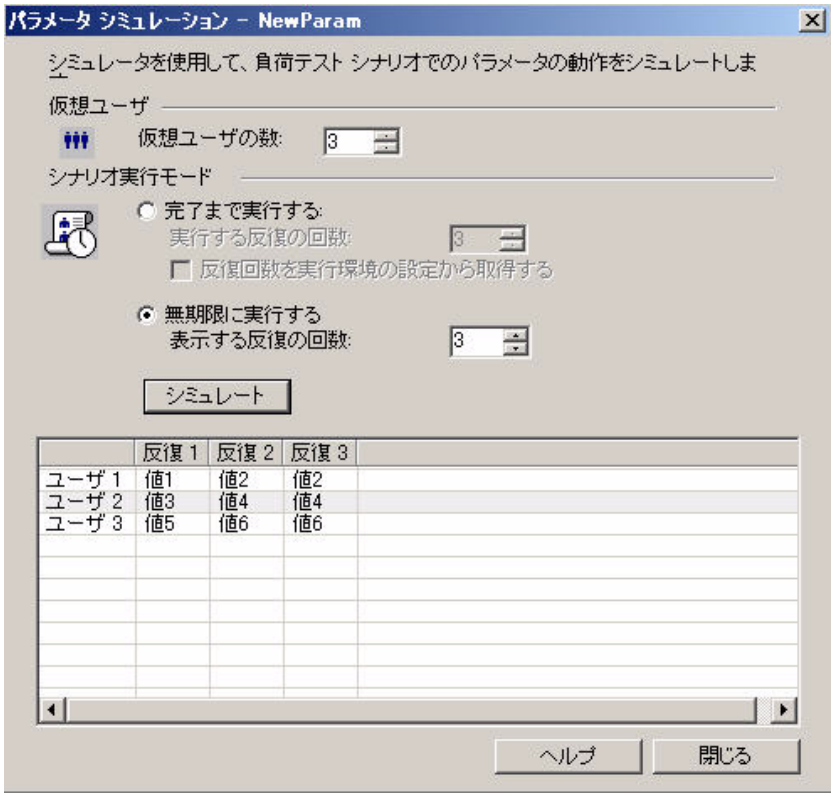
シナリオ実行モードを［**完了まで実行する**］に設定すると，各仮想ユーザが受け取る行の数は反復の数と同じになります。テーブルに十分な行がなくなると，範囲割り当ては停止します。

シミュレーションが実行されると，最初の仮想ユーザは最初の 3 つの値を取得します（これが反復数です）。2 番目の仮想ユーザは次の 3 つの値を取得します。3 番目の仮想ユーザは，最初の反復において残りの値を取得します。残りの反復については，［パラメータ リスト］ダイアログ・ボックスの［**使用可能な値の終了後**］オプションは［**Continue with last value**］（最後の値で継続）に設定されているため，3 番目の仮想ユーザは同じ値で続行します。

4 番目の仮想ユーザは失敗します。

［シナリオ実行モード］:［無期限に実行する］

次の例では，ユーザは 3 つの仮想ユーザを選択し，シナリオ実行モードを［無期限に実行する］に設定して 3 回の反復が表示されるように選択しています。



シナリオ実行モードを［無期限に実行する］に設定すると，各仮想ユーザの割り当て範囲は，.dat ファイル内のセルの数を仮想ユーザ数で割ることによって計算されます。このシナリオでは，7/3 = 2 です（シミュレータは，最も近い最小の整数を取得します）。

シミュレーションを実行すると，最初の仮想ユーザは値 1 と値 2 を取得します。2 番目の仮想ユーザは値 3 と値 4 を取得し，3 番目の仮想ユーザは値 5 と値 6 を取得します。仮想ユーザは 3 つしかないため，値 7 は振り分けられません。

注：テーブルの最初のカラムのセル上にマウスを移動すると，ツールチップに，その仮想ユーザに割り当てられ値に関する情報が表示されます。

値が割り当てられていないセル上にマウスを移動すると，ツールチップに，値が割り当てられなかった理由が表示されます。

適切な値が割り当てられていない場合は，ツールチップは表示されません。

［パラメータ リスト］ダイアログ・ボックス

このダイアログ・ボックスでは，パラメータの表示，作成，削除，選択，および変更が行えます。パラメータ・リストには，作成したパラメータがすべて表示されます。これには，入力パラメータと出力パラメータの両方が含まれます。

利用方法	**［仮想ユーザ］>［パラメータ リスト］**
重要情報	unique というパラメータ名は使用しないでください。VuGen によってすでに使用されています。

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
新規作成(N)	新しいパラメータを作成します。強調表示されたテキストはパラメータで置き換えられません。*unique* というパラメータ名は使用しないでください。VuGen によってすでに使用されています。
削除(D)	選択したパラメータを削除します。

UI 要素	説明
<［パラメータのプロパティ］表示枠>	この表示枠の表示は，使用するパラメータのタイプにより異なります。この表示枠の詳細については，次のいずれかのセクションを参照してください。 ➤ 日時，グループ名，反復回数，Load Generation 名，仮想ユーザ ID のパラメータについては，292 ページの「日時，グループ名，反復回数，Load Generation 名，仮想ユーザ ID のパラメータ」を参照してください。 ➤ ファイル・パラメータについては，294 ページの「ファイル・パラメータ」を参照してください。 ➤ テーブル・パラメータについては，296 ページの「テーブル・パラメータ」を参照してください。 ➤ BPT パラメータについては，299 ページの「BPT パラメータ」を参照してください。 ➤ 乱数パラメータについては，299 ページの「乱数パラメータ」を参照してください。 ➤ 一意の数のパラメータについては，301 ページの「一意の数のパラメータ」を参照してください。 ➤ ユーザ定義関数のパラメータについては，303 ページの「ユーザ定義関数のパラメータ」を参照してください。 ➤ XML パラメータについては，304 ページの「XML パラメータ」を参照してください。
パラメータのタイプ	このドロップダウン・リストでは，パラメータ・タイプを選択できます。各種パラメータ・タイプの詳細については，266 ページの「パラメータ・タイプ」を参照してください。

データベース・クエリ・ウィザード

このウィザードを使用すると，既存のデータベースからパラメータ用のデータを選択できます。

利用方法	**パラメータを右クリック > [パラメータのプロパティ] > [データ ウィザード]（ファイル / テーブル・タイプのパラメータに対してのみ使用できます）**
関連タスク	287 ページの「データベースからパラメータ・データをインポートする方法」

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
クエリの定義	次のオプションのどちらかを選択します。 ➤ Microsoft Query でクエリを作成する ➤ SQL ステートメントを手動で指定する
Microsoft Query の使い方を表示する	次をクリックした後に，Microsoft Query の使用方法を表示します。
最大行数	指定したクエリに基づいて .dat ファイルに作成する行の最大数。

トラブルシューティングと制限事項

このセクションでは，パラメータのトラブルシューティングと制限事項について説明します。

関数の引数の制限事項

パラメータの対象にできるのは，関数内の引数だけです。関数の引数以外の文字列はパラメータ化できません。また，すべての関数の引数をパラメータ化できるわけでもありません。パラメータ化できる引数については，*オンライン関数リファレンス*（[**ヘルプ**] > [**関数リファレンス**]）で各関数を参照してください。

lrd_stmt 関数を例にとってみます。この関数の構文は次のとおりです。

```
lrd_stmt (LRD_CURSOR FAR *mptCursor, char FAR *mpcText, long mliTextLen,
LRDOS_INT4 mjOpt1, LRDOS_INT4 mjOpt2, int miDBErrorSeverity);
```

*オンライン関数リファレンス*によれば，パラメータ化できるのは *mpcText* 引数だけです。

記録された **lrd_stmt** 関数が次のようになっていたとします。

```
lrd_stmt(Csr4, "select name from sysobjects where name =¥"Kim¥" ", -1, 148, -99999,
0);
```

これは次のようにパラメータ化できます。

```
lrd_stmt(Csr4, "select name from sysobjects where name =¥"<name>¥" ", -1, 148,
-99999, 0);
```

注：lr_eval_string 関数を使えば，標準ではパラメータ化できない関数の引数でも「パラメータ化」することができます。さらに，**lr_eval_string** 関数を使えば，仮想ユーザ・スクリプトの任意の文字列も「パラメータ化」できます。

VB，COM，および Microsoft .NET プロトコルには，パラメータを定義するのに **lr.eval string** 関数を使用する必要があります。例：

```
lr.eval_string("{Custom_param}")
```

lr_eval_string 関数の詳細については，*オンライン関数リファレンス*を参照してください。

10

記録オプション

本章の内容

概念

リファレンス

概念

ポート・マッピングの概要

ソケット・レベルでネットワーク・トラフィックを記録する仮想ユーザ・スクリプト（HTTP，SMTP，POP3，FTP，IMAP，Oracle NCA，Windows Sockets）を記録する場合，［ポートの割り当て］オプションを設定できます。これらのオプションで，特定のサーバとポートの組み合わせを通じて受け取るトラフィックを，特定の通信プロトコルに割り当てることができます。

割り当ての対象にできる通信プロトコルは，FTP，HTTP，IMAP，NCA，POP3，SMTP および SOCKET です。割り当ては，サーバ名，ポート番号，または「サーバ：ポート」の組み合わせを指定することで作成できます。たとえば，*twilight* というサーバのポート 25 番からのトラフィックをすべて SMTP として扱うよう指定できます。また，*viper* というサーバからのすべてのトラフィックを，ポートに関係なく FTP プロトコルへ割り当てることもできます。さらには，サーバ名に関係なく，ポート 23 のすべてのトラフィックを SMTP に割り当てることも可能です。

マルチ・プロトコル・モードで記録する場合には，少なくとも 1 つのプロトコルがソケット・レベルで記録していると，［ポート マッピング］ノードが利用できます。ただし，シングル・プロトコル・スクリプトで HTTP または WinSock を記録する場合は例外です。

ポート・マッピングの自動検出

VuGen の高度なポート割り当てオプションでは，**自動検出**オプションを設定できます。VuGen の自動検出では，サーバに送られるデータが解析されます。さらに，シグネチャ・データやデータのパターンが調べられ，プロトコルが特定されます。シグネチャを検出するため，最初の受信バッファまで，すべての送信バッファが蓄積されます。受信バッファが返されるまでに送信されたすべての送信バッファは単一のデータ**遷移**とみなされます。標準設定では，割り当ては定義されず，VuGen は自動検出を行います。一部のプロトコル（HTTP など）は，単一の遷移のなかでタイプが判別されます。ほかのネットワーク・プ

ロトコルでは，タイプを判別されるまでにいくつかの遷移が必要です。このため，VuGen ではサーバとポートの組み合わせごとに一時バッファが作成されます。VuGen によって最初の遷移バッファが読み取られてもプロトコル・タイプが判別できない場合は，データが一時バッファに格納されます。そして，プロトコルを特定できるシグネチャが検出されるまで，着信バッファの読み取りが継続されます。

標準では，VuGen がプロトコルのシグネチャの検出に使用できるトランザクションは 4 つ，バッファは最大 2048 バイトまでです。VuGen が最大トランザクションに達するか，バッファ・サイズの上限に達してもプロトコルを特定できなかった場合，データは WinSock プロトコルに割り当てられます。VuGen に（マルチ・プロトコルを選択している場合）WinSock プロトコルを記録するように設定していない場合には，VuGen によってデータが破棄されます。

プロトコルのタイプを検出するために VuGen が読み取るバッファの最大サイズを変更することができます。また一時バッファのサイズを指定することも可能です。最初の送信バッファに格納されたデータの合計が，一時バッファのサイズより大きくなった場合，VuGen ではプロトコル・タイプの自動検出が行えなくなります。この場合には一時バッファのサイズを増やす必要があります。

前述のネットワーク・レベル・プロトコルを使用する場合は，VuGen がプロトコル・タイプを判別するよう，自動検出オプションをオンにした設定を推奨します。ほとんどの場合，VuGen のレコーダでは，これらのプロトコルのシグネチャが認識できます。そして，プロトコルの仕様に従ってプロトコルが自動的に処理されます。ただし，VuGen でプロトコルが認識されないこともあります。次に例を示します。

- 既存のプロトコルに類似するプロトコル・シグネチャがあり，誤った処理が行われた場合。
- プロトコルに一意のシグネチャがない場合。
- プロトコルが SSL による暗号を使用しているため，WinSock レベルで認識されない場合。

上記のどの場合も，プロトコルをホストするサーバとポートを一意に識別する情報を提供することができます。

EUC エンコーディング（日本語版 Windows のみ）

Windows の標準文字セット以外を扱う場合には，コード変換が必要になることがあります。文字セットは，文字の集合と整数の集合との対応関係を表します。この対応関係によって，与えられた 1 つの文字について，整数との一意の組み合わせが成立します。EUC（Extended UNIX Code，拡張 UNIX コード）と SJIS（Shift Japan Industry Standard，シフト JIS）は，Windows の標準文字セットではなく，Web サイトで日本語文字を表示するために使用されます。

Windows では SJIS コードを使いますが，UNIX では EUC コードを使います。Web サーバの実行環境が UNIX で，クライアントの実行環境が Windows の場合，コードが違うため，Web サイトの文字がクライアント・マシンで正しく表示されません。このことは，EUC コードの日本語文字を仮想ユーザ・スクリプトで表示するときに影響します。

記録中，VuGen は HTTP ヘッダーを通じて Web ページで使われているコードを検出します。文字セットに関する情報が HTTP ヘッダーに存在しない場合は，HTML のメタ・タグを調べます。

それでも Web ページで EUC コードが使われていることが事前にわかっていれば，記録オプションを使用して正しいコードを使用するよう VuGen に指示できます。ページを EUC コードで記録するには，［記録オプション］の［**記録**］ノードの［**EUC**］オプションを有効にします（日本語版 Windows でのみ表示されます）。

［**EUC**］オプションを有効にすると，EUC コードで書かれていない Web ページも強制的に EUC コードで記録されます。したがって，このオプションを有効にする必要があるのは，VuGen が HTTP ヘッダまたは HTML のメタ・タグからコードを検知できない場合か，ページが EUC コードで書かれていることが事前にわかっている場合だけです。

記録中，VuGen は EUC コードの文字列を Web サーバから受信すると，それを SJIS に変換します。変換された SJIS 文字列はスクリプトの **Action** 関数に保存されます。しかし，再生を正常に実行するためには，文字列を再び EUC に変換してから Web サーバに送り返す必要があります。このため，VuGen は **Action** 関数の前に **web_sjis_to_euc_param** 関数を追加します。この関数で SJIS 文字列を EUC に再変換します。

次の例では，ユーザが EUC で書かれた Web ページに移動してリンクをクリックします。VuGen は **Action** 関数を記録し，**web_sjis_to_euc_param** 関数を **Action** 関数の前のスクリプトに追加します。

```
web_sjis_to_euc_param("param_link","Search");
web_link("LinkStep","Text={param_link}");
```

詳細については，355 ページの「[URL 詳細設定] ダイアログ・ボックス」を参照してください。

スクリプト生成のプリファレンスの概要

VuGen では，セッションを記録する前に，スクリプト生成言語を指定できます。使用可能なスクリプト生成言語は，プロトコルによって異なります。使用可能な言語としては，C，C#，Visual Basic，Visual Basic .NET，VB Script，および JavaScript です。標準設定では，対象プロトコルに応じた最も一般的な言語でスクリプトが生成されますが，この言語は［**スクリプト**］記録オプション・ノードで変更できます。

ユーザ・インタフェースの詳細については，358 ページの「[一般] の［スクリプト］ノード」を参照してください。

ヒント：ある言語でスクリプトを記録した後に，別の言語でそのスクリプトを再生成することもできます。タスクの詳細については，119 ページの「仮想ユーザ・スクリプトの再生成方法」を参照してください。

生成言語の選択後，スクリプトに何を含め，それをどのように生成するかをレコーダに指示する，言語固有の記録オプションを有効にできます。

記録するプロトコルの少なくとも 1 つにマルチ・プロトコルの機能がある場合は，［スクリプト］ノードを使用できます。ただし，シングル・プロトコル・スクリプトで HTTP または WinSock を記録する場合は例外です。

スクリプト言語オプション

セッションを記録するときに，VuGen は，ユーザのアクションをエミュレートするスクリプトを作成します。標準のスクリプト生成言語は C 言語または C# 言語（MS .NET の場合）です。FTP，COM/DCOM，およびメール・プロトコル（IMAP，POP 3，SMTP）の場合は，Visual Basic，VBScript，および JavaScript でもスクリプトを生成できます。次のリストに各言語に適しているプロトコルを示します。

- **C**：複雑な COM 構成要素と C++ オブジェクトを使用するアプリケーション用。
- **C #**：複雑なアプリケーションと環境を使用するアプリケーション用（MS .NET プロトコルのみ）。
- **Visual Basic .NET**：VB のすべての機能を使用する VB .NET アプリケーション用。
- **Visual Basic for Applications**：（VBScript とは異なり）VB のすべての機能を使用する VB ベースのアプリケーション用。
- **Visual Basic Scripting**：VBScript ベースのアプリケーション用（ASP など）。
- **Java Scripting**：JavaScript ベースのアプリケーション用（**js** ファイルやダイナミック HTML アプリケーションなど）。

記録セッションの後は，通常の C，C#，Visual Basic，VB Script，JavaScript コード，または制御フロー・ステートメントを使ってスクリプトに変更を加えられます。

記録レベルの概要

［**記録オプション**］ダイアログ・ボックスの［**一般：記録**］ノードで記録レベルを選択することによって，仮想ユーザ・スクリプトの生成時に，記録する情報と使用する関数を指定できます。記録レベルは，目的または環境に応じて選択します。指定できるレベルは，［**GUI ベースのスクリプト**］，［**HTML ベースのスクリプト**］，および［**URL ベースのスクリプト**］です。ユーザ・インタフェースの詳細については，354 ページの「［一般］の［記録］ノード」を参照してください。

3 つの記録レベルを使用したスクリプト例を次に示します。

GUI ベースのスクリプト

HTML アクションが **web_text_link** などのコンテキスト・センシティブ GUI 関数として記録されます。

```
/* GUI ベース・モード：JavaScript をサポートしている CS タイプ関数 */
vuser_init()
{
web_browser("WebTours",
        DESCRIPTION,
        ACTION,
        "Navigate=http://localhost:1080/WebTours/",
        LAST);

web_edit_field("username",
        "Snapshot=t2.inf",
        DESCRIPTION,
        "Type=text",
        "Name=username",
        "FrameName=navbar",
        ACTION,
        "SetValue=jojo",
        LAST);
…
```

HTML ベースのスクリプト

HTML ユーザ・アクションごとに個別のステップが生成されます。これらのステップはわかりやすいのですが，JavaScript コードの忠実なエミュレーションが反映されているわけではありません。

```
/* HTML ベース・モード : ユーザ・アクションを記述するスクリプト */
...
web_url("WebTours",
        "URL=http://localhost/WebTours/",
        "Resource=0",
        "RecContentType=text/html",
        "Referer=",
        "Snapshot=t1.inf",
        "Mode=HTML",
        LAST);

web_link("Click Here For Additional Restrictions",
        "Text=Click Here For Additional Restrictions",
        "Snapshot=t4.inf",
        LAST);

web_image("buttonhelp.gif",
        "Src=/images/buttonhelp.gif",
        "Snapshot=t5.inf",
        LAST);
…
```

URL ベースのスクリプト

ユーザのアクションによって送信されたサーバからのすべてのブラウザ要求とリソースが記録されます。自動的にあらゆる HTTP リソースが URL ステップ（**web_url** ステートメント）として記録されます。ブラウザの通常の記録では，URL ベース・モードは相関関連の問題が起こりやすいため推奨されません。ただし，アプレットや非ブラウザ・アプリケーションなどのページを記録している場合は，このモードが最適です。

URL ベースのスクリプトは，HTML ベースのスクリプトほどわかりやすくはありません。これは，すべてのアクションが **web_link** や **web_image** などではなく **web_url** のステップとして記録されるためです。

```
/* URL ベース・モード：web_url 関数のみ */
…
web_url("spacer.gif",
        "URL=http://graphics.hplab.com/images/spacer.gif",
        "Resource=1",
        "RecContentType=image/gif",
        "Referer=",
        "Mode=HTTP",
        LAST);

web_url("calendar_functions.js",
        "URL=http://www.im.hplab.com/travelp/calendar_functions.js",
        "Resource=1",
        "RecContentType=application/x-javascript",
        "Referer=",
        "Mode=HTTP",
        LAST);
…
```

マルチ・プロトコル・スクリプトを記録していなければ，記録中に記録レベルと詳細記録オプションを切り替えることができます。記録レベルを切り替える方法は，パフォーマンスのテストを行う上級ユーザ向けです。

記録後，元の記録とは異なる方式でスクリプトを再生成することもできます。たとえば，スクリプトを HTML ベース・レベルで記録している場合，URL ベース・レベルで再生成できます。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，［**オプション**］をクリックして，再生成のための記録オプションを設定します。

シリアル化の概要

VuGen は記録中に未知のオブジェクトに遭遇した場合，オブジェクトがシリアライズをサポートしていればシリアライズを使用します。未知のオブジェクトは，たとえばフィルタに包含されていなかったために，その生成が記録されなかった入力引数などが考えられます。シリアライズによって，未知の引数をメソッドに渡すことによるコンパイルエラーがなくなります。オブジェクトがシリアル化される場合，カスタム・フィルタを設定してこのオブジェクトを記録することをお勧めします。

イベントのリッスンと記録を行うためのヒント

理想的なリッスンと記録の設定を見つけるのが困難な場合があります。これらの設定を行うときには，次のガイドラインに留意します。

➤ オブジェクトのイベントを記録するには，VuGen がイベントをリッスンし，イベントが生じたときにそれを記録するように指定します。子プロジェクトのイベントは，親オブジェクトにそのイベントに対するハンドラまたは動作が含まれている場合にもリッスンできます。また，親オブジェクトのイベントは，子オブジェクトにそのイベントに対するハンドラまたは動作が含まれていても，リッスンできます。

ただし，ソース・オブジェクト（どの親オブジェクトがハンドラまたは動作を含んでいるかによらずイベントが実際に発生したオブジェクト）に対するイベントは記録しなければなりません。

たとえば，**onmouseover** イベント・ハンドラのあるテーブル・セルに 2 つのイメージが含まれているとします。ユーザがマウス・ポインタでどちらかのイメージに触れると，バブリングによってイベントがそのセルに送られます。このバブリングにはどちらのイメージが実際に触れられたかの情報が含まれています。この mouseover イベントは，次のようにして記録できます。

- WebTable の mouseover イベントに対する**応答**を［**ハンドラの場合**］に設定する一方で（イベントが生じたときに VuGen にイベントが「聞こえる」ように），イベントは記録しないようにした上で，Image の mouseover イベントに対する**応答**は［**しない**］に設定する一方で，Image に対する記録は［**有効**］（WebTable レベルでリッスンした後の画像に対する mouseover イベントを記録します）に設定します。
- Image の mouseover イベントに対する**応答**を［**常に**］（イメージ・タグが動作またはハンドラを含んでいなくても mouseover イベントをリッスン）に設定し，Image オブジェクトに対する記録ステータスを［**有効**］（イメージに対する mouseover イベントを記録）に設定します。

- 多数のオブジェクト上の多数のイベントをリッスンするように設定すると，VuGen のパフォーマンスが低下することがあるので，リッスンの設定は，必要なオブジェクトに限定するようにします。
- まれに，イベントが発生したオブジェクト（ソース・オブジェクト）をリッスンしていると，イベントが妨害されることがあります。

リファレンス

Click and Script のコンテキスト外の記録の例

次の例は，コンテキスト外の記録オプションを有効にしてスクリプトを再生成したものです。

```
web_image_link("Search Flights Button",
        "Snapshot=t5.inf",
        DESCRIPTION,
        "Alt=Search Flights Button",
        "FrameName=navbar",
        ACTION,
        "ClickCoordinates=58,9",
        LAST);

web_add_cookie("MSO=SID&1141052844; DOMAIN=localhost");

web_add_cookie("MTUserInfo=hash&47&firstName&Joseph&expDate&%0A&creditCa
rd&&address1&234%20Willow%20Drive&lastName&Marshall%0A&address2&San%2
0Jose%2FCA%2F94085&username&jojo; DOMAIN=localhost");

web_url("FormDateUpdate.class",
        "URL=http://localhost:1080/WebTours/FormDateUpdate.class",
        "Resource=0",
        "RecContentType=text/html",
        "Referer=",
        "UserAgent=Mozilla/4.0 (Windows 2000 5.0) Java/1.4.2_08",
        "Mode=HTTP",
        LAST);
…
```

このオプションを無効にすると，VuGen は，ActiveX コントロールおよび Java アプレットのコードを生成しません。次の例では，**web_image_link** 関数のみが生成され，クラス・ファイルが含まれる **web_url** 関数は生成されません。

```
web_image_link("Search Flights Button",
        "Snapshot=t5.inf",
        DESCRIPTION,
        "Alt=Search Flights Button",
        "FrameName=navbar",
        ACTION,
        "ClickCoordinates=58,9",
        LAST);
```

詳細については，361 ページの「[GUI プロパティ］の［詳細］ノード」を参照してください。

プロトコルの互換性に関する表

次の表に，仮想ユーザ・スクリプトのタイプと各タイプで使用できる［記録オプション］のノードを示します。

プロトコル	［記録オプション］ノード
AMF	➤ 一般 - スクリプト，プロトコル，記録 ➤ AMF - コードの生成 ➤ ネットワーク - ポート マッピング ➤ HTTP プロパティ - 詳細，相関
AJAX	➤ 一般 - スクリプト，記録 ➤ GUI プロパティ - 詳細，Web イベントの定義 ➤ ネットワーク - ポート マッピング ➤ HTTP プロパティ - 詳細，相関
C 仮想ユーザ	➤ なし
Citrix	➤ 一般 - スクリプト ➤ Citrix - 設定，レコーダ，コード生成，ログイン

プロトコル	［記録オプション］ノード
COM/DCOM	➤ 一般 - スクリプト ➤ COM/DCOM - フィルタ，オプション
DB2 CLI	➤ 一般 - スクリプト ➤ データベース - データベース
DNS	➤ なし
EJB	➤ Java 環境の設定 - Classpath ➤ EJB オプション - コード生成オプション
FTP	➤ 一般 - スクリプト ➤ ネットワーク - ポート マッピング
Flex	➤ 一般 - スクリプト，プロトコル，記録 ➤ Flex - 設定，外部オブジェクト ➤ ネットワーク - ポート マッピング ➤ HTTP プロパティ - 詳細，相関
i モード	➤ 一般 - 記録 ➤ HTTP プロパティ - i モード ツールキット，詳細，相関
Informix	➤ 一般 - スクリプト ➤ データベース - データベース
IMAP	➤ 一般 - スクリプト ➤ ネットワーク - ポート マッピング
Java over HTTP	➤ 一般 - 記録 ➤ Java 環境の設定 - Java VM，Classpath ➤ トラフィック アナリシス - トラフィック フィルタ ➤ ネットワーク - ポート マッピング ➤ HTTP プロパティ - ブラウザ，記録用プロキシ，詳細，相関
Java Record Replay	➤ Java 環境の設定 - Java VM，Classpath ➤ 記録のプロパティ - レコーダ オプション，シリアル化オプション，相関オプション，ログ オプション，Corba オプション
Javascript 仮想ユーザ	➤ なし

プロトコル	[記録オプション] ノード
LDAP	➤ 一般 - スクリプト
MMS（Media Player）	➤ なし
.NET	➤ 一般 - スクリプト ➤ .NET - 記録，フィルタ
RDP	➤ 一般 - スクリプト ➤ RDP - ログイン，コード生成（基本，詳細，およびエージェント） ➤ ネットワーク - ポート マッピング
MAPI	➤ なし
MS SQL Server	➤ 一般 - スクリプト ➤ データベース - データベース
MMS（マルチメディア・メッセージング・サービス）	➤ なし
ODBC	➤ 一般 - スクリプト ➤ データベース - データベース
Oracle（2 層）	➤ 一般 - スクリプト ➤ データベース - データベース
Oracle NCA	➤ 一般 - スクリプト，プロトコル，記録 ➤ ネットワーク - ポート マッピング ➤ HTTP プロパティ - 詳細，相関
Oracle Web Applications 11i	➤ 一般 - スクリプト，記録 ➤ GUI プロパティ - 詳細，Web イベントの定義 ➤ ネットワーク - ポート マッピング ➤ HTTP プロパティ - 詳細，相関
PeopleSoft Enterprise	➤ 一般 - スクリプト，記録 ➤ GUI プロパティ - 詳細，Web イベントの定義 ➤ ネットワーク - ポート マッピング ➤ HTTP プロパティ - 詳細，相関

プロトコル	［記録オプション］ノード
PeopleSoft Tuxedo	➤ なし
POP3	➤ 一般 - スクリプト ➤ ネットワーク - ポート マッピング
Real	➤ 一般 - スクリプト
SAP Web	➤ 一般 - スクリプト，記録 ➤ ネットワーク - ポート マッピング ➤ HTTP プロパティ - 詳細，相関
SAP Click and Script	➤ 一般 - スクリプト，記録 ➤ GUI プロパティ - 詳細，Web イベントの定義 ➤ ネットワーク - ポート マッピング ➤ HTTP プロパティ - 詳細，相関
SAPGUI	➤ 一般 - スクリプト ➤ SAPGUI - 一般，コード生成，自動ログオン
Siebel Web	➤ 一般 - スクリプト，プロトコル，記録 ➤ ネットワーク - ポート マッピング ➤ HTTP プロパティ - 詳細，相関
Silverlight	➤ 一般 - スクリプト，プロトコル，記録 ➤ データ形式拡張機能 - 設定，ヘッダ チェーン，ボディ チェーン，Cookie チェーン，クエリ文字列チェーン ➤ Silverlight - サービス ➤ ネットワーク - ポート マッピング ➤ HTTP プロパティ - 詳細，相関
SMTP	➤ 一般 - スクリプト ➤ ネットワーク - ポート マッピング
Sybase CTlib / DBlib	➤ 一般 - スクリプト ➤ データベース - データベース
RTE	➤ RTE - 設定，RTE
Tuxedo，Tuxedo 6	➤ なし

プロトコル	［記録オプション］ノード
VB スクリプト仮想ユーザ	➤ なし
VB 仮想ユーザ	➤ なし
WAP	➤ 一般 - スクリプト，プロトコル，記録 ➤ ネットワーク - ポート マッピング ➤ HTTP プロパティ - 詳細，相関
Web（Click and Script）	➤ 一般 - スクリプト，記録 ➤ GUI プロパティ - 詳細，Web イベントの定義 ➤ ネットワーク - ポート マッピング ➤ HTTP プロパティ - 詳細，相関
Web（HTTP/HTML）	➤ 一般 - スクリプト，プロトコル，記録 ➤ ネットワーク - ポート マッピング ➤ HTTP プロパティ - 詳細，相関 ➤ データ形式拡張機能 - 設定，ヘッダ チェーン，ボディ チェーン，Cookie チェーン，クエリ文字列チェーン
Web サービス	➤ 一般 - スクリプト，プロトコル，記録 ➤ トラフィック アナリシス - トラフィック フィルタ ➤ ネットワーク - ポート マッピング ➤ HTTP プロパティ - 詳細，相関
Windows Sockets	➤ ソケット - Winsock

［記録オプション］のユーザ・インタフェース

このセクションの内容

［AMF］の［コードの生成］ノード

AMF プロトコルのコード生成を設定できます。

利用方法	**［ツール］>［記録オプション］>［AMF］>［コード生成］**
重要情報	このノードは特定のプロトコルにのみ適用されます。プロトコルの完全な一覧と各プロトコルに関連付けられたノードについては，325 ページの「プロトコルの互換性に関する表」を参照してください。

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
LoadRunner パーサを使用して外部オブジェクトをエンコード	Flex アプリケーションを記録する際，状況によっては，VuGen で外部オブジェクトをデコードできないことがあります。これは，Flex 2 アプリケーションで使用されている独自のエンコード・スキームによるものです。 この問題を克服するために，VuGen は，外部オブジェクトに遭遇したときに，未解析の AMF3 バイナリ・データが含まれるユーザ定義要求関数を生成します。 これらのオブジェクトは，LoadRunner パーサを使用してエンコードすることもできます。これにより，すべての外部オブジェクトが標準の AMF3 オブジェクトとしてデコードされ，**amf_call** 関数が生成されます。ただし，解析できないオブジェクトはデコードされません。したがって，生成されたスクリプトをチェックして，解析が有効であったことを確認する必要があります。また，このオプションを有効にすると，コード生成の安定性が低下することがあります。 **標準設定値**：有効。

［Citrix］の［コード生成］ノード

記録時に VuGen が情報をキャプチャする方法を設定できます。

利用方法	［ツール］>［記録オプション］>［Citrix］>［コード生成］
重要情報	➤ このノードは特定のプロトコルにのみ適用されます。プロトコルの完全な一覧と各プロトコルに関連付けられたノードについては，325 ページの「プロトコルの互換性に関する表」を参照してください。 ➤ 記録中に手動で追加したテキストの同期化ステップは，前述の設定の影響を受けません。前述のオプションを無効にしても，手動で追加したステップはスクリプトに現れます。 ➤ 同期化オプションは，スクリプトの再生成にも有効です。たとえば，当初［**テキスト同期化呼び出しを自動的に生成する**］を無効にしてスクリプトを記録した場合でも，テキストの同期化が含まれるように記録後にスクリプトを再生成できます。

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
Citrix エージェント入力をコード生成に使用する	Citrix エージェント入力を使用して，同期化関数が追加されたより説明的なスクリプトが生成されます。 **標準設定値**：有効。 ➤ **[Add text synchronization calls]**：各マウス・クリックの前に，テキストの同期化を行う［**テキストの同期**］ステップが追加されます。 **標準設定値**：有効。

［Citrix］の［設定］ノード

記録セッション中の Citrix クライアントに対するウィンドウ・プロパティと暗号化設定を設定できます。

利用方法	**［ツール］>［記録オプション］>［Citrix］>［設定］**
重要情報	このノードは特定のプロトコルにのみ適用されます。プロトコルの完全な一覧と各プロトコルに関連付けられたノードについては，325 ページの「プロトコルの互換性に関する表」を参照してください。

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
暗号レベル	ICA 接続の暗号化レベル：［**基本**］，［**128 ビット（ログインのみ）**］，［**40 ビット**］，［**56 ビット**］，［**128 ビット**］，または［**サーバ標準設定値を使用**］から選択します。
ウィンドウのサイズ	クライアント・ウィンドウのサイズ。 **標準設定値**：800 x 600。

［Citrix］の［ログイン］ノード

記録セッションに対する接続情報とログイン情報を設定できます。

利用方法	**［ツール］>［記録オプション］>［Citrix］>［ログイン］**
重要情報	➤ このノードは特定のプロトコルにのみ適用されます。プロトコルの完全な一覧と各プロトコルに関連付けられたノードについては，325 ページの「プロトコルの互換性に関する表」を参照してください。 ➤ ログオン情報を省略すると，指定したサーバがクライアントによって検出されたときに，情報を入力するよう求められます。

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
接続	➤ **[ネットワークのプロトコル]**：TCP/IP や TCP/IP+HTTP のプロトコルが適しています。TCP/IP は，ほとんどの Citrix サーバでサポートされていますが，11.2 で始まる Citrix クライアントではサポートされていません。しかし，一部のサーバは，ある決まった HTTP ヘッダを持つ TCP/IP のみ許可するように，管理者によって設定されています。通信上の問題が発生した場合は，TCP/IP + HTTP オプションを選択します。 ➤ **[サーバ]**：Citrix サーバの名前。新しいサーバをリストに追加するには，［**追加**］をクリックし，サーバ名（およびサーバの TCP/IP + HTTP 用ポート）を入力します。 **注**：複数のサーバが適用されるのは［発行されたアプリケーション］を指定したときのみです。特定のアプリケーションのないデスクトップに接続しようとしている場合は，1 つのサーバのみが表示されます。 ➤ **[発行されたアプリケーション]**：Citrix サーバで認識される，**発行されたアプリケーション**の名前。使用可能なアプリケーションのリストがドロップダウン・メニューに含まれています。発行されたアプリケーションが指定されていない場合，VuGen はサーバのデスクトップを使用します。発行されたアプリケーションの追加または名前変更を行った場合は，この記録オプションを閉じてから再度開き，新しいリストを表示します。また，発行されたアプリケーションが存在していることがわかっている場合，その名前を手動で入力することもできます（ドロップダウン・リストが正確でない場合に便利です）。 Citrix クライアントで，発行されたアプリケーションの名前を変更するには，Citrix サーバ・マシンで変更を行う必要があります。［**Manage Console**］>［**Application**］を選択し，新しいアプリケーションを作成するか，既存のアプリケーションの名前を変更します。 発行されたアプリケーションを指定しなければ，Citrix のロード・バランシングは機能しません。サーバのデスクトップにアクセスする場合にロード・バランシングを使用するには，デスクトップをサーバ・マシンで発行されたアプリケーションとして登録し，［発行されたアプリケーション］ドロップダウン・リストからこの名前を選択します。

UI 要素	説明
接続パラメータを定義する	ログオン情報と接続情報を手動で定義できます。
ログオン情報	Citrix ユーザの［**ユーザ名**］，［**パスワード**］，および［**ドメイン**］を指定します。必要に応じて，MetaFrame サーバがクライアントの識別に使用する［**クライアント名**］を指定することもできます。
接続パラメータに ICA ファイルを使用する	ログ設定情報が含まれている ICA ファイルを指定します。

［Citrix］の［レコーダ］ノード

記録中にタイトルの変更されたウィンドウ名を生成する方法を指定できます。画面のスナップショットをスクリプト・ファイルと一緒に保存するかどうか，また，テキスト同期化関数を生成するかどうかを指定することもできます。

利用方法	**［ツール］>［記録オプション］>［Citrix］>［レコーダ］**
重要情報	このノードは特定のプロトコルにのみ適用されます。プロトコルの完全な一覧と各プロトコルに関連付けられたノードについては，325 ページの「プロトコルの互換性に関する表」を参照してください。

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
画面ショットを保存する	該当する場合，スクリプトのステップごとに，Citrix クライアント・ウィンドウのスナップショットが保存されます。記録したアクションをより的確に理解するために，このオプションを有効にすることをお勧めします。ただし，スナップショットを保存すると使用するディスク領域が増え，記録セッションの速度が低下します。
ウィンドウ名	アプリケーションには，記録中にアクティブなウィンドウの名前が変わるものがあります。スクリプトをそのまま再生しようとすると，仮想ユーザは最初のウィンドウ名を使用するため再生が失敗します。次のようなウィンドウの命名規則を指定できます。この命名規則に基づいて，ウィンドウを特定する共通のプレフィックスまたはサフィックスが使用されます。 ➤ **[新規のウィンドウ名をそのまま使用する]**：ウィンドウのタイトルをそのままウィンドウ名として表示するように設定します。(標準設定)。 ➤ **[新規名に共通のプレフィックスを使用する]**：ウィンドウ・タイトルの先頭の共通文字列をウィンドウ名として使用する場合に選択します。 ➤ **[新規名に共通のサフィックスを使用する]**：ウィンドウ・タイトルの末尾の共通文字列をウィンドウ名として使用する場合に選択します。 あるいは，記録後に実際のスクリプトでウィンドウ名を変更することもできます。スクリプト・ビューでウィンドウ名を見つけ，ウィンドウ名の先頭または末尾をワイルドカードの「*」で置き換えます。 **例**：ctrx_sync_on_window ("My Application*", ACTIVATE, …CTRX_LAST);

［COM/DCOM］の［フィルタ］ノード

記録する COM/DCOM オブジェクトを定義できます。

利用方法	**［ツール］>［記録オプション］>［COM/DCOM］>［フィルタ］**

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
DCOM プロファイル	**次のフィルタ・タイプのいずれかを指定します。** ➤ **［Default Filter］**：COM 仮想ユーザ・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。 また，［**名前を付けて保存**］ボタンや［**削除**］ボタンを使用して，現在の設定の保存やフィルタの削除を行えます。

UI 要素	説明
［DCOM リスナーの設定］リスト	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 オブジェクト，RDS オブジェクト，および Remote オブジェクトがあります。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］**：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。 ➤ **［パス］**：タイプ・ライブラリのパス。 ➤ **［インタフェース ID］**：タイプ・ライブラリの GUID（Global Unique Identifier）。

UI 要素	説明
	別の COM タイプ・ライブラリを追加します。 ➤ **[レジストリの参照]**：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。 ➤ **[ファイル システムの参照]**：ローカル・ファイル・システムからタイプ・ライブラリを選択できます。 ➤ **[MTS の参照]**：Microsoft Transaction Server のコンポーネントを追加します。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。
	COM タイプ・ライブラリを削除します。

UI 要素	説明
除外する	［除外インタフェース］ダイアログ・ボックスを使用して，フィルタのインタフェースを除外します。 このダイアログ・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。 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 によって警告が表示されます。［**Don't Ask me again**］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。

［COM/DCOM］の［オプション］ノード

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。

利用方法	［ツール］> ［記録オプション］> ［COM/DCOM］> ［オプション］

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
ADO Recordset フィルタ	複数のレコードセットの処理が，1 行の fetch ステートメントにまとめられます（標準設定では有効）。
一時的 VARIANTs As Globals を宣言する	一時 VARIANT が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されます（標準設定では有効）。
配列を別々の範囲に入れる	各配列が独立のスコープに入力されます（標準設定では有効）。
構成を別々の範囲に入れる	各構造体が独立のスコープに入力されます（標準設定では有効）。
COM の例外を生成する	記録中に例外が発生した COM 関数およびメソッドが生成されます（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が生成されます（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が 16 に制限されます（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が記録されます（標準設定では有効）。
Recordset の内容を保存する	Recordset の内容が VuGen で表示できるように記録中にグリッドとして保存されます（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・オブジェクトがすべてトラップされます（標準設定では無効）。

［データベース］ノード

データベース・プロトコル用の記録オプションを設定できます。

利用方法	**［ツール］>［記録オプション］>［データベース］>［データベース］**
重要情報	このノードは特定のプロトコルにのみ適用されます。プロトコルの完全な一覧と各プロトコルに関連付けられたノードについては，325 ページの「プロトコルの互換性に関する表」を参照してください。

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
	［詳細記録オプション］ダイアログ・ボックスを開きます
自動トランザクション	**lrd_exec** 関数と **lrd_fetch** 関数がすべてトランザクションとしてマークされます。これらのオプションを有効にすると，VuGen によってすべての **lrd_exec** 関数または **lrd_fetch** 関数の前後に **lr_start_transaction** 関数と **lr_end_transaction** 関数が挿入されます。**標準設定値**：無効。
スクリプト オプション	記録されたスクリプトに **lrd_stmt** のオプションの値を説明するコメントが生成されます。また，スクリプト行の最大長を指定できます。 **標準設定値**：80 文字。
思考遅延時間	VuGen ではオペレータの思考遅延時間が自動的に記録されます。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 LRD 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。 **標準設定値**：5 秒。

［詳細記録オプション］ダイアログ・ボックス

データベース・プロトコル用の詳細記録オプションを設定できます。

利用方法	**［ツール］＞［記録オプション］＞［データベース］＞［データベース］＞［詳細設定］**
重要情報	このダイアログ・ボックスは特定のプロトコルにのみ適用されます。プロトコルの完全な一覧と各プロトコルに関連付けられたノードについては，325 ページの「プロトコルの互換性に関する表」を参照してください。

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
コード作成のバッファ サイズ	コード生成バッファの最大サイズをキロバイト単位で指定します。 **標準設定値**：128 キロバイト。
CtLib 関数のオプション	送信データのタイム・スタンプや，拡張結果ステートメントを作成するよう VuGen に指示できます。 ➤ **[送信データのタイム スタンプを作成する]**：**mpszReqSpec** パラメータに **TotalLen** キーワードと **Log** キーワードを設定して **lrd_send_data** ステートメントが生成されます。［詳細記録オプション］ダイアログ・ボックスでは，**TimeStamp** キーワードも生成するように指定することができます。既存のスクリプトでこの設定を変更した場合は，［**ツール**］>［**スクリプトの再生成**］を選択して仮想ユーザ・スクリプトを再生成する必要があります。標準設定で **Timestamp** キーワードを生成することは推奨されません。記録中に生成されたタイム・スタンプは再生中に生成されるものと異なるため，スクリプトの実行が失敗するからです。このオプションを使用する必要があるのは，（**lrd_send_data** の後の **lrd_result_set** が失敗する）スクリプト実行中に試行が失敗した後のみです。オプションを指定すれば，生成されたタイム・スタンプと，**lrd_send_data** を実行して失敗したときのタイム・スタンプを相関できるようになります。 ➤ **[拡張結果ステートメントを作成する]**：結果セットを準備する段階で **lrd_result_set** 関数が生成されます。このオプションを選択すると，**lrd_result_set** 関数を拡張した関数である **lrd_result_set_ext** が生成されます。この関数は，結果セットを準備するほかに，**ct_results** からリターン・コードとタイプを発行します。

UI 要素	説明
記録エンジン	旧バージョンの LRD 記録エンジンを使ってスクリプトを記録するよう VuGen に指示して，VuGen の旧バージョンと互換性を保つことができます。 **注：**このオプションはシングル・プロトコルのスクリプトにのみ適用されます。
記録ログのオプション	トレース・ファイルと ASCII ログ・ファイルの詳細レベルを設定できます。トレース・ファイルに対して指定できるレベルとしては，［**オフ**］，［**エラー トレース**］，［**簡易トレース**］，および［**完全トレース**］があります。［エラー トレース］に設定すると，エラー・メッセージだけがログに書き込まれます。［簡易トレース］に設定すると，エラーのほかに，記録中に生成された関数のリストがログに書き込まれます。［完全トレース］に設定すると，メッセージ，告示，警告などがすべてログに書き込まれます。 記録セッションに関して ASCII タイプのログが生成されるようにも設定できます。指定できるレベルとしては，［**オフ**］，［**簡易詳細**］，［**完全詳細**］があります。［簡易詳細］設定では，すべての関数がログに書き込まれます。［完全詳細］設定では，生成されたすべての関数とメッセージが ASCII コードでログに書き込まれます。

［Data Format Extension］-［Chain Configuration］ノード

チェーンとデータ形式拡張機能を追加，削除，変更できます。拡張機能のチェーンにデータ形式拡張機能を追加できます。

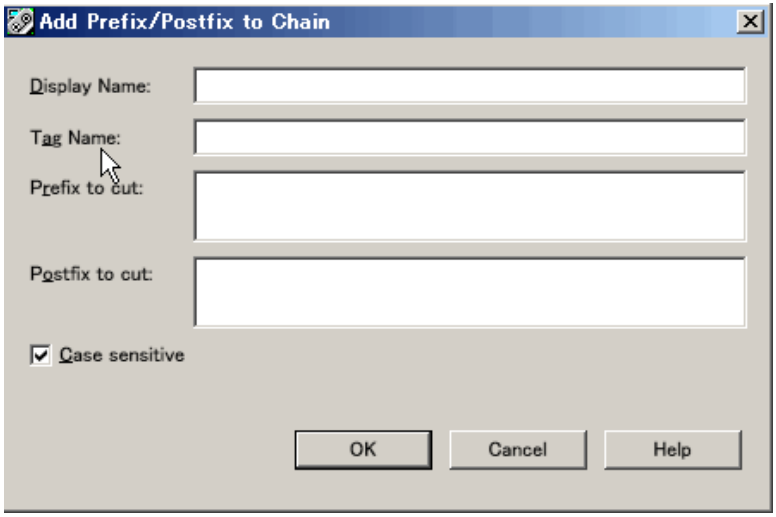
利用方法	**［ツール］>［記録オプション］>［Data Format Extension］>［Chain Configuration］**
重要情報	このノードは特定のプロトコルにのみ適用されます。プロトコルの完全な一覧と各プロトコルに関連付けられたノードについては，325 ページの「プロトコルの互換性に関する表」を参照してください。
関連項目	903 ページの「データ形式拡張機能」 913 ページの「データ形式拡張機能リスト」

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
［チェーン］表示枠	チェーンのリストが表示されます。
	［**Add Chain**］：新しいチェーンを追加できます。
	［**Edit Chain Name**］：チェーンの名前を変更できます。
	［**Delete Chain**］：選択したチェーンを削除します。
［チェーン：< チェーン名 >］表示枠	
	［**Add DFE**］：［**チェーン**］表示枠で選択したチェーンにデータ形式拡張機能を追加できます。データ形式拡張機能の詳細については，913 ページの「データ形式拡張機能リスト」を参照してください。
	［**Edit DFE**］：プレフィックス / ポストフィックス拡張機能を選択した場合，［**Add Prefix/Postfix to Chain**］ダイアログ・ボックスで詳細を変更できます。ダイアログ・ボックスの詳細については，347 ページの「［Add Prefix/Postfix to Chain］ダイアログ・ボックス」を参照してください。
	［**Delete DFE**］：選択したデータ形式拡張機能をチェーンから削除します。
	［**上へ移動 / 下へ移動**］：チェーンのデータ形式拡張機能が上または下に移動します。拡張機能は，拡張機能のリストに表示されている順序で実行されます。
名前	データ形式拡張機能の表示名。
タグ	拡張機能の一意の ID。
プロバイダ	データ形式拡張機能の作成者。
［処理の続行］：	データ形式拡張機能でデータが正常に再フォーマットされたときのチェーンの動作を決定します。 ➤ 条件が true の場合，変換されたデータが VuGen によって引き続きチェーンの次のデータ形式拡張機能に渡されます。 ➤ 条件が false の場合，チェーンは終了します。

［Add Prefix/Postfix to Chain］ダイアログ・ボックス

このダイアログ・ボックスでは，プレフィックス / ポストフィックス拡張機能の編集や選択したチェーンへのプレフィックス / ポストフィックス拡張機能の追加を実行できます。



利用方法	**3 ［ツール］>［記録オプション］>［Data Format Extension］>［Chain Configuration］ノードに移動します。** **4 ［チェーン：<チェーン名>］領域で ，ボタンをクリックします。** **5 ［Prefix/Postfix Extension］オプションを選択します。**
関連項目	345 ページの「［Data Format Extension］-［Chain Configuration］ノード」 903 ページの「データ形式拡張機能」 913 ページの「データ形式拡張機能リスト」

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
大文字と小文字を区別する	大文字と小文字が一致した場合にのみ，文字列の定義したプレフィックスやポストフィックスから切り取るように拡張機能を設定します。
Display name	プレフィックス / ポストフィックス拡張機能の名前。

UI 要素	説明
Postfix to cut	文字列の末尾から切り取る部分。
Prefix to cut	文字列の先頭から切り取る部分。
タグ名	プレフィックス / ポストフィックス拡張機能の一意の ID。

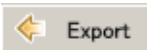
 ［Data Format Extension］- ［Code Generation］ノード

コード生成時にデータ形式拡張機能が有効になります。HTTP メッセージの各メッセージ・セクションのチェーンを定義できます。

利用方法	**［ツール］>［記録オプション］>［Data Format Extension］>［Code Generation］**
重要情報	このノードは特定のプロトコルにのみ適用されます。プロトコルの完全な一覧と各プロトコルに関連付けられたノードについては，325 ページの「プロトコルの互換性に関する表」を参照してください。
関連項目	903 ページの「データ形式拡張機能」 913 ページの「データ形式拡張機能リスト」

ユーザ・インタフェース要素の説明は次のとおりです（ラベルのない要素は山括弧で囲んで示します）。

UI 要素	説明
データ形式拡張機能を有効にする	HTTP メッセージの各メッセージ・セクションのチェーンを選択できます。標準設定では選択が解除されています。
設定	
フォーマット	➤ **Code and snapshots**：（標準設定）コードおよびスナップショット・データに対してデータ形式拡張機能が有効になります。 ➤ **スナップショット**：スナップショット・データに対してデータ形式拡張機能が有効になりますが，スクリプト自体のデータはフォーマットされません。

UI 要素	説明
フォーマットされたデータの検証	フォーマットされたデータが変換されて元の状態に戻り，元のデータと一致することが検証されて，その結果が確認されます。 **注：**Base64 拡張機能でのみ使用できます。
チェーン	
	ファイルからデータ形式拡張機能をインポートします。 **注：**ファイルをインポートするマシンで作成されたファイルのみをインポートできます。自分のマシンで設定されていない拡張機能がファイルに含まれている場合，関連するチェーンの選択した項目が赤色で強調表示されます。
	ファイルにデータ形式拡張機能をエクスポートします。
< メッセージ・セクションのリスト >	スクリプトに含まれている HTTP メッセージの次のセクションのリストが表示されます。 ➤ 本文 ➤ ヘッダ ➤ Cookie ➤ クエリ文字列 リストからメッセージ・セクションを選択すると，セクション・チェーン表示枠（下記参照）のタイトルに選択した内容が反映され，表示枠にそのセクションのチェーンのリストが表示されます。
< セクション・チェーン >	
	［**Add Chain**］：選択したメッセージ・セクションにチェーンを追加します。 **注：** ➤ ［ヘッダ］セクションと［Cookie］セクションでのみ有効です。選択したメッセージ・セクションにチェーンを追加できます。 ➤ VuGen で［ヘッダ］セクションまたは［Cookie］セクションに対してチェーンを適切に照合できるようにするには，［**名前**］カラムの名前が［ヘッダ］セクションまたは［Cookie］セクションの名前と一致している必要があります。

UI 要素	説明
	**[Delete Chain]**：対応するメッセージ・セクションからチェーンを削除します。 **注**：メッセージ・セクションから標準設定のオプションを削除することはできません。
	**[リセット]**：選択したチェーンを［Chain］カラムから消去します。

 ［EJB］の［コード生成オプション］ノード

EJB の記録オプションを設定できます。

利用方法	**［ツール］>［記録オプション］>［EJB オプション］>［コード生成オプション］**
重要情報	このノードは特定のプロトコルにのみ適用されます。プロトコルの完全な一覧と各プロトコルに関連付けられたノードについては，325 ページの「プロトコルの互換性に関する表」を参照してください。

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
自動トランザクション	すべての EJB メソッドがトランザクションとして扱われるように自動的にマークされます。これで，すべてのメソッドを **lr.start_transaction** と **lr.end_transaction** 関数で囲むことになります。 **標準設定値**：有効。
EJB 初期化メソッド	EJB/JNDI 初期化プロパティが書き込まれるメソッド。利用可能なメソッドは，**init** と **action** です。 **標準設定値**：init。
値チェックを挿入	**lr.value_check** 関数が各 EJB メソッドの後に自動的に挿入されます。この関数は，プリミティブな値および文字列で返される期待値をチェックします。

［Flex］-［RTMP］ノード

このノードでは，flex_rtmp_recive_stream ステップを Flex RTMP スクリプトに含めることができます。

利用方法	**［ツール］>［記録オプション］>［Flex］>［RTMP］**
重要情報	このノードは特定のプロトコルにのみ適用されます。プロトコルの完全な一覧と各プロトコルに関連付けられたノードについては，325 ページの「プロトコルの互換性に関する表」を参照してください。

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
flex_rtmp_receive_stream ステップの生成	ストリームを記録するときに 1 つのステップを生成します。このステップでは特定のアクション（一時停止やシークなど）が再生されません。スクリプトでこれらのアクションが必要な場合，このチェック・ボックスをクリアして，すべての受信ステップおよび送信ステップを記録します。ただし，この場合，LoadRunner の『最初にお読みください』で説明しているようにスクリプトを手動で変更する必要があります。

［Flex］-［設定］ノード（記録）

外部 JVM（Java 仮想マシン）のパスを設定できます。

利用方法	**［ツール］>［記録オプション］>［Flex］>［設定］**
重要情報	このノードは特定のプロトコルにのみ適用されます。プロトコルの完全な一覧と各プロトコルに関連付けられたノードについては，325 ページの「プロトコルの互換性に関する表」を参照してください。

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
Use External JVM	外部 JVM を使用できます。
External JVM Path	外部 JVM のパスを指定します。

［Flex］-［外部オブジェクト］ノード（記録）

このダイアログ・ボックスでは，LoadRunner で Flex スクリプトの外部オブジェクトをどのように処理するのかを設定できます。

利用方法	**［ツール］>［記録オプション］>［Flex］>［外部オブジェクト］**
関連タスク	679 ページの「外部 Java シリアライザを使用してシリアル化する方法」 680 ページの「LoadRunner シリアライザを使用してスクリプトをシリアル化する方法」
関連項目	674 ページの「Flex の概要」 675 ページの「Flex スクリプトの外部オブジェクト」

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
外部オブジェクトをシリアル化しない	標準設定を使用してスクリプトを生成します。
［Classpath エントリ］リスト	クラスパスのエントリのリスト。
オブジェクトのシリアル化方法	ドロップダウン・リストから該当のオプションを選択します。 ➤ LifeCycle Data Services または BlazeDS サーバを使用している場合，［**カスタム Java クラス**］を選択します。 ➤ LifeCycle Data Services または BlazeDS サーバを使用していない場合，［**LoadRunner AMF シリアライザ**］を選択します。

UI 要素	説明
	**下向き矢印**：クラスパスのエントリをリストの下方向に移動します。
	**上向き矢印**：クラスパスのエントリをリストの上方向に移動します。
	**[クラスパスの追加]**：クラスパスのリストに新しい行を追加します。
	**[フォルダのすべてのクラス ファイルを追加]**：フォルダのファイルをすべてクラスパス・リストに追加します。
	[**削除**]：クラスパスを永久に削除します。

 [一般] の [プロトコル] ノード

スクリプト言語とオプションを設定することで，スクリプト生成のプリファレンスを設定できます。

利用方法	**[ツール] > [記録オプション] > [一般] > [プロトコル]**
重要情報	このノードは特定のプロトコルにのみ適用されます。プロトコルの完全な一覧と各プロトコルに関連付けられたノードについては，325 ページの「プロトコルの互換性に関する表」を参照してください。

ユーザ・インタフェース要素の説明は次のとおりです。

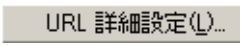
UI 要素	説明
[アクティブなプロトコル] リスト	マルチ・プロトコル・スクリプトを構成するプロトコルのリスト。VuGen では，記録セッション中にコードを生成するプロトコルのリストを修正できます。次回の記録セッションで記録するプロトコルについて，それらの横にあるチェック・ボックスを選択します。次回の記録セッションで記録しないプロトコルについて，それらの横にあるチェック・ボックスをクリアします。

［一般］の［記録］ノード

記録レベルを選択することによって，仮想ユーザ・スクリプトの生成時に，記録する情報と使用する関数が指定できます。

利用方法	**［ツール］>［記録オプション］>［一般］>［記録］**
重要情報	このノードは特定のプロトコルにのみ適用されます。プロトコルの完全な一覧と各プロトコルに関連付けられたノードについては，325 ページの「プロトコルの互換性に関する表」を参照してください。

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
	［HTML 詳細設定］ダイアログ・ボックスを開きます
	［URL 詳細設定］ダイアログ・ボックスを開きます
GUI ベースのスクリプト	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推奨されます。また，PeopleSoft Enterprise および Oracle Web Applications の場合にも推奨されます。
HTML ベースのスクリプト	これは，Web（HTTP/HTML）仮想ユーザの標準設定の記録レベルです。現在の Web ページのコンテキストで HTML アクションを記録するよう VuGen に指示します。記録セッション中，リソースのすべては記録されませんが，再生時にはダウンロードされます。このオプションは，アプレットおよび VB スクリプトを使用するブラウザ・アプリケーションの場合に推奨されます。
URL ベースのスクリプト	サーバからのすべての要求とリソースが記録されます。自動的にあらゆる HTTP リソースが URL ステップ（**web_url** ステートメント）として，フォームの場合には，**web_submit_data** として記録されます。**web_link**，**web_image**，および **web_submit_form** 関数は生成されず，フレームも記録されません。このオプションは，非ブラウザ・アプリケーションの場合に推奨されます。

［URL 詳細設定］ダイアログ・ボックス

URL 記録モードを使用して，スクリプトの詳細オプションを設定できます。

利用方法	**［ツール］>［記録オプション］>［一般］>［記録］>［URL 詳細設定］**
重要情報	このダイアログ・ボックスは特定のプロトコルにのみ適用されます。プロトコルの完全な一覧と各プロトコルに関連付けられたノードについては，325 ページの「プロトコルの互換性に関する表」を参照してください。

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
標準設定を使用(U)	このダイアログ・ボックスの既定の設定に戻します。
HTML ページの後にリソースの同時実行グループを作成する	URL の後，リソースが同時実行グループとして記録されます（**web_concurrent_start** ステートメントと **web_concurrent_end** ステートメントで囲まれます）。リソースには，画像などのファイルと **js** ファイルが含まれます。このオプションを無効にすると，リソースは別々の **web_url** ステップとしてリストされますが，同時実行グループとしてのマークは付きません。
Enable EUC-Encoded Web Pages	（日本語版 Windows のみ）EUC エンコーディングを使用するよう VuGen に指示します。詳細については，316 ページの「EUC エンコーディング（日本語版 Windows のみ）」を参照してください。
web_custom_request のみを使用する	すべての HTTP 要求がカスタム要求として記録されます。VuGen によって内容に関係なくすべての要求に **web_custom_request** 関数が生成されます。非ブラウザ・アプリケーションの場合に推奨されます。

［HTML 詳細設定］ダイアログ・ボックス

HTTP ベースのスクリプトの詳細オプションを設定できます。

利用方法	**［ツール］>［記録オプション］>［一般］>［記録］>［HTML 詳細設定］**
重要情報	このダイアログ・ボックスは特定のプロトコルにのみ適用されます。プロトコルの完全な一覧と各プロトコルに関連付けられたノードについては，325 ページの「プロトコルの互換性に関する表」を参照してください。

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
標準設定を使用(U)	このダイアログ・ボックスの既定の設定に戻します。

UI 要素	説明
生成された HTML 以外の要素	多くの Web ページには，アプレット，XML，ActiveX 要素，JavaScript など，HTML 以外の項目が含まれています。通常こうした HTML 以外の要素には，それ自身のリソースが含まれているか，取得されるかします。下記のオプションを使って，HTML 以外の要素をどのように記録するかを制御することができます。 ➤ **[現在のスクリプト ステップ内に記録する]**：HTML によらずに生成されたリソースについて新規関数が生成されません。すべてのリソースが，**web_url**，**web_link**，**web_submit_data** などの関連する関数の引数としてリストされます。Web 関数の引数となったリソースには **EXTRARES** フラグが付けられます。 ➤ **[個別ステップに記録し，同時実行グループを使用する]**：非 HTML 生成リソースの 1 つ 1 つについて新しい関数が生成されます。そのページの関数（**web_url** や **web_link** など）にはそれらが項目として含まれません。特定のリソースについて生成されたすべての **web_url** 関数は，同時実行グループ内に配置されます（**web_concurrent_start** と **web_concurrent_end** で囲まれます）。 ➤ **[記録しない]**：非 HTML 生成リソースは記録されません。

UI 要素	説明
スクリプト タイプ	➤ **[ユーザ アクションを記述するスクリプト]**：アクションに直接対応する関数が作成されます。URL（**web_url**），リンク（**web_link**），画像（**web_image**），フォーム送信（**web_submit_form**）などの関数が作成されます。コンテキスト・センシティブ記録と似た非常にわかりやすいスクリプトが作成されます。 ➤ **[明示的な URL のみを含むスクリプト]**：すべてのリンク，画像および URL が **web_url** ステートメントとして記録されます。ただし，フォームの場合は，web_submit_data ステートメントとして記録されます。**web_link**，**web_image**，および **web_submit_form** 関数は生成されません。生成されるスクリプトは直観的ではなくなります。サイト内の多数のリンクが同じリンク・テキストを使用している場合に役立ちます。1 つ目のオプションを使用してサイトを記録すると，リンクの出現（インスタンス）が記録されますが，2 つ目のオプションを使用して記録すると，それぞれのリンクが URL のリストとして記録されます。このことにより，ステップの相関，およびパラメータ化を容易に行うことができます。

［一般］の［スクリプト］ノード

スクリプト言語とオプションを設定することで，スクリプト生成のプリファレンスを設定できます。

利用方法	［ツール］>［記録オプション］>［一般］>［スクリプト］
重要情報	このノードは特定のプロトコルにのみ適用されます。プロトコルの完全な一覧と各プロトコルに関連付けられたノードについては，325 ページの「プロトコルの互換性に関する表」を参照してください。

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
記録停止時に AUT の全プロセスを閉じる	VuGen が記録を停止すると，すべての AUT（テスト対象アプリケーション）の処理が自動的に終了します。 **標準設定値**：有効。
配列の相関	文字列，構造体，数値など，すべてのデータ型の配列が追跡されて相関されます。 **標準設定値**：有効。
大きい数の相関	int，long int，64 ビットの char，float，double などの長いデータ型が相関されます。 **標準設定値**：有効。
単純文字列の相関	単純で，配列ではない文字列や文章が相関されます。 **標準設定値**：有効。
小さい数の相関	byte，char，および short int などの短いデータ型が相関されます。 **標準設定値**：有効。
構造の相関	複雑な構造要素が追跡されて相関されます。 **標準設定値**：有効。
プリミティブをローカルとして宣言する	プリミティブ値の変数がクラス変数ではなくローカル変数として宣言されます（C，C#，.NET のみ）。 **標準設定値**：有効。
バリアントを明示的に宣言する	ByRef バリアントを処理するため，バリアント・タイプが明示的に宣言されます（Visual Basic for Applications のみ）。 **標準設定値**：有効。
end_transaction 後に固定の思考遅延時間を生成する	トランザクション終了後，固定思考遅延時間が秒単位で追加されます。このオプションを有効にする場合は，思考遅延時間の値を指定できます。 **標準設定値**：無効（有効の場合は 3 秒）。
記録されたイベントのログを生成する	記録中に発生したすべてのイベントのログが生成されます。 **標準設定値**：有効。

UI 要素	説明
しきい値を超える場合に思考遅延時間を生成する	思考遅延時間のしきい値が使用されます。記録された思考遅延時間がしきい値に満たない場合，VuGen は思考遅延時間ステートメントを生成しません。しきい値も指定します。標準設定の値は 3 です。思考遅延時間が 3 秒以内の場合は，VuGen は思考遅延時間のステートメントを生成しません。このオプションを無効にすると，VuGen は思考遅延時間を生成しません。 **標準設定値**：有効（3 秒）。
出力パラメータ値を挿入する	各呼び出しの後に出力パラメータ値が挿入されます（C，C#，.NET のみ）。 **標準設定値**：有効。
呼び出し後情報を挿入する	各メッセージ呼び出しの後に，その内容を表すログ・メッセージが挿入されます（C 以外のみ）。 **標準設定値**：有効。
呼び出し前情報を挿入する	各メッセージ呼び出しの前に，その内容を表すログ・メッセージが挿入されます（C 以外のみ）。 **標準設定値**：有効。
アクション ファイル内の最大行数	アクションの行数が指定されたしきい値を超えた場合に新しいファイルが作成されます。標準のしきい値は 60000 行です（C，C#，.NET のみ）。 **標準設定値**：有効。
長い文字列をパラメータで置換する	最大文字数を超える文字列がパラメータに保存されます。このオプションの初期の最大長は 100 文字です。パラメータと文字列全体は，次の形式で，スクリプトのフォルダ内の **lr_strings.h** ファイルに保存されます。 **const char <paramName_uniqueID> =”string”.** このオプションにより，スクリプトが読みやすくなります。スクリプトのパフォーマンスには影響しません。 **標準設定値**：有効。
プリミティブの戻り値に変数を再使用する	メソッド呼び出しから受け取るプリミティブに同じ変数が再利用されます。これは［**プリミティブをローカルとして宣言する**］設定よりも優先されます。 **標準設定値**：有効。

UI 要素	説明
COM ローカル サーバとして作成されたプロセスを追跡する	記録されたアプリケーションのサブプロセスの 1 つが COM ローカル・サーバとして作成されている場合は，そのアプリケーションの動作が追跡されます（C および COM のみ）。 **標準設定値**：有効。
フルネームを使用する	新しい変数を宣言するときに完全な型名が使用されます（C# および .NET のみ）。 **標準設定値**：有効。
配列に対してヘルパを使用する	ヘルパ関数を使って，バリアントの配列からコンポーネントが抽出されます（Java および VBScript のみ）。 **標準設定値**：有効。
オブジェクトに対してヘルパを使用する	ヘルパ関数を使って，バリアントのオブジェクトの参照が引数として関数に渡されたときに，その参照が抽出されます（Java および VBScript のみ） **標準設定値**：有効。

［GUI プロパティ］の［詳細］ノード

Click and Script 仮想ユーザの詳細記録オプションを設定できます。

利用方法	**［ツール］＞［記録オプション］＞［GUI プロパティ］＞［詳細］**
重要情報	このノードは特定のプロトコルにのみ適用されます。プロトコルの完全な一覧と各プロトコルに関連付けられたノードについては，325 ページの「プロトコルの互換性に関する表」を参照してください。

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
コード生成の設定	このセクションの詳細については，363 ページの「コード生成設定のプロパティ」を参照してください。
記録の設定	このセクションの詳細については，362 ページの「記録設定のプロパティ」を参照してください。

記録設定のプロパティ

UI 要素	説明
AJAX ステップ用にスナップショットを生成	AJAX ステップのスナップショットの生成を有効にします。このオプションを有効にすると，記録中にエラーが発生することがあります。 **標準設定値**：有効。
マウス イベントによる「クリック」を記録する	Click () メソッドではなく，マウス・イベントをキャプチャすることでマウスのクリックを記録します。記録対象アプリケーションで DOM の click() メソッドが使用されている場合に有効にします。同じユーザ・アクションに対して複数の関数が生成されるのを防ぎます。 **標準設定値**：有効。
描画関連のプロパティ値の記録	DOM オブジェクトのレンダリング関連プロパティ（**offsetTop** など）の値を記録して，再生時に利用できるようにします。これにより，再生速度が大幅に低下することがあります。 **標準設定値**：有効。
ソケット レベル データを記録する	ソケット・レベル・データの記録を有効にします。このオプションを無効にした場合は，記録の前に，開始 URL を手動で追加する必要があります。また，スクリプトを HTML レベルで再生成できなくなります。 **標準設定値**：有効。

コード生成設定のプロパティ

UI 要素	説明
ブラウザ タイトルの自動検証を有効にする	ブラウザ・タイトルの自動検証が有効になります。 **標準設定値**：有効。
コンテキスト外ステップの生成を有効にする	ActiveX コントロールや Java アプレットの URL ベースのスクリプトが作成され，再生できるようになります。これらの関数はネイティブの記録の一部ではないため，コンテキスト外の記録と呼ばれます。 **標準設定値**：有効。
タイトル検証を行う	➤ **[ナビゲーションのたびに]**：ナビゲーションの後にのみ，タイトル検証が行われます。フィールドが複数あるフォームの入力など，ユーザが同じページでいくつかの操作を実行した場合，タイトルは変わらないため，検証は不要です。 ➤ **[ステップごとに]**：ステップごとにタイトル検証が行われ，ステップによってブラウザのタイトルが変更されていないことが確認されます。ブラウザのタイトルが変更されると，スクリプトが失敗する原因となることがあります。 ➤ **[タイトルがない場合に，URL を使用してタイトル検証を行います]**：ブラウザ・ウィンドウにタイトルがない場合に，その URL を使用してステップごとにタイトル検証を行います。

[GUI プロパティ] の [Web イベントの定義] ノード

スクリプトに記録される詳細レベルを設定できます（Web イベント記録）。

利用方法	**[ツール] > [記録オプション] > [GUI プロパティ] > [Web イベントの定義]**
重要情報	このノードは特定のプロトコルにのみ適用されます。プロトコルの完全な一覧と各プロトコルに関連付けられたノードについては，325 ページの「プロトコルの互換性に関する表」を参照してください。

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
［基本］イベント設定レベル	➤ 画像，ボタン，ラジオ・ボタンなどの標準的な Web オブジェクトに対するクリック・イベントを必ず記録します。 ➤ フォーム内での送信イベントを必ず記録します。 ➤ ハンドラまたは動作が関連付けられているその他のオブジェクトでのクリック・イベントを記録します。 ➤ イメージおよびイメージ・マップ上の mouseover イベントは，そのイベントの後に同じオブジェクトに対するイベントが実行される場合にのみ記録します。
ユーザ定義設定	［ユーザ定義 Web イベント記録設定］ダイアログ・ボックスが開き，イベント記録設定をカスタマイズできます。
［高］イベント設定レベル	**中**レベルで記録されるオブジェクトに加えて，ハンドラまたは動作が関連付けられているオブジェクトに対する mouseover イベント，mousedown イベント，および double-click イベントを記録します。
［中］イベント設定レベル	**基本**レベルで記録されるオブジェクトに加えて，HTML タグ・オブジェクトの <DIV>，<SPAN>，<TD> に対するクリック・イベントも記録します。

［ユーザ定義 Web イベント記録設定］ダイアログ・ボックス

Web イベント記録のレベルをカスタマイズできます。

利用方法	**［ツール］>［記録オプション］>［GUI プロパティ］>［Web イベントの定義］>［ユーザ定義設定］**
重要情報	このダイアログ・ボックスは特定のプロトコルにのみ適用されます。プロトコルの完全な一覧と各プロトコルに関連付けられたノードについては，325 ページの「プロトコルの互換性に関する表」を参照してください。

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
<オブジェクト・リスト>	Web オブジェクトのリスト。このダイアログ・ボックスのほかの設定に基づいて，各 Web オブジェクトをカスタマイズできます。
<オブジェクト・メニュー>	➤ **[追加]**：新しい HTML タグ・オブジェクトがオブジェクト・リストに追加されます。タグの名前を入力します。 ➤ **[削除]**：オブジェクト・リストからオブジェクトが削除されます。
[イベント] メニュー	➤ **[追加]**：該当のオブジェクトの［イベント名］カラムにイベントが追加されます。 ➤ **[削除]**：該当のオブジェクトの［イベント名］カラムからイベントが削除されます。
イベント名	オブジェクトに関連付けられているイベントのリスト。
[ファイル] メニュー	➤ **[読み込みの設定]**：以前に作成したカスタム設定が読み込まれます。 ➤ **[設定に名前を付けて保存]**：現在の設定が保存されます。
応答	VuGen がいつイベントをリッスンするのかを決定する基準。 ➤ **[常に]**：常にイベントがリッスンされます。 ➤ **[ハンドラの場合]**：ハンドラが付加されているイベントがリッスンされます。ハンドラは，Web ページに含まれているコードであり，通常はスクリプト言語で書かれている関数またはルーチンです。対応するイベントが発生したときに制御が渡されます。 ➤ **[動作の場合]**：DHTML 動作が付加されているイベントがリッスンされます。DHTML 動作は，ページにおける特定の機能や振る舞いをカプセル化します。ページ上の標準的な HTML 要素に適用されている場合，その要素の標準設定の動作が拡張されます。 ➤ **[ハンドラまたは動作の場合]**：ハンドラまたは動作が付加されているイベントがリッスンされます。 ➤ **[しない]**：イベントは一切リッスンされません。 詳細については，322 ページの「イベントのリッスンと記録を行うためのヒント」を参照してください。

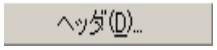
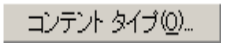
UI 要素	説明
記録	VuGen がいつイベントを記録するのかを決定する基準。 ➤ **[有効]**：VuGen が対象オブジェクトあるいはイベントの「バブリング先」である別のオブジェクトをリッスンしている場合，イベントが生じるたびにそれが記録されます。「バブリング」とは，子オブジェクトで発生したイベントが，イベントを処理するイベント・ハンドラに遭遇するまで，HTML コード内の階層をさかのぼる処理です。 ➤ **[無効]**：指定したイベントは記録されず，イベント・バブリングがある場合はそれが無視されます。 ➤ **[次のイベントで有効]**：［**有効**］との唯一の違いは，以降のイベントが同じオブジェクトで発生した場合にのみイベントが記録されることです。たとえば，マウスオーバ操作によって画像リンクが変わるとします。この画像の上をマウスが通過するたびに，mouseover イベントを記録する必要はないかもしれません。ただし，リンクが mouseover イベント後に表示される画像によってのみ有効になるので，mouseover イベントを，同じオブジェクトに対するクリック・イベントの前に記録することが重要となります。 詳細については，322 ページの「イベントのリッスンと記録を行うためのヒント」を参照してください。
設定のリセット	カスタム設定が［**基本**］，［**中**］，または［**高い**］の設定にリセットされます。

HTTP の［詳細］ノード

思考遅延時間，コンテキストのリセット，スナップショットの保存，および **web_reg_find** 関数にかかわるコード生成の方法をカスタマイズできます。

利用方法	**［ツール］>［記録オプション］>［HTTP プロパティ］>［詳細］**
重要情報	➤ このノードは特定のプロトコルにのみ適用されます。プロトコルの完全な一覧と各プロトコルに関連付けられたノードについては，325 ページの「プロトコルの互換性に関する表」を参照してください。 ➤ マルチ・プロトコル・スクリプトを使用している場合，このノードのいくつかのオプションは使用できません。

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
	［ヘッダ］ダイアログ・ボックスを開きます
	［コンテント タイプのフィルタ］ダイアログ・ボックスを開きます
	［リソース以外の項目］ダイアログ・ボックスを開きます
記録中，HTTP エラー時にスクリプトにコメントを追加する	HTTP 要求のエラーが発生するたびにスクリプトにコメントが追加されます。エラー要求は，記録中に 400 番以降の値を持つサーバ応答を生成した要求として定義されます。
ページ タイトルで web_reg_find 関数を生成する	すべての HTML ページのタイトルに対して，**web_reg_find** 関数が生成されます。VuGen によって，ページのタイトル・タグから文字列が取得され，それが **web_reg_find** の引数として使用されます。 ➤ **［サブ フレームで web_reg_find 関数を生成する］**：記録されたページのすべてのサブフレームにあるページのタイトルに対して，**web_reg_find** 関数が生成されます。 **注**：このオプションは Web プロトコルと Oracle NCA プロトコルにのみ適用されます。

UI 要素	説明
以前の記録エンジンを使用してスクリプトを記録する	シングル・プロトコルの記録エンジンを使用して記録されます。VuGen の標準設定では，Web（HTTP/HTML）仮想ユーザに対して，シングル・プロトコルの記録を含むすべての記録でマルチ・プロトコルの記録エンジンが使用されます。
思考遅延時間を記録する	思考遅延時間が記録され，**lr_think_time** 関数が生成されます。また，実際の思考遅延時間がしきい値より長かった場合にのみ **lr_think_time** 関数が生成されるように［**思考遅延時間しきい値**］の値を設定することもできます。 **注：**このオプションはワイヤレス・プロトコルのスクリプトにのみ適用されます。
各アクションごとにコンテキストをリセットする	アクションごとに HTTP コンテキストがすべてリセットされます。この設定により，仮想ユーザは新規ユーザがブラウザ・セッションを開始する様子をより正確にエミュレートできます。このオプションは，HTML コンテキストをリセットし，コンテキストを持たない関数が常にアクションの先頭に記録されるようにします。また，このオプションは，キャッシュをクリアし，ユーザ名とパスワードをリセットします。 **注：**このオプションは Web プロトコルと Oracle NCA プロトコルにのみ適用されます。

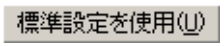
UI 要素	説明
スナップショットのリソースをローカルに保存する	記録時と再生時にスナップショット・リソースのローカル・コピーが保存されます。このため，正確なスナップショットが作成され，よりすばやく表示することができます。
サポート対象文字セット	➤ **[UTF-8]**：UTF-8 エンコーディングのサポートが有効になります。この設定を有効にすると，非 ASCII の UTF-8 文字がマシンのロケールで使用されているロケールに変換され，VuGen に正しく表示されます。このオプションは英語以外の UTF-8 エンコードのページでのみ有効にしてください。記録するサイトの言語がオペレーティング・システムの言語と一致している必要があります。英語以外の Web ページは，同じスクリプト内で異なるエンコーディング（ISO-8859-1 または shift_jis を組み合わせて）では記録できません。 ➤ **[EUC-JP]**：日本語版 Windows を使用している場合，このオプションを選択することで，EUC-JP 文字エンコーディングを使用する Web サイトに対するサポートを有効にできます。この設定を有効にすると，EUC-JP 文字がマシンのロケールで使用されているロケールに変換され，VuGen に正しく表示されます。VuGen によって，すべての EUC-JP（日本語 UNIX）文字列をマシンのロケールに合わせて Shift-JIS（日本語版 Windows）エンコーディングに変換し，スクリプトに **web_sjis_to_euc_param** 関数を追加します（漢字のみ）。

［ヘッダ］ダイアログ・ボックス

サーバに送信されるすべての HTTP 要求とともに，追加の HTTP ヘッダを自動的に送信できます。

利用方法	**［ツール］>［記録オプション］>［HTTP］>［詳細］>［ヘッダ］**
重要情報	➤ このダイアログ・ボックスは特定のプロトコルにのみ適用されます。プロトコルの完全な一覧と各プロトコルに関連付けられたノードについては，325 ページの「プロトコルの互換性に関する表」を参照してください。 ➤「リスキー・ヘッダ」とみなされる標準ヘッダには，「Authorization」，「Connection」，「Content-Length」，「Cookie」，「Host」，「If-Modified-Since」，「Proxy-Authenticate」，「Proxy-Authorization」，「Proxy-Connection」，「Referer」，「WWW-Authenticate」があります。［コンテント タイプ リスト］で選択しないかぎり，これらのヘッダは記録されません。

ユーザ・インタフェース要素の説明は次のとおりです。

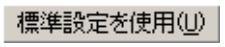
UI 要素	説明
	**［プラス］**：新しいエントリを追加します。
	**［マイナス］**：エントリを削除します。
	現在のリストを標準設定の値とエントリに戻します。
	すべてのリストを標準設定の値とエントリに戻します。
<ドロップダウン・メニュー>	このダイアログ・ボックスのオプションを制御します。 ➤ **ヘッダを記録しない。** ➤ **リスト内のヘッダを記録。** ➤ **リストに定義されていないヘッダを記録。**
<コンテント・タイプ・リスト>	記録される（されない）ヘッダのリスト。リストは，選択されたドロップダウン項目によって異なります。個々のチェックボックスを使用して，各項目を選択または選択解除できます。

［コンテント タイプのフィルタ］ダイアログ・ボックス

記録したスクリプトのコンテント・タイプをフィルタリングできます。記録するコンテント・タイプ，またはスクリプトから除外するコンテント・タイプを指定できます。

利用方法	**［ツール］＞［記録オプション］＞［HTTP］＞［詳細］＞［コンテント タイプ］**
重要情報	このノードは特定のプロトコルにのみ適用されます。プロトコルの完全な一覧と各プロトコルに関連付けられたノードについては，325 ページの「プロトコルの互換性に関する表」を参照してください。

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
	**［プラス］**：新しいエントリを追加します。
	**［マイナス］**：エントリを削除します。
	現在のリストを標準設定の値とエントリに戻します。
	すべてのリストを標準設定の値とエントリに戻します。
＜ドロップダウン・メニュー＞	このダイアログ・ボックスのオプションを制御します。 ➤ **コンテント タイプをフィルタしない。** ➤ **リスト内のコンテント タイプを除外。** ➤ **リストに定義されていないコンテント タイプを除外。**
＜コンテント・タイプ・リスト＞	フィルタされる（されない）コンテント・タイプのリスト。リストは，選択されたドロップダウン項目によって異なります。個々のチェックボックスを使用して，各項目を選択または選択解除できます。

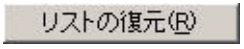
［リソース以外の項目］ダイアログ・ボックス

スクリプトを記録するときに，VuGen によって **web_url** 関数の Resource 属性を使って，再生中に対象リソースを取得するかどうかが指示されます。Resource 属性が 0 に設定されていると，対象リソースはスクリプトの実行中に取得されます。Resource 属性が 1 に設定されていると，そのリソース・タイプは仮想ユーザによってスキップされます。

特定のコンテント・タイプをリソースとして処理しないように除外することができます。たとえば，**gif** タイプのリソースがリソースとして処理されないように設定し，無条件にダウンロードされるように設定できます。

利用方法	**［ツール］>［記録オプション］>［HTTP］>［詳細］>［リソース以外の項目］**
重要情報	このダイアログ・ボックスは特定のプロトコルにのみ適用されます。プロトコルの完全な一覧と各プロトコルに関連付けられたノードについては，325 ページの「プロトコルの互換性に関する表」を参照してください。

ユーザ・インタフェース要素の説明は次のとおりです。

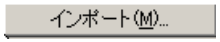
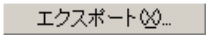
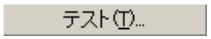
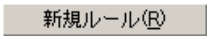
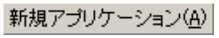
UI 要素	説明
	**［追加］**：リストに新しいエントリが追加されます。
	**［削除］**：リストからエントリが削除されます。
	標準設定のリストに戻ります。
< リソース以外のコンテント・タイプのリスト >	リソースとして記録しない項目のリスト。個々のチェックボックスを使用して，各項目を選択または選択解除できます。

HTTP の［相関］ノード

記録中にステートメントを自動的に相関させる相関ルールを作成できます。

利用方法	**［ツール］>［記録オプション］>［HTTP プロパティ］>［相関］**
重要情報	このノードは特定のプロトコルにのみ適用されます。プロトコルの完全な一覧と各プロトコルに関連付けられたノードについては，325 ページの「プロトコルの互換性に関する表」を参照してください。

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
	ルール情報が含まれている相関ファイルがインポートされます。
	ルール情報が含まれている相関ファイルがエクスポートされます。
	［トークン置換テスト パッド］ダイアログ・ボックスを開きます
	選択したルールまたはアプリケーションが削除されます。
	［新規ルール］表示枠を開きます
	［アプリケーション］リストに新しいアプリケーションが追加されます。
< アプリケーション・リスト >	アプリケーションとそのルールのリスト。
スクリプトにコメントを追加	相関ステップに記述コメントが追加されます。
記録中に相関を有効にする	指定した相関ルール基づいて，記録中に相関が有効になります。

［新規ルール］表示枠

新しいカスタム・ルールを定義できます。

利用方法	**［ツール］>［記録オプション］>［HTTP］>［相関］>［新規ルール］**
重要情報	この表示枠は特定のプロトコルにのみ適用されます。プロトコルの完全な一覧と各プロトコルに関連付けられたノードについては，325 ページの「プロトコルの互換性に関する表」を参照してください。

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
詳細(A)...	［詳細相関プロパティ］ダイアログ・ボックスを開きます

UI 要素	説明
アクション	次のオプションからルールのアクションのタイプを指定します。 ➤ **[本体テキスト全体の中でパラメータを検索]**：リンク，フォーム・アクションまたは Cookie だけではなく，本体の全体が検索されます。テキストは，指定した境界を使って検索されます。 ➤ **[リンクおよびフォームのアクションの中でパラメータを検索]**：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が検索されます。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を現在のリンクで定義します。 ➤ **[Cookie ヘッダーの中でパラメータを検索]**：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，Cookie 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 フィールド値をパラメータ化]**：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。 ➤ [**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 Cookie 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 Cookie のプレフィックスを指定する必要があります。
フィールド名	フィールド名。このフィールドは，[**フォーム フィールド値をパラメータ化**] タイプのアクションを作成する場合に入力する必要があります。
左の境界	ルールが適用される最も左側の境界。
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が区別されます。

UI 要素	説明
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，プレフィックスが使用されます。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。
右の境界	ルールが適用される最も右側の境界。ドロップダウン・メニューを使用して，文字列の末尾，改行文字，またはユーザ定義のテキストのいずれかとしてこの境界を定義します。
'#' で数字を置き換える	数値がすべてハッシュ記号（#）に置き換わります。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。 **例**：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。

 ［詳細相関プロパティ］ダイアログ・ボックス

相関ルールの詳細オプションを設定できます。

利用方法	**［ツール］>［記録オプション］>［HTTP］>［相関］>［新規ルール］>［詳細］**
重要情報	このノードは特定のプロトコルにのみ適用されます。プロトコルの完全な一覧と各プロトコルに関連付けられたノードについては，325 ページの「プロトコルの互換性に関する表」を参照してください。

ユーザ・インタフェース要素の説明は次のとおりです。

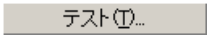
UI 要素	説明
代替の右境界	あらかじめ指定されている右境界が見つからなかった場合の代替条件です。[**ユーザ定義のテキスト**]，[**改行文字**]，[**ページの最後**] のいずれかのオプションを選択します。
常に新規のパラメータを作成する	パラメータに置換された値が，前のインスタンスから変わっていない場合も，このルールに応じて新しいパラメータが作成されます。
左の境界インスタンス	一致として考えられる左境界の一致件数です。
長さ	パラメータに保存される，オフセットで始まる文字列の長さ。これを指定しないと，検出された値の末尾までがパラメータになります。
オフセット	検出された値内の文字列のオフセット。
完全一致のみパラメータで置換する	テキストが検出された値に正確に一致する場合にだけ，その値がパラメータで置換されます。
逆方向検索	逆方向に検索されます。

 [トークン置換テスト パッド] ダイアログ・ボックス

相関ルールを適用する前にその相関ルールをテストできます。

利用方法	[**ツール**] > [**記録オプション**] > [**HTTP**] > [**相関**] > [**テスト**]
重要情報	このダイアログ・ボックスは特定のプロトコルにのみ適用されます。プロトコルの完全な一覧と各プロトコルに関連付けられたノードについては，325 ページの「プロトコルの互換性に関する表」を参照してください。

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
	テストが実行されます。
適用されたルール	テスト中に適用されたルールのリスト。
置換用のソース テキスト	置換用のソース・テキストを入力します。
置換結果	テストの結果。

Java の［Classpath］ノード

システムのクラスパス環境変数に含まれていない追加のクラスの場所を指定できます。これらのクラスは，Java アプリケーションの実行や，正しい記録を保障するのに必要な場合があります。

利用方法	**［ツール］>［記録オプション］>［Java 環境の設定］>［Classpath］**
重要情報	このノードは特定のプロトコルにのみ適用されます。プロトコルの完全な一覧と各プロトコルに関連付けられたノードについては，325 ページの「プロトコルの互換性に関する表」を参照してください。

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
	**下向き矢印：**クラスパスのエントリをリストの下方向に移動します。
	**上向き矢印：**クラスパスのエントリをリストの上方向に移動します。
	**［クラスパスの追加］：**クラスパスのリストに新しい行を追加します。

UI 要素	説明
	[削除]：クラスパスを永久に削除します。
[Classpath エントリ] リスト	クラスパスのエントリのリスト。

[Java VM] ノード

Java アプリケーションの記録時に使用する付加的なパラメータを指定できます。

利用方法	[ツール] > [記録オプション] > [Java 環境の設定] > [Java VM]
重要情報	このノードは特定のプロトコルにのみ適用されます。プロトコルの完全な一覧と各プロトコルに関連付けられたノードについては，325 ページの「プロトコルの互換性に関する表」を参照してください。

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
追加の VM パラメータ	ここには Java コマンド・ライン・パラメータが表示されます。任意の Java VM の引数をパラメータとして指定できます。よく使用する引数としては，デバッグ・フラグ（**-verbose**）や，メモリの設定（**-ms**，**-mx**）があります。さらに，**-D** フラグの形式で Java アプリケーションに対してプロパティを渡すこともできます。Java VM フラグの詳細については，JVM のドキュメントを参照してください。
CLASSPATH を -Xbootclasspath パラメータに付加する	クラスパスを **Xbootclasspath** の前に追加する（文字列を挿入する）よう VuGen に指示します。

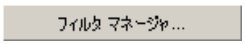
UI 要素	説明
従来の Java VM を使用する	従来の VM のバージョン（Sun の Java Hotspot 以外など）を使用するよう VuGen に指示します。
再生時に指定された追加の VM パラメータを使用する	再生時に同じ追加 VM パラメータを使用するよう VuGen に指示します。

［Microsoft .NET］の［フィルタ］ノード

Microsoft .NET プロトコル用の記録オプションを設定できます。

利用方法	**［ツール］>［記録オプション］>［Microsoft .NET］>［フィルタ］**
重要情報	このノードは特定のプロトコルにのみ適用されます。プロトコルの完全な一覧と各プロトコルに関連付けられたノードについては，325 ページの「プロトコルの互換性に関する表」を参照してください。
関連項目	767 ページの「Microsoft .NET フィルタの概要」 768 ページの「Microsoft .NET フィルタの詳細設定」 770 ページの「Microsoft .NET フィルタの設定についてのガイドライン」

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
	［新規フィルタの作成］ダイアログ・ボックスが開き，新規フィルタを作成できます。詳細については，381 ページの「［新規フィルタの作成］ダイアログ・ボックス」を参照してください。
	フィルタ・マネージャが開き，すべての Microsoft .NET プロトコル・フィルタを表示および変更できます。
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。

UI 要素	説明
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。
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。

［新規フィルタの作成］ダイアログ・ボックス

このダイアログ・ボックスでは，Microsoft .NET スクリプトの新規フィルタを作成できます。

利用方法	**［ツール］>［記録オプション］>［Microsoft .NET］>［フィルタ］>［新規フィルタ］>［作成］**

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
ユーザ定義フィルタを元に作成	カスタム・フィルタに基づいて新規フィルタを作成します。ドロップダウン・メニューを使用して，カスタム・フィルタを選択します。
環境フィルタに基づく	環境フィルタに基づいて新規フィルタを作成します。環境フィルタの横にあるチェック・ボックスを使用して，フィルタのベースとなる環境フィルタを指定します。
空のフィルタから開始	既存のフィルタに基づいていない新規フィルタを作成します。

フィルタ・マネージャ

このダイアログ・ボックスでは，Microsoft .NET フィルタを作成および編集できます。

利用方法	**［ツール］>［記録オプション］>［Microsoft .NET］>［フィルタ］>［環境 / カスタム フィルタ］>［フィルタ マネージャ］**
関連項目	767 ページの「Microsoft .NET フィルタの概要」 768 ページの「Microsoft .NET フィルタの詳細設定」 770 ページの「Microsoft .NET フィルタの設定についてのガイドライン」

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
	選択した要素が包含されます。親ノードを手動で追加すると，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。
	選択した要素が除外されます。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，除外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。

UI 要素	説明
	手動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る必要があることとして，**.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動の包含または除外のルールに優先しません。たとえば，メソッドを手動で**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，771 ページの「包含または除外する要素の指定」を参照してください。
	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。

UI 要素	説明
影響のログ	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，手動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。 影響の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**］>［**影響ログを表示**］を選択します。
<フィルタ・マネージャ・ツリー>	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が示されます。各アイコンの詳細については，次の表を参照してください。 ➤ 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，ユーザの手動によって，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。
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。詳細については，386 ページの「［参照の追加］ダイアログ・ボックス」を参照してください。
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。
新規作成	［新規フィルタの作成］ダイアログ・ボックスが開きます。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。詳細については，381 ページの「［新規フィルタの作成］ダイアログ・ボックス」を参照してください。

UI 要素	説明
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。
保存	フィルタに加えた変更を保存します。
影響ログを表示	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，825 ページの「影響ログの表示」を参照してください。

次の表に，各種の要素を表すフィルタ・マネージャ・ツリーのアイコンを示します。

アイコン	説明		アイコン	説明
	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

アイコン	説明		アイコン	説明
	イベント			構造体
	静的イベント			

 ［参照の追加］ダイアログ・ボックス

このダイアログ・ボックスでは，Microsoft .NET フィルタへの参照を追加できます。

利用方法	**［ツール］＞［記録オプション］＞［Microsoft .NET］＞［フィルタ］＞［環境 / カスタム フィルタ］＞［フィルタ マネージャ］＞［参照の追加］**

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
＜コンポーネント・リスト＞	.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。 ➤ 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。 ➤ 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。
＜選択済みコンポーネント・リスト＞	選択済みのコンポーネントのリスト。［タイプ］カラムは，Public Assemblies フォルダのコンポーネントに対応する［**.NET**］と，［**参照**］を選択して追加したコンポーネントに対応する［**ファイル**］を表します。 ➤ 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。

［Microsoft .NET］の［記録］ノード

Microsoft .NET プロトコル用の記録オプションを設定できます。

利用方法	**［ツール］>［記録オプション］>［Microsoft .NET］>［記録］**
重要情報	このノードは特定のプロトコルにのみ適用されます。プロトコルの完全な一覧と各プロトコルに関連付けられたノードについては，325 ページの「プロトコルの互換性に関する表」を参照してください。

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
コード生成	コード生成中に警告，スタック・トレース，またはすべてのイベント・サブスクリプションを表示するかどうかを指定できます。 ➤ **［警告を表示する］**：コード生成中に発行された警告メッセージを表示します。 ➤ **［スタック トレースを表示する］**：記録されたスタック・トレースが存在する場合，表示されます。 ➤ **［すべてのイベント サブスクリプションを表示する］**：記録されたすべてのイベント・サブスクリプションのコードが生成されます。このオプションが無効な場合，VuGen は，発行者（イベントを発行するオブジェクト）と登録者（イベントの通知を受け取るオブジェクト）の両方がフィルタに追加されているイベントのコードのみを生成します。 **標準設定値**：有効。

UI 要素	説明
デバッグ・オプション	スタックを追跡してサイズを指定できます。 ➤ **[スタック トレース]**：スクリプト内の各呼び出しのスタックの内容を追跡します。これにより，アプリケーションで使用されたクラスおよびメソッドを確認できます。これは，フィルタに追加する参照，名前空間，クラス，またはメソッドを調べるのに便利です。追跡を有効にすると，記録時のアプリケーションのパフォーマンスに影響します。 **標準設定値**：有効。 ➤ **[スタック トレースの制限指定]**：スタックに格納される呼び出しの最大数。呼び出しの数が上限を超えた場合，VuGen はそれらを切り捨てます。 **標準設定値**：20 回。
フィルタ	➤ **[すべてのアセンブリを無視することを標準設定にする]**：選択したフィルタによって明示的に含まれていないすべてのアセンブリが無視されます。このオプションが無効の場合，Vugen は，記録中にロードされる各アセンブリで一致するフィルタ・ルールを検索します。
ログ記録	記録ログ・ファイルに記録される情報の詳細度を設定するログ記録オプション。 ➤ **[ログの重大度]**：ログのレベルは［**エラーのみ**］（標準設定）または［**デバッグ**］に設定します。重大度の設定は以降の説明に従って有効にするすべてのログに適用されます。ログを詳細に記録すると記録時間が大幅に増えることがあるので，HP のサポートによる特別な指示がある場合を除き，必ず［**エラーのみ**］ログを使用してください。 ➤ **[インスツルメンテーション ログ]**：インスツルメンテーション・プロセスに関するメッセージがログに記録されます。 **標準設定値**：有効。 ➤ **[記録ログ]**：記録中に発行されたメッセージがログに記録されます。 **標準設定値**：有効。 ➤ **[コード生成ログ]**：コード生成段階で発行されたメッセージがログに記録されます。 **標準設定値**：有効。

UI 要素	説明
リモート・オブジェクト	このプロパティの詳細については，389 ページの「リモート・オブジェクトのプロパティ」を参照してください。
シリアライズ	➤ **[シリアライズ形式]**：シリアライズをサポートするクラスの記録時に VuGen によって作成されるシリアライズ・ファイルの形式です。[**バイナリ**]，[**XML**]，または[**両方**]を設定できます。バイナリ形式の利点は，圧縮率が高いのより高速であるということです。バイナリ形式の不利な点は，XML と同じようにはデータを操作できない点です。 ➤ **[長配列のシリアライズ]**：シリアル化可能なオブジェクトが含まれる長配列（たとえばプリミティブの配列）には，VuGen のシリアル化メカニズムを使用します。このオプションを有効にすると，配列サイズがしきい値以上の場合に，**LrReplayUtils.GetSerializedObject** 呼び出しが生成されます。 ➤ **[長配列のしきい値]**：**長**配列とみなされる配列のしきい値サイズ。配列サイズがこのサイズ以上の場合，シリアル化可能なオブジェクトが検出されたときにシリアル化されます。 **ヒント**：XML のシリアル化の場合は，XML ファイルの内容を表示できます。ファイルの内容を表示するには，右クリック・メニューから［**XML の表示**］を選択します。

リモート・オブジェクトのプロパティ

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
インプロセス オブジェクトを記録する	サーバがクライアントと同じプロセスでホストされている場合に，クライアントとサーバの間のやり取りが記録されます。このアクションは純粋なクライアント / サーバ・トラフィックではないため，通常は重要ではありません。インプロセス・メソッドが関係する場合，たとえば，特定のエンタープライズ・サービス・アプリケーションでは，このオプションを有効にしてインプロセス・メソッドをキャプチャできます。 **標準設定値**：有効。

UI 要素	説明
非同期呼び出し	リモート・オブジェクトおよびコールバック・メソッドで使用する非同期呼び出しの処理方法を指定します。 ➤ **[標準設定で元のコールバックを呼び出す]**：スクリプトの生成と再生を行うときに，記録されたアプリケーションの元のコールバックを使用します。コールバック・メソッドがフィルタで明示的に除外されている場合は，このオプションが有効になっていてもコールバックは除外されます。 ➤ **[非同期コールバックを生成する]**：このオプションは，元のコールバックが記録されなかった場合に VuGen がコールバックを処理する方法を定義します。 詳細については，763 ページの「非同期呼び出し」を参照してください。
WCF 全二重バインディング	➤ **[ダミー コールバック ハンドラを生成]**：次のアクションを実行して，全二重通信の元のコールバックをダミー・コールバックで置き換えます。 ➤ **引数の格納**：サーバは，再生中にハンドラを呼び出した場合，そのメソッドの引数をメモリ・マップのキー値に保存します。 ➤ **再生の同期化**：次の応答が到着するまで，スクリプトの実行が停止されます。VuGen は，記録中にコールバックが発生したポイントに同期化を配置します。スクリプトでは警告で示されます。 ➤ **[一意のクライアント ベース アドレスを生成]**：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。このオプションでは，元のクライアント・ベース・アドレスのポート番号が一意のポート番号に置き換わります。 WCF 全二重バインディングの参考情報については，756 ページの「WCF 双方向通信の記録」を参照してください。

［ネットワーク］の［ポート マッピング］ノード

ポート・マッピングの記録オプションを設定できます。

利用方法	**［ツール］>［記録オプション］>［ネットワーク］>［ポート マッピング］**
重要情報	このノードは特定のプロトコルにのみ適用されます。プロトコルの完全な一覧と各プロトコルに関連付けられたノードについては，325 ページの「プロトコルの互換性に関する表」を参照してください。

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
エントリの新規作成	［サーバ エントリ］ダイアログ・ボックスが開き，新しいマッピングを追加できます。ユーザ・インタフェースの詳細については，392 ページの「［サーバ エントリ］ダイアログ・ボックス」を参照してください。
エントリの編集	［サーバ エントリ］ダイアログ・ボックスが開き，選択したエントリを編集できます。ユーザ・インタフェースの詳細については，392 ページの「［サーバ エントリ］ダイアログ・ボックス」を参照してください。
オプション(O)...	［詳細設定］ダイアログ・ボックスが開き，通信プロトコルと SSL レベルの自動検出を有効にできます。ユーザ・インタフェースの詳細については，395 ページの「［ポート マッピングの詳細設定］ダイアログ・ボックス」を参照してください。
< ポート・マッピング・リスト >	ポート・マッピングのリスト。各項目のチェック・ボックスをクリアすることで，一時的に割り当てを無効にできます。割り当てを無効にすると，VuGen は無効にした「サーバ：ポート」の組み合わせに割り当てられた，すべてのトラフィックを無視します。データが無関係な場合やプロトコルがサポートされていない場合は，ポートの割り当てを無効にしてください。

UI 要素	説明
キャプチャ レベル	キャプチャするデータのレベル（HTTP ベースのプロトコルの場合のみ）。 ➤ **[ソケット レベル データ]**：トラップを使用してソケット・レベルのデータだけがキャプチャされます。この場合はポート割り当てが適用されます（標準設定）。 ➤ **WinINet レベル データ**：特定の HTTP アプリケーションで使用されている **WinINet.dll** API のフックを使ってデータがキャプチャされます。このようなフックを使用している最も一般的なアプリケーションは Internet Explorer です。このレベルにはポート割り当てを適用できません。 ➤ **[ソケット レベルおよび WinINet レベルのデータ]**：両方のメカニズムを使用してデータがキャプチャされます。WinINet レベルでは，**WinINet.dll** を使用しているアプリケーションの情報が送信されます。ソケット・レベルでは，**WinINet.dll** から発生したのではないと判断された場合にのみデータが送信されます。ポート・マッピングは，**WinINet.dll** から発生していないデータに適用されます。
次のネットワーク レベルのサーバ アドレス割り当て	プロトコルごとの割り当てを表示するよう指定します。たとえば，FTP の割り当てだけを表示したい場合には［FTP］を選択します。

［サーバ エントリ］ダイアログ・ボックス

［ネットワーク］の［ポート マッピング］ノードのサーバ・リストからサーバを定義できます。

利用方法	**［ツール］>［記録オプション］>［ネットワーク］>［ポート マッピング］>［エントリの新規作成 / エントリの編集］**
重要情報	このダイアログ・ボックスは特定のプロトコルにのみ適用されます。プロトコルの完全な一覧と各プロトコルに関連付けられたノードについては，325 ページの「プロトコルの互換性に関する表」を参照してください。

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
次のローカル ポートからターゲット サーバに転送する	特定のポートからのすべてのトラフィックが別のサーバに転送されます。このオプションは，特別な UNIX マシンの場合など，クライアント上で VuGen を正常に実行できない場合や，VuGen を介してアプリケーション・サーバを起動できない場合に特に役立ちます。クライアント・マシンに問題がある場合は，そのマシンからのトラフィックを捕獲して，サーバに渡すよう VuGen の設定を行います。こうすることで，VuGen はデータを処理し，アクションに対応するコードを生成することが可能となります。 **例 :** host1 という UNIX のクライアントが，サーバ server1 と，ポート番号 8080 経由で通信しており，［ポート マッピング］には，server1，ポート 8080 のエントリが設定されていたとします。［サーバ エントリ］ダイアログ・ボックスの［トラフィックの転送］セクションで，［**次のローカル ポートからターゲット サーバに転送する**］チェック・ボックスを選択して，トラフィックの転送を有効にします。トラフィックの転送に使用するポート（この例では 8080）を指定します。 次にクライアントを server1 ではなく，VuGen を実行している host1 に接続します。VuGen はクライアント・マシンからの通信を受信し，その通信をローカルのポート 8080 を経由してサーバに転送します。トラフィックは VuGen を経由するため，トラフィックの分析と適切なコードの生成が可能となります。
接続タイプ	接続のセキュリティ・レベル。［**非認証**］，［**SSL**］，［**自動**］があります。［自動］を選択すると，レコーダによって SSL シグネチャについて最初の 4 バイトが検査されます。SSL 署名を検出すると，SSL が使用されていると推定されます。

UI 要素	説明
ポート	エントリ項目に登録するターゲット サーバのポート番号。「**0**」を入力すると，すべてのポートが指定されます。 設定したポートとサーバの名前が全部ではない場合，VuGen は次の優先順位に従ってデータをサービスに割り当てます。 ➤ 優先度 1：指定したポートとサーバ ➤ 優先度 2：指定していないポート，指定したサーバ ➤ 優先度 3：指定したポート，指定していないサーバ ➤ 優先度 4：指定していないポートとサーバ 優先順位の高い割り当てがある場合に，それよりも優先順位の低い割り当てを指定しても，優先順位の低い割り当ては無効です。たとえば，twilight というサーバのポート番号 25 番のトラフィックを SMTP として扱うよう指定した後で，すべてのサーバのポート 25 番を HTTP として扱うよう指定しても，データは SMTP として扱われます。 ➤ **強制割り当て**：ポート番号，サーバ名，または「サーバ：ポート」の組み合わせの割り当てを指定した場合，VuGen では，ネットワーク・トラフィックがそのサービスを使用するよう強制されます。たとえば，「< 任意のサーバ >」のポート 80 番が FTP を使用するよう指定した場合，VuGen では，実際の通信が HTTP であったとしても，FTP プロトコルを使用してその通信が記録されます。この例では，仮想ユーザ・スクリプトは空となる可能性があります。
レコード タイプ	記録のタイプ（直接か，それともプロキシ・サーバを経由するか）。
サービス ID	接続のタイプを識別するためにレコーダが使用するプロトコルまたはサービス名（HTTP，FTP など）。新しい名前を指定することもできます。指定できる名前の長さは最長 8 文字です。
サービス タイプ	現在は TCP/IP に設定されています。
SSL 暗号	リモートのセキュア・サーバに接続するのに使用する SSL 暗号を指定します。

UI 要素	説明
SSL バージョン	クライアント・アプリケーションおよびサーバとの通信に使用する SSL のバージョン。 **標準設定値**：SSL 2/3。ただし，サービスによっては SSL 3.0 または SSL 2.0 のみが必要になることがあります。新しいワイヤレス・アプリケーションでは新しいセキュリティ・アルゴリズム，TLS 1.0 を必要とします。
ターゲット サーバ	エントリ項目に登録するターゲット サーバの IP アドレスまたはホスト名。 **標準設定値**：すべてのサーバ。
指定されたクライアント側証明書を使用する	リモート・サーバに接続する際に使用する標準のクライアント側の証明書を指定します。**txt**，**crt**，**pem** 形式のいずれかの証明書を指定するか一覧から探し，パスワードを入力します。
指定されたプロキシ サーバ証明書を使用する	サーバの証明書を要求するクライアント・アプリケーションに示す標準の証明書を指定します。**txt**，**crt**，**pem** 形式のいずれかの証明書を指定するか一覧から探し，パスワードを入力します。［**SSL のテスト**］をクリックすると，サーバに対する認証情報をチェックできます。

［ポート マッピングの詳細設定］ダイアログ・ボックス

ポート・マッピングの詳細設定を設定できます。詳細については，314 ページの「ポート・マッピングの自動検出」を参照してください。

利用方法	**［ツール］> ［記録オプション］> ［ネットワーク］> ［ポート マッピング］> ［オプション］**
重要情報	このダイアログ・ボックスは特定のプロトコルにのみ適用されます。プロトコルの完全な一覧と各プロトコルに関連付けられたノードについては，325 ページの「プロトコルの互換性に関する表」を参照してください。

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
SOCKET ベース コミュニケーションの自動検出を有効にする	自動的に通信のタイプが検出されます。必要な場合，VuGen によるプロトコルの検出が成功するまで 1 つずつ［移行しきい値の送受信］の最大数を増やします。また，VuGen によるプロトコルの検出が成功するまで，一度に 1024 バイト（1KB）ずつ［バッファサイズしきい値の送受信］の最大数を増やすこともできます。これらを行うと VuGen によるシグネチャのためにより多くのデータを調査することができるようになります。
自動 SSL 検出を有効にする	SSL 通信が自動的に検出されます。検出したい SSL のバージョンと標準の SSL 暗号の形式を指定します。これはポートの割り当てが，［**接続タイプ**］ボックスで［**自動**］に指定されているか，まったく指定のない場合にだけ適用されることに注意してください。サーバ，ポート，もしくは「サーバ：ポート」の組み合わせが，［**非認証**］，［**SSL**］のいずれかに指定されている場合には，自動 SSL 検出は適用されません。
ログのレベル	自動ソケット検出のログ・レベルを設定します。

［RDP］の［コードの生成 - 高度］ノード

VuGen による RDP スクリプトの作成方法を制御できます。この設定は上級ユーザが変更することをお勧めします。

利用方法	**［ツール］>［記録オプション］>［RDP］>［コード生成 - 高度］**
重要情報	このノードは特定のプロトコルにのみ適用されます。プロトコルの完全な一覧と各プロトコルに関連付けられたノードについては，325 ページの「プロトコルの互換性に関する表」を参照してください。

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
クリップボード パラメータの相関	ユーザが送信した記録済みクリップボード・テキストが，サーバから受信した同一のテキストを含む相関パラメータに置換されます。
ダブルクリックのタイムアウト (ミリ秒)	ダブルクリックとして認識されるための，マウス・ボタンの 2 つの連続クリック間の最大時間（ミリ秒単位)。 **標準設定値**：500 ミリ秒。
クリップボード パラメータのプレフィックス	現在のスクリプトで生成されるクリップボード・パラメータのプレフィックス。これは，スクリプトを結合する場合に便利です。スクリプトごとに異なるプレフィックスを指定できます。**標準設定値**：**ClipboardDataParam_**。
スナップショット名のプレフィックス	現在のスクリプトで生成されるスナップショット・ファイルの名前のプレフィックス。これは，スクリプトを結合する場合に便利です。スクリプトごとに異なるプレフィックスを指定できます。 **標準設定値**：**snapshot_**。

［RDP］の［コード生成 - エージェント］ノード

記録中に Microsoft Agent for Terminal Server エージェントと VuGen が連携する方法を制御できます。

利用方法	**［ツール］>［記録オプション］>［RDP］>［コード生成 - エージェント］**
重要情報	このノードは特定のプロトコルにのみ適用されます。プロトコルの完全な一覧と各プロトコルに関連付けられたノードについては，325 ページの「プロトコルの互換性に関する表」を参照してください。

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
RDP エージェント ログの有効化	RDP エージェント・ログを有効にします。 ➤ **[RDP エージェント ログ詳細レベル]**：RDP エージェント・ログで生成される詳細のレベルを設定します。[**標準設定**] を指定すると最も低いレベルになり，[**拡張デバッグ**] を指定すると最も高いレベルになります。 ➤ **[RDP エージェント ログの出力先]**：RDP エージェント・ログ・データの出力先を設定します。[**File**] を指定すると，リモート・サーバ側にのみログ・メッセージが保存されます。[**Stream**] を指定すると，Vugen マシンにログ・メッセージが送信されます。[**FileAndStream**] を指定すると，両方の宛先にログ・メッセージが送信されます。 ➤ **[RDP エージェント ログ フォルダ]**：RDP エージェント・ログ・ファイルが生成されるリモート・サーバ上のフォルダのパス。
RDP エージェントの使用法	記録セッション中に RDP エージェントが収集した情報を使用してスクリプトが生成されます。サーバに LoadRunner RDP エージェントがインストールされている必要があります。

［RDP］の［コードの生成 - 基本］ノード

VuGen がスクリプトを作成する方法，つまり詳細レベル，トリガ，タイムアウトを制御できます。

利用方法	**［ツール］>［記録オプション］>［RDP］>［コード生成 - 基本］**
重要情報	このノードは特定のプロトコルにのみ適用されます。プロトコルの完全な一覧と各プロトコルに関連付けられたノードについては，325 ページの「プロトコルの互換性に関する表」を参照してください。

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
常に接続名を生成する	有効になっているときは，関数呼び出しに **ConnectionName** パラメータが含まれます。無効になっているときは，スクリプトに複数の **rdp_connect_server** が現れている場合でも，関数にはこのパラメータしか含まれないようになります。 **標準設定値**：有効。
同期化ポイントの自動生成	同期化ポイントによって，ポップアップやそれ以外の制御の対象となるウィンドウまたはダイアログ・ボックスが特定の状態になるのを待っている間，スクリプトの再生を一時停止できます。このオプションにより，マウスのクリックおよびドラッグの前に自動的に **sync_on_image** 関数が生成されます（標準設定では有効）。［**同期化半径**］は，マウス操作から，同期化領域を定義する四角形の端までの距離です。標準設定値は 20 ピクセルです。次のオプションのいずれかを選択します。 ➤ **［なし］**：同期化ポイントは自動的に追加されません。 ➤ **［四角形］**：クリックまたはドラッグした位置を中心とした四角形の同期化ポイントが作成されます。 ➤ **［拡張］**：必要な場所（たとえばボタン）のみ選択されるように，また，UI の変更（たとえばボタンの移動）に反応するように設計された同期化ポイントが作成されます。同期化領域が認識されないときは，四角形の同期化設定が使用されます。
マウス移動呼び出しの生成	スクリプトに **rdp_mouse_move** 呼び出しが生成されます。このオプションを有効にすると，スクリプト・サイズが非常に大きくなります。 **標準設定値**：有効。
未処理のキーボード呼び出しの生成	スクリプト・レベルが［**未処理**］に設定されているかのように **rdp_raw_key_up/down** 呼び出しが生成されます。マウス呼び出しは，スクリプト・レベルのとおりに生成されます。無効になっているときは，スクリプト・レベルのとおりにキーボード呼び出しが生成されます。スクリプト・レベルが［**未処理**］に設定されている場合，このオプションは無視されます。 **標準設定値**：有効。

UI 要素	説明
未処理のマウス呼び出しの生成	スクリプト・レベルが［未処理］に設定されているかのように **rdp_mouse_button_up/down** 呼び出しが生成されます。キーボード呼び出しは，スクリプト・レベルのとおりに生成されます。無効になっているときは，スクリプト・レベルのとおりにマウス呼び出しが生成されます。スクリプト・レベルが［**未処理**］に設定されている場合，このオプションは無視されます。 **標準設定値**：有効。
スクリプト生成レベル	スクリプト生成時に使用されるスクリプト・レベルおよび API 関数タイプ。 ➤ **［高い］**：高レベルのスクリプトが生成されます。キーボード・イベントは **rdp_type** 呼び出しに変換されます。同じ座標での連続する 2 回のマウス・クリックは，ダブルクリックと解釈されます。 ➤ **［低い］**：低レベルのスクリプトが生成されます。キーの解放 / 押下イベントは **rdp_key** イベントに変換されます。修飾キー（Alt，Ctrl，Shift）は，ほかの関数の **KeyModifier** パラメータとして使用されます。マウスの解放 / 押下 / 移動イベントは，マウスのクリック / ドラッグ・イベントに変換されます。 ➤ **［未処理］**：ネットワーク・バッファから入力イベントを抽出し，最も単純な形（キーの解放 / 押下，マウスの解放 / 押下 / 移動）の呼び出しを生成することによって，未処理レベルのスクリプトが生成されます。**KeyModifier** パラメータは使用されません。

［RDP］の［ログイン］ノード

RDP ログインの記録オプションを設定できます。

利用方法	［ツール］>［記録オプション］>［RDP］>［ログイン］
重要情報	このノードは特定のプロトコルにのみ適用されます。プロトコルの完全な一覧と各プロトコルに関連付けられたノードについては，325 ページの「プロトコルの互換性に関する表」を参照してください。

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
RDP クライアント アプリケーションの実行	ターミナル・サービス・クライアントを実行してターミナル・サーバに接続します。
カスタム接続ファイルを使用	既存の接続ファイルを使用してターミナル・サーバに接続します。このファイルには ***.rdp** 拡張子が必要です。ファイル・システムおよびネットワークのファイルを参照できます。
標準設定接続ファイルを使用	ドキュメント・ディレクトリにある **Default.rdp** ファイルを使用してターミナル・サーバに接続します。

［記録のプロパティ］の［Corba オプション］ノード

CORBA 固有の記録プロパティといくつかのコールバック・オプションを設定できます。

利用方法	**［ツール］>［記録オプション］>［記録のプロパティ］>［Corba オプション］**
重要情報	このノードは特定のプロトコルにのみ適用されます。プロトコルの完全な一覧と各プロトコルに関連付けられたノードについては，325 ページの「プロトコルの互換性に関する表」を参照してください。

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
コールバック接続の記録	各コールバック・オブジェクトについて，ORB への接続のための接続ステートメントを生成するよう VuGen に指示します。 **標準設定値**：有効。
DLL のみ記録	DLL レベルのみで記録するよう VuGen に指示します。 **標準設定値**：有効。

UI 要素	説明
プロパティを記録する	プロトコルに関連するシステム・プロパティとカスタム・プロパティを記録するよう VuGen に指示します。 **標準設定値**：有効。
CORBA オブジェクトを解決する	相関によって，CORBA オブジェクトが解決できなかった場合に，バイナリ・データを使って解決されます。 **標準設定値**：有効。
IDL コンストラクトを表示する	パラメータを CORBA の呼び出しに対して渡すために使用される，インタフェース定義言語（IDL）の構成要素が表示されます。 **標準設定値**：有効。
ローカル ベンダ クラスを使用する	ローカル・ベンダ・クラスを使用して，**srv** フォルダが BOOT クラスパスに追加されます。このオプションを無効にした場合は，VuGen がネットワーク・クラスを使用して，スクリプトのクラスをクラスパスに追加します。 **標準設定値**：有効。
ベンダ	CORBA ベンダ（**Inprise Visibroker**，**Iona OrbixWeb**，または **Bea Weblogic**）。

［記録のプロパティ］の［相関オプション］ノード

自動相関を有効にし，その深さを制御できます。

利用方法	**［ツール］＞［記録オプション］＞［記録のプロパティ］＞［相関オプション］**
重要情報	このノードは特定のプロトコルにのみ適用されます。プロトコルの完全な一覧と各プロトコルに関連付けられたノードについては，325 ページの「プロトコルの互換性に関する表」を参照してください。

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
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できます。このタイプの相関は，「深い」相関としても知られています。 **標準設定値**：有効。
コレクション タイプを相関する	JDK 1.2 以降の Collection クラスのオブジェクトが相関されます。 **標準設定値**：有効。
文字列配列を相関する	記録中に，文字列配列内の文字列が相関されます。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。 **標準設定値**：有効。
文字列を相関する	記録中にスクリプト内の文字列が相関されます。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，ほかのすべての相関オプションが無視されます。 **標準設定値**：有効。
相関レベル	相関のレベルの深さを，スキャン対象の内部コンテナの数として指定します。 **標準設定値**：15。

［記録のプロパティ］の［ログ オプション］ノード

記録中に生成されるデバッグ情報のレベルを指定できます。

利用方法	**［ツール］＞［記録オプション］＞［記録のプロパティ］＞［ログ オプション］**
重要情報	このノードは特定のプロトコルにのみ適用されます。プロトコルの完全な一覧と各プロトコルに関連付けられたノードについては，325 ページの「プロトコルの互換性に関する表」を参照してください。

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
クラスのダンプ	ロードしたすべてのクラスがスクリプト・ディレクトリにダンプされます。 **標準設定値**：有効。
計算の概要を表示する	記録されたすべてのオブジェクトのダイジェストが生成されます。 **標準設定値**：有効。 ➤ **[概要から除外]**：ダイジェスト計算に含めないオブジェクトの一覧です。 **構文**：java.lang.Object class format, delimiter = ","
ログのレベル	生成する記録ログのレベル。 ➤ **[なし]**：ログ・ファイルは作成されません。 ➤ **[簡略]**：標準記録ログおよび出力リダイレクションが生成されます。 ➤ **[詳細]**：メソッド，引数，および戻り値に関する詳細ログが生成されます。 ➤ **[デバッグ]**：上記のすべての情報に加えて，フックおよび記録のデバッグ情報が記録されます。
スレッドを同期化する	マルチ・スレッド・アプリケーションの場合に，各スレッド間の同期をとるよう VuGen に指示します。 **標準設定値**：有効。

［記録のプロパティ］の［レコーダ オプション］ノード

記録する Java プロトコルやほかのプロトコルに固有の記録オプションを設定できます。

利用方法	**［ツール］>［記録オプション］>［記録のプロパティ］>［レコーダ オプション］**
重要情報	このノードは特定のプロトコルにのみ適用されます。プロトコルの完全な一覧と各プロトコルに関連付けられたノードについては，325 ページの「プロトコルの互換性に関する表」を参照してください。

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
バイト配列の形式	スクリプト内のバイト配列の形式で，［**標準**］，［**展開済みシリアル化オブジェクト**］，または［**未展開のシリアル化オブジェクト**］から選択できます。非常に長いバイト配列を記録する場合は，シリアル化オブジェクト・オプションのいずれかを使用します。 **標準設定値**：**標準**。
バイト形式オプション	可読文字が，必要なキャスティングを行うことによって，バイトまたは 16 進形式ではなく文字として表示されます。 **標準設定値**：有効。
コメント行の内容	指定した文字列のいずれか 1 つを含むスクリプト行をすべてコメントアウトします。複数の文字列を指定するには，項目をカンマで区切ります。 **標準設定値**：<undefined> を含む文字列がある行がコメントアウトされます。
拡張子リスト	サポートされているすべての拡張子のカンマ区切りリスト。各拡張子には，固有のフック・ファイルがあります。 **標準設定値**：JNDI。
関数チェックを挿入する	再生中に受け取った戻り値を，記録中に生成された戻り値の期待値と比較する検証コードが挿入されます。このオプションは，プリミティブ型の戻り値にのみ適用されます。 **標準設定値**：有効。
クラスの前に親クラスを読み込む	子クラスの前に親クラスが読み込まれるように，読み込みの順序が変更されます。これは，継承が深いツリーのフックを特定するのに役立ちます。 **標準設定値**：有効。
LoadRunner コールバックを記録する	LoadRunner スタブ・オブジェクトがコールバックとして記録されます。無効にした場合は，元のクラスがコールバックとして記録されます。 **標準設定値**：有効。
記録済みプロトコル	記録するプロトコル（RMI，CORBA，JMS，または Jacada）を指定します。 **標準設定値**：RMI。

UI 要素	説明
削除する行の内容	指定した文字列のいずれか 1 つを含むスクリプト行をすべて削除します。複数の文字列を指定するには，項目をカンマで区切ります。この機能は，テストの目的に応じてスクリプトをカスタマイズするときに役立ちます。
解読不可能な文字列をバイトとして扱う	不可読文字を含む文字列がバイトの配列として表されます。このオプションは，パラメータとして呼び出しに渡される文字列に適用されます。 **標準設定値**：有効。
_JAVA_OPTIONS フラグを使用する	Version 1.2 以降の JVM に対して，目的の JVM パラメータを指定する _JAVA_OPTION 環境変数を使用するようにします。 **標準設定値**：有効。
DLL フックを使用して LoadRunner サポートをアタッチする	DLL フックを使用して，LoadRunner サポートが自動的に任意の JVM にアタッチされます。

［記録のプロパティ］の［シリアル化オプション］ノード

オブジェクトのシリアル化の方法を制御できます。多くの場合，シリアル化は，オブジェクト値のパラメータ化のために ASCII 形式でオブジェクトを表示する際に適用されます。

利用方法	**［ツール］>［記録オプション］>［記録のプロパティ］>［シリアル化オプション］**
重要情報	このノードは特定のプロトコルにのみ適用されます。プロトコルの完全な一覧と各プロトコルに関連付けられたノードについては，325 ページの「プロトコルの互換性に関する表」を参照してください。
関連項目	198 ページの「Java スクリプトの相関 - シリアル化」

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
未展開のシリアル化オブジェクト	シリアル化されたオブジェクトが ASCII 形式で展開され，パラメータ化を行うために，オブジェクトの ASCII 値を表示できます。 ➤ **[オブジェクト サイズを制限する]**：シリアル化可能なオブジェクトのサイズが指定した値に制限されます。指定した値を超えるオブジェクトは，ASCII 形式でスクリプト内に表現されません。 **標準設定値**：3072 バイト。 ➤ **[シリアル化オブジェクトを無視する]**：記録されたスクリプトの中で遭遇しても展開しないシリアル化オブジェクトが示されます。オブジェクトはカンマで区切ります。 **構文**：java.lang.Object class format, delimiter = "," ➤ **[シリアル化区切り文字]**：ASCII 形式のオブジェクトで要素を区切る区切り文字が示されます。VuGen では，これらの区切り文字に囲まれた文字列のみがパラメータ化されます。標準設定は「#」です。 ➤ **[配列を展開する]**：シリアル化されたオブジェクトの配列の要素が ASCII 形式に展開されます。このオプションを無効にし，オブジェクトに配列が含まれている場合，オブジェクトは展開されません。 **標準設定値**：有効。つまり，シリアル化解除されたオブジェクトはすべて展開されます。 ➤ **[配列エントリの制限]**：指定した数より多くの要素が含まれる配列が展開されないようレコーダに指示します。**標準設定値**：200。

［RTE］の［設定］ノード

ターミナル・エミュレーション中に使用される文字セットに合わせて，記録オプションを設定できます。

利用方法	**［ツール］>［記録オプション］>［RTE］>［設定］**
重要情報	このノードは特定のプロトコルにのみ適用されます。プロトコルの完全な一覧と各プロトコルに関連付けられたノードについては，325 ページの「プロトコルの互換性に関する表」を参照してください。

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
文字セット	ターミナル・エミュレーション中に使用される文字セットと一致します。標準設定の文字セットは ANSI です。漢字その他のマルチバイト・プラットフォームの場合は，DBCS（ダブルバイト文字セット）を指定できます。

RTE ノード

一般的な RTE の記録オプションを設定できます。

利用方法	**［ツール］>［記録オプション］>［RTE］>［RTE］**
重要情報	このノードは特定のプロトコルにのみ適用されます。プロトコルの完全な一覧と各プロトコルに関連付けられたノードについては，325 ページの「プロトコルの互換性に関する表」を参照してください。

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
自動同期化コマンドを作成	記録中にいくつかの **TE 同期化**関数が自動的に生成され，それらがスクリプトに挿入されます。 ➤ **[カーソル]**：それぞれの **TE_type** 関数の前に **TE_wait_cursor** 関数が生成されます。 ➤ **[プロンプト]**：それぞれの **TE_type** 関数の前に **TE_wait_text** 関数が生成されます（適切な場合）。 ➤ **[X- システム]**：記録中に新しい画面が表示されるたびに，**TE_wait_sync** 関数が生成されます。 **注**：VT タイプのターミナルのみを記録しているときは，意味のある **TE_wait_text** 関数が生成されます。ブロック・モード（IBM）ターミナルを記録しているときは，**TE_wait_text** 関数の自動生成は使用しないでください。
X- システム用自動トランザクションを作成する	シナリオの実行中に，システムが X SYSTEM モードになっていた時間が記録されます。この場合，それぞれの **TE_wait_sync** 関数の後に **TE_wait_sync_transaction** 関数が挿入されます。**TE_wait_sync_transaction** 関数はそれぞれ，**default** という名前を持つトランザクションを作成します。**TE_wait_sync_transaction** 関数はそれぞれ，システムが以前の X SYSTEM 状態を保っていた時間を記録します。
スクリーン ヘッダでコメントを作成する	仮想ユーザ・スクリプトの記録中に画面ヘッダ・コメントが自動的に生成され，それらのコメントがスクリプトに挿入されます。生成されるコメントには，ターミナル・エミュレータ・ウィンドウの 1 行目に表示されているテキストが格納されます。 **注**：コメントの自動生成は，IBM 5250 などのブロック・モードのターミナル・エミュレータを使用しているときのみ可能です。
キーボード記録タイムアウト	記録中にターミナル・エミュレータにテキストを入力すると，テキスト入力が VuGen によって監視されます。各キーストロークの後，VuGen は次のキーストロークが発生するまで，指定の時間だけ待ちます。指定の時間内に次のキーストロークが発生しなかった場合は，コマンドが完了したものとみなされます。

［SAPGUI］の［自動ログオン］ノード

記録開始時に自動的にログオンできます。ログオン関数はスクリプトの vuser_init セクションに置かれます。

利用方法	**［ツール］>［記録オプション］>［SAPGUI］>［自動ログオン］**
重要情報	このノードは特定のプロトコルにのみ適用されます。プロトコルの完全な一覧と各プロトコルに関連付けられたノードについては，325 ページの「プロトコルの互換性に関する表」を参照してください。

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
自動ログオンを有効にする	記録開始時に自動的にログオンできます。SAP サーバの［**サーバ名**］，［**ユーザ**］，［**パスワード**］，［**クライアント**］名，およびインタフェースの［**言語**］を入力します。

［SAPGUI］の［コード生成］ノード

SAPGUI プロトコルのコード生成を設定できます。

利用方法	**［ツール］>［記録オプション］>［SAPGUI］>［コード生成］**
重要情報	このノードは特定のプロトコルにのみ適用されます。プロトコルの完全な一覧と各プロトコルに関連付けられたノードについては，325 ページの「プロトコルの互換性に関する表」を参照してください。

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
ヘッダ ファイルに常にオブジェクト ID を生成する	オブジェクト ID がスクリプトではなく個別のヘッダ・ファイルに置かれます。このオプションを無効にすると，一般スクリプト設定で指定された文字列の長さに従って ID が生成されます。これにより，スクリプトがよりコンパクトで読みやすくなります。
データ設定ステップを生成する	各セルに対して個別のデータ挿入ステップが生成されるのではなく，テーブルおよびグリッド・コントロールに対してデータ挿入ステップが生成されます。
ログオン操作を単一ステップとして生成	すべてのログオン操作に対して，単一の **sapgui_logon** メソッドが生成されます。これによって，コードが簡略化されます。ログインで問題が生じた場合は，このオプションを無効にします。

［SAPGUI］の［一般］ノード

SAPGUI プロトコルの一般的な記録オプションを設定できます。

利用方法	**［ツール］>［記録オプション］>［SAPGUI］>［一般］**
重要情報	このノードは特定のプロトコルにのみ適用されます。プロトコルの完全な一覧と各プロトコルに関連付けられたノードについては，325 ページの「プロトコルの互換性に関する表」を参照してください。

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
画面ショットのキャプチャ	記録中に表示される SAPGUI 画面のスナップショットの保存方法を指定します（[**ActiveScreen のスナップショット**]，[**通常のスナップショット**]，[**なし**]）。ActiveScreen のスナップショットは記録後に双方向性と画面情報を提供しますが，より多くのリソースを必要とします。
記録中にイベントを変更する	➤ **[テキストによるショートカット メニューのプロセス]**：テキストによってショートカット・メニューが処理され，**sapgui_toolbar_select_context_menu_item_by_text** 関数が生成されます。このオプションが無効のときは，ID によってショートカット・メニューが処理され，ショートカット・メニューに対して **sapgui_toolbar_select_context_menu_item** が生成されます。これは，日本語文字を使用している場合に役立ちます。

Silverlight サービス・ノード

Silverlight プロトコル・スクリプトの WSDL ファイルを管理できます。

利用方法	**[ツール] > [記録オプション] > [Silverlight] > [サービス]**
重要情報	このノードは特定のプロトコルにのみ適用されます。プロトコルの完全な一覧と各プロトコルに関連付けられたノードについては，325 ページの「プロトコルの互換性に関する表」を参照してください。
関連タスク	883 ページの「WSDL ファイルをインポートする方法」

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
< サービス・リスト >	インポートされた WSDL ファイルとその場所のリスト。ツールバーを使用して，WSDL ファイルの追加，削除，および編集ができます。また，［プロトコルおよびセキュリティ データ］ボタンを選択して，プロトコルとセキュリティ・データを編集できます。
WSDL ファイルを自動的に検出してコード生成中にサービスをインポートする	スクリプトで使用される WSDL ファイルが自動的に特定されてインポートされます。
WSDL ファイルを使用しない	WSDL ファイルがスクリプトで無効になり，代わりに SOAP 要求が生成されます。これにより，スクリプトのレベルは下がりますが，一般的にスクリプトのパフォーマンスは向上します。
サービス エンドポイント	特定の WSDL を使用できるエンドポイントの場所。
スクリプト内に含まれる WSDL ファイルを使用する	自動または手動でインポートされた WSDL ファイルを有効にできます。
WSDL の位置	選択した WSDL ファイルの場所。

［サービスの追加 / 編集］ダイアログ・ボックス

WSDL を特定して Silverlight プロトコル・スクリプトにインポートできます。

利用方法	**［ツール］>［記録オプション］>［Silverlight］>［サービス］>［追加］**
重要情報	このノードは特定のプロトコルにのみ適用されます。プロトコルの完全な一覧と各プロトコルに関連付けられたノードについては，325 ページの「プロトコルの互換性に関する表」を参照してください。
関連タスク	883 ページの「WSDL ファイルをインポートする方法」

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
接続の設定	［接続の設定］ダイアログ・ボックスが開き，指定した WSDL ファイルのプロキシ情報と認証情報を設定できます。ユーザ・インタフェースの詳細については，414 ページの「［接続の設定］ダイアログ・ボックス」を参照してください。
WSDL を次から選択	➤ **［URL］**: URL を指定して WSDL を選択します。 ➤ **［ファイル］**: ローカル・パスを指定して WSDL を選択します。 ➤ **［インポート済み］**: WSDL 履歴（インポート済み WSDL ファイルのリスト）から WSDL を選択します。
サービス エンドポイント	特定の WSDL を使用できるエンドポイントの場所。
WSDL の位置	WSDL の URL またはローカル・パス。

［接続の設定］ダイアログ・ボックス

Silverlight プロトコル・スクリプトの WSDL ファイルのプロキシ情報と認証情報を設定します。

利用方法	**［ツール］>［記録オプション］>［Silverlight］>［サービス］>［追加］>［接続の設定］**
重要情報	➤ このノードは特定のプロトコルにのみ適用されます。プロトコルの完全な一覧と各プロトコルに関連付けられたノードについては，325 ページの「プロトコルの互換性に関する表」を参照してください。 ➤ これらの設定が関連するのは，WSDL ファイルのインポートのみです。再生中に使用する WSDL ファイルの認証情報とプロキシ情報を設定するには，必要な値を指定した **web_set_user_step** 関数を追加します。
関連タスク	883 ページの「WSDL ファイルをインポートする方法」

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
認証	［**認証設定を使用する**］を選択してユーザ名とパスワードを入力し，認証設定を有効にします。
プロキシ	［**プロキシ設定を使用する**］を選択してユーザ名，パスワード，サーバ，およびポート番号を入力し，プロキシ設定を有効にします。

［プロトコルおよびセキュリティ シナリオ データ］ダイアログ・ボックス

プロトコルおよびセキュリティ・シナリオ・データを設定できます。

利用方法	**［ツール］>［記録オプション］>［Silverlight］>［サービス］>［プロトコルおよびセキュリティ データ］ボタン**
重要情報	➤ このノードは特定のプロトコルにのみ適用されます。プロトコルの完全な一覧と各プロトコルに関連付けられたノードについては，325 ページの「プロトコルの互換性に関する表」を参照してください。 ➤ これらの設定が関連するのは，WSDL ファイルのインポートのみです。再生中に使用する WSDL ファイルの認証情報とプロキシ情報を設定するには，必要な値を指定した **web_set_user_step** 関数を追加します。 ➤ このダイアログ・ボックスの設定は，コード生成中にリセットされます。
関連タスク	883 ページの「WSDL ファイルをインポートする方法」

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
ポート	WSDL バインディングの個々のエンドポイント。 **注：**別のポートを選択すると，このダイアログ・ボックスの値は最後に保存された値にリセットされます。まだ保存されていない変更はすべてリセットされます。
説明	選択したセキュリティ・シナリオの説明。
トランスポート	VuGen がサーバにサービス要求を送信するために使用するトランスポート層のプロトコル。HTTP，HTTPS，または LrHTTP を選択できます。 **注：**HTTP は UserNameOverTransport セキュリティと互換性はありません。HTTPS の場合は UserNameOverTransport セキュリティを選択する必要があります。
エンコード	サーバに送信されるサービス要求で使用するエンコーディング方式。
WS Addressing	選択した WSDL ファイルの WS-Addressing のバージョン。
セキュリティ	サーバで認証が必要な場合，認証モードとして［**UserNameOverTransport**］を選択し，有効なユーザ名とパスワードを入力します。

［WinSock］ノード

WinSocket の記録オプションを設定できます。

利用方法	［ツール］>［記録オプション］>［WinSock］
重要情報	Solaris マシンで変換テーブルを使用しているときは，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
関連タスク	1180 ページの「Windows Sockets スクリプトを記録する方法」

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
	除外されたソケットのリストに新しいエントリが追加されます。
	除外されたソケットのリストから選択したエントリが削除されます。
除外したソケットをログに含めない	リストのソケットがログから除外されます。このオプションをクリアすると，除外されたソケットがログに記録されるようになります。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。
除外する設定 / ソケット リスト	スクリプトの記録または再生成から除外するソケットのホストおよびポート。次の構文を使用します。 ➤ **host:port** の形式では特定のポートが除外されます。 ➤ **host** の形式では，指定したホストのすべてのポートが除外されます。 ➤ **:port** の形式では，ローカル・ホストの特定のポートが除外されます。 ➤ ***:port** の形式では，すべてのホストの特定のポートが除外されます。

UI 要素	説明
思考遅延時間のしきい値	記録中，アクションの合間に一時停止するときに VuGen によって思考遅延時間のステップ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。 標準設定値：5 秒。
変換テーブル	［変換テーブル］では，WinSock シングル・プロトコルを使用する場合の記録セッションの形式や，WinSock マルチ・プロトコルを使用する場合のコード生成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。 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。［**なし**］オプション（標準設定）を選択します。

11

実行環境の設定

本章の内容

概念

リファレンス

概念

実行環境の設定の概要

仮想ユーザ・スクリプトを記録した後，そのスクリプトの実行環境を設定できます。実行環境の設定は，スクリプトの実行方法を規定します。これらの設定は，仮想ユーザ・スクリプトのディレクトリにある **default.cfg** ファイルに格納されます。実行環境の設定は，VuGen，Controller，または Business Process Monitor を使ってスクリプトを実行するときに，仮想ユーザに適用されます。

実行環境の設定を行うことによって，さまざまな種類のユーザの動作をエミュレートできます。たとえば，サーバの出力にすぐに応答するユーザをエミュレートすることも，作業を停止して考えてから応答するユーザをエミュレートすることもできます。また実行環境の設定では，仮想ユーザがアクションを反復する回数も指定できます。

エラー処理

スクリプト実行時の仮想ユーザによるエラー処理の方法を指定することができます。標準設定では，仮想ユーザによってエラーが検出されると，スクリプトの実行が停止されます。次のいずれかの方法を使って，エラーが発生したときに次の反復を継続するように仮想ユーザを指定できます。

- 実行環境の設定を使用する：［**エラーでも処理を継続する**］実行環境の設定を指定できます。実行環境の設定で［**エラーでも処理を継続する**］を設定すると，仮想ユーザ・スクリプト全体に適用されます。スクリプトの一部分で，実行環境の設定の［**エラーでも処理を継続する**］をオーバーライドするには，**lr_continue_on_error** 関数を使用します。
- **lr_continue_on_error** 関数を使用する：**lr_continue_on_error** 関数を使用して，仮想ユーザ・スクリプトの特定のセグメントでのエラー処理を制御できます。対象セグメントを指定するには，セグメントを lr_continue_on_error(1) と lr_continue_on_error(0); ステートメントで囲みます。新しいエラー設定は，囲まれた部分の仮想ユーザ・スクリプト・セグメントに適用されます。詳細については，以降の説明を参照してください。

たとえば，実行環境の設定で［エラーでも処理を継続する］を有効にしていて，仮想ユーザが次に示すスクリプトのセグメントの再生中にエラーに遭遇した場合には，仮想ユーザはスクリプトの実行を継続します。

```
    web_link("EBOOKS",
        "Text=EBOOKS",
        "Snapshot=t2.inf",
        LAST);

    web_link("Find Rocket eBooks",
        "Text=Find Rocket eBooks",
        "Snapshot=t3.inf",
        LAST);
```

特定のセグメントを対象に，仮想ユーザにエラーでもスクリプトの実行を継続させるには，適切な lr_continue_on_error ステートメントで対象セグメントを囲みます。

```
lr_continue_on_error(1);
    web_link("EBOOKS",
        "Text=EBOOKS",
        "Snapshot=t2.inf",
        LAST);

    web_link("Find Rocket eBooks",
        "Text=Find Rocket eBooks",
        "Snapshot=t3.inf",
        LAST);
lr_continue_on_error(0);
```

コンテンツ・チェックの概要

VuGen の内容チェックのメカニズムを使って，ページ・コンテンツの特定の文字列を検査することができます。このオプションは，標準的ではないエラーを検出するのに役立ちます。通常の運用では，アプリケーション・サーバで障害が発生するとブラウザが汎用の HTTP エラー・ページを表示して，エラーの性質を通知します。この標準のエラー・ページは VuGen に認識され，エラーとして処理されるので，スクリプトは失敗として報告されます。しかし，アプリケーション・サーバの中には，VuGen がエラー・ページとして認識しない固有のエラー・ページを発行するものもあります。エラー・ページはサーバによって送信され，エラーが発生したことを示す特定の形式のテキスト文字列を含んでいます。

たとえば，エラーが発生したときに，「**ASP Error**」というテキストを含んでいるユーザ定義ページを発行するアプリケーションがあるとします。その場合には，VuGen に対して，サーバから返されたすべてのページでこのテキストを検索するように指定します。VuGen はこの文字列を検出すると，再生を失敗とします。なお，VuGen ではページ本体は検索されますが，ヘッダは検索されません。

CtLib サーバ・メッセージのログの記録

CtLib 仮想ユーザ・スクリプト（クライアント / サーバ・タイプのプロトコルの下にある Sybase CtLib）を実行する場合，CtLib クライアントによって生成されるメッセージはすべて，標準ログおよび出力ファイルのログに記録されます。標準設定では，サーバ・メッセージはログに記録されません。サーバ・メッセージのログの記録を（デバッグ用に）有効にするには，次の行を仮想ユーザ・スクリプトに挿入します。

```
LRD_CTLIB_DB_SERVER_MSG_LOG;
```

VuGen によって，すべてのサーバ・メッセージが標準ログに書き込まれます。

サーバ・メッセージを（標準ログ以外の）出力に送信するには，次のように入力します。

```
LRD_CTLIB_DB_SERVER_MSG_ERR;
```

サーバ・エラーを記録しない標準のモードに戻るには，次の行をスクリプトに入力します。

```
LRD_CTLIB_DB_SERVER_MSG_NONE;
```

注：生成されるサーバ・メッセージは長く，ログの書き込み処理によってシステムの処理速度が低下することがあるため，サーバ・メッセージのログの記録はスクリプト内の特定のコード・ブロックだけを対象に有効にします。

マルチスレッド

Controller は，ドライバ・プログラム（たとえば *mdrv.exe*，*r3vuser.exe* など）を使って仮想ユーザを実行します。各仮想ユーザを個別のプロセスとして実行した場合，同じドライバ・プログラムが仮想ユーザのインスタンスごとにメモリ上でいくつも実行（およびロード）されます。同じドライバ・プログラムをメモリにロードすると，多くの RAM とその他のシステム・リソースが消費されます。そのため，Load Generator で実行できる仮想ユーザの数が限定されることもあります。

別の方法として，各仮想ユーザをスレッドとして実行すると，Controller は 50 の仮想ユーザに対してドライバ・プログラム（*mdrv.exe* など）のインスタンスを 1 つだけ起動します（標準設定）。このドライバ・プロセス / プログラムは，いくつかの仮想ユーザを起動し，各仮想ユーザはスレッドとして実行されます。これらのスレッド化された仮想ユーザは，親ドライバ・プロセスのメモリのセグメントを共有します。ドライバ・プロセス / プログラムを何度も再ロードする必要がなくなるので，メモリ空間を大幅に節約でき，1 台の Load Generator で実行できる仮想ユーザの数を増やせます。

これらのオプションの設定については，450 ページの「[一般] > [その他] ノード」を参照してください。

リファレンス

プロトコルの互換性に関する表

次の表に，仮想ユーザ・スクリプトのタイプ，および各タイプで使用可能な実行環境の設定ノードを示します。

プロトコル	実行環境の設定ノード
AMF	➤ 一般 - 実行論理，ペースの設定，ログ，思考遅延時間，追加属性，その他 ➤ ネットワーク - 速度のシミュレーション ➤ ブラウザ - ブラウザのエミュレーション ➤ インターネット プロトコル - プロキシ，プリファレンス，ダウンロード フィルタ，コンテンツ チェック
AJAX	➤ 一般 - 実行論理，ペースの設定，ログ，思考遅延時間，追加属性，その他 ➤ ネットワーク - 速度のシミュレーション ➤ ブラウザ - ブラウザのエミュレーション ➤ インターネット プロトコル - プロキシ，プリファレンス，ダウンロード フィルタ，コンテンツ チェック
Ajax TruClient	➤ 一般 - ペースの設定，追加属性，ログ，その他の設定
C 仮想ユーザ	➤ 一般 - 実行論理，ペースの設定，ログ，思考遅延時間，追加属性，その他
Citrix	➤ 一般 - 実行論理，ペースの設定，ログ，思考遅延時間，追加属性，その他 ➤ ネットワーク - 速度のシミュレーション ➤ Citrix - 設定，同期
COM/DCOM	➤ 一般 - ペースの設定，ログ，思考遅延時間，追加属性，その他
DB2 CLI	➤ 一般 - ペースの設定，ログ，思考遅延時間，追加属性，その他

プロトコル	実行環境の設定ノード
DNS	➤ 一般 - ペースの設定，ログ，思考遅延時間，追加属性，その他
EJB	➤ 一般 - ペースの設定，ログ，思考遅延時間，追加属性，その他 ➤ Java 環境の設定 - Classpath，Java VM
FTP	➤ 一般 - ペースの設定，ログ，思考遅延時間，追加属性，その他 ➤ ネットワーク - 速度のシミュレーション
Flex	➤ 一般 - 実行論理，ペースの設定，ログ，思考遅延時間，追加属性，その他 ➤ ネットワーク - 速度のシミュレーション ➤ ブラウザ - ブラウザのエミュレーション ➤ Flex - RTMP，設定，外部オブジェクト ➤ インターネット プロトコル - プロキシ，プリファレンス，ダウンロード フィルタ，コンテンツ チェック
i モード	➤ 一般 - 実行論理，ペースの設定，ログ，思考遅延時間，追加属性，その他 ➤ ネットワーク - 速度のシミュレーション ➤ ブラウザ - ブラウザのエミュレーション ➤ インターネット プロトコル - プロキシ，プリファレンス，ダウンロード フィルタ
Informix	➤ 一般 - ペースの設定，ログ，思考遅延時間，追加属性，その他
IMAP	➤ 一般 - 実行論理，ペースの設定，ログ，思考遅延時間，追加属性，その他 ➤ ネットワーク - 速度のシミュレーション ➤ ブラウザ - ブラウザのエミュレーション ➤ インターネット プロトコル - プロキシ，プリファレンス，ダウンロード フィルタ，コンテンツ チェック

プロトコル	実行環境の設定ノード
Java over HTTP	➤ 一般 - ペースの設定，ログ，思考遅延時間，追加属性，その他 ➤ ネットワーク - 速度のシミュレーション ➤ Java 環境の設定 - Classpath，Java VM
Java Record Replay，Java 仮想ユーザ	➤ 一般 - ペースの設定，ログ，思考遅延時間，追加属性，その他 ➤ Java 環境の設定 - Classpath，Java VM
Javascript 仮想ユーザ	➤ 一般 - ペースの設定，ログ，思考遅延時間，追加属性，その他 ➤ ネットワーク - 速度のシミュレーション ➤ インターネット プロトコル - プロキシ，プリファレンス，ダウンロード フィルタ
LDAP	➤ 一般 - ペースの設定，ログ，思考遅延時間，追加属性，その他 ➤ ネットワーク - 速度のシミュレーション
MMS（Media Player）	➤ 一般 - ペースの設定，ログ，思考遅延時間，追加属性，その他 ➤ ネットワーク - 速度のシミュレーション
.NET	➤ 一般 - ペースの設定，ログ，思考遅延時間，追加属性，その他 ➤ .NET - .NET
RDP	➤ 一般 - ペースの設定，ログ，思考遅延時間，追加属性，その他 ➤ ネットワーク - 速度のシミュレーション ➤ RDP - 設定，同期，詳細設定，RDP エージェント
MAPI	➤ 一般 - ペースの設定，ログ，思考遅延時間，追加属性，その他
MS SQL Server	➤ 一般 - ペースの設定，ログ，思考遅延時間，追加属性，その他

プロトコル	実行環境の設定ノード
MMS（マルチメディア・メッセージング・サービス）	➤ 一般 - 実行論理，ペースの設定，ログ，思考遅延時間，追加属性，その他 ➤ ブラウザ - ブラウザのエミュレーション ➤ MMS - サーバとプロトコル ➤ WAP - Radius，ゲートウェイ ➤ インターネット プロトコル - プロキシ，プリファレンス，ダウンロード フィルタ
ODBC	➤ 一般 - ペースの設定，ログ，思考遅延時間，追加属性，その他
Oracle（2 層）	➤ 一般 - ペースの設定，ログ，思考遅延時間，追加属性，その他
Oracle NCA	➤ 一般 - 実行論理，ペースの設定，ログ，思考遅延時間，追加属性，その他 ➤ ネットワーク - 速度のシミュレーション ➤ Oracle NCA - クライアントのエミュレーション
Oracle Web Applications 11i	➤ 一般 - 実行論理，ペースの設定，ログ，思考遅延時間，追加属性，その他 ➤ ネットワーク - 速度のシミュレーション ➤ ブラウザ - ブラウザのエミュレーション ➤ インターネット プロトコル - プロキシ，プリファレンス，ダウンロード フィルタ，コンテンツ チェック ➤ Oracle NCA - クライアントのエミュレーション
PeopleSoft Enterprise	➤ 一般 - 実行論理，ペースの設定，ログ，思考遅延時間，追加属性，その他 ➤ ネットワーク - 速度のシミュレーション ➤ ブラウザ - ブラウザのエミュレーション ➤ インターネット プロトコル - プロキシ，プリファレンス，ダウンロード フィルタ，コンテンツ チェック
PeopleSoft Tuxedo	➤ 一般 - ペースの設定，ログ，思考遅延時間，追加属性，その他
POP3	➤ 一般 - ペースの設定，ログ，思考遅延時間，追加属性，その他 ➤ ネットワーク - 速度のシミュレーション

プロトコル	実行環境の設定ノード
Real	➤ 一般 - ペースの設定，ログ，思考遅延時間，追加属性，その他 ➤ ネットワーク - 速度のシミュレーション
SAP Web	➤ 一般 - 実行論理，ペースの設定，ログ，思考遅延時間，追加属性，その他 ➤ ネットワーク - 速度のシミュレーション ➤ ブラウザ - ブラウザのエミュレーション ➤ インターネット プロトコル - プロキシ，プリファレンス，ダウンロードフィルタ，コンテンツ チェック
SAP Click and Script	➤ 一般 - 実行論理，ペースの設定，ログ，思考遅延時間，追加属性，その他 ➤ ネットワーク - 速度のシミュレーション ➤ ブラウザ - ブラウザのエミュレーション ➤ インターネット プロトコル - プロキシ，プリファレンス，ダウンロードフィルタ，コンテンツ チェック
SAPGUI	➤ 一般 - 実行論理，ペースの設定，ログ，思考遅延時間，追加属性，その他 ➤ ネットワーク - 速度のシミュレーション ➤ SAPGUI - 一般
Siebel Web	➤ 一般 - 実行論理，ペースの設定，ログ，思考遅延時間，追加属性，その他 ➤ ネットワーク - 速度のシミュレーション ➤ ブラウザ - ブラウザのエミュレーション ➤ インターネット プロトコル - プロキシ，プリファレンス，ダウンロードフィルタ，コンテンツ チェック
Silverlight	➤ 一般 - ペースの設定，ログ，思考遅延時間，追加属性，その他 ➤ ネットワーク - 速度のシミュレーション ➤ ブラウザ - ブラウザのエミュレーション ➤ Silverlight - サービス ➤ インターネット プロトコル - プロキシ，プリファレンス，ダウンロードフィルタ，コンテンツ チェック

プロトコル	実行環境の設定ノード
SMTP	➤ 一般 - ペースの設定，ログ，思考遅延時間，追加属性，その他 ➤ ネットワーク - 速度のシミュレーション
Sybase CTlib / DBlib	➤ 一般 - ペースの設定，ログ，思考遅延時間，追加属性，その他
RTE	➤ 一般 - 実行論理，ペースの設定，ログ，思考遅延時間，追加属性，その他 ➤ RTE - RTE
Tuxedo，Tuxedo 6	➤ 一般 - ペースの設定，ログ，思考遅延時間，追加属性，その他
VB スクリプト仮想ユーザ	➤ 一般 - ペースの設定，ログ，思考遅延時間，追加属性，その他 ➤ ネットワーク - 速度のシミュレーション ➤ インターネット プロトコル - プロキシ，プリファレンス，ダウンロード フィルタ
VB 仮想ユーザ	➤ 一般 - ペースの設定，ログ，思考遅延時間，追加属性，その他 ➤ ネットワーク - 速度のシミュレーション ➤ インターネット プロトコル - プロキシ，プリファレンス，ダウンロード フィルタ ➤ VBA - VBA
WAP	➤ 一般 - 実行論理，ペースの設定，ログ，思考遅延時間，追加属性，その他 ➤ ネットワーク - 速度のシミュレーション ➤ WAP - Radius，ゲートウェイ ➤ インターネット プロトコル - プロキシ，プリファレンス，ダウンロード フィルタ，コンテンツ チェック
Web（Click and Script）	➤ 一般 - 実行論理，ペースの設定，ログ，思考遅延時間，追加属性，その他 ➤ ネットワーク - 速度のシミュレーション ➤ ブラウザ - ブラウザのエミュレーション ➤ インターネット プロトコル - プロキシ，プリファレンス，ダウンロード フィルタ，コンテンツ チェック

プロトコル	実行環境の設定ノード
Web（HTTP/HTML）	➤ 一般 - 実行論理，ページの設定，ログ，思考遅延時間，追加属性，その他 ➤ ネットワーク - 速度のシミュレーション ➤ ブラウザ - ブラウザのエミュレーション ➤ インターネット プロトコル - プロキシ，プリファレンス，ダウンロード フィルタ，コンテンツ チェック ➤ データ形式拡張機能 - 設定
Web サービス	➤ 一般 - 実行論理，ページの設定，ログ，思考遅延時間，追加属性，その他 ➤ ネットワーク - 速度のシミュレーション ➤ JMS - 詳細
Windows Sockets	➤ 一般 - ページの設定，ログ，思考遅延時間，追加属性，その他 ➤ ネットワーク - 速度のシミュレーション

実行環境の設定のユーザ・インタフェース

このセクションの内容

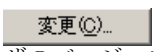
- ［一般］>［その他］ノード（450 ページ）
- ［一般］>［ページの設定］ノード（451 ページ）
- ［一般］>［実行論理］ノード（452 ページ）
- ［一般］>［思考遅延時間］ノード（453 ページ）
- ［インターネット プロトコル］>［コンテンツ チェック］ノード（454 ページ）
- ［インターネット プロトコル］>［ダウンロード フィルタ］ノード（456 ページ）
- ［インターネット プロトコル］の［プリファレンス］ノード（456 ページ）
- ［インターネット プロトコル］>［プロキシ］ノード（468 ページ）
- ［Java 環境の設定］>［Java Classpath］ノード（470 ページ）
- ［Java 環境の設定］>［Java VM］ノード（471 ページ）
- ［JMS］>［詳細］ノード（472 ページ）
- ［.Net］>［.NET 環境］ノード（474 ページ）
- ［MMS］>［サーバとプロトコル］ノード（475 ページ）
- ［ネットワーク］>［速度のシミュレーション］ノード（476 ページ）
- ［Oracle NCA］>［クライアントのエミュレーション］ノード（477 ページ）
- ［RDP］>［詳細］ノード（478 ページ）
- ［RDP］>［RDP エージェント］ノード（480 ページ）
- ［RDP］>［設定］ノード（481 ページ）
- ［RDP］>［同期化］ノード（482 ページ）
- ［RTE］>［RTE］ノード（483 ページ）
- ［SAPGUI］>［一般］ノード（485 ページ）
- ［Silverlight］>［サービス］ノード（486 ページ）
- ［VBA］>［VBA］ノード（487 ページ）
- ［WAP］>［ゲートウェイ］ノード（487 ページ）
- ［WAP］>［Radius］ノード（491 ページ）

［ブラウザ］>［ブラウザのエミュレーション］ノード

ブラウザに関連する実行環境を設定できます。

利用方法	**［仮想ユーザ］>［実行環境の設定］>［ブラウザ］>［ブラウザのエミュレーション］**
重要情報	このノードは特定のプロトコルにのみ適用されます。プロトコルの完全な一覧と各プロトコルに関連付けられたノードについては，424 ページの「プロトコルの互換性に関する表」を参照してください。

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
ユーザ エージェント	エミュレートされるブラウザに関する情報が表示されます。 インターネット仮想ユーザのすべてのヘッダには，エミュレートされているブラウザの種類（またはワイヤレスのツールキット）を示す「User Agent」ヘッダが含まれています。たとえば，**User-Agent:Mozilla/3.01Gold (WinNT; I)** は，ブラウザは Intel 社の CPU を搭載したマシン上の Windows NT で動作する Mozilla/3.01Gold がエミュレーション対象のブラウザであることを示しています。 User Agent ヘッダを変更するには，  をクリックします。ブラウザの種類，ブラウザのバージョン，言語，オペレーティング・システムを指定するか，カスタム User Agent ヘッダを入力します。
［変更］ボタン	［ユーザ エージェント］ダイアログ・ボックスが開き，エミュレートされるブラウザを変更できます。

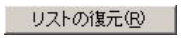
UI 要素	説明
ブラウザのキャッシュをシミュレートする	キャッシュを利用しながらブラウザをシミュレートするよう仮想ユーザに指示します（標準設定では有効）。このオプションが無効になっていて，リソースがページ上に複数回表示されていても，各リソースはページごとに一度だけダウンロードされます。リソースは画像，フレーム，または別の種類のスクリプト・ファイルです。このオプションを有効にすると，次のオプションを設定できます。 ➤ **[内容を要求する URL をキャッシュする (HTML)]**：HTML の内容を要求する URL のみをキャッシュするよう VuGen に指示します。内容は，解析，検証，または相関に必要となる場合があります。このオプションを選択すると，HTML 内容は自動的にキャッシュに保存されます。 **標準設定値**：有効。 **ヒント**：仮想ユーザのメモリ使用量を減らすには，テストの明示的な要件でなければ，このオプションを無効にします。 ➤ **[保管したページにアクセスする度に更新バージョンを確認する]**：指定した URL について，キャッシュに格納されているものより新しいバージョンがあるかどうかを確認するようブラウザに指示します。このオプションを有効にすると，VuGen によって「If-Modified-Since」属性が HTTP ヘッダに追加されます。このオプションを有効にすると，ページの最新版が表示されるようになりますが，シナリオまたはセッションの実行時に発生するトラフィックが増えます。 **標準設定値**：有効。 ➤ 詳細(A)... ：[URL の内容をキャッシュする - 詳細設定] ダイアログ・ボックスが開き，キャッシュに格納する URL コンテント・タイプを指定できます。

UI 要素	説明
HTML 以外のリソースをダウンロードする	仮想ユーザは，再生中に Web ページにアクセスするときに画像ファイルをロードします。これには，ページとともに記録された画像イメージと，明示的にはページとともに記録されていない画像イメージの両方が含まれます。実際のユーザは，Web ページにアクセスするとき，画像がロードされるのを待ちます。したがって，エンド・ユーザ時間を含め，システム全体をテストする場合には，このオプションを有効にします。パフォーマンスを向上させるために，実際のユーザをエミュレートしない場合は，このオプションを無効にします。 Web ページにアクセスするたびに画像が変わり（広告業者のバナーなど），画像チェックで不一致が生じる場合には，このオプションを無効にします。
反復ごとに新規ユーザをシミュレートする	このオプションを選択すると，VuGen は，反復を行う前にすべての HTTP コンテキストを **init** セクションの終了時の状態にリセットします。この設定により，仮想ユーザは新規ユーザがブラウザ・セッションを開始する様子をより正確にエミュレートできます。クッキーをすべて削除するには，すべての TCP 接続を閉じ（キープ・アライブを含む），エミュレートされたブラウザのキャッシュをクリアします。次に HTML フレーム階層をリセットして（フレームは番号 1 から番号付けされる）ユーザ名およびパスワードをクリアします。 **標準設定値**：有効。 ➤ **[反復ごとにキャッシュをクリアする]**：Web ページに初めてアクセスしたユーザをシミュレートするために，反復ごとにブラウザのキャッシュをクリアします。仮想ユーザがブラウザのキャッシュに格納された情報を使用して，最近ページにアクセスしたユーザをシミュレートできるようにする場合は，チェック・ボックスをクリアしてこのオプションを無効にします。

[URL の内容をキャッシュする - 詳細設定] ダイアログ・ボックス

[詳細設定] ダイアログ・ボックスでは，キャッシュに格納する URL 内容タイプを指定できます。

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
	標準設定のコンテント・タイプが削除された場合は復元されます。
	**[追加]**：新しいコンテント・タイプが追加されます。
	**[削除]**：コンテント・タイプが削除されます。
< コンテント タイプ リスト >	各タイプを有効または無効にするチェックボックスと一緒に表示されるコンテント・タイプのリスト。
HTML ページ以外に URL が要求する内容を指定する	URL を指定できます。

［Citrix］の >［設定］ノード

Citrix 設定の実行環境を設定できます。

利用方法	［仮想ユーザ］>［実行環境の設定］>［Citrix］>［設定］
重要情報	このノードは特定のプロトコルにのみ適用されます。プロトコルの完全な一覧と各プロトコルに関連付けられたノードについては，424 ページの「プロトコルの互換性に関する表」を参照してください。

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
マウスの移動とキーストロークをキューに挿入する	マウスの動きとキーストロークのキューを作成し，これらをパケットとして低い頻度でサーバに送信するよう仮想ユーザに指示します。これにより，低速接続の場合のネットワーク・トラフィックが減少します。このオプションを有効にすると，セッションにおけるキーボードとマウスの動きに対する応答が鈍くなります。 **標準設定値**：有効。
サウンド クウォリティ	サウンドのクウォリティを指定します。クライアント・マシンに 16 ビットの Sound Blaster 互換サウンド・カードが搭載されていない場合は，［**サウンド オフ**］を選択します。サウンドのサポートを有効にすると，クライアント・マシンの発行されたアプリケーションからサウンド・ファイルを再生できるようになります。
SpeedScreen 待ち時間の減少	ネットワークの速度が遅いときにユーザとのやり取りを向上させるために使用する機能。ネットワークの速度に応じてこの機能を「**オン**」または「**オフ**」にします。「**自動**」オプションを選択すると，現在のネットワーク速度に基づいてオプションのオン / オフが切り替わります。ネットワーク速度がわからない場合は，このオプションを［**サーバ標準設定値の使用**］に設定し，マシンの標準設定を使用するようにします。

UI 要素	説明
データ圧縮を使用する	転送するデータを圧縮するよう仮想ユーザに指示します。帯域幅が限られている場合は，データ圧縮を有効にします。 **標準設定値**：有効。
ビットマップにディスク キャッシュを使用する	ビットマップや，よく使用するグラフィカル・オブジェクトをローカルのキャッシュに格納するよう仮想ユーザに指示します。帯域幅が限られている場合は，このオプションを有効にします。 **標準設定値**：有効。

［Citrix］>［同期］ノード

Citrix 同期の実行環境を設定できます。

利用方法	［仮想ユーザ］>［実行環境の設定］>［Citrix］>［同期］
重要情報	このノードは特定のプロトコルにのみ適用されます。プロトコルの完全な一覧と各プロトコルに関連付けられたノードについては，424 ページの「プロトコルの互換性に関する表」を参照してください。

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
標準設定の画像同期許容範囲	この設定により，同期化されているとみなされるために 2 つの画像が共有しなければならない等しさのレベルが制御されます。 **［完全一致］**：100% の一致が必要。 **［低い］**：95 % の一致が必要。 **［中］**：85% の一致が必要。 **［高い］**：70% の一致が必要。 **標準設定値**：完全一致。

UI 要素	説明
タイムアウト	➤ **［接続時間］**：確立されている接続を終了する前に，アイドル状態で待機する秒数。 ➤ **［待ち時間］**：接続を終了する前に，同期ポイントでアイドル状態のまま待機する秒数。
入力の割合	キーストローク間の遅延時間（ミリ秒）。

［Flex］>［設定］ノード（実行環境の設定）

外部 JVM（Java 仮想マシン）のパスを設定できます。

利用方法	［仮想ユーザ］>［実行環境の設定］>［Flex］>［設定］
重要情報	このノードは特定のプロトコルにのみ適用されます。プロトコルの完全な一覧と各プロトコルに関連付けられたノードについては，424 ページの「プロトコルの互換性に関する表」を参照してください。

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
Use External JVM	外部 JVM を使用できます。
External JVM Path	外部 JVM のパスを指定します。

［Flex］>［外部オブジェクト］ノード（実行環境の設定）

このダイアログ・ボックスでは，Flex スクリプトの外部オブジェクトの実行環境を設定できます。

利用方法	**［仮想ユーザ］>［実行環境の設定］>［Flex］>［外部オブジェクト］**
関連タスク	679 ページの「外部 Java シリアライザを使用してシリアル化する方法」 680 ページの「LoadRunner シリアライザを使用してスクリプトをシリアル化する方法」
関連項目	674 ページの「Flex の概要」 675 ページの「Flex スクリプトの外部オブジェクト」

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
［Classpath エントリ］リスト	クラスパスのエントリのリスト。
	**下向き矢印**：クラスパスのエントリをリストの下方向に移動します。
	**上向き矢印**：クラスパスのエントリをリストの上方向に移動します。
	**［クラスパスの追加］**：クラスパスのリストに新しい行を追加します。
	**［削除］**：クラスパスを永久に削除します。

［Flex］>［RTMP］ノード

Flex RTMP の実行環境を設定できます。

利用方法	**［仮想ユーザ］>［実行環境の設定］>［Flex］>［RTMP］**
重要情報	このノードは特定のプロトコルにのみ適用されます。プロトコルの完全な一覧と各プロトコルに関連付けられたノードについては，424 ページの「プロトコルの互換性に関する表」を参照してください。

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
繰り返しを終了するたびに開いていた RTMP 接続はすべて閉じてください	各反復の終了時に開かれている RTMP 接続が自動的に切断されます。
TCP receive timeout (milliseconds)	クライアントが TCP/IP パケットの受信を開始するまで LoadRunner が待機する時間間隔をミリ秒単位で指定します。タイムアウトに達した場合，LoadRunner は［**Status for TCP receive timeout**］の実行環境設定に応じて警告またはエラーを発行します。
Status for TCP receive timeout	タイムアウトを超えた場合にステップで発行するステータスを示します。
警告	タイムアウトを超えた場合にステップで警告を発行します。LoadRunner は次のステップまで続行します。
エラー	タイムアウトを超えた場合にステップでエラーを発行します。スクリプトは終了します。

［一般］>［追加属性］ノード

［追加属性］ノードを使用して仮想ユーザ・スクリプトの追加属性を指定できます。［追加属性］の設定は，すべてのタイプの仮想ユーザ・スクリプトに適用されます。

lr_get_attrib_string を使用すると，テスト実行中のある時点で取得できるコマンド・ライン引数を指定できます。このノードを使用して，作成されたスクリプトに外部パラメータを渡すことができます。

利用方法	**［仮想ユーザ］>［実行環境の設定］>［一般］>［追加属性］**
重要情報	このノードは特定のプロトコルにのみ適用されます。プロトコルの完全な一覧と各プロトコルに関連付けられたノードについては，424 ページの「プロトコルの互換性に関する表」を参照してください。

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
< 追加属性テーブル >	追加属性とその値のリスト。
	新しい引数が追加されます。
	引数が削除されます。

［一般］>［ログ］ノード

ログに記録する情報の量と種類を設定できます。

利用方法	［仮想ユーザ］>［実行環境の設定］>［一般］>［ログ］
重要情報	➤ このノードは特定のプロトコルにのみ適用されます。プロトコルの完全な一覧と各プロトコルに関連付けられたノードについては，424 ページの「プロトコルの互換性に関する表」を参照してください。 ➤ **lr_error_message** 関数と **lr_output_message** 関数を使用して，出力ログにメッセージを送信するように仮想ユーザ・スクリプトを手作業でプログラミングすることもできます。 ➤ Web プロトコルの場合は，**default.cfg** ファイルを編集することによってより詳細な情報を得ることができます。［**Web**］セクションで，**LogFileWrite** フラグを **1** に設定します。生成されるトレース・ファイルには，スクリプト実行時のすべてのイベントが記録されます。

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
ログを有効にする	再生中の自動ログ記録が有効化されます。このオプションを無効にすると，自動ログ記録と **lr_log_message** を通じて発行されるログ・メッセージにのみ影響します。**lr_message**，**lr_output_message**，**lr_error_message** を使って手作業で送信されるメッセージは，変わらずに発行されます。
ログ オプション	メッセージがいつログに送信されるかを示します。 ➤ **［エラー発生時のみメッセージを送信する］**：エラーが発生したときにだけメッセージがログに送信されます。ログのキャッシュ・サイズを設定するには，［**詳細**］をクリックします。キャッシュの内容が指定したサイズを超えると，一番古い項目が削除されます。 ➤ **［常にメッセージを送信する］**：すべてのメッセージがログに送信されます。

UI 要素	説明
標準ログ	これらの標準ログはデバッグで使用します。システム・リソースを節約する必要がある場合，大規模な負荷テストのシナリオまたはプロファイルではこのオプションを無効にできます。
拡張ログ	警告やメッセージを含めた詳細なログが作成されます。システム・リソースを節約する必要がある場合，大規模な負荷テストのシナリオまたはプロファイルではこのオプションを無効にできます。 次のオプションも指定できます。 ➤ **[パラメータ置換]**：スクリプトに割り当てられたすべてのパラメータがそれらの値とともにログに記録されます。 ➤ **[サーバが返したデータ]**：サーバによって返されたすべてのデータがログに記録されます。 ➤ **[詳細トレース]**：セッション中に仮想ユーザが送信したすべての関数とメッセージがログに記録されます。

［一般］>［ログ］ノード（TruClient）

TruClient スクリプトのログにレポートする情報の量とタイプを設定できます。

利用方法	**［仮想ユーザ］>［実行環境の設定］>［一般］>［ログ］**

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
ログのレベルと関連オプション	エラーのみ ➤ エラー・メッセージのみをログに記録する場合に選択します。 標準ログ ➤ エラー/警告のみをログに記録 (標準ログ内) エラーと警告のみをログに記録する場合に選択します。情報メッセージは含まれません。 **注：**拡張ログでは，この設定に関係なく，このオプションによって無視されたすべてのメッセージが含まれます。 拡張ログ 低レベルの情報メッセージおよび標準ログに含まれるすべてのメッセージをログに記録する場合に選択します。 **HTTP 関連のメッセージ** ➤ **[HTTP 要求ヘッダをログに記録]**：各要求の HTTP 要求ヘッダがログに記録されます。 ➤ **[HTTP 要求本体 (POST データ) をログに記録]**：各要求の HTTP 要求本体がログに記録されます。 ➤ **[HTTP 応答ヘッダをログに記録]**：各要求の HTTP 応答ヘッダがログに記録されます。 ➤ **[HTTP 応答本体をログに記録]**：各要求の HTTP 応答本体がログに記録されます。 ➤ **[HTTP トレースをログに記録]**：HTTP 要求 / 応答処理がログに記録されます。 **テスト対象アプリケーション (AUT) のメッセージ** ➤ **[AUT エラー メッセージをログに記録]**：テスト対象アプリケーションから受信したエラー・メッセージがログに記録されます。 ➤ **[AUT 非エラー メッセージをログに記録]**：テスト対象アプリケーションから受信した情報メッセージがログに記録されます。 **パラメータ化** ➤ **[パラメータ置換ログに記録]**：仮想ユーザ・スクリプトの実行時に使用されたパラメータとその値がログに記録されます。

UI 要素	説明
必要な場合にのみログ ファイルを作成する	[メモリにメッセージを累積し，必要な場合にのみ書き込む]： ➤ エラー発生時のみメッセージをログに記録する場合に選択します。 ➤ **[[メモリのメッセージを累積] のバッファ サイズ]**：重要なメッセージのログが記録される先のメモリ・バッファ・サイズを入力します。 **注**：サイズが小さすぎるとメッセージが失われます。サイズが大きすぎると，ページングの問題が発生したり，実行時間が遅くなったりする場合があります。
ログ メッセージの フォーマット オプション	[ユーザ ログ行の長さ]： ➤ ユーザ・ログ・ファイルでメッセージ行が折り返されるまでの 1 行の長さの制限を入力します。 [ユーザ ログにタイムスタンプを含める]： ➤ 各メッセージの時間トレースをログに含める場合に選択します。
内部のサポート オプション	これらのオプションは，カスタマ・サポート担当者のみが使用します。カスタマ・サポートから特に指示されないかぎり，変更しないでください。

［一般］>［Load Mode Browser Settings］ノード

このノードでは，ロード・モードで実行するスクリプトの TruClient ブラウザを設定できます。

利用方法	**［実行環境の設定］>［一般］>［Load Mode Browser Settings］**
重要情報	➤［Load Mode Browser Settings］ノードで変更した設定は，ロード・モードにのみ影響します。 ➤［Load Mode Browser Settings］ノードで使用できる設定は，［**Ajax TruClient 一般設定**］>［**ブラウザ設定**］タブで使用できる設定と同じです。 ［ブラウザ設定］タブで変更した設定は，対話式モードにのみ影響します。対話式モードでスクリプトを保存すると，［ブラウザ設定］タブで変更した設定が［Load Mode Browser Settings］に適用されます。

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
プロキシ選択	
［プロキシなし (インターネットへ直接接続)］	（標準設定）すべての仮想ユーザは，プロキシ・サーバを使用せずにインターネットに直接接続します。
［手動プロキシ設定］	プロキシ・サーバ・ルールの例外を指定できます。また，すべての HTTP/HTTPS 接続に対応するプロキシ・サーバを指定できます。HTTP 接続専用のプロキシ・サーバが必要な場合は，［**Use separate proxy for HTTPS protocol**］ボックスを選択し，プロキシ・サーバ設定を指定します。
自動プロキシ設定 (PAC)	プロキシ自動設定ファイル（PAC）から自動的にプロキシ設定を読み込みます。［**URL**］フィールドで，PAC ファイルの URL を入力します。
詳細	

UI 要素	説明
[ホームページの URL]	JavaScript で window home が呼び出されたときに移動する URL を入力します（標準設定は about:blank）
[ユーザ エージェント]	要求ヘッダのブラウザの標準設定を上書きするユーザ エージェント文字列を入力します。
キャッシュにあるページとネットワーク上のページが比較されます。	キャッシュにある URL ページとネットワークから呼び出された URL ページを比較するために使用する対応する番号を入力します。 ➤ **0。**（標準設定）セッションごとに 1 回。 ➤ **1。**ページ・アクセスごと。 ➤ **2。**なし。 ➤ **3。**ページの期限が切れている場合（標準設定）。
[Keep-Alive]	持続的（プロキシを使用しない）ネットワーク接続を許可します。 ➤ **選択。**（標準設定）持続的（プロキシを使用しない）ネットワーク接続を許可します。そのため，開いている接続が再利用されることがあります。 ➤ **クリア。**要求の完了後，各接続を閉じます。
[プロキシ Keep-Alive]	持続的プロキシ接続を許可します。 ➤ **選択。**（標準設定）持続的プロキシ接続を許可します。そのため，開いている接続が再利用されることがあります。 ➤ **クリア。**要求の完了後，各接続を閉じます。
[Keep-Alive タイムアウト (秒)]	アイドル中の接続を開いたままにする秒数を入力します（標準設定は 300）。
HTTP のバージョン	プロキシを経由しないでネットワーク / アプリケーションにアクセスしたときに使用する HTTP バージョンを入力します。（標準設定は 1.1）。
[プロキシ HTTP バージョン]	プロキシを経由してネットワーク / アプリケーションにアクセスしたときに使用するプロキシ HTTP バージョンを入力します。（標準設定は 1.1）。
[DNS キャッシュ エントリ]	DNS キャッシュ内に保存するエントリの最大数を入力します。このオプションを無効にする場合は「0」と入力します。（標準設定は 20）。

UI 要素	説明
SSL	セキュア接続設定を選択します。 ➤ SSL 2.0 ➤（標準設定）SSL 3.0 ➤（標準設定）TLS 1.0

［一般］>［その他の設定］ノード

TruClient スクリプトのログに記録する情報の量と種類を設定できます。

利用方法	**［仮想ユーザ］>［実行環境の設定］>［一般］>［その他の設定］**

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
反復ごとに新規ユーザをシミュレートする	反復間のすべての HTTP コンテキストをリセットするよう VuGen に指示します。この設定により，仮想ユーザは新規ユーザがブラウザ・セッションを開始する様子をより正確にエミュレートできます。 **標準設定値**：有効。

UI 要素	説明
スナップショットの生成	Recording snapshots generation ➤ **[Never]**：選択すると，スクリプトの記録時のスナップショットが自動的に保存されなくなります。 ➤ **[Always]**：（標準設定）選択すると，スクリプトのすべてのステップで記録時のスナップショットが自動的に保存されるようになります。 再生時のスナップショットの生成 ➤ **[Never]**：選択すると，再生時のスナップショットが生成されなくなります。 **注**：これは標準設定です。 ➤ **[On error]**：エラー発生時にスナップショットを生成する場合に選択します。このオプションは，スクリプトのバグを特定するのに役立ちます。詳細については，517 ページの「スナップショットを使用したスクリプトのデバッグ」を参照してください。 ➤ **[Always]**：選択すると，スクリプトのすべてのステップでスナップショットが生成されるようになります。
エラー発生時のアクション	スクリプトを中止する ➤ エラー発生時にスクリプトの実行を中止する場合に選択します。 次の反復を続行する ➤ エラー発生時に反復を中止して，次の反復を続行する場合に選択します。
再生オプション	検出されないオブジェクトの最大時間（秒数） ➤ アプリケーションでエラー・メッセージが表示されるまでにオブジェクトを検索する最大時間を入力します。（標準設定は 20）。 ステップ間の間隔（ミリ秒） ➤ ステップ間の最小間隔を入力します。値を大きくすると，同期の問題が解決される場合がありますが，大きすぎるとスクリプトの実行が必要以上に遅くなる可能性があります。（標準設定は 500）。 ネットワークの終了識別タイムアウト（ミリ秒） ➤ ステップのネットワーク終了は，ネットワーキングの動作がなく指定した時間が経過すると認識されます。（標準設定は 150）。

UI 要素	説明
非反復ウィンドウ サイズ：	対話式モード以外のブラウザ・ウィンドウの初期サイズ（ピクセル単位の幅と高さ）（標準設定は 1024 x 1280）

［一般］>［その他］ノード

その他の実行環境を設定できます。

利用方法	**［仮想ユーザ］>［実行環境の設定］>［一般］>［その他］**
重要情報	➤ このノードは特定のプロトコルにのみ適用されます。プロトコルの完全な一覧と各プロトコルに関連付けられたノードについては，424 ページの「プロトコルの互換性に関する表」を参照してください。 ➤ 負荷テスト環境で［**エラーでも処理を継続する**］オプションと［**エラー時にスナップショットを生成する**］オプションを両方とも有効にすることはお勧めしません。このように設定すると，仮想ユーザのパフォーマンスに悪影響を与える可能性があります。 ➤ Sybase-CtLib，Sybase-DbLib，Informix，Tuxedo，PeopleSoft-Tuxedo の各プロトコルはスレッドとして実行しないでください。 ➤ シナリオの実行中に診断ブレークダウン・データ（J2EE）を生成する必要がある場合は，自動トランザクションを使用してはなりません。各トランザクションの開始と終了を手作業で定義してください。

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
自動トランザクション	➤ **[各アクションをトランザクションとして定義する]**：スクリプト内の各アクションをトランザクションとして処理するよう LoadRunner に指示します（HP Business Availability Center には適用されません）。 ➤ **[各ステップをトランザクションとして定義する]**：スクリプト内の各ステップをトランザクションとして処理するよう LoadRunner に指示します（HP Business Availability Center には適用されません）。
エラー処理	➤ **[エラーでも処理を継続する]**：エラーが発生してもスクリプトの実行が継続されます。 **標準設定値**：有効。 ➤ **[lr_error_message 通知時に処理中のトランザクションを失敗にする]**：**lr_error_message** 関数が発行されたすべてのトランザクションを「*失敗*」としてマークするよう VuGen に指示します。**lr_error_message** 関数は，手作業で定義された *If* ステートメントを通して発行されます。 ➤ **[エラー時にスナップショットを生成する]**：エラーが発生したらスナップショットが生成されます。スナップショットは，仮想ユーザ・ログでエラーが発生した行をダブルクリックすると表示できます。
マルチスレッド	➤ **[仮想ユーザをプロセスとして実行]**：マルチスレッド機能が無効になります。 ➤ **[仮想ユーザをスレッドとして実行]**：マルチスレッド機能が有効になります。

［一般］>［ペースの設定］ノード

反復の間隔を制御できます。ペースは，アクションの反復の間で待機する時間を仮想ユーザに指示します。

利用方法	**［仮想ユーザ］>［実行環境の設定］>［一般］>［ペースの設定］**
重要情報	このノードは特定のプロトコルにのみ適用されます。プロトコルの完全な一覧と各プロトコルに関連付けられたノードについては，424 ページの「プロトコルの互換性に関する表」を参照してください。

ユーザ・インタフェース要素の説明は次のとおりです。

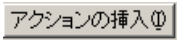
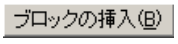
UI 要素	説明
前回の反復が終了後	前回の反復が終了後，指定した時間で新しい反復が開始されます。正確な秒数または時間の範囲を指定します。
前回の反復が終了次第すぐ	直前の反復の終了後，すぐに次の反復が開始されます。
< 一定 / 値の範囲 > 間隔	反復と反復の間の時間を指定します。秒単位で固定時間または時間の範囲を定義します。スケジュールが設定された各反復は，前回の反復が完了した後に開始されます。

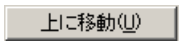
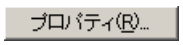
［一般］>［実行論理］ノード

実行論理の実行環境を設定できます。

利用方法	**［仮想ユーザ］>［実行環境の設定］>［一般］> ［実行論理］**
重要情報	このノードは特定のプロトコルにのみ適用されます。プロトコルの完全な一覧と各プロトコルに関連付けられたノードについては，424 ページの「プロトコルの互換性に関する表」を参照してください。
関連項目	40 ページの「スクリプトのセクション」

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
	挿入ポイントに新しいアクションが挿入されます。
	挿入ポイントに新しいアクション・ブロックが挿入されます。
	項目が削除されます。

UI 要素	説明
	項目が上に移動します。
	項目が下に移動します。
	［プロパティ］ダイアログ・ボックスが開き，実行論理と反復を設定できます。 ［**実行論理**］：アクションを順番またはランダムに実行されるように設定します。 ［**反復**］：項目が実行される回数を設定します。
＜実行論理ツリー＞	このスクリプトの実行論理を視覚的に示す図。
反復回数	スクリプトによって実行論理ツリー内の項目が実行される回数。

［一般］＞［思考遅延時間］ノード

思考遅延時間を設定し，アクション間で待機する時間を制御できます。これらの設定は，現実のユーザをエミュレートするために設計されています。

利用方法	**［仮想ユーザ］＞［実行環境の設定］＞［一般］＞［思考遅延時間］**
重要情報	このノードは特定のプロトコルにのみ適用されます。プロトコルの完全な一覧と各プロトコルに関連付けられたノードについては，424 ページの「プロトコルの互換性に関する表」を参照してください。
関連項目	➤ VuGen では **lr_think_time** 関数を使って，仮想ユーザ・スクリプトに思考遅延時間の値が記録されます。lr_think_time 関数の詳細と手作業での変更方法については，*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。

ユーザ・インタフェース要素の説明は次のとおりです。

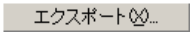
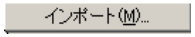
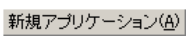
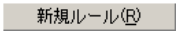
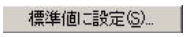
UI 要素	説明
思考遅延時間を記録通りに再生する	再生時に，lr_think_time 関数の引数の値が使用されます。たとえば，lr_think_time(10) は 10 秒間待機します。
思考遅延時間を無視する	記録された思考遅延時間を無視します。つまり，すべての **lr_think_time** 関数を無視してスクリプトを再生します。
思考遅延時間を X 秒に制限する	思考遅延時間の最大値を制限します。
記録された思考遅延時間に掛ける倍数	再生時に，記録された思考遅延時間の倍数を使用します。これにより再生中に適用される思考遅延時間を増減させることができます。たとえば，4 秒間の思考遅延時間が記録されている場合に，その値を 2 倍するように指定すれば，仮想ユーザの思考遅延時間は 8 秒になります。思考遅延時間を 2 秒に減らすには，記録された時間に 0.5 を掛けます。
思考遅延時間を再生する	記録された思考遅延時間をカスタマイズできるオプションが有効になります。
記録された思考遅延時間の乱数率を使用	［記録された思考遅延時間の乱数率を使用する］：思考遅延時間の範囲を指定することによって，思考遅延時間の値の範囲を設定します。たとえば，思考遅延時間の引数が 4 の場合，下限を 50%，上限を 150% に設定すると，最短の思考遅延時間は 2（50%），最長の思考遅延時間は 6（150%）となります。

［インターネット プロトコル］>［コンテンツ チェック］ノード

実行時に Web サイトのコンテンツをチェックできます。

利用方法	**［仮想ユーザ］>［実行環境の設定］>［インターネット プロトコル］>［コンテンツ チェック］**
重要情報	このノードは特定のプロトコルにのみ適用されます。プロトコルの完全な一覧と各プロトコルに関連付けられたノードについては，424 ページの「プロトコルの互換性に関する表」を参照してください。

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
	選択したルールまたはアプリケーションが削除されます。
	ルールが xml ファイルにエクスポートされます。
	ルールが既存の xml ファイルからインポートされます。
	新しいアプリケーションがアプリケーションとルールのリストに追加されます。名前を変更するにはクリックします。
	右側の表示枠にルール条件が表示され，現在選択されているアプリケーションの新しいルールを入力できるようになります。 ➤ **[検索するテキスト]**：検索するテキスト文字列。 ➤ **[検索対象プレフィックスとサフィックス]**：検索する文字列の前置記号と後置記号。 ➤ **[大文字と小文字を区別する]**：大文字と小文字を区別して検索します。 ➤ **[失敗条件]**：文字列が検出された場合，または検出されなかった場合に，検索の結果が失敗するように設定します。 ➤ **[JavaScript 警告ボックス テキストを検索する]**：JavaScript 警告ボックス内のテキストのみを検索します（Web（Click and Script），PeopleSoft Enterprise，および Oracle Web Applications 11i 仮想ユーザのみ）。
	現在のコンテンツ チェックの構成が標準として設定されます（新しいスクリプトはこの構成で開始されます）。
< アプリケーションおよびルール・リスト >	アプリケーションとルールのリスト。各項目のチェック・ボックスを使用して，個々の項目を有効または無効にできます。
再生中にコンテンツ チェックを有効にする	再生中のコンテンツ チェックが有効になります。なお，アプリケーションに対するルールを定義した後でも，このオプションを無効にすることで，テスト実行ごとにルールを無効にできます。 **標準設定値**：有効。

［インターネット プロトコル］>［ダウンロード フィルタ］ノード

ダウンロード・フィルタを設定できます。

利用方法	**［仮想ユーザ］>［実行環境の設定］>［インターネット プロトコル］>［ダウンロード フィルタ］**
重要情報	このノードは特定のプロトコルにのみ適用されます。プロトコルの完全な一覧と各プロトコルに関連付けられたノードについては，424 ページの「プロトコルの互換性に関する表」を参照してください。

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
< フィルタ・リスト >	スクリプトのフィルタのリスト。それぞれのフィルタにはタイプとデータがあります。たとえば，タイプ URL のフィルタには，URL アドレスとデータがあります。リストのエントリは，［**追加**］，［**編集**］，［**削除**］，または［**すべて削除**］できます。
リストされたアドレスを除外する	リストされたサイトまたはホストからの要求が無視されます。
リストされたアドレスのみを含める	再生はリストされたサイトまたはホストに限定されます。

［インターネット プロトコル］の［プリファレンス］ノード

さまざまなインターネット関連の実行環境を設定できます。

利用方法	**［仮想ユーザ］>［実行環境の設定］>［インターネット プロトコル］>［プリファレンス］**
重要情報	このノードは特定のプロトコルにのみ適用されます。プロトコルの完全な一覧と各プロトコルに関連付けられたノードについては，424 ページの「プロトコルの互換性に関する表」を参照してください。

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
チェック	➤ **[画像とテキスト チェックを有効にする]**：仮想ユーザは再生中に **web_find** または **web_image_check** 検証関数を実行して，検証チェックを行うことができます。このオプションは，HTTP モードで記録されたステートメントにのみ適用されます。検証チェックを実行する仮想ユーザは，検証チェックを実行しない仮想ユーザより多くのメモリを消費します。 **標準設定値**：有効。
Web パフォーマンス グラフを作成	このオプションを選択すると，仮想ユーザによって Web パフォーマンス・グラフの作成に使用するデータが収集されます。テストの実行中はオンライン・モニタ，テストの実行後は Analysis を使用して，[**秒ごとのヒット数および HTTP コード**]，[**秒ごとのヒット数（HTML モードのみ）**]，[**秒ごとの応答バイト数**]（スループット）のグラフを表示できます。Analysis を使用して，テスト実行後にコンポーネント・ブレークダウン・グラフを表示できます。仮想ユーザに収集させるグラフ・データのタイプを選択します。 **注**：Web パフォーマンス・グラフを使用しない場合は，メモリを節約するためにこれらのオプションを無効にしておきます。

UI 要素	説明
詳細	➤ **[WinInet 再生]**：このオプションを選択すると，VuGen は標準のソケット再生の代わりに WinInet 再生エンジンを使用します。VuGen には，Sockets ベース（標準設定）および WinInet ベースの 2 つの HTTP 再生エンジンがあります。WinInet は Internet Explorer で使用されるエンジンであり，IE ブラウザに組み込まれている機能をすべてサポートしています。WinInet 再生エンジンの制約として，スケーラブルでないこと，UNIX をサポートしていないことが挙げられます。さらに，スレッドでの作業時に WinInet エンジンがモデム速度および接続数を正確にエミュレートしないことがあります。VuGen 独自のソケット・ベースの再生は，負荷テストにおけるスケーラビリティに優れた小型軽量のエンジンです。また，スレッドでの使用時にも正確に機能します。ソケット・ベースのエンジンの制限事項としては，SOCKS プロキシがサポートされていない点があります。このタイプの環境を記録する場合は，WinInet 再生エンジンを使用してください。 **標準設定値**：無効（ソケットベースの再生エンジン）。 ➤ **[自動トランザクション名にファイルと行を追加する]**：トランザクション名にファイル名と行番号を追加し，自動トランザクション時に一意となるトランザクション名が作成されます。 **標準設定値**：有効。 ➤ **[クリティカルではないリソースのエラーを警告する]**：ダウンロードに失敗した画像や Java アプレットなど，負荷テストにとってさほど重要ではない項目で，失敗した関数についての警告ステータスを返します。標準設定では，このオプションは有効になっています。特定の警告をエラーとみなしてテストを失敗させる場合は，このオプションを無効にできます。特定の内容タイプを重要なものとして設定するには，非リソースのリストにその内容タイプを加えます。詳細については，372 ページの「[リソース以外の項目］ダイアログ・ボックス」を参照してください。 ➤ **[スナップショットのリソースをローカルに保存する]**：VuGen はスナップショット・リソースをローカル・マシン上にファイルとして保存します。この機能を使用することで，実行時ビューアはスナップショットをより正確に作成し，よりすばやく表示できます。 ➤ オプション(O)...：[詳細オプション］ダイアログ・ボックスを開きます

［詳細オプション］ダイアログ・ボックス

詳細なインターネット・プリファレンスの実行環境を設定できます。

利用方法	［仮想ユーザ］>［実行環境の設定］>［インターネット プロトコル］>［プリファレンス］>［オプション］
重要情報	➤ このダイアログ・ボックスは特定のプロトコルにのみ適用されます。プロトコルの完全な一覧と各プロトコルに関連付けられたノードについては，424 ページの「プロトコルの互換性に関する表」を参照してください。 ➤ このダイアログ・ボックスでは，［**HTTP**］，［**一般**］，［**認証**］，［**ログ**］，［**Web (Click and Script) 固有**］というカテゴリにプロパティが分けられています。

HTTP

UI 要素	説明
HTTP のバージョン	HTML バージョンとしてバージョン 1.0 または 1.1 のどちらを使用するかを指定します。バージョン情報は，仮想ユーザが Web サーバに要求を送信するときの HTTP 要求ヘッダに含まれています。 **標準設定値**：HTTP 1.1。 HTTP 1.1 では次の機能がサポートされています。 ➤ 持続的な接続。下の「Keep-Alive HTTP 接続」を参照してください。 ➤ HTML 圧縮。サーバ サイド圧縮を受け入れるを参照してください。 ➤ 仮想ホスティング。複数のドメイン名が同一の IP アドレスを共有します。
Keep-Alive HTTP 接続	Keep-Alive（キープ・アライブ）は，持続的な接続を可能にする HTTP 拡張機能を表す用語です。こうした長時間の HTTP セッションでは，複数のリクエストを同一の TCP 接続を介して送信できます。これにより，Web サーバとクライアントのパフォーマンスが向上します。 この Keep-Alive オプションは，Keep-Alive 接続をサポートする Web サーバがあって初めて機能します。この設定により，仮想ユーザ・スクリプトを実行するすべての仮想ユーザに対して Keep-Alive HTTP 接続を有効にすることができます。 **標準設定値**：有効。

UI 要素	説明
Accept-Language 要求ヘッダ	受け入れた言語のカンマ区切りリストを提供します。たとえば，**en-us**，**fr** などがあります。詳細については，1224 ページの「ページ要求ヘッダの言語」を参照してください。
HTTP エラーを警告とする	HTTP エラーによりリソースのダウンロードが失敗した場合に，エラーの代わりに警告が発行されます。
HTTP 要求接続タイムアウト (秒)	仮想ユーザが，ステップ内の特定の HTTP 要求への接続を中断するまで待機する時間（秒単位）です。タイムアウトによって，サーバは安定し，ユーザに応答するための機会が与えられます。このタイムアウトは，仮想ユーザが **wap_connect** 関数によって開始された WAP 接続を待機する時間にも適用されます。 **標準設定値**：120 秒。
HTTP 要求受信タイムアウト (秒)	仮想ユーザがステップ内の特定の HTTP 要求への応答を受信するまで待機する時間（秒単位）です。タイムアウトによって，サーバは安定し，ユーザに応答するための機会が与えられます。 **標準設定値**：120 秒。

UI 要素	説明
Zlib ヘッダを要求	リクエスト・データが **zlib** 圧縮ライブラリ・ヘッダとともにサーバに送信されます。標準では，サーバに送信されるリクエストには **zlib** ヘッダが含まれます。このオプションを使用すると，**zlib** ヘッダがリクエストに含まれないブラウザ以外のアプリケーションをエミュレートできます。これらのヘッダを除外するには，このオプションを「**いいえ**」に設定します。 **標準設定値**：はい。
サーバ サイド圧縮を受け入れる	再生で圧縮データを受け入れることができることをサーバに示します。使用可能なオプションは，「**None**（圧縮なし）」，「**gzip**（gzip 圧縮を受け入れる）」，「**gzip，deflate**（gzip または deflate 圧縮を受け入れる）」，および「**deflate**（deflate 圧縮を受け入れる）」です。圧縮データを受け入れると，CPU の処理量が大幅に増えることがあります。 標準設定値：**gzip** と **deflate** 圧縮を受け入れる。 圧縮を手作業で追加するには，スクリプトの先頭に次の関数を入力します。 web_add_auto_header("Accept-Encoding", "gzip"); サーバによって圧縮データが送信されていることを検証するには，Content-Encoding:gzip という文字列が再生ログのサーバの応答セクションにあることを確認します。ログには展開前と後のデータ・サイズも表示されます。

一般

UI 要素	説明
DNS のキャッシュ	ドメイン・ネーム・サーバによって解決されたホストの IP アドレスをキャッシュに保存するよう仮想ユーザに指示します。これにより，それ以降の同じサーバに対する呼び出しに費やす時間を節約できます。ロード・バランサーなどの技術によって自動的に IP アドレスが変更される場合には，このオプションを確実に無効にしておき，仮想ユーザがキャッシュの値を使用しないようにします。 **標準設定値**：有効。
UTF-8 から，または UTF-8 への変換	受信した HTML ページおよび送信されたデータの UTF-8 への変換と UTF-8 からの変換をします。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，313 ページの「記録オプション」を参照してください。 **標準設定値**：いいえ。
リソースによって発生したステップ タイムアウトを警告とする	タイムアウトの時間内にロードされなかったリソースが原因でタイムアウトが発生した場合に，エラーではなく警告を発行します。リソース以外の場合は，エラーが発行されます。 **標準設定値**：有効。
HTML の Content-Type を解析する	HTML を期待している場合に，Content-Type が **HTML**，**text¥html**，**TEXT**（任意のテキスト），**ANY**（任意の content-type）の場合にのみ応答が解析されます。text/xml は HTML としては解析されません。 **標準設定値**：**TEXT**。 タイムアウト設定は主に，該当システム環境下では，許容タイムアウト値を変えるのが望ましいと判断できる上級ユーザ向けです。ほとんどの場合，標準の値で問題ないはずです。サーバが妥当な時間内に応答しない場合は，タイムアウト時間を長くするのではなく，接続に関するほかの問題がないか調べてください。タイムアウト時間を長くすると，スクリプトが不必要に待機することになります。
ステップ ダウンロード タイムアウト (秒)	仮想ユーザがスクリプトのステップの実行を中止するまでに待機する時間です。このオプションはページのダウンロードに特定秒数以上は待たないユーザの操作をエミュレートするのに使用します。

UI 要素	説明
ネットワーク バッファ サイズ	HTTP 応答の受信に使用するバッファ・サイズの最大値を設定します。データのサイズが指定サイズより大きいと，サーバはデータをチャンクに分けて送信するため，システムのオーバーヘッドが増加します。Controller で複数の仮想ユーザを実行すると，各仮想ユーザは自身のネットワーク・バッファを使用します。この設定は，主にネットワーク・バッファ・サイズがスクリプトのパフォーマンスに影響を与える可能性があると判断した上級ユーザを対象にしています。標準設定は 12 キロバイトです。最大サイズは 0x7FFF FFFF です。
NTLM 情報を出力します	NTLM ハンドシェイクに関する情報が標準ログに出力されます。
SSL 情報を出力します	SSL ハンドシェイクに関する情報が標準ログに出力されます。
エラーとして発行された一致エラー数の上限	LB（左境界）または RB（右境界）を使用して，内容チェックにエラーとして発行されたエラーの一致数を制限します。これは，文字列が見つかったときに失敗となる（Fail=Found）一致に適用されます。以降の一致は，情報メッセージとして一覧表示されます。 **標準設定値**：10。
同じページに対する 'META refresh' の上限	META 更新がページごとに実行される最大回数。 **標準設定値**：2。
UTF-8 による ContentCheck の値	ContenCheck XML ファイルに UTF-8 形式の値が保存されます。

認証

UI 要素	説明
認証再試行時の固定思考遅延時間 (ミリ秒)	ユーザによる認証情報（ユーザ名とパスワード）の入力をエミュレートするため，思考遅延時間が仮想ユーザ・スクリプトに自動的に追加されます。この思考遅延時間はトランザクションに含まれます。 **標準設定値**：0。
NTLM2 セッション セキュリティを無効にする	より基本的な NTLM 2 セッション・セキュリティ応答ではなく，完全な NTLM 2 ハンドシェイク・セキュリティを使用します。 **標準設定値**：いいえ。
Windows のネイティブ NTLM 実装を使用する	固有 API の代わりに NTLM 認証のための Microsoft セキュリティ API を使用します。 **標準設定値**：いいえ。
統合認証を有効にする	Kerberos ベースの認証を有効にします。サーバによって認証スキームが提示されている場合は，別のスキームに優先して［**Negotiate**］を使用します。 **標準設定値**：いいえ。
大きい KDC 負荷をかける	前の反復で取得した資格情報を再利用しません。この設定を有効にすると，KDC（キー配布サーバ）にかかる負荷が高くなります。サーバにかかる負荷を減らすには，このオプションを「**Yes**」に設定して，前の反復で取得した資格情報を再利用します。このオプションは Kerberos 認証が使用されている場合にのみ関係します。 **標準設定値**：いいえ。

ログ

UI 要素	説明
出力バッファの行の長さ	行の折り返しを無効にして，要求，応答ヘッダ，ボディ，または JavaScript ソース，あるいはこれらすべてを印刷するための行の長さです。
出力バッファはバイナリ ゼロのみエスケープする	➤ **[はい]**：要求，応答ヘッダ，ボディ，または JavaScript ソース，あるいはこれらすべてを印刷する場合に，バイナリのゼロ値のみをエスケープします。 ➤ **[いいえ]**：印刷不能な文字や制御文字をすべてエスケープします。

Web（Click and Script）固有

UI 要素	説明
一般	➤ **[ホームページの URL]**：ブラウザを開いたときに表示されるホーム・ページの URL（標準設定値は about:blank）。 ➤ **[DOM ベースのスナップショット]**：サーバ応答の代わりに DOM からスナップショットを生成するよう VuGen に指示します。 **標準設定値**：はい。 ➤ **[HTTP による文字セットの変換]**："Content-Type:....; charset=..." HTTP 応答ヘッダに基づいて文字セットの変換を実行します。［UTF-8 から，または UTF-8 への変換］に優先します。 ➤ **[META による文字セットの変更があったら解析しなおす]**：META タグによる文字セットの変更があったら解析しなおします。［**HTTP による文字セットの変換**］が有効な場合にのみ設定できます。［**自動**］は，最初の反復で使用したときにのみ解析が有効になることを意味します。 ➤ **[JavaScript エラー時に失敗する]**：JavaScript の評価エラーが発生した場合に，仮想ユーザが失敗します。 **標準設定値**：いいえ（JavaScript エラーの後に警告メッセージだけが表示され，スクリプトの実行が継続されます）。 ➤ **[新規ウィンドウ オブジェクトごとに標準クラスを初期化する]**：有効にすると，スクリプト（SRC コンパイル済みスクリプト）はキャッシュされません。 ➤ **[無効になっている対象要素を無視する]**：無効にされている仮想ユーザ・スクリプト関数の対象要素が無視されます。

UI 要素	説明
タイマ	➤ **[ステップ終了時にタイマを最適化する]**：可能であれば，期限前に，ステップの終了時に期限切れになる setTimeout/setInterval/<META 更新 > を実行します。 **標準設定値**：はい。 ➤ **[Single setTimeout/setInterval threshold (秒)]**：window.setTimeout および window.setInterval メソッドの上限タイムアウトを指定します。遅延時間がこのタイムアウトを超えた場合，これらのメソッドは渡された関数を呼び出さなくなります。これにより，ユーザが指定した時間，次の要素をクリックするまでに待機する操作がエミュレートされます。 **標準設定値**：5 秒。 ➤ **[累積 setTimeout/setInterval しきい値 (秒)]**：window.setTimeout および window.setInterval メソッドのタイムアウトを指定します。遅延時間がこのタイムアウトを超えた場合，それ以上の window.setTimeout および window.setInterval 呼び出しは無視されます。タイムアウトはステップごとに累積されます。 **標準設定値**：30 秒。 ➤ **[ステップ終了時に setInterval を設定しなおす]**：**0** = いいえ，**1** = 一度のみ，**2** = はい
履歴	➤ **[履歴のサポート]**：テスト実行での window.history オブジェクトのサポートを有効にします。[**有効**]，[**無効**]，[**自動**] のいずれかを選択できます。[**自動**] を選択すると，最初の反復で使用されている場合にのみ window.history オブジェクトをサポートするよう仮想ユーザに指示します。このオプションを無効にすると，パフォーマンスが向上します。 **標準設定値**：自動。 ➤ **[履歴の最大サイズ]**：履歴リストに保存されるステップの最大数。 **標準設定値**：100 ステップ。

UI 要素	説明
ナビゲータのプロパティ	➤ **navigator.browserLanguage.** ナビゲータ DOM オブジェクトの **browserLanguage** プロパティに設定されているブラウザ言語です。 **標準設定値**：記録された値。標準設定では，古い記録エンジンで作成されたスクリプトは，**en-us** を使用します。 ➤ **navigator.systemLanguage.** ナビゲータ DOM オブジェクトの **systemLanguage** プロパティに設定されているシステム言語です。 **標準設定値**：記録された値。標準設定では，古い記録エンジンで作成されたスクリプトは，**en-us** を使用します。 ➤ **navigator.userLanguage.** ナビゲータ DOM オブジェクトの **userLanguage** プロパティに設定されているユーザ言語です。 **標準設定値**：記録された値。標準設定では，古い記録エンジンで作成されたスクリプトは，**en-us** を使用します。
画面のプロパティ	➤ **[screen.width]**：screen DOM オブジェクトの width プロパティをピクセル単位で設定します。 **標準設定値**：1024 ピクセル。 ➤ **[screen.height]**：screen DOM オブジェクトの height プロパティをピクセル単位で設定します。 **標準設定値**：768 ピクセル。 ➤ **[screen.availWidth]**：screen DOM オブジェクトの availWidth プロパティをピクセル単位で設定します。 **標準設定値**：1024 ピクセル。 ➤ **screen.availHeight**：screen DOM オブジェクトの availHeight プロパティをピクセル単位で設定します。 **標準設定値**：768 ピクセル。

UI 要素	説明
メモリ管理	➤ **[DOM メモリ割り当ての標準設定のブロック サイズ]**：DOM のメモリ割り当てに対して標準のブロック・サイズを設定します。この値が小さすぎると，余分な malloc 呼び出しが増え，実行時間が遅くなる場合があります。ブロック・サイズが大きすぎると，不必要に大きな領域が占有される場合があります。 **標準設定値**：16384 バイト。 ➤ **[動的に作成される DOM オブジェクトためのメモリ マネージャ]**：［**はい**］- 動的に作成される DOM オブジェクトにメモリ・マネージャが使用されます。［**いいえ**］- メモリ・マネージャは使用されません（例：SAP のように同じドキュメントで複数の DOM オブジェクトが動的に作成される場合）。［**自動**］- プロトコルが推奨する標準設定値が使用されます（標準設定では，SAP を除くすべてのプロトコルが「はい」）。 ➤ **[JavaScript ランタイム メモリのサイズ (KB)]**：JavaScript ランタイム・メモリのサイズをキロバイトで指定します。 **標準設定値**：256 KB。 ➤ **[JavaScript スタック メモリのサイズ (KB)]**：JavaScript スタック・メモリのサイズをキロバイトで指定します。 **標準設定値**：32 KB。

［インターネット プロトコル］>［プロキシ］ノード

プロキシ・サーバの接続を設定できます。

利用方法	**［仮想ユーザ］>［実行環境の設定］>［インターネット プロトコル］>［プロキシ］**
重要情報	このノードは特定のプロトコルにのみ適用されます。プロトコルの完全な一覧と各プロトコルに関連付けられたノードについては，424 ページの「プロトコルの互換性に関する表」を参照してください。

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
プロキシなし	すべての仮想ユーザが，インターネットに常に直接接続します。つまり，プロキシ・サーバを使用せずに接続します。
標準設定のブラウザからプロキシ設定を取得する	すべての仮想ユーザは，各仮想ユーザが実行されているマシンにある標準ブラウザのプロキシ設定を使用します。
ユーザ定義のプロキシを使用する	すべての仮想ユーザで，カスタム・プロキシ・サーバが使用されます。実際のプロキシ・サーバの構成情報や，自動設定のための設定スクリプト（**.pac** ファイル）へのパスなどで設定を行います。 ➤ **[自動設定スクリプトを使用する]**：プロキシ割り当て情報を含む JavaScript ファイルを指定できます。このスクリプトは，URL に基づいて，どの場合にはプロキシ・サーバを使用して，どの場合には直接サイトへ接続するのかをブラウザに指定します。また，このスクリプトでは，アドレスごとに異なるプロキシ・サーバを使うよう指定することができます。［**アドレス**］フィールドでスクリプトの場所を指定します。 ➤ ［プロキシ サーバを使用する］：**HTTP** サイト用にプロキシ・サーバを 1 つ指定し，**HTTPS** サイト（セキュア・サイト）用に別のプロキシ・サーバを指定するか，［**すべてのプロトコルに同じプロキシ サーバを使用する**］ボックスをチェックできます。 ➤ 例外(X)...：プロキシ・サーバ・ルールの例外を指定できます。 ➤ 認証(E)...：［プロキシ認証］ダイアログ・ボックスが開きます。プロキシ・サーバで仮想ユーザごとに認証を必要とする場合は，このダイアログ・ボックスを使用して，適切なパスワードとユーザ名を入力します。認証を記録時に動的に追加する場合や，複数のプロキシ・サーバの認証を追加する場合は，**web_set_user** 関数を使用します。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。

［Java 環境の設定］>［Java Classpath］ノード

［Classpath］セクションでは，システムのクラスパス環境変数に含まれていない追加のクラスの場所を指定できます。これらのクラスは，Java アプリケーションの実行や，正しい再生を保証するのに必要な場合があります。

コンピュータまたはネットワークから必要なクラスを検索し，特定のテストの場合にそのクラスを無効にすることができます。また，クラスの順序を変更して，クラスパスのエントリを操作することもできます。

利用方法	**［仮想ユーザ］>［実行環境の設定］>［Java 環境の設定］>［Classpath］**
重要情報	このノードは特定のプロトコルにのみ適用されます。プロトコルの完全な一覧と各プロトコルに関連付けられたノードについては，424 ページの「プロトコルの互換性に関する表」を参照してください。

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
	**［クラスパスの追加］**：クラスパスのリストに新しい行を追加します。
	**［削除］**：クラスパスを永久に削除します。
	**下向き矢印**：クラスパスのエントリをリストの下方向に移動します。
	**上向き矢印**：クラスパスのエントリをリストの上方向に移動します。
［Classpath エントリ］リスト	クラスパスのエントリのリスト。

［Java 環境の設定］>［Java VM］ノード

Java VM の実行環境を設定できます。

利用方法	**［仮想ユーザ］>［実行環境の設定］>［Java 環境の設定］>［Java VM］**
重要情報	このノードは特定のプロトコルにのみ適用されます。プロトコルの完全な一覧と各プロトコルに関連付けられたノードについては，424 ページの「プロトコルの互換性に関する表」を参照してください。

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
仮想マシンの設定	➤ **［JDK の検索に内部ロジックを使う］**：PATH，レジストリ，Windows フォルダで，再生時に使用する JDK が検索されます。 ➤ **［指定した JDK を使う］**：［再生時に，以下で指定する JDK を使います］： ➤ **［JDK］**：指定した JDK のホーム・ディレクトリ。 ➤ **［追加の VM パラメータ］**：仮想マシンで使用する任意のパラメータ。 ➤ **［-Xbootclasspath パラメータを使用しています…］**：Xbootclasspath /p オプションを指定してスクリプトを再生します。
クラスのロードの設定	➤ **［それぞれ専用のクラス ローダを使って各仮想ユーザをロードする］**：それぞれ専用のクラス ローダを使って各仮想ユーザがロードされます。これにより，仮想ユーザごとに一意の名前空間を使用して，リソースを別々に管理できるようになります。

［JMS］>［詳細］ノード

JMS の詳細な実行環境を設定できます。

利用方法	［仮想ユーザ］>［実行環境の設定］>［JMS］>［詳細］
重要情報	このノードは Web サービス・プロトコルにのみ適用されます。プロトコルの完全な一覧と各プロトコルに関連付けられたノードについては，424 ページの「プロトコルの互換性に関する表」を参照してください。
関連タスク	1058 ページの「実行環境を設定する（任意）」

ユーザ・インタフェース要素の説明は次のとおりです。

VM（仮想マシン）

UI 要素	説明
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，仮想ユーザは 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 以降がサポートされています。
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

UI 要素	説明
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。

UI 要素	説明
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。
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。
1 プロセスあたりの JMS 接続数	**mdrv** プロセス，または仮想ユーザごとの JMS 接続の数です。接続を共有するすべての仮想ユーザは，同じメッセージを受信します。標準設定は 1 仮想ユーザ，最大値は 50 仮想ユーザです。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。
受信メッセージのタイムアウト オプション	受信メッセージのタイムアウトです。 ➤ **[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です。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（秒単位）。標準設定は 20 秒です。 **標準設定値**：待機しない。
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［**いいえ**］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。

［.Net］>［.NET 環境］ノード

.NET の実行環境を設定できます。

利用方法	**［仮想ユーザ］>［実行環境の設定］>［.NET］>［.NET 環境］**
重要情報	このノードは特定のプロトコルにのみ適用されます。プロトコルの完全な一覧と各プロトコルに関連付けられたノードについては，424 ページの「プロトコルの互換性に関する表」を参照してください。

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
AUT 設定	AUT の場所および設定ファイルの設定。 ➤ **［AUT アプリケーション基底パス］**：再生時に DLL のロード元となる AUT（テスト対象アプリケーション）の基底ディレクトリです。標準設定では，記録時に必要なすべての DLL はスクリプトのディレクトリに格納されます。このオプションを使用して，見つからなかった AUT の DLL ファイルの場所を指定します。これは通常は，記録対象アプリケーションのインストール先パスです。AUT は，スクリプトを実行するマシンにインストールされている必要があります。このボックスを空白のままにしておくと，VuGen は再生時にアプリケーションの基底ディレクトリとしてローカルの script¥bin ディレクトリを使用します。 ➤ **［AUT 設定ファイル］**：記録されるアプリケーションの設定ファイルの名前。VuGen は，AUT 設定ファイルを **script¥bin** ディレクトリにコピーし，ローカルに保存されたファイルをロードします。別の場所を指定するには，完全パスを入力します。ファイル名だけが指定され，ファイルが **script¥bin** フォルダに存在しない場合，アプリケーションの基底ディレクトリからファイルがロードされます。
同時実行	**［仮想ユーザごとにアプリケーション ドメイン］**：各仮想ユーザを個別のアプリケーション・ドメインで実行できるようにします。仮想ユーザを個別のアプリケーション・ドメインで実行すると，静的変数が仮想ユーザの間で共有されないので，互いをロックすることなしに仮想ユーザを個別に実行できます。 **標準設定値**：True。

［MMS］>［サーバとプロトコル］ノード

MMS（マルチメディア・メッセージング・サービス）の実行環境を設定できます。

利用方法	**［仮想ユーザ］>［実行環境の設定］>［MMS］>［サーバとプロトコル］**
重要情報	➤ このノードは特定のプロトコルにのみ適用されます。プロトコルの完全な一覧と各プロトコルに関連付けられたノードについては，424 ページの「プロトコルの互換性に関する表」を参照してください。 ➤［**一般**］>［**その他**］ノードの［マルチスレッド］で［**仮想ユーザをプロセスとして実行する**］を選択します。

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
自動 WAP 接続	WAP ゲートウェイに対する接続 / 接続解除を行うタイミングを定義します。この設定は，WAP ゲートウェイを使用する場合にのみ関係します。次の値を指定できます。 **［Per Iteration］**：各反復の始まりに接続し，各反復の終わりに接続を解除します。 **［Per Send or Receive］**：各メッセージの始まりに接続し，各メッセージの終わりに接続を解除します。 **［なし］**：自動 WAP 接続を使用しません。 **標準設定値**：Per Iteration。
標準設定の送信元アドレス	Sender ヘッダで送信される標準設定のアドレスです。 **標準設定値**：+999999。
MMS バージョン	スクリプトで使用される MMS プロトコルのバージョンです。
MMSC URL	MMSC（マルチメディア・メッセージング・センタ）サーバの URL です。
SMSC IP	SMPP を介して MMS 通知を送信するのに使用される SMSC サーバの IP アドレスです。

UI 要素	説明
SMSC ポート	SMPP を介して MMS 通知を送信するのに使用される SMSC サーバの IP ポートです。
タイムアウト (秒)	サーバが着信メッセージを待機する時間です。 **標準設定値**：60 秒。

［ネットワーク］>［速度のシミュレーション］ノード

ネットワーク速度の実行環境を設定できます。

利用方法	**［仮想ユーザ］>［実行環境の設定］>［ネットワーク］>［速度のシミュレーション］**
重要情報	このノードは特定のプロトコルにのみ適用されます。プロトコルの完全な一覧と各プロトコルに関連付けられたノードについては，424 ページの「プロトコルの互換性に関する表」を参照してください。

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
帯域幅を使用する	仮想ユーザをエミュレートする帯域幅のレベルを指定します。アナログ・モデム，ISDN，DSL をエミュレートするため，14.4Kbps から 512 Kbps の範囲の速度を選択できます。
ユーザ定義の帯域幅を使用する	仮想ユーザをエミュレートする帯域幅の上限を指定します。帯域幅をビットで指定します（1 キロビット =1024 ビット）。
最大帯域幅を使用する	仮想ユーザは，ネットワークで利用可能な最大帯域幅で実行されます。 **標準設定値**：有効。

［Oracle NCA］>［クライアントのエミュレーション］ノード

Oracle NCA の実行環境を設定できます。

利用方法	**［仮想ユーザ］>［実行環境の設定］>［Oracle NCA］>［クライアントのエミュレーション］**
重要情報	このノードは特定のプロトコルにのみ適用されます。プロトコルの完全な一覧と各プロトコルに関連付けられたノードについては，424 ページの「プロトコルの互換性に関する表」を参照してください。

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
ネットワーク	➤ **［ソケット モード］**：クライアントとサーバ間の接続をソケット・レベルで設定します。 ➤ **Timeout**：サーバからの応答を Oracle NCA 仮想ユーザが待機する時間です。**標準設定値**：-1（タイムアウトは無効になり，クライアントはいつまでも待機します）。 ➤ **［プラグマ モード］**：Oracle で定義されたプラグマ・モード（ソケットおよび HTTP よりも高レベル）で接続を設定します。 ➤ **［再試行限度］**：クライアントがエラーを発行するまでにサーバから受け付ける **IfError** メッセージの最大数。**IfError** メッセージは，サーバからクライアントに定期的に送られるメッセージで，できるだけ早くデータを返すことを通知するものです。 ➤ **［再試行間隔 (ミリ秒)］**：**IfError** メッセージが生じた場合の再試行の間隔（ミリ秒）。 ➤ **［再試行間隔をトランザクションに含める］**：再試行の間の間隔もトランザクション持続時間に含まれます。 ➤ **［ハートビートを有効にする］**：サーバへハートビート信号が送信さます。ハートビートの頻度は，［**頻度**］プロパティで設定できます。 **標準設定値**：有効，120。

UI 要素	説明
接続	➤ **[Forms のバージョン]**：記録時に検出された Oracle Forms のバージョン。この設定は，記録を行った後にサーバがアップグレードされた場合にのみ変更します。 ➤ **[コマンド ライン パラメータのセパレータ]**：コマンド・ライン文字列で個々のパラメータ間を区切るマーカ。
診断	➤ **[アプリケーションのバージョン]**：Oracle Application のバージョン。このオプションは，Oracle Application を使用する場合に使用できます（ユーザ定義の Oracle NCA アプリケーションでは使用できません）。これは，Oracle データベース・ブレークダウンを使用するばあいにのみ必要です。

［RDP］>［詳細］ノード

RDP の詳細な実行環境を設定できます。

利用方法	［仮想ユーザ］>［実行環境の設定］>［RDP］>［詳細］
重要情報	このノードは特定のプロトコルにのみ適用されます。プロトコルの完全な一覧と各プロトコルに関連付けられたノードについては，424 ページの「プロトコルの互換性に関する表」を参照してください。

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
ビットマップのキャッシュ	リモート・デスクトップ・サーバでビットマップ・キャッシュを使用できるようにします。この設定を有効にすれば，リモート・デスクトップ・サーバのシステム・リソースを節約できます。
フォント スムージング	リモート・デスクトップ・サーバでフォント・スムージングを使用できるようにします。この設定を無効にすれば，リモート・デスクトップ・サーバのシステム・リソースを節約できます。
メニューとウィンドウのアニメーション	リモート・デスクトップ・サーバでメニューとウィンドウのアニメーションを可能にします。この設定を無効にすれば，リモート・デスクトップ・サーバのシステム・リソースを節約できます。

UI 要素	説明
リモート デスクトップの構成	リモート・デスクトップの構成を有効にします。
ドラッグ中にウィンドウの内容を表示	ドラッグしている間，ウィンドウの内容を表示します。この設定を無効にすれば，リモート・デスクトップ・サーバのシステム・リソースを節約できます。
リモート デスクトップの背景画像を表示	リモート・デスクトップにデスクトップの背景画像が表示されないようにして，リモート・デスクトップのアプリケーションを実行できます。この設定を無効にすれば，リモート・デスクトップ・サーバのシステム・リソースを節約できます。
ソケット受信バッファサイズ (バイト)	ソケットの受信バッファに割り当てるバイト数。バッファが小さすぎると，一杯になってサーバが切断される原因となります。バッファが大きすぎると，ローカルのシステム・リソース（メモリ）をより多く使用することになります。
テーマ	リモート・デスクトップ・サーバでの Windows のテーマの使用を可能にします。この設定を無効にすれば，リモート・デスクトップ・サーバのシステム・リソースを節約できます。

［RDP］>［RDP エージェント］ノード

RDP エージェントの実行環境を設定できます。

利用方法	**［仮想ユーザ］>［実行環境の設定］>［RDP］>［RDP エージェント］**
重要情報	このノードは特定のプロトコルにのみ適用されます。プロトコルの完全な一覧と各プロトコルに関連付けられたノードについては，424 ページの「プロトコルの互換性に関する表」を参照してください。

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
RDP エージェントの使用法	記録中に RDP エージェントを使用し，記録セッション中に RDP エージェントによって収集された情報を使用してスクリプトを生成するよう VuGen に指示します。サーバに LoadRunner RDP エージェントがインストールされている必要があります。
RDP エージェント ログの有効化	RDP エージェント・ログを有効にします。この機能は，デバッグの目的でのみ使用します。 ➤ **［RDP エージェント ログ詳細レベル］**：RDP エージェント・ログで生成される詳細のレベルを設定します。［**標準設定**］を指定すると最も低いレベルになり，［**拡張デバッグ**］を指定すると最も高いレベルになります。 ➤ **［RDP エージェント ログの出力先］**：RDP エージェント・ログ・データの出力先を設定します。［**File**］を指定すると，リモート・サーバ側にのみログ・メッセージが保存されます。［**Stream**］を指定すると，Vugen マシンにログ・メッセージが送信されます。［**FileAndStream**］を指定すると，両方の宛先にログ・メッセージが送信されます。 ➤ **［RDP エージェント ログ フォルダ］**：RDP エージェント・ログ・ファイルが生成されるリモート・サーバ上のフォルダのパス。このオプションに何も指定せず，エージェント・ログの宛先を［**File**］に設定した場合，ログは，サーバ上のユーザの一時フォルダに保存されます。

［RDP］>［設定］ノード

RDP 設定の実行環境を設定できます。

利用方法	［仮想ユーザ］>［実行環境の設定］>［RDP］>［設定］
重要情報	このノードは特定のプロトコルにのみ適用されます。プロトコルの完全な一覧と各プロトコルに関連付けられたノードについては，424 ページの「プロトコルの互換性に関する表」を参照してください。

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
RDP キャッシュ使用を有効にする	RDP でのデータ・キャッシュの並べ替えをサポートします（標準設定では有効）。
RDP クライアント バージョン エミュレーション	［**As Recorded**］または特定のバージョン番号を指定できます。
リモート デスクトップの色深度	再生の色深度設定。［**記録通り**］または特定の深度を指定できます。
リモート デスクトップ解像度 (ピクセル)	アプリケーションが実行されるウィンドウのサイズ。［**記録通り**］または特定のサイズ（ピクセル）を指定できます。
接続時に次のプログラムを開始	RDP 接続を開き，指定したアプリケーションを起動します。［**プログラム パスとファイル名**］および［**開始フォルダ**］（任意）を指定します。

［RDP］>［同期化］ノード

RDP 同期化の実行環境を設定できます。

利用方法	［仮想ユーザ］>［実行環境の設定］>［RDP］>［同期化］
重要情報	このノードは特定のプロトコルにのみ適用されます。プロトコルの完全な一覧と各プロトコルに関連付けられたノードについては，424 ページの「プロトコルの互換性に関する表」を参照してください。

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
標準設定の入力原点	標準設定の入力操作の原点。
標準設定のオフセット追加	後続のすべての関数のために，同期中に移動した画像のオフセットを保存します。 **標準設定値**：いいえ。
標準設定の同期のタイムアウト (秒)	同期化操作を待機する時間（秒単位）。0 ～ 1000 の値を入力します。 **標準設定値**：60。
画像の同期の標準設定の許容レベル	画像の同期化を実行するときの許容レベル。［**完全一致**］，［**低い**］，［**普通**］，［**高い**］のいずれかを選択します。［**高い**］は，変更や不一致に対して最も寛大です。［**低い**］はおよそ 95% の一致を要求，［**普通**］はおよそ 85% の一致を要求，そして［**高い**］はおよそ 70% の一致を要求します。また，［**完全一致**］は 100% の一致を要求します。 **標準設定値**：普通。
[同期化の失敗を無効にする] ダイアログ	選択した場合は，［同期化の失敗］ダイアログ・ボックスが開かないようになります。 **標準設定値**：選択されない。

UI 要素	説明
タイムアウトで画像の同期化ステップを失敗にする	同期化中に画像が見つからなかった場合の処理方法を指定します。［**はい**］を指定すると，失敗のステータスが設定され，仮想ユーザは［エラー時も処理を継続する］の設定に従います。［**いいえ**］を指定すると，LR_NOT_FOUND フラグが返され，ステップが警告を報告し，スクリプトが続行されます。 **標準設定値**：はい。
記録済み	入力原点が指定されていないすべての入力操作に座標を使用します。 **標準設定値**：有効。
同期済み	前のいずれかの同期化関数で保存された最新のオフセットを，入力原点が指定されていない各入力操作の記録済み座標に追加します。
タイピング速度 (ミリ秒 / 文字)	キーボード・コマンドの連続した文字を送信する間隔（ミリ秒単位）。0 ～ 1000 の値を入力します。 **標準設定値**：150。

［RTE］> ［RTE］ノード

RTE の実行環境を設定できます。

利用方法	**［仮想ユーザ］> ［実行環境の設定］> ［RTE］> ［RTE］**
重要情報	このノードは特定のプロトコルにのみ適用されます。プロトコルの完全な一覧と各プロトコルに関連付けられたノードについては，424 ページの「プロトコルの互換性に関する表」を参照してください。

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
最高接続試行回数	TE_connect 関数は，ホストへの接続を記録したときに VuGen によって生成されます。RTE 仮想ユーザ・スクリプトを再生すると，TE_connect 関数はターミナル・エミュレータを指定されたホストに接続します。最初の接続試行に失敗すると，仮想ユーザは一定の回数だけ接続を試行します。 **標準設定値**：5。
元のデバイス名を使用する	特定の環境では，各セッション（仮想ユーザ）に固有のデバイス名を付ける必要があります。TE_connect 関数は，各仮想ユーザに対して 8 文字からなる一意のデバイス名を生成し，この名前を使って接続します。TE_connect 関数の com_string パラメータ内に格納されているデバイス名を使用して接続する場合はこのオプションを選択します。 **注**：元のデバイス名の設定は，IBM ブロック・モード・ターミナルにのみ適用されます。
入力開始までの遅延時間	入力遅延の設定では，仮想ユーザによる **TE_type** 関数の実行方法を指定します。 ➤ **[第一キー]**：仮想ユーザが文字列の最初の文字を入力する前の待機時間（ミリ秒）を指定します。 ➤ **[後続キー]**：仮想ユーザが後続の文字を送信する前の待機時間（ミリ秒）を指定します。 **注**：仮想ユーザ・スクリプトの一部分で入力遅延の設定をオーバーライドするには，**TE_typing_style** 関数を使用します。
X- システムの同期化	➤ **Timeout**：**TE_wait_sync** 関数を再生するときに，エラーが返される前にシステムが安定するのを待機するタイムアウト（秒）。 ➤ **[安定時間]**：**TE_wait_sync** 関数を実行した後に，ターミナルが X-SYSTEM モードに入らなくなるのを確認するために仮想ユーザが待機する時間（ミリ秒）。

［SAPGUI］>［一般］ノード

SAPGUI の実行環境を設定できます。

利用方法	［仮想ユーザ］>［実行環境の設定］>［SAPGUI］>［一般］
重要情報	このノードは特定のプロトコルにのみ適用されます。プロトコルの完全な一覧と各プロトコルに関連付けられたノードについては，424 ページの「プロトコルの互換性に関する表」を参照してください。

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
ステータス バーのテキストを送信する	ステータス・バーからログ・ファイルへテキストが送信されます。
作業中ウィンドウのタイトルを送信する	作業中ウィンドウのタイトル・テキストがログ・ファイルへ送信されます。
再生時に SAP クライアントを表示する	再生中に SAP クライアントにアクションのアニメーションが表示されます。ユーザ・インタフェース（UI）を表示させる利点は，フォームにどのように入力が行われているかを確認でき，仮想ユーザのアクションを詳細に追えることです。しかし，このオプションではより多くのリソースが必要になるため，負荷テストのパフォーマンスに影響を与える場合があります。 ➤ **［再生時に ActiveScreen のスナップショットをとる］**：すべてのアクティブ・オブジェクトの Control ID 情報を含む再生スナップショットがキャプチャされます。ActiveScreen スナップショットは，通常のスナップショットと異なり，SAPGUI クライアントの中でどのオブジェクトが VuGen によって認識されているかを知ることができます。スナップショット上でマウスを動かすと，VuGen は検出したオブジェクトを強調表示します。スナップショット内からスクリプトに直接新しいステップを追加できます。また，特定のオブジェクトに対して，スナップショット内から対話的にステップを追加することもできます。詳細については，862 ページの「SAPGUI スクリプトを拡張する方法」を参照してください。

UI 要素	説明
オプション(O)...	［SAPGUI］の［詳細オプション］ダイアログ・ボックスが開き，次を設定できます。 ➤ **［実行中の SAPlogon アプリケーションを使用して再生する］**：再生のために現在実行中の SAPlogon アプリケーションを使うように仮想ユーザに指示します。 ➤ **［SAPfewgsvr プロセスのタイムアウトを設定する］**：**SAPfewgsvr.exe** プロセスのタイムアウトを修正できます。 ➤ **［SAPfewgsvr へのタイムアウト］**：SAPfewgsvr.exe プロセスのタイムアウトを秒単位で指定します。 **標準設定値**：300 秒。

［Silverlight］>［サービス］ノード

スクリプトに関連付けられた WSDL ファイルが表示され，再生段階の設定を変更できます。

利用方法	**［仮想ユーザ］>［実行環境の設定］>［Silverlight］>［サービス］**
重要情報	このノードは特定のプロトコルにのみ適用されます。プロトコルの完全な一覧と各プロトコルに関連付けられたノードについては，424 ページの「プロトコルの互換性に関する表」を参照してください。

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
<サービス・リスト>	このスクリプトに使用できる WSDL ファイルのリスト。
プロトコルとセキュリティ データ	［プロトコルおよびセキュリティ シナリオ データ］ダイアログ・ボックスが開き，選択された各 WSDL の設定を設定できます。詳細については，415 ページの「［プロトコルおよびセキュリティ シナリオ データ］ダイアログ・ボックス」を参照してください。

［VBA］>［VBA］ノード

Visual Basic の実行環境を設定できます。

利用方法	［仮想ユーザ］>［実行環境の設定］>［VBA］>［VBA］
重要情報	このノードは特定のプロトコルにのみ適用されます。プロトコルの完全な一覧と各プロトコルに関連付けられたノードについては，424 ページの「プロトコルの互換性に関する表」を参照してください。

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
［VBA 参照］リスト	スクリプト実行中に使用する参照ライブラリを選択します。ライブラリを選択すると，そのライブラリの詳細とバージョンがダイアログ・ボックスの下部に表示されます。
VBA コンパイラ オプション	➤ **［VBA IDE からスクリプトをデバッグする］**: Visual Basic IDE (統合開発環境) を使用したデバッグが可能になります。 ➤ **［エラー時 VBA IDE を常に表示する］**: スクリプト実行時に Visual Basic IDE が常に表示されます。

［WAP］>［ゲートウェイ］ノード

WAP ゲートウェイの実行環境を設定できます。

利用方法	［仮想ユーザ］>［実行環境の設定］>［WAP］>［ゲートウェイ］
重要情報	このノードは特定のプロトコルにのみ適用されます。プロトコルの完全な一覧と各プロトコルに関連付けられたノードについては，424 ページの「プロトコルの互換性に関する表」を参照してください。

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
HTTP 直接	Web サーバに直接アクセスする仮想ユーザが HTTP モードで実行されます。
[WAP ゲートウェイ] プロパティ	Web サーバに WAP ゲートウェイ経由でアクセスする仮想ユーザが実行されます。 ➤ **[IP]**：ゲートウェイの IP。 ➤ **[ポート]**：ゲートウェイのポート。WAP ゲートウェイ経由で仮想ユーザを実行する場合は，選択したモードに応じて標準のポート番号が自動的に設定されます。ただし，設定をカスタマイズして，ユーザ定義の IP アドレスとポートを指定することもできます。 ➤ **[WAP 1.x (WSP)]**：適切な WAP バージョンを選択します。WAP 1.x（WSP）で記録した場合は，1.x（WSP）モードまたは 2.0（HTTP プロキシ）モードで仮想ユーザを実行できます。このオプションを選択すると，WAP 1.x（WSP）のプロパティを設定できます。詳細については，下記を参照してください。 ➤ **[WAP 2.0 (HTTP)]**：適切な WAP バージョンを選択します。WAP 2.0（HTTP プロキシ）で記録した場合は，その同じモードでのみ仮想ユーザを実行できます。

WAP 1.x（WSP）のプロパティ

UI 要素	説明
詳細	詳細プロパティを設定する場合に展開します。詳細については，下記を参照してください。
接続オプション	➤ **[コネクション指向型]**：WSP セッションの接続モードを「コネクション指向」に設定します。 ➤ **[コネクションレス型]**：WSP セッションの接続モードを「コネクションレス」に設定します。
セキュリティを有効にする	[セキュリティを有効にする]：WAP ゲートウェイへの接続のセキュリティを有効にします。

詳細プロパティ

UI 要素	説明
確認応答ヘッダ	ゲートウェイに情報を提供する標準ヘッダを返します。 **標準設定値**：無効。
BearerType	伝送の基盤として使用されるベアラのタイプ。
CAPSessionResume	セッションの一時停止または再開の要求を可能にします。
クライアント SDU バッファ サイズ	セッションの実行中にクライアントへ送信可能な最大のトランザクション・サービス・データ・ユニット。 **標準設定値**：4000。
プッシュのサポートの確認	CO モードでプッシュ型メッセージが受信されたときに，メッセージの受信を確認するよう仮想ユーザに指示します。詳細については，1199 ページの「VuGen でのプッシュのサポート」を参照してください。
MethodMOR	同時に発生可能な未処理の方式の数。
ネットワーク MTU サイズ	ネットワーク・パケットの最大サイズ（バイト）。 **標準設定値**：4096。
プッシュのサポート	プッシュ・タイプのメッセージがゲートウェイを通過できるようにします。 **標準設定値**：無効。
PushMOR	同時に発生可能な未処理のプッシュ・トランザクションの数。
メッセージ取得	プッシュ型メッセージを受信すると，そのメッセージに示された URL からメッセージ・データを取得するよう仮想ユーザに指示します。 **標準設定値**：無効。
サーバ SDU バッファ サイズ	セッションの実行中にサーバへ送信可能な最大のトランザクション・サービス・データ・ユニット。 **標準設定値**：4000。
Cookie をサポート	クッキーの保存と取得をサポートします。 **標準設定値**：無効。

UI 要素	説明
WTLS 簡易ハンドシェーク	リダイレクト・メッセージの受信時に，完全なハンドシェイクではなく略式のハンドシェイクを使用します。 **標準設定値**：False。
WTLS Deffie Hellman	WTLS（Wireless Transport Layer Security）で，標準の暗号化方式である RSA ではなく Deffie-Hellman 暗号化方式を使用します。 **標準設定値**：False。
WTLS Deffie Hellman 識別子	Deffie-Hellman 暗号化方式の識別子。この識別子は，Deffie-Hellman 暗号化方式を使用する Operwave ゲートウェイによる略式のハンドシェイクに必要です。
WTP 再送信時間	WTP 層が応答の受信に失敗して PDU を再送信する前に待機する秒数。 **標準設定値**：5000。
WTP の分割と再組み立て	WTP（Wireless Transport Protocol）による分割と再組み立て（SAR）を有効にします。 **標準設定値**：True。

［WAP］>［Radius］ノード

WAP Radius の実行環境を設定できます。

利用方法	**［仮想ユーザ］>［実行環境の設定］>［WAP］>［Radius］**
重要情報	このノードは特定のプロトコルにのみ適用されます。プロトコルの完全な一覧と各プロトコルに関連付けられたノードについては，424 ページの「プロトコルの互換性に関する表」を参照してください。

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
アカウント ポート番号	Radius サーバのアカウント・ポート番号。
認証ポート番号	Radius サーバの認証ポート番号。
接続タイムアウト（秒）	Radius サーバが応答を待機する秒数。 **標準設定値**：120 秒。
IP アドレス	Radius サーバの IP アドレス。
ネットワークの種類	アカウント ネットワークの種類：GPRS（General Packet Radio Service）または CSD（Circuit-Switched Data）を選択します。
Radius クライアントの IP	Radius パケットの発信元 IP。通常は，1 台の Load Generator マシン上の異なる NIC カードから送信されるパケットを区別するために使用されます。
再送の再試行回数	伝送の失敗後に再送を試みる回数。 **標準設定値**：0。
秘密鍵	Radius サーバの秘密鍵。
サーバから返された属性をパラメータに保存する	仮想ユーザが，サーバから返された属性をパラメータ（後で使用可能）として保存できるようにします。 **標準設定値**：False。

第 II 部

プロトコル

12

AJAX プロトコル

本章の内容

概念

概念

AJAX プロトコルの概要

AJAX（Asynchronous JavaScript and XML）は，対話式の 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 手法を使用する 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 でサポートされているフレームワーク

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 のスクリプト例

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，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 がエ編集ボックスに入力されました。

13

Ajax TruClient プロトコル

本章の内容

概念

タスク

リファレンス

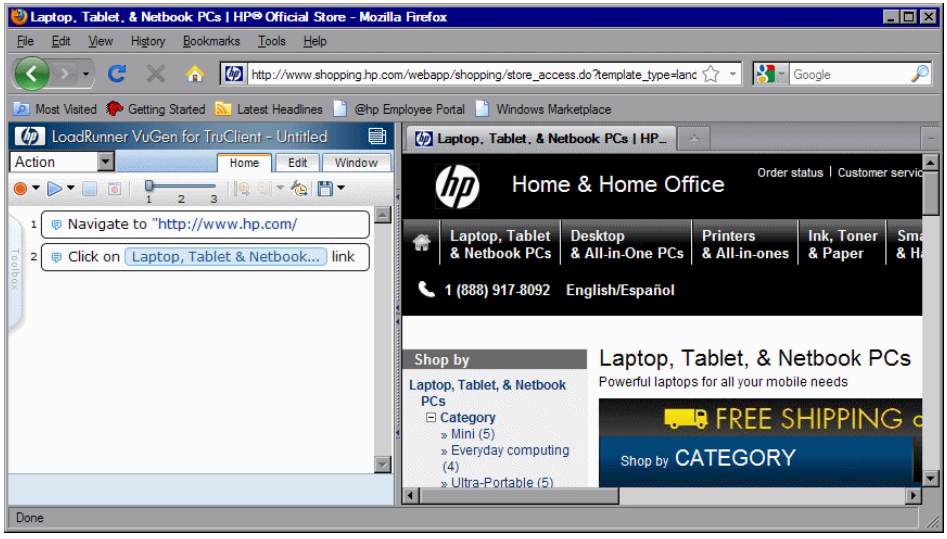
➤ Ajax TruClient のユーザ・インタフェース（535 ページ）

TruClient の制限（553 ページ）

概念

Ajax TruClient プロトコルの概要

Ajax TruClient プロトコルでは，ユーザ・レベルで対話的にスクリプトを記録します。これにより，VuGen は動的で複雑な Web ベース・アプリケーションを記録して，ユーザ・フレンドリなスクリプトを作成できます。スクリプトはリアルタイムで作成され，実行時に Mozilla Firefox の［LoadRunner VuGen for TruClient］タブでステップを表示できます。



ユーザ・インタフェースの詳細については，535 ページの「Ajax TruClient のユーザ・インタフェース」を参照してください。

VuGen の［スタート ページ］でデモ・ビデオを閲覧できます。ワークフローは刷新されており，大部分のワークフローはほかの VuGen プロトコルとは異なります。Ajax TruClient プロトコルとほかの VuGen プロトコルの主な違いを次に示します。

- VuGen のスクリプト・ビューにスクリプトは表示されない。スクリプトは、Mozilla Firefox の［VuGen］タブで作成および変更されます。
- TruClient スクリプトは非同期。前のステップが完了するまで次のステップを待機する必要はありません。各ステップで **［End Event］**が定義されています。終了イベントは，後続のステップが実行を開始できるポイントを定義します。
- TruClient スクリプトはユーザ・レベルで記録されるため相関がない。
- 記録時のすべてのイベントがスクリプトに保存される。無関係と判断されるイベントは別のスクリプト・レベルに割り当てられ，ユーザがこのレベルを手動で変更するまで再生されません。
- TruClient トランザクションは，ほかのプロトコルのようにステップ自体で定義されるのではなく，ステップのイベントで定義される。たとえば，ステップの［End Event］でスクリプトの続行を許可できます。そのステップで終了するトランザクションはトランザクションを定義するステップのイベントに達するまで続行されます。
- TruClient スクリプトの［実行論理］の制御が異なる。1 つのアクションしかありません。
- TruClient のステップ引数は値として JavaScript コードを使用できる。

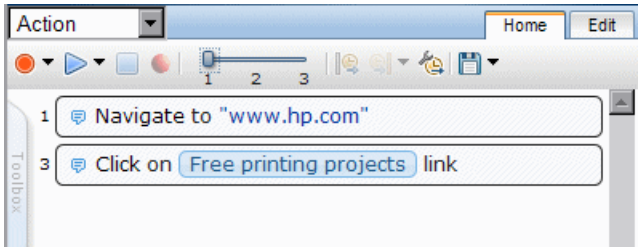
スクリプトの記録，再生，変更に伴う大部分のタスクは，Mozilla Firefox の［LoadRunner VuGen for Truclient］タブを使用して行われます。

スクリプト・レベル

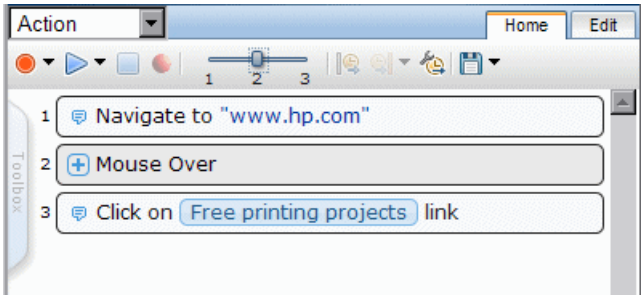
ビジネス・プロセスを記録するプロセスの一部として、再生中に記録が必要ない場合、ユーザによって実行されるステップもあります。VuGen は，必要ないと判断したステップを削除し，別のスクリプト・レベルに配置します。たとえば，影響のないアプリケーションの領域で発生したクリックのステップはレベル 2 に配置されます。VuGen は，このステップが重要ではなく，ユーザがアプリケーションのビジネス・プロセスをエミュレートするのに役立たないとみなします。再生フェーズでは，表示されているステップのみが実行されます。標準設定のビューでは，レベル 1 のステップのみが表示されます。レベル 2 および 3 のステップも同様に表示するには，［Home］タブのスライド・バーを使用します。

場合によっては，VuGen による決定を無効にして，特定のステップのレベルを手動で変更することもできます。これは，マウス・オーバーなどのステップが再生時に必要な場合に行います。通常，VuGen ではマウス・オーバーは再生する必要がないステップとしてみなされ，レベル 3 に割り当てられます。詳細については，518 ページの「スクリプト・レベルの変更および表示」を参照してください。

次のスクリーン・ショットには小さなスクリプトが表示されています。ステップ番号が 1 から 3 に飛んでいます。ステップ 2 は異なるレベルであるため非表示になっています。



スライド・バーを使用して表示設定を変更すると，すべてのステップが表示されるようになり，対話式モードで再生すると実行されます。



TruClient のスナップショット

TruClient では，記録時に自動的にスナップショットが生成されます。各ステップのアイコンの上にマウスを移動すれば，これらのスナップショットを表示できます。スナップショットは，ステップのアクションが実装される前に作成されます。スナップショットは .png ファイルとして保存されます。各スナップショットをクリックすると，Firefox の新しいタブにスナップショットが表示されます。再生する前に，適切なタブがアクティブになっていることを確認してください。記録したスナップショットは，スナップショット・ディレクトリに保存されます。

TruClient では，［実行環境の設定］で指定した内容に応じて再生時やロード・モード時にスナップショットを生成することもできます。詳細については，448 ページの「［一般］> ［その他の設定］ノード」を参照してください。

再生時のスナップショットは結果ディレクトリに保存され，再生のタイプ（対話式またはロード），スクリプト・セクション，反復に応じて編成されます。

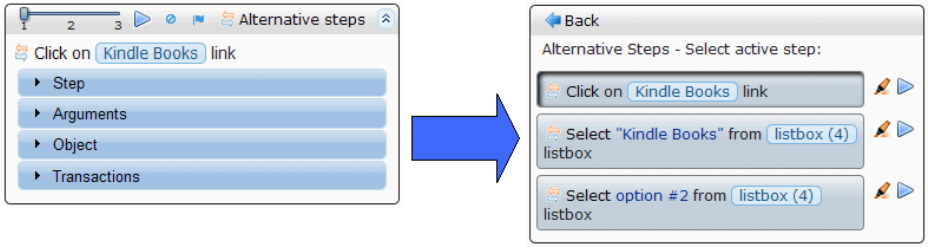
注：スナップショットの標準設定の場所は今後のリリースで変更される可能性があります。

代替ステップ

代替ステップでは，1 つのステップで同じアクションを複数の方法で実行できるインスタンスを表示できます。このようなステップを変更して特定のアクションを実行し，スクリプトをデバッグまたは拡張できます。たとえば，ドロップダウン・リストの場合，VuGen ではリストに表示される名前で選択するのか，番号で選択するのかを指定できます。

代替オプションのあるステップには，代替ステップの記号 が表示されます。そのステップの代替オプションを表示するには，この記号をクリックします。目的の代替オプションをクリックして［**戻る**］を選択します。

「Kindle Books」というドロップダウン・リストの 2 番目の項目が選択されたステップのスナップショットを次に示します。代替ステップ機能により，リンク「Kindle Books」をクリックするステップを定義するときに，ドロップダウン・メニューからオブジェクト「Kindle Books」を選択するか，ドロップダウン・メニューの 2 番目の項目を選択するかを選択できるようになります。



Firefox ブラウザのグローバル設定

Firefox で開かれる各 TruClient スクリプトには異なるプロファイルが使用されます。Firefox のプロファイルでは，Cookie，クライアント証明書，履歴，キャッシュなどのユーザ・データが保存されます。スクリプト・プロファイルに保存されているデータを変更するには，スクリプトを対話的に作成するときに Firefox ウィンドウで変更します。これらの変更は現在のスクリプトにのみ適用されます。

対話式モードに影響する Firefox のオプションは，［**グローバル設定**］ダイアログ・ボックスで変更できます。このダイアログ・ボックスのすべての設定は，新規作成される各スクリプトにインポートされますが，対話式モードにのみ影響します。スクリプトを保存すると，これらの設定が［実行環境の設定］にもコピーされます。

ロード・モードで実行されるスクリプトでは，［**実行環境設定**］ダイアログ・ボックスの［**Load Mode Browser Settings**］ノードで定義した設定が使用されます。

Firefox の拡張設定は，Firefox でスクリプトを開くたびに各スクリプトにインポートされます。詳細については，537 ページの「［Ajax TruClient General Settings］ダイアログ・ボックス」を参照してください。

TruClient でスクリプトを作成するために使用する Firefox のバージョンでは，Firefox の一部の標準オプションが無効になります。［ファイル］，［ブックマーク］，［ツール］，［ヘルプ］オプションの一部が無効になりますが，これに限定されるわけではありません。スクリプトで使用するためにブックマークを保存するには，537 ページで説明されているように［Ajax TruClient General Settings］ダイアログ・ボックスを使用します。

Ajax TruClient スクリプトでの作業に関する注意事項

次のセクションでは，Ajax TruClient スクリプトを記録するときの注意事項について説明します。

JavaScript がサポートされる部分

各ステップの［Arguments］セクションに表示される引数はすべて JavaScript ベースで，JavaScript 式を使用できます。文字列値を入力するには，引用符が必要です。たとえば，City は変数として解釈されますが，"City" や 'City' は文字列として評価されます。

ほかのすべてのセクション（［Step］，［Object］，［Transactions］など）は JavaScript ベースではないため JavaScript として評価されません。また JavaScript 式もサポートされていません。オブジェクトの識別だけは例外的に JavaScript を使用できます。

ただし，オブジェクト識別変数とステップ変数は同じコンテキストを共有しません。一方のコンテキストで定義した変数はもう一方のコンテキストでは認識されません。ステップで定義した変数をオブジェクトの識別に使用するには，変数名の前に **ArgsContext** というプレフィックスを追加します。たとえば，ステップの引数値として変数 **firstname** を定義しても，オブジェクトの識別には使用できません。**firstname** 変数をオブジェクトの識別に使用するには，**ArgsContext.firstname** として変数を参照します。

JavaScript の学習方法

JavaScript は Web のスクリプト言語です。JavaScript は，機能の追加，フォームの検証，ブラウザの検出などを目的として何百万もの Web ページで使用されています。インターネットには JavaScript を学習するためのリソースが豊富にあり，検索エンジンで簡単に特定できます。

次のようなチュートリアルやリファレンスがあります。

http://www.javascriptkit.com/jsref/

http://www.w3schools.com/js/default.asp

http://www.learn-javascript-tutorial.com/

スクリプトのデバッグに関するヒント

TruClient のすべての引数で JavaScript がサポートされているため，スクリプトの作成時に［警告］機能を使用して情報を表示できます。また，［場所］などの通常の機能を使用して DOM 要素を参照することもできます。

デバッグ機能をさらに向上させるには，アプリケーション・オブジェクトのプロパティに関する追加情報を提供する DOM Inspector や Firebug などのプラグインをインストールします。

対話式再生を成功させるための一般的なヒント

➤ 記録と再生間や再生中に Firefox ブラウザのサイズを変更しない。

サイズを変更すると，オブジェクトが移動して，VuGen でオブジェクトを特定できなくなる可能性があります。

注： これは，サイズを変更するとオブジェクトの相対位置が変わってしまう可能性があるため，［Related Objects］機能を使用している場合は特に重要です。

- 対話式再生中にアプリケーションを切り替えない。Firefox にフォーカスした状態を保ち，ブラウザで作業しないようにします。
- アプリケーションがフォーカス・ステータスの影響を強く受ける場合，特定のステップの再生に問題がある可能性があります。これは，フォーカスをキャプチャするオブジェクトまたは特定のフォーカスを要求するオブジェクトが原因で発生する場合があります。いずれの場合も，問題のあるステップの直前で別のオブジェクトをクリックしてフォーカスの状態を変更できます。

記録を成功させるための一般的なヒント

記録中に矢印キー，Tab キー，Esc キー，マウスの中央キーを使用しない。

- これらの機能に対応するマウス操作を行ってください。
- 記録中にアプリケーションのサイズを変更しない。

ステップ・タイムアウトの解決

ステップはいくつかの理由によりタイムアウトする場合があります。

- アプリケーションの応答が遅くなり，場合によっては負荷がかかった状態になる。実際には，これは重要なテスト結果です。
- ステップ・タイムアウトが不正で，ステップのプロパティの［Step］セクションで変更する必要がある。
- ステップの終了イベントが不正で，発生しないイベントをステップが待機している。ステップのプロパティの［Step］セクションで終了イベントを変更する必要があります。

正規表現の使用

正規表現を使用する方法は 2 つあります。

- 「/」表記を使用する（文字列の引用符をスラッシュに置き換えます）。

  次に例を示します。

  /LoadRunner/ は，単語 "LoadRunner" を含む任意の文字列に一致する正規表現です。

- （パラメータなどを使用して）動的に正規表現を作成する場合は，正規表現コンストラクタを使用して文字列を指定する。たとえば，上記の例と同じようにするには RegExp(「LoadRunner」) を使用します。

  サポートされている正規表現の一覧は次の場所にあります。

  https://developer.mozilla.org/en/JavaScript/Reference/Global_Objects/RegExp

Firefox のプライベート・ブラウジング

プライベート・ブラウジングは，ユーザのセッションに関する情報を保存せずに参照できる Mozilla Firefox のモードです。保存されない項目は，パスワード，Cookie，履歴などです。

VuGen では，実際のユーザをより正確にエミュレートするためにプライベート・ブラウジング・モードでスクリプトが再生されます。これにより，スクリプトを複数回実行しても，保存されたセッション情報がブラウザで使用されなくなります。

Firefox のメニューでプライベート・ブラウジングの有効 / 無効を手動で変更できます。

タスク

Ajax TruClient スクリプトの記録方法

このタスクでは，Ajax TruClient 仮想ユーザ・スクリプトを対話的に記録する場合の基本的なステップについて説明します。

このタスクでは，次の手順を実行します。

- ➤ 511 ページの「実行環境を設定する」
- ➤ 511 ページの「ブラウザのグローバル設定を設定する」
- ➤ 511 ページの「スクリプトの作成を開始する」
- ➤ 511 ページの「対話的に記録する」
- ➤ 512 ページの「スクリプトを拡張する」
- ➤ 512 ページの「Firefox でスクリプトを再生する」
- ➤ 512 ページの「作成を停止する」
- ➤ 512 ページの「ロード・モードでスクリプトを再生する」

1 実行環境を設定する

記録して負荷テストを実行する前に実行環境を設定します。[実行環境設定] ダイアログ・ボックスを開くには，F4 キーを押します。詳細については，419 ページの「実行環境の設定」を参照してください。

2 ブラウザのグローバル設定を設定する

Firefox ブラウザの設定では，すべての TruClient スクリプトに適用される設定を指定できます。この設定は新規作成されるスクリプトにインポートされます。詳細については，505 ページの「Firefox ブラウザのグローバル設定」を参照してください。これらの設定を編集するには，VuGen のメイン・ウィンドウにある［記録］ツールバーの［**Ajax TruClient General Settings**］ボタン をクリックし，［**Browser Settings**］タブを選択します。詳細については，537 ページの「[Ajax TruClient General Settings］ダイアログ・ボックス」を参照してください。

3 スクリプトの作成を開始する

［**スクリプトの開発**］をクリックして，Mozilla Firefox 対話的な記録のセッションを初期化します。

4 対話的に記録する

開始点となる目的の Web サイトに移動し，［記録］をクリックします。ビジネス・プロセスを実行すると，すべてのアクションが記録されて左側の［VuGen］タブに表示されます。スクリプトを一時停止または停止したり，スクリプトの任意のポイントから記録を再開したりできます。

スクリプトの別のセクションに記録するには，ツールバーの上にあるドロップダウン・バーを使用します。

5 スクリプトを拡張する

さまざまな方法（パラメータ，トランザクション，ループ，検証ステップの挿入など）でスクリプトを拡張できます。タスクの詳細については，513 ページの「Ajax TruClient スクリプトの拡張方法」を参照してください。

6 Firefox でスクリプトを再生する

スクリプトを最低でも 2 回は再生し，このプロセスで発生したエラーを修正します。2 回連続で正常に再生できたら，次のステップに移ることができます。エラーが継続して発生する場合は，516 ページの「Ajax TruClient スクリプトのデバッグ方法」を参照してください。

7 作成を停止する

［Save］ボタン をクリックしてスクリプトを保存します。Firefox ウィンドウを閉じます。

8 ロード・モードでスクリプトを再生する

TruClient スクリプトの実行は負荷テストの実行時とは若干異なるため，ロード・モードが作成されました。このモードでは，負荷テストとまったく同じようにスクリプトを実行できます。ロード・モードでスクリプトを再生するには，VuGen のメイン・ウィンドウで［スクリプトの開発］ボタンの横にある矢印をクリックします。進行状況は対話式再生ログで監視できます。Firefox は開きません。またスナップショットも表示されません。

注：Firefox のこのインスタンス内で行ったカスタマイズ（ブックマークなど）はグローバルに保存されません。これは，VuGen では各スクリプトが固有の Firefox プロファイルで開かれるためです。このスクリプトを作成する以外の目的（インターネット・ブラウジングなど）で Firefox を使用する場合，ほかの Firefox ウィンドウを開くことをお勧めします。

Ajax TruClient スクリプトの拡張方法

基本的なワークフローには含まれていないさまざまな拡張機能をスクリプトに追加できます（任意）。このタスクでは，拡張機能とその使用方法について説明します。

このタスクでは，次の手順を実行します。

- 513 ページの「ステップの変更」
- 513 ページの「ループの挿入」
- 514 ページの「If ブロックまたは If-else ブロックと exit ステップの挿入」
- 514 ページの「コメントの挿入」
- 514 ページの「トランザクションを挿入する」
- 514 ページの「パラメータの作成」
- 516 ページの「Catch Error ステップの挿入」
- 516 ページの「オブジェクトの存在の検証」
- 516 ページの「汎用ステップの挿入」

ステップの変更

ステップの引数およびオブジェクトを変更するには，目的のステップを選択してオプションを展開します。これにより，ステップが展開してオブジェクトおよびプロパティを変更できるようになります。ステップの構造の詳細なリストについては，540 ページの「ツールボックス」を参照してください。

ループの挿入

ループでは，特定の条件を満たすか，指定した回数に達するまでスクリプトの選択した部分が繰り返されます。ループを挿入するには，［**Toolbox**］>［**Flow Control**］>［**For loop**］を選択します。詳細については，524 ページの「ループを挿入および変更する方法」を参照してください。

If ブロックまたは If-else ブロックと exit ステップの挿入

スクリプトの特定部分に条件を設定するには，If ブロックまたは If-else ブロックを挿入します。If ブロックを挿入するには，［**Toolbox**］>［**Flow Control**］>［**If block**］を選択します。else 条件を追加するには，If ステップのタイトルの横にある［**Add else**］リンクをクリックします。詳細については，526 ページの「TruClient のステップ引数」を参照してください。

exit ステップでは，スクリプトの反復またはスクリプト全体が終了します。exit ステップと If ステートメントを併用すると，指定した条件が発生した場合にスクリプトまたは反復を終了させることができます。exit ステップを挿入するには，［**Toolbox**］>［**Flow Control**］>［**Exit**］を選択します。

コメントの挿入

コメントをスクリプトに挿入するには，［**Toolbox**］>［**Miscellaneous**］を選択し，［**Comment**］アイコンを目的の場所にドラッグします。

トランザクションを挿入する

トランザクション・エディタを使用してトランザクションを追加できます。トランザクション・エディタを開くには，［Home］タブの［Transaction Editor］ボタン をクリックするか，Ctrl + Alt + F7 キーを押します。TruClient トランザクションの動作は，TruClient ステップの非同期性により，ほかのプロトコルとは異なります。トランザクションは，開始ステップと終了ステップおよびステップのイベントに基づいて定義されます。この定義により，実際にステップが終了する前にトランザクションの終了がトリガされる可能性があります。

パラメータの作成

すべてのプロトコルに関して，Ajax TruClient スクリプトのパラメータは標準的な方法で作成できます。

パラメータは，ステップ引数または Eval JavaScript ステップ内で参照および作成できます。

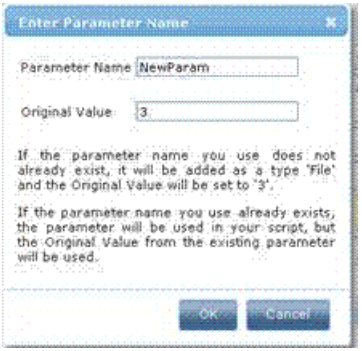
値をパラメータ化するには，次の手順で行います。

a ステップの［Arguments］領域のフィールドを選択します。

b 文字列全体を強調表示します。

c 右クリックして［**Replace with a Parameter**］を選択します。

d ［Enter Parameter Name］ダイアログ・ボックスに新規または既存のパラメータの名前を入力します。



［**パラメータ リスト**］にパラメータがない場合，パラメータが作成されて標準設定のパラメータ・タイプが［**File**］に設定されます。［**パラメータ リスト**］に移動して，指定したパラメータの値を追加または削除できます。標準設定では，新しいパラメータの値リストに既定値が含まれます。

スクリプトでパラメータを使用する方法の例を次に示します。この文字列は，Evaluate JavaScript ステップの引数として使用できます。

再生時にはパラメータ **paramname** の現在の値が返されます。

```
LR.getParam("paramname")
```

再生時にはパラメータ **paramname** に値 **value1** が割り当てられます。

```
LR.setParam("paramname", "value1")
```

たとえば，入力したテキストに基づいてさまざまな値のセットを表示するオートコンプリート・リストがあるとします。

入力するテキストがパラメータで定義されると，オプションのテキストに基づいてリストのオプションを選択できなくなります。オプションはパラメータ値が更新されるたびに変更されます。

このような場合，序数値を使用するステップの方がより適しています。

パラメータの詳細については，263 ページの「パラメータ」のセクションを参照してください。

Catch Error ステップの挿入

catch error ステップは，前のステップにエラーがある場合にコンテンツを実行するグループ・ステップです。また，エラーは「捕捉」されますが，返されることはありません。Catch Error ステップを定義して，すべてのエラーや特定のタイプのエラーを捕捉できます。2 つの catch error ステップが連続して存在する場合，両方とも同じステップに適用されます。Catch Error ステップを挿入するには，［**Toolbox**］>［**Flow Control**］>［**Catch Error**］を選択します。

オブジェクトの存在の検証

文字列またはオブジェクトがアプリケーションに存在しているかどうかを検証するには，verify ステップを挿入します。

a ［**Toolbox**］>［**Functions**］を選択し，［**Verify**］アイコンを目的の場所にドラッグします。

b verify ステップのオブジェクトをクリックします。

c 検証するオブジェクトを選択します。

汎用ステップの挿入

空のステップを挿入し，手動で設定できます。汎用ステップを挿入するには，［**Toolbox**］>［**Functions**］>［**Generic Object/Browser Action**］を選択してステップを展開し，目的のステップのプロパティを入力します。［Generic Object Actions］では，未指定のアクションがオブジェクトで実行されます。［Generic Browser Actions］では，未指定のアクション（戻る，再ロード，タブの切り替えなど）がブラウザで実行されます。

Ajax TruClient スクリプトのデバッグ方法

このタスクでは，Ajax TruClient 仮想ユーザ・スクリプトを対話的に記録する場合の基本的なステップについて説明します。

このタスクでは，次の手順を実行します。

➤ 517 ページの「Firefox での再生エラーの表示」

➤ 517 ページの「段階的なスクリプトの実行」

➤ 517 ページの「再生ログの表示」

➤ 517 ページの「ブレークポイントの挿入」

➤ 517 ページの「スナップショットを使用したスクリプトのデバッグ」

➤ 518 ページの「スクリプト・レベルの変更および表示」

➤ 518 ページの「Wait ステップの挿入」

Firefox での再生エラーの表示

再生時に失敗したステップにはエラー・アイコン が付きます。これらのアイコンの上にマウスを移動すると，エラーの説明が表示されます。

段階的なスクリプトの実行

スクリプトを段階的に実行し，自由なタイミングでゆっくりと再生を表示できます。スクリプトを段階的に実行するには，Firefox の［Replay］ボタンの下向き矢印を選択し，［**Replay step by step**］を選択します。段階的再生を続けるには，各ステップの後でこの手順を繰り返します。

再生ログの表示

VuGen の出力ウィンドウで，再生ログおよび対話式再生ログのスクリプトの再生の詳細を表示できます。詳細については，87 ページの「出力ウィンドウ」を参照してください。

ブレークポイントの挿入

対話式モードの場合，再生時にブレークポイントによってスクリプトの実行が停止します。ブレークポイントはスクリプトのデバッグに役立ちます。ブレークポイントを挿入するには，目的のステップを選択して［ブレークポイント］ボタン をクリックします。

スナップショットを使用したスクリプトのデバッグ

再生時に生成されたスナップショットを使用してスクリプトをデバッグできます。これを行うには，失敗したステップのスナップショットを表示します。

a ［実行環境設定］ダイアログ・ボックスを開き，［**一般**］>［**その他の設定**］ノードに移動します。

b ［**Replay Snapshot Generation**］を［**On Error**］に設定します。

c スクリプトを再生します。

d 再生ログまたは対話式再生ログにエラーがないか調べます。エラーのあったステップのステップ番号を記録します。

e VuGen のメイン・ウィンドウで，右クリックして［**スクリプト ディレクトリを開く**］>［**Results**］>［**Interactive**］を選択し，関連するセクションおよび反復を選択します。

これで，スクリプトでエラーが発生した一連のスナップショットを取得しました。

スクリプト・レベルの変更および表示

記録された再生に必要なステップがレベル 2 やレベル 3 に配置されることがあります。この場合，これらのステップのレベルを手動でレベル 1 に変更する必要があります。

- スクリプトの再生レベルを変更するには，ツールバーのスライダを目的のレベルにドラッグします。スライダをレベル 3 にドラッグすると，レベル 1，2，3 のステップが表示および再生されます。
- ステップを別のレベルに移動するには，ステップを開いて［Step］セクションをクリックします。スライダを目的のレベルに移動します。ステップがグループ・ステップの一部である場合，グループ・ステップと個々のステップの両方を変更する必要があります。

詳細については，503 ページの「スクリプト・レベル」を参照してください。

Wait ステップの挿入

Wait ステップでは，次のステップを続行する前に指定の時間スクリプトが一時停止します。Wait for Object ステップでは，次のステップを続行する前に指定のオブジェクトがロードされるまでスクリプトが待機します。Wait ステップは，TruClient プロトコルの動的性により，ほかのプロトコルの思考遅延時間ステップとは異なります。Wait ステップは，前のステップの **End Event** に達してから開始されます。つまり，Wait ステップに達しても前のステップの実行が続行されている可能性があります。待機を挿入するには，［**Toolbox**］>［**Functions**］を選択し，［**Wait**］アイコンまたは［**Wait for Object**］アイコンをスクリプトの目的の場所にドラッグします。Wait ステップでは指定の時間待機し，Wait for Object ステップでは指定のオブジェクトがアプリケーションに表示されるまで待機します。Wait for Object ステップでは，［**Click to choose an object**］ボタンをクリックしてアプリケーションのターゲット・オブジェクトを選択します。

オブジェクトの識別の問題を解決する方法

動的な Web サイトでは，記録されたオブジェクトの移動やコンテンツの変更が頻繁に発生する可能性とがあります。オブジェクトの識別は，Web 2.0 アプリケーションを記録および再生するときの最大の課題の 1 つです。この問題により，スクリプトでオブジェクトを特定できなくなる可能性があります。

Ajax TruClient には，この問題を解決するための非常に洗練されたメカニズム（[Highlight]，[Improve Object Identification]，[Replace Object]，[Related Object] オプションなど）があります。ウィンドウで記録されたアプリケーションのオブジェクトを識別する場合，[Window] タブを使用して適切なウィンドウが選択されていることを確認します。次の手順では，これらの問題を解決する方法について説明します。

[Highlight]，[Improve Identification]，[Replace]，[Related Objects] のいずれでも，ユーザがアプリケーションでオブジェクトを選択する必要があります。オブジェクトを表示するためにアプリケーションでさまざまなアクション（マウス・オーバーやマウス・クリックなど）を実行する必要がある場合もあります。このような場合，CTRL+ALT+F4 オプションを使用して，オブジェクトが表示されるまで TruClient のオブジェクト選択モードを一時停止します。オブジェクトを選択するには CTRL+ALT+F4 を再度押します。

ヒント：変更したら，失敗した問題のステップを 1 つ再生し，その後でスクリプト全体を再度実行します。こうすることで，発生した問題がその変更で解決したのかどうかを確認しやすくなります。

次の手順では，オブジェクトの識別の問題を解決する方法について説明します。

➤ 520 ページの「オブジェクトの強調表示」

➤ 520 ページの「オブジェクトの識別の改善」

➤ 521 ページの「オブジェクトの識別メソッドの変更」

➤ 522 ページの「スクリプトのタイミングの変更」

➤ 522 ページの「オブジェクト間の関連付け」

➤ 523 ページの「オブジェクトの置換」

オブジェクトの強調表示

使用しているオブジェクトの識別メソッドに関係なく，［Highlight］ボタン を使用して，オブジェクトがアプリケーションで表示されているかどうかをいつでも確認できます。オブジェクトが見つからない場合，ペースとタイミングの問題である可能性があります。オブジェクトが見つからない場合，エラー・メッセージが表示されます。

オブジェクトの識別の改善

［Highlight］オプションで失敗した場合，［Improve Object Identification］を使用します。これにより，TruClient がオブジェクトのプロパティを再学習し，記録時に学習したプロパティと比較します。その差異に基づいて，必要な調整を行うことができます。アプリケーションの動的性によっては，［Improve Object Identification］機能を複数回使用する必要があります。

これが完了したら，ステップをもう一度再生して問題が解決しているかどうかを確認します。

代替ステップ

代替ステップでは，1 つのステップで同じアクションを複数の方法で実行できるインスタンスを表示できます。［Improve Object Identification］で失敗した場合，いずれかの代替ステップを使用します。

たとえば，特定の値に基づいてテキストが変わるドロップダウン・リストのオプションをクリックする場合を考えます。

テキストに基づいてクリックすると，ステップが失敗する可能性があります。

リスト内のオプションの序数値に基づいてリストの項目を選択する代替ステップを使用すると，このクリックはテキストに関係なく成功します。

注：いずれかの代替方法を選択する前に，代替ステップで使用されるオブジェクトを強調表示して再生してください。こうすることで，代替ステップで必要なアクションが再生されます。

オブジェクトの識別メソッドの変更

TruClient によるオブジェクトの識別メソッドを変更するには，ステップのプロパティの［object］セクションでオブジェクトの識別メソッドを変更します。次のコンポーネントがあります。

- **［Automatic］：**TruClient の標準設定のオブジェクトの識別メソッド。［Automatic］のメソッドを使用すると，TruClient は高度な内部アルゴリズムを使用してオブジェクトを特定できます。このメソッドで再生時にオブジェクトを正常に検出できない場合，［Improve Object Identification］ボタンをクリックしてスクリプトをもう一度再生します。
- **XPath。**［Automatic］で識別できない場合（［Improve Identification］または［Related Objects］（下記参照）を使用しても識別できない場合），［XPath］の識別メソッドを使用します。このメソッドでは，DOM ツリーでオブジェクトを定義する XPath 式に基づいてオブジェクトを識別します。オブジェクトの推奨 XPath を選択するには，［**XPath**］編集ボックスの横にあるドロップダウン矢印をクリックします。推奨パスは手動で変更できます。VuGen によって生成された元のいずれかの式に戻すには，ドロップダウンのいずれかのオプションを再度選択します。

  たとえば，検索する用語に関係なく，最初の検索結果を選択する必要がある場合，XPath による識別が役立ちます。
- **JavaScript。**オブジェクトを返す JavaScript コード。たとえば，document.getElementById(「SearchButton」) は，DOM ID 属性が「SearchButton」の要素を返します。

  ［JavaScript］の識別メソッドを使用すると，返されるドキュメントを参照する JavaScript コードを作成したり，CSS セレクタやほかの標準機能を使用したりできます。

たとえば，サーバから返されるページに同じ「title」属性（検索結果）のリンクが複数ある場合，使用可能なリンクのいずれかをランダムにクリックするスクリプトが求められます。

この場合，オブジェクトの識別に［JavaScript］の識別メソッドを使用すると，次のようになります。

```
var my_results = document.querySelectorAll('a[title="SearchResult"]');
random(my_results);
```

スクリプトのタイミングの変更

タイミングや同期の問題でオブジェクトが見つからないことがあります。たとえば，スクリプトの再生が速すぎて，アプリケーションに存在していたオブジェクトを検索しているときにすでに別のページに進んでしまっていることがあります。オブジェクトが見つからない原因がタイミングや同期の問題にあることが疑われる場合，Wait ステップを挿入できます。詳細については，518 ページの「Wait ステップの挿入」を参照してください。

オブジェクト間の関連付け

［Improve Object Identification］機能や代替ステップでこの問題を解決できない場合，［Related Objects］オプションを使用します。

そのままではオブジェクトの識別が難しくなる場合，別のより安定したオブジェクトに基づいてオブジェクトにラベルを付けることができます。たとえば，動的ではなく，ターゲット・オブジェクトに「関連」していないオブジェクトを選択できます。関係は視覚的に定義され，ほかのオブジェクトからの距離（ピクセル）に応じてオブジェクトが関連付けられます。関係は各オブジェクトの識別メソッドごとに定義されます。特定のオブジェクトの識別メソッドで複数の関係が定義されている場合，ステップが成功するには両方の関係で同じオブジェクトを特定する必要があります。VuGen は，このオブジェクトを使用してターゲット・オブジェクトを特定できるようにします。この機能を使用するには，ステップを展開して［**Object**］>［**Related Objects**］を選択し，追加ボタン をクリックします。指示に従って関係を作成します。オブジェクトとその関連オブジェクトの両方を強調表示して，関係が機能していることを確認します。

ヒント：

- この機能は，リソースの消費量が大きくなる可能性があるため，ほかの識別メソッドで失敗した場合にのみ使用する。
- 検索領域を最小限に抑えてパフォーマンスを向上させる。
- ［Related Objects］はウィンドウ・サイズの影響を強く受ける。サイズを変更するとオブジェクトの位置や関係が変わる可能性があります。この点を考慮してください。
- 各識別メソッド（［Automatic］，［XPath］，［JavaScript］）には，それぞれの関連オブジェクトのセットがある。これらの関連オブジェクトはそれぞれの識別メソッドで共有されません。
- 複数の関係が存在する場合，すべて検出されないと識別に成功しない。

オブジェクトの置換

記録時に間違ったオブジェクトを選択した場合や，オブジェクトが永久的に変更された場合，ステップを置き換えずにそのオブジェクトを別のオブジェクトに置き換えることができます。これにより，元のステップへの変更（関係など）を削除してステップを効率的にリセットできます。ステップを展開して［**オブジェクト**］を選択し，［**Replace**］ボタン をクリックします。新しいオブジェクトを選択し，スクリプトを再生します。

［Replace Object］では，ステップで現在参照されているオブジェクトが正しくないことを TruClient に通知します。TruClient は，オブジェクトの現在の情報を削除し，選択したオブジェクトをゼロから学習します。

そのため，［Replace Object］オプションは，記録時に使用したオブジェクトが間違っている場合にのみ使用してください。

ループを挿入および変更する方法

ループでは，特定の条件を満たすか，指定した反復回数に達するまでスクリプトの選択した部分が繰り返されます。［**Toolbox**］の［**Functions**］セクションからループまたはループ修飾子を挿入できます。

➤ 524 ページの「For ループ」

➤ 524 ページの「Break ステートメント」

➤ 524 ページの「Continue ステートメント」

For ループ

For ループでは，終了条件を満たすか，コードが break ステートメントに達するまでループで囲まれたステップが実行されます。ループの引数には JavaScript 構文が使用されます。for ループを挿入するには，［**Toolbox**］>［**Functions**］>［**For Loop**］を選択します。

Break ステートメント

Break ステートメントは，現在のループがすぐに終了することを示します。たとえば，5 回反復する for ループの 2 回目の反復で break ステートメントに遭遇すると，残りの反復を実行せずにすぐにループが終了します。break ステートメントを挿入するには，［**Toolbox**］>［**Functions**］>［**Break**］を選択します。

Continue ステートメント

Continue ステートメントは，ループの現在の反復がすぐに終了することを示します。その後，ループ全体も終了するのかどうかを確認するためにループ条件が確認されます。たとえば，5 回反復する for ループの 2 回目の反復で continue ステートメントに遭遇すると，2 回目の反復がすぐに終了して 3 回目の反復が開始されます。continue ステートメントを挿入するには，［**Toolbox**］>［**Functions**］>［**Continue**］を選択します。

カスタム JavaScript および C コードを Ajax TruClient スクリプトに挿入する方法

このタスクでは，コードを Ajax TruClient スクリプトに挿入する方法について説明します。コードをステップ引数の一部として既存のステップに挿入したり，すべて外部コード（C または JavaScript）で構成されるステップを挿入したりできます。

このタスクでは，次の手順を実行します。

- 525 ページの「既存のステップへのコードの挿入」
- 525 ページの「すべてコードで構成されるステップの挿入」

1 既存のステップへのコードの挿入

大部分のステップの引数フィールドで JavaScript コードを既存のステップに挿入できます。これにより，さまざまなカスタマイズを実行できます。

2 すべてコードで構成されるステップの挿入

すべてコードで構成されるステップをスクリプトに挿入できます。これを行うには，［**Toolbox**］>［**Function**］を選択し，［**Eval Javascript**］，［**Eval C**］，［**Eval JS on Object**］アイコンを目的の場所にドラッグします。**Eval JS on Object** ステップでは，指定したオブジェクトがロードされてから JavaScript コードが実行されます。可能であれば，Eval C ではなく Eval JavaScript を使用することをお勧めします。ユーザはこのオブジェクトをステップ内の JavaScript コードの変数「object」として参照できます。

例：

次のコードでは，1 ～ 5 の間でランダムな数値を生成する amount という変数を作成しています。この変数をほかのステップの引数フィールドで使用できます。

```
var amount=Math.floor(Math.random()*5)+1;
```

リファレンス

TruClient のステップ引数

次の表に，ステップ引数をロールごとに分類して示します。必須引数の場合，ユーザ・インタフェースの引数名の左側に赤い星が付いています。すべての引数で JavaScript コードおよび LoadRunner 関数を引数値として使用できます。LoadRunner 関数のリストについては，531 ページの「TruClient 関数」を参照してください。

ロール	アクション	引数
要素	JavaScript の評価	Code : JavaScript コード
要素	マウス・アクション：マウス押下，マウス解放，マウス・オーバー，クリック，ダブル・クリック	➤ **Button :** クリックされたマウス・ボタン。 ➤ **X Coordinate :** オブジェクトの左上隅に対するアクションの相対的なオフセット位置。これは必ず正の数値になります。指定しない場合，オブジェクトの中心が標準設定になります。 ➤ **Y Coordinate :** オブジェクトの左上隅に対するアクションの相対的なオフセット位置。指定しない場合，オブジェクトの中心が標準設定になります。 ➤ **Ctrl Key :** Ctrl キーがアクション中に押されたかどうか。 ➤ **Alt Key :** Alt キーがアクション中に押されたかどうか。 ➤ **Shift Key :** Shift キーがアクション中に押されたかどうか。

ロール	アクション	引数
要素	ドラッグ	➤ **Button**：クリックされたマウス・ボタン。 ➤ **X Offset**：オブジェクトをドラッグした距離（X 軸方向のピクセル数）。正の数値は右方向へのドラッグを示します。 ➤ **Y Offset**：オブジェクトをドラッグした距離（Y 軸方向のピクセル数）。正の数値は下方向へのドラッグを示します。 ➤ **Path**：ユーザのドラッグ・パスを表す座標のリスト。この引数は変更しないでください。
要素	ターゲットへドラッグ	➤ **Target Object**：ステップのオブジェクトがこのターゲット・オブジェクトにドラッグされます。 ➤ **X Offset**：ターゲット・オブジェクトの左上からのオフセット（X 軸方向）。これは必ず正の数値になります。 ➤ **Y Offset**：ターゲット・オブジェクトの左上からのオフセット（Y 軸方向）。これは必ず正の数値になります。
要素	プロパティの取得	➤ **Property**：指定した変数に値が格納されるプロパティ。使用可能なプロパティのリストはオブジェクトのロールによって異なります。 次の標準設定のプロパティはすべてのオブジェクトで使用できます。 ➤ **Visible text**：項目の可視テキスト。DOM の textContent プロパティに対応します。 ➤ **All text**：項目の全テキスト。DOM の textContent プロパティに対応します。 ➤ **Inner HTML**：オブジェクトの内部 html マークアップ。DOM の innerHTML プロパティに対応します。 ➤ **Variable**：指定したプロパティ値の格納先の変数名。

ロール	アクション	引数
要素	検証	➤ **Value**：検証する文字列または数値。 ➤ **Property**：値が検証されるオブジェクト・プロパティ。検証できるプロパティのリストはオブジェクトのロールによって異なります。 次の標準設定のプロパティはすべてのオブジェクトで検証できます。 ➤ **Visible text**：アプリケーションに表示される項目。 ➤ **All text**：アプリケーションに存在するが必ずしも表示する必要のない項目。このカテゴリの項目は，DOM の textContent プロパティに含まれています。 ➤ **Inner HTML**：DOM の innerHTML プロパティに含まれている項目。 ➤ **Condition**：値とプロパティ引数間の関係。
要素	プロパティを待機	➤ **Value**：ステップが成功する前にステップが待機する指定のプロパティの値。 ➤ **Property**：スクリプトが値を待機するオブジェクト・プロパティ。待機できるプロパティのリストはオブジェクトのロールによって異なります。 次の標準設定のプロパティはすべてのオブジェクトで使用できます。 ➤ **Visible text**：アプリケーションに表示される項目。 ➤ **All text**：アプリケーションに存在するが必ずしも表示する必要のない項目。このカテゴリの項目は，DOM の textContent プロパティに含まれています。 ➤ **Inner HTML**：DOM の innerHTML プロパティに含まれている項目。 ➤ **Condition**：値とプロパティ引数間の関係。
フォーカス	キーを押す	➤ **Key name**：Enter または Space。

ロール	アクション	引数
テキスト・ボックス	タイプ	➤ **Value**：入力内容。 ➤ **Clear**：入力前にテキスト・ボックスをクリアします。標準設定は true です。 ➤ **Typing Interval**：キーストローク間の平均時間（ミリ秒）。
チェックボックス	設定	➤ **Checked**：チェックボックスをオン（T）またはオフ（F）に設定します。
リストボックス	選択	➤ **Text**：選択した文字列。 ➤ **Ordinal**：リストの選択項目の順序。text 引数も指定されている場合，この引数はリストボックスの指定したテキスト値のインスタンスを参照します。序数が 0 の場合，ランダムな値が生成されます。
ラジオグループ	選択	➤ **Text**：選択した文字列。 ➤ **Ordinal**：リストの選択項目の順序。text 引数も指定されている場合，この引数はリストボックスの指定したテキスト値のインスタンスを参照します。序数が 0 の場合，ランダムな値が生成されます。
ファイルボックス	設定	➤ **Path**：選択したパス。
スライダ	設定	➤ **Value**：スライダで設定された値。
日付選択	日付の設定	➤ **Day**：月の日付を表す 1 ～ 31 の整数。
ブラウザ	［アクティブ化］	➤ **Ordinal**：整数として定義されます。指定したブラウザ・ウィンドウを最前面に移動します。
ブラウザ	タブのアクティブ化	➤ **Ordinal**：アクティブにするタブ（整数）。
ブラウザ	タブを閉じる	➤ **Ordinal**：閉じるタブ（整数）。
ブラウザ	タブの追加	➤ **Location**：新しく開くタブの移動先の URL。
ブラウザ	移動	➤ **Location**：移動先の URL。
ブラウザ	戻る	➤ **Count**：戻るページ数。

ロール	アクション	引数
ブラウザ	進む	➤ **Count**：進むページ数。
ブラウザ	サイズ変更	➤ **Width**：新しい幅。これを空白のままにすると幅のサイズは変更されません。 ➤ **Height**：新しい高さ。これを空白のままにすると高さのサイズは変更されません。
ブラウザ	スクロール	➤ **X Coordinate**：新しい X 座標。これを空白のままにすると X 軸方向にスクロールされません。 ➤ **Y Coordinate**：新しい Y 座標。これを空白のままにすると Y 軸方向にスクロールされません。
ブラウザ	ダイアログ - 確認	➤ **Button**：OK または Cancel。
ブラウザ	ダイアログのプロンプト	➤ **Value**：入力する文字列。 ➤ **Button**：OK または Cancel。
ブラウザ	ダイアログ - 認証	➤ **Username**：入力するユーザ名。 ➤ **Password**：入力するパスワード。 ➤ **Domain**：入力するドメイン。 ➤ **Button**：OK または Cancel。

TruClient 関数

TruClient のステップ要素の値として次の関数を挿入できます。

メソッド	説明	引数	関連する関数
この行の引数はすべてのメソッドで使用できます。		➤ **Object**：アプリケーションで定義されたステップのオブジェクト。 ➤ **Window**：アプリケーションのグローバル・ウィンドウ・オブジェクトを示します。 ➤ **Document**：アプリケーションのグローバル・ドキュメント・オブジェクト。	
IO.createDir(path)	指定したディレクトリを作成します。必要に応じて，パスを完成させるために必要なすべてのフォルダを作成します。	**path**：ディレクトリの絶対パス。	
IO.delete(path)	指定したディレクトリまたはファイルを削除します。ディレクトリを指定すると，そのディレクトリ内にあるすべてのファイルとサブディレクトリが削除されます。	**path**：ディレクトリまたはファイルの絶対パス。	
IO.isExist(path)	指定したディレクトリまたはファイルが存在している場合は True を返し，存在していない場合は False を返します。マップされたドライブが未接続または未承認の場合は False を返します。	**path**：ディレクトリまたはファイルの絶対パス。	
IO.read(path, charset)	指定したファイルのすべてのデータを返します。指定した文字セットのデータを Unicode に変換します。	**path**：ファイルへの絶対パス。 **charset**：ファイルの文字セット（UTF 8 ではない場合）。	

メソッド	説明	引数	関連する関数
IO.write(path, string, append, charset)	指定したファイルに文字列を書き込みます。ファイルが存在していない場合は作成されます。	**absolute path**：ファイルへの絶対パス。 **string**：ファイルに書き込む文字列。 **append**：ブール値。 **[True]**：(標準設定) 文字列をファイルの最後に追加します。 **[False]**：文字列でファイルを上書きします。 **charset**：ファイルの文字セット（UTF 8 ではない場合)。	
LR.advanceParam(parameter)	指定したパラメータをファイルの次の値に進めます。	**parameter**：進めるパラメータの名前。パラメータのタイプはファイルまたは一意の数値である必要があります。	lr_advance_param
LR.setParam(name, value)	文字列をパラメータに保存します(パラメータが存在しない場合は作成します)。	**name**：値の保存先となるパラメータの名前。 **value**：値。	lr_save_string
LR.getParam(name)	指定したパラメータの値を返します。	**name**：パラメータ名。	lr_eval_string
LR.getLRAttr(name)	指定した mdrv コマンド・パラメータの値を返します。	**name**：コマンドライン・パラメータの名前。	lr_get_attrib_*
LR.evalC(funcname, filename)	指定したファイルに定義がある指定の関数を実行します。	**funcname**：関数名。 **filename**：関数が定義されているファイル。スクリプトの場所に相対的です。指定しない場合，標準設定（C-functions.c）が使用されます。	

メソッド	説明	引数	関連する関数
LR.log(text, level)	メッセージを記録します。	**text**：メッセージ。 **level**：次のいずれかになります。 ➤ "Error" ➤ "Warning" ➤ "Standard" ➤ "Extended" ➤ "Status" **例**：LR.log("text", "Error");	lr_debug_message
LR.decrypt(text)	復号化した後のテキストを返します。	**text**：暗号化されたテキスト。	lr_decrypt
LR.userDataPoint(name, value)	分析に使用するユーザ定義データ・ポイントを記録します。	**name**：データ・ポイントの名前。データポイント名の先頭を HTTP, NON_HTTP，RETRY, mic_，stream_，mms_ の文字列にしないでください。 **value**：数値。	lr_user_data_point
Utils.clearCookies()	仮想ユーザによって現在保存されている Cookie をすべて削除します。		web_cleanup_cookies
Utils.clearCache()	キャッシュ・シミュレータの内容を消去します。		web_cache_cleanup
Utils.getEnv(name)	指定した環境変数の値を返します。指定した変数が存在しない場合，空の文字列を返します。	**name**：環境変数の名前。	
Utils.import(path)	引数のコンテキストで指定した JavaScript ファイルを評価します。	**path**：JavaScript ファイルへの絶対パス。	

メソッド	説明	引数	関連する関数
Utils.isEnvExists(name)	指定した環境変数が存在している場合は **True** を返し，存在していない場合は **False** を返します。	**name**：環境変数の名前。	
Utils.setEnv(name, value)	指定した名前の環境変数を指定の値に設定します。変数に値が格納されている場合は上書きされます。	➤ **name**：環境変数の名前。 ➤ **value**：環境変数に設定する値。	

TruClient のプロパティ

次のプロパティは TruClient 関数の引数として使用できます。

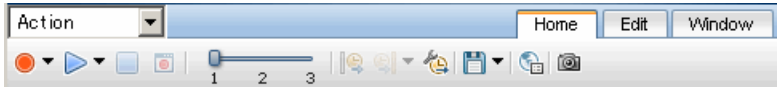
プロパティ	説明
LR.userId	MDRV コマンド・ラインに表示されるユーザ ID。MDRV は，すべてのプロトコルを実行するメイン・プロセスです。
LR.groupName	MDRV コマンド・ラインに表示されるグループ名。MDRV は，すべてのプロトコルを実行するメイン・プロセスです。プロセスが VuGen によって開始された場合，この値は 0 になります。
LR.scenarioId	MDRV コマンド・ラインに表示されるシナリオ ID。MDRV は，すべてのプロトコルを実行するメイン・プロセスです。シナリオ ID が存在するのは，MDRV が Controller によって開始された場合のみです。VuGen によって開始された場合，この値は 0 になります。
LR.outputDir	スクリプトの出力がすべて格納されているユーザ出力ディレクトリ。VuGen の場合，出力ディレクトリとスクリプト・ディレクトリは同じです。Controller の場合は異なります。返されるパスには，最後のフォルダ区切り記号が含まれます。
LR.scriptDir	ユーザのスクリプト・ディレクトリ。外部ファイルはスクリプト・ディレクトリに保存できます。スクリプトに外部ファイルを含めるには，外部ファイルのファイル名を LR.scriptDir に追加します。

Ajax TruClient のユーザ・インタフェース

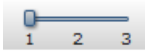
このセクションの内容

［Home］タブ

このタブでは，TruClient スクリプトの記録プロセスの基本的なフローを制御できます。



ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
	**[Record]**：スクリプトの記録を開始します。また，矢印を使用して，選択したステップの前に記録するのか，後に記録するのかを指定できます。
	**[Play]**：スクリプトが再生されます。また，矢印を使用して，選択したステップのみを再生するのか，スクリプトをステップごとに実行するのかを指定できます。段階的にスクリプトを実行すると，各ステップの後で再生が一時停止します。詳細については，517 ページの「段階的なスクリプトの実行」を参照してください。
	**[Stop]**：スクリプトの記録または再生を停止します。
	**[Toggle Breakpoint]**：選択したステップのブレークポイントを設定 / 解除します。
	**[Script Level]**：スクリプトで表示および再生されるスクリプト・レベルを変更します。詳細については，503 ページの「スクリプト・レベル」を参照してください。
	**[Start/End Transaction]**：トランザクションの開始点または終了点を挿入します。
	**[Transaction Editor]**：トランザクション・エディタが開き，新しいトランザクションの定義や既存のトランザクションの変更ができます。
	**[Save]**：スクリプトを保存します。

UI 要素	説明
	**[General Settings]**：[Ajax TruClient General Settings] ダイアログ・ボックスが開きます。詳細については，537 ページの「[Ajax TruClient General Settings] ダイアログ・ボックス」を参照してください。
	**[Snapshot View]**：スナップショット・ビューが右側の表示枠に表示されます。次の要素を右側の表示枠に追加します。 ➤ **[Snapshot Type]**：これらのボタンを使用して，別のモード時に取得されたスナップショットを表示できます。 ➤ **[View]**：次のビューから選択できます。 ➤ **[Single]**：1 つのステップのスナップショットが表示されます。 ➤ **[Compare]**：画面が分割して，異なるモードのスナップショットを比較できるようになります。表示するスナップショットを選択するには，各表示枠にある [Snapshot View] ボタンを使用します。表示枠間でスクロールを同期させるには，[**Synchronize Scrolling**] ボタン  をクリックします。[snapshot error] アイコン  は，ステップのスナップショットが最新ではないことを示します。 ➤ **[Thumbnails]**：[サムネイル] ビューでスナップショットが表示されます。 ➤ **[Previous/Next]**：前のステップまたは次のステップのスナップショットに移動します。スクリプトの対応するステップが強調表示されます。

[Ajax TruClient General Settings] ダイアログ・ボックス

このダイアログ・ボックスでは，対話式モードでスクリプトを作成するときに Firefox ブラウザに影響する多くのオプションを設定できます。

利用方法	[**Ajax TruClient General Settings**] ボタン 
重要情報	[Ajax TruClient General Settings] ダイアログ・ボックスは，Vugen ウィンドウまたは TruClient ブラウザ（対話式モード）からアクセスできます。

［Browser Settings］タブ

このタブでは，対話式モードで実行するスクリプトの TruClient ブラウザを設定できます。

［Browser Settings］タブで使用できる設定は，［**実行環境の設定**］>［**Load Mode Browser Settings**］ノードで使用できる設定と同じです。使用可能なオプションの詳細については，446 ページの「［一般］>［Load Mode Browser Settings］ノード」を参照してください。

［Browser Settings］タブで変更した設定は，対話式モードにのみ影響します。対話式モードでスクリプトを保存すると，［Browser Settings］タブで変更した設定が［Load Mode Browser Settings］に適用されます。

［Encryption］，［Extensions］，［Bookmarks］タブ

［Encryption］，［Extensions］，［Bookmarks］タブで使用できる設定は，標準の Firefox ブラウザで使用できる設定と同じです。

- **［Encryption］タブ**：暗号化に関する Firefox のドキュメントについては，http://support.mozilla.com/en-US/kb/Options%20window%20-%20Advanced%20panel?as=u#w_encryption-tab を参照してください。
- **［Extensions］タブ**：拡張機能に関する Firefox のドキュメントについては，http://support.mozilla.com/en-US/kb/Using%20extensions%20with%20Firefox#w_installing-from-the-add-ons-window を参照してください。

  標準の Firefox の［Extensions］ダイアログに表示される［**プラグイン**］および［**テーマ**］ボタンは，TruClient の Firefox の［Extensions］タブでは無効になっています。
- **［Bookmarks］タブ**：ブックマークに関する Firefox のドキュメントについては，http://support.mozilla.com/en-US/kb/Smart%20Bookmarks%20folders を参照してください。

http://support.mozilla.com/en-US/kb/Options%20window%20-%20Advanced%20panel?as=u#w_encryption-tab を参照 http://support.mozilla.com/en-US/kb/Using%20extensions%20with%20Firefox#w_installing-from-the-add-ons-window を参照 http://support.mozilla.com/en-US/kb/Smart%20Bookmarks%20folders を参照してください。TruClient を実行するたびに，新しいプロファイルで Firefox が開きます。あるセッションで作成したブックマークを後のセッションで使用することはできません。［Bookmarks］タブの［Add Bookmark］ボタンをクリックします。

［Edit］タブ

このタブでは，TruClient スクリプトのステップやデータの切り取り，コピー，貼り付けができます。

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
	選択したデータまたはステップを切り取ります。
	選択したステップまたはデータをコピーします。
	選択したステップの前に貼り付けます。
	選択したステップの後に貼り付けます。
	選択したステップに貼り付けます。
	選択したステップを削除します。

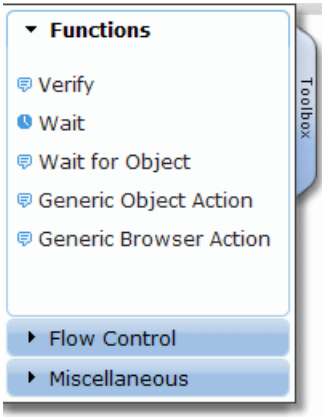
UI 要素	説明
	［Find］ダイアログ・ボックスが開き，ステップのスクリプトを名前や数値で検索できます。
	指定したステップに移動します。

［Window］タブ

このタブでは，同一スクリプトの複数の Firefox ウィンドウを制御できます。フォーカスするアプリケーションを含むウィンドウを選択します。これは，ステップのオブジェクトを強調表示する場合など，スクリプト開発のデバッグ・フェーズで必要になります。

ツールボックス

ツールボックスでは，TruClient スクリプトにステップを追加できます。ツールボックスはドラッグして上や下に移動できます。



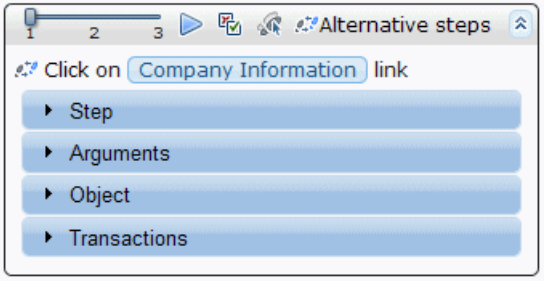
ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
Functions	➤ **[Verify]**：アプリケーションのオブジェクトが存在しているかどうかを検証します。 ➤ **[Wait]**：次のステップを続行する前に指定の秒数待機します。 ➤ **[Wait for Object]**：次のステップを続行する前にオブジェクトがロードされるまで待機します。 ➤ **[Generic Object/Browser Action]**：挿入して手動で設定できる空のステップ。
Flow Control	➤ **[For Loop]**：ループ内のステップを指定の回数繰り返す論理構造。 ➤ **[If Block]**：条件を満たした場合にブロック内のステップを実行する論理構造。 ➤ **[Add else]**：**If** ブロックに **else** セクションを追加するには，[**Add else**] リンクをクリックします。条件を満たさない場合，**else** セクションのステップが実行されます。 ➤ **[Remove else]** **If** ブロックから **else** セクションを削除します。**注**：**else** セクションにステップが含まれている場合，[**Remove else**] をクリックするとステップも削除されます。ステップを保存するには，スクリプトの本文にコピーして貼り付けてください。 ➤ **[Break]**：現在または残りの反復を実行せずにすぐにループを終了します。 ➤ **[Continue]**：ループの現在の反復をすぐに終了します。スクリプトの次の反復から続行されます。 ➤ **[Catch Error]**：直前のステップのエラーを捕捉して，catch error ステップの内容を実行します。詳細については，516 ページの「Catch Error ステップの挿入」を参照してください。 ➤ **[Exit]**：指定した設定に応じて反復またはスクリプト全体を終了します。

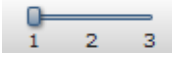
UI 要素	説明
Miscellaneous	➤ **[Evaluate JavaScript]**：ステップに含まれている JavaScript コードを実行します。 ➤ **[Evaluate JS on Object]**：指定したオブジェクトをアプリケーションにロードしてから，ステップに含まれている JavaScript コードを実行します。 ➤ **[Evaluate C]**：ステップに含まれている C コードを実行します。 ➤ **[Comment]**：スクリプトにコメントを記載できる空のステップ。

TruClient のステップの構造

TruClient のステップは多数のセクションで構成されています。各セクションに含まれるセクションや要素はステップのタイプによって異なります。



ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
	**[Script Level] セレクタ**：ステップのスクリプト・レベルを表示および変更できます。詳細については，503 ページの「スクリプト・レベル」を参照してください。
	**再生**：このステップのみを再生します。

UI 要素	説明
	**ステップを無効 / 有効にする**：無効にしたステップは再生されません。この機能により，スクリプトからステップを削除せずに一時的に除外できます。
	**オプション ステップ**：ステップをオプションとしてマークすると，ステップでオブジェクトを検出できない場合でもエラーを返さずにスクリプトが続行されます。
	**Alternative Steps**：このアイコンは，別の方法で再定義できるステップを表しています。ステップを再定義するには，このアイコンをクリックして目的のステップ定義を選択し，［Back］をクリックします。詳細については，505 ページの「代替ステップ」を参照してください。
Step	➤［**Action**］：ステップを定義するアクション。オブジェクトのあるステップの場合，ステップのロールによってこのリストが決まります。 ➤［**Object Timeout**］：この秒数の前にオブジェクトが表示されない場合，ステップでエラーが返されます。 ➤［**Step Timeout**］：この秒数までに終了イベントに達しない場合，ステップでエラーが返されます。エラーが発生したときのスクリプトの動作方法は［実行環境設定］ダイアログ・ボックスで定義できます。 ➤［**End Event**］：終了イベントが発生した場合，このステップでスクリプトの後続のステップを継続して実行することを許可できます。
Arguments	ステップの引数を格納します。これらの引数は，ステップのアクションやロールによって異なります。ステップ引数のリストについては，526 ページの「TruClient のステップ引数」を参照してください。
Roles	TruClient が認識するオブジェクトの機能。この情報は，読み取り専用で，記録時にどのようにオブジェクトが使用されるかに応じて動的に更新されます。使用可能なステップのアクションのリストは，これらのロールごとに定義されています。
Name	オブジェクトの論理名。これは再生には影響しません。また，わかりやすい名前に変更することもできます。

UI 要素	説明
ID Method	オブジェクトを識別するメソッド。 ➤ **[Automatic]**：TruClient の標準設定のオブジェクトの識別メソッド。このメソッドで再生時にオブジェクトを正常に検出できない場合，［Improve Object Identification］ボタンをクリックし，アプリケーションで適切なオブジェクトを選択しなおして，スクリプトをもう一度再生します。 ➤ **XPath。**DOM ツリーでオブジェクトを定義する XPath 式に基づいてオブジェクトを識別します。このオプションを選択すると，画面の次の編集ボックスに［**XPath**］というラベルが表示され，オブジェクトを定義する XPath を選択できるようになります。詳細については，下記を参照してください。 ➤ **JavaScript。**オブジェクトを返す JavaScript コード。このオプションを選択すると，画面の次の編集ボックスに［JavaScript］というラベルが表示され，オブジェクトを定義する JavaScript を定義できるようになります。詳細については，下記を参照してください。
XPath/JavaScript	
XPath	VuGen によって，オブジェクトに応じていくつかの推奨 XPath がいくつか生成されます（生成されない場合もあります）。オブジェクトの推奨 XPath を選択するには，［XPath］編集ボックスの横にあるドロップダウン矢印をクリックします。推奨 XPath は手動で変更できます。VuGen によって生成された元のいずれかの式に戻すには，ドロップダウンのいずれかのオプションを再度選択します。 ［**Regenerate expression**］ボタンをクリックしてオブジェクトを選択することもできます。VuGen により，選択したオブジェクトに基づいて新しい一連の推奨が作成されます。

UI 要素	説明
JavaScript	TruClient でオブジェクトの推奨 XPath を生成できる場合，その XPath は［JavaScipt］フィールドの **evalXPath** 関数の引数として入力されます。**evalXPath** 関数は，引数内の XPath によって定義されるオブジェクトの配列を返します。異なるオブジェクトのリストが返されるように引数内の推奨 XPath を変更できます。また，別の JavaScript を入力することもできます。 たとえば，document.getElementById("SearchButton") は，DOM ID 属性が「SearchButton」の要素を返します。 TruClient には，引数として提供されている配列からランダムな項目を返す **random** 関数もあります。次に例を示します。 random(document.getElementsByTagName("a")) **注：**evalXPath 関数と random 関数はオブジェクトの識別メソッドとしてのみ使用できます。Evaluate JavaScript code ステップでは認識されません。
Transactions	トランザクションの作成，変更，表示ができます。詳細については，514 ページの「トランザクションを挿入する」を参照してください。

［Transaction Editor］ダイアログ・ボックス

このダイアログ・ボックスでは，Ajax TruClient 仮想ユーザ・スクリプトのトランザクションを管理できます。

UI 要素	説明
	新しいトランザクションの追加または選択したトランザクションの削除を行います。
General	トランザクションの名前を編集できます。
Start Point	トランザクションの開始を示すステップやステップ内のイベント。
End Point	トランザクションの終了を示すステップやステップ内のイベント。

オブジェクトの識別に関する問題のトラブルシューティング

ステップ間でのオブジェクトの共有方法

アプリケーションの同じオブジェクトが複数のステップで使用されている場合，Ajax TruClient では各ステップでオブジェクトが共有されます。

つまり，1 つのステップでオブジェクト名を変更するとほかのステップでも変更され，1 つのステップで［Improve Identification］を使用するとほかのステップにも影響します。

これらのステップのいずれかで［Replace］機能を使用して別のオブジェクトが選択されると，そのオブジェクトは共有されなくなります。

［XPath］および［JavaScript］の識別メソッドをサポートするための［Related Objects］の使い方

［XPath］および［JavaScript］の識別メソッドの場合，使用する式によっては複数の要素が返されます。

たとえば，XPath の値が //button の場合，問題の Web ページに複数のボタン要素があると複数のオブジェクトが返されます。

1 つのオブジェクトが返されるようにするには，関連オブジェクトを追加して識別を絞り込みます。

オブジェクトが見つからないエラーによって対話式再生が失敗する

オブジェクトが見つからなかったことを示すエラーで再生が停止した場合，次の操作を行います。

- 失敗したステップを選択し，［Highlight］を押します。適切なオブジェクトが強調表示されない場合，オブジェクトの識別に問題があります。この場合，［Improve Identification］を使用してオブジェクトの識別を改善する必要があります。
- オブジェクトが強調表示される場合，オブジェクトが表示される前にステップに達した可能性があります。問題のステップの前に Wait ステップまたは Wait for Object ステップを追加します。

問題を解決するために代替ステップを選択する必要がある場合もあります。たとえば，特定の値に基づいてテキストが変わるドロップダウン・リストのオプションをクリックする場合を考えます。テキストに基づいてクリックすると，ステップが失敗する可能性があります。リスト内のオプションの序数値に基づいてリストの項目を選択する代替ステップを使用すると，このクリックはテキストに関係なく成功します。

［Highlight］でオブジェクトを特定できるがオブジェクトが見つからないことが原因で対話式再生が失敗する

［Highlight］オプションで適切なオブジェクトを検出できるのに再生が失敗する場合，ペースの問題がある可能性があります。

オブジェクトのロード時間が少し長くなると，ステップが先に実行されてしまいます。そのため，デバッグ時には失敗したステップで［Highlight］オプションを使用するとオブジェクトが検出されるのに，実行時にはステップでオブジェクトが特定できないということが起こります。

この場合，オブジェクトをロードする時間を確保するためにスクリプトを「遅らせる」ことをお勧めします。これを行うには，次のいずれかの方法を使用します。

- 失敗したステップの［Object Timeout］を変更する。これは，ステップのプロパティの［Step］セクションで行うことができます。
- 失敗したステップの前に Wait ステップまたは Wait for Object ステップを追加する。

再生でリストから項目を選択できない

この現象の一般的な原因の 1 つとして，リストの項目名が動的であることが挙げられます。

たとえば，これまでに入力したテキストに基づいて一連の都市をリストに表示できる場合（オートコンプリート）などが該当します。

テキストのタイプに基づいてリストは常に変わります。

この問題を解決する方法は 2 つあります。

- 実際の項目のテキストではなく序数識別子を使用してリストから項目を選択する代替ステップを使用する。
- テキストの一部のみが動的である場合は，正規表現を使用してテキストの部分一致に基づいて必要な項目を特定する。

TruClient スクリプトのトラブルシューティング

特定のテキストを確認する方法（分岐を含む）

ツールボックスの［Functions］セクションから Verify ステップを追加する方法もあります。このステップでは，オブジェクトや検索テキストなどのさまざまな検証設定を選択できます。

検証の成功または失敗に基づいて特定のアクションを実行する場合，ツールボックスの［Flow Control］セクションから Catch Error ステップを追加します。

こうすることで，検証が失敗した場合でもステップを続行できます。また，Catch Error グループ内に検証が失敗した場合に実行する一連のステップを定義できます。

また，よりプログラム的なアプローチで検証することもできます。JavaScript を使用すれば，DOM にアクセスして目的のプロパティを検証できます。その後，この検証に基づいて条件付き Break ステップまたは Exit ステップ（ツールボックスの［Flow Control］セクションにあります）を追加できます。

また，IF ステートメントで直接必要なテキストを確認することもできます。IF ステートメントの「condition」引数は単純な JavaScript コードです。テスト対象アプリケーションのグローバル・ウィンドウ・オブジェクトにアクセスする JavaScript コードを使用できます。これを行うには，ウィンドウを参照します。

その後，現在のページ内にテキストが存在しているかどうかを手動で検証できます。たとえば，単一フレーム・アプリケーションの場合，次のように作成できます。

```
window.document.body.textContent.indexOf("Off") == -1
```

ここで，"Off" は検索テキストで，-1 はテキストが見つからなかったことを示します。

このコードはアプリケーション固有です。

アプリケーションに精通していれば，（特定の要素を取得して）コードを最適化できます。

大文字と小文字を区別しないで特定のテキストを確認する方法

標準設定では，Verify ステップの大文字と小文字は区別されます。たとえば，「test」が見つかっても「Test」の検索は失敗となります。

Verify ステップの大文字と小文字が区別されないようにするには，次の操作を行います。

➤ Verify ステップで，Condition 引数を「正規表現」に設定します。

➤ 大文字 / 小文字に関係なく文字列「test」がテキストに含まれているかどうかを確認するには，次のようにします。

```
RegExp("test", "i")
```

リストからランダムにオプションを選択する方法

Ordinal 引数を 0 に設定します。TruClient によって，自動的にリストからオプションがランダムに選択されます。

たとえば，入力したテキストに基づいて都市のリストを表示するオートコンプリート・リストがあるとします。現在 2 番目のオプションを選択していて，ステップは Select option #2 from City autocomplete となっています。

ステップのプロパティの［Arguments］セクションを開いて，Ordinal 引数を 0 に変更します。ステップは Select a random option from City autocomplete のようになります。

この方法は，入力したテキストがパラメータであるためにリストに存在する値やその数を事前に知ることが難しい場合に非常に重要です。

スクリプトで外部関数を使用する方法

スクリプトの一部である JS-Function.js ファイルや C-functions.c ファイルに JavaScript 関数や C 関数を追加すると，VuGen の左側のナビゲーション表示枠に表示されます。

JavaScript 関数は，すべての引数とパラメータで JavaScript がサポートされているため Ajax TruClient スクリプトから直接呼び出すことができます。また，ツールボックスで Evaluate JavaScript ステップを追加して JavaScript 関数を呼び出すこともできます。

C 関数を呼び出すには，ツールボックスで Evaluate C ステップを追加します。

一部のイベントやアクションが記録したスクリプトに表示されない

次のいずれかの方法でこの問題を解決します。

- Ajax TruClient では，アプリケーションのすべてのイベントが記録されます。探しているイベントは，表示されているスクリプト・レベルとは異なるスクリプト・レベルに存在している可能性があります。
  - 表示されているレベルのステップの番号が連続していなければ，ほかのスクリプト・レベルに別のステップが存在していることがわかります。
  - ツールバーのスライダを使用して現在のスクリプト・レベルを設定します。

    スライダの値を変更して，欠落しているイベントまたはステップがほかのレベルにないか探します。
  - 必要なイベントが見つかったら，そのレベルを変更してレベル 1 に含めることができます。
  - スクリプト・レベルをレベル 1 に戻して，もう一度再生します。
  - スクリプト・レベルの概念の詳細については，Ajax TruClient のドキュメントを参照してください。
- スクリプトにステップを手動で追加できます。
  - ツールボックスで［Generic Object Action］を選択し，ステップをカスタマイズして必要なアクションを実行します。

スライダまたはマップをドラッグしても正常に再生されない

ドラッグが機能していないために（スライダ・オプションの設定やマップのドラッグなど），適切な場所にコントロールが移動していない場合，次の操作を行います。

- いずれかの代替ステップ使用して，効果があったかどうかを確認する
- 目的を達成するまで値を手動で設定する（関連する各方向にドラッグする正確なピクセル数など）
- ［Drag to］機能を使用する（ステップのプロパティの［Step］セクションで Drag ステップのアクションを変更する）。こうすることで，オブジェクトを別のオブジェクトの相対位置にドラッグできます。

オブジェクトのポーリング

次のステップを含むループを作成します。

- オブジェクトでアクションを実行するステップ。
- Continue ステップを含む Catch Error セクション。
- Break ステップ。

Catch Error セクションにより，オブジェクトが検出されてステップが成功するまでループが続行されるようになります。

WHILE ループの作成方法

「For」ループには，Init，Condition，Increment の 3 つの引数があります。

「While」は基本的に Condition 引数のみがある「For」ループです。

while ループを作成するには，（ショートカット・メニューまたはツールボックスを使用して）「For」ループを追加し，Init および Increment 引数を削除して Condition を指定します。

ロード・モードのトラブルシューティング

スクリプトをできるだけ安定させる方法

動的な属性を持つオブジェクトは，Web 2.0 アプリケーションの特徴の 1 つです。

たとえば，オブジェクトの ID は，アプリケーションにアクセスするたびにさまざまな値に変わる可能性があります。

これらの変更が発生したとしてもスクリプトを安定して実行できるようにするには，次のステップを実行することをお勧めします。

- スクリプトを正常に再生できたら，Firefox の対話式スクリプト作成用サイドバーでさらに数回実行します。
- 数回の再生のいずれかで失敗した場合，さまざまなメソッド（[Improve Identification]，[Related Objects]，[Alternative Steps] など）でオブジェクトの識別を改善します。

- 特定のイベント（IIS のリセットなど）に基づいてアプリケーション・オブジェクトが変わる場合，スクリプトの安定性を検証するために，これらのイベントが各再生で発生することを確認します。
- VuGen の［実行］オプションを使用してロード・モードでもう一度スクリプトを実行し，負荷テストに組み込む準備が整っているかどうかを検証します。

対話形式の開発では再生できるが負荷をかけると失敗するスクリプトの修正方法

実際に負荷テストでスクリプトを使用する前に VuGen の［実行］オプションを使用してロード・モードでスクリプトをテストすることを強くお勧めします。［実行］機能では，負荷テストと同じようにスクリプトが再生されるため，スクリプトを負荷テストに含める前に検証できます。

スクリプトが VuGen の［実行］や実際の負荷テストで失敗するが，対話式再生では正常に機能している場合，次のステップを実行してください。

- 再生ログを参照して何のエラーなのか調べます。
- 「オブジェクトが見つからない」というエラーの場合，ペースの問題である可能性があります。ロード・モードの再生は対話式再生よりも速いため，アプリケーションへのオブジェクトのロードが間に合わないことがあります。
- この問題は次のいずれかの方法で解決できます。
  - ステップ間の時間（ミリ秒）を指定できる［ステップ間の間隔］実行環境設定を使用する。この設定はスクリプトのすべてのステップで使用されます。
  - 失敗したステップの前に Wait ステップまたは Wait for Object ステップを追加する。これにより，再生が遅くなります。
- 一部のアプリケーションでは，ブラウザ・サイズの影響を強く受けることがあります。負荷がかかった状態でスクリプトを実行する場合に特定の定義済みウィンドウ・サイズが使用されると，オブジェクトの場所に関するエラーが発生する可能性があります。

  この問題を解決するには，Resize ステップをスクリプトに追加するか，対話式モード以外の再生で使用する幅と高さの初期値を設定します。これを行うには，［実行環境の設定］の［その他の設定］にある［非反復ウィンドウ サイズ］設定を使用します。

ロード・モードでスクリプトをデバッグする方法

ログを参照してエラーを詳しく調べます。

エラーが発生したときにアプリケーションの状態を表示するには，［実行環境の設定］に移動して［エラー時のスナップショット］を選択し，スクリプトを再生します。エラーが発生するとスナップショットが生成され，［実行論理］（［**結果**］>［**ロード**］>［**[アクション名]**］>［**[反復回数]**］）に基づいて結果ディレクトリに保存されます。

スクリプト内で LR.log 関数を使用してログにメッセージを追加することもできます。

この関数の詳細については，LoadRunner Ajax TruClient のドキュメントを参照してください。

TruClient の制限

このセクションでは，Ajax TruClient スクリプトの制限について説明します。

一般

- TruClient では，Flash または Silverlight アプリケーションを記録できない。
- パラメータ値はスクリプト全体を再生する場合にのみ使用され，1 つのステップを再生する場合には使用されない。

ビジネス・プロセスの文字確認機能

多くの Web サイトでは，特別な文字確認フィールドを使用しています。このフィールドには実際のユーザがフォームを入力しているかどうかを検証するためにユーザが入力する必要があるテキストが表示されます。

これは，クローラやスパイダなどによってサイトが使用されて，貴重なシステム・リソースが占領されないようにするためのものです。

これらのフィールドは，特に LoadRunner などの自動ツールをブロックするように設計されています。

ビジネス・プロセスを自動的に完了するには，負荷をかける Web サイトでこの関数を無効にする必要があります。

14

AMF プロトコル

本章の内容

概念

概念

AMF プロトコルの概要

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。

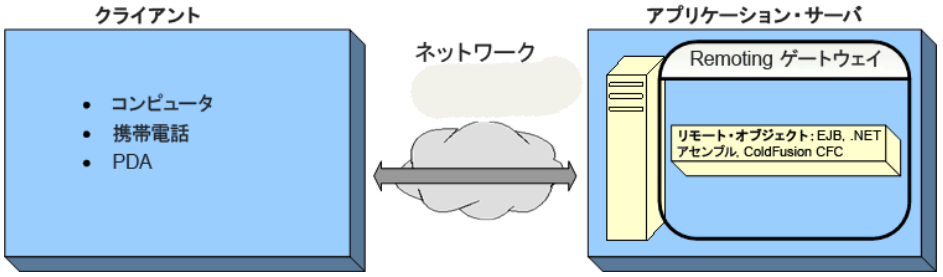
HTTP は，すべてのインターネット・ブラウザおよびサーバでサポートされます。したがって，インターネット経由で作業する場合，HTTP は，クライアント・アプリケーションとサーバ間の通信における好ましい方法であるといえます。

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 の用語

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 のスクリプト例

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，*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。

AMF 記録モードの設定

AMF プロトコルおよび Web プロトコルを使用して Flash Remoting セッションからスクリプトを生成する方法を VuGen に指定できます。次のオプションがあります。

- AMF および Web
- AMF のみ
- Web のみ

標準設定では，VuGen はスクリプトに AMF 呼び出しのみを生成します。これらのオプションを設定するには，［**ツール**］>［**記録オプション**］>［**一般**］>［**プロトコル**］ノードを選択します。詳細については，353 ページの「[一般]の［プロトコル］ノード」を参照してください。

注：前述のいずれかのオプションで記録した場合，後でオプションを変更してスクリプトを再生成し，ほかのプロトコルを含める，または除外できます。

AMF および Web

AMF プロトコルと Web プロトコルの両方を有効にした場合，VuGen はビジネス・プロセス全体の関数を生成します。AMF データに遭遇すると，適切な AMF 関数を生成します。

次のコードの抜粋では，VuGen は Web 関数（**web_url**）と AMF 関数（**amf_call**，**amf_define_envelope_header_set**）の両方を生成しています。

```
web_url("flash",
    "URL=http://testlab:8200/flash/","Resource=0",
    …
    "Snapshot=t1.inf",
    EXTRARES,
    "Url=movies/XMLExample.swf", "Referer=", ENDITEM,
    "Url=movies/JavaBeanExample.swf", "Referer=", ENDITEM,
    LAST);

web_link("Sample JavaBean Movie Source",
    "Text=Sample JavaBean Movie Source",
    "Snapshot=t2.inf",
    EXTRARES,
    "Url=XMLExample.swf", "Referer=", ENDITEM,
    "Url=JavaBeanExample.swf", "Referer=", ENDITEM,
    LAST);

amf_set_version("0");

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);
```

AMF のみ

Flash Remoting をエミュレートするのに AMF 呼び出しのみを必要とする場合は，Web 呼び出しを無効にして，AMF 呼び出しのみを生成できます。

次の例は，AMF プロトコルを有効にし，Web プロトコルを無効にして，前述のセッションを記録したものです。

```
Action()
{
amf_set_version("0");

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);
…
```

なお，この記録は，完全なビジネス・プロセスを示すものではありません。AMF を使用する Flash Remoting 呼び出しの例を示しているに過ぎません。

Web のみ

Web のみのオプションは，代替として Web HTTP 技術を使用します。AMF 呼び出しは生成されません。代わりに，Flash Remoting 情報を持つ **web_custom_request** 関数が生成されます。

次の例は，前述のセグメントを AMF なしで再生成したものです。

```
web_url("flash",
    "URL=http://testlab:8200/flash/",
    "Resource=0",
    "RecContentType=text/html",
    "Referer=",
    "Snapshot=t1.inf",
    "Mode=HTML",
    EXTRARES,
    "Url=movies/XMLExample.swf", "Referer=", ENDITEM,
    "Url=movies/JavaBeanExample.swf", "Referer=", ENDITEM,
    LAST);

web_link("Sample JavaBean Movie Source",
    "Text=Sample JavaBean Movie Source",
    "Snapshot=t2.inf",
    EXTRARES,
    "Url=XMLExample.swf", "Referer=", ENDITEM,
    "Url=JavaBeanExample.swf", "Referer=", ENDITEM,
    LAST);

web_custom_request("gateway",
    "URL=http://testlab:8200/flashservices/gateway",
    "Method=POST",
    "Resource=0",
    "RecContentType=application/x-amf",
    "Referer=",
    "Snapshot=t3.inf",
    "Mode=HTML",
    "EncType=application/x-amf",

"BodyBinary=¥¥x00¥¥x00¥¥x00¥¥x01¥¥x00¥¥x10amf_server_debug¥¥x01¥¥x00¥¥x00¥¥x0
0`¥¥x03¥¥x00¥ncoldfusion¥¥x01¥¥x01¥¥x00¥namfheaders¥¥x01¥¥x00¥¥x00¥¥x03amf¥¥x0
1¥¥x00¥¥x00¥¥x0Bhttpheaders¥¥x01¥¥x00¥¥x00¥trecordset¥¥x01¥¥x01¥¥x00¥¥x05error¥¥
x01¥¥x01¥¥x00¥¥x05trace¥¥x01¥¥x01¥¥x00¥¥x07m_debug¥¥x01¥¥x01¥¥x00¥¥x00¥t¥¥x0
0¥¥x01¥¥x00/flashgateway.samples.FlashJavaBean.testDocument¥¥x00¥¥x02/
1¥¥x00¥¥x00¥¥x004¥n¥¥x00¥¥x00¥¥x00¥¥x01¥¥x0F¥¥x00¥¥x00¥¥x00*<TEST
message=¥"test¥"><INSIDETEST /></TEST>",
        LAST);
```

15

Citrix プロトコル

本章の内容

概念

タスク

リファレンス

概念

Citrix プロトコルの概要

Citrix 仮想ユーザ・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，仮想ユーザ・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 の仮想ユーザ用のマルチ・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。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，*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。

Ctrix 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。

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，40 ページの「スクリプトのセクション」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**］>［**すべてのプログラム**］>［**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，571 ページの「自動同期化」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。

Ctrix 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，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。

初期化クォータを設定する

接続中に複数の仮想ユーザによって過負荷になるのを防ぐため，仮想ユーザ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仮想ユーザ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** 関数の前には適用します。

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，119 ページの「仮想ユーザ・スクリプトの再生成方法」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 マシンごとの仮想ユーザ数の増加

Citrix 仮想ユーザを稼動する 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，実行できる仮想ユーザ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仮想ユーザの数を増やすには，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。

同期化

同期化とは，ウィンドウやオブジェクトが使用可能になるまで待ってからアクションを実行することを指します。これは，Citrx スクリプトを記録するときに必要です。たとえば，スクリプトのあるステップでウィンドウを開き，次のステップでそのウィンドウ内でアクションを実行する場合，2 番目のステップはこのウィンドウが開くまで実装できないからです。VuGen では，スクリプトが不正確に再生されないようにするために，ウィンドウまたはオブジェクトが使用可能になるまで待機してスクリプトを同期化する関数が自動的に生成されます。また，同期化関数は手動で追加することもできます。

自動同期化の詳細については，571 ページの「自動同期化」を参照してください。

同期化ポイントを手動で追加する方法の詳細については，583 ページの「Citrix スクリプトを手動で同期化する方法」を参照してください。

自動同期化

記録中，仮想ユーザ・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

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，仮想ユーザ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，仮想ユーザ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［Citrix エージェント入力をコード生成に使用する］オプションが有効になっている場合，VuGen では自動的に **sync_on_obj_info** ステップが生成されます。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，333 ページの「［Citrix］の［コード生成］ノード」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，仮想ユーザ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，仮想ユーザは，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（575 ページの「Citrix Presentation Server エージェントの概要」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。詳細については，333 ページの「［Citrix］の［コード生成］ノード」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。

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，*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。

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

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

ctrx_win_exist 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必要があります。

これらの関数の詳細については，*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。

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。仮想ユーザは，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：［**Sync on Bitmap**］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 10 章，「記録オプション」を参照してください。

Citrix Presentation Server エージェントの概要

Citrix Presentation Server エージェント（Citrix 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。以降の各セクションでは，これらの機能強化について説明します。

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。Citrix エージェントのインストール方法の詳細については，584 ページの「Citrix エージェントをインストールおよびアンインストールする方法」を参照してください。

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);
```

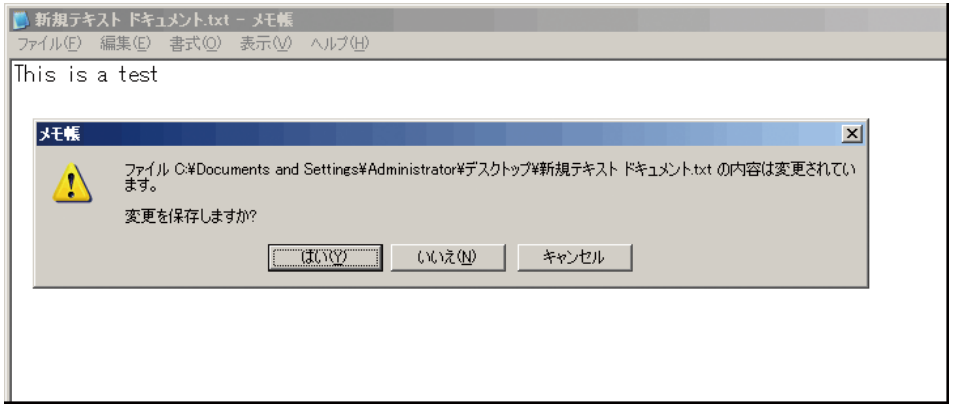
上の例で，**ctrx_obj_mouse_click** 関数の 1 番目の引数には，ウィンドウのタイトルとクラスのテキスト，つまりメモ帳が格納されています。エージェントは個々のオブジェクトに関する追加情報を提供しますが，仮想ユーザはウィンドウ名とオブジェクトの座標によってのみ，オブジェクトにアクセスします。

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［**はい**］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。
- **テキスト同期を取得**と**テキスト取得**：詳細については，579 ページの「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。

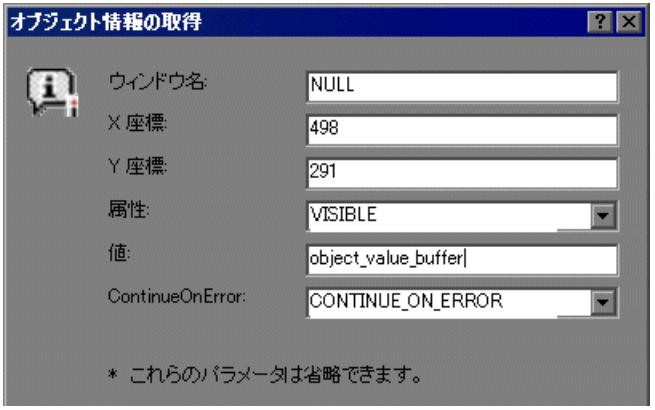
3 ビットマップをマークするには，ビットマップを右クリックして［**ビットマップ同期を挿入**］を選択します。

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。

タスクの詳細については，583 ページの「Citrix スクリプトを手動で同期化する方法」を参照してください。

Xenapp 5.0 のトラブルシューティング

Citrix および Web マルチプロトコル・スクリプトを使用して Xenapp 5.0 で記録する場合，適切に記録するためには手動で変更を行う必要があります。

Xenapp 5.0 を使用して記録するには，次の手順を実行します。

1 XenApp の設定を変更する

a Web インタフェースから XenApp サーバに接続します。

b ［**プリファレンス**］>［**Session Settings**］を選択し，［**ウィンドウのサイズ**］を［**No Preference**］に設定します。

2 相関ルールを更新する

3 記録オプションを変更する

a ［記録オプション］ダイアログ・ボックスの［**一般**］>［**記録**］ノードを開きます。

b ［**HTML 詳細設定**］を選択して［HTML 詳細設定］ダイアログ・ボックスを開きます。

c ［**明示的な URL のみを含むスクリプト**］を選択します。

タスク

Citrix クライアントおよびサーバを設定する方法

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。次の手順では，このプロセスについて説明します。

➤ 581 ページの「Citrix クライアントを設定する」

➤ 582 ページの「MetaFrame サーバをインストールする」

➤ 582 ページの「MetaFrame サーバを設定する」

Citrix クライアントを設定する

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。

MetaFrame サーバをインストールする

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。

MetaFrame サーバを設定する

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。［**すべてのプログラム**］>［**Citrix**］>［**管理ツール**］>［**Citrix 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。［Advanced Connection Settings］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。

Citrix スクリプトを手動で同期化する方法

自動同期化に加えて，記録中および記録後，手動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，手動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。

記録中に手動で同期化する

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**および**テキストで同期**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・ウィンドウ内の領域またはテキストを指定できます。

- **ビットマップで同期**ステップを挿入するには，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，対象の領域を囲むように矩形領域を指定します。

- **テキストで同期**ステップを挿入するには（Citrix エージェントが必要），ツールバーの［**テキスト同期を挿入**］ボタンをクリックし，対象のテキストを囲むように矩形領域を指定します。

記録後に手動で同期化する

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。

- **［ビットマップ取得時に同期化］**：ビットマップが表示されるまで待機します。
- **［オブジェクト情報取得時に同期化］**：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）。
- **［テキストで同期化］**：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）。

Citrix エージェントをインストールおよびアンインストールする方法

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する必要があります。

Citrix Presentation Server エージェントをインストールする

1 エージェントをアップグレードするには，新しいバージョンをインストールする前に以前のバージョンをアンインストールしてください。

2 ソフトウェアをインストールするのにサーバの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。

3 Windows 2003 で稼動しているマシンにエージェントをインストールするのに RDP（リモート・デスクトップ接続）を使用している場合は，インストールを開始する前に目的のマシンで次のコマンドを実行してください。

```
Change user /install
```

4 製品のインストール・ディスクの **Additional Component¥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。

5 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。

Citrix Presentation Server エージェントをアンインストールする

1 ソフトウェアを削除するのにサーバの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。

リファレンス

Citrix 関数

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。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。どの関数も記録セッションの後，仮想ユーザ・スクリプトに手動で編集または追加できます。

ctrx 関数の詳細については，*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。

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。

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。

［ビットマップ同期に失敗しました］ダイアログ・ボックス

このダイアログ・ボックスでは，ビットマップ同期に失敗した場合に実行する処理を指定できます。

利用方法	ビットマップの同期化ステップで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に不一致があった場合，再生中にこのダイアログ・ボックスが自動的に開きます。

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
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，仮想ユーザは失敗しません。
記録時のスナップショット	記録時のスナップショットのビュー。
再生時のスナップショット	再生時のスナップショットのビュー。
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，標準設定で実行が停止されます。または，特定の関数に［エラーでも処理を継続する］を指定できます。詳細については，576 ページの「エラー時も処理を継続する」を参照してください。

トラブルシューティングと制限事項

このセクションでは，Citrix プロトコルのトラブルシューティングと制約事項について説明します。

Citrix エージェントの効果とメモリ要件

エージェントのインストールされた Citrix 仮想ユーザを実行すると，各仮想ユーザは **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仮想ユーザの数がわずかに減少します（約 7%)。

Citrix 仮想ユーザごとの記憶容量（各仮想ユーザがそれぞれ **ctrxagent.exe** プロセスを実行した場合）は約 4.35 MB です。25 仮想ユーザを実行するには，110 MB のメモリが必要です。

データ実行防止機能（DEP）と Citrix エージェントのパフォーマンス

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，VuGen の記録が停止される場合があります。

これらの環境で記録中に異常な動作があった場合は，DEP 設定を変更してください。

Windows の DEP 設定を変更するには，次の手順で行います。

1 ［**スタート**］>［**コントロール パネル**］>［**システム**］を開きます。

2 ［詳細設定］タブの［**パフォーマンス**］の［**設定**］をクリックします。

3 ［パフォーマンス オプション］の［データ実行防止］タブで，［**重要なサービスについてのみ有効にする**］という最初のオプションを選択します。

このオプションを変更できない場合は，［**追加**］をクリックします。IEXPLORE.EXE など，クライアント・プログラムを参照します。

これらのオプションのいずれも使用できない場合は，DEP を完全に無効にしてみてください。

a ［コントロール パネル］で［**システム**］セクションの［**詳細設定**］タブをクリックします。

b ［**起動と回復**］で［**編集**］をクリックします。

c 「**NoExecute=OptOut**」を「**NoExecute=AlwaysOff**」に置き換えます。

4 ［**OK**］をクリックして設定を保存します。

5 コンピュータを再起動します。

デバッグに関するヒント

次のセクションでは，Citrix スクリプトのデバッグに関するヒントを示します。

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動で追加することもできますが，571 ページの「自動同期化」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仮想ユーザを設定できます。［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 **hash_value.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。

仮想ユーザの表示

シナリオ実行時に仮想ユーザを表示するには，仮想ユーザ・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，問題のある仮想ユーザ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，-lr_citrix_vuser_view **<** 仮想ユーザ ID**>** のように，コマンド・ラインの最後に仮想ユーザの ID を追加して，個々の仮想ユーザの詳細を表示します。

複数の仮想ユーザを開くには，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。仮想ユーザ 2，6 は表示されず，ID が 7 より大きい仮想ユーザも表示されません。

Citrix および Web マルチプロトコル・スクリプトを使用して Xenapp 5.0 で記録する場合，適切に記録するためには手動で変更を行う必要があります。

XenApp 5.0 を使用して記録するには，次の手順を実行します。

1 XenApp の設定を変更する

a Internet Explorer から XenApp サーバに接続します。

b 記録時に使用するのと同じアカウントでログインします。

c ［**プリファレンス**］>［**Session Settings**］を選択し，［**ウィンドウのサイズ**］を［**No Preference**］に設定します。

2 相関ルールを更新する

a ［記録オプション］ダイアログ・ボックスの［**HTTP 相関**］ノードを開きます。

b ［**Citrix_XenApp**］ルールが選択されていることを確認します。

3 記録オプションを変更する

a ［記録オプション］ダイアログ・ボックスの［**一般**］>［**記録**］ノードを開きます。

b ［**HTML 詳細設定**］を選択して［HTML 詳細設定］ダイアログ・ボックスを開きます。

c ［**明示的な URL のみを含むスクリプト**］を選択します。

16

Click and Script プロトコル

本章の内容

概念

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

［**高い**］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，608 ページの「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［**記録オプション**］>［**GUI プロパティ**］>［**詳細**］ノードを有効にできます。

 再生に関するヒント

次のセクションでは，Click and Script スクリプトを再生する場合のヒントを示します。

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の順序は変更しないようにします。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**］を選択し，［**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［**はい**］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，*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。

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，*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。
- 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，*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。

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，*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。

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 Software のサポートに問い合わせてください。

Web（Click and Script）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，*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，「*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。

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。仮想ユーザが，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，「*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。

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。仮想ユーザは，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。仮想ユーザは，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**。

スクリプト例（Web Click and Script）

通常，Click and Script 仮想ユーザ・スクリプトには，ビジネス・プロセスを構成する複数のアクションが含まれています。GUI レベルで生成された記録済みの関数を見れば，記録されたセッションの間のユーザの正確なアクションを知ることができます。

たとえば，標準的な記録では，第 1 段階にサインインのプロセスが含まれています。ブラウザでサインイン・ページが開いたら，ユーザはユーザ名とパスワードを入力し，［**Sign In**］をクリックしてサインインします。

Web（Click and Script）仮想ユーザでは，エディット・フィールドに入力されたデータを表す **web_edit_field** 関数が生成されます。次の例では，ユーザは［userid］フィールドにテキストを入力し，暗号化される［pwd］フィールドにパスワードを入力しています。

```
vuser_init() {
    web_browser("WebTours",
        DESCRIPTION,
        ACTION,
        "Navigate=http://localhost:1080/WebTours/",
        LAST);

    web_edit_field("username",
        "Snapshot=t2.inf",
        DESCRIPTION,
        "Type=text",
        "Name=username",
        "FrameName=navbar",
        ACTION,
        "SetValue=jojo",
        LAST);

    web_edit_field("password",
        "Snapshot=t3.inf",
        DESCRIPTION,
        "Type=password",
        "Name=password",
        "FrameName=navbar",
        ACTION,
        "SetEncryptedValue=440315c7c093c20e",
        LAST);…
```

リファレンス

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);
```

トラブルシューティングと制限事項

このセクションでは，Click and Script プロトコルのトラブルシューティングと制約事項について説明します。一部の項目は，特定の Click and Script プロトコルにのみ適用されます。

記録に関する問題と制限事項

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（597 ページの「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 イベントの定義**］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**］を選択し，［**GUI プロパティ**］>［**Web イベントの定義**］ノードを選択します。

➤［**ユーザ定義設定**］をクリックして［**Web Objects**］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。これらの設定は，［**記録オプション**］>［**GUI プロパティ**］>［**Web イベントの定義**］ノードにあります。

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。

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**］を選択し，［**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［**はい**］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**］を選択し，［**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［**はい**］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，*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。

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，*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。
- 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，*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。

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。

再生の失敗

特定のステップで再生が失敗する場合は，ステップの説明を確認します。VuGen では，シングル・スペースがダブル・スペースとして読み取られる場合があります。文字列に不適切なダブル・スペースがないことを確認してください。

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**］を選択し，［**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**］を選択し，［**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。

17

COM プロトコル

本章の内容

概念

概念

COM プロトコルの概要

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。仮想ユーザ・スクリプトの作成に使用するプログラミング言語を，C または Visual Basic として設定できます。

VuGen では，COM 仮想ユーザ・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 技術の概要

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，仮想ユーザを使ってリモート・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 スクリプトの構造

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。

```
Irc_ <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}"
```

仮想ユーザ・スクリプトのステートメント

COM 仮想ユーザ・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。

COM サンプル・スクリプト

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。これは，COM スクリプトが行う基本的な処理に分けられています。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，仮想ユーザ・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，*オンライン関数リファレンス*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*オンライン関数リファレンス*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，*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。

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 **ActiveX Data Object (ADO)** 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 Objects** 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。

18

データベース・プロトコル

本章の内容

概念

リファレンス

概念

データベース・プロトコルの概要

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，データベース仮想ユーザを使って，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。データベース仮想ユーザでは，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

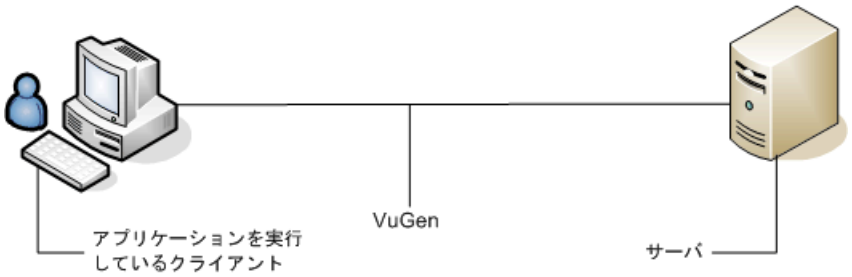
まず，利用可能な Load Generator に数百の DB 仮想ユーザを振り分けます。各仮想ユーザはサーバ API を使ってデータベースにアクセスします。これにより，多数のユーザによる負荷をかけた状態でのサーバのパフォーマンスを測定できます。

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）仮想ユーザ・スクリプトといいます。データベース仮想ユーザ・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。仮想ユーザ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。仮想ユーザ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。

VuGen では，CtLib，DbLib，Informix，Oracle，ODBC，DB2-CLI というデータベース・タイプがサポートされています。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。

VuGen データベースの記録技術

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 仮想ユーザ・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仮想ユーザと同様に，データベース仮想ユーザも，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，各データベース仮想ユーザではサーバ API 関数を直接呼び出すスクリプトが実行されます。

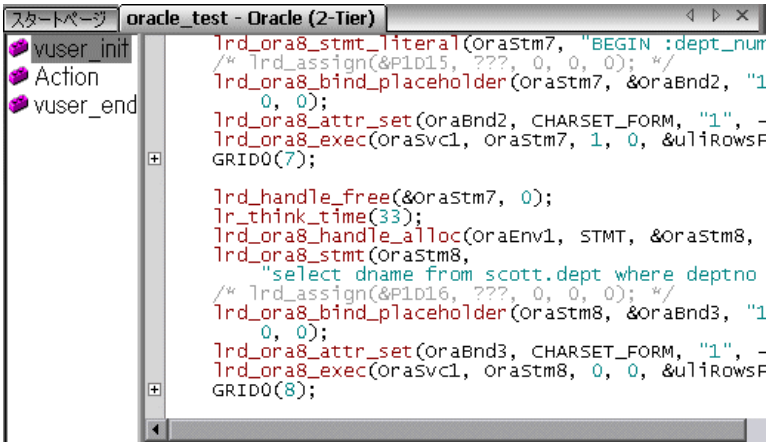
データベース仮想ユーザ・スクリプトは，Windows 環境で VuGen を使って作成します。しかし，作成したスクリプトは，Windows および UNIX のどちらの環境でも仮想ユーザに割り当てることができます。

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 仮想ユーザ・スクリプトを作成できます。UNIX での DB 仮想ユーザ・スクリプトのプログラミングの詳細については，第 46 章，「UNIX でのスクリプトの作成と実行」。を参照してください。

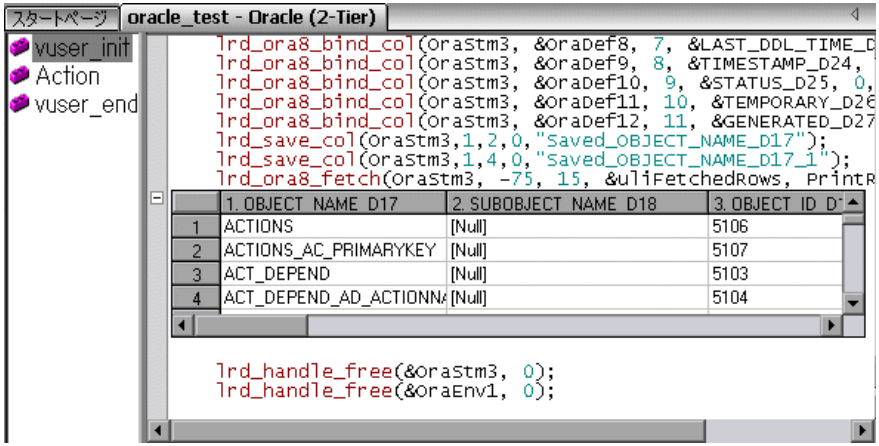
データベース・グリッド

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。



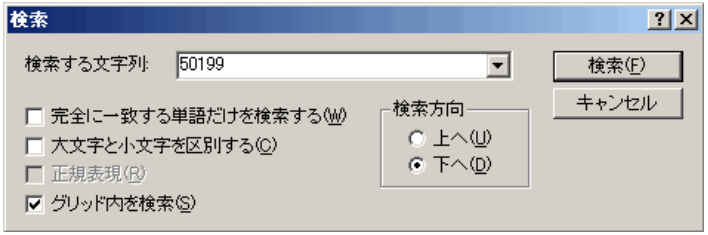
データ値が非常に長い場合は，その一部のみグリッドに表示されます。この切り捨ては，表示されているグリッドでのみ生じ，データには影響を与えません。

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。

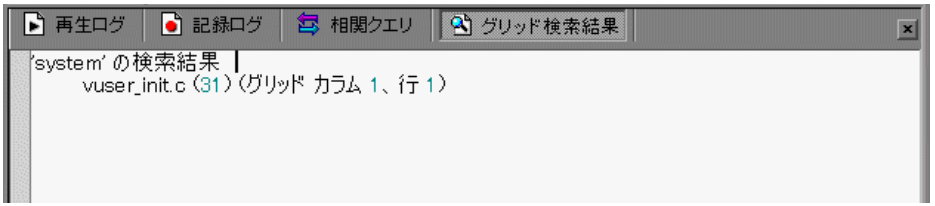
```
[general]
max_line_at_grid=200
```

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。



Oracle アプリケーション

Oracle Applications は，2-tier（「ファット」クライアント）パッケージのアプリケーションで，35 ものさまざまなモジュール（Oracle Human Resources, Oracle Financials など）で構成されています。

Oracle Applications 用の仮想ユーザの記録と再生のために，いくつか知っておくべきことがあります。

- 一般的なスクリプトには，何千ものイベント，バインド，およびアサインが含まれています。
- 一般的なスクリプトには，各ユーザ・セッションに多くの db 接続が含まれています。
- スクリプトには，ほとんどの場合相関したクエリが必要です。
- Oracle Applications のクライアントは 16 ビットのみです（Oracle Developer 2000 で開発されたもの）。つまり，デバッグに際して Oracle 32 ビット・クライアントがなければ，VuGen の Force 16-bit オプションを使う必要があります。

新しいウィンドウが作成されると，アプリケーションは，表示用にファイル・システムから .xpf ファイルを取得します。VuGen はクライアント / サーバ・レベルで記録を行うため，現在はこれを考慮に入れていません。したがって，パフォーマンスの測定はかなり不正確になります。多くの場合，パフォーマンスの問題はクライアントとファイル・サーバの間のボトルネックに関係するものだからです。現在，この問題の解決に向け，検討中です。

エラー処理

データベース仮想ユーザ・スクリプトの実行時に，データベース仮想ユーザ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。仮想ユーザ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。次に説明するように，この動作を別の方法で適用できます。

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，仮想ユーザ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，エラーが生じると仮想ユーザ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，仮想ユーザ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，仮想ユーザ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 仮想ユーザに指示します。

```
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，仮想ユーザ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 仮想ユーザ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，仮想ユーザ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），仮想ユーザ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，仮想ユーザ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。

データベース・アプリケーションのデバッグ

以降のヒントは，データベース・アプリケーション（Oracle，ODBC，および Ctlib）に適用されます。

デバッグ情報の生成

注：本項で説明する情報の大部分は，VuGen のユーザ・インタフェースを使って表示するように設定できます。

VuGen には，インスペクタ「エンジン」が含まれています。*¥WINDOWS_DIR¥vugen.ini* を次のように編集して，VuGen レコーダが「インスペクタ」出力を作成するようにできます。

```
[LogMode]
EnableAscii=ASCII_LOG_ON
```

このオプションが有効になっている場合，VuGen は記録終了時に Data ディレクトリに **vuser.asc** ファイルを作成します。このオプションは，デバッグの目的に限って使うべきものです。出力ファイルが非常に大きくなり（数 MB），マシンのパフォーマンスとディスク領域に深刻な影響を及ぼす可能性があるからです。

ODBC ベースのアプリケーションの場合は，［ODBC データ ソース アドミニストレータ］（Windows の［コントロール パネル］にあります）で同様の追跡出力を得るように設定できます。ODBC オプションを開いて，［Trace ODBC calls］を ON にします。同様に，ODBC Developer Kit には呼び出しの追跡を行うスパイ・ユーティリティが用意されています。

詳細なデバッグ情報を有効にするには，¥WINDOWS_DIR¥vugen.ini ファイルに次のセクションを追加します。

```
[INSPECTOR]
TRACE_LEVEL=3
TRACE_FILENAME=c:¥tmp¥sqltrace.txt
```

sqltrace.txt ファイルには，記録中に行われたフック呼び出しについての有用な内部情報が含まれます。trace_level は 1 から 3 まであり，3 は最も詳細なデバッグ・レベルを表します。VuGen バージョン 5.02 以上では，ユーザ・インタフェースから追跡レベルを設定できます。

コンパイラ情報の検証

コード生成，前処理，コンパイルの各段階についての情報を表示して，エラーの原因を特定できます。

コード生成情報

Data ディレクトリ内の **vuser.log** ファイルを見ます。このファイルには，コード生成段階のログが含まれており，lrd 記録（すなわちすべてのデータベース・プロトコル）が終わるたびに，自動的に作成されます。

次にログ・ファイルの例を示します。

```
lrd_init: OK
lrd_option: OK
lrd_option: OK
lrd_option: OK
Code generation successful
lrd_option: OK
lrd_end: OK
```

いずれかのメッセージが OK（成功）ではない場合は，コード生成中に問題が生じています。

前処理とコンパイルの情報

実行中，VuGen は前処理とコンパイル処理の両方についての情報を表示します。

リファレンス

トラブルシューティングと制限事項

このセクションでは，データベース・プロトコルのトラブルシューティングと制約事項について説明します。

すべてのデータベース・プロトコルのトラブルシューティング

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・コードの評価

仮想ユーザが 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，*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。

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);
```

2-tier データベースのスクリプト作成のヒント

本項では 2-tier データベース・スクリプトのための解決策を示します。

疑問 1：アプリケーション自体は同じ値を扱えるのに，データ駆動のスクリプトで失敗する原因は？

回答：データ値の後に続く空白が原因である可能性があります。GUI に直接入力するデータ値ではおそらく切り詰められているとしても，データ・ファイルから手作業で除去する必要があります。タブ区切りファイルでは，後続の空白が見えにくく，問題が発見しづらくなります。一般に，カンマ区切りファイルをお勧めします。Excel を使って問題がないかどうか確かめることができます。

疑問 2：2 回目の反復でカーソル状態が無効という SQL エラーが発生する原因は？

回答：**lrd_close_cursor** 関数が生成されていないか，スクリプトの *action* セクションではなく，*end* セクションに生成されている可能性があります。スクリプトを反復できるようにするには，カーソル・クローズ関数を追加するか，*end* セクションから移動する必要があります。

反復のたびに新しいカーソルを開くのは資源効率の点から好ましくありません。したがって，最初の反復の *actions* セクションで 1 回だけカーソルを開くことをお勧めします。それから［Iteration Number］タイプを使用して，反復番号を文字列として含む新しいパラメータを追加します。このパラメータに *IterationNum* パラメータという名前を付けます。次に *actions* セクション内で新しいカーソルを開く呼び出し

```
lrd_open_cursor(&Csr1, Con1, 0);
```

を次のように置換します。

```
if (!strcmp(lr_eval_string("<IterationNum>"), "1"))
   lrd_open_cursor(&Csr1, Con1, 0);
```

疑問 3：*vdf.h* ファイルのデータ宣言が原因で VuGen で生成したコードをコンパイルできない場合の修正方法は？

回答：問題は，おそらく，VuGen でサポートされていない SQL のデータ型です。Microsoft SQL では多くの場合，*vdf.h* ファイル内の未定義エラー・メッセージを「DT_SZ」（ヌル終端文字列）に置き換えれば回避できます。これは実際のデータ型ではありませんが，VuGen でそのスクリプトを正常にコンパイルできます。カスタマ・サポートに問題を通知し，元のスクリプトをお送りください。

疑問 4：LRD Error 2048 の意味は？

回答：VuGen が失敗します。記録時の割り当てよりも長い変数をバインドしようとしているためです。*vdf.h* ファイルで変数定義を拡大して，データベースから長い文字列を受け取れるようにします。このファイルで一意の数値識別子を検索します。その定義と長さがわかります。長さは構成要素の内 3 つ目の要素です。この長さを増やせばスクリプトを正常に再生できるようになります。

たとえばスクリプト内に次の行があるものとします。

```
lrd_assign(&_2_D354, "<ROW_ID>", 0, 0, 0);
```

vdf.h ファイルで *_2_D354* を検索すると次のようになっています。

```
static LRD_VAR_DESC _2_D354 = {
            LRD_VAR_DESC_EYECAT, 1, 10, LRD_BYTYPE_ODBC,
            {0, 0, 0}, DT_SZ, 0, 0, 15, 12};
```

これを次のように変更します。

```
static LRD_VAR_DESC _2_D354 = {
            LRD_VAR_DESC_EYECAT, 1, 12, LRD_BYTYPE_ODBC,
            {0.0, 0, 0}, DT_SZ, 0, 0, 15, 12};
```

LRD_VAR_DESC の定義全体は *lrd.h* ファイルにあります。これを見つけるには「typedef struct LRD_VAR_DESC」で検索します。

疑問 5：ODBC および Oracle の使用で UPDATE，INSERT，DELETE によって影響を受けた行数を取得する方法は？

回答：**lrd** 関数を使用して情報を取得します。ODBC には **lrd_row_count** 関数を使用します。構文は次のとおりです。

```
int rowcount;
   :
   :
   :
lrd_row_count(Csr33, &rowcount, 0);
```

lrd_row_count 関数は関連するステートメント実行の直後に使用します。

Oracle には **lrd_exec** の 4 番目の引数を使用します。

```
lrd_exec(Csr19, 1, 0, &rowcount, 0, 0);
```

Oracle の OCI 8 を使用している場合は，**lrd_ora8_exec** の 5 番目の引数を使用します。

```
lrd_ora8_exec(OraSvc1, OraStm3, 1, 0, &uliRowsProcessed, 0, 0, 0, 0, 0);
```

疑問 6：キーの重複割り当て違反を防ぐ方法は？

回答：挿入を実行するときに重複キー違反が生じることがあります。問題を識別するために 2 つの記録を比較して，主キーを見つけることができます。このステートメントまたはそれ以前に出てきた UPDATE または INSERT ステートメントで相関クエリが使われていたかどうかチェックします。データ辞書を使って一意制約に違反しているカラムを見つけることができます。

Oracle では，一意制約違反が生じると，次のメッセージが表示されます。

```
ORA-00001: unique constraint (SCOTT.PK_EMP) violated
```

この例では，SCOTT は関連する一意インデックスの所有者で，PK_EMP がそのインデックス名です。SQL*Plus を使用してデータ辞書のクエリを行い，カラムを見つけます。このクエリのパターンは次のようになります。

```
select column_name from all_ind_columns where index_name = '<IndexName> and
index_owner = '<IndexOwner>';
select column_name from all_ind_columns where index_name = 'PK_EMP' and
index_owner = 'SCOTT';
```

データベースに挿入された値が新しいため，以前のクエリでは見つかっていないかもしれませんが，以前のクエリよりも 1 つ多い戻り値として，以前のクエリの結果と関係していることがあります。

Microsoft SQL サーバでは次のいずれかのメッセージが表示されます。

```
Cannot insert duplicate key row in object 'newtab' with unique index 'IX_newtab'.

Violation of UNIQUE KEY constraint 'IX_Mark_Table'. Cannot insert duplicate key in
object 'Mark_Table'.

Violation of PRIMARY KEY constraint 'PK_NewTab'. Cannot insert duplicate key in
object 'NewTab'.
```

Query Analyzer を使用して，どのカラムがキーまたはインデックスで使用されているのかを検索します。このクエリのパターンは次のようになります。

```
select C.name
    from sysindexes A, sysindexkeys B, syscolumns C
    where C.colid = B.colid and C.id = B.id and
    A.id = B.id and A.indid = B.indid
    and A.name = '<IndexName>' and A.id = object_id('<TableName>')
select C.name
    from sysindexes A, sysindexkeys B, syscolumns C
    where C.colid = B.colid and C.id = B.id and
    A.id = B.id and A.indid = B.indid
    and A.name = 'IX_newtab' and A.id = object_id('newtab')
```

DB2 では次のメッセージが表示されます。

```
SQL0803N One or more values in the INSERT statement, UPDATE statement, or
foreign key update caused by a DELETE statement are not valid because they would
produce duplicate rows for a table with a primary key, unique constraint, or unique
index. SQLSTATE=23505
```

まだ問題が続くようであれば，記録時，再生時のスクリプトの両方で更新および挿入で変更した行番号を確認します。UPDATE では，WHERE 句の条件式が間違っているために再生時に行を変更できないことがよくあります。これは直接エラーになりませんが，テーブルが正確に更新されず，後続の SELECT がクエリの相関時に間違った値を選択する原因になります。

また，マルチ・ユーザの再生中に問題がないことも確認します。特定のインスタンスでは，1 ユーザだけしか UPDATE を実行できません。この現象は Siebel で発生します。その場合には，手作業でループを書いて問題を回避する必要があります。

疑問 7：スクリプト再生後にデータベースが変更されているはずなのに変更されていない。

回答：ユーザ・アプリケーションの UI からアプリケーションからアクセスできる現在のデータを調べ，それが更新された値かどうかを確認してください。値が更新されていない場合，それが変更されていないことを判定する必要があります。アプリケーションの記録中に UPDATE ステートメントが 1 つ以上の行を変更したために，再生時には変化がなかったということも考えられます。

次の項目を確認します。

- **ステートメントの検証**：UPDATE ステートメント内の WHERE 句条件式が正しいことを確認します。
- **相関の確認**：アプリケーションを 2 回記録して，それぞれの記録の UPDATE ステートメントを比較し，必要な相関が行われているか確認します。

- **行の総数の確認**：UPDATE の後で変更された行数を確認します。Oracle ではこの情報は **lrd_exec** の 4 番目のパラメータに格納されています。ODBC では，**lrd_row_count** を使用して行数を調べます。スクリプトに更新された行数を出力するコードを追加できます。出力値が 0 ならば，UPDATE はデータベースの変更に失敗したことがわかります。
- **SET 句のチェック**：UPDATE ステートメントの SET 句の条件式を確認します。必要な値がすべて相関されており，ハードコードされていないかどうかチェックします。UPDATE の 2 つの記録を比べることによって判断できます。

これが問題なのは，UPDATE が単独の仮想ユーザの再生時には動作するのに，複数の仮想ユーザでは動作しない場合です。ある仮想ユーザの UPDATE がほかの仮想ユーザのものと干渉していることが考えられます。各仮想ユーザに同じ値で更新を行うように指定する場合を除いて，各仮想ユーザをパラメータ化してそれぞれの仮想ユーザが UPDATE 時に異なる値を使用するようにします。この場合，再試行論理を追加して UPDATE を再び試みます。

疑問 8：Oracle Application を使用して記録されたステートメントの再生時に一意カラム名エラーを防ぐ方法は？次に例を示します。

```
lrd_stmt(Csr9,"SELECT UOM_CODE, UOM_CODE, DESCRIPTION FROM "
    "MTL_UNITS_OF_MEASURE "
    "WHERE NVL(DISABLE_DATE, SYSDATE + 1) > "
    "SYSDATE ORDER BY UOM_CODE", -1, 1, 1, 0);
```

この例では次のエラー・メッセージが発行されます。

```
"lrdo.c/fjParse: "oparse" ERROR return-code=960, oerhms=ORA-00960:
ambiguous column naming in select list"
```

回答：一意でないカラムの少なくとも 1 つに別名を追加して，この新しい一意の名前にマッピングするようにステートメントを変更します。次に例を示します。

```
lrd_stmt(Csr9,"SELECT UOM_CODE,UOM_CODE second, DESCRIPTION
FROM"
    "MTL_UNITS_OF_MEASURE "
    "WHERE NVL(DISABLE_DATE, SYSDATE + 1) > "
    "SYSDATE ORDER BY UOM_CODE", -1, 1, 1, 0);
```

Oracle 2 層仮想ユーザのトラブルシューティング

本項では，Oracle 仮想ユーザを扱っているときに生じるいくつかの一般的な問題と，その解決策を示します。

ORA-20001 と ORA-06512

lrd_stmt に fnd_signon.audit_responsibility(...) という PL/SQL ブロックが含まれている場合，再生中にエラー ORA-20001 と ORA-06512 が発生します。

このステートメントが再生中に失敗するのは，新しい接続をするたびに一意のサインオン番号が割り当てられるためです。

解決策

この問題を解決するには，サインオン番号を扱う新しい相関ツールを使用する必要があります。サインオン番号は，ステートメント内で 2 番目に割り当てられる値です。

相関候補の値を検索した後，失敗したステートメントの 2 番目の lrd_assign_bind() の値を強調表示します。「相関クエリ」ウィンドウでは，値が実際に記録されたステートメントと同じ順番で表示されない場合があります。

置換する値が含まれるグリッドは，nd_signon.audit_user(...) という PL/SQL ブロックが含まれる lrd_stmt の後に現れます。

注：サインオン番号は接続ごとに一意なので，記録する新しい接続のそれぞれについて相関を行う必要があります。

解決策の例

次のステートメントは，2 番目の値「1498224」がそれぞれの新しい接続に一意のサインオン番号なので，再生で失敗します。

```
lrd_stmt(Csr6, "begin fnd_signon.audit_responsibility(:s,:l,:f,:a,:r,:t,:p)"
       "; end;", -1, 1, 1, 0);
   lrd_assign_bind(Csr6, "s", "D", &s_D216, 0, 0, 0);
   lrd_assign_bind(Csr6, "l", "1498224", &l_D217, 0, 0, 0);
   lrd_assign_bind(Csr6, "f", "1", &f_D218, 0, 0, 0);
   lrd_assign_bind(Csr6, "a", "810", &a_D219, 0, 0, 0);
   lrd_assign_bind(Csr6, "r", "20675", &r_D220, 0, 0, 0);
   lrd_assign_bind(Csr6, "t", "Windows PC", &t_D221, 0, 0, 0);
   lrd_assign_bind(Csr6, "p", "", &p_D222, 0, 0, 0);
   lrd_exec(Csr6, 1, 0, 0, 0, 0);
```

このサインオン番号は，lrd_stmt で「fnd_signon.audit_user」を手がかりに見つけることができます。最初のプレースホルダの値「a」はそのままにしておきます。「a」への入力値はいつも「0」ですが，出力は要求された値です。

変更後のコード：

```
 lrd_stmt(Csr4, "begin fnd_signon.audit_user(:a,:l,:u,:t,:n,:p,:s); end;", -1, 1, 1, 0);
   lrd_assign_bind(Csr4, "a", "0", &a_D46, 0, 0, 0);
   lrd_assign_bind(Csr4, "l", "D", &l_D47, 0, 0, 0);
   lrd_assign_bind(Csr4, "u", "1001", &u_D48, 0, 0, 0);
   lrd_assign_bind(Csr4, "t", "Windows PC", &t_D49, 0, 0, 0);
   lrd_assign_bind(Csr4, "n", "OraUser", &n_D50, 0, 0, 0);
   lrd_assign_bind(Csr4, "p", "", &p_D51, 0, 0, 0);
   lrd_assign_bind(Csr4, "s", "14157", &s_D52, 0, 0, 0);
   lrd_exec(Csr4, 1, 0, 0, 0, 0);
lrd_save_value(&a_D46, 0, 0, "saved_a_D46");
   Grid0(17);
lrd_stmt(Csr6, "begin fnd_signon.audit_responsibility(:s,:l,:f,:a,:r,:t,:p)"
       "; end;", -1, 1, 1, 0);
   lrd_assign_bind(Csr6, "s", "D", &s_D216, 0, 0, 0);
   lrd_assign_bind(Csr6, "l", "<saved_a_D46>", &l_D217, 0, 0, 0);
   lrd_assign_bind(Csr6, "f", "1", &f_D218, 0, 0, 0);
   lrd_assign_bind(Csr6, "a", "810", &a_D219, 0, 0, 0);
   lrd_assign_bind(Csr6, "r", "20675", &r_D220, 0, 0, 0);
   lrd_assign_bind(Csr6, "t", "Windows PC", &t_D221, 0, 0, 0);
   lrd_assign_bind(Csr6, "p", "", &p_D222, 0, 0, 0);
   lrd_exec(Csr6, 1, 0, 0, 0, 0);
```

大きな数値の処理

大きな数（NUMBER データ型）は，GRID と ASCII ファイルでは異なる形式で表示されます。この違いにより，相関用に保存する値の検索時に数値を特定するのが困難になります。

たとえば，グリッド内に 1000003 と表示される値がある場合，これは記録ログ（ASCII ファイル）では 1e+0006 と表示されます。

解決法

再生中にエラーが生じ，相関ツールが前回の結果の中で値を見つけられない場合は，この値が別の形式で表現されているものとしてグリッドの中を探します。

ORA-00960

このエラーは，一意でないカラム名に発生することがあります。次に例を示します。

```
lrd_stmt(Csr9,"SELECT UOM_CODE, UOM_CODE, DESCRIPTION FROM "
        "MTL_UNITS_OF_MEASURE "
        "WHERE NVL(DISABLE_DATE, SYSDATE + 1) > "
        "SYSDATE ORDER BY UOM_CODE", -1, 1, 1, 0);
```

この場合，次のエラーが返されます。

```
"lrdo.c/fjParse: "oparse" ERROR return-code=960, oerhms=ORA-00960:
ambiguous column naming in select list"
```

解決法

少なくとも 1 つの一意でないカラムに別名を追加し，この別名を新しい一意の名前にして使うように，ステートメントを変更します。次に例を示します。

```
lrd_stmt(Csr9,"SELECT UOM_CODE,UOM_CODE second, DESCRIPTION
FROM"
        "MTL_UNITS_OF_MEASURE "
        "WHERE NVL(DISABLE_DATE, SYSDATE + 1) > "
        "SYSDATE ORDER BY UOM_CODE", -1, 1, 1, 0);
```

別の対処方法 : lrd ステートメントから ORDER BY を削除します。

ORA-2002

このエラーは，開いていないカーソルを使用しようとしたときに発生します。これは，複数の反復でユーザを再生し，スクリプトの複数のセクションに記録したときに生じます。

具体的には，カーソルが vuser_init セクションで開き，Actions セクションで閉じた場合に，カーソルを使用しようとすると 2 回目の反復でこのエラーが生じます。これは，カーソルが閉じられており，再び開かれていないからです。

たとえば，vuser_init セクションに *lrd_open_cursor* があり，Actions セクションに *lrd_close_cursor* がある場合，このユーザの再生で複数回の反復を行うと，2 回目の反復でエラーが生じます。これは，開いていないカーソルを使用しようとしたためです（カーソルを最初の反復で閉じ，2 回目の反復では開いていないためです)。

解決法

この問題を最も簡単に解決するには，問題のカーソルの **lrd_close_cursor** または **lrd_close_connection** を *vuser_end* セクションに移動します。

データベース・プロトコル (lrd)

記録された非同期操作の再生はサポートされていません。

クライアント・バージョンの誤り

実行している Oracle クライアントのバージョンが正しくない場合，次のエラーが返されます。

"Error: lrdo_open_connection: "olog" LDA/CDA return-code_019: unable to allocate memory in the user side"

解決法

製品の bin ディレクトリにあるライブラリ情報を *lrd.ini* ファイルで修正します。このファイルには記録および再生中にどのバージョンのデータベース・サポートがロードされるかを示す設定情報が含まれています。ファイルに各タイプのホストごとのセクションがあります。たとえば，*lrd.ini* ファイル内の HP/UX 上の Oracle セクションは次のようになります。

```
[ORACLE_HPUX]
;816=liblrdhpo816.sl
;81=liblrdhpo81.sl
;80=liblrdhpo80.sl
73=liblrdhpo73.sl
72=liblrdhpo72.sl
```

この設定により，クライアントが Oracle 8.1.6 を使用していれば LoadRnner ライブラリ liblrdhpo816.sl を，Oracle 8.1.5 を使用していれば，liblrdhpo81.sl を仮想ユーザが使用するということがわかります。

UNIX 上の再生では，lrd *ini* ファイルで使用するデータベースの正しいバージョンを表示するようにします。Oracle 8.1.5 を使用して HP/UX 用仮想ユーザを再生するとします。その場合，Oracle のそのほかのバージョンを示す行は行の先頭を「;（セミコロン）」でコメントアウトします。

すると lrd.ini ファイルは次のようになります。

```
[ORACLE_HPUX]
;816=liblrdhpo816.sl
81=liblrdhpo81.sl
;80=liblrdhpo80.sl
73=liblrdhpo73.sl
72=liblrdhpo72.sl
```

アプリケーションが lrd.ini ファイルに記されている DLL を使用していない場合，Win32 も変更します。たとえば，PowerBuilder 6.5 は Oracle 8.0.5 を使用しますが，DLL は ora803.dll を使用し，ora805.dll は使用しません。その場合，ORACLE_WINNT の項目で 805 および 804 をコメントアウトするか，805 のセクション

```
805=lrdo32.dll+ora805.dll
```

を次のように変更します。

```
805=lrdo32.dll+ora803.dll
```

19

Enterprise Java Beans（EJB）プロトコル

本章の内容

概念

タスク

リファレンス

概念

EJB テストの概要

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。仮想ユーザ・スクリプトを記録して生成するには，Java Record/Replay 仮想ユーザを使用します。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 仮想ユーザ・タイプを使用します。

EJB テスト用 仮想ユーザと標準 Java 仮想ユーザとの違いは，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，658 ページの「EJB 仮想ユーザ・スクリプトを実行する方法」を参照してください。

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 仮想ユーザ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。

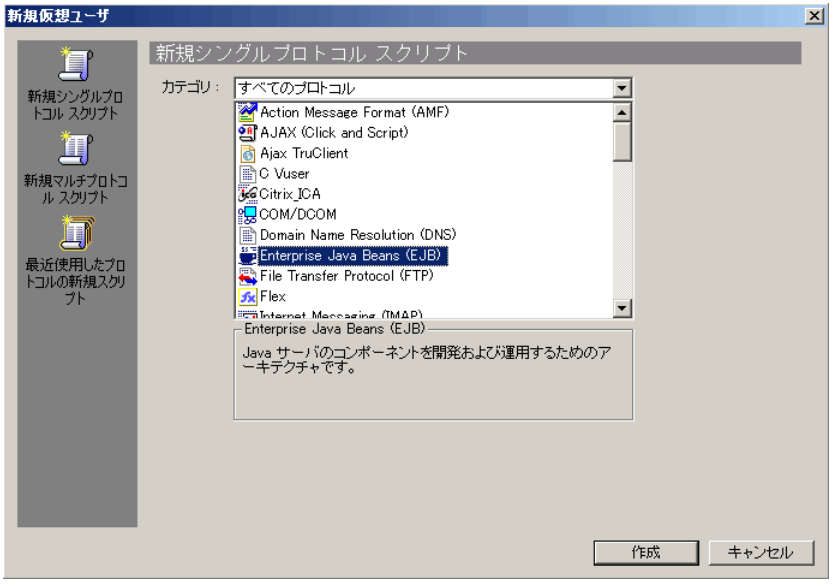
これにより，**webserver.log** ファイルに詳しいデバッグ情報や，その他の潜在的に重要なエラー・メッセージが出力されるようになります。

タスク

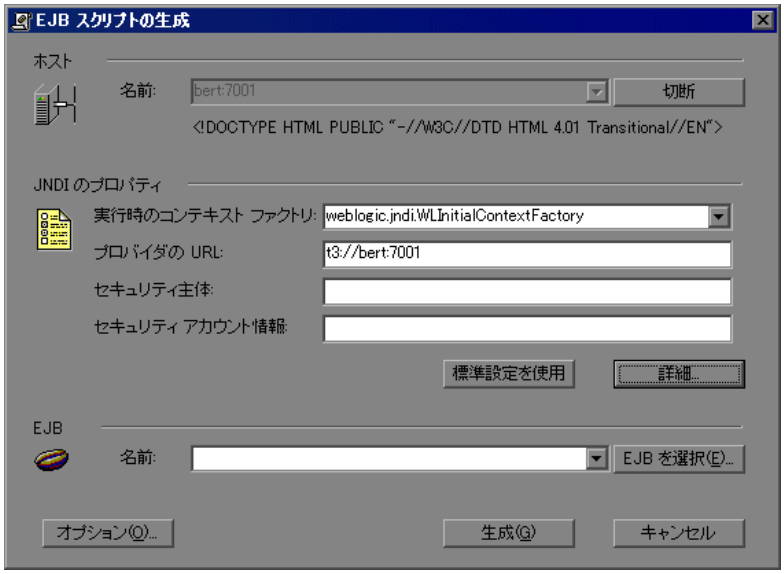
EJB 仮想ユーザ・スクリプトを作成する方法

EJB 仮想ユーザ・スクリプトの作成は，ほかのプロトコルを使用した仮想ユーザ・スクリプトの作成とは異なります。このタスクでは，EJB 仮想ユーザ・スクリプトを作成する方法について説明します。

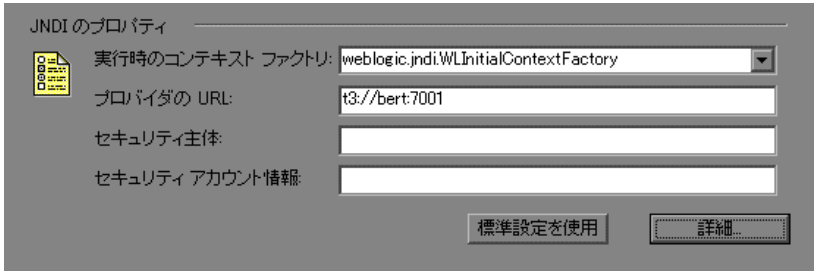
1 ［**ファイル**］>［**新規作成**］を選択するか，［**新規作成**］ボタンをクリックします。［新規仮想ユーザ］ダイアログ・ボックスが開きます。



2 ［**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 仮想ユーザ・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。

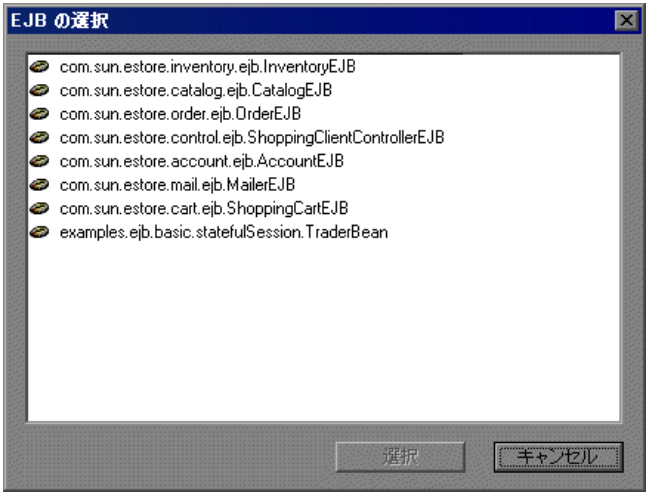
次に JNDI の必須プロパティの 2 つの標準設定値を示します。

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
OracleOracle	com.evermind.server.AppplicationClientInitialContextFactory	ormi://<appserver_host>/<application_name>（<oc4j>/config/server.xml 内の EJB の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 仮想ユーザ・スクリプトを実行する方法

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，671 ページの「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 21 章，「Java プロトコル」の Java 関数の挿入に関する項を参照してください。

4 実行環境を設定します。詳細については，419 ページの「実行環境の設定」を参照してください。

5 有効な Java 環境であることを確認します。詳細については，663 ページの「Java 環境を設定および確認する方法」を参照してください。

6 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，スクリプトがコンパイルされます。

EJB Detector をインストールして実行する方法

このタスクでは，EJB Detector をインストールして実行する方法について説明します。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。

このタスクでは，次の手順を実行します。

➤ 659 ページの「EJB Detector のインストール」

➤ 659 ページの「EJB Detector の実行」

1 EJB Detector のインストール

a 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。

b 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)。

c 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。

2 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 をコマンド・ラインから実行するには，660 ページの「EJB Detector をコマンド・ラインから実行する方法」を参照してください。

➤ EJB Detector をバッチ・ファイルから実行するには，662 ページの「EJB Detector をバッチ・ファイルから実行する方法」を参照してください。

EJB Detector をコマンド・ラインから実行する方法

このタスクでは，EJB Detector をコマンド・ラインから実行する方法について説明します。

1. 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。
3. WebLogic 4.x の場合：<WebLogic ディレクトリ >¥lib¥weblogicaux.jar
4. WebSphere 3.x の場合：<WebSphere ディレクトリ >¥lib¥ujc.jar
5. 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。

6 コマンド・ラインに次の文字列を入力します。

```
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**：デプロイ可能な **.ear** ファイルへの完全パス，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 をバッチ・ファイルから実行する方法

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。

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。

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 :** デプロイ可能な **.ear** ファイルへの完全パス，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 ディレクトリ >¥lib¥weblogicaux.jar
- WebSphere 3.x の場合：<WebSphere 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」）。

Java 環境を設定および確認する方法

このタスクでは，Java 環境を設定および確認する方法について説明します。

このタスクでは，次の手順を実行します。

- 663 ページの「Websphere 3.x ユーザの場合：」
- 664 ページの「WebSphere 4.0 ユーザの場合：」
- 664 ページの「Oracle OC4J ユーザの場合：」
- 665 ページの「Sun J2EE ユーザの場合：」
- 665 ページの「WebLogic 4.x - 5.x ユーザの場合：」
- 665 ページの「WebLogic 6.x および 7.0 ユーザの場合：」

1 Websphere 3.x ユーザの場合：

a 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。

b 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。
- ファイル iioprt.jar と rmiorb.jar を，<WS>¥jdk¥jre¥lib¥ext フォルダから <JDK>¥jre¥lib¥ext ディレクトリにコピーします。
- 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。

2 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。

---

3 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。

4 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 ファイルへの完全パスを指定します。

5 WebLogic 4.x - 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。

6 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。

リファレンス

EJB ユーザ・インタフェース

このセクションの内容

➤［EJB スクリプトの生成］ダイアログ・ボックス（666 ページ）

［EJB スクリプトの生成］ダイアログ・ボックス

EJB スクリプトの記録を開始できます。

利用方法	**［ファイル］>［新規作成］>［Enterprise Java Beans (EJB)］**
関連タスク	659 ページの「EJB Detector をインストールして実行する方法」

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
詳細(A)...	［JNDI の詳細プロパティ］ダイアログ・ボックスが開きます。必要なプロパティを指定します。
オプション(O)...	関連する記録オプションのダイアログ・ボックスが開きます。詳細については，313 ページの「記録オプション」を参照してください。
接続 切断	サーバに接続または切断します。スクリプトを生成するには，サーバに接続している必要があります。
生成(G)	仮想ユーザ・スクリプトが生成されます。
EJB を選択(E)...	［EJB の選択］ダイアログ・ボックスが開きます。
EJB 名	テストを作成する対象の EJB の名前。

UI 要素	説明
ホスト名	EJB Detector がインストールされているマシン。接続するためには Detector が稼働している必要があります。
JDNI のプロパティ	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。

EJB 仮想ユーザ・スクリプトの例

VuGen は，仮想ユーザ・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 仮想ユーザ・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();
        }
```

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 のホスト名を「dod」と指定しています。

```
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・サーバ名に置き換えます。たとえば次のようになります。

```
p.put(javax.naming.Context.PROVIDER_URL, "t3://bealogic:7001")
```

記録する前に［EJB スクリプトの生成］ダイアログの［**JDNI のプロパティ**］セクションでプロバイダの URL を指定できます。そうすると，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「create」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 仮想ユーザ・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 仮想ユーザ・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 仮想ユーザ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。

20

Flex プロトコル

本章の内容

概念

- Flex の概要（674 ページ）
- Flex スクリプトの外部オブジェクト（675 ページ）

タスク

- XML ツリーをクエリする方法（677 ページ）
- 外部 Java シリアライザを使用してシリアル化する方法（679 ページ）
- LoadRunner シリアライザを使用してスクリプトをシリアル化する方法（680 ページ）

リファレンス

- Flex 関数と例（681 ページ）

トラブルシューティングと制限事項（682 ページ）

概念

Flex の概要

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 仮想ユーザ・スクリプトを作成できます。VuGen の **Flex** プロトコルでは，AMF0 や 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**仮想ユーザ・プロトコルを使用します。

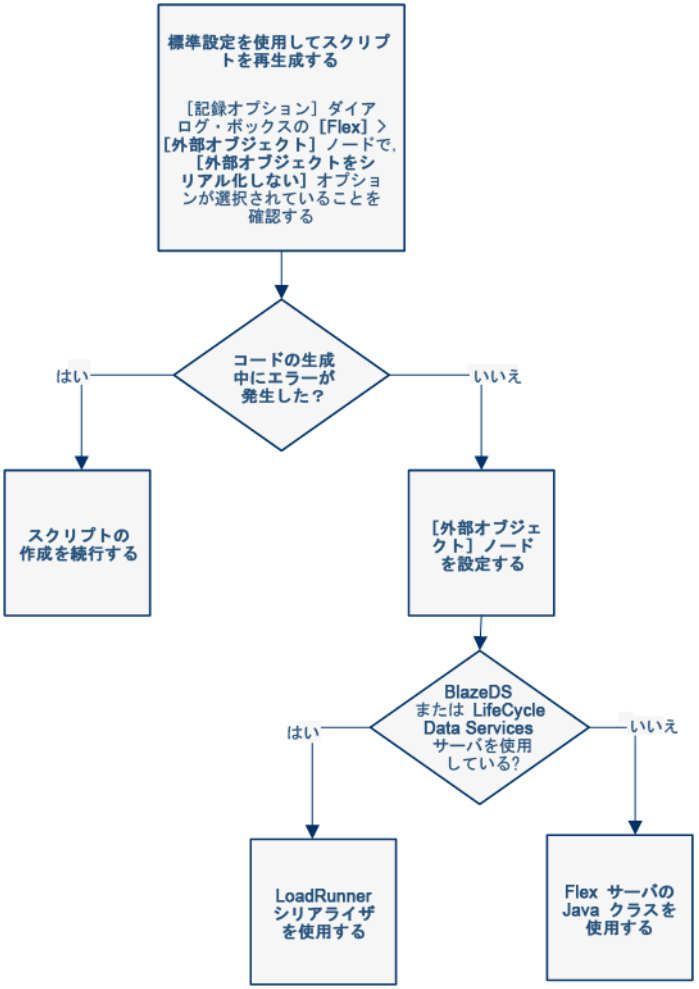
Flex スクリプトの外部オブジェクト

Flex アプリケーションを記録するとき，通常，既知のシリアル化メソッド（AMF）を使用してクライアントとサーバ間で情報が渡されます。この場合，VuGen によって **flex_amf_call** が作成されます。

ただし，特定の AMF オブジェクトでカスタム・シリアル化メソッド（外部）が使用されている場合，コード生成通知ダイアログ・ボックスが開きます。このダイアログ・ボックスには，各例外および推奨アクションに関する詳細が表示されます。詳細については，129 ページの「コード生成通知」を参照してください。

［**記録オプション**］>［**Flex**］>［**外部オブジェクト**］ノードの設定の詳細については，352 ページの「［Flex］-［外部オブジェクト］ノード（記録）」を参照してください。

次のフローチャートは，Flex スクリプトの外部オブジェクトを解決する手順を示しています。



外部オブジェクトをシリアル化する方法の詳細については，次のセクションを参照してください。

- 352 ページの「[Flex] -［外部オブジェクト］ノード（記録）」
- 679 ページの「外部 Java シリアライザを使用してシリアル化する方法」
- 680 ページの「LoadRunner シリアライザを使用してスクリプトをシリアル化する方法」

タスク

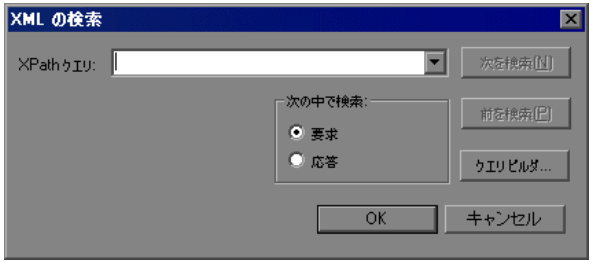
XML ツリーをクエリする方法

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。

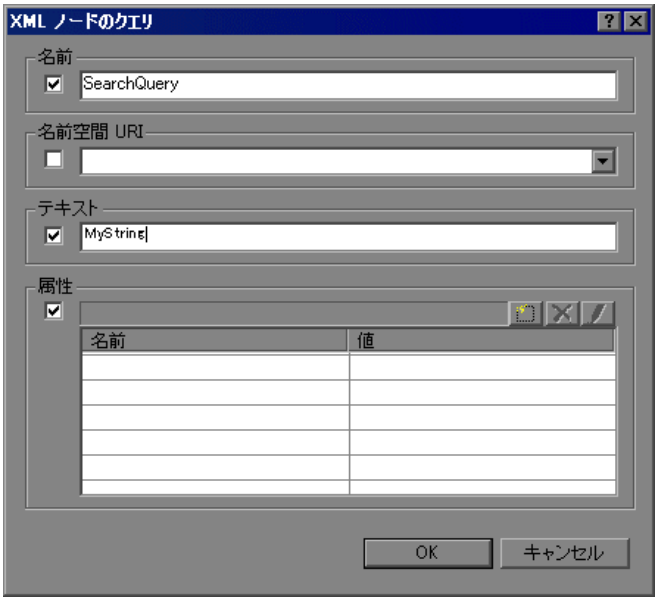
クエリを行うには，次の手順を実行します。

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。［**XML の検索**］ボタンをクリックします。［XML 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。



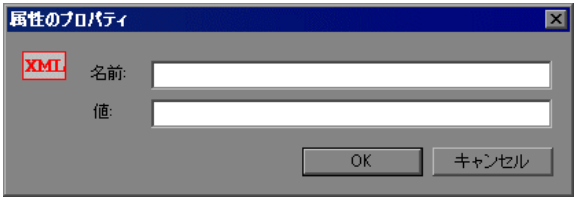
4 ［**名前**］セクションを有効にして，ノードまたは要素の名前を検索します。

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［XML 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。

外部 Java シリアライザを使用してシリアル化する方法

Flex サーバの Java クラスを使用して，スクリプトの AMF メッセージをシリアル化できます。このプロセスは簡略化されており，AMF オブジェクトで外部インタフェースが実装されている場合，アプリケーションの JAR ファイルを含めるだけで済みます。

外部シリアライザを使用して外部オブジェクトを処理するには，次の手順で行います。

1 ［**記録オプション**］>［**Flex**］>［**外部オブジェクト**］ノードで，［**オブジェクトのシリアル化方法**］を選択し，ドロップダウン・メニューから［**カスタム Java クラス**］を選択します。

2 ［**フォルダのすべてのクラス ファイルを追加**］ボタンまたは［**JAR ファイルまたは ZIP ファイルを追加**］ボタンを使用して，関連するファイルを追加します。次のファイルを追加します。

a **BlazeDS または LCDS の場合**，次の JAR ファイルを追加します。

- flex-messaging-common.jar
- flex-messaging-core.jar

b スクリプトを再生成し，エラーがないかを確認します。［**生成オプション**］ボタンを使用して［記録オプション］ダイアログ・ボックスを開き，必要なアプリケーションの JAR ファイルを追加します。

3 追加したファイルが，VuGen マシンとすべての Load Generator で同じ場所にあることを確認します。

Java シリアライザの注意と制限事項

- サポートされる JDK バージョン：1.6 以前
- サポートされるサーバ：BlazeDS および LifecycleDS
- Microsoft .NET クラスはサポートされません。
- コード生成時に，VuGen では指定された jar ファイルを使用してバッファの読み取りと書き込みが可能であるかどうかを確認することによって，要求バッファの有効性テストが実行されます。この有効性テストに失敗した場合は，クラスに LoadRunner との互換性がないことを示します。

LoadRunner シリアライザを使用してスクリプトをシリアル化する方法

LoadRunner シリアライザを使用して外部オブジェクトのシリアル化を試みることができます。このオプションを使用すると，予期しないエラーや無効なステップが発生する可能性があるので，開いているすべてのスクリプトを必ず保存してから実行してください。

LoadRunner シリアライザを使用するには，次の手順で行います。

1 VuGen で開いているすべてのスクリプトを保存します。

2 ［**記録オプション**］>［**Flex**］>［**外部オブジェクト**］ノードで，［**オブジェクトのシリアル化方法**］を選択し，ドロップダウン・メニューから［**LoadRunner AMF シリアライザ**］を選択します。

リファレンス

Flex 関数と例

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 仮想ユーザ・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。

例：

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，*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。

トラブルシューティングと制限事項

このセクションでは，Flex プロトコルのトラブルシューティングと制約事項について説明します。

FLEX プロトコルは，Unix LoadGenerator では実行できません。

21

Java プロトコル

本章の内容

概念

タスク

➤ スクリプト生成を手作業で設定する方法（702 ページ）
➤ カスタム Java フィルタの作成方法（706 ページ）

リファレンス

➤ フック・ファイルの構造（708 ページ）
➤ Java アイコン参照リスト（711 ページ）

概念

Java プロトコルの記録の概要

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 を仮想ユーザ 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 コンパイラの **javac.exe** によって，スクリプトに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 設定に組み込みます。

スクリプトを記録と手作業で拡張する場合，Java 仮想ユーザ・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，仮想ユーザ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 22 章，「Java プロトコル - 手作業による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 のインストール手順に従って設定されていることを確認してください。

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 仮想ユーザ・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，907 ページの「Web 仮想ユーザ・スクリプトを Java に変換する方法」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 コンパイラの **javac.exe** によって，スクリプトに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 設定）に組み込みます。詳細については，*HP LoadRunner Controller*，*Performance Center*，または *HP Business Availability Center* のドキュメントを参照してください。

Java 仮想ユーザ・スクリプトの概要

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。

- Imports
- Code
- Variables

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 仮想ユーザ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，*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。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，718 ページの「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。

RMI-IIOP の概要

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仮想ユーザ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

Corba 記録オプション

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］
- ［Use DLL hooking to attach VuGen support］

CORBA アプリケーション・ベンダ・クラス

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 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，723 ページの「Java 環境の設定」を参照してください。

Jacada 仮想ユーザの記録

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 タイプの仮想ユーザを使用します。詳細については，第 34 章，「Web 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 仮想ユーザ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［実行環境の設定］の［Classpath］ノードの［**Classpath エントリ**］リストで設定されているクラスパス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，HP サポート窓口までお問い合わせください。

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 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);

  // 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 つの仮想ユーザのシャットダウンによってほかのすべての仮想ユーザ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

このセクションでは，RMI 特有の Java 仮想ユーザ・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，198 ページの「Java スクリプトの相関 - シリアル化」を参照してください。

Jacada を使った作業

Jacada が使用された Java 仮想ユーザ・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 22 章，「Java プロトコル - 手作業によるスクリプトのプログラミング」を参照してください。

Java カスタム・フィルタ - 概要

このセクションでは，カスタム Java フィルタを作成するために必要な参考情報について説明します。タスクの詳細については，706 ページの「カスタム Java フィルタの作成方法」を参照してください。

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 仮想ユーザを使用して，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。カスタム・フィルタを定義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。

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。

Java カスタム・フィルタ - 包含する要素の指定

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。この引数がシリアル化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ル化します。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ル化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ル化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，198 ページの「Java スクリプトの相関 - シリアル化」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。

タスク

Java 仮想ユーザ・スクリプトの記録方法

このタスクでは，Java 仮想ユーザ・スクリプトの記録方法について説明します。

このタスクでは，次の手順を実行します。

- 699 ページの「前提条件」
- 699 ページの「新しい Java Record Replay プロトコル・スクリプトの作成」
- 699 ページの「[記録開始] ダイアログ・ボックスの設定」
- 699 ページの「[記録オプション] の設定」

1 前提条件

仮想ユーザ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）。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。仮想ユーザ・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。

2 新しい Java Record Replay プロトコル・スクリプトの作成

［**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。

3［記録開始］ダイアログ・ボックスの設定

［記録開始］ダイアログ・ボックスでアプリケーションの詳細を入力します。ユーザ・インタフェースの詳細については，126 ページの「［記録開始］ダイアログ・ボックス」を参照してください。

4［記録オプション］の設定

［記録開始］ダイアログ・ボックスの［オプション］をクリックして，［記録オプション］ダイアログ・ボックスを開きます。［**レコーダ オプション**］ノードの［**記録済みプロトコル**］フィールドで，記録するメイン・プロトコルを設定します。複数の Java プロトコルを記録する場合，［**拡張子リスト**］フィールドに追加のプロトコルを入力します。

Windows XP および 2000 Server を使用して Java スクリプトを記録する方法

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。［**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。

パッケージの一部としてスクリプトを実行する方法

このセクションは，Jacada タイプのスクリプトには適用されません。

Java 仮想ユーザ・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仮想ユーザで使用するには，必要なメソッドのパッケージにその仮想ユーザを追加します。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。

手作業で Java メソッドを挿入する方法

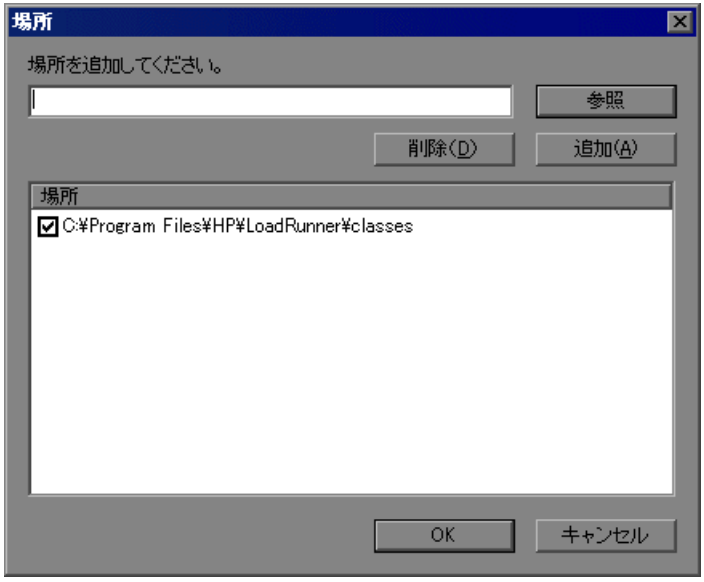
Java 関数ナビゲータを使用して Java 関数の表示やスクリプトへの追加をします。設定ファイルを変更して，関数の生成についての設定をカスタマイズできます。詳細については，358 ページの「［一般］の［スクリプト］ノード」を参照してください。

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。

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。ダイアログ・ボックスの下部には，Java オブジェクトの詳細が表示されます。

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 ステートメントや仮想ユーザ API Java 関数（第 22 章，「Java プロトコル - 手作業による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。

スクリプト生成を手作業で設定する方法

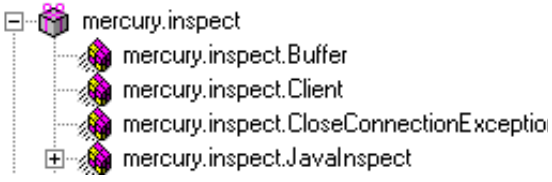
ナビゲータによるスクリプトへのメソッドの追加方法をカスタマイズできます。

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
```

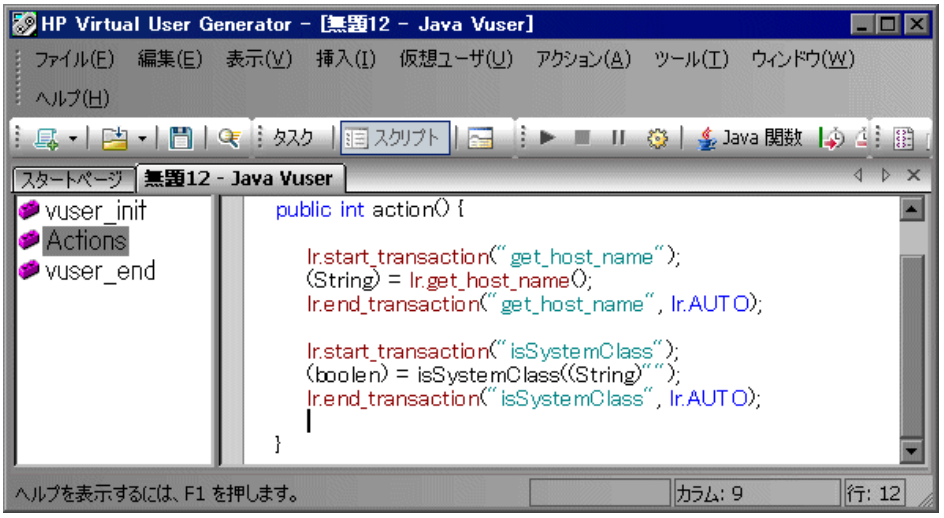
クラス名のパス

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仮想ユーザ・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.message((String));
lr.think_time((int)0);	lr.think_time((int));
lr.enable_redirection((boolean)false);	lr.enable_redirection((boolean));
lr.save_data((byte[])null, (String)"");	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)"",...);
```

カスタム Java フィルタの作成方法

このタスクでは，カスタム Java フィルタの作成方法について説明します。参考情報については，695 ページの「Java カスタム・フィルタ - 概要」を参照してください。

フック・ファイルの構造の詳細については，708 ページの「フック・ファイルの構造」を参照してください。

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を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。

カスタム java フィルタを定義するには，次の手順で行います。

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。user.hook ファイルの構造の詳細については，708 ページの「フック・ファイルの構造」を参照してください。
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。［ログのレベル］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，696 ページの「Java カスタム・フィルタ - 包含する要素の指定」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，195 ページの「Java スクリプトの相関」および 198 ページの「Java スクリプトの相関 - シリアル化」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。

リファレンス

フック・ファイルの構造

次のセクションでは，標準的な .hooks 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¥.*

メソッド

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がフックされます。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，そのクラス継承したクラスがフックされます。フックがインタフェースを記述している場合，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。

Java アイコン参照リスト

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 ()

22

Java プロトコル - 手作業によるスクリプトのプログラミング

本章の内容

概念

タスク

概念

手作業による Java スクリプトのプログラミング - 概要

Java コードを使って仮想ユーザ・スクリプトを作成するには，**Java** タイプの仮想ユーザを使用します。この仮想ユーザ・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。仮想ユーザ・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 21 章，「Java プロトコル」では，**Java Record Replay** 仮想ユーザ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 互換の仮想ユーザ・スクリプトを作成する第一歩は，**Java 仮想ユーザ**タイプの新しい仮想ユーザ・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 仮想ユーザ関数を追加して，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 仮想ユーザ・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 仮想ユーザ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，スクリプトがコンパイルされます。

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 設定）に組み込みます。詳細については，*HP LoadRunner Controller*，*Performance Center*，または *HP Business Availability Center* のドキュメントを参照してください。

Java プロトコルのプログラミングに関するヒント

Java 仮想ユーザ・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仮想ユーザを 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，仮想ユーザの数が限られていれば，Java 仮想ユーザ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 個の仮想ユーザを実行すると，**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仮想ユーザ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，すべての仮想ユーザ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，それらの仮想ユーザをプロセスとして実行できます。仮想ユーザ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 11 章，「実行環境の設定」を参照してください。

基本的な Java 仮想ユーザ・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 仮想ユーザが 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;
}
```

Java 仮想ユーザ・スクリプトの実行

Java 仮想ユーザ・スクリプトは，コンパイル後に実行される点が C 仮想ユーザ・スクリプトとは異なります。C 仮想ユーザ・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，仮想ユーザを**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，仮想ユーザ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 仮想ユーザ・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，仮想ユーザ・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。

タスク

手作業で Java スクリプトを作成する方法

このタスクでは，手作業でカスタム Java スクリプトを作成および編集する方法について説明します。

このタスクでは，次の手順を実行します。

- 719 ページの「新しいスクリプトの作成」
- 720 ページの「スクリプトへのコードの挿入」
- 722 ページの「追加の LoadRunner API 関数の挿入」
- 722 ページの「追加の Java 関数の挿入」

1 新しいスクリプトの作成

a VuGen を開きます。

b ［**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。

c 仮想ユーザのタイプを選択するリストから［**ユーザ定義**］>［**Java 仮想ユーザ**］を選択し，［**OK**］をクリックします。空の Java 仮想ユーザ・スクリプトが表示されます。

d 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。

2 スクリプトへのコードの挿入

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。このタイプの仮想ユーザ・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	仮想ユーザを初期化（ロード）するとき
action	クライアントの動作	仮想ユーザが「実行」状態のとき
end	ログオフ処理	仮想ユーザ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，仮想ユーザ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。このセクションに **org.omg.CORBA.ORB** 関数をインポートして，この関数が反復のたびに呼び出されないようにします。

```
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

仮想ユーザ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 11 章,「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，仮想ユーザ 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;
  }
```

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仮想ユーザに実行させるコードを配置します。

end メソッドは，仮想ユーザ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;
  }
```

3 追加の LoadRunner API 関数の挿入

VuGen には，Java 仮想ユーザ・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，*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。Java 仮想ユーザ・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。

4 追加の Java 関数の挿入

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
{
...
}
```

5 スクリプト拡張機能の追加

ランデブー・ポイント，トランザクション，および出力メッセージなどのスクリプト拡張機能を追加します。詳細については，723 ページの「Java スクリプトの拡張方法」を参照してください。

6 Java 環境の設定

Java 仮想ユーザ・スクリプトを実行する前に，仮想ユーザ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 仮想ユーザ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［実行環境の設定］の［Classpath］ノードの［**Classpath エントリ**］リストで設定されているクラスパスに含まれている必要があります。

Java スクリプトの拡張方法

このタスクでは，カスタム Java スクリプトを拡張する方法について説明します。

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，仮想ユーザ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仮想ユーザ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，多数の仮想ユーザ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。ある仮想ユーザがランデブー・ポイントに到達すると，Controller によって，ほかのすべての仮想ユーザがランデブー・ポイントに到着するまでその仮想ユーザは待機させられます。

仮想ユーザ・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。

➤ 仮想ユーザ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;
}
```

仮想ユーザ情報の取得

次の関数を仮想ユーザ・スクリプトに追加して，仮想ユーザ情報を取得できます。

lr.get_attrib_string	仮想ユーザ・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	仮想ユーザ・グループの名前を返します。
lr.get_host_name	仮想ユーザ・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 が返されます。（LoadRunner のみ）
lr.get_vuser_id	現在の仮想ユーザの ID が返されます。（LoadRunner のみ）

次の例では，lr_get_host_name 関数を使って 仮想ユーザ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，*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。

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 仮想ユーザ・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。

仮想ユーザ・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	仮想ユーザ・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，*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。

仮想ユーザが 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，仮想ユーザ・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*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。

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。仮想ユーザは，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，仮想ユーザがループの中で 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。仮想ユーザ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 11 章，「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，*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。

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 仮想ユーザ・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 仮想ユーザ・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ようになります。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
```

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，*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。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，*LoadRunner Controller*，*Performance Center*，または *HP Business Availability Center* のドキュメントを参照してください。

23

Java over HTTP プロトコル

本章の内容

概念

タスク

リファレンス

概念

Java over HTTP プロトコルの概要

Java over HTTP プロトコルは，Java ベースのアプリケーションとアプレットを記録するように設計されています。Web 関数を使用して Java 言語のスクリプトが生成されます。このプロトコルは，HTTP 上で Java リモート呼び出しの記録と再生ができるという点で，ほかの Java プロトコルと区別されます。

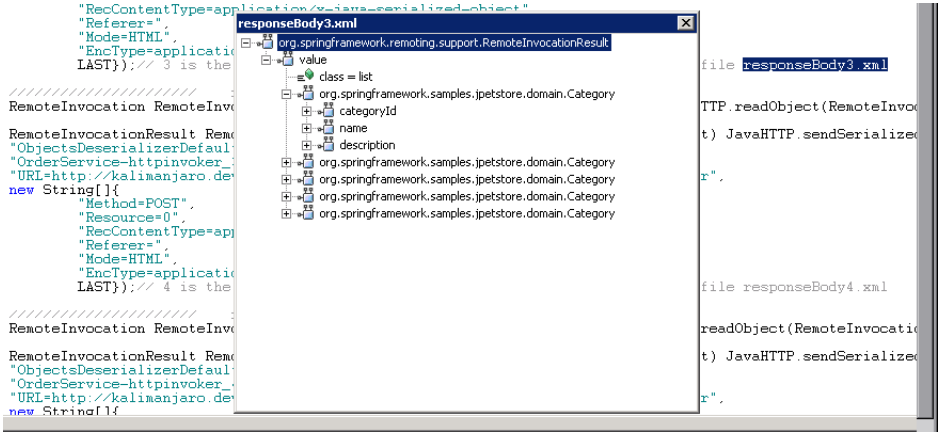
XML 形式での応答と要求の表示

記録フェーズでは，応答および要求ごとに対応する XML（バイナリ java オブジェクトを表す）を表示できます。

xml データを表示するには，次の手順で行います。

1 コード内にある目的の要求または応答のセクションを探します。コメントアウトされた **RequestBodyX.xml** または **ResponseBodyX.xml** を右クリックします。

2［**XML の表示**］を選択します。XML が別のウィンドウに表示されます。



タスク

Java over HTTP での記録方法

Java over HTTP で記録するには，記録するデータをシリアル化解除するために使用する .jar ファイルを指定する必要があります。

このタスクでは，関連する .jar ファイルを探してクラスパスに追加する方法について説明します。

このタスクでは，次の手順を実行します。

- 733 ページの「Java アプレットの記録」
- 734 ページの「ローカル Java アプリケーションの記録」

Java アプレットの記録

アプリケーションで Java アプレットを使用している場合，関連する .jar ファイルを探して，クラスパスで有効にする必要があります。

関連する .jar ファイルを探すには，次の手順で行います。

1 ［**コントロール パネル**］>［**Java**］>［**基本**］タブ >［**インターネット一時ファイル**］>［**設定**］>［**ファイルの削除**］を選択して JAR キャッシュをクリアします。

2 アプリケーションを開いていくつかのビジネス・プロセスを実行し，アプリケーションの .jar ファイルを JAR キャッシュに再度読み込みます。完了したら，アプリケーションを閉じます。

3 ［**コントロール パネル**］>［**Java**］>［**基本**］タブ >［**インターネット一時ファイル**］>［**表示**］を選択します。この JAR キャッシュのリストには，アプリケーションで使用された .jar ファイルのみが表示されます。

4 これらのファイルをダウンロードします。記載されている順序でオプションを実行します。成功したら次の手順に進み，.jar ファイルをクラスパスに追加します。

a オプション 1：.jar ファイルごとに，表示されている URL にアクセスしてファイルをダウンロードします。1 つまたはすべての .jar ファイルをダウンロードできない場合，次のオプションを続行します。

b オプション 2：［**コントロール パネル**］>［**Java**］>［**基本**］タブ >［**インターネット一時ファイル**］>［**設定**］>［**ファイルの削除**］を選択して，もう一度キャッシュをクリアします。再度アプリケーションを開き，いくつかのビジネス・プロセスを実行します。アプリケーションは閉じないでください。Java コンソールを開きます。.jar ファイルごとに，保存場所（コンピュータの一時ファイル内）を示すメッセージが表示されます。通常，ファイルはハッシュ化されており，.jar 拡張子はありません。名前を変更（それぞれの拡張子も .jar に変更）して，ファイルを既知の場所にコピーします。

c オプション 3：ファイルが Java コンソールに表示されない場合，［**コントロール パネル**］>［**Java**］>［**基本**］タブ >［**インターネット一時ファイル**］>［**設定**］>［**場所**］に表示されている一時ディレクトリを探します。指定した場所を開いて，サブフォルダにあるすべてのファイルの名前を .jar に変更します。メイン・フォルダにあるすべてのファイルは，名前を変更しないでください。

5［**記録オプション**］>［**Java 環境の設定**］>［**Classpath**］ノード内のクラスパスに .jar ファイルを追加します。詳細については，378 ページの「Java の［Classpath］ノード」を参照してください。

ローカル Java アプリケーションの記録

（アプレットではなく）ローカル Java アプリケーションを記録する場合，すべての .jar ファイルはすでにコンピュータ上に存在しています。

関連する .jar ファイルを探すには，次の手順で行います。

1 アプリケーションを起動したバッチ・ファイルを検索します。参照されるすべての .jar ファイルをクラスパスに追加する必要があります。

2 バッチ・ファイルが見つからない，またはわからない場合，アプリケーションのフォルダおよびサブフォルダにあるすべての .jar ファイルをクラスパスに追加します。

3［**記録オプション**］>［**Java 環境の設定**］>［**Classpath**］ノード内のクラスパスに .jar ファイルを追加します。詳細については，378 ページの「Java の［Classpath］ノード」を参照してください。

Java over HTTP スクリプトのデバッグ方法

このタスクでは，記録ステージおよび再生ステージの両方の要求データと応答データを比較して，Java over HTTP スクリプトをデバッグする方法について説明します。

このタスクでは，次の手順を実行します。

- 735 ページの「VM Param ノードへの引数の追加」
- 735 ページの「記録時のデータと再生時のデータの比較」
- 736 ページの「負荷テスト前の引数の削除」

1 VM Param ノードへの引数の追加

［**仮想ユーザ**］>［**実行環境の設定**］>［**Java VM**］ノードを選択します。［**追加の VM パラメータ**］フィールドに，次の文字を入力します。

```
-DdumpServerRequests=true -DdumpServerResponses=true
```

2 記録時のデータと再生時のデータの比較

スクリプト・ビューで，右クリックして［**スクリプト ディレクトリを開く**］を選択します。記録フェーズのデータは main ディレクトリにあります。再生フェーズのデータは，replay ディレクトリにあります。

RequestBodyX 形式のファイルには，要求データが含まれています。ResponseBodyX 形式のファイルには，応答データが含まれています。

記録時のデータと再生時のデータを比較してデバッグするには，記録フェーズと再生フェーズで名前が同じファイルを比較します。たとえば，main ディレクトリの RequestBody1 ファイル（記録フェーズ）と replay ディレクトリの RequestBody1 ファイルを比較します。通常，ファイルは同じになります。ファイルが同じでない場合は，スクリプトに問題があることを示している可能性があります。

3 負荷テスト前の引数の削除

［Java VM］ノードの［追加の VM パラメータ］フィールドに追加した項目に戻ります。

Java over HTTP スクリプトへのパラメータの挿入方法

応答または要求の各本体テキストの特定の場所にパラメータ関数を追加できます。通常，この場所は，応答または要求の本体の開始部分の 1 ～ 2 行下にある空白行になります。次の例では，各 requestBody セクションの空白行にパラメータ関数を追加できます。

```
//////////////////////    requestBody2.xml    //////////////////////
emoteInvocation RemoteInvocation_getUsernameList2 =
   (RemoteInvocation) JavaHTTP.readObject(RemoteInvocationBA0);
/INSERT PARAMETERIZATION AND CORRELATION CODE HERE
emoteInvocationResult RemoteInvocationResult_ArrayList2 =
   (RemoteInvocationResult) JavaHTTP.sendSerialized(RemoteInvocation_getUsernameList2, 2,
ObjectsDeserializerDefaultImpl",
OrderService-httpinvoker",
URL=http://kalimanjaro.devlab.ad:8080/jpetstore/remoting/OrderService-httpinvoker",
ew String[]{
       "Method=POST",
       "Resource=0",
       "RecContentType=application/x-java-serialized-object",
       "Referer=",
       "Mode=HTML",
       "EncType=application/x-java-serialized-object",
       LAST});// 2 is the number of the header file, record time response is at file responseBody2.xml

//////////////////////    requestBody3.xml    //////////////////////
emoteInvocation RemoteInvocation_getCategoryList3 =
   (RemoteInvocation) JavaHTTP.readObject(RemoteInvocationBA1);
/INSERT PARAMETERIZATION AND CORRELATION CODE HERE
emoteInvocationResult RemoteInvocationResult_ArrayList3 =
   (RemoteInvocationResult) JavaHTTP.sendSerialized(RemoteInvocation_getCategoryList3, 3,
ObjectsDeserializerDefaultImpl",
OrderService-httpinvoker_2",
URL=http://kalimanjaro.devlab.ad:8080/jpetstore/remoting/OrderService-httpinvoker",
ew String[]{
       "Method=POST",
       "Resource=0",
       "RecContentType=application/x-java-serialized-object",
       "Referer=",
       "Mode=HTML",
       "EncType=application/x-java-serialized-object",
       LAST});// 3 is the number of the header file, record time response is at file responseBody3.xml
```

リファレンス

トラブルシューティングと制限事項

このセクションでは，Java over HTTP プロトコルのトラブルシューティングと制限事項について説明します。

制限事項

- JDK 1.5 以上が必要です。
- 遅延モードの休止状態など，遅延評価オブジェクトはサポートされていません。
- アプリケーション・サーバにステートフルなシリアル化メカニズムがある場合，LoadRunner のシリアル化解除が妨害され，シリアル化されていないデータや予期しないエラーが発生する可能性があります。
- このプロトコルでは，次のメニュー項目は使用できません。
  - ［挿入］>［新規ステップ］/［トランザクション開始］/［トランザクション終了］/［ランデブー］

例外エラー検査の無効化

例外エラーが発生してもそのエラーが重要でないことがわかっている場合，VuGen ではそのようなエラー・メッセージをすべて無効にできます。これを行うには，［**仮想ユーザ**］>［**実行環境の設定**］>［**Java VM**］ノードを選択します。［**追加の VM パラメータ**］フィールドの現在のエントリの最後に次の文字を追加します。

```
-DvalidateServerResponse=false
```

また，スクリプト・ビューで sendSerialized 関数に終了引数を追加して，特定のステップのエラー検査の動作を変更できます。詳細については，*HP LoadRunner オンライン関数リファレンス*を参照してください。

オブジェクトのプライベート・メンバを相関できない

オブジェクトのプライベート・メンバであるデータを相関またはパラメータ化する必要がある場合，**lrapi.lr2.fieldSetter** および **lrapi.lr2.fieldGetter** 関数を使用できます。

```
 RemoteInvocation RemoteInvocation2 = (RemoteInvocation)
JavaHTTP.readObject(RemoteInvocationBA0);
        RemoteInvocation.methodName="applyToSchool";
        Student student=RemoteInvocation.arguments[0];

        Map grades=lr2.fieldGetter(student,"grades");//grades は Student のプライベート・
メンバです。
        grades.put("Math","95");
        lr2.fieldSetter(student,"super.name","Tom");
         //Student クラスは，Person から name フィールドを継承します。name フィールド
は文字列です。
        lr2.fieldSetter(student,"super.ID","98764321");
         //Student クラスは，Person から ID フィールドを継承します。ID フィールドは整数
です。


        RemoteInvocationResult RemoteInvocationResult_ArrayList2 =
         (RemoteInvocationResult) JavaHTTP.sendSerialized(RemoteInvocation2, 2,
         "ObjectsDeserializerDefaultImpl",....
```

24

LDAP プロトコル

本章の内容

概念

概念

LDAP プロトコルの概要

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，742 ページの「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例

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。

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。

```
Action()
{
                // 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
¥	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
```

LDAP 接続オプション

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][username:[password]@][server[:port]]
```

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，*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。

25

メール・サービス・プロトコル

本章の内容

概念

概念

メール・サービス・プロトコルの概要

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 仮想ユーザ・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。

IMAP プロトコルの概要

IMAP 仮想ユーザ・スクリプト関数は，Internet Mail Application Protocol（IMAP）を記録します。このプロトコルでは，記録はサポートされていません。

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，*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 プロトコルの概要

MAPI 仮想ユーザ・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。これらの関数の構文情報の詳細については，*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。

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。

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 プロトコルの概要

POP3 仮想ユーザ・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，*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 プロトコルの概要

SMTP 仮想ユーザ・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，*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;
}
```

26

Microsoft .NET プロトコル

本章の内容

概念

Microsoft .NET プロトコルの概要

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仮想ユーザ・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。

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。

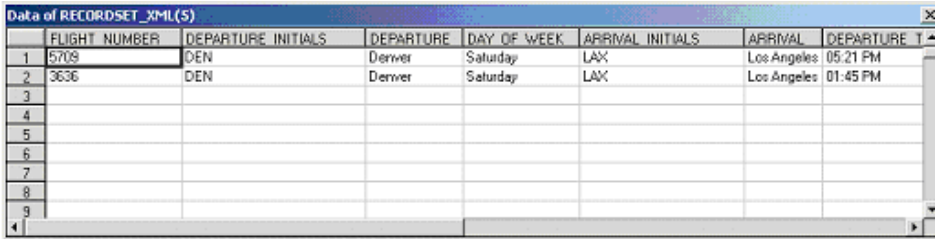
```
SqlDataReader_1 = SqlCommand_1.ExecuteReader();
LrReplayUtils.DoDataRead(SqlDataReader_1, out valueTable_1, true, 27);
```

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，*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。

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。

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。

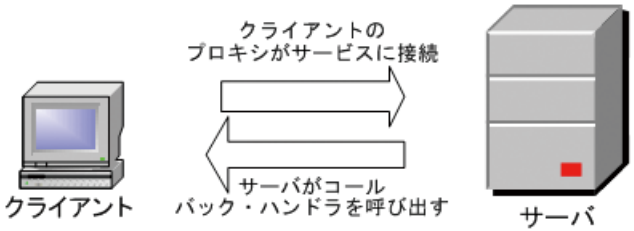
WCF はプロキシ・オブジェクトを作成し，サービスにデータを提供します。また，サービスによって返されたデータを呼び出し元が期待する形式にまとめます。

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

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。

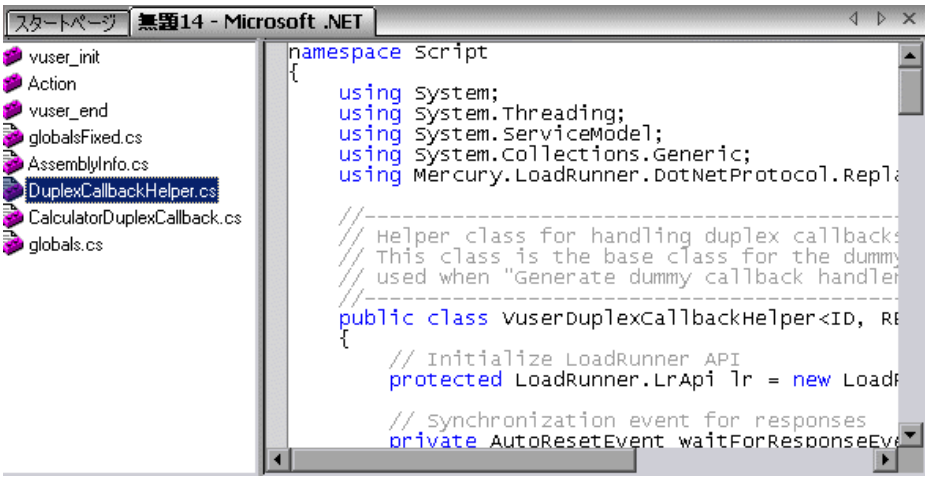
VuGen に対して元のコールバックを置き換えるよう指定するには，［**ダミー コールバック ハンドラを生成**］記録オプションをアクティブにします。詳細については，389 ページの「リモート・オブジェクトのプロパティ」を参照してください。

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 **仮想ユーザ** *<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機。
    // 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「仮想ユーザ＜*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。

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。このセクションでは，推奨するカスタマイズを示します。

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 仮想ユーザ XXX : 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。

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。

詳細については，387 ページの「［Microsoft .NET］の［記録］ノード」を参照してください。

```
#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);
```

非同期呼び出し

VuGen でリモート・オブジェクトの非同期呼び出しが記録されると，387 ページの「[Microsoft .NET］の［記録］ノード」で呼び出しが処理される方法を指定できます。このオプションは，特に .NET Remoting と WCF の環境に関係します。

次のいずれかのオプションで VuGen を設定できます。

- **［標準設定で元のコールバックを呼び出す］**：スクリプトの生成と再生を行うときに，記録されたアプリケーションの元のコールバックを使用します。コールバック・メソッドがフィルタで明示的に除外されている場合は，このオプションが有効になっていてもコールバックは除外されます。GUI の更新など，ビジネス・プロセスに直接関係のないアクションをコールバックが実行している場合は，このオプションを必ず無効にしてください。
- **［非同期コールバックを生成する］**：このオプションは，元のコールバックが記録されなかった場合に VuGen がコールバックを処理する方法を定義します。これは，上記のオプション（［**標準設定で元のコールバックを呼び出す**］）が無効になっているとき，あるいは，コールバックが明示的に除外されているときに関係してきます。

  このオプションを有効にすると，再生中に元のコールバックの代わりに呼び出されるダミー・メソッドが作成されます。このダミー・コールバックは，スクリプトの **callbacks.cs** セクションに生成されます。

  このオプションを無効にすると，コールバックの代わりに NULL 値が挿入され，イベントがその都度記録されます。

次のコードは，［**非同期コールバックを生成する**］が有効になっているときの Calculator クライアントのスクリプト生成を示しています。

```
lr.log("Event 2: CalculatorClient_1.Add(2, 3);");
Int32RetVal = CalculatorClient_1.Add(2, 3);
// Int32RetVal = 5;

callback_1 = new AsyncCallback(this.OnComplete1);
lr.log("Event 3: CalculatorClient_1.BeginAdd(2, 3, callback_1, null);");
IAsyncResult_1 = CalculatorClient_1.BeginAdd(2, 3, callback_1, null);
```

コールバック・メソッド OnComplete1 を表示するには，左側の表示枠で **callback.cs** ファイルをクリックします。

次のコードは，［非同期コールバックを生成する］が無効になっているときのスクリプト生成を示しています。コールバックの代わりに NULL が生成され，コールバック・イベントがその都度記録されています。

```
lr.log("Event 3: CalculatorClient_1.BeginAdd(2, 3, null, null);");
IAsyncResult_1 = CalculatorClient_1.BeginAdd(2, 3, null, null);

lr.log("Event 5: CalculatorClient_1.EndAdd(IAsyncResult_1);");
Int32RetVal = CalculatorClient_1.EndAdd(IAsyncResult_1);
// Int32RetVal = 5;

lr.log("Event 6: ((ManualResetEvent)(IAsyncResult_1.AsyncWaitHandle));");
ManualResetEvent_1 = ((ManualResetEvent)(IAsyncResult_1.AsyncWaitHandle));

lr.log("Event 7: ManualResetEvent_1.Close();");
ManualResetEvent_1.Close();
```

注：特定の記録オプションを指定してスクリプトを記録した後で，記録オプションを変更する場合，スクリプトを再度記録する必要はありません。代わりに，新しい設定でスクリプトを再生成します。

詳細については，387 ページの「［Microsoft .NET］の［記録］ノード」を参照してください。

接続プール

ADO.NET プロバイダは，負荷テストの精度に大きな影響を与える**接続プール**と呼ばれる機能をデプロイします。すべての仮想ユーザで使用されるアプリケーション・ドメインが 1 つだけのときは常に，接続プールは有効です。.NET Framework はデータベース接続を開いたまま維持し，新しい接続が要求された場合にそれらの再利用を試みます。多くの 仮想ユーザが 1 つのアプリケーション・ドメインで実行されるため，仮想ユーザが互いに干渉する可能性があります。この動作は直線的ではなく，そのことが 仮想ユーザの精度を下げる場合があります。

.NET 実行環境の設定では，［仮想ユーザごとにアプリケーション ドメイン］プロパティを使用して，各仮想ユーザを個別のアプリケーション・ドメインで実行できるようにします（標準設定では有効）。つまり，仮想ユーザごとに接続プールはあるけれども，仮想ユーザは互いに干渉しません。この設定により精度は向上しますが，スケーラビリティは低下します。

このオプションを無効にした場合，データベースの接続プールを手作業で無効にする必要があります。

次の表に，手作業で接続プールを無効にする方法を示します。

プロバイダ	オプション
.NET Framework Data Provider for SQL Server	"Pooling=false" または "Pooling=no"
.NET Framework Data Provider for Oracle	"Pooling=false" または "Pooling=no"
.NET Framework Data Provider for ODBC	接続プールは ODBC Driver Manager により管理されます。接続プールを有効または無効にするには，ODBC データ・ソース・アドミニストレータを使用します（［コントロール パネル］または［管理ツール］フォルダにあります）。［**接続プール**］タブでは，インストールされている ODBC ドライバごとに接続プール・パラメータを指定できます。
.NET Framework Data Provider for OLE DB	"OLE DB Services=-2"

プロバイダ	オプション
Oracle Data Provider for .NET	"pooling=false"
Adaptive Server Enterprise ADO.NET Data Provider	"Pooling=False"

Microsoft .NET スクリプトのデバッグ

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 仮想ユーザの場合には 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**］>［**ソリューションを 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。

Microsoft .NET フィルタの概要

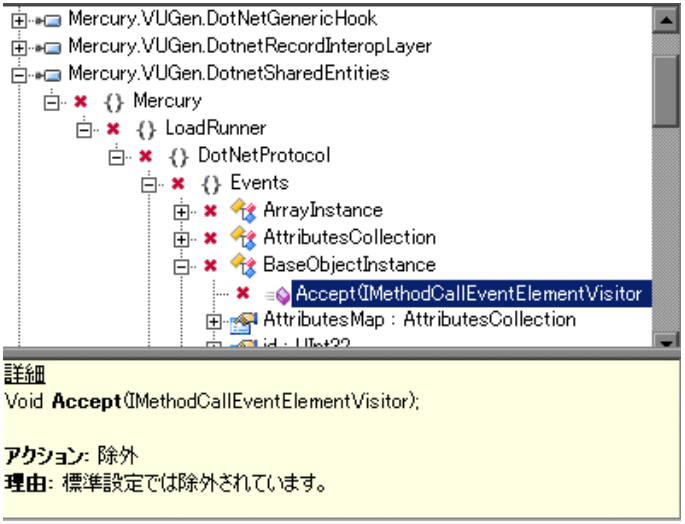
記録用のフィルタには，記録およびスクリプトの生成時に包含または除外するアセンブリ，インタフェース，名前空間，クラス，またはメソッドを指定します。

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。

- **Remoting**：.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。

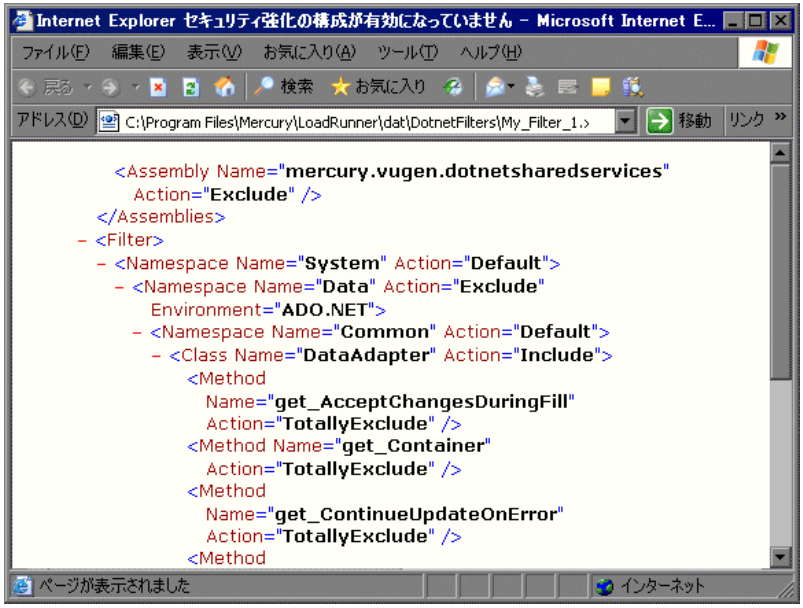
Microsoft .NET フィルタの詳細設定

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，382 ページの「フィルタ・マネージャ」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。

Microsoft .NET フィルタの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を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。この引数がシリアル化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**シリアル化**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ル化します。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ル化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ル化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，シリアル化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。引数がシリアル化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，382 ページの「フィルタ・マネージャ」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を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，771 ページの「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。詳細については，218 ページの「スクリプトを相関させる方法 - Microsoft .NET」を参照してください。

タスク

アプリケーションのセキュリティと権限を設定する方法

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。

Full Trust 権限を特定のフォルダに付与するには，次の手順で行います（Visual Studio がインストールされていない場合）。

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。

Full Trust 権限を特定のフォルダに付与するには，次の手順で行います（Visual Studio がインストールされている場合）。

1 ［.NET Configuration］の設定を開きます。［**スタート**］>［**すべてのプログラム**］>［**管理ツール**］>［**Microsoft .NET Framework 2.0 構成**］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完全パスを指定し（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**］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 仮想ユーザ・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。

リファレンス

トラブルシューティングと制限事項

このセクションでは，Microsoft .NET プロトコルのトラブルシューティングと制限事項について説明します。

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，仮想ユーザ・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。

27

Oracle NCA プロトコル

本章の内容

概念

タスク

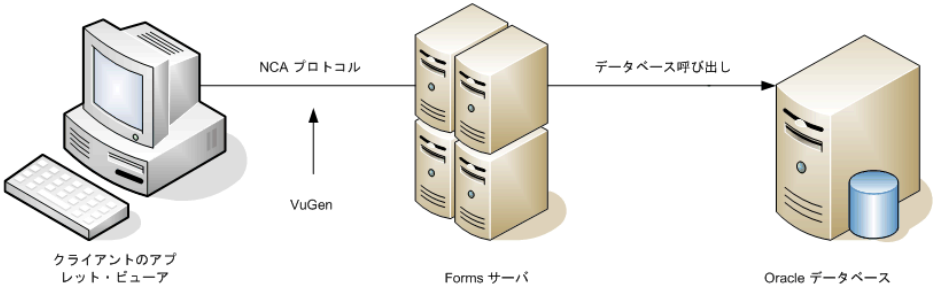
リファレンス

概念

Oracle NCA プロトコルの概要

Oracle NCA は，Oracle Forms サーバとの通信を処理するプロトコルです。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。

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。クライアント（アプレット・ビューア）は，独自の NCA プロトコルを使用して，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。

Oracle NCA プロトコルは，Web（HTTP/HTML）または Web（Click and Script）と組み合わせてマルチ・プロトコルとして使用することが一般的です。Oracle NCA で記録する場合，この方法をお勧めします。Oracle NCA をシングル・プロトコルとして使用する場合，Web イベントは記録されますが，標準設定ではステップは生成（再生）されません。

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，［記録オプション］の［プロトコル］ノードで指定します。

Oracle NCA プロトコルのスクリプト例

次の例では，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。

注：web_reg_save_param パラメータが自動的に生成されている場合，このパラメータは変更しないことをお勧めします。代わりに，新しい **web_reg_save_param** 関数や相関ルールを手作業で追加できます。

Oracle NCA の記録と再生のヒント

Oracle NCA 仮想ユーザ・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。

- スクリプトを記録する前に Jinitiator をインストールしておくことをお勧めします。
- 記録を開始する前にすべてのブラウザを閉じます。
- 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。
- **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，40 ページの「スクリプトのセクション」を参照してください。
- 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，**dat** > **mdrv** ディレクトリにある **mdrv_oracle_nca.dat** ファイルを次の例のように変更します。

```
[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。

プラグマ・モード

Oracle NCA 仮想ユーザ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，477 ページの「[Oracle NCA] > [クライアントのエミュレーション] ノード」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"
     ...
```

タスク

名前によるオブジェクトの記録を有効にする方法

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，反復ごとに異なる可能性があるため，オブジェクト ID を使用してスクリプトを記録すると，再生は失敗する場合があります。**nca_connect_server** 関数を調べれば，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えます。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。

- 786 ページの「スタートアップ HTML ファイル」
- 787 ページの「Forms 設定ファイル」
- 788 ページの「記録する URL」

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。

スタートアップ HTML ファイルを使用してオブジェクト名の記録を有効にするには，次の手順で行います。

1 次に示す行を変更して，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。

```
<PARAM name="serverArgs … fndnam=APPS">
```

2 次のように Oracle キー **record=names** を追加します。

```
<PARAM name="serverArgs … fndnam=APPS record=names">
```

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。

設定ファイルを使用してオブジェクト名の記録を有効にするには，次の手順で行います。

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 次に示す **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.htm 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。

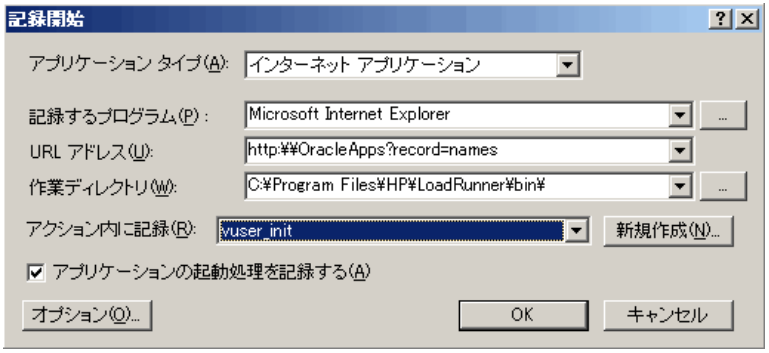
```
serverApp="%serverApp%"
logo="%logo%"
imageBase="%imageBase%"
formsMessageListener="%formsMessageListener%"
recordFileName="%recordFileName%"
```

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。

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。



［ホーム ページ (個人)］から Oracle アプリケーションを起動する方法

［**ホーム ページ (個人)**］からログインして Oracle Forms アプリケーション（バージョン 6i 以上）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，次の手順で行います。

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 ［**表示**］>［**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 **?**play=&record=names という文字列を追加します。

   ➤ 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 **&**play=&record=names という文字列を追加します。

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。

リファレンス

トラブルシューティングと制限事項

このセクションでは，Oracle NCA プロトコル・スクリプトのトラブルシューティングと制限事項について説明します。

セキュア Oracle NCA アプリケーションのテスト

- ［記録オプション］ダイアログ・ボックスの［ポート マッピング］ノードで，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
  ［**ターゲット 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
  ［**接続タイプ**］: 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。

  ［**サービス ID**］は，［**NCA**］ではなく［**HTTP**］です。

  詳細については，391 ページの「［ネットワーク］の［ポート マッピング］ノード」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。

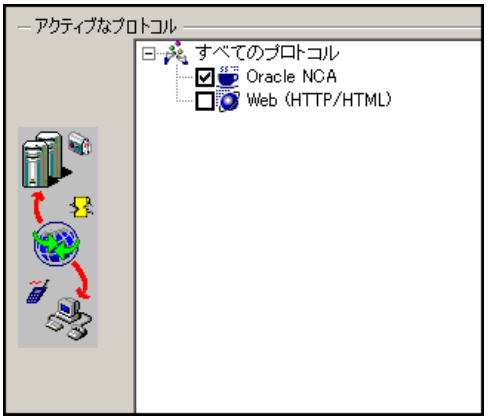
```
web_set_sockets_option("SSL_VERSION","3");
```

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，Oracle Apps IIi プロトコルを使用するか，Oracle NCA 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，スクリプトを記録した後で script ディレクトリの **default.cfg** ファイルを確認します。「**UseServletMode=**」というエントリを探します。

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（Forms Server 4.5 で作業していない場合）。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，仮想ユーザ・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・マッピングを定義する場合，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 10 章，「［ネットワーク］の［ポート マッピング］ノード」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，*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。

28

RDP プロトコル

本章の内容

概念

タスク

リファレンス

概念

RDP プロトコルの概要

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。

RDP の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。

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，40 ページの「スクリプトのセクション」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**］>［**すべてのプログラム**］>［**Microsoft Word**］の場合は，「**プログラム**」という語をクリックします。

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，*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。

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);
```

画像の同期化の概要

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

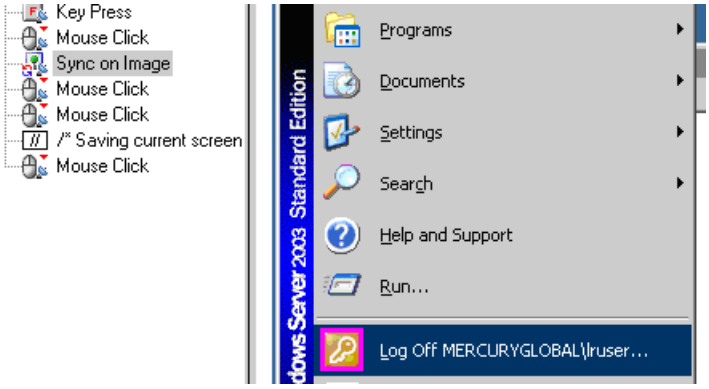
画像の同期化を効果的に行うために，次のガイドラインを参考にしてください。

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。

画像の同期化 - 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。

RDP エージェント（Microsoft Terminal Server エージェント）の概要

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。このエージェントは，より直感的で読みやすいスクリプト，組み込み同期化，関連オブジェクトに関する詳細情報を提供します。また，エージェントのインストールされた RDP 仮想ユーザを実行すると，各仮想ユーザによって lrrdpagent.exe の独自のプロセスが実行されます。その結果，サーバ・マシンで実行できる仮想ユーザの数がわずかに減少します。

Microsoft Terminal Server エージェントにより，通常の RDP 機能が次のように強化されます。

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**」および「**マウス・ダブルクリック**」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・アクションを示しています。エージェントを使用する場合，VuGen はすべてのマウス・アクションに対応する 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・ボックスがフォーカスを得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** ステップを手動で追加した場合，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。

タスク

RDP エージェントのインストール / アンインストール方法

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。

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する必要があります。

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（後述のアンインストール手順を参照してください）。

このタスクでは，次の手順を実行します。

➤ 806 ページの「RDP エージェントをインストールする」

➤ 807 ページの「RDP エージェントをアンインストールする」

RDP エージェントをインストールする

1 ソフトウェアをインストールするのにサーバの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 Windows 2003 で稼動しているマシンにエージェントをインストールするのに RDP（リモート・デスクトップ接続）を使用している場合は，インストールを開始する前に目的のマシンで次のコマンドを実行してください。

```
Change user /install
```

3 LoadRunner DVD の **Additional Components¥ Agent for Microsoft Terminal Server** ディレクトリから **Setup.exe** インストール・ファイルを見つけます。

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，仮想ユーザ・スクリプトを記録する前に記録オプションを設定する必要があります。［記録開始］ダイアログ・ボックスで［オプション］をクリックします。［高度なコード生成オプション］ノードで［**RDP エージェントの使用法**］をチェックします。

RDP エージェントをアンインストールする

1 ソフトウェアを削除するのにサーバの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 ［**コントロール パネル**］>［**プログラムの追加と削除**］で「HP Software Agent for Microsoft Terminal Server」を選択し，［**変更**］と［**削除**］をクリックします。

リファレンス

RDP ユーザ・インタフェース

このセクションの内容

［画像の同期化に失敗］ダイアログ・ボックス

同期化が失敗すると，このダイアログ・ボックスが開きます。このダイアログ・ボックスでは，スクリプトを停止したり，エラーに関係なくスクリプトを続行したりできます。

利用方法	同期化が失敗すると，自動的に開きます。
重要情報	**このダイアログ・ボックスにはいくつかの異なるフォームがあります。そのフォームは同期の失敗原因によって決まります。** ➤［**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。 ➤［**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 - 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。ただし，画像の同期化の許容レベルが緩やかな場合は，画像を発見できていることがあります。 ➤［**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い場合は，［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。 ➤**指定なし**：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い場合は，［画像の同期化に失敗］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。
関連項目	800 ページの「画像の同期化の概要」

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
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	このボタンでは，ダイアログ・ボックスの種類によって異なるアクションが実行されます。 ［**スナップショットを追加する**］：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，仮想ユーザは失敗しません。この場合，VuGen によって新しい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。 ［**誤差許容レベルを下げる**］：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。 ［**誤差許容レベルを上げる**］：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。 **指定なし**：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。 **注意：**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，［**実行環境の設定**］>［**RDP**］>［**同期化**］ノードで［**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。

トラブルシューティングと制限事項

このセクションでは，Microsoft Terminal Server エージェントを使用する RDP スクリプトのトラブルシューティング情報について説明します。

rdp_sync_object_mouse_click/double_click ステップで再生が失敗する

特定の **rdp_sync_object_mouse_click** または **rdp_sync_object_mouse_double_click** ステップで再生が失敗する場合，この問題を解決する回避策があります。次に示す順序で回避策を試すことをお勧めします。

回避策：RDPAgentCodeGen.cfg ファイルを変更する

特定のウィンドウで発生する各 **rdp_sync_object_mouse_click/double_click** ステップのスクリプトが次に生成されるときに **rdp_sync_on_image** と **rdp_mouse_click** のステップが VuGen によって自動的に作成されるように，**RDPAgentCodeGen.cfg** ファイルを設定できます。これを行うには，ウィンドウの名前を指定し，このプロセスの発生元のウィンドウの合計数をカウントする変数を更新して，スクリプトを再生成します。

RDPAgentCodeGen.cfg ファイルを変更するには，次の手順を実行します。

1. **Script Directory > data** ディレクトリで **RDPAgentCodeGen.cfg ファイル**を開きます。
2. スクリプトをツリー・ビューで開き，失敗したステップをダブルクリックします。
3. ウィンドウの名前をコピーします
4. **RDPAgentCodeGen.cfg** ファイル内の **NumberOfTitles** の値に 1 を加えます。

5 次の行を追加します。

```
WindowTitleX=< ウィンドウの名前 >
```

ここで，**X** は **NumberOfTitles** の新しい値です。

6 スクリプトを再生成します。

注：RDPAgentCodeGen.cfg ファイルを使用して，別の方法にも指定されている **rdp_sync_object_mouse_click/double_click** ステップの場合と似た方法で，**rdp_sync_on_image** ステップと **rdp_mouse_click** ステップを自動的に生成できます。ステップはコントロールのクラス属性に基づいて対象設定できます。詳細については，HP Software のサポートに問い合わせてください。

回避策：新しいステップを手動で挿入する

失敗した各ステップに対して **rdp_sync_on_image** ステップと **rdp_mouse_click** ステップを挿入することで，回避策を手動で実行できます。この方法で追加したステップはスクリプトが再生成されると失われるため，この回避策を使用することはお勧めしません。

29

RTE プロトコル

本章の内容

概念

タスク

概念

RTE プロトコルの概要

RTE 仮想ユーザは，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 仮想ユーザを使用して，この事例をエミュレートできます。RTE 仮想ユーザ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。

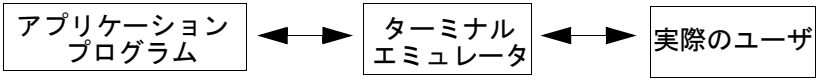
6 「Q」と入力してアプリケーション・プログラムを閉じます。

RTE 仮想ユーザ・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，それらを仮想ユーザ・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，823 ページの「RTE 同期の概要」を参照してください。

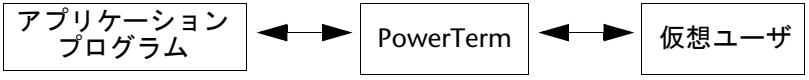
ターミナルとサーバの通信をエミュレートするために開発された関数を，TE 仮想ユーザ 関数と呼びます。TE 仮想ユーザ 関数の名前には，**TE** というプレフィックスが付きます。VuGen は，RTE セッション中に，本項に列挙する TE 関数のほとんどを自動的に記録します。また，手動でスクリプトに任意の関数をプログラミングすることもできます。

TE 関数の構文と使用例については，*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。

RTE 仮想ユーザは，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 仮想ユーザ環境では，仮想ユーザが実際のユーザの代わりに操作を行います。仮想ユーザは 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。

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。

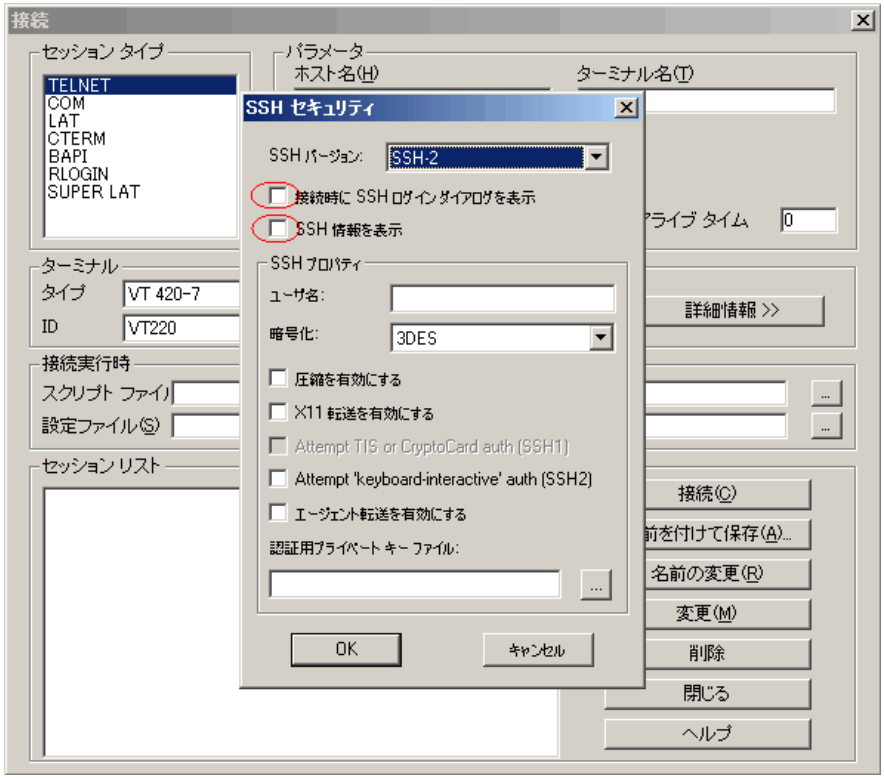
Ericom サポートでは，記録と再生中にエスケープ・シーケンスを扱えます。Ericom PowerTerm を使うと，PC キーをユーザ定義エスケープ・シーケンスに割り当てられます。割り当ての詳細については，PowerTerm のヘルプを参照してください。

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，*オンライン関数リファレンス*（[**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，［**詳細情報**］ボタンをクリックします。



ターミナル・エミュレータへの入力

2 つの TE 仮想ユーザ関数を使用すると，仮想ユーザが 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，次を参照してください。
- **TE_typing_style** は，仮想ユーザの入力速度を決めます。**TE_typing_style** 関数を仮想ユーザ・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，819 ページの「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 11 章，「実行環境の設定」を参照してください。

注：RTE 仮想ユーザ・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，*オンライン関数リファレンス*（[**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，仮想ユーザが **TE_type** ステートメントを送信しようとした場合，その仮想ユーザは 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。

仮想ユーザ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，仮想ユーザ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，仮想ユーザに対して 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 モードにある間も仮想ユーザ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，*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。

入力スタイルの設定

RTE 仮想ユーザに対しては，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，仮想ユーザからターミナル・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仮想ユーザが一時停止します。このため，人間のユーザによるキーボードでの入力が，仮想ユーザによって，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");
```

上記の場合は，仮想ユーザは文字 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，*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。

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 11 章，「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** 関数から仮想ユーザごとに 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 11 章,「実行環境の設定」を参照してください。

TE_connect 関数が仮想ユーザ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** 関数を仮想ユーザ・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。

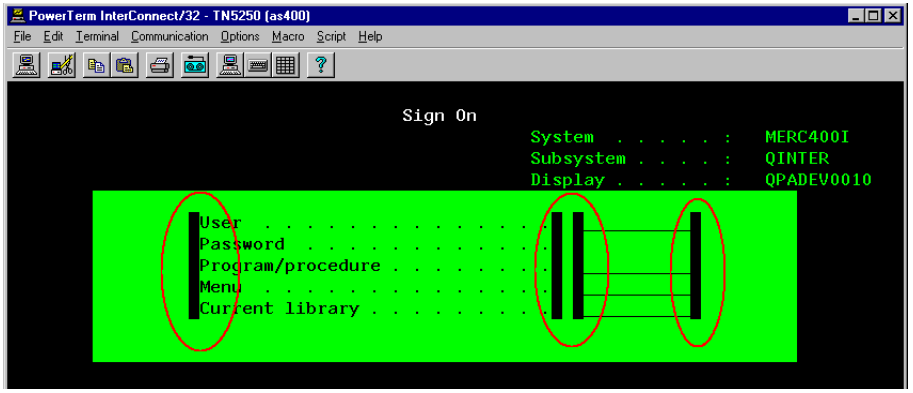
RteGenerateDeviceName 関数が仮想ユーザ・スクリプト内に存在する場合は，新しいデバイス名が必要になるたびに，仮想ユーザからこの関数が呼び出されます。**RteGenerateDeviceName** 関数がスクリプト内で定義されておらず，一意のデバイス名が必要になった場合は，**TE_connect** 関数によって必要な名前が生成されます。

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の設定

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。

ターミナル画面からのテキストの読み取り

RTE 仮想ユーザ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仮想ユーザ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 仮想ユーザ・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，*オンライン関数リファレンス*を参照してください。

TE_find_text ステートメントは，仮想ユーザ・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，*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。

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，仮想ユーザ・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，*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。

RTE 同期の概要

テスト対象のシステムによっては，仮想ユーザがターミナル・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仮想ユーザを設定します。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。

仮想ユーザの入力は，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 仮想ユーザ・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：仮想ユーザ・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。仮想ユーザ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 仮想ユーザ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。

仮想ユーザ・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，829 ページの「画面上のテキスト表示の待機」，および*オンライン関数リファレンス*（[**ヘルプ**] > [**関数リファレンス**]）参照してください。

次のスクリプト・セグメントでは，仮想ユーザが「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。仮想ユーザは，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，仮想ユーザ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，仮想ユーザ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，次の手順で行います。

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 ［実行環境の設定］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。

仮想ユーザは，**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，仮想ユーザ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，仮想ユーザがスクリプトの実行を再開できるようになります。仮想ユーザが 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。仮想ユーザ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，次の手順で行います。

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，*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。

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，次の手順で行います。

1 ［**ツール**］>［**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 仮想ユーザ・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仮想ユーザに指示します。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，*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。

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，次の手順で行います。

1 [**ツール**] > [**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 仮想ユーザ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，*オンライン関数リファレンス*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，*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。

次の例では，仮想ユーザ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，次の手順で行います。

1 ［**ツール**］>［**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，仮想ユーザはターミナル・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，*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。

タスク

ターミナル・キーを PC キーボード・キーに割り当てる方法

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，次の手順で行います。

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。

RTE スクリプトを記録する方法

VuGen を使用して，Windows ベースの RTE 仮想ユーザ・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。

このタスクでは，RTE スクリプトを記録する方法について説明します。この手順は，第 3 章，「記録」で説明した一般的な記録の手順とは異なります。

このタスクでは，次の手順を実行します。

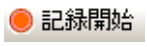
➤ 834 ページの「ターミナルの設定と接続を記録する」
➤ 837 ページの「典型的なユーザ・アクションを記録する」
➤ 837 ページの「ログオフ手順を記録する」

1 ターミナルの設定と接続を記録する

a 既存の RTE 仮想ユーザ・スクリプトを開くか，新しい RTE 仮想ユーザ・スクリプトを作成します。

b ［**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 **vuser_init** セクションを選択します。

c 仮想ユーザ・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。



d ［**記録開始**］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。

e PowerTerm のメニュー・バーから［**ターミナル**］>［**設定**］を選択し，［ターミナル設定］ダイアログ・ボックスを表示します。

f 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。

---

g ［**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。

h ［**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。

i ［**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。

---

j ［**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：仮想ユーザ・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。

---

2 典型的なユーザ・アクションを記録する

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仮想ユーザ・スクリプトの **Actions** セクションに記録します。仮想ユーザ・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。

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。

a ［**セクション**］ボックスで［**アクション**］セクションを選択します。

b 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，それらを仮想ユーザ・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。

3 ログオフ手順を記録する

a 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。

b 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。

c 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。

d 記録ツールバーの［**記録停止**］ボタンをクリックします。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。

e ［**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。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。

エラー時の処理継続を実装する方法

RTE スクリプトで，エラー時の処理継続機能を設定するには，次の手順で行います。

- エラーが発生してもスクリプトの実行を継続するには，次の関数を挿入します。

  TE_setvar(TE_IGNORE_ERRORS, 1)

- エラーが発生するとスクリプトが失敗する標準の動作に戻すには，次の関数を挿入します。

  TE_setvar(TE_IGNORE_ERRORS, 0)

30

SAP プロトコル

本章の内容

概念

タスク

リファレンス

概念

SAP プロトコル・タイプの選択

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 仮想ユーザ・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）または SAP-Web プロトコルを使用します。

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，SAPGUI および SAP-Web プロトコルを使用してマルチ・プロトコルの仮想ユーザ・スクリプトを作成します。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。これは，SAPGUI プロトコルと Web プロトコルを組み合わせようとすると，うまくいきません。

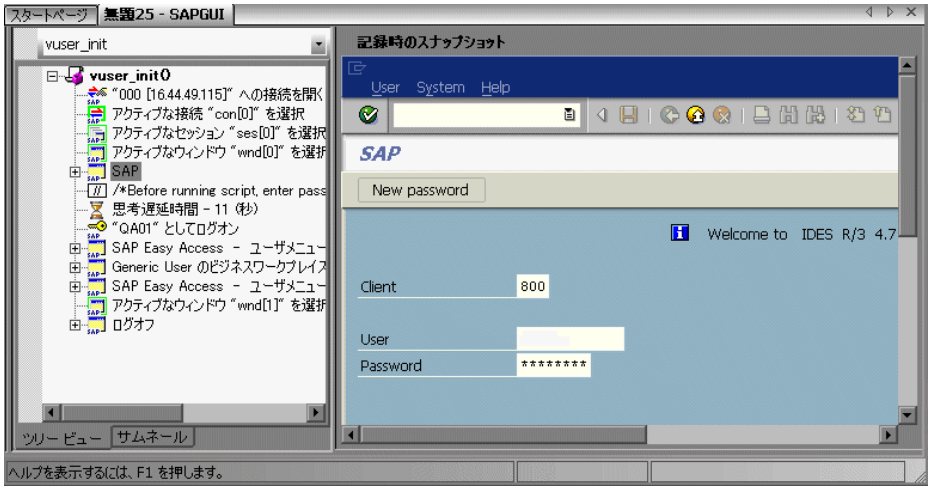
セッションを記録する前に，モジュールとクライアント・インタフェースが VuGen によってサポートされていることを確認します。次の表に，SAP ビジネス・アプリケーションおよび関連ツールの SAP のクライアント・モジュールを示します。

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 プロトコルを使用します。
SAPGUI for Windows	SAPGUI プロトコルを使用します。また，APO モジュールの記録もサポートします（SAP 6.20 の 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。

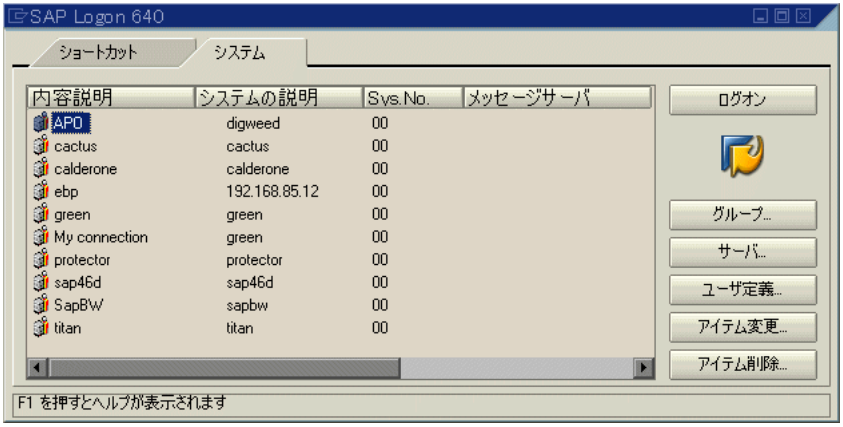
SAPGUI プロトコル

通常 SAPGUI 仮想ユーザ・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・アクションが仮想ユーザ・スクリプトのステップとして表示されます。

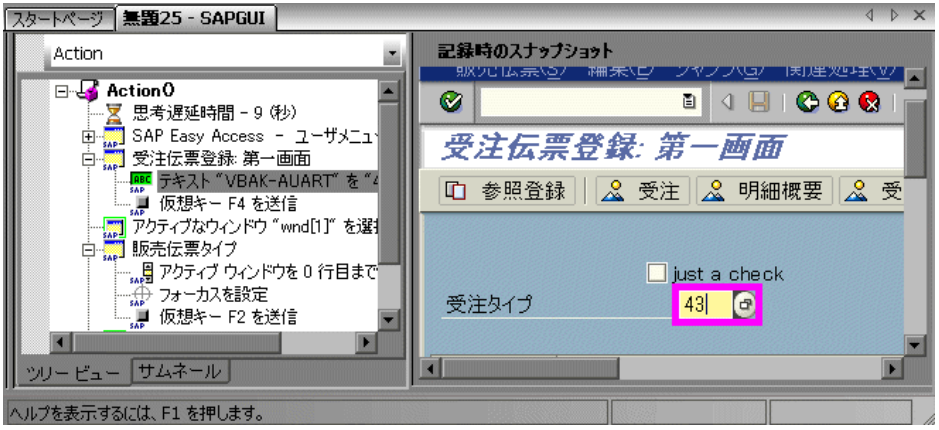
次の例は，SAPGUI クライアントの典型的な記録を示しています。最初のセクションである **vuser_init** には，接続の開始とログインが含まれます。



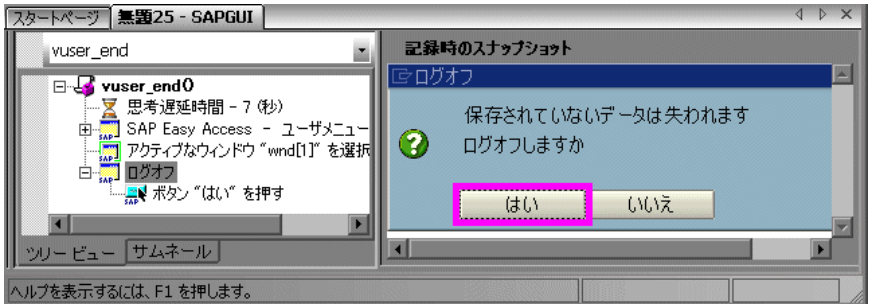
［接続を開く］ステップは，［SAP Logon］の［**詳細**］リストにある接続名の 1 つを使用します。指定された名前がリストにない場合，仮想ユーザはその名前のサーバを検索します。



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);
```

SAP Web プロトコル

通常 SAP-Web 仮想ユーザ・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，*オンライン関数リファレンス*（[**ヘルプ**] > [**関数リファレンス**]）の「Web 関数」を参照してください。

例：

SAP Portal クライアントにおける典型的な記録の例を次に示します。

```
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 SAP Web および SAPGUI マルチ・プロトコル記録を示します。**web_xxx** 関数から **sapgui_xxx** 関数に切り替わることに注意してください。

```
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&C
ompFName=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（Click and Script）プロトコル

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。

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。

例：

次の例では，ユーザは［**User Profile**］タブを選択しました。VuGen は **sap_portal** 関数を生成しています。

```
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。

SAPGUI のオプションのウィンドウの再生

SAPGUI 仮想ユーザ・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。仮想ユーザは，［**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，仮想ユーザは次の［**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，仮想ユーザ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。

タスク

SAP 環境を設定する方法

このタスクでは，VuGen で使用する SAP 環境を設定および確認する方法について説明します。

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，仮想ユーザは 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。

このタスクでは，次の手順を実行します。

- ➤ 851 ページの「Windows クライアントのパッチ・レベルの SAPGUI の確認」
- ➤ 851 ページの「カーネル・パッチ・レベルを確認する」
- ➤ 852 ページの「R/3 サポート・パッケージを確認する」
- ➤ 854 ページの「SAP Application Server のスクリプティングを有効にする」
- ➤ 856 ページの「SAPGUI 6.20 Client でのスクリプティングを有効にする」

Windows クライアントのパッチ・レベルの SAPGUI 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，次の手順で行います。

1 SAPGUI ログオン・ウィンドウを開きます。［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。

カーネル・パッチ・レベルを確認する

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 ［**ほかのカーネル情報**］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。

レベルは，使用している SAP バージョンに応じて，次の表にあるレベルより高い必要があります。

ソフトウェア・コンポーネント	SAP リリース	カーネル・パッチ・レベル
SAP_APPL	31I	Kernel 3.1I レベル 650
SAP_APPL	40B	Kernel 4.0B レベル 903
SAP_APPL	45B	Kernel 4.5B レベル 753
SAP_BASIS	46B	Kernel 4.6D レベル 948

ソフトウェア・コンポーネント	SAP リリース	カーネル・パッチ・レベル
SAP_BASIS	46C	Kernel 4.6D レベル 948
SAP_BASIS	46D	Kernel 4.6D レベル 948
SAP_BASIS	610	Kernel 6.10 レベル 360

R/3 サポート・パッケージを確認する

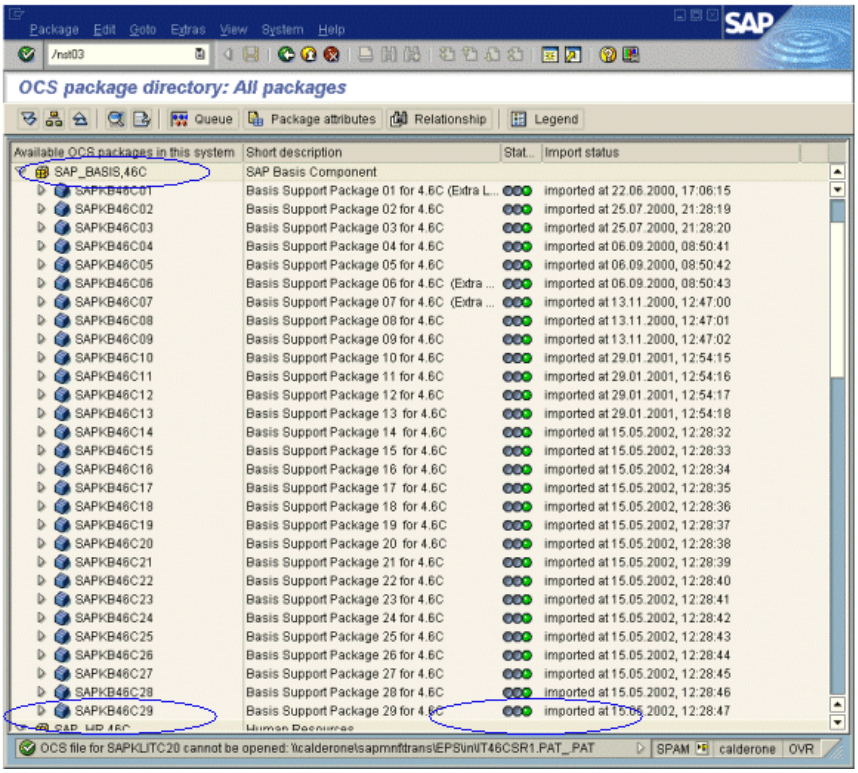
1 SAP システムにログインし，SPAM トランザクションを実行します。

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 次の表に従って，使用している SAP のバージョンに適切なパッケージがインストールされていることを確認します。

ソフトウェア・コンポーネント	リリース	パッケージ名
SAP_APPL	31I	SAPKH31I96
SAP_APPL	40B	SAPKH40B71
SAP_APPL	45B	SAPKH45B49
SAP_BASIS	46B	SAPKB46B37
SAP_BASIS	46C	SAPKB46C29
SAP_BASIS	46D	SAPKB46D17
SAP_BASIS	610	SAPKB61012

適切なバージョンが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。



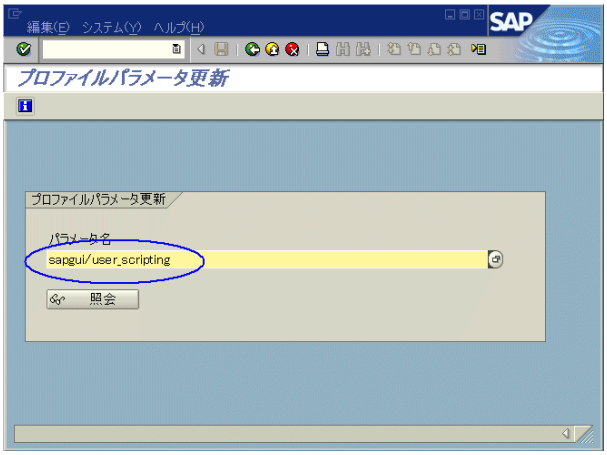
OCS パッケージがインストールされていない場合は，www.sap.com Web サイトからダウンロードしてインストールします。詳細については，『SAP OSS note # 480149』を参照してください。

SAP Application Server のスクリプティングを有効にする

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。

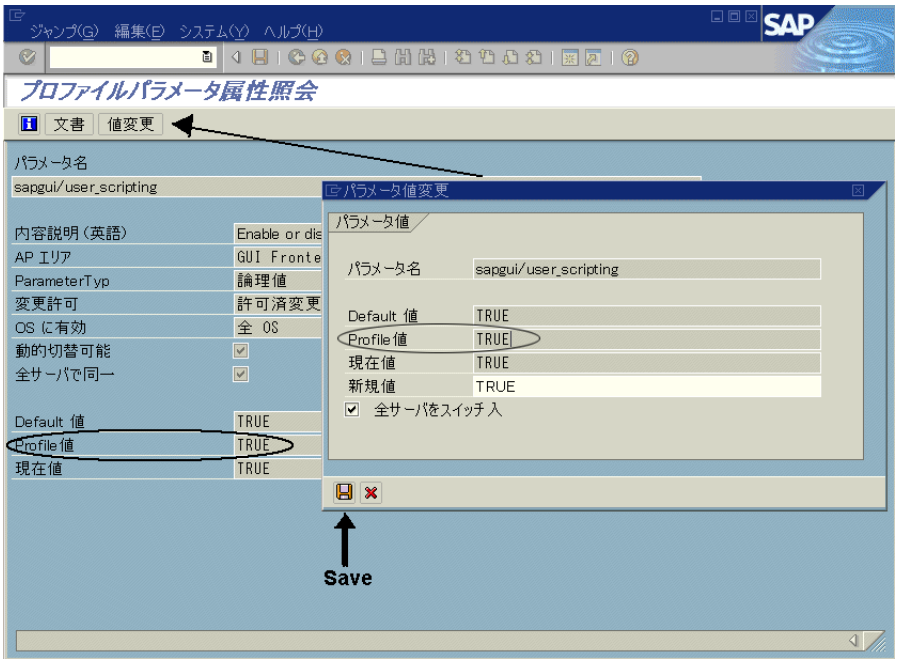
プロファイル・パラメータを変更するには，次の手順で行います。

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，前述の手順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，前述の手順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。

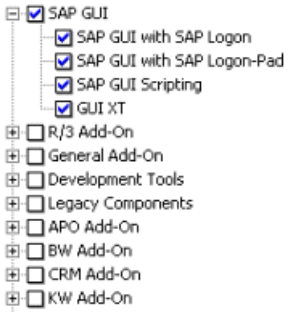
リリース	カーネル・バージョン	パッチ・レベル
4.6B，4.6C，4.6D	4.6D	972
6.10	6.10	391
6.20	すべてのバージョン	すべてのレベル

SAPGUI 6.20 Client でのスクリプティングを有効にする

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，次の手順で行います。

1 **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。



2 **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。［**Scripting**］タブを選択し，次のオプションをクリアします。

- **［スクリプトが実行中の GUI に接続するときに通知］**
- **［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 スクリプトを記録する方法

次の手順では，SAPGUI スクリプトを記録するための前提条件について説明します。

- 857 ページの「マルチ・プロトコルを使用して記録する場合に SAPLogon アプリケーションを終了する」
- 857 ページの「F1 ヘルプに対応したモーダル・ダイアログ・ボックスを使用する」
- 858 ページの「F4 ヘルプに対応したモーダル・ダイアログ・ボックスを使用する」

マルチ・プロトコルを使用して記録する場合に SAPLogon アプリケーションを終了する

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 ヘルプに対応したモーダル・ダイアログ・ボックスを使用する

次のように，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 クライアントに指示します。

1 ［**ヘルプ**］>［**設定**］を選択します。

2 ［**F1 Help**］タブをクリックします。

3 ［照会］セクションの［**Modal ダイアログ ボックス**］を選択します。

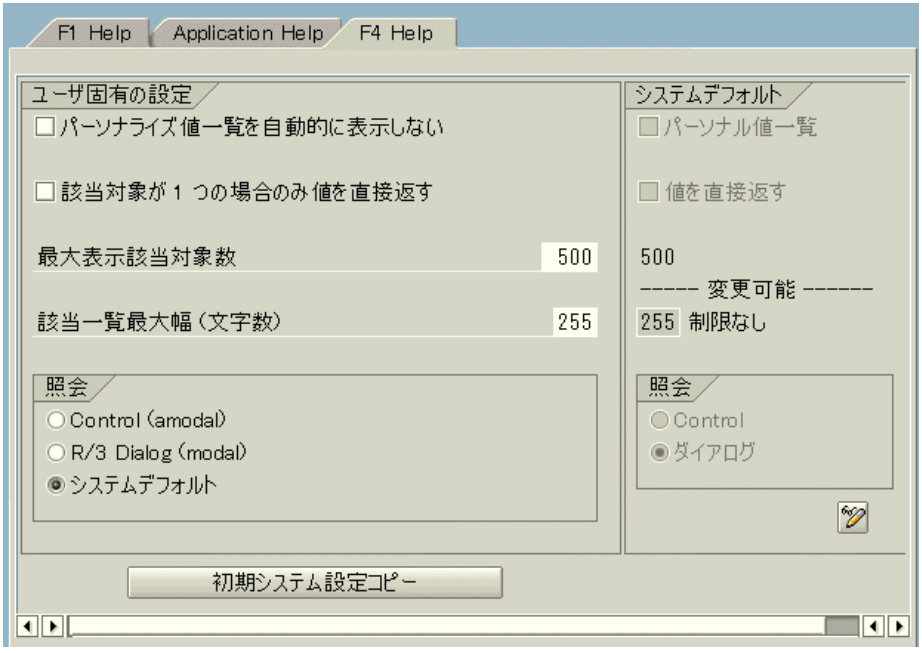
F4 ヘルプに対応したモーダル・ダイアログ・ボックスを使用する

注：この手順は，管理者のみが実行できます。

次のように，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 クライアントに指示します。

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］部分で，［**ダイアログ**］を選択します。

5 ［**初期システム設定コピー**］をクリックして変更を保存します。

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，SAP Management Console を使ってサービスを再起動します。

SAPGUI スクリプトを再生する方法

次の手順では，SAPGUI スクリプトを再生するための前提条件について説明します。

- 859 ページの「暗号化されたパスワードを置き換える」
- 860 ページの「再生中に SAPGUI ユーザ・インタフェースを表示する（任意）」

暗号化されたパスワードを置き換える

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。これは，次のユーザ名の後の関数の 2 番目の引数です。

```
sapgui_logon("user", "pswd", "800", "EN");
```

セキュリティを強化するには，コード内のパスワードを暗号化できます。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。次のように，**lr_decrypt** 関数がパスワードの位置に挿入されます。

```
sapgui_logon("user", lr_decrypt("3ea037b758"), "800", "EN");
```

再生中に SAPGUI ユーザ・インタフェースを表示する（任意）

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。［**仮想ユーザ**］>［**実行環境の設定**］>［**SAPGUI**］>［**一般**］ノードを選択し，［**再生時に SAP クライアントを表示する**］を選択します。複数の仮想ユーザを実行すると 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 SAPGUI スクリプトを実行する方法

次の手順では，シナリオ内で SAPGUI スクリプトを再生するためのヒントについて説明します。

➤ 860 ページの「LoadRunner コントローラの設定」

➤ 860 ページの「Process モードでエージェントが実行されていることを確認する」

LoadRunner コントローラの設定

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 ごとユーザ数**：512 MB のメモリを搭載したマシンの場合，［Load Generators］ダイアログ・ボックスで仮想ユーザ数を 50 に設定します。

Process モードでエージェントが実行されていることを確認する

Process 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，次の手順で行います。

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 仮想ユーザ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，実行できる仮想ユーザの数が限られる可能性があります。各マシンの仮想ユーザ数を増やすには，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**］>［**すべてのプログラム**］>［**< 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 スクリプトを拡張する方法

次の手順では，SAPGUI スクリプトを拡張するために使用できる追加オプションについて説明します。

➤ 862 ページの「カーソル位置で記録する」

➤ 863 ページの「SAPGUI スクリプトにステップを対話的に挿入する」

➤ 866 ページの「検証関数を追加する」

➤ 867 ページの「情報を取得する」

➤ 869 ページの「日付情報を保存する」

カーソル位置で記録する

VuGen では，新しいアクションを挿入または既存のアクションを置換して，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。

➤ 記録中にアクションを誤った場合。

➤ アクションは正しくても，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。

カーソルで記録するには，次の手順で行います。

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 [**カーソル位置から記録**] 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。

a ［**アクションにステップを挿入する**］を選択すると，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が挿入されます。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。

b ［**スクリプトの残りを上書きする**］を選択すると，カーソル位置から後のすべてのステップが置換されます。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。

4 ［**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。

5 ［記録］フローティング・ツールバーが開いたら，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。

6 記録セッションを終了するには，フローティング・ツールバーの［**停止**］ボタンをクリックします。

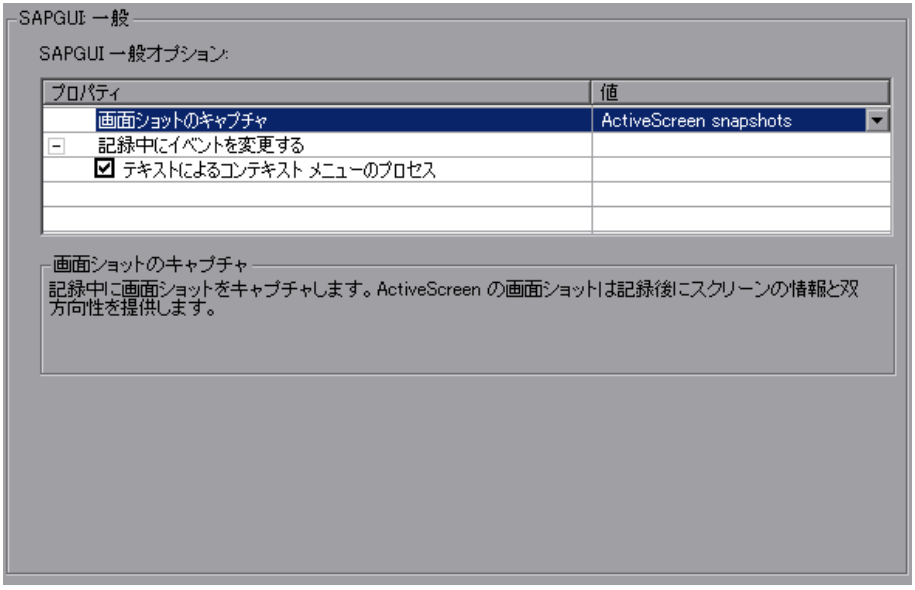
SAPGUI スクリプトにステップを対話的に挿入する

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。新しい関数を手作業で追加するだけでなく，Citrix 仮想ユーザ 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］の［一般］ノードの中で Active Screen スナップショットを無効にしていない場合）。

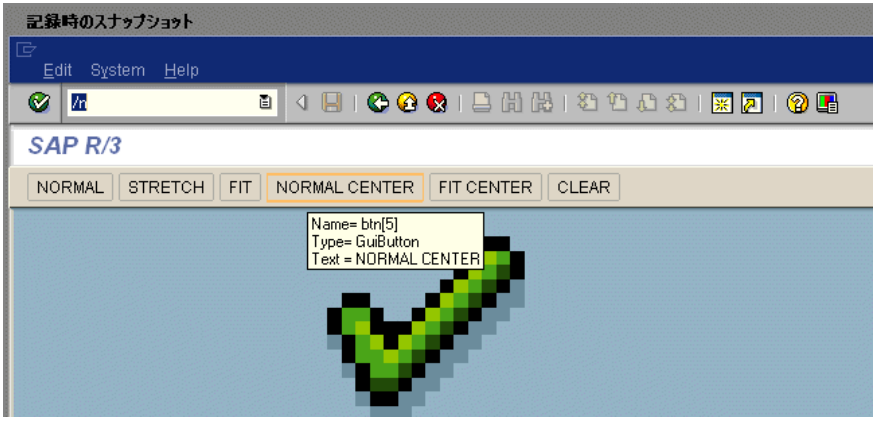
特定のオブジェクトにステップを対話的に挿入するには，次の手順で行います。

1 ［記録オプション］の［SAPGUI］の［一般］ノードで Active Screen スナップショットが選択されたときにスクリプトを記録したことを確認します（標準設定では有効）。



2 ［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。

3 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。



4 右クリックで表示されるメニューから［**新規ステップを挿入**］を選択します。［ステップの挿入］ボックスが開きます。

5 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。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。

6 ［**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。

注：特定の場所に貼り付けるオブジェクトの Control ID を取得できます。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。［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。

検証関数を追加する

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，仮想ユーザ・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，仮想ユーザ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);
}
```

情報を取得する

SAPGUI 仮想ユーザ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。

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。

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，仮想ユーザがステータス・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"
```

日付情報を保存する

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);
```

リファレンス

その他の SAP リソース

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** - 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** - https://websmp104.sap-ag.de/patches

SAP GUI for Windows - SAPGUI 6.20 パッチ（パッチ・レベル 32 以上）

トラブルシューティングと制限事項

このセクションでは，SAPGUI，SAP Web，および SAP（Click and Script）プロトコルのトラブルシューティングと制限事項について説明します。

SAPGUI 仮想ユーザ・スクリプトのトラブルシューティング

疑問 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，859 ページの「SAPGUI スクリプトを再生する方法」を参照してください。

疑問 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。

疑問 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。850 ページの「SAP 環境を設定する方法」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。

疑問 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，866 ページの「検証関数を追加する」を参照してください。

疑問 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，［詳細オプション］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。

疑問 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。

疑問 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。

疑問 8：［記録オプション］の［自動ログオン］ノードに移動すると，サーバ名のリストが空なのはなぜですか？

回答：これは，SAPGUI Client 7.20 を使用しているときに起こる場合があります。この問題を解決するには，**%APPDATA%¥SAP¥Common** から **saplogon.ini** ファイルをコピーします。**%APPDATA%** は，ユーザ・プロファイル・ディレクトリの直下にある Application Data ディレクトリを指定する環境変数を表します。このファイルを %WINDIR% ディレクトリ（C:¥Windows）に貼り付けます。

31

Siebel Web プロトコル

本章の内容

概念

タスク

リファレンス

概念

Siebel Web プロトコルの概要

Siebel-Web プロトコルは標準の Web 仮想ユーザに似ていますが，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。

Siebel Web の記録オプションと実行環境の設定

Siebel Web 仮想ユーザを記録する前に，記録オプションを次のように設定することをお勧めします。

➤［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］：［**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

Siebel Web 仮想ユーザを再生または負荷テストを行う前に，実行環境を次のように設定することをお勧めします。

➤ 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。

タスク

トランザクションのブレークダウン情報を記録する方法

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ステップを追加することをお勧めします。思考遅延時間ステップの追加の詳細については，163 ページの「ステップをスクリプトに挿入する方法」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，次の手順で行います。

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仮想ユーザ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。

リファレンス

トラブルシューティングと制限事項

このセクションでは，Siebel Web 仮想ユーザ・スクリプトのトラブルシューティングと制約事項について説明します。

「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，または正しく相関されてい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。

32

SilverLight プロトコル

本章の内容

概念

タスク

概念

Silverlight プロトコルの概要

Microsoft Silverlight は，グラフィック，アニメーション，および双方向性をサポートする Web アプリケーション・フレームワークです。VuGen の Silverlight プロトコルでは，Microsoft Silverlight で構築されたアプリケーションを記録できます。VuGen の Silverlight プロトコルには，サブセットとして Web（HTTP/HTML）プロトコルのほかに，多数の新しい関数，記録オプション，および実行環境の設定が含まれます。

高レベルのスクリプトを記録するために，記録オプションで，アプリケーションによって使用される WSDL ファイルをインポートできます。

タスク

WSDL ファイルをインポートする方法

次の手順では，Silverlight スクリプトに手動または自動的に WSDL ファイルをインポートする方法について説明します。また，WSDL ファイルを無効にし，SOAP 要求を生成することもできます。これらのオプションはすべて［**記録オプション**］ダイアログ・ボックスの［**Silverlight**］>［**サービス**］ノードで実行します。ユーザ・インタフェースの詳細については，412 ページの「Silverlight サービス・ノード」を参照してください。

➤ 883 ページの「WSDL ファイルを自動的に指定する」

➤ 883 ページの「WSDL ファイルを手動で指定する」

➤ 884 ページの「WSDL ファイルを無効にする」

➤ 884 ページの「高度なセキュリティ設定」

WSDL ファイルを自動的に指定する

スクリプトで使用される WSDL ファイルが自動的に検出および指定されるように VuGen を設定するには，［**スクリプト内に含まれる WSDL ファイルを使用する**］と［**WSDL ファイルを自動的に検出してコード生成中にサービスをインポートする**］を選択します。インポートできない WSDL ファイルが検出された場合，［コード生成通知］ボックスで通知されます。

WSDL ファイルを手動で指定する

［サービスの追加］ダイアログ・ボックスで WSDL ファイルを手動で指定できます。その方法はいくつかあります。URL がわかっている WSDL ファイルを指定するには，［**URL**］オプションを使用します。WSDL ファイルがローカル・マシンにある場合は，［**ファイル**］オプションを使用します。WSDL ファイルの履歴（以前にインポートされた WSDL ファイルのリスト）にある WSDL ファイルを検索するには，［**インポート済み**］を選択し，［**...**］をクリックしてリストを開きます。

ユーザ・インタフェースの詳細については，413 ページの「［サービスの追加 / 編集］ダイアログ・ボックス」を参照してください。

WSDL ファイルを無効にする

WSDL ファイルを無効にし，代わりに SOAP 要求を生成できます。スクリプトのレベルは低くなりますが，スクリプトのパフォーマンスが向上します。WSDL ファイルを無効にするには，［**WSDL ファイルを使用しない**］を選択します。

高度なセキュリティ設定

セキュリティとパスワードの設定は［プロトコルおよびセキュリティ シナリオ データ］ダイアログ・ボックスで変更できます。詳細については，415 ページの「［プロトコルおよびセキュリティ シナリオ データ］ダイアログ・ボックス」を参照してください。

33

Tuxedo プロトコル

本章の内容

概念

リファレンス

概念

Tuxedo プロトコル - 概要

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 Tuxedo 関数は **lrt**，**tp**，**tx**，または **F** というプレフィックスで始まります。

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 スクリプトでの作業に関する注意事項

Tuxedo スクリプトを記録および実行する前に，次の重要な注意事項を確認する必要があります。

- Tuxedo 6.x 以前の記録では Tuxedo 6 プロトコルを使用し，Tuxedo 7.x 以上では Tuxedo プロトコルを使用することをお勧めします。
- 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 7.x で PeopleSoft-Tuxedo 仮想ユーザを実行するには，*mdrv.dat* ファイル内のライブラリ拡張子を次のように変更する必要があります。

```
[PeopleSoft-Tuxedo]
WINNT_EXT_LIBS=lrt7.dll
```

Tuxedo 仮想ユーザの環境設定の定義

次の項では，Windows および UNIX プラットフォームで動作する Tuxedo 仮想ユーザ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)
```

リファレンス

Tuxedo バッファ・データ

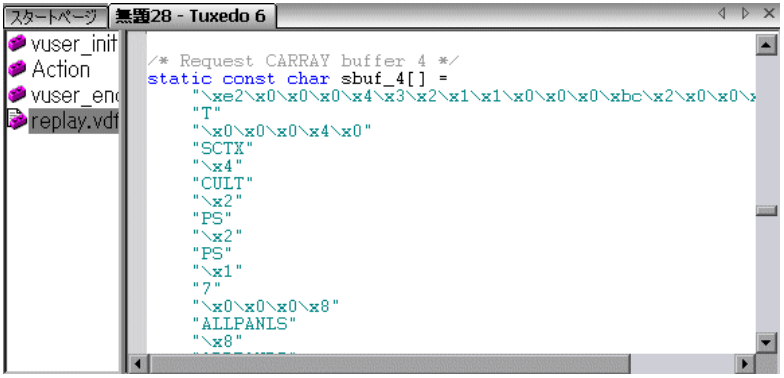
VuGen を使用して Tuxedo 仮想ユーザ・スクリプトを作成すると，アクションはスクリプトの 3 つのセクション **vuser_init**，**Actions**，**vuser_end** に記録されます。

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。



トラブルシューティングと制限事項

このセクションでは，Tuxedo および Tuxedo 6 の仮想ユーザのトラブルシューティングと制限事項について説明します。

- 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・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・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 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，890 ページの「Tuxedo バッファ・データ」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。

34

Web プロトコル

本章の内容

概念

タスク

リファレンス

概念

Web プロトコルの概要

VuGen を使用して，Web 仮想ユーザ・スクリプトを作成できます。標準的なユーザ・アクションを実行し 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仮想ユーザ・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，仮想ユーザ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。

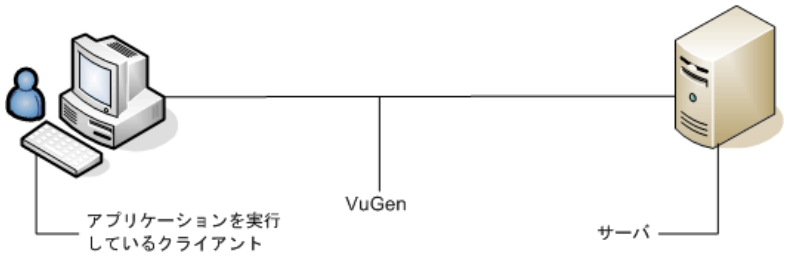
会社の製品情報を表示する Web サイトがあったとします。このサイトには，見込み顧客がアクセスします。このサイトでは，多数のユーザ（200 ユーザなど）が同時にサイトにアクセスしたときでも，顧客の問い合わせに対する応答時間が必ず指定値（20 秒など）未満になるようにします。そのために，仮想ユーザを使って，Web サーバが同時に複数の要求に対してサービスを提供する状況をエミュレートします。このとき各仮想ユーザは次のような操作を行うものと考えられます。

- 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

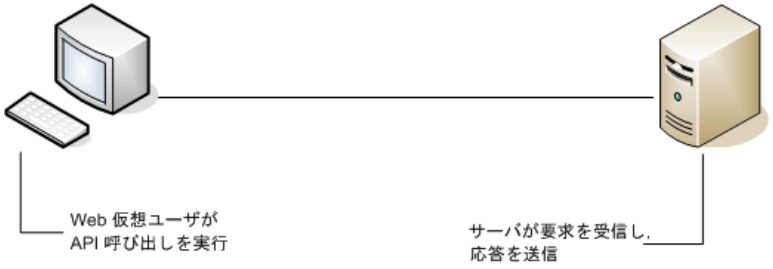
利用可能な複数のテスト用マシンに，数百の仮想ユーザを分散配置できます。各仮想ユーザでは API を通じてサーバにアクセスします。これにより，多数のユーザによる負荷の下でサーバのパフォーマンスを測定できます。

Web 仮想ユーザ・テクノロジ

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 仮想ユーザ・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。



記録された仮想ユーザ・スクリプトを実行するとき，仮想ユーザ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。仮想ユーザ・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 仮想ユーザ・タイプ

新しい Web 仮想ユーザ・スクリプトを作成する場合，次のいずれかのタイプの Web 仮想ユーザを選択できます。

Web （Click and Script）

Web （Click and Script）仮想ユーザは，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，編集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）仮想ユーザは，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）仮想ユーザ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，記録後に手動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）仮想ユーザでは，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［ログイン］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）仮想ユーザは，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/HTML）

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）仮想ユーザには，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**］です。これらのレベルにより，仮想ユーザ・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，319 ページの「記録レベルの概要」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）仮想ユーザを使用します。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）仮想ユーザを使用します。

プッシュ技術に対するサポート

仮想ユーザ・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。

仮想ユーザ・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，仮想ユーザ・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，*オンライン関数リファレンス*（[**ヘルプ**] > [**関数リファレンス**]）の **web_reg_cross_step_download** 関数を参照してください。

キャッシュ・データを使った作業

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。

タスクの詳細については，908 ページの「キャッシュ関数を挿入する方法」を参照してください。

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 個の仮想ユーザを 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)
```

テキストと画像の検証

VuGen を使って Web 仮想ユーザ・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，多数の仮想ユーザ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仮想ユーザ・スクリプトを作成します。

仮想ユーザ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。仮想ユーザ・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，単独の仮想ユーザ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，多数の仮想ユーザ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 チェックによって仮想ユーザの処理が増えるので，Load Generator ごとの仮想ユーザ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。

画像チェックまたはテキスト・チェックを追加するには，906 ページの「テキスト・チェックと画像チェックを追加する方法」を参照してください。

Web 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。concurrent functions 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。

データ形式拡張機能

VuGen では，多くのさまざまなデータ・タイプの記録がサポートされます。ただし，新しい形式が常に作成されており，VuGen はその形式をサポートするように適応する必要があります。一部の独自の形式では，カスタムのシリアル化が使用されます。これはフォーマットされていないバイナリ・データのため，ユーザはコードを理解することが困難です。VuGen では，データ形式拡張機能（DFE）を使用してコードを可読性の高い形式に変換する方法が開発されました。これにより，ユーザはこのデータをパラメータ化したり，相関させたりできます。

データ形式拡張機能は，順序付けられたチェーン内で動作します。チェーンは拡張機能で構成されます。［ボディ］セクションと［クエリ文字列］セクションは各 HTTP メッセージに 1 つしかないため，それらのセクションでユーザが選択できるのは標準設定チェーンのみになりますが，［ヘッダ］セクションと［Cookie］セクションは HTTP メッセージに複数含めることができます。そのため，ユーザはメッセージの［ヘッダ］セクションと［Cookie］セクションのぞれぞれに別のチェーンを割り当てることができます。VuGen で［ヘッダ］セクションまたは［Cookie］セクションに対してチェーンを適切に照合できるようにするには，［名前］カラムの名前が［ヘッダ］セクションまたは［Cookie］セクションの名前と一致している必要があります。

VuGen は，チェーン内で DFE を 1 つずつ使用してデータのシリアル化解除を試みます。各拡張機能は，チェーン内で特定のデータ形式をシリアル化解除 / シリアル化するように設計されています。1 つの拡張機能で，指定されたデータが変換されない場合，そのデータはチェーン内の次の拡張機能に渡されます。拡張機能でデータの変換に成功した場合，変換されたデータに対して DFE チェーンの残りを引き続き実行するか，プロセスを終了するかを VuGen に設定できます。

VuGen には，多くのデータ形式拡張機能が組み込まれています。各 DFE の詳細については，913 ページの「データ形式拡張機能リスト」を参照してください。

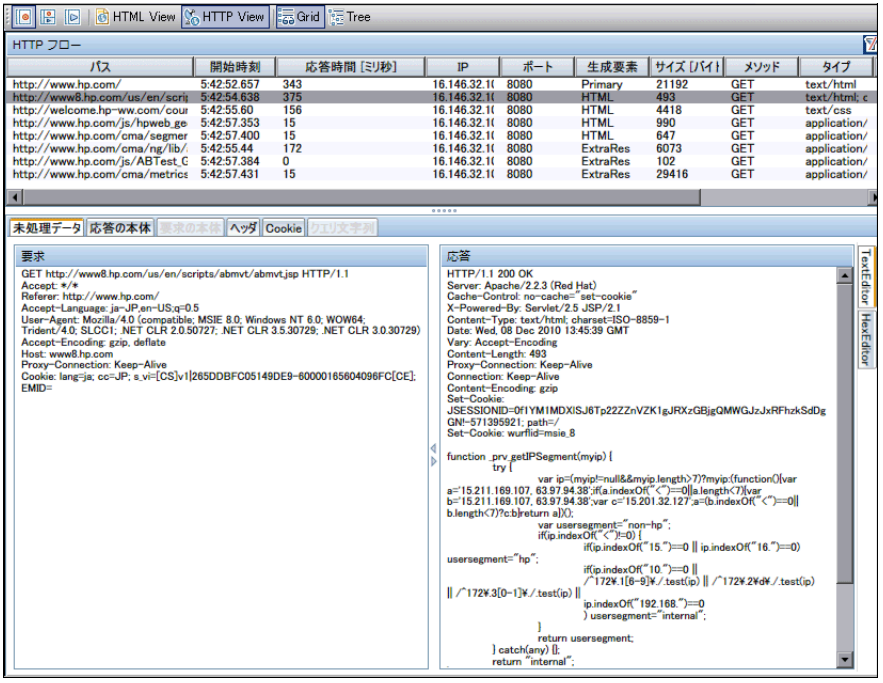
また，上級ユーザは，チェーンの追加や変更だけでなく新しいデータ形式拡張機能の作成を手動で行うこともできます。

詳細については，『HP LoadRunner Data Format Extensions Developer Guide』（英語版）を参照してください。

Web スナップショット

Web（HTTP/HTML）プロトコルに基づく VuGen プロトコルには，独自のスナップショット表示枠があります。スナップショットには，各 Web ステップに関する詳細情報が表示されます。各ステップでは，記録中，再生中，またはその両方の期間に取得されたスナップショットを表示できます。各スナップショットは，HTML ビューや，より詳細な HTTP ビューを使用して表示できます。

HTTP ビューには，HTTP フローまたはツリー内（ビューによって異なる）のすべての HTTP トランザクションが表示されます。トランザクション・データは，応答データと要求データ，ヘッダ，Cookie，およびクエリ文字列に分割されます。



データは，テキスト・エディタ，バイナリ・エディタ，XML エディタなど，さまざまなエディタを使用して表示できます。

対象のテキストを選択し，右クリックすることで，応答データに対して相関とパラメータを作成できます。

操作が困難なデータ（バイナリ・データなど）に対して，VuGen ではさまざまなデータ形式拡張機能が用意されています。これらの機能を使用すると，特定のデータ・タイプを可読性の高い形式に変換できます。データ形式拡張機能によってフォーマットされたデータは，その元の状態またはフォーマットされた状態で表示できます。詳細については，903 ページの「データ形式拡張機能」を参照してください。

新しい Web スナップショット・モデルは以前のバージョンの LoadRunner との後方互換性が確保されていますが，一部のスナップショット・データがなくなる場合があります。この問題が発生する場合は，スクリプトを再生成します。

XML ページ

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，*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。

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。

タスク

テキスト・チェックと画像チェックを追加する方法

VuGen では，さまざまな種類のチェックをスクリプトに追加できます。参考情報については，900 ページの「テキストと画像の検証」を参照してください。

記録中にテキスト・チェックを追加する

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。

テキスト・チェックを追加する（記録後）

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・メニューから［**テキスト チェックを追加 (web_reg_find)**］を選択し，［文字列の検索］ダイアログ・ボックスに入力します。詳細については，関数リファレンスを開くときにダイアログ・ボックスで F1 キーを押してください。

その他のテキスト・チェックを追加する（記録後）

1 検証する画像が含まれるステップを選択します。

2［**挿入**］>［**新規ステップ**］を選択します。

a **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。**

b **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。

3 ダイアログ・ボックスに入力します。詳細については，関数リファレンスを開くときにダイアログ・ボックスで F1 キーを押してください。

画像チェックを追加する（記録後）

1 検証する画像が含まれるステップを選択します。

2 ［**挿入**］>［**新規ステップ**］>［**Web チェック**］>［**画像チェック**］を選択します。

3 ［画像チェックのプロパティ］ダイアログ・ボックスに入力します。詳細については，関数リファレンスを開くときにダイアログ・ボックスで F1 キーを押してください。

Web 仮想ユーザ・スクリプトを Java に変換する方法

VuGen には，Web 仮想ユーザ用に作成したスクリプトを Java 仮想ユーザ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 の仮想ユーザの両方を混合した仮想ユーザ・スクリプトを作成できます。

Web 仮想ユーザ・スクリプトを Java 仮想ユーザ・スクリプトに変換するには，次の手順で行います。

1 空の Java 仮想ユーザ・スクリプトを作成して保存します。

2 空の Web 仮想ユーザ・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 仮想ユーザ・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・ファイルの完全パスとファイル名を指定し，**outputfilename** には出力ファイルの完全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，ファイルの内容を仮想ユーザ・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 仮想ユーザ・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 仮想ユーザ（RMI および CORBA）では，この変更は自動的に行われます。

8 通常の Java スクリプトと同様に，仮想ユーザ・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。

キャッシュ関数を挿入する方法

このタスクでは，キャッシュ関数の使用方法について説明します。キャッシュ関数を使用すると，格納されたデータをブラウザのキャッシュに保存しておいて，スクリプト内の後の時点で読み込むことができます。詳細については，899 ページの「キャッシュ・データを使った作業」を参照してください。

キャッシュ関数を使用するには，次の手順で行います。

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仮想ユーザ・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。

例：

給与明細を参照している PeopleSoft Enterprise 仮想ユーザの例を次に示します。

```
Action()
{
//    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;
}
```

データ形式拡張機能のチェーンを作成または変更する方法

1 ［チェーン］表示枠からチェーンを追加または選択する

［**ツール**］>［**記録オプション**］>［**データ形式拡張機能**］>［**Chain Configuration**］ノードに移動します。

詳細については，345 ページの「[Data Format Extension］-［Chain Configuration］ノード」を参照してください。

2 ［チェーン：< チェーン名 >］表示枠で選択したチェーンのデータ形式拡張機能を追加または変更する

拡張機能をチェーンに追加したら，［**処理の続行**］ドロップダウン・リストから該当のオプションを選択します。

［**チェーン：< チェーン名 >**］表示枠の詳細については，345 ページの「[Data Format Extension］-［Chain Configuration］ノード」を参照してください。

［**Prefix Postfix Extension**］をチェーンに追加すると，［**Add Prefix/Postfix to Chain**］ダイアログ・ボックスが開き，拡張機能を設定できます。

詳細については，345 ページの「[Data Format Extension］-［Chain Configuration］ノード」の［**Edit DFE**］の行を参照してください。

HTTP メッセージのセクションにチェーンを適用する方法

1 コード生成を設定する

［**データ形式拡張機能を有効にする**］チェック・ボックスを選択したら，コードで必要な［**形式**］オプションを選択します。

拡張機能は，コードとスナップショットまたはスナップショットのみに適用できます。また，コードが正確に変換されたかどうかを検証するよう LoadRunner に指示することもできます。

詳細については，348 ページの「[Data Format Extension］-［Code Generation］ノード」を参照してください。

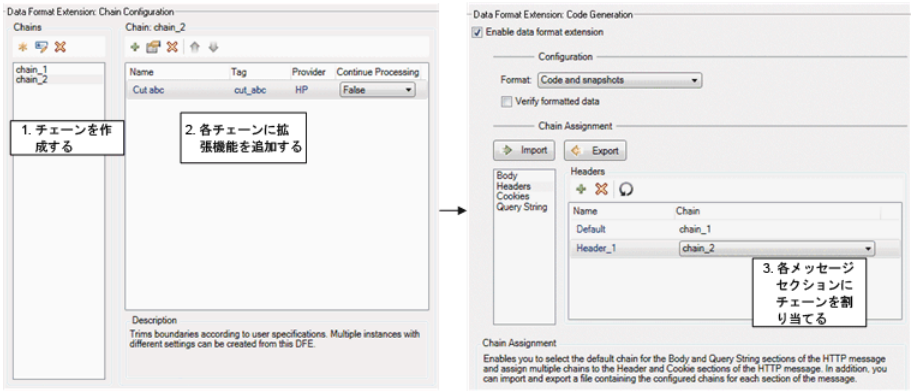
2 HTTP メッセージのセクションにチェーンを適用する

セクションのリスト（[ボディ]，[ヘッダ]，[Cookie]，[クエリ文字列]）から HTTP メッセージのセクションを選択します。

[< **メッセージ セクション** >] 表示枠で，選択したメッセージ・セクションのチェーンを追加または変更します。[**ボディ**] セクションと [**クエリ文字列**] セクションで変更できるのは，標準設定チェーンのみです。[**ヘッダ**] セクションと [**Cookie**] セクションでは，複数のチェーンを追加できます。

VuGen で [ヘッダ] セクションまたは [Cookie] セクションに対してチェーンを適切に照合できるようにするには，[**名前**] カラムの名前がメッセージの [ヘッダ] セクションまたは [Cookie] セクションの名前と一致している必要があります。

詳細については，348 ページの「[Data Format Extension] - [Code Generation] ノード」を参照してください。



リファレンス

データ形式拡張機能リスト

VuGen に組み込まれたデータ形式拡張機能のリストを次の表に示します。データ形式拡張機能の詳細については，903 ページの「データ形式拡張機能」を参照してください。

データ形式拡張機能	説明
Base64 拡張機能	BASE64 エンコーダでエンコードされた文字列をデコードします。
URL エンコード拡張機能	URL エンコード形式でエンコードされた文字列をデコードします。
JSON 拡張機能	JSON データを XML 形式に変換します。
XML バリデータ拡張機能	データを受け取り，そのデータが XML 構文に準拠しているかどうかをチェックします。このチェックにより，VuGen は XPath に基づいて相関を実行でき，スナップショット・データを xml ビューアに表示できます。
Prefix Postfix Extension	デコードしない文字列の先頭や末尾からデータを切り取ることができます。必要に応じて，いくらでもプレフィックス / ポストフィックス拡張機能を追加およびカスタマイズできます。作成する各プレフィックス / ポストフィックス拡張機能には一意の表示名とタグ名を付ける必要があります。

35

Web サービス - スクリプト内容の追加

本章の内容

概念

タスク

リファレンス

概念

Web サービスのテストの概要

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，スクリプトを手動で作成できます。

Web サービス・スクリプトの内容の追加の概要

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，125 ページの「[新規仮想ユーザ] ダイアログ・ボックス」で説明しているように，記録，手動による 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。

Service Test ライセンスでは，スクリプトを作成するための追加の方法（SOA テスト・ジェネレータ）が提供されます。

このセクションには次の内容も含まれます。

- 917 ページの「Web サービス・スクリプトの記録」
- 917 ページの「新しい Web サービス呼び出しの追加」
- 918 ページの「SOAP 要求のインポート」
- 919 ページの「サーバ・トラフィックの分析」
- 919 ページの「Business Process Testing」

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。

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。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 に変更がある場合に簡単に更新できる記述的なスクリプトが作成されます。

サービスを事前にインポートしないでスクリプトを記録する（非推奨）場合，VuGen では Web サービス呼び出しステップではなく，SOAP 要求が作成されます。SOAP 要求の引数は，直感的に理解することが難しいため保守が困難です。

詳細については，934 ページの「セッションを記録する（任意）」を参照してください。

新しい Web サービス呼び出しの追加

Web サービス呼び出しを手動で追加して，スクリプトを作成できます。呼び出しは，操作，トランスポート，引数，およびその他のプロパティに基づいて設計します。

詳細については，935 ページの「新しいサービス呼び出しを追加する（任意）」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 では，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に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。値は，［入力引数］ノードの下の［**ステップのプロパティ**］タブに表示できます。

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。

オンライン関数リファレンス（［**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。

詳細については，936 ページの「SOAP をインポートする（任意）」を参照してください。

サーバ・トラフィックの分析

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**トラフィックの分析**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。

詳細については，940 ページの「トラフィックを分析することでスクリプトを作成する方法」を参照してください。

Business Process Testing

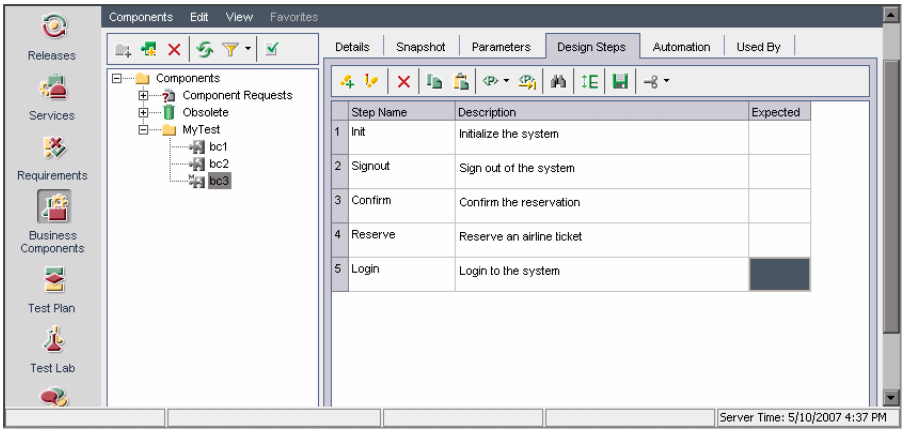
BPT（Business Process Testing）は，いくつかのテストを組み合わせることで完全なビジネス・プロセスを作成する 1 つの方法論です。BPT ユーザは，一連のテスト・コンポーネントと，それらのテスト・コンポーネント間のデータ・フローを組み合わせて，完全なテストを組み立てます。

注：BPT 機能は，Service Test ライセンスでのみ使用できます。詳細については，HP のサポートにお問い合わせください。

コンポーネントはステップで構成されます。たとえば，ログイン・コンポーネントの 1 番目のステップとして，アプリケーションを開くことが考えられます。2 番目のステップとして，ユーザ名の入力が考えられます。3 番目のステップはパスワードの入力，4 番目のステップは **Enter** ボタンのクリックが考えられます。

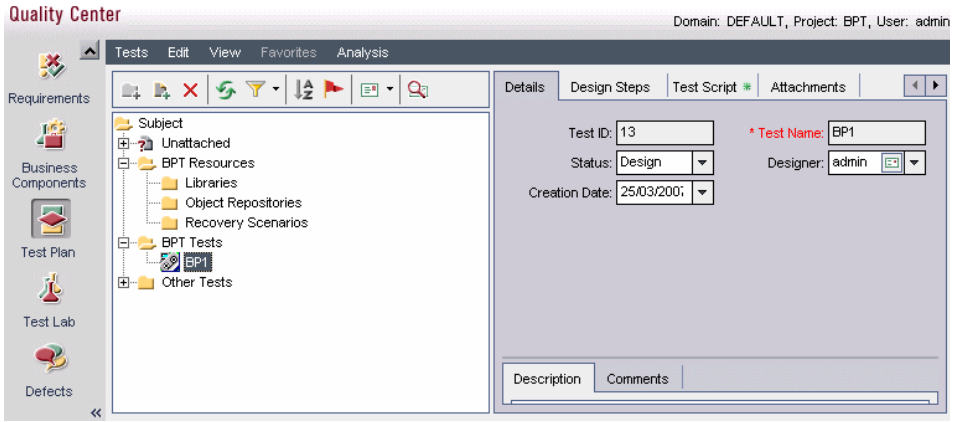
ビジネス・プロセスのテスト・コンポーネントは ALM 内から作成できます。また，VuGen with Service Test を使用して作成することもできます。

Application Lifecycle Management では，非技術系の SME（主題専門家）が，テストに必要な設計ステップを定義します。



この設計ステップの定義が完了すると，技術エンジニアが，必要なアクションを実行する Service Test を使用してスクリプトを作成します。ビジネス・コンポーネントを Application Lifecycle Management 内から作成する方法については，『Business Process Testing ユーザ・ガイド』を参照してください。

アプリケーションを使用してセッションを記録するか，スクリプトを手動で編集することで，Service Test にコンポーネントを作成します。チェックポイントを追加したり，選択した項目をパラメータ化したり，フロー・ステートメントおよびその他のテスト機能を使用してコンポーネントを拡張したりできます。次に，コンポーネントを Application Lifecycle Management 内のプロジェクトに保存します。主題専門家は，Application Lifecycle Management の Business Process Testing を使用して，保存されたコンポーネントを 1 つ以上のビジネス・プロセス・テストに結合します。このテストを使用して，アプリケーションが予測どおりに動作することを確認します。



Application Lifecycle Management でビジネス・コンポーネントのパラメータを作成すると，そのパラメータは Service Test のパラメータ・リストで使用できます。

個々のスクリプトで BPT モジュールを使って作業する利点には，次のようなものがあります。

- 技術的でない主題の専門家がテストを作成できる
- テストの構造化および自動化が可能
- 手動によるテストと自動スクリプトを組み合わせると，作業の重複が削減される
- コンポーネントを再利用して，自動化プロセスを短縮できる

- ビジネス・プロセス内のステップ間でパラメータを受け渡すことができる。ステップの出力をパラメータに保存し，その出力を以降のステップの入力値として使用できる。
- 実行中のテストの保守が容易になる
- テストまでの時間を最小限に抑えられる

タスクの詳細については，第 35 章，「VuGen でビジネス・コンポーネントを作成する方法」を参照してください。

Application Lifecycle Management で BPT コンポーネントを作成する方法については，『Business Process Testing ユーザ・ガイド』を参照してください。

スクリプトの統合

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 37 章，「Web サービス - スクリプトの再生の準備」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**：Business Process Monitor 設定で，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。

HP Application Lifecycle Management のアドオンである Service Test Management を使用すると，Application Lifecycle Management でサービスのインポート，保管，定義ができるようになり，SOA テストを管理できます。詳細については，HP の担当者にお問い合わせください。

Web サービス呼び出しの添付ファイル

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 では，DIME はすべてのツールキットでサポートされますが，MIME は Axis ツールキットでのみサポートされます。.NET ツールキットで MIME 添付ファイルを使用するには，1054 ページの「MIME 添付ファイルの包含」を参照してください。

VuGen では，SOAP メッセージとともに添付ファイルを送受信できます。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。タスクの詳細については，936 ページの「添付ファイルを追加する（任意）」を参照してください。

出力添付ファイルを使用して，応答を添付ファイルとして保存します。［**すべての添付ファイルを保存する**］または［**添付ファイルをインデックスとして保存する**］のいずれかのオプションを選択できます。

［**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添付ファイルごとに次の 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。

特別な引数の型

VuGen では，派生，繰り返し，選択，オプション要素など，特別な引数の型が処理されます。

このセクションの内容

派生型

VuGen では，派生型を使用した WSDL がサポートされます。Web サービス呼び出しのプロパティを設定する場合，VuGen では，引数に基本データ型または派生型を使用できます。型を選択した後に，VuGen によって，引数ツリー・ノードが更新されて新しい型が反映されます。詳細については，954 ページの「< 入力引数名 > ノード」を参照してください。

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 は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。

オプションの要素

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" />
```

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［**自動値の生成**］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を解釈します。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。

Choice オプション要素

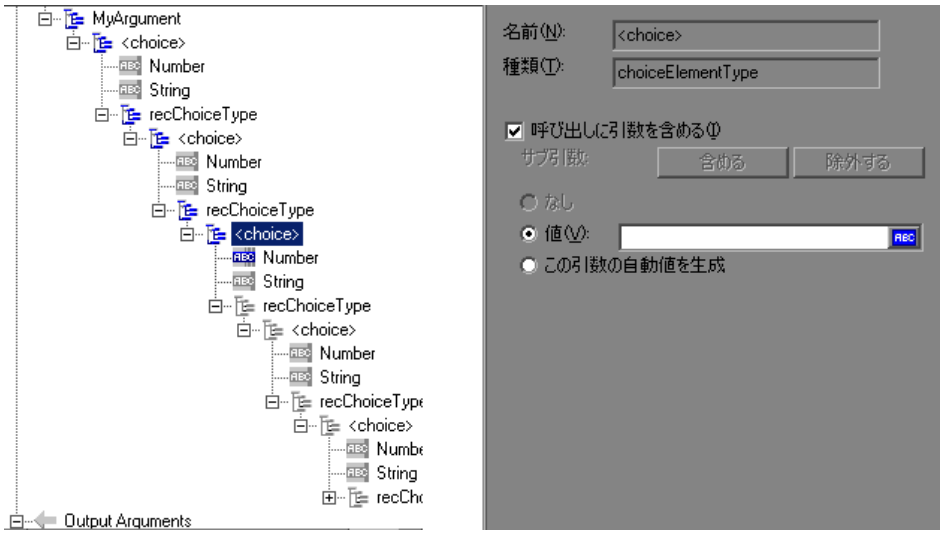
WSDL の Choice 要素では一連の要素を定義し，その 1 つだけが SOAP メッセージに表示されます。場合によっては，Choice 要素の 1 つがオプションで，その他はオプションではありません。Choice 要素を選択して，そのオプションの要素が SOAP エンベロープに表示されないようにできます。ツリー・ビューでは，Choice 要素を選択して，［**呼び出しに引数を含める**］オプションをクリアします。スクリプト・ビューでは，Choice 引数を定義している行を削除します。

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，含める最低の親ノードを選択して，［**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，そのすべての親ノードが含まれます。

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。除外されたノードはグレー表示になっています。



Base 64 の引数

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。

- **[ファイルから取得]**：ファイル名を参照します。
- **[エンコード テキストを埋め込む]**：エンコードするテキストを指定します。

詳細については，962 ページの「[Process Base64 Data - Simple Data] ダイアログ・ボックス」を参照してください。

サービスの要件とテストの生成の概要

SOA 環境をテストするには，テストを手動で作成するか，SOA テスト・ジェネレータを使用してスクリプトを自動的に生成できます。

注：自動スクリプト・ジェネレータは，Service Test ライセンスでのみ使用できます。詳細については，HP のサポートにお問い合わせください。

このセクションでは，SOA テスト・ジェネレータの使用方法について説明します。テストを手動で作成する方法については，934 ページの「内容の追加方法」を参照してください。

SOA テスト・ジェネレータの手順に従って，サービスをテストするためのスクリプトを作成できます。このウィザードを使用して，テストするサービスのアスペクトを指定します。このアスペクトには，各種ツールキットとの相互運用性，境界テスト，標準準拠が含まれます。

テスト対象アスペクトを選択した後に，VuGen with Service Test によって，側面ごとに 1 つ以上のスクリプトが自動的に生成されます。

テスト対象アスペクト

SOA テスト生成ウィザードを使用すると，サービスのさまざまなアスペクトを検証するための要件とテストを容易に作成できます。サービス・テスト管理 では，アスペクトとサブアスペクトごとに独立した要件が作成され，サブアスペクトごとに独立したテストが作成されます。サブアスペクトがない場合は，アスペクト用の独立したテストのみが作成されます。

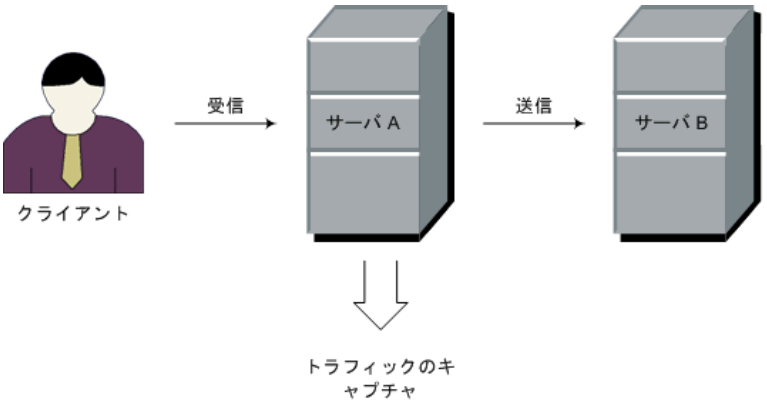
標準設定では，サービス・テスト管理 では，次のテスト対象アスペクトがサポートされます。詳細については，964 ページの「アスペクト・リファレンス」を参照してください。

サーバ・トラフィック・スクリプトの概要

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**トラフィックの分析**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 940 ページの「トラフィックを分析することでスクリプトを作成する方法」で説明します。



エミュレーションには，**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。

キャプチャ・ファイル

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。

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**。

外部のキャプチャ・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。

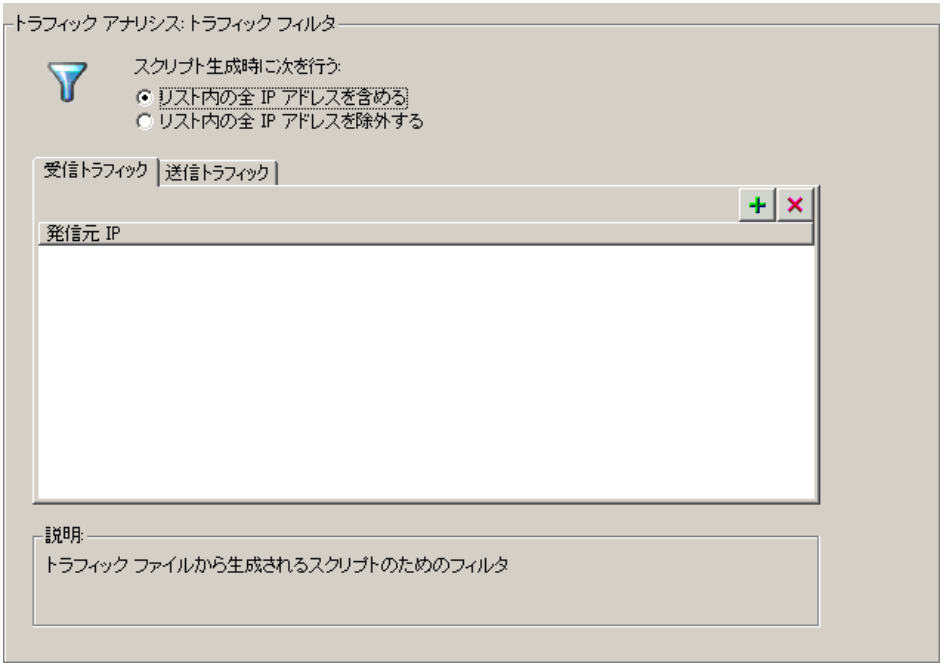
一部のキャプチャ・ユーティリティには引数を追加する必要があります。たとえば，**tcpdump** には，データを切り捨てずにパケットをキャプチャするために引数 **-s 0** が必要です。

トラフィックのフィルタ処理

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。

ヒント：いくつかの外部キャプチャ・ツールでは，トラフィックのキャプチャ中に IP アドレスをフィルタ処理できます。

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。



詳細については，942 ページの「IP アドレスをフィルタ処理する（任意）」を参照してください。

セキュア・サーバ上のデータ

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。

詳細については，942 ページの「SSL を設定する（任意）」を参照してください。

タスク

内容の追加方法

このタスクでは，Web サービス呼び出しなど，内容をスクリプトに追加する方法について説明します。

このタスクでは，次の手順を実行します。

- 934 ページの「前提条件」
- 934 ページの「セッションを記録する（任意）」
- 935 ページの「新しいサービス呼び出しを追加する（任意）」
- 936 ページの「SOAP をインポートする（任意）」
- 937 ページの「サーバ・トラフィックを分析する（任意）」

前提条件

空の Web サービス・スクリプトを作成します。［**ファイル**］>［**新規作成**］をクリックし，仮想ユーザのタイプとして［**Web サービス**］を選択します。シングル・プロトコル・タイプまたはマルチ・プロトコル・タイプのスクリプトを作成できます。

セッションを記録する（任意）

1 ［**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。

タスクの詳細については，946 ページの「［サービスの選択］画面」を参照してください。

2 サービスをリストに追加します。［**インポート**］をクリックして，テスト用の WSDL をロードします。WSDL ファイルの場所を指定します。

ユーザ・インタフェースの詳細については，1014 ページの「［サービスのインポート］ダイアログ・ボックス」を参照してください。

3 ［**次へ**］をクリックします。アプリケーションの場所とその他の関連する引数を指定します。947 ページの「［記録］ダイアログ・ボックスを開始します。」を参照してください。

新しいサービス呼び出しを追加する（任意）

1 サービスをインポートします。[**サービスの管理**] をクリックして，[インポート] ダイアログ・ボックスにアクセスします。

ユーザ・インタフェースの詳細については，1014 ページの「[サービスのインポート] ダイアログ・ボックス」を参照してください。

2 スクリプト（スクリプト・ビュー）内またはテスト・ステップ（ツリー・ビュー）内の目的の場所でカーソルをクリックします。

3 [**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。

4 [Web サービス呼び出しの選択] セクションで，[**サービス**]，[**ポート名**]，[**操作**] を選択します。

5 標準設定の [**ターゲット アドレス**] 以外のエンドポイントを指定するには，[**優先アドレス**] を選択し，要求の送信先とする新しいエンドポイントを挿入します。

6 ノードを展開し，引数値を指定します。すべての入力引数にサンプル値を作成するには，[**入力引数**] ノードを選択して，[**生成**] をクリックします。要素の XML 構造の編集，インポート，エクスポートを行うには，937 ページの「値を XML 要素に割り当てる方法」を参照してください。

7 入力引数をパラメータ化するには，ノードをクリックし，[**値**] オプションを選択します。[ABC] アイコンをクリックして，パラメータ化を進めます。詳細については，263 ページの「パラメータ」を参照してください。

8 [**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，1061 ページの「JMS でメッセージを送信する方法」を参照してください。

添付ファイルを追加する（任意）

1 添付ファイルを入力引数に追加するには，左の表示枠で操作を選択します。［**要求に追加する（入力）**］を選択します。VuGen によって，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。詳細については，959 ページの「[入力添付ファイルの追加］ダイアログ・ボックス」を参照してください。

2 出力引数の格納先とする出力添付ファイルを指定するには，左の表示枠で操作を選択します。［**受信したものを保存する（出力）**］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**］のいずれかのオプションを選択します。詳細については，923 ページの「Web サービス呼び出しの添付ファイル」を参照してください。

3 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 ヘッダを指定する（任意）

左の表示枠で［**カスタム SOAP ヘッダ**］ノードを選択し，［**SOAP ヘッダを使用する**］オプションを有効にします。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。独自の SOAP ヘッダを構成するには，［**編集**］をクリックし，XML を編集します。SOAP ヘッダ用の XML ファイルをインポートするには，［**インポート**］をクリックします。

SOAP をインポートする（任意）

1 サービスが利用可能な場合は，サービスをインポートします。［**サービスの管理**］をクリックして，［インポート］ダイアログ・ボックスにアクセスします。詳細については，1014 ページの「[サービスのインポート］ダイアログ・ボックス」を参照してください。

2 ［**SOAP のインポート**］ボタンをクリックして，［SOAP のインポート］ダイアログ・ボックスを開きます。

3 SOAP 要求を示す XML ファイルを参照します。

4 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。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。

5 ［**ロード**］をクリックして，XML 要素の値をインポートします。

Web サービス呼び出しでは，950 ページの「［新規 Web サービス呼び出し］ダイアログ・ボックス」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。

6 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します。

7 ［**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。

サーバ・トラフィックを分析する（任意）

サーバ・トラフィックのダンプを含むファイルを分析することでスクリプトを作成するには，［**トラフィックの分析**］をクリックします。

詳細については，929 ページの「サーバ・トラフィック・スクリプトの概要」を参照してください。

値を XML 要素に割り当てる方法

このタスクでは，手動によるコードの編集，外部ファイルのインポート，および後で使用するための XML のエクスポートなど，XML 要素を使った作業について説明します。

このタスクでは，次の手順を実行します。

- 938 ページの「前提条件」
- 938 ページの「要素を選択する」
- 938 ページの「ファイルをインポートする（任意）」
- 938 ページの「XML 要素を編集する（任意）」
- 938 ページの「ファイルをエクスポートする（任意）」

1 前提条件

サービスをインポートし，新しい Web サービス呼び出しを作成します。または，ステップを選択して［**ステップのプロパティ**］タブをクリックします。

2 要素を選択する

左の表示枠で，複合型または配列引数を選択します。右の表示枠で，［**XML**］をクリックします。XML フィールドに，XML コードが単独の文字列として表示されます。

3 ファイルをインポートする（任意）

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。

4 XML 要素を編集する（任意）

XML 構造と要素の値を編集するには，［**編集**］をクリックします。XML エディタが開きます。以前に保存した XML ファイルをインポートするには，［**ファイルのインポート**］をクリックします。

- コードを手動で編集するには，［**テキスト ビュー**］タブをクリックします。
- XML をグラフィカル・インタフェースで変更するには，［**ツリー ビュー**］をクリックします。ショートカット・メニューを使用して，子要素と兄弟要素を追加し，ノード名を変更します。ショートカット・メニューで［**挿入**］をクリックし，選択した要素の前または後に新しい要素を追加します。

5 ファイルをエクスポートする（任意）

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。

テストを自動的に生成する方法

このタスクでは，サービスをチェックするための要件またはテストを作成する方法について説明します。

このタスクでは，次の手順を実行します。

- 934 ページの「セッションを記録する（任意）」
- 941 ページの「サービスをインポートする（任意）」

➤ 942 ページの「トラフィック情報を指定する」

➤ 939 ページの「位置を指定する」

➤ 939 ページの「テストの生成を完了する」

➤ 939 ページの「スクリプトを開く」

1 ウィザードを開く

［**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。左側の表示枠で［**SOA テスト ジェネレータ**］を選択し，［**作成**］をクリックします。

2 サービスを追加する

次の画面に進み，［**追加**］をクリックして 1 つ以上のサービスをインポートします。サービスが準備されていない場合は，エミュレートしたサービスを使用できます。詳細については，997 ページの「サービスを追加および管理する方法」を参照してください。［**次へ**］をクリックします。

3 テスト対象アスペクトを選択する

ノードを展開し，必要なテスト対象アスペクトを選択します。アスペクトの詳細については，964 ページの「アスペクト・リファレンス」を参照してください。［**次へ**］をクリックします。

4 位置を指定する

テストの名前，およびテスト・スクリプトの場所（**HP ALM** または**ローカル・ファイル・システム**）を指定します。ALM を指定する場合は，［**接続**］をクリックしてサーバにログオンし，［**参照**］をクリックしてテスト・ノードを見つけます。

5 テストの生成を完了する

サマリを確認し，スクリプトを生成に含めるかまたは除外します。［**生成**］をクリックします。

6 スクリプトを開く

最後の画面で，生成されたスクリプトのリストを確認し，開くスクリプトを指定します。［**完了**］をクリックします。

トラフィックを分析することでスクリプトを作成する方法

このタスクでは，ネットワーク・トラフィック・ファイルを使用してスクリプトを作成する方法について説明します。

このタスクでは，次の手順を実行します。

- 934 ページの「前提条件」
- 934 ページの「セッションを記録する（任意）」
- 941 ページの「サービスをインポートする（任意）」
- 942 ページの「トラフィック情報を指定する」
- 942 ページの「IP アドレスをフィルタ処理する（任意）」
- 942 ページの「SSL を設定する（任意）」

1 Windows プラットフォームでキャプチャ・ファイルを作成する（任意）

Windows プラットフォーム上の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むキャプチャ・ファイルを作成します。ダウンロード可能なキャプチャ・ツールを使用するか，製品の **bin** フォルダに提供されているツール（**lrtcpdump.**< *プラットフォーム* >）を使用します。

a コマンド・ウィンドウでキャプチャ・ユーティリティ **lrtcpdump -f**<*file_name*>**.cap** を実行します。**lrtcpdump** により，ネットワーク・カードの選択を求められます。

b インタフェース・カードの番号を入力し（複数個ある場合），**Enter** キーを押します。

c 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。

d 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。

e 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。

2 UNIX プラットフォームでキャプチャ・ファイルを作成する（任意）

UNIX プラットフォーム上のネットワークを通過したすべての TCP トラフィックを含むキャプチャ・ファイルを作成します。

a 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。

b root ユーザに切り替えて，次の実行パーミッションを指定します：
chmod 755 lrtcpdump.< *プラットフォーム* **>**

c インタフェース・カードが複数ある場合，**lrtcpdump** では，アルファベット順で最初のインタフェース・カードが使用されます。全インタフェースのリストを取得するには，**ifconfig** コマンドを使用します。

d 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 -f<***file_name***>.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。

e 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。

f **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。

g 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。

3 トラフィック分析ウィザードを開く

［**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。詳細については，946 ページの「[サービスの選択］画面」を参照してください。

4 サービスをインポートする（任意）

リストに 1 つ以上のサービスを追加します（任意）。［**インポート**］をクリックして，WSDL ファイルをロードします。詳細については，1014 ページの「[サービスのインポート］ダイアログ・ボックス」を参照してください。

［**次へ**］をクリックします。

5 トラフィック情報を指定する

キャプチャ・ファイルを指定し，トラフィックのロード先とするスクリプトのセクション（**vuser_init**，**Action**，または **vuser_end**）を指定します。

受信または**送信**のどちらを分析するかを指定します。トラフィックを分析するサーバを指定します。

詳細については，947 ページの「[記録] ダイアログ・ボックスを開始します。」を参照してください。

6 IP アドレスをフィルタ処理する（任意）

[**フィルタ オプション**] ボタンをクリックして [記録] オプションを開き，無視する IP アドレスまたは含める IP アドレスを指定します。

詳細については，313 ページの「記録オプション」を参照してください。

7 SSL を設定する（任意）

[**SSL 設定**] ボタンをクリックして，SSL 証明書を追加します。セキュア・サーバからトラフィックを分析するには，この証明書が必要です。

詳細については，974 ページの「[SSL 設定] ダイアログ・ボックス」を参照してください。

VuGen でビジネス・コンポーネントを作成する方法

このタスクでは，VuGen でビジネス・コンポーネントを作成する方法について説明します。この機能は，Service Test ライセンスでのみ使用できます。詳細については，HP のサポートにお問い合わせください。

このタスクでは，次の手順を実行します。

- 943 ページの「前提条件」
- 943 ページの「新しいビジネス・コンポーネントを作成する」
- 943 ページの「内容を追加する」
- 943 ページの「テストを保存する」

1 前提条件

Application Lifecycle Management に接続していることを確認します。詳細については，252 ページの「ALM を使ったスクリプト管理の概要」を参照してください。

2 新しいビジネス・コンポーネントを作成する

［**ファイル**］>［**ビジネス コンポーネント**］>［**新規作成**］を選択します。［新規ビジネス コンポーネント］ダイアログ・ボックスが開きます。新しい Web サービス・スクリプトを作成します。

3 内容を追加する

記録する，SOAP をインポートする，または Web サービス呼び出しを手動で追加することによって，スクリプトの内容を追加します。詳細については，916 ページの「Web サービス・スクリプトの内容の追加の概要」を参照してください。

4 引数をパラメータ化する（任意）

必要な引数をパラメータ化します。Application Lifecycle Management 内のビジネス・コンポーネント間でパラメータを共有するには，BPT タイプのパラメータを作成します。詳細については，266 ページの「パラメータ・タイプ」を参照してください。

5 テストを保存する

スクリプトを Application Lifecycle Management 内の目的の場所に保存します。Service Test で作成したビジネス・コンポーネントは，Application Lifecycle Management では，自動化テストとみなされます。

既存の仮想ユーザ・スクリプトをビジネス・コンポーネントに変換するには，［**ファイル**］>［**ビジネス コンポーネント**］>［**ビジネス コンポーネントとして保存**］を選択し，スクリプトを Application Lifecycle Management 内の目的の場所に保存します。

6 ビジネス・コンポーネントを編集する（任意）

［**ファイル**］>［**ビジネス コンポーネント**］>［**開く**］を選択します。開くスクリプトを指定します。ほかのスクリプトの場合と同様に，スクリプトを編集し，パラメータを設定して，内容を追加します。スクリプトを保存します。

Application Lifecycle Management でビジネス・コンポーネントを作成する方法

次のセクションでは，Application Lifecycle Management でビジネス・コンポーネントを作成し，VuGen との互換性が確保されるようにビジネス・コンポーネントを自動化する方法について簡単に説明します。Application Lifecycle Management でビジネス・コンポーネントを使用する方法については，『Business Process Testing ユーザ・ガイド』を参照してください。

この機能は，Service Test ライセンスでのみ使用できます。詳細については，HP のサポートにお問い合わせください。

このタスクでは，次の手順を実行します。

- 944 ページの「新しいビジネス・コンポーネントを作成する」
- 944 ページの「ステップを作成する」
- 944 ページの「ビジネス・コンポーネントを自動化する」
- 945 ページの「ビジネス・コンポーネントを編集する（任意）」

1 新しいビジネス・コンポーネントを作成する

Application Lifecycle Management で［**ビジネス コンポーネント**］モジュールを選択し，［**コンポーネント**］>［**New Component**］を選択します。

2 ステップを作成する

［**Component Steps**］タブを選択し，手動により新しいステップを作成して，パラメータを定義（任意）します。

3 ビジネス・コンポーネントを自動化する

［**Component Steps**］タブで，［**Automate Component**］（右端のボタン）>［**Service Test**］を選択します。

4 ビジネス・コンポーネントを編集する（任意）

ツリー階層でビジネス・コンポーネントを選択します。[**Automation**] タブを選択し，[**Launch**] をクリックします。VuGen でスクリプトが開きます。

ほかのスクリプトの場合と同様に，スクリプトを編集し，パラメータを設定して，内容を追加します。

Application Lifecycle Management でビジネス・コンポーネントを使用する方法については，『Business Process Testing ユーザ・ガイド』を参照してください。

リファレンス

［Add Script Content］のユーザ・インタフェース

このセクションの内容

［サービスの選択］画面

このダイアログ・ボックスを使用して，スクリプトの記録を開始するのに必要な基本的な詳細を指定できます。

利用方法	［**記録開始**］ボタン
関連タスク	934 ページの「セッションを記録する（任意）」

ユーザ・インタフェース要素の説明は次のとおりです（ラベルのない要素は山括弧で囲んで示します）。

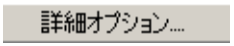
UI 要素	説明
詳細(D)...	サービスに関する詳細情報を指定するための［サービスの管理］ダイアログ・ボックスを開きます。詳細については，1009 ページの「[サービスの管理］ダイアログ・ボックス」を参照してください。
インポート(I)	［サービスのインポート］ダイアログ・ボックスが開きます。詳細については，1014 ページの「[サービスのインポート］ダイアログ・ボックス」を参照してください。 ユーザ・インタフェースの詳細については，1014 ページの「[サービスのインポート］ダイアログ・ボックス」を参照してください。
削除(R)	選択したサービスをリストから削除します。
< サービス・リスト >	使用可能なサービスのリスト。 ➤ **[サービス名]**：サービスのネイティブ名。 ➤ **[WSDL の位置]**：WSDL のソース。

［記録］ダイアログ・ボックスを開始します。

このダイアログ・ボックスを使用して，Web サービス・スクリプトの記録を開始するために必要な基本的な詳細情報を指定できます。

利用方法	［**記録開始**］ボタン，［**次へ**］
関連タスク	934 ページの「セッションを記録する（任意）」

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
	［記録オプション］ダイアログ・ボックスが開きます。ユーザ・インタフェースの詳細については，313 ページの「記録オプション」を参照してください。
標準設定の Web ブラウザを記録	標準設定のブラウザ・アクションを記録します。開始 URL を指定するか，［参照］ボタンをクリックして場所に移動します。
すべてのアプリケーションを記録	あらゆる Win32 アプリケーションを記録します。次の項目を指定することもできます。 ➤ **［記録するプログラム］**：記録するブラウザ，インターネット・アプリケーション，または Win32 アプリケーションを選択します。 ➤ **［プログラム引数］**（Win32 アプリケーションのみ）：アプリケーションのコマンド・ライン引数。たとえば，plus32.exe にコマンド・ライン・オプション peter@neptune を指定した場合，plus32.exe を起動すると，ユーザ Peter がサーバ Neptune に接続されます。 ➤ **［作業ディレクトリ］**：アプリケーションの作業ディレクトリ（アプリケーションから要求された場合のみ）
アクション内に記録	記録先のセクション。**vuser_init**，**Action**，または **vuser_end**。繰り返すアクションについては，**Action** セクションを使用します。初期化のステップでは，**vuser_init** を使用します。
アプリケーションの起動を記録する	次の場合は，起動を記録しないことをお勧めします。 ➤ マルチ・アクションの場合。起動が必要なアクションは 1 つだけです。 ➤ アプリケーションで特定の位置まで移動し，その位置から記録を開始する場合。 ➤ 既存のスクリプトに記録する場合。

［SOAP のインポート］ダイアログ・ボックス

このダイアログ・ボックスでは，SOAP ファイルに基づいてテスト・ステップを作成できます。

利用方法	次のいずれかの方法を使用します。 ➤ SOAP のインポート をクリックする ➤ **［SOA ツール］＞［SOAP のインポート］**
関連タスク	936 ページの「SOAP をインポートする（任意）」 918 ページの「SOAP 要求のインポート」

ユーザ・インタフェース要素の説明は次のとおりです（ラベルのない要素は山括弧で囲んで示します）。

UI 要素	説明
...	**［参照］**：SOAP トラフィックを含む XML ファイルを探します。
ロードする	SOAP ファイルから要素値をロードします。
サービスの管理	サービスのインポートと設定を行うための［サービスの管理］ダイアログ・ボックスを開きます。
SOAP の表示	ブラウザで SOAP ファイルを表示用として開きます。
＜呼び出しの種類＞	スクリプト / テスト内に生成する呼び出しの種類 ➤ **［Web サービス呼び出し］**：サービスのインポートを要求します。 ➤ **［SOAP 要求］**：スクリプトに soap_request ステップが生成されます。
SOAP 要求のプロパティ	SOAP 要求のプロパティ（呼び出しの種類が［**SOAP 要求**］の場合にのみ表示される）。次の情報を指定します。 ➤ **［URL］**：要求の送信先のサーバの URL または IP アドレス。 ➤ **［SOAP アクション］**：要求に含める SOAP アクション（複数のアクションがある場合に適用される） ➤ **［応答パラメータ］**：SOAP 要求または Web サービス呼び出し要求の応答を格納するパラメータの名前。

［新規 Web サービス呼び出し］ダイアログ・ボックス

このダイアログ・ボックスでは，新しい Web サービス呼び出しの作成と設定ができます。

利用方法	 をクリックする
重要情報	既存の Web サービス呼び出しの Web サービス呼び出しプロパティにアクセスするには，ツリー・ビューでステップを選択し，ショートカット・メニューから［**プロパティ**］を選択します。 参考情報については，次を参照してください。
関連タスク	935 ページの「新しいサービス呼び出しを追加する（任意）」

ユーザ・インタフェース要素の説明は次のとおりです（ラベルのない要素は山括弧で囲んで示します）。

UI 要素	説明
< サービス引数ツリー >（左の表示枠）	次のノードを含むサービスの，展開可能なツリー階層。 ➤ < 操作名 > ➤ トランスポート層の設定 ➤ カスタム SOAP ヘッダ ➤ 入力引数 ➤ 出力引数
< パラメータ値 >（右の表示枠）	左の表示枠の各ノードに対する値の設定と選択ができます。
Web サービス呼び出しの選択	次の項目を設定できます。 ➤ **[サービス]**：インポートされたすべてのサービスのドロップダウン・リスト。名前は WSDL から派生しています。 ➤ **[ポート名]**：要求の送信が可能なポートのドロップダウン・リスト。 ➤ **[操作]**：サービスの操作のドロップダウン・リスト。 ➤ **[ターゲット アドレス]**：サービスの標準設定のエンドポイント。 ➤ **[優先アドレス]**：［**ターゲット アドレス**］ボックスで，代替のエンドポイント・アドレスを入力できます。

< 操作名 > ノード

操作の入力引数のサンプル値の生成と，添付ファイルの追加ができます。

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
メソッド	選択した操作の名前（読み取り専用）。
入力引数用の自動値を生成する	すべての入力引数に対して，そのデータ・タイプに基づいてサンプル値を自動的に作成します。
添付ファイル	入力と出力の添付ファイルを処理します。 ➤ **[要求に追加する（入力）]**：ファイルまたはパラメータ値を要求に付加します。 ➤ **[受信したものを保存する（出力）]**：応答をパラメータに保存します。
ステップのプロパティ	次のサービス呼び出しプロパティとその値が表示されます。 ➤ WSDL ファイルの場所 ➤ サービス名 ➤ ポート名 ➤ ターゲット アドレス ➤ SOAP アクション ➤ SOAP 名前空間

［トランスポート層の設定］ノード

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
HTTP/S トランスポート	トランスポート・メソッドを HTTP または HTTPS トランスポートに設定します。
非同期サポート	Web サービス呼び出しを，イベントによってアクティブ化される非同期メッセージとしてマークします。 **［非同期イベント］**：イベントの任意の名前。 **注**：イベントを待機するよう再生エンジンに指示するには，［**Web サービス イベント待機**］ステップをスクリプトに追加します。
WSA サポート	WS-Addressing を有効にします。次のいずれかの応答オプションを選択します。 ➤ **［WS-A 応答対象］**：イベントが発生した場合の応答先サーバの IP アドレス。 ➤ **［自動検出］**：イベントが発生すると，現在のホストに応答します。これは，複数のマシンで同じスクリプトを実行する場合に便利です。 **ヒント**：同期モードで WS-Addressing 呼び出しを使用するには，［**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，完全な応答がサーバから受信されるまで，再生はスクリプトの実行をブロックします。
JMS トランスポート	トランスポート・メソッドを*同期メッセージ*用の JMS に設定します。詳細については，1031 ページの「Web サービス・トランスポート層のテストの概要」を参照してください。 **注**：JMS *非同期*メッセージの場合，メッセージ・キュー情報を設定するには，［**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**］ステップをスクリプトに手動で追加します。
JMS キューをオーバライド	要求キューと応答キューを提供できます。
要求キュー	要求メッセージ用のキュー名。
応答キュー	応答メッセージ用のキュー名。

[カスタム SOAP ヘッダ] ノード

アプリケーションで生成した追加のヘッダ要素を HTTP メッセージの SOAP エンベロープに含めるように指定できます。タスクの詳細については，936 ページの「SOAP ヘッダを指定する（任意）」を参照してください。

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
編集(E)...	SOAP ヘッダの XML コードの表示と編集ができる XML エディタを開きます。
インポート(M)...	[**XML ファイルのインポート先を選択**] ダイアログ・ボックスを開きます。
エクスポート(X)...	[**ファイルに SOAP ヘッダをエクスポート**] ダイアログ・ボックスを開きます。
	[**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。
SOAP ヘッダを使用する	SOAP ヘッダを HTTP 要求に含めます。
ヘッダ	ヘッダのソースは次のとおりです。 ➤インポートされたファイルの場合：インポートされたファイルに含まれるヘッダ要素 ➤パラメータの場合：パラメータ名（中括弧内）

[入力引数] ノード

すべての入力引数について，プロパティの設定と値の生成ができます。

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
全て含む	メソッドのすべての引数を Web サービス呼び出しに含めます。
リセット	引数を元の状態にリセットします。これにより，引数は呼び出しから削除され，WSDL 内の値に設定されます。

UI 要素	説明
生成	すべての入力引数のサンプル・データを生成します。
引数の編集(E)	選択した引数の値を編集するための表示枠が開きます。
名前	操作の名前（読み取り専用）
引数リスト	入力引数のリスト。 ➤  単純パラメータ ➤  配列（トップ・レベルのみが表示されます）。

< 入力引数名 > ノード

入力引数を選択すると，右の表示枠で引数値を指定できます。

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
追加(A)	新しい配列要素を入力引数に追加するための［配列要素の追加］ダイアログ・ボックスを開きます（入力配列内の親ノードを選択した場合にのみ表示されます）。詳細については，961 ページの「［配列要素の追加］ダイアログ・ボックス」を参照してください。
削除(R)	入力引数内の選択された配列要素を削除します（入力配列内の親ノードを選択した場合にのみ表示されます）。
含める	選択した引数のサブ引数を Web サービス呼び出しに含めます。これは，［**呼び出しに引数を含める**］オプションが有効になっている，サブ引数を持つ引数に対してのみ有効です。
除外する	親引数のサブ引数を Web サービス呼び出しから除外します。
編集(E)...	引数値を含む XML コードを編集するための XML エディタを開きます。保存される変更は，要素値と配列要素の数のみです。

UI 要素	説明
インポート(M)...	［**XML ファイルのインポート先を選択**］ダイアログ・ボックスを開きます。
エクスポート(X)...	［**ファイルに引数 XML をエクスポート**］ダイアログ・ボックスを開きます。
	［**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。
名前	引数または配列の名前。
呼び出しに引数を含める	呼び出しに引数を含めます。配列では，［**含める**］をクリックすると，サブ引数が呼び出しに追加されます。省略可能なすべての引数を除外するには，［**除外する**］をクリックします。
タイプ	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，924 ページの「派生型」を参照してください。
なし	Nillable 属性を **true** に設定します。
XML（配列の場合のみ）	➤ **[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，引数値を手動で挿入できます。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。 ➤ **[この引数の自動値を生成]**：すべての子要素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。
値（配列でない要素の場合）	引数値。この値をパラメータ化するには，［ABC］アイコン（配列でない場合にのみ使用可能）をクリックします。
この引数の自動値を生成	選択した引数のサンプル値を生成します。

［出力引数］ノード

すべての出力引数のプロパティを表示できます。

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
引数の編集(E)	選択した引数の値を編集するための表示枠が開きます。
名前	操作の名前（読み取り専用）
ネガティブ テスト	**ネガティブ・テストを有効にします。**実行するように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します。この場合，アプリケーションが SOAP エラーを発行し，SOAP 結果応答を発行していないことを確認する必要があります。 ［**期待される応答**］を選択します。 ➤ **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。ネガティブ・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。 詳細については，1046 ページの「ネガティブ・テストの概要」を参照してください。 **注意**：Service Test ライセンスがある場合にのみ利用できます。
引数リスト	出力引数と，値を格納する対応するパラメータのリスト。

< 出力引数名 > ノード

出力引数の値を格納するパラメータを指定できます。

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
追加(A)	新しい配列要素を出力引数に追加するための［配列要素の追加］ダイアログ・ボックスを開きます（出力配列内の親ノードを選択した場合にのみ表示されます）。詳細については，961 ページの「[配列要素の追加］ダイアログ・ボックス」を参照してください。
削除(R)	出力引数内の選択された配列要素を削除します（出力配列内の親ノードを選択した場合にのみ表示されます）。
名前	出力引数または配列の名前。
パラメータに戻り値を保存する	選択した引数の値をパラメータに保存します。カスタム・パラメータ名を指定するには，［**パラメータ**］フィールド内の標準設定の Param_*<arg_name>* を変更します。
なし	現在の引数の値を **nil=true** に設定します。
XML（配列の場合のみ）	引数値を含む XML コード。この値をパラメータ化するには，［ABC］アイコン（配列の場合にのみ使用可能）をクリックします。

［出力添付ファイル］ノード

出力添付ファイル・パラメータのプロパティを設定できます。これは，951 ページの「< 操作名 > ノード」で，出力添付ファイルを有効にした場合にのみ表示されます。

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
追加(A)	インデックス・ベースの新しい出力引数を追加します。これは，［**添付ファイルをインデックスとして保存する**］を選択する場合にのみ使用できます。
すべての添付ファイルを保存する	すべての出力添付ファイルを，次のプロパティを使用して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編集可能な名前。 ➤ **［Content-type］**：パラメータのタイプ（読み取り専用）。 ➤ **［Content-ID］**：パラメータの一意の ID（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	出力添付ファイルをインデックス・ベースのパラメータに保存します。インデックスを設定するには，パラメータの 1 つを選択し，右の表示枠でインデックス番号を変更します。

< 出力引数 > ノード

出力添付ファイル・パラメータのプロパティを設定できます。これは，951 ページの「< 操作名 > ノード」で，出力添付ファイルを有効にした場合にのみ表示されます。

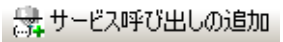
ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
添付ファイルの削除(D)	選択した添付ファイル・パラメータを削除します。添付ファイルをインデックスとして保存している場合，選択したアイテムのみが削除されます。
インデックス	パラメータのインデックス番号。このフィールドは，958 ページの「［出力添付ファイル］ノード」で［**添付ファイルをインデックスとして保存する**］を選択した場合にのみ有効になります。

UI 要素	説明
内容	添付ファイルを格納するパラメータの編集可能な名前。
Content-type	パラメータのコンテント・タイプ（読み取り専用）。
Content-ID	パラメータの一意のコンテント ID（読み取り専用）。

［入力添付ファイルの追加］ダイアログ・ボックス

このページでは，入力添付ファイルを Web 要求に追加できます。

利用方法	 をクリックして最上位ノード（操作名）を選択します。［**添付ファイル**］セクションで［**要求に追加する（入力）**］を選択します。
重要情報	添付ファイルをサービス呼び出しに追加する前に，サービスをインポートする必要があります。 参考情報については，923 ページの「Web サービス呼び出しの添付ファイル」を参照してください。
関連タスク	935 ページの「新しいサービス呼び出しを追加する（任意）」

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
データ取得先	データの場所。 ➤ **[ファイル]**：ファイルの場所。 ➤ 絶対パス：ファイルの完全パス。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]**：データを含むパラメータの名前。
Content-type	データを含むファイルのコンテント・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。［**値**］ボックスでは，値を手動で入力できます。また，一般的なコンテント・タイプのドロップダウン・リストも提供されます。
Content-ID	添付ファイルの一意の識別子。標準設定では，これは VuGen によって自動的に生成されます。必要に応じて，［**値**］ボックスに別の ID を指定できます。

［配列要素の追加］ダイアログ・ボックス

このページでは，既存の配列と同じ構造の引数配列に要素を追加できます。これは，入力引数と出力引数で使用できます。

入力要素では，既存の要素に基づいて新しい配列の値を設定できます。

利用方法	 をクリックします。配列になっている引数ノードを選択します。
重要情報	このダイアログ・ボックスを表示するには，引数ツリー内に配列が含まれている必要があります。
関連タスク	935 ページの「新しいサービス呼び出しを追加する（任意）」

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
名前	配列の親ノードの名前とインデックス。
開始インデックス	新しい配列要素を追加するときの開始インデックス。
要素	引数ツリーに追加する同一配列要素の数。
インデックスから値をコピー	特定の配列要素の値で新しい配列要素を作成します（入力引数でのみ使用できます）。

［Process Base64 Data - Simple Data］ダイアログ・ボックス

このダイアログ・ボックスでは，単純な base64 データのエンコード・オプションを設定できます。

利用方法	複雑でない単純な Base64 値の場合。 ➤ Web サービス呼び出しプロパティ内で，**Base64** タイプの入力引数を選択します。 ➤［**値**］オプションを選択します。 ➤［**エンコード テキストを埋め込む**］を選択します。 ➤［参照］ボタンをクリックします。
重要情報	複雑な配列では，963 ページの「［Process Base64 Data - Complex Data］ダイアログ・ボックス」を使用します。
関連タスク	935 ページの「新しいサービス呼び出しを追加する（任意）」

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
Decode to File ...	デコードしたテキストをファイルに保存できます。
Text to encode	複雑なデータでは，963 ページの「［Process Base64 Data - Complex Data］ダイアログ・ボックス」を使用します。
Encoding Options	エンコーディング方式のリスト。 **標準設定**：Unicode（UTF-8）
Encoded data	［**Text to encode**］表示枠のデータのエンコードされたバージョン。

［Process Base64 Data - Complex Data］ダイアログ・ボックス

このダイアログ・ボックスでは，複雑な base64 データのエンコード・オプションを設定できます。

利用方法	複雑な Base64 値の場合。 ➤ Web サービス呼び出しプロパティ内で，**Base64** タイプの複雑な入力引数を選択します。 ➤［**値**］オプションを選択し，［パラメータ］アイコンをクリックします。 ➤ 値をパラメータで置き換えます。 ➤［値］ボックス内の［パラメータ］アイコンを右クリックし，［**パラメータのプロパティ**］を選択します。 ➤［**データの編集**］ボタンをクリックします。 ➤ 対象の値セット・カラムで，B64 ボタンをクリックします。 **注 :**［チェックポイント］データ・グリッドと［サービスのエミュレーション］データ・グリッドからもアクセスできます。
重要情報	複雑でない単純なデータでは，962 ページの「［Process Base64 Data - Simple Data］ダイアログ・ボックス」を使用します。
関連タスク	935 ページの「新しいサービス呼び出しを追加する（任意）」

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
Encode	指定したファイルをエンコードします。
Decode to File ...	デコードしたデータをファイルに保存できます。通常，これは再生中に取得されるデータです。

UI 要素	説明
ファイル	参照またはファイルの内容によって，ファイルをエンコードします。 ➤ **[File path]**：エンコードするファイル。 ➤ **[Link to file]**：値を含むファイルを参照します。クリアすると，指定したファイルの内容が使用されます。内容がスクリプト・フォルダにコピーされます。 **ヒント**：10 KB を超えるテキストでは，［**Link to file**］を有効にします。
テキスト	指定したテキスト文字列をエンコードします。 ➤ **[Text to encode]**：エンコードする Base64 テキスト。テキストを入力すると，VuGen によってエンコードされたテキストが［**Encoded data**］表示枠に表示されます。 ➤ **[Encoding Options]**：エンコーディング方式のリスト。標準設定は Unicode（UTF-8）です。

アスペクト・リファレンス

このセクションの内容

アスペクトのリスト

使用できるテスト対象アスペクトを次の表に示します。

アスペクト名	説明
ポジティブ テスト	サービスの各操作を確認する完全なポジティブ・テストを生成します。
標準準拠	サービスが，WS-I や SOAP などの業界標準に準拠していることを確認します。
サービスの相互運用性	サービスの操作に，サポートされるすべての Web サービス・ツールキットとの相互運用性があることをテストします。 次のサブアスペクトが含まれます。 ➤ **[.NET Framework]**：標準設定値 / 期待値を使用してサービスのすべての操作を呼び出すことで，サービスに，.NET Framework WSE 2 ツールキットとの完全な相互運用性があることをテストします。 ➤ **[Axis/Java ベースの Web サービス]**：標準設定値 / 期待値を使用してサービスのすべての操作を呼び出すことで，サービスに，Axis 1.3 Web Services Framework との完全な相互運用性があることをテストします。
セキュリティ テスト	サービスのセキュリティをテストします。次のサブアスペクトが含まれます。 ➤ **[SQL インジェクションの脆弱性]**：関連するパラメータに SQL ステートメントとエラーを挿入することで，SQL インジェクションに対する脆弱性がサービスにあるかどうかを確認します。 ➤ **[クロスサイト スクリプティング（XSS）]**：サービスの機能を混乱させるコードを Web サイトに挿入することで，サービスのハッキングを試みます。

アスペクト名	説明
境界テスト	ネガティブ・テストの手法を使用して，データ，タイプ，パラメータ，および実際の SOAP メッセージを操作するテストを作成し，サービスを可能な限りテストします。 次のサブアスペクトが含まれます。 ➤ **[Extreme Values]**：無効なデータ・タイプをサービスに提供して，それが受け入れられないことを確認します。 ➤ **[Null Values]**：NULL パラメータをサービスに提供して，それが受け入れられないことを確認します。
パフォーマンス テスト	次のサブアスペクトが含まれます。 ➤ **[Stress Testing]**：アプリケーションにかけることができる最大の負荷をテストします。 ➤ **[Overload Sustainability Testing]**：アプリケーションに割り当てられたハードウェアが，予測されるユーザ数にどの程度対応できるかをテストします。 ➤ **[Volume Testing]**：システムが大量のデータ入力を処理できることをテストします。 ➤ **[Longevity Test]**：ピークに近い容量でトランザクションを実行する一貫した数の同時仮想ユーザを，システムが 24 時間以上持続できることをテストします。 ➤ **[Scalability Testing]**：さまざまなサーバ設定またはネットワーク・ハードウェア設定で，負荷テスト，過負荷テスト，ボリューム・テスト，有効期間テストを繰り返します。

テスト・ジェネレータ・ウィザードのユーザ・インタフェース

このセクションの内容

- ［配列要素の追加］ダイアログ・ボックス（961 ページ）
- ［Process Base64 Data - Simple Data］ダイアログ・ボックス（962 ページ）
- ［Process Base64 Data - Complex Data］ダイアログ・ボックス（963 ページ）
- テスト生成ウィザード（967 ページ）
- ［サービスの指定］画面（973 ページ）
- ［トラフィック情報を指定してください］画面（973 ページ）
- ［SSL 設定］ダイアログ・ボックス（974 ページ）

テスト生成ウィザード

このウィザードでは，サービスの要件またはテスト，あるいはその両方を作成できます。

利用方法	**［ファイル］＞［新規作成］。**［新規仮想ユーザ］ダイアログ・ボックスで［**SOA テスト ジェネレータ**］を選択し，［**作成**］をクリックします。
重要情報	このウィザードは，有効な Service Test ライセンスがインストールされている場合にのみ使用できます。
関連タスク	938 ページの「テストを自動的に生成する方法」
ウィザード・マップ	このウィザードには，次のページが含まれています。 ［ようこそ］＞［サービス］ページ＞ **［テスト対象アスペクトの選択］ページ**＞［場所の選択］ページ＞［サマリ］ページ＞［生成］ページ＞［完了］ページ

［サービス］ページ

このウィザード・ページでは，作成するエントリ（要件とテスト，テストのみ，または要件のみ）を指定できます。

重要情報	このウィザードに関する一般情報は，916 ページの「Web サービス・スクリプトの内容の追加の概要」に記載されています。
ウィザード・マップ	このウィザードには，次のページが含まれています。 ［ようこそ］>［**サービス**］ページ >［**テスト対象アスペクトの選択］ページ** >［場所の選択］ページ >［サマリ］ページ >［生成］ページ >［完了］ページ

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
追加(A)...	［サービスのインポート］ダイアログ・ボックスが開きます。詳細については，1014 ページの「［サービスのインポート］ダイアログ・ボックス」を参照してください。
Remove	選択したサービスを削除します。
< サービス・リスト >	インポートされたサービスのリスト。
サービス詳細	➤ **［WSDL の位置］**：WSDL のインポート元の場所。 ➤ **［説明］**：WSDL からのサービスの説明。

［テスト対象アスペクトの選択］ページ

このウィザード・ページでは，どのテスト対象アスペクトについて要件とテストを作成するかを選択できます。

重要情報	このウィザードに関する一般情報は，916 ページの「Web サービス・スクリプトの内容の追加の概要」に記載されています。
ウィザード・マップ	このウィザードには，次のページが含まれています。 ［ようこそ］> ［サービス］ページ > ［**テスト対象アスペクトの選択**］ページ > ［場所の選択］ページ > ［サマリ］ページ > ［生成］ページ > ［完了］ページ

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
☐	**アスペクトをクリアします。**アスペクトを除外します。
☑	**アスペクトを選択します。**アスペクトを含めます。親を選択すると，そのすべての子が含まれます。 **ヒント：**すべてのアスペクトを選択するには，親の［**テスト対象アスペクト**］ノードを選択します。
< 対象のアスペクト >	サービスで使用できるすべてのアスペクトのツリー・ビュー。子アスペクトを表示または非表示にするには，［**展開**］⊞ または［**折りたたみ**］⊟ ボタンを使用します。

［場所の選択］ページ

このウィザード・ページでは，生成したテストの保存先を選択できます。

重要情報	このウィザードに関する一般情報は，916 ページの「Web サービス・スクリプトの内容の追加の概要」に記載されています。
ウィザード・マップ	このウィザードには，次のページが含まれています。 ［ようこそ］>［サービス］ページ > **［テスト対象アスペクトの選択］ページ** >［**場所の選択**］ページ >［サマリ］ページ >［生成］ページ >［完了］ページ

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
Connect...	［Application Lifecycle ManagementConnection - Server Connection］ダイアログ・ボックスを開きます。詳細については，258 ページの「［HP ALM 接続］ダイアログ・ボックス」を参照してください。
参照(B)...	➤ Application Lifecycle Management への出力の場合：ALM リポジトリ内の対象のフォルダを探します。 ➤ ファイル・システムへの出力の場合：ファイル・システム上の対象のフォルダを探します。
Application Lifecycle Management に出力	スクリプトを Application Lifecycle Management のリポジトリに格納します。 ➤ **［名前］**：スクリプトの格納先のサブフォルダ名。 ➤ **［位置］**：親フォルダ。［**名前**］は，この場所のサブフォルダになります。
ローカル ファイル システムに出力	スクリプトをファイル・システムに格納します。 ➤ **［名前］**：スクリプトの格納先のサブディレクトリ名。 ➤ **［位置］**：親ディレクトリ。［**名前**］は，この場所のサブディレクトリになります。

［生成］ページ

このウィザード・ページでは，作成するエントリ（要件とテスト，テストのみ，または要件のみ）を指定できます。

重要情報	このウィザードに関する一般情報は，916 ページの「Web サービス・スクリプトの内容の追加の概要」に記載されています。
ウィザード・マップ	このウィザードには，次のページが含まれています。 ［ようこそ］>［サービス］ページ > **［テスト対象アスペクトの選択］ページ** >［場所の選択］ページ >［サマリ］ページ >［**生成**］ページ >［完了］ページ

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
Generate	選択したアスペクトに基づいて，リストに含まれたスクリプトを生成します。
< テスト・リスト >	生成するスクリプトのリスト。項目を除外するには，該当の項目のチェック・ボックスをクリアします。

［完了］ページ

このウィザード・ページでは，作成するエントリ（要件とテスト，テストのみ，または要件のみ）を指定できます。

重要情報	このウィザードに関する一般情報は，916 ページの「Web サービス・スクリプトの内容の追加の概要」に記載されています。
ウィザード・マップ	このウィザードには，次のページが含まれています。 ［ようこそ］>［サービス］ページ > **［テスト対象アスペクトの選択］ページ** >［場所の選択］ページ >［サマリ］ページ >［生成］ページ >［**完了**］ページ

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
	選択したスクリプトを VuGen で開きます。
< 生成されたスクリプト >	生成されたスクリプトのリスト。項目を除外するには，該当の項目のチェック・ボックスをクリアします。

［トラフィックの分析］のユーザ・インタフェース

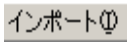
このセクションの内容

［サービスの指定］画面

このウィザード画面では，Web サービスを選択してトラフィック・ベースのスクリプトに関連付けることができます。

利用方法	［**トラフィックの分析**］ボタン
関連タスク	934 ページの「セッションを記録する（任意）」

ユーザ・インタフェース要素の説明は次のとおりです（ラベルのない要素は山括弧で囲んで示します）。

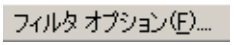
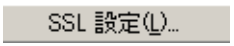
UI 要素	説明
	サービスに関する詳細情報を指定するための［**サービスの管理**］ダイアログ・ボックスを開きます。詳細については，1009 ページの「[サービスの管理］ダイアログ・ボックス」を参照してください。
	［**サービスのインポート**］ダイアログ・ボックスを開きます。詳細については，1014 ページの「[サービスのインポート］ダイアログ・ボックス」を参照してください。
	選択したサービスをリストから削除します。
< **サービス・リスト** >	使用可能なサービスのリスト。 ➤ **[サービス名]**：サービスのネイティブ名。 ➤ **[WSDL の位置]**：WSDL のソース。

［トラフィック情報を指定してください］画面

このウィザード画面では，受信または送信トラフィック用のキャプチャ・ファイルを指定できます。

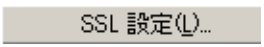
利用方法	［**トラフィックの分析**］ボタン，［**次へ**］
関連タスク	934 ページの「セッションを記録する（任意）」

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
	**[参照]**：インポートするキャプチャ・ファイルを選択できます。
	［記録オプション］ダイアログ・ボックスで［**トラフィック フィルタ**］ノードを開きます。これを使用して，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ます。詳細については，313 ページの「記録オプション」を参照してください。
	974 ページの「[SSL 設定］ダイアログ・ボックス」を開きます。これを使用して，SSL 証明書を追加し，セキュア・サーバからのトラフィックを分析できます。
キャプチャ ファイル	サーバ・トラフィックを含むキャプチャ・ファイルの名前。通常は，**cap** 拡張子が付きます。
受信トラフィック	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	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。
アクション内に記録	作成するスクリプトの挿入先のセクション。**vuser_init**，**Action**，または **vuser_end**。繰り返すアクションについては，**Action** セクションを使用します。初期化のステップでは，**vuser_init** を使用します。

［SSL 設定］ダイアログ・ボックス

このダイアログ・ボックスでは，トラフィック・ファイル用の SSL 証明書を設定できます。

利用方法	［**トラフィックの分析**］ボタン，［**次へ**］ 
関連タスク	934 ページの「セッションを記録する（任意）」

ユーザ・インタフェース要素の説明は次のとおりです（ラベルのない要素は山括弧で囲んで示します）。

UI 要素	説明
	**[証明書の追加]**：証明書リストに新しい行を追加します。
	**[削除]**：選択した証明書を削除します。
< 証明書リスト >	証明書のプロパティ。次の情報を指定します。 ➤ **[IP]**：証明書ファイルをホストするマシンの URL または IP アドレス。 ➤ **[ポート]**：証明書ファイルをホストするマシンのポート。 ➤ **[ファイル]**：秘密鍵を含む証明書ファイル（**pem** 拡張子付き）のパス。ファイルを検索するには，[**参照**] ボタンを使用します。 ➤ **[パスワード]**：証明書ファイルを復号するためのパスワード。

36

Web サービス - サービスの管理

本章の内容

概念

タスク

リファレンス

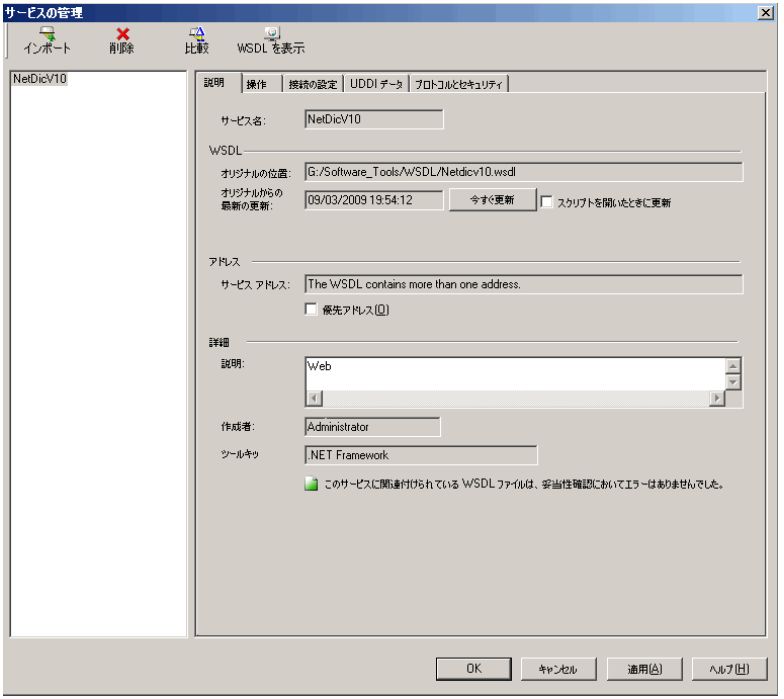
概念

サービスの管理の概要

［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。

［サービスの管理］ウィンドウには，サービスのインポート，削除，および比較をすばやく行うためのボタンがあります。



このタブでは，WSDL のプロパティを設定できます。

説明

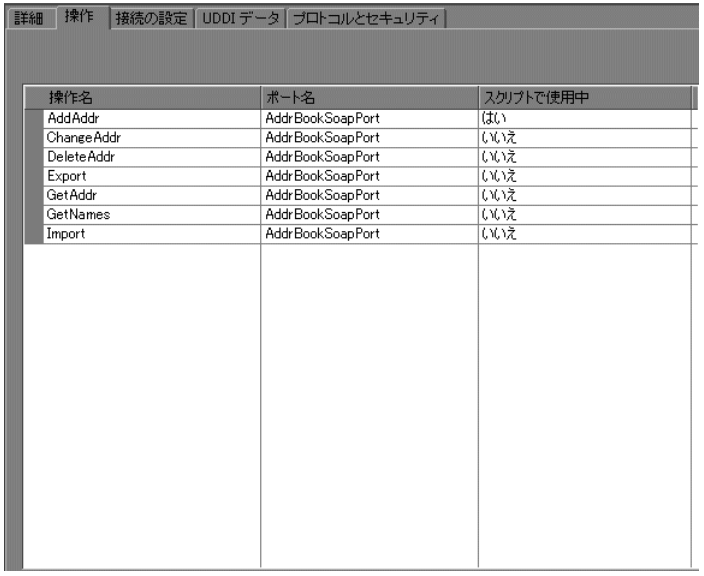
［**説明**］タブには，サービスに関する次のような情報が表示されます。

- **［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（読み取り専用）。
- **［オリジナルからの最新の更新］**：ローカル・コピーが元のソース（読み取り専用）から更新された最終日付。手動でバージョンを更新することも，テストを再度開くたびに自動的に取得することもできます。
- **［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。必要に応じて，WSDL ファイルで指定したエンドポイントをオーバーライドできます。

  これは，Service Test ライセンスで使用できるエミュレーション・サービスを実装する場合に便利です。VuGen with Service Test では，Web サービス呼び出しの targetAddress として**優先アドレス**が使用されます。このオーバーライドは，すべての Web サービス呼び出しに影響します。特定の Web サービス呼び出しに別のターゲット・アドレスを使用するには，該当するステップのプロパティでそのアドレスを指定します。詳細については，1145 ページの「エミュレーション・サービスを作成する方法 - ワークフロー」を参照してください。
- **［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（読み取り専用）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。



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。次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 ツールキット用 Kerberos

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，997 ページの「サービスをインポートする」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。

プロトコルとセキュリティの設定

［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。詳細については，999 ページの「セキュリティ・シナリオを設定する（任意）」を参照してください。

このセクションには次の内容も含まれます。

 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。

インポート・メカニズムでは，次の情報が必要になります。

➤ **［ソース］:** WSDL のソース：URL，ファイル，UDDI，または Application Lifecycle Management UDDI（Universal Description, Discovery, and Integration）は，サービスの共通リポジトリです。サービス・プローカは公開された 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。

- **[位置]**：WSDL のパスまたは URL。手動または参照で入力します。
- **[ツールキット]**：以降のすべてのインポートおよび再生で使用される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）。ツールキットの設定では，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 VuGen に指示します。

  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。
- **[接続の設定]**：認証またはプロキシ・サーバ情報。この設定は，セキュア・サーバに WSDL がある場合や，プロキシ・サーバを経由して WSDL にアクセスする必要がある場合に便利です。

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。

タスクの詳細については，997 ページの「サービスを追加および管理する方法」を参照してください。

比較レポート

VuGen では，ファイル間の相違が比較レポートに一覧表示されます。

比較設定を設定できます（比較時に無視する項目を指定）。詳細については，1017 ページの「[XML / WSDL 比較] ダイアログ・ボックス」を参照してください。

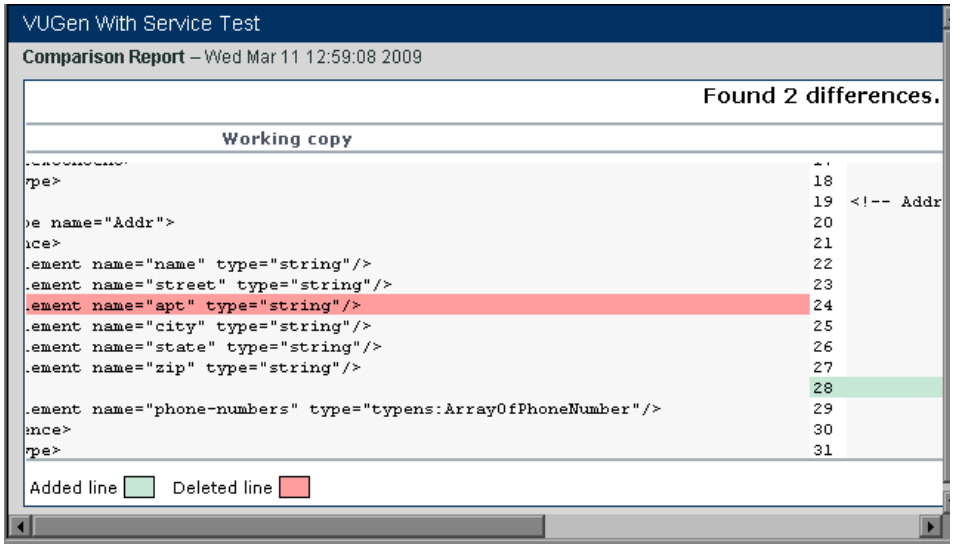
WSDL 比較レポートには，[**Working Copy**] と [**Original File**] の 2 つのカラムがあります。[Working Copy] はスクリプトと一緒に格納されている WSDL です。一方，[Original File] は元の場所（ネットワーク・ファイル・パスまたは URL）にある WSDL です。

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）
- **ピンク**：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。



Web 参照アナライザ

多くの WSDL ファイルでは，ほかのファイル（XSD やほかの XML ファイルなど）を参照しています。スクリプトを実行する前に，どのファイルを参照しているのか，およびそれらが使用できるかどうかを確認できます。

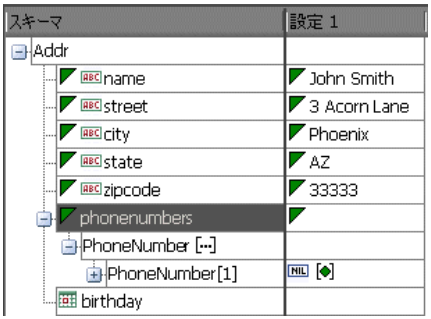
VuGen の WSDL 参照アナライザは，WSDL の依存関係をチェックして，［WSDL 参照アナライザ］ウィンドウやログ・ファイルにそれらをリストします。

アナライザは，WSDL ファイルとその依存ファイルを zip アーカイブ・ファイルに格納します。依存関係情報はログ・ファイルに保存され，アナライザのウィンドウにはそのパスが表示されます。タスクの詳細については，999 ページの「WSDL の依存関係を分析する方法」を参照してください。

XML の編集

XML エディタでは，XML および XSD スキーマの表示と編集ができます。このエディタは，Service Test ライセンスでのみ使用できます。詳細については，HP のサポートにお問い合わせください。

VuGen 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 妥当性検証など，VuGen with Service Test のほかの機能も統合されています。

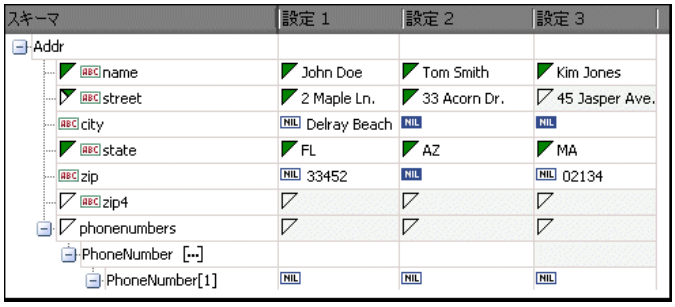
要素の値を設定するには，XML を手動で編集するか，XML ファイルを値と一緒にインポートします。

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。これにより，反復ごとに送信する値を変えることが可能になるため，一部の反復には特定の配列要素を含めて，それ以外の反復ではその配列要素を除外できます。

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。

タスクの詳細については，1002 ページの「スキーマを編集する方法」を参照してください。

XML 妥当性検証の概要

Web サービスを実行する場合，Web サービスによってサーバへのデータ要求とサーバからのデータ応答が転送されます。Web サービス・スクリプトでは，ハードコード化された値ではなくデータのパラメータを使用するのが一般的です。各パラメータには，転送される実際の XML データが含まれています。パラメータを使用することで，さまざまな値を使用して Web サービスをテストできます。

VuGen には，パラメータの XML データを検証するためのインタフェースが用意されています。検証内容は次のとおりです。

- XML が整形式であるかどうか
- チェックポイントの値が正しいかどうか
- XML スキーマに準拠しているかどうか（手動でスキーマをロードした場合のみ）

整形式であるためには，World Wide Web Consortium（W3C）で定義されている 100 を超えるさまざまなルールに XML ドキュメントが準拠している必要があります。たとえば，各要素には開始タグと終了タグが必要である，という重要なルールがあります。

チェックポイント検証では，XML ファイルの引数値と期待値が比較されます。期待値は，手動で指定することも，以前のセッションからロードすることもできます。

XML スキーマは，XML オブジェクトの属性と要素の相互関係を表しています。XML ファイルがそのスキーマに準拠しているかどうかを確認するには，XML ファイルをその XSD（XML スキーマ定義）ファイルとともに指定します。この確認は，手動でロードした XSD にのみ適用され，VuGen によって自動的に検出された XSD には適用されません。

このセクションには次の内容も含まれます。

- 987 ページの「XSD スキーマ検証」
- 988 ページの「チェックポイントと期待される応答」
- 992 ページの「検証結果」
- 995 ページの「REST サービスと XML 妥当性検証」

XSD スキーマ検証

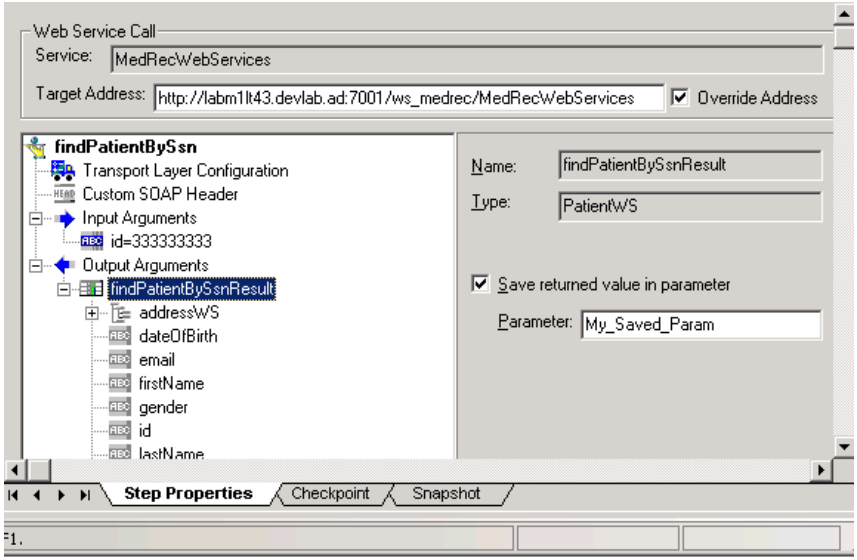
完全な XML 妥当性検証を実行するには，パラメータを選択し，そのパラメータに対応する XSD スキーマを指定します。XML 妥当性検証でパラメータを指定する場合，入力パラメータまたは出力パラメータのいずれかを使用できます。

入力パラメータでは，以前に保存したパラメータを使用します。［**XML ソース**］ドロップダウン・リストから［**パラメータの作成 / 選択**］を選択して新しいパラメータを作成することもできます。ファイルの内容が XML の場合，パラメータ・タイプは XML パラメータかファイル・タイプ・パラメータにする必要があります。

このパラメータは，ANY 型要素の XML であることが理想です。VuGen は，XSD を解析して，WSDL がスキーマに準拠しているかどうかを判断します。

［プロパティ］タブで出力パラメータを作成します。出力値の選択時に［**パラメータに戻り値を保存する**］を有効にします。出力パラメータの最上位ノードを選択する場合，VuGen によってその値が XML パラメータとして保存されます。

次の例は，パラメータ **My_Saved_Param** に保存される出力配列を示しています。



パラメータの定義の詳細については，263 ページの「パラメータ」を参照してください。

［XML の妥当性検証］ダイアログ・ボックスで，保存したパラメータを［**XML ソース**］に入力します。

チェックポイントと期待される応答

次のセクションでは，チェックポイントのタイプや，期待される応答の検証にチェックポイントがどのように役立つかについて説明します。

基本的なチェックポイントと詳細なチェックポイント

チェックポイントの基本検証と詳細検証の両方を実行できます。

基本検証の場合，VuGen によって［**期待値**］カラムの値に完全に一致する値が検索されます。記録時のスナップショットまたは再生時のスナップショットから期待値をロードできます。

［**詳細なチェックポイント**］を使用して，非リーフ・ノードのチェックポイントを検証するか，正規表現の用語で期待値を定義します。

［**詳細なチェックポイント**］セクションに XPATH クエリを入力して詳細検証の値を定義します。最初の XPATH 式を取得するには，［**基本検証**］の行からコピーします。右クリック・メニューから［**行 XPath のコピー**］を選択し，［**詳細検証**］セクションにその内容を貼り付けます。

同じステップに基本検証と詳細検証の両方を定義できます。

非リーフ・ノードを選択する場合，そのノードの下の XML をすべて入力する必要があります。

仮想ユーザ・スクリプトの場合，VuGen では **Value=** と **Expression=** の正規表現で完全一致を表します。

```
BEGIN_CHECKPOINTS,
        CHECKPOINT, "XPATH=/AddResult[1]", "Value=50"
        CHECKPOINT, "XPATH=AddResult", "Expression=Hel*?",
END_CHECKPOINTS,
```

XPATH 式

チェックポイントの式では，ノードセットと拡張 XPath の構文がサポートされています。

ノードセットの式は，1 つの XML ノードまたは一連の XML ノードとして評価される XPath 式です。次に例を示します。

```
/a/b/c
/x/y[1]
//a/b
```

VuGen では，完全な XPath 式（整数として評価される **count(/a/b/c)** やブール値として評価される **count(/a/b/c) < 3** など）もサポートされています。この XPATH のサポートにより，数値を比較できます。たとえば，結果が 8 ～ 16 であるかどうかを検証するには，次の式を使用します。

```
number(/a/b/c) > 8 and number(/a/b/c) < 16
```

期待される応答

XML 妥当性検証を使用して，Web サービス呼び出しの応答全体を確認できます。この機能は標準のチェックポイントとほぼ同じですが，各引数を個々のパラメータに保存するのではなく，応答全体を 1 つのパラメータに保存する追加機能があります。

応答を検証するには，検証する Web サービス呼び出しの後に XML 妥当性検証ステップを追加します（Web サービス呼び出しの直後にする必要はありません）。

Web サービス呼び出しごとに，VuGen によって XML 応答全体が 1 つのパラメータに保存されます。これにより，1 つのステップで応答全体を検証できます。Web サービス呼び出しのステップ名に **_Response** サフィックスを加えた名前がパラメータ名になります。たとえば，ステップ名が **GetAddr_101** の場合，応答パラメータは **GetAddr_101**_Response になります。

このパラメータは実際の SOAP 応答ではありません。SOAP 応答には，応答値と SOAP エンベロープ全体が含まれています。この応答には XML 値はありますが，SOAP ヘッダや SOAP エンベロープはありません。

応答は各 Web サービス呼び出しで一意であるため，保存したパラメータごとに個別の検証ステップを追加する必要があります。応答パラメータを選択すると，スキーマ検証を実行するために VuGen によって自動的に XSD がロードされます。

設計時に手動で値をパラメータに保存した個々の引数を検証することもできます。このパラメータは［**XML ソース**］ドロップダウンに表示されます。

検証の比較演算子

XML 妥当性検証にチェックポイントを使用して XSD の期待値をチェックします。このとき，次の比較演算子を使用できます。

- **［次と一致する］**：テキストに完全に一致します。
- **［次と一致しない］**：テキストに一致しません。
- **［次を含む］**：テキストに部分一致します。
- **［次で開始］**：返されたテキストが次のフレーズで始まる場合に一致します。
- **［次で終了］**：返されたテキストが次のフレーズで始まる場合に一致します。
- **［正規表現］**：正規表現の構文を使用した文字列パターンに一致します。

数値要素の場合，標準的な数値関係（[次と等しい]，[次と等しくない]，[次より大きい]，[次より小さい］など）のいずれかを指定します。

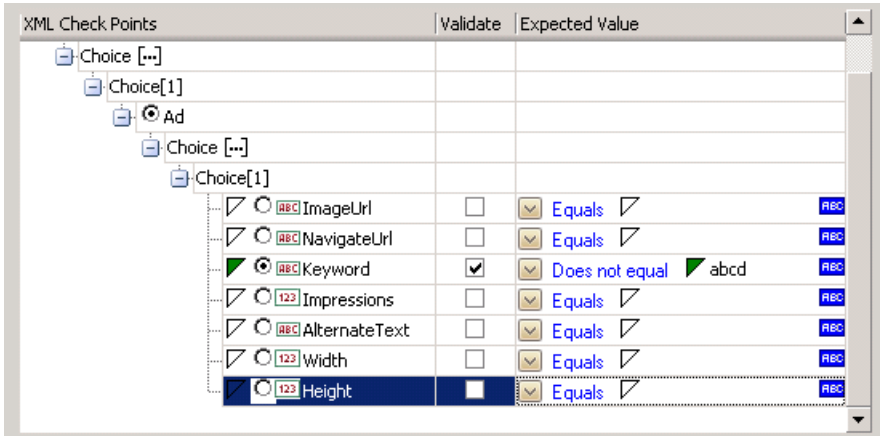
正規表現を使用した高度なクエリの定義の詳細については，989 ページの「XPATH 式」を参照してください。

オプションの要素

XML 妥当性検証ツールでは，オプションの値が存在するかどうかを確認できます。値が存在すること確認する必要がある場合もあれば，値が存在しないことを確認する必要がある場合もあります。

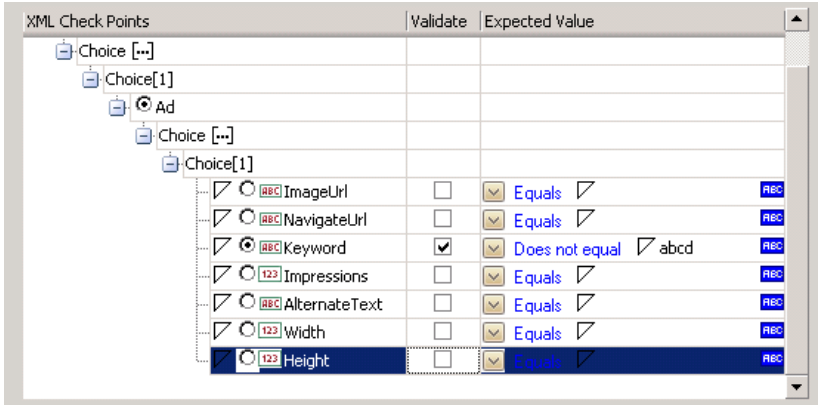
オプション要素を含める場合，アイコンを塗りつぶすと，サービスによって値が返されたかどうかが再生でチェックされます。オプション要素を除外する場合，オプション・アイコンをクリアすると，サービスによって値が返されていなかったかどうかが再生で検証されます。

次の例では，オプション要素が含まれていて，その値が **abcd** になっています。



再生時，VuGen によって要素の値がチェックされます。再生時に値が検出されない場合，含まれている要素の値が検出されなかったことを示すエラー・メッセージが VuGen によって発行されます。テスト結果レポートで詳細情報を確認できます。詳細については，1002 ページの「スクリプトを実行および結果を表示する」を参照してください。

次の例では，サーバから値が返されていないかどうかが VuGen によってチェックされます。要素の値が検出された場合，除外される要素の値が VuGen によって検出されたことを示すエラー・メッセージが表示されます。



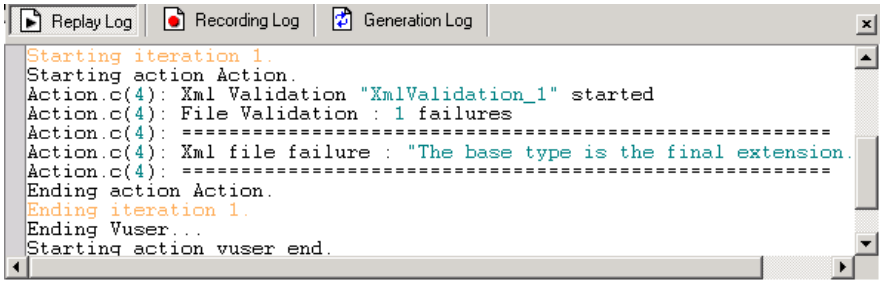
両方のタイプのテスト（包含と除外）で，［**検証**］カラムのチェックボックスを選択する必要があります。選択しないと，検証ツールでその要素が無視されます。

検証結果

XML 妥当性検証ステップをスクリプトに追加すると，実行時に VuGen によって検証が行われます。いくつかの方法で検証結果を確認できます。

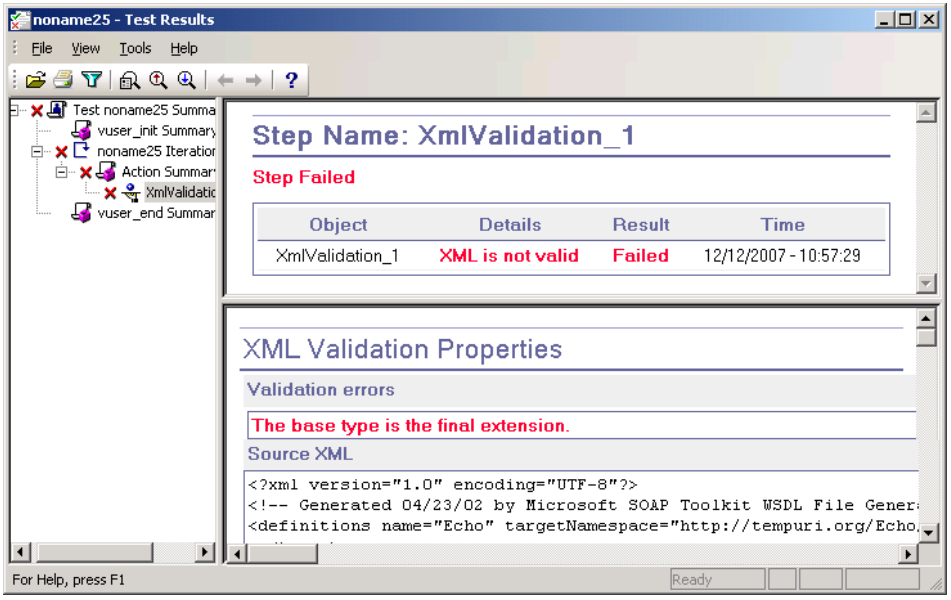
再生ログ

再生時，VuGen によって再生ログに結果が表示されます。［拡張ログ］の［実行環境の設定］を有効にして，検証の詳細情報を取得することをお勧めします。



テスト結果レポート

テスト結果レポートには，検証ステップの結果と XML ソース・コードが表示されます。このレポートを開くには，［**表示**］>［**テスト結果**］を選択します。



戻り値

再生ログを表示する以外に，**soa_xml_validate** の戻り値を評価して検証結果を確認することもできます。考えられる戻り値を次の表示に示します。

値	説明
0	検証は成功しました。XML は整形式で，準拠していて，チェックポイントは正常な状態です。
1	パラメータが空か存在しません。
2	XML が整形式ではありません。
3	XML は整形式ですが，XSD が整形式ではありません。
4	XML が XSD に準拠していません。

値	説明
5	XML 文字列の値が，チェックポイントの XPATH クエリで指定した値に一致しません。
6	XSD ファイルを見つけることができませんでした。
7	この表にないエラー（内部エラーまたは不明なエラーなど）が発生しました。
8	XSD ファイルが整形式ではなく，XML 文字列の値が，チェックポイントの XPATH クエリで指定した値に一致しません。
9	XSD ファイルに対する XML 妥当性検証に失敗しました。また，XML 文字列の値が，チェックポイントの XPATH クエリで指定した値に一致しません。

関数とその戻り値の詳細については，『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。

REST サービスと XML 妥当性検証

SOAP 以外の Web サービス（REST など）に XML 妥当性検証を使用できます。次の例は，REST Web サービスと XML 妥当性検証のコラボレーションを示しています。

```
// カンマで区切られた内容を Design Run タイプのパラメータに保存します。
web_reg_save_param("WCSParam_Text1",
          "LB=",
          "RB=",
          "RelFrameId=1",
          "Search=Body",
          "IgnoreRedirections=Yes",
          LAST);

//URL として送信されるサービス要求
       web_url("xml",
"URL=http://ecs.amazonaws.com/onca/
xml?Service=AWSECommerceService&Operation=ItemSearch&AWSAccessKeyId=12
3456789ABCD&SearchIndex=Books&Keywords=Cortisol&Author=Tolstoy",
          "Resource=0",
          "RecContentType=text/xml",
          "Referer=",
          "Snapshot=t1.inf",
          "Mode=HTML",
          LAST);

// 検証ステートメントにパラメータを値を使用します。
          soa_xml_validate ("StepName=XmlValidation_1",
          "Snapshot=t3e9876543214.inf",
          "XML={WCSParam_Text1}",
          "StopOnValidationError=0",
          "XSD=C:¥¥Bugs¥¥67571_Rest¥¥AWSECommerceService.xsd",
           BEGIN_CHECKPOINTS,
    …
          END_CHECKPOINTS,
          LAST);
```

POST，PUT，または DELETE アクションを使用するには，*オンライン関数リファレンス*で説明しているように **web_custom_request** を使用します。

XML クエリ

特定の要素と値を見つけて調査するために XML ツリーを検索できます。

XML ファイルを検索するには，標準の XPATH 構文のクエリを使用します。有効な XPATH クエリを作成するには，組み込みのクエリ・ビルダを使用します。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。

タスク

サービスを追加および管理する方法

このタスクでは，テストで呼び出すことができるサービスのリストを作成する方法について説明します。［サービスの管理］ウィンドウを使用して，サービスのインポートおよび設定を行います。

このタスクでは，次の手順を実行します。

- 997 ページの「［サービスの管理］ダイアログ・ボックスを開く」
- 997 ページの「サービスをインポートする」
- 998 ページの「WSDL を理解する」
- 998 ページの「WSDL の更新を確認する（任意）」
- 998 ページの「サービス・アドレスをオーバーライドする（任意）」
- 999 ページの「セキュリティ・シナリオを設定する（任意）」

［サービスの管理］ダイアログ・ボックスを開く

［**SOA ツール**］>［**サービスの管理**］を選択するか，ツールバー・ボタンをクリックして，［サービスの管理］ダイアログ・ボックスを開きます。

サービスをインポートする

［**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 ソースを選択して場所を参照します。

URL タイプのインポートの場合，［参照］ボタンをクリックして新しいブラウザを開きます。WSDL に移動してブラウザを閉じます。この操作で［場所］ボックスに URL が配置されます。詳細については，1014 ページの「［サービスのインポート］ダイアログ・ボックス」を参照してください。

サービスで認証が必要な場合やプロキシを使用している場合，WSDL をインポートする前にこれらを設定します。［サービスのインポート］ダイアログ・ボックスを展開し，［**設定**］をクリックします。詳細については，1013 ページの「［接続の設定］ダイアログ・ボックス」を参照してください。

テストに含めるすべてのサービスについて，この手順を繰り返します。

WSDL を理解する

WSDL の理解を深めます。1011 ページの「［説明］タブ」で説明しているように WSDL の詳細情報を表示します。

［**WSDL を表示**］をクリックして，ローカルに保存されている WSDL ファイルを Internet Explorer で開き，その構造を学習します。

WSDL の更新を確認する（任意）

比較ツールを使用して，最後にインポートまたは更新してから WSDL に変更がないかどうかを確認します。

まず，比較オプションを設定します。［**SOA ツール**］>［**SOA 設定**］>［**XML / WSDL 比較**］をクリックします。無視する差異を指定します。詳細については，1017 ページの「［XML / WSDL 比較］ダイアログ・ボックス」を参照してください。

［サービスの管理］ウィンドウで［**比較**］をクリックして，WSDL の作業用コピーと元の場所にある WSDL を比較したレポートを開きます。

比較レポートで変更を見つけたら，［**今すぐ更新**］をクリックしてソースから WSDL の最新バージョンを取得します。

サービス・アドレスをオーバーライドする（任意）

［**サービス アドレス**］ボックスでアドレスを表示します。これは，WSDL から取得した標準設定のエンドポイント・アドレスです。このアドレスをオーバーライドする場合，［**優先アドレス**］を選択してサービス要求の別のエンドポイント・アドレスを入力します。

標準設定のアドレスに戻すには，［**優先アドレス**］オプションをクリアします。詳細については，1011 ページの「［説明］タブ」を参照してください。

セキュリティ・シナリオを設定する（任意）

［**プロトコルとセキュリティ**］タブをクリックして，WS-Security または別のタイプのセキュリティ・シナリオを使用します。

詳細については，第 38 章，「Web サービスーセキュリティ」。を参照してください。

WSDL の依存関係を分析する方法

このタスクでは，参照アナライザを使用して WSDL の依存関係を調べる方法について説明します。ユーザ・インタフェースの詳細については，1020 ページの「[WSDL 参照アナライザ］ダイアログ・ボックス」を参照してください。

このタスクでは，次の手順を実行します。

- 999 ページの「参照アナライザを開く」
- 998 ページの「WSDL の更新を確認する（任意）」
- 1000 ページの「分析を開始する」
- 1000 ページの「ログを表示する」

1 参照アナライザを開く

［**SOA ツール**］>［**WSDL 参照アナライザ**］を選択します。

2 ソースとターゲットを選択する

［**WSDL ファイルの選択**］ボックスで，分析する WSDL の場所を指定します。

［**出力ファイル パス**］ボックスで，zip ファイルの場所を指定します。

3 分析を開始する

［**分析を開始**］をクリックします。アナライザによって，すべての依存関係とそのパスが出力ウィンドウに表示されます。

4 ログを表示する

ログ・ウィンドウに結果が表示されます。結果をクリアして別の分析を実行するには，［**ログをクリア**］をクリックします。

XML の妥当性を検証する方法

このタスクでは，XML が整形式であるかどうかの検証方法や，期待される応答の確認方法について説明します。概念の詳細については，986 ページの「XML 妥当性検証の概要」を参照してください。

このタスクでは，次の手順を実行します。

- 1001 ページの「XML 妥当性検証ツールを開く」
- 1001 ページの「パラメータを選択する」
- 1001 ページの「チェックポイントを有効にする」
- 1001 ページの「期待値を設定する」
- 1001 ページの「詳細なチェックポイントを設定する（任意）」

1 XML 妥当性検証ツールを開く

テスト・ステップ（スクリプト・ビューの Web サービス呼び出し）を選択し，上部のツールバーにある［**XML の妥当性検証**］ボタンをクリックします。ユーザ・インタフェースの詳細については，1018 ページの「[XML の妥当性検証］ダイアログ・ボックス」を参照してください。

2 パラメータを選択する

［XML ソース］ボックスで，ドロップダウン・リストから応答パラメータを選択します。Web サービス呼び出しの完全な応答の場合，応答パラメータ（_Response サフィックス付きの Web サービス呼び出しのステップ名）を選択します。個々のパラメータの応答引数を手動で保存した場合，その応答引数を選択することもできます。

3 チェックポイントを有効にする

［**チェックポイント**］オプションを選択します（選択されていない場合）。［**検証**］カラムで，検証する引数にチェックを入れます。すべての引数とその子の検証を有効にするには，［**すべて選択**］をクリックします。

4 期待値を設定する

a マウスを引数名の上に移動すると，そのデータ型がわかります。

b ［**次と一致する**］や［**次と一致しない**］などの比較演算子を選択します。

c 期待値を指定します。特定のデータ型（ブール型や日付型など）には値を選択するためのインタフェースがあります。

d 値をパラメータ化するには，［**ABC**］アイコンをクリックして新しいパラメータを作成します。

5 詳細なチェックポイントを設定する（任意）

a ［**詳細なチェックポイント**］オプションを選択します。

b 既存の引数から XPath 式をコピーします。たとえば，チェックポイント・グリッドで引数値を選択し，ショートカット・メニューから［XPath のコピー］を選択します。

c XPath 式を［**XPath クエリ**］カラムに貼り付けます。詳細については，989 ページの「XPATH 式」を参照してください。

d 検証方法（[**次を含む**]，[**正規表現**]，または［**完全一致**]）を選択します。

e 期待値を指定します。

6 検証ステップを完了する

［OK］をクリックして XML 妥当性検証ステップを追加します。これは **soa_xml_validate** としてスクリプトに表示されます。

7 テスト実行を準備する

拡張ログ・オプションを設定します。F4 キーをクリックして［実行環境設定］を開きます。［**一般**］>［**ログ**］ノードを選択し，［**拡張ログ オプション**］を選択します。サブ・オプションを有効にする必要はありません。［OK］をクリックします。

8 スクリプトを実行および結果を表示する

a F5 をクリックしてスクリプトを実行します。再生ログを表示してエラーがないか確認します。

b テスト結果を表示します。［テスト結果］ウィンドウが自動的に表示されない場合は，［**表示**］>［**テスト結果**］を選択します。

c XML 妥当性検証ステップを展開します。ステータス，ソース XML，およびチェックポイントの結果を表示します。

スキーマを編集する方法

このタスクでは，XML エディタを使用して XML および XSD ファイルの値をロードする方法について説明します。このエディタは，Service Test ライセンスでのみ使用できます。ユーザ・インタフェースの詳細については，1021 ページの「[XML エディタ］ダイアログ・ボックス」を参照してください。

このタスクでは，次の手順を実行します。

➤ 1003 ページの「XML エディタを開く」
➤ 1003 ページの「スキーマをロードする」
➤ 1003 ページの「ルートを選択する」
➤ 1003 ページの「値を割り当てる」
➤ 1003 ページの「値セットを追加する（任意）」
➤ 1004 ページの「XML を編集する（任意）」

➤ 1004 ページの「値セットを削除する（任意）」

➤ 1004 ページの「カラムとその値を保存する（任意）」

1 XML エディタを開く

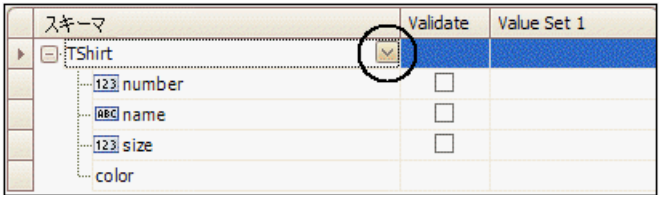
［**SOA ツール**］>［**XML エディタ**］を選択します。エディタが開きます。

2 スキーマをロードする

［**スキーマ**］>［**ロード**］を選択し，値セットを作成および編集する XSD ファイルを選択します。

3 ルートを選択する

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



4 値を割り当てる

➤ ［**値**］カラムにエントリを入力します。手動で値を入力するには，マウスを要素のアイコン上に移動して，そのデータ型を確認します。データ型の詳細については，1005 ページの「データ型を確認する」を参照してください。

➤ XML ファイルから値をロードするには，［**Load XML from file into the selected column**］ボタンをクリックします。

5 値セットを追加する（任意）

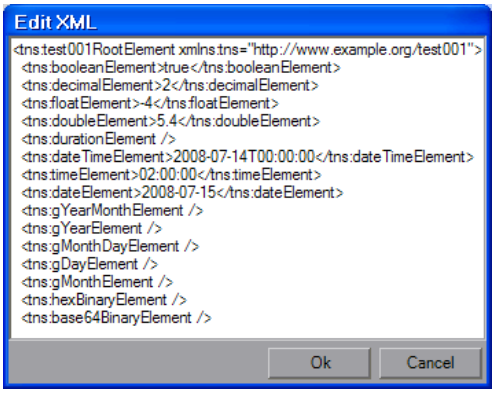
［**カラム**］>［**追加**］を選択するか，［列の追加］ボタンをクリックして，別の値セットを追加します。手動で，または前のステップで説明したように XML をロードして値を設定します。

選択したカラムと同じ値のカラムを追加するには，［カラムの複製］ボタンをクリックします。

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。

6 XML を編集する（任意）

実際のスキーマを編集するには，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。



7 値セットを削除する（任意）

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

8 カラムとその値を保存する（任意）

現在の値セットの値で XSD を保存するには，カラムを選択して［**XML**］>［**保存**］を選択するか，［**Save XML from the selected column**］ボタンをクリックします。

XML グリッドの値を編集する方法

このタスクでは，XML グリッドで作業する方法について説明します。XML グリッドは VuGen with Service Test コンポーネントのいくつか（XML エディタ，パラメータ化，XML 妥当性検証，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，263 ページの「パラメータ」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。

XML 妥当性検証では，グリッドを使用して XSD スキーマが表示されます。サービス・エミュレーションでは，グリッドを使用してルールが表示されます。

XML エディタ，XML 妥当性検証，およびサービス・エミュレーションのツールは，Service Test ライセンスでのみ使用できます。詳細については，HP のサポートにお問い合わせください。

ユーザ・インタフェースの詳細については，1021 ページの「[XML エディタ] ダイアログ・ボックス」を参照してください。

このタスクでは，次の手順を実行します。

- 1005 ページの「XML グリッドを入力する」
- 1005 ページの「データ型を確認する」
- 1006 ページの「base64 プロパティを設定する」
- 1006 ページの「オプション要素を包含または除外する」
- 1006 ページの「配列要素を包含または除外する」
- 1006 ページの「配列要素を追加または削除する」
- 1007 ページの「<any> 要素の構造を作成する」

XML グリッドを入力する

XML グリッド内でクリックします。XML グリッドは XML エディタ，パラメータ化，XML 妥当性検証，サービス・エミュレーションに統合されています。複数のルート要素がある場合は，その 1 つを選択します。

データ型を確認する

データ型を確認するには，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。次に例を示します。

- **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。

base64 プロパティを設定する

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。UI の詳細については，962 ページの「［Process Base64 Data - Simple Data］ダイアログ・ボックス」を参照してください。

オプション要素を包含または除外する

オプション要素を包含するには，要素の横にある緑色の三角形を塗りつぶします。

配列要素を包含または除外する

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。

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。

スキーマ	設定 1	設定 2	設定 3
phone-numbers			
PhoneNumber [...]			
PhoneNumber[1]	[◆]	[◆]	[◆]
description	Home	Home	Home
phone-number	888-8888	111-1111	444-4444
PhoneNumber[2]	[◆]	[ ]	[◆]
description	Office	Office	Office
phone-number	666-6666	222-2222	999-9999
PhoneNumber[3]	[◆]	[◆]	[ ]
description	Mobile	Mobile	Mobile
phone-number	555-5555	333-3333	123-4567

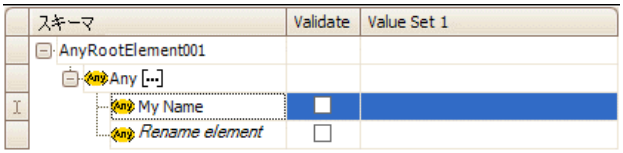
配列要素を追加または削除する

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。

配列を削除するには，親ノードを右クリックし，［**配列要素の削除**］を選択します。

<any> 要素の構造を作成する

- <any> 要素の配列を追加するには，親 <any> 要素を選択し，ショートカット・メニューから［**配列要素の追加**］を選択します。
- 子 < **任意** > 要素を追加するには，親要素を選択し，ショートカット・メニューから［**子の追加**］を選択します。
- 子 <any> 要素をさらに追加するには，子要素を選択し，ショートカット・メニューから［**兄弟の追加**］を選択します。
- <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。



- <any> 要素の値を読み込むには，ショートカット・メニューから［**XML の読み込み**］を選択します。

XML クエリを作成する方法

このタスクでは，クエリ・ビルダを使用して，特定の要素を見つけて調査する方法について説明します。詳細については，996 ページの「XML クエリ」を参照してください。

この機能を使用するには，Service Test ライセンスが必要です。

このタスクでは，次の手順を実行します。

- 1008 ページの「前提条件」
- 1008 ページの「[XML の検索］ダイアログ・ボックスを開く」
- 1008 ページの「クエリを指定する」
- 1008 ページの「クエリ・ビルダの詳細情報を指定する」
- 1008 ページの「クエリを実行する」

1 前提条件

スクリプトを作成して最低 1 回実行します。

2［XML の検索］ダイアログ・ボックスを開く

［スナップショット］タブで，［**XPath の検索**］をクリックします。［XML の検索］ダイアログ・ボックスで，［**要求**］または［**応答**］を選択します。
ユーザ・インタフェースの詳細については，1022 ページの「［XML の検索］ダイアログ・ボックス」を参照してください。

3 クエリを指定する

手動で XPath クエリを入力するか，［**クエリ ビルダ**］をクリックします。

4 クエリ・ビルダの詳細情報を指定する

クエリ・ビルダを使用する場合，該当するボックスに検索テキストを入力します。

- ［**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。
- ［**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。
- ［**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。
- ［**属性**］セクションを有効にして，属性を検索します。

  - 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。

5 クエリを実行する

［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 クエリ］ボックスに挿入されます。［**次を検索**］をクリックして検索を開始します。

リファレンス

［サービス管理］のユーザ・インタフェース

このセクションの内容

［サービスの管理］ダイアログ・ボックス

このダイアログ・ボックスでは，WSDL の管理，認証情報の入力，およびセキュリティ・シナリオの設定ができます。

利用方法	次のいずれかを使用します。 ➤  ➤ **［SOA ツール］> ［サービスの管理］**
関連タスク	997 ページの「［サービスの管理］ダイアログ・ボックスを開く」

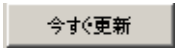
ユーザ・インタフェース要素の説明は次のとおりです（ラベルのない要素は山括弧で囲んで示します）。

UI 要素	説明
	［サービスのインポート］ダイアログ・ボックスが開きます。
	選択したサービスをリストから削除します。
	WSDL の作業用コピーと元のコピーが並んで表示される WSDL 比較レポートが開きます。 比較設定を設定する方法については，［XML / WSDL 比較］ダイアログ・ボックスを参照してください。
	WSDL が WS-I に準拠しているかどうかを確認する WS-I 検証ツールが開きます。 **注**：Service Test ライセンスがインストールされている場合にのみ表示されます。
	WSDL がブラウザに表示されます。
<WSDL リスト>	インポートされた WSDL のリスト。
［説明］タブ	WSDL，WSDL のエンドポイント・アドレス，ツールキット，および更新情報に関する情報を入力します。
［操作］タブ	サービスの操作が表示されます（［**操作名**］，［**ポート名**］，および［**スクリプトで使用中**］（［はい］または［いいえ］））。 カラムをクリックすると，そのカラムのデータごとに操作が並べ替えられます。再度クリックすると，並べ替え順が逆になります。
［接続の設定］タブ	サービスのインポート元となるマシンの認証設定を入力できます。詳細については，1012 ページの「［接続の設定］タブ」を参照してください。 **注**：これは URL および UDDI タイプのインポートにのみ適用されます。

UI 要素	説明
UDDI データ	UDDI サーバやバージョンやサービス・キー。詳細については，1013 ページの「[UDDI データ] タブ」を参照してください。
[プロトコルとセキュリティ] タブ	Web サービス呼び出しのセキュリティ・シナリオを表示および設定できます。詳細については，下記を参照してください。

[説明] タブ

[**説明**] タブには，次の要素が表示されます。

UI 要素	説明
	最新バージョンの WSDL が元の場所からロードされます。
作成者	ログインに使用される名前。このフィールドを編集して別の名前を指定できます。これは，レポートのサービス（読み取り専用）を並べ替えるのに役立ちます。
説明	サービスに関する情報を入力できる編集可能なフィールド。
オリジナルからの最新の更新	WSDL が更新された最終日時（読み取り専用）。
オリジナルの位置	WSDL のインポート元の場所（読み取り専用）。
優先アドレス	[**サービス アドレス**] ボックスのサービスの別のエンドポイントを入力できます。
サービス アドレス	サービス要求の送信先となるサービスのエンドポイントで，WSDL ファイルから取得されます（読み取り専用）。標準設定のアドレスをオーバーライドするには，[**優先アドレス**] を選択します。
サービス名	サービスのインポート時に標準設定で表示される WSDL ファイルのネイティブ・サービス名（読み取り専用）。

UI 要素	説明
ツールキット	サービスと関連するツールキット。サービスをインポートするときにこれを設定します（読み取り専用）。
スクリプトを開いたときに更新	スクリプトを開くたびに，ソースの WSDL が更新されます。

［接続の設定］タブ

含まれている要素は次のとおりです。

UI 要素	説明
認証	**［認証設定を使用する］**：認証の資格情報を入力できます。 ➤ **［ユーザ名］，［パスワード］**：WSDL の取得に使用するユーザ名とパスワード。 **ヒント**：標準設定のドメインに存在しないユーザの場合，ユーザ名の前にドメイン名を入力します。たとえば，domain1/alex_qc のように入力します。
プロキシ	**［プロキシ設定を使用する］**：プロキシの詳細情報や資格情報を入力できます。 ➤ **［サーバ］**：プロキシ・サーバの名前または IP アドレス。 ➤ **［ポート］**：WSDL のアクセスに使用するポート。 ➤ **［ユーザ名］，［パスワード］**：認証に使用するユーザ名とパスワード。標準設定のドメインに存在しないユーザの場合，ユーザ名の前にドメイン名を入力します。たとえば，domain1/alex_qc のように入力します。

［UDDI データ］タブ

含まれている要素は次のとおりです。

UI 要素	説明
サービス キー	UDDI サーバのサービスの一意の識別子。サービスの更新時にサービス定義を見つけるために使用されます。
UDDI サーバ	サービス定義のインポート元となる UDDI サーバの URL アドレスおよびバージョン。
UDDI のバージョン	UDDI レジストリのバージョン（**2** または **3**）。

［接続の設定］ダイアログ・ボックス

WSDL ファイルをホストするマシンの認証資格情報およびそのプロキシ・サーバの詳細情報を入力できます。

利用方法	新しいサービスの場合：［**SOA ツール**］>［**サービスのインポート**］を選択します。［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**］をクリックします。 既存のサービスの場合：［サービス管理］ダイアログ・ボックスでサービスを選択し，［**接続**］タブをクリックします。
重要情報	URL および UDDI を使用してインポートされたサービスにのみ適用されます。
関連タスク	997 ページの「サービスをインポートする」

含まれている要素は次のとおりです。

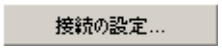
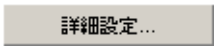
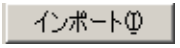
UI 要素	説明
認証	**[認証設定を使用する]**：認証の資格情報を入力できます。 ➤ **[ユーザ名]**，**[パスワード]**：WSDL の取得に使用するユーザ名とパスワード。 **ヒント**：標準設定のドメインに存在しないユーザの場合，ユーザ名の前にドメイン名を入力します。たとえば，domain1/alex_qc のように入力します。
プロキシ	**[プロキシ設定を使用する]**：プロキシの詳細情報や資格情報を入力できます。 ➤ **[サーバ]**：プロキシ・サーバの名前または IP アドレス。 ➤ **[ポート]**：WSDL のアクセスに使用するポート。 ➤ **[ユーザ名]**，**[パスワード]**：認証に使用するユーザ名とパスワード。標準設定のドメインに存在しないユーザの場合，ユーザ名の前にドメイン名を入力します。たとえば，domain1/alex_qc のように入力します。

［サービスのインポート］ダイアログ・ボックス

ファイル・システム，URL，Application Lifecycle Management，UDDI，または Systinet から WSDL をインポートできます。

利用方法	次のいずれかを使用します。 ➤［**サービス**］>［**新規作成**］>［**サービスのインポート**］を選択します。 ➤ ショートカット・メニューから［**新規作成**］>［**サービスのインポート**］を選択します。
関連タスク	997 ページの「サービスをインポートする」

含まれている要素は次のとおりです（ラベルのない GUI 要素は山括弧で囲んで示します）。

UI 要素	説明
	**[参照]**：ブラウザ，UDDI レジストリ，または Application Lifecycle Management リポジトリを使用して，ファイル・システムのサービスを見つけることができます。これは，[**Import WSDL from**] セクションの選択内容によって異なります。
	WSDL をホストするサーバの認証およびプロキシを設定する［接続の設定］ダイアログ・ボックスが開きます。詳細については，1013 ページの「[接続の設定］ダイアログ・ボックス」を参照してください。
	テストに使用するツールキットを選択できます。［**自動**］，［**.NET**］，または［**軸**］を選択します。［**自動**］設定では，アルゴリズムを使用して最適なツールキットが判断されます。
	［Application Lifecycle Management Connection］ダイアログ・ボックスが開き，ALM ホスト・マシンを指定できます。
	インポート・プロセスが開始されます。
WSDL を次から選択	WSDL の場所。情報を参照するか，手動で入力します。 ➤ **[ファイル]**：完全パスとファイル名。 ➤ **[URL]**：完全な URL。短縮版ではなく，完全な URL を挿入してください。 ➤ **[ALM]**：Application Lifecycle Management のフォルダ名。プロジェクトに接続している場合，［参照］ボタンをクリックすると［Import Service from Application Lifecycle Management］ダイアログ・ボックスが開きます。 ➤ **[UDDI]**：UDDI レジストリ ID。［参照］ボタンをクリックすると 1016 ページの「[UDDI 内でのサービス検索］ダイアログ・ボックス」が開きます。

［UDDI 内でのサービス検索］ダイアログ・ボックス

このダイアログ・ボックスでは，UDDI レジストリの特定のサービスを見つけることができます。

利用方法	➤［サービスの管理］ウィンドウで，［**インポート**］をクリックします。 ➤［インポート］ダイアログ・ボックスの［**WSDL を次から選択**］セクションで，［**UDDI**］を選択します。 ➤ ... をクリックします。
関連タスク	997 ページの「サービスをインポートする」

ユーザ・インタフェース要素の説明は次のとおりです（ラベルのない要素は山括弧で囲んで示します）。

UI 要素	説明
検索	［**サービス名の一部または全体**］ボックスのテキストに基づいてサービスの検索が開始されます。
< サービス・リスト >	文字列に一致するすべてのサービスのリスト。グリッドには，［**サービス名**］，［**サービス キー**］，［**サービスの説明**］，［**サービス WSDL**］のカラムが表示されます。
サービス名の一部または全体	目的のサービス名またはサービス名の一部が含まれている文字列。ワイルドカード表現を使用する必要はありません。 次のオプションで検索を絞り込みます。 ➤ **［完全一致］**：サービス名がテキストに完全一致している必要があります。 ➤ **［大文字と小文字を区別する］**：サービス名と指定したテキストの大文字と小文字が一致している必要があります。
UDDI サーバ照会アドレス	UDDI サーバの照会用完全パス。次に例を示します。
UDDI バージョン 2/3	リストに表示するサービスの UDDI バージョン。

［XML / WSDL 比較］ダイアログ・ボックス

このダイアログ・ボックスでは，異なるバージョンの WSDL を比較するための設定を構成できます。大文字と小文字やコメントなどの特定の差異を無視するよう比較ツールを設定できます。

利用方法	**［SOA ツール］>［SOA 設定］>［XML / WSDL 比較］**
関連タスク	998 ページの「WSDL の更新を確認する（任意）」

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
差異のみ表示	差異のみがレポートに表示され，一致テキストは表示されません。
大文字と小文字を区別しない	大文字と小文字の不一致が差異として表示されません。
コメントを無視	コメントの不一致が差異としてマークされません。
処理命令を無視	処理命令の不一致が差異としてマークされません。
名前空間を無視	名前空間の不一致が差異としてマークされません。

SOA ツールのユーザ・インタフェース

このセクションの内容

➤［XML エディタ］ダイアログ・ボックス（1021 ページ）
➤［XML の検索］ダイアログ・ボックス（1022 ページ）

［XML の妥当性検証］ダイアログ・ボックス

このダイアログ・ボックスでは XML パラメータとその期待値を検証できます。

利用方法	Validate XML（上部のツールバー）
重要情報	Service Test ライセンスがある場合にのみ利用できます。
関連タスク	1000 ページの「XML の妥当性を検証する方法」

ユーザ・インタフェース要素の説明は次のとおりです（ラベルのない要素は山括弧で囲んで示します）。

UI 要素	説明
Load	チェックポイント・グリッドの XSD スキーマ・ファイルが再ロードされます。
すべて選択	すべての要素の［**検証**］ボックスが選択されます。
すべて選択解除	すべての要素の［**検証**］ボックスがクリアされます。
すべて削除	➤ 上部の表示枠：すべてのチェックポイントの期待値が削除されます。 ➤ 下部の表示枠：すべての詳細なチェックポイントが削除されます。
記録	記録されたデータから XML 要素の期待値がロードされます。
再生	最新の再生時のデータから XML 要素の期待値がロードされます。
行を削除	選択した詳細なチェックポイントが削除されます。

UI 要素	説明
<チェックポイント・グリッド>	応答要素のリスト。グリッド・カラムには次が含まれています。 **XML チェックポイント** ➤ 確認する応答要素 **検証** ➤ 特定の要素の検証を無効 / 有効にします。 **期待値** ➤ 比較演算子のドロップダウン。該当する比較メソッド（[**次と一致する**］や［**次と一致しない**］など）が表示されます。 ➤ 応答を比較する値。 ➤ ハードコードされた値ではなくパラメータを期待値に使用できるようになります。
<詳細なチェックポイント・グリッド>	応答要素のリスト。グリッド・カラムには次が含まれています。 ➤ **[XPath クエリ]**：比較に使用する XPATH 式。 ➤ **[検証方法]**：検証方法は，［**次を含む**］，［**正規表現**］，または［**完全一致**］になります。 ➤ **[期待値]**：応答を比較する値。
XML ソース	検証する XML パラメータ。このドロップダウンには，使用可能なすべてのパラメータのリストと新しいパラメータを作成するためのオプションが表示されます。 ➤ **[出力パラメータのみ表示する]**：［XML ソース］ドロップダウン・リストに出力パラメータのみが表示されるよう制限します。
整形式の XML ファイルかをチェックする	XML が整形式であるかどうかがチェックされ，その結果がレポートに表示されます。
XML スキーマ	XML と関連するスキーマ・ファイル。これは自動的に取得される場合もあります。
チェックポイント	テスト実行中にチェックポイントの検証が有効になります。

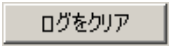
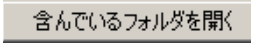
UI 要素	説明
詳細なチェックポイント	テスト実行中に詳細なチェックポイントの検証が有効になります。
検証エラー時にテストを失敗させる	検証エラーが発生した場合，テストのステータスが［**失敗**］としてマークされます。

［WSDL 参照アナライザ］ダイアログ・ボックス

このダイアログ・ボックスでは，WSDL ファイルの依存関係を定義できます。

利用方法	［**SOA ツール**］>［**WSDL 参照アナライザ**］
関連タスク	999 ページの「WSDL の依存関係を分析する方法」

ユーザ・インタフェース要素の説明は次のとおりです（ラベルのない要素は山括弧で囲んで示します)。

UI 要素	説明
	分析が開始され，すべての結果がログ・ウィンドウに表示されます。
	ログ・ウィンドウとログ・ファイルがクリアされます。
	出力ファイルが含まれているディレクトリが開きます。
< ログ・ウィンドウ >	参照分析の実行ログ。
WSDL ファイルの選択	分析する WSDL ファイルのローカル・パスまたは URL。
出力ファイル パス	出力用の zip ファイルの場所。

［XML エディタ］ダイアログ・ボックス

このダイアログ・ボックスでは，WSDL ファイルの依存関係を定義できます。

利用方法	**［SOA ツール］>［XML エディタ］**
重要情報	Service Test ライセンスがインストールされている場合にのみ利用できます。
関連タスク	1002 ページの「スキーマを編集する方法」

ユーザ・インタフェース要素の説明は次のとおりです（ラベルのない要素は山括弧で囲んで示します）。

UI 要素	説明
	**［スキーマの読み込み］**：エディタで XSD スキーマ・ファイルが開きます。
	**［列の追加］**：［**値**］カラムが最後のカラムの右側に追加されます。
	**［カラムの複製］**：選択したカラムと同じ値で新しいカラムが作成されます。
	**［カラムの削除］**：選択した［**値**］カラムが削除されます。
	**［XML の読み込み］**：選択したカラムにファイルの XML データがロードされます。
	**［XML の編集］**：選択したカラムのデータを使用したスキーマがテキスト・エディタで開きます。
	**［XML の保存］**：選択したカラムの値が XML ファイルに保存されます。
< スキーマ >	グリッド形式の要素値を使用した XSD スキーマ。
XML の値	選択したカラム（右側の表示枠）の値を使用したスキーマの XML 表現。

［XML の検索］ダイアログ・ボックス

このダイアログ・ボックスでは，WSDL ファイルの依存関係を定義できます。

利用方法	ツリー・ビューの［**スナップショット**］タブにある［**XPath の検索**］
重要情報	Service Test ライセンスがある場合にのみ利用できます。このダイアログ・ボックスは，Web サービスと Flex プロトコルの両方で使用されます。参考情報については，996 ページの「XML クエリ」を参照してください。
関連タスク	➤ 1007 ページの「XML クエリを作成する方法」 ➤ 677 ページの「XML ツリーをクエリする方法」

ユーザ・インタフェース要素の説明は次のとおりです（ラベルのない要素は山括弧で囲んで示します)。

UI 要素	説明
クエリ ビルダ...	クエリ情報を指定するための［XML ノードのクエリ］ダイアログ・ボックスが開きます。
XPath クエリ	クエリ式。これは，手動で入力することも，クエリ・ビルダを使用して入力することもできます。
検索	検索を実行する SOAP メッセージ（［**要求**］または［**応答**］)。

［XML ノードのクエリ］ダイアログ・ボックス

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
	**［属性の追加］**：［属性のプロパティ］ダイアログ・ボックスが開きます。
	**［削除］**：選択した属性がリストから削除されます。
	**［編集］**：選択した属性を編集できます。
名前	ノードまたは要素の名前を検索します。

UI 要素	説明
名前空間 URI	名前空間を検索します。
テキスト	［**名前**］ボックスで指定した要素の値を検索します。
属性	属性リストの属性を検索します。

37

Web サービス - スクリプトの再生の準備

本章の内容

概念

タスク

リファレンス

- ［ツリー ビュー］タブ（1075 ページ）
- データベース統合のユーザ・インタフェース（1078 ページ）

概念

再生の準備の概要

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。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，*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。

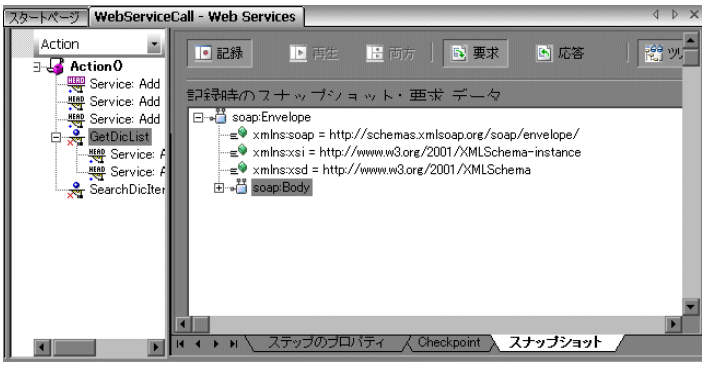
[実行環境の設定] では，実際のユーザをより正確にエミュレートして，実行環境を設定できます。この設定には，一般設定と Web サービス固有の設定が含まれます。詳細については，419 ページの「実行環境の設定」を参照してください。

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，1028 ページの「チェックポイント」を参照してください。チェックポイントの検証は，Service Test ライセンスでのみ使用できます。詳細については，HP サポートに問い合わせてください。

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。これを行うには，結果をパラメータに保存し，後で参照します。詳細については，1057 ページの「入力パラメータ値を割り当てる」を参照してください。

Web サービス・スクリプトのタブ

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，950 ページの「[新規 Web サービス呼び出し] ダイアログ・ボックス」を参照してください。
- **[チェックポイント]**：ステップに定義されているチェックポイントのリスト。
- **[SOAP 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，1075 ページの「[スナップショット] タブ」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，テストによってアクション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。[**チェックポイント**] 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。

チェックポイントを設定するには，テストを再生する前に引数の期待値を設定する必要があります。値は手動で入力できます。また，記録または再生中にキャプチャした一連の期待値をロードできます。多くの引数値がある場合は，この方法が役に立ちます。この方法では，値を手動で入力する代わりに，値が自動的にロードされます。

テストの実行後に，再生ログまたはテスト結果を表示して，結果が期待どおりであったかどうかを確認できます。

スクリプトでは，［**チェックポイント**］タブで有効にした行ごとにチェックポイント引数が生成されます。

```
web_service_call("StepName=Add_2",
        "SOAPMethod=Calc.CalcSoapPort.Add",
…
        BEGIN_CHECKPOINTS,
            StopOnValidationError=1,
            CHECKPOINT, "XPATH=Result[1]", "Value=13",
            CHECKPOINT, "XPATH=AddResult", "Expression=Hel*?",
        END_CHECKPOINTS,
        LAST);
```

チェックポイントの検証では，**lr_xml_find** を使用する標準の XPATH 検証もサポートされます。

SOAP 本文（.NET ツールキットでは SOAP ヘッダ）を検証するには，チェックポイントを使用できます。ただし，.NET 以外のツールキットを使用する場合，SOAP ヘッダでチェックポイントはサポートされません。代わりに，XML 妥当性検証を使用します。詳細については，986 ページの「XML 妥当性検証の概要」を参照してください。

タスクの詳細については，1060 ページの「チェックポイントを設定する方法」を参照してください。

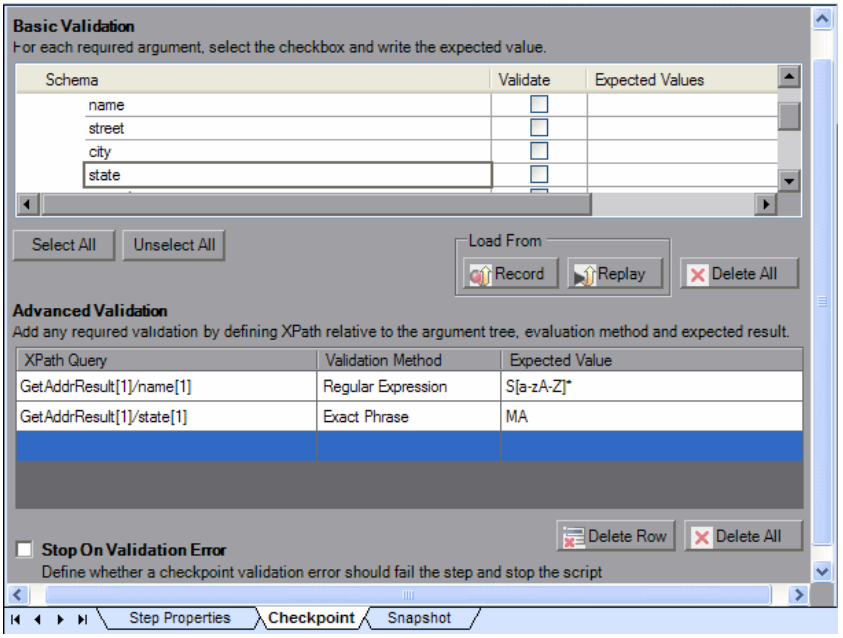
基本検証と詳細検証

基本検証または**詳細**検証を実行できます。

基本検証では，［**期待値**］カラムの値の完全一致が，VuGen with Service Test によって検索されます。記録時のスナップショットまたは再生時のスナップショットから期待値をロードできます。

非リーフ・ノード上のチェックポイントを検証したり，正規表現の点から期待値を定義したりするには，**詳細検証**を使用します。

同じステップに基本検証と詳細検証の両方を定義できます。



非リーフ・ノードを選択する場合，そのノードの下の XML をすべて入力する必要があります。

仮想ユーザ・スクリプトの場合，VuGen では **Value=** と **Expression=** の正規表現で完全一致を表します。

```
BEGIN_CHECKPOINTS,
        CHECKPOINT, "XPATH=/AddResult[1]", "Value=50"
        CHECKPOINT, "XPATH=AddResult", "Expression=Hel*?",
END_CHECKPOINTS,
```

XPath 式

チェックポイントの式では，ノードセットと拡張 XPath の構文がサポートされています。

ノードセットの式は，1 つの XML ノードまたは一連の XML ノードとして評価される XPath 式です。次に例を示します。

```
/a/b/c
/x/y[1]
//a/b
```

VuGen では，完全な XPath 式（整数として評価される **count(/a/b/c)** やブール値として評価される **count(/a/b/c) < 3** など）もサポートされています。この XPath のサポートにより，数値を比較できます。たとえば，結果が 8 ～ 16 であるかどうかを検証するには，次の式を使用します。

```
number(/a/b/c) > 8 and number(/a/b/c) < 16
```

Web サービス・トランスポート層のテストの概要

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に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，1048 ページの「ユーザ・ハンドラ」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。

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，1035 ページの「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。

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，1036 ページの「WS-Addressing」を参照してください。

JMS トランスポートの概要

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ます。これにより，ポイントツーポイント通信がサポートされます。

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]**：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。

これらは，JMS 実行環境の設定で設定できます。詳細については，472 ページの「[JMS] > [詳細] ノード」を参照してください。

このセクションには次の内容も含まれます。

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_publish_message_topic	特定のトピックに対してメッセージを発行します。
jms_receive_message_queue	キューからメッセージを受信します。
jms_receive_message_topic	サブスクリプションの特定のトピックに対して発行された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。

関数名	説明
jms_subscribe_topic	トピックのサブスクリプションを作成します。
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するか，JMS ヘッダまたはプロパティを使用して受信メッセージをフィルタ処理します。

JMS ステップ / 関数は，手動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。

メッセージ・キューを使用するピアツーピア通信とは異なり，パブリッシュ-サブスクライブ関数（**jms_publish_message_topic**，**jms_subscribe_topic**，および **jms_receive_message_topic**）は Web サービス呼び出しでサポートされません。これらの関数を Web サービス呼び出しで使用するには，ユーザ・ハンドラを手動で設定して JMS メッセージのペイロードを生成する必要があります。詳細については，1069 ページの「ユーザ・ハンドラを作成する方法」を参照してください。

JMS の関数の詳細については，［*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 メッセージ構造

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。次に例を示します。

```
jms_set_general_property("step1","JMS_MESSAGE_TYPE","BytesMessage");
```

詳細については，*オンライン関数リファレンス*を参照してください。

非同期メッセージの概要

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。

このセクションには次の内容も含まれます。

- 1035 ページの「HTTP/HTTPS を使用した非同期呼び出しの送信」
- 1036 ページの「WS-Addressing」

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仮想ユーザに指示します。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は，次の項目を指定します。

- **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）にするかを仮想ユーザに指示します。**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。
- **Timeout**：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。
- **イベント**：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。

タスクの詳細については，1064 ページの「非同期 HTTP メッセージを送信する（任意）」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，1036 ページの「WS-Addressing」を参照してください。

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>
      <!-- SOAP 要求メッセージの本文 -->
   </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。

タスクの詳細については，1065 ページの「WS-Addressing を使用して非同期メッセージを送信する（任意）」を参照してください。

データベース統合の概要

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。

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 6 章，「テスト結果の表示」。を参照してください。

データベース統合関数は次のシナリオで役立ちます。

- 1032 ページの「HTTP/HTTPS でのメッセージ送信」
- 1039 ページの「SQL クエリから取得したデータの使用」
- 1042 ページの「Web サービス呼び出し後のデータベース値の検証」
- 1044 ページの「データベース経由の戻り値のチェック」
- 1046 ページの「データセットに対するアクション実行」

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。組み込みの接続文字列ジェネレータによって，データベースと資格情報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。仮想ユーザは **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。

タスクの詳細については，1061 ページの「JMS でメッセージを送信する方法」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用して，データをスクロールする

詳細については，オンライン・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，オンライン・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，期待値と対照して，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。XML 妥当性検証ツールの詳細については，986 ページの「XML 妥当性検証の概要」を参照してください。

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，オンライン・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。

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で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。
- **[印刷]**：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，その内容を出力できません。

この関数の構文と使い方については，オンライン・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。

ネガティブ・テストの概要

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**ポジティブ・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ことを検査します。

さらに，**ネガティブ・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，入力データを受け付けるフォームについて考えます。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付けていることを検査するには，ポジティブ・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，ネガティブ・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。ネガティブ・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 では，標準の SOAP 結果または一般的な SOAP エラー応答があるかどうかを確認できます。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 エラーが発生します。ネガティブ・テストを使用すると，SOAP が有効であっても再生は失敗します。

テスト方法

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
ポジティブ テスト	SOAP 結果応答を受け入れ，SOAP エラーで失敗します。
ネガティブ テスト	SOAP エラーを受け入れ，SOAP 結果応答で失敗します。
すべてのタイプ	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 はポジティブ・テストのみを実行し，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。ネガティブ・テストのみを実行する，または，すべての SOAP 応答を受け入れるよう VuGen に指示することができます。ネガティブ・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。

すべての SOAP 応答（SOAP 結果および SOAP エラー）を受け入れるよう VuGen に指示できます。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。

Application Lifecycle Management 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。

カスタマイズの概要

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。

このセクションの内容

➤ 1048 ページの「ユーザ・ハンドラ」

➤ 1052 ページの「カスタム設定ファイル」

ユーザ・ハンドラ

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できるか，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。

- ハンドラ関数の定義
- イベント・ハンドラのリターン・コード

タスクの詳細については，1069 ページの「ユーザ・ハンドラを作成する方法」を参照してください。

サンプルのユーザ・ハンドラについては，1052 ページの「ユーザ・ハンドラの例」を参照してください。

ハンドラ関数の定義

ユーザ・ハンドラの基本実装では，ユーザ・ハンドラ関数は仮想ユーザ・スクリプト内に次の構文で定義されます。

```
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。

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，関数名を **UserHandlerFunction** 引数の値として使用します。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。

イベント・ハンドラのリターン・コード

VuGen は，ハンドラ関数の次のリターン・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;
```

カスタム設定ファイル

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。このファイルでは，実行時のテストの動作を制御できます。

標準の .NET 設定ファイル（**mmdrv.exe.config**）は VuGen のインストール・フォルダにあります。一部のアプリケーションには独自の設定ファイル（**app.config**）があります。

入力または出力を除外することで，テストの実行をより詳細にカスタマイズできます。また，トークン情報や，テスト用の無署名の証明書を許可するかどうかなど，セキュリティ情報を設定できます。

タスクの詳細については，1073 ページの「設定ファイルをカスタマイズする方法」を参照してください。

ユーザ・ハンドラの例

このセクションでは，ユーザ・ハンドラの一般的な使用方法についていくつか説明します。

.NET フィルタ

.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。

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。

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，次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。次に例を示します。

```
    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

次の例は，トランスポート層をオーバーライドするユーザ・ハンドラ関数を示しています。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，次のコマンドを使用して応答エンベロープを設定します。

```
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** を指定します。トランスポート層をハンドラ内に定義します。

```
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。

以降の項では，.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，実際のパスとコンテント・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** 内で MIME 添付ファイルを送受信するには，Web サービス呼び出しを次のように変更します。

```
"UserHandlerName=LrWsAttachmentsHandler",
 "UserHandlerArgs=ATTACHMENT_SAVE_ALL;ParamNamePrefix=attach;
ATTACHMENT_ADD; ATTACHMENTS_FORMAT_MIME; ContentType=text/plain;
FileName=C:¥¥temp¥¥results.discomap",
```

タスク

スクリプトの再生の準備をする方法

このタスクでは，スクリプトの再生準備を行ってスクリプトを実行する方法について説明します。

このタスクでは，次の手順を実行します。

- 1057 ページの「入力パラメータ値を割り当てる」
- 1058 ページの「実行環境を設定する（任意）」
- 1058 ページの「Any タイプ要素を含む XSD を設定する（任意）」
- 1059 ページの「スクリプトを実行する」
- 1059 ページの「テスト結果を確認する」

入力パラメータ値を割り当てる

まず，出力結果をパラメータに保存します。その後，そのパラメータを Web サービス呼び出しで参照します。

1 出力パラメータを保存します。

a ツリー・ビューで，出力を使用する Web サービス呼び出しをダブルクリックして，そのプロパティを表示します。

b 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。

c 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。［**パラメータ**］ボックスで名前を指定します。

2 保存したパラメータを入力に使用します。

a ツリー・ビューで，入力パラメータを設定する Web サービス呼び出しをダブルクリックします。

b 左側の表示枠で，保存したパラメータを使用する入力パラメータを選択します。

c 右側の表示枠で，［**値**］を選択して［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。

d ［**パラメータ名**］リストで，保存した出力パラメータを選択します。

e 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，ショートカット・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。

実行環境を設定する（任意）

実行環境の設定を開いて（F4 キー），JMS と VM を設定します。［**JMS**］>［**詳細**］ノードをクリックします。ユーザ・インタフェースの詳細については，472 ページの「[JMS］>［詳細］ノード」を参照してください。

Any タイプ要素を含む XSD を設定する（任意）

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ます。

```
BEGIN_ARGUMENTS,
                "xml:Any_element="
                        "<Any_element>"
                                "<string> 送信する文字列 </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>** タイプに対して複合型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>",
```

チェックポイントを設定する

チェックポイントまたは XML 妥当性検証を設定します。詳細については，1060 ページの「チェックポイントを設定する方法」または 1000 ページの「XML の妥当性を検証する方法」を参照してください。

ヒント：XML 妥当性検証をお勧めします。XML 妥当性検証では，応答をより包括的に検証できるためです。XML 妥当性検証では，独立した検証ステップをスクリプトに挿入でき，また，完全な XML ツリーを期待値としてロードできます。

スクリプトを実行する

［**仮想ユーザ**］>［**実行**］をクリックします。出力ログで，関連するメッセージを確認します。

テスト結果を確認する

テストの実行後に［テスト結果］ビューアが自動的に開きます。X は失敗したステップを示します。

ノードを展開すると, SOAP 応答とチェックポイントに関する詳細が表示されます。詳細については，173 ページの「テスト結果の表示」を参照してください。

チェックポイントを設定する方法

このタスクでは，XML が整形式であるかどうかの検証方法や，期待される応答の確認方法について説明します。チェックポイントの概要については，1028 ページの「チェックポイント」を参照してください。

このタスクでは，次の手順を実行します。

- 1060 ページの「[チェックポイント] タブを開く」
- 1060 ページの「期待値を設定する」
- 1060 ページの「チェックポイントを有効にする」
- 1060 ページの「詳細なチェックポイントを設定する（任意）」
- 1061 ページの「スクリプトを実行および結果を表示する」

1 [チェックポイント] タブを開く

ツリー・ビューで，ステップを選択し，[**チェックポイント**] タブをクリックします。ユーザ・インタフェースの詳細については，1077 ページの「[チェックポイント] タブ」を参照してください。

2 期待値を設定する

[**期待値**] カラムで値を指定するか，[**記録**] または [**再生**] ボタンをクリックして SOAP メッセージから値をロードします。

3 チェックポイントを有効にする

[**検証**] カラムで，検証する結果を確認します。[**すべて選択**] をクリックすると，すべての結果の検証が有効になります。

4 詳細なチェックポイントを設定する（任意）

a [**詳細なチェックポイント**] オプションを選択します。

b [**詳細検証**] セクションで，XPath 式を既存の引数からコピーします。たとえば，[チェックポイント] リストで引数値を選択し，ショートカット・メニューから [XPath のコピー] を選択します。

c XPath 式を［**XPath クエリ**］カラムに貼り付けます。詳細については，989 ページの「XPATH 式」を参照してください。

d ［**正規表現**］または［**完全一致**］のいずれかの検証方法を選択します。

e 期待値を指定します。

5 スクリプトを実行および結果を表示する

a F5 をクリックしてスクリプトを実行します。再生ログを表示してエラーがないか確認します。

b テスト結果を表示します。［テスト結果］ウィンドウが自動的に表示されない場合は，［**表示**］>［**テスト結果**］を選択します。

c ［チェックポイント］ノードを展開します。実際の値と期待値を表示します。

JMS でメッセージを送信する方法

このタスクでは，JMS トランスポート・メソッドを使用してメッセージを送信する方法について説明します。

このタスクでは，次の手順を実行します。

- 1069 ページの「前提条件 - Web サービス呼び出しを作成する」
- 1057 ページの「入力パラメータ値を割り当てる」
- 1058 ページの「実行環境を設定する（任意）」
- 1062 ページの「同期 JMS メッセージを送信する（任意）」
- 1062 ページの「非同期 JMS メッセージを送信する（任意）」
- 1062 ページの「SOAP メッセージを使用してメッセージを JMS で送信する（任意）」

1 ステップのプロパティを開く

ツリー・ビューで，トランスポートを設定するステップを選択します。ショートカット・メニューから［**プロパティ**］を選択します。

2 JMS トランスポート・メソッドを選択する

［**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。

UI の詳細については，952 ページの「［トランスポート層の設定］ノード」を参照してください。

3 実行環境を設定する（任意）

実行環境を設定します。詳細については，472 ページの「[JMS] > [詳細] ノード」を参照してください。

4 同期 JMS メッセージを送信する（任意）

Web サービス呼び出しを作成し，トランスポート・メソッドを JMS と指定すると，VuGen によって JMS メッセージが同期方式で送信されます。必要に応じて，キュー情報を指定します。

5 非同期 JMS メッセージを送信する（任意）

JMS で非同期メッセージを実行するには，Web サービス呼び出しではなく JMS ステップを使用して，要求の送信または応答の受信を行います。

a 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。

b JMS 関数を選択します。[**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。

c [**OK**] をクリックして，JMS 関数のプロパティを開きます。

d 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。

これらの関数の詳細については，[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。

6 SOAP メッセージを使用してメッセージを JMS で送信する（任意）

Web サービス呼び出しを使用せずに SOAP メッセージを使用してメッセージを JMS で送信するには，次の手順で行います。

a 標準の Web プロトコルを使用して SOAP メッセージを記録します。

b 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。

c JMS 関数を選択します。［**メッセージ送信キュー**］または［**JMS メッセージ受信キュー**］のいずれかを選択します。

d ［**OK**］をクリックして，JMS 関数のプロパティを開きます。

e 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。

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。

HTTP/S でメッセージを送信する方法

このタスクでは，HTTP トランスポート・メソッドを使用してメッセージを送信する方法について説明します。

このタスクでは，次の手順を実行します。

- 1063 ページの「ステップのプロパティを開く」
- 1063 ページの「HTTP/S トランスポート・メソッドを選択する」
- 1063 ページの「HTTP 同期メッセージを送信する（任意）」
- 1064 ページの「非同期 HTTP メッセージを送信する（任意）」
- 1065 ページの「WS-Addressing を使用して非同期メッセージを送信する（任意）」

1 ステップのプロパティを開く

ツリー・ビューで，トランスポートを設定するステップを選択します。ショートカット・メニューから［**プロパティ**］を選択します。

2 HTTP/S トランスポート・メソッドを選択する

［**トランスポート層の設定**］ノードを選択し，［**HTTP/S トランスポート**］を選択します。

3 HTTP 同期メッセージを送信する（任意）

メッセージを HTTP を介して同期モードで送信するには，［**非同期サポート**］オプションを有効にしないで，標準の Web サービス呼び出しを作成します。

4 非同期 HTTP メッセージを送信する（任意）

a ［**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。

b ［**非同期イベント**］ボックスにイベント名を入力します。

c ［**OK**］クリックすると，Web サービス呼び出しが生成されます。

d **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。

e ステップ名，修飾子，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，追加パラメータの **AsyncEvent** で非同期メッセージングが示されます。

```
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仮想ユーザに指示します。

5 WS-Addressing を使用して非同期メッセージを送信する（任意）

a ［**非同期サポート**］オプションを選択し，［**非同期イベント**］ボックスでイベント名を指定します。任意の名前を指定できます。

b ［**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。

c ［**OK**］をクリックして設定を保存します。

d 仮想ユーザ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。

e ステップ名，修飾子，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。

f ［**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。

テスト方法を定義する方法

このタスクでは，テスト方法をどのように選択するかについて説明します。

このタスクでは，次の手順を実行します。

➤ 1069 ページの「前提条件 - Web サービス呼び出しを作成する」

➤ 1057 ページの「入力パラメータ値を割り当てる」

➤ 1058 ページの「実行環境を設定する（任意）」

➤ 1066 ページの「スクリプトで関数を検証する」

➤ 1059 ページの「スクリプトを実行する」

1 ステップのプロパティを開く

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 引数を選択する

［出力引数］ノードを選択します。詳細については，956 ページの「［出力引数］ノード」を参照してください。

3 テスト方法を選択し，期待される応答を選択する

- ネガティブ・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。
- ポジティブ・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。

4 スクリプトで関数を検証する

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。次のスクリプト例では，**SoapFault** 値で指定されたネガティブ・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);
```

5 SOAP エラー値を評価する

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。SOAP エラーの戻り値を確認するには，**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，*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。

データベース接続を追加する方法

このタスクでは，ツリー・ビューでデータベース接続ステップを追加する方法について説明します。

このタスクでは，次の手順を実行します。

- 1069 ページの「前提条件 - Web サービス呼び出しを作成する」
- 1057 ページの「入力パラメータ値を割り当てる」
- 1058 ページの「実行環境を設定する（任意）」
- 1066 ページの「スクリプトで関数を検証する」
- 1059 ページの「スクリプトを実行する」

1 ツリー・ビューを開く

[**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 セクションを選択する

目的とするオプション，**vuser_init** または **Action** を選択します。接続シーケンスを各反復で繰り返さないようにするには，接続シーケンスを **vuser_init** セクションに配置します。

3 データベース接続ステップを挿入する

［**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。［**ステップ名**］，［**接続名**］，および［**データ プロバイダ**］，OLEDB または SQL を指定します。

4 データベース接続文字列を作成する

a ［**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。

b 接続プロパティを指示します。

- **サーバ名**
- **データベース名**
- **認証**方式：Windows 認証またはユーザ / パスワード。
- ［**ユーザ名**］および［**パスワード**］

c ［**接続のテスト**］をクリックして，提示した情報が正しいことを確認します。

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

5 スクリプトで関数を検証する

lr_db_connect 関数がスクリプトに書き込まれていることを確認します

ユーザ・ハンドラを作成する方法

このタスクでは，スクリプトのユーザ・ハンドラを記述する方法について説明します。詳細については，1048 ページの「カスタマイズの概要」を参照してください。

このタスクでは，次の手順を実行します。

- 1069 ページの「前提条件 - Web サービス呼び出しを作成する」
- 1057 ページの「入力パラメータ値を割り当てる」
- 1070 ページの「ユーザ・ハンドラ関数を呼び出す」
- 1070 ページの「ハンドラ関数を評価する」
- 1058 ページの「実行環境を設定する（任意）」
- 1059 ページの「スクリプトを実行する」
- 1072 ページの「ユーザ・ハンドラを必要なすべてのマシンにコピーする」
- 1073 ページの「ユーザ・ハンドラを実装する（任意）」

1 前提条件 - Web サービス呼び出しを作成する

WSDL ファイルをインポートし，標準の Web サービス呼び出しを作成します。詳細については，917 ページの「新しい Web サービス呼び出しの追加」を参照してください。

2 ユーザ・ハンドラ関数を定義する

Web サービス呼び出しの前にユーザ・ハンドラを定義します。

```
int MyScriptFunction(const char* pArgs, int isRequest)
{
…
}
```

3 ユーザ・ハンドラ関数を呼び出す

Web サービス呼び出し内で，関数名を **UserHandlerFunction** 引数の値として指定して，ハンドラ関数を呼び出します。

```
web_service_call(
...
"UserHandlerFunction=MyScriptFunction",
"UserHandlerArgs=<handler arguments>",
  LAST);
```

4 ハンドラ関数を評価する

ハンドラのリターン・コードを評価して，ハンドラが成功したかどうかを確認します。1050 ページの「イベント・ハンドラのリターン・コード」の説明にしたがってリターン・コードを使用します。

```
// 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;
}
```

5 DLL ファイルを作成する（任意）

DLL を使用してユーザ・ハンドラを定義するには，API ヘッダ・ファイル（**LrWsHandlerAPI.h**）を探します。このファイルは，製品の **include** ディレクトリにあります。

ハンドラを作成するには，samples/WebServices/SampleWsHandler ディレクトリにあるサンプルの Visual Studio プロジェクトをテンプレートとして使用できます。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 **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** ディレクトリに置いてください。

6 ユーザ・ハンドラを設定する（任意）

DLL ユーザ・ハンドラをグローバルまたはローカルで宣言します。

ユーザ・ハンドラをスクリプト内のすべての要求にグローバルに適用するには，スクリプトのディレクトリにある **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>
```

- **［名前］**：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**：仮想ユーザが要求のユーザ・ハンドラを処理する順序。**Before Security**（セキュリティの前），**After Security**（セキュリティの後），または **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，仮想ユーザは，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，仮想ユーザ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>
```

7 ユーザ・ハンドラを必要なすべてのマシンにコピーする

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 を 製品の **/bin** フォルダにコピーできます。

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。

8 ユーザ・ハンドラを実装する（任意）

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。次に例を示します。
  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**：再生ログに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**：カスタム・プロパティ値を取得します。
- **SetProperty**：カスタム・プロパティ値を設定します。

詳細については，製品の **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。

設定ファイルをカスタマイズする方法

次の手順では，設定ファイルの変更方法について説明します。詳細については，1052 ページの「カスタム設定ファイル」を参照してください。

- 1074 ページの「設定ファイルを検索する」
- 1074 ページの「アプリケーションの設定ファイルを保存する」
- 1074 ページの「セキュリティを設定する（任意）」

設定ファイルを検索する

設定ファイルの場所を確認します。標準の .NET 設定ファイル（**mmdrv.exe.config**）は製品の **bin** フォルダにあります。一部のアプリケーションには独自のファイル（**app.config**）があります。

アプリケーションの設定ファイルを保存する

アプリケーションに独自の app.config ファイルがある場合。

- 設定情報をすべてのスクリプトにグローバルに適用するには，**app.config** ファイルを **bin** フォルダに **mmdrv.exe.config** として保存して，既存のファイルを上書きします。
- 設定情報を，特にこのスクリプトに対してローカルに適用するには，**app.config** ファイルをスクリプトのフォルダにコピーします。これにより，**mmdrv.exe.config** ファイルがオーバーライドされます。このファイルをほかのマシンにコピーしても，このスクリプトとの関連付けは維持されます。

セキュリティを設定する（任意）

標準設定では，VuGen では，テストを円滑にするために無署名の証明書を使用できます。無署名の証明書を不許可にするには，<security> セクションの **allowTestRoot** フラグを **false** に変更します。

```
<security>
  <x509 storeLocation="currentuser" alllowTestRoot="false"
```

リファレンス

［ツリー ビュー］タブ

このセクションでは，ツリー・ビューに表示される［**スナップショット**］タブと［**チェックポイント**］タブについて説明します。［**プロパティ**］タブの詳細については，950 ページの「[新規 Web サービス呼び出し］ダイアログ・ボックス」を参照してください。

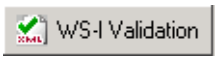
［スナップショット］タブ

このタブでは，Web サービス呼び出しのスナップショットを表示できます。記録または再生のスナップショット，および要求と応答を表示できます。

利用方法	ツリー・ビューの［**スナップショット**］タブ
関連タスク	1057 ページの「スクリプトの再生の準備をする方法」

ユーザ・インタフェース要素の説明は次のとおりです（ラベルのない要素は山括弧で囲んで示します)。

UI 要素	説明
	記録スナップショットを表示します。
	最新の再生スナップショットを表示します。 **ヒント**：特定の反復を選択するには，［**表示**］>［**スナップショット**］>［**反復の選択**］を選択します。
	最新の記録と再生のスナップショットを表示します。
	Web サービス呼び出しによってサーバに送信された SOAP 要求を表示します。
	サーバから返された SOAP 応答を表示します。

UI 要素	説明
	SOAP メッセージをツリー階層で表示します。 **ヒント**：ツリー・ノードを展開すると，引数値が表示されます。
	SOAP メッセージを XML テキスト形式で表示します。
	［XML の検索］ダイアログボックスとクエリ・ビルダを開きます。詳細については，1007 ページの「XML クエリを作成する方法」を参照してください。 **注**：Service Test ライセンスがある場合にのみ利用できます。
	現在のスナップショットの WS-I 準拠を確認します。 **注**：Service Test ライセンスがある場合にのみ利用できます。
< 表示領域 >	選択した項目に基づいて SOAP メッセージを表示します。 ➤ 記録，応答，または両方のスナップショット。 ➤ 要求または応答メッセージ ➤ ツリーまたは XML 表現
ショートカット・メニュー	［**応答**］ビューで，出力引数をクリックします。 ➤ **［ノードのプロパティ］**：［XML ノードのプロパティ］ダイアログ・ボックスを開きます。 ➤ **［XML チェックを挿入］**：**XML 検索**ステップを挿入します。これにより，検索文字列，大文字と小文字の区別，およびエラー発生時の動作を指定できます。 ➤ **［パラメータに値を保存］**：選択した値を単純パラメータに保存します。 ➤ **［パラメータに XML を保存］**：選択した値を XML パラメータに保存します。 ➤ **［XML のコピー］**：XML ノードをクリップボードにコピーします。

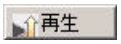
［チェックポイント］タブ

このダイアログ・ボックスでは XML パラメータとその期待値を検証できます。

利用方法	ツリー・ビューの［**チェックポイント**］タブ
重要情報	➤ Service Test ライセンスがある場合にのみ利用できます。 ➤ 応答に対してより包括的な検証を行うには，XML 妥当性検証ツールを使用します。このツールでは，独立した検証ステップをスクリプトに挿入でき，また，完全な XML ツリーを期待値としてロードできます。詳細については，1060 ページの「チェックポイントを設定する方法」を参照してください。
関連タスク	1060 ページの「チェックポイントを設定する方法」

ユーザ・インタフェース要素の説明は次のとおりです（ラベルのない要素は山括弧で囲んで示します）。

UI 要素	説明
すべて選択	すべての要素の［**検証**］ボックスが選択されます。
すべて選択解除	すべての要素の［**検証**］ボックスがクリアされます。
すべて削除	➤［**基本検証**］表示枠では，すべての期待値を削除します。 ➤［**詳細検証**］表示枠では，すべての詳細検証エントリを削除します。
行を削除	選択した**詳細検証**エントリを削除します。
基本検証	応答要素のリスト。カラムには次の項目が含まれます。 ➤ **［スキーマ］**：確認する応答要素 ➤ **［検証］**：特定の要素の検証を無効 / 有効にします。 ➤ **［期待値］**：応答を比較する値。

UI 要素	説明
［読み込み元］:	期待値のソース。 ➤ **［記録］**  : 記録されたメッセージから値を取得します。 ➤ **［応答］**  : 最新の再生から値を取得します。
詳細検証	応答要素のリスト。グリッド・カラムには次が含まれています。 ➤ **［XPath クエリ］:** 比較に使用する XPATH 式。 ➤ **［検証方法］:** 検証方法は，［**次を含む**］，［**正規表現**］，または［**完全一致**］になります。 ➤ **［期待値］:** 応答を比較する値。
検証エラーで停止	検証エラーが発生した場合，テストの実行を停止し，テストのステータスを**失敗**としてマークします。

データベース統合のユーザ・インタフェース

このセクションの内容

［Database Connection］ダイアログ・ボックス

このダイアログ・ボックスは，データベースの接続文字列の作成に役立ちます。

利用方法	［Database Connection］ダイアログ・ボックスで **［Connection String Generator］**をクリックします。
関連タスク	1061 ページの「JMS でメッセージを送信する方法」
関連項目	1079 ページの「［Connection String Generator］ダイアログ・ボックス」

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
Connection String Generator	Connection String Generator を開きます。詳細については，1079 ページの「［Connection String Generator］ダイアログ・ボックス」を参照してください。
ステップ名	データベース・サーバの名前または IP アドレス。
接続文字列	データベースへの接続で使用する文字列。**Connection String Generator** を使用します。
データ プロバイダ	SQL プロバイダ。**OLEDB** または **SQL** です。

［Connection String Generator］ダイアログ・ボックス

このダイアログ・ボックスは，データベースの接続文字列の作成に役立ちます。

利用方法	［Database Connection］ダイアログ・ボックスで **［Connection String Generator］**をクリックします。
関連タスク	1061 ページの「JMS でメッセージを送信する方法」
関連項目	1079 ページの「［Database Connection］ダイアログ・ボックス」

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
Test Connection...	データベースへの接続をテストします。
生成	データベース接続文字列を生成し，その文字列を，［Database Connection］ダイアログ・ボックスの［**接続文字列**］フィールドに書き込みます。
サーバ名	データベース・サーバの名前または IP アドレス。
DB 名	データベースの名前。
認証	データベースの認証方式。**Windows 認証**または**ユーザ / パスワード**。 ➤［**ユーザ名**］，［**パスワード**］：データベースの資格情報。
SQL プロバイダ	SQL プロバイダ。**OLEDB** または **SQL** です。

38

Web サービス－セキュリティ

本章の内容

概念

セキュリティの設定の概要

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・メカニズムがサポートされます。

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，1087 ページの「SAML セキュリティ・オプション」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がサポートされます。この章では，**Web Service Set Security** ステップを使用したレガシ・セキュリティ・モデルについて説明します。セキュリティ・シナリオの詳細については，1088 ページの「セキュリティ・シナリオの概要」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。

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，1013 ページの「［接続の設定］ダイアログ・ボックス」を参照してください。

このセクションには次の内容も含まれます。

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 アカウント情報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 アカウント情報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 アカウント情報が 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，そのアカウント情報が信頼できるかどうか検証します。アカウント情報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送信しないことがあります。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**のトークンには，認証のためのユーザ識別情報（**ユーザ名**と**パスワード**）が含まれます。

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。

仮想ユーザ・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認証するために使用されます。共有秘密鍵を使用して，ユーザの資格情報を暗号化し，署名します。そして，KDC（Kerberos Key Distribution Center）と呼ばれる第三者機関が，この資格情報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，現在のユーザの資格情報を使用して取得されます。

VuGen では，Kerberos と Kerberos2 の両方のセキュリティ・トークンに基づいたトークンがサポートされます。Kerberos と Kerberos2 のトークンの主な違いは，Kerberos2 では Security Support Provider Interface（SSPI）が使用されるため，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。

仮想ユーザ・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。

仮想ユーザ・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**，**終了ポイント 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。

スクリプトでのトークン属性の詳細については，*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。

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，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，資格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 メッセージを暗号化しません。したがって，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。

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，1108 ページの「セキュリティを Web サービス・スクリプトに追加する方法」を参照してください。

SAML セキュリティ・オプション

VuGen では，Web サービス用の SAML（Security Assertion Markup Language）がサポートされます。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。詳細については，1110 ページの「SAML セキュリティを追加する方法」を参照してください。

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。出力として，VuGen は符号付き SAML アサーションを提供します。タスクの詳細については，1110 ページの「SAML セキュリティを追加する方法」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。

セキュリティ・シナリオの概要

VuGen 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，VuGen では独自の標準とトランスポートもサポートされます。

このサポートを有効にするには，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。VuGen 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，1130 ページの「ヒントとガイドライン」を参照してください。

このセクションの内容

セキュリティ・モデルの選択

VuGen 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*がサポートされます。この章では，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。

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

プライベート・シナリオ，インポート・シナリオ，および共有シナリオ

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。

シナリオのカテゴリ

シナリオには，Web サービスの設定が記述されています。セキュリティ，エンコーディング，プロキシなどの情報が含まれています。VuGen 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，次のセクション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，**カスタム・バインド**・シナリオの使用をお勧めします。詳細については，1097 ページの「カスタム・バインド・シナリオ」を参照してください。

標準設定の **< シナリオなし >** は，次のような場合に使用します。

- 高度な標準を必要としない単純な Web サービス。
- レガシ・セキュリティ・モデルを使用するスクリプト
- 特定のセキュリティ設定を必要とし，既存のどのシナリオでも利用できない Web サービス。

組み込みシナリオを選択して再生に問題がある場合は，シナリオは必要でなく，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。

組み込みセキュリティ・シナリオは，次のカテゴリに分類されます。

- コア・シナリオ
- セキュリティ・シナリオ
- WCF シナリオ
- 最適化シナリオ

コア・シナリオ

次の表で，組み込みのコア・シナリオについて説明します。

シナリオ名	使用する場合
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

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。

セキュリティ・シナリオ

次の表で，組み込みのセキュリティ・シナリオについて説明します。

シナリオ名	使用する場合
ユーザ名認証	➤ ユーザ名とパスワードを使用して，クライアントをメッセージ・レベルで認証する

このタイプのシナリオでは，ユーザ名 / パスワードを指定します。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。

WCF シナリオ

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

WCF シナリオ名	使用する場合
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・バインド	➤ WS-* standards を使用する Web サービス ➤ 任意の設定の WCF サービス

最適化シナリオ

次の表で，組み込みの最適化シナリオについて説明します。

シナリオ名	使用する場合
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

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

WCF シナリオの設定

このセクションでは，WCF セキュリティ・シナリオに必要な値について説明します。

このセクションの内容

WsHttpBinding シナリオ

認証なし（匿名）

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，1125 ページの「[証明書の選択] ダイアログ・ボックス」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，1125 ページの「[証明書の選択] ダイアログ・ボックス」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。

ユーザ名認証（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資格情報。

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，1125 ページの「［証明書の選択］ダイアログ・ボックス」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）認証

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資格情報。

Federation シナリオ

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，1116 ページの「シナリオを作成する（既存のシナリオがない場合）」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，1125 ページの「［証明書の選択］ダイアログ・ボックス」を参照してください。
- **［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。

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。

- **[発行者のアドレス]**：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。
- **[参照されるバインディング]**：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・シナリオ

Custom Binding シナリオでは，最高度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 などのほかのプラットフォーム上のサービスとともに，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**，**クライアント証明書**，またはカスタム Windows ID などです。証明書の選択については，1125 ページの「[証明書の選択］ダイアログ・ボックス」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，1098 ページの「詳細なシナリオ設定」で説明しているように詳細属性を設定できます。

詳細なシナリオ設定

このセクションでは，エンコーディング，高度な標準，セキュリティ，または HTTP と プロキシの各領域で，セキュリティ・シナリオをカスタマイズする詳細なシナリオ設定について説明します。

すべての設定がすべてのシナリオに関連するとは限りません。したがって，一部の設定をシナリオに応じて無効または非表示にできます。

このセクションの内容

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。

➤ なし
➤ WSA 1.0
➤ WSA 04/08

高度な標準

このタブでは，信頼できるメッセージングやアドレス経由オプションなど，高度な 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ングが要求される］**：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

アドレス経由

場合によって，実際のサーバにメッセージを送信する中間サービスにメッセージを送信することが必要になることがあります。また，メッセージをデバッグ・プロキシに送信する場合もこれを適用できます。これは，WCF の **clientVia** の動作に対応します。

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>
```

論理アドレスは，ユーザ・インタフェースから取得します。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。

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	➤ Any ➤ Thumbprint ➤ IssuerSerial ➤ SubjectKeyIdentifier ➤ RawDataKeyIdentifier

HTTP とプロキシ

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
最大応答サイズ (KB)	連結する前の応答の最大サイズ	標準設定は 65 KB
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，IntegratedWindows Authentication，Basic，Anonymous
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
既定の 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を使わない	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，IntegratedWindows Authentication，Basic，Anonymous

セキュリティ・シナリオの実行準備

次のセクションでは，再生で必要になる場合のある，Web サービス・スクリプトの詳細な変更についていくつか説明します。

このセクションの内容

➤ 1105 ページの「セキュリティ要素のパラメータ化」

➤ 1105 ページの「ユーザ定義ヘッダの保護」

➤ 1106 ページの「反復を使用したユーザのエミュレート」

セキュリティ要素のパラメータ化

スクリプト内のセキュリティ要素は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，それぞれの仮想ユーザ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。

ユーザ定義ヘッダの保護

操作で SOAP ヘッダが使用される場合，VuGen は SOAP ヘッダの署名または暗号化を自動で行いません。署名や暗号化などの保護スキームを組み込むには，シナリオの設定ファイル（.stss）の **behavior** 要素に次の情報を手作業で追加する必要があります。

- 関連操作の soapAction
- ヘッダの XML 名と名前空間
- 保護レベル

次の例は，soapAction（**http://mySoapAction**）を含む送信メッセージを示しています。名前空間 **http://myServiceNamespace** からの XML 要素 **header1** では，暗号化と署名が行われています。同じ名前空間からの **header2** 要素では，署名のみが行われています。

```
<protocols …>
          …
          <behaviors>
               <contractBehaviors>
                    <behavior>
                         <channelProtectionBehavior>
                         <protectionRequirements action="http://mySoapAction">
                              <incomingEncryptionParts>
                                   <header localName="header1" namespace="http://
myServiceNamespace" />
                              </incomingEncryptionParts>
                              <incomingSignatureParts>
                                   <header localName="header1" namespace=" http://
myServiceNamespace " />
                                   <header localName="header2" namespace=" http://
myServiceNamespace " />
                              </incomingSignatureParts>
                         </protectionRequirements>
                    </channelProtectionBehavior>
                    </behavior>
               </contractBehaviors>
          </behaviors>
        </protocols>
```

反復を使用したユーザのエミュレート

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，VuGen は各反復の終了後にすべてのセッションを閉じ，次の反復が始まるとそれらのセッションを再開します。つまり，すべての反復では新しいセッションと仮想ユーザ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，実行環境の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。

タスク

セキュリティを Web サービス・スクリプトに追加する方法

このタスクでは，Web サービス呼び出しにセキュリティを追加する方法について説明します。Web サービスのセキュリティの詳細については，1082 ページの「セキュリティの設定の概要」を参照してください。

このタスクでは，次の手順を実行します。

- 1108 ページの「新しい Web サービス・セキュリティ・ステップを挿入する」
- 1109 ページの「トークンを追加する（任意）」
- 1109 ページの「メッセージの署名または暗号化を追加する（任意）」
- 1110 ページの「メッセージ・タイムアウトを設定する（任意）」
- 1110 ページの「セキュリティ設定を取り消す（任意）」

新しい Web サービス・セキュリティ・ステップを挿入する

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。ほとんどの場合で，セキュリティ設定を **vuser_init** に配置して，セキュリティ範囲がスクリプト全体に適用されるように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適用するには，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 Service 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。

トークンを追加する（任意）

1 ［**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。

2 トークンのタイプを選択します。詳細については，1083 ページの「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。

メッセージの署名または暗号化を追加する（任意）

1 ［**追加**］> ［**メッセージ署名**］，または［**追加**］> ［**暗号化済みデータを追加**］をクリックします。

2 メッセージの署名または暗号化で使用するトークンを選択します。署名および暗号化については，署名 / 暗号化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。

3 対象のトークンを指定するか，フィールドを空白のままにして署名または暗号化をメッセージ本文全体に適用します。詳細については，1083 ページの「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。

SOAP で特定の XPath を暗号化したり署名したりする方法については，1126 ページの「Web サービス・セキュリティの例」を参照してください。

メッセージ・タイムアウトを設定する（任意）

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

セキュリティ設定を取り消す（任意）

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。

SAML セキュリティを追加する方法

このタスクでは，Web サービス呼び出しに SAML セキュリティを追加する方法について説明します。SAML セキュリティの詳細については，1087 ページの「SAML セキュリティ・オプション」を参照してください。

構文については，*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。

このタスクでは，次の手順を実行します。

- 1110 ページの「新しい Web サービス・セキュリティ・ステップを挿入する」
- 1111 ページの「セキュリティ・ポリシーを設定する（任意）」
- 1110 ページの「セキュリティ設定を取り消す（任意）」

1 新しい Web サービス・セキュリティ・ステップを挿入する

a 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。

b ［**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。

c ［**Web サービスのセキュリティ SAML の設定**］を選択して，［**OK**］をクリックします。プロパティ・ボックスが開きます。

2 SAML アサーションを挿入します。

SAML アサーション・メソッドを追加するには，［ステップの追加］ダイアログ・ボックス（［**挿入**］>［**新規ステップ**］）を使用して，**Web サービス署名の SAML アサーション**のステップを追加します。符号なしアサーション，証明書ファイル，およびパスワード（任意）を指定します。

3 セキュリティ・ポリシーを設定する（任意）

ポリシー・ファイルを指定するか，空白のままにして標準設定を使用します。手作業で入力された値は，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。

4 SAML 設定を取り消す（任意）

スクリプト内の特定の位置の設定を削除するには，**Web サービスのセキュリティ SAML のキャンセル**のステップを挿入します。

セキュリティをカスタマイズする方法

このタスクでは，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。

このタスクでは，次の手順を実行します。

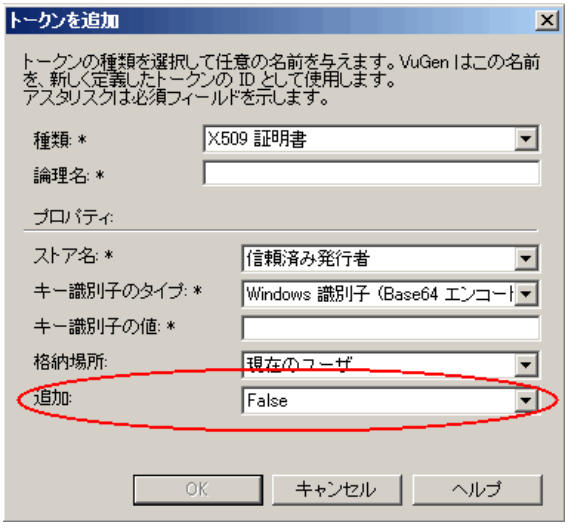
- 1112 ページの「SubjectKeyIdentifier でトークンを参照する（任意）」
- 1113 ページの「Username トークンをカスタマイズする（任意）」
- 1114 ページの「暗号化をカスタマイズする（任意）」
- 1114 ページの「WS-Security をカスタマイズする（任意）」

SubjectKeyIdentifier でトークンを参照する（任意）

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，SKI（Subject Key Identifier）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。

1 1108 ページの「セキュリティを Web サービス・スクリプトに追加する方法」で説明しているようにトークンを追加します。

2 ［トークンを追加］ダイアログ・ボックスで，［追加］オプションを［**False**］に設定します。



3 あるいは，次の設定をスクリプトで行います。

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

4 必要に応じて，1114 ページの「WS-Security をカスタマイズする（任意）」で説明しているように，**useRFC3280** 設定を設定します。

Username トークンをカスタマイズする（任意）

nonce とタイムスタンプを使用して Username トークンをカスタマイズできます。

1 スクリプトで **web_service_set_security** 関数を見つけます。

2 次の表に従って，属性とその値を追加します。

名前	意味	指定可能な値
IsNonceIncluded	トークンに nonce を含める。	**True**（標準設定）または **False**
TimestampFormat	トークンで使用するタイムスタンプ形式。	➤［**なし**］：タイムスタンプなし ➤ **Full**：<created> および <expired> 内部要素を含む <timestamp> 要素 ➤ **Created**：（標準設定）<created> 要素のみ

次に例を示します。

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

暗号化をカスタマイズする（任意）

暗号化をカスタマイズするには，要素全体を暗号化するか，またはその内容のみを暗号化するかを指示します。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。標準では，内容のみが暗号化されます。

トークン全体を暗号化するには，次の手順で行います。

1 スクリプトで **web_service_set_security** 関数を見つけます。

2 値 **Element** を持つ **EncryptionType** 属性を追加します。

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

3 標準設定に戻すには，**EncryptionType** 属性を削除するか，その属性を **Content** に設定します。

WS-Security をカスタマイズする（任意）

Service Test で暗号化に使用するアルゴリズムを変更したり，ほかの低レベル・セキュリティの詳細を変更するには，次の手順で行います。

1 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。

2 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>
```

3 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。	➤ 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	相互運用可能で，Windows 固有ではない Subject Key Identifier を生成する。	➤ True ➤ False（標準設定）

セキュリティ・シナリオを作成して管理する方法

次の手順では，特定サービスのセキュリティ・シナリオを作成してカスタマイズする方法について説明します。

- 1116 ページの「セキュリティ・シナリオ・データ・ダイアログ・ボックスを開く」
- 1116 ページの「シナリオを作成する（既存のシナリオがない場合）」
- 1117 ページの「セキュリティ・シナリオをロードする（既存のセキュリティ・シナリオがある場合）」
- 1118 ページの「詳細設定を行う（任意）」
- 1118 ページの「既存のセキュリティ・シナリオを変更する（任意）」

1 セキュリティ・シナリオ・データ・ダイアログ・ボックスを開く

a ［**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，1014 ページの「[サービスのインポート］ダイアログ・ボックス」で説明しているように，サービスをインポートします。

b ［**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。

2 シナリオを作成する（既存のシナリオがない場合）

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

b ［**シナリオ タイプ**］ボックスで，シナリオを選択します。詳細については，1089 ページの「セキュリティ・モデルの選択」を参照してください。

c シナリオに必要な値を指定します。詳細については，1093 ページの「WCF シナリオの設定」を参照してください。

d 証明書（一部のシナリオのみに適用可能）を指定するには，［**クライアント証明書**］または［**サービス証明書を指定する**］ボックスの横にある［参照］ボタンを押して，［証明書の選択］ダイアログ・ボックスを開きます。詳細については，1125 ページの「［証明書の選択］ダイアログ・ボックス」を参照してください。

- ファイルから証明書を取得するには，［**ファイル**］を選択して，そのパスを検索します。
- Windows の保存場所から証明書を取得するには，［**Windows の保存場所**］を選択します。保存場所と名前を選択します。検索文字列を指定します。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。必要に応じて，秘密鍵のパスワードを指定します。［**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。

3 セキュリティ・シナリオをロードする（既存のセキュリティ・シナリオがある場合）

a 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，1093 ページの「WCF シナリオの設定」で説明しているように設定を変更します。

b 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。

注：誰かがソースの共有シナリオ・ファイルを変更すると，スクリプトはその影響を受けます。

---

4 詳細設定を行う（任意）

［**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。ほとんどのシナリオでは，標準設定が最適です。詳細については，1098 ページの「詳細なシナリオ設定」を参照してください。［**OK**］をクリックして，セキュリティ・シナリオを保存します。

5 既存のセキュリティ・シナリオを変更する（任意）

この特定のサービスだけでなくすべてのスクリプトでグローバルに使用できるセキュリティ・シナリオの作成と変更を行うには，セキュリティ・シナリオ・エディタを使用します。エディタを使用してシナリオを保存することもできます。これにより，ほかのユーザがそのシナリオをロードできます。

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

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

c 必要に応じて，シナリオ設定を変更します。

d ［**保存**］または［**名前を付けて保存**］をクリックします。

6 SOAP ヘッダを保護する（任意）

シナリオの設定ファイル内の **behavior** 要素を手作業で変更します。

a VuGen で，スクリプト・ビューを開きます。［**表示**］>［**スクリプト ビュー**］を選択します。

b スクリプト・エディター内をクリックし，ショートカット・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。

c **WSDL/@config** ディレクトリで，セキュリティ・シナリオの設定ファイル <*service_name*>**.stss** を探します。

d ファイルの behavior セクションを変更します。詳細については，1105 ページの「ユーザ定義ヘッダの保護」を参照してください。

7 反復モードを設定する（任意）

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，次の手順で行います。

a 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，ショートカット・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。

b テキスト・エディタで **default.cfg** ファイルを開きます。

c **[WebServices]** セクションで，toolkit の下に 1 行を追加します。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。

```
[WebServices]
Toolkit=.Net
SimulateNewUserInNewIteration=0
```

d ファイルを保存して閉じます。

詳細については，1106 ページの「反復を使用したユーザのエミュレート」を参照してください

セキュリティ要素をパラメータ化する方法

このタスクでは，スクリプト内でセキュリティ要素を独立してパラメータ化する方法について説明します。

このタスクでは，次の手順を実行します。

- 1120 ページの「セキュリティ・シナリオ・エディタを開く」
- 1120 ページの「各仮想ユーザのシナリオを設定する」
- 1120 ページの「[パラメータ リスト] ウィンドウを開いて，パラメータを作成します。」
- 1120 ページの「パラメータ値を追加する」
- 1120 ページの「パラメータを呼び出す」

1 セキュリティ・シナリオ・エディタを開く

［**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 各仮想ユーザのシナリオを設定する

1116 ページの「セキュリティ・シナリオを作成して管理する方法」で説明しているように各仮想ユーザのシナリオを設定します。**user1**，**user2** などの名前を使用し，新しいフォルダ，**%script root%/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］ウィンドウを開いて，パラメータを作成します。

［**仮想ユーザ**］>［**パラメータ リスト**］を選択します。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。

4 パラメータ値を追加する

値テーブルの各行で，.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追加します。スクリプト・ディレクトリ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。［パラメータ リスト］ダイアログ・ボックスを閉じます。

5 パラメータを呼び出す

a ［**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。

b ［**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。

リファレンス

セキュリティの設定のユーザ・インタフェース

このセクションの内容

［セキュリティ プロパティの設定］ダイアログ・ボックス

このダイアログ・ボックスでは，Web サービス呼び出しのセキュリティ・プロパティを設定できます。

利用方法	**［挿入］>［新規ステップ］>［Web Service Set Security］。［OK］をクリックします。**
関連タスク	1108 ページの「セキュリティを Web サービス・スクリプトに追加する方法」
関連項目	1110 ページの「SAML セキュリティを追加する方法」

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
追加 ▼	［トークンを追加］，［メッセージ署名を追加］，または［暗号化済みデータを追加］ダイアログ・ボックスを開きます。後の 2 つのオプションは，1 つ以上のトークンを定義した後にのみ使用できます。
削除	リストからセキュリティ要素を削除します。
存続期間（秒）	メッセージ・パケットが有効であるとみなされる期間。 **標準設定**：60 秒。

UI 要素	説明
セキュリティ要素	トークン，メッセージ署名，データ暗号化など，セキュリティ要素の名前と種類。
属性	セキュリティ要素の属性。トークンでは，次の属性が含まれます。 ➤ **[ユーザ名]**，**[パスワード]**：トークンの資格情報。 ➤ **[パスワードのオプション]**：パスワードの送信方法。SendPlainText など。 ➤ **[追加]**：トークンをヘッダに含めます。 　➤ **[True]**：SOAP エンベロープ・ヘッダに明示的にトークンを送信します。 　➤ **[False]**：SOAP エンベロープ・ヘッダからトークンを除外します。

[トークンを追加] ダイアログ・ボックス

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
タイプ	トークンのタイプ。 ➤ **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** ➤ **派生トークン**：

UI 要素	説明
論理名	VuGen でトークンの認識に使用されるトークンの任意の名前。
プロパティ	トークンの属性。 ➤ **[ユーザ名]**，**[パスワード]**：トークンの資格情報。 ➤ **[パスワードのオプション]**：パスワードの送信方法。SendPlainText など。 ➤ **[追加]**：トークンをヘッダに含めます。 　➤ **[True]**：SOAP エンベロープ・ヘッダに明示的にトークンを送信します。 　➤ **[False]**：SOAP エンベロープ・ヘッダからトークンを除外します。

セキュリティ・シナリオのユーザ・インタフェース

このセクションの内容

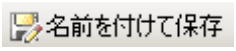
➤［セキュリティ プロパティの設定］ダイアログ・ボックス（1121 ページ）
➤［セキュリティ シナリオ エディタ］ダイアログ・ボックス（1123 ページ）
➤［証明書の選択］ダイアログ・ボックス（1125 ページ）

［セキュリティ シナリオ エディタ］ダイアログ・ボックス

このダイアログ・ボックスでは，スクリプトのセキュリティ・シナリオを定義できます。

利用方法	**[SOA ツール] > [セキュリティ シナリオ エディタ]**
重要情報	特定サービスのシナリオを定義することもできます。詳細については，1116 ページの「セキュリティ・シナリオを作成して管理する方法」を参照してください。
関連タスク	1118 ページの「既存のセキュリティ・シナリオを変更する（任意）」

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
	**[新規作成]**：新しいセキュリティ・シナリオを定義するために，エディタをリセットします。現在のシナリオが変更されていると，保存するように求められます。
	**[読み込む]**：URL またはファイルから既存の共有シナリオを開きます。
	**[保存]**：シナリオ・ファイルを保存します。ファイルを 1 度以上保存していない場合は，名前を要求されます。
	**名前をつけて保存**：新しい場所でシナリオ・ファイルを保存します。
	**[ヘルプ]**：セキュリティ・シナリオのオンライン・ヘルプを開きます。
	**閉じる**：ダイアログ・ボックスを閉じます。
	エンコーディング，信頼できるメッセージング，セキュア・セッション情報，およびプロキシの設定を行うための［詳細設定］ダイアログ・ボックスを開きます。 詳細については，1098 ページの「詳細なシナリオ設定」を参照してください。
シナリオ タイプ	セキュリティ・シナリオのタイプ。シナリオなし，あるいはコア・シナリオ，セキュリティ・シナリオ，WCF シナリオ，または最適化シナリオのサブタイプ。

［証明書の選択］ダイアログ・ボックス

このダイアログ・ボックスでは，ファイルまたは Windows の保存場所から証明書の検索と特定ができます。

利用方法	次のいずれかの方法を使用して，証明書を使用するシナリオを選択します。 ➤ 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）。 ➤ ［サービスの管理］ダイアログ・ボックスで，［**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。 **WsHttpBinding** または **Federation** など，クライアント証明書またはサービス証明書を使用する WCF シナリオを選択します。［**証明書**］フィールドで，「参照］ボタンをクリックします。
重要情報	これが適用されるのは，クライアント，サーバ，またはサービスの証明書を指定できるセキュリティ・シナリオのみです。
関連タスク	➤ 1116 ページの「シナリオを作成する（既存のシナリオがない場合）」 ➤ 1117 ページの「セキュリティ・シナリオをロードする（既存のセキュリティ・シナリオがある場合）」

ファイルから証明書を選択する

［**ファイル**］を選択すると，ダイアログ・ボックスに，次の表に示されたユーザ・インタフェース要素が表示されます。

UI 要素	説明
	**［参照］**：.pem，.arm，.der，または .pfx 拡張子付きの証明書ファイルを検索できます。
ファイル	証明書ファイルの完全パス。
パスワード（任意）	証明書にアクセスするために必要なパスワード。

Windows の保存場所から証明書を選択する

［**Windows の保存場所**］を選択すると，ダイアログ・ボックスに，次の表に示されたユーザ・インタフェース要素が表示されます。

UI 要素	説明
検索(F)	証明書の検索を開始します。
インポート元	証明書の場所。 ➤ **Windows の保存場所** ➤ **ファイル**
保存場所	保存場所。例：**現在のユーザ**。
保存名	保存名。例：**AuthRoot**。
検索テキスト	証明書名に一致させるテキスト。
パスワード（任意）	証明書にアクセスするために必要なパスワード。
< 証明書リスト >	Windows の保存場所にある証明書のリスト。件名，発行者，プライベート，保存場所，保存名で並べ替えられています。

Web サービス・セキュリティの例

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);
```

注：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，1112 ページの「SubjectKeyIdentifier でトークンを参照する（任意）」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=pOI0+1iuotKLIO91nhjDg5reEw0=",
  "StoreLocation=CurrentUser", "Add=False",
  ENCRYPTED_DATA, "UseToken=serviceCert",
 LAST);
```

ヒントとガイドライン

このセクションでは，WCF サービスのテストおよびセキュリティ・シナリオの定義についての入門ガイドを提供します。

WCF ガイドライン

このセクションでは，VuGen を使用して WCF をテストする方法について説明します。

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の種類を選択します。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** authentication 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，サービス資格情報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

1090 ページの「シナリオのカテゴリ」で説明しているように，［**WCF**］>［**カスタム バインド**］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

1090 ページの「シナリオのカテゴリ」で説明しているように，［**WCF**］>［**カスタム バインド**］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 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，*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。

［**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。

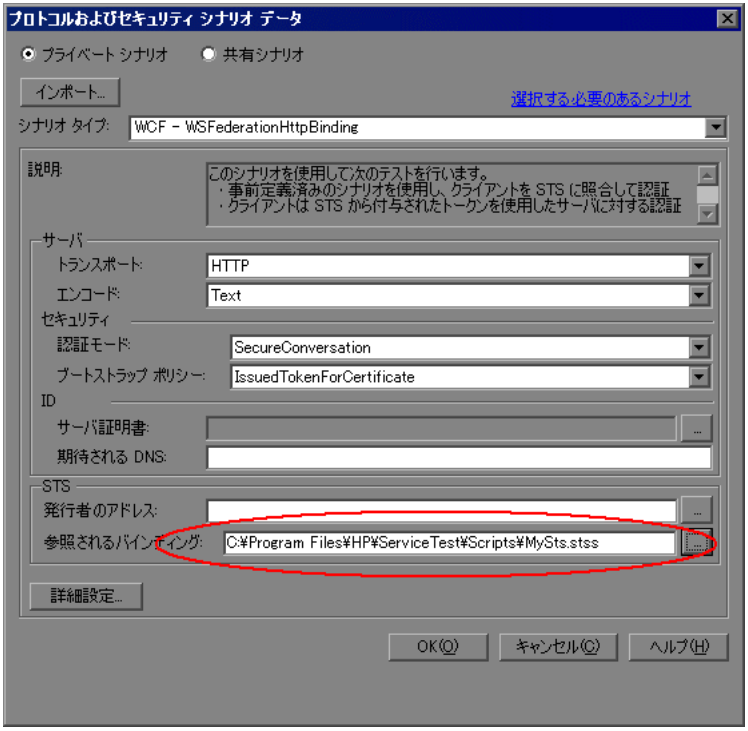
STS との通信を設定するには，次の手順で行います。

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。



一般的なセキュリティ・テスト

このセクションでは，Service Test を使用して一般的なセキュリティ・テストを行う方法についてのサマリが提供されます。

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資格情報モードを選択して，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，1098 ページの「詳細なシナリオ設定」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，1082 ページの「セキュリティの設定の概要」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，*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。

［**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，次の手順で行います。

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。

高度な標準のテストに関するヒント

このセクションでは，VuGen を使用して高度な標準のテストを行う方法について要約します。

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，1098 ページの「詳細なシナリオ設定」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［**エンコーディング**］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，1098 ページの「詳細なシナリオ設定」を参照してください。

39

Web サービス - サービスのエミュレーション

本章の内容

概念

タスク

リファレンス

概念

エミュレーション・サービスの概要

サービス・エミュレーション・コンソールを使用して，環境内のほかの Web サービスをテストするためのサービスのエミュレーションを作成できます。

注：サービス・エミュレーション・ツールは，Service Test ライセンスでのみ使用できます。詳細については，HP のサポートにお問い合わせください。

エミュレーション・サービスには，次の利点があります。

- **早い段階での開発**：実際のサービスにアクセスできない開発の早い段階でテストを設計，実行できます。たとえば，サービスの開発が不完全，またはサービスのホストが使用できない場合，エミュレーション・サービスを使用してアプリケーションのその他のサービスをテストできます。
- **クライアントのテスト**：サービス・エミュレータを使用して，クライアント・アプリケーションの機能をテストできます。
- **コンポーネントの分離**：テスト対象のサービスが複合システムの一部である場合，サービス・エミュレータを使用すると，依存関係なしで 1 つサービスをテストできます。依存するサービスは，不完全，一時的に使用できない，または単にテスト計画を妨げるものである場合があります。

サービス・エミュレーション・コンソールを使用すると，遅延やルールを通じてサービスの動作を定義できます。

サービスは，サービスの操作およびパラメータを定義する WSDL を指定して作成します。WSDL ファイルを指定すると，サービス・エミュレーション・ツールによって，WSDL の現在の構造を使用してサービスの入力および出力データの構造が定義されます。

元の WSDL を変更しても，エミュレーション・サービスには反映されません。更新された WDSL を使用するには，エミュレーション・サービスを再作成します。

このセクションの内容

サーバ・エミュレーション・ホスト

ホストは，エミュレーション・サービスの要求を送信するマシンです。エミュレーション・サーバは，HP Service Test の設定時にインストールした Apache Tomcat サーバです。Tomcat サーバがインストールされているほかのマシンを指定することもできます。

ホストの選択の詳細については，1145 ページの「ホストを追加する」を参照してください。

エミュレーション・サーバの起動と停止の詳細については，1145 ページの「サーバを起動する」を参照してください。

エミュレーション・サービスの動作

サービス・エミュレーション・コンソールでは，各操作の動作を指定できます。サービスの操作の動作は次によって指定します。

➤ 操作の応答の遅延

➤ 標準設定の応答の提供

➤ サービス・エミュレーション・ルールの設定

操作の応答の遅延

サーバが応答するまでの時間遅延を設定できます。

操作全体にグローバル遅延を設定する方法については，1138 ページの「標準設定の応答の提供」を参照してください。

特定のルールに遅延を設定する方法については，1153 ページの「[応答] 表示枠」を参照してください。

標準設定の応答の提供

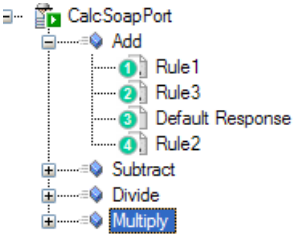
標準設定の応答は，ルールが存在しない場合に操作で使用される値のセットです。

標準設定の応答の値は，手作業で指定するか，サンプル結果を含む XML ファイルからインポートできます。

サービス・エミュレーション・ルールの設定

標準設定の応答の設定に加え，動作ルールも設定できます。ルールを通して，サービスの個別の動作（要求，入力データに基づいて期待される応答）を定義します。

操作には複数のルールを設定できます。ルールは優先順位に従って並べます。ルール間に矛盾がある場合，エミュレーション・サービスはルールの位置に従います。番号アイコンはルールの優先順位を示します。次の例では，**Rule 1** の優先順位が最も高くなっています。ルールの優先順位を変更する方法については，1148 ページの「コンソール」を参照してください。



設定できる動作ルールには，XML ルールとコード・ルールという 2 種類があります。

XML ルール

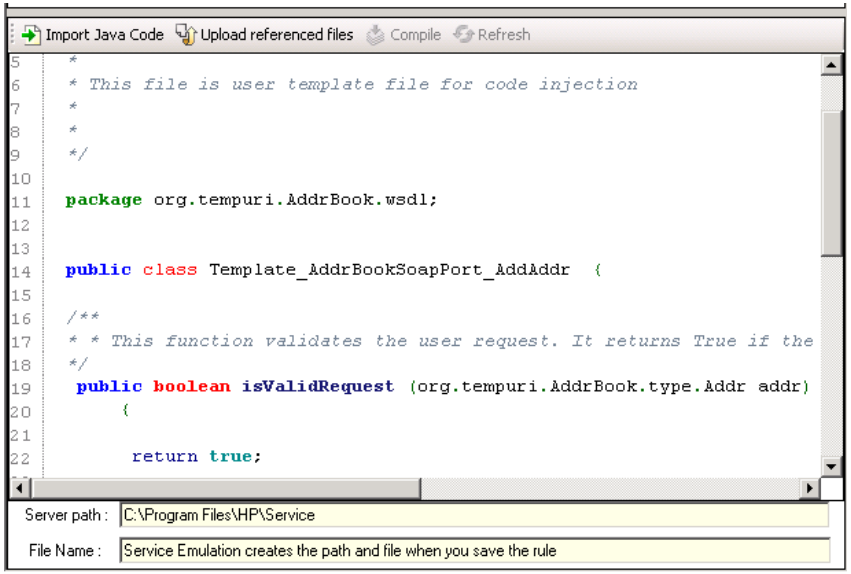
XML ルールは，要求または応答 XML 要素に定数値を割り当てるルールです。値は手作業で設定するか，既存の XML SOAP ファイルをインポートできます。たとえば，加法演算で，1 番目の引数を 4，2 番目の引数を 5，結果を 9 と指定できます。

ルール定義から引数を除外できます。これは，特定の応答が返されるようにサービスを設定し，引数のいずれかの値を無視する場合に便利です。たとえば，乗法演算で，2 番目の引数の値に関係なく，1 番目の引数が 0 の場合，結果が 0 になることを示すルールを設定できます。詳細については，1146 ページの「XML ルールを定義する（任意）」を参照してください。

コード・ルール

コード・ルールは，エミュレーション・サービスの応答値をプログラムできるコードのスニペットです。特定のルールが適用されるかどうかを判断し，それに応じて応答するには，条件ステートメントを使用できます。

コード・ルールでは，標準の Java コードおよびクラスを使用して，特定のルールが適用されるかどうかを判断できます。



コードでは，**isValidRequest** と **getResponse** という 2 つのメソッドが主に使用されます。**isValidRequest** メソッドによって，入力引数の値がチェックされ，**getResponse** メソッドによって計算され応答が返されます。

isValidRequest が **true** を返すと，サービス・エミュレーションではこのルールを使用し，**getResponse** を使用して応答が返されます。**isValidRequest** が **false** を返すと，ツリー内の次のルールへと移動します。ほかにルールがない場合は，標準設定の応答が使用されます。

コード・ルールの例

次の例では，コード・ルールによって応答の値がチェックされます。値が **1** と **2** の場合，応答として値の合計を使用するようサービスに指示します。

```
/**
* Service: CalcSoapPort
* Operation: Add
*
* This file is user template file for code injection
*
*/
package org.tempuri.Calc.wsdl;

public class Template_CalcSoapPort_Add  {

 public boolean isValidRequest (double a, double b) throws java.rmi.RemoteException
    {

     if ((a == 1) && (b ==2))
          return true;
     else
          return false;

    }

   public double getResponse (double a, double b) throws java.rmi.RemoteException
    {
     return (a+b);
    }
    }
```

次の例では，応答に外部クラスを使用する類似のスニペットを示します。参照クラスを使用する場合は，［**参照されたファイルのアップロード**］ボタンを使用して，手作業でサーバにインポートする必要があります。

```
package org.tempuri.Calc.wsdl;

public class Template_CalcSoapPort_Add  {

 public boolean isValidRequest (double a, double b) throws java.rmi.RemoteException
    {

     if ((a == 1) && (b ==2))
          return true;
     else
          return false;

    }
public double getResponse (double a, double b) throws java.rmi.RemoteException
    {

     double resp = -1;

     try {
          MyTestAdd adr = new MyTestAdd();
          adr.SetNumbers (10,20);

          resp=adr.GetAdrResponse();

          return resp;
          }
     catch(Exception ex) {
               System.out.println ("getResponse Exception:" + ex);
          }
          return resp;
    }
    }
```

コード・ルールの設定については，1147 ページの「コード・ルールを定義する（任意）」を参照してください。

クライアントのテスト

ホストは，エミュレーション・サービスの要求を送信するマシンです。エミュレーション・サーバは，HP Service Test の設定時にインストールした Apache Tomcat サーバです。Tomcat サーバがインストールされているほかのマシンを指定することもできます。

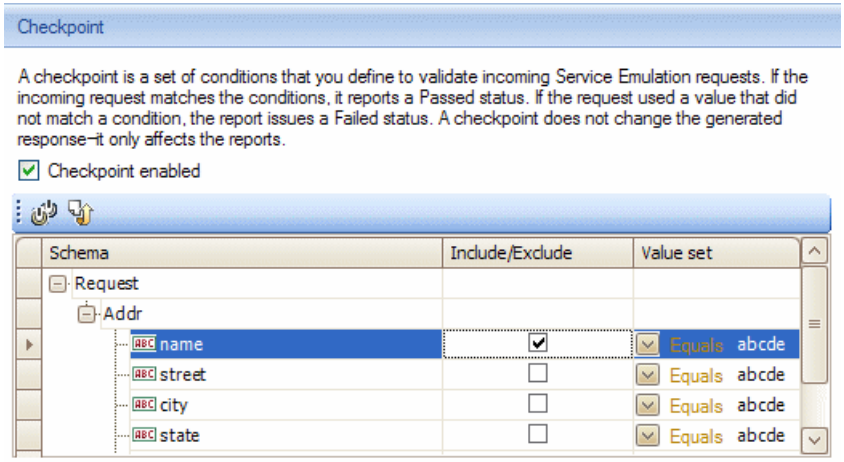
クライアントのテストについては，1147 ページの「クライアントのテスト・ツールを設定する（任意）」を参照してください。

クライアントのテストは，次の領域で実行できます。

- チェックポイント
- ラベル
- サービス・エミュレーション・レポート

チェックポイント

サービス・エミュレーション・チェックポイントとは，受信要求を検証するために定義する一連の条件です。条件は引数の値と比較演算子を組み合わせたもので，要求はこれらの条件に適合することが期待されます。たとえば，引数 **A** の値が 2 以上であることを検証するチェックポイントを設定できます。



エミュレーション・サービスの受信要求が条件に一致すると，「**成功**」ステータスがレポートされます。ただし，条件に一致しない値が要求で使用されていた場合，レポートでは「**失敗**」ステータスが発行されます。

チェックポイントによって生成される応答が変更されることはありません。

チェックポイントは操作全体または特定のルールに対して設定できます。操作およびその下のルールの両方にチェックポイントを定義する場合，「**成功**」とみなされるためには，受信要求は両方のチェックポイントに適合する必要があります。

ルール・チェックポイントは，親ルールがアクティブ化されている場合にのみアクティブ化されます。つまり，ルールのチェックポイントは要求がルールに適合する場合にのみ検証されます。

チェックポイントについては，1147 ページの「チェックポイントを有効にし定義する（任意）」を参照してください。

ラベル

ラベルを使用して，エミュレーション・サーバに送信されたサービス・コールに日時およびタイム・スタンプを作成できます。

これらのラベルを使用して，レポートの生成時にデータをフィルタリングできます。詳細については，1147 ページの「クライアントのテスト・ツールを設定する（任意）」を参照してください。

サービス・エミュレーション・レポート

レポートを作成して，エミュレーション・サービスに関する情報を表示，フィルタリング，および並べ替えることができます。レポートに含める情報と表示順序を選択します。日付またはラベル，サービス，操作，チェックポイント・ステータス，クライアント IP，セッション ID でフィルタリングできます。

詳細については，1147 ページの「クライアントのテスト・ツールを設定する（任意）」を参照してください。

ログのクリア

Service Test では，エミュレーション・サービスを通じて送信されたサービス呼び出しを含むデータベース・ログをクリアできます。削除するエントリの範囲を日付またはラベルで指定できます。ルール，チェックポイント，標準設定の応答などのユーザ定義設定には影響しません。

詳細については，1147 ページの「クライアントのテスト・ツールを設定する（任意）」を参照してください。

タスク

エミュレーション・サービスを作成する方法 - ワークフロー

このタスクでは，エミュレーション・サービスを作成するワークフローについて説明します。

このタスクでは，次の手順を実行します。

- 1145 ページの「ホストを追加する」
- 1145 ページの「サーバを起動する」
- 1146 ページの「エミュレーション・サービスを新規作成する」
- 1146 ページの「標準設定の動作を設定する（任意）」
- 1146 ページの「XML ルールを定義する（任意）」
- 1147 ページの「コード・ルールを定義する（任意）」
- 1147 ページの「チェックポイントを有効にし定義する（任意）」
- 1147 ページの「エミュレーション・サービスを統合する」

1 ホストを追加する

Web サービス呼び出しの送信先のホストを指定します。詳細については，1156 ページの「[ホストの選択] ダイアログ・ボックス」を参照してください。

2 サーバを起動する

Service Test エミュレーション・サーバが，指定したホスト・マシン上でアクティブになっていることを確認します。標準設定では，開いたときに Service Test によって自動的に Tomcat エミュレーション・サーバが起動されます。

サーバがアクティブかどうかを確認するには，次の手順で行います。

ブラウザに次の URL を入力します。
http://<hostname>:8080/ServiceEmulation/index.jsp
サーバがアクティブな場合は，ブラウザに **HP サービス・エミュレーション** が表示されます。

サーバを起動するには，次の手順で行います。

［**スタート**］>［**すべてのプログラム**］>［**HP LoadRunner**］>［**Service Test**］>［**エミュレーション サービスの開始**］を選択します。

サーバを停止するには，次の手順で行います。

［**スタート**］>［**すべてのプログラム**］>［**HP LoadRunner**］>［**Service Test**］>［**エミュレーション サービスの停止**］を選択します。

サーバのトラブルシューティングについては，1164 ページの「トラブルシューティングと制限事項」を参照してください。

3 エミュレーション・サービスを新規作成する

WSDL のファイルまたは URL を指定して，このサービスのホストを選択します。詳細については，1158 ページの「[新規のエミュレーション サービス］ダイアログ・ボックス」を参照してください。

4 標準設定の動作を設定する（任意）

関連するルールがない場合に使用する，サービスの標準設定の応答を設定します。詳細については，1151 ページの「[詳細］表示枠 - 標準設定の応答」を参照してください。

5 XML ルールを定義する（任意）

ルールの動作を設定するために，各操作に 1 つ以上の XML ルールを定義します。XML ルールを追加するには，操作を選択し，ショートカット・メニューから［**新規 XML ルール**］を選択します。

- 1153 ページの「[要求］表示枠」の説明に従って，要求にルールを設定します。
- 1153 ページの「[応答］表示枠」の説明に従って，応答の詳細を設定します。

6 コード・ルールを定義する（任意）

組み込みの Java テンプレートを使用して，必要に応じてコードを変更し，各操作に 1 つ以上のコード・ルールを定義します。コード・ルールを追加するには，操作を選択し，ショートカット・メニューから［**新規コード・ルール**］を選択します。変更を加えた後にルールを保存します。詳細については，1154 ページの「［詳細］表示枠 - コード・ルール」を参照してください。

7 チェックポイントを有効にし定義する（任意）

各ルールごと，または各操作ごとに 1 つ以上のチェックポイントを定義します。チェックポイントを追加するには，エンティティ（操作またはルール）を選択し，ショートカット・メニューから［**新規チェックポイント**］を選択します。詳細については，1155 ページの「［詳細］表示枠 - チェックポイント」を参照してください。

8 クライアントのテスト・ツールを設定する（任意）

次のツールを使用して，クライアントのテストを実行します。

- **ラベルの定義。**1163 ページの「［新規ラベル］ダイアログ・ボックス」を参照してください。
- **レポートの生成。**1159 ページの「サービス・エミュレーション・レポート・ウィザード」を参照してください。
- **Web サービス・ログのクリア。**1162 ページの「［サービス呼び出しログのクリア］ダイアログ・ボックス」を参照してください。

9 エミュレーション・サービスを統合する

エミュレーション・サービスを作成したら，テストのためにそれをスクリプトに組み込みます。

a サービス・エミュレーション・コンソールの左側の表示枠でサービスを選択します。［**エミュレーション・サービス**］セクションから［**WSDL の場所**］をクリップボードにコピーします。

b ［**サービスの管理**］ダイアログ・ボックスを開き，サービスを選択します。［**優先アドレス**］オプションを選択します。［**サービス アドレス**］ボックスに WSDL の場所を貼り付けます。テストの実行中に，VuGen からサービス要求がその場所へ送信されます。

リファレンス

サービス・エミュレーション・コンソールのユーザ・インタフェース

このセクションで説明するコンソールの要素

- コンソール（1148 ページ）
- ［ホストの選択］ダイアログ・ボックス（1156 ページ）
- ［ホスト設定］ダイアログ・ボックス（1157 ページ）
- ［ホスト名］表示枠（1157 ページ）
- ［新規のエミュレーション サービス］ダイアログ・ボックス（1158 ページ）
- サービス・エミュレーション・レポート・ウィザード（1159 ページ）
- ［サービス呼び出しログのクリア］ダイアログ・ボックス（1162 ページ）
- ［新規ラベル］ダイアログ・ボックス（1163 ページ）

コンソール

サービス・エミュレーション・コンソールでは，エミュレーション・サービスを作成し，その動作を定義できます。

利用方法	**［スタート］＞［すべてのプログラム］＞［HP LoadRunner］＞［Service Test］＞［サービス エミュレーション コンソール］**

重要情報	サービス・エミュレーション・ツールでは，ルールおよび期待値を保存するために，SQL サーバ・データベースが使用されます。LoadRunner with Service Test の場合は，SQL Microsoft SQL Server Management Studio Express 9.00 の再頒布可能バージョンを手作業でインストールする必要があります。HP LoadRunner メディアの **Additional Components¥MSSQL2005Express** フォルダにあるインストール・ファイル **install_MSSQL2005Express.bat** を実行します。
関連タスク	➤ 1146 ページの「標準設定の動作を設定する（任意）」 ➤ 1146 ページの「XML ルールを定義する（任意）」 ➤ 1147 ページの「コード・ルールを定義する（任意）」 ➤ 1147 ページの「チェックポイントを有効にし定義する（任意）」

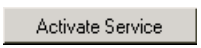
ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
	**[新規ホストの追加]**：[ホストの選択] ダイアログ・ボックスが開きます。
	**[新規のエミュレーション サービス]**：[新規のエミュレーション サービス] ダイアログ・ボックスが開きます。
	**[新規 XML ルール]**：右側の表示枠にグリッドが表示され，新規の XML ルールを定義できます。
	**[新規コード ルール]**：右側の表示枠にエディタが表示され，新規のコードベース・ルールを定義できます。
	**[新規チェックポイント]**：チェックポイント・グリッドが右側の表示枠に表示されます。
	**[保存]**：すべてのルールおよび変更でデータベースが更新されます。 **ヒント**：タイトル・バーのアスタリスクはサービスが保存されていないことを示します。
	**[更新]**：元の場所から WSDL が更新されます。
	**[削除]**：選択したエンティティ（エミュレーション・サービス，ルール，またはチェックポイント）が削除されます。

UI 要素	説明
	**[アクティブ化]**：サービスがアクティブ化されます。
	**[サービスを非アクティブ化]**：サービスが一時的に非アクティブ化されます。
	**[ルールを上に移動]**：ルールが上に移動して，優先順位が上がります。
	**[ルールを下に移動]**：ルールが下に移動して，優先順位が下がります。
	**[ラベルの追加]**：フィルタリングのためにサービスにラベルが追加されます。
	**[レポートの生成]**：レポートの生成ウィザードが開きます。
	**[サービス呼び出しログのクリア]**：特定の日付範囲内に送信されたデータベース・ログからサービス呼び出しをクリアできます。
< ホスト名表示枠 >	定義されているすべてのホストのリスト。 **標準設定値**：**localhost**
< 詳細表示枠 >	左側の表示枠で選択した次のエンティティの詳細。[**サービス**]，[**操作**]，[**標準設定の応答**]，[**XML ルール**]，[**コードルール**]，または [**チェックポイント**]。

[詳細] 表示枠 - サービス

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
	左側の表示枠で選択したサービスのみがアクティブ化されます。
	左側の表示枠で選択したサービスのみが非アクティブ化されます。
名前 / パッケージ	エミュレーション・サービスの名前とパッケージ。
元のサービス /WSDL の場所	元の WSDL のインポート元のパスまたは URL。

UI 要素	説明
エミュレーション・サービス /WSDL の場所	エミュレーション・サーバが実行されているマシン上のサービスの URL。
エミュレーション・サービス / エンドポイントの場所	要求の送信先のエンドポイント。

［詳細］表示枠 - 操作

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
名前	操作名（読み取り専用）
コメント	操作に関するコメントの編集可能なフィールド。

［詳細］表示枠 - 標準設定の応答

ルールが存在しない場合や要求に一致するルールがない場合，サービス・エミュレータによって標準設定の応答が返されます。ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
	**［列の追加］**：新しい［**Value Set**］カラムが追加されます。
	**［削除］**：選択した［**Value Set**］カラムが削除されます。
	**［リセット］**：選択した値セットの変更が破棄され，WSDL の標準設定の応答値が使用されます。
	**［SOAP のインポート］**：［**SOAP メッセージを含むファイルを選択してください**］ダイアログ・ボックスが開き，SOAP メッセージから値をロードできます。
応答タイプ	SOAP 応答の種類は次のとおりです。［**応答**］または［**Fault**］

UI 要素	説明
応答の遅延 (秒)	サーバの応答前に導入される遅延の種類は次のとおりです。［**定数**］または［**ランダム**］。定数の遅延の場合，秒で遅延を指定します。ランダムな遅延の場合，時間の範囲を指定します。 **ヒント：ミリ秒を指定するには，小数点以下の値を使用します。たとえば，20 ミリ秒の場合は 0.02 と入力します。**
スクロール モード	**[Freeze schema column]：**スキーマ・カラムが固定され，値セットを自由にスクロールできます。
［スキーマ］表示枠	サービス応答のスキーマは次のとおりです。 ➤［**スキーマ**］：要素の XML 表現 ➤［**Value set**］：各引数の値セット。複数の値セットを追加するには，［**列の追加**］ボタンを使用します。SOAP メッセージから値をロードするには，手動で値を入力するか，［**SOAP のインポート**］ボタンを使用します。
値の選択	値の選択方法は次のとおりです。［**順次**］または［**ランダム**］。

［詳細］表示枠 - XML ルール

ルールが存在しない場合や要求に一致するルールがない場合，サービス・エミュレータによって標準設定の応答が返されます。ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
名前	左側の表示枠のツリー階層およびレポートに表示される XML ルールの名前。
［要求］表示枠	エミュレーション・サーバに送信される要求についての詳細。詳細については，**1153 ページの「［要求］表示枠」**を参照してください。
［応答］表示枠	エミュレーション・サーバから受信する応答についての詳細。詳細については，**1153 ページの「［応答］表示枠」**を参照してください。

［要求］表示枠

UI 要素	説明
	**［リセット］**：選択した値セットの変更が破棄され，WSDL の標準設定の応答値が使用されます。
	**［SOAP のインポート］**：［**SOAP メッセージを含むファイルを選択してください**］ダイアログ・ボックスが開き，SOAP メッセージから値をロードできます。
	**［データ タイプ］**：マウスを引数の上に移動すると，そのデータ・タイプが表示されます。
	比較演算子のリスト。 ➤ 数値および日付の場合は次のとおりです。**Equals**，**GreaterOrEqual**，**LessOrEqual**，**GreaterThan**，または **LessThan**。 ➤ 文字列の場合は次のとおりです。**Equals**，**Not Equal**，**Contains**，**EndsWith**，または **StartsWith**。 **ヒント：正規表現を使用することもできます。**
［スキーマ］カラム	XML 要素のスキーマ。
［含める］/［除外する］カラム	サーバへの要求に引数を含めるのか除外するのかを指定します。
［Value set］カラム	各引数の値セットと比較演算子。比較演算子を選択し，引数のデータ・タイプを決定して値を指定します。

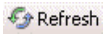
［応答］表示枠

UI 要素	説明
	**［列の追加］**：新しい［**Value Set**］カラムが追加されます。
	**［削除］**：選択した［**Value Set**］カラムが削除されます。
	**［リセット］**：選択した値セットの変更が破棄され，WSDL の標準設定の応答値が使用されます。
	**［SOAP のインポート］**：［**SOAP メッセージを含むファイルを選択してください**］ダイアログ・ボックスが開き，SOAP メッセージから値をロードできます。
応答タイプ	SOAP 応答の種類は次のとおりです。［**応答**］または［**Fault**］

UI 要素	説明
応答の遅延 (秒)	サーバの応答前に導入される遅延の種類は次のとおりです。［**標準設定**］(標準設定の応答の値)，［**定数**］または［**ランダム**］。定数の遅延の場合，秒で遅延を指定します。ランダムな遅延の場合，時間の範囲を指定します。 **ヒント：ミリ秒を指定するには，小数点以下の値を使用します。たとえば，20 ミリ秒の場合は 0.02 と入力します。**
スクロール モード	**[Freeze schema column]**：スキーマ・カラムが固定され，値セットを自由にスクロールできます。
［スキーマ］表示枠	サービス応答のスキーマは次のとおりです。 ➤［**スキーマ**］：要素の XML 表現 ➤［**Value set**］：各引数の値セット。複数の値セットを追加するには，［**列の追加**］ボタンを使用します。SOAP メッセージから値をロードするには，手動で値を入力するか，［**SOAP のインポート**］ボタンを使用します。
値の選択	値の選択方法は次のとおりです。［**順次**］または［**ランダム**］。

［詳細］表示枠 - コード・ルール

サービス・エミュレーションの［コード ルール］領域では，サービスに適用される Java ベースのルールを編集できます。ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
	**[Java コードをインポート]**：既存の Java ファイルが編集表示枠にコピーされます。必要に応じてファイルを編集します。 **注**：インポートされた Java コードは，コード・ルールのコードと同じパッケージに格納される必要があります。
	**[参照されたファイルのアップロード]**：コードによって参照された外部クラスをアップロードできます。
	**[コンパイル]**：コードの有効性をチェックするためにサーバ上でコードがコンパイルされます。
	**[更新]**：サーバ・マシン上で変更された場合は，コードが同期されます。

UI 要素	説明
<ユーザ・テンプレート>	エミュレーション・サービスに適用されるコードが含まれるテンプレート。 **注：**コードがアプリケーションに悪影響を与える可能性があります。**System.exit(0)** など，サービス・エミュレーション・サービスに干渉したり，停止させたりするコードは使用しないでください。組み込みの **isValidRequest** および **getResponse** メソッドの名前またはパラメータを削除したり，変更したりしないでください。
ファイル名	.java 拡張子を使用して自動的に生成されたコード・ルールのファイル。
サーバ パス	サーバの場所。

複雑な入力引数については，http://ws.apache.org/axis/java/apiDocs/javax/xml/rpc/holders/Holder.html（英語サイト）のサーバに関する注意事項を参照してください。

［詳細］表示枠 - チェックポイント

サービス・エミュレーション・チェックポイント領域では，受信要求を検証するための条件を設定できます。チェックポイントは操作全体または個々のルールに適用できます。ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
	**［リセット］**：［Value set］カラムのすべての変更が破棄されます。
	**［SOAP のインポート］**：［**SOAP メッセージを含むファイルを選択してください**］ダイアログ・ボックスが開き，SOAP メッセージから値をロードできます。
	**［データ タイプ］**：マウスを引数の上に移動すると，そのデータ・タイプが表示されます。
	比較演算子のリスト。 ➤ 数値および日付の場合は次のとおりです。**Equals**，**GreaterOrEqual**，**LessOrEqual**，**GreaterThan**，または **LessThan**。 ➤ 文字列の場合は次のとおりです。**Equals**，**Not Equal**，**Contains**，**EndsWith**，または **StartsWith**。 **ヒント：正規表現を使用することもできます。**

UI 要素	説明
チェックポイントは有効です	チェックポイント検証メカニズムが有効化されます。
［スキーマ］カラム	XML 要素のスキーマ。
［含める］/［除外する］カラム	サーバへの要求に引数を含めるのか除外するのかを指定します。
［Value set］カラム	各引数の値セットと比較演算子。比較演算子を選択し，引数のデータ・タイプを決定して値を指定します。

［ホストの選択］ダイアログ・ボックス

エミュレーション・サービスのホスト・マシンとポートを選択できます。

利用方法	［**ファイル**］>［**新規ホスト**］，または［新規ホスト］ボタン
重要情報	ホスト・マシンには Tomcat サーバが必要です。これは，localhost に自動的にインストールされます。
関連タスク	1145 ページの「ホストを追加する」

含まれている要素は次のとおりです。

GUI 要素	説明
	**［参照］**：［Browse for Computer］ダイアログ・ボックスが開きます。
ホスト名	エミュレーション・サービスをホストするマシンの名前または IP アドレス
ポート	指定したホスト・マシンのポート番号 **標準設定値**：**8080** **ヒント**：8080 ポートが競合する場合は，Service Test のインストール先の **apache-tomcat-5.5.17¥conf** フォルダにある **Server.xml** ファイルを編集します。ポートを 8080 から別のポートに変更します。新しいポート番号で新しいローカルホスト・サーバを作成します。

［ホスト設定］ダイアログ・ボックス

ホスト名を表示し，データベース接続を設定できます。

利用方法	左側の表示枠のホスト名ツールバーで，［**ホスト設定**］ボタン をクリックします。
重要情報	データベースでユーザ名とパスワードが必要な場合は，ここに入力する必要があります。
関連タスク	1145 ページの「ホストを追加する」
関連項目	1156 ページの「［ホストの選択］ダイアログ・ボックス」

含まれている要素は次のとおりです。

GUI 要素	説明
サーバ	サーバの詳細：データベースのホスト名 ¥ インスタンス
ポート	エミュレーション・サービスによって使用されるデータベース・サーバのポート
Username¥Password	データベース・サーバにログインするための資格情報 **標準設定値：localhost の場合は空**

［ホスト名］表示枠

ホスト名が表示され，サーバを開始または停止，および［ホスト設定］ダイアログ・ボックスを開くことができます。

利用方法	左側の表示枠に自動的に表示されます。
重要情報	左側の表示枠の下部に，使用可能な各ホストのツールバーが表示されます。
関連タスク	1145 ページの「サーバを起動する」
関連項目	➤ 1156 ページの「［ホストの選択］ダイアログ・ボックス」 ➤ 1157 ページの「［ホスト設定］ダイアログ・ボックス」 ➤ 1148 ページの「コンソール」

含まれている要素は次のとおりです（ラベルのない GUI 要素は山括弧で囲んで示します）。

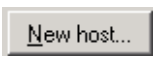
GUI 要素	説明
	**[Start Emulation Server]**：Tomcat サーバを開始し，ServiceEmulationClient プロセスが起動されます。
	**[Stop Emulation Server]**：Tomcat サーバと ServiceEmulationClient プロセスが停止されます。
	**[ホストの設定]**：[ホスト設定] ダイアログ・ボックスが開き，データベース接続を設定できます。
<hostname>	表示されたエミュレーション・サービスをホストするマシン **標準設定値**：localhost

[新規のエミュレーション サービス] ダイアログ・ボックス

このダイアログ・ボックスでは，WSDL を指定し，ホストを選択することによってエミュレーション・サービスを定義できます。

利用方法	**[サービス] > [新規のエミュレーション サービス]**
重要情報	スキーマがインポートされる WDSL のインポートはサポートされていません。このタイプの WSDL をインポートするには，ローカル・サーバにデプロイしてから，生成された URL を使用してインポートします。
関連タスク	1146 ページの「エミュレーション・サービスを新規作成する」
関連項目	1156 ページの「[ホストの選択] ダイアログ・ボックス」

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
	**[参照]**：ブラウザ（[**URL**] を選択した場合）または [ファイルを開く] ダイアログ・ボックス（[**ファイル**] を選択した場合）が開きます。
	[ホストの選択] ダイアログ・ボックスを開きます

UI 要素	説明
ホストの選択	使用可能なホスト・マシンのドロップダウン・リスト。
WSDL を次から選択	WSDL のソース。 ➤ **URL** ➤ **ファイル**

サービス・エミュレーション・レポート・ウィザード

このウィザードでは，エミュレーション・サービスのレポートを生成できます。

利用方法	**［クライアントのテスト］ > ［レポートの生成］**
関連タスク	1147 ページの「クライアントのテスト・ツールを設定する（任意）」
ウィザード・マップ	このウィザードには，次のページが含まれています。 ［ようこそ］ > ［列の選択］ページ > ［フィルタ］ページ > ［並べ替え］ページ
関連項目	1163 ページの「［新規ラベル］ダイアログ・ボックス」

［列の選択］ページ

このウィザード・ページでは，レポートに含めるカラムを選択できます。

重要情報	このウィザードに関する一般情報は，1159 ページの「サービス・エミュレーション・レポート・ウィザード」に記載されています。
ウィザード・マップ	サービス・エミュレーション・レポート・ウィザードには，次のページが含まれています。 ［ようこそ］ > ［**列の選択**］ページ > ［フィルタ］ページ > ［並べ替え］ページ

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
All fields	レポートで使用できるすべてのフィールド。 ➤ クライアント IP アドレス ➤ 日時 ➤ 操作 ➤ ルール ➤ サービス ➤ セッション ID
Fields to be displayed	レポートに表示される順に並べられたフィールド。 ➤ レポートのカラムの順序を変更するには，[**Fields to be displayed**] 表示枠で [上へ] および [下へ] ボタンを使用します。 ➤ レポートのフィールドを追加または削除するには，横向きの矢印ボタンを使用します。 ➤ >: 選択したフィールドがレポートに追加されます。 ➤ >>: すべてのフィールドがレポートに追加されます。 ➤ <: 選択したフィールドがレポートから削除されます。 ➤ <<: すべてのフィールドがレポートから削除されます。

[フィルタ] ページ

このウィザード・ページでは，レポートに含めるカラムを選択できます。

重要情報	このウィザードに関する一般情報は，1159 ページの「サービス・エミュレーション・レポート・ウィザード」に記載されています。
ウィザード・マップ	サービス・エミュレーション・レポート・ウィザードには，次のページが含まれています。 [ようこそ] > [列の選択] ページ > [**フィルタ**] ページ > [並べ替え] ページ

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
開始	時間またはラベルを使用したレポート生成の下限。詳細については，ラベルを参照してください。 **ヒント：サービス呼び出しログの最初からすべてのデータを含めるには，オプションを無効にします。**
終了	時間またはラベルを使用したレポート生成の上限。詳細については，ラベルを参照してください。 **ヒント：サービス呼び出しログの最後まですべてのデータを含めるには，オプションを無効にします。**
サービス	レポートに含めるサービス。各サービスを個別に選択します。すべて表示するには，このオプションを無効にします。
操作	レポートに含める操作。各操作を個別に選択します。すべて表示するには，このオプションを無効にします。
チェックポイント ステータス	表示されるステータスは次のとおりです。「**Failure**」，「**成功**」，「**未定義**」。すべて表示するには，このオプションを無効にします。
クライアント IP アドレス	レポートに含めるクライアント IP アドレス。すべて表示するには，このオプションを無効にします。
SessionID	レポートに含めるセッション ID。すべて表示するには，このオプションを無効にします。

［並べ替え］ページ

このウィザード・ページでは，レポートに含めるカラムを選択できます。

重要情報	このウィザードに関する一般情報は，1159 ページの「サービス・エミュレーション・レポート・ウィザード」に記載されています。
ウィザード・マップ	サービス・エミュレーション・レポート・ウィザードには，次のページが含まれています。 ［ようこそ］>［列の選択］ページ >［フィルタ］ページ >［**並べ替え**］ページ

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
Asc/Desc	レポートにデータを表示する順序。並べ替え順序は，グローバルではなく，項目ごとに設定できます。 **標準設定：項目ごとに昇順。**
All fields	レポートの並べ替えができるすべてのフィールド。[日時]，[サービス]，[操作]，[クライアント IP アドレス]，[セッション ID]。
Ordering fields	レポート・データの並べ替え基準となるフィールド。並べ替えリストに項目を含めるには，右向きの矢印を使用します。優先順位を設定するには，上向きまたは下向きの矢印を使用します。

[サービス呼び出しログのクリア] ダイアログ・ボックス

このダイアログ・ボックスでは，データベースのサービス・コール呼び出しログをクリアできます。日付またはラベルに基づいて，ログ全体または一部をクリアできます。

利用方法	[クライアントのテスト] > [Clear Service Log] >
関連タスク	1147 ページの「クライアントのテスト・ツールを設定する（任意）」

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
開始	時間またはラベルを使用したレポート生成の下限。詳細については，ラベルを参照してください。 **ヒント：サービス呼び出しログの最初から［*終了*］しきい値までのすべてのデータをクリアするには，このオプションを無効にします。**
終了	時間またはラベルを使用したレポート生成の上限。詳細については，ラベルを参照してください。 **ヒント：サービス呼び出しログの［*開始*］しきい値からログの最後までのすべてのデータをクリアするには，このオプションを無効にします。**

［新規ラベル］ダイアログ・ボックス

このダイアログ・ボックスでは，受信するエミュレーション・サービス呼び出しに日時およびタイム・スタンプを作成できます。これらのラベルを使用して，レポートの生成時にデータをフィルタリングできます。

利用方法	**［クライアントのテスト］>［ラベルの追加］**
関連タスク	1147 ページの「クライアントのテスト・ツールを設定する（任意）」
関連項目	➤ 1159 ページの「サービス・エミュレーション・レポート・ウィザード」 ➤ 1162 ページの「［サービス呼び出しログのクリア］ダイアログ・ボックス」

ユーザ・インタフェース要素の説明は次のとおりです。

UI 要素	説明
現在時刻	ラベルに関連付けられたタイム・スタンプ（読み取り専用）。
ラベル名	レポートおよびフィルタ・オプションに表示されるラベル名。

トラブルシューティングと制限事項

このセクションでは，エミュレーション・サービスおよびそのデータベースのトラブルシューティングについて説明します。

ポート番号の変更

標準のポートが使用できない場合は，設定ファイルのポート番号を変更して，エミュレーション・サービスのポートを変更できます。設定ファイル **httpd.conf** は，Service Test の **apache/conf** ディレクトリに格納されています。

ポートを番号を変更するには，次の手順で行います。

1 テキスト・エディタで **httpd.conf** ファイルを開きます。

2 エントリ "Listen 80" を目的のポート番号に変更します（例："Listen 8080. "）。

3 "ServerName localhost:80" を含むエントリを変更して，目的のポートを指定します（例："ServerName localhost:8080."）。

4 Tomcat サーバを再起動します。［**スタート**］>［**すべてのプログラム**］>［**HP Service Test**］>［**Start Emulation Server**］を選択します。

コンピュータでサービス・エミュレーション・サービスを開けない

サービス・エミュレーション・コンソールで，手作業で起動した後でもサーバにアクセスできないと表示された場合は，次のトラブルシューティングのヒントを試してください。

- サーバが起動されていることを確認します。ブラウザに次の URL を入力します。http://localhost:8080/ServiceEmulation/index.jsp。サーバが停止している場合は，［スタート］メニューから手作業で起動します。
- ポート 8080 が使用できることを確認します。使用できない場合は，ポートを解放し，サーバを再起動します。または，Service Test のインストール先の **apache-tomcat-5.5.17¥conf** フォルダにある **Server.xml** ファイルを編集します。ポートを 8080 から別のポートに変更します。新しいポート番号で新しいローカルホスト・サーバを作成します。
- <product install dir>¥Service Test¥apache-tomcat-5.5.17¥logs ディレクトリにある Apache Tomcat ログ・ファイルを開き，サーバがロードされなかった原因を判断します。問題を解決し，サーバを再ロードします。

- サービスをエミュレートするときに，エミュレーション サービスのエンドポイントの場所で「サービスはアクティブ化されていません」という警告が表示される場合は，次の操作を実行できます。
  - サービスがアクティブであることを確認します。左側の表示枠でサービスを選択し，右側の表示枠に［サービスをアクティブ化］ボタンが表示されている場合は，このボタンをクリックします。
  - サービスがすでにアクティブである場合は，URL を確認します。エミュレーション サービスの右側の表示枠の［WSDL の位置］から URL をコピーし，ブラウザに貼り付け，文字列からサフィックス「**?wsdl**」を削除します。たとえば，http://localhost:8080/axis/services/MyService?wsdl の代わりに，http://localhost:8080/axis/services/MyService を使用します。ブラウザで有効なページが開いた場合，サービスはアクティブです。このエミュレーション・サービスを使用するには，Service Test に元の WSDL をインポートします。そのアドレスを上書きし，ブラウザで直前に使用した URL に設定します。

エミュレーション・サービスのテストへの統合の詳細については，1147 ページの「エミュレーション・サービスを統合する」を参照してください。

データベース・サーバ接続に関する問題

サービス・エミュレーションでは，Service Test と一緒にインストールされた MS SQL データベースにデータが格納されます。

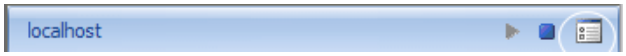
データベースに接続できない場合（たとえば，エミュレーション・サービスをアクティブ化しようとして，リモート・サーバに接続できないとコンソールに表示された場合），次の手順に従います。

サーバ接続をトラブルシューティングするには，次の手順で行います。

1. SQL Server 2005 がサーバ・ホストにインストールされていることを確認します。このインストールは必須で，通常はセットアップ時に含まれます。
2. ［**スタート**］メニューから，［**Microsoft SQL Server 2005**］>［**構成ツール**］>［**SQL Server 構成マネージャー**］を選択します。［**SQL Server 2005 のサービス**］ノードをクリックし，SQL Server のインスタンス（例：**SQLEXPRESS**）があることを確認します。
3. ［**SQL Server 2005 ネットワークの構成**］ノードを展開し，［**TCP/IP**］が実行中のインスタンスに対して［**有効**］になっていることを確認します。有効になっていない場合は，右クリック・メニューを使用して有効にします。

4 ［**TCP/IP**］をダブルクリックしてプロパティを表示します。［**IP アドレス**］タブをクリックして，インスタンスが構成されているポートを書き留めます。ポートが指定されていない場合は，ポート番号を設定し，SQL Server インスタンスを再起動します。

5 サービス・エミュレーション・コンソールを開き，ホスト（例：**localhost**）のツールバーの右上にある［**ホスト設定**］ボタンをクリックします。詳細については，1157 ページの「［ホスト設定］ダイアログ・ボックス」を参照してください。



6 実行中の SQL Server インスタンスと TCP/IP ポート番号に一致するように，データベース接続を編集します。

セットアップでは，標準設定のインスタンスとして **SQLEXPRESS**，標準設定のポートとして **1433** が使用されます。構成マネージャーの情報が異なる場合は，それに従ってダイアログ・ボックスの値を編集します。［**サーバ**］ボックスに，**< ホスト >¥< インスタンス >** の組み合わせを入力します。名前が指定されていないインスタンスが実行されている場合は，［**サーバ**］ボックスに「**localhost**」と入力します。

40

Windows Sockets プロトコル

本章の内容

概念

タスク

リファレンス

概念

Windows Sockets の記録の概要

Windows Sockets プロトコルでは，Microsoft WinSock DLL を使用して TCP/IP プロトコルで通信するアプリケーションがサポートされます。WinSock プロトコルでは，バッファから送受信される実際のデータを確認できます。

WinSock プロトコルでは，ソケット，データ・バッファ，および Windows Sockets 環境に関連する関数が記録されます。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。

スクリプトを作成したら，記録したバッファを未処理のデータまたはスナップショットとして表示できます。詳細については，1170 ページの「Windows Sockets データ」または 1172 ページの「Windows Sockets データ・バッファのスナップショット」を参照してください。

このセクションには次の内容も含まれます。

変換テーブル

Windows Sockets データは，変換テーブルを使用して EBCDIC 形式で表示できます。

変換テーブルを使用すると，WinSock シングル・プロトコルを使用する場合に記録形式を指定でき，WinSock マルチ・プロトコルを使用する場合にコード生成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。データが ASCII 形式の場合，変換の必要はありません。

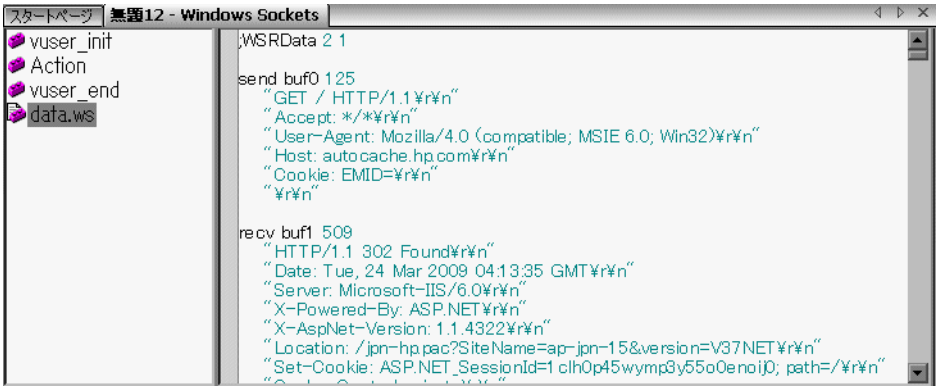
リスト・ボックスの項目の最初の 4 桁はサーバの形式を表します。後半の 4 桁はクライアントの形式を表します。上の例で選択された変換テーブルは 002501b5 です。サーバの形式が 0025，クライアントの形式が 01b5 であり，サーバからクライアントへの送信を表しています。クライアントからサーバへの送信では，これらの形式を逆転した項目 01b50025 を選択します。これは，クライアントの 01b5 形式をサーバの 0025 形式に変換する必要があることを示します。

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。

記録オプションでの変換テーブルの選択方法の詳細については，416 ページの「[WinSock] ノード」を参照してください。

Windows Sockets データ

VuGen を使用して Windows Sockets 仮想ユーザ・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。仮想ユーザ・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 のスクリプト・ビューを使用して，スクリプト・エディタ内にデータ・ファイルの内容を表示できます。



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。仮想ユーザ・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番号が付けられます（1，2，3...）。

次に示す例では，**lrs_receive** 関数が仮想ユーザ・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。

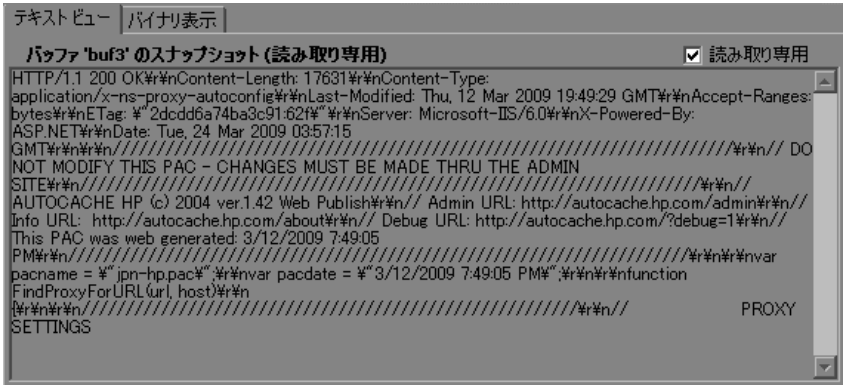
```
recv buf4 39
    "¥xff¥xfb¥x01¥xff¥xfb¥x03¥xff¥xfd¥x01"
    "¥r¥n"
    "¥r¥n"
    "SunOS UNIX (sunny)¥r¥n"
    "¥r"
    "¥x0"
    "¥r¥n"
    "¥r"
    "¥x0"
```

タスクの詳細については，1182 ページの「Windows Sockets バッファを表示および変更する方法」を参照してください。

Windows Sockets データ・バッファのスナップショット

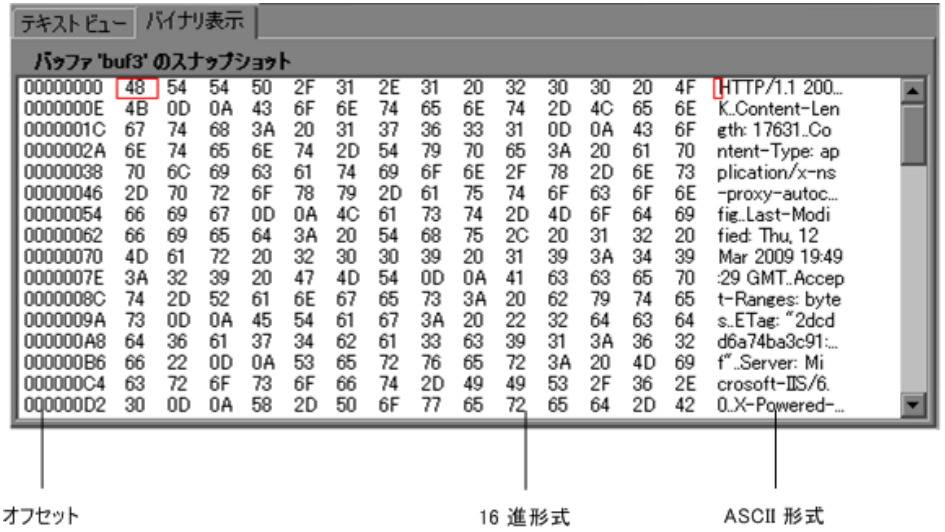
ツリー・ビューに Windows Sockets スクリプトを表示すると，編集が可能な［バッファのスナップショット］ウィンドウにデータが表示されます。スナップショットを［**テキスト ビュー**］または［**バイナリ表示**］に表示できます。

［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。



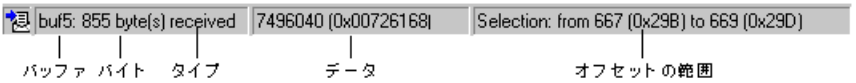
標準設定では，バッファ・データは読み取り専用として保存されます。バッファの内容を変更する場合は，バッファのテキスト • ビューで［**読み取り専用**］ボックスをクリアします。VuGen から，ブックマークとパラメータに影響がある可能性があるという警告が発行されます。

［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位置（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。

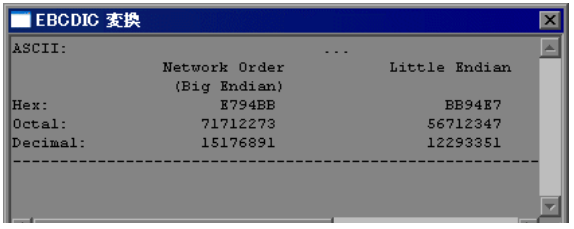
データを選択している場合は，ステータス・バーにも，データに関する情報がビッグ・エンディアン形式およびリトル・エンディアン形式で表示されます。

データ・ビューア

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。

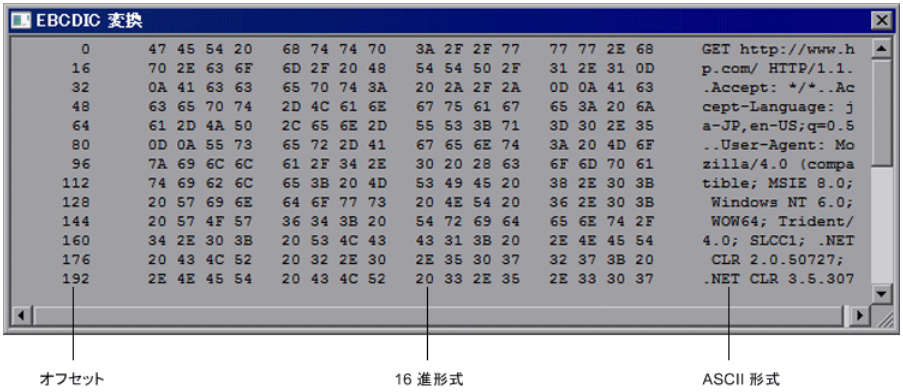
ビューア・ウィンドウでデータを表示するには，データを選択して F7 キーを押します。

➤ 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。



➤ 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。

長い形式では，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。

短い形式および長い形式のカスタマイズ方法の詳細については，1188 ページの「表示形式」を参照してください。

データ・ナビゲーション・ツール

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・ナビゲーション

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］では，受信および送信バッファ・ステップ（**lrs_send**，**lrs_receive**，**lrs_receive_ex**，および **lrs_length_receive**）だけを表示できます。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。

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。

場所に移動

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。

ブックマーク設定

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。

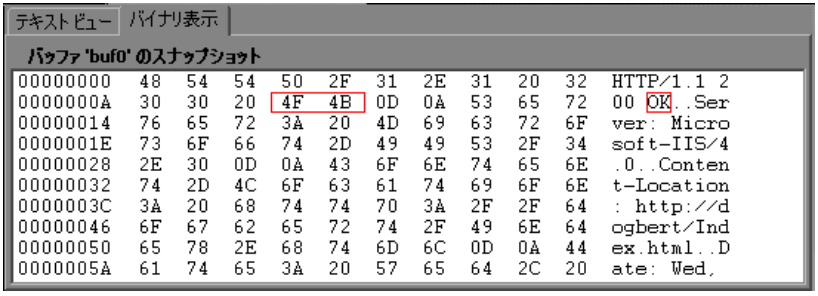
ブックマークは 1 バイトにも複数バイトにも付けられます。リスト中のブックマークをクリックすると，対応する場所が選択された状態で［バッファのスナップショット］ウィンドウに表示されます。初期設定では，そのデータがテキスト・ビューでは青で強調表示され，バイナリ表示ではブックマーク・ブロックが赤で表示されます。また，バイナリ表示でブックマークにカーソルを置くと，ポップアップ・テキスト・ボックスが開いてブックマークの名前が表示されます。

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。

バッファ・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。

タスクの詳細については，1185 ページの「データのブロックをコピーして貼り付ける（任意）」を参照してください。

タスク

Windows Sockets スクリプトを記録する方法

このタスクでは，Windows Sockets 記録の設定方法およびセッションの記録方法について説明します。

このタスクでは，次の手順を実行します。

- ➤ 1180 ページの「記録オプションを開く（任意）」
- ➤ 1180 ページの「変換テーブルを選択する（任意）」
- ➤ 1181 ページの「関連のないすべてのソケットを除外する（任意）」
- ➤ 1181 ページの「思考遅延時間のしきい値を設定する（任意）」
- ➤ 1181 ページの「セッションを記録する」
- ➤ 1181 ページの「スクリプトをパラメータ化する（任意）」
- ➤ 1181 ページの「スクリプトを再生成する（任意）」

1 記録オプションを開く（任意）

WinSock スクリプトを作成した後に，［**ツール**］>［**記録オプション**］を選択し，［**WinSock**］ノードをクリックします。

2 変換テーブルを選択する（任意）

EBCDIC セクションで，変換テーブルを選択します。データが ASCII 形式の場合は［**なし**］オプションを選択します。このオプションを選択しない場合は，VuGen によって ASCII データが変換されます。詳細については，1169 ページの「変換テーブル」を参照してください。

3 関連のないすべてのソケットを除外する（任意）

［**除外する設定**］セクションで，関連のないすべてのソケットをリストに追加します。ローカル・ホストや DNS ポート（53）のような，テスト下のサーバの負荷に影響しないホストとポートを除外する必要があります。これらは標準設定では除外されています。

エントリを記録から除外し，それらのエントリをログに含めるには，［**除外したソケットをログに含めない**］オプションをクリアします。

ユーザ・インタフェースの詳細については，416 ページの「[WinSock] ノード」を参照してください。

4 思考遅延時間のしきい値を設定する（任意）

思考遅延時間のしきい値を示します。操作の途中の一時停止が VuGen によって検出され，その一時停止がしきい値の時間よりも短い場合，**思考遅延ステップ / lr_think_time** 関数は生成されません。詳細については，416 ページの「[WinSock] ノード」を参照してください。

5 セッションを記録する

セッションを記録して，スクリプトを保存します。

6 スクリプトをパラメータ化する（任意）

ショートカット・メニューを使用し，記録された値をパラメータで置き換えます。詳細については，第 9 章，「パラメータ」。を参照してください。

7 スクリプトを再生成する（任意）

スクリプトの再生成が必要な場合，たとえば，除外したホスト：ポートを含める場合や変換が正しくなかった場合は，［**ツール**］>［**再生成**］を選択します。WinSock ノードで，設定を変更します。

注：スクリプトの再生成は，マルチ・プロトコルのスクリプトでのみサポートされます。

Windows Sockets バッファを表示および変更する方法

次の手順では，WinSock データを表示，変更，ナビゲートする方法について説明します。

➤ 1182 ページの「スクリプト・ビューでデータの表示と変更を行う（任意）」

➤ 1182 ページの「ビューアでバッファを開く（任意）」

➤ 1183 ページの「ツリー・ビューでデータの表示と変更を行う（任意）」

➤ 1183 ページの「データ内をナビゲートする（任意）」

➤ 1184 ページの「ブックマークの作成とナビゲーションを行う（任意）」

➤ 1185 ページの「データをバッファに挿入する（任意）」

➤ 1185 ページの「データのブロックをコピーして貼り付ける（任意）」

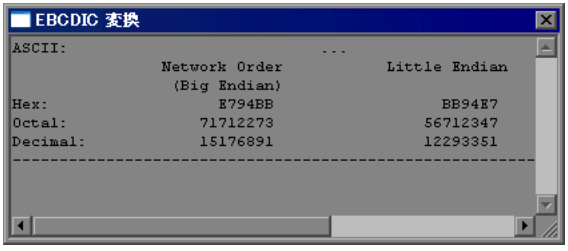
スクリプト・ビューでデータの表示と変更を行う（任意）

スクリプト・ビューの左の表示枠で，**data.ws** ファイルを選択します。必要に応じて，VuGen のエディタ・ウィンドウでデータを直接変更します。詳細については，1170 ページの「Windows Sockets データ」を参照してください。

ビューアでバッファを開く（任意）

データを EBCDIC 変換された形式で表示するには，スクリプト・ビューでデータを選択して 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。



➤ 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。

詳細については，1174 ページの「データ・ビューア」を参照してください。

ツリー・ビューでデータの表示と変更を行う（任意）

ツリー・ビューでは，右の表示枠にデータのスナップショットが表示されます。データを編集するには，バッファの**テキスト・ビュー**で［**読み取り専用**］オプションをクリアします。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。詳細については，1172 ページの「Windows Sockets データ・バッファのスナップショット」を参照してください。

データ内をナビゲートする（任意）

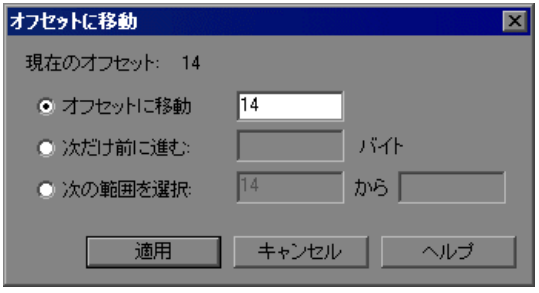
バッファ・データ内をナビゲートするには，［**バッファ ナビゲータ**］または［**オフセットに移動**］ダイアログ・ボックスを使用します。

バッファ・ナビゲータ

ツリー・ビューに入ります。［**表示**］>［**バッファ ナビゲータ**］を選択します。必要に応じて，フィルタを選択します。たとえば，受信バッファを選択します。データを表示するバッファを選択します。必要に応じて，スナップショットのテキスト・ビューでデータを変更します。詳細については，1176 ページの「バッファ・ナビゲーション」を参照してください。

オフセットに移動

スナップショット・ウィンドウ内をクリックします。ショートカット・メニューから［**オフセットに移動**］を選択します。



➤ 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**］を選択して，オフセット値を指定します。［**適用**］をクリックします。

- ➤ 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。［**適用**］をクリックします。
- ➤ 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。［**適用**］をクリックします。

詳細については，1177 ページの「場所に移動」を参照してください。

ブックマークの作成とナビゲーションを行う（任意）

ブックマークを使用すると，データ・バッファ内の特定の場所を記憶できます。詳細については，1177 ページの「ブックマーク設定」を参照してください。

1. ブックマークを作成するには，ツリー・ビューを開き，バッファのスナップショット内の 1 つ以上のバイトを選択します（テキスト・ビューまたはバイナリ表示)。ショートカット・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，ショートカット・メニュー内の［**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。

データをバッファに挿入する（任意）

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，次の手順で行います。

1 バッファのスナップショット内をクリックします（テキスト・ビューまたはバイナリ表示）。

2 ショートカット・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。

データのブロックをコピーして貼り付ける（任意）

データは，文字，10 進数，または 16 進数として変更できます。詳細については，1178 ページの「バッファ・データの編集」を参照してください。

1 スナップショットのバイナリ表示を開きます。

2 バッファ・データをコピーするには，次の手順で行います。

- 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，ショートカット・メニューで［**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，ショートカット・メニューで［**詳細設定**］>［**数値としてコピー**］>［**16 進法**］を選択します。

3 データを貼り付けるには，次の手順で行います。

- 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，ショートカット・メニューで［**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，ショートカット・メニューで［**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。

4 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択し，ショートカット・メニューから［**削除**］を選択します。

リファレンス

データ・バッファ

データ・ファイル **data.ws** の形式は次のとおりです。

- 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，416 ページの「[WinSock] ノード」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"
```

表示形式

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。**lrun/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ーです。
- **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
¥¥	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。

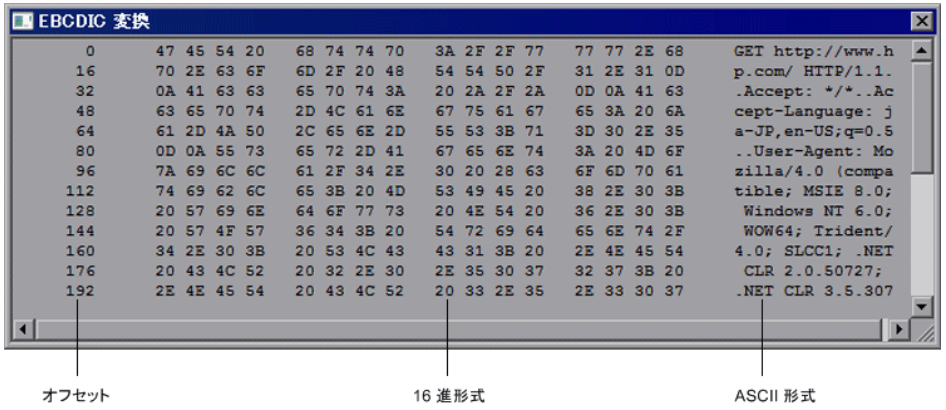
%a	ASCII 表記
%BX	ビッグ・エンディアン（ネットワーク順序）16 進数
%BO	ビッグ・エンディアン（ネットワーク順序）8 進数
%BD	ビッグ・エンディアン（ネットワーク順序）10 進数
%LX	リトル・エンディアン 16 進数
%LO	リトル・エンディアン 8 進数
%LD	リトル・エンディアン 10 進数

- **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ー。
- **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。

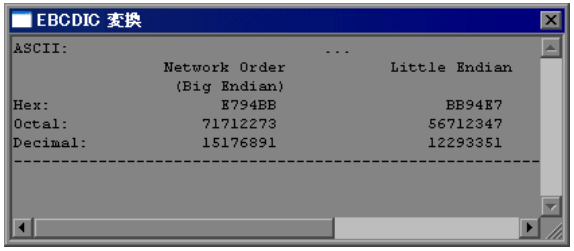
標準設定では，長い形式と短い形式が設定され，非印字文字はドットとして出力するように指定されています。

```
[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。



Windows Sockets ユーザ・インタフェース

このセクションの内容

➤［データ ナビゲータ］ダイアログ・ボックス（1192 ページ）

➤［オフセットに移動］ダイアログ・ボックス（1192 ページ）

➤［ブックマーク］ダイアログ・ボックス（1193 ページ）

［データ ナビゲータ］ダイアログ・ボックス

このダイアログ・ボックスには，送信と受信のバッファ・ステップが表示されます。［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。

利用方法	**［表示］>［バッファ ナビゲータ］**
重要情報	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。
関連タスク	1183 ページの「データ内をナビゲートする（任意）」

ユーザ・インタフェース要素の説明は次のとおりです（ラベルのない要素は山括弧で囲んで示します）。

UI 要素	説明
<バッファ・リスト>	記録中に生成されたすべての送信バッファと受信バッファのリスト。
フィルタ	［バッファのスナップショット］ウィンドウに表示するデータをフィルタ処理します。ドロップダウン・リストには次のオプションが含まれます。 ➤ すべてのバッファ ➤ 受信バッファ ➤ 送信バッファ

［オフセットに移動］ダイアログ・ボックス

このダイアログ・ボックスを使用すると，記録されたデータ内の特定の場所に移動できます。

利用方法	スナップショット・ウィンドウ内をクリックし，ショートカット・メニューから［**オフセットに移動**］を選択します。
関連タスク	1183 ページの「データ内をナビゲートする（任意）」

ユーザ・インタフェース要素の説明は次のとおりです（ラベルのない要素は山括弧で囲んで示します）。

UI 要素	説明
適用	指定したオフセットにカーソルを移動します。
現在のオフセット	カーソルの現在のオフセット（読み取り専用）
オフセットに移動	データ内の特定の絶対オフセットに移動します。
Advance by…bytes	カーソルを基準にした相対位置にバイト数単位でジャンプします。正の値は前方方向を示します。負の値は逆方向を示します。
Select range from...to...	バッファ内のデータの範囲を選択します。

［ブックマーク］ダイアログ・ボックス

このダイアログ・ボックスでは，ブックマークの設定とブックマークへのナビゲーションができます。

利用方法	ブックマーク・リストを表示するには，［**表示**］>［**出カウィンドウ**］を選択し，［**ブックマーク**］タブを選択します。
関連タスク	1184 ページの「ブックマークの作成とナビゲーションを行う（任意）」

UI 要素	説明
移動(G)	選択したブックマークに移動します。
新規作成(N)	カーソルの現在の位置で新しいブックマークを作成します。
削除(D)	選択したブックマークを削除します。
変更(M)	選択したブックマークを，データ・バッファ内のカーソルの現在の位置に変更します。

UI 要素	説明
< ブックマーク・リスト >	すべてのブックマークのリストおよび各ブックマークの情報。 ➤ **[名前]**：ブックマークに指定された名前。 ➤ **[アクション]**：ブックマークを含むスクリプト・セクション ➤ **[バッファ]**：ブックマークのバッファ名。 ➤ **[位置]**：バッファ内のブックマークの範囲。 ➤ **[説明]**：ブックマークの種類。**永久**または**標準**。 **ヒント**： ➤ エントリをダブルクリックすると，カーソルはブックマークされたデータに移動します。 ➤ カラムをクリックすると，そのカラムの情報で並べ替えられます。
フィルタ	次の項目でブックマーク・リストをフィルタ処理します。 ➤ すべてのバッファ ➤ 受信バッファ ➤ 送信バッファ ➤ 選択したバッファのみ
永久ブックマークのみ表示する	リストに永久ブックマークのみを表示します。標準ブックマークは表示しません。

41

ワイヤレス・プロトコル

本章の内容

概念

- WAP プロトコルの概要（1196 ページ）
- WAP ツールキット（1197 ページ）
- プッシュおよびプル技術（1198 ページ）
- VuGen でのプッシュのサポート（1199 ページ）
- MMS（マルチメディア・メッセージング・サービス）プロトコルの概要（1201 ページ）

タスク

- Controller で MMS シナリオを実行する方法（1202 ページ）

概念

WAP プロトコルの概要

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。

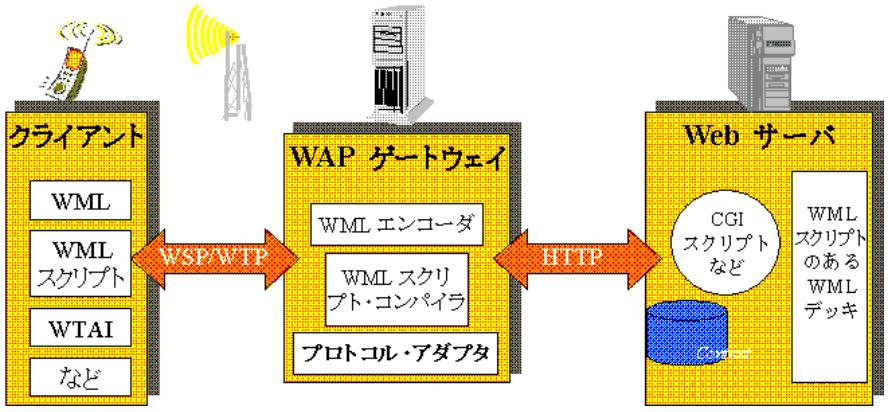
WAP ではさらに，次の条件を満たすプロキシ・サーバを規定しています。

- 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 仮想ユーザ固有のアクションをエミュレートします。

プッシュおよびプル技術

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 仮想ユーザ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，仮想ユーザにプッシュ・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，仮想ユーザ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，仮想ユーザはメッセージを取得しません。

プッシュ・イニシエータのエミュレート

WAP HTTP 仮想ユーザ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，仮想ユーザは，**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** でサポートされます。

プッシュ型メッセージのフォーマット

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，*オンライン関数リファレンス*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）プロトコルの概要

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 仮想ユーザ・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 仮想ユーザを使用することで，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，*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。

タスク

Controller で MMS シナリオを実行する方法

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，次の手順で行います。

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。［グループ情報］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。

第 III 部

上級ユーザのために

42

VuGen エディタを使用する，手動によるスクリプトのプログラミング

本章の内容

概念

概念

手動によるスクリプトのプログラミング - 概要

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。仮想ユーザ API または標準のプログラミング関数を使用できます。仮想ユーザ API 関数では，仮想ユーザについての情報を収集できます。たとえば，仮想ユーザ関数を使って，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，テストに含まれる仮想ユーザまたは監視対象の仮想ユーザ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仮想ユーザ・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仮想ユーザ・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仮想ユーザ API 関数のライブラリをインポートして仮想ユーザ・スクリプトを作成します。詳細については，1215 ページの「Visual Studio によるスクリプトの作成」を参照してください。

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，手動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 仮想ユーザ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 仮想ユーザ・スクリプト

C 仮想ユーザ・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 仮想ユーザ・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 仮想ユーザ］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仮想ユーザ・スクリプトであれば，どのスクリプトでも C 仮想ユーザ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*オンライン関数リファレンス*（［**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 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 仮想ユーザ・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。

仮想ユーザ・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 言語関数を仮想ユーザ・スクリプトに追加するときは，次の制限に注意してください。

- 仮想ユーザ・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** 関数は仮想ユーザ・スクリプトではサポートされていません。
- 仮想ユーザ・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。
- 仮想ユーザ・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。例：
  extern char * strtok();

libc 関数の呼び出し

仮想ユーザ・スクリプトで，**libc** 関数を呼び出すことができます。ただし，仮想ユーザ・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

仮想ユーザ・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。*/
```

JavaScript 仮想ユーザ

空の JavaScript 仮想ユーザ・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 仮想ユーザ・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 仮想ユーザ］を選択します。

```
function Actions()
{
    //"TO DO: ビジネス・プロセス / アクション・コードをここに配置する



    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，仮想ユーザ 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")
```

VBScript 仮想ユーザ

空の VBScript 仮想ユーザ・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 仮想ユーザ・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 仮想ユーザ］を選択します。VuGen によって，空の VBScript 仮想ユーザ・スクリプトが作成されます。

```
Public Function Actions()

    '"TO DO : アクション・コードをここに配置する



    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，仮想ユーザ 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")
```

Java 仮想ユーザ

Java 仮想ユーザ・スクリプトでは，標準 Java コードが使用できます。空の Java 仮想ユーザ・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 仮想ユーザ］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 仮想ユーザ・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 仮想ユーザ

空の Visual Basic 仮想ユーザ・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 仮想ユーザ・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 仮想ユーザ］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：アクション・コードをここに配置する



    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，仮想ユーザと 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 仮想ユーザ・スクリプトの再生エラー

VB 仮想ユーザ・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 Software の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。

43

Visual Studio によるスクリプトの作成

本章の内容

概念

タスク

概念

Visual Studio による仮想ユーザ・スクリプトの作成 - 概要

仮想ユーザ・スクリプトを作成するには，VuGen を使用したり Visual Studio などの開発環境を使用したりする方法があります。

VuGen	VuGen の記録機能を使用したり，VuGen エディタを使用して手動でプログラミングしたりすることにより，Windows や UNIX プラットフォームで実行する仮想ユーザ・スクリプトを作成できます。Windows 環境で作成したスクリプトは Windows と UNIX 環境の両方で実行できます。記録は UNIX 環境で行うことはできません。
Visual Studio	Visual Studio を使えば，仮想ユーザ・スクリプトを Visual Basic，C，C++ でプログラミングすることができます。プログラムはダイナミック・リンク・ライブラリ（dll）としてコンパイルします。

本章では，Visual C と Visual Basic の開発環境でプログラミングによって仮想ユーザ・スクリプトを作成する方法について説明します。これらの環境では，開発アプリケーションに仮想ユーザ API のライブラリをインポートして仮想ユーザ・スクリプトを作成します。

また，VuGen エディタ上でも，アプリケーションのライブラリやクラスを組み込んで仮想ユーザ・スクリプトのプログラミングをすることもできます。VuGen では，C，Java，Visual Basic，VBScript または JavaScript のプログラミングが行えます。詳細については，第 42 章，「VuGen エディタを使用する，手動によるスクリプトのプログラミング」。を参照してください。

プログラミングで仮想ユーザ・スクリプトを作成する場合，VuGen テンプレートを大規模な仮想ユーザ・スクリプトの原形として使用できます。テンプレートで提供するものは，次のとおりです。

- 正しいプログラム構造
- 仮想ユーザ API 呼び出し
- ダイナミック・リンク・ライブラリを作成するためのソース・コードとメイクファイル

仮想ユーザ・スクリプトで使用可能なオンラインの C 言語の共通関数リファレンスについては，*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。

タスク

Visual C を使用して仮想ユーザ・スクリプトを作成する方法

仮想ユーザ・スクリプトを作成するには，Visual C のバージョン 6.0 以上を使用します。

1 Visual C で，ダイナミック・リンク・ライブラリ（dll）を作成する新規プロジェクトを開きます。［**ファイル**］>［**新規作成**］を選択し，［プロジェクト］タブをクリックします。

2 ［ウィザード］で［**空の DLL プロジェクト**］を選択します。

3 プロジェクトに次のファイルを追加します。

- 新規 *cpp* ファイルに，*init*，*run*，*end* の 3 つの関数をエクスポートしたもの（関数名は変更できます）。
- ライブラリ・ファイル lrun50.lib（<LoadRunner のインストール・ディレクトリ >¥lib にあります）。

4 プロジェクトの設定で次の変更を行います。

- ［C/C++］タブを選択して，［**コード生成**（カテゴリ）］>［**使用するランタイム ライブラリ**（リスト）］を選択します。これを［**マルチスレッド**（DLL）］に変更します。
- ［C/C++］タブを選択して，［**プリプロセッサ**（カテゴリ）］>［**プリプロセッサの定義**（編集フィールド）］を選択し，_DEBUG を削除します。

5 これで，クライアント・アプリケーションからコードを追加したり，通常どおりプログラミングします。

6 仮想ユーザ API 関数を使ってスクリプトを拡張します。たとえば，メッセージを発行するには **lr_output_message**，トランザクションの開始を示すには **lr_start_transaction** を使用します。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**］）の「一般関数」を参照してください。

7 プロジェクトをビルドします。DLL として出力されます。

8 DLL と同じ名前のディレクトリを作成し，作成した DLL をこのディレクトリにコピーします。

9 *Template* ディレクトリの **lrvuser.usr** ファイルを開き，USR ファイル・キー *BinVuser* を次のようにその DLL 名で上書きします：BinVuser=<*DLL_name*>

例：

次の例では，**lr_output_messsage** 関数で，どのセクションが実行されているかを示すメッセージを発行します。**lr_eval_string** 関数は，ユーザ名を取得します。次の例を使用する場合は，仮想ユーザ API のインクルード・ファイル **lrun.h** へのパスが正しいことを確認してください。

```
#include "c:¥lrun_5¥include¥lrun.h"

extern "C" {
int __declspec(dllexport) Init (void *p)
{
    lr_output_message("in init");
return 0;
}

int __declspec(dllexport) Run (void *p)
{
    const char *str = lr_eval_string("<name>");
    lr_output_message("in run and parameter is %s", str);
return 0;
}

int __declspec(dllexport) End (void *p)
{
    lr_output_message("in end");
return 0;
}
} //extern C end
```

Visual Basic を使用して仮想ユーザ・スクリプトを作成する方法

Visual Basic を使用して仮想ユーザを作成するには，次の手順で行います。

1 Microsoft Visual Basic で，［**ファイル**］>［**新しいプロジェクト**］>［**LoadRunner Virtual User**］を選択し，新しいプロジェクトを作成します。新しいプロジェクトが 1 つのクラスと 仮想ユーザ用のテンプレートで作成されます。

2 ［**ファイル**］>［**プロジェクトの上書き保存**］を選択して，プロジェクトを保存します。

3 オブジェクト・ブラウザ（[表示] メニュー）を開きます。「LoadRunner 仮想ユーザ」ライブラリを選択し，仮想ユーザ・クラス・モジュールをダブルクリックしてテンプレートを開きます。テンプレートには，**Vuser_Init**，**Vuser_Run**，**Vuser_End** の 3 つのセクションが含まれています。

```
Option Explicit

Implements Vuser

Private Sub Vuser_Init()
' ここで 仮想ユーザの初期化コードを実装する
End Sub

Private Sub Vuser_Run()
' ここで 仮想ユーザの主なアクションを示すコードを実装する
End Sub

Private Sub Vuser_End()
' ここで 仮想ユーザの終了コードを実装する
End Sub
```

4 これで，クライアント・アプリケーションからコードを追加したり，通常どおりプログラミングします。

5 オブジェクト・ブラウザを使用して，トランザクション，思考遅延時間，ランデブー・ポイント，メッセージなど，必要な VuGen 要素をコードに追加します。

6 実行環境の設定とパラメータで，プログラムを拡張します。詳細については，1221 ページの「実行環境設定とパラメータを設定する方法」を参照してください。

7 [**ファイル**] > [*project_name*.dll の作成] を選択して，仮想ユーザ・スクリプトをビルドします。

プロジェクトは，プロジェクトと同じディレクトリに仮想ユーザ・スクリプト形式（.usr）で保存されます。

実行環境設定とパラメータを設定する方法

スクリプト用に DLL を作成したら，スクリプト・ファイル（*.usr*）を作成して，その設定を行います。VuGen で提供される *lrbin.bat* ユーティリティを使って，パラメータを定義し，Visual C や Visual Basic を使って作成したスクリプトの実行環境の設定を行います。このユーティリティは，製品のインストール・ディレクトリにある *bin* ディレクトリにあります。

実行環境の設定を行い，スクリプトをパラメータ化するには，次の手順で行います。

1 製品の *bin* ディレクトリにある *lrbin.bat* をダブルクリックします。［Standalone Vuser Configuration］ダイアログ・ボックスが開きます。

2 ［**ファイル**］>［**新しいプロジェクト**］を選択します。*usr* ファイルとなるスクリプトの名前を指定します。このスクリプト名は，DLL を保存したディレクトリの名前と同じでなくてはなりません。

3 ［**仮想ユーザ**］>［**詳細設定**］を選択し，［詳細設定］ダイアログ・ボックスに DLL の名前を入力します。

4 ［**仮想ユーザ**］>［**実行環境の設定**］を選択して，実行環境の設定を定義します。［実行環境設定］ダイアログ・ボックスは，VuGen のインタフェースに表示されるものと同じです。詳細については，419 ページの「実行環境の設定」を参照してください。

5 **仮想ユーザ**］>［**パラメータ リスト**］を選択して，スクリプトにパラメータを定義します。［パラメータ］ダイアログ・ボックスは，VuGen のインタフェースに表示されるものと同じです。詳細については，263 ページの「パラメータ」を参照してください。

スクリプトをスタンドアロン・モードで実行して，テストします。［**仮想ユーザ**］>［**仮想ユーザの実行**］を選択します。スクリプトの実行中は，仮想ユーザの実行ウィンドウが現れます。

44

言語のサポート

本章の内容

概念

タスク

リファレンス

概念

言語のサポート - 概要

VuGen では多国語環境がサポートされており，スクリプトを作成，実行する際に自国語のマシン上で英語その他の言語を使用できます。

英語以外の言語で作業を行うときは，記録や再生時に VuGen がテキストのエンコーディングを認識していることを確認することが，主な問題になります。エンコーディングは，スクリプトで使用するすべてのテキストに適用されます。これには，Web 仮想ユーザの場合の HTTP ヘッダ内のテキストや HTML ページ，パラメータ・ファイル内のデータなどが含まれます。

Windows 2000 以降では，ANSI，Unicode，Unicode ビッグ・エンディアン，UTF-8 など，特定のエンコーディングがされたテキスト・ファイルを，メモ帳から直接保存できます。

標準設定では，VuGen はローカル・マシンのエンコーディング（ANSI）で動作します。多国語で作業を行っているサーバによっては，UTF-8 エンコーディングでの作業が必要になることがあります。このようなサーバに対して作業を行うためには，詳細記録オプションの中で，スクリプトが UTF-8 エンコーディングであるよう指定する必要があります。

ページ要求ヘッダの言語

Web スクリプトを実行する前に，現在使用している言語に合わせてページのリクエスト・ヘッダを設定しておくことができます。［実行環境の設定］の［インターネット プロトコル］で，*Accept-Language* リクエスト・ヘッダを設定します。このヘッダを通じて，許容されるすべての言語のリストがサーバに提供されます。

この値を設定するには，［**仮想ユーザ**］>［**実行環境の設定**］>［**インターネット プロトコル**］>［**プリファレンス**］>［**詳細**］>［**オプション**］>［**Accept-Language 要求ヘッダ**］を選択し，目的の言語を選択します。

ユーザ・インタフェースの詳細については，456 ページの「[インターネット プロトコル］の［プリファレンス］ノード」を参照してください。

タスク

文字列のエンコーディング形式の変換方法

lr_convert_string_encoding 関数を使用すると，次の構文を使用して文字列のエンコーディング（UTF-8，Unicode，またはロケール・マシン・エンコーディング）を手作業で変換できます。

```
lr_convert_string_encoding(char * sourceString, char * fromEncoding, char *
toEncoding, char * paramName)
```

この関数は，結果の文字列（終端の NULL を含む）を第 3 引数 *paramName* に保存します。変換に成功した場合は 0 を返し，失敗した場合は -1 を返します。

fromEncoding 引数および **toEncoding** 引数の形式は次のとおりです。

LR_ENC_SYSTEM_LOCALE	NULL
LR_ENC_UTF8	"utf-8"
LR_ENC_UNICODE	"ucs-2"

次の例では，**lr_convert_string_encoding** によって「Hello world」をシステム・ロケールから Unicode に変換しています。

```
Action()
{
    int rc = 0;
    unsigned long converted_buffer_size_unicode = 0;
    char              *converted_buffer_unicode = NULL;

    rc = lr_convert_string_encoding("Hello world", NULL, LR_ENC_UNICODE,
"stringInUnicode");
    if(rc < 0)
    {
        // エラー
    }
return 0;
}
```

再生ログでは，出力ウィンドウに次の情報が表示されます。

```
Output:
Starting action Action.
Action.c(7): Notify: Saving Parameter "stringInUnicode = H¥x00e¥x00l¥x00l¥x00o¥x00
¥x00w¥x00o¥x00r¥x00l¥x00d¥x00¥x00¥x00"
Ending action Action.
```

変換の結果は *paramName* 引数に保存されます。

パラメータ・ファイルのエンコーディング形式の変換方法

パラメータ・ファイルには，スクリプトの中で定義されたパラメータに対応するデータが収められています。このファイルはスクリプトのディレクトリに格納され，拡張子 **.dat* が付けられます。スクリプトを実行するとき，仮想ユーザはこのデータを使用して，アクションをさまざまな値で実行します。

標準設定では，パラメータ・ファイルはマシンのエンコーディングを使用して保存されます。しかし，英語以外の言語で作業を行うときは，サーバにおいて文字列を UTF-8 で受信することが前提となっている場合に，パラメータ・ファイルを UTF-8 に変換しなければならないことがあります。Windows 2000 以降で作業を行っていれば，メモ帳から直接この変換を実行できます。

UTF-8 エンコーディングをパラメータ・ファイルに適用するには，次の手順を実行します。

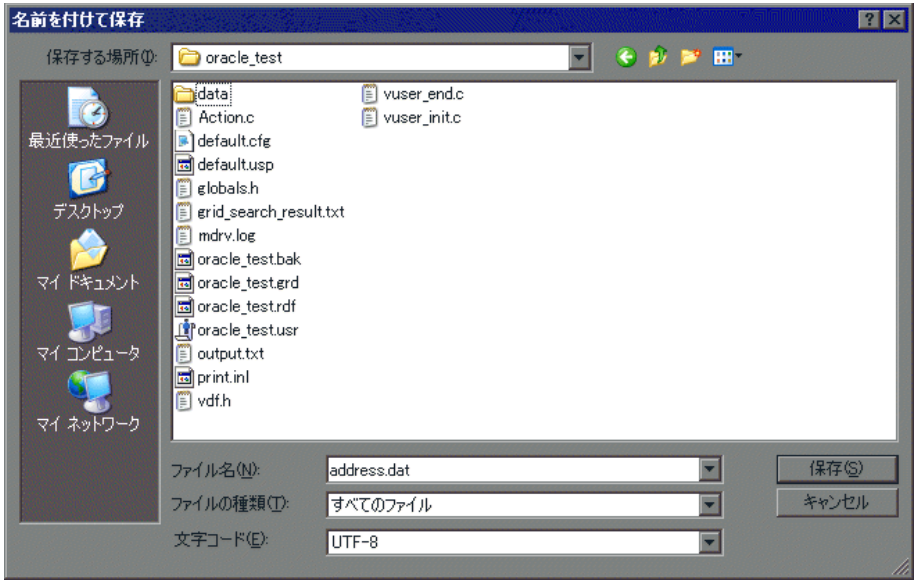
1 ［**仮想ユーザ**］>［**パラメータ リスト**］を選択し，パラメータ・プロパティを表示します。

2 右側の表示枠にある［**ファイル パス**］ボックスで，パラメータ・ファイルを探します。

3 パラメータ・テーブルを表示した状態で，［**メモ帳で編集**］をクリックします。メモ帳が開き，パラメータ・ファイルが csv 形式で表示されます。

4 ［**ファイルの種類**］ボックスで，［*すべてのファイル*］を選択します。

［**文字コード**］ボックスで，エンコーディングのタイプとして［*UTF-8*］を選択します。



5 ［**保存**］をクリックします。既存のパラメータ・ファイルを上書きしてもよいかどうかの確認を求められます。［**はい**］をクリックします。

これで，パラメータ・ファイルが VuGen によって UTF-8 テキストとして認識されるようになります。ただし，表示上は通常の文字で表示されます。

英語以外の言語の Web ページの記録方法

Web またはその他のインターネット・プロトコルを使って作業を行うときは，Web ページ・テキストの記録用のエンコーディングを指定できます。記録するサイトの言語がオペレーティング・システムの言語と一致している必要があります。1 つの記録の中で複数のエンコーディングを組み合わせることはできません。たとえば，UTF-8 を ISO-8859-1 や shift_jis と一緒に使用することはできません。

このタスクでは，英語以外の言語の Web ページを VuGen を使用して記録する方法について説明します。

このタスクでは，次の手順を実行します。

- 1228 ページの「英語以外の言語の Web ページの自動記録」
- 1229 ページの「手作業による英語以外の言語の Web ページの記録」

1 英語以外の言語の Web ページの自動記録

英語以外の言語の Web ページとして認識させるためには，ページの HTTP ヘッダまたは HTML メタ・タグの中で文字セットを指定する必要があります。そうしないと，EUC-JP エンコーディングが VuGen によって認識されず，Web サイトが正しく記録されません。英語以外の言語の要求を **EUC-JP** または **UTF-8** として記録するよう VuGen に指示するには，［**ツール**］>［**記録オプション**］>［**HTTP プロパティ**］>［**詳細**］>［**サポート対象文字セット**］を選択し，［記録オプション］ダイアログ・ボックスの［**HTTP プロパティ：詳細**］ノードで該当するオプションを選択します。ユーザ・インタフェースの詳細については，367 ページの「HTTP の［詳細］ノード」を参照してください。

［記録オプション］で［**EUC-JP**］または［**UTF-8**］を選択すると，Web ページで異なるエンコーディングが使用されている場合でも，選択したエンコーディングによって強制的に Web ページが記録されます。たとえば，非 EUC でエンコードされた Web ページを EUC-JP として記録した場合，スクリプトでは正しく再生されません。

2 手作業による英語以外の言語の Web ページの記録

web_sjis_to_euc_param 関数を使用すると，EUC-JP でエンコードされている HTML ページについて，その記録と再生を行うためのすべてのサポートを手作業で追加できます。これにより，EUC エンコードされた日本語文字を 仮想ユーザ・スクリプトで正しく表示できるようになります。

web_sjis_to_euc_param を使用すると，パラメータの値が実行ログで EUC-JP エンコーディングを使用して表示されます。たとえば，**web_find** 関数を再生するとき，VuGen ではエンコードされた値が表示されます。これらの値には，**web_sjis_to_euc_param** 関数によって EUC に変換された文字列値や，［**実行環境の設定**］>［**ログ**］>［**拡張ログ**］で有効になっているときのパラメータ置換が含まれます。

リファレンス

トラブルシューティングと制限事項

このセクションでは，英語以外の言語で作業する場合のトラブルシューティングと制限事項について説明します。

ブラウザの設定

記録中，スクリプト内の非英語の文字がエスケープされた 16 進数として表示されることがあります（たとえば文字列 "&" が "%DC%26" になるなど）。そのような場合は，URL を UTF-8 エンコーディングで送信しないようブラウザを設定することによって修正できます。Internet Explorer では，［**ツール**］>［**インターネット オプション**］を選択し，［**詳細設定**］タブをクリックします。そして，［ブラウズ］セクションにある［**常に UTF-8 として URL を送信する**］オプションをクリアします。

web_sjis_to_euc_param の詳細については，*オンライン関数リファレンス*を参照してください。

プロトコルに関する制限

SMTP プロトコル

SMTP プロトコルを MS Outlook や MS Outlook Express を通じて使用している場合は，仮想ユーザ・スクリプトに記録される日本語テキストが正しく表示されません。ただし，スクリプトの記録と再生は正しく実行されます。

スクリプト名の長さ

COM，FTP，IMAP，SMTP，POP3，REAL，または VBA の VB モードで記録するときは，スクリプト名の長さがマルチバイト文字で 10 文字（21 バイト）に制限されます。

Application Lifecycle Management の統合

Application Lifecycle Management プロジェクトに保存されているスクリプトを VuGen から開く場合や，ALM プロジェクトに保存されているシナリオを Controller から開く場合は，「Default」（英語）という名前の新しいテスト・セットを ALM プロジェクトに追加します。

45

上級ユーザのために

本章の内容

概念

タスク

リファレンス

概念

Dll 内の外部関数の呼び出し

外部 DLL で定義されている関数を呼び出せます。スクリプトから外部関数を呼び出すことにより，スクリプトと実行環境全体で必要とするメモリを減らせます。

外部関数を呼び出すには，その関数が定義されている DLL をロードします。

DLL は，次のいずれかの方法でロードできます。

- ローカル：1 つのスクリプトにロードする場合には，**lr_load_dll** 関数を使用します。タスクの詳細については，1245 ページの「DLL をローカルにロードする方法」を参照してください。
- グローバル：すべてのスクリプトにロードする場合には，**vugen.dat** ファイルにステートメントを追加します。タスクの詳細については，1247 ページの「DLL をグローバルにロードする方法」を参照してください。

OLE サーバの記録

VuGen では現在，OLE アプリケーションの記録はサポートされていません。OLE アプリケーションでは，実際のプロセスが標準のプロセス生成ルーチンによってではなく，OLE オートメーション・システムによって起動されます。ただし，以下に示すガイドラインに沿って，OLE アプリケーション用の Vuser スクリプトが作成できます。

OLE サーバには，実行可能ファイルと DLL の 2 つの種類があります。

DLL サーバ

サーバが DLL である場合，このサーバは最終的にアプリケーションのプロセス空間にロードされ，VuGen は LoadLibrary への呼び出しを記録します。この場合，ユーザはこれが OLE アプリケーションであることに気付かないかもしれません。

実行可能サーバ

サーバが実行形式の場合は，以下に示す方法で VuGen から実行ファイルを起動する必要があります。

- まず，実際に記録する必要があるプロセスを特定します。多くの場合，アプリケーションの実行可能ファイルの名前がわかっています。名前がわからない場合は，対象アプリケーションを起動し，NT のタスク・マネージャでその名前を確認します。
- 必要なプロセスを特定したら，VuGen で［**記録開始**］をクリックします。アプリケーション名の入力を要求されるので，OLE アプリケーションの名前と，その後ろに「/Automation」というフラグを入力します。次に，VuGen からではなく通常の方法でユーザ・プロセスを実行します。VuGen は実行中の OLE サーバを記録し，同じサーバを別に起動することはありません。VuGen で OLE サーバのアクションを記録するには，ほとんどの場合，この手順でうまくいきます。
- それでもうまく記録できない場合は，*CmdLine* プログラムを使って，直接起動されないプロセスの完全なコマンド・ラインを調べます（このプログラムは，カスタマ・サポート Web サイト http://support.hp.com の知識ベースの記事からダウンロードできます)。

CmdLine の使用法

次の例では，*CmdLine.exe* を使って，ほかのプロセスによって起動されるプロセス MyOleSrv.exe の完全なコマンド・ラインを調べています。

完全なコマンド・ラインを調べるには，次の手順を実行します。

1. *MyOleSrv.exe* の名前を *MyOleSrv.orig.exe* に変更します。
2. アプリケーションと同じディレクトリに *CmdLine.exe* を入れ，この名前を *MyOleSrv.exe* に変更します。
3. *MyOleSrv.exe* を起動します。この *MyOleSrv.exe* は，元のアプリケーションの完全なコマンド・ラインを含むポップアップ・メッセージ（追加情報を含む）を表示し，その情報を *c:¥temp¥CmdLine.txt* に書き込みます。
4. それぞれを元の名前に戻し，正しいコマンド・ライン・パラメータを使って OLE サーバ *MyOleSrv.exe* を VuGen から起動します。ユーザ・アプリケーションは VuGen からではなく，通常の方法で起動します。ほとんどの場合，VuGen は正しく記録を行います。

それでもうまく記録できない場合は，次の処理を行います。

1. OLE サーバ名を MyOleSrv.1.exe に，CmdLine を MyOleSrv.exe に変更します。
2. 環境変数「CmdStartNotepad」と「CmdNoPopup」を「1」に設定します。CmdLine 環境変数については，1236 ページの「CmdLine 環境変数」の一覧を参照してください。
3. VuGen 以外からアプリケーションを起動します。「メモ帳」が開き，完全なコマンド・ラインが表示されます。コマンド・ライン引数を調べます。アプリケーションを数回起動し，コマンド・ライン引数を比較します。何度アプリケーションを起動しても引数が同じである場合は，CmdStartNotepad 環境変数をリセットします。そうでなければ，設定を「1」のままにしておきます。
4. VuGen で，コマンド・ライン・パラメータを使用して（「メモ帳」のウィンドウからコピー / 貼り付けで指定してください）プログラム MyOleSrv.1.exe を起動します。
5. VuGen 以外からアプリケーションを起動します。

CmdLine 環境変数

以下に示す環境変数を使うことで，CmdLine の実行を制御できます。

CmdNoPopup これが設定されていると，ポップアップ・ウィンドウが現れません。

CmdOutFileName これが設定されており，空でない場合は，CmdLine は c:¥temp¥CmdLine.txt のかわりにこのファイルを作成しようとします。

CmdStartNotepad これが設定されていると，出力ファイルがメモ帳に表示されます（CmdNoPopup との併用をお勧めします）。

UNIX コマンド・ラインからの仮想ユーザの実行

VuGen には，仮想ユーザと同じ操作をコマンド・ラインから自動的に実行する UNIX シェル・スクリプト・ユーティリティ *run_db_Vuser.sh* が含まれています。このユーティリティは各再生ステップを個別に実行できます。このツールは，UNIX 上で再生するテストをデバッグするのに便利です。

run_db_Vuser.sh を $M_LROOT/bin ディレクトリに配置します。仮想ユーザタイプを再生するには，次のように入力します。

```
run_db_Vuser.sh   Vuser.usr
```

次のコマンド・ライン・オプションを使用することもできます。

-cpp_only	このオプションは，前処理のフェーズを開始します。この処理によって，「*Vuser.c*」が生成されます。
-cci_only	このオプションは，コンパイルのフェーズを実行します。「*Vuser.c*」ファイルは入力データとして使用されます。この処理によって，「*Vuser.ci*」ファイルが生成されます。
-exec_only	このオプションは，「*Vuser.ci*」ファイルを入力データとして受け取り，再生ドライバを介して仮想ユーザを実行します。
-ci ci_file	このオプションを使って，実行する .ci ファイルの名前と場所を指定できます。2 つ目のパラメータに，.ci ファイルの場所を指定します。
-out output_directory	このオプションを使って，各種の処理によって作成される出力ファイルの場所を指定できます。2 つ目のパラメータに，ディレクトリの名前と場所を指定します。
-driver driver_path	このオプションを使って，仮想ユーザの実行に使用する実際のドライバ実行可能ファイルを指定できます。標準設定では，ドライバ実行可能ファイルは VuGen の .dat ファイル内の設定から取得されます。

最初の 3 つのオプションは，run_db_vuser を実行するとき，一度にどれか 1 つだけを使用できます。

仮想ユーザの動作の指定

VuGen は仮想ユーザ・スクリプトとの動作を 2 つの独立した情報源として作成するので，たとえば待機時間，時間間隔，反復のループ，ログの記録などのユーザの動作を，仮想ユーザ・スクリプトを直接参照せずに設定できます。この機能により，仮想ユーザの設定が変更できると同時に，同じ仮想ユーザ スクリプトについて「プロファイル」を複数格納できます。

標準設定では，仮想ユーザの動作は VuGen の［実行環境設定］ダイアログ・ボックスで指定されているとおりに，「*Vuser.cfg*」ファイルに定義されています。このファイルの，ユーザ動作ごとに異なる複数のバージョンを保存することができます。そして，関連する *.cfg* ファイルを参照する仮想ユーザ・スクリプトを実行できます。

サーバ・マシンから，使用する設定ファイルを指定して仮想ユーザ ・スクリプトを実行できます。これを行うには，次のパラメータ指定を仮想ユーザのコマンド・ラインに追加します。

```
-cfg c:¥tmp¥profile2.cfg
```

コマンド・ライン・パラメータについては，1239 ページの「コマンド・ライン・パラメータ」を参照してください。

VuGen からは，振る舞いを定義したファイルを指定できません。VuGen は仮想ユーザと同じ名前の *.cfg* ファイルを自動的に使用します。（もちろん，ファイル名を「*Vuser.cfg*」に変更できます）。ただし，上で説明した -cfg パラメータをドライバのコマンド・ラインの最後に追加すれば，コマンド・ラインから手作業でファイルを指定できます。

注： UNIX 用のユーティリティ *run_db_vuser* では，このオプションはまだサポートされていません。

コマンド・ライン・パラメータ

仮想ユーザは，起動時にコマンド・ライン・パラメータを受け付けます。仮想ユーザ API には，コマンド・ライン・パラメータを参照するための関数がいくつかあります（**lr_get_attrib_double** など）。環境内で，スクリプト・ウィンドウのコマンド・ライン・エントリにパラメータを追加することで，仮想ユーザにコマンド・ライン・パラメータを送ることができます。

VuGen から仮想ユーザを実行するときは，コマンド・ライン・パラメータを指定できません。ただし，Windows のコマンド・ラインでほかのすべてのドライバ・パラメータの後，つまり行の最後にパラメータを追加することでこれを手作業で指定できます。

```
mdrv.exe -usr c:¥tmp¥Vuser¥Vuser.usr    -out c:¥tmp¥vuser
vuser_command_line_params
```

注：UNIX 用のユーティリティ *run_db_vuser* では，このオプションはまだサポートされていません。

タスク

新しい仮想ユーザ・タイプを作成する方法

次の手順では，新しい仮想ユーザ・タイプを作成する方法について説明します。

- 1241 ページの「mdrv.dat ファイルの編集」
- 1243 ページの「CFG ファイルの追加（任意）」
- 1244 ページの「LRP ファイルの挿入」
- 1245 ページの「テンプレートの指定」

1 mdrv.dat ファイルの編集

mdrv.dat ファイルを編集します。このファイルは，M_LROOT¥dat ディレクトリにあります。次のリストから該当するすべてのパラメータを使用し，新しい仮想ユーザ・タイプのためのセクションを追加します。

```
[<extension_name>]
ExtPriorityType=< {internal, protocol}>
WINNT_EXT_LIBS=<NT 用 DLL 名 >
WIN95_EXT_LIBS=<95 用 DLL 名 >
SOLARIS_EXT_LIBS=<Solaris 用 dll 名 >
LINUX_EXT_LIBS=<Linux 用 dll 名 >
HPUX_EXT_LIBS=<HP 用 dll 名 >
AIX_EXT_LIBS=<IBM 用 dll 名 >
LibCfgFunc=< 設定関数名 >
UtilityExt=< ほかの拡張子リスト >
WINNT_DLLS=< インタプリタ・コンテキストにロードする DLL（NT 用）>
WIN95_DLLS=< インタプリタ・コンテキストにロードする DLL（95 用）>
SOLARIS_DLLS=< インタプリタ・コンテキストにロードする dll（Solaris 用）>
LINUX_DLLS=< インタプリタ・コンテキストにロードする dll（Linux 用）>
HPUX_DLLS=< インタプリタ・コンテキストにロードする dll（HP 用）>
AIX_DLLS=< インタプリタ・コンテキストにロードする dll（IBM 用）>
ExtIncludeFiles=< 追加インクルード・ファイル。複数のファイルをカンマで区
切って指定できる >
ExtCmdLineConc=< 追加コマンド・ライン（属性がある場合は値を連結する）>
ExtCmdLineOverwrite=<追加コマンド・ライン（属性がある場合は値を上書きする）>
CallActionByNameFunc=< インタプリタ exec_action 関数 >
GetFuncAddress=< インタプリタ get_location 関数 >
RunLogicInitFunc=<action_logic init 関数 >
RunLogicRunFunc=<action_logic run 関数 >
RunLogicEndFunc=<action_logic end 関数 >
```

たとえば，Oracle NCA 仮想ユーザ・タイプは，以下で表されます。

```
[Oracle_NCA]
ExtPriorityType=protocol
WINNT_EXT_LIBS=ncarp11i.dll
WIN95_EXT_LIBS=ncarp11i.dll
LINUX_EXT_LIBS=liboranca11i.so
SOLARIS_EXT_LIBS=liboranca11i.so
HPUX_EXT_LIBS=liboranca11i.sl
AIX_EXT_LIBS=liboranca11i.so
LibCfgFunc=oracle_gui_configure
UtilityExt=lrun_api,HttpEngine
ExtCmdLineOverwrite=-WinInet No
ExtCmdLineConc=-UsingWinInet No
SecurityRequirementsFiles=oracle_nca.asl
SecurityMode=On
```

VuGen は，コードに変更を加えずに新しい仮想ユーザ ・タイプを処理できるように設計されています。ただし，特別なビューを追加しなければならない場合もあります。

VuGen では汎用のドライバは提供されていませんが，既存のドライバをカスタマイズできます。カスタマイズしたドライバを使用するには，*mdrv.dat* を変更します。プラットフォームと既存のドライバの行を追加した後，カスタマイズしたドライバの名前の行を < *プラットフォーム* >*_DLLS*=< *再生用 DLL 名* > の形式で追加します。たとえば，SAP の再生用 DLL が SAPPLAY32.DLL という名前の場合は，次の 2 行を *mdrv.dat* の [sap] セクションに追加します。

```
WINNT=sapdrv32.exe
WINNT_DLLS=sapplay32.dll
```

2 CFG ファイルの追加（任意）

プロトコルに標準の［実行環境の設定］を設定するために，設定ファイルを指定できます。これを行うには，mdrv.dat ファイル内の LibCfgFunc 変数で定義するか，テンプレートの下の新しいプロトコル・サブディレクトリに default.cfg というファイルをおきます。サンプルの .cfg は次のとおりです。

```
[ThinkTime]
Options=NOTHINK
Factor=1
LimitFlag=0
Limit=1

[Iterations]
NumOfIterations=1
IterationPace=IterationASAP
StartEvery=60
RandomMin=60
RandomMax=90

[Log]
LogOptions=LogExtended
MsgClassData=0
MsgClassParameters=0
MsgClassFull=1
```

3 LRP ファイルの挿入

dat¥protocols ディレクトリに，プロトコルを定義する *lrp* ファイルを挿入します。このファイルには，Protocol，Template，VuGen，API というセクションにプロトコルの設定情報が含まれています。プロトコルの中には，追加の実行環境設定オプションに応じて，これ以外のセクションがあるものもあります。

Protocol セクションには，プロトコルの名前，カテゴリ，説明，ビットマップの場所などが記述されています。

```
 [Protocol]
Name=WAP
CommonName=WAP
Category=Wireless
Description=Wireless Application Protocol - used for Web-based, wireless
communication between mobile devices and content providers.
Icon=bitmaps¥wap.bmp
Hidden=0
Single=1
Multi=0
```

Template セクションには，スクリプトのさまざまなセクションの名前と標準のテスト名が記載されています。

```
[Template]
vuser_init.c=init.c
vuser_end.c=end.c
Action1.c=action.c
Default.usp=test.usp
@@TestName@@.usr=wap.usr
default.cfg=default.cfg
```

VuGen セクションには，記録と再生エンジン，必要な DLL と実行時ファイルに関する情報が記述されています。

API セクションには，プロトコルのスクリプト API 関数についての情報が記述されています。

プロトコル・ディレクトリ内の任意の lrp ファイルを新しいプロトコルのひな形として使用できます。

4 テンプレートの指定

lrp ファイルを追加したら，*M_LROOT*¥template の下にサブディレクトリを作成し，*lrp* ファイルで定義したプロトコル名に対応する名前を付けます。このサブディレクトリに，一般設定および実行環境設定のための標準の設定を収めた *default.cfg* ファイルをおきます。

新しいプロトコルのすべてのスクリプトでグローバルなヘッダー・ファイルを使用する場合には，*globals.h* という名前のファイルを追加します。このファイルには，新しいプロトコルのためのヘッダー・ファイルを指す include ステートメントを含めておきます。たとえば，template¥http サブディレクトリには，*globals.h* というファイルがあり，include ディレクトリにある *as_web.h* ファイルをインクルードしています。

```
#include #as_web.h"
```

DLL をローカルにロードする方法

このタスクでは，仮想ユーザ・スクリプトに DLL をロードするために **lr_load_dll** 関数を使用する方法について説明します。一度 DLL をロードすれば，その DLL で定義されている任意の関数をスクリプト内で宣言せずに呼び出せます。

DLL 内で定義されている関数を呼び出すは，次の手順を実行します。

1 **lr_load_dll** 関数を使用して，スクリプトの先頭で DLL を読み込みます。このステートメントを *vuser_init* セクションの先頭に置きます。**ci_load_dll** 関数が **lr_load_dll** 関数に置き換えられます。

次の構文を使用します。

```
lr_load_dll(library_name);
```

UNIX プラットフォームでは，DLL は共有ライブラリと呼ばれています。ライブラリの拡張子はプラットフォームによって異なります。

2 DLL 内で定義されている関数をスクリプト内の適切な場所で呼び出します。

次の例では，**Test_1** テーブル作成後，**orac1.dll** で定義されている **insert_vals** 関数を呼び出します。

```
int LR_FUNC Actions(LR_PARAM p)
{
lr_load_dll("orac1.dll");

lrd_stmt(Csr1, "create table Test_1 (name char(15), id integer)¥n", -1,
        1 /*Deferred*/, 1 /*Dflt Ora Ver*/, 0);
lrd_exec(Csr1, 0, 0, 0, 0, 0);

/* insert_vals 関数を呼び出し，値をテーブルに挿入する。*/
insert_vals();

lrd_stmt(Csr1, "select * from Test_1¥n", -1, 1 /*Deferred*/, 1 /*Dflt Ora Ver*/, 0);
lrd_bind_col(Csr1, 1, &NAME_D11, 0, 0);
lrd_bind_col(Csr1, 2, &ID_D12, 0, 0);
lrd_exec(Csr1, 0, 0, 0, 0, 0);
lrd_fetch(Csr1, -4, 15, 0, PrintRow14, 0);
…
```

注：DLL の完全パス名を指定できます。**lr_load_library** 関数は，パスが指定されなかった場合，Windows プラットフォームでは C++ 言語関数 LoadLibrary で使用される標準的な検索順序に従って DLL を検索します。UNIX プラットフォーム（または同等のプラットフォーム）では，**LD_LIBRARY_PATH** 環境変数を設定できます。**lr_load_dll** 関数は，**dlopen** と同じ検索規則を使用します。詳細については，**dlopen** の main ページなどを参照してください。

DLL をグローバルにロードする方法

このタスクでは，その関数をすべての仮想ユーザ・スクリプトで利用できるようにするために，DLL をグローバルにロードする方法について説明します。一度 DLL をロードすれば，その DLL で定義されている任意の関数をスクリプト内で宣言せずに呼び出せます。

DLL をグローバルにロードするには，次の手順で行います。

1 （アプリケーションの *dat* ディレクトリにある）*mdrv.dat* ファイルの適切なセクションに，ロードする DLL のリストを追加します。

次の構文を使用します。

```
PLATFORM_DLLS=my_dll1.dll, my_dll2.dll, …
```

文字列 *PLATFORM* を使用するプラットフォームに置き換えます。プラットフォームのリストについては，*mdrv.dat* ファイルの最初のセクションを参照してください。

たとえば，NT プラットフォーム上の WinSock 仮想ユーザの DLL をロードするには，mdrv.dat ファイルに次の文を追加します。

```
[WinSock]
ExtPriorityType=protocol
WINNT_EXT_LIBS=wsrun32.dll
WIN95_EXT_LIBS=wsrun32.dll
LINUX_EXT_LIBS=liblrs.so
SOLARIS_EXT_LIBS=liblrs.so
HPUX_EXT_LIBS=liblrs.sl
AIX_EXT_LIBS=liblrs.so
LibCfgFunc=winsock_exten_conf
UtilityExt=lrun_api
ExtMessageQueue=0
ExtCmdLineOverwrite=-WinInet No
ExtCmdLineConc=-UsingWinInet No
WINNT_DLLS=user_dll1.dll, user_dll2.dll, …
```

2 DLL 内で定義されている関数をスクリプト内の適切な場所で呼び出します。

リファレンス

.dat ファイル

VuGen は，vugen.dat と mdrv.dat という 2 つの .dat ファイルを使用します。

vugen.dat

この vugen.dat ファイルは M_LROOT¥dat ディレクトリにあり，VuGen についての一般情報が含まれています。VuGen と Controller およびコンソールの両方で使用されます。

```
[Templates]
RelativeDirectory=template
```

Templates セクションは，VuGen プロトコル用のテンプレートの場所を示します。標準のエントリは，これらのテンプレートが相対 *template* ディレクトリにあることを示します。各プロトコルには，*template* の下にサブディレクトリがあり，この中には，そのプロトコル用のテンプレート・ファイルが含まれています。

次のセクションは **GlobalFiles** セクションです。

```
[GlobalFiles]
main.c=main.c
@@TestName@@.usr=test.usr
default.cfg=test.cfg
default.usp=test.usp
```

GlobalFiles セクションには，新規テストが作成されたときに VuGen がテスト・ディレクトリにコピーしたファイルの一覧が含まれます。たとえば，「user1」というテストがある場合，VuGen は *main.c*，*user1.usr*，および *user1.cfg* をテスト・ディレクトリにコピーします。

ActionFiles セクションには，仮想ユーザによって実行されるアクションを含むファイルの名前と，反復を実行する仮想ユーザが含まれます。

```
[ActionFiles]
@@actionFile@@=action.c
```

上に示した設定に加え，*vugen.dat* には，オペレーティング・システムおよびコンパイルに関連するそのほかの設定が含まれます。

mdrv.dat

mdrv.dat ファイルには，ライブラリ・ファイルとドライバの実行可能ファイルの場所を定義する，プロトコル別のセクションがあります。新しいプロトコルを作成するためにこのファイルを使用する方法の詳細については，1240 ページの「新しい仮想ユーザ・タイプを作成する方法」を参照してください。

46

UNIX でのスクリプトの作成と実行

本章の内容

概念

タスク

概念

UNIX でのスクリプトの作成と実行 - 概要

UNIX 環境では，次の方法で VuGen を使用できます。

- VuGen を使って，UNIX プラットフォームで実行可能な仮想ユーザ・スクリプトを作成できます。Windows 環境でアプリケーションを記録して，それを UNIX で実行します（記録は UNIX ではサポートされていません）。
- UNIX だけの環境を使用している場合は，仮想ユーザ・スクリプトをプログラミングによって作成できます。スクリプトは C 言語または C++ 言語でプログラミングして，ダイナミック・ライブラリとしてコンパイルする必要があります。

プログラミングによってスクリプトを作成するには，仮想ユーザのテンプレートを土台に，より大きな仮想ユーザ・スクリプトを作成します。テンプレートで提供するものは，次のとおりです。

- 正しいプログラム構造
- 仮想ユーザ API 呼び出し
- ダイナミック・リンク・ライブラリを作成するためのソース・コードとメイクファイル

仮想ユーザのアクションのプログラミング

仮想ユーザ・スクリプトの *test*.c，*test*.usr，*test*.cfg ファイルは，使用する仮想ユーザに応じてカスタマイズできます。

実際の仮想ユーザのアクションをプログラミングして，*test.c* ファイルに挿入します。このファイルには，プログラミングされる仮想ユーザ・スクリプトに必要な構造になっています。仮想ユーザ・スクリプトには，vuser_init，Actions，vuser_end の 3 つのセクションが含まれます。

C++ のユーザに対しては，テンプレートは extern C を定義します。エクスポートされる関数が不用意に変更されることがないように，すべての C++ ユーザに対して この定義を行う必要があります。

```
#include "lrun.h"
#if defined(__cplusplus) || defined(cplusplus) extern "C"
{
#endif
int LR_FUNC vuser_init(LR_PARAM p)
{
                lr_message("vuser_init done¥n");
                return 0;
}
 int Actions(LR_PARAM p)
{
                lr_message("Actions done¥n");
                return 0;
}
 int vuser_end(LR_PARAM p)
{
                lr_message("vuser_end done¥n");
                return 0;
}
#if defined(__cplusplus) || defined(cplusplus)}
#endif
```

仮想ユーザのアクションをプログラミングして空のスクリプトに直接挿入します。場所は，各セクションの **lr_message** 関数の前です。

vuser_init セクションは，初期化中に最初に実行されます。このセクションには，接続情報とログオンの処理を挿入します。*vuser_init* セクションは，スクリプトの実行時に一度だけ実行されます。

Actions セクションは初期化の後に実行されます。このセクションには，仮想ユーザによって実行される実際の操作を挿入します。Actions セクションを繰り返すように仮想ユーザを設定できます（*test*.cfg ファイルを使います）。

vuser_end セクションは，最後，つまり仮想ユーザのすべてのアクションの実行後に実行されます。このセクションには，クリーンアップとログオフの処理を挿入します。*vuser_end* セクションは，スクリプトの実行時に一度だけ実行されます。

注：LoadRunner は，UNIX のシグナル，SIGHUP，SIGUSR1，SIGUSR2 を送信することによって仮想ユーザを制御します。仮想ユーザ・プログラムでは，これらのシグナルを使用しないでください。

タスク

テンプレートの作成方法

VuGen には，テンプレートを作業ディレクトリにコピーするユーティリティが含まれています。このユーティリティは mkdbtest と呼ばれているもので，$M_LROOT¥bin にあります。このユーティリティを実行するには，次のように入力します。

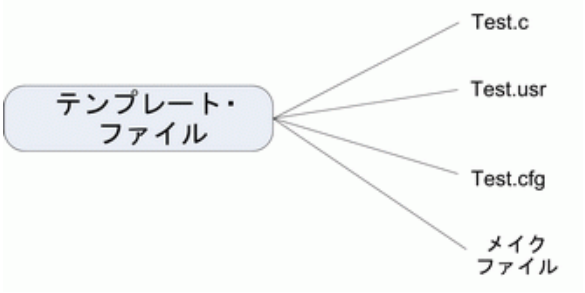
```
mkdbtest name
```

mkdbtest を実行すると，mkdbtest はテンプレート・ファイル name.c を格納するためのディレクトリ name を作成します。たとえば，次のように入力するとします。

```
mkdbtest test1
```

mkdbtest は，テンプレート・スクリプト *test1.c* を格納するためのディレクトリ *test1* を作成します。

mkdbtest ユーティリティを実行すると，4 つのファイル（*test.c*，*test.usr*，*test.cfg*，*Makefile*）を格納するためのディレクトリが作成されます。これらのファイルの「*test*」の部分には，mkdbtest で指定したテスト名が入ります。



実行環境設定を手動で設定する方法

仮想ユーザの実行環境を設定するには，スクリプトともに作成する *default.cfg* と *default.usp* に変更を加えます。このような実行環境の設定は，VuGen の実行環境の設定に相当します。詳細については，第 11 章，「実行環境の設定」を参照してください。*default.cfg* ファイルには，一般設定，思考遅延時間，およびログに関する設定が含まれています。*default.usp* ファイルには，実行論理とペース設定のための設定が含まれています。

一般オプション

UNIX 仮想ユーザ・スクリプト用の一般オプションが 1 つあります。

ContinueOnError は，エラーが発生しても実行を継続するように仮想ユーザに指示します。このオプションを有効にするには，値を 1 にします。このオプションを無効にするには，値を 0 にします。

次の例では，仮想ユーザはエラーが発生しても実行を継続します。

```
[General]
ContinueOnError=1
```

思考遅延時間のオプション

思考遅延時間のオプションを設定して，スクリプト実行時の仮想ユーザによる思考遅延時間の使用法を制御できます。次の図に基づいて，パラメータ Options，Factor，LimitFlag，Limit のパラメータを設定します。

オプション	Options	Factor	LimitFlag	Limit
思考遅延時間を無視する	NOTHINK	なし	なし	なし
記録された思考遅延時間を使用	RECORDED	1.000	なし	なし
記録された思考遅延時間を乗じる値	MULTIPLY	数値	なし	なし

オプション	Options	Factor	LimitFlag	Limit
記録された思考遅延時間の乱数率を使用	RANDOM	範囲	最小の割合	最大の割合
記録された思考遅延時間の上限	RECORDED/ MULTIPLY	数値（MULTIPLY 用）	1	秒単位の値

実行時に使用する思考遅延時間を制限するには，LimitFlag 変数を 1 に設定し，Limit により思考遅延時間の上限を秒単位で指定します。

次の例では，思考遅延時間を 50% ～ 150% のランダムな割合で乗じるように仮想ユーザに指定しています。

```
[ThinkTime]
Options=RANDOM
Factor=1
LimitFlag=0
Limit=0
ThinkTimeRandomLow=50
ThinkTimeRandomHigh=150
```

ログ・オプション

ログ・オプションは，スクリプトの実行中に簡略または詳細ログ・ファイルを作成するために設定できます。

```
[Log]
LogOptions=LogBrief
MsgClassData=0
MsgClassParameters=0
MsgClassFull=0
```

次の図に基づいて，パラメータ LogOptions．MsGClassData，MsgClassParameters，MsgClassFull の変数を設定します。

ログの種類	LogOptions	MsgClassData	MsgClassParameters	MsgClassFull
ログを無効にする	LogDisabled	なし	なし	なし
標準ログ	LogBrief	なし	なし	なし
パラメータの置換（のみ）	LogExtended	0	1	0
サーバから返されたデータ（のみ）	LogExtended	1	0	0
詳細トレース（のみ）	LogExtended	0	0	1
すべて	LogExtended	1	1	1

次の例では，仮想ユーザはサーバによって返されたすべてのデータと，置換に使用されたパラメータのログを記録します。

```
[Log]
LogOptions=LogExtended
MsgClassData=1
MsgClassParameters=1
MsgClassFull=0
```

反復と実行論理

反復のオプションを設定して，反復を複数回実行したり，反復のペースを制御したりできます。また，アクションの順番と重み付けを手動で設定することもできます。スクリプトの実行論理と反復の設定を変更するには，*default.usp* ファイルを編集する必要があります。

Actions セクションを複数回反復するように仮想ユーザに指示するには，反復の回数を RunLogicNumOfIterations の値として設定します。

反復の間隔（ペース）を指定するには，次の表に従って RunLogicPaceType 変数と関連する値を設定します。

ペースの設定	RunLogicPaceType	関連変数
すぐに次の反復を開始する	Asap	なし
次の反復を開始する前に，指定した時間だけ待機する	Const	RunLogicPaceConstTime
反復の間隔をランダムにする	ランダム	RunLogicRandomPaceMin, RunLogicRandomPaceMax
反復の後，一定の時間待機する	ConstAfter	RunLogicPaceConstAfterTime
反復の後，ランダムな時間だけ待機する	After	RunLogicAfterPaceMin, RunLogicAfterPaceMax

次の例は，反復を 4 回実行し，反復の間隔はランダムな長さにするよう仮想ユーザに対して指示する設定です。ランダムな間隔の範囲は 60 ～ 90 秒です。

```
[RunLogicRunRoot]
MercIniTreeFather=""
MercIniTreeSectionName="RunLogicRunRoot"
RunLogicRunMode="Random"
RunLogicActionOrder="Action,Action2,Action3"
RunLogicPaceType="Random"
RunLogicRandomPaceMax="90.000"
RunLogicPaceConstTime="40.000"
RunLogicObjectKind="Group"
RunLogicAfterPaceMin="50.000"
Name="Run"
RunLogicNumOfIterations="4"
RunLogicActionType="VuserRun"
RunLogicAfterPaceMax="70.000"
RunLogicRandomPaceMin="60.000"
MercIniTreeSons="Action,Action2,Action3"
RunLogicPaceConstAfterTime="30.000"
```

トランザクションを定義してランデブー・ポイントを手動で挿入する方法

VuGen を使用せずに仮想ユーザ・スクリプトをプログラミングするときには，トランザクションとランデブーを有効にするために，仮想ユーザ・ファイルを手動で設定する必要があります。これらの設定は，test.usr ファイルに含まれています。

```
[General]
Type=any
DefaultCfg=Test.cfg
BinVuser=libtest.libsuffix
RunType=Binary

[Actions]
vuser_init=
Actions=
vuser_end=

[Transactions]
transaction1=

[Rendezvous]
Meeting=
```

各トランザクションとランデブーは，*usr* ファイルに定義する必要があります。トランザクション名を Transactions セクションに追加します（トランザクション名に続けて "=" を指定します）。各ランデブー名を Rendezvous セクションに追加します（ランデブー名に続けて "=" を指定します）。セクションがない場合は，前述のように *usr* ファイルにセクションを追加します。

スクリプトを手動でコンパイルする方法

テンプレートを修正したら，スクリプト・ディレクトリの中で適切な *Makefile* を使用してスクリプトをコンパイルします。C++ でコンパイルをするときは，gnu コンパイラではなく，ネイティブ・コンパイラを使用する必要があります。コンパイラは，次のダイナミック・ライブラリを作成します。

- libtest.so（solaris）
- libtest.a（AIX）
- libtest.sl（HP）

Makefile の適切なセクションを変更することで，ほかのコンパイラ・フラグやライブラリを指定できます。

汎用のテンプレートを使った作業の場合には，アプリケーションのライブラリとヘッダ・ファイルをインクルードする必要があります。たとえば，アプリケーションで testlib というライブラリを使用している場合は，そのライブラリを LIBS セクションに指定します。

```
LIBS        = ¥
    -testlib ¥
    -lLrun50 ¥
    -lm
```

Makefile の変更後，作業ディレクトリのコマンド・ラインで「Make」と入力して，仮想ユーザ・スクリプト用のダイナミック・ライブラリ・ファイルを作成します。

スクリプトを作成したら，コマンド・ラインでその機能を確認します。

UNIX コマンド・ラインから仮想ユーザ・スクリプトを実行するには，次のように入力します。

```
mdrv -usr ‘pwd’ test.usr
```

ここで，*pwd* は 仮想ユーザ・スクリプトが格納されているディレクトリへの完全パスで，*test*.usr は 仮想ユーザ・ファイル名です。スクリプトがサーバと通信し，要求されているすべてのタスクを実行することを確認します。

スクリプトが正常に機能することを確認した後，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 設定）に組み込みます。詳細については，*HP LoadRunner Controller*，*Performance Center*，または *HP Business Availability Center* のドキュメントを参照してください。

47

XML API プログラミング

本章の内容

概念

タスク

概念

XML API プログラミング - 概要

VuGen の XML サポート機能により，テスト実行中に XML コードを動的に使用し，値を取得できます。効果的な XML スクリプトを作成するには，次の手順で行います。

- 使用するプロトコル（通常 Web，Web サービス，またはワイヤレス）でスクリプトを記録します。
- XML の構造をスクリプトにコピーします。
- 動的データと XML 要素の値を取得するには，LR API の XML 関数を追加します。

LR API は，XML Path 言語である XPath を使用して，XML ドキュメントのテキストを操作します。

［実行環境の設定］を使用することによって，［実行ログ］ウィンドウに XML 要素の出力値が表示されるようにできます。VuGen には，行番号，一致した件数，値が表示されます。値を表示するには，パラメータ置換を有効にする必要があります。［実行環境設定］ダイアログ・ボックスで［**一般：ログ**］ノードを開き，［**拡張ログ**］を選択して，［**パラメータ置換**］を選択します。詳細については，第 11 章，「実行環境の設定」。を参照してください。

仮想ユーザ API の XML 関数はすべて，正常に検索された一致の件数か，失敗を表す 0 を返します。

XML 関数の使用方法

このセクションでは，XML ツリーのデータの使用方法について例を示します。いくつかの関数では情報を取得できます。またいくつかの関数では XML ツリーに情報を書き込めます。これらの例では，Acme という会社に所属する複数の従業員の名前と内線番号が入っている次の XML ツリーを使用します。

```
<acme_org>
   <accounting_dept>
      <employee type=’PT’>
         <name>John Smith</name>
         <extension>2145</extension>
      </employee>
   </accounting_dept>
   <engineering_dept>
      <employee type=’PT’>
         <name>Sue Jones</name>
         <extension>2375</extension>
      </employee>
   </engineering_dept>
</acme_org>
```

XML ツリーからの情報の読み取り

XML ツリーから情報を読み取る関数は次のとおりです。

lr_xml_extract	XML 文字列から XML 文字列フラグメントを抽出します。
lr_xml_find	XML 文字列に対するクエリを実行します。
lr_xml_get_values	クエリで検出された XML 要素の値を取得します。

クエリを使用して特定の値を取得するには，パス形式で親ノードと子ノードのタグを指定します。

たとえば，Accounting 部門の従業員の名前を取得するには，次の文字列を使用します。

```
lr_xml_get_values("XML={XML_Input_Param}",
"ValueParam=OutputParam",
"Query=/acme_org/accounting_dept/employee/name",
LAST);
```

拡張ログ機能が有効な場合，［実行ログ］ウィンドウには，この関数の出力が表示されます。

```
Output:
Action.c(20): "lr_xml_get_values" was successful, 1 match processed
Action.c(25): Query result = John Smith
```

XML 構造への書き込み

XML ツリーに値を書き込む関数は次のとおりです。

lr_xml_delete	XML 文字列からフラグメントを削除します。
lr_xml_insert	新しい XML フラグメントを XML 文字列に挿入します。
lr_xml_replace	XML 文字列のフラグメントを置き換えます。
lr_xml_set_values	クエリで検出された XML 要素の値を設定します。
lr_xml_transform	XML データに XSL（Extensible Stylesheet Language）変換を適用します。

最もよく使用される*書き込み*関数は **lr_xml_set_values** です。この関数は，XML 文字列の指定された要素の値を設定します。次の例では，**lr_xml_set_values** を使用して，XML 文字列の 2 つの *employee* 要素の内線番号を変更しています。

まず，*XML_Input_Param* というパラメータに XML 文字列を保存します。2 つの値を検索して置き換えます。そこで，*ExtensionParam_1* と *ExtensionParam_2* という新しい 2 つのパラメータを用意し，それらの値を新しい 2 つの内線番号 1111 および 2222 に設定します。

lr_xml_set_values には，ExtensionParam_1 および ExtensionParam_2 の値を受け取る「ValueName=ExtensionParam」という引数が含まれています。2 人の従業員の現在の内線番号が，これらのパラメータの値，1111 および 2222 で置き換えられます。その後 OutputParam の値が評価され，新しい内線番号に確かに置き換えられたことが証明されます。

```
Action() {

    int i, NumOfValues;
    char buf[64];

    lr_save_string(xml_input, "XML_Input_Param"); // 入力をパラメータとして保存する
    lr_save_string("1111", "ExtensionParam_1");
    lr_save_string("2222", "ExtensionParam_2");

    lr_xml_set_values("XML={XML_Input_Param}",
        "ResultParam=NewXmlParam", "ValueParam=ExtensionParam",
        "SelectAll=yes", "Query=//extension", LAST);

    NumOfValues= lr_xml_get_values("XML={NewXmlParam}",
        "ValueParam=OutputParam", "Query=//extension",
        "SelectAll=yes", LAST);

    for (i = 0; i < NumOfValues; i++) {/* MultiParam の複数の値を出力する */

        sprintf(buf, "Retrieved value %d : {OutputParam_%d}", i+1, i+1);
        lr_output_message(lr_eval_string(buf));
    }

    return 0;
}
Output:
Action.c(40): Retrieved value 1: 1111
Action.c(40): Retrieved value 2: 2222
```

XML 関数のパラメータの指定

ほとんどの XML API 関数では，**XML 要素**と**クエリ**を指定する必要があります。また，すべての結果を取得するか，それとも 1 つの結果を取得するか指定できます。

XML 要素の定義

検索する XML 要素を定義するには，XML 要素をそのまま文字列として指定するか，XML 要素が含まれるパラメータを指定します。次の例では，XML 入力文字列をそのまま文字列として定義する場合を示します。

```
"XML=<employee>JohnSmith</employee>"
```

また，**XML** 文字列は，XML データを含むパラメータでもかまいません。次に例を示します。

```
"XML={EmployeeNameParam}"
```

XML ツリーのクエリ

XML タグに含まれている値（従業員の内線番号など）を検索するとします。必要とする値に対するクエリを作成します。クエリには，要素の場所と，取得または設定する要素を指定します。指定したパスにより，検索範囲が特定のタグに限定されます。また，ルートの下のすべてのノードで特定の種類の要素をすべて検索することもできます。

特定のパスを指定するには，「”Query=/<*XML フル・パス名*>/<*要素名*>”」を指定します。

同じ名前の要素をすべてのノードの下で検索するには，「”Query=//<*要素名*>”」を指定します。

VuGen における XML 関数の実装では，クエリの範囲は XML ツリー全体です。ツリー情報は，*xml* 引数の値として仮想ユーザ API 関数に送られます。

クエリの複数の検索

XML 要素に対してクエリを実行すると，標準では最初に一致したものだけが返されます。クエリで複数の値を取得するには，関数内で "SelectAll=yes" 属性を指定します。VuGen は，複数のパラメータを表すために _< *連番* > という形式のサフィックスを追加します。たとえば，*EmployeeName* という名前のパラメータを定義した場合は，*EmployeeName_1*，*EmployeeName_2*，*EmployeeName_3* などが作成されます。

```
lr_xml_set_values("XML={XML_Input_Param}",
"ResultParam=NewXmlParam", "ValueParam=ExtensionParam",
"SelectAll=yes", "Query=//extension", LAST);
```

パラメータに*書き込む*関数を使用すれば，書き込んだ後にパラメータの値を評価できます。たとえば次のコードでは，クエリによる複数の検索結果を取得して出力しています。

```
NumOfValues = lr_xml_get_values("Xml={XmlParam}", "Query=//name",
        "SelectAll=yes", "ValueParam=EmployeeName", LAST);
```

パラメータから値を*読み取る*関数の場合，パラメータの値は事前に定義しておく必要があります。また，パラメータは，< *パラメータ名* >_< *連番* > という形式（たとえば *Param*_1，*Param*_2，*Param*_3 など）を使用する必要もあります。このようなパラメータの集合をパラメータ・セットとも言います。

次の例では，lr_xml_set_values はパラメータ・セットから値を読み取り，その値を XPath クエリで使用しています。従業員の内線番号を表すパラメータ・セットには，ExtensionParam という名前が付いています。このパラメータ・セットには，ExtensionParam_1 および ExtensionParam_2 という 2 つのメンバがあります。**lr_xml_set_values** 関数は XML 入力文字列を検索し，最初に一致した値を 1111 に，2 番目に一致した値を 2222 に設定します。

```
lr_save_string("1111", "ExtensionParam_1");
lr_save_string("2222", "ExtensionParam_2");

lr_xml_set_values("XML={XML_Input_Param}",
    "ResultParam=NewXmlParam", "ValueParam=ExtensionParam",
    "SelectAll=yes", "Query=//extension", LAST);
```

XML 属性

VuGen では属性がサポートされています。要素を操作する場合と同じように，簡単な表現を使用して XML の要素とノードの属性を操作できます。特定の属性，または特定の値を持つ属性を変更することもできます。

次の例では，**lr_xml_delete** を使って，name 属性を持っている最初の cubicle 要素を削除しています。

```
lr_xml_delete("Xml={ParamXml}",
                        "Query=//cubicle/@name",
                            "ResultParam=Result",
                        LAST
    );
```

次の例では，**lr_xml_delete** を使って，Paul という値の name 属性を持っている最初の cubicle 要素を削除しています。

```
lr_xml_delete("Xml={ParamXml}",
          "Query=//cubicle/@name="Paul",
          "ResultParam=Result",
          LAST
    );
```

XML スクリプトの構造化

最初に，使用するプロトコルで新しいスクリプトを作成します。そのプロトコルでセッションを記録することも，また記録せずにスクリプト全体をプログラミングすることもできます。次のように，スクリプトの Actions セクションを作成します。

- XML 入力の宣言
- Actions セクション

XML 入力セクションには，入力変数として使用する XML ツリーを含めます。XML ツリーを char 型の変数として定義します。次に例を示します。

```
char *xml_input=
"<acme_org>"
    "<employee>"
        " <name>John Smith</name>"
        "<cubicle>227</cubicle>"
        "<extension>2145</extension>"
    "</employee>"
    "<employee>"
        "<name>Sue Jones</name>"
        "<cubicle>227</cubicle>"
        "<extension>2375</extension>"
    "</employee>"
"</acme_org>";
```

Action セクションには，変数の評価，および要素の値に対するクエリを含めます。次の例では，**lr_save_string** を使って XML 入力文字列を評価しています。入力変数を対象に，従業員名と内線番号を検索しています。

```
Action() {

    /* 入力をパラメータとして保存する */
    lr_save_string(xml_input, "XML_Input_Param");

    /* クエリ 1 - 指定された要素から従業員名を取得する */
    lr_xml_get_values("XML={XML_Input_Param}",
        "ValueParam=OutputParam",
        "Query=/acme_org/employee/name", LAST);

    /* クエリ 2 - ルート以下のすべてのパスで内線番号を取得する */
    lr_xml_get_values("XML={XML_Input_Param}",
        "ValueParam=OutputParam",
        "Query=//extension", LAST);

    return 0;
}
```

XML を使用した記録されたセッションの拡張

セッションを記録し，必要な XML 関数と仮想ユーザ API 関数を手作業で追加することによって，XML スクリプトを作成できます。

次の例では，記録されたセッションを仮想ユーザ API 関数を使って拡張する方法を示します。記録された関数は，太字で示した **web_submit_data** だけです。

最初のセクションには，SOAP メッセージを表す変数 SOAPTemplate が XML 入力宣言として含まれています。

```
#include "as_web.h"

// SOAP メッセージ
const char*pSoapTemplate=
        "<soap:Envelope xmlns:soap=¥"http://schemas.xmlsoap.org/soap/envelope/¥">"
            "<soap:Body>"
                "<SendMail xmlns=¥"urn:EmailIPortTypeInft-IEmailService¥"/>"
            "</soap:Body>"
        "</soap:Envelope>";
```

次のセクションは，ユーザのアクションを表しています。

```
Action1()
{
    // 応答の本体を取得する
    web_reg_save_param("ParamXml", "LB=", "RB=", "Search=body", LAST);

    // HTTP GET で天気をフェッチする
    web_submit_data("GetWeather",                    "Action=http://
glkev.net.innerhost.com/glkev_ws/
                                WeatherFetcher.asmx/GetWeather",
                            "Method=GET",
                            "EncType=",
                            "RecContentType=text/xml",
                            "Referer=http://glkev.net.innerhost.com
                                /glkev_ws/WeatherFetcher.asmx?op=GetWeather",
        "Snapshot=t2.inf",
        "Mode=HTTP",
        ITEMDATA,
        "Name=zipCode", "Value=10010", ENDITEM,
        LAST);

    // City の値を取得する
    lr_xml_get_values("Xml={ParamXml}",
                      "Query=City",
                      "ValueParam=ParamCity",
                      LAST
    );

    lr_output_message(lr_eval_string("***** City = {ParamCity} *****"));

    // State の値を取得する
    lr_xml_get_values("Xml={ParamXml}",
                      "Query=State",
                      "ValueParam=ParamState",
                      LAST
    );

    lr_output_message(lr_eval_string("***** State = {ParamState} *****"));
```

```
// テンプレート lr_xml_get_values_ex("Xml={ParamXml}" を使用して，複数の値を
一度に取得する。
                        "Template="
                            "<Weather>"
                                "<Time>{ParamTime}</Time>"
                                "<Temperature>{ParamTemp}</Temperature>"
                                "<Humidity>{ParamHumid}</Humidity>"
                                "<Conditions>{ParamCond}</Conditions>"
                            "</Weather>",
                        LAST
    );

    lr_output_message(lr_eval_string("***** Time = {ParamTime}, Temperature =
                                                {ParamTemp}, "
                                        "Humidity = {ParamHumid}, Conditions =
                                            {ParamCond} *****"));

    // 人に読める形式で予報を生成する
    lr_save_string(lr_eval_string("¥r¥n¥r¥n*** Weather Forecast for {ParamCity}, {ParamState} ***¥r¥n"
                                    "¥tTime: {ParamTime}¥r¥n"
                                    "¥tTemperature: {ParamTemp} deg. Fahrenheit¥r¥n"
                                    "¥tHumidity: {ParamHumid}¥r¥n"
                                    "¥t{ParamCond} conditions expected¥r¥n"
                                   "¥r¥n"),
                                    "ParamForecast"
    );

    // SOAP テンプレートをパラメータに保存する
    lr_save_string(pSoapTemplate, "ParamSoap");
```

```
// 要求の本体を SOAP テンプレートに挿入する
    lr_xml_insert("Xml={ParamSoap}",
                    "ResultParam=ParamRequest",
                    "Query=Body/SendMail",
                    "position=child",
                    "XmlFragment="
                            "<FromAddress>taurus@merc-int.com</FromAddress>"
                            "<ToAddress>support@merc-int.com</ToAddress>"
                            "<ASubject>Weather Forecast</ASubject>"
                            "<MsgBody/>",
                    LAST
    );

//
//      "<soap:Envelope xmlns:soap=¥"http://schemas.xmlsoap.org/soap/envelope/¥">"
//          "<soap:Body>"
//              "<SendMail xmlns=¥"urn:EmailIPortTypeInft-IEmailService¥"/>"
//                          "<FromAddress>taurus@merc-int.com</FromAddress>"
//                          "<ToAddress>support@merc-int.com</ToAddress>"
//                          "<ASubject>Weather Forecast</ASubject>"
//                          "<MsgBody/>"
//              "</SendMail>"
//          "</soap:Body>"
//      "</soap:Envelope>";
//

    // 実際の予報のテキストを挿入する
    lr_xml_set_values("Xml={ParamRequest}",
                        "ResultParam=ParamRequest",
                        "Query=Body/SendMail/MsgBody",
                        "ValueParam=ParamForecast",
                        LAST);
```

```
    // SOAP 用のヘッダを追加する
    web_add_header("SOAPAction", "urn:EmailIPortTypeInft-IEmailService");

    // 応答の本体を取得する
    web_reg_save_param("ParamXml", "LB=", "RB=", "Search=body", LAST);

    // SOAP 要求を使用して予報を受信者に送信する
    web_custom_request("web_custom_request",
        "URL=http://webservices.matlus.com/scripts/emailwebservice.dll/soap/IEmailservice",
        "Method=POST",
        "TargetFrame=",
        "Resource=0",
        "Referer=",
        "Body={ParamRequest}",
        LAST);

    // メールが送信されたことを確認する
    lr_xml_find("Xml={ParamXml}",
                "Query=Body/SendMailResponse/return",
                "Value=0",
                LAST
    );

    return 0;
}
```

タスク

結果パラメータを使用する方法

lr_xml 関数の中には，**ResultParam** のように結果パラメータを返すものがあります。このパラメータには，関数が実行された後の結果の XML データが含まれます。結果パラメータは，［パラメータの選択または作成］ダイアログ・ボックスのパラメータ・リストから使用できます。

たとえば，**lr_xml_insert** 関数の場合，ResultParam には新しい XML フラグメントの挿入の結果である完全 XML データが含まれます。

結果パラメータを Web Service 呼び出しのようなほかの XML 関連関数への入力として使用できます。再生時，VuGen によって結果パラメータの値がキャプチャされます。この値は，後の手順で入力引数として使用できます。

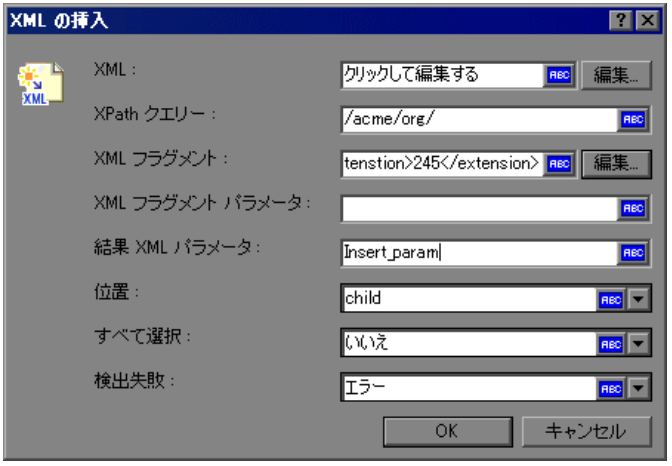
結果パラメータをサポートする関数には，**lr_xml_insert**，**lr_xml_transform**, **lr_xml_replace**，**lr_xml_delete**, and **lr_xml_set_values**. があります。

次の関数は，値を resultParam 以外のパラメータに保存します。**lr_xml_get_values** は値を ValueParam に，**lr_xml_extract** は値を XMLFragmentParam に保存します。これらの値は，パラメータ置換にも使用されます。

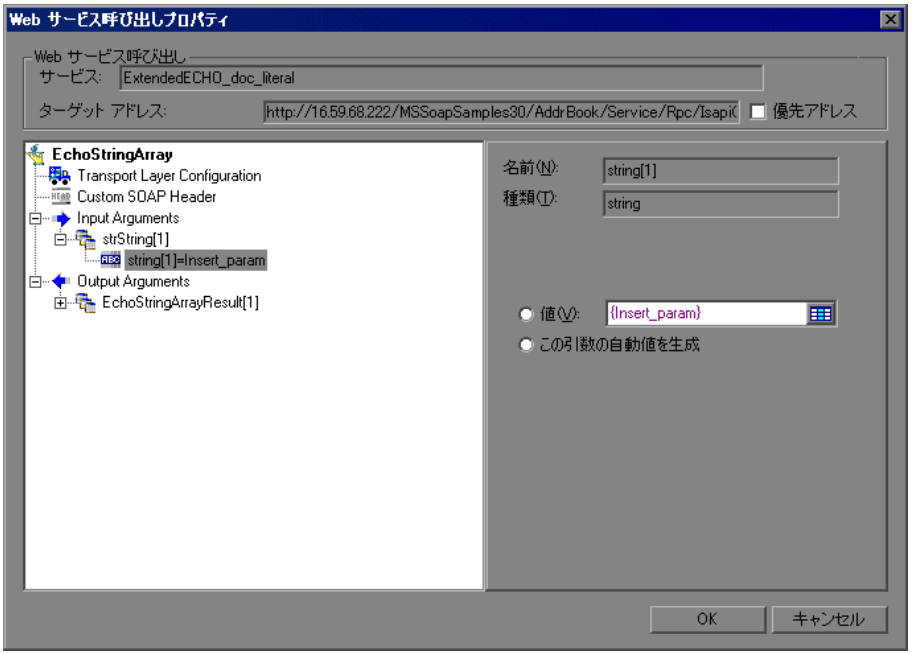
結果パラメータを入力に使用するには，次の手順で行います。

1 ツリー・ビューで，XML ステップをダブルクリックして，そのプロパティを表示します。

2 ［結果 XML パラメータ］ボックスで，**結果 XML パラメータ** の名前（または ValueParam および XMLFragmentParam）を指定します。



3 パラメータ名を入力引数として参照します。



詳細については，953 ページの「[入力引数] ノード」を参照してください。

索引

記号

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

z

あ

い

う

え

お

か

き

く

け

こ

さ

し

す

せ

と

な

に

わ