Mercury 仮想ユーザ・ジェネレータ

ユーザーズ・ガイド

第 2 巻 - プロトコル

Version 8.1

MERCURY™

Mercury 仮想ユーザ・ジェネレータ・ユーザーズ・ガイド 第 2 巻 － プロトコル，Version 8.1

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

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

LRDBUG8.1-02JP/01

目次

第 9 部 : ERP/CRM プロトコル

第 1 部

プロトコルについて

第 1 章

プロトコルについて

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

本章では，次の項目について説明します。

- 本書の使用
- 仮想ユーザのタイプ

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

本書の使用

本書『**VuGen プロトコル**』は，『Mercury 仮想ユーザ・ジェネレータ・ユーザーズ・ガイド』の第 2 巻です。『**第 1 巻 - 仮想ユーザ・ジェネレータの使い方**』では，VuGen の使用方法とテストの作成方法について説明します。第 2 巻では，個々のプロトコルに応じた設定とガイドラインを説明します。例えば，第 2 巻には，プロトコル固有の記録オプションや実行環境の設定について説明し，『**第 1 巻 - 仮想ユーザ・ジェネレータの使い方**』ではすべてのあるいはほとんどのプロトコルに共通の設定について示しています。

記録する仮想ユーザ・タイプを決めるとき，アプリケーションで複数のプロトコル（例えば，Web と FTP，または Web と NCA など）を使用していることをに気づくかもしれません。VuGen は複数のプロトコルを使用するスクリプトの記録をサポートしています。また，2 つのプロトコルを自動的に記録するデュアル・タイプもいくつか用意されています（SAPGUI/SAP-Web や Web/WinSock

など)。詳細については,『**第 1 巻 - 仮想ユーザ・ジェネレータの使い方**』を参照してください。

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

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。

仮想ユーザのタイプ

VuGen は，システムのエミュレートを可能にするさまざまな仮想ユーザ・テクノロジを備えています。それぞれのテクノロジは特定のアーキテクチャに対応しており，アーキテクチャごとに専用の仮想ユーザ・スクリプトが作成されます。例えば，Web 仮想ユーザ・スクリプトを使用して，ユーザの Web ブラウザでの操作をエミュレートしたり，FTP 仮想ユーザを使用して，FTP セッションをエミュレートしたりします。さまざまな仮想ユーザ・テクノロジを単独で使用したり，組み合わせて使用したりすることによって，効果的な負荷テスト，Tuning Console セッション，または Business Process Monitor プロファイルを作成できます。

仮想ユーザのタイプは次のカテゴリに分類されます。

- ［**すべてのプロトコル**］: サポートされている全プロトコルのアルファベット順リスト。
- ［**アプリケーションの導入ソルーション**］: Citrix プロトコル用。
- ［**クライアント / サーバ**］: MS SQL，ODBC，Oracle 2-Tier，DB2 CLI，Sybase Ctlib，Sybase Dblib，Windows Sockets，DNS の各プロトコル用。
- ［**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，Microsoft .NET，Web （HTTP/HTML），Web サービス，Web/WinSocket プロトコル用。
- ［**Enterprise Java Beans**］: EJB テストおよび RMI-Java プロトコル用。

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

第 2 部

Java 言語プロトコルを使った作業

「Java 言語プロトコルを使った作業」では，RMI-Java，CORBA-Java，EJB および Jacada タイプについて説明します。各プロトコルの詳細については，該当する項を参照してください。この部には，Java 仮想ユーザのすべてのタイプを対象とする情報が含まれています。

第 2 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，以下の項目について説明します。

- Java 言語仮想ユーザ・スクリプトの記録について
- 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 を仮想ユーザ API Java 固有の関数で拡張したスクリプトが生成されます。記録後，JDK ライブラリまたはユーザ定義クラスを使った標準 Java コードで，そのスクリプトの拡張や変更ができます。

スクリプトの準備ができたら，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，Tuning Module のセッション・ステップまたはビジネス・プロセス・モニタ・プロファイルに組み込みます。

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 14 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，VuGen を使わずにそれらをロードする場合よりも，多少時間がかかります。

VuGen には，Web 用に作成された仮想ユーザ・スクリプトを Java に変換するツールがあります。詳細については，283 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 14 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 3 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。**

仮想ユーザ API 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 14 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，14 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 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「VuGen パラメータを使った作業」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 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「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 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「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，関数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 Mercury 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，199 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。

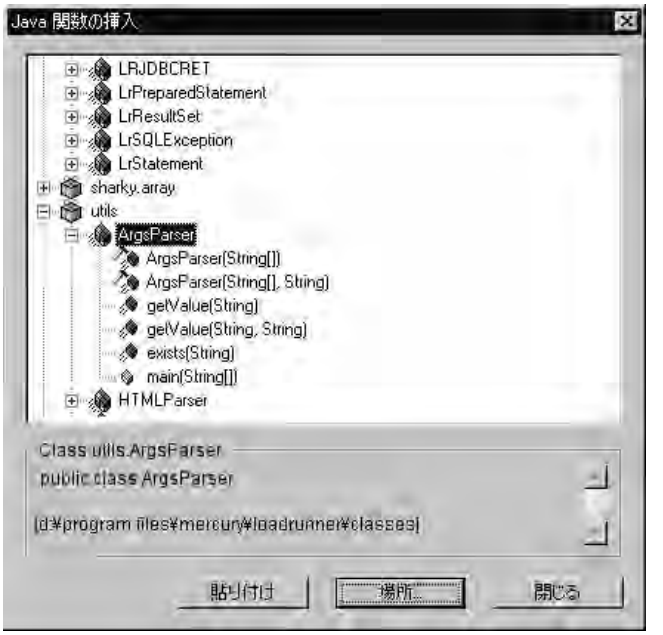
保護されているメソッドを仮想ユーザで使用するには，必要なメソッドのパッケージにその仮想ユーザを追加します。スクリプトの先頭に次の行を追加します。

```
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，16 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 ()

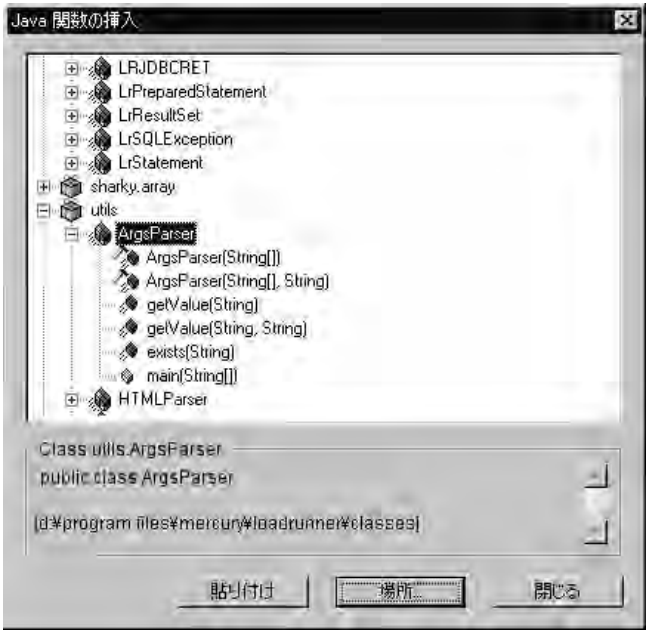
Java メソッドの手作業による挿入

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，18 ページ「スクリプト生成の設定」を参照してください。

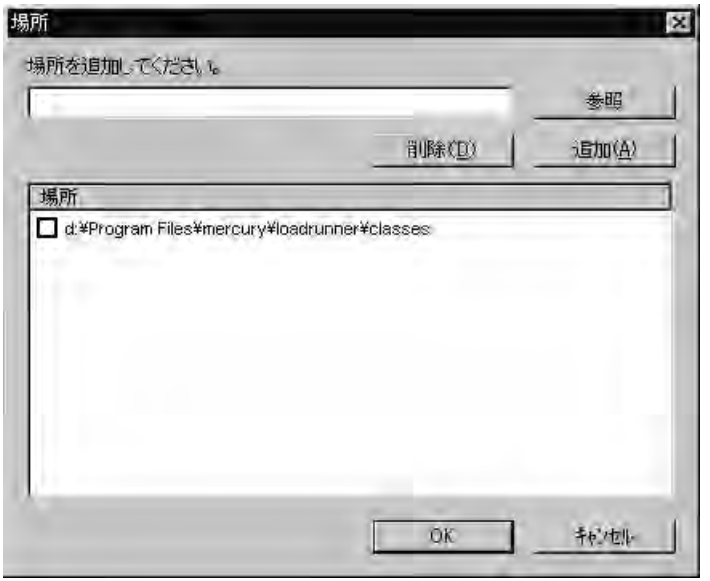
Java 関数を挿入するには，次の手順を実行します。

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

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



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



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

5［**追加**］をクリックして，リストに項目を追加します。

6 追加するパスまたはアーカイブごとに手順 4 と 5 を繰り返します。

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

8［**OK**］をクリックして［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。例えば，スクリプトで「(String)=LavaVersion.getVersionId();」というステートメントが生成された場合，**(String)** を文字列型変数に置き換えます。

13 import ステートメントや仮想ユーザ API Java 関数（第 14 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，VuGen の dat ディレクトリにある **jquery.ini** ファイルを開きます。

```
[Display]
FullClassName=False

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

クラス名のパス

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

FullClassName が有効	FullClassName が無効
	

自動トランザクション

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

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 3 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 記録オプションの設定について
- 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
- CORBA 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 39 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。

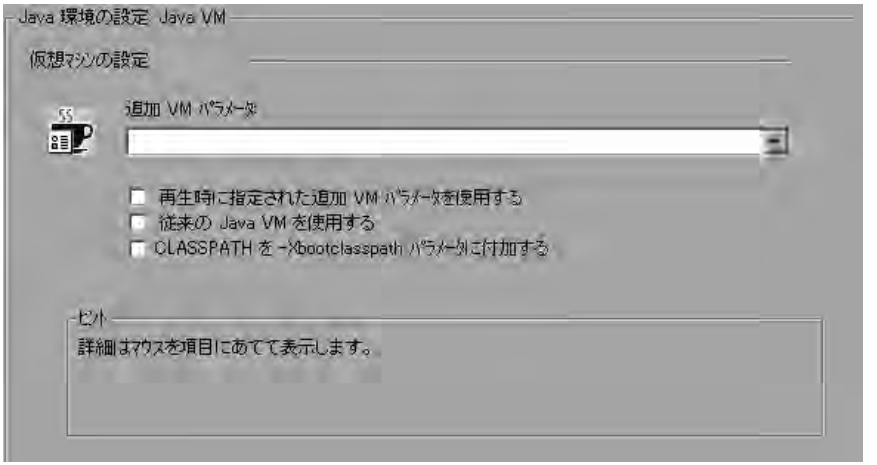
次の項目について記録オプションの設定が可能です。

- 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，次の手順を実行します。

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

Java 環境設定：**Classpath** ノードでは，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，次の手順を実行します。

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，次の手順を実行します。

［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（標準設定では有効）。

［**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，テストまたはチューニング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

［**バイト形式オプション**］：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，次の手順を実行します。

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，44 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。

［**シリアル化オブジェクトを無視する**］：記録されたスクリプトでは折りたたむことのできないシリアライズされたオブジェクトを一覧表示します。オブジェクトはカンマで区切ります。

［**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）。

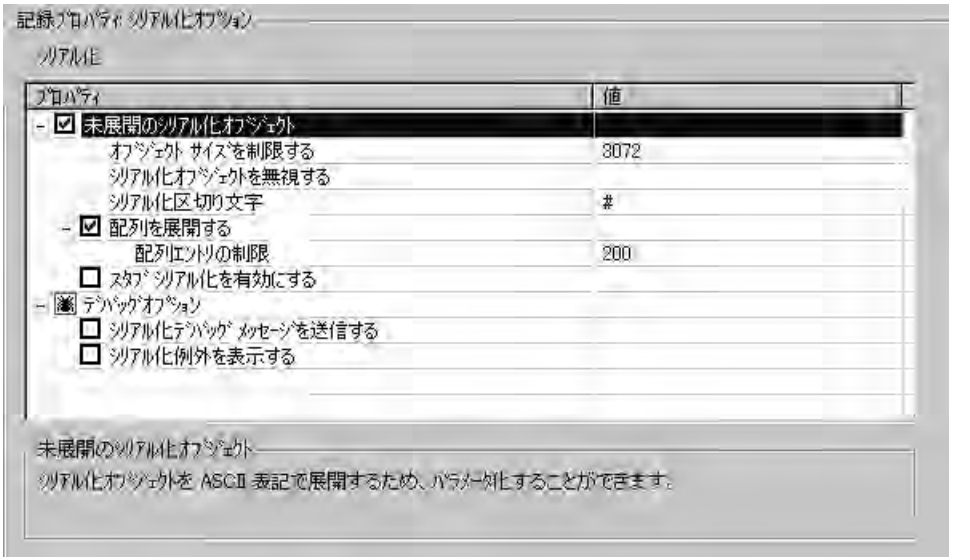
［デバッグ オプション］

［**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，次の手順を実行します。

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。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。

3 ［**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 4 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）。

［**文字列配列を相関する**］：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）。

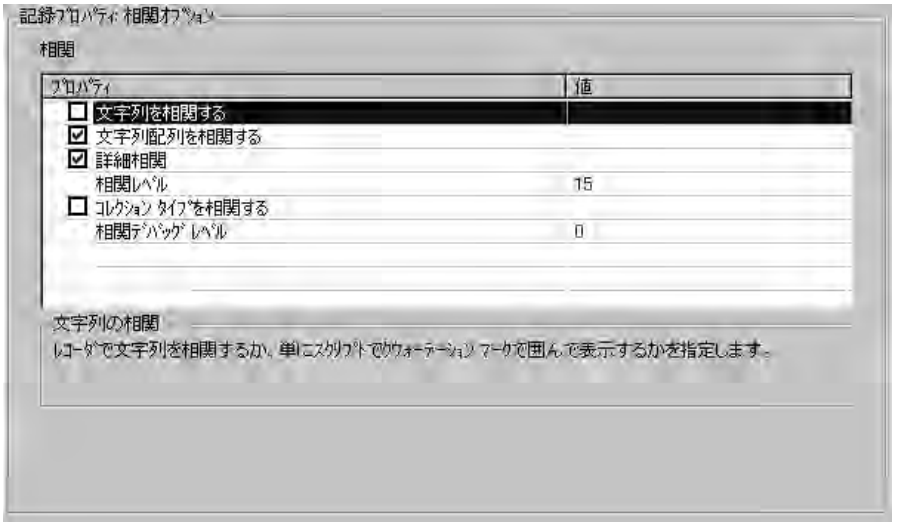
［**相関レベル**］: 相関のレベルの深さを，スキャン対象の内部コンテナの数として指定します（標準設定は 15）。

［**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，次の手順を実行します。

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。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。

3 ［**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**］オプションを使用します。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**］: Mercury 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，次の手順を実行します。

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。

［**DII 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。

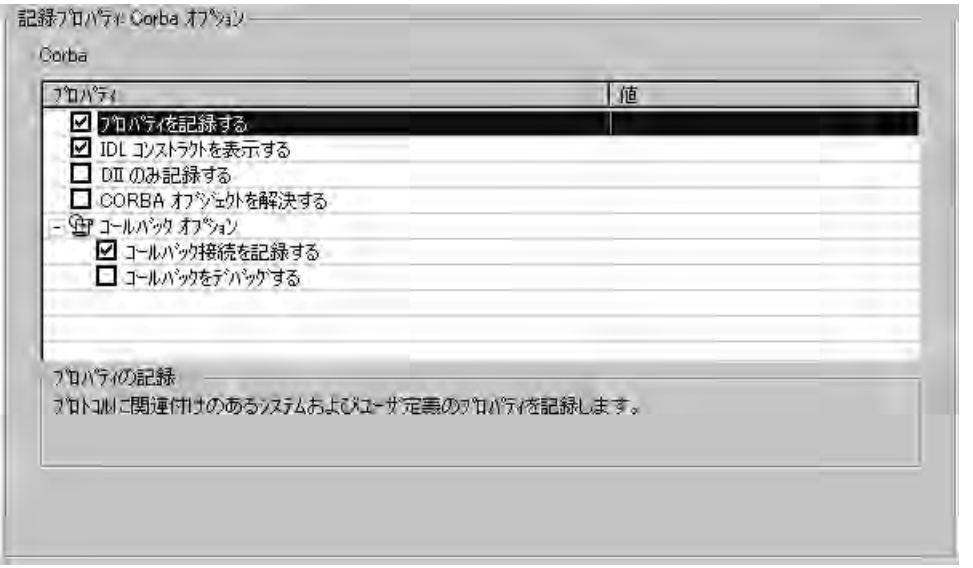
コールバック・オプション

［**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，次の手順を実行します。

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 4 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，以下の項目について説明します。

- Java スクリプトの相関について
- 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，44 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。

［**文字列を相関する**］：記録中にスクリプト内の文字列を相関させます。このオプションを無効にすると，実際に記録された値はスクリプトに引用符付きで含まれます。このオプションが無効になると，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）。

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

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

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

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

シリアル化メカニズムの使用

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

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

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

```
public class Actions {

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

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

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

```
public class Actions {

 // Public 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 3 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・ファイルのインデックス番号です。

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

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

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

```
public class Actions {

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

文字列値は，区切り文字の間に置かれます。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。例えば，**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。例えば，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 3 章「Java 記録オプションの設定」を参照してください。

第 5 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，以下の項目について説明します。

- Java 実行環境の設定について
- 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**］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 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「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，次の手順を実行します。

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。

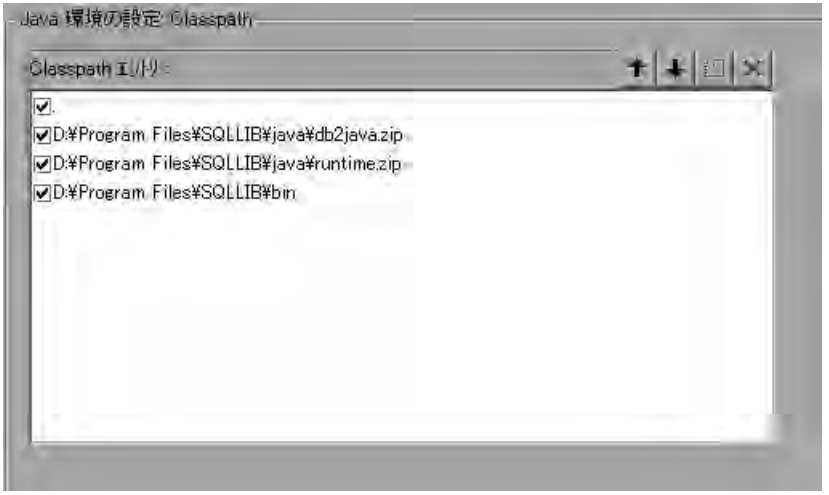
3 **-Xbootclasspath/p** オプションを使って再生するには，［**-Xbootclasspath VM パラメータを使用する**］オプションを選択します。

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，次の手順を実行します。

1 ［実行環境設定］（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します。

［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 3 部

アプリケーション配備ソリューション・プロトコル

第 6 章

Citrix 仮想ユーザ・スクリプトの作成

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。

オプションの **Mercury Citrix Agent** は，組み込み同期化を提供する直感的なスクリプトの作成を支援します。詳細については，第 7 章「LoadRunner Citrix エージェントの使用方法」を参照してください。スクリプトの作成の役立つヒントについては，87 ページ「Citrix 仮想ユーザ・スクリプトの再生とトラブルシューティングのヒント」を参照してください。

本章では，次の項目について説明します。

- Citrix 仮想ユーザ・スクリプトを使った作業の開始
- クライアントとサーバのセットアップ
- 記録に関するヒント
- Citrix 記録オプションについて
- Citrix 記録オプションの設定
- Citrix の表示設定
- Citrix 実行環境の設定
- Citrix 仮想ユーザ・スクリプトの表示と変更
- 再生の同期化
- ICA ファイルについて
- Citrix 関数の使用方法
- Citrix 仮想ユーザ・スクリプトの再生とトラブルシューティングのヒント
- ロード・ジェネレータ・マシンごとの仮想ユーザ数の増加

Citrix 仮想ユーザ・スクリプトの作成について

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。各関数の先頭には，**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 では Citrix Nfuse セッションの記録も可能です。Citrix NFuse を使用する場合，クライアントはインストールされますが，インタフェースはクライアントのインタフェースではなくブラウザとなります。Nfuse セッションを記録するには，Citrix と Web の仮想ユーザ用のマルチ・プロトコル・スクリプトの記録を実行する必要があります（『**第 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「VuGen を使った記録」を参照）。マルチ・プロトコル・モードでは，VuGen は記録中に Citrix と Web プロトコルの両方から関数を生成します。

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

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

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。VuGen では，セッション中に記録された API 呼び出しが表示されるので，アクションを追うことができます。

Citrix 仮想ユーザ・スクリプトを使った作業の開始

本項では，VuGen を使って Citrix ICA 仮想ユーザ・スクリプトの作成プロセスの概略を説明します。このほか，87 ページ「Citrix 仮想ユーザ・スクリプトの再生とトラブルシューティングのヒント」も参照してください。

Citrix ICA スクリプトを作成するには，次の手順を実行します。

1 **クライアントとサーバの設定が正しいことを確認します。**

これらの設定の詳細については，60 ページ「クライアントとサーバのセットアップ」を参照してください。

2 **VuGen を使ってアクションを記録します。**

VuGen を起動して，新しい仮想ユーザ・スクリプトを作成します。必ず 60 ページ「クライアントとサーバのセットアップ」に従ってください。

スクリプトの表示の詳細については，『**第 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「VuGen を使った記録」を参照してください。

3 **仮想ユーザ・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 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「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）。**

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 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「VuGen パラメータを使った作業」を参照してください。

5 **Citrix の表示オプションを設定します。**

Citrix 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，70 ページ「Citrix の表示設定」を参照してください。

6 **実行環境を設定します。**

実行環境を設定することによって，スクリプト実行中の仮想ユーザの動作を制御します。この設定には，ペースの設定，ログ，思考遅延時間，接続情報が含まれます。

Citrix 固有の実行環境の設定方法の詳細については，71 ページ「Citrix 実行環境の設定」を参照してください。一般的な実行環境の設定を行うには，『**第 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「実行環境の設定」を参照してください。

7 **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。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および設定情報が含まれます。VuGen は，これらのファイルをスクリプトと一緒に保存します。

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，87 ページ「Citrix 仮想ユーザ・スクリプトの再生とトラブルシューティングのヒント」および『**第 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「スタンドアロン・モードでの仮想ユーザ・スクリプトの実行」を参照してください。

スクリプトを作成した後，スクリプトを自分の環境（LoadRunner シナリオ，Performance Center 負荷テスト，Tuning Module セッション，またはビジネス・プロセス・モニタ・プロファイル）に統合します。詳細については，『**Mercury LoadRunner コントローラ・ユーザーズ・ガイド**』，**Tuning Console**，**Performance Center**，または **Application Management** のマニュアルを参照してください。

クライアントとサーバのセットアップ

スクリプトを作成する前に，サポートされている Cirix クライアントがマシンにインストールされており，サーバが正しく設定されていることを確認します。本項では，次の項目について説明します。

- クライアントのバージョン
- サーバの設定

クライアントのバージョン

スクリプトを実行するには，各ロード・ジェネレータ • マシンに Citrix クライアントがインストールされている必要があります。クライアントがインストールされていない場合は，Citrix の Web サイト（www.citrix.com）の **download** セクションからダウンロードできます。

VuGen は，バージョン 8.00，バージョン 6.30.1060 以前，および Citirx Web クライアントを除くすべての Citrix クライアントをサポートします。

サーバの設定

VuGen で記録を行うには，次の項目について Citrix サーバを設定する必要があります。

MetaFrame : MetaFrame サーバ（1.8 または 3）がインストールされていることを確認します。サーバのバージョンを調べるには，サーバのコンソール・ツールバーにある［**Citrix コネクション構成ツール**］を選択して，［**Help**］＞［**About**］を選択します。

［**Configure Server to Close Sessions**］: Citrix サーバがセッションを完全に終了するように設定します。Citrix クライアントが接続を終了するとき，標準設定では，次回そのクライアントが新しい接続を開いたときのためにセッションを保存するようにサーバが設定されています。したがって，同じクライアントで新たに接続すると，前回切断したときと同じ作業領域が表示されます。新しいテスト実行のたびに初期状態の作業領域を使用できるようにすることが望まれます。

テストのたびに初期状態の作業領域を確保するには，以前のセッションを保存しないよう Citrix サーバを設定する必要があります。その代わりに，クライアントがタイムアウトまたは接続断になるたびにクライアントから切断することによって，接続をリセットするようにします。

- MetaFrame 1.8 Server
- MetaFrame 3 Server

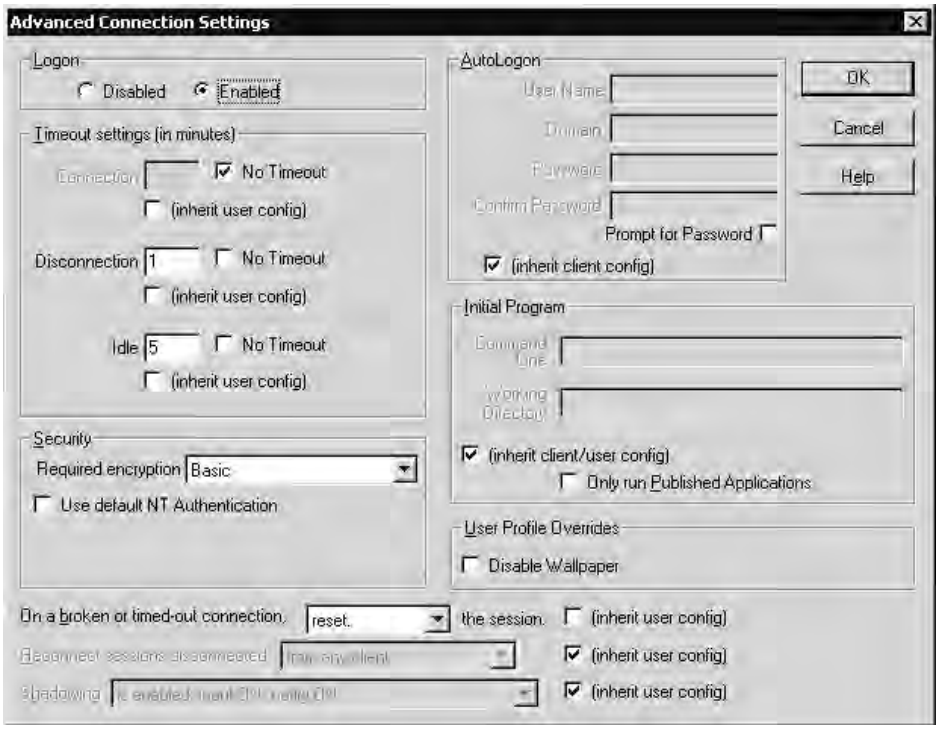
MetaFrame 1.8 Server

MetaFrame サーバの各セッションの接続をリセットするには，次の手順を実行します。

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**］＞［**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。［Advanced Connection Settings］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［**On a broken or timed-out connection**］リスト・ボックス横の［**inherit user config**］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。

5［**OK**］をクリックします。

MetaFrame 3 Server

MetaFrame 3 サーバの各セッションの接続をリセットするには，次の手順を実行します。

1［Citrix Connection Configuration］ダイアログ・ボックスを開きます。［プログラム］＞［Citrix］＞［Administration Tools］＞［Citrix Connection Configuration Tool］を選択します。

2 ica-tcp 接続名を選択し，［**Connection**］＞［**Edit**］を選択します。あるいは，接続をダブルクリックします。［コネクションの編集］ダイアログ・ボックスが表示されます。

3［**Advanced**］ボタンをクリックします。［Advanced Connection Settings］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［**On a broken or timed-out connection**］リスト・ボックス横の［**inherit user config**］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。

5［**OK**］をクリックします。

記録に関するヒント

スクリプトを記録するときは，効果的なスクリプトを作成できるように必ず次のガイドラインに従ってください。

シングル・プロトコル・スクリプトとマルチ・プロトコル・スクリプト

スクリプトを新規に作成するときは，シングル・プロトコルかマルチ・プロトコルのスクリプトを作成できます。簡単な Citrix ICA セッションを記録する場合は，シングル・プロトコル・スクリプトを使用します。ただし，Nfuse セッションの記録時には，Citrix ICA と Web（HTML/HTTP）を対象とするマルチ・プロトコル・スクリプトを作成する必要があります。これによって，両方のプロトコルを記録できるようになります。詳細については，『**第 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「VuGen を使った記録」を参照してください。

適切なセクションに記録する

接続処理は「**vuser_init**」セクションに，終了処理は「**vuser_end**」セクションに記録します。これにより，接続もしくは切断時に反復が実行されないようになります。セクションへの記録の詳細については，58 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。

初期状態のセッションを実行する

セッションを記録するときは必ず，接続操作から始まって，クリーンアップで終わる完全なビジネス・プロセスを実行するようにします。ビジネス・プロセス全体を始めから再実行できるところでセッションを終了するようにします。クライアントやアプリケーションのウィンドウを開いたままにしないようにします。

明示的にクリックする

サブメニューを開くときは，メニューの自動展開に頼らずに，各オプションを明示的にクリックします。例えば，［**スタート**］＞［**プログラム**］＞［**Microsoft Word**］の場合は，「**プログラム**」という語をクリックします。

ウィンドウのサイズを変更しない

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。

解像度の設定に矛盾がないことを確かめる

ビットマップの同期化を確実に成功させるために，解像度の設定が一致していることを確認します。記録用マシンの ICA クライアントの設定，記録オプション，および実行環境の設定を確認します。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。

1024 × 768 の解像度を使用する

サポートされている解像度（ウィンドウのサイズ）は，640 x 480，800 x 600，および 1024 x 768 です。記録マシンでは，1024 x 768 の設定が推奨されます。標準のサイズが 800 x 600 の Citrix ウィンドウが正しく表示されるためです。VuGen ではデスクトップのカラー設定が使用されます。

手動の同期ポイントを追加する

記録中に，イベント（例えば 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，77 ページ「再生の同期化」を参照してください。

クライアントの更新を無効にする

Citrix クライアントの更新を有効にするかどうか尋ねられた場合には，クライアントの更新を無効にします。これにより，VuGen とまだテストされていない最新の Citrix クライアント間の前方互換性の問題が回避されます。

ウィンドウの形式

すべてのウィンドウを XP 形式ではなく「クラシック」のウィンドウ形式で記録します。これは **sync_on_bitmap** 関数に関連します。

ウィンドウの形式を「クラシック」に変更するには，次の手順を実行します。

1 デスクトップをクリックします。

2 右クリック・メニューから［**プロパティ**］を選択します。

3［テーマ］タブを選択します。

4［テーマ］ドロップ・ダウン・リストから，［**Windows クラシック**］を選択します。

5［**OK**］をクリックします。

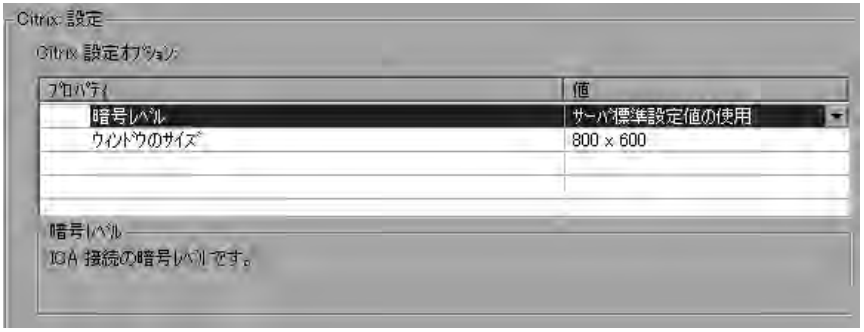
Citrix 記録オプションについて

次の項目について Citrix 記録オプションの設定が可能です。

- 設定
- レコーダ
- ログイン（シングル・プロトコルの Citrix ICA スクリプトの場合のみ）

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**」，「**1280 x 1024**」，「**1600 x 1200**」から選択できます。

レコーダ

［**Citrix レコーダ オプション**］で，記録中にタイトルの変更されたウィンドウ名を生成する方法を指定できます。画面のスナップショットをスクリプト・ファイルと一緒に保存するかどうか，また，テキスト同期化関数を生成するかどうかを指定することもできます。

ウィンドウ名

Citrix セッションには，記録中にアクティブなウィンドウの名前が変わるものがあります。スクリプトをそのまま再生しようとすると，仮想ユーザは最初のウィンドウ名を使用するため再生が失敗します。記録オプションを使用して，ウィンドウの命名規則（VuGen がウィンドウを特定するのに使用する共通のプレフィックスまたはサフィックス）を指定できます。

例えば，最初のウィンドウ名が「無題 － メモ帳」であり，アプリケーションの実行中にこのウィンドウの名前が「my_test － メモ帳」に変わる場合は，VuGen に共通のサフィックス（「メモ帳」）のみを使用するよう指示できます。

記録中のウィンドウ名の生成には次のオプションを使用できます。

［**新規のウィンドウ名をそのまま使用する**］：ウィンドウのタイトルをそのままウィンドウ名として使用する場合に選択します（標準設定）。

［**新規名に共通のプレフィックスを使用する**］：ウィンドウ・タイトルの先頭の共通文字列をウィンドウ名として使用する場合に選択します。

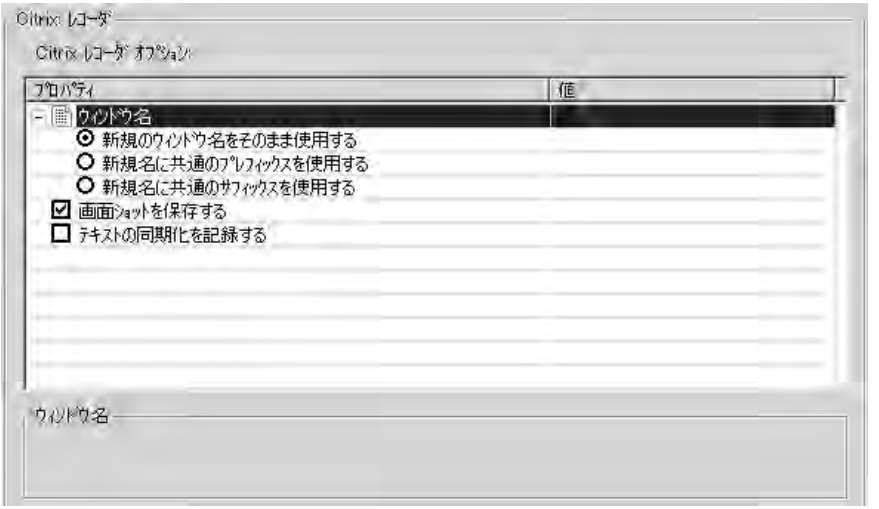
［**新規名に共通のサフィックスを使用する**］：ウィンドウ・タイトルの末尾の共通文字列をウィンドウ名として使用する場合に選択します。

注：あるいは，記録後に実際のスクリプトでウィンドウ名を変更することもできます。スクリプト・ビューでウィンドウ名を見つけ，ウィンドウ名の末尾をワイルドカードの「＊」で置き換えます。

例：ctrx_sync_on_window ("My Application*", ACTIVATE, …);

［**画面ショットを保存する**］：該当する場合，スクリプトのステップごとに，Citrix クライアント ウィンドウのスナップショットが VuGen によって保存されます。記録したアクションをより的確に理解するために，このオプションを有効にすることをお勧めします。ただし，スナップショットを保存するとより多くのディスク領域が使用され，記録セッションの速度が低下します。

［**テキストの同期化を記録する**］: マウスをクリックする前に VuGen によってテキストの同期化が記録されます（この設定は，標準設定です）。



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（NFUSE を使用する場合は，ログインが Web ページで行われるためログイン・オプションは使用できません）。

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。

サーバの名前または VuGen が **ctrx_connect_server** 関数を生成する接続を指定する必要があります。

```
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。

接続パラメータの定義

［**接続**］セクションに，次の接続情報を入力します。

- Citrix サーバへログインするための**ユーザ名**。
- 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**。
- MetaFrame サーバがクライアントの識別に使用する**クライアント名**（任意）。

ログイン情報セクション：ログイン情報を入力します。

- 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。ブラウザを使用する場合は，TCP/IP+HTTP オプションを選択します。その他のアプリケーションの場合はすべて，TCP/IP を選択します。

➤ Citrix **サーバ**の名前。新しいサーバをリストに追加するには，［**追加**］をクリックし，サーバ名（およびサーバの TCP/IP + HTTP 用ポート）を入力します。複数のサーバが適用されるのは［公開アプリケーション］を指定したときのみです。特定のアプリケーションのないデスクトップに接続しようとしている場合は，1 つのサーバのみが表示されます。

➤ Citrix サーバで認識される，**発行されたアプリケーション**の名前。使用可能なアプリケーションのリストがドロップダウン・メニューに含まれています。公開アプリケーションを指定されていない場合，VuGen はサーバのデスクトップを使用します。

公開アプリケーションを指定しなければ，Citrix のロード・バランシングは機能しません。サーバのデスクトップにアクセスする場合にロード・バランシングを使用するには，デスクトップをサーバ • マシンで公開されたアプリケーションとして登録し，［**発行されたアプリケーション**］ドロップダウン・リストからこの名前を選択します。

［接続パラメータの ICA ファイルを使用する］

既存の .ica ファイルと関連するすべての設定情報がある場合は，［**接続パラメータの 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，82 ページ「ICA ファイルについて」を参照してください。

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。

Citrix 記録オプションを設定するには，次の手順を実行します。

1 ［記録オプション］ダイアログ・ボックスを開きます。［**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。

2 ［Citrix］の［**ログイン**］ノードを選択します（シングル・プロトコルの Citrix ICA スクリプトの場合のみ）。

➤ 既存の ica ファイルと関連するすべての設定情報がある場合は，［**接続パラメータの ICA ファイルを使用する**］を選択します。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のファイルの場所を見つけます。

➤ ica ファイルがない場合は，［**接続パラメータを定義する**］を選択します。これは標準設定です。［**接続**］情報と［**識別**］情報を入力します。

3 ［**Citrix：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：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 VuGen が各ステップのスナップショットを保存しないようにするには，［**スナップショットを保存する**］をクリアします。

6 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。

7 ［**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。

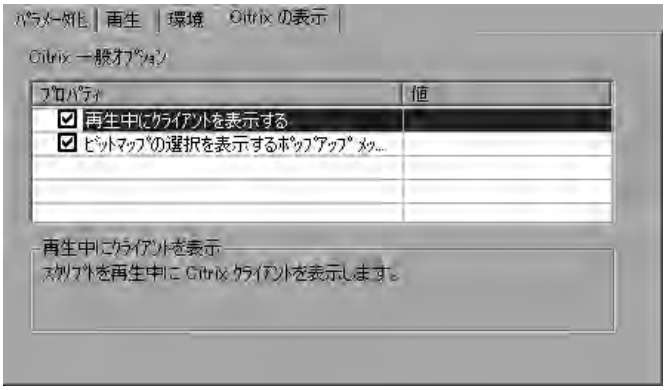
Citrix の表示設定

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。

Citrix 表示オプションを設定するには，次の手順を実行します。

1 ［一般オプション］ダイアログ・ボックスを開きます。VuGen メイン・ウィンドウで，［**ツール**］＞［**一般オプション**］を選択します。

2［**Citrix の表示**］タブを選択します。



3 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 スナップショットの中で対話形式で作業を開始するときにポップアップ・メッセージを発行するには，［**ビットマップの選択を表示するポップアップメッセージ**］を選択します。ビットマップまたはテキストを選択する前に，右クリック・メニュー・オプションで［**Insert Sync Bitmap**］または［**Insert Get Text**］を選択すると，このメッセージが表示されます。

5［**OK**］をクリックします。

Citrix 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［**アプリケーションセットの設定**］を選択します。［**デフォルトのオプション**］タブを選択します。

注 : Citrix 仮想ユーザでは IP スプーフィングはサポートされていません。

一般的な実行環境の設定を行うには,『**第 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「実行環境の設定」を参照してください。[速度のエミュレーション] プロパティの設定の詳細については,『**第 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「インターネット実行環境の設定」を参照してください。

次の項目について Citrix 固有の実行環境を設定できます。

- Citrix 設定実行環境の設定
- Citrix 時間設定実行環境の設定

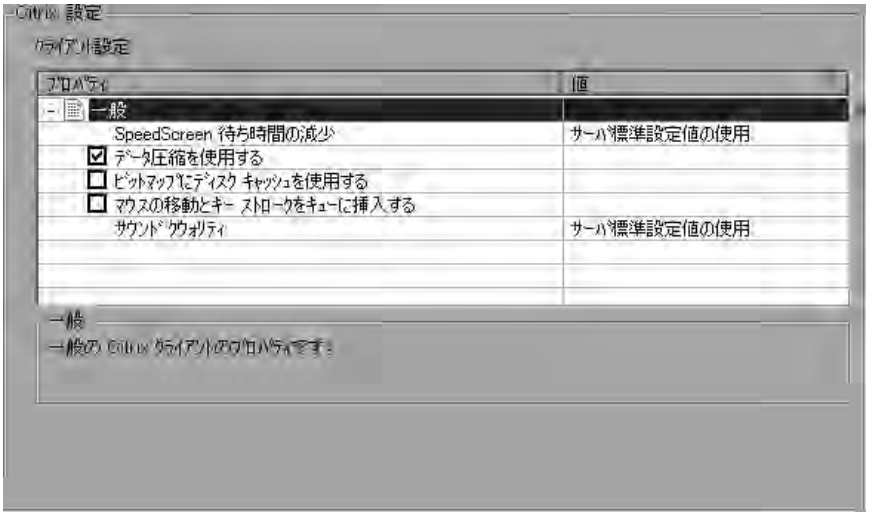
Citrix 設定実行環境の設定

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，次の手順を実行します。

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

2［**Citrix：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。これにより，低速接続の場合のネットワーク・トラフィックが減少します。このオプションを有効にすると，セッションにおけるキーボードとマウスの動きに対する応答が鈍くなります。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）。

6 リストから［**サウンド クウォリティ**］オプションの 1 つを選択します。［**サーバ標準設定値の使用**］，［**サウンド オフ**］，［**ハイ サウンド クウォリティ**］，［**ミディアム サウンド クウォリティ**］，および［**ロー サウンド クウォリティ**］があります。クライアント・マシンに 16 ビットの Sound Blaster 互換サウンド・カードが搭載されていない場合は，［**サウンド オフ**］を選択します。サウンドのサポートを有効にすると，クライアント・マシンの公開アプリケーションからサウンド・ファイルを再生できるようになります。

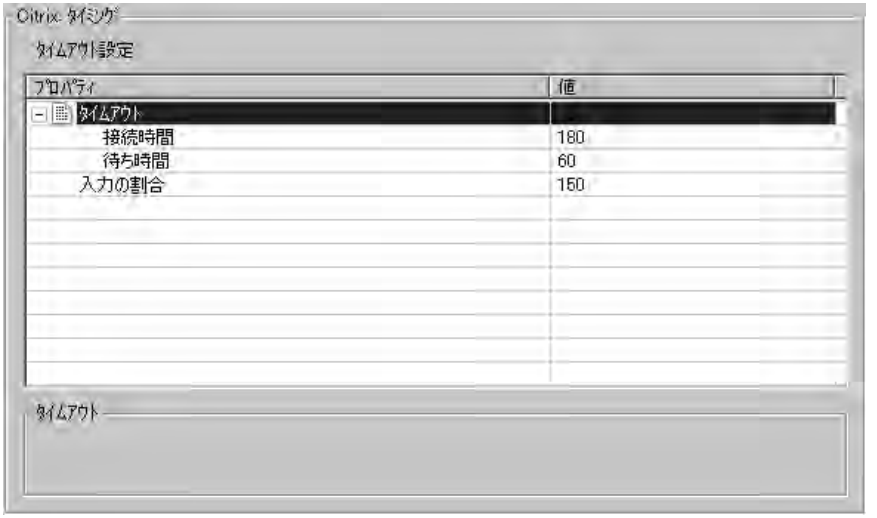
Citrix 時間設定実行環境の設定

タイムアウト設定は接続時間と待ち時間に関係します。

時間実行環境を設定するには，次の手順を実行します。

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

2［**Citrix：タイムアウト**］ノードを選択します。



3［**接続時間**］には，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を示します。標準設定は 180 秒です。

4［**待ち時間**］には，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を示します。標準設定は 60 秒です。

5［**入力の割合**］には，キーストローク間の遅延時間をミリ秒で指定します。

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。

Citrix 仮想ユーザ・スクリプトの表示と変更

仮想ユーザ・スクリプトの内容は，VuGen のスクリプト・ビューまたはツリー・ビューで表示できます。スクリプトの表示の詳細については，『**第 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「VuGen の紹介」を参照してください。

ツリー・ビューでは，Citrix 仮想ユーザのスナップショットを表示できます。次の Citrix ステップには，仮想ユーザに関連付けられたスナップショットがあります。

- **Obj Mouse Click**，**Obj Mouse Double Click**，および **Obj Mouse Down**
- **Sync on Window** および **Sync on Bitmap**

スナップショットでは，クライアント・ウィンドウが表示されるほか，アクションが実行されたオブジェクトも強調表示されます。

- **Mouse** ステップでは，ユーザがクリックした場所がピンク色の小さな四角形で示されます。
- **Sync on Bitmap** では，ビットマップ領域がピンク色のボックスで囲まれます。

Sync on Window では，ウィンドウ全体がピンク色のボックスで囲まれます。次の例では，スナップショットに **Sync on Window** ステップが示されていま

す。操作が実行されたウィンドウを正確に示すボックスによって，メモ帳の確認ボックスが囲まれています。

VuGen は，スナップショットをスクリプトの **data\snapshots** ディレクトリにビットマップ・ファイルとして保存します。スナップショット・ファイルの名前は関数の引数を調べることでわかります。

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

記録後，スクリプト・ビューかツリー・ビューのどちらかで，手作業でステップをスクリプトに追加できます。様々なスクリプト・ビューの詳細については，『**第 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「VuGen の紹介」を参照してください。

新しい関数を手作業で追加するだけでなく，Citrix 仮想ユーザのために，新しいステップをスナップショットから直接，対話形式で追加できます。右クリック・メニューを使用して，ビットマップ関連またはテキスト関連のステップを追加できます。エージェントのインストール時には，右クリック・メニューからいくつかの追加ステップも使用できます。詳細については，第 7 章「LoadRunner Citrix エージェントの使用方法」を参照してください。

関数を対話形式で挿入するには，次の手順を実行します。

1 ツリー・ビューの中のステップをクリックします。スナップショットが表示されていることを確認します。

2 スナップショットの中をクリックします。

3 コマンドの 1 つを右クリックして選択します。ダイアログ・ボックスが開き，ステップのプロパティが表示されます。

4 必要なプロパティを変更して［**OK**］をクリックします。VuGen によってステップがスクリプトに挿入されます。

再生の同期化

スクリプトの実行中，再生が正常に行われるようにするために，しばしばアクションを同期化する必要があります。同期化とは，スクリプトの中でイベントのタイミングを調整し，ウィンドウやオブジェクトが使用可能になるまで待つからアクションを実行することを指します。例えば，ボタンを押す前に特定のウィンドウがすでに開いているかどうかを知りたい場合があります。

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

自動同期化

記録中，仮想ユーザによるスクリプトの再生の同期化を支援する関数 **ctrx_sync_on_window** とインストールされているエージェントとの同期化を支援する関数 **ctrx_sync_on_text**（第 6 章「Citrix 仮想ユーザ・スクリプトの作成」参照）が VuGen によって自動的に生成されます。

Sync on Window

ctrx_sync_on_window 関数は，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 は通常 CREATE イベントを待機する関数を生成します。標準のウィンドウの場合，VuGen は ACTIVE イベントを待機する関数を生成します。

ctrx_sync_on_window 関数で記録されたすべてのウィンドウで，スナップショットをスクリプトのツリー・ビューから表示できます。

Sync on Text

ctrx_sync_on_text 関数は，指定された文字列が指定された場所に表示されるまで待機してから処理を続ける同期化関数です。この関数は指定した座標から半径 40 ピクセル内でテキストを検索します。

VuGen は LoadRunner Citrix エージェントがインストールされると，あらゆるマウスのクリックまたはダブルクリックの前に **ctrx_sync_on_text** を記録します。

次のコードの一部は，LoadRunner Citrix Agent がインストールされている Citrix の記録中に記録された **ctrx_sync_on_text** 関数を示しています。

```
ctrx_sync_on_window ("ICA Seamless Host Agent", ACTIVATE, 0,
0,391,224, "snapshot1", CTRX_LAST);

ctrx_sync_on_text (196, 198, "OK", TEXT, "ICA Seamless Host
Agent=snapshot2", CTRX_LAST);

ctrx_obj_mouse_click ("<class=Button text=OK>", 196, 198,
LEFT_BUTTON, 0, "ICA Seamless Host Agent=snapshot2",
CTRX_LAST);
```

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**」を参照してください（［**ヘルプ**］＞［**関数リファレンス**］）。

手作業による同期化

VuGen のユーザ・インターフェイスを通じて，またはカスタムの同期化関数を記録後に挿入することによって，記録中に手作業で同期化を追加できます。通常，この機能は，実際のウィンドウは変更されていないのに，ウィンドウ内部のオブジェクトが変更された場合に使用します。ウィンドウは変更されていないので，VuGen は **Sync on Window** ステップを検出も記録もしていません。例えば，特定のグラフィック・イメージがブラウザ・ウィンドウに出現するまで再生を待機する場合は，手作業で同期化を挿入します。また，複数のタブを持つ大きなウィンドウを記録している場合，新しいタブの内容が開くのを待機する同期化ステップを挿入できます。

記録中の同期化

記録中に同期化を追加するには，フローティング・ツールバーのマーカ・ツールを使用します。マーカ・ツールを使用すると，再生を再開する前にフォーカスを得る必要のあるクライアント・ウィンドウの領域をマークできます。

同期化対象ビットマップ領域をマークするには，次の手順を実行します。

1 [**マーカ**] ボタン をクリックします。

2 領域の該当部分の左上から右下に，マウスをドラッグします。ツリー・ビューの現在のステップの後に **Sync on Bitmap** ステップが生成されます。スクリプト・ビューでは，選択した座標を引数として **ctrx_sync_on_bitmap** 関数が生成されます。

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

再生中，仮想ユーザは指定された座標でビットマップを探し，それが使用可能になるまで待って，テストを再開します。

記録後の同期化

記録セッションの後に同期化を追加することもできます。別の種類の同期化関数を追加するには，同期化ステップをスクリプトに手作業で入力します。関数を挿入するには，[挿入] > [ステップの追加] を選択し，希望の関数を選択します。詳細については，の『**第 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「仮想ユーザ・スクリプトの拡張」を参照してください。

Sync on Bitmap

- **Sync on Obj Info**（エージェント・インストールのみ）
- **Sync on Text**（エージェント・インストールのみ）

記録中，**Sync on Bitmap** ステップに生成されたビットマップはスクリプトの **data\snapshots** ディレクトリに保存されます。同期化が失敗した場合は，VuGen によって新しいビットマップが生成されるので，同期化が失敗した原因

を調べることができます。ビットマップ名は sync_bitmap_ ＜ hash_value ＞ .bmp. の形式となります。このビットマップは，スクリプトの出力ディレクトリ格納されます。シナリオ，プロファイル，またはチューニング・セッションの場合には，出力ファイルの書き込み先に格納されます。

また，同期化に間接的に影響するその他のいくつかのステップを追加することもできます。

- **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。例えば，**ctrx_sync_on_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。
- **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。フロー制御ステートメントを追加することにより，この関数を使用して，常に開いているとはかぎらない警告ダイアログ・ボックスなどのウィンドウを調べることができます。次の例では，**ctrx_win_exist** によってブラウザが起動されたかどうかを調べます。2 番目の引数は，ブラウザ・ウィンドウが開くまでの待機時間を示します。指定した時間開かなかった場合は，ブラウザによって自身のアイコンがダブルクリックされます。

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

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

ビットマップ変更の待機

領域内に表示されるデータまたは画像がどのようなものかわからなくても，変更が行われることはわかっている場合があります。
ctrx_sync_on_bitmap_change 関数を使用すると，この状況をエミュレートできます。VuGen が変更を待つ領域（座標とサイズ）を指定します。ビットマップ領域の正確な座標を取得するには，すでに説明したようにマーカ・ツールを使って **Sync on Bitmap** ステップを記録し，関数名を手作業で変更して，5 番目の引数を削除するのが簡単です。

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 ,]
[初期ビットマップ値 ,] CTRX_LAST);

ctrx_sync_on_bitmap_change にオプション引数を追加できます。

- 初期待機時間の値――変更の有無の確認を開始する時間。
- タイムアウト――失敗する前に変更が発生するまで待機する最大の秒数。
- 初期ビットマップ値――ビットマップの初期ハッシュ値。仮想ユーザは，ハッシュ値が指定した初期ビットマップ値と異なる値になるまで待機します。

```
/* 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);
```

注：［**Sync on Bitmap**］を使用している場合は，コントローラ / コンソール，ロード・ジェネレータおよび画面の設定が同じであることを確認します。設定が同じでないと，VuGen が再生中に正しいビットマップを見つけることができない場合があります。クライアントの設定方法の詳細については，65 ページ「設定」を参照してください。

ICA ファイルについて

Citrix ICA クライアント・ファイルは，Citrix クライアントを通じてアクセスされるアプリケーションの設定情報が含まれているテキスト・ファイルです。これらのファイルには，.ica 拡張子が付き，次の形式に準拠している必要があります。

```
[WFClient]
Version=
TcpBrowserAddress=

[ApplicationServers]
AppName1=

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

Username=
Domain=
ClearPassword=
```

注 : ［記録オプション］を使って ICA ファイルをロードすると，VuGen によってファイルがスクリプトと一緒に保存されるので，ICA ファイルを各インジェクタ・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 関数の使用方法

Citrix 記録セッション中，VuGen によってクライアントとリモート・サーバ間のやり取りをエミュレートする関数が生成されます。生成された関数には，**ctrx** というプレフィックスが付きます。どの関数も記録セッションの後，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と末尾を含めて任意の場所に指定できます。

接続関数

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_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。

マウス関数

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_up	マウス・ボタンの解放をエミュレートします。

オブジェクト関数 （エージェントのみ）

ctrx_obj_get_info	オブジェクトに関するクラス情報を取得します。
ctrx_obj_mouse_click	指定したオブジェクトをマウスでクリックすることをエミュレートします。
ctrx_obj_mouse_double_click	指定したオブジェクトをマウスでダブルクリックすることをエミュレートします。
ctrx_obj_mouse_down	指定したオブジェクト上でマウス・ボタンを押すことをエミュレートします。
ctrx_obj_mouse_up	指定したオブジェクト上でマウス・ボタンを放すことをエミュレートします。
ctrx_sync_on_obj_info	指定したオブジェクトが作成されるかアクティブになるのを待機します。

同期化関数

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_obj_info（エージェント使用の場合のみ）	指定したオブジェクトが作成されるかアクティブになるのを待機します。
ctrx_sync_on_text（エージェント使用の場合のみ）	特定のテキストが特定の位置に表示されるのを待機します。
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_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。

キーボード関数

ctrx_key	英数字以外のキーの入力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_type	英数字キーの入力をエミュレートします。

情報取得関数

ctrx_button_get_info	ボタンに関するクラス情報を取得します。
ctrx_edit_get_info	編集フィールドに関するクラス情報を取得します。
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_text（エージェント使用の場合のみ）	境界の定められたテキストをバッファに格納します。
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_list_get_info（エージェント使用の場合のみ）	リストに関するクラス情報を取得します。
ctrx_list_select_item（エージェント使用の場合のみ）	リストから項目を選択します。
ctrx_menu_select_item（エージェント使用の場合のみ）	メニュー項目を選択します。
ctrx_obj_get_info（エージェント使用の場合のみ）	オブジェクトに関するクラス情報を取得します。

選択関数

ctrx_list_select_item	リストから項目を選択します。
ctrx_menu_select_item	メニュー項目を選択します。

一般関数

ctrx_save_bitmap	境界の定められたビットマップをバッファに格納します。
ctrx_set_exception	例外処理を指定します。

Citrix 仮想ユーザ・スクリプトの再生とトラブルシューティングのヒント

ここでは，Citrix 仮想ユーザの次の項目に関するガイドラインやヒントを示します。

- 再生に関するヒント
- デバッグに関するヒント

記録のヒントについては，63 ページ「記録に関するヒント」を参照してください。

再生に関するヒント

初期化クォータを設定する

接続中に複数の仮想ユーザによって過負荷になるのを防ぐため，仮想ユーザの初期化クォータを（サーバの性能に応じて）4 から 10 に設定するか，スケジューラを使用して仮想ユーザのランプアップによる初期化を実施します。

思考遅延時間を有効にする

最良の結果を得るには，実行環境の設定で思考遅延時間を無効にしないようにします。思考遅延時間は，特に安定するまでに時間を要する **ctrx_sync_on_window** 関数や **ctrx_sync_on_bitmap** 関数の前には重要です。

マシン間で一貫性を保つ

別のマシンでスクリプトを再生する場合には，記録マシンと再生マシンの間で，Citrix クライアントのウィンドウ・サイズ（解像度），ウィンドウの色設定，システム・フォント，その他の標準オプションの設定が同じであることを確認します。これらの設定はビットマップのハッシュ値に影響し，不一致があった場合は，再生が失敗する可能性があります。Citrix クライアントの設定を表示するには，Citrix プログラム・グループから項目を選択し，［**Application Set Settings**］を選択するか，右クリック・メニューから［**Custom Connection Settings**］を選択します。［Default Options］タブを選択します。

ロード・ジェネレータ・マシンごとの仮想ユーザ数の増加

Citrix 仮想ユーザを稼動するロード・ジェネレータ・マシンでは，マシンで使用可能なグラフィック・リソース（GDI - Graphics Device Interface）により，実行できる仮想ユーザの数が制限される場合があります。マシンごとに実行できる仮想ユーザの数を増やすには，マシンでターミナル・サーバ・セッションを開き，追加のインジェクタ・マシンとして動作させることができます。

GDI カウントはオペレーティング・システムによって異なります。LoadRunner を使用している負荷の重いマシンの実際の GDI（Graphics Device Interface）カウントは，約 7,500 です。Windows 2000 マシンで使用可能な GDI の最大数は，16,384 個です。

ターミナル・サーバ・セッションの作成方法の詳細については，『**Mercury LoadRunner コントローラ・ユーザーズ・ガイド**』を参照してください。

注：標準設定では，ターミナル・サーバでのセッションには，256 色のカラー・セットが使用されます。ターミナル・セッションを負荷テスト用に使用しようとしている場合は，必ず 256 色のカラー・セットを備えたマシンに記録してください。

デバッグに関するヒント

クライアントを 1 つだけインストールする

Citrix セッションに任意のアクションを記録するのに失敗した場合は，お使いのマシンに Citrix クライアントが 1 つしかインストールされていないことを確認してください。インストールされているクライアントが 1 つだけかどうかを調べるには，コントロールパネルから［プログラムの追加と削除］ダイアロ

グ・ボックスを開き，Citrix ICA クライアントのエントリが 1 つだけあることを確認します。

ブレークポイントを追加する

問題が生じているコードの行を知るには，VuGen でスクリプトにブレークポイントを追加します。

スクリプトを同期化する

再生が失敗した場合，スクリプトに同期化関数を挿入して，対象ウィンドウがフォーカスを得るまで，待機時間を延ばす必要があるかもしれません。**lr_think_time** 関数を使って遅延時間を手作業で追加することもできますが，77 ページ「再生の同期化」で説明している同期化関数を使用することをお勧めします。

拡張ログ

［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。

スナップショット・ビットマップ

エラーの発生時，VuGen はスクリプトの**出力**ディレクトリに画面のスナップショットを保存します。このビットマップを見て，エラーが起きた原因を確認できます。

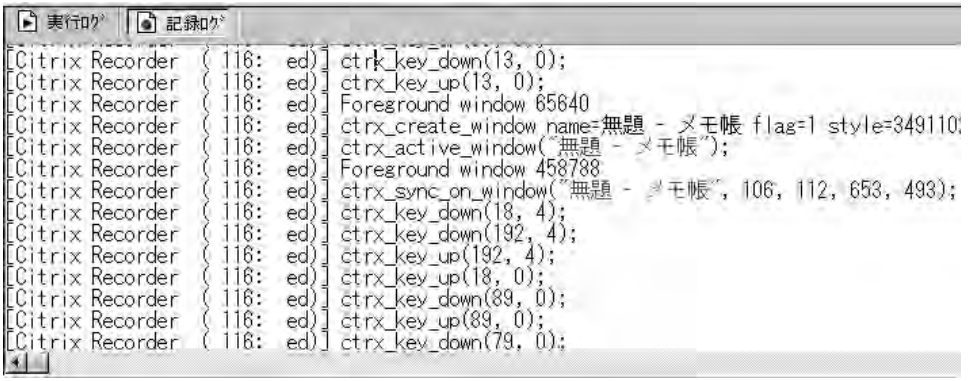
記録中，**ctrx_sync_on_bitmap** 関数に生成されたビットマップはスクリプトの **data** ディレクトリに保存されます。ビットマップ名は hash_value.bmp の形式となります。再生中に同期化に失敗した場合には，生成されたビットマップはスクリプトの出力ディレクトリに書き込まれます。あるいは，シナリオまたはチューニング・セッションで実行している場合には，出力ファイルが書き込まれる場所に書き込まれます。新しいビットマップを確認して，同期化が失敗した理由を調べることができます。

仮想ユーザの表示

シナリオまたはチューニング・セッションの実行中に仮想ユーザを表示するには，仮想ユーザ・コマンド行・ボックスに「-lr_citrix_vuser_view」と入力します。コントローラまたは Tuning Module Console 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ため，問題のある仮想ユーザの振る舞いを調べる場合にだけ行うようにします。

出力ウィンドウの表示

記録に関する詳細情報を調べるには，出力ウィンドウの記録ログを表示します。［**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。



第 7 章

LoadRunner Citrix エージェントの使用方法

LoadRunner Citrix エージェントは，Citrix サーバにインストールできる任意のユーティリティです。このユーティリティにはいくつかの重要な利点があります。

- 直感的かつ可読性の高いスクリプト
- 組み込み同期化
- すべてのオブジェクトの詳細な情報
- クライアント・ウィンドウ内で対話的に作業することが可能

本章には，次の項が含まれます。

- LoadRunner Citrix エージェントについて
- Citrix エージェントからの利点
- インストール
- Citrix エージェントの効果と記憶容量
- サンプル・スクリプト

以降の情報は，Citrix ICA プロトコルのみを対象とします。

LoadRunner Citrix エージェントについて

LoadRunner Citrix エージェントは，Citrix サーバにインストールできる任意のユーティリティです。このユーティリティは，LoadRunner の CD に含まれており，任意の Citrix サーバにインストールできます。

エージェントは，ロード・ジェネレータ・マシンに対してクライアント・ウィンドウ内のオブジェクトとイベントの詳細を提供します。また，オブジェクト

固有のステップを追加して，クライアント画面内で対話的に作業できるようにします。

Citrix エージェントからの利点

Citrix エージェントは，次の領域で向上しました。

- **オブジェクトの詳細 :** Citrix クライアント・ウィンドウ内の個々のオブジェクトについての詳細情報を提供します。
- **アクティブ・オブジェクトの認識** : クライアント・ウィンドウ内のどのオブジェクトが VuGen によって認識されるかを示します。
- **拡張された右クリック・メニュー** : 同期化ステップ，検証ステップ，およびテキスト取得ステップを追加できる，右クリック・メニュー項目が追加されました。
- **テキストの取得** : スクリプトにテキスト検索を挿入できます。

オブジェクトの詳細

Citirix エージェントがインストールされると，VuGen はアクションに関する一般情報の代わりにアクティブ・オブジェクトの特定の情報を記録します。例えば，VuGen は，エージェントなしで生成される **Mouse Click** および **Mouse Double Click** の代わりに **Obj Mouse Click** および **Obj Mouse Double Click** ステップを生成します。

次の例は，エージェントをインストールした場合とインストールしない場合とで記録された同じマウスクリック操作を示しています。エージェントが 1 つだと，VuGen はクリック，ダブルクリック，リリースなどのすべてのマウス・アクションに **ctrx_obj_xxx** 関数を生成します。

```
/* エージェントをインストールしない場合 */
ctrx_mouse_click(573, 61, LEFT_BUTTON, 0, test3.txt - Notepad);

/* エージェントをインストールした場合 */
ctrx_obj_mouse_click("<text=test3.txt - Notepad class=Notepad>" 573,
        61, LEFT_BUTTON, 0, test3.txt - Notepad=snapshot21,
CTRX_LAST);
```

上の例で，**ctrx_obj_mouse_click** 関数の 1 番目の引数には，ウィンドウのタイトルとクラスのテキスト，つまりメモ帳が格納されています。エージェントは

個々のオブジェクトに関する追加情報を提供しますが，仮想ユーザはウィンドウ名とオブジェクトの座標によってのみ，オブジェクトにアクセスします。

アクティブ・オブジェクトの認識

エージェントをインストールすると，クライアント・ウィンドウでどのようなオブジェクトが VuGen によって検出されたのかを調べることができます。これには，現在のウィンドウ内の編集ボックスやボタン，項目リストなど，Windows の基本的なオブジェクトがすべて含まれます。

どのオブジェクトが検出されたのかを調べるには，スナップショットの上でマウスを動かします。マウスがオブジェクト上を通過すると，検出されたオブジェクトの境界が強調表示されます。

次の例では，検出されたオブジェクトの 1 つは［**はい**］ボタンです。

拡張された右クリック・メニュー

スナップショット内をクリックした後，右クリック・メニューを使用して，いくつかの関数をスクリプトに挿入できます。エージェントがインストールされていない場合は，**Insert Mouse Click**，**Insert Mouse Double Click**，および **Insert Sync on Bitmap** に限られます。256 色表示を使用している場合は，右クリック・メニューから **Insert Sync on Bitmap** ステップを追加するためのオプションは使用できません。

エージェントがインストールされている場合は，フォーカスを得ているウィンドウの中で，右クリック・メニューから **Insert Get Text**，**Insert Obj Get Info**，および **Insert Sync on Obj Info** の追加オプションを使用できます。これらのコマンドは対話形式であり，スクリプトに挿入するときはスナップショット内のオブジェクトまたはテキスト領域をマークします。

Insert Obj Get Info 関数と **Insert Sync on Obj Info** 関数は，オブジェクトの状態に関する情報として，ENABLED，FOCUSED，VISIBLE，TEXT，CHECKED，および LINES を提供します。**ctrx_sync_on_obj_info** 関数として生成された **Insert Sync on Obj Info** ステップは，特定の状態を待機してからスクリプトを続行するよう VuGen に指示します。**ctrx_get_obj_info** 関数として生成された **Insert Obj Get Info** ステップは任意のオブジェクトのプロパティの現在の状態を取得します。**Insert Get Text** ステップについては，95 ページ「テキストの取得」で説明します。

次の例では，**ctrx_sync_on_obj_info** 関数は［フォント］ダイアログ・ボックスがフォーカスされるまで待機することで同期化を行います。

```
ctrx_sync_on_obj_info("Font", 31, 59, FOCUSED, "TRUE", CTRX_LAST);
```

VuGen のオブジェクト検出機能を使用して，特定のオブジェクトに対してスナップショットの中から対話形式でアクションを実行できます。

エージェントの機能を使用して関数を対話形式で挿入するには，次の手順を実行します。

1 ツリー・ビューの中で，新しいステップを挿入する場所をクリックします。スナップショットが表示されていることを確認します。

2 スナップショットの中をクリックします。

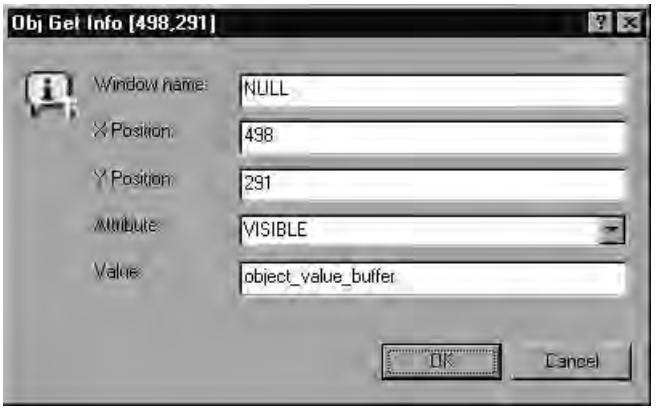
3 ビットマップをマークするには，ビットマップを右クリックして［**Insert Sync on Bitmap**］を選択します。

カーソルをドラッグして対象領域をマークする必要があることを示すメッセージが表示されます。［**OK**］をクリックし，選択するビットマップを対角方向にドラッグします。

マウスのボタンを放すと，スクリプトで現在選択されているステップの後に，ステップが挿入されます。

4 他のすべてのステップには，スナップショット・オブジェクト上にマウスを移動し，どの項目がアクティブかを特定します。VuGen では，マウスがアクティブ・オブジェクトの境界線上を通過すると境界線が強調表示されます。

［挿入］コマンドの 1 つを右クリックして選択します。ダイアログ・ボックスが開き，ステップのプロパティが表示されます。



必要なプロパティを設定して［**OK**］をクリックします。VuGen によってステップがスクリプトに挿入されます。

テキストの取得

エージェントをインストールすると，標準のテキストをバッファに保存できます。VuGen では純粋なテキストのみ保存できます。画像の形式でグラフィカルに表現されているテキストは保存できません。

テキストは，記録中または記録後に，**Get Text** ステップを使用して保存します。記録中は，追加の［**テキスト取得**］ツールバー・ボタンが表示されます。

記録後は，スナップショットの右クリック・メニューから **Get Text** ステップを挿入します。

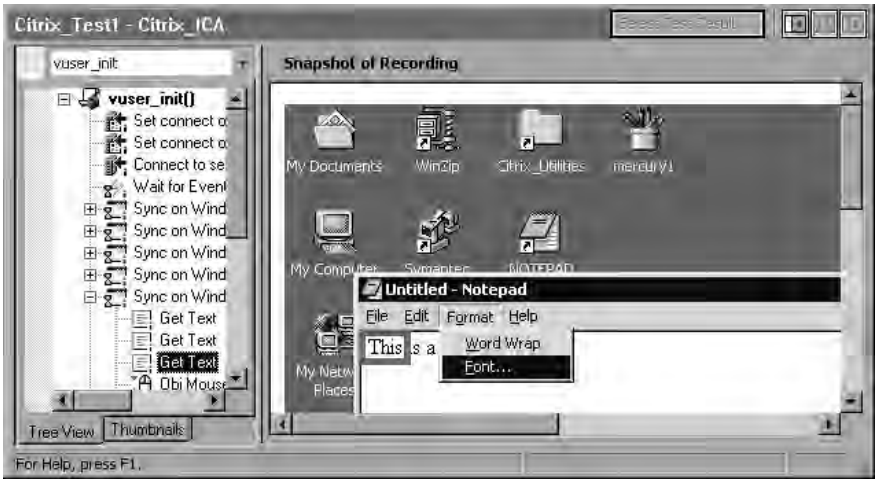
文字列を取得するには，次の手順を実行します。

1 記録中：［**テキスト取得**］ボタンをクリックします。

記録後：スナップショットの右クリック・メニューから［**Insert Get Text**］を選択します。［ビットマップの選択］ダイアログが開き，同期化関数または情報提供関数を挿入するため領域をマークする必要があることを示します。

2 キャプチャするテキストの角をクリックし，マウスを対角方向にドラッグして保存するテキストをマークし，マウス・ボタンを離します。

VuGen によって，**Get Text** ステップが現在の位置に置かれ，テキストがバッファに保存されます。保存されたテキストはピンク色のボックスでマークされます。次のスナップショットでは，**Get Text** ステップによってテキスト **This** が取得されています。



インストール

Citrix エージェントのインストール・ファイルは，LoadRunner CD #2 の Additional Components¥CitrixAgent フォルダに含まれています。Citrix エージェントのインストールに必要なディスク領域は，25 MB です。

Citrix エージェントは，ロード・ジェネレータ・マシンではなく，Citrix サーバ • マシンにのみインストールされなければなりません。

Citrix エージェントをアップグレードする場合は，新しいバージョンをインストールする前に，前のバージョンをアンインストールしてください。

Citrix エージェントをインストールするには，次の手順を実行します。

1 サーバへのソフトウェアのインストールに管理者権限が必要な場合は，サーバに管理者としてログインします。

2 LoadRunner CD #2 の Additional Components¥CitrixAgent フォルダでインストール・ファイル，**CitrixAgent.exe** を見つけます。

注 : インストール後，LoadRunner から Citrix セッションが呼び出されると Citrix エージェントがアクティブになります。LoadRunner なしで Citrix セッションを開始してもアクティブにはなりません。

Citrix エージェントを無効にするには，アンインストールする必要があります。

Citrix エージェントをアンインストールするには，次の手順を実行します。

1 サーバからのソフトウェアの削除に管理者権限が必要な場合は，サーバに管理者としてログインします。

2［**スタート**］>［**LoadRunner Citrix Agent**］>［**Uninstall LoadRunner Citrix Agent**］を選択し，アンインストールの手順に従います。

あるいは，サーバ・マシンのコントロール・パネルから [**プログラムの追加と削除**] を開きます。［**LoadRunner Citrix Agent**］を選択し，［**変更と削除**］をクリックします。

Citrix エージェントの効果と記憶容量

エージェントのインストールされた Citrix 仮想ユーザを実行すると，各仮想ユーザは **ctrxagent.exe** の独自の手順を実行します。その結果，サーバ・マシンで実行できる仮想ユーザの数がわずかに減少します（約 7%)。

Citrix 仮想ユーザごとの記憶容量（各 仮想ユーザがそれぞれ **ctrxagent.exe** プロセスを実行した場合）は約 4.35 MB です。25 仮想ユーザを実行するには，110 MB のメモリが必要です。

サンプル・スクリプト

次のスクリプトには，エージェントを含む実際の Citrix ICA セッションが示されています。

```
vuser_init()
{
  ctrx_set_connect_opt (NETWORK_PROTOCOL, "TCP/IP + HTTP");
  ctrx_connect_server ("Plato", "test", lr_decrypt("428c4445a14409b9"),
"QAlab");
  ctrx_wait_for_event ("LOGON");
  ctrx_sync_on_window ("ICA Seamless Host Agent", ACTIVATE, 0,
0,391,224, "snapshot1", CTRX_LAST);
  ctrx_sync_on_text (196, 198, "OK", TEXT, "ICA Seamless Host
Agent=snapshot2", CTRX_LAST);
  ctrx_obj_mouse_click ("<class=Button text=OK>", 196, 198,
LEFT_BUTTON, 0, "ICA Seamless Host Agent=snapshot2",
CTRX_LAST);
  lr_think_time(73);
return 0;
}
```

第 4 部

クライアント・サーバ・プロトコル

第 8 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，以下の項目について説明します。

- データベース仮想ユーザ・スクリプトの作成について
- 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
- データベースの詳細記録オプション
- 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（Sybase CtLib，Sybase DbLib，Informix，MS SQL Server，Oracle 2-Tier，ODBC，DB2 CLI）および ERP/CRM Siebel 仮想ユーザ・スクリプトのみを対象とします。

データベース仮想ユーザ・スクリプトの作成について

サーバと通信を行っているデータベース・アプリケーションを記録すると，VuGen によってデータベース仮想ユーザ・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。例えば，**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 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「実行環境の設定」を参照してください。

VuGen を使って，記録された定数をパラメータで置き換えることによって，スクリプトをパラメータ化できます。詳細については，『**第 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「VuGen パラメータを使った作業」を参照してください。

さらに，スクリプトのクエリや他のデータベース・ステートメントを相関させて，特定のクエリの結果を別のクエリに結び付けることができます。詳細については，『**第 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「ステートメントの相関」を参照してください。

トラブルシューティング情報とスクリプト作成のヒントについては，『**第 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「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。

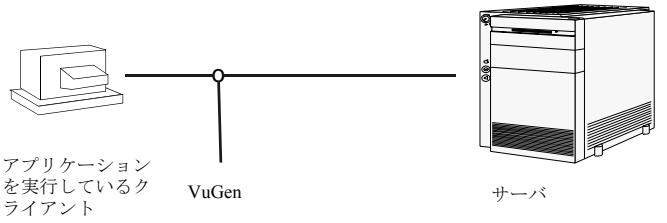
- サーバへの接続
- SQL クエリの発行
- 情報の検索と処理
- サーバからの切断

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。

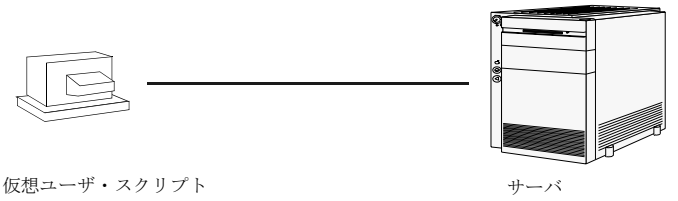
サーバ API への呼び出しが含まれるプログラムをデータベース（DB）仮想ユーザ・スクリプトといいます。データベース仮想ユーザ・スクリプトによって，クライアント・アプリケーションと，そのすべての操作がエミュレートされます。仮想ユーザによってスクリプトが実行されると，クライアント / サーバ・システムにユーザ負荷をかけた状態がエミュレートされます。仮想ユーザによって生成されるパフォーマンス・データは，レポートやグラフを使って分析できます。

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 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「VuGen を使った記録」を参照してください。

UNIX 環境では，VuGen のテンプレートを土台にしてスクリプトのプログラミングを行うことにより，DB 仮想ユーザ・スクリプトを作成できます。UNIX での DB 仮想ユーザ・スクリプトのプログラミングの詳細については，『**第 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の付録 「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略を説明します。

データベース仮想ユーザ・スクリプトを作成するには，次の手順を実行します。

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 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「VuGen を使った記録」を参照してください。

2 **スクリプトを拡張します。**

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 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「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，同じクエリを異なる値を使って反復実行できます。

詳細については，『**第 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「VuGen パラメータを使った作業」を参照してください。

4 **クエリを相関させます（任意）。**

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

詳細については，『**第 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「ステートメントの相関」を参照してください。

5 **実行環境を設定します。**

実行環境を設定することによって，スクリプト実行時の仮想ユーザ・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 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「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 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「スタンドアロン・モードでの仮想ユーザ・スクリプトの実行」を参照してください。

スクリプトを作成した後，スクリプトを自分の環境（LoadRunner シナリオ，Performance Center 負荷テスト，Tuning Module セッション，またはビジネス・プロセス・モニタ・プロファイル）に統合します。詳細については，『**Mercury LoadRunner コントローラ・ユーザーズ・ガイド**』，**Tuning Console**，**Performance Center**，または **Application Management** のマニュアルを参照してください。

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，次の手順を実行します。

1 ［**ツール**］＞［**記録オプション**］を選択します。［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。

- ［**送信データのタイム 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**］＞［**スクリプトを再生成**］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**］：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 LRD 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。このプロトコルはシングル・プロトコルのスクリプトにのみ適用されます。

詳細記録オプションを設定するには，次の手順を実行します。

1 ［記録オプション］ダイアログ・ボックスの［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します。

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。例えば，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_oci8_to_oci7	Oracle OCI 8 接続を Oracle OCI 7 接続に変換します。
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_lob_locator_assign	ラージ・オブジェクト・ロケータを別のラージ・オブジェクト・ロケータに割り当てます。
lrd_ora8_lob_locator_temporary	一時ラージ・オブジェクトを作成します
lrd_ora8_lob_read	ラージ・オブジェクト記述子から文字を読み取ります。
lrd_ora8_lob_write	ラージ・オブジェクト記述子に文字を書き込みます。
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_print	Oracle autofetch 操作によって取得された行を出力します。

lrd_ora8_save_last_rowid	行 ID をパラメータに保存します。
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・クエリによって返されたデータを表示するグリッド
- クエリ中に取り出された行の数

関数の並び

VuGen ウィンドウに仮想ユーザ・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。例えば，標準的な Oracle データベース・セッションでは，次のような関数の並びが記録されます。

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

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

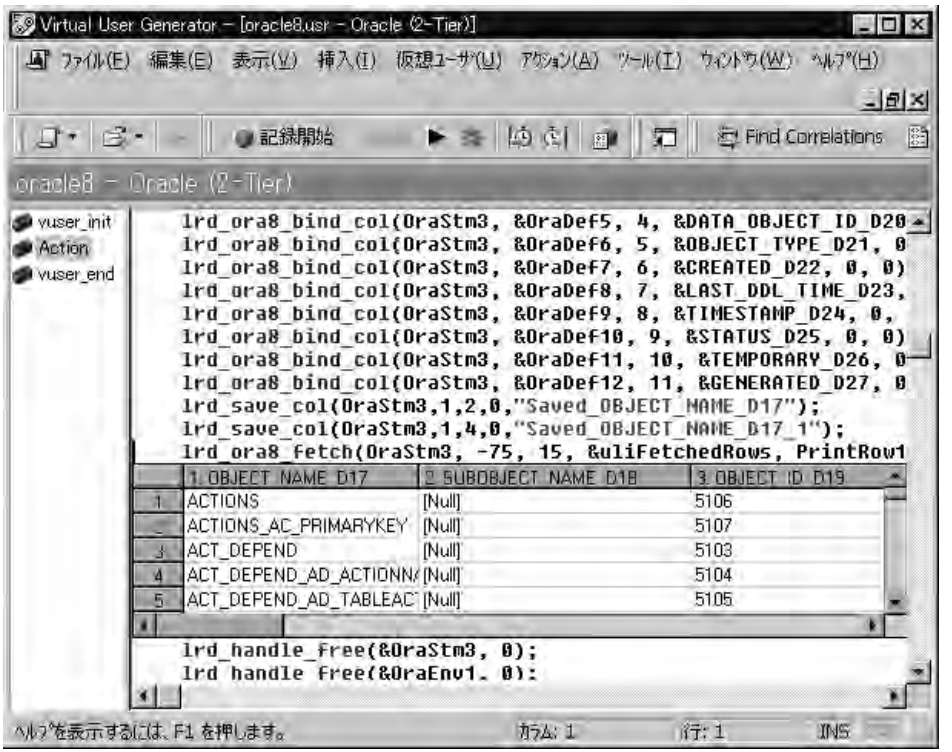
```
lrd_init(&InitInfo, DBTypeVersion);
lrd_open_connection(&Con1, LRD_DBTYPE_ORACLE, "s1", "tiger",
"hp1", "", 0, 0, 0);
lrd_open_cursor(&Csr1, Con1, 0);
lrd_stmt(Csr1, "select parameter, value     from v$nls_parameters "
      "    where (upper(parameter) in ('NLS_SORT','NLS_CURRENCY',"
      "'NLS_ISO_CURRENCY',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

グリッド

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

データ・グリッドは，**GRID** ステートメントまたは Oracle 8 の場合は **GRID8** ステートメントによって表されます。データ・グリッドを開くには，GRID ステートメントの横の余白部分をクリックします。

次の例では，ウィンドウにフライト予約データベースを対象に実行されたクエリが表示されています。クエリでは，フライト番号，航空会社コード，出発地，日付，およびその他のフライトに関する情報が取得されています。）



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。

値を相関する，あるいはファイルにデータを保存する場合は，セル内でクリックして，右クリック・メニューから［**相関を作成**］または［**ファイルに保存**］を選択します。

行情報

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

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

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

正数の値

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

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

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

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

負数の値

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

次の例では，結果セットの 4 行すべてを取り出しています。

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

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

lrd_fetch の詳細については，「**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。例えば，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。例えば，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。次の方法で振る舞いを適用できます。

- 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。例えば，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。

例えば，以下のデータベース・ステートメントが失敗した場合は（例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 9 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。

- データベース仮想ユーザ・スクリプトの相関について
- スクリプトでの相関候補の検索
- 既知の値の相関
- データベース相関関数
  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。

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，クエリを相関させることによって問題を解決できます。クエリの相関とは，実行時の値をパラメータに保存することです。保存した値は，同じスクリプト内の以降の個所で使用します。つまり，相関とは，あるステートメントの結果を別のステートメントへの入力として使用することです。

多くのクエリは，その入力が前のクエリの結果に依存します。この動作をエミュレートするには，VuGen の相関機能を使用します。

スクリプトでの相関候補の検索

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

- ➤ 相関候補を検索する。
- ➤ 適切な相関関数を挿入して，結果をパラメータに保存する。
- ➤ ステートメントの値をパラメータで置き換える。

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

この項では，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分かっている場合は，次の項に進んで，特定の値を相関させる方法を参照してください。

自動相関でスクリプトを検出して相関させるには，次の手順を実行します。

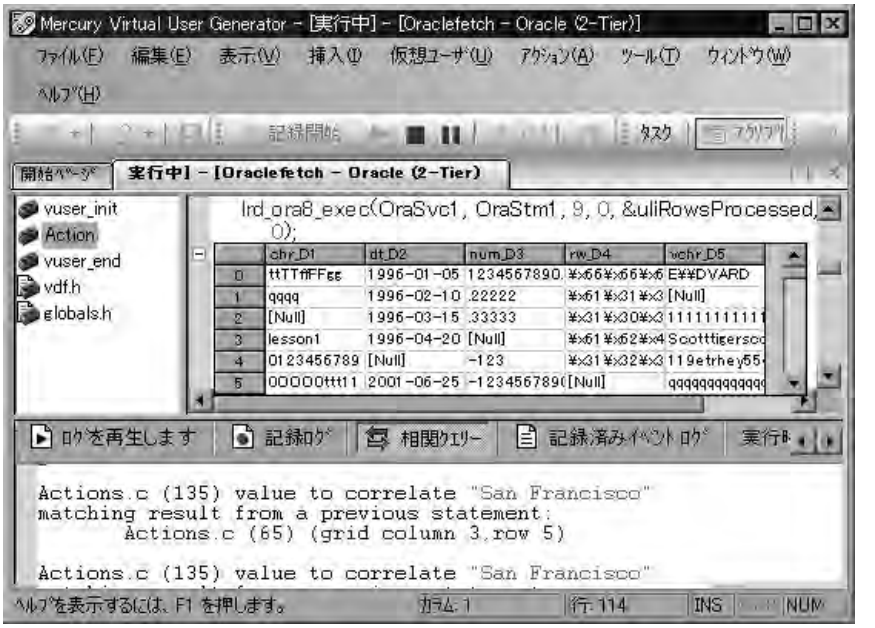
1［出力］ウィンドウを開きます。

［**表示**］＞［**出力ウィンドウ**］を選択して，ウィンドウの下部に出力タブを表示します。［Replay Log］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で検索が行われ，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** 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，相関させるクエリ結果をダブルクリックします。語句（**grid column x, row y**）をクリックします。グリッド内の値の位置にカーソルが移動します。

4 右クリックして表示されるメニューから［**相関を作成**］を選択します。結果の値を保存するパラメータの名前を入力するように求められます。

5 名前を指定するか，標準の名前をそのまま使用します。［**OK**］をクリックして作業を続けます。VuGen によって，パラメータに結果を保存する適切な相関ス

テートメント（**lrd_save_value**，**lrd_save_col**，**lrd_save_ret_param**，**lrd_ora8_save_col** のいずれか）が挿入されます。

6［**はい**］をクリックして相関を確定します。

7 スクリプト内に現れる値をすべて検索するかどうかを尋ねるメッセージ・ボックスが開きます。選択したステートメント内の値だけを置き換える場合は，［**いいえ**］をクリックします。次の候補を検索して相関させる場合は，［**はい**］をクリックします。

8［検索と置換］ダイアログ・ボックスが開きます。オリジナルのステートメントも含め，すべての置き換えを確認します。

9［検索と置換］ダイアログ・ボックスを閉じます。ステートメントの値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

相関させる必要がある値が分かっている場合は，次の手続きを行います。

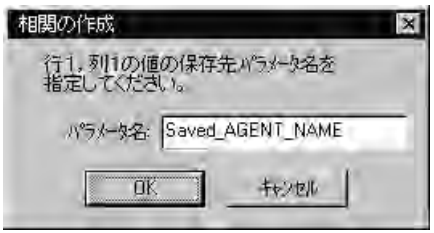
特定の値を相関させるには，次の手順を実行します。

1 相関させる値を含むクエリを使って，スクリプト内のステートメントを検索します。これは，通常 **lrd_assign**，**lrd_assign_bind**，**lrd_stmt** のいずれかの関数の引数です。引用符を除く値を選択します。

2 右クリックして表示されるメニューから［**相関を検索（カーソル位置）**］を選択します。選択した値の相関が検索されます。

3 出力ウィンドウの［**相関クエリー**］タブで，相関させるクエリ結果をダブルクリックします。語句（**grid column x, row y**）をクリックします。グリッド内の値の位置にカーソルが移動します。

4 グリッド内で，相関させる値をクリックし，右クリックして表示されるメニューから［**相関を作成**］を選択します。結果の値を保存するパラメータの名前を入力するように求められます。



5 名前を指定するか，標準の名前をそのまま使用します。［**OK**］をクリックして作業を続けます。VuGen によって，パラメータに結果を保存する適切な相関ステートメント（**lrd_save_value**，**lrd_save_col**，**lrd_save_ret_param,** **lrd_ora8_save_col** のいずれか）が挿入されます。

6［**はい**］をクリックして相関を確定します。

7 スクリプト内に現れる値をすべて検索するかどうかを尋ねるメッセージ・ボックスが開きます。選択したステートメント内の値だけを置き換える場合は，［**いいえ**］をクリックします。次の候補を検索して相関させる場合は，［**はい**］をクリックします。

8［検索と置換］ダイアログ・ボックスが開きます。オリジナルのステートメントも含め，すべての置き換えを確認します。

9［検索と置換］ダイアログ・ボックスを閉じます。ステートメントの値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も消去されます。

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）を使用するときは，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（Oracle 8 以降は **lrd_ora8_save_col** ）。
- **lrd_save_value** 関数では，プレースホルダ記述子の現在値がパラメータに保存されます。出力プレースホルダを設定するデータベース関数（Oracle の特定のストアド・プロシージャなど）と一緒に使用します。
- **lrd_save_ret_param** 関数では，ストアド・プロシージャの戻り値がパラメータに保存されます。この関数は主に，戻り値を生成する DbLib 内のデータベース・プロシージャと組み合わせて使用します。

注： VuGen では，保存された値が無効または NULL（行が返されない）の場合，相関が適用されません。

これらの関数と引数の詳細については，「**オンライン関数リファレンス**」を参照してください。

第 10 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，以下の項目について説明します。

- DNS 仮想ユーザ・スクリプトの作成について
- 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，仮想ユーザ API 関数，C 関数を使ってスクリプトをプログラミングします。これらの関数の詳細については，「**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セッション・ステップまたはシナリオに組み込みます。仮想ユーザ・スクリプトのセッション・ステップまたはシナリオへの組み込み方法の詳細については，『**Mercury LoadRunner コントローラ・ユーザーズ・ガイド**』または『**Tuning Module User's Guide**』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 11 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・スクリプトの記録について
- Windows Sockets 仮想ユーザ・スクリプトの作成の概要
- WinSock 記録オプションの設定
- LRT 関数の使用

以降の情報は，Web/Winsock デュアル・プロトコルなど，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。例えば，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。

例えば，**telnet** アプリケーションのアクションを記録して，スクリプトを作成できます。

次に示す例では，**lrs_send** を使って指定したソケットにデータを送信しています。

```
lrs_send("socket22", "buf44", LrsLastArg);
```

記録されたスクリプトは，VuGen のメイン・ウィンドウで表示および編集ができます。このウィンドウには，セッション中に記録された Windows Sockets API 呼び出しが表示されるので，記録時のネットワークの動作状況を追うことができます。

VuGen では，WinSock スクリプトを次の 2 つの方法で表示できます。

- スクリプトをアイコン形式で表示します。これは標準設定のビューで，「**ツリー・ビュー**」といいます。
- スクリプトをテキスト形式で表示して Windows Sockets API 呼び出しを示します。これを「**スクリプト・ビュー**」といいます。

VuGen では，スクリプトをツリー・ビューとスクリプト・ビューの両方で表示して編集できます。詳細については，『**第 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「VuGen の紹介」を参照してください。

スクリプトを作成したら，記録したデータをスナップショットか未処理のデータ・ファイルで表示できます。詳細については，第 12 章「Windows Sockets データを使った作業」を参照してください。

Windows Sockets 仮想ユーザ・スクリプトの作成の概要

本項では，VuGen を使って Windows Sockets 仮想ユーザ・スクリプトの作成プロセスの概略を説明します。

Windows Sockets スクリプトを作成するには，次の手順を実行します。

1 **VuGen を使ってア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。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 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「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。**

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 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「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 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「パラメータの作成」を参照してください。

4 **ステートメントを相関させます（任意）。**

ステートメント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 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「ステートメントの相関」を参照してください。

5 **実行環境を設定します。**

実行環境を設定することによって，スクリプト実行中の仮想ユーザの動作を制御します。設定には，ループ，ログ，タイミング情報などがあります。

詳細については，「実行環境の設定」および『**第 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「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認します。

詳細については，『**第 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「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」のを参照してください。

スクリプトを作成した後，スクリプトを自分の環境（LoadRunner シナリオ，Performance Center 負荷テスト，Tuning Module セッション，または Business Process Monitor プロファイル）に統合します。詳細については，『**Mercury**

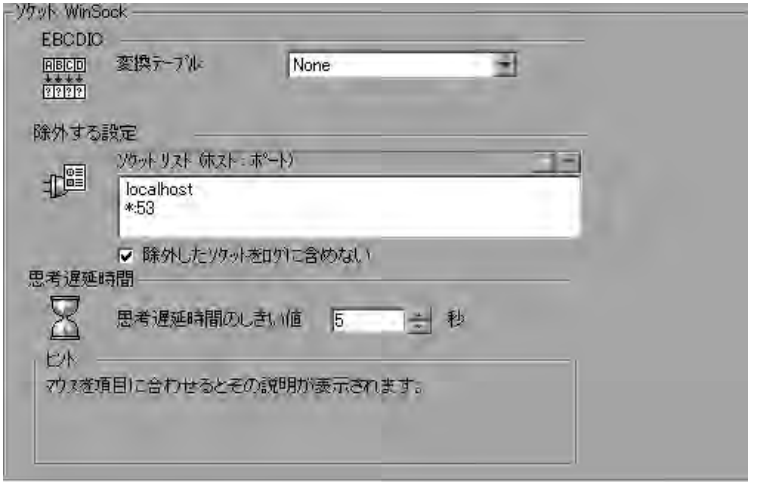
LoadRunner コントローラ・ユーザーズ・ガイド』，**Tuning Console**，**Performance Center**，または **Application Management** のマニュアルを参照してください。

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，VuGen のインストール先ディレクトリの **ebcdic** ディレクトリにあります。システムで別の変換テーブルを使用している場合，テーブルを **ebcdic** ディレクトリにコピーします。

注：データが ASCII 形式の場合，変換の必要はありません。［**None**］オプション（標準設定）を選択します。変換テーブルを選択した場合は，VuGen は ASCII データを変換します。

Solaris マシンで作業しているときは，仮想ユーザ・スクリプトを実行するすべてのマシンで次の環境変数を設定する必要があります。

```
setenv LRSDRV_SERVER_FORMAT 0025
setenv LRSDRV_CLIENT_FORMAT 04e4
```

ソケットの除外

VuGen ではソケット除外機能を使用して，記録セッションから特定のソケットを除外できます。スクリプトから特定のソケットを対象とするすべてのアク

ションをから除外するには，［除外する設定］の［ソケット リスト］からそのソケットのアドレスを選択します。ソケットをこのリストに追加するには，ボックスの右上角にあるプラス記号をクリックし，ソケットのアドレスを次のいずれかの形式で入力します。

値	意味
ホスト : ポート	指定したホストの指定したポートだけを除外します。
ホスト	指定したホストのすべてのポートを除外します。
: ポート	ローカル・ホストの指定したポートを除外します。
*: ポート	すべてのホストの指定したポートを除外します。

複数のホストとポートを除外するには，それらをリストに追加します。除外対象リストからソケットを削除するには，ソケットのアドレスを選択し，ボックスの右上角にあるマイナス記号をクリックします。ローカル・ホストや DNS ポート（53）など，テスト対象サーバの負荷に影響しないホストとポートを除外することをお勧めします。これらは標準設定では除外されています。

標準設定では，［除外する設定］の［ソケット リスト］で除外されたソケットのアクションがログに記録されることはありません。除外されたソケットのアクションをログに記録するには，［**除外したソケットをログに含めない**］チェック・ボックスをクリアします。除外されたソケットについてのログ記録を有効にすると，ログ・ファイルでは，アクションは「Exclude」という単語の後に記録されます。

```
Exclude :/* recv():15 bytes were received from socket 116 using flags 0 */
```

思考遅延時間のしきい値の設定

VuGen では記録中にオペレータの思考遅延時間が自動的に挿入されます。思考遅延時間のしきい値を設定して，それよりも低い場合には思考遅延時間を無視するようにできます。記録された思考遅延時間がしきい値を越えていると，VuGen によって LRS 関数の前に **lr_think_time** ステートメントが挿入されます。記録された思考遅延時間がしきい値を越えている場合，**lr_think_time** ステートメントは生成されません。

思考遅延時間のしきい値を設定するには，［**思考遅延時間 のしきい値**］ボックスに必要な値（秒単位）を入力します。標準設定の値は 5 秒です。

LRT 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 が動作状況を記録したシーケンスを確認できます。次は，一般的なセッションで記録される関数の並びを示します。

lrs_startup Windows Sockets DLL を初期化します。

lrs_create_socket ソケットを初期化します。

lrs_send データグラムで，またはストリーム・ソケットへデータを送信します。

lrs_receive データグラムまたはストリーム・ソケットからデータを受信します。

lrs_disable_socket ソケットの処理を無効にします。

lrs_close_socket 開いているソケットを閉じます。

lrs_cleanup WinSock DLL の使用を終了します。

VuGen では，Windows 上で Windows Socket 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。

第 12 章

Windows Sockets データを使った作業

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。

- Windows Sockets データを使った作業について
- 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
- スクリプト・ビューでの Windows Sockets データの表示
- 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 データを使った作業について

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。

WinSocket スクリプトをツリー・ビューで表示すると，VuGen によって，データ・バッファ内の移動とデータの修正が可能なスナップショット・ウィンドウが表示されます。

スクリプト・ビューで作業しているときには，**data.ws** ファイルの未処理のデータを表示できます。詳細については，158 ページ「スクリプト・ビューでの Windows Sockets データの表示」を参照してください。

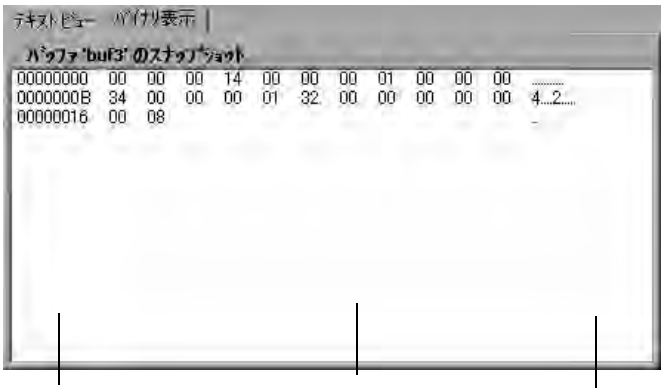
スナップショット・ウィンドウでのデータの表示

ツリー・ビューに Windows Sockets スクリプトを表示すると，編集が可能な［バッファのスナップショット］ウィンドウにデータが表示されます。スナップショットを［**テキスト ビュー**］または［**バイナリ表示**］に表示できます。

［テキスト ビュー］には，バッファのスナップショットの内容がテキストとして表示されます。

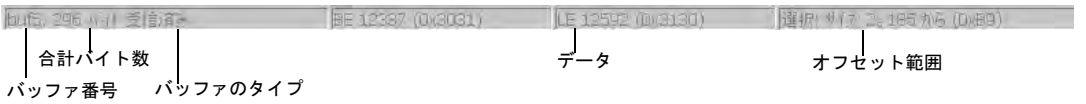
標準設定では，バッファ・データは読み取り専用として保存されます。バッファの内容を変更する場合は，バッファのテキスト • ビューで［**読み取り専用**］ボックスをクリアします。VuGen から，ブックマークとパラメータに影響がある可能性があるという警告が発行されます。

［バイナリ表示］にはデータが 16 進形式で表示されます。左のカラムには，行の最初の文字のオフセットが表示されます。中央のカラムには，データの 16 進値が表示されます。右のカラムには，データが ASCII 形式で表示されます。



バッファのスナップショットの下のステータス・バーには，データとバッファについての情報が提供されます。

- **バッファ番号**：選択されたバッファのバッファ番号。
- **合計バイト数**：バッファの合計バイト数。
- **バッファのタイプ**：バッファのタイプ（「受信済み」または「送信済み」）。
- **データ**：選択したデータの値をリトル・エンディアン順（バッファ内とは逆）の 10 進および 16 進で表示。
- **オフセット**：バッファの先頭からの選択（テキスト・ビュー内でのカーソル位置）のオフセット。複数のバイトを選択した場合は，選択範囲が示されます。



ステータス・バーには元のデータが変更されたかどうかも示されます。

データ内の移動

ツリー・ビューには，データ内を移動して，特定の値の識別と分析を行うためのいくつかのツールがあります。

- ➤ バッファ・ナビゲータ
- ➤ オフセットへの移動
- ➤ 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。

オフセットへの移動

オフセットを指定して，データ・バッファ内を移動できます。バッファ内におけるデータの絶対位置またはカーソルの現在位置に対する相対位置を指定できます。また，このダイアログ・ボックスで先頭と末尾のオフセットを指定することによって，ある範囲のデータを選択することもできます。

特定のオフセットへ移動するには，次の手順を実行します。

1 ［バッファのスナップショット］ウィンドウ内をクリックします。次に，右クリック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**］をクリックして，オフセット値を指定します。

3 カーソルの相対位置へジャンプするには，［**次だけ前に進む**］を選択して，移動するバイト数を指定します。バッファ内を前進する場合は，正の数値を入力します。後退する場合は，負の数値を入力します。

4 バッファ内で，ある範囲のデータを選択するには，［**次の範囲を選択**］を選択して，先頭と末尾のオフセットを指定します。

ブックマーク

バッファ内の場所にブックマークとして印を付けることができます。それぞれのブックマークにわかりやすい名前を付けます。ブックマークをクリックすると，直接その場所に移動します。ブックマークは，バッファのスナップショットの下にある出力ウィンドウの［**ブックマーク**］タブに表示されます。

ブックマークは，テキスト・ビューでもバイナリ・ビューでも使用できます。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・ビューの中でそのブックマークに直接ジャンプできます。

ブックマークは 1 バイトにも複数バイトにも付けられます。リスト中のブックマークをクリックすると，対応する場所が選択された状態で［バッファのスナップショット］ウィンドウに表示されます。初期設定では，そのデータがテキスト・ビューでは青で強調表示され，バイナリ・ビューではブックマーク・ブロックが赤で表示されます。また，バイナリ・ビューでブックマークにカーソルを置くと，ポップアップ・テキスト・ボックスが開いてブックマークの名前が表示されます。

永久ブックマークと標準ブックマークを作成できます。永久ブックマークは，バッファの「バイナリ・ビュー」内で常に印が付けられています（青いボックスで囲まれています)。バッファ内の異なる位置を指している場合でも，このブックマークは選択された状態で青いボックスの中に表示されています。カーソルの位置は赤でマークされます。一方，標準ブックマークには常に印が付けられているわけではありません。標準ブックマークは，そこにジャンプしたときには赤く印が付きますが，バッファ内でカーソルを移動すると選択されていない状態になります。標準設定のブックマークは永久ブックマークです。

ブックマーク処理するには，次の手順を実行します。

1 ブックマークを作成するには［バッファのスナップショット］（テキスト・ビューまたはバイナリ・ビュー）で 1 つ以上のバイトを選択し，右クリックで表示されるメニューから［**新規ブックマーク**］を選択します。

2 ブックマーク・リストを表示するには，［**表示**］＞［**出力ウィンドウ**］を選択し，［**ブックマーク**］タブを選択します。

3 ブックマークに名前を割り当てるには，ブックマーク・リストのブックマークをクリックして，タイトルを編集します。

4 ブックマークの場所を変更するには，［**ブックマーク**］タブでブックマークを選択してから［バッファのスナップショット］で新しいデータを選択します。［**ブックマーク**］タブの［**変更**］をクリックします。

5 永久ブックマークを標準ブックマークに変更する場合（永久ブックマークは，カーソルを新しい場所に移動しても常にマークされた状態を維持します）は，ブックマークを選択してマウスを右クリックし，［**永久ブックマーク**］の横にあるチェックをクリアします。

6 リストに永久ブックマークだけを表示するには，［**ブックマーク**］タブで［**永久ブックマークのみ表示する**］チェック・ボックスを選択します。

7 特定のバッファのブックマークを表示するには，そのバッファからブックマークを選択し，［**フィルタ**］ボックスの［**選択バッファのみ**］を選択します。

8 ブックマークを削除するには，［**ブックマーク**］タブでブックマークを選択して［**削除**］をクリックします。

バッファ・データの修正

ツリー・ビューには，既存のデータを削除，変更，追加することによってデータを修正するためのいくつかのツールがあります。

- データの挿入
- データの編集
- データのパラメータ化

データの挿入

データ・バッファに数値を挿入できます。数値はシングルバイト，ダブルバイト，または 4 バイト値として挿入できます。

データ・バッファに数値を挿入するには，次の手順を実行します。

1 バッファ内をクリックします。

2 右クリックして表示されるメニューから［**詳細設定**］＞［**数値の挿入**］＞［**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。

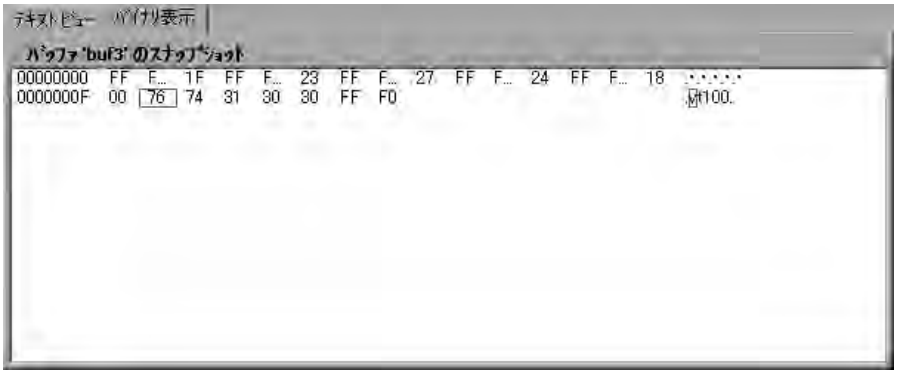
4 挿入するデータのサイズとして，1 バイト，2 バイト，または 4 バイトを［**サイズ**］ボックスから選択します。

5［**OK**］をクリックして完了します。VuGen によってデータの 16 進表現がバッファに挿入されます。

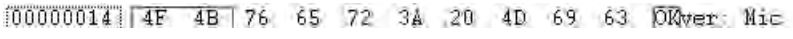
データの編集

バッファ・データに対して，標準的な編集操作（コピー，貼り付け，切り取り，削除，元に戻す）が行えます。バイナリ・ビューでは，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・ビューの右カラムには，10 進数または 16 進数を表示できます。

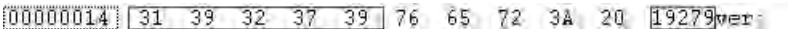
次の例では「OK」という語が選択されています。



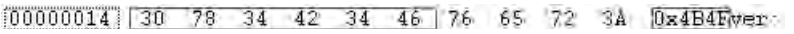
データの次行の先頭で単純なコピー（CTRL+C）と貼り付け（CTRL+V）操作を実行すると，実際のテキストが挿入されます。



［**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・ビューでバッファ・データを編集するには，次の操作を行います。

1 バッファ・データをコピーするには，次の手順を実行します。

- ➤ 文字としてコピーする場合は，1 つ以上のバイトを選択し，CTRL+C キーを押します。
- ➤ 10 進数としてコピーする場合は，右クリックして表示されるメニューから［**詳細設定**］＞［**数値としてコピー**］＞［**10 進法**］を選択します。
- ➤ 16 進数としてコピーする場合は，右クリックして表示されるメニューから［**詳細設定**］＞［**数値としてコピー**］＞［**16 進法**］を選択します。

2 データを貼り付けるには，次の手順を実行します。

- ➤ 単一バイト（クリップボードのデータのサイズを単一バイトと仮定した場合）として貼り付ける場合は，バッファ内の望みの場所をクリックして CTRL+V キーを押します。
- ➤ 短い形式（2 バイト）として貼り付ける場合は，右クリックして表示されるメニューから［**詳細設定**］＞［**数字の挿入**］＞［**短形式で貼り付け（2 バイト）**］を選択します。
- ➤ 長い形式（4 バイト）として貼り付ける場合は，右クリックして表示されるメニューから［**詳細設定**］＞［**数字の挿入**］＞［**長形式で貼り付け（4 バイト）**］を選択します。

3 データを削除するには，テキスト・ビューまたはバイナリ・ビュー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。パラメータ化する文字列の境界を指定しないと，VuGen によってスクリプトに **lrs_save_param** 関数が挿入されます。境界を指定すると，境界引数を指定できる **lrs_save_searched_string** 関数がスクリプトに挿入されます。

lrs_save_param 関数と **lrs_save_searched_string** 関数によってデータの相関が行われます。つまり，受信したデータが格納され，そのテスト内の以降の任意の場所で使用されます。相関では受信データが格納されるので，受信バッファにだけ適用され，送信バッファには適用されません。お勧めする手順は，受信バッファ内でパラメータ化する動的データの文字列を選択することです。同じパラメータを以降の送信バッファで使用します。

このタイプの相関を，単純なパラメータ化と混同しないでください。単純なパラメータ化（［**挿入**］＞［**新規パラメータ**］）は送信バッファ内のデータにだけ適用されます。パラメータを設定し，それに複数の値を割り当てます。VuGen によって，テストの実行ごと，または反復ごとに異なる値が割り当てられたパラメータが使用されます。詳細については，『**第 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「VuGen パラメータを使った作業」を参照してください。

次の各項では，受信バッファのデータの相関について説明します。

パラメータ作成後，VuGen によって文字列をパラメータに置き換えたすべての場所が表示されます。パラメータ作成についての情報，つまりパラメータが作成されたバッファやバッファ内のオフセットなどの情報も表示されます。［バッファのスナップショット］ビューの下の出力ウィンドウの［**パラメータ**］タブに，パラメータのすべての出現箇所のリストが表示されます。

VuGen でパラメータを操作できます。

絞り込み：パラメータ名を指定してパラメータ置換の絞り込み表示ができます。

ソースへの移動：置換を選択して［ソースに移動］をクリックすると，バッファ内の置換されたパラメータの場所に移動します。

パラメータの削除：任意のパラメータを削除できます。パラメータを削除すると，データが元の値に置き換わり，スクリプトからパラメータ化関数が削除されます。

名前：各置換に名前を付けることができます。

置換を元に戻す：リストに表示された 1 つまたは複数の置換を元に戻すことができます。

スナップショット・ウィンドウでデータのパラメータ化を行うには，次の手順を実行します。

1 パラメータ化するデータを選択して，右クリックすると表示されるメニューから［**パラメータの作成**］を選択します（受信バッファにだけ有効）。ダイアログ・ボックスが開きます。

2［**パラメータ名**］ボックスにパラメータの名前を指定します。

3 パラメータ化するデータ範囲を選択します。標準設定では，バッファで選択されたデータ範囲が使用されます。ダイアログ・ボックスに表示された範囲とは異なる範囲を選択するには，［**範囲の選択**］をクリックします。選択されている範囲を示す小さなダイアログ・ボックスが表示されます。

［バッファのスナップショット］ウィンドウで範囲を選択して［**完了**］をクリックします。

4 パラメータ・データ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。

境界を指定するには，次の手順を実行します。

- ［**境界を使ってパラメータ データを抽出する**］チェック・ボックスを選択します。［**スクリプト ステートメント**］表示枠の関数が **lrs_save_param** から **lrs_save_searched_string** に変更されます。［**完了**］をクリックします。
- ［**境界**］セクションの［**左**］ボックスの隣にある参照ボタンをクリックします。バッファ内の選択を示す小さいダイアログ・ボックスが表示されます。バッファ内の境界を選択して［**完了**］をクリックします。右の境界についてもこの手順を繰り返します。

5［**スクリプト ステートメント**］セクションの引数に必要な修正を行います。例えば，**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。

6［**OK**］をクリックしてパラメータを作成します。パラメータの置換前には確認を求められます。［**はい**］をクリックします。［**パラメータ**］タブにすべての置換を表示できます。

7 バッファ内のパラメータの元の場所に移動するには，パラメータを選択して［**ソースに移動**］をクリックします。

8 選択された置換のバッファの場所に移動するには，パラメータを選択して［**移動**］をクリックします。

9 パラメータ全体を削除するには，［**フィルタ**］ボックスのパラメータを選択して［**パラメータを削除**］をクリックします。

10 置換を取り消すには，［**パラメータ**］タブでパラメータを選択して［**元に戻す**］をクリックします。表示されているパラメータの置換をすべて取り消すには，［**パラメータ**］タブでパラメータを選択して［**すべて元に戻す**］をクリックします。

11 特定の置換を取り消すと，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。

12 パラメータ全体を削除してすべての置換を元に戻すには，［**フィルタ**］ボックスのパラメータを選択して［**パラメータを削除**］をクリックします。

13［**仮想ユーザ**］＞［**パラメータ リスト**］を選択してデータをパラメータに割り当てます。

バッファ名の修正

バッファ名を修正するには，**data.ws** ファイルのスクリプト・ビューを使います。記録後にバッファ名を修正すると，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。

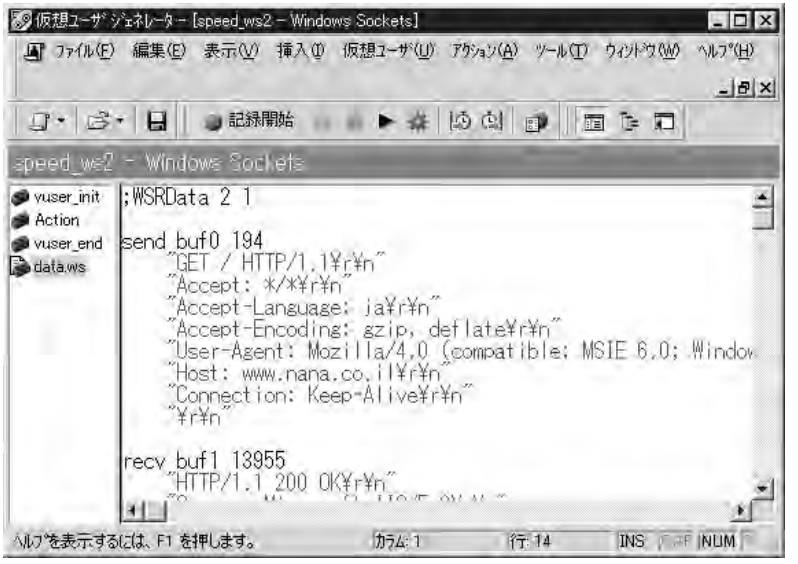
バッファで作成したパラメータが使えなくなっている場合，VuGen によって，パラメータが定義されたバッファ内のパラメータを削除するよう求められます。パラメータを削除すると，他のバッファも含め，すべての置換が元に戻ります。

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と，アクションはスクリプトの 3 つのセクション **vuser_init**，**Actions**，**vuser_end** に記録されます。仮想ユーザ・スクリプトに加えて，記録セッション中に転送または受け取ったデータを含む **data.ws** というデータ・ファイルも作成されます。VuGen のメイン・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，**socket1** で受信したデータが **lrs_receive** によって処理されています。データは 5 番目の受信記録（buf4）に保管されています。バッファ・インデックスは 0（ゼロ）から始まります。**data.ws** ファイルの対応するセクションは，バッファとその内容を示します。

```
recv buf4 39
        "\xff\xfb\x01\xff\xfb\x03\xff\xfd\x01"
        "\r\n"
        "\r\n"
        "SunOS UNIX (sunny)\r\n"
        "\r"
        "\x0"
        "\r\n"
        "\r"
        "\x0"
```

データ・ファイルの形式について

データ・ファイル **data.ws** の形式は以下のとおりです。

- ファイル・ヘッダー
- 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，138 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"
```

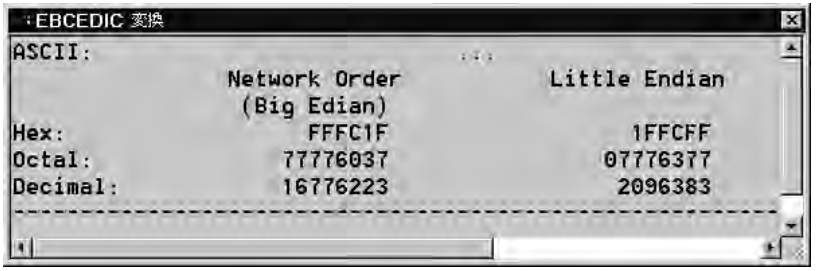
次の例は通常のデータ・ファイルのヘッダー，記述子およびデータを示します。

```
;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。

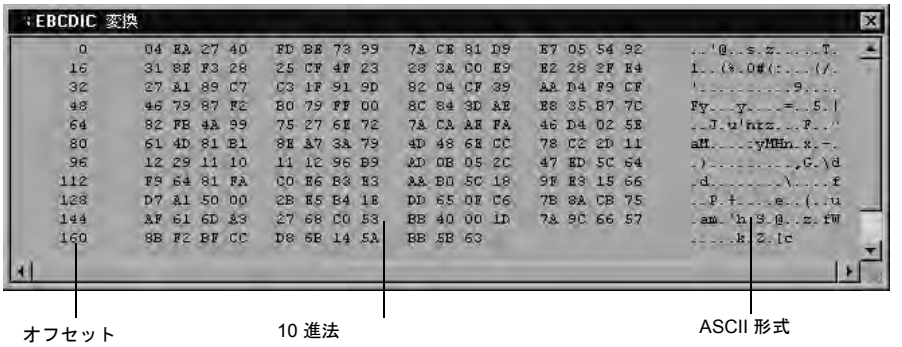
ビューア・ウィンドウでデータを表示するには，データを選択して F7 キーを押します。選択されたテキストが 4 文字以下の場合，VuGen によってそのデータが「**短い形式**」の 16 進，10 進，および 8 進で表示されます。



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，164 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，164 ページ「表示形式の設定」を参照してください。

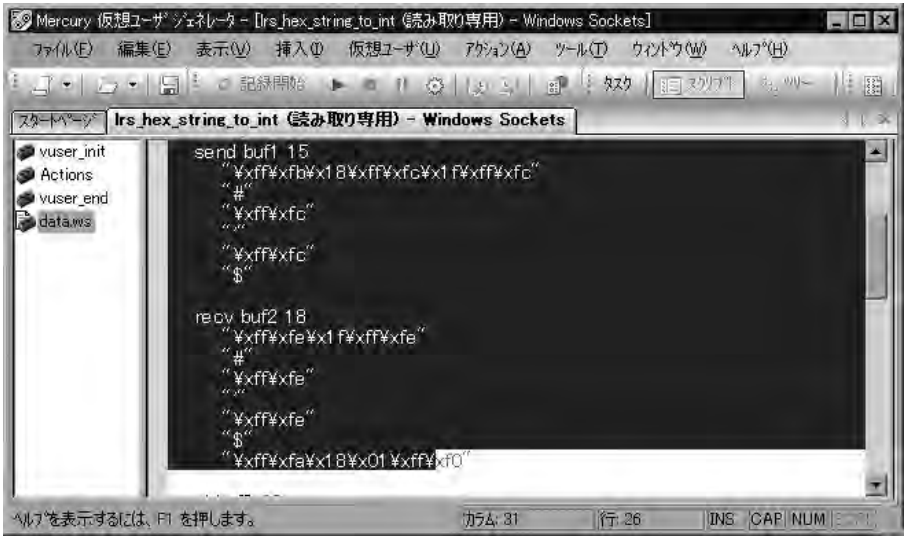
標準設定では，最初のカラムに，マークされた領域の先頭からの文字オフセットが表示されます。2 番目のカラムには，データが 16 進形式で表示されます。3 番目のカラムには，ASCII 形式でデータが表示されます。EBCDIC データを表示するときは，ASCII 形式の非印字文字（「/n」など）はすべてドットで表されます。



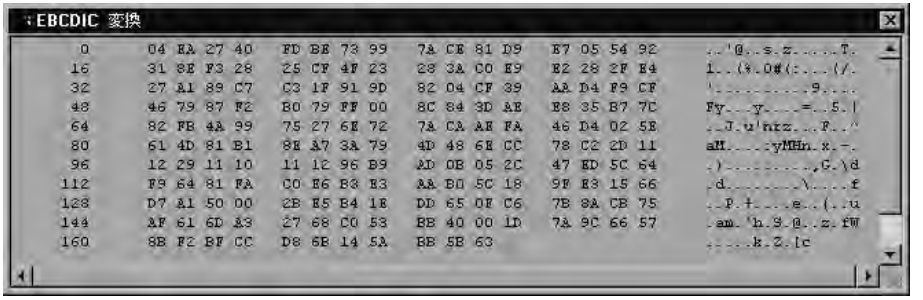
F7 キーを押すと表示されるビューア・ユーティリティは，特にパラメータ化を行う際に役立ちます。このユーティリティを使うことで，パラメータに保存するデータのオフセットを決定できます。

特定の文字のオフセットを決定するには，次の手順を実行します。

1 **data.ws** を表示して，バッファの最初からデータを選択します。



2 F7 キーを押してデータと文字のオフセットを表示します。4 文字以上選択すると，データが長い形式で表示されます。



3 ASCII データの中で相関させる値を検索します。この例では，31 番目の文字で始まり，最後の文字が 2 行目にある 13546 番（UNIX セッション中のプロセス ID）を相関させます。

4 **lrs_save_param_ex** 関数のオフセット値を使って，プロセス ID の値を相関させます。詳細については，『**第 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「ステートメントの相関」を参照してください。

表示形式の設定

VuGen のビューア・ウィンドウ（F7 キー）でのバッファ・データの表示方法を指定できます。**＜LoarRunner ディレクトリ＞/dat** ディレクトリの **conv_frm.dat** ファイルには，次の表示パラメータが含まれています。

LongBufferFormat：5 文字以上を表示するときに使われる形式です。オフセットには nn，16 進データには XX，ASCII データには aa を使います。

LongBufferHeader：長いバッファ形式で表示する場合に，各バッファの先頭に表示するヘッダーです。

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。

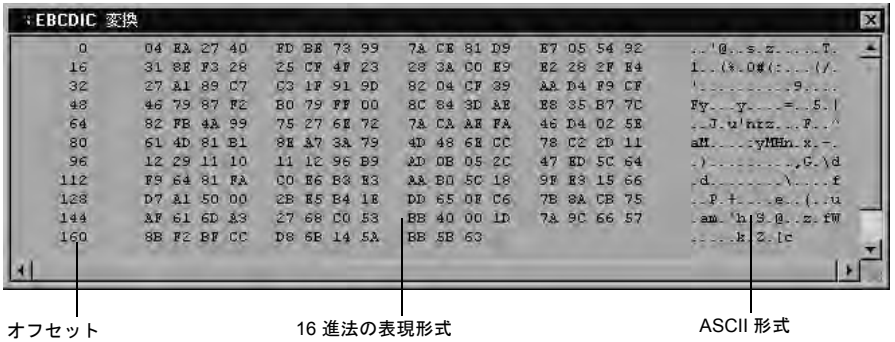
NonPrintableChar：非印字 ASCII 文字を表す文字。

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。

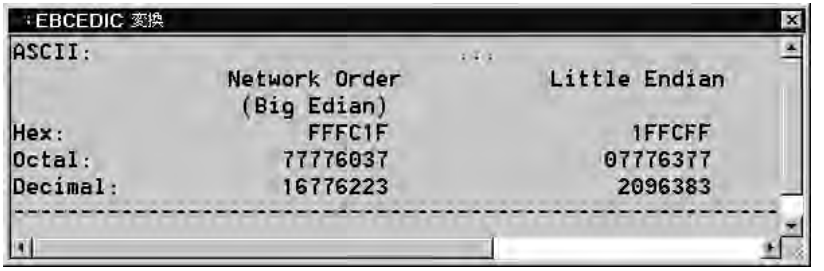
標準設定では，長い形式と短い形式が設定され，非印字文字はドットとして出力するように指定されています。

```
[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。例えば，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・スクリプトでよくある問題の 1 つに，ポート番号が動的に割り当てられる「動的ポート」があります。特定のアプリケーションが常に同じポートを使用していると，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。次の例では，**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方法については，「**オンライン関数リファレンス**」を参照してください。

WinSock 仮想ユーザのステートメントを相関させるには，次の手順を実行します。

1 スクリプト内の，バッファの内容を保存する場所に **lrs_save_param_ex** ステートメントを挿入します。ユーザ・バッファ，静的バッファ，または受信バッファを保存できます。

```
lrs_save_param_ex (socket, type, buffer, offset, length, encoding, parameter);
```

2 パラメータを参照します。

VuGen のメイン・ウィンドウの［データ ファイル］ボックスで **data.ws** ファイルを選択して，バッファの内容を表示します。保存したバッファの内容で置換するデータを探します。置換する値のすべてのインスタンスを山括弧（< >）で囲まれたパラメータ名で置き換えます。

次の例では，ユーザが telnet セッションを実行しています。ユーザは **ps** コマンドを使ってプロセス ID（PID）を調べ，その PID を使ってアプリケーションを強制終了しています。

```
frodo:/u/jay>ps
  PID TTY      TIME CMD
14602 pts/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。

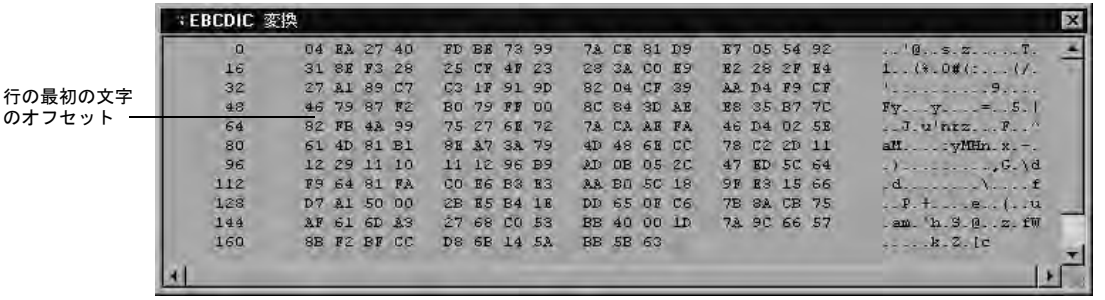
3 **data.ws** ファイル内で，データを受け取ったバッファを調べます（この例では buf47）。

```
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，159 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 5 部

ユーザ定義の仮想ユーザ・スクリプト

第 13 章

ユーザ定義の仮想ユーザ・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。仮想ユーザ 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，次の項目について説明します。

- ユーザ定義の仮想ユーザ・スクリプトの作成について
- C 仮想ユーザ
- C 仮想ユーザ・スクリプトのワークフロー・ウィザード
- 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・スクリプトの作成について

VuGen では，実際のセッションを記録せずに，独自の関数をスクリプトにプログラミングできます。仮想ユーザ API または標準のプログラミング関数を使用できます。仮想ユーザ API 関数では，仮想ユーザについての情報を収集できます。例えば，仮想ユーザ関数を使って，サーバ・パフォーマンスの測定，サーバ負荷の制御，デバッグ・コードの追加，テストに含まれる仮想ユーザまたは監視対象の仮想ユーザについてのランタイム情報の取得などができます。

本章では，対象アプリケーションのライブラリやクラスと連携して動作する仮想ユーザ・スクリプトを VuGen エディタでプログラミングする方法について説明します。

また，Visual C および Visual Basic 環境からプログラミングを行って仮想ユーザ・スクリプトを作成することもできます。これらの環境では，開発アプリケーションに 仮想ユーザ API 関数のライブラリをインポートして仮想ユーザ・スクリプトを作成します。詳細については，『**第 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「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。

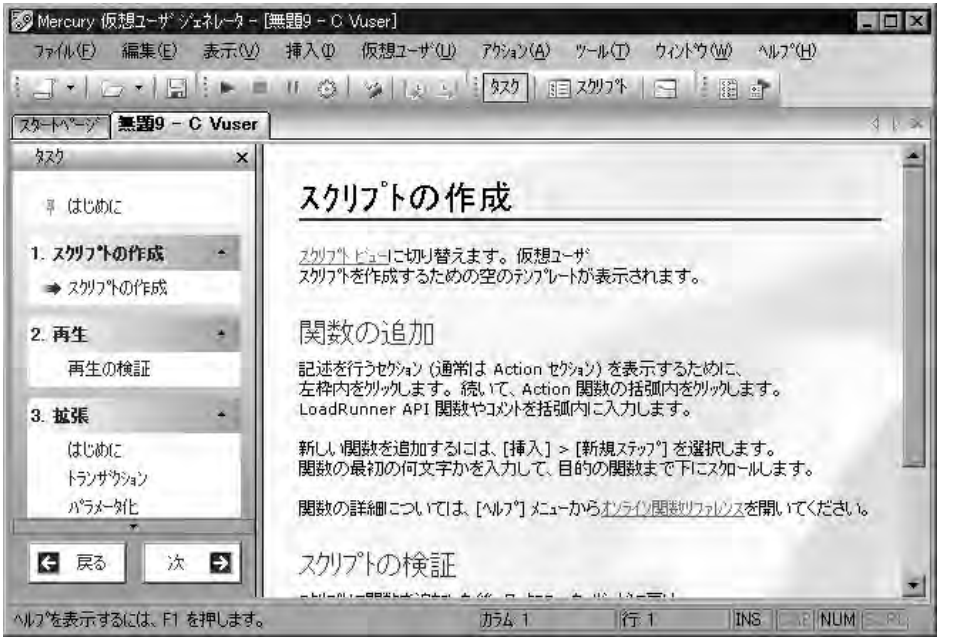
```
Action1()
{

    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 仮想ユーザ・スクリプトのワークフロー・ウィザード

ワークフロー・ウィザードは，スクリプトの作成の手順を導いてくれます。［タスク］表示枠のリンクをクリックすると，スクリプト作成の手順についての説明を読むことができ，再生に関する情報を表示できます。［**戻る**］および［**次**］ボタンを使用して，画面間を移動できます。



ワークフロー・ウィザードが見えない場合は，［タスク］表示枠が開いていることを確認してください（［タスク］表示枠は，ツールバーの［**タスク**］ボタンを使用して表示 / 非表示を切り替えます。）次に最初のリンクである［**はじめに**］をクリックします。

ウィザードの詳細については，第 3 章「ワークフロー・ウィザードの使用」を参照してください。

スクリプトの作成

［スクリプトの作成］ウィンドウには，Web Services スクリプトの作成のいくつかのガイドラインが含まれます。

➤ ［**関数の追加**］：関数の入力方法と入力場所を示します。

➤［**スクリプトの検証**］：関数の追加後のスクリプトの検証方法を示します。

C 言語の関数の使用についてのガイドライン

制御フローや構文など，標準 ANSI-C の規約はすべて，C 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 の規則に従って変数を宣言して定義できます。

仮想ユーザ・スクリプトの実行に使用される C インタプリタは，標準の ANSI C 言語を受け付けます。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。
- 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。自分自身のプロトタイプを書くか，Mercury 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。次に例を示します。

```
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。*/
```

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。

Corba-Java および RMI-Java 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 VuGen に取り入れることができます。空の VB 仮想ユーザ・スクリプトを作成するには，［新規仮想ユーザ］ダイアログ・ボックスの［**ユーザ定義**］カテゴリから［**VB 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，仮想ユーザと VB アプリケーションのオブジェクトおよび変数グローバル宣言が含まれる **global.vba** ファイルです。

VBScript 仮想ユーザ

空の VBScript 仮想ユーザ・スクリプトを作成して，VBScript コードを配置できます。このスクリプト・タイプでは，VBScript アプリケーションを VuGen に取り入れることができます。空の VBScript 仮想ユーザ・スクリプトを作成するには，［新規仮想ユーザ］ダイアログ・ボックスの［**ユーザ定義**］カテゴリから［**VB Script 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，仮想ユーザ API 関数と VB スクリプトのオブジェクトを作成する **global.vbs** ファイルです。例えば，次のコードは標準のオブジェクト **lr** を作成します。

```
Set lr = CreateObject("LoadRunner.LrApi")
```

JavaScript 仮想ユーザ

空の JavaScript 仮想ユーザ・スクリプトを作成して，JavaScript コードを配置できます。このスクリプト・タイプでは，既存の javascript アプリケーションを VuGen に取り入れることができます。空の JavaScript 仮想ユーザ・スクリプトを作成するには，［新規仮想ユーザ］ダイアログ・ボックスの［**ユーザ定義**］カテゴリから［**Javascript 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，仮想ユーザ API 関数と Javascript のオブジェクトを作成する **global.js** ファイルです。例えば，次のコードは標準のオブジェクト **lr** を作成します。

```
var lr = new ActiveXObject("LoadRunner.LrApi")
```

第 14 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・スクリプトのプログラミングについて
- Java 仮想ユーザの作成
- Java 仮想ユーザ・スクリプトの編集
- Java 仮想ユーザ API 関数
- Java 仮想ユーザ関数を使った作業
- 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
- プログラミングに関するヒント

以降の情報は，Java，EJB テスト，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。例えば，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，次の各項を参照してください。

Java 互換の仮想ユーザ・スクリプトを作成する第一歩は，**Java 仮想ユーザ・**タイプの新しい仮想ユーザ・スクリプトのテンプレートを作成することです。次に，任意の Java コードをスクリプト・テンプレートにプログラミングするか，貼り付けます。Java 仮想ユーザ関数を追加して，スクリプトを拡張したり，反復時にさまざまな値を使用できるように引数をパラメータ化したりできます。

Java 仮想ユーザ・スクリプトは，スケーラブルなマルチ・スレッド・アプリケーションとして稼動します。スクリプトにユーザ定義のクラスを含める場合は，コードをスレッドセーフにする必要があります。コードをスレッドセーフにしないと，結果が不正確になることがあります。スレッドセーフでないコードの場合は，Java 仮想ユーザをプロセスとして実行します。つまり，プロセスごとに個別の Java 仮想マシンを作成します。その結果，スクリプトの拡張性は低くなります。

スクリプトを作成したら，VuGen でスタンドアロン・テストとして実行します。Java コンパイラ（Sun の javac）によってスクリプトに誤りがないか調べられた後，コンパイルが実行されます。

スクリプトを作成した後，スクリプトを自分の環境（LoadRunner シナリオ，Performance Center 負荷テスト，Tuning Module セッション，または Business Process Monitor プロファイル）に統合します。詳細については，『**Mercury LoadRunner コントローラ・ユーザーズ・ガイド**』，**Tuning Console** のマニュアル，**Performance Center** のマニュアル，または **Application Management** のマニュアルを参照してください。

Java 仮想ユーザの作成

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・テンプレートを作成することです。

Java 仮想ユーザ・スクリプトを作成するには，次の手順を実行します。

1 VuGen を開きます。

2 ［**ファイル**］＞［**新規作成**］を選択するか，［**新規作成**］ボタンをクリックします。［新規仮想ユーザ］ダイアログ・ボックスが開きます。

3 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・スクリプトを使用する場合は，すべてのコードを **Actions** クラスに配置します。Actions クラスを表示するには，左側の表示枠の［**Actions**］をクリックします。その内容が右側の表示枠に表示されます。

```
import lrapi.*;
public class Actions
{
	public int init() {
		return 0;
	}

	public int action() {
		return 0;
	}

	public int end() {
		return 0;
	}
}
```

Actions クラスには，**init**，**action**，**end** の 3 つのメソッドが含まれています。次の表に，各メソッドに何を含める必要があり，各メソッドがどのタイミングで呼び出されるかを示します。

スクリプトのメソッド	エミュレーション内容	実行のタイミング
init	サーバへのログイン	仮想ユーザを初期化（ロード）するとき
action	クライアントの動作	仮想ユーザが「実行」状態のとき
end	ログオフ処理	仮想ユーザを終了または停止するとき

Init メソッド

すべてのログイン手続きと一度だけ行う設定を **init** メソッドに置きます。**init** メソッドは，仮想ユーザがスクリプトの実行を開始するときに 1 回だけ実行されます。次のサンプル **init** メソッドではアプレットを初期化しています。

```
import org.omg.CORBA.*;
import org.omg.CORBA.ORB.*;
import lrapi.lr;

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

        // Orb インスタンスを初期化する ..
        MApplet mapplet = new MApplet("http://chaos/classes/", null);
        orb = org.omg.CORBA.ORB.init(mapplet, null);
...
```

action メソッド

仮想ユーザで行うすべての操作を **action** メソッドに配置します。**action** メソッドは，実行環境の設定で指定した反復回数に従って実行されます。反復の設定の詳細については，『**第 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「実行環境の設定」を参照してください。次のサンプル **action** メソッドでは，仮想ユーザ ID を取り出して出力しています。

```
public int action() {
     lr.message("vuser:" + lr.get_vuser_id() + " xxx");
          return 0;
 }
```

end メソッド

end メソッドには，サーバからのログオフや環境の後始末など，スクリプトの終了時に仮想ユーザに実行させるコードを配置します。

end メソッドは，仮想ユーザがスクリプトの実行を終了するときに 1 回だけ実行されます。次の例に示す **end** メソッドでは「**End**」というメッセージを実行ログに出力しています。

```
public int end() {
      lr.message("End");
      return 0;
  }
```

Java 仮想ユーザ API 関数

VuGen には，Java 仮想ユーザ・スクリプト専用の Java API があります。これらの関数はすべて lrapi.lr クラスの静的メソッドです。個々の関数の詳細については，「**オンライン関数リファレンス**」（［**ヘルプ**］ ＞ ［**関数リファレンス**］）を参照してください。Java 仮想ユーザ・スクリプトを新規作成すると，**import lrapi.*** がスクリプトに自動的に挿入されます。

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始まりを示します。
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・ラインで使われている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 コントローラまたは Administration Console を実行しているマシンの名前を返します。
lr.get_object	Java オブジェクトを取り込み，データ・ファイルにダンプします（Corba-Java のみ）。

lr.get_scenario_id	現在のシナリオまたはセッション・ステップの 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	パラメータをその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	場所の詳細情報を含むエラー・メッセージを仮想ユーザ・ログ・ファイルと［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	仮想ユーザ・ログ・ファイルにメッセージを送信します。
lr.message	メッセージを［出力］ウィンドウに送ります。

lr.output_message	場所の情報を含むメッセージをログ・ファイルと［出力］ウィンドウに送ります。
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またはコンソールのウィンドウの仮想ユーザ・ステータス領域に送ります（LoadRunner のみ）。

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，次の例に示すようにそれらをスクリプトの先頭でインポートします。

インポートするクラスのディレクトリや関連 jar ファイルをクラスパスに忘れずに追加します。また，追加するクラスがスレッドセーフかつスケーラブルであることを確認します。

```
import java.io.*;
import lrapi.*;

public class Actions
{
...
}
```

Java 仮想ユーザ関数を使った作業

Java 仮想ユーザ関数を使って次の作業を行うことにより，スクリプトを拡張できます。

- トランザクションの挿入
- ランデブー・ポイントの挿入
- 仮想ユーザ情報の取得
- 出力メッセージの発行
- ユーザの思考遅延時間のエミュレート
- コマンド・ライン引数の処理

トランザクションの挿入

サーバのパフォーマンスを測定するには，トランザクションを定義します。各トランザクションは，仮想ユーザからの特定の要求に対するサーバの応答時間を測定します。これらの要求は，単純な場合と複雑な場合があります。LoadRunner を使用している場合は，シナリオの実行中および実行後に，オンライン・モニタとグラフを使ってトランザクションごとのパフォーマンスを分析できます。

またトランザクションのステータスとして，lr.PASS および lr.FAIL を指定することもできます。トランザクションが正常終了したかどうか Vuser に判定させることができます。また，条件ループを使って自分で判定することもできます。例えば，コードの中で，リターン・コードが特定の値になっているかどうかを調べることができます。リターン・コードが正しい値であれば，lr.PASS ステータスを発行します。リターン・コードの値が正しくなければ，lr.FAIL ステータスを発行します。

トランザクションの開始と終了を示すには，次の手順を実行します。

1 **lr.start_transaction** 関数は，スクリプト内で処理の実行時間の計測を開始する場所に挿入します。

2 **lr.end_transaction** 関数は，スクリプト内で処理の実行時間の計測を終了する場所に挿入します。**lr.start_transaction** 関数で指定したトランザクション名を指定します。

3 トランザクションのステータス（lr.PASS または lr.FAIL）を指定します。

```
public int action() {

for(int i=0;i<10;i++)
  {
     lr.message("action()"+i);
     lr.start_transaction("trans1");
     lr.think_time(2);
     lr.end_transaction("trans1",lr.PASS);
  }
     return 0;
}
```

ランデブー・ポイントの挿入

次の項は，Application Management には適用されません。

クライアント / サーバ・システムを対象に多数のユーザによる負荷をエミュレートするには，「**ランデブー・ポイント**」を作成して，多数の仮想ユーザが同時にタスクを実行するように同期させなければなりません。ある仮想ユーザがランデブー・ポイントに到達すると，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は待機させられます。

仮想ユーザ・スクリプトにランデブー関数を挿入することによって，この「待ち合わせ場所」を指定します。

ランデブー・ポイントを挿入するには，次の手順を実行します。

➤ 仮想ユーザを待機させる場所に **lr.rendezvous** 関数を挿入します。

```
public int action() {

  for(int i=0;i<10;i++)
  {
     lr.rendezvous("rendz1");
     lr.message("action()"+i);
     lr.think_time(2);
  }
  return 0;
}
```

仮想ユーザ情報の取得

次の関数を仮想ユーザ・スクリプトに追加して，仮想ユーザ情報を取得できます。

lr.get_attrib_string	仮想ユーザ ID やロード・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，Administration Console，または Tuning Module Console を実行しているマシンの名前を返します。
lr.get_scenario_id	現在のシナリオまたはセッション・ステップの ID を返します（LoadRunner のみ）。
lr.get_vuser_id	現在の仮想ユーザの ID を返します（LoadRunner のみ）。

次の例では，**lr_get_host_name** 関数を使って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

シナリオまたはセッション・ステップを実行しているときは，コントローラまたはコンソールの［出力］ウィンドウに，スクリプトの実行に関するメッセージが表示されます。エラーおよび通知メッセージをコントローラまたはコンソール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，コントローラまたはコンソールの［出力］ウィンドウに表示されます。例えば，クライアント・アプリケーションの現在の状態を表示するメッセージを挿入できます。また，これらのメッセージをファイルに保存することもできます。

注：トランザクション内からメッセージを送ってはなりません。トランザクション内からメッセージを送ると，トランザクションの実行時間が長くなり，実際のトランザクションの結果に偏りが生じる可能性があります。

仮想ユーザ・スクリプトの中で，次のメッセージ関数を使用できます。

lr.debug_message	デバッグ・メッセージを［出力］ウィンドウに送ります。
lr.log_message	仮想ユーザ・ログ・ファイルにメッセージを送信します。
lr.message	メッセージを［出力］ウィンドウに送ります。
lr.output_message	場所の情報を含むメッセージをログ・ファイルと［出力］ウィンドウに送ります。

次の例では，**lr.message** を使用してループ回数を示すメッセージを［出力］ウィンドウに送っています。

```
for(int i=0;i<10;i++)
  {
     lr.message("action()"+i);
     lr.think_time(2);
  }
```

メッセージ関数の詳細については，189 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。

仮想ユーザが Java の標準出力ストリームと標準エラー・ストリームを VuGen の実行ログにリダイレクトするように指定できます。これは，仮想ユーザ・スクリプトに既存の Java コードを貼り付けるときや，**System.out** と **System.err** の呼び出しが含まれる既成のクラスを使用するときに，特に役に立ちます。実行ログでは，標準出力メッセージは青，標準エラーは赤で示されます。

lr.enable_redirection を使って，特定のメッセージを標準出力および標準エラーへリダイレクトする方法を次の例に示します。

```
lr.enable_redirection(true);

System.out.println("This is an informatory message..."); // リダイレクトされる
System.err.println("This is an error message..."); // リダイレクトされる

lr.enable_redirection(false);

System.out.println("This is an informatory message..."); // リダイレクトされない
System.err.println("This is an error message..."); // リダイレクトされない
```

注：lr.enable_redirection 関数を **true** に設定すると，この設定が既存のすべてのリダイレクト設定に優先します。以前のリダイレクト設定を復元するには，この関数を **false** に設定します。

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

ユーザの思考遅延時間のエミュレート

連続する操作の合間のユーザの待ち時間を「**思考遅延時間**」といいます。仮想ユーザは，**lr_think_time** 関数を使ってユーザの思考遅延時間をエミュレートします。次の例では，仮想ユーザがループの中で 2 秒待機しています。

```
for(int i=0;i<10;i++)
  {
     lr.message("action()"+i);
     lr.think_time(2);
  }
```

思考遅延時間の設定では，スクリプト中の値をそのまま使うことも，それらの値の倍数を使うこともできます。仮想ユーザによる思考遅延時間関数の処理方法を設定するには，［実行環境設定］ダイアログ・ボックスを開きます。詳細については，『**第 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「実行環境の設定」を参照してください。

lr.think_time 関数の詳細については，「**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。コントローラ，Tuning Module，または Administration Console で，スクリプトのパスとファイル名に続けてコマンド・ライン・オプションを挿入します。コマンド・ライン引数を読み取って，値を仮想ユーザ・スクリプトに渡すことができる関数として，次の 3 つがあります。

lr.get_attrib_double	double float 型の引数を取得します。
lr.get_attrib_long	long int 型の引数を取得します。
lr.get_attrib_string	文字列を取得します。

コマンド・ラインの形式は次の 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・オプションの挿入方法の詳細については，**LoadRunner コントローラ**のマニュアル，**Tuning Module Console** のマニュアル，**Performance Center** のマニュアル，または **Application Management** のマニュアルを参照してください。

Java 環境の設定

Java 仮想ユーザ・スクリプトを実行する前に，仮想ユーザを実行するすべてのマシンで環境変数 PATH と CLASSPATH が正しく設定されていることを確認してください。

- スクリプトをコンパイルして再生するためには，JDK 1.1，1.2，または 1.3 のコンポーネントの完全インストールを実行しておく必要があります。JRE をインストールするだけでは不十分です。1 つのマシンに複数の JDK または JRE がインストールされているのは望ましくありません。可能ならば，不要なバージョンはすべてアンインストールします。
- 環境変数 PATH には，JDK\bin のパスがなければなりません。
- JDK 1.1.x の場合は，環境変数 CLASSPATH に **classes.zip** のパス（**JDK/lib** サブディレクトリ）およびすべての Mercury クラス（**classes** サブディレクトリ）が含まれていなければなりません。
- Java 仮想ユーザによって使用されるすべてのクラスが，マシンの CLASSPATH 環境変数，または，実行環境設定の［**Classpath**］パネルで設定されている **Classpath Entries** に含まれている必要があります。

Java 仮想ユーザ・スクリプトの実行

Java 仮想ユーザ・スクリプトは，コンパイル後に実行される点が C 仮想ユーザ・スクリプトとは異なります。C 仮想ユーザ・スクリプトは，インタプリタによって解釈されます。VuGen ではインストールされている JDK から **javac** コンパイラが探し出され，スクリプト内の Java コードがコンパイルされます。このとき，VuGen ウィンドウの最下部に「**コンパイルしています ...**」というステータス・メッセージが表示されます。コンパイル中に発生したエラーは，実行ログに表示されます。スクリプトの中でエラーが発生したコードの位置に移動するには，エラーの行番号が示されているエラー・メッセージをダブルクリックします。エラーを修正し，スクリプトを再実行します。

コンパイルが完了すると，ステータス・メッセージが「**コンパイルしています ...**」から「**実行しています ...**」に変わり，スクリプトの実行が始まります。スクリプトに変更を加えていなければ，次にスクリプトを実行したときに，VuGen によるコンパイルが行われずにスクリプトが実行されます。スクリプトをさらに詳細にデバッグする場合は，ステップ実行オプションを使って，ブレークポイントを設定したり，表示しながらの実行を指定したりできます。

注：スクリプト内から JNDI の拡張機能を呼び出している場合には，仮想ユーザを**スレッド**として実行しようとするときに問題が生じることがあります。これは，JNDI ではスレッドごとに独自のコンテキスト・クラス・ローダが必要とされているために発生します。スレッドとして実行するには，次の行を **init** セクションの先頭に追加して，仮想ユーザが実行時に固有のコンテキスト・クラス・ローダを使用するように指定します。

```
DummyClassLoader.setContextClassLoader();
```

パッケージの一部としてのスクリプトのコンパイルと実行

Java 仮想ユーザ・スクリプトを作成するときに，クラスまたはメソッドが保護されている（protected）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。次の例では，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
{
    :
}
```

上記の例では，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，関連パッケージを使ったコンパイルが可能になります。package ステートメントは，Java の場合と同様にスクリプトの（コメントを除く）1 行目になければなりません。

プログラミングに関するヒント

Java 仮想ユーザ・スクリプトのプログラミングを行うときは，既成のコードのセグメントをスクリプトに貼り付けるか，既成のクラスをインポートして，そのメソッドを呼び出せるようにします。スケーラビリティを考慮して仮想ユーザをコントローラまたはコンソールの管理下でスレッドとして実行する必要がある場合は，インポートするコードがすべてスレッドセーフであることを確認する必要があります。

多くの場合，コードがスレッドセーフであることを検出するのは困難です。VuGen，コントローラ，およびコンソールでは，仮想ユーザの数が限られていれば，Java 仮想ユーザは問題なく実行されるかもしれません。しかし，ユーザ数が増えると問題が発生します。スレッドセーフでないコードは，通常は静的クラス・メンバを使用していることに起因します。次に例を示します。

```
import lrapi.*;
public class Actions
{
  private static int iteration_counter = 0;

    public int init() {
        return 0;
    }

    public int action() {
        iteration_counter++;
        return 0;
    }

    public int end() {
        lr.message("Number of Vuser iterations:"+iteration_counter);
        return 0;
    }
}
```

1 個の仮想ユーザを実行すると，**iteration_counter** メンバは実行された反復の回数を正確に示します。1 台の仮想マシンで複数の仮想ユーザを複数のスレッドとして実行すると，静的クラス・メンバの **iteration_counter** がすべてのスレッドで共有されるため，反復回数のカウントが不正確になります。これは，すべての仮想ユーザの反復回数の合計がカウントされるからです。

スレッドセーフでないことが分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 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「実行環境の設定」を参照してください。

基本的な Java 仮想ユーザ・スクリプトを実行する場合，スクリプトは通常 1 個のスレッド（メイン・スレッド）で構成されています。Java 仮想ユーザ API にアクセスできるのは，メイン・スレッドだけです。Java 仮想ユーザが 2 次的なワーカー・スレッドを生成したときに Java API を使用すると，予期しない結果が生じます。したがって，Java 仮想ユーザ 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** 関数にも影響を及ぼします。

次の例で，LR API を使用してもよい場所と使用してはいけない場所を示します。実行ログの最初のログ・メッセージは，**flag** の値が false であることを示します。その後，仮想マシンによって新規スレッド **set_thread** が生成されます。このスレッドは実行され，**flag** が true に設定されますが，**lr.message** への呼び出しにもかかわらず，ログにはメッセージが発行されません。最後のログ・メッセージは，スレッド内のコードが実行され，**flag** が true に設定されたことを示します。

```
boolean flag = false;

public int action() {
  lr.message("Flag value:"+flag);
  Thread set_thread = new Thread(new Runnable();{
          public void run() {
            lr.message("LR-API NOT working!");
            try { Thread.sleep(1000); } catch(Exception e) {}
            flag = true;
          }
  });
  set_thread.start();
  try { Thread.sleep(3000); } catch(Exception e) {}
  lr.message("Flag value:"+flag);
  return 0;
}
```

第 6 部

分散コンポーネント・プロトコル

第 15 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 1 巻 - 仮想ユーザ・ジェネレータの使い方』**の「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 16 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 仮想ユーザ・スクリプトの記録について
- 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 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「スクリプト生成オプションの設定」を参照してください。

COM 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 呼び出しを追跡するには，これらのインタフェースを理解しておく必要があります。COM インタフェースのメソッドおよびパラメータにアクセスするための参照として使用されるタイプ・ライブラリには，COM オブジェクトとインタフェース

に関する記述が含まれています。各 COM クラス，インタフェース，タイプ・ライブラリは，GUID（Global Unique Identifier）によって識別されます。

COM インタフェース

COM インタフェースは，相互に関連するメソッドの集合を提供します。例えば，**Clock** オブジェクトには，**Clock**，**Alarm**，および **Timer** 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。例えば，**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。例えば，**Alarm Status** プロパティを **On** に設定した場合の結果は，**AlarmOn** メソッドを呼び出した場合の結果と同じです。

COM オブジェクトは，多数のインタフェースをサポートすることができます。コンポーネントには必ず **IUnknown** インタフェースが実装されており，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，スクリプト言語による COM オートメーションの実装を可能にします。

COM のクラスのコンテキストと場所の透過性

COM オブジェクトは，クライアント・アプリケーションを実行しているマシン，またはリモート・サーバ上で実行できます。アプリケーションによって作成される COM オブジェクトは，ローカル・ライブラリ，ローカル・プロセス，リモート・マシン（「リモート・オブジェクト・プロキシ」）のいずれかにあります。「コンテキスト」と呼ばれる COM オブジェクトのありかは，アプリケーションにとっては透過的です。大半のユーザは，仮想ユーザを使ってリモート・サーバの負荷を検査します。このため，リモート・オブジェクト・プロキシがアクセスするオブジェクトが，通常，負荷テストの対象として最も意味があります。

COM のデータ型

COM は，セーフ配列，BSTR 文字列およびバリアントなど，いくつかの特殊なデータ型も提供します。これらのデータ型は，デバッグやパラメータ化のような作業で使用する必要があるかもしれません。

COM 仮想ユーザを使った作業の開始

本項では，COM 仮想ユーザ・スクリプトの作成するプロセスについて説明します。

COM 仮想ユーザ・スクリプトを作成するには，次の手順を実行します。

1 **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。仮想ユーザのタイプとして COM を指定します。記録対象のアプリケーションを選び，記録オプションを設定します。スクリプト記録オプションの設定については，『**第 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「スクリプト生成オプションの設定」を参照してください。COM 固有のオプションとフィルタの設定方法については，212 ページ「COM 記録オプションの設定」を参照してください。アプリケーションを使って，一般的な操作を記録します。

記録方法の詳細については，『**第 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「VuGen を使った記録」を参照してください。

2 **オブジェクト・フィルタを適切に設定し直します。**

生成されたログ・ファイルを使って，フィルタで記録対象のオブジェクトを選択し直します。詳細については，「記録対象の COM オブジェクトの選択」の項を参照してください。

3 **スクリプトを拡張します。**

トランザクション，ランデブー・ポイント，および制御フロー構造をスクリプトに挿入することによって，仮想ユーザ・スクリプトを拡張します。

詳細については，『**第 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「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の代わりにパラメータを定義します。固定値をパラメータで置き換えることにより，同じビジネス・プロセスを，異なる値で何度も繰り返すことができます。

詳細については，『**第 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「VuGen パラメータを使った作業」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 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「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 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「スタンドアロン・モードでの仮想ユーザ・スクリプトの実行」を参照してください。

スクリプトを作成した後，スクリプトを自分の環境（LoadRunner シナリオ，Performance Center 負荷テスト，Tuning Module セッション，または Business Process Monitor プロファイル）に統合します。詳細については，『**Mercury LoadRunner コントローラ・ユーザーズ・ガイド**』，**Tuning Console** のマニュアル，**Performance Center** のマニュアル，または **Application Management** のマニュアルを参照してください。

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，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。

例えば，以下は Visual Basic ライブラリのローカルの COM の例です。

```
Class JetES {039EA4C0-E696-11D0-878A-00A0C91EC756}
was loaded from type library "JET Expression Service Type Library"
({2358C810-62BA-11D1-B3DB-00600832C573} ver 4.0)
```

これは，サーバ上で負荷を生成しないため，追加してはなりません。

同様に，次に示す OLE DB および Microsoft Windows Common Controls はローカル・オブジェクトなので，そのクラスとライブラリは，サーバへの負荷を生成しません。したがって，これらも記録する必要はありません。

```
Class DataLinks {2206CDB2-19C1-11D1-89E0-00C04FD7A829} was
loaded from type library "Microsoft OLE DB Service Component 1.0 Type
Library"({2206CEB0-19C1-11D1-89E0-00C04FD7A829} ver 1.0)

Class DataObject {2334D2B2-713E-11CF-8AE5-00AA00C00905}
was loaded from type library "Microsoft Windows Common Controls 6.0
(SP3)"
({831FDD16-0C5C-11D2-A9FC-0000F8754DA1} ver 2.0)
```

ただし，例えば，次の一覧は，サーバで負荷を生成するため記録する必要があるクラスを示します。

```
Class Order {B4CC7A90-1067-11D4-9939-00105ACECF9A}
was loaded from type library "FRS"
({B4CC7A8C-1067-11D4-9939-00105ACECF9A} ver 1.0)
```

例えば 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。例えば，アプリケー

ションが **FRS** クラス関数を呼び出すたびに，**FRS** ライブラリは **ADO（ActiveX Data Object）**ライブラリを呼び出します。このような呼び出しの連鎖に含まれるいくつかの関数がフィルタに表示されている場合，VuGen によって連鎖を開始する最初の呼び出しだけが記録されます。**FRS** および **ADO** 呼び出しの両方を選択した場合は，**FRS** 呼び出しだけが記録されます。

また，フィルタで **ADO** ライブラリだけを選択した場合は，**ADO** ライブラリへの呼び出しが記録されます。多くの場合，連鎖における最初のリモート・オブジェクトへの呼び出しを記録するのが簡単です。ただし，場合によっては，アプリケーションが複数の異なる COM オブジェクトのメソッドを使用します。それらのメソッドがいずれもサーバに負荷をかける単独のオブジェクトを使用する場合は，最終的な共通オブジェクトだけを記録することもできます。

選択可能なオブジェクト

VuGen では，オブジェクトのタイプ・ライブラリを読み込むことができれば，そのオブジェクトを記録できます。タイプ・ライブラリがシステムにインストールされていない場合や VuGen でタイプ・ライブラリが見つからない場合には，対応する COM オブジェクトは［記録オプション］ダイアログ・ボックスに表示されません。これらの COM オブジェクトがアプリケーションによって使用されている場合，VuGen ではこれらのオブジェクトを識別できず，ファイル内に **INoTypeInfo** として示されます。

除外できるインタフェース

［記録オプション］ダイアログ・ボックスでは，タイプ・ライブラリのリストに含まれているすべてのインタフェースが，オブジェクトごとに表示されます。このダイアログ・ボックスで，各インタフェースを記録に含めるか除外するかを指定できます。ただし，**ADO**，**RDS**，および **Remote の各オブジェクト**は，フィルタに 1 つのグループとして含めることができます。その場合，フィルタには，これらの環境またはインタフェースの個々のオブジェクトまたはインタフェースは表示されません。また，タイプ・ライブラリから記録対象にインクルードしたオブジェクトのインタフェースが，タイプ・ライブラリにないために［記録オプション］ダイアログ・ボックスに表示されない場合があります。その場合は，VuGen のスクリプトを生成した後で，スクリプトに含まれるこれらのインタフェースを特定して，VuGen によって生成された interfaces.h ファイルでそれらのインタフェースの GUID 番号を確認します。この情報を使って，以下に説明する方法で，インタフェースを除外できます。

COM 記録オプションの設定

COM の［記録オプション］ダイアログ・ボックスを使って，フィルタ・オプションと COM のスクリプト編集オプションを設定します。レジストリ，ファイル・システム，Microsoft トランザクション・サーバ（MTS）のタイプ・ライブラリを探すには，オンライン・ブラウザを使用します。

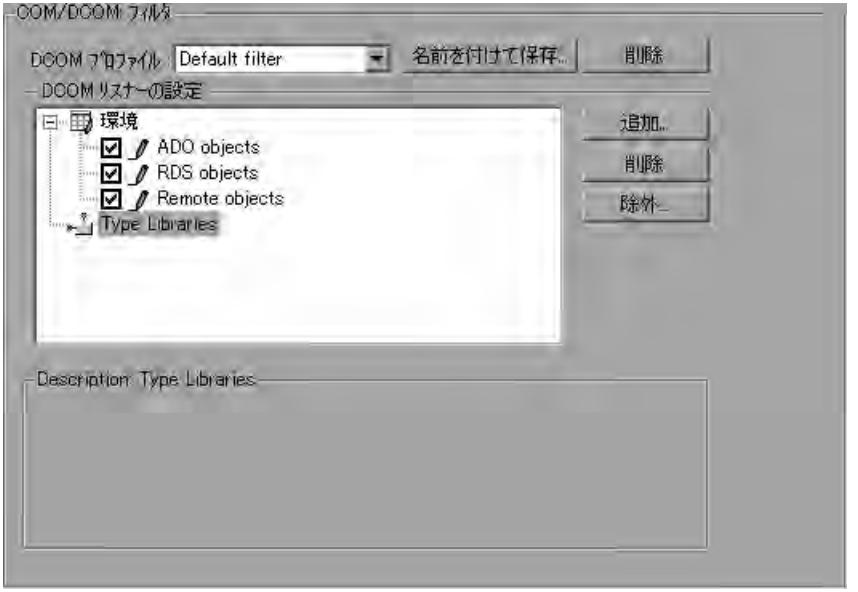
詳細については，次を参照してください。

- ➤ オブジェクトのフィルタリング
- ➤ フィルタの設定
- ➤ COM スクリプト編集オプションの設定

オブジェクトのフィルタリング

フィルタ・オプションを使って，VuGen で記録する COM オブジェクトを指定できます。オブジェクトは環境およびライブラリの中から選択できます。

フィルタ・オプションで，標準フィルタの設定または代替フィルタの作成を行います。環境およびタイプ・ライブラリを指定することで，記録セッションを絞り込めます。



DCOM プロファイル

➤ ［**Default Filter**］: COM 仮想ユーザ・スクリプトを記録するときに標準で使用されるフィルタ。

➤ ［**新規フィルタ**］: 標準の環境設定に基づく新規のフィルタ。このフィルタの設定を使って記録するには，あらかじめフィルタに名前を付ける必要があります。

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**］: タイプ・ライブラリ。記録する COM オブジェクトを表す **.tlb** または **.dll** ファイルです。すべての COM オブジェクトには，オブジェクトを表すタイプ・ライブラリがあります。タイプ・ライブラリは，レジストリ，Microsoft Transaction Server，またはファイル・システムから選択できます。

［**Type Libraries**］: ダイアログ・ボックスの下側に，各タイプ・ライブラリの次の情報が表示されます。

➤ ［**TypLib**］: タイプ・ライブラリ（tlb ファイル）の名前。

➤［**Path**］: タイプ・ライブラリのパス。

➤［**Guid**］: タイプ・ライブラリの GUID（Global Unique Identifier）。

フィルタの設定

この項では，フィルタの設定方法について説明します。

記録対象の COM オブジェクトを選択するには，次の手順を実行します。

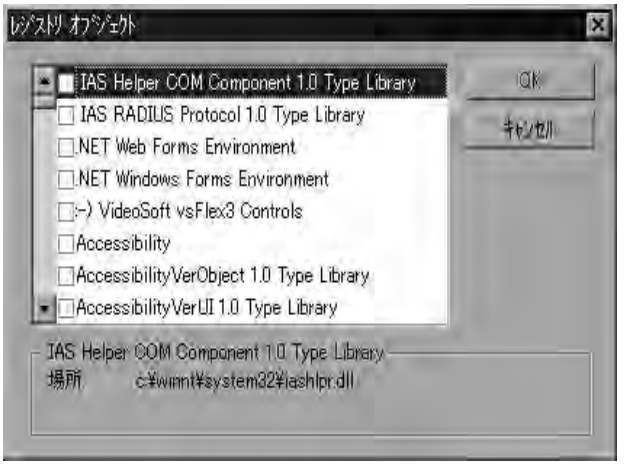
1 メイン・メニューの［**ツール**］＞［**記録オプション**］を選択するか，［記録開始］ダイアログ・ボックスの［**オプション**］をクリックします。記録オプションのツリーが表示されたダイアログ・ボックスが開きます。［COM/DCOM］の［**フィルタ**］ノードを選択します。

［環境］サブツリーをクリックすると，**ADO** オブジェクト，**RDS** オブジェクト，および **Remote** オブジェクトが一覧表示されます。フィルタには，［**Type Libraries**］ツリー（最初は空）も含まれています。タイプ・ライブラリは，次の手順で追加できます。

標準設定では，［環境］のすべての項目が選択されており，それらのオブジェクトへの呼び出しがフィルタに含められています（記録対象になります）。これらのオブジェクトをフィルタから除外するには，［**ADO objects**］，［**RDS objects**］，［**Remote objects**］の横にあるチェック・ボックスをクリアします。

2 別の COM タイプ・ライブラリを追加するには，［**追加**］をクリックし，［レジストリの参照］，［ファイル システムの参照］，［MTS の参照］のいずれかを選択します。

3［**レジストリの参照**］を選択すると，ローカル・コンピュータのレジストリに登録されているタイプ・ライブラリのリストが表示されます。



使用するライブラリ（1 つまたは複数）の横のチェック・ボックスを選択し，［**OK**］をクリックします。

4 ファイル・システムからタイプ・ライブラリを追加するには，［**追加**］をクリックして［**ファイル システムの参照**］を選択します。

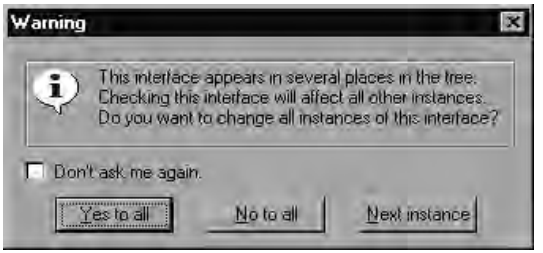
使用するファイルを選択して，［**OK**］をクリックします。

5 タイプ・ライブラリが［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。



［**Don't Ask me again**］をチェック・ボックスを選択してこのダイアログ・ボックスを閉じると，それ以後，このフィルタを使用し 1 つのオブジェクトに含まれるインタフェースのステータスを変更するたびに，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，[**Yes to all**] をクリックします。その他のクラスにあるそのインタフェースのステータスを変更しないときには，[**No to all**] をクリックします。そのインタフェースを使用する次のクラスを表示するには，[**Next instance**] をクリックします。

7 Microsoft Transaction Server のコンポーネントを追加するには，[**追加**] をクリックして [**MTS の参照**] を選択します。MTS サーバの名前を入力するための [MTS コンポーネント] ダイアログ・ボックスが表示されます。

MTS サーバの名前を入力して [**接続**] をクリックします。MTS のコンポーネントを記録するには，マシンに MTS クライアントをインストールしておく必要があります。

使用可能なパッケージのリストから MTS コンポーネントのパッケージを 1 つまたは複数選択して，[**追加**] をクリックします。パッケージが [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。

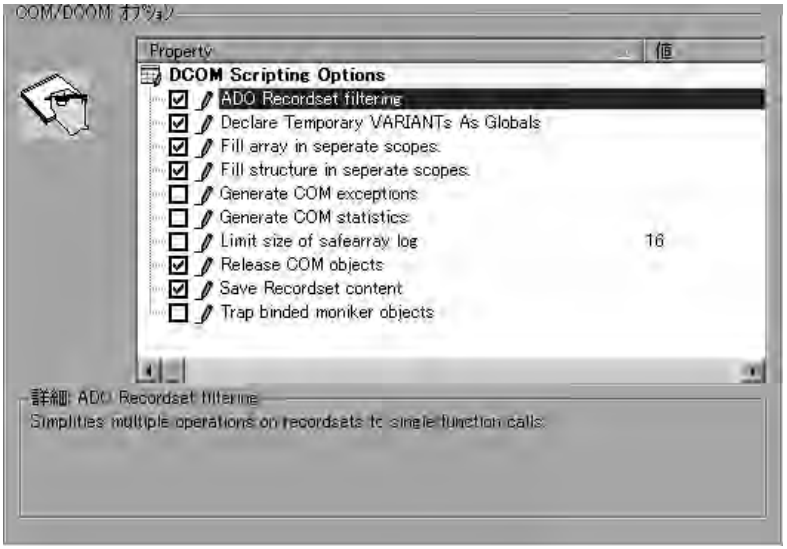
［除外インタフェース］ダイアログ・ボックスでチェックの印が付いているインタフェースが除外されます。表示されていないインタフェースを追加することもできます。［除外インターフェース］ダイアログ・ボックスで［**インタフェースの追加 ...**］をクリックし，GUID 番号（インタフェース ID）とインタフェース名を入力します。VuGen によって作成され，VuGen 画面の左側のカラムの選択ツリーに表示されている interfaces.h ファイルから，GUID をコピーすることもできます。［**インタフェースの追加 ...**］は，スクリプトによって不必要に呼び出されたものの，フィルタに表示され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 Recordset filtering**］：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）。

［**Declare Temporary VARIANTs as Globals**］：一時 VARIANT をローカル変数としてではなく，グローバル変数として定義します（標準設定では有効）。

［**Fill array in separate scopes**］：各配列を独立のスコープに入力します（標準設定では有効）。

［**Fill structure in separate scopes**］：各構造体を独立のスコープに入力します（標準設定では有効）。

［**Generate COM exceptions**］：記録中に例外が生成された COM 関数およびメソッドを生成します（標準設定では無効）。

［**Generate COM statistics**］：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）。

［**Limit size of SafeArray log**］: 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）。

［**Release COM Objects**］: 使用されなくなった COM オブジェクトの解放を記録します（標準設定では有効）。

［**Save Recordset content**］: 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）。

［**Trap binded moniker objects**］: バインドされているすべてのモニカ・オブジェクトをトラップします（標準設定では無効）。

COM スクリプトのオプションを設定するには，次の手順を実行します。

1 メイン・メニューの［**ツール**］＞［**記録オプション**］を選択するか，［記録開始］ダイアログ・ボックスの［**オプション ...**］をクリックします。記録オプションのツリーが表示されます。［COM/DCOM］オプションの［**オプション**］ノードを選択します。

2 オプション名の隣のチェック・ボックスをクリックしてオプションを選択します。

3［**OK**］をクリックして設定を保存し，終了します。

第 16 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 15 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，以下の項目について説明します。

- COM 仮想ユーザ・スクリプトについて
- VuGen COM スクリプトの構造について
- VuGen COM スクリプトの検証
- スクリプト内での相関候補の検索
- **既知の値の相関**

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。

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 ヘッダー・ファイルは，スクリプト内で使用されるインタフェース・ポインタとその他の変数を定義できます。各インタフェースには，そのインタフェースを一意に識別するインタフェース ID（IID）が割り当てられています。

インタフェースの定義の形式は，次のとおりです。

＜インタフェース・タイプ＞ * ＜インタフェース名＞ = 0; //"{ ＜インタフェース・タイプの IID ＞ }"

次の例では，インタフェース・タイプは **IDispatch**，インタフェースのインスタンスの名前は **IDispatch_0**，**IDispatch** タイプの **IID** は long 型の文字列です。

```
IDispatch* IDispatch_0= 0;//"{00020400-0000-0000-C000-
000000000046}"
```

仮想ユーザ・スクリプトのステートメント

COM 仮想ユーザ・スクリプトは，オブジェクトのインスタンスの作成，インタフェース・ポインタの取得，インタフェース・メソッドの呼び出しを行うコードで構成されます。各ユーザ・アクションは，1 つまたは複数の COM 呼び出しを生成します。COM 呼び出しはそれぞれ，VuGen によってステートメント・グループとして記述されます。グループは，それぞれが独立したスコープとして括弧で囲まれます。いくつかのステートメントによって，値の代入と型変換を行うことによって，main の呼び出しに備えます。オブジェクトの作成に必要な呼び出しのグループの例を次に示します。

```
{
GUID pClsid = lrc_GUID("student.student.1");
IUnknown * pUnkOuter = (IUnknown*)NULL;
unsigned long dwClsContext = lrc_ulong("7");
GUID riid = IID_IUnknown;
lrc_CoCreateInstance(&pClsid, pUnkOuter, dwClsContext, &riid,
(void**)&IUnknown_0, CHECK_HRES);
}
```

エラーの検査

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 スクリプトの検証

本項では，VuGen が COM クライアント・アプリケーションをエミュレートする仕組みを，例を使って説明します。

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 Ç… よって，作成するオブジェクトのコンテキストが設定されます。

```
unsigned long dwClsContext = lrc_ulong("7");
```

dwClsContext には，オブジェクトのコンテキストが含まれます（プロセス，ローカル，リモート，またはこれらのうちの複数の場所）。

4 VuGen によって，要求されたインタフェース ID を保持する変数が設定されます。この例では，**IUnknown** です。

```
GUID riid = IID_IUnknown;
```

riid には，**IUnknown** インタフェースのインタフェース ID が含まれます。

5 入力パラメータが用意された後に，**lrc_CoCreateInstance** への呼び出しにより，用意されたパラメータを使ってオブジェクトが作成されます。**IUnknown** インタフェースへのポインタがパラメータ IUnknown_0 に割り当てられます。このポインタは，次の呼び出しに必要です。

```
lrc_CoCreateInstance(&pClsid, pUnkOuter, dwClsContext, &riid,
(void**)&IUnknown_0, CHECK_HRES);
```

前述のとおり**入力パラメータが用意されています。**呼び出しが成功しているため，VuGen によって **CHECK_HRES** 値が挿入され，ユーザのシミュレーションの実行時にエラーの検査を行うように設定されます。**呼び出しの結果，IUnknown_0 に IUnknown インタフェースへのポインタが返されます。**IUnknown_0 は，以降の呼び出しで使用されます。

インタフェースの取得

オブジェクトを作成した時点で VuGen がアクセスできるのは **IUnknown** インタフェースだけです。VuGen は，オブジェクトと通信するために **IUnknown** インタフェースを使用します。これは，**IUnknown** 標準インタフェースの **QueryInterface** メソッドを使って行われます。VuGen のメソッド呼び出しの最初のパラメータは，インタフェースのインスタンスです。この例では，最初のパラメータは事前に **CoCreateInstance** によって設定された **IUnknown_0** ポインタです。**QueryInterface** 呼び出しには，取得すべきインタフェースの ID が入力項目として必要です。QueryInterface 呼び出しによって，指定した ID に対応するインタフェースへのポインタが返されます。

インタフェースを取得するには，次の手順を実行します。

1 最初に VuGen によって Istudent インタフェースの ID のパラメータである **riid** が設定されます。

```
GUID riid = IID_Istudent;
```

2 **QueryInterface** を呼び出すと，Istudent オブジェクトに **Istudent** インタフェースがあれば，出力パラメータ Istudent_0 にそのインタフェースへのポインタが割り当てられます。

```
lrc_IUnknown_QueryInterface(IUnknown_0, &riid, (void**)&Istudent_0,
CHECK_HRES);
```

インタフェースを使ったデータの設定

インタフェースのメソッドを使って，データを設定する例を以下に示します。例えば，アプリケーションでユーザが名前を入力するように求められるとします。この場合，名前を設定するためのメソッドが起動されます。VuGen によって 2 つのステートメントが記録されます。1 つは，名前の文字列を組み立てるために使用され，もう 1 つは名前のプロパティを設定します。

この関数呼び出し全体を組み立てるには，次の手順を実行します。

1 最初に，VuGen によって変数（**Prop Value**）に，入力された文字列と同じ値が設定されます。パラメータのタイプは，COM ファイルで使用される文字列型である BSTR です。

```
BSTR PropValue = lrc_BSTR("John Smith");
```

後で，「John Smith」をパラメータで置き換えることによって，この呼び出しをパラメータ化すると便利です。これによって，仮想ユーザ・スクリプトを実行するたびに，異なる名前を使用できるようになります。

2 次に，VuGen は名前を入力するための，**Istudent** インタフェースの **Put_Name** メソッドを呼び出します。

```
lrc_Istudent_put_name(Istudent_0, PropValue, CHECK_HRES);
```

インタフェースを使ったデータの返却

値を格納して，以降の呼び出しで入力項目として使用できるようにパラメータ化を行いたい場合は，データを入力するだけでは不十分で，アプリケーションからデータを返さなければなりません。

アプリケーションがデータを取得したときに VuGen によって何が行われるかを次の例に示します。

1 プロパティの値を格納する適切なタイプの変数（この例では BSTR）を作成します。

```
BSTR pVal;
```

2 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。

例えば，**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。例えば，lrc_DispMethod への次の呼び出しは，任意の引数「#3」および「var3」を渡します。

```
{
        GUID riid = IID_IDispatch;
        lrc_IOptional_QueryInterface(IOptional_0, &riid,
(void**)&IOptional_0, CHECK_HRES);
    }
    {
        VARIANT P1 = lrc_variant_short("47");
        VARIANT P2 = lrc_variant_short("37");
        VARIANT P3 = lrc_variant_date("3/19/1901");
        VARIANT var3 = lrc_variant_scode("4");
        lrc_DispMethod((IDispatch*)IOptional_0, "in_out_optional_args",
/*locale*/1024, &P1, &P2, OPTIONAL_ARGS, "#3", &P3, "var3", &var3,
LAST_ARG, CHECK_HRES);
```

IDispatch インタフェースを使用するさまざまな **lrc_Disp** メソッドの詳細については，第 17 章「COM 仮想ユーザ関数について」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 17 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（次の例を参照）。

```
VARIANT retValue = lrc_variant_empty();
retValue = lrc_DispMethod1((IDispatch*)IDispatch_0, "GetAgentsArray",
/*locale*/1033, LAST_ARG, CHECK_HRES);
```

次の例では，VuGen で文字列の配列として読み取られるバリアントである **retValue** から文字列を取り出す方法を示します。

まず，VuGen によってバリアントから BSTR 配列が抽出されます。

```
BstrArray array0 = 0;
array0 = lrc_GetBstrArrayFromVariant(&retValue);
```

array0 内にすべての値が含まれている状態で，後でパラメータ化の際に使用できるように，VuGen によって配列の要素を抽出するためのコードが生成されます。次に例を示します。

```
//GetElementFrom1DBstrArray(array0, 0); // 値：Alex
//GetElementFrom1DBstrArray(array0, 1); // 値：Amanda
....
```

VuGen には，さまざまなタイプの変換関数や，バリアントから従来の型を抽出するための関数があります。これらの詳細については第 17 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。

自動相関で検出されたスクリプトの検索と相関を行うには，次の手順を実行します。

1［**表示**］＞［**出力ウィンドウ**］を選択して，ウィンドウの下部に出力タブを表示します。［再生ログ］タブ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**］タブに表示されます。

3 値を相関させます。［相関クエリ］タブで，相関させるクエリ結果をダブルクリックします。この例では，メッセージの 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 表示枠の中で右クリックし，コンテキスト・メニューから［**相関を作成**］を選択します。結果の値を保存するパラメータの名前を入力するように求められます。

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

6［**はい**］をクリックして相関を確定します。

7 スクリプト内に現れる値をすべて検索するかどうかを尋ねるメッセージ・ボックスが表示されます。選択したステートメント内の値だけを置き換える場合は，［**いいえ**］をクリックします。次の候補を検索して置換する場合は，［**はい**］をクリックします。

8［検索と置換］ダイアログ・ボックスが開きます。オリジナルのステートメントも含め，すべての置き換えを確認します。

9［検索と置換］ダイアログ・ボックスを閉じます。ステートメントの値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，次の手順を実行します。

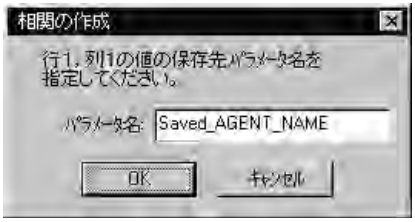
1 相関させる引数を見つけ（通常は **lrc_variant** ステートメント内にあります），値を選択します（引用符は除きます）。

2 ［**仮想ユーザ**］＞［**アクション内で相関をスキャン**］を選択します。

VuGen によって値が検索され，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］タブに一覧表示されます。

3 ［相関クエリ］タブで，相関させるクエリ結果をダブルクリックします。この例では，メッセージの 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。

```
lrc_save_rs_param (Recordset20_0, 1, 1, 0, "Saved_AGENT_NAME”);
```

6 ［**はい**］をクリックして相関を確定します。

7 スクリプト内に現れる値をすべて検索するかどうかを尋ねるメッセージ・ボックスが表示されます。選択したステートメント内の値だけを置き換える場合は，［**いいえ**］をクリックします。次の候補を検索して置換する場合は，［**はい**］をクリックします。

8［検索と置換］ダイアログ・ボックスが開きます。オリジナルのステートメントも含め，すべての置き換えを確認します。

第 17 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，以下の項目について説明します。

- COM 仮想ユーザ関数について
- 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 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「スクリプト生成オプションの設定」を参照してください。次の項では 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。例えば，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 18 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仮想ユーザ API Java 固有の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。

- AbCORBA-Java 仮想ユーザ・スクリプトの作成
- 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
- アプリケーション固有のヒント

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。

AbCORBA-Java 仮想ユーザ・スクリプトの作成

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，仮想ユーザ API 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 ライブラリまたはユーザ定義クラスを使った標準 Java コードで，そのスクリプトの拡張や変更ができます。

スクリプトを用意したら，VuGen を使ってスタンドアロン・モードで実行します。Sun の標準 Java コンパイラ，**javac.exe** によってエラーのないことが確認された後，スクリプトがコンパイルされます。

スクリプトを作成した後，スクリプトを自分の環境（LoadRunner シナリオ，Performance Center 負荷テスト，Tuning Module セッション，または Business Process Monitor プロファイル）に統合します。詳細については，『**Mercury LoadRunner コントローラ・ユーザーズ・ガイド**』，**Tuning Console** のマニュアル，**Performance Center** のマニュアル，または **Application Management** のマニュアルを参照してください。

スクリプトを記録と手作業で拡張する場合，Java 仮想ユーザ・スクリプトに関連したすべてのガイドラインと制限が適用されます。また，スクリプトで使用するクラス（例：**org.omg.CORBA.ORB**）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 14 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，254 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

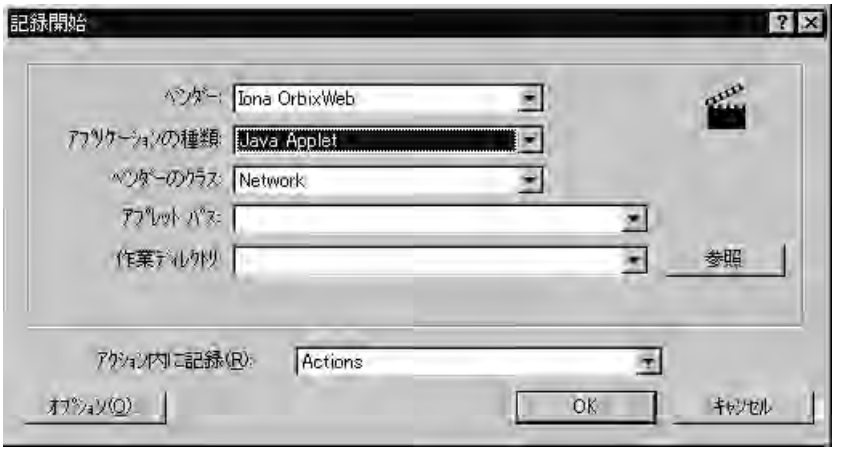
Corba 仮想ユーザの記録を開始する前に，アプリケーションまたはアプレットが記録用のマシンで正しく動作することを確認します。

VuGen を実行するマシンに，Sun の JDK を正しくインストールしておく必要があります（JRE だけでは不十分です）。スクリプトを記録する前に，インストールを完了させておく必要があります。**classpath** および **path** 環境変数が，JDK のインストール時の指示に従って設定されていることを確認します。

必要な環境設定の詳細については，第 14 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，次の手順を実行します。

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］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	N/A

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（例えば，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 3 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。**init**, **action**, and **end** の 3 つで，それぞれ **vuser_init**，**Actions**，**vuser_end** に対応しています。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。

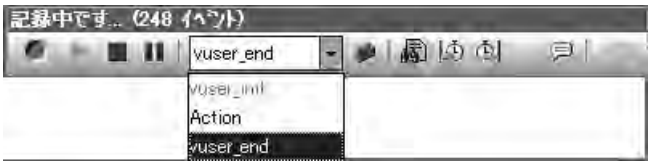
Action クラス内のメソッド	記録対象アクション	エミュレーション内容	実行のタイミング
init	**vuser_init**	サーバへのログイン	初期化時
action	**Action**	クライアントの動作	実行中
end	**vuser_end**	ログオフ処理	終了時または停止時

注：必ず **vuser_init** セクションで **org.omg.CORBA.ORB** 関数をインポートして，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。

9 アプリケーション内で，標準的な操作を実行します。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 ［**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。

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，次の手順を実行します。

1［コントロール パネル］から［Java Plug-in］を開きます。［**スタート**］＞［**設定**］＞［**コントロール パネル**］を選択して，［**Java Plug-in**］コンポーネントを開きます。［基本］タブが開きます。

2［**Java Plug-In の有効化**］チェック・ボックスをクリアして，［**適用**］をクリックします。その後，［**Java Plug-In の有効化**］チェック・ボックスを選択しなおして，［**適用**］をクリックします。

3［ブラウザ］タブを開きます。

4［**Microsoft Internet Explorer**］チェック・ボックスをクリアして，［**適用**］をクリックします。その後，［**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。

アプリケーション固有のヒント

JDK1.2 以降と Corba アプリケーションを組み合わせて実行すると，ベンダ固有の Corba クラスではなく，JDK 内部の Corba クラスがロードする場合があります。仮想マシンがベンダ固有のクラスを必ず使用するようにするには，コマンド・ラインで次の java.exe パラメータを指定します。

Visigenic 3.4
-Dorg.omg.CORBA.ORBClass=com.visigenic.vbroker.orb.ORB
-
Dorg.omg.CORBA.ORBSingletonClass=com.visigenic.vbroker.orb. ORBSingleton

Visigenic 4.0
-Dorg.omg.CORBA.ORBClass=com.inprise.vbroker.orb.ORB
-Dorg.omg.CORBA.ORBSingletonClass=com.inprise.vbroker.orb.ORBSingleton

OrbixWeb 3.x

-Dorg.omg.CORBA.ORBClass=IE.Iona.OrbixWeb.CORBA.ORB
-
Dorg.omg.CORBA.ORBSingletonClass=IE.Iona.OrbixWeb.CORBA. singleton
ORB

OrbixWeb 2000

-Dorg.omg.CORBA.ORBClass=com.iona.corba.art.artimpl.ORBImpl
-
Dorg.omg.CORBA.ORBSingletonClass=com.iona.corba.art.artimpl. ORBSing
leton

第 19 章

RMI-Java 仮想ユーザ・スクリプトの作成

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仮想ユーザ API Java 固有の 関数を使って拡張したりすることができます。

本章では，以下の項目について説明します。

- RMI-Java 仮想ユーザ・スクリプトの作成について
- 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・スクリプトの作成について

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，仮想ユーザ API 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。

スクリプトを作成した後，スクリプトを自分の環境（LoadRunner シナリオ，Performance Center 負荷テスト，Tuning Module セッション，またはビジネス・プロセス・モニタ・プロファイル）に統合します。詳細については，『**Mercury LoadRunner コントローラ・ユーザーズ・ガイド**』，**Tuning Console**，

Performance Center，または **Application Management** のマニュアルを参照してください。

スクリプトを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 14 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，254 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

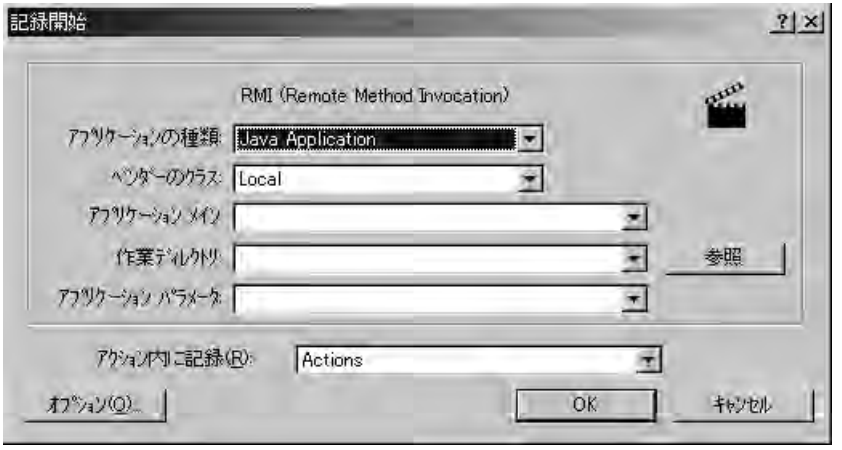
RMI 仮想ユーザを記録する前に，記録するマシンでアプリケーションまたはアプレットが正しく機能することを確認してください。

スクリプト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 14 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，VuGen を使わずにそれらをロードする場合よりも，多少時間がかかります。

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（例えばプロパティ・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 3 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。

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 function: 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，44 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 7 部

E- ビジネス・プロトコル

第 20 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，以下の項目について説明します。

- FTP 仮想ユーザ・スクリプトの作成について
- 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・スクリプトの作成について

FTP プロトコルは，FTP サーバに対して作業しているユーザのアクションをエミュレートする，下層プロトコルです。

FTP に関して，ユーザが FTP サーバにログインし，ファイルを転送してログアウトするのをエミュレートします。スクリプトを作成するには，FTP セッションを記録するか FTP 関数を手作業で入力します。

FTP セッションを記録するときに，VuGen は，メール・クライアントのアクションをエミュレートする関数を生成します。FTP，HTTP，およびメール・プロトコルなど，複数のプロトコルを介して通信を行う場合は，それらを全部記録できます。複数のプロトコルを指定する手順については，『**第 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「VuGen を使った記録」を参照してください。

FTP プロトコルのスクリプトを作成するには，［e ビジネス］カテゴリで［File Transfer Protocol（FTP)］プロトコル・タイプを選択し，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 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「VuGen を使った記録」を参照してください。

スクリプトを作成した後，スクリプトを自分の環境（LoadRunner シナリオ，Performance Center 負荷テスト，Tuning Module セッション，またはビジネス・プロセス・モニタ・プロファイル）に統合します。詳細については，『**Mercury LoadRunner コントローラ・ユーザーズ・ガイド**』，**Tuning Console**，**Performance Center**，または **Application Management** のマニュアルを参照してください。

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 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「スクリプト生成オプションの設定」を参照してください。次の節では 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。例えば，**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 21 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 仮想ユーザ・スクリプトの作成について
- LDAP 関数の処理
- 識別名エントリの定義

以降の情報は，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，275 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 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「VuGen を使った記録」を参照してください。

スクリプトを作成した後，スクリプトを自分の環境（LoadRunner シナリオ，Performance Center 負荷テスト，Tuning Module セッション，またはビジネス・プロセス・モニタ・プロファイル）に統合します。詳細については，『**Mercury LoadRunner コントローラ・ユーザーズ・ガイド**』，**Tuning Console**，**Performance Center**，または **Application Management** のマニュアルを参照してください。

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を指定できます。詳細については，『**第 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「スクリプト生成オプションの設定」を参照してください。次の節では 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。例えば，**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 22 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，以下の項目について説明します。

- Web 仮想ユーザ・スクリプトの作成について
- 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）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。

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またはセッション・ステップ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またはセッション・ステップに組み込む方法の詳細については，『**Mercury LoadRunner コントローラ・ユーザーズ・ガイド**』および『**ProTune Console User's Guide（英語版）**』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（例えば 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に対する応答時間が指定時間（例えば 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。

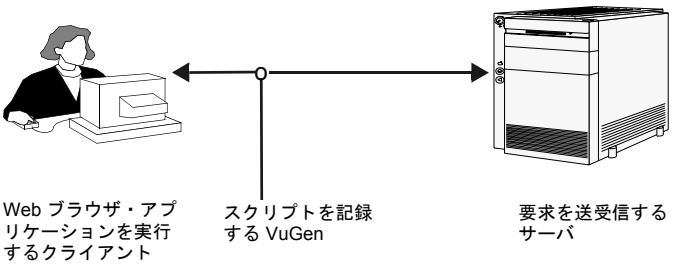
- ホーム・ページのロード
- 製品情報が掲載されているページへの移動
- クエリの発行
- サーバからの応答の待機

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。コンソールまたは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 Tuning Module Console 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略を説明します。

Web 仮想ユーザ・スクリプトを作成するには，次の手順を実行します。

1 **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 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「VuGen を使った記録」を参照してください。

2 **記録オプションを設定します。**

記録オプションを設定します。記録オプションの設定については，第 24 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 25 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 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「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 28 章「負荷下の Web ページ検証」，第 29 章「Web とワイヤレス仮想ユーザ・スクリプトの変更」，および第 30 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 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「VuGen パラメータを使った作業」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 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「実行環境の設定」を参照してください。

7 **相関を実行します。**

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 30 章「Web 仮想ユーザ・スクリプトの相関ルールの設定」および第 31 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 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「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 33 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。

スクリプトを作成した後，スクリプトを自分の環境（LoadRunner シナリオ，Performance Center 負荷テスト，Tuning Module セッション，またはビジネス・プロセス・モニタ・プロファイル）に統合します。詳細については，『**Mercury LoadRunner コントローラ・ユーザーズ・ガイド**』，**Tuning Console**，**Performance Center**，または **Application Management** のマニュアルを参照してください。

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。

VuGen は，ユーザ・アクションを記述するスクリプトを作成します。標準設定では，行われたアクションに直接対応する関数を使ってスクリプトが作成されます。URL（**web_url**），リンク（**web_link**），画像（**web_image**），フォーム送信（**web_submit_form**）などの関数を作成します。作成されるスクリプトは非常に直感的で，コンテキスト・センシティブ記録に似ています。

```
/* HTML ベース・モード ーユーザ・アクション 記述スクリプト */
...
web_url("MercuryWebTours",
        "URL=http://localhost/MercuryWebTours/",
        "Resource=0",
        "RecContentType=text/html",
        "Referer=",
        "Snapshot=t1.inf",
        "Mode=HTML",
        LAST);

web_link("Click Here For Additional Restrictions",
        "Text=Click Here For Additional Restrictions",
        "Snapshot=t4.inf",
        LAST);

web_image("buttonhelp.gif",
        "Src=/images/buttonhelp.gif",
        "Snapshot=t5.inf",
        LAST);
...
```

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・スクリプトを Java 仮想ユーザ・スクリプトに変換するには，次の手順を実行します。

1 空の Java 仮想ユーザ・スクリプトを作成して保存します。

2 空の Web 仮想ユーザ・スクリプトを作成して保存します。

3 標準の HTML/HTTP 記録オプションで Web セッションを記録します。

4 仮想ユーザ・スクリプトを再生します。正常に再生できたら，スクリプト全体を切り取り，テキスト・ドキュメントに貼り付け，テキストとして **.txt** ファイルに保存します。テキスト・ファイルでパラメータの括弧を Web 形式の「"{ }"」から Java 形式の「"< >"」に変更します。

5 DOS コマンド・ウィンドウを開いてお使いの製品の ¥dat ディレクトリに移動します。

6 次のコマンドを入力します。

＜アプリケーションのインストール先＞ ¥bin¥sed -f web_to_java.sed **filename** ＞ **outputfilename**

ここで **filename** には先に保存したテキスト・ファイルのフル・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 23 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。

- Web 仮想ユーザ関数について
- 関数の追加と編集
- Web 関数リスト
- キャッシュを使用したパフォーマンスの向上

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 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）。

➤ 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，『**第 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「VuGen の紹介」を参照してください。

関数の追加と編集

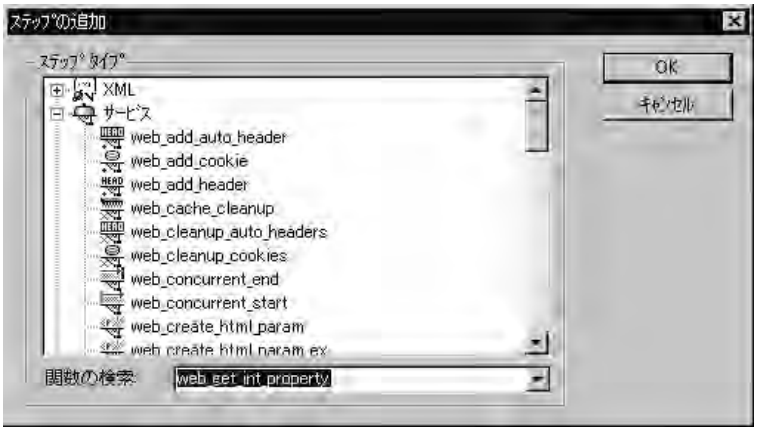
多くの Web 仮想ユーザ関数はブラウザまたはツールキットのセッション中に記録されます。

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 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「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，次の手順を実行します。

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



2 使用する関数を選択して，［**OK**］をクリックします。ほとんどの Web 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。このダイアログ・ボックスでは，関数の引数を指定できます。

3 プロパティを指定して［**OK**］をクリックします。関数が引数と一緒にカーソルの位置に挿入されます。

［プロパティ］ダイアログ・ボックスを開いて，引数の値を変更することによって既存のステップを編集できます。これは，ツリー・ビューをサポートするプロトコルの場合にのみ有効です（WAP には使用できません）。

既存のステップを編集するには，次の手順を実行します。

1 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
- 並行グループ関数

- 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

GUI レベルの仮想ユーザは，ユーザ・アクションをエミュレートするのに上記以外の関数を使用します。詳細については，563 ページ「GUI レベル仮想ユーザ関数の使用」を参照してください。

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。

web_dump_cache

リソースをブラウザ・キャッシュにダンプします。

web_load_cache

キャッシュの内容をロードします。

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。

並行グループ関数

web_concurrent_end	並行グループの終了を示します。
web_concurrent_start	並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またはチューニング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またはチューニング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 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「仮想ユーザ・スクリプトの拡張」を参照してください。

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。

キャッシュを使用したパフォーマンスの向上

VuGen のキャッシュのシミュレート機能を利用して，ユーザ・パフォーマンスを大幅に向上できます。キャッシングを行うと，CPU の使用量を約 15 % 削減できます。

スクリプト内にキャッシングを実装する場合は，**web_dump_cache** および **web_load_cache** 関数を手動で追加します。

キャッシュへの情報のダンプ

キャッシングを実装するにはまず，キャッシュ・ファイルに情報をダンプします。**web_dump_cache** 関数を実行して，**FileName** 引数で指定した場所にキャッシュ・ファイルを生成します。キャッシュ・ファイルを生成するためには，この関数を 1 回必ず実行する必要があります。

次の例では，**web_dump_cache** 関数は，スクリプト実行中の C:¥temp for each **VuserName** パラメータの C:¥temp にキャッシュ・ファイルを生成しています。

```
web_dump_cache("paycheckcache","FileName=c:¥¥temp¥¥{VuserName}paych
eck", "Replace=yes", LAST)
```

1 個の仮想ユーザを 10 回実行すると，VuGen は次の形式で 10 個のキャッシュ・ファイルを生成します。接頭辞は VuserName の値になります。

```
Ku001paycheck.cache
Ku002paycheck.cache
Ku003paycheck.cache
…
```

1 番目と 2 番目の引数（この例では paycheckcache と paycheck)）を変更して，現在のトランザクション名を反映させることができます。すべてのリソースをロードした後，スクリプトの最後にこの関数を置きます。

キャッシュからの情報のロード

キャッシング実装の最終ステップは，キャッシュ・ファイルに保存された情報のロードです。**web_load_cache** 関数は，**FileName** 引数で指定された場所にあるキャッシュ・ファイルをロードします。**web_load_cache** 関数にはキャッシュ・ファイルが必須です。したがって，この関数は **web_dump_cache** 関数を実行した後にのみ実行できます。

web_load_cache 関数が C:¥temp から **paycheck** キャッシュ・ファイルをロードする例を次に示します。

```
web_load_cache("ActionLoad","FileName=c:¥¥temp¥¥{VuserName}paycheck",LAST)
```

キャッシュ関数の挿入

スクリプト内にキャッシングを実装するにはまず，キャッシュ・ファイルに情報を保存する必要があります。再生中に，各仮想ユーザはこの情報を呼び出します。

キャッシュ関数を使用するには，次の手順に従います。

1 **web_dump_cache** 関数をスクリプトの先頭に挿入します。

2 スクリプトを最低 1 回実行します。

3 **web_load_cache** 関数をスクリプトの Vuser アクションの前に挿入します。

4 **web_dump_cache** 関数をコメントにします。

5 スクリプトを実行し，保存します。

キャッシングの例

給与明細を参照している PeopleSoft Enterprise 仮想ユーザの例を次に示します。

```
Action()
{
//     web_add_cookie("storedCookieCheck=true; domain=pbntas05;
path=/");

web_load_cache("ActionLoad","FileName=c:\\temp\\{VuserName}pay-
check",LAST);

      web_browser("signon.html",
           DESCRIPTION,
           ACTION,
           "Navigate=http://pbntas05:8200/ps/signon.html",
           LAST);
      lr_think_time(35);

      web_edit_field("userid",
           "Snapshot=t1.inf",
           DESCRIPTION, "Type=text",
           "Name=userid",
           ACTION,
           "SetValue={VuserName}",
           LAST);
```

```
        web_edit_field("pwd",
            "Snapshot=t2.inf",
            DESCRIPTION,
            "Type=password",
            "Name=pwd",
            ACTION,
            "SetValue=HCRUSA_KU0007",
            LAST);

        lr_start_transaction("login");
            web_button("Sign In",
                    "Snapshot=t3.inf",
                    DESCRIPTION,
                    "Type=submit",
                    "Tag=INPUT",
                    "Value=Sign In",
                    LAST);
        lr_end_transaction("login", LR_AUTO);

        web_image_link("CO_EMPLOYEE_SELF_SERVICE",
            "Snapshot=t4.inf",
            DESCRIPTION,
            "Alt=",
            "Name=CO_EMPLOYEE_SELF_SERVICE",
            "Ordinal=1",
            LAST); ...

        web_text_link("Sign out",
            "Snapshot=t7.inf",
            DESCRIPTION,
            "Text=Sign out",
            "FrameName=UniversalHeader",
            LAST);
    web_dump_cache("paycheck","FileName=c:\\{VuserName}paycheck",
    "Replace=yes", LAST);
        return 0;
    }
```

第 24 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，次の項目について説明します。

- 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
- 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，現実の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の生成にかかわる 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。

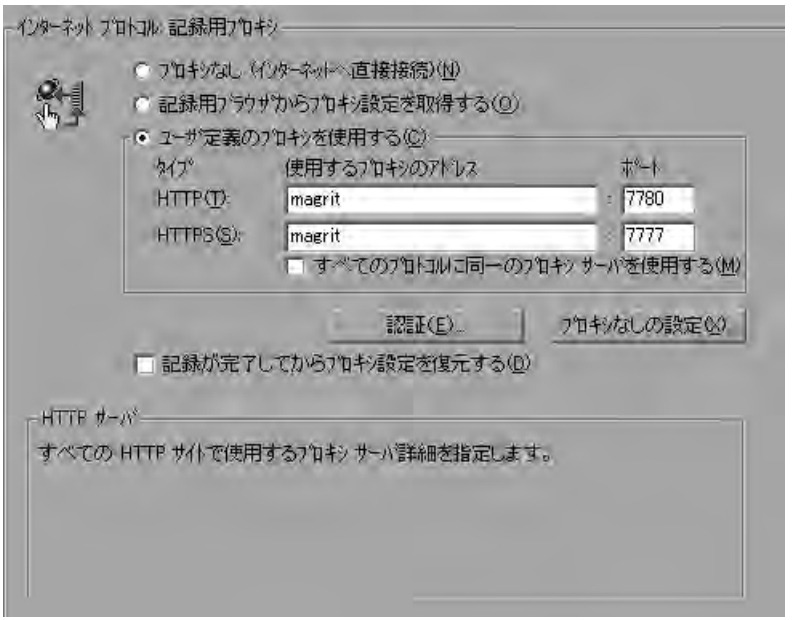
プロキシ設定を使った作業

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。



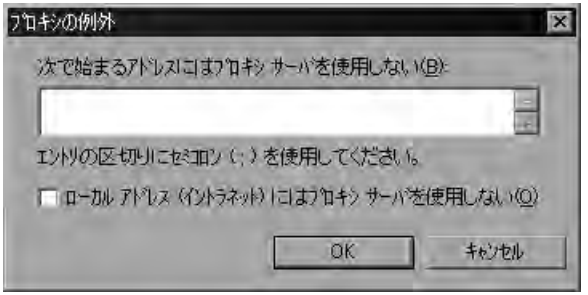
次のプロキシ・オプションのいずれかを選択できます。

- ［**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 および 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，［**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**］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。例えば，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
- 記録エンジンの選択
- 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。

［**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。

［**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。

［**記録中，HTTP エラー時にスクリプトにコメントを追加する**］：HTTP 要求のエラーが発生するたびにスクリプトにコメントを追加します。エラー要求は，記録中に 400 番以降の値を持つサーバ応答を生成した要求として定義されます。

［**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 に正しく表示されます。このオプションは英語以外の UTF-8 エンコードのページでのみ有効にしてください。記録するサイトの言語がオペレーティング・システムの言語と一致している必要があります。英語以外の Web ページは，同じスクリプト内で異なるエンコーディング（ISO-8859-1 または shift_jis を組み合わせて）では記録できません。

➤ ［**EUC-JP**］：日本語版の 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 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）。

記録エンジンの選択

VuGen の標準設定では，シングル・プロトコルの記録を含むすべての記録でマルチプロトコルの記録エンジンが使用されます。

下位互換性を維持するためにシングル・プロトコルの記録エンジンを使用するには，［詳細記録オプション］の［**Record script using single-protocol recording engine**］オプションを選択します。このオプションを有効にすると，次回のセッション記録でシングル・プロトコルが使用されます。

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 ヘッダーを自動的に送信します。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。［**ヘッダを記録しない**］，［**リスト内のヘッダを記録**］，［**リストに定義されていないヘッダを記録**］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，［**ヘッダを記録しない**］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，［**Record headers in list**］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，次の手順を実行します。

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，［**ヘッダを記録しない**］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。［**コンテント タイプをフィルタにかけない**］，［**リスト内のコンテント タイプを除外**］，［**リストに定義されていないコンテント タイプを除外**］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コンテント 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。

例えば，Web サイトのテキストと画像だけを対象とするには，［**リスト内のコンテント 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-java スクリプト**」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，次の手順を実行します。

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 ［**コンテント タイプ**］をクリックします。［コンテント 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，［**コンテント 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，［**リスト内のコンテント 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コンテント 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。［**リスト内のコンテント タイプを除外**］と［**リストに定義されていないコンテント 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，［コンテント 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("MercuryWebTours",
        "URL=http://localhost/MercuryWebTours/",
        "Resource=0",
        "RecContentType=text/html",
        "Referer=",
        "Snapshot=t1.inf",
        "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。例えば，**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，次の手順を実行します。

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 25 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。

- 記録オプションの設定について
- 記録用のブラウザの指定
- 記録レベルの選択

以降の情報は，Web および PeopleSoft Enterprise 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。

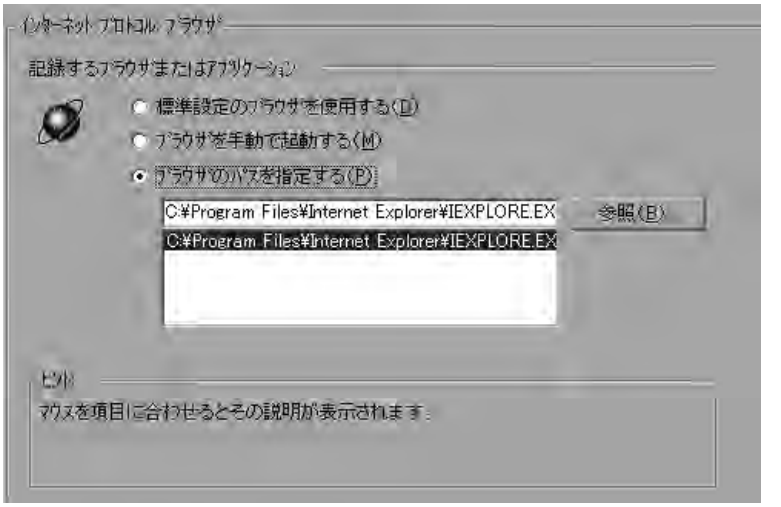
記録する前に，［記録オプション］を設定し，記録する情報，記録に使用するブラウザまたはクライアント，スクリプトの内容を指定します。

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 24 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 30 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。

記録用のブラウザの指定

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。

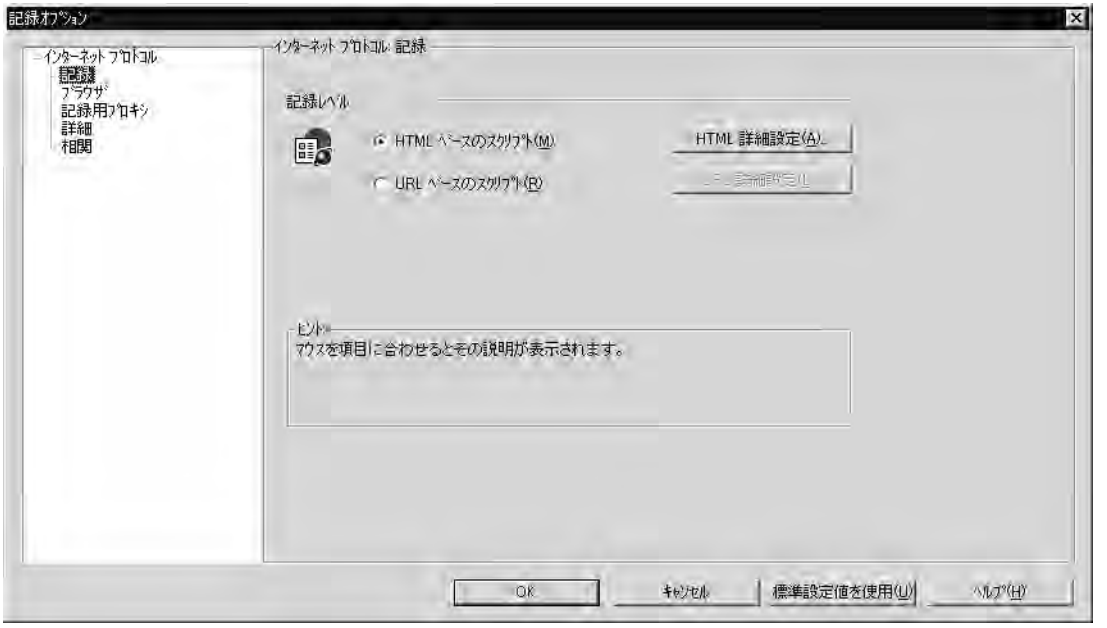
- ［**標準設定のブラウザを使用する**］：VuGen によって記録用コンピュータで通常使用する Web ブラウザとして指定されているブラウザが使用されます。
- ［**ブラウザを手動で起動する**］：VuGen によって記録の開始時にブラウザが起動されません。ユーザは，記録セッションを開始した後に，自分でブラウザまたはアプリケーションを起動しなければなりません。
- ［**ブラウザのパスを指定する**］：VuGen によって指定したブラウザが使用されます。パスのリストからパスを選択するか，［**参照**］ボタンをクリックして，使用するアプリケーションの場所を見つけます。

記録レベルの選択

記録レベルを選択することによって，仮想ユーザ・スクリプトまたは Tuning Module セッション・ステップの生成時に記録する情報と使用する関数を指定できます。記録レベルは，目的または環境によって選択します。指定できるレベルは，**HTML ベースのスクリプト**および **URL ベースのスクリプト**です。

以下のガイドラインに従って，どちらの記録レベルを選択するかを決定します。

- JavaScript を使用しないブラウザ・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。



PeopleSoft Enterprise 仮想ユーザと Oracle Web Applications 11i 仮想ユーザには，**GUI ベースのスクリプト**という記録レベルもあります。このオプションを指定すると，すべてのユーザ・アクションに対してわかりやすいコンテキスト・センシティブ関数が生成されます。詳細については，860 ページ「GUI レベル仮想ユーザに対する記録レベルの選択」を参照してください。

HTML ベースのスクリプト・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。これらのステップもわかりやすいのですが，JavaScript コードの忠実なエミュレーションが反映されているわけではありません。

```
/* HTML ベース・モード ーユーザ・アクション 記述スクリプト */
...
web_url("MercuryWebTours",
        "URL=http://localhost/MercuryWebTours/",
        "Resource=0",
        "RecContentType=text/html",
        "Referer=",
        "Snapshot=t1.inf",
        "Mode=HTML",
        LAST);

web_link("Click Here For Additional Restrictions",
        "Text=Click Here For Additional Restrictions",
        "Snapshot=t4.inf",
        LAST);

web_image("buttonhelp.gif",
        "Src=/images/buttonhelp.gif",
        "Snapshot=t5.inf",
        LAST);
...
```

URL ベースのスクリプト・モード・オプションでは，ユーザのアクションによって送信されたサーバからのすべてのブラウザ要求とリソースが，VuGen によって記録されます。自動的にあらゆる HTTP リソースが URL ステップ（**web_url** ステートメント）として記録されます。ブラウザの通常の記録では，URL ベース・モードは相関関連の問題が起こりやすいため推奨されません。ただし，アプレットや非ブラウザ・アプリケーションなどのページを記録している場合は，このモードが最適です。

URL ベースのスクリプトは，HTML ベースのスクリプトほどわかりやすくはありません。これは，すべてのアクションが **web_link** や **web_image** などではなく **web_url** のステップとして記録されるためです。

```
/* URL ベース・モード : web_url 関数のみ */ …
...
web_url("spacer.gif",
        "URL=http://graphics.mercury.com/images/spacer.gif",
        "Resource=1",
        "RecContentType=image/gif",
        "Referer=",
        "Mode=HTTP",
        LAST);

web_url("calendar_functions.js",
        "URL=http://www.im.mercury.com/travelp/calendar_functions.js",
        "Resource=1",
        "RecContentType=application/x-javascript",
        "Referer=",
        "Mode=HTTP",
        LAST);
...
```

マルチ・プロトコル・スクリプトを記録していなければ，記録中に記録レベルと詳細記録オプションを切り替えることができます。記録レベルを切り替える方法は，パフォーマンスのチューニングを行う上級ユーザ向けです。

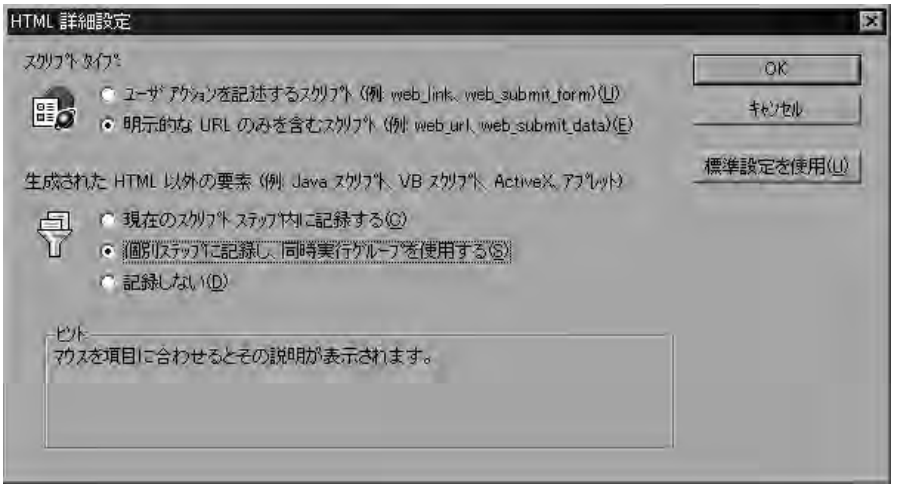
記録後，元の記録とは異なる方式でスクリプトを再生成することもできます。例えば，スクリプトを HTML ベース・レベルで記録している場合，URL ベース・レベルで再生成できます。スクリプトを再生成するには，［**ツール**］＞［**スクリプトの再生成**］を選択し，［**オプション**］をクリックして，再生成のための記録オプションを設定します。

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

web_link("Click Here For Additional Restrictions",
        "Text=Click Here For Additional Restrictions",
        "Snapshot=t4.inf",
        LAST);

web_image("buttonhelp.gif",
        "Src=/images/buttonhelp.gif",
        "Snapshot=t5.inf",
        LAST);
...
```

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次に示します。

```
/* 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.mercury.com/restrictions.html",
        "TargetFrame=",
        "Resource=0",
        "RecContentType=text/html",
        "Referer=http://www.mercury.com/home?...
        "Snapshot=t4.inf",
        "Mode=HTML",
        LAST);

web_url("buttonhelp.gif",
        "URL=http://www.mercury.com/home?com/rstr?BV_EngineID...,
        "TargetFrame=main",
        "Resource=0",
        "RecContentType=text/html",
        "Referer=http://www.mercury.com/home?...
        "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。例えば，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。

次のオプションが使用できます。

➤ 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。すべてのリソースが，ページについて生成された **web_url**，**web_link**，**web_submit_data** などのステートメントの引数としてリストされます。Web 関数の引数となったリソースには **EXTRARES** フラグが付けられます。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。そのページの関数（**web_url** や **web_link** など）にはそれらが項目として含まれません。特定のリソースについて生成されたすべての **web_url** 関数は，並行グループ内に配置されます（**web_concurrent_start** と **web_concurrent_end** で囲まれます）。

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.mercury.com/home?com/rstr?BV_EngineID...,
        "TargetFrame=main",
        "Resource=0",
        "RecContentType=text/html",
        "Referer=http://www.mercury.com/home?...
        "Snapshot=t5.inf",
        "Mode=HTML",
        LAST);
```

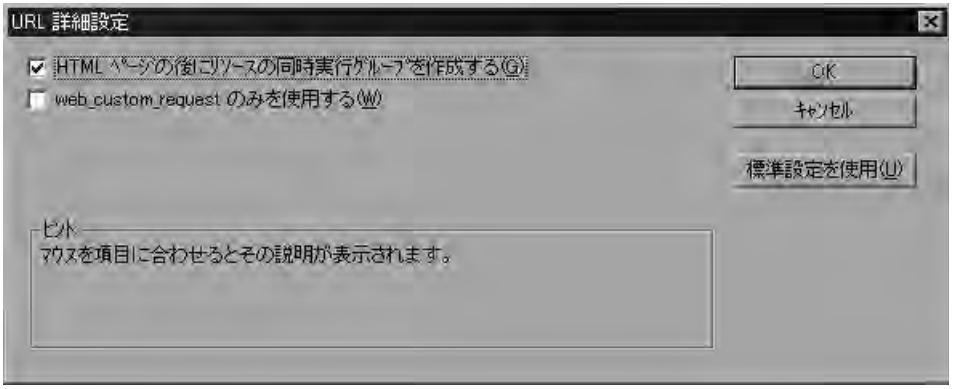
URL ベースのモードでの記録では，VuGen によってページ上のすべてのフレームの内容が記録され，TargetFrame 属性が省略されます。

URL ベースのオプションの詳細設定

URL ベース・モード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，ブラウザ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，リソースが並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，画像などのファイルと **js** ファイルが含まれます。このオプションを無効にすると，リソースは別々の **web_url** ステップとしてリストされますが，同時実行グループとしてのマークは付きません。

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**］オプションを有効にして記録したセッションを示すものです。

```
web_concurrent_start(NULL);
...
web_url("Click Here For Additional Restrictions",
        "URL=http://www.mercury.com/restrictions.html",
        "Resource=0",
        "RecContentType=text/html",
        "Referer=http://www.mercury.com/home?...
        "Snapshot=t4.inf",
        "Mode=HTTP",
        LAST);

web_url("buttonhelp.gif",
        "URL=http://www.mercury.com/home?com/rstr?BV_EngineID...,
        "Resource=0",
        "RecContentType=text/html",
        "Referer=http://www.mercury.com/home?...
        "Snapshot=t5.inf",
        "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こと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，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。

ブラウザのキャッシュ・オプションはシングル・プロトコル Web（HTTP/HTML）仮想ユーザにのみ適用されます。マルチ・プロトコルには適用されません。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，305 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.mercury.com",
        "URL=http://www.mercury.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，VuGen は HTTP ヘッダーを通じて Web ページで使われているコードを検出します。文字セットに関する情報が HTTP ヘッダーに存在しない場合は，HTML のメタ・タグを調べます。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，VuGen 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，VuGen に正しいコードで記録させることができます。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必要があるのは，VuGen 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，ページが EUC コードで書かれていることが事前にわかっている場合だけです。

記録中，VuGen は EUC コードの文字列を Web サーバから受信すると，それを SJIS に変換します。変換された SJIS 文字列はスクリプトの Action 関数に保存されます。しかし，再生を正常に実行するためには，文字列を再び EUC に変換してから Web サーバに送り返す必要があります。このため，VuGen は Action 関数の前に **web_sjis_to_euc_param** 関数を追加します。この関数で SJIS 文字列を EUC に再変換します。

次の例では，ユーザが EUC で書かれた Web ページに移動してリンクをクリックします。VuGen は Action 関数を記録し，**web_sjis_to_euc_param** 関数を Action 関数の前のスクリプトに追加します。

```
web_sjis_to_euc_param("param_link","Search");

web_link("LinkStep","Text={param_link}");
```

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。記録レベルと詳細記録オプションは記録中に切り替えることができます。ただし，シングル・プロトコル Web（HTTP/HTML）仮想ユーザを記録していることが条件です。

記録オプションを設定するには，次の手順を実行します。

1 ［**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は，［**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，316 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，321 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。日本語版 Windows のユーザの場合は，［**EUC**］オプションを選択して，VuGen で EUC エンコーディングを使用するよう設定します。ページに EUC コード（日本語のコンテンツ）だけが使用されている Web サイトを記録する場合は，［**EUC**］オプションを選択します。VuGen は，EUC の文字列を SJIS に変換し，**web_sjis_to_euc_param** 関数を追加します。この情報がサーバからブラウザに（HTTP ヘッダーまたは HTML メタ・タグで）送信される場合は，このオプションを有効にする必要はありません。

第 26 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。

- インターネット実行環境の設定について
- 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
- インターネットお気に入りの設定
- Web サイトのフィルタリング
- デバッグ情報の取得
- HTML 圧縮の実行

以降の情報は，Web およびワイヤレスなど，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 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「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の詳細については，『**第 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「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。インターネット実行環境の設定では，プロキシ，ブラウザ，その他の設定が行えます。

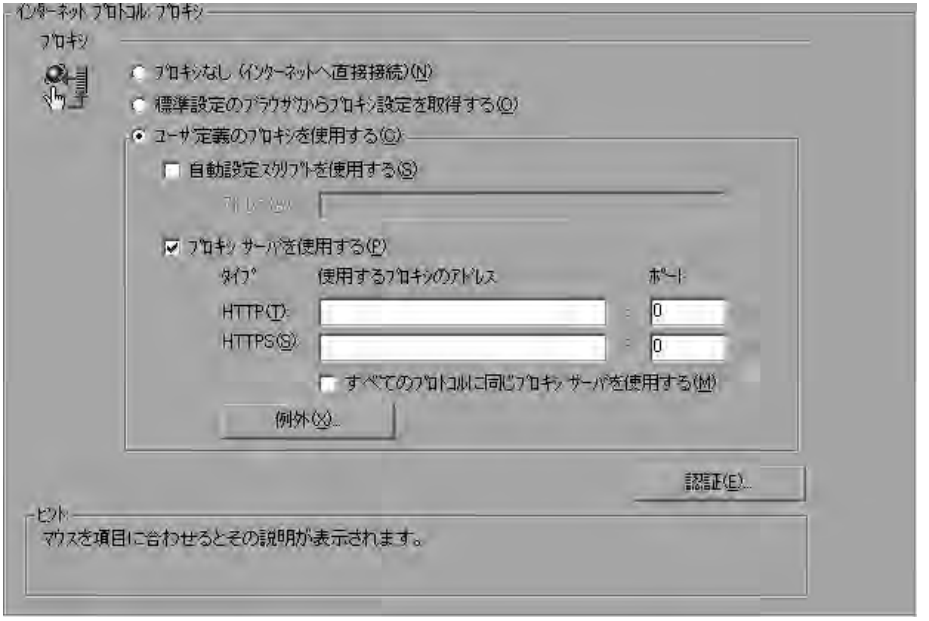
インターネット関連の実行環境の設定は，［実行環境設定］ダイアログ・ボックスで行います。必要な設定を行うためのノードをクリックします。

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

注：仮想ユーザ関数を使って割り当てら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用方法については，第 23 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 : 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を使用します。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］オプションについては，第 25 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**］: すべての仮想ユーザが，インターネットに常に直接接続します。つまり，プロキシ・サーバを使用せずに接続します。
- ［**標準設定のブラウザからプロキシ設定を取得する**］: すべての仮想ユーザが実行されているマシンで，通常使うブラウザとなっているプロキシの設定を使用します。

➤［**ユーザ定義のプロキシを使用する**］: すべての仮想ユーザが同じユーザ定義のプロキシ・サーバを使用します。実際のプロキシ・サーバの構成情報や，自動設定のための設定スクリプト（.**pac** ファイル）へのパスなどで設定を行います（331 ページ「自動プロキシ設定の設定方法」を参照）。

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。

プロキシ情報を設定した後，プロキシ・サーバの認証情報を指定し，例外をプロキシのルールに指定できます。

注 : プロキシで基本認証がサポートされていないという前提のもとで，再生中にプロキシの応答があるまで待機するよう仮想ユーザに指示するには，ステートメント web_set_sockets_option("PROXY_INITIAL_BASIC_AUTH", "0"); を追加します。

認証

プロキシ・サーバで仮想ユーザごとに認証を必要とする場合は，このダイアログ・ボックスを使用して，適切なパスワードとユーザ名を入力します。

［**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**］: 仮想ユーザによるプロキシ・サーバへのアクセスに必要なパスワードを入力します。

注 : 認証を記録時に動的に追加する場合や，複数のプロキシ・サーバの認証を追加する場合は，**web_set_user** 関数を使用します。詳細については，「**オンライン関数リファレンス**」（［**ヘルプ**］ > ［**関数リファレンス**］）を参照してください。

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，仮想ユーザに直接アクセスさせる（つまり，プロキシ・サー

バを使用せずにアクセスさせる）URL がある場合は，それらの 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。各エントリ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場合は，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

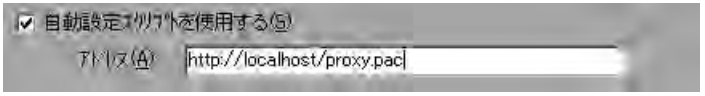
自動プロキシ設定は，ほとんどのブラウザでサポートされています。この機能では，プロキシ割り当て情報を含む JavaScript ファイル（通常は **.pac** 拡張子付き）を指定できます。このスクリプトは，URL に基づいて，どの場合にはプロキシ・サーバを使用して，どの場合には直接サイトへ接続するのかをブラウザに指定します。また，このスクリプトでは，アドレスごとに異なるプロキシ・サーバを使うよう指定することができます。

設定スクリプトを使用するように VuGen または Internet Explorer ブラウザを設定できます。自動プロキシ設定のファイルを指定すると，仮想ユーザはテストの実行時にそのプロキシ・ファイルから得られたルールを使用します。

VuGen で設定スクリプトを指定するには，次の手順を実行します。

1 ［**仮想ユーザ**］＞［**実行環境設定**］を選択し，［**インターネット プロトコル：プロキシ**］ノードを選択します。

2 ［**ユーザ定義のプロキシを使用する**］を選択し，［**自動設定スクリプトを使用する**］オプションを選択します。スクリプトの場所を指定します。

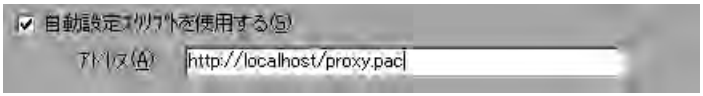


Internet Explorer（IE）で設定スクリプトを指定するには，次の手順を実行します。

1 ［**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。

2 ［**LAN の設定**］ボタンをクリックします。［LAN 設定］ダイアログ・ボックスが開きます。

3［**自動構成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。



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

本項では，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。

プロキシを設定するには，次の手順を実行します。

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

2［**インターネット プロトコル：プロキシ**］ノードをクリックします。

3 必要なプロキシ・オプションを，［**プロキシなし**］，［**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**］から選択します。

4 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。
- **pac** ファイルまたは JavaScript ファイルを使用してプロキシを指定するには，［**自動設定スクリプトを使用する**］オプションを選択してスクリプトの場所を指定します。http:// で始まる Web の場所（例えば，**http://hostname/proxy.pac**）またはファイル・サーバ上の場所（例えば，**C:\temp\proxy.pac**）を指定できます。

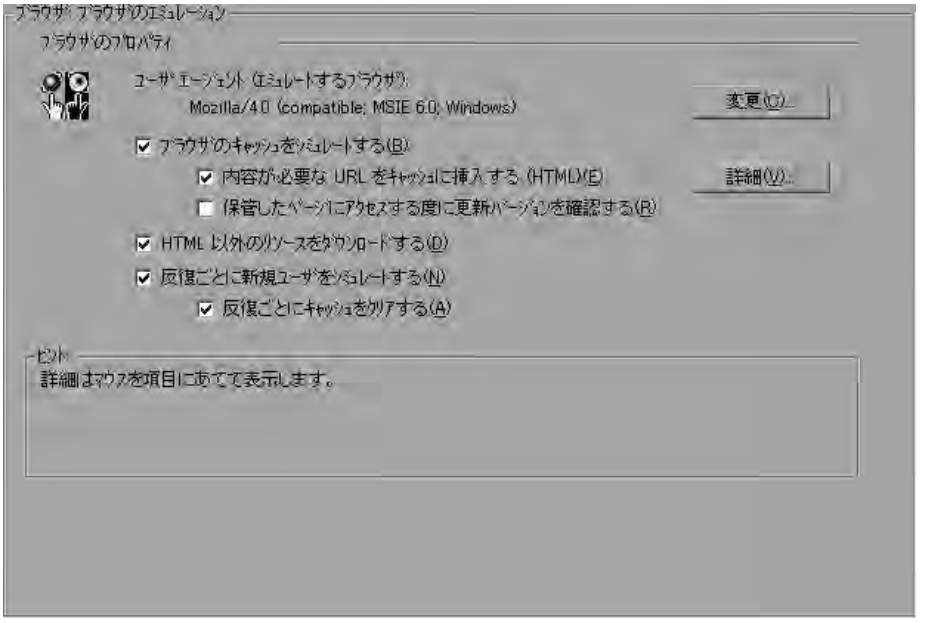
5 仮想ユーザがプロキシ・サーバを使わずに直接アクセスする URL を指定するには，［**例外**］をクリックし，それらの URL をリストに追加します。［例外］ダイアログ・ボックスでは，ローカル（イントラネット）アドレスへの直接アクセスも指定できます。

6 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。

7 セキュリティが保護されたサイト用に専用サーバを指定するのではなく，すべてのインターネット・プロトコル（HTTP，HTTPS）用に同じプロキシ・サーバを使用するよう仮想ユーザに指示するには，［**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 : ブラウザのエミュレーション**］ノードでは，チューニング環境や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

キャッシュと新しいリソースのチェックに関する詳細オプションを設定することもできます。

エミュレートするユーザ・エージェント・ブラウザ

仮想ユーザが Web サーバに要求を送信するときは，要求に 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（例えば「pclt/default.htm」），プロトコルのバージョン（例えば「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。次に例を示します。

```
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，ユーザ・エージェントは Microsoft Internet Explorer 5.5 ブラウザ・エージェントをエミュレートします。

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場合でも，仮想ユーザは各ページの画像を一度だけダウンロードします。LoadRunner や Performance Center のように複数の仮想ユーザが実行されている場合，各仮想ユーザはそれぞれのキャッシュを使用し，キャッシュから画像を取得します。このオプションを無効にすると，すべての仮想ユーザはキャッシュが利用できない状態でブラウザをエミュレートします。

次のように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 に合わ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。

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，このオプションは有効になっています。詳細については，337 ページ「内容が必要な URL をキャッシュに挿入する－詳細」を参照してください。［**ブラウザのキャッシュをシミュレートする**］を有効にした場合は，このオプションを無効にしても，VuGen によってグラフィック・ファイルが格納されます。

［**保管したページにアクセスする度に更新バージョンを確認する**］：指定した URL について，キャッシュに保存されているものより新しいバージョンがあるかどうかをブラウザがチェックするようにします。このオプションを有効にすると，VuGen によって「If-Modified-Since」属性が HTTP ヘッダーに追加されます。このオプションを有効にすると，ページの最新版が表示されるようになりますが，シナリオまたはセッションの実行時に発生するトラフィックが増えます。標準設定では，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，ページと共に記録された画像イメージと，明示的にはページと共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。

［**反復ごとにキャッシュをクリアする**］: Web ページに初めてアクセスしたユーザをシミュレートするために，反復ごとにブラウザのキャッシュをクリアします。仮想ユーザがブラウザのキャッシュに格納された情報を使用して，最近ページにアクセスしたユーザをシミュレートできるようにする場合は，チェック・ボックスをクリアしてこのオプションを無効にします。

内容が必要な URL をキャッシュに挿入する－詳細

［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：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。

複数のグループに対して詳細設定を同時に変更することはできません。グループごとに個別に設定を編集してください。

内容タイプを追加するには，次の手順を実行します。

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 などの追加内容タイプを選択します。内容名をテキスト・ボックスに入力します。

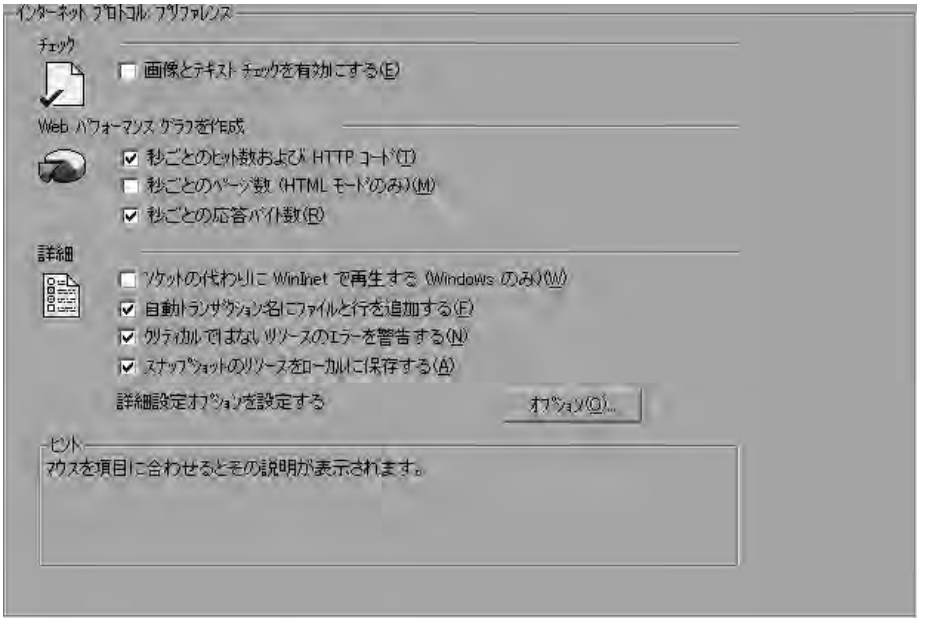
3 内容タイプをリストから削除するには，削除する内容タイプを選択してマイナス記号をクリックします。

インターネットお気に入りの設定

［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：プリファレンス**］ノードを使用して，次の項目に関連する設定を行います。

- 画像とテキストのチェック
- Web パフォーマンス・グラフの作成

➤ 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**］オプションを選択すると，仮想ユーザは再生中に **web_find** または **web_image_check** 検証関数を実行して，画像とテキストの検証チェックを行うことができます。このオプションは，HTTP モードで記録されたステートメントにのみ適用されます。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・グラフの作成

このオプションを選択すると，仮想ユーザによって Web パフォーマンス・グラフの作成に使用するデータが収集されます。テストの実行中はオンライン・モニタ，テストの実行後はアナリシスを使用して，［**秒ごとのヒット数**］，［**秒ごとのページ数**］，［**秒ごとの応答バイト数（スループット）**］のグラフを表示できます。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は，メモリを節約するためにこれらのオプションを無効にしておきます。

Web 実行環境の詳細オプション

［**ソケットの代わりに WinInet で再生する**］：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。特定の警告をエラーとみなしてテストを失敗させる場合は，このオプションを無効にできます。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，309 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。

［**スナップショットのリソースをローカルに保存する**］：VuGen はスナップショット・リソースをローカル・マシン上にファイルとして保存します。この機能を使用することで，実行時ビューアはスナップショットをより正確に作成し，よりすばやく表示できます。

インターネットお気に入りのその他のオプション

［プリファレンス］ノードの［詳細］セクションの［**オプション**］ボタンをクリックすると，各種の詳細オプションを設定できます。設定できるオプションは，［DNS キャッシュ］，［HTTP のバージョン］，［HTTP 接続の Keep-Alive］，［サーバ サイド圧縮を受け付ける］，［Accept-Language リクエスト ヘッダ］，［HTTP リクエスト接続タイムアウト］，［HTTP リクエスト受信タイムアウト］，［ネットワーク バッファ サイズ］，および［ステップ ダウンロード タイムアウト］です。GUI モード（PeopleSoft Enterprise 仮想ユーザと Oracle Applications 仮想ユーザの場合）の記録だけに適用されるオプションには，**GUI-Mode** という接頭辞が付きます。

［**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 バージョンとしてバージョン 1.0 または 1.1 のどちらを使用するかを指定します。バージョン情報は，仮想ユーザが Web サーバに要求を送信するときの HTTP 要求ヘッダ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。348 ページ「HTML 圧縮の実行」を参照してください。
- 仮想ホスティング。複数のドメイン名が同一の IP アドレスを共有します。

［**HTTP 接続の Keep-Alive**］：キープ・アライブは，持続的な接続を可能にする HTTP 拡張機能を表す用語です。こうした長時間の HTTP セッションでは，複数のリクエストを同一の TCP 接続を介して送信できます。これにより，Web サーバとクライアントのパフォーマンスが向上します。

この Keep-Alive オプションは，Keep-Alive 接続をサポートする Web サーバがあって初めて機能します。この設定により，仮想ユーザ・スクリプトを実行するすべての仮想ユーザに対して Keep-Alive HTTP 接続を有効にすることができます（標準では有効）。

［**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，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，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。対象となる内容タイプは，**HTML**，**text\html**，**TEXT**（任意のテキスト），および **ANY**（任意の内容タイプ）です。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受け入れることができることをサーバに示します。使用可能なオプションは，「**None**（圧縮なし）」，「**gzip**（gzip 圧縮を受け入れる）」，「**gzip，deflate**（gzip または deflate 圧縮を受け入れる）」，および「**deflate**（deflate 圧縮を受け入れる）」です。圧縮データを受け入れると，CPU の処理量が大幅に増えることがあります。標準設定は，「**gzip, deflate**（gzip または deflate 圧縮を受け入れる）」です。

［**Accept-Language リクエスト ヘッダ**］：受け入れた言語のカンマ区切りリストを提供します。例えば，**en-us**，**fr** などがあります。

［**警告としての HTTP エラー**］：HTTP エラーによりリソースのダウンロードが失敗した場合に，エラーの代わりに警告を発行します。

［**HTTP リクエスト接続タイムアウト（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（標準設定の値は 120 秒です）。このタイムアウトは，仮想ユーザが **wap_connect** 関数によって開始された WAP 接続を待機する時間にも適用されます。

［**HTTP リクエスト受信タイムアウト（秒）**］：仮想ユーザがステップ内の特定の HTTP 要求への応答を受信するまで待機する時間（秒単位）です。タイムアウトは，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（標準設定の値は 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，接続に関する他の問題がないか調べてください。タイムアウト時間を長くすると，スクリプトが不必要に待機することになります。

［**ステップ ダウンロード タイムアウト（秒）**］：仮想ユーザがスクリプトのステップの実行を中止するまでに待機する時間です。このオプションはページの

ダウンロードに特定秒数以上は待たないユーザの操作をエミュレートするのに使用します。

［**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。コンソールまたはコントローラで複数の仮想ユーザを実行すると，各仮想ユーザは自身のネットワーク・バッファを使用します。この設定は，主にネットワーク・バッファ・サイズがスクリプトのパフォーマンスに影響を与える可能性があると判断した上級ユーザを対象にしています。標準設定は 12 キロバイトです。

［**認証の再試行時の定数思考遅延時間（ミリ秒）**］: ユーザによる認証情報（ユーザ名とパスワード）の入力をエミュレートするため，思考遅延時間を仮想ユーザ・スクリプトに自動的に追加します。この思考遅延時間はトランザクションに含まれます（標準設定値は 0 です）。

［**ERRORS として発行されるエラー一致の上限数**］: LB（左境界）または RB（右境界）を使用して，内容チェックに ERRORS として発行されたエラーの一致数を制限します。これは，文字列が見つかったときに失敗となる（Fail=Found）一致に適用されます。以降の一致は，情報メッセージとして一覧表示されます。標準設定値は 10 です。

［**zlib ヘッダの要求**］: リクエスト・データを **zlib** 圧縮ライブラリ・ヘッダーとともにサーバに送信します。標準では，サーバに送信されるリクエストには **zlib** ヘッダーが含まれます。このオプションを使用すると，**zlib** ヘッダーがリクエストに含まれないブラウザ以外のアプリケーションをエミュレートできます。これらのヘッダーを除外するには，このオプションを「**いいえ**」に設定します（標準設定は「はい」です）。

［**統合認証を有効にする**］: Kerberos ベースの認証を有効にします。サーバによって認証スキームが提示されている場合は，別のスキームに優先して［**Negotiate**］を使用します。標準値は［**いいえ**］です。

［**KDC アドレス**］: Kerberos KDC（キー配布サーバ）のアドレス。TGS（チケット保証サービス）を取得するために使用されます。

［**AS アドレス**］: Kerberos AS（管理サーバ）のアドレス。原則に関する情報を取得する場合に使用します。

［**同じページで "META refresh" を行える最高回数**］: META 更新がページごとに実行される最大回数。標準設定は 2 です。

［**GUI モードの DOM メモリ割り当て用標準設定ブロック サイズ**］: DOM のメモリ割り当てに対して標準のブロック・サイズを設定します。この値が小さすぎると，余分な malloc 呼び出しが増え，実行時間が遅くなる場合があります。ブロック・サイズが大きすぎると，不必要に大きな領域が占有される場合があります（標準設定値は 16384 バイトです）。

［**GUI モードの単一 setTimeout/setInterval しきい値（秒）**］: window.setTimeout および window.setInterval メソッドの上限タイムアウトを指定します。遅延時間がこのタイムアウトを超えた場合，これらのメソッドは渡された関数を呼び出さなくなります。これにより，ユーザが指定した時間，次の要素をクリックするまでに待機する操作がエミュレートされます（標準設定値は 5 秒です）。

［**GUI モードの累積的 setTimeout/setInterval のしきい値（秒）**］: window.setTimeout および window.setInterval メソッドのタイムアウトを指定します。遅延時間がこのタイムアウトを超えた場合，それ以上の window.setTimeout および window.setInterval 呼び出しは無視されます。タイムアウトはステップごとに累積されます（標準設定値は 15 秒です）。

［**JavaScript エラー時に GUI モード失敗にする** ］: JavaScript の評価エラーが発生した場合に，仮想ユーザが失敗します。標準設定値は「No」で，JavaScript エラーの後に警告メッセージだけが表示され，スクリプトの実行が継続されます。

［**GUI モードで ActiveX コントロールをサポートする**］: 仮想ユーザが ActiveX コントロールを実行できるようにします（標準設定値は「No」です。）

［**GUI モードの履歴サポート**］: テスト実行での window.history オブジェクトのサポートを有効にします。［有効］，［無効］，［自動］のいずれかを選択できます。［自動］（標準設定）を選択すると，仮想ユーザは最初の反復で使用されている場合にのみ window.history オブジェクトをサポートします。このオプションを無効にすると，パフォーマンスが向上します。

［**GUI モード JavaScript の実行時のメモリ サイズ（KB）**］: JavaScript 実行時メモリのサイズをキロバイト単位で指定します（標準設定値は 256 KB）です。

［**GUI モード JavaScript のスタック メモリのサイズ（KB）**］: JavaScript スタック・メモリのサイズをキロバイト単位で指定します（標準設定値は 32 KB）です。

［**GUI モードの履歴の最大サイズ**］: 履歴リストに保存するステップの最大数（標準設定値は 10 ステップ）。

［**GUI モード ホームページ URL**］: ブラウザを開いたときに表示されるホーム・ページの URL（標準設定値は about:blank）。

［**GUI モード DOM ベースのスナップショット**］：サーバ応答の代わりに DOM からスナップショットを生成するよう VuGen に指示します（標準設定値は「はい」）。

Web サイトのフィルタリング

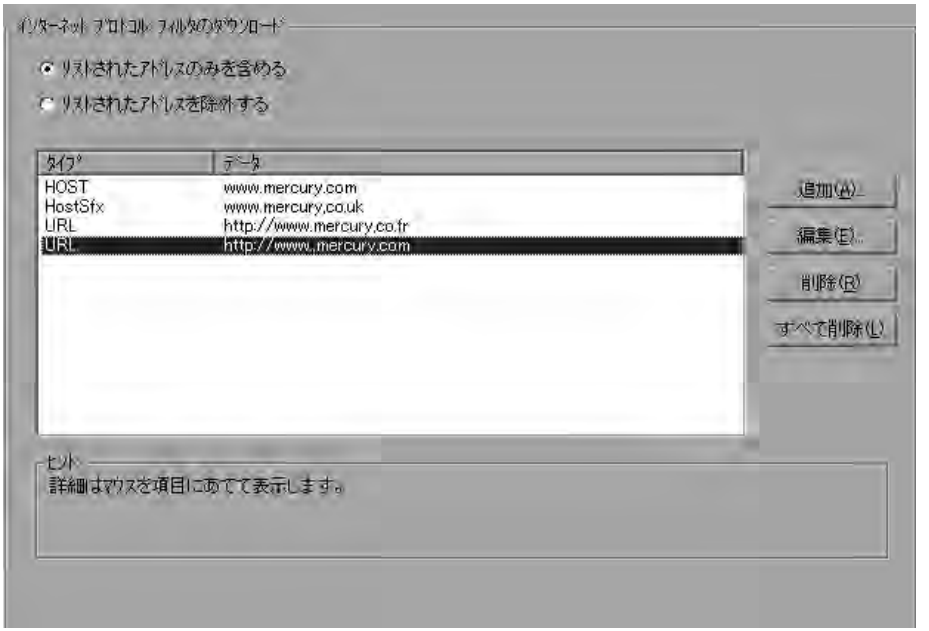
再生中における仮想ユーザのダウンロード元となる Web サイトを指定できます。除外するサイトか，追加するサイトのどちらかを指定できます。許可または禁止するリソースを，URL，ホスト名，またはホスト接尾辞を指定して制御します。

「**URL**」は，Web サイトの完全な URL アドレスで，http:// または https:// で始まります。「**ホスト**」は，ドメインを持つホスト・マシンの名前（www.mercury.com など）です。

「**Host suffix**」は，複数のホスト名に共通の接尾辞です（例：mercury.com）これは，共通のドメインに複数の Web サイトがある場合に便利です。

除外するサイトを指定した場合は，リストで指定したサイトを除くすべての Web サイトからリソースをダウンロードします。追加するサイトを指定した場

合は，［追加］リスト内のサイト以外のすべての Web サイトについて，それらからのリソースを除外します。



フィルタする Web サイトのリストを作成するには，次の手順を実行します。

1 ［**インターネット プロトコル：フィルタのダウンロード**］ノードをクリックします。

2 オプションとして，［**リストされたアドレスのみを含める**］または［**リストされたアドレスを除外する**］を選択します。

3 エントリをリストに追加します。エントリを追加するには，［**追加**］をクリックします。［フィルタの追加］ダイアログ・ボックスが開きます。

フィルタのタイプとして，［**URL**］，［**Host**］，または［**Host Suffix**］を選択し，URL などのフィルター・データを入力します。URL を入力するときは，必ず「**http://**」または「**https://**」で始まる完全な URL を入力します。［**OK**］をクリックします。

4 エントリを編集するには，エントリを選択して［**編集**］をクリックします。

5 エントリを削除するには，エントリを選択して［**削除**］をクリックします。すべてのエントリを削除するには，［**すべて削除**］をクリックします。

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［出力］ウィンドウまたはログ・ファイルに表示されます。［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，『**第 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「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，仮想ユーザが大きなトレース・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。圧縮は，自動的に，または手作業で有効にできます。

VuGen で自動的に圧縮を有効にするには，［実行環境設定］の［**インターネット プロトコル：お気に入り**］ノードを使用します。［**オプション**］をクリックして［Advanced Options］を開き，［**サーバ サイド圧縮を受け付ける**］オプションを有効にします。標準設定では，このオプションは有効になっています。詳細については，341 ページ「インターネットお気に入りのその他のオプション」を参照してください。

圧縮を手作業で追加するには，スクリプトの先頭に次の関数を入力します。

```
web_add_auto_header("Accept-Encoding", "gzip");
```

サーバによって圧縮データが送信されていることを検証するには，**Content-Encoding:gzip** という文字列が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」を参照）。

第 27 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，以下の項目について説明します。

- Web ページの内容のチェックについて
- 内容チェック実行環境の設定

以降の情報は，Web 仮想ユーザ・スクリプトを対象とします。

Web ページの内容のチェックについて

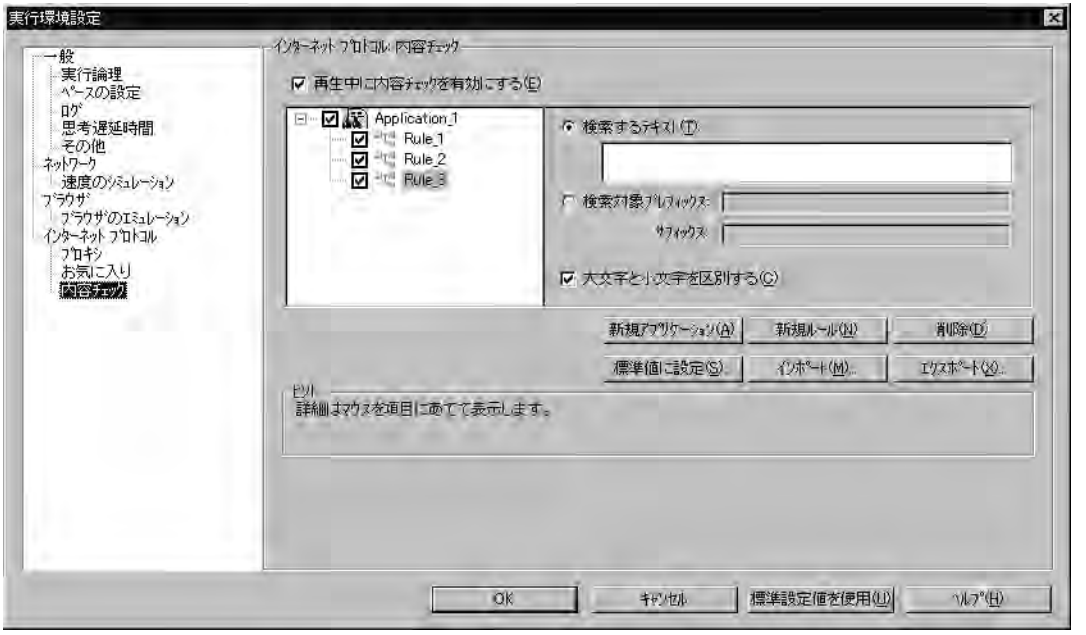
VuGen の内容チェックのメカニズムを使って，ページ内容の特定の文字列を検査することができます。このオプションは，標準的ではないエラーを検出するのに役立ちます。通常の運用では，アプリケーション・サーバで障害が発生するとブラウザが汎用の HTTP エラー・ページを表示して，エラーの性質を通知します。この標準のエラー・ページは VuGen に認識され，エラーとして処理されるので，スクリプトは失敗として報告されます。しかし，アプリケーション・サーバの中には，VuGen がエラー・ページとして認識しない固有のエラー・ページを発行するものもあります。エラー・ページはサーバによって送信され，エラーが発生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。

例えば，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。

例えば，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。

［**JavaScript 警告ボックステキストを検索**］：JavaScript 警告ボックス内のテキストのみを検索します（PeopleSoft Enterprise および Oracle Web Applications 11i 仮想ユーザのみ）。

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，他のユーザとの共有や，他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，他のユーザとの共有や，他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・ルールを定義するには，次の手順を実行します。

1 ［実行環境設定］ダイアログ・ボックスを開き，［**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。例えば，「**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 28 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。

- 負荷下の検証について
- テキスト・チェックの追加
- テキスト・チェック関数について
- 画像チェックの追加
- 追加プロパティの定義

以降の情報は，Web 仮想ユーザ・スクリプトを対象とします。

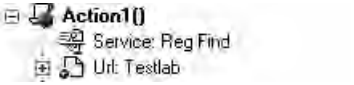
負荷下の検証について

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。オブジェクトは，テキスト文字列または画像です。

Web チェックによって，多数の仮想ユーザがアクセスしているときに Web サイトが正しく機能するか，つまりサーバが正しい Web ページを返すかどうかを確認できます。これが特に重要なのは，サイトに多数のユーザの負荷がかかるときです。大きな負荷がかかった状態では，サーバが間違ったページを返す可能性が高くなるからです。

例えば，世界の主要都市の気温に関する情報を表示する Web サイトがあるとします。VuGen を使って，その Web サイトにアクセスする仮想ユーザ・スクリプトを作成します。

仮想ユーザはサイトにアクセスし，この Web ページのテキストをチェックします。例えば，ページ内に「**Temperature**」という単語が表示されれば，チェックは合格です。サーバから正しいページが返されなかったために，「**Temperature**」という単語が表示されなければ，チェックは不合格になります。テキスト・チェックのステップは URL ステップの前に出現します。これは，VuGen が次のステップに関連する検索情報を「登録」する，つまり，事前に準備するためです。仮想ユーザ・スクリプトを実行すると，VuGen はそれに続いて Web ページのチェックを実行します。



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。

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	詳細
テキスト 	テキスト・チェック。次のアクション関数（**web_reg_find**）ステップ内またはビジネス・プロセス（**web_global_verification**）ステップ全体の中で，特定の文字列を検索します。
テキスト 	テキスト・チェック。次のアクション関数内で，**web_find** ステップを使用して特定の文字列を検索します。詳細については，360 ページ「テキスト・チェック関数について」を参照してください。
画像 	画像チェック。Web ページ内で，特定の画像を検索します。詳細については，360 ページ「テキスト・チェック関数について」を参照してください。

本章では，VuGen を使ってツリー・ビューに Web チェックを追加する方法を説明します。テキスト・ベースのスクリプト・ビューでのスクリプトへのチェックの追加については，「**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。

記録中にテキスト・チェックを追加するには，次の手順を実行します。

1 VuGen のメイン・ウィンドウまたはアプリケーションが最小化されている場合は，元のサイズに戻します。アプリケーションまたは Web ブラウザのウィンドウで，対象のテキストを選択します。

2 記録ツールバーの［**テキスト チェックを挿入**］ボタン をクリックします。VuGen によって **web_reg_find** 関数がスクリプトに追加されます。

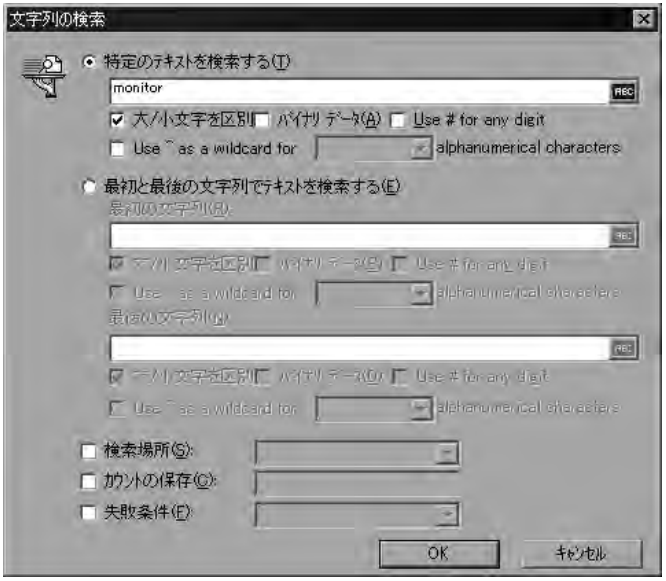
記録後のテキスト・チェックを追加するには，次の手順を実行します。

1 テキストをチェックする対象となるステップのスナップショットに移動します。

2 スナップショットの中で，検証対象テキストを選択します。

3 右クリックして表示されるポップアップ・メニューから［**テキスト チェックを追加 (web_reg_find)**］を選択します。［文字列の検索］ダイアログ・ボックスが開きます。

注 : プロトコルによっては，スナップショットからではなく［サーバの応答］タブからテキスト・チェックを追加しなければならないというメッセージが発行されます。［**サーバの応答**］タブをクリックし，［**HTML ドキュメント**］タブを選択します。本文のノードを拡張し，次に示すとおり続行します。



web_reg_find では次の属性を使用できます。

➤［**特定のテキストを検索する**］: 検索するテキスト文字列です。この属性は空でない，NULL 終端文字列である必要があります。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**］に対して特定の文字列がない場合は，次の 2 つの属性について値を入力できます。

- ［**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，NORESOURCE 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**］を指定した場合，標準設定は「None-no failure」です。［**カウントの保存**］を省略した場合，標準設定は「Not Found」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。

テキスト・チェックのプロパティを作成後に表示または変更するには，［**ツリー ビュー**］タブをクリックして新しい「**Services: Reg Find**」ステップをダブルクリックします。［テキストの検索］ダイアログ・ボックスでは，すべてのステップの属性を表示または変更できます。

テキスト・チェック関数について

テキスト・チェックを追加すると，VuGen によって **web_reg_find** 関数がスクリプトに追加されます。この関数によって，HTML ページ上でのテキスト文字列の検索が「登録」されます。登録とは，直ちに検索を実行しないで，**web_url** など，次の Action 関数の実行後にだけチェックを実行することを意味します。同時実行関数グループで作業する場合，**web_reg_find** 関数はグループ化の後にだけ実行されます。

次の例では，**web_reg_find** 関数がテキスト文字列「Welcome」を検索します。文字列が検出されない場合は，次のアクション関数が失敗し，スクリプトの実行が停止します。

```
web_reg_find("Text=Welcome", "Fail=Found", LAST);
web_url("Step", "URL=...", LAST);
```

web_reg_find 関数以外に，他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。

ほかにもいくつかの関数を使用してテキストを検索できます。

- web_find
- web_global_verification

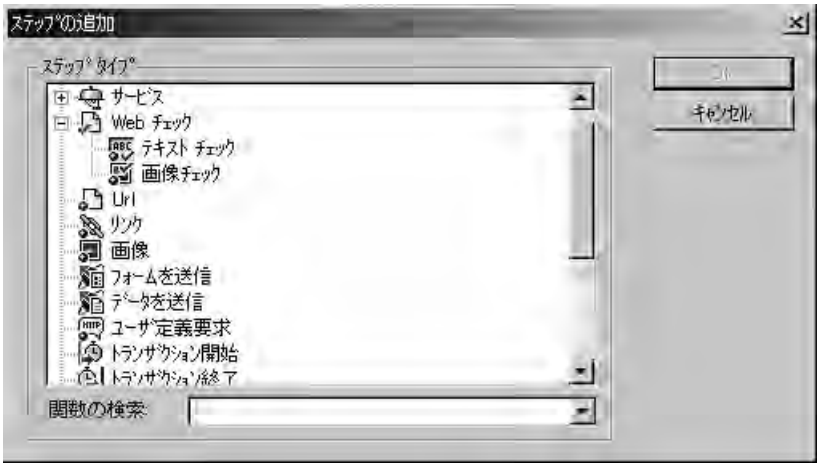
web_find 関数は下位互換性を保つためにだけ提供されているもので，HTML ベースのスクリプトに限定されている点が **web_reg_find** 関数とは異なります（［**記録オプション**］＞［**記録**］ノードを参照）。この関数にはインスタンスなどの追加の属性もあり，テキストの出現回数を決めることができます。標準のテキスト検索を実行する場合には，**web_reg_find** 関数のほうが便利です。

web_global_verification 関数を使用すると，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。

仮想ユーザ・スクリプトへの関数を追加するには，次の手順を実行します。

1 VuGen のメイン・ウィンドウで，テキスト・チェックを追加する位置をクリックします。［**挿入**］＞［**新規ステップ**］を選択します。



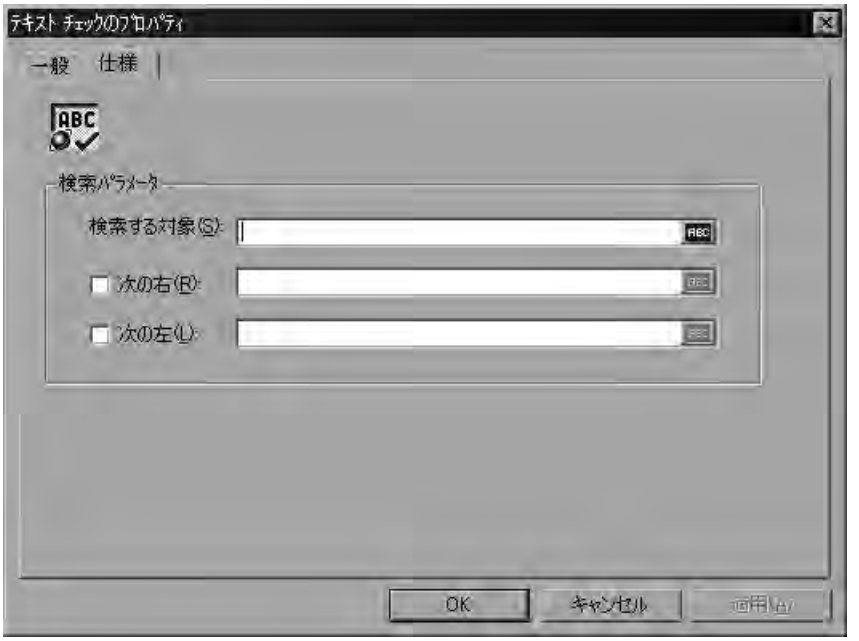
2 **web_find** 関数の場合は，［**Web チェック**］ノードを展開して［**テキスト チェック**］を選択します。**web_global_verification** 関数の場合は，［**サービス**］ノードを展開して関数名を選択します。［プロパティ］ダイアログ・ボックスが開きます。

3 これらの関数のプロパティを設定します（以降の説明を参照）。

4［**OK**］をクリックします。VuGen によって新しい関数がスクリプトに挿入されます。

web_find のプロパティの設定

web_find 関数では次のプロパティを設定できます。



［**検索する対象**］: 確認対象文字列。［ABC］アイコンは，［**検索する対象**］ボックスの文字列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の詳細については，『**第 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「VuGen パラメータを使った作業」第 8 章「VuGen パラメータを使った作業」を参照してください。

［**次の右**］または［**次の左**］: 隣接するテキストを基準とする検索文字列の位置。該当するフィールドにテキストを入力します。例えば，「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。

［**ステップ名**］: テキスト・チェックの名前。［**一般**］タブをクリックし，テキスト・チェックの名前をボックスに入力します。後で識別しやすいようにわかりやすい名前を付けます。

注：スクリプト実行時の仮想ユーザによる Web チェックの実行は，チェックが有効になっていて，スクリプトが HTML モードで実行されているときに限られます。チェックを有効にするには，［実行環境設定］ダイアログ・ボックスの［**プリファレンス**］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 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「実行環境の設定」を参照してください。

---

web_global_verification のプロパティの設定

web_global_verification 関数では次のプロパティを設定できます。

［**特定のテキストを検索する**］：存在の有無を確認する対象となる文字列。［ABC］アイコンは，［**検索する対象**］ボックスの文字列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の詳細については，『**第 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「VuGen パラメータを使った作業」を参照してください。

［**最初と最後の文字列でテキストを検索する**］：境界。テキストを囲む「**開始**」および「**終了**」文字列とも呼ばれます。［**大 / 小文字を区別**］するのか，または［**バイナリ データ**］を検索するのかに応じて，該当するオプションを選択して指定します。

［**失敗条件**］を満たす場合にスクリプトを失敗させます。失敗条件として，テキストが［**Found**］なのか（見つかったのか），または［**Not found**］なのか（見つからなかったのか）を指定することもできます。必要な動作を［**失敗条件**］ボックスで選択します。

テキスト・フラグ

登録された検索 **web_reg_find** を使用して検索テキストを指定するときは，フラグを追加して検索範囲を制御できます。

/IC は，大文字小文字を区別しないよう指定します。

/BIN は，バイナリ・データを指定します。

/DIG は，シャープ記号（#）を 1 桁の数値を表すワイルドカードとして解釈するよう指定します。DIG フラグではリテラルのシャープ記号は一致しません。

/ALNUM ＜大文字小文字＞は，キャレット記号（^）を 1 文字の US-ASCII 英数字を表すワイルドカードとして解釈するよう指定します。これには 3 つの構文があります。**ALNUMIC** は，大文字小文字を区別しません。**ALNUMLC** は，小文字にのみ一致します。**ALNUMUC** は，大文字にのみ一致します。ALNUM フラグではリテラルのキャレットは一致しません。

フラグを使用するには，属性 **TEXT** を入力し，その直後にスラッシュとフラグ名を入力します。例えば，大文字小文字を区別せずに文字列を検索するには，"Text/IC= **検索対象テキスト** '" と入力します。

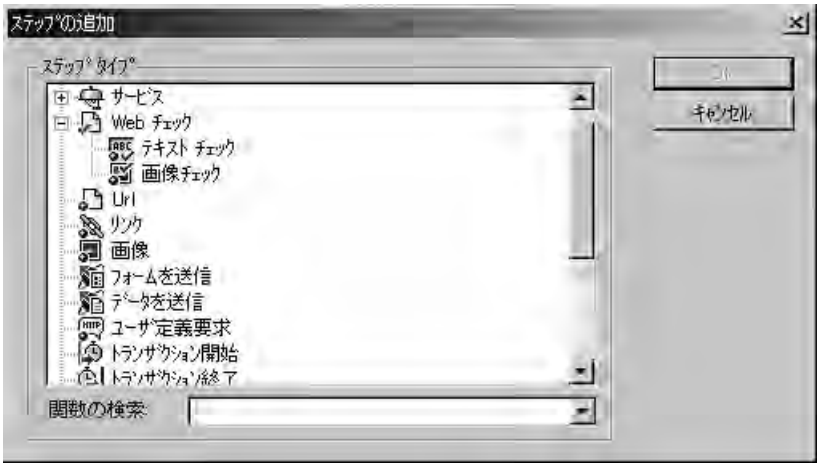
画像チェックの追加

VuGen を使って Web ページ内の画像を検索するユーザ定義チェックを追加できます。画像は ALT 属性と SRC 属性のどちらかまたは両方によって識別できます。

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。記録後は，スクリプト内の既存の任意の画像チェックを編集できます。

画像チェックを追加するには，次の手順を実行します。

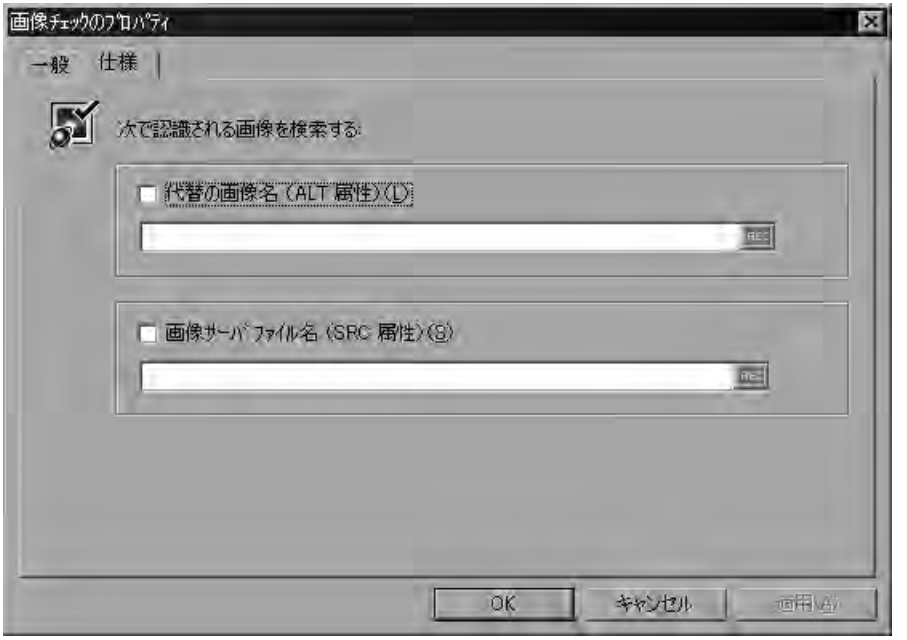
1 VuGen のメイン・ウィンドウで，チェックを実行する Web ページに対応するステップを右クリックします。ポップアップ・メニューから［**後に挿入**］を選択します。［ステップの追加］ダイアログ・ボックスが表示されます。



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の詳細については，『**第 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「パラメータの作成」を参照してください。

注 :［ALT 属性］チェック・ボックスと［SRC 属性］チェック・ボックスを両方選択した場合，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。

---

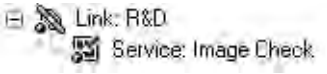
5 画像チェックに名前を付けるには，［**一般**］タブをクリックします。［**ステップ名**］ボックスに，画像チェックの名前を入力します。後で識別しやすいようにわかりやすい名前を付けます。

6 プロパティ・テーブルにチェックを定義する追加プロパティが表示されます。

［**アクティブなプロパティのみを表示する**］チェック・ボックスをクリアし，アクティブなプロパティと非アクティブなプロパティの両方を表示します。プロパティを有効にするには，プロパティ名の左のセルをクリックします。［**値**］カラムでプロパティの値を指定します。

プロパティ値の指定の詳細については，368 ページ「追加プロパティの定義」を参照してください。

7［**OK**］をクリックして設定を適用します。新しい［画像チェック］アイコンが，仮想ユーザ・スクリプト内の関連ステップに追加されます。



追加プロパティの定義

仮想ユーザ・スクリプトに挿入する各 Web チェックの追加のプロパティを指定できます。チェック・プロパティの［**一般**］タブにあるプロパティ・テーブルで追加オプションを設定します。以降の説明は **web_find** および **web_image_check** 関数のみが対象であり，**web_reg_find** 関数は対象外です。

追加プロパティを設定するには，次の手順を実行します。

1 プロパティを編集する Web チェックを右クリックして，ポップアップ・メニューから［**プロパティ**］を選択します。該当するチェックのプロパティのダイアログ・ボックスが表示されます。［**一般**］タブが表示されていることを確認します。

2［**アクティブなプロパティのみを表示する**］チェック・ボックスをクリアして，利用可能なすべてのプロパティを表示します。

3 プロパティを有効にするには，プロパティ名の左のセルをクリックします。プロパティの横に赤いチェック・マークが表示されます。

4［**値**］カラムでプロパティの値を指定します。

- ［**Frame**］: チェック対象オブジェクトの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，チェックが失敗した場合に仮想ユーザ・スクリプト全体を中止します。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，チェックに失敗したスクリプトを中止するかどうかは，実行環境の設定で定義されているエラー処理方法に従います。

標準設定の値は［**Continue**］です。エラー処理方法の定義の詳細については，『**第 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「実行環境の設定」を参照してください。

➤ ［**Expect**］:［**Not Found**］を選択すると，仮想ユーザが指定したチェック対象オブジェクトを見つけられない場合に，チェック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チェック対象オブジェクトを見つけた場合に，チェックに合格したことになります。標準設定の値は［Found］です。

➤ ［**Repeat**］:［**YES**］を選択すると，指定したチェック対象オブジェクトの複数の出現を検索します。［**NO**］を選択すると，指定したチェック対象オブジェクトが 1 つ見つかった時点で直ちにチェックが終了します。仮想ユーザ・スクリプトは次のステップから実行を続けます。このオプションは，チェック対象オブジェクトが複数含まれている可能性のある Web ページ全体を検索するときに役立ちます。標準設定の値は［YES］です。

➤ ［**Report**］:［**Always**］を選択すると，実行ログでチェック結果の詳細を常に表示できます。［**Failure**］を選択すると，チェックに不合格だった場合にのみチェック結果の詳細を表示します。［**Success**］を選択すると，チェックに合格した場合にのみチェック結果の詳細を表示します。標準設定の値は［**Always**］です。

［ABC］アイコンは，プロパティの値にパラメータが割り当てられていないことを示します。アイコンをクリックしてパラメータを割り当てます。詳細については，『**第 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「パラメータの作成」を参照してください。

第 29 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，以下の項目について説明します。

- 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
- 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 22 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。

例えば，¥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。例えば，¥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，次の手順を実行します。

1 スクリプトのツリー・ビューで，新しいステップを追加する位置の前または後のステップを選択します。

2［**挿入**］＞［**新規ステップ**］を選んで，選択したステップの後にステップを挿入するか，右クリックして表示されるメニューから［**前に挿入**］または［**後に挿入**］を選択します。［ステップの追加］ダイアログ・ボックスが開きます。

3［**ステップ タイプ**］ツリーまたは［**関数の検索**］リストから，追加するステップの種類を選択します。

4［**OK**］をクリックします。追加するステップに関する情報を入力するダイアログ・ボックスが開きます。このダイアログ・ボックスは，追加するステップの種類に応じて異なります。

このダイアログ・ボックスの使い方の詳細については，以下に示す各項を参照してください。

追加項目	参照先
仮想ユーザ API 関数	『**第 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「仮想ユーザ・スクリプトの拡張」
サービス・ステップ	394 ページ「サービス・ステップの変更」
Web チェック	395 ページ「Web チェックの変更（Web のみ）」
トランザクション	391 ページ「トランザクションの変更」
ランデブー・ポイント	393 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	393 ページ「思考遅延時間の変更」

追加項目	参照先
「URL」ステップ	375 ページ「「URL」ステップの変更」
「リンク」ステップ	377 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	379 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	381 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	385 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	388 ページ「「ユーザ定義要求」ステップの変更」
「ユーザ定義」ステップ	『**第 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「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，次の手順を実行します。

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，313 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

仮想ユーザによって，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。

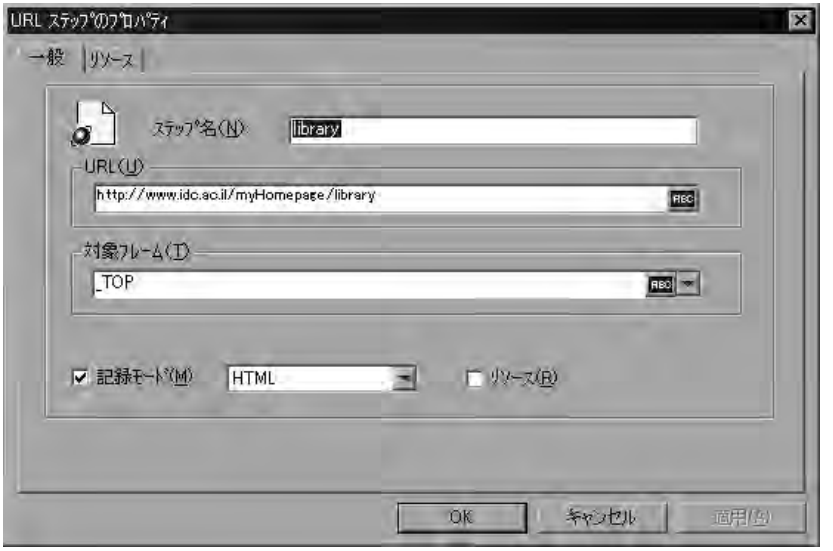
HTTP：再生時に，このステップのためにリソースをダウンロードしません。関数によって明示的に指定されたリソースだけをダウンロードします。

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。例えば，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，322 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，次の手順を実行します。

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 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「VuGen パラメータを使った作業」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 25 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，次の手順を実行します。

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 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「VuGen パラメータを使った作業」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 25 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，次の手順を実行します。

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 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「VuGen パラメータを使った作業」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）

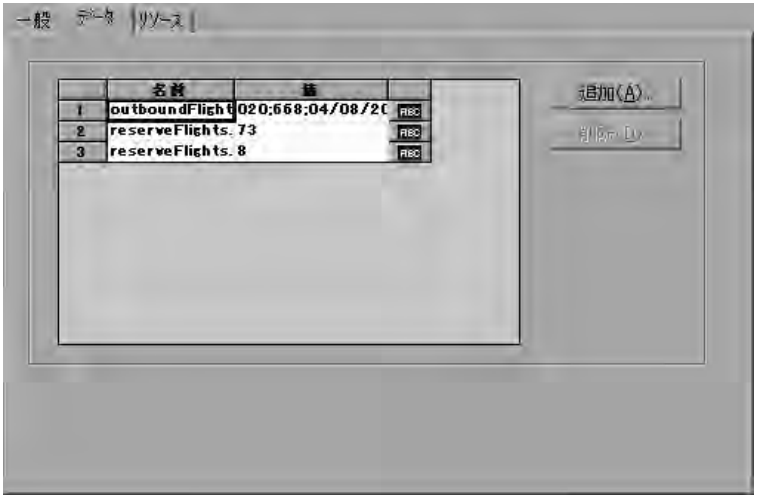
フォームを送信すると，「フォームを送信」ステップが仮想ユーザ・スクリプトに追加されます。このステップは，HTML（コンテキスト・センシティブ）モードで記録するためのオプションを選択した場合にのみ記録されます。詳細については，第 25 章「Web 仮想ユーザの記録オプションの設定」を参照してください。

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，およびステップのリソースです。

「フォームを送信」ステップのプロパティを変更するには，次の手順を実行します。

1 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**］タブをクリックします。



➤［**名前**］カラムには，フォームのすべてのデータ引数が一覧表示されます。

➤［**値**］カラムには，データ引数に対応する入力値が表示されます。

➤ タイプを示す［値］の右のカラムには，アイコンが表示されます。最初は，すべての値が定数かパラメータ化されていない値なので，アイコンは［ABC］です。『**第 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「VuGen パラメータを使った作業」で説明されているとおり，データ値にパラメータを割り当てると，［ABC］アイコンはテーブル・アイコンに変わります。

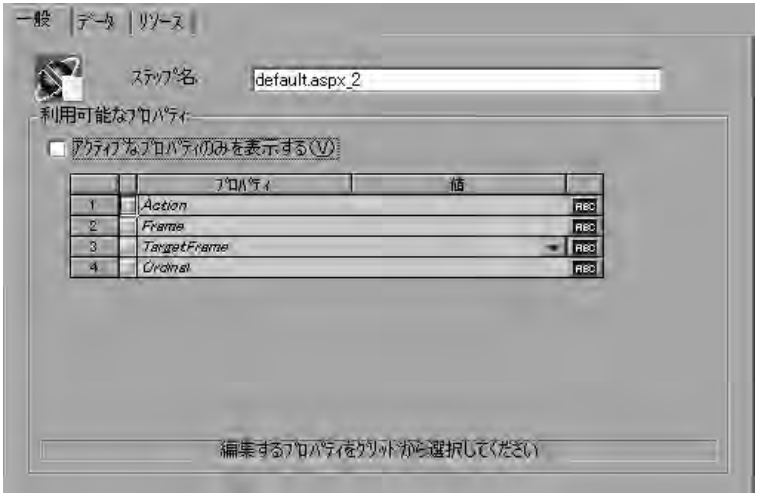
3 データ引数を編集するには，データ引数をダブル・クリックしてセル内でカーソルをアクティブにし，ボックス内に新しい値を入力します。

4 フォームの送信に新規データ引数を追加するには，［**追加**］をクリックします。［データの追加］ダイアログ・ボックスが開きます。

5 データ引数の［**名前**］と［**値**］を入力して，［**OK**］をクリックします。

6 引数を削除するには，引数を選択して［**削除**］をクリックします。

7［フォームを送信］ステップの名前を変更するには，［**一般**］タブをクリックします。



8 ステップ名を変更するには，［**ステップ名**］ボックスに新しい名前を入力します。記録時の標準の名前は，フォームの処理に使用される実行プログラム名です。

9 プロパティ・テーブルに，フォーム送信のプロパティが表示されます。

［**アクティブなプロパティのみを表示する**］オプションをクリアし，アクティブなプロパティと非アクティブなプロパティの両方を表示します。プロパティを有効にするには，プロパティ名の左のセルをクリックします。［**値**］カラムでプロパティの値を割り当てます。

- ［**Action**］：フォーム・アクションの実行に使用されるアドレス。
- ［**Frame**］：送信フォームが配置されているフレームの名前。
- ［**TargetFrame**］：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。

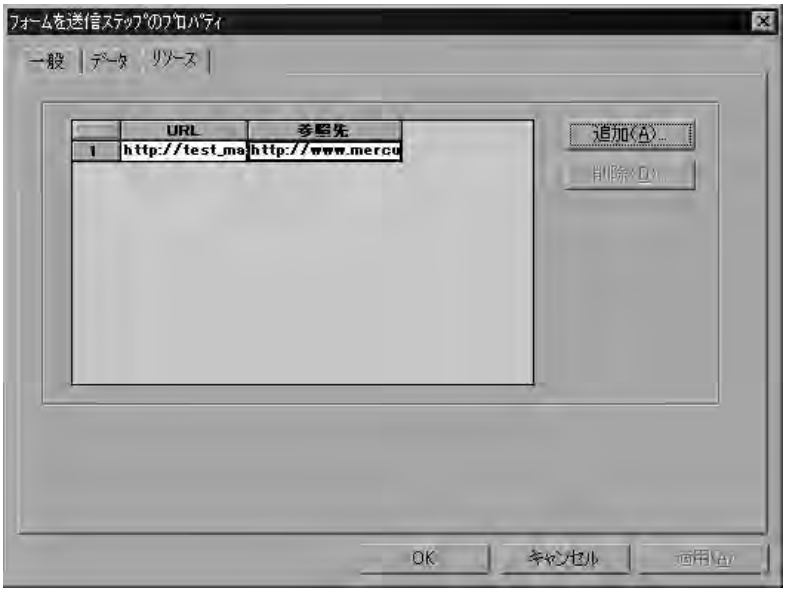
  ［**_BLANK**］：新規ウィンドウを開きます。

  ［**_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 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「VuGen パラメータを使った作業」を参照してください。

10 ステップのリソースを指定するには，［**リソース**］タブをクリックします。［**追加**］をクリックして，リソースの URL と参照ページを追加します。



11 ［**OK**］をクリックして［フォームを送信ステップのプロパティ］ダイアログ・ボックスを閉じます。

「データを送信」ステップの変更

「データを送信」ステップは，Web サイトにデータ・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，次の手順を実行します。

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**］タブをクリックします。

➤［**名前**］カラムには，フォームのすべてのデータ引数が一覧表示されます。これにはすべての隠しフィールドが含まれます。

➤［**値**］カラムには，データ引数に対応する入力値が表示されます。

➤ タイプを示す［値］の右のカラムには，アイコンが表示されます。最初は，すべての値が定数かパラメータ化されていない値なので，アイコンは［ABC］です。『**第 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「VuGen パラメータを使った作業」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。

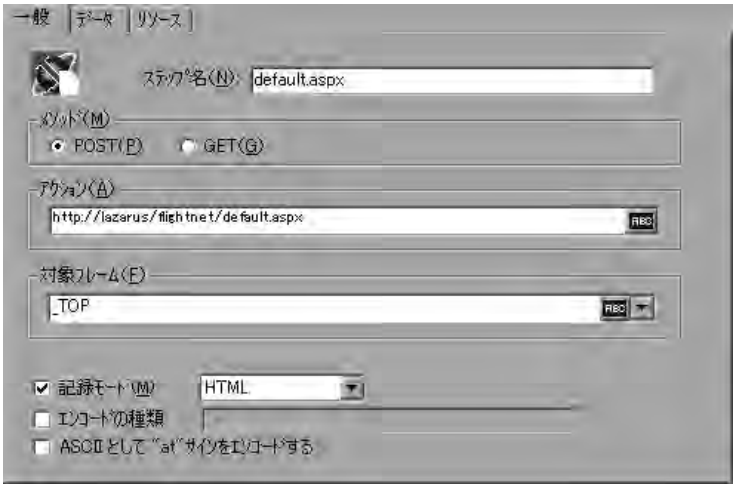
3 データ引数を編集するには，データ引数をダブルクリックしてセル内でカーソルをアクティブにし，新しい値を入力します。

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 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「VuGen パラメータを使った作業」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，375 ページ「再生モードの設定」を参照してください。

13 **multipart/www-urlencoded** など，エンコーディング・タイプを指定するには，［**エンコードの種類**］チェック・ボックスを選択して，エンコーディング方式を指定します。

14 URL 中の「@」をエンコードするには，［**ASCII として "at" サインをエンコードする**］を選択します。

15［**OK**］をクリックして，［データを送信ステップのプロパティ］ダイアログ・ボックスを閉じます。

16 ステップのリソースを指定するには，［**リソース**］タブをクリックします。［**追加**］をクリックして，リソースの URL と参照ページを追加します。

「ユーザ定義要求」ステップの変更

「ユーザ定義要求」とは，HTTP がサポートするいずれかの方法を使用した，URL に対するユーザ定義の HTTP 要求です。「ユーザ定義要求」ステップには，コンテキストはありません。

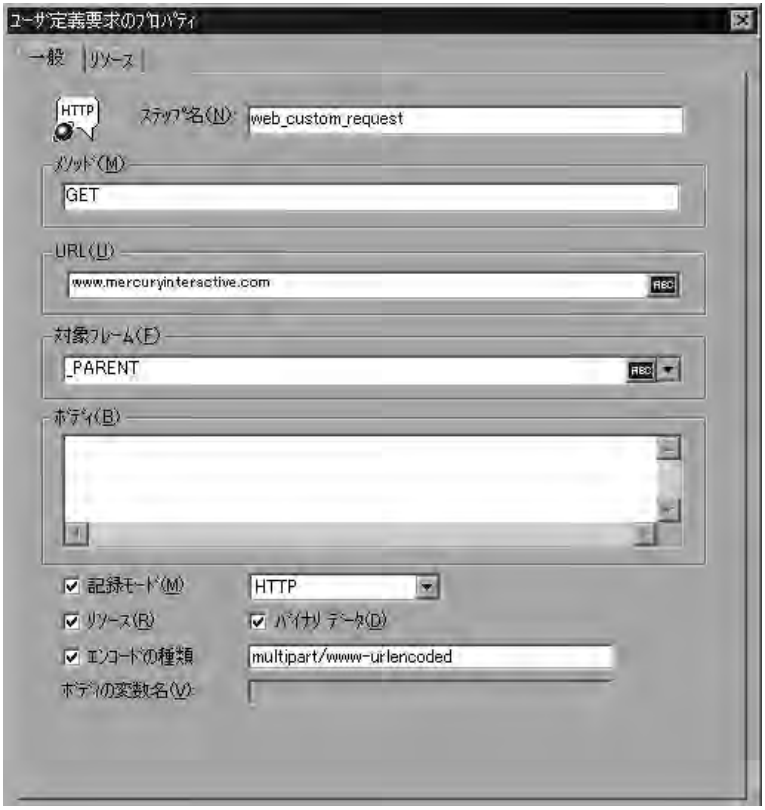
変更できるプロパティは，ステップ名，メソッド，URL，対象フレームおよび本体です。

VuGen には，ユーザ定義要求の本体を C 形式に変換する機能があります。例えば，XML ツリーまたは大量のデータをユーザ定義要求の本体領域にコピーすることで，現在の関数で使用できるように，文字列を C 形式に変換できます。これにより，必要なエスケープ・シーケンス文字が挿入され，文字列の改行が削除されます。

［ユーザ定義要求］ステップのプロパティを変更するには，次の手順を実行します。

1 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。例えば，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［**ボディ**］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［**ボディ**］ボックスに貼り付けた文字列には，テキストを選択し，右クリックして表示されるメニューから［**C フォーマットに変換**］を選択します。

9 再生モードをカスタマイズするには，［**記録モード**］オプションを選択します。次の中から使用するモードを選択します : HTML または HTTP。詳細については，375 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，テストまたはチューニング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，次の手順を実行します。

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，次の手順を実行します。

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，次の手順を実行します。

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，次の手順を実行します。

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：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 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「実行環境の設定」を参照してください。

サービス・ステップの変更

「サービス」ステップは，プロキシの設定，認証情報の送信，カスタム・ヘッダーの発行などの，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，次の手順を実行します。

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，次の手順を実行します。

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 28 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 30 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。次の項目で構成されます。

- 相関ステートメントについて
- 相関の方法について
- VuGen の相関ルールの使用
- 相関のルールの設定
- 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 および PeopleSoft Enterprise 仮想ユーザ・スクリプトを対象とします。

相関ステートメントについて

HTML ページには，動的データが含まれていることがよくあります。動的データとは，ユーザがサイトにアクセスするたびに変わるデータです。例えば，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。しばしば，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，問題が解決します。

HTML ページ内の動的データは，次の形式になっている可能性があります。

- 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，仮想ユーザは記録されたスクリプトを正しく再生できませんでした。

ステートメントを相関させることで，上の 2 つのケースにおける問題を解決できます。記録されたスクリプトの動的データを，1 つまたは複数のパラメータで置き換えます。スクリプトを実行すると，各パラメータに値が割り当てられます。

相関の方法について

本章では，組み込みルールまたはユーザ定義のルールを使った自動相関について説明します。ステートメントを手作業で相関する場合や，ワイヤレスの仮想ユーザ・スクリプトを実行する場合は，422 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，313 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 VuGen を設定できます。本章では，記録時のソリューションとして，スクリプト内のステートメントを自動的に相関させます。また，VuGen のスナップショット相関機能を使って，記録後にスクリプトを相関させることもできます。記録後の相関の詳細については，第 31 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，402 ページ「一致条件の追加」および 403 ページ「詳細相関ルール」を参照してください。

組み込み相関

組み込み相関では，サポートされているアプリケーション・サーバを対象に動的データを検出し，相関させることができます。ほとんどのサーバには，リンクおよび参照の作成時に必ず使用される，明確な構文ルール，つまり「コンテキスト」があります。

例えば，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ひととおり実行できます。詳細については，第 31 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

標準設定では，ルール用に保存できる文字列の最大サイズは 4096 文字です。必要ならば，Windows インストール・ディレクトリにある **CorrelationSettings.xml** ファイルの **MaxParamLen** 属性の値を増やすことによって，この値を変更できます。

本体テキスト全体

［**本体テキスト全体の中でパラメータを検索します**］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，本体の全体を探します。テキストは，指定した境界を使って検索されます。

リンク / フォーム・アクション

［**リンクおよびフォームのアクションの中でパラメータを検索します**］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分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。

例えば，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。

例えば，Web ページに次の形式のリンクが含まれていたとします。

"SessionID=122@page.htm"
"Page.htm@SessionID=122&test.htm"

右の境界が一定でないので（「@」の場合と「&」の場合がある），右の境界として 1 つを指定するだけでは不十分です。この場合，代わりに使用する右の境界として「&」を指定します。

左と右の境界は，文字列を一意に特定するものでなければなりません。境界に動的データを含めることはできません。また，ドロップダウン・メニューの選択肢として用意されている［**文字列終端**］または［**改行文字**］を，右の境界として指定することもできます。

このオプションでは，左右の境界はスクリプトに含まれている文字列内に必ず含まれている必要があります。サーバが境界を返すだけでは十分ではありません。この制限は他のアクション・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検索します**］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，自動的にパラメータに置換するため，スクリプト内で境界を表示する必要がありません。

フォーム・フィールド値

［**フォーム フィールド値をパラメータ化します**］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 関数を入力する際使用するテキストです**］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。例えば，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。ハッシュ記号はワイルドカードとして機能し，任意の数字を含むテキスト文字列を検索できます。例えば，このオプションを有効にし，左の境界として「**Mercury###**」を指定した場合は，「**Mercury193** 」や「 **Mercury284**」が有効な一致文字列として検索されます。

コメントの追加

わかりやすいコメントをスクリプト内の相関ステップに挿入するように VuGen を設定できます。このオプションを有効にするには，［**スクリプトにコメントを追加**］オプションを選択します。

詳細相関ルール

VuGen では，次の詳細相関ルールも指定できます。

［**常に新規のパラメータを作成する**］：パラメータに置換された値が，前のインスタンスから変わっていない場合も，このルールに応じて新しいパラメータを作成します。Web サーバがページごとに異なる値を割り当てる場合には，このオプションを設定します。例えば，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。

例えば，フォーム送信の際，VuGen によって境界 aaa と bbb の間の 1234 という文字群が記録され，aaa1234bbb となったとします。以後このフォームを送信する際，VuGen は 1234 という文字列を見つけると（つまり，Name=1234 の場合），記録された値をパラメータで置き換える処理のみ実行します。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（例えば，Name=12345）が含まれている場合でも，VuGen は，その値をパラメータで置き換えずに，12345 という値を使用します。

［**逆方向検索**］：文字列の最後から左境界を検索します。

［**左の境界インスタンス**］：一致として考えられる，文字列内（ボディではなく）の左境界の一致件数です。

［**オフセット**］：一致した値の部分文字列の開始位置を指定するオフセットです。部分文字列がパラメータに保存されます。標準設定は，一致した文字列の最初の部分です。必ず負数以外を指定してください。

［**長さ**］：パラメータに保存する，一致した文字列の部分文字列の，オフセットからの長さです。このオプションを無効にすると，標準の値が使用され，指定したオフセットから一致文字列の末尾までの文字列が保存されます。

［**代替の右境界**］：あらかじめ指定されている右境界が見つからなかった場合の代替条件です。テキスト，**文字列の末尾**，または**改行文字**を指定できます。

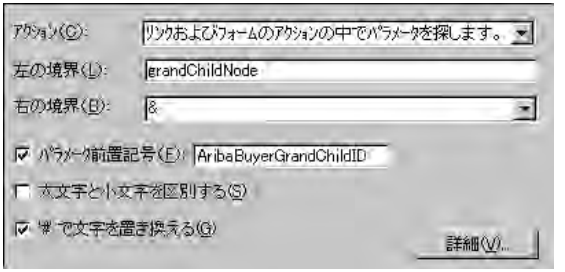
相関のルールの設定

［相関］記録オプションを使用して，ルールを追加，変更または削除できます。アプリケーション・サーバの環境を対象に自動的に作成されたルールを編集することもできます。

記録前に記録オプションを使用してルールを作成するだけでなく，記録後にルールを作成することもできます。スクリプトを実行した後，ルールを対象に相関をスキャンします（CTRL キーを押しながら F8 キーを押します）。相関結果の 1 つを選択し，そのプロパティに基づくルールを作成します。詳細については，418 ページ「相関の検索の実行」を参照してください。

相関のルールを定義するには，次の手順を実行します。

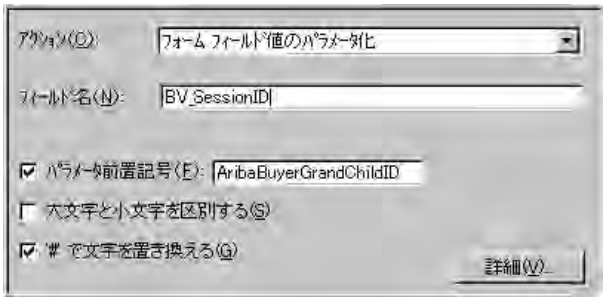
1 既存のルールをクリックするか，表示枠の下の［**新規ルール**］をクリックします。右側の表示枠に相関ルールが表示されます。



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，本体全体，フォーム・フィールド，または web_reg_add_cookie から選択します。

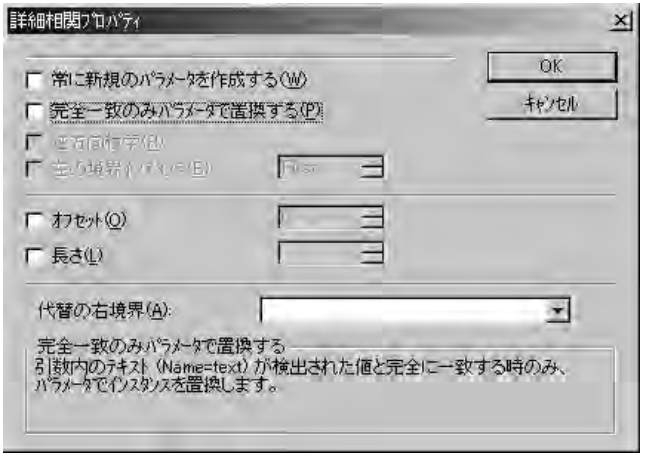
3 最初の 3 種類では，［**左の境界**］と［**右の境界**］ボックスでデータの境界を指定します。

4 フォーム・フィールド・タイプのアクションの場合は，フィールド名を指定します。



5［**大文字と小文字を区別する**］および［**パラメータ前置記号**］のいずれか一方，または両方のオプションを指定します。パラメータの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，［相関］ノードの［**詳細**］をクリックします。［詳細相関プロパティ］ダイアログ・ボックスが開きます。



➤［**常に新規のパラメータを作成する**］を選択すると，パラメータによって置き換えられる値が前のインスタンスから変わっていない場合も，このルールに応じて新しいパラメータを作成します。

- ➤［**完全一致のみパラメータで置換する**］を選択すると，テキストが検出された値に正確に一致する場合にだけ，その値をパラメータで置換します。
- ➤［**逆方向検索**］を選択すると，逆方向に検索します。
- ➤［**左の境界インスタンス**］ボックスを選択して，必要なインスタンスを指定します。
- ➤［**オフセット**］を選択して，一致した文字列内の文字列のオフセットを指定します。
- ➤［**長さ**］を選択して，一致した文字列の何文字までパラメータに保存するかを指定します。このオプションは［**オフセット**］オプションと組み合わせて使用することができます。
- ➤［**代替の右境界**］ボックスで別の右境界を指定するか，ドロップダウン・メニューで［**ページの最後**］または［**改行文字**］を選択します。

7［**テスト**］をクリックして，定義したルールをテストします。詳細については，407 ページ「ルールのテスト」を参照してください。

8［**OK**］をクリックしてルールを保存し，ダイアログ・ボックスを閉じます。

ルールのテスト

本項の内容は，コンテキストが知られているサーバを対象に作成したユーザ定義のルールを対象とします。[相関] パネルで新しいルールを定義したら，セッションを記録する前に，サンプルの文字列に対してルールを適用することによってテストを実行できます。[トークン置換テスト パッド] を使用してルールをテストします。テスト・パッドを使用するには，次の手順を実行します。

1 左側の表示枠でルールを選び，[**テスト**] をクリックします。[トークン置換テスト パッド] ダイアログ・ボックスが開きます。

2 [**置換用のソース テキスト**] ボックスにテキストを入力します。

3 [**テスト**] をクリックします。

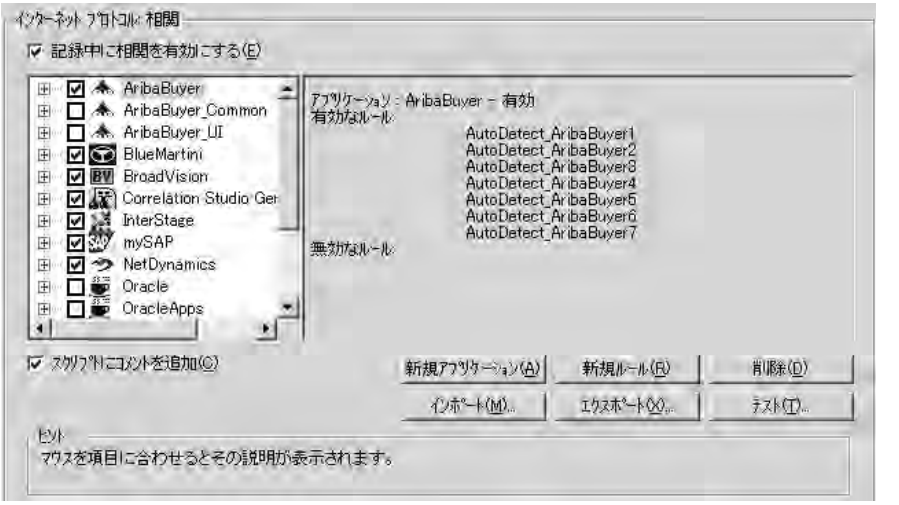
置換が行われた場合，パラメータ化されたソース・テキストは [**置換結果**] ボックスで確認でき，その置換に適用されたルールのリストは [**適用されたルール**] ボックスで確認できます。

相関記録オプションの設定

VuGen 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，次の手順を実行します。

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，404 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，VuGen 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，削除するアプリケーションやルールを選択して［**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 から確認メッセージが表示されます。

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。

第 31 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，次の項目について説明します。

- ステートメントの相関について
- 相関結果タブの表示
- 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，SAP Web，および Siebel 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 30 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，記録中および再生中に Web ページのスナップショットをキャプチャします。記録時のスナップショットと再生時のスナップショットを比較することによって，スクリプトの実行を成功させるために相関させる必要のある値を特定します。記録時および再生時のスナップショットの詳細については，21 ページ「スナップショットについて」を参照してください。

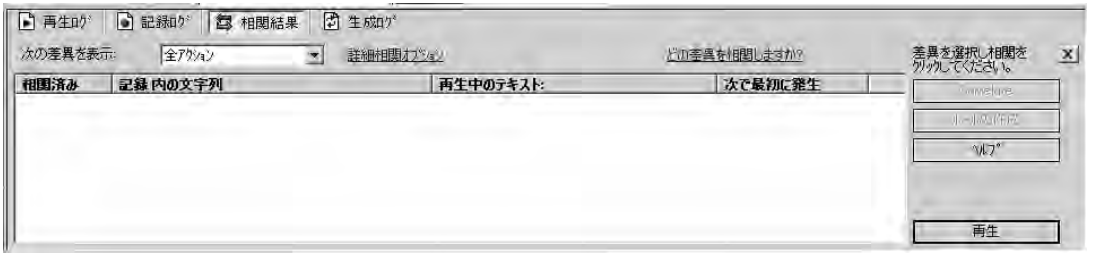
Web 相関メカニズムには，スナップショット間のテキストまたはバイナリの差異を表示できる，比較ユーティリティが組み込まれています。比較の後，差異を 1 つずつ相関させることも，一度にすべてを相関させることもできます。

VuGen の相関メカニズムでは十分でない場合や，こうしたメカニズムをサポートしないプロトコル（ワイヤレスや手作業による相関など）に対しては，手作業による相関を使用します。詳細については，422 ページ「手作業による相関」を参照してください。

相関結果タブの表示

［相関結果］タブには，記録時のスナップショットと再生時のスナップショットの差異が表示されます。

スクリプト内を検索して相関を見つけるように指示すると，［出力］ウィンドウが開き，［**相関結果**］タブに記録時のスナップショットと再生時のスナップショットの差異が表示されます。



［**次の差異を表示**］リスト・ボックスから対象を選択することによって，スクリプト内のすべての差異を表示したり，現在のステップまたはアクションの差異だけを表示したりできます。

相関され差異が発見された項目には，［**相関済み**］カラムにチェック・マークが付きます。その右にある［**記録内の文字列**］と［**再生中のテキスト**］の 2 つのカラムには，スナップショット間のテキストの差異が表示されます。その右の［**次で最初に発生**］カラムには，相関が最初に検出されたアクションが表示されます。

スナップショット間の差異を検出したら，相関を選択して［**Correlate**］をクリックすることにより，差異を一度に 1 つずつ相関させます。また，［**Remove Correlation**］ボタンを使用して特定の相関を取り消すこともできます。検出された相関のいずれかをその後の記録で発生させるには，新しい相関ルールを作成します。ルールを作成することにより，記録中に差異を認識し，自動的に相関させることができます。詳細については，413 ページ「ルールの作成」を参照してください。

このメカニズムを使って値を相関させると，VuGen は **web_reg_save_param** 関数と，パラメータを対象に相関が行われたことを示すコメントをスクリプトに挿入します。コメントには，元の値も示されます。

```
// [WCSPARAM WCSParam_Diff1 14 reserveFlights] Parameter
{WCSParam_Diff1} created by Correlation Studio
    web_reg_save_param( "WCSParam_Diff1", "LB= NAME=¥"",
"RB=¥"", "Ord=5", "Search=Body", "RelFrameId=1", LAST );
    web_submit_form("reservations.pl",
        "Snapshot=t4.inf",
        ITEMDATA,
        "Name=depart", "Value=Denver", ENDITEM,
        "Name=departDate", "Value=06/25/2004", ENDITEM,
        "Name=arrive", "Value=Los Angeles", ENDITEM,
        "Name=returnDate", "Value=06/26/2004", ENDITEM,
        "Name=numPassengers", "Value=1", ENDITEM,
        "Name=roundtrip", "Value=<OFF>", ENDITEM,
        "Name=seatPref", "Value=None", ENDITEM,
        "Name=seatType", "Value=Coach", ENDITEM,
        "Name=findFlights.x", "Value=44", ENDITEM,
        "Name=findFlights.y", "Value=12", ENDITEM,
        LAST);
    lr_think_time(12);
```

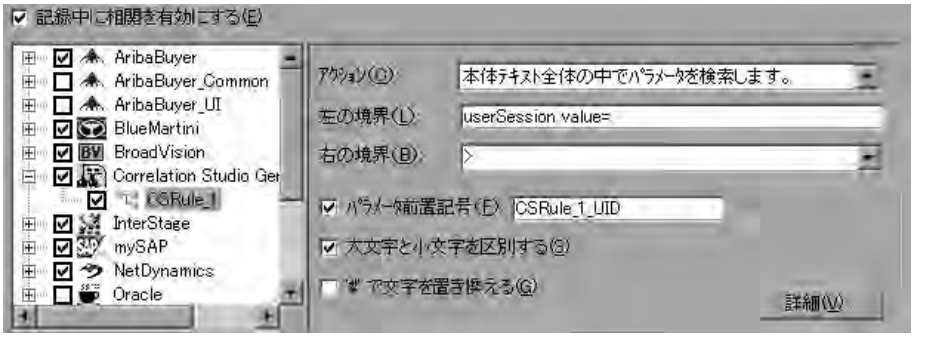
ルールの作成

相関結果のリストから直接ルールを作成できます。ルールを作成することにより，記録中に差異を認識し，自動的に相関させることができます。

検出された相関からルールを作成するには，次の手順を実行します。

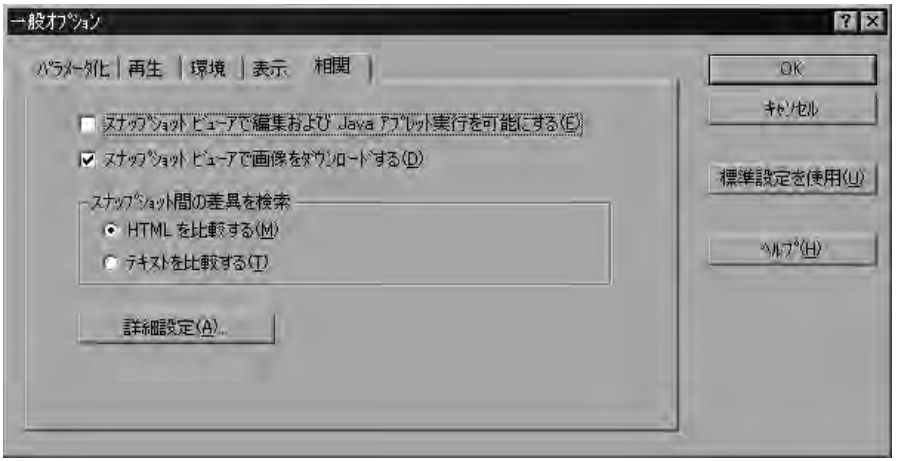
相関を選択し，［**ルールの作成**］をクリックします。相関を選択し，右クリックして表示されるメニューから［**相関ルールを作成**］を選択してルールを作成することもできます。

このルールが**相関**ルールのリストに追加されます。このルールは，［記録オプション］の［相関］ノードに表示されます。次の例では，ルールが CSRule_1 として追加されています。



VuGen の相関の設定

［一般オプション］でグローバルな相関の設定を行います。これらのオプションを設定すると，仮想ユーザは後で使用できるように，再生中に相関情報を保存します。スナップ・ショットを比較する際に，HTML またはテキストのどちらを比較するかを指定できます。［詳細相関］ダイアログ・ボックスでは，区切り文字として扱う文字も指定できます。



［**スナップショット ビューアで編集および Java アプレット実行を可能にする**］：スナップショット・ウィンドウでアプレットと JavaScript を実行できるようにします。多量のリソースを使用するため，このオプションは標準設定では無効になっています。

［**スナップショット ビューアで画像をダウンロードする**］：画像をスナップショット・ビューアに表示するよう VuGen に指示します。ビューアでの画像の表示が遅い場合は，このオプションを無効にすることもできます。標準設定では，このオプションは有効になっています。

［**スナップショット間の差異を検索**］：比較の方法を選択します。

- ［**HTML を比較する**］：HTML コードの差異のみを表示します。
- ［**テキストを比較する**］：すべてのテキスト，HTML，およびバイナリの差異を表示します。

注： ほとんどの場合，標準の HTML を比較する方法を使用することをお勧めします。スクリプトに HTML タグ以外のタグが含まれている場合は，テキストを比較する方法を使用できます。

［**詳細設定**］：［詳細相関］ダイアログ・ボックスを開きます。

［詳細相関］ダイアログ・ボックス

このダイアログ・ボックスでは，区切り文字として扱う文字を指定できます。

［**区切り文字として扱う文字**］：1 つまたは複数の標準設定以外の区切り文字を指定します。

［**追加区切り文字**］：復帰文字，復帰改行，タブ文字など，標準設定の区切り文字を指定できます。設定を変更するには，区切り文字の横のチェック・ボックスをクリアします。

［**X 文字以下の差異を無視する**］：相関を行うしきい値を指定できます。VuGen は記録時の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。

［**大きな相関に警告を発行**］：サイズが 10 KB 以上の文字列を相関させようとしたときに警告を表示します。

相関オプションの設定

セッションの記録を開始する前に，相関オプションを設定します。

相関のオプションを設定するには，次の手順を実行します。

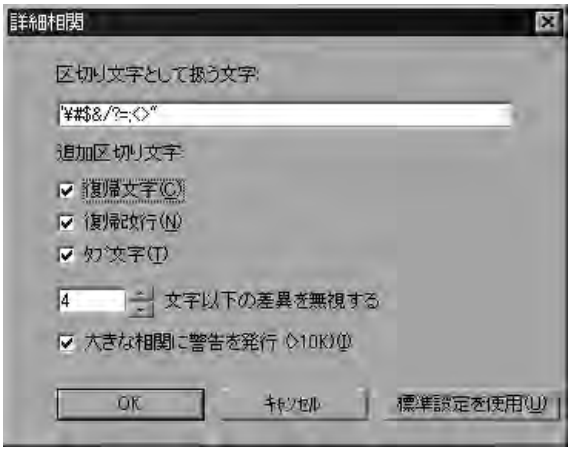
1 ［**ツール**］＞［**一般オプション**］を選び，［**相関**］タブを選択します。

2 ［**スナップショット ビューワで編集および Java アプレット実行を可能にする**］を選択して，スナップショット・ウィンドウでアプレットと Java スクリプトを実行できるようにします。

3 画像をスナップショット・ビューアに表示するよう VuGen に指示するには，［**スナップショット ビューアで画像をダウンロードする**］オプションを選択します。

4 比較の方法として，［**HTML を比較する**］または［**テキストを比較する**］（HTML の要素以外に対してのみ）を選択します。

5 区切り文字を設定するには，［**詳細設定**］をクリックして，［詳細相関］ダイアログ・ボックスを開きます。



6［**区切り文字として扱う文字**］ボックスで，区切り文字として扱うすべての文字を指定します。

7［**追加区切り文字**］セクションで必要なオプションを選択して，標準で使用する 1 つまたは複数の区切り文字を指定します。

8［**X 文字以下の差異を無視する**］ボックスで，相関のしきい値を指定します。VuGen は記録時の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。

9［**大きな相関に警告を発行**］には，このオプションのチェック・ボックスを選択します。

10［**OK**］をクリックして詳細相関設定を受け入れ，ダイアログ・ボックスを閉じます。

11［一般オプション］ダイアログ・ボックスの［**OK**］をクリックし，［相関］の設定を受け入れてダイアログ・ボックスを閉じます。

相関の検索の実行

VuGen のスナップショット・ウィンドウを使って，スクリプト内のどの値が動的で相関が必要かを判断します。次の項では，スクリプト内を自動的に検索して差異を見つける方法と，VuGen を使って必要な相関を行う方法について説明します。

スクリプト内を検索して相関を実行するには，次の手順を実行します。

1 スクリプトを開き，ツリー・ビューで表示します（[**表示**] > [**ツリービュー**]）。スナップショットを表示します（[**表示**] > [**スナップショット**] > [**スナップショットを表示**]）。

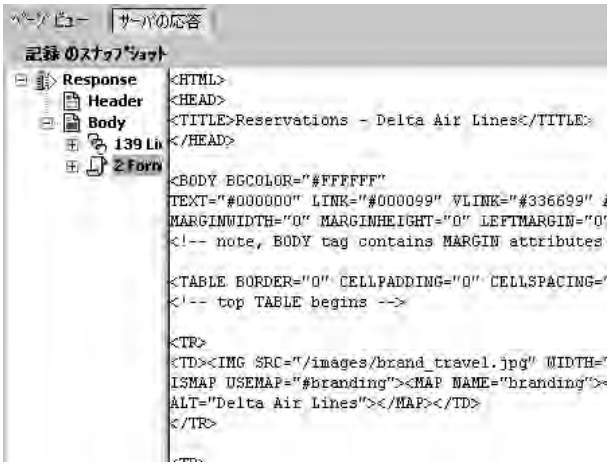
2 左側の表示枠のツリー・ビューでスクリプトのステップを選択します。スナップショットが右側の表示枠に表示されます。

3 記録時のスナップショットと最初の再生時のスナップショットを両方とも表示するには，[**表示**] > [**スナップショット**] > [**記録と再生済み**] をクリックします。

4 2 回目以降の再生時のスナップショットを使用するには，［**表示**］＞［**スナップショット**］＞［**反復の選択**］をクリックします。［テスト結果を選択］ダイアログ・ボックスが開き，スナップショット・ファイルが格納されているフォルダが表示されます。通常，スクリプトのフォルダの下には **result** フォルダと **Iteration** フォルダが表示されます。

5 スクリプトのサブ・フォルダ以外のフォルダにあるスナップショット・ファイルを選択するには，［**フォルダを選択**］をクリックします。目的のフォルダの場所を見つけ，［**OK**］をクリックします。

6 HTML コードを表示するには，［**サーバの応答**］タブをクリックします。「Body」の分岐を開きます。



ページ・ビューに戻るには，［**ページ ビュー**］タブをクリックします。

7［**仮想ユーザ**］＞［**アクション内で相関をスキャン**］を選択するか，［**Find Correlations**］ボタンをクリックします。VuGen はスクリプト内を検索し，相関が必要な動的データを見つけ，それらを［相関結果］タブに表示します。

8［**次の差異を表示**］リスト・ボックスで，すべての差異を表示するか，フィルタ方法を選択します。［**全アクション**］，［**現在のアクション**］，または［**現在のステップのみ**］を選択できます。

相関させる差異の決定

差異のリストを生成したら，相関させる差異を決定する必要があります。相関させる必要がない差異を誤って相関させると，再生に悪影響を与える可能性があります。

相関を必要とする可能性が最も高いのは，次の文字列です。

- **ログイン文字列** - セッション ID やタイム・スタンプなどの動的なデータを含むログイン文字列。
- **日付やタイム・スタンプ** - 日付，タイム・スタンプ，またはその他のユーザ・アカウント情報を使った文字列。
- **共通の接頭辞** - 文字列の前に付く **SessionID** や **CustomerID** などの共通の接頭辞。

相関させる必要がある差異かどうかが不明な場合は，その差異だけを相関させてスクリプトを実行し，問題が解決したかどうかを再生ログで確認します。

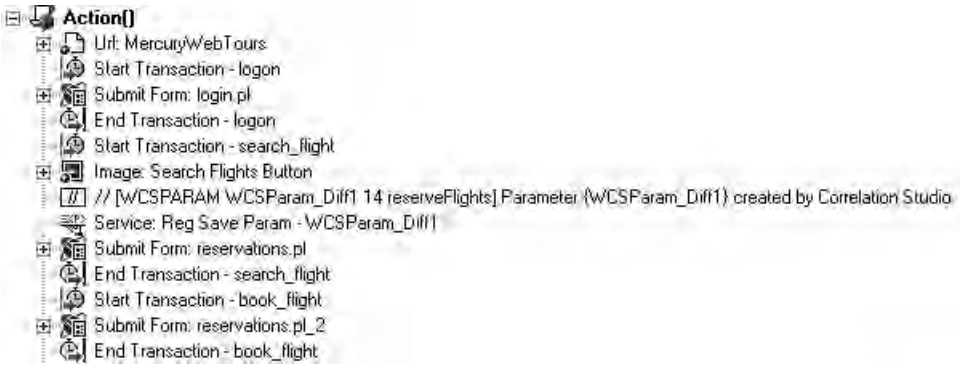
また，記録された文字列と再生された文字列の一部だけが異なる場合も，差異を相関させる必要があります。例えば，SessionID 文字列の接頭辞と接尾辞が同じでも，中間の文字が異なる場合は相関させる必要があります。

差異を相関させる必要があることを確認したら，それを相関させるように VuGen を設定します。

差異を相関させるには，次の手順を実行します。

1［相関］タブに差異を表示し，相関させる差異を選択します。一度に相関させる差異は 1 つだけにするをことをお勧めします。

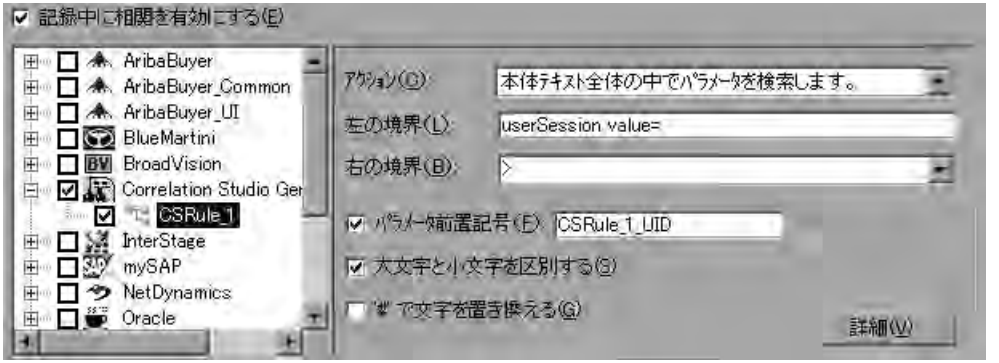
2［**相関**］をクリックします。相関された差異の横に緑のチェック・マークが付けられ，スクリプトに **web_reg_save_param** 関数が挿入されます。



相関させるすべての差異について，この手順を繰り返します。

3 検出された相関からルールを作成するには，相関を選択して［**Create Rule**］をクリックします。これは，右クリックして表示されるメニューからも選択できます。VuGen は，ルールが作成されたことを確認するメッセージを発行します。

このルールを表示するには，［記録オプション］（CTRL +F7）を開き，［**相関**］ノードを選択します。「**Correlation Studio**」エントリを拡張し，ルールを選択します。



4 相関を取り消すには，差異を選択して［**相関を削除**］をクリックします。

5［**ファイル**］＞［**上書き保存**］を選択して，スクリプトへの変更を保存します。

手作業による相関

Web 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。

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・スクリプトについては，手作業でスクリプトを相関させることができます。スクリプトを手作業で相関させるには，コード相関関数を追加します。パラメータにデータを動的に保存する関数は，**web_reg_save_param** です。

スクリプトを実行すると，**web_reg_save_param** 関数はそれ以降にアクセスする HTML ページ内を検索します。左と右の境界を指定すると，VuGen によってこれらの境界の範囲内でテキストが検索されます。テキストが見つかると，テキストはパラメータに保存されます。

関数の構文は次のとおりです。

int **web_reg_save_param** (const char ***mpszParamName**, **＜属性のリスト＞**, LAST);

使用できる属性を次の表に示します。属性値の文字列（Search=all など）では，大文字と小文字は区別されません。

NotFound	境界が見つからず，空の文字列が生成されたときの処理方法。標準設定の「ERROR」では，境界が見つからない場合にエラーが発行されます。「EMPTY」に設定した場合，エラー・メッセージは発行されず，スクリプトの実行が継続されます。スクリプトに対して［エラーでも処理を継続する］を有効にしている場合は，NOTFOUND を「ERROR」に設定していても，境界が見つからない場合にスクリプトが継続されます。ただし，エラー・メッセージは詳細ログ・ファイルに記録されます。
LB	パラメータまたは動的データの左の境界。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。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，次の手順を実行します。

1 動的データを含むステートメントと，データの境界を示すパターンを探します。427 ページ「動的文字列の境界の定義」を参照してください。

2 スクリプトの中で，動的データを独自のパラメータ名に置き換えます。詳細については，下記を参照してください。

3 スクリプト内の動的データが含まれるステートメントの前に **web_reg_save_param** 関数を追加します。詳細については，424 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

記録されたステートメントから実際の動的データを特定します。次に，スクリプト全体からその動的データを検索し，パラメータに置き換えます。パラメータに名前を付け，{param_name} のように中括弧で囲みます。1 つのスクリプトには，最大 64 個のパラメータを設定できます。

動的データをパラメータに置き換えるには，次の手順を実行します。

VuGen のメイン・ウィンドウで［**編集**］>［**置換**］を選択して，［検索と置換］ダイアログ・ボックスを表示します。動的データをスクリプト全体から検索して，パラメータに置き換えます。

相関関数の追加

スクリプトの動的データを保存するには，**web_reg_save_param** ステートメントを挿入します。この関数は，再生中に動的データの実行時の値を保存するパラメータを作成するように VuGen に指示します。

スクリプトを実行すると，**web_reg_save_param** 関数はそれ以降にアクセスする HTML ページ内を検索します。左の境界，任意の文字列，右の境界が並んでいる文字列を検索します。文字列が見つかると，VuGen は左と右の境界の間にある文字列を，関数の引数に指定したパラメータに代入します。指定された数の出現箇所が見つかると，**web_reg_save_param** はそれ以上 HTML ページを検索しません。仮想ユーザはスクリプト内の次のステップから実行を続けます。

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** は，境界文字の重複をサポートしません。例えば，入力バッファが {a{b{c} で，「{」が左の境界，「}」が右の境界の場合，検索に一致するのは「c」で，それ以外に一致するものはありません。この場合，

左右の境界は検出されましたが，境界の重複は認識されないので，「c」が唯一の一致項目となります。

標準では，境界文字列は最大 256 文字です。最大文字数を増やすには，スクリプトに **web_set_max_html_param_len** 関数を挿入します。例えば，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");A
```

第 32 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，以下の項目について説明します。

- XML ページのテストについて
- 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。

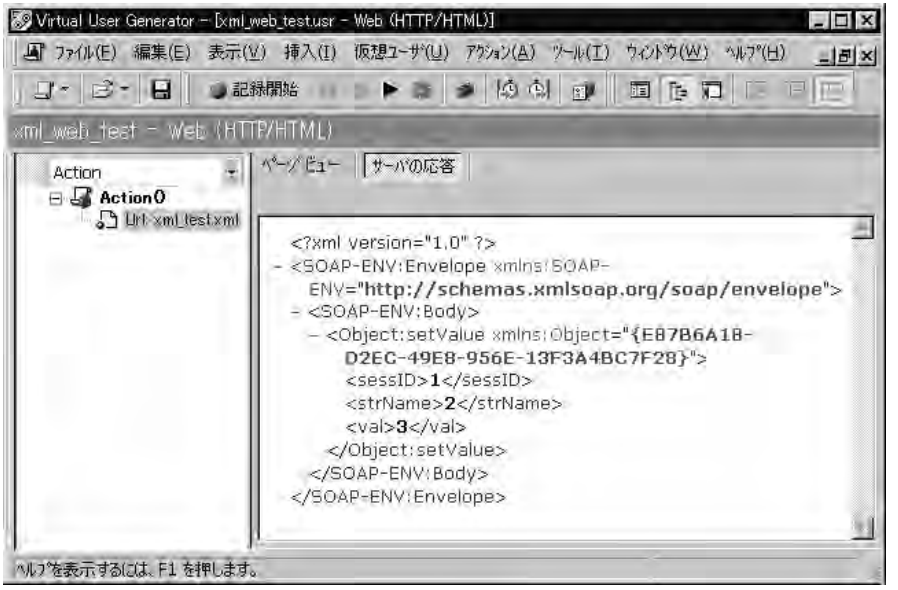
XML コードは，スクリプトに通常の URL ステップまたはユーザ定義の要求として現われます。VuGen は XTML を検出し，ユーザがそれぞれの文書型定義（DTD）およびそのエンティティと属性を参照できるようにします。VuGen can interpret the XML when the MIME タイプが **RecContentType** 属性に表示された場合，あるいは再生中に返された MIME タイプが **xml**（**application/xml** または **text/xml**）で終了した場合に XML を解釈します。DTD は色分けされているので，各要素が識別できます。DTD のツリー・ビューの分岐の展開と折りたたみも可能です。

DTD を展開する場合には，属性の値をパラメータ化できます。また標準の相関関数を使って相関できるように値を保存できます。相関関数の詳細については，「**オンライン関数リファレンス**」（［**ヘルプ**］＞［**関数リファレンス**］）を参照してください。

注：HTML ページに埋め込まれた断片的な XML である **XML アイランド**を含む DTD は表示できません。VuGen は，全体が XML であるページのみを表示します。

XML の URL ステップとしての表示

XML コードを含むページをテストする 1 つの方法は，VuGen で記録することです。まず，XML ページを通常の Web ページと同じ方法で記録します。VuGen は，DTD およびすべての XML 要素を記録します。XML ページのスナップショットは作成されませんが，XML ステップごとに，XML コードが［サーバの応答］タブのスナップショット・フレームに表示されます。

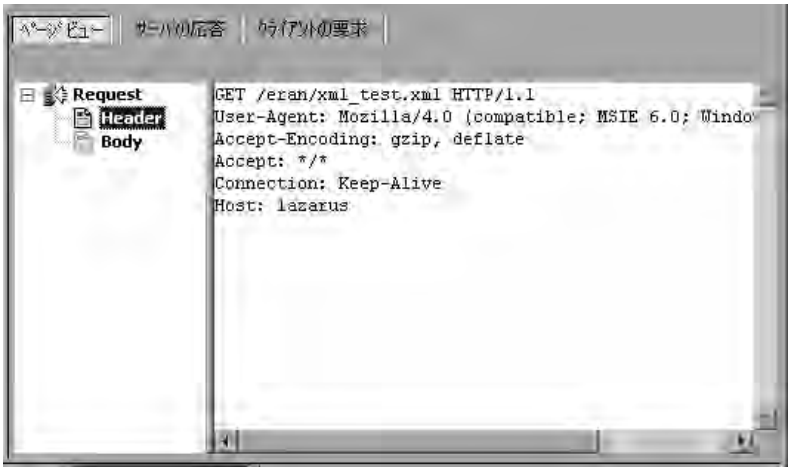


VuGen が作成した展開可能な DTD 階層は，色分けされてスナップショット表示枠に表示されます。項目の分岐を展開するにはプラス（+）記号をクリックし，折りたたむにはマイナス（－）記号をクリックします。XML タグは茶色，値は黒で表示されます。

定数値をパラメータで置き換えるには，値を選択して，右クリックし［**パラメータで置換**］を選択します。パラメータ化の標準手続きに従います。詳細については，『**第 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「VuGen パラメータを使った作業」を参照してください。

XML ページの［サーバの応答］と［クライアントの要求］もタブをクリックして表示できます。次の例は，XML ページのサーバ応答を示しています。XML ツリーのすべての分岐は展開することも折りたたむことも可能です。

次の例は，XML ページのヘッダに対するクライアントの要求を示しています。



XML をユーザ定義の要求として挿入

XML コードをユーザ定義の要求として挿入して，XML ページをテストすることもできます。このモードでは，DTD の要素が［**ユーザ定義要求プロパティ**］ボックス内にテキスト形式または XML 形式で表示されます。

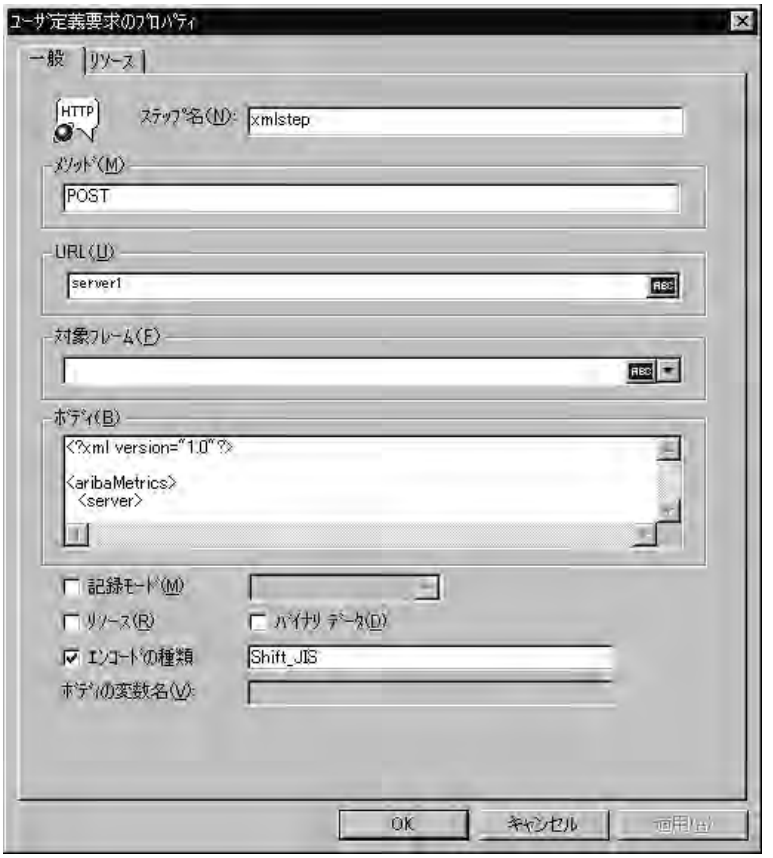
XML コードをユーザ定義の要求として追加するには，次の手順を実行します。

1 ツリー・ビュー・モードでスクリプトを表示し，希望する場所にカーソルを置き，［**挿入**］＞［**新規ステップ**］を選択します。［ステップの追加］ダイアログ・ボックスが開きます。

2 ［**ユーザ定義要求**］を選択します。［**OK**］をクリックします。［ユーザ定義要求のプロパティ］ダイアログ・ボックスが開きます。

3 ステップ名，メソッド（GET または POST），URL，対象フレーム（任意）を入力します。

4 XML コードをブラウザまたはエディタからコピーし，［ユーザ定義要求のプロパティ］ボックスの［**ボディ**］セクションに貼り付けます。



5［**記録モード**］，［**リソース**］，［**バイナリ データ**］などの再生に関するオプションを選択します。詳細については，第 29 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。

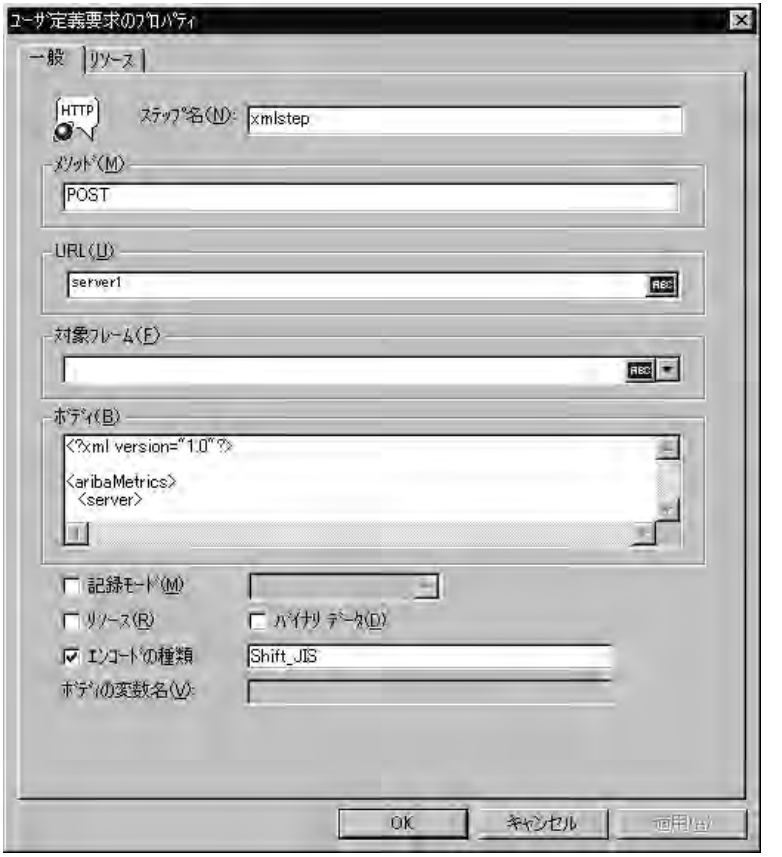
XML ユーザ定義要求のステップの表示

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，次の手順を実行します。

1 スクリプトをツリー・ビューで表示し，XML コードを表示するステップを選択します。

2 右クリックして表示されるメニューから［**プロパティ**］を選択します。［ユーザ定義要求のプロパティ］ダイアログ・ボックスが開きます。



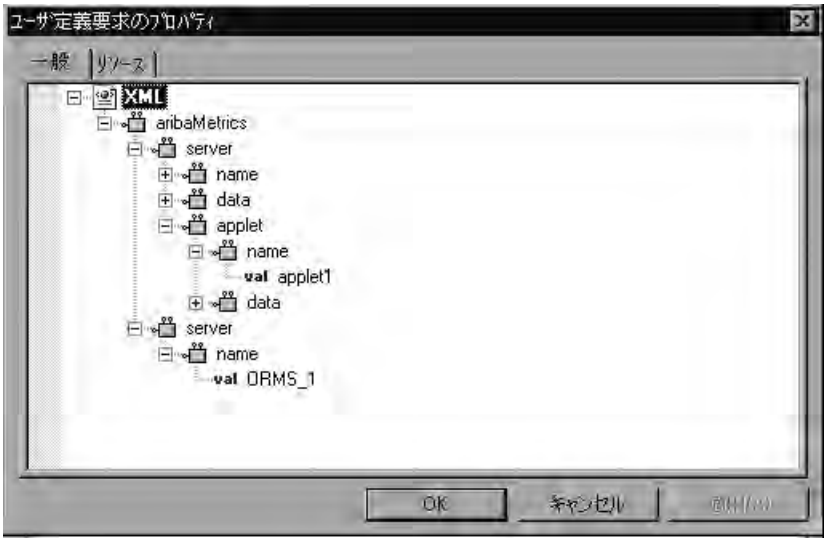
ダイアログ・ボックスの最下部に XML コードが表示されます。**RecContentType** 属性が「**text/xml**」に設定されていると，標準設定では，コードが XML 形式の階層として表示されます。このモードのときには，XML コードの編集はできません。

RecContentType 属性が「**text/xml**」以外に設定されているときには，コードはテキスト形式で表示されます。このモードのときには，XML コードは編集可能です。

3 テキストと XML の表示を切り替える場合は，右クリックして表示されるメニューから［**XML として表示**］または［**テキストとして表示**］を選択します。

4 XML 表示になっている場合，ウィンドウを大きくしてコードを表示できます。右クリックして表示されるメニューから［**拡張表示**］を選択します。ダイアロ

グ・ボックスの表示に戻るには，右クリックして表示されるメニューから［**標準表示**］を選択します。



第 33 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，以下の項目について説明します。

- レポートを使った仮想ユーザ・スクリプトのデバッグについて
- 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
- 実行結果の検索
- 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 5.0 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について

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 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「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。

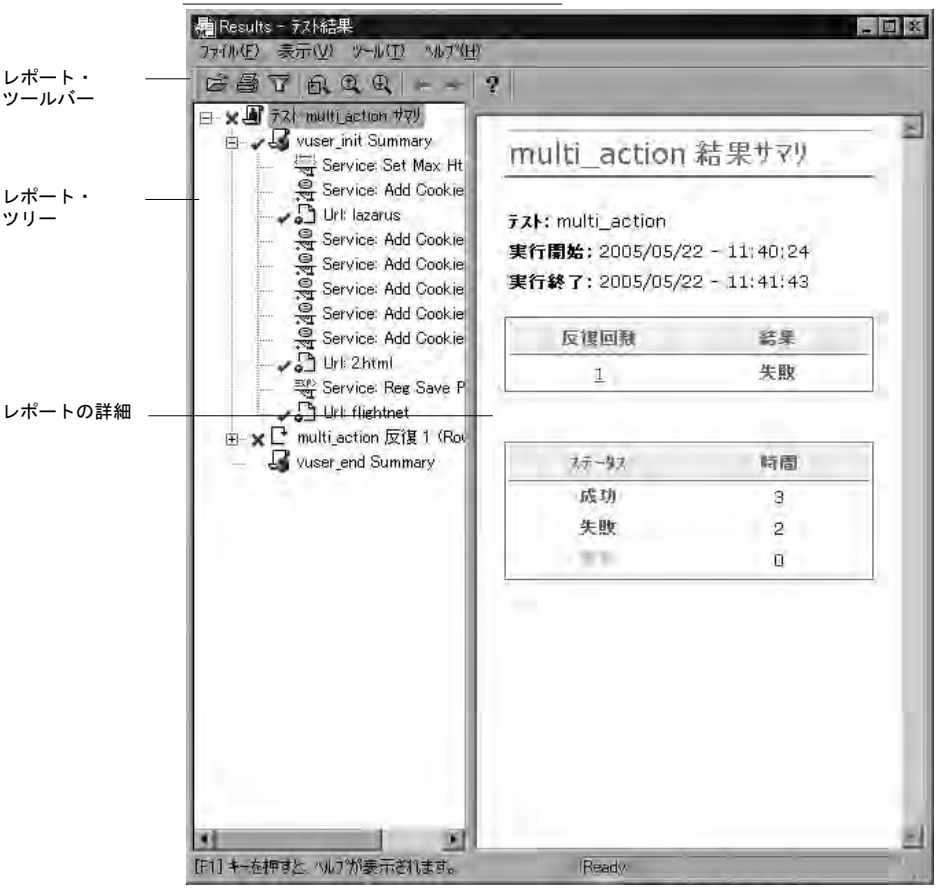
VuGen を使って Web 仮想ユーザ・スクリプトを実行した後に，結果サマリ・レポートが表示されます。

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 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「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。

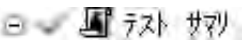
結果サマリ・レポートについて

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。

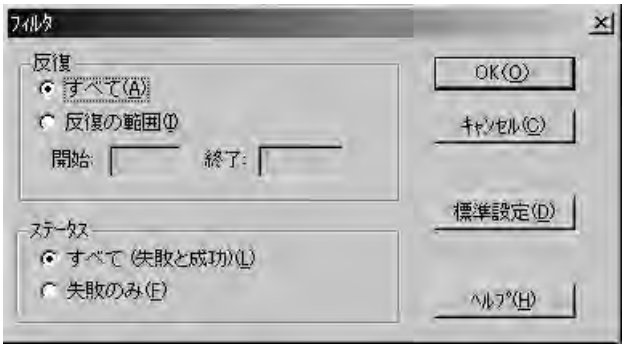
- 分岐を折りたたむには，折りたたみたい分岐の左側にあるマイナス（−）記号をクリックします。レポート・ツリーは分岐を隠し，マイナス（−）記号はプラス記号（+）に変わります。
- レポート・ツリー内のすべての分岐を折りたたむには，［**表示**］＞［**すべて折りたたみ**］を選択します。
- 分岐を展開して表示するには，表示したい分岐の左側にあるプラス（+）記号をクリックします。レポート・ツリーは分岐を展開して表示し，プラス（+）記号はマイナス（−）記号に変わります。
- レポート・ツリー内のすべての分岐を表示するには，［**表示**］＞［**すべて展開**］を選択します。

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，次の手順を実行します。

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［**上へ**］または［**下へ**］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 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「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，次の手順を実行します。

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，次の手順を実行します。

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 34 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訪ねられる質問に答えます。質問と回答は次の項に分類されています。

- セキュリティの問題
- クッキーの処理
- 実行時ビューア（オンライン・ブラウザ）
- ブラウザ
- 設定と互換性の問題

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**：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，IE 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：VuGen 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たびに **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。

VuGen 7.0 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。例えば，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。このようなサイトを記録するには，証明書と鍵の **pem** ファイルを入手し，それらをアプリケーションの **bin** ディレクトリの下にある **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：Sockets モードでは，VuGen はブラウザのバージョンに関係なく 128 ビットの暗号化をサポートします。WinINet モードでは，VuGen はロード・ジェネレータ・マシン上のブラウザと同じ暗号化をサポートします。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。

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。

クッキーの処理

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 が JavaScript や meta タグによって設定されるクッキーを正しく処理できない場合もあります。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されているときでも，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 が HTML ページ内の JavaScript によって設定さ

れたクッキーを正しく処理できない場合もあります。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すべての **web_add_cookie** ステートメントのために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** 呼び出しは削除しないでください。

再生時

Web 仮想ユーザは，HTML ページに埋め込まれた JavaScript を実行しません。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・スクリプトの中で次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：コントローラから仮想ユーザを実行したときに実行時ビューアを表示できますか？

回答：できます。コントローラから実行時ビューアを表示するには，仮想ユーザの実行を開始し，［**仮想ユーザ**］＞［**Show Vuser**］を選択します。

質問 20：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か？

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 21：仮想ユーザ・スクリプトを実行したときに，仮想ユーザから Web サーバに送信されたデータが実行時ビューアに表示されないのはなぜですか？

回答：実行時ビューアは，サーバから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 22：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 23：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。

VuGen 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方法は，Internet Explorer 4.0 以上をインストールすることです。

質問 24：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。次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。

- SOCKS プロキシの記録と再生
- Kerberos 認証

質問 25：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：Web 仮想ユーザ・スクリプトを記録するときは，任意のブラウザを使用できます。ブラウザ以外にも，HTTP（S）要求を生成するアプリケーションを使用できます。使用するアプリケーションの唯一の要件は，シングル Web プロトコル・スクリプトの場合に HTTP（S）要求を記録できるようにプロキシの設定を localhost:7777 とすることです。これは，マルチ・プロトコル・スクリプトでは必要ありません。

質問 26：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：マルチ・プロトコル・スクリプトの場合は，［記録開始］ダイアログ・ボックスでアプリケーションを選択します。適切なコマンド・ライン・パラメータを入力してください。シングル・プロトコル仮想ユーザ・スクリプトの場合は，次の手順を実行します。

1 ［**ツール**］＞［**記録オプション**］を選択し，［**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 ［**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。

3 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。元の設定をメモして，記録後に復元できるようにします。

4 ［**アプリケーションの記録開始**］ボタンをクリックして，セッションの記録を開始します。

5 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 27：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：シングル・プロトコル・スクリプトの場合のみ，変更することがあります。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 28：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 27：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。これは，シングル・プロトコル・スクリプトにのみ適用されます。

質問 29：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，［**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。

［**仮想ユーザ**］＞［**実行環境の設定**］を選択します。［**プロキシ**］タブをクリックし，［**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 30 : Netscape はインストールされていますが Internet Explorer はインストールされていない場合，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 31 :［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。詳細については，「**オンライン関数リファレンス**」の **web_set_sockets_options** を参照してください。

設定と互換性の問題

質問 32：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。

質問 33：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 35 章

Web サービスのテスト計画

VuGen を使用し，Web サービス・セッションを記録するか，WSDL をインポートすることによって，スクリプトを作成します。スクリプトまたは LoadRunner Tuning Module セッション・ステップを実行すると，仮想ユーザは Web サービスと実ユーザのやり取り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。

- Web サービスのテスト計画について
- Web サービスの実装
- Web サービスのテストにおける課題
- Web サービス・スクリプト・タイプの選択
- 負荷テストの実施
- クライアントのエミュレーション

Web サービスのテスト計画について

「**Web サービス**」は，インターネット上で広く実行できる自己完結型アプリケーションを表す言葉です。Web サービスは，Extensible Markup Language（XML）と Simple Object Access Protocol（SOAP）を使用して，新しいアプリケーションの開発と配備を短期間で実現する積み木の役割を果たします。通信がすべて XML によって行われるため，Web サービスは特定のオペレーティング・システムやプログラミング言語に限定されません。したがって，各種のアプリケーションで Web サービスを使用することで，コードを別途追加したり，ファイアウォールの背後にある互いの IT システムを詳しく知っていたりしなくても，アプリケーションどうしが互いに通信できるようになります。

従来の Web ページや Web サーバ・システムなどのクライアント / サーバ・モデルとは異なり，Web サービスはユーザ・インタフェースを備えていません。代わりに，Web サービスはビジネス・ロジック，データ，およびプロセスを，プログラムによるインタフェースを通じてネットワーク経由で共有します。開発者は Web サービスを既存のユーザ・インタフェース（ブラウザや実行可能プログラムなど）に組み込むことで，特定の機能をユーザに提供できます。

Web サービスは B2B（Business-to-Business）アプリケーションとみなされます。そのため，応答要件が厳密に決まっています。Web サービス B2B アプリケーションどうしがユーザ・インタフェースを使用せずに通信することになるため，ユーザ・インタフェースを持つアプリケーション（ブラウザ・ベースのアプリケーションなど）間の場合よりも応答時間が良好になります。このようなことから，Web サービス・アプリケーションの提供を開始する前のパフォーマンスや負荷のテストが欠かせないものとなります。

しかし，効果的なテスト計画を立てるためには，システムの構成や要件について理解しておくことが重要になります。VuGen を利用してテスト・プロセスを簡素化することはできますが，自システムの要件（容量やスケーラビリティ）について十分理解しておく必要があります。これらの要件は，負荷テストを本格的に計画，設計，および実施するうえで，きわめて重要になります。

以降の各項では，Web サービスのテストにおける課題と，Mercury のツールを使用してそれらを克服するための方法について説明します。

Web サービスの実装

Web サービスを使用するには，次の 4 つの主要なステップを踏みます。

- 適切な Web サービスの検索
- Web サービスの URL の取得
- 通信プロトコルと構文の決定
- 通信

Web サービスの検索

最初のステップでは，適切な Web サービスを探します。大手のソフトウェア・ベンダ各社が，Universal Description, Discovery, and Integration（UDDI）サービスと呼ばれる Web サービスの共通ディレクトリを組織しています。

Web サービスの URL の確立

目的のサービスを利用するには，URL を通じてアクセスする必要があります。例えば，http://myservice.com/WebServices/MyService.asmx などです。

通信手段の決定

Web サービスを見つけたら，そのサービスとの望ましい通信方法に関する情報を取得します。この情報は，通常は Web Services Description Language（WSDL）文書に格納されています。WSDL 文書は，Web サービスを，物理的なネットワークの特性を示すエンドポイント（またはポート）の集まりとして，XML を使って定義したものです。VuGen では，WSDL 文書をインポートしてスクリプト内に判読可能なコードを生成することができます。

通信

通信手段とネットワーク・サービスのプロパティを設定した後，通信が可能になります。通信を行うには，構造化 XML 文書を使用します。VuGen は，Web サービス・プロトコルを使ってすべての通信を記録し，判読可能な関数を生成します。

WSDL 文書について

WSDL 文書では，Web サービスの次の要素を定義します。

- **タイプ**：何らかの型体系（XSD など）を使ったデータ・タイプ定義のコンテナ。
- **メッセージ**：交換されるデータの定義。
- **操作**：サービスによってサポートされるアクションの記述。
- **ポート・タイプ**：1 つまたは複数のエンドポイントによってサポートされる一連の操作。
- **バインディング**：SOAP 1.1，HTTP GET/POST，MIME など，特定のポート・タイプに対するプロトコルとデータ書式。
- **ポート**：バインディングとネットワーク・アドレスの組み合わせとして定義される単一のエンドポイント。
- **サービス**：関連するエンドポイントの集まり。

WSDL 文書では，要素のいくつかは抽象的であり，その要素が特定のネットワーク・サービスに固有のものではなく，再利用できることを示しています。抽象的な要素の例として，**メッセージ**と**操作**があります。

その他の要素は具象的であり，その要素が通信ごとに固有の値を持つことを示しています。具象的な要素の例としては，**ポート**があります。

WSDL では，**バインディング**要素を使って特定のプロトコル，データ形式，構造体を，抽象メッセージ，操作，エンドポイントにアタッチします。バインディング要素は抽象定義であるため，メッセージ，操作，およびエンドポイントの要素は別の通信に対しても再利用できます。**ポート**は，ネットワーク・アドレスとバインディングを関連付けることによって定義します。ポートの集まりによって，**サービス**が定義されます。

次の例は，株価を提供するサービスの WSDL 定義を示しています。このサービスは，GetLastTradePrice という単一の操作をサポートします。この操作は，HTTP を介した SOAP 1.1 プロトコルを使って配備されます。この要求は，文字列タイプの銘柄を受け取り，株価を浮動小数点数として返します。

この例では，SOAP エンコーディングの代わりに固定の XML フォーマットを使用しています。

```
<?xml version="1.0"?>
<definitions name="StockQuote"

targetNamespace="http://example.com/stockquote.wsdl"
        xmlns:tns="http://example.com/stockquote.wsdl"
        xmlns:xsd1="http://example.com/stockquote.xsd"
        xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
        xmlns="http://schemas.xmlsoap.org/wsdl/">
```

次のセクションではデータの **type** 要素を定義しています。

```
<types>
    <schema targetNamespace="http://example.com/stockquote.xsd"
        xmlns="http://www.w3.org/2000/10/XMLSchema">
      <element name="TradePriceRequest">
        <complexType>
          <all>
            <element name="tickerSymbol" type="string"/>
          </all>
        </complexType>
      </element>
      <element name="TradePrice">
        <complexType>
          <all>
            <element name="price" type="float"/>
          </all>
        </complexType>
      </element>
    </schema>
  </types>
```

次のセクションでは，**message** 要素をいくつか定義しています。

```
<message name="GetLastTradePriceInput">
    <part name="body" element="xsd1:TradePriceRequest"/>
  </message>

<message name="GetLastTradePriceOutput">
    <part name="body" element="xsd1:TradePrice"/>
  </message>
```

次のセクションでは，**portType** 要素を操作と関連付けて定義しています。

```
<portType name="StockQuotePortType">
    <operation name="GetLastTradePrice">
      <input message="tns:GetLastTradePriceInput"/>
      <output message="tns:GetLastTradePriceOutput"/>
    </operation>
  </portType>
```

次のセクションでは，**binding** 要素を定義しています。このバインディングでは，GetLastTaradePrice 操作を SOAP over HTTP プロトコルにアタッチしています。

```
 <binding name="StockQuoteSoapBinding"
type="tns:StockQuotePortType">
      <soap:binding style="document"
transport="http://schemas.xmlsoap.org/soap/http"/>
      <operation name="GetLastTradePrice">
        <soap:operation
soapAction="http://example.com/GetLastTradePrice"/>
        <input>
          <soap:body use="literal"/>
        </input>
        <output>
          <soap:body use="literal"/>
        </output>
      </operation>
    </binding>
```

次のセクションでは，最後の要素 **service** を定義しています。サービスは 1 つまたは複数のポートで構成されます。

```
 <service name="StockQuoteService">
      <documentation>My first service</documentation>
      <port name="StockQuotePort" binding="tns:StockQuoteBinding">
        <soap:address location="http://example.com/stockquote"/>
      </port>
    </service>

</definitions>
```

Web サービスのテストにおける課題

Web サービスのテストにはいくつかの解決すべき課題があります。

非標準 UI：複雑な Web サービスをテストするうえで最も大きな問題の 1 つとなるのが，クライアントのテストです。Web サービスでは，テストが可能なユーザ・インタフェースが表示されません。Web サービスは多くの場合、従来のクライアント / サーバ手法を使用した Web ブラウザや軽量クライアント・アプリケーションではなく、アプリケーションまたはサーバです。また，クライアント・アプリケーションが，ユーザ・インタフェースを持たない別のアプリケーションによって駆動されていることもあります。

ビジネス・プロセスの分離：実際のビジネス・プロセスを Web サービスの他のトラフィックから分離するのは容易ではありません。クライアント・アプリケーションを記録したとき，テスト対象のビジネス・プロセスとは関係のない不必要なメソッド呼び出しがスクリプトに含まれることがあります。

複雑なロジック：Web サービスのロジックは複雑なことが多く，テストを実行するたびに要素の値が異なる場合もあります。これらの値は，以降の Web サービス・メソッド呼び出しで使用できるように，相関させてパラメータに保存する必要があります。

タイミングと順序：ビジネス・プロセスとは関係のない非同期呼び出しをクライアントが実行することもあります。呼び出しが，アプリケーションによって非同期に発行されたり，別のアプリケーションによって開始された別のビジネス・プロセスによって発行されたりすることがあります。例えば Web サービスで，現在のビジネス・プロセスに関係なく 1 時間ごとにバックアップ手続きが開始されることが考えられます。また，Web サービス・メソッドの呼び出し順序が実行のたびに変わることがあります。

パフォーマンスの分析：パフォーマンスの分析もまた，Web サービスのテストにおいて困難な作業です。負荷シナリオの実行が完了した後は，問題の発生場所を調べるための作業が必要になります。Web サービス・システムには，複数のサーバが関与していることがあり，場合によっては分散システムに配備されている可能性もあります。そのため，問題が生じているコンポーネントを正確に特定することが難しい場合があります。

上記に示した課題のほとんどは，Web サービス仮想ユーザの使用，思考遅延時間の追加，および相関の適用によって克服できます。スクリプト作成のための最適な方法の選択の詳細については，466 ページ「Web サービス・スクリプト・タイプの選択」を参照してください。

Mercury の LoadRunner コントローラおよびアナリシス・ツールを使用すると，モニタを設定し，分散システム全体をまたいでトランザクション・ブレークダ

ウンを実行することによって，パフォーマンスの分析に関する課題を克服できます。

Web サービス・スクリプト・タイプの選択

Web サービス・スクリプトは，次のいずれかの方法で作成できます。

- Web サービス・スクリプトの記録
- WSDL 文書のスキャン
- IDE 統合機能

Web サービス・スクリプトの記録

VuGen の組み込みレコーダを使用して，Web サービス・アプリケーションを記録できます。

アプリケーションを記録するときは，WSDL ファイルを使用するかどうかを問わず記録が可能です。WSDL を記録にインポートすると，記録された SOAP トラフィックを解析して Web サービス呼び出しを抽出することによって，VuGen による高レベル・コードの作成が可能になります。

クライアント・セッションの記録には，記録が単純であるという利点があります。記録中，アクションを実行して，特定のビジネス・プロセスをエミュレートするスクリプトを作成します。トランザクションを作成し，そのトラフィック，タイミング，および呼び出しの順序を監視でき，ピア間で受け渡しされるデータを監視できます。また，スクリプトを機能テストと負荷テストの両方に使用できます。

この方法の欠点は，アプリケーションからのすべての HTTP がキャプチャされ，記録対象の特定のビジネス・プロセスとは関係のないパケットであってもキャプチャされることです。したがって，記録されたスクリプトに対して追加のフィルタを適用しなければならない場合があります。

WSDL 文書のスキャン

この方法は，スクリプトを依存性のない，相互運用可能なものにする必要があるシステムの場合に最適です。VuGen は，ヘッダー（HTTP）と名前空間接頭辞（XML）を送信するクライアント・エミュレーションを使用して，ツールキットを正確にエミュレートします。また，このスクリプトを機能テストと負荷テストの両方に使用できるという利点もあります。

WSDL 文書のスキャンによる方法の欠点は，WSDL にタイミング情報がなく，メソッドの呼び出し順序もわからないことです。タイミングの問題は，思考遅延時間による遅延を挿入することで克服できます。

この方法のもう 1 つの欠点は，パラメータ値を手作業で入力しなければならないことです。引数値を手作業で入力するため，特に複雑な Web サービス・メソッドの場合は，誤ったデータを入力する可能性が増し，誤りのあるテスト結果が得られることになります。値を手作業で入力した後に，メソッドごとに XML ファイルをエクスポートすることで，この作業を容易にすることができます。そうすることで，後でテストを行う際に，以前に入力した値を格納した XML を使用できるようになります。詳細については，第 36 章「Web サービス仮想ユーザの開発」を参照してください。

IDE 統合機能

IDE 統合機能を使用してスクリプトを作成するには，主に使用している開発言語でスクリプトを記述した後，そのスクリプトを LoadRunner などの自分の環境に組み込みます。スクリプトは，Web サービスをエミュレートする Java または .NET の開発環境で作成できます。

この方法には，標準の作業環境の中でスクリプトを作成できるという利点があります。通常，この方法では相関とパラメータ化が自動的に処理されます。自動的に処理されない場合は，相関を手作業で実行できます。

また，IDE 統合機能では，WSDL のスキャンや記録による方法では処理されない，独自の高度なテクノロジも処理されるという利点もあります。IDE 統合機能では，アプリケーションの実際の動作がエミュレートされます。

IDE 統合機能の主な欠点は，スケーラビリティがないことです。CPU への負担が大きい Java や .Net などのコードを使用してテストを記述した場合，負荷テストで実行できる仮想ユーザの数が制限されることになります。また，IDE 統合機能では，他の方法よりも多くの人的労力や開発時間をソフトウェアや QA のエンジニアから得なければならない可能性があります。

負荷テストの実施

Web サービスを正常にテストするため，サービスを完全な環境の一部としてチェックする負荷テストを実行する必要があります。テストを実行した後，その結果を確認し，問題の場所を調べます。

テストでは，最初のステップとして，ユーザにとって一般的なビジネス・プロセスを定義します。そして，それらのプロセスの呼び出し方法とスクリプトの

生成方法（記録，WSDL のスキャン，または IDE 統合機能）を決める必要があります。

次のステップとして，エミュレートするユーザ数と，実行するビジネス・プロセスについて計画を立てます。これらは自システムの要件に従って定義する必要があります。分散型の Web サービス構成では，同じ負荷テストの中でどのビジネス・プロセスを実行するかについても考慮する必要があります。

最後に，不可欠なステップとして，テスト実行の間の情報収集を行うモニタを，システム全体にわたって設定します。モニタを設定する前に，収集する測定情報を定義しておく必要があります。Mercury では広範な環境に対応した各種モニタを提供しています。

クライアントのエミュレーション

VuGen は，.NET バージョン 1.1，Axis バージョン 1.1，Glue バージョン 4.1.2，MS SOAP バージョン 3.0 など，いくつかの Web サービス・ツールキットに対応するエミュレーション機能をサポートしています。ツールキットでは，Web サービスがそれぞれ少し異なる方式で解釈されます。このような独自の動作をエミュレートするために，VuGen はいくつかの要素を，サーバがメッセージを作成する方法を表す差別化要素（引数の型，形式，接頭辞，ヘッダーなど）として使用します。

ツールキットのエミュレーションの選択方法については，500 ページ「Web サービスの実行環境の設定」を参照してください。

第 36 章

Web サービス仮想ユーザの開発

VuGen を使用し，SOAP セッションを記録するか，WSDL 文書内をスキャンすることによって，Web サービス・スクリプトを作成します。スクリプトを実行すると，仮想ユーザは Web サービスと実ユーザのやり取り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。

- VuGen での Web サービスに関する作業の概要
- Web サービス・スクリプトのワークフロー・ウィザードの使用
- Web サービス・スクリプトの新規作成
- Web サービス・スクリプトの記録
- WSDL 文書のインポート
- WSDL 文書の管理
- WSDL の検証オプションと比較オプションの設定
- XML ツリーの編集
- IDE 統合機能

Web サービスについて

Web サービスは，インターネットをまたいでさまざまなプラットフォームの上で広く実行できる自己完結型アプリケーションです。Extensible Markup Language（XML）と Simple Object Access Protocol（SOAP）を使って作成されます。Web サービスは，新しいアプリケーションの開発，配備を短期間で実現する積み木の役割を果たします。

VuGen での Web サービスに関する作業の概要

本項では，VuGen を使って Web サービス・スクリプトの作成プロセスの概略を説明します。

Web サービス・スクリプトを作成するには，次の手順を実行します。

1 **空の Web サービス・スクリプトを作成します。**

ワークフロー•ウィザードまたは標準のメニューを使用して新規スクリプトを作成し，「Web Services」仮想ユーザ・タイプを選択します。

2 **スクリプトを生成します。**

Web サービス・ウィザードを実行してスクリプトを生成し，Web サービス・セッションを記録するか WSDL 文書内をスキャンします。最適な方法を選択するための詳細については，473 ページ「Web サービス・スクリプトの新規作成」を参照してください。記録の場合は，記録オプションを設定します（[**ツール**] > [**記録オプション**]）。WSDL ファイル内をスキャンする場合は，検証ユーティリティを実行します。詳細については，488 ページ「WSDL 文書の管理」を参照してください。

IDE 統合の使用によるスクリプトの作成については，498 ページ「IDE 統合機能」を参照してください。

3 **スクリプトを拡張します。**

トランザクション，ランデブー・ポイント，および XML 関数を挿入することによって，スクリプトを拡張します。

詳細については，『**第 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「仮想ユーザ・スクリプトの拡張」および「XML API プログラミング」を参照してください。

4 **実行環境を設定します。**

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

一般的な実行環境の設定の詳細については，500 ページ「Web サービスの実行環境の設定」および『**第 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「実行環境の設定」を参照してください。

5 **VuGen でスクリプトを保存して実行します。**

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。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実行環境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。

WSDL ファイル内のスキャンによって作成したスクリプトの場合は，WSDL ファイルの比較を実行して，元のファイルに何も変更が加えられていないことを確かめます。詳細については，499 ページ「Web サービス仮想ユーザの実行」を参照してください。

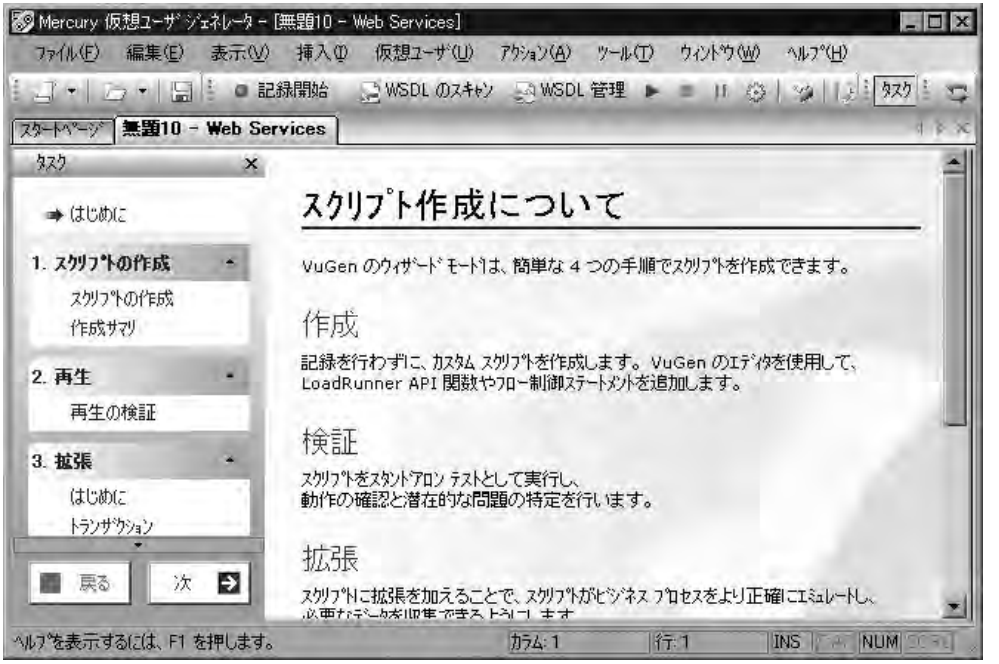
6 スクリプトをスタンドアロン・テストとして実行する方法の詳細については，『**第 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「スタンドアロン・モードでの仮想ユーザ・スクリプトの実行」を参照してください。**テスト結果を表示します。**

テスト結果ユーティリティを開き，反復ごとに再生のサマリを表示します。詳細については，517 ページ「Web サービス・レポートの表示」を参照してください。

スクリプトを作成した後，スクリプトを自分の環境（LoadRunner シナリオ，Performance Center 負荷テスト，Tuning Module セッション，または Business Process Monitor プロファイル）に統合します。詳細については，『**Mercury LoadRunner コントローラ・ユーザーズ・ガイド**』，**Tuning Console**，**Performance Center**，または **Application Management** のマニュアルを参照してください。

Web サービス・スクリプトのワークフロー・ウィザードの使用

ワークフロー画面は，スクリプトの作成の手順を導いてくれます。［タスク］表示枠のリンクをクリックすると，スクリプト作成の手順についての説明を読むことができ，記録と再生に関する情報を表示できます。［**Back**］および［**Next**］ボタンを使用して，画面間を移動できます。



ワークフロー・ウィザードが見えない場合は，［タスク］表示枠が開いていることを確認してください（［タスク］表示枠は，ツールバーの［**タスク**］ボタンを使用して表示 / 非表示を切り替えます）。次に最初のリンクである［**はじめに**］をクリックします。

詳細については，第 3 章「ワークフロー・ウィザードの使用」を参照してください。

Web サービス専用のウィザード画面は，［スクリプトの作成］と［作成サマリ］です。

スクリプトの作成

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

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

サマリの作成

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

また，スクリプトを変更するための次のリンクも含まれます。

- ［**プロトコル**］：スクリプトの作成中に使用されたプロトコルの一覧が表示されます。
- ［**アクション**］：アクションが記録またはスキャンされたセクションについて説明します。
- ［**Modify Script**］：スクリプトの変更方法について説明します。

Web サービス・スクリプトの新規作成

Web サービス・スクリプトは，次のいずれかの方法で作成できます。

- Web サービス・スクリプトの記録
- WSDL 文書のインポート
- IDE 統合機能

特定のビジネス・プロセスについて実ユーザの動作をエミュレートするスクリプトを作成する場合，通常は，クライアントを記録する方法が最も簡単です。

WSDL ファイルをスキャンする方法では，WSDL ファイルをインポートし，Web サービスと通信するテストを作成できます。

IDE 統合機能を利用する場合は，通常使用している開発環境を使用してスクリプトを記述します。VS .NET などの開発環境で記述されたスクリプトは，アプ

リケーションのネイティブの言語でビジネス・プロセスをエミュレートします。

テストのための最適な方法の選択の詳細については，466 ページ「Web サービス・スクリプト・タイプの選択」を参照してください。

Web サービス・スクリプトの記録

Web サービス・クライアントを記録することによって，Web サービス・スクリプトを作成できます。

アプリケーションを記録すると，Web サービスを記述する WSDL ファイルを任意で指定することができます。これにより，記録された SOAP トラフィックを解析して Web サービスへの呼び出しを抽出する，高レベル・コードを作成できます。

記録のためのウィザードの手順は次のとおりです。

- 記録のための WSDL ファイルの指定（WSDL ファイルをスクリプトに含める場合のみ）
- WSDL ファイル検証サマリ（WSDL ファイルをスクリプトに含める場合のみ）
- 記録対象アプリケーションの指定

記録のための WSDL ファイルの指定

この画面で，WSDL ファイルを指定します。アプリケーションを記録するときは，WSDL ファイルを使用するようにも，または WSDL ファイルを使用しないようにも，記録できます。WSDL ファイルをインポートし，記録された SOAP トラフィックを解析して Web サービス呼び出しを抽出する，高レベル・コードを作成します。

［**記録中に WSDL ファイルを使用しない**］：VuGen は **soap_request** 関数を使用するスクリプトを作成します。

［**指定した WSDL ファイルを使用する**］：［Add］ボタンを使用して項目をリストに追加します。VuGen は，**web_service_call** 関数を使用してスクリプトを作成します。

記録の詳細については，474 ページ「Web サービス・スクリプトの記録」を参照してください。

記録対象アプリケーションの指定

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

［**Record Web Browser**］：指定した URL で始まる，標準設定の Web ブラウザのトラフィックを記録します。

［**Record any application**］：特定のクライアント・アプリケーションのアクションを記録します。［**記録対象アプリケーション**］ボックスにアプリケーションのフル・パスを指定します。関連する引数や作業用ディレクトリをすべて指定します。

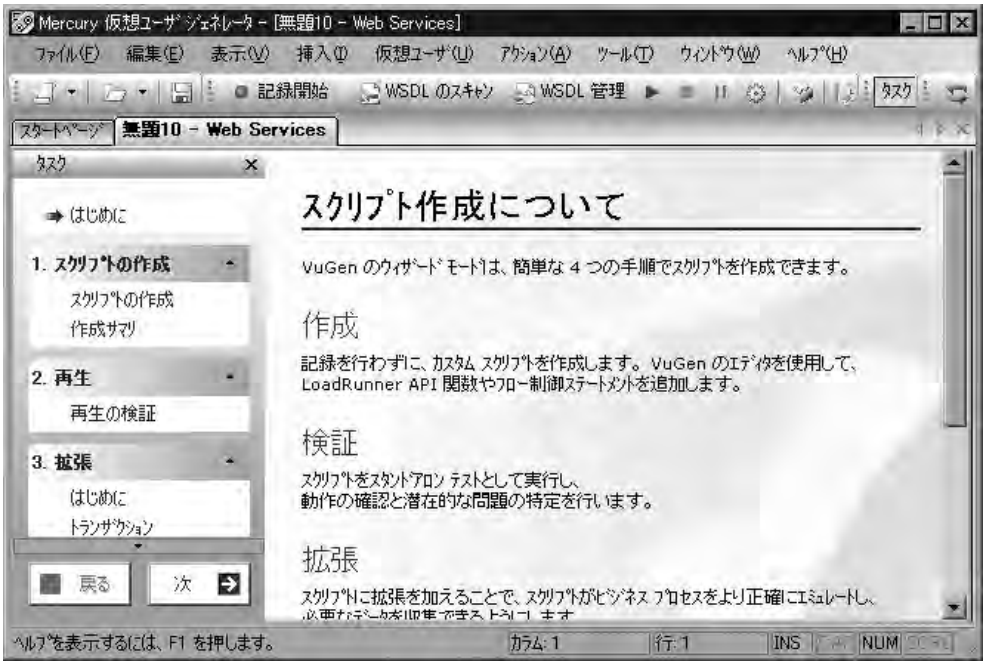
［**アクション内に記録**］：コードを生成して挿入する対象となるアクション。繰り返す必要のない起動処理がある場合は，それらを **vuser_init** セクションに置きます。記録中に **Action** などの別のセクションに切り替えることができます。

［**アプリケーションの起動を記録する**］：アプリケーションの起動処理をスクリプトの一部として記録します。特定の位置から，起動処理を含めずに記録を開始する場合は，このオプションをクリアします。

記録の詳細については，474 ページ「Web サービス・スクリプトの記録」を参照してください。

Web サービス・アプリケーションを記録するには，次の手順を実行します。

1 新規スクリプトを生成するには，［**ファイル**］＞［**新規作成**］を選択します。［e ビジネス］仮想ユーザ・カテゴリから［**Web Services**］仮想ユーザ・タイプを選択します。ワークフロー・ウィザードが起動します。



ワークフロー・ウィザードが表示されなければ，［タスク］表示枠の［**はじめに**］リンクをクリックします（［タスク］表示枠は，ツールバーの［**タスク**］ボタンを使用して表示 / 非表示を切り替えます）。

2［タスク］表示枠の［スクリプトの作成］リンクをクリックします。右側の表示枠で，［**Web サービス ウィザード**］ボタンをクリックして，スクリプトの作成を開始します。

3［**クライアント アプリケーションの記録**］を選択し，［**次へ**］をクリックします。ウィザードが次の手順［**記録対象 WSDL ファイルの指定**］に移ります。

4 WSDL ファイルを使用せずにアプリケーションを記録する場合は，［**記録中に WSDL ファイルを使用しない**］を選択します。［**次へ**］をクリックし，手順 6 に進んでアプリケーションを指定します。

5 1 つ以上の WSDL ファイルを使用してスクリプトを記録する場合は，［**指定した WSDL ファイルを使用する**］を選択します。使用した WSDL ファイルの履歴リストが表示されます。リストへの WSDL ファイルを追加するには，次の手順を実行します。

項目の追加ボタン をクリックします。新しい行がリストに追加されます。

WSDL ファイルのパスまたは URL 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。ステータスが有効な状態で，新しい場所がリストに追加されます。

エントリをリストから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。WSDL エントリを一時的に無効にするには，エントリの左にあるチェック・ボックスをクリアします。

［**次へ**］をクリックします。［記録対象アプリケーションの指定］画面が開きます。

6 Web ベースのアプリケーションの場合は，［**標準の Web ブラウザを記録**］を選択し，記録対象 URL を指定します。

他の Windows アプリケーションの場合は，［**すべてのアプリケーションを記録**］を選択します。記録対象クライアント・アプリケーションのフル・パスを入力または参照します。必要ならば，プログラム引数や作業ディレクトリを指定します。

［**アクション内に記録**］からアクションを選択し，［**アプリケーションの起動を記録する**］を選択して，アプリケーションの起動を記録します。

7［**完了**］をクリックしてウィザードを終了します。記録ツールバーが表示されます。VuGen によってクライアント・アプリケーションが起動され，記録セッションが開始します。

WSDL 文書のインポート

WSDL 文書をスキャンすることによって，仮想ユーザ・スクリプトを作成できます。パスまたは URL を指定し，値を自分でメソッドに割り当てるか，または VuGen による値の自動割り当てを有効にするかを指定します。また，スクリプトに含めるメソッドを指定することもできます。WSDL ファイルに定義されているすべてのメソッドを使用する必要はありません。

WSDL ファイルをスキャンすると，VuGen によってメソッドごとに **web_service_call** 関数が自動的に生成されます。各メソッド呼び出しの合間には 3 秒の思考遅延時間が追加されます。

WSDL ファイルのスキャン時に，文書の XML コードと構文が VuGen によって検証されます。エラーまたは警告が発生した場合は，その情報に関するポップアップ・メッセージが表示されます。WSDL 検証機能を単体のユーティリティとして使用し，スクリプトを作成する前に WSDL を確認することもできます。詳細については，488 ページ「WSDL 文書の管理」を参照してください。

WSDL スキャン・ウィザードを使用して，VuGen による Web サービス・スクリプトの作成方法を制御できます。WSDL スキャンのウィザードの手順を次に示します。

- スキャンのための WSDL ファイルの指定
- WSDL ファイル検証サマリ
- テストに含めるメソッドの選択
- 引数値の指定
- ウィザードの最後の画面

スキャンのための WSDL ファイルの指定

この画面で，スキャンの対象となる WSDL ファイルのパスまたは URL を指定します。VuGen は，WSDL ファイルに定義されているメソッドを呼び出すスクリプトを作成します。

［**URL**］：WSDL ファイルの URL。

［**ファイル**］：ローカル・ドライブまたはネットワーク・ドライブに存在する WSDL ファイルのフル・パス。

［**自動テストの作成**］：WSDL ファイルのすべてのメソッドを呼び出してすべての要素に値を自動的に割り当てる，簡単なスクリプトを作成します。特定のメ

ソッドを除外する場合や，値を手作業で割り当てる場合は，自動テストを作成せずにウィザードの実行を続けます。

スキャンの詳細については，488 ページ「WSDL 文書の管理」を参照してください。

WSDL ファイル検証サマリ

VuGen による WSDL ファイルのスキャンが正常に実行されると，ウィザードの［WSDL Files Validation Summary］画面が開きます。この画面は，WSDL ファイルにエラーがあったかどうかを示します。

［**妥当性チェック レポートを開く**］：検証レポートを開きます。このビューでは，WSDL 文書のテキストと，すべての検証エラーまたは警告を表示できます。

注：WSDL ファイルをスキャンする際，VuGen によって作業用のコピーが作成され，スクリプトと一緒に保存されます。標準設定では，検証と変更はすべて作業用コピーに対して実行されます。WSDL ファイルを更新する場合は，WSDL スキャン・ウィザードを使ってファイルを再度スキャンします。

テストに含めるメソッドの選択

この画面で，テストの対象となる WSDL ファイルのメソッドを指定します。VuGen によって，メソッドごとに **web_service_call** 関数が作成されます。

［**サービス名**］：WSDL ファイルに定義されている Web サービスの名前。テストの対象となるメソッドの Web サービスを選択します。

［**詳細**］：現在の Web サービスの説明。

［**利用可能なメソッド**］：選択した Web サービスにあるすべてのメソッドのリスト。

［**選択したメソッド**］：テストに含める Web サービス・メソッドのリスト。メソッドをこのリストに追加するには，左側の表示枠でメソッドをダブルクリックし，右方向の矢印をクリックします。

詳細については，488 ページ「WSDL 文書の管理」を参照してください。

引数値の指定

この画面で，メソッドの引数の値を指定します。選択するツリー・ノードのレベルに応じて，右側の表示枠が変わります。

値（入力，出力，または SOAP ヘッダー）を変更すると，値のアイコンの色が青に変わります。入力引数とヘッダーの場合，この値は VuGen からサーバに送られる実際の値を示します。

選択対象 ...	**右側の表示枠の表示**
単純な入力引数	各メソッドの［**名前**］とその［**値**］
複雑な入力引数	次のオプションが表示されます。 • ［**呼び出しに引数を含める**］: 引数の値を，この引数に対して生成された **web_service_call** 関数に含めます。 • ［**XML**］: ［**XML のインポート**］ボタンと［**XML の編集**］ボタンを有効にします。XML を編集することで，引数値を手作業で挿入できます。詳細については，497 ページ「XML ツリーの編集」を参照してください。 • ［**この引数の自動値を生成**］: この複合型ノードのすべての引数に対して，自動値を挿入します。 • 2 つの追加のボタン : ［**追加**］および［**削除**］。これらのボタンを使用して，添字を指定して配列引数を追加および削除できます。
引数単体	次の情報が表示されます。 • ［**値**］: 引数の値。 • ［**この引数の自動値を生成**］: このノードに対して自動値を挿入します。
出力引数	引数の［**名前**］と，値を格納できる［**パラメータ**］。
SOAP ヘッダー	編集ボックス。現在の要素に対する SOAP ヘッダーの値を指定します。詳細については，481 ページ「SOAP ヘッダーの使用」を参照してください。

SOAP ヘッダーの変更方法については，次項 481 ページ「SOAP ヘッダーの使用」を参照してください。

SOAP ヘッダーの使用

このビューは，メソッドのツリー・ビューで「**SOAP ヘッダー**」を選択したときに使用できます。右側の表示枠で，SOAP ヘッダーを使用するかどうかを指定できます。SOAP ヘッダーを使用するには，［**SOAP ヘッダを使用する**］を選択します。各要素について SOAP ヘッダーを個別に指定する必要があります。SOAP ヘッダーの XML コードをインポートするか，［XML の編集］オプションを使用して自分で作成できます。詳細については，497 ページ「XML ツリーの編集」を参照してください。

WSDL ファイルのスキャンの詳細については，478 ページ「WSDL 文書のインポート」を参照してください。

ウィザードの最後の画面

ウィザードの最後の画面では，WSDL 文書のスキャンを通じてスクリプトを生成するのに必要な情報をすべて指定したことが通知されます。

［**完了**］をクリックすると，スクリプトが生成されます。

スクリプトを実行する前に実行環境を設定する場合は，［**実行環境の設定**］ボタンをクリックします。これらの設定では，スクリプト実行中の思考遅延時間，反復，および実行論理を指定します。

WSDL 文書のインポートによるスクリプトの作成

次に示す手順では，アプリケーションの記録によってではなく WSDL ファイルをインポートすることによってスクリプトを作成する方法について説明します。

WSDL 文書をインポートすることによってスクリプトを作成するには，次の手順を実行します。

1 Web サービス仮想ユーザ・スクリプトを新規作成します。［e ビジネス］仮想ユーザ・カテゴリから［**Web Services**］仮想ユーザ・タイプを選択します。ウィザードのメイン画面が開きます。

2 ［**WSDL ファイルのスキャン**］を選択し，［**次へ**］をクリックします。ウィザードが次の手順［**スキャン対象 WSDL の指定**］に移ります。

すでにスクリプトを生成していて，WSDL を既存のスクリプトにスキャンする場合は，［**仮想ユーザ**］＞［**WSDL のスキャン**］を選択するか，［WSDL のスキャン］ボタンをクリックします。WSDL インポート・ウィザードから［**スキャン対象 WSDL の指定**］の手順が開きます。

3 WSDL 文書ファイルの URL を指定するには，［**URL**］を選択します。ローカル・ドライブまたはネットワーク・ドライブ上にある WSDL を指定または参照するには，［**ファイル**］を選択します。

4 スキャン中に WSDL ファイルを検証するには，［**WSDL ファイルの検証**］を選択します。今すぐ検証をしない場合でも，［WSDL 管理］ウィンドウを使用して後で検証できます。

5 入力値がすでに割り当て済みのスクリプトを作成するには，［**自動テストの作成**］をクリックします。ウィザードが次の手順に移り，自動値がすべての要素に割り当てられます。

［**自動テストの作成**］を選択した場合は，次の手順に進みます。

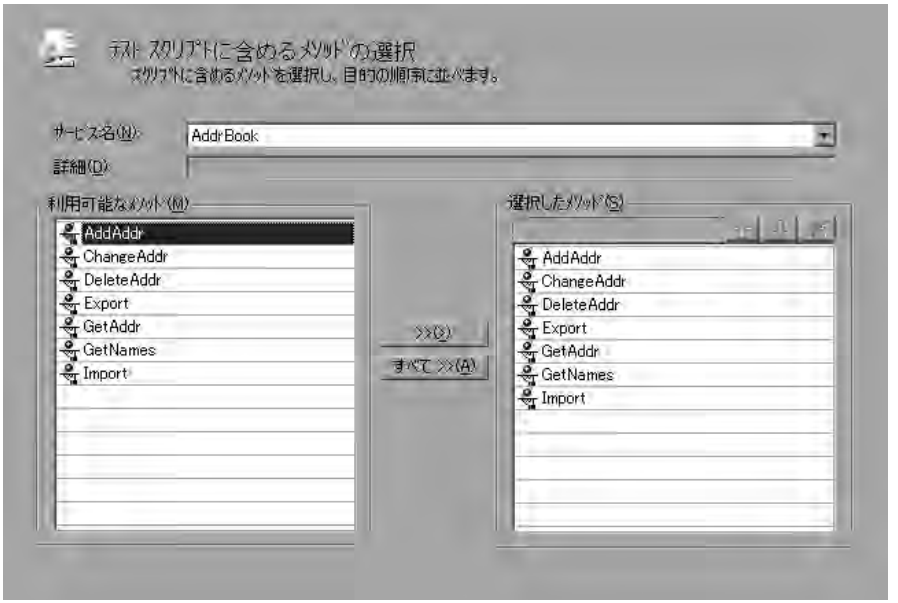
6 自動テストを作成せず，使用するメソッドを手作業で指定する場合は，［**次へ**］をクリックします。VuGen による WSDL ファイルのインポートが正常に実行された場合は，［WSDL Files Validation Summary］の手順が開きます。

VuGen によって WSDL ファイルに問題があることが検出された場合は，問題を説明する警告が表示されます。

検証レポートを開いてエラーや警告を表示するよう促されます。検証レポートを開くには，［**はい**］をクリックします。

7 レポートを開いていない場合は，現在のウィザード画面で［**妥当性チェック レポートを開く**］をクリックします。

8 スキャン・ウィザードの［**次へ**］をクリックします。［**テスト スクリプトに含めるメソッドの選択**］の手順が開き，各オブジェクトの使用可能なすべてのメソッドが一覧表示されます。



［**サービス名**］リストから，使用する Web サービスを選択します。

［**利用可能なメソッド**］フレームからメソッドを 1 つまたは複数選択し，右矢印をクリックして，それらを［**選択したメソッド**］セクションに移動します。複数のメソッドを選択するには，Windows の標準のキー操作を使用します。

必要に応じて，上矢印または下矢印をクリックして，メソッドの順番を変更します。メソッドを削除するには，［**削除**］ボタン（［**X**］）をクリックします。

［**次へ**］をクリックします。次の手順［**引数値の指定**］が開きます。

9 値を設定する対象となるメソッドを展開します。個々の引数の値を設定するには，左側の表示枠で引数を選択し，右側の表示枠で［**値**］オプションを選択して，値を入力します。

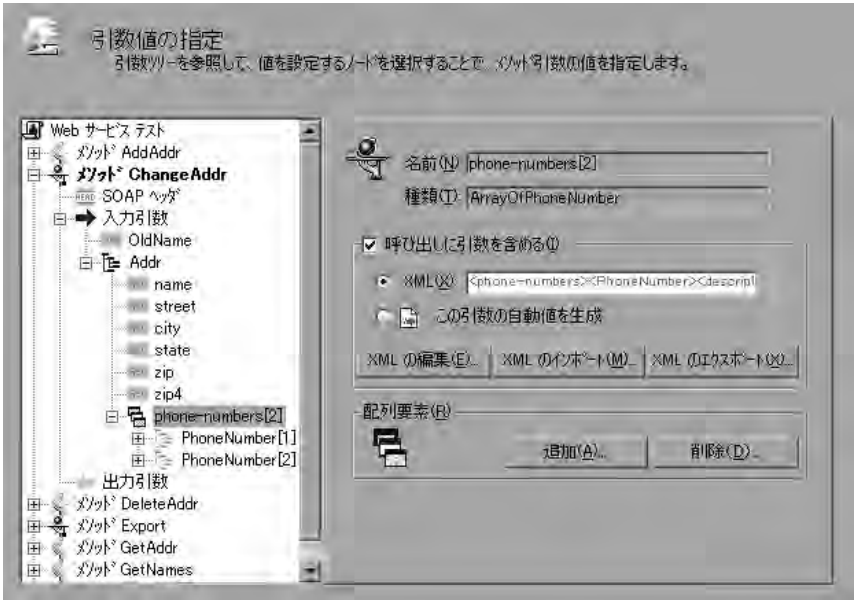
この引数の値が自動的に生成されるようにするには，［**この引数の自動値を生成**］を選択します。

引数に対するパラメータを作成するには，［**値**］ボックスの右隅にある［ABC］アイコンをクリックして［パラメータの選択または作成］ダイアログ・ボックスを開きます。

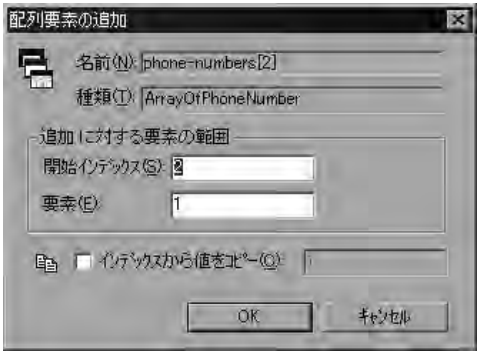
メソッド呼び出しから引数を除外するには，［**呼び出しに引数を含める**］オプションをクリアします。

複合型および配列を変更するには，エントリを選択して［**XML の編集**］をクリックします。XML をインポートするには，［**XML のインポート**］をクリックします。また，選択したエントリを別の XML ファイルにエクスポートするには，［**XML のエクスポート**］をクリックします。詳細については，497 ページ「XML ツリーの編集」を参照してください。

10 配列を操作するには，左側の表示枠で配列をクリックします。



配列要素を追加するには，単純または複合のどちらの場合も，［配列要素］セクションの［**追加**］をクリックします。［配列要素の追加］ダイアログ・ボックスが表示されます。



追加する要素の開始添字と個数を指定します。

既存の配列要素の値を新しい要素に割り当てるには，［**インデックスから値をコピー**］を選択し，使用する値を持つ要素の配列添字を指定します。

［**OK**］をクリックします。各要素が完全な構造で作成されます。

個々の配列要素の値を設定するには，引数を選択して値を挿入します。

配列要素を削除するには，［**削除**］をクリックし，削除する要素の開始添字と個数を指定します。

11 メソッド内の引数の値をすべて表示するには，左側の表示枠でメソッドを選択し，［**引数の表示**］をクリックします。右側の表示枠に，すべての入力引数とその値が一覧表示されます。この一覧には単純な引数値のみ表示されます。複合型の引数値は表示されません。

12 ［**次へ**］をクリックします。情報をすべて指定したことがウィザードから通知されます。スクリプトをすぐに実行するには，［**生成後にスクリプトを実行**］を選択します。実行環境の設定を表示または変更するには，［実行環境の設定］をクリックします。これらの設定の変更は必須ではありません。

13 ［**完了**］をクリックします。生成されたコードが VuGen のエディタに表示されます。選択したメソッドごとに，個別のメソッド呼び出しが生成されます。

14 スクリプトを保存します。

WSDL 文書の管理

VuGen の［WSDL 管理］ウィンドウでは，WSDL ファイルを VuGen の作業リストに追加できます。ファイルをリストに追加すると，VuGen によって作業用のコピーが作成され，スクリプトと一緒に保存されます。VuGen はいくつかの管理ユーティリティを備えています。これらの各ユーティリティについて，スクリプトの作成前および作成後のどちらでも，オプションを設定できます。

［WSDL 管理］ウィンドウを利用して次の作業を実行できます。

- WSDL ファイルの検証
- WSDL ファイルの比較
- WSDL の検証オプションと比較オプションの設定

WSDL ファイルの検証

WSDL ファイルの検証は，スクリプトの作成中に実行するか，スクリプトの作成前に独立に実行できます。検証レポートには，WSDL ファイルの構造や内容に関するすべてのエラー，警告，および通知が表示されます。次の項目が VuGen によって確認されます。

- XML 形式 : WSDL コードが整形式 XML であるかどうか。
- WSDL スキーマ : WSDL ファイルが WSDL スキーマに準拠しているか。必須属性が含まれているか。名前空間が必要に応じて指定されているか。インポート・ファイルおよびインクルード・ファイルが利用可能か。

検証をいくつかのレベルで実行するよう VuGen を設定できます。詳細については，493 ページ「WSDL の検証オプションと比較オプションの設定」を参照してください。

また，VuGen では Web サービスが WS-I（Web Services Interoperability）に準拠しているかどうかをテストすることもできます。プラットフォームを選択して，WS-I 準拠テストの詳細の範囲を指定できます。詳細については，495 ページ「WS-I 検証オプション」を参照してください。

WSDL ファイルを（スクリプト作成時ではなく）独立検証するには，次の手順を実行します。

1 **Web サービス仮想ユーザ・スクリプトを開きます。**

既存または新規の Web サービス・タイプ・スクリプトを開きます。

2 **WSDL ファイルの管理を行います。**

［**WSDL 管理**］ボタン をクリックするか，［**仮想ユーザ**］＞［**WSDL 管理**］を選択します。［WSDL 管理］ダイアログ・ボックスが開きます。

3 **WSDL ファイルを追加します。**

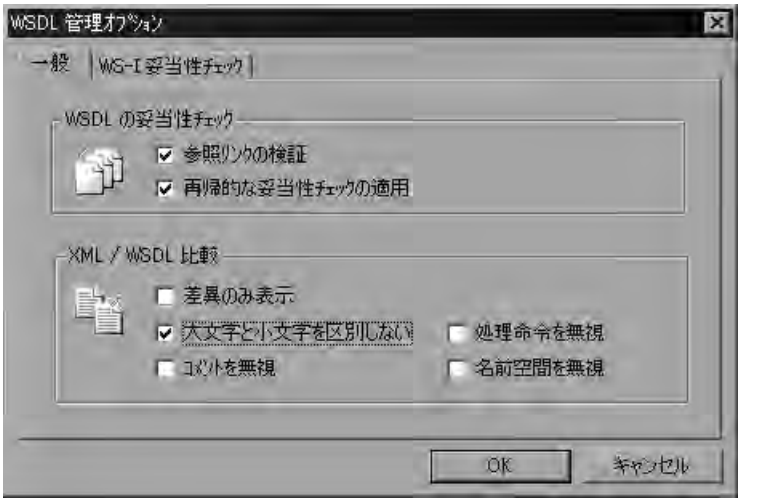
リストの右上隅にある**項目の追加**ボタン をクリックし，WSDL ファイルを探して，検証対象ファイルのリストに追加します。

注：WSDL ファイルをリストに追加すると，VuGen によって作業用のコピーが作成され，スクリプトと一緒に保存されます。標準設定では，検証はすべて作業用コピーに対して実行されます。WSDL ファイルを更新する場合は，ファイルを再度スキャンします。

---

4 **検証レベルを設定します。**

［**オプション**］ボタンをクリックします。［WSDL 管理オプション］ダイアログ・ボックスが開きます。

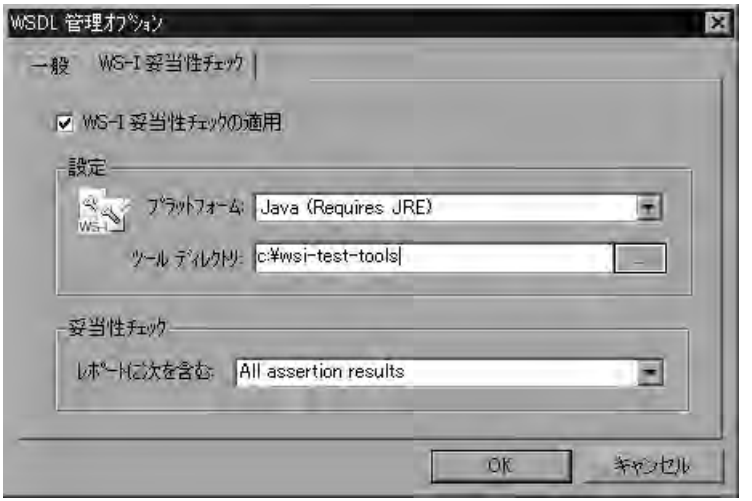


使用するレベルを［WSDL 妥当性チェック］セクションで選択します。

5 **WS-I 検証オプションを設定します。**

［**WS-I 妥当性チェック**］タブをクリックします。WS-I 検証を有効にするには，［**WS-I 妥当性チェックの適用**］を選択します。プラットフォームを選択し，

WS-I 検証ツールのディレクトリを入力します。レポートのタイプを［**レポートに次を含む**］リストから選択します。［**OK**］をクリックします。



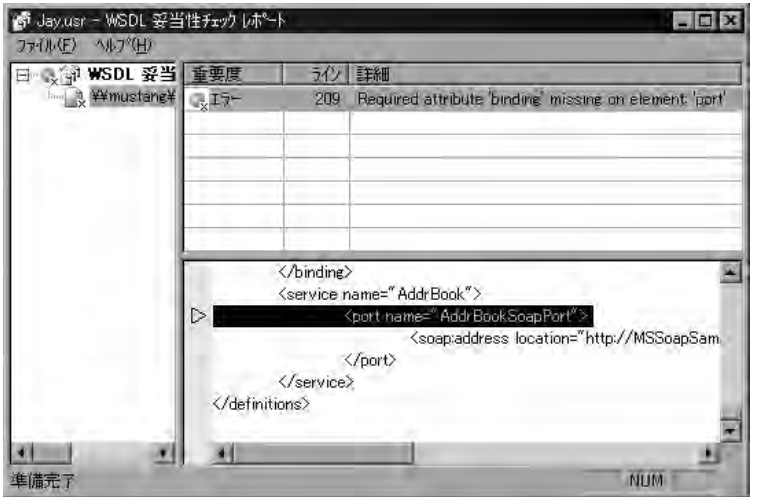
6 **検証対象 WSDL ファイルを選択します。**

［WSDL 管理］ウィンドウで，検証対象となるそれぞれの WSDL ファイルの横のボックスにチェック・マークが付いていることを確認します。チェック・マークを追加または削除するには，ボックスをクリックします。

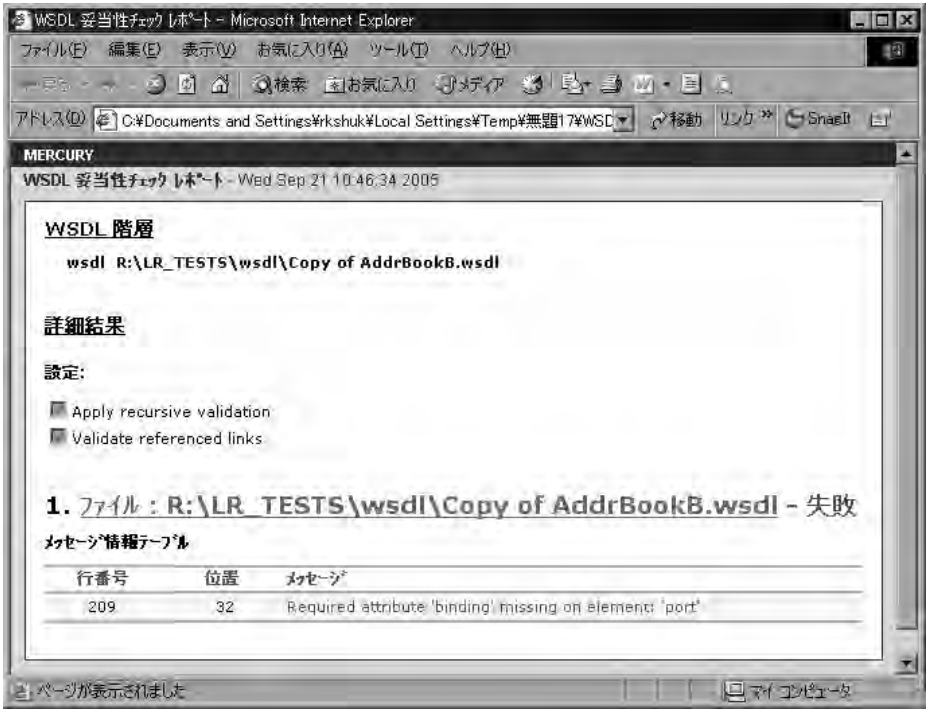
7 **ファイルを検証します。**

［**印の付いているファイルを検証**］をクリックします。［WSDL 妥当性チェック レポート］が開きます。

WSDL ファイルにエラーがある場合は，その詳細が右側の表示枠に表示されます。右上の表示枠でエラーをクリックすると，右下の表示枠に XML WSDL ファイル内のエラーの箇所が表示されます。



HTML 形式の検証サマリ・レポートを開くには，［**ファイル**］＞［**HTML としてエクスポート**］を選択します。ブラウザが開き，HTML 形式の検証結果レポートが表示されます。



8 HTML ファイルを保存するには，ブラウザのウィンドウから［**ファイル**］＞［**名前を付けて保存**］を選択します。

9［WSDL 妥当性チェック レポート］ウィンドウを閉じるには，［**ファイル**］＞［**終了**］を選択します。

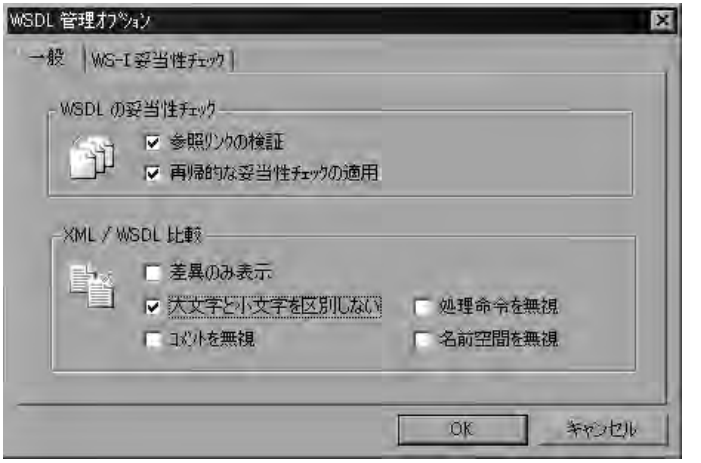
WSDL の検証オプションと比較オプションの設定

次の項目について WSDL 管理のオプションの設定が可能です。

- WSDL 検証オプション
- WSDL 比較オプション

➤ WS-I 検証オプション

オプションを表示するには，［WSDL 管理］ウィンドウ（［**仮想ユーザ**］＞［**WSDL 管理**］）で［**オプション**］をクリックします。



WSDL 検証オプション

WSDL の検証は，スクリプトの作成中に実行するか，［WSDL 管理］ウィンドウを使用して独立に実行できます。この手順では，WSDL 文書内の XML コードを調べ，コードが WSDL の規格に従った有効なものであることを確かめます。

VuGen では WSDL ファイルの検証が 3 つのレベルで提供されています。

➤ **簡易** － XML コードのみを検証し，インポートまたはインクルードされたファイルのコードは検証しません。このレベルを選ぶ場合は，どのチェック・ボックスも選択しないでください。

➤ **参照先リンクの検証** － XML コードを検証し，インポートまたはインクルードされたファイルが存在することと，それらが有効であることを確認します。

➤ **再帰的検証の適用** － XML コード，インポートされたコード，およびインクルードされたコードを，すべて検証します。参照先のインポートされたファイルまたはインクルードされたファイルにある XML コードも検証するほか，それらのコードから参照されるすべての項目も検証します。

WSDL 比較オプション

VuGen では，WSDL 文書のローカル・コピーやグローバル・コピーを比較する際に，いくつかの比較オプションが提供されています。

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

WS-I 検証オプション

WS-I（Web Services Interoperability）は，各種のプラットフォーム，オペレーティング・システム，およびプログラミング言語の間の互換性推進を目的として設立された組織で，WS-I によって Web サービスのための規格が策定されました。

テスト・ツールの入手

VuGen では，WS-I から提供されているツールを使用して，WS-I への準拠をチェックできます。これらのツールは WS-I の Web サイト http://www.ws-i.org/ からダウンロードできます。

C# と .NET，Sun Java など，使用しているプラットフォームに対応したツールをダウンロードしてください。テスト・ツールにはバージョン 1.0 や 1.1 などのいくつかのバージョンも存在します。zip ファイルをダウンロードした後，元のディレクトリ構造 **wsi-test-tools** が維持されるように，ファイルをローカル・ドライブに展開します。

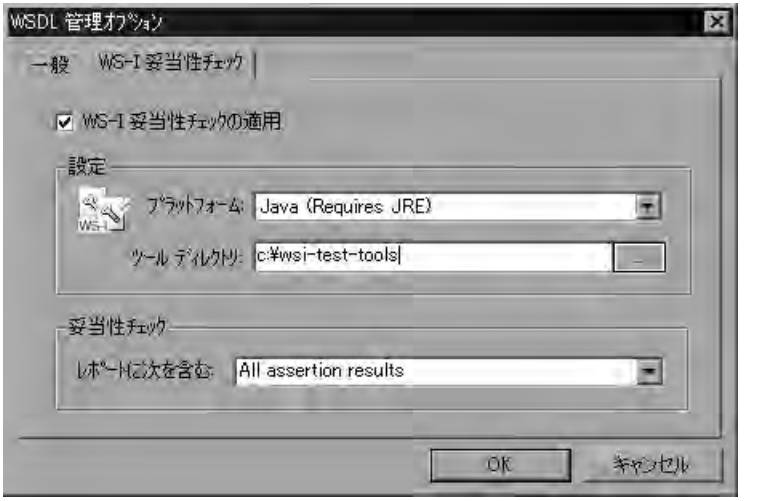
WSI_HOME という環境変数に **wsi-test-tools** のパスの値を定義すれば，VuGen によってパスが認識され，WSDL に対する WS-I 検証が自動的に有効になります。環境変数を定義していない場合は，WS-I 検証を手作業で有効にし，WS-I テスト・ツールの場所を参照します。

WS-I 検証レポートおよびログ

WSDL 用の Web サービスを検証したら，WSDL と WSI の両方の検証レポートを表示できます。

WSI レポートには WSDL が WS-I に準拠しているかどうかの情報が含まれます。

検証レポートの生成時に WS-I 検証を有効にするには，［WSDL 管理］ウィンドウで［**オプション**］をクリックし，［**WS-I 妥当性チェック**］タブをクリックします。［**WS-I 妥当性チェックの適用**］オプションを設定します。



［**プラットフォーム**］: Web サービスのプラットフォーム。Microsoft .NET（.NET Framework が必要），または Java（Sun Java）になります。

［**ツール ディレクトリ**］: WS-I 検証ツール（wsi-testing-tool）のパス。

［**レポートに次を含む**］: レポートに含めるメッセージを指定します。

- ［**All assertion results**］: すべての結果を表示します。
- ［**Assertions with the result "Failed"**］: 結果のステータスが「失敗」のアサーションだけを表示します。
- ［**Assertions with a Result different than"Passed"**］: 結果のステータスが「合格」とならなかったアサーションを表示します。

WSI 検証レポートを開くには，WSDL 検証レポート内のメニューから右クリック・メニューを使用します。

XML ツリーの編集

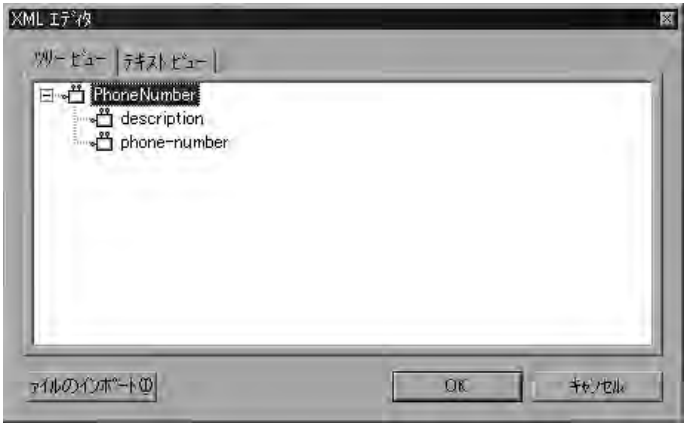
VuGen の XML エディタを使用すると，複合型（構造体やオブジェクトなど）および配列の XML 表現を表示および編集できます。

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

XML 文字列を処理するには，次の手順を実行します。

1 左側の表示枠で複合型または配列を選択します。

2 そのエントリに対応する XML コードを編集するには，［**XML の編集**］をクリックします。［XML エディタ］ウィンドウが開きます。



3 コードをテキスト・モードで編集するには，［**テキスト ビュー**］タブをクリックします。XML コードを手作業で編集します。

4 以前に保存した XML ファイルをインポートするには，［**ファイルのインポート**］をクリックし，ファイルの場所を指定します。ファイルを XML エディタ・ウィンドウで編集します。

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

IDE 統合機能

IDE 統合機能を利用する方法では，クライアント・アプリケーションのコンポーネントを使用してスクリプトを作成できます。例えば，Web サービスで非同期呼び出しに WS セキュリティまたは別の独自の実装を使用している場合，通常はそのトラフィックを記録しても意味のあるスクリプトが生成されることはありません。このような場合に，実際のクライアント・アプリケーション・コードを使用するスクリプトを作成できます。負荷テスト環境の中で，記録を行わずにクライアントを直接実行します。

この方法でスクリプトを作成する場合は，QA およびソフトウェア開発者との連携作業が必要になります。開発者が社内での API のテスト用に作成したテストは，通常は負荷テストにも使用できます。

第 37 章

Web サービス仮想ユーザの実行

Web サービス・スクリプトを作成したら，それを実行して正常に機能することを確かめます。スクリプトを実行した後，テスト結果を表示して，スクリプトが Web サービスと正常に通信できているかどうかを確かめることができます。

本章では，次の項目について説明します。

- Web サービスの実行環境の設定
- WSDL ファイルの比較
- XML ファイルの比較
- スキャンされた WSDL プロパティの設定
- Web サービス・スクリプト・スナップショットの表示
- Web サービス関数の使用
- Web サービス・レポートの表示

Web サービス仮想ユーザの実行について

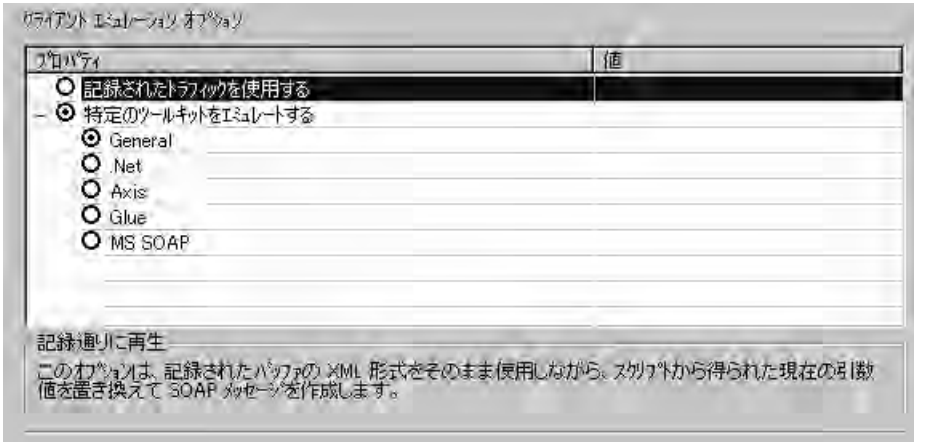
スクリプトを実行すると，仮想ユーザは Web サービスと実ユーザのやり取りをエミュレートします。

スクリプトを実行する前に，スクリプトの効果と正しさを高める目的で，いくつかのことを実行できます。

- **実行環境の設定**：実ユーザをより正確にエミュレートするのに役立つ実行環境の設定が行えます。
- **WSDL の比較**：スクリプトを再生する前に，WSDL ファイルに更新が加えられていないかを調べることをお勧めします。以前に作成したスクリプトの場合には，このことが特に重要になります。

Web サービスの実行環境の設定

クライアント・アプリケーションや特定のツールキットをエミュレートするために，Web サービスに対する実行環境を設定できます。



クライアント・アプリケーションを記録することで作成したスクリプトの場合は（WSDL ファイルを走査することで作成したスクリプトは除きます），記録したスクリプト内のスタイルと属性を使用してクライアントをエミュレートできます。このタイプのエミュレーションを使用するには，［**記録されたトラフィックを使用する**］オプションを有効にします。

WSDL ファイルを走査することで作成したスクリプトの場合は，2 番目のオプションである［**特定のツールキットをエミュレートする**］のみ使用できます。必要なエミュレーションのタイプを選択できます。エミュレート可能な特定のツールキットに加えて，一般タイプのエミュレーションもあります。エミュレーションの詳細については，468 ページ「クライアントのエミュレーション」を参照してください。

Web サービスの実行環境を設定するには，次の手順を実行します。

1 ［実行環境設定］ダイアログ・ボックスを開き（［**仮想ユーザ**］＞［**実行環境の設定**］，または F4 キー），［**Web Services**：**クライアント エミュレーション**］ノードを選択します。

2 使用するエミュレーションまたは一般エミュレーション（［**一般**］）を選択します。

WSDL ファイルの比較

WSDL ファイルをスキャンすると，VuGen によって作業用のコピーが作成され，スクリプトと一緒に保存されます。その結果，リソースが節約され，環境の拡張性が高まります。

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

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

両者の相違が大きければ，［更新］オプションを使用して元のコピーから WSDL を更新できます。

VuGen で，ファイル間の相違が比較レポートに一覧表示されます。［比較レポート］ウィンドウには，［**Working Copy**］と［**Original File**］の 2 つのカラムがあります。［Working Copy］はスクリプトと一緒に格納されている WSDL です。一方，［Original File］は元の場所（ネットワーク・ファイル・パスまたは URL）にある WSDL です。

VuGen はまた，任意の 2 つの XML ファイルを比較できる一般ユーティリティも備えています。詳細については，505 ページ「XML ファイルの比較」を参照してください。

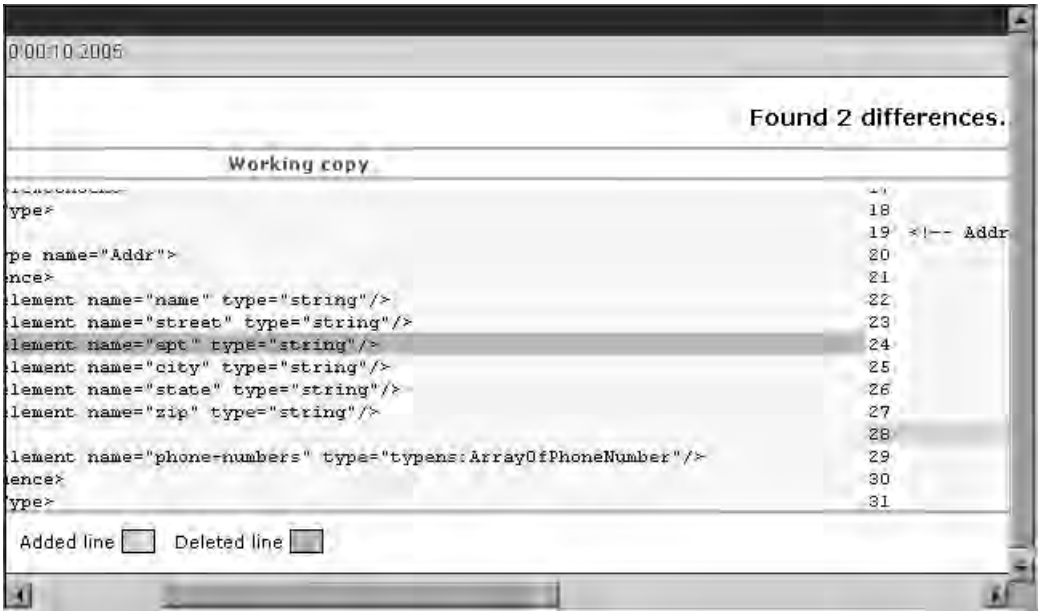
比較レポートでは，2 つのファイル間の相違を示すために次の凡例を使用します。

黄：既存の要素の変更（双方のバージョンに表示）

緑：新しい要素の追加（元のファイル・コピーに表示）

桃：要素の削除（作業用コピーに表示）

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



WSDL 比較オプション

VuGen では，WSDL 文書のローカル・コピーとグローバル・コピーを比較する際に，次の比較オプションが提供されます。

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

WSDL ファイルを比較するには，次の手順を実行します。

1 **Web サービス仮想ユーザ・スクリプトを開きます。**

既存のスクリプトを開きます。

2 **［WSDL 管理］ウィンドウを開きます。**

［**WSDL 管理**］ボタンをクリックするか，［**仮想ユーザ**］＞［**WSDL 管理**］を選択します。［WSDL 管理］ダイアログ・ボックスが開きます。

3 **WSDL ファイルを追加します。**

［**追加**］ボタン をクリックし，WSDL ファイルを探してファイル比較リストに追加します。

4 **比較する WSDL ファイルを選択します。**

比較するそれぞれの WSDL ファイルの横にあるボックス内にチェック・マークがあることを確認します。

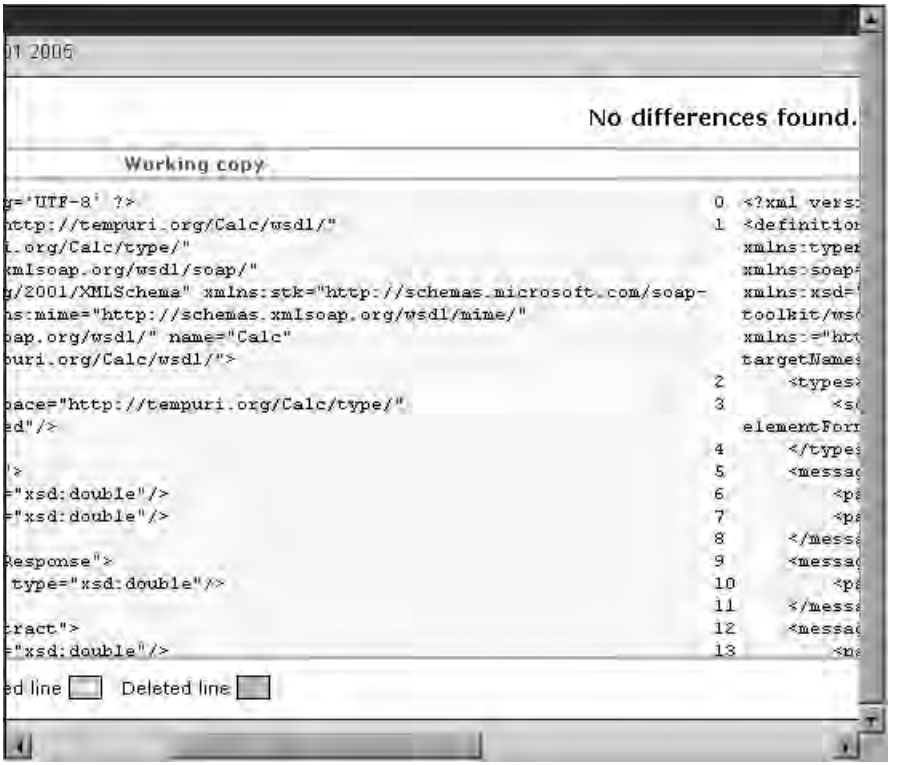
5 **比較オプションを設定します。**

［**オプション**］ボタンをクリックします。［WSDL 管理オプション］ダイアログ・ボックスが開きます。

使用するオプションを［一般］タブの中で選択して，［**OK**］をクリックします。

6 **ファイルを比較します。**

［**印の付いているファイルを比較**］をクリックします。［WSDL 比較レポート］が開きます。



ファイルを下にスクロールして相違点を探します。

7 **作業用コピーを更新します。**

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

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

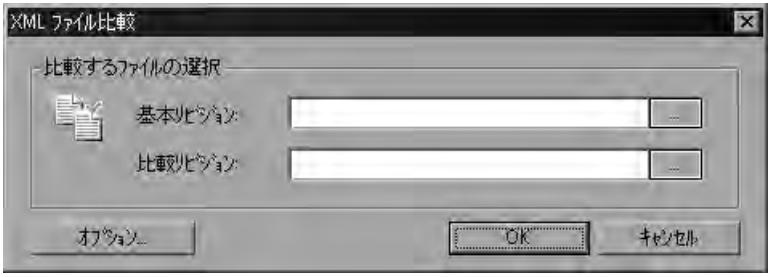
XML ファイルの比較

VuGen は，2 つの XML ファイルを比較するためのユーティリティを備えています。

大文字小文字の区別やコメントなど，無視してよい相違点を指定できます。比較オプションの詳細については，502 ページ「WSDL 比較オプション」を参照してください。

2 つの XML ファイルを比較するには，次の手順を実行します。

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



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

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

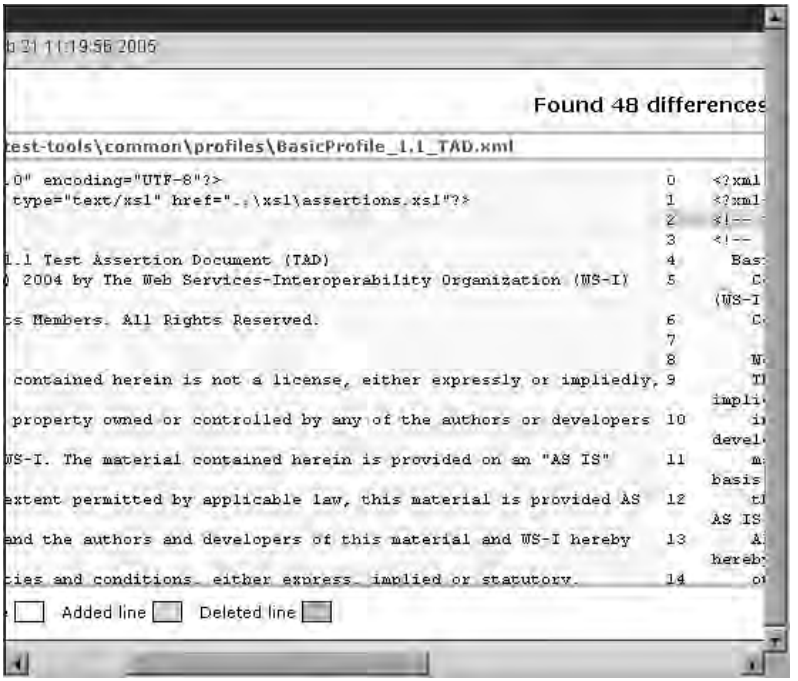
4 ［**OK**］をクリックします。［XML 比較レポート］ウィンドウが開きます。

比較レポートの詳細については，次を参照してください。

XML 比較レポート

VuGen で，ファイル間の相違が比較レポートに一覧表示されます。［比較レポート］ウィンドウには，［基本リビジョン］と［比較リビジョン］の 2 つの

カラムがあります。各カラムのヘッダーには XML ファイルのフル・パスが表示されます。



比較レポートでは次の凡例を使用します。

黄：既存の要素の変更（双方のバージョンに表示）

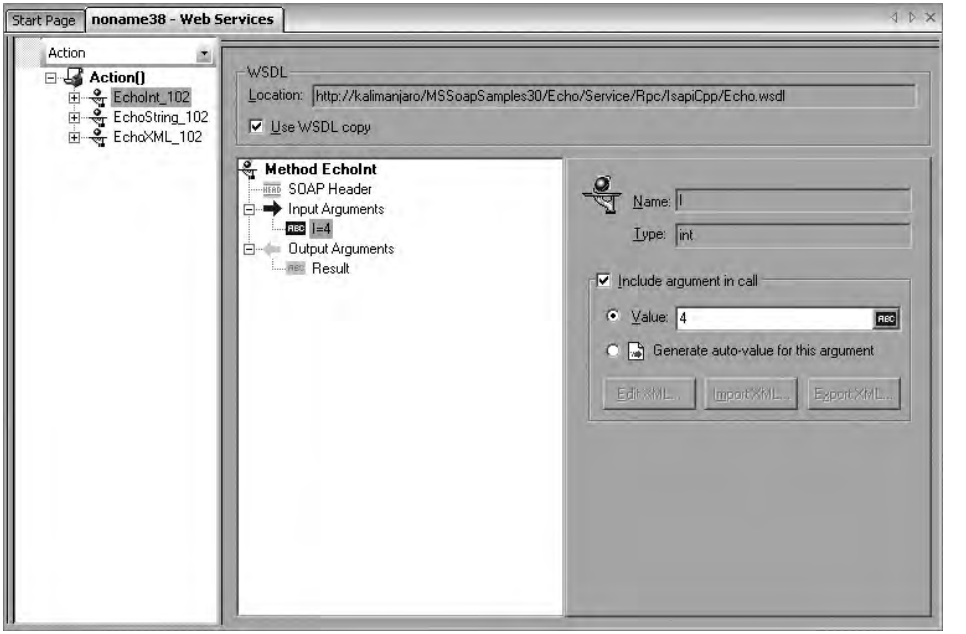
緑：新しい要素の追加（グローバル・バージョンに表示）

桃：要素の削除（ローカル・バージョンに表示）

スキャンされた WSDL プロパティの設定

記録せずに WSDL ファイルをスキャンすると，VuGen のツリー・ビューの右側の表示枠には各要素のプロパティが表示されます。

スキャンされた WSDL をツリー・ビューで表示するには，［**表示**］＞［**ツリービュー**］を選択します。



各引数値を必要に応じて変更します。詳細については，480 ページ「引数値の指定」を参照してください。

Web サービス・スクリプト・スナップショットの表示

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

スクリプトを記録するのではなくスキャンした場合は，スナップショットは表示されません。その代わり，右側の表示枠にメソッドのプロパティが表示され

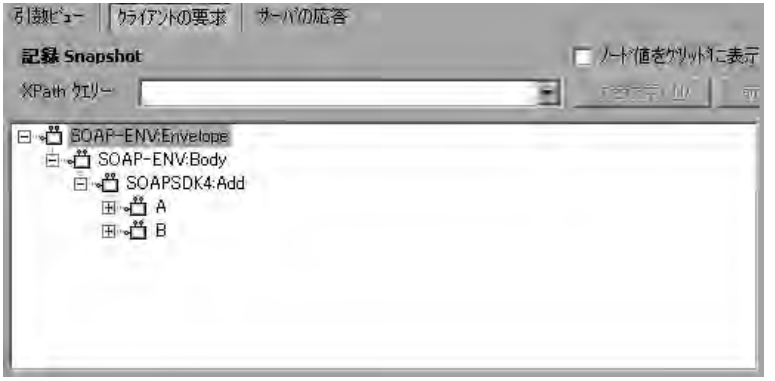
ます。詳細については，「スキャンされた WSDL プロパティの設定」を参照してください。

スナップショットを表示する方法はいくつかあります。

- 引数ビュー
- クライアント要求
- サーバ応答

［**引数ビュー**］には，引数とその値が表示されます。このビューには，Web サービス・セッション中にサーバに送られた値が表示されます。

［**クライアントの要求**］タブには，引数と引数に割り当てた値が，展開可能なツリー階層として表示されます。



［**ノード値をグリッドに表示**］オプションを使用すると，要素とその値をグリッドに表示することもできます。［サーバ応答］ビューで作業をしているときは，グリッドで値を選択し，右クリック・メニューを使用してその値をパラメータ化できます。

［**サーバの応答**］タブには，すべての結果要素が展開可能なツリー階層として表示されます。

［**クライアントの要求**］または［**サーバの応答**］タブを使用すれば，XML 要素とそのプロパティを表示したり（右クリック・メニューを使用します），［**クエリ ビルダ**］ボタンをクリックして XML ツリーに対してクエリを実行したりできます。

この付録では，次の内容について説明します。

- XML ツリーの編集
- web_service_call 関数によって送られた XML の処理
- XML 要素のパラメータ化
- 検証関数の挿入

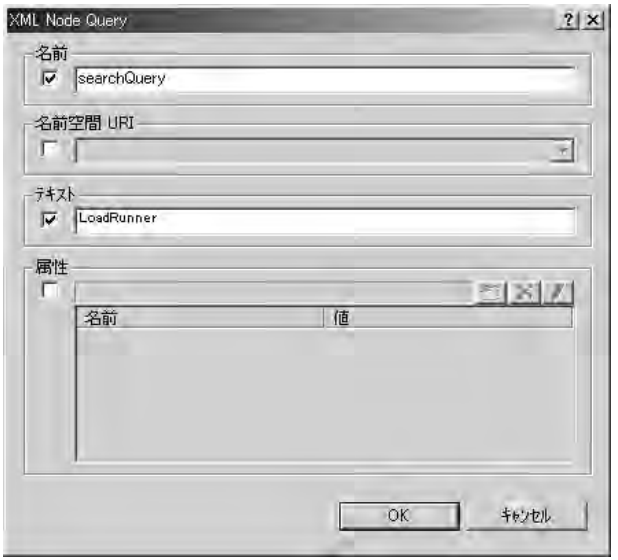
XML ツリーの編集

VuGen は展開可能なツリーに XML コードを表示します。SOAP エンベロープ・パケットの場合は，パケットのさまざまな要素とその値を見ることができます。XML 文書に対するクエリを実行し，特定の名前空間 URI，値，または属性を検索できます。すべてのクエリで大文字と小文字は区別されます。

クエリを実行するには，次の手順を実行します。

1 ［スナップショット］ウィンドウで［**クエリ ビルダ**］をクリックします。［XML Node Query］ダイアログ・ボックスが開きます。

2 1 つまたは複数の項目を検索対象として有効にします。



3［**名前**］セクションを有効にして，ノードまたは要素の名前を検索します。

4［**テキスト**］セクションを有効にして，［名前］ボックスに表示されている要素の値を検索します。

5［**名前空間 URI**］セクションを有効にして，名前空間を検索します。

6［**属性**］セクションを有効にして，属性を検索します。

7 該当するボックスに検索テキストを入力します。属性を追加するには，［**追加**］ボタンをクリックします。［属性のプロパティ］ボックスが開きます。属性の名前と値を入力します。［**OK**］をクリックします。

8［XML Node Query］ダイアログ・ボックスで［**OK**］をクリックします。VuGen によって，クエリ・テキストが［XPath クエリー］ボックスに挿入されます。

9［**次を検索**］をクリックして検索を開始します。

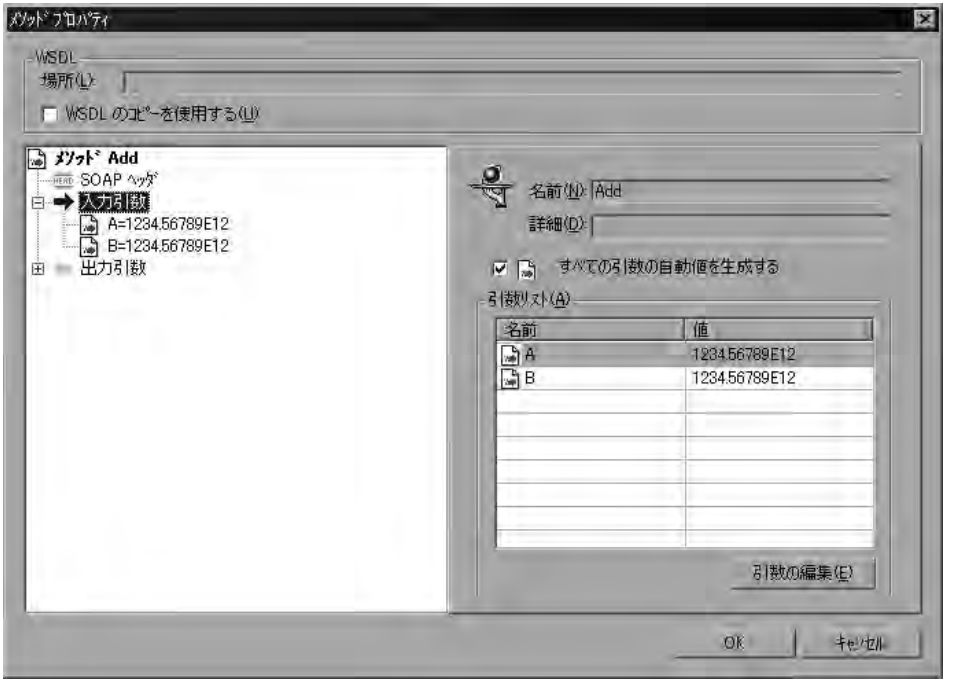
web_service_call 関数によって送られた XML の処理

web_service_call 関数では，Web Service メソッドの一般プロパティを表示したり，その入力および出力引数を変更したりできます。

プロパティを表示または変更するには，次の手順を実行します。

1 ツリー・ビューで，ステップを選択します。

2 ステップをダブルクリックするか，右側の表示枠の一番上にある［**プロパティ**］ボタンをクリックして，［メソッド プロパティ］ダイアログ・ボックスを開きます。

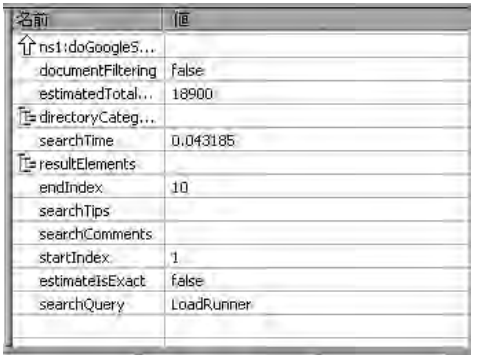


XML 要素のパラメータ化

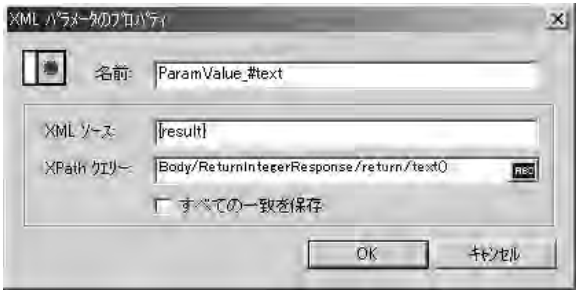
スナップショット・ビューアとグリッドを使用して，XML 文書内の名前とテキストの値をパラメータ化できます。

値をパラメータに置換するには，次の手順を実行します。

1 パラメータ化する引数のツリー・ノードを選択するか，グリッドで値を選択します。



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



3 要素をパラメータ化するには，［**XPath クエリー**］ボックスの［ABC］アイコンをクリックします。［パラメータの選択または作成］ダイアログ・ボックスが開きます。

4 パラメータの名前と種類を指定します。詳細については，『**第 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「VuGen パラメータを使った作業」を参照してください。

検証関数の挿入

スクリプトの実行中に，特定のテキスト文字列が表示されていることを検証したい場合があります。テキストの検証を実行するには，チェックする要素を選択し，XML チェック・ステップを挿入します。VuGen によって，カーソルの位置に **lr_xml_find** 関数が配置されます。

XML チェックを挿入するには，次の手順を実行します。

1 グリッドまたはツリー・ノードの中で，チェックする要素を選択します。

2 右クリック・メニューから［**XML チェックの挿入**］を選択します。［XML チェックのプロパティ］ダイアログ・ボックスが開きます。

要素の値を［値］ボックスに入力します。

3 使用するオプションを選択します。

［**正規表現を使用する**］:（［**UseRegExp**］オプションに対応）検索する値として正規表現を使用できます。

［**エラー時も処理を継続する**］:（［**NotFound**］オプションに対応）検索値が見つからなかった場合でもスクリプトの実行を続けます。クリアした場合，値が見つからなければスクリプトは失敗します。

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 **lr_xml_find** を参照してください。

Web サービス関数の使用

Web サービス・セッションを記録するとき，または WSDL 文書をスキャンするときは，Web サービスのメソッドを呼び出す関数が VuGen によって作成されます。

- **web_service_call** : WSDL 文書をスキャンするか，WSDL ファイルを含む Web サービス・セッション**を記録するとき。**
- **soap_request** : WSDL を指定せずに Web サービス・セッションを記録するとき。

次の例で，2 つの関数間のコードの相違を示します。

web_service_call

```
web_service_call( "StepName=Add_101",
        "SOAPMethod=Calc.CalcSoapPort.Add",
        "ResponseParam=response",
        "WSDL=R:/LR_TESTS/wsdl/Calc.wsdl",
        "UseWSDLCopy=1",
        "Snapshot=t1108909353.inf",
        BEGIN_ARGUMENTS,
        "A=5",
        "B=6",
        END_ARGUMENTS,
        BEGIN_RESULT,
        END_RESULT,
        LAST);
```

上のコードは，**calc.wsdl** ファイルからの WSDL の **Add** メソッドを記述しています。結果のパラメータは，BEGIN_RESULT マーカと END_RESULT マーカの間のリストに置かれます。このコード生成法では，SOAP ヘッダーは使用されません。

スキャン・ウィザードで，［**呼び出しに含める**］オプションを有効にしている場合，VuGen は BEGIN_ARGUMENTS マーカと END_ARGUMENTS マーカの間の入力引数を含めます。

soap_request

```
web_add_header("SOAPAction",
"http://tempuri.org/Calc/action/Calc.Add");

soap_request(
"URL=http://war/MSSoapSamples30/Calc/Service/Rpc/IsapiCpp/Calc.WS
DL",
      "SOAPEnvelope=<?xml version=¥"1.0¥" encoding=¥"UTF-8¥"
standalone=¥"no¥"?><SOAP-ENV:Envelope xmlns:SOAP-
ENV=¥"http://schemas.xmlsoap.org/soap/envelope/¥"><SOAP-
ENV:Body><AddNs:Add
xmlns:AddNs=¥"http://tempuri.org/Calc/message/¥"><A>4</A><B>5</B><
/AddNs:Add>"
            "</SOAP-ENV:Body></SOAP-ENV:Envelope>",
            "Snapshot=t1.inf",
            "ResponseParam=result1",
            LAST);
```

上のコードは，**calc.WSDL** 文書からの WSDL の **Add** メソッドを SOAP 要求として記述しています。このメソッドに関するすべての情報と入力引数は SOAP エンベロープに含まれています。

さらに，Web 関数 **web_<suffix>**，または XML 関数 **lr_xml_<suffix>** を使用して，スクリプトを強化することもできます。詳細については，「**オンライン関数リファレンス**」（［**ヘルプ**］＞［**関数リファレンス**］）を参照してください。

Web サービス・レポートの表示

Web サービス・スクリプトを実行した後，VuGen のテスト結果ユーティリティを使用してテスト結果のサマリを表示できます。テスト結果ユーティリティには，Web 仮想ユーザ・ステップと Web サービス仮想ユーザ・ステップの両方が表示されます。

本項では，サマリ・レポートの Web サービス情報について説明します。テスト結果ユーティリティと利用可能なビューの詳細については，第 33 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。

サマリ・レポートを開くには，［**表示**］＞［**テスト結果**］を選択します。

テスト結果は，反復，アクション，およびステップに分けられます。

結果レポートでは，成功したステップが緑色のチェック印で，失敗したステップが赤色の「**×**」印で，それぞれ示されます。反復は，そのすべてのステップとアクションが成功した場合にのみ，成功したものとして示されます。

Web サービス呼び出しの場合は，［結果］ウィンドウの下の表示枠に SOAP 応答の内容が表示されます。

VuGen がコードを解釈または解析できない場合は，「**Web service failed parsing WSDL（Web サービスによる WSDL の解析に失敗しました）**」というメッセージがレポートに表示されます。

テスト結果での作業の詳細については，第 33 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。

第 38 章

Web/WinSock 仮想ユーザ・スクリプトの記録

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，以下の項目について説明します。

- Web/WinSock 仮想ユーザ・スクリプトの記録について
- 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 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 30 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 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「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・スクリプトの作成工程の概略を説明します。

Web/WinSock 仮想ユーザ・スクリプトを作成するには，次の手順を実行します。

1 **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。標準的な操作を実行します。

詳細については，524 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 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「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 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「VuGen パラメータを使った作業」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 30 章「Web 仮想ユーザ・スクリプトの相関ルールの設定」および『**第 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「ステートメントの相関」を参照してください。

5 **実行環境を設定します。**

実行環境を設定することによって，スクリプト実行時の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 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「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 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「スタンドアロン・モードでの仮想ユーザ・スクリプトの実行」を参照してください。

スクリプトを作成した後，スクリプトを自分の環境（LoadRunner シナリオ，Performance Center 負荷テスト，Tuning Module セッション，またはビジネス・プロセス・モニタ・プロファイル）に統合します。詳細については，『**Mercury LoadRunner コントローラ・ユーザーズ・ガイド**』，**Tuning Console**，**Performance Center**，または **Application Management** のマニュアルを参照してください。

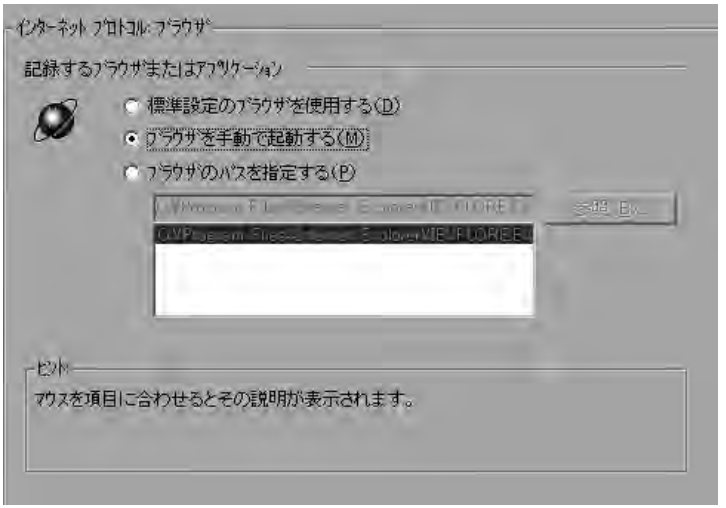
ブラウザとプロキシ記録オプションの設定

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 24 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 11 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（528 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。

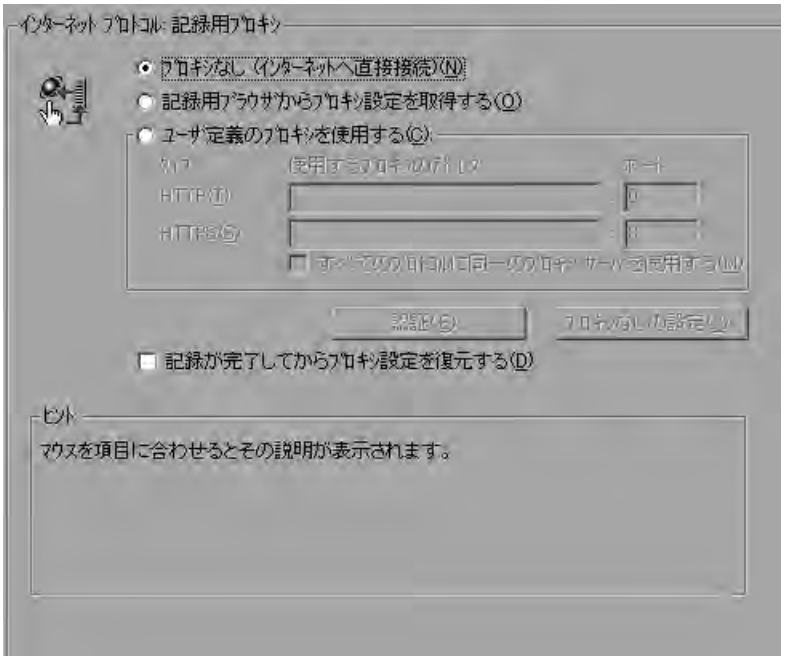
- ［**標準設定のブラウザを使用する**］：記録を行うコンピュータで標準設定の Web ブラウザを使用するよう VuGen に指示する場合に使います。［記録開始］ダイアログ・ボックスの［**記録するアプリケーション**］フィールドのアプリケーションは無視されます。ただし，このフィールドには使用しなくても値を入力する必要があります。ActiveX アプリケーションまたは Java テンプレートを記録するには，このオプションを使用します。
- ［**ブラウザを手動で起動する**］：記録開始時にアプリケーション（この場合，ブラウザ）を自動的に起動しないよう VuGen に指示します。［記録開始］ダイアログ・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。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。

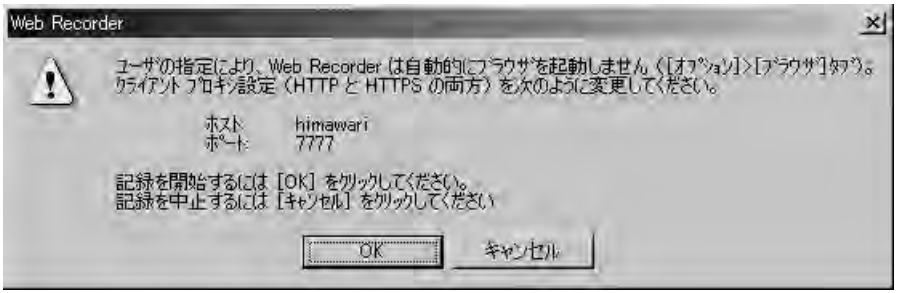
記録用プロキシの設定

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，［**プロキシなしの設定**］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，次の手順を実行します。

➤ 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，［**LAN にプロキシ 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 25 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 11 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，525 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。

例えば，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。例えば，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，525 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，次の手順を実行します。

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，次の手順を実行します。

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**] メニューから Mercury 仮想ユーザ・ジェネレータ（VuGen）を起動します。

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 Dual Protocol**] 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，525 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 28 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 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「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，スクリプト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，『**第 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「VuGen を使った記録」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，次の手順を実行します。

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。

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 8 部

Enterprise Java Bean プロトコル

第 39 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またはチューニング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 テストについて
- EJB Detector での作業
- 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・スクリプトについて
- EJB 仮想ユーザ・スクリプトの実行

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，ユーザ定義 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 の機能をテストまたはチューニングするスクリプトを作成する点です。スクリプトを生成する前に，アプリケーション・サーバに関する JNDI のプロパティなどの情報を指定します。VuGen の EJB Detector は，アプリケーション・サーバ内を検索し，どの EJB が使用可能かを判断します。テストまたはチューニングする EJB を選択すると，VuGen は各 EJB メソッドをエミュレートするスクリプトを生成します。

メソッドごとにトランザクションを作成して，各メソッドのパフォーマンスの測定と問題の特定ができるようにします。さらに，各メソッドは例外処理のために **try / catch** ブロックに入れられます。

EJB テスト・スクリプトを作成するには，EJB Detector がアプリケーション・サーバのホスト上にインストールされてアクティブになっていなければなりません。Detector については，この後の項で説明します。

VuGen には，スクリプトにメソッドを挿入するユーティリティも組み込まれています。このユーティリティを使用して，すべての利用可能なパッケージの表示，使用するメソッドの選択，およびそれらのスクリプトへの挿入ができます。詳細については，554 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 エージェントをインストールするには，次の手順を実行します。

1 アプリケーション・サーバ・マシンまたはクライアント・マシンに EJB Detector のホーム・ディレクトリを作成します（クライアント・マシンに作成する場合は，ファイル・システムを前述のようにマウントします）。

2 EJB Detector フォルダで圧縮ファイル **＜LoadRunner のインストール・フォルダ＞ \ejbcomponent\ejbdetector.jar** を解凍します。

EJB Detector の実行

VuGen で EJB スクリプトの生成を開始する前に，EJB Detector を実行しておく必要があります。EJB Detector はアプリケーション・サーバまたはクライアント・マシンで実行できます（クライアント・マシンの場合は，EJB Detector のクライアント・マシンからアプリケーション・サーバへのマウントを行い，検索ルート・ディレクトリ内にマウント・ディレクトリを指定し，生成されたスクリプトをローカル・マシンではなくマウントされたマシンに接続するように変更します）。

EJB Detector は，コマンド・ラインまたはバッチ・ファイルから実行できます。

EJB Detector をコマンド・ラインから実行するには，次の手順を実行します。

1 コマンド・ラインで EJB Detector を実行する前に，CLASSPATH 環境変数に DETECTOR_HOME¥classes と DETECTOR_HOME¥classes¥xerces.jar を追加します。

2 EJB1.0（Weblogic 4.x，WebSphere 3.x）で作業する場合は，テスト対象の EJB クラスを，次のベンダー EJB クラスとともに CLASSPATH に追加します。

WebLogic 4.x の場合：＜ WebLogic のディレクトリ＞ ¥lib¥weblogicaux.jar

WebSphere 3.x の場合＜ WebSphere のディレクトリ＞ ¥lib¥ujc.jar

3 対象の EJB で使用しているクラス・ディレクトリまたは .jar ファイルがほかにもある場合，それらも CLASSPATH に追加します。

4 コマンド・ラインから EJB Detector を実行するには，次の文字列を使用します。

java EJBDetector [検索ルート・ディレクトリ] [リッスン・ポート]

検索ルート・ディレクトリ EJB を検索する 1 つ以上のディレクトリまたはファイル（セミコロンで区切って指定します）。次のガイドラインに従ってください。
BEA WebLogic Servers 4.x および 5.x：アプリケーション・サーバのルート・ディレクトリを指定します。
BEA WebLogic Servers 6.x：ドメイン・フォルダのフル・パスを指定します。**WebSphere Servers 3.x**：配備された EJB フォルダのフル・パスを指定します。
WebSphere Servers 4.0：**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。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：配備可能な **.ear** ファイルまたは複数の **.ear** ファイルが格納されているディレクトリへのフル・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。

例えば，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，例えばポート 2002 を使用している場合は，「**metal:2002**」と指定します。

EJB Detector をバッチ・ファイルから実行するには，次の手順を実行します。

バッチ・ファイル **EJB_Detector.cmd** を使用して，EJB Detector を起動できます。このファイルは，圧縮ファイル **ejbdetector.jar** を解凍すると，インストールされている EJB Detector のルート・ディレクトリに置かれます。

1 EJB Detector の ルート・ディレクトリで env.cmd を開き，環境に応じて次の変数を変更します。

JAVA_HOME インストールされている JDK のルート・ディレクトリ

DETECTOR_INS_DIR インストールされた Detector のルート・ディレクトリ

APP_SERVER_DRIVE アプリケーション・サーバのインストールをホストするドライブ

APP_SERVER_ROOT 次のガイドラインに従ってください。
BEA WebLogic Servers 4.x および 5.x：アプリケーション・サーバのルート・ディレクトリを指定します。
BEA WebLogic Servers 6.x：ドメイン・フォルダのフル・パスを指定します。
WebSphere Servers 3.x：配備された EJB フォルダのフル・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：配備可能な **.ear** ファイルまたは複数の **.ear** ファイルが格納されているディレクトリへのフル・パスを指定します。

EJB_DIR_LIST（オプション） 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。例えば，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。

例えば，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。

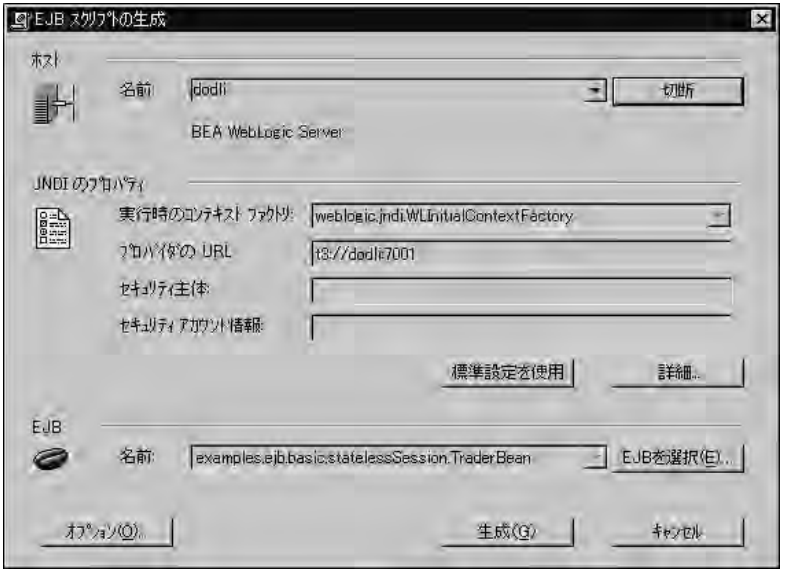
これにより，**webserver.log** ファイルに詳しいデバッグ情報や，その他の潜在的に重要なエラー・メッセージが出力されるようになります。

EJB テストの仮想ユーザの作成

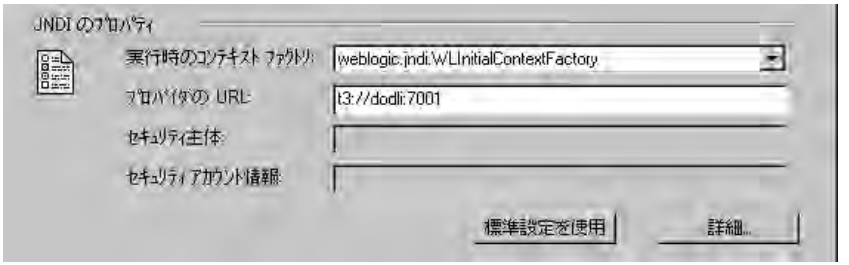
EJB 仮想ユーザ・スクリプトを作成するには，次の手順を実行します。

1［**ファイル**］＞［**新規作成**］を選択するか，［**新規作成**］ボタンをクリックします。［新規仮想ユーザ］ダイアログ・ボックスが開きます。

2［**Enterprise Java Beans**］カテゴリから［**Enterprise Java Beans (EJB)**］を選択し，［**OK**］をクリックします。VuGen が空の Java 仮想ユーザ・スクリプトを開き，［EJB スクリプトの生成］ダイアログ・ボックスが表示されます。



3 VuGen EJB Detector がインストールされているマシンを指定します。接続するためには Detector が稼動していなければなりません。［**接続**］をクリックします。［**JNDI のプロパティ**］セクションが有効になります。



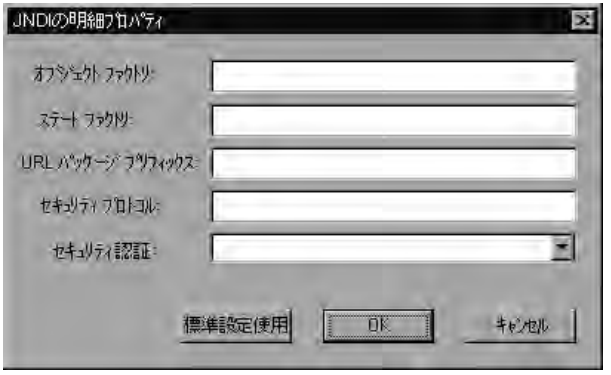
4 EJB Detector は，標準設定の JNDI プロパティを自動的に検出します。これらのプロパティは，編集ボックスで手作業で変更することもできます。変更可能なプロパティは，［**実行時のコンテキスト ファクトリ**］および［**プロバイダの URL**］の文字列です。

アプリケーション・サーバが認証を必要とする場合は，［**セキュリティ主体**］ボックスにユーザ名，［**セキュリティ アカウント情報**］にパスワードを入力します。

次に JNDI の必須プロパティの 2 つの標準設定値を示します。

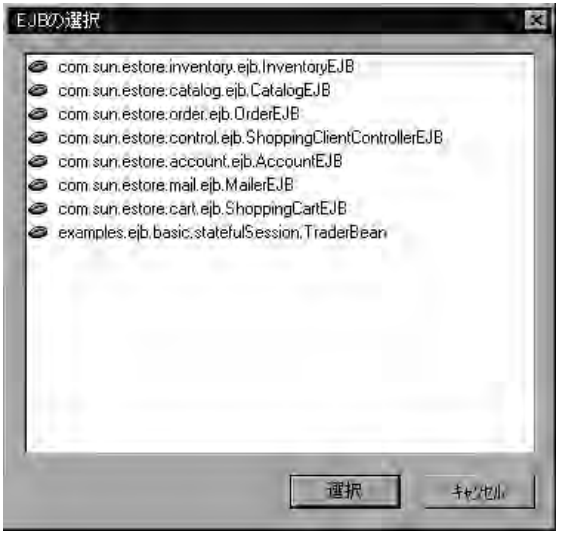
タイプ	実行時のコンテキスト・ファクトリ	プロバイダの URL
WebLogic	weblogic.jndi.WLInitialContextFactory	t3://<appserver_host>:7001
WebSphere 3.x	com.ibm.ejs.ns.jndi.CNInitialContextFactory	iiop://<appserver_host>:900
WebSphere 4.x	com.ibm.websphere.naming.WsnInitialContextFactory	iiop://<appserver_host>:900
Sun J2EE	com.sun.enterprise.naming.SerialInitContextFactory	N/A
Oracle	com.evermind.server.AppplicationClientInitialContextFactory	ormi:// < appserver_host > / < application_name > （< oc4j > /config/server.xml 形式の EJB アプリケーション名）

5 JNDI の詳細プロパティを設定するには，［**詳細**］をクリックして［JNDI の詳細プロパティ］ダイアログ・ボックスを開きます。



必要に応じて，［**オブジェクト ファクトリ**］，［**状態ファクトリ**］，［**URL パッケージの前置記号**］，［**セキュリティ プロトコル**］，［**セキュリティ認証**］プロパティを指定します。［**OK**］をクリックします。

6 ダイアログ・ボックス内の［**EJB**］セクションで［**EJB を選択**］をクリックし，テストを作成する EJB を選択します。ダイアログ・ボックスが開き，現在アプリケーション・サーバからアクセス可能なすべての EJB のリストが表示されます。



7 テスト対象の EJB を強調表示し，［**選択**］をクリックします。

8 ［EJB スクリプトの生成］ダイアログ・ボックスで，［**生成**］をクリックします。VuGen は，Java 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。

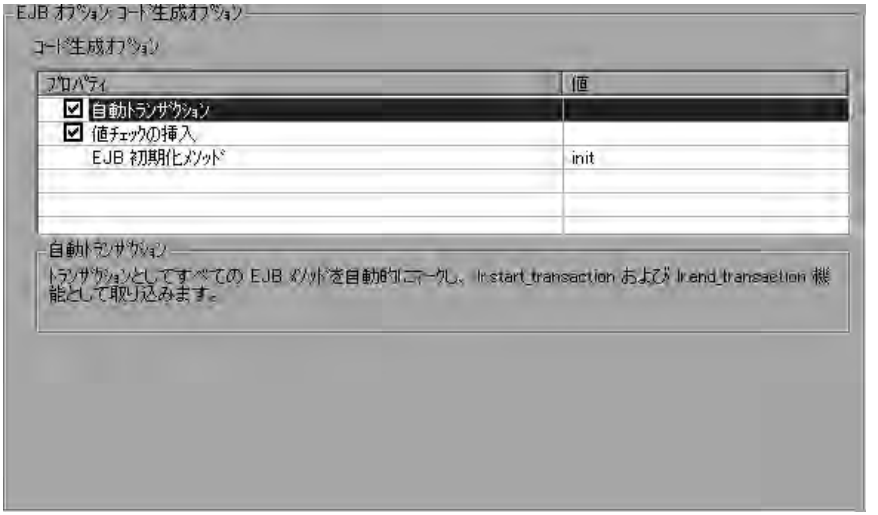
EJB 記録オプションの設定

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 3 章「Java 記録オプションの設定」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を設定するには，次の手順を実行します。

1 ［記録開始］ダイアログ・ボックスの［**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 ［**値チェックの挿入**］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。例えば，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+"]");

      // create Bean instance
      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。次に例を示します。

```
// データベース内の「John」を示す,
// すべての Email EJB インスタンスのリストの取得
Enumeration enum = home.findByName( "John" );
      while ( enum.hasMoreElements() ) {
         Email addr = (Email)enum.nextElement();
            ...
}
```

EJB メソッドの起動

スクリプトの最後の部分には，実際の EJB メソッドが含まれています。各メソッドはトランザクションとしてマークされるので，スクリプト実行後にメソッドを分析できます。さらに，各メソッドは例外処理のために try / catch ブロックに入れられます。例外があった場合には，トランザクションは「**failed**」とマークされ，スクリプトは次のメソッドから続行されます。VuGen は EJB メソッドごとに個別のブロックを作成します。

```
// ------- メソッド ------------

        int _int1 = 0;
        try {
                lr.start_transaction("buyTomatoes");
                _int1 = _carmel.buyTomatoes((int)0);
                //lr.value_check(_int1, 0, lr.EQUALS);
                lr.end_transaction("buyTomatoes", lr.AUTO);
        } catch (Throwable t) {
                lr.end_transaction("buyTomatoes", lr.FAIL);
                t.printStackTrace();
        }
```

VuGen はそれらのメソッドの標準値を挿入します。例えば，整数の場合は 0，文字列の場合は空の文字列（""），複雑な Java オブジェクトの場合は NULL となります。必要に応じて，生成されたスクリプト内の標準値を変更します。

```
                _int1 = _carmel.buyTomatoes((int)0);
```

次の例では，パラメータ化を使用した，複雑なタイプの標準値の変更方法を示しています。

```
Detail details = new Details( < city > , < street > , < zip > , < phone >
);
JobProfile job = new  JobProfile( < department > , < position > , <
job_type > );
Employee employee=new Employee(<first>,<last>, details, job, <salary>);
_int1 = _empbook.addEmployee((Employee)employee);
```

簡単な値や文字列を返すメソッドの場合には，VuGen はコメントアウトされたメソッド **lr.value_check** を挿入します。このメソッドを使用して，EJB メソッドの期待値を指定できます。この検証メソッドを使用するときには，コメント

の記号（//）を削除し，期待値を指定します。例えば，**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。例えば，テキストの前に 2 つのスラッシュ「//」を付けることによって，コメントを挿入できます。

Java 仮想ユーザ・スクリプトは，スケーラブルなマルチ・スレッド・アプリケーションとして稼動します。スクリプトにユーザ定義のクラスを含める場合は，コードをスレッドセーフにする必要があります。コードをスレッドセーフにしないと，結果が不正確になることがあります。スレッドセーフでないコードの場合は，Java 仮想ユーザをプロセスとして実行します。つまり，プロセスごとに個別の Java 仮想マシンを作成します。その結果，スクリプトの拡張性は低くなります。

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，次の手順で実行します。

1 自動的に生成された標準値を変更します。

2 **lr.value_check** メソッドを使用して値の検査を挿入します。これらのメソッドは，スクリプト内にコメントとして生成されます。詳細については，552 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 2 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 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「実行環境の設定」を参照してください。

5 Java VM の実行環境を設定します（追加のクラスパス，VM パラメータをすべて指定します)。必ずアプリケーション・サーバ EJB クラスを入れます。テストされている実際の EJB クラスは仮想ユーザ・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 5 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 ディレクトリから＜ JDK ＞ ¥jre¥lib¥ext ディレクトリにコピーします。iioprt.jar，rmiorb.jar

➤ 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）によってスクリプトに誤りがないか調べられた後，コンパイルが実行されます。

スクリプトを作成した後，スクリプトを自分の環境（LoadRunner シナリオ，Performance Center 負荷テスト，Tuning Module セッション，または Business Process Monitor プロファイル）に統合します。詳細については，『**Mercury LoadRunner コントローラ・ユーザーズ・ガイド**』，**Tuning Console**，**Performance Center**，または **Application Management** のマニュアルを参照してください。

第 9 部

ERP/CRM プロトコル

第 40 章

Web GUI レベル・スクリプトの作成

本項では，Oracle Web Applications 11i と PeopleSoft Enterprise のユーザを対象に，Web GUI レベル・スクリプトの作成について説明します。

本章では，次の項目について説明します。

- Web GUI レベル・スクリプトの作成について
- Web GUI レベル仮想ユーザの概要
- GUI レベル仮想ユーザ関数の使用
- GUI レベル仮想ユーザ・スクリプトについて
- GUI レベル仮想ユーザ・スクリプトで作業する際のヒント

以降の情報は，Oracle Web Applications 11i および PeopleSoft Enterprise プロトコルにのみ適用されます。

Web GUI レベル・スクリプトの作成について

VuGen は，特に Oracle Web Applications 11i と PeopleSoft Enterprise のために，Web GUI レベルでセッションを記録するためのソリューションを備えています。これらのプロトコルは ERP/CRM エンタープライズ・ツールをエミュレートします。企業組織ではこれらのツールを使用することで，会社情報を管理し，共通の環境からビジネス・プロセスを実行できます。

Oracle Web Applications 11i および PeopleSoft Enterprise では，すべてのビジネス・プロセスに Web 経由でアクセスできます。セッションを記録すると，VuGen によってすべての動作状況が記録され，スクリプトが作成されます。再生時には，スクリプトによってブラウザとサーバの間の HTTP プロトコル通信がエミュレートされます。

記録されるページの多くには JavaScript などの HTML 以外のコードが含まれています。URL ベースまたは HTML ベースのモードで記録すると，VuGen によって，JavaScript がページの **web_url** 関数の部分リソースとして追加されます。Web-GUI モードでは，ソース・コード内の JavaScript を正確に解釈するオブジェクト指向スクリプトが VuGen によって作成されます。

VuGen で，Web インタフェース内の操作を直感的に表現する GUI レベル・スクリプトが作成されます。例えば，情報を送信するボタンをクリックすると **web_button** 関数が生成され，テキストをエディット・ボックスに入力すると **web_edit_field** 関数が生成されます。

Web GUI レベル仮想ユーザの概要

本項では，VuGen を使って Oracle Web Applications 11i または PeopleSoft Enterprise のスクリプトを作成するためのプロセスの概略を説明します。これらの手順のほか，570 ページ「GUI レベル仮想ユーザ・スクリプトで作業する際のヒント」も参照することをお勧めします。

Oracle Applications または PeopleSoft 8 のスクリプトを作成するには，次の手順を実行します。

1 **VuGen を使ってアクションを記録します。**

VuGen を起動し，マルチプロトコルまたはシングルプロトコルで新しいファイルを作成します。このとき，Oracle Web Applications 11i または PeopleSoft Enterprise のいずれかをタイプとして ERP/CRM カテゴリから指定します。セッションの中で一般的な操作を記録します。詳細については，『**第 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「VuGen を使った記録」を参照してください。

2 **スクリプトを拡張します。**

トランザクション，ランデブー・ポイント，および制御フロー構造を挿入することによって，スクリプトを拡張します。詳細については，『**第 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「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）。**

記録された固定値に対してパラメータを定義します。固定値をパラメータで置き換えることにより，同じビジネス・プロセスを，異なる値で何度も繰り返すことができます。詳細については，『**第 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「パラメータの作成」を参照してください。

4 **実行環境を設定します。**

実行環境を設定することによって，テスト実行中のスクリプトの動作を制御します。この設定には，ペースの設定，ログ，思考遅延時間，接続情報が含まれます。詳細については，『**第 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「実行環境の設定」を参照してください。

5 **VuGen でスクリプトを保存して実行します。**

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，570 ページ「GUI レベル仮想ユーザ・スクリプトで作業する際のヒント」および『**第 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の 570 ページ「GUI レベル仮想ユーザ・スクリプトで作業する際のヒント」を参照してください。

スクリプトを作成した後，スクリプトをテスト（LoadRunner シナリオ，Performance Center 負荷テスト，Tuning Module セッション，またはビジネス・プロセス・モニタ・プロファイル）に統合します。詳細については，『**Mercury LoadRunner コントローラ・ユーザーズ・ガイド**』，**Tuning Console**，**Performance Center**，または **Application Management** のマニュアルを参照してください。

GUI レベル仮想ユーザ関数の使用

Oracle Web Applications 11i または PeopleSoft 8 の記録セッション中，VuGen によってブラウザとサーバの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，**web** という接頭辞が付きます。

関数リスト

GUI レベルの記録に固有の関数のリストを次に示します。他の **web** 関数の構文などの詳細については，「**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	詳細
web_browser	ブラウザ上でアクションを実行します。
web_button	ユーザによるボタンのクリック，送信，または入力要素のリセットをエミュレートします。

関数名	詳細
web_check_box	チェック・ボックスを選択します。
web_edit_field	入力タイプがテキストまたはパスワードであるフィールドにデータを入力します。
web_dump_cache	ブラウザのキャッシュをファイルに保存します。
web_eval_java_script	JavaScript または DOM オブジェクトを評価します。
web_file	ファイル入力タイプに対してパスを入力します。
web_image_link	ハイパーテキスト・リンクとなっている画像に対するユーザのクリックをエミュレートします。
web_image_submit	画像を送信します。
web_list	リストから項目を選択します。
web_load_cache	ブラウザのキャッシュをファイルから復元します。
web_map_area	クライアント側マップ内部の特定領域をアクティブにします。
web_radio_group	ラジオ・ボタン・グループからボタンを 1 つ選択します。
web_reg_dialog	以降の JavaScript 呼び出し用のデータを設定します。
web_static_image	静止画像に対するユーザのクリックをエミュレートします。
web_text_area	入力テキスト領域にテキストを入力します。
web_text_link	ハイパーテキスト・リンクに対するユーザのクリックをエミュレートします。

API 全般に関する注意事項

以降の各項では，GUI レベル関数全般に関する注意事項について説明します。任意のオブジェクト記述について正規表現を指定できます。そのためには，テキストの前に「/RE」と等号を付けます。次に例を示します。

```
web_text_link("Manage Assets",
        DESCRIPTION,
        "Text/RE=Manage Assets",
            LAST);
```

序数

上記の関数では **Ordinal** 属性を使用します。この属性は，まったく同じ引数値を持つ関数の出現に対応した，1 から始まるインデックスです。次の例では，記録された **web_text_link** 関数は，序数を除いてまったく同じ引数を持っています。序数値の 2 は，2 番目の出現であることを示しています。

```
web_text_link("Manage Assets",
        DESCRIPTION,
        "Text=Manage Assets",
        "FrameName=main",
        LAST);

    web_text_link("Manage Assets_2",
        DESCRIPTION,
        "Text=Manage Assets",
        "Ordinal=2",
        "FrameName=main",
        LAST);
```

空の文字列

引数を指定しないことと，引数を空の文字列として指定することとは，別の意味になります。引数を指定しないときは，標準設定値が使用されるか，または引数が無視されます。引数を列挙したものの，値として空の文字列を割り当てたときは，空の文字列に一致するものか，文字列がまったくないものとして一致するものを見つけようとします。例えば，**id** 引数を省略すると，HTML 要素の **id** プロパティを無視するよう VuGen に指示することになります。"ID=" と指定した場合は，**id** プロパティを持たない HTML 要素か，空の ID を持つ HTML 要素を検索します。

ブラウザまたはフレームの識別引数

- 複数のブラウザで記録を行うときは，**BrowserTitle** または **BrowserOrdinal** 引数を指定する必要があります。
- ブラウザの内部に複数のフレームが存在するときは，**FrameName** または **FrameHierarchyLevel** 引数を指定する必要があります。
- ブラウザ関連の引数はすべて，関数の **DESCRIPTION** セクションの最後に置きます。フレーム関連の引数は，**DESCRIPTION** セクションの最後で，ブラウザ関連の引数の直前に置きます。

GUI レベル仮想ユーザ・スクリプトについて

通常，GUI レベル仮想ユーザ・スクリプトには，ビジネス・プロセスを構成する複数のアクションが含まれています。GUI レベルで生成され記録された関数を確かめることによって，記録セッション中のユーザの正確なアクションを知ることができます。

例えば，PeopleSoft Enterprise の記録では，第 1 段階にサインインのプロセスが含まれています。ブラウザに PeopleSoft サーバの開始ページが開き，ユーザはユーザ名とパスワードを送信して［**Sign In**］をクリックすることでサインインします。

エディット・フィールドにデータを入力すると，VuGen によって **web_edit_field** 関数が生成されます。次の例では，ユーザは **userid** テキスト・

フィールドに VP5 と入力し，**pwd** フィールドにパスワードを入力しています。パスワードは暗号化されます。

```
vuser_init()
{
    web_url("about:blank",
        "URL=http://psft1/servlets/iclientservlet/peoplesoft8/?cmd=login",
        LAST);

    web_edit_field("userid",
        DESCRIPTION,
        "Type=text",
        "Name=userid",
        ACTION,
        "SetValue=VP5",
        LAST);

    web_edit_field("pwd",
        DESCRIPTION,
        "Type=password",
        "Name=pwd",
        ACTION,
        "SetEncryptedValue=3e52677bda7f4b",
        LAST);
...
```

ボタンをクリックしてデータを送信すると，VuGen によって **web_button** 関数が生成されます（ボタンが画像の場合は **web_image_submit** が生成されます）。次の例では，ユーザは［**Sign In**］ボタンをクリックしています。

```
...
    web_button("Sign In",
        DESCRIPTION,
        "Type=submit",
        "Tag=INPUT",
        "Value=Sign In",
        LAST);
    return 0;
}
```

次のセクションでは，ユーザが「**Manage Assets**」分岐の下にある **Asset ExpressAdd** プロセスに移動するという，一般的なアクションを示しています。ユーザは，対象となる分岐のテキスト・リンクをクリックすることで移動します。すると，**web_text_link** 関数が生成されます。

```
web_text_link("Manage Assets_2",
        DESCRIPTION,
        "Text=Manage Assets",
        "Ordinal=2",
        "FrameName=main",
        LAST);

web_text_link("Use",
        DESCRIPTION,
        "Text=Use",
        "FrameName=main",
        LAST);

web_text_link("Asset ExpressAdd",
        DESCRIPTION,
        "Text=Asset ExpressAdd",
        "FrameName=main",
        LAST);
```

次のセクションでは，関数によって，フィールドの設定やリスト項目の選択など，一般的なユーザ・アクションがエミュレートされます。

```
web_edit_field("ASSET_DESCR",
        DESCRIPTION,
        "Type=text",
        "Name=ASSET_DESCR",
        "FrameName=main",
        ACTION,
        "SetValue=Car",
        LAST);
...
web_list("Year",
        DESCRIPTION,
        "Name=Year",
        "FrameName=CalFrame",
        ACTION,
        "Select=2000",
        LAST);
```

画像マップに関連付けられた画像をクリックすると，VuGen によって **web_map_area** 関数が生成されます。

```
web_map_area("map2_2",
        DESCRIPTION,
        "MapName=map2",
        "Ordinal=20",
        "FrameName=CalFrame",
        LAST);
```

GUI レベル仮想ユーザ・スクリプトで作業する際のヒント

記録のヒント

記録中は，ブラウザの表示枠内にある GUI オブジェクトだけを使用してください。ブラウザのアイコン，コントロール，または［停止］，［更新］，［ホーム］などのメニュー項目は使用しないでください。ただし，アドレス・バー，［戻る］ボタン，および［進む］ボタンは，使用しても構いません。

再生のヒント

- 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。
- VuGen のキャッシュのシミュレート機能を利用すると，仮想ユーザのパフォーマンスを向上できます。詳細については，295 ページ「キャッシュを使用したパフォーマンスの向上」を参照してください。

第 41 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 仮想ユーザの開発の概要
- ガイドラインの記録
- 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
- 仮想ユーザの実行環境設定
- 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
- その他に推奨される相関
- 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

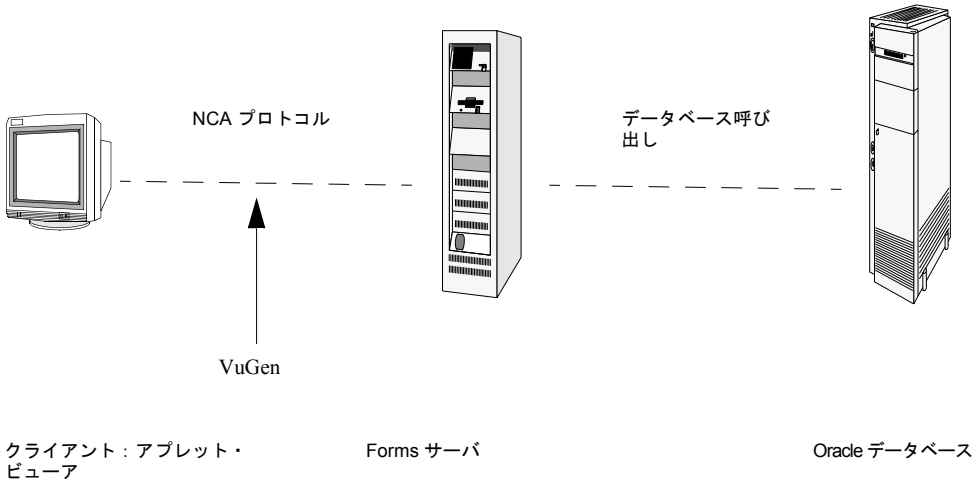
Oracle NCA は，Java ベースのデータベース・プロトコルです。データベース・クライアントであるアプレット・ビューアは，ブラウザを使用して起動します。NCA データベースに対するアクションは，アプレット・ビューアを使用して実行します。

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分かっている場合は，最初から Oracle NCA プロトコルと Web プロトコルを対象としたマルチ・プロトコル・スクリプトを作成します。

最初に Oracle NCA プロトコルを対象としたシングル・プロトコル・スクリプトを作成し，後でテストのために Web 関数が必要となった場合は，セッションを再記録しなくても VuGen でスクリプトを再生成して Web 関数を追加できます。これは，［スクリプトの再生成］ダイアログ・ボックスの［プロトコル］ノードで指定します。詳細については，『**第 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「VuGen を使った記録」を参照してください。

Oracle NCA 仮想ユーザの開発の概要

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 **記録するマシンが正しく設定されていることを確認します。**

VuGen を起動する前に，Oracle NCA アプレット・ビューアが動作するようにマシンが設定されていることを確認します。また，VuGen がお使いの Oracle Forms のバージョンをサポートしていることを確認する必要があります。詳細については，574 ページ「ガイドラインの記録」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，『**第 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「VuGen を使った記録」を参照してください。

3 **一般的なユーザ・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，『**第 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「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。**

［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 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「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で置き換えます。詳細については,『**第 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「パラメータの作成」を参照してください。

6 **スクリプトの実行環境プロパティを設定します。**

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 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「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。**

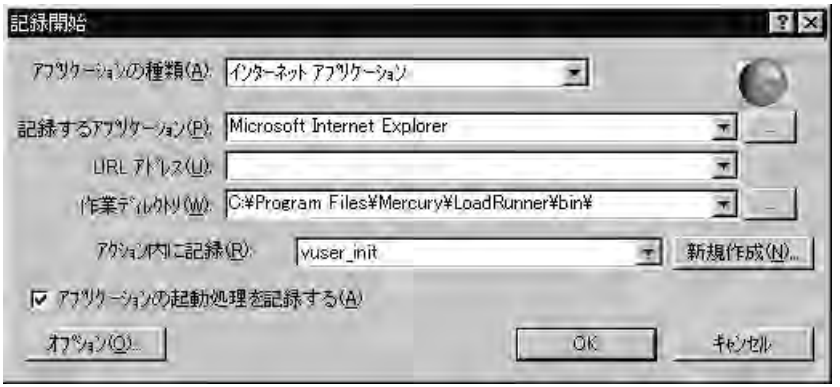
VuGen からスクリプトを実行して，実行時の情報を実行ログで確認します。詳細については,『**第 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「スタンドアロン・モードでの仮想ユーザ・スクリプトの実行」を参照してください。

ガイドラインの記録

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，60 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"/*version=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，**mdrv.dat** ファイルに以下を設定します。

```
[Oracle_NCA]
ExtPriorityType=protocol
WINNT_EXT_LIBS=ncarp110.dll
WIN95_EXT_LIBS=ncarp110.dll
LINUX_EXT_LIBS=liboranca.so
SOLARIS_EXT_LIBS=liboranca.so
HPUX_EXT_LIBS=liboranca.sl
AIX_EXT_LIBS=liboranca.so
LibCfgFunc=oracle_gui_configure
UtilityExt=lrun_api
```

Forms 6 または 9 のサポートに戻すには，最初の値に戻します。

名前によるオブジェクトの記録の有効化

Oracle NCA スクリプトを記録するときには，標準ブジェクト ID ではなくオブジェクト名を使用してセッションを記録する必要があります。オブジェクト ID はサーバによって動的に生成され，記録時と再生時とでは異なるため，オブジェクト ID を使用してスクリプトを記録すると，再生は失敗します。**nca_connect_server** ステートメントを調べれば，スクリプトがオブジェクト名で記録されているかどうかを確認できます。

```
nca_connect_server("199.35.107.119","9002"/*version=11i*/,"module=/d1/oracle
/visappl/fnd/11.5.0/forms/US/FNDSCSGN userid=APPLSYSPUB/PUB@VIS
fndnam=apps record=names ");
```

nca_connect_server 関数に **record=names** 引数が含まれていなければ，オブジェクト ID が記録されているということです。オブジェクト名を記録するように VuGen を設定するには，次のいずれかを変更します。

➤ スタートアップ HTML ファイル

- 記録する URL
- Forms 設定ファイル

すべてのオブジェクトの開発者名を取得する機能は，Oracle Forms6i Patch 9（Oracle Forms Version: 6.0.8.18.3）で初めて導入されました。そのため，Oracle Forms 6i Patch 9 のリリース前に作成された Test Starter Kit スクリプトでは，編集フィールドを除き，オブジェクトの物理的記述に開発者名は含まれません。

スタートアップ HTML ファイル

スタートアップ HTML ファイルにアクセスできる場合は，そのスタートアップ・ファイル，つまり Oracle NCA アプリケーションを起動したときにロードされるファイルに **record=names** フラグを設定すれば，オブジェクト ID ではなくオブジェクト名が記録されます。

アプレット・ビューアの起動時に呼び出されるスタートアップ・ファイルを編集します。次の行を変更します。

＜ PARAM name="serverArgs ..... fndnam=APPS" ＞

次に，Oracle キー「record=names」を次の形式で追加します。

<PARAM name="serverArgs... fndnam=APPS **record=names**">

記録する URL

スタートアップ HTML ファイルにアクセスできない場合は，記録する URL を変更することで，Oracle NCA がオブジェクト ID ではなくオブジェクト名を記録するようにできます。次の方法は，スタートアップ HTML ファイルがロード時にほかのファイルを参照しない場合にのみ有効です。

この方法では，［記録開始］ダイアログ・ボックスの URL の後，つまり記録する URL 名の後に「**?record=names**」を追加します。これにより，VuGen はセッションの中でオブジェクト名を記録できるようになります。

Forms 設定ファイル

Forms Web CGI 設定ファイル **formsweb.cfg** を参照するスタートアップ HTML ファイルがアプリケーションにある場合（よく行われる参照です）は，スタートアップ・ファイルに **record=names** を追加すると，問題が生じる場合があります。

この場合は，設定ファイルに変更を加える必要があります。

設定ファイルに変更を加えてオブジェクト名を記録できるようにするには，次の手順を実行します。

1 Forms Web CGI 設定ファイルに移動します。

2 このファイルに新しいパラメータを定義します（次に示す Web CGI 設定ファイルの例を参照）。

```
serverApp=forecast
serverPort=9001
serverHost=easgdev1.dats.ml.com
connectMode=socket
archive=f60web.jar
archive_ie=f60all.cab
xrecord=names
```

3 スタートアップ HTML ファイルを開き，PARAM NAME="serverArgs" を探します。

4 **record=%xrecord%** のように，変数名を ServerArgs パラメータに引数として追加します。

```
<PARAM NAME="serverArgs" VALUE="module=%form%
userid=%userid% %otherParams% record=%xrecord%">
```

5 または，Oracle Forms のインストール・ディレクトリにある **basejini.htm** ファイルを編集します。このファイルは，JInitiator スタイルのタグを使用して Forms アプレットを読み込む Web 上のフォームを実行するための標準の HTML ファイルです。basejinin.hmt ファイルで，パラメータ定義に次の行を追加します。

＜ PARAM NAME="recordFileName" VALUE="%recordFileName%" ＞

<EMBED> タグに次の行を追加します。

```
...
serverApp="%serverApp%"
logo="%logo%"
imageBase="%imageBase%"
formsMessageListener="%formsMessageListener%"
recordFileName="%recordFileName%"
```

サーブレット設定ファイル **formsweb.cfg** ではなく basejini.htm ファイルを編集することの難点は，Oracle Forms を再インストールすると，このファイルが置き換えられてしまう点です。これを避けるには，**basejini.htm** ファイルのコピーを作成し，そのコピーを別の場所に保管しておきます。サーブレット設定ファイルで，**baseHTMLJinitiator** パラメータを編集して新しいファイルを指すようにします。

Personal Home Page からの Oracle Applications の使用

Personal Home Page にログインして Oracle Forms 6i を起動する場合，ユーザ・レベルでいくつかのシステム・プロファイル・オプションを設定する必要があります。これらの変数は，全ユーザに適用されるサイト・レベルではなく，ユーザ・レベルで設定することをお勧めします。

「ICX:Forms Launcher」プロファイルを設定するには，次の手順を実行します。

1 アプリケーションにサインオンし，［System Administrator］の権限を選択します。

2［Navigator］メニューから［**Profile/System**］を選択します。

3［**Find System Profile Values**］フォームで次の操作をします。

［**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%

［**検索**］をクリックします。

4「**ICX:Forms Launcher**」プロファイルに対する User の値を更新します。

URL に対してパラメータが渡されていない場合，ユーザ指定値の後ろに次の文字列を追加します。

?play=&record=names

URL に対してパラメータが渡されていれば，ユーザ指定値の後ろに次の文字列を追加します。

&play=&record=names

5 トランザクションを保存します。

6 Oracle Forms セッションからログアウトします。

7 Personal Home Page セッションからログアウトします。

8 **Personal Home Page** から，自分の名前を使って再度サインオンします。

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て，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	編集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_pressss_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	オブジェクト内をクリックします。

nca_obj_mouse_dbl_click	オブジェクト内を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	指定された行数だけスクロールします。
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 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「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 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「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，581 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        =m
ult",      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。

仮想ユーザの実行環境設定

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 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「実行環境の設定」を参照してください。ネットワーク速度に関係する設定については，『**第 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「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。

次の項目を設定できます。

ソケット・モード

クライアントとの通信がより高速な HTTP レベルではなくソケット・レベルで実行されます。

［**タイムアウト（秒）**］：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**］は **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，596 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

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。

［**ハートビートを有効にする**］：標準設定では，ハートビートはサーバへ送信される信号です。これを無効にするには，チェックボックスをクリアします。

［**頻度**］：ハートビート信号の頻度です。標準設定は 120 秒です。

Forms

記録時に検出された Oracle Forms のバージョンを示します。

［**バージョン**］：この設定は，記録を行った後にサーバがアップグレードされた場合にのみ変更します。

診断

Oracle Applications のデータベース層の診断モジュールに関する情報を提供します。

［**アプリケーションのバージョン**］: Oracle Application のバージョン。このオプションは，Oracle Application を使用する場合に使用できます（ユーザ定義の Oracle NCA アプリケーションでは使用できません）。これは，Oracle データベース・ブレークダウンを使用するばあいにのみ必要です。

クライアントのエミュレーションを設定するには，次の手順を実行します。

1 ［実行環境設定］ダイアログ・ボックスを開きます。［**仮想ユーザ**］＞［**実行環境の設定**］を選択するか，VuGen のツールバーで［**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 **lfError** メッセージの再試行回数［**最大エントリ数**］を指定します。標準設定は 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**Target Server:** Oracle Forms サーバの IP アドレスまたはロング・ホスト名［**Target Port**］: 443［**接続の種類**］:SSL［**SSL バージョン**］: 使用している SSL のバージョン。不明な場合は「SSL 2/3」を選択します。

詳細については,『**第 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「ポートの割り当て設定」を参照してください。

➤ **nca_connect_server** コマンド実行中に NCA HTTPS スクリプトを再生するときに問題が生じた場合は，スクリプトの先頭に次の関数を挿入します。

```
web_set_sockets_option("SSL_VERSION","3");
```

サーブレットおよびその他の Oracle NCA アプリケーションのテスト

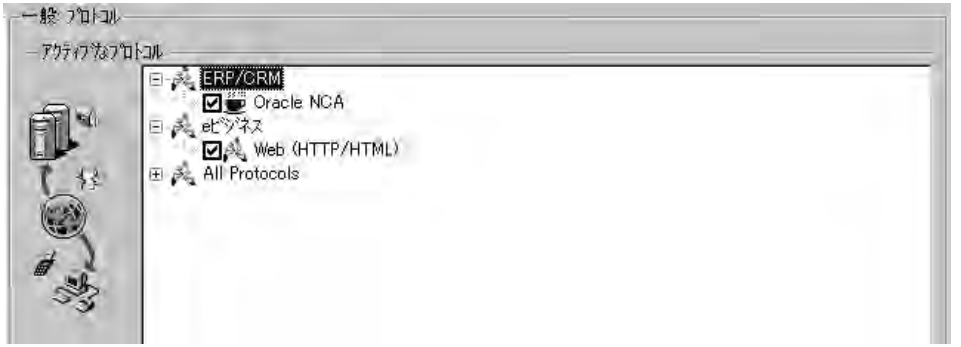
NCA セッションの中には，サーブレットを使用するものがあります。

➤ Forms Listener サーブレット

➤ NCA および HTTP 通信の両方を使用するアプリケーションまたはモジュール（Oracle Configurator など）

➤ NCA アプリケーションの初期化（アプレット，jar ファイル，gif ファイルのダウンロード）

サーブレットを記録するときは，Oracle NCA 関数と Web 関数の両方を記録する必要があります。そのためには，最初にマルチ・プロトコル・スクリプトを作成します。また，Oracle NCA を対象としたシングル・プロトコル・スクリプトを作成した場合は，［記録オプション］で［**一般：プロトコル**］ノードを開き，Web プロトコルを有効にします。これで，記録が開始できます。



アプリケーションでサーブレットが使用されているかどうか不確かな場合は，script ディレクトリの **default.cfg** ファイルを確認します。次のエントリを探します。

UseServletMode=

値が 1 または 2 ならば，サーブレットが使用されています。Oracle NCA に加えて HTTP の記録も有効にする必要があります。

すでにスクリプトが記録されている場合は，Web 関数を含めるようにコードを自動的に再生成できます。再度記録をする必要はありません。［**ツール**］＞［**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 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「ポートの割り当て設定」を参照してください。

Oracle DB の追跡情報の記録

スクリプトをデバッグするには，Oracle DB ブレークダウン・グラフを使用できます。このグラフのデータを収集するには，スクリプトを実行する前に追跡メカニズムを有効にします。

追跡メカニズムを手動で有効にするには，**nca_set_custom_dbtrace** 関数を使用します。詳細については，「**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。

ロード・バランシングに向けた Oracle NCA ステートメントの相関

VuGen は，複数のアプリケーション・サーバを対象とするロード・バランシングをサポートしています。HTTP の戻り値を **nca_connect_server** パラメータと相関させます。以降，仮想ユーザはテスト実行時に，対応するサーバに接続してロード・バランシングを適用します。

ロード・バランシングに向けてステートメントを相関させるには，次の手順を実行します。

1 **マルチ・プロトコル・スクリプトを記録します。**

Oracle NCA および Web プロトコルのマルチ・プロトコル・スクリプトを記録します。必要なアクションを実行し，スクリプトを保存します。

2 **ホストのパラメータと引数を定義します。**

パラメータ化用に 2 つの変数 serverHost および serverArgs を定義します。

```
web_set_max_html_param_len("512");
web_reg_save_param("serverHost", "NOTFOUND=ERROR",
     "LB=<PARAM name=¥"serverHost¥" value=¥"","RB=¥">", LAST);
web_reg_save_param("serverArgs", "NOTFOUND=ERROR",
     "LB=<PARAM name=¥"serverArgs¥" value=¥"","RB=¥">", LAST);
```

3 **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 ");
```

を次のように変更します。

```
nca_connect_server("< serverHost >", "9000"/*version=107*/, "<
serverArgs >");
```

スクリプトは次のようになるはずです。

```
web_set_max_html_param_len("512");
web_reg_save_param("serverHost", "NOTFOUND=ERROR",
      "LB=<PARAM name=¥"serverHost¥" value=¥"","RB=¥">", LAST);
web_reg_save_param("serverArgs", "NOTFOUND=ERROR",
      "LB=<PARAM name=¥"serverArgs¥" value=¥"","RB=¥">", LAST);
      web_url("step_name", "URL=http://server1.merc-int.com/test.htm",
LAST);
nca_connect_server(" < serverHost > ","9000"/*version=107*/," <
serverArgs > ");
```

その他に推奨される相関

Oracle NCA セッションを記録するとき，動的な値，つまり記録セッションおよび再生セッションごとに変化する値が VuGen によって記録されます。よく使用される動的な 2 つの引数が，icx_ticket と JServSessionIdroot です。

icx_ticket

icx_ticket 変数は，**web_url** 関数および **nca_connect_server** 関数で送信する情報の一部です。

```
web_url("fnd_icx_launch.runforms",
"URL=http://ABC-
123:8002/pls/VIS/fnd_icx_launch.runforms¥?ICX_TICKET=5843A55058947ED3&RESP_APP=AR&RESP_KEY=RECEIVABLES_MANAGER&SECGRP_KEY=STANDARD", LAST);
```

この **icx_ticket** の値は記録ごとに異なります。この変数には，クライアントによって送信されたクッキー情報が格納されます。記録を相関させるには，記録された icx_ticket 値の最初の出現の前に **web_reg_save_param** を追加します。

```
web_reg_save_param("icx_ticket", "LB=TICKET=", "RB=&RES", LAST);

...

web_url("fnd_icx_launch.runforms",
"URL=http://ABC-
123:8002/pls/VIS/fnd_icx_launch.runforms¥?ICX_TICKET=<icx_ticket>&
RESP_APP=AR&RESP_KEY=RECEIVABLES_MANAGER&SECGRP_K
EY=STANDARD", LAST);
```

注：web_reg_save_param の左右の境界は，アプリケーションの設定によって異なる場合があります。

JServSessionIdroot

JServSessionIdroot 値は，セッション ID を格納するためにアプリケーションによって設定されるクッキーです。ほとんどの場合，この値は VuGen によって自動的に相関され，**web_reg_save_param** 関数が挿入されます。この関数が自動的に追加されなかった場合は，手作業で追加し，値をすべてパラメータ名で置き換えます。

相関させる必要がある値を特定するには，実行ログを開き（[**表示**] ＞ [**出力ウィンドウ**]），応答の本体を探します。

```
vuser_init.c(8):Set-Cookie: JServSessionIdroot=my1sanw2n1.JS4; path=/¥r¥n
vuser_init.c(8):Content-Length: 79¥r¥n
vuser_init.c(8):Content-Type: text/plain¥r¥n
vuser_init.c(8):    ¥r¥n
vuser_init.c(8):81-byte response body for "http://ABC-
123/servlet/oracle.forms.servlet.ListenerServlet?ifcmd=getinfo&ifhost=mercury&
ifip=123.45.789.12" (RelFrameId=1)
vuser_init.c(8):
/servlet/oracle.forms.servlet.ListenerServlet?JServSessionIdroot=my1san
w2n1.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，587 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 42 章

SAPGUI 仮想ユーザ・スクリプト作成

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（SAP GUI）をテストするためのソリューションについて説明します。mySAP Workplace および Portal クライアントのためのソリューションをテストする方法の詳細については，第 43 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，以下の項目について説明します。

- SAPGUI 仮想ユーザ・スクリプトの作成について
- 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 仮想ユーザ・スクリプトの記録
- SAPGUI 記録オプションの設定
- SAPGUI スクリプトのステップの対話的挿入
- 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 のオプションのウィンドウの再生
- SAPGUI 実行環境の設定
- SAPGUI の関数
- SAP GUI 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

次の情報は SAPGUI プロトコルと SAPGUI/SAP-Web デュアル・プロトコルにのみ適用されます。

SAPGUI 仮想ユーザ・スクリプトの作成について

本章では，SAPGUI for Windows クライアント（SAP GUI）をテストするためのソリューションについて説明します。クライアント上でのみ運用する SAPGUI ユーザをテストするには，SAPGUI 仮想ユーザ・タイプを使用します。Web ブラウザも使用する SAPGUI をテストするには，SAPGUI/SAP-Web デュアル・プロトコルを使用します。

セッションを記録する前に，モジュールとクライアント・インタフェースが VuGen によってサポートされていることを確認します。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で，SAPGUI 仮想ユーザによってエミュレートされます。また，APO モジュールの記録もサポートします（APO 3.0 のパッチ・レベル 24 が必要です）。
- SAPGUI for Windows と Web ブラウザ：SAPGUI/SAP-Web デュアル・プロトコルを使用します。
- SAPGUI for Java：このクライアントはサポートされていません。

バージョン 6.10 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 コントローラまたは Tuning Module コンソール内のスクリプトを 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：SAPGUI または SAPGUI/SAP-Web デュアル・プロトコルを使って VuGen でスクリプトを作成し，コントローラまたはコンソールでシナリオまたはセッション・ステップを実行します。

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。

例えば，**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，次の手順を実行します。

1 SAPGUI ログオン・ウィンドウを起動します。［SAP Logon］ダイアログ・ボックスの左上隅をクリックし，メニューから［**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

Mercury 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。Mercury 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。

VuGen には，システムでスクリプティングがサポートされているかどうかを確認するユーティリティが付属しています。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

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	45B	SAPKH45B49	Kernel 4.5B レベル 753
SAP_BASIS	46B	SAPKB46B37	Kernel 4.6D レベル 948
SAP_BASIS	46C	SAPKB46C29	Kernel 4.6D レベル 948
SAP_BASIS	46D	SAPKB46D17	Kernel 4.6D レベル 948
SAP_BASIS	610	SAPKB61012	Kernel 6.10 レベル 360

カーネル・パッチ・レベルを確認するには，次の手順を実行します。

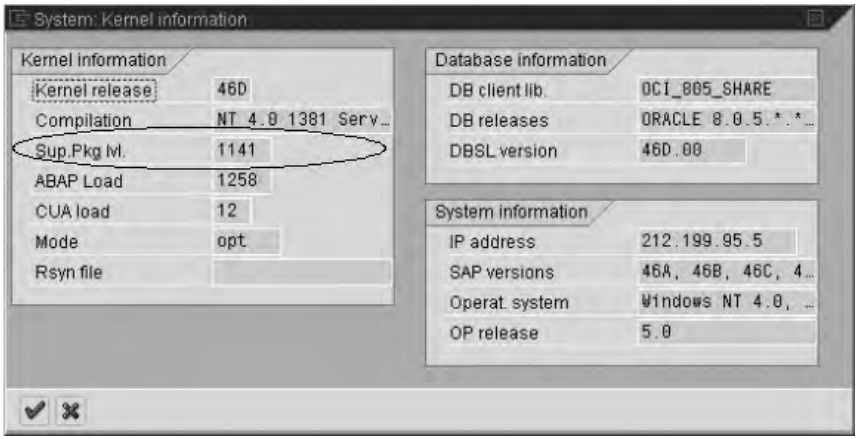
1 SAP システムにログインします。

2［**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。

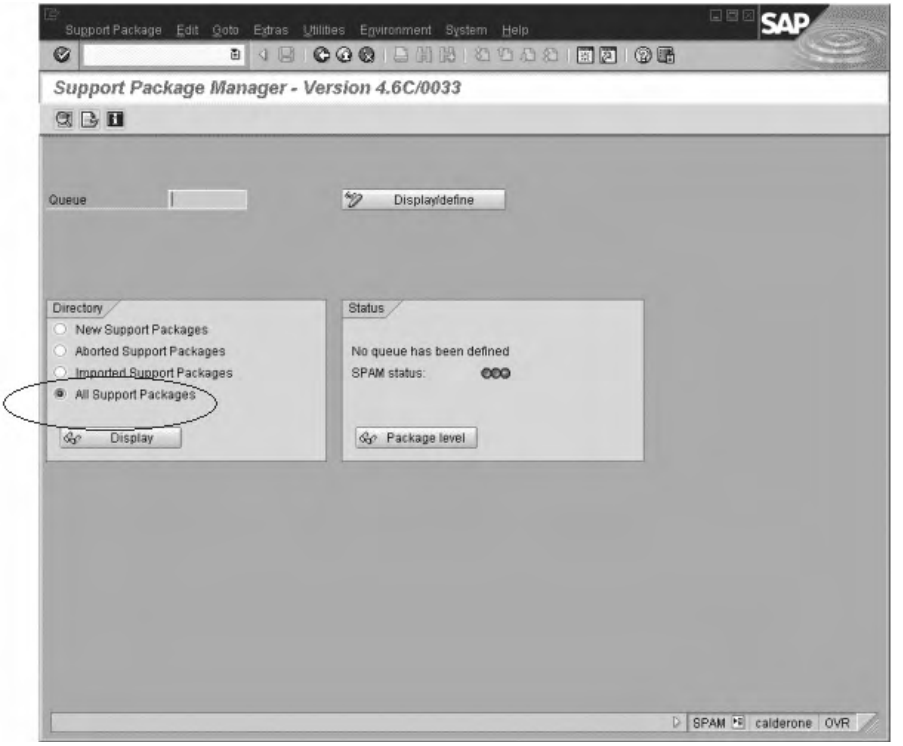
レベルが 948 より低い場合は，最新のバージョンのカーネルをダウンロードして，既存のカーネルをアップグレードする必要があります。このアップグレード方法の詳細については『SAP OSS note #480149』を参照してください。



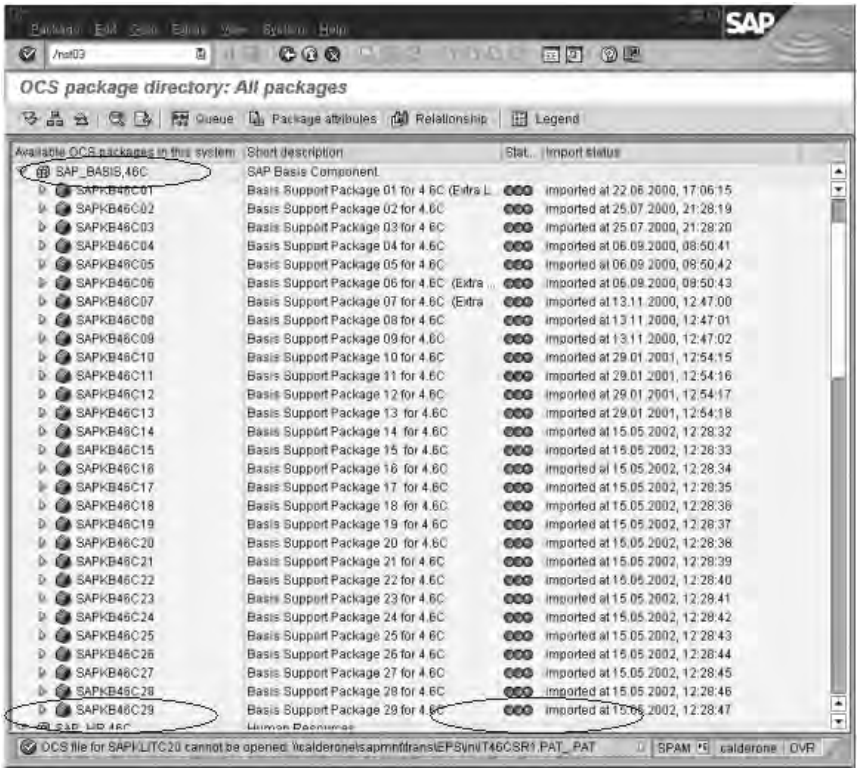
R/3 サポート・パッケージを確認するには，次の手順を実行します。

1 SAP システムにログインして，SPAM トランザクションを実行します。

2［**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。



3 SAP_BASIS, 4.6C に SAPKB46C29 がインストールされていることを確認します。インストールされていれば，［Status］カラムに緑色の丸が表示されます。



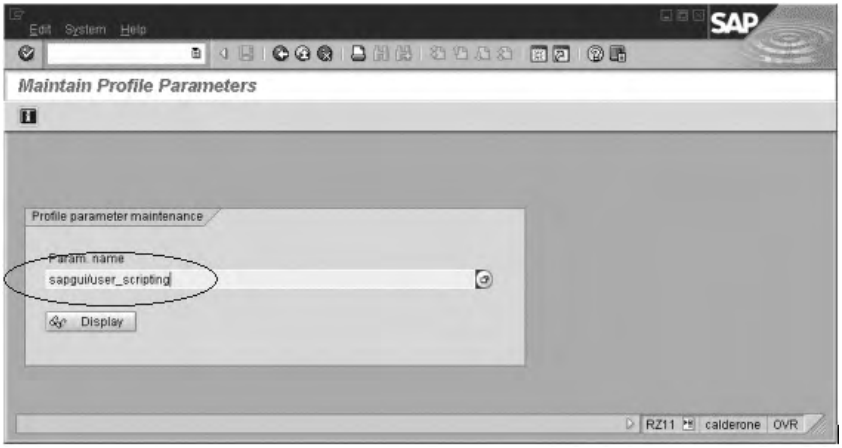
OCS パッケージがインストールされていない場合は，www.sap.com Web サイトからダウンロードしてインストールします。詳細については，『SAP OSS note ＃ 480149』を参照してください。

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，次の手順を実行します。

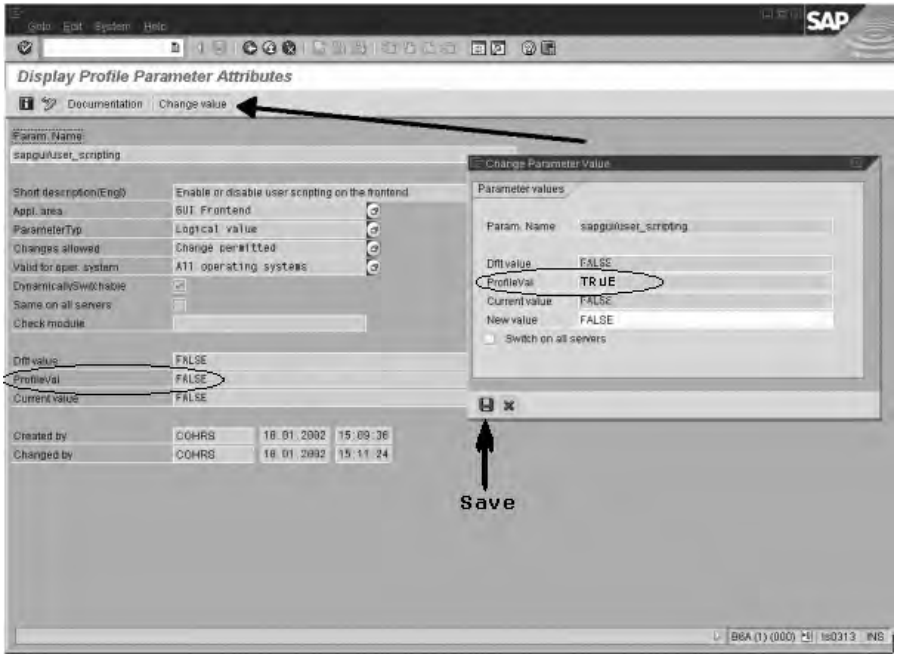
1 トランザクション **rz11** を開きます。パラメータ名 **sapgui/user_scripting** を指定し，［**Display**］ボタンをクリックします。［Display Profile Parameter Attributes］ウィンドウが開きます。



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，603 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** がサーバの再起動後にも変更されていない場合は，アプリケーション・サーバのカーネルが古くなっています。必要なカーネル・パッチをインポートします。詳細については，603 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

リリース	カーネル・バージョン	パッチ・レベル
6.10	6.10	391
6.20	すべてのバージョン	すべてのレベル

SAPGUI 6.20 Client でのスクリプティングの有効化

VuGen でスクリプトを実行できるようにするには，SAPGUI クライアントでもスクリプティングを有効にする必要があります。また，接続が確立されたときやスクリプトが GUI プロセスにアタッチされたときなどに表示される特定のメッセージが表示されないようにクライアントを設定する必要もあります。

VuGen で使用できるように SAPGUI クライアントを設定するには，次の手順を実行します。

➤ **インストール中**：SAPGUI クライアントのインストール中に，［**SAP GUI Scripting**］オプションを有効にします。

➤ **インストール後**：警告メッセージが表示されないようにします。SAPGUI クライアントで［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・スクリプトを作成するには，次の手順を実行します。

1 VuGen を起動し，［**ファイル**］＞［**新規作成**］を選択します。

2 標準的な SAPGUI クライアント・セッション（ブラウザのコントロールなし）を記録するには，**SAPGUI** タイプの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**］をクリックして仮想ユーザ・スクリプトを開きます。

SAPGUI 仮想ユーザ・スクリプトの記録

空のスクリプトの作成後，記録オプションを設定し，SAPGUI セッションを記録します。VuGen はクライアント内のアクションに対応するスクリプトを生成する。

SAPGUI スクリプトを開始するには，次の手順を実行します。

1 ［記録開始］ダイアログ・ボックスが開かない場合は，［**記録開始**］ボタンをクリックします。［記録開始］ダイアログ・ボックスが開きます。

2 VuGen が関連情報を検出して埋めます。

［**記録するアプリケーション**］: VuGen は SAP クライアントのインストール先から **saplogon.exe** ファイルを見つけます。

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

［**アクション内に記録**］: 記録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。

3 ［**OK**］をクリックして記録を開始します。

カーソル位置での記録

VuGen では，既存のスクリプトにアクションを記録することもできます。いくつかの理由から，既存のスクリプトへの記録を選ぶことがあります。

- 記録中に操作を誤った場合。
- 操作は正しくても，ポップアップ・ウィンドウの処理などの追加情報が必要な場合。例えば，SAP サーバは記録セッション中に適用されなかったインベントリ警告を出すことがあります。

これは「**カーソル位置での記録**」と呼ばれる，新しいアクションの挿入または既存のアクションの置換を行えるようにする機能です。カーソルでの記録を開始すると，VuGen は次の 2 つのオプションの入力を求めるプロンプトを表示します。

［**アクションにステップを挿入する**］：既存のステップを一切上書きせずに，新しく記録されたステップを挿入します。新しいセグメントは追加されたセクションの始まりと終わりを表すコメントで囲まれます。このオプションは，記録中には存在しなかった，ときおり現れるポップアップ・ウィンドウの処理に適しています。

```
// Recording at the cursor - 開始
    sapgui_select_active_connection(“con[0]”);
    sapgui_select_active_session(“ses[0]”);
    sapgui_select_active_window(“wnd[0]”);
//Recording at the cursor - 終了
```

［**スクリプトの残りの部分を上書きする**］：カーソル位置から後のすべてのステップを置換します。このオプションは現在のアクションの残りの部分を上書きし，他のすべてのアクションを削除します。**vuser_init** セクションと **vuser_end** セクションでは無効です。このオプションは，記録中の誤操作を訂正するのに適しています。

カーソルでの記録オプションのどちらかを選択すると，VuGen はスクリプトを先頭からカーソルのエントリ・ポイントまで再生します。次に［記録］フローティング・ツールバーを表示して，記録を開始します。

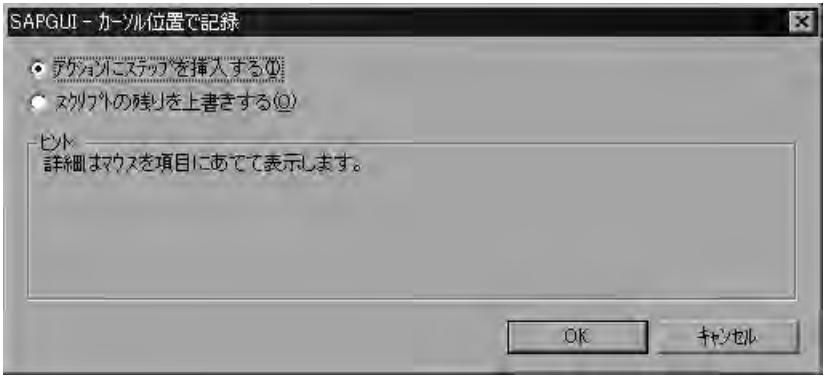
注：カーソルでの記録機能を使う場合には，**スクリプトの再生成**ツールが無効になります。

カーソルで記録するには，次の手順を実行します。

標準設定を選択してから，以降のカーソルでの記録で VuGen がこのダイアログ・ボックスを非表示にするように指定できます。

1［**カーソル位置で記録**］ボタンをクリックします。 

選択を求めるプロンプトが表示されます。



2［**アクションにステップを挿入する**］または［**スクリプトの残りの部分を上書きする**］を選択します。［**OK**］クリックします。VuGen はスクリプトをカーソルの位置まで再生します。

3［記録］フローティング・ツールバーが開くのを待ちます。SAPGUI クライアント内でアクションを開始し，必要に応じてセクションとアクションを切り替えます。

4［**停止**］ボタンをクリックして記録セッションを終了します。

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。

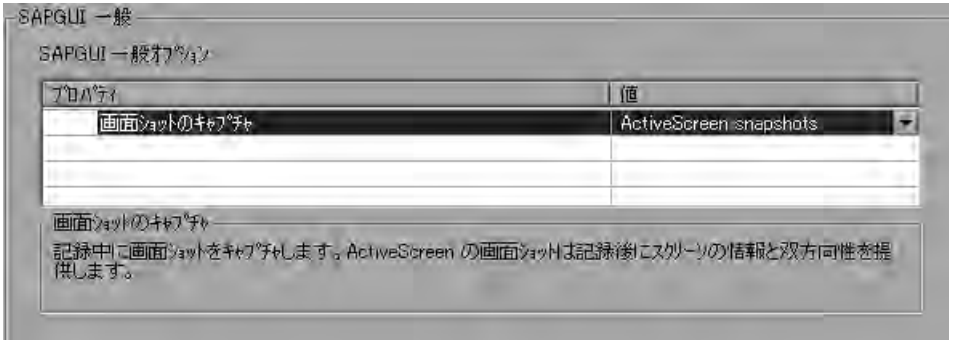
次の項目について記録オプションの設定が可能です。

- SAPGUI：一般記録オプション
- SAPGUI：コード生成記録オプション
- SAPGUI：自動ログオン記録オプション

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 24 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，次の手順を実行します。

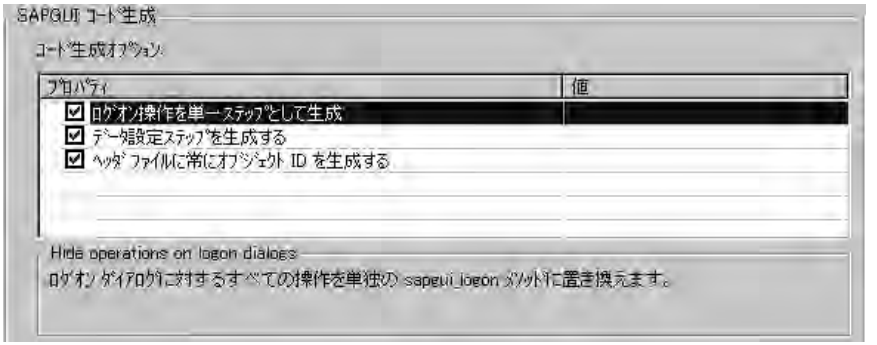
1 ［記録オプション］ダイアログ・ボックスを開き，［**SAPGUI：一般**］ノードを選択します。

2 ［**画面ショットのキャプチャ**］のために，記録中に表示される SAPGUI 画面のスナップショットの保存方法を指定します。リストから項目を選択します。［**ActiveScreen の画面ショット**］，［**一般の画面ショット**］，または［**なし**］があります。

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。

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，次の手順を実行します。

1 ［記録オプション］ダイアログ・ボックスを開き，［**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。

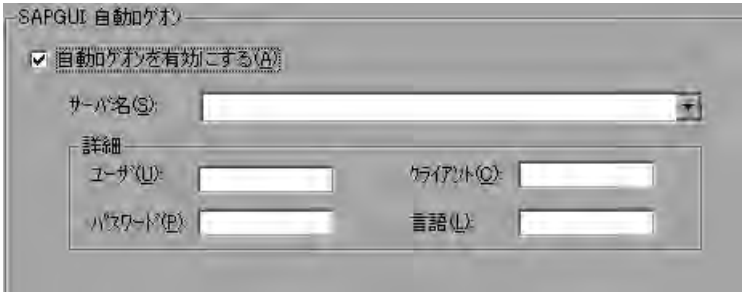
3 各セルに個別のステップを作成するのではなく，テーブルとグリッドを制御するための Fill Data ステップを生成するには，［**データ設定ステップを生成する**］を選択します。

4 よりコンパクトですっきりしたスクリプトを作成するには，オブジェクト ID をスクリプトにではなく個別のヘッダー・ファイルに保存する［**ヘッダ ファイルに常にオブジェクト ID を生成する**］を選択します。このオプションを無効にすると，VuGen は，スクリプトの一般設定で指定された文字列の長さに従って，ID を生成します。このオプションを無効にしても，可読性が増すだけで，オーバーヘッドの差はありません。

5 ［**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：自動ログオン記録オプション

これらの記録オプションを設定して，記録開始時に自動的にログオンするようにします。ログオン関数はスクリプトの vuser_init セクションに置かれます。サーバ名リストには SAP ログオン記述リスト上のすべてのサーバが含まれます。



［自動ログオン］記録オプションを有効にして設定するには，次の手順を実行します。

1 ［記録オプション］ダイアログ・ボックスを開き，［**SAPGUI：コード生成**］ノードを選択します。

2 ［**自動ログオンを有効にする**］を選択します。

3 ログイン情報を入力します。

- SAP **サーバ**の名前。
- SAP サーバの**ユーザ**の名前。
- SAP サーバへログインするための**パスワード**。
- MetaFrame サーバがクライアントの識別に使用する**クライアント名**（任意）。
- インタフェースで使用する**言語**。

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。

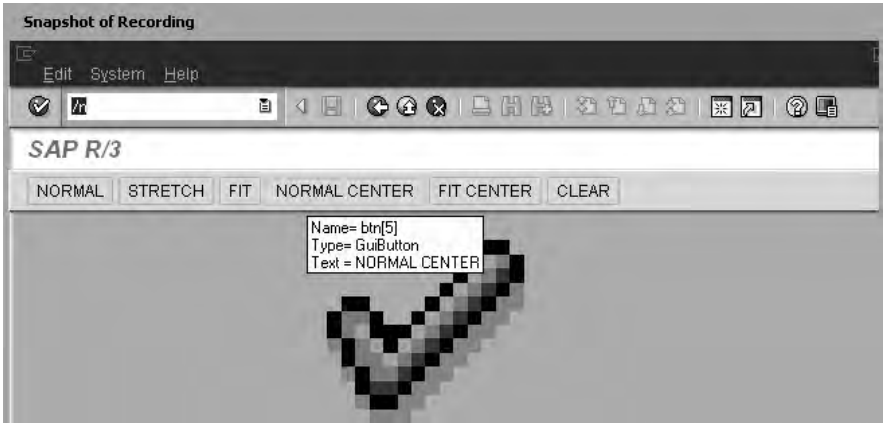
SAPGUI スクリプトのステップの対話的挿入

記録後，スクリプト・ビューかツリー・ビューのどちらかで，手作業でステップをスクリプトに追加できます。各種のスクリプト・ビューの詳細については，『**第 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「VuGen の紹介」を参照してください。

新しい関数を手作業で追加するだけでなく，Citrix 仮想ユーザのために，新しいステップをスナップショットから直接，対話形式で追加できます。右クリック・メニューを使用して，ビットマップ関連またはテキスト関連のステップを追加できます。

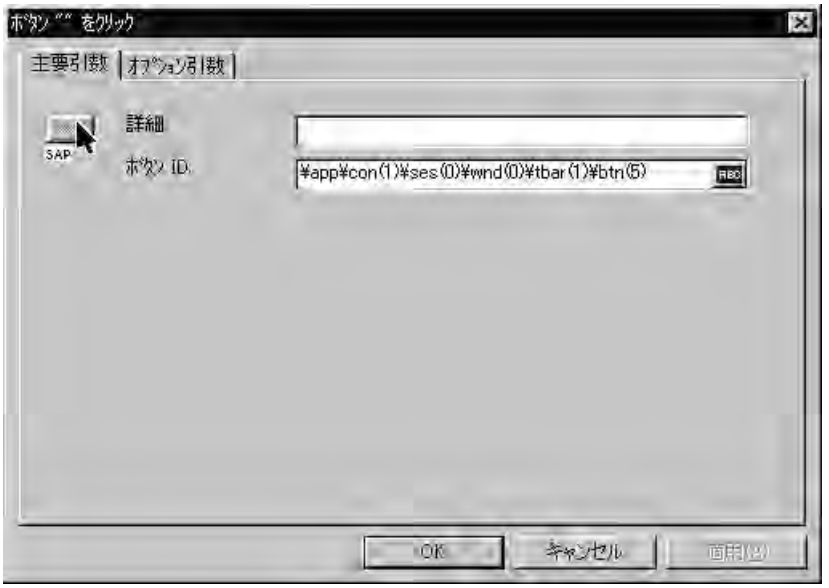
スナップショットの中からステップを追加する場合には，VuGen は Active Screen 機能を使い，SAPGUI クライアント・ウィンドウ内の各オブジェクトの ID を調べます（SAPGUI：一般記録オプション の中で Active Screen スナップショットを無効にしていない場合）。

VuGen によってどのオブジェクトが認識されているかを調べるには，スナップショット上でマウスを動かします。VuGen はオブジェクトの周囲にボックスを描画し，オブジェクトの Control ID を持つツール・チップを表示します。次の例では，選択されたアクティブ・オブジェクトは NORMAL CENTER ボタンです。



認識されたオブジェクト上にマウスがある間にステップを追加すると，VuGen は自動的にそのオブジェクトの Control ID を［Properties］ダイアログ・ボックスの関連フィールドに挿入します。例えば，上に示すように，NORMAL CENTER

ボタンのために **Press Button** ステップを挿入した場合，［プロパティ］ダイアログ・ボックスには次の ID が表示されます。



特定のオブジェクトにステップを対話的に挿入するには，次の手順を実行します。

1 ［バッファのスナップショット］ウィンドウ内をクリックします。

2 関数を追加するオブジェクト上にマウスを移動します。VuGen がそのオブジェクトを認識し，ボックスで囲んでいることを確認します。

3 右クリックで表示されるメニューから［**一時停止**］を選択します。［ステップの追加］ボックスが開きます。

4 メニューからステップを選択します。ステップの「プロパティ」ダイアログ・ボックスが開き，関連するオブジェクトの Control ID が表示されます。

5 ［**詳細**］ボックスにオブジェクトの名前を入力します。［**OK**］クリックします。選択したステップの後に新しいステップが挿入されます。

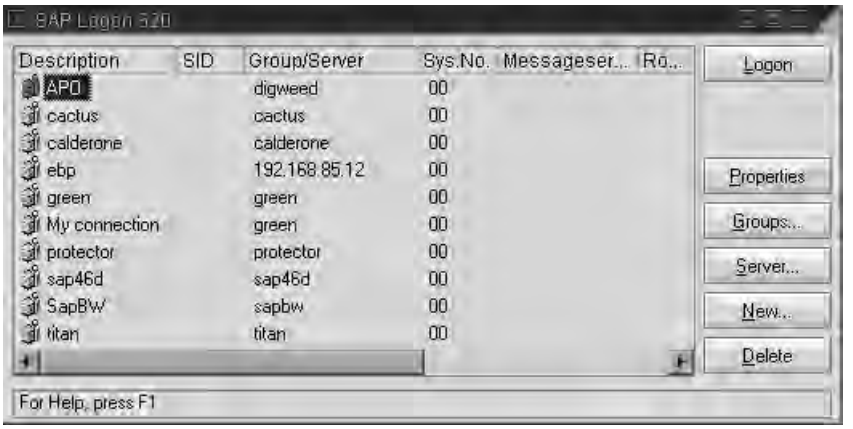
6 特定の場所に貼り付けるオブジェクトの Control ID を取得するには，右クリック・メニューから［**Copy Control ID**］を選択します。Control ID がクリップボードに置かれます。［スクリプト］ビューから［プロパティ］ボックスまたは直接コードに貼り付けることができます。

SAPGUI 仮想ユーザ・スクリプトについて

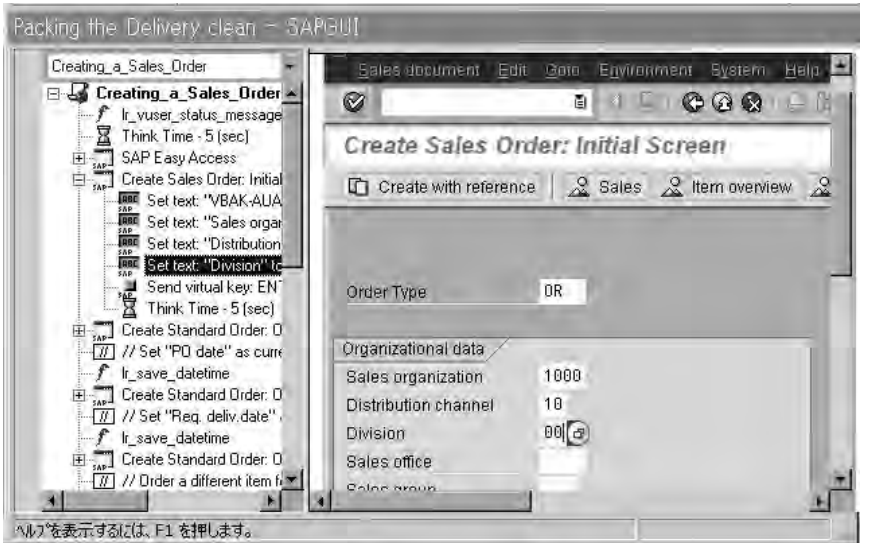
通常 SAPGUI 仮想ユーザ・スクリプトには，ビジネス・プロセスを構成する複数の SAP トランザクションが含まれています。ビジネス・プロセスは，ユーザ・アクションをエミュレートする関数で構成されます。ツリー・ビューを開くと，各ユーザ・アクションが仮想ユーザ・スクリプトのステップとして表示されます。

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, REPTI-
TLE",
      "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 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「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，627 ページ「情報の取得」を参照してください。
- **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・プロセスを何度も繰り返すことができます。詳細については，『**第 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「VuGen パラメータを使った作業」を参照してください。

検証関数の追加

オプションの，または動的なウィンドウやフレームで作業をしているときに，検証関数を使用して，ウィンドウやオブジェクトが使用可能かどうかを調べることができます。オプションのウィンドウは，SAP セッション中に常に表示されるとは限らないウィンドウです。この関数により，オプションのウィンドウが開いたり例外が発生したりした場合でも，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・バーからの注文番号を取得するには，次の手順を実行します。

1 ステータス・バー・テキストを確認する位置に移動して，［挿入］＞［新規ステップ］を選択します。**sapgui_status_bar_get_type** 関数を選択します。この関数は，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。例えば，スクリプトを 6 月 2 日に記録し，6 月 3 日に再生した場合は，日付フィールドが正しくなくなります。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。例えば，＜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 のオプションのウィンドウの再生

SAPGUI 仮想ユーザ・スクリプトの処理中に，SAPGUI クライアントにオプションのウィンドウ（記録時には表示されるのに，再生時には表示されないウィンドウ）が表示されることがあります。記録したスクリプトをそのまま再生しようとしても，存在しないウィンドウを見つけようとして失敗することになります。

VuGen のオプションのウィンドウのメカニズムは，そのウィンドウの存在の確認後にのみ，アクションを実行します。仮想ユーザは，［**アクティブ ウィンドウを選択**］ステップに示されたウィンドウが存在するかどうかを検査します。ウィンドウが再生時に見つかれば，スクリプトに記録されているとおりにアクションを実行します。存在しない場合には，仮想ユーザは次の［**アクティブ ウィンドウを選択**］ステップまでのすべてのウィンドウのアクションを無視します。SAPGUI ステップ（**sapgui** 接頭辞で始まるステップ）のみが無視されます。

この機能を使用するには，ツリー・ビューで適切な［アクティブ ウィンドウ］ステップを選択し，右クリック・メニューから［**ウィンドウのみでステップを実行する**］を選びます。

この機能を無効にし，これらのステップが常に実行されるようにするには，仮想ユーザがウィンドウを見つけるかどうかにかかわらず，右クリック・メニューから［**このウィンドウでは常にステップを実行する**］を選択します。

SAPGUI 実行環境の設定

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。仮想ユーザ・スクリプトを実行する前に，これらの設定を行います。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 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「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 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「スタンドアロン・モードでの仮想ユーザ・スクリプトの実行」を参照してください。

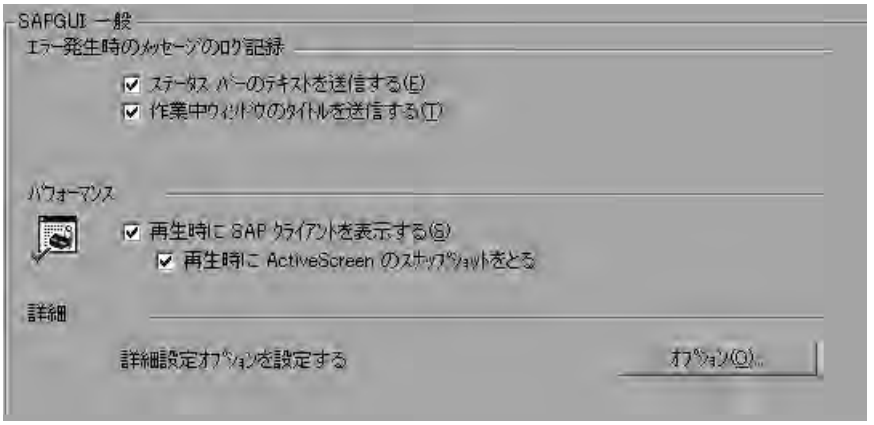
スクリプトを検証したら，スクリプトを環境（LoadRunner シナリオ，Performance Center 負荷テスト，Tuning Module セッション，またはビジネス・プロセス・モニタ・プロファイル）に統合します。詳細については，『**Mercury LoadRunner コントローラ・ユーザーズ・ガイド**』，または **Tuning Console**，**Performance Center**，**Application Management** のマニュアルを参照してください。

次の領域で，SAP GUI 固有の実行環境設定を行えます。

- 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 のテキストを送信する**］：ステータス・バーからログ・ファイル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。

［**再生時に ActiveScreen のスナップショットをとる**］：Control ID 情報を持つ再生スナップショットをすべてのアクティブ・オブジェクトでキャプチャします。ActiveScreen スナップショットは，通常のスナップショットと異なり，SAPGUI クライアントの中でどのオブジェクトが VuGen によって認識されているかを知ることができます。スナップショット上でマウスを動かすと，VuGen は検出したオブジェクトを強調表示します。スナップショット内からスクリプ

トに直接新しいステップを追加できます。また，特定のオブジェクトに対して，スナップショット内から対話的にステップを追加することもできます。詳細については，620 ページ「SAPGUI スクリプトのステップの対話的挿入」を参照してください。

詳細設定のためのオプションを使って，［**SAPfewgsvr.exe**］プロセスのタイムアウトの設定，エラー発生時のスナップショットの保存，および再生中に VuGen が SAPlogon を使用する設定を行うことができます。詳細については，632 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，次の手順を実行します。

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

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，メッセージ・ソース［**ステータス バーのテキストを送信する**］と［**作業中ウィンドウのタイトルを送信する**］から少なくとも 1 つを選択します。

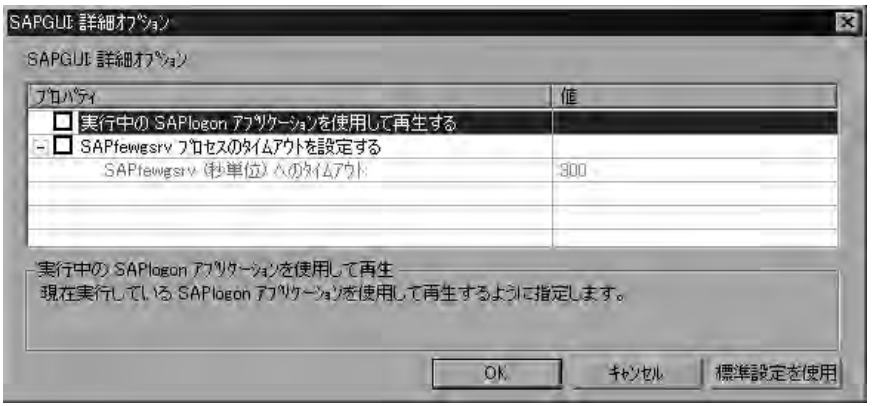
4 再生時に SAP ユーザ・インタフェースを表示するには，［**パフォーマンス**］セクションで［**再生時に SAP クライアントを表示する**］チェック・ボックスを選択します。

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，再生セッションの終了後もプロセスが終了しないことがあります。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。



［**実行中の SAPlogon アプリケーションを使用して再生する**］：仮想ユーザが再生のために現在実行中の SAPlogon アプリケーションを使うように指定します。

［**SAPfewgsvr プロセスのタイムアウトを設定する**］：［**SAPfewgsvr.exe**］プロセスのタイムアウトを変更できます。

［**SAPfewgsvr へのタイムアウト**］：［**SAPfewgsvr.exe**］プロセスの秒単位のタイムアウトです。標準設定は 300 秒です。

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。

次のようなカテゴリの **SAPGUI** 関数があります。接続関数とセッション関数，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。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，APO グリッド関数，ステータス・バー関数，テーブル関数，ツリー関数，ウィンドウ関数，および一般オブジェクト関数があります。

接続関数とセッション関数

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。

APO グリッド関数

sapgui_apogrid_clear_selection	すべての選択されたセルを選択解除します。
sapgui_apogrid_deselect_cell	指定したセルを選択解除します。
sapgui_apogrid_deselect_column	指定したカラムを選択解除します。
sapgui_apogrid_deselect_row	指定した行を選択解除します。
sapgui_apogrid_double_click	APO グリッドの内側をダブルクリックします。
sapgui_apogrid_get_cell_data	指定した APO グリッド・セルからデータを取得します。
sapgui_apogrid_get_cell_format	指定した APO グリッド・セルから形式を取得します。
sapgui_apogrid_get_cell_tooltip	指定した APO グリッド・セルのツールチップを取得します。
sapgui_apogrid_is_cell_changeable	指定したセルが編集可能かどうか検査します。
sapgui_apogrid_open_cell_context_menu	指定されたセルのコンテキスト・メニューを開きます。
sapgui_apogrid_press_ENTER	APO グリッドの中で ENTER キーを押します。
sapgui_apogrid_scroll_to_column	APO グリッド内の指定したカラムにスクロールします。

sapgui_apogrid_scroll_to_row	APO グリッド内の指定した行にスクロールします。
sapgui_apogrid_select_all	APO グリッド内のすべてのセルを選択します。
sapgui_apogrid_select_cell	APO グリッドでセルを選択します。
sapgui_apogrid_select_column	APO グリッドでカラムを選択します。
sapgui_apogrid_select_row	APO グリッドで行を選択します。
sapgui_apogrid_set_cell_data	指定した APO グリッド・セルにデータを設定します。

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・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・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。

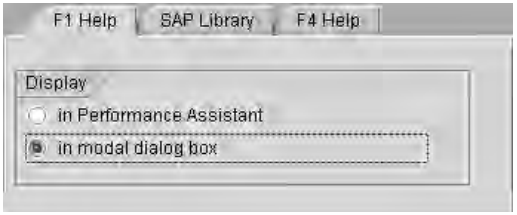
SAP GUI 仮想ユーザ・スクリプトに関するヒント

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，およびシナリオまたはセッション・ステップ内での再生に関するヒントについて説明します。また，SAP のサポート・サイトからも直接情報を参照できます。

記録に関するヒント

本項では，SAPGUI 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 ヘルプをモーダル・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。

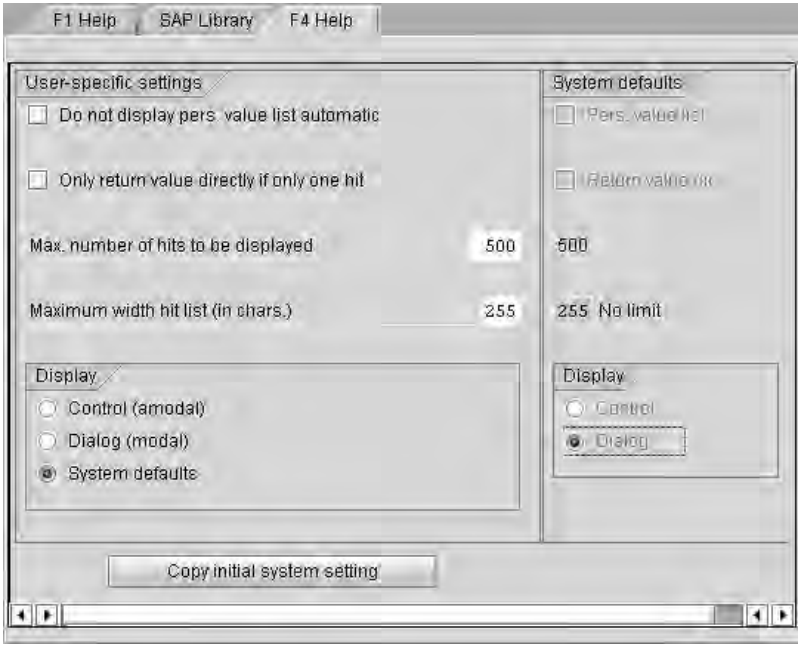


- F4 に対応したモーダル・ダイアログ・ボックスを使用します。F4 ヘルプをモーダル・ダイアログ・ボックスで開くように SAPGUI に指定します。SAP 管理者は次の手順を実行する行う必要があります。

F4 ヘルプをモーダル・ダイアログ・ボックスで開くには，次の手順を実行します。

1 サーバからすべてのユーザがログオフしていることを確認してください。

2［**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に，コード内のパスワードを暗号化できます。パスワード・テキスト（***** ではなく実際のテキスト）を選び，右クリック・メニューで［**文字列を暗号化**］を選択します。**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。

シナリオまたはセッション・ステップ内での再生に関するヒント

以下の各項では，コントローラまたはコンソール，あるいはロード・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。

コントローラおよびコンソールの設定

LoadRunner シナリオまたはセッション・ステップを使って作業する場合は，スクリプトを実行するときに負荷テストの設定で次の値を設定します。

ランプアップ：（適切なログオンを保証するために）1 つずつスケジューラで設定します。

思考遅延時間：実行環境に乱数の思考遅延時間を設定します。

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またはセッション・ステップで実行するときは，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。

［**Agent Mode**］：［Process］モードで LoadRunner（または Performance Center）Remote Agent が実行されていることを確認します。サービス・モードはサポートされていません。

これを調べるには，Windows のタスクバーにあるエージェントのアイコン上にマウスを移動し，説明を読みます。「LoadRunner Agent Service」と説明に表示された場合は，サービスとしてエージェントが実行されています。

プロセスとしてエージェントを再起動するには，次の手順を実行します。

1 エージェントを停止します。LoadRunner Agent のアイコンを右クリックし，［**Close**］を選択します。

2 **magentproc.exe** を実行します。これは，LoadRunner または Tuning Module 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**］＞［**Mercury ＜製品名＞**］＞［**Advanced Settings**］から［**Agent Configuration**］を選択し，［端末サービスを有効にする］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，『**Mercury LoadRunner コントローラ・ユーザーズ・ガイド**』の「端末サービスの設定」を参照してください。

注：ターミナル・セッションで LoadRunner Agent を実行している場合，およびターミナル・セッションのウィンドウが最小化されている場合には，エラー時のスナップショットはキャプチャされません。

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：LoadRunner で［Process］モードで LoadRunner Remote Agent が実行されていることを確認します。サービス・モードはサポートされていません。詳細については，645 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。603 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。例えば，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，次の手順を実行します。

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。例えば，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，626 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の詳細については記録ログを参照してください。

質問 7 : すべてのビジネス・プロセスがサポートされていますか ?

回答 : VuGen では，Microsoft Office のモジュール・コントロールを起動するビジネス・プロセス，あるいは GuiXT の使用を必要とするビジネス・プロセスはサポートされていません。**GuiXT** は SAPGUI for Windows クライアントの［Options］メニューから無効にできます。

その他の参考資料

LoadRunner および Tuning Module

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，SAP の Web サイト（www.sap.com）または，次のいずれかの場所を参照してください。

➤ **SAP に関する注意事項** - https://websmp103.sap-ag.de/notes

Note #480149:New profile parameter for user scripting on the front end（フロント・エンドのユーザ・スクリプティングのための新しいプロファイル・パラメータ）

Note #587202:Drag & Drop is a known limitation of the SAPGUI interface（ドラッグアンドドロップは，SAPGUI インタフェースの既知の制限）

➤ **SAP のパッチ** - https://websmp104.sap-ag.de/patches

SAP GUI for Windows - SAPGUI 6.20 パッチ（パッチ・レベル 32 以上）

第 43 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，以下の項目について説明します。

- SAP-Web 仮想ユーザ・スクリプトの作成について
- 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，これらの関数は 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。例えば，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。また，シングル・プロトコルの SAPGUI/SAP-Web デュアル仮想ユーザ・タイプを使用できます。

SAP-Web 仮想ユーザを作成するには，次の手順を実行します。

1 ［**ファイル**］＞［**新規作成**］を選択します。

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，次のどちらかを選択します。

➤ シングル・プロトコル仮想ユーザ・スクリプト（**SAPGUI/SAP-Web デュアル・プロトコル**.

➤ マルチ・プロトコル仮想ユーザ・スクリプト（**SAP-Web** および **SAPGUI 仮想ユーザ・タイプ**

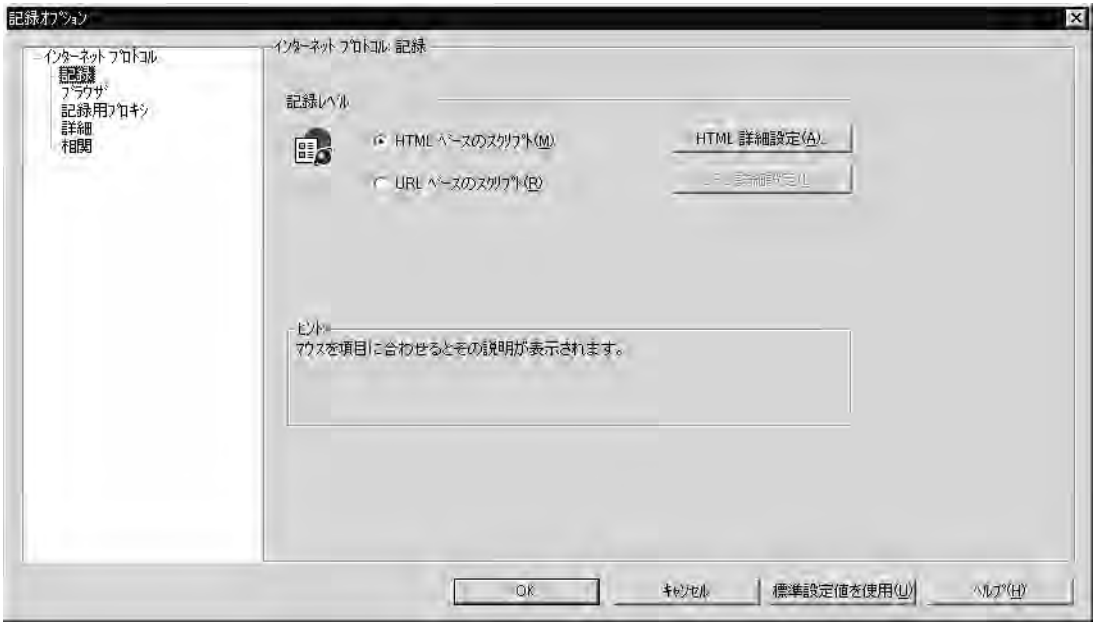
SAP-Web 記録オプションの設定

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 24 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 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「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 26 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 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「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，クリプトを環境（LoadRunner シナリオ，Performance Center 負荷テスト，Tuning Module セッション，またはビジネス・プロセス・モニタ・プロファイル）に統合します。詳細については，『**Mercury LoadRunner コントローラ・ユーザーズ・ガイド**』，または **Tuning Console**，**Performance Center**，**Application Management** のマニュアルを参照してください。

第 44 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 仮想ユーザ・スクリプトの作成について
- Siebel-Web セッションの記録
- Siebel-Web スクリプトの相関
- SWECount，ROWID，および SWET パラメータの相関
- 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，次のガイドラインに従います。

Siebel-Web 仮想ユーザ・スクリプトを記録するには，次の手順を実行します。

1 Siebel-Web タイプの仮想ユーザ・スクリプトを ERP カテゴリから作成します。

2 次の記録オプションを設定します。

➤［記録］ノード：［**HTML ベースのスクリプト**］

［HTML 詳細設定］－［スクリプト タイプ］オプション：［ **明示的な URL のみを含むスクリプト**］

［HTML 詳細設定］－［生成された HTML 以外の要素］：［**記録しない**］

➤［詳細］ノード：［**各アクションごとにコンテキストをリセットする**］オプションをクリア

3 **vuser_init** セクションにログイン手続きを記録します。

4 ビジネス・プロセスを **Action1** に記録します。

5 **vuser_end** セクションにログアウト手続きを記録します。

6 実行環境の設定で，［ブラウザのエミュレーション］ノードの［**反復ごとに新規ユーザをシミュレートする**］オプションをクリアします。

記録オプションと Web 関連の実行環境の設定方法の詳細については，第 24 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 26 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 セッションのテスト・スクリプトを作成する場合は，スクリプトに相関を使用する必要が生じる可能性が大いにあります。相関は，VuGen が記録時および再生時に動的な値をパラメータに保存して，スクリプトの以降の場所で使用できるようにするためのメカニズムです。記録したスクリプトを相関なしでそのまま再生すると，スクリプトを実行するたびに引数の値が変化するため，スクリプトの再生に失敗します。このような変数の例として，**SWECount** や **SWEBMC** があります。

相関を使用すると，VuGen は記録時と再生時の両方で動的な変数を保存し，スクリプト内の適切な箇所でこの変数を使用します。記録時に相関を適用するように VuGen を設定するには，次のいずれかの方法を使用します。

➤ Siebel 相関ライブラリ

Siebel 相関ライブラリを使用すると，動的な値の大部分が自動的に相関され，大幅な変更を加えなくても再生できるスクリプトが作成されます。この相関方法を使用することをお勧めします。

➤ VuGen のネイティブ Siebel 相関

ネイティブ組み込みルールを低レベルで適用すると，スクリプトのデバッグや相関のより詳しい理解が可能になります。

Siebel 相関ライブラリ

相関を簡単に使用できるように，Siebel Application Server バージョン 7.7 の一部として相関ライブラリ・ファイルがリリースされています。このライブラリは，Siebel でのみ使用できます。ライブラリ・ファイル **ssdtcorr.dll** は，Windows では siebsrvr\bin フォルダの下に，UNIX では siebsrvr/lib の下にあります。

ライブラリ・ファイル **ssdtcorr.dll** は，ロード・ジェネレータ，コントローラ，またはコンソールが存在するすべてのマシンで使用できるようにする必要があります。このライブラリのサポートには，VuGen 8.0 以降が必要です。

このライブラリを使って相関を有効にするには，次の手順を実行します。

1 ライブラリの DLL ファイルを Mercury 製品のインストール先の bin ディレクトリにコピーします。

2 **Siebel-Web** 仮想ユーザ・タイプを使用して，マルチ・プロトコル・スクリプトを開きます。

3 記録オプションで UTF-8 のサポートを有効にします。詳細については，303 ページ「記録オプションの詳細設定」を参照してください。

4 記録オプションの［相関］ノードを開き，［**インポート**］をクリックします。\dat\webrulesdefaultsetting ディレクトリからルール・ファイル **WebSiebel77Correlation.cor** をインポートします。警告が表示された場合は，［**Override**］をクリックします。詳細については，407 ページ「相関記録オプションの設定」を参照してください。

標準設定の相関に戻すには，Siebel のルールをすべて削除し，［**標準設定値を使用**］をクリックします。

Siebel 相関ライブラリを使用するときは，SWE カウント・ルール（左の境界に **SWEC** という文字列が含まれているルール）が無効になっていないことを確認します。詳細については，666 ページ「ルールの有効化と無効化」を参照してください。

VuGen のネイティブ Siebel 相関

VuGen の Siebel サーバ用ネイティブ組み込みルールは，Siebel サーバの変数と文字列を検出し，それらをスクリプトの以降の場所で使用できるように保存します。

これらのルールは，相関ルールの一覧で参照できます（399 ページ「VuGen の相関ルールの使用」を参照）。これらのルールは，Siebel サーバの文字列に固有の境界条件を示します。

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

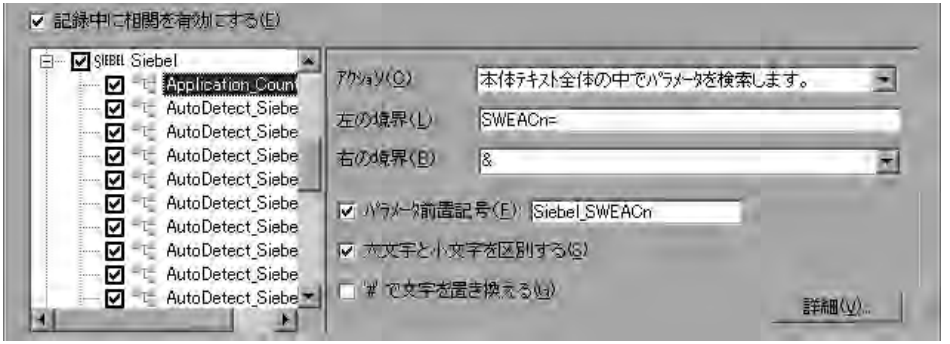
相関記録オプションに独自の境界条件を入力するか（第 30 章「Web 仮想ユーザ・スクリプトの相関ルールの設定」を参照），または記録後に［相関結果］タブを使用して（418 ページ「相関の検索の実行」を参照），独自のルールを作成することもできます。

［再生ログ］タブには，VuGen がパラメータを登録，保存，および使用したタイミングが示されます。この情報を表示するには，拡張ログを有効にする必要

があります。詳細については，189 ページ「実行環境設定のログの設定」を参照してください。

単純な変数の相関

次の例では，左の境界条件が _sn= になっています。左の境界に _sn= が，右の境界に ; が現れるたびに，**Siebel_sn_cookie** という接頭辞の付いたパラメータが作成されます。



次の例では，**_sn** 境界が検出されています。パラメータが **Siebel_sn_cookie6** に保存され，**web_url** 関数で使用されています。

```
/* ソースからパラメータを登録します
web_reg_save_param("Siebel_sn_cookie6",
"LB/IC=_sn=",
"RB/IC=;",
"Ord=1",
"Search=headers",
"RelFrameId=1",
LAST);

...

web_url("start.swe_3",
"URL=http://cannon.mercury.co.il/callcenter_enu/start.swe?SWECmd=GotoPostedAction&SWEDIC=true&_sn={Siebel_sn_cookie6}&SWEC={Siebel_SWECount}&SWEFrame=top._sweclient&SWECS=true",
"TargetFrame=",
"Resource=0",
"RecContentType=text/html",
"Referer=http://cannon.mercury.co.il/callcenter_enu/start.swe?SWECmd=GetCachedFrame&_sn={Siebel_sn_cookie6}&SWEC={Siebel_SWECount}&SWEFrame=top._swe",
"Snapshot=t4.inf",
"Mode=HTML",
LAST);
```

関数の相関

特定のインスタンスでは，境界の一致が関数となります。関数は通常，実行時の値を格納する配列を使用します。これらの値を相関するために，VuGen はその配列を解析し，個々の引数を次の形式で個別のパラメータに保存します。

＜パラメータ名＞ = ＜記録された値＞ (表示名)

表示名は，Siebel アプリケーションで値の隣に表示されるテキストです。

VuGen は，すべてのパラメータ定義を含むコメント・ブロックを挿入します。

```
/* ソース・タスク ID 159 からのパラメータの登録
      // {Siebel_Star_Array_Op33_7} = ""
      // {Siebel_Star_Array_Op33_6} = "1-231"
      // {Siebel_Star_Array_Op33_2} = ""
      // {Siebel_Star_Array_Op33_8} = "Opportunity"
      // {Siebel_Star_Array_Op33_5} = "06/26/2003 19:55:23"
      // {Siebel_Star_Array_Op33_4} = "06/26/2003 19:55:23"
      // {Siebel_Star_Array_Op33_3} = ""
      // {Siebel_Star_Array_Op33_1} = "test camp"
      // {Siebel_Star_Array_Op33_9} = ""
      // {Siebel_Star_Array_Op33_rowid} = "1-6F"
      // */
```

また，関数が見つかると，VuGen は **web_reg_save_param** に新規パラメータ AutoCorrelationFunction を生成します。また，パラメータの接頭辞を特定し，それをパラメータ名として使用します。次の例では，接頭辞は **Siebel_Star_Array_Op33** です。

```
web_reg_save_param("Siebel_Star_Array_Op33",
      "LB/IC=`v`",
      "RB/IC=`",
      "Ord=1",
      "Search=Body",
      "RelFrameId=1",
      "AutoCorrelationFunction=flCorrelationCallbackParseStarArray",
      LAST);
```

VuGen は，パラメータをスクリプトの以降の場所で使用します。次の例では，**web_submit_data** でパラメータが呼び出されています。

```
web_submit_data("start.swe_14",
  "Action=http://cannon.mercury.co.il/callcenter_enu/start.swe",
  "Method=POST",
  "RecContentType=text/html",
  "Referer=",
  "Snapshot=t15.inf",
  "Mode=HTML",
  ITEMDATA,
  "Name=SWECLK", "Value=1", ENDITEM,
  "Name=SWEField", "Value=s_2_1_13_0", ENDITEM,
  "Name=SWER", "Value=0", ENDITEM,
  "Name=SWESP", "Value=false", ENDITEM,
  "Name=s_2_2_29_0", "Value={Siebel_Star_Array_Op33_1}",
ENDITEM,
  "Name=s_2_2_30_0", "Value={Siebel_Star_Array_Op33_2}",
ENDITEM,
  "Name=s_2_2_36_0", "Value={Siebel_Star_Array_Op33_3}",
ENDITEM,
...
```

VuGen は，パラメータとして保存された配列要素を使用して，再生中に public 関数にコールバックを行います。

注：SWEC パラメータの相関は，相関ルールを通じて行われません。組み込み検出方式により自動的に行われます。詳細については，667 ページ「SWEC 相関」を参照してください。

ルールの有効化と無効化

通常の状況では，ルールを無効にする必要はありません。しかし，場合によっては，適用しないルールを無効にすることもできます。例えば，英語専用のアプリケーションをテストするときは，日本語コンテンツのチェック・ルールを無効にします。

ルールを無効にするもう 1 つの理由は，コントローラやコンソールで特定のエラー条件を生成する必要がある場合です。記録オプションでルールのプロパティを表示して，アプリケーションに必要な条件を特定します。

ルールを無効にするには，次の手順を実行します。

1 相関記録オプションを開きます。［**ツール**］＞［**記録オプション**］を選択し，［**相関**］ノードをクリックします。

2［**記録中に相関を有効にする**］オプションを選択します。サポートされているサーバがダイアログ・ボックスに表示されます。

3［Siebel］の下のルールを展開し，プロパティを表示します。

4 無効にする各ルールの横にあるチェック・ボックスをクリアします。

SWEC 相関

SWEC は，Siebel サーバによって使用されるパラメータで，ユーザのクリック数を表します。SWEC パラメータは通常，URL ステートメントまたは POST ステートメントの引数として現れます。次に例を示します。

```
GET /callcenter_enu/start.swe?SWECmd=GetCachedFrame&_sn=2-
mOTFXHWBAAGb5Xzv9Ls2Z45QvxGQnOnPVtX6vnfUU_&SWEC=1&S
WEFrame=top._swe._sweapp HTTP/1.1
```

あるいは

```
POST /callcenter_enu/start.swe HTTP/1.1
...
\r\n\r\n
SWERPC=1&SWEC=0&_sn=2-
mOTFXHWBAAGb5Xzv9Ls2Z45QvxGQnOnPVtX6vnfUU_&SWECmd=In
vokeMethod...
```

VuGen は，関連する各ステップの前にカウンタの数を増やして SWEC の変更を処理します。VuGen は SWEC の現在の値を別の変数（Siebel_SWECount_var）に保存します。各ステップの前に，VuGen はカウンタの値を VuGen パラメータ（Siebel_SWECount）に保存します。

次の例では，**web_submit_data** は SWEC パラメータ Siebel_SWECount. の動的な値を使用しています。

```
Siebel_SWECount_var += 1;

lr_save_int(Siebel_SWECount_var, "Siebel_SWECount");

web_submit_data("start.swe_8",
      "Action=http://cannon.mercury.co.il/callcenter_enu/start.swe",
      "Method=POST",
      "TargetFrame=",
      "RecContentType=text/html",
      "Referer=",
      "Snapshot=t9.inf",
      "Mode=HTML",
      "EncodeAtSign=YES",
      ITEMDATA,
      "Name=SWERPC", "Value=1", ENDITEM,
      "Name=SWEC", "Value={Siebel_SWECount}", ENDITEM,
      "Name=SWECmd", "Value=InvokeMethod", ENDITEM,
      "Name=SWEService", "Value=SWE Command Manager", ENDITEM,
      "Name=SWEMethod", "Value=BatchCanInvoke", ENDITEM,
      "Name=SWEIPS",...
      LAST);
```

SWEC パラメータは参照 URL にも使用されます。ただし，参照 URL の値は要求される URL の値とは異なります。VuGen はこれを自動的に処理します。

SWECount，ROWID，および SWET パラメータの相関

本項では，次のような特殊なパラメータの相関についてヒントを示します。

- SWECount
- 行 ID 長
- SWETS（タイム・スタンプ）

SWECount

SWECount パラメータの値は，通常 1 桁または 2 桁の数字で構成される小さい数値です。記録されたどの値をパラメータで置換するべきか決めるのが困難なことがしばしばあります。

web_submit_data 関数では，VuGen は SWEC フィールドの数値だけを置換します。

URL では，文字列「SWEC=」または「SWEC`」の後に現れる場合にかぎり，この値を置換します。

SWECount 相関のパラメータ名はすべて同じです。

行 ID 長

場合によっては，**rowid** の前に，その長さが 16 進形式でエンコードされて置かれます。この長さは変更される可能性があるため，その値を相関させる必要があります。

例えば，xxx6_1-4ABCyyy という文字列は長さの値と RowID で構成されています。ここで 6 は長さ，1-4ABC は RowID です。

文字列を相関させるパラメータを，詳細相関を使用して

```
xxx{rowid_Length}_{rowid}yyy
```

と定義した場合，VuGen は文字列の前に次の関数を生成します。

```
web_save_param_length("rowid", LAST);
```

この関数は **rowid** の値を取得し，その長さをパラメータ **rowid_length** に 16 進形式で保存します。

SWETS（タイム・スタンプ）

スクリプト内の SWETS の値は，1970 年 1 月 1 日午前 0 時を基点として経過したミリ秒数です。

VuGen は，スクリプト内にある空でないすべてのタイム・スタンプを，パラメータ {SiebelTimeStamp} に置き換えます。このパラメータに値を保存する前に，VuGen は次の関数を生成します。

```
web_save_timestamp_param("SiebelTimeStamp", LAST);
```

この関数によって，現在のタイム・スタンプが **SiebelTimeStamp** パラメータに保存されます。

Siebel-Web 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，スクリプトを作成するときに発生する可能性のあるエラーとブレークダウン診断ツールについて説明します。

- 一般的なエラー
- ブレークダウン情報の記録

一般的なエラー

Siebel-Web 仮想ユーザ・スクリプトの作成中に，次のエラーが 1 つまたは複数発生する場合があります。

- 「戻る」または「更新」のエラー
- 同一の値
- 「No Content」HTTP 応答
- コンテキストの復元
- レコードの検索不能
- ファイルの終わり
- 検索カテゴリの取得不能

「戻る」または「更新」のエラー

「**戻る**」または「**更新**」に関連するエラー・メッセージでは，通常は次のようなテキストが表示されます。

We are unable to process your request.This is most likely because you used the browser BACK or REFRESH button to get to this point.

原因：次の原因が考えられます。

- SWEC が現在の要求と正しく相関されていなかった。
- SWETS が現在の要求と正しく相関されていなかった。
- SWEC が更新されないまま，要求が 2 回 Siebel サーバに送信された。
- 前の要求によってブラウザがダウンロードを行うためのフレームが開かれている必要があった。しかし，おそらくは記録後に SWEMethod が変更されたために，このフレームがサーバに作成されていなかった。

同一の値

同一の値のエラーに対しては，通常は次のような Web ページの応答が表示されます。

@0`0`3`3``0`UC`1`Status`Error`SWEC`10`0`1`Errors`0`2`0`Level0`0`ErrMsg`The same values for 'Name' already exist. If you would like to enter a new record, please 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 ページのレコードの行 ID が入力名 SWERowId に含まれていないことが考えられます。入力名はパラメータ化しておく必要があります。パラメータのソース値でその場所が変更されている可能性があります。

ファイルの終わり

ファイルの終わりタイプのエラーでは，通常は次のような Web ページ応答があります。

@0`0`3`3``0`UC`1`Status`Error`SWEC`28`0`1`Errors`0`2`0`Level0`0`ErrMsg`An end of file error has occurred. Please continue or ask your systems administrator to check your application configuration if the problem persists.`ErrCode`28601`

原因：：Web ページのレコードの行 ID が入力名 SWERowId に含まれていないことが考えられます。入力名はパラメータ化しておく必要があります。パラメータのソース値でその場所が変更されている可能性があります。

検索カテゴリの取得不能

検索カテゴリの取得不能タイプのエラーでは，通常は次のような Web ページ応答があります。

原因：：検索フレームがサーバからダウンロードされなかったことが考えられます。この問題は，前の要求で検索フレームを正しく作成するようサーバに要求していなかったときに発生します。

ブレークダウン情報の記録

VuGen には，テストのトランザクション・コンポーネントを理解するための診断ツールとして，「**トランザクション・ブレークダウン**」が用意されています。トランザクション・ブレークダウン情報を使用して，ボトルネックとなっている場所と解決の必要がある問題を特定できます。

トランザクション・ブレークダウンのスクリプトを準備する場合は，テスト 1 時間当たり 1 秒の割合で各トランザクションの最後に思考遅延時間を追加することをお勧めします。思考遅延時間の入力の詳細については，『**第 1 巻 - 仮想**

ユーザ・ジェネレータの使い方』の「仮想ユーザ・スクリプトの拡張」を参照してください。

トランザクション・ブレークダウン情報を記録するためには，お使いのスクリプト内のパラメータ化されたスクリプトを変更する必要があります。

トランザクション・ブレークダウンのスクリプトを準備するには，次の手順を実行します。

1 Session ID の代わりとなるスクリプト・パラメータを特定します。

```
/* ソース・タスク ID 15 からのパラメータの登録
    // {Siebel_sn_body4} = "28eMu9uzkn.YGFFevN1FdrCfCCOc8c_"
    // */
    web_reg_save_param("Siebel_sn_body4",
        "LB/IC=_sn=",
        "RB/IC=&",
        "Ord=1",
        "Search=Body",
        "RelFrameId=1",
        LAST);
```

2 次の **web_submit_data** 関数を，**lr_start_transaction** 関数と **lr_end_transaction** 関数で囲み，トランザクションとしてマークします。

3 トランザクションの末尾の前に，**lr_transaction_instance_add_info** への呼び出しを追加します。ここで，最初のパラメータ 0 は必須で，セッション ID には SSQLBD 接頭辞が付きます。

```
lr_start_transaction("LoginSQLSync");
    web_submit_data("start.swe_2",
        "Action=http://design/callcenter_enu/start.swe",
        "Method=POST",
        "RecContentType=text/html",
        "Referer=http://design/callcenter_enu/start.swe",
        "Snapshot=t2.inf",
        "Mode=HTML",
        ITEMDATA,
        "Name=SWEUserName", "Value=wrun", ENDITEM,
        "Name=SWEPassword", "Value=wrun", ENDITEM,
        "Name=SWERememberUser", "Value=Yes", ENDITEM,
        "Name=SWENeedContext", "Value=false", ENDITEM,
        "Name=SWEFo", "Value=SWEEntryForm", ENDITEM,
        "Name=SWETS", "Value={SiebelTimeStamp}", ENDITEM,
        "Name=SWECmd", "Value=ExecuteLogin", ENDITEM,
        "Name=SWEBID", "Value=-1", ENDITEM,
        "Name=SWEC", "Value=0", ENDITEM,
        LAST);

lr_transaction_instance_add_info(0,lr_eval_string("SSQLBD:{Siebel_sn_body4}
"));
lr_end_transaction("LoginSQLSync", LR_AUTO);
```

注：Session ID の矛盾を避けるには，各セッションの最後に仮想ユーザがデータベースからログオフしていることを確認してください。

第 45 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，以下の項目について説明します。

- Baan 仮想ユーザ・スクリプトの作成について
- 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・スクリプトの作成について

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。

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。

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 10 部

レガシ・プロトコル

第 46 章

RTE 仮想ユーザ・スクリプトの紹介

RTE 仮想ユーザは，Windows 環境でターミナル・エミュレータを操作します。本章では，Windows ベースの RTE 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。

本章では，次の項目について説明します。

- RTE 仮想ユーザ・スクリプトの作成について
- RTE 仮想ユーザの紹介
- RTE 仮想ユーザ技術について
- RTE 仮想ユーザ・スクリプトの概要
- TE 関数の使用
- ターミナル・キーの PC キーボード・キーへの割り当て

以降の情報は，RTE（Windows）仮想ユーザ・スクリプトを対象とします。

RTE 仮想ユーザ・スクリプトの作成について

RTE 仮想ユーザは，クライアント / サーバ・システムの負荷テストを行うために，ターミナル・エミュレータを操作します。

VuGen を使用してターミナル・エミュレータ・セッションを記録することにより，実際のユーザのアクションを表現します。記録したスクリプトは，トランザクション関数や同期化機能によって拡張できます。

本章では，Windows ベースの RTE 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。

RTE 仮想ユーザの紹介

RTE 仮想ユーザは，文字入力データをターミナル・エミュレータに入力し，それらのデータをサーバに送信した後，サーバから応答があるまで待機します。例えば，あるメンテナンス会社の顧客情報を管理するサーバがあるとします。フィールド・サービス担当者は，修理を行うたびにターミナル・エミュレータを使ってモデム経由でサーバのデータベースにアクセスします。サービス担当者は顧客の情報にアクセスし，自分が行った修理の詳細を記録します。

RTE 仮想ユーザを使用して，この事例をエミュレートできます。RTE 仮想ユーザは次の操作を実行します。

1 コマンド・ラインで「**60**」と入力して，アプリケーション・プログラムを開きます。

2 フィールド・サービス担当者の番号として「**F296**」と入力します。

3 顧客番号として「**NY270**」と入力します。

4 画面上に 「詳細」という単語が表示されるまで待機します。「詳細」の表示は，画面上に顧客の詳細がすべて表示されたことを示します。

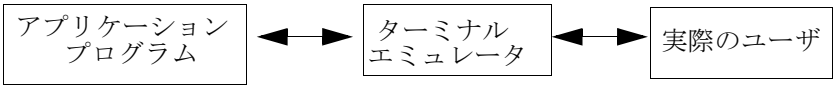
5 最新の修理の詳細として「**ガスケット P249 を交換，主要サービスを実施**」と入力します。

6「**Q**」と入力してアプリケーション・プログラムを閉じます。

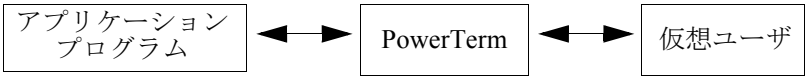
RTE 仮想ユーザ・スクリプトを作成するには，VuGen を使用します。スクリプト・ジェネレータは，実際のユーザがターミナル・エミュレータで行ったアクションを記録します。ターミナル・ウィンドウでのキーボード入力を記録し，対応するステートメントを生成し，それらを仮想ユーザ・スクリプトに挿入します。記録の実行中，スクリプト・ジェネレータはスクリプトに同期化関数を自動的に挿入します。詳細については，第 49 章「RTE 仮想ユーザ・スクリプトの同期化」を参照してください。

RTE 仮想ユーザ技術について

RTE 仮想ユーザは，実際のユーザのアクションをエミュレートします。実際のユーザは，ターミナルまたはターミナル・エミュレータを使用してアプリケーション・プログラムを操作します。



RTE 仮想ユーザ環境では，仮想ユーザが実際のユーザの代わりに操作を行います。仮想ユーザは PowerTerm というターミナル・エミュレータを操作します。



PowerTerm は，標準的なターミナル・エミュレータと同じように動作し，IBM 3270，IBM 5250，VT100，VT220 などの一般的なプロトコルをサポートします。

RTE 仮想ユーザ・スクリプトの概要

本項では，VuGen を使った RTE 仮想ユーザ・スクリプトの作成プロセスの概略を説明します。

RTE 仮想ユーザ・スクリプトを作成するには，次の手順を実行します。

1 **VuGen を使用して基本となるスクリプトを記録します。**

仮想ユーザ・ジェネレータ（VuGen）を使用して，ターミナル・エミュレータで行われる操作を記録します。ターミナル・ウィンドウでのキーボード入力を記録し，対応するステートメントを生成して仮想ユーザ・スクリプトに挿入します。

詳細については，第 47 章「RTE 仮想ユーザ・スクリプトの記録」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，同期化関数，および制御フロー構造を挿入して，スクリプトを拡張します。

詳細については，『**第 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「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ことにより，同じビジネス・プロセスを，異なる値で何度も繰り返すことができます。

詳細については，『**第 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「パラメータの作成」を参照してください。

4 **実行環境を設定します。**

実行環境を設定することによって，スクリプト実行時の仮想ユーザの動作を制御します。設定には，ループ，ログ，タイミング情報などがあります。

詳細については，『**第 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「実行環境の設定」を参照してください。

5 **VuGen からスクリプトを実行します。**

VuGen からスクリプトを実行して，スクリプトが正しく実行されることを確認します。標準出力を表示して，プログラムがサーバと正常に通信していることを確認します。

詳細については，『**第 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「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。

RTE スクリプトが正常に作成されたら，スクリプトをシナリオ，プロファイル，またはセッション・ステップに統合します。スクリプトをシナリオ，プロファイル，またはセッション・ステップに統合する方法の詳細については，該当するユーザーズ・ガイドを参照してください。

TE 関数の使用

ターミナルとサーバの通信をエミュレートするために開発された関数を，TE 仮想ユーザ関数と呼びます。TE 仮想ユーザ関数の名前には，**TE** という接頭辞が付きます。VuGen は，RTE セッション中に，本項に列挙する TE 関数のほとんどを自動的に記録します。スクリプトに任意の関数を手作業でプログラミングすることもできます。TE 関数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。

ターミナル・エミュレータ接続関数

TE_connect	ターミナル・エミュレータを指定されたホストに接続します。

テキスト取得関数

TE_find_text	画面の指定された領域でテキストを検索します。
TE_get_line_attribute	テキストの形式に関する情報を返します。
TE_get_text_line	画面の指定された行からテキストを読み取ります。

カーソル関数

TE_get_cursor_pos	カーソルの現在の位置を返します。
TE_set_cursor_pos	ターミナル画面上のカーソルの位置を設定します。

システム変数関数

TE_getvar	RTE システム変数の値を返します。
TE_setvar	RTE システム変数の値を設定します。

エラー・コード関数

TE_perror	エラー・コードを出力ウィンドウに出力します。
TE_sperror	エラー・コードを文字列に変換します。

入力関数

TE_type	形式を整えた文字列をクライアント・アプリケーションに送信します。
TE_typing_style	ターミナル・エミュレータにテキストを入力する方法を指定します。
TE_unlock_keyboard	メインフレーム・ターミナルのキーボードのロックを解除します。

同期化関数

TE_wait_cursor	ターミナル・ウィンドウの指定された位置にカーソルが表示されるまで待機します。
TE_wait_silent	クライアント・アプリケーションが安定するのを指定された秒数だけ待ちます。
TE_wait_sync	システムが X SYSTEM（入力禁止）モードから戻るまで待機します。
TE_wait_sync_transaction	システムが最後に X SYSTEM モードになっていた時間を記録します。
TE_wait_text	指定された位置に文字列が表示されるまで待機します。

パラメータ化できる TE 関数は，**TE_connect**，**TE_find_text**，**TE_get_text_line**，および **TE_type** です。仮想ユーザ・スクリプトの関数をパラメータ化する方法の詳細については，『**第 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「パラメータの作成」を参照してください。

ターミナル・キーの PC キーボード・キーへの割り当て

ターミナル・エミュレータを使用する場合は，ターミナル・キーボードの代わりに PC キーボードを使用します。ターミナル・キーボードには，PC キーボードにはないキーが多数あります。このようなキーの例として，IBM 5250 キーボードの HELP キー，AUTOR キー，PUSH キーなどがあります。ターミナル・エミュレータや関連するアプリケーション・プログラムを正常に動作させるには，特定のターミナル・キーを PC キーボードのキーに割り当てることが必要な場合があります。

ターミナル・キーを PC キーボードのキーに割り当てるには，次の手順を実行します。

1 ターミナル・エミュレータで，［**オプション**］＞［**キーボードの割り当て**］を選択するか，［**キーボードの割り当て**］ボタンをクリックします。［キーボード割り当て］ダイアログ・ボックスが開きます。

2 ツールバーの［**キーボードの割り当て**］ボタンをクリックします。ターミナル・キーを PC キーに割り当てるには，上側のターミナル・キーボードのキーを下側の PC キーボードのキーにドラッグします。

上側のキーボードで Shift キー，Ctrl キー，あるいはその両方をクリックすると，追加のキー機能が表示されます。追加のキー機能は，これらのキーのいずれかを最初に選択しない限り表示されません。キーが表示された後は，上側のターミナル・キーボードの必要なキーを下側の PC キーボードのキーにドラッグできます。

定義を取り消すには，PC キーの定義をごみ箱にドラッグします。これで，PC キーの機能が標準設定に戻ります。

標準の割り当てに戻すには，［**標準設定**］をクリックします。

第 47 章

RTE 仮想ユーザ・スクリプトの記録

VuGen を使用して，Windows ベースのリモート・ターミナル・エミュレーション（RTE）仮想ユーザ・スクリプトを記録できます。

本章では，次の項目について説明します。

- RTE 仮想ユーザ・スクリプトの記録について
- RTE 仮想ユーザ・スクリプトの新規作成
- ターミナルの設定と接続の手順の記録
- 一般的なユーザ・アクションの記録
- ログオフ手順の記録
- RTE 記録オプションの設定
- ターミナル・エミュレータへの入力
- 一意のデバイス名の生成
- フィールド区分文字

以降の情報は，ターミナル・エミュレーション（RTE）仮想ユーザ・スクリプトを対象とします。

RTE 仮想ユーザ・スクリプトの記録について

VuGen を使用して，Windows ベースの RTE 仮想ユーザ・スクリプトを記録できます。VuGen は PowerTerm ターミナル・エミュレータを使用して，広範囲なターミナル・タイプをエミュレートします。PowerTerm は，一般的な端末接続と，その後の一般的なビジネス・プロセスの実行に使用します。その後に，ログオフ手順を実行します。ターミナル・エミュレータ内で一般的なユーザ・アクションを実行している間，VuGen は対応するステートメントを生成し，それらを仮想ユーザ・スクリプトに挿入します。記録中に，スクリプトを表示および編集することができます。

RTE 仮想ユーザ・スクリプトを記録する前に，記録オプションが正しく設定されていることを確認してください。記録オプションを使用することで，仮想ユーザ・スクリプトの記録中に VuGen によって特定の関数が生成される方法を制御できます。記録オプションは，以降のすべての記録セッションの間適用されます。

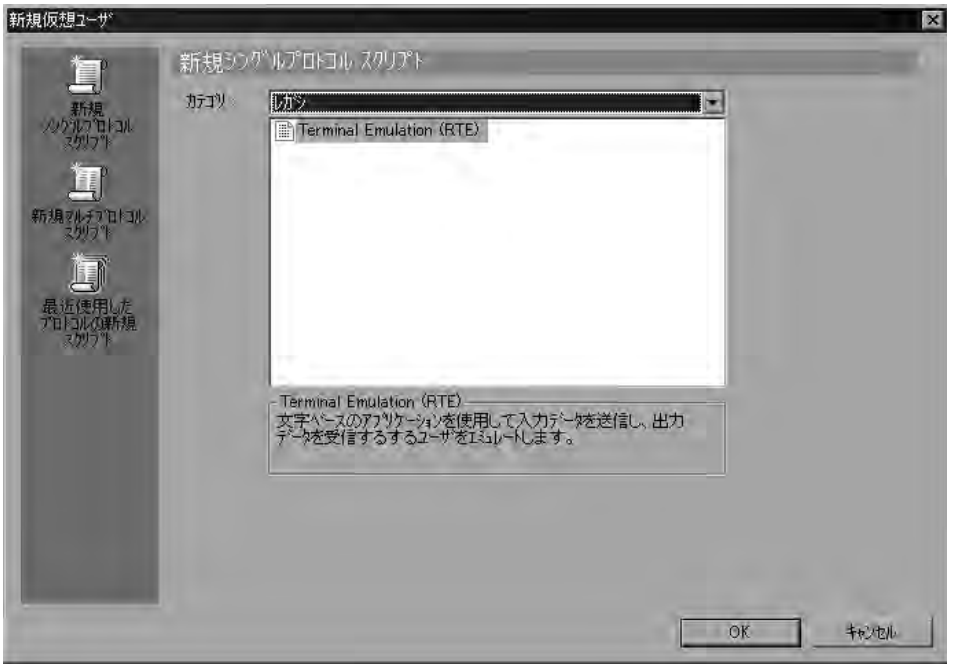
RTE 仮想ユーザ・スクリプトの新規作成

ユーザのアクションを仮想ユーザ・スクリプトに記録する前に，仮想ユーザ・スクリプトを開く必要があります。既存のスクリプトを開くか，新しいスクリプトを作成します。新しい仮想ユーザ・スクリプトを作成するには VuGen を使用します。

新しい RTE 仮想ユーザ・スクリプトを作成するには，次の手順を実行します。

1 製品の開始メニューから［**仮想ユーザ ジェネレータ**］を選択します。VuGen のウィンドウが開きます。

2［**新規作成**］ボタン をクリックします。［**新規仮想ユーザ**］ダイアログ・ボックスが開きます。



3［**レガシ**］プロトコル・タイプを選択し，［**Terminal Emulator（RTE)**］を選択します。［**OK**］をクリックします。VuGen によって空の RTE スクリプトが生成され，カーソルの位置が **vuser_init** セクション内で記録を開始する位置で表示されます。

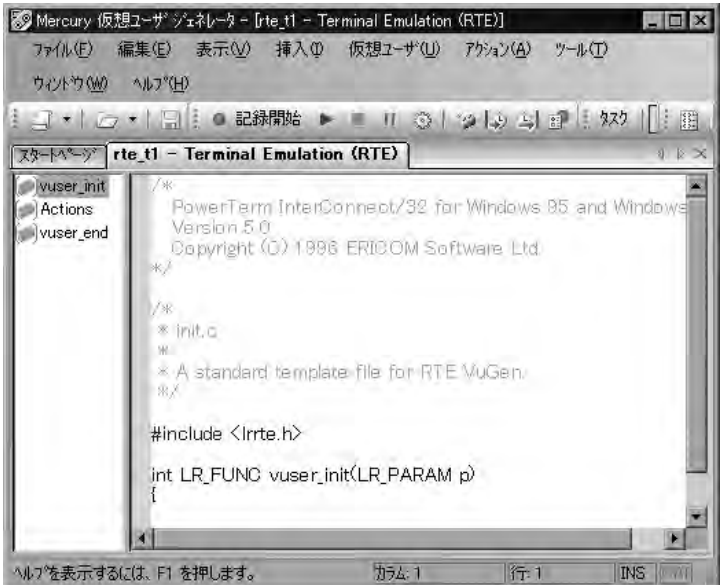
ターミナルの設定と接続の手順の記録

スケルトンの仮想ユーザ・スクリプトを作成した後，ターミナルの設定と接続の手順をスクリプトに記録します。VuGen は，RTE 仮想ユーザ・スクリプトを記録するときに PowerTerm ターミナル・エミュレータを使用します。

ターミナルの設定と接続の手順を記録するには，次の手順を実行します。

1 既存の RTE 仮想ユーザ・スクリプトを開くか，新しい RTE 仮想ユーザ・スクリプトを作成します。

2［**セクション**］ボックスで，記録されたステートメントの挿入先となるセクションを選択します。使用できるセクションは **vuser_init**，**Action**，および **vuser_end** です。



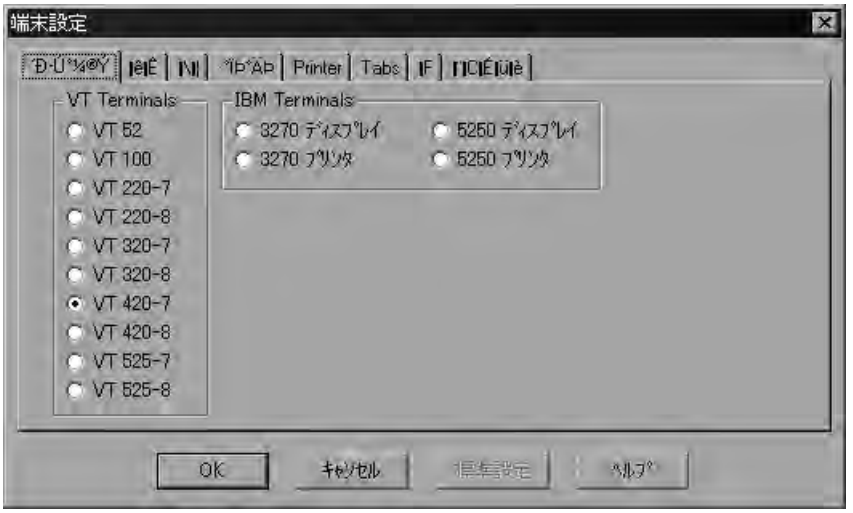
注 : ターミナルの設定と接続の手順は，常に **vuser_init** セクションに記録します。**vuser_init** セクションは，仮想ユーザ・スクリプトの反復を複数回実行するときに繰り返されません。繰り返されるのは **Actions** セクションだけです。反復の設定の詳細については，『**第 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「実行環境の設定」を参照してください。

---

3 仮想ユーザ・スクリプト内で，記録を開始する位置にカーソルを置きます。

4［**記録**］ボタン をクリックします。PowerTerm のメイン・ウィンドウが開きます。

5 PowerTerm のメニュー・バーから［**端末**］＞［**設定**］を選択し，［端末設定］ダイアログ・ボックスを表示します。



6 エミュレーションのタイプを VT ターミナル・タイプおよび IBM ターミナル・タイプから選択し，［**OK**］をクリックします。

注： AS/400 マシンまたは IBM メインフレームに接続する場合は，IBM ターミナル・タイプを選択します。UNIX ワークステーションに接続する場合は，VT ターミナル・タイプを選択します。

7［**通信**］＞［**接続**］を選択し，［接続］ダイアログ・ボックスを表示します。

8［**セッション タイプ**］で，使用する通信のタイプを選択します。

9［**パラメータ**］で，必要なオプションを指定します。使用できるパラメータは，選択したセッションのタイプによって異なります。パラメータの詳細については，［**ヘルプ**］をクリックしてください。

注：定義したパラメータは，保存して将来再利用することができます。パラメータを保存するには，［**名前を付けて保存**］をクリックします。保存したパラメータ・セットが［セッション リスト］ボックスに表示されます。

10［**接続**］をクリックします。PowerTerm は指定されたシステムに接続し，VuGen は **TE_connect** 関数をスクリプトの挿入ポイントに挿入します。

TE_connect ステートメントの形式は次のとおりです。

```
/* *** ターミナル・タイプは VT220-7 */
 TE_connect(
  "comm-type = telnet;"
  "host-name = pharaoh;"
  "set-window-size = true;"
  "security-type = unsecured;"
  "telnet-binary-mode = true;"
  "terminal-type = vt220-7;"
  "terminal-model = vt320;"
  , 60000);
 if (TE_errno != TE_SUCCESS)
  return -1;
```

挿入された **TE_connect** ステートメントの後には if ステートメントが続きます。これは，再生中に **TE_connect** 関数の実行が成功したかどうかを調べます。

注：仮想ユーザ・スクリプトの中でサーバへの接続（**TE_connect**）を複数記録しないでください。

以上で，ターミナルの設定と接続の手順が完了します。これで，仮想ユーザ・スクリプトへの一般的なユーザ・アクションの記録を開始する準備が整います。次項ではこの方法について説明します。

一般的なユーザ・アクションの記録

設定手順を記録した後，一般的なユーザ・アクションまたはビジネス・プロセスを実行します。これらのプロセスは仮想ユーザ・スクリプトの **Actions** セクションに記録します。仮想ユーザ・スクリプトの反復を複数回実行するときに繰り返されるのは，スクリプトの **Actions** セクションだけです。反復の設定の詳細については，『**第 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「実行環境の設定」を参照してください。

セッションを記録する際，VuGen によって記録されるのはテキストのストローク（打鍵内容）であり，テキストではありません。したがって，コマンドを直接入力する代わりに PowerTerm ウィンドウにコピーして貼り付けることはお勧めできません。

ユーザ・アクションを記録するには，次の手順を実行します。

1 既存の RTE 仮想ユーザ・スクリプトを開き，［**セクション**］ボックスで **Actions** をクリックします。

2 引き続き，ターミナル・エミュレータで一般的なユーザ・アクションを実行します。入力中，VuGen は対応するステートメントを生成し，それらを仮想ユーザ・スクリプトに挿入します。必要ならば，記録されたステートメントをスクリプトの記録中に編集できます。

注：標準設定では，VuGen は連続するキーストロークの合間に，対応する **TE_type** 関数が生成されるまで最大 5 秒待ちます。この待ち時間を変更する場合は，705 ページ「RTE 記録オプションの設定」を参照してください。

一般的なユーザ・アクションの記録が完了したら，引き続きログオフ手順を記録します。次項ではこの方法について説明します。

ログオフ手順の記録

仮想ユーザのログオフ・プロセスは仮想ユーザ・スクリプトの **vuser_end** セクションに記録します。**vuser_end** セクションは，スクリプトの反復を多数回実行するときに繰り返されません。反復の設定の詳細については，『**第 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「実行環境の設定」を参照してください。

ログオフ手順を記録するには，次の手順を実行します。

1 前の項で説明した方法で，一般的なユーザ・アクションの実行と記録を必ず済ませておきます。

2 VuGen のメイン・ウィンドウの［**セクション**］ボックスで，**vuser_end** をクリックします。

3 ログオフ手順を実行します。VuGen によって手順がスクリプトの **vuser_end** セクションに記録されます。

4［記録］ツールバーで，［**停止**］ボタンをクリックします。VuGen のメイン・ウィンドウに，記録されたすべてのステートメントが表示されます。

5［**上書き保存**］ボタン をクリックして，記録されたセッションを保存します。［名前を付けて保存］ダイアログ・ボックスが表示されます（新規仮想ユーザ・スクリプトの場合のみ)。スクリプト名を指定します。記録されたスクリプトは，VuGen のメイン・ウィンドウで手作業で編集できます。

RTE 設定オプションの設定

ターミナル・エミュレーション中に使用される文字セットに合わせて，記録オプションを設定できます。標準設定の文字セットは ANSI です。漢字その他のマルチバイト・プラットフォームの場合は，DBCS（ダブルバイト文字セット）を指定できます。

［設定］記録オプションを表示するには，ツールバーの［**記録オプション**］ボタン をクリックするか，［**ツール**］＞［**記録オプション**］を選択します。［RTE］の［**設定**］ノードを選択します。

RTE 記録オプションの設定

記録オプションを設定することで，VuGen によって RTE 関数用に生成されるコードをカスタマイズできます。記録オプションの設定には［記録オプション］ダイアログ・ボックスを使います。

［記録オプション］ダイアログ・ボックスを表示するには，ツールバーの［**記録オプション**］ボタン をクリックするか，［**ツール**］＞［**記録オプション**］を選択します。［RTE］の［**RTE**］ノードを選択します。

次の記録オプションを設定できます。

- 自動同期化コマンド
- 自動画面ヘッダー・コメント（IBM 端末のみ）
- 自動 X-System トランザクション（IBM 端末のみ）
- キーボード記録のタイムアウト

自動同期化コマンド

VuGen では，記録中にいくつかの TE 同期化関数を自動的に生成し，それらをスクリプトに挿入することができます。

1 記録中に新しい画面が表示されるたびに **TE_wait_sync** 関数を生成するよう，VuGen を設定できます。これには，［記録オプション］ダイアログ・ボックスの［**X- システム**］チェック・ボックスを選択します。

標準設定では，記録中に新しい画面が表示されるたびに，VuGen によって **TE_wait_sync** 関数が生成されます。

注 : IBM ブロック・モード・ターミナルのみを記録しているときは，VuGen によって **TE_wait_sync** 関数が生成されます。

---

2 それぞれの **TE_type** 関数の前に **TE_wait_cursor** 関数を生成するよう，VuGen を設定できます。これには，［記録オプション］ダイアログ・ボックスの［**カーソル**］チェック・ボックスを選択します。

標準設定では，**TE_wait_cursor** 関数は自動的には生成されません。

3 それぞれの **TE_type** 関数の前に **TE_wait_text** 関数を生成するよう，VuGen を設定できます（適切な場合）。これには，［記録オプション］ダイアログ・ボックスの［**プロンプト**］チェック・ボックスを選択します。

標準設定では，それぞれの **TE_type** 関数の前に **TE_wait_cursor** 関数が自動的には生成されません。

注 : VT タイプのターミナルのみを記録しているときは，VuGen によって，意味のある **TE_wait_text** 関数が生成されます。ブロック・モード（IBM）ターミナルを記録しているときは，**TE_wait_text** 関数の自動生成は使用しないでください。

自動画面ヘッダー・コメント（IBM 端末のみ）

仮想ユーザ・スクリプトの記録中に画面ヘッダー・コメントを自動的に生成し，それらのコメントをスクリプトに挿入するよう，VuGen を設定できます。

生成されたコメントをもとに，それぞれの新しい画面をターミナル・エミュレータ内で表示されているとおりに識別できるため，記録されたスクリプトが読みやすくなります。生成されるコメントには，ターミナル・エミュレータ・ウィンドウの 1 行目に表示されているテキストが格納されます。次のコメントは，Office Tasks という画面がターミナル・エミュレータに表示されていたことを示しています。

```
/* OFCTSK                Office Tasks                */
```

スクリプトの記録中に VuGen によってコメントが自動的に生成されるようにするには，［記録オプション］ダイアログ・ボックスの［Generate screen header comments］チェック・ボックスを選択します。

標準設定では，画面コメントは自動的には生成されません。

注：コメントの自動生成は，IBM 5250 などのブロック・モードのターミナル・エミュレータを使用しているときのみ可能です。

自動 X-System トランザクション（IBM 端末のみ）

チューニング・セッションまたはシナリオの実行中に，システムが X SYSTEM モードになっていた時間を記録するよう，VuGen を設定できます。この場合，VuGen によって，それぞれの **TE_wait_sync** 関数の後に **TE_wait_sync_transaction** 関数が挿入されます。**TE_wait_sync_transaction** 関数はそれぞれ，「default」という名前を持つトランザクションを作成します。**TE_wait_sync_transaction** 関数はそれぞれ，システムが以前の X SYSTEM 状態を保っていた時間を記録します。

記録中に **TE_wait_sync_transaction** ステートメントを挿入するよう VuGen を設定するには，［記録オプション］ダイアログ・ボックスの［**X- システム用自動トランザクションを作成する**］チェック・ボックスを選択します。

標準設定では，トランザクションは自動的には生成されません。

キーボード記録のタイムアウト

記録中にターミナル・エミュレータにテキストを入力すると，テキスト入力が VuGen によって監視されます。各キーストロークの後，VuGen は次のキーストロークが発生するまで，指定の時間量を最大とする時間だけ待ちます。指定の時間内に次のキーストロークが発生しなかった場合は，コマンドが完了したものと見なされます。すると，VuGen によって，対応する **TE_type** 関数が生成され，スクリプトに挿入されます。

連続するキーストロークの合間に VuGen が待つ最大時間を設定するには，［**キーボード記録タイムアウト**］ボックスにその時間量を入力します。

標準設定では，VuGen は連続するキーストロークの合間に，対応する **TE_type** 関数が生成されるまで最大 5 秒待ちます。

ターミナル・エミュレータへの入力

2 つの TE 仮想ユーザ関数を使用すると，仮想ユーザが PowerTerm ターミナル・エミュレータに文字を「入力」できるようになります。

- **TE_type** は，文字をターミナル・エミュレータに送ります。記録中，ターミナル・ウィンドウへのキーボード入力に対して，**TE_type** 関数が VuGen によって自動的に生成されます。詳細については，709 ページ「TE_type 関数の使用」を参照してください。
- **TE_typing_style** は，仮想ユーザの入力速度を決めます。**TE_typing_style** 関数を仮想ユーザ・スクリプトに挿入することによって，入力のスタイルを手作業で定義できます。詳細については，710 ページ「入力スタイルの設定」を参照してください。または，実行環境の設定から入力スタイルを設定することもできます。詳細については，第 48 章「RTE 実行環境の設定」を参照してください。

注： RTE 仮想ユーザ・スクリプトを記録するときは，マウスを使用してターミナル・エミュレータ・ウィンドウ内のカーソルの位置を変更しないでください。これらのカーソル移動は記録されません。

TE_type 関数の使用

スクリプトを記録すると，VuGen によってすべてのキーボード入力が記録され，対応する **TE_type** 関数が生成されます。実行中は，**TE_type** 関数によって，整形された文字列がターミナル・エミュレータに送られます。

キーボード入力は通常のテキスト文字列として定義されます（空白も含まれます）。次に例を示します。

```
TE_type ("hello, world");
```

2 文字以上の入力キー名は，文字 **k** で始まる識別子によって表され，大なり記号と小なり記号（<>）で囲まれます。

例えば，次の関数は，Return キーと，その後の Control キーと y キーの入力を表しています。

```
TE_type("<kReturn><kControl-y>");
```

その他の例としては，<kF1>，<kUp>，<kF10>，<kHelp>，<kTab> などがあります。

キー名を調べるには，キーの操作を記録した後，記録されたステートメントで名前を調べます。

注：TE_type ステートメントを（記録するのではなく）プログラムするときは，「**オンライン関数リファレンス**」（［**ヘルプ**］＞［**関数リファレンス**］）に記載されているキー定義を使用してください。

TE_type のタイムアウト値の設定

システムが X SYSTEM モード（または入力禁止モード）にある間，仮想ユーザが **TE_type** ステートメントを送信しようとした場合，その仮想ユーザは X SYSTEM モードが終了するまで入力を待たされます。システムが TE_XSYSTEM_TIMEOUT ミリ秒を超えて X SYSTEM モードを保持した場合は，**TE_type** 関数が TE_TIMEOUT エラーを返します。

TE_XSYSTEM_TIMEOUT の値は **TE_setvar** を使用して設定できます。TE_XSYSTEM_TIMEOUT の標準値は 30 秒です。

仮想ユーザに対する事前入力の許可

場合によっては，システムが X SYSTEM モード（または入力禁止モード）にある間も，仮想ユーザに対してキーストロークの送信を許可したいことがあります。例えば，仮想ユーザに対して Break キーの押下を許可することができます。システムが X SYSTEM モードにある間も仮想ユーザに対してキーストロークの送信を許可するには，TE_ALLOW_TYPEAHEAD 変数を使用します。

事前入力を禁止するには，TE_ALLOW_TYPEAHEAD を 0 に設定し，事前入力を許可するには，0 以外の値に設定します。TE_ALLOW_TYPEAHEAD の値を設定するには **TE_setvar** を使用します。標準設定では，TE_ALLOW_TYPEAHEAD は 0 に設定されており，X SYSTEM モード中はキーストロークが送られません。

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

入力スタイルの設定

RTE 仮想ユーザに対しては，FAST と HUMAN の 2 つの入力スタイルを設定できます。FAST スタイルでは，仮想ユーザからターミナル・エミュレータへの入力ができるだけ高速に実行されます。HUMAN スタイルでは，文字を入力するたびに仮想ユーザが一時停止します。このため，人間のユーザによるキーボードでの入力が，仮想ユーザによって，より正確にエミュレートされます。

入力スタイルは **TE_typing_style** 関数を使用して設定します。**TE_typing_style** 関数の構文は次のとおりです。

int **TE_typing_style (**char ***style);**

ここで，**style** には FAST または HUMAN を指定できます。標準設定の入力スタイルは HUMAN です。HUMAN の入力スタイルを選択する場合は，次の形式になります。

HUMAN, **delay** [,**first_delay**]

delay には，キーストローク間の間隔（ミリ秒）を指定します。省略可能なパラメータ **first_delay** には，文字列の 1 文字目を入力する前の待ち時間（ミリ秒）を指定します。次に例を示します。

```
TE_typing_style ("HUMAN, 100, 500");
TE_type ("ABC");
```

上記の場合は，仮想ユーザは文字 A を入力する前に 0.5 秒待ちます。次に，文字 「B」を入力する前に 0.1 秒待ち，その次に文字「C」を入力する前にさらに 0.1 秒待ちます。

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

TE_typing_style 関数を使用する以外に，実行環境の設定を使用して入力スタイルを設定することもできます。詳細については，第 48 章「RTE 実行環境の設定」を参照してください。

一意のデバイス名の生成

APPC などの一部のプロトコルでは，システムにログオンするターミナルごとに一意のデバイス名が必要になります。実行環境の設定を使用して，**TE_connect** 関数から仮想ユーザごとに 8 文字の一意のデバイス名を生成し，その名前を使用して接続するよう指定することができます。この方法では一意性の条件は満たされますが，システムによっては，デバイス名が特定の形式に従う必要があるなど，さらに別の条件が必要になる場合があります。実行環境設定の詳細については，『VuGen ユーザーズ・ガイド』の「実行環境の設定」を参照してください。

TE_connect 関数が仮想ユーザのシステムへの接続の際に使用するデバイス名について，その形式を定義するには，**RteGenerateDeviceName** 関数を仮想ユーザ・スクリプトに追加します。この関数のプロトタイプは次のとおりです。

```
void RteGenerateDeviceName(char buf[32])
```

デバイス名は **buf** に書き込まれます。

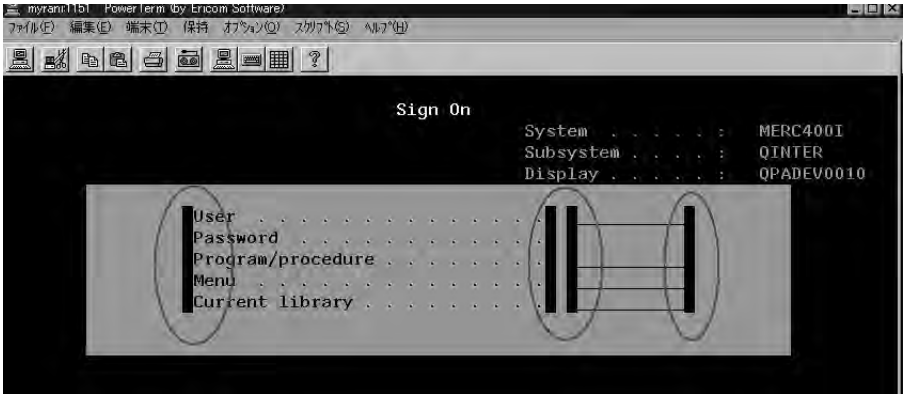
RteGenerateDeviceName 関数が仮想ユーザ・スクリプト内に存在する場合は，新しいデバイス名が必要になるたびに，仮想ユーザからこの関数が呼び出されます。**RteGenerateDeviceName** 関数がスクリプト内で定義されておらず，一意のデバイス名が必要になった場合は，**TE_connect** 関数によって必要な名前が生成されます。

次の例では，**RteGenerateDeviceName** 関数によって「TERMx」という形式の一意のデバイス名が生成されます。最初の名前が TERM0 となり，以降 TERM1，TERM2，という具合になります。

```
RteGenerateDeviceName(char buf[32])
{
      static int n=0;
      sprintf(buf, "TERM%d", n);
      n=n+1;
}
```

フィールド区分文字

一部のターミナル・エミュレータでは，各フィールドの先頭と末尾を示すために区分文字が使用されています。これらの区分文字は表示されず，画面上では空白として出現します。次に示すターミナル・エミュレータでは，フィールド区分文字を表示するために画面の中央部分の色が反転されています。これらの文字は楕円で囲まれます。



TE_wait_text，**TE_get_text**，および **TE_find_text** 関数は，画面の指定の部分にある文字を識別することによって動作します。指定の部分の中にフィールド区分文字がある場合は，その文字を空白として識別するか，または ASCII 文字として識別するかを選択できます。識別の方法を指定するには，TE_FIELD_CHARS システム変数を使用します。TE_FIELD_CHARS は 0 または 1 に設定できます。

➤ 0 の場合，フィールド区分文字の位置にある文字は空白として返されます。

➤ 1 の場合，フィールド区分文字の位置にある文字は ASCII コード（ASCII 0 または ASCII 1）として返されます。

標準設定では，TE_FIELD_CHARS は 0 に設定されています。

TE_FIELD_CHARS の値の取得および設定には，**TE_getvar** および **TE_setvar** 関数を使用します。

第 48 章

RTE 実行環境の設定

ターミナル・エミュレータ・スクリプトを記録した後は，そのスクリプトの実行環境を設定します。本章では，次のターミナル・エミュレータ仮想ユーザ実行環境の設定について説明します。

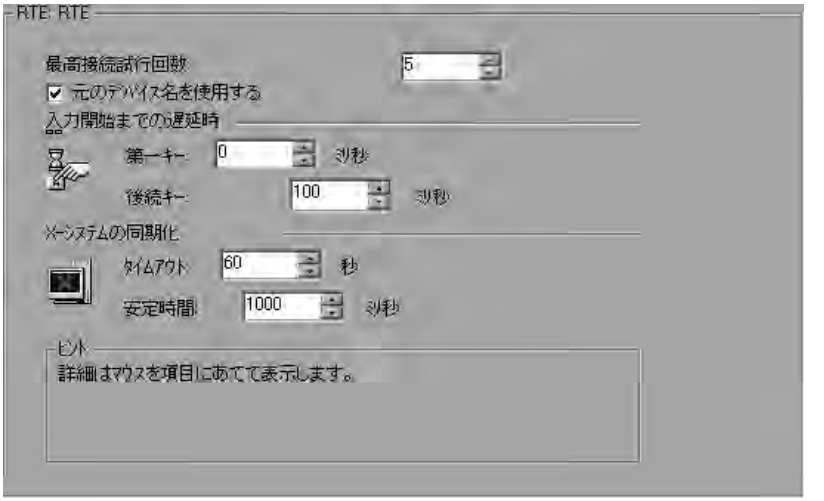
- ターミナル・エミュレータ実行環境の設定について
- 接続試行回数の変更
- 元のデバイス名の指定
- 入力遅延の設定
- X SYSTEM 同期化の設定

以降の情報は，ターミナル・エミュレータ（TE）タイプの仮想ユーザを対象とします。

ターミナル・エミュレータ実行環境の設定について

ターミナル・エミュレータ仮想ユーザ・スクリプトを作成した後は，実行環境の設定を行います。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。ターミナル・エミュレータ実行環境の設定によって，リモート・ターミナルのエミュレーションを実行する実際のユーザを正しくエミュレートするように TE 仮想ユーザを設定できます。接続試行回数，デバイス名，入力遅延，および X SYSTEM 同期化に関する設定を行うことができます。

ターミナル・エミュレータ関連の実行環境の設定は，［実行環境設定］ダイアログ・ボックスの［**RTE**］ノードで行います。



［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**］ボタンをクリックします。実行環境の設定は，LoadRunner コントローラまたは Mercury Tuning Module から変更することもできます。詳細については，各マニュアルを参照してください。

注：本章では，ターミナル・エミュレータ仮想ユーザ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 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の " 実行環境の設定 " を参照してください。

接続試行回数の変更

TE_connect 関数は，ホストへの接続を記録したときに VuGen によって生成されます。RTE 仮想ユーザ・スクリプトを再生すると，**TE_connect** 関数はターミナル・エミュレータを指定されたホストに接続します。最初の接続試行に失敗すると，仮想ユーザは一定の回数だけ接続を試行します。接続の詳細は，**output.txt** というレポート・ファイルに記録されます。

仮想ユーザが行う接続試行の最大回数を設定するには，RTE 実行環境の設定の［**最高接続試行回数**］ボックスに回数を入力します。

標準設定では，仮想ユーザは接続を 5 回試行します。

TE_connect 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。

元のデバイス名の指定

特定の環境では，各セッション（仮想ユーザ）に固有のデバイス名を付ける必要があります。**TE_connect** 関数は，各仮想ユーザに対して 8 文字からなる一意のデバイス名を生成し，この名前を使って接続します。デバイス名（**TE_connect** 関数の com_string パラメータ内に含まれる）を使って接続するには，RTE 実行環境の設定の［**元のデバイス名を使用する**］オプションを選択します。

注：元のデバイス名の設定は，IBM ブロック・モード・ターミナルにのみ適用されます。

標準設定では，仮想ユーザは元のデバイス名を使って接続します。

TE_connect 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。

入力遅延の設定

入力遅延の設定では，仮想ユーザによる **TE_type** 関数の実行方法を指定します。

仮想ユーザが文字列の最初の文字を入力する前の待機時間を指定するには，［第一キー］ボックスに値（ミリ秒単位）を入力します。

仮想ユーザがそれ以降の文字を送信するときの間隔を指定するには，［後続キー］ボックスに値（ミリ秒単位）を入力します。

最初のキー入力遅延とそれ以降のキー入力遅延の両方に 0 を入力すると，文字間の遅延はなくなり，仮想ユーザは複数の文字を 1 つの文字列として送信します。

仮想ユーザ・スクリプトの一部分で入力遅延の設定をオーバーライドするには，**TE_typing_style** 関数を使用します。

TE_type 関数と **TE_typing_style** 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。

X SYSTEM 同期化の設定

RTE 仮想ユーザ・スクリプトは，**TE_wait_sync** 関数を使って同期化を行います。すべての **TE_wait_sync** 関数に適用されるタイムアウト値と安定時間値を設定できます。**TE_wait_sync** 関数の詳細については，「**オンライン関数リファレンス**」（［**ヘルプ**］＞［**関数リファレンス**］）を参照してください。

タイムアウト

TE_wait_sync 関数を再生したときに，システムが安定しないまま同期化のタイムアウトに達すると，**TE_wait_sync** 関数からエラー・コードが返されます。同期化のタイムアウトを設定するには，RTE 実行環境の設定の［タイムアウト］セクションに値（秒単位）を入力します。

標準のタイムアウト値は 60 秒です。

安定時間

仮想ユーザは，**TE_wait_sync** 関数を実行した後，ターミナルが X SYSTEM モードに入らなくなるまで待機します。ターミナルが X SYSTEM モードから戻った後も，仮想ユーザは少しの間システムを監視します。これによって，ターミナルが安定した（システムが X SYSTEM モードに戻らない）ことを確認します。この確認ができた場合にのみ，**TE_wait_sync** 関数が終了します。

仮想ユーザが X SYSTEM モードから戻った後のシステムを監視し続ける時間を設定するには，RTE 実行環境の設定の［Stable time］ボックスに値（ミリ秒単位）を入力します。

標準の安定時間は 1000 ミリ秒です。

第 49 章

RTE 仮想ユーザ・スクリプトの同期化

RTE 仮想ユーザ・スクリプトで同期化関数を使用すると，仮想ユーザがターミナル・エミュレータに送信する入力をサーバからの応答と同期させることができます。

本章では，次の項目について説明します。

- 仮想ユーザ・スクリプトの同期化について
- ブロック・モード（IBM）・ターミナルの同期化
- キャラクタ・モード（VT）・ターミナルの同期化

以降の情報は，RTE（Windows）仮想ユーザ・スクリプトを対象とします。

仮想ユーザ・スクリプトの同期化について

テスト対象のシステムによっては，仮想ユーザがターミナル・エミュレータに送信する入力をサーバからの応答と同期させることが必要な場合があります。入力を同期化するには，次のアクションを実行する前にスクリプトの実行を一時停止し，システムからの合図を待つように仮想ユーザを設定します。例えば，実際のユーザが次の一連のキーストロークを銀行アプリケーションに送信する場合を考えます。

1「1」と入力して，銀行アプリケーションのメニューから「金融情報」を選択します。

2「必要な情報を選択してください。」というメッセージが表示されたら，「3」と入力して，メニューから「ダウ・ジョーンズ工業平均株価」を選択します。

3 詳細なレポートが画面に表示されたら，「5」と入力して，銀行アプリケーションを終了します。

この例では，実際のユーザが各ステップで入力する前に画面上の合図を待っているので，銀行アプリケーションへの入力は同期化されています。この合図は，特定のメッセージが画面上に表示されること，または画面上のすべての情報が安定した状態になることのいずれかです。

仮想ユーザの入力は，TE 同期化関数 **TE_wait_sync**，**TE_wait_text**，**TE_wait_silent**，および **TE_wait_cursor** を使って同じ方法で同期化できます。これらの関数は，ターミナル・ウィンドウに入力する実際のユーザをエミュレートし，次のコマンドを入力する前にサーバの応答を待ちます。

TE_wait_sync 関数は，ブロック・モード（IBM）・ターミナルを同期化する場合にのみ使用されます。他の TE 同期化関数は，キャラクタ・モード（VT）・ターミナルを同期化する場合に使用されます。

RTE 仮想ユーザ・スクリプトを記録すると，**TE_wait_sync**，**TE_wait_text**，および **TE_wait_cursor** の各ステートメントが自動的に生成され，スクリプトに挿入されます。挿入される同期化関数を指定するには，VuGen の記録オプションを使用します。

注：仮想ユーザ・スクリプトの **Vuser_end** セクションには，同期化ステートメントを挿入しないでください。仮想ユーザの実行はいつでも中止できるため，**Vuser_end** セクションがいつ実行されるかは予測できません。

ブロック・モード（IBM）・ターミナルの同期化

TE_wait_sync 関数は，ブロック・モード（IBM）・ターミナルを操作する RTE 仮想ユーザを同期化するために使用されます。ブロック・モード・ターミナルでは，システムが入力禁止モードになっていることを示すために「X SYSTEM」というメッセージが表示されます。システムが入力禁止モードになっているときは，ターミナル・エミュレータがサーバからデータが転送されるのを待っているため，入力できません。

ブロック・モード・ターミナルでスクリプトを記録すると，標準設定では「X SYSTEM」メッセージが表示されるたびに **TE_wait_sync** 関数が生成され，スクリプトに挿入されます。**TE_wait_sync** 関数を自動的に挿入するかどうかを指定するには，VuGen の記録オプションを使用します。

仮想ユーザ・スクリプトを実行すると，**TE_wait_sync** 関数はシステムが X SYSTEM モードになっているかどうかを確認します。システムが X SYSTEM モードになっている場合，**TE_wait_sync** 関数はスクリプトの実行を一時停止します。「X SYSTEM」メッセージが画面から削除されると，スクリプトの実行が再開されます。

注：TE_wait_sync 関数は，IBM ブロック・モード・ターミナル（5250 および 3270）エミュレータでのみ使用できます。

TE_wait_sync 関数は，すべてのブロック・モード・ターミナル・エミュレータに対して適切な同期化機能を提供します。ただし，特定の状況で **TE_wait_sync** 関数がうまく機能しない場合は，**TE_wait_text** 関数を挿入することで同期化機能を拡張できます。**TE_wait_text** 関数の詳細については，727 ページ「画面上のテキスト表示の待機」，および「**オンライン関数リファレンス**」（[**ヘルプ**] ＞［**関数リファレンス**]）参照してください。

TE_wait_sync 関数の構文は次のとおりです。

```
TE_wait_sync();
```

次のスクリプト・セグメントでは，仮想ユーザが「QUSER」というユーザ名と「MERCURY」というパスワードでログオンします。仮想ユーザは，Enter を押してサーバにログイン情報を送信します。サーバから応答を待っている間は，ターミナル・エミュレータに X SYSTEM メッセージが表示されます。

TE_wait_sync ステートメントによって，仮想ユーザはスクリプトの次の行を実行する前に，サーバがログイン要求に応答するまで（X SYSTEM メッセージが削除されるまで）待機します。

```
TE_type("QUSER");
lr_think_time(2);
TE_type("<kTab>MERCURY");
lr_think_time(3);
TE_type("<kEnter>");
TE_wait_sync();
....
```

X SYSTEM メッセージの表示に合わせて **TE_wait_sync** 関数がスクリプトの実行を一時停止している間，仮想ユーザはシステムを監視し続け，X SYSTEM

メッセージが消えるのを待ちます。X SYSTEM メッセージが消えないまま同期化のタイムアウトに達すると，**TE_wait_sync** 関数はエラー・コードを返します。標準のタイムアウトは 60 秒です。

TE_wait_sync による同期化のタイムアウトを設定するには，次の手順を実行します。

1［**仮想ユーザ**］>［**実行環境の設定**］を選択します。［実行環境設定］ダイアログ・ボックスが表示されます。

2［実行環境設定］ダイアログ・ボックスの［**RTE：RTE**］ノードを選択します。

3［**X- システムの同期化**］の［**タイムアウト**］ボックスに，値（秒単位）を入力します。

4［**OK**］をクリックして，［実行環境設定］ダイアログ・ボックスを閉じます。

仮想ユーザは，**TE_wait_sync** 関数を実行した後，ターミナルが X SYSTEM モードに入らなくなるまで待機します。ターミナルが X SYSTEM モードから戻ると，仮想ユーザはターミナルが完全に安定した（システムが X SYSTEM に戻らない）ことを確認するため，少しの間システムを監視し続けます。この確認ができた場合にのみ，**TE_wait_sync** 関数が終了し，仮想ユーザがスクリプトの実行を再開できるようになります。仮想ユーザが X SYSTEM モードから戻った後のシステムを監視し続ける時間を，安定時間と呼びます。標準の安定時間は 1000 ミリ秒です。

システムに次のような動作が見られる場合は，安定時間を増やす必要があります。

システムが X SYSTEM モードから戻るときに，一部のシステムでは X SYSTEM モードとの間を短時間に何度か「行き来」してからでないとシステムが安定しません。システムが 1 秒以上 X SYSTEM モードにならず，その後で X SYSTEM モードに戻った場合，**TE_wait_sync** 関数はシステムが安定したとみなします。仮想ユーザがシステムに情報を入力しようとすると，システムはキーボード・ロック・モードに移行します。

逆に，システムが X SYSTEM から戻るときにモード間の行き来がない場合は，安定時間を標準値の 1 秒より短く設定できます。

TE_wait_sync 関数の安定時間を変更するには，次の手順を実行します。

1［**仮想ユーザ**］>［**実行環境の設定**］を選択します。［実行環境設定］ダイアログ・ボックスが表示されます。

2［**RTE：RTE**］ノードを選択します。

3［**X- システムの同期化**］の［**安定時間**］ボックスに，値（ミリ秒単位）を入力します。

4［**OK**］をクリックして，［実行環境設定］ダイアログ・ボックスを閉じます。

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

システムが X SYSTEM モードに移行するたびにその継続時間を記録するように VuGen を設定できます。その場合は，次のスクリプト・セグメントに示すように，各 **TE_wait_sync** 関数の後に **TE_wait_sync_transaction** 関数が挿入されます。

```
TE_wait_sync();
TE_wait_sync_transaction("syncTrans1");
```

TE_wait_sync_transaction 関数は，「default」という名前のトランザクションを作成します。これによって，ターミナル・エミュレータがチューニング・セッションやシナリオ実行の間にサーバからの応答を待った時間を分析できます。**TE_wait_sync_transaction** ステートメントを生成して挿入するかどうかを指定するには，記録オプションを使用します。

TE_wait_sync_transaction ステートメントを挿入するように VuGen を設定するには，次の手順を実行します。

1［**仮想ユーザ**］＞［**記録オプション**］を選択します。［記録オプション］ダイアログ・ボックスが表示されます。

2［**X- システム用自動トランザクションを作成する**］オプションを選択して，［**OK**］をクリックします。

キャラクタ・モード（VT）・ターミナルの同期化

キャラクタ・モード（VT）・ターミナルでは，3 種類の同期化を使用できます。選択できる同期化の種類は，次の要因で決まります。

- ターミナル・エミュレータで実行するアプリケーションの設計
- 同期化する具体的なアクション

特定位置でのカーソル表示の待機

VT タイプのターミナルで推奨される同期化方法は，カーソル同期化です。カーソル同期化は，スクロール形式や TTY 形式のアプリケーションよりも，全画面形式やフォーム形式のアプリケーションで特に有効です。

カーソル同期化では，**TE_wait_cursor** 関数を使用します。RTE 仮想ユーザ・スクリプトを実行すると，**TE_wait_cursor** 関数は画面上の指定された位置にカーソルが表示されるまでスクリプトの実行を一時停止するように仮想ユーザに指示します。指定された位置にカーソルが表示されることは，アプリケーションがターミナル・エミュレータから次の入力を受け付ける準備ができたことを意味します。

TE_wait_cursor 関数の構文は次のとおりです。

int **TE_wait_cursor (**int **col**, int **row**, int **stable**, int **timeout);**

スクリプトの実行中，**TE_wait_cursor** 関数はカーソルが **col** と **row** で指定された位置に移動するまで待機します。

stable パラメータには，カーソルが指定された位置で安定している時間（ミリ秒）を指定します。VuGen を使ってスクリプトを記録する場合，**stable** は標準で 100 ミリ秒です。クライアント・アプリケーションが **timeout** パラメータで指定された時間内に安定しない場合，この関数は TIMEOUT を返します。VuGen を使ってスクリプトを記録する場合，**timeout** は標準で TIMEOUT の値（90 秒）に設定されます。**stable** パラメータの値と **timeout** パラメータの値は，どちらも記録されたスクリプトを直接編集することによって変更できます。

次のステートメントは，カーソルが安定した状態で 3 秒間待機します。カーソルが 10 秒以内に安定しない場合，この関数は TIMEOUT を返します。

```
TE_wait_cursor(10, 24, 3000, 10);
```

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

スクリプトの記録中に **TE_wait_cursor** ステートメントを自動的に生成してスクリプトに挿入するように VuGen を設定できます。VuGen によって自動的に生成された **TE_wait_cursor** ステートメントの例を次に示します。

```
TE_wait_cursor(7, 20, 100, 90);
```

記録中に TE_wait_cursor ステートメントを自動的に生成してスクリプトに挿入するように VuGen を設定するには，次の手順を実行します。

1 ［**仮想ユーザ**］＞［**記録オプション**］を選択します。［記録オプション］ダイアログ・ボックスが表示されます。

2 ［**自動同期化コマンドを作成**］の［**カーソル**］チェック・ボックスを選択して，［**OK**］をクリックします。

画面上のテキスト表示の待機

VT ターミナル・エミュレータ上で RTE 仮想ユーザを同期化する場合は，テキスト同期化を使用します。テキスト同期化では，**TE_wait_text** 関数を使用します。スクリプトの実行中，**TE_wait_text** 関数はスクリプトの実行を一時停止し，スクリプトの実行を再開する前に特定の文字列がターミナル・ウィンドウに表示されるのを待ちます。テキスト同期化は，カーソルが画面上のあらかじめ定義された領域に常に表示されるとは限らないアプリケーションで有効です。

注 : テキスト同期化は，キャラクタ・モード（VT）・ターミナルでの使用を目的としていますが，IBM ブロック・モード・ターミナルでも使用できます。ただし，ブロック・モード・ターミナルでは自動的なテキスト同期化を使用しないでください。

TE_wait_text 関数の構文は次のとおりです。

int **TE_wait_text (**char ***pattern**, int **timeout**, int **col1**, int **row1**, int **col2**, int **row2**, int ***retcol**, int ***retrow**, char ***match);**

この関数は，col1，row1，col2，row2 で定義される四角形の内部に **pattern** に一致するテキストが表示されるまで待機します。パターンに一致するテキストは **match** に返され，実際の行位置と列位置は **retcol** と **retrow** に返されます。**pattern** が表示されないまま **timeout** に達すると，この関数はエラー・コードを返します。**pattern** には正規表現を含めることができます。正規表現の使用方法の詳細については，「**オンライン関数リファレンス**」を参照してください。**pattern** と **timeout** 以外のパラメータは，すべて省略可能です。

pattern に空の文字列を指定すると，この関数は四角形の内部に任意のテキストが表示されたときにタイムアウトを待ちます。テキストが表示されなかったときは，すぐに戻ります。

pattern が表示されなかった場合，この関数はエミュレータが安定する（再描画を終了する，または新しい文字を表示しなくなる）のを，TE_SILENT_SEC システム変数と TE_SILENT_MILLI システム変数で定義された時間だけ待ちます。これは，結果的にターミナルを安定させ，実際のユーザをエミュレートすることになります。

ターミナルが TE_SILENT_TIMEOUT で定義された時間内に安定しない場合は，スクリプトの実行が再開されます。この関数は成功を示す 0 を返すと同時に，テキストの表示後にターミナルが安定しなかったことを示すために TE_errno 変数を設定します。

TE_SILENT システム変数の値を変更および取得するには，**TE_getvar** 関数および **TE_setvar** 関数を使用します。詳細については，「**オンライン関数リファレンス**」（［**ヘルプ**］＞［**関数リファレンス**］）を参照してください。

次の例では，仮想ユーザが自分の名前を入力し，アプリケーションの応答を待ちます。

```
/* TE_wait_text の変数を宣言する */
int ret_row;
int ret_col;
char ret_text [80];

/* ユーザ名を入力する */
TE_type ("John");

/* 窓口の応答を待つ */
TE_wait_text ("Enter secret code:", 30, 29, 13, 1, 13, &ret_col, &ret_row,
ret_text);
```

スクリプトの記録中に **TE_wait_text** ステートメントを自動的に生成してスクリプトに挿入するように VuGen を設定できます。

記録中に TE_wait_text ステートメントを自動的に生成してスクリプトに挿入するように VuGen を設定するには，次の手順を実行します。

1 ［**仮想ユーザ**］＞［**記録オプション**］を選択します。［記録オプション］ダイアログ・ボックスが表示されます。

2 ［**自動同期化コマンドを作成**］の［**プロンプト**］チェック・ボックスを選択して，［**OK**］をクリックします。

VuGen によって自動的に生成された **TE_wait_text** ステートメントの例を次に示します。この関数は，「keys」という文字列が画面上の任意の場所に表示されるのを最大 20 秒間待ちます。VuGen が **TE_wait_text** 関数を生成するときは，省略可能なパラメータはすべて省略されます。

```
TE_wait_text("keys", 20);
```

ターミナルの安定の待機

カーソル同期化もテキスト同期化もうまく機能しない場合は，「安定同期化」を使用してスクリプトを同期化できます。「安定同期化」では，仮想ユーザはターミナル・エミュレータが安定するのを指定された時間だけ待ちます。エミュレータは，指定された期間内にサーバからの入力を受信しなかったときに安定したとみなされます。

注：安定同期化は，カーソル同期化とテキスト同期化がどちらもうまく機能しない場合にのみ使用します。

ターミナルが安定するまで待機するようにスクリプトを設定するには，**TE_wait_silent** 関数を使用します。ターミナルが安定している期間を指定します。ターミナルが指定された期間だけ安定していれば，**TE_wait_silent** 関数は，アプリケーションがターミナル画面へのテキストの表示を終了し，画面が安定したとみなします。

この関数の構文は次のとおりです。

int **TE_wait_silent (**int **sec**, int **milli**, int **timeout);**

TE_wait_silent 関数は，ターミナル・エミュレータが安定するのを **sec**（秒）と **milli**（ミリ秒）で指定された期間だけ待ちます。エミュレータは，サーバからの入力を受信しなかったときに安定したとみなされます。**timeout** で指定された時間内にエミュレータが安定しない（文字の受信が終了しない）場合，この関数はエラーを返します。

例えば，次のステートメントは画面が安定した状態で 3 秒間待機します。10 秒経過しても画面が安定しない場合，この関数はエラーを返します。

```
TE_wait_silent (3, 0, 10);
```

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

第 50 章

ターミナル画面からのテキストの読み取り

RTE 仮想ユーザは，ターミナル・エミュレータのユーザ・インタフェースからテキストを読み取り，そのテキストを使ってさまざまなタスクを実行できます。

本章では，次の項目について説明します。

- ターミナル画面からのテキストの読み取りについて
- 画面上のテキストの検索
- 画面からのテキストの読み取り

以降の情報は，RTE（Windows）仮想ユーザ・スクリプトを対象とします。

ターミナル画面からのテキストの読み取りについて

RTE 仮想ユーザがターミナル画面からテキストを読み取るために使用できる仮想ユーザ関数が複数用意されています。これらの関数は **TE_find_text** と **TE_get_text_line** で，ターミナル・エミュレータが正常に応答していることを確認するため，またはスクリプトのロジックを拡張するために使用できます。

TE_find_text と **TE_get_text_line** は，記録後，RTE 仮想ユーザ・スクリプトに手作業で直接挿入します。

画面上のテキストの検索

TE_find_text 関数は，画面上のテキスト行を検索します。この関数の構文は次のとおりです。

int **TE_find_text (**char ***pattern**, int **col1**, int **row1**, int **col2**, int **row2**,int ***retcol**, int ***retrow**, char ***match);**

この関数は，**col1**，**row1**，**col2**，**row2** で定義される四角形の内部で，**pattern** に一致するテキストを検索します。パターンに一致するテキストは **match** に返され，実際の行位置と列位置は **retcol** と **retrow** に返されます。検索は左上隅から行われます。複数の文字列が **pattern** に一致した場合は，左上隅に最も近い文字列が返されます。

pattern には正規表現を含めることができます。正規表現の使用方法の詳細については，「**オンライン関数リファレンス**」を参照してください。

TE_find_text ステートメントは，仮想ユーザ・スクリプトに手作業で入力する必要があります。**TE_find_text** 関数の構文の詳細については，「**オンライン関数リファレンス**」（[**ヘルプ**] > [**関数リファレンス**]）を参照してください。

画面からのテキストの読み取り

TE_get_text_line 関数は，画面上の指定された領域からテキスト行を読み取ります。この関数の構文は次のとおりです。

char ***TE_get_text_line (**int **col**, int **row,** int **width**, char ***text);**

この関数は，テキスト行をターミナル画面から **text** バッファにコピーします。行の最初の文字は **col** と **row** で定義します。行の最後の文字の列座標は **width** で定義します。画面から読み取られたテキストは，**text** バッファに返されます。行内にタブや空白が含まれる場合は，それらに相当する数の空白が返されます。

また，**TE_get_cursor_position** 関数を使ってターミナル画面上のカーソルの現在位置を取得することもできます。**TE_get_line_attribute** 関数は，テキスト行内の文字の書式設定（太字，下線など）を返します。

TE_get_text_line ステートメントは，仮想ユーザ・スクリプトに手作業で入力する必要があります。**TE_get_text_line** 関数の構文の詳細については，「**オンライン関数リファレンス**」（[**ヘルプ**] > [**関数リファレンス**]）を参照してください。

第 11 部

メール・サービス・プロトコル

第 51 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，以下の項目について説明します。

- メール・サービス仮想ユーザ・スクリプトの作成について
- 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 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「スクリプト生成オプションの設定」を参照してください。通信に複数のプロトコルが使われている場合は，両方を記録できます。複数のメール・プロトコルを同時に，または 1 つのメール・プロトコルを HTTP または WinSock と一緒に記録できます。マルチ・プロトコルの指定の詳細については，『**第 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「VuGen を使った記録」を参照してください。

メール・サービス関数はすべて，対で用意されています。一方がグローバル・セッション用で，もう一方で特定のメール・セッションを指定します。例えば，**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・スクリプトの作成プロセスの概略を説明します。

メール・サービス仮想ユーザ・スクリプトを作成するには，次の手順を実行します。

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 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「VuGen を使った記録」を参照してください。

MAPI では，記録はサポートされていません。その代わりに空の MAPI スクリプトを作成し，**mapi** 関数を手作業で挿入します。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 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「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 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「VuGen パラメータを使った作業」を参照してください。

5 **ステートメントを相関させます（任意）。**

ステートメントの相関によって，あるビジネス・プロセスの結果を以降の別のビジネス・プロセスで使用できるようになります。

詳細については，『**第 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「ステートメントの相関」を参照してください。

6 **実行環境を設定します。**

実行環境の設定では，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 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「実行環境の設定」を参照してください。

7 **VuGen でスクリプトを実行します。**

VuGen でスクリプトを保存し実行して，正しく動作することを確認します。

詳細については，『**第 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「スタンドアロン・モードでの仮想ユーザ・スクリプトの実行」を参照してください。

スクリプトを作成した後，スクリプトを自分の環境（LoadRunner シナリオ，Performance Center 負荷テスト，Tuning Module セッション，またはビジネス・プロセス・モニタ・プロファイル）に統合します。詳細については，『**Mercury LoadRunner コントローラ・ユーザーズ・ガイド**』，**Tuning Console**，**Performance Center**，または **Application Management** のマニュアルを参照してください。

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_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 52 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 Java 仮想ユーザ API 関数を使って拡張したり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。

- Jacada 仮想ユーザ・スクリプトについて
- 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・スクリプトについて

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 22 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，スクリプトを環境（LoadRunner シナリオ，Performance Center 負荷テスト，Tuning Module セッション，またはビジネス・プロセス・モニタ・プロファイル）に統合します。詳細については，『**Mercury LoadRunner コントローラ・ユーザーズ・ガイド**』，または **Tuning Console**，**Performance Center**，**Application Management** のマニュアルを参照してください。

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 14 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 14 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 3 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。**

仮想ユーザ API 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 14 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 39 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 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「VuGen パラメータを使った作業」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 5 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。**

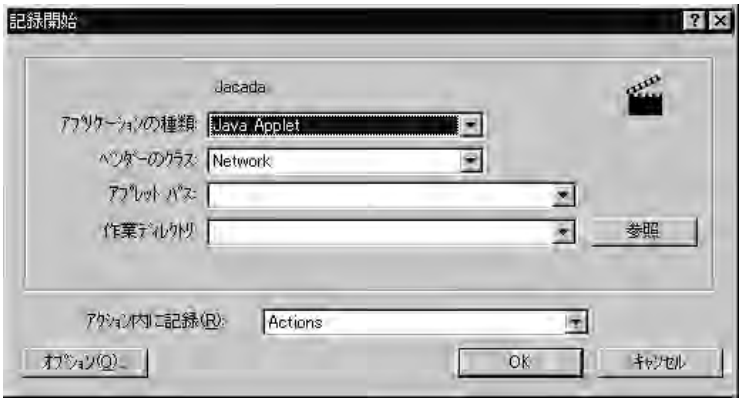
VuGen から スクリプトを実行して，実行時の情報を実行ログで確認します。詳細については，『**第 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「スタンドアロン・モードでの仮想ユーザ・スクリプトの実行」を参照してください。

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，次の手順を実行します。

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 3 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

仮想ユーザ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，Mercury 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 14 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，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 または仮想ユーザ API 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 Java 仮想ユーザ API 関数を追加できます。これらの関数の一覧とスクリプトへの追加方法については，第 14 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 53 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，以下の項目について説明します。

- Tuxedo 仮想ユーザ・スクリプトについて
- 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。例えば，**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。例えば，**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。例えば，**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。例えば，**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・スクリプトの作成プロセスの概略を説明します。

Tuxedo 仮想ユーザ・スクリプトを作成するには，次の手順を実行します。

1 **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 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「VuGen を使った記録」を参照してください。

2 **スクリプトを拡張します。**

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

詳細については，『**第 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「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 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「VuGen パラメータを使った作業」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 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「ステートメントの相関」を参照してください。

5 **実行環境を設定します。**

実行環境を設定することによって，スクリプト実行時の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 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「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 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「スタンドアロン・モードでの仮想ユーザ・スクリプトの実行」を参照してください。

スクリプトを作成した後，スクリプトを自分の環境（LoadRunner シナリオ，Performance Center 負荷テスト，Tuning Module セッション，またはビジネス・プロセス・モニタ・プロファイル）に統合します。詳細については，『**Mercury LoadRunner コントローラ・ユーザーズ・ガイド**』，**Tuning Console**，**Performance Center**，または **Application Management** のマニュアルを参照してください。

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_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・ウィンドウにスクリプトを表示すると，VuGen が動作状況を記録したシーケンスを確認できます。

次の例では，VuGen がクライアントのアクションを Tuxedo の銀行アプリケーションに記録しています。クライアントは，銀行の口座を開き，必要な詳細のすべてを指定するアクションを実行しました。クライアントが開始残高としてゼロを指定したところで，セッションは中止されています。

```
lrt_abort_on_error();
lr_think_time(65);
tpresult_int = lrt_tpbegin(30, 0);
data_0 = lrt_tpalloc("FML", "", 512);
lrt_Finitialize((FBFR*)data_0);

/* データ・バッファ data_0 に新規口座情報を入力する */
lrt_Fadd_fld((FBFR*)data_0, "name=BRANCH_ID", "value=8",
LRT_END_OF_PARMS);
lrt_Fadd_fld((FBFR*)data_0, "name=ACCT_TYPE", "value=C",
LRT_END_OF_PARMS);
lrt_Fadd_fld((FBFR*)data_0, "name=MID_INIT", "value=Q",
LRT_END_OF_PARMS);
lrt_Fadd_fld((FBFR*)data_0,"name=PHONE","value=123-456-7890",
            LRT_END_OF_PARMS);

lrt_Fadd_fld((FBFR*)data_0, "name=ADDRESS", "value=1 Broadway
            New York, NY 10000", LRT_END_OF_PARMS);
lrt_Fadd_fld((FBFR*)data_0,"name=SSN","value=111111111",
LRT_END_OF_PARMS);

lrt_Fadd_fld((FBFR*)data_0,"name=LAST_NAME",
"value=Doe",LRT_END_OF_PARMS);

lrt_Fadd_fld((FBFR*)data_0,"name=FIRST_NAME",
"value=BJ",LRT_END_OF_PARMS);

lrt_Fadd_fld((FBFR*)data_0, "name=SAMOUNT",
"value=0.00",LRT_END_OF_PARMS);
```

```
/* 新規口座を開く */
tpresult_int = lrt_tpcall("OPEN_ACCT", data_0, 0, &data_0, &olen_2, 0);
lrt_tpabort(0);
lrt_tpcommit(0);
lrt_tpfree(data_0);
lrt_tpterm();
```

Tuxedo スクリプトでのパラメータの使用

『**第 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「VuGen パラメータを使った作業」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。例えば，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，「Tuxedo バッファ・データの表示」次項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

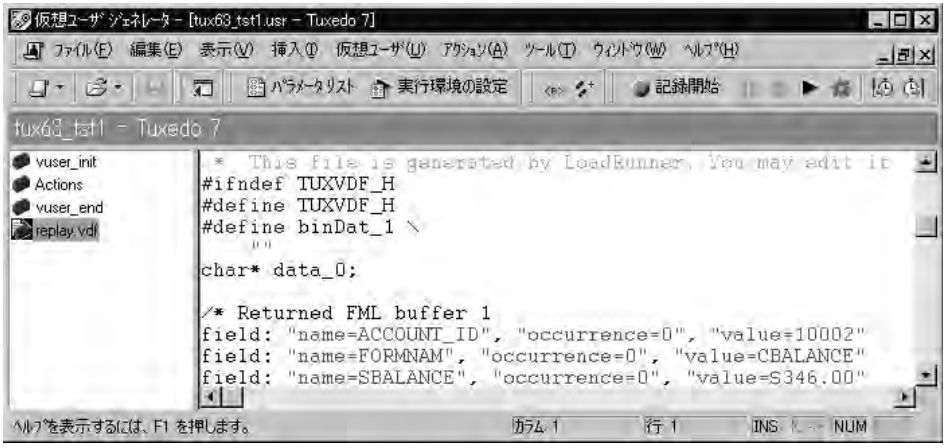
VuGen を使用して Tuxedo 仮想ユーザ・スクリプトを作成する場合，アクションはスクリプトの 3 つのセクション，**vuser_init**，**Actions**，および **vuser_end** に記録されます。

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。

例えば，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。

例えば，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を相関するには，次の手順を実行します。

1 スクリプト内の，現在の FML（または FML32）バッファの内容を保存する場所に，**lrt_save_fld_val** ステートメントを挿入します。

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

2 パラメータを参照します。

保存したバッファの内容で記録されている値を置き換える **lrt** ステートメントを見つけます。記録されている値のすべてのインスタンスを，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 で相関するには，次の手順を実行します。

```
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

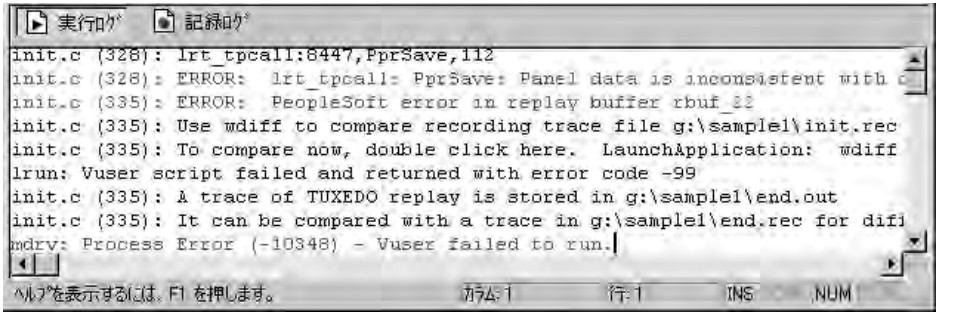
CARRAY バッファで作業を行っている場合，VuGen は Wdiff ユーティリティを使って比較できるログ・ファイルを生成します（記録中であれば拡張子は **.rec** に，再生中であれば拡張子は **.out** になります）。記録ログと再生ログの相違を調べて，CARRAY バッファのどの部分を相関させる必要があるか判断できます。

ログ・ファイルを比較するには，次の手順を実行します。

1 ［**表示**］＞［**出力ウィンドウ**］を選択して，スクリプトの実行ログと記録ログを表示します。

2 ［再生ログ］タブを検査します。

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 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「ステートメントの相関」を参照してください。

lrt_save_parm を使用したステートメントを相関するには，次の手順を実行します。

相関させる値を決定したら，**lrt_save_parm** を使って，文字配列の一部分（STRING または CARRAY バッファなど）をパラメータに保存できます。

1 スクリプト内の，現在のバッファの内容を保存する場所に **lrt_save_parm** ステートメントを挿入します。

lrt_save_parm (buffer, offset, length, "**param_name**");

2 **replay.vdf** ファイル内で，保存されたバッファの内容で置き換えたいデータを見つけます。

VuGen のメイン・ウィンドウの［データ ファイル］ボックスにある **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を相関するには，次の手順を実行します。

lrt_save_searched_string を使用して，バッファ内で文字列を検索し，文字列に関連するバッファの一部をパラメータに保存します。

1 スクリプト内の現在のバッファの一部を保存する場所に **lrt_save_searched_string** ステートメントを挿入します。

lrt_save_searched_string (buffer, buf_size, occurrence, string, offset,length, "**param_name**");

offset は文字列の先頭からのオフセットです。

2 **replay.vdf** ファイル内で，保存されたバッファの内容で置き換えたいデータを見つけます。

VuGen のメイン・ウィンドウの［データ ファイル］ボックスにある **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「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 54 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 は，ストリーミング・コンテンツを表示するアプリケーションです。

VuGen を使って，RealPlayer プロトコルでやり取りするクライアント・アプリケーションとサーバ間の通信を記録することができます。記録の結果として作成されるスクリプトを，「Real 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・スクリプトの記録について
- 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

以降の情報は，Real 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。

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 を使って RealPlayer ストリーミング・データ仮想ユーザ・スクリプトの作成プロセスの概略を説明します。

RealPlayer 仮想ユーザ・スクリプトを作成するには，次の手順を実行します。

1 **VuGen を使用して，基本スクリプトを記録します。**

VuGen を起動して，新しい Real 仮想ユーザ・スクリプトを作成します。記録するアプリケーションを選択し，アプリケーションを対象とする一般的な操作を記録します。詳細については，『**第 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「VuGen を使った記録」を参照してください。

2 **スクリプトを拡張します。**

トランザクション，ランデブー・ポイント，および制御フロー構造をスクリプトに挿入することによって，スクリプトを拡張します。

詳細については，『**第 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「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 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「VuGen パラメータを使った作業」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 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「ステートメントの相関」を参照してください。

5 **実行環境を設定します。**

実行環境を設定することによって，スクリプト実行中の仮想ユーザの動作を制御します。設定には，ループ，ログ，タイミング情報などがあります。

詳細については，『**第 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「実行環境の設定」を参照してください。

6 **VuGen でスクリプトを実行します。**

VuGen でスクリプトを保存して実行し，正しく動作することを確認します。

詳細については，『**第 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「スタンドアロン・モードでの仮想ユーザ・スクリプトの実行」を参照してください。

スクリプトを作成した後，スクリプトを自分の環境（LoadRunner シナリオ，Performance Center 負荷テスト，Tuning Module セッション，またはビジネス・プロセス・モニタ・プロファイル）に統合します。詳細については，『**Mercury LoadRunner コントローラ・ユーザーズ・ガイド**』，**Tuning Console**，**Performance Center**，または **Application Management** のマニュアルを参照してください。

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。

第 14 部

ワイアレス・プロトコル

第 55 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，以下の項目について説明します。

- ワイヤレス仮想ユーザについて
- 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。

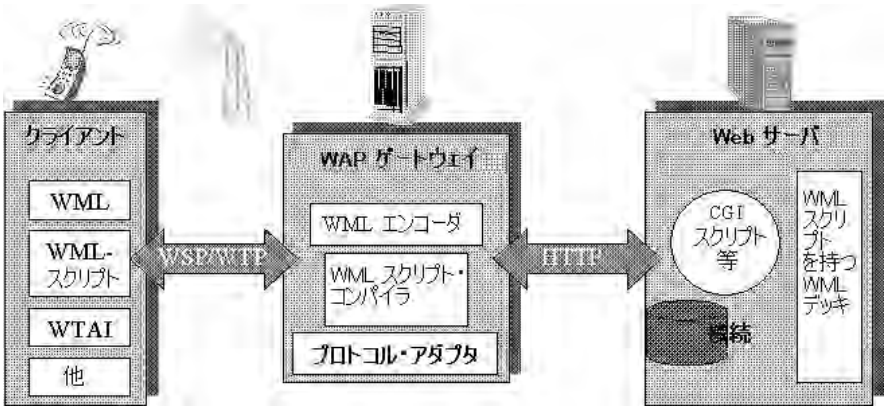
WAP ではさらに，以下の条件を満たすプロキシ・サーバを規定しています。

- 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，794 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。例えば，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。例えば，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。例えば，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 56 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，以下の項目について説明します。

- ワイヤレス仮想ユーザ・スクリプトの記録について
- ワイヤレス仮想ユーザ・スクリプトの作成の概要
- 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またはセッション・ステップ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・スクリプトの作成の概要

本項では，VuGen を使ったワイヤレス仮想ユーザ・スクリプトの作成プロセスの概略を説明します。

ワイヤレス・スクリプトを作成するには，次の手順を実行します。

1 **VuGen を使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 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「VuGen を使った記録」を参照してください。

2 **記録オプションを設定します。**

記録オプションを設定します。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 24 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 58 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 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「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 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「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 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「VuGen パラメータを使った作業」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 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「実行環境の設定」を参照してください。

インターネット・プロトコルの一般的な実行環境設定の詳細については，第 26 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 59 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，422 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 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「スタンドアロン・モードでの仮想ユーザ・スクリプトの実行」を参照してください。

スクリプトを作成した後，スクリプトを自分の環境（LoadRunner シナリオ，Performance Center 負荷テスト，Tuning Module セッション，またはビジネス・

プロセス・モニタ・プロファイル）に統合します。詳細については，『**Mercury LoadRunner コントローラ・ユーザーズ・ガイド**』，**Tuning Console**，**Performance Center**，または **Application Management** のマニュアルを参照してください。

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，仮想ユーザ・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 1 巻 - 仮想ユーザ・ジェネレータの使い方**』の「VuGen の紹介」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，34 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 23 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**

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，次の手順を実行します。

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 57 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，以下の項目について説明します。

- WAP 仮想ユーザについて
- 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
- VuGen でのプッシュのサポート

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 55 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，794 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 58 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 58 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。ベアラには例えば，SMS-CIMD2，UDP, CSD, GSM GPRS，GSM CSD，Packet Data などがあります。

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［**Bearers**］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［**Bearers**］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［**Bearers**］ノードで CIMD2 ベアラ情報を設定します（第 59 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。例えば，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 59 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 は基本的なプロキシ・ゲートウェイ・サービスを提供するほか，プッシュ・イニシエータにプッシュ操作の最終ステータスを通知することができます。また，双方向のモバイル・ネットワークにおいては，クライアントがコンテンツを受け入れるか拒否するまで待機することができます。

プッシュ・サービスのタイプ

プッシュ・サービスのタイプには，SL と SI があります。

- **SL** ― サービス・ロード（SL）コンテンツ・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。例えば，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。例えば，新規メールの到着，株価の変動，ニュースのヘッドライン，広告などの通知が考えられます。

  最も基本的な形式の SI には，ショート・メッセージとサービスを示す URI が含まれています。メッセージは，エンド・ユーザの受信時に提示され，ユーザは URI が示すサービスをすぐに開始するか，後で処理するために SI を延期するかを選択できます。SI を延期すると，クライアントによってサービスが保存され，エンド・ユーザは後でそのサービスを開始できます。

VuGen でのプッシュのサポート

VuGen でのプッシュのサポートは，次の 3 つに分類できます。

- ➤ クライアント側でのプッシュのサポート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，仮想ユーザによってメッセージが解析され，タイプの識別と属性の取得が行われます。解析が正常に行われると，プル・トランザクションが生成された後に実行され，該当データが取得されます。［実行環境設定］でプル・イベントを無効にすると，仮想ユーザはメッセージを取得しません。詳細については，第 59 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

WAP HTTP 仮想ユーザのプッシュ機能のサポートにより，PPG の負荷テストが可能です。プッシュのサポートにより，仮想ユーザは，**Push Access Protocol**（PAP）をサポートするプッシュ・イニシエータとして機能できます。PAP では，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，仮想ユーザ・ジェネレータ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。

第 58 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，以下の項目について説明します。

- ワイヤレス記録オプションの設定について
- 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

ワイヤレス記録オプションの設定について

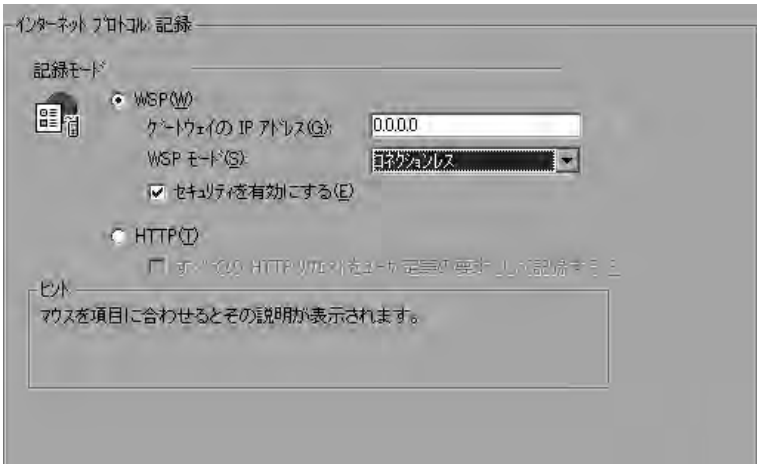
VuGen では，ユーザがワイヤレス・インタフェースを利用して Web サイトで実行する標準的なプロセスを記録することによって，ワイヤレス仮想ユーザ・スクリプトを生成できます。

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 24 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，次の手順を実行します。

［**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［**コネクションレス**］または［**コネクション指向**］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。

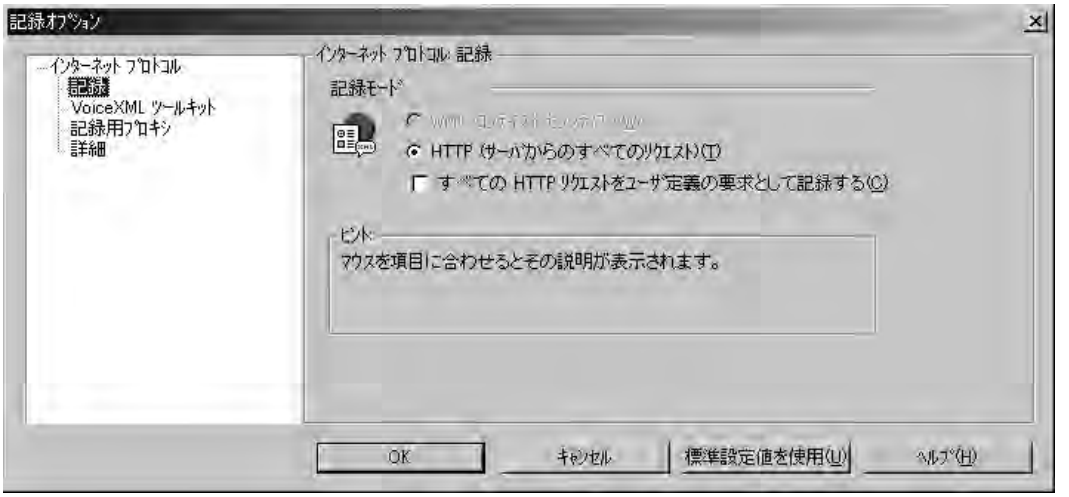
WSP モードで記録できるように，VuGen では Phone.com UP Simulator 4.1 ツールキットをネイティブでサポートとしています。サポート機能は，ツールキットがインストールされていることを検出すると，設定パラメータを自動的に設定した後，ツールキットを起動します。Nokia ツールキット（1.3 および 2.0）の場合には，正しい IP アドレスを指定して手作業で起動する必要があります。詳細については，796 ページ「ワイヤレス仮想ユーザ・スクリプトのトラブルシューティング」を参照してください。

➤［**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，次の手順を実行します。

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。

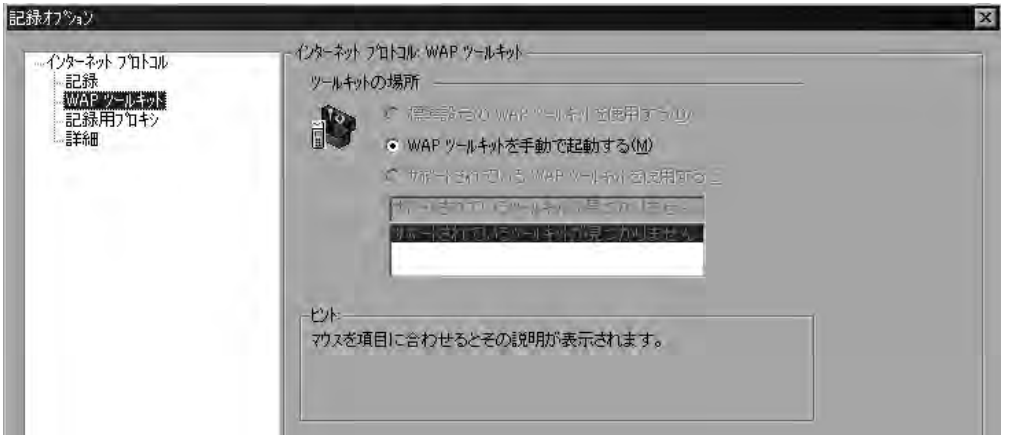
どの記録モードを選択するかは，必要性と環境に応じて決めます。利用可能なモードは，「HTTP」と「ユーザ定義要求」です。

［**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，WAP VoiceXML 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，次の手順を実行します。

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，次の手順を実行します。

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，次の手順を実行します。

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 59 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，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。

- WAP 実行環境の設定について
- 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 24 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。

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。

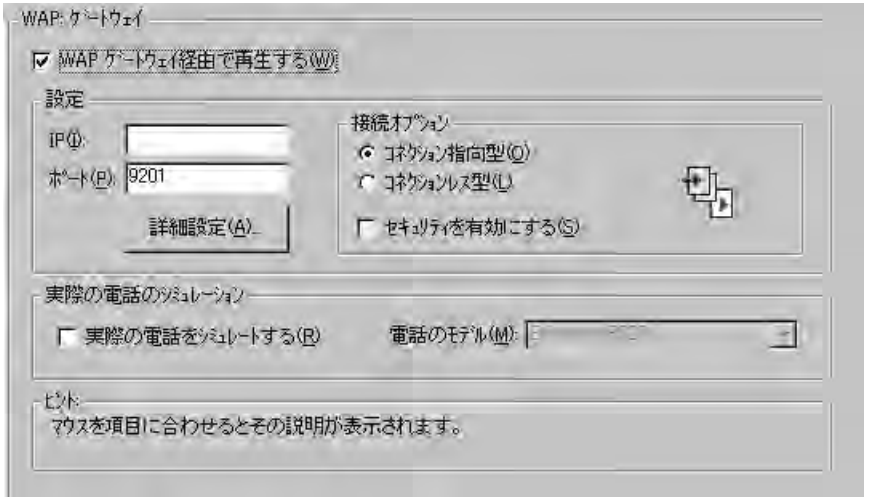
WAP の実行環境の設定は，［実行環境設定］ダイアログ・ボックスで行います。各ノードをクリックして，設定項目を表示し，必要な設定を行います。

［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**］ボタンをクリックします。

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 26 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［**WAP：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**］オプションを選択した）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は，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する（［**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た）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は，選択したモードに応じて標準のポート番号が自動的に設定されます。ただし，設定をカスタマイズして，ユーザ定義の IP アドレスとポートを指定することもできます。

［**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，クライアントの機能とゲートウェイのその他の詳細オプションを設定できます。

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。例えば，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，アプリケーションの **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。このダイアログ・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ーを返します（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**］：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**］：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）。
- ［**WTP Segmentation and Reassembly**］：WTP（Wireless Transport Protocol）による分割と再組み立て（SAR）を有効にします（標準設定では有効）。
- ［**WTP Retransmission Time**］：WTP レイヤが応答の受信に失敗して PDU を再送信する前に待機する秒数（標準設定では 5000 秒）。
- ［**WTLS Abbreviated Handshake**］：リダイレクト・メッセージの受信時に，完全なハンドシェイクではなく略式のハンドシェイクを使用します（標準設定では無効）。
- ［**WTLS Deffie Hellman**］：WTLS（Wireless Transport Layer Security）で，標準の暗号化方式である RSA ではなく Deffie-Hellman 暗号化方式を使用します（標準設定では無効）。
- ［**WTLS Deffie Hellman identifier**］：Deffie-Hellman 暗号化方式の識別子。この識別子は，Deffie-Hellman 暗号化方式を使用する Operwave ゲートウェイによる略式のハンドシェイクに必要です。
- ［**Network MTU Size**］：ネットワーク・パケットの最大バイト数（標準設定では 4096 バイト）。

ゲートウェイのオプション設定

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，次の手順を実行します。

1 ［**実行環境の設定**］ボタンをクリックするか，［**仮想ユーザ**］＞［**実行環境の設定**］を選択して，［実行環境設定］ダイアログ・ボックスを表示します。［**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択します。

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 サイズ**］および［**クライアント SDU バッファ サイズ**］の値を入力します。
- 仮想ユーザで確認応答ヘッダーを取得するよう設定するには，［**確認応答ヘッダ**］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，［**プッシュのサポート**］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，［**プッシュ サポートの確認**］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，［**メッセージ取得**］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。ベアラには例えば，SMS，UDP，CSD，GSM，GPRS，および Packet Data などがあります。

VuGen 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。

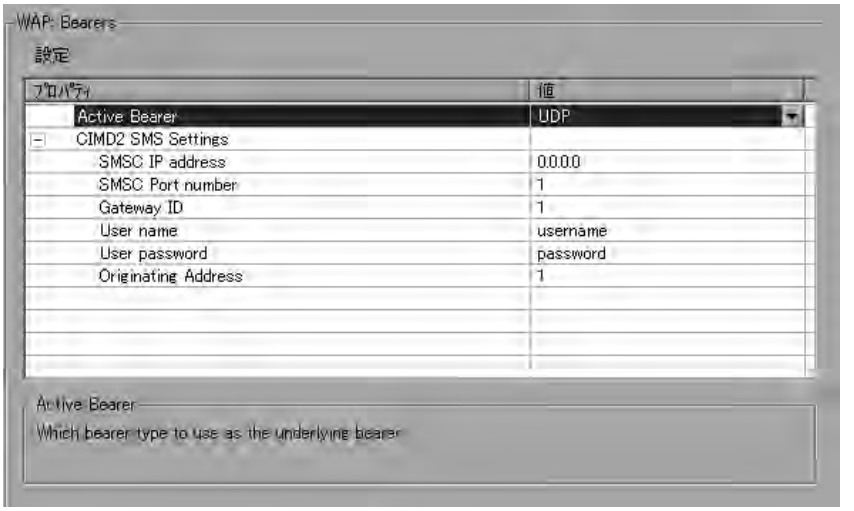
SMS-CIMD2 ベアラを使用する場合は，ショート・メッセージ・サービス・センター（SMSC）に接続し，ログイン情報を入力します。［実行環境設定］の［**WAP：Bearers**］ノードで，ポート情報を定義します。

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［**Bearers**］ノードで，ベアラの属性を設定します。

ベアラの設定	説明
Active Bearer	標準設定のベアラのタイプ（UDP または CIMD2）。
CIMD2 SMS Settings	
SMSC IP Address	SMSC サーバの IP アドレス。
SMSC Port Number	SMSC サーバのポート番号。
Gateway 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。
User name	サーバへのログイン・ユーザ名。
User Password	ユーザのパスワード。
Originating Address	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，次の手順を実行します。

1［**実行環境の設定**］ボタンをクリックするか，［**仮想ユーザ**］＞［**実行環境の設定**］を選択して，［実行環境設定］ダイアログ・ボックスを表示します。［**Bearers**］ノードを選択します。



2 アクティブ・ベアラのタイプとして **UDP** または **CIMD2** のいずれかを選択します。

3 CIMD2 ベアラについては，次の設定を行います。

- **SMSC IP アドレス**をドット区切りの形式で入力します。
- **ポート番号**を入力します。
- **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）。
- **SMSC ユーザ名**を入力します。
- **SMSC ユーザ・パスワード**を入力します。
- **SMSC 発信アドレス**を入力します。

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。

RADIUS 接続データの設定

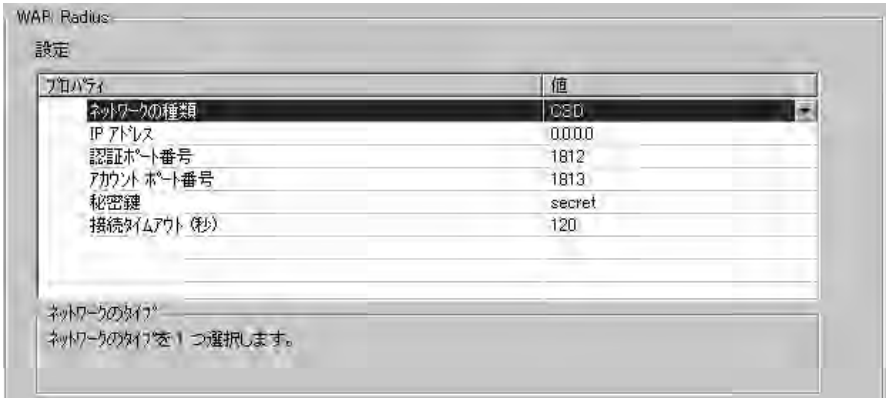
RADIUS（Remote Authentication Dial-In User Service）は，クライアント / サーバのプロトコルとソフトウェアの組み合わせで，リモート・アクセス・サーバと中央サーバとの通信によって，ダイアルアップ・ユーザの認証と，そうしたユーザのシステムやサービスへのアクセス要求に対する権限を付与することを可能にします。

［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
ネットワークの種類	アカウント・ネットワークの種類：GPRS（General Packet Radio Service）または CSD（Circuit-Switched Data）を選択します。
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。
接続タイムアウト（秒）	Radius サーバが応答を待機する秒数。

WAP Radius のオプションを設定するには，次の手順を実行します。

1 ［**実行環境の設定**］ボタンをクリックするか，［**仮想ユーザ**］＞［**実行環境の設定**］を選択して，［実行環境設定］ダイアログ・ボックスを表示します。［**Radius**］ノードをクリックします。



2 アカウンティング用の［**ネットワーク タイプ**］に GPRS（General Packet Radio Service）または CSD（Circuit-Switched Data）のどちらかを指定します。

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［**秘密鍵**］の値を入力します。

6［**接続タイムアウト**］に値を入力します。

7［**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。

索引