LoadRunner

仮想ユーザ・スクリプトの作成

Version 7.8

LoadRunner 仮想ユーザ・スクリプトの作成（Windows & UNIX），Version 7.8

Mercury Interactive Corporation
1325 Borregas Avenue
Sunnyvale, CA 94089 USA
Tel: (408) 822-5200
Toll Free: (800) TEST-911, (866) TOPAZ-4U
Fax: (408) 822-5300

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

LRDBUG7.8JP/01

目次

第 9 部 : Enterprise Java Bean プロトコル

第 10 部 : ERP/CRM プロトコル

LoadRunner へようこそ

LoadRunner は，マーキュリー・インタラクティブが提供する，アプリケーションのパフォーマンスをテストするツールです。LoadRunner は，アプリケーション全体に負荷をかけて，クライアント，ネットワーク，サーバの潜在的なボトルネックを検出，特定します。

LoadRunner では，制御された負荷およびピーク時の負荷のもとでのシステムの動作をテストできます。LoadRunner は，ネットワーク上に分散している多数の仮想的なユーザ，つまり**仮想ユーザ**を実行することによって負荷を生成します。こうした仮想ユーザは，最小限のハードウェア・リソースしか使わずに，一貫性を維持し，再現性があり，測定可能な負荷を生み出して，お使いのアプリケーションをあたかも実際のユーザが使っているかのように動作させます。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。

オンライン・リソース

LoadRunner には，以下のオンライン・リソースがあります。

最初にお読みください：LoadRunner 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。Acrobat Reader は，LoadRunner のインストール・パッケージに含まれています。LoadRunner オンライン・マニュアルのアップデートについては，マーキュリー・インタラクティブのカスタマー・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の関数をすべて，その使用例とともに参照できます。「**オンライン関数リファレンス**」のアップデートについては，マーキュリー・インタラクティブのカスタマー・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：LoadRunner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，LoadRunnerを使った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。LoadRunnerヘルプ・ファイルのアップデートについては，マーキュリー・インタラクティブのカスタマー・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，マーキュリー・インタラクティブのカスタマー・サポートWebサイトを開きます。このWebサイトのURLは，http://www.mercury.co.jp/supportです。

サポート情報：マーキュリー・インタラクティブのWebサイトとカスタマー・サポート・サイト，世界のマーキュリー・インタラクティブ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，マーキュリー・インタラクティブのホーム・ページを開きます。このサイトでは，マーキュリー・インタラクティブの最新情報や製品に関する情報をご覧になれます。マーキュリー・インタラクティブのWebサイトのURLは，http://www.mercury.co.jpです。

LoadRunnerのマニュアル

LoadRunnerには，以下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コントローラの使用
- 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，アナリシス・ユーザーズ・ガイド，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，『**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，以下のインストールについて説明します。

- LoadRunner コントローラー Windows ベースのマシンへのインストール
- 仮想ユーザ・コンポーネントー 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，コントローラのユーザーズ・ガイドが 1 冊含まれます。

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

アナリシス・ユーザーズ・ガイド

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

『**LoadRunner アナリシス・ユーザーズ・ガイド**』は，シナリオの実行後に LoadRunner アナリシスのグラフとレポートを使用してシステムのパフォーマンスを分析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，仮想ユーザ・ジェネレータ（VuGen）ユーザーズ・ガイドが 1 冊含まれます。

『**仮想ユーザ・スクリプトの作成**』は，VuGen を使った仮想ユーザ・スクリプトの作成方法を説明しています。必要に応じ，このマニュアルと併せて「**オンライン関数リファレンス**」と，GUI 仮想ユーザ・スクリプト用の『**WinRunner ユーザーズ・ガイド**』もお読みください。

情報	参照先
LoadRunner のインストール	『**LoadRunner インストール・ガイド**』
LoadRunner のテスト・プロセス	『**LoadRunner コントローラ・ユーザーズ・ガイド**』
仮想ユーザ・スクリプトの作成	『**仮想ユーザ・スクリプトの作成**』
シナリオの作成と実行	『**LoadRunner コントローラ・ユーザーズ・ガイド**』
テスト結果の分析	『**LoadRunner アナリシス・ユーザーズ・ガイド**』

表記規則

本書は，次の表記規則に従っています。

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**］	アクションを実行する際のインタフェース要素の名前は，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	メソッド名または関数名，メソッドや関数の引数，ファイル名，パスは，**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 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**）。
[ ]	省略可能な引数は，半角の大括弧で囲んで示します。
{ }	引数に割り当てる値の候補は，中括弧で囲んで示します。値をいずれか 1 つ割り当てる必要があります。
...	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

仮想ユーザの紹介

第 1 章

仮想ユーザ・スクリプトの作成

LoadRunner は何千人ものユーザが同時にクライアント / サーバ・システムで作業を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，LoadRunner では実際のユーザの代わりに**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されます。LoadRunner には仮想ユーザ・スクリプトの作成を支援するさまざまなツールがあります。

本章では，以下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
- 本書の使用法

仮想ユーザの紹介

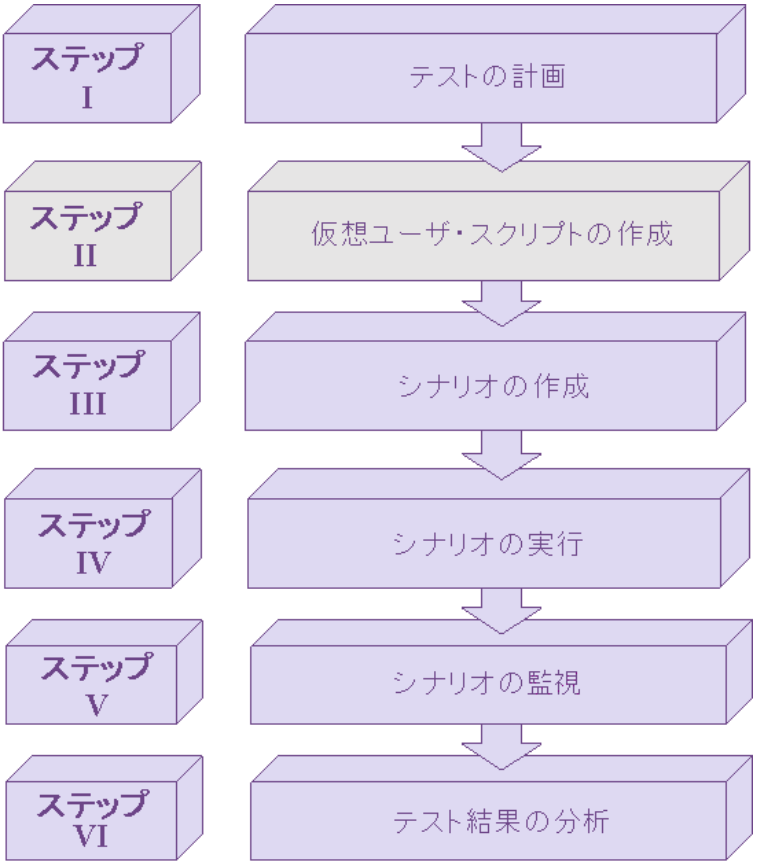
LoadRunner は人間のユーザを「**仮想的なユーザ**」，つまり「**仮想ユーザ**」に置き換えます。仮想ユーザは実際のユーザのアクションを，標準的なビジネス・プロセスを実行することでエミュレートします。仮想ユーザが実行する各アクションに対し，LoadRunner は入力をサーバまたは同様のエンタープライズ・システムに送信します。仮想ユーザの数を増やすことで，システムへの負荷を増やすことができます。ワークステーションを操作できるユーザは一度に 1 人だけですが，多数の仮想ユーザであれば 1 台のワークステーションで一度に多数を実行することができます。

ユーザ負荷の高い状態をエミュレートするために，一連のタスクを実行する多数の仮想ユーザを作成します。たとえば 銀行の ATM から現金を引き出す 100 の仮想ユーザを同時に実行して，サーバの動作を観察することができます。

LoadRunner では，クライアント / サーバのパフォーマンス・テスト要件を，いくつかのシナリオに分割できます。シナリオは，各テスト・セッション中に発生する出来事を規定します。たとえば，「シナリオ」は，エミュレートするユーザの数，それらが実行するアクション，エミュレーションを実行するマシンを規定し，制御します。

LoadRunner には，さまざまなタイプの仮想ユーザがあり，それぞれが特定の負荷テスト環境に対応しています。これにより，仮想ユーザを使って現実的な状況をモデルにした正確なエミュレーションを実行できます。シナリオ実行時に仮想ユーザが実行する動作は，「**仮想ユーザ・スクリプト**」に記述されています。仮想ユーザ・スクリプトには，シナリオ実行時にサーバのパフォーマンスを測定して記録する関数が含まれます。仮想ユーザのタイプごとに，対応するタイプの仮想ユーザ・スクリプトが必要です。シナリオに必要な仮想ユーザ・スクリプトの作成は LoadRunner のテスト・プロセスの一部です。

次の図は，6 段階の LoadRunner のテスト・プロセスを示します。本書では，ステップ II「仮想ユーザ・スクリプトの作成」について説明します。ほかのステップの詳細については，『**LoadRunner コントローラ・ユーザーズ・ガイド**』を参照してください。


ステップ I
テストの計画
ステップ II
仮想ユーザ・スクリプトの作成
ステップ III
シナリオの作成
ステップ IV
シナリオの実行
ステップ V
シナリオの監視
ステップ VI
テスト結果の分析

仮想ユーザのタイプ

LoadRunner はさまざまな仮想ユーザ・テクノロジを備えており，クライアント / サーバ・アーキテクチャのタイプに応じたサーバ負荷を生成できます。それぞれの仮想ユーザ・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が作成されます。たとえば，Web ブラウザを操作するユーザをエミュレートする場合は Web 仮想ユーザを使い，Tuxedo アプリケーション・サーバと通信する Tuxedo クライアントをエミュレートする場合は Tuxedo 仮想ユーザを使います。それぞれのタイプの仮想ユーザ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・シナリオを作成できます。

仮想ユーザのタイプは次のカテゴリに分類されます。

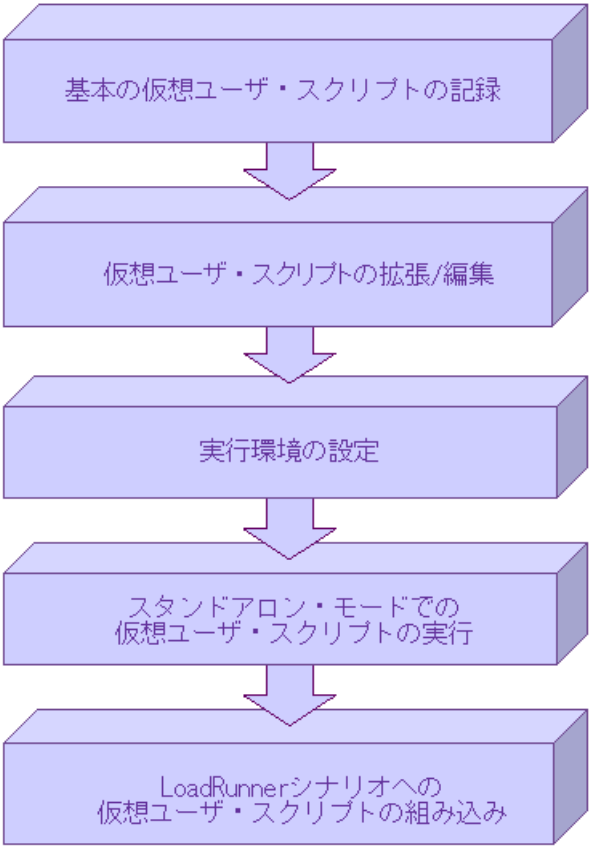
- **Application Deployment ソリューション**：Citrix ICA
- **クライアント / サーバ**：DB2 CLI，DNS，Informix，MS SQL Server，ODBC，Oracle（2-Tier），Sybase CTlib，Sybase DBlib，Windows Sockets プロトコル用
- **ユーザ定義**：C テンプレート，Visual Basic テンプレート，Java テンプレート，Javascript，VBScript タイプ・スクリプト用
- **分散コンポーネント**：COM/DCOM，Corba-Java，Rmi-Java プロトコル用
- **e ビジネス**：FTP，LDAP，Palm，PeopleSoft 8 mulit-lingual，SOAP，Web（HTTP/HTML），Web/WinSocket Dual Protocol 用
- **Enterprise Java Beans**：EJB テストおよび Rmi-Java プロトコル用
- **ERP/CRM**：Baan，Oracle NCA，Peoplesoft-Tuxedo，SAP-Web，SAPGUI, Siebel-DB2 CLI，Siebel-MSSQL，Siebel-Web，Siebel-Oracle プロトコル用
- **メール・サービス**：Internet Messaging (IMAP)，MS Exchange (MAPI)，POP3，SMTP
- **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
- **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］リスト・ボックスで［**All Protocols**］（すべてのプロトコル）を選択します。

仮想ユーザ・スクリプトの作成

仮想ユーザ・スクリプトの構造と内容は，仮想ユーザのタイプごとに異なります。たとえば，データベース仮想ユーザには必ず 3 つのセクションがあり，C 言語に似たコードで書かれ，データベース・サーバへの SQL 呼び出しが含まれています。対照的に，GUI 仮想ユーザ・スクリプトには 1 つのセクションしかなく，TSL（テスト・スクリプト言語）で書かれます。

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・スクリプトの作成は，基本のスクリプトを記録するところから始まります。LoadRunner では，仮想ユーザ・スクリプトを記録するためのいくつかのツールを利用できます。ツールの一覧については次の表を参照してください。基本のスクリプトを拡張するには，制御フロー構造を始め，他の LoadRunner API をスクリプトに追加します。スクリプトが完成したら実行環境を設定しま

す。実行環境の設定には，反復，ログ，タイミング情報の設定が含まれ，仮想ユーザ・スクリプトを実行するときの仮想ユーザの動作を定義します。スクリプトが正しく動作することを確認するには，スタンドアロン・モードで実行します。スクリプトを正しく実行できたら，LoadRunner のシナリオにスクリプトを組み込みます。

仮想ユーザ・スクリプトの作成時には，次の LoadRunner ツールを使用します。	
VuGen	仮想ユーザ・スクリプトを作成するための LoadRunner の主要なツールです。**VuGen** とも呼ばれる仮想ユーザ・スクリプト・ジェネレータは，さまざまな仮想ユーザ・スクリプトを作成できる Windows ベースのアプリケーションです。VuGen ではスクリプトの記録だけでなく，実行も可能です。VuGen によって生成されたスクリプトの多くは，Windows と UNIX の両方のプラットフォームで実行できます。
WinRunner	マーキュリー・インタラクティブが提供する Windows ベース GUI アプリケーションを対象としたテスト自動化ツールです。WinRunner で生成したテスト・スクリプトを拡張することによって，Windows プラットフォームで実行できる GUI 仮想ユーザ・スクリプトを作成できます。
QuickTest Professional	複雑な Web 環境（Java アプレット，Flash などを含む）をテストする，マーキュリー・インタラクティブのアイコン・ベースのツールです。LoadRunner では，QuickTest Professional を使って作成した仮想ユーザ・スクリプトを実行できます。本製品は別途購入が必要です。

本書の使用法

本書は，次の部で構成されています。

- 第 1 部「仮想ユーザの紹介」は，すべてのタイプの仮想ユーザ・スクリプトを対象とします。
- 第 2 部「VuGen を使った作業」は VuGen を使って記録，実行する仮想ユーザ・スクリプトだけを対象とします。第 2 部は，GUI 仮想ユーザ・スクリプトを作成する場合は対象外となります。
- 第 3 部から第 15 部は，個別のタイプの仮想ユーザ・スクリプトが対象となります。各仮想ユーザ・タイプの説明箇所については，目次を参照してください。
- 第 16 部には，上級ユーザのための情報が含まれています。デバッグのための一般的なヒント，VuGen によって生成されたファイルの説明，Visual C および Visual Basic でのスクリプトのプログラミング方法などを紹介しています。
- 第 17 部には，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。Java 環境，EJB のアーキテクチャ，外部関数の呼び出し，UNIX 環境でのプログラミング，キーボード・ショートカットについて説明します。

注 : GUI 仮想ユーザ・スクリプトの作成方法の詳細については，『**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 を使った仮想ユーザ・スクリプトの記録
- VuGen の起動
- VuGen 環境オプションについて
- 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する LoadRunner の主要ツールの 1 つです。

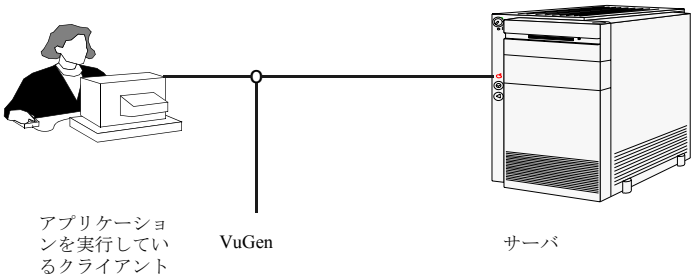
VuGen では，仮想ユーザ・スクリプトの記録だけでなく実行も可能です VuGen でのスクリプトの実行はデバッグ時に役立ちます。スクリプトを実行することによって，仮想ユーザ・スクリプトが負荷テスト・シナリオの一部としてどのように動作するかを確認できます。

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 を使った仮想ユーザ・スクリプトの記録

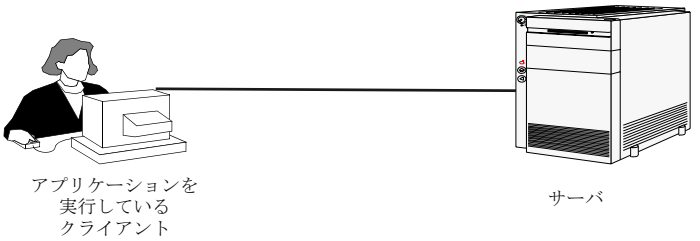
VuGen では，クライアント・アプリケーションで典型的なビジネス・プロセスを実行しているユーザの操作を記録することによって，仮想ユーザ・スクリプトを作成します。VuGen はクライアントとサーバの間の動作状況を記録してスクリプトを作成します。たとえば，データベース・アプリケーションでは，VuGen はデータベースのクライアント側を監視して，データベース・サーバとの間で送受信されるすべてのリクエストを追跡します。



アプリケーションによるサーバへの API 関数呼び出しを手作業でプログラミングして，仮想ユーザ ・スクリプトを作成する方法の代わりに，VuGen では次のことができます。

- アプリケーションとサーバの間の通信の監視
- 必要な関数呼び出しの生成
- 生成された関数呼び出しの仮想ユーザ・スクリプトへの挿入

VuGen で作成した仮想ユーザ・スクリプトは，サーバ API への呼び出しを実行することによって直接サーバと通信できます。クライアント・ソフトウェアに依存しません。このため，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。



さらに，仮想ユーザがサーバと直接通信する場合は，システム・リソースがユーザ・インタフェースで消費されません。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ます。つまり，数台のテスト用マシンで，サーバに対する大きな負荷をエミュレートできます。

VuGen の起動

［スタート］メニューから VuGen を起動します。［**スタート**］＞［**プログラム**］＞［**LoadRunner**］＞［**Virtual User Generator**］を選択します。VuGen を起動すると，起動画面が表示されます。この画面が開かないようにするには，［仮想ユーザ ジェネレータへようこそ］ダイアログ・ボックスで［**今後起動ダイアログを表示しない**］を選択します。

VuGen の起動時に［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かどうかを切り替えるには，［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。ダイアログ・ボックスの一番下のセクションで，［起動ダイアログを表示する］オプションを選択するか，クリアします。

［仮想ユーザ ジェネレータへようこそ］ダイアログ・ボックスには，以下のショートカットが含まれます。

［**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設定のオプションです。すべてのプロトコルを表示したり，使用可能なプロトコルをカテゴリ別に表示したりすることができます。

［**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 に使用可能なすべてのプロトコルが表示され，記録するプロトコルを指定できます。

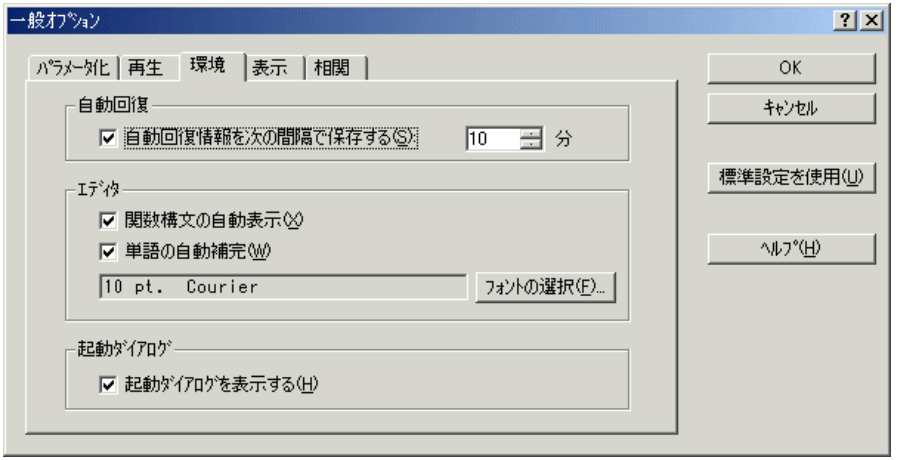
［**最近使用したプロトコルの新規スクリプト**］：新規仮想ユーザ・スクリプトを作成するのに最近使用したプロトコルの一覧を表示します。

［**スクリプトを開く**］：ファイル・システム，ネットワークまたは TestDirector データベースにすでにあるスクリプトを開きます。

［**最近使用したスクリプト**］：VuGen で最近開いたスクリプトの一覧を表示します。

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仮想ユーザ・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，保存する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。［フォント］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。

起動ダイアログ

VuGen を起動したときに［仮想ユーザ ジェネレータへようこそ］ダイアログ・ボックスを開きます。［仮想ユーザ ジェネレータへようこそ］ダイアログ・ボックスには，新規スクリプトを作成したり，既存のスクリプトを開いたり，最近開いたスクリプトを表示したりすることが簡単に行えるリンクが含まれます。このオプションを無効にすると，VuGen で空の画面が開きます。

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。自動補完は，10 秒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の設定は，次の手順で行います。

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 現在の仮想ユーザ・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。［フォント］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。

VuGen を使った仮想ユーザ・スクリプトの実行

仮想ユーザ・スクリプトを使用して負荷テストを実行するためには，スクリプトを LoadRunner のシナリオ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査できます。スクリプトの実行が成功したら，シナリオに組み込むことができます。LoadRunner のシナリオの詳細については，『**LoadRunner コントローラ・ユーザーズ・ガイド**』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 9 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何らかの構文エラーがあると，インタプリタによってエラーが検出されます。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 C 「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 6 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
- プロトコル固有仮想ユーザ関数

「**一般**」仮想ユーザ関数と「**プロトコル固有**」関数の組み合わせが LoadRunner API を形成し，仮想ユーザがサーバと直接通信できるようにします。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。たとえば，**lr_start_transaction**（Java では **lr.start_transaction**）関数はトランザクションの開始を，**lr_end_transaction**（Java では **lr.end_transaction**）関数はトランザクションの終了を示します。詳細については，第 6 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 関数のリストは，22 ページ「C 仮想ユーザ関数の使用方法」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，たとえばデータベース・スクリプトには LRD 関数を，TUXEDO スクリプトには LRT 関数を，Windows Sockets スクリプトには LRS 関数を挿入します。

標準では，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に C 言語（Corba-Java/Rmi-Java 仮想ユーザの場合は 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 4 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 / 生成され，スクリプトに挿入される関数の例です。

```
#include "as_web.h"


Action1()
{

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

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

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

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

	lr_start_transaction("Purchase_Order");
...
```

仮想ユーザ・スクリプトでの C 関数の使用方法の詳細については，第 6 章 「仮想ユーザ・スクリプトの拡張」を，Java スクリプトの変更の詳細については，第 24 章 「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 6 章 「仮想ユーザ・スクリプトの拡張」を参照してください。

次のリストに，ANSI C スクリプト用の一般 LoadRunner 関数を示します。Java と GUI を除くすべてのプロトコルが含まれています。Java の LoadRunner 関数のリストについては，第 24 章 「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。

トランザクション関数

lr_end_sub_transaction	パフォーマンス分析で使用するサブトランザクションの終わりを示します。
lr_end_transaction	LoadRunner トランザクションの終わりを示します。
lr_end_transaction_instance	パフォーマンス分析で使用するトランザクション・インスタンスの終わりを示します。
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 に設定して，エラー・メッセージを送信します。
lr_get_trans_instance_duration	トランザクション・インスタンスをハンドルで指定し，その経過時間を取得します。
lr_get_trans_instance_wasted_time	トランザクション・インスタンスをハンドルで指定し，その浪費時間を取得します。
lr_get_transaction_duration	トランザクションを名前で指定し，その経過時間を取得します。
lr_get_transaction_think_time	トランザクションを名前で指定し，その思考遅延時間を取得します。
lr_get_transaction_wasted_time	トランザクションを名前で指定し，その浪費時間を取得します。
lr_resume_transaction	パフォーマンス分析で使用するトランザクション・データの収集を再開します。

lr_resume_transaction_instance	パフォーマンス分析で使用するトランザクション・インスタンス・データの収集を再開します。
lr_set_transaction_instance_status	トランザクション・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始まりを示します。
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・ラインで使われている **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・ラインで使われている **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。
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 コントローラを実行しているマシンの名前を返します。

文字列関数

lr_eval_string	パラメータをその現在の値で置き換えます。
lr_save_string	NULL 終端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	現在の日付と時刻をパラメータに保存します。
lr _advance_param	使用可能な次のパラメータに進みます。
lr _decrypt	暗号化された文字列を復号します。
lr_eval_string_ext	パラメータ・データを格納しているバッファを指すポインタを取得します。
lr_eval_string_ext_free	**lr_eval_string_ext** 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特定の出現を検索し，当該文字列を基準とする相対的な指定範囲のバッファの一部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］に送ります。
lr_error_message	エラー・メッセージを［出力ウィンドウ］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］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，コントローラの仮想ユーザ・ステータス領域に出力します。
lr_message	メッセージを仮想ユーザ・ログと［出力ウィンドウ］に送ります。

ランタイム関数

lr_load_dll	外部の 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	エラー処理メソッドを指定します。
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。

関数のヘルプ

LoadRunner 関数に関する情報を得る方法はいくつかあります。

- LoadRunner オンライン関数リファレンス
- IntelliSense
- ヘッダー・ファイル

LoadRunner オンライン関数リファレンス

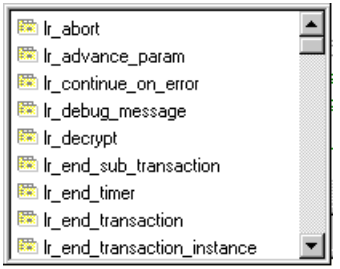
「**オンライン関数リファレンス**」には，すべての LoadRunner 関数の詳しい構文情報が記載されています。関数の使用例も含まれています。関数名の検索が可能なほか，カテゴリまたはアルファベット順のリストから参照できます。

「**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。

IntelliSense

VuGen エディタには，**単語補完機能**とも呼ばれる **IntelliSense** が採用されています。関数のタイプ入力を始めると，IntelliSense 機能によってその関数の接頭辞に一致する使用可能なすべての関数を示すリスト・ボックスが開きます。



関数を使用するには，使用する関数を選択します。VuGen によって関数がカーソルの位置に置かれます。リスト・ボックスを閉じるには，Esc キーを押します。

VuGen 全体でこの機能を有効にするには，［**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**］チェック・ボックスを選択します。最初のアンダーバーを入力したときにリスト・ボックスが開きます。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**］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。

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，プロトタイプが自動的に表示されます。

VuGen 全体でこの機能を有効にするには，［**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・ボックスを選択します。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，LoadRunner 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。たとえば，**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**
MediaPlayer	**mic_media.h**
Oracle NCA	**orafuncs.h**
POP3	**mic_pop3.h**
RealPlayer	**lreal.h**
SAPGUI	**as_sap.gui.h**
Siebel	**lrdsiebel.h**
SMTP	**mic_smtp.h**
端末エミュレータ	**lrrte.h**
Tuxedo	**lrt.h**
Web	**as_web.h**
Windows Sockets	**lrs.h**

第 3 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。

- 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
- アクションのインポート
- 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 つまたは複数の **Actions**，および **vuser_end** です。記録中に，VuGen が記録する関数を挿入す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。

テストは作成した後 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 6 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 つまたは複数の **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
Actions	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 9 章「実行環境の設定」を参照してください。

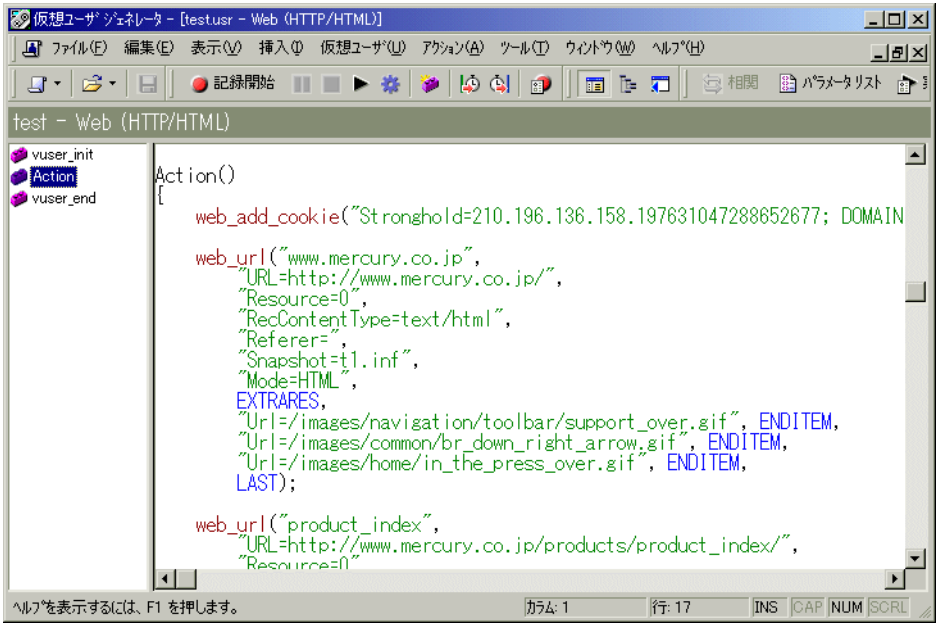
各スクリプト・セクションの内容の表示と編集には，VuGen スクリプト・エディタを使用します。一度に表示できるのは，1 つのセクションの内容だけです。セクションを表示するには，左側の表示枠でセクションの名前を選択して強調表示します。

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ほかのプロトコルの場合に作成されるスクリプトの各セクションに対応します。つまり，初期化ルーチンは **init** メソッドに，クライアントの動作は **action** メソッドに，ログオフの手順は

end メソッドに挿入されます。詳細については，第 24 章「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;}
}
```

次の例は，VuGen スクリプト・エディタに Web 仮想ユーザ・スクリプトの **Action1** セクションが表示されている様子を示します。

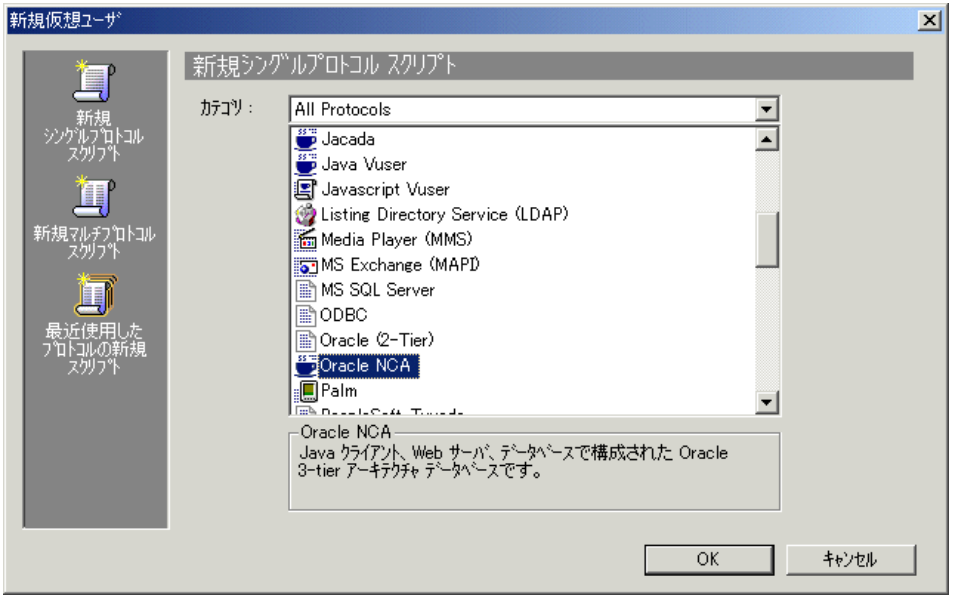


仮想ユーザ・スクリプトの新規作成

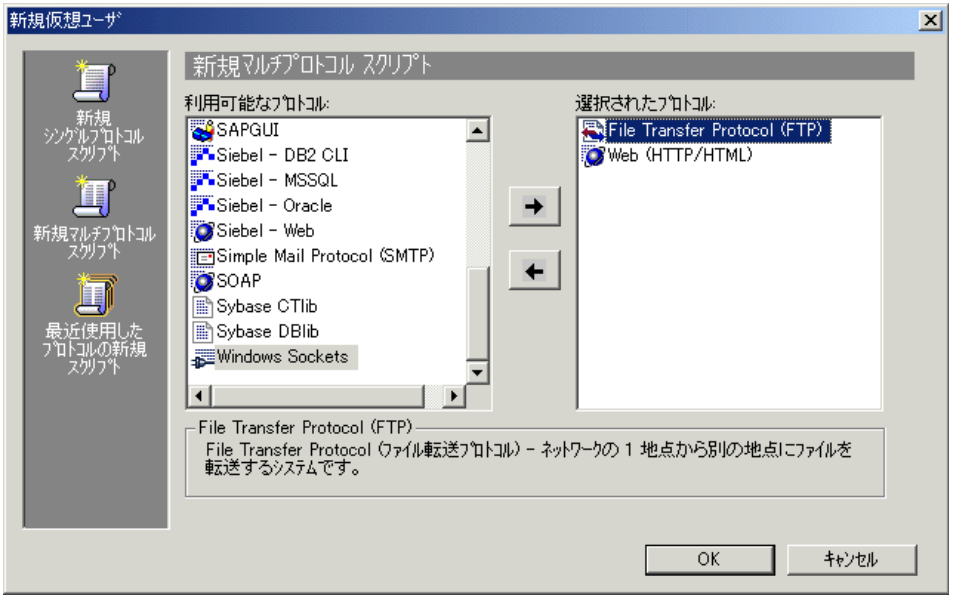
VuGen では，シングル・プロトコル・モードまたはマルチ・プロトコル・モードで記録することによって，スクリプトを新規作成できます。

［**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。

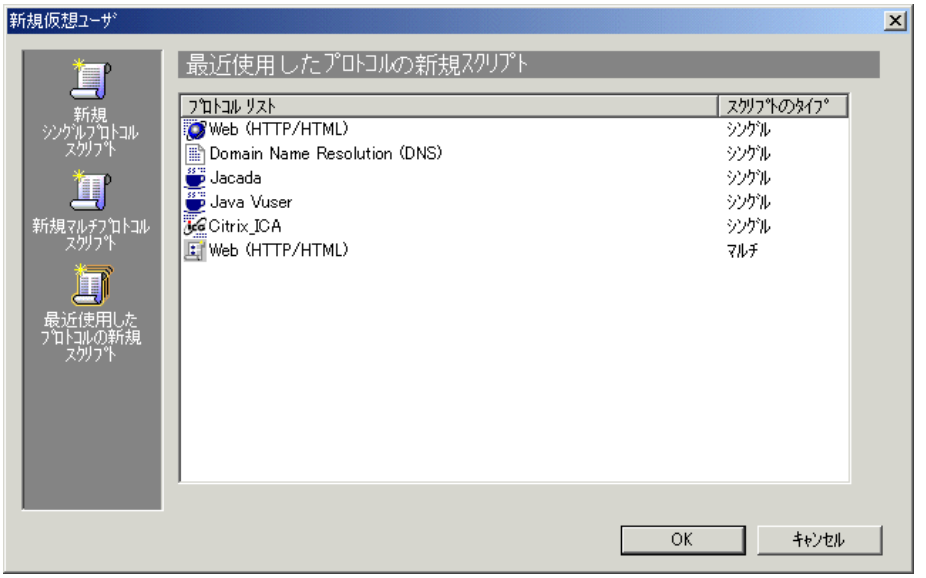
［**新規シングルプロトコル スクリプト**］：シングル・プロトコルの仮想ユーザ・スクリプトを作成します。これは［仮想ユーザ ジェネレータへようこそ］ダイアログ・ボックスが開いたときの標準のオプションです。シングル・プロトコルのスクリプトを作成するには，ウィンドウの右側で「カテゴリ」を選び，プロトコルを選択します。



［**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，および Web。**Web/Winsocket Dual Protocol** 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。このプロトコルをほかのマルチ・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，および 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。

LoadRunner はさまざまなプロトコルをサポートしているため，次の記録手順のいくつかは，特定のプロトコルにしか適用されません。

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 13 章「Java 言語仮想ユーザ・スクリプトの記録」，あるいは各プロトコルを解説している章を参照してください。

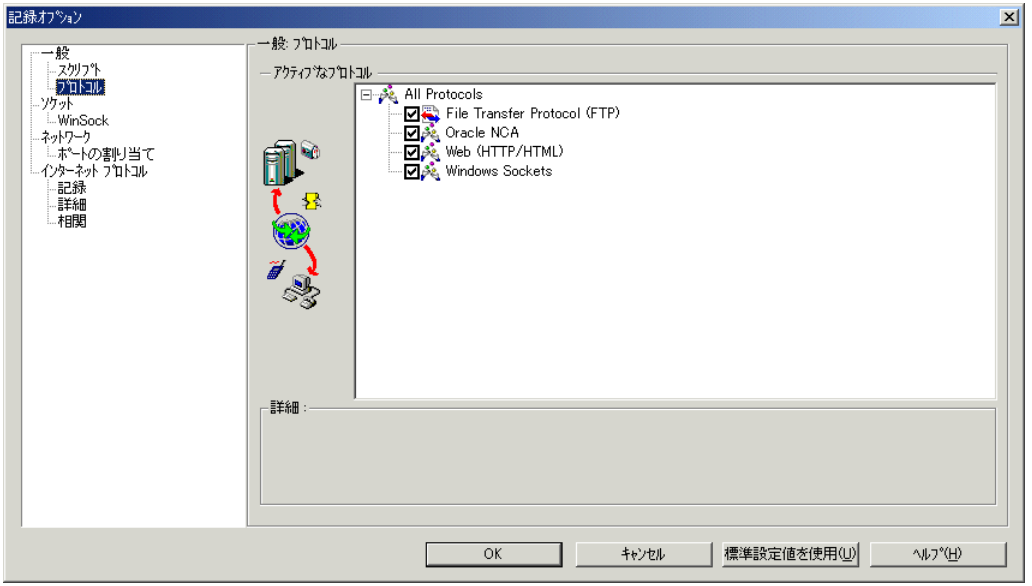
プロトコルの追加と削除

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**］の［**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。

- ［**All Protocols**］：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です。
- ［**Application Deployment ソリューション**］：Citrix プロトコルに対応します。
- ［**クライアント・サーバ**］：MS SQL，ODBC，Oracle（2-tier），DB2 CLI，Sybase CTlib，Sybase DBlib，Windows Sockets および DNS の各プロトコルに対応します。
- ［**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 および VBscript タイプの各スクリプトに対応します。
- ［**分散コンポーネント**］：COM/DCOM，Corba-Java，および Rmi-Java の各プロトコルに対応します。
- ［**e ビジネス**］：FTP，LDAP，Palm，SOAP，Web（HTTP/HTML），およびデュアル Web/Winsocket の各プロトコルに対応します。

- ［**Enterprise Java Beans (EJB)**］：EJB テストおよび Rmi-Java の各プロトコルに対応します。
- ［**ERP/CRM**］：Baan，Oracle NCA，Peoplesoft-Tuxedo，Peoplesoft 8 Web マルチリンガル，SAPGUI，SAP-Web，および Siebel（Siebel-DB2CLI，Siebel-MSSQL，Siebel-Web，および Siebel-Oracle）の各プロトコルに対応します。
- ［**レガシ**］：ターミナル・エミュレーション（RTE）に対応します。
- ［**メール サービス**］：Internet Messaging（IMAP），MS Exchange（MAPI），POP3，および SMTP に対応します。
- ［**ミドルウェア**］：Jacada および Tuxedo（6, 7）プロトコルに対応します。
- ［**ストリーミング**］：MediaPlayer および RealPlayer の各プロトコルに対応します。
- ［**ワイヤレス**］：i モード，VoiceXML，および WAP の各プロトコルに対応します。

スクリプトの新規作成

新しい仮想ユーザ・スクリプトの作成は，次の手順で行います。

1 ［**スタート**］＞［**プログラム**］＞［**LoadRunner**］＞［**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 47 章「Oracle NCA 仮想ユーザ・スクリプトの作成」を参照してください。SOAP の場合は，［新規シングルプロトコル スクリプト］リストから Web/WinSock Dual Protocol を選択します。

4 次回 VuGen を開くときにこの［仮想ユーザ ジェネレータへようこそ］ウィンドウが表示されないようにするには，［**今後起動ダイアログを表示しない**］チェック・ボックスを選択します。再度［仮想ユーザ ジェネレータへようこそ］ウィンドウを有効にするには，［**ツール**］＞［**一般オプション**］を選択し，［環境］タブの［**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。

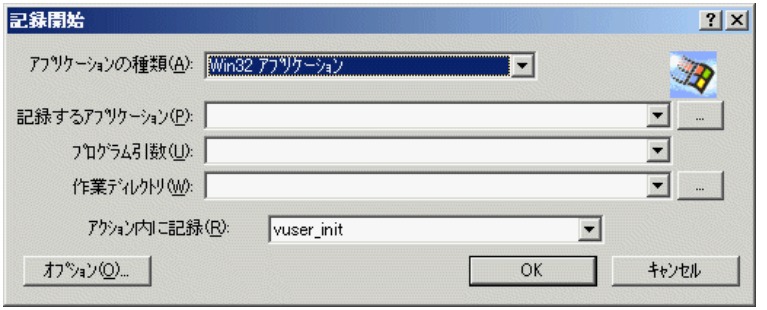
5 ［**OK**］をクリックしてダイアログ・ボックスを閉じ，仮想ユーザ・スクリプトの生成を開始します。

スクリプトの記録

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。

1 ［記録開始］ダイアログ・ボックスが表示されない場合は，［**記録開始**］ボタンをクリックし，［記録開始］ダイアログ・ボックスを表示します。このダイアログ・ボックスは，記録しているプロトコルによって異なります。

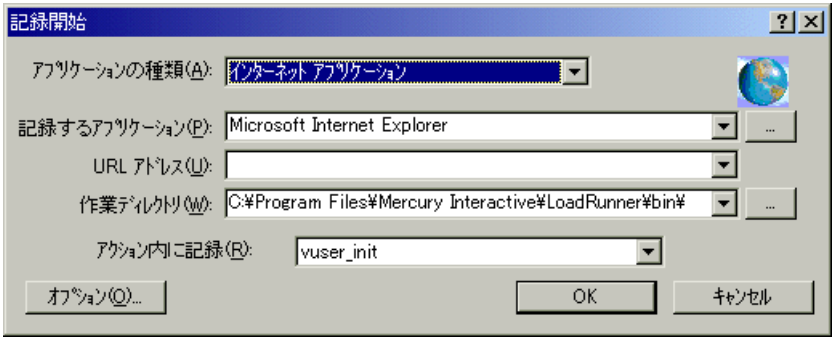
2 ほとんどのクライアント / サーバ・プロトコルの場合，次のダイアログ・ボックスが表示されます。



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。たとえば，Web および Oracle NCA スクリプトはインターネット・アプリケーションを記録し，Windows Sockets 仮想ユーザは Win32 アプリケーションを記録します。

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。

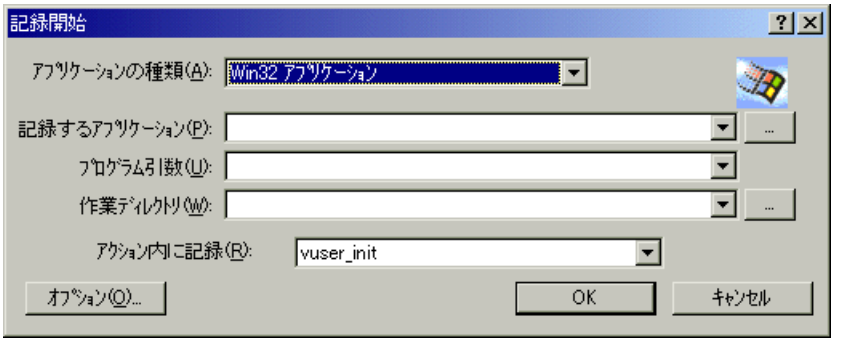


［**記録するアプリケーション**］：記録するブラウザまたはインターネット・アプリケーションを選択します。Citrix の場合は，bin ディレクトリで **rundlg.exe** ファイルを探します。

［**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。たとえば，**plus32.exe** にコマンド・ライン・オプション peter@neptune を指定した場合，**plus32.exe** を起動すると，ユーザ **Peter** がサーバ **Neptune** に接続されます。

［**作業ディレクトリ**］：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**］＞［**新規アクションの作成**］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。

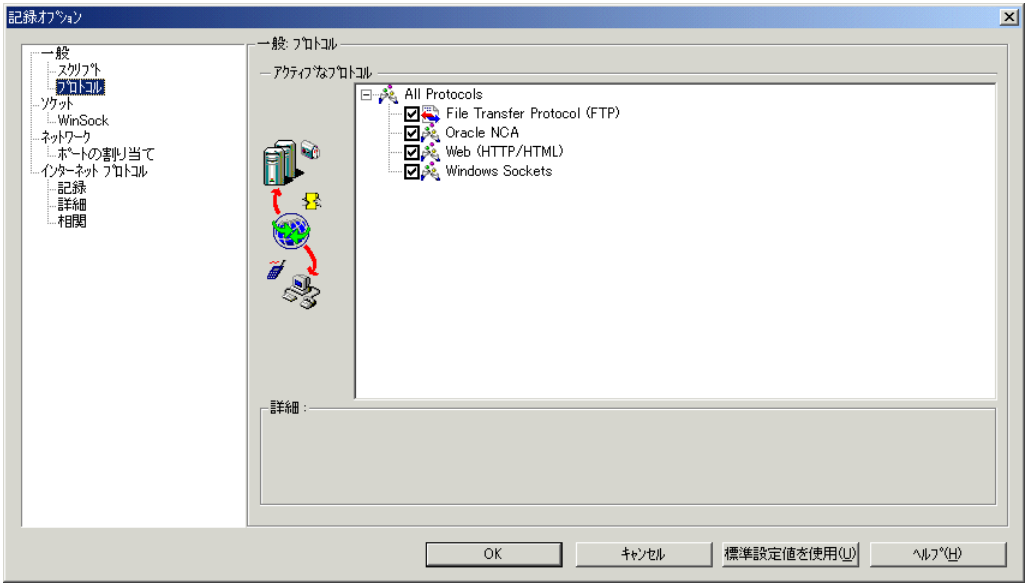
- アプリケーションで特定の位置まで移動し，その位置から記録を開始する場合。
- 既存のスクリプトに記録する場合。

8 ［**オプション**］または［**記録オプション**］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 4 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。

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 5 章「ポートの割り当て設定」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・ツールバーが表示されます。

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。

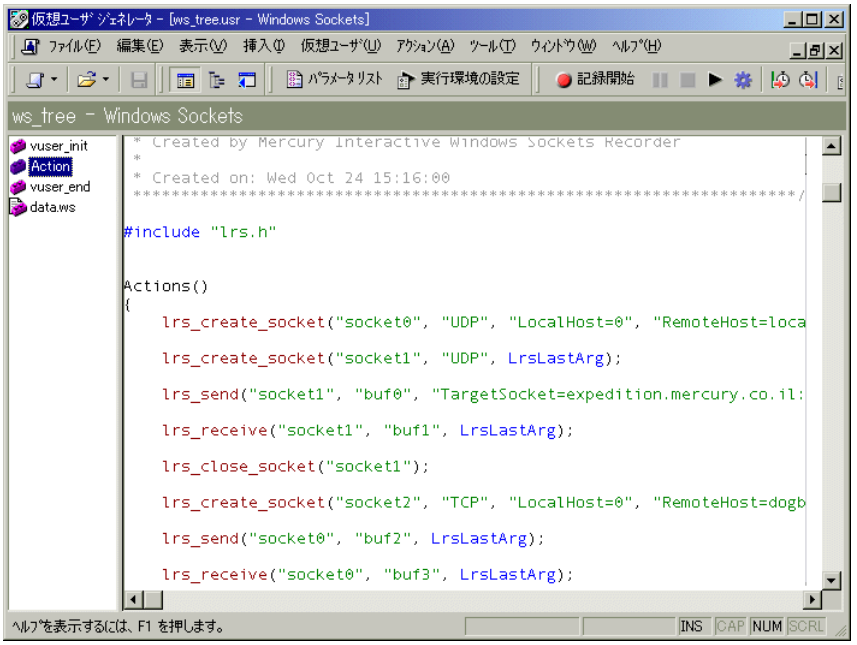
記録セッションの終了

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。

記録の終了は，次の手順で行います。

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。記録されたすべてのステートメントが VuGen エディタに表示されます。



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 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 して電子メールで送信**］を選択します。

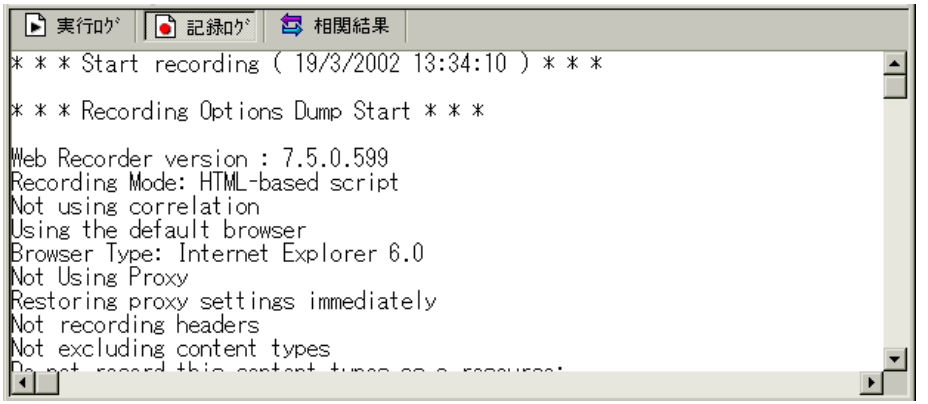
［**OK**］をクリックします。電子メールの作成フォームが表示されます。

電子メール・アドレスを入力してメールを送信します。

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。

記録中に発行されたメッセージのログを表示するには，［**表示**］＞［**出力ウィンドウ**］を選択し，［**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。



VuGen は記録中に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と設定情報が含まれます。VuGen は，これらのファイルをスクリプトとともに保存します。

スクリプトは，VuGen エディタで手作業で編集できます。マルチアクションをサポートするプロトコルの場合，いつでも追加アクションを記録できます。

アクションのインポート

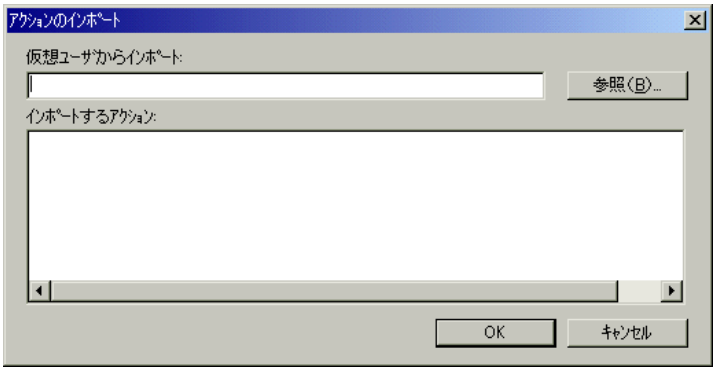
複数のアクションをサポートする仮想ユーザ・タイプの場合，別の仮想ユーザ・スクリプトから現在のスクリプトにアクションをインポートできます。インポートできるのは，同じタイプの仮想ユーザのアクションだけです。インポートされたアクションに関連付けられているパラメータは，スクリプトに組み込まれます。以下のオプションが使用できます。

［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のアクションのインポートは，次の手順で行います。

1 ［**アクション**］＞［**仮想ユーザにアクションをインポート**］を選択します。［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 9 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 6 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 7 章「パラメータの定義」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。

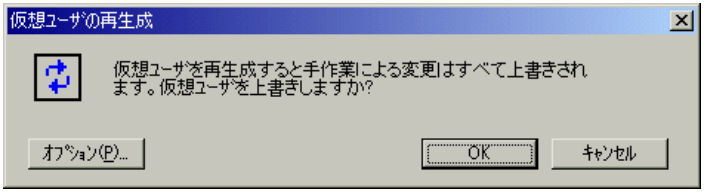
以下のボタンが［仮想ユーザの再生成］ダイアログ・ボックスから使用できます。

［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・スクリプトの場合は，次の手順で行います。

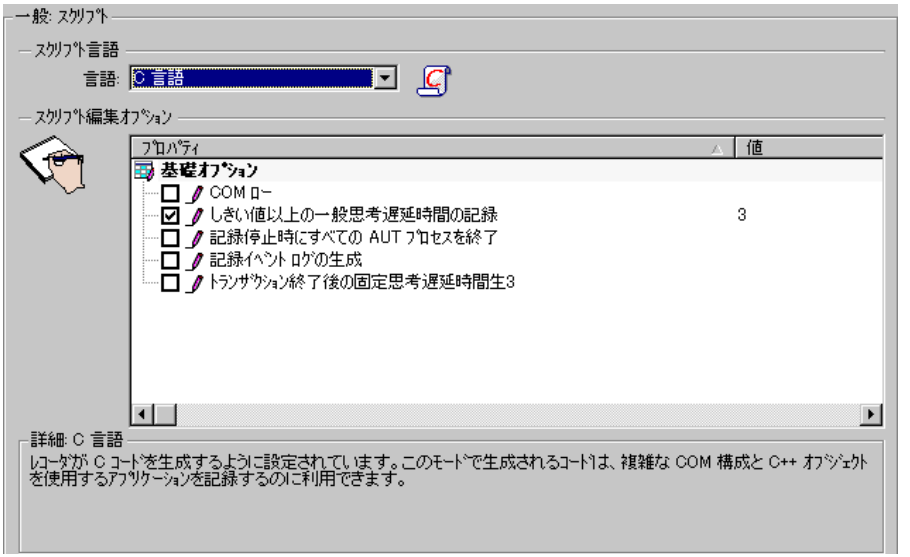
1 ［**ツール**］＞［**仮想ユーザを再生成**］を選択します。手作業で行ったすべての変更が上書きされることを示す警告が表示されます。



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 4 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，**スクリプト**・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，**スクリプト**・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 for Applications**］：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。

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

［**Java Scripting**］：Javascript ベースのアプリケーション用（**js** ファイルや動的 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，51 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，52 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，52 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。

［**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）。

［**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）。

［**トランザクション終了後の固定思考遅延時間生成**］：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）。

［**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）。

［**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）。

［**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）。

［**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）。

［**COM ローカル サーバとして作成されたプロセスを記録**］：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（標準設定では無効）。

［**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）。

［**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，52 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，52 ページ「記録オプションの設定」を参照してください。

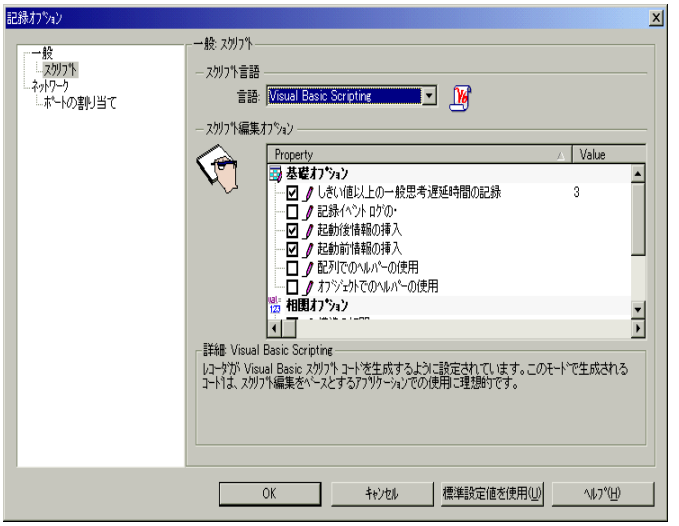
記録オプションの設定

［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の設定は，次の手順で行います。

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 言語**］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 5 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。たとえば，**twilight** というサーバのポート 25 番からのトラフィックをすべて SMTP として扱うよう指定できます。また，**viper** というサーバからのすべてのトラフィックを，ポートに関係なく FTP プロトコルへ割り当てることもできます。

さらには，サーバ名に関係なく，ポート 23 のすべてのトラフィックを SMTP に割り当てることも可能です。

マルチ・プロトコル・モードで記録する場合には，少なくとも 1 つのプロトコルがソケット・レベルで記録していると，［**ポートの割り当て**］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。たとえば，FTP の割り当てだけを表示したい場合には［FTP］を選択します。

［**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。58 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。60 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。たとえば，**twilight** という

サーバのポート番号 25 番のトラフィックを SMTP として扱うよう指定した後で，すべてのサーバのポート 25 番を HTTP として扱うよう指定しても，データは SMTP として扱われます。

さらに，次のガイドラインが適用されます。

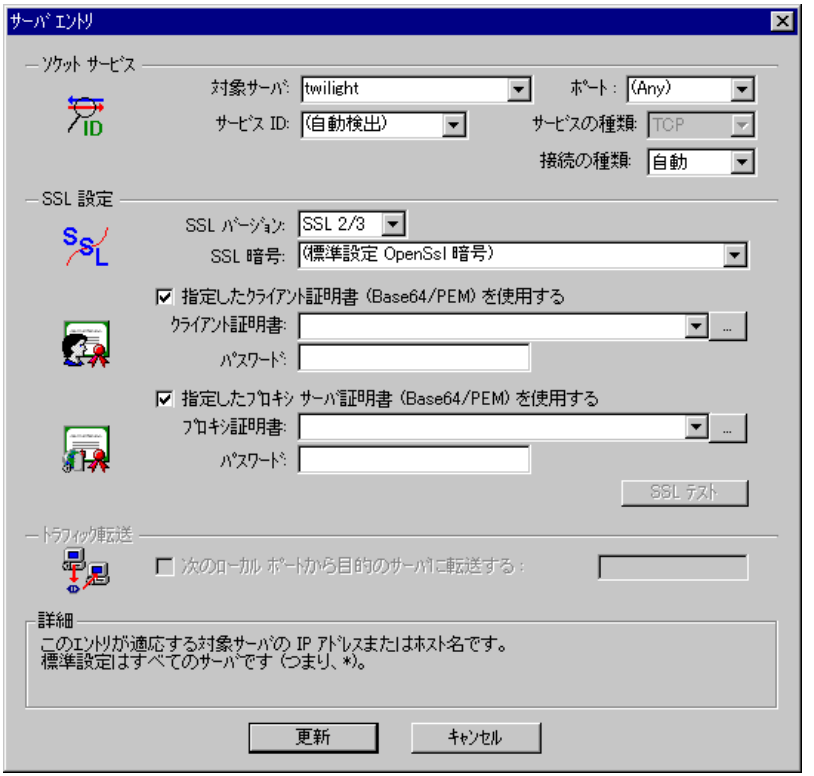
- **ポート 0**：ポート番号 0 は任意のポートを表します。
- **強制割り当て**：ポート番号，サーバ名，または「サーバ：ポート」の組み合わせの割り当てを指定した場合，VuGen では，ネットワーク・トラフィックがそのサービスを使用するよう強制されます。たとえば，「＜任意のサーバ＞」のポート 80 番が FTP を使用するよう指定した場合，VuGen では，実際の通信が HTTP であったとしても，FTP プロトコルを使用してその通信が記録されます。この例では，仮想ユーザ・スクリプトは空となる可能性があります。

  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，62 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。

トラフィックの転送

［**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。

たとえば，**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，62 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。たとえば，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，62 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

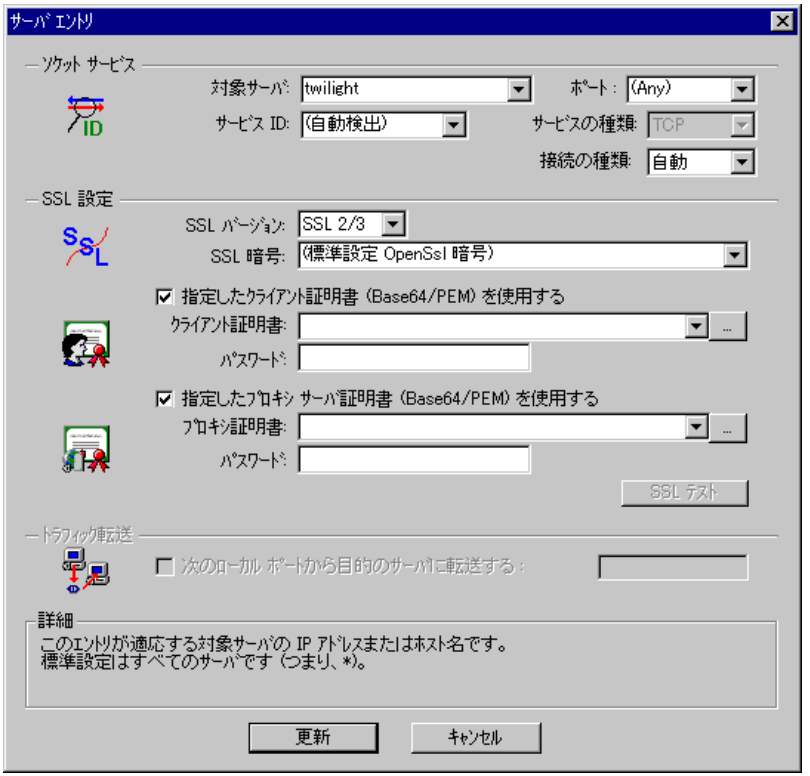
［記録オプション］ダイアログ・ボックスは，次のいくつかの方法で開けます。

- ツールバー・ボタン： 
- 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，次の手順で行います。

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 ［**対象サーバ**］，［**ポート**］，［**サービス ID**］，および［**接続の種類**］のサーバ情報を入力します。

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。

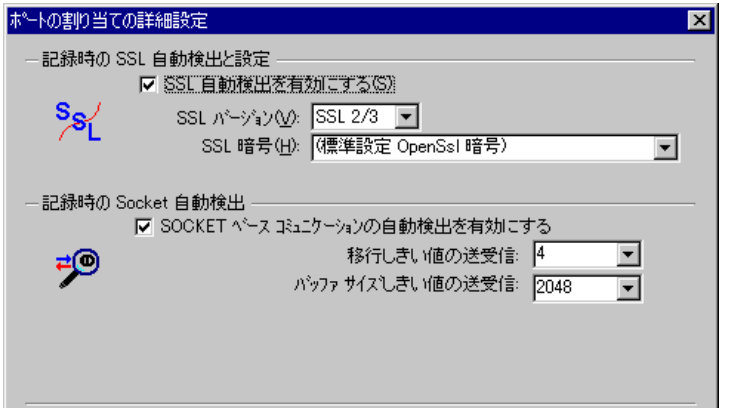
［**SSL バージョン**］，［**SSL 暗号**］を指定します。証明書を使用する場合には，［**指定したクライアント証明書（Base64/PEM）を使用する**］または［**指定したプロキシ サーバ証明書（Base64/PEM）を使用する**］を選択してユーザ情報を指定します。

［**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 ベースの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の割り当て**］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 6 章

仮想ユーザ・スクリプトの拡張

記録中や記録後に，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・スクリプトへのトランザクションの挿入
- 仮想ユーザ・スクリプトへのランデブー・ポイントの挿入
- 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
- ユーザ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
- 仮想ユーザ・スクリプトでの C 関数の使用方法

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，次のタイプの関数を追加することによって機能を拡張できます。

- 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。たとえば，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，シナリオ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。たとえば，Windows Sockets 仮想ユーザ・スクリプトでは **LRS** 関数，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，83 ページ「仮想ユーザ・スクリプトでの C 関数の使用方法」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の時間を測定します。これらの要求は，1 つのクエリーに対する応答を待つような簡単な処理の場合や，いくつかのクエリーを発行してレポートを作成するといった複雑な処理の場合があります。

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

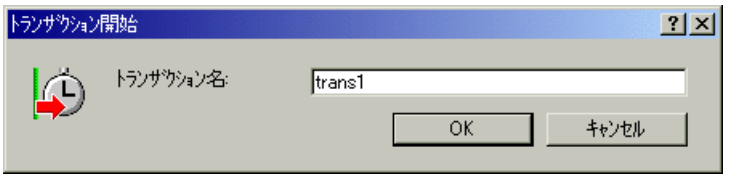
シナリオを実行すると，コントローラによって各トランザクションの実行にかかる時間が測定されます。シナリオの実行後，LoadRunner のグラフとレポートを使用して，トランザクションごとのサーバ・パフォーマンスを分析します。

トランザクションの開始を示す方法

1 つのビジネス・プロセスの開始をトランザクションとして示すことができます。

トランザクションの開始を示すには，次の手順で行います。

1 仮想ユーザ・スクリプトの記録中に，フローティング・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。

［**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。たとえば，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");
```

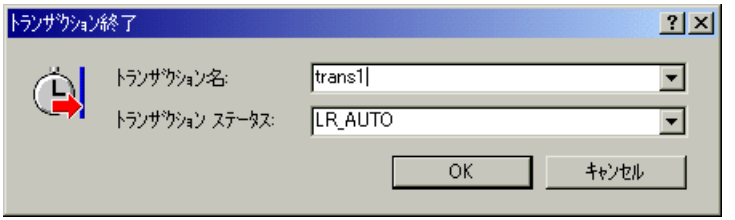
注：記録セッションを完了した後でスクリプトにトランザクションを挿入するには，VuGen のメニューから［**挿入**］＞［**トランザクション開始**］を選択してトランザクションの開始を示し，［**挿入**］＞［**トランザクション終了**］を選択してトランザクションの終了を示します。

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，次の手順で行います。

1 スクリプトの記録中に，［記録中です］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。たとえば，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);
```

仮想ユーザ・スクリプトへのランデブー・ポイントの挿入

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。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，その仮想ユーザが待機させられます。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，コントローラからの再開の許可を待ちます。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：ランデブーを **init** または **end** セクションには追加せずに，**Action** セクションにのみ追加することもできます。

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。

1 仮想ユーザ・スクリプト記録中に，フローティング・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**］をクリックしてランデブー名を確定します。VuGen は **lr_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。たとえば，次の関数は **rendezvous1** という名前のランデブー・ポイントを定義します。

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

注：記録セッションを完了した後でスクリプトにランデブー・ポイントを挿入するには，VuGen のメニューから［**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。たとえば，データベースの動作を記録している場合は，「これは最初のクエリーです」のように最初のクエリー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，次の手順で行います。

1 スクリプト記録中に，フローティング・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。次のスクリプトのセグメントに，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ーです
*/
```

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，仮想ユーザの情報を取得できます。

lr_get_attrib_string	仮想ユーザ ID やロード・ジェネレータの名前などのコマンド・ライン引数値またはランタイム情報を含む文字列を返します。
lr_get_host_name	当該仮想ユーザのロード・ジェネレータの名前を返します。
lr_get_master_host_name	LoadRunner コントローラの名前を返します。
lr_whoami	仮想ユーザの ID，グループ，シナリオ ID を返します。

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

シナリオを実行しているとき，コントローラの［出力］ウィンドウに，スクリプトの実行に関するメッセージが表示されます。エラーおよび通知メッセージをコントローラに送るためのステートメントを仮想ユーザ・スクリプトに挿入することができます。コントローラはこれらのメッセージを［出力］ウィンドウに表示します。たとえば，クライアント・アプリケーションの現在のステータスを示すメッセージを挿入することができます。また，これらのメッセージをファイルに保存することもできます。

注：トランザクション内からメッセージを送信しないでください。トランザクションの実行時間が伸びて，トランザクション結果が不正確になることがあります。

以下のメッセージ関数を仮想ユーザ・スクリプトで使用できます。

lr_debug_message	デバッグ・メッセージを［出力］ウィンドウに送信します。
lr_error_message	エラー・メッセージを［出力］ウィンドウに送信します。
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］ウィンドウに送信します。
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	形式を整えた出力を生成し，コントローラの仮想ユーザ・ステータス領域に出力します。
lr_message	メッセージを仮想ユーザ・ログと［出力］ウィンドウに送信します。

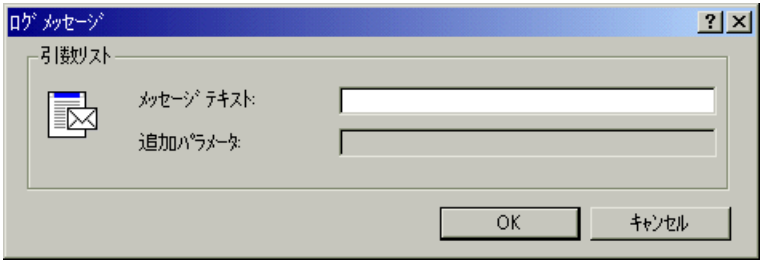
注：実行環境の設定を使ってスクリプトのデバッグ・レベルを変更しても，**lr_message**，**lr_output_message**，および **lr_log_message** の各関数の動作は変わらず，メッセージの送信が続けられます。

ログ・メッセージ

VuGen を使って **lr_log_message** 関数を生成して，仮想ユーザ・スクリプトに挿入できます。たとえば，データベースを対象としたアクションを記録しているときに，最初のクエリーを示すために「これは最初のクエリーです」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，次の手順で行います。

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。

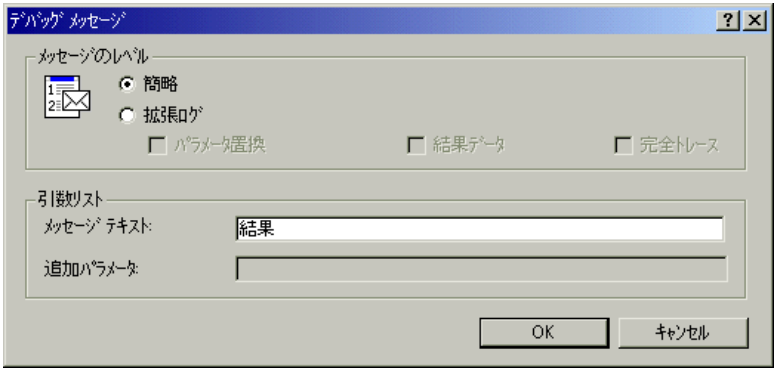
デバッグ・メッセージ

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，次の手順で行います。

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。

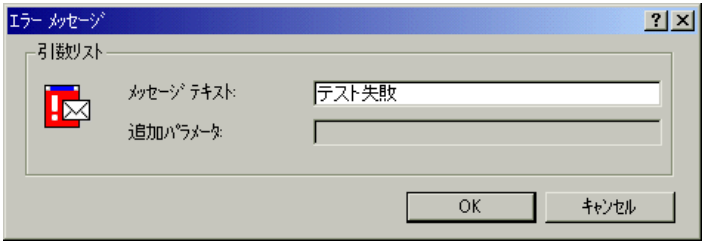
エラーおよび出力メッセージ

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，次の手順で行います。

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** 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：実行環境の設定で [**エラーでも処理を継続する**] を指定します。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，145 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);** 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。

たとえば，実行環境の設定で［エラーでも処理を継続する］を有効にしておけば，仮想ユーザが次のスクリプト・セグメントを再生しているときにエラーが発生しても，仮想ユーザはスクリプトを実行し続けます。

```
    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);
```

ユーザ思考遅延時間のエミュレート

連続する操作の間のユーザの待ち時間を「**思考遅延時間**」といいます。仮想ユーザでは，**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の手作業での追加は，次の手順で行います。

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。[デバッグ メッセージ] 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。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。

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。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，以下の 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
```

たとえば次の例は，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，パスワードやその他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。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，以下の手順で行います。

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

2 暗号化するテキストを選択します。

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

暗号化された文字列を元に戻すには，次の手順で行います。

1 ツリー・ビューを表示できるプロトコルでは，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。

仮想ユーザ・スクリプトでの C 関数の使用方法

VuGen では，仮想ユーザ・スクリプトが C 言語で生成されます。フロー制御や構文を含め，標準の ANSI C のすべての規則がスクリプトに適用されます。他の C 言語プログラムと同じように，コメントや条件ステートメントをスクリプトに追加できます。ANSI C の規則に従って変数を宣言して定義できます。

「**オンライン関数リファレンス**」（[**ヘルプ**] ＞ [**関数リファレンス**]）には，構文やよく使用する C 関数の例が記載された C 言語リファレンスが含まれています。

仮想ユーザ・スクリプトの実行に使用される C インタプリタは，標準の ANSI C 言語を受け付けます。Microsoft 社による ANSI C への拡張はサポートされません。

C 言語関数を仮想ユーザ・スクリプトに追加するときは，以下の制限に注意してください。

- 仮想ユーザ・スクリプトでは，関数のアドレスをコールバックとしてライブラリ関数に渡すことはできません。
- **stdargs**，**longjmp**，および **alloca** 関数は仮想ユーザ・スクリプトではサポートされていません。
- 仮想ユーザ・スクリプトには，構造体引数や戻り値の型はありません。構造体へのポインタはサポートされています。
- 仮想ユーザ・スクリプト内のリテラル文字列は読み取り専用です。リテラル文字列に書き込もうとするとアクセス違反が起こります。

libc 関数の呼び出し

仮想ユーザ・スクリプトで，**libc** 関数を呼び出すことができます。ただし，仮想ユーザ・スクリプトの実行で使われるインタプリタでは Microsoft 社による ANSI C への拡張をサポートしていないため，Microsoft 社のインクルード・ファイルを使うことはできません。必要なときにはユーザが独自にプロトタイプを書くか，マーキュリー・インタラクティブのカスタマー・サポートに連絡して，**libc** 関数のプロトタイプを含む ANSI 準拠のインクルード・ファイルを入手してください。

リンク・モード

仮想ユーザ・スクリプトの実行に使用される C インタプリタでは，使用前までに関数を定義しておけば実行開始時に定義しておく必要のない「遅延」リンク・モードが使用されます。たとえば，次のような場合です。

```
lr_load_dll("mydll.dll");
    myfun();  /* mydll.dll で定義 ― myfun.dll のロード直後に
                 直接呼び出せる。*/
```

第 7 章

パラメータの定義

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（クエリー，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。つまり，スクリプトを「**パラメータ化**」します。

本章では，次の項目について説明します。

- パラメータに関する制限
- パラメータの作成
- パラメータのプロパティの定義
- パラメータのタイプについて
- 内部データの割り当て
- パラメータ形式の指定
- パラメータ値のソースに使うファイルの選択
- 既存データベースからのデータのインポート
- ユーザ定義関数
- パラメータ化のオプション

以下の内容は，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。

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。

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

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

複数の仮想ユーザや反復を使ってスクリプトを再生するときに，「UNIX」という同じ値を繰り返し利用する必要がないときは，定数値をパラメータで置き換えます。

```
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，93 ページ「パラメータのタイプについて」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。たとえば，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。仮想ユーザに対して特定の項目を探すように指示するのではなく，パラメータを代わりに使用するようにします。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 =¥" ＜名前＞ ¥"
", -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を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をパラメータ化できます。**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**」（[**ヘルプ**] ＞ [**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

パラメータの名前とタイプを指定してパラメータを作成します。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，次の手順で行います。

1 [**スクリプト ビュー**]：文字列を選択して，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。

2 [**ツリー ビュー**]：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。対応するプロパティ・ダイアログ・ボックスが開きます。

パラメータ化する引数の横にある［**ABC**］アイコンをクリックします。

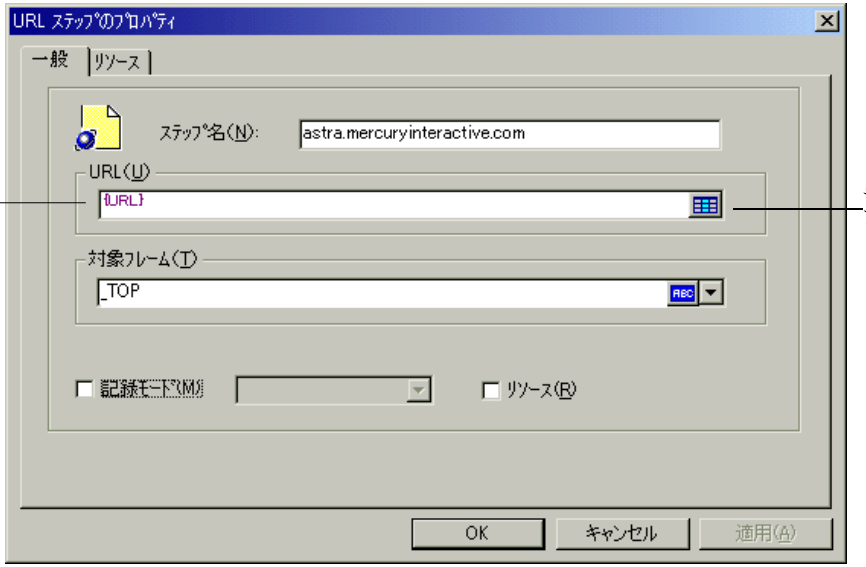
［パラメータの選択または作成］ダイアログ・ボックスが開きます。

3 ［**パラメータ名**］ボックスにパラメータの名前を入力するか，リストから既存のパラメータ名を選択します。

4 ［**パラメータのタイプ**］リストからパラメータのタイプを選択します。選択可能なパラメータのタイプとして，「Date/Time」（日時），「File」（ファイル），「Group Name」（グループ名），「Load Generator Name」（ロード・ジェネレータ名），「Iteration Number」（反復回数），「Random Number」（乱数），「Unique Number」（一意の数），「User Defined Function」（ユーザ定義関数），「Vuser ID」（仮想ユーザ ID）があります。パラメータのタイプの詳細については，93 ページ「パラメータのタイプについて」を参照してください。

5 ［**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，括弧で囲んだパラメータ名で置き換えられます。

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。下の例では，元の URL の値「http://www.merc-int.com/」を，パラメータ **{url}** に置き換えています。



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，文字列を部分的にパラメータ化することはできません。文字列全体をパラメータ化してください。

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。変更する場合は，［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］タブでパラメータの括弧を変更できます。詳細については，115 ページ「パラメータ化のオプション」を参照してください。

複数の出現の置き換え

特定の文字列の一部または全部の出現箇所を新しいパラメータで置き換えることができます。

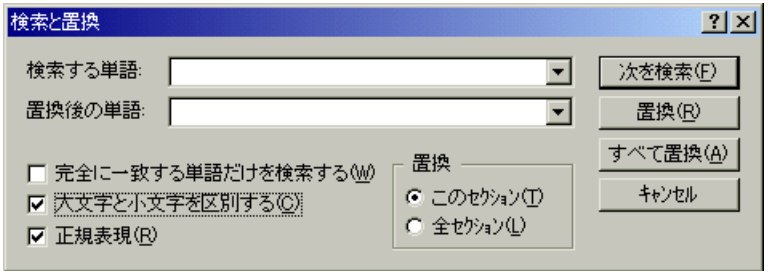
複数の出現を置き換えるには，次の手順で行います。

1 パラメータを選択し，右クリックして表示されるポップアップ・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。詳細については，522 ページ「正規表現の使用」を参照してください。

3 ［**置換**］か［**すべて置換**］をクリックします。



注 : [すべて置換] を使うとき，特に数字文字列を全置換するときには注意してください。VuGen によって，一致した文字列がすべて置換されます。

---

4 以前に定義したパラメータを使って文字列を置換するには，スクリプト・ビューで行います。パラメータ化する文字列を右クリックして，[**既存のパラメータで置換**] を選択します。[**既存のパラメータで置換**] サブメニューが開きます。

このサブメニューからパラメータを選ぶか，[**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・ボックスを開きます。

注 : [**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文字列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，114 ページ「パラメータ・リストの使用」を参照してください。

元の文字列の復元

VuGen では，記録された元の文字列を復元することでパラメータ化を取り消すことができます。

パラメータを元の値に戻すには，次の手順で行います。

1 [**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。

2 [**ツリー ビュー**] : ステップを右クリックしてプロパティを表示し，テーブル・アイコンをクリックして，ポップアップ・メニューから [**パラメータを元に戻す**] を選択します。元の値に戻ります。

パラメータのプロパティの定義

パラメータを作成したら，次にプロパティを定義します。パラメータのプロパティは，スクリプト実行中にパラメータが使用するデータ・ソースを定義します。

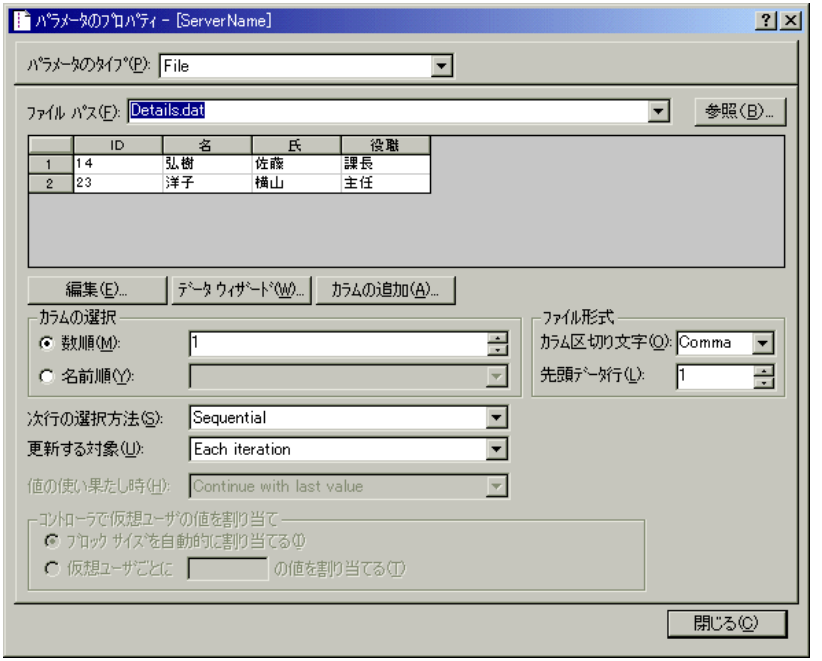
パラメータのプロパティを設定するには，次の手順で行います。

1 ［**スクリプト ビュー**］：パラメータを選択します。右クリックして表示されるポップアップ・メニューから［**パラメータのプロパティ**］を選択します。

2 ［**ツリー ビュー**］：プロパティを定義する必要のあ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プロパティ・ダイアログ・ボックスが開きます。

3 プロパティを定義する必要のあ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。

［パラメータのプロパティ］ダイアログ・ボックスが開き，現在のパラメータ・タイプに対応するプロパティが表示されます。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。



4 パラメータのプロパティを入力します。詳細については，93 ページ「パラメータのタイプについて」を参照してください。

5 ［**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。

パラメータのタイプについて

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・ソースを指定できます。

内部データの割り当て	仮想ユーザによって内部で生成されるデータ。これには，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 が含まれます。
データ・ファイル	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。
ユーザ定義関数	外部の DLL の関数を使用して生成されたデータ。ユーザ定義関数の詳細については，112 ページ「ユーザ定義関数」を参照してください。

内部データの割り当て

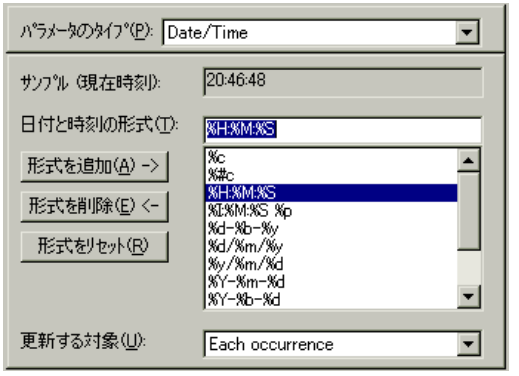
内部データは仮想ユーザの実行中に自動的に生成されるデータです。以下の各項で，内部データの種類について説明します。

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，メニュー・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。新しい形式を作成するには，［**日付と時刻の形式**］ボックスに形式を入力し，［**形式を追加**］をクリックします。形式を削除するには，削除する形式を選択して［**形式を削除**］をクリックします。形式を標準の設定に戻すには，［**形式をリセット**］をクリックします。

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	日付
m	月（01 から 12 までの数字）
b	月（省略した文字列。たとえば Dec）
B	月（省略しない文字列。たとえば December）
Y	年（短い形式。たとえば 03）
Y	年（長い形式。たとえば 2003）



日付と時刻のパラメータのプロパティを設定するには，次の手順で行います。

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**］ボックスで確認できます。

2 ［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，102 ページ「更新する対象の選択」を参照してください。

3 ［**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，次の手順で行います。

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，101 ページ「パラメータ形式の指定」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。

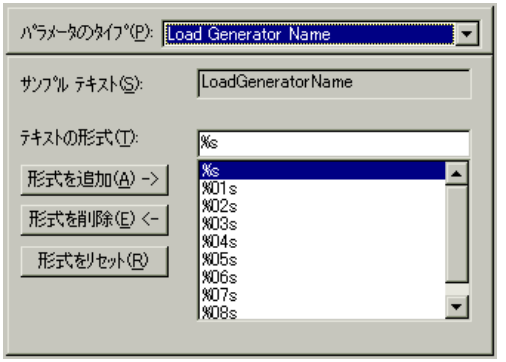
反復回数のパラメータのプロパティを設定するには，次の手順で行います。

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，101 ページ「パラメータ形式の指定」を参照してください。

2 ［**閉じる**］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。

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，次の手順で行います。

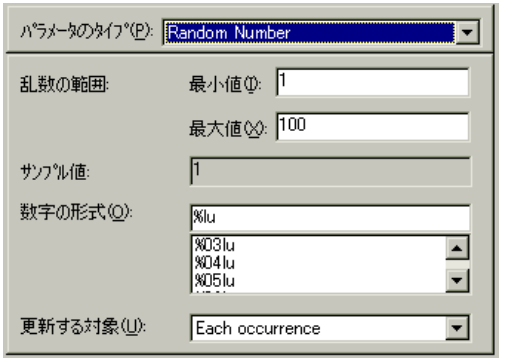
1 利用可能な既存の形式の中から 1 つを選択するか，新しい形式を作成します。形式を選択することによって，パラメータの文字列の長さを指定します。詳細については，101 ページ「パラメータ形式の指定」を参照してください。

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。

乱数を使用すれば，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。たとえば，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ーを実行するには，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，次の手順で行います。

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，一意の数値の長さを指定します。1 桁の数字には %01lu，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，102 ページ「更新する対象の選択」を参照してください。

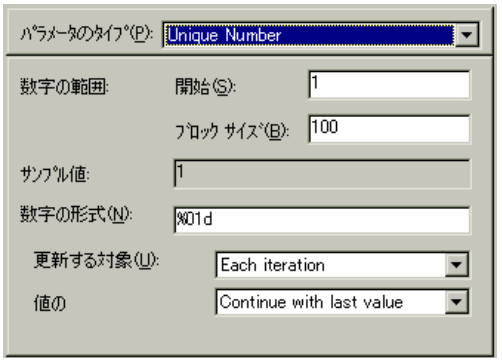
4 ［**閉じる**］をクリックして設定を適用し，［パラメータのプロパティ］ダイアログ・ボックスを閉じます。

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字で置き換えられます。開始値とブロック・サイズを指定します。

一意の数を使用する場合は，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。たとえば開始値として 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。たとえば，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 500 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。たとえば，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ーを実行するには，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 Parameter］（一意のパラメータ）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。たとえば，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 つめのグループは 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，次の手順で行います。

1 開始値とブロック・サイズを指定します。たとえば，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，［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，一意の数値の長さを指定します。1 桁の数字には %01d，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，102 ページ「更新する対象の選択」を参照してください。

4 一意の値がなくなった場合には，［**値を使い果たした時**］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。

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

［Vuser ID］（仮想ユーザ ID）を選ぶと，パラメータは，シナリオ実行中にコントローラが仮想ユーザに割り当てる ID で置き換えられます。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。スクリプトを VuGen から実行すると，Vuser ID は常に -1 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，次の手順で行います。

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，101 ページ「パラメータ形式の指定」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。

データ・ファイル

データ・ファイルには，スクリプト実行中に仮想ユーザがアクセスするデータが格納されています。データは，ローカル・ファイルやグローバル・ファイルとして格納できます。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，データベースをインポートしたりできます。パラメータで使用する既知の値がたくさんあることがわかっている場合は，データ・ファイルが役立ちます。

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

次の例では，データ・ファイルに ID 番号と名前を格納しています。

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

データ・ファイルのパラメータのプロパティ設定の詳細については，103 ページ「パラメータ値のソースに使うファイルの選択」を参照してください。データベースのパラメータのプロパティ設定の詳細については，109 ページ「既存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。

パラメータ形式の指定

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指定します。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。たとえば，%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。たとえば，%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。たとえば，「VuserNo: %03s」という形式を指定した場合には，仮想ユーザ ID の 1 という数字は「Vuser No: 001」と表示されます。

形式を追加するには，次の手順で行います。エディット・ボックス内に形式指定記号を入力し，［**形式の追加**］をクリックします。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，次の手順で行います。既存の形式を選択し，［**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，次の手順で行います。［**形式をリセット**］をクリックします。

更新する対象の選択

Date/Time（日時），Random（乱数），Unique（一意の数），User Defined Function（ユーザ定義関数）などのパラメータ・タイプを使用するとき，VuGen ではパラメータの更新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。以下の 3 つ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**」方式を選択すると，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。たとえば，ランダムなデータを使用する場合には，パラメータが現れるたびに異なる値を使うと有効です。

Each Iteration（反復ごと）

「**Each iteration**」方式を選択すると，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，131 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**」方式を選択すると，仮想ユーザはシナリオの実行時に，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。

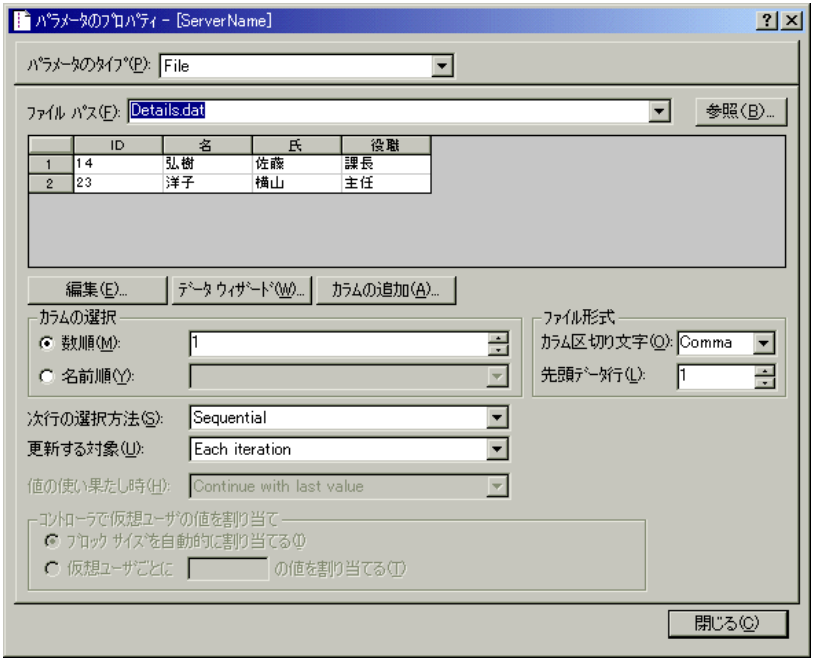
パラメータ値のソースに使うファイルの選択

パラメータを使用するための非常に一般的な方法は，仮想ユーザに外部ファイルから値を取ってくるように指定することです。次の手順で行います。

- ➤ データ・ファイ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

データ・ファイルの選択または作成

パラメータ・タイプが［**File**］（ファイル）のときに［パラメータのプロパティ］ダイアログ・ボックスを開くと，そのファイルのプロパティの設定が表示されます。



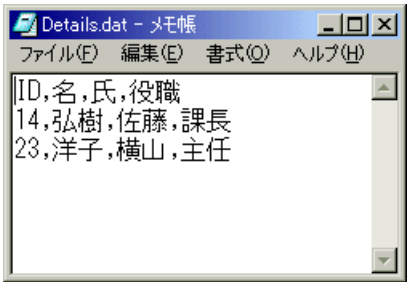
データの最初の 100 行だけが表示されます。すべてのデータを表示するには，［編集］をクリックし，［メモ帳］でデータを表示します。

パラメータのソース・ファイルを選択するには，次の手順で行います。

1 ［**ファイル パス**］ボックスにデータ・ファイルの名前を入力するか，［**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。既存のデータ・ファイルの拡張子は，**.dat** でなければなりません。

グローバル・ディレクトリも指定できます。グローバル・ディレクトリは，LoadRunner の以前のバージョンとの上位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，117 ページ「グローバル・ディレクトリ」を参照してください。

2 ［**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。追加するカラム名と値を，表形式で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ってカラムの区切りを示します。表の新しい行ごと（すなわちデータ行ごと）に改行します。



注 : ［メモ帳］を使わずにデータ・ファイルにカラムを追加するには，［パラメータのプロパティ］ダイアログ・ボックスで［**カラムの追加**］をクリックします。［**新規カラムの追加**］ダイアログ・ボックスが開きます。［**カラム名**］ボックスに新しいカラム名を入力し，［**OK**］をクリックします。VuGen によって，新しいカラムがテーブルに追加され，1 行目にパラメータの元の値が表示されます。

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，割り当ての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。たとえば，これらのプロパティは，使用するカラム，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，次の手順で行います。

1 パラメータ用の値を含むテーブルのカラム番号を指定します。［**カラムの選択**］セクションで，カラム番号かカラム名を指定します。

カラム番号で指定するには，［**数順**］を選択してカラム番号を選びます。カラム番号とはデータを含んでいるカラムのインデックスです。たとえば，パラメータのデータがテーブルの第 1 カラムにある場合，「1」を選択します。

カラム名で指定するには，［**名前順**］を選択して，リストからカラム名を選びます。カラム名は，各カラムの最初の行（0 行）にあります。カラム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，カラム名を使ってカラムを指定してください。

2 ［**ファイル形式**］セクションの［**カラム区切り文字**］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 ［**次行の選択方法**］リストからオプションを選択して，仮想ユーザが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。更新方法には，［Sequential］（順次），［Random］（ランダム），［Unique］（一意）または［Same Line As parameter］（パラメータと同じ行）があります。詳細については，106 ページ「ファイル・タイプのパラメータに値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，102 ページ「更新する対象の選択」を参照してください。

6［**次行の選択方法**］で「**Unique**」を選択した場合には，以下を指定します。
［**値を使い果たした時**］：一意の値がなくなった場合の処理として，「**Abort Vuser**」，「**Continue in 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**］：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。［**ブロック サイズを自動的に割り当てる**］または［**仮想ユーザごとに X の値を割り当てる**］のいずれかを選択し，後者の場合には，割り当てる値の数を指定します。

注：［**File**］（ファイル）タイプの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。新しい［**File**］パラメータを作成すると，VuGen によってデータ・ファイルを作成するように求められます。［**作成**］をクリックします。［**作成**］ボタンが［**編集**］ボタンに置き換わります。［**編集**］ボタンをクリックして［メモ帳］を開き，前述したとおりにデータを入力します。

ファイル・タイプのパラメータに値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 ではパラメータに値を割り当てる方法を指定できます。以下の方法があります。

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

Sequential（順次）

「**Sequential**」方式は，パラメータの値を順次（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。

［**更新する対象**］リスト・ボックスで［**Each Iteration**］（反復ごと）を選択すると，仮想ユーザによって，反復ごとにデータ・テーブルから次の値が取得されます。

［**更新する対象**］リスト・ボックスで［**Each Occurrence**］（すべての出現）を選択すると，仮想ユーザによって，同じ反復の中でもパラメータが出現するたびにデータ・テーブルから次の値が取得されます。

［**更新する対象**］リスト・ボックスで［**Once**］（1 回のみ）を選択すると，仮想ユーザごとに，最初の反復で割り当てられた値が以降のすべての反復で使用されます。

First Name
Kim
David
Michael
Jane
Ron
Alice
Ken
Julie

たとえば，テーブルに，左の表に示すような値があるとします。

［**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。

［**Each Occurrence**］を選択すると，すべての仮想ユーザが最初の出現では 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。

［**Once**］オプションを選択すると，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。

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。

［**更新する対象**］リスト・ボックスで［**Each Iteration**］（反復ごと）を選択すると，仮想ユーザによって，反復ごとにデータ・テーブルから新しい値がランダムに取得されます。

［**更新する対象**］リスト・ボックスで［**Each Occurrence**］（すべての出現）を選択すると，仮想ユーザによって，同じ反復の中でもパラメータが出現するたびにデータ・テーブルから次の値がランダムに取得されます。

［**更新する対象**］リスト・ボックスで［**Once**］（1 回のみ）を選択すると，仮想ユーザごとに，最初の反復でランダムに割り当てられた値が以降のすべての反復で使用されます。

LoadRunner コントローラからシナリオ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。各シード値は，テスト実行で使用される個別の乱数シーケンスを表します。同じシード値を使用すれば，シナリオの仮想ユーザには，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，『**LoadRunner コントローラ・ユーザーズ・ガイド**』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。

［**更新する対象**］リスト・ボックスで［**Each Iteration**］を選択すると，仮想ユーザは反復ごとにデータ・テーブルから次の一意の値を取り出します。

［**更新する対象**］リスト・ボックスで，［**Once**］選択すると，最初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反復で使用されます。

［**更新する対象**］リスト・ボックスで［**Each Occurrence**］（すべての出現）を選択すると，仮想ユーザによって，同じ反復の中でもパラメータが出現するたびにデータ・テーブルから次の値が一意に取得されます。

First Name
Kim
David
Michael
Jane
Ron
Alice
Ken
Julie
Fred

たとえば，テーブルに，左の表に示すような値があるとします。

［**Each Iteration**］オプションを指定し，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。

［**Once**］オプションを指定すると，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。

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ければ，［**Abort the 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）からいずれかを選んで VuGen の以降の処理の進め方を指定します。最後の値を使い続ける（Continue with last value）ようにした場合，仮想ユーザは以降のすべての反復でデータ・テーブルの最後の行のデータを使用します。

各仮想ユーザに値を割り当てるときに，仮想ユーザ間で同じ値を使用させたくない場合を考えてみます。そのためには，［**コントローラで仮想ユーザの値を割り当て**］セクションで各仮想ユーザに特定の数の値を割り当てるよう指示します。標準設定では，仮想ユーザに必要な数の値が VuGen によって自動的に割り当てられます。

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

既存データベースからのデータのインポート

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

- 新規クエリーの作成
- 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ーを作成するか，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 クエリー ウィザード］が開きます。

新規クエリーの作成

新規クエリーの作成には，Microsoft のデータベース・クエリー・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。

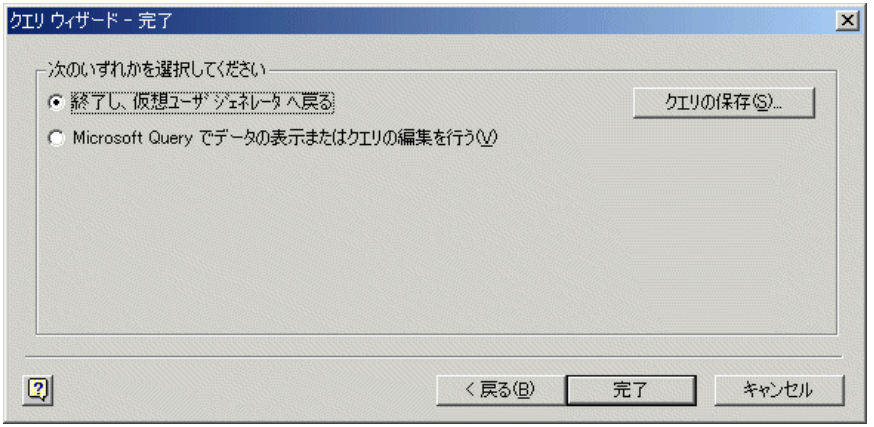
新規クエリーを作成するには，次の手順で行います。

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 LoadRunner によって表示されます。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 リスト］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。105 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

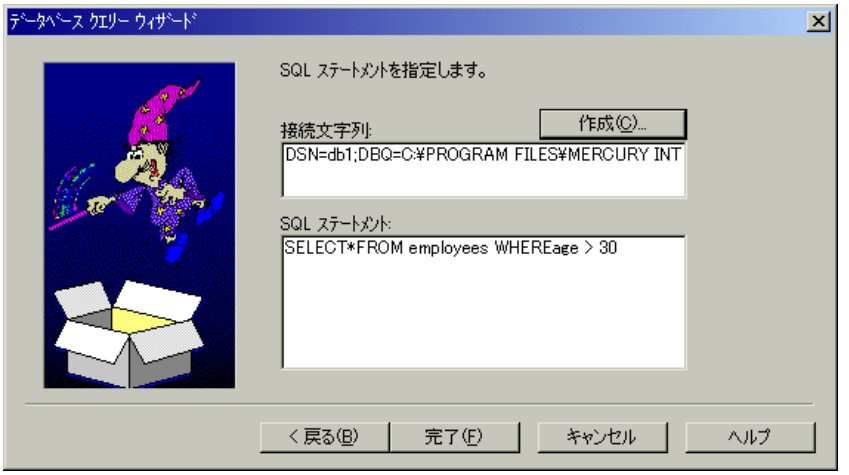
データベース接続と SQL ステートメントを指定するには，次の手順で行います。

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 リスト] 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。105 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。

ユーザ定義関数

ユーザ定義関数は，パラメータを外部 DLL の関数から返される値で置き換えます。

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

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

この関数に送られる引数は両方とも NULL です。

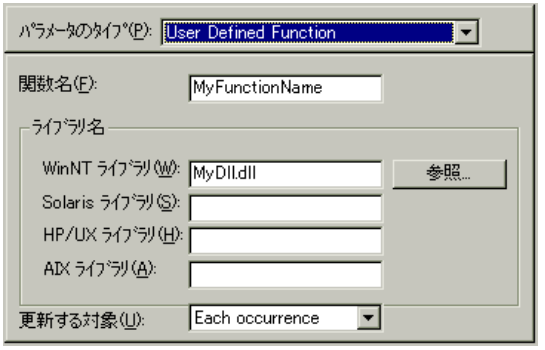
ライブラリを作成するときには，標準のダイナミック・ライブラリ・パスを使うことをお勧めします。そうすれば，ライブラリの完全パス名を入力する必要がなく，ライブラリ名を入力するだけで済みます。仮想ユーザ・ジェネレータ

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;}
```

［**User Defined Function**］タイプを選択すると，ユーザ定義関数のプロパティのタブが開きます。



ユーザ定義関数のプロパティを設定するには，次の手順で行います。

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。

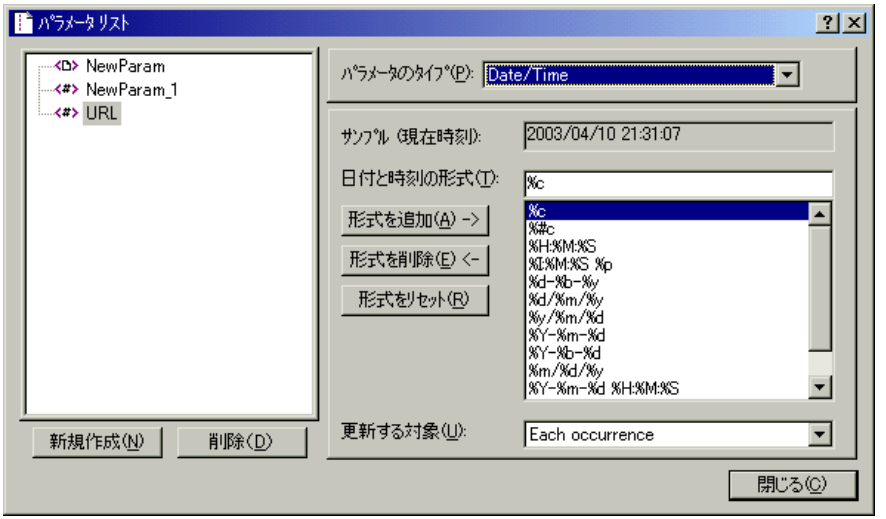
3 値の更新方法を選択します。ユーザ定義関数の更新方法の詳細については，102 ページ「更新する対象の選択」を参照してください。

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，既存パラメータのプロパティの変更が行えます。

パラメータ・リストを使用するには，次の手順で行います。

1 ［**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。



2 パラメータを新規作成するには，［**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。

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。

パラメータのタイプとプロパティを設定して［**閉じる**］をクリックし，［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。

3 既存のパラメータを削除するには，パラメータ・ツリーからパラメータを選択して，［**削除**］をクリックします。確認を求められるので承諾します。

4 既存のパラメータを変更するには，パラメータ・ツリーからパラメータを選択して，パラメータのタイプとプロパティを編集します。

パラメータのプロパティ設定の詳細については，93 ページ「パラメータのタイプについて」を参照してください。

パラメータ化のオプション

VuGen では，次のパラメータ化オプションを設定できます。

- パラメータの括弧
- グローバル・ディレクトリ

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き，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);
```

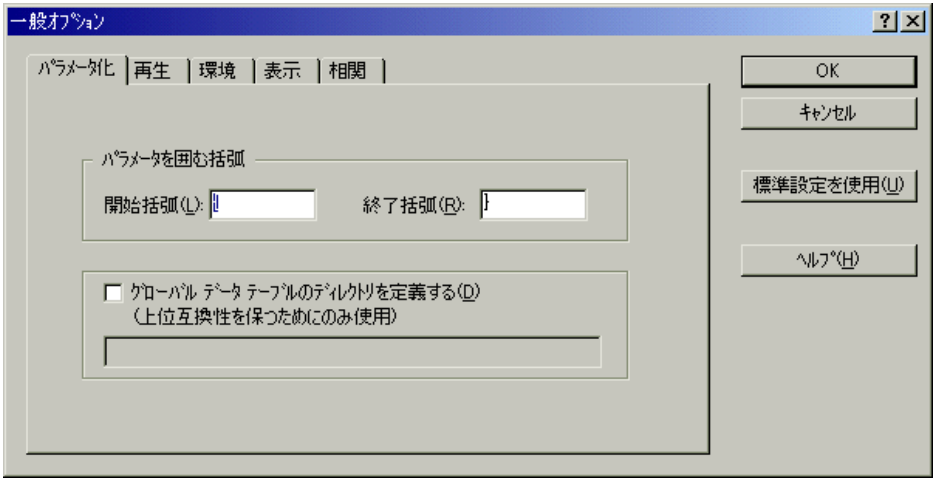
1 文字または複数の文字で構成される文字列を指定して，パラメータの括弧のスタイルを変更できます。スペース以外のすべての文字を使用できます。

注：標準のパラメータの括弧は，仮想ユーザの種類に応じて中括弧か大括弧のどちらかです。

パラメータの括弧のスタイルを変更するには，次の手順で行います。

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，LoadRunner 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。

LoadRunner の新しいバージョンでは，［パラメータのプロパティ］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。LoadRunner 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，100 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，次の手順で行います。

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 8 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 の相関クエリー機能によって，ステートメントの結果をほか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，次の項目について説明します。

- 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

たとえば，相互に依存するクエリー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ー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ーを連結できます。

➤ 動的データ使用のため

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（たとえばデータベース）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。たとえば，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，相関クエリーによって，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，124 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，クエリー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。**

クエリー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 15 章「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 19 章「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 41 章「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 26 章「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，123 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ー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 15 章「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。

ほか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。たとえば，実行の時刻を示す受信バッファは相関を必要とはしません。

WDiff を使って相関を検索するには，次の手順で行います。

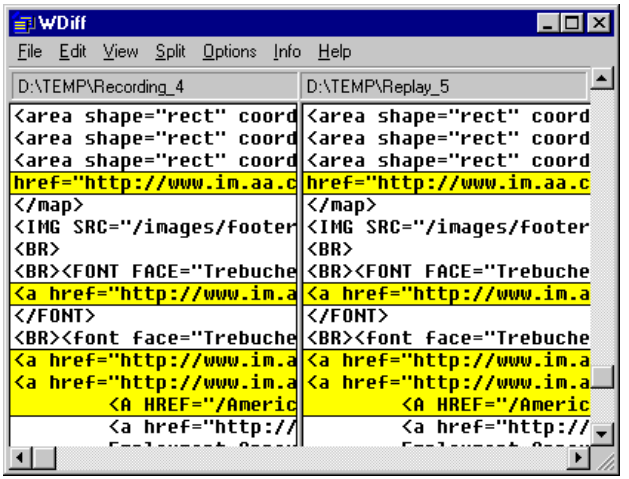
1 スクリプトを記録し，保存します。

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。

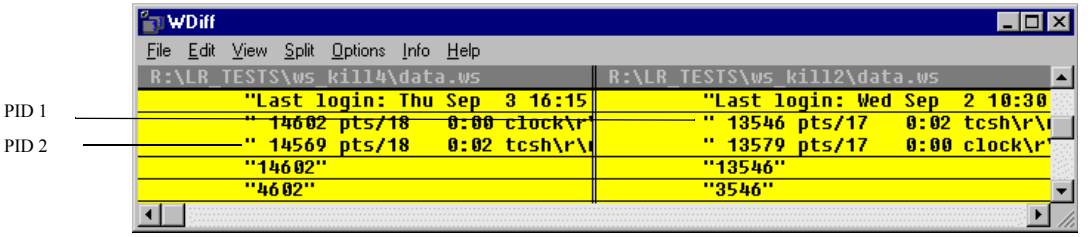
3 比較したい部分（**Actions** や **data.ws** など）を選択します。

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，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 15 章「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 19 章「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 41 章「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 26 章「COM 仮想ユーザ・スクリプトについて」を参照してください。
- WinSock 仮想ユーザについては，第 22 章「Window Sockets データの処理」を参照してください。
- Tuxedo 仮想ユーザについては，第 54 章「Tuxedo 仮想ユーザ・スクリプトの作成」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 9 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，以下の項目について説明します。

- 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
- 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 またはコントローラを使ってスクリプトを実行するときに，仮想ユーザに適用されます。

実行環境の設定を行うことによって，さまざまな種類のユーザの動作をエミュレートできます。たとえば，サーバの出力にすぐに応答するユーザをエミュレートすることも，作業を停止して考えてから応答するユーザをエミュレートすることもできます。また実行環境の設定では，仮想ユーザがアクションを反復する回数も指定できます。

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，以下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［**実行環境の設定を編集**］ボタンをクリックします。
- キーボードのショートカット・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 コントローラからも変更できます。［**デザイン**］タブをクリックし，［**実行環境の設定**］ボタンをクリックします。

注：VuGen のデバッグ環境とコントローラの負荷テスト環境をサポートするために，VuGen とコントローラでは，仮想ユーザ・スクリプトの実行環境の標準設定が異なります。

VuGen とコントローラにおける仮想ユーザ・スクリプトの標準設定は，以下のとおりです。

［**思考遅延時間**］：VuGen では無効になっており，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 では標準で設定されており，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。以下の設定があります。

- 実行論理（反復）
- ペースの設定
- ログ
- 思考遅延時間
- その他

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，反復とペースの設定の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。

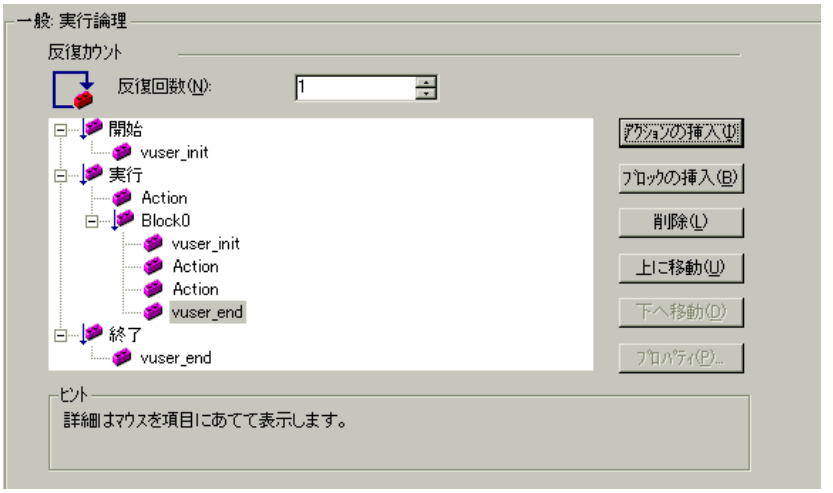
実行論理の設定（マルチ・アクション）

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，136 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。スクリプト実行時に，仮想ユーザが「**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。LoadRunner によって，指定した回数だけ，すべての Action セクションが繰り返し実行されます。

コントローラのスケジュールの設定でシナリオの継続時間を指定すると，こ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。

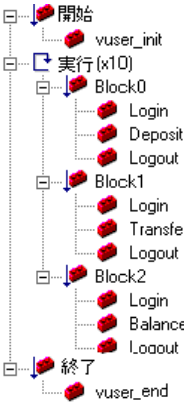
反復：「**実行**」セクション全体の反復回数を設定する以外に，各アクションまたはアクション・ブロックの反復回数を設定することもできます。これはたとえば，製品を探すのに多数のクエリーを実行するけれども，購入する場合は 1 回だけというような商用サイトでの操作のエミュレーションに役立ちます。

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**，つまり，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，次の手順で行います。

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，136 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
- 一定 / 値の範囲

[**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**］：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。たとえば，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実際に待機した時間は，実行ログに示されます。

［**一定**］/［**値の範囲**］：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。たとえば，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実際に待機した時間は，実行ログに示されます。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。

たとえば，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。LoadRunner 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。

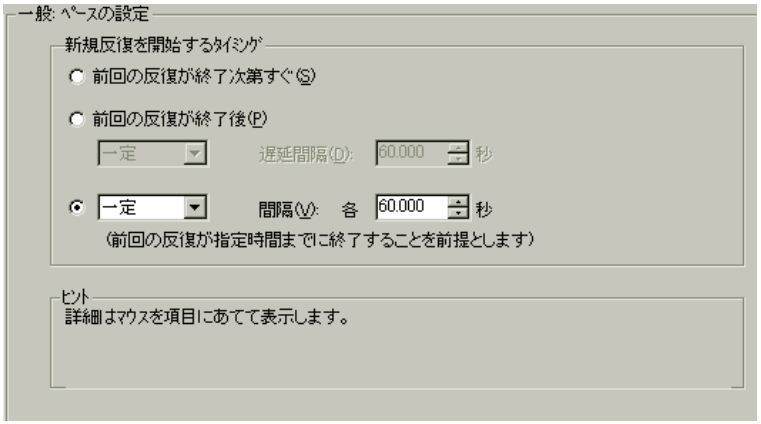
ペースの設定のオプションの概要については，135 ページ「実行環境のペースの設定（マルチ・アクション）」を参照してください。

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。

反復の間隔の設定は，次の手順で行います。

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］/［遅延間隔］［… 秒］
- ［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［**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 ［**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）

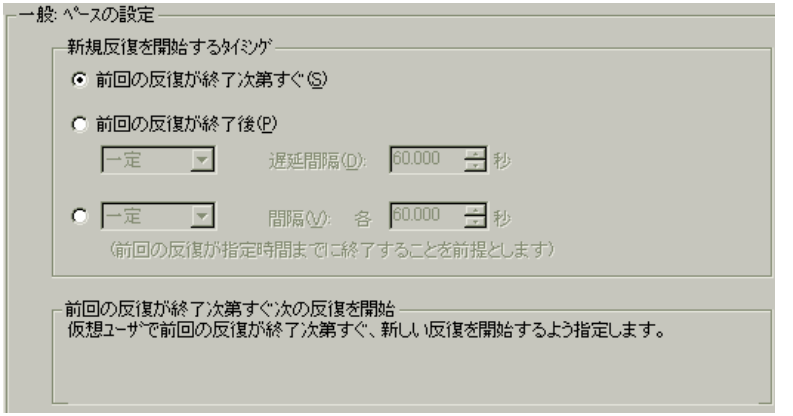
注：次の項の内容は，複数のアクションではなく単一のアクションを使用するプロトコルを対象としています。［実行環境設定］の［一般］の下に［**ペースの設定**］ノードと［**実行論理**］ノードがなければ，単一アクションのプロトコルです。

スクリプト実行時に **Action** セクションを繰り返すよう仮想ユーザに指示します。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。

コントローラのスケジュールの設定でシナリオの継続期間を指定すると，こ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。

反復とペースの設定を設定するには，次の手順を行います。

1 VuGen ツールバーの［**実行環境の設定を編集**］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，［**実行環境の設定**］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。LoadRunner によって，指定した回数だけ，すべての Action セクションが繰り返し実行されます。

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］/［遅延間隔］［… 秒］
- ［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［**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。

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，133 ページ「ペースの設定」を参照してください。

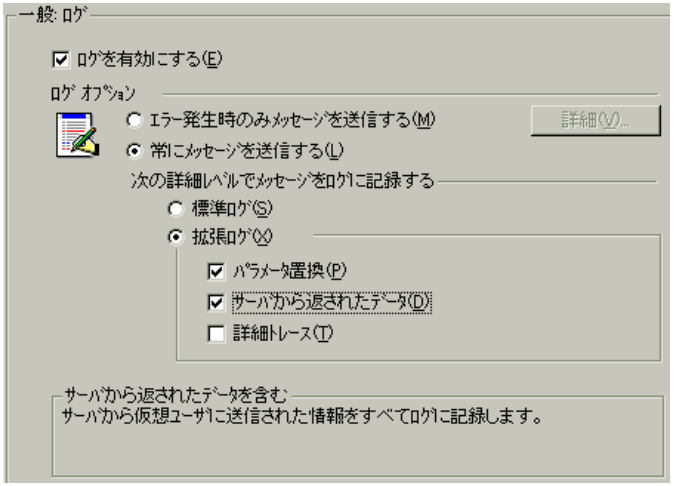
実行環境設定のログの設定

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，仮想ユーザ・スクリプトをプログラミングして，出力にメッセージを送信できます。

［**実行環境の設定を編集**］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，［**実行環境の設定**］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。開発時には，ログ記録を有効にしておけます。開発中には，すべてのログを有効にし，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・ログのみを有効にします。

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。141 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：[**実行環境の設定**] 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送信された関数や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に追加した場合，ログ記録モードが自動的に JIT ログ記録に設定されます。

[**拡張ログ**]：警告やメッセージを含めた詳細なログが作成されます。大規模な負荷テスト・シナリオでは，このオプションは無効にしてください。ログ記録が無効になっているスクリプト，またはログ記録レベルが**拡張**に設定されているスクリプトをシナリオに追加してもログ記録の設定には影響ありません。

[拡張ログ] オプションで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。詳細については，第 7 章 「パラメータの定義」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。

lr_set_debug_message 関数を使えば，スクリプト内にメッセージ・クラスを手作業で設定できます。これは，スクリプトのごく一部分についてだけデバッグ情報を受け取りたい場合などに便利です。

たとえば，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は拡張ログをとりたいとします。その場合には，**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。

ログのモードが［**標準ログ**］に設定されていた場合，スクリプトをシナリオにコピーするときに，ログ・モードが VuGen によって自動的に JIT モードに設定されます。ログのモードが［**拡張ログ**］に設定されているか，ログが無効になっている場合，スクリプトをシナリオにコピーしても，これらのログの設定は何の影響も受けません。

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。たとえば，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・メッセージをコントローラの出力ウィンドウに（標準ログ以外に）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

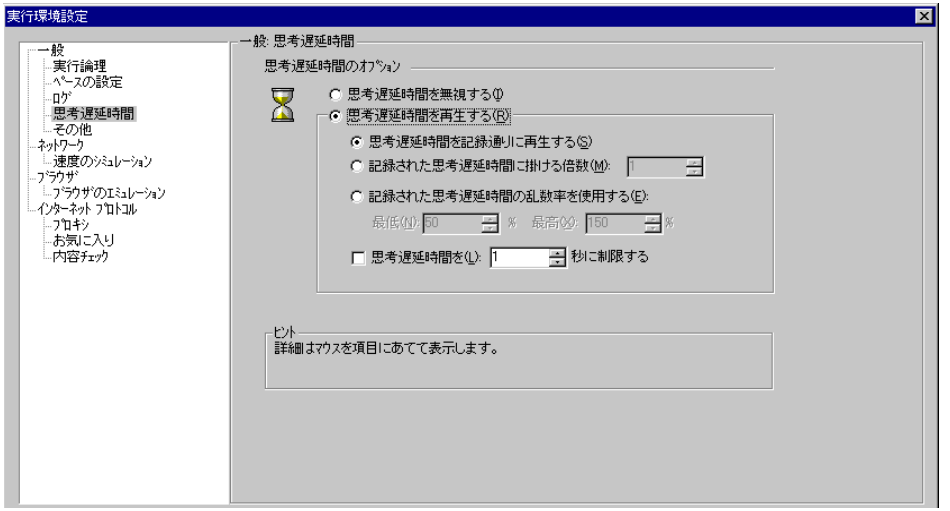
仮想ユーザの「**思考遅延時間**」は実際のユーザがアクションとアクションの間で待つ時間をエミュレートします。たとえば，ユーザはサーバからデータを受け取ったときに，データを数秒間かけて確認してから応答するかもしれません。この遅れを「**思考遅延時間**」といいます。VuGen は **lr_think_time** 関数を使って，仮想ユーザ・スクリプトに思考遅延時間の値を記録します。次に示す記録された関数は，次のアクションを実行する前にユーザが 8 秒待機したことを示します。

```
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** ステートメントに遭遇すると，仮想ユーザによって次のアクションが実行されるまで 8 秒間の待機時間があります。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［**実行環境の設定を編集**］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，［**実行環境の設定**］ダイアログ・ボックスを表示します。［**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**］：思考遅延時間オプションの第 2 セクションでは，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。たとえば，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。たとえば，4 秒間の思考遅延時間が記録されている場合に，その値を 2 倍するように指定すれば，仮想ユーザの思考遅延時間は 8 秒になります。思考遅延時間を 2 秒に減らすには，記録された時間に 0.5 を掛けます。

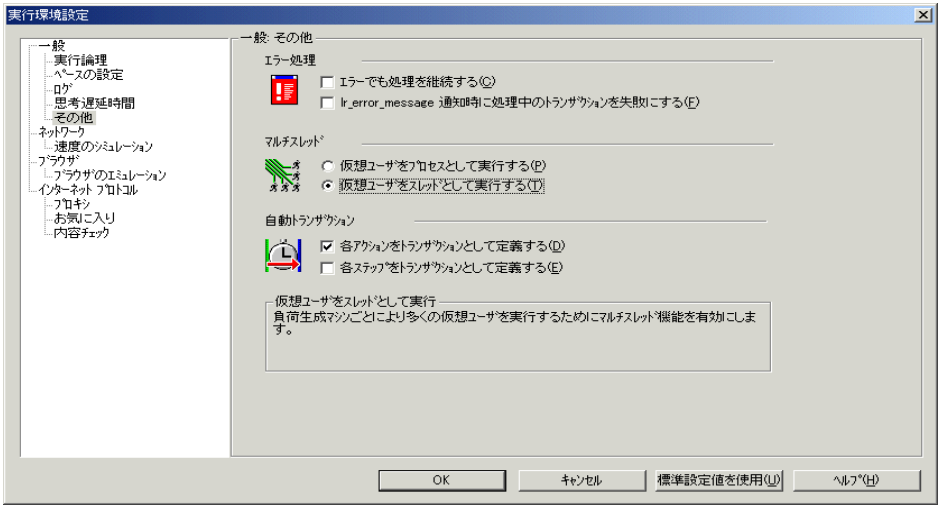
- ［**記録された思考遅延時間の乱数率を使用する**］：記録された思考遅延時間のランダムな割合（%）を使用します。思考遅延時間の範囲を指定することによって，思考遅延時間の値の範囲を設定します。たとえば，思考遅延時間の引数が 4 の場合，下限を 50%，上限を 150% に設定すると，最短の思考遅延時間は 2（50%），最長の思考遅延時間は 6（150%）となります。
- ［**思考遅延時間を：・・・秒に制限する**］：思考遅延時間の最大値を制限します。

その他の設定

仮想ユーザ・スクリプトでは，次のその他の実行環境オプションを設定できます。

- エラー処理
- マルチ・スレッド
- 自動トランザクション

［**実行環境の設定を編集**］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

仮想ユーザがスクリプトの実行時にどのようにエラーを処理するかを指定できます。標準設定では，仮想ユーザはエラーを検出すると終了します。実行環境の設定を使って，仮想ユーザにエラーが発生してもスクリプトの実行を継続させることができます。実行を継続するように設定するには，実行環境設定の［その他］タブの［**エラーでも処理を継続する**］チェック・ボックスを選択します。

VuGen で，**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**」にするよう指定できます。**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。

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。

たとえば，［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行う関数はすべて，関数の最後のパラメータ（**miDBErrorSeverity**）で指定される重要度のレベルに基づいて処理を行います。

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

たとえば，次のデータベース・ステートメントが失敗した場合（たとえば，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。

データベース仮想ユーザのエラー処理

RTE 仮想ユーザを使っているときに，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，スクリプトの終了または別の **lr_continue_on_error** ステートメントに遭遇するまで新しい設定を使用します。

たとえば，［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( . . . );
```

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 **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 を渡してオプションを有効にします。

```
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。スレッドセーフ・プロトコルのみ，スレッドとして実行できます。

注：Sybase-CtLib，Sybase-DbLib，Informix，Tuxedo，PeopleSoft-Tuxedo の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。

コントローラは，ドライバ・プログラム（たとえば mdrv.exe，r3vuser.exe）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，コントローラは 50 の仮想ユーザに対してドライバ・プログラム（たとえば mdrv.exe）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 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。これを「自動トランザクションの使用」といいます。LoadRunner 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 6 章 「仮想ユーザ・スクリプトの拡張」を参照してください。

VB 実行環境の設定

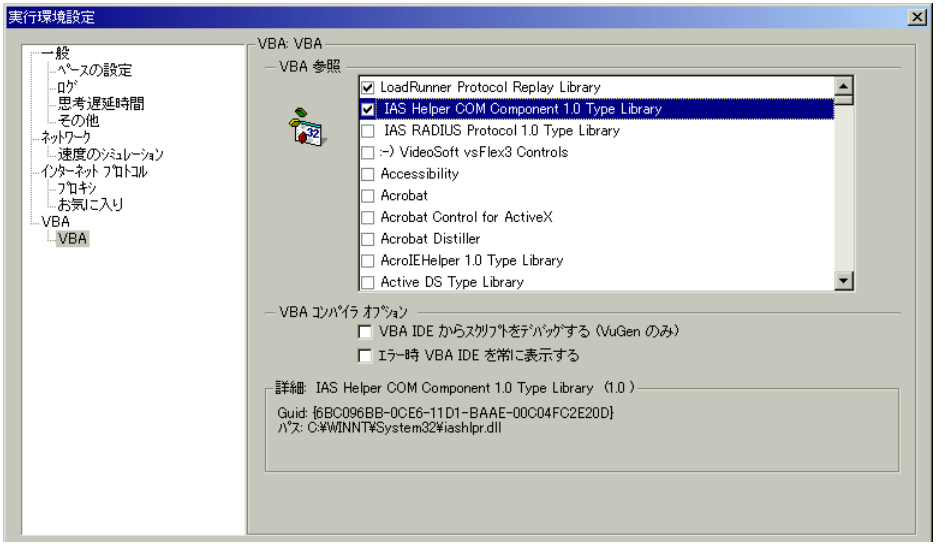
Visual Basic によるスクリプトを実行する前に，再生時に参照するライブラリを指定します。

［実行環境設定］ダイアログ・ボックスを使って，実行環境設定の表示と設定をします。［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［**実行環境の設定を編集**］ボタンをクリックします。

実行環境の設定は，LoadRunner コントローラからも変更できます。［**デザイン**］タブをクリックし，［**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，次の手順で行います。

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 10 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，次の項目について説明します。

- 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，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 9 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。

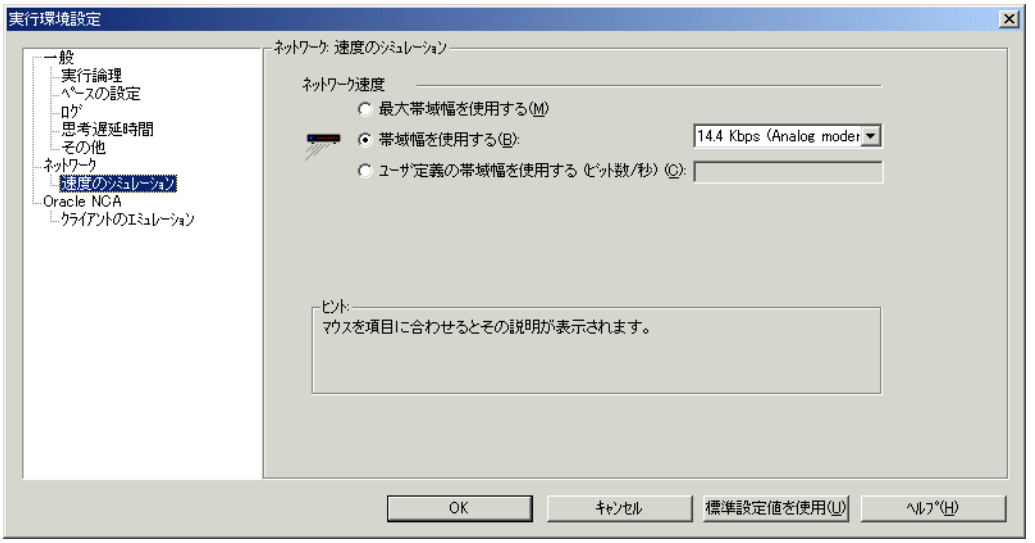
［実行環境設定］ダイアログ・ボックスの表示は，次のいずれかの手順で行います。

- VuGen ツールバーの［**実行環境の設定を編集**］ボタンをクリックします。
- キーボードのショートカット・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 コントローラからでも実行環境の設定を変更できます。［コントローラ］ウィンドウで［**デザイン**］タブをクリックし，［**実行環境の設定**］ボタンをクリックします。

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 11 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。

- VuGen での仮想ユーザ・スクリプトの実行
- 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 ウィンドウでの作業
- 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
- シナリオへの仮想ユーザ・スクリプトの組み込み

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。

仮想ユーザ・スクリプトのスタンドアロン・モードでの実行について

仮想ユーザ・スクリプトを使って負荷テストを実行するには，コントローラを使ってスクリプトをシナリオに組み込みます。スクリプトを負荷テスト・シナリオに組み込む前に，スクリプトをスタンドアロン・モードで実行して，正しく機能することを確認しておきます。

スクリプトをスタンドアロン・モードで実行するということは，コントローラを使わずにスクリプトを実行するということです。これは，コントローラで実行したときにスクリプトがどのように動作するかを確かめるために行います。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。詳細については，第 61 章 「GUI 仮想ユーザ・スクリプトの作成」を参照してください。その他の Windows ベースのスクリプトについては，VuGen を使ってスタンドアロン・モードで実行します。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。

スタンドアロンでの実行が成功したら，スクリプトをシナリオに組み込みます。シナリオの詳細については，『**LoadRunner コントローラ・ユーザーズ・ガイド**』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 6 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 7 章「パラメータの定義」参照）
- スクリプトのクエリーの相関（第 8 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，165 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。

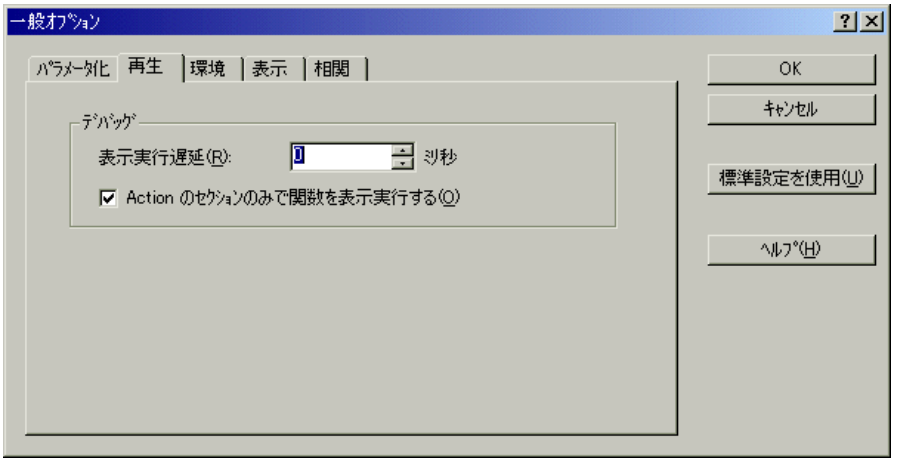
［**表示実行遅延**］：コマンドを実行する遅延間隔をミリ秒単位で指定します。標準の遅延時間は，0 です。

［**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。

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，次の手順で行います。

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。

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**］）を設定します。［表示］オプションによって，VuGen でブラウザを表示するか，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，161 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

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。

仮想ユーザ・スクリプトはスクリプトの先頭から実行されます。［**実行ログ**］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の中で実行されたときに，どのように実行されるかを示します。

注：ツリー・ビューをサポートするプロトコルの場合，ツリー・ビュー（［**表示**］メニュー）から仮想ユーザ・スクリプトを実行すると，VuGen によって仮想ユーザ・スクリプトの最初のアイコンからスクリプトが実行されます。

---

2 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。

3 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。

出力ウィンドウ実行ログ

［実行ログ］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の中で実行されたときに，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，実行ログのメッセージを調べます。

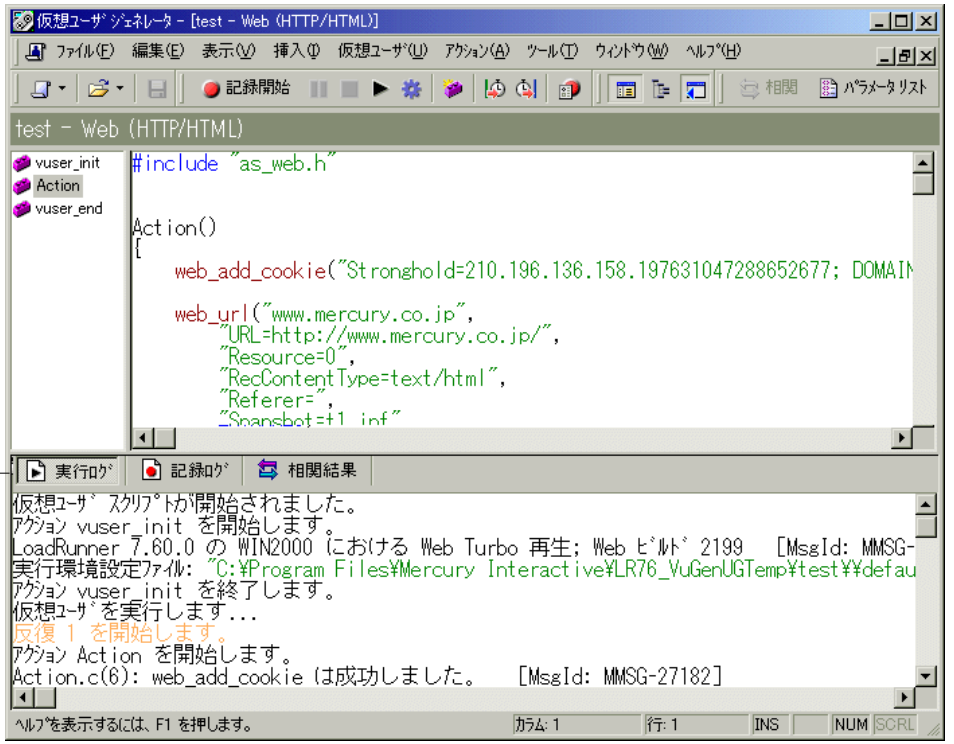
［実行ログ］では，テキストがさまざまに色分けされています。

- **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）

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ついては，157 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実行ログのメッセージを示します。



実行ログ・メッセージ

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，161 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。

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を実行するには，次の手順で行います。

1 ［**仮想ユーザ**］＞［**ステップごとに実行**］を選択するか，［**ステップ実行**］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検証できます。

ブレークポイントを設定するには，次の手順で行います。

1 スクリプト内の実行を停止する行にカーソルを移動します。

2 ［**ブレークポイントの設定 / 解除**］ボタンをクリックします。ブレークポイントの記号（  ）がスクリプトの左マージンに現れます。

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

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，次の手順で行います。

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検証して，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。

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 向けにマーキュリー・インタラクティブ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。実行時ビューアの詳細については，第 45 章 「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。

注：実行時ビューアを表示するには，Microsoft Internet Explorer 4.0 以上のブラウザがインストールされている必要があります。

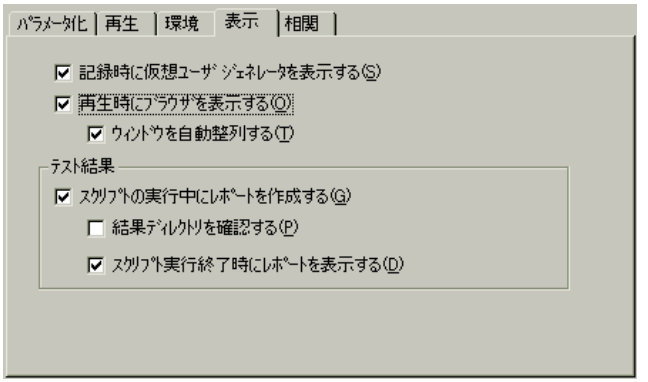
➤ Web 仮想ユーザに対して，スクリプトの実行時に結果サマリ・レポートを生成させるかどうかを指定できます。結果サマリ・レポートとは，Web 仮想ユーザ・スクリプト内の各ステップの成功や失敗についてまとめたもので，各ステップで返された Web ページも確認できます。結果サマリ・レポートの使い方の詳細については，第 44 章 「レポートを使った仮想ユーザ・スクリプトの デバッグ」を参照してください。

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。コントローラを使って Web 仮想ユーザ・スクリプトを実行するとき，仮想ユーザではレポートを生成できません。

Web 仮想ユーザ・スクリプトのデバッグ機能を有効にするには，次の手順で行います。

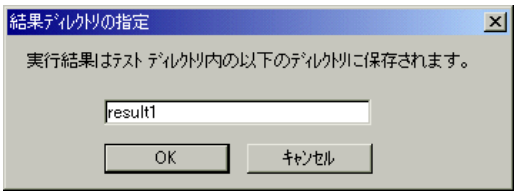
1 VuGen のメニューから［**ツール**］＞［**一般オプション**］を選択します。［一般オプション］ダイアログ・ボックスが開きます。［**表示**］タブを選択します。



2 記録中に VuGen を表示するには，［**記録時に仮想ユーザ ジェネレータを表示する**］チェック・ボックスを選択し，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。

4 仮想ユーザ・スクリプトを実行する前に［結果ディレクトリの指定］ダイアログ・ボックスを表示させるには，［**結果ディレクトリを確認する**］チェック・ボックスを選択します。



実行結果を格納するフォルダ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。

［**結果ディレクトリを確認する**］チェック・ボックスがチェックされていない場合，VuGen によって自動的にフォルダに **result1** という名前を付けられます。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・フォルダのサブフォルダに格納されます。

5 スクリプト実行の最後に結果サマリ・レポートを自動的に表示させるには，［**スクリプト実行終了時にレポートを表示する**］チェック・ボックスを選択します。このオプションを選択していない場合は，スクリプト実行後に，［**表示**］＞［**テスト結果**］を選択して，結果サマリ・レポートを開くことができます。

6 スナップショット相関メカニズムを使用するには，［**再生時に相関情報を保存する**］チェック・ボックスを選択します。各スナップショットが，記録された元のスクリプトと比較されるようになります。

7 ［**OK**］をクリックして設定を適用し，［一般オプション］ダイアログ・ボックスを閉じます。

VuGen ウィンドウでの作業

仮想ユーザ・スクリプトを作成するとき，複数のスクリプトやウィンドウを見なくてはならないことがあります。次の VuGen の機能を使用します。

➤ **［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。

➤ **グリッドの表示**

［**表示**］＞［**データ グリッド**］を選択して，結果データが含まれているグリッドの表示 / 非表示を切り替えます。

➤ **すべてのウィンドウを閉じる**

［**ウィンドウ**］＞［**すべて閉じる**］を選択して，開いているウィンドウをすべて閉じます。

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

コントローラや 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・ボックスからスクリプトを実行するには，次の手順で行います。

1 ［**スタート**］＞［**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。

＜ **LoadRunner VuGen のパス**＞ ¥bin¥mdrv.exe -usr **script_name** -vugen_win 0

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

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。

注：UNIX マシン上で仮想ユーザの設定をまだ確認していなければ，**vu_verify** を使って確認します。UNIX 仮想ユーザの設定の詳細については，『**LoadRunner インストール・ガイド**』を参照してください。

---

3 **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
```

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，以下のコマンド・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 が成功しないと，cci は実行できません。

-driver ＜ドライバのパス＞：指定されたドライバ・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。たとえば，/bin ディレクトリにある CtLib のドライバは **mdrv** です。このオプションを使って，外部ドライバを指定することができます。

-exec_only：仮想ユーザの .ci ファイルを実行します。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＜ **LoadRunner 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。

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

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

シナリオへの仮想ユーザ・スクリプトの組み込み

スタンドアロン・モードでのスクリプトの実行が成功し，スクリプトが機能することを確認したら，そのスクリプトをシナリオに組み込みます。シナリオには，次の情報が含まれています。

- 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

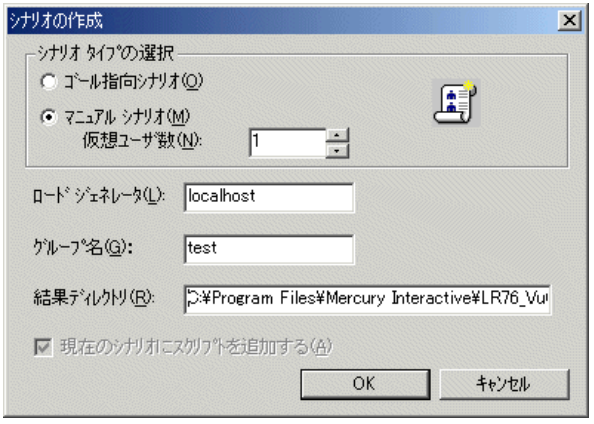
コントローラの使用

通常は，LoadRunner コントローラでシナリオを作成します。現在の仮想ユーザ・スクリプトを含む簡単なシナリオを VuGen から作成することもできます。詳細については，『**LoadRunner コントローラ・ユーザーズ・ガイド**』を参照してください。

VuGen の使用

VuGen でシナリオを作成するには，次の手順で行います。

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，パス変換を行う必要があるかもしれません。

詳細については，『**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 12 章

TestDirector を使ったスクリプトの管理

VuGen を TestDirector と統合することにより，TestDirector を使用して仮想ユーザ・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。

- TestDirector との接続と切断
- TestDirector プロジェクトからスクリプトを開く
- TestDirector プロジェクトへのスクリプトの保存

TestDirector を使ったスクリプトの管理

VuGen は，マーキュリー・インタラクティブの Web ベースのテスト管理ツール，TestDirector と連携して機能します。TestDirector は，仮想ユーザ・スクリプトとシナリオの格納と検索，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 TestDirector 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 TestDirector プロジェクトにアクセスするには，LoadRunner を TestDirector 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。

TestDirector との連携の詳細については，『**TestDirector ユーザーズ・ガイド**』を参照してください。

TestDirector との接続と切断

VuGen と TestDirector の両方を使用して作業している場合，VuGen は TestDirector プロジェクトと通信できます。VuGen と TestDirector プロジェクトの接続と切断は，テスト・プロセス中にいつでもできます。

VuGen の TestDirector への接続

接続プロセスには 2 つの段階があります。最初に，VuGen をローカルまたはリモートの TestDirector Web サーバに接続します。このサーバは，VuGen と TestDirector プロジェクト間の接続を処理します。

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。TestDirector 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 の TestDirector への接続は，次の手順で行います。

1 VuGen で，［**ツール**］＞［**TestDirector への接続**］を選択します。［TestDirector 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，TestDirector 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で表示されます。

4 ［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，TestDirector プロジェクトを選択します。

5 ユーザ名を［**ユーザ名**］ボックスに入力します。

6 パスワードを［**パスワード**］ボックスに入力します。

7 ［**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，［プロジェクト］ボックスにプロジェクトの名前が読み取り専用で表示されます。

8 起動時に TestDirector 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。

9 ［**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いと，VuGen を起動して TestDirector に接続するときに，パスワードの入力を要求されます。

10 ［**閉じる**］をクリックして，［TestDirector への接続］ダイアログ・ボックスを閉じます。

VuGen が現在 TestDirector プロジェクトに接続されているかどうかは，ステータスバーに示されます。

VuGen の TestDirector からの切断

VuGen を，選択した TestDirector プロジェクトや Web サーバから切断できます。

VuGen の TestDirector からの切断は，次の手順で行います。

1 VuGen で，［**ツール**］＞［**TestDirector への接続**］を選択します。［TestDirector 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，［TestDirector への接続］ダイアログ・ボックスを閉じます。

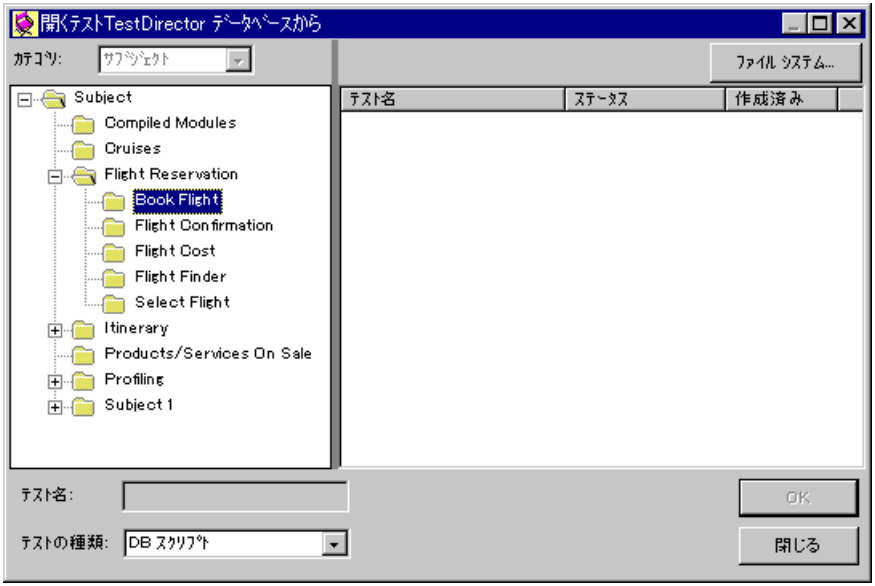
TestDirector プロジェクトからスクリプトを開く

VuGen が TestDirector プロジェクトに接続されている場合は，TestDirector からスクリプトを開くことができます。ファイル・システム内の実際の格納場所ではなく，テスト計画ツリーの中でテストを探します。

TestDirector プロジェクトからスクリプトを開くには，次の手順で行います。

1 TestDirector サーバに接続します。詳細については 170 ページ「VuGen の TestDirector への接続」を参照してください。

2 VuGen で，［**ファイル**］＞［**開く**］を選択するか，［**開く**］ボタンをクリックします。［TestDirector プロジェクトからテストを開く］ダイアログ・ボックスが開き，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・ボックスが表示されます （［テストを開く］ダイアログ・ボックスからは，［**TestDirector**］ボタンをクリックすれば［TestDirector 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。ツリーを展開してサブ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。VuGen が現在 TestDirector プロジェクトに接続されていない状態でそのプロジェクト内のスクリプトを選択した場合，［TestDirector への接続］ダイアログ・ボックスが開きます。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，［**閉じる**］をクリックします。

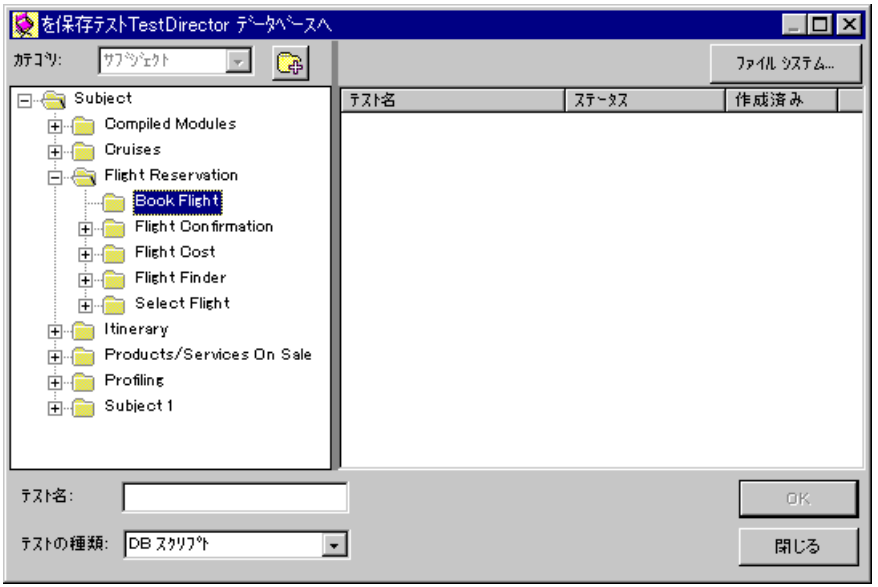
TestDirector プロジェクトへのスクリプトの保存

VuGen が TestDirector 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，わ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すばやく表示できるようになります。

スクリプトの TestDirector プロジェクトへの保存は，次の手順で行います。

1 TestDirector サーバに接続します。詳細については 170 ページ「VuGen の TestDirector への接続」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。［Test Director プロジェクトへテストを保存］ダイアログ・ボックスが開き，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］ダイアログ・ボックスからは，［**TestDirector**］ボタンをクリックすれば［Test Director プロジェクトへテストを保存］ダイアログ・ボックスに戻ることができます）。

3 テスト計画ツリー内の適切なサブジェクトを選択します。ツリーを展開してサブ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。

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わ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。

次回 TestDirector を起動すると，新しいスクリプトが TestDirector のテスト計画ツリーに表示されます。

第 3 部

Java 言語プロトコルでの作業

Java 言語に関連するプロトコルには，RMI-Java，CORBA-Java，EJB，および Jacada タイプがあります。プロトコルの詳細については，それぞれのプロトコルを説明する項を参照してください。「Java 言語プロトコルでの作業」では，すべての Java 仮想ユーザに共通して適用される項目を説明します。

第 13 章

Java 言語仮想ユーザ・スクリプトの記録

VuGen では，Java で書かれた，CORBA，RMI，EJB または Jacada などのプロトコルを使うアプリケーションまたはアプレットを記録できます。VuGen のナビゲーション・ツールを使用して，スクリプトに任意のメソッドを追加することもできます。

本章では，次の項目について説明します。

- 記録を始める前に
- Java 言語仮想ユーザ・スクリプトについて
- パッケージの一部としてのスクリプトの実行
- Java メソッドの表示
- Java メソッドの手作業による挿入
- スクリプト生成の設定

以降の情報は，CORBA-Java，RMI-Java，EJB および Jacada 仮想ユーザ・スクリプトを対象とします。

Java 言語仮想ユーザ・スクリプトの記録について

VuGen を使用して，Java アプリケーションまたはアプレットを記録できます。記録を行うと，VuGen によって，ピュア Java を LoadRunner 固有の Java 関数で拡張したスクリプトが生成されます。記録後，JDK ライブラリまたはユーザ定義クラスを使った標準 Java コードで，そのスクリプトの拡張や変更ができます。

スクリプトの準備ができたら，VuGen からスタンドアロン・モードで実行します。Sun の標準 Java コンパイラ，**javac.exe** によってエラーのないことが確認された後，スクリプトがコンパイルされます。スクリプトが機能することを確認したら，LoadRunner のシナリオに組み込みます。

スクリプトを記録と手作業で拡張する場合，Java 仮想ユーザ・スクリプトに関連したすべてのガイドラインと制限が適用されます。さらに，スクリプトで使用されるクラスはすべて仮想ユーザを実行するマシンにあって，**CLASSPATH** 環境変数で指定されている必要があります。関数の構文とシステムの設定で留意すべき点については，第 24 章「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。

記録中に VuGen でアプレットまたはアプリケーションをロードすると，LoadRunner を使わずにそれらをロードする場合よりも，多少時間がかかります。

VuGen には，Web 用に作成された仮想ユーザ・スクリプトを Java に変換するツールがあります。詳細については，430 ページ「Web 仮想ユーザ・スクリプトの Java への変換」を参照してください。

記録を始める前に

次の手順は，Java 言語仮想ユーザ・スクリプトの記録方法の概要です。

1 **記録するマシンの設定が正しいことを確認します。**

記録を開始する前に，マシンが Java を扱えるように正しく設定されていることを確認します。詳細については，第 24 章「Java 仮想ユーザ・スクリプトのプログラミング」と「Read Me」ファイルを参照してください。

2 **仮想ユーザ・スクリプトを新規作成します。**

プロトコル・タイプ（分散型コンポーネント，EJB，またはミドルウェア）を選択して，使用する仮想ユーザ・タイプを選びます。

3 **スクリプトの記録パラメータとオプションを設定します。**

作業ディレクトリやパスなど，アプレットまたはアプリケーションに対するパラメータを指定します。JVM，シリアル化，相関，レコーダ，デバッグなどの記録オプションを設定することもできます。詳細については，第 14 章「Java 記録オプションの設定」を参照してください。

4 **標準的なユーザ・アクションを記録します。**

スクリプトの記録を開始します。アプレットまたはアプリケーションで一般的な操作をします。VuGen によってアクションが記録され，仮想ユーザ・スクリプトが生成されます。

5 **仮想ユーザ・スクリプトを拡張します。**

LoadRunner 特定の関数を追加して，仮想ユーザ・スクリプトを拡張します。詳細については，第 24 章「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。組み込みの Java Function Navigator（Java 関数ナビゲータ）を使用できます。詳細については，184 ページ「Java メソッドの表示」を参照してください。

6 **仮想ユーザ・スクリプトをパラメータ化します。**

記録された定数をパラメータで置き換えます。文字列の全体または一部をパラメータ化できます。複数の引数を持つ関数には，複数のパラメータを定義できます。詳細については，第 7 章「パラメータの定義」を参照してください。

7 **スクリプトの実行環境の設定を行います。**

仮想ユーザ・スクリプトの実行環境の設定を行います。実行環境の設定では，スクリプト実行時の環境を定義します。Java 固有の実行環境の設定については，第 16 章「Java 実行環境の設定」を参照してください。

8 **仮想ユーザ・スクリプトを保存して実行します。**

VuGen からスクリプトを実行して，実行時の情報を実行ログで確認します。詳細については，第 11 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。

記録手順の詳細については，各仮想ユーザ・タイプの章を参照してください。

Java 言語仮想ユーザ・スクリプトについて

セッションを記録すると，VuGen によってサーバに対するすべての呼び出しが記録され，LoadRunner による拡張を伴うスクリプトが生成されます。これらの関数は，アプリケーションまたはアプレットにおけるユーザのすべてのアクションを表します。スクリプトにはプロパティの設定やネーミング・サービスの初期化（JNDI）など，適切な再生に必要な追加コードも含まれています。

記録されたスクリプトは，次の 3 つのセクションで構成されています。

- インポート
- コード
- 変数

インポート・セクションはスクリプトの先頭にあります。ここにはスクリプトのコンパイルに必要なすべてのパッケージへの参照が含まれます。**コード**・セクションには，Actions クラスと，**init**，**actions**，および **end** メソッド内に記録されたコードが含まれます。**end** メソッドの後の**変数**セクションには，コードで使用される変数のすべての型宣言が含まれます。

記録終了後，スクリプトの関数の修正や，Java 関数または LoadRunner 関数の追加によって，スクリプトを拡張できます。Java 仮想ユーザをスレッドとして実行する場合，スクリプトに追加する Java コードはスレッドセーフでなければなりません。関数の構文の詳細については，「**オンライン関数リファレンス**」（[**ヘルプ**] > [**関数リファレンス**]）を参照してください。さらに，スクリプトを別のパッケージの一部として実行できるように変更することも可能です。詳細については，350 ページ「パッケージの一部としてのスクリプトのコンパイルと実行」を参照してください。

パッケージの一部としてのスクリプトの実行

このセクションは，Jacada タイプのスクリプトには適応されません。

Java 仮想ユーザ・スクリプトを作成または記録するときに，メソッドまたはクラスが保護されているクラスのメソッドを使用する必要が生じる場合があります。そのようなスクリプトをコンパイルすると，メソッドにアクセスできないことを示すコンパイル・エラーを受け取ることになります。

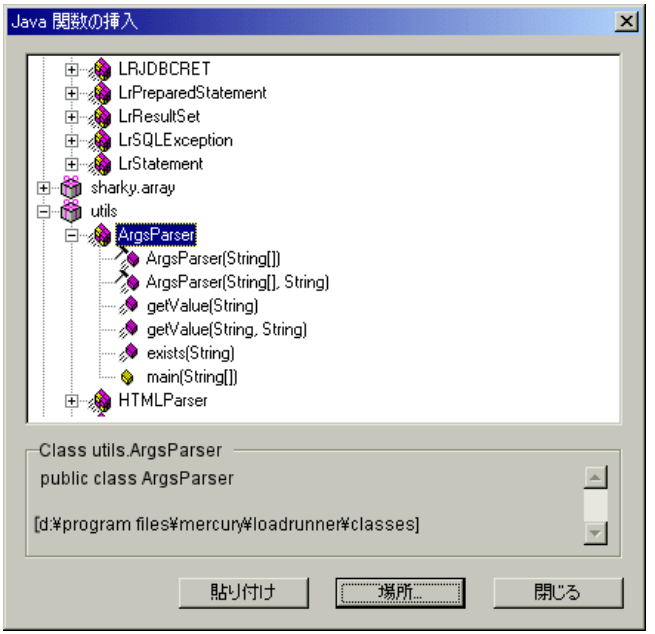
保護されているメソッドを仮想ユーザで使用するには，必要なメソッドのパッケージにその仮想ユーザを追加します。スクリプトの先頭に次の行を追加します。

```
package a.b.c;
```

ここで，a.b.c はディレクトリ階層を表します。VuGen はユーザ・ディレクトリにディレクトリ階層 a/b/c を作成し，そこで **Actions.java** ファイルをコンパイルして，パッケージの一部にします。**package** ステートメントは記録されません。手作業で挿入する必要があります。

Java メソッドの表示

VuGen では，アプリケーションのパッケージに含まれているすべての Java クラスとメソッドを参照できるナビゲータが利用できます。



クラスまたはメソッドをスクリプトに挿入するには，クラスまたはメソッドを選択してスクリプトに貼り付けます。詳細な手順については，186 ページ「Java メソッドの手作業による挿入」を参照してください。

ダイアログ・ボックスの下部には，Java オブジェクトの詳細，プロトタイプ，戻り値，およびパスが表示されます。次の例に示す詳細情報は，**deserialize** メソッドが，文字列と整数の 2 つのパラメータを取る public な静的クラス・メソッドであることを示します。このメソッドは java.lang.object を返し，例外をスローします。

```
public static synchronized java.lang.Object deserialize (java.lang.String,
int) throws Exception
```

次の表に，各種の Java オブジェクトを表すアイコンの説明を示します。

アイコン	項目	例
	パッケージ	java.util
	クラス	public class Hashtable extends java.util.Dictionary implements java.lang.Cloneable, java.io.Serializable
	インタフェース・クラス（灰色のアイコン）	public interface Enumeration
	メソッド	public synchronized java.util.Enumeration keys ()
	静的メソッド（黄色のアイコン）	public static synchronized java.util.TimeZone getTimeZone
	コンストラクタ・メソッド	public void Hashtable ()

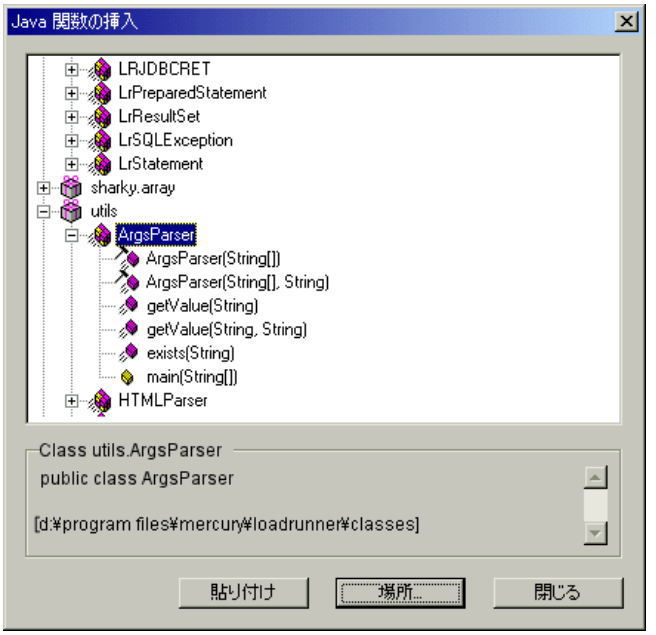
Java メソッドの手作業による挿入

Java 関数ナビゲータを使用して Java 関数の表示やスクリプトへの追加をします。本項の情報は，EJB テスト，RMI-Java，および CORBA-Java 仮想ユーザに適用されます。設定ファイルを変更して，関数の生成についての設定をカスタマイズできます。詳細については，188 ページ「スクリプト生成の設定」を参照してください。

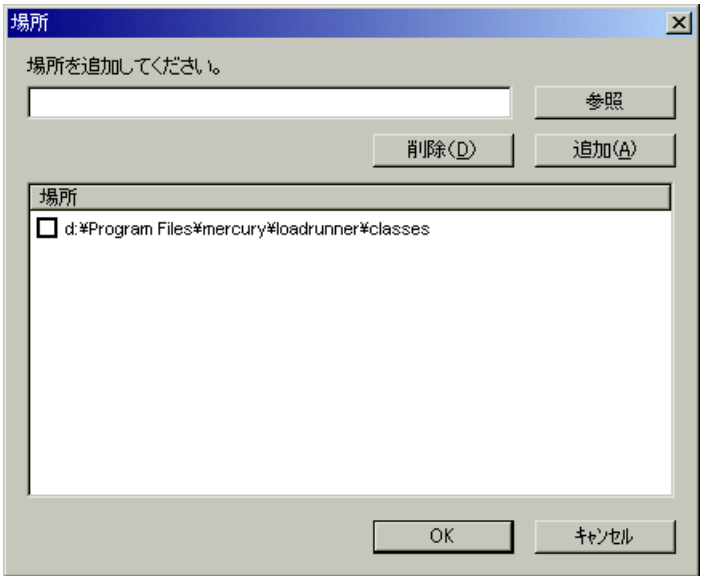
Java 関数の挿入は，次の手順で行います。

1 スクリプトの中で，挿入を行う場所をクリックします。関数の貼り付けを行うと，VuGen によってカーソルの位置に関数が貼り付けられます。

2 ［**挿入**］＞［**Java 関数の挿入**］を選択します。［Java 関数の挿入］ダイアログ・ボックスが表示されます。



3 ［**場所**］をクリックします。［場所］ダイアログ・ボックスが表示されます。標準設定では，CLASSPATH 環境変数に定義されているパスの一覧が表示されます。



4 ［**参照**］をクリックして別のパスまたはアーカイブをリストに追加します。パスを追加するには，［**参照**］＞［**フォルダ**］を選択します。アーカイブ（**jar** または **zip**）を追加するには，［**参照**］＞［**ファイル**］を選択します。フォルダまたはファイルを選択すると，VuGen によって［**場所を追加してください。**］ボックスにその名前が挿入されます。

5 ［**追加**］をクリックして，リストに項目を追加します。

6 追加するパスまたはアーカイブごとに手順 4 と 5 を繰り返します。

7 リスト項目の左にあるチェック・ボックスを選択するかクリアします。選択した項目のメンバのリストが，Java クラス・ナビゲータに表示されます。

8 ［**OK**］をクリックして［位置］ダイアログ・ボックスを閉じると，使用可能なパッケージが表示されます。

9 ナビゲータの各項目の左にあるプラスとマイナスの記号をクリックして，ツリーの分岐の表示と非表示を切り替えます。

10 オブジェクトを選択し，［**貼り付け**］をクリックします。VuGen によってスクリプトのカーソルの位置にオブジェクトが挿入されます。1 つのクラスのすべてのメソッドをスクリプトに貼り付けるには，そのクラスを選択して［**貼り付け**］をクリックします。

11 使用するすべてのメソッドとクラスについて，手順 10 を繰り返します。

12 メソッドのパラメータを変更します。スクリプト生成の設定で **DefaultValues** を **true** に設定しておくと，VuGen によって挿入される標準設定値を使用できます。**DefaultValues** を **false** に設定すると，スクリプトに挿入するすべてのメソッドについてパラメータを追加する必要があります。

次に戻り値を変更します。たとえば，スクリプトで「(String)=LavaVersion.getVersionId();」というステートメントが生成された場合，**(String)** を文字列型変数に置き換えます。

13 import ステートメントや LoadRunner Java 関数（第 24 章「Java 仮想ユーザ・スクリプトのプログラミング」で説明）など，必要な任意のステートメントをスクリプトに追加します。

14 スクリプトを保存して，VuGen から実行します。

スクリプト生成の設定

スクリプトの次の要素について，ナビゲータによるメソッドの追加方法をカスタマイズできます。

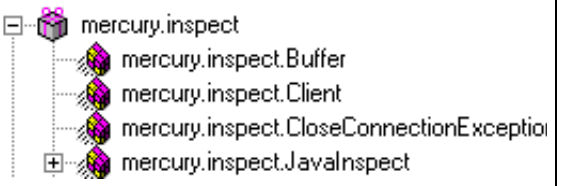
- クラス名のパス
- 自動トランザクション
- 標準のパラメータ値
- クラスの貼り付け

設定を表示するには，LoadRunner の dat ディレクトリにある **jquery.ini** ファイルを開きます。

```
[Display]
FullClassName=False

[Insert]
AutoTransaction=False
DefaultValues=True
CleanClassPaste=False
```

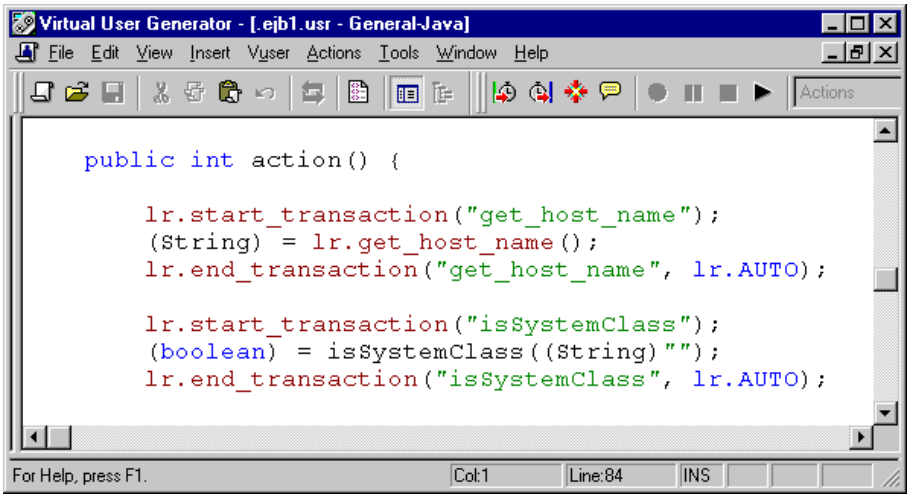
クラス名のパス

FullClassName オプションを有効にすると，Java 関数ナビゲータにパッケージとクラスの完全な名前が表示されます。このオプションは関数がスクリプトにどのように追加されるかには影響しません。ナビゲータでのクラスの表示が変わるだけです。標準設定では，このオプションは false に設定されています。パッケージに多数のクラスが含まれていてパッケージとクラスの名前が同時に見えない場合は，このオプションを有効にします。

FullClassName が有効	FullClassName が無効
	

自動トランザクション

AutoTransaction を有効にすると，スクリプトに貼り付けるすべてのメソッドについて LoadRunner トランザクションが作成されます。このオプションを有効にすると，VuGen によってすべての Java メソッドが自動的に **lr.start_transaction** 関数と **lr.end_transaction** 関数で囲まれます。これにより，各メソッドのパフォーマンスを個別に追跡できます。標準設定では，このオプションは無効になっています。



標準のパラメータ値

DefaultValues を有効にすると，スクリプトに貼り付けるすべてのメソッドが標準設定の値を持ちます。標準設定ではこのオプションは有効で，すべてのオブジェクトについて null が挿入されます。このオプションを無効にした場合は，スクリプトのすべての関数のパラメータ値を手作業で挿入する必要があります。次の表に，**DefaultValues** フラグが有効になっている場合と無効になっている場合を示します。

DefaultValues が有効	DefaultValues が無効

lr.message((String)"");
lr.think_time((int)0);
lr.enable_redirection((boolean)false);
lr.save_data((byte[])null, (String)"");
	lr.message((String));
	lr.think_time((int));
	lr.enable_redirection((boolean));
	lr.save_data((byte[]), (String));

クラスの貼り付け

CleanClassPaste を有効にすると，クラスがエラーなくコンパイルされるように貼り付けられます。つまり，コンストラクタからインスタンスが返され，パラメータに標準の値が設定され，import ステートメントを必要としません。このオプションを使うと，多くの場合，他の修正をせずにスクリプトを実行できます。このオプションが無効の場合（標準設定），パラメータを手作業で定義し，import ステートメントを含める必要があります。この設定が適用されるのは，1 つのメソッドでなく，クラス全体をスクリプトに貼り付ける場合だけです。

次のコードは，**CleanClassPaste** オプションを有効にしてスクリプトに **toString** メソッドを貼り付けた場合を示します。

```
_class.toString();    // 戻り値 : java.lang.String
```

CleanClassPaste オプションが無効の場合，同じメソッドが次のように貼り付けられます。

```
(String) = toString();
```

次のコードは，**CleanClassPaste** オプションを有効にしてスクリプトに **NumInserter** コンストラクタ・メソッドを貼り付けた場合を示します。

```
utils.NumInserter _numinserter = new utils.NumInserter
        ((java.lang.String)"", (java.lang.String)"", (java.lang.String)""...;);
// 戻り値 : void
```

CleanClassPaste オプションを無効にすると，同じメソッドが次のように貼り付けられます。

```
new utils.NumInserter((String)"", (String)"", (String)"",...);
```

第 14 章

Java 記録オプションの設定

VuGen では，CORBA，RMI，または EJB アプリケーションの記録方法を制御できます。標準の記録オプションを使用することも，必要に応じてそれらをカスタマイズすることもできます。

本章では，以下の項目について説明します。

- Java 仮想マシン（JVM）記録オプション
- クラスパス記録オプションの設定
- レコーダ・オプション
- シリアル化オプション
- 相関オプション
- デバッグ・オプション

以降の情報は，CORBA-Java，RMI-Java，EJB 仮想ユーザ・スクリプトにのみ適用されます。

Java 記録オプションの設定について

VuGen では，CORBA（Common Object Request Broker Architecture）または RMI（Remote Method Invocation）に対応した Java アプリケーションまたはアプレットを記録できます。EJB テストの記録の詳細については，第 46 章「EJB テストの実行」を参照してください。

記録をする前に，Java 仮想マシン（JVM），およびコード生成段階で使われる記録オプションを設定します。記録オプションの設定は，必須ではありません。設定しない場合は，標準設定の値が使用されます。

本章で説明するオプションは，以前は **mercury.properties** ファイルに変更を加えることにより設定していました。

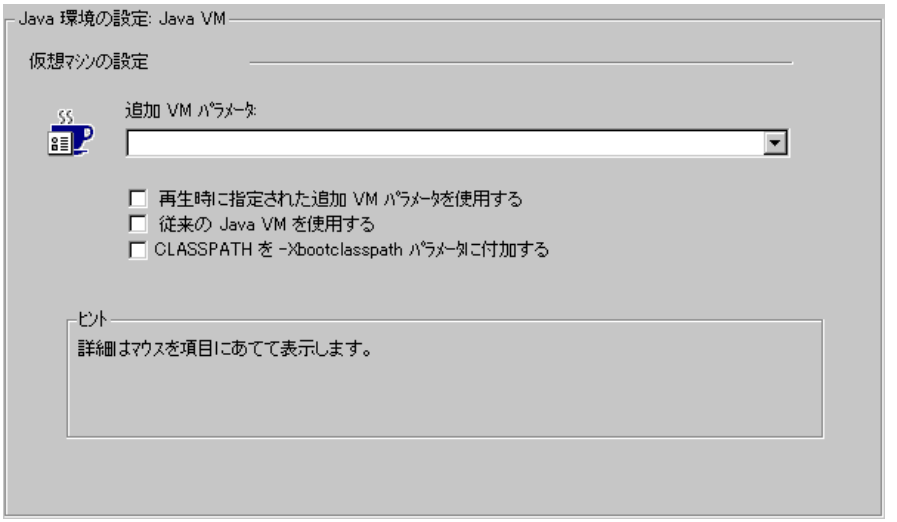
次の項目について記録オプションの設定が可能です。

- Java 仮想マシン（JVM）記録オプション
- クラスパス記録オプションの設定
- レコーダ・オプション
- シリアル化オプション
- 相関オプション
- デバッグ・オプション

Java 仮想マシン（JVM）記録オプション

Java VM オプションは，Java アプリケーションの記録時に使用する付加的なパラメータを指定します。

仮想ユーザを記録する際，VuGen によって **Xbootclasspath** 変数に，標準のパラメータが設定されます。このダイアログ・ボックスを使って，**Xbootclasspath** に対して別のパラメータを設定すると，標準のパラメータに代えて，指定したコマンド・パラメータが使用されます。



また，単一のクラスパス文字列を作成するために，VuGen に対して，クラスパスを **Xbootclasspath** の前に追加する（文字列を挿入する）ように指定することもできます。

標準設定では，VuGen は記録中に従来型の VM を使用します。VuGen を別の仮想マシン（Sun の Java Hotspot VM）を使用するように設定することもできます。

Java 仮想マシンの記録オプションを設定するには，次の手順で行います。

1 ［記録開始］ダイアログ・ボックスの［**オプション**］をクリックします。記録オプションの［**Java 環境の設定：Java VM**］ノードを選択します。

2 ［**追加 VM パラメータ**］ボックスに，一連の Java コマンド・ライン・パラメータを指定します。任意の Java VM の引数をパラメータとして指定できます。よく使用する引数としては，デバッグ・フラグ（**-verbose**）や，メモリの設定（**-ms**，**-mx**）があります。Java VM フラグの詳細については，JVM のマニュアルを参照してください。さらに，**-D** フラグの形式で Java アプリケーションに対してプロパティを渡すこともできます。

VuGen では，**-Xbootclasspath** 変数（JDK 1.2 以上）に対して，標準のパラメータが自動的に設定されます。**Xbootclasspath** に対して，追加のパラメータとしてパラメータの値を指定すると，VuGen によって標準設定の値の代わりにその設定が使用されます。

3 再生時に同じ追加 VM パラメータを使用するには，［**再生時に指定された追加 VM パラメータを使用する**］チェック・ボックスを選択します。

4 従来型の VM を使用するには，［**従来の Java VM を使用する**］チェック・ボックスを選択します（標準設定）。別の VM（Sun の Java HotSpot）を使用するには，このチェック・ボックスをクリアします。

5 クラスパスを **Xbootclasspath** の前に追加する（つまり，文字列を前置する）には，［**CLASSPATH を -Xbootclasspath パラメータに付加する**］チェック・ボックスを選択します。

6 ［**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。

クラスパス記録オプションの設定

Classpath セクションでは，システムのクラスパス環境変数に含まれていない追加のクラスの場所を指定できます。これらのクラスは，Java アプリケーションの実行や，正しい記録を保障するのに必要な場合があります。

コンピュータまたはネットワークから必要なクラスを検索し，特定のテストの場合にそのクラスを無効にすることができます。また，クラスの順序を変更して，クラスパスのエントリを操作することもできます。

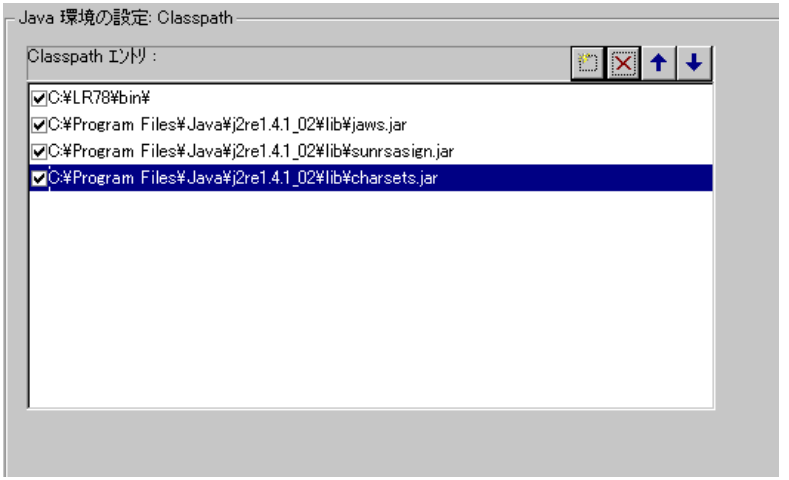
クラスパス記録オプションを設定するには，次の手順で行います。

1 ［記録開始］ダイアログ・ボックスの［**オプション**］をクリックします。記録オプションの［**Java 環境の設定：Classpath**］ノードを選択します。

2 クラスパスをリストに追加するには次の手順で行います。

［クラスパスの追加］ボタンをクリックします。VuGen によって新しい行がクラスパス・リストに追加されます。

クラスが含まれている **jar**，**zip**，または他のアーカイブ・ファイルのパスまたは名前を入力します。または，フィールドの右にある参照ボタンをクリックして，対象ファイルを選択します。VuGen によって，クラスパスのリストに新しい場所が，ステータスが有効な状態で追加されます。



3 エントリを恒久的に削除するには，エントリを選択して［削除］ボタンをクリックします。

4 特定のテストでエントリを一時的に無効にするには，エントリの左側のチェック・ボックスをクリアします。

5 一覧の中でエントリを下方向に移動するには，エントリを選択し，下向き矢印ボタンをクリックします。

6 一覧の中でクラスパス・エントリを上方向に移動するには，エントリを選択し，上向き矢印ボタンをクリックします。

7 [**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。

レコーダ・オプション

[**レコーダ オプション**] は，VuGen による仮想ユーザ・スクリプトの生成の基準となります。次の項目についてオプションの設定が可能です。

- 一般オプション
- 記録ログ
- スタイル・オプション
- バイト形式オプション

一般オプション

［**リターン値の記録**］：各呼び出しの戻り値を示すコメントをスクリプト内に生成します（標準設定では無効）。

［**進行状況メッセージを記録する**］：再生の進行を追うことができるように，各呼び出しの前に lr.log_message 関数を記録します（標準設定では無効）。

［**思考遅延時間を記録する**］：思考遅延時間を記録し，スクリプトに思考遅延時間関数 lr.think_time を加えます（標準設定では有効）。

［**例外処理を記録する**］：例外が発生したときに，その呼び出しを「try/catch」ブロックに入れます（標準設定では有効）。

［**関数チェックを挿入する**］：再生中に受け取った戻り値を，記録中に生成された戻り値の期待値と比較する検証コードを挿入します。このオプションは，プリミティブ型の戻り値にのみ適用されます（標準設定では無効）。

［**LR API を使用する**］：スクリプトに LR API 関数を含めます。VuGen の外部でスクリプトを使用する場合は，このオプションを無効にして，遅延思考時間やその他の定数など，すべての LR API 関数を削除します（標準設定では有効）。

［**出力を転送する**］：Java アプリケーションの **Stdout** 出力と **Stderr** 出力をファイルにリダイレクトします（標準設定では無効）。

［**拡張子リスト**］：サポートされている拡張子のリスト。各拡張子には，固有のフック・ファイルがあります。追加の拡張子を指定するには，その拡張子を標準の拡張子のリストに追加します。リストに拡張子を追加した場合は，仮想ユーザ・スクリプトがそのフック・ファイルを使用できるようにします。標準の拡張子は，JNDI，JMS，EJB です。

［**_JAVA_OPTIONS フラグを使用する**］：Version 1.2 以降の JVM に対して，使いたい JVM パラメータを指定する _JAVA_OPTION 環境変数を使用するようにします（標準設定では無効）。

記録ログ

[**記録ログを生成する**]：出力ウィンドウの［記録］タブに表示される記録ログを生成します。このオプションを無効にすると，パフォーマンスが向上する可能性はありますが，記録中は出力ウィンドウに情報が出力されなくなります（標準設定では有効）。

[**変数情報を生成する**]：変数の内部値を記録ログに書き込みます。このオプションを有効にするとパフォーマンスが低下することがあります（標準設定では無効）。

[**詳細レベル**]：配列型のパラメータまたは戻り値を記録する場合の，ログに表示する配列要素の数です。標準設定のレベルは 5 です。

スタイル・オプション

[**ブロック セマンティックスを使用する**]：各呼び出しを中括弧（{ }）で囲むことによって，それぞれを独立のスコープに置きます。このオプションが無効な場合，呼び出しごとではなく，Action メソッド全体が中括弧で囲まれます（標準設定では無効）。

[**アンダースコア付きの変数名を使用する**]：スクリプトで生成されたすべての変数の前に，プレフィクスとしてアンダースコアを付けます。これは，同じ名前のパッケージとの衝突を防ぐために必要です（標準設定では有効）。

[**行の最大長**]：記録される行の最大の長さ。記録された行がこの値を超えると，それ以降は切り捨てられます。VuGen では，コードの整合性を保つために引用符や関数のパラメータなどが整合性を考慮して，切り捨てられます。標準設定の長さは 1000 文字です。指定できる最大の長さは 30000 文字です。

[**アクションの最大長**]：アクション・メソッドの最大サイズです。標準設定の長さは 3000 文字です。アクション・メソッドがこの値を超える場合，より小さいサイズのアクション・メソッドに分割されます。

[**コメント行の内容**]：指定した文字列のいずれか 1 つを含むスクリプト行をすべてコメントアウトします。複数の文字列を指定するには，項目をカンマで区切ります。標準設定では，<undefined> を含む文字列がある行をコメントアウトします。

[**削除する行の内容**]：指定した文字列のいずれか 1 つを含むスクリプト行をすべて削除します。複数の文字列を指定するには，項目をカンマで区切ります。この機能は，テストの目的に応じてスクリプトをカスタマイズするときに役立ちます。

バイト形式オプション

［**バイトを文字として扱う**］：可読文字を，必要なキャスティングを行うことによって，バイトまたは 16 進形式ではなく文字として表示します（標準設定では有効）。

［**暗示的キャスト**］：すべての呼び出しに自動的にキャスティングを適用します。このオプションを有効にすると，記録された呼び出しに対してキャスティングが追加されません。つまり，コンパイラが暗黙的にその処理を行います。このオプションを無効にすると，VuGen によって呼び出しに対してキャスティングが追加されるので，スクリプトが長くなります（標準設定では無効）。

［**解読不可能な文字列をバイトとして扱う**］：不可読文字を含む文字列をバイトの配列として表します。このオプションは，パラメータとして呼び出しに渡される文字列に適用されます（標準設定では有効）。

［**バイト配列形式**］：スクリプト内のバイト配列の形式で，［**標準**］，［**展開済みシリアル化オブジェクト**］，または［**未展開シリアル化オブジェクト**］から選択できます。非常に長いバイト配列を記録する場合は，シリアル化オブジェクト・オプションのいずれかを使用します。標準設定は［標準］です。

［**システム プロパティを無視する**］：EJB プロパティの記録時に，指定したシステム・プロパティをフィルタリングによって除外します。

Java 記録オプションを設定するには，次の手順で行います。

1 ［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，［**記録プロパティ：レコーダ オプション**］ノードを選択します。

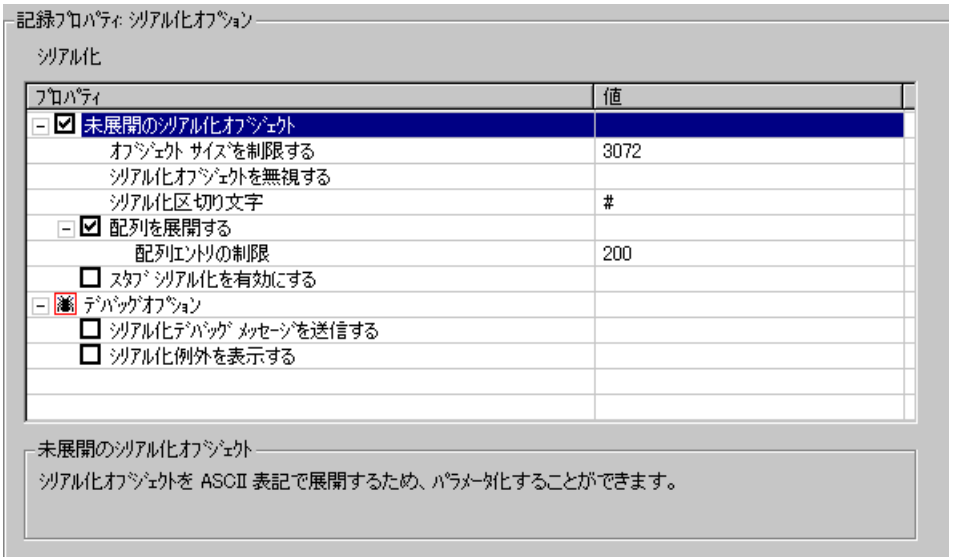
2 必要なオプションを設定します。チェック・ボックス付きのオプションには，オプションの横のチェック・ボックスを選択またはクリアします。数字や文字列が必要なオプションには，必要な値を入力します。

3 すべてのオプションを標準の値に設定するには，［**標準設定値を使用**］をクリックします。

4 ［**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。

シリアル化オプション

［**シリアル化オプション**］では，要素のシリアル化の方法を制御できます。シリアル化の概要については，214 ページ「シリアル化メカニズムの使用」を参照してください。次のオプションが使用できます。



［**未展開のシリアル化オブジェクト**］：シリアライズされたオブジェクトを ASCII 形式に展開します。このオプションを指定すると，パラメータ化を行うために，オブジェクトの ASCII 値を表示できます（標準設定では有効）。

［**オブジェクト サイズを制限する**］：シリアル化可能なオブジェクトのサイズを指定した値に制限します。指定した値を超えるオブジェクトは，ASCII 形式でスクリプト内に表現されません。標準設定の値は 3072 です。

［**シリアル化区切り文字**］：ASCII 形式のオブジェクトで要素を区切る区切り文字を示します。VuGen では，これらの区切り文字に囲まれた文字列のみがパラメータ化されます。標準設定は「#」です。

［**配列を展開する**］：シリアル化されたオブジェクトの配列の要素を ASCII 形式に展開します。このオプションを無効にし，オブジェクトに配列が含まれている場合，オブジェクトは展開されません。標準設定では，このオプションは有効になっています。つまり，シリアル化解除されたオブジェクトはすべて展開されます。

［**配列エントリの制限**］：レコーダによって，指定した数より多くの要素が含まれる配列が展開されないようにします。標準設定の値は 200 です。

［**スタブ シリアル化を有効にする**］：シリアル化しなければ <undefined> となる相関されなかったスタブ・オブジェクトをシリアル化します。このコードを新しいサーバ・コンテキストで再生するには，再記録が必要となる場合があります（標準設定では無効）。

［デバッグ オプション］

［**シリアル化デバッグ メッセージを送信する**］：シリアル化メカニズムから得られたデバッグ情報を出力します（標準設定では無効）。

［**シリアル化例外を表示する**］：シリアル化のすべての例外をログに示します（標準設定では無効）。

シリアル化のオプションを設定するには，次の手順で行います。

1 ［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，［**記録プロパティ：シリアル化オプション**］ノードを選択します。

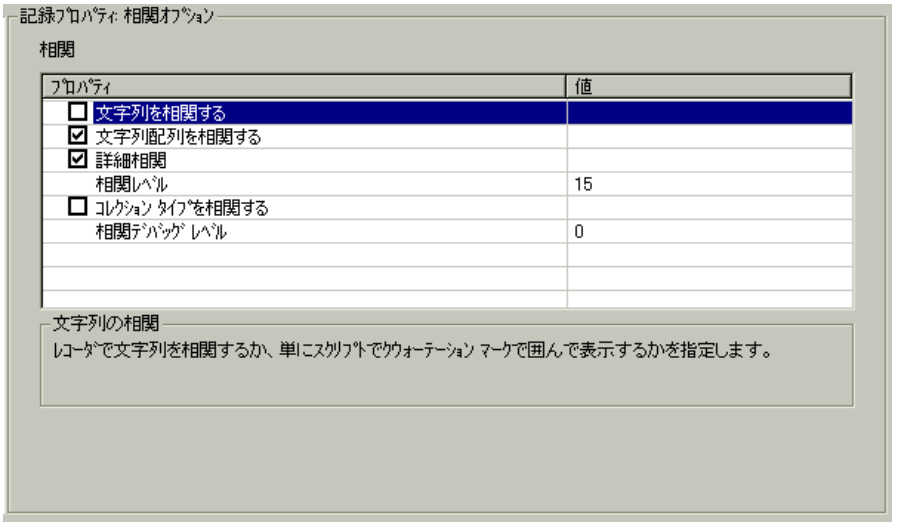
2 必要なオプションを設定します。

3 すべてのオプションを標準の値に設定するには，［**標準設定値を使用**］をクリックします。

4 ［**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。

相関オプション

［**相関オプション**］では，VuGen で自動相関を行うかどうかを指定し，自動相関の範囲を制御します。相関については，第 15 章「Java スクリプトの相関」を参照してください。次のオプションを使用できます。



［**文字列を相関する**］：相関が必要なすべての文字列を相関します。このオプションが無効の場合，相関が必要な文字列は引用符で囲んでスクリプトに出力されます（標準設定では無効）。

［**文字配列を相関する**］：文字配列内のテキストを相関します（標準設定では無効）。

［**詳細相関**］：CORBA コンテナの構造要素と配列で深い相関を有効にします（標準設定では有効）。

［**相関レベル**］：相関のレベルの深さを，スキャン対象の内部コンテナの数として指定します（標準設定は 15）。

［**コレクション タイプを相関する**］：JDK 1.2 以降の Collection クラスのオブジェクトを相関します（標準設定では無効）。

［**相関デバッグ レベル**］：相関に関連するデバッグ情報をログに送ります。0 から 5 の間の値を指定します（標準設定は 0 です。相関デバッグ情報がないことを表します）。

相関のオプションを設定するには，次の手順で行います。

1 ［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，［**記録プロパティ：相関オプション**］ノードを選択します。

2 必要なオプションを有効にするか，値が必要なオプションに値を入力します。

3 すべてのオプションを標準の値に設定するには，［**標準設定値を使用**］をクリックします。

4 ［**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。

デバッグ・オプション

［**デバッグ オプション**］では，記録中に生成するデバッグ情報のレベルを指定します。次のオプションが使用できます。

一般オプション

［**一般デバッグ オプションを有効にする**］：一般デバッグ・オプションを有効にします。デバッグ・オプションには，［**クラスをダンプする**］，［**フックのデバッグ レベル**］，［**スタック トレースを有効にする**］，［**トレース サポートを有効にする**］があります。このオプションを有効にすると，最初の［**スタック トレース**］オプションが無効でも，VuGen によってスタック・トレースが実行されます。アプリケーション内でフックが掛けられた部分に対するコンテキストを特定するには，「**クラスをダンプする**」とともに［**スタック トレースを有効にする**］を使用します。このトレースは，追加のフックを配置する場所を判断するときに役立ちます（標準設定では無効）。

［**スタック トレースを有効にする**］：スタック・トレースにすべての呼び出しを記録します。この設定によって，記録されたすべての関数について Java スタック・トレースが作成されます。このオプションは［クラスをダンプする］と組み合わせて使い，アプリケーション内のフックされている部分のコンテキストを調べます。このトレースは，パラメータが相関されない場所や，追加のフックを配置する場所を判断するのに役立ちます。ただし，このオプションを有効にすると，アプリケーションの動作が遅くなります（標準設定では無効）。

［**スタック トレースの制限を指定する**］：スタックに格納される呼び出しの最大数。スタック・トレースが有効になっているときに，呼び出しの数が指定した値を超えると，以降のスタック・トレースが切り捨てられます。標準設定の値は 20 です。

［**トレース サポートを有効にする**］：主要なサポート呼び出しすべてをトレースし，Vuser ディレクトリの **Tracer.log** ファイルに書き込みます（標準設定では無効）。

［**進行状況ウィンドウを表示する**］：マーキュリー製品の進行状況を示すウィンドウを有効にします（標準設定では有効）。

［**クラスローダをデバッグする**］：非システム ClassLoader サポート用にデバッグ情報を出力します（標準設定では無効）。

［**スレッドを同期化する**］：マルチ・スレッド・アプリケーションの場合に，各スレッド間の同期をとるようにします（標準設定では無効）。

［**計算の概要を表示する**］：記録されたすべてのオブジェクトのダイジェストを生成します（標準設定では無効）。

［**概要から除外する**］：ダイジェスト計算に含めないオブジェクトの一覧です。

［**デバッグ変数を定義する**］：<undefined> 変数をそれぞれのタイプにキャスティングします。また，インタフェースでもある変数セクションの各変数には，元のタイプを示すコメントが付加されます（標準設定では無効）。

［**フックを指定する**］：呼び出しのきっかけとなったフックを示す文字列をスクリプトの呼び出しの前に挿入します（標準設定では無効）。

［**スレッドを指定する**］：呼び出しが実行されるスレッドを示す文字列をスクリプトの呼び出しの前に挿入します。

フック・オプション

［**フックのデバッグ レベル**］：レコーダ内部からのフック・デバッグ情報出力のレベルです。レベル 0 はデバッグ情報を出力しないことを示します。

［**クラスを無視する**］：無視するクラスの一覧です。指定した文字列を含むすべてのクラスが，フック・メカニズムから除外されます。

［**出力を転送する**］：フック・メカニズムからの出力のリダイレクト先を指定します。［Console］，個別の［File］，または［Debug］ファイルが選択できます。標準設定は［Console］です。

［**メソッドを Public として定義する**］：フックしたメソッドを public メソッドにします（標準設定では無効）。

［**クラスを Public として定義する**］：フックしたクラスを public にします（標準設定では有効）。

［**クラス フックをログ記録する**］：フックの前と後に，すべてのクラスを表す文字列表現を含むログ・ファイルを作成します。このオプションは，パフォーマンスを大幅に低下させるため，デバッグを集中的に行う場合のみ使用してください。（標準設定では無効）。

［**特定のクラス フックをログ記録する**］：フックのログ・ファイルを生成するクラスの一覧です。クラスを指定しない場合は，すべてのクラスがログに出力されます。

クラスのダンプ・オプション

［**クラスをダンプする**］：ロードしたクラスを Vuser ディレクトリにダンプします（標準設定では無効）。

［**ダンプするクラス**］：フックの後にダンプするクラスの一覧です。指定した文字列が 1 つでも含まれるクラスはすべてダンプされます。クラスを指定しない場合は，すべてのクラスがダンプされます。

［**クラスのダンプ サフィックス**］：ダンプしたクラス名に付加する接尾辞です。標準設定値は _DUMP です。

［**クラスのダンプ ディレクトリ**］：クラスのダンプ先のディレクトリです。

［**全クラスをダンプする**］：すべてのクラスを 1 つのディレクトリにダンプし，各クラス名の先頭にパッケージの全体を付加します。このオプションが無効の場合は，ディレクトリ階層が作成されます（標準設定では無効）。

デバッグ・オプションを設定するには，次の手順で行います。

1 ［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，［**記録プロパティ：デバッグ オプション**］ノードを選択します。

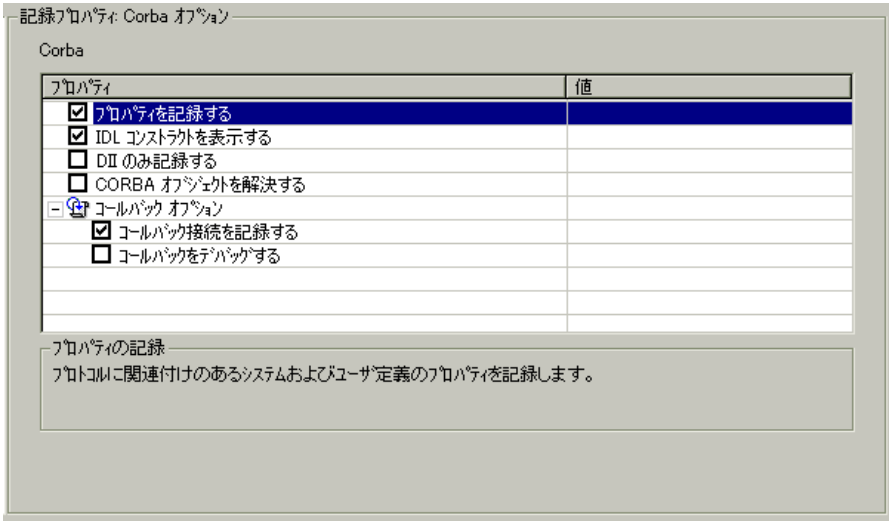
2 必要なオプションを有効にするか，値が必要なオプションに値を入力します。

3 すべてのオプションを標準の値に設定するには，［**標準設定値を使用**］をクリックします。

4 ［**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。

CORBA オプション

次のオプションは Corba-Java プロトコルを対象としています。このオプションでは，CORBA 固有の記録プロパティといくつかのコールバック・オプションを設定できます。次のオプションが使用できます。



［**プロパティを記録する**］：プロトコルに関連するシステム・プロパティとユーザ定義のプロパティを記録します。標準設定では，このオプションは有効になっています。

［**IDL コンストラクトを表示する**］：パラメータを CORBA の呼び出しに対して渡すために使用される，インタフェース定義言語（IDL）の構成要素を表示します。標準設定では，このオプションは有効になっています。

［**DLL のみ記録する**］：DLL レベルのみで記録するよう VuGen に指示します。標準設定では，このオプションは有効になっています。

［**CORBA オブジェクトを解決する**］：相関によって，CORBA オブジェクトが解決できなかった場合に，バイナリ・データを使って解決します。標準設定では，このオプションは有効になっています。

コールバック・オプション

［**コールバック接続を記録する**］：各コールバック・オブジェクトについて，ORB への接続のための接続ステートメントを生成させます。標準設定では，このオプションは有効になっています。

［**コールバックをデバッグする**］：コールバック時にデバッグ情報が生成されるようにします。標準設定では，このオプションは有効になっています。

Corba オプションを設定するには，次の手順で行います。

1 ［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，［**記録プロパティ：Corba オプション**］ノードを選択します。

2 必要なオプションを有効または無効にします。

3 すべてのオプションを標準の値に設定するには，［**標準設定値を使用**］をクリックします。

4 ［**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。

第 15 章

Java スクリプトの相関

VuGen の相関機能を使うと，あるステートメントの結果を別のステートメントへの入力として使用して，Java 仮想ユーザ関数同士をリンクさせることができます。

本章では，次の項目について説明します。

- 標準的な相関
- 詳細相関
- 文字列の相関
- シリアル化メカニズムの使用

以降の情報は，CORBA-Java および RMI-Java による 仮想ユーザ・スクリプトを対象とします。

Java スクリプトの相関について

Java コードを含む仮想ユーザ・スクリプトには，多くの場合，動的データが含まれています。CORBA または RMI による仮想ユーザ・スクリプトを記録すると，動的データはスクリプトに記録されますが，再生時には再使用することができません。仮想ユーザの実行中にエラーが発生した場合は，スクリプト内でエラーが発生した場所を調べます。多くの場合は，相関を行って，あるステートメントの結果を別のステートメントへの入力として使用できるようにすることで，問題を解決できます。

VuGen の CORBA レコーダは，生成されたスクリプト内のステートメントを自動的に相関させようとします。相関の対象は，Java オブジェクトに限ります。CORBA レコーダが記録中に Java のプリミティブ（byte，character，boolean，integer，float，double，short，long）を見つけると，引数の値が変数に割り当てられていない状態でスクリプトに示されます。VuGen では，オブジェクト，オブジェクトの配列，プリミティブの配列がすべて自動的に相関されます。Java の配列と文字列もオブジェクトと見なされます。

VuGen では，複数の相関レベル（標準，詳細，文字列）を使用します。［記録オプション］から相関を有効にしたり無効にしたりできます。前述の方法でスクリプトを処理できない場合は，シリアル化というもう 1 つの方法を使用してスクリプトを処理できます。詳細については，214 ページ「シリアル化メカニズムの使用」を参照してください。

標準的な相関

標準的な相関は，記録中にオブジェクトの配列，ベクトル，コンテナ構成要素を除く単純なオブジェクトを対象に実行される自動相関を指します。

記録されたアプリケーションが，オブジェクトを返すメソッドを起動すると，VuGen の相関メカニズムによってこれらのオブジェクトが記録されます。スクリプトを実行すると，生成されたオブジェクトと記録されたオブジェクトが比較されます。オブジェクトが一致すると，同じオブジェクトが使用されます。次の例は，2 つの CORBA オブジェクト **my_bank** と **my_account** を示しています。最初のオブジェクト **my_bank** が呼び出され，2 つ目のオブジェクト **my_account** が相関され，コードの最後の行でパラメータとして渡されます。

```
public class Actions {

  // Public 関数：init
     public int init() throws Throwable {

      Bank my_bank = bankHelper.bind("bank", "shunra");
      Account my_account = accountHelper.bind("account","shunra");

      my_bank.remove_account(my_account);
     }
:
}
```

詳細相関

詳細相関または「**深い**」相関は，オブジェクトの配列や CORBA コンテナ構成要素などの複雑なオブジェクトを対象に，記録中に実行される自動相関を指します。

詳細相関メカニズムは，CORBA の構成要素（構造体，共用体，シーケンス，配列，ホルダ，「any」）をコンテナとして処理します。これによって，コンテナ，追加のオブジェクト，または他の各種コンテナの内部メンバを参照できます。オブジェクトは，呼び出されるかパラメータとして渡されると，コンテナの内部メンバとも比較されます。

次の例では，VuGen によって配列の要素を参照する詳細相関が実行されています。**remove_account** オブジェクトが account オブジェクトをパラメータとして受け取ります。相関メカニズムによって記録中に，返された配列 **my_accounts** が検索され，その 6 番目の要素をパラメータとして渡すことが検出されています。

```
public class Actions {

  // Public 関数：init
  public int init() throws Throwable {

        my_banks[] = bankHelper.bind("banks", "shunra");
        my_accounts[] = accountHelper.bind("accounts","shunra");

        my_banks[2].remove_account(my_accounts[6]);
   }
:
}
```

次のコードは，詳細相関の別の例を示します。スクリプトによって **address** 型の引数を受け取った **send_letter** オブジェクトが呼び出されています。相関メカニズムによって，**my_accounts** 配列の 6 番目の要素の内部メンバ **address** が取得されます。

```
public class Actions {

  // Public 関数 ; init
  public int init() throws Throwable {

    my_banks = bankHelper.bind("bank", "shunra");
    my_accounts = accountHelper.bind("account", "shunra");

        my_banks[2].send_letter(my_accounts[6].address);
  }
:
}
```

文字列の相関

文字列の相関は，記録された値を実際の文字列または変数として表すことを指します。文字列の相関を無効にすると（標準設定），記録された実際の文字列の値が，スクリプト内で明示されます。文字列の相関を有効にすると，各文字列の代わりに変数が作成され，スクリプトの以降の部分でこの変数を使用できるようになります。

次のコードでは，文字列の相関が有効になっており，**get_id** メソッドから返された値を，スクリプトのそれ以降の部分で使用できるように文字列（**string**）型変数に格納します。

```
public class Actions {

  // Public 関数：init
  public int init() throws Throwable {

    my_bank = bankHelper.bind("bank", "shunra");
    my_account1 = accountHelper.bind("account1", "shunra");
    my_account2 = accountHelper.bind("account2", "shunra");

    string = my_account1.get_id();
    string2 = my_account2.get_id();
    my_bank.transfer_money(string, string2);
  }
:
}
```

相関メソッドの設定は，［記録オプション］の［**相関**］タブで行います。

［**文字列の相関**］：記録中にスクリプト内の文字列を相関させます。このオプションを無効にすると，実際に記録された値はスクリプトに引用符付きで含まれます。このオプションが無効になると，他のすべての相関オプションが無視されます（標準設定では無効）。

［**文字配列の相関**］：記録中に，文字列配列内の文字列を相関させます。このオプションを無効にすると，配列内の文字列は相関されず，実際の値がスクリプトに挿入されます（標準設定では有効）。

［**詳細相関**］：配列，CORBA コンテナの構成要素および配列といった複雑なオブジェクトを対象に相関させます。このタイプの相関は，「**深い**」相関としても知られています（標準設定では有効）。

［**相関レベル**］：複雑な相関のレベルの深さ（検索する内部コンテナの数）を指定します。

［**コレクションタイプを相関する**］：JDK 1.2 以上の Collection クラスに含まれるオブジェクトを相関させます（標準設定では無効）。

シリアル化メカニズムの使用

RMI および CORBA（一部）では，クライアントの AUT によって，**java.io.serializable** インタフェースに基づく Java オブジェクトの新しいインスタンスが作成されます。サーバの呼び出しに対して，このインスタンスがパラメータとして渡されます。次のコードでは，インスタンス **p** が作成され，パラメータとして渡されています。

```
// AUT コード：
java.awt.Point p = new java.awt.Point(3,7);
map.set_point(p);
:
```

前のどの呼び出しからもオブジェクトが返されなかったので，自動相関メカニズムはここでは適用されません。この場合，VuGen ではシリアル化メカニズムが実行され，パラメータとして渡されるオブジェクトが格納されます。この情報はユーザ・ディレクトリのバイナリ・データ・ファイルに保存されます。その他のパラメータは，新しいバイナリ・データ・ファイルとして保存され，順番に番号が付けられます。VuGen によって次のコードが生成されます。

```
public class Actions {

  // Public 関数：init
     public int init() throws Throwable {
         java.awt.Point p = (java.awt.Point)lr.deserialize(0, false);
         map.set_point(p);
     }
:
}
```

lr.deserialize に渡された整数は，仮想ユーザ・ディレクトリのバイナリ・データ・ファイルの番号です。

記録された値をパラメータ化するには，public メソッドの **setLocation** メソッドを使用します。詳細については，JDK の関数リファレンスを参照してください。次の例では，**setLocation** メソッドを使って，オブジェクト **p** の値を設定しています。

```
public class Actions {

 // Public 関数：init
    public int init() throws Throwable {
       java.awt.Point p = (java.awt.Point)lr.deserialize(0, false);
       p.setLocation(2,9);
       map.set_point(p);
    }
:
:
}
```

インスタンスによっては，public メソッドの **setLocation** を適用できないものもあります。代わりに，クラスのアクセッサ・メソッドである get または set メソッドを使う API を使用できます。AUT のクラスに get および set メソッドがないか，プライベート・メソッドを使っている場合や，クラスの API に慣れていない場合は，VuGen に組み込まれているシリアル化メカニズムを使用できます。このメカニズムによって，オブジェクトを ASCII 表現に展開しスクリプトを手作業でパラメータ化できます。このメカニズムを有効にするには，［記録オプション］ダイアログ・ボックスで設定します。詳細については，第 14 章「Java 記録オプションの設定」を参照してください。

VuGen によって，データが デシリアル化されます。つまり複雑なデータ構造を一連の文字列として表示する **lr.deserialize** メソッドを生成します。データ構造体をコンポーネントに分解すると，パラメータ化が簡単になります。**lr.deserialize** メソッドは文字列と整数の 2 つの引数を受け取ります。文字列は，再生中に置換されるパラメータの値です。整数は，ロードするバイナリ・ファイルの番号です。

スクリプトのオブジェクトを展開しないように，［**シリアル化オブジェクトの展開**］チェック・ボックスをクリアすると，**lr.deserialize** メソッドに引数を渡すことによって，シリアル化メカニズムを制御できます。最初の引数は整数で，ロードするバイナリ・ファイルの数を示しています。2 つ目の整数はブール値です。

true	VuGen のシリアル化メカニズムを使用します。
false	標準の Java シリアル化メカニズムを使用します。

次のコードは，シリアル化メカニズムが有効になっている状態で生成されたスクリプトを示します。

```
public class Actions {

 // Public 関数：init
    public int init() throws Throwable {
        _string = "java.awt.Point __CURRENT_OBJECT = {" +
                  "int x = "#5#" +
                  "int y = "#8#" +
        "}";
        java.awt.Point p = (java.awt.Point)lr.deserialize(_string,0);
        map.set_point(p);
    }
:
}
```

文字列値は，区切り文字の間に置かれます。標準設定の区切り文字は「#」です。区切り文字を変更するには，［記録オプション］の［**シリアル化オプション**］パネルで行います。区切り文字を使用するのは，再生時の文字列の解析処理を高速化するためです。

文字列を変更するときには，次のルールに従わなければなりません。

- 行の順序は変更できません。パーサは，メンバ名ではなく，値を 1 つずつ読み取ります。
- 2 つの区切り文字の間にある値だけを変更できます。
- オブジェクト参照は変更できません。オブジェクト参照は，内部の一貫性を保つためだけに示されています。
- 「_NULL_」が値（Java の null 定数）として示されていることがあります。これは文字列型の値にのみ置換できます。
- オブジェクトはスクリプト内の任意の場所でシリアル化解除できます。たとえば，**init** メソッド内のすべてのオブジェクトをシリアル化解除し，**Action** メソッドの値を使用することができます。
- オブジェクトの内部的な一貫性を保ちます。たとえば，ベクトル・オブジェクトに要素数を示すメンバ（**element count**）があり，要素を追加した場合は，エレメント数を変更する必要があります。

次のコードでは，ベクトルに，2 つの要素が含まれています。

```
public class Actions {

 // Public 関数：init
    public int init() throws Throwable {
       _string = "java.util.Vector CURRENTOBJECT = {" +
                 "int capacityIncrement = "#0#" +
                 "int elementCount = #2#" +
                 "java/lang/Object elementData[] = {" +
                       "elementData[0] = #First Element#" +
                       "elementData[1] = #Second Element#" +
                       "elementData[2] = _NULL_" +
                        ....
                       "elementData[9] = _NULL_" +

                 "}" +
       "}";
       _vector = (java.util.Vector)lr.deserialize(_string,0);
       map.set_vector(_vector);
    }
:
}
```

次の例では，ベクトルの要素の 1 つを「_NULL_」から「Third element」に変更しています。新しい要素の追加に伴って，「elementCount」メンバを「3」に変更しています。

```
public class Actions {

 // Public 関数：init
    public int init() throws Throwable {
        _string = "java.util.Vector CURRENTOBJECT = {" +
                "int capacityIncrement = "#0#" +
                "int elementCount = #3#" +
                "java/lang/Object elementData[] = {" +
                    "elementData[0] = #First Element#" +
                    "elementData[1] = #Second Element#" +
                    "elementData[2] = #Third Element#" +
                     ....
                    "elementData[9] = _NULL_" +
                "}" +
        "}";
        _vector = (java.util.Vector)lr.deserialize(_string,0);
        map.set_vector(_vector);
    }
:
}
```

オブジェクトを ASCII 表現に展開するシリアル化メカニズムは複雑なため，記録中に大きなオブジェクトを開くと，スクリプトの生成に時間がかかることがあります。この時間を短縮するために，シリアル化メカニズムのパフォーマンスを向上させるフラグを指定できます。

スクリプトに **lr.deserialize** を追加するときは，**action** メソッドではなく，**init** メソッドに追加することをお勧めします。VuGen によって文字列が 1 度だけシリアル化解除されればよいので，パフォーマンスが向上します。lr.deserialize が **action** メソッドにあると，VuGen では反復処理が行われるたびに文字列のシリアル化解除が行われることになります。

［記録オプション］の［**シリアル化オプション**］パネルでは，次のオプションを設定できます。

- ［シリアル化区切り文字］
- ［未展開のシリアル化オブジェクト］
- ［配列を展開する］
- ［配列エントリの制限］
- ［シリアル化オブジェクトを無視する］

記録オプションの詳細については，第 14 章「Java 記録オプションの設定」を参照してください。

第 16 章

Java 実行環境の設定

Java 仮想ユーザ・スクリプトを記録した後で，Java 仮想マシンの実行環境を設定できます。

本章では，次の項目について説明します。

- JVM 実行環境の設定
- 実行環境の設定のクラスパス・オプションの設定

以降の情報は，Java，EJB テスト，CORBA-Java および RMI-Java タイプの仮想ユーザを対象とします。

Java 実行環境の設定について

Java 仮想ユーザ・スクリプトを作成した後，Java VM（仮想マシン）の実行環境の設定を行います。これらの設定で追加のパスおよびパラメータを指定し，実行モードを決定できます。

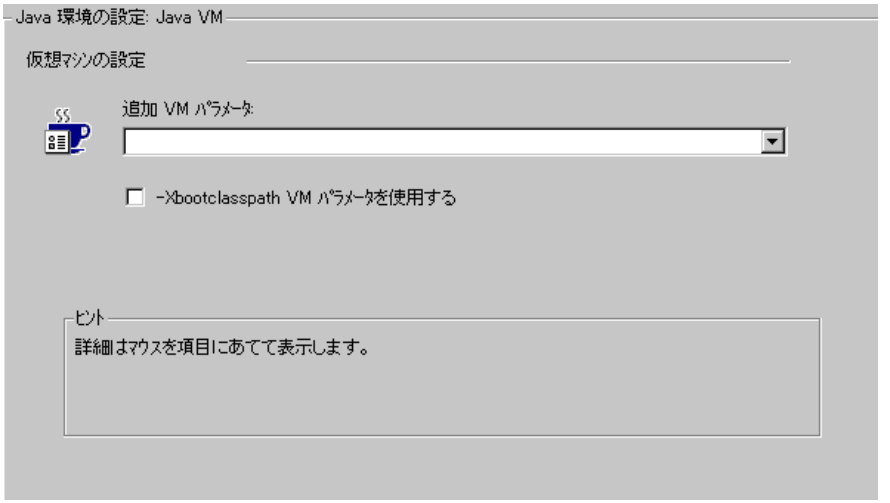
Java 関連の実行環境の設定は，［実行環境の設定］ダイアログ・ボックスの［**Java VM**］オプションで行います。

［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［**実行環境の設定を編集**］ボタンをクリックします。LoadRunner コントローラからでも実行環境の設定を変更できます。それには，［コントローラ］ウィンドウで，設定を変更する必要があるスクリプトを選択し，［**実行環境設定**］ボタンをクリックします。

本章では，Java タイプの仮想ユーザ（Java，EJB テスト，CORBA-Java，RMI-Java）の実行環境の設定についてのみ説明します。すべての仮想ユーザに適用される実行環境の設定の詳細については，第 9 章「実行環境の設定」を参照してください。

JVM 実行環境の設定

［Java VM］セクションで，Java 仮想マシンの設定情報を入力します。以下の項目の設定が可能です。



追加 VM パラメータ：仮想マシンで使用する任意のパラメータを入力します。

［**-Xbootclasspath VM パラメータを使用する**］：仮想ユーザを実行する際，自動的に **Xbootclasspath** 変数が設定されます。このダイアログ・ボックスを使って，**Xbootclasspath** で定義されているもの以外のパラメータを指定します。記録用に追加で VM パラメータを指定した場合，パラメータが保存され，再生時に使用されるように設定します。

Java VM の実行環境を設定するには，次の手順で行います。

1 ［**仮想ユーザ**］＞［**実行環境の設定**］を選択し，［実行環境の設定］ツリーの［**Java 環境の設定：Java VM**］ノードを選択します。

2 ［**追加 VM パラメータ**］ボックスに，ロード・ジェネレータ・マシンで使用する任意のパラメータを入力します。

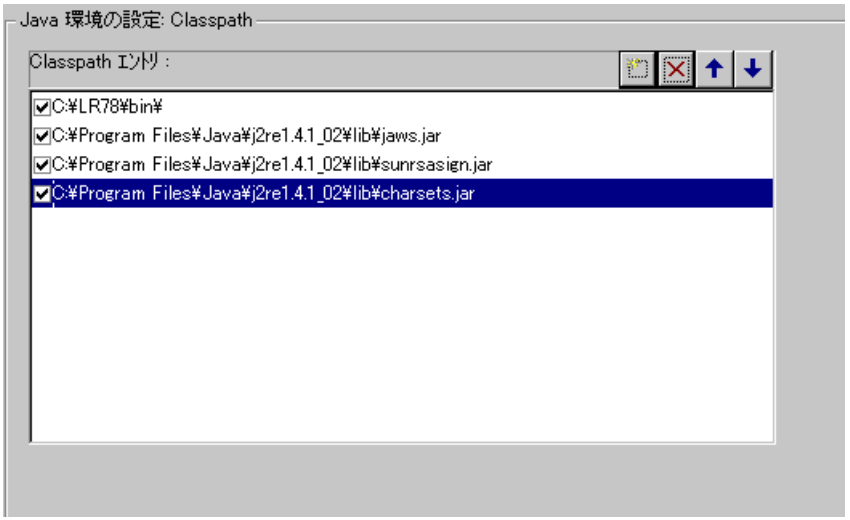
3 **-Xbootclasspath/p** オプションを使って再生するには，［**-Xbootclasspath VM パラメータを使用する**］オプションを選択します。

4 ［**OK**］をクリックします。

実行環境の設定のクラスパス・オプションの設定

［**Classpath**］セクションでは，システムのクラスパス環境変数に含まれていない追加のクラスの場所を指定できます。これらのクラスは，Java アプリケーションの実行や，正しい再生を保証するのに必要な場合があります。

コンピュータまたはネットワークから必要なクラスを検索し，特定のテストの場合にそのクラスを無効にすることができます。また，クラスの順序を変更して，クラスパスのエントリを操作することもできます。



クラスパスの実行環境を設定するには，次の手順で行います。

1 ［実行環境の設定］（F4）を開きます。［実行環境の設定］ツリーの［**Java 環境の設定：Classpath**］ノードを選択します。

2 リストにクラスパスを追加するには，次の手順で行います。

［クラスパス追加］ボタンをクリックします。クラスパスのリストに新しい行が追加されます。

新しいクラスに対して，**jar**，**zip** または他のアーカイブ・ファイルのパスと名前を入力します。または，フィールドの右にある［**参照**］ボタンをクリックして，指定したいファイルを選択します。クラスパスのリストに，ステータスが有効な状態で，新しい場所が追加されます。

3 クラスパスのエントリを完全に削除するには，対象のエントリを選択して［削除］ボタンをクリックします。

4 特定のテストの場合にだけクラスパスのエントリを無効にするには，エントリの左にあるチェック・ボックスをクリアします。

5 クラスパスをリストの下方へ移動するには，対象のエントリを選択して下向き矢印をクリックします。

6 クラスパスをリストの上方へ移動するには，対象のエントリを選択して上向きの矢印をクリックします。

7［**OK**］をクリックして，ダイアログ・ボックスを閉じます。

第 4 部

アプリケーション配備ソリューション・プロトコル

第 17 章

Citrix 仮想ユーザ・スクリプトの開発

VuGen では Citrix ICA プロトコルを使ってサーバとやり取りを行う Citrix クライアントのアクションを記録できます。記録の結果として作成されるスクリプトを，「Citrix 仮想ユーザ・スクリプト」と呼びます。

本章では，次の項目について説明します。

- Citrix 仮想ユーザ・スクリプト開発の開始
- Citrix 記録オプションの設定
- 再生の同期化
- Citrix 関数の使用
- Citrix 仮想ユーザ・スクリプトの表示
- Citrix 表示の設定
- Citrix 実行環境の設定
- ICA ファイルについて
- Citrix サーバからの切断
- Citrix 仮想ユーザ・スクリプトで作業する際のヒント

以降の情報は，Citrix ICA プロトコルを対象とします。

Citrix 仮想ユーザ・スクリプトの記録について

Citrix 仮想ユーザ・スクリプトは，Citrix クライアントとサーバ間の Citrix ICA プロトコルの通信をエミュレートします。VuGen では，通信中のすべての活動状況を記録し，仮想ユーザ・スクリプトを作成します。

リモート・サーバに対してアクションを実行すると，VuGen によってこれらのアクションを表す関数が生成されます。各関数には，**ctrx** という接頭辞が付きます。これらの関数は，マウスやキーボードのアナログ動作をエミュレートします。また，**ctrx** 関数では，特定のウィンドウが開くまで待機することで，アクションの再生を同期させることもできます。

VuGen を使って Citrix NFUSE セッションを記録できます。NFUSE セッションはクライアントではなくブラウザを対象に動作します。NFUSE セッションを記録するには，Citrix と Web の仮想ユーザ用のマルチ・プロトコル・スクリプトの記録を実行する必要があります（第 3 章「VuGen を使った記録」を参照）。マルチ・プロトコル・モードでは，VuGen によって記録中に Citrix プロトコルと Web プロトコルの両方に対応する関数が生成されます。

次に示す例では，**ctrx_mouse_click** によってマウスの左クリックをシミュレートしています。

```
ctrx_mouse_click(44, 318, LEFT_BUTTON, 0);
```

構文およびパラメータの詳細については，「**オンライン関数リファレンス**」（[**ヘルプ**] ＞ [**関数リファレンス**]）を参照してください。

記録されたスクリプトは，VuGen のメイン・ウィンドウで表示および編集ができます。このウィンドウには，セッション中に記録された API 呼び出しが表示されるので，アクションを追うことができます。

Citrix 仮想ユーザ・スクリプト開発の開始

本項では，VuGen を使って Citrix ICA 仮想ユーザ・スクリプトを開発する工程の概略を説明します。250 ページ「Citrix 仮想ユーザ・スクリプトで作業する際のヒント」も参照してください。

Citrix ICA スクリプトの開発は，次の手順で行います。

1 **VuGen を使ってアクションを記録します。**

VuGen を起動して，新しい仮想ユーザ・スクリプトを作成します。Citrix ICA クライアントを記録するときは，Citrix ICA を対象とするシングル・プロトコル仮想ユーザ・スクリプトを使用できます。NFUSE セッションの記録時には，Citrix ICA と HTTP を対象とするマルチ・プロトコル仮想ユーザ・スクリプを作成する必要があります。この仮想ユーザ・スクリプトでは両方のプロトコルを記録できます。

Citrix ICA クライアント・セッションを記録するときは，［**記録するアプリケーション**］ボックスに，記録対象アプリケーションとしてマーキュリー・インタラクティブが提供するクライアントを指定する必要があります。このクライアントは，**runDlg.exe** という名前のファイルで，LoadRunner のインストール先ディレクトリの下の **bin** ディレクトリにあります。

記録方法の詳細については，第 3 章「VuGen を使った記録」を参照してください。

2 **仮想ユーザ・スクリプトを拡張します。**

仮想ユーザ・スクリプトにトランザクション，ランデブー・ポイント，および制御フロー構造を挿入して，スクリプトを拡張します。

詳細については，第 6 章「仮想ユーザ・スクリプトの拡張」を参照してください。

3 **パラメータを定義します（任意）。**

仮想ユーザ・スクリプトに記録された固定値に対してパラメータを定義します。固定値をパラメータに置き換えることによって，異なる値を使って同じビジネス・プロセスを反復実行することができます。

詳細については，第 7 章「パラメータの定義」を参照してください。

4 **Citrix の表示オプションを設定します。**

Citrix 仮想ユーザを再生する際の表示オプションを設定します。これらのオプションを設定すると，再生中に Citrix クライアントを表示したり，エラー発生時のスナップショットを開いたりできます。詳細については，243 ページ「Citrix 表示の設定」を参照してください。

5 **実行環境を設定します。**

実行環境の設定では，スクリプト実行時の仮想ユーザの振る舞いを制御します。この設定には，ペースの設定，ログ，思考遅延時間，接続情報が含まれます。

一般的な実行環境の設定の詳細については，第 9 章「実行環境の設定」を参照してください。Citrix 固有の実行環境の設定方法の詳細については，244 ページ「Citrix 実行環境の設定」を参照してください。

6 **VuGen で仮想ユーザ・スクリプトを保存して実行します。**

VuGen で生成した仮想ユーザ・スクリプトを保存して実行し，正しく動作することを確認します。記録中，VuGen によって一連の設定ファイル，データ・ファイル，ソース・コード・ファイルが生成されます。これらのファイルには，仮想ユーザの実行環境情報およびセットアップ情報が含まれます。VuGen ではこれらのファイルがスクリプトと一緒に保存されます。

仮想ユーザ・スクリプトをスタンドアロン・テストとして実行する方法の詳細については，250 ページ「Citrix 仮想ユーザ・スクリプトで作業する際のヒント」と第 11 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。

Citrix 仮想ユーザ・スクリプトを作成し終わったら，シナリオに組み込みます。詳細については，『**LoadRunner コントローラ・ユーザーズ・ガイド**』を参照してください。

Citrix 記録オプションについて

次の項目について Citrix 記録オプションの設定が可能です。

- ログイン
- 設定
- レコーダ

ログイン

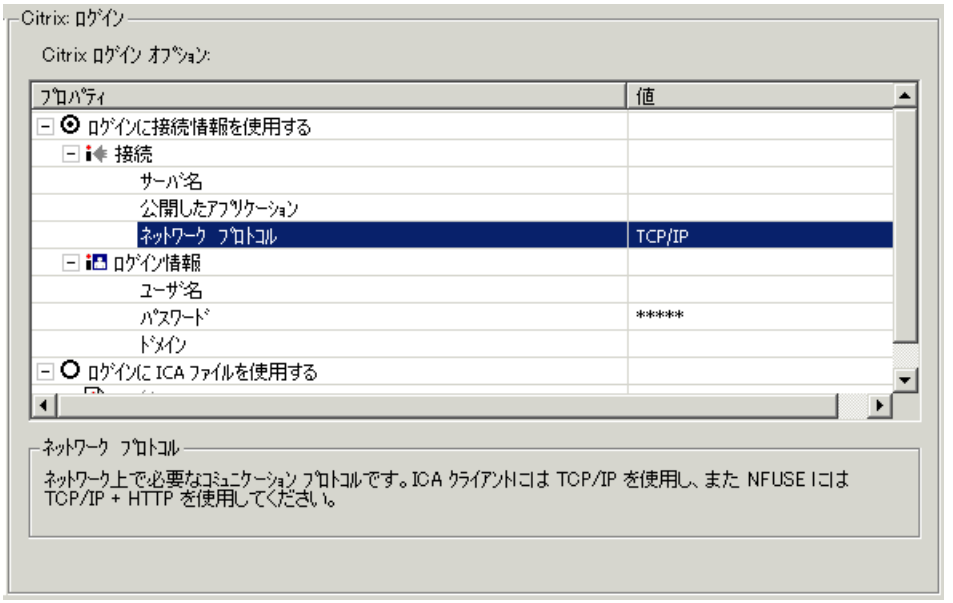
［**Citrix：ログイン**］記録オプションで，記録セッションに対する接続情報とログイン情報を設定します。サーバ名は必ず指定してください。省略すると，接続に失敗します。ログイン情報を省略すると，指定したサーバがクライアントによって検出されたときに，情報を入力するよう求められます。

ログイン情報を直接設定することも，VuGen に **ica** ファイルに格納されている既存の設定情報を使用させることもできます。

サーバ名は必ず指定してください。省略すると，接続に失敗します。ログイン情報を指定すると，この情報は **ctrx_connect_server** 関数として記録されます。

```
ctrx_connect_server(“steel“, “test“;, “test“, “testlab“;);
```

ログイン情報を省略すると，指定したサーバがクライアントによって検出されたときに，情報を入力するよう求められます。



ログインのための接続情報の使用

［**接続**］セクションに，以下の接続情報を入力します。

- Citrix サーバの名前。
- Citrix サーバで認識される，公開アプリケーションの名前。公開アプリケーションを指定していない場合，VuGen ではサーバのデスクトップが使用されます。
- 優先する［**ネットワーク プロトコル**］。TCP/IP か TCP/IP+HTTP のどちらかを選択します。

［**ログイン情報**］セクションに，以下のログイン情報を入力します。

- Citrix サーバへのログイン名。
- Citrix サーバへログインするためのパスワード。
- Citrix サーバのドメイン。

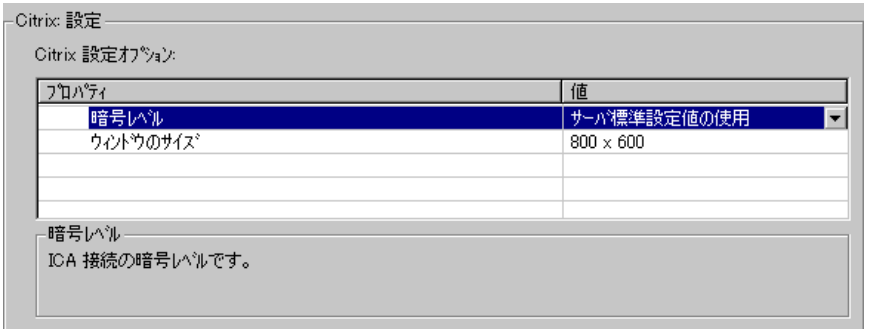
ログインのための ICA ファイルの使用

関連するすべての構成情報が含まれた既存の **.ica** ファイルがある場合は，［**ログインに ICA ファイルを使用する**］を選択します。次の行で，**.ica** ファイルのフル・パスを指定します。

ICA ファイルの形式については，247 ページ「ICA ファイルについて」を参照してください。手順 6 に進んでください。

設定

［**Citrix：設定**］記録オプションでは，記録セッション中の Citrix クライアントに対するウィンドウ・プロパティと暗号化設定を設定します。



- ［**暗号レベル**］: ICA 接続の暗号化レベル。［**基礎値**］，［**128 ビット（ログインのみ）**］，［**40 ビット**］，［**56 ビット**］，［**128 ビット**］，または［**サーバ標準設定値の使用**］から選択します。
- ［**ウィンドウのサイズ**］: クライアント・ウィンドウのサイズ。「**640 x 480**」，「**800 x 600**」（標準設定），「**1024 x 768**」から選択できます。

レコーダ

［**Citrix レコーダ オプション**］で，タイトルの変更されたウィンドウ名を生成する方法を指定できます。

次のオプションが使用できます。

- ［**新規のウィンドウ名をそのまま使用する**］は，ウィンドウのタイトルをそのままウィンドウ名として使用する場合に選択します。
- ［**新規名に共通のプレフィックスを使用する**］は，ウィンドウ・タイトルの先頭の共通文字列をウィンドウ名として使用する場合に選択します。
- ［**新規名に共通のサフィックスを使用する**］は，ウィンドウ・タイトルの末尾の共通文字列をウィンドウ名として使用する場合に選択します。

Citrix 記録オプションの設定

記録を行う前に必要な記録オプションを設定します。

注：コントローラ，ロード・ジェネレータ・マシン，および画面の各設定が同じであることを確認します。設定が同じでないと，VuGen が再生中に正しいビットマップを見つけることができない場合があります。

Citrix 記録オプションの設定は，次の手順で行います。

1 ［記録オプション］ダイアログ・ボックスを開きます。［**ツール**］＞［**記録オプション**］を選択するか，［記録開始］ダイアログ・ボックスで［**オプション**］ボタンをクリックします。キーボード・ショートカット・キーは，Ctrl キーを押しながら F7 キーを押します。

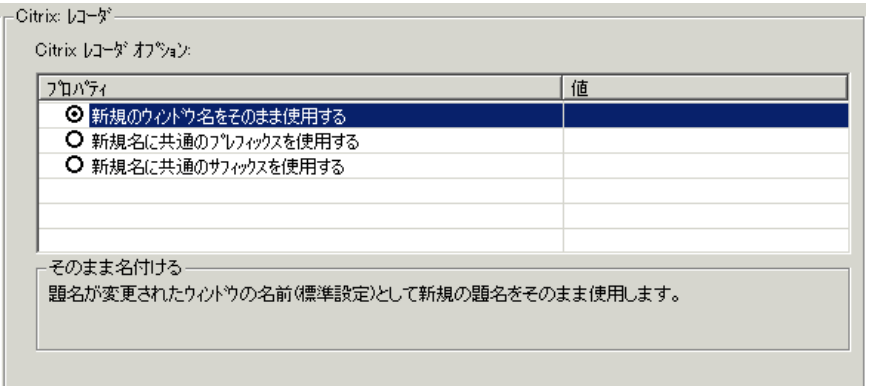
2 ［Citrix］の［**ログイン**］ノードを選択します。

- 関連するすべての構成情報が含まれた既存の **.ica** ファイルがある場合は，［**ログインに ICA ファイルを使用する**］を選択します。続く行では，ICA ファイルのフル・パスを指定するか，［参照］ボタンをクリックして，ローカル・ディスクまたはネットワーク上のファイルを探します。

➤ **ICA** ファイルがない場合は，［**ログインに接続情報を使用する**］を選択します。これは，標準設定です。［**接続**］セクションに，以下の接続情報を入力します。［**ログイン情報**］セクションに，以下のログイン情報を入力します。

3 ［Citrix］の［**設定**］ノードを選択します。暗号化レベルとウィンドウ・サイズを選択します。

4 ［Citrix］の［**レコーダ**］ノードを選択します。記録セッション中にタイトルが変わるウィンドウのウィンドウ名を生成する方法を指定します。



5 NFUSE セッションの記録時には，Web の記録モードを「URL ベース」に設定します。［**インターネット プロトコル：記録**］記録オプションを選択し，［**URL ベースのスクリプト**］ノードを選択します。

6 ［**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。

再生の同期化

VuGen では再生中のアクションの同期化を行う関数が自動的に生成されます。同期化関数は手作業で追加することもできます。

自動同期化

記録中，仮想ユーザ・スクリプトの再生の同期化を支援する関数，**ctrx_sync_on_window** が VuGen によって自動的に生成されます。

ctrx_sync_on_window は，指定されたイベントが発生するまで，仮想ユーザに再生の再開を待機するよう指示する関数です。指定できるイベントは **Create** または **Active** です。Create イベントを指定した場合は，ウィンドウが作成されるまで待機します。Active イベントを指定した場合は，ウィンドウが作成されてアクティブになるまで（フォーカスを得るまで）待機します。メニューなどのウィンドウ以外のオブジェクトは，完全にアクティブになることはないため，VuGen では通常 **Create** イベントを待機する関数が生成されます。標準のウィンドウの場合，VuGen では **Active** イベントを待機する関数が生成されます。

VuGen で，**ctrx_sync_on_window** 関数と併せて記録されたすべてのウィンドウについて，画面ショットが取得され，それがスクリプトの data¥bitmap ディレクトリに格納されます。この関数の 7 番目の引数がビットマップ・ファイルの名前です。

```
ctrx_sync_on_window("ICA Administrator Toolbar", ACTIVATE, 768, 0, 33,
                    573, "129c54d64c9679ea1a0cb8bb02c2a981", CTRX_LAST);
```

手作業による同期化

VuGen のインタフェースを使うか，ユーザ定義の同期化関数を挿入することで，手作業で同期を追加することもできます。

手作業で同期化関数を追加する方法の 1 つは，VuGen のユーザ・インタフェースを使用する方法です。再生中，VuGen の記録フローティング・ツールバーに，マーカ・ツールが表示されます。このツールを使って，再生を再開する前にフォーカスを得る必要のあるクライアント・ウィンドウのビットマップ領域をマークします。この方法は一般に，ブラウザ・ウィンドウの特定の画像に対して使用します。記録セッション中に，VuGen のフローティング・ツールバー

のマーカ・ツールを使って，領域をビットマップとしてマークします。VuGen によって **ctrx_sync_on_bitmap** 関数が生成されます。

```
ctrx_sync_on_bitmap(93, 227, 78, 52,
                                    ”66de3122a58baade89e63698d1c0d5dfa”);
```

再生時には，VuGen によって指定の領域のビットマップが調べられます。同期化する部分をマークするには，マーカ・ボタンをクリックします。ウィンドウ内の待機する対象となる領域の左上角から右下角までマウスをドラグしてマークします。選択した座標を引数として **ctrx_sync_on_bitmap** 関数が生成されます。

別の種類の同期化関数を追加するには，次のいずれかの関数をスクリプトに手作業で入力します。

- **ctrx_sync_on_bitmap_change**：変更があるまで待機する対象となる領域（位置とサイズ）を指定します。これは，変更があることはわかっていても，どのようなデータあるいは画像が表示されるかわからない領域を指定する場合に便利です。ビットマップ領域の正確な座標を取得するには，まず先に説明したようにマーカ・ツールを使って **ctrx_sync_on_bitmap** 関数を記録し，関数名を手作業で変更して，5 番目の引数を削除すると簡単です。

  この関数の構文は次のとおりです。

  ctrx_sync_on_bitmap (x 座標 , y 座標 , 幅 , 高さ , ハッシュ値);
  ctrx_sync_on_bitmap_change (x 座標 , y 座標 , 幅 , 高さ ,
  [初期待機時間 ,] [タイムアウト ,] CTRX_LAST);

  ctrx_sync_on_bitmap_change に，初期待機時間の値（変更の有無の確認を開始する時間）とタイムアウト（変更を待機する最大の秒数。それを超えると失敗となる）の引数を任意で追加できます。

```
/* 記録された関数 */
ctrx_sync_on_bitmap(93, 227, 78, 52,
                                    ”66de3122a58baade89e63698d1c0d5dfa”);
```

```
/* 関数に変更を加えて初期待機時間を 300，タイムアウトを 400 に設定 */
ctrx_sync_on_bitmap_change(93, 227, 78, 52, 300, 400, CTRX_LAST);
```

記録中，**ctrx_sync_on_bitmap** 関数のために生成されたビットマップは，スクリプトの **data¥bitmap** ディレクトリに保存されます。ビットマップ名は＜**ハッシュ値**＞ .bmp という形式です。再生中に同期化に失敗した場合には，生成されたビットマップはスクリプトの出力ディレクトリに書き込まれます。あるいは，シナリオで実行している場合には，出力ファイルが書き込まれる場所に書き込まれます。新しいビットマップを確認して，同期化が失敗した理由を調べることができます。

- **ctrx_set_waiting_time :** 他の Citrix 同期化関数の待機時間を設定します。この設定は，同一スクリプト内でこの関数以降のすべての関数に適用されます。たとえば，**ctrx_set_window** 関数がタイムアウトする場合，標準設定のタイムアウトである 60 秒を 180 秒に延ばすことができます。

```
ctrx_set_waiting_time(180);
```

- **ctrx_win_exist :** Citrix クライアントでウィンドウが表示されているかどうかを調べます。たとえば，この関数を使って，ブラウザが起動されたかどうかを確認できます。2 番目の引数は，ウィンドウが表示されるまで関数が待機する時間を示します。ブラウザが起動されなかった場合は，ブラウザ・アイコンをダブルクリックします。

```
if (!ctrx_win_exist(“Welcome to MSN.com- Microsoft Internet Explorer”,6))
  ctrx_mouse_double_click(34, 325, LEFT_BUTTON, 0)
```

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

Citrix 関数の使用

Citrix 記録セッション中，VuGen によってクライアントとリモート・サーバ間のやり取りをエミュレートする関数が生成されます。生成された関数には，**ctrx** という接頭辞が付きます。どの関数も記録セッションの後，仮想ユーザ・スクリプトを手作業で編集して追加できます。**ctrx** 関数の詳細については，「**オンライン関数リファレンス**」（[**ヘルプ**] > [**関数リファレンス**]）を参照してください。

ウィンドウ名を指定する関数では，ワイルドカード記号であるアスタリスク (*) を使用できます。ワイルドカードは，文字列の先頭，中，末尾のどこにでも置くことができますが，1 つのウィンドウ名に使用できるワイルドカードは 1 つだけです。

Citrix 関数

ctrx_connect_server	Citrix クライアントを使ってリモート・サーバに接続します。
ctrx_disconnect_server	サーバへの接続を閉じます。
ctrx_get_bitmap_value	指定したビットマップのハッシュ値を取得します。
ctrx_get_window_name	フォーカスを得ているウィンドウの名前を取得します。
ctrx_get_window_position	指定したウィンドウまたはフォーカスを得ているウィンドウの位置を取得します。
ctrx_key	英数字以外のキーの押下をエミュレートします。
ctrx_key_down	キーボードのキーの押下をエミュレートします。
ctrx_key_up	キーボードのキーの解放をエミュレートします。
ctrx_mouse_click	マウスのクリックをエミュレートします。
ctrx_mouse_double_click	マウスのダブルクリックをエミュレートします。
ctrx_mouse_down	マウス・ボタンの押下をエミュレートします。

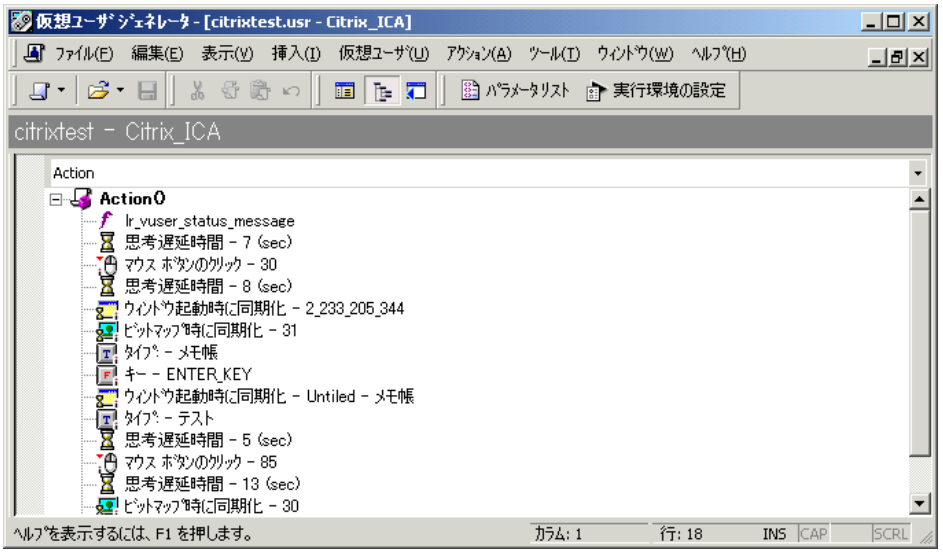
ctrx_mouse_move	マウスの動きをエミュレートします。
ctrx_mouse_up	マウス・ボタンの解放をエミュレートします。
ctrx_nfuse_connect	Citrix サーバに NFUSE ポータル経由で接続します。
ctrx_set_connect_opt	接続オプションを設定します。
ctrx_set_exception	例外処理を指定します。
ctrx_set_waiting_time	以降のすべての計時関数に待機時間を設定します。
ctrx_set_window	指定したウィンドウが表示されるのを待機します。
ctrx_set_window_ex	指定したウィンドウが表示されるのを,指定秒数の間待機します。
ctrx_sync_on_bitmap	座標で指定したビットマップが表示されるまで待機します。
ctrx_sync_on_bitmap_change	座標で指定した領域に変化があるまで待機します。
ctrx_sync_on_window	ウィンドウが作成されるかアクティブになるのを待機します。
ctrx_type	英数字キーの入力をエミュレートします。
ctrx_unset_window	指定したウィンドウが閉じるのを待機します。
ctrx_wait_for_event	指定したイベントが発生するのを待機します。
ctrx_win_exist	指定したウィンドウが存在するかどうか確認します。

Citrix 仮想ユーザ・スクリプトの表示

vuser_init，**Actions**，**vuser_end** の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。その内容が右側の表示枠に表示されます。VuGen で Citrix 仮想ユーザ・スクリプトを表示または編集するときには，スクリプトをアイコン形式のツリー・ビューで表示するか，テキスト形式のスクリプト・ビューで表示するかを選択します。

ツリー・ビューを表示するには，次の手順で行います。

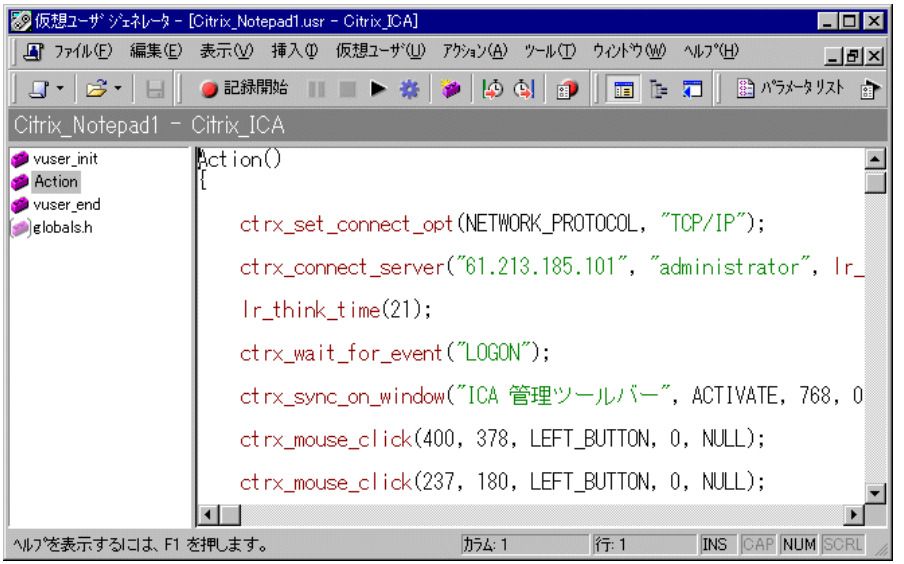
VuGen のメイン・メニューから，［**表示**］＞［**ツリー ビュー**］を選択するか，［**ツリーを表示**］アイコンをクリックします。仮想ユーザ・スクリプトの Actions セクションがアイコン形式のツリー・ビューで表示されます。すでにツリー・ビューが表示されている場合，このメニュー項目は無効になります。



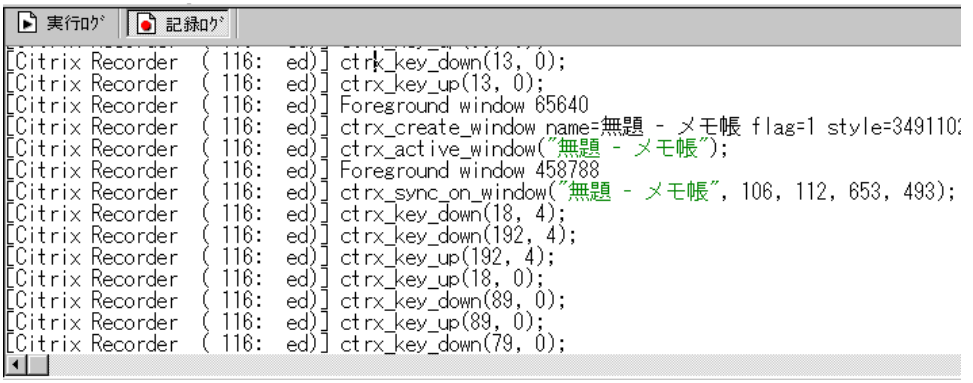
スクリプト・ビューを表示するには，次の手順で行います。

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。仮想ユーザ・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，このメニュー項目は無効になっています。



記録セッションの詳細については，記録中に生成されるアクションのログを参照してください。［**表示**］＞［**出力ウィンドウ**］を選択し，［**記録ログ**］タブを選択します。VuGen で実行されたすべてのアクションの詳細なログが表示されます。



Citrix 表示の設定

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

［一般オプション］ダイアログ・ボックスで，Citrix の表示を設定します。

Citrix 表示オプションを設定するには，次の手順で行います。

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

2 ［**Citrix の表示**］タブを選択します。

3 仮想ユーザ・スクリプトの再生中に Citrix クライアントを表示するには，［**再生中にクライアントを表示する**］を選択します。

4 エラー発生時に画面のスナップショットを表示するには，［**エラー時に画面のスナップショットを表示する**］を選択します。

5 ［**OK**］をクリックします。

Citrix 実行環境の設定

Citrix 仮想ユーザ・スクリプトを作成したら，実行環境を設定します。これらの設定によって，スクリプト実行時の仮想ユーザの振る舞いを制御できます。［**設定**］ノードでの Citrix 実行環境の設定は，Citrix クライアントのプロパティと一致していなくてはなりません。これらの設定は，サーバにかかる負荷に影響します。接続のプロパティを表示するには，［Citrix Program Neighborhood］で ICA 接続を表すアイコンを選択し，右クリックして表示されるメニューから［**Properties**］を選択します。［**Default Options**］タブを選択します。

注：Citrix 仮想ユーザではモデム速度のエミュレーションと IP スプーフィングはサポートされていません。

［実行環境設定］ダイアログ・ボックスを使って，以下の領域についての Citrix の実行環境を設定します。

- Citrix 設定実行環境の設定
- Citrix タイムアウト実行環境の設定

Citrix 設定実行環境の設定

画面の遅延，データ圧縮，ディスク・キャッシュ，マウスの動きのキューイングに関する設定です。

実行環境を設定するには，次の手順で行います。

1 ［実行環境設定］ダイアログ・ボックスを開きます。VuGen ツールバーにある［**実行環境の設定を編集**］ボタンをクリックするか，［**仮想ユーザ**］＞［**実行環境の設定**］を選択します。

2 [Citrix] の [**設定**] ノードを選択します。[**一般**] プロパティを指定します。

- [**SpeedScreen 待ち時間の減少**]：ネットワークの速度が遅いときにユーザとのやり取りを向上させるために使用する機能。ネットワークの速度に応じてこの機能を「**オン**」または「**オフ**」にします。「**自動**」オプションを選択すると，現在のネットワーク速度に基づいてオプションのオン / オフが切り替わります。ネットワーク速度がわからない場合は，このオプションを [**サーバ標準設定値の使用**] に設定し，マシンの標準設定を使用するようにします。

3 [**データ圧縮を使用する**] オプションを設定します。このオプションは仮想ユーザに，転送するデータを圧縮するよう指示します。このオプションを有効にするには，このオプションの左側にあるチェック・ボックスを選択します。無効にするには，チェック・ボックスをクリアします。帯域幅が限られている場合は，データ圧縮を有効にします（標準設定では有効）。

4 [**ビットマップにディスク キャッシュを使用する**] オプションを設定します。このオプションは仮想ユーザに，ビットマップや，よく使用するグラフィカル・オブジェクトをローカルのキャッシュに格納するよう指示します。このオプションを有効にするには，このオプションの左側にあるチェック・ボックスを選択します。無効にするには，チェック・ボックスをクリアします。帯域幅が限られている場合は，このオプションを有効にします（標準設定では無効）。

5 [**マウスの移動とキーストロークをキューに挿入する**] オプションを設定します。このオプションを有効にすると，マウスの動きとキーストロークのキューが作成され，これらがパケットとして低い頻度でサーバに送られます。このオプションは，低速接続の場合のネットワーク・トラフィックを減らすためのものです。このオプションを有効にすると，セッションにおけるキーボードとマウスの動きに対する応答が鈍くなります。このオプションを有効にするには，このオプションの左側にあるチェック・ボックスを選択します。無効にするには，チェック・ボックスをクリアします（標準設定では無効）。

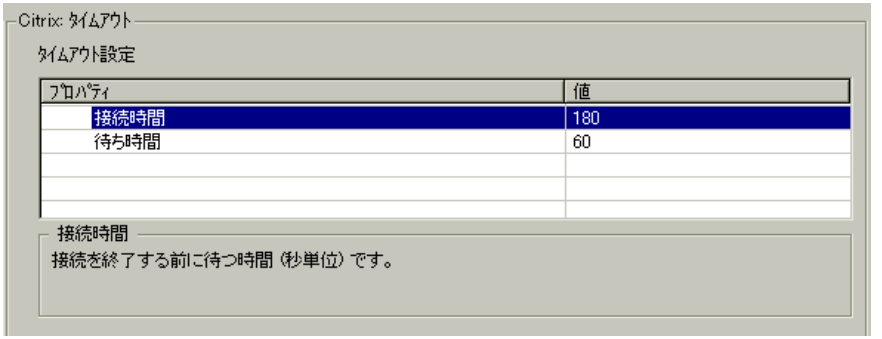
Citrix タイムアウト実行環境の設定

タイムアウト設定は接続時間と待ち時間に関係します。

タイムアウト実行環境を設定するには，次の手順で行います。

1 ［実行環境設定］ダイアログ・ボックスを開きます。VuGen ツールバーにある［**実行環境の設定を編集**］ボタンをクリックするか，［**仮想ユーザ**］＞［**実行環境の設定**］を選択します。

2 ［Citrix］の［**タイムアウト**］ノードを選択します。



➤ ［**接続時間**］：確立されている接続を終了する前に，アイドル状態で待機する秒数です。標準では 180 秒に設定されています。

➤ ［**待ち時間**］：接続を終了する前に，同期ポイントでアイドル状態のまま待機する秒数です。標準では 60 秒に設定されています。

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。

ICA ファイルについて

Citrix ICA クライアント・ファイルは，Citrix クライアントを通じてアクセスされるアプリケーションの設定情報が含まれているテキスト・ファイルです。これらのファイルの拡張子は **ica** でなければなりません。また，以下の形式に適合していなければなりません。

```
[WFClient]
Version=
TcpBrowserAddress=

[ApplicationServers]
AppName1=

[AppName1]
Address=
InitialProgram=#
ClientAudio=
AudioBandwidthLimit=
Compress=
DesiredHRES=
DesiredVRES=
DesiredColor=
TransportDriver=
WinStationDriver=

Username=
Domain=
ClearPassword=
```

注：［記録オプション］を使って ICA ファイルをロードすると，VuGen によってファイルがスクリプトと一緒に保存されるので，ICA ファイルを各インジェクタ・マシンにコピーする手間が省けます。

次の例は，Citrix クライアントを通じて Microsoft Word をリモート・マシン上で使うための ICA ファイルのサンプルです。

```
[WFClient]
Version=2
TcpBrowserAddress=235.119.93.56

[ApplicationServers]
Word=

[Word]
Address=Word
InitialProgram=#Word
ClientAudio=On
AudioBandwidthLimit=2
Compress=On
DesiredHRES=800
DesiredVRES=600
DesiredColor=2
TransportDriver=TCP/IP
WinStationDriver=ICA 3.0

Username=test
Domain=user_lab
ClearPassword=test
```

Citrix サーバからの切断

Citrix クライアントが接続を終了するとき，標準設定では，次回そのクライアントが新しい接続を開いたときのためにセッションを保存するようにサーバが設定されています。したがって，同じクライアントで新たに接続すると，前回切断したときと同じ作業領域が表示されます。これは，テストを実行する場合には望ましくありません。一般に，テストを実行するときは，毎回初期状態の作業領域が必要だからです。

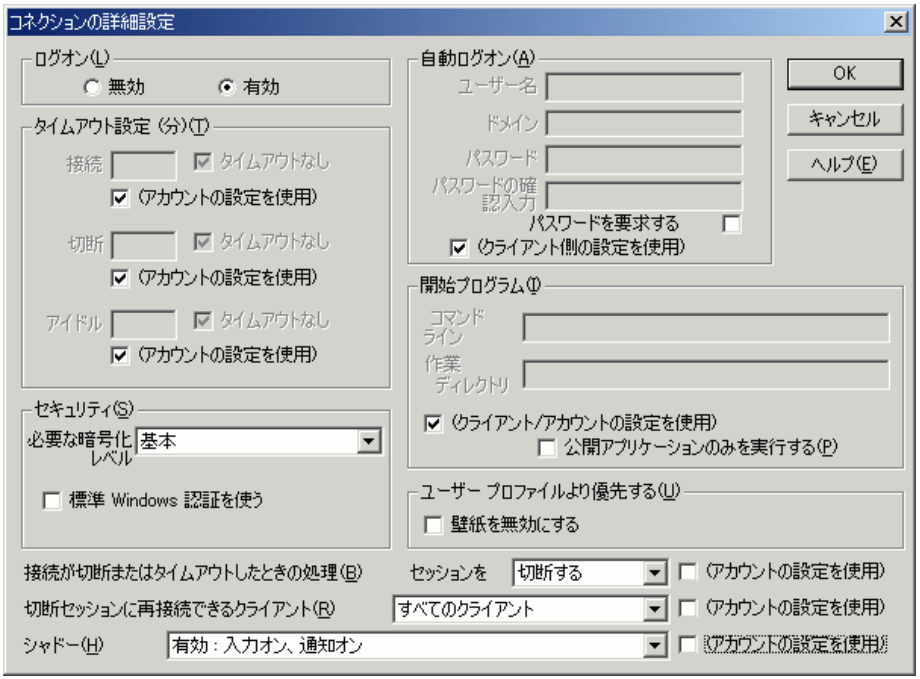
それを解決するには，クライアントによって接続が切断またはタイムアウトしたときに，Citrix サーバに前のセッションを保存させずに，クライアントから完全に切断できるよう設定します。

Citrix サーバを XP サーバから強制的に切断するには，次の手順で行います。

1 ［Citrix Connection Configuration］ダイアログ・ボックスを開きます。［**スタート**］＞［**プログラム**］＞［**Citrix**］＞［**MetaFrame XP**］＞［**Citrix Connection Configuration**］を選択します。

2 ica-tcp 接続名をダブルクリックします。［Edit Connection］ダイアログ・ボックスが表示されます。

3 ［**Advanced**］ボタンをクリックします。［コネクションの詳細設定］ダイアログ・ボックスが表示されます。



4 このダイアログの一番下のセクションにある［**接続が切断またはタイムアウトしたときの処理**］リスト・ボックス横の［**(アカウントの設定を使用)**］チェック・ボックスをクリアします。リスト・ボックスのエントリを「**reset**」に変更します。

Citrix 仮想ユーザ・スクリプトで作業する際のヒント

記録に関するヒント

- セッションを記録するときは必ず，接続操作から始まって，クリーンアップで終わる完全なビジネス・プロセスを実行するようにします。ビジネス・プロセス全体を始めから再実行できるところでセッションを終了するようにします。クライアントやアプリケーションのウィンドウを開いたままにしないようにします。
- 記録セッション中はウィンドウを動かしたり，ウィンドウの大きさを変えたりしないことをお勧めします。
- VuGen はデスクトップのカラー設定を使用します。
- ビットマップの同期化を確実に成功させるために，解像度の設定が一致していることを確認します。記録用マシンの ICA クライアントの設定，記録オプション，および実行環境の設定を確認します。インジェクタ・マシンで，ICA クライアントの設定を調べ，それらがすべてのインジェクタ・マシンおよび記録用マシンと一致することを確認します。
- サポートされている解像度（ウィンドウのサイズ）は，640 × 480，800 × 600，および 1024 × 768 です。記録用マシンの表示設定は 1024 × 768 にすることをお勧めします。標準の大きさが 800 × 600 の Citrix のウィンドウを正しく表示できるからです。
- 記録中に，イベント（たとえば HTML ページの読み込みなど）を待機する場合には，**ctrx_sync_on_bitmap** 関数を使って同期化ポイントを手作業で追加することをお勧めします。詳細については，236 ページ「再生の同期化」を参照してください。
- Citrix サーバがセッションを完全に終了するように設定します。248 ページ「Citrix サーバからの切断」を参照してください。
- 接続処理は「**vuser_init**」セクションに，終了処理は「**vuser_end**」セクションに記録します。こうすることで，接続処理を反復実行するのを避けることができます。
- Citrix セッションでまったくアクションを記録できなかった場合は，マシンにインストールされている Citrix クライアントが 1 つだけ（マーキュリー・インタラクティブが提供しているクライアント）であることを確認してください。インストールされているクライアントが 1 つだけかどうかを調べるには，コントロールパネルから［プログラムの追加と削除］ダイアログ・ボックスを開き，Citrix ICA クライアントのエントリが 1 つだけあることを確認します。

- サブメニューを開くときは，メニューの自動展開に頼らずに，各オプションを明示的にクリックします。たとえば，［**スタート**］＞［**プログラム**］＞［**Microsoft Word**］の場合は，「**プログラム**」という語をクリックします。
- Citrix クライアントの更新を有効にするかどうか尋ねられた場合には，クライアントの更新を無効にします。

再生に関するヒント

- 接続中に複数の仮想ユーザによって過負荷になるのを防ぐため，仮想ユーザの初期化クォータを（サーバの性能に応じて）4 から 10 に設定するか，スケジューラを使用して仮想ユーザのランプアップによる初期化を実施します。
- 最良の結果を得るには，実行環境の設定で思考遅延時間を無効にしないようにします。思考遅延時間は，特に安定するまでに時間を要する **ctrx_sync_on_window** 関数や **ctrx_sync_on_bitmap** 関数の前には重要です。
- 別のマシンでスクリプトを再生する場合には，記録マシンと再生マシンの間で，Citrix クライアントのウィンドウ・サイズ（解像度），ウィンドウの色設定，システム・フォント，その他の標準オプションの設定が同じであることを確認します。これらの設定はビットマップのハッシュ値に影響し，不一致があった場合は，再生が失敗する可能性があります。Citrix クライアントの設定を表示するには，Citrix プログラム・グループから項目を選択し，［**Application Set Settings**］を選択するか，ICA 接続アイコンを選択するときは右クリック・メニューから［**Custom Connection Settings**］を選択します。［Default Options］タブをクリックします。
- Citrix 仮想ユーザを実行するマシンは，使用できるグラフィック・リソースによっては，実行できる仮想ユーザの数が限られる可能性があります。各マシンの仮想ユーザ数を増やすには，そのマシンでターミナル・サーバ・セッションを開始します。このターミナル・サーバは新規インジェクタ・マシンとなります。この仮想インジェクタ・マシンをコントローラから指定するには，＜**マシン名**＞:1，＜**マシン名**＞:2 のような形式を使用します。＜マシン名＞にはマシン名か IP アドレスを使用します。ターミナル・サーバのセッションでは標準で 256 色のカラー・セットが使用されます。ターミナル・セッションを負荷テスト用に使用しようとしている場合は，必ず 256 色のカラー・セットを備えたマシンに記録してください。

デバッグに関するヒント

- 問題が生じているコードの行を知るには，VuGen でスクリプトにブレークポイントを追加します。
- 再生が失敗した場合，スクリプトに同期化関数を挿入して，対象ウィンドウがフォーカスを得るまで，待機時間を延ばす必要があるかもしれません。**lr_think_time** 関数を使って遅延時間を手作業で追加することもできますが，236 ページ「再生の同期化」で説明した同期化関数を使用することをお勧めします。
- ［拡張ログ］で他の再生情報を参照できます。拡張ログを有効にするには，実行環境の設定（F4 ショートカットキー）の［**ログ**］パネルを使います。この情報は［実行ログ］タブまたは，スクリプト・ディレクトリの「**output.txt**」ファイルで参照できます。
- エラーが発生すると，VuGen によって画面のスナップショットがスクリプトの「**output**」ディレクトリに保存されます。このビットマップを見て，エラーが起きた原因を確認できます。
- 記録中，**ctrx_sync_on_bitmap** 関数のために生成されたビットマップは，スクリプトの **data¥bitmap** ディレクトリに保存されます。ビットマップ名は＜**ハッシュ値**＞ .bmp という形式です。再生中に同期化に失敗した場合には，生成されたビットマップはスクリプトの出力ディレクトリに書き込まれます。あるいは，シナリオで実行している場合には，出力ファイルが書き込まれる場所に書き込まれます。新しいビットマップを確認して，同期化が失敗した理由を調べることができます。
- シナリオ実行時にすべての仮想ユーザを表示するには，仮想ユーザ・コマンド・ライン・ボックスに次のように入力します。-lr_citrix_vuser_view. コントローラで，［グループ情報］ダイアログ・ボックスを開き［**詳細表示**］をクリックして，ダイアログ・ボックスを拡張します。この操作は，テストのスケーラビリティに影響するので，問題のある仮想ユーザの振る舞いを調査する場合にだけ行うようにします。
- サーバのバージョンを調べるには，MetaFrame XP または MetaFrame 1.8 サーバがインストールされていることを確認します。サーバのコンソール・ツールバーにある［Citrix Connection Configuration］を選択して，［**Help**］＞［**About**］を選択します。

第 5 部

クライアント・サーバ・プロトコル

第 18 章

データベース仮想ユーザ・スクリプトの作成

VuGen を使用して，データベース・クライアント・アプリケーションとサーバの間の通信を記録することができます。その結果生成されるスクリプトをデータベース仮想ユーザ・スクリプトといいます。

本章では，次の項目について説明します。

- データベース仮想ユーザの紹介
- データベース仮想ユーザ技術について
- データベース仮想ユーザ・スクリプトの概要
- データベース記録オプションの設定
- LRD 関数の使用法
- データベース仮想ユーザ・スクリプトについて
- エラー・コードの分析
- エラー処理

以降の情報は，クライアント / サーバ型データベース（CtLib，DbLib，Informix，MS SQL Server，Oracle，ODBC，DB2-CLI）および ERP Siebel 仮想ユーザ・スクリプトを対象とします。

データベース仮想ユーザ・スクリプトの記録について

サーバと通信を行っているデータベース・アプリケーションを記録すると，VuGen によってデータベース仮想ユーザ・スクリプトが生成されます。VuGen では，以下のデータベース・タイプがサポートされています。CtLib，DbLib，Informix，Oracle，ODBC，DB2-CLI。生成されたスクリプトには，データベース操作を表す LRD 関数が含まれています。LRD 関数の名前には **lrd** という接頭辞が付いており，1 つまたは複数のデータベース機能に対応します。たとえば，**lrd_fetch** 関数はフェッチ操作を表します。

記録されたセッションを実行すると，仮想ユーザ・スクリプトとデータベース・サーバとの間で直接通信が行われ，実際のユーザと同じ操作が実行されます。仮想ユーザの動作を設定して（実行環境の設定），操作の反復回数と反復間隔を指定できます。詳細については，第 9 章「実行環境の設定」を参照してください。

VuGen を使って，記録された定数をパラメータで置き換えることによって，スクリプトをパラメータ化できます。詳細については，第 7 章「パラメータの定義」を参照してください。

さらに，スクリプトのクエリーや他のデータベース・ステートメントを相関させて，特定のクエリーの結果を別のクエリーに結び付けることができます。詳細については，第 8 章「ステートメントの相関」を参照してください。

トラブルシューティング情報とスクリプト作成のヒントについては，第 64 章「VuGen のデバッグのヒント」を参照してください。

データベース仮想ユーザの紹介

全国のカスタマー・サービス担当者がアクセスする顧客情報のデータベースがあるとします。この場合には，データベース仮想ユーザを使って，データベース・サーバが多数の情報の問い合わせに対応するという状況をエミュレートします。データベース仮想ユーザでは，以下のことが可能です。

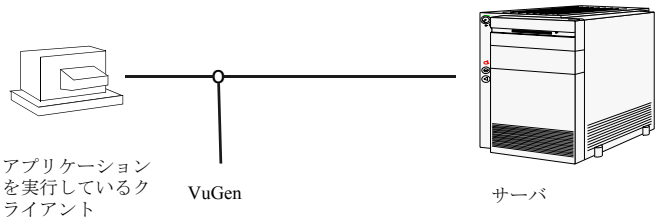
- サーバへの接続
- SQL クエリーの発行
- 情報の検索と処理
- サーバからの切断

まず，利用可能なロード・ジェネレータに数百の DB 仮想ユーザを振り分けます。各仮想ユーザはサーバ API を使ってデータベースにアクセスします。これにより，多数のユーザによる負荷をかけた状態でのサーバのパフォーマンスを測定できます。

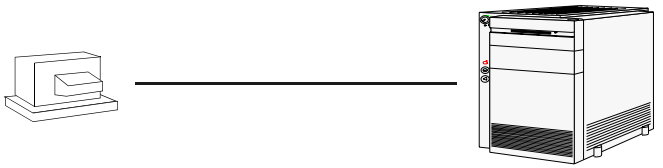
サーバ API への呼び出しが含まれるプログラムをデータベース（DB）仮想ユーザ・スクリプトといいます。データベース仮想ユーザ・スクリプトによって，クライアント・アプリケーションと，そのすべての操作がエミュレートされます。コントローラを使用して，1 つのスクリプトを複数の仮想ユーザに割り当てることができます。仮想ユーザによってスクリプトが実行されると，クライアント / サーバ・システムにユーザ負荷をかけた状態がエミュレートされます。LoadRunner によって生成されるパフォーマンス・データは，レポートやグラフを使って分析できます。

データベース仮想ユーザ技術について

VuGen では，データベース・クライアントとサーバの間のやり取りをすべて記録することによって，データベース仮想ユーザ・スクリプトを作成します。VuGen で，データベースのクライアント側を監視し，データベース・サーバとの間で送受信されるすべての要求を追跡します。



VuGen を使って作成する他のすべての仮想ユーザと同様に，データベース仮想ユーザも，サーバと通信を行うときにクライアント・ソフトウェアに依存しません。その代わりに，各データベース仮想ユーザではサーバ API 関数を直接呼び出すスクリプトが実行されます。



データベース仮想ユーザ・スクリプトは，Windows 環境で VuGen を使って作成します。しかし，作成したスクリプトは，Windows および UNIX のどちらの環境でも仮想ユーザに割り当てることができます。スクリプトの記録の詳細については，第 3 章「VuGen を使った記録」を参照してください。

UNIX 環境では，LoadRunner のテンプレートを土台にしてスクリプトのプログラミングを行うことにより，DB 仮想ユーザ・スクリプトを作成できます。UNIX での DB 仮想ユーザ・スクリプトのプログラミングの詳細については，付録 D「UNIX プラットフォームでのスクリプトの プログラミング」を参照してください。

データベース仮想ユーザ・スクリプトの概要

本項では，VuGen を使ったデータベース仮想ユーザ・スクリプトの作成プロセスの概要を説明します。

データベース仮想ユーザ・スクリプトを作成するには，次の手順で行います。

1 **VuGen を使って基本となるスクリプトを記録します。**

VuGen を起動して，新しい仮想ユーザ・スクリプトを作成します。仮想ユーザのタイプ（「**クライアント / サーバ**」または「**ERP/CRM**」プロトコル・タイプ）を指定します。記録対象アプリケーションを選び，記録オプションを設定します。アプリケーションを使用した標準的な操作を記録します。

詳細については，第 3 章「VuGen を使った記録」を参照してください。

2 **スクリプトを拡張します。**

仮想ユーザ・スクリプトにトランザクション，ランデブー・ポイント，および制御フロー構造を挿入して，スクリプトを拡張します。

詳細については，第 6 章「仮想ユーザ・スクリプトの拡張」を参照してください。

3 **パラメータを定義します（任意）。**

スクリプトに記録された固定値に対してパラメータを定義します。固定値をパラメータで置き換えることで，同じクエリーを異なる値を使って反復実行できます。

詳細については，第 7 章「パラメータの定義」を参照してください。

4 **クエリーを相関させます（任意）。**

データベース・ステートメントを相関させることによって，クエリーの結果を以降の他のクエリーで使用できるようになります。この機能は，ユーザに制約が課せられるデータベースで作業をするときに有用です。

詳細については，第 8 章「ステートメントの相関」を参照してください。

5 **実行環境を設定します。**

実行環境を設定することによって，スクリプト実行時の仮想ユーザ・スクリプトの振る舞いを制御します。設定には，反復，ログ，タイミング情報などがあります。

詳細については，第 9 章「実行環境の設定」を参照してください。

6 **VuGen からスクリプトを実行します。**

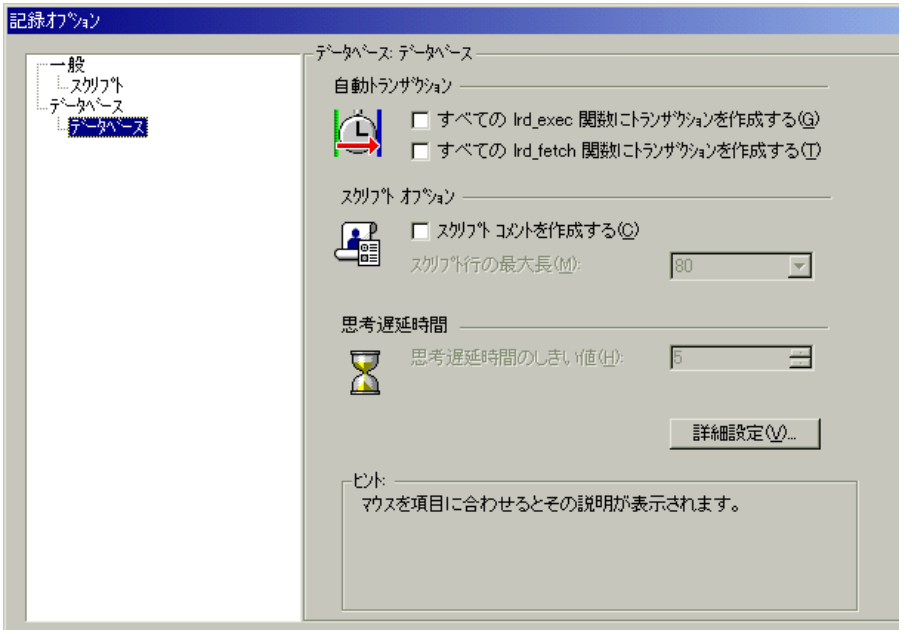
VuGen でスクリプトを保存して実行し，正しく動作することを確認します。

詳細については，第 11 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。

データベース仮想ユーザ・スクリプトが完成したら，Windows または UNIX プラットフォームで，スクリプトをシナリオに組み込みます。仮想ユーザ・スクリプトをシナリオに組み込む方法の詳細については，『**LoadRunner コントローラ・ユーザーズ・ガイド**』を参照してください。

データベース記録オプションの設定

データベース・セッションの記録を開始する前に，記録オプションを設定します。自動関数生成，スクリプト・オプション，思考遅延時間の基本的な記録オプションを設定できます。



[**自動トランザクション**]：**lrd_exec** 関数と **lrd_fetch** 関数をすべてトランザクションとしてマークするように VuGen に指定できます。これらのオプションを有効にすると，VuGen によってすべての **lrd_exec** 関数または **lrd_fetch** 関数の前後に **lr_start_transaction** と **lr_end_transaction** が挿入されます。標準設定では，自動トランザクションは無効になっています。

[**スクリプト・オプション**]：記録されたスクリプトに **lrd_stmt** のオプションの値を説明するコメントを生成するよう VuGen に指示できます。また，スクリプト行の最大長を指定できます。標準設定の長さは 80 文字までです。

[**思考遅延時間**]：VuGen ではオペレータの思考遅延時間が自動的に記録されます。思考遅延時間のしきい値を設定して，それよりも低い場合には思考遅延時間を無視するようにできます。記録された思考遅延時間がしきい値を越えていると，VuGen によって LRD 関数の前に **lr_think_time** ステートメントが挿入されます。記録された思考遅延時間がしきい値を越えない場合，**lr_think_time** ステートメントは生成されません。標準設定の値は 5 秒です。

データベース記録オプションを設定するには次のようにします。

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

2 **lrd_exec** ステートメントに対して自動トランザクションを有効にするには，［**すべての lrd_exec 関数にトランザクションを作成する**］を選択します。

lrd_fetch ステートメントに対して自動トランザクションを有効には，［**すべての lrd_fetch 関数にトランザクションを作成する**］を選択します。

3 スクリプトにわかりやすいコメントを挿入するように設定するには，［**スクリプト コメントを作成する**］を選択します。

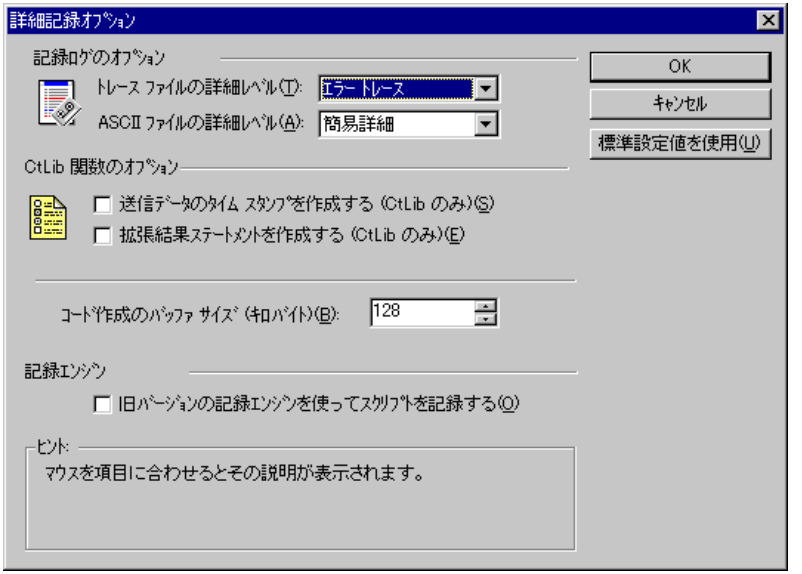
4 VuGen エディタの行の最大長を変更するには，［**スクリプト行の最大長**］ボックスで最大長を指定します。

5 思考遅延時間のしきい値を標準設定の 5 秒から変更するには，［**思考遅延時間のしきい値**］ボックスに値を指定します。

ログのトレース・レベル，CtLib 関数の生成，コード生成バッファに関して詳細な記録オプションを設定することもできます。

データベースの詳細記録オプション

基本的な記録オプションのほかに，ログ・ファイルの詳細，CtLib 固有の関数，バッファ・サイズ，記録エンジンについて詳細な記録オプションも設定できます。



［**記録ログのオプション**］：トレース・ファイルと ASCII ログ・ファイルの詳細レベルを設定できます。トレース・ファイルに対して指定できるレベルとしては，［**オフ**］，［**エラー トレース**］，［**簡易トレース**］，および［**完全トレース**］があります。［エラー トレース］に設定すると，エラー・メッセージだけがログに書き込まれます。［簡易レース］に設定すると，エラーのほかに，記録中に生成された関数のリストがログに書き込まれます。［完全トレース］に設定すると，メッセージ，告示，警告などがすべてログに書き込まれます。

記録セッションに関して ASCII タイプのログが生成されるようにも設定できます。指定できるレベルとしては，［**オフ**］，［**簡易詳細**］，［**完全詳細**］があります。［簡易詳細］設定では，すべての関数がログに書き込まれます。［完全詳細］設定では，生成されたすべての関数とメッセージが ASCII コードでログに書き込まれます。

［**CtLib 関数のオプション**］：送信データのタイム・スタンプと，拡張結果ステートメントが作成されるように設定できます。

- ［**データ送信のタイム スタンプを作成する**］：標準設定では，VuGen は **mpszReqSpec** パラメータに **TotalLen** キーワードと **Log** キーワードを設定して **lrd_send_data** ステートメントを生成します。［詳細記録オプション］ダイアログ・ボックスでは，**TimeStamp** キーワードも生成するように指定することができます。既存のスクリプトでこの設定を変更した場合は，［**ツール**］＞［**仮想ユーザを再生成**］を選択して仮想ユーザ・スクリプトを生成します。標準設定で **Timestamp** キーワードを生成することは推奨されません。記録中に生成されたタイム・スタンプは再生中に生成されるものと異なるため，スクリプトの実行が失敗するからです。このオプションは，スクリプトの実行時に **lrd_send_data** に続く **lrd_result_set** が失敗した場合にだけ使用します。オプションを指定すれば，生成されたタイム・スタンプと，**lrd_send_data** を実行して失敗したときのタイム・スタンプを相関できるようになります。
- ［**拡張結果ステートメントを作成する**］：標準設定では，結果セットを準備する段階で **lrd_result_set** 関数が生成されます。このオプションを選択すると，**lrd_result_set** 関数を拡張した関数である **lrd_result_set_ext** が生成されます。この関数は，結果セットを準備するほかに，**ct_results** からリターン・コードとタイプを発行します。

［**コード作成のバッファ サイズ**］：コード生成バッファの最大サイズをキロバイト単位で指定します。標準設定値は 128 キロバイトです。データベース・セッションが長い場合には，より大きなサイズを指定できます。

［**記録エンジン**］：VuGen バージョン 6.0 以上では，新しい記録エンジンが使用されています。VuGen に対して，旧バージョンの記録エンジンを使ってスクリプトを記録するように指示して，VuGen の旧バージョンと互換性を保つことができます。

詳細記録オプションを設定するには，次の手順で行います。

1 ［記録オプション］ダイアログ・ボックスの［データベース］ノードで［**詳細設定**］ボタンをクリックします。［詳細記録オプション］ダイアログ・ボックスが開きます。

2 ［**トレース ファイルの詳細レベル**］を選択します。トレース・ファイルを無効にするには［オフ］を選択します。

3 ASCII ログ・ファイルを生成するには，［**ASCII ファイルの詳細レベル**］ボックスから詳細レベルを選択します。

4 CtLib の場合のみ：**lrd_send_data** 関数に対する TimeStamp キーワードを生成させるには，［**送信データのタイム スタンプを作成する**］オプションを選択します。

5 CtLib の場合のみ：**lrd_result_set** の代わりに **lrd_result_set_ext** を生成させるには，［**拡張結果ステートメントを作成する**］オプションを選択します。

6 コード生成バッファのサイズを，標準設定の 128 キロバイトから変更するには，［**コード作成のバッファ サイズ**］ボックスに使用する値を入力します。

7 下位互換性を維持するために旧バージョンの VuGen の記録エンジンを使うには，［**旧バージョンの記録エンジンを使ってスクリプトを記録する**］オプションを選択します。

8 ［**OK**］をクリックして設定を保存して，［詳細記録オプション］ダイアログ・ボックスを閉じます。

LRD 関数の使用法

データベース・クライアントとサーバの間の通信をエミュレートするために作成された関数を LRD 仮想ユーザ関数といいます。LRD 仮想ユーザ関数の名前には，**lrd** という接頭辞が付いています。VuGen では，データベース・セッション（CtLib，DbLib，Informix，Oracle（2-Tier），および ODBC）の間に，この項で示すほとんどの LRD 関数を自動的に記録します。また，手作業でスクリプトに任意の関数をプログラミングすることもできます。LRD 関数の構文と使用例については，「**オンライン関数リファレンス**」（[**ヘルプ**] > [**関数リファレンス**]）を参照してください。

アクセス管理関数

lrd_alloc_connection	接続用の構造体を割り当てます。
lrd_close_all_cursors	開いているすべてのカーソルを閉じます。
lrd_close_connection	データベースから接続を解除（ログアウト）します。
lrd_close_context	コンテキストを閉じます。
lrd_close_cursor	データベース・カーソルを閉じます。
lrd_ctlib_cursor	CtLib カーソル・コマンドを指定します。
lrd_commit	現在のトランザクションをコミットします。
lrd_db_option	現在のデータベース用にオプションを設定します。
lrd_free_connection	接続用の構造体を解放します。
lrd_rollback	現在のトランザクションをロールバックします。
lrd_open_connection	データベースに接続（ログオン）します。
lrd_open_context	コンテキストを開きます。
lrd_open_cursor	データベース・カーソルを開きます。

LRD 環境関数

lrd_msg	出力メッセージを発行します。
lrd_option	LRD オプションを設定します。
lrd_end	lrd 環境を閉じます。
lrd_init	lrd 環境を初期化します。

検索処理関数

lrd_col_data	データの所在を示すポインタを設定します。
lrd_fetch	結果セットから次の行を取り出します。
lrd_fetchx	拡張フェッチ機能を使って，結果セットから次の行を取り出します（ODBC のみ）。
lrd_result_set	結果セットを返します（CtLib のみ）。
lrd_result_set_ext	CtLib 結果コードと結果タイプを返します（lrd_result_set を拡張したもの）。
lrd_fetch_adv	拡張フェッチ機能を使って，結果セットから複数の行を取り出します（ODBC のみ）。
lrd_reset_rows	更新作業のために取り出された行を準備します（ODBC のみ）。
lrd_row_count	UPDATE，DELETE または INSERT ステートメントによって影響を受けた行の数を返します。（ODBC，DB2）。

ステートメント処理関数

lrd_bind_col	出力カラムにホスト変数をバインドします。
lrd_bind_cols	ホスト変数配列をカラムにバインドします。
lrd_bind_cursor	カーソルをプレースホルダにバインドします。
lrd_bind_placeholder	ホスト変数またはホスト配列をプレースホルダにバインドします。
lrd_cancel	前回のステートメントを取り消します。
lrd_data_info	入出力情報を取得します（CtLib のみ）。
lrd_dynamic	処理対象の動的 SQL ステートメントを定義します（CtLib のみ）。
lrd_exec	事前に指定した SQL ステートメントを実行します。
lrd_send_data	サーバにデータを送信します。
lrd_stmt	処理対象の SQL ステートメントを定義します。

ステートメント相関関数

lrd_save_col	テーブル・セルの値をパラメータに保存します。
lrd_save_value	プレースホルダ記述子の値をパラメータに保存します。
lrd_save_ret_param	戻りパラメータの値をパラメータに保存します（CtLib のみ）。
lrd_save_last_rowid	最後の rowid をパラメータに保存します（Oracle）。

変数処理関数

lrd_assign	NULL で終了する文字列を変数に割り当てます。
lrd_assign_ext	変数に記憶領域を割り当てます。
lrd_assign_literal	リテラル文字列（NULL 文字を含む）を変数に割り当てます。
lrd_assign_bind	NULL で終了する文字列を変数に割り当て，プレースホルダにバインドします。
lrd_assign_bind_ext	変数に記憶領域の値を割り当て，プレースホルダにバインドします。
lrd_assign_bind_literal	リテラル文字列（NULL 文字を含む）を変数に割り当て，プレースホルダにバインドします。
lrd_to_printable	変数を印字可能な文字列に変換します。

Siebel 関数

lrd_siebel_incr	文字列を指定の値の分だけインクリメントします。
lrd_siebel_str2num	底が 36 の文字列を底が 10 の数値に変換します。
SiebelPostSave_x	Siebel のパラメータの変化後の値を保存します。
SiebelPreSave_x	相関の対象となるパラメータを指定する。

Oracle 8 関数

VuGen は，Oracle 8.x を部分的にサポートしています。Oracle の前のバージョンで記録されていたデータベース・アクションはすべて記録されます。多くの場合，記録される関数は Oracle 8.x に固有のものです。たとえば，フェッチ操作に対しては **lrd_fetch** の代わりに **lrd_ora8_fetch** を記録します。

lrd_attr_set	LRDDBI ハンドルの属性を設定します。
lrd_attr_set_from_handle	LRDDBI ハンドル・ポインタを使用して属性を設定します。
lrd_attr_set_literal	リテラル文字列を使用して LRDDBI ハンドル属性を設定します。
lrd_env_init	LRDDBI ハンドルを割り当て，初期化します。
lrd_handle_alloc	LRDDBI ハンドルを明示的に割り当て，初期化します。
lrd_handle_free	LRDDBI ハンドルを明示的に解放します。
lrd_initialize_db	データベース処理環境を初期化します。
lrd_logoff	単純なデータベース・セッションを終了します。
lrd_logon	単純なデータベース・セッションを開始します。
lrd_logon_ext	（拡張された）単純なデータベース・セッションを開始します。
lrd_ora8_attr_set	LRDDBI ハンドルの属性を設定します（省略形式）。
lrd_ora8_attr_set_from_handle	LRDDBI ハンドル・ポインタを使用して属性を設定します。
lrd_ora8_attr_set_literal	リテラル文字列を使用して LRDDBI ハンドル属性を設定します（省略形式）。
lrd_ora8_bind_col	出力カラムにホスト変数をバインドします。
lrd_ora8_bind_placeholder	プレースホルダにホスト変数をバインドします。
lrd_ora8_commit	Oracle 8.x クライアントの現在のトランザクションをコミットします。
lrd_ora8_exec	Oracle 8.x で SQL ステートメントを実行します。

lrd_ora8_fetch	結果セットから次の行を取り出します。
lrd_ora8_handle_alloc	LRDDBI ハンドルを明示的に割り当て，初期化します（省略形式）。
lrd_ora8_rollback	Oracle 8.x クライアントの現在のトランザクションをロールバックします。
lrd_ora8_save_col	テーブル・セルの値をパラメータに保存します。
lrd_ora8_stmt	NULL で終了する SQL ステートメントを実行用に準備します。
lrd_ora8_stmt_ext	NULL 文字を含む SQL ステートメントを実行用に準備します。
lrd_ora8_stmt_literal	リテラル SQL ステートメントを実行用に準備します。
lrd_server_attach	データベース操作の対象となるデータ・ソースへのアクセス・パスを作成します。
lrd_server_detach	データベース操作の対象となるデータ・ソースへのアクセス・パスを削除します。
lrd_session_begin	サーバを対象にしたユーザ・セッションを作成し，開始します。
lrd_session_end	サーバを対象にしたユーザ・セッションを終了します。

データベース仮想ユーザ・スクリプトについて

データベース・セッションを記録した後，VuGen に組み込まれているエディタを使って，記録されたコードを表示できます。スクリプトをスクロールして，アプリケーションによって生成された SQL ステートメントを確認したり，サーバから返されたデータを調べたりできます。VuGen ウィンドウには，記録されたデータベース・セッションに関する以下の情報が表示されます。

- 記録された関数の並び
- データベース・クエリーによって返されたデータを表示するグリッド
- クエリー中に取り出された行の数

関数の並び

VuGen ウィンドウに仮想ユーザ・スクリプトを表示すると，VuGen によって記録した操作のシーケンスを見ることができます。たとえば，標準的な Oracle データベース・セッションでは，次のような関数の並びが記録されます。

lrd_init	環境を初期化します。
lrd_open_connection	データベース・サーバに接続します。
lrd_open_cursor	データベース・カーソルを開きます。
lrd_stmt	SQL ステートメントとカーソルを関連付けます。
lrd_bind_col	ホスト変数をカラムにバインドします。
lrd_exec	SQL ステートメントを実行します。
lrd_fetch	結果セットから次の記録を取り出します。
lr_commit	データベース・トランザクションをコミットします。
lr_close_cursor	カーソルを閉じます。
lrd_close_connection	データベース・サーバから接続を解除します。
lrd_end	環境をクリーンアップします。

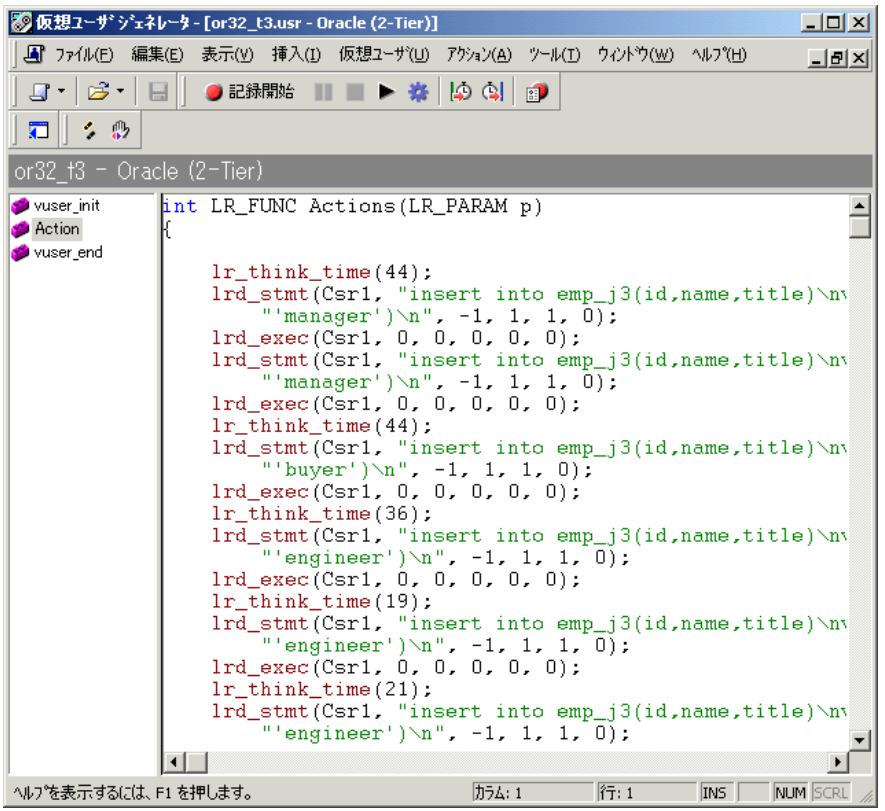
次に示すスクリプトは，Oracle サーバへの接続を開いて，ローカル設定を要求するクエリーを実行したオペレータのアクションを VuGen で記録したものです。

```
lrd_init(&InitInfo, DBTypeVersion);
lrd_open_connection(&Con1, LRD_DBTYPE_ORACLE, "s1", "tiger",
"hp1", "", 0, 0, 0);
lrd_open_cursor(&Csr1, Con1, 0);
lrd_stmt(Csr1, "select parameter, value     from v$nls_parameters "
      "    where (upper(parameter) in ('NLS_SORT','NLS_CURRENCY',"
      "'NLS_ISO_CURRENCY', 'NLS_DATE_LANGUAGE',"
      "'NLS_TERRITORY'))", -1, 0 /*Non deferred*/, 1 /*Dflt Ora Ver*/, 0);
lrd_bind_col(Csr1, 1, &D1, 0, 0);
lrd_bind_col(Csr1, 2, &D2, 0, 0);
lrd_exec(Csr1, 0, 0, 0, 0, 0);
lrd_fetch(Csr1, 7, 7, 0, PrintRow2, 0);
 . . . . . lrd_close_cursor(&Csr1, 0);
lrd_commit(0, Con1, 0);
lrd_close_connection(&Con1, 0, 0);
lrd_end(0);
```

グリッド

記録セッション中にデータベース・クエリーから返されたデータはグリッドに表示されます。グリッドを参照することで，アプリケーションによって SQL ステートメントがどのように生成されたかを確認したり，クライアント / サーバ・システムの効率を把握したりできます。

次の例では，ウィンドウに employee データベースを対象に実行されたクエリーが表示されています。クエリーでは，「**engineer**」という肩書きを持つすべての従業員の，名前，ID 番号，および肩書きの情報を取り出しています。



ウィンドウの表示枠は，幅を調整することが可能です。また，スクロール・バーを使って，最高 100 行までスクロールできます。

表示枠の表示 / 非表示を切り替えるには，［**表示**］＞［**データ グリッド**］を選択します。

行情報

VuGen によって，SQL クエリーごとに **lrd_fetch** 関数が生成されます。

```
lrd_fetch(Csr1, -4, 1, 0, PrintRow7, 0);
```

関数の 2 番目のパラメータは，取り出される行の数を示します。正数，負数のどちらも指定できます。

正数の値

正数の値は記録中に取り出された行の数を示し，すべての行が取り出されていないことを示します（オペレータがクエリー完了前にクエリーを取り消した場合など）。

次の例では，データベース・クエリーの実行時に 4 行を取り出していますが，すべてのデータは取り出していません。

```
lrd_fetch(Csr1, 4, 1, 0, PrintRow7, 0);
```

実行時には，正数によって示される数の行が必ずスクリプトによって取り出されます（行が存在することが前提）。

負数の値

行の値が負数の場合，記録時にすべての行が取り出されたことを示します。負数の絶対値が，取り出された行の数です。

次の例では，結果セットの 4 行すべてを取り出しています。

```
lrd_fetch(Csr1, -4, 1, 0, PrintRow7, 0);
```

負数の値を含む **lrd_fetch** ステートメントを実行すると，実行時にテーブルから取り出せる行がすべて取り出されます。必ずしも記録時の行数と同じではありません。上の例では，テーブルの 4 行すべてが記録時に取り出されています。しかし，スクリプト実行時に，それ以上の行数がある場合は，それらがすべて取り出されます。

lrd_fetch の詳細については，「**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。

エラー・コードの分析

仮想ユーザが LRD 関数を実行すると，関数によって必ずリターン・コードが生成されます。リターン・コード「0」は，関数が成功したことを示します。たとえば，リターン・コード「0」は，結果セットに利用可能な行が残っていることを示します。エラーが発生した場合，リターン・コードはエラーの種類を表します。たとえば，リターン・コード「2014」は，初期化中にエラーが発生したことを表します。

リターン・コードは 4 種類に分類され，それぞれ数値の範囲が決まっています。

リターン・コードの種類	範囲
情報	0 ～ 999
警告	1000 ～ 1999
エラー	2000 ～ 2999
内部エラー	5000 ～ 5999

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

LRD 関数のリターン・コードを調べて，関数が成功したかどうかを確認できます。以下のスクリプトでは，**lrd_fetch** 関数のリターン・コードを評価しています。

```
static int rc;
rc=lrd_fetch(Csr15, -13, 0, 0, PrintRow4, 0);
if (rc==0)
      lr_output_message("The function succeeded");
else
      lr_output_message("The function returned an error code:%d",rc);
```

エラー処理

データベース仮想ユーザ・スクリプトの実行時に，データベース仮想ユーザにエラーをどのように処理させるかを制御できます。標準設定では，スクリプト実行時にエラーが発生すると，スクリプトの実行が終了します。仮想ユーザの標準の振る舞いを変更するには，エラーが発生しても処理を継続するように設定します。次の範囲を対象に振る舞いを設定できます。

- グローバル ——スクリプト全体，またはスクリプトの一部に設定
- ローカル —— 特定の関数のみ

エラー処理のグローバル変更

LRD_ON_ERROR_CONTINUE または LRD_ON_ERROR_EXIT ステートメントを発行することによって，仮想ユーザのエラー処理の方法を変更できます。標準設定では，データベース関連であれパラメータ関連であれ，エラーが生じると仮想ユーザによるスクリプトの実行が中止されます。この標準設定の振る舞いを変更するには，スクリプトに以下の行を挿入します。

```
LRD_ON_ERROR_CONTINUE;
```

以降，仮想ユーザでエラーが発生してもスクリプトの実行が継続されます。

また，スクリプトの特定のセグメントだけでエラーが発生する場合に，仮想ユーザにスクリプトの実行を継続するように指定することもできます。たとえば，以下のコードは，**lrd_stmt** や **lrd_exec** の関数内でエラーが発生した場合は，スクリプトの実行を継続するように仮想ユーザに指示します。

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

LRD_ON_ERROR_CONTINUE ステートメントを使うときは，重要かつ重大なエラーを見逃してしまう可能性があるので注意が必要です。

エラー処理のローカル変更

選択した関数の重要度を変更してエラー処理を設定できます。**lrd_stmt** や **lrd_exec** といったデータベース処理を実行する関数は，重要度を使用します。重要度は関数の最後のパラメータである **miDBErrorSeverity** で示します。このパラメータは，データベース・エラー（エラー・コード 2009）が発生した場合に，仮想ユーザにスクリプトの実行を継続させるかどうかを指示するものです。標準設定の「0」は，エラー発生時に仮想ユーザによるスクリプトの実行を中止することを示します。

たとえば，以下のデータベース・ステートメントが失敗した場合は（たとえばテーブルが存在しない場合など），スクリプトの実行は中止されます。

```
lrd_stmt(Csr1, "insert into EMP values ('Smith',301)¥n", -1, 1 /*Deferred*/,
1 /*Dflt Ora Ver*/, 0);
```

データベース処理でエラーが発生した場合でも，仮想ユーザにスクリプトの実行を継続するように指示するには，ステートメントの重要度パラメータを「0」から「1」に変更します。

```
lrd_stmt(Csr1, "insert into EMP values ('Smith',301)¥n", -1, 1 /*Deferred*/,
1 /*Dflt Ora Ver*/, 1);
```

重要度を「1」に設定した場合，データベース・エラーが発生すると，警告が表示されます。特定の関数に重要度レベルを設定すると，その重要度レベルはその関数にだけ適用されることに注意してください。

CtLib 結果セット・エラー

CtLib の記録時には，アプリケーションによってステートメントが実行された後，利用可能な結果セットがすべて取り出されます。返された結果セットに取り出し可能なデータが含まれている場合は，アプリケーションでそのデータを対象にバインドおよび取り出しの操作が行われます。次に例を示します。

```
lrd_stmt(Csr15, "select * from all_types", -1, 148, -99999, 0);
lrd_exec(Csr15, 0, 0, 0, 0, 0);
lrd_result_set(Csr15, 1 /*Succeed*/, 4040 /*Row*/, 0);
lrd_bind_col(Csr15, 1, &tinyint_D41, 0, 0);
. . .
lrd_fetch(Csr15, -9, 0, 0, PrintRow3, 0);
```

結果セットに取り出し可能なデータが含まれていない場合，バインドと取り出しの操作は行えません。

スクリプトをパラメータ化すると，パラメータによっては結果データが取り出せなくなることがあります。新しいデータを取り出せない場合，特定のステートメントに対するバインドと取り出しの操作を記録した CtLib セッションは，実行できないことがあります。**lrd_bind_col** 関数または **lrd_fetch** 関数を実行しようとすると，エラーが発生し（LRDRET_E_NO_FETCHABLE_DATA：エラー・コード 2064），仮想ユーザによるスクリプトの実行が終了します。

このタイプのエラーが発生したときに，仮想ユーザにスクリプト実行の継続を指示することにより，実行を優先させることができます。次の行をスクリプトに挿入します。

```
LRD_ON_FETCHABLE_SET_ERR_CONT;
```

エラーの発生時にスクリプトの実行が終了する標準設定のモードに戻すには，次の行をスクリプトに入力します。

```
LRD_ON_FETCHABLE_SET_ERR_EXIT;
```

このステートメントを使うときは，重要かつ重大なエラーを見逃してしまう可能性があるので注意が必要です。

第 19 章

データベース仮想ユーザ・スクリプトの相関

データベース・セッションの記録後に，スクリプト内の 1 つ以上のクエリーを相関させる必要が生じることがあります。これによって，データベース・セッション中に取得された値を，セッション内の以降の処理で使用できるようになります。

本章では，次の項目について説明します。

- スクリプトでの相関候補の検索
- 既知の値の相関
- データベース仮想ユーザ相関関数

以降の情報は，データベース（CtLib，DbLib，Informix，Oracle，ODBC，DB2-CLI）仮想ユーザ・スクリプト対象とします。

データベース仮想ユーザ・スクリプトの相関について

スクリプト実行時にエラーが発生した場合は，スクリプトの中でエラーが発生した場所を調べます。多くの場合，相関クエリーによって，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。データベース仮想ユーザ・スクリプトを相関させる必要が生じる理由は，主に次の 2 つです。

- 重複する値を使用できない

  たとえば，スクリプトによってデータベースに新しい従業員レコードが作成され，各従業員に一意の ID が割り当てられるとします。データベースでは，各レコードは一意でなければならず，重複があってはいけません。このスクリプトを再生しようとすると，従業員の ID が記録セッション中にすでに作成されており，同じレコードを作成できないために，再生が失敗します。

この問題を解決するには，相関を行って，新しい従業員に割り当てられた ID を取得し，その ID を以降のデータベース・セッションで使用します。さらに，パラメータ化を行って，従業員ごとに一意のデータを取得することもできます。詳細については，第 7 章「パラメータの定義」を参照してください。

➤ コードを簡素化または最適化するため

たとえば，相互に依存する一連のクエリーを連続して実行すると，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ーをネストすることもできますが，精度が損なわれたり，コードが複雑になっ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，ネストせずにクエリーを連結できます。

スクリプトでの相関候補の検索

VuGen には，スクリプトを修正して確実に再生できるようにするための相関ユーティリティがあります。相関ユーティリティは，次の処理を行います。

➤ 相関候補を探す。

➤ 適切な相関関数を挿入して，結果をパラメータに保存する。

➤ ステートメントの値をパラメータで置き換える。

自動相関は，スクリプト全体またはスクリプト内の特定の場所を対象に実行できます。

本項では，相関させる必要のあるステートメントを見つける方法について説明します。すでに相関させたい値がわかっている場合は，次の項に進んで，特定の値を相関させる方法を参照してください。

自動相関によるスクリプトの検索と相関は，次の手順で行います。

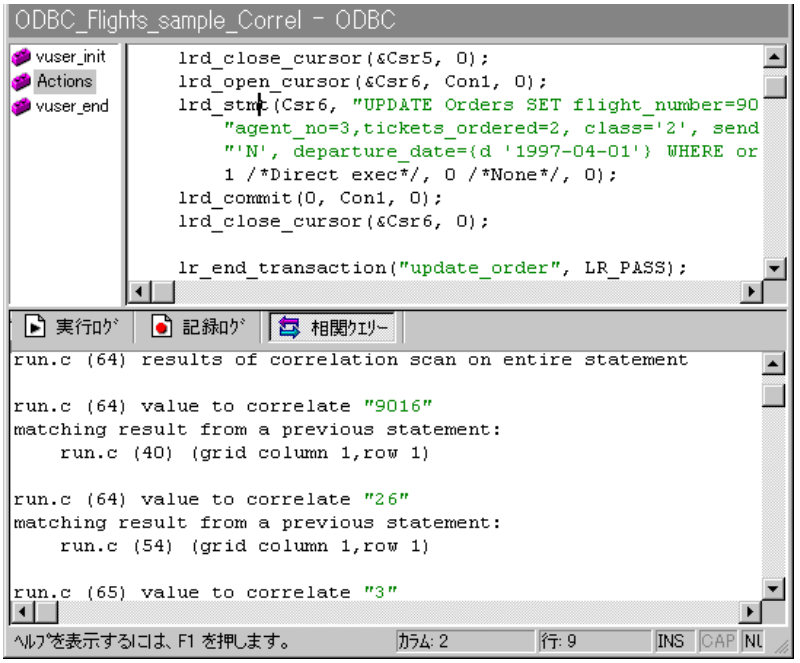
1 ［出力］ウィンドウを開きます。

［**表示**］＞［**出力ウィンドウ**］を選択して，ウィンドウの下部に出力タブを表示します。［実行ログ］タブでエラーがないか確認します。多くの場合，これらのエラーは相関によって修正できます。

2 ［**仮想ユーザ**］＞［**相関を検索**］を選択します。

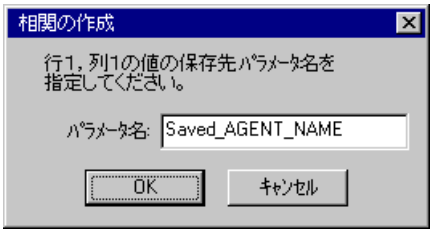
VuGen によってスクリプト全体を対象とする検索が行われ，相関候補の値がすべて［**相関クエリー**］タブに表示されます。

次の例では，**lrd_ora8_stmt_literal** ステートメントの中で，相関させる必要のある値が検出されています。



3 ［相関クエリー］タブで，相関させるクエリー結果をダブルクリックします。この例では，メッセージの 3 行目にあります。3 行目は「grid column x, row x」のようになっています。VuGen によって，スクリプト内の値の位置にカーソルが移動します。

4 表示枠の中で，相関させる値を選び，［**仮想ユーザ**］＞［**相関を作成**］を選択します。結果の値を保存するパラメータの名前を入力するように求められます。



5 名前を指定するか，標準設定の名前をそのまま使用します。［**OK**］をクリックして作業を続けます。VuGen によって，パラメータに結果を保存する適切な相関ステートメント（**lrd_save_value**，**lrd_save_col**，**lrd_save_ret_param** のいずれか）が挿入されます。

6 ［**はい**］をクリックして相関を確定します。

スクリプト内に現れる値をすべて検索するかどうかを尋ねるメッセージが表示されます。

7 選択したステートメント内の値だけを置き換える場合は，［**いいえ**］をクリックします。

8 次の候補を検索する場合は，［**はい**］をクリックします。［検索と置換］ダイアログ・ボックスが開きます。

9 オリジナルのステートメントも含め，すべての置き換えを確認します。［検索と置換］ダイアログ・ボックスを閉じます。

ステートメントの値がパラメータへの参照に置き換えられます。相関を取り消すと，直前のステップで作成されたステートメントは消去されます。

既知の値の相関

相関させる必要がある値がわかっている場合は，次の手続きを行います。

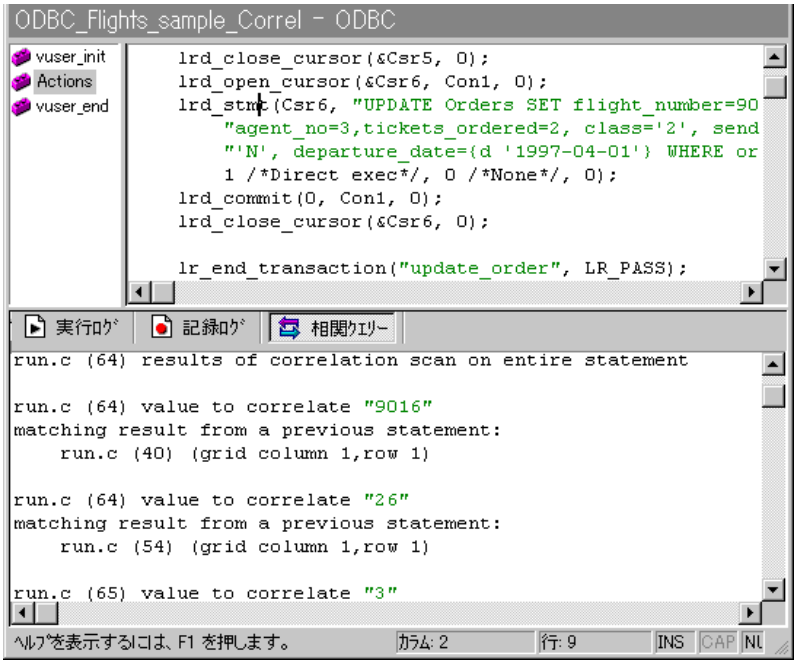
特定の値を相関させるには，次の手順で行います。

1 相関させる値を見つけ，値を選択します（引用符は除きます）。

2 ［**仮想ユーザ**］＞［**アクション内で相関をスキャン**］を選択します。

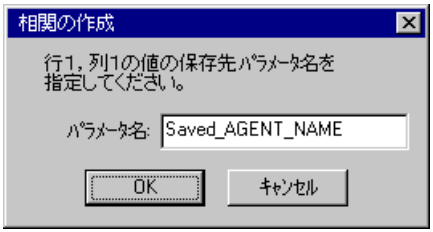
VuGen によって値が検索され，この値と一致するスクリプト内の結果がすべて表示されます。相関値が［相関クエリー］タブに一覧表示されます。

次の例では，「20」に相関させる候補として複数の結果値が見つかっています。



3 ［相関クエリー］タブで，相関させる結果をダブルクリックします。この例では，メッセージの 3 行目にあります。3 行目は「grid column x, row x」のようになっています。VuGen によって，スクリプト内の値の位置にカーソルが移動します。

4 表示枠の中で，相関させる値を選び，［**仮想ユーザ**］＞［**相関を作成**］を選択します。パラメータ名を入力するように求められます。



5 名前を指定するか，標準設定の名前をそのまま使用します。［**OK**］をクリックして作業を続けます。VuGen によって，パラメータに結果を保存する適切な相関ステートメント（**lrd_save_value**，**lrd_save_col**，**lrd_save_ret_param** のいずれか）が挿入されます。

6 ［**はい**］をクリックして相関を確定します。

スクリプト内に現れる値をすべて検索するかどうかを尋ねるメッセージが表示されます。

7 選択したステートメント内の値だけを置き換える場合は，［**いいえ**］をクリックします。

8 次の候補を検索する場合は［**はい**］をクリックします。［検索と置換］ダイアログ・ボックスが開きます。

9 オリジナルのステートメントも含め，すべての置き換えを確認します。［検索と置換］ダイアログ・ボックスを閉じます。

ステートメントの値がパラメータへの参照に置き換えられます。相関を取り消すと，直前のステップで作成されたステートメントは消去されます。

注：lrd_stmt 関数の値を相関させている場合は，データ型 date，time，binary（RAW，VARRAW）はサポートされません。

データベース仮想ユーザ相関関数

データベース仮想ユーザ・スクリプト（DbLib，CtLib，Oracle，Informix など）を使用するときに，VuGen の自動相関機能を使用して，スクリプトに適切な関数を挿入できます。相関関数は次のとおりです。

- **lrd_save_col** 関数では，表示枠内に表示されるクエリー結果がパラメータに保存されます。この関数はデータの取り出しの前に配置されます。lrd_save_col 関数によって，それ以降に **lrd_fetch** によって取り出された値が指定されたパラメータに代入されます。
- **lrd_save_value** 関数では，プレースホルダ記述子の現在値がパラメータに保存されます。出力プレースホルダを設定するデータベース関数（Oracle の特定のストアド・プロシージャなど）と一緒に使用します。
- **lrd_save_ret_param** 関数では，ストアド・プロシージャの戻り値がパラメータに保存されます。この関数は主に，戻り値を生成する DbLib 内のデータベース・プロシージャと組み合わせて使用します。

注：VuGen では，保存された値が無効または NULL（行が返されない）の場合，相関が適用されません。

これらの関数と引数の詳細については，「**オンライン関数リファレンス**」を参照してください。

第 20 章

DNS 仮想ユーザ・スクリプトの作成

VuGen では，DNS サーバに直接アクセスしてネットワークの動作状態をエミュレートできます。

本章では，次の項目について説明します。

- DNS 関数を使った作業

以降の情報は，DNS 仮想ユーザ・スクリプトを対象とします。

DNS 仮想ユーザ・スクリプトの作成について

DNS プロトコルは，低レベルのプロトコルで，DNS サーバに対して行うユーザのアクションをエミュレートします。

DNS プロトコルは，Domain Name Server にアクセスし，IP アドレスでホスト名を解決するユーザをエミュレートします。このプロトコルでは，再生機能のみサポートされているため，手作業でスクリプトに関数を追加します。

DNS プロトコルのスクリプトを作成するには，［**ファイル**］＞［**新規作成**］を選択して［新規仮想ユーザ］ダイアログ・ボックスを開きます。［クライアント / サーバ］カテゴリから［Domain Name Resolution（DNS）］を選択します。DNS プロトコルについてはスクリプトの記録ができないため，適切な DNS 関数，LoadRunner 関数，C 関数を使ってスクリプトをプログラミングします。これらの関数の詳細については，「**オンライン関数リファレンス**」（［**ヘルプ**］＞［**関数リファレンス**］）を参照してください。

仮想ユーザ・スクリプトを作成したら，Windows または UNIX プラットフォームのいずれかでシナリオに組み込みます。仮想ユーザ・スクリプトのシナリオへの組み込み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**』を参照してください。

DNS 関数を使った作業

DNS 仮想ユーザ・スクリプト関数ではドメイン名解決サーバ（DNS）を往復するクエリーが記録されます。DNS 関数はどれも接頭辞 **dns** で始まります。これらの関数の構文についての詳細は，「**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。

関数名	説明
ms_dns_query	ホストの IP アドレス解決をします。
ms_dns_nextresult	**ms_dns_query** 関数によって返される IP アドレス・リスト内の次の IP アドレスに進みます。

次の例では，クエリーを DNS サーバに送信し，その結果をログ・ファイルに出力しています。

```
Actions()
{
int    rescnt = 0;
       char results = NULL;
       results = (char *) ms_dns_query("transaction",
                                       "URL=dns:// < Dns サーバ> ",
                                       "QueryHost= <ホスト名> ",
                                       LAST);

       // ホスト名の全 IP アドレスを表示 ..
       while (*results) {
            rescnt++;
            lr_log_message(lr_eval_string("(%d) IP of < Hostname > is %s"),
                     rescnt, results);
            results = (char *) ms_dns_nextresult(results);
       }
       return 1;
}
```

第 21 章

WinSock 仮想ユーザ・スクリプトの作成

VuGen を使って，Windows Sockets プロトコルでやり取りするクライアント・アプリケーションとサーバ間の通信を記録することができます。その結果生成されるスクリプトを，Windows Sockets 仮想ユーザ・スクリプトと呼びます。

本章では，次の項目について説明します。

- Windows Sockets 仮想ユーザ・スクリプト入門
- WinSock 記録オプションの設定
- LRS 関数の使用
- ツリー・ビューとスクリプト・ビューの切り替え

以降の情報は，Web/Winsock Dual Protocol など，Windows Sockets レベルで記録されるすべてのプロトコルを対象とします。

Windows Sockets 仮想ユーザ・スクリプトの記録について

Windows Sockets プロトコルは，アプリケーションの低レベルのコードを分析するのに適したプロトコルです。たとえば，ネットワークの検査を行うために，Windows Sockets（WinSock）スクリプトを使って，バッファによって送受信される実際のデータを見ることができます。また，WinSock プロトコルは 他の低レベルの通信セッションを記録するためにも使用できます。さらに，他のタイプの仮想ユーザでサポートされていないアプリケーションの記録と再生もできます。

Windows Sockets プロトコルを使用するアプリケーションを記録すると，記録されたアクションを表す関数が生成されます。各関数には，**lrs** という接頭辞が付きます。LRS 関数は，ソケット，データ・バッファ，および Windows Sockets 環境に対応します。VuGen を使って，アプリケーションの Winsock.dll または Wsock32.dll に対する API 呼び出しを記録します。たとえば，*telnet* アプリケーションのアクションを記録して，スクリプトを作成できます。

次に示す例では，**lrs_send** を使って指定したソケットにデータを送信しています。

```
lrs_send("socket22", "buf44", LrsLastArg);
```

記録されたスクリプトは，VuGen のメイン・ウィンドウで表示および編集ができます。このウィンドウには，セッション中に記録された Windows Sockets API 呼び出しが表示されるので，記録時のネットワークの動作状況を追うことができます。

VuGen では，WinSock スクリプトを次の 2 つの方法で表示できます。

- スクリプトをアイコン形式で表示する「**ツリー・ビュー**」（標準設定）。
- スクリプトをテキスト形式で表示して Windows Sockets API 呼び出しを示す**スクリプト・ビュー**。

VuGen では，スクリプトをツリー・ビューとスクリプト・ビューの両方で表示して編集できます。2 つのビューは簡単に切り替えられます。詳細については，299 ページ「ツリー・ビューとスクリプト・ビューの切り替え」を参照してください。

Windows Sockets 仮想ユーザ・スクリプト入門

本項では，VuGen を使って Windows Sockets 仮想ユーザ・スクリプトを開発する工程の概略を説明します。

Windows Sockets スクリプトの開発は，次の手順で行います。

1 **VuGen を使ってアクションを記録します。**

VuGen を起動し，Windows Sockets タイプを指定して，新しい仮想ユーザ・スクリプトを作成します。記録対象のアプリケーションを選び，記録オプションを設定します。選択したアプリケーションで標準的な操作を記録します。

詳細については，第 3 章「VuGen を使った記録」を参照してください。

2 **仮想ユーザ・スクリプトを拡張します。**

仮想ユーザ・スクリプトにトランザクション，ランデブー・ポイント，およびフロー制御構造を挿入して，スクリプトを拡張します。

詳細については，第 6 章「仮想ユーザ・スクリプトの拡張」を参照してください。

3 **パラメータを定義します（任意）。**

仮想ユーザ・スクリプトに記録された固定値に対してパラメータを定義します。固定値をパラメータに置き換えることによって，異なる値を使って同じビジネス・プロセスを反復実行することができます。

詳細については，第 7 章「パラメータの定義」を参照してください。

4 **ステートメントを相関させます（任意）。**

ステートメントを相関することによって，あるビジネス・プロセスの結果を以降のビジネス・プロセスで使用できます。

詳細については，第 8 章「ステートメントの相関」を参照してください。

5 **実行環境を設定します。**

実行環境の設定で，スクリプト実行時の仮想ユーザの動作を制御します。設定には，ループ，ログ，タイミング情報があります。

詳細については，第 9 章「実行環境の設定」と第 10 章「インターネット実行環境の設定」を参照してください。

6 **VuGen から仮想ユーザ・スクリプトを実行します。**

VuGen で生成した仮想ユーザ・スクリプトを保存して実行し，正しく動作することを確かめます。

詳細については，第 11 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。

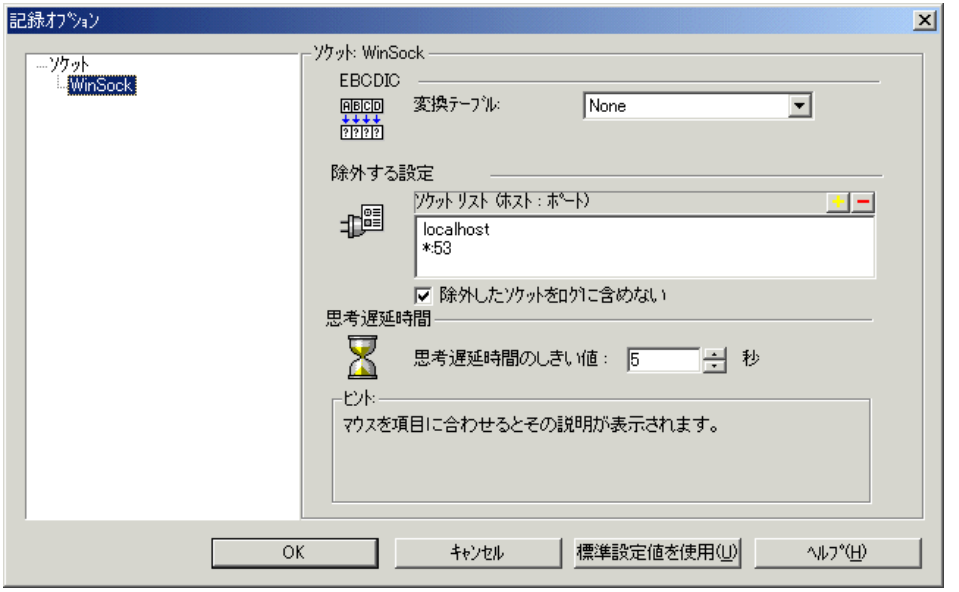
作成した Windows Sockets 仮想ユーザ・スクリプトを Windows または UNIX プラットフォームのシナリオに組み込みます。仮想ユーザ・スクリプトをシナリオに組み込む方法の詳細については，『**LoadRunner コントローラ・ユーザーズ・ガイド**』を参照してください。

WinSock 記録オプションの設定

WinSock 仮想ユーザに対しては，以下の記録オプションを使用できます。

- 変換テーブルの設定
- ソケットの除外
- 思考遅延時間のしきい値の設定

［記録オプション］ダイアログ・ボックスを開くには，［**ツール**］＞［**記録オプション**］を選択するか，［記録開始］ダイアログ・ボックスで［**オプション**］ボタンをクリックします。WinSock 用のオプションが表示されます。



変換テーブルの設定

EBCDIC 形式のデータを表示するには，［記録オプション］で変換テーブルを指定します。

［変換テーブル］で記録セッションの形式を指定できます。この設定は，メインフレーム・マシンや AS/400 サーバで実行しているユーザに適用されます。サーバ・マシンおよびクライアント・マシンは，システムにインストールされている変換テーブルに基づいて，データの形式を判断します。リスト・ボックスから変換オプションを選択します。

リスト・ボックスの項目の最初の 4 桁はサーバの形式を表します。後半の 4 桁はクライアントの形式を表します。上の例で選択した変換テーブルは **002501b5** です。サーバの形式が **0025**，クライアントの形式が **01b5** で，サーバからクライアントへの送信を表しています。クライアントからサーバへの送信の場合，形式を逆転した項目「**01b50025**」を選択することで，クライアントの **01b5** 形式をサーバの **0025** 形式に変換する必要があることを指定します。

変換テーブルは，LoadRunner インストール先ディレクトリの **ebcdic** ディレクトリにあります。システムで別の変換テーブルを使用している場合，テーブルを **ebcdic** ディレクトリにコピーします。

注：データが ASCII 形式の場合，変換の必要はありません。［**None**］オプション（標準設定）を選択します。変換テーブルを選択した場合は，VuGen は ASCII データを変換します。

Solaris マシンで作業しているときは，仮想ユーザを実行するすべてのマシンで次の環境変数を設定する必要があります。

```
setenv LRSDRV_SERVER_FORMAT 0025
setenv LRSDRV_CLIENT_FORMAT 04e4
```

ソケットの除外

VuGen ではソケット除外機能を使用して，記録セッションから特定のソケットを除外できます。スクリプトから特定のソケットを対象とするすべてのアクションをから除外するには，［除外する設定］の［ソケット リスト］からそのソケットのアドレスを選択します。ソケットをこのリストに追加するには，ボックスの右上角にあるプラス記号をクリックし，ソケットのアドレスを次のいずれかの形式で入力します。

値	意味
ホスト : ポート	指定したホストの指定したポートだけを除外します。
ホスト	指定したホストのすべてのポートを除外します。
: ポート	ローカル・ホストの指定したポートを除外します。
*: ポート	すべてのホストの指定したポートを除外します。

複数のホストとポートを除外するには，それらをリストに追加します。除外対象リストからソケットを削除するには，ソケットのアドレスを選択し，ボックスの右上角にあるマイナス記号をクリックします。ローカル・ホストや DNS ポート（53）など，テスト対象サーバの負荷に影響しないホストとポートを除外することをお勧めします。これらは標準設定では除外されています。

標準設定では，［**除外する設定**］の［**ソケット リスト**］で除外されたソケットのアクションがログに記録されることはありません。除外されたソケットのアクションをログに記録するには，［**除外したソケットをログに含めない**］チェック・ボックスをクリアします。除外されたソケットについてのログ記録を有効にすると，ログ・ファイルでは，アクションは「Exclude」という単語の後に記録されます。

```
Exclude : /* recv(): 15 bytes were received from socket 116 using flags 0 */
```

思考遅延時間のしきい値の設定

VuGen では記録中にオペレータの思考遅延時間が自動的に挿入されます。思考遅延時間のしきい値を設定して，それよりも低い場合には思考遅延時間を無視するようにできます。記録された思考遅延時間がしきい値を越えていると，VuGen によって LRS 関数の前に **lr_think_time** ステートメントが挿入されます。記録された思考遅延時間がしきい値を越えている場合，**lr_think_time** ステートメントは生成されません。

思考遅延時間のしきい値を設定するには，［**思考遅延時間**］セクションの［**思考遅延時間のしきい値**］ボックスに必要な値（秒単位）を入力します。標準設定の値は 5 秒です。

LRS 関数の使用

Windows Sockets プロトコルを使ったクライアントとサーバの間の通信をエミュレートするために開発された関数を，LRS 仮想ユーザ関数と呼びます。LRS 仮想ユーザ関数には，**lrs** という接頭辞が付きます。VuGen では，Windows Sockets セッション中に，この項で説明する LRS 関数のほとんどが自動的に記録されます。また，手作業で任意の関数をプログラミングして，スクリプトに挿入することもできます。LRS 関数の詳細については，「**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。

ソケット関数

lrs_accept_connection	受信側ソケットへの接続を受け入れます。
lrs_close_socket	開いているソケットを閉じます。
lrs_create_socket	ソケットを初期化します。
lrs_disable_socket	ソケットの処理を無効にします。
lrs_exclude_socket	再生中にソケットを除外します。
lrs_get_socket_attrib	ソケットの属性を取得します。
lrs_get_socket_handler	指定したソケットのソケット・ハンドラを取得します。

 **lrs_length_receive**	バッファから指定した長さのデータを受信します。
 **lrs_receive**	ソケットからデータを受信します。
 **lrs_receive_ex**	データグラムまたはストリーム・ソケットから指定した長さのデータを受信します。
 **lrs_send**	データグラムで，またはストリーム・ソケットに，データを送信します。
 **lrs_set_receive_option**	ソケット受信オプションを設定します。
 **lrs_set_socket_handler**	指定したソケットのソケット・ハンドラを設定します。
 **lrs_set_socket_options**	ソケットのオプションを設定します。

バッファ関数

 **lrs_free_buffer**	バッファに割り当てられたメモリを解放します。
 **lrs_get_buffer_by_name**	バッファとそのサイズをデータ・ファイルから取得します。
 **lrs_get_last_received_buffer**	ソケットで最後に受信したバッファとそのサイズを取得します。
 **lrs_get_last_received_buffer_size**	ソケットで最後に受信したバッファのサイズを取得します。
 **lrs_get_received_buffer**	最後に受信したバッファまたはその一部を取得します。
 **lrs_get_static_buffer**	静的バッファまたはその一部を取得します。
 **lrs_get_user_buffer**	ソケットのユーザ・データの内容を取得します。
 **lrs_get_user_buffer_size**	ソケットのユーザ・データのサイズを取得します。
 **lrs_set_send_buffer**	ソケットの送信するバッファを指定します。

環境関数

lrs_cleanup	Windows Sockets DLL の使用を終了します。
lrs_startup	Windows Sockets DLL を初期化します。

相関ステートメント関数

lrs_save_param	静的バッファまたは受信したバッファ（または，その一部）をパラメータに保存します。
lrs_save_param_ex	ユーザ・バッファ，静的バッファ，または受信したバッファ（または，その一部）をパラメータに保存します。
lrs_save_searched_string	静的バッファまたは受信したバッファ内で文字列を検索し，文字列に関連するバッファ領域をパラメータに保存します。

変換関数

lrs_ascii_to_ebcdic	バッファのデータを ASCII 形式から EBCDIC 形式に変換します。
lrs_decimal_to_hex_string	10 進数の整数を 16 進の文字列に変換します。
lrs_ebcdic_to_ascii	バッファのデータを EBCDIC 形式から ASCII 形式に変換します。
lrs_hex_string_to_int	16 進の文字列を整数に変換します。

タイムアウト関数

lrs_set_accept_timeout	ソケットを受け入れるときのタイムアウトを設定します。
lrs_set_connect_timeout	ソケットに接続するときのタイムアウトを設定します。
lrs_set_recv_timeout	予想される最初のデータをソケットで受信するときのタイムアウトを設定します。
lrs_set_recv_timeout2	接続確立後に，予想されるデータをソケットで受信するときのタイムアウトを設定します。
lrs_set_send_timeout	ソケットにデータを送信するときのタイムアウトを設定します。

セッション記録後，記録されたコードを VuGen の組み込みエディタに表示できます。スクリプトをスクロールし，アプリケーションによって生成された関数を表示し，転送されたデータを調べることができます。メイン・ウィンドウにスクリプトを表示すると，VuGen によって記録された動作状況の流れを見ることができます。次は，一般的なセッションで記録される関数の並びを示します。

lrs_startup	Windows Sockets DLL を初期化します。
lrs_create_socket	ソケットを初期化します。
lrs_send	データグラムで，またはストリーム・ソケットへデータを送信します。
lrs_receive	データグラムまたはストリーム・ソケットからデータを受信します。
lrs_disable_socket	ソケットの処理を無効にします。
lrs_close_socket	開いているソケットを閉じます。
lrs_cleanup	WinSock DLL の使用を終了します。

VuGen では，Windows 上で Windows Sockets プロトコルを使用するアプリケーションの記録と再生がサポートされます。UNIX プラットフォームでは再生だけがサポートされます。

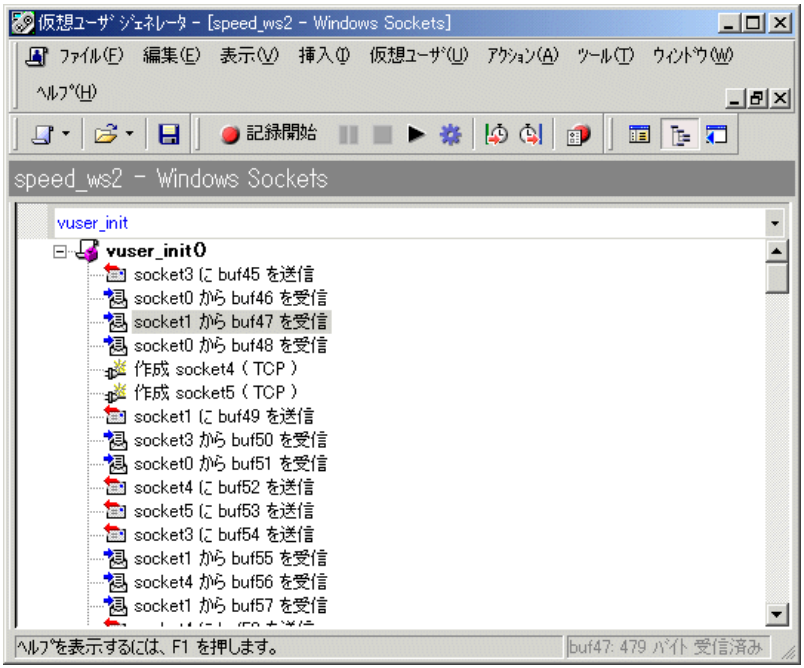
ツリー・ビューとスクリプト・ビューの切り替え

VuGen で Windows Sockets 仮想ユーザ・スクリプトを表示し編集する際，スクリプトをアイコン形式のツリー・ビューで表示するか，テキスト形式のスクリプト・ビューで表示するかを選択できます。

WinSock 仮想ユーザ・スクリプトのツリー・ビューでの表示は，次の手順を行います。

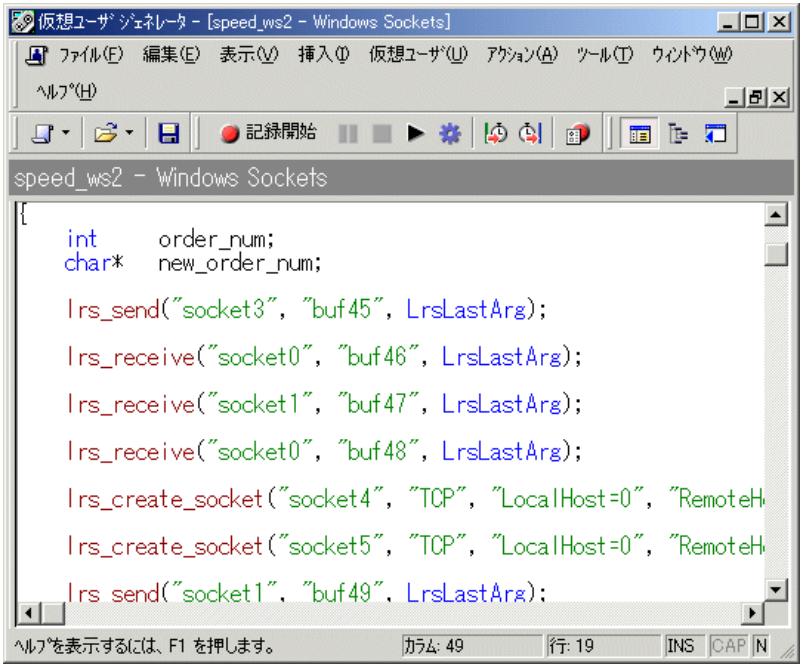
VuGen のメイン・メニューから，［**表示**］＞［**ツリー ビュー**］を選択するか，［**ツリーを表示**］アイコンをクリックします。

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・ビューに表示されます。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。



スクリプト・ビューで表示するには，次の手順を行います。

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。仮想ユーザ・スクリプトがテキスト形式のスクリプト・ビューに表示されます。スクリプト・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。



スクリプト作成後，データをスナップショットまたは未処理のデータ・ファイルとして表示できます。詳細については，第 22 章「Window Sockets データの処理」を参照してください。

第 22 章

Window Sockets データの処理

Windows Sockets プロトコルでセッションを記録後，データを表示および操作できます。

本章では，次の項目について説明します。

- スナップショット・ウィンドウでのデータの表示
- データ内の移動
- バッファ・データの修正
- バッファ名の修正
- データ・ファイルの形式について
- バッファ・データの 16 進形式での表示
- 表示形式の設定
- デバッグに関するヒント
- WinSock スクリプトの手作業による相関

以降の情報は，Windows Sockets レベルで記録されるすべてのプロトコルを対象とします。

Windows Sockets データの処理について

VuGen を使用してアプリケーションを記録すると，データを含む複数のデータ・バッファができます。

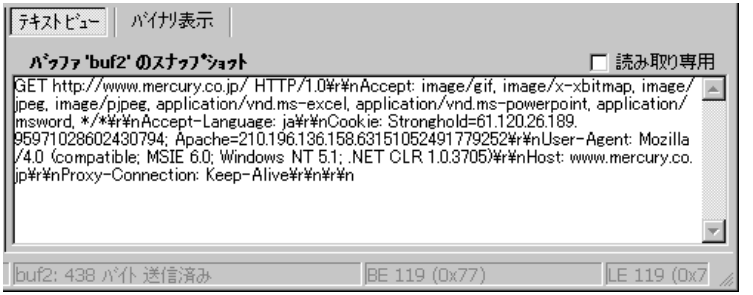
Windows Sockets スクリプトをツリー・ビューで表示すると，VuGen によって，データ・バッファ内の移動とデータの修正が可能なスナップショット・ウィンドウが表示されます。

スクリプト・ビューで作業しているときには，**data.ws** ファイルの未処理のデータを表示できます。詳細については，313 ページ「スクリプト・ビューでの Windows Sockets データの表示」を参照してください。

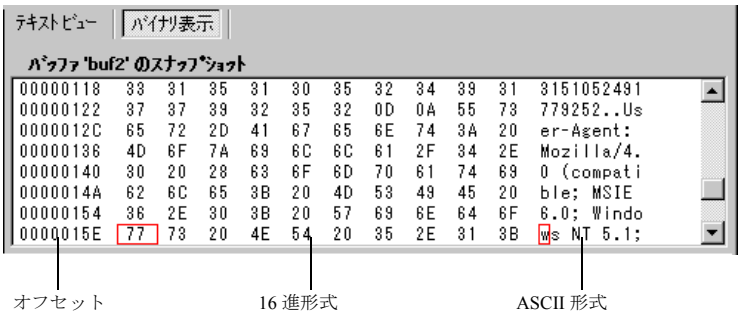
スナップショット・ウィンドウでのデータの表示

ツリー・ビューに Windows Sockets スクリプトを表示すると，編集が可能な［バッファのスナップショット］ウィンドウにデータが表示されます。スナップショットを［**テキスト ビュー**］または［**バイナリ表示**］に表示できます。

［テキスト ビュー］には，バッファのスナップショットの内容がテキストとして表示されます。

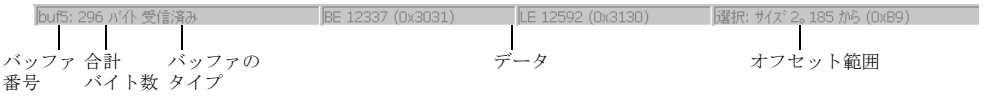


［バイナリ表示］にはデータが 16 進形式で表示されます。左のカラムには，行の最初の文字のオフセットが表示されます。中央のカラムには，データの 16 進値が表示されます。右のカラムには，データが ASCII 形式で表示されます。



バッファのスナップショットの下のステータス・バーには，データとバッファについての情報が提供されます。

- **バッファ番号**：選択されたバッファのバッファ番号。
- **合計バイト数**：バッファの合計バイト数。
- **バッファ・タイプ**：バッファのタイプ（「受信済み」または「送信済み」）。
- **データ**：選択したデータの値をリトル・エンディアン順（バッファ内とは逆）の 10 進および 16 進で表示。
- **オフセット**：バッファの先頭からの選択（［テキスト ビュー］内でのカーソル位置）のオフセット。複数のバイトを選択した場合は，選択範囲が示されます。



ステータス・バーには元のデータが変更されたかどうかも示されます。

データ内の移動

ツリー・ビューには，データ内を移動して，特定の値の識別と分析を行うためのいくつかのツールがあります。

- バッファ・ナビゲータ
- オフセットに移動
- ブックマーク

バッファ・ナビゲータ

標準設定では，VuGen の左の表示枠にすべてのステップおよびバッファが表示されます。［バッファ ナビゲータ］は，送信および受信バッファ・ステップ（**lrs_send**，**lrs_receive**，**lrs_receive_ex**，および **lrs_length_receive**）だけが表示されるフローティング・ウィンドウです。さらに，フィルタを適用して送信バッファまたは受信バッファの一方だけを表示できます。

［バッファ ナビゲータ］でバッファを選択すると，バッファの内容が［バッファのスナップショット］ウィンドウに表示されます。

記録後にバッファの名前を変えると，ステップをクリックしても，バッファの内容は［バッファのスナップショット］ウィンドウに表示されません。名前を変えたバッファのデータを表示するには，［バッファ ナビゲータ］を使って，新しいバッファ名を選択します。VuGen によって，選択したバッファのパラメータ作成が無効になることを示す警告メッセージが表示されます。

［バッファ ナビゲータ］を開くには，［**表示**］＞［**バッファ ナビゲータ**］を選択します。［バッファ ナビゲータ］を閉じるには，［バッファ ナビゲータ］ダイアログ・ボックスの右上角の［×］をクリックします。

左の表示枠のツリー・ビューで，バッファ・ステップをクリックすることによっても，バッファ間を移動できます。［バッファ ナビゲータ］の利点は，それがフィルタ機能のあるフローティング・ウィンドウであることです。

オフセットに移動

オフセットを指定して，データ・バッファ内を移動できます。バッファ内におけるデータの絶対位置またはカーソルの現在位置に対する相対位置を指定できます。また，このダイアログ・ボックスで先頭と末尾のオフセットを指定することによって，ある範囲のデータを選択することもできます。

特定のオフセットへ移動するには，次の手順で行います。

1 ［バッファのスナップショット］ウィンドウ内をクリックします。次に，右クリックして表示されるメニューから［**オフセットに移動**］を選択します。［オフセットに移動］ダイアログ・ボックスが表示されます。

2 バッファ内の特定のオフセット（絶対位置）に移動するには，［**オフセットに移動する**］をクリックして，オフセット値を指定します。

3 カーソルの相対位置へジャンプするには，［**次だけ前に進む**］を選択して，移動するバイト数を指定します。バッファ内を前進する場合は，正の数値を入力します。後退する場合は，負の数値を入力します。

4 バッファ内で，ある範囲のデータを選択するには，［**次の範囲を選択**］を選択して，先頭と末尾のオフセットを指定します。

ブックマーク

バッファ内の場所にブックマークとして印を付けることができます。それぞれのブックマークにわかりやすい名前を付けます。ブックマークをクリックすれば，直接その場所に移動します。ブックマークは，バッファのスナップショットの下にある出力ウィンドウの［**ブックマーク**］タブに表示されます。

ブックマークは，［テキスト ビュー］でも［バイナリ表示］でも使用できます。［テキスト ビュー］で特定のデータの位置を見つけ，その位置をブックマークとして保存し，［バイナリ表示］の中でそのブックマークに直接ジャンプできます。

ブックマークは 1 バイトにも複数バイトにも付けられます。リスト中のブックマークをクリックすると，対応する場所が選択された状態で［バッファのスナップショット］ウィンドウに表示されます。初期設定では，そのデータが［テキスト ビュー］では青で強調表示され，［バイナリ表示］ではブックマーク・ブロックが赤で表示されます。バッファのブックマークにカーソルを置くと，ポップアップ・テキスト・ボックスにブックマークの名前が表示されます。

永久ブックマークと標準ブックマークを作成できます。永久ブックマークは，バッファの［バイナリ表示］内で常に印が付けられています（青いボックスで囲まれています）。バッファ内の異なる位置を指している場合でも，このブックマークは常に選択された状態で，青いボックスの中に表示されています。カーソルの位置は赤でマークされます。一方，標準ブックマークには常に印が付けられているわけではありません。標準ブックマークは，そこにジャンプしたときには赤く印が付きますが，バッファ内でカーソルを移動すると選択されていない状態になります。標準設定のブックマークは永久ブックマークです。

ブックマークの処理は，次の手順で行います。

1 ブックマークを作成するには［バッファのスナップショット］（［テキスト ビュー］または［バイナリ表示］）で 1 つ以上のバイトを選択し，右クリックで表示されるメニューから［**新規ブックマーク**］を選択します。

2 ブックマーク・リストを表示するには，［**表示**］＞［**出力ウィンドウ**］を選択し，［**ブックマーク**］タブを選択します。

3 ブックマークに名前を割り当てるには，ブックマーク・リストのブックマークをクリックして，タイトルを編集します。

4 ブックマークの場所を変更するには，［**ブックマーク**］タブでブックマークを選択してから［バッファのスナップショット］で新しいデータを選択します。［**ブックマーク**］タブの［**変更**］をクリックします。

5 永久ブックマークを標準ブックマークに変更する場合（永久ブックマークは，カーソルを新しい場所に移動しても常にマークされた状態を維持します）は，ブックマークを選択してマウスを右クリックし，［**永久ブックマーク**］の横にあるチェックをクリアします。

6 リストに永久ブックマークだけを表示するには，［**ブックマーク**］タブで［**永久ブックマークのみ表示する**］チェック・ボックスを選択します。

7 特定のバッファのブックマークを表示するには，そのバッファからブックマークを選択し，［**フィルタ**］ボックスの［**選択バッファのみ**］を選択します。

8 ブックマークを削除するには，［**ブックマーク**］タブでブックマークを選択して［**削除**］をクリックします。

バッファ・データの修正

ツリー・ビューには，既存のデータを削除，変更，追加することによってデータを修正するためのいくつかのツールがあります。

- データの挿入
- データの編集
- データのパラメータ化

データの挿入

データ・バッファに数値を挿入できます。数値はシングルバイト，ダブルバイト，または 4 バイト値として挿入できます。

データ・バッファに数値を挿入するには，次の手順で行います。

1 バッファ内をクリックします。

2 右クリックして表示されるメニューから［**詳細設定**］＞［**数値の挿入**］＞［**指定 ...**］を選択します。

3 挿入する ASCII 値を［**値**］ボックスに入力します。

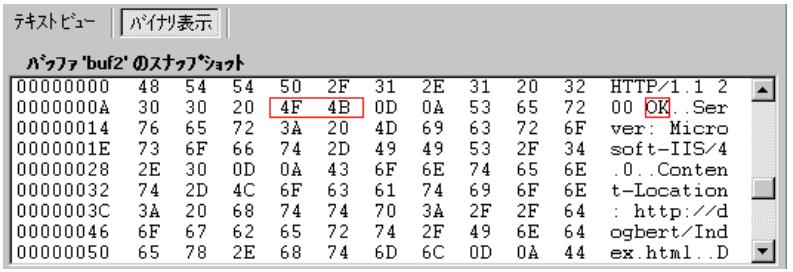
4 挿入するデータのサイズとして，バイト，2 バイト，または 3 バイトを［**サイズ**］ボックスから選択します。

5 ［**OK**］をクリックして完了します。VuGen によってデータの 16 進表現がバッファに挿入されます。

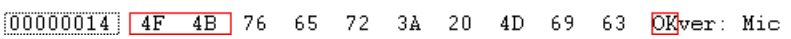
データの編集

バッファ・データに対して，標準的な編集操作（コピー，貼り付け，切り取り，削除，元に戻す）が行えます。［バイナリ表示］では，挿入する実際のデータを指定できます。VuGen では，データの形式（1，2，または 4 バイト，および 16 または 10 進値など）を指定できます。バイナリ・データをコピーし，数値としてバッファに挿入できます。［バイナリ表示］の右カラムには，10 進数または 16 進数を表示できます。

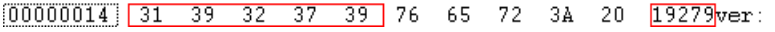
次の例では「OK」という語が選択されています。



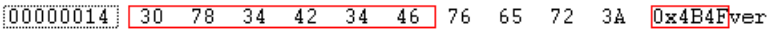
データの次行の先頭で単純なコピー（CTRL+C）と貼り付け（CTRL+V）操作を実行すると，実際のテキストが挿入されます。



［**詳細設定**］＞［**数値としてコピー**］＞［**10 進法**］を選択してからデータを貼り付けると，選択された文字の ASCII コードの 10 進値が挿入されます。



［**詳細設定**］＞［**数値としてコピー**］＞［**16 進法**］を選択してからデータを貼り付けると，選択された文字の ASCII コードの 16 進値が挿入されます。



元戻しバッファに，選択したバッファに対して行われたすべての変更が保持されます。この情報はファイルに保存されるので，ファイルを閉じても利用できます。他の人に変更を取り消されないようにしたい場合は，元戻しバッファを

空にします。元戻しバッファを空にするには，右クリックして表示されるメニューで［**詳細設定**］＞［**元戻しバッファを空にする**］を選択します。

［バイナリ表示］でバッファ・データを編集するには次の操作を行います。

1 バッファ・データのコピーは，次の手順で行います。

- 文字としてコピーする場合は，1 つ以上のバイトを選択し，CTRL+C キーを押します。
- 10 進数としてコピーする場合は，右クリックして表示されるメニューから［**詳細設定**］＞［**数値としてコピー**］＞［**10 進法**］を選択します。
- 16 進数としてコピーする場合は，右クリックして表示されるメニューから［**詳細設定**］＞［**数値としてコピー**］＞［**16 進法**］を選択します。

2 データの貼り付けは，次の手順で行います。

- 単一バイト（クリップボードのデータのサイズを単一バイトと仮定した場合）として貼り付ける場合は，バッファ内の望みの場所をクリックして CTRL+V キーを押します。
- 短い形式（2 バイト）として貼り付ける場合は，右クリックして表示されるメニューから［**詳細設定**］＞［**数字の挿入**］＞［**短形式で貼り付け（2 バイト）**］を選択します。
- 長い形式（4 バイト）として貼り付ける場合は，右クリックして表示されるメニューから［**詳細設定**］＞［**数字の挿入**］＞［**長形式で貼り付け（4 バイト）**］を選択します。

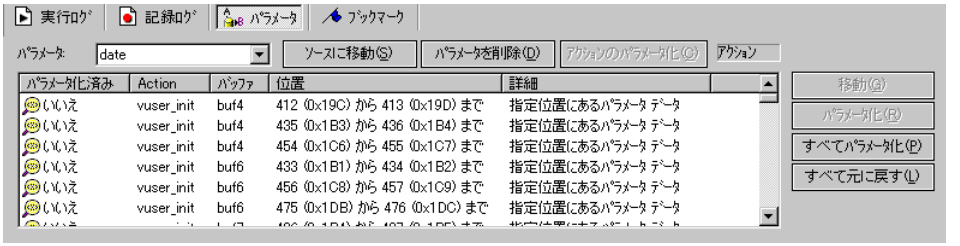
3 データを削除するには，［テキスト ビュー］または［バイナリ表示］で削除するデータを選び，右クリックして表示されるメニューから［**削除**］を選択します。

データのパラメータ化

ツリー・ビューの［バッファのスナップショット］ビューでデータを直接パラメータ化できます。パラメータ化する範囲を指定して境界を指定できます。パラメータ化文字列の境界を指定しなければ，VuGen によってスクリプトに **lrs_save_param** 関数が挿入されます。境界を指定すると，境界引数を指定できる **lrs_save_searched_string** 関数がスクリプトに挿入されます。**lrs_save_param** 関数と **lrs_save_searched_string** 関数によってデータの相関が行われます。つまり，受信したデータが格納され，そのテスト内の以降の任意の場所で使用されます。相関では受信データが格納されるので，受信バッファにだけ適用され，送信バッファには適用されません。お勧めする手順は，受信バッファ内でパラメータ化する動的データの文字列を選択することです。同じパラメータを以降の送信バッファで使用します。

このタイプの相関を，単純なパラメータ化と混同しないでください。単純なパラメータ化（[**挿入**] ＞ [**新規パラメータ**]）は送信バッファ内のデータにだけ適用されます。パラメータを設定し，それに複数の値を割り当てます。VuGen によって，テストの実行ごと，または反復ごとに異なる値が割り当てられたパラメータが使用されます。詳細については，第 7 章 「パラメータの定義」を参照してください。次の各項では，受信バッファのデータの相関についてだけ説明します。

パラメータ作成後，VuGen によって文字列をパラメータに置き換えたすべての場所が表示されます。パラメータ作成についての情報，つまりパラメータが作成されたバッファやバッファ内のオフセットなどの情報も表示されます。[バッファのスナップショット] ビューの下の出力ウィンドウの [**パラメータ**] タブに，パラメータのすべての出現箇所のリストが表示されます。



VuGen でパラメータを操作できます。

絞り込み ：パラメータ名を指定してパラメータ置換の絞込み表示ができます。

ソースへの移動：置換を選択して［ソースに移動］をクリックすると，バッファ内の置換されたパラメータの場所に移動します。

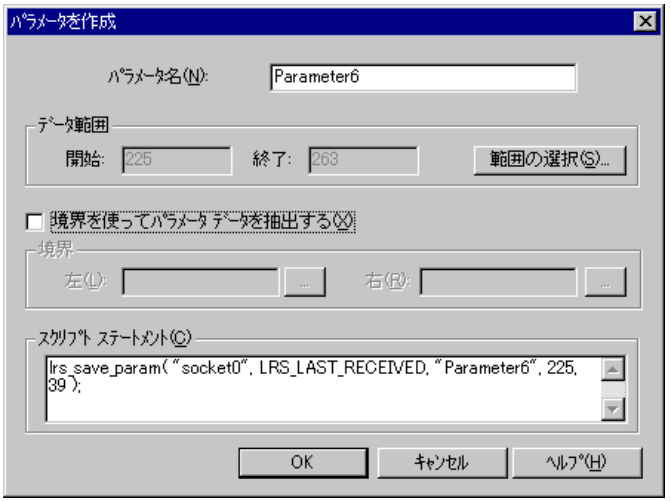
パラメータの削除：任意のパラメータを削除できます。パラメータを削除すると，データが元の値に置き換わり，スクリプトからパラメータ化関数が削除されます。

命名：各置換に名前を付けることができます。

置換を元に戻す：リストに表示された 1 つまたは複数の置換を元に戻すことができます。

［バッファのスナップショット］ウィンドウでのデータのパラメータ化は，次の手順で行います。

1 パラメータ化するデータを選択して，右クリックすると表示されるメニューから［**パラメータの作成**］を選択します（受信バッファにだけ有効）。［**パラメータを作成**］ダイアログ・ボックスが開きます。



2 ［**パラメータ名**］ボックスにパラメータの名前を指定します。

3 パラメータ化するデータ範囲を選択します。標準設定では，バッファで選択されたデータ範囲が使用されます。ダイアログ・ボックスに表示された範囲とは異なる範囲を選択するには，［**範囲の選択**］をクリックします。選択されている範囲を示す小さなダイアログ・ボックスが表示されます。

［バッファのスナップショット］ウィンドウで範囲を選択して［**完了**］をクリックします。

4 パラメータ・データが定数ではないけれども，その境界が一定の場合，左右の境界を指定できます。

5 境界の指定は，次の手順で行います。

［**境界を使ってパラメータ データを抽出する**］チェック・ボックスを選択します。［**スクリプト・ビューのステートメント**］表示枠の関数が **lrs_save_param** から **lrs_save_searched_string** に変更されます。［**Done**］をクリックします。

［**境界**］セクションの［**左**］ボックスの隣にある参照ボタンをクリックします。バッファ内の選択を示す小さいダイアログ・ボックスが表示されます。バッファ内の境界を選択して［**完了**］をクリックします。右の境界についてもこの手順を繰り返します。

6 ［**スクリプト ステートメント**］セクションの引数に必要な修正を行います。たとえば，**lrs_save_param** 関数に **_ex** を追加してエンコード・タイプを指定できます。これらの関数の詳細については，「**オンライン関数リファレンス**」を参照してください。

7 ［**OK**］をクリックしてパラメータを作成します。パラメータの置換前には確認を求められます。［**はい**］をクリックします。［**パラメータ**］タブにすべての置換を表示できます。

8 バッファ内のパラメータの元の場所に移動するには，パラメータを選択して［**ソースに移動**］をクリックします。

9 選択された置換のバッファの場所に移動するには，パラメータを選択して［**移動**］をクリックします。

10 パラメータ全体を削除するには，［**フィルタ**］ボックスのパラメータを選択して［**パラメータを削除**］をクリックします。

11 置換を取り消すには，［**パラメータ**］タブでパラメータを選択して［**元に戻す**］をクリックします。表示されているパラメータの置換をすべて取り消すには，［**パラメータ**］タブでパラメータを選択して［**すべて元に戻す**］をクリックします。

12 特定の置換を取り消すと，ラベルが **Replaced** から **Found** に変更されます。これらの取り消した置換に対して再度パラメータ化ルールを適用するには，［**置換**］または［**すべて置換**］をクリックします。

13 パラメータ全体を削除してすべての置換を元に戻すには，［**フィルタ**］ボックスのパラメータを選択して［**パラメータを削除**］をクリックします。

14 ［**仮想ユーザ**］＞［**パラメータ リスト**］を選択してデータをパラメータに割り当てます。

バッファ名の修正

バッファ名を修正するには，**data.ws** ファイルのスクリプト・ビューを使います。記録後にバッファ名を修正すると，仮想ユーザ・スクリプトの再生に影響が及びます。バッファ・ナビゲータを使うと，名前を変更したバッファの内容をスクリプト・ビューまたはツリー・ビューに表示できます。

バッファで作成したブックマークが使えなくなっている場合は，定義されたバッファ内のブックマークを削除するよう求められます。

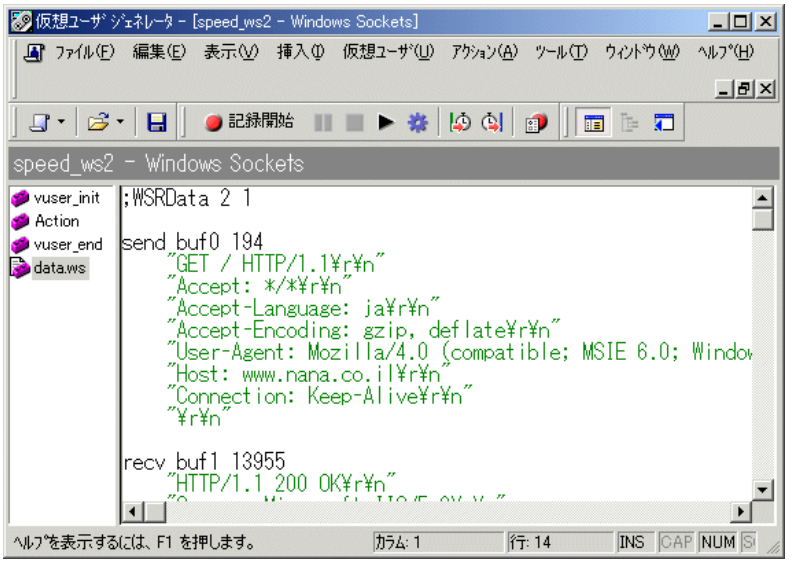
バッファで作成したパラメータが使えなくなっている場合，VuGen によって，パラメータが定義されたバッファ内のパラメータを削除するよう求められます。パラメータを削除すると，他のバッファも含め，すべての置換が元に戻ります。

名前を変更したバッファをバッファ・ナビゲータに表示すると，パラメータの作成がそのバッファ内で無効になることが VuGen によって警告されます。

スクリプト・ビューでの Windows Sockets データの表示

VuGen を使用して Windows Sockets 仮想ユーザ・スクリプトを作成する場合，アクションはスクリプトの 3 つのセクション **vuser_init**，**Actions**，**vuser_end** に記録されます。仮想ユーザ・スクリプトに加えて，記録セッション中に転送または受け取ったデータを含む **data.ws** というデータ・ファイルも作成されます。VuGen のメイン・ウィンドウの［データ ファイル］ボックスで「**data.ws**」を選択すれば，データ・ファイルの内容を表示できます。

データ・ファイルを表示するオプションは，Windows Sockets スクリプトでは標準で使用できます。データはスクリプト・ビューにのみ表示できます。



lrs_receive や **lrs_send** などのいくつかの LRS 関数は，サーバとクライアントの間で送信される実際のデータを処理します。受け取った，または転送されたデータはデータ・バッファに保管されますが，データ・バッファは非常に大きくなることがあります。仮想ユーザ・スクリプトの可読性を高めるために，実際のデータは C ファイルではなく外部ファイルに保管されます。データ転送が発生すると，データは外部ファイルから一次バッファにコピーされます。

外部ファイルである **data.ws** には，すべての一時バッファの内容が含まれています。バッファの内容は連続したレコードとして保管されます。レコードはデータが送信されたか受信されたかを示す識別子とバッファ記述子によってマークされます。LRS 関数では，バッファ記述子を使ってデータにアクセスします。

記述子の形式は次のいずれかです。

recv buf **＜インデックス＞＜受信したバイト数＞**
send buf **＜インデックス＞**

バッファ・インデックスは 0（ゼロ）で始まり，以降のバッファは送受信に関係なくすべて順番に（1，2，3 など）番号が付けられます。

次に示す例では，**lrs_receive** 関数が仮想ユーザ・セッション中に記録されています。

```
lrs_receive("socket1", "buf4", LrsLastArg)
```

この例では，**lrs_receive** によって **socket1** で受信したデータが処理されています。データは 5 番目の受信記録（buf4）に保管されています。バッファ・インデックスは 0（ゼロ）から始まります。**data.ws** ファイルの対応するセクションは，バッファとその内容を示します。

```
recv buf4 39
        "¥xff¥xfb¥x01¥xff¥xfb¥x03¥xff¥xfd¥x01"
        "¥r¥n"
        "¥r¥n"
        "SunOS UNIX (sunny)¥r¥n"
        "¥r"
        "¥x0"
        "¥r¥n"
        "¥r"
        "¥x0"
```

データ・ファイルの形式について

データ・ファイル **data.ws** の形式は以下のとおりです。

- ファイル・ヘッダー
- バッファとその内容のリスト

ファイル・ヘッダーにはデータ・ファイル形式の内部バージョン番号が含まれます。現在のバージョンは 2 です。バージョン 1 の形式のデータ・ファイルからデータにアクセスしようとすると，エラーが発生します。

```
;WSRData 2 1
```

各レコードの前には，データの受信と送信を区別する識別子が付き，バッファ・インデックスと受信したバイト数（**lrs_receive** の場合のみ）が続きます。バッファ・インデックスはバッファを識別する番号です。次に例を示します。

```
recv buf5   25
```

これは，バッファに受信したデータが含まれていることを表します。バッファ・インデックスが「5」なので，この受信操作は 6 番目のデータ転送（バッファ・インデックスは 0 から始まります）で，受信したデータは 25 バイトです。

データが ASCII 形式の場合，ソケットによって転送された実際の ASCII データが記述子の後に続きます。

データが EBCDIC 形式の場合，変換テーブルを通じて変換する必要があります。変換テーブルの設定については，292 ページ「WinSock 記録オプションの設定」を参照してください。EBCDIC データでも，変換後の ASCII 値が印字文字の場合は，ASCII 文字で表示されます。ASCII 値が非印字文字と対応している場合，VuGen によって元の EBCDIC 値が表示されます。

```
recv buf6 39
	"¥xff¥xfb¥x01¥xff¥xfb¥x03¥xff¥xfd¥x01"
	"¥r¥n"
	"SunOS UNIX (sunny)¥r¥n"
```

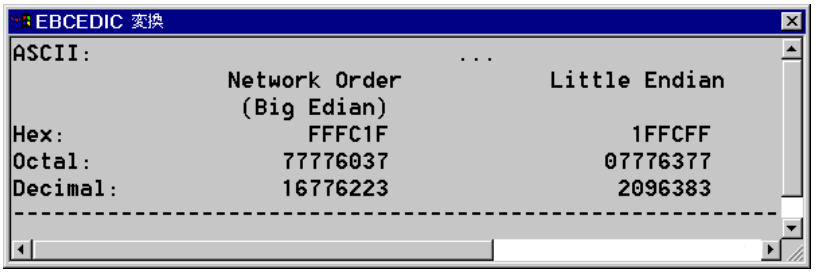
次の例は通常のデータ・ファイルのヘッダー，記述子およびデータを示します。

```
;WSRData 2 1

send buf0
	"¥xff¥xfd¥x01¥xff¥xfd¥x03¥xff¥xfb¥x03¥xff¥xfb¥x18"

recv buf1 15
	"¥xff¥xfd¥x18¥xff¥xfd¥x1f¥xff¥xfd"
	"#"
	"¥xff¥xfd"
	"'"
	"¥xff¥xfd"
	"$"

send buf2
	"¥xff¥xfb¥x18"
```

バッファ・データの 16 進形式での表示

VuGen には，データのオフセットのほかに，データのセグメントを 16 進形式と ASCII 形式で表示できるユーティリティがあります。

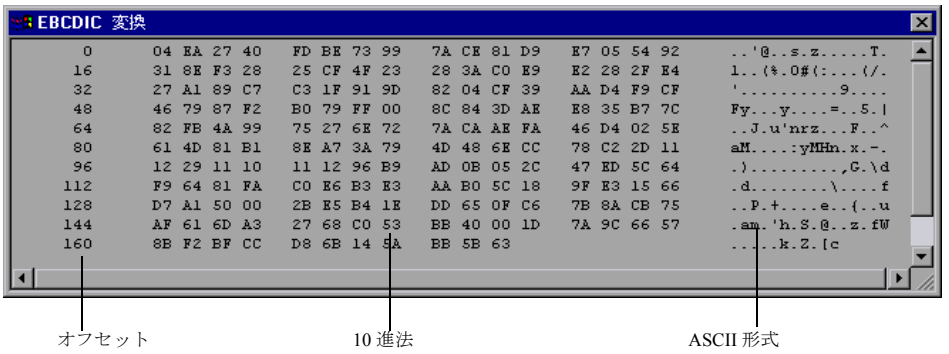
ビューア・ウィンドウでデータを表示するには，データを選択して F7 キーを押します。選択されたテキストが 4 文字以下の場合，VuGen によってそのデータが「**短い形式**」の 16 進，10 進，および 8 進で表示されます。



conv_frm.dat ファイルでは短い形式をカスタマイズできます。詳細については 319 ページ「表示形式の設定」を参照してください。

選択されたテキストが 4 文字を超える場合，VuGen では複数のカラムにデータが「**長い形式**」で表示されます。**conv_frm.dat** ファイルを変更すれば，長い形式をカスタマイズできます。詳細については，319 ページ「表示形式の設定」を参照してください。

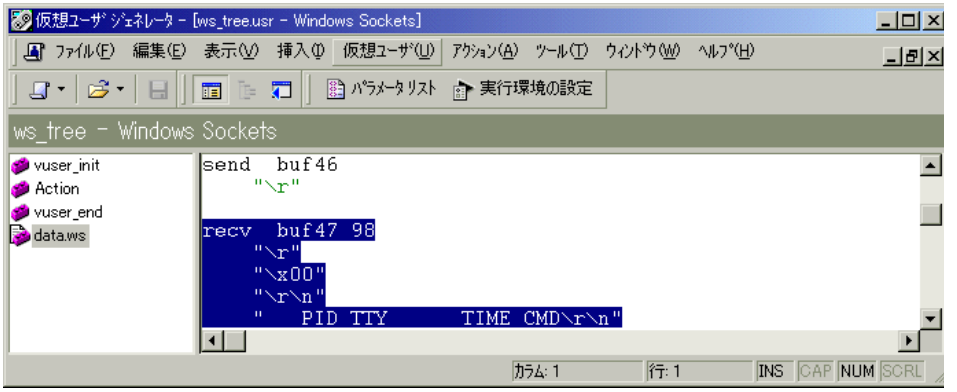
標準設定では，最初のカラムに，マークされた領域の先頭からの文字オフセットが表示されます。2 番目のカラムには，データが 16 進形式で表示されます。3 番目のカラムには，ASCII 形式でデータが表示されます。EBCDIC データを表示するときは，ASCII 形式の非印字文字（「/n」など）はすべてドットで表されます。



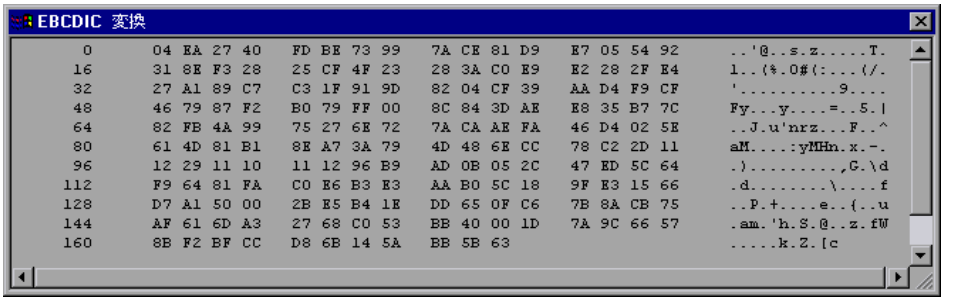
F7 キーを押すと表示されるビューア・ユーティリティは，特にパラメータ化を行う際に役立ちます。このユーティリティを使うことで，パラメータに保存するデータのオフセットを決定できます。

特定の文字のオフセットを決定するには，次の手順で行います。

1 **data.ws** を表示して，バッファの最初からデータを選択します。



2 F7 キーを押してデータと文字のオフセットを表示します。4 文字以上選択すると，データが長い形式で表示されます。



3 ASCII データの中で相関させる値を検索します。この例では，31 番目の文字で始まり，最後の文字が 2 行目にある 13546 番（UNIX セッション中のプロセス ID）を相関させます。

4 **lrs_save_param_ex** 関数のオフセット値を使って，プロセス ID の値を相関させます。詳細については，第 8 章 「ステートメントの相関」を参照してください。

表示形式の設定

VuGen のビューア・ウィンドウ（F7 キー）でのバッファ・データの表示方法を指定できます。**＜ LoarRunner ディレクトリ＞ ¥dat** ディレクトリの **conv_frm.dat** ファイルには，次の表示パラメータが含まれています。

LongBufferFormat：5 文字以上を表示するときに使われる形式です。オフセットには nn，16 進データには XX，ASCII データには aa を使います。

LongBufferHeader：長いバッファ形式で表示する場合に，各バッファの先頭に表示するヘッダーです。

LongBufferFooter：長いバッファ形式で表示する場合に，各バッファの末尾に表示するフッタです。

ShortBufferFormat：4 文字以下を表示するときに使われる形式です。標準的なエスケープ・シーケンスと変換文字を使用できます。

サポートされているエスケープ・シーケンス文字は次のとおりです。

¥a	ベル（アラート）
¥b	バックスペース
¥f	改ページ
¥n	改行
¥r	復帰
¥t	水平タブ
¥v	垂直タブ
¥'	引用符
¥"	二重引用符
¥¥	円記号
¥?	疑問符
¥ooo	ASCII 文字（8 進数）

サポートされている変換文字は次のとおりです。

%a	ASCII 表記
%BX	ビッグ・エンディアン（ネットワーク順序）16 進数
%BO	ビッグ・エンディアン（ネットワーク順序）8 進数
%BD	ビッグ・エンディアン（ネットワーク順序）10 進数
%LX	リトル・エンディアン 16 進数
%LO	リトル・エンディアン 8 進数
%LD	リトル・エンディアン 10 進数

AnyBufferHeader：各バッファの先頭に表示するヘッダー。

AnyBufferFooter：各バッファの末尾に表示するフッタ。

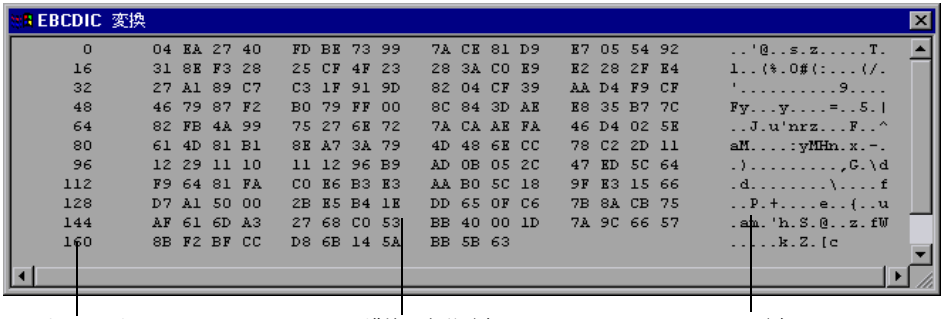
NonPrintableChar ：非印字 ASCII 文字を表す文字。

PrintAllAscii：非印字 ASCII 文字を強制的に出力するには，1 に設定します。

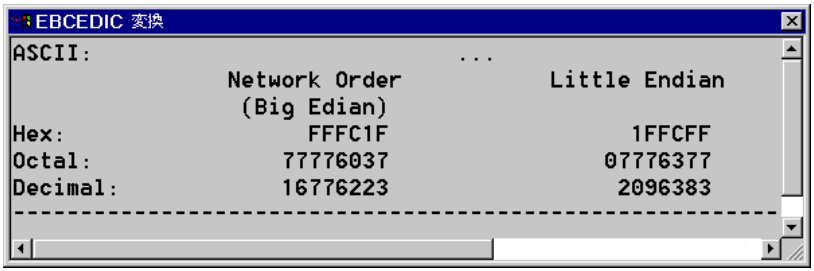
標準設定では，長い形式と短い形式が設定され，非印字文字はドットとして出力するように指定されています。

```
[BufferFormats]
LongBufferFormat=nnnnnnnn     XX XX XX XX   XX XX XX XX   XX XX XX
XX   XX XX XX XX     aaaaaaaaaaaaaaaa¥r¥n
LongBufferHeader=
LongBufferFooter=
ShortBufferFormat=ASCII:¥t¥t¥t%a¥r¥n¥t¥tNetwork Order¥t¥tLittle
Endian¥r¥n¥t¥t (Big
Edian)¥r¥nHex:¥t¥t%BX¥t¥t%LX¥r¥nOctal:¥t¥t%BO¥t¥t%LO¥r¥nDecimal:¥
t%BD¥t¥t%LD¥r¥n
AnyBufferHeader=
AnyBufferFooter=----------------------------------------------------------------------
¥r¥n
NonPrintableChar=.
PrintAllAscii=0
```

標準設定の LongBufferFormat は次のように表示されます。



標準の ShortBufferFormat は次のように表示されます。



デバッグに関するヒント

VuGen では，いくつかの方法でスクリプトのデバッグを行えます。スクリプト実行に関する詳細メッセージを示すさまざまな出力ログとウィンドウを表示できます。

特に Windows Sockets 仮想ユーザ・スクリプトについては，「**バッファの不一致**」に関する追加情報が提供されます。バッファの不一致とは，受け取ったバッファ（再生中に生成されたもの）のサイズと期待バッファ（記録中に生成されたもの）の差を示します。ただし，受け取ったバッファと期待バッファが同じサイズの場合は，内容が異なっていても不一致のメッセージは発行されません。この情報は，システムや仮想ユーザ・スクリプトでの問題を見つけるのに役立ちます。

バッファの不一致情報は実行ログに記録できます。実行ログが表示されていないときは，［**表示**］＞［**出力ウィンドウ**］を選択して，実行ログを表示します。

バッファの不一致は必ずしも問題を示しているわけではありません。たとえば，バッファに以前のログイン時刻などの重要でないデータが含まれている場合，このような不一致は無視できます。

```
Mismatch (expected 54 bytes, 58 bytes actually received)
The expected buffer is:
=================
¥r¥n Last login: Wed Sep 2 10:30:18 from acme.mercury.c¥r¥n
=================
The received buffer is:
=================
¥r¥n Last login: Thu Sep 10 11:19:50 from dolphin.mercury.c¥r¥n
```

ただし，期待バッファと受信バッファの間でサイズが著しく異なる場合は，システムに問題があることを示します。対応するバッファでデータの不一致を確認してください。

重要な不一致かどうかを判断できるように，アプリケーションを完全に理解しておく必要があります。

WinSock スクリプトの手作業による相関

VuGen には，仮想ユーザ・スクリプトを相関させるためのユーザ・インタフェースがあります。相関は，動的なデータを利用する場合に必要です。WinSock 仮想ユーザ・スクリプトにおいてよくある問題の 1 つに，ポート番号が動的に割り当てられる「動的ポート」があります。特定のアプリケーションが常に同じポートを使用していると，他のアプリケーションは次の使用可能なポートを使用します。スクリプトを再生しようとしたときに，記録されたポートが使用できない場合，テストは失敗します。この問題に対処するには，相関を行う必要があります。実際の実行時の値を保存し，それらの値をスクリプト内で使用します。

動的な値をパラメータに保存する相関関数を使用して，仮想ユーザ・スクリプトを手作業で相関させることができます。**lrs_save_param** と **lrs_save_param_ex** 関数では，受信バッファ内のデータのオフセットに基づいて，データをパラメータに保存できます。高度な相関関数 **lrs_save_searched_string** を使えば，データの境界と，一致パターンの何回目の出現をパラメータに保存するかを指定できます。以下の例では，**lrs_save_param_ex** を使った相関について説明します。他の相関関数の使用法については，「**オンライン関数リファレンス**」を参照してください。

WinSock 仮想ユーザ・ステートメントを相関させるには，次の手順で行います。

1 スクリプト内の，バッファの内容を保存する場所に **lrs_save_param_ex** ステートメントを挿入します。ユーザ・バッファ，静的バッファ，または受信バッファを保存できます。

```
lrs_save_param_ex (socket, type, buffer, offset, length, encoding, parameter);
```

2 パラメータを参照します。

VuGen のメイン・ウィンドウの［データ ファイル］ボックスで **data.*ws*** ファイルを選択して，バッファの内容を表示します。保存したバッファの内容で置換するデータを探します。置換する値のすべてのインスタンスを山括弧（＜＞）で囲まれたパラメータ名で置き換えます。

次の例では，ユーザが **telnet** セッションを実行し，**ps** コマンドを使ってプロセス ID（PID）を調べ，その PID を使ってアプリケーションを強制終了しています。

```
frodo:/u/jay>ps
  PID TTY      TIME CMD
14602 pts/18    0:00 clock
14569 pts/18    0:03 tcsh

frodo:/u/jay>kill 14602
[3] Exit 1                clock
frodo:/u/jay>
```

テストの実行時の PID は異なる（UNIX は各実行に固有の PID を割り当てます）ので，記録された PID を強制終了しても意味がありません。この問題に対処するには，**lrs_save_param_ex** を使って，現在の PID をパラメータに保存します。そして定数をパラメータで置き換えます。

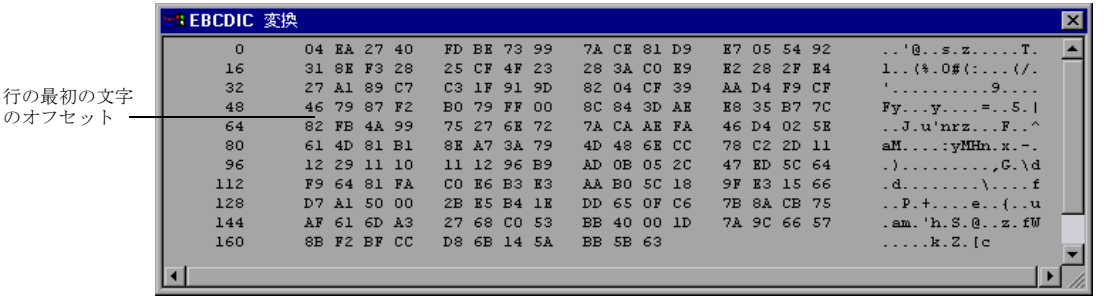
3 **data.ws** ファイル内で，データを受け取ったバッファを調べます（この例では buf47）。

```
recv  buf47 98
      "¥r"
      "¥x00"
      "¥r¥n"
      "   PID TTY      TIME CMD¥r¥n"
      " 14602 pts/18   0:00 clock¥r¥n"
      " 14569 pts/18   0:02 tcsh¥r¥n"
      "frodo:/u/jay>"
.
.
.
send buf58
      "kill 14602"
```

4 **Actions** セクションで，buf47 によって使用されるソケットを調べます。この例では **socket1** です。

```
lrs_receive("socket1", "buf47", LrsLastArg);
```

5 保存するデータ文字列のオフセットと長さを調べます。バッファ全体を強調表示して F7 キーを押します。**PID** のオフセットは 11 で，長さは 5 バイトです。これらのデータの表示方法の詳細については，315 ページ「データ・ファイルの形式について」を参照してください。



6 Actions セクションで，当該バッファの **lrs_receive** 関数の後に **lrs_save_param_ex** 関数を挿入します。この例では，バッファは **buf47** です。PID は **param1** という名前のパラメータに保存されます。**lr_output_message** を使って，出力にパラメータを送信します。

```
lrs_receive("socket1", "buf79", LrsLastArg);
lrs_save_param("socket1", "user" buf47, 11, 5, ascii, param1);
lr_output_message ("param1:%s", lr_eval_string(" < param1 > "));
lr_think_time(10);
lrs_send("socket1", "buf80", LrsLastArg);
```

7 **data.ws** ファイル内で，パラメータで置換するデータを調べます（この例では **PID**）。

```
send buf58
    "kill 14602"
```

8 値を山括弧で囲まれたパラメータで置換します。

```
send buf58
    "kill < param1 > "
```

第 6 部

ユーザ定義の仮想ユーザ・スクリプト

第 23 章

仮想ユーザ・スクリプトの作成

セッションを記録する方法に加えて，ユーザ定義の仮想ユーザ・スクリプトを作成することができます。LoadRunner API 関数と標準の C，Java，VB，VBScript，JavaScript コードの両方を使用できます。

本章では，以下の項目について説明します。

- C 仮想ユーザ
- Java 仮想ユーザ
- VB 仮想ユーザ
- VBScript 仮想ユーザ
- JavaScript 仮想ユーザ

以降の情報は，すべてのユーザ定義の仮想ユーザ・スクリプト（C，JavaScript，Java，VB，VBScript）を対象とします。

ユーザ定義の仮想ユーザ・スクリプト

VuGen では，実際のセッションを記録せずに，独自の関数をスクリプトにプログラミングできます。LoadRunner API または標準のプログラミング関数を使用できます。LoadRunner API 関数では，仮想ユーザについての情報を収集できます。たとえば，仮想ユーザ関数を使用してサーバ・パフォーマンスの測定，サーバ負荷の制御，デバッグ・コードの追加を行ったり，シナリオの仮想ユーザについての実行時情報を取得できます。

本章では，対象アプリケーションのライブラリやクラスと連携して動作する仮想ユーザ・スクリプトを VuGen エディタでプログラミングする方法について説明します。

また，Visual C および Visual Basic 環境からプログラミングを行って仮想ユーザ・スクリプトを開発することもできます。これらの環境では，開発アプリケーションに LoadRunner のライブラリをインポートして仮想ユーザ・スクリプトを作成します。詳細については，第 62 章 「Visual Studio による仮想ユーザ・スクリプトの作成」を参照してください。

ユーザ定義のスクリプトを作成するには，まず最初にスクリプトのスケルトンを作成します。スクリプトのスケルトンには，スクリプトの 3 つの主要セクション，**init**，**actions**，および **end** が含まれています。これらのセクションは最初は空なので，手作業で関数を挿入します。

空のスクリプトは，次のプログラミング言語で作成できます。

- C
- Java
- Visual Basic
- VBScript
- JavaScript

注：JavaScript と VBScript 仮想ユーザを使う場合，スクリプトで使用する COM オブジェクトは完全なオートメーション対応である必要があります。これによって，あるアプリケーションが別のアプリケーションにあるオブジェクトを操作したり，オブジェクトを外部から操作できるように公開できます。

C 仮想ユーザ

C 仮想ユーザ・スクリプトでは，標準 ANSI 規格の C 言語のコードを配置できます。空の C 仮想ユーザ・スクリプトを作成するには，［新規仮想ユーザ］ダイアログ・ボックスの［ユーザ定義］カテゴリから［**C Vuser**］を選択します。VuGen によって，空のスクリプトが作成されます。

```
Action1()
{

    return 0;
}
```

C 言語の関数を使用するタイプの仮想ユーザ・スクリプトであれば，どのスクリプトでも C 仮想ユーザ関数を使用できます。

よく使用される C 言語の関数の構文や使用例については「**オンライン関数リファレンス**」（［**ヘルプ**］＞［**関数リファレンス**］）の C 言語リファレンスを参照することもできます。

C 言語の関数の使用についてのガイドライン

制御フローや構文など，標準 ANSI-C の規約はすべて，C 仮想ユーザ・スクリプトにも適用されます。他の C 言語のプログラムと同じように，スクリプトでもコメント文や条件文が利用できます。ANSI-C の規約に従って変数を宣言して定義できます。

仮想ユーザ・スクリプトの実行で使われている C インタプリタは，標準の ANSI-C を受け付けます。ANSI-C の Microsoft 拡張はサポートしていません。

C 言語の関数を仮想ユーザ・スクリプトに追加する場合，以下の制約があるので注意してください。

- 仮想ユーザ・スクリプトでは関数のアドレスをライブラリ関数へのコールバックとして渡すことはできません。
- **stdargs**，**longjmp**，および **alloca** 関数は仮想ユーザ・スクリプトではサポートされていません。
- 仮想ユーザ・スクリプトには，構造体の引数や戻り値の型はありません。構造体へのポインタはサポートされています。
- 仮想ユーザ・スクリプト内のリテラル文字列は読み取り専用です。リテラル文字列に書き込もうとするとアクセス違反が起こります。

➤ int を返さない C 関数は型変換を行う必要があります。次に例を示します。

```
extern char * strtok();
```

libc 関数の呼び出し

仮想ユーザ・スクリプトで，**libc** 関数を呼び出すことができます。ただし，仮想ユーザ・スクリプトの実行で使われているインタプリタが ANSI C の Microsoft 社による拡張をサポートしていないため，Microsoft 社のインクルード・ファイルを使うことはできません。必要なときには自分自身のプロトタイプを書くか，マーキュリー・インタラクティブのカスタマー・サポートに連絡して，**libc** 関数のプロトタイプを含む ANSI 準拠のインクルード・ファイルを入手してください。

リンク・モード

仮想ユーザ・スクリプトの実行に使用する C インタプリタでは，使用前に関数を定義しておく限りは実行開始時に定義する必要がないようにするために，「遅延」リンク・モードを使用します。たとえば，次のような場合です。

```
lr_load_dll("mydll.dll");
    myfun();  /* mydll.dll で定義済み。myfun.dll がロードされたらすぐに
                 直接呼び出せる。*/
```

Java 仮想ユーザ

Java 仮想ユーザ・スクリプトでは，標準 Java コードが使用できます。空の Java 仮想ユーザ・スクリプトを作成するには，［新規仮想ユーザ］ダイアログ・ボックスの［**ユーザ定義**］カテゴリから［**Java Vuser**］を選択します。VuGen によって，空の Java スクリプトが作成されます。

```
import lrapi.lr;

public class Actions
{

    public int init() {
        return 0;
    }


    public int action() {
        return 0;
    }


    public int end() {
        return 0;
    }
}
```

Java 仮想ユーザ・タイプでは，**Actions** クラスの編集しかできないことに注意してください。Actions クラスの中には，**init**，**action** および **end** の 3 つのメソッドがあります。**init** メソッドに初期化コードを，**action** メソッドにビジネス・プロセスを，**end** メソッドにクリーンアップ・コードを記述します。

Java-CORBA および Java-RMI の仮想ユーザ・スクリプトで Java 仮想ユーザ関数を使うこともできます。

VB 仮想ユーザ

空の Visual Basic 仮想ユーザ・スクリプトを作成して，Visual Basic コードを書くことができます。このスクリプト・タイプでは，Visual Basic アプリケーションを LoadRunner に取り入れることができます。空の VB 仮想ユーザ・スクリプトを作成するには，［新規仮想ユーザ］ダイアログ・ボックスの［**ユーザ定義**］カテゴリから［**VB Vuser**］を選択します。VuGen によって，空の VB スクリプトが作成されます。

```
Public Function Actions() As Long

        '"TO DO: Place your action code here



    Actions = lr.PASS
End Function
```

VuGen によって，**vuser_init**，**action**，および **vuser_end** の 3 つのセクションが作成されます。これらのセクションにはそれぞれ **Init**，**Actions**，および **Terminate** の VB 関数が含まれています。コードは，**TO DO** コメントの指示に従って，これらの関数の中に配置します。

VuGen から表示できる追加のセクションは，LoadRunner と VB アプリケーションのオブジェクトおよび変数グローバル宣言が含まれる **global.vba** ファイルです。

VBScript 仮想ユーザ

空の VBScript 仮想ユーザ・スクリプトを作成して，VBScript コードを配置できます。このスクリプト・タイプでは，VBScript アプリケーションを LoadRunner に取り入れることができます。空の VBScript 仮想ユーザ・スクリプトを作成するには，［新規仮想ユーザ］ダイアログ・ボックスの［**ユーザ定義**］カテゴリから［**VB Script Vuser**］を選択します。VuGen によって，空の VBScript 仮想ユーザ・スクリプトが作成されます。

```
Public Function Actions()

        '"TO DO: Place your action code here



    Actions = lr.PASS
End Function
```

VuGen によって，**vuser_init**，**action**，および **vuser_end** の 3 つのセクションが作成されます。これらのセクションにはそれぞれ **Init**，**Actions**，および **Terminate** の VBScript 関数が含まれています。コードは，**TO DO** コメントの指示に従って，これらの関数の中に配置します。

VuGen から表示できる追加のセクションは，LoadRunner と VB スクリプトのオブジェクトを作成する **global.vbs** ファイルです。たとえば，次のコードは標準の LoadRunner オブジェクトを作成します。

```
Set lr = CreateObject("LoadRunner.LrApr")
```

JavaScript 仮想ユーザ

空の JavaScript 仮想ユーザ・スクリプトを作成して，JavaScript コードを配置できます。このスクリプト・タイプでは，既存の JavaScript アプリケーションを LoadRunner に取り入れることができます。空の JavaScript 仮想ユーザ・スクリプトを作成するには，［新規仮想ユーザ］ダイアログ・ボックスの［**ユーザ定義**］カテゴリから［**Javascript Vuser**］を選択します。

```
function Actions()
{
    //"TO DO: Place your business process/action code here



    return(lr.PASS);
}
```

VuGen によって，**vuser_init**，**action**，および **vuser_end** の 3 つのセクションが作成されます。これらのセクションにはそれぞれ **Init**，**Actions**，および **Terminate** の JavaScript 関数が含まれています。コードは，**TO DO** コメントの指示に従って，これらの関数の中に配置します。

VuGen から表示できる追加のセクションは，LoadRunner と JavaScript のオブジェクトを作成する **global.js** ファイルです。たとえば，次のコードは標準の LoadRunner オブジェクトを作成します。

```
var lr = new ActiveXObject("LoadRunner.LrApr")
```

第 24 章

Java 仮想ユーザ・スクリプトのプログラミング

VuGen では Java タイプのユーザがプロトコル・レベルでサポートされています。本章では，「**プログラミング**」によって Java 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。「**記録**」によって Java 仮想ユーザ・スクリプトを作成する方法については，Corba-Java，RMI-Java，EJB，Jacada タイプのプロトコルに関する章を参照してください。

本章では，**Java** 仮想ユーザを使って，Java で仮想ユーザ・スクリプトのプログラミングを行う方法について説明します。

- Java 仮想ユーザの作成
- Java 仮想ユーザ・スクリプトの編集
- LoadRunner の Java API
- Java 仮想ユーザ関数の使用法
- Java 環境の設定
- Java 仮想ユーザ・スクリプトの実行
- パッケージの一部としてのスクリプトのコンパイルと実行
- プログラミングについてのヒント

以降の情報は，Java，EJB，CORBA-Java，RMI-Java，および Jacada の仮想ユーザ・スクリプトを対象とします。

Java 仮想ユーザ・スクリプトのプログラミングについて

Java コードを使って仮想ユーザ・スクリプトを作成するには，**Java**，**CORBA-Java**，または **RMI-Java** タイプの仮想ユーザを使用します。これらの仮想ユーザ・タイプでは，プロトコル・レベルで Java がサポートされています。仮想ユーザ・スクリプトは Java コンパイラによってコンパイルされ，Java の標準規約をすべてサポートします。たとえば，テキストの前に 2 つのスラッシュ「//」を付けることによって，コメントを挿入できます。

CORBA，RMI，EJB，および Jacada 仮想ユーザに関する章では，記録によってスクリプトを作成する方法を説明しています。プログラミングによって Java コードのスクリプトを作成するには，以下の項を参照してください。

Java 互換の仮想ユーザ・スクリプトを作成する第一歩は，Java 仮想ユーザ・タイプの新しい仮想ユーザ・スクリプトのテンプレートを作成することです。次に，任意の Java コードをスクリプト・テンプレートにプログラミングするか，貼り付けます。LoadRunner Java 仮想ユーザ関数を追加して，スクリプトを拡張したり，反復時にさまざまな値を使用できるように引数をパラメータ化したりできます。

Java 仮想ユーザ・スクリプトは，スケーラブルなマルチ・スレッド・アプリケーションとして動作します。スクリプトにユーザ定義のクラスを含める場合は，コードをスレッドセーフにする必要があります。コードをスレッドセーフにしないと，結果が不正確になることがあります。スレッドセーフでないコードの場合は，Java 仮想ユーザをプロセスとして実行します。これによりプロセスごとに個別の Java 仮想マシンが動作します。その結果，スクリプトのスケーラビリティは低くなります。

スクリプトを作成したら，VuGen でスタンドアロン・テストとして実行します。Java コンパイラ（Sun の javac）によってスクリプトに誤りがないか調べられた後，コンパイルが実行されます。スクリプトが機能することを確認したら，LoadRunner のシナリオに組み込みます。

Java 仮想ユーザの作成

Java 互換仮想ユーザ・スクリプト作成の第一歩は，Java 仮想ユーザ・テンプレートを作成することです。

Java 仮想ユーザ・スクリプトの作成は，次の手順で行います。

1 VuGen を開きます。

2 ［**ファイル**］＞［**新規作成**］を選択するか，［**新規作成**］ボタンをクリックします。［新規仮想ユーザ］ダイアログ・ボックスが開きます。

3 仮想ユーザのタイプを選択するリストから［**ユーザ定義**］＞［**Java Vuser**］を選択し，［**OK**］をクリックします。空の Java 仮想ユーザ・スクリプトが表示されます。

4 左側の表示枠の **Actions** セクションをクリックして，**Actions** クラスを表示します。

Java 仮想ユーザ・スクリプトの編集

空のテンプレートを生成したら，任意の Java コードを挿入できます。このタイプの仮想ユーザ・スクリプトで作業をする場合には，すべてのコードを **Actions** クラスに配置します。Actions クラスを表示するには，左側の表示枠の［**Actions**］をクリックします。その内容が右側の表示枠に表示されます。

```
import lrapi.*;
public class Actions
{
	public int init() {
		return 0;
	}

	public int action() {
		return 0;
	}

	public int end() {
		return 0;
	}
}
```

Actions クラスには，**init**，**action**，**end** の 3 つのメソッドが含まれています。次の表に，各メソッドに何を含める必要があり，各メソッドがどのタイミングで呼び出されるかを示します。

スクリプトのメソッド	エミュレーション内容	実行のタイミング
init	サーバへのログイン	仮想ユーザの初期化時（ロード時）
action	クライアントの動作	仮想ユーザが「実行」状態のとき
end	ログオフ処理	仮想ユーザの終了時または停止時

init メソッド

すべてのログイン手続きと一度だけ行う設定を **init** メソッドに置きます。**init** メソッドは仮想ユーザがスクリプトの実行を開始するときに 1 回だけ実行されます。次のサンプル **init** メソッドではアプレットを初期化しています。

```
import org.omg.CORBA.*;
import org.omg.CORBA.ORB.*;
import lrapi.lr;

// Public 関数 : init
    public int init() throws Throwable {

        // Orb インスタンスを初期化する ..
        MApplet mapplet = new MApplet("http://chaos/classes/", null);
        orb = org.omg.CORBA.ORB.init(mapplet, null);
...
```

action メソッド

仮想ユーザで行うすべての操作を **action** メソッドに配置します。**action** メソッドは，実行環境の設定で指定した反復回数に従って実行されます。反復回数の設定の詳細については，第 9 章 「実行環境の設定」を参照してください。次のサンプル **action** メソッドでは仮想ユーザ ID を取り出して，出力しています。

```
public int action() {
    lr.message("vuser: " + lr.get_vuser_id() + " xxx");
        return 0;
 }
```

end メソッド

end メソッドには，サーバからのログオフ，環境の後始末など，シナリオの終了時に LoadRunner に実行させるコードを配置します。**end** メソッドは仮想ユーザがスクリプトの実行を終了するときに 1 回だけ実行されます。次の例に示す **end** メソッドでは「**End**」というメッセージを実行ログに出力しています。

```
public int end() {
      lr.message("End");
      return 0;
   }
```

LoadRunner の Java API

LoadRunner では仮想ユーザ関数にアクセスするための固有の Java API を提供しています。これらの関数はすべて lrapi.lr クラスの静的メソッドです。この項では，LoadRunner の Java 仮想ユーザ関数の一覧を示します。個々の関数の詳細については，「**オンライン関数リファレンス**」（[**ヘルプ**] > [**関数リファレンス**]）を参照してください。新規 Java 仮想ユーザ・スクリプトを作成するとき，**import lrapi.*** はスクリプトに自動的に挿入されます。

トランザクション関数

lr.declare_transaction	トランザクションを宣言します。
lr.start_transaction	トランザクションの開始を示します。
lr.end_transaction	トランザクションの終了を示します。

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・ラインで指定された文字列を取得します。

情報関数

lr.value_check	パラメータ値を検査します。
lr.user_data_point	ユーザ定義データのサンプルを記録します。
lr.get_group_name	仮想ユーザ・グループの名前を返します。
lr.get_host_name	仮想ユーザ・スクリプトを実行しているロード・ジェネレータの名前を返します。
lr.get_master_host_name	LoadRunner コントローラを実行しているマシンの名前を返します。
lr.get_object	Java オブジェクトをキャプチャし，データ・ファイルにダンプします（Corba-Java のみ）。
lr.get_scenario_id	現在のシナリオの ID を返します。
lr.get_vuser_id	現在の仮想ユーザの ID を返します。

文字列関数

lr.deserialize	ASCII 形式のコンポーネントを表すためにオブジェクトを展開します。
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.next_row	指定したパラメータのデータとして次の行のデータを使用するように指示します。
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。

メッセージ関数

lr.debug_message	デバッグ・メッセージを［出力］ウィンドウに送ります。
lr.enable_redirection	標準のメッセージとエラーを，標準出力と標準エラーとしてログ・ファイルにリダイレクトします。
lr_error_message	仮想ユーザ・ログ・ファイルと［出力］ウィンドウに，場所の詳細情報とともにエラー・メッセージを送ります。
lr.get_debug_message	現在のメッセージ・クラスを取得します。
lr.log_message	メッセージを仮想ユーザ・ログ・ファイルに送ります。
lr.message	メッセージを［出力］ウィンドウに送ります。
lr.output_message	ログ・ファイルと［出力］ウィンドウに，メッセージと場所の情報を送ります。
lr.redirect	文字列をファイルにリダイレクトします。
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	メッセージを［コントローラ］ウィンドウの仮想ユーザ・ステータス領域に送信します。

ランタイム関数

lr.declare_rendezvous	仮想ユーザ・スクリプトにランデブーを宣言します。
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.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。
lr.think_time	スクリプトの実行を一時停止して，思考遅延時間（実際のユーザが次の操作に移る前に考える時間）をエミュレートします。

Java クラスを追加して使うには，以下の例に示すようにそれらをスクリプトの先頭でインポートします。インポートするクラスのディレクトリや関連 jar ファイルをクラスパスに忘れずに追加します。また，追加するクラスがスレッドセーフかつスケーラブルであることを確認します。

```
import java.io.*;
import lrapi.*;

public class Actions
{
...
}
```

Java 仮想ユーザ関数の使用法

Java 仮想ユーザ関数を使って，以下の作業を行うことによりスクリプトを拡張できます。

- トランザクションの挿入
- ランデブー・ポイントの挿入
- 仮想ユーザ情報の取得
- 出力メッセージの発行
- ユーザの思考時間のエミュレート
- コマンド・ライン引数の処理

トランザクションの挿入

サーバのパフォーマンスを測定するには，トランザクションを挿入します。各トランザクションは，仮想ユーザからの特定の要求に対するサーバの応答時間を測定します。これらの要求は，単純な場合や複雑な場合があります。シナリオの実行中および実行後に，LoadRunner のオンライン・モニタとグラフを使って，トランザクションごとのパフォーマンスを分析できます。

またトランザクションのステータスとして，lr.PASS および lr.FAIL を指定することもできます。トランザクションが正常終了したかどうか LoadRunner に判定させることができます。また，条件ループを使って自分で判定することもできます。たとえば，コードの中で，リターン・コードが特定の値になっているかどうかを調べることができます。リターン・コードが正しい値であれば，lr.PASS ステータスを発行します。リターン・コードの値が正しくなければ，lr.FAIL ステータスを発行します。

トランザクションの開始と終了を示すには，次の手順で行います。

1 **lr.start_transaction** 関数は，スクリプトの中で処理の実行時間の計測を開始する場所に挿入します。

2 **lr.end_transaction** 関数は，スクリプトの中で処理の実行時間の計測を終了する場所に挿入します。**lr.start_transaction** 関数で指定したトランザクション名を指定します。

3 トランザクションのステータス（lr.PASS または lr.FAIL）を指定します。

```
public int action() {

for(int i=0;i<10;i++)
  {
     lr.message("action()"+i);
     lr.start_transaction("trans1");
     lr.think_time(2);
     lr.end_transaction("trans1",lr.PASS);
  }
     return 0;
}
```

ランデブー・ポイントの挿入

クライアント / サーバ・システムを対象に多数のユーザによる負荷をエミュレートするには，「**ランデブー・ポイント**」を作成して，多数の仮想ユーザが同時にタスクを実行するように同期させなければなりません。仮想ユーザがランデブー・ポイントに到達すると，コントローラは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，その仮想ユーザを待機させます。

仮想ユーザ・スクリプトにランデブー関数を挿入することによって，この「待ち合わせ場所」を指定します。

ランデブー・ポイントの挿入は，次の手順で行います。

1 仮想ユーザを待機させる場所に **lr.rendezvous** 関数を挿入します。

```
public int action() {

  for(int i=0;i<10;i++)
  {
     lr.rendezvous("rendz1");
     lr.message("action()"+i);
     lr.think_time(2);
  }
  return 0;
}
```

仮想ユーザ情報の取得

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	仮想ユーザ ID やロード・ジェネレータの名前などのコマンド・ライン引数値または実行時情報を含む文字列を返します。
lr.get_group_name	仮想ユーザ・グループの名前を返します。
lr.get_host_name	仮想ユーザ・スクリプトを実行しているロード・ジェネレータの名前を返します。
lr.get_master_host_name	LoadRunner コントローラを実行しているマシンの名前を返します。
lr.get_scenario_id	現在のシナリオの ID を返します。
lr.get_vuser_id	現在の仮想ユーザの ID を返します。

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

```
String my_host = lr.get_host_name( );
```

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。

出力メッセージの発行

シナリオを実行しているとき，コントローラの［出力］ウィンドウに，スクリプトの実行に関するメッセージが表示されます。エラーおよび通知メッセージをコントローラに送るためのステートメントを仮想ユーザ・スクリプトに挿入できます。これらのメッセージは，コントローラによって［出力］ウィンドウに表示されます。たとえば，クライアント・アプリケーションの現在のステータスを示すメッセージを挿入することができます。また，これらのメッセージをファイルに保存することもできます。

注：トランザクション内からメッセージを送ってはなりません。トランザクション内からメッセージを送ると，トランザクションの実行時間が長くなり，実際のトランザクションの結果に偏りが生じる可能性があります。

仮想ユーザ・スクリプトの中で，以下のメッセージ関数が使用できます。

lr.debug_message	デバッグ・メッセージを［出力］ウィンドウに送ります。
lr.log_message	メッセージを仮想ユーザ・ログ・ファイルに送ります。
lr.message	メッセージを［出力］ウィンドウに送ります。
lr.output_message	ログ・ファイルと［出力］ウィンドウに，メッセージと場所の情報を送ります。

次の例では，**lr.message** を使用してループ回数を示すメッセージを［出力］ウィンドウに送っています。

```
for(int i=0;i<10;i++)
  {
     lr.message("action()"+i);
     lr.think_time(2);
  }
```

メッセージ関数の詳細については，341 ページ「メッセージ関数」または「**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。

LoadRunner に対して，Java の標準出力ストリームと標準エラー・ストリームを VuGen の実行ログにリダイレクトするように指示できます。これは，仮想ユーザ・スクリプトに既存の Java コードを貼り付けるときや，**System.out** および **System.err** 呼び出しを含む既成のクラスを使用するときに，非常に役に立ちます。実行ログでは，標準出力メッセージは青，標準エラーは赤で示されます。

lr.enable_redirection を使って，特定のメッセージを標準出力および標準エラーへリダイレクトする方法を以下の例に示します。

```
lr.enable_redirection(true);

System.out.println("This is an informatory message..."); // リダイレクトされる
System.err.println("This is an error message..."); // リダイレクトされる

lr.enable_redirection(false);

System.out.println("This is an informatory message..."); // リダイレクトされない
System.err.println("This is an error message..."); // リダイレクトされない
```

注：lr.enable_redirection 関数を **true** に設定すると，この設定が既存のすべてのリダイレクト設定に優先します。以前のリダイレクト設定を復元するには，この関数を **false** に設定します。

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

ユーザの思考時間のエミュレート

連続する操作の間のユーザの待ち時間を「**思考遅延時間**」といいます。仮想ユーザでは，**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します。次の例では，仮想ユーザで次のループに移る前に 2 秒待機しています。

```
for(int i=0;i<10;i++)
  {
     lr.message("action()"+i);
     lr.think_time(2);
  }
```

思考遅延時間の設定は，スクリプト中の値をそのまま使うことも，これらの値の倍数を使うこともできます。LoadRunner による思考時間関数の処理方法を設定するには，[実行環境設定] ダイアログ・ボックスを開きます。詳細については，第 9 章 「実行環境の設定」を参照してください。

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

コマンド・ライン引数の処理

スクリプトを実行するときにコマンド・ライン引数を指定することで，実行時に仮想ユーザ・スクリプトに値を渡すことができます。LoadRunner コントローラでスクリプトのパスに続けてコマンド・ライン・オプションを挿入できます。コマンド・ライン引数を読み取り，値を仮想ユーザ・スクリプトに渡すための関数が 3 つあります。

lr.get_attrib_double	double float 型の引数を取得します。
lr.get_attrib_long	long int 型の引数を取得します。
lr.get_attrib_string	文字列を取得します。

コマンド・ラインの形式は次の 2 つのどちらかを使用します。スクリプト名の後ろに，引数とその値を 2 つ 1 組で指定します。

```
スクリプト名　- 引数 引数値　 - 引数 引数値
```

```
スクリプト名　 / 引数 引数値　 / 引数 引数値
```

次の例は，pc4 というマシンで **script1** を 5 回繰り返すためのコマンド・ライン文字列を示します。

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

コマンド・ライン解析関数の詳細については，「**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・ラインに引数を指定する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**』を参照してください。

Java 環境の設定

Java 仮想ユーザ・スクリプトを実行する前に，仮想ユーザを実行するすべてのマシンで環境変数 PATH と CLASSPATH が正しく設定されていることを確認してください。

- スクリプトをコンパイルして再生するためには，JDK 1.1，1.2，または 1.3 のコンポーネントの完全インストールを実行しておく必要があります。JRE をインストールするだけでは不十分です。1 つのマシンに複数の JDK または JRE がインストールされているのは望ましくありません。可能ならば，不要なバージョンはすべてアンインストールします。
- 環境変数 PATH には，JDK¥bin のパスがなければなりません。
- JDK 1.1.x の場合，環境変数 CLASSPATH には，**classes.zip** のパス（**JDK/lib**）およびすべての LoadRunner クラス（**loadrunner/classes**）が含まれていなければなりません。
- Java 仮想ユーザによって使用されるクラスはすべて，マシンの CLASSPATH 環境変数，または，実行環境設定の［**Classpath**］パネルで設定されているクラスパスに含まれている必要があります

Java 仮想ユーザ・スクリプトの実行

Java 仮想ユーザ・スクリプトは，コンパイル後に実行される点が C 仮想ユーザ・スクリプトとは異なります。C 仮想ユーザ・スクリプトは，インタプリタによって解釈されます。VuGen ではインストールされている JDK から **javac** コンパイラが探し出され，スクリプト内の Java コードがコンパイルされます。このとき，VuGen ウィンドウの最下部に「**コンパイルしています ...**」というステータス・メッセージが表示されます。コンパイル中に発生したエラーは，実行ログに表示されます。スクリプトの中でエラーが発生したコードの位置に移動するには，エラーの行番号が示されているエラー・メッセージをダブルクリックします。エラーを修正し，スクリプトを再実行します。

コンパイルが完了すると，ステータス・メッセージが「**コンパイルしています ...**」から「**実行しています ...**」に変わり，スクリプトの実行が始まります。スクリプトに変更を加えていなければ，次にスクリプトを実行したときには，VuGen によるコンパイルは行われずにスクリプトが実行されます。スクリプトをさらに詳細にデバッグするには，ステップ実行オプションを使ってブレークポイントを設定したり，表示しながらの実行を指定できます。

注：スクリプト内から JNDI の拡張機能を呼び出している場合には，仮想ユーザを**スレッド**として実行しようとするときに問題が生じることがあります。これは，JNDI ではスレッドごとに独自のコンテキスト・クラス・ローダが必要とされているために発生します。スレッドとして実行するには，次の行を **init** セクションの先頭に追加して，仮想ユーザで個別にコンテキスト・クラス・ローダを使用するように指示します。

```
DummyClassLoader.setContextClassLoader();
```

パッケージの一部としてのスクリプトのコンパイルと実行

Java 仮想ユーザ・スクリプトを作成するときに，クラスまたはメソッドが保護されている（protected）他のクラスのメソッドを使用しなければならないことがあります。このタイプのスクリプトをコンパイルしようとすると，コンパイルの段階で，メソッドにアクセスできないことを示すエラーが報告されます。スクリプトの中で確実にこれらのメソッドにアクセスできるようにするには，標準的な Java プログラムで行うのと同様の方法で，これらのメソッドを含むパッケージ名＜ package_name ＞をスクリプトの先頭に挿入します。以下の例では，スクリプトの中で特定のパスにある **just.do.it** パッケージを定義しています。

```
package just.do.it;

import lrapi.*;
public class Actions
{
    :
}
```

上記の例の場合，仮想ユーザ・ディレクトリの下に **just/do/it** というディレクトリの階層が自動的に作成され，**Actions.java** ファイルが **just/do/it/Actions.java** にコピーされます。これにより，ファイルは関連パッケージを使ってコンパイルできるようになります。package ステートメントは Java の場合と同様に，スクリプトの第 1 行目になければなりません（コメントは除く）。

プログラミングについてのヒント

Java 仮想ユーザ・スクリプトのプログラミングを行うときは，既成のコードのセグメントをスクリプトに貼り付けるか，既成のクラスをインポートして，そのメソッドを呼び出せるようにします。仮想ユーザを（スケーラビリティを考慮して）コントローラの管理下でスレッドとして実行しなければならない場合には，インポートするコードがすべてスレッドセーフであることを確認する必要があります。

一般的に，コードをスレッドセーフにするのは簡単ですが，検出は困難です。VuGen およびコントローラでは，仮想ユーザの数が限られていれば，Java 仮想ユーザは問題なく実行されるかもしれません。しかし，ユーザ数が増えると問題が発生します。スレッドセーフでないコードは，通常は静的クラス・メンバを使用していることに起因します。以下に例を示します。

```
import lrapi.*;
public class Actions
{
  private static int iteration_counter = 0;

    public int init() {
        return 0;
    }

    public int action() {
        iteration_counter++;
        return 0;
    }

    public int end() {
        lr.message("Number of Vuser iterations: "+iteration_counter);
        return 0;
    }
}
```

1 個の仮想ユーザを実行すると，**iteration_counter** メンバは実行された反復の回数を正確に示します。1 台の仮想マシンで複数の仮想ユーザを複数のスレッドとして実行すると，静的クラス・メンバの **iteration_counter** がすべてのスレッドで共有されるため，反復回数のカウントが不正確になります。これは，すべての仮想ユーザの反復回数の合計がカウントされるからです。

スレッドセーフでないことがわかっているコードをスクリプトにインポートしたい場合は，それらの仮想ユーザをプロセスとして実行できます。仮想ユーザをスレッドまたはプロセスとして実行する方法の詳細については，第 9 章 「実行環境の設定」を参照してください。

1 個の Java 仮想ユーザ・スクリプトを実行する場合，通常そのスクリプトは 1 個のスレッド（メイン・スレッド）で構成されています。LoadRunner Java API にアクセスできるのはメイン・スレッドだけです。Java 仮想ユーザが 2 次的なワーカー・スレッドを生成したときに，LoadRunner API を使用すると，予期しない結果が生じます。したがって，LoadRunner Java API は，メイン・スレッドの中だけで使用することをお勧めします。この制限は，**lr.enable_redirection** 関数に影響を及ぼします。

以下の例では，LR API を使用してもよい場所と使用してはいけない場所を示します。実行ログの最初のログ・メッセージは，**flag** の値が **false** であることを示します。その後，仮想マシンによって新規スレッド **set_thread** が生成されます。このスレッドは実行され，**flag** が **true** に設定されますが，**lr.message** への呼び出しにもかかわらず，ログにはメッセージが発行されません。最後のログ・メッセージは，スレッド内のコードが実行され，**flag** が **true** に設定されたことを示します。

```
boolean flag = false;

public int action() {
  lr.message("Flag value: "+flag);
  Thread set_thread = new Thread(new Runnable();{
        public void run() {
          lr.message("LR-API NOT working!");
          try { Thread.sleep(1000); } catch(Exception e) {}
          flag = true;
        }
  });
  set_thread.start();
  try { Thread.sleep(3000); } catch(Exception e) {}
  lr.message("Flag value: "+flag);
  return 0;
}
```

第 7 部

分散コンポーネント・プロトコル

第 25 章

COM 仮想ユーザ・スクリプトの記録

Windows アプリケーションの多くは，COM ベースの関数を直接呼び出すか，またはライブラリ呼び出しを介して使用します。VuGen を使って，COM ベースのクライアントによる COM サーバへのアクセスをエミュレートするスクリプトを記録できます。その結果生成されるスクリプトを COM 仮想ユーザ・スクリプトと呼びます。Visual Basic アドインを使って，COM 仮想ユーザ・スクリプトを作成することもできます。Visual Basic アドインの詳細については，第 62 章「Visual Studio による仮想ユーザ・スクリプトの作成」を参照してください。

第 26 章「COM 仮想ユーザ・スクリプトについて」では，VuGen COM スクリプトの仕組みについて説明し，簡単な関数リファレンスを提供します。

本章では，次の項目について説明します。

- COM の概要
- COM 仮想ユーザを使った作業の開始
- 記録対象の COM オブジェクトの選択
- COM 記録オプションの設定

以降の情報は，COM 仮想ユーザ・スクリプトを対象とします。

COM 仮想ユーザ・スクリプトの記録について

COM クライアント・アプリケーションを記録すると，VuGen によって COM クライアントとサーバの動作状況を表す関数が生成されます。記録されたスクリプトには，インタフェースの宣言，API 呼び出し，メソッドのインスタンス呼び出しが含まれています。各 COM 関数には，**lrc** という接頭辞が付いています。

記録されたスクリプトは，VuGen のメイン・ウィンドウで表示および編集ができます。セッション中に記録された COM の API 呼び出しとメソッド呼び出しはウィンドウに表示されるので，アプリケーションの COM/DCOM 呼び出しを目で追うことができます。

仮想ユーザ・スクリプトの作成に使用するプログラミング言語（C または Visual Basic）を指定できます。詳細については，第 4 章 「スクリプト生成オプションの設定」を参照してください。

COM の概要

この節では，COM 技術の概要を説明します。これらは COM 仮想ユーザ・スクリプトを使用する前に最低限知っておくべき情報です。詳細については，『Microsoft Developer Network（MSDN）』などのドキュメントを参照してください。

COM（Component Object Model）は，再利用可能なソフトウェア・コンポーネント（「プラグイン」）を開発するための技術です。DCOM（Distributed COM）は，リモート・コンピュータ上の COM コンポーネントを使用できるようする技術です。MTS（Microsoft Transaction Server），Visual Basic，エクスプローラはすべて，COM/DCOM 技術を使用します。したがって，テスト対象のアプリケーションも，気が付かないところで COM 技術を間接的に使用していることがあります。多くの場合，アプリケーションによって実行された COM 呼び出しの一部（全部ではない）を仮想ユーザ・スクリプトに加えなければならないでしょう。

オブジェクト，インタフェース，タイプ・ライブラリ

COM オブジェクトはバイナリ・コードのモジュールです。各 COM オブジェクトには，クライアント・プログラムとの通信を可能にする 1 つまたは複数のインタフェースが実装されています。仮想ユーザ・スクリプト内の COM 呼び出しを追うには，これらのインタフェースを理解しておく必要があります。COM インタフェースのメソッドおよびパラメータにアクセスするための参照として使用されるタイプ・ライブラリには，COM オブジェクトとインタフェースに関する記述が含まれています。各 COM クラス，インタフェース，タイプ・ライブラリは，GUID（Global Unique Identifier）によって識別されます。

COM インタフェース

COM インタフェースは，相互に関連するメソッドの集合を提供します。たとえば，**Clock** オブジェクトには，**Clock**，**Alarm**，**Timer** のインタフェースがあるかもしれません。各インタフェースには，1 つまたは複数のメソッドがあります。たとえば，**Alarm** インタフェースには，**AlarmOn** メソッドおよび **AlarmOff** メソッドがあるかもしれません。

また，インタフェースは，1 つまたは複数のプロパティを持つことができます。メソッド呼び出しによる方法と，プロパティの値の設定または取得による方法の両方で同じ機能を実行できることがあります。たとえば，**Alarm Status** プロパティを **On** に設定した場合の結果が，**AlarmOn** メソッドを呼び出した場合の結果と同じだったりします。

COM オブジェクトは，多数のインタフェースをサポートすることができます。コンポーネントには必ず **IUnknown** インタフェースが実装されており，他のインタフェースを調べるために使用できます。多くのコンポーネントは，**IDispatch** インタフェースも実装しています。IDispatch インタフェースは，オブジェクトの他のインタフェースとメソッドをすべて公開して，スクリプティング言語での COM オートメーションの実装を可能にします。

COM のクラスのコンテキストと場所の透過性

COM オブジェクトは，クライアント・アプリケーションを実行しているマシン，またはリモート・サーバ上で実行できます。アプリケーションによって作成される COM オブジェクトは，ローカル・ライブラリ，ローカル・プロセス，リモート・マシン（「リモート・オブジェクト・プロキシ」）のいずれかにあります。「コンテキスト」と呼ばれる COM オブジェクトのありかは，アプリケーションにとっては透過的です。大半のユーザは，LoadRunner を使ってリモート・サーバの負荷を検査します。このため，リモート・オブジェクト・プロキシがアクセスするオブジェクトが，通常，負荷テストの対象として最も意味があります。

COM のデータ型

COM は，セーフ配列，BSTR 文字列およびバリアントなど，いくつかの特殊なデータ型も提供します。これらのデータ型は，デバッグやパラメータ化のような作業で使用する必要があるかもしれません。

COM 仮想ユーザを使った作業の開始

この節では，COM 仮想ユーザ・スクリプトの開発工程について説明します。

COM 仮想ユーザ・スクリプトの作成は，次の手順で行います。

1 **VuGen を使って基本となるスクリプトを記録します。**

VuGen を起動し，新しい仮想ユーザ・スクリプトを作成します。COM を仮想ユーザのタイプとして指定します。記録対象アプリケーションを選び，記録オプションを設定します。スクリプト記録オプションの設定については，第 4 章「スクリプト生成オプションの設定」を参照してください。COM 固有のオプションとフィルタの設定方法については，362 ページ「COM 記録オプションの設定」を参照してください。アプリケーションを使って，一般的な操作を記録します。

記録方法の詳細については，第 3 章 「VuGen を使った記録」を参照してください。

2 **オブジェクト・フィルタを適切に設定し直します。**

生成されたログ・ファイルを使って，フィルタで記録対象のオブジェクトを選択し直します。詳細については，「記録対象の COM オブジェクトの選択」の項を参照してください。

3 **スクリプトを拡張します。**

仮想ユーザ・スクリプトにトランザクション，ランデブー・ポイント，および制御フロー構造を挿入して，スクリプトを拡張します。

詳細については，第 6 章 「仮想ユーザ・スクリプトの拡張」を参照してください。

4 **パラメータを定義します（任意）。**

スクリプトに記録された固定値に対してパラメータを定義します。固定値をパラメータに置き換えることによって，異なる値を使って同じビジネス・プロセスを何度も繰り返すことができます。

詳細については，第 7 章 「パラメータの定義」を参照してください。

5 **実行環境を設定します。**

実行環境を設定することによって，スクリプト実行時の仮想ユーザの動作を制御します。設定には，反復，ログ，タイミング情報などがあります。

詳細については，第 9 章 「実行環境の設定」を参照してください。

6 **VuGen からスクリプトを実行します。**

VuGen でスクリプトを保存して実行し，正しく動作することを確認します。

詳細については，第 11 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。

COM 仮想ユーザ・スクリプトが完成したら，Windows プラットフォームでスクリプトをシナリオに組み込みます。仮想ユーザ・スクリプトをシナリオに組み込む方法の詳細については，『**LoadRunner コントローラ・ユーザーズ・ガイド**』を参照してください。

記録対象の COM オブジェクトの選択

テスト対象のアプリケーションは，多数の COM オブジェクトを使用する可能性があります。しかし，実際に負荷を生むのがごく少数の場合，負荷テストで重要となるのは，そのほんの一握りのオブジェクトだけかもしれません。したがって，COM アプリケーションを記録する前に，負荷テスト用に記録するオブジェクトを選択する必要があります。VuGen では，ローカル・マシンまたはネットワーク上の他のコンピュータから読み込めるタイプ・ライブラリのオブジェクトを参照できます。

使用するオブジェクトの決定

テストに含める COM オブジェクトを指定する方法はいくつかあります。テスト対象のソフトウェアによって使用されるリモート・オブジェクトを調べてみます。どのオブジェクトを選択するべきかわからない場合は，標準のフィルタを使用してみます。フィルタの［環境］ノードの下には，リモート・サーバ上で負荷を生成する可能性のある 3 つのオブジェクト（ADO，RDS，Remote）への呼び出しが含まれています。

実際の呼び出しを調べて，フィルタを適切に設定し直すこともできます。テストを記録した後にファイルを保存し，VuGen によって作成された **data** ディレクトリにある **lrc_debug_list_ ＜ nnn ＞ .log**（**nnn** はプロセス番号）ファイルを調べます。このログ・ファイルには，各 COM オブジェクトが記録用フィルタに含まれていたかどうかに関係なく，記録されたアプリケーションによって呼び出された COM オブジェクトの一覧が含まれています。サーバに対する負荷を生成する呼び出しだけを，記録に含める必要があります。

たとえば，以下は Visual Basic ライブラリのローカルの COM の例です。

```
Class JetES {039EA4C0-E696-11D0-878A-00A0C91EC756}
was loaded from type library "JET Expression Service Type Library"
({2358C810-62BA-11D1-B3DB-00600832C573} ver 4.0)
```

これは記録対象として追加する必要はありません。

同様に，以下に示す OLE DB および Microsoft Windows Common Controls はローカル・オブジェクトなので，そのクラスとライブラリは，サーバへの負荷を生成しません。したがって，これらも記録する必要はありません。

```
Class DataLinks {2206CDB2-19C1-11D1-89E0-00C04FD7A829}
was loaded from type library "Microsoft OLE DB Service Component 1.0
Type Library"
({2206CEB0-19C1-11D1-89E0-00C04FD7A829} ver 1.0)

Class DataObject {2334D2B2-713E-11CF-8AE5-00AA00C00905}
was loaded from type library "Microsoft Windows Common Controls 6.0
(SP3)"
({831FDD16-0C5C-11D2-A9FC-0000F8754DA1} ver 2.0)
```

ただし，以下に例として示すクラスは，記録する必要があります。

```
Class Order {B4CC7A90-1067-11D4-9939-00105ACECF9A}
was loaded from type library "FRS"
({B4CC7A8C-1067-11D4-9939-00105ACECF9A} ver 1.0)
```

たとえば VuGen とともにインストールされる flight_sample で使用される **FRS** ライブラリのクラスは，サーバの処理能力を必要するので記録する必要があります。

COM オブジェクトが別の COM オブジェクトを呼び出す場合，すべての呼び出しがタイプ情報ログ・ファイルにリストアップされます。たとえば，アプリケーションが **FRS** クラス関数を呼び出すたびに，**FRS** ライブラリは **ADO**（ActiveX Data Object）ライブラリを呼び出します。このような呼び出しの連鎖に含まれるいくつかの関数がフィルタに表示されている場合，VuGen によって連鎖を開始する最初の呼び出しだけが記録されます。**FRS** および **ADO** 呼び出しの両方を選択した場合は，**FRS** 呼び出しだけが記録されます。また，フィルタで **ADO** ライブラリだけを選択した場合，**ADO** ライブラリへの呼び出しが記録されます。多くの場合，連鎖における最初のリモート・オブジェクトへの呼び出しを記録するのが最も簡単です。ただし，アプリケーションが多数の異な

る COM オブジェクトのメソッドを使用しているものの，それらがすべて，サーバに負荷をかける単独のオブジェクトを使用する場合には，その最終的な共通オブジェクトだけ記録するということが可能です。

選択可能なオブジェクト

VuGen では，オブジェクトのタイプ・ライブラリを読み込むことができれば，そのオブジェクトを記録できます。タイプ・ライブラリがシステムにインストールされていない場合や VuGen でタイプ・ライブラリが見つからない場合には，対応する COM オブジェクトは［記録オプション］ダイアログ・ボックスに表示されません。これらの COM オブジェクトがアプリケーションによって使用されている場合，VuGen ではこれらのオブジェクトを識別できず，ファイル内に **INoTypeInfo** として示されます。

除外できるインタフェース

［記録オプション］ダイアログ・ボックスでは，タイプ・ライブラリのリストに含まれているすべてのインタフェースが，オブジェクトごとに表示されます。このダイアログ・ボックスで，各インタフェースを記録に含めるか除外するかを指定できます。ただし，**ADO**，**RDS**，**Remote のオブジェクト**は，フィルタに 1 つのグループとして含めることができます。その場合，フィルタには，これらの環境またはインタフェースの個々のオブジェクトまたはインタフェースは表示されません。また，タイプ・ライブラリから記録対象にインクルードしたオブジェクトのインタフェースが，タイプ・ライブラリにないために［記録オプション］ダイアログ・ボックスに表示されない場合があります。その場合には，VuGen のスクリプトを生成した後に，スクリプトに含まれるこれらのインタフェースを特定して，VuGen によって生成された interfaces.h ファイルでそれらのインタフェースの GUID 番号を確認します。この情報を使って，以下に説明する方法で，インタフェースを除外できます。

COM 記録オプションの設定

COM の［記録オプション］ダイアログ・ボックスを使って，フィルタ・オプションと COM のスクリプト編集オプションを設定します。レジストリ，ファイル・システム，Microsoft トランザクション・サーバ（MTS）のタイプ・ライブラリを探すには，オンライン・ブラウザを使用します。

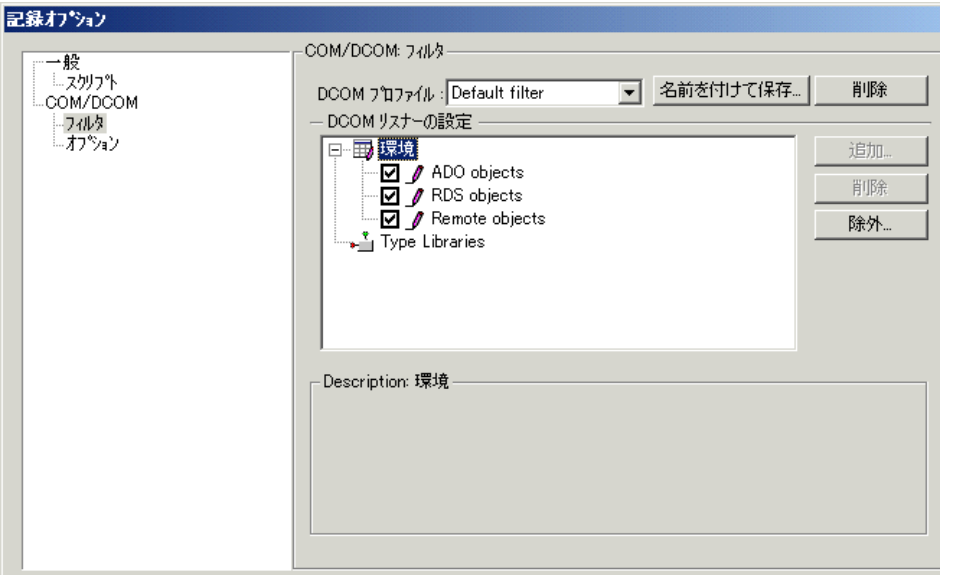
詳細については，下記を参照してください。

- オブジェクトのフィルタリング
- COM スクリプト編集オプションの設定

オブジェクトのフィルタリング

フィルタ・オプションを使って，VuGen で記録する COM オブジェクトを指定できます。オブジェクトは環境およびライブラリの中から選択できます。

フィルタ・オプションで，標準フィルタの設定または代替フィルタの作成を行います。環境およびタイプ・ライブラリを指定することで，記録セッションを絞り込めます。



DCOM Profile

- ［**Default filter**］: COM 仮想ユーザ・スクリプトを記録するときに標準で使用されるフィルタ。
- ［**新規フィルタ**］: 標準環境設定に基づく新しいフィルタ。その設定を記録するには，先にフィルタの名前を指定する必要があります。

DCOM リスナーの設定

［DCOM リスナーの設定］には，タイプ・ライブラリのツリー階層が表示されます。ツリーの分岐を開いて，タイプ・ライブラリで使用できるクラスをすべて表示できます。クラス・ツリーの分岐を開いて，そのクラスによってサポートされているインタフェースをすべて表示できます。

タイプ・ライブラリを除外するには，ライブラリ名の横のチェック・ボックスをクリアします。これによって，そのタイプ・ライブラリに含まれているすべてのクラスが除外されます。ツリーの分岐を開き，各クラスまたはインタフェースの横のチェック・ボックスをクリアすることによって，それらの項目を個別に除外できます。

クラスの種類ごとに，異なるインタフェースを実装できます。除外されていない他のクラスによって実装されているインタフェースを除外しようとすると，このインタフェースを実装しているすべてのクラスのインタフェースも除外するかどうかをたずねるダイアログ・ボックスが表示されます。

インタフェースの横のチェック・ボックスをクリアすると，［**除外インターフェース**］ダイアログ・ボックスでそのインタフェースを選択したときと同じ結果が得られます。

- ［**環境**］: 記録対象の環境。［ADO objects］，［RDS objects］，および［Remote objects］があります。記録しないオブジェクトをクリアします。
- ［**Type Libraries**］: 記録する COM オブジェクトを表す **.tlb** または **.dll** ファイルです。
  すべての COM オブジェクトはそれらを表すタイプ・ライブラリを持ちます。
- タイプ・ライブラリは，レジストリ，Microsoft Transaction Server（MTS），またはファイル・システムから選択できます

ダイアログ・ボックスの下の部分には，タイプ・ライブラリごとに次の情報が表示されます。

- **Description**：タイプ・ライブラリ（tlb ファイル）の名前。

- **Path**：タイプ・ライブラリのパス
- **Guid**：タイプ・ライブラリの GUID（Global Unique Identifier）。

フィルタの設定

この項では，フィルタの設定方法について説明します。

記録対象の COM オブジェクトの選択は，次の手順で行います。

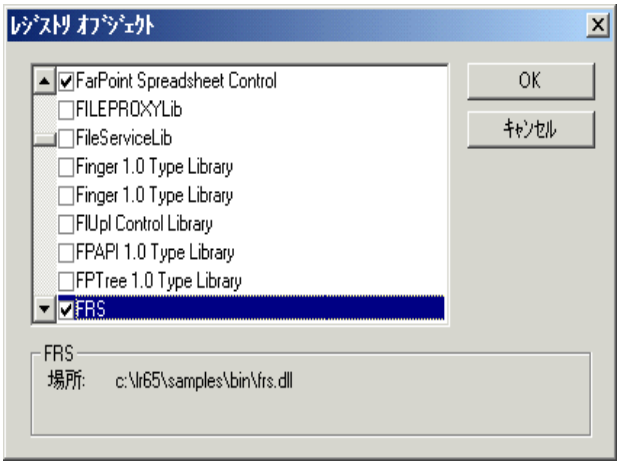
1 メイン・メニューの［**ツール**］＞［**記録オプション**］を選択するか，［記録開始］ダイアログ・ボックスの［**オプション**］をクリックします。記録オプションのツリーが表示されたダイアログ・ボックスが開きます。［COM/DCOM］の［**フィルタ**］ノードを選択します。

［環境］サブツリーをクリックすると，**ADO** オブジェクト，**RDS** オブジェクト，**Remote** オブジェクトが一覧表示されます。フィルタには，［**Type Libraries**］ツリー（最初は空）も含まれています。タイプ・ライブラリは，以下の手順で追加できます。

標準設定では，［環境］のすべての項目が選択されており，それらのオブジェクトへの呼び出しがフィルタに含められています（記録対象になります）。これらのオブジェクトをフィルタから除外するには，［**ADO objects**］，［**RDS objects**］，［**Remote objects**］の横にあるチェック・ボックスをクリアします。

2 別の COM タイプ・ライブラリを追加するには，［**追加**］をクリックし，［レジストリの参照］，［ファイル システムの参照］，［MTS の参照］のいずれかを選択します。

3 ［**レジストリの参照**］を選択すると，ローカル・コンピュータのレジストリに登録されているタイプ・ライブラリのリストが表示されます。



使用するライブラリ（1 つまたは複数）の横のチェック・ボックスを選択し，［**OK**］をクリックします。

4 ファイル・システムからタイプ・ライブラリを追加するには，［**追加**］をクリックして［**ファイル システムの参照**］を選択します。

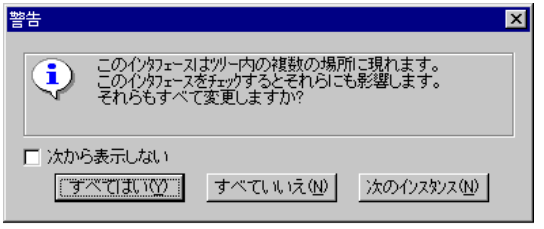
使用するファイルを選択して，［**開く**］をクリックします。

5 タイプ・ライブラリが［Type Libraries］リストに表示されたら，そのツリーの分岐を開いて，タイプ・ライブラリ内の使用可能なクラスをすべて表示できます。クラス・ツリーの分岐を開いて，そのクラスによってサポートされているインタフェースをすべて表示できます。

タイプ・ライブラリを除外するには，ライブラリ名の横のチェック・ボックスをクリアします。これによって，そのタイプ・ライブラリに含まれているすべてのクラスが除外されます。ツリーの分岐を開き，各クラスまたはインタフェースの横のチェック・ボックスをクリアすることによって，それらの項目を個別に除外できます。

インタフェースの横のチェック・ボックスをクリアすると，［除外インターフェース］ダイアログ・ボックスでそのインタフェースを選択したときと同じ結果が得られます。

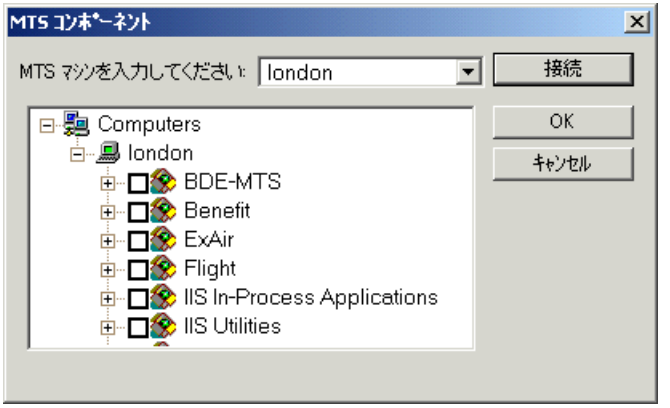
6 クラスの種類ごとに，異なるインタフェースを実装できます。除外されていない他のクラスにも実装されているインタフェースを除外しようとすると，VuGen によって次の警告が表示されます。



［**次から表示しない**］をチェック・ボックスを選択してこのダイアログ・ボックスを閉じると，それ以後，このフィルタを使用し 1 つのオブジェクトに含まれるインタフェースのステータスを変更するたびに，他のクラスに含まれているそのインタフェースのステータスもすべて自動的に変更されます。他のクラスにあるそのインタフェースのステータスもすべて変更するには，［**すべてはい**］をクリックします。その他のクラスにあるそのインタフェースのステータ

スを変更しないときには，［**すべていいえ**］をクリックします。そのインタフェースを使用する次のクラスを表示するには，［**次のインスタンス**］をクリックします。

7 Microsoft Transaction Server のコンポーネントを追加するには，［**追加**］をクリックして［**MTS の参照**］を選択します。MTS サーバの名前を入力するための［MTS コンポーネント］ダイアログ・ボックスが表示されます。

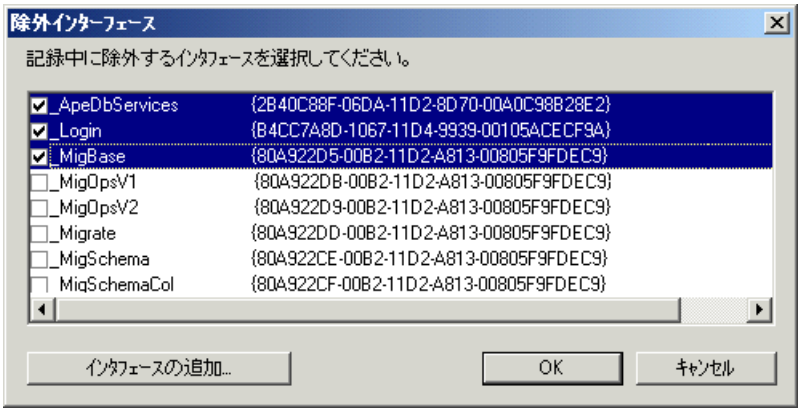


MTS サーバの名前を入力して［**接続**］をクリックします。MTS のコンポーネントを記録するには，マシンに MTS クライアントをインストールしておく必要があります。

使用可能なパッケージのリストから MTS コンポーネントのパッケージを 1 つまたは複数選択して，［**OK**］をクリックします。パッケージが［Type Libraries］のリストに表示されたら，パッケージから特定のコンポーネントを選択します。

8 ツリー表示で各インタフェースの記録を無効にしたり有効にしたりする以外に，［記録オプション］ダイアログ・ボックスの［**除外**］をクリックして，フィルタにインタフェースを含めたり，フィルタから除外したりできます（どのオブジェクトのインタフェースかに関係なく変更できます）。タイプ・ライ

ブラリのツリー階層で，クラスとインタフェースの横のチェック・ボックスをクリアして，対応する項目を除外することもできます。



［除外インターフェース］ダイアログ・ボックスでチェックの印が付いているインタフェースが除外されます。表示されていないインタフェースを追加することもできます。［除外インターフェース］ダイアログ・ボックスで［**インタフェースの追加 ...**］をクリックし，GUID 番号（インタフェース ID）とインタフェース名を入力します。VuGen によって作成され，VuGen 画面の左側のカラムの選択ツリーに表示されている interfaces.h ファイルから，GUID をコピーすることもできます。［**インタフェースの追加 ...**］は，スクリプトによって不必要に呼び出されたものの，フィルタに表示されないインタフェースを除外するために使います。

9 フィルタを変更した場合は，［**OK**］をクリックして保存してから，ダイアログ・ボックスを閉じます。**新しいフィルタ**を保存するとき，または既存のフィルタを新しい名前で保存するときは，［**名前を付けて保存**］をクリックします。保存したフィルタは以降の記録で選択できます。標準設定は，［**Default filter**］で表示できます。

COM スクリプト編集オプションの設定

COM の記録セッションの追加オプションを，オブジェクトの処理，ログの生成，VARIANT 定義に関して設定できます。

DCOM スクリプト編集オプションはすべてのプログラミング言語に適用されます。これらのオプションにより，DCOM メソッドおよびインタフェースの処理に関するスクリプトのオプションを設定できます。

［**ADO レコードセットのフィルタ**］：複数のレコードセットの処理を，1 行の fetch ステートメントにまとめます（標準設定では有効）。

［**レコードセットの内容の保存**］：レコードセットの内容を，後に VuGen で表示できるように記録中にグリッドとして保存します（標準設定では有効）。

［**COM 例外の生成**］：記録中に例外が生成された COM 関数およびメソッドを生成します（標準設定では有効）。

［**COM オブジェクトの解放**］：使用されなくなった COM オブジェクトの解放を記録します（標準設定では無効）。

［**安全配列ログ サイズの制限**］：COM 呼び出しごとの安全配列のログに出力される要素の数を 16 に制限します（標準設定では有効）。

［**COM 統計の生成**］：記録時のパフォーマンスの統計とサマリ情報を生成します（標準設定では無効）。

［**一時 VARIANT のグローバルとしての宣言**］：一時 VARIANT をローカル変数としてではなく，グローバル変数として定義します（標準設定では無効）。

COM スクリプトのオプションの設定は，次の手順で行います。

1 メイン・メニューの［**ツール**］＞［**記録オプション**］を選択するか，［記録開始］ダイアログ・ボックスの［**オプション ...**］をクリックします。記録オプションのツリーが表示されます。［COM/DCOM］オプションの［**オプション**］ノードを選択します。

2 オプション名の隣のチェック・ボックスをクリックしてオプションを選択します。

3 ［**OK**］をクリックして設定を保存し，終了します。

第 26 章

COM 仮想ユーザ・スクリプトについて

本章では，VuGen が COM クライアントの通信を記録して生成するスクリプト，その関数呼び出し，および使用例について詳しく説明します。COM 仮想ユーザ・スクリプトを使った作業を開始する際に知っておくべき基本的な情報については，第 25 章 「COM 仮想ユーザ・スクリプトの記録」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。

- VuGen COM スクリプトの構造について
- VuGen COM スクリプトの例
- スクリプト内での相関候補の検索

以降の情報は，COM 仮想ユーザ・スクリプトを対象とします。

COM 仮想ユーザ・スクリプトについて

COM クライアントの通信を記録すると，COM API 関数およびインタフェース・メソッドへの呼び出しを含むスクリプトが作成されます。このスクリプトに，COM タイプの変換関数をプログラミングすることもできます。各関数呼び出しには，**lrc** という接頭辞が付いています（**lrc_CoCreateInstance** や **lrc_long** など）。本章では，COM API 呼び出しと型変換呼び出しの概要を説明します。各関数の構文と使用例については，「**オンライン関数リファレンス**」（[**ヘルプ**] ＞ [**関数リファレンス**]）を参照してください。

インタフェース・メソッドへの呼び出しの名前と構文の形式は，次のとおりです。

lrc_ ＜インタフェース名＞ _ ＜メソッド名＞ (instance, ...);

instance は常に，最初に渡されるパラメータです。

一般的に，インタフェース関数に関するドキュメントは各 COM コンポーネントのベンダから提供されます。

VuGen では，COM 仮想ユーザ・スクリプトごとに次のものが作成されます。

- interfaces.h ファイルに，インタフェース・ポインタとその他の宣言
- vuser_init，Actions，vuser_end の各セクションに，記録可能な関数呼び出し
- user.h ファイルに，下位レベルの呼び出しに変換された仮想ユーザ・スクリプトのコード

スクリプトを記録すると，VuGen 画面の左側のツリーでこれらのファイルを選択して表示できるようになります。

VuGen COM スクリプトの構造について

VuGen COM スクリプトは，COM インタフェースの要件を満たすために特別な方法で構成されています。

インタフェース・メソッド

インタフェース・メソッドへの呼び出しの名前と構文の形式は，次のとおりです。

lrc_ ＜インタフェース名＞ _ ＜メソッド名＞ (instance, ...);

instance は常に，最初に渡されるパラメータです。

一般的に，インタフェース関数に関するドキュメントは各 COM コンポーネントのベンダから提供されます。

インタフェース・ポインタ

interfaces.h ファイルは，後でスクリプト内で使用されるインタフェース・ポインタとその他の変数を定義します。各インタフェースには，そのインタフェースを一意に識別するインタフェース ID（IID）が割り当てられています。

インタフェースの定義の形式は，次のとおりです。

＜インタフェース・タイプ＞ * ＜インタフェース名＞ = 0; //"{ ＜インタフェース・タイプの IID ＞ }"
次の例では，インタフェース・タイプは **IDispatch**，インタフェースのインスタンスの名前は **IDispatch_0**，**IDispatch** タイプの **IID** は long 型の文字列です。

```
IDispatch* IDispatch_0= 0;//"{00020400-0000-0000-C000-
000000000046}"
```

仮想ユーザ・スクリプトのステートメント

COM 仮想ユーザ・スクリプトは，オブジェクトのインスタンスの作成，インタフェース・ポインタの取得，インタフェース・メソッドの呼び出しを行うコードで構成されます。各ユーザ・アクションは，1 つまたは複数の COM 呼び出しを生成します。COM 呼び出しはそれぞれ，VuGen によってステートメント・グループとして記述されます。グループは，それぞれが独立したスコープとして括弧で囲まれます。いくつかのステートメントによって，値の代入と型変換を行うことによって，main の呼び出しに備えます。オブジェクトの作成に必要な呼び出しのグループの例を次に示します。

```
{
GUID pClsid = lrc_GUID("student.student.1");
IUnknown * pUnkOuter = (IUnknown*)NULL;
unsigned long dwClsContext = lrc_ulong("7");
GUID riid = IID_IUnknown;
lrc_CoCreateInstance(&pClsid, pUnkOuter, dwClsContext, &riid,
(void**)&IUnknown_0, CHECK_HRES);
}
```

エラーの検査

各 COM メソッドまたは API 呼び出しは，エラーの値を返します。VuGen によって，最初の記録時に呼び出しが成功したかどうかに応じて，再生時にエラーを検査するかどうかを示すフラグが設定されます。フラグは関数呼び出しの最後の引数として指定されます。フラグの値は次のいずれかです。

CHECK_HRES	記録時に関数が正しく実行されたため，再生時にエラーを検査する必要がある場合は，この値が挿入されます。
DONT_CHECK_HRES	記録時に関数が失敗したため，再生時にエラーを検査する必要がない場合は，この値が挿入されます。

VuGen COM スクリプトの例

この節では，VuGen が COM クライアント・アプリケーションをエミュレートする仕組みを，例を使って説明します。

COM スクリプトが行う基本的な処理

基本的な処理として，次のことを行います。

- オブジェクトのインスタンスの作成
- インタフェース・ポインタの取得
- インタフェース・メソッドの呼び出し

それぞれの処理は，独立のスコープで実行されます。

オブジェクトのインスタンスの作成

アプリケーションは，COM オブジェクトを使用するために，最初にインスタンスを作成し，そのオブジェクトのインタフェースへのポインタを取得します。

VuGen では，次の手順でオブジェクトのインスタンスが作成されます。

1 VuGen によって **lrc_GUID** が呼び出され，そのオブジェクト用の一意の ProgID が取得されて **pClsid** に格納されます。

```
GUID pClsid = lrc_GUID("student.student.1");
```

pClsid は，**ProgID** の「**student.student.1**」から変換された，オブジェクトの一意のグローバル CLSID です。

2 IUnknown インタフェース・ポインタが，集約オブジェクトへのポインタである場合，VuGen によってそのオブジェクトへのポインタが取得されます。取得できない場合は，VuGen によってポインタが NULL に設定されます。

```
IUnknown * pUnkOuter = (IUnknown*)NULL;
```

3 VuGen によって，作成するオブジェクトのコンテキストが設定されます。

```
unsigned long dwClsContext = lrc_ulong("7");
```

dwClsContext には，オブジェクトのコンテキストが含まれます（プロセス，ローカル，リモート，またはこれらのうちの複数の場所）。

4 VuGen によって，要求されたインタフェース ID を保持する変数が設定されます。この例では，**IUnknown** です。

```
GUID riid = IID_IUnknown;
```

riid には，**IUnknown** インタフェースのインタフェース ID が含まれます。

5 入力パラメータが用意された後に，**lrc_CoCreateInstance** への呼び出しにより，用意されたパラメータを使ってオブジェクトが作成されます。この呼び出しは，次の段階で必要になる **IUnknown** インタフェースへのポインタを返します。

```
lrc_CoCreateInstance(&pClsid, pUnkOuter, dwClsContext, &riid,
(void**)&IUnknown_0, CHECK_HRES);
```

前述のとおり**入力パラメータが用意されています。**呼び出しが成功しているため，VuGen によって **CHECK_HRES** 値が挿入され，ユーザのシミュレーションの実行時にエラーの検査を行うように設定されます。**呼び出しの結果，IUnknown_0 に IUnknown インタフェースへのポインタが返されます。**IUnknown_0 は，以降の呼び出しで使用されます。

インタフェースの取得

オブジェクトを作成した時点で VuGen がアクセスできるのは **IUnknown** インタフェースだけです。VuGen は，オブジェクトと通信するために **IUnknown** インタフェースを使用します。これは，**IUnknown** 標準インタフェースの **QueryInterface** メソッドを使って行われます。VuGen のメソッド呼び出しの最初のパラメータは，インタフェースのインスタンスです。この例では，最初のパラメータは事前に **CoCreateInstance** によって返された **IUnknown_0** ポインタです。**QueryInterface** 呼び出しには，取得すべきインタフェースの ID が入力項目として必要です。QueryInterface 呼び出しによって，指定した ID に対応するインタフェースへのポインタが返されます。

インタフェースの取得は，次の手順で行います。

1 最初に VuGen によって Istudent インタフェースの ID のパラメータである **riid** が設定されます。

```
GUID riid = IID_Istudent;
```

2 **QueryInterface** を呼び出すと，Istudent オブジェクトに **Istudent** インタフェースがあれば，そのインタフェースへのポインタが返されます。

```
lrc_IUnknown_QueryInterface(IUnknown_0, &riid, (void**)&Istudent_0,
CHECK_HRES);
```

インタフェースを使ったデータの設定

インタフェースのメソッドを使って，データを設定する例を以下に示します。たとえば，アプリケーションでユーザが名前を入力するように求められるとします。この場合，名前を設定するためのメソッドが起動されます。VuGen によって 2 つのステートメントが記録されます。1 つは，名前の文字列を組み立てるために使用され，もう 1 つは名前のプロパティを設定します。

この関数呼び出し全体を組み立てるには，次の手順で行います。

1 最初に，VuGen によって変数（**Prop Value**）に，入力された文字列と同じ値が設定されます。パラメータのタイプは，COM ファイルで使用される文字列型である BSTR です。

```
BSTR PropValue = lrc_BSTR("John Smith");
```

後で，「John Smith」をパラメータで置き換えることによって，この呼び出しをパラメータ化すると便利です。これによって，仮想ユーザ・スクリプトを実行するたびに，異なる名前を使用できるようになります。

2 次に，VuGen は名前を入力するための，**Istudent** インタフェースの **Put_Name** メソッドを呼び出します。

```
lrc_Istudent_put_name(Istudent_0, PropValue, CHECK_HRES);
```

インタフェースを使ったデータの返却

値を格納して，以降の呼び出しで入力項目として使用できるようにパラメータ化を行いたい場合は，データを入力するだけでは不十分で，アプリケーションからデータを返さなければなりません。

アプリケーションがデータを取得したときに VuGen によって何が行われるかを次の例に示します。

1 プロパティの値を格納する適切なタイプの変数（この例では BSTR）を作成します。

```
BSTR pVal;
```

2 上記の手順で作成した変数 **pVal** に，プロパティの値（この例では名前）を保存します。この例では，**Istudent** の **get_name** メソッドを使用します。

```
Irc_Istudent_get_name(Istudent_0, &pVal, CHECK_HRES);
```

3 次に，VuGen によってこれらの値を保存するためのステートメントが生成されます。

```
//Irc_save_BSTR("param-name",pVal);
```

このステートメントは，コメントアウトされています。コメントを表す記号（//）を削除して，**param-name** を変数に変更できます。変数には，この値を格納することを表すような名前を付けることができます。VuGen によって，直前の呼び出しによって返された **pVal** の値を保存するために，この変数が使用されます。以降，パラメータ化した入力項目として，他のメソッドへの呼び出しでこの変数を使用できます。

IDispatch インタフェース

ほとんどの COM オブジェクトには，固有のインタフェースがあります。また多くは，汎用インタフェース **IDispatch** も実装しています。このインタフェースは VuGen によって特別な方法で変換されます。**IDispatch** は，IDispatch 以外の COM オブジェクト・インタフェースおよびメソッドをすべて公開する「特別なインタフェース」です。VuGen スクリプトからの **IDispatch:Invoke** メソッドへの呼び出しは，**lrc_Disp** 関数を使って実装されます。これらの呼び出しは，他のインタフェースへの呼び出しとは異なる形式で構成されます。

IDispatch インタフェースの **Invoke** メソッドは，メソッドの実行，プロパティ値の取得，プロパティの値またはプロパティの参照の値の設定を行うことができます。標準の **IDispatch:Invoke** メソッドでは，これらのさまざまな用途は **wflags** パラメータで示されます。VuGen では，これらはメソッドの呼び出し，またはプロパティの設定や取得を行う別々のプロシージャ呼び出しとして実装されます。

たとえば，**GetAgentsArray** メソッドを呼び出すための **IDispatch** への呼び出しは，次のようになります。

```
retValue = lrc_DispMethod1((IDispatch*)IDispatch_0, "GetAgentsArray",
/*locale*/1033, LAST_ARG, CHECK_HRES);
```

上記の呼び出しのパラメータは，次のとおりです。

IDispatch_0	以前に実行された **IUnknown:Queryinterface** メソッドへの呼び出しによって返された **IDispatch** インタフェースへのポインタです。
GetAgentsArray	呼び出すメソッドの名前です。VuGen の実際の動作では，この名前からメソッドの ID が取得されます。
1033	言語ロケールです。
LAST_ARG	引数がこれ以上ないことを **IDispatch** インタフェースに知らせるためのフラグです。
CHECK_HRES	記録時に呼び出しが成功したため，HRES の検査を行うことを示すフラグです。

さらに，**OPTIONAL_ARGS** という別のパラメータが存在することもあります。これは，VuGen によって標準パラメータ以外に追加引数が送られていることを示します。追加引数はそれぞれ，ID または名前と，その値の組み合わせで構成されています。たとえば，次のような場合です。

```
{
        GUID riid = IID_IDispatch;
        lrc_IOptional_QueryInterface(IOptional_0, &riid,
(void**)&IOptional_0, CHECK_HRES);
    }
    {
        VARIANT P1 = lrc_variant_short("47");
        VARIANT P2 = lrc_variant_short("37");
        VARIANT P3 = lrc_variant_date("3/19/1901");
        VARIANT var3 = lrc_variant_scode("4");
        lrc_DispMethod((IDispatch*)IOptional_0, "in_out_optional_args",
/*locale*/1024, &P1, &P2, OPTIONAL_ARGS, "#3", &P3, "var3", &var3,
LAST_ARG, CHECK_HRES);
```

IDispatch インタフェースを使用するさまざまな **lrc_Disp** メソッドの詳細については，「LRC Function Reference（LRC 関数リファレンス）」に記載されています。

型変換とデータ抽出

上の例で示したように，COM パラメータの多くはバリアントとして定義されます。これらの値を抽出するために，COM 関数を基に作成したいくつかの変換関数が VuGen によって使用されます。変換関数の一覧については第 27 章「COM 仮想ユーザ関数について」を参照してください。**lrc_DispMethod1** 呼び出しを使って，名前の文字列の配列を取得する方法については，すでに説明しました（以下の例を参照）。

```
VARIANT retValue = lrc_variant_empty();
retValue = lrc_DispMethod1((IDispatch*)IDispatch_0, "GetAgentsArray",
/*locale*/1033, LAST_ARG, CHECK_HRES);
```

次の例では，VuGen で文字列の配列として読み取られるバリアントである **retValue** から文字列を取り出す方法を示します。

まず，VuGen によってバリアントから BSTR 配列が抽出されます。

```
BstrArray array0 = 0;
array0 = lrc_GetBstrArrayFromVariant(&retValue);
```

array0 内にすべての値が含まれている状態で，後でパラメータ化の際に使用できるように，VuGen によって配列の要素を抽出するためのコードが生成されます。次に例を示します。

```
//GetElementFrom1DBstrArray(array0, 0); // 値：Alex
//GetElementFrom1DBstrArray(array0, 1); // 値：Amanda
....
```

VuGen には，さまざまなタイプの変換関数や，バリアントから従来の型を抽出するための関数があります。これらの詳細については第 27 章「COM 仮想ユーザ関数について」または「**オンライン関数リファレンス**」を参照してください。

スクリプト内での相関候補の検索

VuGen には，スクリプトを修正して確実に再生できるようにするための相関ユーティリティがあります。相関ユーティリティは，次の処理を行います。

- 相関候補を探す。
- 適切な相関関数を挿入して，結果をパラメータに保存する。
- ステートメントの値をパラメータで置き換える。

自動相関は，スクリプト全体またはスクリプト内の特定の場所を対象に実行できます。

この節では，相関させる必要のあるステートメントを見つける方法について説明します。すでに相関させたい値がわかっている場合は，次の項に進んで，特定の値を相関させる方法を参照してください。

自動相関によるスクリプトの検索と相関は，次の手順で行います。

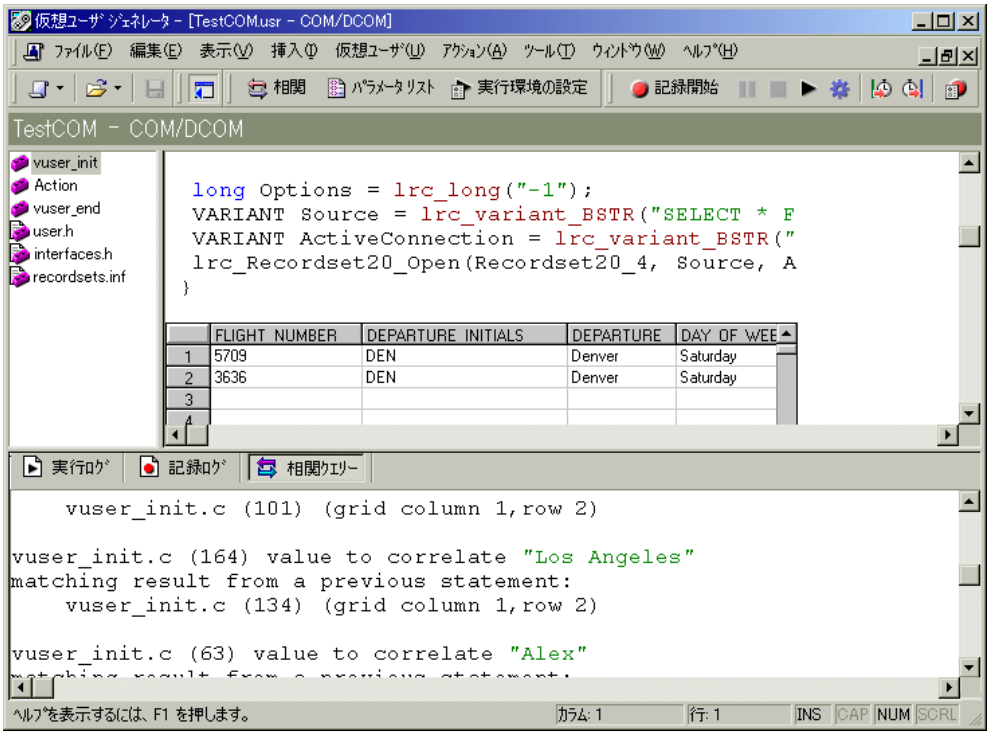
1 ［出力］ウィンドウを開きます。

［**表示**］＞［**出力ウィンドウ**］を選択して，ウィンドウの下部に出力タブを表示します。［実行ログ］タブ内でエラーを確認します。多くの場合，これらのエラーは相関によって修正できます。

2 ［**仮想ユーザ**］＞［**相関を検索**］を選択します。

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

次の例では，VuGen によって lrc_variant_BSTR ("SELECT...") ステートメントの中でいくつかの相関候補値が見つかりました。



3 値を相関させます。

［相関クエリー］タブで，相関させるクエリー結果をダブルクリックします。この例では，メッセージの 3 行目にあります。3 行目は「grid column x, row x」のようになっています。VuGen によって，スクリプト内の値の位置にカーソルが移動します。

4 表示枠の中で，相関させる値を選び，［**仮想ユーザ**］＞［**相関を作成**］を選択します。結果の値を保存するパラメータの名前を入力するように求められます。

5 名前を指定するか，標準設定の名前をそのまま使用します。［**OK**］をクリックして作業を続けます。VuGen によって，パラメータに結果を保存する相関ステートメント（**lrc_save_ ＜タイプ＞**）が挿入されます。

6 ［**はい**］をクリックして相関を確定します。

スクリプト内に現れる値をすべて検索するかどうかを尋ねるメッセージが表示されます。

7 選択したステートメント内の値だけを置き換える場合は，［**いいえ**］をクリックします。

8 次の候補を検索する場合は，［**はい**］をクリックします。

［検索と置換］ダイアログ・ボックスが開きます。オリジナルのステートメントも含め，すべての置き換えを確認します。必要な値をすべて置き換えたら，［**キャンセル**］をクリックして［検索と置換］ダイアログ・ボックスを閉じます。

ステートメントの値はパラメータへの参照に置き換えられます。相関を取り消すと，直前のステップで作成されたステートメントは消去されます。

既知の値の相関

相関させるべき値がわかっている場合は以下の手続きを行います。

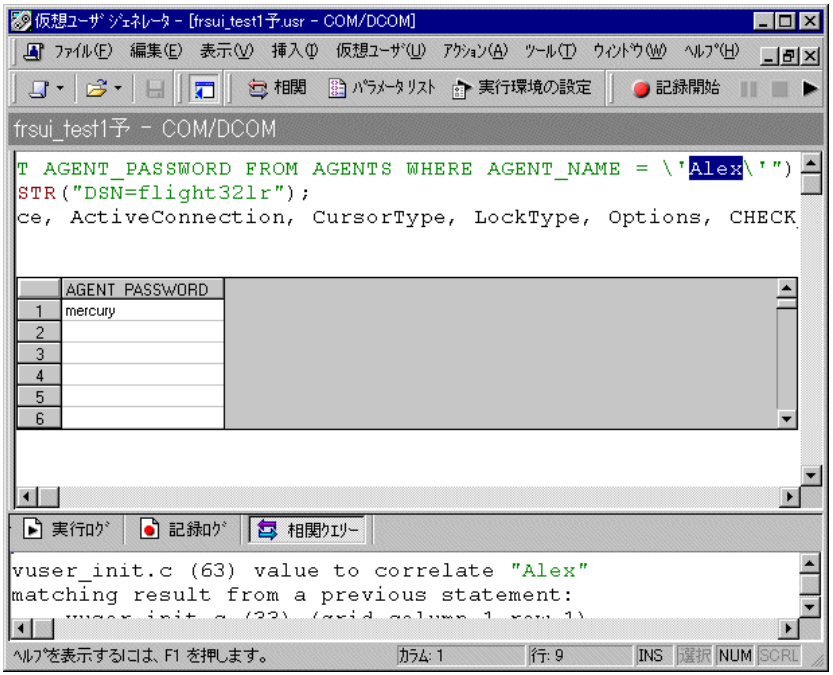
特定の値を相関させるには，次の手順で行います。

1 相関させる値を見つけ，値を選択します（引用符は除きます）。

2 ［**仮想ユーザ**］＞［**アクション内で相関をスキャン**］を選択します。

VuGen によって値が検索され，スクリプト内でこの値と一致した結果がすべて表示されます。相関値は［相関クエリー］タブに一覧表示されます。

次の例では，「Alex」に相関させる候補として 1 つの結果値が見つかっています。



［相関クエリー］タブで，相関させるクエリー結果をダブルクリックします。この例では，メッセージの 3 行目にあります。3 行目は「grid column x, row x」のようになっています。VuGen によって，スクリプト内の値の位置にカーソルが移動します。

3 表示枠の中で，相関させる値を選び，［**仮想ユーザ**］＞［**相関を作成**］を選択します。結果の値を保存するパラメータの名前を入力するように求められます。

4 名前を指定するか，標準設定の名前をそのまま使用します。［**OK**］をクリックして作業を続けます。VuGen によって，パラメータに結果を保存する相関ステートメント（**lrc_save_ ＜タイプ＞**）が挿入されます。

```
lrc_save_rs_param (Recordset20_0, 1, 1, 0, "Saved_AGENT_NAME”);
```

5 ［**はい**］をクリックして相関を確定します。

スクリプト内に現れる値をすべて検索するかどうかを尋ねるメッセージが表示されます。

6 選択したステートメント内の値だけを置き換える場合は，［**いいえ**］をクリックします。

7 次の候補を検索する場合は，［**はい**］をクリックします。

［検索と置換］ダイアログ・ボックスが開きます。オリジナルのステートメントも含め，すべての置き換えを確認します。必要な値をすべて置き換えたら，［**キャンセル**］をクリックして［検索と置換］ダイアログ・ボックスを閉じます。

ステートメントの値はパラメータへの参照に置き換えられます。相関を取り消すと，直前のステップで作成されたステートメントは消去されます。

第 27 章

COM 仮想ユーザ関数について

COM 仮想ユーザ関数は，COM アプリケーションを実行するユーザのアクションをエミュレートします。

本章では，次の項目について説明します。

- インスタンスの作成
- IDispatch インタフェース起動メソッド
- 型指定関数
- バリアント型
- バリアントへの参照の代入
- パラメータ化関数
- バリアントからの抽出
- バリアントへの配列の代入
- 配列の型と関数
- バイト配列関数
- ADO RecordSet 関数
- デバッグ関数
- VB Collection のサポート

以降の情報は，COM 仮想ユーザ・スクリプトを対象とします。

COM 仮想ユーザ関数について

VuGen の COM 関数の名前には lrc という接頭辞が付いています。VuGen では，本章で紹介する COM API 呼び出しとメソッド呼び出しを記録します。また，lrc 型変換呼び出しを手作業でプログラミングすることもできます。lrc 関数の構文と例については，「**オンライン関数リファレンス**」（[**ヘルプ**] > [**関数リファレンス**]）を参照してください。

仮想ユーザ・スクリプトの作成に使用するプログラミング言語（C 言語または Visual Basic スクリプト）を指定できます。詳細については，第 4 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。次の項では C 言語タイプの仮想ユーザ・スクリプトに対して生成される関数について説明します。

インスタンスの作成

オブジェクトの作成と解放を行うための次のような関数があります。これらは，対応する COM 関数から派生したものです。

lrc_CoCreateInstance	オブジェクトのインスタンスを作成し，IUnknown インタフェースを返します。
lrc_CreateInstanceEx	リモート・マシン上のオブジェクトのインスタンスを作成し，複数のインタフェースを返すことができます。
lrc_CoGetClassObject	指定したクラスのクラス・ファクトリを取得します。このクラス・ファクトリを使って，そのクラスの複数のオブジェクトを作成できます。
lrc_Release_Object	使用されていない COM オブジェクトを解放します。

IDispatch インタフェース起動メソッド

以下の呼び出しは，**Invoke** メソッドを使って **IDispatch** インタフェースを起動し，**Invoke** の **wflags** パラメータに，異なるフラグ値を設定します。

lrc_DispMethod	IDispatch:Invoke メソッドを使って，インタフェースのメソッドを起動します。
lrc_DispMethod1	メソッドを起動し，IDispatch インタフェースを使って同じ名前のプロパティを取得します。
lrc_DispPropertyGet	IDispatch インタフェースを使って，プロパティを取得します。
lrc_DispPropertyPut	IDispatch インタフェースを使って，プロパティを設定します。
lrc_DispPropertyPutRef	IDispatch インタフェースを使って，参照によってプロパティを設定します。

型指定関数

VuGen が自動的に記録する関数を補うために，スクリプトに手作業のプログラミングで型指定関数を挿入できます。この型変換関数は文字列データを指定した型に代入します。関数名の形式は次のとおりです。

lrc_ **＜型の名前＞**

ここで，**＜型の名前＞**は，次のデータ型のいずれかです。

ascii_BSTR	ascii BSTR
bool	ブール型
BSTR	BSTR
BYTE	バイト型
char	文字変数
currency	通貨
date	日付
double	倍精度

dword	倍精度ワード
float	浮動小数点数
GUID	指定したオブジェクトの GUID
hyper	hyper 型整数
int	整数
long	long 型整数
short	short 型整数
uint	符号なし整数
ulong	符号なし long 型整数
uhyper	符号なし 64 ビット hyper 型整数
ushort	符号なし short 型整数

バリアント型

バリアントには任意の型の情報を含めることができます。たとえば，バリアントは文字列の配列にも倍精度ワードにもなります。また，バリアントの配列をバリアントとすることもできます。VuGen では文字列データをさまざまなバリアント型に変換できます。関数名の形式は，次のとおりです。

lrc_variant_ **＜型の名前＞**

ここで，**＜型の名前＞**は，次のデータ型のいずれかです。

ascii BSTR	ascii BSTR のバリアント
bool	ブール型のバリアント
BSTR	BSTR のバリアント
BYTE	符号なし文字（バイト型）のバリアント
char	文字
CoObject	**IUnknown** インタフェース・ポインタ
currency	通貨のバリアント

date	日付のバリアント
DispObject	**IDispatch** インタフェース・ポインタ
float	浮動小数点数のバリアント
int	整数のバリアント
long	long 型整数のバリアント
scode	エラー・コードのバリアント
short	short 型整数のバリアント
uint	符号なし整数のバリアント
ulong	符号なし long 型整数のバリアント
ushort	符号なし short 型整数バリアント

バリアント型変換関数以外に，次の 3 つの関数でも新しいバリアントが作成されます。

lrc_variant_empty	空のバリアントを作成します。
lrc_variant_null	NULL のバリアントを作成します。
lrc_variant_variant_by_ref	既存のバリアントを格納する新しいバリアントを作成します。

バリアントへの参照の代入

VuGen は変数をバリアントに変換し，その参照をバリアントとして代入することができます。関数名の形式は，次のとおりです。

lrc_variant_ ＜型の名前＞ _by_ref

ここで，**＜型の名前＞**は，次のデータ型のいずれかです。

ascii BSTR	ascii BSTR のバリアント
bool	ブール型のバリアント
BSTR	BSTR のバリアント
BYTE	BYTE のバリアント
char	文字のバリアント
CoObject	**IUnknown** インタフェース・ポインタ
currency	通貨のバリアント
date	日付のバリアント
DispObject	**IDispatch** インタフェース・ポインタ
float	浮動小数点数のバリアント
int	整数のバリアント
long	long 型整数のバリアント
scode	scode 型のバリアント
short	short 型整数のバリアント
uint	符号なし整数のバリアント
ulong	符号なし long 型整数のバリアント
ushort	符号なし short 型整数のバリアント
from_variant	バリアント内からバリアントを取得

パラメータ化関数

パラメータ化関数は，指定された型の値を文字列パラメータに保存します。パラメータ化関数名の形式は，次のとおりです。

lrc_save_ ＜型の名前＞
lrc_save_VARIANT_ ＜型の名前＞
＜型の名前＞で指定された型の変数をバリアントとして保存します。

lrc_save_VARIANT_ ＜型の名前＞ _by_ref
＜型の名前＞で指定された型のバリアントを参照としてバリアントに保存します。

「**値**」は**＜型の名前＞**で指定された型から文字列に変換されます。値はパラメータに格納されます。これらのステートメントは VuGen によってコメントアウトされます。これらのステートメントを使用するには，パラメータ名をその内容を表すものに変更し，ステートメントのコメント記号を削除します。これでパラメータを以降の呼び出しの入力として使用できます。ここで，**＜型の名前＞**は，次のデータ型のいずれかです。

ascii_BSTR	ascii BSTR
bool	ブール型
BSTR	BSTR
BYTE	バイト型
char	文字型
currency	通貨
date	日付
double	倍精度
dword	倍精度ワード
float	浮動小数点数
hyper	hyper 型整数
int	整数
long	long 型整数
uint	符号なし整数

ulong	符号なし long 型整数
short	short 型整数
uhyper	符号なし hyper 型整数
ushort	符号なし short 型整数
VARIANT	バリアント

グリッド内で相関を要求した場合，VuGen は COM スクリプトをパラメータ化するための save ステートメントも追加します。

バリアントからの抽出

バリアントからデータを抽出できるようにするいくつかの関数があります。

lrc_CoObject_from_variant	バリアントから IUnknown インタフェースへのポインタを抽出します。
lrc_CoObject_by_ref_from_variant	バリアントに保存されている参照を使って IUnknown インタフェースへのポインタを抽出します。
lrc_DispObject_from_variant	バリアントから IDispatch インタフェースへのポインタを抽出します。
lrc_DispObject_by_ref_from_variant	バリアントに保存されている参照を使って IDispatch インタフェースへのポインタを抽出します。

バリアントへの配列の代入

次の関数は，配列をバリアントに変換します。

lrc_variant_ ＜型の名前＞ Array	**＜型の名前＞**で指定された型の配列をバリアントに代入します。
lrc_variant_ ＜型の名前＞ Array_by_ref	**＜型の名前＞**で指定された型の配列をバリアントに代入します。バリアントには配列の参照が保存されます。

配列の型と関数

VuGen の COM は，セーフ配列を扱う関数をサポートしています。

Create ＜ n ＞ D ＜型の名前＞ Array	＜型の名前＞で指定された型の n 次元配列を作成します。
Destroy ＜型の名前＞ Array	＜型の名前＞で指定された型の配列を破棄します。
GetElementFrom ＜ n ＞ D ＜型の名前＞ Array	＜型の名前＞で指定された型の要素を SafeArray から取得します。
PutElementIn ＜ n ＞ D ＜型の名前＞ Array	適切な型の配列に要素を格納します。
lrc_Get ＜型の名前＞ ArrayFromVariant	＜型の名前＞で指定された型の配列をバリアントから抽出します。
lrc_Get ＜型の名前＞ Array_by_refFromVariant	＜型の名前＞で指定された型の配列をバリアント内のポインタ参照から抽出します。
Fill ＜ n ＞ DbyteArray	バイト配列の最後の次元を，指定された n-1 のインデックス位置から始まるバッファで埋めます。

上記の関数の**＜型の名前＞**は，次のデータ型のいずれかです。

Bstr	BSTR
Byte	バイト（符号なし文字）
Char	文字配列
CoObject	**IUnknown** インタフェース
Currency	通貨（CY）
Date	日付変数
DispObject	**IDispatch** インタフェース
Double	倍精度

Dword	倍精度ワード
エラー	scode 型のエラー
Float	浮動小数点数
Int	整数
Long	long 型整数
Short	短整数
UInt	符号なし整数
ULong	符号なし long 型整数
UShort	符号なし short 型整数
Variant	バリアント型

バイト配列関数

次の 2 つの関数は，バイトの配列からだけデータを取得できます。

Fill ＜ n ＞ DByteArray	バイト配列の最後の次元を，指定された n-1 のインデックス位置から始まるバッファで埋めます。
GetBufferFrom ＜ n ＞ DByteArray	n 次元のバイト配列の最後の次元から，指定された n-1 のインデックス位置にあるバッファを取得します。

lrc_CreateVBCollection 呼び出しは，バリアントのセーフ配列である Visual Basic のコレクションに特別な方法で対応しています。VuGen はこのコレクションをインタフェースのように扱います。このコレクションに初めて遭遇したときに，VB は **lrc_CreateVBCollection** を使って「インタフェース」を作成します。これによって，VB はインタフェースのアドレスにあるデータを参照できます。

ADO RecordSet 関数

次に ADO Recordset 関数を示します。

lrc_FetchRecordset	Recordset 内でポインタを移動します。
lrc_FetchRecordsetUntillEOF	Recordset の終わりまでのレコードを取得します。
lrc_RecordsetWrite	ADO Recordset 内のフィールドを更新します。
lrc_RecordsetAddColumn	新しいカラムを Recordset に追加します。
lrc_RecordsetDeleteColumn	Recordset のカラムを削除します。

デバッグ関数

lrc_print_variant 関数はバリアントの内容を出力します。

VB Collection のサポート

lrc_CreateVBCollection 関数は Visual Basic Collection オブジェクトを作成します。

第 28 章

CORBA-Java 仮想ユーザ・スクリプトの作成

VuGen では，CORBA を使用する Java アプリケーションまたはアプレットを記録できます。記録したスクリプトは，そのまま実行したり，標準の Java ライブラリ関数および LoadRunner 固有の Java 関数を使って拡張したりできます。

本章では，次の項目について説明します。

- Corba-Java 仮想ユーザの記録
- Corba-Java 仮想ユーザ・スクリプトを使った作業
- Windows XP および Windows 2000 サーバでの記録

以降の情報は，CORBA-Java 仮想ユーザ・スクリプトを対象とします。

Corba-Java 仮想ユーザ・スクリプトについて

VuGen を使って，CORBA（Common Object Request Broker Architecture）Java アプリケーションやアプレットを記録できます。VuGen によって，LoadRunner 固有の Java 関数で拡張されたピュア Java スクリプトが作成されます。記録後，JDK ライブラリまたはカスタムのクラスを使って，標準 Java コードでスクリプトの拡張や修正ができます。

スクリプトを用意したら，VuGen を使ってスタンドアロン・モードで実行します。Sun の標準 Java コンパイラ，**javac.exe** によってエラーのないことが確認された後，スクリプトがコンパイルされます。スクリプトが正しく機能することを確認したら LoadRunner のシナリオに組み込みます。

記録および手動拡張によってスクリプトを作成する場合，Java 仮想ユーザ・スクリプトに関連したすべてのガイドラインと制限が適用されます。また，スクリプトで使用するクラス（**org.omg.CORBA.ORB** など）は，仮想ユーザを実行しているマシン上になければならず，**classpath** 環境変数に指定されていなければなりません。関数の構文とシステム設定に関する重要な情報については，第 24 章 「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。

Windows XP および 2000 Server で記録を行う場合には，403 ページ「Windows XP および Windows 2000 サーバでの記録」のガイドラインに従います。

以降の各章で，Java の記録オプション，実行環境の設定，相関について説明します。

Corba-Java 仮想ユーザの記録

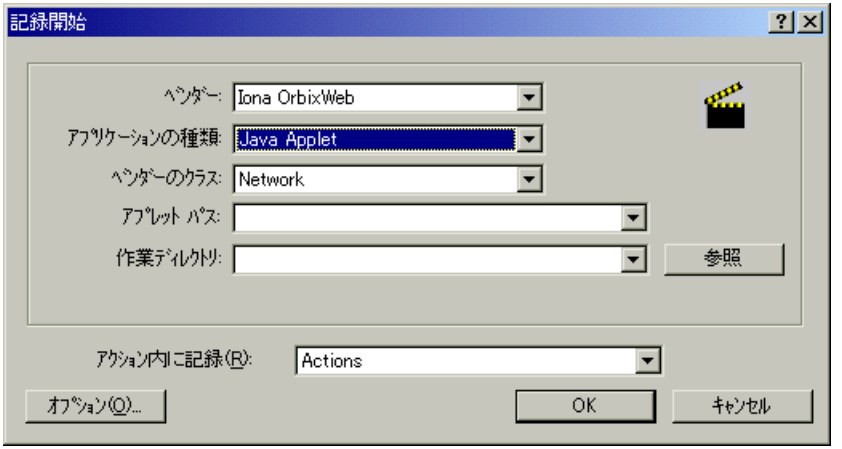
CORBA 仮想ユーザの記録を開始する前に，アプリケーションまたはアプレットが記録用のマシンで正しく動作することを確認します。

LoadRunner を実行しているマシンに，Sun の JDK が正しくインストールされていなければなりません（JRE がインストールされているだけでは不十分です）。スクリプトを記録する前に，インストールを完了させておく必要があります。**classpath** および **path** 環境変数が，JDK のインストール時の指示に従って設定されていることを確認します。

必要な環境設定の詳細については，第 24 章 「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。

記録を開始するには，次の手順で行います。

1 ［**ファイル**］ > ［**新規作成**］を選び，［分散コンポーネント］カテゴリから［Corba-Java］を選択します。［記録開始］ダイアログ・ボックスが開きます。



2 ［ベンダー］リストから CORBA ベンダーを選択します。

3 ［**アプリケーションの種類**］ボックスで，以下の選択肢から適切な値を選択します。

［**Java Applet**］: Sun のアプレットビューアを使って Java アプレットを記録します。

［**Java Application**］: Java アプリケーションを記録します。

［**Netscape**］または［**IExplore**］: ブラウザ内のアプレットを記録します。

［**Executable/Batch**］: バッチ・ファイルから起動されるアプレットまたはアプリケーションを記録します。

［**Listener**］: 記録の前に設定を初期化してアプリケーションを実行するバッチ・ファイルを待機するよう VuGen に指示します。このモードでは，JDK 1.2.x 以上を使って，システム変数 _JAVA_OPTIONS に値 **--Xrunjdkhook** を設定しなければなりません（JDK 1.1.x の場合，環境変数 **_classload_hook=JDKhook** を定義します）。

4 CORBA クラスがネットワークからダウンロードされる場合は，［**ベンダーのクラス**］ボックスで［**Network**］を選択します。CORBA クラスがローカルでロードされる場合（JDK 1.2 以上など）は，［**Local**］だけがサポートされます。

5 次の表に従って，追加パラメータを指定します。

アプリケーションの種類	設定するフィールド
Java Applet	［アプレット パス］，［作業ディレクトリ］
Java Application	［アプリケーション メイン］，［作業ディレクトリ］，［アプリケーション パラメータ］
IExplore	［IExplore パス］，［URL］
Netscape	［Netscape パス］，［URL］
Executable/Batch	［実行可能 / バッチ］，［作業ディレクトリ］
Listener	なし

作業ディレクトリを指定する必要があるのは，アプリケーションに作業ディレクトリの場所を指定する必要がある（たとえば，プロパティ・ファイルの読み込みや，ログ・ファイルの作成をする）場合だけです。

6 JVM のコマンド・ライン・パラメータなどの記録オプションを設定するには，［**オプション**］をクリックします。記録オプションの設定の詳細については，第 14 章 「Java 記録オプションの設定」を参照してください。

7 [**アクション内に記録**] ボックスで，記録を開始するメソッドを選択します。メソッドは，3 つあります。**vuser_init**，**Actions**，**vuser_end** の 3 つです。次の表に，各メソッドに何を含め，どのタイミングで呼び出されるかを示します。

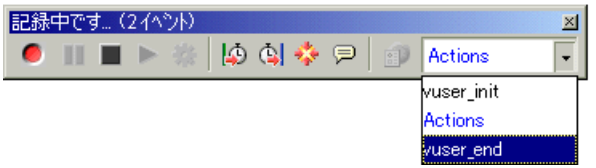
Action クラス内のメソッド	記録対象アクション	エミュレーション内容	実行のタイミング
init	**vuser_init**	サーバへのログイン	初期化時
action	**Actions**	クライアントの動作	実行時
end	**vuser_end**	ログオフ処理	終了時または停止時

注：必ず vuser_init セクションで **org.omg.CORBA.ORB** 関数をインポートして，この関数が反復のたびに呼び出されないようにします。

8 [**OK**] をクリックして記録を開始します。VuGen によってアプリケーションが開始されます。VuGen は最小化され，進行状況バーとフローティング・ツールバーが開きます。進行状況バーには，そのときロードされているクラスの名前が表示されます。これは，Java 記録機能が動作中であることを示します。

9 アプリケーション内で，記録したい標準的な操作を行います。記録中にメソッドを切り替えるには，フローティング・ツールバーを使います。

10 標準的なユーザ・アクションを記録した後で，フローティング・ツールバーで **vuser_end** メソッドを選択します。



ログオフ手順を実行します。VuGen によって手順がスクリプトの **vuser_end** メソッドに記録されます。

11 記録ツールバーの［**記録停止**］をクリックします。VuGen スクリプト・エディタに，記録されたすべてのステートメントが表示されます。

12 ［**保存**］をクリックして，スクリプトを保存します。［テストを保存］ダイアログ・ボックスが開きます（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。

Corba-Java 仮想ユーザ・スクリプトを使った作業

通常，CORBA 固有のスクリプトには，明確なパターンがあります。最初のセクションには，ORB の初期化処理と設定が含まれています。次のセクションでは，CORBA オブジェクトの場所を指定します。その次のセクションは，CORBA オブジェクトでのサーバ呼び出しで構成されます。最後のセクションには，ORB を閉じるシャットダウンの処理が含まれています。必ずこのパターンに従わなければならないわけではありません。また，これらのセクションは 1 つのスクリプト内に複数現れることがあります。

次に示すスクリプトのコードでは，ORB インスタンスを初期化し，バインドの処理を実行して CORBA オブジェクトを取得しています。VuGen で必要なクラスをすべてインポートしている点に注目してください。

```
import org.omg.CORBA.*;
import org.omg.CORBA.ORB.*;
import lrapi.lr;

public class Actions {

  // public 関数：init
  public int init() throws Throwable {

  // Orb インスタンスを初期化する ...
  MApplet mapplet = new MApplet("http://chaos/classes/", null);
  orb = org.omg.CORBA.ORB.init(mapplet, null);

  // サーバへのバインド
  grid = grid_dsi.gridHelper.bind("gridDSI", "chaos");
  return lr.PASS;
}
```

org.omg.CORBA.ORB 関数は，ORB への接続を行います。そのため，この関数は 1 回だけ呼び出します。複数の反復を実行するときは，この関数を **init** セクションに置きます。

次に示すコードでは，CORBA オブジェクト（grid）を対象に実行したアクションが記録されています。

```
// public 関数：action
public int action() throws Throwable {

   grid.width();
   grid.height();
   grid.set(2, 4, 10);
   grid.get(2, 4);

   return lr.PASS;
}
```

セッションの最後に，VuGen によって ORB のシャットダウンが記録されました。記録されたコード全般に渡って使用された変数は，**end** メソッドの末尾から **Actions** クラスの終了の括弧（}）までの間に現れます。

```
// public 関数：end
    public int end() throws Throwable {

        if ( lr.get_vuser_id() == -1 )
            orb.shutdown();

        return lr.PASS;
}

// 変数セクション
    org.omg.CORBA.ORB orb;
    grid_dsi.grid grid;
}
```

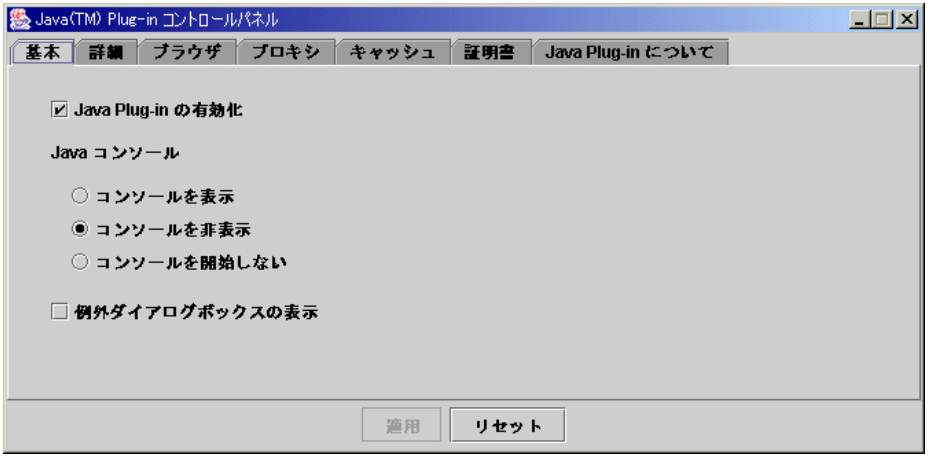
ORB シャットダウン・ステートメントは本製品用にカスタマイズされています。このカスタマイズは，1 つの仮想ユーザのシャットダウンによって他のすべての仮想ユーザがシャットダウンされないようにするものです。

Windows XP および Windows 2000 サーバでの記録

Windows XP および Windows 2000 サーバで記録を行う場合，Java プラグインが VuGen のレコーダとの互換性がないことがあります。正常に動作させるには，Java プラグインのインストール後，スクリプトの記録を開始する前に次の手順を実行します。

Corba-Java または Rmi-Java の記録用にマシンを設定するには，次の手順で行います。

1 ［コントロール パネル］から［Java Plug-in］を開きます。［**スタート**］＞［**設定**］＞［**コントロール パネル**］を選択して，［**Java Plug-in**］コンポーネントを開きます。［基本］タブが開きます。



2 ［**Java Plug-In の有効化**］チェック・ボックスをクリアして，［**適用**］をクリックします。その後，［**Java Plug-In の有効化**］チェック・ボックスを選択しなおして，［**適用**］をクリックします。

3［ブラウザ］タブを開きます。

4［**Microsoft Internet Explorer**］チェック・ボックスをクリアして，［**適用**］をクリックします。その後，［**Micorosoft Internet Explorer**］チェック・ボックスを選択しなおして，［**適用**］をクリックします。

第 29 章

RMI-Java 仮想ユーザ・スクリプトの開発

VuGen では，Java で書かれた RMI を使用するアプリケーションまたはアプレットを記録できます。記録したスクリプトは，そのまま実行したり，標準の Java ライブラリ関数および LoadRunner 固有の Java 関数を使って拡張したりすることができます。

本章では，次の項目について説明します。

- RMI over IIOP の記録
- RMI 仮想ユーザの記録
- RMI 仮想ユーザ・スクリプトを使った作業

以降の情報は，RMI-Java 仮想ユーザ ・スクリプトを対象とします。

RMI-Java 仮想ユーザ・スクリプトの開発について

VuGen を使って，RMI（Remote Method Invocation）Java アプリケーションまたはアプレットを記録できます。VuGen によって，LoadRunner 固有の Java 関数で拡張されたピュア Java スクリプトが作成されます。記録後，JDK のライブラリまたはユーザ定義のクラスを使って，標準 Java コードでスクリプトの拡張や修正ができます。

スクリプトの準備ができたら，VuGen からスタンドアロン・モードで実行します。Sun の標準 Java コンパイラ，**javac.exe** によって，スクリプトにエラーがないか確認された後，コンパイルされます。スクリプトが正しく機能することを確認したら，LoadRunner のシナリオに組み込みます。

スクリプトを記録と手動で拡張する場合，Java 仮想ユーザ・スクリプトに関連したすべてのガイドラインと制限が適用されます。また，スクリプトで使用するクラスは，仮想ユーザを実行するマシンに存在する必要があり，**classpath** 環境変数に指定されている必要があります。関数の構文とシステムの設定で留意すべき点については，第 24 章「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。

Windows XP および 2000 Server で記録を行う場合には，403 ページ「Windows XP および Windows 2000 サーバでの記録」のガイドラインに従います。

RMI over IIOP の記録

IIOP（Internet Inter-ORB Protocol）技術は，CORBA のソリューションをインターネット上で実装することを目的に開発されました。HTTP とは異なり，IIOP では，ブラウザとサーバが配列などの複雑なオブジェクトを交換できます。HTTP は，テキストの送信のみをサポートしています。

「**RMI-IIOP**」技術によって，これまで RMI または CORBA クライアントからのみアクセス可能だったサービスに，1 つのクライアントからアクセスできるようになります。この技術は，RMI で使用される JRMP プロトコルと，CORBA で使用される IIOP を組み合わせたものです。**RMI-IIOP** によって，CORBA クライアントは，**EJB**（Enterprise Java Beans）や，その他の J2EE 標準の新しい技術にアクセスすることができます。

VuGen では **RMI-IIOP** プロトコルを使用する仮想ユーザの記録と再生を完全サポートします。記録する内容によりますが，VuGen の RMI レコーダを使用して実際のユーザを適切にエミュレートするスクリプトを作成できます。

- **ピュア RMI クライアント**：リモート呼び出しのためのネイティブの JRMP プロトコルを使用するクライアントの記録
- **RMI-IIOP クライアント**：（CORBA サーバに対応するために）JRMP の代わりに IIOP プロトコルを使用してコンパイルされたクライアント・アプリケーションの記録

RMI 仮想ユーザの記録

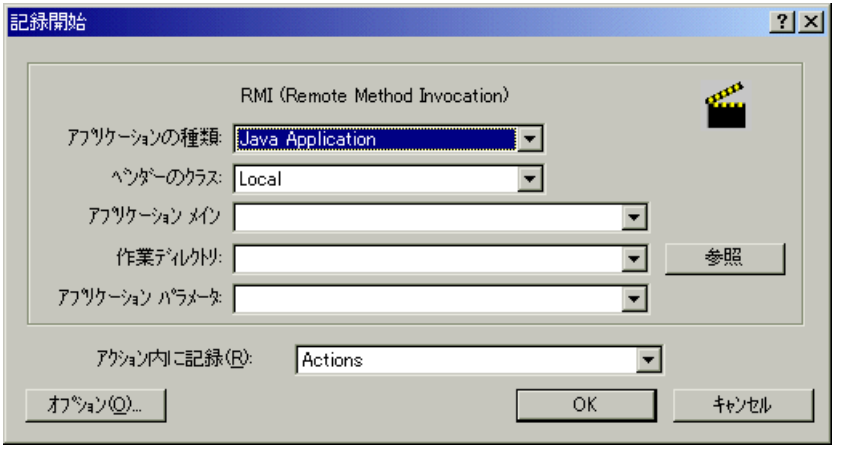
RMI 仮想ユーザを記録する前に，記録するマシンでアプリケーションまたはアプレットが正しく機能することを確認してください。

LoadRunner を実行するマシンに，Sun の JDK を正しくインストールしておく必要があります（JRE だけでは不十分です）。仮想ユーザ・スクリプトを記録する前に，このインストールを完了させておく必要があります。**classpath** および **path** 環境変数が，JDK のインストール時の指示に従って設定されていることを確認します。

記録を行う前に，使用する環境が正しく設定されていることを確認します。使用するクラスがクラスパスに含まれており，JDK の完全インストールが済んでいることを確認します。必要な環境設定の詳細については，第 24 章 「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。

記録中に VuGen でアプレットまたはアプリケーションをロードすると，LoadRunner を使わずにそれらをロードする場合よりも，多少時間がかかります。

1 記録を開始するには，［**ファイル**］＞［**新規作成**］を選択し，［分散コンポーネント］グループから［Rmi-Java］を選択します。［記録開始］ダイアログ・ボックスが開きます。



2 ［**アプリケーションの種類**］ボックスで，以下の選択肢から適切な値を選択します。

［**Java Applet**］：Sun のアプレットビューアを使って Java アプレットを記録します。

［**Java Application**］：Java アプリケーションを記録します。

［**Netscape**］または［**IExplore**］：ブラウザ内のアプレットを記録します。

［**Executable/Batch**］：バッチ・ファイルから起動されるアプレットまたはアプリケーションを記録します。

［**Listener**］：記録の前に設定を初期化してアプリケーションを実行するバッチ・ファイルを待機するよう VuGen に指示します。このモードでは，JDK 1.2.x 以上を使って，システム変数 **_JAVA_OPTIONS** に値 **--Xrunjdkhook** を設定しなければなりません（JDK 1.1.x の場合，環境変数 **_classload_hook=JDKhook** を定義します）。

3 ［**ベンダーのクラス**］ボックスで［**Network**］または［**Local**］を選択します。

4 次の表に従って，追加パラメータを指定します。

アプリケーションの種類	**設定するフィールド**
Java Applet	［アプレット パス］，［作業ディレクトリ］
Java Application	［アプリケーション メイン］，［作業ディレクトリ］，［アプリケーション パラメータ］
IExplore	［IExplore パス］，［URL］
Netscape	［Netscape パス］，［URL］
Executable/Batch	［実行可能 / バッチ］，［作業ディレクトリ］
Listener	なし

［作業ディレクトリ］は，アプリケーションに作業ディレクトリの場所を指定する必要がある（たとえばプロパティ・ファイルの読み込みやログ・ファイルの作成をする）場合だけです。

5 JVM のコマンド・ライン・パラメータなどの記録オプションを設定するには，［**オプション**］をクリックします。記録オプションの設定の詳細については，第 14 章 「Java 記録オプションの設定」を参照してください。

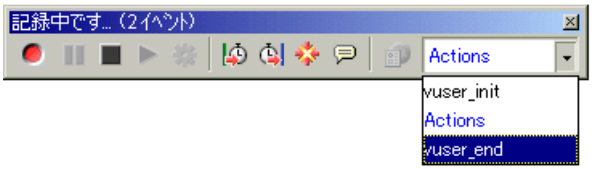
6 ［**アクション内に記録**］ボックスで，記録を挿入するセクションを選択します。Actions クラスには，**init**，**action**，**end** の 3 つのメソッドが含まれています。次の表に，各メソッドに何を含め，どのタイミングで呼び出されるかを示します。

Actions クラス内のメソッド	記録対象アクション	エミュレーション内容	実行のタイミング
init	**vuser_init**	サーバへのログイン	初期化時
action	**Actions=**	クライアントの動作	実行時
end	**vuser_end**	ログオフ処理	終了時または停止時

7 ［**OK**］をクリックして記録を開始します。VuGen によってアプリケーションが開始されます。VuGen は最小化され，進行状況バーとフローティング記録ツールバーが開きます。進行状況バーには，そのときロードされているクラスの名前が表示されます。これは，Java 記録機能が動作中であることを示します。

8 アプリケーション内で，記録したい標準的な操作を行います。記録中にメソッドを切り替えるには，フローティング・ツールバーを使います。

9 標準的なユーザ・アクションを記録した後で，フローティング・ツールバーで **vuser_end** セクションを選択します。



ログオフ手順を実行します。VuGen によって手順がスクリプトの **end** メソッドに記録されます。

10 記録ツールバーの［**記録停止**］をクリックします。VuGen スクリプト・エディタに，記録されたすべてのステートメントが表示されます。

11 ［**上書き保存**］をクリックして，スクリプトを保存します。［テストを保存］ダイアログ・ボックスが開きます（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。

RMI 仮想ユーザ・スクリプトを使った作業

本項では，RMI 仮想ユーザ特有の Java 仮想ユーザ・スクリプトの要素について説明します。RMI は CORBA のような構造要素はなく，Serializable Java オブジェクトを使用します。最初のセクションでは，ネーミング・レジストリの初期化と設定を行います。次のセクションは，Java オブジェクト（Remote および Serializable）が見つかり，キャストされた場合に生成されます。その次のセクションには，Java オブジェクトを対象とするサーバ呼び出しが含まれます。RMI は CORBA とは異なり，専用のシャットダウン・セクションがありません。スクリプトの中でオブジェクトが何度も現れることがあります。

次に示すコードでは，ネーミング・レジストリを検索しています。この処理の後に，特定の Java オブジェクトを取得するための検索処理が行われます。その後，オブジェクトを使って **set_sum**，**increment**，**get_sum** といった呼び出しを実行できます。このコード例は，VuGen で必要とされるすべての RMI クラスをどのようにインポートするかを示します。

```
Import java.rmi.*;
Import java.rmi.registry.*;

:
:

public 関数：action
public int action() throws Throwable {

    _registry = LocateRegistry.getRegistry(“localhost”,1099);

    counter = (Counter)_registry.lookup("Counter1");

    counter.set_sum(0);
    counter.increment();
    counter.increment();
    counter.get_sum();

    return lr.PASS;
    }
 :
```

RMI 仮想ユーザを記録する際，スクリプトにはすべての関連オブジェクトのシリアル化を解除する **lr.deserialize** への複数の呼び出しが含まれていることがあります。**lr.deserialize** 呼び出しは，次の呼び出しに渡されるオブジェクトが，それ以前の呼び出しの戻り値と相関できない場合に生成されます。この場合，VuGen によってオブジェクトの状態が記録され，再生時に **lr.deserialize** 呼び出しを使って，オブジェクトの値が提示されます。シリアライズの解除は，VuGen によってオブジェクトがパラメータとして呼び出しに渡される前に行われます。詳細については，214 ページ「シリアル化メカニズムの使用」を参照してください。

第 8 部

E - ビジネス・プロトコル

第 30 章

FTP 仮想ユーザ・スクリプトの作成

VuGen では，FTP サーバに直接アクセスすることによってネットワークの動作状況をエミュレートできます。

本章では，次の項目について説明します。

- FTP 関数の処理

以降の情報は，FTP 仮想ユーザ・スクリプトを対象とします。

FTP 仮想ユーザ・スクリプトの開発について

FTP プロトコルは，FTP サーバに対して作業しているユーザのアクションをエミュレートする，下層プロトコルです。

FTP に関して，ユーザが FTP サーバにログインし，ファイルを転送してログアウトするのをエミュレートします。スクリプトを作成するには，FTP セッションを記録するか FTP 関数を手作業で入力します。

FTP セッションを記録するときに，VuGen は，メール・クライアントのアクションをエミュレートする関数を生成します。FTP，HTTP，およびメール・プロトコルなど，複数のプロトコルを介して通信を行う場合は，それらを全部記録できます。複数のプロトコルを指定する手順については，第 3 章「VuGen を使った記録」を参照してください。

FTP プロトコルのスクリプトを作成するには，［e ビジネス］カテゴリで［File Transfer Protocol（FTP)］プロトコル・タイプを選択し，FTP サーバに対する標準的なアクションを実行し記録します。スクリプトの作成と記録の詳細については，第 3 章 「VuGen を使った記録」を参照してください。

作成した仮想ユーザ・スクリプトは，Windows または UNIX プラットフォームの シナリオに統合します。仮想ユーザ・スクリプトのシナリオへの統合の詳細については，『**LoadRunner コントローラ・ユーザーズ・ガイド**』を参照してください。

FTP 関数の処理

仮想ユーザ・スクリプトの作成に使用するプログラミング言語を指定できます。詳細については，第 4 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。次の節では C 言語を使用した仮想ユーザ・スクリプトについて説明します。

FTP 仮想ユーザ・スクリプト関数はファイル転送プロトコル（FTP）を記録します。各 FTP 関数は，**ftp** 接頭辞で始まります。

大部分の FTP 関数は，グローバル・セッションに使う関数と特定のメール・セッションの場所を示す関数が対になっています。すべてのセッションにアクションを適用するには，**ex** 接尾辞のないバージョンを使用します。特定のセッションにアクションを適用するには，**ex** 接尾辞のセッション識別子を持つバージョンを使用します。たとえば，**ftp_logon** はグローバルに FTP サーバにログオンしますが，**ftp_logon_ex** を使うと特定のセッションの FTP サーバにログオンします。

関数名	説明
ftp_delete[_ex]	FTP サーバからファイルを削除します。
ftp_dir[_ex]	FTP サーバで dir コマンドを実行します。
ftp_get[_ex]	FTP サーバからファイルを取得します。
ftp_get_last_error	FTP サーバから受信した最後のエラーを取り出します。
ftp_get_last_error_id	FTP サーバから受信した最後のエラーの ID を取り出します。
ftp_logon[_ex]	FTP サーバにログオンします。
ftp_logout[_ex]	FTP サーバからログアウトします。
ftp_mkdir[_ex]	FTP サーバ・マシン上にディレクトリを作成します。
ftp_put[_ex]	FTP サーバにファイルを置きます。
ftp_rendir[_ex]	FTP サーバ・マシン上のディレクトリ名を変更します。
ftp_rmdir[_ex]	FTP サーバ・マシン上のディレクトリを削除します。

ftp_get[_ex]，**ftp_put[_ex]**，および **ftp_dir[_ex]** 関数には，FTP セッションを正確にエミュレートできるようにする属性を設定できます。

PATH：FTP サーバにアップロードするファイルを指定します（MSOURCE_PATH が指定されていない場合にだけ使用できます）。

MPATH：FTP サーバにアップロードする複数のファイルを指定します（**ftp_dir** 以外）。

TARGET_PATH（任意）：サーバ・マシンにファイルを置くパスとファイル名を指定します（**ftp_put** のみ）。

LOCAL_PATH（任意）：サーバ・マシンにファイルを置くパスとファイル名を指定します（**ftp_get** のみ）。

MODE（任意）：検索モードを ASCII またはバイナリ（標準）に指定します。

PASSIVE（任意）：サーバとの通信を PASSIVE 転送モードに設定します。

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。

次の例では，**ftp_delete** 関数を使って FTP サーバから **test.txt** ファイルを削除しています。

```
Actions()
{
        ftp_logon("FTP","URL=ftp://user:pwd@ftp.merc-int.com",
                "LocalAddr=ca_server:21",
                LAST);

        ftp_delete("Ftp_Delete",
                "PATH=/pub/for_jon/test.txt", ENDITEM ,
                LAST);

        ftp_logout();
    return 1;
}
```

第 31 章

LDAP 仮想ユーザ・スクリプトの作成

VuGen で LDAP サーバとの通信をエミュレートできます。

本章では，以下の項目について説明します。

- LDAP 関数の処理
- 識別名エントリの定義

本章の内容は，LDAP 仮想ユーザ・スクリプトにのみ適用されます。

LDAP 仮想ユーザ・スクリプトの作成について

LDAP（**Lightweight Directory Access Protocol**）は，ディレクトリ・データベースにアクセスするのに使用するプロトコルです。LDAP ディレクトリは，多くの LDAP エントリで構成されています。各 LDAP エントリは，DN（識別名）と呼ばれる名前と属性の集合です。DN の詳細については，423 ページ「識別名エントリの定義」を参照してください。

LDAP ディレクトリ・エントリは，政治的，地理的，組織的な境界を反映した階層構造で配置されています。国を表すエントリは，ツリーの一番上に現れます。その下には州や全国的な組織名を表すエントリが表示されます。さらにその下には，個人や組織，プリンタ，ドキュメントなどのエントリが表示されます。

VuGen では LDAP サーバとの通信を記録できます。VuGen によってユーザのアクションをエミュレートする関数を使ったスクリプトが生成されます。このスクリプトには，LDAP サーバへのログインとログアウト，エントリの追加と削除，およびエントリの照会が記述されます。

LDAP プロトコルのスクリプトを作成するには，［e ビジネス］カテゴリで［Lightweight Directory Access Protocol（LDAP）］プロトコル・タイプを選択し，LDAP サーバに対する典型的な操作を実行し記録します。記録の手順については，第 3 章 「VuGen を使った記録」を参照してください。

仮想ユーザ・スクリプトを作成したら，Windows または UNIX プラットフォームのシナリオに統合します。仮想ユーザ・スクリプトのシナリオへの統合の詳細については，『**LoadRunner コントローラ・ユーザーズ・ガイド**』を参照してください。

LDAP 関数の処理

仮想ユーザ・スクリプトの作成に使用するプログラミング言語が指定できます。詳細は，第 4 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。次の節では C 言語を使用した仮想ユーザ・スクリプトについて説明します。

LDAP 仮想ユーザ・スクリプト関数は，LDAP プロトコルをエミュレートします。LDAP 関数は，**mldap** という接頭辞が付いています。

LDAP 関数はすべて，グローバル・セッション用の関数と局所的な特定のセッションを指定できる関数の対になっています。すべてのセッションにアクションを適用するには，接尾辞 **ex** のないバージョンを使用します。特定のセッションにアクションを適用するには，セッション識別子を指定できる接尾辞 **ex** を持つバージョンを使用します。たとえば，**mldap_logon** はグローバルに LDAP サーバにログオンしますが，**mldap_logon_ex** は特定のセッションの LDAP サーバにログオンします。

関数名	説明
mldap_add	LDAP ディレクトリにエントリを追加します。
mldap_add_ex	特定のセッションで LDAP ディレクトリにエントリを追加します。
mldap_delete	エントリまたは属性を削除します。
mldap_delete_ex	特定のセッションでエントリまたは属性を削除します。
mldap_get_attrib_name	属性名を取得します。
mldap_get_attrib_name_ex	特定のセッションの属性名を取得します。
mldap_get_attrib_value	現在のエントリの属性値を取得します。
mldap_get_attrib_value_ex	特定のセッションで現在のエントリの属性値を取得します。
mldap_get_next_entry	次の検索結果を表示します。

mldap_get_next_entry_ex	特定のセッションで次の検索結果を表示します。
mldap_logon	LDAP サーバへのログオンを実行します。
mldap_logon_ex	特定のセッションで LDAP サーバへのログオンを実行します。
mldap_logoff	LDAP サーバからのログアウトを実行します。
mldap_logoff_ex	特定のセッションで LDAP サーバへからのログアウトを実行します。
mldap_modify	エントリの属性値を変更します。
mldap_modify_ex	特定のセッションでエントリの属性値を変更します。
mldap_rename	エントリ名を変更します。
mldap_rename_ex	特定のセッションでエントリ名を変更します。
mldap_search	LDAP サーバに対して検索を実行します。
mldap_search_ex	特定のセッションで LDAP サーバに対する検索を実行します。

これらの関数の構文についての詳細は，「**オンライン関数リファレンス**」（[**ヘルプ**] ＞ [**関数リファレンス**]）を参照してください。

次の例では，ユーザが LDAP サーバ「ldap1」にログオンしています。このユーザはエントリを追加し，OU 属性を「Sales」から「Marketing」に変更しています。

```
Actions()
{
		// LDAP サーバにログオン
		mldap_logon("Login",
		"URL=ldap://johnsmith:tiger@ldap1:80",
				LAST);

		// Sally R. Jones にエントリを追加
		mldap_add("LDAP Add",
			"DN=cn=Sally R. Jones,OU=Sales, DC=com",
			"Name=givenName", "Value=Sally", ENDITEM,
			"Name=initials", "Value=R", ENDITEM,
			"Name=sn", "Value=Jones", ENDITEM,
			"Name=objectClass", "Value=contact", ENDITEM,
			LAST);

		// Sally の OU を「Marketing」に変更
		mldap_rename("LDAP Rename",
			"DN=CN=Sally R. Jones,OU=Sales,DC=com",
			"NewDN=OU=Marketing",
			LAST);

		// LDAP サーバからログアウト
		mldap_logoff();
	return 0;
}
```

識別名エントリの定義

LDAP API では，オブジェクトをその「**識別名**」（DN）で参照します。DN は，カンマで区切られた一連の「**相対識別名**」（RDN）です。

RDN は，属性と関連する値を **attribute=value** という形式で表したものです。属性名では大文字と小文字は区別されません。最も一般的な RDN 属性の型を次の表に示します。

文字列	属性の型
DC	domainComponent
CN	commonName
OU	organizationalUnitName
O	organizationName
STREET	streetAddress
L	localityName
ST	stateOrProvinceName
C	countryName
UID	userid

次に識別名の例を示します。

```
DN=CN=John Smith,OU=Accounting,DC=Fabrikam,DC=COM
DN=CN=Tracy White,CN=admin,DC=corp,DC=Fabrikam,DC=COM
```

次に属性値に使用できない予約文字を示します。

文字	説明
	文字列の先頭にスペースまたは # 文字は指定できません。 文字列の末尾にスペース文字は指定できません。
,	カンマ
+	プラス記号

"	二重引用符
¥	バックスラッシュまたは円記号
<	左山括弧
>	右山括弧
;	セミコロン

予約語を属性値の一部として使用するには，その前にエスケープ文字であるバックスラッシュまたは円記号（¥）を付けます。属性値に等号（=）や非 UTF-8 文字など他の予約文字が含まれる場合は，その文字を 16 進形式でエンコードする必要があります（バックスラッシュまたは円記号の後ろに 16 進数が 2 桁）。

次にエスケープ文字を含む DN の例を示します。最初の例は，カンマの埋め込まれた部門名で，2 番目の例は，キャリッジ・リターンを含む値です。

```
DN=CN=Bitwise,OU=Docs¥, Support,DC=Fabrikam,DC=COM
DN=CN=Before¥0DAfter,OU=Test,DC=North America,DC=Fabrikam,DC=COM
```

第 32 章

Web 仮想ユーザ・スクリプトの作成

VuGen を使用して，Web 仮想ユーザ・スクリプトを作成できます。VuGen では，クライアントのブラウザを操作するユーザ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。

本章では，次の項目について説明します。

- Web 仮想ユーザの紹介
- Web 仮想ユーザ技術について
- Web 仮想ユーザ・スクリプト入門
- Web セッションの記録
- Web 仮想ユーザ・スクリプトの Java への変換

以降の情報は，Web（HTML/HTTP），SOAP，および PeopleSoft8 仮想ユーザ・スクリプトを対象とします。

Web 仮想ユーザ・スクリプトの作成について

VuGen を使用して，Web 仮想ユーザ・スクリプトを作成できます。標準的なユーザ操作を実行し Web サイトをナビゲートしている間に，VuGen によってユーザのアクションが記録され仮想ユーザ・スクリプトが生成されます。生成されたスクリプトを実行すると，仮想ユーザによってインターネットにアクセスするユーザがエミュレートされます。PeopleSoft8 プロトコルは，Web プロトコルに UTF-8 文字エンコーディングのサポート機能が追加されたものです。

仮想ユーザ・スクリプトを作成した後，VuGen を使用して，スクリプトをスタンドアロン・モードで実行します。実行に成功すれば，仮想ユーザ・スクリプトを シナリオに組み込むことができます。仮想ユーザ・スクリプトをシナリオに組み込む方法の詳細については，『**LoadRunner コントローラ・ユーザーズ・ガイド**』を参照してください。

Web 仮想ユーザの紹介

会社の製品情報を表示する Web サイトがあったとします。このサイトには，見込み顧客がアクセスします。多数のユーザ（たとえば 200 人）がサイトに同時にアクセスしても，顧客のあらゆるクエリーに対する応答時間が指定時間（たとえば 20 秒）以内であることを確認するには，仮想ユーザを使用して，Web サーバに対する情報の同時要求をエミュレートします。このとき各仮想ユーザは次のような操作を行うものと考えられます。

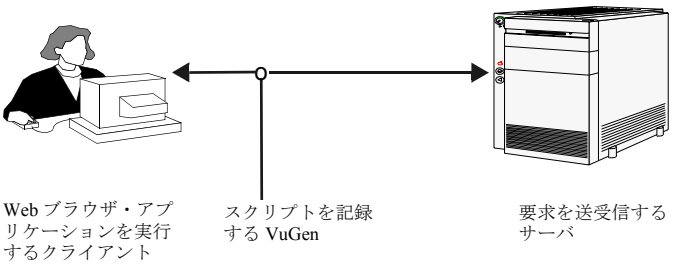
- ホーム・ページのロード
- 製品情報が掲載されているページへの移動
- クエリーの発行
- サーバからの応答の待機

利用可能なテスト用マシンに，数百の仮想ユーザを分散配置できます。各仮想ユーザでは API を通じてサーバにアクセスします。これにより，多数のユーザによる負荷の下でサーバのパフォーマンスを測定できます。

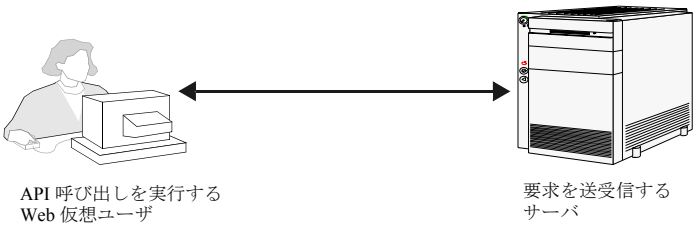
サーバ API の呼び出しを含むプログラムを仮想ユーザ・スクリプトと呼びます。仮想ユーザ・スクリプトでは，ブラウザ・アプリケーションと，ブラウザが実行するすべてのアクションをエミュレートします。コントローラを使用して，1 つのスクリプトを複数の仮想ユーザに割り当てます。これらの仮想ユーザによってスクリプトが実行され，Web サーバのユーザ負荷がエミュレートされます。

Web 仮想ユーザ技術について

VuGen では，ブラウザと Web サーバの間のやり取りを記録することによって，Web 仮想ユーザ・スクリプトを作成します。VuGen でシステムのクライアント側（ブラウザ）を監視し，サーバとの間で送受信されるすべての要求を追跡します。



記録された仮想ユーザ・スクリプトを VuGen あるいは LoadRunner コントローラから実行するときに，仮想ユーザはサーバと直接通信し，クライアント・ソフトウェアに依存しません。仮想ユーザ・スクリプトでは，クライアント・ソフトウェアを使わず，API 関数を使って Web サーバへの呼び出しを直接実行します。



Web 仮想ユーザ・スクリプト入門

本項では，Web 仮想ユーザ・スクリプトの作成プロセスの概要を説明します。

Web 仮想ユーザ・スクリプトの作成は，次の手順で行います。

1 **VuGen を使って新しいスクリプトを作成します。**

［**ファイル**］＞［**新規作成**］を選択するか，［**新規作成**］ボタンをクリックして，シングル・プロトコルまたはマルチ・プロトコル・モードで，「e ビジネス」カテゴリから新規の「Web（HTTP/HTML）」スクリプトを作成します。

新規スクリプトの作成の詳細については，第 3 章 「VuGen を使った記録」を参照してください。

2 **記録オプションを設定します。**

記録オプションを設定します。記録オプションの設定については，第 35 章「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。Web プロトコル固有の記録オプションの詳細については，第 36 章 「Web 仮想ユーザの記録オプションの設定」を参照してください。

3 **ブラウザ・セッションを記録します。**

Web サイトをナビゲートしている間のアクションが記録されます。

新規スクリプトの作成の詳細については，第 3 章 「VuGen を使った記録」を参照してください。

4 **記録した仮想ユーザ・スクリプトの機能を拡張します。**

トランザクション，ランデブー・ポイント，チェック，サービス・ステップを挿入して，仮想ユーザ・スクリプトを拡張します。

詳細については，第 39 章 「負荷下の Web ページ検証」，第 40 章 「Web とワイヤレス仮想ユーザ・スクリプトの変更」と第 41 章 「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。

5 **パラメータを定義します（任意）。**

スクリプトに記録された固定値の代わりにパラメータを定義します。固定値をパラメータで置き換えれば，その値を毎回変えて，同じ仮想ユーザのアクションを何度でも繰り返せます。

詳細については，第 7 章 「パラメータの定義」を参照してください。

6 **実行環境を設定します。**

実行環境を設定することによって，スクリプト実行中の仮想ユーザの動作を制御します。この設定には，一般的な実行環境の設定（反復，ログ，思考遅延時間，一般情報）と Web 関連の設定（プロキシ，ネットワーク，HTTP の詳細）が含まれます。

詳細については，第 9 章 「実行環境の設定」を参照してください。

7 **相関を実行します。**

仮想ユーザ・スクリプトの相関を探し出し，仮想ユーザのメカニズムの 1 つを使って，その相関を実現します。詳細については，第 41 章「Web 仮想ユーザ・スクリプトの相関ルールの設定」または第 42 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。

8 **VuGen で仮想ユーザ・スクリプトを実行してデバッグします。**

VuGen から仮想ユーザ・スクリプトを実行して，スクリプトが正しく実行されることを確認します。詳細については，第 11 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」と第 44 章 「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。

作成した仮想ユーザ・スクリプトを，LoadRunner のシナリオに組み込みます。仮想ユーザ・スクリプトをシナリオに組み込む方法の詳細については，『**LoadRunner コントローラ・ユーザーズ・ガイド**』を参照してください。

Web セッションの記録

Web セッションを記録するとき，VuGen によって Web ブラウザで実行されるすべてのアクションが監視されます。ハイパーリンク・ジャンプ（ハイパーテキストとハイパーグラフィックの両方）やフォーム送信などもアクションに含まれます。記録中，VuGen によって記録対象のアクションが Web 仮想ユーザ・スクリプトに保存されます。

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。VuGen で記録中に，記録対象の関数を挿入する対象となるスクリプトのセクションを選択できます。通常，**vuser_init** と **vuser_end** セクションは，サーバのログインとログオフ手続きの記録に使います。これらのセクションは仮想ユーザ・スクリプトを何度も反復するときに，反復されない部分です。したがって，ブラウザ・セッションを完全な形で反復するためには，Web セッションを Actions セクションに記録する必要があります。

記録中，VuGen によって関数および関連リソースが**同時実行グループ**に挿入されます。同時実行グループは，同時にページにロードされるリンクとリソースを表します。たとえば，ブラウザでは通常，最初の画像をロードしているときに，2 つ目あるいは 3 つ目の画像のロードが開始されます。同時実行グループのステートメントは **web_concurrent_start** 関数と **web_concurrent_end** 関数で囲まれます。再生時には，LoadRunner によって **web_concurrent_start** 関数が見つかるとそれに続く関数が登録されますが，グループの終わりの **web_concurrent_end** 関数が見つかるまでは実行しません。同時実行グループ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。

Web 仮想ユーザ・スクリプトの Java への変換

VuGen には，Web 仮想ユーザ用に作成したスクリプトを Java 仮想ユーザに変換するユーティリティがあります。これにより，Web および Java の仮想ユーザの両方を混合した仮想ユーザ・スクリプトを作成できます。

Web 仮想ユーザ・スクリプトの Java 仮想ユーザ・スクリプトへの変換は，次の手順で行います。

1 空の Java 仮想ユーザ・スクリプトを作成して保存します。

2 空の Web 仮想ユーザ・スクリプトを作成して保存します。

3 標準の HTML/HTTP 記録オプションで Web セッションを記録します。

4 仮想ユーザ・スクリプトを再生します。正常に再生できたら，スクリプト全体を切り取り，テキスト・ドキュメントに貼り付け，テキストとして **.txt** ファイルに保存します。テキスト・ファイルでパラメータの括弧を Web 形式の「"{ }"」から Java 形式の「"< >"」に変更します。

5 DOS コマンド・ウィンドウを開いて＜ LoadRunner のインストール先＞ ¥dat ディレクトリに移動します。

6 次のコマンドを入力します。

＜ LoadRunner のインストール先＞ ¥bin¥sed -f web_to_java.sed **filename** ＞ **outputfilename**

ここで **filename** には先に保存したテキスト・ファイルのフル・パスとファイル名を指定し，**outputfilename** には出力ファイルのフル・パスとファイル名を指定します。

7 出力ファイルを開いて，ファイルの内容を仮想ユーザ・スクリプトの Action 部分の適切な場所にコピーします。内容を空のユーザ定義 Java テンプレート（Java 仮想ユーザ・タイプ）に貼り付ける場合は，public int action() を含む行を次のように変更します。

```
public int action() throws Throwable
```

記録を行うことによって作成する Java 仮想ユーザ（RMI および Corba）では，この変更は自動的に行われます。

通常の Java スクリプトと同様に，仮想ユーザ・スクリプトのパラメータ化と相関を行った後，スクリプトを実行して動作を確認します。

第 33 章

Web 仮想ユーザ関数の使用

VuGen を使用して，Web 仮想ユーザ・スクリプトを作成できます。VuGen では，クライアントのブラウザを操作している間のユーザのアクションを記録して，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。

- 関数の追加と編集
- Web 関数リスト
- ツリー・ビューでのスクリプトの表示
- スクリプト・ビューでの仮想ユーザ・スクリプトの表示

以降の情報は，Web (HTML/HTTP)，SOAP，ワイヤレス，および PeopleSoft8 仮想ユーザ・スクリプトを対象とします。

Web 仮想ユーザ関数について

ブラウザまたはツールキットと Web サーバの間のインターネット通信をエミュレートするために開発された関数を，**Web 仮想ユーザ関数**といいます。各 Web 仮想ユーザ関数の名前には，**web** という接頭辞が付きます。関数の中には，スクリプトの記録時に生成されるものと，手作業でスクリプトに挿入しなければならないものがあります。

インターネット・プロトコル関数（Web，ワイヤレスなど）の詳細な情報や例については，「**オンライン関数リファレンス**」（[**ヘルプ**] > [**関数リファレンス**]）を参照してください。

VuGen は，Web またはワイヤレス仮想ユーザ・スクリプトを次の 2 つの方法で表示できます。

- 仮想ユーザ・スクリプトをアイコン形式で表示する**ツリー・ビュー**（標準設定，WAP 仮想ユーザでは使用できません）。詳細については，442 ページ「ツリー・ビューでのスクリプトの表示」を参照してください。
- 仮想ユーザ・スクリプトをテキスト形式で表示する「**スクリプト・ビュー**」。詳細については，445 ページ「スクリプト・ビューでの仮想ユーザ・スクリプトの表示」を参照してください。

関数の追加と編集

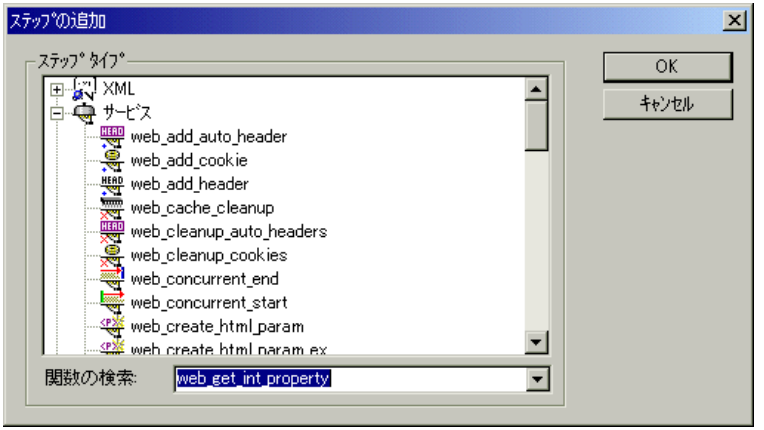
多くの Web 仮想ユーザ関数はブラウザまたはツールキットのセッション中に記録されます。

トランザクション，ランデブー，コメント，ログ関数などの一般的な仮想ユーザ関数は，記録中に手作業で追加できます。詳細については，第 6 章 「仮想ユーザ・スクリプトの拡張」を参照してください。

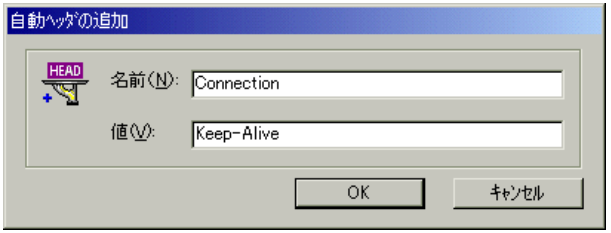
この項では，記録中と記録後に Web 仮想ユーザ関数を追加，編集する方法を，ツリー・ビューとスクリプト・ビューのそれぞれについて説明します。

仮想ユーザ・スクリプトに新しい関数を追加するには，次の手順で行います。

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



2 使用する関数を選択して，［**OK**］をクリックします。ほとんどの Web 仮想ユーザ関数は，［**サービス**］カテゴリの下にあります。選んだ関数の［プロパティ］ダイアログ・ボックスが開きます。関数の［プロパティ］ダイアログ・ボックスでは，関数の引数を指定できます。



3 プロパティを指定して［**OK**］をクリックします。関数が引数と一緒にカーソルの位置に挿入されます。

［プロパティ］ダイアログ・ボックスを開いて，引数の値を変更することによって既存のステップを編集できます。これは，ツリー・ビューをサポートするプロトコルの場合にのみ有効です（WAP には使用できません）。

既存のステップを編集するには，次の手順で行います。

1 右クリック・メニューから［**プロパティ**］を選択します。選んだ関数の［プロパティ］ダイアログ・ボックスが開きます。

2 必要に応じて引数の値を変更し，［**OK**］をクリックします。

Web 関数リスト

インターネットを介した通信を行う Web 仮想ユーザ関数の名前には **web** という接頭辞 が付きます。Web 関数は以下のように分類されます。

- アクション関数
- 認証関数
- キャッシュ関数
- チェック関数
- 接続定義関数
- 同時実行グループ関数
- クッキー関数
- 相関関数
- フィルタ関数
- ヘッダー関数
- プロキシ・サーバ関数
- その他の関数

アクション関数

Web 仮想ユーザ・スクリプトを記録すると，VuGen は次のアクション関数を生成してスクリプトに挿入します。

web_custom_request	HTTP によってサポートされている任意のメソッドで，ユーザ定義の HTTP 要求を作成できます。
web_image	指定された画像に対するマウス・クリックをエミュレートします。
web_link	指定されたテキスト・リンクに対するマウス・クリックをエミュレートします。
web_submit_data	「無条件の」（つまり「コンテキストに依存しない」）フォーム送信を実行します。

web_submit_form

フォームの送信をエミュレートします。

web_url

「URL」属性で指定される URL をロードします。

認証関数

web_set_certificate

この関数を使うと，仮想ユーザによって Internet Explorer のレジストリにリストされている所定の証明書が使用されます。

web_set_certificate_ex

証明書および鍵ファイルの場所と形式の情報を指定します。

web_set_user

Web サーバ（Web サーバのユーザ認証エリア）に，ログイン文字列とパスワードを指定します。

キャッシュ関数

web_cache_cleanup

キャッシュ・シミュレータの内容をクリアします。

チェック関数

web_find

HTML ページの中で，指定されたテキスト文字列を検索します。

web_global_verification

以降の HTTP 要求の中で，指定されたテキスト文字列を検索します。

web_image_check

指定された画像が HTML ページ内に存在することを確認します。

web_reg_find

以降の HTTP 要求を対象とする，HTML ソースまたは未処理のバッファ内のテキスト文字列の検索を登録します。

接続定義関数

web_disable_keep_alive	HTTP のキープ・アライブ接続を無効にします。
web_enable_keep_alive	HTTP のキープ・アライブ接続を有効にします。
web_set_connections_limit	スクリプトの実行中に 1 つの仮想ユーザが同時に開くことができる最大接続数を設定します。

同時実行グループ関数

web_concurrent_end	同時実行グループの終了を示します。
web_concurrent_start	同時実行グループの開始を示します。

クッキー関数

web_add_cookie	新しいクッキーを追加するか，既存のクッキーを変更します。
web_cleanup_cookies	仮想ユーザによって現在格納されているすべてのクッキーを削除します。
web_remove_cookie	指定されたクッキーを削除します。

相関関数

web_create_html_param	HTML ページの動的な情報をパラメータに保存します（LR 6.5 以前）。
web_create_html_param_ex	HTML ページの動的な情報に基づいてパラメータを作成します。埋め込まれている境界を使用します（LR 6.5 以前）。
web_reg_save_param	HTML ページの動的な情報に基づいてパラメータを作成します。埋め込まれている境界は使用しません。
web_set_max_html_param_len	取得される動的な HTML 情報の最大長を設定します。

フィルタ関数

web_add_filter	ダウンロード時に URL を含めたり，除外したりするための基準を設定します。
web_add_auto_filter	ダウンロード時に URL を含めたり，除外したりするための基準を設定します。
web_remove_auto_filter	ダウンロードするコンテンツのフィルタリングを無効にします。

ヘッダー関数

web_add_auto_header	以降の HTTP 要求すべてにユーザ定義のヘッダーを追加します。
web_add_header	次の HTTP 要求にユーザ定義のヘッダーを追加します。
web_cleanup_auto_headers	以降の HTTP 要求へのユーザ定義のヘッダーの追加を中止します。
web_remove_auto_header	以降の HTTP 要求への特定のヘッダーの追加を中止します。
web_revert_auto_header	以降の HTTP 要求への特定のヘッダーの追加を中止しますが，黙示的なヘッダーを生成します。
web_save_header	要求と応答のヘッダーを変数に保存します。

プロキシ・サーバ関数

web_set_proxy	以降のすべての HTTP 要求を指定のプロキシ・サーバに送るようにします。
web_set_proxy_bypass	仮想ユーザが，指定のプロキシ・サーバ経由ではなく，直接アクセスするサーバのリストを指定します。
web_set_proxy_bypass_local	仮想ユーザがローカル（イントラネット）アドレスにアクセスする際，プロキシをバイパスするかどうかを指定します。
web_set_secure_proxy	以降のすべての HTTP 要求が指定されたプロキシ・サーバに送られるようにします。

再生関数

web_set_max_retries	アクション・ステップの再試行回数の上限を指定します。
web_set_timeout	1 つの仮想ユーザを，指定のタスクを実行させるまで待機させる時間の上限を指定します。

その他の関数

web_convert_param	HTML パラメータを URL またはプレーン・テキストに変換します。
web_get_int_property	以前の HTTP 要求に関する特定の情報を返します。
web_report_data_point	データ・ポイントを指定し，テスト結果に追加します。
web_set_option	エンコーディング，リダイレクション，非 HTML リソースのダウンロードに関する Web オプションを設定します。
web_set_sockets_option	ソケットのオプションを設定します。

制御型関数

Web 仮想ユーザ関数に加え，以下の制御関数が仮想ユーザ・スクリプトに表示される場合があります。

lr_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を示します。
lr_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を示します。
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。
lr_think_time	仮想ユーザ・スクリプトのコマンド間で実行を一時停止します。

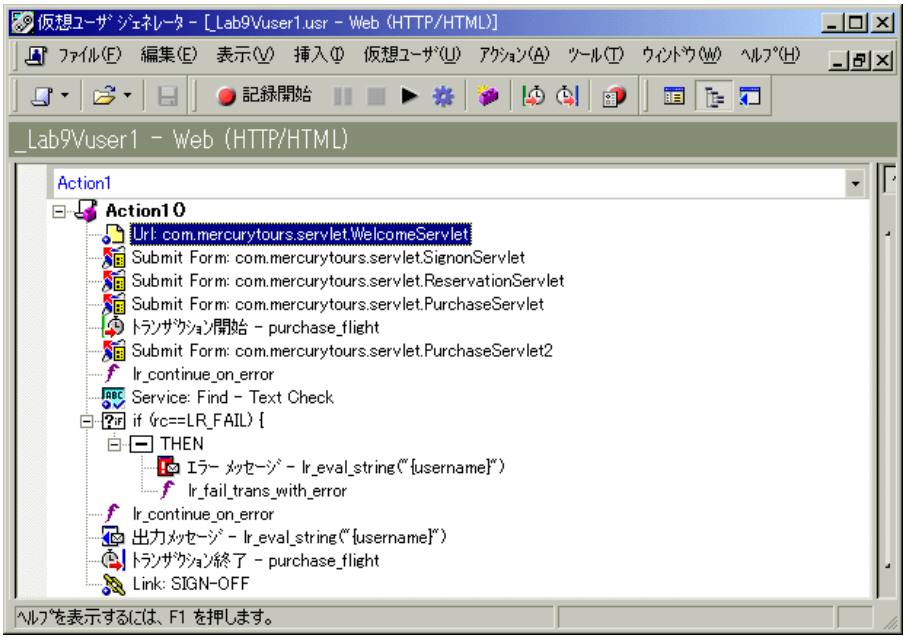
一般仮想ユーザ関数をスクリプトに追加する方法の詳細については，第 6 章「仮想ユーザ・スクリプトの拡張」を参照してください。

ツリー・ビューでのスクリプトの表示

ツリー・ビューでは，アイコン形式のビューに仮想ユーザ・スクリプトが表示されます。各仮想ユーザ関数はアイコンによって表されます。このビューは WAP 仮想ユーザでは使用できません。

Web 仮想ユーザ・スクリプトをツリー・ビューで表示するには，次の手順で行います。

➤ VuGen のメインメニューで［**表示**］＞［**ツリー ビュー**］を選択するか，［**スクリプトをツリー形式で表示**］ボタンをクリックします。仮想ユーザ・スクリプトが，アイコン形式のツリー・ビューに表示されます。すでにツリー・ビューが表示されている場合，このメニュー項目は無効になります。



仮想ユーザ・スクリプトのツリー・ビューは，仮想ユーザのアクションを表すアイコンと仮想ユーザ・スクリプトのステップで構成されています。ステップの種類は，ステップ名の前に付く文字列で示されます。以下に例を示します。

以下のステップの種類が VuGen でサポートされています。

アイコンの種類	説明
サービス	**サービス**・ステップは，Web アプリケーション・コンテキストをまったく変更しないステップを表します。サービス・ステップはコンテキストを変更するのではなく，各種プロキシの設定，認証情報の送信，ユーザ定義のヘッダーの発行などのカスタマイズ作業を実行します。
URL	［**URL**］アイコンは，ブックマークを使用したり，URL を入力したりして特定の Web ページにアクセスすると仮想ユーザ・スクリプトに追加されます。各［URL］アイコンは，仮想ユーザ・スクリプト内の **web_url 関数**を表します。ターゲット・ページの URL の最後の部分が，［URL］アイコンの標準ラベルとなります。
リンク	記録中にハイパーテキスト・リンクをクリックすると，［**リンク**］アイコンが追加されます。各［リンク］アイコンは，仮想ユーザ・スクリプトの **web_link** 関数を表します。ハイパーテキスト・リンクのテキスト文字列が，このアイコンの標準のラベルとなります。
画像	記録中にハイパーグラフィック・リンクをクリックすると，［**画像**］アイコンが仮想ユーザ・スクリプトに追加されます。各［**画像**］アイコンは，仮想ユーザ・スクリプトの **web_image** 関数を表します。HTML コードの画像に ALT 属性がある場合，アイコンの標準のラベルとしてこの属性の値が使用されます。HTML コードの画像に ALT 属性がない場合は，アイコンのラベルとして SRC 属性の最後の部分が使用されます。
フォームを送信 / データを送信	記録中にフォームを送信すると，［**フォームを送信**］ステップまたは［**データを送信**］ステップが追加されます。フォームの処理に使用される実行プログラムの名前が，ステップの標準のラベルとなります。
ユーザ定義要求	VuGen で標準的なアクション（URL，リンク，画像，フォームの送信など）として認識されないアクションを記録する場合は，［**カスタム要求**］アイコンが仮想ユーザ・スクリプトに追加されます。非標準 HTTP アプリケーションに適用されます。

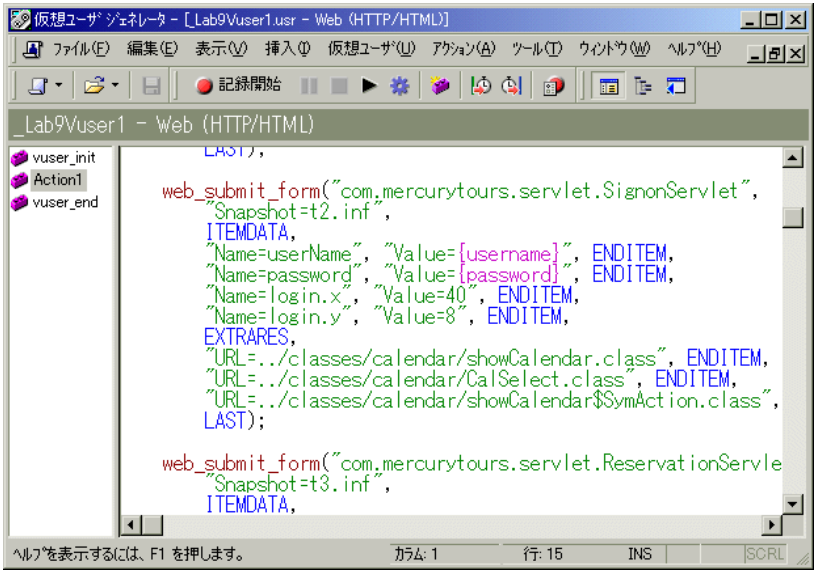
注：HTML（コンテキストセンシティブ）モードで記録するオプションを選択したときのみ，［リンク］，［画像］，［フォーム送信］ステップが記録されます。詳細については，第 36 章 「Web 仮想ユーザの記録オプションの設定」を参照してください。

スクリプト・ビューでの仮想ユーザ・スクリプトの表示

Web 仮想ユーザ・スクリプトをテキスト形式で表示したり，編集したりするには，スクリプト・ビューを選択します。

Web 仮想ユーザ・スクリプトのスクリプト・ビューを表示するには，次の手順で行います。

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。仮想ユーザ・スクリプトが，テキスト形式のスクリプト・ビューに表示されます。すでにスクリプト・ビューが表示されている場合，このメニュー項目は無効になります。



スクリプト・ビューでは，ブラウザ・アプリケーションによって生成された関数を確認できるほか，必要に応じてスクリプトを変更できます。

注：スクリプト・ビュー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が認識されなかった場合には，変更されたスクリプト・ビューのツリー・ビューへの変換はできません。

第 34 章

Web/WinSock および SOAP 仮想ユーザの記録

VuGen では，Web および Windows Sockets にアクセスするアプリケーションをエミュレートする Web/WinSock デュアル・プロトコル仮想ユーザ・スクリプトを作成できます。このプロトコルの一般的な使用例に Palm HotSync プロセスがあります。

本章では，次の項目について説明します。

- Web/WinSock 仮想ユーザ・スクリプトでの作業の開始
- ブラウザとプロキシ記録オプションの設定
- ［Web トラップ］記録オプションの設定
- Web/WinSock セッションの記録
- Palm アプリケーションの記録

以降の情報は Web/Winsocket Dual Protocol，SOAP および Palm 仮想ユーザ・スクリプトを対象とします。

Web/WinSock 仮想ユーザ・スクリプトの記録について

VuGen の Web/WinSock デュアル・プロトコル・タイプでは，HTML 以外の Web アプリケーションを記録できます。VuGen では，Web プロトコル関数と Windows Sockets プロトコル関数の両方を使用してこれらのアプリケーションを記録し，Web ページへのアクセスとソケット操作をエミュレートするスクリプトを作成します。このプロトコルを使用する一般的なアプリケーションに，Palm OS プロトコルを使用したハンドヘルド機器の HotSync プロセスの記録が挙げられます。VuGen によって，データ転送が記録され，関連する関数が生成されます。Palm のワイヤレス・データ転送は記録されません。

デュアル・プロトコル・スクリプトを実行すると，仮想ユーザによって Web ブラウザ，非 HTML アプリケーションおよび Web サーバ間の処理がエミュレートされます。デュアル・プロトコル機能を使えば，Web プロトコルと WinSock プロトコルの両方を一度に記録できるため，重複する呼び出しを回避できます。VuGen で 2 つのプロトコルの記録が同期化され，Web と WinSock 仮想ユーザ関数の両方を含んだ 1 つのスクリプトが作成されます。

可能であれば，Web と WinSock プロトコルを指定し，マルチ・プロトコル・スクリプトを使って，Web と WinSock セッションを記録します。ただし，マルチ・プロトコル・モードでは UDP ソケットはサポートされていないため，UDP ソケットを記録する必要がある場合は，本章で説明する Web/WinScok デュアル・プロトコル仮想ユーザを使用します。

WinSock 関数は，記録セッション中のソケット操作を低レベルのコードで表します。WinSock 関数は，**lrs** という接頭辞が付いており，ソケット，データ・バッファ，環境に関係する処理を行います。セッション中に送受信された実際のデータを表示するには，VuGen の左側の表示枠の **data.ws** を選択します。UDP タイプのソケットの記録は，このモードではサポートされていません。

Web 関数は，**web** という接頭辞が付いています。これらの関数は，URL への移動（**web_url**），データの送信（**web_submit_data**），クッキーの追加（**web_add_cookie**）など，標準的な Web 操作に関する処理を行います。

WinSock 関数および Web 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。

デュアル・プロトコル・スクリプトを記録した後，スクリプトを編集するには，スクリプト・ビューでスクリプトのテキストを修正します。標準の Web 仮想ユーザ・スクリプトで使用できるツリー・ビューとスナップショット・ウィンドウは，Web/WinSock スクリプトではサポートされていません。

Web/WinSock 仮想ユーザ・スクリプトの値は，シングル・プロトコル・スクリプトの場合と同じように相関させます。ただし，Web 関数と WinSock 関数では，相関手順が異なります。Web 関数の相関の詳細については，第 41 章「Web 仮想ユーザ・スクリプトの相関ルールの設定」，WinSock 関数の相関の詳細については，第 8 章 「ステートメントの相関」を参照してください。

Web/WinSock 仮想ユーザ・スクリプトでの作業の開始

この項では，VuGen を使用したデュアル・プロトコルの Web/WinSock 仮想ユーザ・スクリプトの開発工程の概略を説明します。

Web/WinSock 仮想ユーザ・スクリプトの作成は，次の手順で行います。

1 **VuGen を使用して，基本のスクリプトを記録します。**

VuGen を起動して，新しい仮想ユーザ・スクリプトを作成します。仮想ユーザのタイプとして「Web/Winsocket Dual Protocol」を指定します。記録対象のアプリケーションを選び，Web および WinSock の記録オプションを設定します。アプリケーションを使用した標準的な操作を記録します。

詳細については，450 ページ「ブラウザとプロキシ記録オプションの設定」を参照してください。

2 **スクリプトを拡張します。**

トランザクション，ランデブー・ポイント，および制御フロー構造をスクリプトに挿入することによって，仮想ユーザ・スクリプトを拡張します。

詳細については，第 6 章 「仮想ユーザ・スクリプトの拡張」を参照してください。

3 **パラメータを定義します（任意）。**

スクリプトに記録された固定値の代わりにパラメータを定義します。固定値をパラメータで置き換えると，同じビジネス・プロセスを異なる値で繰り返し実行できます。

詳細については，第 7 章 「パラメータの定義」を参照してください。

4 **ステートメントを相関させます（任意）。**

ステートメントを相関することにより，あるビジネス・プロセスの結果を後続のビジネス・プロセスで使用できます。

詳細については，第 8 章「ステートメントの相関」または第 41 章 「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。

5 **実行環境を設定します。**

実行環境を設定することによって，スクリプト実行時の仮想ユーザの動作を制御します。設定には，反復，ログ，タイミング情報などがあります。

詳細については，第 9 章 「実行環境の設定」を参照してください。

6 **VuGen からスクリプトを実行します。**

VuGen でスクリプトを保存して実行し，正しく動作することを確認します。

詳細については，第 11 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。

仮想ユーザ・スクリプトを作成した後，スクリプトをシナリオに統合します。詳細については，『**LoadRunner コントローラ・ユーザーズ・ガイド**』を参照してください。

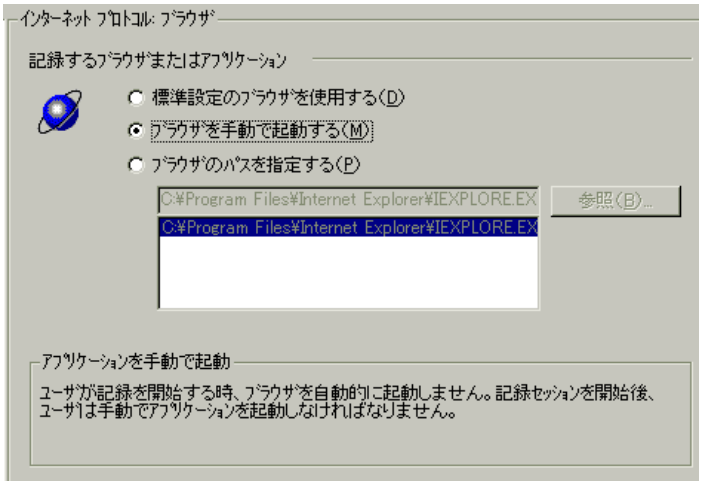
ブラウザとプロキシ記録オプションの設定

スクリプトを記録する前に，Web と WinSock の記録オプションを設定します。Web の記録オプションとして，［ブラウザ］，［記録用プロキシ］，［記録］，［相関］について設定します。WinSock の記録オプションとしては，ソケットの除外，思考遅延時間のしきい値の設定，変換テーブルの指定を行います。この項では，［ブラウザ］および［記録用プロキシ］記録オプションについて説明します。その他のインターネット・プロトコルの記録オプションの詳細については，第 35 章 「インターネット・プロトコルの記録オプションの設定」，WinSock の記録オプションについては，第 21 章 「WinSock 仮想ユーザ・スクリプトの作成」を参照してください。

記録オプションを表示するには，［**ツール**］＞［**記録オプション**］を選択するか，［記録開始］ダイアログ・ボックスの［**オプション**］ボタンをクリックします。

［ブラウザ］記録オプションの設定

［ブラウザ］記録オプションを使って，仮想ユーザ・スクリプトの記録時に VuGen が使うブラウザを指定できます。



［**ブラウザ**］ノードで次の 3 つの選択肢から 1 つを選択します。これらのオプションは，Web トラップを無効にしている場合にだけ有効です（454 ページ「［Web トラップ］記録オプションの設定」を参照）。Web トラップを有効にすると，［**記録するアプリケーション**］フィールドのアプリケーションが必ず起動されます。

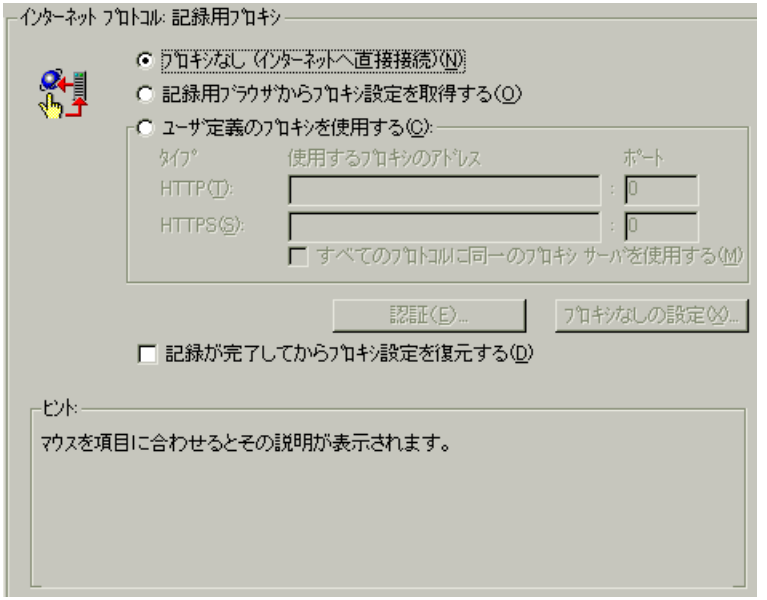
- ［**標準設定のブラウザを使用する**］：記録を行うコンピュータで標準設定の Web ブラウザを使用するよう VuGen に指示する場合に使います。［記録開始］ダイアログ・ボックスの［**記録するアプリケーション**］フィールドのアプリケーションは無視されます。ただし，このフィールドには使用しなくても値を入力する必要があります。ActiveX アプリケーションまたは Java テンプレートを記録するには，このオプションを使用します。
- ［**ブラウザを手動で起動する**］：記録開始時にアプリケーション（この場合，ブラウザ）を自動的に起動しないよう VuGen に指示します。［記録開始］ダイアログ・ボックスの［**記録するアプリケーション**］フィールドにブラウザのパスを指定すると，VuGen によって記録開始時にそのブラウザが起動され，プロキシ設定の変更を求められます（452 ページを参照）。このオプションは，スタンドアロンのアプリケーションか，ブラウザを起動するアプリケーションに使用します。

➤［**ブラウザのパスを指定する**］：特定のアプリケーションを自動的に起動するよう VuGen に指示します。リストからアプリケーションとそのパスを選択するか，［**参照**］ボタンをクリックして，使用するアプリケーションを探します。［記録開始］ダイアログ・ボックスの［**記録するアプリケーション**］フィールドのアプリケーションは無視されます。ただし，このフィールドには使用しなくても値を入力する必要があります。このオプションは，非ブラウザ・アプリケーションや標準設定以外のブラウザを使用する場合に使用します。

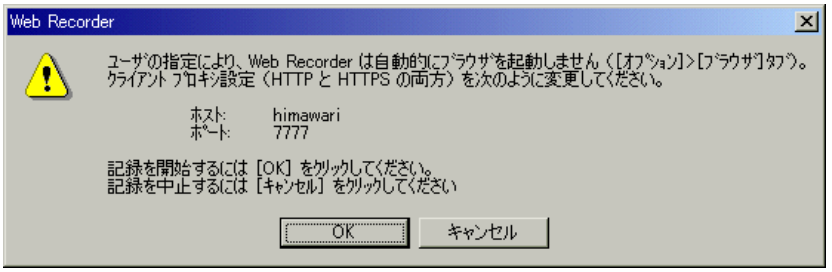
記録用プロキシの設定

ブラウザを手作業で起動するための記録オプションを設定し（前の項を参照してください），Web トラップを有効にしない場合，プロキシ設定を変更しなければならないことがあります。ブラウザを自動的に起動しないため，記録用ブラウザからプロキシ設定を取得するように VuGen に指定できません。

代わりに，［**プロキシなし（インターネットへ直接接続）**］オプションを選択します。



記録開始後，ブラウザのプロキシ設定の変更を促すメッセージと，使用すべき設定を示すメッセージが VuGen によって発行されます。



ブラウザの設定を変更せずに［**OK**］をクリックすると，VuGen によってアプリケーションだけが記録され，ブラウザ・アクションは記録されません。プロキシの設定を行うには，記録を中止し，ブラウザの設定を行います。

プロキシの設定を変更するには，次の手順で行います。

- Netscape の場合は，［**編集**］＞［**設定**］＞［**詳細**］＞［**プロキシ**］＞［**手動でプロキシを設定する**］を選択し，ホスト名として **localhost**（小文字）と，上記のダイアログ・ボックスに示されているポート番号を入力します。
- Internet Explorer の場合は，［**ツール**］＞［**インターネット オプション**］＞［**接続**］＞［**LAN の設定**］を選び，［**プロキシ サーバーを使用する**］を選択します。ホスト名として **localhost**（小文字）と，上記のダイアログ・ボックスに示されているポート番号を入力します。

その他の Web 記録オプションについては，第 36 章 「Web 仮想ユーザの記録オプションの設定」を参照してください。WinSock 記録オプションについては，第 21 章 「WinSock 仮想ユーザ・スクリプトの作成」を参照してください。

［Web トラップ］記録オプションの設定

VuGen で Web/WinSock 仮想ユーザのスクリプトを記録するとき，VuGen によってブラウザのプロキシ設定が変更されます。VuGen によって，すべての HTTP および HTTPS 要求が，再設定されたプロキシ・ポートを通るようになります。プロキシ・ポートを通った Web 要求は，［**記録用プロキシ**］タブで指定したポートを通されます。指定されたプロキシ・ポートを使って送受信されないすべての要求は WinSock 関数として記録され，HTTP Web 要求としては記録されません。記録後，プロキシ設定は元どおりに復元されます。

特定の Java アプレットなど，アプリケーションによっては，Web イベントは発行してもプロキシ設定をサポートしないものがあります。VuGen ではこれらのアプリケーションに対しては必要な内部プロキシの設定が行えません。その結果，これらのアプリケーションのアクションは Web イベントではなく，WinSock の要求として記録されるため，スクリプトが読みにくく，直観的に理解しにくいものとなります。アプリケーションと起動処理の記録方法については，451 ページ「［ブラウザ］記録オプションの設定」を参照してください。

［Web トラップ］の設定では，通常は WinSock 関数として記録されるイベントを，Web 関数としてトラップまたは保存することができます。トラップのオプションを有効にすると，VuGen によって指定のポートでイベントが待機され，それらのイベントを Web イベントと見なし，適切な Web 関数が生成されます。この結果，読みやすく直観的に理解しやすいスクリプトを作成できます。

VuGen が Web イベントをリッスンするポートを指定する必要があります。そのポート上で行われるすべてのやり取りが，Web イベントとして処理され，Web 仮想ユーザ関数で表されます。標準ポート（HTTP の場合は 80，HTTPS の場合は 443）を使用するか，任意の IP とポートの組み合わせ（IP: ポート）を指定することができます。VuGen では，ワイルドカードを組み合わせて使用できるので，特定のホストのすべてのポートを指定することもできます。

たとえば，207.232.15.30:* は，ホスト・マシン 207.232.15.30 上のすべてのポートを示します。207.232.*.*:80 と指定した場合は，ドメイン 207.232 のすべてのマシン上の標準ポート 80 を示します。IP アドレスの 1 つの要素の中で数字とワイルドカードを混在させることはできません。たとえば，207.2*.32.9 という指定は無効です。

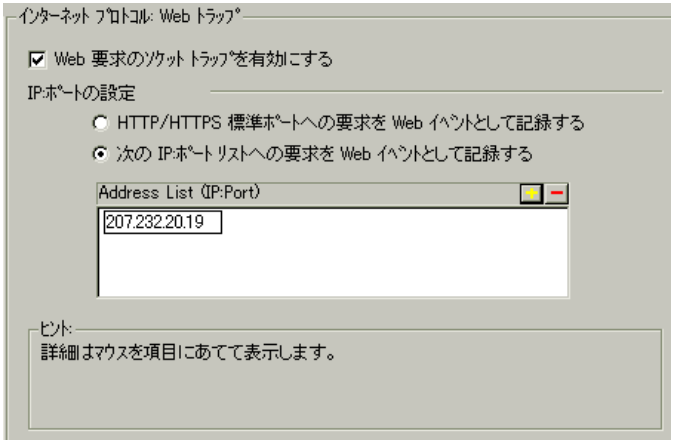
Web トラップを有効にするかどうかを決めるために，まず記録セッションを実行します。データ・ファイル **data.ws** を表示します。WinSock バッファ・データとして記録された HTTP または HTTPS データがある場合，これは別のポートを通じて要求を受け取ったことを示していることがあります。この場合，それらの要求について Web 関数が生成されるように，Web トラップを有効にする必要があります。

このオプションは，ブラウザを使って記録するのではなく，記録するアプリケーションを手作業で起動するときに特に役立ちます。アプリケーションを手作業で起動する方法については，451 ページ「[ブラウザ］記録オプションの設定」を参照してください。

Web トラップを有効にすると，指定されたポートで行われる Windows Sockets 通信はすべて無視されます。

［Web トラップ］記録オプションを設定するには，次の手順で行います。

1 ［**ツール**］＞［**記録オプション**］を選び，［Web トラップ］ノードを選択します。



2 Web イベントのトラップを有効にするには，［**Web 要求のソケット トラップを有効にする**］チェック・ボックスを選択します。

3 標準のポートで Web イベントをトラップするには，［**HTTP/HTTPS 標準ポートへの要求を Web イベントとして記録する**］を選択します。

4 標準以外のポートの Web イベントをトラップするには，［**次の IP: ポート リストへの要求を Web イベントとして記録する**］を選択します。新しい IP ポート・エントリをリストに追加するには，「+」をクリックします。既存のエントリを削除するには，「－」をクリックします。前の節で説明したように，ワイルドカードを使用できます。

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。

Web/WinSock セッションの記録

デュアル・プロトコル・セッションは，スタンドアロンの Web と Windows Sockets 仮想ユーザの記録と同じように記録します。デュアル・プロトコル・セッションを記録すると，VuGen によって Web ブラウザまたはアプリケーション内で実行したすべてのアクションが監視され，適切な Web 関数または WinSock 関数が生成されます。

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**，**Actions**，**vuser_end** の 3 つです。記録中に，VuGen が記録する関数を挿入するスクリプトのセクションを選択できます。**vuser_init** と **vuser_end** セクションは通常，複数の反復を持つスクリプトを実行するときに繰り返して実行しないサーバ・ログインとログオフの手順の記録に使用します。したがって，ブラウザ・セッションが毎回完全な形で反復されるように，**Actions** セクションに記録しなければなりません。

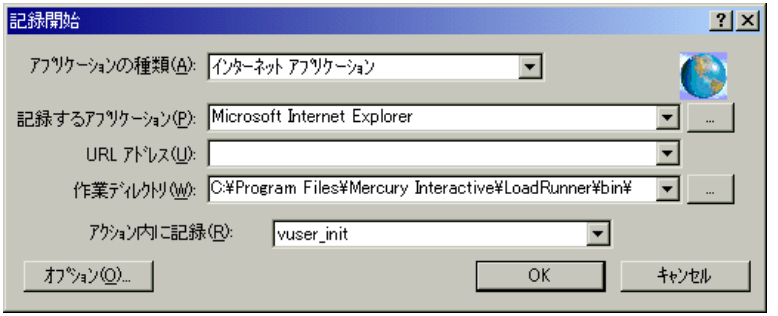
Web/WinSock セッションを記録するには，次の手順で行います。

1 記録用のブラウザを開き，ホームページを記録対象 URL に設定します。

2 ［**スタート**］＞［**プログラム**］＞［**LoadRunner**］＞［**Virtual User Generator**］を選択します。VuGen のメイン・ウィンドウが開きます。

3 新しい Web/WinSock スクリプトを作成します。［**ファイル**］＞［**新規作成**］を選択するか，［**新規作成**］ボタンをクリックします。

4 ［**e ビジネス**］フォルダから［**Web/Winsocket デュアル プロトコル**］を選択し，［**OK**］をクリックします。VuGen によって仮想ユーザ・スクリプトのスケルトンが開き，［記録開始］ダイアログ・ボックスが表示されます。



5 ［**オプション**］をクリックし，ソケット，ブラウザ，プロキシ，その他の詳細設定の記録オプションを設定します。ブラウザを使って記録している場合は，ブラウザを指定します。ブラウザ以外のアプリケーション（ストリーミング・データなど）を記録している場合は，［**ブラウザを手動で起動する**］ように［ブラウザ］記録オプションを設定します。手動で起動する場合，プロキシ・オプションを［**プロキシなし（インターネットへ直接接続）**］に設定し，ブラウザのプロキシ設定を **localhost** に変更します。これらの記録オプションの設定の詳細については，451 ページ「［ブラウザ］記録オプションの設定」を参照してください。

6 ［**参照**］をクリックして，記録するアプリケーションを選択します。このエントリは，記録オプション（［**ブラウザ**］ノード）で［**ブラウザを手動で起動する**］を指定した場合にのみ使用されます。［**記録するアプリケーション**］ボックスに，ブラウザ以外のアプリケーションのパスと名前を指定します。ブラウザを使って記録する場合，この入力項目は無視されますが，このボックスには値を入力しなければなりません。

7 ［**アクション内に記録**］リストから，記録を開始するセクションを選択します。

8 ［**OK**］をクリックすると，アプリケーションが起動し，記録が開始されます。フローティング記録ツールバーが表示されます。

注：Web 仮想ユーザ・スクリプトを記録するときに実行できる Netscape Navigator のインスタンスは 1 つだけです。したがって，記録を開始する前に Netscape Navigator が起動されていると，そのブラウザを閉じるように求められます。ブラウザを閉じて，VuGen によって Netscape ブラウザが起動されるようにします。

---

9 記録したいビジネス・プロセスを実行します。リンクをクリックするたびに，**web_url** 関数がスクリプトに追加されます。フォームを送信するたびに，**web_submit_form** 関数が仮想ユーザ・スクリプトに追加されます。ブラウザ機能を使わないアプリケーションのアクションはソケット・データとして記録されます。

記録中，VuGen のフローティング・ツールバーを使って，トランザクション，ランデブー・ポイント，インスタント・テキスト検査を挿入できます。詳細については，下記を参照してください。テキストまたは画像チェックの挿入の詳細については，第 39 章 「負荷下の Web ページ検証」を参照してください。

10 必要なすべてのユーザ・プロセスを実行したら，フローティング・ツールバーの［**記録停止**］ボタンをクリックします。VuGen のメイン・ウィンドウに戻ります。

11 ［**ファイル**］＞［**保存**］を選択するか，［**上書き保存**］ボタンをクリックして仮想ユーザ・スクリプトを保存します。［テストを保存］ダイアログ・ボックスでファイルの名前と場所を指定して，［**保存**］をクリックします。

記録の後で，トランザクション，ランデブー・ポイント，および制御フロー構造をスクリプトに挿入することによって仮想ユーザ・スクリプトを編集できます。詳細については，第 6 章 「仮想ユーザ・スクリプトの拡張」を参照してください。

スクリプトに変更を加えた後で，スクリプトを記録時の状態に戻す必要が生じた場合には，［**仮想ユーザの再生成**］ユーティリティを使用します。このユーティリティは，WinSock ステートメントのみを再生成します。つまり，Web ステートメントには影響を及ぼしません。詳細については，46 ページ「仮想ユーザ・スクリプトの再生成」を参照してください。

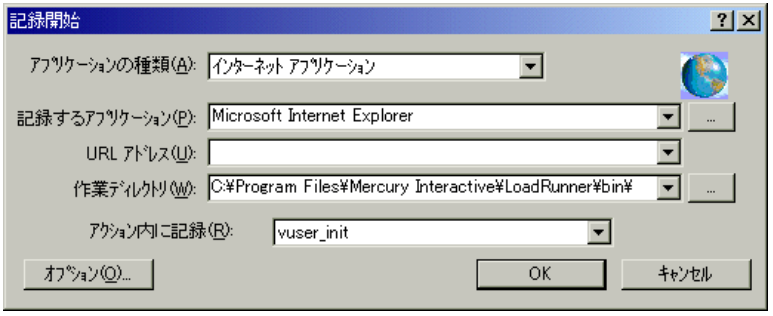
Palm アプリケーションの記録

Palm ベースのアプリケーションがリモート・サーバと通信する手段は，クレードルとワイヤレスの 2 つがあります。クレードルにドッキングされた Palm アプリケーションは，HotSync サービスを使用してインターネット経由で直接サーバと通信します。VuGen では，Palm の HotSync サービスによるすべてのトラフィックをキャプチャできます。多くのアプリケーションではサーバと通信する際に HTTP がトランスポート・レイヤとして使用されるため，生成されるスクリプトは Web スクリプトに似たものになり，Web スクリプトの構文と機能が使用されます。まれに，このトラフィックで独自のプロトコルが使用されることがあります。この独自プロトコルのトラフィックも，スクリプトの中で WinSock 関数として表され，記録されます。

Palm アプリケーションを記録するには，次の手順で行います。

1 新しいスクリプトを作成します。［**ファイル**］＞［**新規作成**］を選択するか，［**新規作成**］ボタンをクリックします。

2 ［**e ビジネス**］フォルダから［**Palm**］を選択し，［**OK**］をクリックします。VuGen によって仮想ユーザ・スクリプトのスケルトンが開き，［記録開始］ダイアログ・ボックスが表示されます。



3 記録するアプリケーションとして HotSync.Exe を指定し，［**OK**］をクリックします。

HotSync.Exe を VuGen から起動する前に，すでに実行されていないことを確認してください。

4 クレードルに Palm Pilot をセットし，アプリケーションと通信します。

Palm Pilot とサーバの間の通信を開始するには，Palm Pilot の［**HotSync**］ボタンを押す必要がある場合があります。

5 必要なすべてのユーザ・プロセスを実行したら，フローティング・ツールバーの［**記録停止**］ボタンをクリックします。VuGen のメイン・ウィンドウに戻ります。

6 ［**ファイル**］＞［**保存**］を選択するか，［**上書き保存**］ボタンをクリックして仮想ユーザ・スクリプトを保存します。［テストを保存］ダイアログ・ボックスでファイルの名前と場所を指定して，［**保存**］をクリックします。

記録の後で，トランザクション，ランデブー・ポイント，および制御フロー構造をスクリプトに挿入することによって仮想ユーザ・スクリプトを編集できます。詳細については，第 6 章 「仮想ユーザ・スクリプトの拡張」を参照してください。

スクリプトは，Web および WinSock プロトコルの組み合わせとして表されます。HTTP に埋め込まれたすべての Palm トラフィックは，**web_url** ステートメントおよび **web_submit_data** 要求として表されます。独自プロトコルの操作は，WinSock 関数への呼び出しによって表されます。

第 35 章

インターネット・プロトコルの記録オプションの設定

インターネット上で動作するプロトコルについて，インターネット関連の記録オプションをカスタマイズできます。

本章では，以下の項目について説明します。

- プロキシ設定を使った作業
- 記録オプションの詳細設定
- 記録スキーマの設定

以降の情報は，Web，ワイヤレス，および Oracle NCA プロトコルに適用されます。

インターネット・プロトコルの記録オプションの設定について

VuGen では，現実を忠実に再現するインターネット環境をエミュレートする仮想ユーザ・スクリプトを作成できます。

記録を開始する前に，プロキシおよびスクリプト生成にかかわる VuGen の記録オプションを設定できます。

また，Web 仮想ユーザ・スクリプトについては，プロトコル固有の記録オプションも設定できます。詳細については，使用するプロトコルに関する記録オプションの章を参照してください。

［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くことができます。

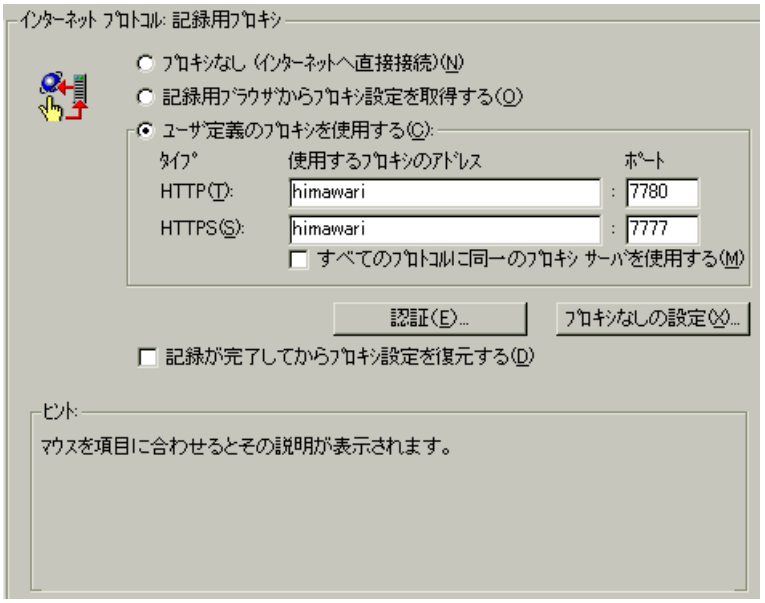
- ツールバー・ボタン： 
- キーボードのショートカット：Ctrl キーを押しながら F7 キーを押します。
- ［ツール］メニュー：［**ツール**］＞［**記録オプション**］を選択します。

プロキシ設定を使った作業

プロキシ・サーバは，クライアント（Web ブラウザなど）と Web サーバの間にあるサーバです。Web サーバに送信されるすべての要求がそこで横取りされ，可能であればそこで要求が満たされます。プロキシ・サーバは主に，パフォーマンスの向上と要求のフィルタリングの 2 つの目的で使用されます。パフォーマンスを向上させるために，ユーザがアクセスした Web ページが格納され，別のユーザが再度サーバにアクセスしなくてもそのページを利用できるようにします。また，管理者はプロキシ・サーバを使って，ブラウザに表示される内容をフィルタリングすることもできます。

プロキシ・サーバを使うには，ブラウザの設定でプロキシ・サーバの名前または IP アドレスを指定します。一般的に，インターネット・サービス・プロバイダはユーザに対して，プロキシ・サーバを介して接続することを勧めています。また企業でも，社員はプロキシ・サーバを介してインターネットにアクセスすることを求められます。

標準では，VuGen では記録用ブラウザのプロキシ設定を使用します。VuGen で記録セッション用に使用するプロキシ設定をカスタマイズすることもできます。アプリケーションのユーザがプロキシ・サーバを介さずにインターネットに直接アクセスすることや，ブラウザの標準設定ではなく特定のプロキシ・サーバを使うことが事前にわかっている場合は，プロキシ設定をカスタマイズできます。設定をカスタマイズするには，［記録オプション］ツリーの［**インターネット プロトコル : 記録用プロキシ**］ノードを選択して，記録プロキシ設定を変更します。

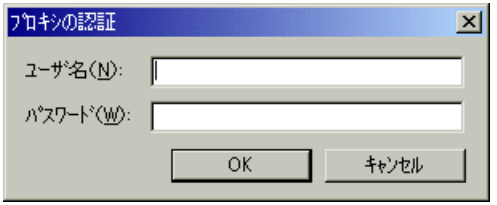


以下のプロキシ・オプションのいずれかを選択できます。

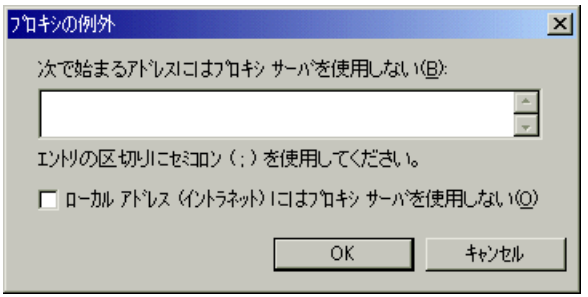
- ［**プロキシなし（インターネットへ直接接続）**］：常にインターネットへの直接接続を利用します。つまり，プロキシ・サーバを使用せずに直接接続します。通常，これは Internet Explorer の［設定を自動的に検出する］設定に対応します。
- ［**記録用ブラウザからプロキシ設定を取得する**］：記録用ブラウザのプロキシ設定を使用します。標準設定では，このオプションが選択されています。このオプションは，Web/WinSock 仮想ユーザに対しては使用できません。
- ［**ユーザ定義のプロキシを使用する**］：記録中，指定されたプロキシ・サーバを使用します。セキュアでない HTTP サイト用と，セキュアな（HTTPS）サイト用に，それぞれ別のプロキシ・サーバを指定できます。このセクションは，上の 2 つのオプションが選択されていないときのみ使用できます。

  HTTP および HTTPS プロキシ・サーバが同じである場合は，HTTP アドレスとポートだけを指定し，［**すべてのプロトコルに同一のプロキシ サーバを使用する**］オプションを選択します。

プロキシ・サーバによっては，ユーザ名とパスワードによる認証が必要なものもあります。認証が必要なプロキシを介してセッションを記録する場合は，［**認証**］ボタンをクリックして，［プロキシの認証］ダイアログ・ボックスで［**ユーザ名**］と［**パスワード**］を入力します。



VuGen から直接（つまり，プロキシ・サーバを使わずに）アクセスしたいホスト名や IP アドレスを指定するには，［**プロキシなしの設定**］ボタンをクリックします。［プロキシの例外］ダイアログ・ボックスが開きます。



VuGen に直接アクセスさせるアドレスを入力します。各アドレスはセミコロンで区切ります。

ローカル（イントラネット）のアドレスにアクセスするときにプロキシ・サーバを使用しないようにするには，［**次で始まるアドレスにはプロキシ サーバを使用しない**］オプションを選択します。

プロキシ設定の復元

記録用にマシンの通常のブラウザ設定と異なるプロキシ設定を指定した場合は，VuGen によって元のブラウザ設定が復元されます。標準では，起動したブラウザの設定を読み取った直後にプロキシ設定が復元されます。記録を停止したときだけ元のプロキシ設定を復元するには，［**記録が完了してからプロキシ設定を復元する**］チェック・ボックスを選択します。このオプションは Internet Explorer にのみ適用されます。

マシンのセキュリティを確保するために，プロキシ設定を直ちに復元することをお勧めします。記録後に設定を復元するためのオプションは最も安全とは言えませんが，後からプロキシ設定を読み取る可能性があるときには必要です。たとえば，アプレット，ActiveX コントロール，およびマルチウィンドウ・アプリケーションの HTTP アクションを記録するときなどです。

記録オプションの詳細設定

［**インターネット プロトコル：詳細**］では，次の設定が可能です。

- ［お気に入り］インターネット記録オプション
- 記録スキーマの設定

［お気に入り］インターネット記録オプション

［お気に入り］オプションを使用して，思考遅延時間，コンテキストのリセット，スナップショットの保存，および **web_reg_find** 関数にかかわるコード生成の方法をカスタマイズできます。オプションのいくつかは，マルチ・プロトコル・モードでは利用できません。

［**思考遅延時間を記録する**］：（シングル・プロトコルの場合のみ）思考遅延時間は，アクション間に実際のユーザが待機する時間をエミュレートします。仮想ユーザが，記録された思考遅延時間をスクリプト実行時にどのように使用するかを指定できます。ユーザの思考遅延時間を記録するには，［**思考遅延時間を記録する**］を選択します。ユーザが最低何秒間待機したら思考遅延時間として記録するかを，［**思考遅延時間のしきい値**］で定義します。たとえば，思考遅延時間のしきい値に「5」を設定した場合，ユーザの待機時間が 5 秒未満の場合は，思考遅延時間は記録されません。このオプションは Web 仮想ユーザおよびワイヤレス仮想ユーザにのみ適用されます。

［**各アクションごとにコンテキストをリセットする**］：（Web，Oracle NCA の場合のみ）この設定は，標準で有効になっており，VuGen に対してアクションごとに HTTP コンテキストをすべてリセットするよう指示します。この設定により，仮想ユーザは新規ユーザがブラウザ・セッションを開始する様子をより正確にエミュレートできます。このオプションは，HTML コンテキストをリセットし，コンテキストを持たない関数が常にアクションの先頭に記録されるようにします。また，このオプションは，キャッシュをクリアし，ユーザ名とパスワードをリセットします。

［**完全トレース記録ログ**］：（シングル・プロトコルの場合のみ）記録中にトレース・ログを作成します。このログはマーキュリー・インタラクティブのカスタマー・サポートが内部的に使用します（標準設定では無効）。

［**スナップショットのリソースをローカルに保存する**］：VuGen によって記録時と再生時にスナップショット・リソースのローカル・コピーが作成されます。この機能によって，より正確なスナップショットが作成され，よりすばやく表示することができます。

［**ページタイトルで web_reg_find 関数を生成する**］：（Web，Oracle NCA の場合のみ）すべての HTML ページのタイトルに対して，**web_reg_find** 関数の生成を有効にします。VuGen によって，ページのタイトル・タグから文字列が取得され，それが **web_reg_find** の引数として使用されます。

記録されたページのすべてのサブフレームにあるページのタイトルに対して，**web_reg_find** 関数の生成を有効にするには，［**サブフレームで web_reg_find 関数を生成する**］を選択します。

［**サポート対象文字セット**］：

［**UTF-8**］：UTF-8 エンコーディングのサポートを有効にします。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 エディタに正しく表示されます。UTF-8 のサポートを有効にした場合，UTF-8 以外の文字セットを使用するサイトは記録できません。

［**EUC-JP**］：日本語版の Windows を利用するユーザは，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する Web サイトに対するサポートを有効にできます。この設定を有効にすると，EUC-JP 文字がマシンのロケールで使用されているロケールに変換され，VuGen エディタ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）

記録スキーマの設定

次の領域で記録スキーマを指定することによって，記録方法をさらにカスタマイズできます。

- ユーザ定義ヘッダーの記録
- 内容タイプのフィルタリング
- リソース以外の内容タイプの指定

ユーザ定義ヘッダーの記録

Web 仮想ユーザでは，サーバに送信されるすべての HTTP 要求とともに，いくつかの標準的な HTTP ヘッダー自動的に送信されます。その他の HTTP ヘッダーを記録するには，［**ヘッダ**］をクリックします。次の 3 つのモードがあります。［**ヘッダは記録しない**］，［**リスト内のヘッダを記録**］，［**リストに定義されていないヘッダを記録**］の 3 つです。最初のモードでは，ヘッダーが記録されません。2 つ目のモードでは，チェック・マークを入れたユーザ定義ヘッダーだけが記録されます。［**リストに定義されていないヘッダを記録**］を指定すると，チェック・マークを入れたヘッダーと，他のリスキー・ヘッダーを除くすべてのユーザ定義ヘッダーが記録されます。

「**リスキー・ヘッダー**」と呼ばれる標準ヘッダーには，「Authorization」，「Connection」，「Content-Length」，「Cookie」，「Host」，「If-Modified-Since」，「Proxy-Authenticate」，「Proxy-Authorization」，「Proxy-Connection」，「Referer」，「WWW-Authenticate」があります。［ヘッダ リスト］で選択しない限り，これらのヘッダーは記録されません。標準設定は，［**ヘッダは記録しない**］です。

［**リスト内のヘッダを記録**］モードでは，チェック・マークを入れたヘッダーが検出され，それらのヘッダーに対してスクリプトに **web_add_auto_header** 関数が挿入されます。これは，明示的な指定のない限り記録されないリスキー・ヘッダーを記録する場合に理想的なモードです。［**リストに定義されていないヘッダを記録**］モードでは，チェック・マークを入れていないヘッダーが記録中に検出され，それらのヘッダーに対してスクリプトに **web_add_auto_header** 関数が挿入されます。

記録するユーザ定義ヘッダーを決めるときに，すべてのヘッダーを記録するように VuGen に指定して記録セッションを実行できます（下記の手順を参照）。その後，記録するヘッダーと除外するヘッダーを決定します。

前出の画面では，［**リスト内のヘッダを記録**］モードで「**Content-Type**」のヘッダーを指定しています。VuGen によってヘッダーが検出され，次のステートメントがスクリプトに追加されます。

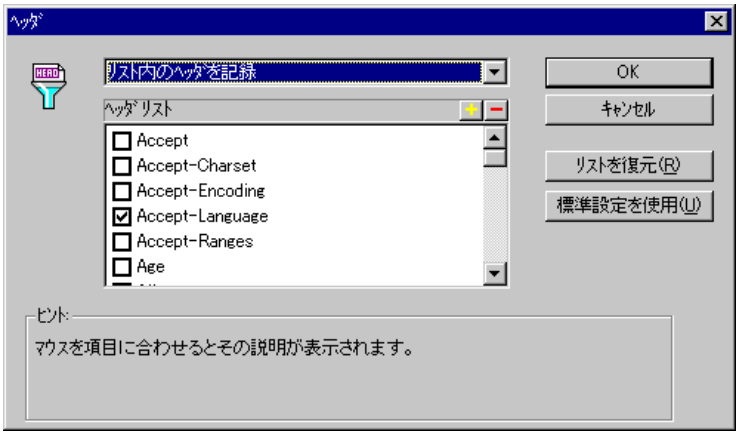
```
web_add_auto_header("Content-Type","application/x-www-form-urlencoded");
```

これはサーバに対して，アプリケーションの「Content-Type」が x-www-form-urlencode であることを示しています。

ユーザ定義ヘッダーの記録を制御するには，次の手順で行います。

1 ［記録オプション］ツリーで，［**インターネット プロトコル：詳細**］ノードを選択します。

2 ［**ヘッダ**］をクリックします。［ヘッダ］ダイアログ・ボックスが開きます。



3 次のいずれかのメソッドを使用します。

- ヘッダーを記録しないように設定するには，［**ヘッダは記録しない**］を選択します。
- 特定のヘッダーだけを記録するには，［**リスト内のヘッダを記録**］を選択し，ヘッダー・リストから必要なユーザ定義のヘッダーを選びます。標準設定では，標準ヘッダー（**Accept** など）が選択されています。
- すべてのヘッダーを記録するには，［**リストに定義されていないヘッダを記録**］を選択し，リストからは項目を選択しません。
- 特定のヘッダーだけを除外するには，［**リストに定義されていないヘッダを記録**］を選択し，除外するヘッダーを選択します。

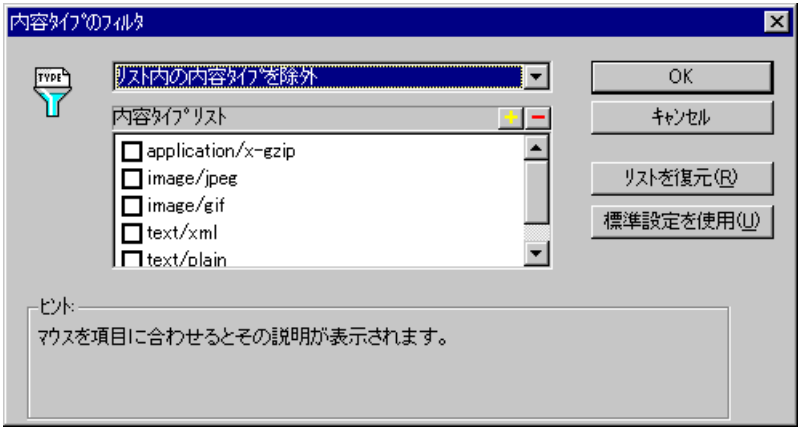
4 リストを対応する標準設定のリストに戻すには，［**リストを復元**］をクリックします。［**リスト内のヘッダを記録**］と［**リストに定義されていないヘッダを記録**］のそれぞれに，対応する標準設定のリストがあります。

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

内容タイプのフィルタリング

VuGen を使って，記録したスクリプトの内容タイプをフィルタリングできます。記録する内容タイプ，またはスクリプトから除外する内容タイプを指定します。次の 3 つのモードがあります。［**内容タイプをフィルタにかけない**］，［**リスト内の内容タイプを除外**］，［**リストに定義されていない内容タイプを除外**］の 3 つです。最初のモードで作業すると，内容タイプがフィルタリングされません。2 つ目のモードでは，選択された内容タイプだけが除外されます。［**リストに定義されていない内容タイプを除外**］を指定すると，チェック・マークを入れた内容タイプ以外のすべての内容タイプが除外されます。標準では，フィルタは設定されていません。

たとえば，Web サイトのテキストと画像だけを対象とするには，［**リストに定義されていない内容タイプを除外**］を選択し，タイプとして「**text/html**」，「**image/gif**」，「**image/jpeg**」を指定します。VuGen によってすべての HTML ページと画像が記録され，サイトに表示される「**text/css**」，「**application/x-javascript**」などのリソースが除外されます。



記録中に内容をフィルタリングするには，次の手順で行います。

1 ［記録オプション］ツリーで，［**インターネット プロトコル：詳細**］ノードを選択します。

2 ［**内容タイプ**］をクリックします。［内容タイプのフィルタ］ダイアログ・ボックスが開きます。

3 次のいずれかのメソッドを使用します。

- 内容をフィルタリングしないように設定するには，［**内容タイプをフィルタにかけない**］を選択します。
- 特定の内容タイプだけを記録対象から除外するには，［**リスト内の内容タイプを除外**］を選び，記録対象から除外する内容タイプをリストから選択します。
- 特定の内容タイプだけを記録するには，［**リストに定義されていない内容タイプを除外**］を選び，記録する内容タイプを選択します。

4 リストを対応する標準設定のリストに戻すには，［**リストを復元**］をクリックします。［**リスト内の内容タイプを除外**］と［**リストに定義されていない内容タイプを除外**］のそれぞれに，対応する標準設定のリストがあります。

5 ［**OK**］をクリックして設定を受け入れ，［内容タイプのフィルタ］ダイアログ・ボックスを閉じます。

リソース以外の内容タイプの指定

スクリプトを記録するときに，VuGen によって **web_url** 関数の **Resource** 属性を使って，再生中に対象リソースを取得するかどうかが指示されます。**Resource** 属性が 0 に設定されていると，対象リソースはスクリプトの実行中に取得されます。**Resource** 属性が 1 に設定されていると，そのリソース・タイプは仮想ユーザによってスキップされます。

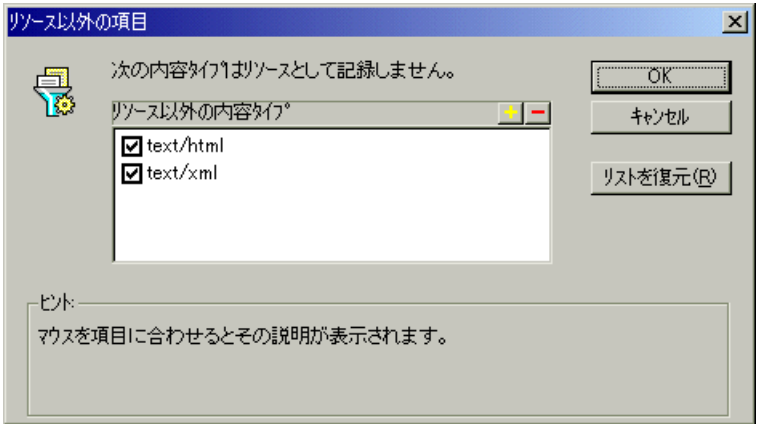
```
web_url("nav_tpo.gif",
        "URL=http://graphics.aa.com/images/navimg/nav_tpo.gif",
        "Resource=1",
        "RecContentType=image/gif",
        "Referer=http://www.im.aa.com/American?BV_EngineID=...",
        "Mode=HTML",
        LAST);
```

特定の内容タイプをリソースとして処理しないように除外することができます。たとえば，**gif** タイプのリソースがリソースとして処理されないように設定し，無条件にダウンロードされるように設定できます。VuGen によって **gif** タイプのリソースが検出されると，**Resource** 属性が 0 に設定され，再生時に gif を無条件にダウンロードされるようになります。

リソースとして記録しない内容を指定するには，次の手順で行います。

1 ［記録オプション］ツリーで，［**インターネット プロトコル：詳細**］ノードを選択します。

2 ［**リソース以外の項目**］をクリックしてダイアログ・ボックスを開き，リソースとして記録しない内容タイプのリストを表示します。



3 リストに内容タイプを追加するには，「+」記号をクリックします。既存のエントリを削除するには，「−」をクリックします。

4 項目を有効にするには，その横のチェック・ボックスを選択します。

5 リストを標準設定のリストに戻すには，［**リストを復元**］をクリックします。

6 ［**OK**］をクリックして設定を受け入れ，［リソース以外の項目］ダイアログ・ボックスを閉じます。

第 36 章

Web 仮想ユーザの記録オプションの設定

Web セッションを記録する前に，記録オプションをカスタマイズできます。

本章では，次の項目について説明します。

- 記録に使用するブラウザの指定
- 記録レベルの選択
- HTML ベースのオプションの詳細設定
- URL ベースのオプションの詳細設定
- 記録レベルの設定

以降の情報は，Web 仮想ユーザ・スクリプトを対象とします。

記録オプションの設定について

VuGen を使用して，ユーザが Web サイトで実行する標準的な操作を記録して，Web 仮想ユーザ・スクリプトを生成できます。

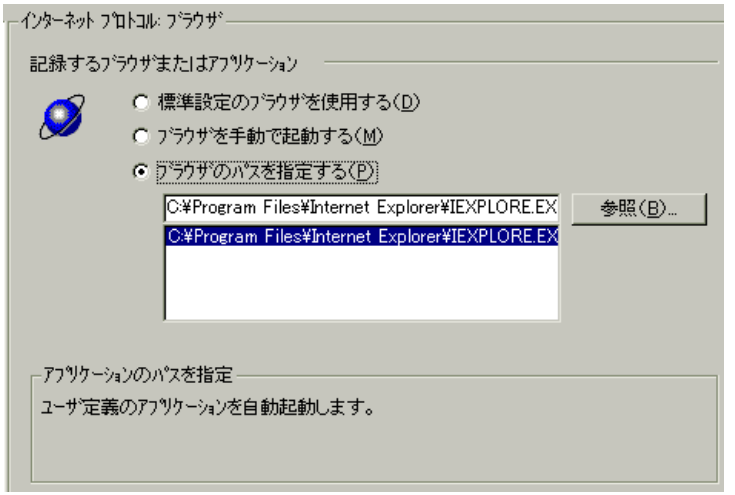
記録する前に，［記録オプション］を設定し，記録する情報，記録に使用するブラウザまたはクライアント，スクリプトの内容を指定します。

プロキシ設定やその他の詳細設定など，一般的なインターネット・プロトコルの記録オプションを設定できます。詳細については，第 35 章 「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。

また，Web 仮想ユーザ・スクリプトに対して，相関記録オプションを設定することもできます。詳細については，第 41 章 「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。

記録に使用するブラウザの指定

Web 仮想ユーザ・スクリプトを記録するときに，VuGen で使用するブラウザを指定できます。ブラウザの場所を指定するには，［記録オプション］ツリーの［**インターネットプロトコル：ブラウザ**］ノードを使います。



次の［**ブラウザ**］オプションを使用できます。

- ［**標準設定のブラウザを使用する**］：VuGen によって記録用コンピュータで通常使用する Web ブラウザとして指定されているブラウザが使用されます。
- ［**ブラウザを手動で起動する**］：VuGen によって記録の開始時にブラウザが起動されません。ユーザは，記録セッションを開始した後に，自分でブラウザまたはアプリケーションを起動しなければなりません。
- ［**ブラウザのパスを指定する**］：VuGen によって指定したブラウザが使用されます。パスのリストからパスを選択するか，［**参照**］ボタンをクリックして，使用するアプリケーションの場所を見つけます。

記録レベルの選択

記録レベルの選択によって，スクリプトの生成時に，記録する情報と使用する関数が指定できます。記録レベルは，目的または環境によって選択します。指定できるレベルは **HTML ベースのスクリプト**と **URL ベースのスクリプト**です。

HTML ベースの スクリプト記録レベルでは，HTML ユーザ・アクションごとにステップが生成されます。非 HTML 要素の記録は，HTML 以外の要素の処理の設定によって異なります（下記参照）。476 ページ「HTML ベースのオプションの詳細設定」も参照してください。

URL ベースのスクリプト記録レベルでは，ユーザの操作の結果，サーバに送信された HTTP 要求がすべてキャプチャされます。この記録モードでは，アプレットやブラウザ以外のアプリケーションなど，HTML 以外のアプリケーションもキャプチャされます。URL ベースのスクリプトのほうがスケーラブルであり，負荷テストの作成には効果的です。また，ユーザ・アクションに関する情報がより詳細に記録されますが，HTML ベースのスクリプトほど直観的ではありません。481 ページ「URL ベースのオプションの詳細設定」も参照してください。

以下のヒントに従って，どちらの記録レベルを選択するかを決定します。

- ブラウザ・アプリケーションの場合は，HTML ベースの記録レベルを使用します。
- 非ブラウザ・アプリケーションの場合は，URL ベースの記録レベルを使用します。

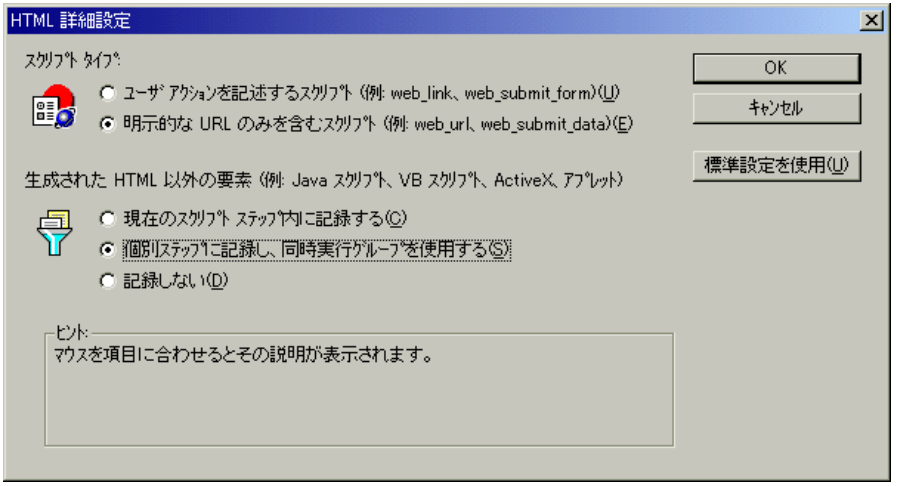
記録中に，記録レベルと詳細記録オプションを切り替えることができます。記録レベルを切り替える方法は，パフォーマンスのチューニングを行う上級ユーザ向けです。

HTML ベースのオプションの詳細設定

標準設定の記録レベルである［**HTML ベースのスクリプト**］オプションを選択すると，VuGen によって現在の Web ページのコンテキストにおいて HTML アクションが記録されます。記録セッション中，リソースのすべては記録されませんが，再生時にはダウンロードされます。

次の範囲について，HTML ベースのレベルの詳細オプションを設定できます。

- スクリプト・タイプの指定
- HTML 以外の要素の処理



スクリプト・タイプの指定

HTML ベースのレベルでは，次のタイプのスクリプトを指定できます。

- ユーザ・アクションを記述するスクリプト
- 明示的な URL のみを含むスクリプト

最初のオプション，［**ユーザ アクションを記述するスクリプト**］は標準設定のオプションです。アクションに直接対応する関数が作成されます。URL（**web_url**），リンク（**web_link**），画像（**web_image**），フォーム送信（**web_submit_form**）関数が作成されます。コンテキスト・センシティブ記録と似た非常にわかりやすいスクリプトが作成されます。

```
/* HTML ベース・モード：ユーザ・アクションを記述するスクリプト */
...
web_url("Click Here For Additional Restrictions",
        "URL=http://www.im.aa.com/American...restrictions.html",
        "TargetFrame=",
        "Resource=0",
        "RecContentType=text/html",
        "Referer=http://www.im.aa.com/American?...,
        "Snapshot=t4.inf",
        "Mode=HTML",
        LAST);

web_link("Click Here For Additional Restrictions",
        "Text=Click Here For Additional Restrictions",
        "Snapshot=t4.inf",
        LAST);

web_image("buttonhelp.gif",
        "Src=/images/buttonhelp.gif",
        "Snapshot=t5.inf",
        LAST);
...
```

2 つ目のオプションの［**明示的な URL のみを含むスクリプト**］では，すべてのリンク，画像および URL が **web_url** ステートメントとして記録されます。ただし，フォームの場合は，**web_submit_data** ステートメントとして記録されます。**web_link** 関数，**web_image** 関数，および **web_submit_form** 関数は生成されません。生成されるスクリプトは直観的ではなくなります。サイト内の多数のリンクが同じリンク・テキストを使用している場合に役立ちます。1 つ目のオプションを使用してサイトを記録すると，リンクの出現（インスタンス）が記録されますが，2 つ目のオプションを使用して記録すると，それぞれのリンクが URL のリストとして記録されます。このことにより，ステップの相関，およびパラメータ化を容易に行うことができます。

［明示的な URL のみを含むスクリプト］を選択した状態で記録したセッションの例を以下に示します。

```
/* HTML ベース：明示的な URL のみを含むスクリプト */
...;
web_url("Click Here For Additional Restrictions",
        "URL=http://www.im.aa.com/American...restrictions.html",
        "TargetFrame=",
        "Resource=0",
        "RecContentType=text/html",
        "Referer=http://www.im.aa.com/American?...;
        "Snapshot=t4.inf",
        "Mode=HTML",
        LAST);

web_url("buttonhelp.gif",
        "URL=http://www.im.aa.com/American?BV_EngineID...,
        "TargetFrame=aamain",
        "Resource=0",
        "RecContentType=text/html",
        "Referer=http://www.im.aa.com/American?...
        "Snapshot=t5.inf",
        "Mode=HTML",
        LAST);
...
```

HTML 以外の要素の処理

多くの Web ページには，アプレット，XML，ActiveX 要素，Java スクリプトなど，HTML 以外の項目が含まれています。通常こうした HTML 以外の要素には，それ自身のリソースが含まれているか，取得されるかします。たとえば，記録された Web ページから呼び出された Java スクリプト **js** ファイルによって，いくつかの画像がロードされることがあります。アプレットによって外部テキスト・ファイルがロードされることもあります。下記のオプションを使って，HTML 以外の要素をどのように記録するかを制御することができます。

次のオプションがあります。

- 現在のスクリプトステップ内に記録する（標準設定）
- 個別ステップに記録し，同時実行グループを使用する
- 記録しない

1 つ目のオプション［**現在のスクリプトステップ内に記録する**］では，HTML によらずに生成されたリソースについて新規関数が生成されません。非 HTML 要素について生成された **web_url** ステートメントの引数としてリソースがリストされます。**web_url** の引数となったリソースには **EXTRARES** フラグが付けられます。次の例では，ページにロードされた非 HTML 生成リソースがすべて **web_url** 関数の引数としてリストされています。

```
web_url("index.asp",
        "URL=http://www.daisy.com/index.asp",
        "TargetFrame=",
        "Resource=0",
        "RecContentType=text/html",
        "Referer=",
        "Snapshot=t2.inf",
        "Mode=HTML",
        EXTRARES,
        "Url=http://www.daisy.com/ScrollApplet.class", "Referer=", ENDITEM,
        "Url=http://www.daisy.com/board.txt", "Referer=", ENDITEM,
        "Url=http://www.daisy.com/nav_login1.gif", ENDITEM,
        ...
        LAST);
```

2 つ目のオプション，［**個別ステップに記録し，同時実行グループを使用する**］では，非 HTML 生成リソースの 1 つ 1 つについて新しい関数が生成されます。そのページの **web_url** 関数にはそれらが項目として含まれません。特定のリソースについて生成されたすべての **web_url** 関数は，同時実行グループ内に配置されます（**web_concurrent_start** と **web_concurrent_end** に囲まれます）。

次の例では，前述のセッションを，このオプションを設定した状態で記録したものです。アプレットとそのアプレットによってロードされたテキスト・ファイルに対して **web_url** 関数が生成されています。

```
web_url("index.asp",
        "URL=http://www.daisy.com/index.asp",
        "Resource=0",
        "RecContentType=text/html",
        "Referer=",
        "Snapshot=t2.inf",
        "Mode=HTML",
        LAST);

web_concurrent_start(NULL);
    web_url("ScrollApplet.class",
        "URL=http://www.daisy.com/ScrollApplet.class",
        "Resource=1",
        "RecContentType=application/octet-stream",
        "Referer=",
        LAST);

    web_url("board.txt",
        "URL=http://www.daisy.com/board.txt",
        "Resource=1",
        "RecContentType=text/plain",
        "Referer=",
        LAST);
web_concurrent_end(NULL);
```

3 つ目のオプション，［**記録しない**］では，非 HTML 要素によって生成されたリソースが記録されないよう設定されます。

HTML ベースのモードで作業している場合は，VuGen によって **web_url** ステートメント内に **TargetFrame** 属性が挿入されます。この情報は，実行時ブラウザとテスト結果レポートに Web ページを正しく表示するために使用されます。

```
web_url("buttonhelp.gif",
        "URL=http://www.im.aa.com/American?BV_EngineID=...",
        "TargetFrame=aamain",
        "Resource=0",
        "RecContentType=text/html",
        "Referer=http://www.im.aa.com/American?BV_EngineID=...",
        "Snapshot=t5.inf",
        "Mode=HTML",
        LAST);
```

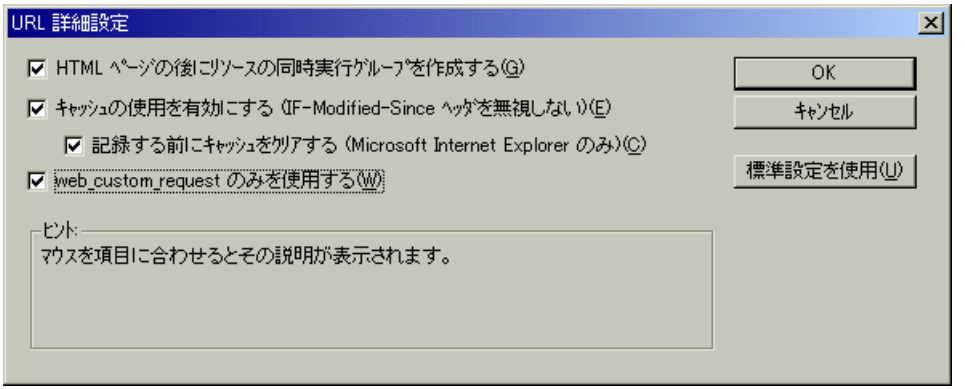
URL ベースのモードでの記録では，VuGen によってページ上のすべてのフレームの内容が記録され，TargetFrame 属性が省略されます。

URL ベースのオプションの詳細設定

URL ベース・レベルのオプションを選択すると，VuGen によってサーバからの**すべての**要求とリソースが記録されます。自動的にあらゆる HTTP リソースが URL ステップ（**web_url** ステートメント）として，フォームの場合には，**web_submit_data** として記録されます。**web_link**，**web_image**，および **web_submit_form** 関数は生成されず，フレームも記録されません。

次の範囲について，URL 記録モードの詳細オプションを設定できます。

- リソースの処理
- ブラウザのキャッシュ



リソースの処理

URL ベースでの記録では，VuGen によって，HTTP 要求の結果としてダウンロードされたすべてのリソースがキャプチャされます。標準設定ではこのオプションが有効になっており，URL の後，リソースが同時実行グループとして記録されます（**web_concurrent_start** と **web_concurrent_end** で囲まれます）。リソースには，画像，インポート・ファイル，**js** ファイルなどのファイルが含まれます。このオプションを無効にすると，リソースは別々の **web_url** ステップとしてリストされますが，同時実行グループとしてのマークは付きません。

次のコードは，［**HTML ページの後にリソースの同時実行グループを作成する**］オプションを無効にして記録したセッションを示します。

```
web_concurrent_start(NULL);
...
web_url("spacer.gif",
        "URL=http://graphics.aa.com/images/spacer.gif",
        "Resource=1",
        "RecContentType=image/gif",
        "Referer=http://www.im.aa.com/American?BV_EngineID...",
        "Mode=HTTP",
        LAST);

web_url("calendar_functions.js",
        "URL=http://www.im.aa.com/travelp/calendar_functions.js",
        "Resource=1",
        "RecContentType=application/x-javascript",
        "Referer=http://www.im.aa.com/American?BV_Operation=...l",
        "Mode=HTTP",
        LAST);
...
web_concurrent_end(NULL);
```

スクリプトに，**gif** ファイルと **js** ファイルが含まれているのに注目してください。このモードでは，**imp**，**txt**，カスケーディング・スタイル・シート（**css**）や，その他のグラフィック・ファイル，インポートされたファイルも記録の対象となります。

ブラウザのキャッシュ

ブラウザのキャッシュによって，Web ページへのアクセスに要する時間を短縮するために，直近に表示されたページがマシンのメモリに格納されます。標準設定では，［**キャッシュの使用を有効にする**］オプションが無効になっています。つまり，VuGen によってサーバから直接すべてのページが取得され，記録時にブラウザのキャッシュが使用されません。

ただし，アプリケーションによっては，キャッシュなしでは実行できないものもあります。キャッシュを有効にして，新しく変更されたページのみを直接サーバから取得するには，［**キャッシュの使用を有効にする**］オプションを選択します。

HTTP ヘッダー，「**If-Modified-Since**」は，キャッシュされたリソースが最後のダウンロードからサーバ側で更新されたかどうかをクライアント側で検査するときに使用される要求です。リソースが変更されていると，クライアントによってそのリソースが再びキャッシュにダウンロードされます。そうでない場合は，サーバから HTTP ステータス・コード 304「Not Modified」が返されます。キャッシュが無効になっている場合，「**If-Modified-Since**」ヘッダーは抑止され，サーバからすべてのページが直接取得されます。このモードでは，応答ヘッダーの **Last-Modified**，**Expires**，および **Etag** だけでなく，要求ヘッダーの **If-None-Match** も削除されます。ブラウザ側でこれらの応答ヘッダーのいずれも受信されなかった場合，画像はブラウザのキャッシュに格納されません。

これらのヘッダーは，［ヘッダ］オプションで手作業で調整できます。詳細については，467 ページ「ユーザ定義ヘッダーの記録」を参照してください。

ブラウザのキャッシュのクリア

標準設定では，ブラウザのキャッシュが有効になっている場合，VuGen によって記録の前にブラウザのキャッシュがクリアされます。つまり，キャッシュ内のすべての内容を期限切れにすることによって，内容をサーバから直接取得しなければならないようにします。

キャッシュがクリアされるため，最近アクセスがあったページも含めて，Web サイトのすべてのページに直接アクセスしなければなりません。1 つのサイトに繰り返しアクセスする仮想ユーザを記録している場合は，記録の前にブラウザのキャッシュをクリアしないように設定することもできます。

記録前にブラウザのキャッシュをクリアしないようにするには，［**記録する前にキャッシュをクリアする**］チェック・ボックスをクリアします。このオプションは Internet Explorer を使って記録している場合のみ適用されます。

ユーザ定義要求の生成

ブラウザ以外のアプリケーションで記録している場合，すべての HTTP 要求をユーザ定義要求として記録するよう設定できます。VuGen によって内容に関係なくすべての要求に **web_custom_request** 関数が生成されます。

```
web_custom_request("www.aa.com",
        "URL=http://www.aa.com/",
        "Method=GET",
        "Resource=0",
        "RecContentType=text/html",
        "Referer=",
        "Snapshot=t1.inf",
        "Mode=HTTP",
        LAST);
```

EUC エンコードされた Web ページの有効化

（以下の説明は日本語版 Windows のみに関するものです）Windows の標準文字セット以外を扱う場合には，コード変換が必要になることがあります。文字セットは，文字の集合と整数の集合との対応関係を表します。この対応関係によって，与えられた 1 つの文字について，整数との一意の組み合わせが成立します。EUC（Extended UNIX Code，拡張 UNIX コード）と SJIS（Shift Japan Industry Standard，シフト JIS）は，Windows の標準文字セットではなく，Web サイトで日本語を表示するために使用されます。

Windows では SJIS コードを使いますが，UNIX では EUC コードを使います。Web サーバが UNIX マシン上にあり，クライアントが Windows の場合，コードが違うため，Web サイトの文字がクライアント側で正しく表示されません。このことは，EUC コードの日本語文字を仮想ユーザ・スクリプトで表示するときに影響します。

記録中，LoadRunner は HTTP ヘッダーを通じて Web ページで使われているコードを検出します。文字セットに関する情報が HTTP ヘッダーに存在しない場合は，HTML のメタ・タグを調べます。文字セットの情報がページから HTTP ヘッダーまたはメタ・タグに送信されない場合，LoadRunner は EUC コードが使われていることを検知しません。

それでも Web ページで EUC コードが使われていることが事前にわかっていれば，LoadRunner が正しいコードで記録するようにできます。ページを EUC コードで記録するには，［記録オプション］**記録**タブの［**EUC**］オプションを有効にします（日本語版 Windows でのみ表示されます）。

［**EUC**］オプションを有効にすると，EUC コードで書かれていない Web ページも強制的に EUC コードで記録されます。したがって，このオプションを有効にするべきなのは，LoadRunner が HTTP ヘッダーまたは HTML のメタ・タグからはコードを検知できず，なおかつページが EUC コードで書かれていることが事前にわかっているときだけです。

記録中，LoadRunner は EUC コードの文字列を Web サーバから受信すると，それを SJIS に変換します。変換された SJIS 文字列はスクリプトの Action 関数に保存されます。しかし，再生を正常に実行するためには，文字列を再び EUC に変換してから Web サーバに送り返す必要があります。このため，LoadRunner は Action 関数の前に **web_sjis_to_euc_param** 関数を追加します。この関数で SJIS 文字列を EUC に再変換します。

次の例では，ユーザが EUC で書かれた Web ページに移動してリンクをクリックします。LoadRunner は Action 関数を記録し，**web_sjis_to_euc_param** 関数を Action 関数の前のスクリプトに追加します。

```
web_sjis_to_euc_param("param_link","Search");

web_link("LinkStep","Text={param_link}");
```

記録レベルの設定

本項では，記録レベルとそれらの詳細オプションの設定の手順について説明します。なお，記録のレベルと詳細記録オプションは記録中にも切り替えることができます。

記録オプションを設定するには，次の手順で行います。

1 ［**ツール**］＞［**記録オプション**］を選択して，［記録オプション］ダイアログ・ボックスを開きます。記録オプションの［**インターネット プロトコル**］の［**記録**］ノードを選択します。

2 記録モードとして［**HTML ベースのスクリプト**］または［**URL ベースのスクリプト**］を選択します。

3 HTML ベースのモードの場合，［**HTML 詳細設定**］をクリックし，スクリプトのタイプや非 HTML 要素の処理についての追加のオプションを設定します。

スクリプトのタイプを選択します。

非 HTML リソースを処理する方法を選択します。詳細については，476 ページ「HTML ベースのオプションの詳細設定」を参照してください。

4 URL ベースのモードの場合，［**URL 詳細設定**］をクリックし，リソースの処理やキャッシュの有効化について追加のスクリプト・オプションを設定します。

リソースを記録し，同時実行グループとしてマークするには，［**HTML ページの後にリソースの同時実行グループを作成する**］を選択します（同時実行グループは，**web_concurrent_start** と **web_concurrent_end** で囲まれます）。

記録中にブラウザ・キャッシュを使用するには，［**キャッシュの使用を有効にする**］を選択します。このオプションを有効にした場合は，［**記録する前にキャッシュをクリアする**］チェック・ボックスをクリアしておくと，記録前に，キャッシュがクリアされず，以前にアクセスしたページが使用されます。

すべての HTTP 要求を **web_custom_request** 関数として生成するには，［**web_custom_request のみを使用する**］を選択します。

5 これらのオプションの詳細については，481 ページ「URL ベースのオプションの詳細設定」を参照してください。

第 37 章

インターネット実行環境の設定

インターネット・プロトコル仮想ユーザ・スクリプトを記録した後で，そのスクリプトの実行環境を設定できます。

本章では，次の項目について説明します。

- プロキシ・オプションの設定
- ブラウザのエミュレーション・プロパティの設定
- インターネット環境の設定

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

インターネット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。

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 9 章「実行環境の設定」を参照してください。ネットワーク速度の実行環境の設定に関する詳細については，第 10 章「インターネット実行環境の設定」を参照してください。

インターネット実行環境の設定によって，仮想ユーザが実際のユーザを正しく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，プロキシやブラウザその他の詳細な環境設定が行えます。

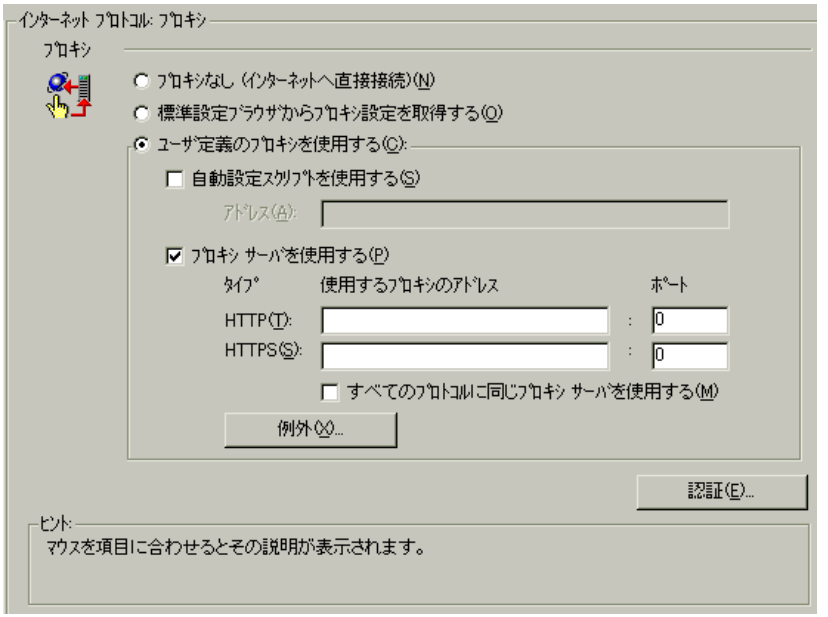
インターネット関連の実行環境の設定は，［実行環境設定］ダイアログ・ボックスで行います。必要な設定を行うためのノードをクリックします。

LoadRunner コントローラからでも実行環境の設定を変更できます。［コントローラ］ウィンドウで［**デザイン**］タブをクリックし，［**実行環境の設定**］ボタンをクリックします。

注：仮想ユーザ関数を使って行われた実行環境の設定は，［実行環境設定］ダイアログ・ボックスで行った設定に優先します。仮想ユーザ関数の使い方については，第 33 章「Web 仮想ユーザ関数の使用」を参照してください。

プロキシ・オプションの設定

［実行環境設定］ツリーの［**インターネット プロトコル**］の［**プロキシ**］ノードで，プロキシ関連の設定を行います。



標準設定では，仮想ユーザでは Web 記録オプションで指定している記録用ブラウザのプロキシ設定が使用されます。記録と再生には同じ設定を使用することを推奨します。［記録用プロキシ］オプションに関する情報は，第 36 章「Web 仮想ユーザの記録オプションの設定」を参照してください。

［実行環境設定］には，次のようなプロキシの設定オプションがあります。

- ［**プロキシなし（インターネットへ直接接続）**］：すべての仮想ユーザが，インターネットに常に直接接続する。つまり，プロキシ・サーバを使用せずに接続します。
- ［**標準設定のブラウザからプロキシ設定を取得する**］：すべての仮想ユーザが実行されているマシンから，通常使うブラウザとなっているプロキシの設定を使用する。
- ［**ユーザ定義のプロキシを使用する**］：すべての仮想ユーザが同じユーザ定義のプロキシ・サーバを使用する。実際のプロキシ・サーバの構成情報や，自動設定のための設定スクリプト（**.js** 形式の pac ファイル）へのパスなどで設定を行います（詳細については，490 ページ「自動プロキシ設定の設定方法」を参照してください）。

  サーバの構成情報で指定する場合，IP アドレス，名前，ポート番号を使用します。HTTP サイト用にプロキシ・サーバを 1 つ指定し，HTTPS サイト（セキュア・サイト）用に別のプロキシ・サーバを指定することもできます。

  プロキシ情報を指定した後，プロキシ・サーバの認証情報と，プロキシ・ルールに対する例外を指定できます。

注：再生中にプロキシの応答を待つように仮想ユーザに指示し，プロキシが基本認証をサポートしていると仮定しないようにするには，次のステートメントを追加します。web_set_sockets_option("PROXY_INITIAL_BASIC_AUTH", "0");

認証

プロキシ・サーバが各仮想ユーザの認証を必要とする場合は，このダイアログ・ボックスを使用して適切なパスワードとユーザ名を入力します。

［**ユーザ名**］：仮想ユーザがプロキシ・サーバへのアクセスに使用するユーザ名を入力します。

［**パスワード**］：仮想ユーザがプロキシ・サーバにアクセスするために必要なパスワードを入力します。

例外

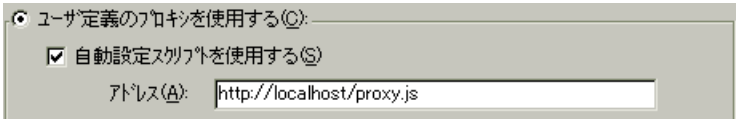
すべての仮想ユーザが指定されたプロキシ・サーバを使用するように指定できます。この場合，仮想ユーザからプロキシ・サーバを使用せずに直接アクセスできるようにしたい URL があれば，その URL のリストをテキスト・ボックスに入力します。

［**次で始まるアドレスにはプロキシ サーバを使用しない**］：プロキシ・サーバから除外するアドレスを入力します。各項目はセミコロンで区切ります。

［**ローカル アドレス（イントラネット）にはプロキシ サーバを使用しない**］：イントラネットからのアドレスなど，ローカル・アドレスをプロキシ・サーバから除外するには，このチェック・ボックスを選択します。

自動プロキシ設定の設定方法

自動プロキシ設定は，ほとんどのブラウザでサポートされています。この機能では，プロキシ割り当て情報が含まれた JavaScript ファイル（**.js** 拡張子付き）を指定できます。このスクリプトは，URL に基づいて，どの場合にはプロキシ・サーバを使用して，どの場合には直接サイトへ接続するのかをブラウザに指定します。また，このスクリプトでは，アドレスごとに異なるプロキシ・サーバを使うよう指定することができます。Internet Explorer（IE）でスクリプトを設定する場合は，［**ツール**］＞［**インターネット オプション**］を選択し，［接続］タブを選択します。［**LAN の設定**］ボタンをクリックします。LAN 設定ダイアログ・ボックスで，［**自動設定スクリプトを使用する**］オプションを選択し，スクリプトの場所を指定します。VuGen では，自動プロキシ設定をサポートしています。自動プロキシ設定には JavaScript を指定できるため，LoadRunner ではテストの実行時に，その JavaScript ファイルから得られたルールが使用されます。



仮想ユーザの振る舞いを追跡するには，テキスト実行中にログを生成し，［**実行ログ**］タブまたは **mdrv.log** ファイルを調べます。ログには，各 URL に対して使用されたプロキシ・サーバが表示されます。次の例では，URL australia.com へ直接接続していますが，URL http://www.google.com についてはプロキシ・サーバ **aqua** が使われています。

```
Action1.c(6):t=1141ms:FindProxyForURL returned DIRECT
Action1.c(6):t=1141ms:Resolving australia.com
Action1.c(6):t=1141ms:Connecting to host 199.203.78.255:80
...
Action1.c(6):t=1281ms:Request done
"http://australia.com/GetElementByName.htm"


...
Action1.c(6):web_url was successful, 357 body bytes, 226 header bytes
Action1.c(15):web_add_cookie was successful
Action1.c(17):t=1391ms:FindProxyForURL returned PROXY aqua:2080
Action1.c(17):t=1391ms:Auto-proxy configuration selected proxy
aqua:2080
Action1.c(17):t=1391ms:Resolving aqua
Action1.c(17):t=1391ms:Connecting to host 199.203.139.139:2080
...l
Action1.c(17):t=1578ms:"http://www.google.com/" (RelFrameId=1) へ送出した 168 バイトの要求ヘッダー
Action1.c(17):GET http://www.google.com/ HTTP/1.1¥r¥n
```

プロキシ実行環境の設定

以下の項では，プロキシ実行環境の設定に必要な手順について説明します。

プロキシの設定を行うには，次の手順で行います。

1 ［実行環境設定］ダイアログ・ボックスを開きます。VuGen ツールバーにある［**実行環境の設定を編集**］ボタンをクリックするか，［**仮想ユーザ**］＞［**実行環境の設定**］を選択します。

2 プロキシ・オプションを選択します。［**プロキシなし（インターネットへ直接接続）**］，［**標準設定のブラウザからプロキシ設定を取得する**］，［**ユーザ定義のプロキシを使用する**］があります。

3 ユーザ定義のプロキシを指定した場合は，次を設定します。

➤ HTTP および HTTPS プロキシ・サーバの IP アドレスを指定します。

➤ プロキシの指定に javascript ファイルを使う場合は，［**自動設定スクリプトを使用する**］オプションを選択し，スクリプトの場所を指定します。http:// で始まる Web の場所（たとえば，**http://hostname/proxy.js**）や，ファイル・サーバの場所（たとえば，**C¥temp¥proxy.js**）を指定します。

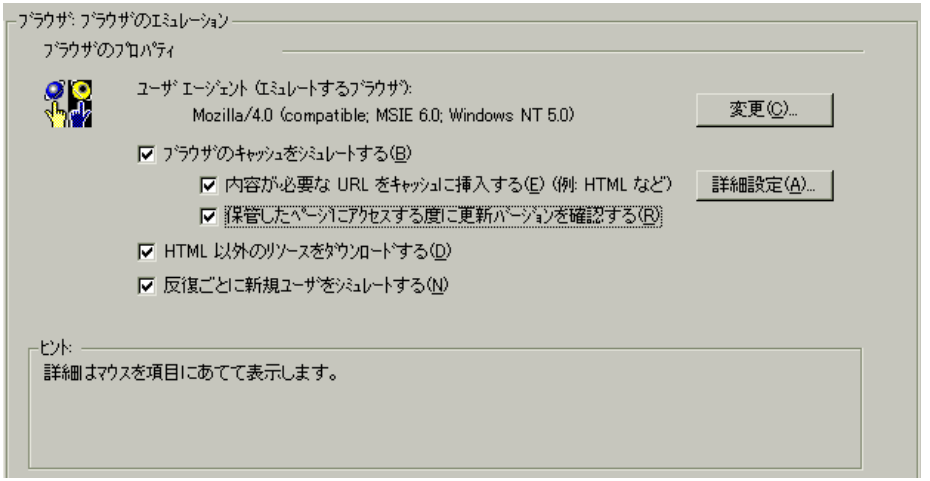
4 仮想ユーザが直接（すなわちプロキシ・サーバを使わずに）アクセスする URL を指定するには，［**例外**］をクリックしてそれらの URL をリストに追加します。［例外］ダイアログ・ボックスでは，ローカル（イントラネット）アドレスへの直接アクセスも指定できます。

5 プロキシ・サーバが各仮想ユーザの認証を必要とする場合は，［**認証**］をクリックして，適切なパスワードとユーザ名を入力します。

6 ［**すべてのプロトコルに同じプロキシ サーバを使用する**］チェック・ボックスを選択した場合，仮想ユーザはセキュリティが保護されたサイトのためのサーバを指定せず，すべてのインターネット・プロトコル（HTTP，HTTPS）で同じプロキシ・サーバを使用します。

ブラウザのエミュレーション・プロパティの設定

［実行環境設定］ダイアログ・ボックスの［**ブラウザ**］の［**ブラウザのエミュレーション**］ノードでは，テスト環境におけるブラウザのプロパティについて設定します。



ブラウザのプロパティ

次の項目についてブラウザのプロパティの設定が可能です。

- エミュレートするユーザ・エージェント・ブラウザ
- ブラウザのキャッシュをシミュレートする
- HTML 以外のリソースをダウンロードする
- 反復ごとに新規ユーザをシミュレートする

新しいリソースのキャッシュと確認のための詳細オプションも設定できます。

エミュレートするユーザ・エージェント・ブラウザ

仮想ユーザによって Web サーバに要求が送信されるとき，要求には必ず HTTP ヘッダーが含まれています。テキストの 1 行目には，メソッド（通常，「GET」または「POST」），リソース名（たとえば「pclt/default.htm」），プロトコルのバージョン（たとえば「HTTP/1.0」）が含まれています。2 行目以降には，「ヘッダー情報」が「属性名，コロン，値」の形式で含まれています。要求は，空白行で終わります。

仮想ユーザのヘッダーには，エミュレートされているブラウザの種類を示す「**User-Agent**」ヘッダーが含まれています。例を以下に示します。

```
User-Agent:Mozilla/3.01Gold (WinNT; I)
```

上記の例は，ブラウザは Intel 社の CPU を搭載したマシン上の Windows NT で動作する Mozilla/3.01Gold がエミュレーション対象のブラウザであることを示しています。

［ブラウザのエミュレーション］ノードから［**変更**］をクリックし，ヘッダーに含めるブラウザ情報を指定します。Web 仮想ユーザに対して，いくつかの標準的なブラウザをエミュレートするように指定できます。または，ブラウザ以外の HTTP アプリケーションの場合は，そのアプリケーションと組み合わせる HTTP クライアントを指定できます。この場合は，以降の HTTP ヘッダーに含める「**ユーザ定義のブラウザを使用する**」を示す文字列を指定しなければなりません。標準設定では，仮想ユーザは Internet Explorer 4.0 ブラウザ・エージェントを使用します。

このオプションは，再生に使用するブラウザではなく，スクリプトの実行時に使用するブラウザを指定するものです。この設定の影響を受けるのは，サーバに送られる HTTP ヘッダーの **User-Agent** 属性のみです。

ブラウザのキャッシュをシミュレートする

このオプションを選択すると，仮想ユーザはキャッシュを利用しながらブラウザをシミュレートします。キャッシュは頻繁にアクセスするドキュメントのローカル・コピーを保存するのに使われます。キャッシュを使うことにより，仮想ユーザのネットワーク接続時間を削減します。標準設定では，キャッシュ・シミュレーションは有効になっています。キャッシュが無効になっているときにも，ページの画像は LoadRunner によって一度だけダウンロードされます。コントローラで複数の仮想ユーザを実行する場合は，各仮想ユーザではそれぞれのキャッシュが使用され，キャッシュから画像が取得されます。このオプションを無効にすると，すべての仮想ユーザではキャッシュが利用されずにブラウザがエミュレートされます。

実行環境の設定を変更して，使用しているブラウザの設定を Internet Explorer に一致させることができます。

ブラウザの設定	実行環境の設定
［ページを表示するごとに確認する］	［**ブラウザのキャッシュをシミュレートする**］を選択して，［**保管したページにアクセスする度に更新バージョンを確認する**］を有効にします。
［Internet Explorer を起動するごとに確認する］	［**ブラウザのキャッシュをシミュレートする**］だけを選択します。
［自動的に確認する］	［**ブラウザのキャッシュをシミュレートする**］だけを選択します。
［確認しない］	［**ブラウザのキャッシュをシミュレートする**］を選択して，［**保管したページにアクセスする度に更新バージョンを確認する**］を無効にします。

注：通常のブラウザ・キャッシュとは異なり，仮想ユーザに割り当てられたキャッシュは，画像ファイルの格納だけをシミュレートします。このキャッシュは，Web ページに関連付けられたテキストやそのほかの内容は格納しません。

VuGen では，次の 2 つのブラウザ・キャッシュ・オプションを設定できます。

［**内容が必要な URL をキャッシュに挿入する（例：HTML など）**］：このオプションを有効にすると，VuGen は HTML 内容を要求する URL のみをキャッシュに格納します。内容は，解析，検証，または相関に必要となる場合があります。このオプションを選択すると，HTML 内容は自動的にキャッシュに保存されます。キャッシュに格納するその他の内容タイプを定義するには，［**詳細設定**］をクリックします（このオプションを有効にすると，仮想ユーザのメモリ使用量が増加します）。標準設定では，このオプションは無効になっています。詳細については，496 ページ「内容が必要な URL のキャッシュの詳細設定」を参照してください。

［**保管したページにアクセスする度に更新バージョンを確認する**］：この設定を有効にすると，ブラウザは指定された URL について，キャッシュに格納されたものではなく，その最新版がないか確かめます。このオプションを有効にすると，HTTP ヘッダーに「If-modified-since」属性が追加されます。このオプションを有効にすると，ページの最新版が常に表示されるようになりますが，シナリオの実行時に発生するトラフィックが増えます。標準設定では，このオプションは無効となっており，ブラウザは新しいリソースの有無をチェックしません。このオプションは，エミュレートするブラウザの設定と一致するよう設定します。

HTML 以外のリソースをダウンロードする

このオプションを選択すると，仮想ユーザは，再生中に Web ページにアクセスするときに画像ファイルをロードします。これには，ページとともに記録された画像イメージと，明示的にはページとともに記録されていない画像イメージの両方が含まれます。実際のユーザは，Web ページにアクセスするとき，画像がロードされるのを待ちます。したがって，エンド・ユーザ時間を含め，システム全体をテストする場合には，このオプションを有効にします（標準設定では有効）。パフォーマンスを向上させるために，実際のユーザをエミュレートしない場合は，このオプションを無効にします。

注：Web ページにアクセスするたびに画像が変わり（広告業者のバナーなど），画像チェックで不一致が生じる場合には，このオプションを無効にします。

反復ごとに新規ユーザをシミュレートする

このオプションを選択すると，VuGen は，反復を行う前にすべての HTTP コンテキストを **init** セクションの終了時の状態にリセットします。この設定により，仮想ユーザは新規ユーザがブラウザ・セッションを開始する様子をより正確にエミュレートできます。クッキーをすべて削除するには，すべての TCP 接続を閉じ（キープ・アライブを含む），エミュレートされたブラウザのキャッシュをクリアします。次に HTML フレーム階層をリセットして（フレームは番号 1 から番号付けされる）ユーザ名およびパスワードをクリアします。標準設定では，このオプションは有効になっています。

内容が必要な URL のキャッシュの詳細設定

［詳細設定］ダイアログ・ボックスでは，キャッシュに格納する URL 内容タイプを指定できます。このダイアログ・ボックスには［実行環境設定］ダイアログ・ボックスの［ブラウザ］の［ブラウザのエミュレーション］ノードからアクセスできます。

複数のグループの詳細オプションを一度に変更する操作はサポートされていません。グループごとに個別に設定を編集します。

内容タイプを追加するには，次の手順で行います。

1 ［**HTML ページ以外にも内容が必要な URL を指定する**］オプションを有効にします。

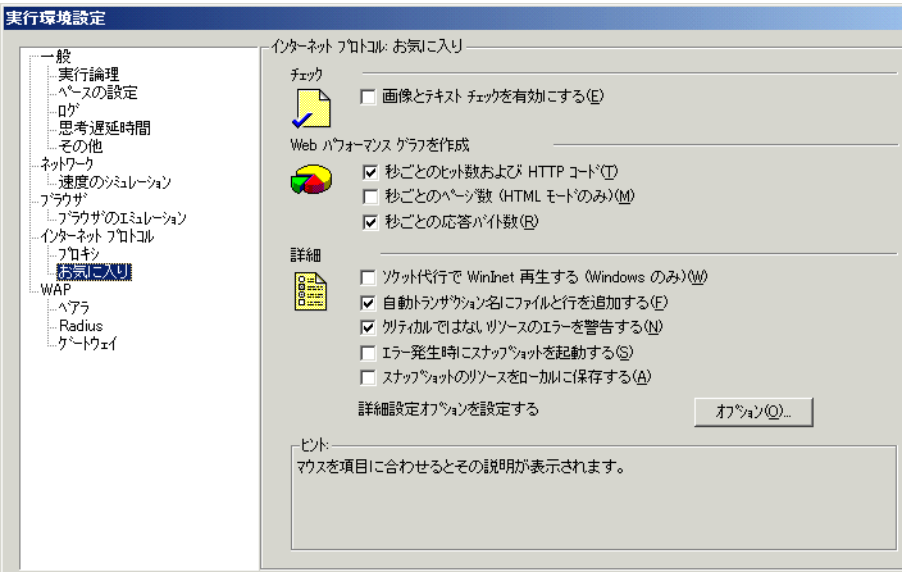
2 プラス記号をクリックし，text/plain，text/xml，image/jpeg，image/gif などの内容タイプを追加します。内容名をテキスト・ボックスに入力します。

3 内容タイプをリストから削除するには，削除する内容タイプを選択してマイナス記号をクリックします。

インターネット環境の設定

［実行環境設定］ダイアログ・ボックスの［**インターネット プロトコル**］の［**お気に入り**］ノードでは，次の項目に関連する設定を実行できます。

- 画像とテキストのチェック
- Web パフォーマンス・グラフを作成
- Web 実行環境の詳細オプション



画像とテキストのチェック

［**画像とテキストチェックを有効にする**］チェック・ボックスを選択すると，仮想ユーザの再生中に **web_find** または **web_image_check** チェック関数を実行して，画像とテキストの検証を行うことができます。このオプションは，HTTP モードで記録されたステートメントにのみ適用されます。仮想ユーザで検証チェックを実行すると，検証チェックを実行しない仮想ユーザより多くのメモリが消費されます（標準設定では無効）。

Web パフォーマンス・グラフを作成

このオプションを選択すると，仮想ユーザによって Web パフォーマンス・グラフの作成に使用するデータが収集されます。テスト実行中はオンライン・モニタ，テスト実行後はアナリシスを使用して，**毎秒のヒット数**，**毎秒のページ数**，**毎秒の応答バイト数（スループット）**のグラフを表示できます。アナリシスを使用して，テスト実行後にコンポーネント・ブレークダウン・グラフを表示できます。仮想ユーザに収集させるグラフ・データのタイプを選択します。

注：Web パフォーマンス・グラフを使用しない場合には，これらのオプションは，メモリを節約するために無効にしておきます。

Web 実行環境の詳細オプション

［**ソケット代行で WinInet 再生する**］：このオプションを選択すると，VuGen は WinInet 再生エンジンを使用します。VuGen には，Sockets ベース（標準設定）および WinInet ベースの 2 つの HTTP 再生エンジンがあります。WinInet は Internet Explorer で使用されるエンジンであり，IE ブラウザに組み込まれている機能をすべてサポートしています。WinInet 再生エンジンの制約として，スケーラブルでないこと，UNIX をサポートしていないことが挙げられます。さらに，スレッドでの作業時に WinInet エンジンがモデム速度および接続数を正確にエミュレートしないことがあります。

VuGen 独自のソケット・ベースの再生機能は負荷テストにおけるスケーラビリティに優れた小型軽量のエンジンです。また，スレッドでの使用時にも正確に機能します。ソケット・ベースのエンジンの制限事項としては，SOCKS プロキシがサポートされていない点があります。このタイプの環境を記録する場合は，WinInet 再生エンジンを使用してください。

［**自動トランザクション名にファイルと行を追加する**］：トランザクション名にファイル名と行番号を追加し，自動トランザクション時に一意となるトランザクション名を作成します（標準設定では有効）。

注：このオプションを選択すると，ログ・ファイルに記録される情報が増えるため，より多くのメモリが必要になります。

［**クリティカルではないリソースのエラーを警告する**］：ダウンロードに失敗した画像や Java アプレットなど，負荷テストにとってさほど重要ではない項目で，失敗した関数についての警告ステータスを返します。標準設定では，このオプションは有効になっています。クリティカルではないエラーも失敗として扱う環境では，このオプションを無効にできます。非リソースのリストに content-type を加えることで，その内容タイプを重要なものとして設定できます。詳細については，470 ページ「リソース以外の内容タイプの指定」を参照してください。

［**エラー発生時にスナップ ショットを起動する**］：エラーの発生時にスナップショットを作成します。コントローラで仮想ユーザ・ログを表示し，エラーをダブルクリックするとスナップショットを表示できます。

［**スナップショットのリソースをローカルに保存する**］：VuGen はスナップショット・リソースをローカル・マシン上にファイルとして保存します。この機能を使用することで，実行時ビューアはより正確なスナップショットを作成し，よりすばやく表示することができます。

お気に入りのその他のオプション

［お気に入り］では，次の詳細設定オプションの設定が行えます：DNS のキャッシュ，HTTP バージョン，HTTP 接続の Keep-Alive，HTTP 要求接続タイムアウト，HTTP 要求受領タイムアウト，ネットワーク・バッファ・サイズ，ステップダウンロード・タイムアウト。

［**DNS のキャッシュ**］：このオプションを選択すると，仮想ユーザは，ドメイン・ネーム・サーバによって解決されたホストの IP アドレスをキャッシュに保存します。これにより，それ以降の同じサーバに対する呼び出しに費やす時間を節約できます。ロード・バランサーなどの技術によって自動的に IP アドレスが変更される場合には，このオプションを確実に無効にしておき，仮想ユーザがキャッシュの値を使用しないようにします（標準では有効）。

［**HTTP のバージョン**］：HTML バージョン：バージョン 1.0 または 1.1。バージョン情報は，仮想ユーザが Web サーバにリクエストを送信するときの HTTP 要求ヘッダーに含まれています。標準設定は 1.1 です。HTTP 1.1 では次の機能がサポートされています。

- 持続的な接続 ：下の「HTTP 接続 の Keep-Alive」を参照してください。
- HTML 圧縮：502 ページ「HTML 圧縮の実行」を参照してください。
- 複数のドメイン名が同一の IP アドレスを共有する，仮想ホスティング。

［**HTTP 接続 の Keep-Alive**］：は，持続的な接続を可能にする HTTP 拡張機能を表す用語です。こうした長時間の HTTP セッションでは，複数のリクエストを同一の TCP 接続を介して送信できます。これにより，Web サーバとクライアントのパフォーマンスが向上します。

この Keep-Alive オプションは，Keep-Alive 接続をサポートする Web サーバがあって初めて機能します。この設定により，仮想ユーザ・スクリプトを実行するすべての仮想ユーザに対して Keep-Alive HTTP 接続を有効にすることができます（標準では有効です）。

［**リソースによるステップのタイムアウトを警告とする**］：タイムアウトの時間内にロードされなかったリソースが原因でタイムアウトが発生した場合に，エラーではなく警告を発行します。リソース以外の場合は，常にエラーが発行されます（標準では無効）。

［**HTML Content-Type を解析する**］：要求した HTML について，次の特定の内容タイプが指定されていれば，その応答を解析します。**HTML**, **text¥html**, **TEXT**（任意のテキスト），**ANY**,（任意の内容タイプ）。text/xml は HTML としては解析されません。標準値は［**TEXT**］です。

［**サーバ サイド圧縮を受け付ける**］：再生で圧縮データを扱うことができることをサーバに指示します。圧縮データを扱うことにより，CPU の消費量が著しく増加する場合があります。

［**HTTP 要求接続タイムアウト（秒）**］：仮想ユーザが，ステップ内の特定の HTTP 要求への接続を中断するまで待機する時間（秒単位）です。タイムアウトは，要求に対してサーバが安定しユーザに応答するための猶予を与えるものです。標準時間は 120 秒です。

［**HTTP 要求受領タイムアウト（秒）**］：仮想ユーザがステップ内の特定の HTTP 要求への接続を中断するまで待機する時間（秒単位）です。タイムアウトは，要求に対してサーバが安定しユーザに応答するための猶予を与えるものです。標準時間は 120 秒です。

タイムアウト設定は主に，該当システム環境下では，許容タイムアウト値を変えるのが望ましいと判断できる上級ユーザ向けです。ほとんどの場合，標準の値で問題ないはずです。サーバが妥当な時間内に応答しない場合は，タイムアウト時間を長くするのではなく，接続に関するほかの問題がないか調べてください。タイムアウト時間を長くすると，スクリプトが不必要に待機することになります。

［**ステップ ダウンロード タイムアウト（秒）**］：これは，仮想ユーザがスクリプトのステップの実行を中止するまでに待機する時間です。このオプションはページのダウンロードに特定秒数以上は待たないユーザの操作をエミュレートするのに使用します。

［**ネットワーク バッファ サイズ**］：HTTP 応答の受信に使用するバッファ・サイズの最大値を設定します。データのサイズが指定サイズより大きいと，サーバはデータをチャンクに分けてで送信するため，システムのオーバーヘッドが増加します。複数の仮想ユーザをコントローラで実行する場合，各仮想ユーザは自身のネットワーク・バッファを使用します。この設定は主として，ネットワーク・バッファ・サイズがスクリプトのパフォーマンスに影響を与える可能性があると判断した上級ユーザを対象にしています。標準設定は 12 キロバイトです。

［**認証再試行の固定遅延時間（ミリ秒）**］：ユーザによる認証情報（ユーザ名とパスワード）の入力をエミュレートするため，自動的に思考遅延時間を仮想ユーザ・スクリプトに追加します。この思考遅延時間はトランザクションに含まれます。標準設定は 0 です。

デバッグ情報の取得

仮想ユーザ・スクリプトを実行すると，実行情報が LoadRunner コントローラの［出力］ウィンドウに表示されます。［実行環境の設定］の［**ログ**］ノードを使って，出力ウィンドウとログ・ファイルに送られる情報量を制御できます（詳細については，137 ページ「実行環境設定のログの設定」を参照してください）。

デバッグ情報には，次のものがあります。

- ログ情報
- トランザクションの失敗
- ゲートウェイとの接続ステータス（接続中，切断中，リダイレクト中のいずれか）（WAP のみ）

より詳細なデバッグ情報を得るには，**default.cfg** ファイルを編集します。［**Web**］セクションで，**LogFileWrite** フラグを **1** に設定します。生成されるトレース・ファイルには，スクリプト実行時のすべてのイベントが記録されます。

負荷テストを実施する場合は，必ず **LogFileWrite** フラグをクリアして，LoadRunner が大きなトレース・ファイルを作成してリソースを浪費しないようにしてください。

HTML 圧縮の実行

HTTP 1.1 をサポートするブラウザは，圧縮されている HTML ファイルを展開できます。サーバは送信するファイルを圧縮して，データ転送に必要な帯域幅を節約します。

VuGen で圧縮を有効にするには，次の関数

```
web_add_auto_header("Accept-Encoding", "gzip");
```

をスクリプトの先頭に追加します。サーバによって圧縮データが送信されていることを検証するには，**Content-Encoding :gzip** という文字列が実行ログのサーバの応答セクションにあることを確認します。ログには展開前と後のデータ・サイズも表示されます。

圧縮は，大規模なデータ転送には大きな効果があります。つまり，データが大きければ大きいほど，圧縮が効果的になります。より大きなデータで作業するには，ネットワークのバッファ・サイズを増やして，データを 1 つのチャンクとして受け取るようにします（[ネットワーク バッファ サイズ] オプションを参照）。

第 38 章

Web ページの内容のチェック

仮想ユーザ・スクリプトを記録した後で，ページの内容をチェックするように実行環境を設定できます。

本章では，次の項目について説明します。

- 内容チェック実行環境の設定
- 内容チェック・ルールの定義

以降の情報は，Web 仮想ユーザ・スクリプトを対象とします。

Web ページの内容のチェックについて

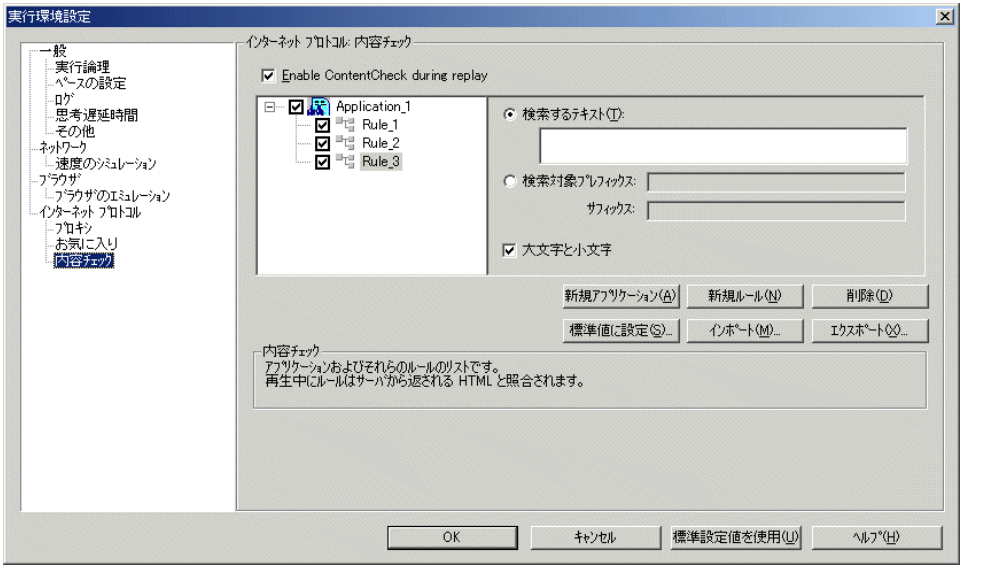
VuGen の内容チェックのメカニズムを使って，ページ内容の特定の文字列を検査することができます。このオプションは，標準的ではないエラーを検出するのに役立ちます。通常の運用では，アプリケーション・サーバで障害が発生するとブラウザが汎用の HTTP エラー・ページを表示して，エラーの性質を通知します。この標準のエラー・ページは VuGen に認識され，エラーとして処理されるので，スクリプトは失敗として報告されます。しかし，アプリケーション・サーバの中には，VuGen がエラー・ページとして認識しない固有のエラー・ページを発行するものもあります。エラー・ページはサーバによって送信され，エラーが発生したことを示す特定の形式のテキスト文字列を含んでいます。

たとえば，エラーが発生したときに，「**ASP Error**」というテキストを含んでいるユーザ定義ページを発行するアプリケーションがあるとします。その場合には，VuGen に対して，サーバから返されたすべてのページでこのテキストを検索するように指定します。VuGen はこの文字列を検出すると，再生を失敗とします。なお，VuGen はページ本体は検索しますが，ヘッダーは検索しません。

内容チェック実行環境の設定

［実行環境設定］の［**インターネット プロトコル : 内容チェック**］で，検索する文字列を指定します。複数の内容チェック・ルールを使って，複数のアプリケーションの内容を定義できます。以降では，次の項目について説明します。

- 内容チェック・ルールについて
- 内容チェック・ルールの定義



内容チェック・ルールについて

VuGen の内容チェックのメカニズムを使って，ページ・コンテンツの特定の文字列を検査することができます。このオプションは，標準的ではないエラーを検出するのに役立ちます。通常の運用では，アプリケーション・サーバで障害が発生するとブラウザが汎用の HTTP エラー・ページを表示して，エラーの性質を通知します。この標準のエラー・ページは VuGen に認識され，エラーとして処理されるので，スクリプトは失敗として報告されます。しかし，アプリケーション・サーバの中には，VuGen がエラー・ページとして認識しない固有のエラー・ページを発行するものもあります。エラー・ページはサーバによって送信され，エラーが発生したことを示す特定の形式のテキスト文字列を含んでいます。

たとえば，エラーが発生したときに，「**ASP Error**」というテキストを含んでいるユーザ定義ページを発行するアプリケーションがあるとします。その場合には，VuGen に対して，サーバから返されたすべてのページでこのテキストを検索するように指定します。VuGen はこの文字列を検出すると，再生を失敗とします。なお，VuGen はページ本体は検索しますが，ヘッダーは検索しません。

内容チェックの設定に対するグローバルな変更はサポートしていません。各グループの設定は，個別に編集してください。

［**再生中に内容チェックを有効にする**］：再生中の内容チェックを有効にします（標準で有効です）。なお，アプリケーションに対するルールを定義した後でも，このオプションを無効にすることで，テスト実行ごとにルールを無効にできます。

ルール情報

右の表示枠には，検索するテキストの検索条件が表示されます。検索するテキストそのもの，または対象テキストの前後にあるテキストを表すプレフィックスとサフィックスを指定できます。

［**検索するテキスト**］：検索するテキスト文字列を指定します。

［**検索対象プレフィックス / サフィックス**］：検索するテキスト文字列のプレフィックスとサフィックスを指定します。

［**大文字と小文字を区別する**］：検索時に大文字と小文字を区別します。

アプリケーションとルールの追加と削除

［**新規アプリケーション**］：左側の表示枠のアプリケーション・リストに新規アプリケーションを自動的に追加します。標準の名前は，Application_ **＜インデックス番号＞**で，最初は **Application_1** から始まります。新しいグループを作成したら，［**新規ルール**］をクリックしてグループにルールを追加します。アプリケーションの名前を変更するには，名前を選択してから名前をクリックします。

［**新規ルール**］：右側の表示枠にルール条件が表示され，現在選択されているアプリケーションの新しいルールを入力できるようになります。ルールはスクリプトとともに標準の XML ファイルに保存されます。このルール・ファイルをエクスポートすれば，ほかのユーザとの共有や，ほかのマシンでのインポートができます。

［**削除**］：選択したルールまたはアプリケーションを削除します。

ルールのインポートとエクスポート

［**インポート**］/［**エクスポート**］: ルール・ファイルをインポートまたはエクスポートします。アプリケーションとルールの指定は，**xml** という拡張子を持ったルール・ファイルに格納されます。ルールをファイルにエクスポートすることで他のマシンでも利用できるようになります。他のルール・ファイルをインポートすることも可能です。選択してインポートしたルールが既存のルールと矛盾する場合，矛盾するルールであることを示す警告が VuGen によって表示されます。その場合，既存のスクリプトに対して作成したルールを，インポートしようとしているルールと統合するか，現在のルールを上書きするように指定できます。［エクスポート］をクリックすると，［エクスポートするアプリケーションの選択］ダイアログ・ボックスが表示されます。

標準のルールの設定

［**標準値に設定**］: 内容チェックには，**インストール**，**標準設定**，**スクリプトごと**の 3 つのタイプがあります。「インストール」ルールは製品のインストール時に自動的に定められます。「標準設定」ルールはマシンで実行するすべてのスクリプトに適用されます。「スクリプトごと」のルールは，現在のスクリプトに対して定義されているものです。ルールを変更または追加したとき，その変更は現在のスクリプトにのみ適用されます。VuGen で，あるルールを標準設定ルールのリストに加えて，そのマシンのすべてのスクリプトに適用するには，［**標準値に設定**］をクリックします。

複数のスクリプトを使う場合や，製品のアップグレードを行う場合には，標準設定のルールとスクリプトごとのルールの間に矛盾が生じる場合があります。その場合，VuGen から，ルールを統合するかどうかの確認を求められます。ルールを統合すると（推奨），アプリケーションのルール・リストにルールが追加されます。

この操作は，左側の表示枠のアプリケーション・リストで有効になっているアプリケーションにのみ適用されます。現在のスクリプトで**有効**になっているアプリケーションがなければ，Defaults ファイルでも**有効**なアプリケーションはありません。既存の Defaults ファイルを上書きするには，［**はい**］をクリックします。操作を取り消して，既存の Defaults ファイルを維持する場合は，［**いいえ**］をクリックします。

ルールは標準の XML ファイルに保存されます。このルール・ファイルをエクスポートすれば，ほかのユーザとの共有や，ほかのマシンでのインポートができます。

［標準値に設定］をクリックして上書きを承認すると，VuGen によって次の処理が行われます。

1 Defaults ファイルの全アプリケーションに**無効**の印を付けます。

2 現在のスクリプトで**有効**になっているアプリケーションについて，Defaults ファイル内でのアプリケーションの有無に応じて，統合またはコピーを行います。アプリケーションが存在する場合，現在のスクリプトのルールが Defaults ファイルのルールに統合されます。アプリケーションが Defaults ファイルになければ，ルールは Defaults ファイルにそのままコピーされます。

3 スクリプトにおいて有効となっていたアプリケーションについて，Defaults ファイルでも**有効**の印を付けます。現在のスクリプトで**有効**になっているアプリケーションがなければ，Defaults ファイルでも**有効**の印が付くアプリケーションはありません。

標準設定値を使用

Defaults ファイルからルールをインポートします。このボタンをクリックすると，アプリケーションとその標準設定のリストを示すダイアログ・ボックスが表示されます。このダイアログ・ボックスで，これらのルールインポートするか，変更するかを選択できます。このルールが既存のルールと矛盾する場合，矛盾するルールであることを示す警告が VuGen によって表示されます。また，標準設定ファイルで定義されているルールを現在有効なルールに統合することもできます。

アプリケーションのすべての標準設定を使用するには［**標準設定値を使用**］をクリックします。これにより，標準設定ファイルから定義がインポートされます。このとき，アプリケーションとその標準設定のリストを示すダイアログ・ボックスが表示されます。このダイアログ・ボックスで，これらの定義をインポートするか，変更するかを選択できます。このルールが他のルールと矛盾する場合，VuGen から矛盾するルールであることを示す警告が表示されます。また，標準設定ファイルで定義されているルールを現在有効なルールに統合することや，現在有効なルールで上書きすることもできます。

内容チェック・ルールの定義

［実行環境設定］の［**インターネット プロトコル : 内容チェック**］を使って，Web ページの内容を検査するルールを定義します。

内容チェック・ルールの定義は，次の手順で行います。

1 ［実行環境設定］ダイアログ・ボックスを開き，［**インターネット プロトコル : 内容チェック**］ノードを選択します。

2 ［**再生中に内容チェックを有効にする**］オプションを選択します。

3 ［**新規アプリケーション**］ボタンをクリックして，内容を検査するアプリケーションのリストに新しいエントリを追加します。

4 既存のアプリケーションに対するルールを追加するには，［**新規ルール**］をクリックします。各アプリケーション・サーバに 1 つまたは複数のルールを適用できます。左側の表示枠でルールの横のチェック・ボックスのオン / オフを切り替えて，ルールの有効 / 無効を切り替えます。

5 実際のテキスト文字列を検索するには，［**検索するテキスト**］を選択して，検索したいテキストを入力します。テキストは，可能な限り具体的にすることをお勧めします。たとえば，「**エラー**」ではなく「**ASP エラー**」のように，アプリケーション固有のテキストを入力するようにします。

6 文字列の前後に付くテキストに基づいて検索するには，［**検索対象プレフィックス / サフィックス**］を選択し，前置記号と後置記号を指定します。

7 大文字と小文字を区別して指定するには，［**大文字と小文字を区別する**］を選択します。

8 ルールが，マシンに格納されているすべてのスクリプトに適用されるよう標準として設定するには，ルールまたはアプリケーションを選択して［**標準値に設定**］をクリックします。

9 ルールのファイルをエクスポートするには，［**エクスポート**］をクリックして，保存場所を指定します。

10 ルールのファイルをインポートするには，［**インポート**］をクリックし，ファイルの場所を探します。

11 アプリケーションやルールを削除するには，ルールを選択して［**削除**］ボタンをクリックします。

12 アプリケーション全体に標準の設定を使用するには，［**標準設定値を使用**］をクリックします。アプリケーションの一覧と標準の設定を含むダイアログ・ボックスが表示されます。矛盾がある場合は，ルールを上書きまたは統合できます。

第 39 章

負荷下の Web ページ検証

Web 仮想ユーザ・スクリプトに Web チェックを追加できます。これを使って，仮想ユーザ・スクリプトを実行したときにサーバが正しい Web ページを返すかどうか判定します。

本章では，次の項目について説明します。

- テキスト・チェックの追加
- その他のテキスト・チェック・メソッドの使用
- 画像チェックの追加
- 追加プロパティの定義
- 正規表現の使用

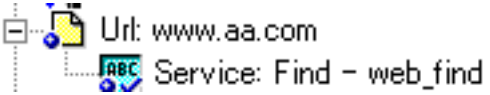
以降の情報は，Web 仮想ユーザ・スクリプトを対象とします。

負荷下の検証について

VuGen を使って Web 仮想ユーザ・スクリプトに Web チェックを追加できます。Web チェックで，Web ページに特定のオブジェクトがあるかどうかを検証します。ここでいうオブジェクトは，テキスト文字列，画像，または Java アプレットです。

Web チェックによって，多数の仮想ユーザがアクセスしているときに Web サイトが正しく機能するか，つまりサーバが正しい Web ページを返すかどうかを確認できます。これが特に重要なのは，サイトに多数のユーザの負荷がかかるときです。大きな負荷がかかった状態では，サーバが間違ったページを返す可能性が高くなるからです。

たとえば，世界の主要都市の気温に関する情報を表示する Web サイトがあるとします。VuGen を使って，その Web サイトにアクセスする仮想ユーザ・スクリプトを作成します。次のスクリプト・セグメントで，仮想ユーザはサイト **www.aa.com** のハイパーテキスト・リンクにアクセスします。



ブラウザはこの URL のページを表示します。仮想ユーザは，この Web ページのテキストをチェックします。たとえば，ページ内に「**Specials**」という単語が表示されれば，チェックは合格です。サーバから正しいページが返されなかったために，「**Specials**」という単語が表示されなければ，チェックは不合格になります。

スクリプトを記録したときや，単独の仮想ユーザでスクリプトを実行したときには，サーバから正しいページを返されていたとしても，多数の仮想ユーザでサーバに負荷をかけた場合には，正しいページが返されないことがあります。サーバが過負荷になり，そのために無意味な，あるいは不正な HTML コードを返すことがあります。また，過負荷になったサーバが「**500 Server Error**」ページを返すこともあります。どちらの場合も，サーバから正しいページが返されたかどうかを判定するチェックを挿入できます。

注：Web チェックによって仮想ユーザの処理が増えるので，ロード・ジェネレータごとの仮想ユーザ数を減らす必要が生じることがあります。Web チェックは，実際にサーバから不正確なページが返されたことがある場合に限って使うことをお勧めします。

Web チェックは，仮想ユーザ・スクリプトの記録中または記録後に定義できます。一般的には，チェック対象の Web ページが表示される記録中にチェックを定義するほうが手間がかかりません。

Web チェックを追加すると，VuGen はツリー・ビューの中で，仮想ユーザ・スクリプトの現在のステップに Web チェック・アイコンを追加します。Web チェック・アイコンは，常に関連ステップのすぐ下に字下げされて表示されます。仮想ユーザ・スクリプトを実行すると，VuGen はステップ実行後に表示される Web ページのチェックを実行します。

注：仮想ユーザが Web チェックを実行するのは，チェックが有効になっているスクリプトの実行中で，スクリプトが HTML モードで実行されているときに限られます。チェックを有効にするには，［実行環境設定］ダイアログ・ボックスの［**お気に入り**］ノードで［**画像とテキストチェックを有効にする**］オプションを選択します。詳細については，第 9 章 「実行環境の設定」を参照してください。

VuGen は，いくつかの種類の Web チェック・アイコンを使用し，それぞれのアイコンは別のタイプのチェックを表します。

Web チェック・アイコン	**説明**
テキスト 	テキスト・チェックは，Web ページ内で，指定されたテキスト文字列を検索します。
画像 	画像チェックは，Web ページ内で，指定された画像を検索します。

本章では，VuGen を使ってツリー・ビューに Web チェックを追加する方法を説明します。テキスト・ベースのスクリプト・ビューでのスクリプトへのチェックの追加については，「**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。

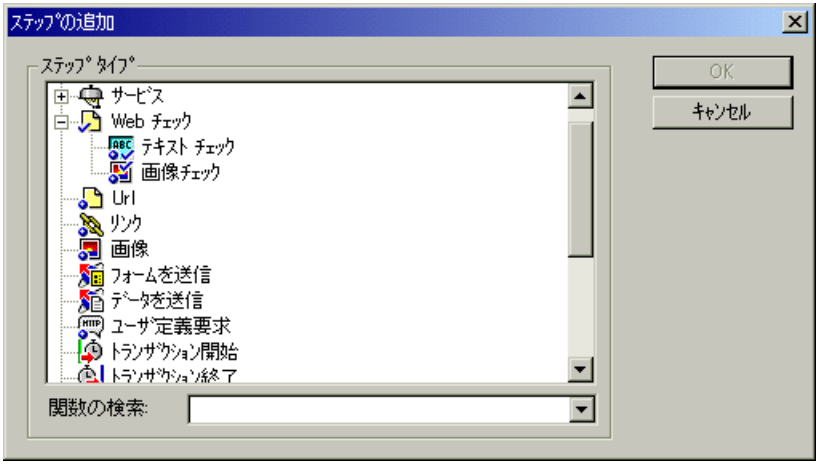
テキスト・チェックの追加

VuGen で，Web ページのテキスト文字列を検索するチェックを追加できます。テキスト・チェックは記録中または記録後に追加できます。

テキスト・チェックを作成するときに，チェックの名前，チェックの範囲，チェックするテキスト，および検索条件を定義します。

記録後のテキスト・チェックの追加は，次の手順で行います。

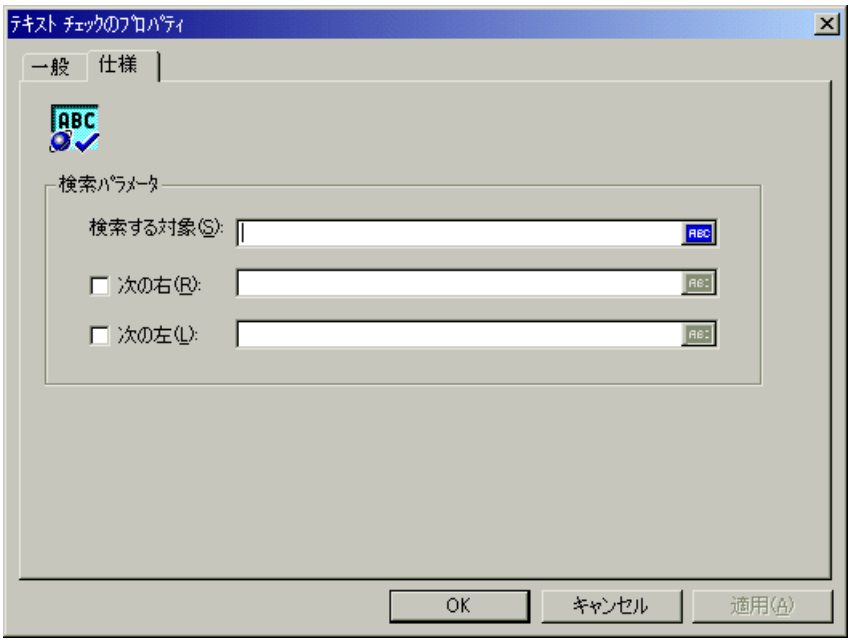
1 VuGen のメイン・ウィンドウで，チェックを実行する Web ページに対応するステップを右クリックします。ポップアップ・メニューから［**後に挿入**］を選択します。［ステップの追加］ダイアログ・ボックスが表示されます。



注：Web ブラウザの記録セッション中は，VuGen のメイン・ウィンドウは最小化されます。記録中にテキスト・チェックを追加するには，VuGen のメイン・ウィンドウを開きます。

2［**ステップ タイプ**］ツリーで［**Web チェック**］を展開します。

3 ［**テキスト チェック**］を選択して，［**OK**］をクリックします。［テキスト チェックのプロパティ］ダイアログ・ボックスが表示されます。［**仕様**］タブが表示されていることを確認します。



4 ［**検索する対象**］ボックスに，検証する文字列を入力します。［ABC］アイコンは，［**検索する対象**］ボックスの文字列がパラメータを割り当てられていないことを示します。パラメータの指定については，第 7 章 「パラメータの定義」を参照してください。

5 隣接するテキストを基準に検索文字列の位置を指定するには，［**次の右**］または［**次の左**］チェック・ボックスを選択します。その後，適切なフィールドにテキストを入力します。たとえば，「support@mercuryinteractive.com」という文字列が「e-mail:」という単語の右に表示されることを検証する際，［**次の右**］を選択して，［**次の右**］ボックスに「e-mail:」と入力します。

［ABC］アイコンは，［**次の右**］ボックスや［**次の左**］ボックスの文字列がパラメータを割り当てられていないことを示します。パラメータの指定については，第 7 章 「パラメータの定義」を参照してください。

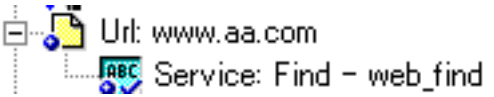
6 テキスト・チェックに名前を付けます。[**一般**] タブをクリックし，[**ステップ名**] ボックスに，テキスト・チェックの名前を入力します。後で識別しやすいような名前を付けます。

7 プロパティ・テーブルにチェックを定義する追加プロパティが表示されます。

[**アクティブなプロパティのみを表示する**] チェック・ボックスをクリアし，有効なプロパティと無効になっているプロパティの両方を表示します。プロパティを有効にするには，プロパティ名の左のセルをクリックします。[**値**] カラムでプロパティの値を指定します。

プロパティ値の指定の詳細については，521 ページ「追加プロパティの定義」を参照してください。

8 [**OK**] をクリックして設定を受け入れます。新しい [テキスト チェック] アイコンが，スクリプトの関連ステップに追加されます。スクリプト・ビューで [テキスト チェック] アイコンが **web find** 関数として表示されます。



記録中のテキスト・チェックの追加は，次の手順で行います。

1 マウスを使用して，対象テキストをマークします。

2 記録ツールバーの［**テキスト チェックを挿入**］アイコンをクリックします。

その他のテキスト・チェック・メソッドの使用

web_find 関数以外に，次の 2 つの拡張関数を使用して HTML ページ内のテキストを検索できます。

- web_reg_find
- web_global_verification

web_reg_find 関数は，登録タイプの関数です。HTML ページ上でのテキスト文字列の検索を登録します。登録とは，直ちに検索を実行しないで，**web_url** など，次の Action 関数の実行後にだけチェックを実行することを意味します。同時実行関数グループで作業する場合，**web_reg_find** 関数はグループ化の後にだけ実行されます。

この関数は，HTML ベースのスクリプトだけに制限されない点が，**web_find** 関数とは異なります（［**記録オプション**］＞［**記録**］ノードを参照）。この関数にはインスタンスなどの追加の属性もあり，テキストの出現回数を決めることができます。標準のテキスト検索を実行する場合には，**web_reg_find** 関数のほうが便利です。

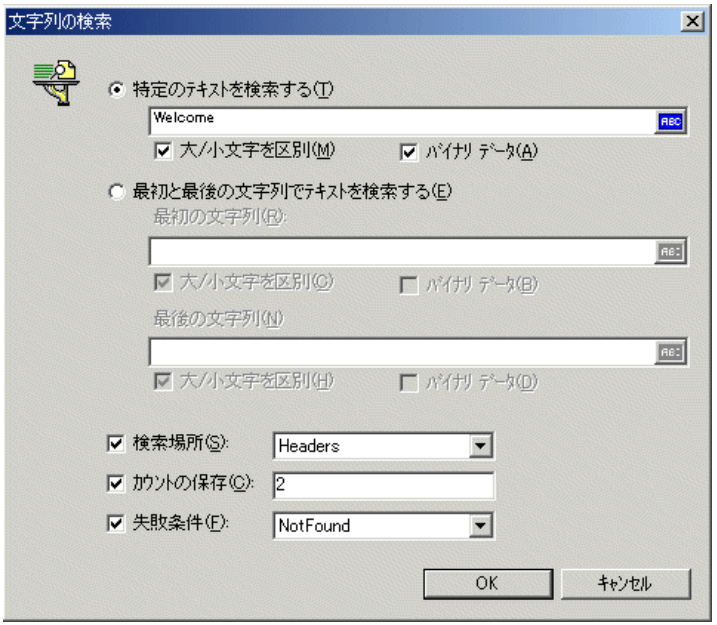
web_reg_find では次の属性を使用できます。

- ［**特定のテキストを検索する**］: 検索するテキスト文字列です。この属性は空でない，NULL 終端文字列である必要があります。この検索メカニズムは大文字と小文字を区別します。大文字と小文字を区別しない場合は，境界の後に「/IC」を追加します。バイナリ・データを指定するには，テキストの後に，「/BIN」を指定します（あるいは，ステップのプロパティにある［**バイナリ データ**］チェック・ボックスを選択します）。「Text=string」の形式を使用します。

［**特定のテキストを検索する**］を指定する代わりに，次の 2 つの属性を指定することもできます。

- ［**最初の文字列**］：検索するテキスト文字列の接頭辞です。大文字と小文字を区別しないようにするには，境界の後に「/IC」を追加します。バイナリ・データを指定するには，境界の後に「/BIN」を指定します。「TextPfx=string」の形式を使用します。
- ［**最後の文字列**］：検索するテキスト文字列の接尾辞です。大文字と小文字を区別しないようにするには，境界の後に「/IC」を追加します。バイナリ・データを指定するには，境界の後に「/BIN」を指定します。「TextSfx=string」の形式を使用します。
- ［**検索場所**］：テキストを検索する対象です。利用できる値は，Headers, Body，NORESOUCE または All です。標準設定は Body です。「Search=value」の形式を使用します（任意）。
- ［**カウントの保存**］：検出された一致の数です。この属性を使用するには，SaveCount=param_name を指定します。param_name は NULL 終端の ASCII 値を保存する変数です（任意）。
- ［**失敗条件**］：文字列が検出されない場合の処理メソッドです。使用可能なメソッドは，**Found**，**NotFound**，および **None** です。**Found** は，テキストが検出されたときにエラーが発生することを示します（「Error」など）。**Not Found** は，テキストが検出されないときにエラーが発生することを示します。［**カウントの保存**］を指定する場合，標準設定は「None-no failure」です。［**カウントの保存**］が省略される場合，標準設定は「NotFound」です。［**失敗条件**］に値「None」を明示的に割り当てることはできません。

また，［文字列の検索］ダイアログ・ボックスから上記の属性を設定することもできます。ツリー・ビュー内で関数を選択して，右クリックで表示されるメニューから［**プロパティ**］を選択します。すべての引数値を入力できるダイアログ・ボックスが表示されます。



次の例では，**web_reg_find** 関数がテキスト文字列「Welcome」を検索します。文字列が検出されない場合はエラーが発生し，スクリプトの実行が停止します。

```
web_reg_find("Text=Welcome", "Fail=Found", LAST);
web_url("Step", "URL=...", LAST);
```

web_global_verification 関数によって，ビジネス・プロセス全体のデータを検索できます。次のアクション関数のみに適用される **web_reg_find** とは対照的に，この関数は **web_url** など，後続の**すべての**アクション関数に適用されます。標準設定では，検索範囲は「NORESOURCE」で，ヘッダーとリソースを除いた HTML 本体のみを検索します。**web_global_verification** 関数は，HTTP ステータス・コードに含まれないアプリケーション・レベルのエラーの検出に最適です。この関数は HTML ベースのスクリプトだけに制限されません。詳細については，［**記録オプション**］＞［**記録**］ノードを参照してください。

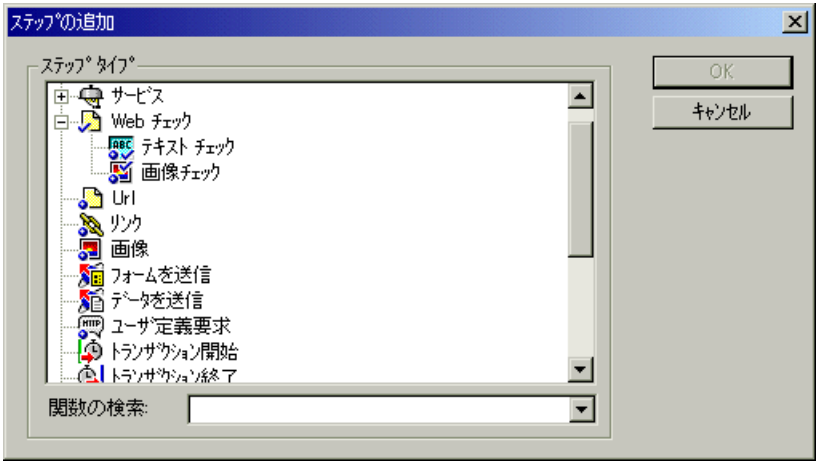
画像チェックの追加

VuGen を使って Web ページ内の画像を検索するユーザ定義チェックを追加できます。画像は ALT 属性と SRC 属性のどちらかまたは両方によって識別できます。

記録中または記録後にユーザ定義の画像チェックを追加できます。記録中に追加された画像チェックは，記録後に編集できます。

画像チェックの追加は，次の手順で行います。

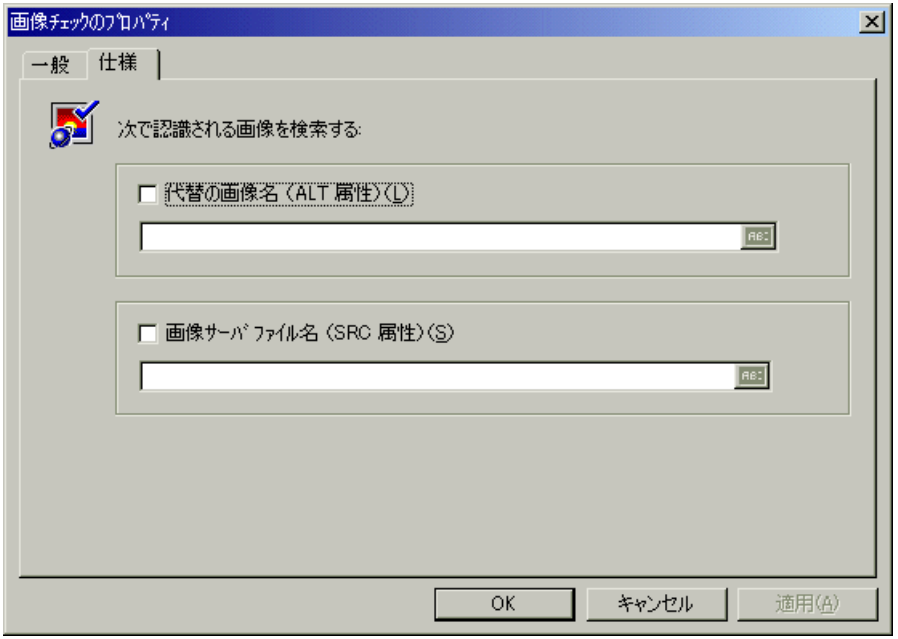
1 VuGen のメイン・ウィンドウで，チェックを実行する Web ページに対応するステップを右クリックします。ポップアップ・メニューから［**後に挿入**］を選択します。［ステップの追加］ダイアログ・ボックスが表示されます。



注：Web ブラウザの記録セッション中は，VuGen のメイン・ウィンドウは最小化されます。記録中に画像チェックを追加するには，VuGen のメイン・ウィンドウを開きます。

2［**ステップ タイプ**］ツリーで［**Web チェック**］を展開します。

3 ［**画像チェック**］を選択して［**OK**］をクリックします。［画像チェックのプロパティ］ダイアログ・ボックスが表示されます。［**仕様**］タブが表示されていることを確認します。



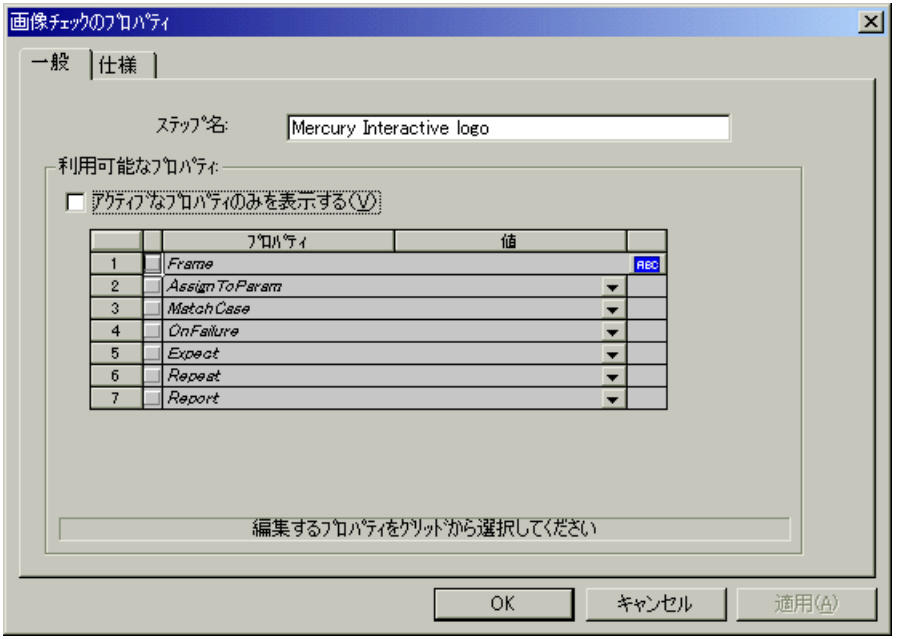
4 画像を識別するメソッドを選択します。

- ALT 属性を使って画像を識別する場合，［**代替の画像名（ALT 属性)**］チェック・ボックスを選択して ALT 属性を入力します。スクリプトを実行すると，仮想ユーザは指定された ALT 属性を持つ画像を検索します。
- SRC 属性を使って画像を識別する場合，［**画像サーバ ファイル名（SRC 属性)**］チェック・ボックスを選択して SRC 属性を入力します。スクリプトを実行すると，仮想ユーザは指定された SRC 属性を持つ画像を検索します。

［ABC］アイコンは，ALT 属性または SRC 属性にパラメータが指定されていないことを示します。パラメータの指定については，第 7 章 「パラメータの定義」を参照してください。

注：［ALT 属性］チェック・ボックスと［SRC 属性］チェック・ボックスを両方選択した場合，仮想ユーザは指定された ALT 属性と指定された SRC 属性の両方を持つ画像を検索します。

5 画像チェックに名前を付けるには，［**一般**］タブをクリックします。［**ステップ名**］ボックスに，画像チェックの名前を入力します。後で識別しやすい名前を付けます。

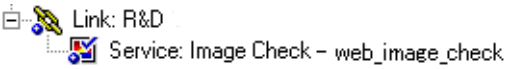


6 プロパティ・テーブルにチェックを定義する追加プロパティが表示されます。

［**アクティブなプロパティのみを表示する**］チェック・ボックスをクリアし，有効なプロパティと無効になっているプロパティの両方を表示します。プロパティを有効にするには，プロパティ名の左のセルをクリックします。［**値**］カラムでプロパティの値を指定します。

プロパティ値の指定の詳細については，521 ページ「追加プロパティの定義」を参照してください。

7 [**OK**] をクリックして設定を受け入れます。新しい [画像チェック] アイコンが，仮想ユーザ・スクリプトの関連ステップに追加されます。



追加プロパティの定義

仮想ユーザ・スクリプトに挿入する各 Web 検査の追加のプロパティを指定できます。チェック・プロパティの [**一般**] タブにあるプロパティ・テーブルで追加オプションを設定します。

追加プロパティの設定は，次の手順で行います。

1 プロパティを編集する Web チェックを右クリックして，ポップアップ・メニューから [**プロパティ**] を選択します。該当するチェックのプロパティのダイアログ・ボックスが表示されます。[**一般**] タブが表示されていることを確認します。

2 [**アクティブなプロパティのみを表示する**] チェック・ボックスをクリアして，利用可能なすべてのプロパティを表示します。

3 プロパティを有効にするには，プロパティ名の左のセルをクリックします。プロパティの横に赤いチェック・マークが表示されます。

4 [**値**] カラムでプロパティの値を指定します。

- [**Frame**]：検査オブジェクトがあるフレーム名を入力します。
- [**AssignToParm**]：[**YES**] を選択すると，パラメータへの代入が有効になります。[**NO**] を選択するとこの機能は無効になります。標準設定の値は [**NO**] です。
- [**MatchCase**]：[**YES**] を選択すると，大文字と小文字を区別して検索します。[**NO**] を選択すると，大文字と小文字を区別しないで検索します。標準設定の値は [**NO**] です。
- [**OnFailure**]：[**Abort**] を選択すると，検査が失敗した場合に仮想ユーザ・スクリプトをすべて中止します。VuGen は，実行環境の設定で指定されているエラー処理方法にかかわらず，仮想ユーザ・スクリプトを中止します。[**Continue**] を選択すると，検査に失敗したスクリプトを中止するかどうかは，実行環境の設定で定義されているエラー処理方法に従います。標準設定

の値は［**Continue**］です。エラー処理方法の定義の詳細については，第 9 章「実行環境の設定」を参照してください。

- ［**Expect**］:［**NotFound**］を選択すると，仮想ユーザが指定した検査オブジェクトを見つけられない場合に，検査に合格したことになります。［**Found**］を選択すると，仮想ユーザが指定した検査オブジェクトを見つけた場合に，検査に合格したことになります。標準設定の値は［**Found**］です。
- ［**Repeat**］:［**YES**］を選択すると，選択された検査オブジェクトを複数検索します。指定した検査オブジェクトが 1 つ見つかった時点で検査を終了するには，［**NO**］を選択します。仮想ユーザ・スクリプトはスクリプト内の次のステップから実行を続けます。このオプションは，検査オブジェクトが複数含まれる可能性のある Web ページを検索するときに役立ちます。標準設定の値は［**YES**］です。
- ［**Report**］:［**Always**］を選択すると，実行ログで検査結果の詳細を常に表示できます。［**Failure**］を選択すると，検査に不合格だった場合にのみ検査結果の詳細を表示します。［**Success**］を選択すると，検査に合格した場合にのみ検査結果の詳細を表示します。標準設定の値は［**Always**］です。

［ABC］アイコンは，プロパティの値がパラメータを割り当てられていないことを示します。アイコンをクリックしてパラメータを割り当てます。詳細については，第 7 章 「パラメータの定義」を参照してください。

正規表現の使用

テキスト・チェックを追加する際，値のタイプを「**正規表現**」として指定できます。正規表現を使用すると，チェックの柔軟性が向上します。

正規表現では，下記に示した特殊文字以外の文字は，表記通りに検索されます。特殊文字の直前にバックスラッシュまたは円記号（¥）がある場合，仮想ユーザは実際の文字のみを検索します。

正規表現を作成するときに使用できるオプションは次のとおりです。

- 任意の 1 文字と一致
- 範囲内の任意の 1 文字と一致
- 特定の文字の繰り返しと一致

任意の 1 文字と一致

疑問符（?）は，任意の 1 文字を検索するように VuGen に指示します。次の例を示します。

```
welcome?
```

たとえば，「welcome?」は「welcomes」，「welcomed」など，「welcome」の後に 1 つの空白文字またはその他の 1 文字が続く文字列が一致します。疑問符の連続は，不特定の文字の連続を示します。連続する文字数は連続する疑問符の数によって決まります。

範囲内の任意の 1 文字と一致

特定の範囲内の任意の 1 文字と一致させるには，角括弧（[ ]）を使用します。たとえば，1968 年か 1969 年のどちらかを検索する場合は，次のように表記します。

```
196[89]
```

範囲を示すには，ハイフン（-）を使用します。たとえば，1960 年代のすべての年と一致させるには，次のように指定します。

```
196[0-9]
```

ハイフンが括弧内の最初か最後，またはキャレット（^）の後にある場合には，範囲の指定としては解釈されません。

キャレットは，文字列で指定した文字以外の文字と一致させる場合に指定します。次に例を示します。

```
[^A-Za-z]
```

これは，アルファベット以外の文字と一致します。キャレットは，括弧内の最初の文字として使用された場合にのみ，このような特別な意味を持ちます。

大括弧内で，次の文字は実際の文字として認識されます。

ピリオド（.）
アスタリスク（*）
左大括弧（[）
バックスラッシュまたは円記号（¥）

括弧内の先頭にある右大括弧は，リテラル文字として認識されます。次に例を示します。

```
[]g-m]
```

この場合は，] と，g から m が一致します。

特定の文字の繰り返しと一致

アスタリスク（*）は直前の文字の 0 回以上の繰り返しと一致します。次に例を示します。

```
Q*
```

これは，Q，QQ，QQQ などと一致します。

第 40 章

Web とワイヤレス仮想ユーザ・スクリプトの変更

Web またはワイヤレス仮想ユーザ・スクリプトの記録が終わったら，VuGen を使って，記録したスクリプトを変更できます。新規ステップの追加のほか，既存のステップの編集，名前の変更，削除ができます。

本章では，次の項目について説明します。

- 仮想ユーザ・スクリプトへのステップの追加
- 仮想ユーザ・スクリプトからのステップの削除
- アクション・ステップの変更
- 制御ステップの変更
- 「サービス」ステップの変更
- Web チェックの変更（Web のみ）

以降の情報は，Web 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。

Web およびワイヤレス仮想ユーザ・スクリプトの変更について

ブラウザ・セッションまたはツールキット・セッションの記録後，ステップのプロパティを編集したり，ステップを追加，削除して，記録したスクリプトを VuGen で変更できます。

変更は，アイコン・ベースのツリー・ビューまたはテキスト・ベースのスクリプト・ビューのどちらでも行えます。2 つの表示モードの詳細については，第 32 章「Web 仮想ユーザ・スクリプトの作成」を参照してください。

本章では，VuGen を使用してツリー・ビューでスクリプトを変更する方法について説明します。テキスト・ベースのスクリプト・ビューでのスクリプトの変更については，「**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。

バイナリ・データの追加

バイナリ・コード・データを HTTP 要求の本体に含めるには，次の形式に従います。

¥x[char1][char2]

これは，[char1][char2] によって表される 16 進値です。

たとえば，¥x24 は 10 進数で表せば 16*2+4=36 となり，対応する文字は $ 記号です。同様に，¥x2B は + 記号です。

16 進法のシーケンスとして 1 桁のシーケンスは使用しないでください。たとえば，¥x2 は有効なシーケンスではありませんが，¥x02 は有効なシーケンスです。

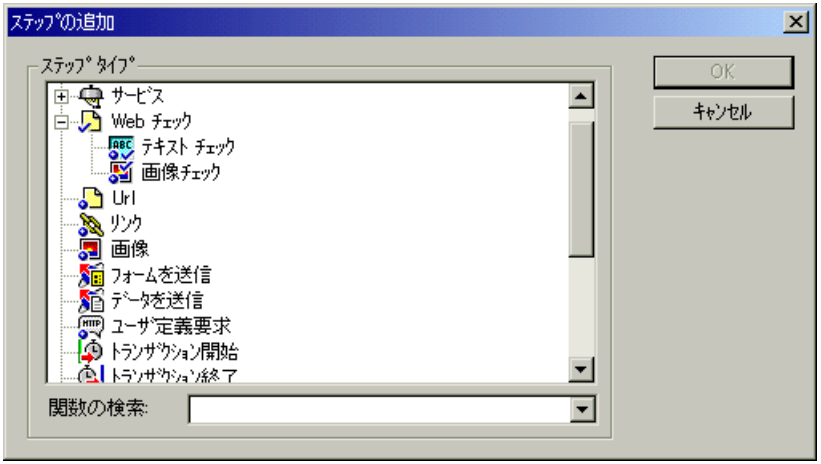
仮想ユーザ・スクリプトへのステップの追加

Web ブラウザまたはツールキットの記録セッション中に VuGen が記録するステップに加え，記録されたスクリプトにステップを追加することもできます。

仮想ユーザ・スクリプトにステップを追加するには，次の手順で行います。

1 仮想ユーザ・スクリプトのツリー・ビューで，新しいステップを追加する位置の前または後のステップを選択します。

2 [**挿入**] > [**新規ステップ**] を選んで，選択したステップの後にステップを挿入するか，右クリックして表示されるメニューから [**前に挿入**] または [**後に挿入**] を選択します。[ステップの追加] ダイアログ・ボックスが開きます。



3 [**ステップ タイプ**] ツリーまたは [**関数の検索**] リストから，追加するステップの種類を選択します。

4 [**OK**] をクリックします。追加するステップに関する情報を入力するダイアログ・ボックスが開きます。このダイアログ・ボックスは，追加するステップの種類に応じて異なります。

このダイアログ・ボックスの使い方の詳細については，以下に示す各項を参照してください。

追加項目	参照先
LoadRunner 関数	第 6 章「仮想ユーザ・スクリプトの拡張」
サービス・ステップ	547 ページ「「サービス」ステップの変更」
Web チェック	548 ページ「Web チェックの変更（Web のみ）」
トランザクション	544 ページ「トランザクションの変更」
ランデブー・ポイント	546 ページ「ランデブー・ポイントの変更」
「思考遅延時間」ステップ	546 ページ「思考遅延時間の変更」
「URL」ステップ	529 ページ「「URL」ステップの変更」
「リンク」ステップ	531 ページ「ハイパーテキスト・リンク・ステップの変更（Web のみ）」
「画像」ステップ	533 ページ「「画像」ステップの変更（Web のみ）」
「フォームを送信」ステップ	535 ページ「「フォームを送信」ステップの変更（Web のみ）」
「データを送信」ステップ	538 ページ「「データを送信」ステップの変更」
「ユーザ定義要求」ステップ	541 ページ「「ユーザ定義要求」ステップの変更」
「ユーザ定義」ステップ	第 6 章「仮想ユーザ・スクリプトの拡張」

仮想ユーザ・スクリプトからのステップの削除

ブラウザ・セッションまたはツールキット・セッションの記録後に，VuGen を使って仮想ユーザ・スクリプトからステップを削除できます。

仮想ユーザ・スクリプトからステップを削除するには，次の手順で行います。

1 仮想ユーザ・スクリプトのツリー・ビューで，削除するステップを右クリックして，ポップアップ・メニューから［**削除**］を選択します。

2 ［**OK**］をクリックして，このステップの削除を確認します。

ステップがスクリプトから削除されます。

アクション・ステップの変更

アクション・ステップは，記録中のユーザ・アクションを表します。つまり，新しい URL へのジャンプまたは Web コンテキストの変更などを表します。

仮想ユーザ・スクリプトのツリー・ビューにアクション・アイコンとして表示されているアクション・ステップは，記録中に自動的にスクリプトに追加されます。記録後，記録されたアクション・ステップを変更できます。

本項では，以下について説明します。

- 「URL」ステップの変更
- ハイパーテキスト・リンク・ステップの変更（Web のみ）
- 「画像」ステップの変更（Web のみ）
- 「フォームを送信」ステップの変更（Web のみ）
- 「データを送信」ステップの変更
- 「ユーザ定義要求」ステップの変更

「URL」ステップの変更

URL を入力したり，特定の Web ページにアクセスするブックマークを使用したりすると，仮想ユーザ・スクリプトに「URL」ステップが追加されます。

変更できるプロパティは，ステップ名，URL のアドレス，ターゲット・フレーム，記録モードです。

標準では，VuGen は記録時のモードに基づいて「URL」ステップを実行します。記録モードには，**HTML**，または **HTTP**（リソースを含まないモード）があります。記録モードの詳細については，475 ページ「記録レベルの選択」を参照してください。

再生モードの設定

LoadRunner によって，スクリプトを記録時のモードとは異なるモードで実行するには，［URL ステップのプロパティ］ダイアログ・ボックスでモードの設定を変更します。再生モードを変更するには，［**記録モード**］チェック・ボックスを選択します。以下に示す再生モードを使用できます。

HTML：自動的にすべてのリソースと画像をダウンロードし，以降のステップに必要な HTTP 情報を格納します。このモードは，Web のリンクが含まれるスクリプトに適しています。

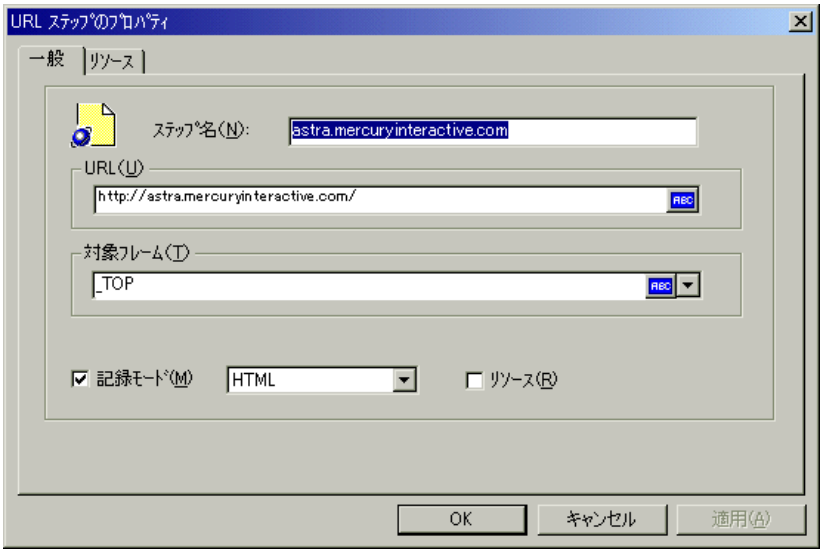
HTTP：再生時に，このステップのためにリソースをダウンロードしません。関数によって明示的に指定されたリソースだけをダウンロードします。

特定のステップをリソースとして見なさないようにすることもできます。たとえば，省略する必要のある特定の画像を表すステップがある場合，その種類のリソースを含めないように設定できます。詳細については，482 ページ「リソースの処理」を参照してください。

「URL」ステップのプロパティを変更するには，次の手順で行います。

1 仮想ユーザ・スクリプトのツリー・ビューで，編集する「URL」ステップを選択します。「URL」ステップは［URL］アイコンで表示されます。

2 VuGen ツールバーの［**プロパティ**］ボタンをクリックします。［URL ステップのプロパティ］ダイアログ・ボックスが開きます。



3 ステップ名を変更するには，［**ステップ名**］ボックスに新しい名前を入力します。記録中は，URL の最後の部分が標準名となります。

4 ［**URL**］ボックスに，「URL」ステップがアクセスした Web ページの Web アドレス（URL）を入力します。［**ABC**］アイコンは，この URL にパラメータが割り当てられていないことを示します。パラメータの割り当て方法の詳細については，第 7 章「パラメータの定義」を参照してください。

5 ［**対象フレーム**］リストで，次の値から 1 つを選択します。

［**_TOP**］：ページ全体を置き換えます。

［**_BLANK**］：新規ウィンドウを開きます。

［**_PARENT**］：最後の（変更された）フレームの親を置き換えます。

6 再生モードを変更するには，［**記録モード**］チェック・ボックスを選択します。

次の中から使用するモードを選択します：HTML または HTTP。

7 項目をリソースとしてダウンロードしないようにするには，［**リソース**］チェック・ボックスをクリアします。

8 ［**OK**］をクリックして，［URL ステップのプロパティ］ダイアログ・ボックスを閉じます。

ハイパーテキスト・リンク・ステップの変更（Web のみ）

［リンク］ステップは，ハイパーテキスト・リンクをクリックすると，仮想ユーザ・スクリプトに追加されます。このステップは，**HTML** モードで記録するためのオプションを選択した場合にのみ記録されます。詳細については，第 36 章「Web 仮想ユーザの記録オプションの設定」を参照してください。

変更できるプロパティは，ステップ名，ハイパーテキスト・リンクの識別方法，配置場所です。

ハイパーテキスト・リンク・ステップのプロパティを変更するには，次の手順で行います。

1 仮想ユーザ・スクリプトのツリー・ビューで，編集する「リンク」ステップを選択します。「リンク」ステップは［リンク］アイコンで表示されます。

2 右クリックして表示されるメニューから［**プロパティ**］を選択します。［リンク ステップのプロパティ］ダイアログ・ボックスが開きます。

3 ステップ名を変更するには，［**ステップ名**］ボックスに新しい名前を入力します。記録中は，ハイパーテキスト・リンクのテキスト文字列が標準名となります。

4 プロパティ・テーブルに，リンクを識別するプロパティが表示されます。

［**アクティブなプロパティのみを表示する**］チェック・ボックスをクリアし，アクティブなプロパティと非アクティブなプロパティの両方を表示します。プロパティを有効にするには，プロパティ名の左のセルをクリックします。［**値**］カラムでプロパティの値を割り当てます。

- ［**Text**］：ハイパーテキスト・リンクの正確な文字列。
- ［**Frame**］：リンクが属しているフレームの名前。
- ［**TargetFrame**］：以下のターゲット・フレーム。

  ［**_TOP**］：ページ全体を置き換えます。

  ［**_BLANK**］：新規ウィンドウを開きます。

  ［**_PARENT**］：最後の（変更された）フレームの親を置き換えます。

➤ ［**Ordinal**］：他のすべての属性が，同じ Web ページ上にある他のリンク（1 つまたは複数）と同じときに，リンクを一意に識別する番号。詳細については，「**オンライン関数リファレンス**」を参照してください。

［ABC］アイコンは，プロパティの値にパラメータが割り当てられていないことを示します。パラメータの割り当て方法の詳細については，第 7 章「パラメータの定義」を参照してください。

5 ［**OK**］をクリックして，［リンク ステップのプロパティ］ダイアログ・ボックスを閉じます。

「画像」ステップの変更（Web のみ）

ハイパーグラフィック・リンクをクリックすると，「画像」ステップが仮想ユーザ・スクリプトに追加されます。このステップは，HTML（コンテキスト・センシティブ）モードで記録するためのオプションを選択した場合にのみ記録されます。詳細については，第 36 章「Web 仮想ユーザの記録オプションの設定」を参照してください。

変更できるプロパティは，ステップ名，ハイパーグラフィック・リンクの識別方法，配置場所です。

「画像」ステップのプロパティを変更するには，次の手順で行います。

1 仮想ユーザ・スクリプトのツリー・ビューで，編集する「画像」ステップを選択します。「画像」ステップは［画像］アイコンによって示されます。

2 右クリックして表示されるメニューから［**プロパティ**］を選択します。［画像ステップのプロパティ］ダイアログ・ボックスが開きます。

3 ステップ名を変更するには，［**ステップ名**］ボックスに新しい名前を入力します。記録時の標準の名前は，画像の ALT 属性の値です。画像に ALT 属性がない場合は，SRC 属性の最後の部分が標準名として使用されます。

4 プロパティ・テーブルに，リンクを識別するプロパティが表示されます。

［**アクティブなプロパティのみを表示する**］チェック・ボックスをクリアし，アクティブなプロパティと非アクティブなプロパティの両方を表示します。プロパティを有効にするには，プロパティ名の左のセルをクリックします。［**値**］カラムでプロパティの値を割り当てます。

- ［**ALT**］：画像の ALT 属性。
- ［**SRC**］：画像の SRC 属性。
- ［**MapName**］：画像に対応するマップ名。クライアント側イメージ・マップにのみ適用されます。
- ［**AreaAlt**］：クリックする領域の ALT 属性。クライアント側イメージ・マップにのみ適用されます。

- ［**AreaOrdinal**］：クリックする領域のシリアル番号。クライアント側イメージ・マップにのみ適用されます。
- ［**Frame**］：画像が配置されているフレームの名前。
- ［**TargetFrame**］：以下のターゲット・フレーム。

  ［**_TOP**］：ページ全体を置き換えます。

  ［**_BLANK**］：新規ウィンドウを開きます。

  ［**_PARENT**］：最後の（変更された）フレームの親を置き換えます。

  ［**_SELF**］：最後の（変更済み）フレームを置き換えます。
- ［**Ordinal**］：他のすべての属性が，同じ Web ページ上にある他の画像（1 つまたは複数）と同じときに，画像を一意に識別する番号。詳細については，「**オンライン関数リファレンス**」を参照してください。
- ［**XCoord**］，［**YCoord**］：画像上でマウスをクリックした場所の座標。

［ABC］アイコンは，プロパティの値にパラメータが割り当てられていないことを示します。パラメータの割り当て方法の詳細については，第 7 章「パラメータの定義」を参照してください。

5 ［**OK**］をクリックして［画像ステップのプロパティ］ダイアログ・ボックスを閉じます。

「フォームを送信」ステップの変更（Web のみ）

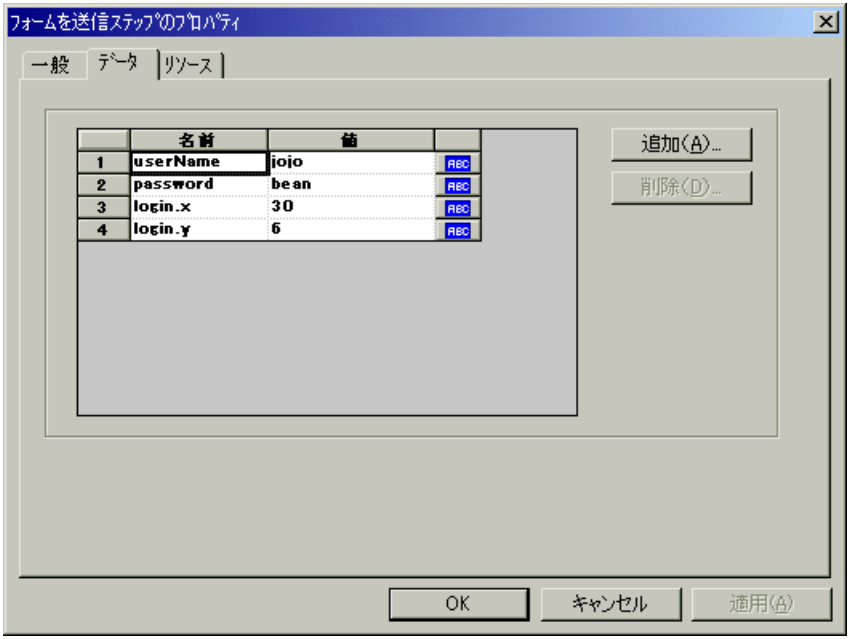
フォームを送信すると，「フォームを送信」ステップが仮想ユーザ・スクリプトに追加されます。このステップは，HTML（コンテキスト・センシティブ）モードで記録するためのオプションを選択した場合にのみ記録されます。詳細については，第 36 章「Web 仮想ユーザの記録オプションの設定」を参照してください。

変更できるプロパティはステップ名，フォームの場所，フォーム送信の識別方法，フォーム・データです。

「フォームを送信」ステップのプロパティを変更するには，次の手順で行います。

1 仮想ユーザ・スクリプトのツリー・ビューで，編集する「フォームを送信」ステップを選択します。「フォームを送信」ステップは，［フォームを送信］アイコンで示されます。

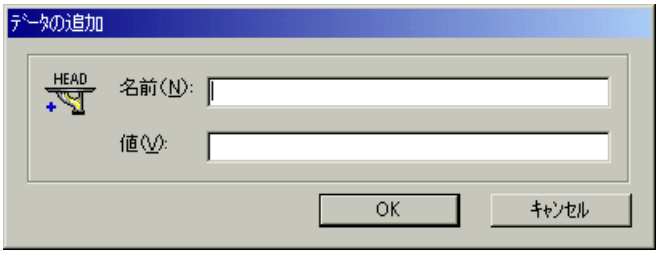
2 右クリックして表示されるメニューから［**プロパティ**］を選択します。
［フォームを送信ステップのプロパティ］ダイアログ・ボックスが開きます。
［**データ**］タブが選択されていることを確認してください。



- ［**名前**］カラムには，フォームのすべてのデータ引数が一覧表示されます。
- ［**値**］カラムには，データ引数に対応する入力値が表示されます。
- タイプを示す［値］の右のカラムには，アイコンが表示されます。最初は，すべての値が定数かパラメータ化されていない値なので，アイコンは［ABC］です。第 7 章「パラメータの定義」で説明されているとおり，データ値にパラメータを割り当てると，［ABC］アイコンはテーブル・アイコンに変わります。

3 データ引数を編集するには，データ引数をダブルクリックしてセル内でカーソルをアクティブにし，新しい値を入力します。

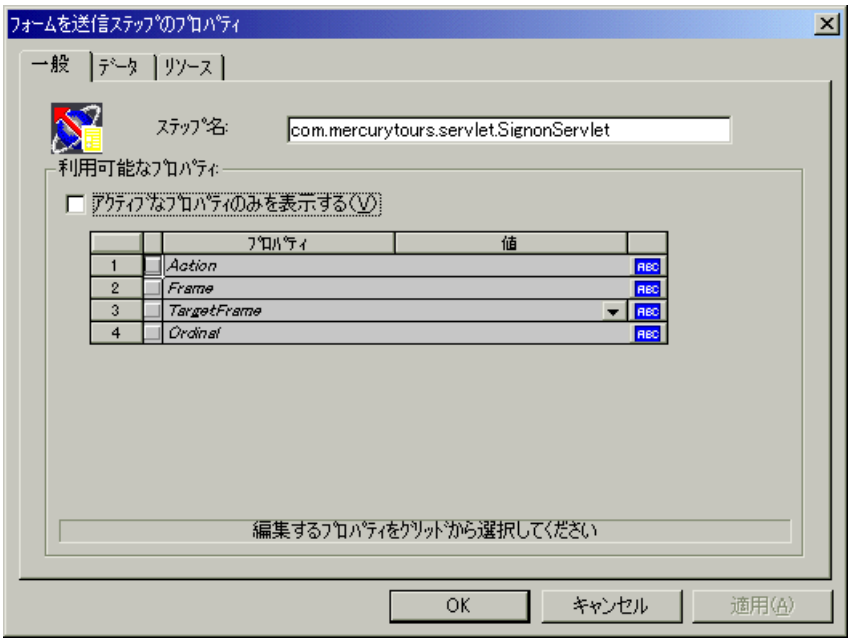
4 フォームの送信に新規データ引数を追加するには，［**追加**］をクリックします。［データの追加］ダイアログ・ボックスが開きます。



5 データ引数の［**名前**］と［**値**］を入力して，［**OK**］をクリックします。

6 引数を削除するには，引数を選択して［**削除**］をクリックします。

7 ［フォームを送信］ステップの名前を変更するには，［**一般**］タブをクリックします。



8 ステップ名を変更するには，［**ステップ名**］ボックスに新しい名前を入力します。記録時の標準の名前は，フォームの処理に使用される実行プログラム名です。

9 プロパティ・テーブルに，フォーム送信のプロパティが表示されます。

［**アクティブなプロパティのみを表示する**］オプションをクリアし，アクティブなプロパティと非アクティブなプロパティの両方を表示します。プロパティを有効にするには，プロパティ名の左のセルをクリックします。［**値**］カラムでプロパティの値を割り当てます。

- ［**Action**］：フォーム・アクションの実行に使用されるアドレス。
- ［**Frame**］：送信フォームが配置されているフレームの名前。
- ［**TargetFrame**］：以下のターゲット・フレーム。

  ［**_TOP**］：ページ全体を置き換えます。

  ［**_BLANK**］：新規ウィンドウを開きます。

  ［**_PARENT**］：最後の（変更された）フレームの親を置き換えます。

  ［**_SELF**］：最後の（変更された）フレームを置き換えます。
- ［**Ordinal**］：他のすべての属性が，同じ Web ページ上にある他のフォーム（1 つまたは複数）と同じときに，フォームを一意に識別する番号。詳細については，「**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。

［ABC］アイコンは，「フォームを送信」ステップのプロパティの値にパラメータが割り当てられていないことを示します。パラメータの割り当て方法の詳細については，第 7 章「パラメータの定義」を参照してください。

10 ［**OK**］をクリックして［フォームを送信ステップのプロパティ］ダイアログ・ボックスを閉じます。

「データを送信」ステップの変更

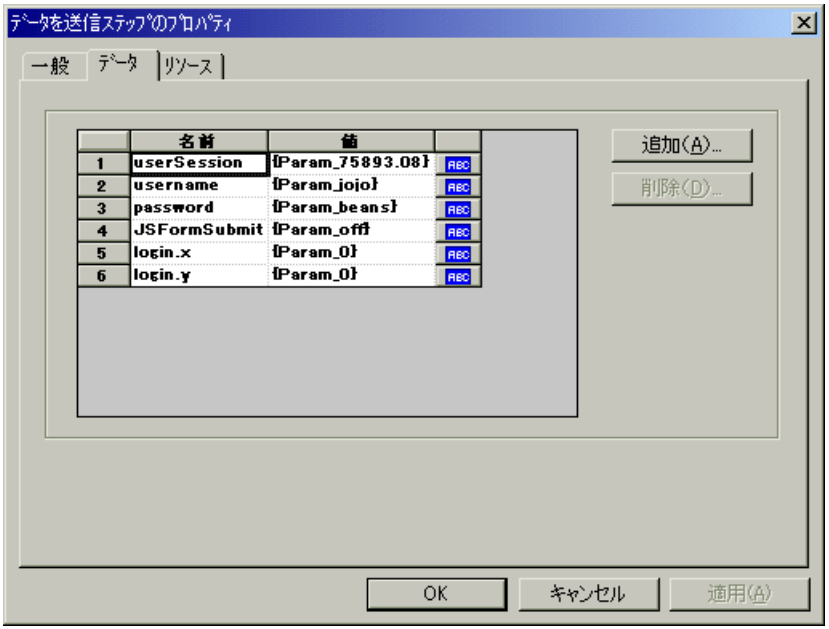
「データを送信」ステップは，Web サイトにデータ・フォームを送信して処理することを表します。「フォームを送信」ステップとは異なり，この要求を実行するときにフォームのコンテキストは必要ありません。

変更できるプロパティはステップ名，メソッド，アクション，対象フレームおよびフォームのデータ項目です。

「データを送信」ステップのプロパティを変更するには，次の手順で行います。

1 仮想ユーザ・スクリプトのツリー・ビューで，編集する「データを送信」ステップを選択します。「データを送信」ステップは，［データを送信］アイコンによって示されます。

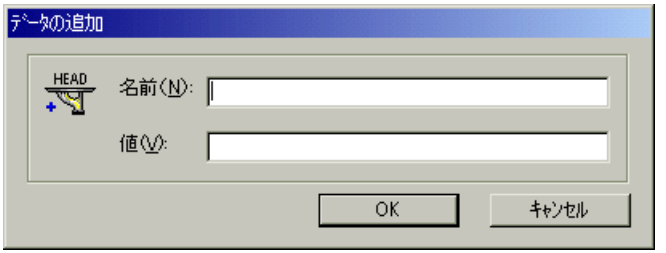
2 右クリックして表示されるメニューから［**プロパティ**］を選択します。データを送信ステップのプロパティ］ダイアログ・ボックスが開きます。［**データ**］タブが表示されていることを確認してください。



- ［**名前**］カラムには，フォームのすべてのデータ引数が一覧表示されます。これにはすべての隠しフィールドが含まれます。
- ［**値**］カラムには，データ引数に対応する入力値が表示されます。
- タイプを示す［値］の右のカラムには，アイコンが表示されます。最初は，すべての値が定数かパラメータ化されていない値なので，アイコンは［ABC］です。第 7 章「パラメータの定義」で説明されているとおり，データ値にパラメータを割り当てると，［ABC］アイコンはテーブル・アイコンに変わります。

3 データ引数を編集するには，データ引数をダブルクリックしてセル内でカーソルをアクティブにし，新しい値を入力します。

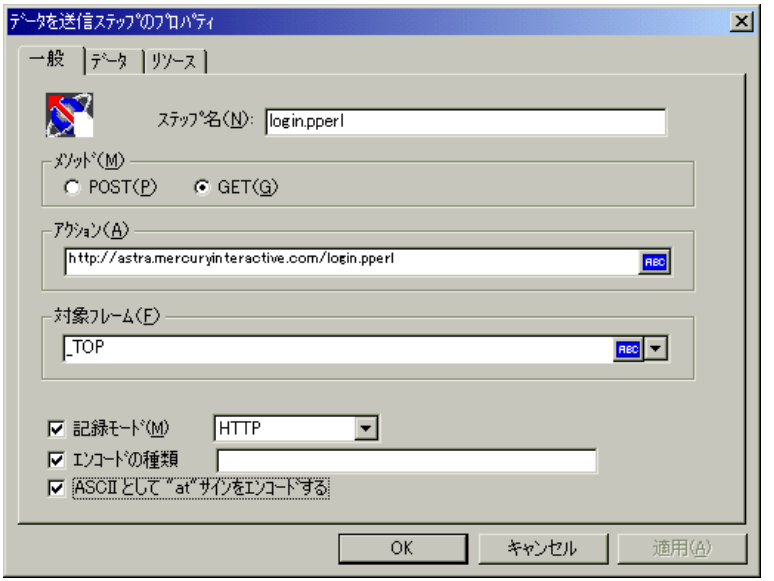
4 新規データを追加するには，［**追加**］をクリックします。［データの追加］ダイアログ・ボックスが開きます。



5 データ引数の［**名前**］と［**値**］を入力して，［**OK**］をクリックします。

6 引数を削除するには，引数を選択して［**削除**］をクリックします。

7「データを送信」ステップの名前を変更するには，［**一般**］タブをクリックします。



8 ステップ名を変更するには，［**ステップ名**］ボックスに新しい名前を入力します。

9［**メソッド**］ボックスで，［**POST**］か［**GET**］をクリックします。標準のメソッドは［**POST**］です。

10 ［**アクション**］ボックスに，データ送信時に使用するアドレスを入力します。［ABC］アイコンは，このアクションにパラメータが割り当てられていないことを示します。パラメータの割り当て方法の詳細については，第 7 章「パラメータの定義」を参照してください。

11 以下の［**対象フレーム**］から 1 つを選択します。

［**_TOP**］：ページ全体を置き換えます。

［**_BLANK**］：新規ウィンドウを開きます。

［**_PARENT**］：最後の（変更された）フレームの親を置き換えます。

［**_SELF**］：最後の（変更された）フレームを置き換えます。

12 再生モードをカスタマイズするには，［**記録モード**］オプションを選択します。次の中から使用するモードを選択します：HTML または HTTP。詳細については，529 ページ「再生モードの設定」を参照してください。

13 項目をリソースとしてダウンロードしないようにするには，［**リソース**］チェック・ボックスをクリアします。

14 **multipart/www-urlencoded** など，エンコーディング・タイプを指定するには，［**エンコードの種類**］チェック・ボックスを選択して，エンコーディング方式を指定します。

15 URL 中の「@」をエンコードするには，［**ASCII として "at" サインをエンコードする**］を選択します。

16 ［**OK**］をクリックして，［データを送信ステップのプロパティ］ダイアログ・ボックスを閉じます。

「ユーザ定義要求」ステップの変更

「ユーザ定義要求」とは，HTTP がサポートするいずれかの方法を使用した，URL に対するユーザ定義の HTTP 要求です。「ユーザ定義要求」ステップには，コンテキストはありません。

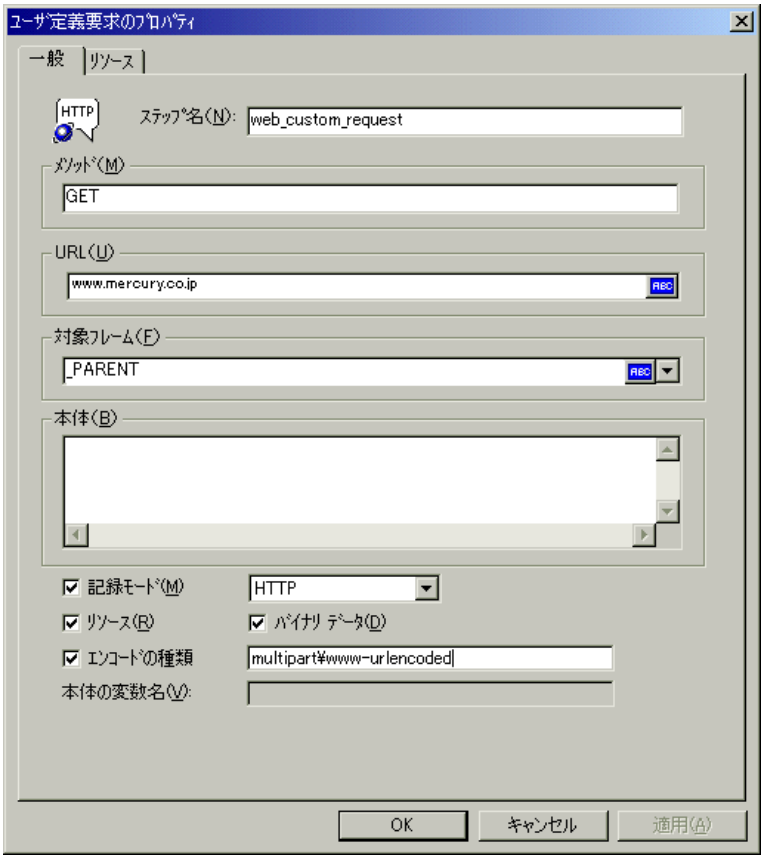
変更できるプロパティは，ステップ名，メソッド，URL，対象フレームおよび本体です。

VuGen には，ユーザ定義要求の本体を C 形式に変換する機能があります。たとえば，XML ツリーまたは大量のデータをユーザ定義要求の本体領域にコピーすることで，現在の関数で使用できるように，文字列を C 形式に変換できます。これにより，必要なエスケープ・シーケンス文字が挿入され，文字列の改行が削除されます。

［ユーザ定義要求］ステップのプロパティを変更するには，次の手順で行います。

1 仮想ユーザ・スクリプトのツリー・ビューで，編集する「ユーザ定義要求」ステップを選択します。「ユーザ定義要求」ステップは，［ユーザ定義要求］アイコンで表示されます。

2 右クリックして表示されるメニューから［**プロパティ**］を選択します。［ユーザ定義要求の［プロパティ］ダイアログ・ボックスが開きます。



3 ステップ名を変更するには，［**ステップ名**］ボックスに新しい名前を入力します。記録中は，URL の最後の部分が標準名となります。

4 ［**メソッド**］ボックスに，HTTP によってサポートされているメソッドを入力します。たとえば，GET，POST，HEAD です。

5 ［**URL**］ボックスに，要求する URL を入力します。

6 以下の［**対象フレーム**］から 1 つを選択します。

［**_TOP**］：ページ全体を置き換えます。

［**_BLANK**］：新規ウィンドウを開きます。

［**_PARENT**］：最後の（変更された）フレームの親を置き換えます。

［**_SELF**］：最後の（変更された）フレームを置き換えます。

7 ［**本体**］ボックスに，要求の本体を入力するかテキストを貼り付けます。［**バイナリ データ**］チェック・ボックスを選択すると，テキストは ASCII ではなく 2 進数として処理されます。バイナリ・データの使い方の詳細については，「**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。

8 ［**本体**］ボックスに貼り付けた文字列には，テキストを選択し，右クリックして表示されるメニューから［**C フォーマットに変換**］を選択します。

9 再生モードをカスタマイズするには，［**記録モード**］オプションを選択します。次の中から使用するモードを選択します：HTML または HTTP。詳細については，529 ページ「再生モードの設定」を参照してください。

10 項目をリソースとしてダウンロードしないようにするには，［**リソース**］オプションをクリアします。

11 **multipart/www-urlencoded** など，エンコーディング・タイプを指定するには，［**エンコードの種類**］を選択して，エンコーディング方式を指定します。

12 ［**OK**］をクリックして，［ユーザ定義要求のプロパティ］ダイアログ・ボックスを閉じます。

制御ステップの変更

制御ステップは，負荷テスト中に使用するコントロールを表します。制御ステップには，トランザクション，ランデブー・ポイント，思考遅延時間があります。

記録中または記録後に，仮想ユーザ・スクリプトのツリー・ビューに制御アイコンで表示される制御ステップをスクリプトに追加します。

本項では，以下について説明します。

- トランザクションの変更
- ランデブー・ポイントの変更
- 思考遅延時間の変更

トランザクションの変更

トランザクションとは，サーバの応答時間を測定するタスクや一連のアクションです。

変更できるプロパティは，トランザクション名（[トランザクション開始］と［トランザクション終了]），とそのステータス（[トランザクション終了］のみ）です。

「トランザクション開始」制御ステップを変更するには，次の手順で行います。

1 仮想ユーザ・スクリプトのツリー・ビューで，編集する「トランザクション開始」制御ステップを選択します。「トランザクション開始」制御ステップは，［トランザクション開始］アイコンによって示されます。

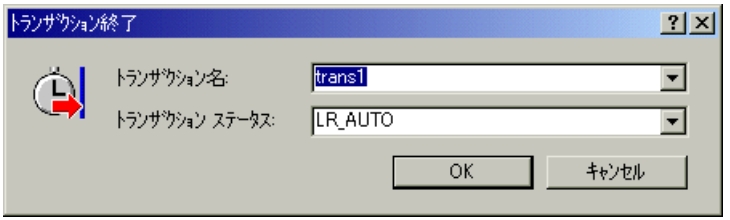
2 右クリックして表示されるメニューから［**プロパティ**］を選択します。［トランザクション開始］ダイアログ・ボックスが開きます。

3 トランザクション名を変更するには，［**トランザクション名**］ボックスに新しい名前を入力して［**OK**］をクリックします。

「トランザクション終了」制御ステップを変更するには，次の手順で行います。

1 仮想ユーザ・スクリプトのツリー・ビューで，編集する「トランザクション終了」制御ステップを選択します。「トランザクション終了」制御ステップは，［トランザクション終了］アイコンによって示されます。

2 右クリックして表示されるメニューから［**プロパティ**］を選択します。［トランザクション終了］ダイアログ・ボックスが開きます。



3 終了するトランザクションの名前を，［**トランザクション名**］リストから選択します。

4 トランザクションのステータスを［**トランザクション ステータス**］リストから選択します。

［**LR_PASS**］:「成功」のリターン・コードを返します。

［**LR_FAIL**］:「失敗」のリターン・コードを返します。

［**LR_STOP**］:「停止」のリターン・コードを返します。

［**LR_AUTO**］: 検出されたステータスを自動的に返します。

詳細については，「**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。

5 ［**OK**］をクリックして，［トランザクション終了］ダイアログ・ボックスを閉じます。

ランデブー・ポイントの変更

ランデブー・ポイントを使用して，複数の仮想ユーザが同時にタスクを実行するように同期させることができます。

変更できるプロパティは，ランデブー・ポイントの名前です。

ランデブー・ポイントを変更するには，次の手順で行います。

1 仮想ユーザ・スクリプトのツリー・ビューで，編集するランデブー・ポイントを選択します。ランデブー・ポイントは，［ランデブー］アイコン によって表されます。

2 右クリックして表示されるメニューから［**プロパティ**］を選択します。［ランデブー］ダイアログ・ボックスが開きます。

3 ランデブーの名前を変更するには，［**ランデブー名**］ボックスに新しい名前を入力し，［**OK**］をクリックします。

思考遅延時間の変更

思考遅延時間は，アクション間で実際のユーザが待機する時間をエミュレートします。記録中，アクション間の時間があらかじめ定義したしきい値の 4 秒を超えると，VuGen によってそれぞれのユーザ・アクションの後に，思考遅延時間が自動的に仮想ユーザ・スクリプトに追加されます。

変更できるプロパティは思考遅延時間（秒単位）です。

思考遅延時間を変更するには，次の手順で行います。

1 仮想ユーザ・スクリプトのツリー・ビューで，編集する「思考遅延時間」ステップを選択します。「思考遅延時間」ステップは，［思考遅延時間］アイコンによって示されます。

2 右クリックして表示されるメニューから［**プロパティ**］を選択します。［思考遅延時間］ダイアログ・ボックスが開きます。

3 ［**思考遅延時間**］ボックスに思考遅延時間を入力し，［**OK**］をクリックします。

注：VuGen またはコントローラから Web 仮想ユーザ・スクリプトを実行するときは，記録されたとおりに思考遅延時間を再生するか，記録された思考遅延時間を無視するかを，仮想ユーザに対して設定できます。詳細については，第 9 章「実行環境の設定」を参照してください。

「サービス」ステップの変更

「サービス」ステップは，プロキシの設定，認証情報の送信，カスタム・ヘッダーの発行などの，カスタマイズのためのタスクを実行する関数です。「サービス」ステップは，Web サイトのコンテキストを一切変更しません。

記録中または記録後に「サービス」ステップをスクリプトに追加できます。

「サービス」ステップのプロパティを変更するには，次の手順で行います。

1 仮想ユーザ・スクリプトのツリー・ビューで，編集する「サービス」ステップを選択します。「サービス」ステップはサービス・アイコンによって示されます。

2 右クリックして表示されるメニューから［**プロパティ**］を選択します。サービス・ステップ・プロパティのダイアログ・ボックスが開きます。このダイアログ・ボックスは，変更するサービス・ステップの種類により異なります。選択したサービス・ステップについての説明が，ダイアログ・ボックスのタイトル・バーに表示されます。

注：一部のサービス・ステップ関数には，引数がありません。この場合，プロパティのメニュー項目は無効です。

---

3 サービス・ステップに必要な引数を入力または選択します。各関数の詳細については，「**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。

4 [**OK**] をクリックして，サービス・ステップのプロパティ・ダイアログ・ボックスを閉じます。

Web チェックの変更（Web のみ）

Web チェックは Web ページで特定のオブジェクトの存在を確認する機能です。オブジェクトは，テキスト文字列，画像，または Java アプレットです。

記録中または記録後に Web チェックをスクリプトに追加できます。

Web チェックのプロパティを変更するには，次の手順で行います。

1 仮想ユーザ・スクリプトのツリー・ビューで，編集する Web チェックを選択します。Web チェックは Web チェック・アイコンで表示されます。

2 右クリックして表示されるメニューから [**プロパティ**] を選択します。該当する Web チェックのプロパティ・ダイアログ・ボックスが開きます。このダイアログ・ボックスは，変更するチェックの種類により異なります。

3 チェックに必要なプロパティを入力または選択します。詳細については，第 39 章「負荷下の Web ページ検証」を参照してください。

4 [**OK**] をクリックして，チェックのプロパティ・ダイアログ・ボックスを閉じます。

第 41 章

Web 仮想ユーザ・スクリプトの相関ルールの設定

VuGen の相関機能を使うと，1 つのステートメントの結果を別のステートメントの入力項目として使用して，仮想ユーザ関数どうしを結び付けることができます。

本章では，記録中にステートメントを相関させる方法について説明します。以下の項目で構成されます。

- 相関の方法について
- 相関処理方式の選択
- ルールのテスト
- 相関記録オプションの設定

以降の情報は，Web 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

HTML ページには，動的データが含まれていることがよくあります。動的データとは，ユーザがサイトにアクセスするたびに変わるデータです。たとえば，Web サーバによっては，現在の日時を含むリンクを使用するものもあります。

Web 仮想ユーザ・スクリプトを記録すると，動的データがスクリプトに記録されることがあります。そのスクリプトによって，記録された変数が Web サーバに送信されても，変数は有効なものではなくなっています。これらの変数は Web サーバによって拒否され，エラーが発行されます。このようなエラーは必ずしもはっきりわかるわけではなく，仮想ユーザ・ログ・ファイルを注意深く調べないと検出できないことがあります。

仮想ユーザの実行時にエラーが発生した場合は，スクリプト内でのエラーの発生箇所を調べてください。多くの場合，相関によって，1 つのステートメントの結果を別のステートメントの入力項目として使用することで，問題を解決できます。

HTML ページ内の動的データは，次の形式になっていることがあります。

- 対応する Web ページにアクセスするたびに変わる URL
- フォーム送信時に記録されたフィールド（場合によっては，隠しフィールド）
- Java スクリプトのクッキー

ケース 1

「Buy me now!」というテキストのハイパーテキスト・リンクを含む Web ページがあるとします。HTTP データを含む仮想ユーザ・スクリプトを記録すると，その URL は VuGen によって次のように記録されます。

"http://host//cgi-bin/purchase.cgi?date=170397&ID=1234"

日付「**170397**」と ID「**1234**」は記録中に動的に生成されるので，新たなブラウザ・セッションを実行するたびに日付と ID が再生成されます。スクリプトを実行すると，「Buy me now!」のリンクは，記録時の URL ではなく，別の URL に関連付けられています。このため，Web サーバはその URL を取得できません。

ケース 2

名前と顧客 ID をフォームに入力し，フォームを送信するとします。

このフォームを送信すると，一意のシリアル番号がユーザのデータと一緒にサーバに送られます。このシリアル番号は HTML コード内の隠しフィールドに含まれていますが，VuGen によってスクリプトに記録されます。シリアル番号はブラウザ・セッションごとに変わるので，LoadRunner は記録されたスクリプトを正しく再生できません。

ステートメントを相関させることで，上の 2 つのケースにおける問題を解決できます。記録されたスクリプトの動的データを，1 つまたは複数のパラメータで置き換えます。仮想ユーザ・スクリプトを実行すると，LoadRunner によってパラメータに値が割り当てられます。

相関の方法について

本章では，組み込みルールまたはユーザ定義のルールを使った自動相関について説明します。ステートメントを手作業で相関する場合や，ワイヤレスの仮想ユーザ・スクリプトを実行する場合は，575 ページ「手作業による相関」を参照してください。

ブラウザ・セッションを記録するときには，まず HTML モードで記録してみます。このモードを使用すると，相関が必要な箇所が少なくなります。各種の記録モードの詳細については，475 ページ「記録レベルの選択」を参照してください。

記録中または記録後に，スクリプト内のステートメントを相関させるように LoadRunner を設定できます。本章では，記録時のソリューションとして，スクリプト内のステートメントを自動的に相関させます。また，VuGen のスナップショット相関機能を使って，記録**後**にスクリプトを相関させることもできます。記録後の相関の詳細については，第 42 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。

VuGen の相関ルールの使用

VuGen の相関エンジンを使えば，記録セッション中に，次のいずれかのメカニズムを使用して動的なデータを自動的に相関できます。

- 組み込み相関
- ユーザ定義ルールによる相関

詳細については，554 ページ「一致条件の追加」および 555 ページ「高度な相関ルール」を参照してください。

組み込み相関

組み込み相関では，サポートされているアプリケーション・サーバを対象に動的データを検出し，相関させることができます。ほとんどのサーバには，リンクおよび参照の作成時に必ず使用される，明確な構文ルール，つまり「**コンテキスト**」があります。たとえば，BroadVision サーバで作成されるセッション ID は，次に示すように，必ず特定の区切り文字の間に挟まれています。左側は「BV_SessionID=」，右側は「&」です。

```
BV_SessionID=@@@@1303778278.0969956817@@@@&
```

サポートされているアプリケーション・サーバを対象にセッションを記録するときは，VuGen に組み込まれている既存ルールの 1 つを使用できます。各アプリケーション・サーバには 1 つ以上のルールを適用できます。また，ルールの横にあるチェック・ボックスを設定またはクリアして，特定のルールを有効または無効にできます。ルールの定義は VuGen の右側の表示枠に表示されます。

サポート対象外のアプリケーション・サーバのセッションを記録するとき，コンテキストが不明で相関のルールを決められない場合には，VuGen のスナップショット比較方式を使用することができます。この方式では，記録後に相関処理の手順を導いてくれます。詳細については，第 42 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。

ユーザ定義ルールによる相関

アプリケーションに固有のルールがあり，それらを明確に定義できる場合は，［記録オプション］タブで新規ルールを定義できます。

ユーザ定義のルールによる相関を行う場合は，セッションを記録する前に，相関のルールを定義する必要があります。相関のルールは，［記録オプション］ダイアログ・ボックスで作成します。ルールには，相関させる動的データの境界などの情報や，バイナリ，大文字と小文字の一致，インスタンス番号など，一致に関する仕様が含まれます。

VuGen に対して，どの場所で条件を検索するかを指定します。

- 本体テキスト全体
- リンク / フォーム・アクション
- クッキー・ヘッダー
- フォーム・フィールド値
- クッキー挿入関数

本体テキスト全体

［**本体テキスト全体の中でパラメータを探します**］オプションを選択すると，レコーダはリンク，フォーム・アクションまたはクッキーだけではなく，ボディ全体の中で一致する項目を探します。テキストは，指定した境界を使って検索されます。

リンク / フォーム・アクション

［**リンクおよびフォームのアクションの中でパラメータを探します**］方式では，VuGen はリンク・タイプおよびフォーム・タイプのアクションの中でパラメータ化するテキストを検索します。この方式は，コンテキストのルールがわかっているアプリケーション・サーバ向けです。左の境界，右の境界，代替の右境界，左の境界のインスタンス（左の境界の出現）を現在のリンクで定義します。

たとえば，文字列「sessionid=」の 2 回目の出現と「@」の間にあるテキストをパラメータに置き換えるとします。［**左の境界**］ボックスに，左の境界として「**sessionid=**」を指定し，［**右の境界**］ボックスに，右の境界として「**@**」を指定します。2 回目の出現を検索しているので，［**左の境界インスタンス**］ボックスで［**2**］を指定します。

右の境界の文字列が一定でない場合は，［**代替の右境界**］ボックスに，代わりに使用する右の境界を指定できます。指定した右の境界を一意に特定できないときに，この値が使用されます。

たとえば，Web ページに次の形式のリンクが含まれていたとします。

"SessionID=122@page.htm"
"Page.htm@SessionID=122&test.htm"

右の境界が一定でないので（「@」の場合と「&」の場合がある），右の境界として 1 つを指定するだけでは不十分です。この場合，代わりに使用する右の境界として「**&**」を指定します。

左と右の境界は，文字列を一意に特定するものでなければなりません。境界に動的データを含めることはできません。また，プルダウン・リストの選択肢として用意されている［**End of String**］または［**改行文字**］を，右の境界として指定することもできます。どちらの右の境界も見つからない場合は，左の境界からソース文字列の終わりまでのテキストがパラメータに保存されます。

このオプションでは，左右の境界はスクリプトに含まれている文字列内に必ず含まれている必要があります。サーバが境界を返すだけでは十分ではありません。この制限は他のアクション・タイプにはありません。

クッキー・ヘッダー

［**クッキー ヘッダーの中でパラメータを探します**］方式は，前述のルールと似ていますが，値がリンクまたはフォームのアクションからではなく，クッキーのテキストから（記録ログに示されるとおりに）抽出される点が異なります。

さらに，リンクまたはフォーム・アクションのルールでは，境界と一致する URL の部分だけがパラメータ化されます。クッキーのルールでは，リンクまたはフォームおよびアクション・フォームのフィールドで抽出された値を検索し，パラメータに置換するため，境界を指定する必要がありません。

フォーム・フィールド値

［**フォーム フィールド値のパラメータ化**］方式を指定すると，レコーダは名前付きのフォーム・フィールドをすべてパラメータとして保存します。この方式は，パラメータを作成し，それをスクリプト内のフォームのアクション・ステップの前に配置します。このオプションでは，フィールド名を指定する必要があります。

クッキー挿入関数

［**web_reg_add_cookie 関数を挿入します**］方式は，バッファの中で特定の文字列を検出すると **web_reg_add_cookie** 関数を挿入します。指定した接頭辞を持つクッキーを検出したときにのみ関数が追加されます。このオプションでは，検索するテキストとクッキーの接頭辞を指定する必要があります。

一致条件の追加

上記のルール以外に，文字列で次の項目を指定することによって，相関検索の種類を制御することもできます。

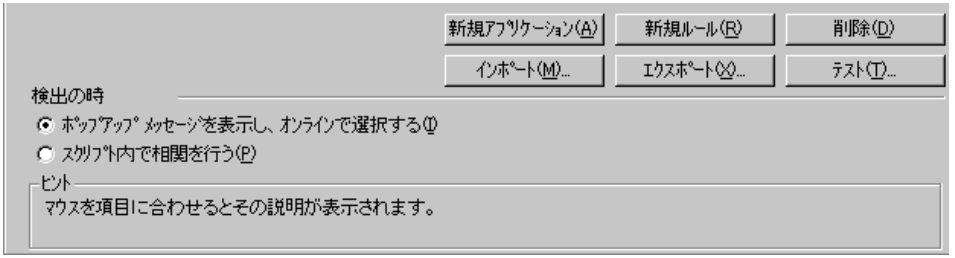
［**パラメータ前置記号**］：このルールに基づいて自動的に生成されたすべてのパラメータに，接頭辞を使用します。接頭辞によって，既存のユーザ・パラメータへの上書きを防ぐことができます。さらに，接頭辞によって，スクリプト内のパラメータが見分けやすくなります。たとえば，組み込みのルールの 1 つである Siebel-Web は Siebel_row_id という接頭辞を検索します。

［**大文字と小文字を区別する**］：境界を検索するときに大文字と小文字を区別します。

［**"#" で数字を置き換える**］：数値をすべてハッシュ記号（#）で置き換えます。このオプションを有効にすると，数値以外のすべてが一致するテキスト文字列を探せます。たとえば，左側の境界が **Mercury193** の場合，**Mercury284** も一致します。この場合，左の境界ボックスで **Mercury###** と指定します。

相関処理方式の選択

相関を有効にすると，使用するアプリケーションに対応するルールがあるかどうかが VuGen によって確認されます。動的データが既存のルールに従う場合，VuGen はダイアログ・ボックスの［**検出の時**］セクションに指定されている次の設定に基づいて相関の準備を行います。



- ［**ポップアップ メッセージを表示し，オンラインで選択する**］：記録中に動的データを検出したときに，相関させる前にメッセージを表示します。
- ［**スクリプト内で相関を行う**］：スクリプト内でステートメントを自動相関します。

VuGen では，次の高度な相関ルールも指定できます。

高度な相関ルール

VuGen では，次の詳細オプションも指定できます。

［**常に新規のパラメータを作成する**］：パラメータに置換された値が，前のインスタンスから変わっていない場合も，このルールに応じて新しいパラメータを作成します。Web サーバがページごとに異なる値を割り当てる場合には，このオプションを設定します。たとえば，NetDynamics サーバは，不正行為を最小限に抑えるために，ページごとにセッション ID を変更する場合などがあります。

［**完全一致のみパラメータで置換する**］：境界と境界の間のテキストが（最初のスナップショットから）見つかった値と完全に一致した場合のみ，記録された値をパラメータで置き換えます。見つかった文字列の前または後に別の文字がある場合，パラメータの置き換えは実行されません。たとえば，フォーム送信の際，VuGen によって境界 **aaa** と **bbb** の間の 1234 という文字群が記録されたとします。**web_submit_data** の引数 Name は Name=aaa1234bbb です。以後このフォームを送信する際，VuGen は **1234** という文字を見つけると（つまり，Name=1234 の場合），記録された値をパラメータで置き換える処理のみ実行します。他の値が入力された場合，その値に前方一致となる文字列（たとえば，

Name=12345）が含まれている場合でも，VuGen はその値をパラメータで置き換えずに 12345 という値を使用します。

［**逆方向検索**］：文字列の最後から左境界を検索します。

［**左の境界インスタンス**］：一致として考えられる，文字列内（ボディではなく）の左境界の一致件数です。

［**オフセット**］：一致した値の部分文字列の開始位置を指定するオフセットです。部分文字列がパラメータに保存されます。標準設定は，一致した文字列の最初の部分です。必ず負数以外を指定してください。

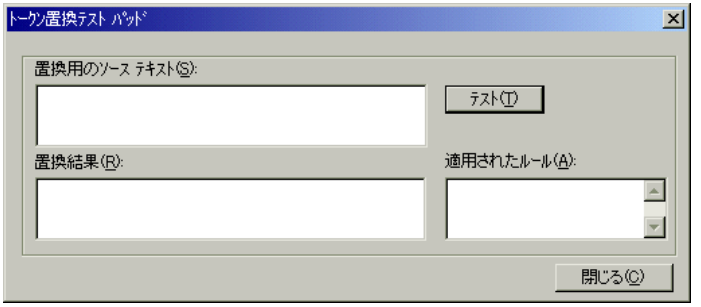
［**長さ**］：パラメータに保存する，一致した文字列の部分文字列の，オフセットからの長さです。このオプションを無効にすると，標準の値が使用され，指定したオフセットから一致文字列の末尾までの文字列が保存されます。

［**代替の右境界**］：あらかじめ指定されている右境界が見つからなかった場合の代替条件です。テキスト，文字列の末尾または改行文字を指定できます。

ルールのテスト

本項の内容は，コンテキストが知られているサーバを対象に作成したユーザ定義のルールを対象とします。［相関］パネルで新しいルールを定義したら，セッションを記録する前に，サンプルの文字列に対してルールを適用することによってテストを実行できます。［トークン置換テスト パッド］を使用してルールをテストします。テスト・パッドを使用するには，次の手順で行います。

1 左側の表示枠でルールを選び，［**テスト**］をクリックします。［トークン置換テスト パッド］ダイアログ・ボックスが開きます。



2 [**置換用のソース テキスト**] ボックスにテキストを入力します。

3 [**テスト**] をクリックします。

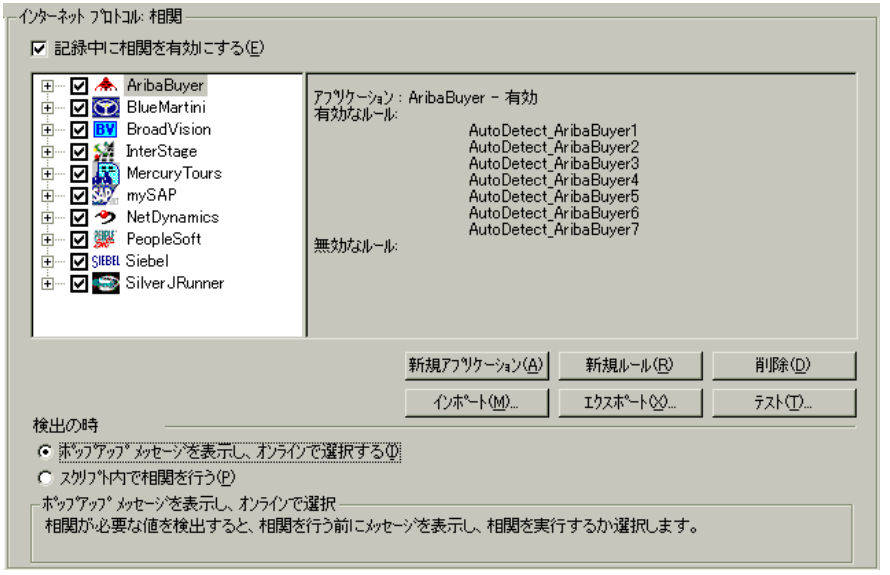
置換が行われた場合，パラメータ化されたソース・テキストは [**置換結果**] ボックスで確認でき，その置換に適用されたルールのリストは [**適用されたルール**] ボックスで確認できます。

相関記録オプションの設定

LoadRunner を使用して，記録中にステートメントを相関させるには，[相関] 記録オプションを設定します。これらのオプションは，Web 仮想ユーザ・スクリプトを開いた後，セッションの記録を開始する前に設定します。

[相関] 記録オプションを設定するには，次の手順で行います。

1 スクリプトの作成後，記録を開始する前に [**ツール**] > [**記録オプション**] で [**インターネット プロトコル：相関**] ノードを選択します。



2 [**記録中に相関を有効にする**] オプションを選択します。

3 相関ルールを適用する対象となるサーバを指定します。サーバ名の隣のチェック・ボックスを選択し，指定サーバでルールを有効にします。サーバ・グルー

プの特定のルールを有効にするには，「+」印をクリックしてツリーを展開し，必要なルールを選択します。

4 既存サーバに新しいルールを追加するには，既存エントリの中から 1 つを選択し，［**新規ルール**］をクリックします。ルールのプロパティは右側の表示枠で設定します。詳細については，559 ページ「相関のルールの設定」を参照してください。

5 新しいアプリケーションにルールのセットを追加するには，［**新規アプリケーション**］をクリックします。次に［**新規ルール**］をクリックして，アプリケーションのルールを作成します。

6 既存ルールのプロパティを変更するには，ルールを左側の表示枠で選択し，右側の表示枠で変更をします。

7 相関させる必要がある値を検出したときの VuGen のアクションとして，［**ポップアップ メッセージを表示し，オンラインで選択する**］または［**スクリプト内で相関を行う**］を指定します。標準では，LoadRunner はポップアップ・メッセージを表示します。

8 アプリケーションやルールを削除するには，それを選択して［**削除**］ボタンをクリックします。選択した項目を削除する前に，VuGen によって警告を表示します。

9 相関ルールのセットをエクスポートするには，［**エクスポート**］をクリックして **cor** ファイルを目的の場所に保存します。以前のセッションで作成した相関ルールのセットをインポートするには，［**インポート**］をクリックしてその場所からファイルを開きます。

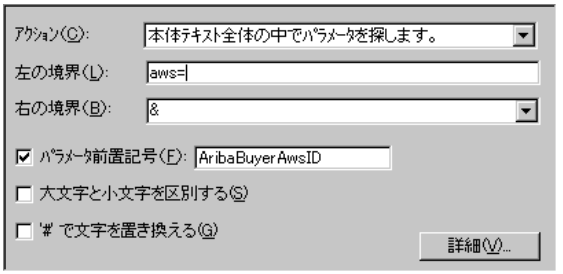
10 ［**OK**］をクリックします。

相関のルールの設定

［相関］パネルでは，ルールを追加，変更または削除できます。アプリケーション・サーバの環境を対象に自動的に作成されたルールを編集することもできます。

相関のルールを定義するには，次の手順で行います。

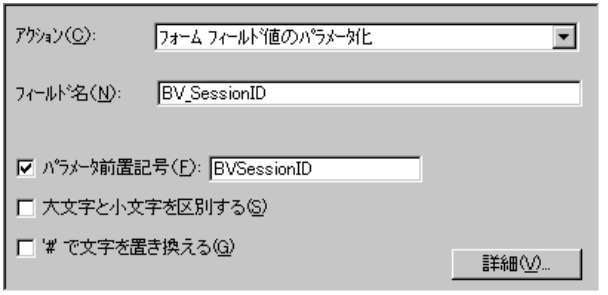
1 既存ルールをクリックするか，表示枠の下の［**新規ルール**］をクリックします。右側の表示枠に相関ルールが表示されます。



2 アクションの種類をリンクまたはフォーム・アクション，クッキー，すべてのボディ，またはフォーム・フィールドから選択します。

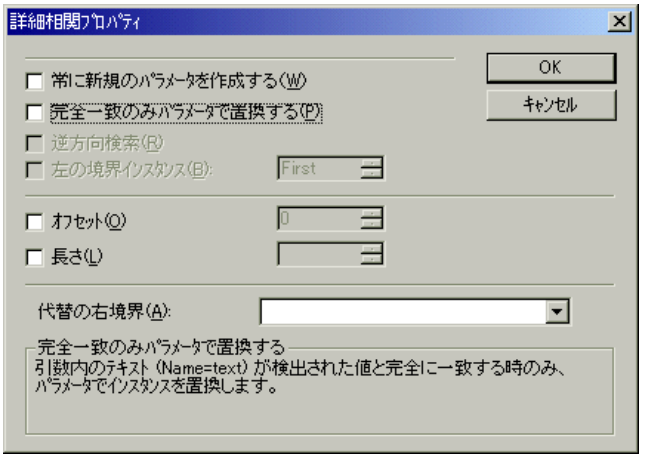
3 最初の 3 種類では，［**左の境界**］と［**右の境界**］ボックスでデータの境界を指定します。

4 フォーム・フィールド・タイプのアクションの場合は，フィールド名を指定します。



5 ［**大文字と小文字を区別する**］および［**パラメータ前置記号**］のいずれか一方，または両方のオプションを指定します。パラメータの接頭辞を指定します。すべての数字を # 記号に変換するには，［**'#' で文字を置き換える**］を選択します。

6 詳細ルールを設定するには，［**詳細**］をクリックします。［詳細相関プロパティ］ダイアログ・ボックスが開きます。



- ［**常に新規のパラメータを作成する**］を選択し，パラメータによって置き換えられる値が前のインスタンスから変わっていない場合も，このルールに応じて新しいパラメータを作成します。
- ［**完全一致のみパラメータで置換する**］を選択して，テキストが検出された値に正確に一致する場合にだけ，その値をパラメータで置換します。
- ［**逆方向検索**］を選択すると，逆方向に検索します。
- ［**左の境界インスタンス**］ボックスを選択し，必要なインスタンスを指定します。
- ［**オフセット**］を選択して，一致した文字列内の文字列のオフセットを指定します。
- ［**長さ**］を選択して，一致した文字列の何文字までパラメータに保存するかを指定します。このオプションは［**オフセット**］オプションと組み合わせて使用することができます。
- ［**代替の右境界**］ボックスで別の右境界を指定するか，プルダウン・リストで［**End of String**］または［**改行文字**］を選択します。

7 ［**テスト**］をクリックして，定義したルールをテストします。詳細については，556 ページ「ルールのテスト」を参照してください。

8 ［**OK**］をクリックしてルールを保存し，ダイアログ・ボックスを閉じます。

第 42 章

記録後の仮想ユーザ・スクリプトの相関

記録中に相関を行わなかった場合，VuGen に組み込まれている Web 相関メカニズムを使って，記録セッション後に仮想ユーザ・スクリプトを相関させることができます。

本章では，以下の項目について説明します。

- スナップショットについて
- VuGen の相関の設定
- 相関の検索の実行
- 手作業による相関
- 動的文字列の境界の定義

以降の情報は，Web，ワイヤレス，および<アプリケーション> -Web 仮想ユーザ・スクリプトを対象とします。

スナップショットによる相関について

VuGen には，Web 仮想ユーザ・スクリプトを対象にしたいくつかの相関メカニズムが用意されています。第 41 章「Web 仮想ユーザ・スクリプトの相関ルールの設定」で説明した自動による方法は，記録中に動的な値を検出するので，それらの値を直ちに相関させることができます。自動相関を無効にした場合や，自動による方法で差異のすべてが検出されなかった場合は，本章で説明する VuGen の組み込み相関メカニズムを使って差異を検出し，値を相関させることができます。部分的に相関されたスクリプトに対しても，このメカニズムを使用できます。

相関メカニズムは「**スナップショット**」を使用して，スクリプトの実行結果を追跡します。スナップショットは，Web ページを視覚的に表したものです。VuGen は記録中に基本となるスナップショットを作成し，スクリプトを実行するたびに新しいスナップショットを作成します。記録済みのスナップショット

と再生時のスナップショットを比較することによって，スクリプトの実行を成功させるために相関させる必要のある値を特定します。

Web 相関メカニズムには，スナップショット間のテキストまたはバイナリの差異を表示できる，比較ユーティリティが組み込まれています。比較の後，差異を 1 つずつ相関させることも，一度にすべてを相関させることもできます。

VuGen の相関メカニズムでは十分でない場合や，こうしたメカニズムをサポートしないプロトコル（ワイヤレスや手作業による相関など）に対しては，手作業による相関を使用します。詳細については，575 ページ「手作業による相関」を参照してください。

スナップショットについて

相関メカニズムは「**スナップショット**」を使用して，スクリプトの実行結果を追跡します。スナップショットは，Web ページを視覚的に表すものです。VuGen は記録中に基本となるスナップショットを作成し，スクリプトを実行するたびに新しいスナップショットを作成します。スナップショットと HTML コード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。

スナップショットのファイルは，拡張子 .**inf** が付けられ，スクリプト・ディレクトリに格納されます。記録中に作成されたスナップショットは，仮想ユーザ・スクリプトの「**data**」フォルダに格納されます。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反復フォルダ（**Iteration1**，**Iteration2** など）に置かれます。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。

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，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は生成されていない。
- 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。

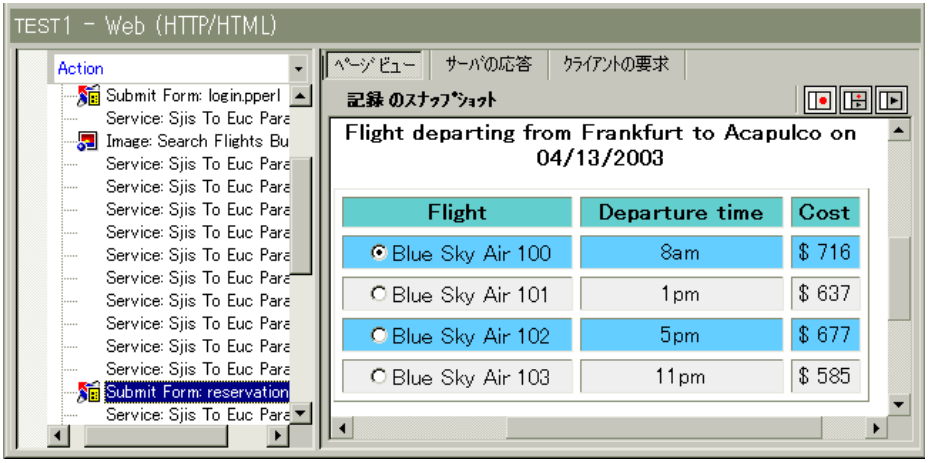
選択したステップの再生時のスナップショットがない場合，次のような原因が考えられます。

- 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。
- 次のオプションが無効になっているため，スナップショットが生成されない。

  ［**ツール**］＞［**一般オプション**］＞［**相関**］タブ＞［**再生時に相関情報を保存する**］

標準では，ツリー・ビューで作業しているときに，選択したステップのスナップショットは VuGen の右側の表示枠に表示されません。スナップショットの表示と非表示を切り替えるには，［**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。



［**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，再生時の一方または両方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。

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・ファイルの名前を調べるには，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**］）。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。

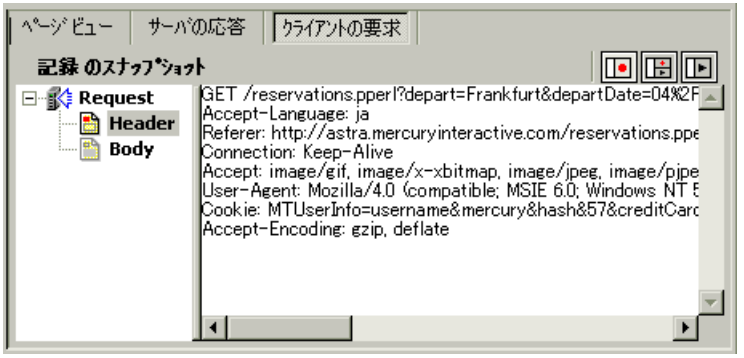
```
web_url("www.aa.com",
        "URL=http://www.aa.com/",
        "Resource=0",
        "RecContentType=text/html",
        "SupportFrames=0",
        "Snapshot=t1.inf",
            LAST);
```

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わかりません。

［**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。

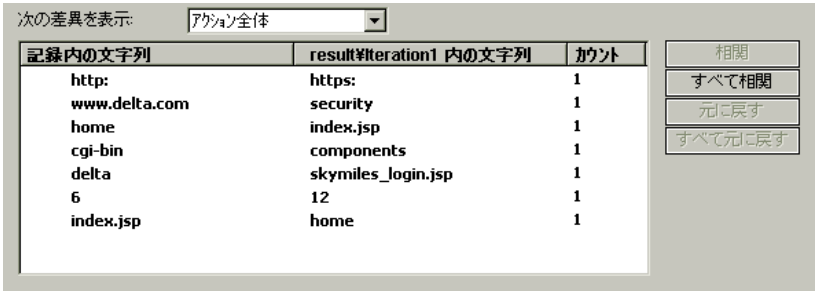


［**テスト結果を選択**］：（再生時のスナップショットのみ）記録時のスナップショットと比較する再生時のスナップショットを選択できます。このボタンをクリックすると，現在のスクリプトの結果フォルダを一覧表示するダイアログ・ボックスが開きます。再生時のスナップショットは，結果セットごとにスクリプトの反復フォルダ（**Iteration1**，**Iteration2** など）に置かれます。

相関結果タブの表示

相関結果］タブには，記録時のスナップショットと再生時のスナップショットの差異が表示されます。

スクリプト内を検索して相関を見つけるように指示すると，出力ウィンドウが開いて［**相関結果**］タブが表示されます。［**相関結果**］タブには，記録時のスナップショットと再生時のスナップショットの差異が表示されます。



［**次の差異を表示**］リスト・ボックスから対象を選択することによって，スクリプト内のすべての差異を表示したり，現在のステップの差異だけを表示したりできます。

相関され差異が発見された項目には，左端のカラムにチェック・マークが示されます。その右の 2 つのカラムには，スナップショット間の HTML の差異が表示されます。右端のカラム［**カウント**］には，記録されたスナップショット間でその差異が出現した回数が示されます。

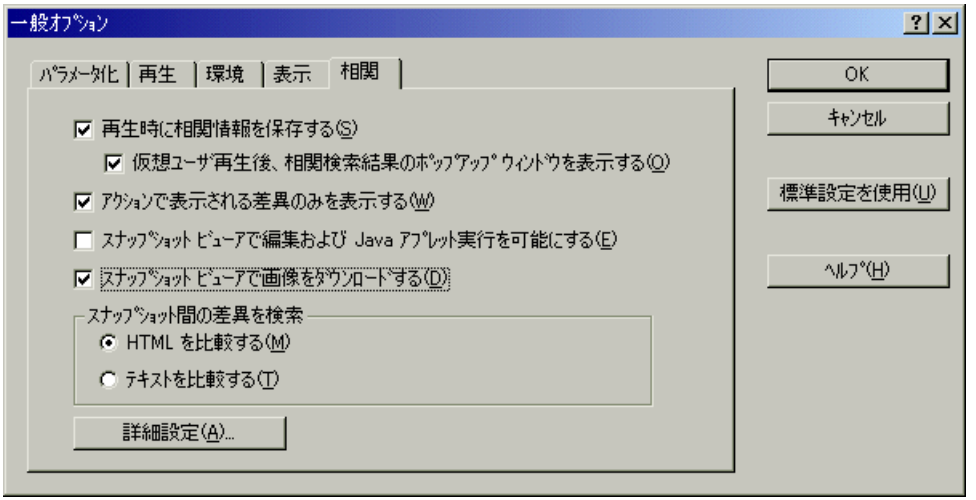
スナップショット間の差異を見つけたら，それらを一度に 1 つずつ相関させるか（［**相関**］），すべての差異を一度に相関させます（［**すべて相関**］）。また，特定の相関を取り消したり（［**元に戻す**］），すべての相関を取り消したり（［**すべて元に戻す**］）することもできます。

このメカニズムを使って値を相関させると，VuGen は **web_reg_save_param** 関数と，パラメータを対象に相関が行われたことを示すコメントをスクリプトに挿入します。コメントには，元の値も示されます。

```
// Correlation Studio created parameter {WCSParam_Diff1}; replaced
value:falillgidgkbfdlcflmcfkgdgff.0
web_reg_save_param( "WCSParam_Diff1", "LB=BV_EngineID=", "RB=&",
"Ord=1", "Search=body", LAST );
web_url("American2",

"URL=http://www.im.aa.com/American?BV_EngineID={WCSParam_Diff1}
&BV_Operation=Dyn_Frame&form%25framespacing=0&BV_SessionID=
%40%40%40%401303778278.0969956817%40%40%40%40&form%25d
estination=%2fnavguest.tmpl&form%25destination_type=template&form%
25border=0&BV_ServiceName=American&form%25frameborder=no",
        "TargetFrame=",
        "Resource=0",
        "RecContentType=text/html",
        "SupportFrames=0",
        "Referer=http://www.im.aa.com/Ameri
can?BV_Operation=Dyn_AAPage&ref
erer=index.html&form%25referrer_site=None",
        "Snapshot=t3.inf",
            LAST);
```

VuGen の相関の設定

セッションの記録を開始する前に，［一般オプション］の［相関］タブで相関の設定を行います。これらのオプションを設定すると，仮想ユーザは後で使用できるように，記録中に相関情報を保存します。スナップショットを比較する際に，HTML またはテキストのどちらを比較するかを指定できます。［詳細相関］ダイアログ・ボックスでは，区切り文字として扱う文字も指定できます。



［**再生時に相関情報を保存する**］：再生情報をスナップショットとして保存します。再生時のスナップショットのいずれかと記録時のスナップショットを比較できます。

［**仮想ユーザ再生後，相関検索結果のポップアップ ウィンドウを表示する**］：アクションを対象に相関のスキャンを行う前に確認を求めるように VuGen に指示します。VuGen によって，再生後にアクションを対象に相関のスキャンを行うかどうか確認を求められます。

［**アクションで表示される差異のみを表示する**］：現在のアクションに現れるステートメントは，HTML ページの呼び出しを生成するステートメントであり，サーバから返された HTML データではありません。多くの場合，相関を行う際にこの情報があれば十分です。後でステートメントで使用される，相関が必要な動的データはすべて，現在のアクションに示されます。現在のアクションに現れる差異を表示する場合にはこのオプションを選択します（標準設定）。まれに，現在のアクションにはないデータからパラメータを作成する必要が生じることがあります。その場合は，このオプションを無効にします。

［**スナップショット ビューアで編集および Java アプレット実行を可能にする**］: スナップショット・ウィンドウでアプレットと JavaScript を実行できるようにします。多量のリソースを使用するので，このオプションは標準設定では無効になっています。

［**スナップショット ビューアで画像をダウンロードする**］: 画像をスナップショット・ビューアに表示するよう VuGen に指示します。このオプションを無効にすると，リソースを節約できます。標準設定では，このオプションは有効になっています。たとえば，相関が必要な画像がないことがわかっている場合，このオプションを無効にできます。このオプションを無効にすると，リソースを節約できます。標準設定では，このオプションは有効になっています。

［**スナップショット間の差異を検索**］: 比較の方法を選択します。

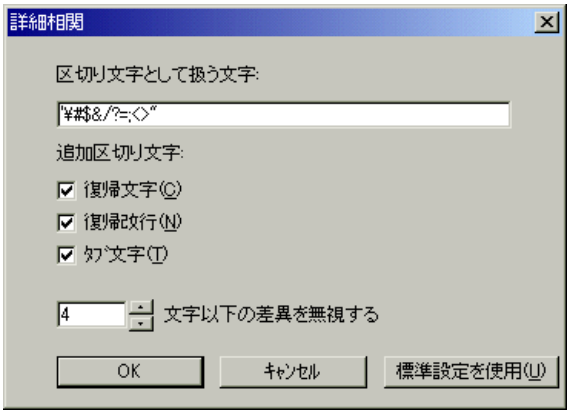
- ［**HTML を比較する**］: HTML コードの差異のみ表示します。
- ［**テキストを比較する**］: テキスト，HTML，バイナリの差異をすべて表示する。

注： ほとんどの場合，標準の HTML を比較する方法を使用することをお勧めします。スクリプトに HTML タグ以外のタグが含まれている場合は，テキストを比較する方法を使用できます。

［**詳細設定**］:［詳細相関］ダイアログ・ボックスを開きます。

［詳細相関］ダイアログ・ボックス

このダイアログ・ボックスでは，区切り文字として扱う文字を指定できます。



［**区切り文字として扱う文字**］：1 つまたは複数の標準設定以外の区切り文字を指定します。

［**追加区切り文字**］：復帰，復帰改行，タブ文字など，標準設定の区切り文字を指定できます。設定を変更するには，区切り文字の横のチェック・ボックスをクリアします。

［**. . . 文字以下の差異を無視する**］：相関を行うしきい値を指定できます。VuGen は記録時のスナップショットを再生時のスナップショットと比較する検索処理で差異を検出します。差異の文字数がしきい値以上でない限り相関は行われません。標準の値は 4 文字です。

相関オプションの設定

セッションの記録を開始する前に，相関オプションを設定します。

相関のオプションを設定するには，次の手順で行います。

1 ［**オプション**］＞［**一般**］を選び，［**相関**］タブを選択します。

2 再生情報をスナップショットとして保存するには，［**再生時に相関情報を保存する**］オプションを選択します。再生時のスナップショットのいずれかと記録時のスナップショットを比較できます。

3 スクリプト内を検索して相関を見つけるかどうかを尋ねるメッセージを表示するには，［**仮想ユーザ再生後、相関検索結果のポップアップ ウィンドウを表示する**］を選択します。これによって，再生後，スクリプト内を検索する前にポップアップ・ウィンドウが表示されます。

4 スクリプトに現れる差異を表示するには，［**アクションで表示される差異のみを表示する**］を選択します。現在のアクションにはないデータからパラメータを作成する場合は，このオプションを無効にします。

5 ［**スナップショット ビューアで編集および Java アプレット実行を可能にする**］を選択して，スナップショット・ウィンドウでアプレットと Java スクリプトを実行できるようにします。

6 画像をスナップショット・ビューアに表示するよう VuGen に指示するには，［**スナップショット ビューアで画像をダウンロードする**］オプションを選択します。

7 比較の方法として，［**HTML を比較する**］または［**テキストを比較する**］（HTML の要素以外に対してのみ）を選択します。

8 区切り文字を設定するには，［**詳細設定**］をクリックして，［詳細相関］ダイアログ・ボックスを開きます。

9 区切り文字として扱う文字をすべて指定します。

10 ［**追加区切り文字**］セクションで必要なオプションを選択して，標準で使用する 1 つまたは複数の区切り文字を指定します。

11 ［**X 文字以下の差異を無視する**］ボックスで，相関のしきい値を指定します。VuGen は記録時のスナップショットを再生時のスナップショットと比較する検索処理で差異を検出します。差異の文字数がしきい値以上でない限り相関は行われません。

12 ［**OK**］をクリックして詳細相関設定を受け入れ，ダイアログ・ボックスを閉じます。

13 ［一般オプション］ダイアログ・ボックスの［**OK**］をクリックし，［相関］の設定を受け入れてダイアログ・ボックスを閉じます。

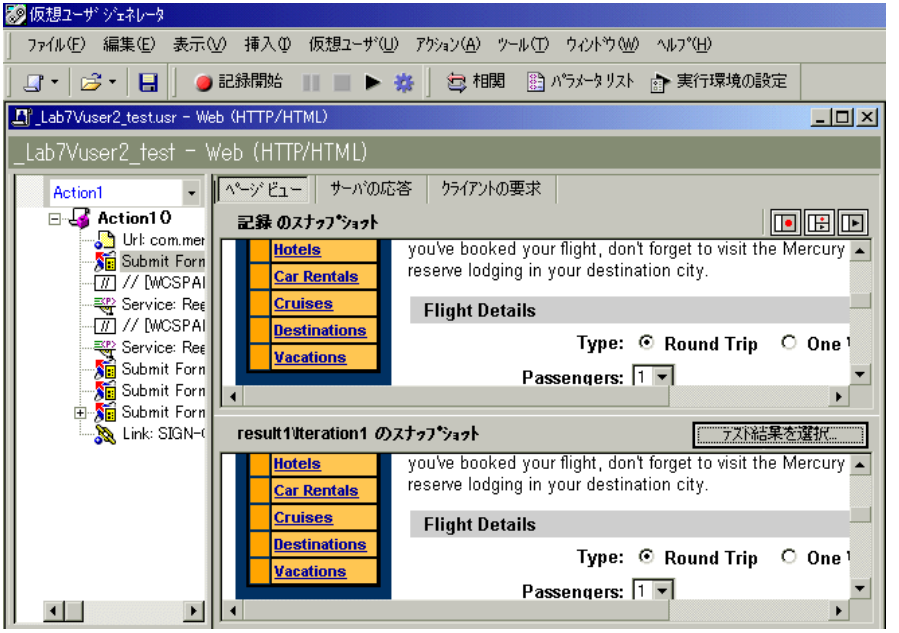
相関の検索の実行

VuGen のスナップショット・ウィンドウを使って，スクリプト内のどの値が動的で相関が必要かを判断します。次の節では，スクリプト内を自動的に検索して差異を見つける方法と，VuGen を使って必要な相関を行う方法について説明します。

スクリプト内を検索して相関を実行するには，次の手順で行います。

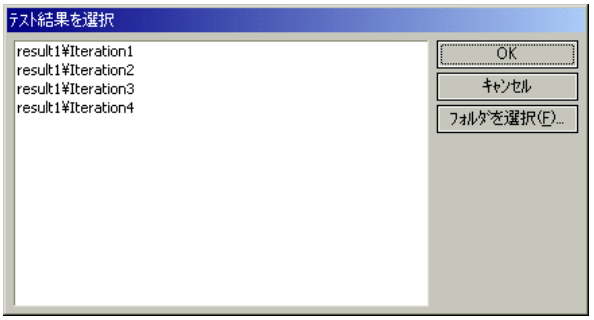
1 スクリプトを開き，ツリー・ビューで表示します（[**表示**] > [**ツリービュー**]）。スナップショットを表示します（[**表示**] > [**スナップショット**] > [**スナップショットを表示**]）。

2 左側の表示枠のツリー・ビューでスクリプトのステップを選択します。右側の表示枠に，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が開きます。



3 2 回目以降の再生時のスナップショットを使用するには，[**テスト結果を選択**] をクリックします。[テスト結果を選択] ダイアログ・ボックスが開き，スナップショット・ファイルが格納されているフォルダが表示されます。通常，

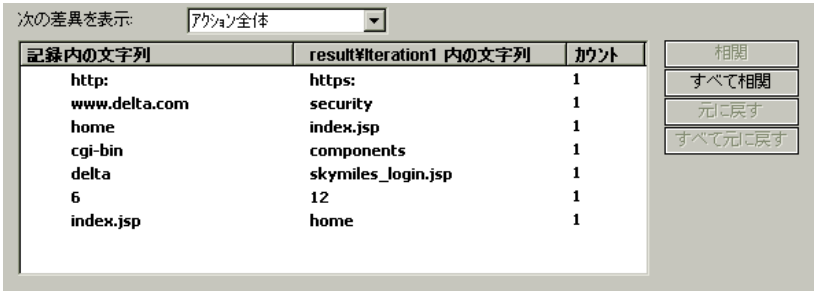
スクリプトのフォルダの下には［**result**］および［**Iteration**］フォルダが表示されます。



4 スクリプトのサブ・フォルダ以外のフォルダにあるスナップショット・ファイルを選択するには，［**フォルダを選択**］をクリックします。目的のフォルダの場所を見つけ，［**OK**］をクリックします。

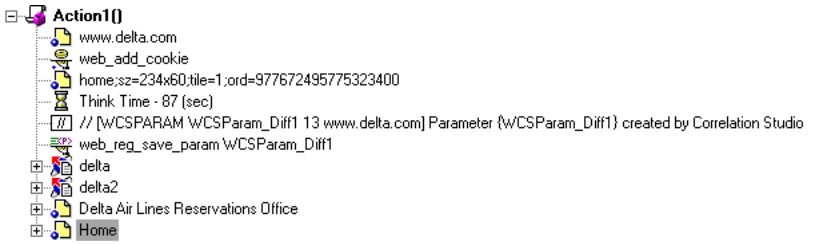
5 HTML コードを表示するには，［**サーバの応答**］タブをクリックします。「Body」の分岐を開きます。ページ・ビューに戻るには，［**ページ ビュー**］タブをクリックします。

6 [**仮想ユーザ**] > [**アクション内で相関をスキャン**] を選択するか，[**アクション内で相関をスキャン**] ボタンをクリックします。VuGen はスクリプト内を検索して，相関が必要な動的データを見つけ，それらを [相関結果] ウィンドウに表示します。



7 仮想ユーザ・スクリプトの特定のステップでの差異を表示するには，VuGen のツリー・ビューでステップを選び，[**次の差違を表示**] リスト・ボックスで [**現在のステップのみ**] を選択します。すべての差異を表示するには，[**次の差異を表示**] リスト・ボックスで [**スクリプト全体**] を選択します。

8 すべての差異を相関させるには，[**すべて相関**] をクリックします。特定の差異を相関させるには，差異を選択して [**相関**] をクリックします。VuGen によって，相関された差異の横に緑のチェック・マークが付けられ，仮想ユーザ・スクリプトに **web_reg_save_param** 関数が挿入されます。



9 相関を取り消すには，差異を選択して [**元に戻す**] をクリックします。すべての相関を取り消すには，[**すべて元に戻す**] をクリックします。

10 [**ファイル**] > [**上書き保存**] を選択して，変更を保存します。

手作業による相関

Web 仮想ユーザの場合，VuGen の自動相関またはルールに基づいた相関では，通常はスクリプトを正常に実行できるよう，スクリプトの動的関数を相関させます。VuGen のスナップショットの比較機能を使って，記録セッションの後に相関を行うこともできます。詳細については，第 42 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。

自動相関が適用されなかったワイヤレス仮想ユーザやその他の仮想ユーザ・スクリプトには，VuGen で手作業によるスクリプトの相関が行えます。コード相関関数を追加してスクリプトを手作業で相関します。パラメータにデータを動的に保存する関数は，**web_reg_save_param** です。

スクリプトを実行すると，**web_reg_save_param** 関数は，以降にアクセスする HTML ページ内を検索します。左と右の境界を指定すると，VuGen によってこれらの境界の範囲内でテキストが検索されます。テキストが見つかると，テキストはパラメータに保存されます。

関数の構文は次のとおりです。

int **web_reg_save_param** (const char ***mpszParamName**, **＜属性のリスト＞** , LAST);

使用できる属性を次の表に示します。属性値の文字列（Search=all など）では，大文字と小文字は区別されません。

NotFound
境界が見つからず，空の文字列が生成されたときの処理方法。標準設定の「ERROR」では，境界が見つからない場合，LoadRunner によりエラーが発行されます。「EMPTY」に設定した場合，エラー・メッセージは発行されず，スクリプトの実行が継続されます。スクリプトに対して［エラーでも処理を継続する］を有効にしている場合は，NOTFOUND を「ERROR」に設定していても，境界が見つからない場合に スクリプトは継続されます。ただし，エラー・メッセージは詳細ログ・ファイルに記録されます。

LB
パラメータまたは動的データの左の境界。このパラメータは，空でない，NULL で終わる文字列でなければなりません。境界のパラメータでは，大文字と小文字が区別されます。大文字と小文字の区別をしないようにするには，境界の後に「/IC」を追加します。バイナリ・データを指定するには，境界の後に「/BIN」を指定します。

RB

パラメータまたは動的データの右の境界。このパラメータは，空でない，NULL で終わる文字列でなければなりません。境界のパラメータでは，大文字と小文字が区別されます。大文字と小文字の区別をしないようにするには，境界の後に「/IC」を追加します。バイナリ・データを指定するには，境界の後に「/BIN」を指定します。

RelFrameID

要求された URL を基準にした HTML ページの階層レベル。値は，ALL か数値です。

Search

検索の範囲（区切り文字で区切られたデータを検索する場所）。値は，Headers（ヘッダーだけを検索），Body（ヘッダーでなく本体のデータだけを検索），ALL（本体とヘッダーを検索）のいずれかです。標準の値は ALL です。

ORD

このパラメータは省略可能で，何回目に出現する検索内容かを序数で示します。標準の序数は 1 です。「All」を指定すると，パラメータの値が配列に保存されます。

SaveOffset

見つかった値において，パラメータに保存する部分の文字列の開始位置を示すオフセットです。標準設定は 0 です。オフセットの値は負でない数字でなくてはなりません。

Savelen

パラメータに保存する部分文字列の長さ。部分文字列は，見つかった値においてオフセット位置から始まります。標準設定は -1 で，文字列の末尾までを保存することを示します。

Convert

データに適用する変換方式。

HTML_TO_URL : HTML エンコードされたデータを URL エンコード形式に変換する。

HTML_TO_TEXT : HTML エンコードされたデータをプレーン・テキスト形式に変換する。

スクリプトを手作業で相関させるには，次の手順で行います。

1 動的データを含むステートメントと，データの境界を示すパターンを探します。詳細については，580 ページ「動的文字列の境界の定義」を参照してください。

2 スクリプトの中で，動的データを独自のパラメータ名に置き換えます。詳細については，下記を参照してください。

3 スクリプト中の，動的データが含まれるステートメントの前に，**web_reg_save_param** 関数を追加します。詳細については，577 ページ「相関関数の追加」または「**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。

動的データのパラメータへの置き換え

まず，実際の動的データを探します。記録されたステートメントで動的データが見つかったら，スクリプト全体を対象に動的データを検索し，パラメータに置き換えます。パラメータに名前を付け，**{param_name}** のように中括弧で囲みます。1 つのスクリプトには，最大 64 個のパラメータを設定できます。

動的データをパラメータに置き換えるには，次の手順で行います。

VuGen のメイン・ウィンドウで［**編集**］＞［**置換**］を選択して，［検索と置換］ダイアログ・ボックスを表示します。動的データをスクリプト全体から検索して，パラメータに置き換えます。

相関関数の追加

スクリプトの動的データを保存するには，**web_reg_save_param** ステートメントを挿入します。この関数は，再生中に動的データの実行時の値を保存するパラメータを作成するように VuGen に指示します。

スクリプトを実行すると，**web_reg_save_param** 関数は，以降にアクセスする HTML ページ内を検索します。左の境界，任意の文字列，右の境界が並んでいる文字列を検索します。文字列が見つかると，VuGen は左と右の境界の間にある文字列を，関数の引数に指定したパラメータに代入します。
web_reg_save_param は，指定された出現回数分の文字列を見つけると，それ以上は HTML ページ内を検索せず，スクリプト内の次のステップから実行を継続します。

Web 仮想ユーザの相関の例

次のように，スクリプトに動的なセッション ID が含まれているとします。

```
web_url("FirstTimeVisitors", "
URL=/exec/obidos/subst/help/first-time-visitors.html/002-8481703-
        4784428>Buy books for a penny ",
      "TargetFrame=",
      "RecContentType=text/html",
      "SupportFrames=0",
      LAST);
```

上記のステートメントの前に，次のように **web_reg_save_param** ステートメントを挿入します。

```
web_req_save_param ("user_access_number", "NOTFOUND=ERROR",
"LB=first-time-visitors.html/","RB=>Buy books for a penny" , "ORD=6",
LAST );
```

相関ステートメントを実装した後，変更後のスクリプトは次のようになります。**user_access_number** は動的なデータを表すパラメータの名前です。

```
web_url("FirstTImeVisitors", "
URL=/exec/obidos/subst/help/first-time-
"       ";visitors.html/{user_access_number}Buy books for a penny ",
      "TargetFrame=",
      "RecContentType=text/html",
      "SupportFrames=0",
      LAST);
```

注：各相関関数は，以降の HTTP 要求について，動的データを一度だけ取得します。スクリプト内の後方にある別の HTTP 要求によって新しい動的データが生成される場合は，相関関数をもう 1 つ挿入しなければなりません。

ワイヤレス仮想ユーザの相関の例

次のように，スクリプトに動的なセッション ID が含まれていると仮定します。

```
web_url("login.po;sk=IuZSuuRlHUMnpF-wpK8PzEpy(1YOSBSMy)",
    "URL=http://room33.com/portal/login.po;sk=IuZSuuRlHUMnpF-
      wpK8PzEpy(1YOSBSMy)",
    "Resource=0",
    "RecContentType=text/vnd.wap.wml",
    "Mode=HTML",
    LAST);
```

web_reg_save_param ステートメントを上記のステートメントの前に挿入し，動的な値をパラメータに置き換えます。次の例では，**web_reg_save_param** 関数を使って，ログイン ID 文字列を SK という変数に保存しています。**RB/BIN** 属性に従ってバイナリ・データを保存し，左の境界を「sk=」として設定しています。

```
web_reg_save_param(
   "SK",
"LB=sk=",
"RB/BIN=#login¥¥x00¥¥x01¥¥x03",
"Ord=1",
    LAST);


web_url("login.po;sk={SK}",
    "URL=http://room33.com/portal/login.po;sk={SK}",
    "Resource=0",
    "RecContentType=text/vnd.wap.wml",
    "Mode=HTML",
        LAST);
```

動的文字列の境界の定義

動的データの特定と指定は，以下のガイドラインに従って行います。

- 動的データの場所は，記録されたスクリプト内ではなく，必ず HTML コード内で探すようにします。
- 動的データのすぐ左側にある文字列を特定します。この文字列は動的データの左の境界を示します。
- 動的データのすぐ右側にある文字列を特定します。この文字列は動的データの右の境界を示します。
- **web_reg_save_param** は，指定された境界の間（境界を含まない）にある文字を検索し，左の境界の 1 バイト後から，右の境界の 1 バイト前までの情報を保存します。**web_reg_save_param** では，境界文字の重複はサポートされません。たとえば，入力バッファが {a{b{c} で，「{」が左の境界，「}」が右の境界の場合，検索に一致するのは「c」で，それ以外に一致するものはありません。この場合，左右の境界は検出されたが，境界の重複は認識されないので，「c」が唯一の一致項目となります。

  標準では，境界文字列は最大 256 文字です。最大文字数を増やすには，スクリプトに **web_set_max_html_param_len** 関数を挿入します。たとえば，次の関数は最大文字数を 1024 文字に増やします。

  ```
  web_set_max_html_param_len("1024");
  ```

第 43 章

XML ページのテスト

VuGen の Web 仮想ユーザは XML コードを含む Web ページをサポートします。

本章では，次の項目について説明します。

- XML の URL ステップとしての表示
- XML をユーザ定義の要求として挿入
- XML ユーザ定義要求のステップの表示

以降の情報は，Web 仮想ユーザ・スクリプトを対象とします。

XML ページのテストについて

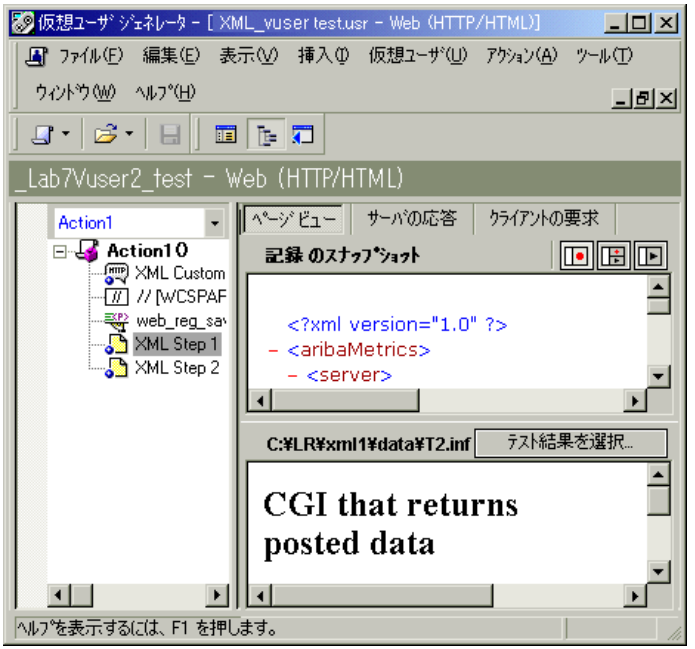
VuGen は，Web ページに含まれる XML コードの記録および再生をサポートしています。

XML コードは，スクリプトに通常の URL ステップまたはユーザ定義の要求として現われます。VuGen は XTML を検出し，ユーザがそれぞれの文書型定義（DTD）およびそのエンティティと属性を参照できるようにします。DTD は色分けされているので，要素が容易に識別できます。DTD のツリー・ビューの分岐の展開と折りたたみも可能です。VuGen は，**RecContentType** 属性によって指定される MIME タイプが **text/xml** に設定されていない場合でも XML を検出できます。

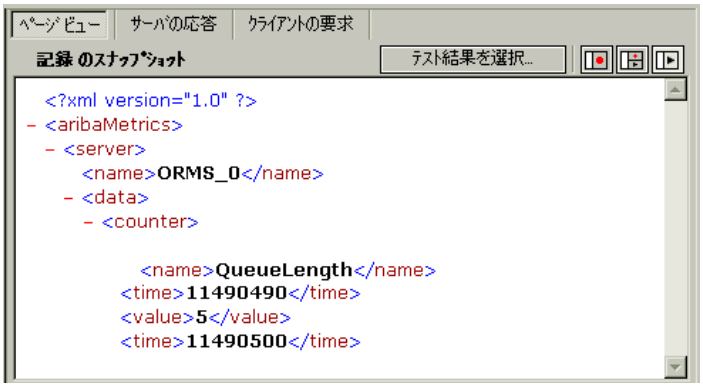
DTD を展開する場合には，属性の値をパラメータ化できます。また標準の相関関数を使って相関できるように値を保存できます。相関関数の詳細については，「**オンライン関数リファレンス**」（［**ヘルプ**］＞［**関数リファレンス**］）を参照してください。

XML の URL ステップとしての表示

XML コードを含むページをテストする 1 つの方法は，VuGen で記録することです。まず，XML ページを通常の Web ページと同じ方法で記録します。VuGen は，DTD およびすべての XML 要素を記録します。XML ページのスナップショットは作成されませんが，XML ステップごとに，XML コードがスナップショット表示枠に表示されます。



VuGen が作成した展開可能な DTD 階層は，色分けされてスナップショット表示枠に表示されます。項目の分岐を展開するにはプラス（+）記号をクリックし，折りたたむにはマイナス（−）記号をクリックします。XML タグは茶色，値は黒で表示されます。



定数値をパラメータで置き換えるには，値を選択して，右クリックし［**パラメータで置換**］を選択します。パラメータ化の標準手続きに従います。詳細については，パラメータ化に関する章を参照してください。

XML をユーザ定義の要求として挿入

XML コードをユーザ定義の要求として挿入して，XML ページをテストすることもできます。このモードでは，DTD の要素が［**ユーザ定義要求プロパティ**］ボックス内にテキスト形式または XML 形式で表示されます。

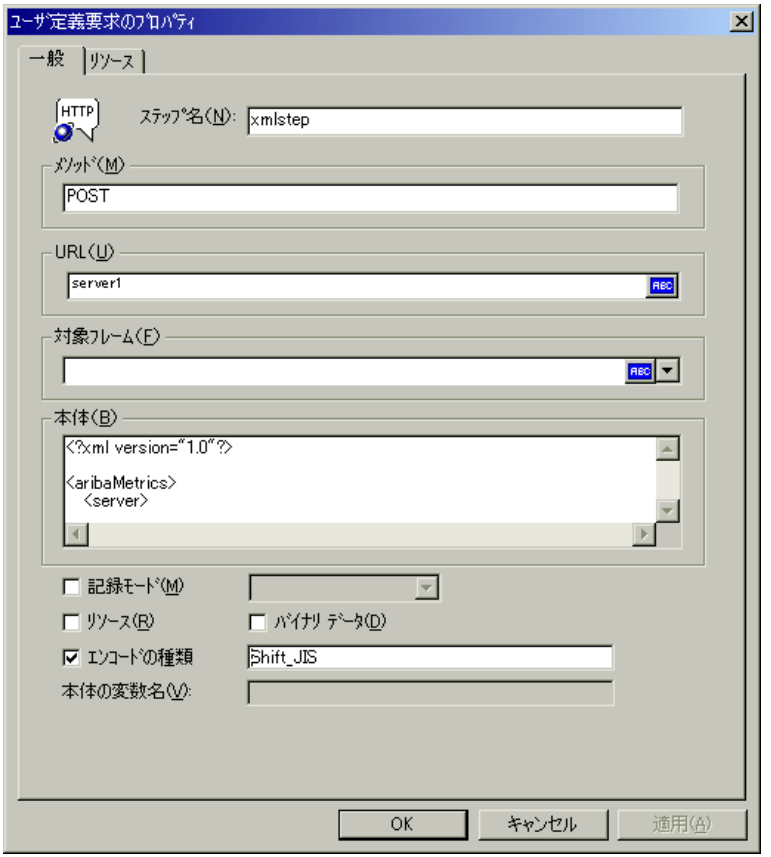
XML コードをユーザ定義の要求として追加するには，次の手順で行います。

1 ツリー・ビュー・モードでスクリプトを表示し，希望する場所にカーソルを置き，［**挿入**］＞［**新規ステップ**］を選択します。［ステップの追加］ダイアログ・ボックスが開きます。

2 ［**ユーザ定義要求**］を選択します。［**OK**］をクリックします。［ユーザ定義要求のプロパティ］ダイアログ・ボックスが開きます。

3 ステップ名，メソッド（GET または POST），URL，対象フレーム（任意）を入力します。

4 XML コードをブラウザまたはエディタからコピーし，［ユーザ定義要求のプロパティ］ボックスの［**本体**］セクションに貼り付けます。



5 ［**記録モード**］，［**リソース**］，［**バイナリ データ**］などの再生に関するオプションを選択します。詳細については，第 40 章「Web とワイヤレス仮想ユーザ・スクリプトの変更」を参照してください。

6 ［**OK**］をクリックします。ユーザ定義の要求のステップがスクリプトに挿入されます。

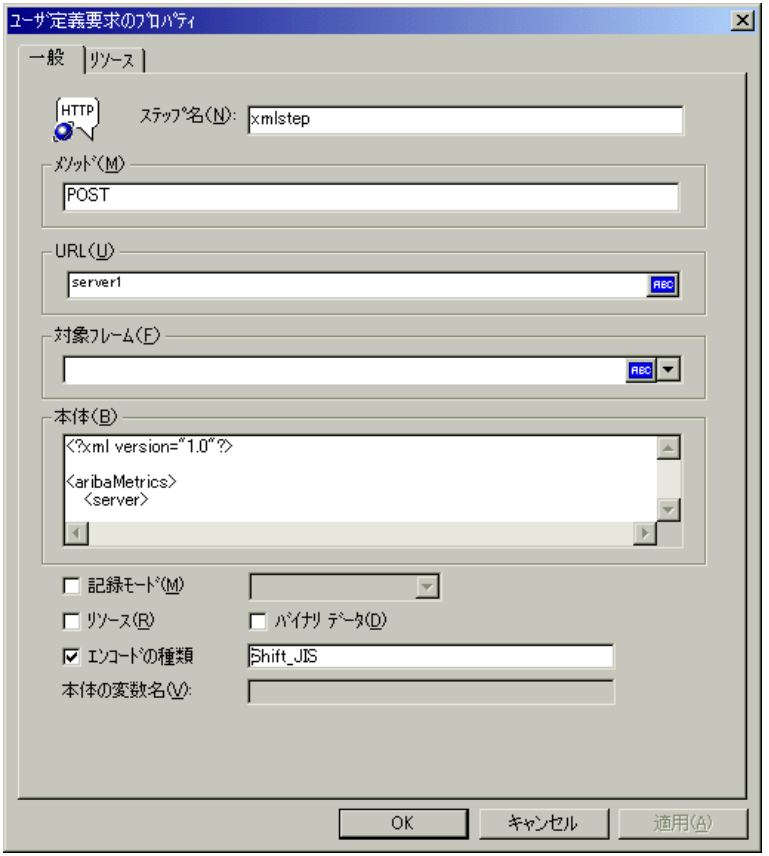
XML ユーザ定義要求のステップの表示

ユーザ定義の要求のステップとして挿入された XML コードは，いつでも表示したり変更したりすることができます。VuGen のビューアを使って，DTD の階層を表示し，必要に応じて要素の分岐を展開したり折りたたんだりできます。

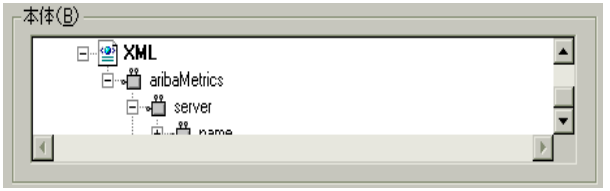
ユーザ定義の要求のステップの XML コードを表示するには，次の手順で行います。

1 スクリプトをツリー・ビューで表示し，XML コードを表示するステップを選択します。

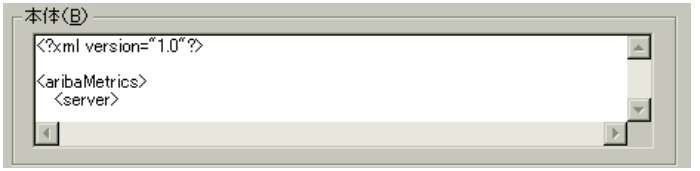
2 右クリックして表示されるメニューから［**プロパティ**］を選択します。［ユーザ定義要求のプロパティ］ダイアログ・ボックスが開きます。



ダイアログ・ボックスの最下部に XML コードが表示されます。
RecContentType 属性が「**text/xml**」に設定されていると，標準設定では，コードが XML 形式の階層として表示されます。このモードのときには，XML コードの編集はできません。

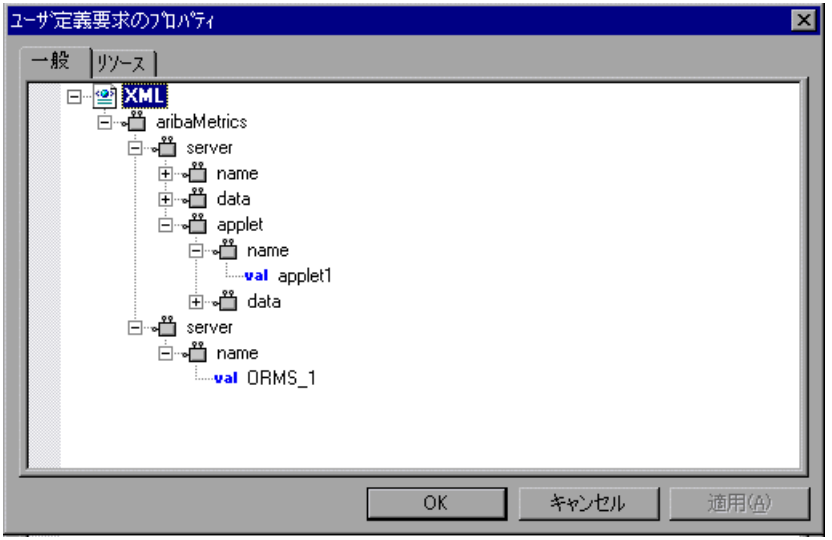


RecContentType 属性が「**text/xml**」以外に設定されているときには，コードはテキスト形式で表示されます。このモードのときには，XML コードは編集可能です。



3 テキストと XML の表示を切り替える場合は，右クリックして表示されるメニューから［**XML として表示**］または［**テキストとして表示**］を選択します。

4 XML 表示になっている場合，ウィンドウを大きくしてコードを表示できます。右クリックして表示されるメニューから［**拡張表示**］を選択します。ダイアログ・ボックスの表示に戻るには，右クリックして表示されるメニューから［**標準表示**］を選択します。



第 44 章

レポートを使った仮想ユーザ・スクリプトのデバッグ

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

本章では，次の項目について説明します。

- 結果サマリ・レポートについて
- レポート情報のフィルタリング
- 実行結果の管理

注：VuGen のすべての Web レポート機能を使用するには，Microsoft Internet Explorer 4.0 以上を使用することをお勧めします。

以降の情報は，Web 仮想ユーザ・スクリプトを対象とします。

レポートを使った仮想ユーザ・スクリプトのデバッグ

VuGen の Web 仮想ユーザ・スクリプトをデバッグするとき，スクリプトの実行中に**結果サマリ**・レポートを生成するかどうかを指定します。結果サマリ・レポートには，仮想ユーザが訪れたすべての Web ページおよび仮想ユーザが実行した検査の詳細が含まれます。この情報は，Web 仮想ユーザ・スクリプトのデバッグに有用です。VuGen を使って仮想ユーザ・スクリプトを実行する方法の詳細については，第 11 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。

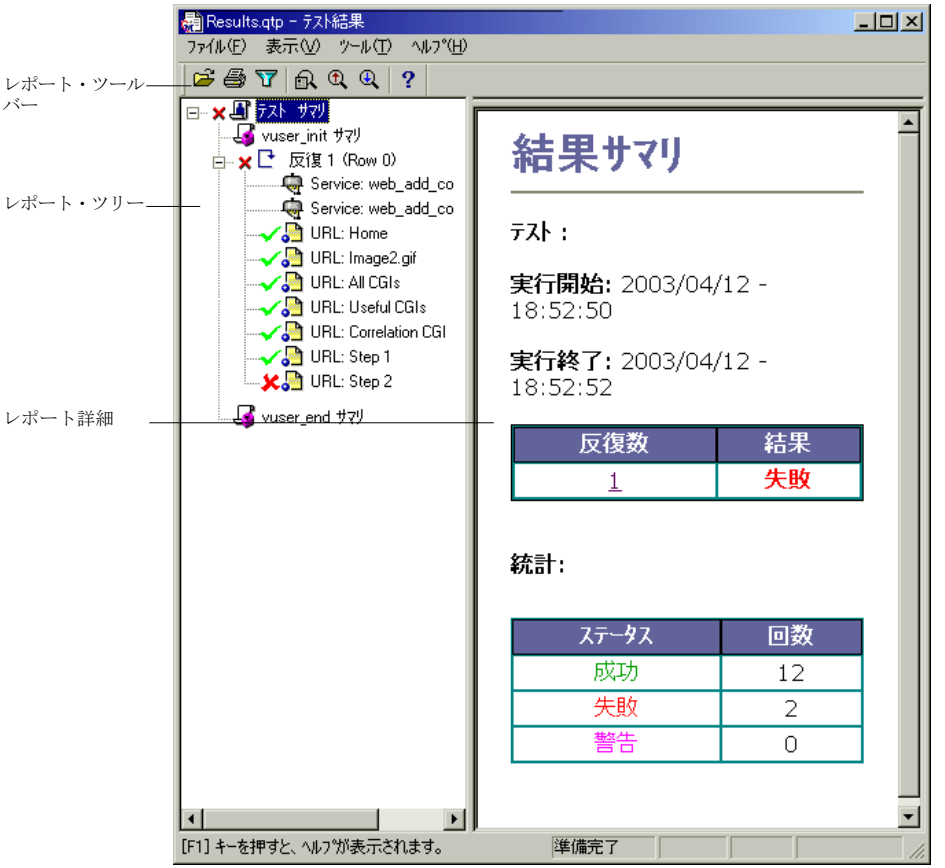
VuGen を使って Web 仮想ユーザ・スクリプトを実行した後に，結果サマリ・レポートが表示されます。

Microsoft Internet Explorer 4.0（またはそれ以上）がマシンにインストールされている場合には，結果は VuGen レポート形式（拡張子 **.qtp**）で生成されます。この結果は仮想ユーザ・ジェネレータの［テスト結果］ウィンドウに表示されます。VuGen の［テスト結果］ウィンドウは，インタフェースが洗練されており追加機能も提供されるので，この環境をお勧めします。

［表示］オプション（[**ツール**］＞［**一般 オプション**]）で，結果サマリ・レポートを生成するかどうかを指定します。レポート生成するように指定した場合は，スクリプトの実行後にレポートを自動的に開くかどうかも指定します。［表示］オプションの設定の詳細については，第 11 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。

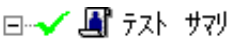
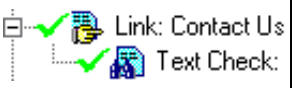
結果サマリ・レポートについて

仮想ユーザ・スクリプトの実行後，結果サマリ・レポートが表示されます。レポートには，スクリプトの実行結果のサマリが表示されます。



- 左側の表示枠には，結果をグラフィカルに示す「**レポート・ツリー**」が表示されます。レポート・ツリーでは，成功したステップを緑色のチェック印で，失敗したステップが赤色の「×」印で示されます。
- 右側の表示枠には「**レポートの詳細**」が表示されます。ここには，スクリプト実行全般についてのサマリや，レポート・ツリーで選択された分岐についての補助的な情報が表示されます。

レポート・ツリーの分岐を 1 つ選択して，その項目の情報を表示します。

ツリーの分岐	表示内容
テスト名 	スクリプトの実行全般についての結果サマリ。
テストの反復 	1 つの反復についての実行サマリ。
テストのステップまたはチェック 	仮想ユーザ・スクリプト内で選択されたステップやチェックに対応する Web ページ。

レポート・ツリー内の分岐を展開したり，折りたたんだりすることで，ツリーに表示される情報の詳しさを変更することができます。

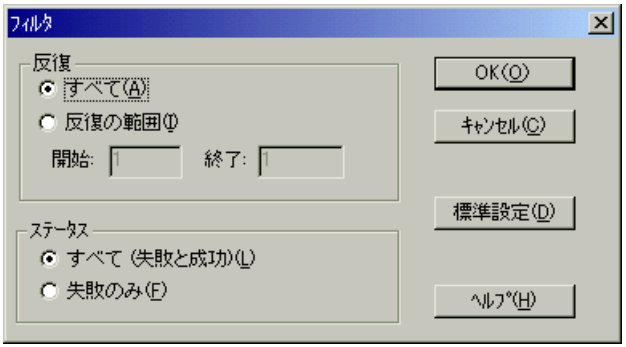
- 分岐を折りたたむには，折りたたみたい分岐の左側にあるマイナス（−）記号をクリックします。レポート・ツリーは分岐を隠し，マイナス（−）記号はプラス記号（+）に変わります。
- レポート・ツリー内のすべての分岐を折りたたむには，［**表示**］＞［**すべて折りたたみ**］を選択します。
- 分岐を展開して表示するには，表示したい分岐の左側にあるプラス（+）記号をクリックします。レポート・ツリーは分岐を展開して表示し，プラス（+）記号はマイナス（−）記号に変わります。
- レポート・ツリー内のすべての分岐を表示するには，［**表示**］＞［**すべて展開**］を選択します。

レポート情報のフィルタリング

VuGen 結果サマリ・レポートに表示される情報にフィルタを適用することができます。フィルタによる選別は，反復回数か，反復のステータスに基づきます。

レポートに含まれる情報にフィルタを適用するには，次の手順に従います。

1 **レポート**・ツールバーの［**フィルタ**］ボタンをクリックするか，［**表示**］＞［**フィルタ**］を選択します。［フィルタ］ダイアログ・ボックスが開きます。



2 必要なオプションを設定します。標準設定のフィルタ・オプションは，図に示したとおり［**すべて**］です。

レポートの出力対象を指定の反復回数に制限するには，［**反復**］セクションで［**反復の範囲**］を選択して，［**開始**］ボックスと［**終了**］ボックスで範囲を指定します。

失敗した反復のみレポートを出力する場合には，［**ステータス**］セクションで［**失敗のみ**］を選択します。

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

実行結果の検索

テスト結果内で最終のステータス（**失敗**，**成功**，**完了**，**警告**）を指定して結果ステップの検索を行うことができます。検索時には，複数のステータスを指定できます。

ステータスを指定してステップを検索するには，次の手順で行います。

1 ［**ツール**］＞［**検索**］を選択するか，［**レポート**］ツールバーで［**検索**］ボタンを選択します。［文字列検索］ダイアログ・ボックスが開きます。

2 見つけたいステップのステータス（複数も可）を選択します。

3 検索方向を**上向き矢印**または**下向き矢印**で選択します。

4 ［**次を検索**］をクリックします。次の一致にカーソルが移動します。

5 検索を繰り返すには，［**次を検索**］ボタンをクリックします。

実行結果の管理

結果サマリ・レポートを開いたり，印刷したり，終了したりするには，［ファイル］メニュー内のコマンドを使います。

結果サマリ・レポートの設定の詳細については，第 11 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」の 161 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。

結果サマリ・レポートを開く

Web 仮想ユーザ・スクリプトを実行すると，VuGen は結果サマリ・レポート・ファイルをスクリプト・フォルダの下の結果フォルダに保存します。レポート・ファイルの名前の形式は＜**スクリプト名**＞ **.qtp** です。

結果サマリ・レポートを開くには，次の手順で行います。

1 ［**ファイル**］ > ［**開く**］を選択するか，［**レポート**］ツールバーの［**開く**］ボタンをクリックします。［結果ファイルを選択］ダイアログ・ボックスが開きます。

2 開くレポート・ファイル名を選択して，［**開く**］をクリックします。

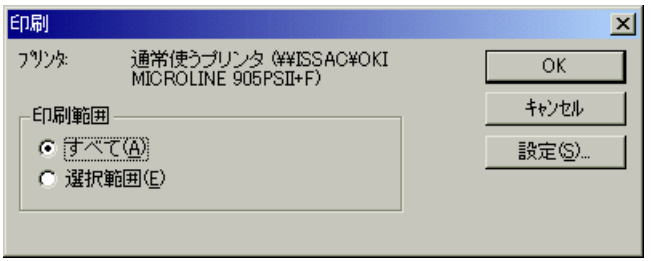
3 最近表示したレポートを開くには，［**ファイル**］メニューのレポート履歴リストからそのレポートを選択します。

レポート結果の印刷

テスト結果サマリ・レポートは印刷が可能です。

テスト・サマリ・レポートを印刷するには，次の手順で行います。

1 ［**ファイル**］ > ［**印刷**］を選択するか，［**レポート**］ツールバーの［**印刷**］ボタンをクリックします。［印刷］ダイアログ・ボックスが開きます。



2 ［**印刷範囲**］ボックスから範囲を選択します。

［**すべて**］：レポート全体を印刷します。反復の中の各ステップに対応した Web ページも含まれます。

［**選択範囲**］：レポート・ツリー内で選択した分岐を印刷します。

3 ［**OK**］をクリックして，印刷します。

4 プリンタの設定オプションを変更するには，［**ファイル**］ > ［**印刷設定**］を選択して，［プリンタの設定］ダイアログ・ボックスで設定を変更します。

結果サマリ・レポートを閉じる

テスト・サマリ・レポートを閉じるには，［**ファイル**］ > ［**終了**］を選択します。［テスト結果］ウィンドウが閉じます。

第 45 章

Web 仮想ユーザのヒント（パワー・ユーザ向け）

本章では，Web 仮想ユーザの「**上級ユーザ**」からよく出される質問に答えます。質問と回答は以下の項に分類されています。

- セキュリティの問題
- クッキーの取り扱い
- 実行時ビューア（オンライン・ブラウザ）
- ブラウザ
- 設定
- 互換性について

以降の情報は，Web 仮想ユーザ・スクリプトを対象とします。

セキュリティの問題

質問 1：Web 仮想ユーザはセキュアなトランザクション（HTTPS）とセキュアでないトランザクション（HTTP）の両方をサポートしていますか？

回答：はい。Web 仮想ユーザはセキュアなトランザクション（HTTPS）とセキュアでないトランザクション（HTTP）の両方をサポートしています。

質問 2：Web 仮想ユーザはデジタル証明書をサポートしていますか？

回答：はい。Web 仮想ユーザはクライアント側のデジタル証明書をサポートしています。デジタル証明書は，電子的メッセージのセキュリティを強化するために付加されます。デジタル証明書の最も一般的な用途としては，メッセージの送信元の出所証明や，受信者が返信するために使用する所定のエンコード方式を渡すためなどに使われています。

VuGen は，クライアント側のデジタル証明書をサポートしていますが，次のような制約があります。

- **記録**：記録：クライアント側の証明書は，記録中に実際に使われたブラウザに関係なく，常に IE（Internet Explorer）のデータベースから取得されます。したがって，IE 以外のアプリケーションやブラウザで記録した場合，まず，記録を行うブラウザから証明書をエクスポートし，それを IE にインポートしなければなりません。IE に証明書をインポートするときに，秘密鍵をエクスポート可能にしなければなりません。

- **記録**：LoadRunner 7.0 より以前のバージョンでは，クライアントの証明書が使われると，**web_set_certificate** が生成されます。この関数の唯一の引数は，証明書リストの中の証明書番号（序数）です。WinInet モードの場合にのみ，この関数を再生できます。

  LoadRunner7.0 以降のバージョンでは，**web_set_certificate_ex** が生成されます。この関数の追加のパラメータは，署名を含むファイルのパスです。証明書ファイルは記録中に自動的に生成され，仮想ユーザ・スクリプトと一緒に保存されます。WinInet 再生モードを使用するときは常に，最初のパラメータが使用されます。ソケット再生（標準設定）では，2 番目のパラメータ（証明書ファイル）が使用されます。たとえば，秘密鍵がエクスポートできないなどの理由で，特定の証明書をダンプすることができない場合，**web_set_certificate_ex** がファイル名なしで生成されます。この場合には，WinInet 再生モードを使用しなければなりません。

  再生：**web_set_certificate_ex** が使用されていて，ファイル名の引数がある場合は，ソケット再生でのみ使用でき，ロード・ジェネレータ・マシンでの特別な設定は不要です。**web_set_certificate**，またはファイル名のない **web_set_certificate_ex** が使われている場合には，WinInet ベースの再生でのみ使用可能です。この場合，記録用のマシン上にすべての証明書を，**証明書リストに記載されている順番で**インストールする必要があります。これは，エクスポートとインポートによって行います。

質問 3：SSL サイトにアクセスする仮想ユーザ・スクリプトを記録すると，数多くの警告ポップアップ・メッセージが表示されます。これらのメッセージは表示が必要なのですか？もしそうなら，どのように対処すればよいのでしょうか？

回答：SSL サイトへのアクセスが記録できるように，VuGen ではオリジナルのサーバ証明書に代えて，VuGen 独自のサーバ証明書を提供します。これは次の 2 つのセキュリティ違反になります。

- 発行された証明書はユーザが接続しているサイト用のものではない。

➤ 発行された署名は未知の認証機関によるものである。

これらのセキュリティ違反があるため，記録用ブラウザによって警告ポップアップ・メッセージが表示されます。

使用しているブラウザが，Netscape 3.0 以上または Internet Explorer 4.0 以上であれば，これらの警告メッセージを無視するオプションがあります。これらのメッセージは無視しても構いません。

注：ポップアップ・メッセージは，スクリプトを記録するときにだけ表示され，スクリプトを実行するときには表示されません。ポップアップ・メッセージの一部（全部ではありません）を表示しないようにできます。

質問 4：IE や Netscape ではない Web アプリケーションを使っています。認知されていない証明書を使ってセキュア・サイトにアクセスすると，アプリケーションは自動的に処理を中断します。このようなアプリケーションを記録することはできますか？

回答：認知されていない証明書でセキュア・サイトにアクセスすると，IE と Netscape は警告を出します。一部のブラウザやアプリケーションは，認知されていない証明書に対して警告を出さずに，単にセキュア・サイトへの接続を遮断します。こうしたサイトを記録するには，証明書と鍵の **pem** を入手し，それらを LoadRunner¥bin フォルダの下の **certs** ディレクトリに追加します。次に，**index.txt** ファイルのリストに，**pem** ファイルを既存のエントリと同じように追加します（ホスト名とポートのセクション名の後に **pem** のファイル名を指定します）。

```
[demoserver:443]
Certfile=xxx.pem
Keyfile=yyy.pem
```

質問 5：VuGen は 128 ビットの暗号化をサポートしていますか？

回答：Netscape（4.5 以上）と Internet Explorer（5.0 以上）はどちらも，128 ビットの暗号化をサポートしています。現時点では Netscape がサポートしているのは Verisign のみです。

質問 6：VuGen は Internet Explorer 用のクライアント側の証明書をサポートしていますか？

回答：はい。VuGen は Internet Explorer 用のクライアント側の証明書をサポートしています。

質問 7：VuGen は Netscape 用のクライアント側の証明書をサポートしていますか？

回答：いいえ。VuGen がサポートしているのは Internet Explorer 用の証明書だけです。Netscape 用の証明書しか持っていない場合には，まず Netscape から必要な証明書をエクスポートして，それを Internet Explorer にインポートします。エクスポート / インポートを行うときは，必ずオリジナルの証明書リストと同じ順番で行ってください。この処理を，Web 仮想ユーザ・スクリプトの記録や実行で証明書が必要なすべてのコンピュータで行います。

質問 8：Web 仮想ユーザ・スクリプトを見て，仮想ユーザが通常の（HTTP）サーバと SSL 対応（HTTPS）サーバのどちらにアクセスするかを見分けられるでしょうか？

回答：場合によります。Web 仮想ユーザ・スクリプトはセキュアな要求とセキュアでない要求を区別しません。グラフィカル表示の仮想ユーザ・スクリプトは，同じアイコンをセキュアな要求とセキュアでない要求の両方に使います。また，テキスト形式の仮想ユーザ・スクリプトも同じ関数をセキュアな要求とセキュアでない要求の両方に使います。しかし，仮想ユーザ・スクリプトのステップに URL が含まれている場合には，URL を見て，そのステップが通常の（HTTP）サーバにアクセスするものか，SSL 対応（HTTPS）サーバにアクセスするものなのかを区別できるでしょう。

質問 9：Web 仮想ユーザがサポートしている認証には，どのような形式がありますか？

回答：Web 仮想ユーザは基本認証と NTLM 認証（NT challenge response authentication）をサポートしています。

クッキーの取り扱い

質問 10：VuGen は仮想ユーザ・スクリプトを記録するときにクッキーを処理しますか？

回答：VuGen は HTTP ヘッダーを通じて設定されたすべてのクッキーを自動的に処理します。しかし，VuGen は JavaScripts や **<meta- >** タグによって設定されるクッキーを常に正しく処理することはできません。詳細については，質問 14 を参照してください。

質問 11：Web 仮想ユーザ・スクリプトを実行する場合，仮想ユーザは，仮想ユーザ・スクリプトの記録時に使われたのと同じクッキーを再利用するのでしょうか？

回答：クッキーのタイプによります。クッキーはパーシステント・クッキーとセッション・クッキーの 2 つのカテゴリに分類されます。

パーシステント・クッキー	Web サーバにユーザを識別させる，テキストのみの文字列で，有効期限があります。パーシステント・クッキーは，ユーザのハードディスクに格納されます。
セッション・クッキー	Web サーバにユーザを識別させる，テキストのみの文字列で，現在のセッションに限り有効です。セッション・クッキーはユーザのハードディスクには格納されません。

Web 仮想ユーザ・スクリプトを記録するとき，VuGen はユーザのブラウザに送られてきたすべてのクッキーを検出します。VuGen は次のようにパーシステント・クッキーとセッション・クッキーを区別します。

パーシステント・クッキー

パーシステント・クッキーは，詳細が仮想ユーザ・スクリプトに直接記録されます。VuGen では **web_add_cookie** を使ってパーシステント・クッキーを仮想ユーザ・スクリプトに含めます。仮想ユーザ・スクリプトを実行すると，必要に応じてこれらのパーシステント・クッキーが使用されます。

セッション・クッキー

記録セッション中に使われたセッション・クッキーは保存されません。記録中はセッション・クッキーはキャッシュに格納され，記録を終了すると破棄されます。

仮想ユーザ・スクリプトを実行すると，仮想ユーザは Web サーバから受け取った新しいセッション・クッキーを使います。つまり，仮想ユーザはスクリプト記録時に生成されたセッション・クッキーをそのまま再利用することはありません。セッション・クッキーは仮想ユーザ・クッキー・キャッシュに格納され，仮想ユーザが終了すると破棄されます。仮想ユーザはこれらのセッション・クッキーを保存しません。

質問 12：仮想ユーザごとに，専用のクッキー・キャッシュがあるのでしょうか？

回答：はい。仮想ユーザごとに専用のクッキー・キャッシュがあります。したがって，複数の仮想ユーザが同じロード・ジェネレータで実行しているときでも，セッション・クッキーが共用されることはありません。

質問 13：記録した仮想ユーザ・スクリプトを実行するには，あらかじめその中のクッキーをパラメータ化しておかなければならないのでしょうか？

回答：場合によります。質問 11 で述べたように，VuGen はスクリプトを記録するときに，パーシステント・クッキーを仮想ユーザ・スクリプトの中にコピーします。仮想ユーザ・スクリプトを実行するときに，仮想ユーザは記録されたパーシステント・クッキーを使用します。各仮想ユーザに専用のパーシステント・クッキーが必要な場合には，仮想ユーザ・スクリプトの中でクッキーをパラメータ化する必要があります。

質問 14：Web 仮想ユーザは JavaScript で設定されたクッキーを処理しますか？

回答：VuGen は HTTP ヘッダーを通じて設定されたすべてのクッキーを自動的に処理します。しかし，VuGen は HTML ページ内の JavaScript によって設定されたクッキーを常に正しく処理できるわけではありません。JavaScript によって設定されたクッキーには，記録および再生時に独特の問題が生じます。

記録時

VuGen ではパーシステント・クッキーは仮想ユーザ・スクリプト内に記録されますが（**web_add_cookie** ステートメントを使用），セッション・クッキーは記録されません。しかし，JavaScript によって生成されたクッキーは，それがセッション・クッキーであったとしても，技術的制約により，パーシステント・クッキーとして記録されます。

回避手段：仮想ユーザ・スクリプトを記録した後で，セッション・クッキーを設定する **web_add_cookie** ステートメントをすべて相関させる相関ステートメントを挿入します。パーシステント・クッキーを設定する **web_add_cookie** ステートメントを削除してはなりません。

再生

Web 仮想ユーザは HTML ページに埋め込まれた JavaScript は実行しません。したがって，JavaScript によるセッション・クッキーは，仮想ユーザの実行では作成されません。

回避手段：仮想ユーザ・スクリプトを記録した後で，スクリプトに相関ステートメントを挿入し，適切なクッキーを調べます。次に仮想ユーザ・スクリプトに **web_add_cookie** ステートメントを挿入してクッキーを設定します。

質問 15：仮想ユーザは実行時にクッキーを操作できますか？

回答：はい。仮想ユーザの実行中に，仮想ユーザはそのクッキー・キャッシュに格納されているクッキーを操作できます。仮想ユーザ・スクリプトの中で以下の関数を使うことにより，クッキー・キャッシュを操作できます。

- web_add_cookie()
- web_remove_cookie()
- web_cleanup_cookies()

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

実行時ビューア（オンライン・ブラウザ）

質問 16：実行時ビューアは Web ページをどのように表示するのでしょうか？

回答：Web 仮想ユーザ・スクリプトを実行すると，仮想ユーザがアクセスした Web サーバの情報は仮想ユーザにダウンロードされます。この情報は通常は HTML 形式です。仮想ユーザはこの情報を仮想ユーザの結果ディレクトリに保存します。各 Web ページは，独立した **.htm** ファイルとして HTML 形式で保存されます。仮想ユーザの実行中に，実行時ビューアが仮想ユーザの結果ディレクトリに保存されている **.htm** ファイルをロードし，それを Web ページとして表示します。

質問 17：実行時ビューアを使っていると，JavaScript エラーが頻繁に出ます。何が原因で，どのようにすれば防げるでしょうか？

回答：実行時ビューアを使うときには，実行時ビューアの［**オプション**］メニューの［**スクリプティングを有効化する**］オプションがチェックされていないことを確認します。これにより，実行時ビューアは，JavaScript を一切実行しなくなるので，実行時ビューアに JavaScript エラーは出なくなります。

質問 16 の回答で述べたように，仮想ユーザ・スクリプトを実行すると，VuGen はサーバによって返された情報を保存します。実行時ビューアが表示するのは，この保存された情報であって，サーバから直接返された情報ではありません。

質問 18：実行時ビューアが表示できるデータには，どのような種類がありますか？

回答：実行時ビューアが表示できるのは HTML ページだけです。他の形式の情報は表示できません。

質問 19：実行時ビューアを表示できるようにするには，ロード・ジェネレータに何をインストールすればよいでしょう？

回答：実行時ビューアは Internet Explorer の ActiveX コントロールを使用するので，Microsoft Internet Explorer 4.0 またはそれ以上が必ずマシンにインストールされていることを確認してください。

質問 20：仮想ユーザ・スクリプトを実行するときに，仮想ユーザが Web サーバに対して送信したデータを実行時ビューアが表示しないのはなぜですか？

回答：実行時ビューアは，サーバによって仮想ユーザに返された HTML ページだけを表示します。実行時ビューアは仮想ユーザが Web サーバに送信したデータは一切表示しません。詳細については，質問 16 の回答を参照してください。

質問 21：実行時ビューアはマルチウィンドウ・アプリケーションを正しく表示しますか？

回答：いいえ。現在のところ，実行時ビューアはマルチウィンドウ・アプリケーションを正しく表示しません。

ブラウザ

質問 22：スクリプトを記録するのには，いつも Netscape を使っているのですが，それでも Internet Explorer 4.0 以上のインストールが推奨されているのはなぜですか？

回答：VuGen は WinInet，つまり Microsoft Internet API に強く依存しています。これは Web 仮想ユーザ・スクリプトの記録と再生の両方に用いられています。WinInet.dll はインターネット接続を行うための Microsoft の基盤となる要素なのです。

LoadRunner では，WinInet.dll の version 3.0 がコンピュータにインストールされます（上位のバージョンがインストールされている場合を除きます）。version 3.0 には多くの制限があります。version 4.0 のほうがはるかに優れているので，Web 仮想ユーザを使って最良の結果を得るためには，version 4.0 をインストールすることをお勧めします。WinInet.dll の version 4.0 を最も簡単にインストールする正当な手段は，Internet Explorer 4.0 以上をインストールすることです。

質問 23：Internet Explorer 3.0 はインストールしていますが，Internet Explorer 4.0 以上はインストールしていません。これが原因で使えない機能にはどのようなものがあるでしょうか？

回答：Internet Explorer には WinInet.dll が含まれています。以下の機能を使うためには，version 4.0 の WinInet.dll ファイルが必要です。

- プロキシ認証
- NTLM 認証（NT challenge response authentication）

- デジタル証明書
- 実行時ブラウザ
- レポート
- WAP 記録

質問 24：記録には，標準ブラウザである Netscape または Internet Explorer を使わなければなりませんか？

回答：いいえ。Web 仮想ユーザ・スクリプトを記録するときには，任意のブラウザを選ぶことができます。実際のところ，ブラウザなしでも構いません。HTTP(S) 要求を生成するアプリケーションであればどのようなものでも利用できます。使用するアプリケーションの唯一の要件は，プロキシの設定を localhost:7777 とし，VuGen が HTTP(S) 要求を記録できるようにすることです。

質問 25：非標準 HTTP (S) アプリケーションを記録するにはどうすればよいでしょうか？

回答：次の手順で行います。

1 ［**ツール**］＞［**記録オプション**］を選択し，［**ブラウザ**］ノードをクリックします。［**ブラウザを手動で起動する**］を選択します。

2 ［**記録用プロキシ**］ノードをクリックして，［**ユーザ定義のプロキシを使用する**］オプションを選択します。［**OK**］をクリックして［記録オプション］を閉じます。

3 ［**アプリケーションの記録開始**］ボタンをクリックします。VuGen は記録されるアプリケーションに必要なプロキシ設定を要求するプロンプトを出します。ホスト名とポート名をメモします。

4 ［**キャンセル**］をクリックし，CTRL+F7 キーで［記録オプション］を開きます。［**記録用プロキシ**］ノードをクリックして，［**ユーザ定義のプロキシを使用する**］セクションに推奨されるプロキシ設定を入力します。［**OK**］をクリックして［記録オプション］を閉じます。

5 記録対象アプリケーションに合わせて必要な変更を行います。

6 ［**アプリケーションの記録開始**］ボタンをクリックします。

7 記録を終えたらアプリケーションを閉じて，プロキシの設定を元に戻します（戻し忘れると，アプリケーションが使えなくなることがあります）。

質問 26：VuGen が記録用ブラウザのプロキシ設定を変更することはありますか？

回答：あります。Web 仮想ユーザ・スクリプトの記録を開始すると，VuGen はユーザが指定したブラウザを起動します。次に VuGen は，そのブラウザが VuGen プロキシ・サーバを使うように設定します。このため，VuGen は記録用ブラウザのプロキシ設定を変更することになります。標準設定では，VuGen はプロキシ設定を直ちに localhost:7777 に変えます。記録終了後，VuGen は記録用ブラウザのプロキシの設定を元に戻します。VuGen が記録を行っている間はプロキシの設定を変更しないでください。

質問 27：記録をしているときにブラウザがクラッシュしました。その後，記録中でなくても，どのサイトにもアクセスできなくなってしまいました。なぜでしょうか？

回答：質問 26 の回答で VuGen が記録中にブラウザのプロキシ設定をどのように変更するかを説明しました。記録中にブラウザがクラッシュすると，VuGen はブラウザの元のプロキシの設定を復元できない場合があります。その場合，ブラウザのプロキシが localhost:7777 に設定されたままになり，どのサイトにもアクセスできなくなります。ブラウザのプロキシ設定を手作業で復元しなくてはなりません。

質問 28：VuGen は Socks プロキシをサポートしていますか？

回答：はい。VuGen は Socks プロキシをサポートしています。Socks プロキシを使う場合，記録用ブラウザは Netscape ではなく，Internet Explorer を使わなければなりません。さらに，以下を参照してください。

- Socks プロキシを定義するには，Internet Explorer 4.0 以上を使います。

  Internet Explorer で，［**ツール**］＞［**インターネット オプション**］を選択します。［**接続**］タブを選んで，［**LAN の設定**］ボタンをクリックします。［**プロキシ サーバ**］グループの［**詳細**］をクリックします。［プロキシの設定］ダイアログ・ボックスで，適切な Socks プロキシ・サーバ設定を入力します。

  この手順は，仮想ユーザ・スクリプトを記録するために使うコンピュータと，Socks プロキシ・サーバにアクセスする仮想ユーザを実行するすべてのコンピュータに適用されます。

- Internet Explorer を通常使用するブラウザとして指定します。

  これは，拡張子 **.htm** を持つすべてのファイルを Internet Explorer に関連付けることにより指定できます。

この手順は，仮想ユーザ・スクリプトを記録するために使うコンピュータと，Socks プロキシ・サーバにアクセスする仮想ユーザを実行するすべてのコンピュータに適用されます。

➤ 仮想ユーザ・スクリプトを実行するときに，記録用ブラウザからプロキシ設定を取り出すように指定します。

VuGen で，［**ツール**］＞［**記録オプション**］を選択します。［**記録プロキシ**］ノードをクリックします。［**記録用ブラウザからプロキシ設定を取得する**］オプションを選択します。

この手順は，仮想ユーザ・スクリプトを記録するコンピュータにのみ適用され，仮想ユーザを実行するコンピュータには適用されません。

➤ スクリプトを実行するすべての仮想ユーザが標準設定のブラウザからプロキシ設定を取り出すように指定します。

VuGen で，［**仮想ユーザ**］＞［**実行環境の設定**］を選択し，［**プロキシ**］タブをクリックして，［**標準設定のブラウザからプロキシ設定を取得する**］オプションを選択します。

この設定は仮想ユーザ・スクリプトを実行するすべての仮想ユーザに適用されます。

質問 29：Netscape はインストールしていて，Internet Explorer はインストールしていない場合，実行レポートを表示できますか？

回答：VuGen で実行レポートを表示するには，Internet Explorer 4.0 以上が必要です。

質問 30：［実行環境設定］で［**並行接続回数**］が使用できなくなったようですが，今でもこの設定を変更することはできますか？

回答：できます。**web_set_sockets_options** 関数を使って変更できます。ホストごとの最大接続数を設定するには，MAX_CONNECTIONS_PER_HOST フラグに必要な値を割り当てます。グローバルな接続数，つまり仮想ユーザごとの最大同時接続数を定義するには，MAX_TOTAL_CONNECTIONS フラグで必要な数値を指定します。Internet Explorer を使用している場合，並行接続数の標準設定値は，HTTP 1.0 では「4」，HTTP 1.1 では「2」です。詳細は，関数リファレンスで **web_set_sockets_options** を参照してください。

設定

質問 31：スナップショットの比較を行いましたが，非常に不正確な結果しか得られませんでした。

回答：［**ツール**］＞［**一般オプション**］を選択して［一般オプション］ダイアログ・ボックスを開き，［相関］タブを選択します。［**スナップショット間の差異を検索**］セクションで，［テキストを比較する］ではなく，［**HTML を比較する**］オプションが選択されていることを確認します。テキスト比較モードは，非 HTML スナップショットだけに適用できます。

互換性について

質問 32：記録したスクリプトは，UNIX システム上で再生できますか？

回答：はい。再生は UNIX プラットフォームでサポートされています。

第 9 部

Enterprise Java Bean プロトコル

第 46 章

EJB テストの実行

EJB（Enterprise Java Beans）テスト・ツールは，EJB オブジェクトをテストするためのスクリプトを生成します。

本章では，次の項目について説明します。

- EJB Detector での作業
- EJB 仮想ユーザ・スクリプトについて
- EJB テストの仮想ユーザの作成
- EJB 記録オプションの設定
- EJB 仮想ユーザ・スクリプトの実行

EJB テストの詳細については，付録 B「EJB アーキテクチャとテスト」を参照してください。

以降の情報は，EJB テスト用仮想ユーザ・スクリプトを対象とします。

EJB テストについて

VuGen には，Java アプリケーションをテストするためのスクリプトを作成する，いくつかのツールがあります。仮想ユーザ・スクリプトを記録して生成するには，Jacada，CORBA または RMI 仮想ユーザを使用します。スクリプトをプログラミングによって作成する場合には，標準 Java 仮想ユーザを使用します。

EJB テスト用仮想ユーザと標準 Java 仮想ユーザとの違いは，記録やプログラミングを行わずに，VuGen が自動的に EJB の機能をテストするスクリプトを作成する点です。スクリプトを生成する前に，アプリケーション・サーバに関する JNDI のプロパティなどの情報を指定します。LoadRunner の EJB Detector は，アプリケーション・サーバ内を検索し，どの EJB が使用可能かを判断します。テスト対象の EJB を選択すると，LoadRunner は，EJB の各メソッドをテストするスクリプトを生成します。メソッドごとにトランザクションを作成して，各メ

ソッドのパフォーマンスの測定と問題の特定ができるようにします。さらに，各メソッドは例外処理のために **try / catch** ブロックに入れられます。

EJB テスト・スクリプトを作成するには，LoadRunner EJB Detector がアプリケーション・サーバのホスト上にインストールされてアクティブになっていなければなりません。Detector については，この後の項で説明します。

VuGen には，スクリプトにメソッドを挿入するユーティリティも組み込まれています。このユーティリティを使用して，すべての利用可能なパッケージの表示，使用するメソッドの選択，およびそれらのスクリプトへの挿入ができます。詳細については，630 ページ「EJB 仮想ユーザ・スクリプトの実行」を参照してください。

EJB Detector での作業

EJB Detector は独立したエージェントで，EJB を検索するマシンごとにインストールする必要があります。このエージェントは，マシン上の EJB を検出します。EJB Detector をインストールする前に，マシンに有効な JDK 環境が設定されていることを確認します。

EJB Detector のインストール

EJB Detector は，アプリケーション・サーバ・マシンまたはクライアント・マシンにインストールして実行できます。クライアント・マシンで EJB Detector を実行するには，アプリケーション・サーバ・マシンにマウントされたドライブが必要です。

EJB Detector エージェントのインストールは，次の手順で行います。

1 アプリケーション・サーバ・マシンまたはクライアント・マシンに EJB Detector のホーム・ディレクトリを作成します（クライアント・マシンに作成する場合は，ファイル・システムを前述のようにマウントします）。

2 EJB Detector フォルダで圧縮ファイル< **LR_root** > **¥ejbcomponent¥ejbdetector.jar** を解凍します。

EJB Detector の実行

VuGen で EJB スクリプトの生成を開始する前に，EJB Detector を実行しておく必要があります。EJB Detector はアプリケーション・サーバまたはクライアント・マシンで実行できます（クライアント・マシンの場合は，EJB Detector のクライアント・マシンからアプリケーション・サーバへのマウントを行い，検索ルート・ディレクトリ内にマウント・ディレクトリを指定し，生成されたスクリプトをローカル・マシンではなくマウントされたマシンに接続するように変更します）。

EJB Detector は，コマンド・ラインまたはバッチ・ファイルから実行できます。

EJB Detector のコマンド・ラインからの実行は，次の手順で行います。

1 コマンド・ラインで EJB Detector を実行する前に，CLASSPATH 環境変数に DETECTOR_HOME¥classes と DETECTOR_HOME¥classes¥xerces.jar を 追加します。

2 EJB1.0（Weblogic 4.x，WebSphere 3.x）で作業する場合は，テスト対象の EJB クラスを，次のベンダー EJB クラスとともに CLASSPATH に追加します。

WebLogic 4.x の場合：＜ WebLogic のディレクトリ＞ ¥lib¥weblogicaux.jar

WebSphere 3.x の場合：＜ WebSphere のディレクトリ＞ ¥lib¥ujc.jar

3 対象の EJB で使用しているクラス・ディレクトリまたは .jar ファイルがほかにもある場合，それらも CLASSPATH に追加します。

4 コマンド・ラインから EJB Detector を実行するには，次の文字列を使用します。

java EJBDetector [検索ルート・ディレクトリ] [リッスン・ポート]

search root dir	1 つ以上のディレクトリまたはファイルで（セミコロンで区切られた）複数の EJB を検索するには，次のガイドラインにしたがいます。**BEA WebLogic Servers 4.x および 5.x**：アプリケーション・サーバのルート・ディレクトリを指定します。**BEA WebLogic Servers 6.x**：ドメイン・フォルダの完全パスを指定します。**WebSphere Servers 3.x**：配備された EJB フォルダの完全パスを指定します。**WebSphere Servers 4.0**：アプリケーション・サーバのルート・ディレクトリを指定します。**Oracle OC4J**：アプリケーション・サーバのルート・ディレクトリを指定します。**Sun J2EE Server**：**.ear** ファイルまたは複数の **.ear** ファイルがあるディレクトリへの完全パスを指定します。 指定しない場合は，クラスパスが検索されます。
リッスン・ポート	EJB Detector のリッスン・ポートです。標準設定のポートは 2001 です。ポート番号を変更したら，［**EJB スクリプトの生成**］ダイアログ・ボックスの［**ホスト**］の［**名前**］ボックスでも指定しなおす必要があります。 たとえば，ホストが「metal」で，標準設定のポートを使用している場合，「**metal**」と指定します。別のポート，たとえばポート 2002 を使用している場合は，「**metal:2002**」と指定します。

EJB Detector のバッチ・ファイルからの実行は，次の手順で行います。

バッチ・ファイル **EJB_Detector.cmd** を使用して，EJB Detector を起動できます。このファイルは，圧縮ファイル **ejbdetector.jar** を解凍すると，インストールされている EJB Detector のルート・ディレクトリに置かれます。

1 EJB Detector の ルート・ディレクトリで env.cmd を開き，環境に応じて次の変数を変更します。

JAVA_HOME インストールされている JDK のルート・ディレクトリ。

DETECTOR_INS_DIR インストールされた Detector のルート・ディレクトリ。

APP_SERVER_DRIVE アプリケーション・サーバのインストールをホストするドライブ。

APP_SERVER_ROOT 次のガイドラインにしたがいます。**BEA WebLogic Servers 4.x および 5.x**：アプリケーション・サーバのルート・ディレクトリを指定します。**BEA WebLogic Servers 6.x**：ドメイン・フォルダの完全パスを指定します。**WebSphere Servers 3.x**：配備された EJB フォルダの完全パスを指定します。**WebSphere Servers 4.0**：アプリケーション・サーバのルート・ディレクトリを指定します。**Oracle OC4J**：アプリケーション・サーバのルート・ディレクトリを指定します。**Sun J2EE Server**：**.ear** ファイルまたは複数の **.ear** ファイルがあるディレクトリへの完全パスを指定します。

EJB_DIR_LIST（任意） 配備可能な ear/.jar ファイル，および任意の追加クラス・ディレクトリまたは .jar ファイルを含んでいるか，テスト対象の EJB で使用されている，セミコロン（;）で区切られたディレクトリとファイルのリスト。

2 **env.cmd** を保存します。

3 EJB1.0（Weblogic 4.x，WebSphere 3.x）で作業する場合は，テスト対象の EJB のクラスとともに，次のベンダー EJB クラスを env ファイル の CLASSPATH に追加します。

WebLogic 4.x の場合：＜ WebLogic のディレクトリ＞ ¥lib¥weblogicaux.jar

WebSphere 3.x の場合：＜ WebSphere のディレクトリ＞ ¥lib¥ujc.jar

4 バッチ・ファイル EJB_Detector.cmd または EJB_Detector.sh（Unix プラットフォームの場合）を実行して，EJB を含む導入可能なアプリケーションに関する情報を収集します。たとえば，次のように指定します。

```
C:¥>EJB_Detector [listen_port]
```

「listen_port」は，EJB Detector が受信する要求を読み取るポート番号を指定するための任意の引数です（標準設定では「2001」）。

EJB Detector の出力およびログ・ファイル

EJB Detector の出力を調べて，アクティブな EJB をすべて検出したかどうかを確認できます。出力ログには，EJB で検査されたパスが表示されます。検索が終わると，見つかった EJB の名前と場所のリストが表示されます。

たとえば，次のように表示されます。

```
Checking EJB Entry: f:/weblogic/myserver/ejb_basic_beanManaged.jar...
Checking EJB Entry: f:/weblogic/myserver/ejb_basic_statefulSession.jar...
Checking EJB Entry: f:/weblogic/myserver/ejb_basic_statelessSession.jar...
----------------------- Found 3 EJBs -------------------------
** PATH: f:/weblogic/myserver/ejb_basic_beanManaged.jar
      - BEAN: examples.ejb.basic.beanManaged.AccountBean
** PATH: f:/weblogic/myserver/ejb_basic_statefulSession.jar
      - BEAN: examples.ejb.basic.statefulSession.TraderBean
** PATH: f:/weblogic/myserver/ejb_basic_statelessSession.jar
      - BEAN: examples.ejb.basic.statelessSession.TraderBean
```

EJB が検出されなかった場合（「Found 0 EJBs」と表示されます），EJB jar ファイルが「Checking EJB Entry:...」行に表示されているかどうかを確認してください。リストに表示されていない場合は，「**検索ルート・ディレクトリ**」のパスが正しいかどうか確認します。検査が行われているのにもかかわらず EJB が検出されない場合は，EJB jar ファイルが配備可能か（アプリケーション・サーバに配備できるか）確認します。配備可能な jar ファイルには，Home Interface, Remote Interface, Bean の実装，Deployment Descriptor ファイル（xml ファイルまたは .ser ファイル）およびその他のベンダー固有のファイルが含まれています。

それでも問題が生じる場合は，DETECTOR_HOME¥classes ディレクトリにある「**detector.properties**」ファイルにデバッグ・プロパティを設定し，さらに詳しいデバッグ情報を取得します。

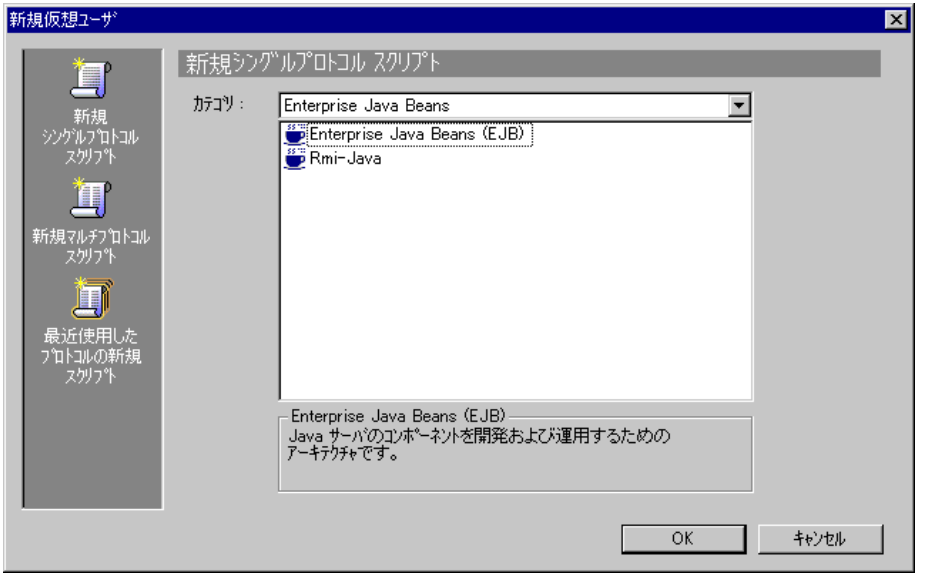
EJB が検出されると，HTTP サーバは初期化されて，VuGen の EJB テスト仮想ユーザからの要求を待機します。この通信過程に問題がある場合は，DETECTOR_HOME¥classes ディレクトリにある **webserver.properties** ファイルで webserver.enableLog プロパティを有効にします。

これにより，**webserver.log** ファイルに詳しいデバッグ情報や，その他の潜在的に重要なエラー・メッセージが出力されるようになります。

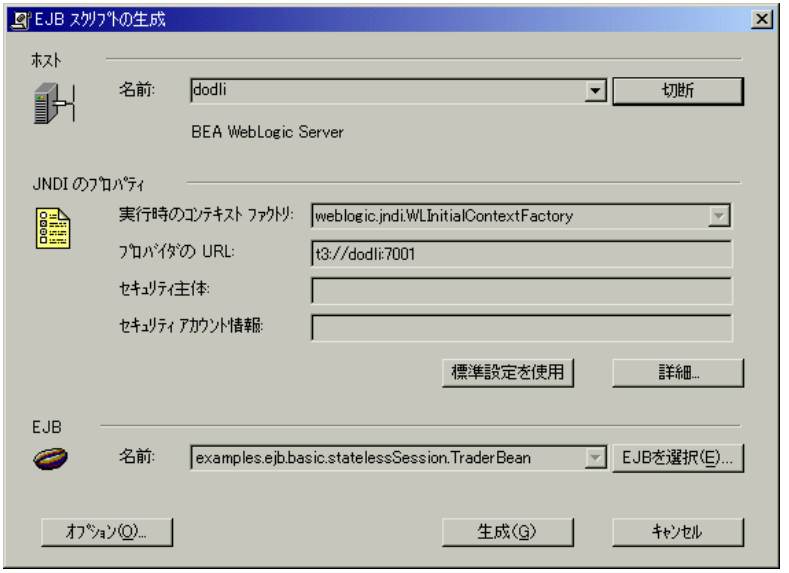
EJB テストの仮想ユーザの作成

EJB 仮想ユーザ・スクリプトの作成は，次の手順で行います。

1 ［**ファイル**］＞［**新規作成**］を選択するか，［**新規作成**］ボタンをクリックします。［新規仮想ユーザ］ダイアログ・ボックスが表示されます。



2［**Enterprise Java Beans**］カテゴリから［**Enterprise Java Beans（EJB）**］を選択し，［**OK**］をクリックします。VuGen が空の Java 仮想ユーザ・スクリプトを開き，［EJB スクリプトの生成］ダイアログ・ボックスが表示されます。



3 LoadRunner EJB Detector がインストールされているマシンを指定します。接続するためには Detector が稼動していなければなりません。［**接続**］をクリックします。［**JNDI のプロパティ**］セクションが有効になります。

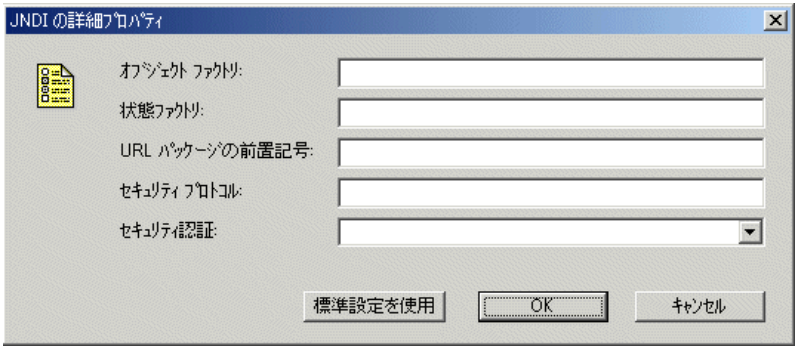
4 EJB Detector は，標準設定の JNDI プロパティを自動的に検出します。これらのプロパティは，編集ボックスで手作業で変更することもできます。変更可能なプロパティは，［**実行時のコンテキストファクトリ**］および［**プロバイダの URL**］の文字列です。

アプリケーション・サーバが認証を必要とする場合は，［**セキュリティ主体**］ボックスにユーザ名，［**セキュリティ アカウント情報**］にパスワードを入力します。

次に JNDI の必須プロパティの 2 つの標準設定値を示します。

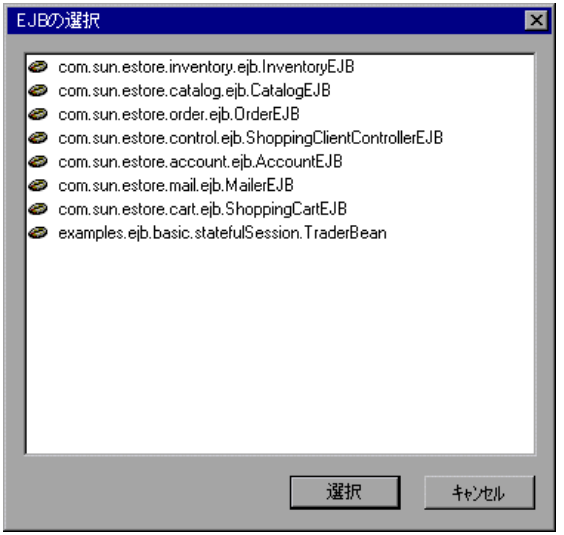
タイプ	実行時のコンテキスト・ファクトリ	プロバイダの URL
WebLogic	weblogic.jndi.WLInitialContextFactory	t3:// ＜ appserver_host ＞ :7001
WebSphere 3.x	com.ibm.ejs.ns.jndi.CNInitialContextFactory	iiop:// ＜ appserver_host ＞ :900
WebSphere 4.x	com.ibm.websphere.naming.WsnInitialContextFactory	iiop:// ＜ appserver_host ＞ :900
Sun J2EE	com.sun.enterprise.naming.SerialInitContextFactory	なし
Oracle	com.evermind.server.AppplicationClientInitialContextFactory	ormi:// ＜ appserver_host ＞ / ＜ application_name ＞ （＜ oc4j ＞ /config/server.xml 形式の EJB アプリケーション名）

5 JNDI の詳細プロパティを設定するには，［**詳細**］をクリックして［JNDI の詳細プロパティ］ダイアログ・ボックスを開きます。



必要に応じて，［**オブジェクトファクトリ**］，［**状態ファクトリ**］，［**URL パッケージの前置記号**］，［**セキュリティプロトコル**］，［**セキュリティ認証**］プロパティを指定します。［**OK**］をクリックします。

6 ダイアログ・ボックス内の［**EJB**］セクションで［**EJB を選択**］をクリックし，テストを作成する EJB を選択します。ダイアログ・ボックスが開き，現在アプリケーション・サーバからアクセス可能なすべての EJB のリストが表示されます。



7 テスト対象の EJB を強調表示し，［**選択**］をクリックします。

8 ［EJB スクリプトの生成］ダイアログ・ボックスで，［**生成**］をクリックします。VuGen は，LoadRunner Java 仮想ユーザ関数を含むスクリプトを作成します。スクリプトには，アプリケーション・サーバに接続し，EJB のメソッドを実行するためのコードが含まれます。

9 スクリプトを保存します。

既存のスクリプト内には追加 EJB のテストコードを生成できません。別の EJB を作成するには，新規のスクリプトを開き，前述のステップ 2 ～ 9 を繰り返します。

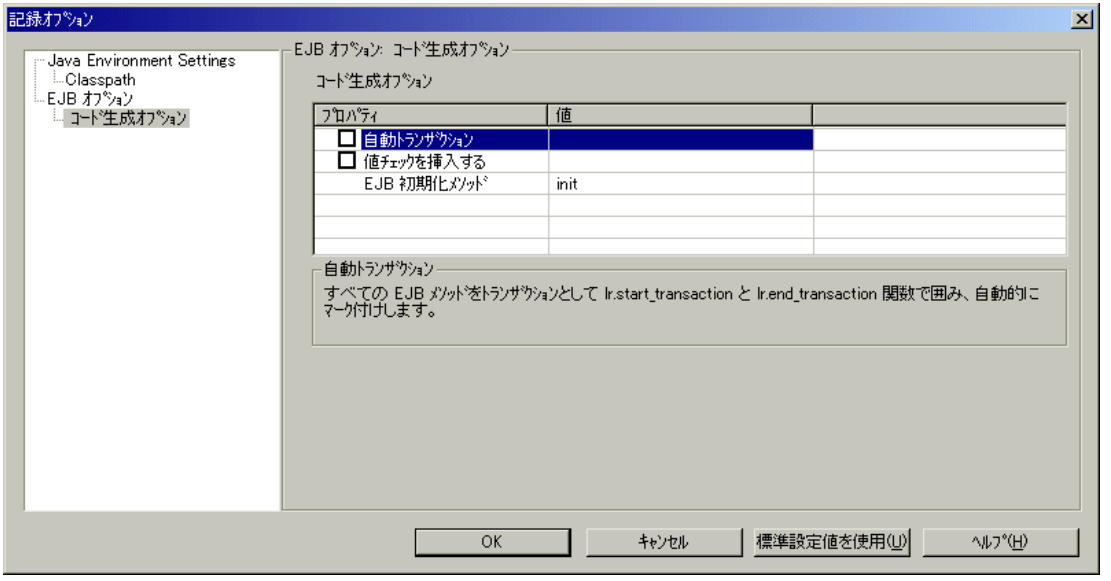
EJB 記録オプションの設定

EJB 仮想ユーザに使用可能な記録オプションは［Classpath］と［コード生成］に含まれています。記録モードの詳細については，第 13 章 「Java 言語仮想ユーザ・スクリプトの記録」を参照してください。

［**EJB オプション：コード生成**］オプションを使用して，自動トランザクションと値チェックのセクションでプロパティを設定できます。また，初期化メソッドの保管場所も指定できます。

EJB コード生成記録オプションの設定は，次の手順で行います。

1 ［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。コード生成オプションを編集するには［記録オプション］の［**EJB オプション：コード生成オプション**］ノードを選択します。



2 ［**自動トランザクション**］オプションを有効にし，すべての EJB メソッドをトランザクションとして扱うように自動的にマークします。これで，すべてのメソッドを **lr.start_transaction** と **lr.end_transaction** 関数で囲むことになります。標準設定では，このオプションは有効になっています（つまり，値が「true」に設定されています）。

3［**値チェックを挿入する**］オプションを有効にし，**lr.value_check** 関数を各 EJB メソッドの後に自動挿入します。この関数は，プリミティブな値および文字列で返される期待値をチェックします。

4［**EJB 初期化メソッド**］を選択します。このメソッドには，EJB/JNDI 初期化プロパティが書き込まれます。利用可能なメソッドは，**init**（標準設定）と **action** です。

EJB 仮想ユーザ・スクリプトについて

VuGen は，仮想ユーザ・スクリプトの作成時に指定された JNDI（Java Naming and Directory Interface）のプロパティに基づいて，EJB をテストするスクリプトを生成します。JNDI は，Java プログラムを DNS および LDAP などのネーム・サービスやディレクトリ・サービスに接続するときに使用される Sun のプログラミング・インタフェースです。

各 EJB 仮想ユーザ・スクリプトは，次の 3 つの主要部分からなります。

- JNDI による EJB Home の検索
- インスタンスの作成
- EJB メソッドの起動

JNDI による EJB Home の検索

スクリプトの最初のセクションには，JNDI のプロパティを取得するためのコードが含まれています。このコードは指定されたコンテキスト・ファクトリおよびプロバイダの URL を使用して，アプリケーション・サーバに接続し，指定された EJB を検索し，EJB Home を見つけます。

次の例では，JNDI Context Factory は weblogic.jndi.WLInitialContextFactory，プロバイダの URL は t3://dod:7001，選択された EJB の JNDI 名は carmel.CarmelHome です。

```
public class Actions
{

      public int init() {
            CarmelHome _carmelhome = null;
            try {
                  // JNDI Initial Context を取得する
                  java.util.Properties p = new java.util.Properties();
                  p.put(javax.naming.Context.INITIAL_CONTEXT_FACTORY,
"weblogic.jndi.WLInitialContextFactory");
                  p.put(javax.naming.Context.PROVIDER_URL,
"t3://dod:7001");
                  javax.naming.InitialContext _context =
new javax.naming.InitialContext(p);

                  // JNDI コンテキストで Home Interface を検索し，絞り込む
                  Object homeobj = _context.lookup("carmel.CarmelHome");
                  _carmelhome = (CarmelHome)javax.rmi.PortableRemoteObject.
narrow(homeobj, CarmelHome.class);

            } catch (javax.naming.NamingException e) {
                  e.printStackTrace();
            }
```

注：スクリプトがアプリケーション・サーバではなくクライアントで実行されている EJB Detector で生成されている場合は，プロバイダの URL を手作業で変更する必要があります。たとえば，次の行では，プロバイダは EJB Detector のホスト名として dod を指定しています。
p.put(javax.naming.Context.PROVIDER_URL, "t3://dod:7001") 記録されたホスト名をアプリケーション・サーバ名で置き換えます。次に例を示します。
p.put(javax.naming.Context.PROVIDER_URL, "t3://bealogic:7001")［EJB スクリプトの生成］ダイアログ・ボックスの［**JDNI のプロパティ**］セクションで，記録を開始する前にプロバイダの URL を指定しておくと，手作業で変更する必要がなくなります。

インスタンスの作成

EJB メソッドを実行する前に，スクリプトは，EJB の Bean インスタンスを作成します。インスタンスの作成は，トランザクションとしてマークされるので，スクリプトの実行後に分析できます。さらに，インスタンスの作成プロセスは，例外処理のために **try / catch** ブロックに入れられます。

セッション Beans では，EJB Home「作成」メソッドを使用して新しい EJB インスタンスを作成します。

次の例では，スクリプトは，Carmel EJB のインスタンスを作成します。

```
// Bean インスタンスの作成
Carmel _carmel = null;
try {
        lr.start_transaction("create");
        _carmel = _carmelhome.create();
        lr.end_transaction("create", lr.AUTO);
} catch (Throwable t) {
        lr.end_transaction("create", lr.FAIL);
        t.printStackTrace();
}
```

エンティティ Beans では，findByPrimaryKey メソッドを使用して既存のデータベースで EJB インスタンスを検索します。見つからなかった場合は，create メソッドを使用してその中に作成します。

次の例では，スクリプトは，アカウント EJB のインスタンスを検索し，見つからない場合には作成します。

```
// Bean インスタンスの検索
try {
      com.ibm.ejs.doc.account.AccountKey _accountkey = new
com.ibm.ejs.doc.account.AccountKey();
      _accountkey.accountId = (long)0;

      lr.start_transaction("findByPrimaryKey");
      _account = _accounthome.findByPrimaryKey(_accountkey);
      lr.end_transaction("findByPrimaryKey", lr.AUTO);
} catch (Throwable thr) {

      lr.end_transaction("findByPrimaryKey", lr.FAIL);
      lr.message("Couldn't locate the EJB object using a primary key.
Attempting to manually create the object... ["+thr+"]");

      // Bean インスタンスの作成
      try {
            lr.start_transaction("create");
            _account =
_accounthome.create((com.ibm.ejs.doc.account.AccountKey)null);
            lr.end_transaction("create", lr.AUTO);
      } catch (Throwable t) {
            lr.end_transaction("create", lr.FAIL);
            t.printStackTrace();
      }

}
```

エンティティ Bean によって提供されるその他の find... メソッドを使用して EJB インスタンスを検出することもできます。たとえば，次のような場合です。

```
// データベース内の「John」を示す,
// すべての Email EJB インスタンスのリストの取得
Enumeration enum = home.findByName( "John" );
      while ( enum.hasMoreElements() ) {
         Email addr = (Email)enum.nextElement();
            ...
}
```

EJB メソッドの起動

スクリプトの最後の部分には，実際の EJB メソッドが含まれています。各メソッドはトランザクションとしてマークされるので，スクリプト実行後にメソッドを分析できます。さらに，各メソッドは例外処理のために **try / catch** ブロックに入れられます。例外があった場合には，トランザクションは「**failed**」とマークされ，スクリプトは次のメソッドから続行されます。VuGen は EJB メソッドごとに個別のブロックを作成します。

```
// ------- メソッド ------------

        int _int1 = 0;
        try {
                lr.start_transaction("buyTomatoes");
                _int1 = _carmel.buyTomatoes((int)0);
                //lr.value_check(_int1, 0, lr.EQUALS);
                lr.end_transaction("buyTomatoes", lr.AUTO);
        } catch (Throwable t) {
                lr.end_transaction("buyTomatoes", lr.FAIL);
                t.printStackTrace();
        }
```

VuGen はそれらのメソッドの標準値を挿入します。たとえば，整数の場合は 0，文字列の場合は空の文字列（""），複雑な Java オブジェクトの場合は NULL となります。必要に応じて，生成されたスクリプト内の標準値を変更します。

```
                _int1 = _carmel.buyTomatoes((int)0);
```

次の例では，パラメータ化を使用した，複雑なタイプの標準値の変更方法を示しています。

```
Detail details = new Details( < city > , < street > , < zip > , < phone >
);
JobProfile job = new  JobProfile( < department > , < position > , <
job_type > );
Employee employee=new Employee( < first > , < last > , details, job, <
salary > );
_int1 = _empbook.addEmployee((Employee)employee);
```

簡単な値や文字列を返すメソッドの場合には，VuGen はコメントアウトされたメソッド **lr.value_check** を挿入します。この LoadRunner メソッドを使用して，

EJB メソッドの期待値を指定できます。この検証メソッドを使用するときには，コメントの記号（//）を削除し，期待値を指定します。たとえば，**carmel.buyTomatoes** メソッドは整数を返します。

```
_int1 = _carmel.buyTomatoes((int)0);
//lr.value_check(_int1, 0, lr.EQUALS);
```

メソッドが 500 という値を返すようにするには，コードを次のように変更します。

```
_int1 = _carmel.buyTomatoes((int)0);
lr.value_check(_int1, 500, lr.EQUALS);
```

メソッドが特定の値を返さなかったかどうかを検査する場合は，コードを次のように変更します。

```
_int1 = _carmel.buyTomatoes((int)0);
lr.value_check(_int1, 10, lr.NOT_EQUALS);
```

期待値が検出されない場合は，例外処理が行われ，その情報は出力ウィンドウのログに記録されます。

```
System.err:java.lang.Exception:lr.value_check failed.[Expected:500
Actual:5000]
```

EJB 仮想ユーザ・スクリプトは，標準 Java 規約をすべてサポートしています。たとえば，テキストの前に 2 つのスラッシュ「//」を付けることによって，コメントを挿入できます。

Java 仮想ユーザ・スクリプトは，スケーラブルなマルチ・スレッド・アプリケーションとして稼動します。スクリプトにユーザ定義のクラスを含める場合は，コードをスレッドセーフにする必要があります。スレッドセーフでないコードは，結果が不正確になることがあります。スレッドセーフでないコードの場合は，Java 仮想ユーザをプロセスとして実行します。つまり，プロセスごとに個別の Java 仮想マシンを作成します。その結果，スクリプトの拡張性は低くなります。

EJB 仮想ユーザ・スクリプトの実行

EJB テストのスクリプトを生成した後，必要な変更を行えば，スクリプト実行の準備が完了します。EJB スクリプトでは，機能テストと負荷テストの 2 種類のテストを実行できます。機能テストでは，EJB が実際の環境下で正しく機能することを検証します。負荷テストでは，一度に多数のユーザを実行したときの EJB のパフォーマンスを評価します。

機能テストは，次の手順で行います。

1 自動的に生成された標準値を変更します。

2 **lr.value_check** メソッドを使用して値の検査を挿入します。これらのメソッドは，スクリプト内にコメントとして生成されます。詳細については，628 ページ「EJB メソッドの起動」を参照してください。

3 追加のメソッドを挿入し，標準値を変更します。詳細については，第 13 章「Java 言語仮想ユーザ・スクリプトの記録」の Java 関数の挿入に関する項を参照してください。

4 スクリプトの一般的な実行環境を設定します。詳細については，第 9 章 「実行環境の設定」を参照してください。

5 Java VM の実行環境を設定します（追加のクラスパス，VM パラメータをすべて指定します）。必ずアプリケーション・サーバ EJB クラスを入れます。テストされている実際の EJB クラスは仮想ユーザ・ディレクトリに保存され，再生中に自動的に取り出されます。追加のクラスパスの指定，JavaVM 実行環境の設定の詳細については，第 16 章 「Java 実行環境の設定」を参照してください。

6 **Websphere 3.x** ユーザの場合：

IBM JDK 1.2 以上を使用した場合：

- クラスパスに＜ WS ＞ ¥lib¥ujc.jar を追加します。

Sun JDK 1.2.x を使用する場合：

- ファイル＜ JDK ＞ ¥jre¥lib¥ext¥iiimp.jar を削除します。
- 次のファイルを，＜ WS ＞ ¥jdk¥jre¥lib¥ext フォルダから＜ JDK ＞ ¥jre¥lib¥ext フォルダにコピーします。
- ujc.jar を＜ WS ＞ ¥lib フォルダから＜ JDK ＞ ¥jre¥lib¥ext フォルダにコピーします。
- ファイル＜WS＞¥jdk¥jre¥bin¥ioser12.dll を＜JDK＞¥jre¥bin フォルダにコピーします。

ここで＜ **WS** ＞は WebSphere インストールの ホーム・ディレクトリ，＜ **JDK** ＞は JDK インストールのホーム・ディレクトリです。

このオプションを無効にするには，［**-Xbootclasspath VM パラメータを使用する**］チェック・ボックスをクリアします。

7 **WebSphere 4.0** ユーザの場合：

マシンに IBM JDK1.3 の Java 環境が正しく設定されていることを確認します。［実行環境設定］ダイアログ・ボックスを開いて，［**Java VM**］ノードを選択します。次のエントリを［**Additional Classpath**］に追加します。

```
＜ WS ＞ ¥lib¥webshpere.jar;
＜ WS ＞ ¥lib¥j2ee.jar;
```

＜ WS ＞は，WebSphere インストールのホーム・ディレクトリです。

このオプションを無効にするには，［**-Xbootclasspath VM パラメータを使用する**］チェック・ボックスをクリアします。

注：アプリケーション・サーバが UNIX マシンにインストールされている場合，または WebSphere 3.0.x を使用している場合は，必要なファイルを取得するために，クライアントに IBM JDK 1.2.x をインストールします。

---

8 **Oracle OC4J** ユーザの場合：

マシンに JDK1.2 またはそれ以上（JDK1.3 推奨）の Java 環境が正しく設定されていることを確認します。［実行環境設定］ダイアログ・ボックスを開いて，［**Java VM**］ノードを選択します。次のエントリを［**Additional Classpath**］に追加します。

```
＜ OC4J ＞ ¥orion.jar; ＜ OC4J ＞ ¥ejb.jar; ＜ OC4J ＞ ¥jndi.jar; ; ＜ OC4J ＞
¥xalan.jar;
＜ OC4J ＞ ¥crimson.jar
```

＜ OC4J ＞は，アプリケーション・サーバのインストールのホーム・ディレクトリです。

9 **Sun J2EE** ユーザの場合：

マシンに JDK1.2 またはそれ以上の Java 環境が正しく設定されていることを確認します。［実行環境設定］ダイアログ・ボックスを開いて，［**Java VM**］ノードを選択します。次のエントリを［**Additional Classpath**］に追加します。

```
< J2EE > ¥j2ee.jar; < AppClientJar >
```

< J2EE >にはアプリケーション・サーバをインストールするホーム・フォルダを指定します。< AppClientJar >は，配備工程の間に sdk ツールによって自動的に生成されるアプリケーション・クライアントの jar ファイルへのパスを指定します。

10 **WebLogic 4.x - 5.x** ユーザの場合：

マシンに Java 環境（パスおよびクラスパス）が正しく設定されていることを確認します。［実行環境設定］ダイアログ・ボックスを開いて，［**Java VM**］ノードを選択します。次の 2 つのエントリを［**Additional Classpath**］セクションに追加します。

```
< WL > ¥classes; < WL > ¥lib¥weblogicaux.jar
```

< WL >は，WebLogic インストールのホーム・ディレクトリです。

11 **WebLogic 6.x** および **7.0** ユーザの場合：

マシンに Java 環境（パスおよびクラスパス）が正しく設定されていることを確認します。WebLogic 6.1 を使用する場合は，JDK 1.3 をインストールする必要があります。［実行環境設定］ダイアログ・ボックスを開いて，［**Java VM**］ノードを選択します。次のエントリを［**Additional Classpath**］セクションに追加します。

```
< WL > ¥lib¥weblogic.jar;   // Weblogic 6.x
< WL > ¥server¥lib¥weblogic.jar    // Weblogic 7.x
```

< WL >は，WebLogic インストールのホーム・ディレクトリです。

このオプションを無効にするには，［**-Xbootclasspath VM パラメータを使用する**］チェック・ボックスをクリアします。

12 スクリプトを実行します。［**実行**］ボタンをクリックするか，［**仮想ユーザ**］＞［**実行**］を選択します。［実行ログ］ノードを開き，実行時のエラーを表示します。実行ログは，スクリプトのフォルダ内の **mdrv.log** ファイルに保存されます。Java コンパイラ（Sun の javac）はスクリプトにエラーがないか調べた後，コンパイルを実行します。

EJB が機能することを確認した後，それを負荷シナリオの中で多数の仮想ユーザに割り当てることによって，負荷テストを実行できます。詳細については，『**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 10 部

ERP/CRM プロトコル

第 47 章

Oracle NCA 仮想ユーザ・スクリプトの作成

VuGen を使って，Oracle NCA ユーザをエミュレートするスクリプトを作成できます。まず，VuGen で典型的な NCA のビジネス・プロセスを記録します。次に，システムと対話するユーザをエミュレートするスクリプトを実行します。

本章では，次の項目について説明します。

- Oracle NCA 仮想ユーザを使った作業の開始
- 記録作業のガイドライン
- 名前によるオブジェクトの記録の有効化
- Personal Home Page からの Oracle Applications の使用
- Oracle NCA 仮想ユーザ関数の使用
- Oracle NCA 仮想ユーザについて
- ツリー・ビューとスクリプト・ビューの切り替え
- 実行環境の設定
- Oracle NCA アプリケーションのテスト
- ロード・バランシングに向けた Oracle NCA ステートメントの相関
- プラグマ・モードでの記録

以降の情報は，Oracle NCA プロトコルを対象とします。

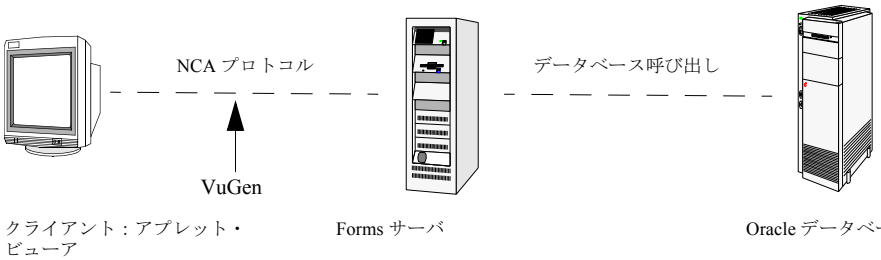
Oracle NCA 仮想ユーザ・スクリプトの作成について

Oracle NCA は，Java ベースのデータベース・プロトコルです。データベース・クライアントであるアプレット・ビューアは，ブラウザを使用して起動します。NCA データベースに対するアクションは，アプレット・ビューアを使用して実行します。アプレット・ビューアによりクライアント・ソフトウェアが不要となり，アプレット・ビューアをサポートするあらゆるプラットフォームでデータベース・アクションを実行できるようになります。Oracle NCA クライアントをエミュレートするために特別に設計された仮想ユーザの種類があります。

NCA の環境は 3 層構造の環境です。ユーザはまずブラウザから Web サーバへ http 呼び出しを送信します。この呼び出しは，Oracle Applications アプレットを起動するスタートアップ HTML ページにアクセスします。このアプレットはクライアント・マシンでローカルに実行します。以降の呼び出しはすべて，独自の NCA プロトコルを使ってクライアントと Forms サーバの間で通信されます。

クライアント（アプレット・ビューア）は，データベース・サーバ（Oracle 8.x）に情報を送信するアプリケーション・サーバ（Oracle Forms サーバ）と通信します。

VuGen は，クライアントと Forms サーバ（アプリケーション・サーバ）間の NCA 通信を記録し，再生します。



シングル・プロトコル・スクリプトを作成した場合であっても，Oracle NCA セッションが記録される際には，VuGen によってすべての NCA アクションおよび Web アクションが記録されます。テストにおいて Web 関数が必要であることが事前にわかっている場合は，最初から Oracle NCA プロトコルと Web プロトコルを対象としたマルチ・プロトコル・スクリプトを作成します。

最初に Oracle NCA プロトコルを対象としたシングル・プロトコル・スクリプトを作成し，後でテストのために Web 関数が必要となった場合は，セッションを再記録しなくても VuGen でスクリプトを再生成して Web 関数を追加できます。これは，［オプションの再生成］ダイアログ・ボックスの［プロトコル］ノードで指定します。詳細については，第 3 章 「VuGen を使った記録」を参照してください。

Oracle NCA 仮想ユーザを使った作業の開始

次の手順は，Oracle NCA 仮想ユーザ・スクリプトの作成方法の概要を示します。

1 **記録するマシンの設定が正しいことを確認します。**

VuGen を起動する前に，Oracle NCA アプレット・ビューアが動作するようにマシンが設定されていることを確認します。また，使用している Oracle Forms のバージョンが LoadRunner でサポートされていることを確認します。詳細については，640 ページ「記録作業のガイドライン」と Readme ファイルを参照してください。

2 **スケルトン Oracle NCA 仮想ユーザ・スクリプトを作成します。**

VuGen を使用して，スケルトン Oracle NCA 仮想ユーザ・スクリプトを作成します。詳細については，30 ページ「仮想ユーザ・スクリプトのセクション」を参照してください。

3 **典型的なユーザ・アクションを記録します。**

記録を開始し，アプレット・ビューアを使って典型的なアクションとビジネス・プロセスを実行します。VuGen によってアクションが記録され，仮想ユーザ・スクリプトが生成されます。詳細については，32 ページ「仮想ユーザ・スクリプトの新規作成」を参照してください。

4 **仮想ユーザ・スクリプトを拡張します。**

［挿入］メニューを使って，トランザクション，ランデブー・ポイント，コメント，メッセージなどを追加して，仮想ユーザ・スクリプトを拡張します。詳細については，第 6 章 「仮想ユーザ・スクリプトの拡張」を参照してください。

5 **スクリプトをパラメータ化します。**

記録された定数をパラメータと置き換えます。詳細については，第 7 章 「パラメータの定義」を参照してください。

6 **スクリプトの実行環境プロパティを設定します。**

仮想ユーザ・スクリプトの実行環境の設定を行います。実行環境設定は，スクリプト実行のいくつかの特性を規定します。詳細については，第 9 章 「実行環境の設定」を参照してください。

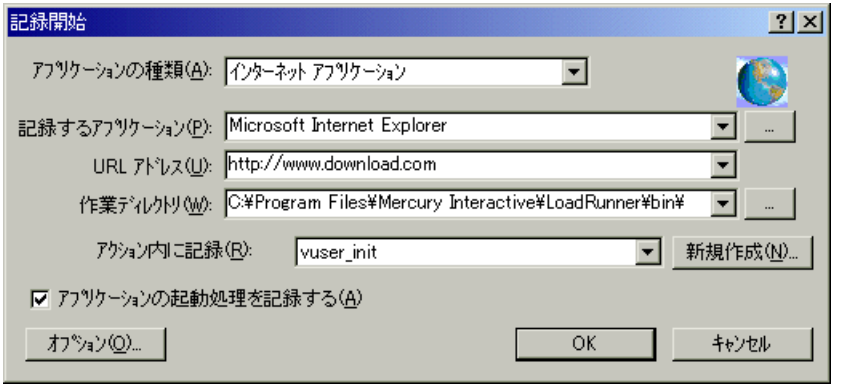
7 **仮想ユーザ・スクリプトを保存して実行します。**

VuGen からスクリプトを実行して，実行時の情報を実行ログで確認します。詳細については，第 11 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。

記録作業のガイドライン

Oracle NCA 仮想ユーザ・スクリプトを記録する際には，次のガイドラインに従います。

➤ Oracle NCA セッションを記録するときに VuGen が使用するブラウザを指定します。［記録開始］ダイアログ・ボックスの［**記録するアプリケーション**］リストで，使用するブラウザを選択します。このリストには，使用可能なブラウザがすべて含まれています。



➤ 記録を開始する前にすべてのブラウザを閉じます。

➤ **vuser_init** セクションにログイン・プロシージャを記録します。**Actions** セクションに典型的なビジネス・プロセスを記録します。スクリプトを実行するときに，特定のビジネス・プロセスを複数回反復するように指定できます。詳細については，32 ページ「仮想ユーザ・スクリプトの新規作成」を参照してください。

```
vuser_init()
{
connect_server("199.203.78.170", "9000"/* バージョン
=107*/,"       module=e:¥¥appsnca¥¥fnd¥¥7.5¥¥forms¥¥us¥¥fndscsgn
      userid=applsyspub/pub@vision fndnam=apps");
edit_set("FNDSCSGN.SIGNON.USERNAME.0","VISION");
edit_set("FNDSCSGN.SIGNON.PASSWORD.0","WELCOME");
button_press("FNDSCSGN.SIGNON.CONNECT_BUTTON.0");
lov_retrieve_items("Responsibilities",1,17);

      return 0;
}
```

- Netscape の制限により，使用しているマシンで別の Netscape ブラウザがすでに動いている場合は，Netscape 内で Oracle NCA セッションを起動することができません。
- VuGen では，マルチ・プロトコル・モードで Forms Listener Servlet を使用することで Oracle Forms アプリケーションを記録できます。Oracle Forms では，アプリケーション・サーバが **Forms Listener Servlet** を使用して各クライアントの実行時プロセスを作成します。**Forms Server Runtime** プロセスという実行時プロセスは，クライアントとの永続的な接続を維持し，サーバとの間で情報をやり取りします。

  再生時に Forms 4.5 をサポートするには，**cmmdrv.dat** ファイルに以下を設定します。

  ```
  [Oracle_NCA]
  ExtPriorityType=protocol
  WINNT_EXT_LIBS=ncarp110.dll
  WIN95_EXT_LIBS=ncarp110.dll
  LINUX_EXT_LIBS=liboranca.so
  SOLARIS_EXT_LIBS=liboranca.so
  HPUX_EXT_LIBS=liboranca.sl
  AIX_EXT_LIBS=liboranca.so
  LibCfgFunc=oracle_gui_configure
  UtilityExt=lrun_api
  ```

  Forms 6 または 9 のサポートに戻すには，最初の設定に戻します。

名前によるオブジェクトの記録の有効化

Oracle NCA スクリプトを記録するときには，標準ブジェクト ID ではなくオブジェクト名を使用してセッションを記録する必要があります。オブジェクト ID はサーバによって動的に生成され，記録時と再生時とでは異なるため，オブジェクト ID を使用してスクリプトを記録すると，再生は失敗します。**nca_connect_server** ステートメントを調べれば，スクリプトがオブジェクト名で記録されているかどうかを確認できます。

```
nca_connect_server("199.35.107.119","9002"/*version=11i*/,"module=/d1/oracle
/visappl/fnd/11.5.0/forms/US/FNDSCSGN userid=APPLSYSPUB/PUB@VIS
fndnam=apps record=names ");
```

nca_connect_server 関数に **record=names** 引数が含まれていなければ，オブジェクト ID が記録されているということです。オブジェクト名を記録するように VuGen を設定するには，次のいずれかを変更します。

- スタートアップ HTML ファイル
- 記録する URL
- 設定ファイル

すべてのオブジェクトの開発者名を取得する機能は，Oracle Forms6i Patch 9（Oracle Forms Version: 6.0.8.18.3）で初めて導入されました。そのため，Oracle Forms 6i Patch 9 のリリース前に作成された Test Starter Kit スクリプトでは，編集フィールドを除き，オブジェクトの物理的記述に開発者名は含まれません。

スタートアップ HTML ファイル

スタートアップ HTML ファイルにアクセスできる場合は，そのスタートアップ・ファイル，つまり Oracle NCA アプリケーションを起動したときにロードされるファイルに **record=names** フラグを設定すれば，オブジェクト ID ではなくオブジェクト名が記録されます。

アプレット・ビューアの起動時に呼び出されるスタートアップ・ファイルを編集します。次の行を変更します。

```
< PARAM name="serverArgs ..... fndnam=APPS" >
```

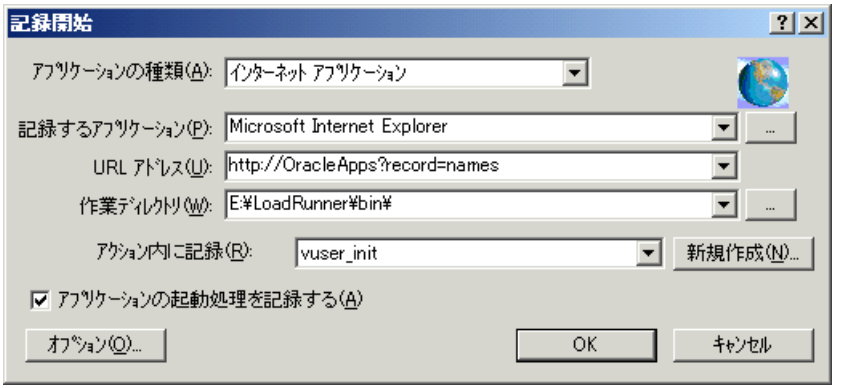
次に，Oracle キー「record=names」を次の形式で追加します。

```
< PARAM name="serverArgs ..... fndnam=APPS record=names" >
```

記録する URL

スタートアップ HTML ファイルにアクセスできない場合は，記録する URL を変更することで，Oracle NCA がオブジェクト ID ではなくオブジェクト名を記録するようにできます。次の方法は，スタートアップ HTML ファイルがロード時にほかのファイルを参照しない場合にのみ有効です。

この方法では，［記録開始］ダイアログ・ボックスの URL の後，つまり記録する URL 名の後に「**?record=names**」を追加します。これにより，VuGen はセッションの中でオブジェクト名を記録できるようになります。



Forms 設定ファイル

Forms Web CGI 設定ファイル **formsweb.cfg** を参照するスタートアップ HTML ファイルがアプリケーションにある場合（よく行われる参照です）は，スタートアップ・ファイルに **record=names** を追加すると，問題が生じる場合があります。

この場合は，設定ファイルに変更を加える必要があります。

設定ファイルに変更を加えてオブジェクト名を記録できるようにするには，次の手順で行います。

1 Forms Web CGI 設定ファイルに移動します。

2 このファイルに新しいパラメータを定義します（以下に示す Web CGI 設定ファイルの例を参照）。

```
serverApp=forecast
serverPort=9001
serverHost=easgdev1.dats.ml.com
connectMode=socket
archive=f60web.jar
archive_ie=f60all.cab
xrecord=names
```

3 スタートアップ HTML ファイルを開き，PARAM NAME="serverArgs" を探します。

4 **record=%xrecord%** のように，変数名を ServerArgs パラメータに引数として追加します。

```
< PARAM NAME="serverArgs" VALUE="module=%form%
userid=%userid% %otherParams% record=%xrecord%" >
```

5 または，Oracle Forms のインストール・ディレクトリにある **basejini.htm** ファイルを編集します。このファイルは，JInitiator スタイルのタグを使用して Forms アプレットを読み込む Web 上のフォームを実行するための標準の HTML ファイルです。basejinin.hmt ファイルで，パラメータ定義に次の行を追加します。

```
< PARAM NAME="recordFileName" VALUE="%recordFileName%" >
```

<EMBED> タグに次の行を追加します。

```
...
serverApp="%serverApp%"
logo="%logo%"
imageBase="%imageBase%"
formsMessageListener="%formsMessageListener%"
recordFileName="%recordFileName%"
```

サーブレット設定ファイル **formsweb.cfg** ではなく basejini.htm ファイルを編集することの難点は，Oracle Forms を再インストールすると，このファイルが置き換えられてしまう点です。これを避けるには，**basejini.htm** ファイルのコピーを作成し，そのコピーを別の場所に保管しておきます。サーブレット設定ファイルで，**baseHTMLJinitiator** パラメータを編集して新しいファイルを指すようにします。

Personal Home Page からの Oracle Applications の使用

Personal Home Page にログインして Oracle Forms 6i を起動する場合，ユーザ・レベルでいくつかのシステム・プロファイル・オプションを設定する必要があります。これらの変数は，全ユーザに適用される**サイト**・レベルではなく，**ユーザ**・レベルで設定することをお勧めします。

「ICX: Forms Launcher」プロファイルを設定するには，次の手順で行います。

1 アプリケーションにサインオンし，［System Administrator］の権限を選択します。

2 ［Navigator］メニューから［**Profile/System**］を選択します。

3 ［**Find System Profile Values**］フォームで次の操作をします。

［**Display:Site**］オプションを選択します。

Users = ＜**ログオン・ユーザ名**＞（operations，mfg など）

Enter Profile =%ICX%Launch%

［**Find**］をクリックします。

4 「**ICX: Forms Launcher**」プロファイルに対する User の値を更新します。

URL に対してパラメータが**渡されていない**場合，ユーザ指定値の後ろに次の文字列を追加します。**?**play=&record=names

URL に対してパラメータが**渡されている**場合，ユーザ指定値の後ろに次の文字列を追加します。**&**play=&record=names

5 トランザクションを保存します。

6 Oracle Forms セッションからログアウトします。

7 Personal Home Page セッションからログアウトします。

8 Personal Home Page から，自分の名前を使って再度サインオンします。

ユーザ・レベルで「ICX: Forms Launcher」プロファイルを更新することができない場合には，［**Application Developer**］権限を開き，ICX_FORMS_LAUNCHER プロファイルに対する［**Updatable**］オプションを選択します。

URL に渡す最初のパラメータは，疑問符（?）で始めなければなりません。その後に続くパラメータはすべてアンパサンド（&）を付けて渡します。ほとんどの場合，URL にパラメータが含まれています。含まれているかどうかは，疑問符を検索することで調べることができます。

Oracle NCA 仮想ユーザ関数の使用

VuGen は，NCA を使った一般的なビジネス・プロセスを実行する間に，本項で説明する関数のほとんどを自動的に記録します。記録される関数には，**nca** という接頭辞が付いています（VuGen の以前のバージョンで **nca** 接頭辞なしで記録された NCA 関数もサポートされています）。また，手作業で任意の関数をプログラミングし，スクリプトに挿入することもできます。ツリー・ビューで作業しているときには，対象となるステップを示す視覚的なアイコンをクリックします。テキスト・ビューでは，必要な関数を手作業で追加できます。Oracle NCA 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。

ボタン・オブジェクト関数

nca_button_double_press	プッシュ・ボタンを 2 回押します。
nca_button_press	プッシュ・ボタンを押します。
nca_button_set	指定されたボタンの状態を設定します。

コンボ・ボックス・オブジェクト関数

nca_combo_select_item	コンボ・ボックスの項目を選択します。
nca_combo_set_item	コンボ・ボックスに新しい項目を設定します。

接続関数

nca_connect_server	Oracle NCA サーバに接続します。
nca_logon_connect	Oracle NCA データベースにログインします。
nca_logon_cancel	Oracle NCA データベースから接続を解除します。

編集オブジェクト関数

nca_edit_box_press	エディット・ボックスのメッセージをクリックします。
nca_edit_click	エディット・オブジェクト内でクリックします。
nca_edit_get_text	エディット・オブジェクト内のテキストを返します。
nca_edit_press	エディット・フィールド内の参照ボタンを押します。
nca_edit_set	エディット・オブジェクト内のすべての内容を置き換えます。

フレックス・オブジェクト関数

nca_flex_click_cell	［Flexfield］ウィンドウ内のテーブル・セルをクリックします。
nca_flex_get_cell_data	Flexfield セルからデータを取得します。
nca_flex_get_column_name	［Flexfield］ウィンドウ内のカラムの名前を取得します。
nca_flex_press_clear	［Flexfield］ウィンドウ内の［**Clear**］をクリックします。
nca_flex_press_find	［Flexfield］ウィンドウ内の［**Find**］をクリックします。
nca_flex_press_help	［Flexfield］ウィンドウ内の［**Help**］をクリックします。
nca_flex_press_lov	［Flexfield］ウィンドウ内の［**List of Values**］ボタンをクリックします。
nca_flex_press_ok	［Flexfield］ウィンドウ内の［**OK**］をクリックします。
nca_flex_set_cell_data	［Flexfield］ウィンドウにデータを挿入します。
nca_flex_set_cell_data_press_ok	データ入力後に［Flexfield］ウィンドウ内の［**OK**］をクリックします。

リスト項目関数

nca_list_activate_item	リスト内で項目を有効にします（ダブルクリック）。
nca_list_select_index_item	インデックスを使ってリスト項目を選択します。
nca_list_select_item	名前を使ってリスト項目を選択します。
nca_lov_auto_select	項目の最初の文字を指定します。
nca_lov_find_value	［List of Values］ウィンドウ内の［**Find**］をクリックします。
nca_lov_get_item_name	エントリのインデックス番号を使って候補値リスト内のエントリの名前を取得します。
nca_lov_retrieve_items	候補値リストを取得します。
nca_lov_select_index_item	インデックス番号を使って候補値リストから項目を選択します。
nca_lov_select_item	候補値リストから項目を選択します。
nca_lov_select_random_item	候補値リストから項目をランダムに選択します。

Java オブジェクト関数

nca_java_action	Java オブジェクトを対象にイベントを実行します。
nca_java_get_value	Java オブジェクトの値を取得します。
nca_java_set_reply_property	Java オブジェクトの応答プロパティを設定します。

メニュー・オブジェクト関数

nca_menu_select_item	メニューから項目を選択します。

メッセージ関数

nca_popup_message_press	ポップアップ・ウィンドウ内のボタンをクリックします。
nca_message_box_press	メッセージ・ウィンドウ内のボタンをクリックします。

オブジェクト関数

nca_obj_get_info	オブジェクト・プロパティの値を返します。
nca_obj_mouse_click	オブジェクト内でクリックします。
nca_obj_mouse_dbl_click	オブジェクト内でダブルクリックします。
nca_obj_status	指定したオブジェクトのステータスが返されます。
nca_obj_type	エディット・ボックスに特殊文字を入力します。

応答オブジェクト関数

nca_response_press_lov	［Response］ボックスのドロップダウン・リスト・ボタンをクリックします。
nca_response_press_ok	［Response］ボックス内の［**OK**］をクリックします。
nca_response_set_cell_data	［Response］ボックス内のセルにデータを挿入します。
nca_response_set_data	［Response］ボックスにデータを挿入します。

スクロール・オブジェクト関数

nca_scroll_drag_from_min	最小位置（0）から指定された距離だけスクロール・オブジェクトをドラッグします。
nca_scroll_line	指定された行数分スクロールします。
nca_scroll_page	指定されたページ数分スクロールします。

セッション関数

nca_console_get_text	コンソール・メッセージを取得します。
nca_set_iteration_offset	オブジェクト ID のオフセット値を設定します。
nca_set_server_response_time	サーバの応答時間を設定します。
nca_set_exception	例外処理の方法を指定します。
nca_set_think_time	思考遅延時間の範囲を設定します。

ツリー・オブジェクト関数

nca_tree_activate_item	NCA ツリー内の項目を有効にします。
nca_tree_collapse_item	ツリー項目を折りたたみます。
nca_tree_expand_item	ツリー項目を展開します。
nca_tree_select_item	NCA ツリー内の項目を選択します。

ウィンドウ・オブジェクト関数

nca_win_get_info	ウィンドウ・プロパティの値を返します。
nca_win_close	ウィンドウを閉じます。
nca_set_window	現在のウィンドウの名前を示します。

C 仮想ユーザ関数（**lr_output_message** や **lr_rendezvous** など）を使ってスクリプトをさらに拡張できます。これらの関数の使用法については，第 6 章 「仮想ユーザ・スクリプトの拡張」を参照してください。

Oracle NCA 仮想ユーザについて

Oracle NCA 仮想ユーザ・スクリプトの作成時，VuGen は，クライアントとアプリケーション・サーバ間の NCA 通信をすべて記録します。記録中，VuGen は「**コンテキスト・センシティブ**」関数を生成します。コンテキスト・センシティブ関数は，データベースに対するアクションを GUI オブジェクト（ウィンドウ，リスト，ボタンなど）を基準にして表します。記録の実行中，VuGen によってコンテキスト・センシティブ関数が仮想ユーザ・スクリプトに挿入されます。

記録を終えた後で，スクリプト内の関数に変更を加えたり，関数を追加してスクリプトを拡張したりできます。仮想ユーザ・スクリプトの拡張については，第 6 章 「仮想ユーザ・スクリプトの拡張」を参照してください。使用可能な Oracle NCA 仮想ユーザ関数の一覧については，646 ページ「Oracle NCA 仮想ユーザ関数の使用」を参照してください。これらの関数の詳細については，「**オンライン関数リファレンス**」（[**ヘルプ**] > [**関数リファレンス**]）を参照してください。

次のコード例では，ユーザがリストから項目を選択して（**nca_list_activate_item**），ボタンを押し（**nca_button_press**），リストの値を取得（**nca_lov_retrieve_items**）しました。そして，エディット・フィールド内をクリック（**nca_edit_click**）しています。オブジェクトの論理名は，これらの関数のパラメータとなっています。

```
...
nca_lov_select_item("Responsibilities","General Ledger, Vision
Operations");
nca_list_activate_item("FNDSCSGN.NAVIGATOR.LIST.0","+ Journals");
nca_list_activate_item("FNDSCSGN.NAVIGATOR.LIST.0","      Enter");
nca_button_press("GLXJEENT.TOOLBAR.LIST.0");
nca_lov_find_value("Batches","");
nca_lov_retrieve_items("Batches",1,9);
nca_lov_select_item("Batches","AR 1020 Receivables 2537: A 1020");
nca_edit_click("GLXJEENT.FOLDER_QF.BATCH_NAME.0");
...
```

Oracle Configurator アプリケーションを対象に実行するテストなど特定のテストでは，ある関数によって返される情報がセッション全体で必要になります。VuGen は，スクリプトに **web_reg_save_param** 関数を挿入することによって，動的な情報を自動的にパラメータに保存します。次の例では，接続情報が **NCAJServSessionID** という名前のパラメータに保存されています。

```
web_reg_save_param ("NCAJServSessionId", "LB=¥r¥n¥r¥n", "RB=¥r",
        LAST);
web_url("f60servlet",
        "URL=http://usscifforms05.sfb.na/servlet/f60servlet¥?config=mult",
        LAST);
```

上記の例では，右の境界は ¥r です。実際の右の境界は，システムによって異なる場合があります。

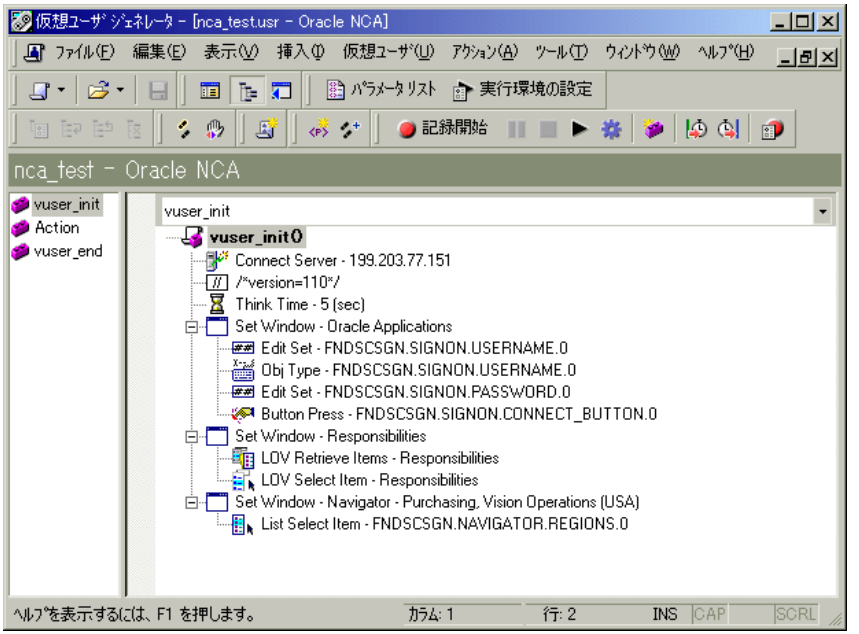
ツリー・ビューとスクリプト・ビューの切り替え

VuGen で Oracle NCA 仮想ユーザ・スクリプトを表示または編集するときには，スクリプトをアイコン形式のツリー・ビューで表示するか，テキスト形式のスクリプト・ビューで表示するかを選択します。

Oracle NCA 仮想ユーザ・スクリプトのツリー・ビューを表示するには，次の手順で行います。

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・スクリプトが，アイコン形式のツリー・ビューで表示されます。すでにツリー・ビューを表示している場合は，このメニュー項目は選択できません。

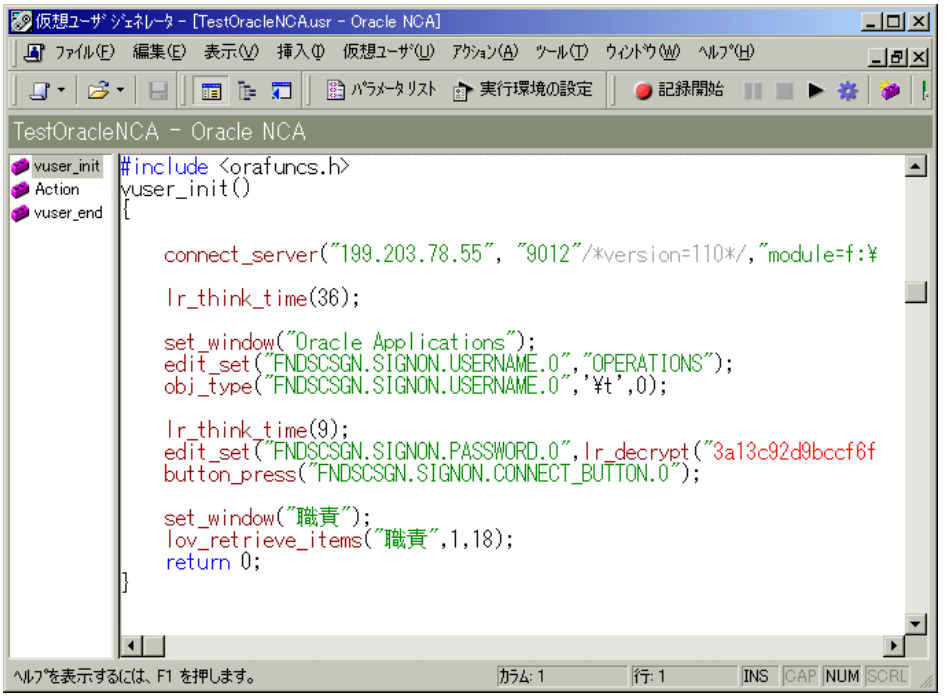
ツリー・ビュー



スクリプト・ビューを表示するには，次の手順で行います。

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。仮想ユーザ・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・ビュー



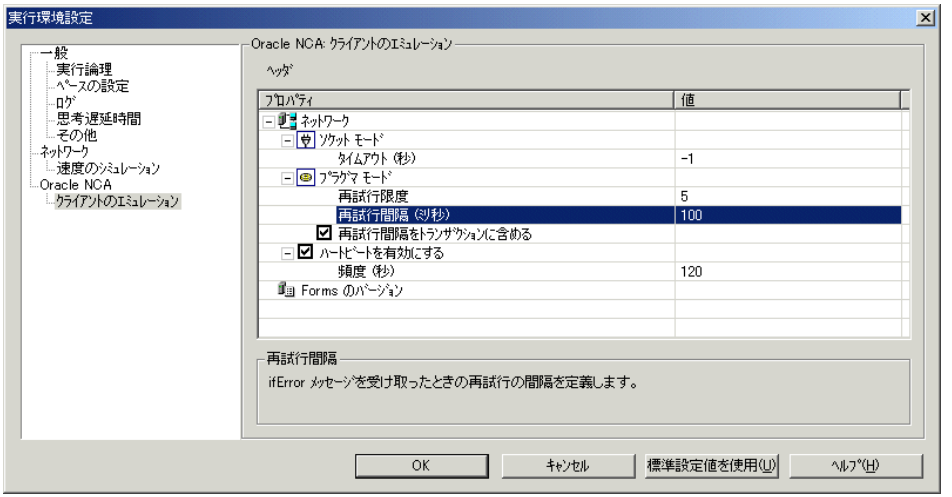
実行環境の設定

スクリプトを実行する前に，実行環境の設定を行って，スクリプトが実際のユーザを正確にエミュレートするようにします。すべてのプロトコルに共通の一般的な実行環境の設定（思考遅延時間，間隔，ログ記録など）については，第 9 章 「実行環境の設定」を参照してください。ネットワークに関係する設定については，第 10 章 「インターネット実行環境の設定」を参照してください。

次の項では，Oracle NCA 仮想ユーザ専用の実行環境の設定について説明します。これらの実行環境の設定を行うことで，エミュレートする通信パラメータを指定できます。

クライアント・エミュレーションの実行環境設定

Oracle NCA クライアントが正確にエミュレートされるように，ネットワークを設定できます。



以下の項目を設定できます。

［**ネットワーク**］＞［**ソケット モード**］＞［**タイムアウト**］：サーバからの応答を Oracle NCA 仮想ユーザが待機する時間です。標準設定値は -1 です。-1 の場合，タイムアウトは無効になり，クライアントはいつまでも待機します。

［**プラグマ モード**］：プラグマ・モードでは，Oracle によって定義されているプラグマ・モードで通信が行われます。プラグマ・モードの通信は，HTTP および Servlet よりも上位の通信レベルで，メッセージを定期的に送信するという特徴があります。このモードでは，クライアントはサーバが直ちにデータを返さ

ないことを認識します。サーバは，要求されたデータを送ることができるまで，所定の間隔でメッセージを送信します。

［**再試行限度**］：クライアントがエラーを発行するまでにサーバから受け付ける **IfError** メッセージの最大数を指定します。IfError メッセージは，サーバからクライアントに定期的に送られるメッセージで，できるだけ早くデータを返すことを通知するものです。

［**再試行間隔**］：**IfError** メッセージが生じた場合の再試行の間隔を指定します。

［**再試行間隔をトランザクションに含める**］：再試行の間の間隔もトランザクション持続時間に含めます。

プラグマ・モードでの記録の詳細については，662 ページ「プラグマ・モードでの記録」を参照してください。

［**ハートビートを有効にする**］：Oracle サーバに送信されるハートビートを有効または無効にします。ハートビートは，サーバとの通信が正常に行われていることを確認する処理です。Oracle NCA サーバに高い負荷がかかっている場合は，ハートビートを無効にします。ハートビートを有効にした場合は，ハートビート・メッセージをサーバに送信する間隔を設定できます。

［**Forms のバージョン**］：記録時に検出された Oracle Forms のバージョンを示します。この設定は，記録を行った後にサーバがアップグレードされた場合にのみ変更します。

クライアントのエミュレーションを設定するには，次の手順で行います。

1 ［実行環境設定］ダイアログ・ボックスを開きます。［**仮想ユーザ**］＞［**実行環境の設定**］を選択するか，VuGen のツールバーで［**実行環境の設定を編集**］ボタンをクリックします。LoadRunner コントローラから［実行環境設定］ダイアログ・ボックスを開くには，［**実行環境の設定**］ボタンをクリックします。

2 実行環境設定ツリーから［**Oracle NCA：クライアントのエミュレート**］ノードを選択します。

3 ネットワーク・タイムアウト値を秒単位で設定します。クライアントにサーバの応答をいつまでも待機させるには，標準設定値 -1 を使用します。

4 プラグマ・モードで作業をするときは，クライアントがエラーを発行するまでに受け付ける **IfError** メッセージの再試行回数［**再試行限度**］を指定します。標準設定値は 20 です。

5 Oracle NCA サーバへのハートビートの送信を有効にするには，［**ハートビートを有効にする**］オプションを選択します。次の行に，ハートビートを送信する間隔を秒単位で指定します。標準設定値は 120 秒です。

6 ［**OK**］をクリックして設定を適用し，スクリプトを実行します。

Oracle NCA アプリケーションのテスト

次の項では，安全な Oracle NCA アプリケーションおよびサーブレットをテストするためのヒントをいくつか取り上げます。

安全な Oracle NCA アプリケーションのテスト

- 記録するプロトコルを選択するとき，プロトコル・リストから **Oracle NCA** だけ選択すればよく，**Web** プロトコルを選択する必要はありません。VuGen は，内部でセキュリティ情報を記録するため，明示的な Web 関数は不要です。
- ［ポートの割り当て］記録オプションで，ポート 443 の既存のエントリを削除して，Oracle サーバ名の新しいエントリを作成します。

  ［**サービス ID**］: HTTP
  ［**対象サーバ**］: Oracle Forms サーバの IP アドレスまたはロング・ホスト名
  ［**ポート**］: 443
  ［**接続の種類**］: SSL
  ［**SSL バージョン**］: 使用している SSL のバージョン。不明な場合は「SSL 2/3」を選択します。

  詳細については，第 5 章 「ポートの割り当て設定」を参照してください。
- **nca_connect_server** コマンド実行中に NCA HTTPS スクリプトを再生するときに問題が生じた場合は，スクリプトの先頭に次の関数を挿入します。

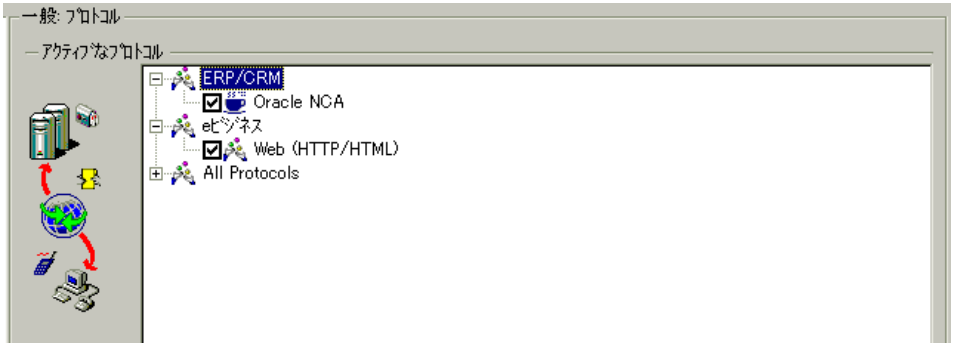
```
web_set_sockets_option("SSL_VERSION","3");
```

サーブレットおよびその他の Oracle NCA アプリケーションのテスト

NCA セッションの中には，サーブレットを使用するものがあります。

- Forms Listener サーブレット
- NCA および HTTP 通信の両方を使用するアプリケーションまたはモジュール（Oracle Configurator など）
- NCA アプリケーションの初期化（アプレット，jar ファイル，gif ファイルのダウンロード）

サーブレットを記録するときは，Oracle NCA 関数と Web 関数の両方を記録する必要があります。そのためには，最初にマルチ・プロトコル・スクリプトを作成します。また，Oracle NCA を対象としたシングル・プロトコル・スクリプトを作成した場合は，［記録オプション］で［**一般：プロトコル**］ノードを開き，Web プロトコルを有効にします。これで，記録が開始できます。



アプリケーションでサーブレットが使用されているかどうか不確かな場合は，script ディレクトリの **default.cfg** ファイルを確認します。次のエントリを探します。

UseServletMode=
値が 1 または 2 ならば，サーブレットが使用されています。Oracle NCA に加えて HTTP の記録も有効にする必要があります。

すでにスクリプトが記録されている場合は，Web 関数を含めるようにコードを自動的に再生成できます。再度記録をする必要はありません。［**ツール**］＞［**仮想ユーザを再生成**］を選択し，［プロトコル］セクションで Web プロトコルを選択します。

記録モードの指定

Oracle NCA スクリプトを記録するとき，VuGen は自動的に正しい接続モード，つまり HTTP モードかソケット・モードかを判断します。通常は VuGen によってシステムの構成が自動的に検出されるため，記録の設定を変更する必要はありません。標準のポート割り当てがほかのアプリケーションによって予約されているシステムの場合，記録モードに応じて［ポートの割り当て］の設定を変更しなければならないことがあります。

記録モードは，次のいずれかの方法で判断できます。

➤ NCA アプリケーションを使用しているときに，Java コンソールを開きます。

```
proxyHost=null
proxyPort=0
connectMode=HTTP
Forms Applet version is : 60812
```

connectMode エントリに，**HTTP**，**HTTPS**，または **socket** が表示されます。

➤ NCA セッションの記録後に，仮想ユーザ・ディレクトリの **default.cfg** ファイルを開き，**UseHttpConnectMode** エントリの値を確認します。

```
[HttpConnectMode]
UseHttpConnectMode= 2
// 0 = socket 1 = http 2 = https
```

［サーバ エントリ］ダイアログ・ボックスで新しいポート割り当てを定義する場合，HTTP または HTTPS モードのときは［**サービス ID**］として「**HTTP**」を選択します。ソケット・モードのときは，［**サービス ID**］として「**NCA**」を選択します。

ポート割り当ての設定の詳細については，第 5 章 「ポートの割り当て設定」を参照してください。

ロード・バランシングに向けた Oracle NCA ステートメントの相関

LoadRunner は，複数のアプリケーション・サーバを対象とするロード・バランシングをサポートしています。HTTP の戻り値を **nca_connect_server** パラメータと相関させます。以降，LoadRunner はテスト実行時に，対応するサーバに接続してロード・バランシングを適用します。

ロード・バランシングに向けてステートメントを相関させるには，次の手順で行います。

1 **マルチ・プロトコル・スクリプトを記録します。**

Oracle NCA および Web プロトコルのマルチ・プロトコル・スクリプトを記録します。必要なアクションを実行し，スクリプトを保存します。

2 **ホストのパラメータと引数を定義します。**

パラメータ化用に 2 つの変数 serverHost および serverArgs を定義します。

```
web_set_max_html_param_len("512");
web_reg_save_param("serverHost", "NOTFOUND=ERROR",
        "LB= < PARAM name=¥"serverHost¥" value=¥"","RB=¥" > " ,
LAST );
web_reg_save_param("serverArgs", "NOTFOUND=ERROR",
        "LB= < PARAM name=¥"serverArgs¥" value=¥"","RB=¥" > "
,LAST );
```

3 **web_url** 関数を呼び出して，serverHost と serverArgs に値を割り当てます。

```
web_url("step_name", "URL=http://server1.merc-int.com/test.htm", LAST);
```

4 次の **nca_connect_server** ステートメントを変更します。

```
nca_connect_server("199.203.78.170",
        9000"/*version=107*/, "module=e:¥¥appsnca...fndnam=apps ");
```

変更後は次のようになります。

```
nca_connect_server(" < serverHost > ", "9000"/*version=107*/, " <
serverArgs > ");
```

スクリプトは次のようになるはずです。

```
web_set_max_html_param_len("512");
web_reg_save_param("serverHost", "NOTFOUND=ERROR",
        "LB= < PARAM name=¥"serverHost¥" value=¥"","RB=¥" > " ,
LAST );
web_reg_save_param("serverArgs", "NOTFOUND=ERROR",
        "LB= < PARAM name=¥"serverArgs¥" value=¥"","RB=¥" > "
,LAST );
web_url("step_name", "URL=http://server1.merc-int.com/test.htm", LAST);
nca_connect_server(" < serverHost > ","9000"/*version=107*/," <
serverArgs > ");
```

そのほかに推奨される相関

Oracle NCA セッションを記録するとき，動的な値，つまり記録セッションおよび再生セッションごとに変化する値が VuGen によって記録されます。よく使用される動的な 2 つの引数が，icx_ticket と JServSessionIdroot です。

icx_ticket

icx_ticket 変数は，**web_url** 関数および **nca_connect_server** 関数で送信する情報の一部です。

```
web_url("fnd_icx_launch.runforms",
"URL=http://ABC-
123:8002/pls/VIS/fnd_icx_launch.runforms¥?ICX_TICKET=5843A55058947ED3
&RESP_APP=AR&RESP_KEY=RECEIVABLES_MANAGER&SECGRP_KEY=S
TANDARD", LAST);
```

この **icx_ticket** の値は記録ごとに異なります。この変数には，クライアントによって送信されたクッキー情報が格納されます。記録を相関させるには，記録された icx_ticket 値の最初の出現の前に **web_reg_save_param** を追加します。

```
web_reg_save_param("icx_ticket", "LB=TICKET=", "RB=&RES", LAST);

...

web_url("fnd_icx_launch.runforms",
"URL=http://ABC-
123:8002/pls/VIS/fnd_icx_launch.runforms¥?ICX_TICKET= ＜ icx_ticket ＞
&RESP_APP=AR&RESP_KEY=RECEIVABLES_MANAGER&SECGRP_KEY=
STANDARD", LAST);
```

注：web_reg_save_param の左右の境界は，アプリケーションの設定によって異なる場合があります。

JServSessionIdroot

JServSessionIdroot 値は，セッション ID を格納するためにアプリケーションによって設定されるクッキーです。ほとんどの場合，この値は VuGen によって自動的に相関され，**web_reg_save_param** 関数が挿入されます。この関数が自動的に追加されなかった場合は，手作業で追加し，値をすべてパラメータ名で置き換えます。

相関させる必要がある値を特定するには，実行ログを開き（[**表示**] > [**出力ウィンドウ**]），応答の本体を探します。

vuser_init.c(8): Set-Cookie: JServSessionIdroot=my1sanw2n1.JS4; path=/¥r¥n
vuser_init.c(8): Content-Length: ¥r¥n
vuser_init.c(8): Content-Type: text/plain¥r¥n
vuser_init.c(8): ¥r¥n
vuser_init.c(8): 81-byte response body for "http://ABC-123/servlet/oracle.forms.servlet.ListenerServlet?ifcmd=getinfo&ifhost=mercury&ifip=123.45.789.12" (RelFrameId=1)
vuser_init.c(8): **/servlet/oracle.forms.servlet.ListenerServlet?JServSessionIdroot=my1sanw2n1.JS4**
¥r¥n

この動的な値を相関させるには，最初の出現の前に **web_reg_save_param** 関数を挿入し，スクリプト全体にわたって変数値をパラメータ名で置き換えます。この例では，左右の境界は ¥r と ¥n ですが，使用する環境での正確な境界を知るために，個別の環境を確認する必要があります。

```
web_reg_save_param("NCAJServSessionId","LB=¥r¥n¥r¥n","RB=¥r","ORD=1",LAST);

web_url("f60servlet",
    "URL= http://ABC-
"123/servlet/oracle.forms.servlet.ListenerServlet?ifcmd=getinfo&"
"ifhost=mercury&ifip=123.45.789.12", LAST);

web_url("oracle.forms.servlet.ListenerSer",
    "URL=http://ABC-123 < NCAJServSessionId > ?ifcmd=getinfo&"
"ifhost=mercury&ifip=123.45.789.12", LAST);
```

プラグマ・モードでの記録

Oracle NCA 仮想ユーザのクライアント側では，サーバに対して **Pragma** という名前の追加ヘッダーを送信するように設定できます。このヘッダーは，次のように振る舞うカウンタです。NCA ハンドシェイクの最初のメッセージは 1 という値を持っています。ハンドシェイクに続くメッセージは，3 からカウントが始まります。カウンタの値は，クライアントによって送信されるメッセージごとに 1 つ増加します。

サーバから受信したメッセージの種類が **plain¥text** で，メッセージの本体が ifError:#/#00 で始まる場合，クライアントはサーバに 0 バイトのメッセージを送信し，Pragma 値はマイナスに変更されます。クライアントがサーバからの情報の受信に成功すると，マイナス記号は元に戻ります。

Pragma ヘッダーの記録は，マルチ・プロトコル・モード（Oracle NCA および Web）だけでサポートされます。プラグマ・モードは，スクリプトの **default.cfg** ファイル内で特定できます。プラグマ・モードで操作すると，UseServletMode は 2 に設定されます。

```
[HttpConnectMode]
UseHttpConnectMode=1
RelativeURL= < NCAJServSessionId >
UseServletMode=2
```

プラグマに関係する実行環境の設定の詳細については，654 ページ「クライアント・エミュレーションの実行環境設定」を参照してください。

プラグマ・モードかどうかを知るには，WinSock レベルの記録を実行し，バッファの内容を確認します。最初の例では，バッファにカウンタとして Pragma 値が含まれています。

```
send buf108
        "POST
/ss2servlet/oracle.forms.servlet.ListenerServlet?JServSessionIdss2ser"
        "vlet=gk5q79uqy1 HTTP/1.1¥r¥n"
        "Pragma: 1¥r¥n"
        ...
send buf110
        "POST
/ss2servlet/oracle.forms.servlet.ListenerServlet?JServSessionIdss2ser"
        "vlet=gk5q79uqy1 HTTP/1.1¥r¥n"
        "Pragma: 3¥r¥n"
        ...
```

次の例では，バッファにエラー・インジケータとして Pragma 値が含まれています。

```
recv buf129 281
        "HTTP/1.1 200 OK¥r¥n"
        "Date: Tue, 21 May 2002 00:03:48 GMT¥r¥n"
        "Server: Oracle HTTP Server Powered by Apache/1.3.19 (Unix)
mod_fastcgi/2.2"
        ".10 mod_perl/1.25 mod_oprocmgr/1.0¥r¥n"
        "Content-Length: 13¥r¥n"
        "Content-Type: text/plain¥r¥n"
        "¥r¥n"
        "ifError:8/100"

send  buf130
        "POST
/ss2servlet/oracle.forms.servlet.ListenerServlet?JServSessionIdss2ser"
        "vlet=gk5q79uqy1 HTTP/1.1¥r¥n"
        "Pragma: -12¥r¥n"
        ...
```

第 48 章

SAPGUI 仮想ユーザ・スクリプトの作成

成長を続けている ERP（Enterprise Resource Planning）の分野において，SAP は企業が自社のすべてのビジネス・プロセスを管理できるようにするソフトウェア・ソリューションを提供しています。Mercury は，SAP ソリューションのモジュールを機能テスト・レベルと負荷テスト・レベルの両方でテストするためのツールを提供しています。

本章では，SAPGUI for Windows クライアントをテストするための LoadRunner のソリューションについて説明します。mySAP Workplace および Portal クライアントのためのソリューションをテストする方法の詳細については，第 49 章「SAP-Web 仮想ユーザ・スクリプトの作成」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。

- SAPGUI 仮想ユーザのための環境の確認
- SAPGUI 仮想ユーザ・スクリプトの作成
- SAPGUI 記録オプションの設定
- SAPGUI 仮想ユーザ・スクリプトについて
- SAPGUI 仮想ユーザ・スクリプトの拡張
- SAPGUI 仮想ユーザ・スクリプトの再生
- SAPGUI 実行環境の設定
- SAPGUI の関数
- SAPGUI 仮想ユーザ・スクリプトに関するヒント
- SAPGUI 仮想ユーザ・スクリプトのトラブルシューティング
- その他の参考資料

以降の情報は，SAPGUI プロトコルを対象とします。

SAPGUI 仮想ユーザ・スクリプトの開発について

セッションを記録する前に，モジュールとクライアント・インタフェースが VuGen によってサポートされていることを確認します。以降では，SAP ビジネス・アプリケーションの SAP クライアント・モジュールについて説明します。

- SAP Web クライアントまたは mySAP.com：SAP-Web 仮想ユーザ・タイプを使用します。
- SAPGUI for Windows：Windows ベースのクライアント。
- SAPGUI for Java：このクライアントはサポートされていません。

バージョン 6.10 以前：QuickTest Professional for R/3 を使用します。負荷テストを実行するには，LoadRunner コントローラ内のスクリプトを SAP 仮想ユーザとして実行します。

バージョン 6.20 以降：

機能テストの場合：mySAP.com クライアント用の QuickTest Professional アドインを使用します。

負荷テストの場合：LoadRunner SAPGUI プロトコルを使用し，VuGen にスクリプトを作成して，コントローラでシナリオを実行します。

通常のビジネス・プロセスは VuGen のレコーダを使用して記録します。VuGen では，SAP ビジネス・プロセス中の SAPGUI for Windows クライアントのアクティビティを記録し，仮想ユーザ・スクリプトを生成できます。SAPGUI for Windows クライアント内でアクションを実行すると，このアクティビティを説明する関数が生成されます。各関数の先頭には，**sapgui** という接頭辞が付きます。再生中，この関数は SAPGUI オブジェクトでのユーザ・アクティビティをエミュレートします。たとえば，**sapgui_select_radio_button** によって「**Blue**」ラジオ・ボタンが選択されます。

```
sapgui_select_radio_button("Blue",
        "usr/radRB7",
        BEGIN_OPTIONAL,
           "AdditionalInfo=sapgui1027",
        END_OPTIONAL);
```

SAPGUI 仮想ユーザのための環境の確認

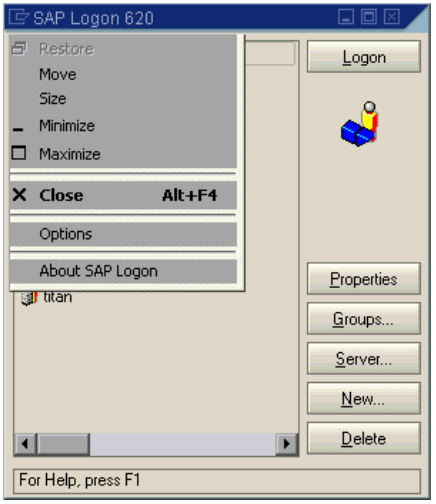
SAPGUI 仮想ユーザを記録するためのシステムの確認および設定の基本的な手順については，「パッチ・レベルの確認」および「スクリプティングの有効化」を参照してください。環境が適切に設定されれば，通常の SAP セッションを記録して VuGen で再生できます。

パッチ・レベルの確認

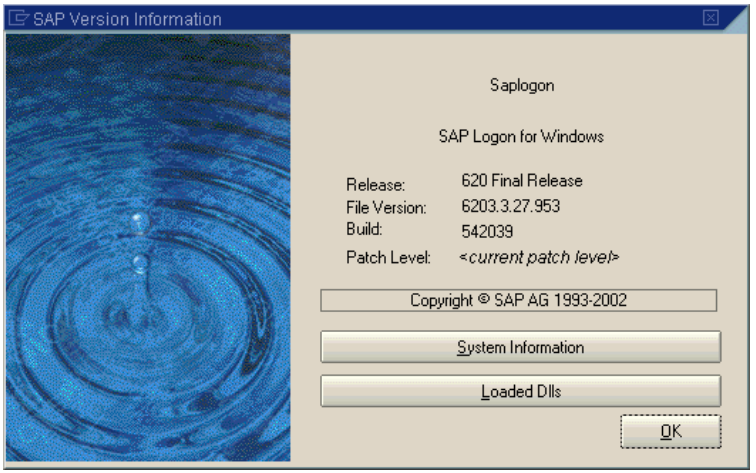
SAPGUI for Windows クライアントのパッチ・レベルは［About］ボックスから確認できます。サポートされている最も低いパッチ・レベルは 32 です。

パッチ・レベルを確認するには，次の手順で行います。

1 SAPGUI ログオン・ウィンドウを起動します。［SAP Logon］ダイアログ・ボックスの左上隅をクリックし，メニューから［**About SAP Logon**］を選択します。



2［SAP Version Information］ダイアログ・ボックスが開きます。パッチ・レベルのエントリが 32 以上であることを確認します。



スクリプティングの有効化

マーキュリー・インタラクティブによる SAPGUI for Windows クライアントに対するサポート機能では，SAP の Scripting API を利用しています。この API により，仮想ユーザは SAPGUI クライアントと対話したり，通知を受け取ったり，操作を実行したりできるようになります。

Scripting API が利用できるのは，SAP Kernel の最近のバージョンだけです。スクリプティングをサポートするバージョンのカーネルでは，オプションは標準で無効になっています。マーキュリー・インタラクティブのツールを使用するには，まず SAP サーバが Scripting API をサポートしていることを確認し，サーバとクライアントの両方で Scripting API を有効にする必要があります。詳細およびパッチのダウンロードについては，『SAP OSS note #480149』を参照してください。

LoadRunner には，システムでスクリプティングがサポートされているかどうかを確認するユーティリティが付属しています。このユーティリティ **VerifyScript.exe** は CD の **Patches and Tools** ディレクトリにあります。詳細については，このユーティリティに付属の Readme ファイルを参照してください。

以降の項では，スクリプティングを有効にするのに必要な手順について詳しく説明します。

- 設定の確認
- SAP Application Server でのスクリプティングの有効化
- SAPGUI 6.20 Client でのスクリプティングの有効化

設定の確認

スクリプティングを有効にするには，まず正しいバージョンのカーネルがインストールされていることを確認し，必要に応じてアップデートします。

SAP Application Server のバージョン別に必要な最低限のカーネル・パッチ・レベルを次の表で確認します。必要に応じて，最新のパッチをダウンロードしてインストールします。

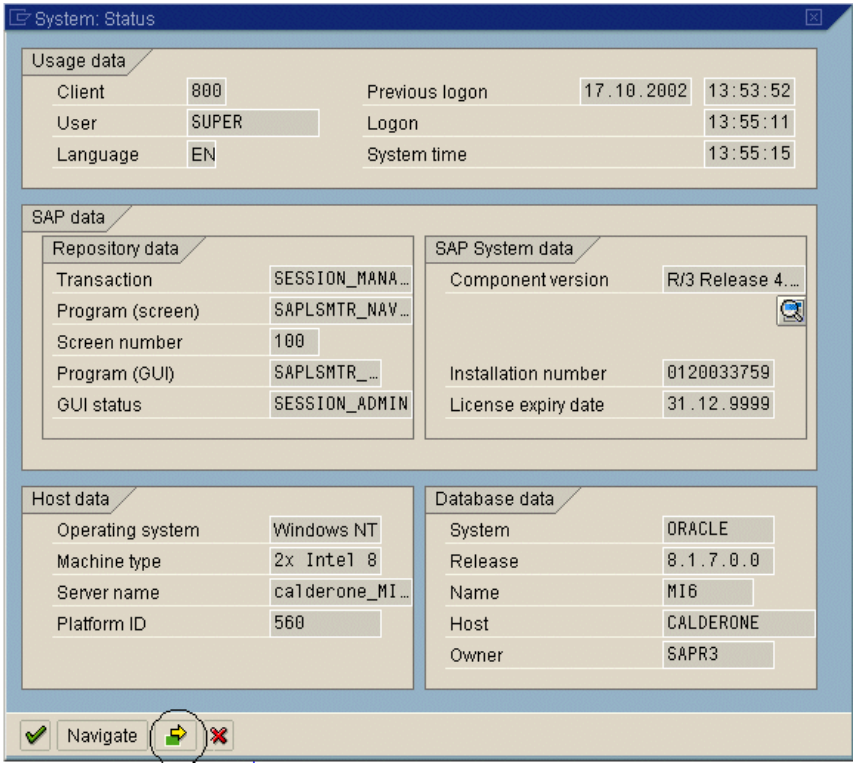
ソフトウェア・コンポーネント	リリース	パッケージ名	カーネル・パッチ・レベル
SAP_APPL	31I	SAPKH31I96	Kernel 3.1I レベル 650
SAP_APPL	40B	SAPKH40B71	Kernel 4.0B レベル 903
SAP_APPL	45B	SAPKH45B49	Kernel 4.5B レベル 753
SAP_BASIS	46B	SAPKB46B37	Kernel 4.6D レベル 948
SAP_BASIS	46C	SAPKB46C29	Kernel 4.6D レベル 948
SAP_BASIS	46D	SAPKB46D17	Kernel 4.6D レベル 948
SAP_BASIS	610	SAPKB61012	Kernel 6.10 レベル 360

カーネル・パッチ・レベルを確認するには，次の手順で行います。

1 SAP システムにログインします。

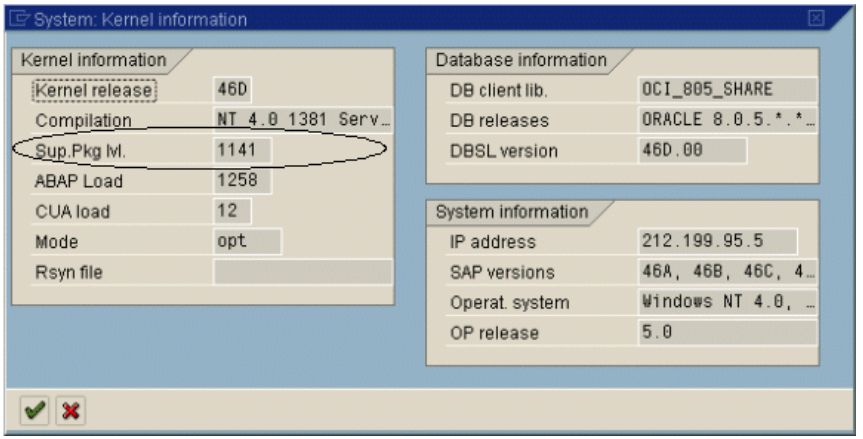
2 ［**System**］ > ［**Status**］を選択します。

3 黄色い矢印の付いた［**Other kernel information**］ボタンをクリックします。



4［**Kernel Information**］セクションで，［**Sup. Pkg. lvl**］の値を確認します。

レベルが 948 より低い場合は，最新のバージョンのカーネルをダウンロードして，既存のカーネルをアップグレードする必要があります。このアップグレード方法の詳細については『SAP OSS note #480149』を参照してください。

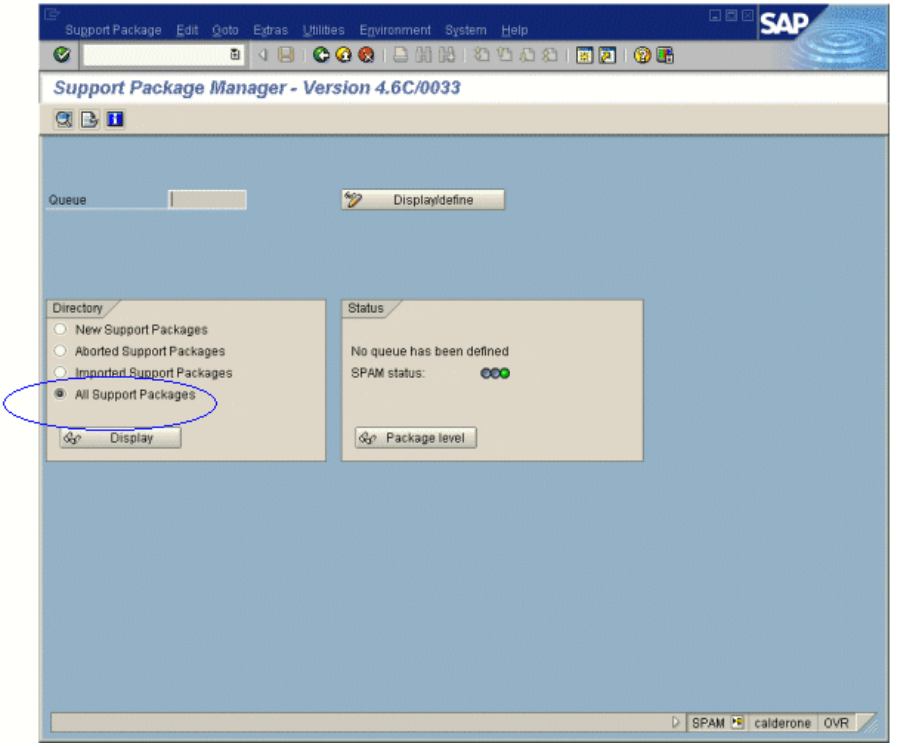


R/3 サポート・パッケージを確認するには，次の手順で行います。

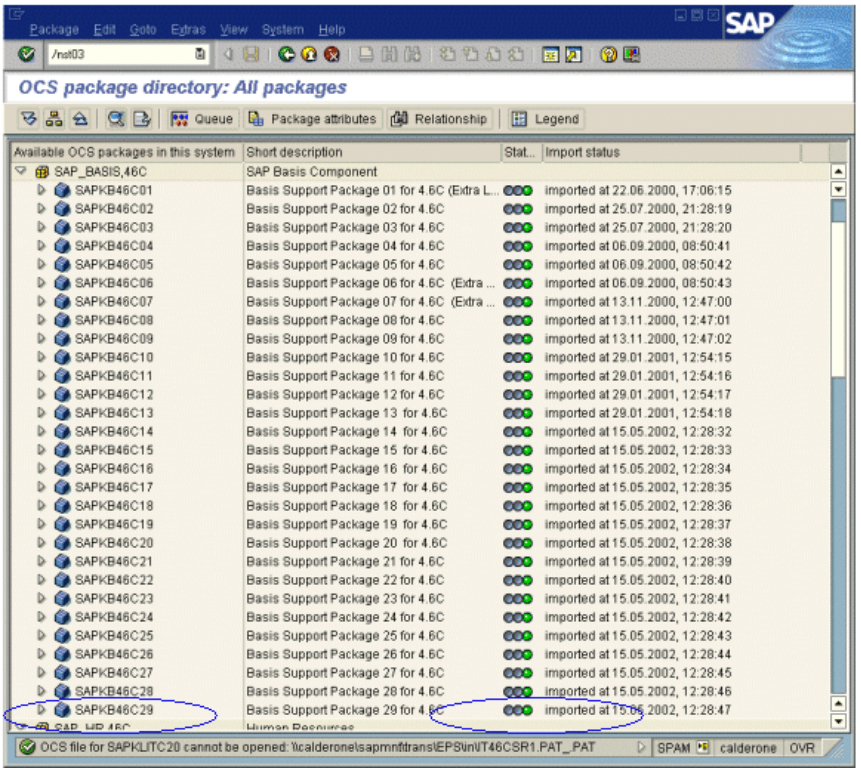
1 SAP システムにログインします。

2 SPAM トランザクションを実行します。

3 ［**Directory**］セクションで，［**All Support Packages**］を選択し，［**Display**］ボタンをクリックします。



4 SAP_BASIS, 4.6C に SAPKB46C29 がインストールされていることを確認します。インストールされていれば，［Status］カラムに緑色の丸が表示されます。



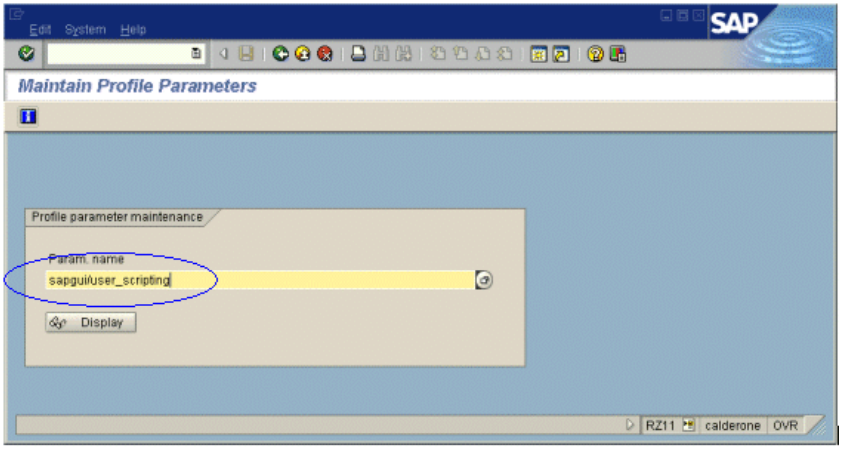
OCS パッケージがインストールされていない場合は，www.sap.com Web サイトからダウンロードしてインストールします。詳細については，『SAP OSS note ＃ 480149』を参照してください。

SAP Application Server でのスクリプティングの有効化

スクリプティングを有効にするには，管理者権限のあるユーザがアプリケーション・サーバで **sapgui/user_scripting** プロファイル・パラメータを **TRUE** に設定します。すべてのユーザに対してスクリプティングを有効にするには，すべてのアプリケーション・サーバでこのパラメータを設定します。特定のユーザ・グループに対してスクリプティングを有効にするには，必要なアクセス制限のかかったアプリケーション・サーバでパラメータを設定します。

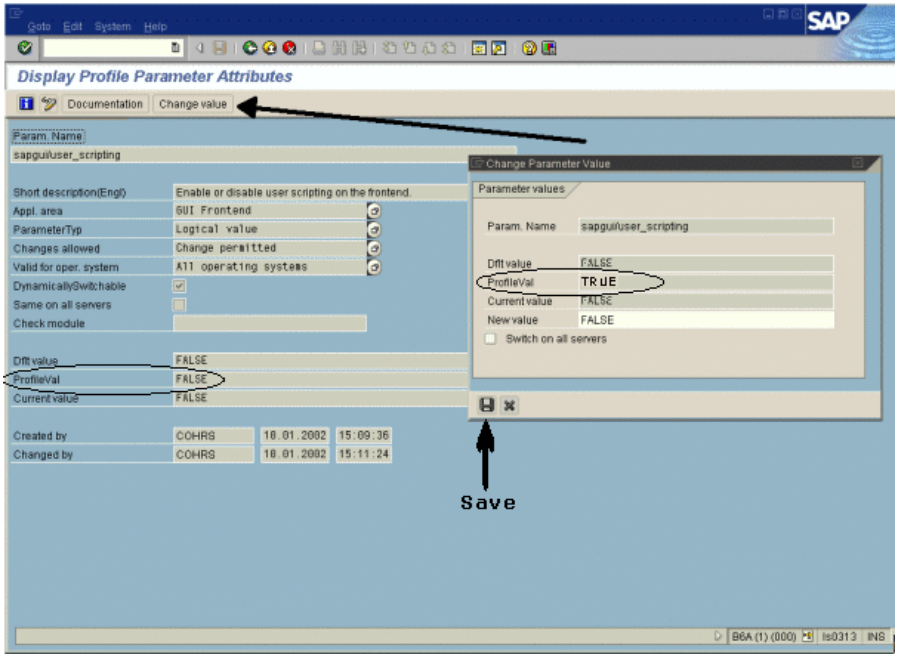
プロファイル・パラメータを変更するには，次の手順で行います。

1 トランザクション **rz11** を開きます。パラメータ名 **sapgui/user_scripting** を指定し，［**Display**］ボタンをクリックします。［Display Profile Parameter Attributes］ウィンドウが開きます。



ステータス・バーに「**Parameter name is unknown**」というメッセージが表示された場合は，最新の Support Package が見当たらないことを示しています。アプリケーション・サーバの SAP BASIS とカーネルのバージョンに対応する Support Package をインポートします。詳細については，669 ページ「設定の確認」を参照してください。

2 **Profile Val** が FALSE の場合は，値を変更する必要があります。ツールバーの［**Change value**］ボタンをクリックします。［Change Parameter Value］ウィンドウが開きます。［**ProfileVal**］ボックスに **TRUE** と入力し，［**Save**］（アイコン）ボタンをクリックします。



変更を保存するとウィンドウが閉じ，**ProfileVal** が TRUE に設定されます。

3 アプリケーション・サーバを再起動します。この変更はシステムにログオンしたときにのみ有効になります。

更新された **ProfileVal** がサーバを再起動しても変更されなければ，アプリケーション・サーバのカーネルが古いので，必要なカーネル・パッチをインポートします。詳細については，669 ページ「設定の確認」に記載されています。

Profile Value は，以下のバージョンのカーネルでは，トランザクション rz11 を使用して動的に有効化できます。アプリケーション・サーバを再起動する必要はありません。

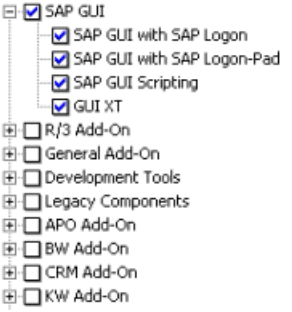
リリース	カーネル・バージョン	パッチ・レベル
4.6B，4.6C，4.6D	4.6D	972
6.10	6.10	391
6.20	すべてのバージョン	すべてのレベル

SAPGUI 6.20 Client でのスクリプティングの有効化

VuGen でスクリプトを実行できるようにするには，SAPGUI クライアントでもスクリプティングを有効にする必要があります。また，接続が確立されたときやスクリプトが GUI プロセスにアタッチされたときなどに表示される特定のメッセージが表示されないようにクライアントを設定する必要もあります。

VuGen で使用できるように SAPGUI クライアントを設定するには，次の手順で行います。

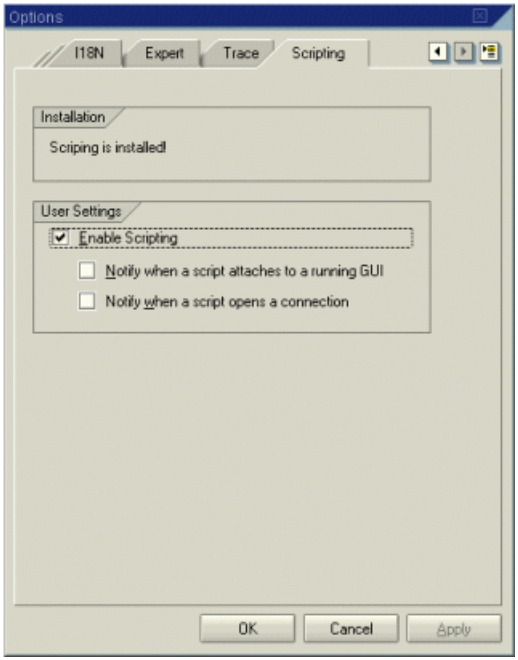
➤ **インストール中：**SAPGUI クライアントのインストール中に，［**SAP GUI Scripting**］オプションを有効にします。



➤ **インストール後：**警告メッセージが表示されないようにします。SAPGUI クライアントで［Options］ダイアログ・ボックスを開きます。［**Scripting**］タブを選択し，次のオプションをクリアします。

1 ［**Notify when a script attaches to a running GUI**］

2 ［**Notify when a script opens a connection**］



また，次のレジストリ・キーの中で **WarnOnAttach** と **WarnOnConnection** の値を 0 に設定することによっても，これらのメッセージが表示されないようにできます。

HKCU¥SOFTWARE¥SAP¥SAPGUI Front¥SAP Frontend Server¥Security

SAPGUI 仮想ユーザ・スクリプトの作成

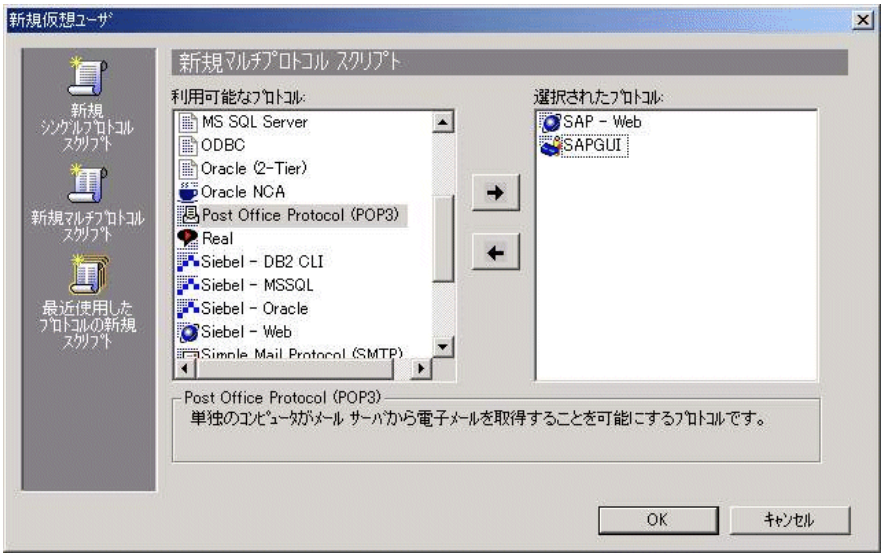
SAPGUI 仮想ユーザ・スクリプト作成の第一歩は，仮想ユーザとスクリプトのタイプを選択することです。SAP の仮想ユーザ・タイプ **SAPGUI** は，**ERP/CRM** カテゴリの下にあります。シングル・プロトコルとマルチ・プロトコルのどちらの仮想ユーザ・スクリプトでも作成できます。

SAPGUI 仮想ユーザ・スクリプトの作成は，次の手順で行います。

1 VuGen を起動し，［**ファイル**］＞［**新規作成**］を選択します。

2 簡単な SAPGUI クライアント・セッション（ブラウザのコントロールなし）を記録するには，**SAPGUI** タイプの仮想ユーザを使用して，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。

3 ブラウザのコントロールを使用する SAPGUI を記録するには，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。**SAPGUI** および **SAP-Web** の両方の仮想ユーザ・タイプを指定します。これで，ブラウザのコントロールが存在するときに VuGen で Web 固有の機能を記録できるようになります。



4 ［**OK**］をクリックしてスクリプトを開きます。

SAPGUI 記録オプションの設定

記録オプションを使って，記録セッションのために SAP 関連の設定を行います。［記録オプション］ダイアログ・ボックスを開くには，［**ツール**］＞［**記録オプション**］を選択するか，［記録開始］ダイアログ・ボックスで［**オプション**］ボタンをクリックします。キーボードのショートカット・キーは CTRL キー + F7 キーです。

次の項目について記録オプションの設定が可能です。

- SAPGUI：一般記録オプション
- SAPGUI：コード生成記録オプション

SAP-Web 仮想ユーザ・タイプを使用してマルチ・プロトコルの仮想ユーザ・スクリプトを記録しようとしている場合は，その他の記録オプションについて第 35 章「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。

SAPGUI：一般記録オプション

これらの記録オプションを使って，記録セッション中の一般的な設定を行います。

［一般］記録オプションを設定するには，次の手順で行います。

1 ［記録オプション］ダイアログ・ボックスを開き，［**SAPGUI：一般**］ノードを選択します。

2 ［**メイン ウィンドウまで自動的に操作する**］を選択して，ログイン後に VuGen がメイン画面に移動するようにします。記録中にログイン画面に情報を入力すると，VuGen は自動的に，メイン・ウィンドウが開くまで，以降のすべての画面で確認の処理を実行します。

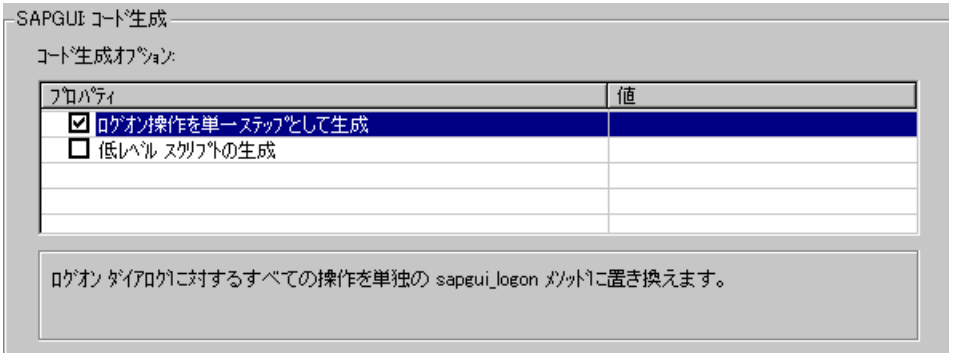
3 ［**開いているセッションで記録の継続を有効にする**］を選択して，前回開いた SAPGUI ウィンドウからイベントを記録できるようにします。

4 記録中に SAPGUI 画面が表示されたときに，画面のスナップショットをキャプチャするには，［**画面のスナップショットをキャプチャする**］を選択します。

5 ［**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。

SAPGUI：コード生成記録オプション

これらの記録オプションを使って，コード生成の設定を行います。



［コード生成］記録オプションを設定するには，次の手順で行います。

1 ［記録オプション］ダイアログ・ボックスを開き，［**SAPGUI：コード生成**］ノードを選択します。

2 ［**ログオン操作を単一ステップとして生成**］を選択し，すべてのログイン操作に対して，単一の **sapgui_logon** メソッドを生成するようにします。これによって，コードが簡略化されます。ログインで問題が生じた場合は，このオプションを無効にします。

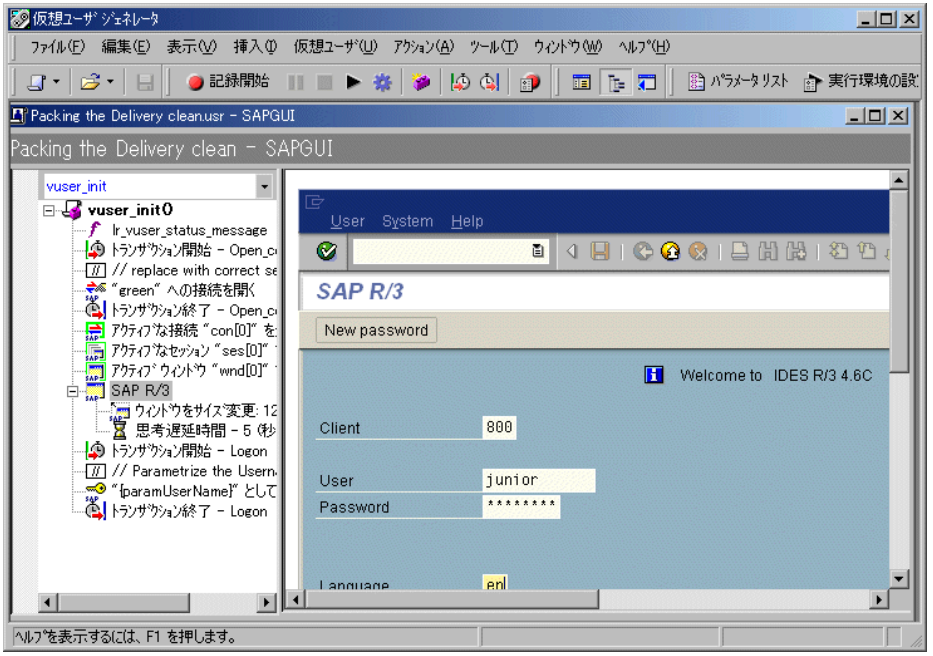
3 SAPGUI スクリプトの読みやすさを向上するために，VuGen では，標準でオブジェクト固有の関数が生成されます。たとえば，SAPGUI グリッド内で記録されたすべての関数には，**sapgui_grid** という接頭辞が付きます。VuGen を使って，**sapgui_set_property** や **sapgui_call_method** などの低レベル関数を含むスクリプトを生成するには，［**低レベル スクリプトの生成**］を選択します。高レベル関数を生成するためにこのオプションを無効にしても，可読性が増すだけです。2 つの記録レベルの間にオーバーヘッドの差はありません。

4 ［**OK**］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。

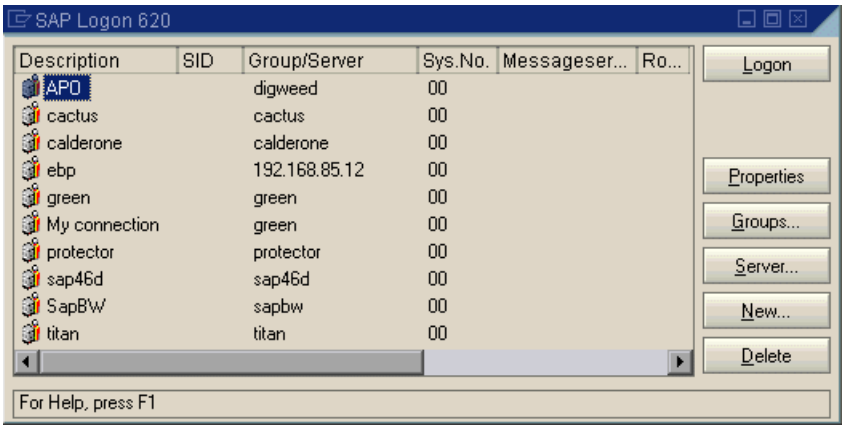
SAPGUI 仮想ユーザ・スクリプトについて

通常 SAPGUI 仮想ユーザ・スクリプトには，ビジネス・プロセスを構成する複数の SAP トランザクションが含まれています。ビジネス・プロセスは，ユーザ・アクションをエミュレートする関数で構成されます。ツリー・ビューを開くと，各ユーザ・アクションが仮想ユーザ・スクリプトのステップとして表示されます。

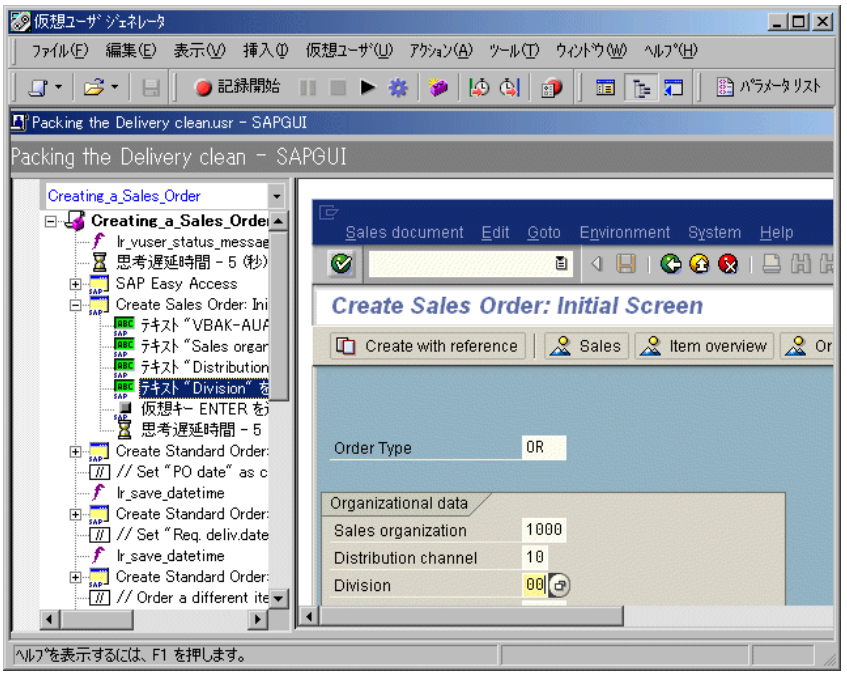
次の例は，SAPGUI クライアントの典型的な記録を示しています。最初のセクションである **vuser_init** には，接続の開始とログインが含まれます。



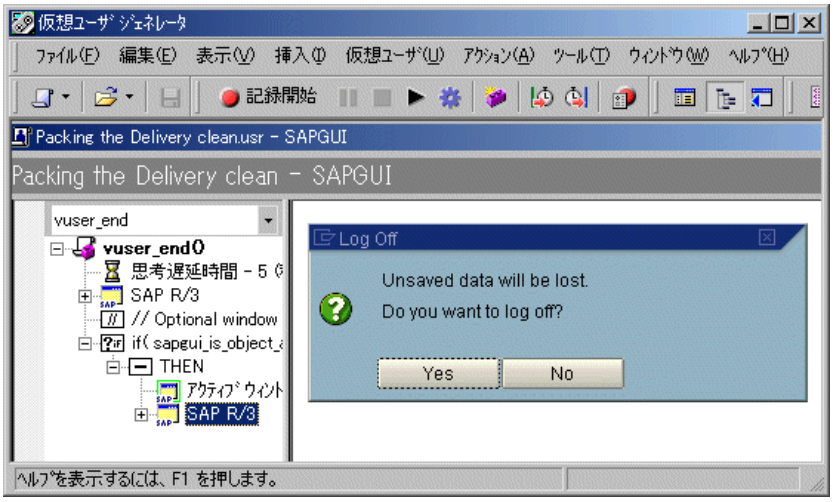
［Open Connection］ステップは，［SAP Logon］の［**Descriptions**］リストにある接続名の 1 つを使用します。指定された名前がリストにない場合，仮想ユーザはその名前のサーバを検索します。



次のセクションでは，関数によって，メニューの選択やチェック・ボックスの設定など，一般的なユーザ操作がエミュレートされます。



最後のセクション **vuser_end** は，ログオフ手順を示しています。



SAPGUI と Web の両方に対してマルチ・プロトコルのスクリプトを記録しているときは，両方のプロトコルに対するステップが VuGen によって生成されます。スクリプト・ビューでは，**sapgui** と **web** の両方の関数を表示できます。次の例は，SAPGUI クライアントによって Web コントロールが開かれるマルチ・プロトコル記録を示しています。**sapgui** 関数から **web** 関数に切り替わることに注意してください。

```
sapgui_tree_double_click_item("Use as general WWW browser,
REPTITLE",
	"shellcont/shell",
	"000732",
	"REPTITLE",
	BEGIN_OPTIONAL,
		"AdditionalInfo=sapgui1020",
	END_OPTIONAL);

...
sapgui_set_text("",
	"http:¥¥¥¥yahoo.com",
	"usr/txtEDURL",
	BEGIN_OPTIONAL,
		"AdditionalInfo=sapgui1021",
	END_OPTIONAL);

...
web_add_cookie("B=7pt5cisv1p3m2&b=2; DOMAIN=www.yahoo.com");

web_url("yahoo.com",
		"URL=http://yahoo.com/",
		"Resource=0",
		"RecContentType=text/html",
		"Referer=",
		"Snapshot=t1.inf",
		"Mode=HTML",
		EXTRARES,
"URL=http://srd.yahoo.com/hpt1/ni=17/ct=lan/sss=1043752588/t1=10437
52575385/d1=1251/d2=1312/d3=1642/d4=4757/0.4097009487287739/*1"
, "Referer=http://www.yahoo.com/", ENDITEM,
		LAST);
```

SAPGUI 仮想ユーザ・スクリプトの拡張

記録された仮想ユーザ・スクリプトを確認し終わったら，次の方法でそれを拡張します。

- **トランザクション**：トランザクション，ランデブー・ポイント，および制御フロー構造を，スクリプトに挿入します。詳細については，第 6 章「仮想ユーザ・スクリプトの拡張」を参照してください。
- **検証**：SAPGUI の検証関数を挿入し，SAPGUI オブジェクトの現在のステータスを検証します。詳細については，「検証関数の追加」を参照してください。
- **情報の取得**：SAPGUI の関数を挿入し，SAPGUI オブジェクトの現在の値を検証します。情報は **sapgui_get_xxx** 関数を使用して取得します。詳細については，686 ページ「情報の取得」を参照してください。
- **パラメータの定義（任意）**：仮想ユーザ・スクリプトに記録された固定値に対してパラメータを定義します。固定値をパラメータに置き換えることによって，異なる値を使って同じビジネス・プロセスを何度も繰り返すことができます。詳細については，第 7 章「パラメータの定義」を参照してください。

検証関数の追加

オプションの，または動的なウィンドウやフレームで作業をしているときに，検証関数を使用して，ウィンドウやオブジェクトが使用可能かどうかを調べることができます。これにより，オプションのウィンドウや例外が発生した場合でも，仮想ユーザ・スクリプトの実行を続けることができます。

最初の例では，ウィンドウが使用可能かどうかを確認しています。ウィンドウが使用可能な場合は，仮想ユーザへ実行を継続する前にそのウィンドウを閉じます。

```
if (!sapgui_is_object_available("wnd[1]"))
        sapgui_call_method("{ButtonID}",
                "press",
                LAST,
                AdditionalInfo=info1011");
sapgui_press_button(.....)
```

次の例は，ME51N トランザクション内の動的なオブジェクトを示しています。［Document overview］フレームはオプションであり，［**Document overview on/off**］ボタンによって開いたり閉じたりできます。

コードでは［Document overview］ボタンのテキストを調べています。ボタンのテキストが **Document overview on** であれば，そのボタンをクリックして［Document overview］フレームを閉じます。

```
        if(sapgui_is_object_available("tbar[1]/btn[9]"))
        {
            sapgui_get_text("Document overview on/off button",
                    "tbar[1]/btn[9]",
                    "paramButtonText",
                    LAST);

            if(0 == strcmp("Document overview off",
    lr_eval_string("{paramButtonText}")))
                    sapgui_press_button("Document overview off",
                        "tbar[1]/btn[9]",
                        BEGIN_OPTIONAL,
                            "AdditionalInfo=sapgui1013",
                        END_OPTIONAL);
        }
```

情報の取得

SAPGUI 仮想ユーザで作業しているときに，**sapgui_get_<xxx>** 関数を使用して SAPGUI オブジェクトの現在の値を取得できます。この値は，別のビジネス・プロセスの入力として使用したり，出力ログに表示したりできます。

ステータス・バー情報の取得

次の例では，ステータス・バー・メッセージの一部を保存して注文番号を取得する方法を示します。

ステータス・バーから注文番号を取得するには，次の手順で行います。

1 ステータス・バー・テキストを確認する位置に移動して，［挿入］＞［新規ステップ］を選択します。**sapgui_status_bar_get_type** 関数を選択します。この関数は，仮想ユーザがステータス・バーからテキストを正常に取得できるかどうかを確かめます。

2 前のステートメントが正常に実行されたかどうかを確かめる **if** ステートメントを挿入します。正常に実行された場合は，**sapgui_status_bar_get_param** を使用して引数の値を保存します。

この **sapgui_status_bar_get_param** 関数は，注文番号をユーザ定義のパラメータに保存します。ここでは，注文番号はステータス・バー文字列の 2 番目のインデックスです。

```
sapgui_press_button("Save (Ctrl+S)",
      "tbar[0]/btn[11]",
      BEGIN_OPTIONAL,
           "AdditionalInfo=sapgui1038",
      END_OPTIONAL);

sapgui_status_bar_get_type("Status");
if(0==strcmp(lr_eval_string("{Status}"),"Success"))
  sapgui_status_bar_get_param("2", " Order_Number ");
```

テストの実行中，Execution ログには次のように値とパラメータ名が示されます。

```
Action.c(240): Pressed button " Save (Ctrl+S)"
Action.c(248): The type of the status bar is "Success"
Action.c(251): The value of parameter 2 in the status bar is "33232"
```

日付情報の保存

日付を使用するスクリプトを作成すると，正しく動作しないことがあります。たとえば，スクリプトを 7 月 2 日に記録し，それを 7 月 3 日に再生した場合，日付フィールドが正しくなりません。そのため，テキスト実行中に日付をパラメータに保存し，保存した値を他の日付フィールドへの入力として使用する必要があります。スクリプト実行中の現在の日付または時刻を保存するには，**lr_save_datetime** 関数を使用します。この関数を，日付情報を必要とする関数の前に挿入します。日付の形式はロケールに固有です。**lr_save_datetime** 関数の中ではロケールに応じた形式を使用します。たとえば，＜月＞.＜日＞.＜年＞の形式にする場合は，「%m.%d.%Y」と指定します。

次の例では，**lr_save_datetime** で現在の日付を保存します。この値を **sapgui_set_text** 関数で使い，2 日後の配送日を設定します。

```
lr_save_datetime("%d.%m.%Y", DATE_NOW + (2 * ONE_DAY),
    "paramDateTodayPlus2");
sapgui_set_text("Req. deliv.date",
        "{paramDateTodayPlus2}","usr/ctxtRV45A-KETDAT",
        BEGIN_OPTIONAL,
            "AdditionalInfo=sapgui1025",
        END_OPTIONAL);
```

SAPGUI 仮想ユーザ・スクリプトの再生

SAPGUI 仮想ユーザ・スクリプトの作成と拡張を終えたら，その実行環境の設定を行い，VuGen から実行してその機能を確かめます。

実行環境を設定することによって，再生時の仮想ユーザの動作を制御します。これらの設定は仮想ユーザ・スクリプトを実行する前に行います。一般の実行環境と SAPGUI 固有の実行環境の両方を設定できます。

この一般設定には，実行論理，ペース設定，ログ，思考遅延時間，パフォーマンスの設定が含まれます。一般の実行環境の設定については，第 9 章「実行環境の設定」を参照してください。SAPGUI 固有の設定については，以降の項を参照してください。

実行環境の設定が完了したら，仮想ユーザ・スクリプトを保存して VuGen から実行し，正しく動作することを確認します。仮想ユーザ・スクリプトをスタンドアロン・テストとして実行する方法の詳細については，第 11 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。仮想ユーザ・スクリプトが機能することを確認したら，それをシナリオに組み込みます。詳細については，『**LoadRunner コントローラ・ユーザーズ・ガイド**』を参照してください。

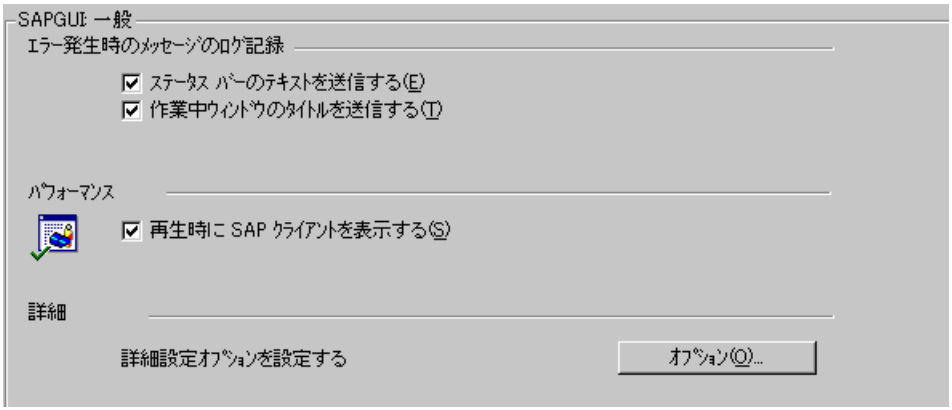
SAPGUI 実行環境の設定

次の項目について SAPGUI 固有の実行環境を設定できます。

- SAPGUI：実行環境の一般設定
- SAPGUI：実行環境の詳細設定

SAPGUI：実行環境の一般設定

実行環境の一般設定では，SAPGUI 仮想ユーザ・スクリプトの一般設定が行えます。VuGen ではこれらの設定がスクリプトの実行時に使用されます。



［SAPGUI：一般］の［エラー発生時のメッセージのログ記録］では，エラーが発生するたびに仮想ユーザが実行ログに送信する情報を指定します。

［**ステータス バー テキストを送信する**］：ステータス・バーからログ・ファイルにテキストを送信します。

［**作業中のウィンドウのタイトルを送信する**］：作業中のウィンドウのタイトル・テキストをログ・ファイルに送信します。

［SAPGUI：一般］の［パフォーマンス］では，再生時に SAP クライアントを表示するかどうかを指定できます。

［**再生時に SAP クライアントを表示する**］：再生中に SAP クライアントにアクションのアニメーションを表示します。ユーザ・インタフェース（UI）を表示させる利点は，フォームにどのように入力が行われているかを確認でき，仮想ユーザのアクションを詳細に追えることです。しかし，このオプションではより多くのリソースが必要になるため，負荷テストのパフォーマンスに影響を与える場合があります。

［詳細］オプションでは，**SAPfewgsvr.exe** プロセスのタイムアウトを設定できます。詳細については，690 ページ「SAPGUI：実行環境の詳細設定」を参照してください。

SAPGUI 用の実行環境を設定するには，次の手順で行います。

1 ［実行環境設定］ダイアログ・ボックスを開きます。VuGen ツールバーの［**実行環境の設定を編集**］ボタンをクリックするか，［**仮想ユーザ**］＞［**実行環境の設定**］を選択します。

2 ［**SAPGUI：一般**］ノードを選択します。

3 ［**エラー発生時のメッセージのログ記録**］セクションで，［**ステータス バー テキストを送信する**］または［**作業中のウィンドウのタイトルを送信する**］のうち 1 つまたは両方のメッセージ・ソースを選択します。

4 再生時に SAP ユーザ・インタフェースを表示するには，［**パフォーマンス**］セクションで［**再生時に SAP クライアントを表示する**］チェック・ボックスを選択します。

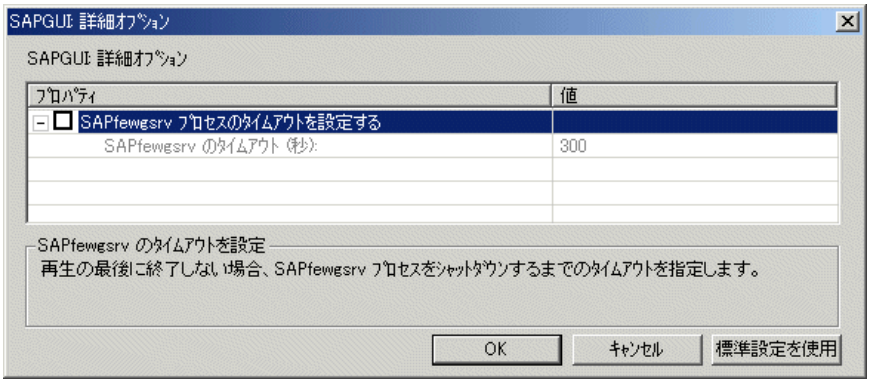
5 ［**オプション**］をクリックし，**SAPfewgsvr.exe** プロセスのタイムアウトを設定します。

SAPGUI：実行環境の詳細設定

各仮想ユーザはテスト実行時に，個別の **SAPfewgsvr.exe** プロセスを呼び出します。場合によっては，再生セッションが終了してもプロセスがオープンのままとなることがあります。プロセスがアクティブかどうかを調べるには，Windows タスク・マネージャを確認します。

［SAPGUI：詳細オプション］では，このアプリケーションのタイムアウトを設定できます。タイムアウトの時間に達した時点で，VuGen はまだ終了していない **SAPfewgsvr** プロセスを終了します。

［SAPGUI：詳細オプション］では，**SAPfewgsvr.exe** プロセスの値を設定できます。



［**SAPfewgsvr プロセスのタイムアウトを設定する**］：**SAPfewgsvr.exe** プロセスのタイムアウトを修正できます。

［**SAPfewgsvr のタイムアウト（秒）：**］：**SAPfewgsvr.exe** プロセスのタイムアウトを秒単位で指定します。標準設定は 300 秒です。

本章で説明したヒントは，シナリオでの記録，再生，および実行に適用されます。また，SAP のサポート・サイトからも直接情報を参照できます。

SAPGUI の関数

SAPGUI の記録セッション中，SAPGUI クライアントでのユーザの作業をエミュレートする関数が生成されます。SAPGUI for Windows クライアントを記録すると，**sapgui** という接頭辞を持つ関数が生成されます。本項ではすべての **sapgui** 関数について説明します。

SAP Workplace または Portal などの Web インタフェースを使用して SAP セッションを記録するとき，または SAPGUI クライアントから Web コントロールを開く場合は，**web** という接頭辞を持つ関数が生成されます。

sapgui および **web** 関数の詳細については，［編集］メニューから［**関数構文の自動表示**］機能を使用するか，または「**オンライン関数リファレンス**」を参照してください（［**ヘルプ**］＞［**関数リファレンス**］）。

ほとんどの関数は記録されますが，任意の関数をスクリプトに手作業で挿入することもできます。**sapgui_get** で始まるデータ取得関数と，**sapgui_is** で始まる検証用の関数は，記録されません。

sapgui 関数には，接続およびセッション関数，メソッドおよびプロパティ関数，検証およびデータ取得関数，およびオブジェクト関数があります。オブジェクト関数は SAPGUI オブジェクトの内部でアクションを実行する関数で，カレンダー関数，グリッド関数，ステータス・バー関数，テーブル関数，ツリー関数，ウィンドウ関数，および一般オブジェクト関数があります。

接続およびセッション関数

sapgui_create_session	新規 SAPGUI セッションを作成します。
sapgui_logon	SAP サーバにログインします。
sapgui_open_connection	SAP サーバへの接続を開きます。
sapgui_open_connection_ex	接続文字列で指定した SAP サーバへの接続を開きます。
sapgui_select_active_connection	指定した接続をアクティブな接続として設定します。
sapgui_select_active_session	アクティブな SAPGUI セッションを設定します。

メソッドおよびプロパティ関数

sapgui_get_property_of_active_object	現在アクティブなオブジェクトのプロパティを取得します。
sapgui_active_object_from_parent_method	親オブジェクトのメソッドを呼び出すことによって，親オブジェクトのオブジェクトを選択します。
sapgui_active_object_from_parent_property	親オブジェクトのプロパティであるオブジェクトを選択します。
sapgui_call_method	SAPGUI オブジェクトのメソッドを呼び出します。
sapgui_call_method_of_active_object	現在アクティブなオブジェクトのメソッドを呼び出します。
sapgui_get_property	SAPGUI オブジェクトのプロパティを取得します。
sapgui_set_collection_property	**SAP GuiCollection** 型のオブジェクトのプロパティを設定します。
sapgui_set_property	SAPGUI オブジェクトのプロパティを設定します。
sapgui_set_property_of_active_object	現在アクティブなオブジェクトのプロパティを設定します。

カレンダー関数

sapgui_calendar_focus_date	特定の日付にフォーカスを設定します。
sapgui_calendar_scroll_to_date	カレンダーの特定の日付までスクロールします。
sapgui_calendar_select_interval	カレンダー内の日付の範囲を選択します。

グリッド関数

sapgui_grid_clear_selection	グリッドの選択をクリアします。
sapgui_grid_click	グリッド内をクリックします。
sapgui_grid_click_current_cell	グリッド内のアクティブなセルをクリックします。
sapgui_grid_double_click	グリッド内をダブルクリックします。
sapgui_grid_double_click_current_cell	グリッド内のアクティブなセルをダブルクリックします。
sapgui_grid_get_cell_data	グリッド・セルのテキストを取得します。
sapgui_grid_get_current_cell_column	グリッド内で，現在のセルのカラムのKEY（Inner ID）を取得します。
sapgui_grid_get_current_cell_row	グリッドの現在のセルの行番号を取得します。
sapgui_grid_is_checkbox_selected	グリッドのチェック・ボックスのステータスを確認します。
sapgui_grid_open_context_menu	グリッドで右クリックし，コンテキスト・メニューを開きます。
sapgui_grid_press_button	グリッド・セルでボタンをクリックします。
sapgui_grid_press_button_current_cell	アクティブなグリッド・セルでボタンをクリックします。
sapgui_grid_press_column_header	グリッドでカラム・ヘッダーを押します。
sapgui_grid_press_ENTER	グリッドで ENTER を押します。
sapgui_grid_press_F1	グリッドで F1 を押します。
sapgui_grid_press_F4	グリッドで F4 を押します。
sapgui_grid_press_toolbar_button	グリッドでツールバー・ボタンをクリックします。

sapgui_grid_press_toolbar_context_button	グリッドでツールバー・コンテキスト・ボタンをクリックします。
sapgui_grid_press_total_row	グリッドで合計行の領域をクリックします。
sapgui_grid_press_total_row_current_cell	現在アクティブなグリッド・セルで合計行のボタンをクリックします。
sapgui_grid_scroll_to_row	グリッドである行までスクロールします。
sapgui_grid_select_cell	グリッドでセルを選択します。
sapgui_grid_select_cell_column	現在の行の指定したカラムのセルを選択します。
sapgui_grid_select_cell_row	現在のカラムの指定した行のセルを選択します。
sapgui_grid_select_cells	グリッドでセルを選択します。
sapgui_grid_select_columns	グリッドでカラムを選択します。
sapgui_grid_select_context_menu	グリッドでコンテキスト・メニューを選択します。
sapgui_grid_select_rows	グリッドで行を選択します。
sapgui_grid_select_toolbar_menu	グリッドでツールバー・メニューを選択します。
sapgui_grid_set_cell_data	グリッド・セルにテキストを挿入します。
sapgui_grid_set_checkbox	グリッドのチェック・ボックスを選択またはクリアします。
sapgui_grid_set_column_order	グリッドのカラム順序を設定します。

ステータス・バー関数

sapgui_status_bar_get_param	ステータス・バーからパラメータを取得します。
sapgui_status_bar_get_text	ステータス・バーからテキストを取得します。
sapgui_status_bar_get_type	ステータス・バー情報（「**成功**」，「**警告**」，または「**エラー**」）を取得します。

テーブル関数

sapgui_table_is_checkbox_selected	テーブルのチェック・ボックスのステータスを確認します。
sapgui_table_is_row_selected	テーブル行が選択されているかどうかを確認します。
sapgui_table_get_text	テーブル行のテキストを取得します。
sapgui_table_is_radio_button_selected	テーブルのラジオ・ボタンのステータスを確認します。
sapgui_table_press_button	テーブルでボタンを押します。
sapgui_table_select_combobox_entry	テーブルのリスト・エントリを選択します。
sapgui_table_select_radio_button	テーブルでラジオ・ボタンを選択します。
sapgui_table_set_checkbox	テーブルのチェック・ボックスを選択またはクリアします。
sapgui_table_set_focus	テーブルにフォーカスを設定します。
sapgui_table_set_password	テーブル内にパスワードを設定します。
sapgui_table_set_row_selected	テーブルの行を選択または選択を解除します。
sapgui_table_set_text	テーブル・セルにテキストを挿入します。

ツリー関数

sapgui_tree_click_link	ツリーのリンクをクリックします。
sapgui_tree_collapse_node	ツリー・ノードを折りたたみます。
sapgui_tree_double_click_item	ツリー項目をダブルクリックします。
sapgui_tree_double_click_node	ツリー・ノードをダブルクリックします。
sapgui_tree_expand_node	ツリー・ノードを展開します。
sapgui_tree_get_item_text	ツリー項目のテキストを取得します。
sapgui_tree_get_node_text	ツリー・ノードのテキストを取得します。
sapgui_tree_is_checkbox_selected	ツリーのチェック・ボックスが選択されているかどうかを確認します。
sapgui_tree_open_default_context_menu	ツリーの標準のショートカット・メニューを開きます。
sapgui_tree_open_header_context_menu	ツリー・ヘッダーのショートカット・メニューを開きます。
sapgui_tree_open_item_context_menu	ツリー項目のショートカット・メニューを開きます。
sapgui_tree_open_node_context_menu	ツリー・ノードのショートカット・メニューを開きます。
sapgui_tree_press_button	ツリーでボタンをクリックします。
sapgui_tree_press_header	ツリーでカラム・ヘッダーをクリックします。
sapgui_tree_press_key	ツリー内からキーを押します。
sapgui_tree_scroll_to_item	ツリー項目までスクロールします。
sapgui_tree_scroll_to_node	ツリー・ノードまでスクロールします。
sapgui_tree_select_column	ツリーのカラムを選択します。
sapgui_tree_select_item	ツリーの項目を選択します。
sapgui_tree_select_node	ツリーのノードを選択します。

sapgui_tree_set_checkbox	ツリーのチェック・ボックスを選択またはクリアします。
sapgui_tree_set_column_width	ツリーのカラム幅を設定します。
sapgui_tree_set_hierarchy_header_width	ツリー階層の幅を設定します。
sapgui_tree_set_selected_node	ツリーのノードを選択します。
sapgui_tree_unselect_all	ツリーの選択をすべて解除します。
sapgui_tree_unselect_column	ツリー・カラムの選択を解除します。
sapgui_tree_unselect_node	ツリー・ノードの選択を解除します。

ウィンドウ関数

sapgui_window_close	SAPGUI クライアント・ウィンドウを閉じます。
sapgui_window_maximize	ウィンドウをフルスクリーン・サイズに設定します。
sapgui_window_resize	ウィンドウを指定したサイズに合わせます。
sapgui_window_restore	ウィンドウを最大化されていない状態に戻します。
sapgui_window_scroll_to_row	ウィンドウのある行までスクロールします。

検証およびデータ取得関数

sapgui_get_active_window_title	アクティブなウィンドウのタイトルを取得します。
sapgui_get_ok_code	［Command］フィールドのテキストを取得します。
sapgui_get_text	オブジェクトからテキストを取得します。
sapgui_is_checkbox_selected	チェック・ボックスが選択されているかどうかを確認します。
sapgui_is_object_available	オブジェクトが利用可能かどうかを確認します。
sapgui_is_object_changeable	変更可能なオブジェクトかどうかを確認します。
sapgui_is_radio_button_selected	ラジオ・ボタンが選択されているかどうかを確認します。
sapgui_is_tab_selected	タブが選択されているかどうかを確認します。

一般オブジェクト関数

sapgui_htmlviewer_send_event	HTML ビューアにイベントを送信します。
sapgui_select_combobox_entry	リスト・エントリを選択します。
sapgui_press_button	ボタンを押します。
sapgui_select_active_window	指定したウィンドウをアクティブ・ウィンドウとして設定します。
sapgui_select_radio_button	ラジオ・ボタンを選択します。
sapgui_select_tab	タブを選択します。
sapgui_send_vkey	仮想キーを送信します。
sapgui_set_checkbox	チェック・ボックスを選択またはクリアします。

sapgui_set_focus	指定したオブジェクトにフォーカスを設定します。
sapgui_select_menu	指定したメニューを選択します。
sapgui_set_password	パスワード・フィールドのテキストを設定します。
sapgui_set_text	テキスト・ボックスにテキストを挿入します。
sapgui_set_ok_code	［Command］フィールドのテキストを設定します。

SAPGUI 仮想ユーザ・スクリプトに関するヒント

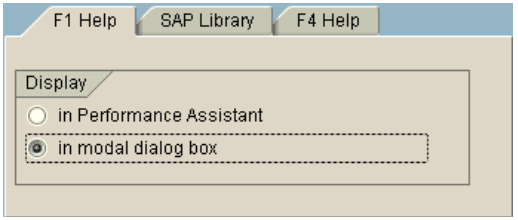
以下の各項では，SAPGUI 仮想ユーザの記録に関するヒント，再生に関するヒント，およびシナリオ内での再生に関するヒント について説明します。

記録に関するヒント

本項では，SAPGUI 仮想ユーザ・スクリプトの記録に関するヒントについて説明します。

- アクションを適切なセクションに記録するようにします。ログイン手順は **vuser_init** セクションに，反復実行するアクションは **Actions** セクションに，ログオフ手順は **vuser_end** セクションに記録します。
- SAPGUI クライアントに Web コントロールが含まれているマルチ・プロトコル・スクリプトを記録する場合は，記録を開始する前に **SAPLogon** アプリケーションを終了します。

➤ F1 に対応したモーダル・ダイアログ・ボックスを使用します。F1 ヘルプをモード・ダイアログ・ボックスで開くように SAPGUI に指定します。［**Help**］＞［**Settings**］を選択します。［**F1 Help**］タブをクリックし，［Display］セクションの［**in modal dialog box**］オプションを選択します。

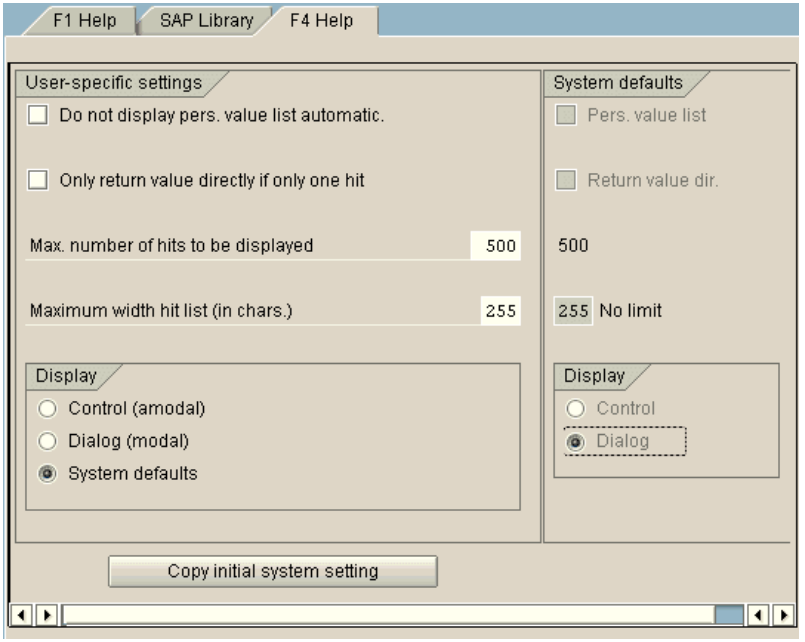


➤ F4 に対応したモーダル・ダイアログ・ボックスを使用します。F4 ヘルプをモード・ダイアログ・ボックスで開くように SAPGUI に指定します。SAP 管理者は次の手順を行う必要があります。

F4 ヘルプをモーダル・ダイアログ・ボックスで開くには，次の手順で行います。

1 サーバからすべてのユーザがログオフしていることを確認してください。

2 ［**Help**］＞［**Settings**］を選択します。［**F4 Help**］タブをクリックします。



3［Display］セクション（左下）で，［System defaults］を選択します。

4［System defaults］セクションの［Display］部分（右下）で，［**Dialog**］を選択します。

5 変更を保存します。［**Copy initial system setting**］をクリックするか，CTRL キーを押しながら S キーを同時に押します。

6 ステータス・バーに「**Data was saved**」というメッセージが表示されているかどうか確認します。

7 セッションを終了します。

8 SAP Management Console を使って，サービスを再起動します。

再生に関するヒント

スクリプトをスタンドアロン・モードで再生する前に，このガイドラインを読んでください。

➤ 記録時に生成された **sapgui_logon** 関数の暗号化されたパスワードを，実際のパスワードに置き換えます。次に示すような関数の 2 つ目の引数（ユーザ名の次）がパスワードです：sapgui_logon("user", "pswd", "800", "EN"); セキュリティ向上のため，コード内のパスワードは暗号化できるようになっています。パスワード・テキスト（***** ではなく実際のテキスト）を選び，右クリック・メニューで［**文字列を暗号化**］を選択します。**lr_decrypt** 関数（sapgui_logon("user", lr_decrypt("3ea037b758"), "800", "EN");）がパスワードの位置に挿入されます。

➤ 初めてスクリプトを実行する場合は，再生時に SAPGUI ユーザ・インタフェースを表示するように VuGen を設定し，その UI を使って実行されている操作を確認できるようにします。再生中にユーザ・インタフェースを表示するには，実行環境の設定を開き，［**SAPGUI：一般**］ノードで［**再生時に SAP クライアントを表示する**］オプションを選択します。複数の仮想ユーザを実行すると UI の表示に多量のシステム・リソースが使用されるので，負荷シナリオの実行中はこのオプションを無効にします。

シナリオ内での再生に関するヒント

以下の各項では，コントローラやロード・ジェネレータ・マシンでスクリプトを実行する際の設定に関するヒントについて説明します。

コントローラの設定

シナリオ内で LoadRunner コントローラとともにスクリプトを実行するときは，以下の値の設定を負荷シナリオ設定で行います。

ランプアップ：(適切なログオンを保証するために) 1 つずつスケジューラで設定します。

思考遅延時間：実行環境に乱数の思考遅延時間を設定します。

ロード・ジェネレータごとのユーザ数：512 MB のメモリを搭載したマシンでは，［ロード・ジェネレータ］ダイアログ・ボックスで 50 仮想ユーザを設定します。

SAP モニタは SAP サーバ・バージョン 4.6 以前でのみサポートされています。

ロード・ジェネレータの設定

スクリプトをシナリオで実行するときは，ロード・ジェネレータ・マシンでエージェント・モードを確認し，ターミナル・セッションを設定します。

エージェントのモード：プロセス・モードで LoadRunner Remote Agent が実行されていることを確認します。サービス・モードはサポートされていません。

これを調べるには，Windows のタスクバーにあるエージェントのアイコン上にマウスを移動し，説明を読みます。「LoadRunner Agent Service」と説明に表示された場合は，サービスとしてエージェントが実行されています。

プロセスとしてエージェントを再起動するには，次の手順で行います。

1 エージェントを停止します。LoadRunner Agent のアイコンを右クリックし，［**Close**］を選択します。

2 **magentproc.exe** を実行します。これは，LoadRunner インストール先の **launch_service¥bin** ディレクトリにあります。

3 次回マシンを起動したときに正しいエージェントが起動されるようにするには，Agent Service の開始方法を［自動］から［手動］に変更します。**magentproc.exe** へのショートカットを Windows の［スタートアップ］フォルダに追加します。

Terminal Sessions：SAPGUI 仮想ユーザを実行するマシンは，使用できるグラフィック・リソースによっては，実行できる仮想ユーザの数が限られる可能性があります。各マシンの仮想ユーザ数を増やすには，ロード・ジェネレータ・マシンで追加の Terminal Server セッションを開始します。［**スタート**］＞［**プログラム**］＞［**LoadRunner**］＞［**Advanced Settings**］から［**Agent Configuration**］を選択し，［Enable Terminal Service］オプションを選択します。ロード・ジェネレータ・マシンのプロパティで，ターミナル・セッションの数を指定します。詳細については，『**LoadRunner コントローラ・ユーザーズ・ガイド**』の「ターミナル・サービスの設定」を参照してください。

SAPGUI 仮想ユーザ・スクリプトのトラブルシューティング

質問 1：スクリプトは記録できました。しかし再生できません。なぜでしょうか？

回答：［Process］モードで LoadRunner Remote Agent が実行されていることを確認します。［Service］モードはサポートされていません。詳細については，702 ページ「再生に関するヒント」を参照してください。

質問 2：特定の SAPGUI コントロールが記録されないのはなぜですか？

回答：一部の SAPGUI コントロールはそのメニューまたはツールバーのコンテキストの中でのみサポートされます。問題のあるタスクについて，メニュー・オプションやコンテキスト・メニュー，ツールバーなどのさまざまな手段を使用して実行を試みます。

質問 3：VuGen でスクリプトの記録や再生ができないのはなぜですか？

回答：

1 SAPGUI 6.20 の最新のパッチがインストールされていることを確認します。最低でもパッチ・レベル 32 である必要があります。

2 スクリプティングが有効になっていることを確認します。669 ページ「設定の確認」を参照してください。

3 SAPGUI for Windows クライアントで通知が無効にされていることを確認します。［Customizing of Local Layout］ボタンをクリックするか，ALT+F12 キーを押します。［**Options**］をクリックして［Scripting］タブを選択します。両方の［**Notify**］オプションをクリアします。

質問 4 : スクリプトを実行しようとしたときに表示されるエラー・ポップアップ・メッセージはどのような意味ですか？

回答 : SAP アプリケーションの中にはユーザごとに直前のレイアウトを格納するものがあります（どのフレームが表示か非表示か，など）。スクリプトの記録後に，格納されているレイアウトが変更された場合，再生に問題が生じることがあります。たとえば，ME52N トランザクションでは，「Document overview Off/On」ボタンによって，表示されるフレームの数が変わります。

この問題が発生した場合は，次のようにします。

1 再生を開始する前に，トランザクションの記録中と同じ位置に移動します。スナップショット・ビューアを使用すると，トランザクションが記録されたレイアウトを表示できます。

2 スクリプトにステートメントを追加して，再生中にトランザクションが望みのレイアウトになるようにします。たとえば，オプションのフレームによって再生が妨げられる場合は，フレームが開いているかどうかを確かめる検証関数を挿入します。フレームが開いている場合は，ボタンをクリックして閉じます。検証の例については，685 ページ「検証関数の追加」を参照してください。

質問 5 : スクリプトをリモート・マシンで実行するときにシングル・サインオンのメカニズムを使用できますか？

回答 : できません。VuGen ではシングル・サインオンの接続メカニズムはサポートされていません。SAPGUI クライアントで，［Advanced Options］を開き，［**Enable Secure Network Communication**］の機能をクリアします。接続の設定を変更した後，スクリプトを記録し直す必要があります。

質問 6 : VuGen ですべての SAP オブジェクトを記録できますか？

回答 : SAPGUI スクリプティングでサポートされていないオブジェクトについては，記録はできません。これらのオブジェクト型の詳細については記録ログを参照してください。

質問 7 : すべてのビジネス・プロセスがサポートされていますか？

回答 : VuGen では，APO や Microsoft Office のモジュール・コントロールを起動するビジネス・プロセス，あるいは GuiXT の使用を必要とするビジネス・プロセスはサポートされていません。**GuiXT** は SAPGUI for Windows クライアントの［Options］メニューから無効にできます。

その他の参考資料

LoadRunner

ダイアログ・ボックスに関するオンライン・ヘルプを参照するには，ダイアログ ボックスの中で F1 キーを押します。［**ヘルプ**］＞［**目次と索引**］を選択してヘルプを手作業で開くこともできます。［目次］タブで，「**SAPGUI 仮想ユーザ・スクリプト**」という項目を探し，該当するサブ項目をクリックします。

関数に関するオンライン・ヘルプを参照するには，関数またはステップの中をクリックし，F1 キーを押して「**オンライン関数リファレンス**」を開きます。

SAP

詳細については，以下の資料を参照してください。

- **SAP の Web サイト** - www.sap.com
- **SAP に関する注意事項** - https://websmp103.sap-ag.de/notes

  Note #480149: New profile parameter for user scripting on the front end（フロント・エンドのユーザ・スクリプティングのための新しいプロファイル・パラメータ）

  Note #587202: Drag & Drop is a known limitation of the SAPGUI interface（ドラッグアンドドロップは，SAPGUI インタフェースの既知の制限）

- **SAP のパッチ** - https://websmp104.sap-ag.de/patches

  SAP GUI for Windows - SAPGUI 6.20 パッチ（パッチ・レベル 32 以上）

第 49 章

SAP-Web 仮想ユーザ・スクリプトの作成

VuGen の SAP-Web 仮想ユーザを使用して，SAP Workplace および SAP Portal クライアントでの作業を記録することができます。

本章では，次の項目について説明します。

- SAP-Web 仮想ユーザ・スクリプトの作成
- SAP-Web 記録オプションの設定
- SAP-Web 仮想ユーザ・スクリプトについて
- SAP-Web 仮想ユーザ・スクリプトの再生

以降の情報は，SAP-Web プロトコルにのみ適用されます。

SAP-Web 仮想ユーザ・スクリプトの作成について

まず，VuGen で典型的な SAP ビジネス・プロセスを記録します。VuGen では，ビジネス・プロセス中の SAP Workpalce または SAP Portal のアクティビティを記録し，仮想ユーザ・スクリプトを生成できます。ブラウザ内でアクションを実行すると，このアクティビティを説明する関数が生成されます。各関数の先頭には，**web** という接頭辞が付きます。再生中，この関数は SAP Workplace または SAP Portal クライアントでのユーザ・アクティビティをエミュレートします。たとえば，次の **web_url** では PageBuilder 開いています。

```
web_url("PageBuilder[myPage]",
"URL=http://sonata.mercury.co.il/hrnp$30001/sonata.mercury.co.il:80/Acti
on/PageBuilder[myPage]?pageName=com.sapportals.pct.home.mynews",
        "Resource=0",
        "RecContentType=text/html",
        "Referer=http://sonata.mercury.co.il/sapportal",
        "Snapshot=t2.inf",
        "Mode=HTML",
        EXTRARES,
        "Url=/irj/services/laf/themes/portal/sap_mango_polarwind/..,
ENDITEM,
        LAST);
```

SAP-Web 仮想ユーザ・スクリプトの作成

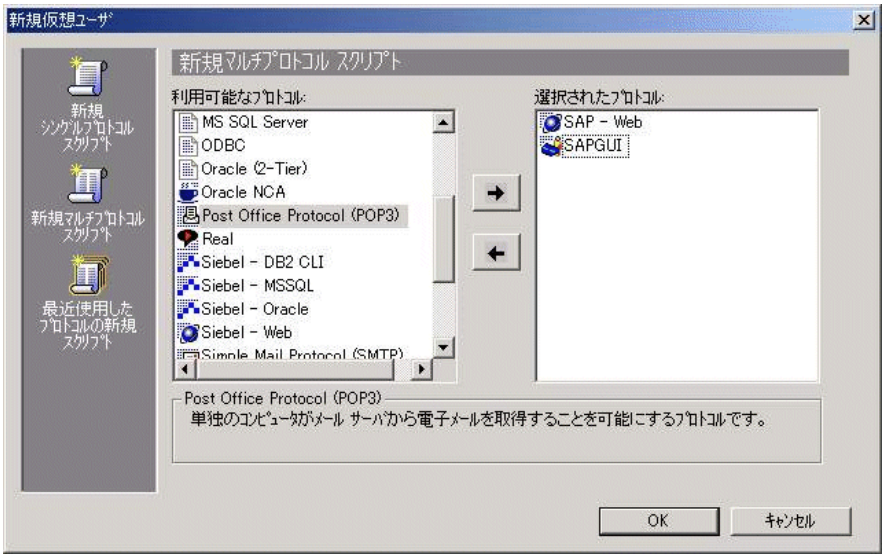
SAP-Web 仮想ユーザ・スクリプト作成の第一歩は，仮想ユーザとスクリプトのタイプを選択することです。**SAP-Web** 仮想ユーザは，［**ERP/CRM**］カテゴリの下にあります。シングル・プロトコルまたはマルチ・プロトコルの仮想ユーザ・スクリプトを作成できます。

SAP-Web 仮想ユーザを作成するは，次の手順で行います。

1 ［**ファイル**］＞［**新規作成**］を選択します。

2 SAPGUI コントロールが含まれない SAP Workplace または SAP Portal クライアントでのセッションを記録するには，**SAP-Web** 仮想ユーザを使ってシングル・プロトコル仮想ユーザ・スクリプトを作成します。

3 SAPGUI コントロールを使用するセッションを記録するには，マルチ・プロトコル仮想ユーザ・スクリプトを作成します。そして **SAP-Web** と **SAPGUI** 仮想ユーザの両方を指定します。



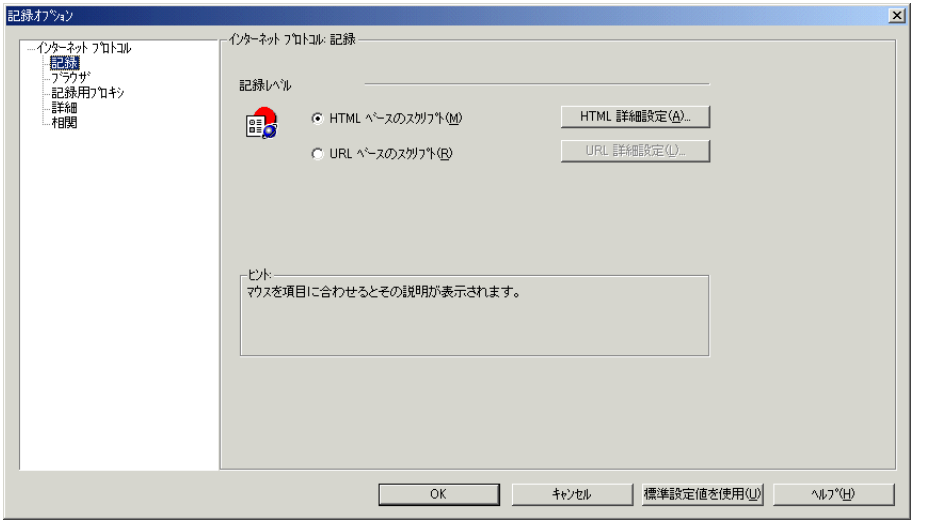
SAP-Web 記録オプションの設定

記録オプションを使用して，VuGen による仮想ユーザ・スクリプトの生成方法を設定します。

［**インターネット プロトコル ：記録**］ノードの推奨設定は次のとおりです。

SAP Workplace の記録の場合：［URL ベースのスクリプト］

SAP Portal の記録の場合：［HTML ベースのスクリプト］（標準設定）



Web に関連する記録オプションの詳細については，第 35 章「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。

SAP-Web 仮想ユーザ・スクリプトについて

通常 SAP-Web 仮想ユーザ・スクリプトには，ビジネス・プロセスを構成する複数の SAP トランザクションが含まれています。ビジネス・プロセスは，ユーザのアクションをエミュレートする関数で構成されます。これらの関数の詳細については，「**オンライン関数リファレンス**」（[**ヘルプ**] ＞ [**関数リファレンス**]）の「Web 関数」を参照してください。

SAP Portal クライアントにおける典型的な記録の例を以下に示します。

```
vuser_init()
{
	web_reg_find("Text=SAP Portals Enterprise Portal 5.0",
		LAST);

	web_set_user("junior{UserNumber}",
		lr_decrypt("3ed4cfe457afe04e"),
		"sonata.mercury.co.il:80");

	web_url("sapportal",
		"URL=http://sonata.mercury.co.il/sapportal",
		"Resource=0",
		"RecContentType=text/html",
		"Snapshot=t1.inf",
		"Mode=HTML",
		EXTRARES,

"Url=/SAPPortal/IE/Media/sap_mango_polarwind/images/header/branding
_image.jpg",
"Referer=http://sonata.mercury.co.il/hrnp$30001/sonata.mercury.co.il:80/A
ction/26011[header]", ENDITEM,

"Url=/SAPPortal/IE/Media/sap_mango_polarwind/images/header/logo.gif",
"Referer=http://sonata.mercury.co.il/hrnp$30001/sonata.mercury.co.il:80/A
ction/26011[header]", ENDITEM,
...
		LAST);
```

このセクションは，SAP Portal クライアントが Web コントロールを開くマルチ・プロトコル記録を示します。**web_xxx** 関数から **sapgui_xxx** 関数に切り替わっている点に注目してください。

```
web_url("dummy",

"URL=http://sonata.mercury.co.il:1000/hrnp$30000/sonata.mercury.co.il:1
000/Action/dummy?PASS_PARAMS=YES&dummyComp=dummy&Tcode
=VA01&draggable=0&CompFName=VA01&Style=sap_mango_polarwind"
,
        "Resource=0",
        "RecContentType=text/html",
        "Referer=http://sonata.mercury.co.il/sapportal",
        "Snapshot=t9.inf",
        "Mode=HTML",
        LAST);

    sapgui_open_connection_ex(" /H/Protector/S/3200 /WP",
        "",
        "con[0]");

    sapgui_select_active_connection("con[0]");

    sapgui_select_active_session("ses[0]");

    /* スクリプト実行時に，ログオン関数でアスタリスクの代わりにパ
スワードを入力 */

    sapgui_logon("JUNIOR{UserNumber}",
        "ides",
        "800",
        "EN",
        BEGIN_OPTIONAL,
            "AdditionalInfo=sapgui102",
        END_OPTIONAL);
```

SAP-Web 仮想ユーザ・スクリプトの再生

SAP-Web 仮想ユーザ・スクリプトを記録した後で，そのスクリプトの実行環境を設定し，VuGen で実行して動作を確認します。

実行環境を設定することによって，再生時の仮想ユーザの動作を制御します。実行環境の設定は，仮想ユーザ・スクリプトの実行前に行います。実行環境の一般設定と Web 関連の設定が可能です。

一般設定には，実行論理，ペース設定，ログ，思考遅延時間，パフォーマンスの設定が含まれます。実行環境の一般設定の詳細については，第 9 章「実行環境の設定」を参照してください。SAP-Web 固有の実行環境の設定については，第 37 章「インターネット実行環境の設定」を参照してください。

実行環境を設定したら，仮想ユーザ・スクリプトを保存し，VuGen でスタンドアロンのテストとして実行し，正しく動作することを確認します。詳細については，第 11 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。

仮想ユーザ・スクリプトが正しく動作していることを確認できたら，シナリオに組み込みます。詳細については，『**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 50 章

Siebel-Web 仮想ユーザ・スクリプトの作成

VuGen を使って，Siebel Web 環境のアクティビティを記録し，仮想ユーザ・スクリプトを生成できます。スクリプトを実行すると，仮想ユーザによって Siebel 環境内のアクションがエミュレートされます。

本章では，次の項目について説明します。

- Siebel-Web セッションの記録
- Siebel-Web スクリプトの相関
- Siebel-Web 仮想ユーザ・スクリプトのトラブルシューティング

以降の情報は，Siebel-Web 仮想ユーザ・スクリプトを対象とします。

Siebel-Web 仮想ユーザ・スクリプトの作成について

Siebel-Web プロトコルは標準の Web 仮想ユーザに似ていますが，Siebel CRM（Customer Relationship Management）アプリケーションを扱えるように，標準設定にいくつかの変更が加えられています。

Siebel セッションの一般的なアクティビティを記録します。VuGen はアクションを記録し，アクションをエミュレートする関数（**web_** という接頭辞が付きます）を生成します。

以降の各項では，記録された Siebel-Web 仮想ユーザ・スクリプト で作業を行う際のヒントについて説明し，相関に必要なパラメータの例を示します。

Siebel-Web セッションの記録

Siebel-Web セッションを記録するときは，以下のガイドラインに従います。

Siebel-Web 仮想ユーザ・スクリプト の記録は，次の手順で行います。

1 Siebel-Web タイプの仮想ユーザ・スクリプトを ERP カテゴリから作成します。

2 以下の記録オプションを設定します。

➤［**記録**］ノード：HTML ベース・スクリプト

［**HTML 詳細設定**］＞［**スクリプト タイプ**］：明示的な URL のみを含むスクリプト

［**HTML 詳細設定**］＞［**生成された HTML 以外の要素**］：記録しない

➤［**詳細**］ノード：［**各アクションごとにコンテキストをリセットする**］オプションをクリア

3 **vuser_init** セクションにログイン手続きを記録します。

4 ビジネス・プロセスを **Action1** に記録します。

5 **vuser_end** セクションにログアウト手続きを記録します。

6 実行環境の設定で，ブラウザ・エミュレーション・ノードの［**反復ごとに新規ユーザをシミュレートする**］オプションをクリアします。

記録オプションと実行環境の設定方法の詳細については，Chapter 第 35 章 , “ インターネット・プロトコルの記録オプションの設定 ” および第 37 章「インターネット実行環境の設定」を参照してください。

Siebel-Web スクリプトの相関

Siebel-Web セッションでは，このプロトコルのための組み込み相関ルールを使用して，変数のほとんどを相関させることができます。詳細については，第 41 章「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。

本項では，以下について説明します。

➤ コールバック

➤ SWECount の相関

➤ 行 ID 長の相関

➤ タイム・スタンプの扱い（SWETS）

コールバック

VuGen で境界を基準にして検索を行った結果，一致する候補が検出されると，パラメータ名とその値へのコールバックが実行されます。コールバックによって，配列が解析され，その引数が個別のパラメータに保存されます。コールバックは，Siebel の相関ルールで定義されています。

このルールよって追加された **web_reg_save_param** 関数群には，追加のパラメータがあります。

AutoCorrelationFunction= ＜コールバック名＞

コールバックは **web_reg_save_param** に対する呼び出しのたびにいくつかのパラメータを保存します。これらのコールバックによって保存されるパラメータの形式は，次のとおりです。

＜コールバック名＞ _1 = arg1
＜コールバック名＞ _2 = arg2
...
＜コールバック名＞ _rowid = ＜ rowid ＞

```
web_reg_save_param("Siebel_Star_Array118",
        "LB/IC=`ValueArray`",
        "RB/IC=`",
        "Ord=2",
        "Search=Body",
        "RelFrameId=1",
        "AutoCorrelationFunction=flCorrelationCallbackParseStarArray",
        LAST);
```

これらのパラメータは，以降の **web_submit_data** 関数で置換できます。

SWECount の相関

SWECount パラメータの値は，通常 1 桁または 2 桁の小さい数値です。記録されたどの値をパラメータで置換するべきか決めるのが困難なことがしばしばあります。

web_submit_data 関数では，VuGen は SWEC フィールドの数値だけを置換します。

URL では，文字列 "SWEC=" または "SWEC`" の後に現れる場合に限り，この数値を置換します。

すべての SWECount 相関のパラメータ名は同じです。

行 ID 長の相関

場合によっては，**rowid** の前に，その長さ 16 進形式でエンコードされて置かれます。この長さは変更される可能性があるため，その値を相関させる必要があります。

たとえば，xxx6_1-4ABCyyy という文字列は長さの値と RowID で構成されています。ここで 6 は長さ，1-4ABC は RowID です。

文字列を相関させるパラメータを，詳細相関を使用して

```
xxx{rowid_Length}_{rowid}yyy
```

と定義した場合，VuGen は文字列の前に次の関数を生成します。

```
web_save_param_length("rowid", LAST);
```

この関数は rowid の値を取得し，その長さをパラメータ **rowid_length** に 16 進形式で保存します。

タイム・スタンプの扱い（SWETS）

スクリプト内の SWETS の値は，1970 年 1 月 1 日午前 0 時を基点として経過したミリ秒数です。

VuGen は，スクリプト内にある空でないすべてのタイム・スタンプを，パラメータ {SiebelTimeStamp} に置き換えます。このパラメータに値を保存する前に，VuGen は次の関数を生成します。

```
web_save_timestamp_param("SiebelTimeStamp", LAST);
```

この関数によって，現在のタイム・スタンプが **SiebelTimeStamp** パラメータに保存されます。

Siebel-Web 仮想ユーザ・スクリプトのトラブルシューティング

本項では，Siebel-Web 仮想ユーザ・スクリプトを作成するときに発生する可能性のある典型的なエラーについて説明します。

- 「戻る」または「更新」のエラー
- 同一の値
- 「No Content」HTTP 応答
- コンテキストの復元
- レコードの検索不能
- ファイルの終わり
- 検索カテゴリの取得不能

「戻る」または「更新」のエラー

「戻る」または「更新」に関連するエラー・メッセージでは，通常は次のようなテキストが表示されます。

We are unable to process your request. This is most likely because you used the browser BACK or REFRESH button to get to this point.

原因：以下の原因が考えられます。

- SWEC が現在の要求と正しく相関されていなかった。
- SWETS が現在の要求と正しく相関されていなかった。
- SWEC が更新されないまま，要求が 2 回 Siebel サーバに送信された。
- 前の要求によってブラウザがダウンロードを行うためのフレームが開かれている必要があった。しかし，おそらくは記録後に SWEMethod が変更されたために，このフレームがサーバに作成されていなかった。

同一の値

同一の値のエラーに対しては，通常は次のような Web ページの応答が表示されます。

@0`0`3`3``0`UC`1`Status`Error`SWEC`10`0`1`Errors`0`2`0`Level0`0`ErrMsg`The same values for 'Name' already exist. If you would like to enter a new record, please ensure that the field values are unique.`ErrCode`28591`

原因：要求内の値の 1 つ（上の例では Name フィールドの値）がデータベース・テーブルの別の行の値と重複していることが考えられます。この値は，ユーザごとに繰り返し使用するたびに，一意な値に置き換える必要があります。解決方法として，行 ID が必ず一意となるようにパラメータに置き換えることをお勧めします。

「No Content」HTTP 応答

「**No Content」HTTP 応答**タイプのエラーでは，通常は次のような HTTP 応答があります。

```
HTTP/1.1 204 No Content
Server: Microsoft-IIS/5.0
Date: Fri, 31 Jan 2003 21:52:30 GMT
Content-Language: en
Cache-Control: no-cache
```

原因：行 ID がまったく相関されていないか，または正しく相関されていないことが考えられます。

コンテキストの復元

コンテキストの復元タイプのエラーでは，通常は次のような Web ページ応答があります。

```
@0`0`3`3``0`UC`1`Status`Error`SWEC`9`0`1`Errors`0`2`0`Level0`0`ErrMsg`An error happened during restoring the context for requested location`ErrCode`27631`
```

原因：行 ID が相関されていないか，または相関が正しくないことが考えられます。

レコードの検索不能

レコードの検索不能タイプのエラーでは，通常は次のような Web ページ応答があります。

```
@0`0`3`3``0`UC`1`Status`Error`SWEC`23`0`2`Errors`0`2`0`Level0`0`ErrMsg`Cannot locate record within view: Contact Detail - Opportunities View applet: Opportunity List Applet.`ErrCode`27573`
```

原因：Web ページにおいてレコードに対応する行 ID が入力名 SWERowId に含まれていないことが考えられます。入力名はパラメータ化しておく必要があります。パラメータのソース値でその場所が変更されている可能性があります。

ファイルの終わり

ファイルの終わりタイプのエラーでは，通常は次のような Web ページ応答があります。

@0`0`3`3``0`UC`1`Status`Error`SWEC`28`0`1`Errors`0`2`0`Level0`0`ErrMsg`An end of file error has occurred. Please continue or ask your systems administrator to check your application configuration if the problem persists.`ErrCode`28601`

原因：Web ページのレコードの行 ID が入力名 SWERowId に含まれていないことが考えられます。入力名はパラメータ化しておく必要があります。パラメータのソース値でその場所が変更されている可能性があります。

検索カテゴリの取得不能

検索カテゴリの取得不能タイプのエラーでは，通常は次のような Web ページ応答があります。

原因：検索フレームがサーバからダウンロードされなかったことが考えられます。この問題は，前の要求で検索フレームを正しく作成するようサーバに要求していなかったときに発生します。

第 51 章

Baan 仮想ユーザ・スクリプトの作成

VuGen を使用して，Baan 仮想ユーザ・スクリプトを作成します。VuGen で一般的な Baan セッションを記録し，スクリプトを Baan 仮想ユーザ関数で拡張します。

本章では，次の項目について説明します。

- Baan 仮想ユーザ・スクリプトの概要
- Baan 仮想ユーザ関数
- Baan 仮想ユーザ・スクリプトの作成
- Baan 仮想ユーザ・スクリプトについて
- Baan 仮想ユーザ・スクリプトのカスタマイズ

以降の情報は，Baan 仮想ユーザ・スクリプトを対象とします。

Baan 仮想ユーザ・スクリプトの作成

Baan タイプの仮想ユーザ・スクリプトを使用すると，Baan アプリケーションのテストや，ロード中のシステムのテストを行えます。VuGen は，Baan サーバへのログイン情報など，Baan セッション全体を記録します。

アクションを記録する際，VuGen は**コンテキスト・センシティブ**関数を使用して，スクリプトを作成します。コンテキスト・センシティブ関数は，テスト対象アプリケーションのアクションを，GUI オブジェクト（ウィンドウ，リスト，ボタンなど）の面から記述します。操作を記録するたびに，選択されたオブジェクトと実行されたアクションを記述する関数が生成されます。

Baan 仮想ユーザ・スクリプトの概要

Baan 仮想ユーザ・スクリプト を VuGen で記録する前に，使用しているマシンで Baan セッションを開けることを確認してください。

Baan 仮想ユーザ・スクリプトを作成するには，次の手順で行います。

1 **VuGen で Baan スクリプトを作成します。**

新しい Baan テンプレートを作成します。

2 **ユーザ・アクションを記録します。**

一般的なユーザ・アクションを記録します。

3 **トランザクション，ランデブー，コメント，およびメッセージを追加します。**

［挿入］メニューを使用してトランザクション，ランデブー，コメント，およびメッセージを追加し，スクリプトを拡張します。

4 **例外処理を追加し，実行時プロパティを設定します。**

例外を処理する関数を追加し，思考遅延時間を設定し，タイムアウト期間を指定します。ログ記録と反復の実行環境を設定します。

5 パラメータ化を実行します。

記録された定数をパラメータと置き換えます。

6 仮想ユーザ・スクリプトを**保存して実行**します。

VuGen から Baan スクリプトを実行し，［実行ログ］タブで実行時情報を表示します。

Baan 仮想ユーザ関数

VuGen は，Baan ユーザ・セッション中に，本項で説明する関数のほとんどを自動的に記録します。スクリプトには手作業で他の関数をプログラミングすることもできます。Baan 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。

セッション関数

init_session	Baan セッションを初期化します。
close_session	すべての Baan セッションおよびウィンドウを閉じます。
start_session	特定の Baan セッションを開始します。
set_exception	例外処理を指定します。
set_think_time	思考遅延時間の範囲を設定します。
set_default_timeout	標準設定のタイムアウトを設定します。

ボタン・オブジェクト関数

button_press	プッシュ・ボタンを押します。
button_set	指定されたラジオ・ボタンまたはチェック・ボックスの状態を設定します。

編集オブジェクト関数

edit_get_text	編集オブジェクト内のテキストを返します。
edit_set	編集オブジェクト内のすべての内容を置き換えます。
edit_set_insert_pos	カーソルを指定ポイントに置きます。
edit_set_selection	編集オブジェクト内でテキストを選択します。
edit_type	編集オブジェクト内で文字列を入力します。

リスト・オブジェクト関数

list_activate_item	リスト内で項目を有効にします。
list_select_item	リスト項目を選択します。
list_get_selected	リスト内で現在選択されている項目を返します。
list_expand_item	リスト内で非表示の項目を表示します。
list_collapse_item	リスト内で項目を非表示にします。

メニュー・オブジェクト関数

menu_select_item	メニューから項目を選択します。

オブジェクト関数

obj_get_info	オブジェクト属性の値を返します。
obj_get_text	オブジェクトからテキストを読み取ります。
obj_mouse_click	オブジェクト内でクリックします。
obj_mouse_dbl_click	オブジェクト内でダブルクリックします。
obj_mouse_drag	オブジェクト内でマウスをドラッグします。
obj_type	オブジェクトにキーボード入力を送ります。

スクロール・オブジェクト関数

scroll_drag_from_min	最小位置から指定された距離だけスクロール・オブジェクトをドラッグします。
scroll_line	指定された行数分スクロールします。
scroll_page	指定されたページ数分スクロール・オブジェクトを移動します。

タブおよびツールバー・オブジェクト関数

tab_select_item	アクティブなウィンドウでタブを選択します。
toolbar_button_press	ツールバー・ボタンをクリックします。

静的オブジェクト関数

static_get_text	静的テキスト・オブジェクトの内容を返します。

同期関数

obj_wait_info	オブジェクト属性の値を待機します。
tbl_wait_selected_cell	指定されたセルがフォーカスされるのを待機します。
win_wait_info	ウィンドウ属性の値を待機します。

テーブル関数

tbl_activate_cell	指定セル内で［Enter］をクリックします。
tbl_get_cell_data	テーブルから指定されたセルの内容を取得します。
tbl_get_selected_cell	テーブル内で現在選択されているセルを返します。
tbl_press_zoom_button	テーブルのズーム・ボタンをクリックします。
tbl_set_cell_data	セルの内容をテーブル内の指定テキストに設定します。
tbl_set_selected_cell	テーブル・セルを選択します。
tbl_set_selected_ rows	テーブル内で指定された行を選択します。

ウィンドウ・オブジェクト関数

set_window	以降の入力を受信するウィンドウを指定します。
win_activate	ウィンドウをアクティブにします。
win_close	ウィンドウを閉じます。
win_get_text	ウィンドウからテキストを読み取ります。
win_get_info	ウィンドウ属性の値を返します。
win_max	ウィンドウを画面いっぱいに最大化します。

win_min	ウィンドウをアイコンに最小化します。
win_mouse_click	ウィンドウ内でクリックします。
win_mouse_dbl_click	ウィンドウ内でダブルクリックします。
win_mouse_drag	ウィンドウ内でマウスをドラッグします。
win_move	ウィンドウを新しい絶対位置に移動します。
win_resize	ウィンドウのサイズを変更します。
win_restore	アイコン化または最大化されたウィンドウを元のサイズに戻します。
win_type	ウィンドウにキーボード入力を送ります。

その他の関数

wait	テキスト実行を指定時間だけ一時停止します。

スクリプトは，一般仮想ユーザ関数（**lr_output_message**，**lr_rendezvous** など）を使用してさらに拡張できます。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。

Baan 仮想ユーザ・スクリプトの作成

Baan テンプレートを作成したら，ユーザ・アクションの記録を開始します。

新しい Baan 仮想ユーザ・スクリプトを作成するには，次の手順で行います。

1 ［**vuser_init**］セクションを選択して，ログイン プロシージャをそのセクション内に記録します。

2 ［**記録**］ボタンをクリックして，［記録開始］ダイアログ・ボックス内で Baan アプリケーションの場所を指定します。

3 ［**アクション**］セクションに切り替えて，一般的なユーザ・アクションを記録します。

4 思考遅延時間，例外処理，タイムアウト設定に対して Baan 仮想ユーザ関数を挿入します。

```
set_think_time(MINTHINK,MAXTHINK);
set_window ("Menu browser [User:bsp ] [812]", 10);
menu_select_item ("File;Run Program...");
. . .
```

5 スクリプトにトランザクションを追加します。［**挿入**］＞［**トランザクション開始**］を選択してトランザクションの開始を指定し，［**挿入**］＞［**トランザクション終了**］を選択してトランザクションの最後を指定します。

```
lr_start_transaction("all_str_ses");
  button_press0 ("F1_OK");
  set_window ("tdpur4101m000: Maintain Purchase Orders [812]", 300);
lr_end_transaction("all_str_ses", LR_PASS);
```

6 ［**挿入**］メニューを使用して，ランデブー・ポイント，コメント，またはメッセージをスクリプトに追加します。

7 スクリプトをパラメータ化します。パラメータで置換する（引用符内の）文字列をクリックし，［**パラメータで置換**］を右クリックして選択します。詳細については，第 7 章 「パラメータの定義」を参照してください。

8 反復とログ記録に適切な実行環境を設定します。

9 スクリプトを保存して，VuGen から実行します。

Baan 仮想ユーザ・スクリプトについて

記録されたスクリプトには，記録時にユーザによって実行されたアクションがすべて表示されます。**コンテキスト・センシティブ**関数には，アプリケーションのオブジェクトで実行されたアクションがすべて表示されます。次の例では，VuGen は，ウィンドウへのフォーカス，メニュー項目の選択，ボタンのクリックを記録しています。また，オブジェクト **Form1** がフォーカス内に現れるまでにかかる時間を分析するために，トランザクションがマークされています。

```
set_window ("tccom1501m000: Display Customers [550]", 30);
menu_select_item ("Edit;Find...Ctrl+F");
set_window ("Display Customers - Find", 300);
type ("100004");
lr_start_transaction("rses_find");
button_press0 ("F1_OK");
set_window ("tccom1501m000: Display Customers [550]", 30);
obj_wait_info("Form 1","focused","1",100);
lr_end_transaction("rses_find", LR_PASS);
```

制御フロー・ロジックを追加してスクリプト内でループを作成しておけば，スクリプト全体を反復させなくてもよくなります。

```
for (loop = 0 ; loop < READLOOP; loop++){
  set_window ("tccom1501m000 :Display Customers [550]", 30);
  menu_select_item ("Edit;Find...Ctrl+F");
  set_window ("Display Customers - Find", 300);
  type ("100004");
  lr_start_transaction("rses_find");
  button_press0 ("F1_OK");
  set_window ("tccom1501m000 :Display Customers [550]", 30);
  obj_wait_info("Form 1","focused","1",100);
  lr_end_transaction("rses_find", LR_PASS);
. . . .
```

データベース内のデータの重複を避けるために，ステートメントをパラメータ化する必要がある場合があります。詳細については，第 7 章 「パラメータの定義」を参照してください。

Baan 仮想ユーザ・スクリプトのカスタマイズ

スクリプトは，いつでも VuGen 内から表示や編集ができます。Baan 固有の関数を使用して，次の領域でスクリプトの実行をカスタマイズできます。

- 思考遅延時間
- 例外処理
- タイムアウトの設定

思考遅延時間

スクリプトの実行に対して，思考遅延時間の範囲を設定できます。思考遅延時間は，実際のユーザの作業パターン，つまりアクション間でユーザが一時停止する時間をエミュレートします。思考遅延時間の範囲の始点と終点は，**set_think_time** 関数を使用して設定します。各ステートメントの後，仮想ユーザは，思考遅延時間の期間，つまり指定範囲内の乱数値に相当する時間だけ一時停止します。

希望する思考遅延時間の範囲がスクリプトを通じて一定の場合は，次の例のように，範囲の最初と最後を定数として定義できます。

次の例では，思考遅延時間の範囲は，500 ～ 1000 ミリ秒に設定されています。

```
#define MINTHINK 500
#define MAXTHINK 1000

int LR_FUNC Actions(LR_PARAM p)
{
      set_think_time(MINTHINK,MAXTHINK);
      set_window ("Menu browser [User:bsp ] [812]", 10);
. . .
```

例外処理

Baan 仮想ユーザでは，メッセージやエラー・ウィンドウなど，再生中に発生した例外を処理する方法を指定できます。

set_exception 関数を使用すると，例外発生時に実行する関数を指定できます。

次の例では，**set_exception** 関数は，［Print Sales Invoices］ウィンドウが開いたときに **close** 関数を実行するよう仮想ユーザに命令します。**close** 関数は，スクリプトですでに定義済みです。

```
int close(char title[])
{
      win_close(title);
}
Actions()
{
      set_exception("ttstps0014: Print Sales Invoices",close);
      set_window ("Menu browser [User:bsp ] [812]", 10);
      menu_select_item ("File;Run Program...");
      set_window ("ttdsk2080m000 :Run Program [812]", 10);
      type ("tdsls4101m000");
      . . . ;
```

タイムアウトの設定

関数に対して，標準のタイムアウトを設定できます。このタイムアウトは，同期を使用してすべての関数（**obj_wait_info**，**win_wait_info** など）に適用されます。

タイムアウト期間を指定するパラメータを含む関数（**set_window** など）では，指定したタイムアウトが標準設定のタイムアウトより優先されます。

```
button_press ("F3_Continue");
win_wait_info("ttstpsplopen: Select Device [000]","displayed","0",10);
```

第 11 部

メール・サービス・プロトコル

第 52 章

メール・サービス仮想ユーザ・スクリプトの作成

VuGen では，プロトコル・レベルでいくつかのメール・サービスをテストできます。VuGen は，メール送信，およびメール・サーバに対して実行されるほとんどの標準的な操作をエミュレートします。

本章では，次の項目について説明します。

- メール・サービス 仮想ユーザ・スクリプトの概要
- IMAP 関数での作業
- MAPI 関数での作業
- POP3 関数での作業
- SMTP 関数での作業

以下の情報は，IMAP，MAPI，POP3，および SMTP 仮想ユーザ・スクリプトを対象とします。

メール・サービス仮想ユーザ・スクリプトの作成について

メール・サービス・プロトコルは，メールの表示や送信など，電子メール・クライアントで作業しているユーザをエミュレートします。サポートされているメール・サービスは次のとおりです。

- IMAP（Internet Messaging）
- MAPI（MS Exchange）
- POP3（Post Office Protocol）
- SMTP（Simple Mail Transfer Protocol）

メール・サービス仮想ユーザ・スクリプトでは，記録と再生の両方がサポートされています。ただし，MAPI では再生だけがサポートされています。

メール・プロトコルの 1 つを使用してアプリケーションを記録するとき，VuGen によってメール・クライアントのアクションをエミュレートする関数が生成されます。仮想ユーザ・スクリプトの作成に使用するプログラミング言語として，C 言語または Visual Basic スクリプティングのいずれかを指定できます。詳細については，第 4 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。通信に複数のプロトコルが使われている場合は，両方を記録できます。複数のメール・プロトコルを同時に，または 1 つのメール・プロトコルを HTTP または WinSock と一緒に記録できます。マルチ・プロトコルの指定の詳細については，第 3 章 「VuGen を使った記録」を参照してください。

メール・サービス関数はすべて，対で用意されています。一方がグローバル・セッション用で，もう一方で特定のメール・セッションを指定します。たとえば，**imap_logon** はグローバルに IMAP サーバにログオンしますが，**imap_logon_ex** は特定のセッションのために IMAP サーバにログオンします。

メール・サービス 仮想ユーザ・スクリプトの概要

本項では，VuGen を使用したメール・サービス仮想ユーザ・スクリプトの開発プロセスの概要を説明します。

メール・サービス仮想ユーザ・スクリプトを作成するには，次の手順で行います。

1 **VuGen を使用して，基本スクリプトを作成します。**

VuGen を起動して，1 つまたは複数のメール・プロトコルを対象とする仮想ユーザ・スクリプトを新規作成します。

2 **VuGen を使用して，基本スクリプトを記録します（MAPI を除く）。**

記録対象アプリケーションを選択します。アプリケーションを対象に標準的な操作を実行します。詳細については，第 3 章 「VuGen を使った記録」を参照してください。

MAPI では，記録はサポートされていません。その代わりに空の MAPI スクリプトを作成し，**mapi** 関数を手作業で挿入します。関数の使用例については，「**オンライン関数リファレンス**」（[**ヘルプ**] ＞ [**関数リファレンス**]）を参照してください。

3 **スクリプトを拡張します。**

スクリプトに，トランザクション，ランデブー・ポイント，制御フロー構造を挿入して，スクリプトを拡張します。

詳細については，第 6 章 「仮想ユーザ・スクリプトの拡張」を参照してください。

4 **パラメータを定義します（任意）。**

スクリプトに記録された固定値に対してパラメータを定義します。固定値をパラメータで置き換えることによって，異なる値を使って同じビジネス・プロセスを何度も繰り返すことができます。

詳細については，第 7 章 「パラメータの定義」を参照してください。

5 **ステートメントを相関させます（任意）。**

ステートメントの相関によって，あるビジネス・プロセスの結果を以降の別のビジネス・プロセスで使用できるようになります。

詳細については，第 8 章 「ステートメントの相関」を参照してください。

6 **実行環境を設定します。**

実行環境の設定では，スクリプト実行時の仮想ユーザの振る舞いを制御します。設定には，ループ，ログ，タイミング情報などがあります。

詳細については，第 9 章 「実行環境の設定」を参照してください。

7 **VuGen でスクリプトを実行します。**

VuGen でスクリプトを保存し実行して，正しく動作することを確認します。

詳細については，第 11 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。

仮想ユーザ・スクリプトを作成したら，Windows または UNIX プラットフォームで シナリオに組み込みます。仮想ユーザ・スクリプトのシナリオへの組み込みの詳細については，『**LoadRunner コントローラ・ユーザーズ・ガイド**』を参照してください。

IMAP 関数での作業

IMAP 仮想ユーザ・スクリプト関数は，Internet Mail Application Protocol（IMAP）を記録します。IMAP 関数の名前には，**imap** という接頭辞が付いています。これらの関数の構文情報の詳細については，「**オンライン関数リファレンス**」（[**ヘルプ**] ＞ [**関数リファレンス**]）を参照してください。

関数名	説明
imap_append[_ex]	メールボックスの最後にメッセージを追加します。
imap_check[_ex]	現在のメールボックスのチェックポイントを要求します。
imap_close[_ex]	現在のメールボックスを閉じます。
imap_copy[_ex]	メール・メッセージを別のメールボックスにコピーします。
imap_create[_ex]	メールボックスを作成します。
imap_custom_request[_ex]	ユーザ定義の IMAP 要求を実行します。
imap_delete[_ex]	指定のメールボックスを削除します。
imap_examine[_ex]	メールボックスを検証します。
imap_expunge[_ex]	マークしたすべてのメッセージを削除します。
imap_fetch[_ex]	メールボックス・メッセージに関連付けられたデータを取得します。
imap_free_ex	IMAP セッション記述子を解放します。
imap_get_attribute_int[_ex]	メールボックス属性を返します。
imap_get_attribute_sz[_ex]	メールボックス属性を文字列として返します。
imap_get_result[_ex]	IMAP サーバのリターン・コードを取得します。
imap_list_mailboxes[_ex]	使用可能なメールボックスを一覧表示します。
imap_list_subscriptions[_ex]	購読されているかアクティブになっているメールボックスを一覧表示します。
imap_logon[_ex]	IMAP サーバにログインします。
imap_logout[_ex]	IMAP サーバからログオフします。

imap_noop[_ex]	NOOP 操作を実行します。
imap_search[_ex]	メールボックス内をキーワードで検索します。
imap_select[_ex]	メールボックスを選択します。
imap_status[_ex]	メールボックスのステータスを要求します。
imap_store[_ex]	メールボックス・メッセージに関連付けられたデータを変更します。
imap_subscribe[_ex]	メールボックスを購読するかアクティブにします。
imap_unsubscribe[_ex]	メールボックスを購読解除またはアクティブ解除します。

次の例では，**imap_create** 関数を使っていくつかの新しいメールボックスを作成します。作成するのは，**Products**，**Solutions**，および **FAQs** です。

```
Actions()
{
    imap_logon("ImapLogon",
                "URL=imap://johnd:letmein@exchange.mycompany.com",
                LAST);

    imap_create("CreateMailboxes",
                "Mailbox=Products",
                "Mailbox=Solutions",
                "Mailbox=FAQs",
                LAST);

    imap_logout( );

    return 1;
}
```

MAPI 関数での作業

MAPI 仮想ユーザ・スクリプト関数は，MS Exchage サーバを対象とする操作を記録します。MAPI 関数の名前には，**mapi** という接頭辞が付いています。これらの関数の構文情報の詳細については，「**オンライン関数リファレンス**」（[**ヘルプ**] ＞ [**関数リファレンス**]）を参照してください。

関数名	説明
mapi_delete_mail[_ex]	現在の電子メール・エントリまたは選択された電子メール・エントリを削除します。
mapi_get_property_sz[_ex]	MAPI セッションからプロパティ値を取得します。
mapi_logon[_ex]	MS Exchange にログオンします。
mapi_logout[_ex]	MS Exchange からログアウトします。
mapi_read_next_mail[_ex]	メールボックスの次のメールを読み取ります。
mapi_send_mail[_ex]	受信者に電子メールを送信します。
mapi_set_property_sz[_ex]	MAPI 属性を設定します。

次の例では，**mapi_send_mail** 関数を使用して，MS Exchange サーバを通じて付せんを送信します。

```
Actions()
{
    mapi_logon("Logon",
                "ProfileName=John Smith",
                "ProfilePass=Tiger",
                LAST);

    // 付せんメッセージを送信
    mapi_send_mail("SendMail",
                "To=user1@techno.merc-int.com",
                "Cc=user0002t@techno.merc-int.com",
                "Subject= < GROUP > : < VUID > @ < DATE > ",
                "Type=Ipm.StickyNote",
                "Body=Please update your profile today.",
                LAST);

    mapi_logout( );

    return 1;
}
```

POP3 関数での作業

POP3 仮想ユーザ・スクリプト関数は，POP3（Post Office Protocol）を使用してアクションをエミュレートします。POP3 関数の名前には，**pop3** という接頭辞が付いています。これらの関数の構文情報の詳細については，「**オンライン関数リファレンス**」（[**ヘルプ**] > [**関数リファレンス**]）を参照してください。

関数名	説明
pop3_command[_ex]	POP3 サーバにコマンドを送信します。
pop3_delete[_ex]	サーバ上のメッセージを削除します。
pop3_free[_ex]	コマンドから POP3 サーバを解放します。
pop3_list[_ex]	POP3 サーバ上のメッセージを一覧表示します。
pop3_logoff[_ex]	POP3 サーバからログオフします。

pop3_logon[_ex]	POP3 サーバにログオンします。
pop3_retrieve[_ex]	POP3 サーバからメッセージを取得します。

次の例では，**pop3_retrieve** 関数を使って POP3 サーバから 5 つのメッセージを取得しています。

```
Actions()
{
pop3_logon("Login", "URL=pop3://user0004t:my_pwd@techno.merc-
int.com", LAST);

// サーバ上のメッセージをすべて表示し，値を取得する
totalMessages = pop3_list("POP3", LAST);

// 取得した値を表示する（pop3_list 関数を使って表示することもできる）
lr_log_message("There are %d total messages on the server.¥r¥n¥r¥n",
totalMessages);

// 5 つのメッセージをサーバから削除せずに取得する
pop3_retrieve("POP3", "RetrieveList=1:5", "DeleteMail=false", LAST);

pop3_logoff();

      return 1;
}
```

SMTP 関数での作業

SMTP 仮想ユーザ・スクリプト関数は，SMTP トラフィックをエミュレートします。SMTP 関数の名前には，**smtp** という接頭辞が付いています。これらの関数の構文情報の詳細については，「**オンライン関数リファレンス**」（[**ヘルプ**] ＞［**関数リファレンス**]）を参照してください。

関数名	説明
smtp_abort_mail[_ex]	SMTP メッセージの転送を中止します。
smtp_free[_ex]	コマンドから SMTP サーバを解放します。
smtp_logon[_ex]	SMTP サーバにログオンします。
smtp_logout[_ex]	SMTP サーバからログオフします。
smtp_send_mail[_ex]	SMTP メッセージを送信します。
smtp_translate[_ex]	SMTP メッセージを変換します。

次の例では，**smtp_send_mail** 関数を使って，SMTP メール・サーバ **techno** を通じてメール・メッセージを送信しています。

```
Actions()
{
	smtp_logon("Logon",
		"URL=smtp://user0001t@techno.merc-int.com",
			"CommonName=Smtp Test User 0001",
		NULL);

	smtp_send_mail("SendMail",
				"To=user0002t@merc-int.com",
				"Subject=MIC Smtp:Sample Test",
				"MAILOPTIONS",
				"X-Priority: 3",
				"X-MSMail-Priority:Medium",
				"X-Mailer:Microsoft Outlook Express 5.50.400¥r¥n",
				"X-MimeOLE:By Microsoft MimeOLE
V5.50.00¥r¥n",
				"MAILDATA",
				"MessageText="
				    "Content-Type:text/plain;¥r¥n"
				    "¥tcharset=¥"iso-8859-1¥"¥r¥n"
				    "Test,¥r¥n"
				    "MessageBlob=16384",
			NULL);

	smtp_logout();

	return 1;
}
```

第 12 部

ミドルウェア・プロトコル

第 53 章

Jacada 仮想ユーザ・スクリプトの作成

VuGen では，Jacada Interface Server との通信を記録できます。記録したスクリプトは，そのまま実行したり，標準の Java ライブラリ関数および LoadRunner 固有の Java 関数を使って拡張したりすることができます。

本章では，次の項目について説明します。

- Jacada 仮想ユーザの概要
- Jacada 仮想ユーザの記録
- Jacada 仮想ユーザの再生
- Jacada 仮想ユーザ・スクリプトについて
- Jacada 仮想ユーザ・スクリプトでの作業

以降の情報は，Jacada 仮想ユーザ・スクリプトを対象とします。

Jacada 仮想ユーザ・スクリプトの作成

Jacada Interface Server は，メインフレーム・アプリケーションのためのインタフェース・レイヤを提供します。このレイヤは，ユーザ・インタフェースとアプリケーション・ロジックを分けることで，規格や技術の変更が組織に影響しないようにします。Jacada サーバでは，単色の端末画面のアプリケーションで作業するのではなく，環境をユーザ・フレンドリなインタフェースに変換します。

VuGen では，Jacada の Java シンクライアントを記録します。HTML シンクライアントを使用した Jacada サーバとの通信を記録するには，Web HTTP/HTML タイプの仮想ユーザを使用します。詳細については，第 32 章 「Web 仮想ユーザ・スクリプトの作成」を参照してください。

スクリプトを作成するには，VuGen を起動して，標準的なアクションとビジネス・プロセスを記録します。VuGen によって，すべてのアクションを表すスクリプトが生成されます。このスクリプトは Java 互換です。

スクリプトの準備ができたら，VuGen からスタンドアロン・モードで実行します。Sun の標準 Java コンパイラ，**javac.exe** によって，スクリプトにエラーがないか確認された後，コンパイルされます。スクリプトが正しく機能することを確認したら，LoadRunner のシナリオに組み込みます。

スクリプトを記録と手動で拡張する場合，Java 仮想ユーザ・スクリプトに関連したすべてのガイドラインと制限が適用されます。関数の構文とシステムの設定で留意すべき点については，第 24 章「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。

以下の項では，記録オプション，実行環境設定，および相関について説明します。

Jacada 仮想ユーザの概要

次の手順では，Jacada 仮想ユーザ・スクリプトの作成方法について概説します。

1 **記録するマシンの設定が正しいことを確認します。**

記録を開始する前に，マシンが Java を扱えるように正しく設定されていることを確認します。詳細については，第 24 章「Java 仮想ユーザ・スクリプトのプログラミング」と「Read Me」ファイルを参照してください。

2 **新しい Jacada 仮想ユーザ・スクリプトを作成します。**

［ミドルウェア］グループから **Jacada** タイプの仮想ユーザを選択します。

3 **スクリプトの記録パラメータとオプションを設定します。**

作業ディレクトリやパスなど，アプレットまたはアプリケーションに対するパラメータを指定します。JVM，相関，レコーダ，およびデバッグに関する記録オプションも設定できます。詳細については，第 14 章 「Java 記録オプションの設定」を参照してください。

4 **標準的なユーザ・アクションを記録します。**

スクリプトの記録を開始します。Jacada サーバを対象に標準的なアクションを実行します。VuGen によってアクションが記録され，仮想ユーザ・スクリプトが生成されます。

5 **仮想ユーザ・スクリプトを拡張します。**

LoadRunner 固有の関数を追加して，仮想ユーザ・スクリプトを拡張します。詳細については，第 24 章 「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。取り込むクラスまたはメソッドを選択するには，組み込みの Java 関数ナビゲータを使用できます。詳細については，第 46 章 「EJB テストの実行」を参照してください。

6 **仮想ユーザ・スクリプトをパラメータ化します。**

記録された定数をパラメータで置き換えます。文字列の全体または一部をパラメータ化できます。詳細については，第 7 章 「パラメータの定義」を参照してください。

7 **スクリプトの実行環境の設定を行います。**

仮想ユーザ・スクリプトの実行環境の設定を行います。実行環境の設定では，スクリプト実行時の環境を定義します。Java 固有の実行環境の設定については，第 16 章 「Java 実行環境の設定」を参照してください。

8 **仮想ユーザ・スクリプトを保存して実行します。**

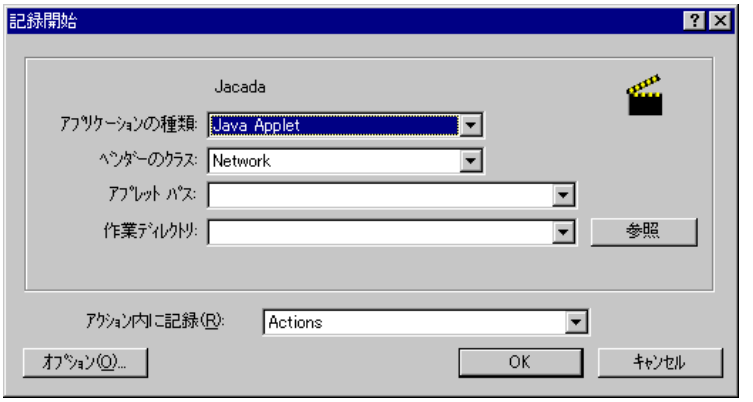
VuGen から スクリプトを実行して，実行時の情報を実行ログで確認します。詳細については，第 11 章 「スタンドアロン・モードでの 仮想ユーザ・スクリプトの実行」を参照してください。

Jacada 仮想ユーザの記録

Jacada スクリプトを記録して，完全互換の Java プログラムを作成します。

Jacada スクリプトを記録するには，次の手順で行います。

1 記録を開始するには，［**ファイル**］＞［**新規作成**］を選択し，［ミドルウェア］カテゴリから［**Jacada**］を選択します。［記録開始］ダイアログ・ボックスが開きます。



2 アプリケーションの種類として，Internet Explorer または Netscape を選択します。

3 ［**ベンダーのクラス**］ボックスでは，［**Network**］クラスが標準設定になっています。クラスパスに **clbase.jar** がある場合は，［**Local**］ベンダー・クラスを選択します。

4 ブラウザのパスと Jacada サーバの開始ページの URL を指定します。

［**作業ディレクトリ**］は，作業ディレクトリにアクセスするアプリケーション（プロパティ・ファイルの読み取り，ログ・ファイルの作成など）だけで必要です。

5 JVM のコマンド・ライン・パラメータなどの記録オプションを設定するには，［**オプション**］をクリックします。記録オプションの設定の詳細については，第 14 章 「Java 記録オプションの設定」を参照してください。

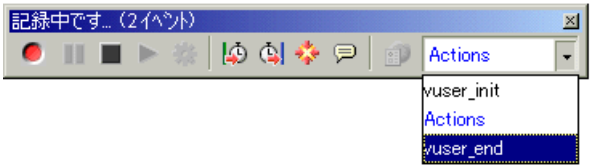
6 ［**アクション内に記録**］ボックスで，記録を挿入するセクションを選択します。vuser_init，Actions，および vuser_end の各セクションに対応する **init**，**action**，および **end** の 3 つのセクションがあります。次の表に，各メソッドに何を含め，どのタイミングで呼び出されるかを示します。

Actions クラス内のメソッド	記録対象アクション	エミュレーション内容	実行のタイミング
init	**vuser_init**	サーバへのログイン	初期化時
action	**Actions=**	クライアントの動作	実行時
end	**vuser_end**	ログオフ処理	終了時または停止時

7 ［**OK**］をクリックして記録を開始します。VuGen によってアプリケーションが開始されます。VuGen は最小化され，進行状況バーとフローティング記録ツールバーが開きます。進行状況バーには，そのときロードされているクラスの名前が表示されます。これは，Java 記録機能が動作中であることを示します。

8 アプリケーション内で，記録したい標準的な操作を行います。記録中にメソッドを切り替えるには，フローティング・ツールバーを使います。

9 標準的なユーザ・アクションを記録した後で，フローティング・ツールバーで **vuser_end** メソッドを選択します。



ログオフ手順を実行します。VuGen によって手順がスクリプトの **vuser_end** メソッドに記録されます。

10 ［記録］ツールバーで，［**停止**］ボタンをクリックします。VuGen エディタにより，記録されたすべてのステートメントが表示されます。

11 ［**上書き保存**］ボタンをクリックして，スクリプトを保存します。［テストを保存］ダイアログ・ボックスが開きます（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。

Jacada 仮想ユーザの再生

LoadRunner を実行するマシンに，Sun が提供している JDK が正しくインストールされていることを確認してください（JRE だけでは十分ではありません）。**classpath** および **path** 環境変数が JDK のインストール手順に従って設定されていることを確認してください。仮想ユーザ・スクリプトを再生する前に，JDK および関連 Java クラスに応じて環境が正しく設定されていることを確認してください。

また，再生前に，Jacada サーバから **clbase.jar** ファイルをダウンロードする必要があります。Java 仮想ユーザによって使用されるクラスはすべて，マシンの CLASSPATH 環境変数，または，実行環境設定の［**Classpath**］パネルで設定されているクラスパス に含まれている必要があります。

Jacada サーバは，記録されたスクリプトにおける順序とは異なる順序でレガシー・システムの画面を戻すことがあります。これにより，再生時に例外が発生する可能性があります。この例外の扱い方については，マーキュリー・インタラクティブのサポート窓口までお問い合わせください。

必要な環境設定の詳細については，第 24 章 「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。

Jacada 仮想ユーザ・スクリプトについて

Jacada セッションを記録すると，VuGen は，サーバへのすべての呼び出しをログに記録し，LoadRunner による拡張機能を備えたスクリプトを生成します。これらの関数は，アプリケーションまたはアプレットにおけるユーザのすべてのアクションを表します。スクリプトには，正しく再生するための例外処理も組み込まれています。

記録されたスクリプトは，次の 3 つのセクションで構成されています。

- **インポート**

  インポート・セクションはスクリプトの先頭にあります。ここにはスクリプトのコンパイルに必要なすべてのパッケージへの参照が含まれます。

- **コード**

  コード・セクションには，Actions クラスと，**init**，**actions**，および **end** メソッド内に記録されたコードが含まれます。コード・セクションには，サーバに送信された各コマンドの例外処理を行う try-catch ブロックも含まれています。

➤ **変数**

end メソッドの後の**変数セクション**には，コードで使用される変数のすべての型宣言が含まれます。

記録が終わったら，スクリプトの関数に変更を加えたり，Java または LoadRunner 関数を追加して，スクリプトを拡張したりできます。Java 仮想ユーザをスレッドとして実行する場合，スクリプトに追加する Java コードはスレッドセーフである必要があります。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。

Jacada 仮想ユーザ・スクリプトでの作業

Jacada スクリプトの Actions クラスには，2 つの主要な部分，**プロパティ** と **本体** があります。プロパティ部分では，サーバのプロパティを取得します。その後 VuGen によって，システム・プロパティが設定され，Jacada サーバへの接続が行われます。

```
// システム・プロパティを設定する
        _properties = new Properties(System.getProperties());
        _properties.put("com.ms.applet.enable.logging", "true");
        System.setProperties(_properties);

        _jacadavirtualuser = new cst.client.manager.JacadaVirtualUser();

        lr.think_time(4);
        _jacadavirtualuser.connectUsingPorts("localhost", 1100,
"LOADTEST", "", "", "");
        . . .
```

スクリプトの本体には，ユーザ・アクションのほか，**checkFieldValue** メソッドと **checkTableCell** メソッドの例外処理ブロックが含まれています。

```
        l. . .
/*
try {
        _jacadavirtualuser.checkFieldValue(23, "S44452BA");
        }catch( java.lang.Exception e ) {
         lr.log_message(e.getMessage());
        }
*/      l. . .
/*
try {
        _jacadavirtualuser.checkTableCell(41, 0, 0, "");
        }catch( java.lang.Exception e ) {
         lr.log_message(e.getMessage());
        }
*/      l. . .
```

checkField メソッドには，フィールド ID 番号と期待値の 2 つの引数があります。**checkTableCell** メソッドには，テーブル ID，行，カラム，期待値の 4 つの引数があります。期待値と戻り値の間に不一致がある場合は，例外が生成されます。

標準設定では，try-catch 例外処理ブロックはコメントになっています。これをスクリプトで使用するには，コメントの印を削除してください。

記録されたスクリプトには，任意の LoadRunner Java 関数を追加できます。これらの関数の一覧とスクリプトへの追加方法については，第 24 章 「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。

第 54 章

Tuxedo 仮想ユーザ・スクリプトの作成

VuGen を使用して，Tuxedo クライアント・アプリケーションと Tuxedo アプリケーション・サーバの間の通信を記録します。その結果生成されるスクリプトを Tuxedo 仮想ユーザ・スクリプトと呼びます。

本章では，次の項目について説明します。

- Tuxedo 仮想ユーザ・スクリプトの概要
- LRT 関数の使用
- Tuxedo 仮想ユーザ・スクリプトについて
- Tuxedo バッファ・データの表示
- Tuxedo 仮想ユーザの環境設定の定義
- Tuxedo アプリケーションのデバッグ
- Tuxedo スクリプトの相関

以降の情報は，PeopleSoft-Tuxedo，Tuxedo 6 および Tuxedo 7 仮想ユーザ・スクリプトを対象とします。

Tuxedo 仮想ユーザ・スクリプトについて

Tuxedo アプリケーションを記録すると，VuGen は，記録されたアクションを記述する LRT 関数を生成します。これらの関数は，Tuxedo クライアントとサーバの間の通信をエミュレートします。各 LRT 関数は，接頭辞 **lrt** で始まります。

接頭辞 **lrt** に加え，**tp**，**tx**，**F** といった補助接頭辞を使用する関数もあります。これらの補助接頭辞は，実際の Tuxedo 関数と同様に，関数の型を示します。補助接頭辞 **tp** は，Tuxedo クライアントの tp セッションを示します。たとえば，**lrt_tpcall** は，サービス要求を送信して応答を待ちます。補助接頭辞 **tx** は，グローバルな tx セッションを表します。たとえば，**lrt_tx_begin** はグローバルなトランザクションを開始します。補助接頭辞 **F** は，FML バッファに関連する関数を表します。たとえば，**lrt_Finitialize** は既存のバッファを初期化します。

補助接頭辞のない関数は，標準 C 関数をエミュレートします。たとえば，**lrt_strcpy** は，C 関数 **strcpy** と同じように文字列をコピーします。

VuGen のメイン・ウィンドウで，記録されたスクリプトの表示と編集ができます。セッション中に記録された LRT 関数が VuGen ウィンドウに表示されるので，ネットワークの動作状況を視覚的に追跡できます。

記録前の作業

記録を行う前に，Tuxedo ディレクトリ %TUXDIR%¥bin がパスに含まれていることを確認します。

VuGen の再起動後に環境変数を変更している場合，VuGen は，環境変数の現在の値ではなく元の値を記録することがあります。

不整合を防ぐため，Tuxedo アプリケーションを記録する前には VuGen を再起動します。

Tuxedo 仮想ユーザ・スクリプトの概要

本項では，VuGen を使用して Tuxedo 仮想ユーザ・スクリプトを作成するプロセスの概要を説明します。

Tuxedo 仮想ユーザ・スクリプトの作成は，次の手順で行います。

1 **VuGen を使って基本となるスクリプトを記録します。**

VuGen を起動して，新しい仮想ユーザ・スクリプトを作成します。仮想ユーザのタイプとして，Tuxedo6（Tuxedo Version 6.x の記録用）または Tuxedo7（Tuxedo Version 7.x の記録用）を指定します。記録するアプリケーションを選択します。アプリケーションを使用した標準的な操作を記録します。

詳細については，第 3 章 「VuGen を使った記録」を参照してください。

2 **スクリプトを拡張します。**

スクリプトに，トランザクション，ランデブー・ポイント，および制御フロー構造を挿入して，スクリプトを拡張します。

詳細については，第 6 章 「仮想ユーザ・スクリプトの拡張」を参照してください。

3 **パラメータを定義します（任意）。**

スクリプトに記録された固定値に対してパラメータを定義します。固定値をパラメータに置き換えることによって，異なる値を使って同じビジネス・プロセスを何度も繰り返すことができます。

詳細については，第 7 章 「パラメータの定義」を参照してください。

4 **ステートメントを相関させます（任意）。**

ステートメントを相関することにより，あるビジネス・プロセスの結果を後続のビジネス・プロセスで使用できます。

詳細については，第 8 章 「ステートメントの相関」を参照してください。

5 **実行環境を設定します。**

実行環境を設定することによって，スクリプト実行時の仮想ユーザの動作を制御します。設定には，反復，ログ，タイミング情報などがあります。

詳細については，第 9 章 「実行環境の設定」を参照してください。

6 **VuGen からスクリプトを実行します。**

VuGen でスクリプトを保存して実行し，正しく動作することを確認します。

詳細については，第 11 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。

作成した Tuxedo 仮想ユーザ・スクリプトは，Windows または UNIX プラットフォームのシナリオに統合します。仮想ユーザ・スクリプトのシナリオへの統合については，『**LoadRunner コントローラ・ユーザーズ・ガイド**』を参照してください。

LRT 関数の使用

Tuxedo クライアントのサーバとの通信をエミュレートするために開発された関数を，LRT 関数と呼びます。各 LRT 仮想ユーザ関数には，接頭辞 **lrt** が付きます。VuGen は，Tuxedo セッション中に，本項に列挙する LRT 関数のほとんどを自動的に記録します。スクリプトに任意の関数を手作業でプログラミングすることもできます。LRT 関数の構文と例については，**「オンライン関数リファレンス」**（[**ヘルプ**] ＞ [**関数リファレンス**]）を参照してください。

注：関数名に省略可能な「32」が含まれている FML バッファ関数があります。これらは FML32 バージョンの関数です。

バッファ操作関数

lrt_Fadd[32]_fld	FML バッファに新しいフィールドを追加します。
lrt_Finitialize[32]	既存の FML バッファ fbfr を初期化します。
lrt_Fldid[32]	フィールド名をフィールド識別子にマップします。
lrt_Fname[32]	フィールド名へのマップ・フィールド識別子を提供します。
lrt_memcpy	指定されたバイト数を，コピー元からコピー先にコピーします。
lrt_strcpy	C 関数の strcpy と同様に，文字列をコピーします。
lrt_tpalloc	**type** 型のバッファへのポインタを返します。
lrt_tprealloc	型付きバッファのサイズを変更します。
lrt_tpfree	型付きバッファを解放します。
lrt_tptypes	型付きバッファに関する情報を調べます。

クライアント / サーバ・セッション関数

lrt_tpchkauth	アプリケーションが認証を要求するかどうかを確認します。
lrt_tpinitialize	クライアントが System/T アプリケーションに参加できるようにします。
lrt_tpterm	クライアントを System/T アプリケーションから削除します。

通信関数

lrt_tpacall	サービス要求を送信します。
lrt_tpbroadcast	名前によって通知をブロードキャストします。
lrt_tpcall	サービス要求を送信し，その応答を待機します。
lrt_tpcancel	呼び出し記述子を取り消します。
lrt_tpchkunsol	非請求メッセージがないか調べます。
lrt_tpconnect	会話サービス接続を確立します。

lrt_tpdequeue	メッセージをキューから除外します。
lrt_tpdiscon	会話型サービス接続を切断します。
lrt_tpenqueue	メッセージをキューに格納します。
lrt_tpgetrply	以前に送信された要求からの応答を返します。
lrt_tpgprio	送受信された最新の要求の優先順位を返します。
lrt_tpnotify	クライアントに通知を送信します。
lrt_tppost	イベントを送信します。
lrt_tprecv	会話接続でメッセージを受信します。
lrt_tpsend	会話接続でメッセージを送信します。
lrt_tpsetunsol	非請求メッセージを処理するメソッドを設定します。
lrt_tpsprio	次に送信または転送される要求の優先順位を設定します。
lrt_tpsubscribe	イベントを受け取ります。
lrt_tpunsubscribe	イベントを購読解除します。

環境変数関数

lrt_set_env_list	環境変数のリストを設定します。
lrt_tuxgetenv	環境名に対応する値を返します。
lrt_tuxputenv	既存の環境値を変更するか，環境に値を追加します。
lrt_tuxreadenv	ファイルから環境に変数を追加します。

エラー処理関数

lrt_abort_on_error	直前の Tuxedo 関数呼び出しがエラーになった場合，現在のトランザクションを中止します。
lrt_Fstrerror[32]	FML エラーのエラー・メッセージ文字列を取得します。
lrt_getFerror[32]	最新の失敗 FML 操作のエラー・ステータス・コードを取得します。
lrt_gettperrno	最新の Tuxedo トランザクション・モニタ関数のエラー・ステータス・コードを取得します。
lrt_gettpurcode	アプリケーションのリターン・コードを取得します。
lrt_tpstrerror	System/T エラーのエラー・メッセージ文字列を取得します。

トランザクション処理関数

lrt_tpabort	現在のトランザクションを中止します。
lrt_tpbegin	トランザクションを開始します。
lrt_tpcommit	現在のトランザクションをコミットします。
lrt_tpgetlev	トランザクションが進行中であるか調べます。
lrt_tpresume	グローバル・トランザクションを再開します。
lrt_tpscmt	lrt_tpcommit をいつ返すか設定します。
lrt_tpsuspend	グローバル・トランザクションを一時停止します。
lrt_tx_begin	グローバル・トランザクションを開始します。
lrt_tx_close	リソース・マネージャのセットを閉じます。
lrt_tx_commit	グローバル・トランザクションをコミットします。
lrt_tx_info	グローバル・トランザクションの情報を返します。
lrt_tx_open	リソース・マネージャのセットを開きます。
lrt_tx_rollback	グローバル・トランザクションをロールバックします。

lrt_tx_set_commit_return	commit_return 特性を when_return で指定された値に設定します。
lrt_tx_set_transaction_control	transaction_control 特性を control で指定された値に設定します。
lrt_tx_set_transaction_timeout	transaction_timeout 特性を timeout で指定された値に設定します。

相関ステートメント関数

lrt_display_buffer	ファイルにバッファ情報を格納します。
lrt_save[32]_fld_val	FML バッファの現在の値をパラメータに保存します。
lrt_save_parm	文字配列の一部（STRING または CARRAY バッファなど）をパラメータに保存します。
lrt_save_searched_string	バッファ内で文字列を検索し，文字列に関連するバッファの一部をパラメータに保存します。

注：一般に，**lrt_save_parm** を使用して文字配列の一部をパラメータに保存することをお勧めします。文字配列内の特定の文字列の位置に関する情報を保存する場合は，**lrt_save_searched_string** を使用します。PeopleSoft 仮想ユーザの場合，**lrt_save_searched_string** を使用することをお勧めします。PeopleSoft サーバから返される応答バッファは，記録時と再生時でサイズが異なることが多いためです。

Tuxedo 仮想ユーザ・スクリプトについて

セッションを記録した後に，記録されたコードを VuGen に組み込まれているエディタで表示できます。スクリプトをスクロールして，アプリケーションで生成された Tuxedo ステートメントの表示や，サーバによって返されたデータの検査ができます。VuGen ウィンドウには，記録された Tuxedo セッションに関する重要な情報が表示されます。メイン・ウインドウにスクリプトを表示すると，VuGen が動作状況を記録したシーケンスを確認できます。

次の例では，VuGen がクライアントのアクションを Tuxedo の銀行アプリケーションに記録しています。クライアントは，銀行の口座を開き，必要な詳細のすべてを指定するアクションを実行しました。クライアントが開始残高としてゼロを指定したところで，セッションは中止されています。

```
lrt_abort_on_error();
lr_think_time(65);
tpresult_int = lrt_tpbegin(30, 0);
data_0 = lrt_tpalloc("FML", "", 512);
lrt_Finitialize((FBFR*)data_0);

/* データ・バッファ data_0 に新規口座情報を入力する */
lrt_Fadd_fld((FBFR*)data_0, "name=BRANCH_ID", "value=8",
LRT_END_OF_PARMS);
lrt_Fadd_fld((FBFR*)data_0, "name=ACCT_TYPE", "value=C",
LRT_END_OF_PARMS);
lrt_Fadd_fld((FBFR*)data_0, "name=MID_INIT", "value=Q",
LRT_END_OF_PARMS);
lrt_Fadd_fld((FBFR*)data_0,"name=PHONE","value=123-456-7890",
            LRT_END_OF_PARMS);

lrt_Fadd_fld((FBFR*)data_0, "name=ADDRESS", "value=1 Broadway
            New York, NY 10000", LRT_END_OF_PARMS);
lrt_Fadd_fld((FBFR*)data_0,"name=SSN","value=111111111",
LRT_END_OF_PARMS);

lrt_Fadd_fld((FBFR*)data_0,"name=LAST_NAME",
"value=Doe",LRT_END_OF_PARMS);

lrt_Fadd_fld((FBFR*)data_0,"name=FIRST_NAME",
"value=BJ",LRT_END_OF_PARMS);

lrt_Fadd_fld((FBFR*)data_0, "name=SAMOUNT",
"value=0.00",LRT_END_OF_PARMS);

/* 新規口座を開く */
tpresult_int = lrt_tpcall("OPEN_ACCT", data_0, 0, &data_0, &olen_2, 0);
lrt_tpabort(0);
lrt_tpcommit(0);
lrt_tpfree(data_0);
lrt_tpterm();
```

Tuxedo スクリプトでのパラメータの使用

第 7 章 「パラメータの定義」の説明に従って，Tuxedo スクリプトのパラメータを定義できます。Tuxedo スクリプトには，「name=...」または「value=...」型の文字列が含まれます。パラメータの定義は，文字列内で等号（=）の後の部分でだけ行えます。たとえば，次のような場合です。

```
lrt_Fadd_fld((FBFR*)data_0,"name=PHONE","value= < parameter_1 > ",
                LRT_END_OF_PARMS);
```

Tuxedo スクリプトの実行

Tuxedo アプリケーションの記録または実行の際に問題が発生した場合は，Tuxedo アプリケーションが VuGen なしで動作し，環境変数が正しく定義されていることを確認します。詳細については，765 ページ「Tuxedo バッファ・データの表示」を参照してください。Tuxedo 変数の設定または変更をした後は，VuGen とアプリケーションを再起動して，変更を有効にする必要があります。アプリケーションが 16 ビットの場合は，NTVDM プロセスを強制終了する必要もあります。

実行時に問題が発生した場合は，サーバ側の Tuxedo ログ・ファイルでエラー・メッセージを確認します。標準設定では，このファイルは，環境変数 APPDIR で示されるディレクトリにあります。ファイル名は，ULOG.mmddyy の形式です（mmddyy は，現在の月，日，年を示してします）。1999 年 3 月 12 日のファイルは ULOG.031299 となります。このファイルの標準設定の場所は，サーバ上の環境変数 ULOGPFX を設定することで変更できます。ログ・ファイルはクライアント側にもあります。ULOGPFX 変数で場所が変更されていなければ，カレント・ディレクトリに置かれます。

Tuxedo バッファ・データの表示

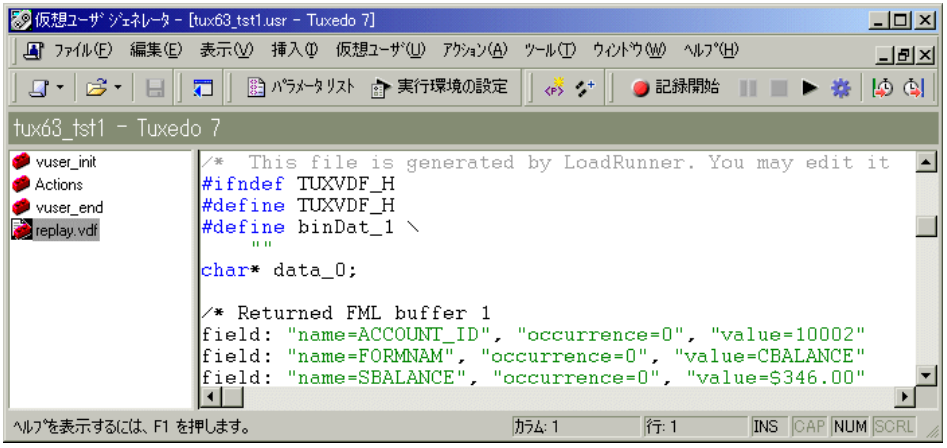
VuGen を使用して Tuxedo 仮想ユーザ・スクリプトを作成する場合，アクションはスクリプトの 3 つのセクション，**vuser_init**，**Actions**，および **vuser_end** に記録されます。

受け取った，または転送されたデータはデータ・バッファに保管されますが，データ・バッファは非常に大きくなることがあります。仮想ユーザ・スクリプトの可読性を高めるために，実際のデータは C ファイルではなく外部ファイルに保管されます。データ転送が発生すると，データは外部ファイルから一次バッファにコピーされます。

この外部ファイルは **replay.vdf** と呼ばれ，すべての一時バッファの内容を含んでいます。バッファの内容は連続したレコードとして保管されます。レコードはデータが送信されたか受信されたかを示す識別子とバッファ記述子によってマークされます。LRT 関数は，バッファ記述子を使用してデータにアクセスします。

VuGen を使用して，左側の表示枠のツリー・ビュー内で **replay.vdf** ファイルを選択することで，データ・ファイルの内容を表示できます。

Tuxedo スクリプトでは，データ・ファイルを表示するオプションを標準設定で利用できます。



Tuxedo 仮想ユーザの環境設定の定義

次の項では，Windows および UNIX プラットフォームで動作する Tuxedo 仮想ユーザのシステム変数の設定について説明します。システム変数は，NT の場合は［コントロールパネル］の［システム］ダイアログ・ボックスで，UNIX の場合は .cshrc または .login ファイルで定義します。

TUXDIR	Tuxedo ソースのルート・ディレクトリ
FLDTBLDIR	FML バッファ情報を含むディレクトリのリスト。Windows では，ディレクトリの名前をセミコロンで区切ります。UNIX プラットフォームでは，ディレクトリの名前をコロンで区切ります。
FIELDTBLS	FML バッファ情報を含むファイルのリスト。Windows と UNIX のどちらのプラットフォームでも，ファイル名はカンマで区切ります。

たとえば，次のような場合です。

```
SET FLDTBLDIR=%TUXDIR%¥udataobj;%TUXDIR%¥APPS¥WS (PC)
SET FIELDTBLS=bankflds,usysflds (PC)
setenv FLDTBLDIR $TUXDIR/udataobj:$TUXDIR/apps/bankapp (Unix)
setenv FIELDTBLS bank.flds,Usysflds (Unix)
```

実行中に，Tuxedo/WS ワークステーションの拡張を使用して，Tuxedo クライアントの次のシステム変数を定義する必要があります。

WSNADDR	ワークステーション・リスナ・プロセスのネットワーク・アドレスを指定します。これによって，クライアント・アプリケーションが Tuxedo にアクセスできるようになります。WSNADDR ステートメントで複数のアドレスを定義するには，各アドレスをカンマで区切る必要があります。
WSDEVICE	ネットワークにアクセスするデバイスを指定します。一部のネットワーク・プロトコルについては，この変数を定義する必要はありません。

たとえば，次のような場合です。

```
SET WSNADDR=0x0002ffffc7cb4e4a (PC)
setenv WSNADDR 0x0002ffffc7cb4e4a (Unix)
setenv WSDEVICE /dev/tcp (Unix)
```

Tuxedo アプリケーションのデバッグ

一般に，Tuxedo 6.x 以前を使用するアプリケーションを記録する場合は **Tuxedo 6** を，Tuxedo 7.1 を使用するアプリケーションを記録する場合は **Tuxedo 7** を使用します。

Tuxedo アプリケーションの記録または再生の際に問題が発生した場合，またはスクリプトが lrt_tpinitialize への呼び出しを行っていない場合は，アプリケーションでどの DLL が使用されているかをカスタマー・サポートに問い合わせてください。

アプリケーションが **libwsc.dll** ではなく **wtuxws32.dll** を使用している場合は，カスタマー・サポートから記録を行えるようにするパッチを入手します。

Tuxedo スクリプトの相関

VuGen は，Tuxedo アプリケーションで記録される仮想ユーザ・スクリプトの相関をサポートしています。相関ステートメントでバッファの一部を保存し，それを後続のステートメントで使用することにより，ステートメント間をリンクすることができます。

ステートメントを相関させるには，記録されたスクリプトを VuGen エディタ内で次の LRT 関数の 1 つを使用して変更する必要があります。

- **lrt_save[32]_fld_val** は，FML または FML32 バッファの現在の値（「name= < NAME >」または「id= < ID >」形式の文字列）をパラメータに保存します。
- **lrt_save_parm** は，文字配列の一部（STRING バッファや CARRAY バッファなど）をパラメータに保存します。
- **lrt_save_searched_string** は，バッファ内で文字列を検索し，その文字列に関連するバッファの一部をパラメータに保存します。

これらの関数の構文の詳細については，「**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。

FML および FML32 バッファの相関

lrt_save_fld_val または **lrt_save32_fld_val** を使って，FML バッファ，FML32 バッファの内容を保存します。

lrt_save_fld_val を使用したステートメントの相関は，次の手順で行います。

1 スクリプト内の，現在の FML（または FML32）バッファの内容を保存する場所に，**lrt_save_fld_val** ステートメントを挿入します。

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

2 パラメータを参照します。

保存したバッファの内容で記録されている値を置き換える **lrt** ステートメントを見つけます。記録されている値のすべてのインスタンスを，山括弧で囲まれたパラメータ名で置換します。

次の例では，銀行口座を開き，口座番号を **account_id** パラメータに格納します。

```
/* data_0 バッファに新規口座情報を入力する */
data_0 = lrt_tpalloc("FML", "", 512);
lrt_Finitialize((FBFR*)data_0);
lrt_Fadd_fld((FBFR*)data_0, "name=BRANCH_ID", "value=1",
LRT_END_OF_PARMS);
lrt_Fadd_fld((FBFR*)data_0, "name=ACCT_TYPE", "value=S",
LRT_END_OF_PARMS);
. . .

LRT_END_OF_PARMS);
lrt_Fadd_fld((FBFR*)data_0, "name=LAST_NAME", "value=Doe", ...);
lrt_Fadd_fld((FBFR*)data_0, "name=FIRST_NAME", "value=John", ...);
lrt_Fadd_fld((FBFR*)data_0, "name=SAMOUNT", "value=234.12", ...);

/* 新規口座を開き，新規口座番号を保存する */
tpresult_int = lrt_tpcall("OPEN_ACCT", data_0, 0,&data_0, &olen_2, 0);
lrt_abort_on_error();
lrt_save_fld_val((FBFR*)data_0, "name=ACCOUNT_ID", 0, "account_id");

/* 最初のクエリの結果を預金バッファに入力する */
lrt_Finitialize((FBFR*)data_0);
lrt_Fadd_fld((FBFR*)data_0, "name=ACCOUNT_ID", "value= < account_id >
", LRT_END_OF_PARMS);
lrt_Fadd_fld((FBFR*)data_0, "name=SAMOUNT", "value=200.11", ...);
```

上の例では，アカウント ID が ACCOUNT_ID というフィールド名で表されています。記録時にフィールドがフィールド名ではなく ID 番号で表されるシステムもあります。

フィールド ID による相関は，次の手順で行います。

```
lrt_save_fld_val((FBFR*)data_0, "id=8302", 0, "account_id");
```

文字列の相関

lrt_save_parm または **lrt_save_searched_string** を使って，キャラクタ文字列を相関させます。

- 一般に，**lrt_save_parm** を使用して文字配列の一部をパラメータに保存することをお勧めします。
- 文字配列内の特定の文字列の位置に関する情報を保存する場合は，**lrt_save_searched_string** を使用します。仮想ユーザが PeopleSoft 用の場合，**lrt_save_searched_string** を使用することをお勧めします。PeopleSoft サーバから返される応答バッファは，記録時と再生時でサイズが異なることが多いためです。

相関させる値の決定

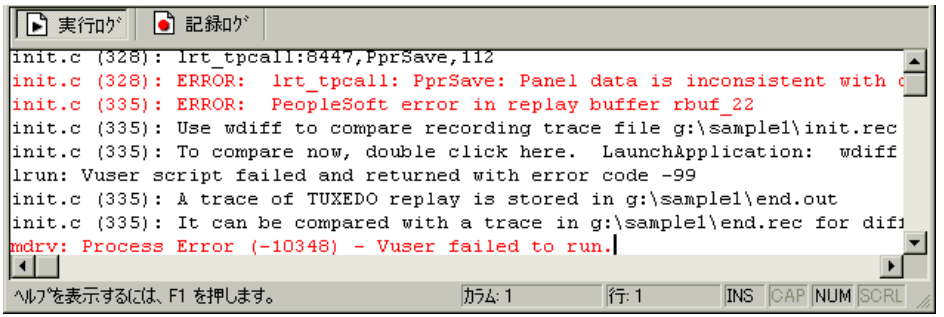
CARRAY バッファで作業を行っている場合，VuGen は **wdiff** ユーティリティを使って比較できるログ・ファイルを生成します（記録中であれば拡張子は **.rec** に，再生中であれば拡張子は **.out** になります）。記録ログと再生ログの相違を調べて，CARRAY バッファのどの部分を相関させる必要があるか判断できます。

ログ・ファイルの比較は，次の手順で行います。

1 ［**表示**］＞［**出力ウィンドウ**］を選択して，スクリプトの実行ログと記録ログを表示します。

2 実行ログを検査します。

エラー・メッセージの後には，**Use wdiff to compare** という句で始まるステートメントが付いています。



3 実行ログのステートメントをダブルクリックして，**wdiff** ユーティリティを起動します。

WDiff が開き，記録ファイルと再生ファイルの間の相違が黄色で強調表示されます。Wdiff ユーティリティの詳細については，第 8 章 「ステートメントの相関」を参照してください。

lrt_save_parm を使用したステートメントの相関は，次の手順で行います。

相関させる値を決定したら，**lrt_save_parm** を使って，文字配列の一部分（STRING または CARRAY バッファなど）をパラメータに保存できます。

1 スクリプト内の，現在のバッファの内容を保存する場所に **lrt_save_parm** ステートメントを挿入します。

lrt_save_parm (buffer, offset, length, "**param_name**");

2 **replay.vdf** ファイル内で，保存されたバッファの内容で置き換えたいデータを見つけます。

VuGen のメイン・ウィンドウの［データ ファイル］ボックスにある **replay.vdf** ファイルを選択して，バッファの内容を表示します。

3 値のすべてのインスタンスを山括弧で囲まれたパラメータ名で置換します。

次の例では，CARRAY バッファの従業員 ID を，後で使用できるように保存する必要があります。出力に表示されているように，記録された値は「G001」です。

```
lrt_tpcall:227, PprLoad, 1782
Reply Buffer received.
...
123 “G001”
126 “...”
134 “Claudia”
```

オフセット 123 を使用して，「PprLoad」と 227 バイトを送信する要求バッファの直後に **lrt_save_parm** を挿入します。

```
/* CARRAY buffer 57 を要求する */
      lrt_memcpy(data_0, buf_143, 227);
      tpresult_int = lrt_tpcall("PprLoad",
           data_0, 227, &data_1, &olen, TPSIGRSTRT);
      lrt_save_parm(data_1, 123, 9, "empid");
```

replayvdf ファイルで，記録された値「G001」をパラメータ **empid** で置換します。

```
char buf_143[] =
"¥xf5¥x0¥x0¥x0¥x4¥x3¥x2¥x1¥x1¥x0¥x0¥x0¥xbc¥x2¥x0¥x0¥x0¥x0¥x0¥x0"
   "X"
"¥x89¥x0¥x0¥x0¥xb¥x0"
   "SPprLoadReq"
   "¥xff¥x0¥x10¥x0¥x0¥x4¥x3¥x6"
" < empid > " // G001
   "¥x7"
“Claudia”
   "¥xe"
   "LAST_NAME_SRCH"
...
```

この関数は，FML バッファ内で文字配列の一部を保存するときにも使用できます。次の例では，電話番号が文字配列で，市外局番は最初の 3 文字です。はじめに，**lrt_save_fld_val** ステートメントが電話番号をパラメータ **phone_num** に保存します。**lrt_save_parm** ステートメントは，**lr_eval_string** を使って電話番号を文字配列に変換し，市外局番を **area_code** という名前のパラメータに保存します。

```
lrt_save_fld_val((FBFR*)data_0, "name=PHONE", 0, "phone_num");
lrt_save_parm(lr_eval_string(" < phone_num > "), 0, 3, "area_code");
lr_log_message("The area code is %s¥n", lr_eval_string(" < area_code > "));
```

lrt_save_searched_string を使用したステートメントの相関は，次の手順で行います。

lrt_save_searched_string を使用して，バッファ内で文字列を検索し，文字列に関連するバッファの一部をパラメータに保存します。

1 スクリプト内の現在のバッファの一部を保存する場所に **lrt_save_searched_string** ステートメントを挿入します。

lrt_save_searched_string (buffer, buf_size, occurrence, string, offset,length, "**param_name**");

offset は文字列の先頭からのオフセットです。

2 **replay.vdf** ファイル内で，保存されたバッファの内容で置き換えたいデータを見つけます。

VuGen のメイン・ウィンドウの［データ ファイル］ボックスにある **replay.vdf** ファイルを選択して，バッファの内容を表示します。

3 値のすべてのインスタンスを山括弧で囲まれたパラメータ名で置換します。

次の例では，Certificate を後で使用できるようにパラメータに保存しています。**lrt_save_searched_string** 関数は，指定された olen バッファの 16 バイトをパラメータ **cert1** に保存します。文字列がバッファ内で保存される場所は，文字列「SCertRep」の最初の出現より 9 バイト後です。

このアプリケーションは，バッファのヘッダー情報が記録環境によって異なる場合に役立ちます。

署名は必ず「SCertRep」の最初の出現より 9 バイト後に来ますが，この文字列より前の情報の長さは不定です。

```
/* CARRAY buffer 1 を要求する */
lrt_memcpy(data_0, sbuf_1, 41);
lrt_display_buffer("sbuf_1", data_0, 41, 41);
data_1 = lrt_tpalloc("CARRAY", "", 8192);
tpresult_int = lrt_tpcall("GetCertificate",
      data_0,
      41,
      &data_1,
      &olen,
      TPSIGRSTRT);

/* CARRAY buffer 1 を再生する */
lrt_display_buffer("rbuf_1", data_1, olen, 51);
lrt_abort_on_error();

lrt_save_searched_string(data_1, olen, 0, "SCertRep", 9, 16, "cert1");
```

第 13 部

ストリーミング・データ・プロトコル

第 55 章

ストリーミング・データ仮想ユーザ・スクリプトの作成

インターネットで音声 / 映像コンテンツを配信するストリーミング・メディアが急成長しています。ストリーミング・メディアの基本的な考え方は，音声 / 映像コンテンツを配信するときに，エンド・ユーザに先にファイル全体をダウンロードする手間をかけさせないようにしようというものです。ストリーミングは，サーバからコンテンツをストリームとして連続して送出させ，それをクライアントが受け取るごとに表示する仕組みになっています。

RealPlayer と Media Player は，ストリーミング・コンテンツを表示するアプリケーションです。

RealPlayer または Media Player プロトコルを使用するクライアント・アプリケーションとサーバ間の通信を VuGen で記録します。これによって生成されるスクリプトを，RealPlayer または Media Player 仮想ユーザ・スクリプトと呼びます。

本章では，次の項目について説明します。

- ストリーミング・データ仮想ユーザ・スクリプトの概要
- RealPlayer LREAL 関数の使用
- Media Player MMS 関数の使用

以降の情報は，RealPlayer および Media Player プロトコルを対象とします。

ストリーミング・データ仮想ユーザ・スクリプトの記録について

ストリーミング・データ・プロトコルにより，メディアまたはストリーミング・データ・ファイルを再生するユーザをエミュレートできます。

ストリーミング・データ・プロトコルを使用してアプリケーションを記録すると，VuGen は記録時のアクションを記述する関数を生成します。RealPlayer セッションの場合，VuGen は接頭辞 **lreal** の付いた関数を生成します。Media Player セッションの場合，VuGen は接頭辞 **mms** の付いた関数を使用します。Media Player 用の mms 関数では記録はサポートされていません。再生だけです。

ストリーミング・データ仮想ユーザ・スクリプトの概要

本項では，VuGen を使用して RealPlayer および Media Player のストリーミング・データ仮想ユーザ・スクリプトを作成する工程の概要を説明します。

RealPlayer または Media Player 仮想ユーザ・スクリプトの作成は，次の手順で行います。

1 **VuGen を使用して，基本スクリプトを記録します。（RealPlayer のみ）**

RealPlayer の場合，VuGen を起動し，新しい Virtual Player スクリプトを作成します。記録するアプリケーションを選択します。選択したアプリケーションの典型的な操作を記録します。詳細については，第 3 章 「VuGen を使った記録」を参照してください。

Media Player では，記録はサポートされていません。空の Media Player スクリプトを作成し，**mms** 関数を手作業で挿入します。使用例は，「**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。

2 **スクリプトを拡張します。**

スクリプトに，トランザクション，ランデブー・ポイント，および制御フロー構造を挿入して，スクリプトを拡張します。

詳細については，第 6 章 「仮想ユーザ・スクリプトの拡張」を参照してください。

3 **パラメータを定義します（任意）。**

スクリプトに記録された固定値に対するパラメータを定義します。固定値をパラメータに置き換えることにより，異なる値を使って同じビジネス・プロセスを何度も繰り返すことができます。

詳細については，第 7 章 「パラメータの定義」を参照してください。

4 **ステートメントを相関させます（任意）。**

ステートメントを相関することにより，あるビジネス・プロセスの結果を後続のビジネス・プロセスで使用できます。

詳細については，第 8 章 「ステートメントの相関」を参照してください。

5 **実行環境を設定します。**

実行環境の設定により，スクリプト実行中の Virtual Player/ 仮想ユーザの動作が制御されます。設定には，ループ，ログ，タイミング情報があります。

詳細については，第 9 章 「実行環境の設定」を参照してください。

6 **VuGen でスクリプトを実行します。**

VuGen で生成したスクリプトを保存して実行し，正しく動作することを確かめます。

詳細については，第 11 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。

作成した仮想ユーザ・スクリプトは，Windows または UNIX プラットフォームのシナリオに統合します。仮想ユーザ・スクリプトのシナリオへの統合については，『**LoadRunner コントローラ・ユーザーズ・ガイド**』を参照してください。

RealPlayer LREAL 関数の使用

RealPlayer プロトコルを使用してクライアントとサーバ間の通信をエミュレートするために作成された関数を，Real Player 関数と呼びます。各 Real Player 関数には，接頭辞 **lreal** が付いています。VuGen は，Real Player セッション中，本項に列挙されている LREAL 関数のほぼすべてを自動的に記録します。スクリプトに任意の関数を手作業でプログラミングすることもできます。LREAL 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。

lreal_clip_size	現在のクリップのサイズを返します。
lreal_close_player	RealPlayer のインスタンスを閉じます。
lreal_open_player	RealPlayer のインスタンスを新規作成します。
lreal_open_url	URL を開きます。
lreal_pause	RealPlayer クリップの再生を一時停止します。
lreal_play	RealPlayer クリップを再生します。
lreal_seek	RealPlayer クリップ内の位置を検索します。
lreal_stop	RealPlayer クリップの再生を停止します。

lreal_play 関数の形式の例を次に示します。

int **lreal_play** (int miplayerID, long **mulTimeToPlay**);

クリップを最後まで再生するには，**mulTimeToPlay** 値に任意の負の値を使用します。クリップの再生を指定した時間だけ継続する場合は，時間をミリ秒単位で指定します。**miplayerID** は，RealPlayer インスタンスの一意の ID を表示します。

Media Player MMS 関数の使用

Media Player の MMS プロトコルを使用してクライアントとサーバ間の通信をエミュレートするために作成された関数を，MMS 仮想ユーザ関数と呼びます。各 MMS 仮想ユーザ関数には，接頭辞 **mms** が付いています。

MMS 関数はすべて，グローバル・セッションと特定のセッション用にペアになっています。たとえば，**mms_close** は Media Player をグローバルに閉じ，**mms_close_ex** は特定のセッションでの Media Player を閉じます。

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。

関数名	説明
mms_close[_ex]	Media Player を閉じます。
mms_get_property[_ex]	Media Player クリップのプロパティを取得します。
mms_isactive[_ex]	Media Player がアクティブであることを検証します。
mms_pause[_ex]	Media Player クリップの再生を一時停止します。
mms_play[_ex]	Media Player クリップを再生します。
mms_resume[_ex]	Media Player クリップの再生を再開します。
mms_sampling[_ex]	Media Player クリップをサンプリングします。
mms_set_property[_ex]	Media Player クリップのプロパティを設定します。
mms_set_timeout[_ex]	Media Player クリップのタイムアウト値を設定します。
mms_stop[_ex]	Media Player クリップの再生を停止します。

たとえば，**mms_play** 関数は，次の形式になります。

int **mms_play** (char message, ＜属性のリスト＞ , LAST);

次の例では，**mms_play** 関数は，**asf** ファイルを継続時間を変えて再生します。

```
// 10 秒再生する
mms_play("Welcome","URL=mms://server/welcome.asf","duration=10",LA
ST);

// 5 秒待機後，クリップを最後まで再生する
mms_play ("Welcome","URL=mms://server/welcome.asf",
"duration=-1",
"starttime=5",LAST);
```

第 14 部

ワイヤレス・プロトコル

第 56 章

ワイヤレス仮想ユーザの紹介

VuGen では，WAP，VoiceXML，または i モードのプロトコルを使用するワイヤレス・アプリケーション用のスクリプトを作成できます。VuGen では，ワイヤレス・ネットワーク上のユーザ・アクションを記録することにより，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。

- WAP プロトコルについて
- i モード・システムについて
- i モードと WAP の比較
- VoiceXML について

ワイヤレス仮想ユーザについて

VuGen では，以下の 3 つのワイヤレス・プロトコルがサポートされています。

- WAP（Wireless Application Protocol）
- i モード
- VoiceXML

各プロトコルにはそれぞれの特徴があり，コンテンツの実装方法および開発方法が異なります。

開発者は，ワイヤレス・プロトコル用のコンテンツおよびアプリケーションの開発環境を提供するツールキットを使用します。

WAP プロトコルについて

WAP（Wireless Application Protocol）は，モバイル・ユーザがワイヤレス・デバイスを使って瞬時に情報およびサービスにアクセスすることを可能にする，世界標準のオープンな規格です。

WAP プロトコルは，ワイヤレス・モバイル端末用に最適化された WML と呼ばれる新しい標準言語を使い，マイクロ・ブラウザによるシン・クライアントを規定しています。WML とは，XML を必要最小限まで簡素化した文書記述言語です。

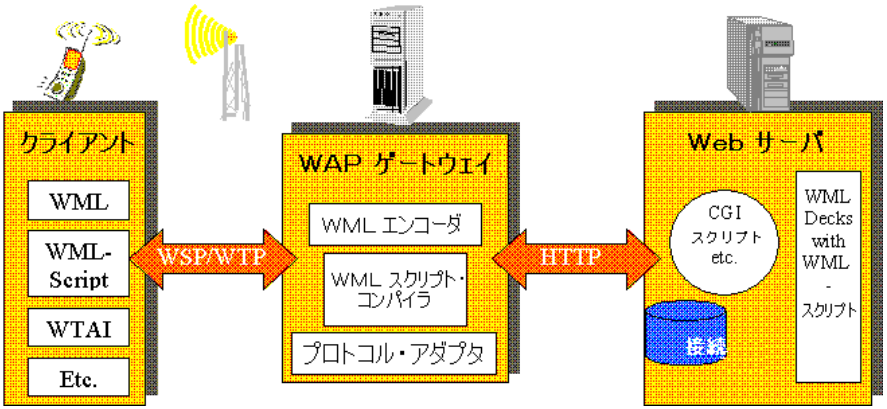
WAP ではさらに，以下の条件を満たすプロキシ・サーバを規定しています。

- ワイヤレス・ネットワークと有線のインターネットの間のゲートウェイとして機能する。
- プロトコル変換機能がある。
- ワイヤレス受信のためにデータ転送を最適化する。

WAP アーキテクチャは WWW と非常によく似ています。すべてのコンテンツがインターネットの標準形式に似た形式で記述されます。コンテンツは WWW の領域では標準プロトコルで，ワイヤレスの領域（Wireless Session Protocol）では HTTP に似た最適化されたプロトコルを使って送信されます。WAP コンテンツはすべて WWW で標準的に使われている URL を使って指定します。

WAP では，オーサリングやパブリッシングの方法など，多くの部分が WWW 規格を使用しています。その一方で，ワイヤレス・デバイスおよびネットワークの特徴に合わせて，いくつかの WWW 規格が強化されています。「**呼制御**」および「**メッセージング**」などのモバイル・ネットワーク・サービスをサポートするための拡張機能が追加されています。WAP は，モバイル端末のメモリ容量や CPU 処理能力の制約を考慮しています。また，帯域幅の狭いネットワークおよび遅延時間の長いネットワークにも対応します。

WAP では，モバイル・クライアントとの間で送受信されるデータのエンコードとデコードを行うゲートウェイが存在することが前提となっています。クライアントに配信されるコンテンツをエンコードする目的は，クライアントにワイヤレスで送信されるデータのサイズを最小化することと，クライアントがデータを処理する際の負荷を軽減することです。このようなゲートウェイの機能は発信元サーバに追加することも可能ですが，次の図に示すように専用ゲートウェイに置くこともできます。



WAP ツールキット

Nokia，Ericsson，Phone.com などの主要通信企業は，WAP アプリケーションおよびサービスの開発を支援する「**ツールキット**」を開発しています。この WAP ツールキットは，モバイル端末用のインターネット・サービスおよびコンテンツの開発環境を提供します。開発者は，WAP ツールキットを使用して，PC ベースの電話シミュレータによるアプリケーションの開発，テスト，デバッグ，および実行ができます。また，ツールキットから HTTP 接続または WAP ゲートウェイを経由して WAP サイトをブラウズすることができます。

携帯電話からは，WSP プロトコルを使ってゲートウェイと通信します。一方，ツールキットはゲートウェイと通信することも，サーバと直接通信することもできます。VuGen を使って記録する場合，WSP と HTTP の 2 つのモードが用意されています。ゲートウェイへのトラフィックを調べたい場合は，WSP モードで記録します。サーバおよびコンテンツ・プロバイダを検査したい場合は，HTTP モードでツールキット・セッションを記録して，ゲートウェイはバイパスすることができます。

VuGen は，ユーザ・セッションをエミュレートするためにユーザ定義の API 関数を使用します。ほとんどの関数は HTTP プロトコルを使用した標準の Web プロトコル関数です。いくつかの WAP 関数では，WAP 仮想ユーザ固有のアクションをエミュレートします。サポートされている関数の一覧については，796 ページ「ワイヤレス仮想ユーザ関数の使用法」を参照してください。

i モード・システムについて

i モード・プロトコルは，NTT ドコモのモバイル・インターネット・アクセス・システムです。技術的に言えば，i モードは通常のモバイル音声システムのオーバーレイ技術です。音声システムは**回線交換方式**を使用しています（つまり，ダイアルアップする必要があります）が，i モードは**パケット交換方式**を使用しています。つまり，i モードでは送受信可能圏内にいる限り，常に接続されている状態にあります。一般に携帯電話で i モード・メニューの項目を選択すると，データが即座にダウンロードされ，ダイアルアップの処理による遅延は必要ありません。ただし，データのサイズやネットワークの帯域幅により，データ受信時に遅延が生じるかもしれません。

i モードを使って行う作業は，ブラウザでインターネットにアクセスする場合とほとんど同じです。たとえば，メールを送信したり，天気予報やスポーツの結果を見たり，ゲームをしたり，オンラインで株取引を行ったり，航空券を購入したり，レストランを検索したりすることができます。

i モード・プロトコルでは，通常の HTML のサブセットである **cHTML**（コンパクト HTML）を使用します。cHTML には，標準の HTML タグのほかに，i モード固有のタグがいくつかあります。たとえば，ある i モード・タグでは特定の電話番号に電話をかけるリンクを設定できます。また，Web ページが i モードのページであることを検索エンジンに知らせる別の i モード固有のタグもあります。

ほかにも特殊記号などドコモ固有の文字が数多くあります。たとえば，喜び，愛情，悲しみ，電話，電車，丸で囲まれた数字などを示す特殊文字があります。

cHTML は HTML のサブセットなので，Netscape または IE ブラウザを使って，http://www.eurotechnology.com/i/ や http://www.eu-japan.com/i/ などの i モードのページを表示できます。ただし，ほとんどの i モード・ユーザは日本人なので，i モードのコンテンツのほとんどは日本語です。したがって，ブラウザには日本語のテキスト表示をサポートする機能が必要です。通常のブラウザで i モードのコンテンツを表示すると，i モード固有のタグは見ることができません。また，ドコモの i モード固有の特殊記号を表示することもできません。

i モード・ツールキット

i モード・サービスの開発を支援するために，いくつかのツールキットが提供されており，VuGen でもサポートしています。i モード用のツールキットは，モバイル端末用のインターネット・サービスおよびのコンテンツの開発環境を提供します。開発者は，i モード・ツールキットの PC ベースの電話のシミュレータを使って，アプリケーションの作成，テスト，デバッグ，および実行ができます。また，ツールキットから標準の HTTP 接続を経由して i モード・サイトをブラウズすることができます。サポートされているツールキットには，CompactViewer や，Pixo 2.0 および 2.1 などがあります。

i モードと WAP の比較

i モードのサービスと WAP のサービスは，実装方法においていくつかの大きな違いがあります。i モードは，HTML のサブセットである cHTML を使用します。cHTML は，WAP のマークアップ言語である WML よりは比較的簡単に習得できます。現在，i モードはパケット交換方式を使用しており，原則として常に接続状態にあります。一方，WAP システムは回線交換方式（すなわちダイアルアップ）を使用します。パケット交換方式と回線交換方式とでは，それぞれのサービスのベースとなっている通信システムが技術的に異なります。原則として，i モードの Web ページと WAP でエンコードされた Web ページは，パケット交換方式でも回線交換方式でも配信できます。

また，両者の料金システムも異なります。i モード・ユーザはダウンロードした情報量と，各種付加サービスに応じて料金が決まりますが，WAP ユーザは接続時間に応じて料金が決まります。

VoiceXML について

VoiceXML または VXML は，音声ブラウザまたは電話の音声認識技術を通じてインターネットと対話するための技術です。VoiceXML を使用すると，前もって録音された音声や，コンピュータによって生成された合成音声を聞きながら音声ブラウザと対話し，電話などを通して自然会話音声またはキーパッドからデータを入力できます。

VoiceXML は，Web 上の静的または動的な VoiceXML コンテンツにアクセスする VoiceXML ゲートウェイから成ります。ゲートウェイには，VoiceXML ブラウザ，Text-To-Speech，自動音声認識（ASR），および公衆電話交換網（PSTN）に接続するテレフォニー機器があります。T1，POTS，または ISDN のいずれかを介して電話網に接続します。一般住宅で使用されるものに類似する一般電話サーバ（POTS）回線は，単一接続のみを処理します。それに対し，T1 回線は 24 本の独立した電話回線で構成されます。

典型的な音声対話は次のとおりです。

1. 電話（ワイヤレスまたは固定）でシステムにダイアルアップします。テレフォニー・ハードウェアが呼び出しを受け取り，VoiceXML ブラウザに渡します。
2. VoiceXML ゲートウェイは，指定された Web サーバから **vxml** 拡張子の付いた VoiceXML 文書を取得し，プロンプト・トーンを発します。
3. ユーザは電話口で応答するか，電話のキーパッドで入力をします。
4. テレフォニー機器は，VoiceXML 文書に含まれている定義済みの辞書を使用して録音された音を（会話の場合）音声認識エンジンに転送します。
5. VoiceXML ブラウザは，音声分析の結果に応じて文書内のコマンドを実行し，別のプロンプト・トーンを発します。プロンプト・トーンは，録音済みのものか，合成されたものです。

VuGen では，VoiceXML セッションの記録がサポートされています。記録されたスクリプトには，ユーザのアクションをエミュレートする **web_url** 関数が含まれています。次の例では，ユーザが株情報のページを要求しています。

```
Action1()
{
	web_add_auto_header("Accept",
		"text/x-vxml, */*");

	web_add_auto_header("Content-Type",
		"application/x-www-form-urlencoded");

	web_add_auto_header("User-Agent",
		"Motorola VoxGateway/2.0");

	web_url("top.vxml",
		"URL=http://testserver1/Vxmlexample/top.vxml?DNIS=-",
		"Resource=0",
		"RecContentType=application/octet-stream",
		"Referer=",
		"Mode=HTTP",
		LAST);

	web_url("stock.vxml",
		"URL=http://testserver1/Vxmlexample/stock.vxml",
		"Resource=0",
		"RecContentType=application/octet-stream",
		"Referer=",
		"Mode=HTTP",
		LAST);
return 0;
}
```

第 57 章

ワイヤレス仮想ユーザ・スクリプトの記録

VuGen を使用して標準的なワイヤレス・セッションを記録することによって，ワイヤレス・仮想ユーザ・スクリプトを生成できます。スクリプトを実行すると，生成された仮想ユーザによって，ツールキットまたは携帯電話と Web サーバ（または WAP 用ゲートウェイ）との間のユーザ・アクションがエミュレートされます。

本章では，次の項目について説明します。

- ワイヤレス仮想ユーザ・スクリプトの開発の概要
- ワイヤレス仮想ユーザ関数の使用法
- ワイヤレス仮想ユーザ・スクリプトのトラブルシューティング

以降の情報は，WAP，i モード，VoiceXML の全ワイヤレス・プロトコルを対象とします。

ワイヤレス仮想ユーザ・スクリプトの記録について

顧客の購入要求の状況を表示する Web サイトがあったとします。このサイトでは，多数のユーザ（200 ユーザなど）が同時にサイトにアクセスしたときでも，顧客の問い合わせに対する応答時間が必ず指定値（20 秒など）未満になるようにしたいとします。そのために，仮想ユーザを使ってこの Web または WAP サーバが同時に要求された情報へのサービスを提供するシナリオをエミュレートします。その際，各仮想ユーザは以下の操作を行うと考えられます。

- 最初のページのロード
- 要求の送信
- サーバからの応答の待機

テストに利用できるマシンに，数百の仮想ユーザを分散配置できます。各仮想ユーザは，サーバの API を使用してサーバにアクセスします。このようにして，多数のユーザによる負荷がかかった状態でのサーバのパフォーマンスを測定できます。

ワイヤレス仮想ユーザ・スクリプトの開発の概要

本項では，VuGen を使ったワイヤレス仮想ユーザ・スクリプトの開発プロセスの概要を説明します。

ワイヤレス・スクリプトを作成するには，次の手順で行います。

1 **VuGen を使用して新しいスクリプトを作成します。**

［**ファイル**］＞［**新規作成**］を選択するか，［**新規作成**］ボタンをクリックして，シングル・プロトコル・モードまたはマルチ・プロトコル・モードで新しいスクリプトを作成します。

新規スクリプトの作成の詳細については，第 3 章 「VuGen を使った記録」を参照してください。

2 **記録オプションを設定します。**

記録オプションを設定します。インターネット・プロトコルの一般記録オプションの設定については，第 35 章 「インターネット・プロトコルの記録オプションの設定」を，ワイヤレス記録オプションについては，第 59 章 「ワイヤレス仮想ユーザの記録オプションの設定」を参照してください。

3 **VuGen を使ってアクションを記録します。**

ツールキット・セッションでのアクションを記録します。

記録方法の詳細については，第 3 章 「VuGen を使った記録」を参照してください。

4 **仮想ユーザ・スクリプトを拡張します。**

仮想ユーザ・スクリプトにトランザクション，ランデブー・ポイント，および制御フロー構造を挿入して，スクリプトを拡張します。

詳細については，第 6 章 「仮想ユーザ・スクリプトの拡張」を参照してください。

5 **パラメータを定義します（任意）。**

仮想ユーザ・スクリプトに記録された固定値に対してパラメータを定義します。固定値をパラメータで置き換えることにより，同じビジネス・プロセスを，異なる値で何度も繰り返すことができます。

詳細については，第 7 章 「パラメータの定義」を参照してください。

6 **実行環境を設定します。**

実行環境を設定することによって，スクリプト実行時の仮想ユーザの振る舞いを制御します。この設定には，実行論理，ペースの設定，ログ，思考遅延時間，その他の情報が含まれます。

一般的な実行環境の設定の詳細については，第 9 章 「実行環境の設定」を参照してください。

インターネット・プロトコルの一般的な実行環境設定の詳細については，第 37 章 「インターネット実行環境の設定」を参照してください。

実行環境の設定の WAP 専用の設定の詳細については，第 60 章 「WAP 実行環境の設定」を参照してください。

7 **相関を実行します。**

スクリプトを検査して，相関の必要な動的な値があるかどうか確認します。ワイヤレス・プロトコルでは，**web_reg_save_param** 関数を追加して手作業による相関を実行します。

詳細については，575 ページ「手作業による相関」を参照してください。

8 **VuGen で仮想ユーザ・スクリプトを保存して実行します。**

VuGen で生成した仮想ユーザ・スクリプトを保存して実行し，正しく動作することを確認します。記録中，VuGen によって一連の設定ファイル，データ・ファイル，ソース・コード・ファイルが生成されます。これらのファイルには，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。

仮想ユーザ・スクリプトをスタンドアロン・テストとして実行する方法の詳細については，第 11 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。

ワイヤレス仮想ユーザ・スクリプトを作成し終わったら，シナリオに組み込みます。詳細については，『**LoadRunner コントローラ・ユーザーズ・ガイド**』を参照してください。

ワイヤレス仮想ユーザ関数の使用法

ワイヤレス・デバイスと Web サーバ（または WAP 用ゲートウェイ）の間の通信をエミュレートするために開発された関数を仮想ユーザ関数といいます。関数の中には，スクリプトの記録時に生成されるものと，手作業でスクリプトに挿入しなければならないものがあります。また記録の後で，LoadRunner メッセージ関数とユーザ定義の C 関数を，仮想ユーザ・スクリプトに追加することもできます。

標準的な HTTP のアクションを表す関数の名前には，**web** という接頭辞が付いています。これらの関数の詳細については，第 33 章 「Web 仮想ユーザ関数の使用」を参照してください。

一般的な仮想ユーザ関数の名前には，**lr** という接頭辞が付いています。詳細については，22 ページ「C 仮想ユーザ関数の使用方法」を参照してください。

次の項では，WAP 固有のアクションを表す関数について説明します。これらの関数の名前には，**wap** という接頭辞が付いています。

Web 関連のすべての関数の一覧については，第 33 章「Web 仮想ユーザ関数の使用」，または「**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。

WSP（Wireless Session Protocol）モードでスクリプトを実行する WAP 仮想ユーザでは，次の関数がサポートされています。

アクション関数：	**web_custom_request**，**web_submit_data**，**web_url**
認証関数：	すべて——**web_set_user**，**web_set_certificate**[_ex]
クッキー関数：	すべて——**web_add_cookie**，**web_cleanup_cookie**，**web_remove_cookie**
ヘッダー関数：	すべて——**web_add_auto_header**，**web_add_header**，**web_cleanup_auto_headers**，**web_save_header**
相関関数：	すべて——**web_create_html_param**[_ex]，**web_reg_save_param**，**web_set_max_html_param_len**

次の項に，WAP 固有の関数の一覧を示します。

WAP 固有の関数

wap_add_const_header	WAP ゲートウェイに渡す定数ヘッダーを指定します。
wap_connect	WAP ゲートウェイに接続します。
wap_disconnect	WAP ゲートウェイとの接続を解除します。
wap_format_si_message	SI タイプのメッセージを組み立てます。
wap_format_sl_message	SL タイプのメッセージを組み立てます。
wap_mms_msg_add_field	MMSC メッセージにフィールドを追加します。
wap_mms_msg_add_mulitpart_entry	MMSC メッセージにマルチパートのエントリを追加します。
wap_mms_msg_create	MMSC 用のメッセージを作成します。
wap_mms_msg_destroy	MMSC 用のメッセージを破棄します。
wap_mms_msg_submit	MMSC にメッセージを送信します。
wap_pi_push_cancel	PPG に送信したメッセージをキャンセルします。
wap_pi_push_submit	プッシュ・メッセージを送信します。
wap_radius_connection	RADIUS サーバに接続したり，サーバとの接続を解除したりします。
wap_send_sms	SMS タイプのメッセージを送信します。
wap_set_bearer	UDP ベアラまたは CIMD2（SMS）ベアラを設定します。
wap_set_capability	クライアントの WAP ゲートウェイ接続機能を設定します。
wap_set_connection_mode	接続モードおよびセキュリテイ・レベルを設定します。
wap_set_gateway	ゲートウェイの IP アドレスとポートを設定します。
wap_set_sms_user	SMSC に対するログイン情報を設定します。
wap_wait_for_push	プッシュ・メッセージの到着を待機します。

詳細については，VuGen エディタで関数を選択して F1 キーを押すか，「**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。

ワイヤレス仮想ユーザ・スクリプトのトラブルシューティング

Nokia ツールキット

Nokia ツールキット（3.0 および 3.1）には，正しい IP アドレスを割り当てて手作業で起動する必要があります。

ツールキットを再起動するには，次の手順で行います。

1 VuGen マシン上でゲートウェイが実行されていないことを確認します。別のゲートウェイが動作していると，疑似ゲートウェイのポートがブロックされます。

2 VuGen を起動します。

3 ツールキットを呼び出します（ツールキットは必ず VuGen を起動した**後**に起動してください）。

4 [記録オプション] で，[**インターネット プロトコル：WAP ツールキット**] ノードを選択し，[**WAP ツールキットを手動で起動する**] を選択します。

5 [**OK**] をクリックします。VuGen により割り当てられたゲートウェイ IP アドレスを示すメッセージ・ボックスが開きます。

6 この IP アドレスをコピーし，ツールキットの接続文字列に貼り付けます。

7 ツールキットに使用する URL を入力します。**ゲートウェイが接続されていない**ことを知らせるメッセージを無視します。URL を再度入力します。VuGen では，この時点までにいくつかの **web_add_const_header** イベントが記録されている場合があります。

新しい記録セッションを開始するたびに，ツールキットを終了し，手順 2 から 7 を繰り返します。

注：他のツールキットでも，記録に関する問題が生じた場合には，上記の手順を使用できます。

第 58 章

WAP 仮想ユーザ・スクリプトでの作業

VuGen を使用して，WAP（ワイヤレス・アプリケーション・プロトコル）仮想ユーザ・スクリプトを作成します。VuGen は，WAP デバイス操作時のユーザのアクションを記録して，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。

- 携帯電話での記録
- ベアラのサポート
- RADIUS のサポート
- プッシュのサポート
- LoadRunner でのプッシュのサポート
- MMS のサポート

WAP 仮想ユーザについて

Wireless Application Protocol（WAP）は，モバイル・ユーザがワイヤレス・デバイスを使って情報およびサービスに即座にアクセスすることを可能にするオープン規格です。WAP 技術の概要については，第 56 章 「ワイヤレス仮想ユーザの紹介」を参照してください。

VuGen は，ユーザ・セッションをエミュレートするためにユーザ定義の API 関数を使用します。ほとんどの関数は HTTP プロトコルを使用した標準の Web プロトコル関数です。いくつかの WAP 関数では，WAP 仮想ユーザ固有のアクションをエミュレートします。サポートされている関数の一覧については，796 ページ「ワイヤレス仮想ユーザ関数の使用法」を参照してください。

ツールキットまたは携帯電話を使用して，WAP セッションを記録できます。ツールキットを使った記録方法については，第 59 章 「ワイヤレス仮想ユーザ

の記録オプションの設定」を参照してください。携帯電話を使った記録方法については，「携帯電話での記録」を参照してください。

wap 仮想ユーザ関数を使用して，WAP セッションをエミュレートするためのスクリプトをプログラミングすることもできます。詳細および例については，「**オンライン関数リファレンス**」（[**ヘルプ**] ＞ [**関数リファレンス**]）を参照してください。

VuGen の WAP サポート機能では，ベアラの選択，RADIUS サーバの指定，プッシュ・メカニズムのエミュレートが可能です。これらのサポート機能については，本章に記載しています。

携帯電話での記録

携帯電話またはツールキットと WAP ゲートウェイとの間の WSP セッションを記録できます。WSP セッションを記録するには，ツールキットまたは携帯電話のゲートウェイが設定可能であることを確認しておく必要があります。

記録時に，VuGen によって疑似ゲートウェイが起動されます。VuGen では，このゲートウェイを使って WSP トラフィック情報をキャプチャすることによって，スクリプトを作成します。

WSP 記録セッション向けに VuGen を設定するには，記録オプションの［**記録モード**］セクションで WSP モードを有効にする必要があります（第 59 章「ワイヤレス仮想ユーザの記録オプションの設定」参照）。

基点となるゲートウェイの IP アドレスを入力し，記録モードを「コネクション指向（CO）」または「コネクションレス（CL）」に設定します。選択する記録モードがツールキットまたは携帯電話でサポートされていることをあらかじめ確認しておいてください。

ワイヤレス接続で携帯電話を使って記録を行うには，インターネット・サービス・プロバイダ（ISP）にダイアルインし，インターネットにアクセスしなければなりません。VuGen マシンの IP アドレスと記録モード（CO または CL）を携帯電話に設定します。

VuGen マシンは，以下の設定環境においてのみ有効となります。

- サード・パーティの ISP 経由で接続する場合，疑似ゲートウェイを実行する VuGen マシンはインターネットから直接アクセスできる必要があります。つまり，ファイアウォールの内側にあってはなりません。
- リモート・アクセス・サーバ（RAS）経由でダイアルインするときは，ネットワークに属しているものとして VuGen マシンにアクセスすることになります。

ベアラのサポート

WAP アーキテクチャのトランスポート・レイヤ・プロトコルは，WTP（Wireless Transaction Protocol）と WDP（Wireless Datagram Protocol）で構成されています。

基礎を形成するベアラは，2 つのデバイス間で WDP プロトコルを使ってデータ伝送を行うメカニズムです。ベアラにはたとえば，SMS-CIMD2，UDP, CSD, GSM GPRS，GSM CSD，Packet Data などがあります。

LoadRunner の WAP 仮想ユーザでは，現在 UDP（User Datagram Protocol）および SMS-CIMD2（Short Message Service）の 2 つのベアラがサポートされています。

UDP ベアラは他の接続を必要とせず，IP ネットワーク上で機能します。しかし，SMS-CIMD2 を使用する場合は，SMS センター（SMSC）に接続して必要な情報を提供します。

- **IP とポートの情報**：UDP ベアラの場合，［実行環境設定］の［**ベアラ**］ノードでポートとログインの情報を定義します（820 ページ「ベアラ情報の設定」参照）。
- **SMS センターへのログイン情報**：［実行環境設定］の［**ベアラ**］ノードで SMS ログイン情報を定義します。この情報は，**wap_set_sms_user** 関数を使用して設定することもできます。この関数は，負荷テストを実施する際に，パラメータを使用して多数の仮想ユーザ用にログイン情報を設定する必要がある場合に便利です。
- **CIMD2 へのログイン情報**：［実行環境設定］の［**ベアラ**］ノードで CIMD2 ベアラ情報を設定します（第 60 章「WAP 実行環境の設定」参照）。

場合によっては，複数のタイプのベアラを使用する必要があるかもしれません。たとえば，携帯電話の電源を切っているときに，誰かがその電話宛てに UDP プロトコルでメッセージを送ったとします。電話の電源を入れたときには，SMS プロトコルを通じてメッセージを受信することになります。スクリプトの実行中にベアラのタイプを切り替えるには，**wap_set_bearer** 関数を使用します。

RADIUS のサポート

RADIUS（Remote Authentication Dial-In User Service）は，リモート・アクセス・サーバが中央のサーバと通信し，ダイアルイン・ユーザを認証したり，要求されたシステムまたはサービスへのアクセス権を付与したりするためのクライアント / サーバ型のプロトコルとソフトウェアの組み合わせです。

RADIUS により，すべてのリモート・サーバから利用できる中央のデータベースにユーザ・プロファイルを維持することができます。その結果，セキュリティが向上し，集中管理を行うネットワーク上の 1 か所でポリシーを設定して適用できます。サービスを集中管理することによって，請求処理のための使用状況の追跡，ネットワークの統計情報の格納が簡単になります。

RADIUS には，次の 2 つのサブプロトコルがあります。

- ［**認証**］: ユーザ・アクセス権を付与して制御します。
- ［**会計**］: 請求処理を行うため，およびネットワークの統計情報を取得するために使用状況を追跡します。

VuGen では，WSP 再生の場合のみ，RADIUS のサブプロトコルである認証とアカウントの両方をサポートしています。

［実行環境設定］の［**Radius**］ノードでダイアルイン情報を入力します。詳細については，第 60 章 「WAP 実行環境の設定」を参照してください。

プッシュのサポート

通常のクライアント / サーバ・モデルでは，クライアントはサーバに情報またはサービスを要求します。サーバが応答して，クライアントに情報を送信したりサービスを提供したりします。これを**プル**技術といい，クライアントがサーバから情報を取得します。

これに対するものとして，「**プッシュ**」技術があります。WAP のプッシュ・フレームワークは，ユーザによるアクションがなくても，情報をデバイスに送信します。この技術もクライアント / サーバ・モデルに基づいていますが，サーバがコンテンツを送る前にクライアントからの明示的な要求はありません。

WAP でプッシュ操作を実行するときには，「**プッシュ・イニシエータ**（PI）」がクライアントにコンテンツを送信します。ただし，プッシュ・イニシエータ・プロトコルは WAP クライアントと完全互換ではありません。プッシュ・イニシエータはインターネット上にあり，WAP クライアントは WAP ドメインにあるためです。したがって，プッシュ・イニシエータと WAP クライアントの間

に，仲介機能を果たす変換ゲートウェイを挿入する必要があります。変換ゲートウェイは，「**プッシュ・プロキシ・ゲートウェイ**（PPG）」といいます。

インターネット側のアクセス・プロトコルは，「**プッシュ・アクセス・プロトコル**（PAP）」といいます。

WAP 側のプロトコルは，「**プッシュ OTA**（Over-The-Air）」プロトコルといいます。

プッシュ・イニシエータは，インターネット上で PAP インターネット・プロトコルを使用して，プッシュ・プロキシ・ゲートウェイ（PPG）にアクセスします。PAP は，HTTP などの一般的なインターネット・プロトコルに埋め込める XML メッセージを使用します。PPG はプッシュされたコンテンツを WAP ドメインに転送します。コンテンツはその後，OTA プロトコルを使用し，モバイル・ネットワークを経由して，目的のクライアントまで送信されます。OTA プロトコルは，WSP サービスに基づいています。

PPG は簡単なプロキシ・ゲートウェイ・サービスを提供するほか，プッシュ・イニシエータにプッシュ操作の最終ステータスを通知することができます。また，双方向のモバイル・ネットワークにおいては，クライアントがコンテンツを受け入れるか拒否するまで待機することができます。

プッシュ・サービスのタイプ

プッシュ・サービスのタイプには，SL と SI があります。

- **SL** ― サービス・ロード（SL）コンテンツ・タイプでは，モバイル・クライアント上のユーザ・エージェントがサービスをロードして実行できます。たとえば，WML デッキなどを実行できます。SL には，ユーザの介入なしに適宜ユーザ・エージェントによってロードされるサービスを示す URI が含まれています。
- **SI** ― サービス通知（SI）。このコンテンツ・タイプでは，エンド・ユーザに非同期で通知を送信できます。たとえば，新規メールの到着，株価の変動，ニュースのヘッドライン，広告などの通知が考えられます。

  最も基本的な形式の SI には，ショート・メッセージとサービスを示す URI が含まれています。メッセージは，エンド・ユーザの受信時に提示され，ユーザは URI が示すサービスをすぐに開始するか，後で処理するために SI を延期するかを選択できます。SI を延期すると，クライアントによってサービスが保存され，エンド・ユーザは後でそのサービスを開始できます。

LoadRunner でのプッシュのサポート

VuGen でのプッシュのサポートは，次の 3 つに分類できます。

- クライアント側でのプッシュのサポートープッシュ型メッセージを受信する機能です。
- WAP HTTP 仮想ユーザに対するプッシュのサポートープッシュ・イニシエータをエミュレートします。
- プッシュ型メッセージ（SI および SL）フォーマット・サービスープッシュ型メッセージをフォーマットします。

クライアント側におけるプッシュのサポート

VuGen は，クライアント側では，すべての再生モード（CO および CL）について，SL および SI の両方のプッシュ・サービスをサポートしています。**wap_wait_for_push** 関数は，仮想ユーザにプッシュ・メッセージの到着まで待機するように指示します。この関数のタイムアウトは，実行環境の設定で指定します。

プッシュ・メッセージが到着すると，LoadRunner によってメッセージが解析され，タイプの識別と属性の取得が行われます。解析が正常に行われると，プル・トランザクションが生成された後に実行され，該当データが取得されます。［実行環境設定］でプル・イベントを無効にすると，LoadRunner はメッセージを取得しません。詳細については，第 60 章 「WAP 実行環境の設定」を参照してください。

プッシュ・イニシエータのエミュレート

LoadRunner では WAP HTTP 仮想ユーザのプッシュ機能がサポートされているため，PPG の負荷テストが可能です。プッシュのサポートにより，仮想ユーザは，**Push Access Protocol**（PAP）をサポートするプッシュ・イニシエータとして機能できます。PAP では，以下の PI と PPG の間の一連の操作が定義されています。

1 プッシュ要求を送信する

2 プッシュ要求を取り消す

3 プッシュ要求のステータスを調べるクエリーを送信する

4 ワイヤレス・デバイスの機能のステータスを調べるクエリーを送信する

5 PPG から PI へ結果通知メッセージを発行する

上記の操作はすべて，要求と応答から成ります。つまり，発行されたすべてのメッセージに対して，応答が PI に返されます。PI の操作は，LoadRunner でサポートされている通常の HTTP POST メソッドに基づいています。現バージョンでは，最初の 2 つの操作だけが **wap_push_submit** および **wap_push_cancel 関数によってサポートされています。**

web_submit_data 関数を使用して，Web サーバにデータを送信できます。ただし，この関数では長く複雑な構造のデータを送信することは困難です。この種のデータの送信を可能にするため，またより直観的に理解できる API 関数を提供するために，XML メッセージ・データを適切にフォーマットする新しい API 関数 **wap_format_si_msg** と **wap_format_sl_msg** が追加されました。これらの関数の詳細については，「**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。

MMS のサポート

マルチメディア・メッセージ・サービス（MMS）は，WAP クライアントがさまざまなタイプのメディアに対応したメッセージ操作を提供できるシステム・アプリケーションです。MMS クライアントのエミュレーションは，次の仮想ユーザ関数を使って実装されます。

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

wap_mms_msg_add_field	MMS メッセージにフィールドを追加します。
wap_mms_msg_add_mulitpart_entry	MMS メッセージにマルチパート・エントリを追加します。
wap_mms_msg_create	MMS 用のメッセージを作成します。
wap_mms_msg_destroy	MMS 用のメッセージを破棄します。
wap_mms_msg_submit	MMS にメッセージを送信します。

第 59 章

ワイヤレス仮想ユーザの記録オプションの設定

ワイヤレス・セッションの記録を開始する前に，記録オプションをカスタマイズできます。

本章では，次の項目について説明します。

- 記録モードの指定（WAP のみ）
- 記録する情報の指定（i モードおよび VoiceXML）
- ツールキットの指定

記録オプションの設定について

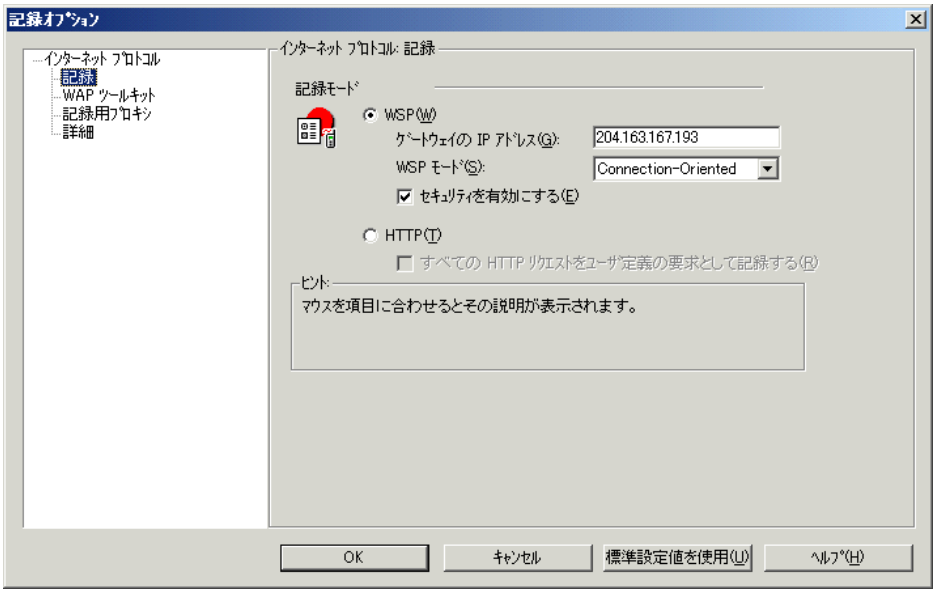
VuGen では，ユーザがワイヤレス・インタフェースを利用して Web サイトで実行する標準的なプロセスを記録することによって，ワイヤレス仮想ユーザ・スクリプトを生成できます。

記録を行う前に，記録オプションを設定して，記録する情報，記録に利用するツールキット，グローバル・プロキシ設定などを指定します。

プロキシ設定や他の詳細設定など，インターネット・プロトコルの一般記録オプションを設定できます。詳細については，第 35 章 「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。

記録モードの指定（WAP のみ）

［記録オプション］ダイアログ・ボックスの［**記録モード**］設定（［**ツール**］＞［**記録オプション**］）を使用して，WAP 仮想ユーザの記録セッション中に VuGen が記録する情報を指定します。



WAP 仮想ユーザ用に VuGen で記録する情報を指定するには，次の手順で行います。

［**記録モード**］セクションで，次のオプションから 1 つを選択します。

➤ ［**WSP**］：このオプションを選択すると，ツールキットまたは携帯電話と，ゲートウェイとの間のすべての WSP トラフィックが VuGen によって記録されます。アクションは URL ステップとして記録されます。ゲートウェイの IP アドレスを入力し，［WSP モード］ボックスから［**Connectionless**］または［**Connection-Oriented**］を選択します。セキュリティを有効にした WAP を使って記録できるようにするには，［**セキュリティを有効にする**］チェック・ボックスを選択します。

WSP モードで記録できるように，VuGen では Phone.com UP Simulator 4.1 ツールキットをネイティブでサポートとしています。サポート機能は，ツールキットがインストールされていることを検出すると，設定パラメータを自動的に設定した後，ツールキットを起動します。Nokia ツールキット（1.3

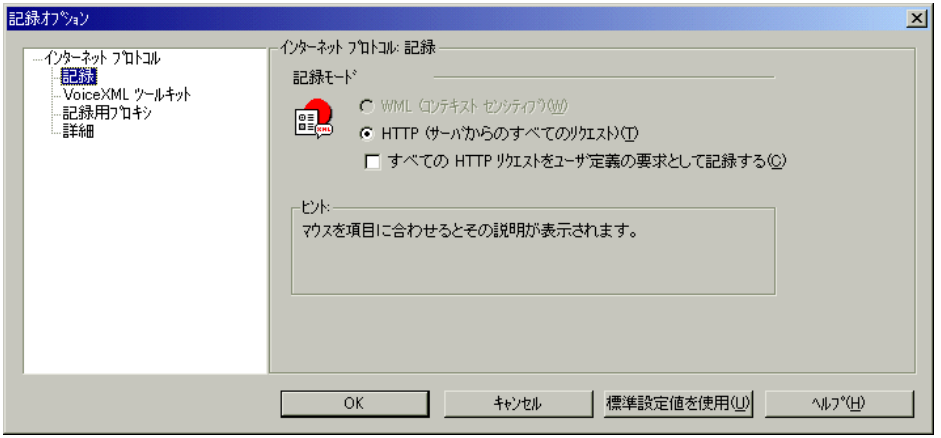
および 2.0）の場合には，正しい IP アドレスを指定して手作業で起動する必要があります。詳細については，798 ページ「ワイヤレス仮想ユーザ・スクリプトのトラブルシューティング」を参照してください。

➤ ［**HTTP**］：このオプションを選択すると，ツールキットと Web サーバとの間の HTTP トラフィックが URL ステップとして VuGen によって記録されます。［**すべての HTTP リクエストをユーザ定義の要求として記録する**］チェック・ボックスを選択すると，すべての HTTP 要求がコンテキストを持たないユーザ定義の HTTP 要求として記録され，**web_custom_request** 関数が生成されます。

記録する情報の指定（i モードおよび VoiceXML）

［記録オプション］ツリーの［**記録**］ノードを使用して，記録セッション中に VuGen が記録する情報を指定します。i モード記録モードと VoiceXML 記録モードの両方を選択します。

使用可能な記録モードは **HTTP** だけです。このモードでは，VuGen では ツールキットと Web サーバの間の HTTP トラフィックが URL ステップとして記録されます。



i モードまたは VoiceXML 仮想ユーザ用に VuGen によって記録される情報を指定するには，次の手順で行います。

1 ［記録オプション］を開き（［**ツール**］＞［**記録オプション**］），記録オプション・ツリーで［**インターネット プロトコル ：記録**］ノードを選択します。

2［**記録モード**］セクションで，［**すべての HTTP リクエストをユーザ定義の要求として記録する**］オプションを選択すると，すべての HTTP 要求がコンテキストを持たないユーザ定義の HTTP 要求として記録され，**web_custom_request** 関数が生成されます。

i モード記録モード

このノードでは，i モード仮想ユーザの記録モードを指定できます。

どの記録モードを選択するかは，必要性と環境に応じて決めます。利用可能なモードは，「HTTP」と「ユーザ定義要求」です。

［**HTTP**］：ユーザのアクションの結果としてサーバに送信されるすべての HTTP 要求をキャプチャし，リクエストごとに **web_url** ステートメントを生成します。この記録モードでは，アプレットや非ブラウザ・アプリケーションなど，HTML 以外のアプリケーションもキャプチャします。

［**すべての HTTP リクエストをユーザ定義の要求として記録する**］：すべての HTTP 要求をユーザ定義の HTTP 要求として記録し，コンテキストは無視します。ユーザ定義要求は，特定の構造や HTTP 要求ステートメントに依存しません。VuGen によって，記録される各ページとリソースごとに，**web_url**，**web_image**，または **web_submit_form** 関数の代わりに，**web_custom_request** 関数が生成されます。

VoiceXML 記録モード

このノードでは，VoiceXML 仮想ユーザの記録モードを指定できます。

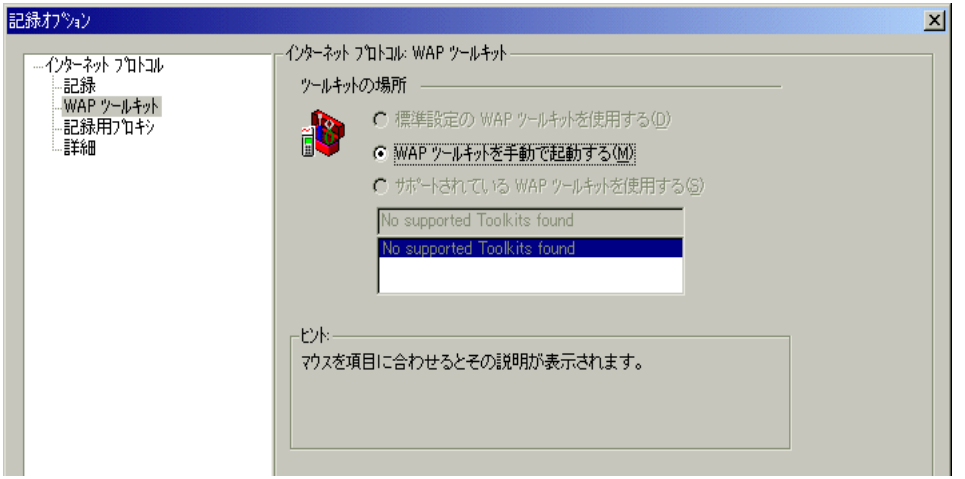
どの記録モードを選択するかは，必要性と環境に応じて決めます。利用可能なモードは，「HTTP」と「ユーザ定義要求」です。

［**HTTP**］：ユーザのアクションの結果としてサーバに送信されるすべての HTTP 要求をキャプチャし，リクエストごとに **web_url** ステートメントを生成します。この記録モードでは，アプレットや非ブラウザ・アプリケーションなど，HTML 以外のアプリケーションもキャプチャします。

［**すべての HTTP リクエストをユーザ定義の要求として記録する**］：すべての HTTP 要求をユーザ定義の HTTP 要求として記録し，コンテキストは無視します。ユーザ定義要求は，特定の構造や HTTP 要求ステートメントに依存しません。VuGen によって，記録される各ページとリソースごとに，**web_url**，**web_image**，または **web_submit_form** 関数の代わりに，**web_custom_request** 関数が生成されます。

ツールキットの指定

ワイヤレス仮想ユーザ・スクリプトを記録するときに使用するツールキットを指定できます。記録オプション・ツリーの［**ツールキット**］ノードで，WAPVoiceXML ツールキット，i モード・ツールキット，または VoiceXML ツールキットを指定します。



次の選択肢があります。

- ［**標準設定の WAP ツールキットを使用する**］：標準設定のツールキットを使用して記録を行います。
- ［**WAP ツールキットを手動で起動する**］：ツールキットを手動で起動します。
- ［**サポートされている WAP ツールキットを使用する**］：サポートされているツールキットのリストのツールキットを使用します。

WAP ツールキット

このモードでは，記録時にどの WAP ツールキットを使用するかを指定します。インストール済みのサポートされているツールキットは下のリストに表示されます。

WAP 仮想ユーザ・スクリプトを記録するためのツールキットを指定するには，次の手順で行います。

1 ［**ツール**］＞［**記録オプション**］を選択し，［**WAP ツールキット（または i モード ツールキット，VoiceXML ツールキット）**］ノードを選択します。

2［**ツールキットの場所**］セクションで，次のオプションから 1 つを選択します。

- ［**標準設定の WAP（i モード または VocieXML）ツールキットを使用する**］: このオプションを選択すると，記録用コンピュータ上で通常使用するように設定されているツールキットが使用されます（現在無効になっています）。
- ［**WAP（i モードまたは VocieXML）ツールキットを手動で起動する**］: このオプションを選択すると，記録を開始したときにツールキットは起動されません。記録セッションを開始した後にツールキットを手作業で起動する必要があります。
- ［**サポートされている WAP（i モードまたは VocieXML）ツールキットを使用する**］: このオプションを選択すると，記録用マシンにインストールされているツールキットの中から選択したツールキットが使用されます。ダイアログ・ボックスに一覧表示されている使用可能なツールキットから 1 つを選択します。

i モード・ツールキット

このモードでは，記録時にどの i モード・ツールキットを使用するかを指定します。インストール済みのサポートされているツールキットは下のリストに表示されます。

i モード仮想ユーザ・スクリプトを記録するためのツールキットを指定するには，次の手順で行います。

1［**ツール**］＞［**記録オプション**］を選択し，［**i モード ツールキット**］ノードを選択します。

2［**ツールキットの場所**］セクションで，次のオプションから 1 つを選択します。

- ［**標準設定の i モード ツールキットを使用する**］: このオプションを選択すると，記録用コンピュータ上で通常使用するように設定されているツールキットが使用されます（現在無効になっています）。
- ［**i モード ツールキットを手動で起動する**］: このオプションを選択すると，記録を開始したときにツールキットは起動されません。記録セッションを開始した後にツールキットを手作業で起動する必要があります。
- ［**サポートされている i モード ツールキットを使用する**］: このオプションを選択すると，記録用マシンにインストールされているツールキットの中から選択したツールキットが使用されます。ダイアログ・ボックスに一覧表示されている使用可能なツールキットから 1 つを選択します。

VoiceXML ツールキット

このモードでは，記録時にどの VoiceXML ツールキットを使用するかを指定します。インストール済みのサポートされているツールキットは下のリストに表示されます。

VoiceXML 仮想ユーザ・スクリプトを記録するためのツールキットを指定するには，次の手順で行います。

1 ［**ツール**］＞［**記録オプション**］を選択し，［**VoiceXML ツールキット**］ノードを選択します。

2 ［**ツールキットの場所**］セクションで，次のオプションから 1 つを選択します。

- ［**標準設定の VoiceXML ツールキットを使用する**］：このオプションを選択すると，記録用コンピュータ上で通常使用するように設定されているツールキットが使用されます（現在無効になっています）。
- ［**VoiceXML ツールキットを手動で起動する**］：このオプションを選択すると，記録を開始したときにツールキットは起動されません。記録セッションを開始した後にツールキットを手作業で起動する必要があります。
- ［**サポートされている VoiceXML ツールキットを使用する**］：このオプションを選択すると，記録用マシンにインストールされているツールキットの中から選択したツールキットが使用されます。ダイアログ・ボックスに表示されている利用可能なツールキットのリストから 1 つを選択します。

第 60 章

WAP 実行環境の設定

WAP 仮想ユーザ・スクリプトを記録した後に，WAP 固有の実行環境を設定します。

本章では，次の項目について説明します。

- ゲートウェイ・オプションの設定
- ベアラ情報の設定
- RADIUS 接続データの設定

WAP および他のすべてのワイヤレス・プロトコルに対する，一般的なインターネット・プロトコルの実行環境の設定の詳細については，第 37 章 「インターネット実行環境の設定」を参照してください。

WAP 実行環境の設定について

WAP 仮想ユーザ・スクリプトを作成した後，WAP 固有の実行環境の設定を行います。これらの設定により，WAP 仮想ユーザの動作を制御して，WAP デバイスの実際のユーザを正確にエミュレートできるようになります。ゲートウェイ，Radius，ベアラに関する WAP 実行環境の設定が行えます。

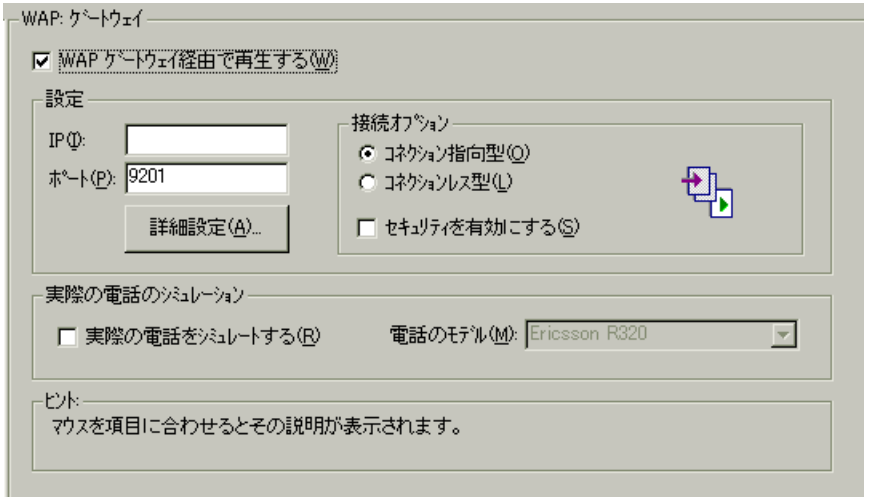
WAP の実行環境の設定は，［実行環境設定］ダイアログ・ボックスで行います。各ノードをクリックして，設定項目を表示し，必要な設定を行います。

［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［**実行環境の設定を編集**］ボタンをクリックします。LoadRunner コントローラの［デザイン］タブの［**実行環境の設定**］ボタンをクリックして，実行環境の設定を変更することもできます。

本章では，WAP 仮想ユーザのゲートウェイ実行環境の設定について説明します。すべてのワイヤレス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 37 章 「インターネット実行環境の設定」を参照してください。

ゲートウェイ・オプションの設定

［実行環境設定］ダイアログ・ボックスの［**ゲートウェイ**］ノードを使って，ゲートウェイの設定を行います。



通信プロトコル

ゲートウェイの設定が必要となるのは，WSP プロトコルを使って仮想ユーザを実行し，WAP ゲートウェイ経由で Web サーバにアクセスする（［**WAP ゲートウェイ経由で再生する**］オプションを選択）場合だけです。ゲートウェイを経由してスクリプトを実行する場合，ゲートウェイの IP とポート・アドレスを指定する必要があります。

HTTP モードで仮想ユーザを実行し，Web サーバに直接アクセス（［**WAP ゲートウェイ経由で再生する**］チェック・ボックスをクリア）している場合，ゲートウェイの設定は適用されません。

設定

［**IP**］：ゲートウェイの IP アドレスを指定します。

［**ポート**］：ゲートウェイのポートを指定します。WAP ゲートウェイ経由で仮想ユーザを実行する場合，選択したモードに応じて標準のポート番号が自動的に設定されます。ただし，設定をカスタマイズして，ユーザ定義の IP アドレスとポートを指定することもできます。

［**詳細設定**］：［ゲートウェイの詳細設定］ダイアログ・ボックスが開き，クライアントその他のゲートウェイの詳細設定を設定できます。

接続オプション

このセクションでは，再生時の接続モードを指定します。

- ［**コネクション指向型**］：WSP セッションの接続モードを「コネクション指向」に設定します。
- ［**コネクションレス型**］：WSP セッションの接続モードを「コネクションレス」に設定します。
- ［**セキュリティを有効にする**］：WAP ゲートウェイへの接続のセキュリティを有効にします。

実際の電話のシミュレーション

VuGen では，仮想ユーザを再生するときの携帯電話の種類を指定できます。一般的なベンダの携帯電話のモデルを一覧から選択できます。VuGen は，選択した電話に対応するクライアント・ヘッダーを決定し，その電話をヘッダーに従ってエミュレートします。

- ［**実際の電話をシミュレートする**］：実際の電話をシミュレートします。
- ［**電話のモデル**］：シミュレートする電話のモデルをメニューから選択します。

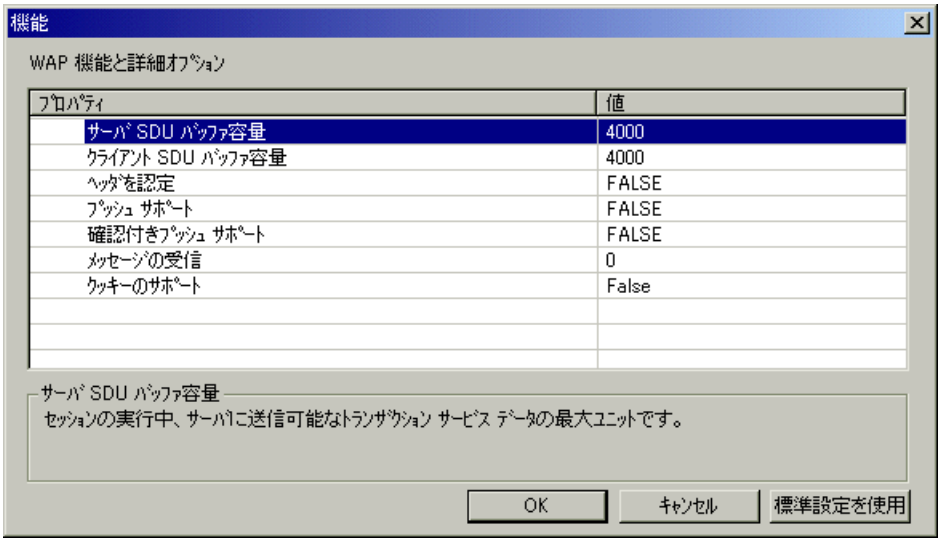
実際の電話のシミュレーションを有効にした場合，［ゲートウェイ］の詳細設定はすべて無視されます。その代わりに，サポート対象の各電話を定義している VuGen の設定ファイルからヘッダーとクライアントの機能情報が取得されます。

実際の電話のシミュレーションは，さまざまな携帯電話を使用してテストを実行する必要がある場合は特に役立ちます。たとえば，**Motorola Timeport** 用のスクリプトを記録し，**Nokia 6110** で再生することもできます。実際の電話のシミュレーションを行ってスクリプトを再生する場合，スクリプト内の **wap_set_capability** 関数と **wap_add_const_header** 関数はすべて無視されます。仮想ユーザは，各電話に対応するヘッダーを定義している設定ファイルから，必要な情報をすべて取得します。

エミュレート対象の電話がリストにない場合は，LoadRunner の **dat** ディレクトリにある **LrwWapPhoneDB.dat** という設定ファイルにその電話を手作業で追加すれば，実行環境の設定のインタフェースに加えることができます。詳細については，設定ファイルの最初にあるコメントを参照してください。

ゲートウェイの詳細設定

ゲートウェイ・ノードで［**詳細設定**］をクリックすると，［機能］ダイアログ・ボックスが開きます。［機能］ダイアログ・ボックスで，WAP の機能およびゲートウェイの詳細オプションを設定できます。



- ［**サーバ SDU バッファ サイズ**］：セッションの実行中に「**サーバ**」へ送信可能な最大のトランザクション・サービス・データ・ユニット（標準設定では 4000）。
- ［**クライアント SDU バッファ サイズ**］：セッションの実行中に「**クライアント**」へ送信可能な最大のトランザクション・サービス・データ・ユニット（標準設定では 4000）。
- ［**確認応答ヘッダ**］：ゲートウェイに情報を提供する標準ヘッダを返します（標準設定では無効）。
- ［**プッシュのサポート**］：プッシュ・タイプのメッセージがゲートウェイを通過できるようにします（標準設定では無効）。
- ［**プッシュのサポートの確認**］：このオプションを選択すると，CO モードでプッシュ型メッセージが受信されたときに，仮想ユーザにメッセージの受信を確認させます（標準設定では無効）。

- ［**メッセージ取得**］：このオプションを選択した場合，仮想ユーザがプッシュ型メッセージを受信すると，そのメッセージに示された URL からメッセージ・データを取得します（標準設定では無効）。
- ［**クッキーをサポート**］：クッキーの保存と取得をサポートします（標準設定では無効）。

ゲートウェイの詳細設定

この項では，WAP ゲートウェイのオプションを設定する手順を示します。

WAP ゲートウェイ・オプションを設定するには，次の手順で行います。

1 ［**実行環境の設定を編集**］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。［**WAP：ゲートウェイ**］ノードを選択します。

2 スクリプトを（HTTP ではなく）WSP モードで再生する場合は，［**WAP ゲートウェイ経由で再生する**］を選択します。

3 ゲートウェイの IP アドレスとポートを指定します。VuGen の標準設定のポートを使用することもできます。

4 接続モードとして，［**コネクション指向型**］または［**コネクションレス型**］を選択します。セキュリティ上安全な接続モードを指定するには，［**セキュリティを有効にする**］オプションを選択します。

5 一般的な携帯電話をエミュレートするには，［**実際の電話をシミュレートする**］を選択し，プルダウン・リストから使用する電話を選びます。

6 一般的でない電話をエミュレートする場合は，［**詳細設定**］をクリックし，クライアントの機能とその他のゲートウェイ詳細オプションを設定します。

- ［**サーバ SDU バッファ容量**］および［**クライアント SDU バッファ容量**］の値を入力します。
- 仮想ユーザで確認応答ヘッダを取得するよう設定するには，［**ヘッダを認定**］オプションを選択します。
- プッシュ・メッセージを有効にするには，［**プッシュ サポート**］の横にあるカラムで「**True**」を選択します。
- プッシュ・メッセージの確認を有効にするには，［**確認付きプッシュ サポート**］の横にあるカラムで「**True**」を選択します。

- プッシュ・メッセージの URL からデータを取得するには，［**メッセージの受信**］の横にあるカラムで「**True**」を選択します。
- クッキーを有効にするには，［**クッキーをサポート**］の横にあるカラムで「**True**」を選択します。

ベアラ情報の設定

基礎を形成する**ベアラ**は，2 つのデバイス間で WDP プロトコルを使ってデータ伝送を行うメカニズムです。ベアラにはたとえば，SMS，UDP，CSD，GSM，GPRS, ，および Packet Data などがあります。

LoadRunner は，UDP と SMS の両方のベアラをサポートしています。実行環境の設定で，最初に使用するベアラを指定します。**wap_set_bearer** 関数を使えば，再生中にベアラを切り替えることができます。両方のベアラを使用する場合は，再生前に実行環境の設定でそれらを有効にしておきます。

SMS-CIMD2 ベアラを使用する場合は，ショート・メッセージ・サービス・センター（SMSC）に接続し，ログイン情報を入力します。［実行環境設定］の［**WAP：ベアラ**］ノードで，ポート情報を定義します。

wap_set_sms_user API 関数を使用するか，［実行環境設定］ダイアログ・ボックスを利用して，SMS のログイン情報を設定できます。ログイン情報を関数によって設定する利点は，パラメータを利用できるので，多くの値を使用してスクリプトを実行できることです。API 関数の値は，実行環境の設定に優先します。［実行環境設定］の［**ベアラ**］ノードで，ベアラの属性を設定します。

ベアラの設定	説明
UDP ベアラを有効にする	UDP ベアラへの接続を開きます。
SMS-CIMD2 ベアラを有効にする	CIMD2 ベアラへの接続を開きます。
ベアラ タイプ	標準設定のベアラのタイプ（UDP または CIMD2）。
CIMD2:IP アドレス	SMSC サーバの IP アドレス。
CIMD2: ポート番号	SMSC サーバのポート番号。
CIMD2: ゲートウェイ ID	SMSC で定義されている WAP ゲートウェイ・アプリケーション ID。
CIMD2: ユーザ名	サーバへのログイン・ユーザ名。

CIMD2: ユーザ パスワード　　ユーザのパスワード。

CIMD2: 発信元アドレス　　ユーザの発信アドレス。

WAP ベアラのオプションを設定するには，次の手順で行います。

1 ［**実行環境の設定を編集**］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。［**Bearers**］ノードを選択します。

2 ［**UDP**］または［**SMS-CIMD2**］ベアラを有効にするには，「True」を選択します。

3 最初に使用する［**ベアラ タイプ**］を，右のカラムで「UDP」または「SMS-CIMD2」から選択します。

4 ［**IP アドレス**］をドット区切りの形式で入力します。

5 ［**ポート番号**］を入力します。

6 ［**ゲートウェイ ID**］を入力します（SMS ゲートウェイ ID ではありません）。

7 ［**ユーザ名**］を入力します。

8 ［**ユーザ パスワード**］を入力します。

9 ［**発信アドレス**］を入力します。

10 ［**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。

RADIUS 接続データの設定

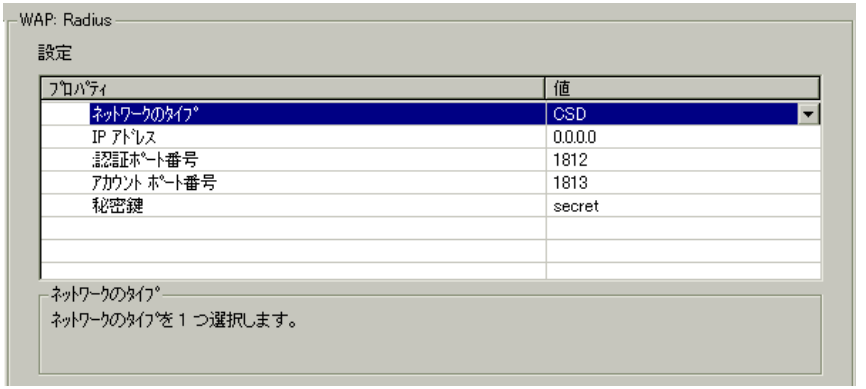
RADIUS（Remote Authentication Dial-In User Service）は，クライアント / サーバのプロトコルとソフトウエアの組み合わせで，リモート・アクセス・サーバと中央サーバとの通信によって，ダイアルアップ・ユーザの認証と，そうしたユーザのシステムやサービスへのアクセス要求に対する権限を付与することを可能にします。

［実行環境設定］の［**Radius**］ノードで，ダイアルアップ情報を入力します。

プロパティ	値
ネットワークのタイプ	アカウント ネットワークのタイプ：GPRS（General Packet Radio Service）または CSD（Circuit-Switched Data）を選択します。
IP アドレス	Radius サーバの IP アドレス。
認証ポート番号	Radius サーバの認証ポート番号。
アカウント ポート番号	Radius サーバのアカウント ポート番号。
秘密鍵	Radius サーバの秘密鍵。

WAP Radius のオプションを設定するには，次の手順で行います。

1 ［**実行環境の設定を編集**］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。**Radius**］ノードをクリックします。



2 ［Radius Authentication］セクションで，認証を有効にするには **True** を選択し，無効にするには **False** を選択します。

3 Radius サーバの［**IP アドレス**］をドット区切りの形式で入力します。

4 Radius サーバの［**認証ポート番号**］と［**アカウント ポート番号**］を入力します。

5 Radius のアカウント認証で使用する［**Secret key**］の値を入力します。

6 Radius サーバにログインするためのユーザ名を［**User name**］に入力します。

7 ユーザ名に対するパスワードを［**User Password**］に入力します。

8 ［**MSISDN**］に識別番号を入力します。

9 アカウント用の［**ネットワーク タイプ**］として，GPRS（General Packet Radio Service）または CSD（Circuit-Switched Data）を選択します。

10 ［**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 15 部

GUI 仮想ユーザ・スクリプト

第 61 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，次の項目について説明します。

- 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。

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，C 言語に似た，高度で使いやすい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），特定のユーザ負荷をエミュレートしたり（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，『**LoadRunner コントローラ・ユーザーズ・ガイド**』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

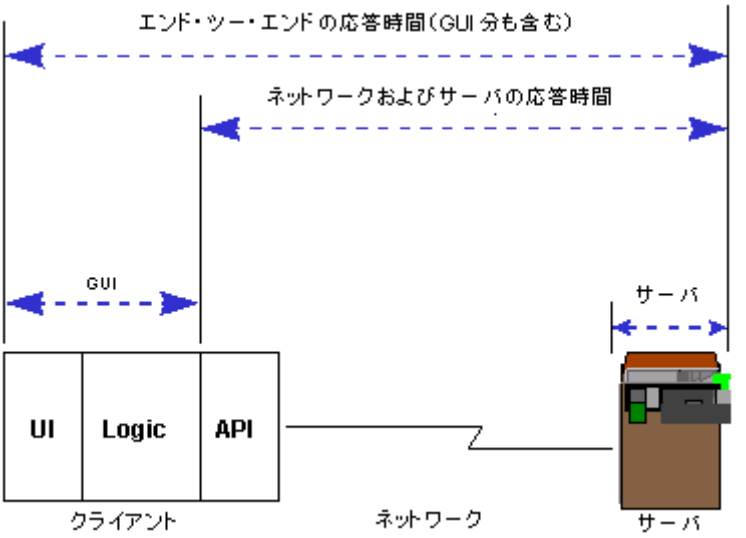
GUI 仮想ユーザの監視と管理は，コントローラだけを使って行います。たとえば，コントローラを使って，仮想ユーザの実行，一時停止，表示およびシナリオのステータスの監視ができます。

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 ユーザーズ・ガイド』**を参照してください。

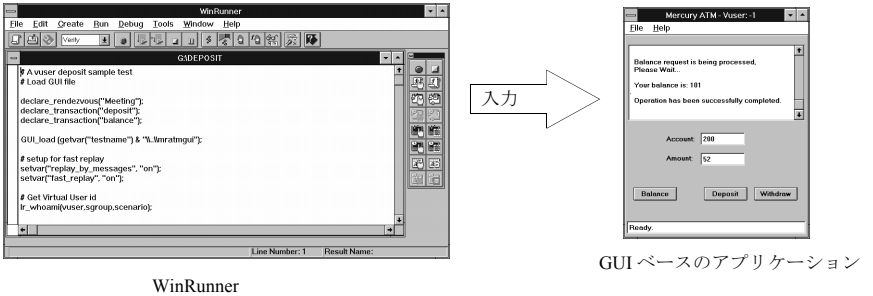
注：VuGen では，GUI 仮想ユーザ・スクリプトを実行できません。GUI 仮想ユーザ・スクリプトをシナリオの一部として実行するには，コントローラを使います。スタンドアロン・モードで GUI 仮想ユーザ・スクリプトを実行するには，WinRunner を使います。

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。



GUI 仮想ユーザ

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 ユーザーズ・ガイド**』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。**

ループやそのほか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 ユーザーズ・ガイド**』を参照してください。TSL の詳細については，「**TSL オンライン・リファレンス**」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 ユーザーズ・ガイド**』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 LoadRunner シナリオに組み込みます。仮想ユーザ・スクリプトをシナリオに組み込む方法の詳細については，『**LoadRunner コントローラ・ユーザーズ・ガイド**』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，832 ページ「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がエミュレートされるようにします。詳細については，833 ページ「大きいユーザ負荷の生成：ランデブー・ポイント」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**」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 つのクエリーに対する応答を待つような簡単な処理の場合や，いくつかのクエリー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。

シナリオの実行中，コントローラは各トランザクションの実行に要する時間を測定します。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。シナリオの実行後に，LoadRunner 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。シナリオ実行中，LoadRunner 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 オンライン・リファレンス**」（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

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ーを実行します。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にします。ある仮想ユーザがランデブー・ポイントに到達すると，コントローラによって，ほかのすべての仮想ユーザがランデブー・ポイントに到着するまでその仮想ユーザは待機させられます。ランデブーの条件が満たされると，コントローラ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。

たとえば，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 オンライン・リファレンス**」（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，C 言語に似た，高度で使いやすい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 オンライン・リファレンス**」（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 で作成した簡単な仮想ユーザ・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 オンライン・リファレンス**」（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。

コントローラへのメッセージの送信

シナリオを実行すると，コントローラの［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 オンライン・リファレンス**」（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

シナリオの実行中，以下の ID を取得できます。

- シナリオ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 オンライン・リファレンス**」（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 16 部

上級ユーザのために

第 62 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，次の項目について説明します。

- 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

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 LoadRunner 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 23 章 「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，LoadRunner 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
- LoadRunner 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 6 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，次の手順で行います。

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 :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 LoadRunner の一般関数を使用してスクリプトを拡張します。たとえば，メッセージを発行するには **lr_output_message**，トランザクションの開始を示すには **lr_start_transaction** を使用します。詳細については，「**オンライン関数リファレンス**」（［**ヘルプ**］＞［**関数リファレンス**］）の「一般関数」を参照してください。

7 プロジェクトをビルドします。DLL として出力されます。

8 DLL と同じ名前のディレクトリを作成し，作成した DLL をこのディレクトリにコピーします。

9 **Template** ディレクトリの **lrvuser.usr** ファイルを開き，USR ファイル・キーを次のようにその 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，LoadRunner 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，次の手順で行います。

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，846 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，LoadRunner 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，LoadRunner のインストール・ディレクトリにある **bin** ディレクトリにあります。

実行環境の設定を行い，スクリプトをパラメータ化するには，次の手順で行います。

1 LoadRunner 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 9 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 7 章 「パラメータの定義」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 63 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。LoadRunner 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 文書について
- 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，SOAP，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。

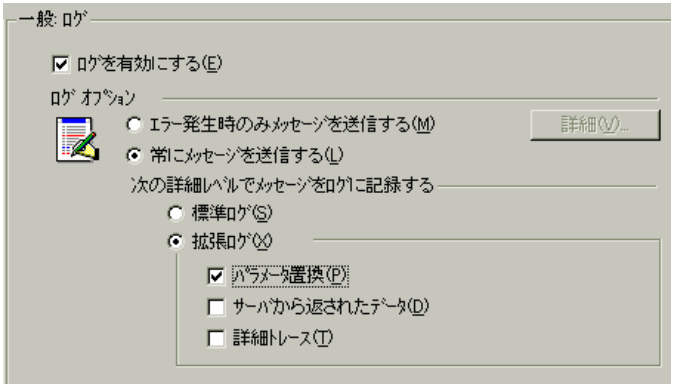
XML API プログラミングについて

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，次の手順で行います。

- 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 9 章 「実行環境の設定」を参照してください。



LoadRunner 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

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。たとえば，**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 の部分文字列を抽出します。
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	クエリーによって見つかった 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。

たとえば，Accounting 部門の従業員の名前を取得するには，次の文字列を使用します。

```
lr_xml_get_values("XML={XML_Input_Param}",
"ValueParam=OutputParam",
"Query=/acme_org/accounting_dept/employee/name",
LAST);
```

拡張ログ機能が有効な場合，［実行ログ］ウィンドウには，この関数の出力が表示されます。

```
Output:
Action.c(20): "lr_xml_get_values" was successful, 1 match processed
Action.c(25): Query result = John Smith
```

XML 構造への書き込み

XML ツリーに値を書き込む関数は次のとおりです。

lr_xml_delete	XML 文字列からフラグメントを削除します。
lr_xml_insert	XML 文字列に新しい XML フラグメントを挿入します。
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	クエリーによって見つかった 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;}
```

出力 :

```
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**クエリー**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ーを作成します。クエリーには，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。

LoadRunner における XML 関数の実装では，クエリーの範囲は XML ツリー全体です。ツリー情報は，**xml** 引数の値として LoadRunner API 関数に送られます。

クエリーの複数の検索

XML 要素に対してクエリーを実行すると，標準では最初に一致したものだけが返されます。クエリー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。たとえば，**EmployeeName** という名前のパラメータを定義した場合は，**EmployeeName_1**，**EmployeeName_2**，**EmployeeName_3** などが作成されます。

```
lr_xml_set_values("XML={XML_Input_Param}","ResultParam=NewXmlParam"
, "ValueParam=ExtensionParam","SelectAll=yes", "Query=//extension", LAST);
```

パラメータに**書き込む**関数を使用すれば，書き込んだ後にパラメータの値を評価できます。たとえば次のコードでは，クエリーによる複数の検索結果を取得して出力しています。

```
NumOfValues = lr_xml_get_values("Xml={XmlParam}", "Query=//name",
        "SelectAll=yes", "ValueParam=EmployeeName", LAST);
```

パラメータから値を**読み取る**関数の場合，パラメータの値は事前に定義しておく必要があります。また，パラメータは，＜**パラメータ名**＞ _ ＜**連番**＞という形式（たとえば **Param**_1，**Param**_2，**Param**_3 など）を使用する必要もあります。このようなパラメータの集合をパラメータ・セットとも言います。

次の例では，**lr_xml_set_values** はパラメータ・セットから値を読み取り，その値を XPath クエリ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。

この表現を使用して，特定の属性，または特定の値を持つ属性を操作します。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，使用するプロトコル（HTTP，SOAP など）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ーを含めます。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");

    /* /* クエリー 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);
```

```
/* /* クエリー 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 LoadRunner 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，LoadRunner 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
                        /glkev_ws/WeatherFetcher.asmx?op=GetWeather",
                    "Snapshot=t2.inf",
                    "Mode=HTTP",
                    ITEMDATA,
                    "Name=zipCode", "Value=10010", ENDITEM,
                    LAST
    );

    // City の値を取得する
    lr_xml_get_values("Xml={ParamXml}",
                      "Query=City",
                      "ValueParam=ParamCity",
                      LAST
    );

    lr_output_message(lr_eval_string("***** City = {ParamCity} *****"));

    // State の値を取得する
    lr_xml_get_values("Xml={ParamXml}",
                      "Query=State",
                      "ValueParam=ParamState",
                      LAST
    );
```

```
lr_output_message(lr_eval_string("***** State = {ParamState} *****"));

// テンプレートを使用して複数の値を一度に取得する
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/IEmailservice",
        "Method=POST",
        "TargetFrame=",
        "Resource=0",
```

```
            "Referer=",
            "Body={ParamRequest}",
            LAST);

        // メールが送信されたことを確認する
        lr_xml_find("Xml={ParamXml}",
                    "Query=Body/SendMailResponse/return",
                    "Value=0",
                    LAST
        );

        return 0;
    }
```

第 64 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
- 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 Applications 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);

たとえば，スクリプトに次のステートメントを追加できます。

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

追跡情報の出力を表示するには，出力コールバック関数 ci_set_print_CB() を使います。

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。

キーワードの追加は，次の手順で行います。

1 vugen_extra_keywords.ini ファイルを開きます。このファイルはお使いのコンピュータの< Windows >または< Windows > ¥System ディレクトリにあります。

2 EXTRA_KEYWORDS_C セクションに，C インタプリタ用のキーワードを追加します。

このファイルの形式は次のとおりです。

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

再生出力の検証

再生出力を見ます（VuGen から，あるいは LoadRunner 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

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 Applications での一般的な問題の解決方法

本項では，Oracle Applications 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。

たとえば，グリッド内に 1000003 と表示される値がある場合，これは記録ログ（ASCII ファイル）では 1e+0006 と表示されます。

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。たとえば，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。たとえば，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。

たとえば，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

LoadRunner 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。たとえば，**lrd.ini** ファイル内の HP/UX 上の Oracle セクションは次のようになります。

```
[ORACLE_HPUX]
;816=liblrdhpo816.sl
;81=liblrdhpo81.sl
;80=liblrdhpo80.sl
73=liblrdhpo73.sl
72=liblrdhpo72.sl
```

この設定により，クライアントが Oracle 8.1.6 を使用していれば Mercury ライブラリ liblrdhpo816.sl を，Oracle 8.1.5 を使用していれば，liblrdhpo81.sl を LoadRunner が使用するということがわかります。

UNIX 上の再生では，lrd **ini** ファイルで使用するデータベースの正しいバージョンを表示するようにします。Oracle 8.1.5 を使用して HP/UX 用仮想ユーザを再

生するとします。その場合，Oracle のそのほかのバージョンを示す行は行の先頭を「;（セミコロン）」でコメントアウトします。すると lrd.ini ファイルは次のようになります。

```
[ORACLE_HPUX]
;816=liblrdhpo816.sl
81=liblrdhpo81.sl
;80=liblrdhpo80.sl
73=liblrdhpo73.sl
72=liblrdhpo72.sl
```

アプリケーションが lrd.ini ファイルに記されている DLL を使用していない場合，Win32 も変更します。たとえば，PowerBuilder 6.5 は Oracle 8.0.5 を使用しますが，DLL は ora803.dll を使用し，ora805.dll は使用しません。その場合，ORACLE_WINNT の項目で 805 および 804 をコメントアウトするか，805 のセクションを，

```
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，次項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 つめの要素です。この長さを増やせばスクリプトを正常に再生できるようになります。たとえばスクリプト内に次の行があるものとします。

```
lrd_assign(&_2_D354, " < ROW_ID > ", 0, 0, 0);
```

vdf.h ファイルで **_2_D354** を検索すると次のようになっています。

```
static LRD_VAR_DESC _2_D354 = {
        LRD_VAR_DESC_EYECAT, 1, 10, LRD_BYTYPE_ODBC,
        {0, 0, 0},  DT_SZ, 0, 0, 15, 12};
```

これを次のように変更します。

```
static LRD_VAR_DESC _2_D354 = {
        LRD_VAR_DESC_EYECAT, 1, 12, LRD_BYTYPE_ODBC,
        {0.0, 0, 0},  DT_SZ, 0, 0, 15, 12};
```

LRD_VAR_DESC の定義全体は **lrd.h** ファイルにあります。これを見つけるには「typedef struct LRD_VAR_DESC」で検索します。

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 がほかの仮想ユーザのものと干渉していることが考えられます。各仮想ユーザに同じ値で更新を行うように指定する場合を除いて，各仮想ユーザをパラメータ化してそれぞれの仮想ユーザが UPDATE 時に異なる値を使用するようにします。この場合，再試行論理を追加して UPDATE を再び試みます。

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？たとえば，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。たとえば，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。

LoadRunner 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，LoadRunner 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。下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 LoadRunner 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 LoadRunner 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：LoadRunner 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 LoadRunner 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。LoadRunner 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（たとえば契約など）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。LoadRunner 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，LoadRunner の Siebel のサポート向上のために，マーキュリー・インタラクティブのカスタマー・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 65 章

上級ユーザのために

本章には，LoadRunner 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。たとえば，**Vuser.c** は，**run.c** のコピーです。

Vuser.usr ファイルの例

```
[General]
Type=Oracle_NCA
DefaultCfg=default.cfg
AppName=C:¥PROGRA~1¥Netscape¥COMMUN~1¥Program¥netscape.exe
```

```
BuildTarget=
ParamRightBrace=>
ParamLeftBrace=<
NewFunctionHeader=0
MajorVersion=5
MinorVersion=0
ParameterFile=nca_test3.prm
GlobalParameterFile=
[Transactions]
Connect=
[Actions]
vuser_init=init.c
Actions=run.c
vuser_end=end.c
```

default.cfg ファイルの例

```
[General]
XlBridgeTimeout=120

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

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

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

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。

注：beta2 用のパッチ **cpp.exe** は，シェアウェアの実行ファイルで，非常に問題の多かった前のバージョンとはまったく異なります。

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。たとえば，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（LoadRunner 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

LoadRunner 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，たとえば待機時間，時間間隔，反復のループ，ログの記録などのユーザの動作を，仮想ユーザ・スクリプトを直接参照せずに設定できます。これにより，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。

LoadRunner コントローラから，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，890 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。LoadRunner 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）。LoadRunner コントローラを使用して，スクリプト・ウィンドウのコマンド・ライン・エントリにパラメータを追加することで，仮想ユーザにコマンド・ライン・パラメータを送ることができます。

VuGen から仮想ユーザを実行するときは，コマンド・ライン・パラメータを指定できません。ただし，Windows のコマンド・ラインでほかのすべてのドライバ・パラメータの後，つまり行の最後にパラメータを追加することでこれを手作業で指定できます。

```
mdrv.exe  -usr c:¥tmp¥Vuser¥Vuser.usr    -out c:¥tmp¥vuser
vuser_command_line_params
```

注：UNIX 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，次の手順で行います。

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，892 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 とコントローラの両方で使用されます。

```
[Templates]
RelativeDirectory=template
```

Templates セクションは，VuGen プロトコル用のテンプレートの場所を示します。標準のエントリは，これらのテンプレートが相対 **template** ディレクトリにあることを示します。各プロトコルには，**template** の下にサブディレクトリがあり，この中には，そのプロトコル用のテンプレート・ファイルが含まれています。

次のセクションは **GlobalFiles** セクションです。

```
[GlobalFiles]
main.c=main.c
@@TestName@@.usr=test.usr
default.cfg=test.cfg
default.usp=test.usp
```

GlobalFiles セクションには，新規テストが作成されたときに VuGen がテスト・ディレクトリにコピーしたファイルの一覧が含まれます。たとえば，「user1」というテストがある場合，VuGen は **main.c**，**user1.usr**，および **user1.cfg** をテスト・ディレクトリにコピーします。

ActionFiles セクションには，仮想ユーザによって実行されるアクションを含むファイルの名前と，反復を実行する仮想ユーザが含まれます。

```
[ActionFiles]
@@actionFile@@=action.c
```

上に示した設定に加え，**vugen.dat** には，オペレーティング・システムおよびコンパイルに関連するそのほかの設定が含まれます。

mdrv.dat

mdrv.dat ファイルには，ライブラリ・ファイルとドライバの実行可能ファイルの場所を定義する，プロトコル別のセクションがあります。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 ファイルを新しいプロトコルの設定を使って編集**します。
- **.cfg** ファイルの追加
- **lrp** ファイルを挿入
- テンプレート・ディレクトリの作成

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＞

たとえば，Oracle NCA 仮想ユーザ・タイプは，以下で表されます。

```
[Oracle_NCA]
ExtPriorityType=protocol
WINNT_EXT_LIBS=ncarp11i.dll
WIN95_EXT_LIBS=ncarp11i.dll
LINUX_EXT_LIBS=liboranca11i.so
SOLARIS_EXT_LIBS=liboranca11i.so
HPUX_EXT_LIBS=liboranca11i.sl
AIX_EXT_LIBS=liboranca11i.so
LibCfgFunc=oracle_gui_configure
UtilityExt=lrun_api,HttpEngine
ExtCmdLineOverwrite=-WinInet No
ExtCmdLineConc=-UsingWinInet No
SecurityRequirementsFiles=oracle_nca.asl
SecurityMode=On
```

VuGen は，コードに変更を加えずに新しい仮想ユーザ・タイプを処理できるように設計されています。ただし，特別なビューを追加しなければならない場合もあります。

VuGen では汎用のドライバは提供されていませんが，既存のドライバをカスタマイズできます。カスタマイズしたドライバを使用するには，**mdrv.dat** を変更します。プラットフォームと既存のドライバの行を追加した後，カスタマイズしたドライバの名前の行を「**＜プラットフォーム＞ _DLLS= ＜再生用 DLL 名＞**」の形式で追加します。たとえば，SAP の再生用 DLL が SAPPLAY32.DLL という名前の場合は，次の 2 行を **mdrv.dat** の [sap] セクションに追加します。

```
WINNT=sapdrv32.exe
WINNT_DLLS=sapplay32.dll
```

CFG ファイルの追加

標準設定の［実行環境の設定］および［記録オプション］を設定するには，プロトコルに任意で設定ファイルを指定できます。これを行うには，**mdrv.dat** ファイル内の **LibCfgFunc** 変数で定義するか，テンプレートの下の新しいプロトコル・サブディレクトリに **default.cfg** というファイルをおきます。サンプルの .cfg は次のとおりです。

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

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

[Log]
LogOptions=LogExtended
MsgClassData=0
MsgClassParameters=0
MsgClassFull=1
```

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，内部マクロの情報が記述されます。

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。たとえば，template¥http サブディレクトリには，**globals.h** というファイルがあり，include ディレクトリにある **as_web.h** ファイルをインクルードしています。

```
#include #as_web.h"
```

第 17 部

付録

付録 A

Java 環境：総合ガイド

本章では，Java 環境について説明します。既存の Java 環境を理解し，特定の Java アプリケーションに合わせて設定するために知っておくべき用語を解説します。Java 言語はクロス・プラットフォームに対応していますが，本付録では，Windows NT に関する事項だけを取り上げます。

- 用語
- JDK のバージョン
- ブラウザ
- Java Plug-in
- その他の環境
- よくある質問と回答

Java 環境について

Java プログラミング言語は，高水準のオブジェクト指向言語です。この言語は，開発が容易で分散型であり，インタプリタによる処理が可能であるとともに，その安全性，ポータビリティ，マルチ・スレッド型であることが知られています。

開発者にとっては Java は簡単な言語ですが，Java を取り巻く技術が発展するにつれ，そのための環境設定が複雑になっています。Java を開発した SUN Microsystems は，Java，API，およびツールの新しいバージョンをリリースし続けており，Java の世界を管理することがますます難しくなっています。

用語

JDK：**J**ava **D**evelopment **K**it（Java 開発キット）。SUN によって開発された，Java プログラムを作成するためのソフトウェア開発環境。JDK の各リリースには，Java コンパイラ，Java インタプリタ，Java クラス・ライブラリ，Java アプレット・ビューア，Java デバッガなどのツールが含まれています。JDK を再配布することはできません。

JRE：**J**ava **R**untime **E**nvironment（Java ランタイム環境）。ランタイム環境を再配布するエンド・ユーザおよび開発者向けの JDK のサブセット。JRE は，Java 仮想マシン，コア・クラス，および支援ファイルから成ります。

JVM：**J**ava **V**irtual **M**achine（Java 仮想マシン）。Java 実行環境のバイトコードの解釈を担う部分。

CLASSPATH：Java 仮想マシンにクラス・ファイルを探すべき場所を示す環境変数。CLASSPATH には，Java プログラムによって使用される JDK コア・ライブラリとクラス・ライブラリが含まれていなければなりません。クラスパス変数はいくつかの項目で構成され，各項目はセミコロンで区切られています。各項目は，ディレクトリ名またはアーカイブ名（jar または zip ファイル）です。カレント・ディレクトリを示す「.」も含まれていなければなりません。JDK 1.1.x の場合は，クラスパス変数に< **JDK** > **¥lib¥classes.zip** も含まれていなければなりません。CLASSPATH 内のクラスは，先頭から順番に検索されます。

PATH：ディレクトリのリストを含む環境変数。オペレーティング・システムは実行ファイルを作業ディレクトリ内で見つけられない場合，この PATH 内を検索します。ディレクトリ名の間はセミコロンで区切ります。JDK を使って作業する場合，JDK ツール（**java.exe**，**javac.exe**，**appletviewer.exe** など）を利用できるようにするために，< **JDK** > **¥bin** を PATH 変数に含める必要があります。

JAR：**J**ava **Ar**chive（Java アーカイブ）。Java アプレットまたはアプリケーションで必要なすべてのコンポーネントとリソース（クラス・ファイル，画像，サウンドなど）をまとめるために使用されるファイル形式。また，JAR はデータの圧縮をサポートしているので，ダウンロード時間が大幅に短縮されます。JAR ファイルには，**.jar** という拡張子を付けることになっています。JAR ファイルは，JDK に含まれている jar.exe ツールまたは標準の WinZip ツールによって内容の表示と取り出しができます。

JIT：**J**ust-**I**n-**T**ime Compiler（ジャストインタイム・コンパイラ）。JIT コンパイラは Java プログラムを実行するときに，「その場で（just in time）」すべてのバイトコードをマシンのネイティブ・コードに変換します。これによって，Java 仮想マシンがコードを解釈して実行するときよりも，実行速度が向上します。現在，JIT コンパイラは JVM のすべてのバージョンと各種ブラウザに含まれています。「(Compiled Code)」というテキストではなくスタック・トレース内の

行番号を取得したいときや，JIT によってバグが生じていると考えられるときなど，JIT コンパイラを無効にしたほうがよい場合もあります。

Applet：HTML ページに含めることができ，Java 対応ブラウザまたはアプレット・ビューアのコンテキストで実行できる Java プログラム。セキュリティ・マネージャが Java アプレットの動作に制限を設けます。

アプリケーション：スタンドアロンの実行プログラム。セキュリティまたは入出力に関する制限は（特別に指定しない限り）ありません。Java アプリケーションを実行するには，インタプリタが必要です。

バイトコード：Java コンパイラによって生成され，Java インタプリタによって実行されるマシンに依存しないコード。

クラス・ファイル：マシンに依存しない Java バイトコードを含むファイル。Java コンパイラは，Java インタプリタで読み取り可能な ***.class*** ファイルを生成します。

例外：プログラムの実行中に発生し，プログラムの通常のフローを継続不能にするイベント。一般的に，例外は予期しないエラーであり，コード内の例外が発生した場所を示すスタック・トレースが出力されます。プログラマは **try**，**catch**，**throw** というキーワードを使って，特定のタイプの例外に対処する例外ハンドラを作成できます。また，例外ハンドラの実行後にプログラムの実行を再開するかどうかを指定できます。

ガーベジ・コレクション：使用されていないメモリを自動的に検出して解放すること。Java ランタイム・システムがガーベジ・コレクションを実行するので，プログラマがオブジェクトを明示的に解放する必要はありません。

java.exe：SUN の JDK に含まれている Java インタプリタ。

appletviewer.exe：ブラウザを使用せずにアプレットを実行するためのツール。SUN の JDK に含まれています。アプレット・ビューアは HTML ページを引数として受け取り，その中の <Applet> タグを検索します。タグを見つけると，そのアプレットをロードします。

javap.exe：Java の **.class** ファイルの逆アセンブルを行うツール。SUN の JDK に含まれています。このツールは，ソースがない API クラスを検証するときに便利です。

javac.exe：SUN の JDK に含まれている Java コンパイラ。

jview.exe：Microsoft の Java 仮想マシン。Internet Explorer に組み込まれています。

HTML に含まれている属性：Java アプレットをロードする HTML ページには，アプレットのプロパティ（名前や場所など）の詳細を指定する <Applet> タグが含まれていなければなりません。以下に，このタグに含まれている，主なアプレット・クラス・ファイルを探すべき場所を指定する 3 つの属性を示します。

code 属性は，主なアプレット・クラス・ファイルの名前を指定します。

codebase 属性は，そのファイルとその他のファイルを格納しているディレクトリの URL を指定します。

archives 属性は，その他の任意のアーカイブ，つまりクラスおよびリソース・ファイルを含む，**jar**，**zip**，あるいは **cab** ファイルを指定します。これらのアーカイブはコードベースの下に置かれます。

classes.zip：JDK 1.1.x のすべての Java コア・クラスを含む zip ファイル。**＜JDK＞¥lib¥classes.zip** にあります。これらのクラスは Java アプリケーションの実行に不可欠なので，CLASSPATH に含まれていなければなりません。

rt.jar：JDK 1.2.x および JDK 1.3 のすべての Java コア・クラスを含む jar ファイル。**＜JDK＞¥jre¥lib¥rt.jar** にあります。この jar ファイルは JVM によって自動的に取得されるので，CLASSPATH に**含まれている必要はありません**。ただし，Xboot クラスパス（下記参照）が置換される場合には，rt.jar があることを確認しておく必要があります。

Java スクリプト：Web ページに動的な機能を組み込めるようにするための HTML の拡張。**Java** 言語という用語と混同しないでください。通常，Java スクリプトはブラウザ内の特殊なメカニズムによって解釈されます。これは Java 言語の仮想マシンとは無関係です。ただし，JavaScript 内から，Java の標準 JDK に含まれている静的メソッドを呼び出すことは可能です。この呼び出しによって，JVM とその DLL がロードされます。

JDK のバージョン

この項では，JDK（Java Development Kit）の各バージョンについて説明します。

JDK 1.1

JDK1.1 の **java.exe** 実行ファイルのコマンド・ライン引数は以下のとおりです。

コマンド・ライン引数	説明
-version	ビルドのバージョンを出力します。
-verbose	verbose モードを有効にして，ロードしようとしている各クラスの名前とパスに関する情報をシステム・クラス・ローダから出力します。
-noasyncgc	非同期のガーベジ・コレクションを無効にします。
-verbosegc	ガーベジ・コレクションが発生したときにメッセージを出力します。
-noclassgc	クラスのガーベジ・コレクションを無効にします。
-ms ＜数値＞	初期の Java ヒープ・サイズを設定します。たとえば，「-ms256M」のように指定します。
-mx ＜数値＞	最大の Java ヒープ・サイズを設定します。
-classpath ＜任意の数のディレクトリ（区切り文字は「;」）＞	CLASSPATH 変数の代わりに，クラスを検索するディレクトリのリストを記述します。
-noverify	クラスを検証しません。
-nojit	JIT コンパイラを無効にします。
-D ＜名前＞ = ＜値＞	システム・プロパティを設定します。

JDK1.1 の各バージョン：

初代の JDK であるバージョン 1.0.1 と 1.0.2 では，古いイベント・モデルを使用していました。JDK1.1 で，新しいイベント・モデルを導入しました。現在，JDK1.1.5 以下のバージョンには，SUN の公式なサポートはありません。

サポートされているバージョンは，1.1.6，1.1.7a，1.1.7b，および 1.1.8 です。JDK1.1.6 は不安定なバージョンであると言われています。その他のバージョンに関しては，互いに多少異なり，主にセキュリティ上の問題の修正について違いがあります。

JDK 1.1 の例：

次の例では，JIT コンパイラを使用せずにアプリケーション TestApp を実行し，jdk とカレント・ディレクトリからクラスを取得する方法を示します。

```
java -nojit -classpath d:/jdk1.1.7b/lib/classes.zip;. TestApp
```

次の例では，初期および最大の Java ヒープ・サイズを 256M に設定して，TestApp アプリケーションを実行する方法を示します。テスト・アプリケーションには 2 つの引数があります。クラスはグローバルな CLASSPATH 変数から取得されます。

```
java -ms256M -mx256M TestApp apparg1 apparg2
```

次の例では，PATH に含まれている現在の Java のバージョンを調べる方法を示します。

```
java -version
```

JDK 1.2

JDK1.1.x では，クラスの検索は CLASSPATH 環境変数に従って行われていました。アプレットの場合も，クラスの検索は HTML で指定されたコードベースとアーカイブで行われました。JDK1.2.x では，「ブートストラップ・クラスパス」という新しい要素でクラスの場所が指定されます。**ブートストラップ・クラスパス**は，クラスパスの前に置かれます。標準では，**＜ JDK ＞ ¥jre¥lib¥rt.jar.** にあり，JDK コア・クラスが含まれています。ブートストラップ・クラスパスは，**-Xbootclasspath** コマンド・ライン変数を使用して変更できます。さらに，**javac** では，コンパイルするプラットフォームのクラスの変更に使用できる同様のオプション（「-bootclasspath」）がサポートされています。

JDK1.2.x には，JFC1.1 ライブラリのクラスと Java IDL CORBA のクラスが含まれています。これらはすべて，システムの **rt.jar** ファイルに含まれています。

JDK1.2 以上には，2 種類の仮想マシンがあります。1 つは「**Classic**」仮想マシンで，**＜ JDK ＞ /jre/bin/classic/jvm.dll** にあり，もう 1 つは「**HotSpot**」仮想マシンで，**＜ JDK ＞ /jre/bin/hotspot/jvm.dll** にあります。HotSpot 技術は，Classic VM の技術よりも優れたパフォーマンスを提供します。Classic VM を使用するには，Java コマンド・ライン・オプション **-classic** を使用します。

JDK1.2 では，「**java.exe - jar ＜ jar ファイル名＞**」のように jar の名前を指定するだけで Java アプリケーションを実行できます。この場合，JVM は jar を開き，マニフェスト・ファイル（Manifest.mf）を検索します。このファイルの中で，main メソッドを含んでいるクラスの名前を指定する必要があります。JVM は指定されたクラスをロードし，main メソッドを実行します。

また JDK 1.2 は，機能，パフォーマンス，セキュリティ，グローバル・サポートにおいて，以前のバージョンの Java プラットフォームよりも優れています。

次の表に JDK 1.2 の java.exe の非標準コマンド・ライン・オプションを示します。

コマンド・ライン引数	説明
-Xbootclasspath: ＜セミコロン（;）で区切られたディレクトリと zip または jar ファイルのリスト）＞	ブートストラップのクラスとリソースを検索するパスを設定します。
-Xbootclasspath/p: ＜セミコロン（;）で区切られたディレクトリと zip または jar ファイルのリスト＞	ブートストラップのクラスとリソースを検索する標準パスを前置します。
-Xbootclasspath/a: ＜セミコロン（;）で区切られたディレクトリと zip または jar ファイルのリスト＞	ブートストラップのクラスとリソースを検索する標準パスを後置します。
-Xnoclassgc	クラスのガーベジ・コレクションを無効にします。
-Xms ＜サイズ＞	初期の Java ヒープ・サイズを設定します。 たとえば，「-Xms128M」のように指定します。
-Xmx ＜サイズ＞	最大の Java ヒープ・サイズを設定します。
-Xrunhprof[:help][: ＜オプション＞ = ＜値＞ , ...]	ヒープ，CPU，またはモニタのプロファイリングを行います。

-Xbootclasspath の使い方

-Xbootclasspath パラメータは非常に強力なので，使用には注意が必要です。適切な Java プラットフォームのクラスが使用されていることを確認する必要があります。これらのクラスは**＜ JDK ＞ ¥jre¥lib¥rt.jar** にあります。

-Xbootclasspath: ＜ ... ＞を使用する場合は，rt.jar ファイルのフル・パスを指定する必要があります。指定しなければ VM を起動できません。また，このパラメータでは JDK の別のバージョンのクラスは指定できません。プリペンド（/p）またはアペンド（/a）の形式を使用する場合は，rt.jar のパスを省略できます。

パラメータの構文は，CLASSPATH 環境変数を使用した場合と似ています。項目に空白文字が含まれるときは引用符を使用する必要があります。

JDK1.2.x のインストール

JDK1.2.x をインストールすると，java.exe ファイルが**＜ Winnt ＞ ¥system32** ディレクトリに置かれるため，PATH 変数に**＜ JDK ＞ ¥bin** ディレクトリを指定する必要はありません。このことは，特に JDK のその他のバージョンを使用する場合に注意する必要があります。**＜ Winnt ＞ ¥system32** ディレクトリは PATH 変数にあり，その他のバージョンの**＜ JDK ＞ ¥bin** ディレクトリの前に現れます。このため，JDK のバージョンを切り替えたときに，予期しないランタイムの競合（JVM に対応しないランタイム・クラス）が生じる可能性が高くなります。

JDK1.2 の各バージョン：

既存の JDK1.2 のバージョンには，1.2，1.2.1，および 1.2.2 があります。バージョンごとに，多少の違いがあります。

JDK 1.2 の例：

次の例では，TestApp アプリケーションを実行し，JDK1.2.2 Java プラットフォームのクラスの前に LoadRunner のクラスを使用しています。これらの 2 つのステートメントは同じタスクを実行します。

```
java -Xbootclasspath:c:¥LoadRunner¥classes;c:¥jdk1.2.2¥jre¥lib¥rt.jar
TestApp
java -Xbootclasspath/p:c:¥LoadRunner¥classes TestApp
```

次の例では，初期および最大の Java ヒープ・サイズを 256M に設定して，TestApp アプリケーションを実行します。-D パラメータを使用して，Java アプリケーションのシステム・プロパティを設定します。

```
java -Xms256M -Xmx256M -Dtestprop=propvalue  TestApp
```

JDK 1.3

JDK1.3 の以前のバージョンとの大きな違いは，その仮想マシン（VM）です。JDK1.3 では，Java HotSpot Client VM が標準実装の VM となっています。Classic VM に切り替えるには，**-classic** オプションを使用します。Java Plug-in 1.3 VM は HotSpot VM だけを提供します。プラグインで Classic VM を使用するには，JDK ディレクトリから< **JDK** > **/jre/bin/classic** ディレクトリを< plug-in > /**bin** ディレクトリにコピーし，コントロール・パネルで -classic フラグを設定します。

JNDI のクラスと RMI/IIOP は，**rt.jar** ファイルに含まれています。

JDK1.3 は，機能，パフォーマンス，GUI，セキュリティ，グローバル・サポートにおいて，以前のバージョンの Java プラットフォームよりも優れています。

JDK1.3 の各バージョン：

JDK1.3 のバージョンとしては，1.3 と 1.3.0_01 があります。この 2 つの違いは小さなものです。

JDK1.3 の例：

次の例では，JIT コンパイラを使用せずに TestApp アプリケーションを実行し，verbose がクラスのソース・ディレクトリを出力します。

```
java -verbose -Djava.compiler=NONE TestApp
```

次の例では，Classic VM を使って TestApp アプリケーションを実行します。

```
java -classic TestApp
```

ブラウザ

この項では，Netscape と Internet Explorer を使って，Java アプレットまたはアプリケーションを実行する方法を説明します。

アプレット・ビューア

アプレット・ビューアは，ブラウザを使用せずに Java アプレットを実行するためのコマンド・ライン・ツールです。アプレット・ビューアは，<Applet> タグを含む URL アドレス（HTML またはファイル）を認識します。

アプレット・ビューア・ツールはほとんどの java.exe コマンド・ライン・パラメータをサポートしていますが，これらのパラメータを使用するには，接頭辞「-J」を使用する必要があります。たとえば，**-J-verbose**, **-J-Xbootclasspath: ＜ ... ＞ ,** のように指定します。

アプレット・ビューアの例

次の例では，Web 上の HTML ファイルから参照されているアプレットを実行します。

```
appletviewer http://www.apptest.com/test/test.html
```

次の例では，ローカルの HTML ファイルから参照されているアプレットを実行し，rt.jar にクラスを前置します（JDK1.2 以上の場合のみ）。

```
appletviewer -J-Xbootclasspath/p:c:¥LoadRunner¥classes
file:c:¥apptest¥test.html
```

次の例では，カレント・ディレクトリにある HTML ファイルから参照されているアプレットを実行し，verbose の出力としてクラスが表示されます。また，最大および最小のヒープ・サイズを指定します（JDK1.1 形式）。

```
appletviewer -J-verbose -J-ms256M -J-mx256M test.html
```

Internet Explorer

Java コンソール

Java アプレットによって **stdout** または **stderr** に送信される出力メッセージは，Internet Explorer の Java コンソールに出力されます。ブラウザの［**表示**］メニューを使って，Java コンソールを開きます。

ブラウザに Java コンソール・オプションが表示されない場合は，Internet Explorer の設定で Java コンソール・オプションを有効にする必要があります。これを行うには，［Internet Explorer］アイコンを右クリックして［プロパティ］オプションを選択します。次に［詳細設定］タブで，Java VM セクションを見つけます。［Java コンソールの使用］チェック・ボックスを選択し，ブラウザを再起動します。

Internet Explorer Java 仮想マシンのバージョン

Internet Explorer にインストールされているコンポーネントの 1 つに，Microsoft Java VM があります。使用しているブラウザにインストールされている JVM のバージョンは，ブラウザのバージョンとは別個のものです。バージョンは Java コンソールの最初の行に示されています。たとえば，「Microsoft (R) VM for Java, 5.0 Release 5.0.0.3176」は IE5 をサポートする JVM があり，そのビルド番号は 3176 であることを示します。

Internet Explorer の JVM コンポーネントだけを更新することもできますが，ダウングレードはできません。

Internet Explorer の Java クラス

Internet Explorer の JVM には，固有のシステム・クラスがあります。これらのクラスは，＜ **Winnt** ＞**¥Java¥JPackages** ディレクトリにある zip ファイルに含まれています。これらの zip ファイルはすべて，ブラウザによって自動的に CLASSPATH に挿入されます。

zip ファイルをダウンロードして，ブラウザのディレクトリに配置するアプレットもあります。これらの zip ファイルに含まれているクラスによって，競合が生じることがあります。

Netscape

Java コンソール

Java アプレットによって **stdout** または **stderr** に送信される出力メッセージは，Netscape の Java コンソールに出力されます。Java コンソールを開くには，［**ツール**］＞［**Web 開発**］＞［**Java コンソール**］を選択します。

Java コンソールには，キーボードで特定の文字を入力することによって起動できるオプションがあります。すべてのオプションを表示するには，「**?**」を入力します。次のオプションがあります。

b : デバッガに割り込みます（Windows のみ）。
c : コンソール・ウィンドウをクリア。
d : アプレットのコンテキストの状態をコンソールにダンプ。
f : ファイナライズ・キューのオブジェクトをファイナライズ。
g : ガーベジ・コレクト。
h : このヘルプ・メッセージを表示。
l : アプレットによってディレクトリにロードされたすべてのクラスをキャプチャ。
m : 現在のメモリ使用率をコンソールに表示。
q : コンソールを非表示。
s : メモリ・サマリを **memory.out** にダンプ。
t : スレッド情報を **memory.out** にダンプ。
x : メモリを **memory.out** にダンプ。
X : メモリ（詳細）を **memory.out** にダンプ。
0-9: アプレットのデバッグ・レベルを＜ n ＞に設定。

コンソールは頻繁にリフレッシュされるため，コンソールからテキストをコピーするのは困難です。コピーするテキストの範囲を選択しても，ブラウザが出力を送出すると，それによって選択したテキストが置き換えられてしまいます。ブラウザに送出するものが何もなくなるまで待ってみます。

Netscape の Java クラス

Netscape には，独自の JVM クラスが含まれています。これらのクラスは，＜**Netscape フォルダ＞ ¥Communicator¥Programs¥java¥classes** あるすべての jar ファイルに含まれています。Java のシステム・クラスを含む主要な jar ファイルは，java40.jar です。また，Netscape の jar ファイル **iiop10.jar** には，CORBA Visigenic3.0 のクラスが含まれています。この jar ファイルによって，他の CORBA クラスを使用する CORBA アプリケーションで競合が生じることがあります。したがって，iiop10.jar をこのディレクトリから削除するほうがよい場合もあります。これらの jar ファイルはすべて，ブラウザによって自動的に CLASSPATH に挿入されます。

環境によっては，Netscape ではなく，別のアプリケーションをインストールしたときに置かれた jar ファイルが含まれている場合があります。これらの jar ファイルに含まれているクラスによって，競合が生じることもあります。verbose フラグ（デバッグ・レベル：0 ～ 9）を分析することによって，特定のクラスのソースを理解することができます。

Java Plug-in

Java Plug-in は，Internet Explorer または Netscape で SUN の JRE を使用して，ブラウザ内部の JVM を使用せずに Java アプレットを実行できるようにするためのツールです。Web ブラウザが Java Plug-in を使用するように指定している Web ページに初めて遭遇したときには，必要なファイルをダウンロードしてインストールしなければなりません。その後 同様の Web ページに遭遇したときには，Java Plug-in がユーザのハード・ディスクから直ちに起動され，SUN のインストール済みの JRE でアプレットが実行されます。

Java Plug-in のコントロール・パネル

Java Plug-in には，いくつかのオプションを設定できるコントロール・パネルが含まれています。コントロール・パネルは，［**スタート**］＞［**プログラム**］＞［**Java Plug-in Control Panel**］から開きます。

設定可能なオプションには，Java コンソールと JIT コンパイラの有効化，コマンド・ライン・パラメータの使用，および SUN の JVM ランタイムのバージョンの変更などがあります。

Plug-in 1.3 からは，Windows の［コントロール パネル］で［**Java Plug-in**］項目を選択することによって，［Java Plug-in コントロールパネル］を開くことができます。

Java コンソール

Java アプレットによって **stdout** または **stderr** に送信される出力メッセージは，Java Plug-in の Java コンソールに出力されます。Java コンソールの有効化は，［Java Plug-in コントロールパネル］の［基本］タブで行います。Plug-in を使ってアプレットを実行しているときは，起動する必要があるコンソールはブラウザ・コンソールではなく，Plug-in コンソールだけです。

Java Plug-in の JVM のバージョン

Java Plug-in を使用しているときは，SUN JVM のバージョンを切り替えることができます。実行する JVM は，［Java Plug-in コントロールパネル］の［**詳細**］タブで選択します。マシンにインストールされている JRE または JDK ならばどれでも使用できます。Java Plug-in で現在稼動しているバージョンを確認するには，Java コンソールの最初の 2 行を調べます。たとえば，1.1.7B JVM を実行している Java Plug-in 1.2.2 では，以下のように出力されます。

```
Java(TM) Plug-in:Version 1.2.2.px
Using JRE version 1.1.7B
```

HTML の Java Plug-in タグ

Java Plug-in を使用する必要がある Web ページをブラウザが識別できるようにするには，<Applet> タグを特定のタグで置き換える必要があります。各ブラウザには，Java Plug-in を制御するための独自の HTML タグがあります。Internet Explorer は，<Object> タグに指定されている Java Plug-in とアプレット属性を調べ，Microsoft の COM/ActiveX 技術を使用して，アプレットをロードします。Netscape Navigator は，<Embed> タグに指定されている Java Plug-in とアプレット属性を調べ，Navigator のプラグイン・アーキテクチャを使用して，アプレットをロードします。HTML ページに両方のタグを含めることによって，Netscape と Internet Explorer のどちらからでも，Java Plug-in を通じてアプレットがロードされるようにできます。

Java Plug-in HTML Converter と呼ばれる簡単なツールを使って，HTML ページで Java Plug-in を使えるようにする変換を容易に行えます。

その他の環境

IBM

IBM 独自の Java ツールがあります。これには Java コンパイラ，仮想マシン，および少し変更を加えられた JDK ランタイム・クラスが含まれています。

Oracle Jinitiator

Oracle には Jinitiator と呼ばれる独自の仮想マシンがあります。この仮想マシンは，アプリケーション用にも，ブラウザのプラグインとしても使用できます。これの JDK は，SUN の JDK 1.1.5 または JDK 1.1.7 を基本に，様々な変更が加えられたものです。この仮想マシンは「**java -version**」で識別できます。これを実行すると，末尾に「o」が付いた JDK のバージョンが表示されます（たとえば，「java version 1.1.5o」）。Jinitiator プラグインは，HTML ページを見るか，**＜ドライブ＞:¥Program Files¥Oracle¥Jinitiator ＜バージョン＞**ディレクトリを調べることによって識別できます。

BEA WebLogic

BEA には仮想マシンはありませんが独自のツール群があります。また，独自の RMI コンパイラ（rmic.exe）があります。これは，SUN の「**rmic**」ツールによって生成されるものとは異なるスタブとスケルトンを生成するために使用されます。

VBJ および OWJAVA

Visigenic と Iona には，独自の java.exe があります。これらは通常，CORBA のパッケージをインストールしたときに同時にインストールされます。これらは新しく実装された仮想マシンではなく，SUN の仮想マシンに CORBA 関連の機能を追加するラッパーというべきものです。SUN の仮想マシンと同じ引数を取ります。

Gemstone/J

Gemstone には，独自バージョンの JVM と JDK があります。java -version コマンドを実行すると，次のメッセージが表示されます。

```
java version "3.2p2"
HotSpot VM (3.2p2, mixed mode, build 3.2p2-Wed-Feb-23-17:58:31-PST-2000-build-97)
```

よくある質問と回答

質問 1 : アプリケーションの出力をファイルにリダイレクトする方法は？

Java アプリケーションの **stdout** と **stderr** をファイルにリダイレクトするには，java.exe コマンド・ラインの最後に「**>out.txt 2>&1**」を追加します。これによって，出力が out.txt ファイルにリダイレクトされます（NT のみ）。

回答 : Internet Explorer では，Java ログ記録オプションを有効にすることによって，出力をリダイレクトできます。Java ログ記録オプションを有効にするには，［Internet Explorer］アイコンを右クリックして［**プロパティ**］を選択します。次に［**詳細設定**］タブの［**Microsoft VM**］セクションで，［**Java のログを使用**］チェック・ボックスを選択します。ブラウザを再起動します。

これによって，出力が**＜ Winnt ＞ \Java\javalog.txt** ファイルにリダイレクトされます。

質問 2 : あるクラスがクラスパスに含まれているかどうかを調べる方法は？

回答 : javap.exe コマンド・ライン・ツールを使用します。javap.exe ＜完全なクラス名＞を実行します（接尾辞「.class」は付けません）。指定したクラスが CLASSPATH にない場合は，「Class ＜クラス名＞ not found」というエラー・メッセージが表示されます。クラスが CLASSPATH にある場合は，そのフィールドとメソッドが表示されます。

質問 3 : ブラウザからクラスをダンプする方法は？

回答 : Netscape では，アプレットの実行中にサーバのクラスをキャプチャして，ローカル・マシン上にそれらのクラスをダンプできます。Java コンソール上で文字「L」を入力することによって，このオプションを有効にしたり無効にしたりできます。Java コンソールに「# Class file capture enabled」というメッセージが表示され，サーバのすべてのクラスが**＜ Netscape フォルダ＞ ¥Communicator¥Programs¥ ＜ URL から構成されるディレクトリ名＞**にダンプされます。

質問 4 : ブラウザに Java コンソールを開くためのオプションが見当たりません。どこにあるのでしょうか？

回答 : Internet Explorer では，［**表示**］メニューにこのオプションがない場合は，Internet Explorer の設定でこのオプションを有効にする必要があります。オプションを有効にするには，［Internet Explorer］アイコンを右クリックして［**プロ**

パティ］を選択します。次に［詳細設定］タブの［Java VM］セクションで，［Java コンソールの使用］チェック・ボックスを選択します。ブラウザを再起動します。Netscape では，このオプションは通常［**Communicator**］＞［**ツール**］メニューにあります。以前のバージョンでは，［**ツール**］または［**Communicator**］メニューにありました。

質問 5 : Java Plug-in を使用して実行しているときに，ブラウザの Java コンソールに出力が表示されない原因は？

回答 : ブラウザの Java コンソールは，ブラウザの VM から送信されたメッセージだけを表示します。Java Plug-in を使用しているときは，Java Plug-in の Java コンソールを使用する必要があります。

質問 6 : Java Plug-in を使用しているときに，Java コンソールを開く方法は？

回答 : プラグインのコントロール・パネルで，［スタート］＞［プログラム］＞［The Java Plug-in Control Panel］を選択します。［Basic］タブで，［Show Java Console］チェック・ボックスを選択します。

質問 7 : マシンに JDK1.1 と JDK 1.2 の両方をインストールしています。「PATH」には＜ **JDK1.1** ＞ **¥bin** が指定されています。「**java-version**」を実行するとまだ JDK1.2 が見える原因は？ JDK1.2 をアンインストールせずに無効にする方法は？

回答 : JDK1.2.x をインストールする際に，その java.exe も＜ **Winnt** ＞ **¥system32** ディレクトリに置かれます。このディレクトリも PATH に含まれているため，JDK1.2 の java.exe が使用されます。JDK1.2 をアンインストールせずに無効にするには，**¥system32** ディレクトリの java.exe の名前を変更するか，このディレクトリから java.exe を削除します。

質問 8 : stack-trace を使うと，エラーが発生します。この場合に JIT が稼動していることを確認する方法は？

回答 : stack-trace に，エラーの原因となった呼び出しのスタックが含まれています。各行に，呼び出されたクラスとメソッドの名前が含まれています。呼び出しの行番号は，括弧で囲まれて表示されます。括弧内に含まれているのが行番号ではなく「Compiled Code」であれば，バイトコードがネイティブ・コードにコンパイルされているので，JIT は稼動しています。

質問 9：java.exe で JIT を無効にする方法は？

回答： java.exe の「-nojit」コマンド・ライン・オプションを使用します。JDK1.2.x および JDK1.3 では，-nojit オプションはありません。これらのバージョンでは，java.exe のコマンド・ラインに次の引数を指定します。

-Djava.compiler=NONE

質問 10：ブラウザで JIT を無効にする方法は？

回答：JIT を無効にする方法は，ブラウザによって異なります。

Internet Explorer ― ［Internet Explorer］アイコンを右クリックして［**プロパティ**］を選択します。［**詳細設定**］タブの［**Java VM**］セクションで，［Java JIT コンパイラの使用（再起動が必要)］チェック・ボックスをクリアします。ブラウザを再起動します。

Netscape ― JIT の DLL の名前を変えて，それが見つからないようにします。DLL へのフル・パスは，**＜ Netscape フォルダ＞¥Communicator¥Programs¥java¥bin¥jit3240.dll** です。

質問 11：次のエラーが生じる原因は？
「NoClassDefFoundError?」

「Can't find class java.lang.NoClassDefFoundError.(Wrong class path?)」

「Exception in thread "main" java.lang.NoClassDefFoundError:Files」

回答：これらのエラー・メッセージが表示される場合の多くは，JDK1.2 の **-Xbootclasspath** 引数の指定が正しくありません。**＜ JDK ＞ ¥jre¥lib¥rt.jar** にある rt.jar ファイルへのフル・パスが指定されているかどうか確認します。また，パラメータに空白が含まれているかどうかを調べます。空白が含まれている場合は，次のようにパラメータを引用符で囲む必要があります。

-Xbootclasspath:” ＜ ... ＞ ”

質問 12：「Unable to initialize threads」エラーが生じる原因は？

回答： JDK1.1.x を使用している場合は，CLASSAPATH 環境変数に**＜ JDK ＞ ¥lib¥classes.zip** が含まれていることを確認します。JDK1.2.x で **-Xbootclasspath** オプションを使用している場合は，**＜ JDK ＞ ¥jre¥lib¥rt.jar** にある rt.jar ファイルのフル・パスが指定されていることを確認します。

質問 13：「UnsatisfiedLinkError」エラーが生じる原因は？

このエラーは Java アプリケーションが特定のライブラリ（または DLL）のネイティブ・メソッドを使用しているときに，そのライブラリ・ファイルが見つからないか，壊れているか，Java のセキュリティ上の制約で特定のクラスからアクセスできない場合に発生します。

質問 14：「OutOfMemoryError」の解決方法は？

回答： アプリケーションの実行中に仮想マシンのスタック・サイズまたはヒープ・サイズが不足すると，OutOfMemoryError が発生します。ほとんどの仮想マシンでは，ヒープとスタックのサイズが 16 M 〜 64 M に設定されます。アプリケーションによっては，メモリの使用量が多いか，長大な再帰呼び出しがあるために，それよりも多くのメモリが必要になることがあります。多数のクラス（または jar）をロードするアプリケーションも起動時に多くのメモリを消費します。この問題を解決するには，仮想マシンを「-ms」と「-mx」引数を使って実行してください（JDK 1.2 の場合は「-Xms」と「-Xmx」）スタックとヒープの最大サイズは 512M です（仮想メモリ）。

注：本項で取り上げた問題は，短期間で状況が大きく変わる可能性があります。最新情報を維持すべく取り組んでおりますが，お気づきの点がございましたら documentation@merc-int.com までお知らせいただければ幸いです。

付録 B

EJB アーキテクチャとテスト

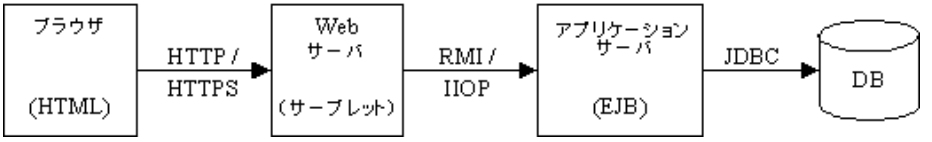
EJB について

EJB（Enterprise Java Beans）は，コンポーネント・ベースの分散ビジネス・アプリケーションの開発と配備を行うためのコンポーネント・アーキテクチャです。

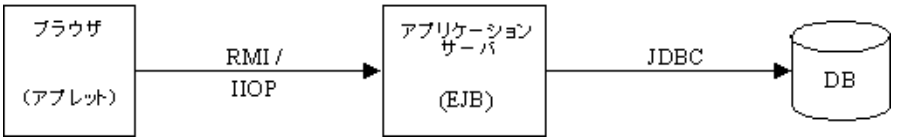
EJB システムは複雑な企業向けミドルウェア機能を提供し，開発者はそのモデルを使って実際のビジネス・アーキテクチャに専念し，さまざまな規模に対応できる企業向けの安全な商取引用マルチユーザ・アプリケーションを短期間で構築できます。

EJB アーキテクチャ

今日の情報処理システムの多くは，厳密に定義された 4 階層アーキテクチャまたは n 階層アーキテクチャを使用して構築されています。これらのアーキテクチャでは一般的に，Java 技術に基づくコンポーネントが使用されています。Web サーバでサーブレットが，アプリケーション・サーバで EJB が，それぞれどのように使われているかについては，下の図を参照してください。EJB 自体は，追加の階層を含むエンティティ EJB またはデータベースと対話する JDBC を直接使用する エンティティ EJB を使用します。



Web サーバとアプリケーション・サーバが同じホスト・マシン上で実行される 3 階層アーキテクチャを実装しているシステムもあります。この 2 つの機能は同じホストで実行されますが，それでもサーブレットと EJB を使用して機能します。あまり一般的ではありませんが，RMI や IIOP，または RMI-over-IIOP プロトコルを使用してアプリケーション・サーバの EJB と直接対話する Java アプレットをブラウザが読み込む形の「純粋な」3 階層アーキテクチャのシステムもあります。



EJB のアーキテクチャとメカニズム

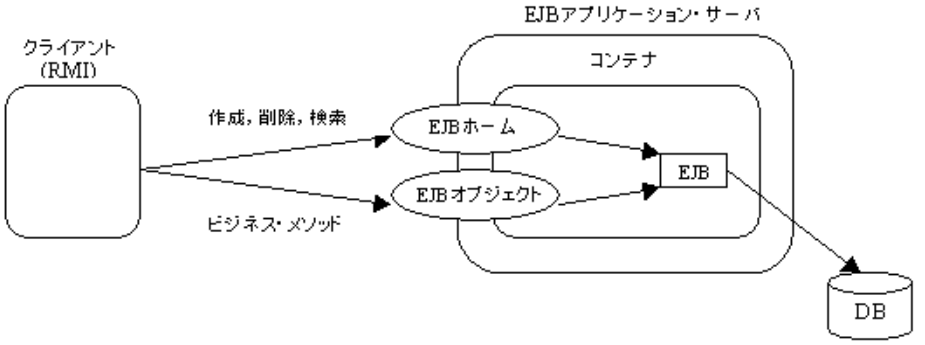
単純な EJB は，3 種類の基本的な Java オブジェクト（Java クラス）で構成されます。これらのクラスは EJB の開発者によって作成され，EJB とその機能を記述します。

Java オブジェクトには，ホーム・インタフェース，リモート・インタフェース，Bean 実装の 3 種類があります。

- **ホーム・インタフェース**：create()，remove()，find() などのメソッドによって EJB のライフ・サイクル・サービスを提供します。
- **リモート・インタフェース**：EJB のビジネス操作を含みます。
- **Bean 実装**：ホーム・インタフェースとリモート・インタフェースのメソッドを実装しています。

開発者は，これらのオブジェクトを配備記述子ファイルおよび EJB コンパイラ（ejbc）とともに使用して，ビジネス・メソッドと呼ばれる EJB 関数へのリモート・アクセスに必要なスタブとスケルトンを生成します。EJB コンパイラは，WinZip がファイルとディレクトリを格納する場合と同様に，Bean 全体を JAR または EAR と呼ばれるアーカイブ・ファイルに格納します。生成されたスタブとスケルトンが，すべての下位レベルの通信を包み込んでプログラマから切り離すので，EJB オブジェクトではクライアント・サーバ間の通信に関する詳細を意識する必要はありません。EJB をアプリケーション・サーバに配備する際に，これらのスタブおよびスケルトン・クラスは，サーバの EJB コンテナが EJB をそれに関係するクライアントまたは他の EJB と接続するために使われます。このコンテナには，トランザクションやセキュリティなどの EJB サービスのためのクラスもあります。

アプリケーション・サーバは起動時に，システムに配備される各 EJB の JNDI ツリーの中に，ホーム・スタブへの名前参照を追加または配備します。JNDI（Java Naming Directory Interface）は Java 言語の拡張で，ハード・ディスク・ドライブのファイル構造に似たツリー構造の名前による，抽象化されたリソース参照手段を提供します。これらのホーム・スタブは，JNDI を通じてアプリケーション・サーバのクライアントで使用できるようになります。各クライアントは，クライアント・マシンでローカルに使用できるように，ホーム・スタブのコピーをダウンロードします。クライアントはホーム・スタブを使用して，アプリケーション・サーバ上に EJB インスタンスを作成します。アプリケーション・サーバ上で，ホーム・スケルトン（コンテナによって制御される）は EJB インスタンスの作成，ステート情報の付与，クライアント・コンテキストへの関連付け，リモート・スケルトンの作成，およびリモート・スタブの作成を行います。ホーム・スケルトンは，クライアントにリモート・スタブを返します。クライアントは，このリモート・スタブをローカルで使用して，EJB 上のビジネス・メソッドを呼び出せます。



配備可能な EJB JAR は，ホーム・インタフェース，リモート・インタフェース，Bean 実装，スタブおよびスケルトン・クラス，配備記述子ファイル，コンテナのクラス・パスに含まれていない任意の参照 Java クラス・ファイルを含みます（クラス・パスは，Java がクラス・ファイルおよびクラス・ファイルを含んでいるアーカイブの位置を特定するために使用する環境変数です）。配備記述子ファイルには，EJB の名前，クラス，ホームおよびリモート・インタフェース，Bean の種類（セッションまたはエンティティ），環境エントリ，リソース・ファクトリ参照，EJB 参照，JNDI 名などの情報が含まれます。

次の 2 種類の配備記述子があります。

- **直列化された配備記述子**：EJB に関する直列化された情報を含むファイル。ファイルの拡張子は「.ser」です（EJB 1.0 仕様書参照）。
- **XML 配備記述子**：JAR ファイル内のすべての EJB オブジェクトに関する情報を含む XML ファイル。ファイル名は ejb-jar.xml です。これらの記述子は，jar または ear ファイルの META-INF ディレクトリに格納されます。EJB コンテナには，EJB の配備に必要な追加の配備記述子ファイルを持つものもあり，たとえば，Weblogic では，weblogic-ejb-jar.xml ファイルが必要です（EJB 1.1 仕様書参照）。

次の 3 種類の EJB オブジェクトがあります。

- **Stateless Session Bean**：常に同じサービスを提供し，クライアントからの複数の呼び出しの間でステート情報を維持しません。
- **Stateful Session Bean**：対話的なやり取りを，複数の呼び出しの間でステート情報を維持します。
- **Entity Bean**：永続的な EJB で，一般にデータベース内のデータを表すために使用されます。

共有アプリケーション・サーバの配備

バッファのタイプ	EJB 仕様書	配備記述子	配備方法
WebLogic4.x	1.0	.ser ファイル	— ejbc ツールを使用して配備可能な Jar ファイルを作成 —< WL > ¥weblogic.properties プロパティ内に次のような EJB JAR ファイルへの完全パスを指定して配備。 weblogic.ejb.deploy
WebLogic5.x	1.1	ejb-jar.xmlweblogic-ejb-jar.xml	
WebLogic6.x	1.1, 2.0	ejb-jar.xmlweblogic-ejb-jar.xml	— ejbc ツールを使用して配備可能な Jar ファイルを作成 — 次のディレクトリに EJB JAR/EAR をコピーすることによって配備。 < WL > ¥config¥ < domain > ¥applications
WebSphere3.0	1.0	.ser ファイル	— jetace ツールを使用して配備可能な Jar を作成 — 管理コンソール UI により配備（配備した EJB がディレクトリに置かれます。 < WS > ¥deployedEJB）
WebSphere3.5	1.0，1.1 の一部		
WebSphere 4.0	1.0, 1.1	ejb-jar.xml	— アプリケーション・アセンブリ・ツールを使用して Bean を作成し配備。
Oracle OC4J	1.1，2.0 の一部	ejb-jar.xmlorion-ejb-jar.xml	— .ear ファイルまたは .ear ファイル構造に従って構成されたディレクトリを作成 — .ear¥directory を server.xml 設定ファイルに登録するか，OC4J がインストールされているアプリケーション・ディレクトリの下に配置。
Sun J2EE	1.1	ejb-jar.xml	— .ear ファイルを作成し，アプリケーション配備ツールをしようしてアプリケーションを配備。

EJB 単体テスト

今までの EJB のテストは，RMI 仮想ユーザの一部として，GUI の操作で動く EJB **Home** と Remote のインタフェース API を記録することによって行われてきました。このテストは，クライアントが実装されていなければ成立しません。そのような場合には，Java 仮想ユーザを使って EJB テスト・スクリプトを作成していました。後者のソリューションは複雑で，ソースコードの内容をすべて理解し，自由に利用できなければなりません。

顧客によっては，システムの構築をアプリケーション・サーバから始めて，その上位層に向けて行う場合もあると想定するのは合理的です。そうした顧客は，アプリケーション・サーバ側を先に実装し，そのスケーラビリティを確認して初めて，Web サーバやクライアント側などのバックエンドを実装します。LoadRunner は，VuGen 単体のパラメータ化機能と Java 機能によって，開発の初期段階から EJB のメソッドとインタフェースの機能をテストできます。この段階ではユーザ・インタフェースがないため，WinRunner のような従来の機能テスト・ツールが使用できません。さらに，LoadRunner は，EJB 仮想ユーザとコントローラ，アナリシスとを組み合わせて，スケーラビリティ問題のテストを支援することができます。

EJB 単体テスト・ソリューションによって，アプリケーション・サーバにインストールされた EJB 全体を概観することができると同時に，選択した EJB に対して直ちに実行できる完全なスクリプトを自動的に生成することができます。

付録 C

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の呼び出しは，次の手順で行います。

1 **lr_load_dll** 関数を使用して，スクリプトの先頭で DLL を読み込みます。このステートメントを **vuser_init** セクションの先頭に置きます。**ci_load_dll** 関数が **lr_load_dll** 関数に置き換えられます。

次の構文を使用します。

lr_load_dll(**library_name**);

UNIX プラットフォームでは，DLL は共有ライブラリと呼ばれています。ライブラリの拡張子はプラットフォームによって異なります。

2 DLL 内で定義されている関数をスクリプト内の適切な場所で呼び出します。

次の例では，Test_1 テーブル作成後，**orac1.dll** で定義されている **insert_vals** 関数を呼び出します。

```
int LR_FUNC Actions(LR_PARAM p)
{
lr_load_dll("orac1.dll");
lrd_stmt(Csr1, "create table Test_1 (name char(15), id integer)¥n", -1,
        1 /*Deferred*/, 1 /*Dflt Ora Ver*/, 0);
lrd_exec(Csr1, 0, 0, 0, 0, 0);
/* insert_vals 関数を呼び出し，値をテーブルに挿入する。
*/*/insert_vals();
lrd_stmt(Csr1, "select * from Test_1¥n", -1, 1 /*Deferred*/, 1 /*Dflt Ora
Ver*/, 0);
lrd_bind_col(Csr1, 1, &NAME_D11, 0, 0);
lrd_bind_col(Csr1, 2, &ID_D12, 0, 0);
lrd_exec(Csr1, 0, 0, 0, 0, 0);
lrd_fetch(Csr1, -4, 15, 0, PrintRow14, 0);
. . .
```

注：DLL の完全パス名を指定できます。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の呼び出しは，次の手順で行います。

1（LoadRunner¥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。

たとえば，NT プラットフォーム上の WinSock 仮想ユーザの DLL をロードするには，**mdrv.dat** ファイルに次の文を追加します。

```
[WinSock]
ExtPriorityType=protocol
WINNT_EXT_LIBS=wsrun32.dll
WIN95_EXT_LIBS=wsrun32.dll
LINUX_EXT_LIBS=liblrs.so
SOLARIS_EXT_LIBS=liblrs.so
HPUX_EXT_LIBS=liblrs.sl
AIX_EXT_LIBS=liblrs.so
LibCfgFunc=winsock_exten_conf
UtilityExt=lrun_api
ExtMessageQueue=0
ExtCmdLineOverwrite=-WinInet No
ExtCmdLineConc=-UsingWinInet No
WINNT_DLLS=user_dll1.dll, user_dll2.dll, ...
```

2 DLL 内で定義されている関数をスクリプト内の適切な場所で呼び出します。

付録 D

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 LoadRunner 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仮想ユーザ・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，LoadRunner の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。

- 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とスクリプトへの挿入
- 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，LoadRunner 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
- LoadRunner 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 6 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

LoadRunner には，テンプレートを作業ディレクトリにコピーするユーティリティが含まれています。このユーティリティは **mkdbtest** と呼ばれているもので，$M_LROOT¥bin にあります。このユーティリティを実行するには，次のように入力します。

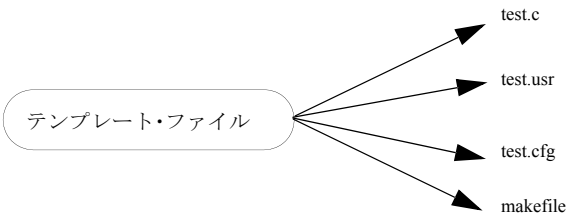
```
mkdbtest name
```

mkdbtest を実行すると，mkdbtest はテンプレート・ファイル **name.c** を格納するためのディレクトリ **name** を作成します。たとえば，次のように入力するとします。

```
mkdbtest test1
```

mkdbtest は，テンプレート・スクリプト **test1.c** を格納するためのディレクトリ **test1** を作成します。

mkdbtest ユーティリティを実行すると，4 つのファイル（**test.c**，**test.usr**，**test.cfg**，**Makefile**）を格納するためのディレクトリが作成されます。これらのファイルの「**test**」の部分には，mkdbtest で指定したテスト名が入ります。



仮想ユーザのアクションのプログラミングとスクリプトへの挿入

仮想ユーザ・スクリプトの **test.c**，**test.usr**，**test.cfg** ファイルは，使用する仮想ユーザに応じてカスタマイズできます。

実際の仮想ユーザのアクションをプログラミングして，**test.c** ファイルに挿入します。このファイルには，プログラミングされる仮想ユーザ・スクリプトに必要な構造になっています。仮想ユーザ・スクリプトには，vuser_init，Actions，vuser_end の 3 つのセクションが含まれます。

C++ のユーザに対しては，テンプレートは **extern C** を定義します。エクスポートされる関数が不用意に変更されることがないように，すべての C++ ユーザに対して この定義を行う必要があります。

```
#include “lrun.h”
#if defined(__cplusplus) || defined(cplusplus) extern “C”
{
#endif
int LR_FUNC vuser_init(LR_PARAM p)
{
                lr_message(“vuser_init done¥n”);
                return 0;
}
 int Actions(LR_PARAM p)
{
                lr_message(“Actions done¥n”);
                return 0;
}
 int vuser_end(LR_PARAM p)
{
                lr_message(“vuser_end done¥n”);
                return 0;
}
#if defined(__cplusplus) || defined(cplusplus)}
#endif
```

仮想ユーザのアクションをプログラミングして空のスクリプトに直接挿入します。場所は，各セクションの **lr_message** 関数の前です。

vuser_init セクションは，初期化中に最初に実行されます。このセクションには，接続情報とログオンの処理を挿入します。**vuser_init** セクションは，スクリプトの実行時に一度だけ実行されます。

Actions セクションは初期化の後に実行されます。このセクションには，仮想ユーザによって実行される実際の操作を挿入します。Actions セクションを繰り返すように仮想ユーザを設定できます（**test**.**cfg** ファイルを使います）。

vuser_end セクションは，最後，つまり仮想ユーザのすべてのアクションの実行後に実行されます。このセクションには，クリーンアップとログオフの処理を挿入します。**vuser_end** セクションは，スクリプトの実行時に一度だけ実行されます。

注：LoadRunner は，UNIX のシグナル，SIGHUP，SIGUSR1，SIGUSR2 を送信することによって仮想ユーザを制御します。仮想ユーザ・プログラムでは，これらのシグナルを使用しないでください。

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。(127 ページ「実行環境の設定」) **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
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
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，ほかのコンパイラ・フラグやライブラリを指定できます。

汎用のテンプレートを使った作業の場合には，アプリケーションのライブラリとヘッダー・ファイルをインクルードする必要があります。たとえば，アプリケーションで **testlib** というライブラリを使用している場合は，そのライブラリを LIBS セクションに指定します。

```
LIBS        = ¥
      -testlib ¥
      -lLrun50 ¥
      -lm
```

Makefile の変更後，作業ディレクトリのコマンド・ラインで「Make」と入力して，仮想ユーザ・スクリプト用のダイナミック・ライブラリ・ファイルを作成します。

これで LoadRunner コントローラからスクリプトを実行できるようになります。仮想ユーザ・スクリプトは，スクリプト・ディレクトリの **script.usr** ファイルです。仮想ユーザ・スクリプトをシナリオに組み込む方法の詳細については，『**LoadRunner コントローラ・ユーザーズ・ガイド**』を参照してください。

スクリプトをシナリオに組み込む前に，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。

付録 E

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

す

と

な

C-Interpreter Copyright Agreement

The Virtual User Generator generates standard C code which can be compiled with any ANSI C compiler. However, for the convenience of our customers we have provided a C Interpreter for running the generated code, without charge. The cci executable which is the front end of the interpreter is based on the freely available "lcc Retargetable C Compiler" by Christopher Fraser and David Hanson, and is covered by the lcc Copyright included below. Any bugs in cci should be reported to Mercury Interactive. The author's copyright notice is below.

MERCURY
INTERACTIVE

マーキュリー・インタラクティブ・ジャパン株式会社
〒 105-0003
東京都港区西新橋 2-38-5 西新橋 MF ビル 7 階

電話 : (03) 5402-9300
ファックス : (03) 5425-2288

Web: http://www.mercury.co.jp
カスタマー・サポート : http://www.mercury.co.jp/support

LRDBUG7.8JP/01