Mercury 仮想ユーザ・ジェネレータ

ユーザーズ・ガイド

第 1 巻 - 仮想ユーザ・ジェネレータの使い方

Version 8.1

MERCURY™

Mercury 仮想ユーザ・ジェネレータ・ユーザーズ・ガイド 第 1 巻 － VuGen の使い方，Version 8.1

Mercury Interactive Corporation
379 North Whisman Road
Mountain View, CA 94043
Tel: (650) 603-5200
Toll Free: (800) TEST-911
Customer Support: (877) TEST-HLP
Fax: (650) 603-5300

本書に関するご意見，ご要望は documentation@mercury.com まで電子メールにてお送りください。

LRDBUG8.1-01JP/01

目次

Mercury 仮想ユーザ・ジェネレータへようこそ

Mercury 仮想ユーザ・ジェネレータ **VuGen** は，仮想ユーザ・スクリプトを作成するためのツールです。VuGen では，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。これらのスクリプトを使用することで，現実世界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，LoadRunner，Performance Center，Tuning Module，Application Management などのいくつかの Mercury 製品と組み合わせて使用できます。

Mercury LoadRunner は，システムの動作やパフォーマンスを予測するための標準的なパフォーマンス・テスト製品です。LoadRunner の詳細なレポートとグラフは，アプリケーションのパフォーマンスを評価するために必要な情報を提供します。

Mercury Performance Center は，LoadRunner の機能をエンタープライズ・レベルで実装しています。

Mercury Tuning Module は，インフラストラクチャのボトルネックを特定，分離し，解決するための，事前対応型ソリューションです。

Mercury の Application Management ツール群は，実運用環境におけるビジネス・アプリケーションおよびシステムの管理と可用性を最適化するのに役立ちます。

この**項**では，Mercury LoadRunner の各種リソースについて説明します。

スクリプトを LoadRunner シナリオ，Performance Center 負荷テスト，Tuning Module セッション，または Business Process Monitor プロファイルに統合する方法の詳細については，該当する各ガイド（『**Mercury LoadRunner コントローラ・ユーザーズ・ガイド**』，**Performance Center**，**Tuning Console**，または **Application Management** の各マニュアル）を参照してください。

本章では，次の項目について説明します。

- ➤ 付属マニュアル
- ➤ LoadRunner 付属マニュアルの使い方
- ➤ 表記規則

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を提供します。

オンライン文書：全マニュアルを PDF 形式で提供します。オンライン文書は Adobe Acrobat Reader を使って読んだり，印刷したりできます（www.adobe.com を参照）。VuGen オンライン文書のアップデートについては，Mercury のカスタマー・サポート Web サイトをご覧ください。

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する LoadRunner の API 関数をすべて，その使用例とともにオンラインで参照できます。オンライン版の **LoadRunner 関数リファレンス**のアップデートについては，Mercury のカスタマー・サポート Web サイトをご覧ください。

LoadRunner 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。ヘルプ・ファイルのアップデートについては，Mercury のカスタマー・サポート Web サイトをご覧ください。

オンライン技術サポート：普段お使いの Web ブラウザで，Mercury のカスタマー・サポート Web サイトを開きます。このサイトでは，Mercury の最新情報や製品に関する情報をご覧になれます。この Web サイトの URL は，http://www.mercury.com/jp/services/support/ です。

サポート情報：Mercury の Web サイトとカスタマー・サポート・サイト，世界の Mercury の営業所を示します。

Mercury Interactive の Web サイト：普段お使いの Web ブラウザで，Mercury のホーム・ページ（http://www.mercury.com/jp）を開きます。このサイトでは，Mercury の最新情報や製品に関する情報をご覧になれます。

付属マニュアル

お使いの製品には，お買い上げ時の契約に応じて，印刷されたマニュアル一式がいくつか付属しています。製品に付属の Package Content カードには，お使いの製品に付属しているマニュアルの一覧が記載されています。

次の項では，LoadRunner のマニュアルについてのみ説明します。

LoadRunner には，次の手順について説明するマニュアル一式が付属しています。

- LoadRunner および Mercury Tuning Module のインストール
- 仮想ユーザ・スクリプトの作成
- LoadRunner コントローラおよび Mercury Tuning Console の使用
- LoadRunner モニタの設定
- LoadRunner アナリシスの使用

LoadRunner 付属マニュアルの使い方

LoadRunner のマニュアルは，インストール・ガイド，コントローラ・ユーザーズ・ガイド，コンソール・ユーザーズ・ガイド，モニタ・リファレンス，アナリシス・ユーザーズ・ガイド（以上各 1 冊ずつ），および，仮想ユーザ・スクリプトの作成に関するマニュアル（2 巻）で構成されています。

インストール・ガイド

LoadRunner のインストール方法については，『**Mercury LoadRunner インストール・ガイド**』を参照してください。『LoadRunner インストール・ガイド』では，次のインストール方法について説明します。

- LoadRunner および Mercury Tuning Module－Windows ベースのマシンへのインストール
- 仮想ユーザ・コンポーネント－ Windows マシンおよび UNIX プラットフォーム用

コントローラ・ユーザーズ・ガイド

LoadRunner の付属マニュアルには，コントローラ・ユーザーズ・ガイドが 1 冊含まれます。

『**Mercury LoadRunner コントローラ・ユーザーズ・ガイド**』は，Windows 環境で LoadRunner コントローラを使用して LoadRunner シナリオを作成し実行する方法を説明しています。仮想ユーザは，UNIX および Windows のプラットフォームで動作します。『コントローラ・ユーザーズ・ガイド』では，LoadRunner のテスト工程の概要を説明します。

コンソール・ユーザーズ・ガイド

Mercury Tuning Module の付属マニュアルには，コンソール・ユーザーズ・ガイドが 1 冊含まれます。

『**ProTune Console User's Guide（英語版）**』は，Windows 環境で Mercury Tuning Console を使用してチューニング・セッションを作成し，実行する方法を説明します。『コンソール・ユーザーズ・ガイド』では，Mercury Tuning Module のテスト工程の概要を説明します。

モニタ・リファレンス

LoadRunner の付属マニュアルには，モニタ・リファレンス・ガイドが 1 冊含まれます。

『**Mercury LoadRunner モニタ・リファレンス**』は，サーバ・モニタ環境を設定する方法，およびシナリオまたはセッション実行時に生成されるデータを監視するように LoadRunner モニタを設定する方法について説明します。

アナリシス・ユーザーズ・ガイド

LoadRunner の付属マニュアルには，アナリシス・ユーザーズ・ガイドが 1 冊含まれます。

『**Mercury LoadRunner アナリシス・ユーザーズ・ガイド**』は，システム・パフォーマンスの分析を行うために，シナリオまたはセッションの実行後に LoadRunner アナリシス・グラフとレポートを作成する方法について説明します。

仮想ユーザ・スクリプトの作成に関するガイド

LoadRunner の付属マニュアルには，スクリプトの作成に関するガイドが 1 冊含まれます。

『**Mercury 仮想ユーザ・ジェネレータ・ユーザーズ・ガイド**』は，VuGen を使ったスクリプトの作成方法を説明しています。GUI スクリプトの作成につい

ては，必要に応じ，このマニュアルと併せてオンラインの『**LoadRunner 関数リファレンス**』と『**WinRunner ユーザーズ・ガイド**』をお読みください。

注 :『**Mercury 仮想ユーザ・ジェネレータ・ユーザーズ・ガイド**』のオンライン版は 1 つのボリュームですが，印刷版は「第 1 巻－ **仮想ユーザの使い方**」および「第 2 巻 - **プロトコル**」の 2 冊から成ります。

情報	参照先
LoadRunner および Mercury Tuning Module のインストール	**Mercury LoadRunner インストール・ガイド**
LoadRunner のテスト・プロセス	**Mercury LoadRunner コントローラ・ユーザーズ・ガイド**
Mercury Tuning Module のテスト・プロセス	**Mercury Tuning Console User's Guide（英語版）**
仮想ユーザ・スクリプトの作成	**Mercury 仮想ユーザ・ジェネレータ・ユーザーズ・ガイド**
負荷テスト・シナリオの作成と実行	**Mercury LoadRunner コントローラ・ユーザーズ・ガイド**
チューニング・セッションの作成と実行	**Mercury Tuning Console User's Guide（英語版）**
サーバ・モニタの設定	**Mercury LoadRunner モニタ・リファレンス**
テスト結果の分析	**Mercury LoadRunner アナリシス・ユーザーズ・ガイド**

表記規則

本書では次の表記規則に従います。

1, 2, 3	太字の数字は，操作手順を示します。
➤	ブリット記号はオプションまたは特徴を示します。
＞	大なり記号はメニュー・レベルを区切ります（例：［**ファイル**］＞［**開く**］）。
［**太字**］	インタフェース要素の名前は，その要素を使用したアクションを実行する手順の説明で全角の大括弧に**太字**で示します（例：［**実行**］ボタンをクリックします）。
太字	メソッド名や関数名，変数名，新機能は，**太字**で示します。
Arial	使用例やユーザがそのまま入力しなければならない文字列は，Arial というフォントで示します。
＜＞	ファイル・パスまたは URL アドレスの中の可変部分は，山括弧で囲んで示します（例：**＜製品のインストール・フォルダ＞¥bin**）。
...	構文内の省略記号は，同じ形式で項目をさらに組み入れることができることを意味します。

第 1 部

仮想ユーザ・スクリプトの紹介

第 1 章

仮想ユーザ・スクリプトの作成

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。Mercury のツールは，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操作は**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，Mercury 仮想ユーザ・ジェネレータ **VuGen** です。

本章では，次の項目について説明します。

- 仮想ユーザの紹介
- 仮想ユーザのタイプ
- 仮想ユーザ・スクリプトの作成手順
- 本書の使用法

注：オンライン版の『**Mercury 仮想ユーザ・ジェネレータ・ユーザーズ・ガイド**』は 1 巻で構成されていますが，製本版は「第 1 巻 － **仮想ユーザ・ジェネレータの使い方**」と「第 2 巻 － **プロトコル**」の 2 巻で構成されています。

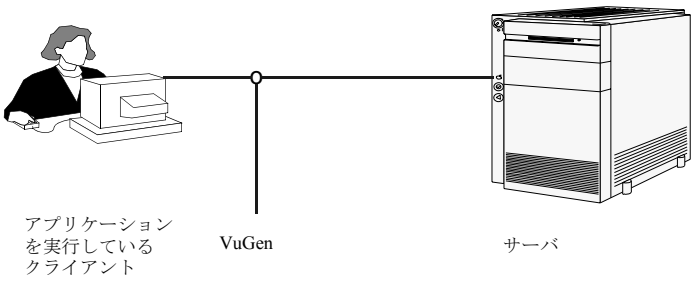
仮想ユーザの紹介

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。Mercury のツールは，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。

そのようなエミュレーションを実現するために，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操作は**仮想ユーザ・スクリプト**によって表現されます。

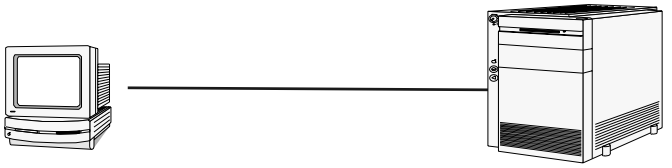
Mercury のツールの中で，仮想ユーザ・スクリプトを作成するための中心的なツールとなるのが仮想ユーザ・ジェネレータ（**VuGen**）です。VuGen では，クライアント・アプリケーションで典型的なビジネス・プロセスを実行しているユーザの操作を記録することによって，仮想ユーザ・スクリプトを作成します。VuGen は記録セッション中に実行された操作を記録しますが，記録されるのはクライアントとサーバの間のやり取りだけです。アプリケーションの API 関数からサーバへの呼び出しを手作業でプログラミングする代わりに，VuGen は，実際に起こった状況を正確にモデル化しエミュレートする関数を，自動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

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

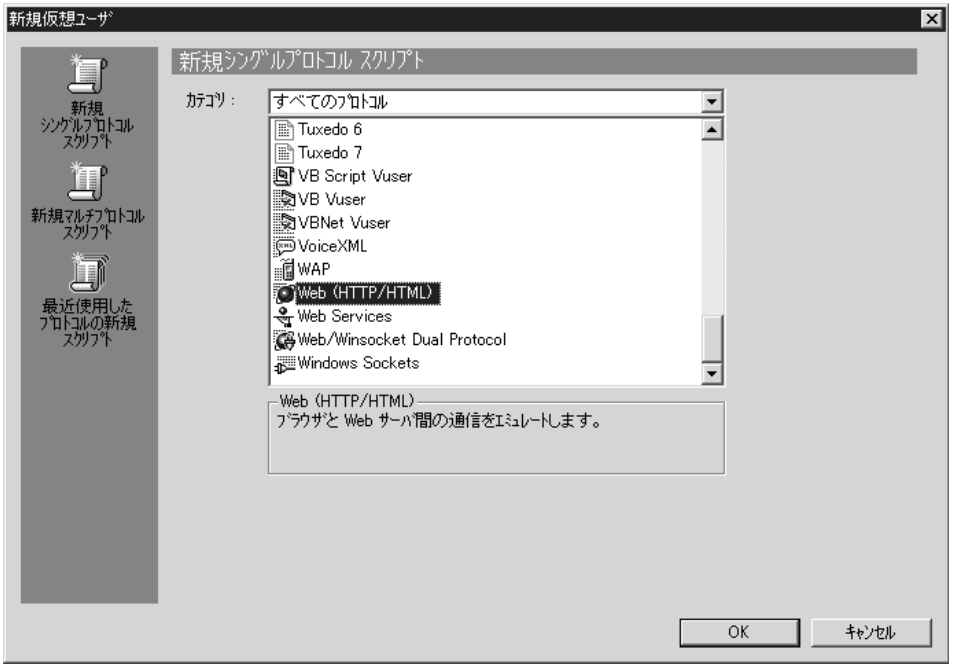
台のテスト用マシンだけで大きなサーバ負荷をエミュレートすることが可能になります。



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。VuGen で新しいテストを開くと，サポートされているすべてのプロトコルの一覧が表示されます。



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，それらの情報をアナリシス・ツールを使って表示できます。例えば 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。

仮想ユーザのタイプ

VuGen は，システムのエミュレートを可能にするさまざまな仮想ユーザ・テクノロジを備えています。それぞれのテクノロジは特定のアーキテクチャに対応しており，アーキテクチャごとに専用の仮想ユーザ・スクリプトが作成されます。例えば，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，Tuning Console セッション，またはビジネス・プロセス・モニタ・プロファイルを作成できます。

仮想ユーザのタイプは次のカテゴリに分類されます。

- ［**すべてのプロトコル**］: サポートされている全プロトコルのアルファベット順リスト。
- ［**アプリケーションの導入ソルーション**］: Citrix プロトコル用。
- ［**クライアント / サーバ**］: DB2 CLI，DNS，MS SQL，ODBC，Oracle（2-Tier），Sybase CTlib，Sybase DBlib，Windows Sockets プロトコル用。
- ［**ユーザ定義**］: C テンプレート，Visual Basic テンプレート，Java テンプレート，Javascript，VB Script，VB，VBNet タイプ・スクリプト用。
- ［**分散コンポーネント**］: COM/DCOM，CORBA-Java，RMI-Java プロトコル用。
- ［**e ビジネス**］: FTP，LDAP，Microsoft .NET，Palm，Web（GUI レベル），Web（HTTP/HTML），Web サービス，Web/WinSocket Dual プロトコル用。
- ［**Enterprise Java Beans**］: EJB テストおよび RMI-Java プロトコル用。
- ［**ERP/CRM**］: Baan，Oracle NCA，Oracle Web Applications 11i，Peoplesoft Enterprise，Peoplesoft-Tuxedo，SAP-Web，SAPGUI，デュアル SAPGUI/SAP-Web，Siebel（Siebel-DB2CLI，Siebel-MSSQL，Siebel-Oracle，Siebel-Web）プロトコル用。
- ［**レガシ**］: ターミナル・エミュレーション（RTE）用。
- ［**メール サービス**］: Internet Messaging（IMAP），MS Exchange（MAPI），Post Office Protocol（POP3），Simple Mail Protocol（SMTP）プロトコル用。
- ［**ミドルウェア**］: Jacada，Tuxedo（6，7）プロトコル用。
- ［**ストリーミング**］: Media Player（MMS）と Real プロトコル用。

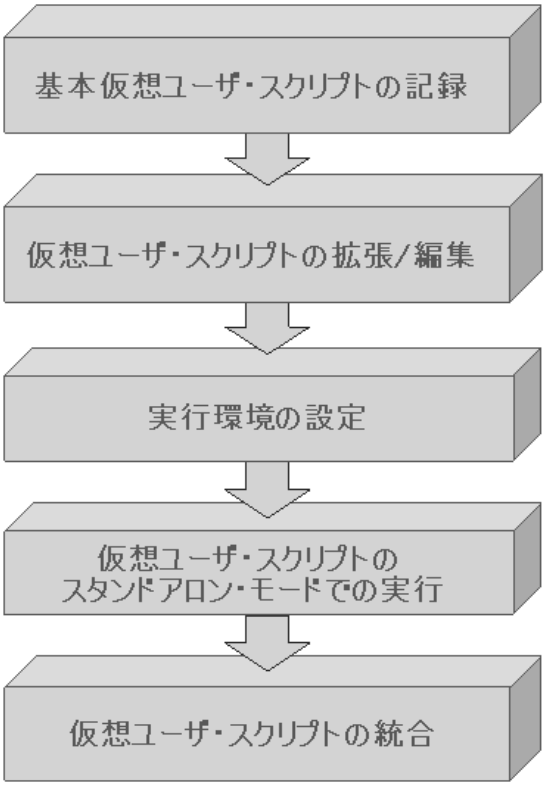
➤［**ワイヤレス**］: i モード，VoiceXML および WAP プロトコル用。

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

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 ランチャ（［**スタート**］＞［**プログラム**］＞［**Mercury LoadRunner**］＞［**LoadRunner**］）で［**設定**］＞［**LoadRunner ライセンス**］を選択します。

仮想ユーザ・スクリプトの作成手順

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，Mercury 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・ステートメントその他の Mercury API 関数を追加して，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，Tuning Module セッション，またはビジネス・プロセス・モニタ・プロファイル）に統合します。詳細については，『**Mercury LoadRunner コントローラ・ユーザーズ・ガイド**』，**Tuning Console**，**Performance Center**，または **Application Management** のユーザーズ・ガイドを参照してください。

本書の使用法

本書は，次の部で構成されています。

➤ **第 1 巻 - 仮想ユーザ・ジェネレータの使用**：この巻では，仮想ユーザ・ジェネレータと，全部またはほとんどすべてのプロトコルに適用される一般的な手順について説明します。また，技術の概要について説明した複数の付録が含まれています。外部関数の呼び出し，UNIX 環境でのプログラミング，多国語での作業，キーボード・ショートカットについて説明しています。

➤ **第 2 巻 - プロトコル**：この巻には，各仮想ユーザ・タイプについてプロトコル固有の情報が記載されています。

注：GUI 仮想ユーザ・スクリプトは WinRunner エンジンを利用します。これらのスクリプトは VuGen を使用して生成しないでください。詳細については，オンライン文書の「GUI 仮想ユーザ・スクリプトの作成」，および『**WinRunner ユーザーズ・ガイド**』を参照してください。

第 2 部

VuGen を使った作業

第 2 章

VuGen の紹介

仮想ユーザ・ジェネレータ（**VuGen**）を使って，さまざまなタイプのアプリケーションと通信プロトコルに対応した仮想ユーザ・スクリプトを作成できます。

本章では，次の項目について説明します。

- VuGen について
- VuGen の起動
- VuGen 環境オプションについて
- 環境オプションの設定
- 仮想ユーザ・スクリプトの表示と変更
- VuGen を使った仮想ユーザ・スクリプトの実行
- VuGen のコードについて
- C 仮想ユーザ関数の使用方法
- 関数のヘルプ

以降の情報は，GUI を除く全タイプの仮想ユーザ・スクリプトを対象とします。

VuGen について

仮想ユーザ・ジェネレータは，「**VuGen**」とも呼ばれ，仮想ユーザ・スクリプトの作成に使用される主要ツールです。

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。

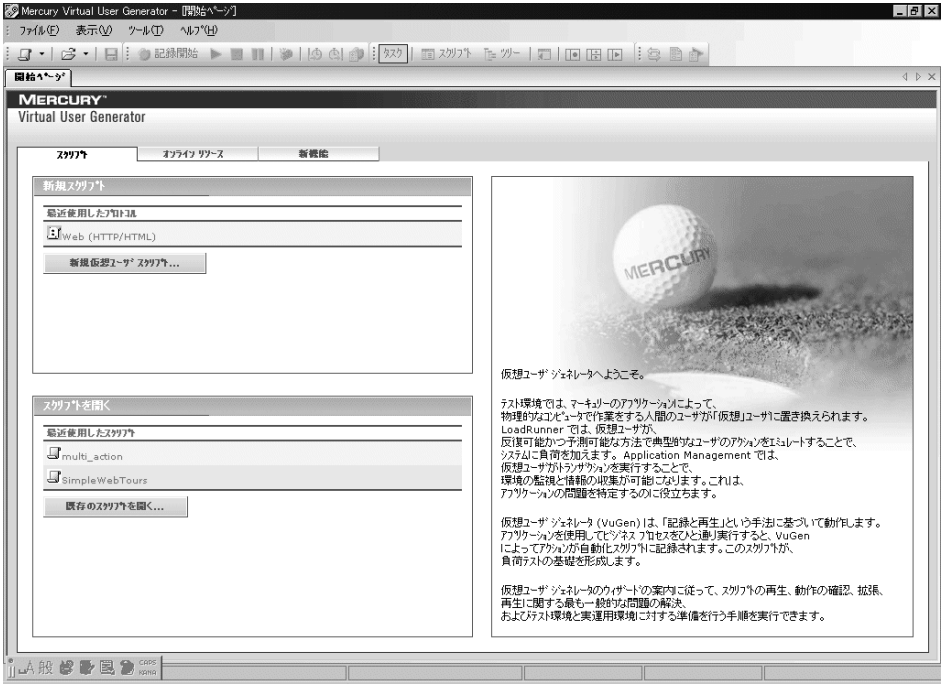
注：VuGen では Windows プラットフォームにおいてのみセッションを記録できます。しかし，記録した仮想ユーザ・スクリプトは，Windows および UNIX の両プラットフォームで実行できます。

仮想ユーザ・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 の起動

VuGen を起動するには，［スタート］メニューから［**スタート**］＞［**プログラム**］＞［**Mercury** ＜**アプリケーション名**＞］＞［**Applications**］＞［**Virtual User Generator**］を選択します。

仮想ユーザ・ジェネレータの開始ページが開きます。



最近使用したスクリプトを開くには，［**最近使用したスクリプト**］の一覧で目的のスクリプトをクリックします。

既存のスクリプトを開くには，最近の一覧ではなく，［**既存のスクリプトを開く**］をクリックします。

最近使用したプロトコルを使って新しいスクリプトを作成するには，［**最近使用したプロトコル**］の一覧で目的のプロトコルをクリックします。

一覧にないプロトコルを使って新しいスクリプトを作成するには，［**新規仮想ユーザ スクリプト**］をクリックします。

Zip アーカイブから既存の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。

個々のダイアログ・ボックスのヘルプ・トピックを参照するには，ダイアログ・ボックスの中をクリックして F1 キーを押します。

VuGen 環境オプションについて

VuGen 作業環境の自動回復，VuGen エディタ，および起動時の設定をカスタマイズできます。VuGen の［一般オプション］の［環境］タブで次のオプションを設定します。

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。

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 IntelliSense 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，プロトタイプが自動的に表示されます。関数構文の自動表示を VuGen 全体で有効にするには，このオプションの隣のチェック・ボックスを選択します。この機

能を無効にするには，［**関数構文の自動表示**］オプションの隣のチェック・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。

［**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，関数の一覧が表示されます。関数全部を手作業で入力しなくても正確な関数を選ぶことができます。単語の自動補完を VuGen 全体で有効にするには，このオプションの隣のチェック・ボックスを選択します。この機能を無効にするには，［**単語の自動補完**］オプションの隣のチェック・ボックスをクリアします。このオプションを 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。

［**フォントの選択**］: エディタに表示されるフォントを設定するには，［**フォントの選択**］をクリックします。［Font Configuration］ダイアログ・ボックスが表示されます。使用するフォント，スタイル，サイズを選択し，［**OK**］をクリックします。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。

標準設定を使用

標準設定では，［**関数構文の自動表示**］や［**単語の自動補完**］は VuGen 全体で有効になっています。自動復元は，10 分に設定されています。［仮想ユーザ ジェネレータへようこそ］ダイアログ・ボックスが開きます。

環境オプションの設定

環境関連オプションを設定するには，次の手順を実行します。

1 ［**ツール**］＞［**一般オプション**］を選択して，［環境］タブをクリックします。

2 現在のスクリプトの自動回復に関する情報を保存するには，［**自動回復情報を次の間隔で保存する**］オプションを選択して，設定を保存する間隔を分単位で指定します。

3 エディタに表示されるフォントを設定するには，［**フォントの選択**］をクリックします。［Font Configuration］ダイアログ・ボックスが表示されます。使用するフォント，スタイル，サイズを選択し，［**OK**］をクリックします。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。

4 VuGen の起動時に必ず［仮想ユーザ ジェネレータへようこそ］ダイアログ・ボックスが表示されるようにするには，［起動ダイアログ］セクションの［**起動ダイアログを表示する**］チェック・ボックスを選択します。

5 設定を承認するには［**OK**］をクリックします。［一般オプション］ダイアログ・ボックスが閉じます。

仮想ユーザ・スクリプトの表示と変更

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・ビューはすべての仮想ユーザ・タイプで使用できますが，ツリー・ビューとサムネイル・ビューは仮想ユーザのタイプによっては使用できません。例えば，サムネイル・ビューは Web，Citrix，および SAPGUI 仮想ユーザでのみ使用できます。

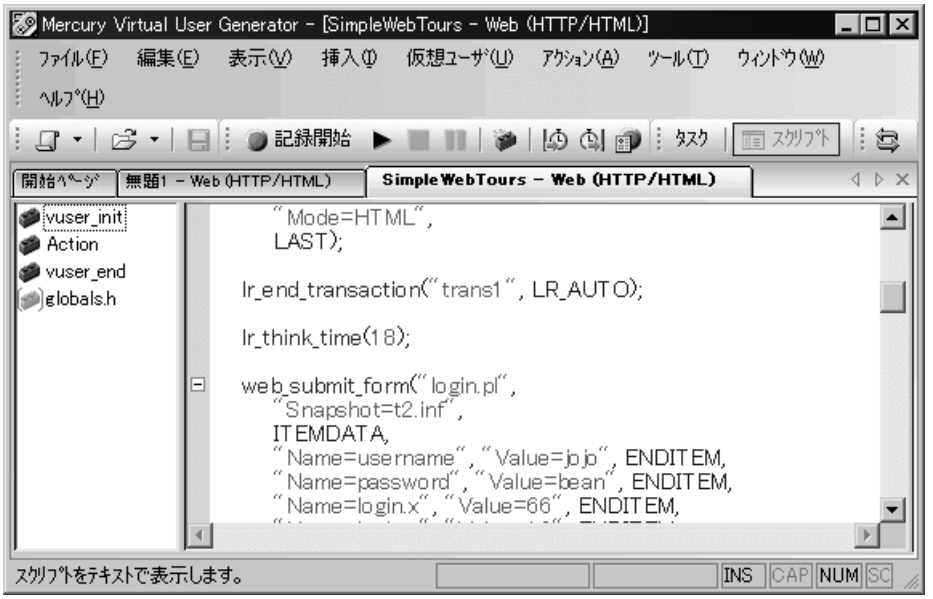
スクリプト・ビューでのコードの表示

スクリプト・ビューでは，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，スクリプトのプログラミングを行うことができます。

スクリプト・ビューを表示するには，次の手順を実行します。

VuGen のメイン・メニューから［**表示**］＞［**スクリプト ビュー**］を選択するか，［**スクリプトを表示**］アイコンをクリックします。スクリプトがテキスト

形式のスクリプト・ビューで表示されます。すでにスクリプト・ビューが表示されている場合は，このメニュー項目は選択できません。



⊞⊟

スクリプトの左側のマージンにあるプラスとマイナスの記号をクリックして，関数を展開したり折りたたんだりできます。これによって，スクリプトの表示を整理したり見やすくしたりできます。

スクリプト・ビューで作業を行うときは，［**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，関数を手作業で入力することもできます。詳細については，38 ページ「関数のヘルプ」を参照してください。

注：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。

ツリー・ビューでのスクリプトの表示

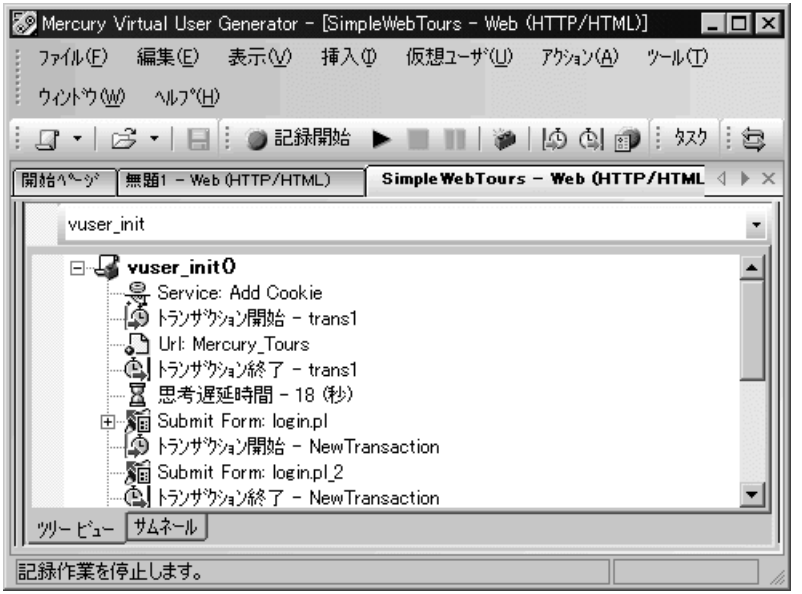
VuGen のツリー・ビューでは，仮想ユーザ・スクリプトがアイコン形式で表示され，各ステップが異なるアイコンで表されます。

ツリー・ビューを表示するには，次の手順を実行します。

VuGen のメイン・メニューから，［**表示**］＞［**ツリー ビュー**］を選択するか，［**ツリーを表示**］アイコンをクリックします。

仮想ユーザ・スクリプトの Actions セクションがアイコン形式のツリー・ビューで表示されます。異なるセクションを表示するには，そのセクションをツリーの上にあるドロップダウン・リストで選択します。

ツリー・ビューがすでに選択されている場合，このメニュー項目は無効になっています。



ツリー・ビュー内では，ステップをドラッグして望み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。

ツリー・ビューにステップを挿入するには，次の手順を実行します。

1 ステップをクリックします。

2 右クリック・メニューから［**前に挿入**］または［**後に挿入**］を選択します。［ステップの追加］ダイアログ・ボックスが開きます。

3 ステップを選択して［**OK**］をクリックします。［プロパティ］ダイアログ・ボックスが開きます。

4 プロパティを指定して［**OK**］をクリックします。現在のステップの前または後ろにステップが挿入されます。

スナップショットについて

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。ツリー・ビューで作業しているときは，選択したステップのスナップショットが VuGen の右側の表示枠に表示されます。スナップショットには，ステップ実行後のクライアント・ウィンドウが表示されます。

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ます。詳細については，第 47 章「記録後の仮想ユーザ・スクリプトの相関」を参照してください。

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をフル・サイズのウィンドウで表示します。

記録時と再生時のスナップショットを分割したウィンドウで表示します。

再生時のスナップショットをフル・サイズのウィンドウで表示します。

スナップショットの表示と非表示を切り替えるには，次の手順を実行します。

1 ツリー・ビューが表示されていることを確認します。表示されていなければ，ツリー・ビューに切り替えます（[**表示**] > [**ツリー ビュー**]）。

2 [**表示**] > [**スナップショット**] > [**スナップショットを表示**] を選択します。クライアント・ウィンドウのスナップショットが表示されます。スナップショットがすでに表示されている場合は，非表示になります。

3 [**表示**] > [**スナップショット**] の展開したメニューを使用して，記録時，および再生時のスナップショットを表示できます。また，ショートカット・ツールバー・ボタンを使用して必要なスナップショットを表示することもできます。

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。

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。

Citirx や SAPGUI など，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。

スナップショット・ファイル

スクリプトを再生するたびに，VuGen は拡張子 **.inf** を付けて**スクリプト**・ディレクトリにスナップショットを保存します。再生時のスナップショットは，結果セットごとにスクリプトの結果ディレクトリ（**Iteration1**，**Iteration2**，など）に格納されます。

Web 仮想ユーザのスナップショット・ファイルの名前を調べるには，スクリプト・ビューに切り替えます（[**表示**] ＞ [**スクリプト ビュー**]）。次の例では，スナップショット情報が **t1.inf** であることがわかります。

```
web_url("MercuryWebTours",
        "URL=http://localhost/MercuryWebTours/",
        "Resource=0",
        "RecContentType=text/html",
        "Referer=",
        "Snapshot=t1.inf",
        "Mode=HTML",
        LAST);
```

Citrix 仮想ユーザ・スクリプトでは，スナップショットがビットマップ・ファイルとしてスクリプトの **data\snapshots** ディレクトリに保存されます。スナップショット・ファイルの名前を調べるには，スクリプト・ビューで関数の引数を調べます。

```
ctrx_sync_on_window("ICA Administrator Toolbar", ACTIVATE, 768, 0, 33,
        573, "snapshot12", CTRX_LAST);
```

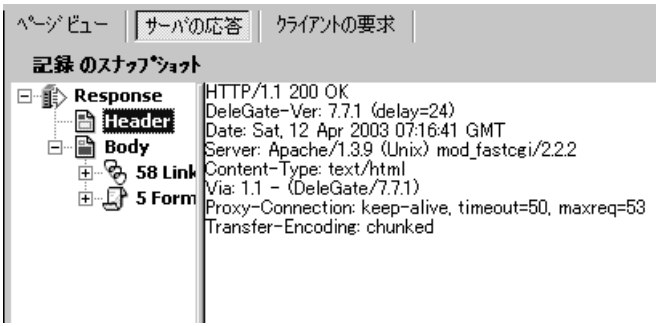
Web 仮想ユーザのスナップショット・タブ

Web 仮想ユーザのスナップショット・ウィンドウには，次のタブがあります。

[**ページ ビュー**]：ブラウザに表示されるとおりに，スナップショットを HTML で表示します。このボタンは，記録時および再生時の両方のスナップショットに対して使用できます。このビューを使って，正しいスナップショットを表示しているかどうかを確認できます。ただし，このビューでは，相関させる必要のある値は分かりません。

[**サーバの応答**]：スナップショットのサーバ応答 HTML コードを表示します。このタブは，記録時および再生時の両方のスナップショットに対して使用できます。[HTML] ビューでは，左の表示枠にスクリプトのツリー構造と，ドキュ

メントのコンポーネントであるヘッダーと本体，およびそのタイトル，リンク，フォームなどの詳細も表示されます。



［**クライアントの要求**］：スナップショットのクライアント要求 HTML コードを表示します。このタブは，記録時および再生時の両方のスナップショットに対して使用できます。［HTML］ビューでは，左の表示枠にスクリプトのツリー構造と，ドキュメントのコンポーネントであるヘッダーと本体，およびそのサブコンポーネントの詳細が表示されます。

スクリプトのサムネイルの表示

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。サムネイルは，ツリー・ビューに表示するか，トランザクション・エディタを使って表示します。

ツリー・ビューでのサムネイルの表示

ツリー・ビューでは，ツリー・ビュー・ウィンドウの下部に［**サムネイル**］タブが表示されます。

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。

注：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，希望の反復を選びます。

ツリー・ビューでサムネイルを表示するには，次の手順を実行します。

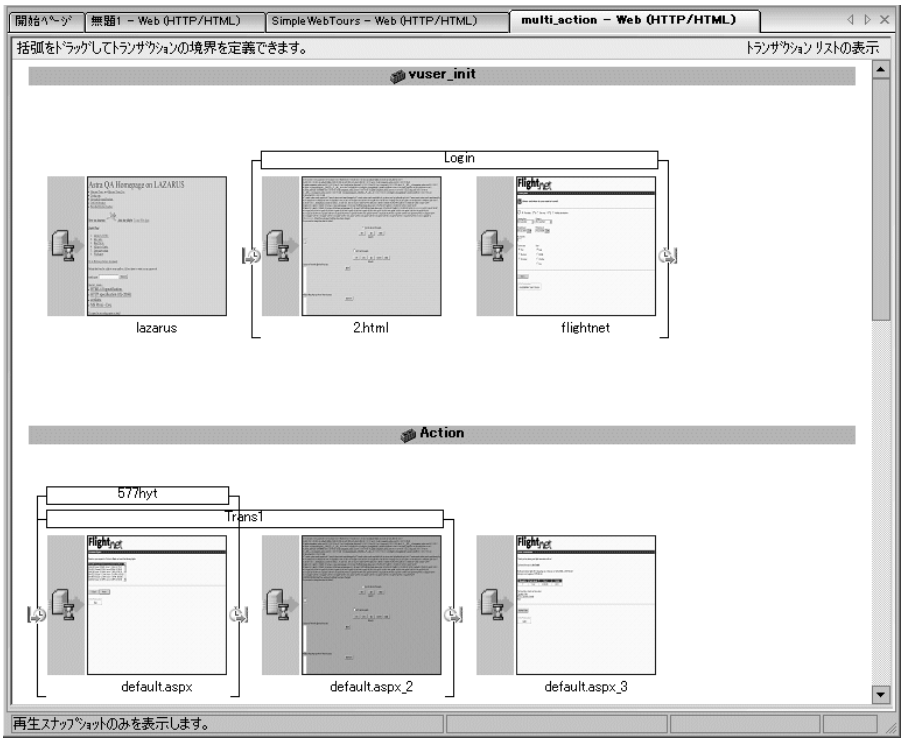
1 左側の表示枠の下部にある［**サムネール**］タブをクリックします。

2 目的のサムネイル画像をクリックすると，右側の表示枠にサムネイルのスナップショットが開きます。

3 大きな画像を表示するには，サムネイル画像をダブルクリックします。別のウィンドウが開き，スナップショットが大きく表示されます。

ワークフロー・ウィザードでのサムネイルの表示

トランザクション・エディタを使ってスナップショットを表示できます。このビューでは，サムネイルがアクション順に並べ替えられ，スクリプトの全ステップのサムネイルが横並びで表示されます。



トランザクション・エディタにサムネイルを表示するには，次の手順を実行します。

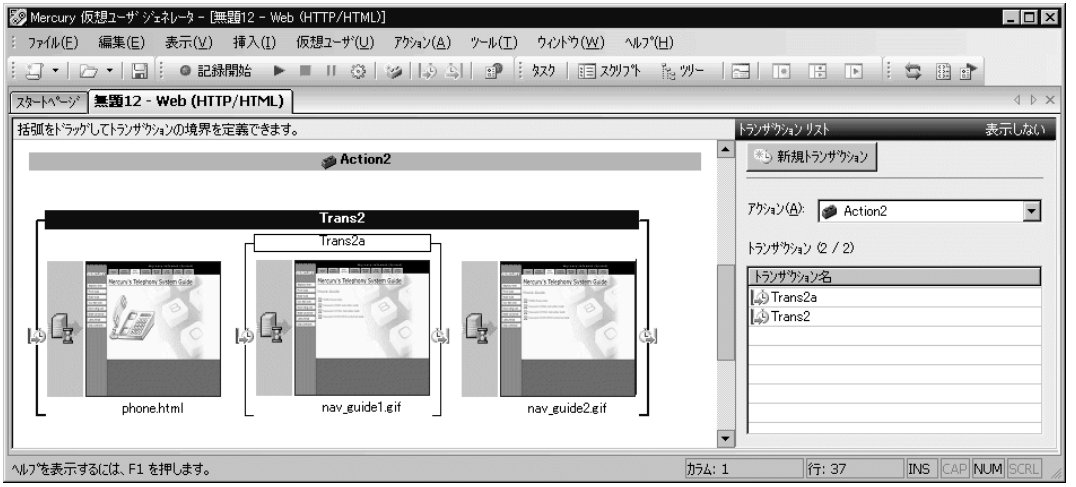
1 ツールバーの［**タスク**］をクリックして，タスク・リスト表示枠を開きます。

2 ［**拡張**］＞［**トランザクション**］リンクをクリックします。中央の表示枠と右側の表示枠にトランザクション・エディタが開きます。

より包括的に表示するには，［**タスク**］をクリックしてタスク・リストを非表示にします。

3 右側の表示枠で，表示するアクションを選択します。選択したアクションが表示されます。

次の例では，「**Action2**」が選択されています。



サムネイルを使った作業

VuGen では，サムネイル名の変更，サムネイルへの注釈の追加，および大きなサイズでのサムネイルの表示ができます。

サムネイルを大きなサイズで表示するには，次の手順を実行します。

1 右クリックして表示されるメニューから［**画像表示を拡大**］を選択するか，**Alt** キーを押しながら **F6** キーを押します。別のウィンドウが開き，スナップショットが大きく表示されます。

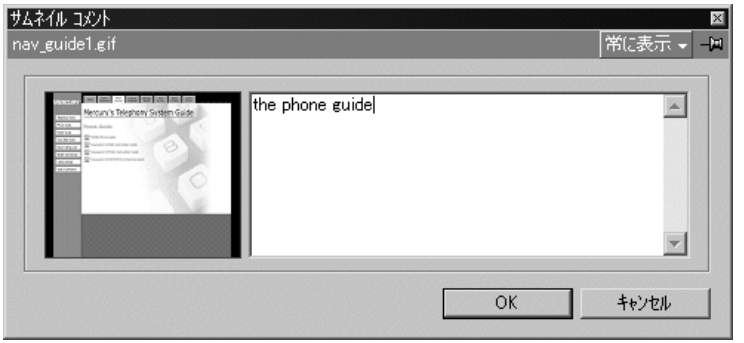
スナップショット名を変更するには，次の手順を実行します。

1 スナップショットを選択し，右クリックして表示されるメニューから［**名前の変更**］を選択するか，**F2** キーを押します。

2 必要なテキストを入力します。

スナップショットに注釈を追加するには，次の手順を実行します。

1 スナップショットを選択し，右クリックして表示されるメニューから［**コメント**］を選択するか，**Alt** キーを押しながら **F2** キーを押します。［サムネイル コメント］ダイアログ・ボックスが開きます。

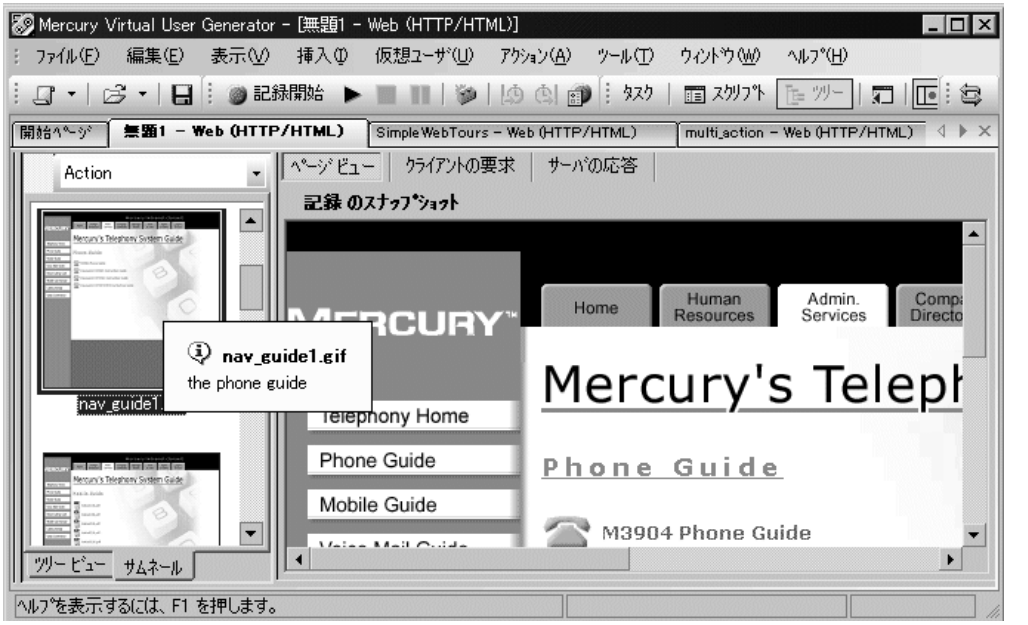


2［サムネイル コメント］ダイアログ・ボックスの右側の表示枠にテキストを入力します。

3［**OK**］をクリックして注釈を保存し，ダイアログ・ボックスを閉じます。

［サムネイル コメント］ダイアログ・ボックスを開いたままにして，テキストの追加や他のスナップショットでの作業を行うには，右上角にある［**常に表示**］を選択します。［**OK**］ボタンが［**適用**］に変わります。

スナップショットに注釈を挿入すると，サムネイルの右下隅に注釈があることを示す赤いマークが付きます。サムネイルの上にマウスを移動すると，注釈テキストのポップアップが表示されます。



VuGen を使った仮想ユーザ・スクリプトの実行

仮想ユーザ・スクリプトを使ってテストを実行したりアプリケーションを監視したりするには，スクリプトを LoadRunner のシナリオ，ビジネス・プロセス・モニタのプロファイル，または Tuning Console のセッション・ステップに組み込む必要があります。これを行う前に，VuGen でスクリプトを実行してその動作を検証します。詳細については，第 14 章「スタンドアロン・モードでの仮想ユーザ・スクリプトの実行」を参照してください。

スクリプトの再生が成功したら，LoadRunner のシナリオ，Performance Center の負荷テスト，Tuning Module のセッション，またはビジネス・プロセス・モニタのプロファイルなどの環境にスクリプトを組み込むことができます。詳細については，『**Mercury LoadRunner コントローラ・ユーザーズ・ガイド**』，**Tuning Console** のマニュアル，**Performance Center** のマニュアル，または **Application Management** のマニュアルを参照してください。

仮想ユーザ・スクリプトを実行する前に，実行環境の設定を変更できます。これらの設定の中には，仮想ユーザの反復実行回数や，スクリプトの実行時に仮想ユーザに適用される反復のペースおよび思考遅延時間も含まれます。実行環境の設定の詳細については，第 12 章「実行環境の設定」を参照してください。

仮想ユーザ・スクリプトを実行すると，スクリプトがインタプリタによって解釈および実行されます。スクリプトをコンパイルする必要はありません。スクリプトに変更を加えたときに，スクリプトに何らかの構文エラーが持ち込まれた場合には，そのエラーはインタプリタによって検出されます。スクリプトからインタプリタが認識して実行できる外部関数を呼び出すこともできます。詳細については，付録 A「外部関数の呼び出し」を参照してください。

上級ユーザは，記録されたスクリプトをコンパイルして実行プログラムを作成することもできます。詳細については，第 7 章「仮想ユーザ・スクリプトの拡張」を参照してください。

VuGen のコードについて

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 2 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

一般仮想ユーザ関数と**プロトコル固有の**関数によって Mercury VuGen 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** という接頭辞が付いているので 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。

➤ 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。例えば，**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。詳細については第 7 章「仮想ユーザ・スクリプトの拡張」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。

LR 関数については，34 ページ「C 仮想ユーザ関数の使用方法」の一覧を参照してください。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。

プロトコル固有の仮想ユーザ関数

一般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，例えばデータベース・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，Corba-Java/Rmi-Java 仮想ユーザには 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，第 5 章「スクリプト生成オプションの設定」を参照してください。

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。他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 C 言語の関数を使用する方法の詳細については，第 7 章「仮想ユーザ・スクリプトの拡張」を参照してください。Java スクリプトを変更する方法の詳細については，『**第 2 巻 - プロトコル**』の「Java 仮想ユーザ・スクリプトのプログラミング」を参照してください。

注：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。

C 仮想ユーザ関数の使用方法

任意の仮想ユーザ・スクリプトに **C** 仮想ユーザ関数を追加して，スクリプトを拡張できます。VuGen を使って記録しているときに生成される一般仮想ユーザ関数は 2 ～ 3 個です。残りの関数は，必要に応じて手作業でスクリプトにプログラミングしなければなりません。一般仮想ユーザ関数の詳細については，第 7 章「仮想ユーザ・スクリプトの拡張」を参照してください。

次の一覧に，ANSI C スクリプト用の一般 API 関数を示します。この中には，Java，VB，および GUI を除くすべてのプロトコルが含まれています。Java 関数の一覧については，『**第 2 巻 - プロトコル**』の「Java 仮想ユーザ・スクリプトのプログラミング」を参照してください。VB 関数の一覧については，「**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。

トランザクション関数

lr_end_sub_transaction	パフォーマンスを分析するため，サブトランザクションの最後に印を付けます。
lr_end_transaction	トランザクションの終了を示します。
lr_end_transaction_instance	パフォーマンスを分析するため，トランザクションのインスタンスの最後に印を付けます。
lr_fail_trans_with_error	開いているトランザクションのステータスを LR_FAIL に設定し，エラー・メッセージを送信します。
lr_get_trans_instance_duration	ハンドルで指定されたトランザクションのインスタンスの継続時間を取得します。
lr_get_trans_instance_wasted_time	トランザクションのインスタンスの浪費時間をハンドルによって取得します。
lr_get_transaction_duration	トランザクションの継続時間を名前によって取得します。
lr_get_transaction_think_time	トランザクションの思考遅延時間を名前によって取得します。

lr_get_transaction_wasted_time	トランザクションの浪費時間を名前によって取得します。
lr_resume_transaction	パフォーマンス分析のためのトランザクション・データ収集を再開します。
lr_resume_transaction_instance	パフォーマンス分析のためのトランザクションのインスタンスのデータ収集を再開します。
lr_set_transaction_instance_status	トランザクションのインスタンスのステータスを設定します。
lr_set_transaction_status	開いているトランザクションのステータスを設定します。
lr_set_transaction_status_by_name	トランザクションのステータスを設定します。
lr_start_sub_transaction	サブトランザクションの始まりを示します。
lr_start_transaction	トランザクションの開始を示します。
lr_start_transaction_instance	ネストしたトランザクションを親ハンドルで指定して開始します。
lr_stop_transaction	トランザクション・データの収集を停止します。
lr_stop_transaction_instance	トランザクションのハンドルで指定し，そのデータの収集を停止します。
lr_wasted_time	開いているすべてのトランザクションから浪費時間を除外します。

コマンド・ライン解析関数

lr_get_attrib_double	スクリプトのコマンド・ラインに指定された **double** 型の変数を取得します。

lr_get_attrib_long	スクリプトのコマンド・ラインに指定された **long** 型の変数を取得します。
lr_get_attrib_string	スクリプトのコマンド・ラインで使われている文字列を取得します。

情報関数

lr_user_data_point	ユーザ定義データのサンプルを記録します。
lr_whoami	仮想ユーザに関する情報を仮想ユーザ・スクリプトに返します。Application Management には適用されません。
lr_get_host_name	仮想ユーザ・スクリプトを実行しているホストの名前を返します。
lr_get_master_host_name	LoadRunner コントローラまたは Tuning Console を実行しているマシンの名前を返します。Application Management には適用されません。

文字列関数

lr_eval_string	パラメータを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。
lr_save_datetime	現在の日付および時刻をパラメータに保存します。
lr _advance_param	次に使用可能なパラメータに進みます。
lr _decrypt	暗号化された文字列を解読します。
lr_eval_string_ext	パラメータのデータが格納されているバッファへのポインタを取得します。
lr_eval_string_ext_free	**lr_eval_string_ext** 関数によって割り当てられたポインタを解放します。
lr_save_searched_string	バッファ内で文字列の出現を検索し，文字列出現に関連する部分のバッファをパラメータに保存します。

メッセージ関数

lr_debug_message	デバッグ・メッセージを［出力］ウィンドウまたはビジネス・プロセス・モニタのログ・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	メッセージをログ・ファイルに送ります。
lr_output_message	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_set_debug_message	デバッグ・メッセージ・クラスを設定します。
lr_vuser_status_message	形式を整えた出力を生成し，コントローラまたはコンソールの仮想ユーザ・ステータス領域に出力します。Application Management には適用されません。
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。

ランタイム関数

lr_load_dll	外部 DLL をロードします。
lr_peek_events	仮想ユーザ・スクリプトが一時停止できるポイントを示します。
lr_think_time	スクリプトの実行を一時停止して思考遅延時間（実際のユーザが動作の間に考えるために動作を停止する時間）をエミュレートします。
lr_continue_on_error	エラー処理方法を指定します。
lr_rendezvous	仮想ユーザ・スクリプトにランデブー・ポイントを設定します。Application Management には適用されません。

関数のヘルプ

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
- ヘッダー・ファイル

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や関数の例が含まれています。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。

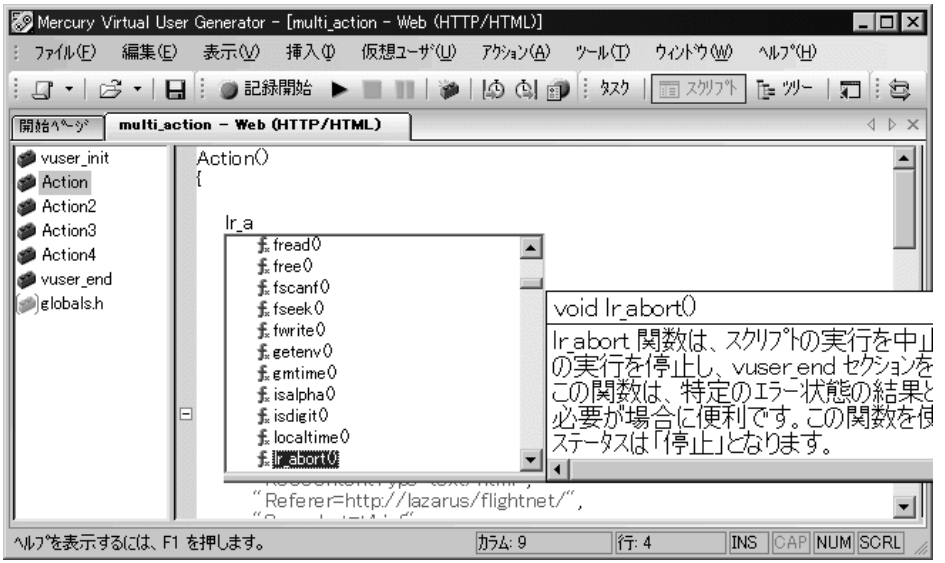
「**オンライン関数リファレンス**」を開くには，VuGen インタフェースから［**ヘルプ**］＞［**関数リファレンス**］を選択します。それからプロトコルを選び，カテゴリを選択します。

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，**IntelliSense** 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，その関数の接頭辞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。

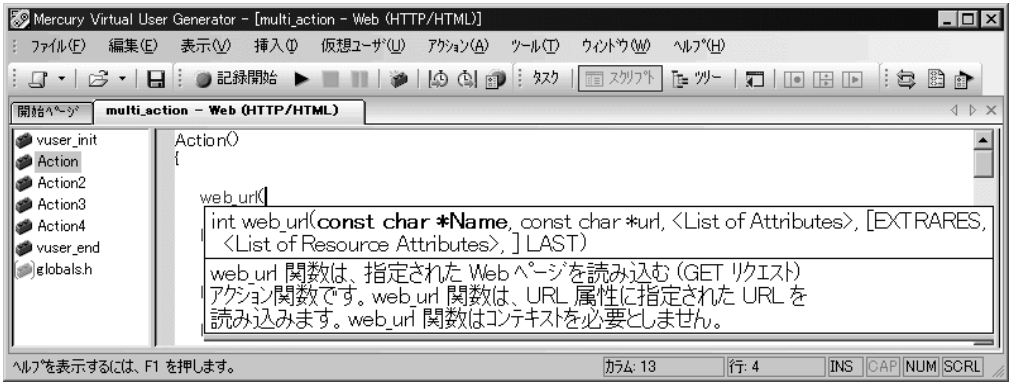


表示された関数を使用するには，その項目を選択するか，目的の項目までスクロールしてから選択します。選択した関数がカーソルの位置に挿入されます。リスト・ボックスを閉じるには，Esc キーを押します。

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。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。［**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 IntelliSense には，このほかにも**関数構文の自動表示**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。

ヘッダー・ファイル

関数のプロトタイプはすべてライブラリ・ヘッダー・ファイルに含まれています。ヘッダー・ファイルは，Mercury 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。ヘッダー・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，ヘッダー・ファイル名はプロトコルの接頭辞に対応しています。例えば，**lrd** という接頭辞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，対応するヘッダー・ファイルを示します。

プロトコル	ファイル
Citrix	**ctrxfuncs.h**
COM/DCOM	**lrc.h**
Database	**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**
RealPlayer	**lreal.h**
SAPGUI	**as_sapgui.h**
Siebel	**lrdsiebel.h**
SMTP	**mic_smtp.h**
ターミナル・エミュレータ	**lrrte.h**
Tuxedo	**lrt.h**
WAP WAP	**as_wap.h**
Web	**as_web.h**
Web サービス	**wssoap.h**
Windows Sockets	**lrs.h**

第 3 章

ワークフロー・ウィザードの使用

VuGen のワークフロー・ウィザードを使って，負荷テストや監視に使用できる仮想ユーザ・スクリプトを作成できます。

本章では，次の項目について説明します。

- ワークフロー・ウィザードについて
- タスク表示枠の表示
- ステップの記録
- スクリプトの検証
- スクリプトの拡張
- ロードの準備
- スクリプトの完了

以降の情報は，全タイプの仮想ユーザ・スクリプトを対象とします。

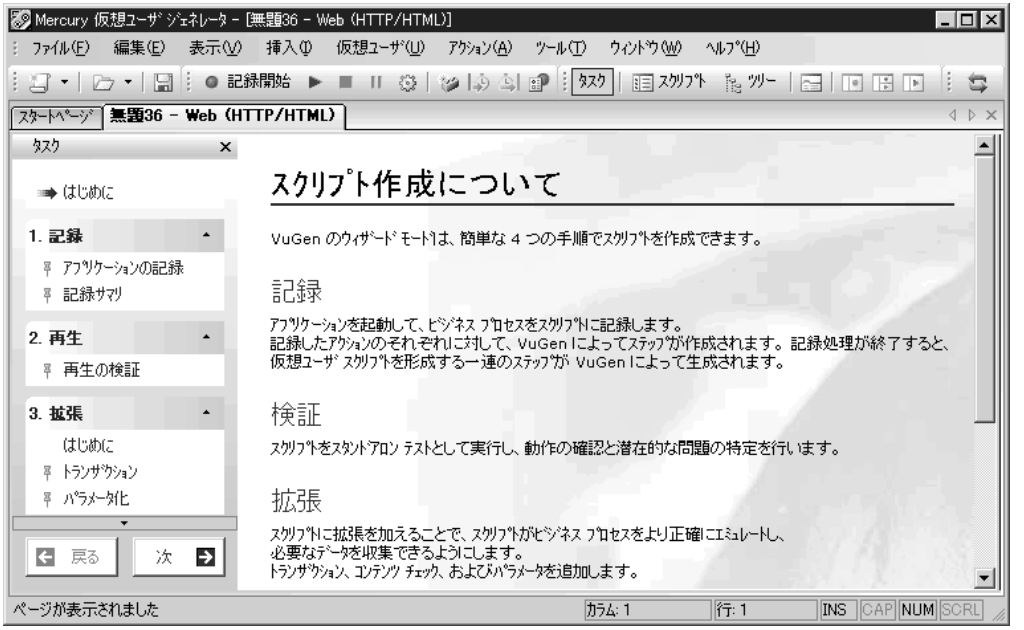
ワークフロー・ウィザードについて

VuGen のワークフロー・ウィザードでは，スクリプト作成における各工程が示されます。最初に基本スクリプトを作成し，それをテスト環境または実運用環境に適合します。

インストール後，標準設定では VuGen がワークフロー・ウィザード・ビューが表示された状態で開きます。ツリー・ビューやスクリプト・ビューでも作業できます。次回 VuGen を起動してスクリプトを開くと，VuGen の終了時に開いていたビューが開きます。タスク表示枠でタスクをクリックすると，ウィザード・ビューに戻ります。

タスク表示枠の表示

VuGen の左側の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。



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。

タスクのリストは，5 つのパートに分かれています。［**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**］，［**ロードの準備**］，［**終了**］です。

タスクの多くにはサブ・タスクがあります。次の表にサブ・タスクを表示します。

タスク	サブ・タスク
記録	アプリケーションの記録，記録サマリ（記録可能なプロトコル）
スクリプトの作成	スクリプトの作成，作成サマリ（Web Services，C の場合）
再生	再生の検証
拡張	はじめに，トランザクション，パラメータ化，内容チェック（Web 仮想ユーザ用）
ロードの準備	はじめに，反復，同時実行ユーザ

ステップの記録

［記録］セクション（Web Services，C，記録不能なプロトコルは除外）には，［アプリケーションの記録］と［記録サマリ］の 2 つのステップがあります。

アプリケーションの記録

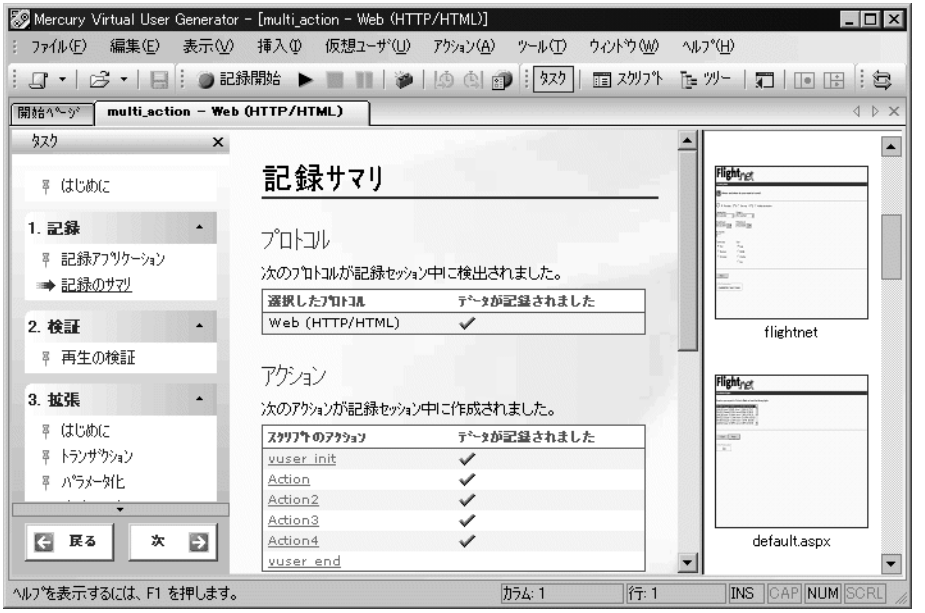
このウィザード・ステップでは記録プロセスを概説します。次のセクションが含まれます。

- ［はじめに］
- ［記録について］
- ［アクション］
- ［記録オプション］

記録サマリ

このウィザード・ステップでは，プロトコル情報とセッションが記録されるアクションを含む記録のサマリが提供されます。

このステップでは，記録されたスナップショットのサムネイルも提供されます。



スクリプトの検証

［再生］セクションには，［**再生の検証**］ステップが含まれます。スクリプトを再生すると，［**終了**］ステップの下の［**一般**］セクションにある［**再生のサマリ**］をクリックして，いつでも再生サマリを表示できます

再生の検証

このウィザード・ステップでは検証を概説します。次のセクションが含まれます。

- ［再生について］
- ［実行環境の設定］
- ［再生の前に］（再生中の検索対象）

再生のサマリ

このウィザード・ステップでは，再生のサマリが提供されます。エラーを一覧表示し，再生ログにエラーへのリンクをはります。

［再生サマリ］では，［記録と再生］のスナップショットのサムネイルも表示されます。スナップショットを目で見て比較し，不一致を調べることができます。

注：反復が複数の場合，［再生サマリ］ウィンドウには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**］＞［**ツリー ビュー**］を選択して，ツリー・ビューを切り替えます。［**表示**］＞［**スナップショット**］＞［**反復の選択**］を選択し，希望の反復を選びます。

スクリプトの拡張

［拡張］セクションには，トランザクション，パラメータ化，内容チェックという 3 つの主要なステップがあります。

トランザクション

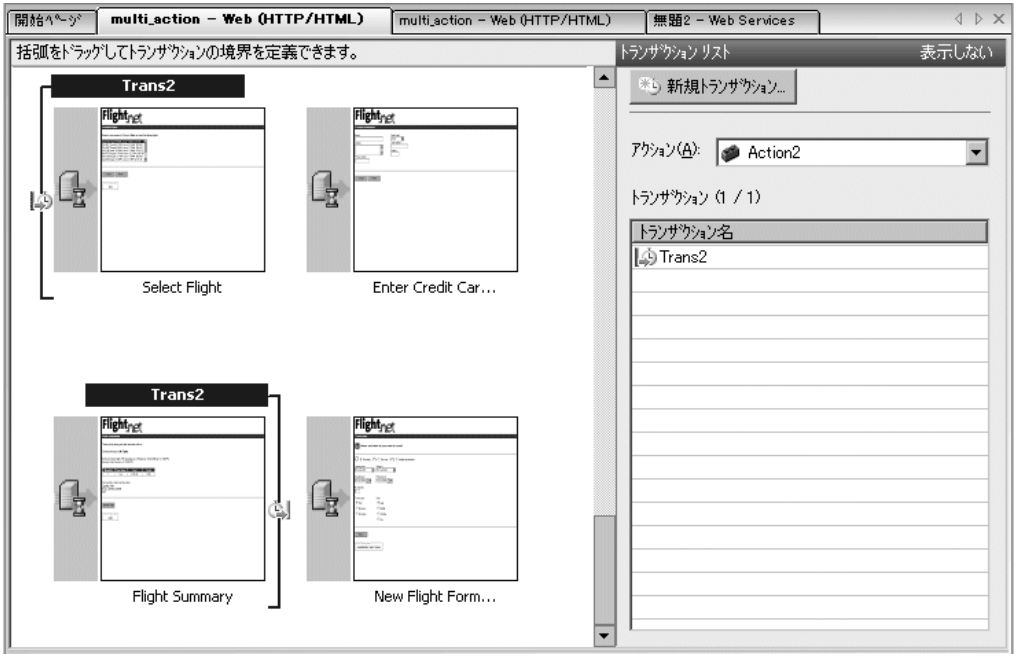
VuGen では，**トランザクション・エディタ**を使用して，スクリプトのサムネイル・ビューから直接トランザクションを追加および管理できます。

標準設定では，スクリプトの主要なステップのサムネイルのみが表示されます。スクリプト内のすべてのステップ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。

次の例では，**Trans2** は **Select Flight**，**Enter Credit Card**，**Flight Summary** という 3 つのステップを測定しています。

トランザクション・リストでの作業

トランザクション・エディタの右側の表示枠にあるトランザクション・リストに，スクリプト内のトランザクションのリストが表示されます。スクリプト内のトランザクションのリスト全体を表示することもできますし，特定のアクションのトランザクションのみを表示することもできます。



すべてのトランザクションを表示するには，［**アクション**］ボックスで［**全て**］を選択します。特定のアクションにおけるトランザクションを表示するには，［**アクション**］ボックスでアクション名を選択します。

トランザクションのリストの表示 / 非表示を切り替えるには，VuGen ウィンドウの右上隅にある［**表示しない**］またボタンは［**トランザクション リストの表示**］ボタンをクリックします。

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。次の項目は表示されません。

- 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

➤ エラーの生じたトランザクション

したがって，トランザクション・リストの上部に次のキャプションが表示されます。

トランザクション（2/4）。

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。

トランザクション・エディタでのトランザクションの定義

開始と終了のサムネイルに印を付けて，トランザクション・エディタ内のトランザクションを定義します。

トランザクションを定義するには，次の手順を実行します。

1 ［タスク］リストで［**トランザクション**］をクリックしてトランザクション・エディタを開きます。

2 サムネイル領域（中央の表示枠）で，トランザクションとして印を付けるステップまでスクロール・ダウンします。

ヒント：より多くのサムネイルを表示するには，ツールバーの［**タスク**］ボタンをクリックして，［タスク］リストを非表示にします。

---

3 単一のステップをトランザクションとして印を付けるには，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。

4 トランザクションとして複数のステップに印を付けるには，右クリック・メニューから［**新規トランザクション**］を選択するか，右側の表示枠で［**新規トランザクション**］ボタンをクリックします。

5 VuGen により，サムネイルの上部のステータス領域に手順が示されます。

［**ステップ 1/3 新規トランザクションの開始点を選択します。**］

トランザクションの最初のステップのサムネイルをクリックします。

［**ステップ 2/3 新規トランザクションの終了位置を選択します。**］

トランザクションの最後のステップのサムネイルをクリックします。

［**ステップ 3/3 新規トランザクションの名前を指定します。**］

トランザクションの最初のステップの上部の大括弧内に，直接トランザクション名を入力します。

トランザクションを完了するには，Enter キーを押します。

前述の手順の最中にトランザクションを終了するには，Esc キーを押します。

6 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。

7 トランザクションを追加するには，前述のステップを繰り返します。

8 トランザクション名を変更するには，右側の表示枠でタイトルを選択して，右クリック・メニューで［**名前変更**］を選びます。新しい名前を入力してください。

9 トランザクションを削除するには，右側の表示枠でタイトルを選択して，右クリック・メニューから［**削除**］を選びます。

トランザクション・エディタのガイドライン

トランザクション・エディタでトランザクションを作成および定義するには，これらのガイドラインに従います。

- トランザクションは単一のアクション内で開始し終了しなければなりません。トランザクションは複数のアクションにまたがって拡張できません。
- トランザクション名は，複数のアクションにおいてでも，スクリプト内で一意でなくてはなりません。
- 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。
- 既存のトランザクション内でトランザクションを作成できます。これらは「**入れ子**」のトランザクションと呼ばれます。

注：トランザクションを入れ子にする場合は，2 番目のトランザクションを 1 番目のトランザクションより前に閉じるか，同時に閉じる必要があります。そうしないと，正しく分析されません。

パラメータ化

［パラメータ化］画面には，スクリプト内のパラメータ化された値の概要が提供されます。パラメータ化のステップが示されます。

- パラメータ化する引数を検索します。
- パラメータに名前を付けます。
- パラメータのタイプを選択します。
- パラメータのタイプのプロパティを定義します。
- 引数をパラメータに置換します。

スクリプト内で引数をパラメータ化したら，［**拡張**］ステップに戻ることも，スクリプトを再生することもできます。

内容チェック

内容チェック・ウィザードのステップで，スクリプトの特定のテキストまたは内容をチェックできます。

［**テキスト チェック**］を使用して，再生中にテキスト文字列を検索できます。

［**内容チェック**］を使用して，サーバによって返されるテキストが厳密にわからない場合にも，VuGen が定義済みの規則を使って自動的にサーバの文字列を検索するようにできます。

ロードの準備

ワークフロー・ウィザードの 4 番目のステップは，システムにおいて負荷テストの実行し，マシンの応答と処理能力を検証するために需要です。このステップには次の 2 つの主要なステップがあります。

- 反復
- 同時ユーザ

反復

このウィザードのステップは，反復について説明し，［実行環境の設定］を開いて値を設定できるようにします。

反復回数を設定するには，次の手順を実行します。

1 ［実行環境の設定］（F4）を開きます。

2 ［実行論理］ノードを選択します。

3 反復回数を指定します。

同時ユーザ

このステップでは，LoadRunner コントローラを使用してシナリオを作成するプロセスが示されます。

シナリオでは，同時に実行するユーザの数を指定して，複数ユーザを稼動しているシステムの振る舞いを観察できます。

スクリプトの完了

ワークフロー・ウィザードの最終ステップは［**終了**］です。

［終了］には，次のセクションが含まれます。

- ［**シナリオの作成**］：LoadRunner コントローラを使用してシステムを対象に負荷テストを実行します。
- ［**Performance Center へのアップロード**］：パフォーマンス・センタ・サーバのインストール中にテストを実行します。
- ［**Quality Center へのアップロード**］：テスト・レポジトリにテストを追加します。

第 4 章

VuGen を使った記録

VuGen はクライアント・アプリケーションとサーバの間の通信を記録することによって，仮想ユーザ・スクリプトを作成します。

本章では，次の項目について説明します。

- VuGen を使った記録について
- 仮想ユーザ・スクリプトのセクション
- 仮想ユーザ・スクリプトの新規作成
- プロトコルの追加と削除
- 仮想ユーザ・カテゴリの選択
- スクリプトの新規作成
- 既存のスクリプトを開く
- アプリケーションの記録
- 記録セッションの終了と保存
- 記録ログの表示
- Zip ファイルの使用
- アクションのインポート
- 認証情報の提供
- 仮想ユーザ・スクリプトの再生成

以降の情報は，GUI を除く全タイプの仮想ユーザ・スクリプトを対象とします。

VuGen を使った記録について

VuGen は，クライアント・アプリケーションでのアクション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。記録されたスクリプトを実行すると，仮想ユーザはクライアントとサーバ間のユーザ操作をエミュレートします。

作成する各仮想ユーザ・スクリプトには，少なくとも次の 3 つのセクションがあります。**vuser_init**，1 つ以上の **Actions**，および **vuser_end** です。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。一般的には，サーバへのログインを **vuser_init** セクションに，クライアントの動作を **Actions** セクションに，ログオフの手順を **vuser_end** セクションに記録します。

テストを作成した後，テストを Zip アーカイブに保存し，電子メールの添付ファイルとして送信できます。

記録中，スクリプトにトランザクション，コメント，ランデブー・ポイントを挿入できます。詳細については，第 7 章「仮想ユーザ・スクリプトの拡張」を参照してください。

仮想ユーザ・スクリプトのセクション

各仮想ユーザ・スクリプトには，少なくとも次の 3 つのセクションがあります。**vuser_init**，**1 つ以上の Actions**，および **vuser_end** です。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，第 12 章「実行環境の設定」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他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，『**第 2 巻 - プロトコル**』の「Java 仮想ユーザ・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;}
}
```

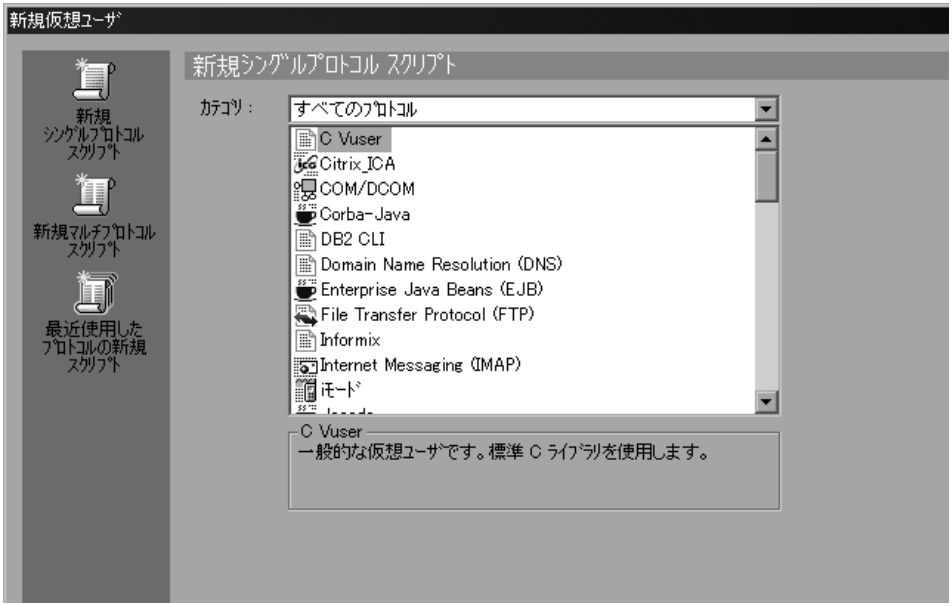
注：Oracle DB のトランザクション・ブレークダウンは，vuser_init セクションで記録されたアクションには使用できません。

仮想ユーザ・スクリプトの新規作成

VuGen では，シングル・プロトコル・モードまたはマルチ・プロトコル・モードで記録することによって，スクリプトを新規作成できます。

［**新規作成**］をクリックすると，いつでも［新規仮想ユーザ］ダイアログ・ボックスが開きます。このダイアログ・ボックスには次のものへのショートカットがあります。

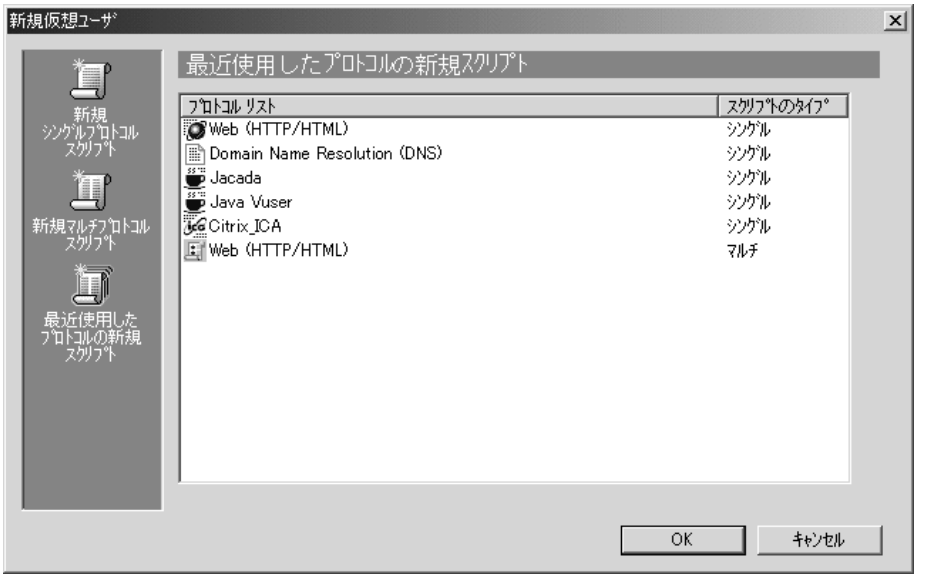
［**新規シングル プロトコル スクリプト**］：シングル・プロトコルの仮想ユーザ・スクリプトを作成します。これは［仮想ユーザ ジェネレータへようこそ］ダイアログ・ボックスが開いたときの標準のオプションです。シングル・プロトコルのスクリプトを作成するには，［カテゴリ］リストからカテゴリを選び（64 ページ「仮想ユーザ・カテゴリの選択」を参照），そのカテゴリ配下のプロトコル・リストでプロトコルを選択します。



［**新規マルチ プロトコル スクリプト**］：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。VuGen には使用可能なすべてのプロトコルが表示され，どのプロトコルを記録するかを指定できます。マルチ・プロトコルのスク

リプトを作成するには，［利用可能なプロトコル］セクションでプロトコルを選択し，右矢印をクリックしてプロトコルを［選択されたプロトコル］セクションに移動します。

［**最近使用したプロトコルの新規スクリプト**］：仮想ユーザ・スクリプトの新規作成に使用された最近のプロトコルを一覧表示し，それらがシングル・プロトコルかマルチ・プロトコルかを示します。リストからプロトコルを選択して［**OK**］をクリックすると，そのプロトコルに対応するスクリプトが新規作成されます。



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，次のプロトコルをサポートします。

COM，FTP，IMAP，Oracle NCA，POP3，RealPlayer，Window Sockets（raw），SMTP，Web

Dual protocol Web/Ws のエンジンは，異なるメカニズムを使用しており，シングル・プロトコルとして扱う必要があります。このプロトコルを他のマルチ・プロトコル・タイプと組み合わせることはできません。

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，次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。

Oracle NCA，Web，RTE，General（C Vuser），WAP，i モード，VoiceXML

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，CORBA-Java，RMI-Java，Web，WAP，i モード，VoiceXML，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，EJB）の記録の詳細については，『**第 2 巻 - プロトコル**』の「Java 言語仮想ユーザ・スクリプトの記録」，または各プロトコルを解説している章を参照してください。

プロトコルの追加と削除

マルチ・プロトコル・セッションを記録する前に，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。スクリプトの作成時に特定のプロトコルを指定した場合は，プロトコルの記録オプションを使用してそれらを有効または無効にできます。

記録オプションを開くには，［**ツール**］＞［**記録オプション**］を選択するか，または Ctrl キーを押しながら F7 キーを押します。［**一般**：**プロトコル**］ノードを選択します。

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。

仮想ユーザ・カテゴリの選択

仮想ユーザのタイプは次のカテゴリに分類されます。

- ［**すべてのプロトコル**］: サポートされている全プロトコルのアルファベット順リスト。
- ［**アプリケーションの導入ソリューション**］: Citrix プロトコル用。
- ［**クライアント / サーバ**］: MS SQL，ODBC，Oracle（2 層），DB2 CLI，Sybase Ctlib，Sybase Dblib，Windows Sockets，および DNS プロトコル用。
- ［**ユーザ定義**］: C テンプレート，Visual Basic テンプレート，Java テンプレート，JavaScript，VBScript タイプ・スクリプト用。
- ［**分散コンポーネント**］: COM/DCOM，CORBA-Java，RMI-Java プロトコル用。
- ［**e ビジネス**］: FTP，LDAP，Palm，Web （HTTP/HTML），Web サービス，Web/WinSocket Dual プロトコル用。

- ［**Enterprise Java Beans**］：EJB テストおよび RMI-Java プロトコル用。
- ［**ERP/CRM**］：Baan，Oracle NCA，Oracle Web Applications 11i，Peoplesoft Enterprise，Peoplesoft-Tuxedo，SAP-Web，SAPGUI，SAPGUI/SAP-Web Dual，Siebel（Siebel-DB2 CLI，Siebel-MSSQL，Siebel-Web，Siebel-Oracle）プロトコル用。
- ［**レガシ**］：ターミナル・エミュレーション（RTE）用。
- ［**メール サービス**］：Internet Messaging（IMAP），MS Exchange（MAPI），POP3，SMTP プロトコル用。
- ［**ミドルウェア**］：Jacada，Tuxedo（6，7）プロトコル用。
- ［**ストリーミング**］：RealPlayer プロトコル用。
- ［**ワイヤレス**］：i モード，VoiceXML，WAP プロトコル用。

スクリプトの新規作成

本項では，VuGen を起動してスクリプトを新規作成する方法について説明します。

新しい仮想ユーザ・スクリプトを作成するには，次の手順を実行します。

1 ［**スタート**］＞［**プログラム**］＞［**Mercury ＜アプリケーション名＞**］＞［**Applications**］＞［**Virtual User Generator**］を選択して，VuGen を起動します。［仮想ユーザ ジェネレータへようこそ］画面が表示されます（VuGen を最後に実行したときに［仮想ユーザ ジェネレータへようこそ］画面が表示されないように設定していない場合）。

2 シングル・プロトコル・スクリプトを作成するには，［カテゴリ］リストからプロトコルを 1 つ選択します。

3 1 つの記録セッションで 2 つ以上のプロトコルを記録できるマルチ・プロトコル・スクリプトを作成するには，左側の表示枠の［**新規マルチ・プロトコル スクリプト**］ボタンをクリックして［新規マルチ・プロトコル スクリプト］ウィンドウを表示します。

使用するプロトコルを［**利用可能なプロトコル**］リストから選択します。右方向の矢印をクリックして，選択したプロトコルを［**選択されたプロトコル**］リストに移動します。使用するすべてのプロトコルについて，この手順を繰り返します。

注：特定の Oracle NCA アプリケーションを記録する場合は，**Web（HTTP/HTML）**ではなく **Oracle NCA** を選択します)。詳細については，『**第 2 巻 - プロトコル**』の「Oracle NCA 仮想ユーザ・スクリプトの作成」を参照してください。

4［**OK**］をクリックしてダイアログ・ボックスを閉じ，仮想ユーザ・スクリプトの生成を開始します。

既存のスクリプトを開く

ローカル・マシンまたはネットワーク上にすでにスクリプトがある場合は，そのスクリプトを変更して追加のアクションを記録できます。

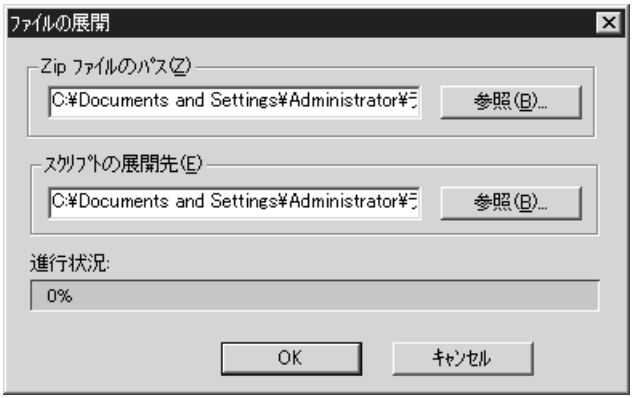
既存のスクリプトを開くには，次の手順を実行します。

1 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。

2 Quality Center リポジトリからファイルを開く方法については（LoadRunner の場合のみ)，235 ページ「Quality Center プロジェクトからスクリプトを開く」を参照してください。

3 圧縮 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。



4 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。

アプリケーションの記録

ほとんどの仮想ユーザ・スクリプト・タイプでは，新しいスクリプトの作成を開始すると，自動的に［記録開始］ダイアログ・ボックスが表示されます。

記録を開始するには，次の手順を実行します。

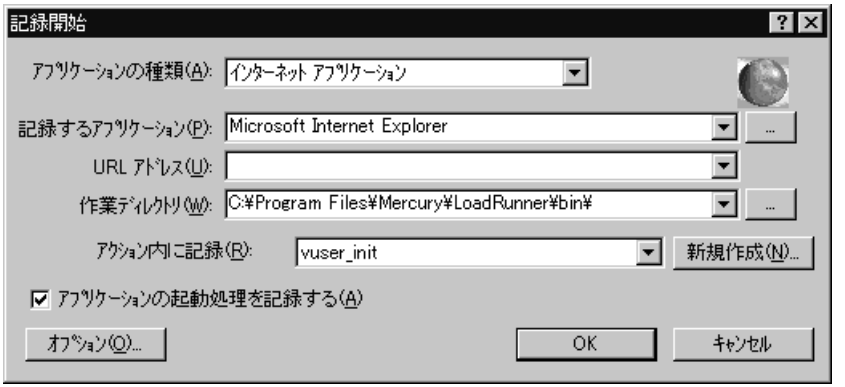
1 ［記録開始］ダイアログ・ボックスが開かない場合は，［**記録開始**］ボタンをクリックします。［記録開始］ダイアログ・ボックスが開きます。このダイアログ・ボックスはプロトコルごとに多少異なります。

2 ほとんどのクライアント / サーバ・プロトコルでは，次のダイアログ・ボックスが表示されます。

記録するアプリケーション，作業ディレクトリ（任意），およびアクションを入力します。必要があれば，［**オプション**］をクリックして記録オプションを設定します。

3 インターネット以外のアプリケーションの場合は，アプリケーションの種類を選択します。アプリケーションの種類には，Win32 アプリケーションおよびインターネット・アプリケーションがあります。例えば，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。Citrix ICA 仮想ユーザの場合は，VuGen によって Citrix クライアントが自動的に記録されるため，［**アクション内に記録**］ボックスにアクションを指定するだけで済みます。

4 インターネット・アプリケーションについて，適切な情報を入力します。



［**記録するアプリケーション**］：記録するブラウザまたはインターネット・アプリケーションを選択します。

［**URL アドレス**］：開始する URL アドレスを指定します。

［**作業ディレクトリ**］：作業ディレクトリを指定する必要があるアプリケーションの場合は，ここで指定します。必要となる情報は，仮想ユーザ・スクリプトのタイプによって異なります。

5 Win32 アプリケーションについて，適切な情報を入力します。

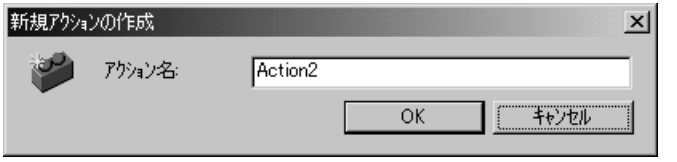
［**記録するアプリケーション**］：記録する Win 32 アプリケーションを入力します。

［**プログラム引数**］：上記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。例えば，**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**］：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。

6 ［**アクション内に記録**］ボックスで，記録を挿入するセクションを選択します。最初に選択できるセクションは **vuser_init**，**Action**，および **vuser_end** です。マルチアクション（Oracle NCA，Web，RTE，C Vusers，WAP，i-Mode，VoiceXML）をサポートするシングル・プロトコル仮想ユーザ・スクリプトの場合は，［**アクション**］＞［**新規アクションを作成**］を選択して新しいセクションを追加し，新しいアクションの名前を指定できます。



7 アプリケーションの起動を記録するには，［**アプリケーションの起動処理を記録する**］を選択します（Java タイプの仮想ユーザ・スクリプトには適用されません）。アプリケーションの起動を記録しない場合は，このチェック・ボックスをクリアします。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。

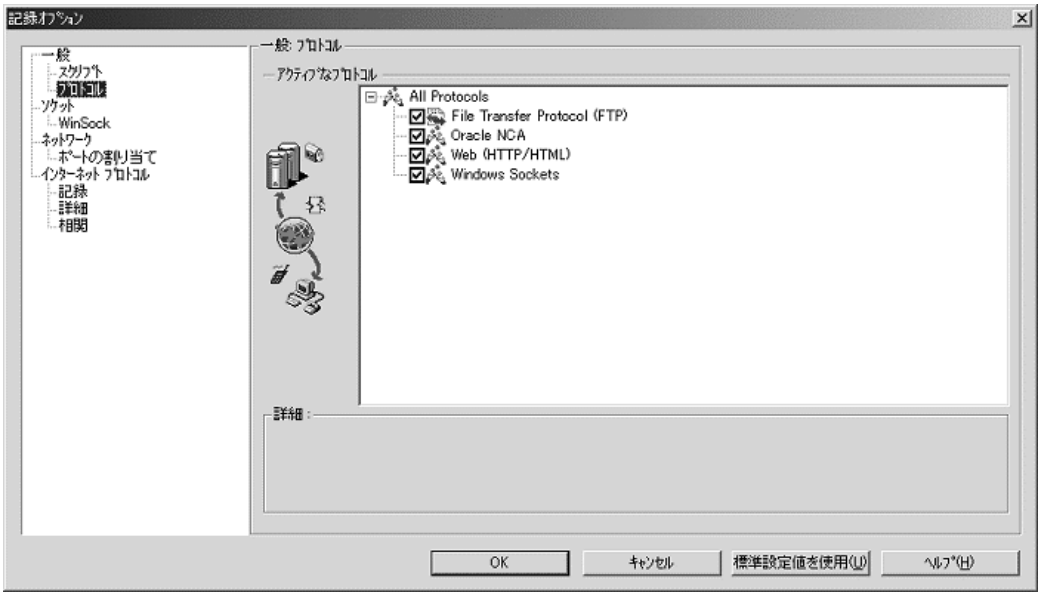
8 ［**オプション**］または［**記録オプションの編集**］ボタンをクリックして［記録オプション］ダイアログ・ボックスを開き，記録オプションを設定します。使用可能なオプションは記録するプロトコルによって異なります。詳細については，対応する各章を参照してください。

9 コード生成のための言語を選択し，スクリプトに関するオプションを設定するには，［**スクリプト**］ノードをクリックします。詳細については，第 5 章「スクリプト生成オプションの設定について」を参照してください。

10 ポート情報を指定するには，［**ポートの割り当て**］ノードをクリックします。これは，非標準ポートの SSL アプリケーションを記録する場合に有用です。ポートのリストを確認します。使用するポートがリストにない場合は，［ポー

トの割り当て］オプションを使用して情報を指定できます。詳細については，第 6 章「ポートの割り当て設定について」を参照してください。

11 マルチ・プロトコル仮想ユーザ・スクリプトのみ：記録するプロトコルのリストを変更するには，［**プロトコル**］ノードをクリックします。ノードを展開し，必要なプロトコルを選択します。



これで，記録開始の準備が整います。

12［**OK**］をクリックしてダイアログ・ボックスを閉じ，記録を開始します。

13［**アプリケーションの起動処理を記録する**］チェック・ボックスがクリアしてある場合，［記録の一時停止］ダイアログ・ボックスが表示されます。操作を進めて，記録を開始したい位置に到達したら，［**記録**］をクリックします。記録を行わない場合は，［**中止**］をクリックします。

14 VuGen によってアプリケーションが起動され，フローティング・ツールバーが開きます。

アプリケーション内で，標準的な操作を実行します。操作と同時に，VuGen によって仮想ユーザ・スクリプトが選択されたアクション・セクションに挿入されます。記録中にセクションを切り替えるには，フローティング・ツールバーを使います。

使用しているアプリケーションまたはサーバで認証を行う必要がある場合は，ユーザ名とパスワードを入力するよう VuGen から求められます。認証の詳細については，該当する項を参照してください。

記録セッションの終了と保存

一般的なビジネス・プロセスを記録したら，ビジネス・プロセスの終了ステップを実行し，仮想ユーザ・スクリプトを保存して，セッションの記録を完了します。

記録を終了するには，次の手順を実行します。

1 フローティング・ツールバーで **vuser_end** セクションに切り替えて，ログオフまたはクリーンアップ処理を実行します。

2 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。VuGen エディタに，記録されたすべてのステップが表示されます（スクリプト・ビューで作業を始めている場合は，記録された関数が表示されます)。

3［**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。［テストを保存］ダイアログ・ボックスが表示されます（新規仮想ユーザ・スクリプトの場合のみ)。スクリプト名を指定します。**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。

4 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。

保存するファイルを指定します。実行可能ファイルだけを保存するには，［**圧縮するファイル**］セクションで［**実行可能ファイル**］を選択します。標準設定では，すべてのファイルがアーカイブに保存されます。

圧縮率を［最高（最低速)］，［標準］，［高速］，［超高速］，または［なし］から選択します。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。

［**OK**］をクリックします。

5 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。

［**OK**］をクリックします。電子メールの作成フォームが表示されます。

電子メール・アドレスを入力してメールを送信します。

6 Performance Center のユーザは，サーバ上のリポジトリにファイルをアップロードできます。ファイルをアップロードするには，［**ファイル**］＞［**Performance Center サーバへのアップロード**］を選択します。ダイアログ・ボックスが開きます。

➤［**URL**］ボックスに，Performance Center の URL を入力します。

➤［**Project**］ボックスで，プロジェクト名を選択します。

➤ サーバにログオンするためのユーザ名とパスワードを入力します。

➤［**OK**］をクリックして設定を適用し，ダイアログ・ボックスを閉じます。

記録ログの表示

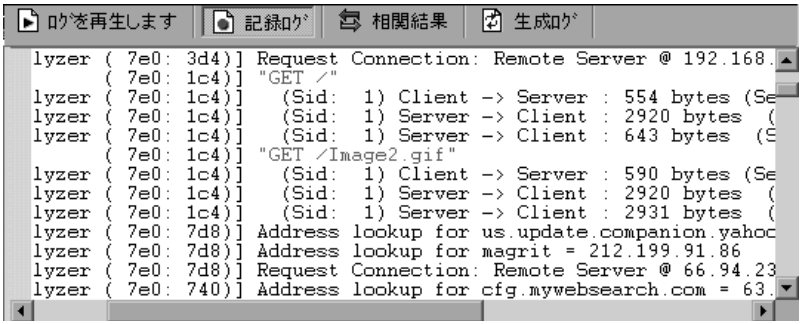
記録終了後に，**vuser_init**，**Actions**，および **vuser_end** セクションの内容を VuGen スクリプト・エディタに表示できます。アクションを表示するには，左側の表示枠でアクション名を選択します。

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セットアップ情報が含まれます。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。

ログを下部のウィンドウに表示することで，記録とスクリプト生成に関する情報を確認できます。出力ウィンドウを開くには，［**表示**］＞［**出力ウィンドウ**］を選択し，［記録ログ］タブまたは［生成ログ］タブを選択します。

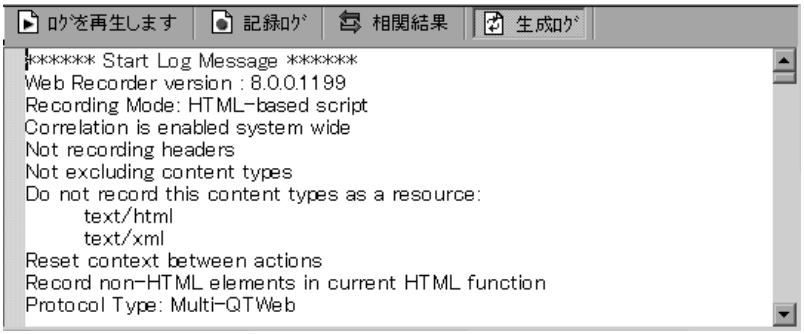
記録ログ

記録中に発行されたメッセージのログを表示するには，［**記録ログ**］タブを選択します。［記録オプション］ダイアログ・ボックスの［**詳細**］タブで，このログの詳細レベルを設定できます。



生成ログ

コード生成に使用されたスクリプトの設定の概要を表示するには，［**生成ログ**］タブを選択します。ここでは，レコーダのバージョンや記録オプションの値その他の追加情報が表示されます。



Zip ファイルの使用

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。

Zip ファイルからのインポート

圧縮 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 ファイルからの作業

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 ファイルへのエクスポート

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。標準設定では，すべてのファイルがアーカイブに保存されます。実行可能ファイルだけを保存するには，［**実行可能ファイル**］オプションを選択します。

また，**圧縮率**として，［最高］，［標準］，［高速］，［超高速］，または［なし］を選択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。したがって，［**最高（最低速）**］の圧縮オプションが最も低速になります。

Zip の作成と電子メール送信

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。

アクションのインポート

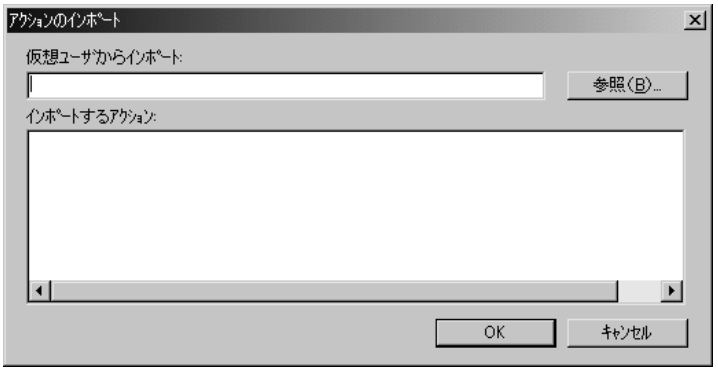
複数のアクションをサポートする仮想ユーザ・タイプの場合，別の仮想ユーザ・スクリプトから現在のスクリプトにアクションをインポートできます。インポートできるのは，同じタイプの仮想ユーザのアクションだけです。インポートされたアクションに関連付けられているパラメータは，スクリプトに組み込まれます。以下のオプションが使用できます。

［**仮想ユーザからインポート**］: インポート元の仮想ユーザ・スクリプトを入力または参照します。

［**インポートするアクション**］: インポートするアクションを選択します。

現在のスクリプトへアクションをインポートするには，次の手順を実行します。

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。

4 アクションの順序を並べ替えるには，最初にアクションの順序の並べ替えを有効にしておく必要があります。アクションを右クリックして［**アクションの順序を並べ替え**］を選択します。アクションをドラッグして順序を並べ替えます。アクションを VuGen の左側の表示枠で並べ替えても，それらの実行順序には影響はありません。実行順序を変更するには，実行環境の設定の［ペースの

設定］ノードを使用します。詳細については，第 12 章「実行環境の設定」を参照してください。

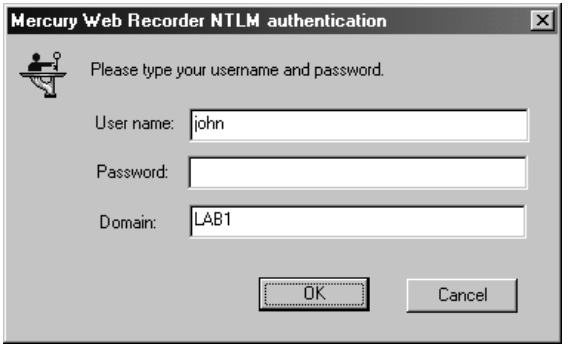
認証情報の提供

次の項の内容は，マルチ・プロトコルによる記録を対象としています。

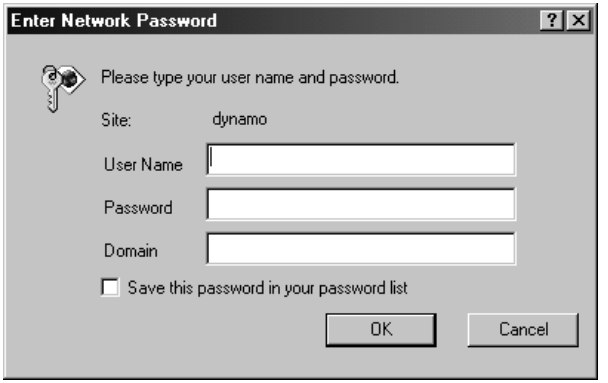
NTLM 認証を使用する Web セッションを記録する際，サーバでユーザ名やパスワードなどの詳細情報を入力しなければならない場合があります。

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，［Mercury Web Recorder NTLM Authentication］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");
```

➤ ［Mercury Web Recorder NTLM Authentication］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,
        "Enter NTLM Password Here（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，パスワードを選択し，右クリック・メニューで［**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，116 ページ「テキストの暗号化」を参照してください。

仮想ユーザ・スクリプトの再生成

スクリプトを記録した後で，トランザクション，ランデブー・ポイント，メッセージ，コメントを追加してスクリプトを拡張できます。詳細については，第 7 章「仮想ユーザ・スクリプトの拡張」を参照してください。

さらに，スクリプトのパラメータ化や，変数の相関も可能です。詳細については，第 8 章「VuGen パラメータを使った作業」を参照してください。

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。

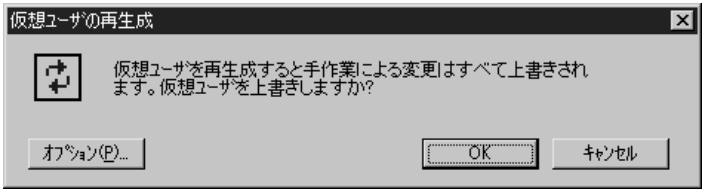
次のボタンが［Regenerate Script］ダイアログ・ボックスから使用できます。

［**OK**］: 元の記録ログから仮想ユーザ・スクリプトを再生成します。再生成では，スクリプトで手動実行したすべての相関とパラメータ化が削除されます。

［**オプション**］: マルチ・プロトコル・スクリプトを処理する場合は，再生成するプロトコルを指定できます。再生成をカスタマイズするには，［仮想ユーザの再生成］ダイアログ・ボックスで［**オプション**］ボタンをクリックし，［オプションの再生成］を開きます。［**プロトコル**］ノードを選択し，再生成するプロトコルとそのままにするプロトコルを指定します。再生成するプロトコルのチェック・ボックスを選択します。再生成しないプロトコルのチェック・ボックスはクリアします。

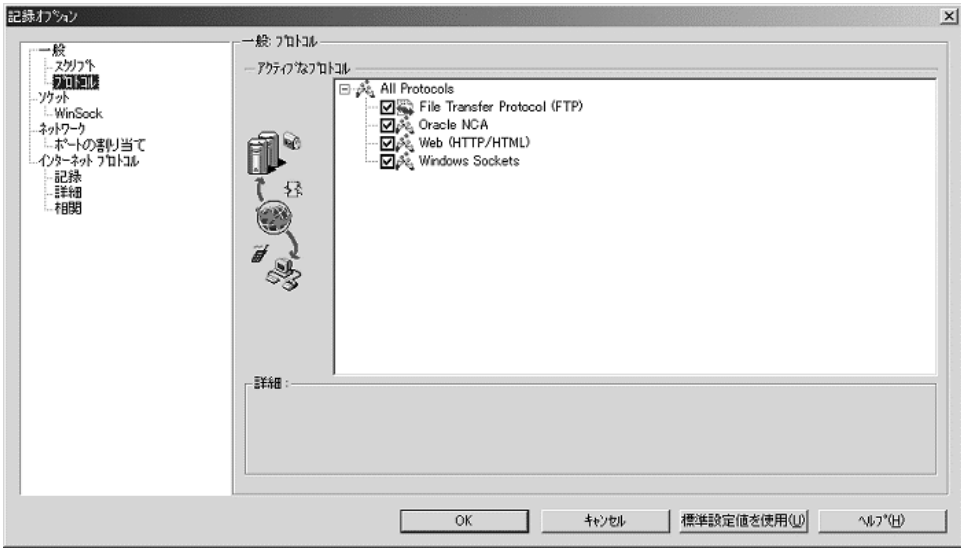
マルチ・プロトコル仮想ユーザ・スクリプトを再生成するには，次の手順を実行します。

1 ［**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。

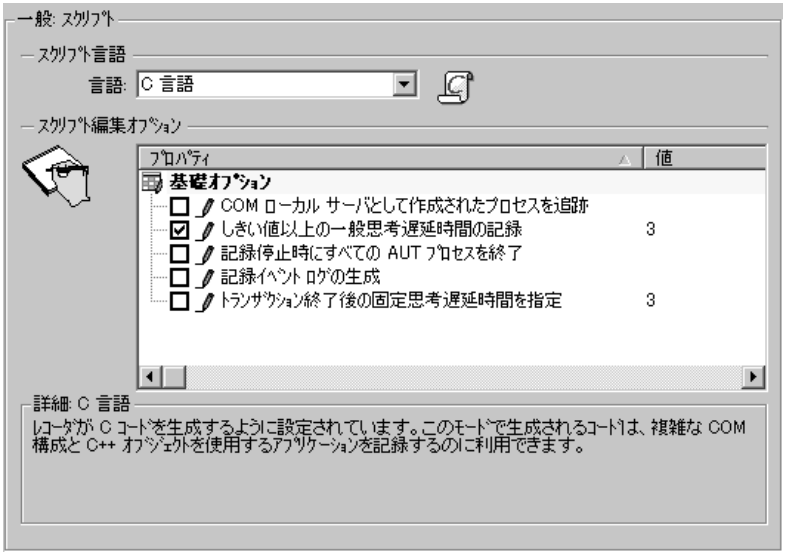


2 ［**オプション**］をクリックして［オプションの再生成］ダイアログ・ボックスを開きます。

3［**一般：プロトコル**］ノードを選択します。再生成するプロトコルとそのままにするプロトコルを指定します。再生成するプロトコルのチェック・ボックスを選択します。変更しないプロトコルのチェック・ボックスはクリアします。



4 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。



第 5 章

スクリプト生成オプションの設定

VuGen を使ってスクリプトを記録する前に，C，Visual Basic，VBScript，JavaScript の中から使用するスクリプト記述言語を指定します。

本章では，サポートされている多くのプロトコルに適用されるスクリプト言語記録オプションについて説明します。

- スクリプト生成オプションの設定について
- スクリプト言語の選択
- 基本オプションの適用
- 相関オプションについて
- 記録オプションの設定

以降の情報は，マルチ・プロトコルの記録をサポートするすべての仮想ユーザ・スクリプトを対象とします。

スクリプト生成オプションの設定について

セッションを記録する前に，スクリプトに含めるものと，その生成方法をレコーダに指示するいくつかの記録オプションを設定できます。

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，**ÉXÉNÉäÉvÉg**・オプションを使用できます。ただし，HTTP または WinSock をシングル・プロトコル・スクリプトとして記録する場合は例外です。この場合，**ÉXÉNÉäÉvÉg**・オプションは利用できません。

スクリプト言語の選択

セッションを記録するときに，VuGen は，標準設定ではユーザの操作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。

［**C 言語**］：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。

［**Visual Basic（アプリケーション使用）**］：VB ベースのアプリケーション用。VB（VBScript とは異なる）の完全な機能を使用します。

［**VBScripting**］：VBScript ベースのアプリケーション用（ASP など）。

［**JavaScripting**］：Javascript ベースのアプリケーション用（**js** ファイルや動的 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，Visual Basic，VBScript コード，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を変更できます 。

以下の項では，スクリプト編集オプションについて説明します。すべてのスクリプトについては，85 ページ「基本オプションの適用」を参照してください。C 以外のスクリプトに相関オプションを設定するには，87 ページ「相関オプションについて」を参照してください。

詳細については，87 ページ「記録オプションの設定」を参照してください。

基本オプションの適用

基本のスクリプト・オプションはすべての生成言語に適用されます。これらのオプションは生成されたスクリプトの詳細レベルを制御します。

［**Close all AUT processes when recording stops（記録停止時にすべての AUT プロセスを終了）**］：VuGen が記録を停止すると，すべてのテスト対象アプリケーション（AUT）の処理が自動的に終了します（標準設定では無効）。

［**Explicit variant declaration（明示的なバリアント宣言）**］：ByRef バリアントを処理するため，バリアント・タイプを明示的に宣言します（Visual Basic for Applications のみ。標準設定では無効）。

［**Generate fixed think time after end transaction（トランザクション終了後の固定思考遅延時間を指定）**］：トランザクション終了後，固定思考遅延時間を秒単位で追加します。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。標準設定は 3 秒です（標準設定では無効）。

［**Generate recorded events log（記録イベント ログの生成**］）：記録中に発生したすべてのイベントのログを生成します（標準設定では無効）。

［**Generate think time greater than threshold（しきい値以上の一般思考遅延時間の記録）**］：思考遅延時間のしきい値を使用します。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（標準設定では有効）。

［**Maximum number of lines in action file（アクション ファイル内の最大行数）**］：アクション内の行数が指定されたしきい値を超えると新規ファイルが作成されます。標準のしきい値は 60000 行です（C 言語のみ）。

［**Insert post-invocation info（起動後情報の挿入）**］：各メッセージ呼び出しの後に，その内容を表すログ・メッセージを挿入します（C 以外のみ，標準設定では有効）。

［**Insert pre-invocation info（起動前情報の挿入）**］：各メッセージ呼び出しの前に，その内容を表すログ・メッセージを挿入します（C 以外のみ。標準設定では有効）。

［**Replace long strings with parameter（長い文字列をパラメータで置換）**］：パラメータへの最大長を超える文字列を保存します。このオプションの初期の最大長は 100 文字です。パラメータと完全な文字列は，スクリプト・フォルダの **lr_strings.h** ファイルに次の形式で保存されます。

```
const char <paramName_uniqueID> ="string".
```

このオプションを使用すると，スクリプトがより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（標準設定では有効）

［**Track processes created as COM local servers（COM ローカル サーバとして作成されたプロセスを追跡）**］：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を追跡します（標準設定では有効）。

［**Use helpers for arrays（配列でのヘルパーの使用）**］：ヘルパー関数を使って，バリアントの配列からコンポーネントを抽出します（Java および VBScript のみ。標準設定では無効）。

［**Use helpers for objects（オブジェクトでのヘルパーの使用）**］：ヘルパー関数を使って，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，その参照を抽出します（Java および VBScript のみ。標準設定では無効）。

詳細については，87 ページ「記録オプションの設定」を参照してください。

相関オプションについて

相関を使って，テストの実行中に動的な値を保持できます。これらのオプションによって，記録時に VuGen によって自動的に行われた相関を拡張設定できます。すべての相関オプションは標準設定では無効になっています。相関オプションは VBScript および JScript 言語だけに適用されます。

［**小さい数の相関**］：記録中に，byte，char，および short int などの短いデータ型を相関させます（標準設定では無効）。

［**大きい数の相関**］：記録中に，int，long int，64 ビットの char，float，double などの長いデータ型を相関させます（標準設定では無効）。

［**単純文字列の相関**］：単純で，配列ではない文字列や文章を相関させます（標準設定では有効）。

［**配列の相関**］：記録中に，文字列，構造体，数値など，すべてのデータ型の配列を追跡して相関させます（標準設定では無効）。

［**構造の相関**］：複雑な構造要素を追跡して相関させます（標準設定では無効）。

詳細については，87 ページ「記録オプションの設定」を参照してください。

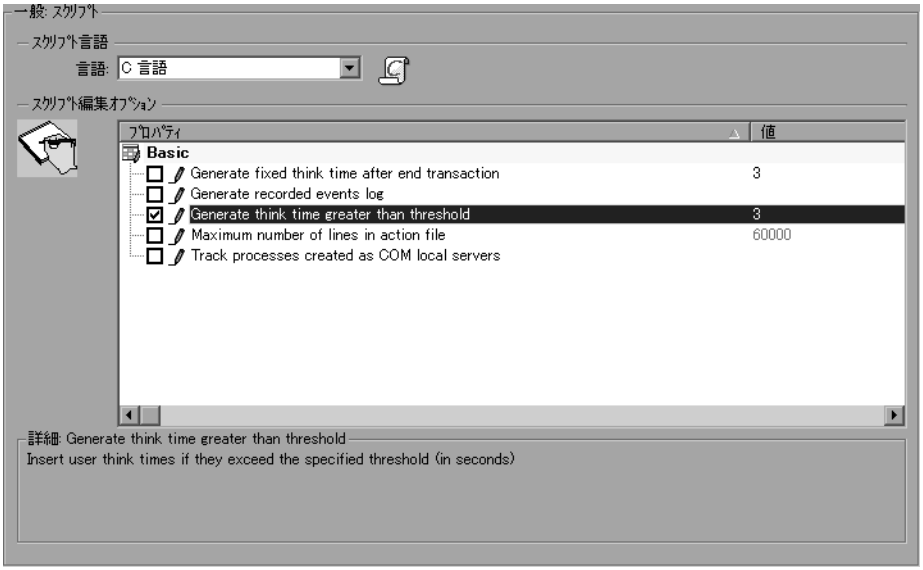
記録オプションの設定

［記録オプション］は，スクリプトに関連する記録を開始する前に設定します。使用できるオプションの数は，スクリプト生成言語によって異なります。

スクリプト記録オプションを設定するには，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。メイン・メニューから［**ツール**］＞［**記録オプション**］を選択するか，［記録開始］ダイアログ・ボックスの［**オプション**］をクリックします。［記録オプション］ダイアログ・ボックスが開きます。

2［**一般：スクリプト**］ノードを選択します。



3［**スクリプト言語**］ボックスで，コード生成モード（［**C åæåÍ**］または［**Visual Basic for Applications**］）を選択します。複雑な構造要素と C++ コードを使用するアプリケーションを記録する場合は，C を使用します。スクリプト・ベースのアプリケーションを記録するには，Visual Basic モードを使用します。

4［**スクリプト編集オプション**］セクションで必要なオプションを選択するには，オプションの横のチェック・ボックスを選択します。このオプションについては，前の項で説明しています。

5［**OK**］をクリックして設定を保存し，ダイアログ・ボックスを閉じます。

第 6 章

ポートの割り当て設定

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プロトコルを使用する場合，トラフィックをどのように割り当てるかをポートに指定することができます。

本章では，以下の項目について説明します。

- ポートの割り当て設定について
- ポート割り当ての定義
- 新規サーバ・エントリの追加
- 自動検出オプションの設定
- ポートの割り当て記録オプションの設定

以降の情報は，ソケット・レベルで記録するすべての仮想ユーザ・スクリプト（HTTP，SMTP，POP3，IMAP，Oracle NCA，Windows Sockets）を対象とします。

ポートの割り当て設定について

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。例えば，**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，［**ポートの割り当て**］オプションが利用できます。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。この場合は，［**ポートの割り当て**］オプションは利用できません。

ポート割り当ての定義

VuGen はポート割り当ての設定を使用して，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。例えば，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。

［**新規エントリ**］：［サーバ エントリ］ダイアログ・ボックスが開き，新しい割り当てを入力できます。92 ページ「新規サーバ・エントリの追加」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。

［**エントリの削除**］：選択したエントリを削除します。

［**オプション**］：［ポートの割り当ての詳細設定］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。94 ページ「自動検出オプションの設定」を参照してください。

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。

優先順位	ポート	サーバ
1	指定あり	指定あり
2	指定なし<すべて>	指定あり
3	指定あり	指定なし<すべて>
4	指定なし<すべて>	指定なし<すべて>

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。例えば，**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。

さらに，次のガイドラインが適用されます。

- **ポート 0**：ポート番号 0 は任意のポートを表します。
- **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。例えば，「＜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。

ポート割り当てを定義すると，その内容は［ポートの割り当て］の一覧に表示されます。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。

手順の詳細については，96 ページ「ポートの割り当て記録オプションの設定」を参照してください。

新規サーバ・エントリの追加

［サーバ エントリ］ダイアログ・ボックスを使用して，ポート割り当てのリストに新規エントリを作成します。

ソケット サービス

［**対象サーバ**］: エントリ項目に登録する対象サーバの IP アドレスまたはホスト名。標準設定は「任意のサーバ」です。

［**ポート**］: エントリ項目に登録する対象サーバのポート番号。ポート番号「0」は「すべてのポート」を意味します。

［**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。

［**サービスの種類**］: サービスのタイプ。現在は TCP/IP に設定されています。

［**接続の種類**］：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。

SSL 設定

接続の種類に［**SSL**］もしくは［**自動**］を選択している場合には，［SSL 設定］セクションの設定を行います。この設定は新規エントリにのみ適用されます。この設定は，アプリケーションの SSL エンコーディングについて明らかな情報がある場合にのみ行ってください。それ以外の場合には，標準設定を使用します。

［**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。標準設定では，SSL 2/3 が指定されています。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。

［**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。

［**指定したクライアント証明書（Base64/PEM）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。

［**指定したプロキシ サーバ証明書（Base64/PEM）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 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。

トラフィック転送

［**次のローカル ポートから目的のサーバに転送する**］：特定のポートからのすべてのトラフィックを別のサーバに転送できます。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。

例えば，**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，［ポートの割り当て］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［**トラフィック転送**］セクションで，［**次のローカル ポートから目的のサーバに転送する**］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。

手順の詳細については，96 ページ「ポートの割り当て記録オプションの設定」を参照してください。

自動検出オプションの設定

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。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。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。他のネットワーク・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。

［**SSL 自動検出を有効にする**］：SSL 通信を自動的に検出します。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，［**接続の種類**］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。

［**SOCKET ベース コミュニケーションの自動検出を有効にする**］：SSL 通信を自動的に検出します。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。

［**更新**］：自動検出オプションの設定を受け入れ，ダイアログ・ボックスを閉じます。

上記のネットワーク・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。例えば，次のような場合です。

- 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。

手順の詳細については，96 ページ「ポートの割り当て記録オプションの設定」を参照してください。

ポートの割り当て記録オプションの設定

［記録オプション］ダイアログ・ボックスは，次のいくつかの方法で開けます。

- ツールバー・ボタン：  ［記録オプションの編集］ボタンをクリックします。
- キーボードのショートカット：Ctrl キーを押しながら F7 キーを押します。
- ［ツール］メニュー：［**ツール**］＞［**記録オプション**］を選択します。

ポートの割り当て記録オプションを設定するには，次の手順を実行します。

1 ［**ネットワーク：ポートの割り当て**］ノードを選びます。

2 新しいサーバのポート割り当てを作成するには，［**新規エントリ**］をクリックします。［サーバ エントリ］ダイアログ・ボックスが開きます。

3［ソケット サービス］セクションで［**サービス ID**］，［**サービスの種類**］，［**対象サーバ**］，［**ポート**］，および［**接続の種類**］を入力します。

4［接続の種類］に［**SSL**］もしくは［**自動**］を選択している場合には，［**SSL 設定**］セクションの設定を行います。この設定は新規エントリにのみ適用されます。この設定は，アプリケーションの SSL エンコーディングについて明らかな情報がある場合にのみ行ってください。それ以外の場合には，標準設定を使用します。

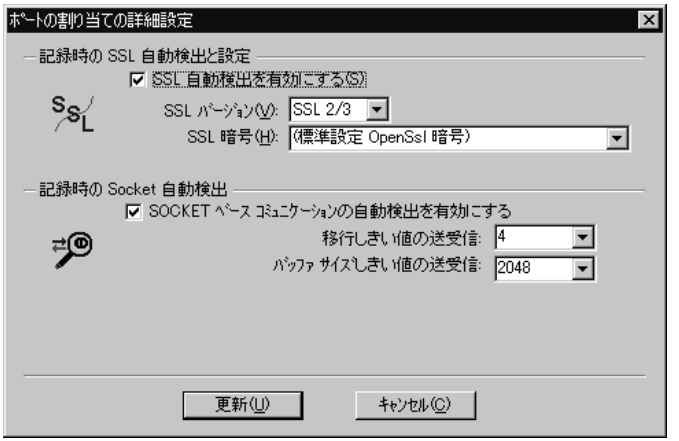
［**SSL バージョン**］，［**SSL 暗号**］を指定します。証明書を使用する場合には，［**指定したクライアント証明書（Base64/PEM）を使用する**］または［**指定したプロキシ サーバ証明書（Base64/PEM）を使用する**］を選択してユーザ情報を指定します。

［**SSL 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。

5 トラフィックの転送ができるようにするには，［**次のローカルポートから目的のサーバに転送する**］オプションを選択し，ポート番号を指定します。このオプションは，**対象のサーバ**と**対象ポート**が一意（<Any> が指定されていない）のときだけ，有効になります。

6［**更新**］をクリックして，ダイアログ・ボックスを閉じます。

7 自動検出機能を設定するには，［**オプション**］をクリックします。［ポートの割り当ての詳細設定］ダイアログ・ボックスが開きます。



SSL 通信の自動検出を行うには，［**SSL 自動検出を有効にする**］チェック・ボックスを選択し，バージョンと暗号の情報を指定します。

通信の種類を自動的に検出するには，［**SOCKET ベース コミュニケーションの自動検出を有効にする**］チェック・ボックスを選択し，必要に応じて［移行しきい値の送受信］の値を大きくします。

［**更新**］をクリックして設定を受け入れ，ダイアログ・ボックスを閉じます。

8 すべてのエントリを表示するには，［**次のネットワークレベルのサーバアドレス割り当て**］ボックスから［(**全 ID**)］を選びます。

9 既存のエントリを修正する場合には，修正したいエントリを選択して，［**エントリの編集**］をクリックします。エントリのサーバ名，ポート番号は変更できないことに注意してください。変更できるのは，接続の種類とセキュリティ設定だけです。

10 割り当てを恒久的に削除するには，エントリをリストから選び，［**エントリの削除**］をクリックします。特定のエントリについて，一時的に割り当て設定を無効にするには，項目の横のチェック・ボックスをクリアします。割り当てを有効にするにはチェック・ボックスを選びます。

11［**OK**］をクリックします。

第 7 章

仮想ユーザ・スクリプトの拡張

記録中または記録後に，一般仮想ユーザ関数，プロトコル固有の仮想ユーザ関数，および標準 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。

本章では，次の項目について説明します。

- 仮想ユーザ・スクリプトの拡張について
- 仮想ユーザ・スクリプトへのトランザクションの挿入
- ランデブー・ポイントの挿入（LoadRunner および Tuning のみ）
- 仮想ユーザ・スクリプトへのコメントの挿入
- 仮想ユーザ情報の取得
- 出力へのメッセージの送信
- 実行中の仮想ユーザ・スクリプトのエラー処理
- 仮想ユーザ・スクリプトの同期化
- ユーザの思考遅延時間のエミュレート
- コマンド・ライン引数の取り扱い
- テキストの暗号化
- パスワードの手動でのエンコーディング
- スクリプト・フォルダへのファイルの追加

以降の情報は，GUI と Java を除く仮想ユーザ・スクリプトの全タイプを対象とします。

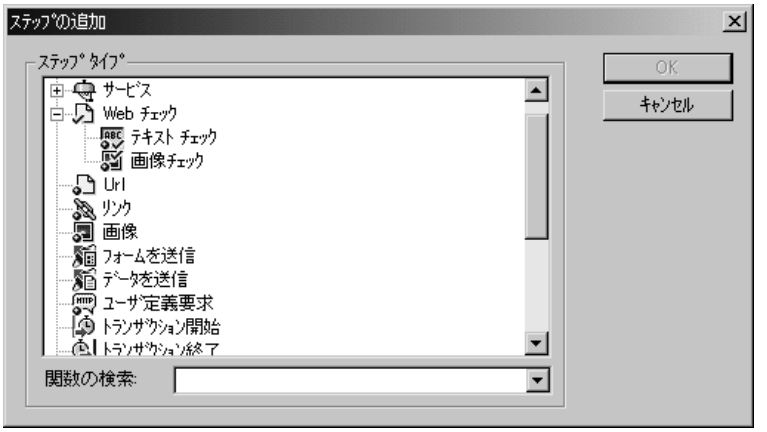
仮想ユーザ・スクリプトの拡張について

仮想ユーザ・スクリプトの記録中または記録後に，ステップ（または関数）を手作業で追加することによって機能を拡張できます。

スクリプトに新規ステップを追加するには，次の手順を実行します。

1 希望の場所にカーソルを置きます。

2 ［**挿入**］＞［**新規ステップ**］を選択します。［ステップの追加］ダイアログ・ボックスが開き，現在のプロトコルに関連するステップが表示されます。



3 ステップを選択して［**OK**］をクリックします。カーソルのある位置にステップ（［スクリプト ビュー］では関数）が挿入されます。

［ステップの追加］ダイアログ・ボックスでは，次の種類の関数が使用できます。

- 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

一般仮想ユーザ関数は，仮想ユーザ・スクリプトの機能を大幅に強化します。例えば，一般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。

一般仮想ユーザ関数は任意のタイプの仮想ユーザ・スクリプトで使用できます。一般仮想ユーザ関数には，必ず **LR** という接頭辞が付いています。VuGen は，スクリプト記録中にいくつかの一般仮想ユーザ関数を生成し，仮想ユーザ・スクリプトに挿入します。自動生成されなかった他の関数を使用するには，VuGen のメイン・ウィンドウで［**挿入**］＞［**新規ステップ**］を選び関数を選択します。

本章では，最もよく使う一般仮想ユーザ関数だけを対象に使い方を説明します。仮想ユーザ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。

プロトコル固有の仮想ユーザ関数

仮想ユーザ・スクリプトを拡張する関数ライブラリがいくつかあります。各ライブラリは，仮想ユーザのタイプに固有のものです。例えば，Windows Sockets 仮想ユーザ・スクリプトでは **LRS** 関数を使用し，Tuxedo 仮想ユーザ・スクリプトでは **LRT** 関数を使用します。プロトコル固有の仮想ユーザ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 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，『**第 2 巻 - プロトコル**』の「ユーザ定義の仮想ユーザ・スクリプトの作成」を参照してください。

仮想ユーザ・スクリプトへのトランザクションの挿入

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には，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 および Tuning Module では，コントローラまたはコンソールによって，各トランザクションの実行にかかる時間が測定されます。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。

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。記録後にトランザクションを追加するには，トランザクション・エディタを使用し，トランザクションのステップをグラフィックで示します。詳細については，48 ページ「トランザクション」を参照してください。または，［**挿入**］メニューを使用して，**トランザクション開始**マーカおよび**トランザクション終了**マーカを追加します。

ここでは，記録中にトランザクションを作成する方法について説明します。

トランザクションの開始を示す方法

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。

トランザクションの開始を示すには，次の手順を実行します。

1 仮想ユーザ・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 をクリックします。［トランザクション開始］ダイアログ・ボックスが開きます。

2 トランザクションの名前を［トランザクション名］ボックスに入力します。トランザクション名の先頭には英数字を使用します。トランザクション名には英数字または記号（!，$，%，&，'，-，[，^，_，`，<，>，{，}，|，~）を使用することができます。ピリオド（.）は使用しないでください。

［**OK**］をクリックしてトランザクション名を確定します。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。例えば，次の関数は **trans1** トランザクションの開始を示します。

```
lr_start_transaction("trans1");
```

トランザクションの終了を示す方法

ビジネス・プロセスの終了を，トランザクション終了ステートメントで示すことができます。

トランザクションの終了を示すには，次の手順を実行します。

1 スクリプトの記録中に，記録ツールバーの［**トランザクション終了マーカを挿入**］ボタン をクリックします。［トランザクション終了］ダイアログ・ボックスが開きます。

2 矢印をクリックして，開始されているトランザクションのリストを表示します。終了するトランザクションを選択します。

［**OK**］をクリックしてトランザクション名を確定します。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。例えば，次の関数は **trans1** トランザクションの終了を示します。

```
lr_end_transaction("trans1", LR_AUTO);
```

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なければなりません。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。

ランデブー・ポイントの挿入（LoadRunner および Tuning のみ）

本項は LoadRunner および Tuning Module にのみ適用されます。
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，他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，コントローラまたはコンソール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。

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。

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。

1 仮想ユーザ・スクリプトの記録中に，記録ツールバーの［**ランデブーを挿入**］ボタン をクリックします。［ランデブー］ダイアログ・ボックスが開きます。

2 ランデブー・ポイントの名前を［**ランデブー名**］ボックスに入力します。

［**OK**］をクリックします。VuGen によって **lr_rendezvous** 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。例えば，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");
```

3 記録セッションの後にスクリプトにランデブー・ポイントを挿入するには，VuGen のツールバーから［**挿入**］＞［**ランデブー**］を選択します。

仮想ユーザ・スクリプトへのコメントの挿入

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。例えば，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。

コメントを挿入するには，次の手順を実行します。

1 スクリプトの記録中に，記録ツールバーの［**コメントを挿入**］ボタン をクリックします。［コメントの挿入］ダイアログ・ボックスが開きます。

2 テキスト・ボックスにコメントを入力します。

3［**OK**］をクリックするとダイアログ・ボックスが閉じ，コメントが挿入されます。テキストはスクリプトの現在位置に，コメント記号で囲まれた状態で挿入されます。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。

```
/*
* これは最初のクエリです
*/
```

注：記録セッションを完了した後でスクリプトにコメントを挿入するには，VuGen のメニューから［**挿入**］＞［**コメント**］を選択します。

仮想ユーザ情報の取得

以下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	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	コントローラまたは Tuning Console を実行しているマシンの名前を返します。Administration Console 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。Administration Console 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( );
```

上記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・ファイルに送信できます。例えば，クライアント・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。LoadRunner コントローラまたは Tuning Console 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。

Application Management 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。例えば，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。

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］ウィンドウまたは Business Process Monitor ログ・ファイルに送ります。
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	メッセージをコントローラまたは Tuning Console の仮想ユーザ・ステータス領域に送信します。Administration Console 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。

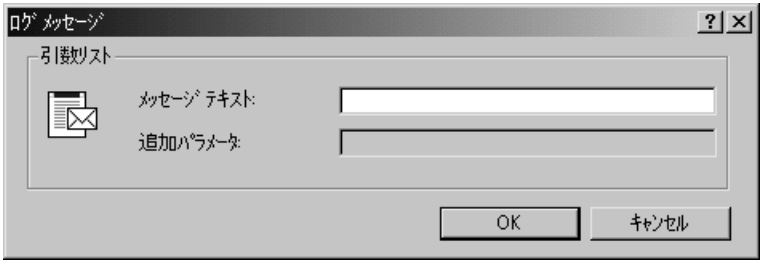
注：lr_message，**lr_output_message**，**lr_log_message** の各関数の振る舞いは，実行環境の設定の［ログ］の設定にある，スクリプトのデバッグ・レベルの影響を受けません。これらの関数は常にメッセージを送信します。

ログ・メッセージ

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。例えば，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。

lr_log_message 関数を挿入するには，次の手順を実行します。

1 ［**挿入**］＞［**ログ メッセージ**］を選択します。［ログ メッセージ］ダイアログ・ボックスが開きます。



2 ［**メッセージ テキスト**］ボックスにメッセージを入力します。

3 ［**OK**］をクリックするとメッセージが挿入され，ダイアログ・ボックスが閉じます。**lr_log_message** 関数がスクリプトの現在の位置に挿入されます。

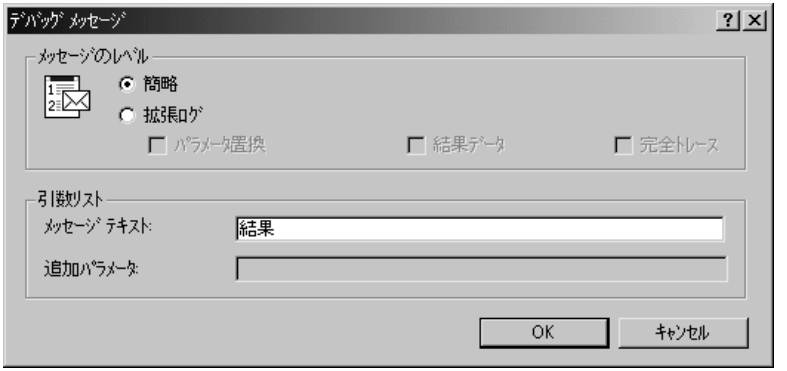
デバッグ・メッセージ

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。

デバッグ関数を挿入するには，次の手順を実行します。

1［**挿入**］＞［**新規ステップ**］を選択します。［ステップの追加］ダイアログ・ボックスが開きます。

2［**デバッグ メッセージ**］ステップを選択して［**OK**］をクリックします。［デバッグ メッセージ］ダイアログ・ボックスが開きます。



3 メッセージのレベルとして，［**簡略**］または［**拡張ログ**］を選択します。［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。

4［**メッセージ テキスト**］ボックスにメッセージを入力します。

5［**OK**］をクリックするとメッセージが挿入され，ダイアログ・ボックスが閉じます。**lr_debug_message** 関数がスクリプトの現在の位置に挿入されます。

エラーおよび出力メッセージ

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。

エラーまたは出力メッセージ関数を挿入するには，次の手順を実行します。

1［**挿入**］＞［**新規ステップ**］を選択します。［ステップの追加］ダイアログ・ボックスが開きます。

2［**エラー メッセージ**］または［**出力メッセージ**］ステップを選択して［**OK**］をクリックします。［エラー メッセージ］または［出力メッセージ］ダイアログ・ボックスが開きます。

3［**メッセージ テキスト**］ボックスにメッセージを入力します。

4［**OK**］をクリックするとメッセージが挿入され，ダイアログ・ボックスが閉じます。**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。

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。

実行中の仮想ユーザ・スクリプトのエラー処理

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。詳細については，199 ページ「エラー処理」を参照してください。

➤ **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);** ステートメントと **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，次ページを参照してください。

例えば，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);
```

仮想ユーザ・スクリプトの同期化

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。

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関数の使用法の詳細については，『**第 2 巻 - プロトコル**』の「RTE 仮想ユーザ・スクリプトの同期化」を参照してください。

ユーザの思考遅延時間のエミュレート

連続する操作の合間のユーザの待ち時間を「**思考遅延時間**」といいます。仮想ユーザでは，**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，手作業でも **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。

思考遅延時間ステートメントを手作業で追加するには，次の手順を実行します。

1 希望の場所にカーソルを置きます。

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

3［**思考遅延時間**］を選択し，［**OK**］をクリックします。［思考遅延時間］ダイアログ・ボックスが開きます。

4 思考遅延時間を秒単位で指定し，［**OK**］をクリックします。

注：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。

思考遅延時間の設定を使って，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。思考遅延時間の設定にアクセスするには，VuGen のメイン・メニューから［**仮想ユーザ**］＞［**実行環境の設定**］を選択して［**思考遅延時間**］ノードをクリックします。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。

コマンド・ライン引数の取り扱い

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。Tuning Console を使用している場合は，［実行環境設定］ダイアログ・ボックスの中でコマンド・ライン引数を指定できます。詳細については，197 ページ「実行環境の追加属性の設定」を参照してください。

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 2 つのどちらかを使用します。スクリプト名の後ろに，引数とその値を 2 つ 1 組で指定します。

```
スクリプト名　- 引数 引数値　- 引数 引数値
```

```
スクリプト名　/ 引数 引数値　/ 引数 引数値
```

例えば次の例は，pc4 というロード・ジェネレータで **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，またはコマンド・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。

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，手作業でも実行できます。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"));
```

文字列は，いつでも復号して元の値に戻せます。

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復元するには，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 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。

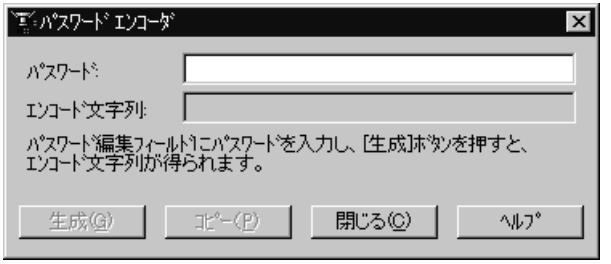
lr_decrypt 関数の詳細については，「**オンライン関数リファレンス**」（［**ヘルプ**］＞［**関数リファレンス**］）を参照してください。

パスワードの手動でのエンコーディング

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。例えば，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確保したいとします。［**パスワード エンコーダ**］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**］＞［**プログラム**］＞［**Mercury LoadRunner**］＞［**Tools**］＞［**Password Encorder**］を選択します。［パスワード エンコーダ］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。［パスワード エンコーダ］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。

5 暗号化したいパスワードごとに，この手順を繰り返します。

6［**閉じる**］をクリックして，［パスワード エンコーダ］を閉じます。

スクリプト・フォルダへのファイルの追加

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。

ファイルを追加するには，次の手順を実行します。

1 スクリプトを表示した状態で，［**ファイル**］＞［**スクリプトにファイルを追加**］を選択します。

2 ファイルを参照し，［**開く**］をクリックします。選択したファイルが追加されます。

第 8 章

VuGen パラメータを使った作業

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。

本章では，次の項目について説明します。

- VuGen パラメータの定義について
- パラメータに関する制限
- パラメータの作成
- パラメータ・タイプについて
- パラメータのプロパティの定義
- 既存のパラメータの使用
- パラメータ・リストの使用
- パラメータ化オプションの設定

以降の情報は，GUI を除く全タイプの仮想ユーザ・スクリプトを対象とします。

VuGen パラメータの定義について

ビジネス・プロセスを記録すると，VuGen によって関数で構成された仮想ユーザ・スクリプトが生成されます。関数内の引数の値は，記録セッション中に実際に使用された値です。

例えば，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。そこで，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);
```

こうしておけば，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。データ・ソースの詳細については，125 ページ「パラメータ・タイプについて」を参照してください。

仮想ユーザ・スクリプトのパラメータ化には次の 2 つの利点があります。

- スクリプトのサイズが小さくなります。
- さまざまな値を使ってスクリプトをテストすることができます。例えば，図書館のデータベースからいくつかの本を探す場合でも，submit 関数を一度書くだけで済みます。仮想ユーザに特定の項目の探索を指示するのではなく，パラメータを使用します。再生時，仮想ユーザによってパラメータが異なるさまざまな値で置き換えられます。

パラメータ化には，次の 2 つの作業が伴います。

- 仮想ユーザ・スクリプト内の定数値をパラメータで置き換える。
- パラメータのプロパティとデータ・ソースを設定する。

パラメータに関する制限

パラメータ化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**］）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。

パラメータの作成

パラメータを作成するには，パラメータに名前を付け，そのタイプとプロパティを指定します。仮想ユーザ・スクリプトに作成できるパラメータの数に制限はありません。パラメータを作成するには，次の手順を実行します。

手順 1：パラメータ化する引数を選択します。

スクリプト・ビューで作業している場合：

パラメータ化する引数を右クリックし，ポップアップ・メニューから［**パラメータで置換**］を選択します。

注：CORBA または General-Java の仮想ユーザ・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・ビューで作業している場合：

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

2 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。

［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。

手順 2：パラメータ名を指定します。

［**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。このパラメータ名はスクリプト内で元の引数の代わりに表示されます。

パラメータ名は，スクリプトの実行中に置き換えられる情報のタイプに適した名前を付けてください。

例えば，一般にユーザ名を入力する場合は，パラメータの名前を Username などとします。

注：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。

手順 3：パラメータ・タイプを選択します。

パラメータを作成するときは，パラメータのデータ・ソースを指定します。これによって，**パラメータ・タイプ**が決まります。

データは，日付や時刻のように内部的に生成したり，ユーザ定義関数の結果として返すようにしたりできます。

パラメータの別のごく一般的な使用法として，ユーザが定義した値が格納されているデータ・テーブルや外部ファイルから値を取得するように，仮想ユーザ

を設定する方法があります。このようなパラメータを，ファイル・タイプ・パラメータ，およびテーブル・タイプ・パラメータと呼びます。

［**パラメータのタイプ**］リストから，［**ファイル**］を選択します。

各種のパラメータ・タイプの詳細については，125 ページ「パラメータ・タイプについて」を参照してください。

手順 4：パラメータ・タイプのプロパティを定義します。

1 ［**プロパティ**］をクリックします。［パラメータのプロパティ］ダイアログ・ボックスが開きます。

2 ［**テーブルの作成**］をクリックします。メッセージ・ボックスが開きます。［**OK**］をクリックします。

VuGen によって，引数の元の値を格納したセルを 1 つ持つテーブルが作成されます。

3 テーブルに別の値を追加するには，［**行の追加**］をクリックし，値を入力します。

値をさらにテーブルに追加するには，この手順を繰り返します。

4 ［**閉じる**］をクリックして，［パラメータのプロパティ］ダイアログ・ボックスを閉じます。

詳細については，128 ページ「パラメータのプロパティの定義」を参照してください。

手順 5：そして引数をパラメータで置き換えます。

［**OK**］をクリックして，［パラメータの選択または作成］ダイアログ・ボックスを閉じます。

VuGen によって，スクリプト内の選択された文字列が，中括弧または丸括弧で囲まれたパラメータ名で置き換えられます。

注：標準のパラメータの括弧は，プロトコルの種類に応じて中括弧か大括弧のどちらかです。パラメータの括弧を変更したければ，［一般オプション］ダイアログ・ボックス（［**ツール**］＞［**一般オプション**］を選択）の［パラメータ化］タブから変更できます。詳細については，135 ページ「パラメータ化オプションの設定」を参照してください。

ツリー・ビューでは，［**ABC**］アイコンがテーブル・アイコン に置き換えられます。

次の例では，元の **username** の値は **jojo** でしたが，パラメータ **{UserName}** に置き換えられています。

パラメータ・タイプについて

パラメータを作成するときは，パラメータのデータ・ソースを指定します。次のタイプのデータ・ソースを指定できます。

- ファイルまたはテーブル・パラメータ・タイプ
- 内部データ・パラメータ・タイプ
- ユーザ定義関数のパラメータ

ファイルまたはテーブル・パラメータ・タイプ

既存のファイルか，VuGen または MS Query で作成したファイルに含まれるデータ。パラメータのごく一般的な使用法の 1 つに，データ・テーブルや外部ファイルから値を取得するように仮想ユーザを設定するために使用するという方法があります。

データ・ファイル

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分かっている場合は，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。

データ・テーブル

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_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。

データ・ファイルまたはデータ・テーブルのパラメータのプロパティについては，第 9 章「ファイルまたはテーブル・パラメータ・タイプ」を参照してください。

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。Date/Time（日時），Group Name（グループ名），Iteration Number（反復回数），Load Generator Name（ロード・ジェネレータ名），Random Number（乱数），Unique Number（一意の数），仮想ユーザ ID などがあります。

内部データのパラメータのプロパティの定義方法については，156 ページ「内部データ・パラメータ・タイプのプロパティの設定」を参照してください。

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，ライブラリのフル・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。Mercury 仮想ユーザ・ジェネレータの bin ディレクトリは標準のダイナミック・ライブラリ・パスに含まれています。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以下に示します。

```
__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;}
```

ユーザ定義関数のプロパティの定義方法については，166 ページ「ユーザ定義関数のプロパティの設定」を参照してください。

パラメータのプロパティの定義

パラメータのプロパティは，［パラメータのプロパティ］ダイアログ・ボックスか［パラメータ リスト］ダイアログ・ボックスで定義できます。

［パラメータのプロパティ］ダイアログ・ボックスでパラメータのプロパティを定義するには，次の手順を実行します。

1 **［パラメータのプロパティ］ダイアログ・ボックスを開きます。**

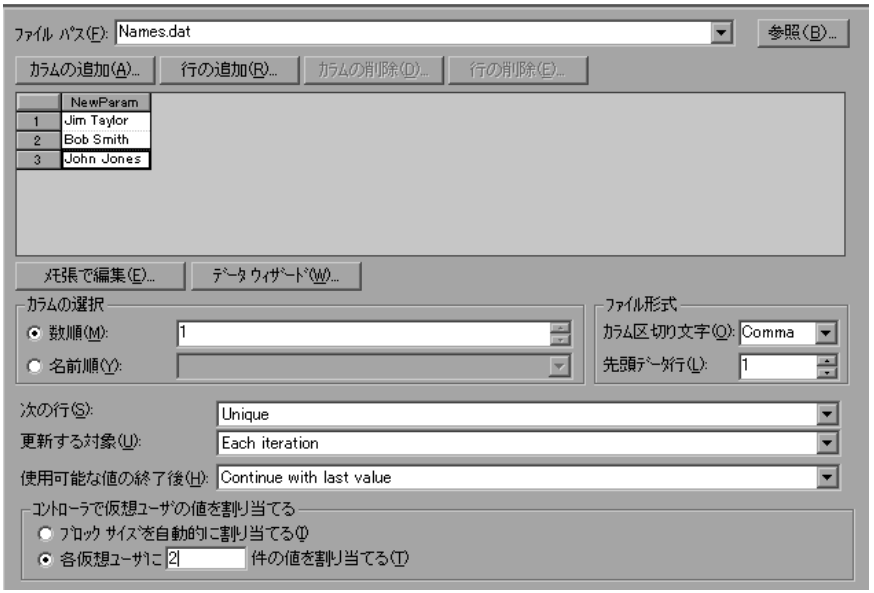
［パラメータのプロパティ］ダイアログ・ボックスは，次のいずれかの方法で開くことができます。

- 新しいパラメータを作成するときに（122 ページ「パラメータの作成」を参照），［パラメータの選択または作成］ダイアログ・ボックスの［**プロパティ**］をクリックして，［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。
- スクリプト・ビュー内でパラメータを選択し，右クリック・メニューから［**パラメータのプロパティ**］を選択します。

➤ ツリー・ビューで，プロパティを定義する対象となるパラメータを含んでいるステップを右クリックし，［**プロパティ**］を選択します。選択したステップに対応する［ステップのプロパティ］ダイアログ・ボックスが開きます。

プロパティを定義する対象となるパラメータの横にあるテーブル･アイコンをクリックし，ポップアップ・メニューから［**パラメータのプロパティ**］を選択します。

次の例では，「**File**」タイプのパラメータのプロパティが表示されています。



2 **パラメータのプロパティを定義します。**

➤ ファイルおよびテーブル・タイプ・パラメータのプロパティを定義する方法については，第 9 章「ファイルまたはテーブル・パラメータ・タイプ」を参照してください。

➤ 内部データ・パラメータ・タイプのプロパティを定義する方法については，156 ページ「内部データ・パラメータ・タイプのプロパティの設定」を参照してください。

➤ ユーザ定義関数のプロパティを定義する方法については，127 ページ「ユーザ定義関数のパラメータ」を参照してください。

3 **［パラメータのプロパティ］ダイアログ・ボックスを閉じます。**

［**閉じる**］をクリックして，［パラメータのプロパティ］ダイアログ・ボックスを閉じます。

［パラメータ リスト］ダイアログ・ボックスでパラメータのプロパティを定義するには，次の手順を実行します。

［**パラメータ リストを開く**］ボタン をクリックするか，［**仮想ユーザ**］＞［**パラメータ リスト**］を選択します。プロパティを表示するパラメータを選択します。

詳細については，133 ページ「パラメータ・リストの使用」を参照してください。

既存のパラメータの使用

作成したパラメータは，VuGen によってパラメータ・リストに格納されます。既存のパラメータを使用して，1 つの引数を置き換えたり，複数出現する引数を置き換えたりできます。

定義済みパラメータを使用した文字列の置換

定義済みパラメータを引数に対して割り当てることができます。

定義済みパラメータで文字列を置換するには，次の手順を実行します。

1 スクリプト・ビューに入ります。

2 パラメータ化する引数を右クリックして，［**既存のパラメータで置換**］を選択します。サブメニューが開きます。

3 次のいずれかの方法でパラメータを選択します。

➤ サブメニューのリストからパラメータを選択します。

➤ ［**パラメータ リストから選択**］を選択して［パラメータ リスト］ダイアログ・ボックスを開き，左側の表示枠からパラメータを選択します。

［**パラメータ リスト**］を使用すると，以前に定義したパラメータで引数を置き換えられるのと同時に，そのプロパティの表示や変更もできるので便利です。パラメータ・リストの使い方の詳細については，133 ページ「パラメータ・リストの使用」を参照してください。

複数の出現の置き換え

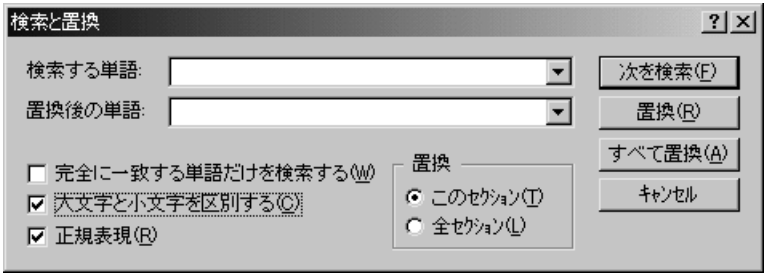
パラメータを作成すると，システムによって引数の元の値が記憶されます。**検索と置換**機能を使用すると，同じ引数が複数出現する場合に，そのうちの選択した引数，またはそれらすべての引数を，同じパラメータや既存の別のパラメータに置換できます。

特定のパラメータへの複数出現する引数を置換するには，次の手順を実行します。

1 パラメータを右クリックし，メニューから［**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・ボックスが開きます。［**検索する単語**］ボックスに，置換対象の値または引数が表示されます。［**置換後の単語**］ボックスに，括弧にはさまれたパラメータ名が表示されます。

2 必要に応じて［完全に一致する単語だけを検索する］や［大文字と小文字を区別する］のチェック・ボックスを選択します。正規表現（.，!，?，+ など）を使って検索するには，［**正規表現**］チェック・ボックスを選択します。



3［**置換**］または［**すべて置換**］をクリックします。

上の例では，値が **15** の引数がすべて，パラメータ **{NewParam}** に置換されます。

注 :［**すべて置換**］の使用は，特に数字文字列の置換の場合は注意が必要です。指定した文字列がすべての出現箇所で置換されます。

元の文字列の復元

VuGen では，パラメータ化を取り消し，記録された元の引数を復元することができます。

パラメータの元の値を復元するには，次の手順を実行します。

➤［スクリプト ビュー］: パラメータを右クリックし，［**既定値を復元**］を選択します。

➤［ツリー ビュー］:

➤ パラメータが含まれているステップを右クリックして，［**プロパティ**］を選択します。

➤ 元の値に戻すパラメータの横にあるテーブル・アイコン をクリックし，［**パラメータを元に戻す**］を選択します。

元の引数が復元されます。

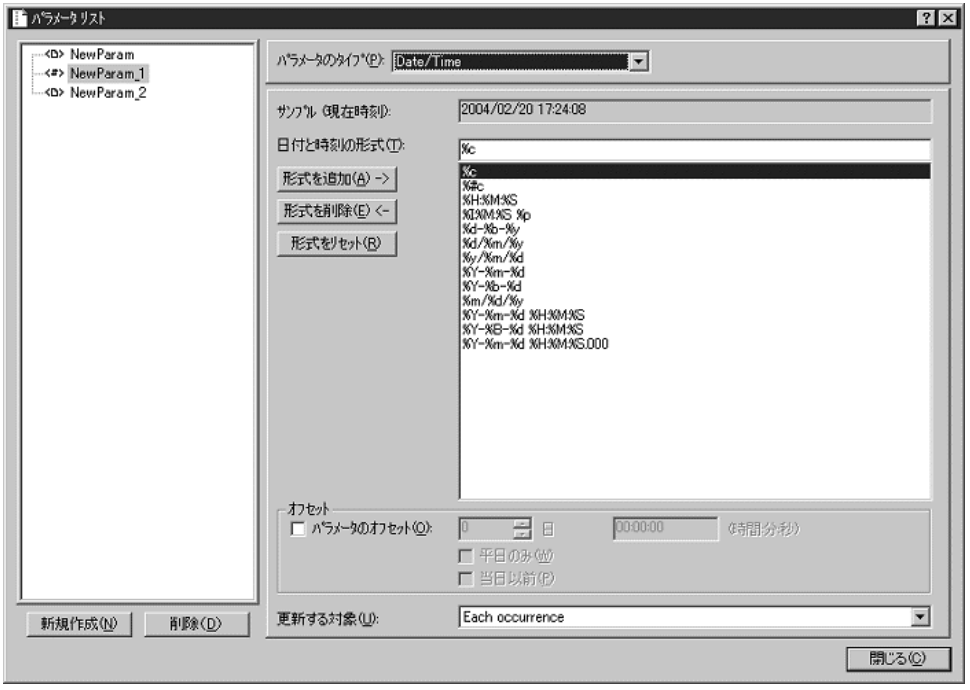
パラメータ・リストの使用

パラメータ・リストを使って，パラメータのリスト表示，パラメータの新規作成，パラメータの削除，またはパラメータのプロパティの変更が行えます。

パラメータ・リストを表示し，パラメータのプロパティを表示するには，次の手順を実行します。

［**パラメータ リストを開く**］ボタン をクリックするか，［**仮想ユーザ**］＞［**パラメータ リスト**］を選択します。プロパティを表示するパラメータを選択します。

次の例では，「**Date/Time**」タイプのパラメータを表示しています。



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。

左側のパラメータ・ツリーからパラメータを選択し，パラメータのタイプとプロパティを右側の表示枠で編集します。

パラメータ・プロパティの設定方法の詳細ついては，第 10 章「パラメータのプロパティの設定」および第 9 章「ファイルまたはテーブル・パラメータ・タイプ」を参照してください。

新しいパラメータを作成するには，次の手順を実行します。

1 ［パラメータ リスト］ダイアログ・ボックスで，［**新規作成**］をクリックします。新規パラメータは，仮の名前でパラメータ・ツリーに表示されます。

2 新規パラメータの名前を入力して，Enter キーを押します。

注：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。

---

3 パラメータのタイプとプロパティを設定します。

4 ［**閉じる**］をクリックして，［パラメータ リスト］ダイアログ・ボックスを閉じます。

注：VuGen によって新規パラメータが作成されますが，スクリプト内で選択されている文字列をそのパラメータで自動的に置き換えることはありません。

既存のパラメータを削除するには，次の手順を実行します。

1 パラメータ・ツリーからパラメータを選択し，［**削除**］をクリックします。［パラメータの削除］ダイアログ・ボックスが開きます。

2 パラメータ・ファイルをディスクから削除したければ，［**ディスクからパラメータ データ ファイルを削除する**］を選択します。

3 ［**はい**］をクリックします。

4 ［**ディスクからパラメータ データ ファイルを削除する**］を選択した場合は，警告メッセージが表示されます。［**はい**］をクリックして操作を確定します。

パラメータ化オプションの設定

パラメータ化オプションは，VuGen の［一般オプション］ウィンドウの［パラメータ化］タブで設定します。

パラメータ化オプションを設定するには，次の手順を実行します。

1［**ツール**］＞［**一般オプション**］を選択します。

2［**パラメータ化**］タブを選択します。

3 以降で説明しているように，パラメータの括弧のスタイルを設定します。

4 136 ページで説明しているように，グローバル・ディレクトリを設定します。

5［**OK**］をクリックして［一般オプション］ウィンドウを閉じます。

パラメータの括弧

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。Web または WAP スクリプトの標準の括弧は中括弧 { } です。次に例を示します。

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

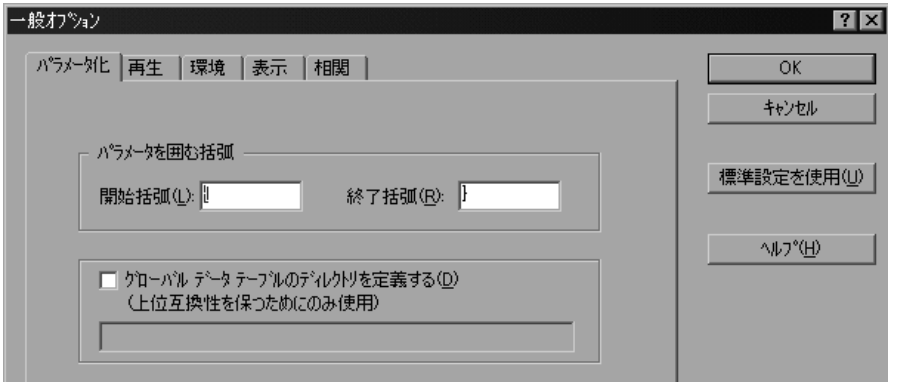
1 文字または複数の文字で構成される文字列を指定して，パラメータの括弧のスタイルを変更できます。スペース以外のすべての文字を使用できます。

注：標準のパラメータの括弧は，プロトコル・タイプに応じて中括弧か大括弧のどちらかです。

パラメータの括弧のスタイルを変更するには，次の手順を実行します。

1［**ツール**］＞［**一般オプション**］を選択します。［一般オプション］ダイアログ・ボックスが開きます。

2［**パラメータ化**］タブを選択し，使用する括弧を入力します。



3［**OK**］をクリックして設定を適用し，ダイアログ・ボックスを閉じます。

グローバル・ディレクトリ

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。以前のバージョン（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，［一般オプション］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，126 ページ「データ・ファイル」を参照してください。

グローバル・ディレクトリを設定するには，次の手順を実行します。

1［**ツール**］＞［**一般オプション**］を選択します。［一般オプション］ダイアログ・ボックスが開きます。

2［**パラメータ化**］タブを選択します。

3［**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。

4［**OK**］をクリックして設定を適用し，ダイアログ・ボックスを閉じます。

第 9 章

ファイルまたはテーブル・パラメータ・タイプ

パラメータのごく一般的な使用法の 1 つに，データ・テーブルや外部ファイルから値を取得するように仮想ユーザを設定するために使用するという方法があります。データは，既存のファイルか，VuGen または MS Query で作成したファイルに含まれています。

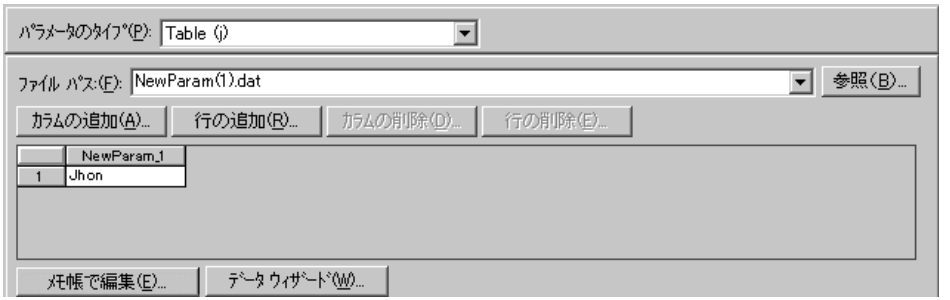
本章では，次の項目について説明します。

- データ・ファイルまたはデータ・テーブルの選択または作成
- ファイル・タイプのパラメータのプロパティ設定
- テーブル・タイプのパラメータのプロパティ設定
- ファイル・タイプまたはテーブル・タイプのパラメータにデータを割り当てる方法の選択

データ・ファイルまたはデータ・テーブルの選択または作成

ファイル・タイプまたはテーブル・タイプのパラメータを作成するときは，データを格納するための .dat ファイルを作成するか，既存の .dat ファイルを開く必要があります。そして，パラメータの他のプロパティを定義します。例えば，仮想ユーザにおけるパラメータへの値の割り当て方法を定義します。

新しいデータ・テーブルを作成するか，既存のデータ・ソースを［ファイル パス］リストから選択できます。



データのソース・ファイルまたはテーブルを選択するには，次の手順を実行します。

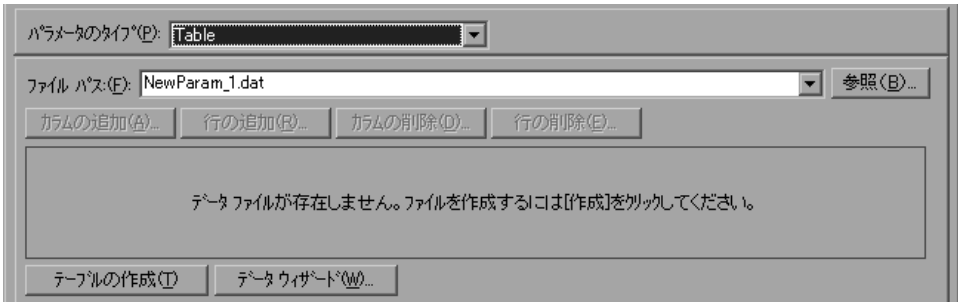
1 ［パラメータのプロパティ］ダイアログ・ボックス，または［パラメータ リスト］を開きます。

手順の詳細については，128 ページ「パラメータのプロパティの定義」を参照してください。

2 テーブルを選択するか，または新しいテーブルを作成します。

- ファイル・パス・リストにテーブル（**.dat** ファイル）が表示されない場合，または新しいテーブルを作成する場合は，［**テーブルの作成**］をクリックします。1 つのセルを持つ新しいテーブルが作成され，引数の元の値がテーブルの 1 番目のカラムに表示されます。
- 既存のデータ・ファイルを開くには，**.dat** ファイルの名前を［**ファイル パス**］ボックスに入力するか，ドロップダウン・リストから名前を選択します。

または，［**参照**］をクリックして，既存のデータ・ファイルの場所を指定します。標準設定では，新しいデータ・ファイルはすべて＜**パラメータ名**＞**.dat** という名前で，スクリプトのディレクトリに格納されます。



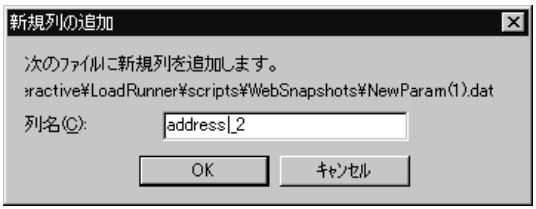
データ・ファイルが開き，最初の 100 行が表示されます。すべてのデータを表示するには，［**メモ帳で編集**］をクリックし，［メモ帳］でデータを表示します。

注：グローバル・ディレクトリを指定することもできます。グローバル・ディレクトリは，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。詳細については，136 ページ「グローバル・ディレクトリ」を参照してください。

➤ 既存のデータベースからデータをインポートするには，［**データ ウィザード**］をクリックし，ウィザードの指示に従います。詳細については，142 ページ「既存のデータベースからのデータのインポート」を参照してください。

3 列と行を テーブルに追加します。

➤ テーブルに列を追加するには，［**列の追加**］を選択します。［新規列の追加］ダイアログ・ボックスが開きます。列名を入力して［**OK**］をクリックします。

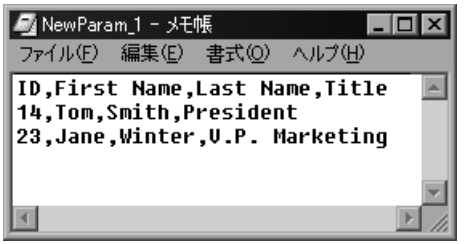


➤ テーブルに行を追加するには，［**行の追加**］を選択します。

4 データ・ファイルを編集します。

➤ 任意のセルの中で値を入力します。

➤ データ・ファイルをメモ帳の中から編集するには，［**メモ帳で編集**］をクリックします。［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最初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。



既存のデータベースからのデータのインポート

VuGen では，データベースからデータをインポートし，パラメータの値として使用できます。データをインポートする方法は 2 つあります。

➤ 新規クエリの作成

➤ SQL ステートメントの指定

VuGen では，ウィザードを実行し，その指示に従ってデータベースからデータをインポートすることができます。ウィザードで，データのインポート方法を指定します。MS Query を使って新しいクエリを作成するか，SQL ステートメントを指定します。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・ファイルとして利用できるようになります。

データベースのインポートを開始するには，まず［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**］）の［**データ ウィザード**］をクリックします。［データベース クエリ ウィザード］が開きます。

新規クエリの作成

新規クエリの作成には，Microsoft のデータベース・クエリ・ウィザードを使用します。システムに MS Query をインストールしておく必要があります。

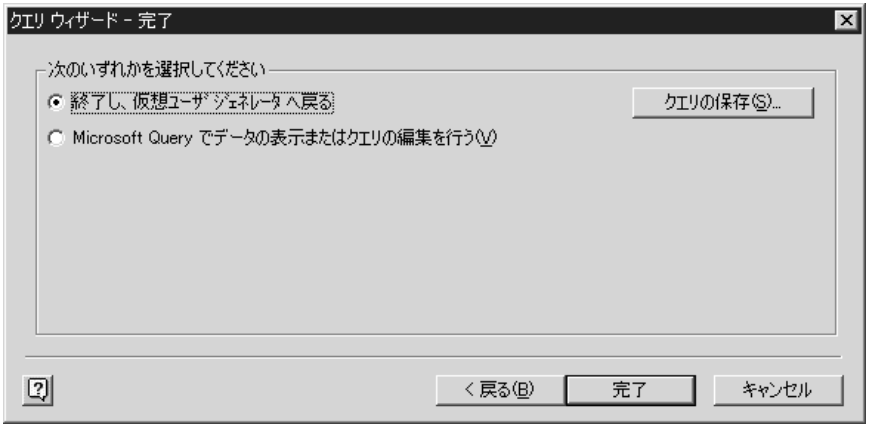
新規クエリを作成するには，次の手順を実行します。

1 ［**Microsoft Query でクエリーを作成する**］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 ［**次へ**］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，仮想ユーザ ジェネレータへ戻る**］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［パラメータのプロパティ］ボックスにデータ・ファイルとして表示されます。



終了せずに MS Query でデータを表示または編集するには，［**Microsoft Query でデータの表示またはクエリの編集を行う**］を選択します。

5 データの割り当てプロパティを設定しま。す。詳細については，146 ページ「ファイル・タイプのパラメータのプロパティ設定」を参照してください。

SQL ステートメントの指定

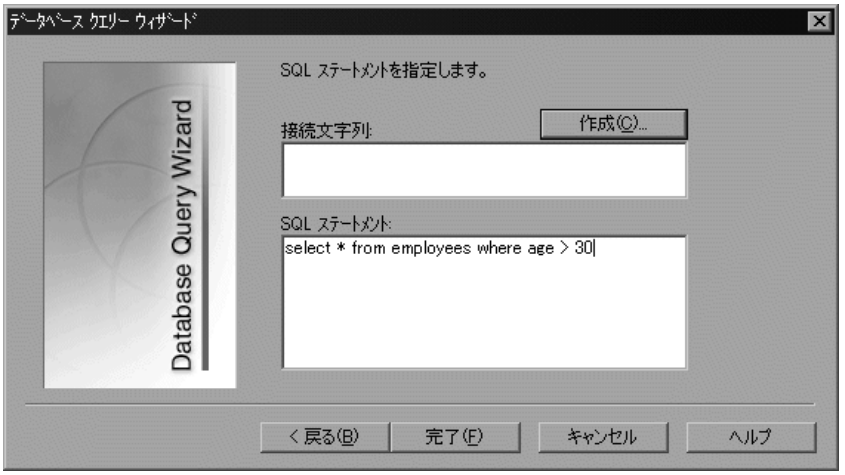
データベース接続と SQL ステートメントの指定を行うには，次の手順を実行します。

1 ［**SQL ステートメントを手作業で指定する**］を選択します。［**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するか貼り付けます。



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［パラメータのプロパティ］ボックスにデータ・ファイルとして表示されます。

6 データの割り当てプロパティを設定します。詳細については，146 ページ「ファイル・タイプのパラメータのプロパティ設定」を参照してください。

テーブルまたはファイルのデータを作成したら，割り当てプロパティを設定します。プロパティでは，使用するカラムと列を指定し，データをランダムに使用するか順次使用するかを指定します。プロパティは，ファイル・タイプ・パラメータとテーブル・タイプ・パラメータで別々に設定します。

注 : パラメータのプロパティは，［パラメータ リスト］ダイアログ・ボックスでも設定できます。左側の表示枠でパラメータを選択し，右側の表示枠でそのプロパティを指定します。詳細については，133 ページ「パラメータ・リストの使用」を参照してください。

ファイル・タイプのパラメータのプロパティ設定

データ・ソースを選択したら，ファイルの割り当てプロパティを設定します。これらのプロパティは，VuGen にデータの使用方法を指示するものです。例えば，これらのプロパティは，使用する列，新規の値を使用する頻度，一意の値がなくなったときにどうするかなどを指定します。

ファイル・タイプのパラメータのプロパティを設定するには，次の手順を実行します。

1 パラメータ用の値を含むテーブルの列番号を指定します。［**列の選択**］セクションで，列番号か列名を指定します。

列番号で指定するには，［**数順**］を選択して列番号を選びます。列番号とはデータを含んでいる列のインデックスです。例えば，パラメータのデータがテーブルの最初の列にある場合は，1 を選択します。

列名で指定するには，［**名前順**］を選択して，リストから列名を選びます。列ヘッダーは，各列の最初の行（0 行）にあります。列番号が変わる可能性がある場合，あるいはヘッダーがない場合は，列名を使って列を指定してください。

2［**ファイル形式**］セクションの［**列の区切り文字**］ボックスに，列の区切り文字を入力します。区切り文字とは，テーブルの列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）を指定できます。

3［**ファイル形式**］セクションの［**先頭データ行**］ボックスで，仮想ユーザ・スクリプト実行時に使用するデータの開始行を選択します。ヘッダーは 0 行目です。ヘッダーの後の最初の行から開始するには，「1」を指定します。ヘッダーがない場合には，「0」を指定します。

4 [**次の行**] リストから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。オプションは次のとおりです。選択肢としては，[**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，150 ページ「ファイル・タイプまたはテーブル・タイプのパラメータにデータを割り当てる方法の選択」を参照してください。

5 [**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**」（すべての出現），「**Once**」（1 回のみ）から選択します。詳細については，152 ページ「ファイル・タイプまたはテーブル・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の選択」を参照してください。

6 データの割り当て方法として [**Unique**]（一意）を選択した場合には（手順 4），次を指定します。

- [**使用可能な値の終了後**]：一意の値がなくなった場合の処理として，「**Abort the Vuser**」，「**Continue in a cyclic manner**」，「**Continue with last value**」のいずれかを指定します。
- [**コントローラで仮想ユーザの値を割り当てる**]（LoadRunner のユーザのみ）：仮想ユーザに手作業でデータ・ブロックを割り当てるかどうか指定します。コントローラにブロック・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，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が表示されます。

テーブル・タイプのパラメータのプロパティ設定

データ・テーブルを選択したら，その割り当てプロパティを設定します。これらのプロパティは，VuGen にテーブル・データの使用方法を指示するものです。例えば，これらのプロパティは，使用する列と行，それらを使用する頻度，一意の値がなくなったときにどうするかなどを指定します。

テーブル・タイプのパラメータのプロパティを設定するには，次の手順を実行します。

1 パラメータ用の値を含むテーブルの列番号を指定します。［**列**］セクションで，どの列を使用するのかを指定します。

すべての列を選択するには，［**すべて選択する**］を選択します。

1 つまたは複数の列を番号で指定するには，［**番号順に表示する**］を選択し，列番号をカンマまたはダッシュで区切って入力します。列番号とはデータを含んでいる列のインデックスです。例えば，パラメータのデータがテーブルの最初の列にある場合は，1 を選択します。

2 ［**列区切り文字**］ボックスに，カラムの区切り文字を入力します。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。

3 ［**行**］セクションで，1 回の反復でいくつの行を使用するかを［**反復ごとの行数**］ボックスに指定します。

注： 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**］（1 回のみ）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。手順 8 を参照してください。

---

4 ［**第 1 行目のデータ**］ボックスで，スクリプト実行時に使用するデータの開始行を選択します。ヘッダーの後の最初の行から開始するには，「1」を入力します。何行のデータが使用可能かなどの，テーブルに関する情報を表示するには，［**テーブル詳細**］をクリックします。

5 データ表現の行区切り文字を［**ログ表示に使用する行区切り文字**］ボックスで指定します。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。

6 ［**行数が不足する場合の対処法**］ボックスで，反復のための十分な行数がテーブルにないときの処理方法を指定します。例えば，設定するテーブルの行数が 3 で，データには 2 行しかないとします。2 つの行のみ設定する場合は，［**Parameter will get less rows than required**］を選択します。［**次の行**］ボックスの［**Random**］または［**Sequential**.］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，［**Use behavior of "Select Next Row"**］を選択します。

7 ［**次の行**］リストから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テーブル・データをどのように選択するかを指定します。選択肢としては，［**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，150 ページ「ファイル・タイプまたはテーブル・タイプのパラメータにデータを割り当てる方法の選択」を参照してください。

8 ［**更新する対象**］リストから更新方法を選択します。「**Each Iteration**」（反復ごと）または「**Once**」（1 回のみ）から選択します。詳細については，152 ページ「ファイル・タイプまたはテーブル・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の選択」を参照してください。

9［**Unique**］（一意）を使用してデータを割り当てるよう選択した場合には（手順 7），以下を指定します。

- ［**使用可能な値の終了後**］：一意の値がなくなった場合の処理として，「**Abort the Vuser**」，「**Continue in a cyclic manner**」，「**Continue with last value**」のいずれかを指定します。
- ［**コントローラで仮想ユーザの値を割り当てる**］（LoadRunner のユーザのみ）：仮想ユーザに手作業でデータ・ブロックを割り当てるかどうか指定します。コントローラにブロック・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，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が表示されます。

ファイル・タイプまたはテーブル・タイプのパラメータにデータを割り当てる方法の選択

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。

- Sequential（順次）
- Random（ランダム）
- Unique（一意）

Sequential（順次）

「**Sequential**」方式は，データを順次（シーケンシャルに）仮想ユーザに割り当てます。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。

Random（ランダム）

「**Random**」方式は，テスト実行の開始時に，データ・テーブルから値を無作為に選んで各仮想ユーザに割り当てます。

シナリオ，セッション・ステップ，またはビジネス・プロセス・モニタ・プロファイルを実行するときには，乱数の生成に使われるシード値を指定できます。この設定は，データ・ファイルの値を割り当てるためにランダム方式を使って，パラメータ化された仮想ユーザ・スクリプトに適用されます。同じシード値を使用すれば，シナリオまたはセッション・ステップの仮想ユーザには，同じ乱数値のシーケンスが割り当てられます。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，『**Mercury LoadRunner コントローラ・ユーザーズ・ガイド**』，**Tuning Console**，**Performance Center**，および **Application Management** のマニュアルを参照してください。

Unique（一意）

「**Unique**」方式は，一意のシーケンシャルな値を仮想ユーザのパラメータに割り当てます。

Unique を選択する場合は，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なければなりません。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 100 個の一意の値が必要です。

データ・テーブルに十分な値がない場合は，処理方法を VuGen に指示できます。詳細については，146 ページ「ファイル・タイプのパラメータのプロパティ設定」または 148 ページ「テーブル・タイプのパラメータのプロパティ設定」を参照してください。

ファイル・タイプまたはテーブル・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の選択

ファイル・タイプまたはテーブル・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，シナリオまたはセッション・ステップ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。

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		
	Sequential（順次）	**Random（ランダム）**	**Unique（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・テーブルから**新しい一意の**値を取得する
Each Occurrence（すべての出現）（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）	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

- ［**Sequential**］（順次）の方法を使用してデータを割り当てるよう選択した場合は，次のようになります。
  - 更新方法で［**Each Iteration**］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**］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**］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。

  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。
- ［**Random**］（ランダム）の方法を使用してデータを割り当てるよう選択した場合は，次のようになります。
  - 更新方法で［**Each Iteration**］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**］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**］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。
- ［**Unique**］（一意）の方法を使用してデータを割り当てるよう選択した場合は，次のようになります。
  - 更新方法で［**Each Iteration**］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**］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［**Once**］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。

コントローラ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，Vuser 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］:［**Unique**］

➤［更新する対象］:［**Each iteration**］（反復ごと）

➤［使用可能な値の終了後］:［**Continue with last value**］

状況	継続時間	結果の動作
値よりも反復が多い	**[完了するまで実行する]**	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。
値よりも仮想ユーザが多い	**[Run indefinitely or Run for ...]**	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「**Error: Insufficient records for param ＜パラメータ名＞ in table to provide the Vuser with unique data**」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**］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	**[Run indefinitely or Run for ...]**	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。

第 10 章

パラメータのプロパティの設定

パラメータは，パラメータによって置換される情報のタイプに従って定義されます。

本章では，次の項目について説明します。

- パラメータのプロパティの設定について
- 内部データ・パラメータ・タイプのプロパティの設定
- 内部データ・パラメータ・タイプのプロパティの設定
- ユーザ定義関数のプロパティの設定
- パラメータ形式のカスタマイズ
- 更新方法の選択

パラメータのプロパティの設定について

パラメータのプロパティを定義するときは，パラメータのデータ・ソースを指定します。次のタイプのデータ・ソースに対してプロパティを定義します。

内部データ・パラメータ・タイプ	仮想ユーザによって内部で生成されるデータ。Date/Time（日時），Group Name（グループ名），Iteration Number（反復回数），Load Generator Name（ロード・ジェネレータ名），Random Number（乱数），Unique Number（一意の数），仮想ユーザ ID があります。
ユーザ定義関数	外部の DLL の関数を使用して生成されるデータ。
データ・ファイルおよびデータ・テーブル	既存のファイルか，VuGen または MS Query で作成したファイルに含まれるデータ。

本章では，プロパティを「内部データ」および「ユーザ定義関数」のパラメータに割り当てる方法について説明します。

テーブルまたはファイル・タイプ・パラメータのプロパティの定義方法の詳細については，第 9 章「ファイルまたはテーブル・パラメータ・タイプ」を参照してください。

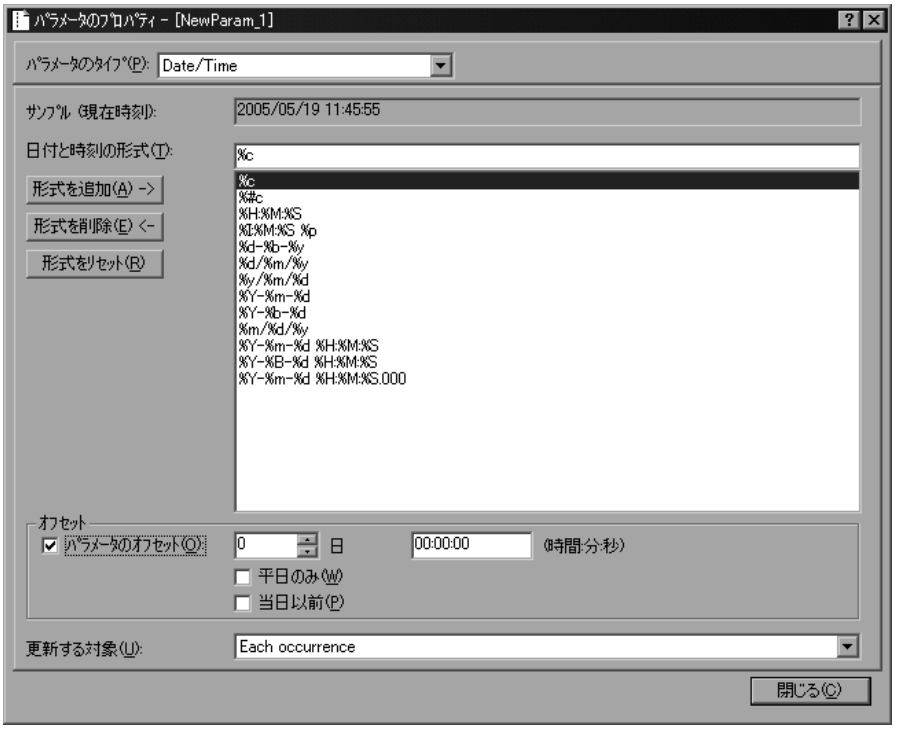
内部データ・パラメータ・タイプのプロパティの設定

本項では，仮想ユーザによって内部で生成されるデータのプロパティの設定方法について説明します。内部データには次のようなデータが含まれます。

- Date/Time（日時）
- Group Name（グループ名）
- Iteration Number（反復回数）
- Load Generator Name（ロード・ジェネレータ名）
- Random Number（乱数）
- Unique Number（一意の数）
- 仮想ユーザ ID

Date/Time（日時）

［Date/Time］（日時）を選ぶと，パラメータは現在の日付 / 時刻で置き換えられます。日時の形式は，形式リストから選択するか，独自の形式を指定します。指定した形式は，スクリプトに記録されている日時の形式と一致しなければなりません。



VuGen では日時パラメータのオフセットを設定できます。例えば，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。未来の時刻でのアプリケーションのテストを行う場合は，時刻オフセットを指定します。未来への前進オフセット（標準設定），または過去への後退オフセットを指定できます。また，平日のみの日付値を使用し，土曜日と日曜日を除外するように VuGen に指示することもできます。

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
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。例えば Dec）
B	月（省略しない文字列。例えば December）
y	年（短い形式。例えば 03）
Y	年（長い形式。例えば 2003）

日付と時刻のパラメータのプロパティを設定するには，次の手順を実行します。

1 既存の日付と時刻の形式から選択するか，新しい形式を作成します。VuGen でどのように表示されるかは［**サンプル（現在時刻）**］ボックスで確認できます。パラメータ形式のカスタマイズの詳細については，167 ページ「パラメータ形式のカスタマイズ」を参照してください。

2 日付と時刻のオフセットを設定するには，［**パラメータのオフセット**］を選択して，日付および時刻の値に対する必要なオフセットを指定します。

週末日を除く平日のみを使用するよう VuGen に指示するには，［**平日のみ**］を選択します。過去の日付をテストするために負のオフセットを指定するには，［**当日以前**］を選択します。

3［更新する対象］で「**Each occurrence**」（すべての出現），「**Each iteration**」（反復ごと），「**Once**」（1 回のみ）のどれかを選択し，仮想ユーザにパラメータ値の更新方法を指定します。詳細については，168 ページ「更新方法の選択」を参照してください。

4［**閉じる**］をクリックして設定を適用し，［パラメータのプロパティ］ダイアログ・ボックスを閉じます。

Group Name（グループ名）

［Group Name］（グループ名）を選ぶと，パラメータは仮想ユーザ・グループの名前で置き換えられます。シナリオまたはセッション・ステップの作成時に，仮想ユーザ・グループの名前を指定します。VuGen でスクリプトを実行するとき，仮想ユーザ・グループ名は常に「**None**」です。

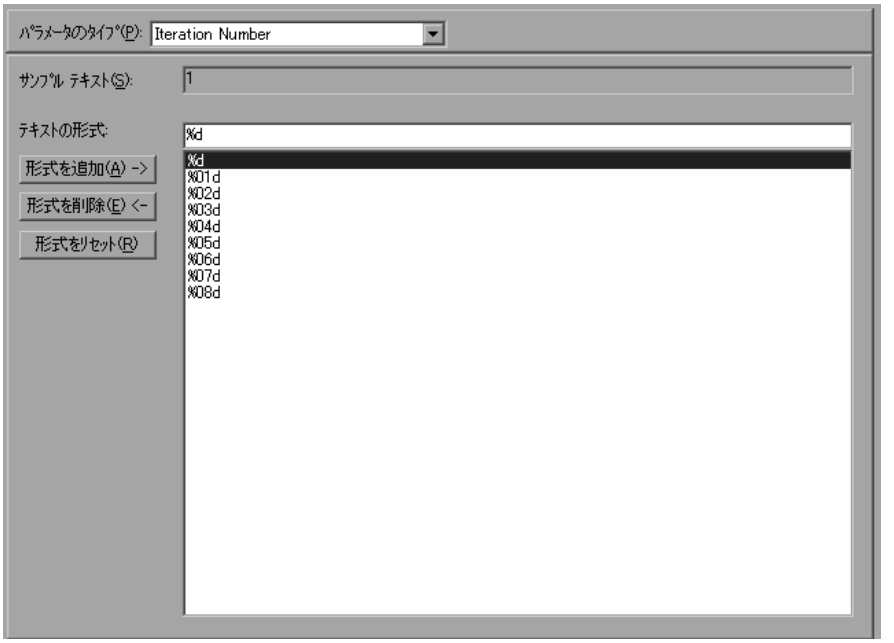
グループ名のパラメータのプロパティを設定するには，次の手順を実行します。

1 利用可能な既存の形式の中から 1 つを選択するか，新しい形式を作成します。形式を選択することによって，パラメータの文字列の長さを指定します。詳細については，167 ページ「パラメータ形式のカスタマイズ」を参照してください。

2 ［**閉じる**］をクリックして設定を適用し，［パラメータのプロパティ］ダイアログ・ボックスを閉じます。

Iteration Number（反復回数）

［Iteration Number］（反復回数）を選ぶと，パラメータは現在の反復回数で置き換えられます。



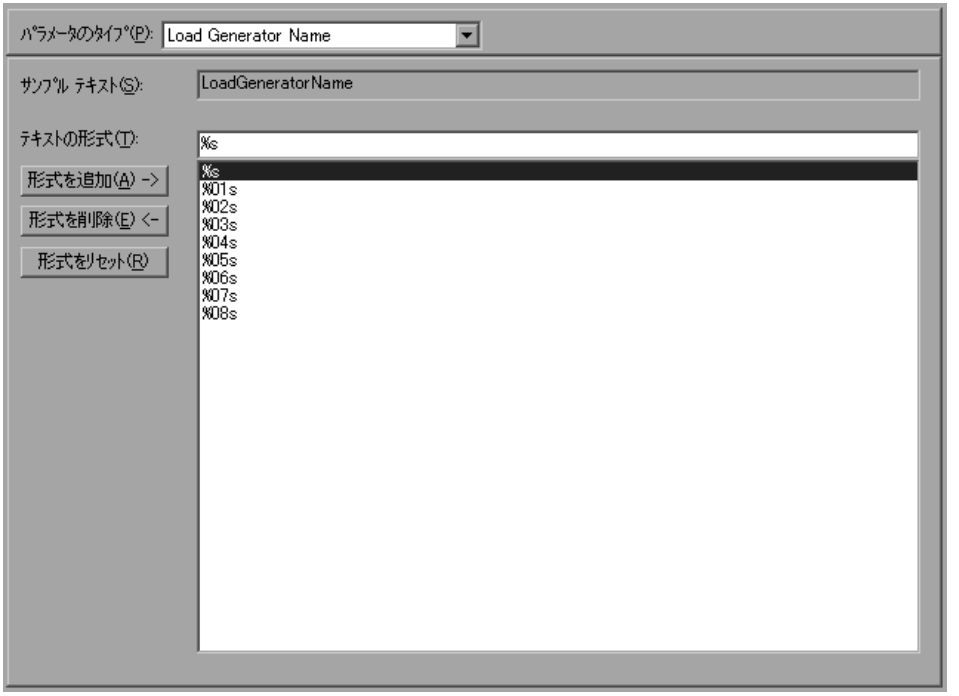
反復回数のパラメータのプロパティを設定するには，次の手順を実行します。

1 利用可能な既存の形式の中から 1 つを選択するか，新しい形式を作成します。形式を選択することによって，パラメータの文字列の長さを指定します。詳細については，167 ページ「パラメータ形式のカスタマイズ」を参照してください。

2 ［**閉じる**］をクリックして設定を適用し，［パラメータのプロパティ］ダイアログ・ボックスを閉じます。

Load Generator Name（ロード・ジェネレータ名）

［Load Generator Name］（ロード・ジェネレータ名）を選ぶと，パラメータが仮想ユーザ・スクリプトのロード・ジェネレータの名前で置き換えられます。ロード・ジェネレータとは，仮想ユーザを実行しているコンピュータのことです。



ロード・ジェネレータ名タイプのパラメータのプロパティを設定するには，次の手順を実行します。

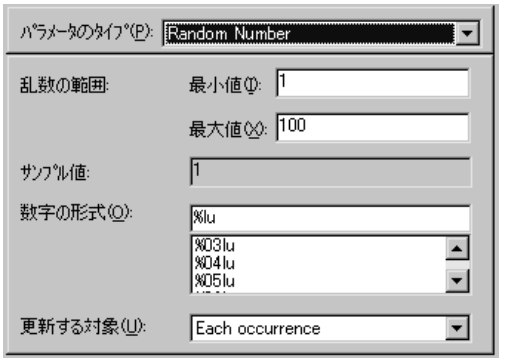
1 利用可能な既存の形式の中から 1 つを選択するか，新しい形式を作成します。形式を選択することによって，パラメータの文字列の長さを指定します。詳細については，167 ページ「パラメータ形式のカスタマイズ」を参照してください。

2 ［**閉じる**］をクリックして設定を適用し，［パラメータのプロパティ］ダイアログ・ボックスを閉じます。

Random Number（乱数）

［Random Number］（乱数）を選ぶと，パラメータは乱数で置き換えられます。最小値と最大値を指定することによって，乱数の範囲を設定します。

乱数を使用すると，特定の範囲の値を使ったときのシステムの動作をテストできます。例えば，従業員 ID 番号の範囲が 1 から 1,000 の場合に，50 人の従業員についてクエリを実行するには，50 個の仮想ユーザを作成し，最小値を 1 にし，最大値を 1,000 にします。各仮想ユーザは 1 から 1,000 までの範囲の乱数を 1 つ受け取ります。



乱数のパラメータのプロパティを設定するには，次の手順を実行します。

1 使用可能なパラメータ値のセットを規定する範囲を指定します。乱数の範囲の最小値と最大値を指定します。

2 ［**数字の形式**］を選択することによって，乱数の長さを指定します。1 桁の数字には **%01lu**（または **%lu**），2 桁の数字には **%02lu** というように指定します。VuGen でどのように表示されるかは［**サンプル値**］ボックスで確認できます。

3 ［更新する対象］で「**Each occurrence**」（すべての出現），「**Each iteration**」（反復ごと），「**Once**」（1 回のみ）のどれかを選択し，仮想ユーザにパラメータ値の更新方法を指定します。詳細については，168 ページ「更新方法の選択」を参照してください。

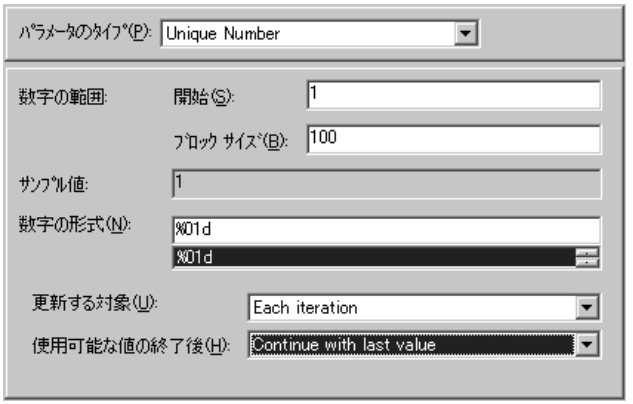
4 ［**閉じる**］をクリックして設定を適用し，［パラメータのプロパティ］ダイアログ・ボックスを閉じます。

Unique Number（一意の数）

［Unique Number］（一意の数）を選ぶと，パラメータは一意の数で置き換えられます。

一意の数のパラメータを作成した場合は，開始値とブロック・サイズを指定します。ブロック・サイズは，各仮想ユーザに割り当てられる数値の範囲を示します。各仮想ユーザはその範囲の開始値で始まり，反復ごとにパラメータの値が大きくなります。例えば開始値として 1 を，ブロック・サイズとして 500 を設定した場合，最初の仮想ユーザは値 1 を，次の仮想ユーザは値 501 を，それぞれの最初の反復で使います。

一意の数を構成する桁数とブロック・サイズによって反復の回数と仮想ユーザの数が決まります。例えば，数の桁数が 5 桁を上限としていてブロック・サイズが 500 の場合，使用できる数値は 100,000 個（0 ～ 99,999）だけとなります。したがって，1 仮想ユーザあたり 200 回の反復を実行する場合に実行できる仮想ユーザ数は 200 のみとなります。



また，ブロック内に一意の数がなくなったときにどのようなアクションを起こすかを，「**Abort Vuser**」（仮想ユーザを終了する），「**Continue in a cyclic manner**」（循環する），「**Continue with last value**」（最後の値を使い続ける－標準設定）から指定することができます。

一意の数を使えば，パラメータに対して使用可能なすべての値についてシステムの動作を確認することができます。例えば，ID 番号が 100 から 199 の範囲にある全従業員についてクエリを実行するには，100 個の仮想ユーザを作成し，開始番号を 100 に設定しブロック・サイズを 100 に設定します。各仮想ユーザには，100 から 199 までの一意の数字が ID として割り当てられます。

注：［Unique Number］（一意の数）パラメータ・タイプではインスタンスが 1 つだけ作成されます。複数のパラメータを定義してそれぞれに［Unique Number］パラメータ・タイプを指定しても，値が互いに重なることはありません。例えば，2 つのパラメータで，5 回の反復を指定し，ブロック・サイズを 100 とすると，最初のグループの仮想ユーザは 1，101，201，301，401 を使用し，2 つ目のグループは 501，601，701，801，901 を使用します。

一意の数のパラメータのプロパティを設定するには，次の手順を実行します。

1 開始値とブロック・サイズを指定します。例えば，1 から始め，500 までの数を使用したい場合，［**開始**］ボックスに 1 と入力し，［**仮想ユーザごとのブロック サイズ**］を 500 に設定します。

2 ［数字の形式］を選択することによって，一意の数の長さを指定します。1 桁の数字には **%01d**（または **%d**），2 桁の数字には **%02d** というように指定します。VuGen でどのように表示されるかは［**サンプル値**］ボックスで確認できます。

3 ［更新する対象］で「**Each occurrence**」（すべての出現），「**Each iteration**」（反復ごと），「**Once**」（1 回のみ）のどれかを選択し，仮想ユーザにパラメータ値の更新方法を指定します。詳細については，168 ページ「更新方法の選択」を参照してください。

4 一意の値がなくなった場合にどうするかを，［**使用可能な値の終了後**］ボックスで，「**Abort Vuser**」（仮想ユーザを終了する），「**Continue in a cyclic manner**」（循環する），「**Continue with last value**」（最後の値を使い続ける－標準設定）から指定することができます。

注：（LoadRunner のユーザへ）コントローラ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［**実行時間：（時間：分：秒）**］オプションに対してのみ適用されます。［**完了するまで実行する**］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。

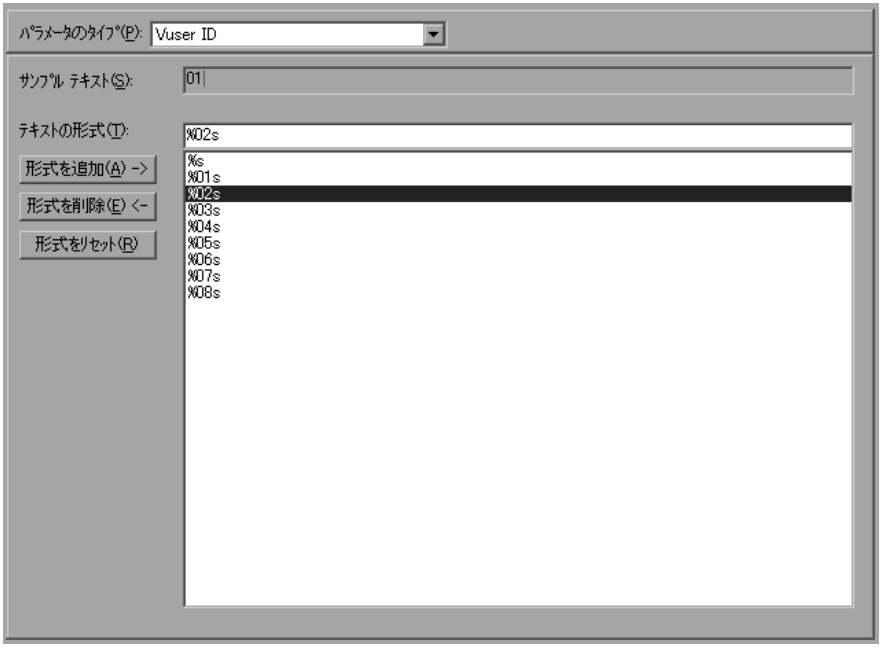
---

5 ［**閉じる**］をクリックして設定を適用し，［パラメータのプロパティ］ダイアログ・ボックスを閉じます。

仮想ユーザ ID

注：このパラメータ・タイプは主に LoadRunner のユーザが対象です。

［Vuser ID］（仮想ユーザ ID）を選ぶと，パラメータは仮想ユーザに割り当てられる ID 番号で置き換えられます。この ID 番号は，シナリオ実行中はコントローラによって割り当てられ，セッション・ステップ実行中はコンソールによって割り当てられます。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 ではなく，実行時に生成される一意の ID 番号です。

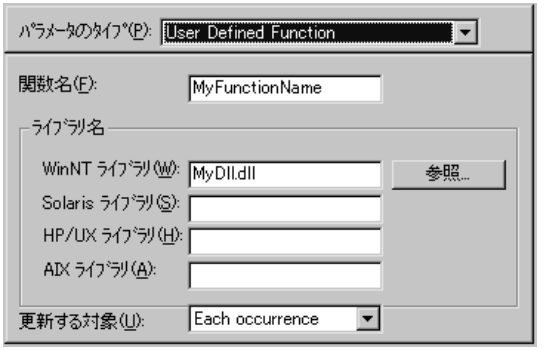
仮想ユーザ ID タイプのパラメータのプロパティを設定するには，次の手順を実行します。

1 利用可能な既存の形式の中から 1 つを選択するか，新しい形式を作成します。形式を選択することによって，パラメータの文字列の長さと構造を指定します。詳細については，167 ページ「パラメータ形式のカスタマイズ」を参照してください。

2 [**閉じる**] をクリックして設定を適用し，[パラメータのプロパティ] ダイアログ・ボックスを閉じます。

ユーザ定義関数のプロパティの設定

[パラメータのプロパティ] ダイアログ・ボックスで，[**パラメータのタイプ**] リストから [**(User Defined Function) ユーザ定義関数**] を選択します。



ユーザ定義関数のプロパティを設定するには，次の手順を実行します。

1 [**関数名**] ボックスに関数名を指定します。DLL ファイルに作成した関数名をそのまま使用します。

2 [**ライブラリ名**] セクションの，該当する [**ライブラリ**] ボックスにライブラリを指定します。必要に応じて，[**参照**] ボタンをクリックしてファイルを検索します。

3 値の更新方法を選択します。ユーザ定義関数の更新方法の詳細については，168 ページ「更新方法の選択」を参照してください。

パラメータ形式のカスタマイズ

ほとんどのデータ・タイプでは，既存の形式を選択するか新しい形式を作成することによって，パラメータの形式をカスタマイズできます。

注：パラメータの形式は，記録された値と一致させるようにします。パラメータの形式が記録されている元の値と異なると，スクリプトを正しく実行できない場合があります。

パラメータの形式は，生成されるパラメータ文字列の長さと構造を規定します。生成されるパラメータ文字列は，実際のパラメータ値と，そのパラメータに付属するテキストから成ります。例えば，%05s という形式（小数点の左に 5 桁）を指定した場合には，仮想ユーザ ID の 5 という数字は 1 桁の数字ですが，他の桁を 0 で埋めて 00005 と表示されます。数値を空白で埋めて長さを揃えるには，空白を使って揃える桁数をゼロ（0）を先頭に付けずに指定します。例えば，%4s と指定した場合，仮想ユーザ ID の先頭に空白を追加することによってパラメータ文字列が 4 桁に揃えられます。

実際のパラメータ値の前後にテキスト文字列を指定できます。

例えば，「Vuser No: %03s」という形式を指定した場合，仮想ユーザ ID の 1 番は，「**Vuser No: 001**」と表示されます。

Date/Time（日時），Group Name（グループ名），Iteration Number（反復回数），Load Generator Name（ロード・ジェネレータ名），仮想ユーザ ID の各パラメータ・タイプについて，形式を追加および削除することができます。

パラメータ・タイプの形式を追加するには，次の手順を実行します。

1 ［パラメータのプロパティ］ダイアログ・ボックスで，形式を指定する対象となるパラメータ・タイプを選択します。

2 エディット・ボックスに形式指定記号を入力し，［**形式を追加**］をクリックします。

注：形式をリストに追加すると，VuGen によって，形式が仮想ユーザとともに保存され，以後使用できるようにします。

形式を削除するには，次の手順を実行します。

［パラメータのプロパティ］ダイアログ・ボックスで，既存の形式をリストから選択し，［**形式を削除**］をクリックします。

一連の元の形式を復元するには，次の手順を実行します。

［**形式をリセット**］をクリックします。

更新方法の選択

パラメータのタイプのいくつかを使用するときは，パラメータの値を更新する方法を指定できます。更新方法を設定するには，［**更新する対象**］ドロップダウン・リストから更新方法を選択します。次の更新方法があります。

- Each Occurrence（すべての出現）
- Each Iteration（反復ごと）
- Once（1 回）

Each Occurrence（すべての出現）

「**Each occurrence**」方法を選択すると，仮想ユーザはパラメータが出現するたびに新しい値を使います。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。例えば，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）

「**Each iteration**」方法を選択すると，仮想ユーザはスクリプトで反復ごとに新しい値が使用されます。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。アクション・ブロックの詳細については，183 ページ「アクション・ブロックの作成」を参照してください。

Once（1 回）

「**Once**」の方法を選択すると，仮想ユーザはシナリオまたはセッション・ステップの実行時に，そのシナリオに対して一度だけパラメータの値を更新します。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。

第 11 章

ステートメントの相関

ステートメントを相関させることで，仮想ユーザ・スクリプトを最適化できます。VuGen の相関クエリ機能によって，ステートメントの結果を別のステートメントへの入力項目として使うことで，ステートメントを相互に結び付けることができます。

本章では，以下の項目について説明します。

- ステートメントの相関について
- C 仮想ユーザ用の相関関数の使用
- Java 仮想ユーザ用の相関関数の使用法
- WDiff を使った仮想ユーザ・スクリプトの比較
- 保存したパラメータの変更

以降の情報は，GUI を除く全タイプの仮想ユーザ・スクリプトを対象とします。

ステートメントの相関について

ステートメントを相関させる主な目的を以下に示します。

➤ コードを簡素化または最適化するため

例えば，相互に依存するクエリを連続して実行する場合，コードが非常に長くなってしまうことがあります。コードのサイズを小さくするためにクエリをネストできますが，精度が損なわれたり，コードが複雑化してわかりにくくなります。ステートメントを相関させることにより，ネストさせずにクエリを連結できます。

➤ 動的データの生成のため

多くのアプリケーションや Web サイトは現在の日時に基づいてセッションを識別します。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，これをシナリオまたはセッション・ステップ実行の全体を通して使用できます。

➤ 固有のデータ・レコードに対応するため

アプリケーション（例えばデータベース）によっては，一意の値を使用する必要があります。記録時に一意だった値も，スクリプト実行時にはもう一意ではありません。例えば，新しい銀行口座を開設するプロセスを記録したとします。各新規口座にはユーザの知らない一意の口座番号が割り当てられます。この番号は記録時に，一意のキーでなければならないという制約のあるテーブルに挿入されます。記録どおりにスクリプトを実行しようとすると，新しい一意の番号ではなく，記録された番号で口座を作成しようとします。その結果，口座番号がすでに存在するという理由でエラーとなります。

スクリプト実行時にエラーが発生した場合は，スクリプトの中でエラーが発生した場所を調べます。多くの場合，相関クエリによって，あるステートメントの結果を別のステートメントの入力として使用できるようにすれば問題を解決できます。

スクリプトの相関の主な手順は次のとおりです。

1 **相関させる値を特定する。**

データベース仮想ユーザ・スクリプトの場合，VuGen の助けを借りて相関対象を決めることができます。実行ログでエラー・メッセージをダブルクリックして，スクリプト内の問題が生じているステートメントに移動し，相関の候補となる値を探します。

あるいは，VuGen に付属の **WDiff** ユーティリティを使って，スクリプト内の不一致を調べることができます。詳細については，176 ページ「WDiff を使った仮想ユーザ・スクリプトの比較」を参照してください。

2 **結果を保存する。**

適切な関数を使って，クエリの値を変数に保存します。相関関数はプロトコルごとに異なります。相関関数の名前には通常 **web_reg_save_param** や **lrs_save_param** といった **save_param** 文字列 が含まれます。相関の方法の詳細については，各プロトコルを説明している章を参照してください。データベースや Web など，いくつかのプロトコルには，VuGen によってスクリプトに関数が自動的に追加されます。

3 **保存した値を参照する。**

クエリまたはステートメント内の定数を保存された変数で置換します。

いくつかのプロトコルには，組み込み式の自動または部分的に自動の相関機能があります。

➤ Java 言語仮想ユーザについては，『**第 2 巻 - プロトコル**』の「Java スクリプトの相関について」を参照してください。

➤ データベース仮想ユーザについては，『**第 2 巻 - プロトコル**』の「データベース仮想ユーザ・スクリプトの相関」を参照してください。

➤ Web 仮想ユーザについては，『**第 2 巻 - プロトコル**』の「Web 仮想ユーザ・スクリプトの相関ルールの設定」を参照してください。

➤ COM 仮想ユーザについては，『**第 2 巻 - プロトコル**』の「COM 仮想ユーザ・スクリプトについて」を参照してください。

C 仮想ユーザ用の相関関数の使用

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。Java，CORBA-Java，RMI-Java 仮想ユーザ用の同様の関数の詳細については，175 ページ「Java 仮想ユーザ用の相関関数の使用法」を参照してください。

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**」（[**ヘルプ**] ＞ [**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");
```

Java 仮想ユーザ用の相関関数の使用法

Java，CORBA-Java，RMI-Java 仮想ユーザ用のステートメントを相関させるには，Java 仮想ユーザ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-Java または RMI-Java スクリプトを記録するときに，VuGen は内部で相関を実行します。詳細については，『**第 2 巻 - プロトコル**』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 仮想ユーザ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
```

WDiff を使った仮想ユーザ・スクリプトの比較

相関させる値を判断するのに役立つツールとして，**WDiff** があります。このツール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，相関させるべき値を判断することができます。

他のプロトコルを使って作業をする場合は，実行ログを参照し，スクリプトが失敗した場所を特定し，その後 **WDiff** ユーティリティを使って相関させる値を調べます。

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。例えば，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。

WDiff を使って相関を検索するには，次の手順を実行します。

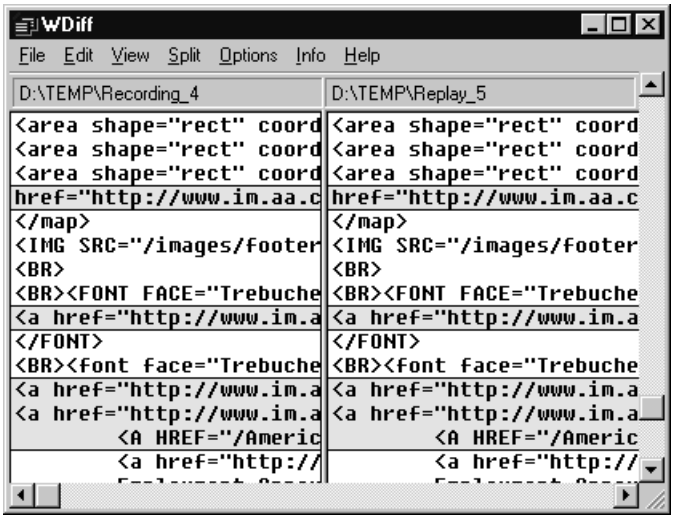
1 スクリプトを記録し，保存します。

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。

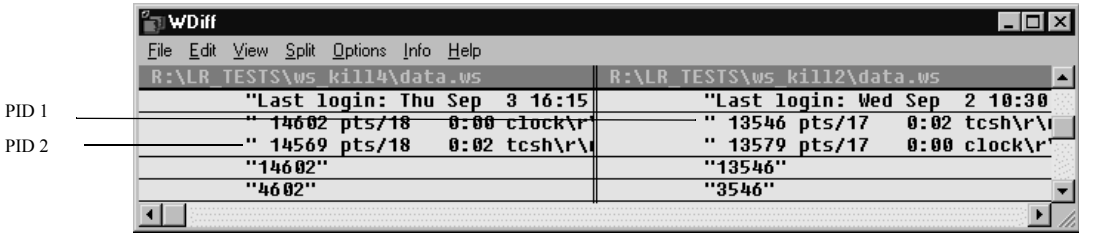
3 比較したい部分（**Actions** や **data.ws** など）を選択します。

4［**ツール**］>［**仮想ユーザと比較**］を選択します。［テストを開く］ダイアログ・ボックスが開きます。

5 比較するスクリプト（現在の VuGen ウィンドウに表示されているもの以外のスクリプト）を指定し，［**開く**］をクリックします。WDiff が開き，仮想ユーザ・スクリプト間の相違が黄色で強調表示されます。



6 相違だけを表示するには，［WDiff］ウィンドウをダブルクリックします。



7 どの値を相関させるかを判断します。

上の例では，WDiff は 2 つの Winsock 仮想ユーザ・スクリプトの **data.ws** を比較しています。この場合，相関すべき値は 2 つの記録の間で違いのある **clock** プロセスの PID です。

相関の以降の作業については，該当する項を参照してください。

- Java 言語仮想ユーザについては，『**第 2 巻 - プロトコル**』の「Java スクリプトの相関」を参照してください。
- データベース仮想ユーザについては，『**第 2 巻 - プロトコル**』の「データベース仮想ユーザ・スクリプトの相関」を参照してください。
- Web 仮想ユーザについては，『**第 2 巻 - プロトコル**』の「Web 仮想ユーザ・スクリプトの相関ルールの設定」を参照してください。
- COM 仮想ユーザについては，『**第 2 巻 - プロトコル**』の「COM 仮想ユーザ・スクリプトについて」を参照してください。
- Tuxedo 仮想ユーザについては，『**第 2 巻 - プロトコル**』の「Tuxedo 仮想ユーザ・スクリプトの作成」を参照してください。
- WinSock 仮想ユーザについては，『**第 2 巻 - プロトコル**』の「Windows Sockets データを使った作業」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"));
```

第 12 章

実行環境の設定

仮想ユーザ・スクリプトを記録した後で，スクリプトの実行環境を設定します。これらの設定は，実行時のスクリプトの振る舞いを指定します。

本章では，次の項目について説明します。

- 実行環境の設定について
- 実行論理の設定（マルチ・アクション）
- ペースの設定
- 実行環境のペースの設定（マルチ・アクション）
- ペースの設定と実行論理オプションの設定（シングル・アクション）
- 実行環境設定のログの設定
- 思考遅延時間の設定
- 実行環境の追加属性の設定
- その他の設定
- VB 実行環境の設定

以降の情報は，GUI を除く全タイプの仮想ユーザ・スクリプトを対象とします。

実行環境の設定について

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，コントローラ，Tuning Console，または Administration Console 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。例えば，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。

実行環境設定の表示と設定は，［実行環境設定］ダイアログ・ボックスを使って行います。これらの設定は，次のいずれかの方法で開くことができます。

- VuGen ツールバーの［**実行環境の設定**］ボタンをクリックします。
- キーボードのショートカット・キー，**F4** キーを押します。
- ［**仮想ユーザ**］＞［**実行環境の設定**］を選択します。

実行環境の設定は，LoadRunner コントローラまたは Tuning Module から変更することもできます。詳細については，各製品のマニュアルを参照してください。

注：LoadRunner の場合は，標準の実行環境の設定で VuGen のデバッグ環境とコントローラの負荷テスト環境がサポートされます。この標準の設定は，次のとおりです。

［**思考遅延時間**］：VuGen では無効になっていますが，コントローラでは記録されたとおりに再生されます。

［**ログ**］：VuGen では標準で有効になっていますが，コントローラでは無効になっています。

［**HTML 以外のリソースをダウンロードする**］：VuGen とコントローラの両方で有効になっています。

本章で説明する一般的な実行環境設定は，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。次の設定があります。

- 実行論理（反復）
- ペースの設定
- ログ
- 思考遅延時間
- その他
- 追加属性（Tuning Module のみ）

WinSock やデータベース（Oracle 2-tier，Sybase，MSSQL など）など，複数のアクションをサポートしないプロトコルでは，反復オプションとペースの設定オプションはどちらもペースの設定ノードで処理できます。多くのプロトコルには，追加の実行環境設定があります。これらのプロトコルの仮想ユーザ固有の実行環境の設定については，該当する項を参照してください。

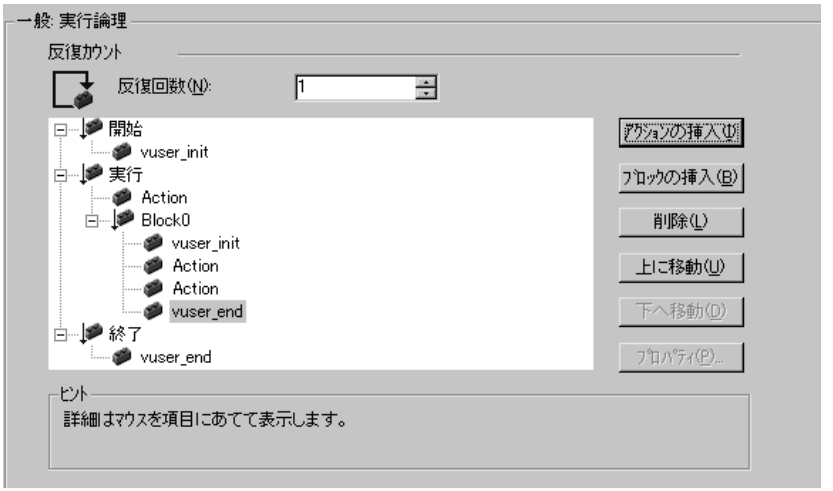
実行論理の設定（マルチ・アクション）

注：次の項の内容は，複数のアクションで動作するプロトコルを対象としています。［実行環境設定］の［一般］の下に［**実行論理**］ノードがあれば，複数アクション対応のプロトコルです。シングル・アクション・プロトコルについては，188 ページ「ペースの設定と実行論理オプションの設定（シングル・アクション）」を参照してください。

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。

［実行環境設定］ダイアログ・ボックスを開き，［**一般：実行論理**］ノードを選択します。



反復回数：反復の回数です。仮想ユーザは，すべての Action セクションを指定した回数だけ繰り返し実行します。

注：LoadRunner コントローラと Tuning Module では，スケジュールの設定でシナリオまたはセッション・ステップの継続期間を指定すると，その設定が仮想ユーザの反復の設定に優先します。つまり，この継続時間が 5 分（標準設定）に設定してあると，実行環境の設定で反復回数を 1 回に設定しても，仮想ユーザは 5 分間反復し続けます。

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。各ブロックのプロパティ（順序，反復，重み付け）を個別に設定できます。

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。

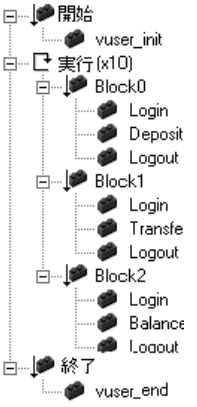
反復：「**実行**」セクション全体の反復回数を設定する以外に，各アクションまたはアクション・ブロックの反復回数を設定することもできます。これは例えば，製品を探すのに多数のクエリを実行するけれども，購入する場合は 1 回だけというような商用サイトでの操作のエミュレーションに役立ちます。

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。

アクション・ブロックの作成

アクション・ブロックは，仮想ユーザ・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。



各ブロックの順序，反復を個別に設定します。

アクションとアクション・ブロックを設定するには，次の手順を実行します。

1 記録またはプログラミングによって，必要なアクションをすべて作成します。

2 ［実行環境設定］ダイアログ・ボックスを開きます。［**一般**：**実行論理**］ノードを選択します。

3 新規アクション・ブロックを追加します。［**ブロックの挿入**］をクリックします。VuGen によって，使用可能な次のインデックスが付いた新しいアクション・ブロック（**Block0**，**Block1**，**Block2**）が挿入ポイントに挿入されます。

4 ブロックにアクションを追加します。［**アクションの挿入**］をクリックします。［アクションを選択］リストが開きます。

5 ブロックに追加するアクションを選択し，［**OK**］をクリックします。VuGen によって，現在のブロックまたはセクションに新規アクションが挿入されます。

6 ブロックに追加するアクションごとに，手順 3 を繰り返します。

7 アクションまたはアクション・ブロックを削除するには，対象を選択して［**削除**］ボタンをクリックします。

8 項目の位置を変えるには，［**上に移動**］または［**下へ移動**］をクリックします。

9 反復回数とアクションの実行論理を設定するには，［**プロパティ**］をクリックします。各アクションまたはアクション・ブロックのプロパティ・ダイアログ・ボックスが開きます。

10 ［**実行論理**］リストから［**Sequential**］または［**Random**］を選択します。これにより VuGen に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します。

11 ［**反復**］ボックスで反復の回数を指定します。アクション・ブロック内でパラメータを定義し，反復ごとにパラメータ値を更新するよう VuGen に指示した場合の「反復ごと」というのは個々のブロックの反復ではなくグローバルな反復のことです。

12 ［**OK**］をクリックします。

13 実行論理が「Random」のブロックには，各アクションの重み付けを設定します。アクションを右クリックして，［**プロパティ**］を選択します。［アクションのプロパティ］ダイアログ・ボックスが開きます。

選択したブロックまたはアクションに対して，必要な割合を指定します。［**乱数率**］ボックスで，対象アクションの割合を指定します。割合の合計は 100% にならなければなりません。

14 プロパティを設定する各要素に対して，上記の手順を繰り返します。

ペースの設定

注：次の項の内容は，複数のアクションで動作するプロトコルを対象としています。［実行環境設定］の［一般］の下に［**実行論理**］ノードがあれば，複数アクション対応のプロトコルです。シングル・アクション・プロトコルについては，188 ページ「ペースの設定と実行論理オプションの設定（シングル・アクション）」を参照してください。

［実行環境の設定］の［ペースの設定］を設定することで，反復の間隔を変えることができます。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。各反復の開始間隔として次のオプションを指定できます。

- 前回の反復が終了次第すぐ
- 前回の反復が終了後，一定 / 値の範囲 遅延間隔 X 秒
- 一定 / 値の範囲 間隔 各 X 秒

［**前回の反復が終了次第すぐ**］：直前の反復の終了後，すぐに次の反復を開始します。

［**前回の反復が終了後**］，［**一定**］/［**値の範囲**］［遅延間隔］［X 秒］：直前の反復の終了後，指定した時間が経過したら，次の反復を開始します。正確な秒数または時間の範囲を指定します。例えば，直前の反復が終了してから 60 ～ 90 秒後に次の反復を開始するように指定できます。

スクリプト実行時に，反復が終了してから次の反復を開始するまでに仮想ユーザが待機した時間は，VuGen の実行ログに示されます。

［**一定**］/［**値の範囲**］［間隔］［各 X 秒］：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。例えば，新しい反復を 30 秒ごとに開始したり，30 ～ 45 秒ごとのランダムな間隔で開始したりするように指定できます。反復は，直前の反復が終了してからのみ開始します。

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。スクリプト実行時に，反復が終了してから次の反復を開始するまでに仮想ユーザが待機した時間は，VuGen の実行ログに示されます。

例えば，4 秒ごとに新しい反復を開始するように定義すると，次のようになります。

- 最初の反復が 3 秒かかると，仮想ユーザは次の反復で 1 秒待機します。
- 最初の反復が 2 秒かかると，仮想ユーザは 2 秒待機します。
- 最初の反復が 8 秒かかると，次の反復は，最初の反復が開始してから 8 秒後に開始します。VuGen の実行ログには，反復のペースを守れなかったことを示すメッセージが表示されます。

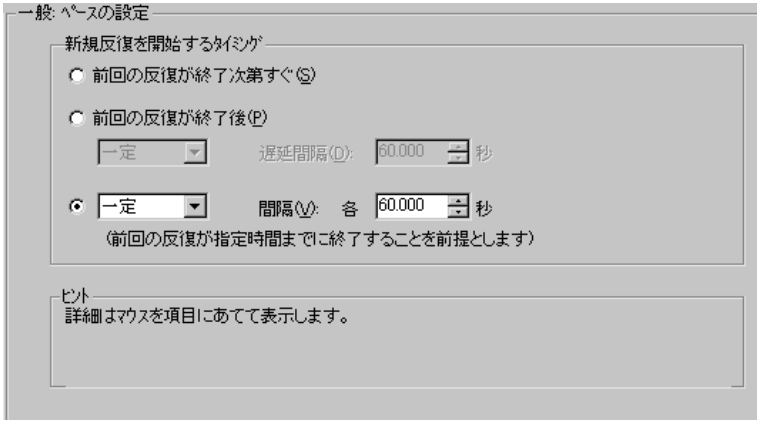
ペースの設定のオプションの詳細については，187 ページ「実行環境のペースの設定（マルチ・アクション）」を参照してください。

実行環境のペースの設定（マルチ・アクション）

ペースの設定のオプションを使用して，アクションの反復の間隔を設定することでアクションのペースを設定できます。

反復の間隔を設定するには，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，［**一般：ペースの設定**］ノードを選択します。



2 ［**新規反復を開始するタイミング**］セクションで，次のいずれかを選択します。

- ［前回の反復が終了次第すぐ］
- ［前回の反復が終了後］
- ［一定］または［ランダム］な間隔

3 ［**前回の反復終了後**］オプションには次の設定があります。

- 遅延の種類を［**一定**］か［**ランダム**］から選択します。
- ［一定］で値を指定するか，［ランダム］で値の範囲を指定します。

4 ［**各 … 秒**］オプションには次の設定があります。

- 間隔の種類を［**一定**］か［**値の範囲**］から選択します。
- ［一定］で値を指定するか，［ランダム］で値の範囲を指定します。

5 ［**OK**］をクリックします。

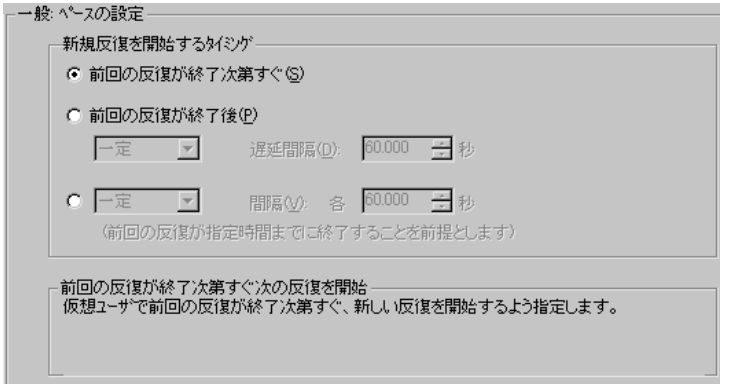
ペースの設定と実行論理オプションの設定（シングル・アクション）

注：次の項の内容は，複数のアクションではなく単一のアクションを使用するプロトコルを対象としています。［実行環境設定］の［一般］の下に［**ペースの設定**］ノードと［**実行論理**］ノードがなければ，単一アクションのプロトコルです。

仮想ユーザがスクリプトの実行時に「**Action**」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。

反復とペースの設定を設定するには，次の手順を実行します。

1 VuGen ツールバーの［**実行環境の設定**］ボタンをクリックするか，［**仮想ユーザ**］＞［**実行環境の設定**］を選択して，［実行環境設定］ダイアログ・ボックスを表示します。［**ペースの設定**］ノードをクリックして，反復とペースのオプションを表示します。



2 ［**反復回数**］ボックスで反復の回数を指定します。仮想ユーザは，すべての Action セクションを指定した回数だけ繰り返し実行します。

3［**新規反復を開始するタイミング**］セクションで，次のいずれかを選択します。

➤［前回の反復が終了次第すぐ］

➤［前回の反復が終了後］

➤［一定］または［ランダム］な間隔

4［**前回の反復終了後**］オプションには次の設定があります。

➤ 遅延の種類を［**一定**］か［**ランダム**］から選択します。

➤［一定］で値を指定するか，［ランダム］で値の範囲を指定します。

5［**各 … 秒**］オプションには次の設定があります。

➤ 間隔の種類を［**一定**］か［**値の範囲**］から選択します。

➤［一定］で値を指定するか，［ランダム］で値の範囲を指定します。

6［**OK**］をクリックします。

ペースの設定のオプションの概要については，185 ページ「ペースの設定」を参照してください。

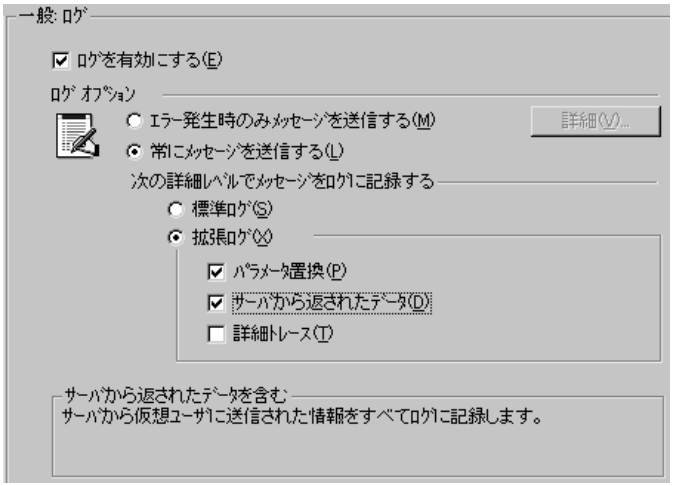
実行環境設定のログの設定

実行中，仮想ユーザは自身に関する情報と，サーバとの通信に関する情報をログに書き込みます。Windows 環境では，この情報はスクリプトのディレクトリにある **output.txt** というファイルに格納されます。UNIX 環境では，この情報は標準出力に送られます。ログの情報は，デバッグの際に役立ちます。

ログの実行環境を設定して，出力へ送られるログ記録の情報量を指定できます。**標準**ログまたは**拡張**ログを選択できます。また，ログの記録を一切無効にすることもできます。ログを無効にすることは，多数の仮想ユーザを実行する場合に有用です。何十，何百という仮想ユーザが実行時の情報をディスクに記録すると，システムの速度が通常より遅くなる場合があります。開発時は，再生に関する情報を得られるようにログ記録を有効にしておきます。ログの記録を無効にするときは，その前に必ずスクリプトが正しく動作することを確認してください。

注：lr_error_message 関数と **lr_output_message** 関数を使って，出力ログにメッセージを送信するように仮想ユーザ・スクリプトをプログラミングできます。

［**実行環境の設定**］ボタンをクリックするか，［**仮想ユーザ**］＞［**実行環境の設定**］を選択して，［実行環境設定］ダイアログ・ボックスを表示します。［**一般：ログ**］タブをクリックして，ログのオプションを表示します。



ログを有効にする

このオプションは，再生中の自動ログ記録を有効にします。つまり，VuGen によって実行ログにログ・メッセージが書き込まれます。このオプションは，自動ログ記録と **lr_log_message** を通じて発行されるログ・メッセージだけを対象とします。**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

実行環境設定のログの設定では，開発の段階に応じてログ記録のレベルを調整できます。

ログ・メッセージをログに送信する条件を指定できます。［**エラー発生時のみメッセージを送信する**］または［**常にメッセージを送信する**］のどちらかを選択します。開発中には，すべてのログ記録を有効にできます。スクリプトのデバッグが終了し，正しく動作することを確認したら，エラー・ログのみを有効にできます。

エラーの発生時にのみメッセージを送信する場合（JIT（ジャスト・イン・タイム）メッセージングとも呼ばれる）は，ログのキャッシュ・サイズなど，追加の詳細オプションを設定できます。193 ページ「ログのキャッシュ・サイズの設定」を参照してください。

ログ詳細レベルの設定

ログに記録される情報の種類を指定したり，すべてのログ記録を無効にしたりできます。

注 :［**実行環境設定**］ダイアログ・ボックスの［**一般**］タブで［**エラー処理**］に［エラーでも処理を継続する］を設定している場合でも，エラー・メッセージは出力ウィンドウに送信されます。スクリプトのログ詳細レベルを変更しても，**lr_message**，**lr_output _message**，および **lr_log_message** の各関数の動作は変更されず，メッセージは送信され続けます。

［**標準ログ**］: スクリプトの実行中に実行された関数や送信されたメッセージの標準ログが生成されます。これらの標準ログはデバッグで使用します。大規模な負荷テスト・シナリオ，チューニング・セッション，またはプロファイルでは，このオプションは無効にしてください。

ログの記録レベルを「**標準ログ**」に設定したスクリプトをシナリオ，セッション・ステップ，またはプロファイルに追加した場合，ログ記録モードは自動的に［**JIT ログ記録**］に設定されます。ただし，ログ記録モードが無効になっているか，［**拡張ログ**］に設定されている場合は，スクリプトをシナリオ，セッション・ステップ，またはプロファイルに追加しても，ログの設定には何の影響もありません。

［**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。大規模な負荷テスト・シナリオ，チューニング・セッション，またはプロファイルでは，このオプションは無効にしてください。

［拡張ログ］オプションでは，拡張ログにどの情報を追加するかを指定できます。

- ［**パラメータ置換**］：このオプションを選択すると，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。パラメータの詳細については，第 8 章「VuGen パラメータを使った作業」を参照してください。
- ［**サーバから返されたデータ**］：このオプションを選択すると，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**］：このオプションを選択すると，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。このオプションは，仮想ユーザ・スクリプトをデバッグするときに役立ちます。

VuGen によるイベントのログの記録の程度（標準，パラメータ置換など）は「**メッセージ・クラス**」ともいいます。メッセージ・クラスには，標準（Brief），詳細（Extended），パラメータ（Parameters），結果データ（Result Data），完全トレース（Full Trace）の 5 つがあります。

手作業でスクリプト内にメッセージ・クラスを設定する場合は，**lr_set_debug_message** 関数を使用します。これは，スクリプトのごく一部分だけについてデバッグ情報を受け取る場合などに便利です。

例えば，ログの設定を［標準ログ］に設定し，スクリプトの特定のセクションについて拡張ログを取得するとします。この場合は，**lr_set_debug_message** 関数を使って，スクリプト内の対象の場所でメッセージ・クラスを［拡張］に設定します。拡張モードのタイプ（Parameter，Result Data，Full Trace）を指定するときは，この関数をもう一度呼び出す必要があります。標準ログ・モードに戻るには，**lr_set_debug_message** 関数を呼び出して標準モードを指定します。メッセージ・クラスの設定の詳細については，「**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。

ログのキャッシュ・サイズの設定

実行環境のログ設定の詳細オプションでは，ログのキャッシュ・サイズを指定できます。ログのキャッシュには，テスト実行に関する未処理のデータが格納され，エラーが生じた場合に参考にできます。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。標準設定のサイズは 1 KB です。

ログ記録は，次のように行われます。

1 エラーが生じたときにだけメッセージをログに記録するには，［**エラー発生時のみメッセージを送信する**］を選択します。

2 テスト実行に関する情報がファイルに書き出されずに，ログのキャッシュに格納されます。この情報が 1KB を超えると，古いデータから上書きされます。［実行ログ］タブは，ログ・ファイルの内容を表示するので，この場合には空のままとなります。

3 エラーが発生すると（内部エラーか **lr_error_message** を使用したプログラムによるものかを問わず），キャッシュの内容がログ・ファイルと［実行ログ］タブに移されます。これを参照することで，エラーが発生するまでの状況を確認できます。

エラーが発生して VuGen によってキャッシュの内容をログ・ファイルに書き出されると，ログ・ファイルのサイズはキャッシュのサイズよりも大きくなります。例えば，キャッシュ・サイズが 1KB ならば，ログ・ファイルのサイズは 50 KB になることもあります。これは正常な動作であり，未処理のデータをわかりやすい文章に整えるために必要なオーバーヘッドを反映しているに過ぎません。

JIT モードの場合，**lr_message** 関数および **lr_log_message** 関数の出力は，エラー発生時に出力がログのキャッシュにあった場合にのみ，［出力］ウィンドウおよびログ・ファイルには送られます。個々のメッセージ文字列については，［実行ログ］を参照してください。

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。

思考遅延時間の設定

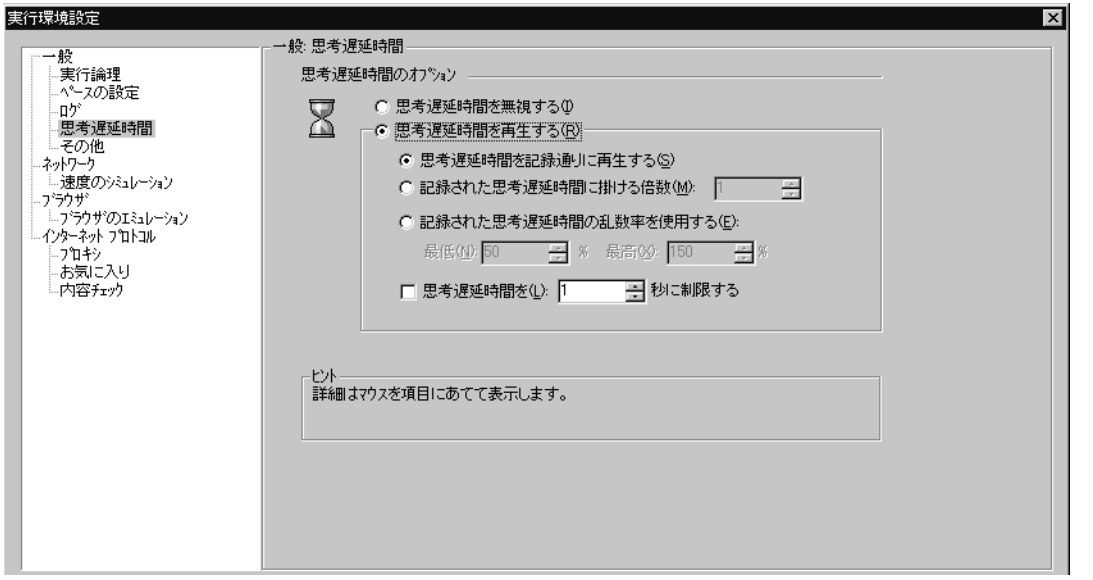
仮想ユーザの**思考遅延時間**は，実際のユーザがアクションとアクションの間で待つ時間をエミュレートしたものです。例えば，ユーザはサーバからデータを受け取ったときに，データを数秒間かけて確認してから応答する可能性もあります。この遅れを「**思考遅延時間**」といいます。VuGen は **lr_think_time** 関数を使って，仮想ユーザ・スクリプトに思考遅延時間の値を記録します。次に示す記録された関数は，次のアクションを実行する前にユーザが 8 秒待機したことを示します。

```
lr_think_time(8);
```

標準設定では，仮想ユーザ・スクリプトを実行して上記の **lr_think_time** ステートメントに到達すると，仮想ユーザは次のアクションを実行する前に 8 秒間待機します。思考遅延時間の設定を行うことにより，記録された思考遅延時間をスクリプト実行時に仮想ユーザにどのように適用するかを設定できます。

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。

VuGen ツールバーの［**実行環境の設定**］ボタンをクリックするか，［**仮想ユーザ**］＞［**実行環境の設定**］を選択して，［実行環境設定］ダイアログ・ボックスを開きます。［**一般 ：思考遅延時間**］ノードをクリックして，思考遅延時間のオプションを表示します。



思考遅延時間のオプション

標準設定では，仮想ユーザ・スクリプトを実行すると，仮想ユーザは記録セッション中にスクリプトに記録された思考遅延時間の値を使用します。VuGen では，記録された思考遅延時間の使用，思考遅延時間の無視，記録された思考遅延時間を基に指定した値の使用が可能です。

［**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。

［**思考遅延時間を再生する**］: 2 つ目の思考遅延時間オプションを選択すると，記録された思考遅延時間を使用できます。

➤［**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。例えば，lr_think_time(10) は 10 秒間待機します。

- ［**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。例えば，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。
- ［**記録された思考遅延時間の乱数率を使用する**］：記録された思考遅延時間のランダムな割合（%）を使用します。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。例えば，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。
- ［**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。

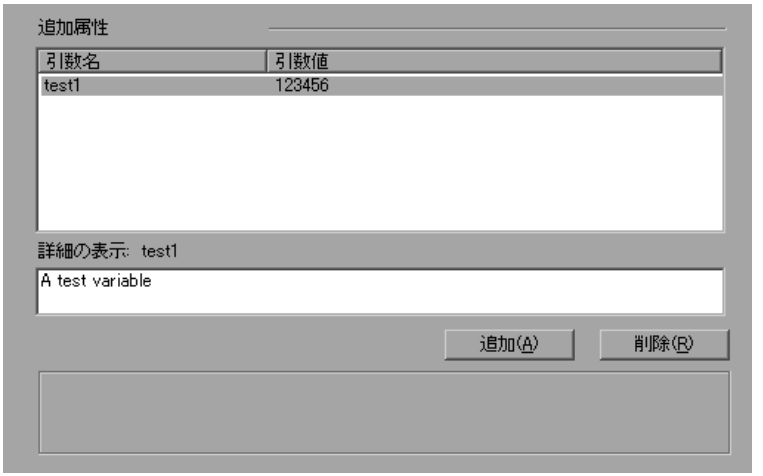
実行環境の追加属性の設定

Mercury Tuning Module では，［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パラメータを渡すことにより，異なるクライアント・パラメータを使ってサーバのテストや監視を実行できます。

追加属性を設定するには，次の手順を実行します。

1 ［実行環境の設定］ボタンをクリックするか，［**仮想ユーザ**］＞［**実行環境の設定**］を選択して，［実行環境設定］ダイアログ・ボックスを表示します。左の表示枠のツリーで［**一般：追加属性**］ノード選択します。



2 ［**追加**］をクリックして，新しいコマンド・ライン引数のエントリを追加します。属性の名前と値を入力します。

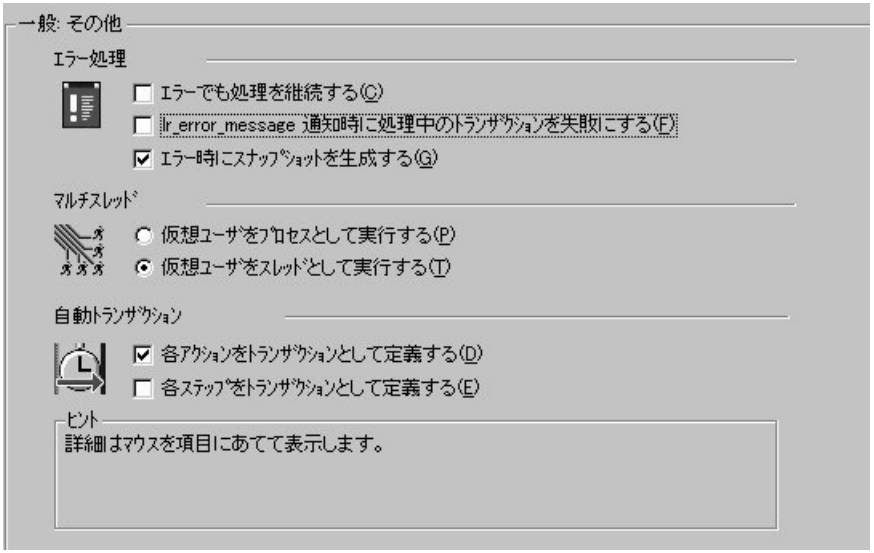
3 選択した引数を削除するには，［**削除**］をクリックします。

その他の設定

仮想ユーザ・スクリプトでは，次に示すその他の実行環境オプションを設定できます。ただし，マルチ・スレッドと自動トランザクションのオプションは Application Management ツールには適用されません。

- エラー処理
- マルチ・スレッド
- 自動トランザクション

［**実行環境の設定**］ボタンをクリックするか，［**仮想ユーザ**］＞［**実行環境の設定**］を選択して，［実行環境設定］ダイアログ・ボックスを表示します。左の表示枠のツリーで［**一般：その他**］ノード選択します。



［**その他**］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。

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仮想ユーザがスクリプト実行を継続するように指定します。標準ではこのオプションは無効になっているため，エラーが発生すると仮想ユーザは終了します。

［**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に指定します。**If** ステートメントをプログラミングし，ある条件が満たされたときに **lr_error_message** 関数が発行されるようにします。

［**エラー時にスナップショットを生成する**］：エラー発生時にスナップショットを生成します。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。

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。

データベース仮想ユーザのエラー処理

データベース・プロトコル（LRD）を使っているときに，スクリプトの特定のセグメントに対するエラー処理を制御できます。対象のセグメントを指定するには，セグメントを LRD_ON_ERROR_CONTINUE ステートメントと LRD_ON_ERROR_EXIT ステートメントで囲みます。仮想ユーザによって指定したセグメント全体に新しいエラー設定が適用されます。［エラーでも処理を継続する］を指定すると，VuGen がエラーに遭遇したとき，それを無視したことを示すメッセージが発行されます。

例えば，［エラーでも処理を継続する］機能を有効にしていて，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，スクリプトの実行は継続されます。

```
lrd_stmt(Csr1, "select.....);
lrd_exec(...);
```

仮想ユーザに，特定のセグメントでエラーが発生した場合を除き，エラーが発生してもスクリプト全体の実行を継続するように指示するには，［**エラーでも処理を継続する**］オプションを選択し，除外するセグメントを LRD_ON_ERROR_EXIT ステートメントと LRD_ON_ERROR_CONTINUE ステートメントで囲みます。

```
LRD_ON_ERROR_EXIT;
lrd_stmt(Csr1, "select.....);
lrd_exec(...);
LRD_ON_ERROR_CONTINUE;
```

LRD_ON_ERROR ステートメントを使用する以外に，「**重要度のレベル**」に基づいてエラー処理を制御する方法もあります。LRD_ON_ERROR ステートメントは，データベース関連，不正なパラメータなど，すべてのタイプのエラーを検出します。データベース操作のエラー（エラー・コード 2009）が発生したときにだけ，仮想ユーザを終了させたい場合には，関数の重要度のレベルを設定します。データベース操作を実行するすべての関数では，関数の最後のパラメータである **miDBErrorSeverity** で示される重大度レベルが使用されます。

VuGen では以下の重要度のレベルがサポートされています。

定義	意味	値
LRD_DB_ERROR_SEVERITY_ERROR	データベース・アクセス・エラーが生じた時点で，スクリプトの実行を終了します（標準設定）。	0
LRD_DB_ERROR_SEVERITY_WARNING	データベース・アクセス・エラーが生じてもスクリプトの実行を継続しますが，警告を出します。	1

例えば，次のデータベース・ステートメントが失敗した場合（例えば，テーブルが存在していないなど），スクリプトの実行は終了します。

```
lrd_stmt(Csr1, "insert into EMP values ('Smith',301)¥n", -1, 1, 1, 0);
```

データベース操作エラーが発生しても，スクリプトの実行を続けるようにするには，ステートメントの重要度を 0 から 1 に変えます。

```
lrd_stmt(Csr1, "insert into EMP values ('Smith',301)¥n", -1, 1, 1, 1);
```

注 : [エラーでも処理を継続する] を有効にすると，その設定が重要度「0」に優先し，データベース・エラーが発生したときでも，スクリプトの実行は継続されます。また［エラーでも処理を継続する］を無効にしても，重要度を「1」に指定すると，データベース・エラーが発生してもスクリプトの実行は継続されます。

RTE 仮想ユーザのエラー処理

RTE 仮想ユーザを使用する場合は，個々の関数のエラー処理を制御できます。動作を変える関数の手前に **lr_contue_on_error(0);** ステートメントを挿入します。仮想ユーザは，スクリプト実行の終了まで，または別の **lr_continue_on_error** ステートメントが発行されるまで，新しい設定を使用します。

例えば，［エラーでも処理を継続する］機能を有効にしていて，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，スクリプトの実行は継続されます。

```
TE_wait_sync();
TE_type(...);
```

特定のセグメントを除くスクリプト全体で，エラーが発生しても仮想ユーザが実行を継続するように指定するには，［**エラーでも処理を継続する**］オプションを選択し，次のように除外するセグメントを **lr_continue_on_error** ステートメントで囲み，0 を渡して［エラーでも処理を継続する］オプションを無効にし，1 を渡して有効にします。

```
lr_continue_on_error(0);
TE_wait_sync();
lr_continue_on_error(1);
....
```

マルチ・スレッド

仮想ユーザではマルチ・スレッド環境がサポートされています。マルチ・スレッド環境の主な利点は，ロード・ジェネレータごとに多数の仮想ユーザを実行できることです。スレッドとして実行できるのは，スレッドセーフのプロトコルだけです（Application Management ツールには適用されません）。

注：Sybase-CtLib，Sybase-DbLib，Informix，Tuxedo，PeopleSoft-Tuxedo の各プロトコルはスレッドセーフではありません。

➤ マルチ・スレッドを実行するには，［**仮想ユーザをスレッドとして実行する**］をクリックします。

➤ マルチ・スレッドを無効にして，各仮想ユーザを個別のプロセスとして実行するには，［**仮想ユーザをプロセスとして実行する**］をクリックします。

コントローラと Tuning Console は，ドライバ・プログラム（例えば **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，ロード・ジェネレータ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，コントローラや Tuning Console 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 台のロード・ジェネレータで実行できる仮想ユーザの数を増やせます。

自動トランザクション

LoadRunner または Tuning Console が仮想ユーザ・スクリプト内の各ステップまたはアクションをトランザクションとして処理するように指定できます（Application Management ツールには適用されません）。これを「自動トランザクションの使用」といいます。LoadRunner や Tuning Console は，ステップ名またはアクション名をトランザクションの名前として割り当てます。標準設定では，各アクションに対して自動トランザクションが有効になっています。

➤ 各アクションに対して自動トランザクションを無効にするには，［**各アクションをトランザクションとして定義する**］チェック・ボックスをクリアします（標準設定では有効）。

➤ 各ステップに対して自動トランザクションを有効にするには，［**各ステップをトランザクションとして定義する**］チェック・ボックスを選択します（標準設定では無効）。

自動トランザクションを無効にしている場合でも，記録中および記録後に手作業でトランザクションを挿入できます。手作業によるトランザクションの挿入の詳細については，第 7 章「仮想ユーザ・スクリプトの拡張」を参照してください。

注：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。

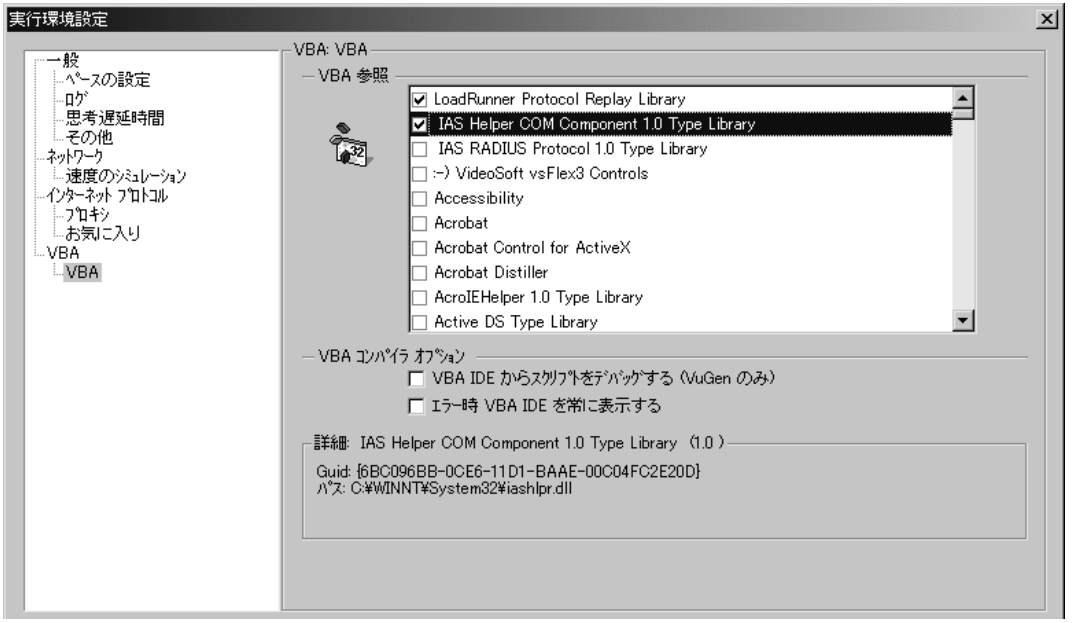
VB 実行環境の設定

Visual Basic によるスクリプトを実行する前に，再生時に参照するライブラリを指定します。VuGen は，マシンに格納されているすべてのライブラリの一覧を表示します。

実行環境設定の表示と設定を行うには，［実行環境設定］ダイアログ・ボックスを使用します。［実行環境設定］ダイアログ・ボックスを表示するには，VuGen ツールバーの［**実行環境の設定**］ボタンをクリックします。

VBA の実行環境を設定するには，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，［**VBA : VBA**］ノードを選択します。



2 ［**VBA 参照**］セクションで，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。

3 コンパイラに関する適切なオプションを選択します。

Visual Basic IDE（統合開発環境）を使ってデバッグを行えるようにするには，［**VBA IDE からスクリプトをデバッグする（VuGen のみ）**］を選択します。

スクリプト実行中に Visual Basic IDE を常に表示しておくには，［**エラー時 VBA IDE を常に表示する**］を選択します。

4 ［**OK**］をクリックして実行環境の設定を適用します。

第 13 章

インターネット実行環境の設定

ネットワークの速度をシミュレートするには，ネットワーク実行環境を設定します。

本章では，以下の項目について説明します。

- ネットワーク実行環境の設定について
- ネットワーク速度の設定

以降の情報は，すべてのインターネット・プロトコル仮想ユーザ・タイプ，Citrix ICA，Oracle NCA，WinSock を対象とします。

すべての仮想ユーザに適用される一般的な実行環境の設定については，第 12 章「実行環境の設定」を参照してください。

ネットワーク実行環境の設定について

インターネット・プロトコル仮想ユーザ・スクリプトを作成した後，実行環境の設定を行います。この設定によって，仮想ユーザが正しく実際のユーザをエミュレートできるようインターネット環境を構成できます。インターネットの実行環境設定では，プロキシ，ブラウザ，速度のシミュレーション，その他の設定が行えます。

インターネット関連の実行環境の設定は，［実行環境設定］ダイアログ・ボックスで行います。必要な設定を行うためのノードをクリックします。

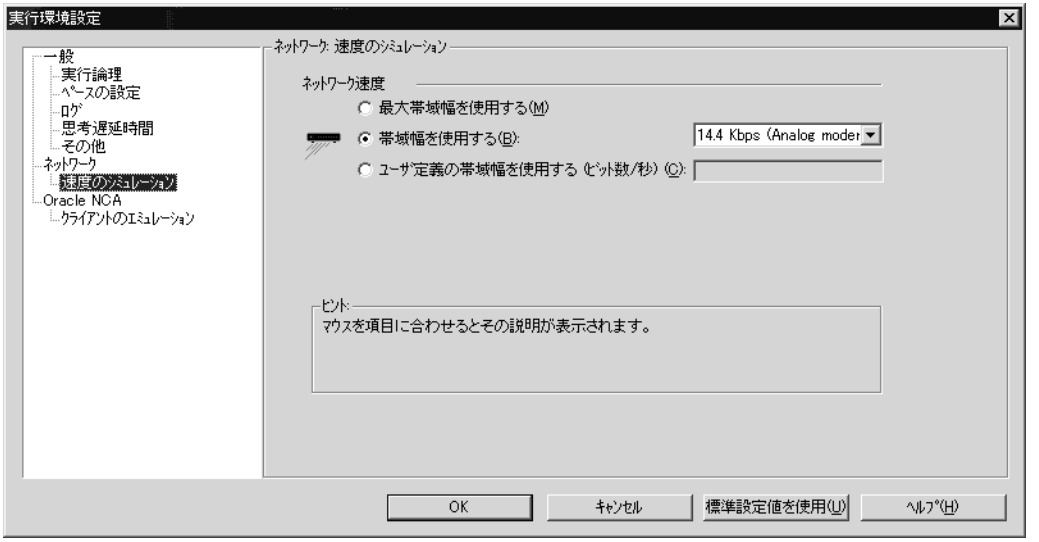
［実行環境設定］ダイアログ・ボックスを表示するには，次のいずれかの手順を実行します。

- VuGen ツールバーの［**実行環境の設定**］ボタンをクリックします。
- キーボードのショートカット・キー，**F4** キーを押します。
- ［**仮想ユーザ**］＞［**実行環境の設定**］を選択します。

LoadRunner コントローラまたは Mercury Tuning Module からでも実行環境の設定を変更できます。詳細については，お使いの製品のマニュアルを参照してください。

ネットワーク速度の設定

［実行環境設定］ダイアログ・ボックスの［**ネットワーク：速度のシミュレーション**］ノードで，テスト環境におけるモデムのエミュレーションを設定します。



速度のシミュレーション

速度のシミュレーション設定を使って，テスト環境をエミュレートするのに最も近い帯域幅を選択できます。次のオプションが使用できます。

［**最大帯域幅を使用する**］：標準では，帯域幅のエミュレーションは無効になっており，仮想ユーザはネットワーク上で利用できる最大の帯域幅で実行されます。

［**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）。

第 14 章

スタンドアロン・モードでの仮想ユーザ・スクリプトの実行

仮想ユーザ・スクリプトを作成し，その実行環境を設定したら，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。

本章では，次の項目について説明します。

- スタンドアロン・モードでの仮想ユーザ・スクリプトの実行について
- VuGen での仮想ユーザ・スクリプトの実行
- 再生ログ
- VuGen のデバッグ機能の使用
- Web 仮想ユーザ・スクリプトでの VuGen のデバッグ機能の使用
- VuGen ウィンドウを使った作業
- コマンド・プロンプトからの仮想ユーザ・スクリプトの実行
- UNIX コマンド・ラインからの仮想ユーザ・スクリプトの実行
- スクリプトのテストへの統合

以降の情報は，GUI を除く全タイプの仮想ユーザ・スクリプトを対象とします。

スタンドアロン・モードでの仮想ユーザ・スクリプトの実行について

スクリプトを作成したら，スクリプトを VuGen インタフェースから直接，スタンドアロン・モードで実行し，その機能を確かめます。UNIX ベースのスクリプトについては，UNIX のコマンド・ラインから実行します。GUI の仮想ユーザをスタンドアロン・モードで実行するには，WinRunner を使用します。

スタンドアロンで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，Tuning Module セッション，またはビジネス・プロセス・モニタ・プロファイル）に統合します。詳細については，『**Mercury LoadRunner コントローラ・ユーザーズ・ガイド**』，**Tuning Console**，**Performance Center**，または **Application Management** のマニュアルを参照してください。

スクリプトをスタンドアロン・モードで実行する前に，次の作業を実施できます。

- 仮想ユーザ関数を使ったスクリプトの拡張（第 7 章「仮想ユーザ・スクリプトの拡張」参照）
- スクリプトのパラメータ化（第 8 章「VuGen パラメータを使った作業」参照）
- スクリプトのクエリの相関（第 11 章「ステートメントの相関」参照）

上記のステップは任意であり，すべてのスクリプトに適用できるわけではありません。

VuGen での仮想ユーザ・スクリプトの実行

仮想ユーザ・スクリプトを作成したら，VuGen を使ってスクリプトを実行し，正しく実行できることを確認します。再生のためのオプションをいくつか設定できます。

注： VuGen で仮想ユーザ・スクリプトを実行できるのは Windows プラットフォームだけです。UNIX ベースの仮想ユーザ・スクリプトを実行するには，224 ページ「UNIX コマンド・ラインからの仮想ユーザ・スクリプトの実行」を参照してください。

再生オプションの設定

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。

［**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** および **end** セクションの内容を除く）を表示実行モードで実行します。

［**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。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。

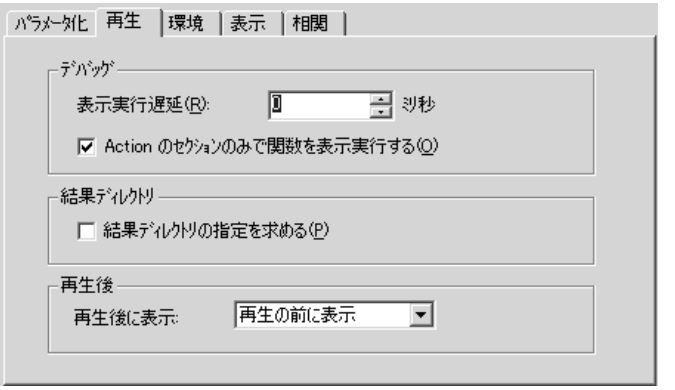
［**再生後に表示**］：再生後の VuGen の動作を指示します。

- ［**再生の前に表示**］：再生前のビューに戻します。
- ［**再生のサマリ**］：ワークフロー・ウィザードの［再生サマリ］ウィンドウに直接移動します。
- ［**テスト結果の視覚化**］：［テスト結果］を開きます（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。）。

表示実行モードを有効にして，そのプロパティを設定するには，次の手順を実行します。

1 ［**表示**］＞［**表示実行**］を選択して，表示実行モードで実行します。VuGen によって，［**表示実行**］メニュー・オプションの横にチェック・マークが付けられ，表示実行モードが有効になります。

2 表示実行に対して遅延を設定するには，［**ツール**］＞［**一般オプション**］を選択します。［一般オプション］ダイアログ・ボックスが開きます。



3 ［**再生**］タブを選択します。

4 ［**表示実行遅延**］ボックスに，コマンド間の遅延時間をミリ秒単位で指定し，［**OK**］をクリックします。

5 Actions セクションの内容のみを表示実行するには，［**Actions のセクションのみで関数を表示実行する**］を選択します。

6 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できるようにするには，［**結果ディレクトリの指定を求める**］を選択します。実行コマンドをクリックすると，［**結果ディレクトリの指定**］ダイアログ・ボックスが開きます。

7 実行結果を格納するディレクトリの名前を入力するか，標準の名前をそのまま使用して［**OK**］をクリックします。

表示オプションの設定

Web 仮想ユーザ・スクリプトを実行しているときは，［表示］オプション（［**ツール**］＞［**一般オプション**］）を設定できます。［表示］オプションによって，VuGen で実行時ビューアを表示するか，スクリプト実行中に VuGen でレポートを生成するかなどを指定します。

詳細については，221 ページ「Web 仮想ユーザ・スクリプトでの VuGen のデバッグ機能の使用」を参照してください。

仮想ユーザ・スクリプトの再生

スクリプトをテスト環境や実運用環境に統合する前に，スクリプトを VuGen から実行して正しく機能することを確かめます。VuGen は，再生を監視し，既存の問題または問題の可能性を特定するためのツールをいくつか備えています。次のツールがあります。

- 再生ログ
- ［実行時データ］タブ
- ステップ実行コマンド
- ブレークポイント
- ブックマーク
- ［移動］コマンド

VuGen でスクリプトを再生するには，次の手順を実行します。

1 ［**仮想ユーザ**］＞［**実行**］を選択します。

VuGen のメイン・ウィンドウの下部に出力ウィンドウが開いた後，あるいは，すでに開いている場合には内容がクリアされた後に，仮想ユーザ・スクリプトの実行が開始されます。ツリー・ビューでは，仮想ユーザ・スクリプトはスクリプト内の 1 番目のアイコンから実行されます。スクリプト・ビューでは，仮想ユーザ・スクリプトはスクリプトの 1 行目から実行されます。

2 仮想ユーザのすべてのアクションのログのほか，警告やエラーを見るには，［出力］ウィンドウの［**再生ログ**］タブをクリックします。詳細については，214 ページ「再生ログ」を参照してください。

3 実行時データおよびパラメータのサマリをそれらの使用中に表示するには，［出力］ウィンドウの［**実行時データ**］タブをクリックします。詳細については，215 ページ「[実行時データ］タブ」を参照してください。

4 スクリプトの実行中または実行後に，［出力］ウィンドウを非表示にするには，［**表示**］＞［**出力ウィンドウ**］を選択します。［出力］ウィンドウが閉じられ，［**表示**］メニューの［**出力ウィンドウ**］のアイコンがくぼんでいない状態になります。

5 実行中の仮想ユーザ・スクリプトを中断するには，［**仮想ユーザ**］＞［**一時停止**］を選択して，スクリプトの実行を一時停止します。または，［**仮想ユーザ**］＞［**停止**］を選択して，スクリプトの実行を終了します。

再生ログ

［出力］ウィンドウの［再生ログ］には，実行時の仮想ユーザのアクションを表すメッセージが表示されます。この情報は，スクリプトがシナリオ，セッション・ステップ，またはプロファイルの中で実行されるときに，どのように実行されるかを示します。

スクリプトの実行が完了したら，スクリプトをエラーなく実行できたかどうかを確認するために，再生ログのメッセージを調べます。

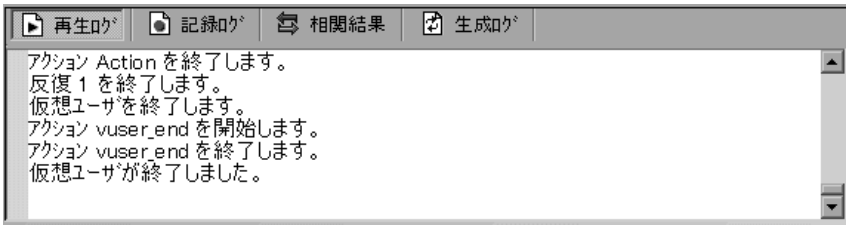
［再生ログ］では，テキストがさまざまに色分けされています。

- **黒**：標準の出力メッセージ
- **赤**：標準のエラー・メッセージ
- **緑**：引用符に囲まれて表示されるリテラル文字列（URL など）
- **青**：トランザクション情報（開始ステータス，終了ステータス，持続時間）
- **橙**：反復の始まりと終わり

アクション名で始まる行をダブルクリックすると，出力ログを生成した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。

［出力］ウィンドウの表示と非表示の詳細については，213 ページ「仮想ユーザ・スクリプトの再生」を参照してください。

次の例は，Web 仮想ユーザ・スクリプトの実行によって生成された再生ログのメッセージを示しています。



[実行時データ] タブ

再生中に最新になったスクリプト情報を，[実行時データ] タブを使用して追跡することができます。

再生中に，右端の［**実行時データ**］タブをクリックします。このタブには展開または折りたたみが可能な 2 つのセクションがあります。

［**一般**］：［一般］セクションには，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**］：［パラメータ］セクションには，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。詳細については，第 8 章「VuGen パラメータを使った作業」を参照してください。

テストの実行後には［実行時データ］タブにはアクセスできません。このタブには，テストの再生中に変化するデータのみが表示されるからです。

VuGen のデバッグ機能の使用

VuGen には，仮想ユーザ・スクリプトのデバッグに役立つ，ステップ実行コマンドとブレークポイントの 2 つのオプションがあります。これらのオプションは，VBScript Vuser および VB Vuser タイプの仮想ユーザでは使用できません。

また VuGen には，Web 仮想ユーザ・スクリプトのデバッグに役立つ追加機能があります。詳細については，221 ページ「Web 仮想ユーザ・スクリプトでの VuGen のデバッグ機能の使用」を参照してください。

デバッグ・ツールバーを表示するには，次の手順を実行します。

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。デバッグ・ツールバーがツールバー領域に表示されます。

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。

ステップ実行コマンドでスクリプトを実行するには，次の手順を実行します。

1 ［**仮想ユーザ**］＞［**ステップごとに実行**］を選択するか，デバッグ・ツールバーの［**ステップ**］ボタン をクリックします。

VuGen によってスクリプトの最初の行が実行されます。

2 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て，スクリプトの実行を続けます。

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。ブックマークを管理するには，217 ページ「ブレークポイント・マネージャ」の手順を実行します。

ブレークポイントを設定するには，次の手順を実行します。

1 スクリプト内で実行を停止する行にカーソルを移動します。

2 ［**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**］ボタン をクリックします。または，キーボードの

F9 キーを押します。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。

3 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。ブレークポイントの記号の中に白い点が表示されます（◎）。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**］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設定したスクリプトを実行するには，次の手順を実行します。

1 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。

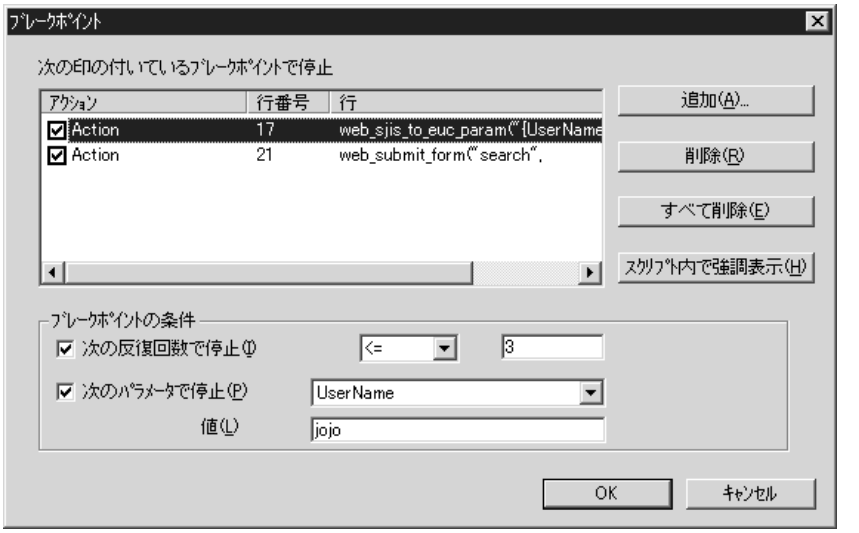
2 実行を再開するには，［**仮想ユーザ**］＞［**実行**］を選択します。

再開すると，次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。

ブレークポイント・マネージャ

ブレークポイント・マネージャを使用してブレークポイントを表示および管理することができます。ブレークポイント・マネージャから，スクリプト内のすべてのブレークポイントを操作できます。

ブレークポイント・マネージャを開くには，［**編集**］＞［**ブレークポイント**］を選択します。



スクリプト内のブレークポイント位置へ移動するには，次の手順を実行します。

1 リストからブレークポイントを選択します。

2 ［**スクリプト内で強調表示**］をクリックします。スクリプト内の行が強調表示されます。

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。

ブレークポイントの管理

ブレークポイント・マネージャから，ブレークポイントの追加，削除，無効化，または条件の設定を実行できます。

ブレークポイントを追加するには，次の手順を実行します。

1 ［**追加**］をクリックします。［ブレークポイントの追加］ダイアログ・ボックスが開きます。

2 ［**アクション**］を選択し，ブレークポイントを追加する［**行番号**］を指定します。

3 ［**OK**］をクリックします。ブレークポイントがブレークポイントのリストに追加されます。

ブレークポイントを削除するには，次の手順を実行します。

1 ブレークポイントを 1 つ削除するには，ブレークポイントを選択して［**削除**］をクリックします。

2 ブレークポイントをすべて削除するには，［**すべて削除**］をクリックします。

ブレークポイントを有効化または無効化するには，次の手順を実行します。

1 ブレークポイントを有効にするには，［アクション］カラムで，アクションのチェック・ボックスを選択します。

2 ブレークポイントを無効にするには，［アクション］カラムで，アクションのチェック・ボックスをクリアします。

ブレークポイント・マネージャから，特定の条件のもとで実行を一時停止するためのブレークポイントを設定できます。

ブレークポイントの条件を設定するには，次の手順を実行します。

1 特定の反復回数後にスクリプトを一時停止するには，［**次の反復回数で停止**］を選択して，必要な回数を入力します。

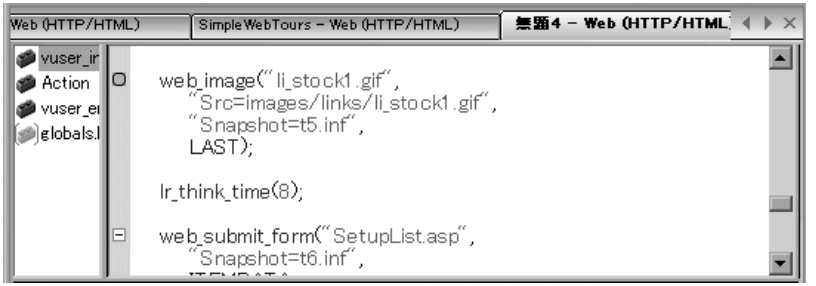
2 パラメータ **X** が特定の値になったときにスクリプトを一時停止するには，［**次のパラメータで停止**］を選択して，必要な値を入力します。パラメータの詳細については，第 8 章「VuGen パラメータを使った作業」を参照してください。

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。

ブックマークを作成するには，次の手順を実行します。

1 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。



2 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。

3 ブックマーク間を移動するには，次の手順を実行します。

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移動するには，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。

注 : 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。

［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。

Web 仮想ユーザ・スクリプトでの VuGen のデバッグ機能の使用

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。

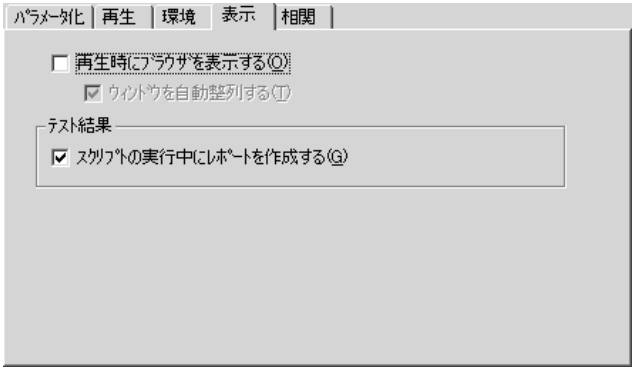
- Web 仮想ユーザ・スクリプトの実行時に実行時ビューアを表示するように VuGen を設定することができます。実行時ビューアは，VuGen 向けに Mercury によって開発されました。仮想ユーザ・スクリプトの記録に使用するブラウザとは無関係です。実行時ビューアには，仮想ユーザによってアクセスされる各 Web ページが表示されます。これにより仮想ユーザが正しい Web ページにアクセスしているかどうかを確認できるので，Web 仮想ユーザ・スクリプトのデバッグ時に役立ちます。『**第 2 巻 - プロトコル**』の「Web 仮想ユーザのヒント (パワー・ユーザ向け)」を参照してください。

- 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。『**第 2 巻 - プロトコル**』の「レポートを使った仮想ユーザ・スクリプトのデバッグ」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。スクリプトをコントローラ，Tuning Module，または Administration Console 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。

Web 仮想ユーザ・スクリプトのデバッグ機能を有効化にするには，次の手順を実行します。

1 VuGen のメニューから［**ツール**］＞［**一般オプション**］を選択します。［一般オプション］ダイアログ・ボックスが開きます。［**表示**］タブを選択します。



2 実行時ビューアを有効にするには，［**再生時にブラウザを表示する**］チェック・ボックスを選択します。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**］チェック・ボックスを選択します。

3 仮想ユーザに結果サマリ・レポートを生成させるには，［**テスト結果**］セクションで［**スクリプトの実行中にレポートを作成する**］チェック・ボックスを選択します。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。

4 設定を承認するには［**OK**］をクリックします。［一般オプション］ダイアログ・ボックスが閉じます。

VuGen ウィンドウを使った作業

次の機能を実行することで，VuGen のウィンドウを表示して配置を変更し，スクリプトの関連データを表示することができます。

➤ ［出力］ウィンドウの表示 / 非表示

［**表示**］＞［**出力ウィンドウ**］を選択して，VuGen スクリプト・エディタの下の出力ウィンドウの表示 / 非表示を切り替えます。出力ウィンドウには，プロトコルに応じていくつかのタブが表示されます。最も一般的なタブは，［再生ログ］，［記録ログ］，［生成ログ］，および［相関結果］です。詳細については，214 ページ「再生ログ」を参照してください。

➤ すべてのサムネイルの表示

［**表示**］＞［**すべてのサムネイルを表示**］を選択して，スクリプトのすべてのステップをサムネイルとして表示します。主要なステップのみについてサムネイルを表示するには，このオプションをクリアします。詳細については，25 ページ「スクリプトのサムネイルの表示」を参照してください。

➤ グリッドの表示

［**表示**］＞［**データ グリッド**］を選択して，データ・グリッドをサポートしているプロトコル（データベース，COM，および Microsoft . NET）について，データのグリッド表示を有効にします。グリッドがスクリプトの中に表示されます。

➤ ウィンドウの操作

開いているスクリプトをすべて閉じるには，［**ウィンドウ**］＞［**すべて閉じる**］を選択します。必要に応じて，保存していないスクリプトを保存するよう VuGen から求められます。

コマンド・プロンプトからの仮想ユーザ・スクリプトの実行

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できます。

DOS 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・ボックスからのスクリプトを実行するには，次の手順を実行します。

1 ［**スタート**］＞［**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。

＜ VuGen のパス＞ /bin/mdrv.exe -usr **＜スクリプト名＞** -vugen_win 0

＜スクリプト名＞は **.usr** スクリプト・ファイルのフル・パスです。例えば，**c:¥temp¥mytest¥mytest.usr** などとなります。

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。

UNIX コマンド・ラインからの仮想ユーザ・スクリプトの実行

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。スクリプトを正しく実行できることを次の手順で確認します。

1 **記録されたスクリプトを VuGen からテストします。**

記録されたスクリプトを VuGen から実行して，Windows ベースのシステムで正しく実行できることを確認します。

2 **仮想ユーザ・スクリプトのファイルを UNIX ドライブにコピーします。**

ファイルをローカルの UNIX ドライブに転送します。

3 **verify_generator を使用して，UNIX マシン上での仮想ユーザの設定を確かめます。**

詳細については，225 ページ「verify_generator によるテスト」を参照してください。

4 **UNIX コマンド・ラインからスクリプトをテストします。**

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 ＜スクリプト名＞ .usr
```

verify_generator によるテスト

この検証ユーティリティは，ローカル・ホストで通信パラメータを調べ，すべての仮想ユーザ・タイプとの互換性を確認します。

仮想ユーザ環境で次の項目を調べます。

- 少なくとも 128 のファイル記述子があること
- .rhost のパーミッションが -rw-r--r-- という正しいものになっていること
- rsh を使用してホストに接続できること。接続できない場合は，.rhosts 内のホスト名を調べる
- 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
- .cshrc がホーム・ディレクトリに存在すること
- 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パーミッションを持っていること
- 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，実行されていること

1 つのホストですべての仮想ユーザを実行する場合は，次を入力します。

```
verify_generator
```

verify_generator は，設定が正しい場合は「OK」を返します。設定が正しくない場合は「Failed」を返し，設定の修正方法を提示します。

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]
```

コマンド・ライン・オプション：run_db_vuser シェル・スクリプト

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。

--help：使用可能なオプションを表示します（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（プリプロセッシング）を実行します。

-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 のファイルを作成します。cpp を正しく実行できた後にのみ，cci を実行できます。

-driver ＜ドライバのパス＞：指定されたドライバ・プログラムを使用します。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。例えば，/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 ファイルを実行します。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＞：指定された .ci ファイルを実行します。

-out ＜出力パス＞：指定されたディレクトリに結果を格納します。

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。run_db_vuser.sh は＜ **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て，結果を仮想ユーザ・スクリプト・ディレクトリ内の出力ファイルに保存します。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。

例えば，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
```

スクリプトのテストへの統合

スタンドアロン・モードでのスクリプトの実行が成功し，スクリプトが正常に機能することを確認したら，そのスクリプトを自分の環境（LoadRunner シナリオ，Performance Center 負荷テスト，Tuning Module セッション，またはビジネス・プロセス・モニタ・プロファイル）に統合します。

テストを統合するときは，次の情報を指定します。

- 統合するスクリプト
- スクリプトを実行する仮想ユーザ
- スクリプトが実行されるロード・ジェネレータ
- スケジュール設定

詳細については，『**Mercury LoadRunner コントローラ・ユーザーズ・ガイド**』，**Tuning Console**，**Performance Center**，または **Application Management** のマニュアルを参照してください。

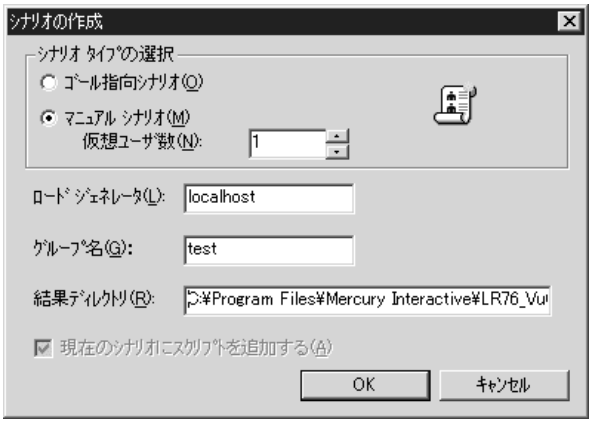
VuGen を使用しての LoadRunner シナリオの作成

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，**Application Management** のマニュアルを参照してください。

通常は，LoadRunner コントローラ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。

VuGen でシナリオを作成するには，次の手順を実行します。

1 ［**ツール**］＞［**コントローラのシナリオを作成**］を選択します。［シナリオの作成］ダイアログ・ボックスが開きます。



2 ゴール指向シナリオか，マニュアル・シナリオかを選択します。

ゴール指向シナリオを選択した場合，指定した目的に応じたシナリオが LoadRunner によって自動的に作成されます。それに対しマニュアル・シナリオでは実行する仮想ユーザ数を指定します。

3 マニュアル・シナリオの場合は，スクリプトを実行する仮想ユーザの数を入力します。

4 仮想ユーザの実行に使用するマシンの名前を［**ロード ジェネレータ**］ボックスに入力します。

5 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。それらの仮想ユーザのグループ名を［**グループ名**］ボックスで指定します。

6 ゴール指向シナリオの場合，［**スクリプト名**］を指定します。

7 結果を格納する場所を，［**結果ディレクトリ**］ボックスに入力します。

8 コントローラ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，［**現在のシナリオにスクリプトを追加する**］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。

9［**OK**］をクリックします。VuGen によって，仮想ユーザのビューにコントローラが表示されます。

10 共有ネットワーク・ドライブにスクリプトを保存するようにコントローラを設定した場合は，パス変換を行わなければならないことがあります。

詳細については，『**Mercury LoadRunner コントローラ・ユーザーズ・ガイド**』を参照してください。

第 15 章

Quality Center を使ったスクリプト管理

VuGen と Quality Center の統合により，Quality Center を使って仮想ユーザ・スクリプトを管理できます。

本章では，次の項目について説明します。

- Quality Center を使ったスクリプト管理について
- Quality Center の接続と切断
- Quality Center プロジェクトからスクリプトを開く
- Quality Center プロジェクトへのスクリプトの保存
- VuGen でのスクリプトのバージョン管理

Quality Center を使ったスクリプト管理について

VuGen は，Mercury の Web ベースのテスト管理ツール，Quality Center と連携して機能します。Quality Center は，仮想ユーザ・スクリプト，シナリオ，またはセッション・ステップの格納と検索，および結果の収集のための能率的な手段を提供します。スクリプトを Quality Center プロジェクトに格納し，固有のグループに編成します。

VuGen で Quality Center プロジェクトにアクセスするには，VuGen を Quality Center 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。

Quality Center を使った作業の詳細については，『**Quality Center ユーザーズ・ガイド**』を参照してください。

Quality Center の接続と切断

VuGen と Quality Center の両方を使って作業している場合には，VuGen から Quality Center プロジェクトとやり取りできます。テスト・プロセスにおいて，VuGen と Quality Center プロジェクトはいつでも接続または切断できます。

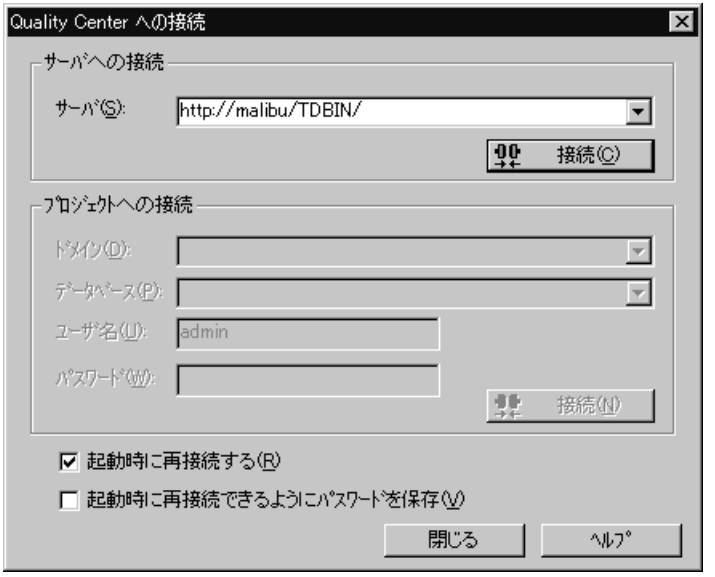
Quality Center への VuGen の接続

接続プロセスには 2 つの段階があります。最初に，VuGen をローカル Quality Center Web サーバまたはリモート Quality Center Web サーバに接続します。このサーバによって，VuGen と Quality Center プロジェクトの間の接続が処理されます。

次に，VuGen で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，テスト対象または監視対象アプリケーションのためのスクリプトが格納されています。Quality Center 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。

VuGen を Quality Center に接続するには，次の手順を実行します。

1 VuGen で，［**ツール**］＞［**Quality Center への接続**］を選択します。［Quality Center への接続］ダイアログ・ボックスが表示されます。



2 ［サーバ］ボックスに，Quality Center がインストールされている Web サーバの URL アドレスを入力します。

注：ローカル・エリア・ネットワーク（LAN）または広域エリア・ネットワーク（WAN）を介してアクセス可能な Web サーバを選択できます。

---

3 ［**接続**］をクリックします。サーバへの接続が確立されると，［サーバ］ボックスにサーバの名前が読み取り専用形式で表示されます。

4 ［プロジェクトへの接続］セクションの［ドメイン］ボックスで，ドメインを選択します。

5 ［プロジェクトへの接続］セクションの［プロジェクト］ボックスで，Quality Center プロジェクトを選択します。

6 ユーザ名を［**ユーザ名**］ボックスに入力します。

7 パスワードを［**パスワード**］ボックスに入力します。

8 ［**接続**］ボタンをクリックして，選択したプロジェクトに VuGen を接続します。

選択したプロジェクトへの接続が確立されると，［データベース］ボックスにデータベースの名前が読み取り専用形式で表示されます。

9 起動時に Quality Center サーバと選択したプロジェクトに自動的に再接続するには，［**起動時に再接続する**］チェック・ボックスを選択します。

10 ［**起動時に再接続する**］を選択すると，指定したパスワードを保存して起動時に再接続できます。［**起動時に再接続できるようにパスワードを保存**］チェック・ボックスを選択します。

パスワードを保存しなければ，起動時に VuGen を Quality Center に接続するときに，パスワードの入力が必要になります。

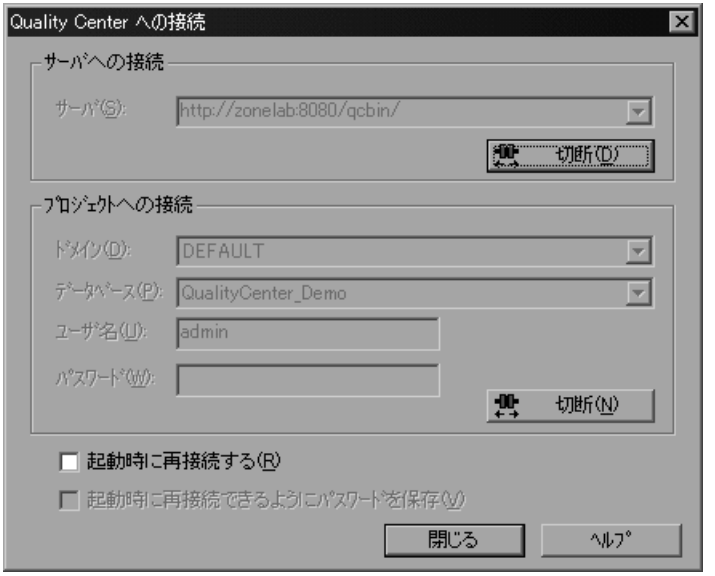
11 ［**閉じる**］をクリックして，［Quality Center への接続］ダイアログ・ボックスを閉じます。

Quality Center からの VuGen の切断

選択した Quality Center プロジェクトと Web サーバから VuGen を切断できます。

Quality Center から VuGen を切断するには，次の手順を実行します。

1 VuGen で，［**ツール**］＞［**Quality Center への接続**］を選択します。［Quality Center への接続］ダイアログ・ボックスが表示されます。



2 選択したプロジェクトから VuGen を切断するには，［プロジェクトへの接続］セクションの［**切断**］ボタンをクリックします。

3 選択したサーバから VuGen を切断するには，［サーバへの接続］セクションの［**切断**］ボタンをクリックします。

4 ［**閉じる**］をクリックして，［Quality Center への接続］ダイアログ・ボックスを閉じます。

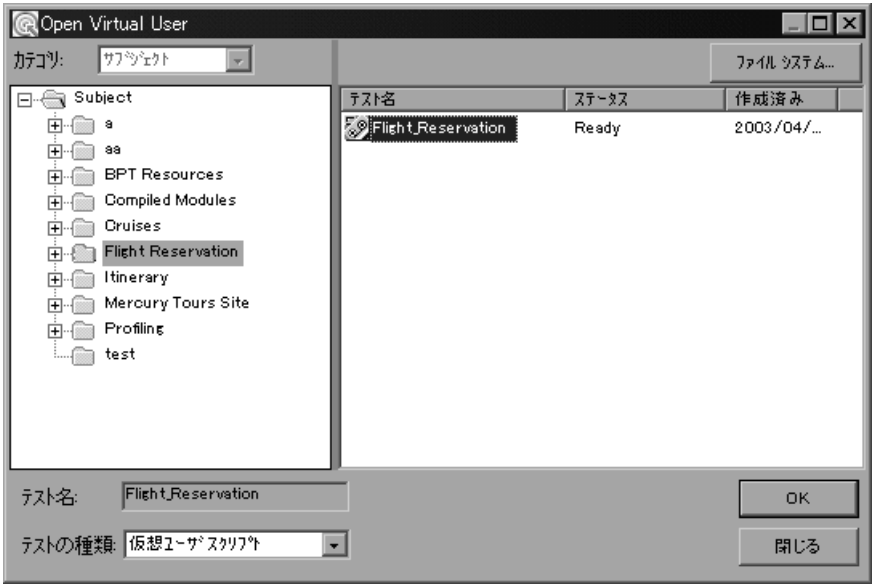
Quality Center プロジェクトからスクリプトを開く

VuGen を Quality Center プロジェクトに接続する場合，Quality Center からシナリオを開くことができます。テストは，ファイル・システムの実際の位置からではなく，テスト計画ツリーでの位置から見つけます。

Quality Center プロジェクトからスクリプトを開くには，次の手順を実行します。

1 Quality Center サーバに接続します（232 ページ「Quality Center への VuGen の接続」を参照してください）。

2 VuGen で，［**ファイル**］＞［**開く**］を選択するか，［**開く**］ボタンをクリックします。［Open Virtual User］ダイアログ・ボックスが表示され，テスト計画ツリーが表示されます。



ファイル・システムから直接スクリプトを開くには，［**ファイル システム**］ボタンをクリックします。［テストを開く］ダイアログ・ボックスが開きます（［テストを開く］ダイアログ・ボックスからは，［**Quality Center**］ボタンをクリックすれば［Quality Center プロジェクトからテストを開く］ダイアログ・ボックスに戻ることができます）。

3 テスト計画ツリー内の適切なサブジェクトをクリックします。ツリーを展開して下位レベルを表示するには，閉じているフォルダをダブルクリックします。ツリーを折りたたむには，開いているフォルダをダブルクリックします。

サブジェクトを選択すると，そのサブジェクトに属しているスクリプトが［テスト名］カラムに表示されます。

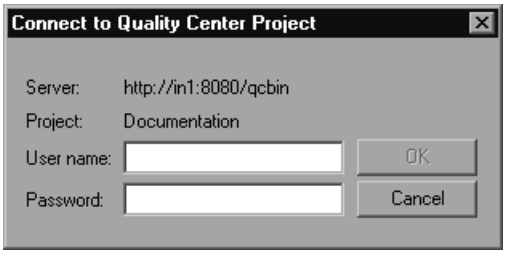
4 ［テスト名］カラムからスクリプトを選択します。スクリプト名が読み取り専用の［テスト名］ボックスに表示されます。

5 ［**OK**］をクリックしてスクリプトを開きます。VuGen によってスクリプトがロードされます。スクリプトの名前が VuGen のタイトルバーに表示されます。［**設計**］タブに，テスト計画ツリーのすべてのスクリプトが表示されます。

注 : ［ファイル］メニューの最新ファイル・リストからスクリプトを開くこともできます。Quality Center プロジェクト内のスクリプトを選択したときに，VuGen がそのプロジェクトに接続されていない場合は，［Quality Center への接続］ダイアログ・ボックスが表示されます。ユーザ名とパスワードを入力してプロジェクトにログインし，［**OK**］をクリックします。

最近使用したテストのリストからテストを開く

［ファイル］メニューの最新ファイル・リストから仮想ユーザ・スクリプトを開くこともできます。Quality Center プロジェクトのスクリプトを選択したときに，現在 VuGen が Quality Center に接続されていないか，スクリプトの正しいプロジェクトに接続されていないと，［Quality Center プロジェクトへの接続］ダイアログ・ボックスが表示され，正しいサーバ，プロジェクト，および前回当該コンピュータでスクリプトを開いたユーザの名前が表示されます。



プロジェクトにログインし，［**OK**］をクリックします。

また，前回異なる Quality Center ユーザ名を使用して編集したテストを開くことを選択した場合も，［Quality Center プロジェクトへの接続］ダイアログ・ボックスが開きます。表示されているユーザ名を使ってログインすることも，［**キャンセル**］をクリックして現在のユーザ名でログインすることもできます。

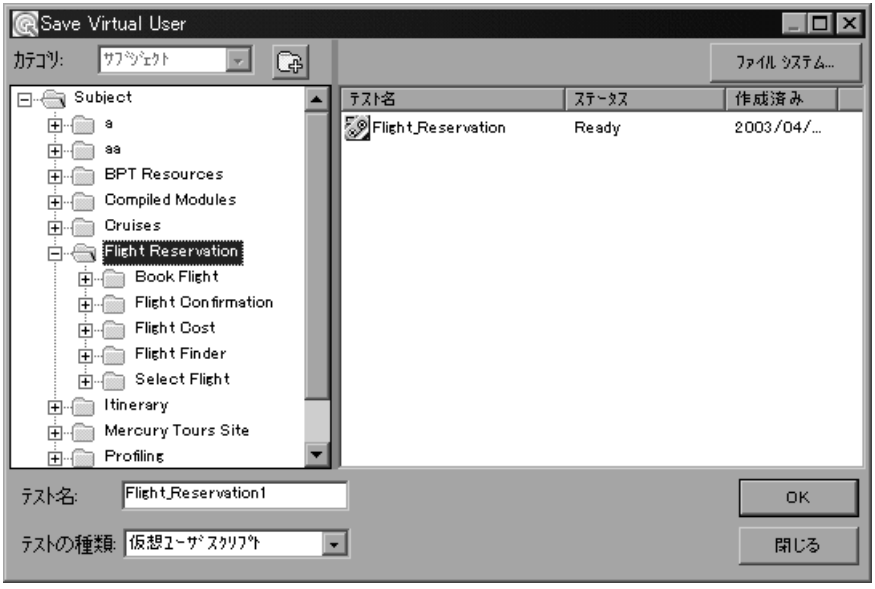
Quality Center プロジェクトへのスクリプトの保存

VuGen が Quality Center プロジェクトに接続されているときには，VuGen で新しいスクリプトを作成して，プロジェクトに直接保存することができます。スクリプトを保存するには，分かりやすい名前を付けて，テスト計画ツリーの中の適切なサブジェクトに関連付けます。これにより，各サブジェクトに対して作成されたスクリプトを把握し，テスト計画とテスト作成の進捗を表示できるようになります。

スクリプトを Quality Center プロジェクトに保存するには，次の手順を実行します。

1 Quality Center サーバに接続します（232 ページ「Quality Center への VuGen の接続」を参照してください）。

2 VuGen で，［**ファイル**］＞［**名前を付けて保存**］を選択します。［Save Virtual User］ダイアログ・ボックスが表示され，テスト計画ツリーが表示されます。



ファイル・システムに直接スクリプトを保存するには，［**ファイル システム**］ボタンをクリックします。［テストを保存］ダイアログ・ボックスが表示されます （［テストを保存く］ダイアログ・ボックスからは，［**Quality Center**］ボタンをクリックすれば［Quality Center プロジェクトにテストを保存］ダイアログ・ボックスに戻ることができます)。

3 テスト計画ツリーの関連するサブジェクトを選択します。ツリーを展開して下位レベルを表示するには，閉じているフォルダをダブルクリックします。サブレベルの表示を折りたたむには，開いたフォルダをダブルクリックします。

4 ［テスト名］ボックスに，スクリプトの名前を入力します。スクリプトを簡単に識別できるような分かりやすい名前を使用します。

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

次に Quality Center を起動したときに，Quality Center のテスト計画ツリーに新しいスクリプトが表示されます。

VuGen でのスクリプトのバージョン管理

VuGen をバージョン・コントロール・サポート付きの Quality Center プロジェクトに接続すると，古いバージョンを維持しながら，自動化されたスクリプトの更新や変更ができます。これにより，各スクリプトに加えた変更の追跡，スクリプトのバージョン間の変更の確認，スクリプトの旧バージョンの復元が容易になります。

バージョン・コントロール・サポート付きのプロジェクトにスクリプトを保存すると，バージョン・コントロール・データベースにスクリプトが追加されます。バージョン・コントロール・データベースにスクリプトをチェック・イン，チェック・アウトすることで，スクリプトのバージョンが管理されます。

最新バージョンのスクリプトは，Quality Center のリポジトリにあり，Quality Center によってすべてのテストの実行に使用されます。

注 :［**ファイル**］メニューの［**Quality Center バージョン管理**］オプションは，バージョン・コントロール・サポートの付いた Quality Center プロジェクトのデータベースに接続している場合にだけ使用できます。

バージョン・コントロール・データベースへのスクリプトの追加

［**名前を付けて保存**］を使ってバージョン・コントロール・サポート付きの Quality Center プロジェクトに新規のスクリプトを保存すると，QuickTest によって自動的にスクリプトがプロジェクトに保存され，バージョン・コントロール・データベースにバージョン番号 1.1.1 でチェック・インされ，その後作業を続けられるようにチェック・アウトされます。

QuickTest ステータス・バーにはそのつど各操作が表示されます。ただし，既存のスクリプトに変更を保存しても，そのスクリプトをチェック・インすることにはなりません。スクリプトを保存して閉じても，チェック・インを選択するまではスクリプトはチェック・アウトしたままです。詳細については，239 ページ「バージョン・コントロール・データベースからのスクリプトのチェック・アウト」を参照してください。

バージョン・コントロール・データベースからのスクリプトのチェック・アウト

［**ファイル**］＞［**開く**］を選択して，現在バージョン・コントロール・データベースにチェック・インしているスクリプトを開くと，スクリプトは読み取り専用モードで開きます。

注：［Quality Center からテストを開く］ダイアログ・ボックスには，プロジェクトの各スクリプトのバージョン・コントロール・ステータスを示すアイコンが表示されます。詳細については，235 ページ「Quality Center プロジェクトからスクリプトを開く」を参照してください。

チェック・インしたスクリプトを確認できます。また，スクリプトを実行して結果を表示することもできます。

スクリプトを変更するには，スクリプトをチェック・アウトする必要があります。スクリプトをチェック・アウトすると，スクリプトは Quality Center によってユーザの一意のチェック・アウト・ディレクトリ（初めてスクリプトをチェック・アウトするときに自動的に作成される）にコピーされ，プロジェクト・データベースにロックされます。これにより，スクリプトに加えた変更を Quality Center プロジェクトのほかのユーザに上書きされないようにします。ただし，この場合でも，ほかのユーザはデータベースにチェック・インされているスクリプトの最新バージョンを実行できます。

スクリプトを保存して閉じることはできますが，Quality Center データベースに戻されるまでスクリプトはロックされています。スクリプトは，スクリプトをチェック・インするか，チェック・アウトの操作を取り消すことによって解放されます。スクリプトのチェック・インの詳細については，241 ページ「バージョン・コントロール・データベースへのスクリプトのチェック・イン」を参照してください。チェック・アウトの取り消しの詳細については，246 ページ「チェック・アウト操作の取り消し」を参照してください。

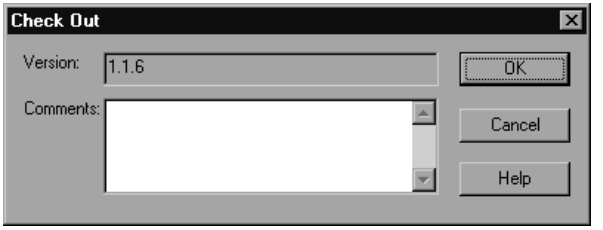
標準では，チェック・アウト・オプションはスクリプトの最新バージョンにアクセスします。古いバージョンのスクリプトもチェック・アウトできます。詳細については，243 ページ「[バージョンの履歴］ダイアログ・ボックスの使用方法」を参照してください。

最新バージョンのスクリプトをチェック・アウトするには，次の手順を実行します。

1 チェック・アウトするスクリプトを開きます。詳細については，235 ページ「Quality Center プロジェクトからスクリプトを開く」を参照してください。

注：開いたスクリプトが現在チェック・インされていることを確認します。チェック・アウトしているスクリプトを開くと，［**チェック アウト**］オプションは無効になります。ほかのユーザがチェック・アウトしているスクリプトを開くと，［**バージョンの履歴**］オプションを除くすべての［**Quality Center バージョン管理**］オプションは無効になります。

2［**ファイル**］＞［**Quality Center バージョン管理**］を選択します。［チェック アウト］ダイアログ・ボックスが開き，チェック・アウトされるスクリプトのバージョンが表示されます。



3［**コメント**］ボックスに変更の詳細を入力します。

4 [**OK**] をクリックします。読み取り専用のスクリプトが閉じて，書き込み可能スクリプトとして再び自動的に開きます。

バージョン・コントロール・データベースへのスクリプトのチェック・イン

あるスクリプトがチェック・アウトされていても，Quality Center ユーザは以前にチェック・インされたバージョンを実行できます。例えば，スクリプトのバージョン 1.2.3 をチェック・アウトし，いくつか変更を加えてスクリプトを保存するとします。バージョン 1.2.4（または別の番号を割り当てる）としてスクリプトをバージョン・コントロール・データベースに再びチェック・インするまで，Quality Center ユーザはバージョン 1.2.3 の実行を続行できます。

スクリプトの変更が終了し，Quality Center ユーザが新しいバージョンを使用できるようになったら，バージョン・コントロール・データベースにスクリプトをチェック・インします。

注：Quality Center データベースにチェック・インしない場合は，チェック・アウト操作を取り消すことができます。詳細については，246 ページ「チェック・アウト操作の取り消し」を参照してください。

スクリプトをバージョン・コントロール・データベースに再びチェック・インすると，Quality Center はチェック・アウト・ディレクトリからスクリプトのコピーを削除し，Quality Center プロジェクトのほかのユーザがそのスクリプトのバージョンを利用できるようにデータベースのロックを解除します。

現在開いているスクリプトをチェック・インするには，次の手順を実行します。

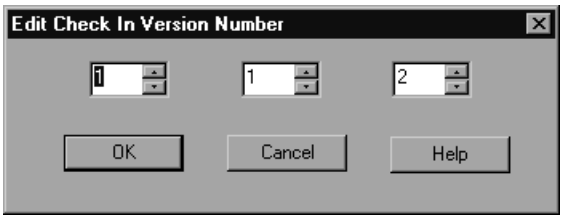
1 現在開いているスクリプトをチェック・アウトしていることを確認します。詳細については，243 ページ「スクリプトのバージョン情報の表示」を参照してください。

注：開いているスクリプトが現在チェック・インされていると，[**チェック イン**] オプションは無効になります。ほかのユーザがチェック・アウトしているスクリプトを開くと，[**バージョンの履歴**] オプションを除くすべての [**Quality Center バージョン管理**] オプションは無効になります。

---

2 ［**ファイル**］＞［**Quality Center バージョン管理**］を選択します。［チェック イン］ダイアログ・ボックスが表示されます。

3 標準設定の新しいバージョン番号を受け入れて手順 7 に進むか，参照ボタンをクリックしてユーザ定義のバージョン番号を指定します。参照ボタンをクリックすると，［チェック イン］ダイアログ・ボックスが開きます。



4 バージョン番号を，手作業またはバージョン番号の各要素の横にある上向き矢印と下向き矢印を使って修正します。最初の要素に入力できる数字は 1 ～ 900 です。第 2，第 3 の要素に入力できる数字は 1 ～ 999 です。バージョン・コントロール・データベースにある対象のスクリプトの最新バージョン番号よりも低いバージョン番号は入力できません。

5 ［**OK**］をクリックしてバージョン番号を保存し，［チェック イン］ダイアログ・ボックスを閉じます。

6 スクリプトのチェック・アウト時に変更の詳細を入力した場合，その詳細は［**コメント**］ボックスに表示されます。コメントの入力や修正はこのボックスで行えます。

7 ［**OK**］をクリックし，スクリプトをチェック・インします。スクリプトが閉じて，読み取り専用のスクリプトとして再び自動的に開きます。

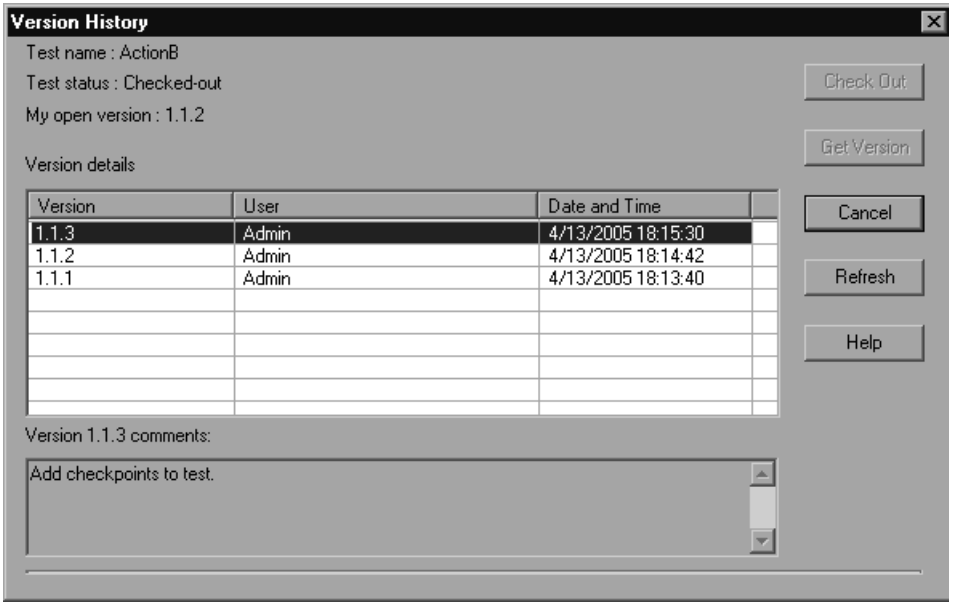
［バージョンの履歴］ダイアログ・ボックスの使用方法

［バージョンの履歴］ダイアログ・ボックスを使って，現在開いているスクリプトに関するバージョン情報の表示や，古いバージョンのスクリプトの表示および取得が行えます。

スクリプトのバージョン情報の表示

Quality Center バージョン・コントロール・データベースに格納されている任意の開いているスクリプトのバージョン情報を，現在のステータスに関係なく表示できます。

スクリプトの［バージョンの履歴］ダイアログ・ボックスを開くには，スクリプトを開き［**ファイル**］＞［**Quality Center バージョン管理**］＞［**バージョンの履歴**］を選択します。



［バージョンの履歴］ダイアログ・ボックスには次の情報が表示されます。

［**Test name**］：現在開いているスクリプトの名前。

［**Test status**］：スクリプトのステータス。

- ［**Checked-in**］：スクリプトは，現在バージョン・コントロール・データベースにチェック・インしています。現在読み取り専用形式で開いています。スクリプトをチェック・アウトして編集できます。
- ［**Checked-out**］：スクリプトをチェック・アウトしています。現在読み取り / 書き込み可能な形式で開いています。
- ［**Checked-out by ＜ユーザ名＞**］：スクリプトは現在別のユーザによってチェック・アウトされています。現在読み取り専用形式で開いています。特定のユーザがスクリプトをチェック • インするまでこのスクリプトをチェック • アウトしたり編集したりすることはできません。

［**My open version**］：現在，お使いの QuickTest コンピュータで開いているスクリプトのバージョン。

［**Version details**］：スクリプトのバージョンの詳細。

- ［**バージョン**］：スクリプトのバージョンの一覧。
- ［**ユーザ**］：各バージョンのチェック・インをしたユーザです。
- ［**日付**］：各バージョンがチェック・インされた日です。［**時刻**］：各バージョンがチェック・インされた時刻です。

［**Version comments**］：選択したスクリプトのバージョンがチェック・インされたときに入力されたコメント。

以前のスクリプトのバージョンを使った作業

古いバージョンのスクリプトは，読み取り専用モードで表示するか，チェック・アウトして最新バージョンとしてチェック・インできます。

古いバージョンのスクリプトを表示するには，次の手順を実行します。

1 Quality Center スクリプトを開きます。最新バージョンのスクリプトが開きます。詳細については，235 ページ「Quality Center プロジェクトからスクリプトを開く」を参照してください。

2 ［**ファイル**］＞［**Quality Center バージョン管理**］＞［**バージョンの履歴**］を選択します。［バージョンの履歴］ダイアログ・ボックスが表示されます。

3 **バージョン詳細リスト**から表示するバージョンを選択します。

4 ［**バージョンの取得**］ボタンをクリックします。QuickTest から，スクリプトがチェック・アウトされていないため読み取り専用モードで開くというメッセージが表示されます。

5［**OK**］をクリックし，QuickTest メッセージを閉じます。選択したバージョンが読み取り専用モードで開きます。

ヒント：QuickTest で現在開いているバージョン番号を確認するには，［バージョンの履歴］ダイアログ・ボックスの［**My open version**］の値を参照します。

［**バージョンの取得**］オプションを使用して読み取り専用モードで古いバージョンを開いたら，［**ファイル**］＞［**Quality Center バージョン管理**］＞［**チェック アウト**］を選択して，開いているスクリプトをチェック・アウトできます。これは，［バージョンの履歴］ダイアログ・ボックスで［**チェック アウト**］ボタンを使用するのと同等の機能です。

古いバージョンのスクリプトをチェック・アウトするには，次の手順を実行します。

1 Quality Center スクリプトを開きます。最新バージョンのスクリプトが開きます。詳細については，235 ページ「Quality Center プロジェクトからスクリプトを開く」を参照してください。

2［**ファイル**］＞［**Quality Center バージョン管理**］＞［**バージョンの履歴**］を選択します。［バージョンの履歴］ダイアログ・ボックスが表示されます。

3 **バージョン詳細リスト**から表示するバージョンを選択します。

4［**チェック アウト**］ボタンをクリックします。確認メッセージが開きます。

5 古いバージョンのスクリプトをチェック・アウトすることを確認します。［チェック アウト］ダイアログ・ボックスが開き，チェック・アウトされるバージョンが表示されます。

6［**コメント**］ボックスに変更の詳細を入力します。

7 [**OK**] をクリックします。開いていたスクリプトが閉じ，選択したバージョンが書き込み可能スクリプトとして開きます。

8 必要に応じてスクリプトを表示または編集します。

9 新規の最新バージョンとしてスクリプトを Quality Center データベースにチェック・インするには，[**ファイル**] > [**Quality Center バージョン管理**] [**チェック イン**] を選択します。変更したスクリプトを Quality Center にアップロードしない場合は，[**ファイル**] > [**Quality Center 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。

スクリプトのチェック・インの詳細については，241 ページ「バージョン・コントロール・データベースへのスクリプトのチェック・イン」を参照してください。チェック・アウトの取り消しの詳細については，246 ページ「チェック・アウト操作の取り消し」を参照してください。

チェック・アウト操作の取り消し

スクリプトをチェック・アウトした後に，変更したスクリプトを Quality Center にアップロードしないと決定した場合は，ほかの Quality Center ユーザがチェック・アウトできるように，チェック・アウト操作を取り消す必要があります。

チェック・アウト操作を取り消すには，次の手順を実行します。

1 チェック・アウトしたスクリプトをまだ開いていない場合は開きます。

2 [**ファイル**] > [**Quality Center 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。

3 [**はい**] をクリックし，チェック・アウト操作の取り消しを確定します。チェック・アウト操作が取り消されます。チェック・アウトしたスクリプトが閉じ，以前チェック・インしたバージョンが読み取り専用モードで再び開きます。

第 16 章

Performance Center によるスクリプトの管理

VuGen を Performance Center と統合することにより，Performance Center サーバへのスクリプトのアップロードおよび Performance Center サーバからのスクリプトのダウンロードが行えます。

本章では，次の項目について説明します。

- Performance Center によるスクリプトの管理について
- VuGen から Performance Center への接続
- 仮想ユーザ・スクリプトのアップロード
- 仮想ユーザ・スクリプトのダウンロード

Performance Center によるスクリプトの管理について

VuGen は，Mercury の Web 対応のグローバル負荷テスト・ツールである Performance Center と統合でき，異なる場所からシステムをテストできるようになっています。

VuGen のユーザ・インタフェースを使って，Performance Center サーバとの間でスクリプトのアップロードとダウンロードを実行できます。スクリプトを Performance Center にアップロードすることにより，スクリプトを仮想ユーザ・スクリプトのリストに追加して，テストで使用できるようになります。また，スクリプトをダウンロードして，編集したりローカルに保存したりすることもできます。

VuGen から Performance Center にアクセスしてアップロードやダウンロードを行うには，Performance Center がインストールされているサーバに接続する必要があります。

Performance Center の使用方法の詳細については，『**Performance Center User's Guide**』（英語のみ）を参照してください。

VuGen から Performance Center への接続

VuGen と Performance Center が連携して動作することにより，Performance Center との間で仮想ユーザ・スクリプトのアップロードとダウンロードを行う効率的な手段が提供されます。VuGen で Performance Center プロジェクトにアクセスするには，最初に VuGen を Performance Center 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。接続後，仮想ユーザ・スクリプトをアップロードまたはダウンロードできます。詳細については，次を参照してください。

➤ 249 ページ「仮想ユーザ・スクリプトのアップロード」

➤ 254 ページ「仮想ユーザ・スクリプトのダウンロード」

Performance Center に接続するには，［Performance Center 接続設定］ダイアログ・ボックスを使用します。

VuGen を Performance Center に接続するには，次の手順を実行します。

1 VuGen で，［**ツール**］＞［**Performance Center への接続**］を選択します。［Performance Center 接続設定］ダイアログ・ボックスが開きます。

2［URL］ボックスに，Performance Center がインストールされている Web サーバの URL アドレスを入力します。URL アドレスは，次の形式で指定します。

http:// ＜サーバ名＞ /loadtest

3 ユーザ名とパスワードを入力します。詳しくは Performance Center の管理者に問い合わせてください。

4 ログイン処理を自動化するには，［**ユーザ名とパスワードを記憶する**］を選択します。指定したユーザ名とパスワードはレジストリに保存され，ダイアログ・ボックスを開くたびに表示されます。

5 VuGen の起動時に Performance Center への接続を自動的に開くには，［**起動時に自動的に接続する**］を選択します。VuGen は，表示された設定情報を使って Performance Center への接続を試行します。

6 Performance Center に接続するには，［**OK**］をクリックします。［Performance Center への接続］ダイアログ・ボックスに，接続ステータスが表示されます。

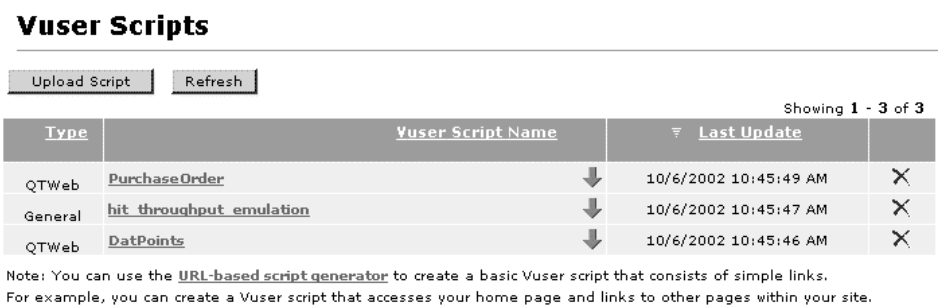
接続が確立されると，すべてのフィールドが読み取り専用で表示されます。

注：接続に失敗すると，接続に失敗した理由を示すダイアログ・ボックスが表示されます。Performance Center と Quality Center に同時に接続することはできません。

仮想ユーザ・スクリプトのアップロード

Performance Center プロジェクトで仮想ユーザ・スクリプトを使用するには，仮想ユーザ・スクリプトを Performance Center サーバにアップロードする必要があります。Performance Center サーバの仮想ユーザ・スクリプトのリストには，プロジェクトで使用できるすべての格納済みの仮想ユーザ・スクリプトが表示されます。

［仮想ユーザ スクリプト］ページを開くには，［**プロジェクト**］メニューの［**仮想ユーザ スクリプト**］を選択します。



仮想ユーザ・スクリプトが仮想ユーザ・スクリプトのリストにすでに追加されている場合は，Performance Center の［仮想ユーザ スクリプト］ページに追加されたスクリプトが表示されます。このリストには，負荷テスト中に仮想ユーザが使用できるすべての仮想ユーザ・スクリプトが示されています。VuGen を使用してアップロードされたスクリプトは，仮想ユーザ・スクリプトのリストに自動的に追加されます。

仮想ユーザ・スクリプトのリストに仮想ユーザ・スクリプトが存在せず，新しいスクリプトを追加する場合は，次の方法でスクリプトを追加できます。

- VuGen からの仮想ユーザ・スクリプトのアップロード
- Performance Center からの仮想ユーザ・スクリプトのアップロード

使用している VuGen のバージョンの確認

アップロードを処理するには，使用しているバージョンの VuGen を正しく設定し，VuGen を Performance Center に接続する必要があります。

VuGen のアップロード設定が正しいことを確認するには，次の手順を実行します。

1 VuGen を起動し，新規または既存の仮想ユーザ・スクリプトを開きます。

2［**ツール**］を選択します。

ドロップダウン・メニューに［**Performance Center 接続設定**］というメニュー項目が存在する場合は，使用しているバージョンの VuGen でスクリプトをアップロードできます。

使用しているバージョンの VuGen でスクリプトをアップロードできない場合や，マシンに VuGen をインストールしていない場合は，新しいバージョンの VuGen をインストールする必要があります。古いバージョンはアンインストールすることをお勧めします。VuGen をアンインストールするには，［**スタート**］メニューから［仮想ユーザ ジェネレータ］のアンインストール・オプションを選択します。

新しいバージョンの VuGen をインストールするには，次の手順を実行します。

1［**その他**］メニューから，［**ダウンロード**］を選択します。

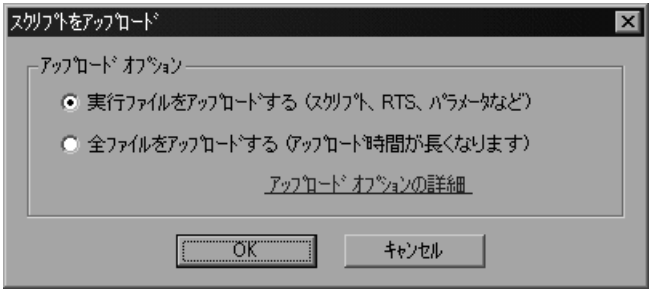
2［**Standalone LoadRunner VuGen**］を選択します。

3 ダウンロードの手順を進めます。

アップロードに対応する VuGen のバージョンをインストールすると，既存のスクリプトをアップロードしたり，アップロードするスクリプトを新たに記録したりできます。

アップロード・オプション

VuGen の内部から Performance Center Web サイトのスクリプト・リポジトリに仮想ユーザ・スクリプトをアップロードできます。要件に応じて，部分的なアップロードと完全なアップロードのいずれかを実行できます。次のオプションが使用できます。



➤ ［**実行ファイルをアップロードする**］: VuGen は最初にサーバからすべての主要なスクリプト・ファイル（**usr**，**c**，**cfg**，および **xml** ファイル）を削除します。データ・ファイルまたは古い記録データは削除されません。次に，VuGen はスクリプト・ファイル，実行環境設定，パラメータ・ファイルをアップロードします。

➤ ［**全ファイルをアップロードする**］: VuGen は最初に**すべての**スクリプト・ファイルおよびデータ・ファイルをサーバから削除します。現在のスクリプト・ファイルとデータ・ファイル（記録データや再生の結果ディレクトリを含む）がアップロードされます。

実行時ファイルのみをアップロードすると，VuGen はすべての記録データと再生結果をアップロードするのではなくスクリプト・ファイルだけをアップロードするため，時間が短縮されます。

注 : スクリプト・ファイルがすでにダウンロードされている場合は，標準ではダウンロードされたファイルだけがアップロードされます。新規に作成されたファイルをアップロードする場合（例えば，ダウンロードしたスクリプトを再生してスナップショットを作成した場合など）は，すべてのファイルをアップロードするように指定する必要があります。

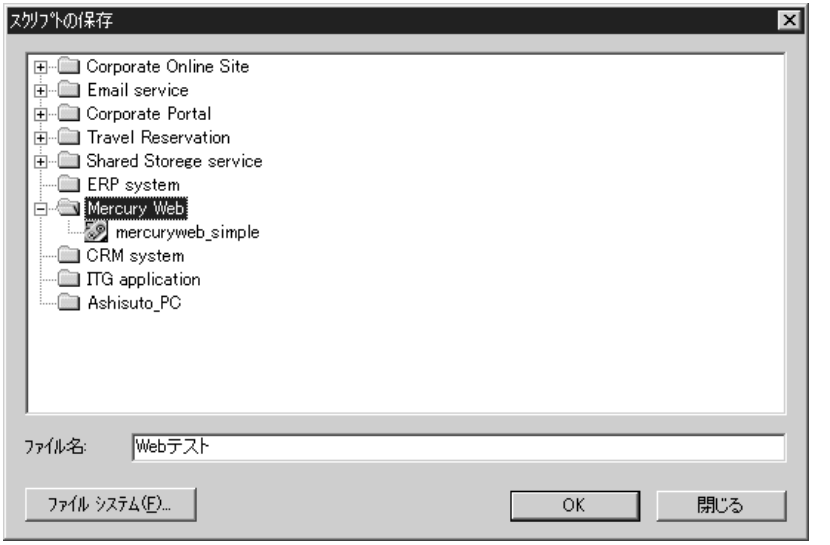
VuGen からの仮想ユーザ・スクリプトのアップロード

VuGen に接続すると，スクリプト・ファイルを Performance Center サーバにアップロードできます。詳細については，248 ページ「VuGen から Performance Center への接続」を参照してください。

VuGen が Performance Center に接続されていない場合は，仮想ユーザ・スクリプトをローカルのファイル・システムに保存できます。後で VuGen が Performance Center に接続されたら，VuGen でスクリプトを開き，それを以下に説明する方法で Performance Center にアップロードします。

VuGen から Performance Center に仮想ユーザ・スクリプトをアップロードするには，次の手順を実行します。

1 VuGen で，［**ファイル**］＞［**保存**］を選択します。［スクリプトの保存］ダイアログ・ボックスが開きます。



2 スクリプトの保存先となるプロジェクトを選択します。スクリプトの名前を［**ファイル名**］ボックスに入力します。

注：ファイル名は，英文字，数字，アンダーバーのみで構成し，250 文字以内で入力する必要があります。

3［**OK**］をクリックします。［スクリプトをアップロード］ダイアログ・ボックスが表示されます。アップロード・オプション（［**実行ファイルをアップロードする**］または［**全ファイルをアップロードする**］）のいずれかを選択します。

4［**OK**］をクリックすると，ファイルが Performance Center サーバにアップロードされます。

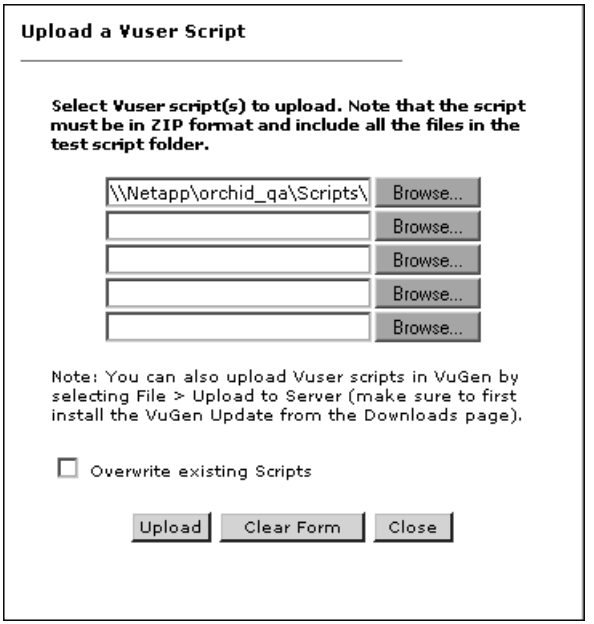
Performance Center からの仮想ユーザ・スクリプトのアップロード

VuGen がインストールされていない場合でも，Performance Center の［**Vuser Scripts**］ページにある［Upload Script］機能を使用してスクリプトをアップロードできます。

Performance Center から仮想ユーザ・スクリプトをアップロードするには，次の手順を実行します。

1［仮想ユーザ スクリプト］ページを開きます。

2［**Upload Script**］をクリックします。［Upload a Vuser Script］ダイアログ・ボックスが表示されます。



3［**Browse**］ボタンをクリックして，アップロードする各仮想ユーザ・スクリプトが格納されている zip ファイルを参照します。zip ファイルには，仮想ユーザ・スクリプト・フォルダの完全な内容（仮想ユーザ・スクリプト・ファイル（「.usr」ファイル）そのものと，関連するすべてのデータ・ファイルを含む）が格納されている必要があります。

4 zip ファイルを選択して，［**OK**］をクリックします。

5 仮想ユーザ・スクリプトのリストにすでに存在するスクリプトを置き換える場合は，［**Overwrite existing Scripts**］チェック・ボックスを選択します。

6［**Upload**］をクリックすると，スクリプトがアップロードされ，仮想ユーザ・スクリプトのリストに追加されます。

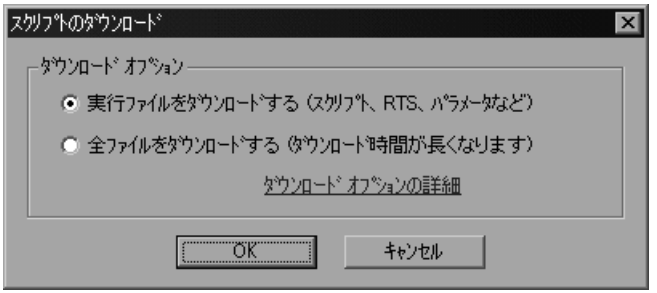
仮想ユーザ・スクリプトのダウンロード

VuGen と Performance Center が連携して動作することにより，Performance Center から仮想ユーザ・スクリプトをダウンロードし，VuGen で自動的に開いて編集するための効率的な手段が提供されます。スクリプト実行時のファイルのみをダウンロードするか，記録データや再生結果を含む完全なファイルをダウンロードするかを選択できます。

VuGen で Performance Center プロジェクトにアクセスするには，最初に VuGen を Performance Center 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。接続後，ダウンロードするスクリプト・ファイルを選択できます。［Performance Center 接続設定］ダイアログ・ボックスから Performance Center に接続します。詳細については，248 ページ「VuGen から Performance Center への接続」を参照してください。

ダウンロード・オプション

Performance Center のスクリプト・リポジトリから VuGen に仮想ユーザ・スクリプトをダウンロードできます。要件に応じて，部分的なダウンロードと完全なダウンロードのいずれかを実行できます。次のオプションが使用できます。



- ［**実行ファイルをダウンロードする**］: VuGen は，ダウンロード時間をより早くするため，スクリプト・ファイルのみをダウンロードします。ダウンロードには，スクリプト・ファイル，実行環境の設定，およびパラメータ・ファイルが含まれます。
- ［**全ファイルをダウンロードする**］: スクリプト・ファイルとデータ・ファイル（記録データや再生の結果ディレクトリを含む）がダウンロードされます。

実行時のファイルのみを対象とする部分的なダウンロードは，スクリプト・ファイルがダウンロードされるだけなので高速です。スクリプト・ファイルとデータ・ファイルをすべてダウンロードすると，ダウンロードに多くの時間がかかります。

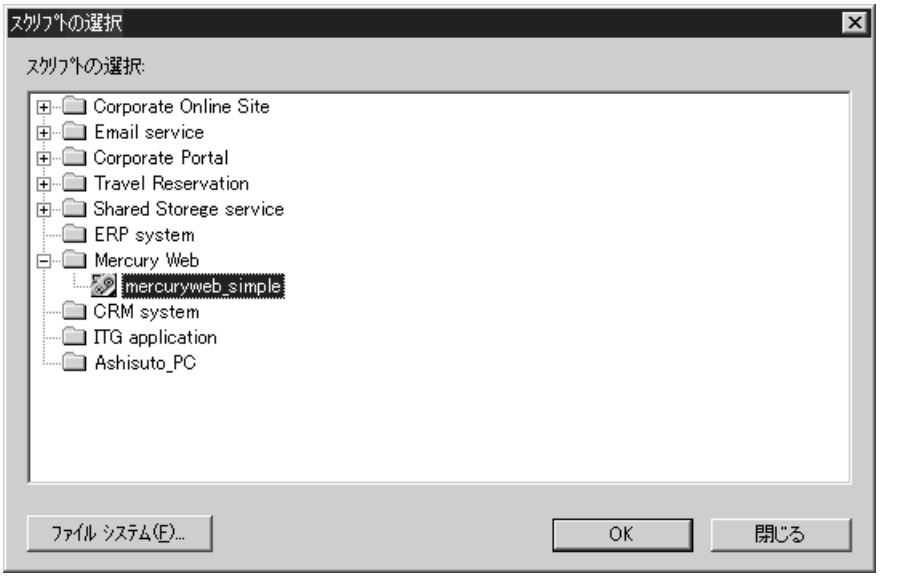
VuGen からの仮想ユーザ・スクリプトのダウンロード

Performance Center サーバに接続すると，スクリプト・ファイルを VuGen にダウンロードできます。

Performance Center から仮想ユーザ・スクリプトをダウンロードするには，次の手順を実行します。

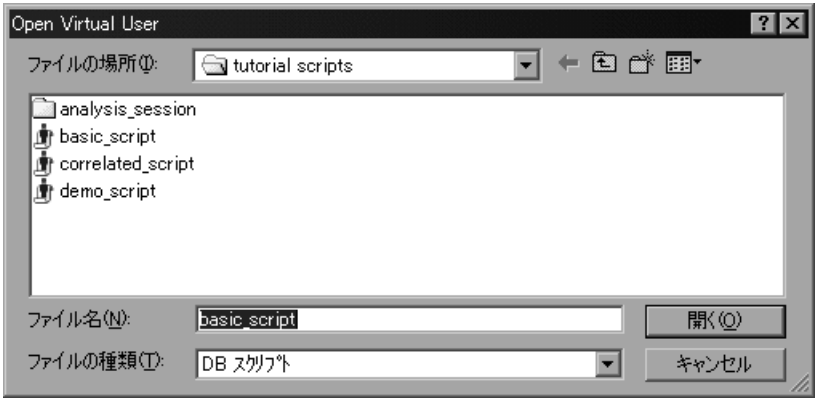
1 Performance Center サーバに接続します。詳細については，248 ページ「VuGen から Performance Center への接続」を参照してください。

2 VuGen で，［**ファイル**］＞［**開く**］を選択します。［スクリプトの選択］ダイアログ・ボックスが表示されます。



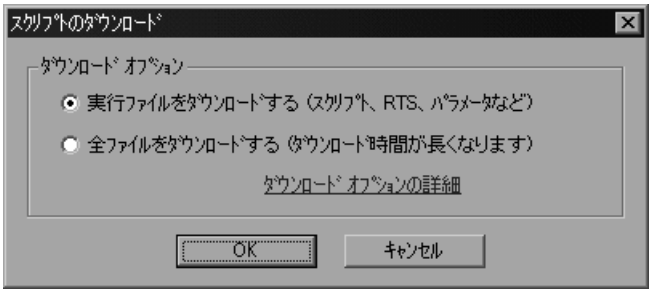
3 ダウンロードするスクリプトを選択します。

（Performance Center に接続していても）ローカル・ドライブからスクリプトを選択するには，［**ファイル システム**］をクリックします。［Open Virtual User］ダイアログ・ボックスが表示されます。



ダウンロードするファイルを選択して，［**開く**］をクリックします。Performance Center の［スクリプトの選択］ダイアログ・ボックスが再度表示されます。ダウンロードするスクリプトを選択します。

4［**OK**］をクリックします。［スクリプトのダウンロード］ダイアログ・ボックスが表示されます。



5 ダウンロード・オプションを［**実行ファイルをダウンロードする**］または［**全ファイルをダウンロードする**］のいずれかから選択します。

6［**OK**］をクリックすると，Performance Center からファイルがダウンロードされます。ダウンロードが完了すると，ダイアログ・ボックスが閉じてスクリプトが表示されます。

標準設定では，ダウンロードしたファイルは temp ディレクトリに保存されます。ファイルを別のディレクトリに保存するには，［**保存**］をクリックしてディレクトリを指定します。

第 3 部

GUI 仮想ユーザ・スクリプト

第 17 章

GUI 仮想ユーザ・スクリプトの作成

GUI 仮想ユーザは，Windows 環境で GUI アプリケーションを操作します。WinRunner（マーキュリー・インタラクティブの GUI アプリケーション用自動テスト・ツール）を使用して，GUI 仮想ユーザ・スクリプトを作成します。

本章では，以下の項目について説明します。

- GUI 仮想ユーザ・スクリプトの作成について
- GUI 仮想ユーザの紹介
- GUI 仮想ユーザ技術について
- GUI 仮想ユーザを使った作業の開始
- WinRunner による GUI 仮想ユーザ・スクリプトの作成
- サーバ・パフォーマンスの測定：トランザクション
- 大きいユーザ負荷の生成：ランデブー・ポイント
- GUI 仮想ユーザ・スクリプトについて
- GUI 仮想ユーザ・スクリプトでの仮想ユーザ関数 の使用法
- コントローラまたはコンソールへのメッセージの送信
- 仮想ユーザとロード・ジェネレータについての情報の取得

以降の情報は，GUI 仮想ユーザ・スクリプトを対象とします。

GUI 仮想ユーザ・スクリプトの作成について

GUI 仮想ユーザを使えば，クライアント / サーバ・システムに負荷をかけたときのエンド・ツー・エンドのユーザ側の応答時間の測定および監視が行えます。GUI 仮想ユーザは，実際のユーザの操作環境を完全にエミュレートします。たとえば，実際のユーザはマシンの前に座り，キーボードとマウスを使用してアプリケーションを操作し，モニタ画面の情報を読みます。これと同様に，GUI 仮想ユーザもそれぞれのマシンで実行され，アプリケーションを操作します。仮想ユーザをプログラミングし，モニタ画面に表示される情報を読み込んだり，操作したりできます。

GUI 仮想ユーザ・スクリプト

各 GUI 仮想ユーザのアクションは，「**GUI 仮想ユーザ・スクリプト**」に記述されます。WinRunner を使って，GUI 仮想ユーザ・スクリプトを作成します。WinRunner は，GUI 仮想ユーザ・スクリプトの作成，編集，デバッグを行うための自動 GUI テスト・ツールです。

GUI 仮想ユーザ・スクリプトは，マーキュリー・インタラクティブのテスト・スクリプト言語（TSL）を使って作成します。TSL は，高度な C 言語に似た，プログラミング言語です。TSL は，強力かつ柔軟な従来のプログラミング言語の特徴と，クライアント / サーバ・システムのテスト用に設計された関数を組み合わせます。

基本的な仮想ユーザ・スクリプトを記録した後で，サーバのパフォーマンスを測定したり（トランザクション），同期されたユーザ負荷を作成したり（ランデブー・ポイント）するためのステートメントをスクリプトに挿入します。GUI 仮想ユーザの詳細については，「**オンライン関数リファレンス**」を参照してください。

GUI 仮想ユーザの紹介

現金自動預払い機（ATM）を管理している銀行のサーバを考えてみます。次のことを行う GUI 仮想ユーザ・スクリプトを作成できます。

- ATM アプリケーションを開く
- 口座番号を入力する
- 引き出す現金の金額を入力する
- 口座から現金を引き出す
- 口座の残高を確認する
- ATM アプリケーションを閉じる
- 処理を繰り返す

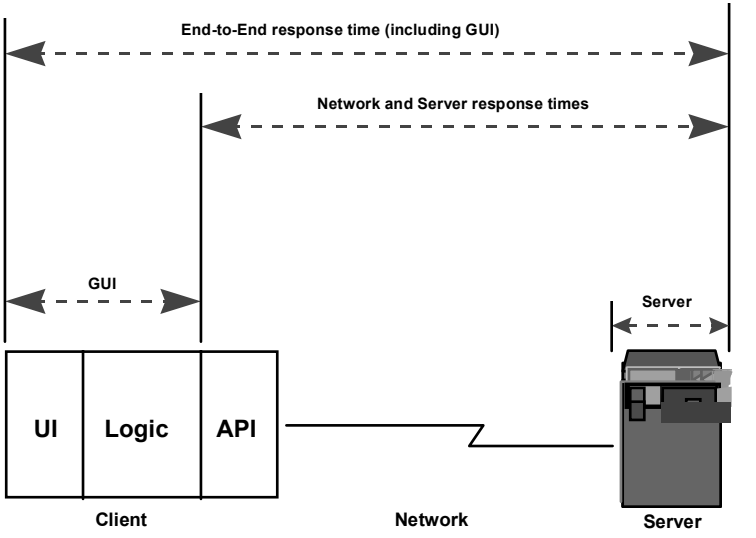
GUI 仮想ユーザの監視と管理は，コントローラまたはコンソールだけを使って行います。たとえば，コントローラまたはコンソールを使って，仮想ユーザの実行，一時停止，表示およびシナリオまたはセッション・ステップのステータスの監視ができます。

GUI 仮想ユーザ技術について

この項では，仮想ユーザ・スクリプトの作成方法について説明します。仮想ユーザ・スクリプトの作成の詳細については，**WinRunner User's Guide** を参照してください。

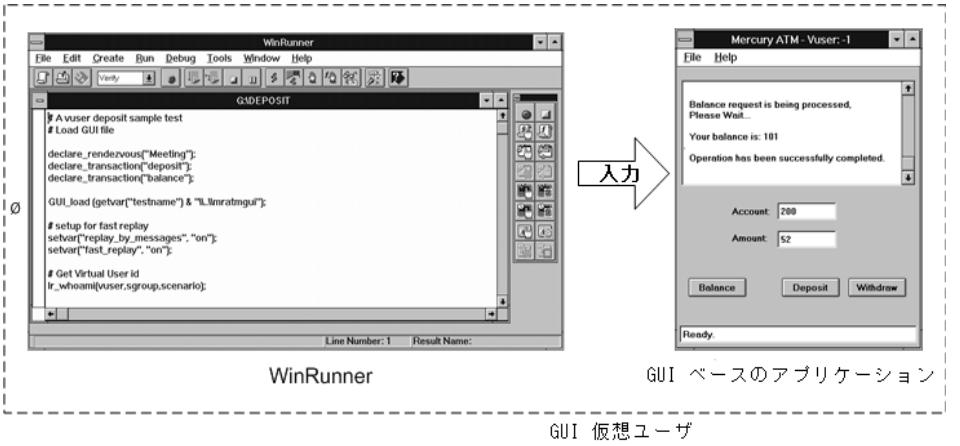
注：VuGen では，GUI 仮想ユーザ・スクリプトを実行できません。GUI 仮想ユーザ・スクリプトをシナリオまたはセッション・ステップの一部として実行するには，コントローラまたはコンソールを使います。スタンドアロン・モードで GUI 仮想ユーザ・スクリプトを実行するには，WinRunner を使います。

GUI 仮想ユーザは，エンド・ツー・エンドの実際の応答時間を測定します。エンド・ツー・エンドの応答時間は，ユーザが要求を出してから応答を得るまでの合計待ち時間を表します。エンド・ツー・エンドの応答時間には，GUI，ネットワーク，サーバそれぞれの応答時間が含まれます。



GUI 仮想ユーザ

GUI 仮想ユーザは，WinRunner（マーキュリー・インタラクティブの Windows ベース・アプリケーション向けの GUI テスト・ツール）と GUI ベースのアプリケーションで構成されています。.



WinRunner は実際のユーザに代わってアプリケーションを操作します。たとえば，WinRunner がメニューからコマンドを選択したり，アイコンをクリックしたり，テキストを入力したりします。サーバにアクセスするアプリケーションはすべて，WinRunner で操作できます。

GUI 仮想ユーザを使った作業の開始

この項では，GUI 仮想ユーザ・スクリプトを作成し，シナリオまたはセッション・ステップに組み込む方法を説明します。

1 **WinRunner を使って GUI 仮想ユーザ・スクリプトを作成します。**

WinRunner で，GUI ベースのアプリケーションに対するキーボードやマウスの操作を記録します。詳細については，**WinRunner User's Guide** を参照してください。

2 **追加の TSL ステートメントをスクリプトにプログラミングします。**

ループやその他のフロー制御構造を使用して，仮想ユーザ・スクリプトを拡張します。スクリプトの編集については，**WinRunner User's Guide** を参照してください。TSL の詳細については，**TSL Online Reference** を参照してください。

トランザクションを挿入してシステムのパフォーマンスを測定します。

仮想ユーザ・スクリプトにトランザクションを挿入して，サーバのパフォーマンスを測定します。

ランデブー・ポイントを挿入して，サーバを対象に大きなユーザ負荷を生成します。

ランデブー・ポイントを使用して，複数の仮想ユーザにまったく同時にタスクを実行させることができます。

3 **仮想ユーザ・スクリプトを実行します。**

スクリプトを実行して，正しく機能することを確認します。必要に応じて，スクリプトをデバッグします。詳細については，**WinRunner User's Guide** を参照してください。

GUI 仮想ユーザ・スクリプトが作成できたら，スクリプトをシナリオまたはセッション・ステップに組み込みます。仮想ユーザ・スクリプトをシナリオまたはセッション・ステップに組み込む方法の詳細については，『**LoadRunner コントローラ・ユーザーズ・ガイド**』および『**Tuning Module Console User's Guide**』を参照してください。

WinRunner による GUI 仮想ユーザ・スクリプトの作成

WinRunner を使って，GUI 仮想ユーザ・スクリプトを作成します。WinRunner を使って基本的な仮想ユーザ・スクリプトを作成した後，以下を手作業で挿入します。

- スクリプトにトランザクション・ステートメントを挿入し，サーバのパフォーマンスを測定します。詳細については，268ÉyÅ[ÉWÅu サーバ・パフォーマンスの測定：トランザクション」を参照してください。
- スクリプトにランデブー・ステートメントを挿入し，指定したユーザ負荷をエミュレートします。詳細については，Åu 大きいユーザ負荷の生成：ランデブー・ポイント」次項を参照してください。

GUI 仮想ユーザ・スクリプトの作成

WinRunner は，Windows ベースの GUI 仮想ユーザ・スクリプトの作成，編集，デバッグを行うための完全な開発環境です。WinRunner を使用して，アプリケーションに対する実際のユーザのアクションを記録します。たとえば，ユーザが ATM に口座番号を入力し，50 ドル引き出す操作を記録できます。これらのアクションは，マーキュリー・インタラクティブのテスト・スクリプト言語（TSL）を使って，スクリプトに自動的に記録されます。

サーバ・パフォーマンスの測定：トランザクション

トランザクション

「**トランザクション**」を定義して，サーバのパフォーマンスを測定します。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基本的な処理の場合や，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には，仮想ユーザ関数を挿入し，タスクの開始と終了を示します。スクリプト内に任意の数の分析用トランザクションを挿入できます。それぞれ異なる名前を付けて，異なる場所を先頭と終了の位置として指定できます。

シナリオまたはセッション・ステップの実行中，コントローラまたはコンソール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。たとえば，仮想ユーザの残高照会の要求を処理して口座の残高を確認するのに銀行のサーバがどれだけの時間を要したか測定するというトランザクションを定義できます。スクリプトの実行後に，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のパフォーマンスを分析します。

トランザクションは一度定義すれば，それを使用して異なる負荷のもとでのサーバ・パフォーマンスを測定できます。たとえば，1 人，100 人，あるいは 1000 人のユーザ負荷のもとで，同じトランザクションを使ってサーバのパフォーマンスを測定できます。シナリオまたはセッション・ステップの実行中，仮想ユーザはトランザクションのパフォーマンス・データを蓄積していきます。シナリオの実行後，この情報を使用してパフォーマンスの分析レポートやグラフを作成します。

トランザクションの開始位置を指定するには，次の手順で行います。

start_transaction ステートメントを仮想ユーザ・スクリプトに挿入します。

トランザクションの終了位置を指定するには，次の手順で行います。

end_transaction ステートメントを仮想ユーザ・スクリプトに挿入します。

start_transaction と **end_transaction** 関数の構文については，**TSL Online Reference**（WinRunner の［ヘルプ］メニューから表示できます）を参照してください。

大きいユーザ負荷の生成：ランデブー・ポイント

ランデブー・ポイント

大きなユーザ負荷をエミュレートしてサーバのパフォーマンスを測定するには，仮想ユーザを同期させて，まったく同時にクエリを実行します。ランデブー・ポイントと呼ばれる「待ち合わせ場所」を作成することによって，複数の仮想ユーザが必ず同時にアクションを実行するよう設定します。ある仮想ユーザがランデブー・ポイントに到達すると，コントローラまたはコンソールによって，他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。ランデブーの条件が満たされると，コントローラまたはコンソールによって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，「待ち合わせ場所」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，コントローラまたはコンソール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。

例えば，銀行のサーバに最大の負荷をかけるには，ランデブー・ポイントを仮想ユーザ・スクリプトに挿入し，すべての仮想ユーザが同時に預金するように指示します。

ランデブー・ポイントを挿入するには，次の手順で行います。

rendezvous ステートメントを仮想ユーザ・スクリプトに挿入します。

rendezvous 関数の構文の詳細については，**TSL Online Reference**（WinRunner の［ヘルプ］メニューから表示できます）を参照してください。

GUI 仮想ユーザ・スクリプトについて

GUI 仮想ユーザ・スクリプトは，マーキュリー・インタラクティブのテスト・スクリプト言語（TSL）を使用して作成します。TSL は，高度な C 言語に似たプログラミング言語です。TSL は，強力かつ柔軟な従来のプログラミング言語の特性と，クライアント / サーバ・システムのテスト用に設計された関数を組み合わせます。TSL の詳細については，**TSL Online Reference**（WinRunner の［ヘルプ］メニューから表示できます）を参照してください。

本項では，WinRunner で作成した基本的な仮想ユーザ・スクリプトを紹介します。このスクリプトは，UNIX マシンでは動作しないので注意してください。このスクリプトは ATM アプリケーション（mratm.exe）を起動し，口座番号を入力し，50 ドル預金して，ATM アプリケーションを終了します。

スクリプトの最初のセクションで，アプリケーションを起動し，そのアプリケーションを画面上の新しい場所に移動します。**system** 関数で，ATM アプリケーションを起動します。**win_move** 関数で，ATM アプリケーションを画面上の指定の場所に動かします。

```
# ATM クライアント・アプリケーションを初期化して起動する
system ("mratm.exe");
win_move ("Mercury ATM", 325, 0);
```

次に，仮想ユーザが口座番号を ATM アプリケーションに入力します。**set_window** 関数で ATM ウィンドウをアクティブにします。**edit_set** 関数で口座番号を ATM アプリケーションの口座番号のフィールドに入力するように仮想ユーザに指示します。

```
# Account フィールドに口座番号を入力する
account = 100;
set_window ("Mercury ATM");
edit_set ("Account", account);
```

口座番号を入力した後，仮想ユーザは預金する金額を入力し，［Deposit］ボタンを押します。**edit_set** 関数で，預金額を金額フィールドに入力します。

button_press 関数は，仮想ユーザに ATM の［Deposit］ボタンを押すよう指示します。

```
# Amount フィールドに預金額を入力する
amount = 50;
set_window ("Mercury ATM");
edit_set ("Amount", amount);
#［Deposit］ボタンを押す。
button_press ("Deposit");
```

このテストの最後のセクションで，仮想ユーザに ATM アプリケーションを終了するよう指示します。**menu_select_item** 関数で［File］メニューから［Exit］コマンドを選択します。

```
# クライアント・アプリケーションを閉じる
menu_select_item ("File; Exit");
```

GUI 仮想ユーザ・スクリプトでの仮想ユーザ関数 の使用法

本項では，GUI 仮想ユーザ・スクリプトの拡張に使用できる仮想ユーザ関数のいくつかを示します。関数の構文と使用例については，本書中の関連する項，または **TSL Online Reference**（WinRunner の［ヘルプ］メニューから表示できます）を参照してください。

declare_rendezvous	ランデブーを宣言します。
declare_transaction	トランザクションを宣言します。
end_transaction	パフォーマンス分析を実行するためのトランザクションの終了位置を示します。
error_message	エラー・メッセージをコントローラまたはコンソールに送信します。
get_host_name	ロード・ジェネレータの名前を返します。
get_master_host_name	コントローラまたはコンソールの名前を返します。
lr_whoami	スクリプトを実行する仮想ユーザに関する情報を返します。
output_message	コントローラまたはコンソールにメッセージを送信します。
rendezvous	仮想ユーザ・スクリプトにランデブー・ポイントを設定します。
start_transaction	パフォーマンス分析を実行するためのトランザクションの開始位置を示します。
user_data_point	ユーザ定義データのサンプル値を記録します。

コントローラまたはコンソールへのメッセージの送信

シナリオまたはセッション・ステップを実行すると，コントローラまたはコンソールの［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。WinRunner によって自動的に送信されるメッセージに加え，エラー・メッセージや通知メッセージをコントローラまたはコンソールに送信するステートメントを各スクリプトに挿入できます。たとえば，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。これらのメッセージはシナリオまたはセッション・ステップの実行後にファイルに保存できます。

error_message 関数は，エラー・メッセージをコントローラの［出力］ウィンドウまたはコンソールに送信します。この関数の構文は次のとおりです。

error_message (message);

message にはテキスト文字列を指定します。次の例では，スクリプトの実行中に致命的なエラーが発生したときに，仮想ユーザ・スクリプトがメッセージを送信します。

```
if (fatal_error < 0){
      mess = sprintf ("fatal error - Exiting.");
      error_message (mess);
      texit (1);
}
```

output_message 関数を使って，エラー・メッセージ以外の特別な通知を送ります。この関数の構文は次のとおりです。

output_message (message);

message にはテキスト文字列を指定します。

error_message 関数と **output_message** 関数の詳細については，**TSL Online Reference**（WinRunner の［ヘルプ］メニューから表示できます）を参照してください。

仮想ユーザとロード・ジェネレータについての情報の取得

シナリオまたはセッション・ステップの実行中，以下の ID を取得できます。

- シナリオまたはセッション・ステップ内のある特定の時点でタスクを実行している仮想ユーザ
- スクリプトを実行しているロード・ジェネレータ
- コントローラまたはコンソールが稼動しているマシン

たとえば，仮想ユーザ・スクリプト内にステートメントを記述し，現在アプリケーションを使用しているアクティブな仮想ユーザの ID を取得し，その情報をファイルに出力できます。

次の関数で仮想ユーザとロード・ジェネレータに関する情報を取得します。

lr_whoami	仮想ユーザ名と，その仮想ユーザが属する仮想ユーザ・グループを返します。
get_host_name	スクリプトを実行しているマシンの名前を返します。
get_master_host_name	コントローラまたはコンソールを実行しているマシンの名前を返します。

次の例では，**get_host_name** 関数を使用して，スクリプトを実行しているロード・ジェネレータの名前を取得しています。その後，**print** ステートメントで，その情報をファイルに保存しています。

```
my_host_name = get_host_name();
print("my local load generator name is:"& my_host_name) > vuser_file;
```

これらの関数の詳細については，**TSL Online Reference**（WinRunner の［ヘルプ］メニューから表示できます）を参照してください。

第 4 部

上級ユーザのために

第 18 章

Visual Studio による仮想ユーザ・スクリプトの作成

Visual C もしくは Visual Basic を使用して，Visual Studio で仮想ユーザ・スクリプトのテンプレートが作成できます。作成したテンプレートは，C または Visual Basic プログラムと同じようにコンパイルします。

本章では，以下の項目について説明します。

- Visual Studio による仮想ユーザ・スクリプトの作成について
- Visual C による仮想ユーザ・スクリプトの作成
- Visual Basic 仮想ユーザ・スクリプトの作成
- 実行環境の設定とパラメータの設定

Visual Studio による仮想ユーザ・スクリプトの作成について

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作業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 2 巻 - プロトコル**』の「ユーザ定義の仮想ユーザ・スクリプトの作成」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されるものは，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

テンプレートから基本的な仮想ユーザ・スクリプトを作成したら，スクリプトを，実行時の情報と統計値を指定して拡張します。詳細については，第 7 章「仮想ユーザ・スクリプトの拡張」を参照してください。

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。

Visual C による仮想ユーザ・スクリプトの作成

Visual C で仮想ユーザ・スクリプトを作成する場合は，バージョン 6.0 以上の Visual C を使用してください。

Visual C を使用して仮想ユーザ・スクリプトを作成するには，次の手順を実行します。

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 プロジェクトに以下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以下の変更を行います。

➤［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，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）］の _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，通常通りプログラミングします。

6 仮想ユーザ API の一般関数を使用してスクリプトを拡張します。例えば，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・キーを次のようにその DLL 名で上書きします。BinVuser= ＜ **DLL_name** ＞

次の例は，**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するものです。**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:¥mercury¥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 仮想ユーザ・スクリプトの作成

Visual Basic を使用して仮想ユーザを作成するには，次の手順を実行します。

1 Micorsoft Visual Basic で新規プロジェクトを作成します。［**ファイル**］＞［**新しいプロジェクト**］を選択します。

2［**LoadRunner Virtual User**］を選択します。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。

3 プログラミングを始める前に，プロジェクトを保存しておきます。［**ファイル**］＞［**プロジェクトの上書き保存**］を選択します。

4 オブジェクト・ブラウザ（［**表示**］メニュー）を開きます。「**LoadRunner Vuser**」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
```

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，通常通りプログラミングします。

6 オブジェクト・ブラウザを使って，トランザクション，思考遅延時間，ランデブー，メッセージなど，使用する VuGen の要素をコードに追加します。

7 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，282 ページ「実行環境の設定とパラメータの設定」を参照してください。

8 仮想ユーザ・スクリプトをビルドします。［**ファイル**］＞［**project_name.dll の作成**］を選択します。

プロジェクトは，仮想ユーザ・スクリプト形式（.usr）で保存されます。スクリプトは，プロジェクトと同じディレクトリに作成されます。

実行環境の設定とパラメータの設定

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，お使いの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，次の手順を実行します。

1 お使いの製品の **bin** ディレクトリで，**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［**File**］＞［**New**］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［**Vuser**］＞［**Advanced**］を選択し，［Advanced］ダイアログ・ボックスに DLL の名前を入力します。

4［**Vuser**］＞［**Run-time Settings**］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，第 12 章「実行環境の設定」を参照してください。

5［**Vuser**］＞［**Parameter List**］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，第 8 章「VuGen パラメータを使った作業」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。［**Vuser**］＞［**Run Vuser**］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。

6［**File**］＞［**Exit**］を選択して，設定ユーティリティを閉じます。

第 19 章

XML API プログラミング

完全な XML 構造をサポートする仮想ユーザ・スクリプトを作成できます。VuGen は，XML データの検索および操作を可能にする関数を提供します。

本章では，次の項目について説明します。

- XML API プログラミングについて
- XML 文書について － XML API プログラミング
- XML 関数の使用方法
- XML 関数のパラメータの指定
- XML 属性での作業
- XML スクリプトの作成
- 記録されたセッションの拡張

以降の情報は，主に Web，Web Services，およびワイヤレス仮想ユーザ・スクリプトを対象とします。

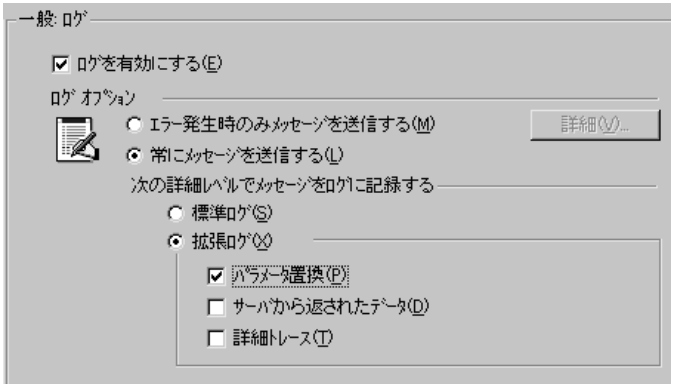
XML API プログラミングについて

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，次の手順を実行します。

- 使用するプロトコル（通常 Web，SOAP，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 文書のテキストを操作します。

［実行環境設定］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 12 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 文書について ー XML API プログラミング

XML（Extensible Markup Language）は，ユーザが独自にタグを定義できるマークアップ言語です。これらのタグを使用することによって，タグに挟まれたテキストに意味を与えます。これは，標準の HTML タグ（H1，P，DIV など）とは対照的です。標準の HTML タグはカスタマイズできず，テキストの内容を指定できません。

XML 文書は，多くのノードと分岐を持つツリーで構成されます。XML 文書を構成する部品を表すために，**タグ**，**要素**，**属性**という 3 つの用語がよく使用されます。次の例は，これらの用語を表しています。

```
<acme_org>
    <accounts_dept>
        <employee type='PT'>
            <name>John Smith</name>
            <cubicle>227</cubicle>
            <extension>2145</extension>
        </employee>
    </accounts_dept>
    <engineering_dept>
        <employee type='PT'>
            <name>Sue Jones</name>
            <extension>2375</extension>
        </employee>
    </engineering_dept>
</acme_org>
```

タグとは，左向きと右向きの山括弧の間にあるテキストです。<acme_org>，<employee>，<name> がタグの例です。タグには，開始タグ（<name> など）と終了タグ（</name> など）があります。上記の XML コードの抜粋は，John Smith と Sue Jones という 2 人の従業員がいる Acme という会社を表しています。

要素とは，開始タグおよび終了タグ，そしてそれらのタグに挟まれたすべての内容です。上記の例では，<employee> 要素には，<name>，<cubicle>，<extension> という 3 つの子要素が含まれています。

属性とは，要素の開始タグ内にある，名前と値の組み合わせです。この例では，**type='PT'** が <employee> 要素の属性です。

上記の例では，**name** というタグは **employee** の要素です。それぞれの要素には値があります。例えば，**name** 要素の値は，「John Smith」という文字列です。

XML 関数の使用方法

以降では，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ーを実行します。
lr_xml_get_values	クエリで検出された XML 要素の値を取得します。

クエリー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。

例えば，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。

出力：
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 ツリーの検索

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**＞ という形式の接尾辞を追加します。例えば，**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。例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（例えば **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 クエリーで使用しています。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 属性での作業

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。希望の属性または特定の値を持つ属性だけを変更できます。

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 スクリプトの作成

最初に，希望の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;
}
```

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 SOAP プロトコルの例では，仮想ユーザ API 関数を使って記録したセッションを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                          /glk
ev_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} *****"));
```

```
// テンプレートを使用して同時に複数の値を取得する
lr_xml_get_values_ex("Xml={ParamXml}",
                                "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 用のヘッダーを追加する
    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/IE
mailservice",
        "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;
}
```

第 20 章

VuGen のデバッグのヒント

本章では，エラーのない仮想ユーザ・スクリプトを作成できるように，詳細なデバッグ情報を取得する方法をいくつか紹介します。

- 一般的なデバッグのヒント
- C 関数を使用した追跡
- **付加的な C 言語のキーワードの追加**
- 再生出力の検証
- データベース・アプリケーションのデバッグ
- Oracle Applications を使った作業
- Oracle 2-Tier 仮想ユーザでの一般的な問題の解決方法
- 2-tier データベースのスクリプト作成のヒント
- PeopleSoft-Tuxedo スクリプトの実行

一般的なデバッグのヒント

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);

例えば，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 言語のキーワードの追加

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，次の手順を実行します。

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 ドライバ実行の出力を表示する **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。

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**
- **コンパイラ情報の検証**
- **コード生成情報**
- **前処理とコンパイルの情報**

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 データ ソース アドミニストレータ］を開いて，［トレース］タブで［トレースの開始］をクリックします。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。

Oracle Applications を使った作業

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。

Oracle 2-Tier 仮想ユーザでの一般的な問題の解決方法

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 に次の 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。fnd_signon.audit_responsibility(...)

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，次の PL/SQL ブロックの含まれる lrd_stmt の後に現れます。fnd_signon.audit_user(...).

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。

例えば，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。

対処方法

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，記録されたスクリプトのカラム名が一意でない場合に生じます。例えば，次のようにします。

```
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".
```

対処方法

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。例えば，次のようにします。

```
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。

例えば，次のような場合です。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）。

対処方法

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"

対処方法

お使いの製品の 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 Mercury 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。例えば，**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 Mercury 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 のその他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。例えば，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 のセクションを，

```
805=lrdo32.dll+ora805.dll
```

から次のように変更します。

```
805=lrdo32.dll+ora803.dll
```

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。Siebel 固有の問題の解決策については，315 ページ「Siebel 固有のスクリプト作成のヒント」を参照してください。

質問 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。

質問 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);
```

質問 3 : vdf.h ファイルのデータ宣言が原因で VuGen で生成したコードのコンパイルが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。カスタマー・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。

質問 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。

例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。

質問 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);
```

質問 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。

質問 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されていないことがあります。

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 が他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。

質問 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？例えば，次のような場合です。

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);
```

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。例えば，次のような場合です。

```
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);
```

Siebel 固有のスクリプト作成のヒント

本項では，Siebel データベース・ユーザのための解決策を示します。前項までの記述にも，一般的なデータベース・スクリプト作成のヒントがありますので参照してください。

質問 9：VuGen で仮想ユーザは正常に実行できるのにコントローラまたはコンソールでは重複キー割り当て違反になります。

回答：Siebel クライアントは S_SSA_ID テーブルの NEXT_SUFFIX カラムにキーを格納します。このクライアントには接尾辞値のブロックを取得できない場合に，その状況を検出してそこから回復するためのコードがあります。

VuGen は自動的に S_SSA_ID テーブルで，NEXT_SUFFIX および MODIFICATION_NUM フィールドを相関します。UPDATE 中，MODIFICATION_NUM フィールドは 1 つ増加し，NEXT_SUFFIX フィールドは 36 ベースで 100 増加します。ただし，クライアントが新しい接尾辞値のブロックを取得できない場合，VuGen ではインスタンスにコードが追加されません。その結果，新しい値をデータベースに挿入しようとすると，再生時に一意制約エラーが生じます。

1 回目の試行が失敗したときに再試行するために，スクリプトで接尾辞のブロックを取得する各箇所に，手作業でコードを追加します。スクリプト内で

「SiebelPreSave」を検索して，該当箇所を見つけます。次の例に似たコードを含む **while** ループも追加する必要があります。この例は，Oracle のみに該当します。ODBC の場合は，**lrd_exec** の 4 番目の引数ではなく，**lrd_row_count** を使用します。

```
unsigned long lRowUpdated;
int nAttempt;

. . .

//「next_suffix」を取得するまでループが継続します。
lRowUpdated = 0;
nAttempt=0;

while (lRowUpdated != 1) {

   nAttempt++;
   if (nAttempt > 1)
      lr_output_message (".....Next suffix retry %d", nAttempt);
   else
   {
      lrd_open_cursor(&Csr13, Con1, 0);
      lrd_stmt(Csr13, "SELECT¥n T1.LAST_UPD,¥n T1.CREATED_BY,¥n "
         "T1.CONFLICT_ID,¥n T1.CREATED,¥n T1.NEXT_SUFFIX,¥n "
         "T1.ROW_ID,¥n T1.NEXT_PREFIX,¥n T1.CORPORATE_PREFIX,¥n "
         "T1.MODIFICATION_NUM,¥n T1.NEXT_FILE_SUFFIX,¥n      "
         "T1.LAST_UPD_BY¥n   FROM ¥n SIEBEL.S_SSA_ID T1", -1, 1, 1, 0);
   }
   lrd_bind_cols(Csr13, BCInfo_D375, 0);
   lrd_exec(Csr13, 0, 0, 0, 0, 0);

   SiebelPreSave_1();
   lrd_fetch(Csr13, -1, 4, 0, PrintRow26, 0);
   GRID(26);
   SiebelPostSave_1();

   if (nAttempt > 1)
   {
    lrd_open_cursor(&Csr14, Con1, 0);
    lrd_stmt(Csr14,"¥nUPDATE SIEBEL.S_SSA_ID SET¥n LAST_UPD_BY=:1,¥n "
```

```
    "NEXT_SUFFIX = :2,¥n      MODIFICATION_NUM = :3,¥n      LAST_UPD = "
    ":4¥n WHERE¥n ROW_ID = :5 AND MODIFICATION_NUM = :6¥n", -1, 1,
    1, 0);
   }
   lrd_assign_bind(Csr14, "6", " < modification_num > ", &_6_D376, 0,
              LRD_BIND_BY_NUMBER, 0);
   lrd_assign_bind(Csr14,"5", "0-11",&_5_D377,0,LRD_BIND_BY_NUMBER, 0);
   strcpy (szTimeAtNewButton, lr_eval_string("<Now>"));
   sprintf (szTimeStamp, "%s %s", lr_eval_string("<Today>"),
           szTimeAtNewButton);
   lr_save_string (szTimeStamp, "DateTimeStamp");
lrd_assign_bind(Csr14, "4", "<DateTimeStamp>", &_4_D378, 0,
      LRD_BIND_BY_NUMBER, 0);
lrd_assign_bind(Csr14, "3", "<next_modnum>", &_3_D379, 0,
      LRD_BIND_BY_NUMBER, 0);
lrd_assign_bind(Csr14, "2", "<next_suffix_x100>", &_2_D380, 0,
      LRD_BIND_BY_NUMBER, 0);
lrd_assign_bind(Csr14, "1", "1-1E1",&_1_D381,0,LRD_BIND_BY_NUMBER, 0);

// このアップデートでは接尾辞を正常に取得しない限り行を更新しません。
lrd_exec(Csr14, 1, 0, &lRowUpdated, 0, 0);
lrd_commit(0, Con1, 0);

}//while
     lr_output_message ("...Rows updated %ld", lRowUpdated);
```

質問 10：主キーの相関に使う正しい値を見つける方法は？

回答：Siebel は< **next_suffix** >をベース 36 で数学的に処理した結果に基づいてキーの値を生成する傾向があります。いくつかの記録同士を比較して，関連性を見つけられるかどうか試してみてください。Siebel で相関を行うときには，スクリプト再生に影響しないので日付フィールドを無視できます。

質問 11：重複キー違反による INSERT into S_SRV_REQ の失敗を解決する方法は？

回答：主キーは SR_NUM です。新しいバージョンの VuGen は，S_SSA_ID テーブルの NEXT_SUFFIX 値をベース 36 からベース 10 の同等の値に変換する lrd_siebel_str2num 関数を使って自動的にこのテーブルへの挿入を相関します。旧バージョンの VuGen ではこの相関が正しく行われないことがあります。

質問 12：VuGen でスクリプトを正しく再生するために必要な相関が自動的に行われません。足りない相関を追加する方法は？

回答：現在のところ VuGen は S_SSA_ID テーブルの NEXT_SUFFIX および MODIFICATION_NUM カラムの値を保存し，スクリプトで使用するときにその値をパラメータで置換しているだけです。したがって手作業で相関を追加する必要があります。**print.inl** ファイル内の **SiebelPreSave** 関数と **SiebelPostSave** 関数の相関コードは，何を相関する必要があるかわかっている場合に，特定の値を相関する方法を示す例です。

- NEXT_FILE_SUFFIX および MODIFICATION_NUM カラムは S_SSA_ID テーブルから選択される場合があります。その場合，UPDATE ステートメントはベース 36 でこの文字列，および MODIFICATION_NUM に 1 を追加して NEXT_FILE_SUFFIX を更新します。NEXT_FILE_SUFFIX の値はテーブル内の FILE_REV_NUM フィールドに挿入されることがしばしばあります。このテーブルの名前はしばしば接尾辞 **_ATT** で終わり，それが添付ファイルであることを示します。
- Siebel が UPDATE ステートメントを実行するときには必ず，値が 1 ずつ増加する MODIFICATION_NUM カラムが存在します。VuGen は S_SSA_ID テーブルに対してだけ，自動的にこの相関を行います。それ以外は手作業で相関する必要があります。
- Siebel は記録を ID 番号で参照します。Siebel は通常，特定のタイプ（例えば契約など）のすべての記録を検索し，そのタイプの特定の記録を更新または削除しようとするときに，ID 番号を使います。意味のある負荷テストを生成するには，再生中に ID 番号をパラメータで置換する必要があります。ID 番号は 1-QPF9 のように，1 桁以上の数字とハイフンに続く 1 桁以上の英数字からなります。VuGen ではこのパラメータ化は自動的に行われないので，手作業で行います。
- ほかにも相関またはパラメータ化の不足が見つかった場合は，VuGen の Siebel のサポート向上のために，Mercury のカスタマー・サポートにご連絡ください。

PeopleSoft-Tuxedo スクリプトの実行

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変更する必要があります。

[PeopleSoft-Tuxedo]
WINNT_EXT_LIBS=**lrt7.dll**

第 21 章

上級ユーザのために

本章には，VuGen の上級ユーザのための情報が含まれます。

- 記録中に生成されるファイル
- 再生中に生成されるファイル
- UNIX コマンド・ラインからの仮想ユーザの実行
- 仮想ユーザの動作の指定
- コマンド・ライン・パラメータ
- OLE サーバの記録
- .dat ファイルの検証
- 新規仮想ユーザ・タイプの追加

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以下に示します。

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・ヘッダーが含まれています。
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。
vdf.h	スクリプトで使用される C 変数定義のヘッダー・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。例えば，**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
```

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。例えば，次のように実行されます。

```
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べき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  (Vuser 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"
```

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
```

以下のコマンド・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，プリプロセス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・スクリプトと仮想ユーザの動作を 2 つの独立した情報源として作成するので，例えば待機時間，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非常に簡単に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，こうした「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，329 ページ「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**」に変更することもできます)。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）。お使いの環境で，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・ラインで他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 ユーティリティ **run_db_vuser** は，このオプションはまだサポートされていません。

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仮想ユーザ・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 カスタマー・サポートの Web サイト **http://support.mercuryinteractive.com** の Patches セクション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，他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，331 ページ「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）。

.dat ファイルの検証

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 とコントローラまたは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。例えば，「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他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。次の項では，新しいプロトコルを定義するためにファイルに追加すべき項目を説明します。

新規仮想ユーザ・タイプの追加

VuGen に新しい仮想ユーザのタイプまたはプロトコルを追加するのに必要な項目は次のとおりです。

- mdrv.dat ファイルの編集
- CFG ファイルの追加
- LRP ファイルの挿入
- テンプレートの指定

mdrv.dat ファイルの編集

まず，**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。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。使用できるすべてのパラメータを以下に示します。

```
[ < 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= <他の拡張子リスト>
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>
```

例えば，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。例えば，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
```

CFG ファイルの追加

プロトコルに標準の［実行環境の設定］を設定するために，設定ファイルを任意で指定できます。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
```

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。

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。例えば，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"

第 5 部

付録

付録 A

外部関数の呼び出し

VuGen 使用時に，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 は次のようにしてロードできます。

- 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。
- 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。

DLL のロード：ローカル

lr_load_dll 関数を使用して，DLL を仮想ユーザ・スクリプトにロードします。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には，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 のフル・パス名を指定できます。パスを指定しなかった場合は，**lr_load_library** が，C++ 関数（Windows プラットフォーム上の LoadLibrary）で使われる標準シーケンスを使って 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** と同じ検索ルールを使います。詳細については，**dlopen** の man ページなどを参照してください。

DLL のロード：グローバル

DLL をグローバルにロードして，その関数をすべての仮想ユーザ・スクリプトで利用できます。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には，次の手順を実行します。

1（VuGen 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へロードする 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。

例えば，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。

付録 B

多国語を使った作業

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。

本付録では，次の項目について説明します。

- 多国語を使った作業について
- 手作業による文字列エンコーディングの変換
- パラメータ・ファイル内の文字列エンコーディングの変換
- Web 記録および再生のための文字列エンコーディングの設定
- Accept-Language ヘッダの言語の指定
- プロトコルに関する制限
- Quality Center の統合

多国語を使った作業について

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。

手作業による文字列エンコーディングの変換

lr_convert_string_encoding 関数を使用すると，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。関数の構文は次のとおりです。

```
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;
}
```

実行ログでは，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・ファイル内の文字列エンコーディングの変換

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。

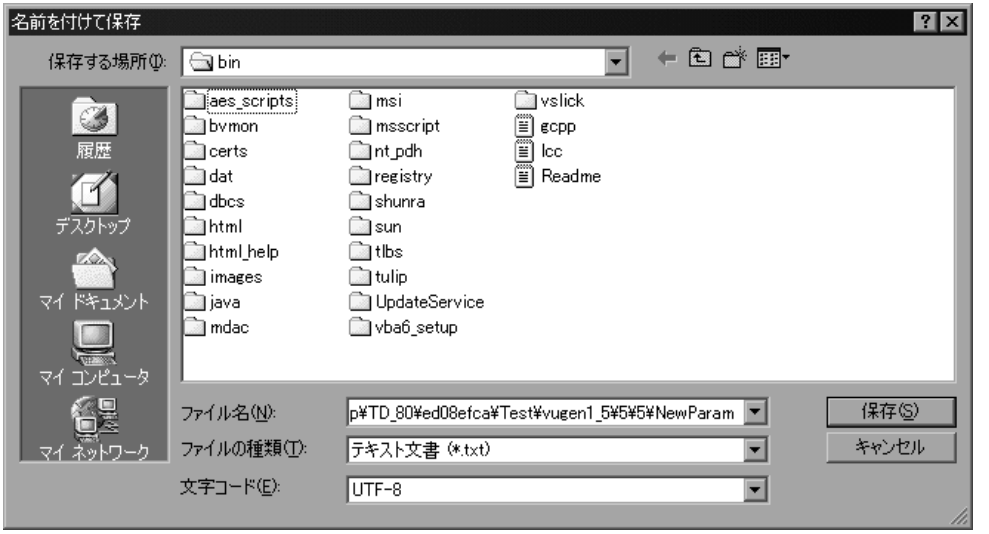
1 [**仮想ユーザ**] > [**パラメータ リスト**] を選択し，パラメータ・プロパティを表示します。

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。

Web 記録および再生のための文字列エンコーディングの設定

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。例えば，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。

本項では，次について説明します。

- ［エンコーディング］記録オプション
- 手作業によるエンコーディングの有効化
- ブラウザの設定

［エンコーディング］記録オプション

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 ヘッダーまたは 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，該当するオプションを［記録オプション］ダイアログ・ボックスの［**インターネット プロトコル：詳細**］ノードで選択します。

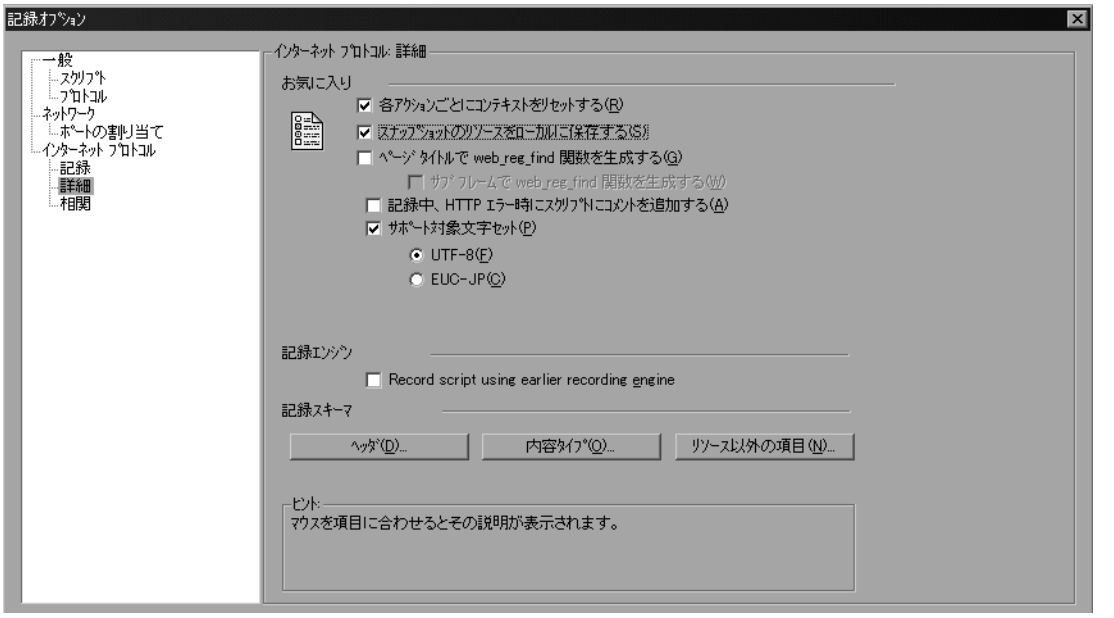
- ［**UTF-8**］: UTF-8 エンコーディングのサポートを有効にします。この設定を有効にすると，非 ASCII の UTF-8 文字がマシンのロケールで使用されているロケールに変換され，VuGen に正しく表示されます。
- ［**EUC-JP**］: 日本語版 Windows のユーザは，このオプションを選択することで，EUC-JP 文字エンコーディングを使用している Web サイトに対するサポートを有効にできます。この設定を有効にすると，EUC-JP 文字がマシンのロケールで使用されているロケールに変換され，VuGen に正しく表示されます。VuGen によって，すべての EUC-JP（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）。

［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。例えば，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。

文字セット・エンコーディングを有効にするには，次の手順を実行します。

1［記録オプション］を開き（Ctrl+F7），［**詳細**］ノードを選択します。



2［**サポート対象文字セット**］を選択します。必要な文字エンコーディングとして，［UTF-8］または［EUC-JP］（漢字対応のオペレーティング・システムでのみ有効）を選択します。

3［**OK**］をクリックします。

これらの設定の詳細については，『**第 2 巻 - プロトコル**』の「インターネット・プロトコルの記録オプションの設定」を参照してください。

手作業によるエンコーディングの有効化

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。例えば，**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。

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（例えば文字列 "&220;&" 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。

Accept-Language ヘッダの言語の指定

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。

Accept-Language ヘッダを設定するには，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（F4）を開き，［**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［詳細］セクションで，［**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［**Accept-Language リクエスト ヘッダ**］オプションを探します。［**値**］カラムで，一覧から必要な言語を選択します。この一覧は，使用しているブラウザの［インターネット オプション］の［言語］設定に基づくものです。

これらの設定の詳細については，『**第 2 巻 - プロトコル**』の「インターネット実行環境の設定」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。

Quality Center の統合

Quality Center 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，Quality Center プロジェクトに保存されているシナリオをコントローラから開く場合は，「Default」（英語）という名前の新しいテスト・セットを Quality Center プロジェクトに追加します。

付録 C

UNIX プラットフォームでのスクリプトのプログラミング

UNIX プラットフォームを利用する仮想ユーザは，プログラミングによって UNIX プラットフォーム向けのスクリプトを作成できます。プログラミングによってスクリプトを作成するには，テンプレートを使用します。

本付録では，以下の項目について説明します。

- UNIX プラットフォームで実行可能な仮想ユーザ・スクリプトのプログラミングについて
- テンプレートの生成
- 仮想ユーザのアクションのプログラミング
- 仮想ユーザの実行環境設定
- トランザクションとランデブー・ポイントの定義
- スクリプトのコンパイル

UNIX プラットフォームで実行可能な仮想ユーザ・スクリプトのプログラミングについて

UNIX プラットフォームで実行可能な仮想ユーザ・スクリプトを作成する方法は 2 つあります。1 つは VuGen を使用する方法で，もう 1 つはプログラミングを行う方法です。

VuGen
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 で実行できます（記録は UNIX ではサポートされていません）。

プログラミング
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。

本付録では，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。

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

テンプレートに基づいて基本となるスクリプトを作成したら，実行時の仮想ユーザ情報と統計値を提供できるようにスクリプトを拡張します。詳細については，第 7 章「仮想ユーザ・スクリプトの拡張」を参照してください。

テンプレートの生成

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。

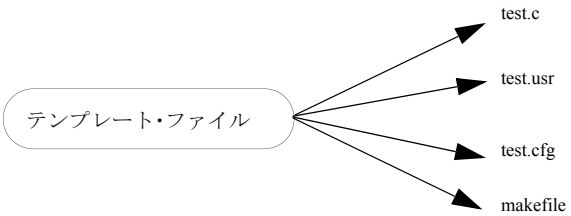
```
mkdbtest name
```

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。例えば，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。



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：Mercury アプリケーションのコントロールは，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。

仮想ユーザの実行環境設定

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 (第 12 章「実行環境の設定」を参照)。**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。次の表に従って，各パラメータ（Options，Factor，LimitFlag，Limit）を設定します。

オプション	Options	Factor	LimitFlag	Limit
思考遅延時間を無視	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
記録された思考遅延時間のランダムな割合	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
```

ログのオプション

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
```

次の表に従って，各パラメータ（LogOptions，MsGClassData，MsgClassParameters，MsgClassFull）を設定します。

ログの種類	LogOptions	MsgClassData	MsgClassParameters	MsgClassFull
Disable Logging（ログの無効化）	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

ログの種類	LogOptions	MsgClassData	MsgClassParameters	MsgClassFull
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作業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	Random（ランダム）	RunLogicRandomPaceMin, RunLogicRandomPaceMax

ペースの設定	RunLogicPaceType	関連変数
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 秒～ 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"
```

トランザクションとランデブー・ポイントの定義

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・ファイルを手作業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，上記のように **usr** ファイルにセクションを追加します。

スクリプトのコンパイル

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，以下のダイナミック・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，他のコンパイラ・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ー・ファイルをインクルードする必要があります。例えば，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，スクリプトをコマンド・ラインから実行して，正しく動作することを確認します。

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フル・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。

スクリプトの機能を検証したら，スクリプトを環境（LoadRunner シナリオ，Performance Center 負荷テスト，Tuning Module セッション，またはビジネス・

プロセス・モニタ・プロファイル）に統合します。詳細については，『**Mercury LoadRunner コントローラ・ユーザーズ・ガイド**』，または **Tuning Console**，**Performance Center**，**Application Management** のマニュアルを参照してください。

付録 D

キーボード・ショートカットの使用

以下は，仮想ユーザ・ジェネレータで使用できるキーボード・ショートカットのリストです。

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（IntelliSense）
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

索引

き