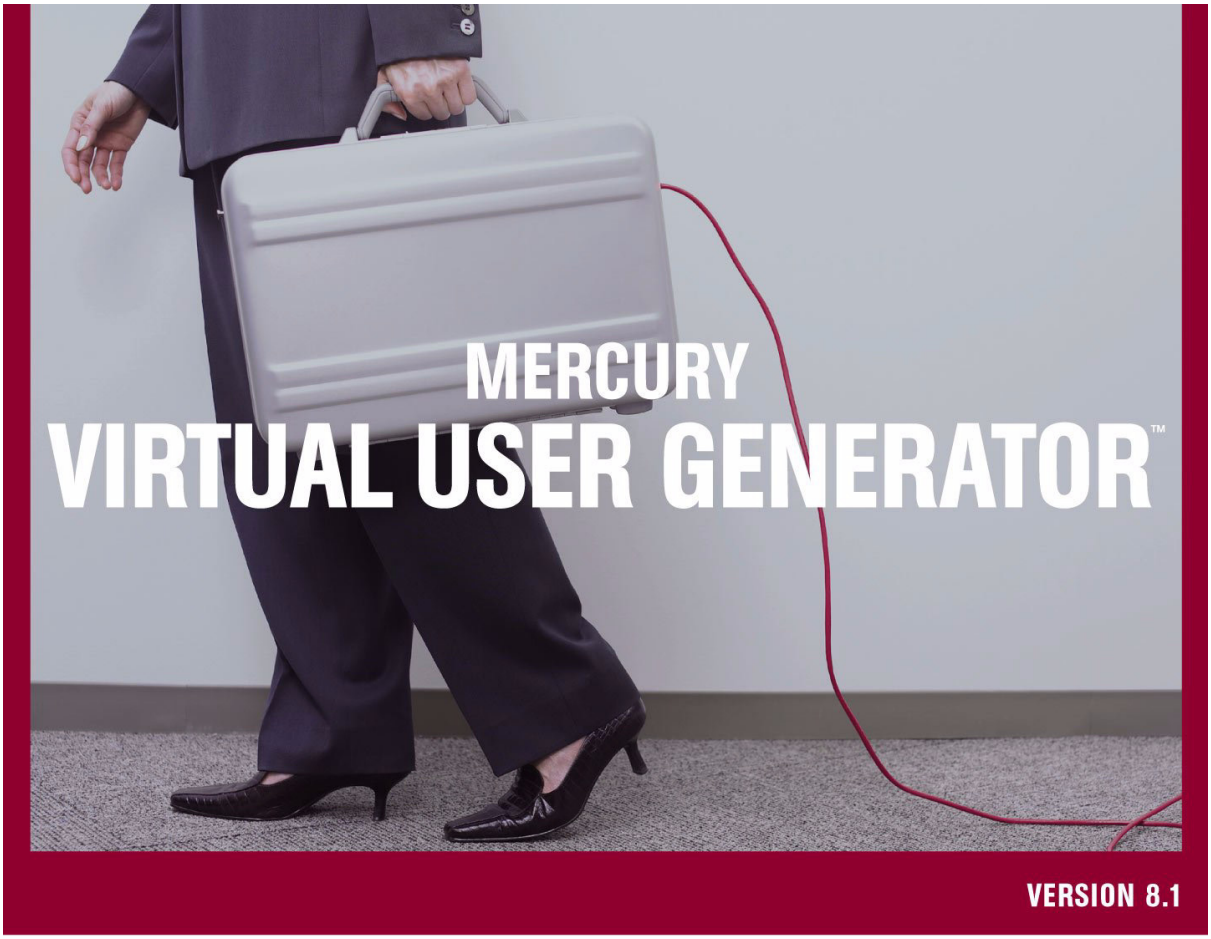


MERCURY
VIRTUAL USER GENERATOR™

VERSION 8.1

ユ ー ザ ー ズ ・ ガ イ ド

MERCURY™

Mercury 仮想ユーザ・ジェネレータ

ユーザーズ・ガイド

Version 8.1 Feature Pack 4

発行日：2006 年 9 月 24 日

MERCURY™

Mercury 仮想ユーザ・ジェネレータ・ユーザーズ・ガイド，Version 8.1，Feature Pack 4

Mercury Interactive Corporation
379 North Whisman Road
Mountain View, CA 94043
Tel: (650) 603-5200
Fax: (650) 603-5300
http://www.mercury.com

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

LRDBUG8.1FP4JP/01

目次

はじめに xxiii
本書の構成 xxiv
対象読者 xxv
LoadRunner オンライン・マニュアル xxv
その他のオンライン・リソース xxvi
文書の更新 xxvii
本書の表記規則 xxviii

第 1 部：仮想ユーザ・スクリプトの紹介

第 1 章： 仮想ユーザ・スクリプトの開発 3
仮想ユーザの紹介 4
仮想ユーザのタイプ 7
仮想ユーザ・スクリプトの作成手順 9

第 2 部：VUGEN を使った作業

第 2 章： VuGen の紹介 13
VuGen について 13
VuGen の起動 15
VuGen 環境オプションについて 16
環境オプションの設定 18
仮想ユーザ・スクリプトの表示と変更 18
VuGen を使った仮想ユーザ・スクリプトの実行 31
VuGen のコードについて 32
C 仮想ユーザ関数の使用方法 35
関数のヘルプ 39

第 3 章： VuGen ワークフローの表示 43
VuGen ワークフローの表示について 43
タスク表示枠の表示 44
ステップの記録 45
スクリプトの検証 46
スクリプトの拡張 48
ロードの準備 54
スクリプトの完了 55

第 4 章： VuGen を使った記録 57
VuGen を使った記録について 58
仮想ユーザ・スクリプトのセクション 58
仮想ユーザ・スクリプトの新規作成 60
プロトコルの追加と削除 63
仮想ユーザ・カテゴリの選択 64
スクリプトの新規作成 66
既存のスクリプトを開く 66
アプリケーションの記録 67
記録セッションの終了と保存 72
記録ログの表示 74
Zip ファイルの使用 76
アクションのインポート 77
認証情報の提供 78
仮想ユーザ・スクリプトの再生成 80

第 5 章： スクリプト生成オプションの設定 83
スクリプト生成オプションの設定について 83
スクリプト言語の選択 84
基本オプションの適用 85
相関オプションについて 87
記録オプションの設定 88

第 6 章： ビジネス・プロセス・レポートの作成 89
Word へのスクリプトのエクスポートについて 89
レポートの詳細の指定 90
レポートの内容の指定 91

第 7 章： ポートの割り当て設定 93
ポートの割り当て設定について 94
ポート割り当ての定義 94
新規サーバ・エントリの追加 96
高度なポート割り当てオプションの設定 99
ポートの割り当て記録オプションの設定 100

第 8 章 : 仮想ユーザ・スクリプトの拡張 105
仮想ユーザ・スクリプトの拡張について 106
仮想ユーザ・スクリプトへのトランザクションの挿入 107
ランデブー・ポイントの挿入（LoadRunner および Tuning のみ）..... 110
仮想ユーザ・スクリプトへのコメントの挿入 111
仮想ユーザ情報の取得 112
出力へのメッセージの送信 112
実行中の仮想ユーザ・スクリプトのエラー処理 116
仮想ユーザ・スクリプトの同期化 118
ユーザの思考遅延時間のエミュレート 118
コマンド・ライン引数の取り扱い 119
テキストの暗号化 120
手動でのパスワードのエンコーディング 121
スクリプト・フォルダへのファイルの追加 122

第 9 章 : VuGen パラメータを使った作業 123
VuGen パラメータの定義について 124
パラメータに関する制限 125
パラメータの作成 126
パラメータ・タイプについて 130
パラメータのプロパティの定義 133
既存のパラメータの使用 135
パラメータ・リストの使用 138
パラメータ化オプションの設定 140

第 10 章 : ファイル，テーブル，および XML パラメータ・タイプ 143
データ・ファイルまたはデータ・テーブルの選択または作成 144
ファイル・パラメータ・タイプのプロパティ設定 150
テーブル・パラメータ・タイプのプロパティ設定 152
ファイル・タイプまたはテーブル・タイプのパラメータに
データを割り当てる方法の選択 154
XML パラメータのプロパティの設定 159

第 11 章 : パラメータのプロパティの設定 167
パラメータのプロパティの設定について 167
内部データ・パラメータ・タイプのプロパティの設定 168
ユーザ定義関数のプロパティの設定 179
パラメータ形式のカスタマイズ 180
更新方法の選択 181

第 12 章 : ステートメントの相関 183
ステートメントの相関について 183
C 仮想ユーザ用の相関関数の使用 185
Java 仮想ユーザ用の相関関数の使用法 187
WDiff を使った仮想ユーザ・スクリプトの比較 188
保存したパラメータの変更 190

第 13 章 : 実行環境の設定 191
実行環境の設定について 192
実行論理の設定（マルチ・アクション） 193
ペースの設定 198
実行環境のペースの設定（マルチ・アクション） 199
ペースの設定と実行論理オプションの設定（シングル・アクション） 200
実行環境設定のログの設定 202
思考遅延時間の設定 207
実行環境の追加属性の設定 209
その他の設定 210
VB 実行環境の設定 217

第 14 章 : ネットワーク実行環境の設定 219
ネットワーク実行環境の設定について 219
ネットワーク速度の設定 220

第 15 章 : スタンドアロン・モードでの仮想ユーザ・スクリプトの実行 221
スタンドアロン・モードでの仮想ユーザ・スクリプトの実行について 222
VuGen での仮想ユーザ・スクリプトの実行 222
仮想ユーザ・スクリプトの再生 226
VuGen のデバッグ機能の使用 229
Web 仮想ユーザ・スクリプトでの VuGen のデバッグ機能の使用 234
VuGen ウィンドウを使った作業 235
コマンド・プロンプトからの仮想ユーザ・スクリプトの実行 236
UNIX コマンド・ラインからの仮想ユーザ・スクリプトの実行 236
スクリプトのテストへの統合 239

第 16 章 : テスト結果の表示 243
テスト結果の表示について 244
結果サマリ・レポートについて 245
レポート情報のフィルタリング 247
実行結果の検索 248
実行結果の管理 248

第 17 章 : Quality Center を使ったスクリプト管理 251
Quality Center を使ったスクリプト管理について 251
Quality Center の接続と切断 252
Quality Center プロジェクトからスクリプトを開く 255
Quality Center プロジェクトへのスクリプトの保存 257
VuGen でのスクリプトのバージョン管理 258

第 18 章 : Performance Center によるスクリプトの管理 269
Performance Center によるスクリプトの管理について 269
VuGen から Performance Center への接続 270
仮想ユーザ・スクリプトのアップロード 271
仮想ユーザ・スクリプトのダウンロード 276

第 3 部 : SOA および WEB サービスのテスト

第 19 章 : サービス・エミュレーションの作成 283
サービス・エミュレーションの作成について 283
エミュレーション・サービスの開始 284
ホストの選択 285
新規サービスの追加 286
エミュレートされたサービスの動作の設定 287
エミュレートされたサービスの操作 290
仮想ユーザ・スクリプトでのエミュレートされたサービスの使用 292

第 20 章 : SOA 向けのテスト・スクリプトの作成 295
SOA テスト・スクリプトの作成について 295
SOA テスト・スクリプトの概要 296
テスト・アスペクトの選択 297
自動テストの作成 298
スクリプトの表示と編集 302

第 21 章 : Web サービス仮想ユーザ・スクリプトの作成 307
Web サービス仮想ユーザ・スクリプトの作成について 307
Web サービス仮想ユーザ・スクリプトの概要 308
ワークフローの表示 310
Web サービス・スクリプトの記録 312
記録対象アプリケーションの選択 315
新しい Web サービス呼び出しの追加 317
チェックポイントの設定 318

第 22 章 : Web サービス呼び出しプロパティの設定 327
Web サービス呼び出しプロパティの設定について 327
Web サービス呼び出しプロパティについて 328
トランスポート層の設定 338
Web サービス・セキュリティ・ポリシーの作成 350
SAML オプションの設定 356
Web サービス・スクリプトの動作のカスタマイズ 360

第 23 章 : Web サービスの管理 365
Web サービス仮想ユーザ・スクリプトの管理について 366
サービス・プロパティの表示と設定 367
サービスのインポート 369
WSDL ファイルの検証 373
WSDL ファイルおよび XML ファイルの比較 380

第 24 章 : SOA/Web サービス・スクリプトの実行 385
Web サービス仮想ユーザの実行について 385
Web サービス・ツールキットの実行環境の設定 386
Web サービス JMS の実行環境の設定 387
Web サービス・スクリプトのスナップショットの表示 389
XML を使った作業 392
Web サービス関数の使用 399
Web サービス・レポートの表示 400

第 25 章 : サーバ・トラフィック・スクリプトの作成 403
サーバ・トラフィック・スクリプトの作成について 403
サーバ・トラフィック・スクリプトの概要 405
キャプチャ・ファイルの生成 406
サーバ・トラフィックに基づく基本スクリプトの作成 408
トラフィック情報の指定 410
受信フィルタまたは送信フィルタの選択 411
SSL 証明書の指定 412

第 4 部 : JAVA 言語プロトコルでの作業

第 26 章 : Java 言語仮想ユーザ・スクリプトの記録 417
Java 言語仮想ユーザ・スクリプトの記録について 418
記録を始める前に 418
Java 言語仮想ユーザ・スクリプトについて 420
パッケージの一部としてのスクリプトの実行 420
Java メソッドの表示 421
Java メソッドの手作業による挿入 423
スクリプト生成の設定 425

第 27 章 : Java 記録オプションの設定 429
Java 記録オプションの設定について 430
Java 仮想マシン（JVM）記録オプション 431
クラスパス記録オプションの設定 433
レコーダ・オプション 434
シリアル化オプション 437
相関オプション 439
デバッグ・オプション 441
CORBA オプション 444

第 28 章 : Java スクリプトの相関 447
Java スクリプトの相関について 448
標準的な相関 449
詳細相関 449
文字列の相関 451
シリアル化メカニズムの使用 452

第 29 章 : Java 実行環境の設定 459
Java 実行環境の設定について 459
JVM 実行環境の設定 460
実行環境の設定のクラスパス・オプションの設定 461

第 5 部 : アプリケーション配備ソリューション・プロトコル

第 30 章 : Citrix 仮想ユーザ・スクリプトの作成 465
Citrix 仮想ユーザ・スクリプトの作成について 466
Citrix 仮想ユーザ・スクリプトの開発の概要 467
クライアントとサーバのセットアップ 469
記録に関するヒント 472
Citrix 記録オプションについて 474
Citrix 記録オプションの設定 480
Citrix の表示設定 481
Citrix 実行環境の設定 482
Citrix 仮想ユーザ・スクリプトの表示と変更 486
再生の同期化 487
ICA ファイルについて 495
Citrix 関数の使用方法 497
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 502

第 31 章 : LoadRunner Citrix エージェントの使用方法 507
LoadRunner Citrix エージェントについて 508
Citrix エージェントからの利点 508
インストール 514
Citrix エージェントの効果と記憶容量 515
サンプル・スクリプト 515

第 6 部 : クライアント・サーバ・プロトコル

第 32 章 : データベース仮想ユーザ・スクリプトの作成 519
データベース仮想ユーザ・スクリプトの作成について 520
データベース仮想ユーザの紹介 521
データベース仮想ユーザ技術について 522
データベース仮想ユーザ・スクリプトの概要 523
データベース記録オプションの設定 525
データベースの詳細記録オプション 526
LRD 関数の使用法 529
データベース仮想ユーザ・スクリプトについて 535
グリッドを使った作業 537
エラー・コードの分析 540
エラー処理 541

第 33 章 : データベース仮想ユーザ・スクリプトの相関 545
データベース仮想ユーザ・スクリプトの相関について 545
スクリプトでの相関候補の検索 546
既知の値の相関 548
データベース仮想ユーザ相関関数 549

第 34 章 : DNS 仮想ユーザ・スクリプトの作成 551
DNS 仮想ユーザ・スクリプトの作成について 551
DNS 関数を使った作業 552

第 35 章 : WinSock 仮想ユーザ・スクリプトの作成 555
Windows Sockets 仮想ユーザ・スクリプトの記録について 555
Windows Sockets 仮想ユーザ・スクリプトの開発の概要 556
WinSock 記録オプションの設定 558
LRS 関数の使用 561

第 36 章 : Windows Sockets データを使った作業 565
Windows Sockets データを使った作業について 566
スナップショット・ウィンドウでのデータの表示 566
データ内の移動 568
バッファ・データの修正 571
バッファ名の修正 577
スクリプト・ビューでの Windows Sockets データの表示 578
データ・ファイルの形式について 580
バッファ・データの 16 進形式での表示 581
表示形式の設定 584
デバッグに関するヒント 587
WinSock スクリプトの手作業による相関 588

第 7 部 : ユーザ定義の仮想ユーザ・スクリプト

第 37 章 : ユーザ定義の仮想ユーザ・スクリプトの作成 593
ユーザ定義の仮想ユーザ・スクリプトの作成について 594
C 仮想ユーザ 595
C 仮想ユーザ・スクリプトのワークフロー・ウィザード 596
Java 仮想ユーザ 599
VB 仮想ユーザ 600
VBScript 仮想ユーザ 601
JavaScript 仮想ユーザ 602

第 38 章 : Java 仮想ユーザ・スクリプトのプログラミング 603
Java 仮想ユーザ・スクリプトのプログラミングについて 604
Java 仮想ユーザの作成 605
Java 仮想ユーザ・スクリプトの編集 605
Java 仮想ユーザ API 関数 608
Java 仮想ユーザ関数を使った作業 611
Java 環境の設定 617
Java 仮想ユーザ・スクリプトの実行 618
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 619
プログラミングに関するヒント 620

第 8 部 : 分散コンポーネント・プロトコル

第 39 章 : COM 仮想ユーザ・スクリプトの記録 625
COM 仮想ユーザ・スクリプトの記録について 626
COM の概要 626
COM 仮想ユーザを使った作業の開始 628
記録対象の COM オブジェクトの選択 629
COM 記録オプションの設定 632

第 40 章 : COM 仮想ユーザ・スクリプトの詳細.....641
COM 仮想ユーザ・スクリプトについて 641
VuGen COM スクリプトの構造について 642
VuGen COM スクリプトの検証 644
スクリプト内での相関候補の検索.....650
既知の値の相関 652

第 41 章 : COM 仮想ユーザ関数について 653
COM 仮想ユーザ関数について.....654
インスタンスの作成 654
IDispatch インタフェース起動メソッド.....655
型指定関数.....655
バリアント型 656
バリアントへの参照の代入.....657
パラメータ化関数.....658
バリアントからの抽出.....660
バリアントへの配列の代入.....660
配列の型と関数 661
バイト配列関数 662
ADO RecordSet 関数.....663
デバッグ関数 663
VB Collection のサポート 663

第 42 章 : CORBA-Java 仮想ユーザ・スクリプトの作成.....665
Corba-Java 仮想ユーザ・スクリプトの作成について 665
Corba-Java 仮想ユーザの記録 666
Corba-Java 仮想ユーザ・スクリプトを使った作業 670
Windows XP および Windows 2000 サーバでの記録.....672
アプリケーション固有のヒント 674

第 43 章 : RMI-Java 仮想ユーザ・スクリプトの作成 675
RMI-Java 仮想ユーザ・スクリプトの作成について.....675
RMI over IIOP の記録 676
RMI 仮想ユーザの記録 677
RMI 仮想ユーザ・スクリプトを使った作業 680

第 9 部：E - ビジネス・プロトコル

第 44 章 : AMF 仮想ユーザ・スクリプトの作成 685
AMF 仮想ユーザ・スクリプトの作成について 685
AMF の用語について 687
AMF 記録モードの設定 687
AMF 関数を使った作業 693
AMF スクリプトの相関 695
AMF データの表示 699
AMF スクリプトについて 701

第 45 章 : FTP 仮想ユーザ・スクリプトの作成 705
FTP 仮想ユーザ・スクリプトの作成について 705
FTP 関数の処理 706

第 46 章 : LDAP 仮想ユーザ・スクリプトの作成 709
LDAP 仮想ユーザ・スクリプトの作成について 709
LDAP 関数の処理 710
識別名エントリの定義 713

第 47 章 : Microsoft .NET 仮想ユーザ・スクリプトの記録 715
.NET 仮想ユーザ・スクリプトの記録について 716
Microsoft .NET 仮想ユーザを使った作業の概要 717
記録用フィルタの利点 719
フィルタ設定についてのガイドライン 721
記録用フィルタの設定 725
フィルタ・マネージャを使った作業 727
Microsoft .NET 記録オプションの設定 736
記録設定の設定 737
VuGen と Visual Studio でのスクリプトの表示 740
.NET 環境の実行環境の設定 742
データセットとグリッドの表示 744
Microsoft .NET スクリプトの相関 746
アプリケーションのセキュリティと権限の設定 749

第 48 章 : Web 仮想ユーザ・スクリプトの作成 753
Web レベル仮想ユーザ・スクリプトの作成について 754
Web 仮想ユーザの紹介 754
Web 仮想ユーザ技術について 755
Web 仮想ユーザのタイプの選択 756
Web 仮想ユーザ・スクリプト入門 757
Web セッションの記録 759
Web 仮想ユーザ・スクリプトの Java への変換 760

第 49 章 : Web（Click and Script）仮想ユーザ・スクリプトでの作業.....761
Web（Click and Script）仮想ユーザ・スクリプトでの作業について....762
Web（Click and Script）仮想ユーザ・スクリプトの表示.....762
Web（Click and Script）記録オプションの設定.....765
Web（Click and Script）仮想ユーザで作業をする際のヒント.....780
記録に関する問題.....781
記録に関するヒント.....782
再生に関する問題.....784
再生に関するヒント.....786
その他の問題.....787
その他のヒント.....789
Web（Click and Script）仮想ユーザ・スクリプトの拡張.....789

第 50 章 : Web 仮想ユーザ関数の使用.....795
Web 仮想ユーザ関数について.....795
関数の追加と編集.....796
Web（Click and Script）仮想ユーザ関数.....798
Web（HTTP/HTML）関数一覧.....801
キャッシュに格納された値の使用.....808

第 51 章 : インターネット・プロトコルの記録オプションの設定.....815
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.....815
プロキシ設定を使った作業.....816
記録オプションの詳細設定.....819
記録スキーマの設定.....821

第 52 章 : Web 仮想ユーザの記録オプションの設定.....827
記録オプションの設定について.....827
記録用のブラウザの指定.....828
記録レベルの選択.....829
記録レベルの設定.....843

第 53 章 : インターネット実行環境の設定.....845
インターネット実行環境の設定について.....846
プロキシ・オプションの設定.....847
ブラウザのエミュレーション・プロパティの設定.....852
インターネット・プリファレンスの設定.....857
Web サイトのフィルタリング.....866
デバッグ情報の取得.....867
HTML 圧縮の実行.....868

第 54 章 : Web ページの内容のチェック.....869
Web ページのコンテンツ・チェックについて.....869
コンテンツ・チェック実行環境の設定.....870

第 55 章 : 負荷下の Web ページ検証 877
負荷下の検証について 877
テキスト・チェックの追加 880
テキスト・チェック関数について 882
画像チェックの追加 887
追加プロパティの定義 890

第 56 章 : Web とワイヤレス仮想ユーザ・スクリプトの変更 893
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 894
仮想ユーザ・スクリプトへのステップの追加 895
仮想ユーザ・スクリプトからのステップの削除 896
アクション・ステップの変更 897
制御ステップの変更 914
サービス・ステップの変更 917
Web チェックの変更（Web のみ） 918

第 57 章 : Web 仮想ユーザ・スクリプトの相関ルールの設定 919
相関ステートメントについて 920
相関の方法について 921
VuGen の相関ルールの使用 922
相関のルールの設定 927
ルールのテスト 929
相関記録オプションの設定 930

第 58 章 : 記録後の仮想ユーザ・スクリプトの相関 933
ステートメントの相関について 934
相関結果タブの表示 935
VuGen の相関の設定 938
相関の検索の実行 941
手作業による相関 945
動的文字列の境界の定義 950

第 59 章 : XML ページのテスト 953
XML ページのテストについて 953
XML の URL ステップとしての表示 954
XML をユーザ定義の要求として挿入 956
XML ユーザ定義要求のステップの表示 958

第 60 章 : Web 仮想ユーザのヒント（パワー・ユーザ向け） 961
セキュリティの問題 961
クッキーの処理 965
実行時ビューア（オンライン・ブラウザ） 968
ブラウザ 970
設定と互換性の問題 974
パフォーマンス問題 974

第 61 章 : Web/WinSock 仮想ユーザ・スクリプトの記録 975
Web/WinSock 仮想ユーザ・スクリプトの記録について 976
Web/WinSock 仮想ユーザ・スクリプトを使った作業の開始 977
Web/WinSocket 記録オプションの設定 978
Web/WinSock セッションの記録 985
Palm アプリケーションの記録 987

第 10 部 : ENTERPRISE JAVA BEAN プロトコル

第 62 章 : EJB テストの実行 991
EJB テストについて 991
EJB Detector での作業 992
EJB テストの仮想ユーザの作成 997
EJB 記録オプションの設定 1001
EJB 仮想ユーザ・スクリプトについて 1002
EJB 仮想ユーザ・スクリプトの実行 1008

第 11 部 : ERP/CRM プロトコル

第 63 章 : Oracle NCA 仮想ユーザ・スクリプトの作成 1015
Oracle NCA 仮想ユーザ・スクリプトの作成について 1016
Oracle NCA 仮想ユーザの開発の概要 1017
記録作業のガイドライン 1018
名前によるオブジェクトの記録の有効化 1020
Personal Home Page からの Oracle Applications の使用 1023
Oracle NCA 仮想ユーザ関数の使用 1025
Oracle NCA 仮想ユーザについて 1029
Oracle NCA 実行環境設定 1031
Oracle NCA アプリケーションのテスト 1034
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 1037
その他に推奨される相関 1038
プラグマ・モードでの記録 1040

第 64 章 : SAPGUI 仮想ユーザ・スクリプトの作成 1043
SAPGUI 仮想ユーザ・スクリプトの作成について 1044
SAPGUI 仮想ユーザのための環境の確認 1045
SAPGUI 仮想ユーザ・スクリプトの作成手順 1056
SAPGUI 仮想ユーザ・スクリプトの記録 1057
SAPGUI 記録オプションの設定 1060
SAPGUI スクリプトのステップの対話的挿入 1063
SAPGUI 仮想ユーザ・スクリプトについて 1065
SAPGUI 仮想ユーザ・スクリプトの拡張 1069

第 65 章 : SAPGUI 仮想ユーザ・スクリプトの実行 1073
SAPGUI 仮想ユーザ・スクリプトの再生について 1073
SAPGUI のオプションのウィンドウの再生 1074
SAPGUI 実行環境の設定 1075
SAPGUI の関数 1078
SAPGUI 仮想ユーザ・スクリプトに関するヒント 1089
SAPGUI 仮想ユーザ・スクリプトのトラブルシューティング 1093
その他の参考資料 1095

第 66 章 : SAP-Web 仮想ユーザ・スクリプトの作成 1097
SAP-Web 仮想ユーザ・スクリプトの作成について 1098
SAP-Web 仮想ユーザ・スクリプトの作成手順 1098
SAP-Web 記録オプションの設定 1100
SAP-Web 仮想ユーザ・スクリプトについて 1101
SAP-Web 仮想ユーザ・スクリプトの再生 1103

第 67 章 : Siebel-Web 仮想ユーザ・スクリプトの作成 1105
Siebel-Web 仮想ユーザ・スクリプトの作成について 1105
Siebel-Web セッションの記録 1106
Siebel-Web スクリプトの相関 1107
SWECount，ROWID，および SWET パラメータの相関 1114
Siebel-Web 仮想ユーザ・スクリプトのトラブルシューティング 1116

第 12 部 : レガシ・プロトコル

第 68 章 : RTE 仮想ユーザ・スクリプトの紹介 1123
RTE 仮想ユーザ・スクリプトの作成について 1123
RTE 仮想ユーザの紹介 1124
RTE 仮想ユーザ技術について 1125
RTE 仮想ユーザ・スクリプトの概要 1125
TE 関数の使用 1127
ターミナル・キーの PC キーボード・キーへの割り当て 1129

第 69 章 : RTE 仮想ユーザ・スクリプトの記録 1131
RTE 仮想ユーザ・スクリプトの記録について 1132
RTE 仮想ユーザ・スクリプトの新規作成 1132
ターミナルの設定と接続の手順の記録 1133
一般的なユーザ・アクションの記録 1138
ログオフ手順の記録 1139
RTE 設定オプションの設定 1140
RTE 記録オプションの設定 1141
ターミナル・エミュレータへの入力 1144
一意のデバイス名の生成 1147
フィールド区分文字の設定 1148

第 70 章 : RTE 実行環境の設定 1151
ターミナル・エミュレータ実行環境の設定について 1152
接続試行回数の変更 1153
元のデバイス名の指定 1153
入力遅延の設定 1154
X SYSTEM 同期化の設定 1154

第 71 章 : RTE 仮想ユーザ・スクリプトの同期化 1157
仮想ユーザ・スクリプトの同期化について 1157
ブロック・モード（IBM）・ターミナルの同期化 1158
キャラクタ・モード（VT）・ターミナルの同期化 1162

第 72 章 : ターミナル画面からのテキストの読み取り 1167
ターミナル画面からのテキストの読み取りについて 1167
画面からのテキストの読み取り 1168

第 13 部 : メール・サービス・プロトコル

第 73 章 : メール・サービス仮想ユーザ・スクリプトの作成 1171
メール・サービス仮想ユーザ・スクリプトの作成について 1171
メール・サービス仮想ユーザ・スクリプトの概要 1172
IMAP 関数での作業 1174
MAPI 関数での作業 1176
POP3 関数での作業 1178
SMTP 関数での作業 1179

第 14 部 : ミドルウェア・プロトコル

第 74 章 : Jacada 仮想ユーザ・スクリプトの作成 1183
Jacada 仮想ユーザ・スクリプトについて 1183
Jacada 仮想ユーザの概要 1184
Jacada 仮想ユーザの記録 1186
Jacada 仮想ユーザの再生 1188
Jacada 仮想ユーザ・スクリプトの詳細 1189
Jacada 仮想ユーザ・スクリプトでの作業 1190

第 75 章 : Tuxedo 仮想ユーザ・スクリプトの作成 1193
Tuxedo 仮想ユーザ・スクリプトについて 1194
Tuxedo 仮想ユーザ・スクリプトの概要 1195
LRT 関数の使用 1196
Tuxedo 仮想ユーザ・スクリプトの詳細 1201
Tuxedo バッファ・データの表示 1203
Tuxedo 仮想ユーザの環境設定の定義 1204
Tuxedo アプリケーションのデバッグ 1205
Tuxedo スクリプトの相関 1205

第 15 部：ストリーミング・データ・プロトコル

第 76 章：ストリーミング・データ仮想ユーザ・スクリプトの作成 ...1215
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 1216
ストリーミング・データ仮想ユーザ・スクリプトの概要.....1216
RealPlayer LREAL 関数の使用.....1218
Media Player MMS 関数の使用 1218

第 16 部：ワイヤレス・プロトコル

第 77 章：ワイヤレス仮想ユーザ・スクリプトの記録.....1223
WAP プロトコルについて 1223
ワイヤレス仮想ユーザ・スクリプトの作成の概要 1225
ワイヤレス仮想ユーザ関数の使用法 1227
プッシュのサポート 1229
VuGen でのプッシュのサポート.....1231

第 78 章：WAP 実行環境の設定.....1233
WAP 実行環境の設定について.....1233
ゲートウェイ・オプションの設定.....1234
Radius 接続データの設定.....1238

第 79 章：MMS 仮想ユーザ・スクリプトの作成.....1241
MMS（マルチメディア・メッセージング・サービス）
　仮想ユーザ・スクリプトについて 1241
MMS 実行環境の設定 1242
コントローラでの MMS シナリオの実行 1244
MMS API 1245

第 17 部：上級ユーザのために

第 80 章：Visual Studio による仮想ユーザ・スクリプトの作成 1251
Visual Studio による仮想ユーザ・スクリプトの作成について 1251
Visual C による仮想ユーザ・スクリプトの作成 1252
Visual Basic による仮想ユーザ・スクリプトの作成.....1254
実行環境の設定とパラメータの設定 1256

第 81 章：XML API プログラミング.....1257
XML API プログラミングについて.....1257
XML 文書について.....1259
XML 関数の使用方法 1260
XML 関数のパラメータの指定.....1263
XML 属性での作業.....1265
XML スクリプトの作成 1265
記録されたセッションの拡張.....1267

第 82 章 : VuGen のデバッグのヒント 1273
一般的なデバッグのヒント....1274
C 関数を使用した追跡....1274
付加的な C 言語のキーワードの追加 1274
再生出力の検証 1275
データベース・アプリケーションのデバッグ 1275
Oracle Applications を使った作業 1277
Oracle 2-Tier 仮想ユーザでの一般的な問題の解決方法 1278
2-tier データベースのスクリプト作成のヒント....1282
PeopleSoft-Tuxedo スクリプトの実行 1291

第 83 章 : その他の詳細情報....1293
記録中に生成されるファイル....1294
再生中に生成されるファイル....1296
UNIX コマンド・ラインからの仮想ユーザの実行....1298
仮想ユーザの動作の指定 1299
コマンド・ライン・パラメータ 1300
OLE サーバの記録....1301
.dat ファイルの検証 1303
新規仮想ユーザ・タイプの追加 1304

第 18 部 : 付録

付録 A: 外部関数の呼び出し 1313
DLL のロード : ローカル....1313
DLL のロード : グローバル 1314

付録 B: 多国語を使った作業 1317
多国語を使った作業について 1317
手作業による文字列エンコーディングの変換....1318
パラメータ・ファイル内の文字列エンコーディングの変換 1320
Web 記録および再生のための文字列エンコーディングの設定 1322
Accept-Language ヘッダの言語の指定 1324
プロトコルに関する制限 1326
Quality Center の統合 1327

付録 C: UNIX プラットフォームでのスクリプトのプログラミング 1329
UNIX プラットフォームで実行可能な
　仮想ユーザ・スクリプトのプログラミングについて 1330
テンプレートの生成 1331
仮想ユーザのアクションのプログラミング 1331
仮想ユーザの実行環境設定....1333
トランザクションとランデブー・ポイントの定義....1338
スクリプトのコンパイル 1339

付録 D: キーボード・ショートカットの使用 1341

付録 E: 記録オプションと実行環境の設定 1343
実行環境の設定 1343
記録オプション 1347

索引 1353

はじめに

Mercury 仮想ユーザ・ジェネレータ **VuGen** は，仮想ユーザ・スクリプトを作成するためのツールです。VuGen では，典型的なビジネス・プロセスを実行しているユーザの操作を記録することによって，仮想ユーザ・スクリプトを作成します。これらのスクリプトを使用することで，現実世界の状況をエミュレートできます。

VuGen で作成したスクリプトは，LoadRunner，Performance Center，Business Availability Center などのほかの Mercury 製品と組み合わせて使用できます。

LoadRunner は，アプリケーションのパフォーマンスをテストするための Mercury のツールです。LoadRunner は，アプリケーション全体に負荷をかけて，クライアント，ネットワーク，サーバの潜在的なボトルネックを検出し，特定します。

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

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

本書には次の項が含まれます。

- 本書の構成
- 対象読者
- LoadRunner オンライン・マニュアル
- その他のオンライン・リソース
- 文書の更新
- 本書の表記規則

本書の構成

本書は次の部で構成されています。

第 1 部　仮想ユーザ・スクリプトの紹介

仮想ユーザ・スクリプトおよび各種仮想ユーザを紹介します。仮想ユーザ・スクリプトを作成する工程の概要も説明します。

第 2 部　VuGen を使った作業

仮想ユーザ・ジェネレータのインタフェースおよびスクリプトの記録と再生について説明します。また，標準の実行環境の設定方法，データ・パラメータの使用方法，スクリプトのカスタマイズ方法も説明します。

第 3 部から第 16 部　記録プロトコル

これらの項では，さまざまなプロトコルを対象に仮想ユーザ・スクリプトを記録する方法について情報を提供します。次の各項があります。

- SOA および Web サービスのテスト
- Java 言語プロトコルでの作業
- アプリケーション配備ソリューション・プロトコル
- クライアント・サーバ・プロトコル
- ユーザ定義の仮想ユーザ・スクリプト
- 分散コンポーネント・プロトコル
- E - ビジネス・プロトコル
- Enterprise Java Bean プロトコル
- ERP/CRM プロトコル
- レガシ・プロトコル
- メール・サービス・プロトコル
- ミドルウェア・プロトコル
- ストリーミング・データ・プロトコル
- ワイヤレス・プロトコル

第 17 部　上級ユーザのために

デバッグのための一般的なヒント，VuGen によって生成されるファイルの説明，Visual C および Visual Basic でのスクリプトのプログラミング方法など，高度な使用方法について情報を提供します。

第 18 部　付録

ほかの高度な話題について，技術的な概説や情報が含まれています。外部関数の呼び出し，UNIX 環境でのプログラミング，多国語での作業，キーボード・ショートカットについて説明しています。

対象読者

本書は Mercury 仮想ユーザ・ジェネレータを利用する次の方々を対象とします。

- スクリプト開発者
- 負荷テスト実施者

本書では，読者がご利用のエンタープライズ・アプリケーションについてある程度の知識を持っていることを前提とします。

本書は，VuGen を使って作成し，実行することが可能なすべての仮想ユーザ・タイプを対象とします。フロント・エンドの GUI を対象としたスクリプトを作成する場合には適用できません。その場合は，『**WinRunner ユーザーズ・ガイド**』を参照してください。

LoadRunner オンライン・マニュアル

LoadRunner には，次のオンライン・マニュアルがあります。

最初にお読みください：LoadRunner に関する最新のお知らせと情報を提供します。「最初にお読みください」には，［**スタート**］メニューからアクセスします。

Mercury LoadRunner クイック・スタートでは，LoadRunner の概要を簡潔に順を追って説明し，その使用法を紹介します。「クイック・スタート」には，［**スタート**］メニューからアクセスします。

オンライン文書 / 印刷マニュアルには PDF 形式の文書が含まれています。［**ヘルプ**］ボタンをクリックし，［**オンライン文書**］を選択します。

オンライン・ヘルプは，任意の VuGen ウィンドウでウィンドウ内をクリックし，**F1** キーまたは［**ヘルプ**］ボタンを押すと表示できます。

Mercury LoadRunner オンライン・ヘルプには，次のヘルプが含まれます。

- **Error Codes Troubleshooting** には，コントローラ接続および Web プロトコル・エラーの分かりやすい説明とトラブルシューティング，および Winsock，SAPGUI，Citrix プロトコルに関する一般的なトラブルシューティングのヒントが含まれます。
- **LoadRunner Agent Configuration Tool Online Help** は，エージェント設定ツールに関するヘルプです。このヘルプには，［**スタート**］メニューから［エージェント設定］ダイアログ・ボックスにアクセスし，［ヘルプ］ボタンをクリックしてアクセスします。
- **LoadRunner Controller and Monitor Automation Reference Automation COM** は，LoadRunner コントローラを実行し，コントローラのユーザ・インタフェースで使用できるほとんどのアクションを実行するためのプログラムを書くことのできるインタフェースです。LoadRunner のオンライン文書からアクセスできます。

LoadRunner 関数リファレンス：仮想ユーザ・スクリプトの作成時に使用する LoadRunner の関数をすべて，その使用例と共に参照できます。オンライン・マニュアル『**LoadRunner 関数リファレンス**』のアップデートについては，Mercury のカスタマー・サポート Web サイトをご覧ください。

その他のオンライン・リソース

Mercury Tours サンプル Web サイトは，本書で説明する多くの例で使用されています。**Mercury Tours にアクセスするには，次のサイトにアクセスします。** Mercury Tours Web サイトの URL は，http://newtours.mercuryinteractive.com です。

Knowledge Base：普段お使いのブラウザで，Mercury のカスタマー・サポート Web サイト（US サイト）の Knowledge Base ページを直接開くことができます。この Web サイトの URL は，http://support.mercury.com/cgi-bin/portal/CSO/kbBrowse.jsp です。

カスタマー・サポート Web サイト：普段お使いのブラウザで，Mercury のカスタマー・サポート Web サイトを開きます。このサイトで，Mercury カスタマー・サポートのナレッジ・ベースを参照し，独自の項目を追加できます。また，ユーザ・ディスカッション・フォーラムへの書き込みや検索、サポート要求の送信、パッチや更新された文書のダウンロードなどを行うこともできます。［**ヘルプ**］＞オンライン技術サポート（または［**サポート情報**］）を選択します。この Web サイトの URL は，http://www.mercury.com/jp/services/support/ です。

Mercury のホームページ：普段お使いのブラウザで，Mercury の Web サイトを開きます。このサイトでは，Mercury とその製品に関する最新情報を提供します。新しいソフトウェアのリリース，セミナー，展示会，カスタマー・サポート，トレーニング・サービスなどの情報も含まれています。［**ヘルプ**］＞［**Mercury Interactive の Web サイト**］を選択します。この Web サイトの URL は http://www.mercury.com/jp です。

Mercury Best Practices：第一級の IT 環境を計画，作成，導入，および管理するためのガイドラインを提供します。Mercury では，Process Best Practices，Product Best Practices，People Best Practices の 3 種類のベスト・プラクティスを用意しています。Mercury ソフトウェアのライセンス契約を結んでいる方は，カスタマー・サポート・サイト http://support.mercury.com から Mercury Best Practice の閲覧と利用が可能です。

文書の更新

Mercury の製品マニュアルは，常に更新されています。この文書の最新版はカスタマー・サポート Web サイト（http://www.mercury.com）（英語サイト）からダウンロードできます。

更新された文書をダウンロードするには，次の手順を実行します。

1 カスタマー・サポート Web サイトの［**Documentation**］リンクをクリックします。

2 ［**Please Select Product**］で［**LoadRunner**］を選択します。

LoadRunner がリストに表示されていない場合は，顧客プロファイルに追加する必要があります。［**My Account**］をクリックしてプロファイルを更新します。

3 ［**Retrieve**］をクリックします。文書のページが開き，現在のリリースと以前のリリースに関する使用可能な文書がリストされます。文書が最近更新された場合，文書名の隣に「**Updated**」のマークが表示されます。

4 文書のリンクをクリックして，文書をダウンロードします。

本書の表記規則

本書では次の表記規則に従います。

［**UI 要素**］	インタフェース要素の名前は，その要素を使用したアクションを実行する手順の説明で全角の大括弧に**太字**で示します。 例：［**保存**］ボタンをクリックします。
＜**置換する値**＞	実際の値と置換するファイル・パスや URL アドレスの一部は大括弧で囲みます。 例：＜ **MyProduct インストール・フォルダ**＞ **¥bin**
Example	使用例やユーザがそのまま入力しなければならない文字列は，この形式で示します。 例：編集ボックスに「Hello」と入力します。
CTRL+C	キーボードのキーはこの形式で示します。 例：ENTER. キーを押します。
Function_Name	メソッド名や関数名はこの形式で示します。 例：**wait_window** ステートメントには次のパラメータがあります。
[ ]	省略可能な引数は，半角の大括弧で囲んで示します。
{ }	引数に割り当てる値の候補は，中括弧で囲んで示します。値をいずれか 1 つ割り当てる必要があります。
...	構文内の省略記号は，同じ形式で項目をさらに組み入れることができることを意味します。プログラミング例に含まれる場合は，何行かが意図的に省略されていることを示します。
|	垂直バー（パイプ記号）は，バーで区切られているオプションのいずれかを指定しなければならないことを示します。

第 1 部

仮想ユーザ・スクリプトの紹介

第 1 章

仮想ユーザ・スクリプトの開発

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

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

本章では，次の項目について説明します。

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

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

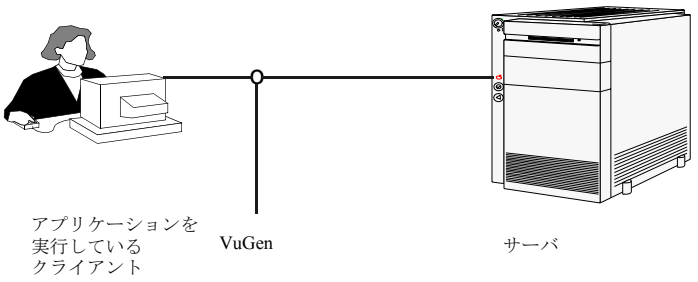
仮想ユーザの紹介

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

そのようなエミュレーションを実現するために，実際のユーザの代わりとして**仮想ユーザ**を使用します。仮想ユーザは実際のユーザのアクションを，一般的なアプリケーションの標準的なビジネス・プロセスを実行することでエミュレートします。仮想ユーザが記録セッション中に実行する操作は**仮想ユーザ・スクリプト**によって表現されます。

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

記録時には，VuGen がデータベースのクライアント側を監視し，ユーザとサーバとの間で送受信されるすべての要求を追跡します。

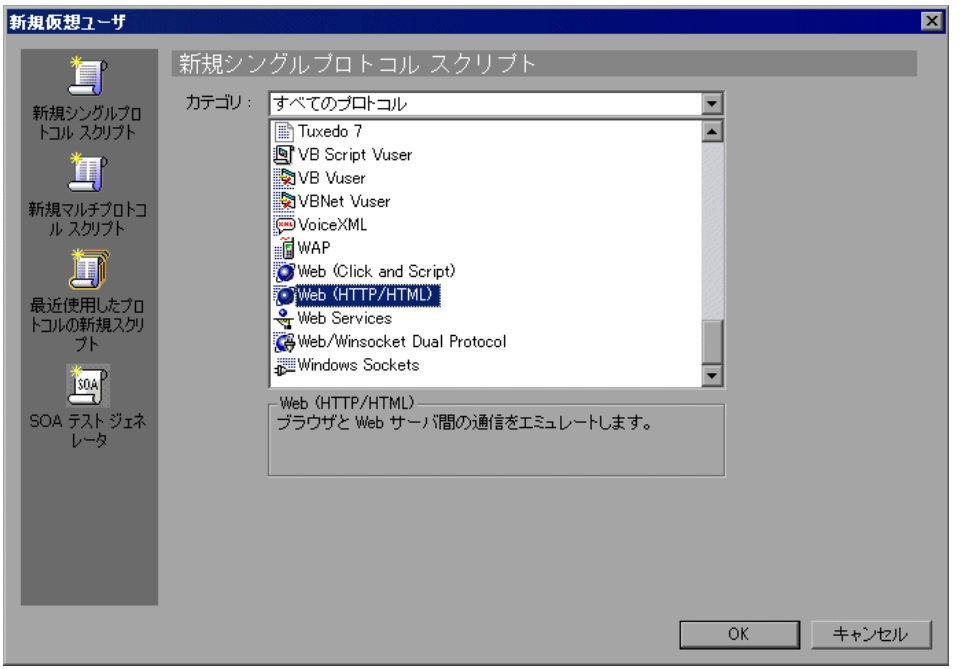


再生時には，仮想ユーザ・スクリプトがサーバ API への呼び出しを実行して，サーバと直接やり取りします。仮想ユーザからサーバと直接通信するときは，システム・リソースがクライアント・インタフェースで必要になりません。このため，1 台のワークステーションで多数の仮想ユーザを同時に実行でき，数台のテスト用マシンだけで大きなサーバ負荷をエミュレートすることが可能になります。

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

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

VuGen ではさまざまなタイプの仮想ユーザを記録でき，それぞれのタイプが特定の負荷テスト環境またはトポロジに対応しています。VuGen で新しいテストを開くと，サポートされているすべてのプロトコルの一覧が表示されます。



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

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。たとえば，Web 仮想ユーザ・スクリプトは，Web ブラウザを操作しているユーザをエミュレートする場合に使用します。FTP 仮想ユーザは，FTP セッションのエミュレートに使用します。さまざまな仮想ユーザ・テクノロジを単独で使用したり，組み合わせて使用したりすることによって，効果的な負荷テスト，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リューション**］：Citrix プロトコル用。
- ［**クライアント / サーバ**］：DB2 CLI，DNS，Microsoft .NET，MS SQL，ODBC，Oracle（2-Tier），Sybase CTlib，Sybase DBlib，Windows Sockets プロトコル用。
- ［**ユーザ定義**］：C テンプレート，Visual Basic テンプレート，Java テンプレート，Javascript，VB Script，VB，VBNet タイプ・スクリプト用。
- ［**分散コンポーネント**］：COM/DCOM，CORBA-Java，Microsoft .NET，RMI-Java プロトコル用です。
- ［**e ビジネス**］：FTP，LDAP，Microsoft .NET，Palm，Web(Click and Script)，Web（HTTP/HTML），Web サービス，Web/Winsocket Dual プロトコル用。
- ［**Enterprise Java Beans**］：EJB テストおよび RMI-Java プロトコル用。
- ［**ERP/CRM**］：Oracle NCA，Oracle Web Applications 11i，Peoplesoft Enterprise，Peoplesoft-Tuxedo，SAP-Web，SAPGUI，デュアル SAPGUI/SAP-Web，Siebel（Siebel-DB2CLI，Siebel-MSSQL，Siebel-Oracle，Siebel-Web）プロトコル用。
- ［**レガシ**］：ターミナル・エミュレーション（RTE）用。
- ［**メール サービス**］：Internet Messaging（IMAP），MS Exchange（MAPI），Post Office Protocol（POP3），Simple Mail Protocol（SMTP）プロトコル用。
- ［**ミドルウェア**］：Jacada，Tuxedo（6，7）プロトコル用。

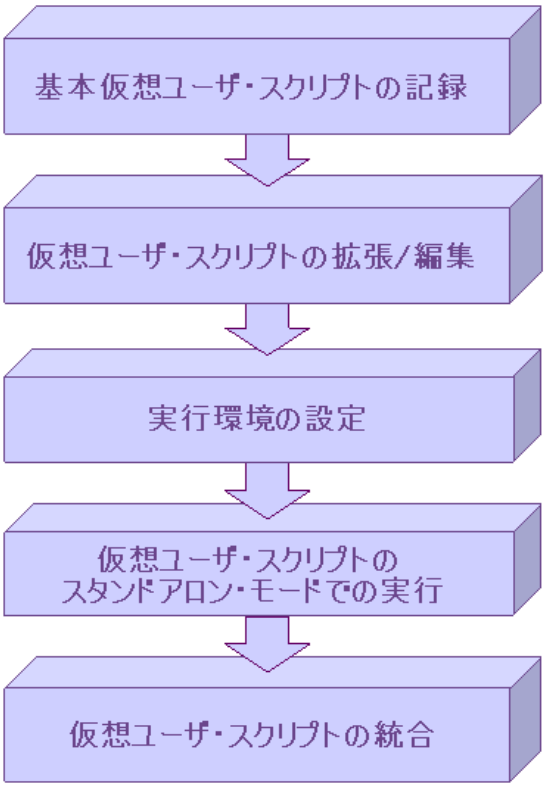
- ［**ストリーミング**］: MediaPlayer と RealPlayer プロトコル用。
- ［**ワイヤレス**］: i モード，マルチメディア・メッセージング・サービス（MMS），VoiceXML，WAP プロトコル用。

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

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

仮想ユーザ・スクリプトの作成手順

次の図は，仮想ユーザ・スクリプトの作成プロセスの概略です。



仮想ユーザ・スクリプトの作成プロセスは次のとおりです。

1 VuGen を使って基本となるスクリプトを記録します。Windows ベースの GUI アプリケーションをテストする場合，またはアプレットや Flash などの複合的な Web 環境をテストする場合は，Mercury の GUI ベースのツールである WinRunner や QuickTest Professional を使用しなければならないことがあります。

2 制御フロー・ステートメントその他の Mercury API 関数を追加して，基本となるスクリプトを拡張します。

3 実行環境を設定します。実効環境の設定には，反復，ログ，タイミングの情報などがあり，これらの設定によってスクリプト実行中における仮想ユーザの振る舞いが決まります。

4 スクリプトの機能を確認し，スタンドアロン・モードで実行します。

5 スクリプトが正常に機能することを確認した後，スクリプトを自分の環境（LoadRunner シナリオ，Performance Center 負荷テスト，Tuning Module セッション，または 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**』，**Performance Center**，**Business Availability Center** の各ユーザーズ・ガイドを参照してください。

第 2 部

VuGen を使った作業

第 2 章

VuGen の紹介

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

本章では，次の項目について説明します。

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

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

VuGen について

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

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

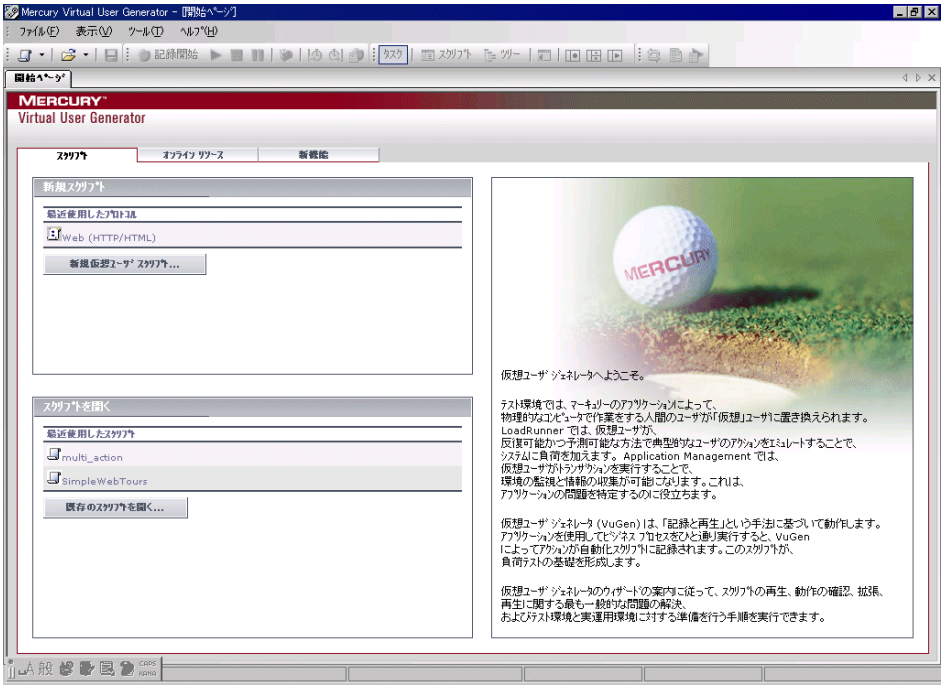
注：VuGen では Windows プラットフォームにおいてのみセッションを記録できます。しかし，記録した仮想ユーザ・スクリプトは，Windows および UNIX の両プラットフォームで実行できます。

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

VuGen の起動

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

仮想ユーザ・ジェネレータの開始ページが開きます。



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

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

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

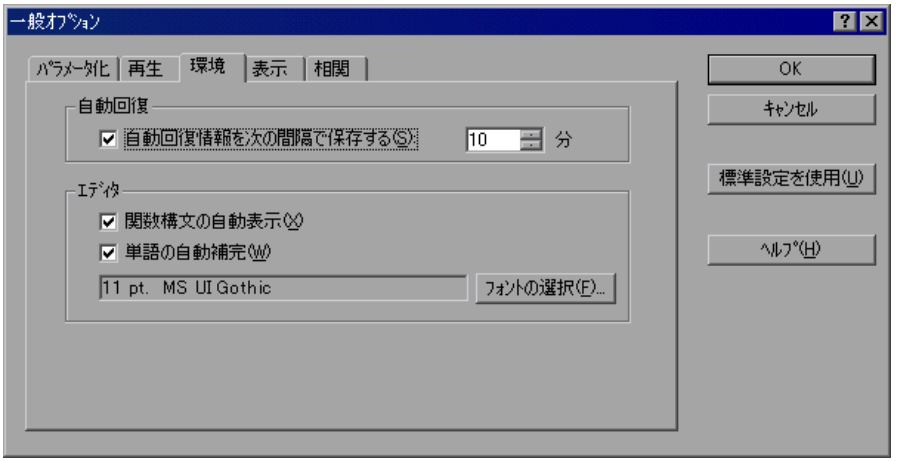
一覧にないプロトコルを使って新しいスクリプトを作成するには，［**新規仮想ユーザ スクリプト**］をクリックします。

Zip アーカイブから既存のスクリプトを開くには，［**ファイル**］＞［**Zip 操作**］＞［**Zip ファイルからインポート**］を選択します。

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

VuGen 環境オプションについて

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



自動回復

自動回復オプションを使用することにより，クラッシュまたは停電が発生したときにスクリプトの設定を復元できます。自動回復オプションを使用するには，［**自動回復情報を次の間隔で保存する**］チェック・ボックスを選択し，保存の実行間隔を分単位で指定します。

エディタ

エディタのオプションで，単語の自動補完や関数構文の自動表示を行う VuGen の IntelliSense 機能を設定できます。

- **[関数構文の自動表示]**：関数の開始括弧を入力すると，関数の構文，引数，プロトタイプが自動的に表示されます。関数構文の自動表示を VuGen 全体で有効にするには，このオプションの隣のチェック・ボックスを選択します。この機能を無効にするには，[**関数構文の自動表示**] オプションの隣のチェック・ボックスをクリアします。[**関数構文の自動表示**] オプションを VuGen 全体で無効にしている場合でも，エディタ画面で開始括弧を入力した後に，Ctrl キーと Shift キーを押しながらスペース・キーを押すか，[**編集**] ＞ [**関数の構文を表示**] を選択すれば，構文を表示させることができます。
- [**単語の自動補完**]：関数の最初のアンダーバーを入力すると，関数の一覧が表示されます。関数全部を手作業で入力しなくても正確な関数を選ぶことができます。単語の自動補完を VuGen 全体で有効にするには，このオプションの隣のチェック・ボックスを選択します。この機能を無効にするには，[**単語の自動補完**] オプションの隣のチェック・ボックスをクリアします。このオプションを VuGen 全体で無効にしている場合でも，Ctrl キーを押しながらスペース・キーを同時に押すか，エディタで入力しながら [**編集**] ＞ [**単語入力候補**] を選択すれば，自動補完のリスト・ボックスを表示できます。
- [**フォントの選択**]：エディタに表示されるフォントを設定するには，[**フォントの選択**] をクリックします。.[フォント] ダイアログ・ボックスが表示されます。使用するフォント，スタイル，サイズを選択し，[**OK**] をクリックします。使用できるフォントは，固定幅のフォント（Courier，Lucida Console，FixedSys など）だけです。

標準設定を使用

標準設定では，[**関数構文の自動表示**] や [**単語の自動補完**] は VuGen 全体で有効になっています。自動復元は，10 分に設定されています。これらの値を元に戻すには，[**標準設定を使用**] をクリックします。

環境オプションの設定

環境関連オプションを設定するには，次の手順を実行します。

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

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

3 エディタに表示されるフォントを設定するには，［**フォントの選択**］をクリックします。［フォント］ダイアログ・ボックスが表示されます。使用するフォント，スタイル，サイズを選択し，［**OK**］をクリックします。使用できるフォントは，固定幅のフォント（Courier，Lucida Console，FixedSys など）だけです。

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

仮想ユーザ・スクリプトの表示と変更

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

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

スクリプト・ビューでのコードの表示

スクリプト・ビューでは，スクリプトに記録または挿入された実際の API 関数を表示できます。上級ユーザは，このビューで C や仮想ユーザ API の関数，あるいはフロー制御ステートメントを追加することで，スクリプトのプログラミングを行うことができます。

スクリプト・ビューを表示するには，次の手順を実行します。

VuGen のメイン・メニューから［**表示**］＞［**スクリプト ビュー**］を選択するか，［**スクリプトを表示**］アイコンをクリックします。スクリプトがテキスト形式のスクリプト・ビューで表示されます。すでにスクリプト・ビューが表示されている場合は，このメニュー項目は選択できません。

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

スクリプト・ビューで作業を行うときは，［**挿入**］＞［**新規ステップ**］コマンドを使用してスクリプトにステップを追加できます。または，単語の補完機能や関数構文の表示機能を使用して，関数を手作業で入力することもできます。詳細については，39 ページ「関数のヘルプ」を参照してください。

注：スクリプト・ビューの表示中に仮想ユーザ・スクリプトを変更すると，それに応じて仮想ユーザ・スクリプトのツリー・ビューも変更されます。テキスト形式のスクリプトに行われた変更を解釈できなかった場合は，スクリプト・ビューがツリー・ビューやサムネイル・ビューに変換されません。

ツリー・ビューでのスクリプトの表示

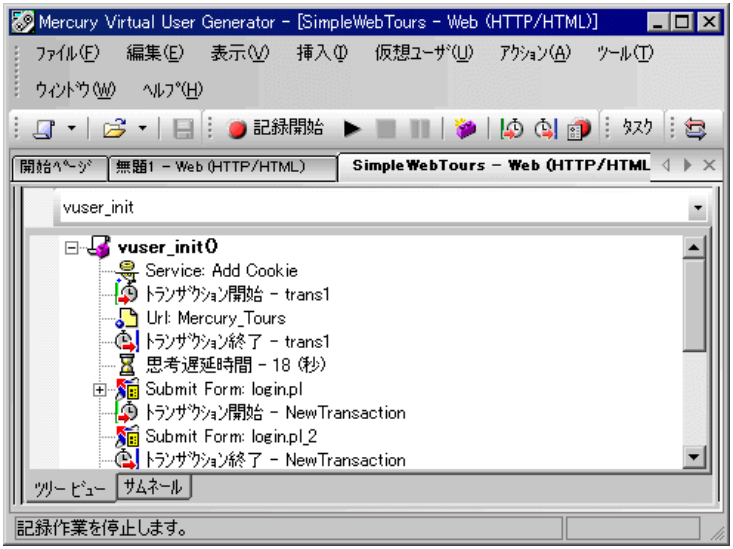
VuGen のツリー・ビューでは，仮想ユーザ・スクリプトがアイコン形式で表示され，各ステップが異なるアイコンで表されます。

ツリー・ビューを表示するには，次の手順を実行します。

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

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

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



ツリー・ビュー内では，ステップをドラッグして望みの位置に移動できます。また，ツリー階層の中で，既存のステップの間にステップを追加することもできます。

ツリー・ビューにステップを挿入するには，次の手順を実行します。

1 ステップをクリックします。

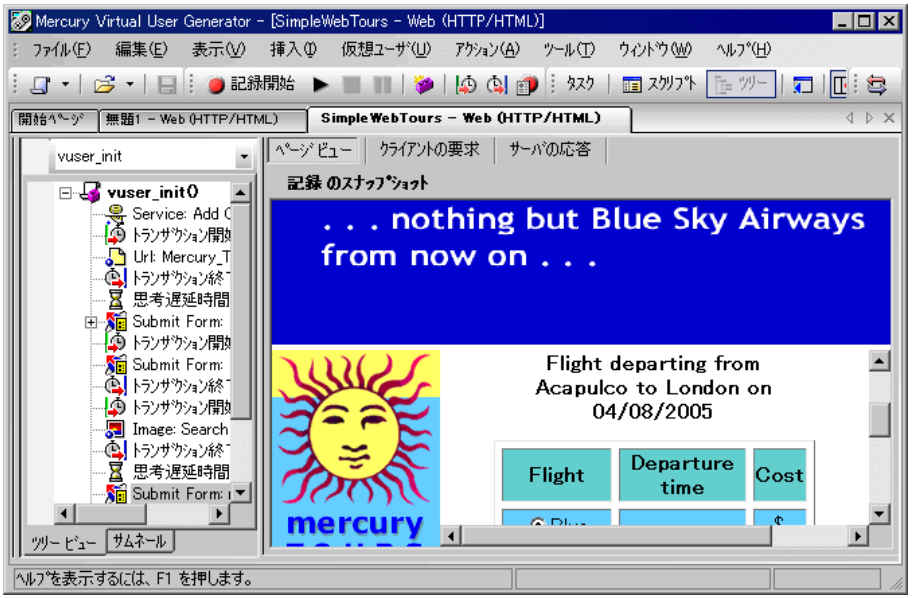
2 右クリック・メニューから［**前に挿入**］または［**後に挿入**］を選択します。［ステップの追加］ダイアログ・ボックスが開きます。

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

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

スナップショットについて

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



VuGen は，記録中に基本となるスナップショットをキャプチャし，再生中にも新たなスナップショットをキャプチャします。記録時のスナップショットと再生時のスナップショットを比較することによって，スクリプトを実行するために相関させる必要がある動的な値を見つけます。詳細については，第 58 章「記録後の仮想ユーザ・スクリプトの相関」を参照してください。

次のツールバー・ボタンを使用して，さまざまなスナップショット・ウィンドウを表示または非表示にすることができます。

記録時のスナップショットをフル・サイズのウィンドウで表示します。

記録時と再生時のスナップショットを分割したウィンドウで表示します。

再生時のスナップショットをフル・サイズのウィンドウで表示します。

スナップショットの表示と非表示を切り替えるには，次の手順を実行します。

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

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

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

スクリプトを再生するたびに，VuGen はスクリプトの結果ディレクトリ（**Iteration1**，**Iteration2**，など）に新たな再生時のスナップショットを保存します。

標準では，VuGen は記録時のスナップショットと，最初に再生されたときのスナップショットを比較します。別のスナップショットを選択して比較することもできます。特定の再生時のスナップショットを選択するには，[**表示**] > [**スナップショット**] > [**反復の選択**] を選択します。結果セットを選択して，[**OK**] をクリックします。

スナップショットのトラブルシューティング

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

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

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

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

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

- スクリプトが VuGen 6.02 以前のバージョンで記録された。
- インポートされたアクションに，スナップショットが含まれていない。
- 仮想ユーザ・ファイルが読み取り専用のディレクトリに格納されているため，VuGen が再生時のスナップショットを保存できなかった。
- ステップがリソースへのナビゲーションを表している。

スナップショット・ファイル

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

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

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

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

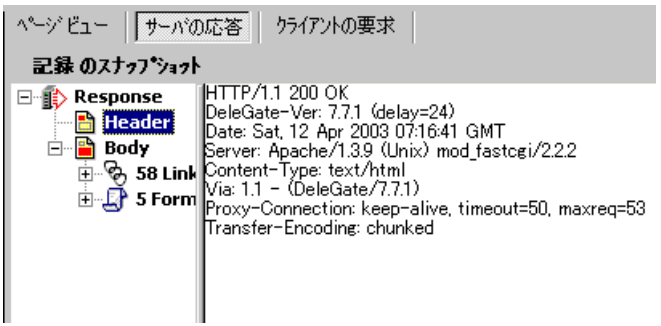
```
ctrx_sync_on_window("ICA Administrator Toolbar", ACTIVATE, 768, 0, 33,
        573, "snapshot12", CTRX_LAST);
```

Web 仮想ユーザのスナップショット・タブ

Web 仮想ユーザのスナップショット・ウィンドウには，次のタブがあります。

- [**ページ ビュー**]：ブラウザに表示されるとおりに，スナップショットを HTML で表示します。このボタンは，記録時および再生時の両方のスナップショットに対して使用できます。このビューを使って，正しいスナップショットを表示しているかどうかを確認できます。ただし，このビューでは，相関させる必要のある値は分かりません。
- [**サーバの応答**]：スナップショットのサーバ応答 HTML コードを表示します。このタブは，記録時および再生時の両方のスナップショットに対して使用できます。[HTML] ビューでは，左の表示枠にスクリプトのツリー構造と，ドキュメントのコンポーネントであるヘッダーと本体，およびそのタイトル，リンク，フォームなどの詳細も表示されます。

右の表示枠のソースの中から要素の HTML テキストを見つけるには，［サーバの応答］の左の表示枠で要素を選択し，右クリック・メニューから［**要素の検索**］を選択します。



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

スクリプトのサムネイルの表示

Web，SAPGUI，Citrix など一部の仮想ユーザ・タイプでは，スナップショットをサムネイルで表示できます。サムネイルは，ツリー・ビューに表示するか，トランザクション・エディタを使って表示します。

ツリー・ビューでのサムネイルの表示

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

標準設定では，スクリプト内の主要なステップのみがサムネイル・ビューに表示されます。すべてのサムネイルを表示するには，［**表示**］＞［**すべてのサムネイルを表示**］を選択します。スクリプト内の全ステップのサムネイルが表示されます。

注：反復が複数の場合，最後の反復の再生のサムネイルが表示されます。特定の反復のサムネイルを表示するには，［**表示**］＞［**スナップショット**］＞［**反復の選択**］を選択し，希望の反復を選びます。

ツリー・ビューでサムネイルを表示するには，次の手順を実行します。

1 左側の表示枠の下部にある［**サムネイル**］タブをクリックします。

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

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

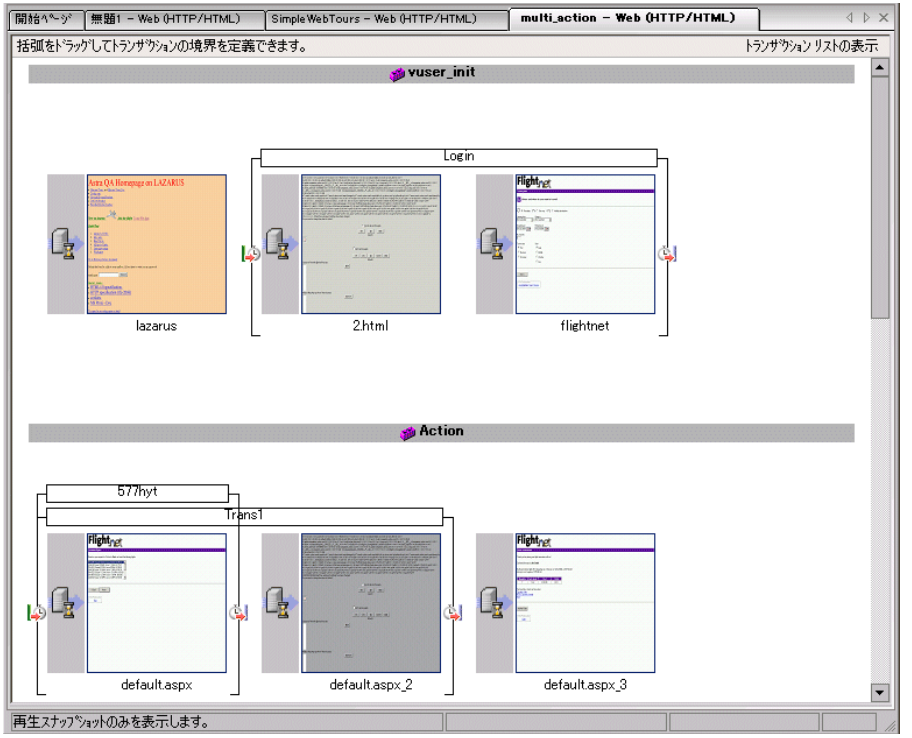
アクションの表示モードの設定

アクションを基準にしてスクリプトを表示する方法を指定できます（[**表示**] > [**アクション**]）。標準設定では，スクリプト・ビューの場合にのみ，左側の表示枠にスクリプト・アクションが表示されます。標準設定（[**自動的に開く**]）をそのまま使用することも，ツリー・ビューやスクリプト・ビューでアクション表示枠を開くこともできます。

- [**自動的に開く**]：スクリプト・ビューの場合のみ，左側の表示枠にスクリプト・アクションが表示されます。これは，標準設定です。アクション表示枠を非表示にするには，選択をクリアします。
- [**ツリー モードで開く**]：ツリー・ビューにアクションを表示します。アクション，ステップ，スナップショットの 3 つの表示枠が開きます（スナップショット・ビューが有効な場合）。各表示枠の幅を調整するには，境界線を任意の方向にドラッグします。アクション表示枠を非表示にするには，選択をクリアします。
- [**スクリプト モードで開く**]：スクリプト・ビューにアクションを表示します。アクションとスクリプト・ビューの 2 つの表示枠が開きます。アクション表示枠の幅を調整するには，境界線を任意の方向にドラッグします。アクション表示枠を非表示にするには，選択をクリアします。

ワークフロー・ウィザードでのサムネイルの表示

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



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

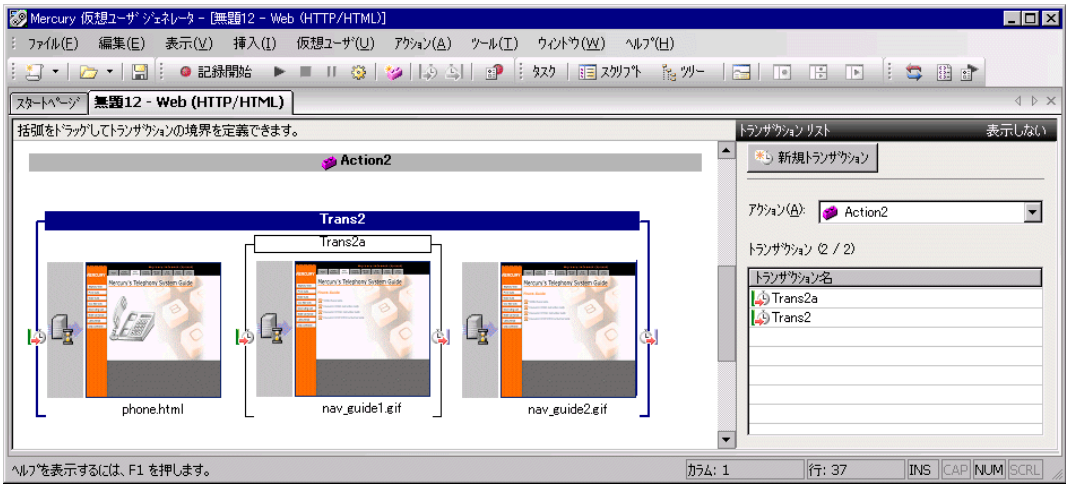
1 ツールバーの［**タスク**］をクリックして，タスク・リスト表示枠を開きます。

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

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

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

次の例では，「**Action2**」が選択されています。



サムネイルを使った作業

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

サムネイルを大きなサイズで表示するには，次の手順を実行します。

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

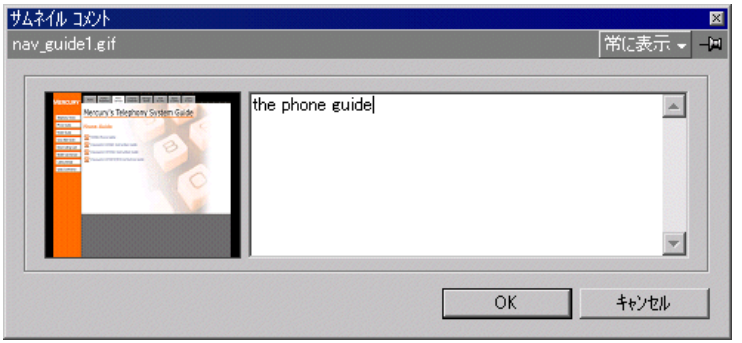
スナップショット名を変更するには，次の手順を実行します。

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

2 必要なテキストを入力します。

スナップショットに注釈を追加するには，次の手順を実行します。

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

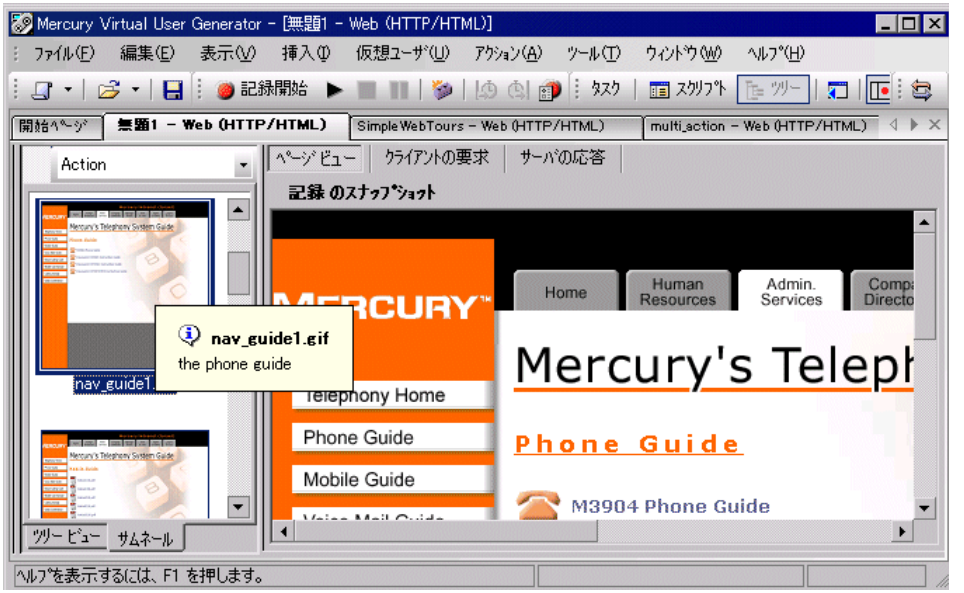


2 ［サムネイル コメント］ダイアログ・ボックスの右側の表示枠にテキストを入力します。

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

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

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



VuGen を使った仮想ユーザ・スクリプトの実行

仮想ユーザ・スクリプトを使ってテストを実行したりアプリケーションを監視したりするには，スクリプトを LoadRunner のシナリオ，ビジネス・プロセス・モニタのプロファイル，または Tuning Console のセッション・ステップに組み込む必要があります。これを行う前に，VuGen でスクリプトを実行してその動作を検証します。詳細については，第 15 章「スタンドアロン・モードでの仮想ユーザ・スクリプトの実行」を参照してください。

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

仮想ユーザ・スクリプトを実行する前に，実行環境の設定を変更できます。これらの設定の中には，仮想ユーザの反復実行回数や，スクリプトの実行時に仮想ユーザに適用される反復のペースおよび思考遅延時間も含まれます。実行環境の設定の詳細については，第 13 章「実行環境の設定」を参照してください。

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

上級ユーザは，記録されたスクリプトをコンパイルして実行プログラムを作成することもできます。詳細については，第 8 章「仮想ユーザ・スクリプトの拡張」を参照してください。

VuGen のコードについて

仮想ユーザ・スクリプトを記録すると，VuGen は仮想ユーザ関数を生成してスクリプトに挿入します。仮想ユーザ関数には，次の 2 つのタイプがあります。

- 一般仮想ユーザ関数
- プロトコル固有の仮想ユーザ関数

一般仮想ユーザ関数と**プロトコル固有の**関数によって Mercury VuGen API が構成されます。仮想ユーザは，この API によってサーバと直接通信できるようになります。VuGen では新しいスクリプトを作成するときに，サポートしている全プロトコルのリストが表示されます。全仮想ユーザ関数の構文については，「**オンライン関数リファレンス**」（[**ヘルプ**] > [**関数リファレンス**]）を参照してください。

一般仮想ユーザ関数

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

- 仮想ユーザ，仮想ユーザ・グループ，およびホストに関する実行情報の取得。

- 仮想ユーザ・スクリプトへのトランザクションと同期ポイントの追加。例えば，**lr_start_transaction**（Java では **lr.start_transaction**）関数はトランザクションの開始を，**lr_end_transaction**（Java では **lr.end_transaction**）関数はトランザクションの終了を示します。詳細については第 8 章「仮想ユーザ・スクリプトの拡張」を参照してください。
- エラーや警告を示すメッセージの出力への送信。

LR 関数については，35 ページ「C 仮想ユーザ関数の使用方法」の一覧を参照してください。詳細については，「**オンライン関数リファレンス**」（[**ヘルプ**] ＞ [**関数リファレンス**]）を参照してください。

プロトコル固有の仮想ユーザ関数

一般仮想ユーザ関数に加えて，VuGen は記録中にプロトコル固有の関数を生成して，仮想ユーザ・スクリプトに挿入します。

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

標準では，VuGen の自動スクリプト・ジェネレータは，ほとんどのプロトコルで C 言語の仮想ユーザ・スクリプトを生成し，Corba-Java/Rmi-Java 仮想ユーザには Java の仮想ユーザ・スクリプトを生成します。VuGen に，Visual Basic や Javascript でコードを生成させることができます。詳細については，第 5 章「スクリプト生成オプションの設定」を参照してください。

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

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

```
#include "as_web.h"


Action1()
{

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

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

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

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

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

仮想ユーザ・スクリプトで C 言語の関数を使用する方法の詳細については，第 8 章「仮想ユーザ・スクリプトの拡張」を参照してください。Java スクリプトを変更する方法の詳細については，第 38 章「Java 仮想ユーザ・スクリプトのプログラミング」を参照してください。

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

C 仮想ユーザ関数の使用方法

任意の仮想ユーザ・スクリプトに C 仮想ユーザ関数を追加して，スクリプトを拡張できます。VuGen を使って記録しているときに生成される一般仮想ユーザ関数は 2 ～ 3 個です。残りの関数は，必要に応じて手作業でスクリプトにプログラミングしなければなりません。一般仮想ユーザ関数の詳細については，第 8 章「仮想ユーザ・スクリプトの拡張」を参照してください。

次の一覧に，ANSI C スクリプト用の一般 API 関数を示します。この中には，Java，VB，および GUI を除くすべてのプロトコルが含まれています。Java 関数の一覧については，第 38 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。VB 関数の一覧については，「**オンライン関数リファレンス**」（［**ヘルプ**］＞［**関数リファレンス**］）を参照してください。

トランザクション関数

lr_end_sub_transaction	パフォーマンス分析で使用するサブトランザクションの終わりを示します。
lr_end_transaction	トランザクションの終了を示します。
lr_end_transaction_instance	パフォーマンスを分析するため，トランザクションのインスタンスの最後に印を付けます。
lr_fail_trans_with_error	開いているトランザクションのステータスを LR_FAIL に設定して，エラー・メッセージを送信します。
lr_get_trans_instance_duration	トランザクション・インスタンスをハンドルで指定し，その経過時間を取得します。
lr_get_trans_instance_wasted_time	トランザクション・インスタンスをハンドルで指定し，その浪費時間を取得します。
lr_get_transaction_duration	トランザクションを名前で指定し，その経過時間を取得します。
lr_get_transaction_think_time	トランザクションを名前で指定し，その思考遅延時間を取得します。

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

コマンド・ライン解析関数

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

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

情報関数

lr_user_data_point	ユーザ定義データのサンプル値を記録します。
lr_whoami	仮想ユーザに関する情報を仮想ユーザ・スクリプトに返します。Business Availability Center では使用できません。
lr_get_host_name	仮想ユーザ・スクリプトを実行しているホストの名前を返します。
lr_get_master_host_name	LoadRunner コントローラまたは Tuning Console を実行しているマシンの名前を返します。Business Availability Center では使用できません。

文字列関数

lr_eval_string	パラメータを現在の値で置き換えます。
lr_save_string	NULL で終わる文字列をパラメータに保存します。
lr_save_var	可変長文字列をパラメータに保存します。
lr_save_datetime	現在の日付と時刻をパラメータに保存します。
lr _advance_param	使用可能な次のパラメータに進みます。
lr _decrypt	暗号化された文字列を復号します。
lr_eval_string_ext	パラメータ・データを格納しているバッファを指すポインタを取得します。
lr_eval_string_ext_free	**lr_eval_string_ext** によって割り当てられたポインタを解放します。
lr_save_searched_string	バッファ内で文字列の特定の出現を検索し，当該文字列を基準とする相対的な指定範囲のバッファの一部をパラメータに保存します。

メッセージ関数

lr_debug_message	デバッグ・メッセージを［出力］ウィンドウまたはビジネス・プロセス・モニタのログ・ファイルに送ります。
lr_error_message	エラー・メッセージを［出力］ウィンドウまたは Business Process Monitor ログ・ファイルに送ります。
lr_get_debug_message	現在のメッセージ・クラスを取得します。
lr_log_message	メッセージをログ・ファイルに送ります。
lr_output_message	メッセージを［出力］ウィンドウまたは Business Process Monitor ログ・ファイルに送ります。
lr_set_debug_message	デバッグ・メッセージ・クラスを設定します。
lr_vuser_status_message	形式を整えた出力を生成し，コントローラまたはコンソールの仮想ユーザ・ステータス領域に出力します。Business Availability Center では使用できません。
lr_message	メッセージを，仮想ユーザ・ログと，［出力］ウィンドウまたは Business Process Monitor ログ・ファイルに送ります。

ランタイム関数

lr_load_dll	外部の DLL をロードします。
lr_peek_events	仮想ユーザ・スクリプトが一時停止できるポイントを示します。
lr_think_time	スクリプトの実行を一時停止して思考遅延時間（実際のユーザが動作の間に考えるために動作を停止する時間）をエミュレートします。
lr_continue_on_error	エラー処理メソッドを指定します。
lr_rendezvous	仮想ユーザ・スクリプトにランデブー・ポイントを設定します。Business Availability Center では使用できません。

関数のヘルプ

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

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

オンライン関数リファレンス

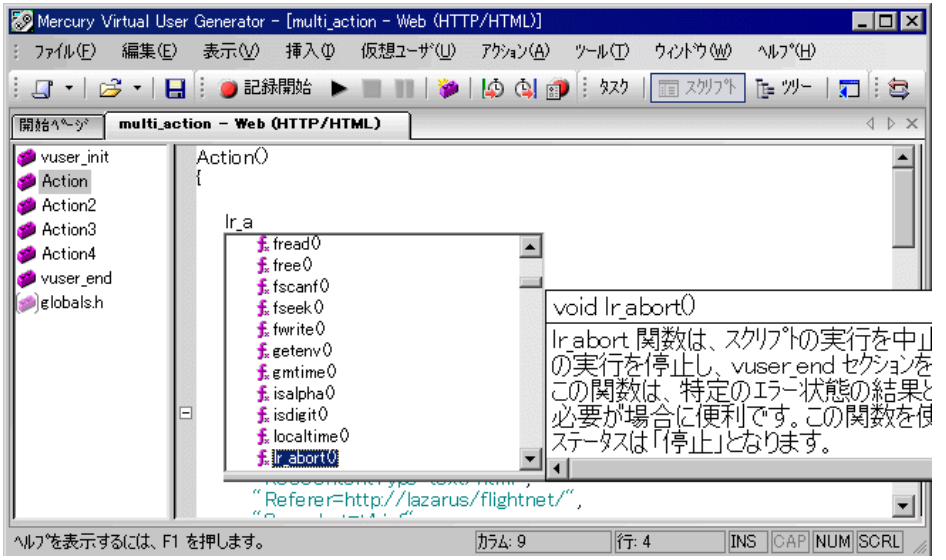
「**オンライン関数リファレンス**」には，すべての VuGen 関数についての詳しい構文情報や関数の例が含まれています。関数の使用例も含まれています。関数名の検索が可能なほか，カテゴリまたはアルファベット順のリストから参照できます。

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

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

単語の補完

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

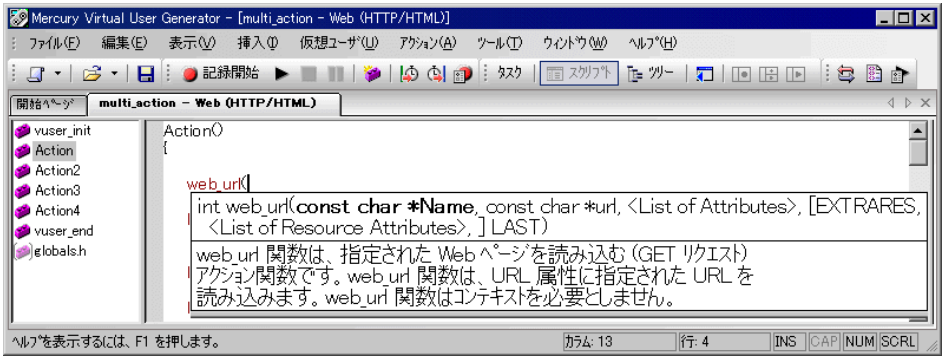


表示された関数を使用するには，その項目を選択するか，目的の項目までスクロールしてから選択します。選択した関数がカーソルの位置に挿入されます。リスト・ボックスを閉じるには，Esc キーを押します。

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

関数構文の表示

VuGen の IntelliSense には，このほかにも**関数構文の自動表示**機能があります。関数の開始括弧を入力すると，関数の構文，引数，プロトタイプ，および簡単な説明が自動的に表示されます。



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

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

ヘッダー・ファイル

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

ほとんどの場合，ヘッダー・ファイル名はプロトコルの接頭辞に対応しています。例えば，**lrd** という接頭辞で始まるデータベース関数のリストは，**lrd.h** ファイルにあります。

次の表に，使用頻度の高いプロトコルと，対応するヘッダー・ファイルを示します。

プロトコル	ファイル
Citrix	**ctrxfuncs.h**
COM/DCOM	**lrc.h**
データベース	**lrd.h**
FTP	**mic_ftp.h**
一般的な C 関数	**lrun.h**
IMAP	**mic_imap.h**
LDAP	**mic_mldap.h**
MAPI	**mic_mapi.h**
Oracle NCA	**orafuncs.h**
POP3	**mic_pop3.h**
RealPlayer	**lreal.h**
SAPGUI	**as_sapgui.h**
Siebel	**lrdsiebel.h**
SMTP	**mic_smtp.h**
ターミナル・エミュレータ	**lrrte.h**
Tuxedo	**lrt.h**
WAP	**as_wap.h**
Web	**as_web.h**
Web サービス	**wssoap.h**
Windows Sockets	**lrs.h**

第 3 章

VuGen ワークフローの表示

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

本章では，次の項目について説明します。

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

以降の情報は，全タイプの仮想ユーザ・スクリプトを対象とします。

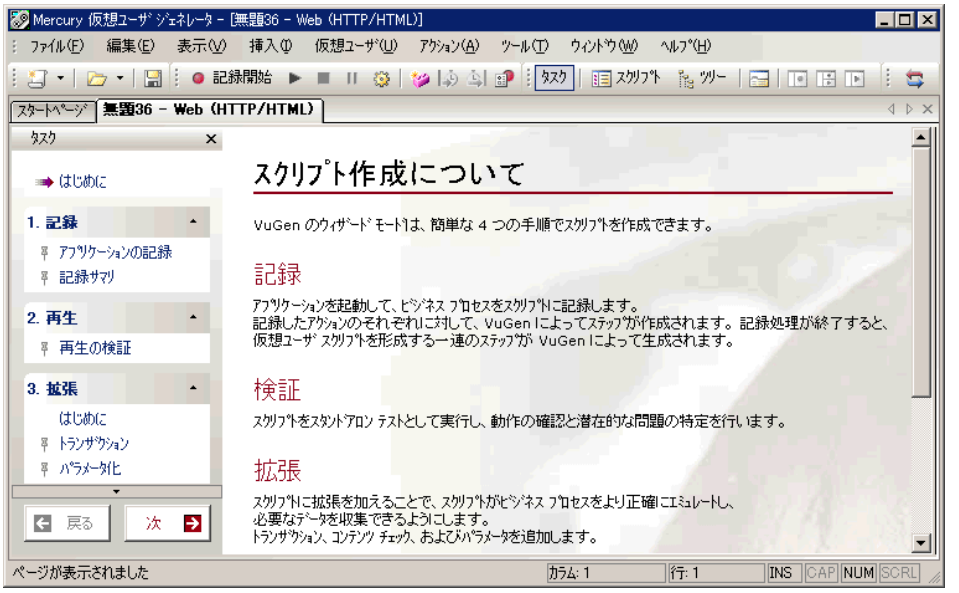
VuGen ワークフローの表示について

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

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

タスク表示枠の表示

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



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

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

タスクの多くにはサブ・タスクがあります。次の表にサブ・タスクを表示します。

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

ステップの記録

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

アプリケーションの記録

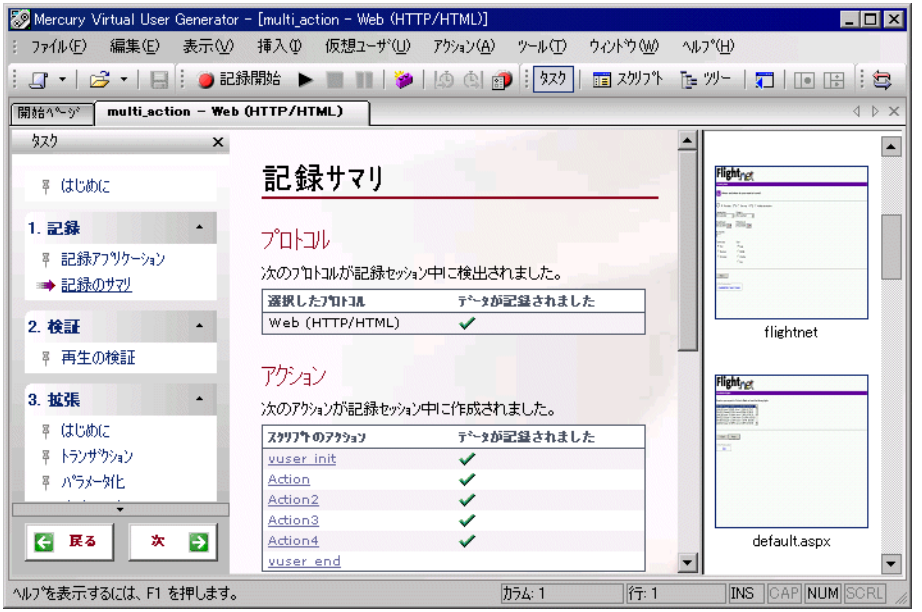
このウィザード・ステップでは記録プロセスを概説します。次のセクションが含まれます。

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

記録サマリ

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

このステップでは，記録されたスナップショットのサムネイルも提供されます。



スクリプトの検証

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

再生の検証

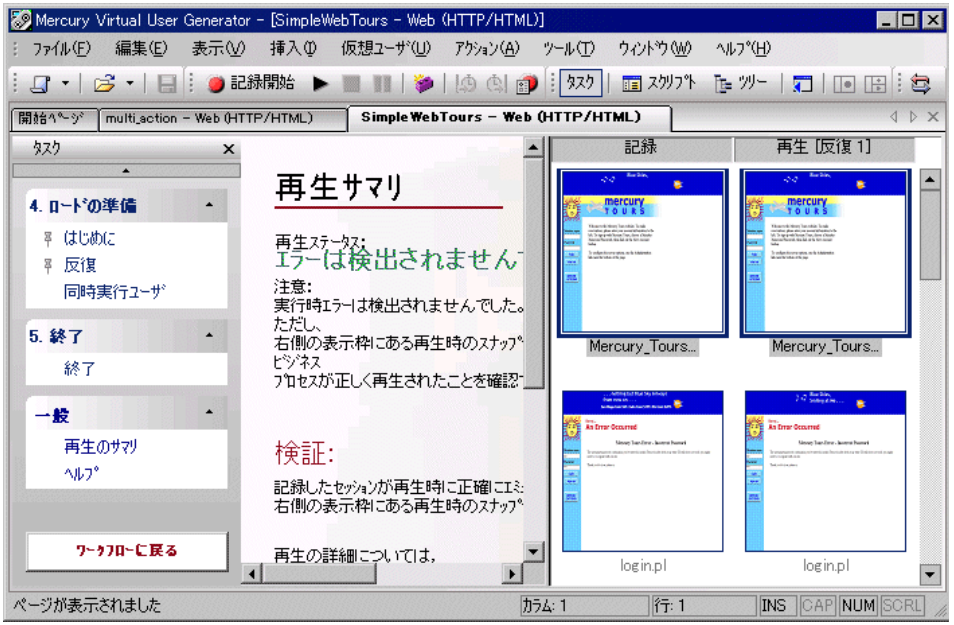
このウィザード・ステップでは検証を概説します。次のセクションが含まれます。

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

再生のサマリ

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

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



注：反復が複数の場合，［再生サマリ］ウィンドウには最後の反復の再生のサムネイルが表示されます。特定の反復のサムネイルを表示するには，［**表示**］＞［**ツリー ビュー**］を選択して，ツリー・ビューに切り替えます。［**表示**］＞［**スナップショット**］＞［**反復の選択**］を選択し，希望の反復を選びます。

スクリプトの拡張

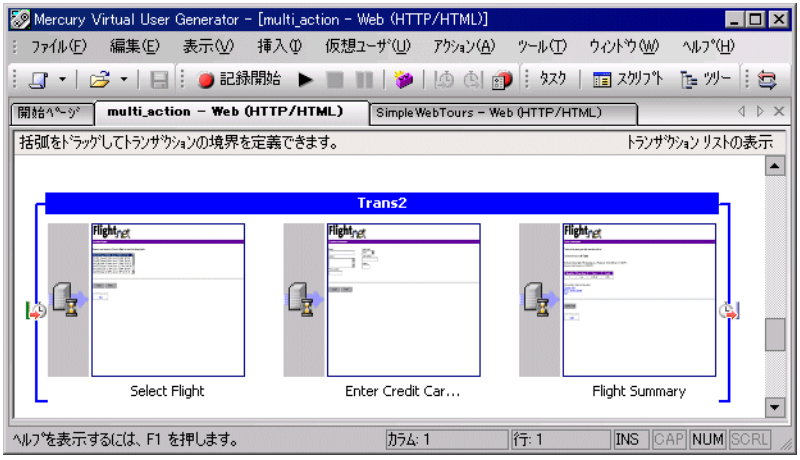
［拡張］セクションには，トランザクション，パラメータ化，コンテンツ・チェックという 3 つの主要なステップがあります。

トランザクション

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

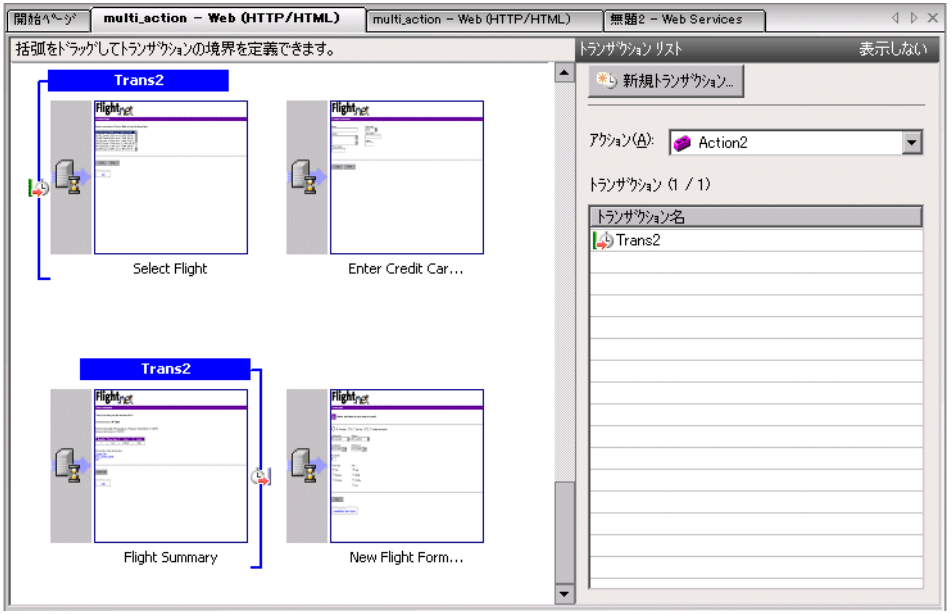
標準設定では，スクリプトの主要なステップのサムネイルのみが表示されます。スクリプト内のすべてのステップのサムネイルを表示するには，［**表示**］＞［**すべてのサムネイルを表示**］を選択します。

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



トランザクション・リストでの作業

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



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

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

標準設定では，トランザクション・リストには，スクリプトの主要なステップに対するサーバの応答を測定する，エラーの生じないトランザクションのみが表示されます。次の項目は表示されません。

- 主要でないステップ
- クライアント側のトランザクション
- エラーの生じたトランザクション

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

トランザクション（2/4）。

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

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

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

トランザクション・エディタでのトランザクションの定義

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

トランザクションを定義するには，次の手順を実行します。

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

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

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

---

3 単一のステップをトランザクションとして印を付けるには，サムネイルをクリックして右クリック・メニューから［**新規シングル ステップ トランザクション**］を選択します。新しいトランザクションの名前を入力するダイアログ・ボックスが表示されます。後からトランザクションを拡張するには，トランザクションの括弧をドラッグしてトランザクションに追加のステップを含めます。

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

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

［**ステップ 1/3 新規トランザクションの開始点を選択します。**］

トランザクションの最初のステップのサムネイルをクリックします。

［**ステップ 2/3 新規トランザクションの終了点を選択します。**］

トランザクションの最後のステップのサムネイルをクリックします。

［**ステップ 3/3 新しいトランザクションの名前を指定します。**］

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

トランザクションを完了するには，Enter キーを押します。

前述の手順の最中にトランザクションを終了するには，Esc キーを押します。

6 トランザクションの開始点を変更するには，トランザクションの左大括弧を新しい場所にドラッグします。トランザクションの終了点を変更するには，トランザクションの右大括弧を新しい場所にドラッグします。

7 トランザクションを追加するには，前述のステップを繰り返します。

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

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

トランザクション・エディタのガイドライン

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

- トランザクションは単一のアクション内で開始し終了しなければなりません。トランザクションは複数のアクションにまたがって拡張できません。
- トランザクション名は，複数のアクションにおいてでも，スクリプト内で一意でなくてはなりません。
- トランザクションの開始点を変更するには，トランザクションの左大括弧を新しい場所にドラッグします。トランザクションの終了点を変更するには，トランザクションの右大括弧を新しい場所にドラッグします。
- トランザクションの追加，名前の変更，削除には，右クリック・メニューを使用します。
- 既存のトランザクション内でトランザクションを作成できます。これらは「**入れ子**」のトランザクションと呼ばれます。

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

パラメータ化

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

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

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

コンテンツ・チェック

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

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

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

ロードの準備

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

- 反復
- 同時ユーザ

反復

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

反復回数を設定するには，次の手順を実行します。

1. ［実行環境の設定］（F4）を開きます。
2. ［実行論理］ノードを選択します。
3. 反復回数を指定します。

同時ユーザ

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

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

スクリプトの完了

ワークフロー・ウィザードの最終ステップは［**完了**］です。

［完了］には，次のセクションが含まれます。

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

第 4 章

VuGen を使った記録

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

本章では，次の項目について説明します。

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

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

VuGen を使った記録について

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

作成する各仮想ユーザ・スクリプトには，少なくとも次の 3 つのセクションがあります。**vuser_init**，1 つまたは複数の **Actions**，および **vuser_end** です。記録中に，VuGen によって記録される関数の挿入先となるスクリプトのセクションを選択できます。一般的には，サーバへのログインを **vuser_init** セクションに，クライアントの動作を **Actions** セクションに，ログオフの手順を **vuser_end** セクションに記録します。

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

記録中，スクリプトにトランザクション，コメント，ランデブー・ポイントを挿入できます。詳細については，第 8 章「仮想ユーザ・スクリプトの拡張」を参照してください。

仮想ユーザ・スクリプトのセクション

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

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

仮想ユーザ・スクリプトの反復を複数回実行するときは，スクリプトの **Actions** セクションだけが繰り返されます。**vuser_init** セクションと **vuser_end** セクションは繰り返されません。反復回数の設定の詳細については，第 13 章「実行環境の設定」を参照してください。

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

Java クラスを使用する仮想ユーザ・スクリプトの場合には，すべてのコードを **Actions** クラスに置きます。Actions クラスには，**init**，**action**，**end** の 3 つのメソッドが含まれています。これらのメソッドは他のプロトコルの場合に作成されるスクリプトの各セクションに対応します。つまり，初期化ルーチンは **init** メソッドに，クライアントの動作は **action** メソッドに，ログオフの手順は **end** メソッドに，それぞれ挿入します。詳細については，第 38 章「Java 仮想ユーザ・スクリプトのプログラミング」を参照してください。

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

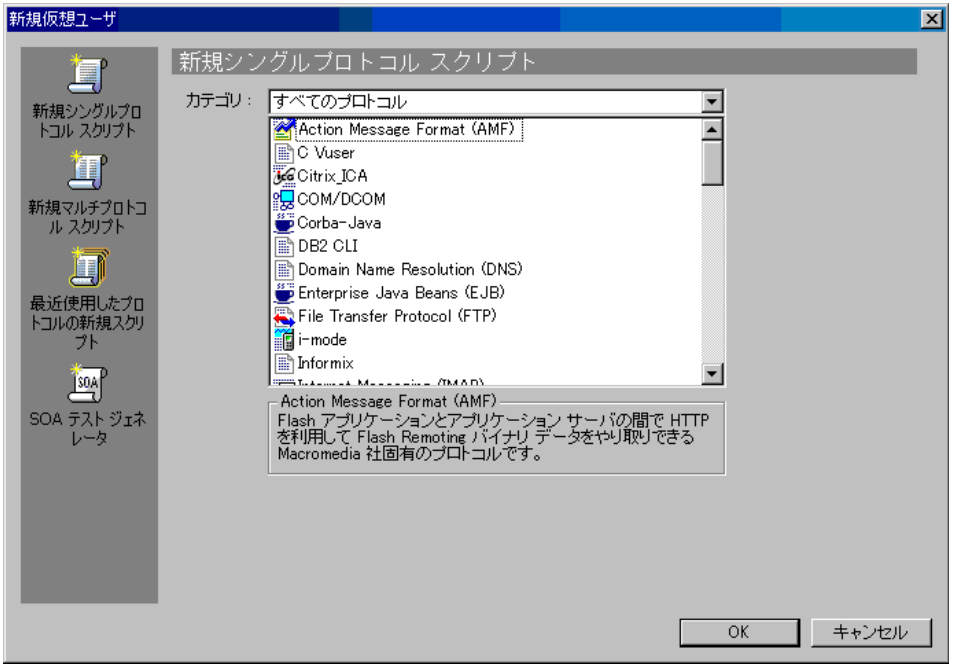
注：Oracle DB のトランザクション・ブレークダウンは，vuser_init セクションで記録されたアクションには使用できません。

仮想ユーザ・スクリプトの新規作成

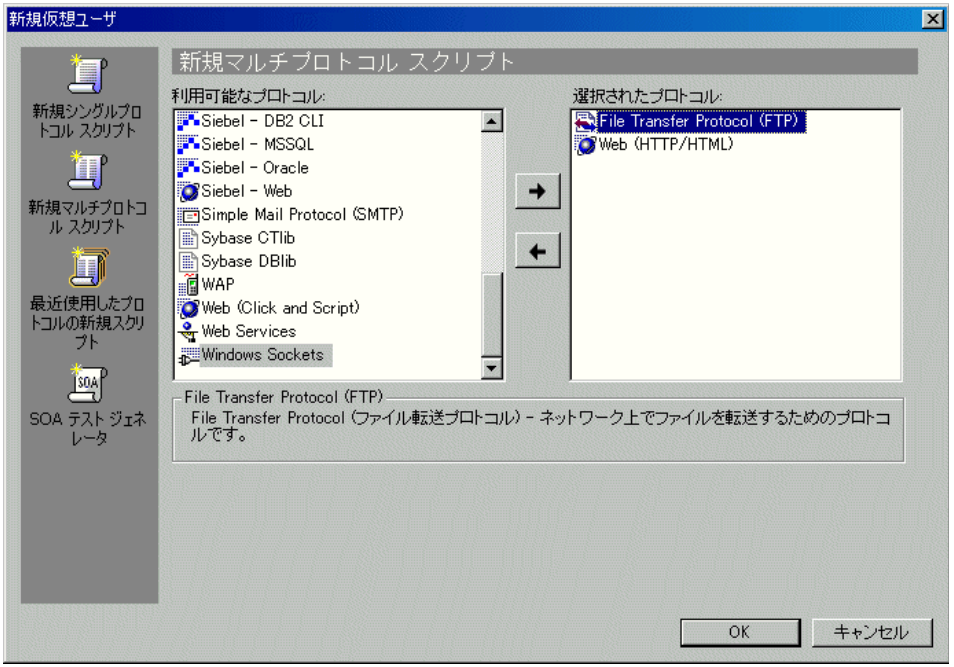
VuGen では，シングル・プロトコル・モードまたはマルチ・プロトコル・モードで記録することによって，スクリプトを新規作成できます。また，SOA（サービス指向アーキテクチャ）環境でスクリプトを作成するためのソリューションも提供されます。

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

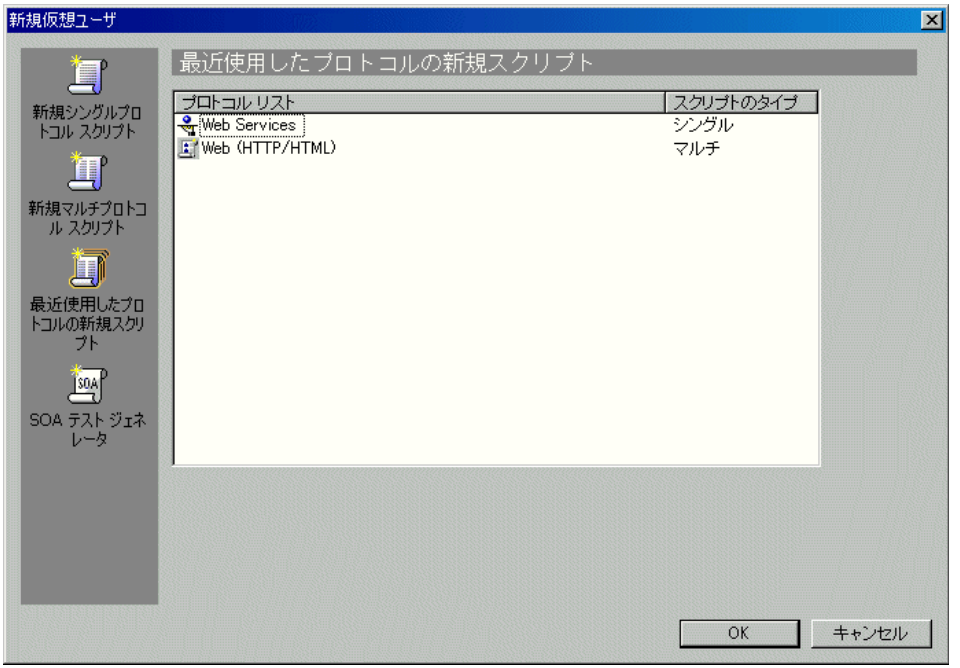
［**新規シングル プロトコル スクリプト**］：シングル・プロトコルの仮想ユーザ・スクリプトを作成します。シングル・プロトコルのスクリプトを作成するには，［カテゴリ］リストからカテゴリを選び（64 ページ「仮想ユーザ・カテゴリの選択」を参照），そのカテゴリ配下のプロトコル・リストでプロトコルを選択します。



［**新規マルチ プロトコル スクリプト**］：マルチ・プロトコルの仮想ユーザ・スクリプトを作成します。VuGen には使用可能なすべてのプロトコルが表示され，どのプロトコルを記録するかを指定できます。マルチ・プロトコルのスクリプトを作成するには，［利用可能なプロトコル］セクションでプロトコルを選択し，右矢印をクリックしてプロトコルを［選択されたプロトコル］セクションに移動します。



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



シングル・プロトコル・モードで記録する場合，VuGen は指定したプロトコルだけを記録します。マルチ・プロトコル・モードで記録する場合，VuGen はアクションを複数のプロトコルで記録します。マルチ・プロトコル・スクリプトは，次のプロトコルをサポートします。

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

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

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

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

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

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

すべての Java 言語仮想ユーザ（CORBA，RMI，Jacada，EJB）の記録の詳細については，第 26 章「Java 言語仮想ユーザ・スクリプトの記録」，または各プロトコルを解説している章を参照してください。

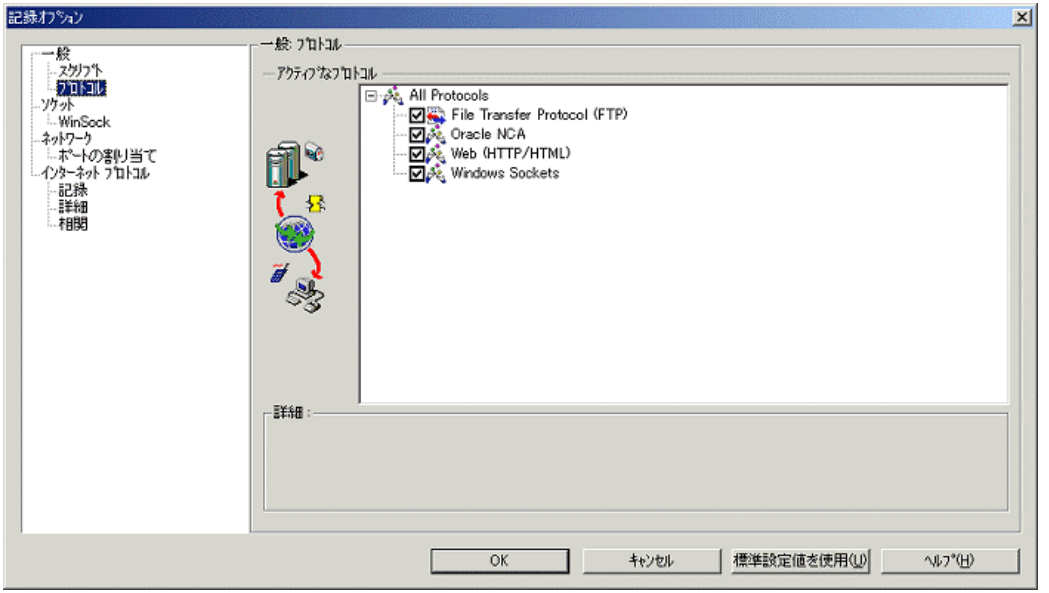
［**SOA テスト ジェネレータ**］: SOA（サービス指向アーキテクチャ）システムでは、配備の前にアプリケーションとサービスの安定性をテストすることは重要です。Mercury SOA テスト・ジェネレータは，サービスをテストするスクリプトの作成手順を導いてくれます。SOA ソリューションの詳細については，281 ページ「SOA および Web サービスのテスト」を参照してください。

プロトコルの追加と削除

マルチ・プロトコル・セッションを記録する前に，VuGen では，記録セッション中にコードを生成するプロトコルのリストを修正できます。スクリプトの作成時に特定のプロトコルを指定した場合は，プロトコルの記録オプションを使用してそれらを有効または無効にできます。

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

次回の記録セッションで記録するプロトコルについて，それらの横にあるチェック・ボックスを選択します。次回の記録セッションで記録しないプロトコルについて，それらの横にあるチェック・ボックスをクリアします。



AMF および Web プロトコルのプロトコル・オプションの設定の詳細については，687 ページ「AMF 記録モードの設定」を参照してください。

仮想ユーザ・カテゴリの選択

仮想ユーザのタイプは次のカテゴリに分類されます。

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

- ［**e ビジネス**］：AMF，FTP，LDAP，Palm，Microsoft .NET，Web(Click and Script)，Web（HTTP/HTML），Web サービス，Web/Winsocket Dual プロトコル用。
- ［**Enterprise Java Beans**］：EJB テストおよび RMI-Java プロトコル用。
- ［**ERP/CRM**］：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）に対応します。
- ［**メール サービス**］：Internet Messaging（IMAP），MS Exchange（MAPI），POP3，SMTP プロトコル用。
- ［**ミドルウェア**］：Jacada，Tuxedo（6，7）プロトコル用。
- ［**ストリーミング**］：Real と Media Player（MMS）プロトコル用。
- ［**ワイヤレス**］：i モード，マルチメディア・メッセージング・サービス（MMS），VoiceXML，WAP プロトコル用。

スクリプトの新規作成

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

新しい仮想ユーザ・スクリプトを作成するには，次の手順を実行します。

1 ［**スタート**］＞［**プログラム**］＞［**Mercury ＜アプリケーション名＞**］＞［**Application**］＞［**Virtual User Generator**］を選択して，VuGen を起動します。

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

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

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

注：特定の Oracle NCA アプリケーションを記録する場合は，（**Web（HTTP/HTML）**ではなく）**Oracle NCA** を選択します。詳細については，第 63 章「Oracle NCA 仮想ユーザ・スクリプトの作成」を参照してください。

---

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

既存のスクリプトを開く

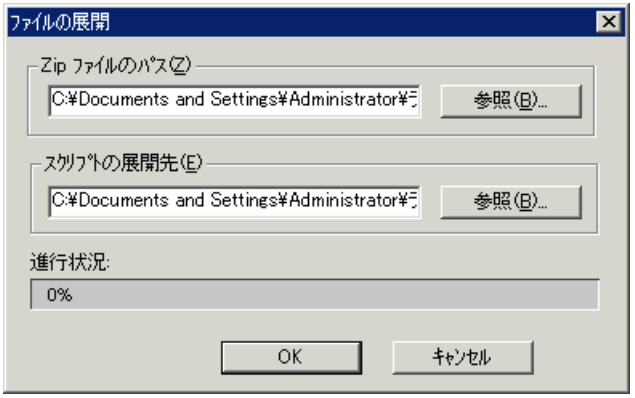
ローカル・マシンまたはネットワーク上にすでにスクリプトがある場合は，そのスクリプトを変更して追加のアクションを記録できます。

既存のスクリプトを開くには，次の手順を実行します。

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

2 Quality Center リポジトリからファイルを開く方法については（LoadRunner の場合のみ），255 ページ「Quality Center プロジェクトからスクリプトを開く」を参照してください。

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



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

アプリケーションの記録

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

記録を開始するには，次の手順を実行します。

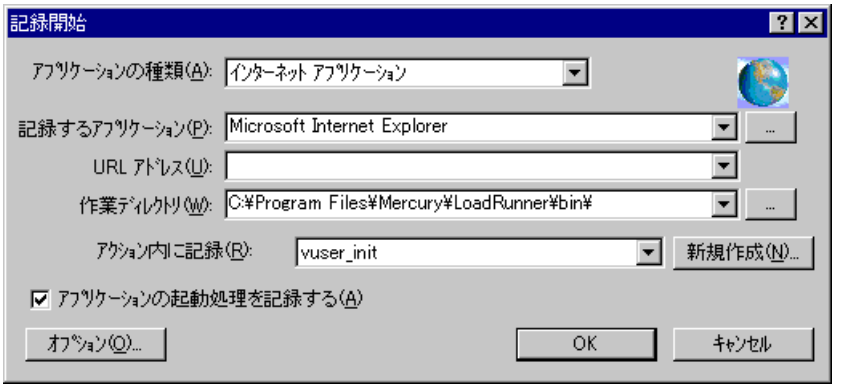
1 ［記録開始］ダイアログ・ボックスが開かない場合は，［**記録開始**］ボタンをクリックします。［記録開始］ダイアログ・ボックスが開きます。このダイアログ・ボックスはプロトコルごとに多少異なります。

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

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

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

4 インターネット・アプリケーションについて，適切な情報を入力します。

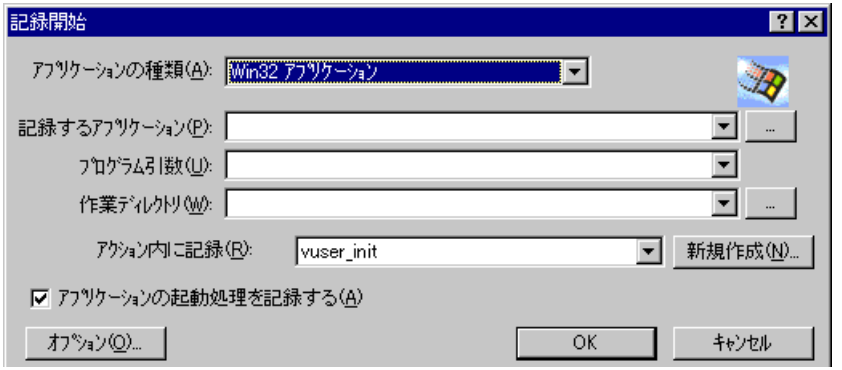


➤ [**記録するアプリケーション**]：記録するブラウザまたはインターネット・アプリケーションを選択します。

➤ [**URL アドレス**]：開始する URL アドレスを指定します。

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

5 Win32 アプリケーションについて，適切な情報を入力します。

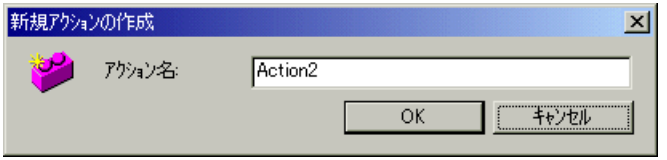


➤ [**記録するアプリケーション**]：記録する Win 32 アプリケーションを入力します。

- ［**プログラム引数**］：上記で指定した実行ファイルのコマンド・ライン引数を指定します。たとえば，**plus32.exe** にコマンド・ライン・オプション peter@neptune を指定した場合，**plus32.exe** を起動すると，ユーザ **Peter** がサーバ **Neptune** に接続されます。

- ［**作業ディレクトリ**］：作業ディレクトリを指定する必要があるアプリケーションの場合は，ここで指定します。

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



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

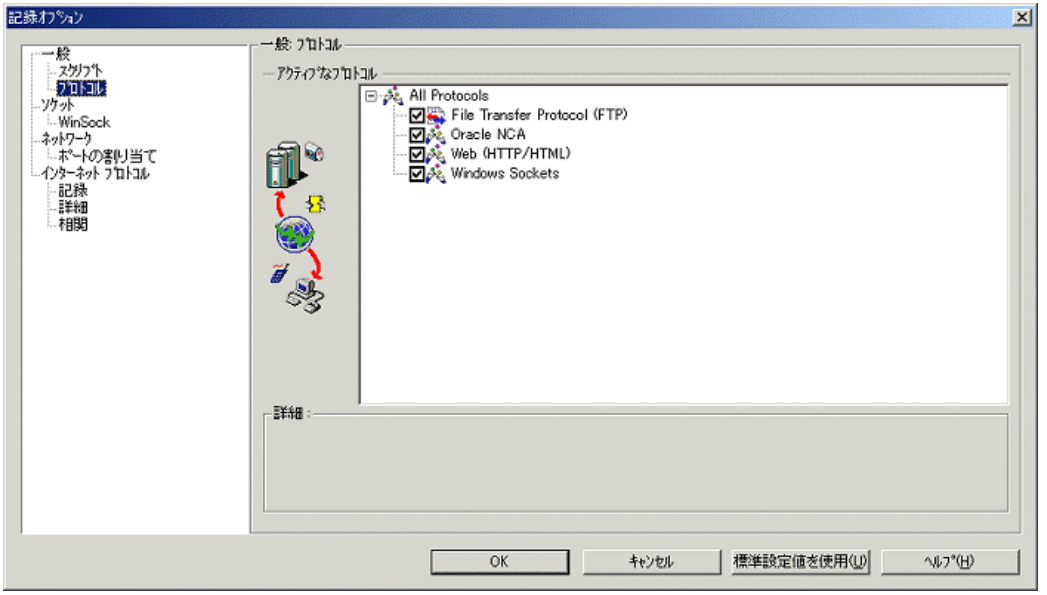
- マルチ・アクションの場合。起動が必要なアクションは 1 つだけです。
- アプリケーションで特定の位置まで移動し，その位置から記録を開始する場合。
- 既存のスクリプトに記録する場合。

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

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

10 ポート情報を指定するには，［**ポートの割り当て**］ノードをクリックします。これは，非標準ポートの SSL アプリケーションを記録する場合に有用です。ポートのリストを確認します。使用するポートがリストにない場合は，［ポートの割り当て］オプションを使用して情報を指定できます。詳細については，第 7 章「ポートの割り当て設定」を参照してください。

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



これで，記録開始の準備が整います。

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

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

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

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

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

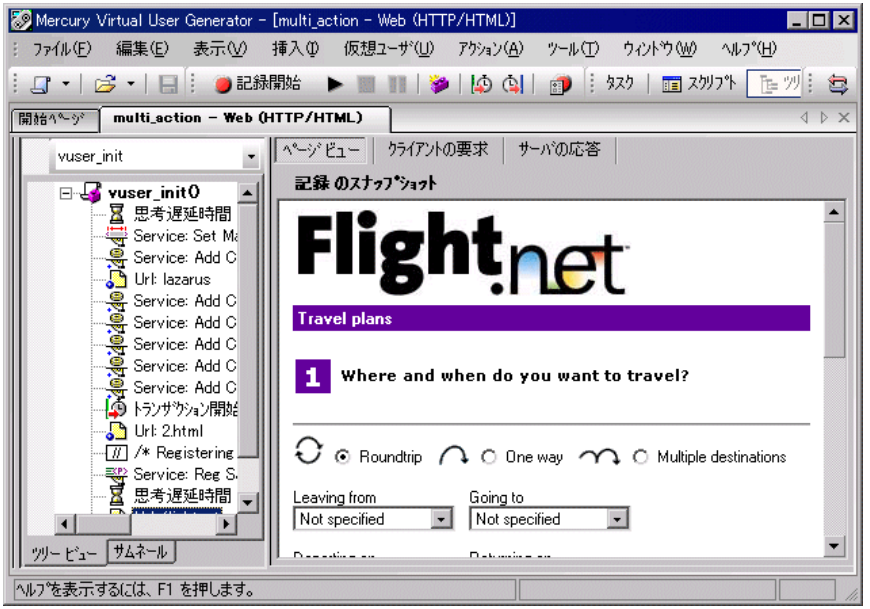
記録セッションの終了と保存

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

記録を終了するには，次の手順を実行します。

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

2 フローティング・ツールバーの［**停止**］ボタンをクリックします。VuGen エディタに，記録されたすべてのステップが表示されます（スクリプト・ビューで作業を始めている場合は，記録された関数が表示されます）。



3 ［**上書き保存**］ボタンをクリックして，記録されたセッションを保存します。［テストを保存］ダイアログ・ボックスが表示されます（新規仮想ユーザ・スクリプトの場合のみ）。スクリプト名を指定します。**注**：スクリプトに **init**，**run**，**end** という名前は付けないでください。これらの名前は VuGen によって使用されます。

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

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

［**圧縮率**］を［最高（最低速）］，［標準］，［高速］，［超高速］，または［なし］から選択します。圧縮率が高いほど，VuGen によるアーカイブ作成に時間がかかります。

［**OK**］をクリックします。

5 Zip ファイルを作成して電子メールの添付ファイルとして送信するには，［**ファイル**］＞［**Zip 操作**］＞［**Zip して電子メールで送信**］を選択します。

［**OK**］をクリックします。電子メールの作成フォームが表示されます。

電子メール・アドレスを入力してメールを送信します。

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

- ［**URL**］ボックスに，Performance Center の URL を入力します。
- ［**Project**］ボックスで，プロジェクト名を選択します。
- サーバにログオンするためのユーザ名とパスワードを入力します。
- ［**OK**］をクリックして設定を適用し，ダイアログ・ボックスを閉じます。

記録ログの表示

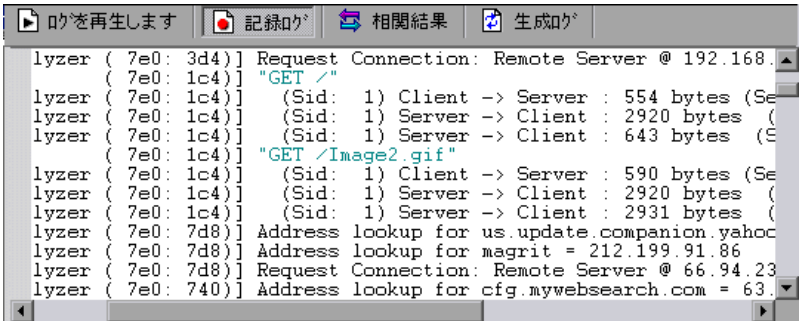
記録終了後に，**vuser_init**，**Actions**，および **vuser_end** セクションの内容を VuGen スクリプト・エディタに表示できます。アクションを表示するには，左側の表示枠でアクション名を選択します。

記録中，VuGen は一連の設定ファイル，データ・ファイル，ソースコード・ファイルを作成します。これらのファイルには，仮想ユーザの実行時の情報および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。

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

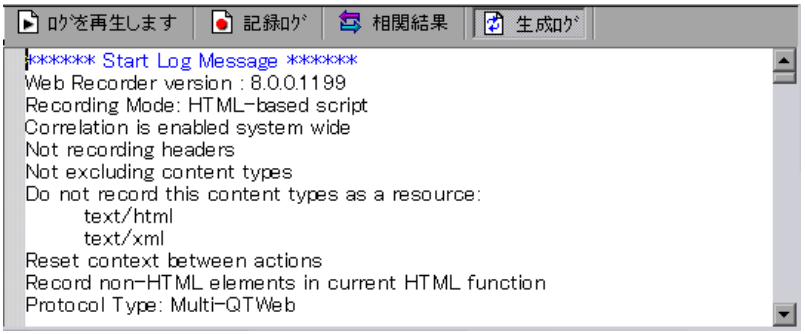
記録ログ

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



生成ログ

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



Zip ファイルの使用

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

Zip ファイルからのインポート

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

Zip ファイルからの作業

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

Zip ファイルへのエクスポート

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

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

また，［**圧縮率**］として，［最高（最低速）］，［標準］，［高速］，［超高速］，または［なし］を選択することもできます。圧縮率が高いほど，VuGen によるアーカイブ作成に時間がかかります。したがって，［**最高（最低速）**］の圧縮オプションが最も低速になります。

Zip の作成と電子メール送信

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

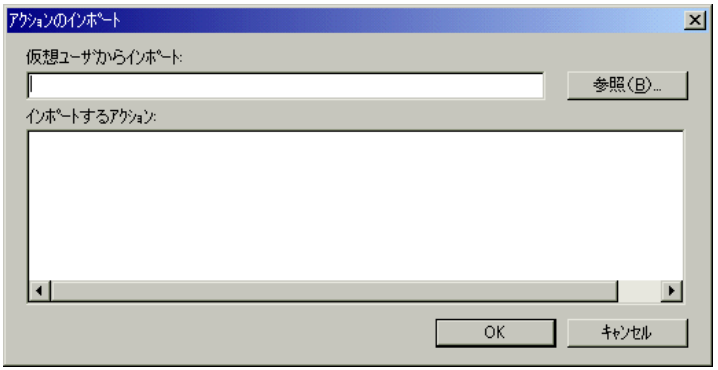
アクションのインポート

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

- ［**仮想ユーザからインポート**］：インポート元の仮想ユーザ・スクリプトを入力または参照します。
- ［**インポートするアクション**］：インポートするアクションを選択します。

現在のスクリプトへアクションをインポートするには，次の手順を実行します。

1 ［**アクション**］＞［**仮想ユーザにアクションをインポート**］を選択します。［アクションのインポート］ダイアログ・ボックスが表示されます。



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

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

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

設定］ノードを使用します。詳細については，第 13 章「実行環境の設定」を参照してください。

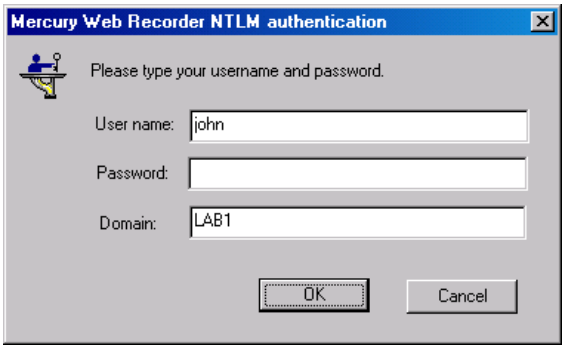
認証情報の提供

次の項の内容は，マルチ・プロトコルによる記録を対象としています。

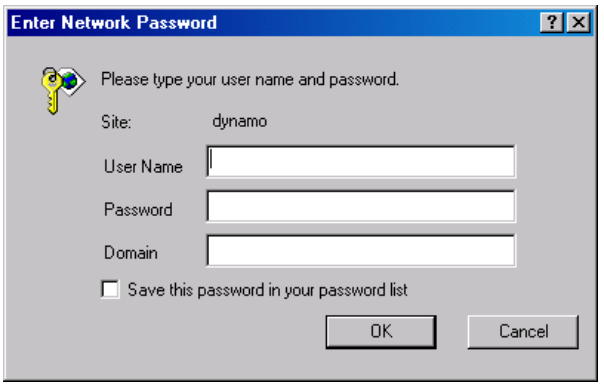
NTLM 認証を使用する Web セッションを記録する際，サーバでユーザ名やパスワードなどの詳細情報を入力しなければならない場合があります。

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

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



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



web_set_user 関数の生成

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

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

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

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

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

注 : パスワードを手作業で入力した場合は，スクリプト内にそのまま出現するため，セキュリティ上問題があります。パスワードを暗号化するには，パスワードを選択し，右クリック・メニューで［**文字列を暗号化**］を選択します。VuGen によって文字列が暗号化され，再生時にパスワードの復号に使用する **lr_decrypt** 関数が生成されます。文字列の暗号化の詳細については，120 ページ「テキストの暗号化」を参照してください。

仮想ユーザ・スクリプトの再生成

スクリプトを記録した後で，トランザクション，ランデブー・ポイント，メッセージ，コメントを追加してスクリプトを拡張できます。詳細については，第 8 章「仮想ユーザ・スクリプトの拡張」を参照してください。

さらに，スクリプトのパラメータ化や，変数の相関も可能です。詳細については，第 9 章「VuGen パラメータを使った作業」を参照してください。

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

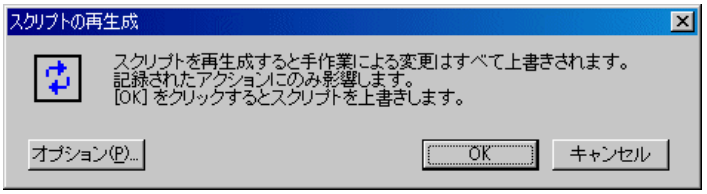
次のボタンが［スクリプトの再生成］ダイアログ・ボックスから使用できます。

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

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

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

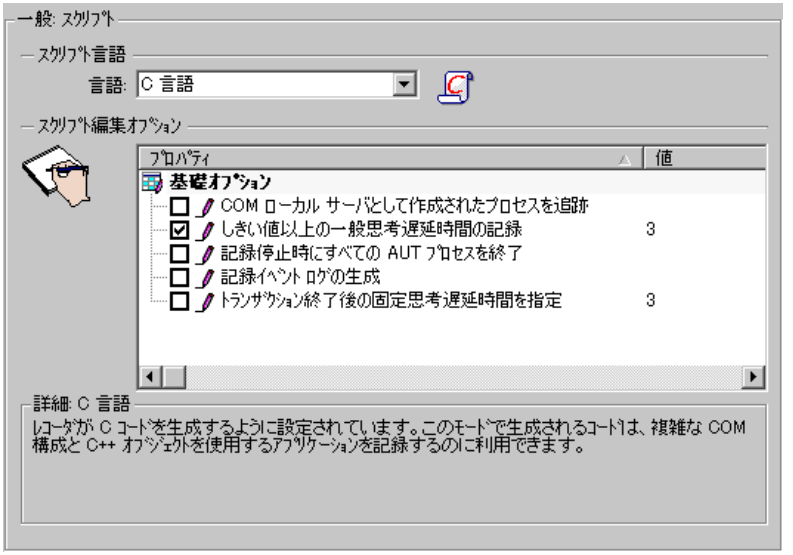
1 ［**ツール**］＞［**スクリプトの再生成**］を選択します。手作業で行ったすべての変更が上書きされることを示す警告が表示されます。



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

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

4 スクリプト・オプションを変更するには，［**一般：スクリプト**］ノードを選択し，適切なチェック・ボックスを選択またはクリアします。



第 5 章

スクリプト生成オプションの設定

VuGen を使ってスクリプトを記録する前に，使用するスクリプト記述言語を指定します。

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

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

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

スクリプト生成オプションの設定について

VuGen では，セッションを記録する前に，スクリプト生成言語を指定できます。

ヒント：ある言語でスクリプトを記録した後に，別の言語でそのスクリプトを再生成することもできます。詳細については，80 ページ「仮想ユーザ・スクリプトの再生成」を参照してください。

生成言語の選択後，スクリプトに何を含め，それをどのように生成するかをレコーダに指示する，言語固有の記録オプションを有効にできます。

記録するプロトコルの少なくとも 1 つにマルチ・プロトコルの機能がある場合は，**スクリプト・オプション**を使用できます。ただし，シングル・プロトコル・スクリプトで HTTP または WinSock を記録する場合は例外です。この場合，**スクリプト・オプション**は利用できません。

スクリプト言語の選択

セッションを記録するときに，VuGen は，標準設定ではユーザの操作をエミュレートするスクリプトを作成します。標準のスクリプト生成言語は C 言語または C# 言語（MS .NET の場合）です。FTP，COM/DCOM，およびメール・プロトコル（IMAP，POP 3，SMTP）の場合は，Visual Basic，VBScript，および JavaScript でもスクリプトを生成できます。

- ［**C 言語**］：複雑な COM 構成要素と C++ オブジェクトを使用するアプリケーション用。
- ［**C# 言語**］：複雑なアプリケーションと環境を使用するアプリケーション用（MS .NET プロトコルのみ）。
- ［**Visual Basic .NET 言語**］：VB .NET アプリケーション用。VB の機能をすべて使用できます。
- ［**Visual Basic for Applications**］：VB ベースのアプリケーション用。（VBScript とは異なり）VB の機能をすべて使用できます。
- ［**Visual Basic Scripting**］：VBScript ベースのアプリケーション用（ASP など）。
- ［**Java Scripting**］：JavaScript ベースのアプリケーション用（**js** ファイルやダイナミック HTML アプリケーションなど）。

記録セッションの後は，通常の C，C#，Visual Basic，VB Script，JavaScript コード，または制御フロー・ステートメントを使ってスクリプトに変更を加えられます。

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

手順の詳細については，88 ページ「記録オプションの設定」を参照してください。

基本オプションの適用

基本スクリプト・オプションは生成されたスクリプトの詳細レベルを制御します。一部のオプションは特定の言語に対してのみ使用できます。

- ［**Close all AUT processes when recording stops（記録停止時にすべての AUT プロセスを終了）**］：VuGen が記録を停止すると，すべての AUT（テスト対象アプリケーション）の処理が自動的に終了します（標準設定では無効）。
- ［**Declare primitives as locals（プリミティブをローカルとして宣言）**］：プリミティブ値の変数をクラス変数ではなくローカル変数として宣言します（C, C#，.NET のみ。標準設定では有効）。
- ［**Explicit variant declaration（明示的なバリアント宣言）**］：**ByRef** バリアントを処理するため，バリアント・タイプを明示的に宣言します（Visual Basic for Applications のみ。標準設定では有効）。
- ［**Generate fixed think time after end transaction（トランザクション終了後の固定思考遅延時間を指定）**］：トランザクション終了後，固定思考遅延時間を秒単位で追加します。このオプションを有効にする場合は，思考遅延時間の値を指定できます。標準設定は 3 秒です（標準設定では無効）。
- ［**Generate recorded events log（記録イベント ログの生成）**］：記録中に発生したすべてのイベントのログを生成します（標準設定では無効）。
- ［**Generate think time greater than threshold（しきい値以上の一般思考遅延時間の記録）**］：思考遅延時間のしきい値を使用します。記録された思考遅延時間がしきい値に満たない場合，VuGen は思考遅延時間ステートメントを生成しません。しきい値も指定します。標準設定の値は 3 です。思考遅延時間が 3 秒以内の場合は，VuGen は思考遅延時間のステートメントを生成しません。このオプションを無効にすると，VuGen は思考遅延時間を生成しません（標準設定では有効）。
- ［**Insert post-invocation info（起動後情報の挿入）**］：各メッセージ呼び出しの後に，その内容を表すログ・メッセージを挿入します（C 以外のみ。標準設定では有効）。

- ［**Insert output parameters values（出力パラメータ値の挿入）**］：各呼び出しの後に出力パラメータ値を挿入します（C，C#，.NET のみ。標準設定では無効）。

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

- ［**Maximum number of lines in action file（アクション ファイル内の最大行数）**］：アクションの行数が指定されたしきい値を超えた場合に新しいファイルを作成します。標準のしきい値は 60000 行です（C，C#，.NET のみ。標準設定では無効）。

- ［**Reuse variables for primitive return values（プリミティブの戻り値に変数を再利用）**］：メソッド呼び出しから受け取るプリミティブに同じ変数を再利用します。これは［**Declare primitives as locals**］設定に優先します（標準設定では有効）。

- ［**Replace long strings with parameter（長い文字列をパラメータで置換）**］：最大文字数を超える文字列をパラメータに保存します。このオプションの初期の最大長は 100 文字です。パラメータと文字列全体は，次の形式で，スクリプトのフォルダ内の **lr_strings.h** ファイルに保存されます。

  const char <paramName_uniqueID> ="string"

  このオプションにより，スクリプトが読みやすくなります。スクリプトのパフォーマンスには影響しません（標準設定では有効）。

- ［**Use full type names（完全な型名を使用）**］：新しい変数を宣言するときに完全な型名を使用します（C# および .NET のみ。標準設定では無効）。

- ［**Track processes created as COM local servers（COM ローカル サーバとして作成されたプロセスを追跡）**］：記録されたアプリケーションのサブプロセスの 1 つが COM ローカル・サーバとして作成されている場合は，そのアプリケーションの動作を追跡します（C および COM のみ。標準設定では有効）。

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

➤ [**Use helpers for objects（オブジェクトでのヘルパーの使用）**]：ヘルパー関数を使って，バリアントのオブジェクトの参照が引数として関数に渡されたときに，その参照を抽出します（Java および VBScript のみ。標準設定では無効）。

手順の詳細については，88 ページ「記録オプションの設定」を参照してください。

相関オプションについて

相関を使って，テストの実行中に動的な値を保持できます。これらのオプションによって，記録時に VuGen によって自動的に行われた相関を拡張設定できます。すべての相関オプションは標準設定では無効になっています。相関オプションは，VB Applications，VBScript，JavaScript などの C 以外の言語にのみ適用されます。

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

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

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

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

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

手順の詳細については，88 ページ「記録オプションの設定」を参照してください。

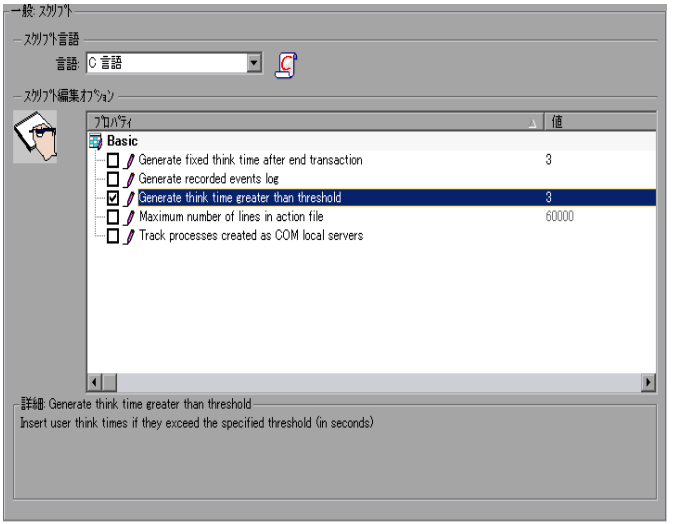
記録オプションの設定

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

スクリプト記録オプションを設定するには，次の手順を実行します。

1 ［記録オプション］ダイアログ・ボックスを開きます。メイン・メニューから［**ツール**］＞［**記録オプション**］を選択するか，［記録開始］ダイアログ・ボックスの［**オプション**］をクリックします。［記録オプション］ダイアログ・ボックスが開きます。

2 ［**一般：スクリプト**］ノードを選択します。



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

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

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

第 6 章

ビジネス・プロセス・レポートの作成

Mercury VuGen では，スクリプトの情報を Microsoft Word 文書にエクスポートすることによって，ビジネス・プロセス・レポートを作成できます。

本章では，次の項目について説明します。

- Word へのスクリプトのエクスポートについて
- レポートの詳細の指定
- レポートの内容の指定

以降の情報は，Web（Click and Script）仮想ユーザ・スクリプトのみを対象とします。

Word へのスクリプトのエクスポートについて

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

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

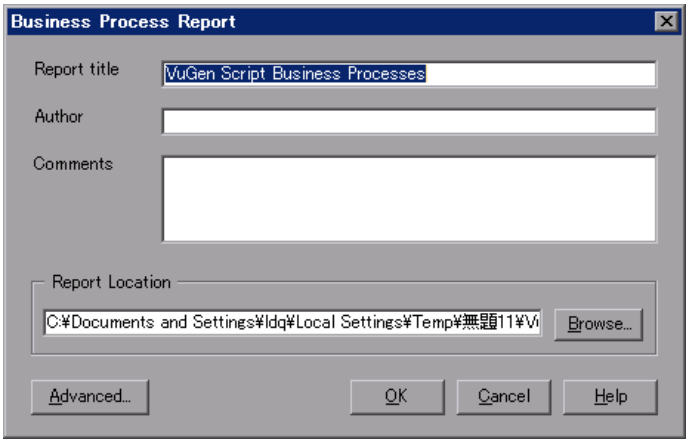
VuGen では，レポートに含める情報のタイプを指定することによって，レポートの内容をカスタマイズできます。

レポートの詳細の指定

レポートの内容を指定する前に，レポートの名前と簡単な説明を指定します。適切なコメントを入力し，スクリプトを格納する場所を指定します。

ビジネス・プロセス・レポートを作成するには，次の手順を実行します。

1 ［**ファイル**］＞［**ビジネス プロセス レポートを作成**］を選択します。［Business Process Report］ダイアログ・ボックスが開きます。



2 ［**Report title**］ボックスにタイトルを指定します。

3 ［**Comment**］ボックスにコメントを入力します。

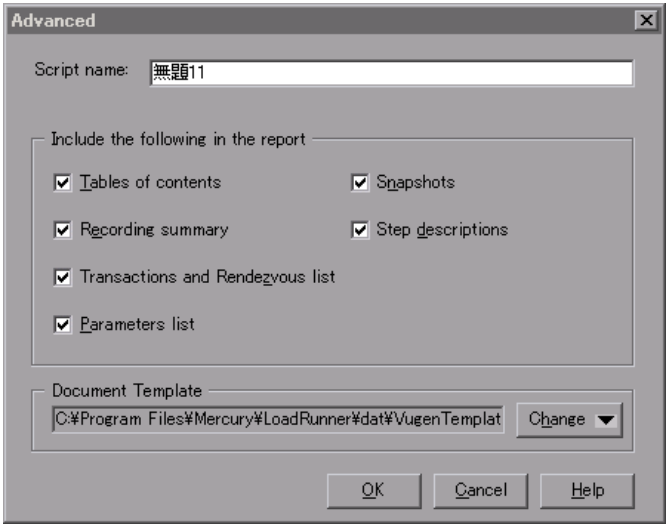
4 標準設定のレポートの場所およびファイル名を受け入れます。または，必要なパスを参照します。標準設定の場所は，スクリプトが保存されている場所です。また，標準設定のレポート名は **VuGen Script Business Process.doc** です。

5 ［**Advanced**］をクリックし，レポートの内容を指定します。内容の選択の詳細については，91 ページ「レポートの内容の指定」を参照してください。

6 レポートを生成するには，［**OK**］をクリックします。

レポートの内容の指定

ビジネス・プロセス・レポートの内容を選択できます。標準設定では，すべてのオプションが有効になっています。



次の内容オプションのうち，1 つ以上を選択できます。

- **Table of Contents**：レポート上の他のすべての内容のページ番号を示す目次。特定のオプションを無効にすると，そのオプションに対応する項目は目次に表示されません。
- **Recording Summary**：［タスク］リストで［記録サマリ］リンクをクリックすると表示される，記録セッションのサマリ。
- **Transactions and Rendezvous list**：スクリプトに定義されているすべてのトランザクションおよびランデブーの包括的なリスト。
- **Parameter list**：スクリプトに対して定義されているすべてのパラメータのリスト。このリストは，［パラメータ リスト］ダイアログ・ボックス（［**仮想ユーザ**］＞［**パラメータ リスト**］）に表示されるパラメータに対応します。
- **Snapshots**：ステップの名前と説明の横に表示される，記録されたステップの実際のスナップショット。
- **Step descriptions**：ツリー・ビューに表示される各ステップの簡単な説明。

➤ **Document Template**：レポートに使用するテンプレートのパスおよびファイル名。標準設定のテンプレートは，LoadRunner の **dat** フォルダにあります。

新しいテンプレートを作成する場合は，新しいテンプレートの基礎として既存のテンプレートを使用することをお勧めします。これにより，必要なブックマークおよびスタイルが，新しいテンプレートでも維持されるようになります。

第 7 章

ポートの割り当て設定

ソケット・レベルでネットワーク・トラフィックを記録するプロトコルを使用する場合，トラフィックをどのように割り当てるかをポートに指定することができます。

本章では，次の項目について説明します。

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

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

ポートの割り当て設定について

ソケット・レベルでネットワーク・トラフィックを記録する仮想ユーザ・スクリプト（HTTP，SMTP，POP3，FTP，IMAP，Oracle NCA，Windows Sockets）を記録する場合，［ポートの割り当て］オプションを設定できます。これらのオプションで，特定のサーバとポートの組み合わせを通じて受け取るトラフィックを，特定の通信プロトコルに割り当てることができます。

割り当ての対象にできる通信プロトコルは，FTP，HTTP，IMAP，NCA，POP3，SMTP および SOCKET です。割り当ては，サーバ名，ポート番号，または「サーバ：ポート」の組み合わせを指定することで作成できます。たとえば，**twilight** というサーバのポート 25 番からのトラフィックをすべて SMTP として扱うよう指定できます。また，**viper** というサーバからのすべてのトラフィックを，ポートに関係なく FTP プロトコルへ割り当てることもできます。さらには，サーバ名に関係なく，ポート 23 のすべてのトラフィックを SMTP に割り当てることも可能です。

マルチ・プロトコル・モードで記録する場合には，少なくとも 1 つのプロトコルがソケット・レベルで記録していると，［**ポートの割り当て**］オプションが利用できます。ただし，シングル・プロトコル・スクリプトで HTTP または WinSock を記録する場合は例外です。この場合は，［**ポートの割り当て**］オプションは利用できません。

ポート割り当ての定義

VuGen はポート割り当ての設定を使用して，特定のサーバとポートの組み合わせを通じて受け取るトラフィックを，特定の通信プロトコルに割り当てます。次の項目についてポート割り当ての設定が可能です。

- ［**キャプチャ レベル**］：キャプチャするデータのレベル。ソケット・レベル・データ，WinINet レベル・データ，またはソケット・レベル・データと WinINet レベル・データの両方。標準設定は，使用可能な最も低いレベルであるソケット・レベル・データです。
- ［**次のネットワーク レベルのサーバ アドレス割り当て**］：プロトコルごとの割り当てを表示するよう指定します。たとえば，FTP の割り当てだけを表示したい場合には［FTP］を選択します。
- ［**新規エントリ**］：［サーバ エントリ］ダイアログ・ボックスが開き，新しい割り当てを入力できます。詳細については，96 ページ「新規サーバ・エントリの追加」を参照してください。

- [**エントリの編集**]：[サーバ エントリ] ダイアログ・ボックスが開き，選択したエントリを編集できます。
- [**エントリの削除**]：選択したエントリを削除します。
- [**オプション**]：[ポートの割り当ての詳細設定] ダイアログ・ボックスが開き，通信プロトコルと SSL レベルの自動検出を有効にできます。詳細については，99 ページ「高度なポート割り当てオプションの設定」を参照してください。

設定したポートとサーバの名前が全部ではない場合，VuGen は次の優先順位に従ってデータをサービスに割り当てます。

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

優先順位の高い割り当てがある場合に，それよりも優先順位の低い割り当てを指定しても，優先順位の低い割り当ては無効です。たとえば，**twilight** というサーバのポート番号 25 番のトラフィックを SMTP として扱うよう指定した後で，すべてのサーバのポート 25 番を HTTP として扱うよう指定しても，データは SMTP として扱われます。

さらに，次のガイドラインが適用されます。

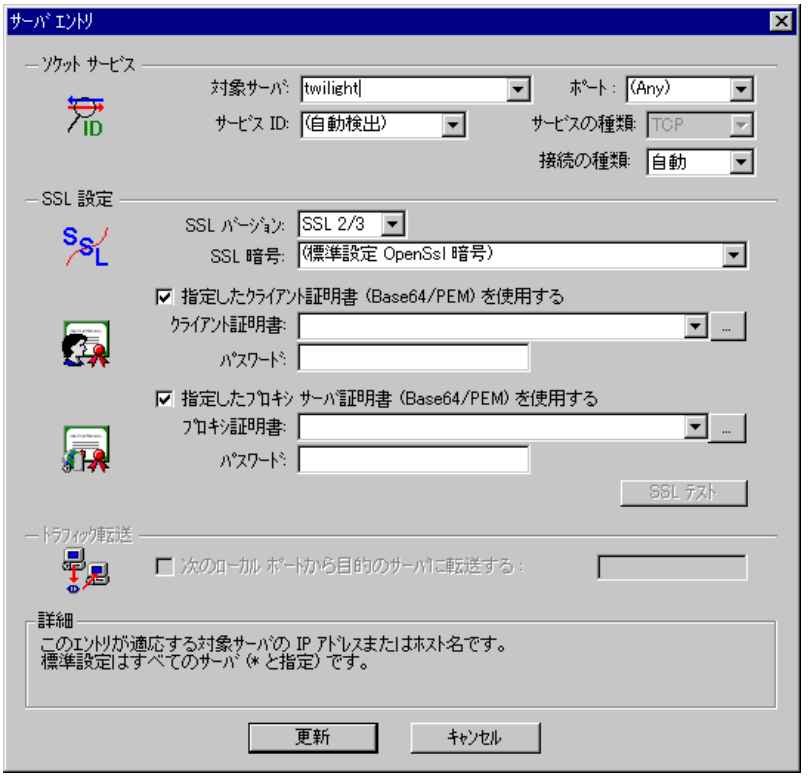
- **ポート 0**：ポート番号 0 は任意のポートを表します。
- **強制割り当て**：ポート番号，サーバ名，または「サーバ：ポート」の組み合わせの割り当てを指定した場合，VuGen では，ネットワーク・トラフィックがそのサービスを使用するよう強制されます。たとえば，「＜任意のサーバ＞」のポート 80 番が FTP を使用するよう指定した場合，VuGen では，実際の通信が HTTP であったとしても，FTP プロトコルを使用してその通信が記録されます。この例では，仮想ユーザ・スクリプトは空となる可能性があります。

ポート割り当てを定義すると，その内容は [ポートの割り当て] の一覧に表示されます。各項目のチェック・ボックスをクリアすることで，一時的に割り当てを無効にできます。割り当てを無効にすると，VuGen は無効にした「サーバ：ポート」の組み合わせに割り当てられた，すべてのトラフィックを無視します。データが無関係な場合やプロトコルがサポートされていない場合は，ポートの割り当てを無効にしてください。

手順の詳細については，100 ページ「ポートの割り当て記録オプションの設定」を参照してください。

新規サーバ・エントリの追加

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



ソケット サービス

- ［**対象サーバ**］：エントリ項目に登録する対象サーバの IP アドレスまたはホスト名。標準設定は「任意のサーバ」です。
- ［**ポート**］：エントリ項目に登録する対象サーバのポート番号。ポート番号「0」は「すべてのポート」を意味します。

- ［**サービス ID**］：接続のタイプを識別するためにレコーダが使用するプロトコルまたはサービス名（HTTP，FTP など）。新しい名前を指定することもできます。指定できる名前の長さは最長 8 文字です。
- ［**サービスの種類**］：サービスのタイプ。現在は TCP/IP に設定されています。
- ［**記録の種類**］：記録のタイプ（直接か，それともプロキシ・サーバを経由するか）。
- ［**接続の種類**］：接続のセキュリティ・レベル。［非認証］，［SSL］，［自動］があります。［自動］を選択すると，レコーダによって SSL シグネチャについて最初の 4 バイトが検査されます。SSL 署名を検出すると，SSL が使用されていると推定されます。

注：SSL 接続は，次のメール・サービス・プロトコルと e ビジネス・プロトコルには適用されません：FTP，LDAP，SMTP，POP3，IMAP，DNS，MAPI。

SSL 設定

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

- ［**SSL バージョン**］：クライアント・アプリケーションおよびサーバとの通信に使用する SSL のバージョン。標準設定では，SSL 2/3 が指定されています。ただし，サービスによっては SSL 3.0 または SSL 2.0 のみが必要になることがあります。新しいワイヤレス・アプリケーションでは新しいセキュリティ・アルゴリズム，TLS 1.0 を必要とします。
- ［**SSL 暗号**］：リモートのセキュア・サーバに接続するのに使用する SSL 暗号を指定します。
- ［**指定したクライアント証明書（Base64/PEM）を使用する**］：リモート・サーバに接続する際に使用する標準のクライアント側の証明書を指定します。**txt**，**crt**，**pem** 形式のいずれかの証明書を指定するか一覧から探し，パスワードを入力します。

➤ ［**指定したプロキシ サーバ証明書（Base64/PEM）を使用する**］：サーバの証明書を要求するクライアント・アプリケーションに示す標準の証明書を指定します。**txt**，**crt**，**pem** 形式のいずれかの証明書を指定するか一覧から探し，パスワードを入力します。［**SSL テスト**］をクリックすると，サーバに対する認証情報をチェックできます。

トラフィックの転送

➤ ［**次のローカル ポートから目的のサーバに転送する**］：特定のポートからのすべてのトラフィックを別のサーバに転送できます。このオプションは，特別な UNIX マシンの場合など，クライアント上で VuGen を正常に実行できない場合や，VuGen を介してアプリケーション・サーバを起動できない場合に特に役立ちます。クライアント・マシンに問題がある場合は，そのマシンからのトラフィックを捕獲して，サーバに渡すよう VuGen の設定を行います。こうすることで，VuGen はデータを処理し，アクションに対応するコードを生成することが可能となります。

たとえば，**host1** という UNIX のクライアントが，サーバ **server1** と，ポート番号 8080 経由で通信しており，［ポートの割り当て］には，**server1**，ポート 8080 のエントリが設定されていたとします。［サーバ エントリ］ダイアログ・ボックスの［**トラフィック転送**］セクションで，［**次のローカル ポートから目的のサーバに転送する**］チェック・ボックスを選択して，トラフィックの転送を有効にします。トラフィックの転送に使用するポート（この例では 8080）を指定します。

次にクライアントを **server1** ではなく，VuGen を実行している **host1** に接続します。VuGen はクライアント・マシンからの通信を受信し，その通信をローカルのポート 8080 を経由してサーバに転送します。トラフィックは VuGen を経由するため，トラフィックの分析と適切なコードの生成が可能となります。

手順の詳細については，100 ページ「ポートの割り当て記録オプションの設定」を参照してください。

高度なポート割り当てオプションの設定

VuGen の高度なポート割り当てオプションでは，**自動検出**オプションを設定できます。VuGen の自動検出では，サーバに送られるデータが解析されます。さらに，シグネチャ・データやデータのパターンが調べられ，プロトコルが特定されます。シグネチャを検出するため，最初の受信バッファまで，すべての送信バッファが蓄積されます。受信バッファが返されるまでに送信されたすべての送信バッファは単一のデータ**遷移**とみなされます。標準設定では，割り当ては定義されず，VuGen は自動検出を行います。一部のプロトコル（HTTP など）は，単一の遷移のなかでタイプが判別されます。他のネットワーク・プロトコルでは，タイプを判別されるまでにいくつかの遷移が必要です。このため，VuGen ではサーバとポートの組み合わせごとに一時バッファが作成されます。VuGen によって最初の遷移バッファが読み取られてもプロトコル・タイプが判別できない場合は，データが一時バッファに格納されます。そして，プロトコルを特定できるシグネチャが検出されるまで，着信バッファの読み取りが継続されます。

標準では，VuGen がプロトコルのシグネチャの検出に使用できるトランザクションは 4 つ，バッファは最大 2048 バイトまでです。VuGen が最大トランザクションに達するか，バッファ・サイズの上限に達してもプロトコルを特定できなかった場合，データは WinSock プロトコルに割り当てられます。VuGen に（マルチ・プロトコルを選択している場合）WinSock プロトコルを記録するように設定していない場合には，VuGen によってデータが破棄されます。

プロトコルのタイプを検出するために VuGen が読み取るバッファの最大サイズを変更することができます。また一時バッファのサイズを指定することも可能です。最初の送信バッファに格納されたデータの合計が，一時バッファのサイズより大きくなった場合，VuGen ではプロトコル・タイプの自動検出が行えなくなります。この場合には一時バッファのサイズを増やす必要があります。

➤ ［**SSL 自動検出を有効にする**］: SSL 通信を自動的に検出します。検出したい SSL のバージョンと標準の SSL 暗号の形式を指定します。これはポートの割り当てが，［**接続の種類**］ボックスで［**自動**］に指定されているか，まったく指定のない場合にだけ適用されることに注意してください。サーバ，ポート，もしくは「サーバ：ポート」の組み合わせが，［非認証］，［SSL］のいずれかに指定されている場合には，自動 SSL 検出は適用されません。

- ［**SOCKET ベース コミュニケーションの自動検出を有効にする**］：SSL 通信を自動的に検出します。必要な場合，VuGen によるプロトコルの検出が成功するまで 1 つずつ［移行しきい値の送受信］の最大数を増やします。また，VuGen によるプロトコルの検出が成功するまで，一度に 1024 バイト（1KB）ずつ［バッファサイズしきい値の送受信］の最大数を増やすこともできます。これらを行うと VuGen によるシグネチャのためにより多くのデータを調査することができるようになります。
- ［**ログ レベル**］：自動ソケット検出のログ・レベル（なし，標準（標準設定），デバッグ，高度なデバッグ）を設定します。

  上記のネットワーク・レベル・プロトコルを使用する場合は，VuGen がプロトコル・タイプを判別するよう，自動検出オプションをオンにした設定を推奨します。ほとんどの場合，VuGen のレコーダでは，これらのプロトコルのシグネチャが認識できます。そして，プロトコルの仕様に従ってプロトコルが自動的に処理されます。ただし，VuGen でプロトコルが認識されないこともあります。たとえば，次のような場合です。

  - 既存のプロトコルに類似するプロトコル・シグネチャがあり，誤った処理が行われた場合。
  - プロトコルに一意のシグネチャが無い場合。
  - プロトコルが SSL による暗号を使用しているため，WinSock レベルで認識されない場合。

  上記のどの場合も，プロトコルをホストするサーバとポートを一意に識別する情報を提供することができます。

  手順の詳細については，100 ページ「ポートの割り当て記録オプションの設定」を参照してください。

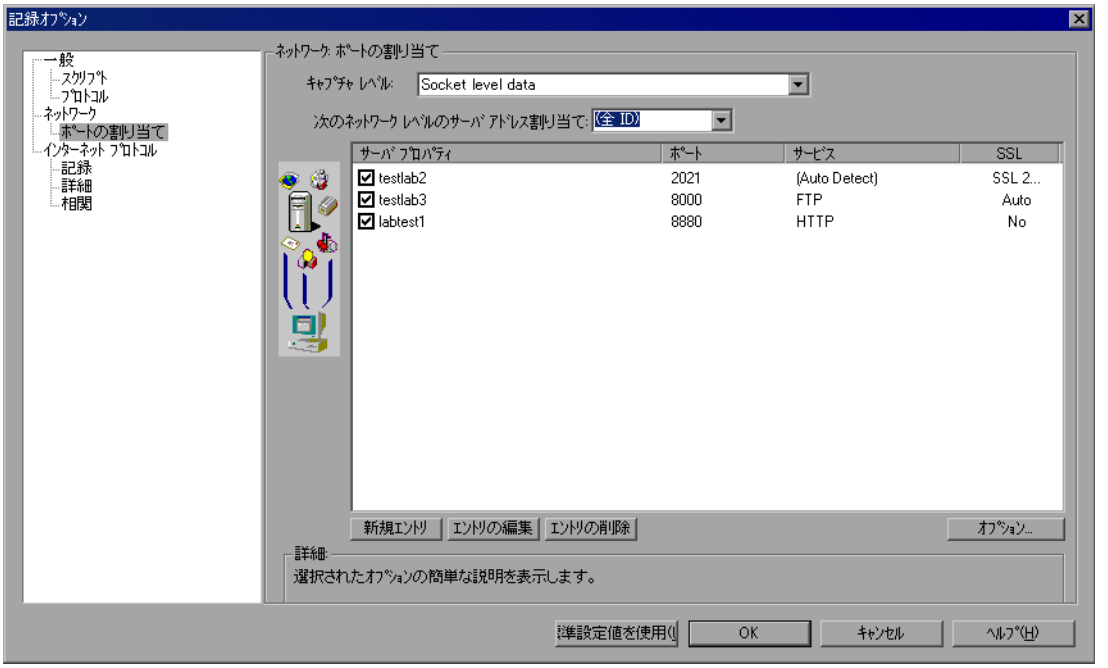
ポートの割り当て記録オプションの設定

［記録オプション］ダイアログ・ボックスは，次のいくつかの方法で開くことができます。

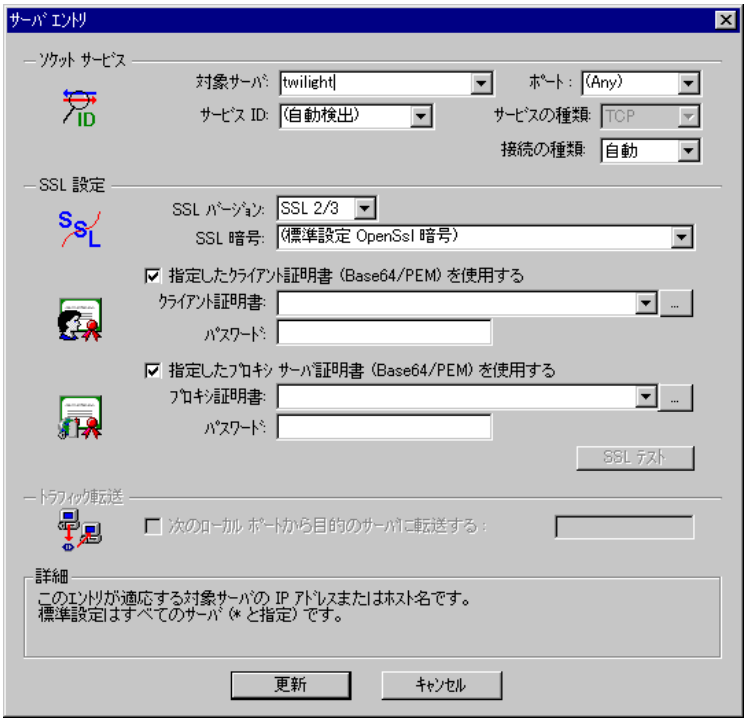
- ツールバー・ボタン： 
- キーボードのショートカット：Ctrl キーを押しながら F7 キーを押します。
- ［ツール］メニュー：［**ツール**］＞［**記録オプション**］を選択します。

ポートの割り当て記録オプションを設定するには，次の手順を実行します。

1 ［記録オプション］を開き，［**ネットワーク：ポートの割り当て**］ノードを選択します。



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



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

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

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

［**SSL テスト**］をクリックすると，サーバに対する認証情報をチェックできます。

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

6 ［**更新**］をクリックして，ダイアログ・ボックスを閉じます。

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

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

通信の種類を自動的に検出するには，［**SOCKET ベース コミュニケーションの自動検出を有効にする**］チェック・ボックスをチェックします。必要に応じて，移行の最大数を増やします。

［**ログ レベル**］（なし，標準，デバッグ，高度なデバッグ）を選択します。

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

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

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

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

11 ［**OK**］をクリックします。

第 8 章

仮想ユーザ・スクリプトの拡張

記録中または記録後に，一般仮想ユーザ関数，プロトコル固有の仮想ユーザ関数，および標準 ANSI C 関数を追加して，仮想ユーザ・スクリプトを拡張できます。

本章では，次の項目について説明します。

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

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

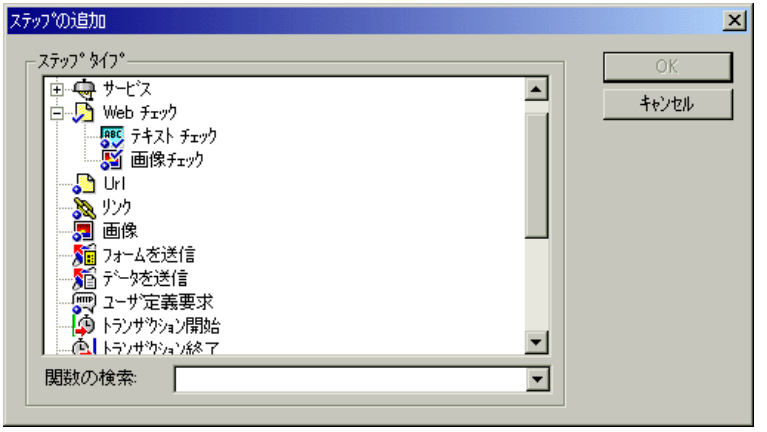
仮想ユーザ・スクリプトの拡張について

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

スクリプトに新規ステップを追加するには，次の手順を実行します。

1 希望の場所にカーソルを置きます。

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



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

［ステップの追加］ダイアログ・ボックスでは，次の種類の関数が使用できます。

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

一般仮想ユーザ関数

一般仮想ユーザ関数は，仮想ユーザ・スクリプトの機能を大幅に強化します。たとえば，一般仮想ユーザ関数を使って，サーバ・パフォーマンスの測定，サーバ負荷の制御，デバッグ・コードの追加，テストの仮想ユーザについてのランタイム情報の取得などができます。

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

本章では，最もよく使う一般仮想ユーザ関数だけを対象に使い方を説明します。仮想ユーザ関数の詳細については，「**オンライン関数リファレンス**」（［**ヘルプ**］＞［**関数リファレンス**］）を参照してください。

プロトコル固有の仮想ユーザ関数

仮想ユーザ・スクリプトを拡張する関数ライブラリがいくつかあります。各ライブラリは，仮想ユーザのタイプに固有のものです。たとえば，Windows Sockets 仮想ユーザ・スクリプトでは **LRS** 関数を使用し，Tuxedo 仮想ユーザ・スクリプトでは **LRT** 関数を使用します。プロトコル固有の仮想ユーザ関数の詳細については，「**オンライン関数リファレンス**」（［**ヘルプ**］＞［**関数リファレンス**］）を参照してください。

標準 ANSI C 関数

標準 ANSI C 関数を追加して仮想ユーザ・スクリプトを拡張できます。ANSI C 関数を使って，仮想ユーザ・スクリプトにコメント文，フロー制御ステートメント，条件文などを追加できます。標準 ANSI C 関数は任意のタイプの仮想ユーザ・スクリプトに追加できます。詳細については，597 ページ「C 言語の関数の使用についてのガイドライン」を参照してください。

仮想ユーザ・スクリプトへのトランザクションの挿入

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

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

LoadRunner および Tuning Module では，コントローラまたはコンソールによって，各トランザクションの実行にかかる時間が測定されます。テスト実行の後，アナリシスのグラフとレポートを使用して，トランザクションごとのサーバ・パフォーマンスを分析します。

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

ここでは，記録中にトランザクションを作成する方法について説明します。

トランザクションの開始を示す方法

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

トランザクションの開始を示すには，次の手順を実行します。

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

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

［**OK**］をクリックしてトランザクション名を確定します。VuGen によって **lr_start_transaction** ステートメントが仮想ユーザ・スクリプトに挿入されます。たとえば，次の関数は **trans1** トランザクションの開始を示します。

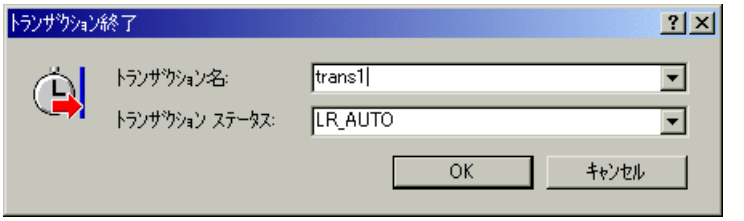
```
lr_start_transaction("trans1");
```

トランザクションの終了を示す方法

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

トランザクションの終了を示すには，次の手順を実行します。

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



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

［**OK**］をクリックしてトランザクション名を確定します。VuGen によって **lr_end_transaction** ステートメントが仮想ユーザ・スクリプトに挿入されます。たとえば，次の関数は **trans1** トランザクションの終了を示します。

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

注：トランザクションにトランザクションが含まれる**入れ子**のトランザクションを作成できます。トランザクションを入れ子にする場合は，含まれる側のトランザクションを含む側のトランザクションをよりも前に閉じなければなりません。そうしないとトランザクションが正しく解析されません。

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

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

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

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

ランデブー・ポイントを挿入するには，次の手順を実行します。

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

2 ランデブー・ポイントの名前を［**名前**］ボックスに入力します。

［**OK**］をクリックします。VuGen によって **lr_rendezvous** ステートメントが仮想ユーザ・スクリプトに挿入されます。たとえば，次の関数は **rendezvous1** という名前のランデブー・ポイントを定義します。

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

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

仮想ユーザ・スクリプトへのコメントの挿入

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

コメントを挿入するには，次の手順を実行します。

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

2 テキスト・ボックスにコメントを入力します。

3 ［**OK**］をクリックするとダイアログ・ボックスが閉じ，コメントが挿入されます。テキストはスクリプトの現在位置に，コメント記号で囲まれた状態で挿入されます。次のスクリプトの抜粋は，仮想ユーザ・スクリプトに挿入されたコメントを示します。

```
/*
* これは最初のクエリです
*/
```

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

仮想ユーザ情報の取得

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	コマンド・ライン・パラメータ文字列を返します。
lr_get_host_name	仮想ユーザ・スクリプトを実行しているマシンの名前を返します。
lr_get_master_host_name	コントローラまたは Tuning Console を実行しているマシンの名前を返します。Mercury Business Availability Center は適用対象外です。
lr_whoami	スクリプトを実行している仮想ユーザの名前を返します。Mercury Business Availability Center は適用対象外です。

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

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

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

出力へのメッセージの送信

仮想ユーザ・スクリプトの中でメッセージ・タイプの関数を使用すると，カスタマイズしたエラー・メッセージや通知メッセージを出力やログ・ファイルに送信できます。たとえば，クライアント・アプリケーションの現在のステータスを示すメッセージを挿入することができます。LoadRunner コントローラまたは Tuning Console は，これらのメッセージを［出力］ウィンドウに表示します。また，これらのメッセージをファイルに保存することもできます。

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

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

次のメッセージ関数を仮想ユーザ・スクリプトの中で使用できます。

lr_debug_message	デバッグ・メッセージを［出力］ウィンドウまたは Business Process Monitor ログ・ファイルに送ります。
lr_error_message	エラー・メッセージを［出力］ウィンドウまたは Business Process Monitor ログ・ファイルに送ります。
lr_get_debug_message	現在のメッセージ・クラスを取得します。
lr_log_message	出力メッセージを仮想ユーザ・スクリプトのディレクトリにあるログ・ファイル **output.txt** に直接送信します。この関数は，出力メッセージによって TCP/IP のトラフィックが妨げられるのを防ぐので便利です。
lr_output_message	メッセージを［出力］ウィンドウまたは Business Process Monitor ログ・ファイルに送ります。
lr_set_debug_message	出力メッセージのメッセージ・クラスを設定します。
lr_vuser_status_message	メッセージをコントローラまたは Tuning Console の仮想ユーザ・ステータス領域に送信します。Mercury Business Availability Center は適用対象外です。
lr_message	メッセージを，仮想ユーザ・ログと，［出力］ウィンドウまたは Business Process Monitor ログ・ファイルに送ります。

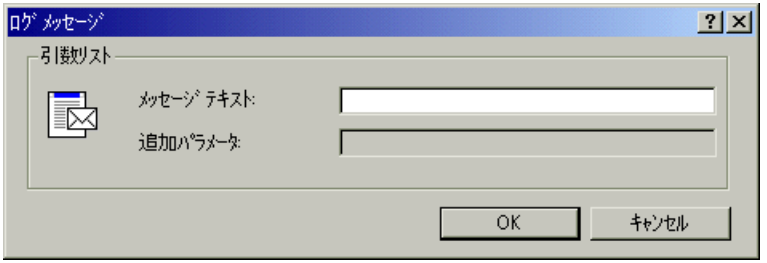
注：lr_message 関数，**lr_output_message** 関数，**lr_log_message** 関数の振る舞いは，実行環境の設定の［ログ］の設定のスクリプトのデバッグ・レベルの影響を受けません。これらの関数は常にメッセージを送信します。

ログ・メッセージ

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

lr_log_message 関数を挿入するには，次の手順を実行します。

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



2 ［**メッセージ テキスト**］ボックスにメッセージを入力します。

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

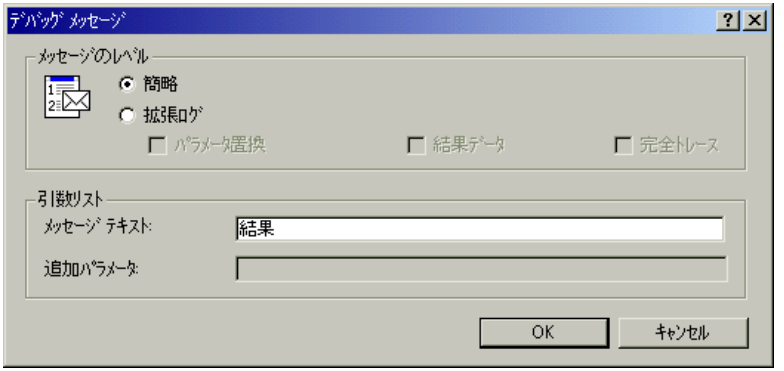
デバッグ・メッセージ

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

デバッグ関数を挿入するには，次の手順を実行します。

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

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



3 メッセージのレベルとして，［**簡略**］または［**拡張ログ**］を選択します。［拡張ログ］を選択した場合には，ログに記録する情報の種類を，［**パラメータ置換**］，［**結果データ**］，［**完全トレース**］の中から指定します。

4 ［**メッセージ テキスト**］ボックスにメッセージを入力します。

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

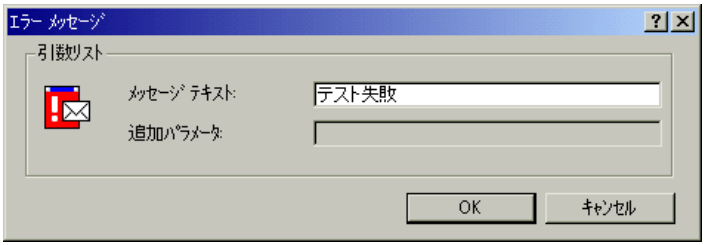
エラーおよび出力メッセージ

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

エラーまたは出力メッセージ関数を挿入するには，次の手順を実行します。

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

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



3 ［**メッセージ テキスト**］ボックスにメッセージを入力します。

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

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

実行中の仮想ユーザ・スクリプトのエラー処理

スクリプト実行時の仮想ユーザによるエラー処理の方法を指定することができます。標準設定では，仮想ユーザによってエラーが検出されると，スクリプトの実行が停止されます。次のいずれかの方法を使って，エラーが発生したときに次の反復を継続するように仮想ユーザを指定できます。

- 実行環境の設定を使用する：［**エラーでも処理を継続する**］実行環境の設定を指定できます。実行環境の設定で［**エラーでも処理を継続する**］を設定すると，仮想ユーザ・スクリプト全体に適用されます。スクリプトの一部分で，実行環境の設定の［**エラーでも処理を継続する**］をオーバーライドするには，**lr_continue_on_error** 関数を使用します。詳細については，211 ページ「エラー処理」を参照してください。

➤ **lr_continue_on_error** 関数を使用する：**lr_continue_on_error** 関数を使用して，仮想ユーザ・スクリプトの特定のセグメントでのエラー処理を制御できます。対象セグメントを指定するには，セグメントを lr_continue_on_error(1); ステートメントと lr_continue_on_error(0); ステートメントで囲みます。新しいエラー設定は，囲まれた部分の仮想ユーザ・スクリプト・セグメントに適用されます。詳細については，以降の説明を参照してください。

たとえば，実行環境の設定で［エラーでも処理を継続する］を有効にしていて，仮想ユーザが次に示すスクリプトのセグメントの再生中にエラーに遭遇した場合には，仮想ユーザはスクリプトの実行を継続します。

```
        web_link("EBOOKS",
            "Text=EBOOKS",
            "Snapshot=t2.inf",
            LAST);

        web_link("Find Rocket eBooks",
            "Text=Find Rocket eBooks",
            "Snapshot=t3.inf",
            LAST);
```

特定のセグメントを対象に，仮想ユーザにエラーでもスクリプトの実行を継続させるには，適切な **lr_continue_on_error** ステートメントで対象セグメントを囲みます。

```
    lr_continue_on_error(1);
        web_link("EBOOKS",
            "Text=EBOOKS",
            "Snapshot=t2.inf",
            LAST);

        web_link("Find Rocket eBooks",
            "Text=Find Rocket eBooks",
            "Snapshot=t3.inf",
            LAST);
    lr_continue_on_error(0);
```

仮想ユーザ・スクリプトの同期化

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

次の同期化関数を使用できます。

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

RTE 仮想ユーザ・スクリプトにおける同期化関数の使用法の詳細については，第 71 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。

ユーザの思考遅延時間のエミュレート

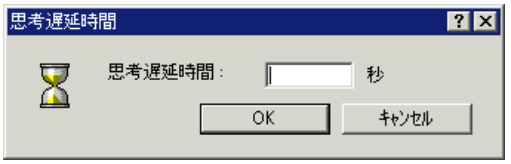
連続する操作の合間のユーザの待ち時間を「**思考遅延時間**」といいます。仮想ユーザでは，**lr_think_time** 関数を使ってユーザの思考遅延時間をエミュレートできます。仮想ユーザ・スクリプトを記録すると，VuGen によって実際の思考遅延時間が記録され，適切な **lr_think_time** ステートメントが仮想ユーザ・スクリプトに挿入されます。記録された **lr_think_time** ステートメントは後で編集できます。また，手作業でも **lr_think_time** ステートメントを仮想ユーザ・スクリプトに追加できます。

思考遅延時間ステートメントを手作業で追加するには，次の手順を実行します。

1 希望の場所にカーソルを置きます。

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

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



4 思考遅延時間を秒単位で指定し，［**OK**］をクリックします。

注：Java 仮想ユーザ・スクリプトを記録する場合，**lr_think_time** ステートメントは仮想ユーザ・スクリプトに挿入されません。

思考遅延時間の設定を使って，仮想ユーザ・スクリプトを実行するときの **lr_think_time** ステートメントの処理方法を設定できます。思考遅延時間の設定にアクセスするには，VuGen のメイン・メニューから［**仮想ユーザ**］＞［**実行環境の設定**］を選択して［**思考遅延時間**］ノードをクリックします。詳細については，「**オンライン関数リファレンス**」（［**ヘルプ**］＞［**関数リファレンス**］）を参照してください。

コマンド・ライン引数の取り扱い

スクリプトを実行するときにコマンド・ライン引数を指定することで，実行時に仮想ユーザ・スクリプトに値を渡すことができます。Tuning Console を使用している場合は，［実行環境設定］ダイアログ・ボックスの中でコマンド・ライン引数を指定できます。詳細については，209 ページ「実行環境の追加属性の設定」を参照してください。

コマンド・ライン引数を読み取って，値を仮想ユーザ・スクリプトに渡すことができる関数として，次の 3 つがあります。

lr_get_attrib_double	double float 型の引数を取得します。
lr_get_attrib_long	long int 型の引数を取得します。
lr_get_attrib_string	文字列を取得します。

コマンド・ラインの形式は次の 2 つのどちらかを使用します。スクリプト名の後ろに，引数とその値を 2 つ 1 組で指定します。

```
スクリプト名　- 引数 引数値　- 引数 引数値
```

```
スクリプト名　/ 引数 引数値　/ 引数 引数値
```

たとえば，次の例は，pc4 というロード・ジェネレータで **script1** を 5 回繰り返すためのコマンド・ライン文字列を示します。

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

コマンド・ライン解析関数またはコマンド・ラインでの引数の使い方の詳細については，「**オンライン関数リファレンス**」（［**ヘルプ**］＞［**関数リファレンス**］）を参照してください。

テキストの暗号化

スクリプト内のテキストを暗号化して，パスワードなどの機密性の高いテキスト文字列を保護できます。暗号化は，ユーザ・インタフェースから自動的に行うことができます。また，手作業でも実行できます。暗号化した文字列は，スクリプト中に暗号化された文字列として表れます。VuGen は 32 ビットの暗号化をサポートしています。

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

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

文字列は，いつでも復号して元の値に戻せます。

文字列を暗号化するには，次の手順を実行します。

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

2 暗号化するテキストを選択します。

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

暗号化された文字列を元に戻すには，次の手順を実行します。

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

2 元に戻す文字列を選択します。

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

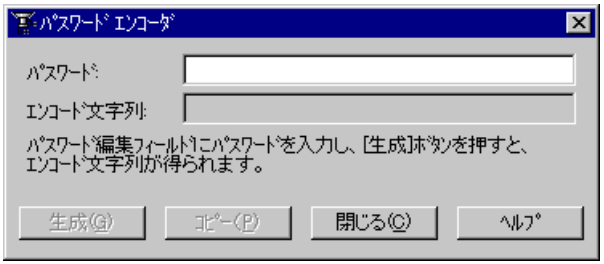
lr_decrypt 関数の詳細については，「**オンライン関数リファレンス**」（［**ヘルプ**］＞［**関数リファレンス**］）を参照してください。

手動でのパスワードのエンコーディング

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

パスワードを暗号化するには，次の手順を実行します。

1 Windows メニューの［**スタート**］＞［**プログラム**］＞［**Mercury LoadRunner**］＞［**Tools**］＞［**Password Encorder**］を選択します。［パスワード エンコーダ］ダイアログ・ボックスが開きます。



2 ［**パスワード**］ボックスにパスワードを入力します。

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

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

5 暗号化したいパスワードごとに，この手順を繰り返します。

6 ［**閉じる**］をクリックして，［パスワード エンコーダ］を閉じます。

スクリプト・フォルダへのファイルの追加

ファイルをスクリプト・ディレクトリに追加し，スクリプトの実行時に使用できるようにすることが可能です。

ファイルを追加するには，次の手順を実行します。

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

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

第 9 章

VuGen パラメータを使った作業

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

本章では，次の項目について説明します。

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

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

VuGen パラメータの定義について

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

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

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

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

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

こうしておけば，仮想ユーザを実行したときに，指定したデータ・ソースにある値でパラメータが置き換えられます。データ・ソースとして，ファイルまたは内部で生成された変数を使用できます。データ・ソースの詳細については，130 ページ「パラメータ・タイプについて」を参照してください。

仮想ユーザ・スクリプトのパラメータ化には次の 2 つの利点があります。

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

パラメータ化には，次の 2 つの作業が伴います。

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

パラメータに関する制限

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

lrd_stmt 関数を例にとってみます。この関数の構文は次のとおりです。

```
lrd_stmt (LRD_CURSOR FAR *mptCursor, char FAR *mpcText, long
mliTextLen, LRDOS_INT4 mjOpt1, LRDOS_INT4 mjOpt2, int
miDBErrorSeverity);
```

「**オンライン関数リファレンス**」によれば，パラメータ化できるのは **mpcText** 引数だけです。

記録された **lrd_stmt** 関数が次のようになっていたとします。

```
lrd_stmt(Csr4, "select name from sysobjects where name =¥"Kim¥" ", -1,
148, -99999, 0);
```

これは次のようにパラメータ化できます。

```
lrd_stmt(Csr4, "select name from sysobjects where name =¥"<name>¥" ",
-1, 148, -99999, 0);
```

注：lr_eval_string 関数を使えば，標準ではパラメータ化できない関数の引数でも「パラメータ化」することができます。さらに，**lr_eval_string** 関数を使えば，仮想ユーザ・スクリプトの任意の文字列も「パラメータ化」できます。

VB，COM，および Microsoft .NET プロトコルには，パラメータを定義するのに **lr.eval string** 関数を使用する必要があります。

例：lr.eval_string("**{Custom_param}**")

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

パラメータの作成

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

手順 1：パラメータ化する引数を選択します。

スクリプト・ビューで作業している場合：

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

注：

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

ツリー・ビューで作業している場合：

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

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

［パラメータの選択または作成］ダイアログ・ボックスが開きます。

手順 2：パラメータ名を指定します。

［**パラメータ名**］ボックスにパラメータの名前を入力します。このパラメータ名はスクリプト内で元の引数の代わりに表示されます。

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

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

注：unique というパラメータ名は使用しないでください。VuGen によってすでに使用されています。

手順 3：パラメータ・タイプを選択します。

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

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

パラメータの別のごく一般的な使用法として，ユーザが定義した値が格納されているデータ・テーブルや外部ファイルから値を取得するように，仮想ユーザを設定する方法があります。このようなパラメータを，ファイル・タイプ・パラメータ，およびテーブル・タイプ・パラメータと呼びます。

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

各種のパラメータ・タイプの詳細については，130 ページ「パラメータ・タイプについて」を参照してください。

手順 4：パラメータ・タイプのプロパティを定義します。

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

2 ［**テーブルの作成**］をクリックします。メッセージ・ボックスが表示されます。［**OK**］をクリックします。

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

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

値をさらにテーブルに追加するには，この手順を繰り返します。

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

詳細については，133 ページ「パラメータのプロパティの定義」を参照してください。

手順 5：そして引数をパラメータで置き換えます。

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

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

注：標準のパラメータの括弧は，プロトコルの種類に応じて中括弧か大括弧のどちらかです。パラメータの括弧を変更したければ，［一般オプション］ダイアログ・ボックス（［**ツール**］＞［**一般オプション**］を選択）の［パラメータ化］タブから変更できます。詳細については，140 ページ「パラメータ化オプションの設定」を参照してください。

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

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

パラメータ・タイプについて

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

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

ファイルまたはテーブル・パラメータ・タイプ

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

データ・ファイル

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

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

次の例では，データ・ファイルに ID 番号と名前を格納しています。

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

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

データ・テーブル

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

データ・ファイルまたはデータ・テーブルのパラメータのプロパティについては，第 10 章「ファイル，テーブル，および XML パラメータ・タイプ」を参照してください。

XML パラメータ・タイプ

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

内部データ・パラメータ・タイプ

内部データは仮想ユーザの実行中に自動的に生成されるデータです。Date/Time（日時），Group Name（グループ名），Iteration Number（反復回数），Load Generator Name（ロード・ジェネレータ名），Random Number（乱数），Unique Number（一意の数），Vuser ID（仮想ユーザ）などがあります。

内部データのパラメータのプロパティの定義方法については，168 ページ「内部データ・パラメータ・タイプのプロパティの設定」を参照してください。

ユーザ定義関数のパラメータ

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

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

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

この関数に送られる引数は両方とも NULL です。

ライブラリを作成するときには，標準のダイナミック・ライブラリ・パスを使うことをお勧めします。そうすれば，ライブラリの完全パス名を入力する必要がなく，ライブラリ名を入力するだけで済みます。Mercury 仮想ユーザ・ジェネレータの bin ディレクトリは標準のダイナミック・ライブラリ・パスに含まれています。このディレクトリにライブラリを追加できます。

ユーザ定義関数の例を以下に示します。

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

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

ユーザ定義関数のプロパティの定義方法については，179 ページ「ユーザ定義関数のプロパティの設定」を参照してください。

パラメータのプロパティの定義

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

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

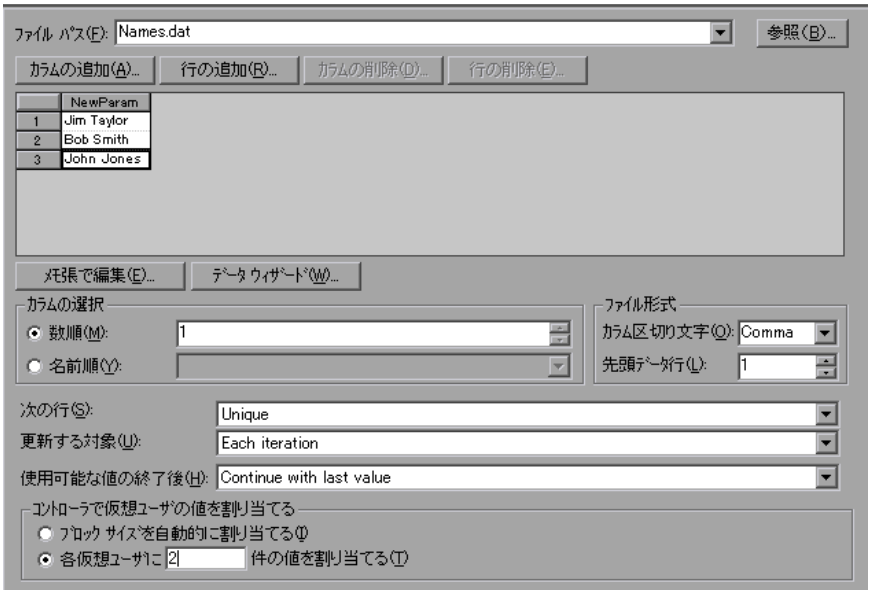
1 **［パラメータのプロパティ］ダイアログ・ボックスを開きます。**

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

- 新しいパラメータを作成するときに（126 ページ「パラメータの作成」を参照），［パラメータの選択または作成］ダイアログ・ボックスの［**プロパティ**］をクリックして，［パラメータのプロパティ］ダイアログ・ボックスを開きます。
- スクリプト・ビュー内でパラメータを選択し，右クリック・メニューから［**パラメータのプロパティ**］を選択します。
- ツリー・ビューで，プロパティを定義する対象となるパラメータを含んでいるステップを右クリックし，［**プロパティ**］を選択します。選択したステップに対応する［ステップのプロパティ］ダイアログ・ボックスが開きます。

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

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



2 **パラメータのプロパティを定義します。**

- ファイルおよびテーブル・タイプ・パラメータのプロパティを定義する方法については，第 10 章「ファイル，テーブル，および XML パラメータ・タイプ」を参照してください。
- 内部データ・パラメータ・タイプのプロパティを定義する方法については，168 ページ「内部データ・パラメータ・タイプのプロパティの設定」を参照してください。
- ユーザ定義関数のプロパティを定義する方法については，132 ページ「ユーザ定義関数のパラメータ」を参照してください。

3 **［パラメータのプロパティ］ダイアログ・ボックスを閉じます。**

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

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

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

詳細については，138 ページ「パラメータ・リストの使用」を参照してください。

既存のパラメータの使用

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

定義済みパラメータを使用した文字列の置換

定義済みパラメータを引数に対して割り当てることができます。

定義済みパラメータで文字列を置換するには，次の手順を実行します。

1 スクリプト・ビューに入ります。

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

3 次のいずれかの方法でパラメータを選択します。

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

［**パラメータ リスト**］を使用すると，以前に定義したパラメータで引数を置き換えられるのと同時に，そのプロパティの表示や変更もできるので便利です。パラメータ・リストの使い方の詳細については，138 ページ「パラメータ・リストの使用」を参照してください。

複数の出現の置き換え

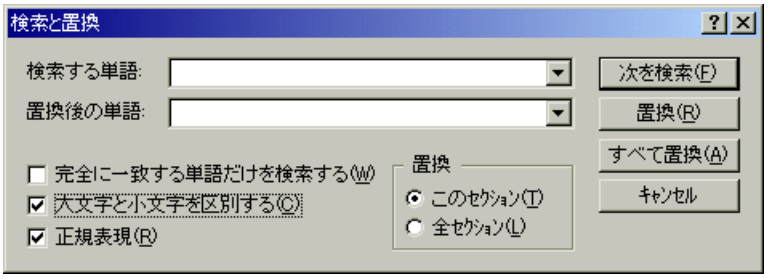
パラメータを作成すると，システムによって引数の元の値が記憶されます。**検索と置換**機能を使用すると，同じ引数が複数出現する場合に，そのうちの選択した引数，またはそれらすべての引数を，同じパラメータや既存の別のパラメータに置換できます。

複数出現する引数を特定のパラメータで置換するには，次の手順を実行します。

1 パラメータを右クリックし，メニューから［**一致する次の文字列を置換**］を選択します。

［検索と置換］ダイアログ・ボックスが開きます。［**検索する単語**］ボックスに，置換対象の値または引数が表示されます。［**置換後の単語**］ボックスに，括弧にはさまれたパラメータ名が表示されます。

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



3 ［**置換**］または［**すべて置換**］をクリックします。

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

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

元の文字列の復元

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

パラメータの元の値を復元するには，次の手順を実行します。

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

➤ ［ツリー ビュー］：

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

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

元の引数が復元されます。

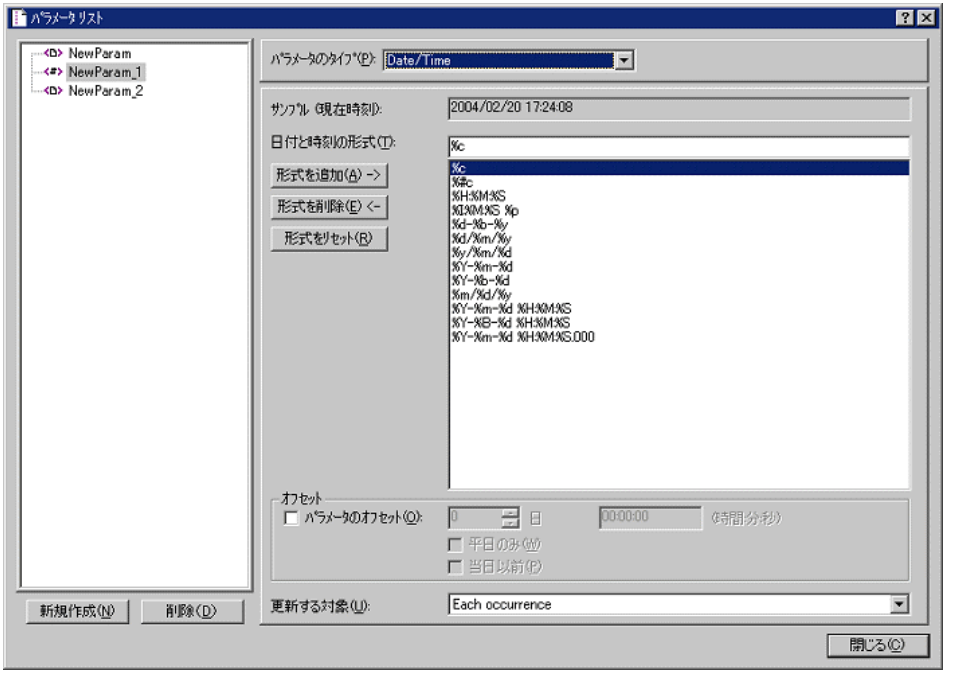
パラメータ・リストの使用

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

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

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

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



パラメータのプロパティを変更するには，次の手順を実行します。

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

パラメータ・プロパティの設定方法の詳細ついては，第 11 章「パラメータのプロパティの設定」および第 10 章「ファイル，テーブル，および XML パラメータ・タイプ」を参照してください。

新しいパラメータを作成するには，次の手順を実行します。

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

2 新規パラメータの名前を入力して，Enter キーを押します。

注：unique というパラメータ名は使用しないでください。VuGen によってすでに使用されています。

---

3 パラメータのタイプとプロパティを設定します。

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

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

既存のパラメータを削除するには，次の手順を実行します。

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

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

3 ［**はい**］をクリックします。

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

パラメータ化オプションの設定

パラメータ化オプションは，VuGen の［一般オプション］ウィンドウの［パラメータ化］タブで設定します。これらのオプションは次の項目を対象とします。

- パラメータの括弧
- グローバル・ディレクトリ

パラメータの括弧

パラメータを仮想ユーザ・スクリプトに挿入すると，VuGen によってパラメータ名の前後にパラメータ用の括弧が追加されます。Web または WAP スクリプトの標準の括弧は中括弧 { } です。次に例を示します。

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

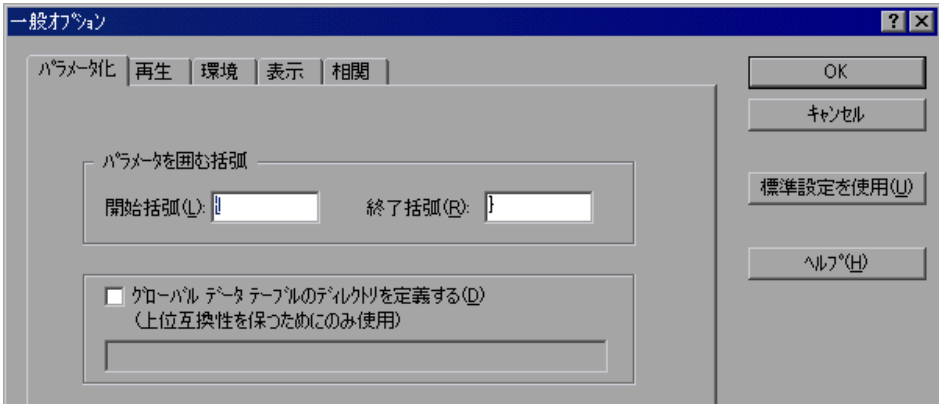
1 文字または複数の文字で構成される文字列を指定して，パラメータの括弧のスタイルを変更できます。スペース以外のすべての文字を使用できます。

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

パラメータの括弧のスタイルを変更するには，次の手順を実行します。

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

2 ［**パラメータ化**］タブを選択し，使用する括弧を入力します。



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

グローバル・ディレクトリ

このオプションは，VuGen の前のバージョンとの互換性を保つためにだけ提供されています。以前のバージョン（4.51 またはそれ以前のバージョン）では，新しいデータ・テーブルを作成するときに，ローカルかグローバルかを指定していました。ローカルのテーブルは現在の仮想ユーザ・スクリプトのディレクトリに保存され，スクリプトを実行している仮想ユーザだけが使用できます。グローバルなテーブルはすべての仮想ユーザ・スクリプトで使用できます。グローバル・ディレクトリはローカル・ドライブ上にもネットワーク・ドライブ上にも置けます。スクリプトを実行するすべてのマシンから，グローバル・ディレクトリを利用できることを確認してください。グローバル・テーブルの場所は，［一般オプション］ダイアログ・ボックスを使っていつでも変更できます。

VuGen の新しいバージョンでは，［パラメータのプロパティ］ダイアログ・ボックスまたは［パラメータ リスト］ダイアログ・ボックスの中でデータ・テーブルの場所を指定します。VuGen は，標準設定のスクリプト・ディレクトリや，ネットワーク上の別のディレクトリなど，指定した任意の場所のデータを取得できます。詳細については，130 ページ「データ・ファイル」を参照してください。

標準設定では，［**グローバル データ テーブルのディレクトリを定義する**］オプションは無効になっています。

グローバル・ディレクトリを設定するには，次の手順を実行します。

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

2 ［**パラメータ化**］タブを選択します。

3 ［**グローバル データ テーブルのディレクトリを定義する**］チェック・ボックスを選択し，グローバル・データ・テーブルがあるディレクトリを指定します。

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

第 10 章

ファイル，テーブル，および XML パラメータ・タイプ

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

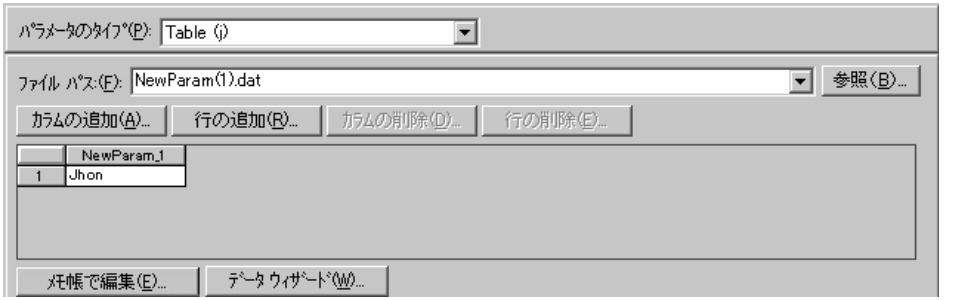
本章では，次の項目について説明します。

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

データ・ファイルまたはデータ・テーブルの選択または作成

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

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



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

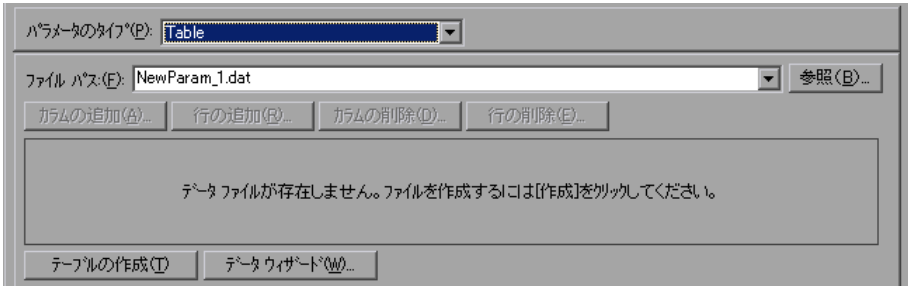
1 **［パラメータのプロパティ］ダイアログ・ボックス，または［パラメータ リスト］を開きます。**

手順の詳細については，133 ページ「パラメータのプロパティの定義」を参照してください。

2 **テーブルを選択するか，または新しいテーブルを作成します。**

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

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



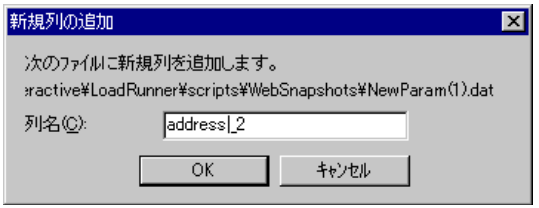
データ・ファイルが開き，最初の 100 行が表示されます。すべてのデータを表示するには，［**メモ帳で編集**］をクリックし，［メモ帳］でデータを表示します。

注：グローバル・ディレクトリを指定することもできます。グローバル・ディレクトリは，VuGen の前のバージョンとの互換性を保つためにだけ提供されています。詳細については，141 ページ「グローバル・ディレクトリ」を参照してください。

➤ 既存のデータベースからデータをインポートするには，［**データ ウィザード**］をクリックし，ウィザードの指示に従います。詳細については，146 ページ「既存のデータベースからのデータのインポート」を参照してください。

3 カラムと行を テーブルに追加します。

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

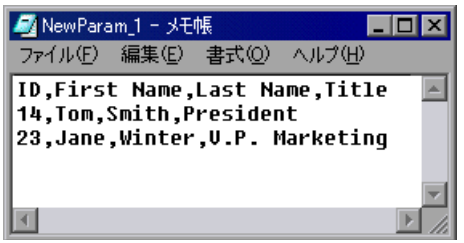


➤ テーブルに行を追加するには，［**行の追加**］を選択します。

4 データ・ファイルを編集します。

➤ 任意のセルの中で値を入力します。

➤ データ・ファイルをメモ帳の中から編集するには，［**メモ帳で編集**］をクリックします。［メモ帳］が開いて，1 行目にパラメータ名，2 行目にパラメータの最初の値が表示されます。追加のカラム名と値をファイルに入力します。カンマやタブなどの区切り文字を使用してカラムの区切りを示します。テーブルの行ごと（新しいデータ行ごと）に改行します。



既存のデータベースからのデータのインポート

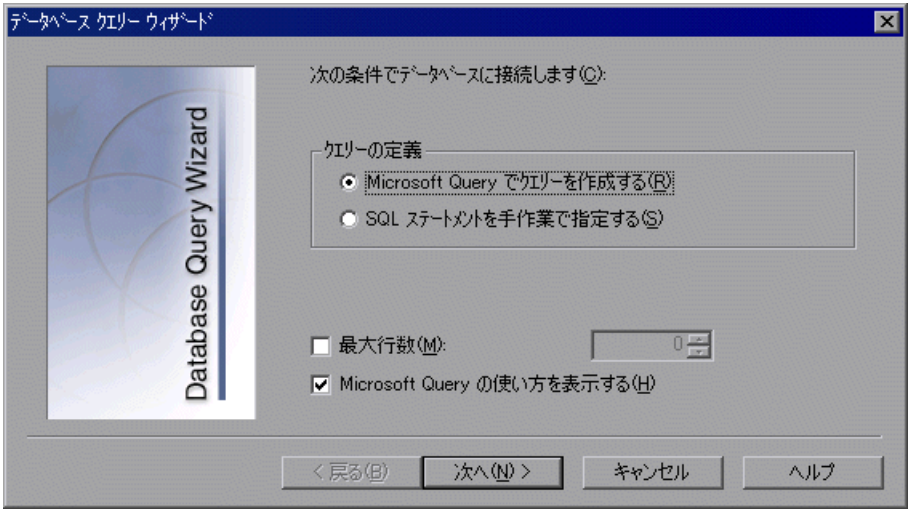
VuGen では，データベースからデータをインポートし，パラメータの値として使用できます。データをインポートする方法は 2 つあります。

➤ 新規クエリの作成

➤ SQL ステートメントの指定

VuGen では，ウィザードを実行し，その指示に従ってデータベースからデータをインポートすることができます。ウィザードで，データのインポート方法を指定します。MS Query を使って新しいクエリを作成するか，SQL ステートメントを指定します。データをインポートすると，**.dat** という拡張子を持つファイルに保存され，通常のパラメータ・ファイルとして利用できるようになります。

データベースのインポートを開始するには，まず［パラメータ リスト］ダイアログ・ボックス（［**仮想ユーザ**］＞［**パラメータ リスト**］）の［**データ ウィザード**］をクリックします。［データベース クエリ ウィザード］が開きます。



新規クエリの作成

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

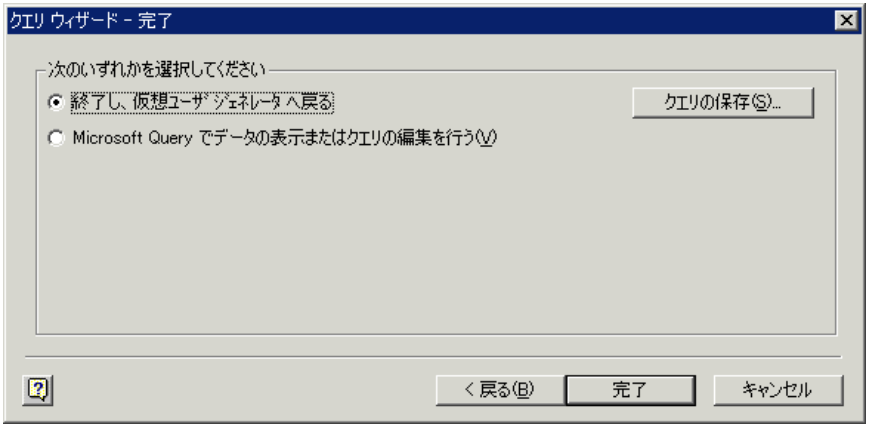
新規クエリを作成するには，次の手順を実行します。

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

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

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

4 データのインポートが終了したら，［**終了し，仮想ユーザ ジェネレータへ戻る**］を選択し，［**完了**］をクリックします。データベース・レコードが［パラメータ リスト］ボックスにデータ・ファイルとして表示されます。



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

5 データの割り当てプロパティを設定します。詳細については，150 ページ「ファイル・パラメータ・タイプのプロパティ設定」を参照してください。

SQL ステートメントの指定

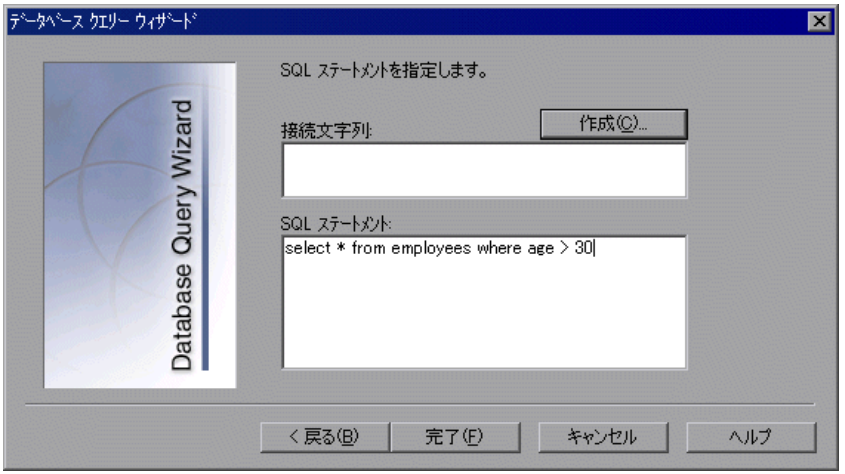
データベース接続と SQL ステートメントの指定を行うには，次の手順を実行します。

1 ［**SQL ステートメントを手作業で指定する**］を選択します。［**次へ**］をクリックします。

2 新規接続文字列を指定するために，［**作成**］をクリックします。［データ ソースの選択］ウィンドウが開きます。

3 既存のデータ・ソースを選択するか，［**作成**］を選択してデータ・ソースを新規作成します。ウィザードの指示に従って，ODBC データ・ソースを作成します。作業が終わると，接続文字列が［**接続文字列**］ボックスに表示されます。

4 ［**SQL ステートメント**］ボックスに，SQL ステートメントを入力するか貼り付けます。



5 ［**完了**］をクリックすると，SQL ステートメントが処理され，データがインポートされます。データベース・レコードが［パラメータ リスト］ボックスにデータ・ファイルとして表示されます。

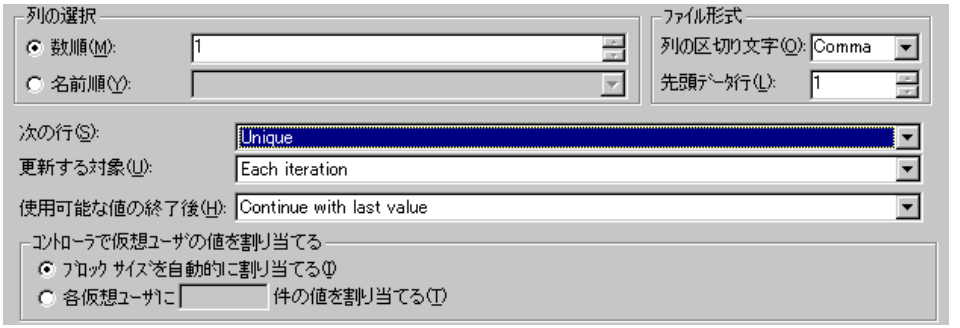
6 データの割り当てプロパティを設定します。詳細については，150 ページ「ファイル・パラメータ・タイプのプロパティ設定」を参照してください。

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

注 : パラメータのプロパティは，［パラメータ リスト］ダイアログ・ボックスでも設定できます。左側の表示枠でパラメータを選択し，右側の表示枠でそのプロパティを指定します。詳細については，138 ページ「パラメータ・リストの使用」を参照してください。

ファイル・パラメータ・タイプのプロパティ設定

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



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

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

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

カラム名で指定するには，［**名前順**］を選択して，リストからカラム名を選びます。カラム・ヘッダーは，各カラムの最初の行（0 行）にあります。カラム番号が変わる可能性がある場合，あるいはヘッダーがない場合は，カラムの指定にカラム名を使用します。

2 ［**ファイル形式**］セクションの［**列の区切り文字**］ボックスに，カラムの区切り文字を入力します。区切り文字とは，テーブルのカラムを区切るために使用する文字です。カンマ（Comma），タブ（Tab），またはスペース（Space）を指定できます。

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

4 ［**次の行**］リストからデータの割り当て方法を選択して，仮想ユーザが仮想ユーザ・スクリプト実行時にファイル・データをどのように選択するかを指定します。選択肢としては，［**Sequential**］（順次），［**Random**］（ランダム），または［**Unique**］（一意）があります。詳細については，154 ページ「ファイル・タイプまたはテーブル・タイプのパラメータにデータを割り当てる方法の選択」を参照してください。

5 ［**更新する対象**］リストから更新オプションを選択します。「**Each Iteration**」（反復ごと），「**Each Occurrence**」（すべての出現），「**Once**」（1 回のみ）から選択します。詳細については，156 ページ「ファイル・タイプまたはテーブル・タイプのパラメータに対するデータの割り当て方法と更新方法の選択」を参照してください。

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

- ［**使用可能な値の終了後**］：一意の値がなくなった場合の処理として，「**Abort Vuser**」（仮想ユーザを中止），「**Continue in a cyclic manner**」（循環して続行する），「**Continue with last value**」（最後の値で続行する）のいずれかを指定します。
- ［**コントローラで仮想ユーザの値を割り当てる**］（LoadRunner のユーザのみ）：仮想ユーザに手作業でデータ・ブロックを割り当てるかどうか指定します。コントローラにブロック・サイズを自動的に割り当てさせるか，必要な値の数を指定します。［**ブロック サイズを自動的に割り当てる**］または［**各仮想ユーザに X 件の値を割り当てる**］を選択します。後者の場合には，割り当てる値の数を指定します。

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

テーブル・パラメータ・タイプのプロパティ設定

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

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

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

すべてのカラムを選択するには，［**すべて選択する**］を選択します。

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

2 ［**列の区切り文字**］ボックスに，カラムの区切り文字を入力します。区切り文字とは，テーブルのカラムを区切るために使用する文字です。カンマ（Comma），タブ（Tab），またはスペース（Space）を使用できます。

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

注：これは［**更新する対象**］フィールドを［**Each iteration**］（反復ごと）に設定したときのみ有効です。［**更新する対象**］が［**Once**］（1 回のみ）に設定されている場合は，すべての反復で同じ行が使用されます。手順 8 を参照してください。

---

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

5 データ表現の行区切り文字を［**ログ表示に使用する行区切り文字**］ボックスで指定します。この区切り文字は出力ログで行を区別するのに使用されます。パラメータ置換ログを有効にしている場合は，置換された値が再生ログに送信されます。再生ログ内の行区切り文字は，新しい行を示します。

6 ［**行数が不足する場合の対処法**］ボックスで，反復のための十分な行数がテーブルにないときの処理方法を指定します。たとえば，設定するテーブルの行数が 3 で，データには 2 行しかないとします。2 つの行のみ設定する場合は，［**パラメータが受け取る行数は必要な数を下回ります** ］を選択します。［**次の行**］ボックスの［**Random**］または［**Sequential**］で指定した方法に従って次の行を反復処理して取得する場合は，［**「次の行を選択」の動作を使用する**］を選択します。

7 ［**次の行**］リストからデータの割り当て方法を選択して，仮想ユーザが仮想ユーザ・スクリプト実行時にテーブル・データをどのように選択するかを指定します。選択肢としては，［**Sequential**］（順次），［**Random**］（ランダム），または［**Unique**］（一意）があります。詳細については，154 ページ「ファイル・タイプまたはテーブル・タイプのパラメータにデータを割り当てる方法の選択」を参照してください。

8 ［**更新する対象**］リストから更新方法を選択します。「**Each Iteration**」（反復ごと）または「**Once**」（1 回のみ）から選択します。詳細については，156 ページ「ファイル・タイプまたはテーブル・タイプのパラメータに対するデータの割り当て方法と更新方法の選択」を参照してください。

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

- ［**使用可能な値の終了後**］：一意の値がなくなった場合の処理として，「**Abort Vuser**」（仮想ユーザを中止），「**Continue in a cyclic manner**」（循環して続行），「**Continue with last value**」（最後の値で続行）のいずれかを指定します。
- ［**コントローラで仮想ユーザの値を割り当てる**］（LoadRunner のユーザのみ）：仮想ユーザに手作業でデータ・ブロックを割り当てるかどうか指定します。コントローラにブロック・サイズを自動的に割り当てさせるか，必要な値の数を指定します。［**ブロック サイズを自動的に割り当てる**］または［**各仮想ユーザに X 件の値を割り当てる**］を選択します。後者の場合には，割り当てる値の数を指定します。

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

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

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

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

Sequential（順次）

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

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

Random（ランダム）

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

シナリオ，セッション・ステップ，またはビジネス・プロセス・モニタ・プロファイルを実行するときには，乱数の生成に使われるシード値を指定できます。各シード値は，テスト実行に使用されるランダム値のシーケンスを表します。同じシード値を使用すれば，シナリオまたはセッション・ステップの仮想ユーザには，同じ乱数値のシーケンスが割り当てられます。テスト実行時に問題が生じ，同じ乱数シーケンスを使ってテストを再実行したいときに，このオプションを有効にします。

詳細については，『**Mercury LoadRunner コントローラ・ユーザーズ・ガイド**』，**Tuning Console** のマニュアル，**Performance Center** のマニュアル，**Business Availability Center** のマニュアルを参照してください。

Unique（一意）

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

Unique を選択する場合は，すべての仮想ユーザとその反復回数に対して十分なデータがテーブルになければなりません。20 個の仮想ユーザがいて 5 回の反復を実行したい場合には，テーブルに少なくとも 100 個の一意の値が必要です。

データ・テーブルに十分な値がない場合は，処理方法を VuGen に指示できます。詳細については，150 ページ「ファイル・パラメータ・タイプのプロパティ設定」または 152 ページ「テーブル・パラメータ・タイプのプロパティ設定」を参照してください。

ファイル・タイプまたはテーブル・タイプのパラメータに対するデータの割り当て方法と更新方法の選択

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

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

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

例

テーブルまたはファイルに次の値が格納されているとします。

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

- ［**Sequential**］（順次）の方法を使用してデータを割り当てるよう選択した場合は，次のようになります。
  - 更新方法で［**Each Iteration**］を選択すると，すべての仮想ユーザが 1 回目の反復では Kim を，2 回目の反復では David を，3 回目の反復では Michael を取得する，という具合になります。
  - 更新方法で［**Each Occurrence**］を選択すると，すべての仮想ユーザが 1 回目の出現では Kim を，2 回目の出現では David を，3 回目の出現では Michael を取得する，という具合になります。
  - 更新方法で［**Once**］を選択すると，すべての仮想ユーザがすべての反復で Kim を取得します。

  データ・テーブルに十分な数の値がない場合，VuGen はテーブルの最初の値に戻り，テストの終了までその循環を繰り返します。
- ［**Random**］（ランダム）の方法を使用してデータを割り当てるよう選択した場合は，次のようになります。
  - 更新方法で［**Each Iteration**］を選択すると，仮想ユーザは反復ごとにテーブルから取得した乱数を使用します。
  - 更新方法で［**Each Occurrence**］を選択すると，仮想ユーザはパラメータの出現ごとに乱数を使用します。
  - 更新方法で［**Once**］を選択すると，すべての仮想ユーザがすべての反復で，最初にランダムに割り当てられた値を取得します。
- ［**Unique**］（一意）の方法を使用してデータを割り当てるよう選択した場合は，次のようになります。
  - 更新方法で［**Each Iteration**］を選択し，1 回のテスト実行で反復を 3 回実行するように指定した場合には，最初の仮想ユーザは，1 回目の反復で Kim を，2 回目の反復で David を，3 回目の反復で Michael を取得します。2 つ目の仮想ユーザは，Jane，Ron，Alice を取得します。3 つ目の仮想ユーザは，Ken，Julie，Fred を取得します。
  - 更新方法で［**Each Occurrence**］を選択すると，仮想ユーザはパラメータの出現ごとに，リストから取得した一意の値を使用します。

➤ 更新方法で［**Once**］を選択すると，最初の仮想ユーザはすべての反復で Kim を取得し，2 番目の仮想ユーザはすべての反復で David を取得する，という具合になります。

コントローラでの仮想ユーザの振る舞い（LoadRunner のみ）

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

➤ ［次の行］:［**Unique**］

➤ ［更新する対象］:［**Each iteration**］（反復ごと）

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

状況	継続期間	結果の動作
値よりも反復が多い	**［完了するまで実行する］**	一意の値が終了すると，各仮想ユーザは最後の値を使って継続しますが，値がもはや一意ではないことを示す警告メッセージが送信されます。
値よりも仮想ユーザが多い	**［Run indefinitely or Run for ...］**	仮想ユーザは一意の値を使い尽くします。その後にテストから「**Error: Insufficient records for param ＜パラメータ名＞ in table to provide the Vuser with unique data**」というエラー・メッセージが発行されます。これを回避するには，［パラメータのプロパティ］の［**使用可能な値の終了後**］オプションまたは［パラメータのプロパティ］の［**次の行**］メソッドを変更します。
2 つのパラメータのうちの 1 つが値を使い果たしたとき	**［Run indefinitely or Run for ...］**	値を使い果たしたパラメータは，2 番目のパラメータの値が一意でなくなるまで循環して継続します。

XML パラメータのプロパティの設定

特定の操作をエミュレートする Web サービス呼び出しを作成すると，その操作の引数に，多くの値を持つ複雑な構造が含まれる場合があります。XML タイプのパラメータを使用すれば，構造全体を 1 つのパラメータで置き換えることができます。XML タイプのパラメータの値セットをいくつか作成して，反復ごとに異なる値セットを割り当てることもできます。また，XML パラメータ・タイプは，配列や <any> 要素など複雑なスキーマ・タイプをサポートします。

注：XML パラメータ・タイプは，<choice> スキーマ・タイプ要素をサポートしていません。

本項では，次の項目について説明します。

- 新規 XML パラメータの作成
- XML パラメータのプロパティの変更

新規 XML パラメータの作成

XML パラメータは，［パラメータ リスト］で，または既存の記録済みスクリプトから作成できます。

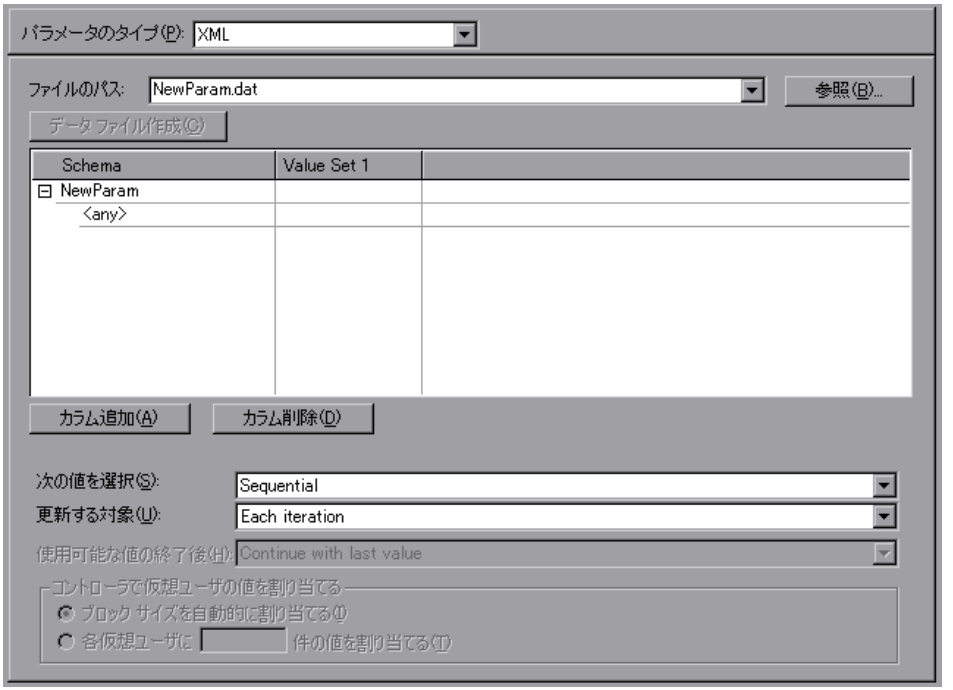
新規 XML パラメータの作成

新規 XML パラメータを作成するには，次の手順を実行します。

1 **［パラメータ リスト］を開いて，XML タイプの新規パラメータを作成します。**

手順の詳細については，133 ページ「パラメータのプロパティの定義」を参照してください。

2 XML パラメータに対するスキーマを定義します。



a ［Schema］カラムで，**<any>** を右クリックし，次のオプションのいずれかを選択します。

- ［**Add New Sub-Element**］：選択された要素の下にサブ要素を挿入します。
- ［**Insert Array Element**］：このオプションで選択した配列と同じ配列構造で新規の配列要素を挿入します。
- ［**Delete Array Element**］：配列要素全体を削除します。
- ［**Copy Row XPath**］：選択した要素の完全 XML パスをクリップボードにコピーします。

3 XML パラメータに対する値のセットを定義します。

［**Value Set**］カラムに，作成したスキーマに対応する値を挿入します。

追加の値セットを挿入するには，［**カラム追加**］をクリックし，新しいカラムに追加する値のセットを挿入します。

4 パラメータに対するデータの割り当て方法と更新オプションを定義します。

a ［**次の値を選択**］リストで，データの割り当て方法を選択して，仮想ユーザが仮想ユーザ・スクリプト実行時にファイル・データをどのように選択するかを指定します。選択肢としては，［**Sequential**］（順次），［**Random**］（ランダム），または［**Unique**］（一意）があります。詳細については，154 ページ「ファイル・タイプまたはテーブル・タイプのパラメータにデータを割り当てる方法の選択」を参照してください。

b ［**更新する対象**］リストから更新オプションを選択します。「**Each Iteration**」（反復ごと），「**Each Occurrence**」（すべての出現），「**Once**」（1 回のみ）から選択します。詳細については，156 ページ「ファイル・タイプまたはテーブル・タイプのパラメータに対するデータの割り当て方法と更新方法の選択」を参照してください。

c （上の手順 7 で）データの割り当て方法として［**Unique** ］（一意）を選択した場合，［**使用可能な値の終了後**］オプションおよび［**コントローラで仮想ユーザの値を割り当てる**］オプションが有効になります。

- ［**使用可能な値の終了後**］：一意の値がなくなった場合の処理として，「**Abort Vuser**」（仮想ユーザを中止），「**Continue in a cyclic manner**」（循環して続行する），「**Continue with last value**」（最後の値で続行する）のいずれかを指定します。
- ［**コントローラで仮想ユーザの値を割り当てる**］（LoadRunner のユーザのみ）：仮想ユーザに手作業でデータ・ブロックを割り当てるかどうか指定します。コントローラにブロック・サイズを自動的に割り当てさせるか，必要な値の数を指定します。［**ブロック サイズを自動的に割り当てる**］または［**各仮想ユーザに X 件の値を割り当てる**］を選択します。後者の場合には，割り当てる値の数を指定します。

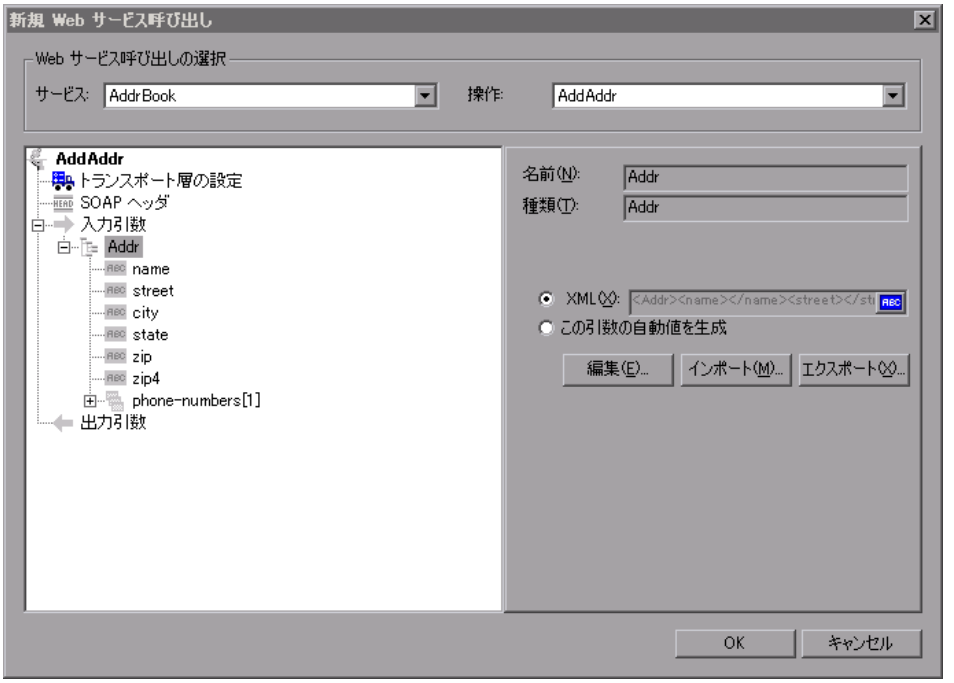
  これを追跡するには，［実行環境設定］の［ログ］で，［**拡張ログ**］＞［**パラメータ置換**］オプションを有効にします。十分なデータがない場合は，仮想ユーザ・ログに「**No more unique values for this parameter in table ＜テーブル名＞**」（＜テーブル名＞テーブルには，このパラメータのための一意の値がこれ以上ありません）という警告メッセージが表示されます。

5 ［パラメータ リスト］にパラメータを追加します。

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

Web サービス・スクリプトからの XML パラメータの作成

Web サービス呼び出しを追加すると，［新規 Web サービス呼び出し］ダイアログ・ボックスが開きます。左側のツリーで，［**入力引数**］の下にある，パラメータ化する複雑なデータ構造のルート要素を選択します。



記録済みサービスの操作または配列をパラメータ化するには，次の手順を実行します。

1 複合データ構造のルート要素を選択します。

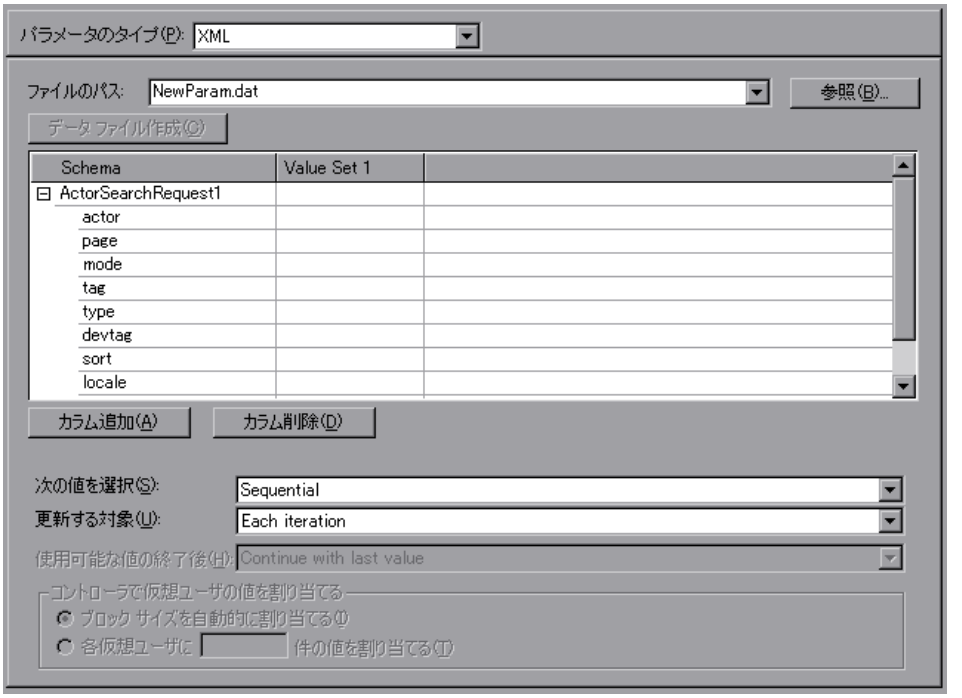
右側の表示枠には，引数の詳細が表示されます。

2 ［**XML**］を選択し，［**ABC**］ボタン をクリックします。［パラメータの選択または作成］ダイアログ・ボックスが開きます。

3 ［**パラメータ名**］ボックスにパラメータの名前を入力します。

4 ［**パラメータのタイプ**］ボックスでは，標準設定で［**XML**］が選択されます。選択されていない場合は，［**XML**］を選択します。

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



- ［ファイルのパス］ボックスには，引数データ・セットが格納される .dat ファイルの名前が表示されます。.dat ファイルには，パラメータに与えたのと同じ名前が割り当てられます。
- テーブルの［**Schema**］カラムには，Web サービス呼び出しを追加したときにツリーに表示された操作の引数が表示されます。

6 2 番目のカラム［**Value Set 1**］で，［**Schema**］カラムの引数のセットにそれぞれ対応する値のセットを入力します。

別の値セットを追加するには，［**カラム追加**］をクリックし，作成されたカラムの下に値を入力します。

7 ［**次の値を選択**］リストで，データの割り当て方法を選択して，仮想ユーザが仮想ユーザ・スクリプト実行時にファイル・データをどのように選択するかを指定します。選択肢としては，［**Sequential**］（順次），［**Random**］（ランダム），または［**Unique**］（一意）があります。詳細については，154 ページ「ファイル・タイプまたはテーブル・タイプのパラメータにデータを割り当てる方法の選択」を参照してください。

8 ［**更新する対象**］リストから更新オプションを選択します。「**Each Iteration**」（反復ごと），「**Each Occurrence**」（すべての出現），「**Once**」（1 回のみ）から選択します。詳細については，156 ページ「ファイル・タイプまたはテーブル・タイプのパラメータに対するデータの割り当て方法と更新方法の選択」を参照してください。

9 （上の手順 7 で）データの割り当て方法として［**Unique** ］（一意）を選択した場合，［**使用可能な値の終了後**］オプションおよび［**コントローラで仮想ユーザの値を割り当てる**］オプションが有効になります。

- ［**使用可能な値の終了後**］：一意の値がなくなった場合の処理として，「**Abort Vuser**」（仮想ユーザを中止），「**Continue in a cyclic manner**」（循環して続行する），「**Continue with last value**」（最後の値で続行する）のいずれかを指定します。
- ［**コントローラで仮想ユーザの値を割り当てる**］（LoadRunner のユーザのみ）：仮想ユーザに手作業でデータ・ブロックを割り当てるかどうか指定します。コントローラにブロック・サイズを自動的に割り当てさせるか，必要な値の数を指定します。［**ブロック サイズを自動的に割り当てる**］または［**仮想ユーザに X 件の値を割り当てる**］を選択します。後者の場合には，割り当てる値の数を指定します。

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

10 ［パラメータ リスト］にパラメータを追加します。

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

b ［パラメータの選択または作成］ダイアログ・ボックスで［OK］をクリックします。

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

XML パラメータのプロパティの変更

XML パラメータを作成した後で，Web サービスの［プロパティの設定］タブから，値のセットを変更できます。

XML パラメータのプロパティを変更するには，次の手順を実行します。

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

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

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

4 159 ページ「新規 XML パラメータの作成」の手順 5 での説明のとおり，パラメータ・プロパティを変更します。

第 11 章

パラメータのプロパティの設定

パラメータは，置き換えの対象となる情報の種類に基づいて定義されます。

本章では，次の項目について説明します。

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

パラメータのプロパティの設定について

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

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

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

テーブルまたはファイル・タイプ・パラメータのプロパティの定義方法の詳細については，第 10 章「ファイル，テーブル，および XML パラメータ・タイプ」を参照してください。

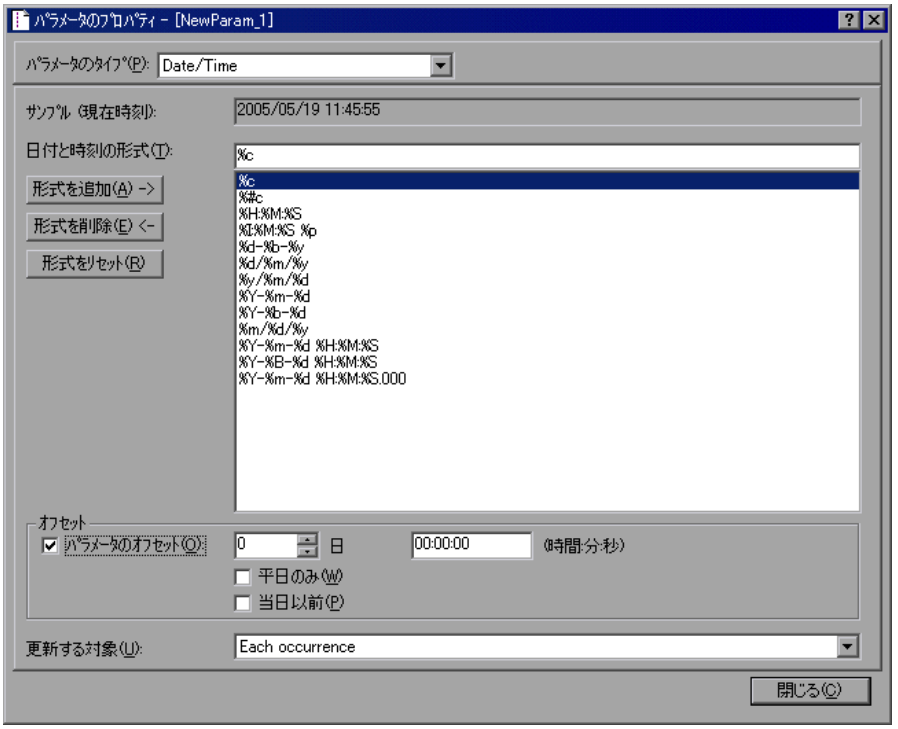
内部データ・パラメータ・タイプのプロパティの設定

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

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

Date/Time（日時）

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



VuGen では日時パラメータのオフセットを設定できます。たとえば，翌月の日付をテストする場合は，日付オフセットを 30 日に設定します。未来の時刻でのアプリケーションのテストを行う場合は，時刻オフセットを指定します。未来への前進オフセット（標準設定），または過去への後退オフセットを指定できます。また，平日のみの日付値を使用し，土曜日と日曜日を除外するように VuGen に指示することもできます。

次の表に日付と時刻の記号の意味を示します。

記号	説明
c	年月日（数字）と時刻
#c	年月日（文字列）と時刻
H	時間（24 時間表示）
I	時間（12 時間表示）
M	分
S	秒
p	午前（AM）または午後（PM）
d	曜日
m	月（01 ～ 12 までの数字）
b	月（省略した文字列。たとえば Dec）
B	月（省略しない文字列。たとえば December）
y	年（短い形式。たとえば 03）
Y	年（長い形式。たとえば 2003）

日付と時刻のパラメータのプロパティを設定するには，次の手順を実行します。

1 既存の日付と時刻の形式から選択するか，新しい形式を作成します。VuGen でどのように表示されるかは［**サンプル（現在時刻）**］ボックスで確認できます。パラメータ形式のカスタマイズの詳細については，180 ページ「パラメータ形式のカスタマイズ」を参照してください。

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

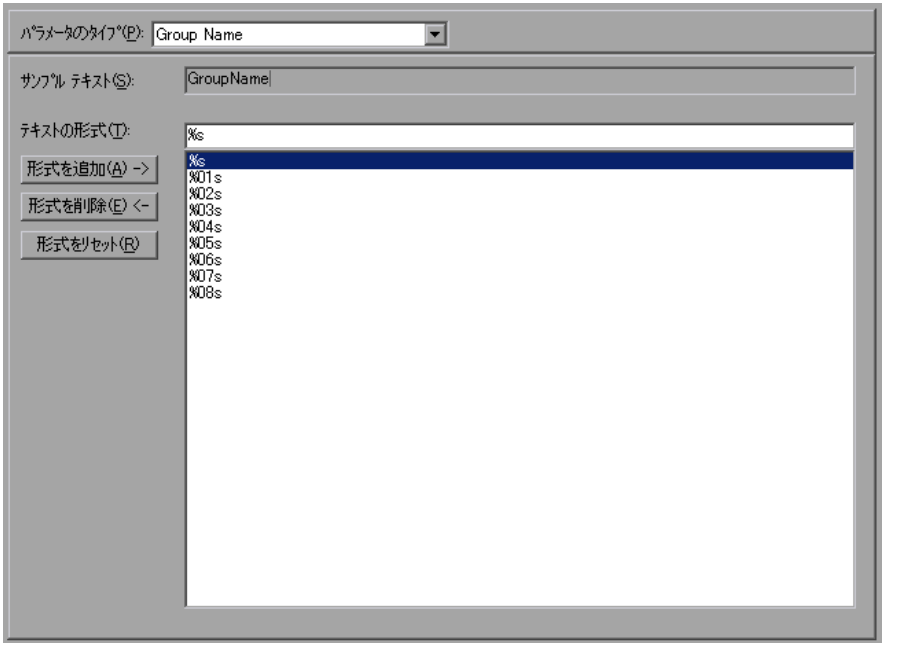
週末日を除く平日のみを使用するよう VuGen に指示するには，［**平日のみ**］を選択します。過去の日付をテストするために負のオフセットを指定するには，［**当日以前**］を選択します。

3 ［更新する対象］で「**Each occurrence**」（発生ごと），「**Each iteration**」（反復ごと），「**Once**」（一度）のどれかを選択し，仮想ユーザにパラメータ値の更新方法を指定します。詳細については，181 ページ「更新方法の選択」を参照してください。

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

Group Name（グループ名）

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



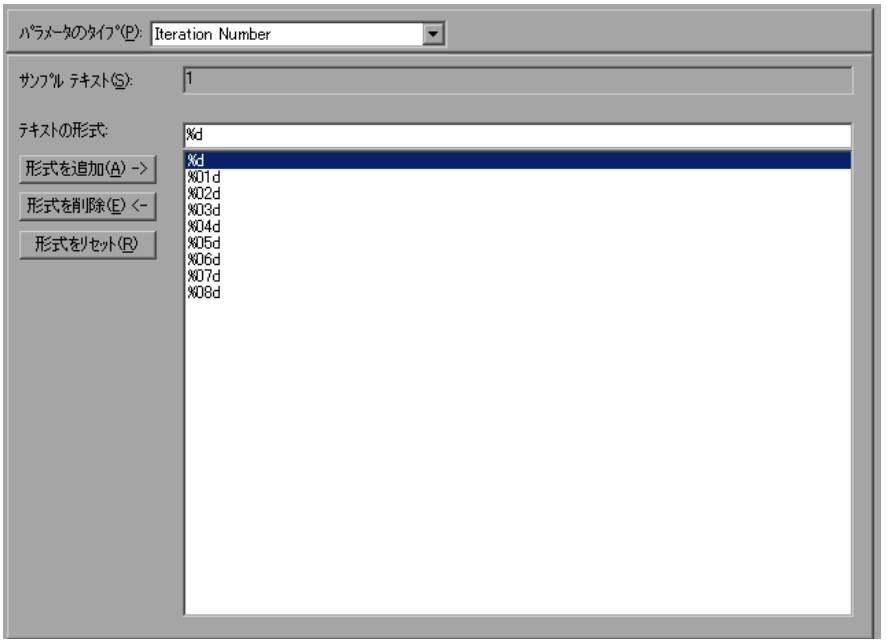
グループ名のパラメータのプロパティを設定するには，次の手順を実行します。

1 利用可能な既存の形式の中から 1 つを選択するか，新しい形式を作成します。形式を選択することによって，パラメータの文字列の長さを指定します。詳細については，180 ページ「パラメータ形式のカスタマイズ」を参照してください。

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

Iteration Number（反復回数）

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



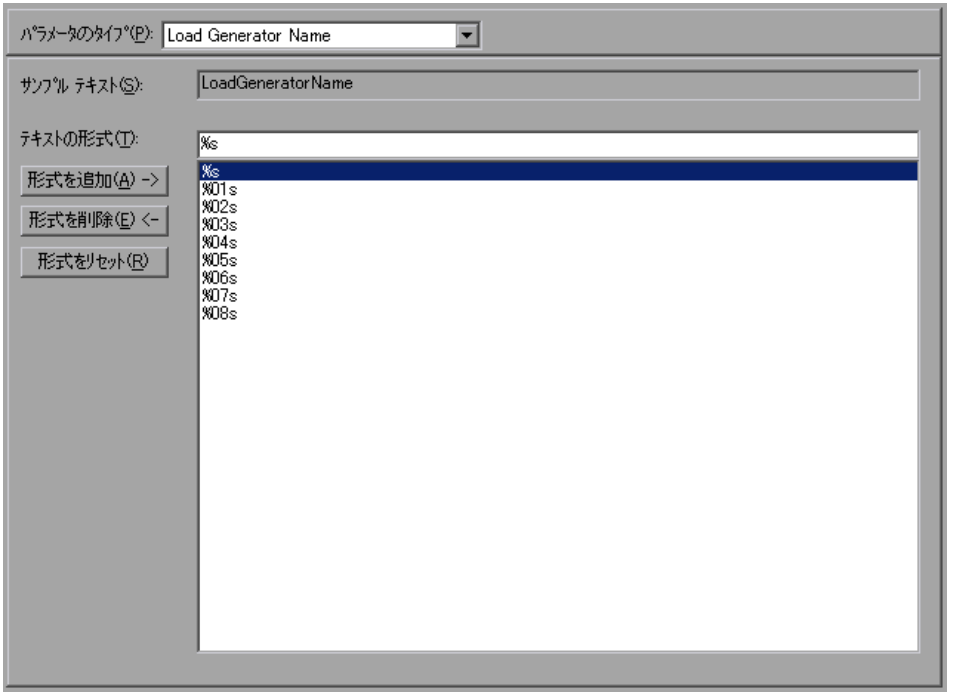
反復回数のパラメータのプロパティを設定するには，次の手順を実行します。

1 利用可能な既存の形式の中から 1 つを選択するか，新しい形式を作成します。形式を選択することによって，パラメータの文字列の長さを指定します。詳細については，180 ページ「パラメータ形式のカスタマイズ」を参照してください。

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

Load Generator Name（ロード・ジェネレータ名）

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



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

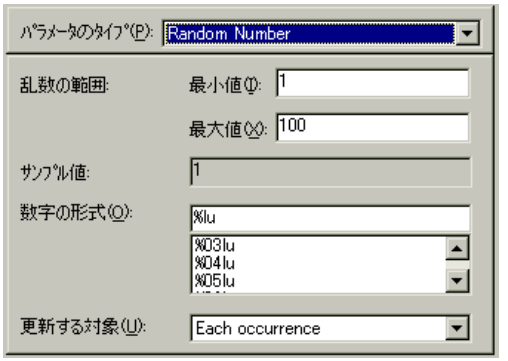
1 利用可能な既存の形式の中から 1 つを選択するか，新しい形式を作成します。形式を選択することによって，パラメータの文字列の長さを指定します。詳細については，180 ページ「パラメータ形式のカスタマイズ」を参照してください。

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

Random Number（乱数）

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

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



乱数のパラメータのプロパティを設定するには，次の手順を実行します。

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

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

3 ［更新する対象］で「**Each occurrence**」（発生ごと），「**Each iteration**」（反復ごと），「**Once**」（一度）のどれかを選択し，仮想ユーザにパラメータ値の更新方法を指定します。詳細については，181 ページ「更新方法の選択」を参照してください。

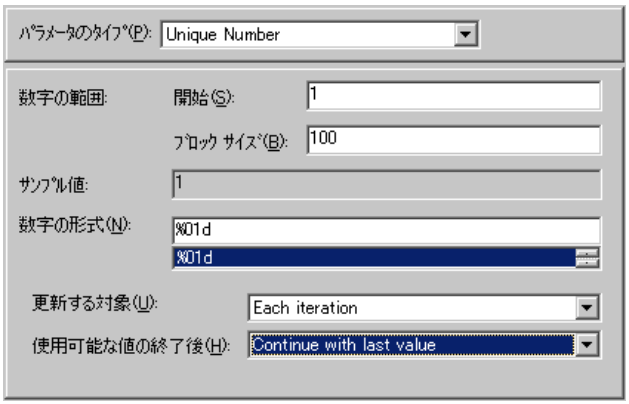
4 ［**閉じる**］をクリックして設定を適用し，［パラメータのプロパティ］ダイアログ・ボックスを閉じます。

Unique Number（一意の数）

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

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

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



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

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

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

一意のパラメータの定義後は，同じパラメータ・ファイルを指定することによって，ほかのスクリプトでも一意のパラメータを使用できます。このパラメータは，元のパラメータに割り当てられたすべてのプロパティを保持します。

一意の数のパラメータのプロパティを設定するには，次の手順を実行します。

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

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

3 ［更新する対象］で「**Each occurrence**」（発生ごと），「**Each iteration**」（反復ごと），「**Once**」（一度）のどれかを選択し，仮想ユーザにパラメータ値の更新方法を指定します。詳細については，181 ページ「更新方法の選択」を参照してください。

4 一意の値がなくなった場合には，［**使用可能な値の終了後**］ボックスで，「**Abort Vuser**」（仮想ユーザを中止），「**Continue in a cyclic manner**」（循環して継続），「**Continue with last value**」（最後の値で継続するー標準設定）から指定することができます。

注：（LoadRunner ユーザの方へ）コントローラでシナリオのスケジュールを設定する際，［**使用可能な値の終了後**］オプションは，スケジュール・ビルダの［継続時間］タブの［**実行時間：（時間：分：秒）**］オプションに対してのみ適用されます。［**完了するまで実行する**］オプションに対しては無視されるので注意してください。

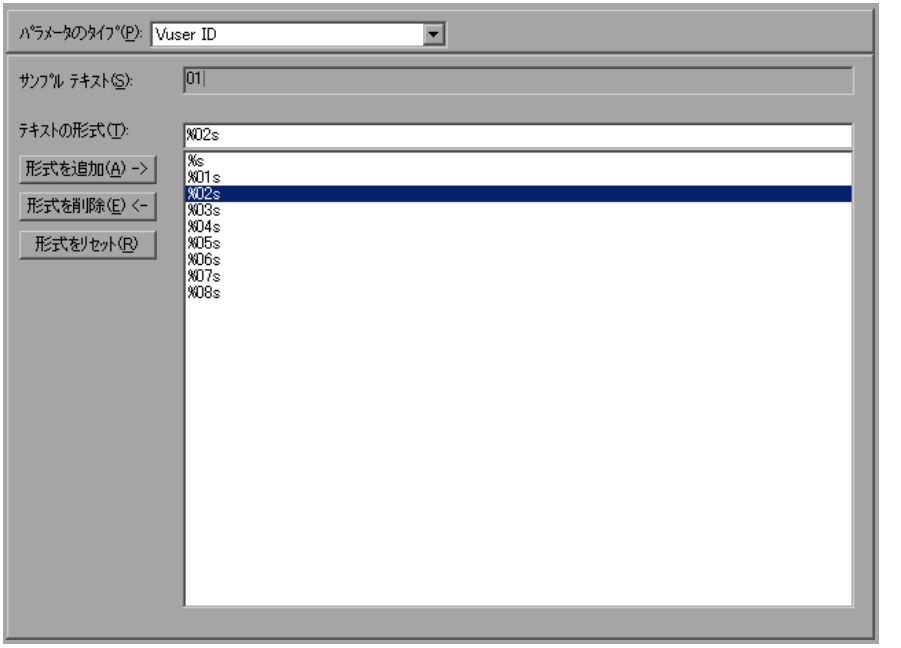
---

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

Vuser ID（仮想ユーザ）

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

［Vuser ID］（仮想ユーザ ID）を選ぶと，パラメータは仮想ユーザに割り当てられる ID 番号で置き換えられます。この ID 番号は，シナリオ実行中はコントローラによって割り当てられ，セッション・ステップ実行中はコンソールによって割り当てられます。VuGen からスクリプトを実行する場合，仮想ユーザ ID は常に -1 です。



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

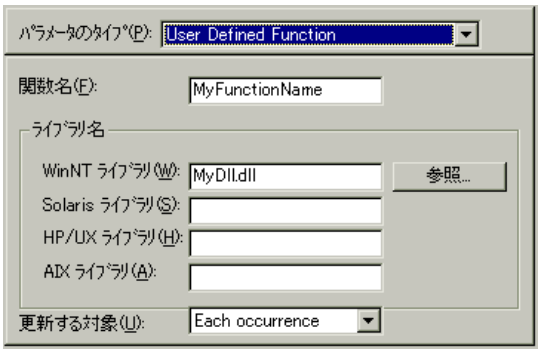
仮想ユーザ ID タイプのパラメータのプロパティを設定するには，次の手順を実行します。

1 利用可能な既存の形式の中から 1 つを選択するか，新しい形式を作成します。形式を選択することによって，パラメータの文字列の長さと構造を指定します。詳細については，180 ページ「パラメータ形式のカスタマイズ」を参照してください。

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

ユーザ定義関数のプロパティの設定

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



ユーザ定義関数のプロパティを設定するには，次の手順を実行します。

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

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

3 値の更新方法を選択します。ユーザ定義関数の更新方法の詳細については，181 ページ「更新方法の選択」を参照してください。

パラメータ形式のカスタマイズ

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

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

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

実際のパラメータ値の前後にテキスト文字列を指定できます。

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

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

パラメータ・タイプに形式を追加するには，次の手順を実行します。

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

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

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

形式を削除するには，次の手順を実行します。

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

一連の元の形式を復元するには，次の手順を実行します。

［**形式をリセット**］をクリックします。

更新方法の選択

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

- Each Occurrence（発生ごと）
- Each Iteration（反復ごと）
- Once（一度）

Each Occurrence（発生ごと）

「**Each occurrence**」（発生ごと）の方法を選択すると，仮想ユーザはパラメータが現われるたびに新しい値を使います。パラメータを使っているステートメントどうしが関連していない場合に有用です。たとえば，ランダムなデータを使用する場合には，パラメータが現れるたびに異なる値を使うと有効です。

Each Iteration（反復ごと）

「**Each iteration**」（反復ごと）の方法を選択すると，仮想ユーザはスクリプトで反復ごとに新しい値が使用されます。そのパラメータが 1 つのスクリプトに複数回現れる場合，仮想ユーザは 1 回の反復でそのパラメータが現れるたびに同じ値を使います。これは，パラメータを使うステートメントどうしが関連している場合に有用です。

注：独自に反復カウントを使って反復処理を行うアクション・ブロックを作成した場合，そこにパラメータが含まれていて，VuGen に反復ごとに値を更新するよう指定した場合でも，VuGen が対象とする反復は当該ブロックの反復ではなく，スクリプト全体のグローバルな反復なので，ブロックの反復ではパラメータが更新されません。アクション・ブロックの詳細については，195 ページ「アクション・ブロックの作成」を参照してください。

Once（一度）

「**Once**」（一度）の方法を選択すると，仮想ユーザはシナリオまたはセッション・ステップの実行時に，そのシナリオまたはセッション・ステップに対して一度だけパラメータの値を更新します。仮想ユーザはすべての反復でパラメータのすべての出現箇所で同じパラメータ値を使います。このタイプは日付と時刻を使う場合に有用です。

第 12 章

ステートメントの相関

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

本章では，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
- C 仮想ユーザ用の相関関数の使用
- Java 仮想ユーザ用の相関関数の使用法
- WDiff を使った仮想ユーザ・スクリプトの比較
- 保存したパラメータの変更

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

ステートメントの相関について

ステートメントを相関させる主な目的を以下に示します。

コードを簡素化または最適化するため

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

動的データの生成のため

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

固有のデータ・レコードに対応するため

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

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

スクリプトの相関の主な手順は次のとおりです。

1 **相関させる値を特定する。**

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

あるいは，VuGen に付属の **WDiff** ユーティリティを使って，スクリプト内の不一致を調べることができます。詳細については，188 ページ「WDiff を使った仮想ユーザ・スクリプトの比較」を参照してください。

2 **結果を保存する。**

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

3 **保存した値を参照する。**

クエリまたはステートメント内の定数を保存された変数で置換します。

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

- Java 言語仮想ユーザについては，第 28 章「Java スクリプトの相関」を参照してください。
- データベース仮想ユーザについては，第 33 章「データベース仮想ユーザ・スクリプトの相関」を参照してください。
- Web 仮想ユーザについては，第 57 章「Web 仮想ユーザ・スクリプトの相関ルールの設定」を参照してください。
- COM 仮想ユーザについては，第 40 章「COM 仮想ユーザ・スクリプトの詳細」を参照してください。

C 仮想ユーザ用の相関関数の使用

固有の相関関数を持たないプロトコルのステートメントを相関させるには，C 仮想ユーザ相関関数を使うことができます。これらの関数は C 言語に基づくすべての仮想ユーザに対して使用できます。これらの関数を使って，文字列をパラメータに保存し，必要に応じて取り出せます。Java，CORBA-Java，RMI-Java 仮想ユーザ用の同様の関数の詳細については，187 ページ「Java 仮想ユーザ用の相関関数の使用法」を参照してください。

lr_eval_string	指定されたパラメータに一致するパラメータをすべて現在の値で置き換えます。
lr_save_string	パラメータに NULL 終了文字列を保存します。
lr_save_var	可変長文字列をパラメータに保存します。

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

lr_eval_string の使用法

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

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

lr_save_string の使用法

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

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

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

Java 仮想ユーザ用の相関関数の使用法

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

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

CORBA-Java または RMI-Java スクリプトを記録するときに，VuGen は内部で相関を実行します。詳細については，第 28 章「Java スクリプトの相関」を参照してください。

Java 文字列関数の使用法

Java 仮想ユーザ・スクリプトをプログラミングするときに，Java 仮想ユーザ文字列関数を使用してスクリプトを相関させることができます。

次の例では，**lr.eval_int** を使って，変数 **ID_num** をその値で置き換えています。ID_num は，スクリプトの中で先に定義されているものとします。

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

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

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

WDiff を使った仮想ユーザ・スクリプトの比較

相関させる値を判断するのに役立つツールとして，**WDiff** があります。このツールを使用することにより，記録したスクリプトと結果を比較し，相関させるべき値を判断することができます。

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

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

WDiff を使って相関を検索するには，次の手順を実行します。

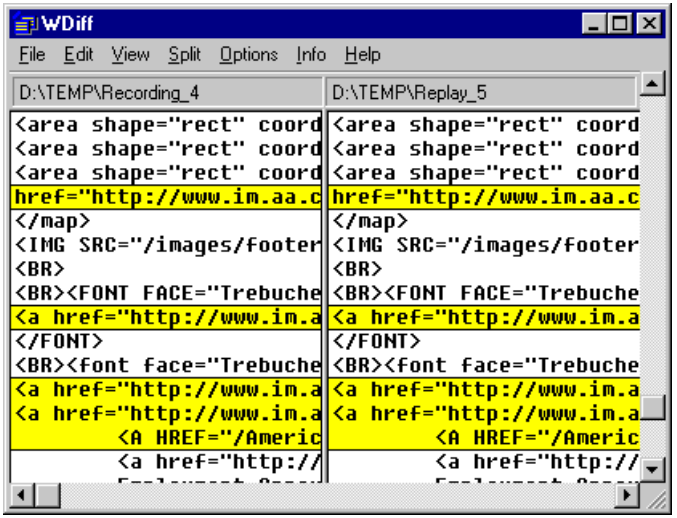
1 スクリプトを記録し，保存します。

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

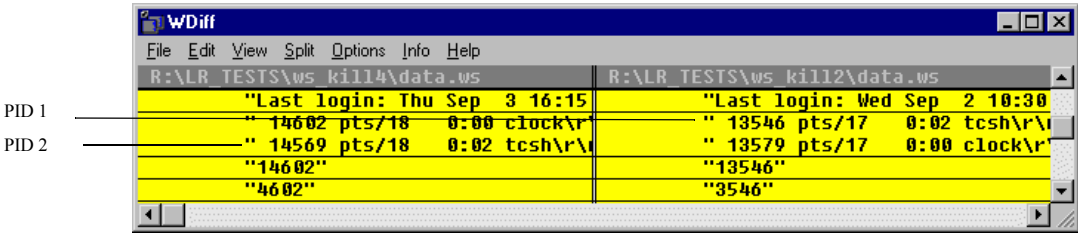
3 比較したい部分（**Actions** や **data.ws** など）を選択します。

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

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



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



7 どの値を相関させるかを判断します。

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

相関の以降の作業については，該当する項を参照してください。

➤ Java 言語仮想ユーザについては，第 28 章「Java スクリプトの相関」を参照してください。

- データベース仮想ユーザについては，第 33 章「データベース仮想ユーザ・スクリプトの相関」を参照してください。
- Web 仮想ユーザについては，第 57 章「Web 仮想ユーザ・スクリプトの相関ルールの設定」を参照してください。
- COM 仮想ユーザについては，第 40 章「COM 仮想ユーザ・スクリプトの詳細」を参照してください。
- Tuxedo 仮想ユーザについては，第 75 章「Tuxedo 仮想ユーザ・スクリプトの作成」を参照してください。
- WinSock 仮想ユーザについては，第 36 章「Windows Sockets データを使った作業」を参照してください。

保存したパラメータの変更

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

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

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

第 13 章

実行環境の設定

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

本章では，次の項目について説明します。

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

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

実行環境の設定について

仮想ユーザ・スクリプトを記録した後，そのスクリプトの実行環境を設定できます。実行環境の設定は，スクリプトの実行方法を規定します。これらの設定は，仮想ユーザ・スクリプトのディレクトリにある **default.cfg** ファイルに格納されます。実行環境の設定は，VuGen，コントローラ，Tuning Console，またはビジネス・プロセス・モニタを使ってスクリプトを実行するときに，仮想ユーザに適用されます。

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

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

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

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

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

- ［**思考遅延時間**］：VuGen では無効になっていますが，コントローラでは記録されたとおりに再生されます。
- ［**ログ**］：VuGen では標準で有効になっていますが，コントローラでは無効になっています。
- ［**HTML 以外のリソースをダウンロードする**］：VuGen とコントローラの両方で有効になっています。

本章で説明する一般的な実行環境設定は，すべてのタイプの仮想ユーザ・スクリプトに適用されます。次の設定があります。

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

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

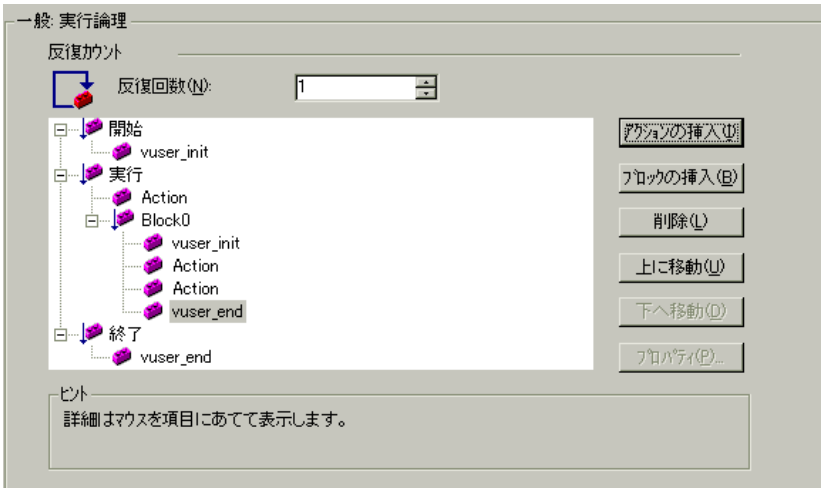
実行論理の設定（マルチ・アクション）

注：次の項の内容は，複数のアクションで動作するプロトコルを対象としています。［実行環境設定］の［一般］の下に［**実行論理**］ノードがあれば，複数アクション対応のプロトコルです。シングル・アクション・プロトコルについては，200 ページ「ペースの設定と実行論理オプションの設定（シングル・アクション）」を参照してください。

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

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

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



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

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

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

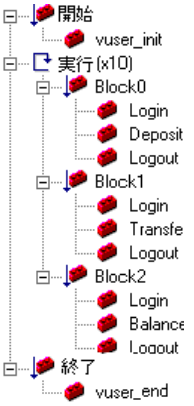
- **アクション・ブロック**：アクション・ブロックは，スクリプト内のアクションのグループです。各ブロックのプロパティ（順序，反復，重み付け）を個別に設定できます。
- **順番**：スクリプト内のアクションの順序を設定できます。アクションを順番に実行するか，ランダムに実行するかを指定することもできます。

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

アクション・ブロックの作成

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

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



各ブロックの順序，反復を個別に設定します。

アクションとアクション・ブロックを設定するには，次の手順を実行します。

1 記録またはプログラミングによって，必要なアクションをすべて作成します。

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

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

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

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

6 ブロックに追加するアクションごとに，手順 3 を繰り返します。

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

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

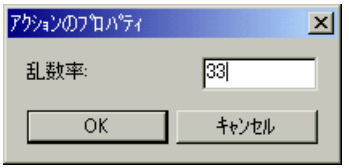
9 反復回数とアクションの実行論理を設定するには，［**プロパティ**］をクリックします。各アクションまたはアクション・ブロックのプロパティ・ダイアログ・ボックスが開きます。

10 ［**実行論理**］リストから［**順次**］または［**ランダム**］を選択します。これにより VuGen にアクションを順番に実行するかランダムに実行するかを指定します。

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

12 ［**OK**］をクリックします。

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



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

14 プロパティを設定する各要素に対して，上記の手順を繰り返します。

ペースの設定

注：次の項の内容は，複数のアクションで動作するプロトコルを対象としています。［実行環境設定］の［一般］の下に［**実行論理**］ノードがあれば，複数アクション対応のプロトコルです。シングル・アクション・プロトコルについては，200 ページ「ペースの設定と実行論理オプションの設定（シングル・アクション）」を参照してください。

［実行環境設定］の［ペースの設定］を設定することで，反復の間隔を変えることができます。ペースは，アクションの反復の間で待機する時間を仮想ユーザに指示します。各反復の開始間隔として次のオプションを指定できます。

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

  スクリプト実行時に，反復が終了してから次の反復を開始するまでに仮想ユーザが待機した時間は，VuGen の実行ログに示されます。
- ［**一定**］/［**値の範囲**］［間隔］［各 X 秒］：反復と反復の間の時間を指定します。秒単位で固定時間または時間の範囲を定義します。たとえば，新しい反復を 30 秒ごとに開始したり，30 ～ 45 秒ごとのランダムな間隔で開始したりするように指定できます。反復は，直前の反復が終了してからのみ開始します。

  スケジュールが設定された各反復は，前回の反復が完了した後に開始されます。スクリプト実行時に，反復が終了してから次の反復を開始するまでに仮想ユーザが待機した時間は，VuGen の実行ログに示されます。

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

  - 最初の反復が 3 秒かかると，仮想ユーザは次の反復で 1 秒待機します。
  - 最初の反復が 2 秒かかると，仮想ユーザは 2 秒待機します。

➤ 最初の反復が 8 秒かかると，次の反復は，最初の反復が開始してから 8 秒後に開始します。VuGen の実行ログには，反復のペースを守れなかったことを示すメッセージが表示されます。

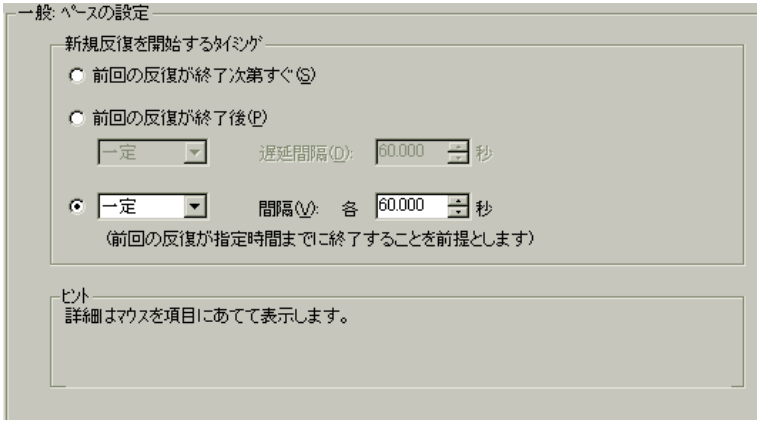
ペースの設定のオプションの詳細については，199 ページ「実行環境のペースの設定（マルチ・アクション）」を参照してください。

実行環境のペースの設定（マルチ・アクション）

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

反復の間隔を設定するには，次の手順を実行します。

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



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

➤ ［前回の反復が終了次第すぐ］

➤ ［前回の反復が終了後］

➤ ［一定］または［値の範囲］

3 ［**前回の反復が終了後**］オプションには次の設定があります。

➤ 遅延の種類を［**一定**］か［**値の範囲**］から選択します。

- ［一定］で値を指定するか，最小 / 最高の乱数値を使用して値の範囲を指定します。

4 ［**各 … 秒**］オプションには次の設定があります。

- 間隔の種類を［**一定**］か［**値の範囲**］から選択します。
- ［一定］で値を指定するか，最小 / 最高の乱数値を使用して値の範囲を指定します。

5 ［**OK**］をクリックします。

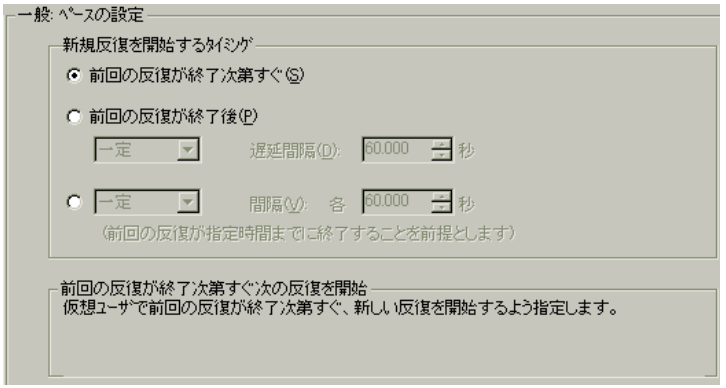
ペースの設定と実行論理オプションの設定（シングル・アクション）

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

仮想ユーザがスクリプトの実行時に「**Action**」セクションを繰り返し実行するように指定できます。この繰り返しを「**反復**」といいます。反復を複数回実行する場合，仮想ユーザ・スクリプトの **vuser_init** セクションと **vuser_end** セクションは繰り返されません。

反復とペースの設定を設定するには，次の手順を実行します。

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



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

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

- ［前回の反復が終了次第すぐ］
- ［前回の反復が終了後］
- ［一定］または［値の範囲］

4 ［**前回の反復が終了後**］オプションには次の設定があります。

- 遅延の種類を［**一定**］か［**値の設定**］から選択します。
- ［一定］で値を指定するか，最小 / 最高の乱数値を使用して値の範囲を指定します。

5 ［**各 … 秒**］オプションには次の設定があります。

- 間隔の種類を［**一定**］か［**値の範囲**］から選択します。
- ［一定］で値を指定するか，最小 / 最高の乱数値を使用して値の範囲を指定します。

6 ［**OK**］をクリックします。

ペースの設定のオプションの概要については，198 ページ「ペースの設定」を参照してください。

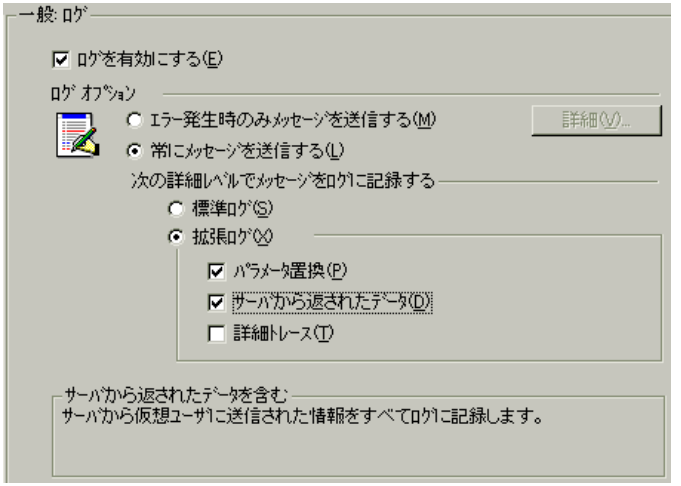
実行環境設定のログの設定

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

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

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

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



ログを有効にする

このオプションは，再生中の自動ログ記録を有効にします。つまり，VuGen によって実行ログにログ・メッセージが書き込まれます。このオプションは，自動ログ記録と **lr_log_message** を通じて発行されるログ・メッセージだけを対象とします。**lr_message**，**lr_output_message**，**lr_error_message** を使って手作業で送信されるメッセージは，変わらずに発行されます。

ログ・オプション

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

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

エラーの発生時にのみメッセージを送信する場合（JIT（ジャスト・イン・タイム）メッセージングとも呼ばれる）は，ログのキャッシュ・サイズなど，追加の詳細オプションを設定できます。詳細については，205 ページ「ログのキャッシュ・サイズの設定」を参照してください。

ログ詳細レベルの設定

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

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

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

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

- ［**拡張ログ**］：警告やメッセージを含めた詳細なログが作成されます。大規模な負荷テスト・シナリオ，チューニング・セッション，またはプロファイルでは，このオプションは無効にしてください。

  ［拡張ログ］オプションでは，拡張ログにどの情報を追加するかを指定できます。

  - ［**パラメータ置換**］：このオプションを選択すると，スクリプトに割り当てられたすべてのパラメータがそれらの値とともにログに記録されます。パラメータの詳細については，第 9 章「VuGen パラメータを使った作業」を参照してください。
  - ［**サーバから返されたデータ**］：このオプションを選択すると，サーバによって返されたすべてのデータがログに記録されます。
  - ［**詳細トレース**］：このオプションを選択すると，セッション中に仮想ユーザが送信したすべての関数とメッセージがログに記録されます。このオプションは，仮想ユーザ・スクリプトをデバッグするときに役立ちます。

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

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

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

ログのキャッシュ・サイズの設定

実行環境のログ設定の詳細オプションでは，ログのキャッシュ・サイズを指定できます。ログのキャッシュには，テスト実行に関する未処理のデータが格納され，エラーが生じた場合に参考にできます。キャッシュの内容が指定したサイズを超えると，一番古い項目が削除されます。標準設定のサイズは 1 KB です。

ログ記録は，次のように行われます。

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

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

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

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

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

CtLib サーバ・メッセージのログの記録

CtLib 仮想ユーザ・スクリプト（クライアント / サーバ・タイプのプロトコルの下にある Sybase CtLib）を実行する場合，CtLib クライアントによって生成されるメッセージはすべて，標準ログおよび出力ファイルのログに記録されます。標準設定では，サーバ・メッセージはログに記録されません。サーバ・メッセージのログの記録を（デバッグ用に）有効にするには，次の行を仮想ユーザ・スクリプトに挿入します。

```
LRD_CTLIB_DB_SERVER_MSG_LOG;
```

VuGen によって，すべてのサーバ・メッセージが標準ログに書き込まれます。

サーバ・メッセージを（標準ログ以外の）出力に送信するには，次のように入力します。

```
LRD_CTLIB_DB_SERVER_MSG_ERR;
```

サーバ・エラーを記録しない標準のモードに戻るには，次の行をスクリプトに入力します。

```
LRD_CTLIB_DB_SERVER_MSG_NONE;
```

注：生成されるサーバ・メッセージは長く，ログの書き込み処理によってシステムの処理速度が低下することがあるため，サーバ・メッセージのログの記録はスクリプト内の特定のコード・ブロックだけを対象に有効にします。

思考遅延時間の設定

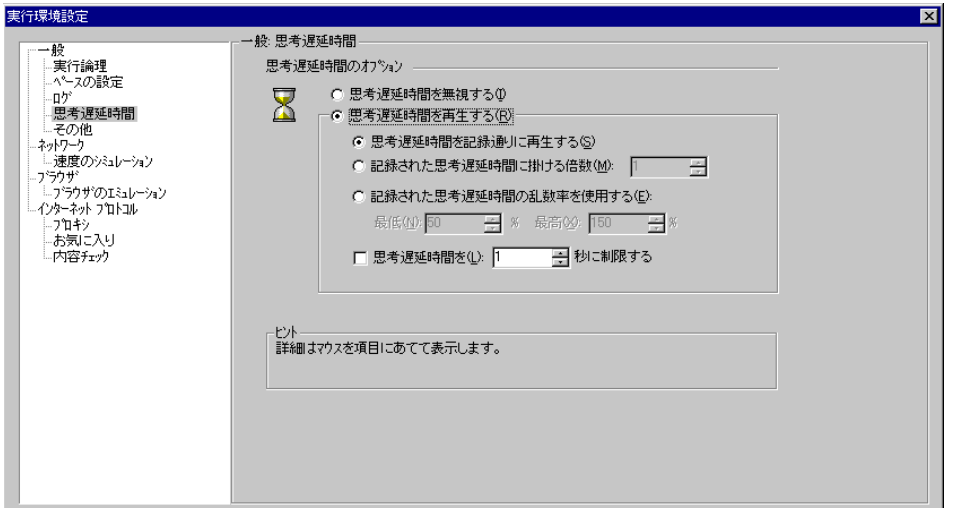
仮想ユーザの**思考遅延時間**は，実際のユーザがアクションとアクションの間で待つ時間をエミュレートしたものです。たとえば，ユーザはサーバからデータを受け取ったときに，データを数秒間かけて確認してから応答する可能性もあります。この遅れを「**思考遅延時間**」といいます。VuGen は **lr_think_time** 関数を使って，仮想ユーザ・スクリプトに思考遅延時間の値を記録します。次に示す記録された関数は，次のアクションを実行する前にユーザが 8 秒待機したことを示します。

```
lr_think_time(8);
```

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

lr_think_time 関数の詳細と手作業での変更方法については，「**オンライン関数リファレンス**」（［**ヘルプ**］＞［**関数リファレンス**］）を参照してください。

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



思考遅延時間のオプション

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

- ［**思考遅延時間を無視する**］: 記録された思考遅延時間を無視します。つまり，すべての **lr_think_time** 関数を無視してスクリプトを再生します。
- ［**思考遅延時間を再生する**］: 2 つ目の思考遅延時間オプションを選択すると，記録された思考遅延時間を使用できます。
  - ［**思考遅延時間を記録通りに再生する**］: 再生時に，lr_think_time 関数の引数の値が使用されます。たとえば，lr_think_time(10) は 10 秒間待機します。

- ［**記録された思考遅延時間に掛ける倍数**］：再生時に，記録された思考遅延時間の倍数を使用します。これにより再生中に適用される思考遅延時間を増減させることができます。たとえば，4 秒間の思考遅延時間が記録されている場合に，その値を 2 倍するように指定すれば，仮想ユーザの思考遅延時間は 8 秒になります。思考遅延時間を 2 秒に減らすには，記録された時間に 0.5 を掛けます。
- ［**記録された思考遅延時間の乱数率を使用する**］：記録された思考遅延時間の乱数率（%）を使用します。思考遅延時間の範囲を指定することによって，思考遅延時間の値の範囲を設定します。たとえば，思考遅延時間の引数が 4 の場合，下限を 50%，上限を 150% に設定すると，最短の思考遅延時間は 2（50%），最長の思考遅延時間は 6（150%）となります。
- ［**思考遅延時間を X 秒に制限する**］：思考遅延時間の最大値を制限します。

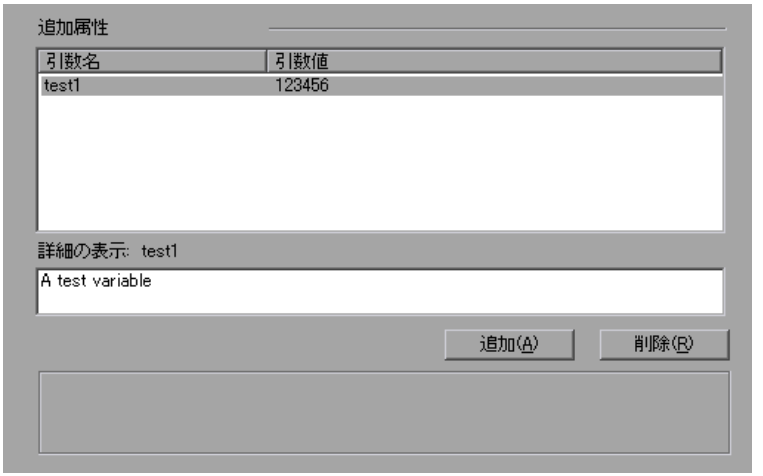
実行環境の追加属性の設定

Mercury Tuning Module では，［追加属性］ノードを使用して仮想ユーザ・スクリプトの追加属性を指定できます。［追加属性］の設定は，すべてのタイプの仮想ユーザ・スクリプトに適用されます。

lr_get_attrib_string を使用すると，テスト実行中のある時点で取得できるコマンド・ライン引数を指定できます。このノードを使用して，作成されたスクリプトにパラメータを渡すことにより，異なるクライアント・パラメータを使ってサーバのテストや監視を実行できます。

追加属性を設定するには，次の手順を実行します。

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



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

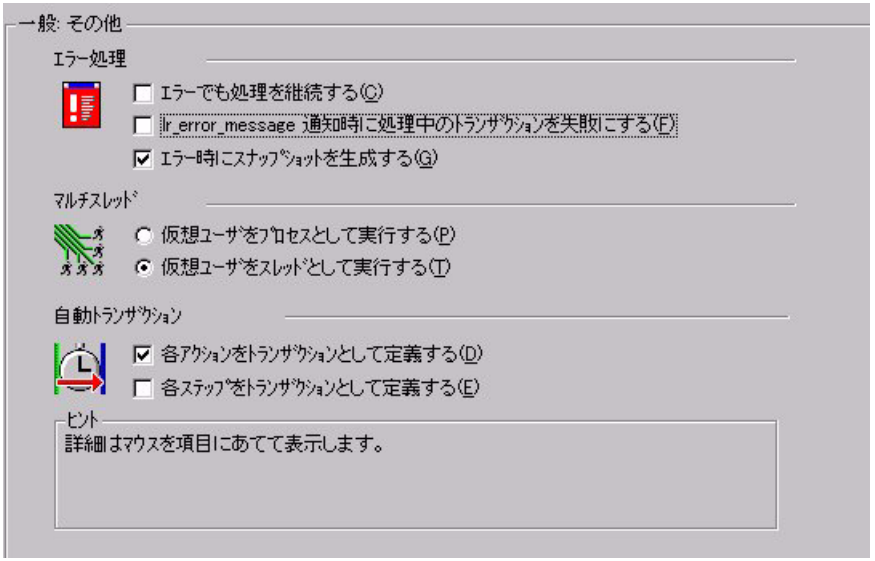
3 選択した引数を削除するには，［**削除**］をクリックします。

その他の設定

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

- エラー処理
- マルチ・スレッド
- 自動トランザクション

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



［**その他**］の設定は，すべてのタイプの仮想ユーザ・スクリプトに適用されます。

エラー処理

- ［**エラーでも処理を継続する**］：エラーが発生しても仮想ユーザがスクリプト実行を継続するように指定します。標準ではこのオプションは無効になっているため，エラーが発生すると仮想ユーザは終了します。
- ［**lr_error_message 通知時に処理中のトランザクションを失敗にする**］：**lr_error_message** 関数が発行されたすべてのトランザクションを「**失敗**」としてマークするように指定します。**If** ステートメントをプログラミングし，ある条件が満たされたときに **lr_error_message** 関数が発行されるようにします。
- ［**エラー時にスナップショットを生成する**］：エラー発生時にスナップショットを生成します。スナップショットは，仮想ユーザ・ログでエラーが発生した行をダブルクリックすると表示できます。

負荷テスト環境で［**エラーでも処理を継続する**］オプションと［**エラー時にスナップショットを生成する**］オプションを両方とも有効にすることはお勧めしません。このように設定すると，仮想ユーザのパフォーマンスに悪影響を与える可能性があります。

データベース仮想ユーザのエラー処理

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

たとえば，［エラーでも処理を継続する］機能を有効にしていて，仮想ユーザが次に示すスクリプトのセグメントの再生中にエラーに遭遇した場合には，スクリプトの実行は継続されます。

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

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

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

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

VuGen では以下の重要度のレベルがサポートされています。

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

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

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

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

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

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

RTE 仮想ユーザのエラー処理

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

たとえば，[エラーでも処理を継続する] 機能を有効にしていて，仮想ユーザが次に示すスクリプトのセグメントの再生中にエラーに遭遇した場合には，スクリプトの実行は継続されます。

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

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

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

マルチ・スレッド

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

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

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

コントローラと Tuning Console は，ドライバ・プログラム（たとえば **mdrv.exe**，**r3vuser.exe** など）を使って仮想ユーザを実行します。各仮想ユーザを個別のプロセスとして実行した場合，同じドライバ・プログラムが仮想ユーザのインスタンスごとにメモリ上でいくつも実行（およびロード）されます。同じドライバ・プログラムをメモリにロードすると，多くの RAM とその他のシステム・リソースが消費されます。そのため，ロード・ジェネレータで実行できる仮想ユーザの数が限定されることもあります。

別の方法として，各仮想ユーザをスレッドとして実行すると，コントローラや Tuning Console は 50 の仮想ユーザに対してドライバ・プログラム（**mdrv.exe** など）のインスタンスを 1 つだけ起動します（標準設定）。このドライバ・プロセス / プログラムは，いくつかの仮想ユーザを起動し，各仮想ユーザはスレッドとして実行されます。これらのスレッド化された仮想ユーザは，親ドライバ・プロセスのメモリのセグメントを共有します。ドライバ・プロセス / プログラムを何度も再ロードする必要がなくなるので，メモリ空間を大幅に節約でき，1 台のロード・ジェネレータで実行できる仮想ユーザの数を増やせます。

自動トランザクション

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

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

自動トランザクションを無効にしている場合でも，記録中および記録後に手作業でトランザクションを挿入できます。手作業によるトランザクションの挿入の詳細については，第 8 章「仮想ユーザ・スクリプトの拡張」を参照してください。

注：シナリオの実行中に診断ブレークダウン・データ（J2EE）を生成する必要がある場合は，自動トランザクションを使用してはなりません。各トランザクションの開始と終了を手作業で定義してください。

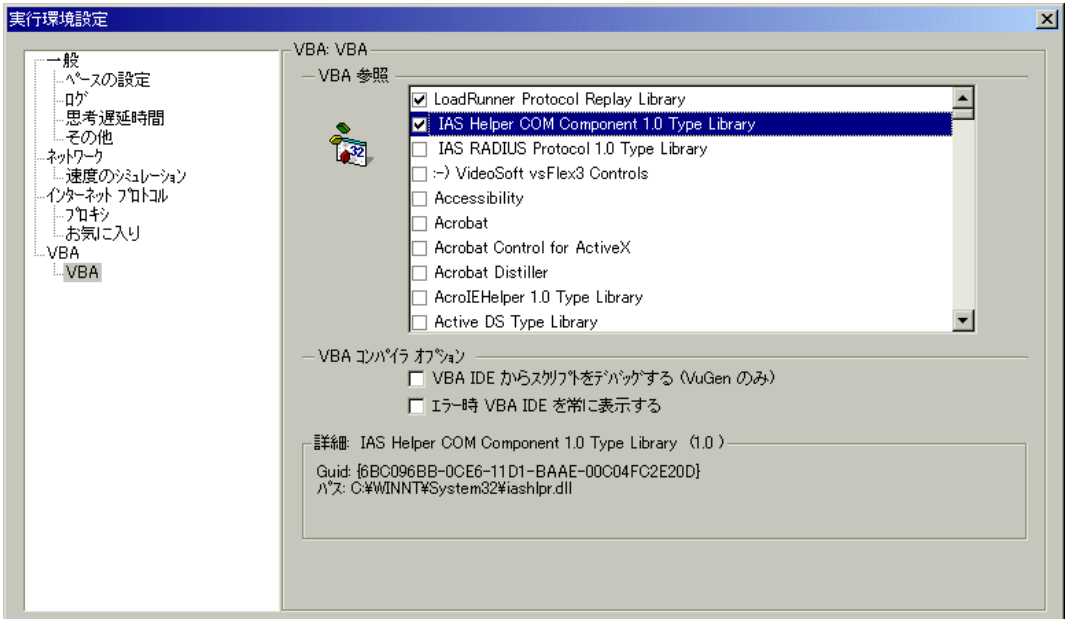
VB 実行環境の設定

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

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

VBA の実行環境を設定するには，次の手順を実行します。

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



2 ［**VBA 参照**］セクションで，スクリプト実行中に使用する参照ライブラリを選択します。ライブラリを選択すると，そのライブラリの詳細とバージョンがダイアログ・ボックスの下部に表示されます。

3 コンパイラに関する適切なオプションを選択します。

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

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

4 ［**OK**］をクリックして実行環境の設定を適用します。

第 14 章

ネットワーク実行環境の設定

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

本章では，次の項目について説明します。

- ネットワーク実行環境の設定について
- ネットワーク速度の設定

以降の情報は，すべてのインターネット関連プロトコル，Citrix ICA，Oracle NCA，WinSock を対象とします。

すべての仮想ユーザに適用される一般的な実行環境の設定については，第 13 章「実行環境の設定」を参照してください。

ネットワーク実行環境の設定について

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

実行環境設定を行うには，［実行環境設定］ダイアログ・ボックスを開き，適切なノードを選択します。

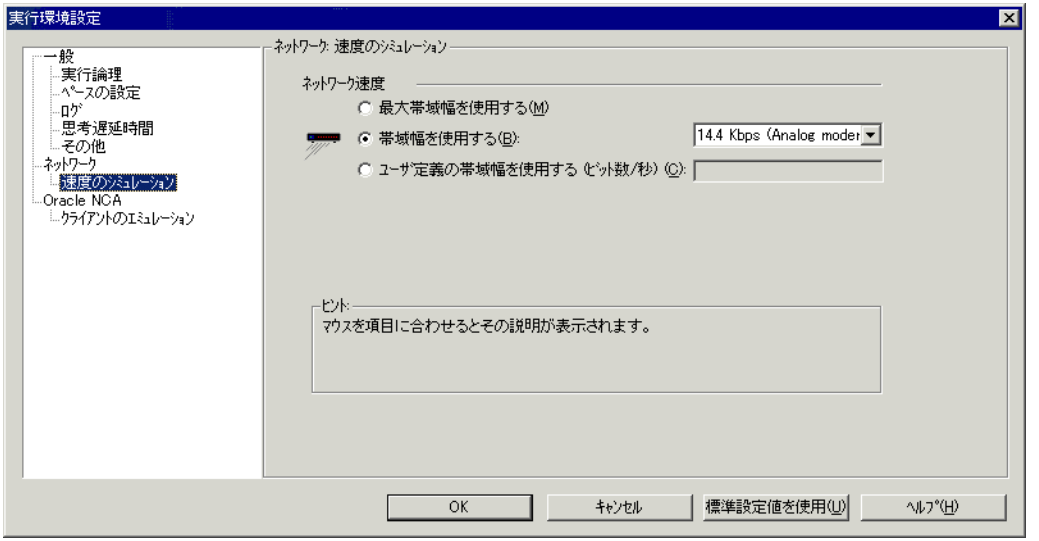
［実行環境設定］ダイアログ・ボックスを表示するには，次のいずれかの手順を実行します。

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

実行環境の設定は，LoadRunner コントローラ または 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。

ネットワーク速度の設定

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



速度のシミュレーション

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

- ［**最大帯域幅を使用する**］：標準では，帯域幅のエミュレーションは無効になっており，仮想ユーザはネットワーク上で利用できる最大の帯域幅で実行されます。
- ［**帯域幅を使用する**］仮想ユーザをエミュレートする帯域幅のレベルを指定します。アナログ・モデム，ISDN，DSL をエミュレートするため，14.4Kbps から 512 Kbps の範囲の速度を選択できます。
- ［**ユーザ定義の帯域幅を使用する**］：仮想ユーザをエミュレートする帯域幅の上限を指定します。帯域幅をビットで指定します（1 キロビット =1024 ビット）。

第 15 章

スタンドアロン・モードでの仮想ユーザ・スクリプトの実行

仮想ユーザ・スクリプトを作成し，その実行環境を設定したら，スクリプトをスタンドアロン・モードで実行してテストします。

本章では，次の項目について説明します。

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

以降の情報は，全タイプの仮想ユーザ・スクリプトを対象とします。

スタンドアロン・モードでの仮想ユーザ・スクリプトの実行について

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

スタンドアロンで正常に実行できたら，スクリプトを自分の環境（LoadRunner シナリオ，Performance Center 負荷テスト，Tuning Module セッション，またはビジネス・プロセス・モニタ・プロファイル）に統合します。詳細については，『**Mercury LoadRunner コントローラ・ユーザーズ・ガイド**』，**Tuning Console** のマニュアル，**Performance Center** のマニュアル，または **Business Availability Center** のマニュアルを参照してください。

スクリプトをスタンドアロン・モードで実行する前に，次の作業を実施できます。

- 仮想ユーザ関数を使ったスクリプトの拡張（第 8 章「仮想ユーザ・スクリプトの拡張」参照）
- スクリプトのパラメータ化（第 9 章「VuGen パラメータを使った作業」参照）
- スクリプトのクエリの相関（第 12 章「ステートメントの相関」参照）

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

VuGen での仮想ユーザ・スクリプトの実行

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

注：VuGen で仮想ユーザ・スクリプトを実行できるのは Windows プラットフォームだけです。UNIX ベースの仮想ユーザ・スクリプトを実行するには，236 ページ「UNIX コマンド・ラインからの仮想ユーザ・スクリプトの実行」を参照してください。

再生オプションの設定

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

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

標準設定値を使用

［**標準設定を使用**］をクリックすると，元の値にリセットされます。

［**表示実行遅延**］は 0 ミリ秒になります。

［**Action のセクションのみで関数を表示実行する**］は有効になります。

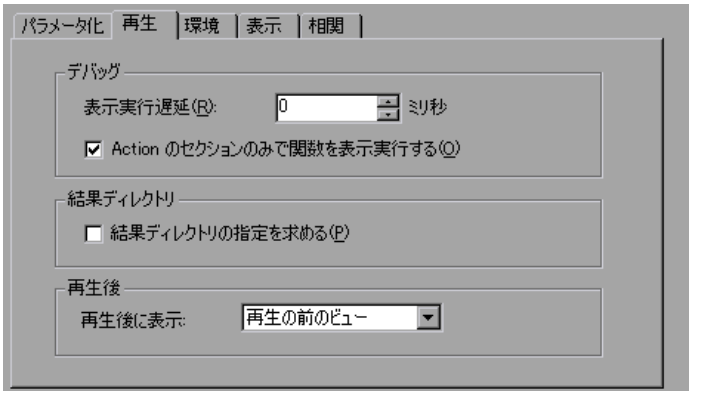
［**結果ディレクトリの指定を求める**］は無効になります。

［**再生後に表示**］は［再生の前のビュー］に設定されます。

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

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

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

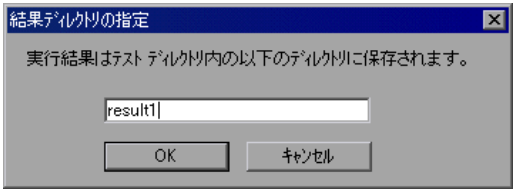


3 ［**再生**］タブを選択します。

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

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

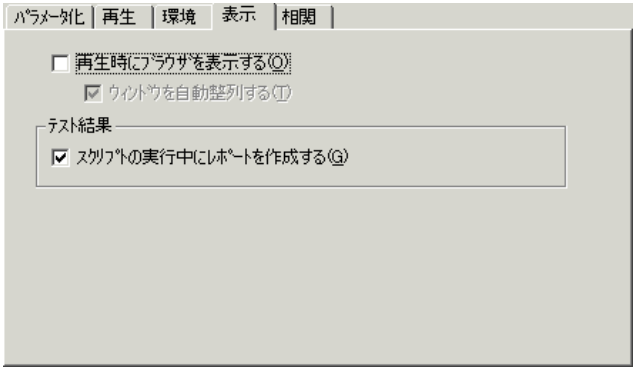
6 VuGen からスクリプトを実行する前に結果ディレクトリを指定できるようにするには，［**結果ディレクトリの指定を求める**］を選択します。実行コマンドをクリックすると，［**結果ディレクトリの指定**］ダイアログ・ボックスが開きます。



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

表示オプションの設定

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



- ［**再生時にブラウザを表示する**］：実行時ビューアを有効にします。スクリプトの実行終了時に実行時ビューアを最小化するには，［**ウィンドウを自動整列する**］オプションを選択します。
- ［**スクリプトの実行中にレポートを作成する**］：結果サマリ・レポートを生成するには，次の手順を実行します。スクリプトの実行後にレポートを開くには，［**表示**］＞［**テスト結果**］を選択します。

標準設定では，［**再生時にブラウザを表示する**］オプションは無効で，［**スクリプトの実行中にレポートを作成する**］オプションは有効になっています。これらの値を元に戻すには，［**標準設定を使用**］をクリックします。

これらのオプションを使用したデバッグ方法の詳細については，234 ページ「Web 仮想ユーザ・スクリプトでの VuGen のデバッグ機能の使用」を参照してください。

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

2 実行時ビューアを有効にするには，［**再生時にブラウザを表示する**］を選択します。スクリプトの実行終了時に実行時ビューアを最小化するには，［**ウィンドウを自動整列する**］を選択します。

3 仮想ユーザに結果サマリ・レポートを生成させるには，［**スクリプトの実行中にレポートを作成する**］を選択します。スクリプトの実行後にレポートを開くには，［**表示**］＞［**テスト結果**］を選択します。

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

仮想ユーザ・スクリプトの再生

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

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

VuGen でスクリプトを再生するには，次の手順を実行します。

1 ［**仮想ユーザ**］＞［**実行**］を選択します。

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

2 仮想ユーザのすべてのアクションのログのほか，警告やエラーを見るには，［出力］ウィンドウの［**再生ログ**］タブをクリックします。詳細については，227 ページ「再生ログの表示」を参照してください。

3 実行時データおよびパラメータのサマリをそれらの使用中に表示するには，［出力］ウィンドウの［**実行時データ**］タブをクリックします。詳細については，228 ページ「[実行時データ］タブ」を参照してください。

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

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

再生ログの表示

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

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

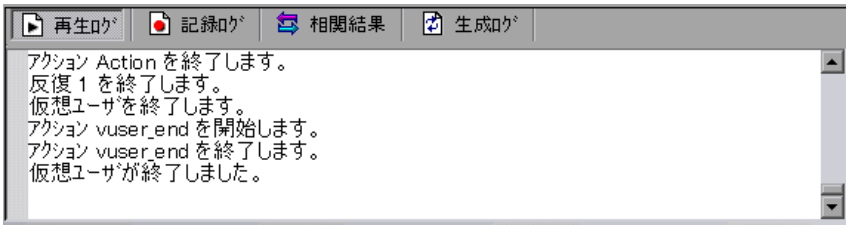
［再生ログ］では，テキストがさまざまに色分けされています。

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

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

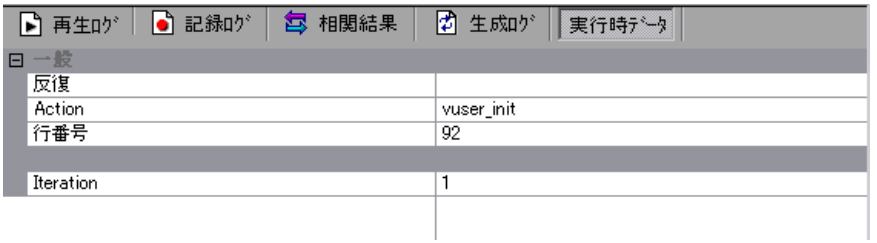
［出力］ウィンドウの表示と非表示の詳細については，226 ページ「仮想ユーザ・スクリプトの再生」を参照してください。

次の例は，Web 仮想ユーザ・スクリプトの実行によって生成された再生ログのメッセージを示します。



[実行時データ] タブ

再生中に最新になったスクリプト情報を，[実行時データ] タブを使用して追跡することができます。



再生中に，右端の［**実行時データ**］タブをクリックします。このタブには展開または折りたたみが可能な 2 つのセクションがあります。

- ［**一般**］：［一般］セクションには，現在の反復数，現在再生されているステップのアクション名，スクリプト内での行番号（スクリプト・ビュー）が表示されます。
- ［**パラメータ**］：［パラメータ］セクションには，スクリプトに定義されているすべてのパラメータと，選択した更新方法（順次，一意など）に基づくそれらのパラメータの置換値が表示されます。これらの情報はパラメータがスクリプトの中で使用されていない場合にも表示されます。詳細については，第 9 章「VuGen パラメータを使った作業」を参照してください。

テストの実行後には［実行時データ］タブにはアクセスできません。このタブには，テストの再生中に変化するデータのみが表示されるからです。

VuGen のデバッグ機能の使用

VuGen には，仮想ユーザ・スクリプトのデバッグに役立つ，ステップ実行コマンドとブレークポイントの 2 つのオプションがあります。これらのオプションは，VBScript Vuser および VB Vuser タイプの仮想ユーザでは使用できません。

また VuGen には，Web 仮想ユーザ・スクリプトのデバッグに役立つ追加機能があります。詳細については，234 ページ「Web 仮想ユーザ・スクリプトでの VuGen のデバッグ機能の使用」を参照してください。

デバッグ・ツールバーを表示するには，次の手順を実行します。

ツールバー領域を右クリックして，［**デバッグ**］を選択します。デバッグ・ツールバーがツールバー領域に表示されます。

ステップ実行コマンド

ステップ実行コマンドは，スクリプトを 1 回に 1 行ずつ実行します。これによりスクリプトの実行を追うことができます。

ステップ実行コマンドでスクリプトを実行するには，次の手順を実行します。

1 ［**仮想ユーザ**］＞［**ステップごとに実行**］を選択するか，デバッグ・ツールバーの［**ステップ**］ボタンをクリックします。

VuGen によってスクリプトの最初の行が実行されます。

2 スクリプトの実行が完了するまで［**ステップ**］ボタンをクリックして，スクリプトの実行を続けます。

ブレークポイント

ブレークポイントは，スクリプトの特定の場所で実行を一時停止します。これにより，スクリプトの実行中に，あらかじめ定義しておいたポイントで，スクリプトによるアプリケーションへの影響を調査できます。ブックマークを管理するには，231 ページ「ブレークポイント・マネージャ」の手順を実行します。

ブレークポイントを設定するには，次の手順を実行します。

1 スクリプト内で実行を停止する行にカーソルを移動します。

2 ［**挿入**］＞［**ブレークポイントの設定 / 解除**］を選択するか，デバッグ・ツールバーの［**ブレークポイント**］ボタンをクリックします。または，キーボードの F9 キーを押します。ブレークポイントの記号（●）がスクリプトの左余白に表示されます。

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

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

ブレークポイントを設定したスクリプトを実行するには，次の手順を実行します。

1 通常どおり，スクリプトの実行を開始します。

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

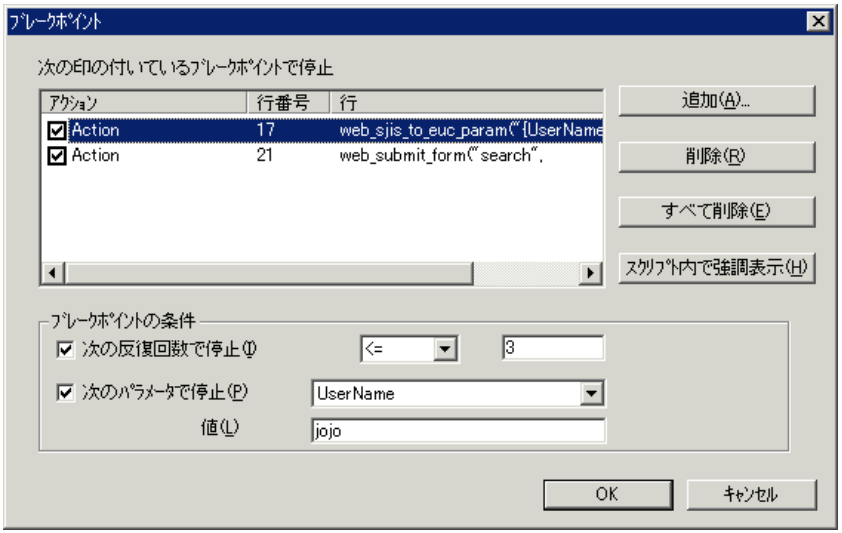
2 実行を再開するには，［**仮想ユーザ**］＞［**実行**］を選択します。

再開すると，次のブレークポイントに到達するか，スクリプトが終了するまで，スクリプトの実行が続けられます。

ブレークポイント・マネージャ

ブレークポイント・マネージャを使用してブレークポイントを表示および管理することができます。ブレークポイント・マネージャから，スクリプト内のすべてのブレークポイントを操作できます。

ブレークポイント・マネージャを開くには，［**編集**］＞［**ブレークポイント**］を選択します。



スクリプト内のブレークポイント位置へ移動するには，次の手順を実行します。

1 リストからブレークポイントを選択します。

2 ［**スクリプト内で強調表示**］をクリックします。スクリプト内の行が強調表示されます。

一度に強調表示できるブレークポイントは 1 つだけです。

ブレークポイントの管理

ブレークポイント・マネージャから，ブレークポイントの追加，削除，無効化，または条件の設定を実行できます。

ブレークポイントを追加するには，次の手順を実行します。

1 ［**追加**］をクリックします。［ブレークポイントの追加］ダイアログ・ボックスが開きます。

2 ［**アクション**］を選択し，ブレークポイントを追加する［**行番号**］を指定します。

3 ［**OK**］をクリックします。ブレークポイントがブレークポイントのリストに追加されます。

ブレークポイントを削除するには，次の手順を実行します。

1 ブレークポイントを 1 つ削除するには，ブレークポイントを選択して［**削除**］をクリックします。

2 ブレークポイントをすべて削除するには，［**すべて削除**］をクリックします。

ブレークポイントを有効化または無効化するには，次の手順を実行します。

1 ブレークポイントを有効にするには，［アクション］カラムで，アクションのチェック・ボックスを選択します。

2 ブレークポイントを無効にするには，［アクション］カラムで，アクションのチェック・ボックスをクリアします。

ブレークポイント・マネージャから，特定の条件のもとで実行を一時停止するためのブレークポイントを設定できます。

ブレークポイントの条件を設定するには，次の手順を実行します。

1 特定の反復回数後にスクリプトを一時停止するには，［**次の反復回数で停止**］を選択して，必要な回数を入力します。

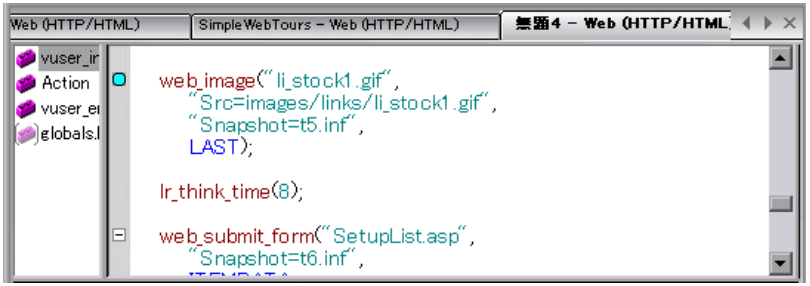
2 パラメータ **X** が特定の値になったときにスクリプトを一時停止するには，［**次のパラメータで停止**］を選択して，必要な値を入力します。パラメータの詳細については，第 9 章「VuGen パラメータを使った作業」を参照してください。

ブックマーク

スクリプト・ビューで作業をしているときに，スクリプト内のさまざまな位置にブックマークを置くことができます。ブックマーク間を移動することで，コードの分析やデバッグを実行できます。

ブックマークを作成するには，次の手順を実行します。

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



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

3 ブックマーク間を移動するには，次の手順を実行します。

次のブックマークに移動するには，F2 キーを押します。

前のブックマークに移動するには，Shift + F2 キーを押します。

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

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

［移動］コマンド

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

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

Web 仮想ユーザ・スクリプトでの VuGen のデバッグ機能の使用

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

- Web 仮想ユーザ・スクリプトの実行時に実行時ビューアを表示するように VuGen を設定することができます。実行時ビューアは，VuGen 向けに Mercury によって開発されました。仮想ユーザ・スクリプトの記録に使用するブラウザとは無関係です。実行時ビューアには，仮想ユーザによってアクセスされる各 Web ページが表示されます。これにより仮想ユーザが正しい Web ページにアクセスしているかどうかを確認できるので，Web 仮想ユーザ・スクリプトのデバッグ時に役立ちます。詳細については，第 60 章「Web 仮想ユーザのヒント(パワー・ユーザ向け)」を参照してください。

- Web 仮想ユーザに対して，スクリプトの実行時に結果サマリ・レポートを生成させるかどうかを指定できます。結果サマリ・レポートとは，Web 仮想ユーザ・スクリプト内の各ステップの成功や失敗についてまとめたもので，各ステップで返された Web ページも確認できます。結果サマリ・レポートを使った作業の詳細については，［**表示**］＞［**テスト結果**］を選択し，F1 キーを押して，オンライン・ヘルプを参照してください。または，第 16 章「テスト結果の表示」を参照してください。

  上記［表示］オプションの設定の詳細については，225 ページ「表示オプションの設定」を参照してください。

注：仮想ユーザに結果サマリ・レポートを生成させると，トランザクションの処理時間が増える場合があります。仮想ユーザから結果サマリ・レポートを生成できるのは，VuGen から実行した場合だけです。スクリプトをコントローラ，Tuning Module，またはビジネス・プロセス・モニタから実行した場合は，仮想ユーザからレポートは生成されません。

VuGen ウィンドウを使った作業

次の機能を実行することで，VuGen のウィンドウを表示して配置を変更し，スクリプトの関連データを表示することができます。

- ［**出力**］ウィンドウの表示 / 非表示：スクリプト・エディタの下の出力ウィンドウの表示 / 非表示を切り替えるには，［**表示**］＞［**出力ウィンドウ**］を選択します。出力ウィンドウには，プロトコルに応じていくつかのタブが表示されます。最も一般的なタブは，［再生ログ］，［記録ログ］，［生成ログ］，および［相関結果］です。詳細については，227 ページ「再生ログの表示」を参照してください。
- **すべてのサムネイルの表示**：スクリプトのすべてのステップをサムネイルとして表示するには，［**表示**］＞［**すべてのサムネイルを表示**］を選択します。主要なステップのみについてサムネイルを表示するには，このオプションをクリアします。詳細については，25 ページ「スクリプトのサムネイルの表示」を参照してください。
- **グリッドの表示**：データ・グリッドをサポートしているプロトコル（データベース，COM，および Microsoft . NET）について，データのグリッド表示を有効にするには，［**表示**］＞［**データ グリッド**］を選択します。グリッドがスクリプトの中に表示されます。
- **ウィンドウの操作**：開いているスクリプトをすべて閉じるには，［**ウィンドウ**］＞［**すべて閉じる**］を選択します。必要に応じて，保存していないスクリプトを保存するよう VuGen から求められます。

コマンド・プロンプトからの仮想ユーザ・スクリプトの実行

VuGen のユーザ・インタフェースを使わずに，コマンド・プロンプトや Windows の［ファイル名を指定して実行］ダイアログ・ボックスから仮想ユーザ・スクリプトをテストできます。

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

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

2 次のとおりに入力して，**Enter** キーを押します。

＜ VuGen のパス＞ /bin/mdrv.exe -usr **＜スクリプト名＞** -vugen_win 0

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

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

UNIX コマンド・ラインからの仮想ユーザ・スクリプトの実行

VuGen を使って UNIX ベースの仮想ユーザを作成するときには，記録されたスクリプトを UNIX プラットフォームで実行できることを確認する必要があります。スクリプトを正しく実行できることを次の手順で確認します。

1 **記録されたスクリプトを VuGen からテストします。**

記録されたスクリプトを VuGen から実行して，Windows ベースのシステムで正しく実行できることを確認します。

2 **仮想ユーザ・スクリプトのファイルを UNIX ドライブにコピーします。**

ファイルをローカルの UNIX ドライブに転送します。

3 **verify_generator を使用して，UNIX マシン上での仮想ユーザの設定を確かめます。**

詳細については，237 ページ「verify_generator によるテスト」を参照してください。

4 **UNIX コマンド・ラインからスクリプトをテストします。**

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

```
run_db_vuser.sh <スクリプト名> .usr
```

verify_generator によるテスト

この検証ユーティリティは，ローカル・ホストで通信パラメータを調べ，すべての仮想ユーザ・タイプとの互換性を確認します。

仮想ユーザ環境で次の項目を調べます。

- 最低 128 のファイル記述子を持っていること
- .rhost のパーミッションが -rw-r--r-- という正しいものになっていること
- rsh を使用してホストに接続できること。接続できない場合は，.rhosts 内のホスト名を調べる
- M_LROOT が定義されていること
- .cshrc で正しい M_LROOT が定義されていること
- .cshrc がホーム・ディレクトリに存在すること
- 現在のユーザが .cshrc の所有者であること
- LoadRunner が $M_LROOT にインストールされていること
- 実行可能ファイルが実行パーミッションを持っていること
- PATH に $M_LROOT/bin および /usr/bin が含まれていること
- rstatd デーモンが存在し，実行されていること

1 つのホストですべての仮想ユーザを実行する場合は，次を入力します。

```
verify_generator
```

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

この検証ユーティリティの詳細については，次を入力してください。

```
verify_generator [-v]
```

コマンド・ライン・オプション : run_db_vuser シェル・スクリプト

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

--help：使用可能なオプションを表示します（このオプションの先頭には，ダッシュを 2 つ付ける必要があります）。

-cpp_only：スクリプトを対象に cpp のみ（プリプロセス）を実行します。

-cci_only：スクリプトを対象に cci のみ（プリコンパイル）を実行して，拡張子が .ci のファイルを作成します。cpp が成功しないと，cci は実行できません。

-driver ＜ドライバのパス＞：指定されたドライバ・プログラムを使用します。各データベースに固有のドライバ・プログラムが，/bin ディレクトリにあります。たとえば，/bin ディレクトリにある CtLib のドライバは **mdrv** です。このオプションを使って，外部ドライバを指定することができます。

-exec_only：仮想ユーザの .ci ファイルを実行します。このオプションは，有効な .ci ファイルが存在するときにのみ使用できます。

-ci ＜ ci ファイル名＞：指定された .ci ファイルを実行します。

-out ＜出力パス＞：指定されたディレクトリに結果を格納します。

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

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

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

スクリプトのテストへの統合

スタンドアロン・モードでのスクリプトの実行が成功し，スクリプトが正常に機能することを確認したら，そのスクリプトを自環境（LoadRunner シナリオ，Performance Center 負荷テスト，Tuning Module セッション，またはビジネス・プロセス・モニタ・プロファイル）に統合します。

テストを統合するときは，次の情報を指定します。

- 統合するスクリプト
- スクリプトを実行する仮想ユーザ
- スクリプトが実行されるロード・ジェネレータ
- スケジュール設定

詳細については，『**Mercury LoadRunner コントローラ・ユーザーズ・ガイド**』，**Tuning Console** のマニュアル，**Performance Center** のマニュアル，または **Business Availability Center** のマニュアルを参照してください。

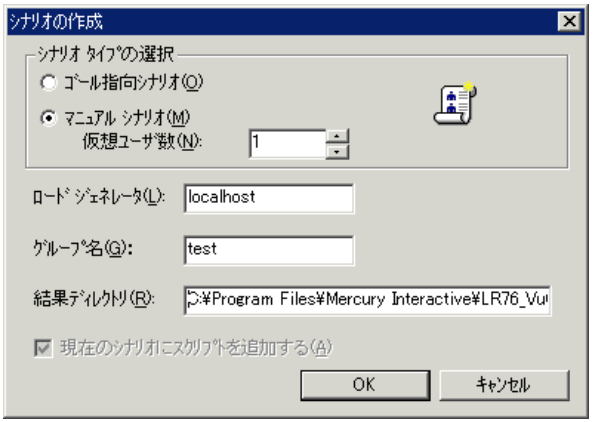
VuGen を使用しての LoadRunner シナリオの作成

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

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

VuGen でシナリオを作成するには，次の手順を実行します。

1 ［**ツール**］＞［**コントローラのシナリオを作成**］を選択します。［シナリオの作成］ダイアログ・ボックスが開きます。



2 ゴール指向シナリオか，マニュアル・シナリオかを選択します。

ゴール指向シナリオを選択した場合，指定した目的に応じたシナリオが LoadRunner によって自動的に作成されます。それに対しマニュアル・シナリオでは実行する仮想ユーザ数を指定します。

3 マニュアル・シナリオの場合は，スクリプトを実行する仮想ユーザの数を入力します。

4 仮想ユーザの実行に使用するマシンの名前を［**ロード ジェネレータ**］ボックスに入力します。

5 マニュアル・シナリオの場合，共通の特性を持つユーザをグループに編成します。それらの仮想ユーザのグループ名を［**グループ名**］ボックスで指定します。

6 ゴール指向シナリオの場合，［**スクリプト名**］を指定します。

7 結果を格納する場所を，［**結果ディレクトリ**］ボックスに入力します。

8 コントローラですでにシナリオを開いていて，このシナリオにスクリプトを追加する場合は，［**現在のシナリオにスクリプトを追加する**］チェック・ボックスを選択します。チェック・ボックスをクリアすると，LoadRunner によって，指定した数の仮想ユーザを含んだ新しいシナリオが作成されます。

9 ［**OK**］をクリックします。VuGen によって，仮想ユーザのビューにコントローラが表示されます。

10 共有ネットワーク・ドライブにスクリプトを保存するようにコントローラを設定した場合は，パス変換を行わなければならないことがあります。

詳細については，『**Mercury LoadRunner コントローラ・ユーザーズ・ガイド**』を参照してください。

第 16 章

テスト結果の表示

仮想ユーザ・スクリプトの実行結果をまとめたレポートは，スクリプトのデバッグ作業に利用できます。VuGen は仮想ユーザ・スクリプトの実行中にレポートを生成し，スクリプトの実行が完了したらレポートを表示します。本章では，次の項目について説明します。

- テスト結果の表示について
- 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 以降を使用することをお勧めします。

以降の情報は，Web および Web サービス仮想ユーザ・スクリプトのみを対象とします。

テスト結果の表示について

VuGen の仮想ユーザ・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 15 章「スタンドアロン・モードでの仮想ユーザ・スクリプトの実行」を参照してください。

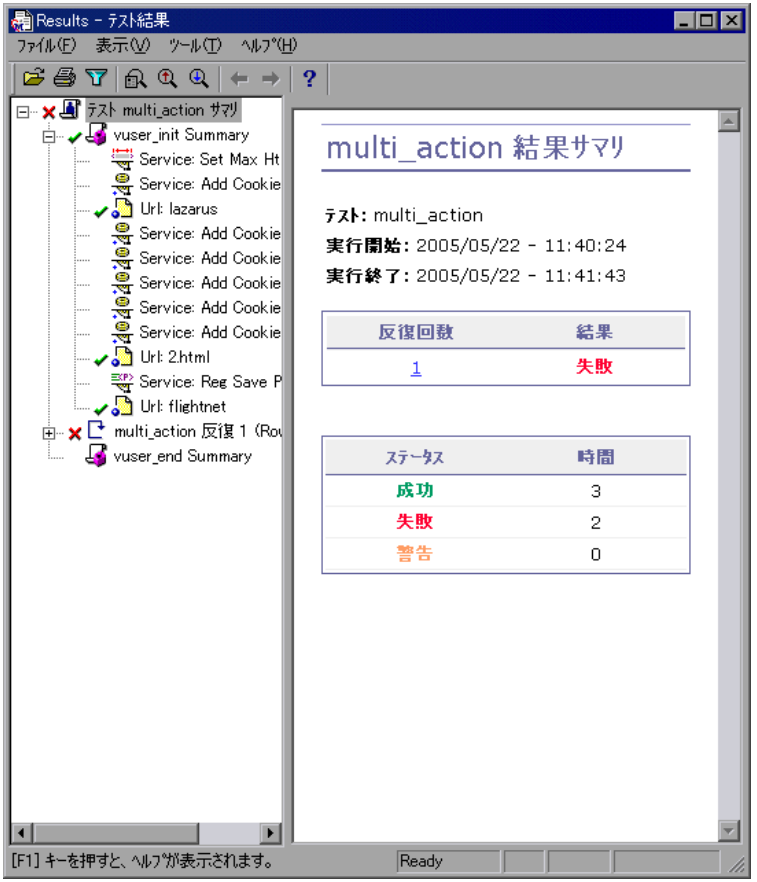
VuGen を使って Web 仮想ユーザ・スクリプトを実行した後に，結果サマリ・レポートが表示されます。

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 15 章「スタンドアロン・モードでの仮想ユーザ・スクリプトの実行」を参照してください。

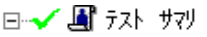
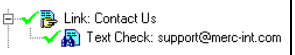
結果サマリ・レポートについて

仮想ユーザ・スクリプトの実行後，結果サマリ・レポートが表示されます。レポートには，スクリプトの実行結果のサマリが表示されます。



- 左側の表示枠には，結果をグラフィカルに示す「レポート・ツリー」が表示されます。レポート・ツリーでは，成功したステップを緑色のチェック印で，失敗したステップが赤色の「×」印で示されます。
- 右側の表示枠には「レポートの詳細」が表示されます。ここには，スクリプト実行全般についてのサマリや，レポート・ツリーで選択された分岐についての補助的な情報が表示されます。

レポート・ツリーの分岐を 1 つ選択して，その項目の情報を表示します。

ツリーの分岐	表示内容
テスト名 	スクリプトの実行全般についての結果サマリ。
テストの反復 	1 つの反復についての実行サマリ。
テストのステップまたはチェック 	仮想ユーザ・スクリプト内で選択されたステップやチェックに対応する Web ページ。

レポート・ツリー内の分岐を展開したり，折りたたんだりすることで，ツリーに表示される情報の詳しさを変更することができます。

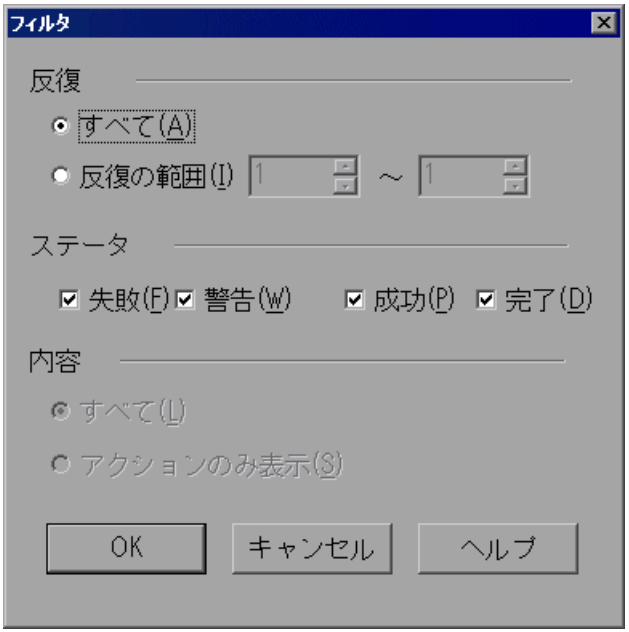
- 分岐を折りたたむには，折りたたみたい分岐の左側にあるマイナス（−）記号をクリックします。レポート・ツリーは分岐を隠し，マイナス（−）記号はプラス記号（+）に変わります。
- レポート・ツリー内のすべての分岐を折りたたむには，［**表示**］＞［**すべて折りたたみ**］を選択します。
- 分岐を展開して表示するには，表示したい分岐の左側にあるプラス（+）記号をクリックします。レポート・ツリーは分岐を展開して表示し，プラス（+）記号はマイナス（−）記号に変わります。
- レポート・ツリー内のすべての分岐を表示するには，［**表示**］＞［**すべて展開**］を選択します。

レポート情報のフィルタリング

VuGen 結果サマリ・レポートに表示される情報にフィルタを適用することができます。フィルタによる選別は，反復回数か，反復のステータスに基づきます。

レポートに含まれる情報にフィルタを適用するには，次の手順を実行します。

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 フィルタを適用する反復を指定します。標準設定のフィルタ・オプションは［**すべて**］です。レポートの出力対象を指定の範囲の反復回数に制限するには，［**反復の範囲**］を選択して範囲を指定します。

3 レポートに対して，［**失敗**］，［**警告**］，［**成功**］，および［**完了**］のうち 1 つ以上の **Status** フィルタを指定します。［テスト結果］ウィンドウには，選択した項目のみが表示されます。

4 ［OK］をクリックして設定を確定し，結果にフィルタを適用します。

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］で［**上へ**］または［**下へ**］を選択します。

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，234 ページ「Web 仮想ユーザ・スクリプトでの VuGen のデバッグ機能の使用」または第 15 章「スタンドアロン・モードでの仮想ユーザ・スクリプトの実行」を参照してください。

結果サマリ・レポートを開く

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。［ファイルを開く］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。

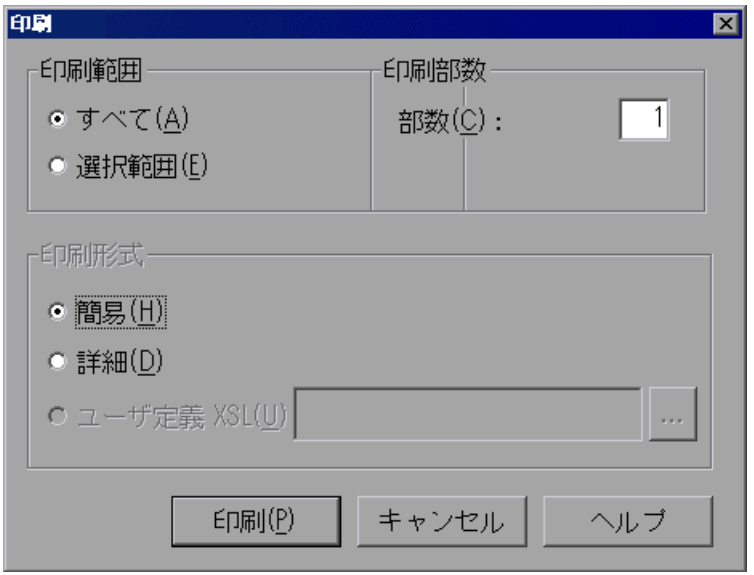
3 最近表示したレポートを開くには，［**ファイル**］メニューのレポート履歴リストからそのレポートを選択します。

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・ボックスが表示されます。



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 印刷する部数を指定します。

4 ［**印刷形式**］を指定します。

➤ ［**簡易**］: テスト結果の簡単なサマリを印刷します。

➤ ［**詳細**］: レポート全体を印刷します。

5 ［**印刷**］をクリックします。

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 17 章

Quality Center を使ったスクリプト管理

VuGen と Quality Center の統合により，Quality Center を使って仮想ユーザ・スクリプトを管理できます。

本章では，次の項目について説明します。

- Quality Center を使ったスクリプト管理について
- Quality Center の接続と切断
- Quality Center プロジェクトからスクリプトを開く
- Quality Center プロジェクトへのスクリプトの保存
- VuGen でのスクリプトのバージョン管理

Quality Center を使ったスクリプト管理について

VuGen は，Mercury の Web ベースのテスト管理ツール，Quality Center と連携して機能します。Quality Center は，仮想ユーザ・スクリプト，シナリオ，またはセッション・ステップの格納と検索，および結果の収集のための能率的な手段を提供します。スクリプトを Quality Center プロジェクトに格納し，固有のグループに編成します。

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

Quality Center を使った作業の詳細については，『**Quality Center ユーザーズ・ガイド**』を参照してください。

Quality Center の接続と切断

VuGen と Quality Center の両方を使って作業している場合には，VuGen から Quality Center プロジェクトとやり取りできます。テスト・プロセスにおいて，VuGen と Quality Center プロジェクトはいつでも接続または切断できます。

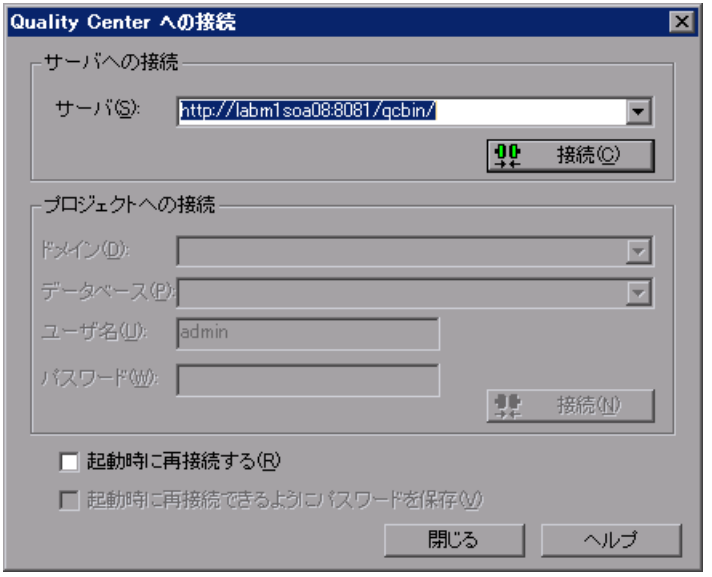
Quality Center への VuGen の接続

接続プロセスには 2 つの段階があります。最初に，VuGen をローカル Quality Center Web サーバまたはリモート Quality Center Web サーバに接続します。このサーバによって，VuGen と Quality Center プロジェクトの間の接続が処理されます。

次に，VuGen でアクセスするプロジェクトを選択します。プロジェクトには，テスト対象または監視対象アプリケーションのためのスクリプトが格納されています。Quality Center プロジェクトはパスワードで保護されているため，ユーザ名とパスワードを指定する必要があります。

VuGen を Quality Center に接続するには，次の手順を実行します。

1 VuGen で，［**ツール**］＞［**Quality Center への接続**］を選択します。［Quality Center への接続］ダイアログ・ボックスが表示されます。



2 ［サーバ］ボックスに，Quality Center がインストールされている Web サーバの URL アドレスを入力します。

注：ローカル・エリア・ネットワーク（LAN）または広域エリア・ネットワーク（WAN）を介してアクセス可能な Web サーバを選択できます。

---

3 ［**接続**］をクリックします。サーバへの接続が確立されると，［サーバ］ボックスにサーバの名前が読み取り専用で表示されます。

4 ［プロジェクトへの接続］セクションの［ドメイン］ボックスで，ドメインを選択します。

5 ［プロジェクトへの接続］セクションの［プロジェクト］ボックスで，Quality Center プロジェクトを選択します。

6 ユーザ名を［**ユーザ名**］ボックスに入力します。

7 パスワードを［**パスワード**］ボックスに入力します。

8 ［**接続**］ボタンをクリックして，選択したプロジェクトに VuGen を接続します。

選択したプロジェクトへの接続が確立されると，［データベース］ボックスにプロジェクトの名前が読み取り専用で表示されます。

9 起動時に Quality Center サーバと選択したプロジェクトに自動的に再接続するには，［**起動時に再接続する**］チェック・ボックスを選択します。

10 ［**起動時に再接続する**］を選択すると，指定したパスワードを保存して起動時に再接続できます。［**起動時に再接続できるようにパスワードを保存**］チェック・ボックスを選択します。

パスワードを保存しなければ，起動時に VuGen を Quality Center に接続するときに，パスワードの入力が必要になります。

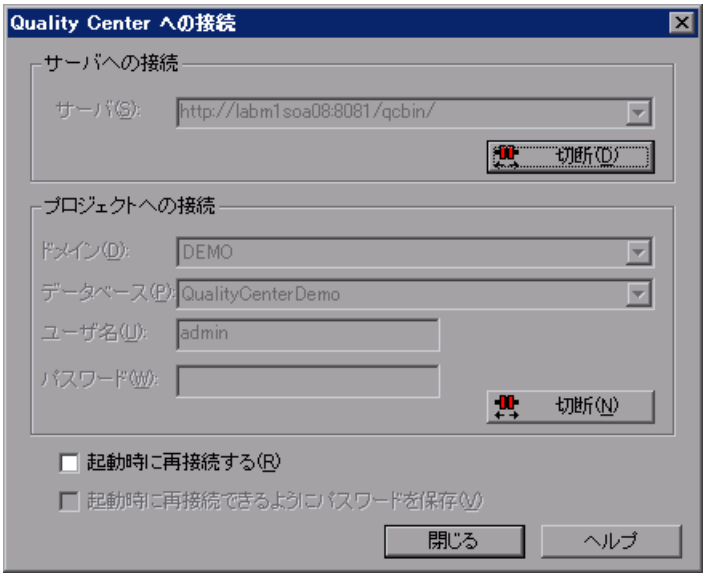
11 ［**閉じる**］をクリックして，［Quality Center への接続］ダイアログ・ボックスを閉じます。

Quality Center からの VuGen の切断

選択した Quality Center プロジェクトと Web サーバから VuGen を切断できます。

Quality Center から VuGen を切断するには，次の手順を実行します。

1 VuGen で，［**ツール**］＞［**Quality Center への接続**］を選択します。［Quality Center への接続］ダイアログ・ボックスが表示されます。



2 選択したプロジェクトから VuGen を切断するには，［プロジェクトへの接続］セクションの［**切断**］ボタンをクリックします。

3 選択したサーバから VuGen を切断するには，［サーバへの接続］セクションの［**切断**］ボタンをクリックします。

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

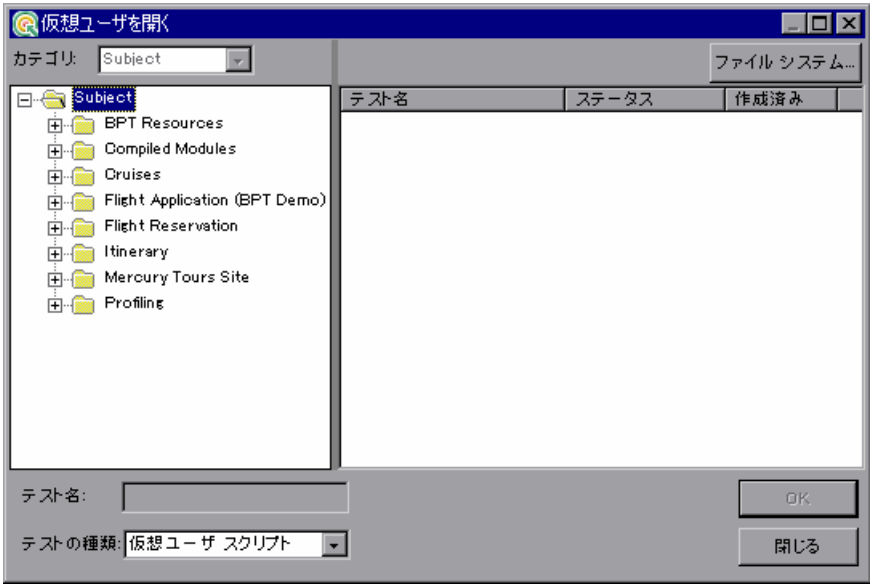
Quality Center プロジェクトからスクリプトを開く

VuGen を Quality Center プロジェクトに接続する場合，Quality Center からシナリオを開くことができます。テストは，ファイル・システムの実際の位置からではなく，テスト計画ツリーでの位置から見つけます。

Quality Center プロジェクトからスクリプトを開くには，次の手順を実行します。

1 Quality Center サーバに接続します（252 ページ「Quality Center への VuGen の接続」を参照してください）。

2 VuGen で，［**ファイル**］＞［**開く**］を選択するか，［**開く**］ボタンをクリックします。［仮想ユーザを開く］ダイアログ・ボックスが表示され，テスト計画ツリーが表示されます。



ファイル・システムから直接スクリプトを開くには，［**ファイル システム**］ボタンをクリックします。［仮想ユーザを開く］ダイアログ・ボックスが開きます（［仮想ユーザを開く］ダイアログ・ボックスからは，［**Quality Center**］ボタンをクリックすれば［仮想ユーザを開く］ダイアログ・ボックスに戻ることができます）。

3 テスト計画ツリー内の適切なサブジェクトをクリックします。ツリーを展開して下位レベルを表示するには，閉じているフォルダをダブルクリックします。ツリーを折りたたむには，開いているフォルダをダブルクリックします。

サブジェクトを選択すると，そのサブジェクトに属しているスクリプトが［テスト名］カラムに表示されます。

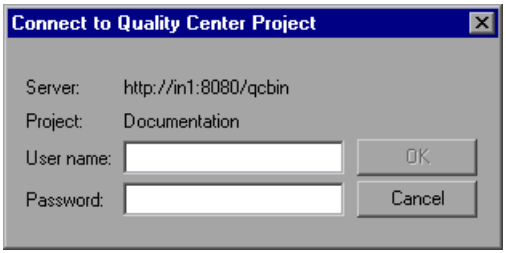
4 ［テスト名］カラムからスクリプトを選択します。スクリプト名が読み取り専用の［テスト名］ボックスに表示されます。

5 ［**OK**］をクリックしてスクリプトを開きます。VuGen によってスクリプトがロードされます。スクリプトの名前が VuGen のタイトルバーに表示されます。［**設計**］タブに，テスト計画ツリーのすべてのスクリプトが表示されます。

注：［ファイル］メニューの最新ファイル・リストからスクリプトを開くこともできます。Quality Center プロジェクト内のスクリプトを選択したときに，VuGen がそのプロジェクトに接続されていない場合は，［Quality Center への接続］ダイアログ・ボックスが表示されます。ユーザ名とパスワードを入力してプロジェクトにログインし，［**OK**］をクリックします。

最近使用したテストのリストからテストを開く

［ファイル］メニューの最新ファイル・リストから仮想ユーザ・スクリプトを開くこともできます。Quality Center プロジェクトのスクリプトを選択したときに，現在 VuGen が Quality Center に接続されていないか，スクリプトの正しいプロジェクトに接続されていないと，［Quality Center プロジェクトへの接続］ダイアログ・ボックスが表示され，正しいサーバ，プロジェクト，および前回当該コンピュータでスクリプトを開いたユーザの名前が表示されます。



プロジェクトにログインし，［**OK**］をクリックします。

また，前回異なる Quality Center ユーザ名を使用して編集したテストを開くことを選択した場合も，［Quality Center プロジェクトへの接続］ダイアログ・ボックスが開きます。表示されているユーザ名を使ってログインすることも，［**キャンセル**］をクリックして現在のユーザ名でログインすることもできます。

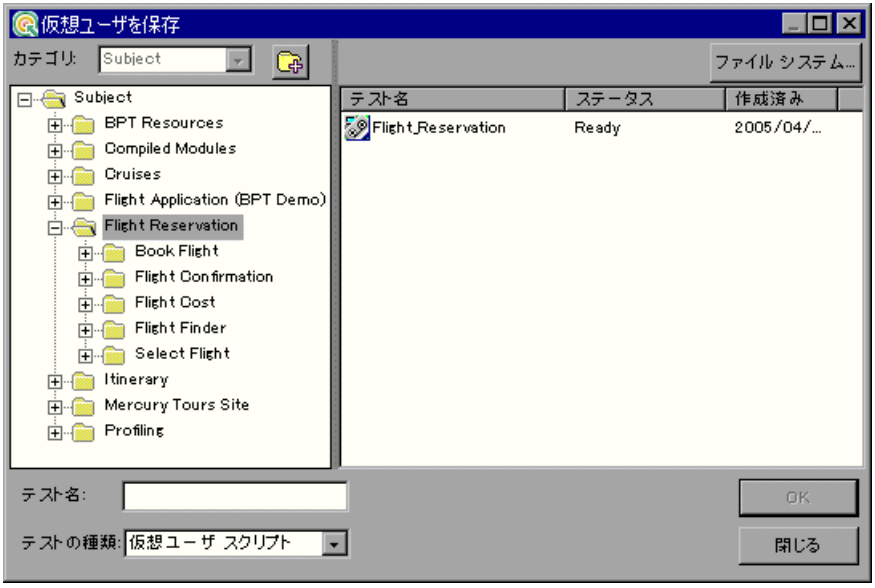
Quality Center プロジェクトへのスクリプトの保存

VuGen が Quality Center プロジェクトに接続されているときには，VuGen で新しいスクリプトを作成して，プロジェクトに直接保存することができます。スクリプトを保存するには，わかりやすい名前を付けて，テスト計画ツリーの中の適切なサブジェクトに関連付けます。これにより，各サブジェクトに対して作成されたスクリプトを把握し，テスト計画とテスト作成の進捗を表示できるようになります。

スクリプトを Quality Center プロジェクトに保存するには，次の手順を実行します。

1 Quality Center サーバに接続します（252 ページ「Quality Center への VuGen の接続」を参照してください）。

2 VuGen で，［**ファイル**］＞［**名前を付けて保存**］を選択します。［仮想ユーザを保存］ダイアログ・ボックスが表示され，テスト計画ツリーが表示されます。



ファイル・システムに直接スクリプトを保存するには，［**ファイル システム**］ボタンをクリックします。［仮想ユーザを保存］ダイアログ・ボックスが表示されます（［仮想ユーザを保存］ダイアログ・ボックスからは，［**Quality Center**］ボタンをクリックすれば Quality Center 用の［仮想ユーザを保存］ダイアログ・ボックスに戻ることができます)。

3 テスト計画ツリー内の適切なサブジェクトを選択します。ツリーを展開してサブレベルを表示するには，閉じているフォルダをダブルクリックします。ツリーを折りたたむには，開いているフォルダをダブルクリックします。

4 ［テスト名］ボックスに，スクリプトの名前を入力します。スクリプトを簡単に識別できるような分かりやすい名前を使用します。

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

次に Quality Center を起動したときに，Quality Center のテスト計画ツリーに新しいスクリプトが表示されます。

VuGen でのスクリプトのバージョン管理

VuGen をバージョン・コントロール・サポート付きの Quality Center プロジェクトに接続すると，古いバージョンを維持しながら，自動化されたスクリプトの更新や変更ができます。これにより，各スクリプトに加えた変更の追跡，スクリプトのバージョン間の変更の確認，スクリプトの旧バージョンの復元が容易になります。

バージョン・コントロール・サポート付きのプロジェクトにスクリプトを保存すると，バージョン・コントロール・データベースにスクリプトが追加されます。バージョン・コントロール・データベースにスクリプトをチェック・イン，チェック・アウトすることで，スクリプトのバージョンが管理されます。

最新バージョンのスクリプトは，Quality Center のリポジトリにあり，Quality Center によってすべてのテストの実行に使用されます。

注 :［**ファイル**］メニューの［**Quality Center バージョン管理**］オプションは，バージョン・コントロール・サポートの付いた Quality Center プロジェクトのデータベースに接続している場合にだけ使用できます。

バージョン・コントロール・データベースへのスクリプトの追加

［**名前を付けて保存**］を使ってバージョン・コントロール・サポート付きの Quality Center プロジェクトに新規のスクリプトを保存すると，VuGen によって自動的にスクリプトがプロジェクトに保存され，バージョン・コントロール・データベースにバージョン番号 1.1.1 でチェック・インされ，その後作業を続けられるようにチェック・アウトされます。

ステータス・バーにはそのつど各操作が表示されます。ただし，既存のスクリプトに変更を保存しても，そのスクリプトをチェック・インすることにはなりません。スクリプトを保存して閉じても，チェック・インを選択するまではスクリプトはチェック・アウトしたままです。詳細については，259 ページ「バージョン・コントロール・データベースからのスクリプトのチェック・アウト」を参照してください。

バージョン・コントロール・データベースからのスクリプトのチェック・アウト

［**ファイル**］＞［**開く**］を選択して，現在バージョン・コントロール・データベースにチェック・インしているスクリプトを開くと，スクリプトは読み取り専用モードで開きます。

注：Quality Center の［仮想ユーザを開く］ダイアログ・ボックスには，プロジェクトの各スクリプトのバージョン・コントロール・ステータスを示すアイコンが表示されます。詳細については，255 ページ「Quality Center プロジェクトからスクリプトを開く」を参照してください。

チェック・インしたスクリプトを確認できます。また，スクリプトを実行して結果を表示することもできます。

スクリプトを変更するには，スクリプトをチェック・アウトする必要があります。スクリプトをチェック・アウトすると，スクリプトは Quality Center によってユーザの一意のチェック・アウト・ディレクトリ（初めてスクリプトをチェック・アウトするときに自動的に作成される）にコピーされ，プロジェクト・データベースにロックされます。これにより，スクリプトに加えた変更を Quality Center プロジェクトのほかのユーザに上書きされないようにします。ただし，この場合でも，ほかのユーザはデータベースにチェック・インされているスクリプトの最新バージョンを実行できます。

スクリプトを保存して閉じることはできますが，Quality Center データベースに戻されるまでスクリプトはロックされています。スクリプトは，スクリプトをチェック・インするか，チェック・アウトの操作を取り消すことによって解放されます。スクリプトのチェック・インの詳細については，261 ページ「バージョン・コントロール・データベースへのスクリプトのチェック・イン」を参照してください。チェック・アウトの取り消しの詳細については，267 ページ「チェック・アウト操作の取り消し」を参照してください。

標準では，チェック・アウト・オプションはスクリプトの最新バージョンにアクセスします。古いバージョンのスクリプトもチェック・アウトできます。詳細については，263 ページ「[バージョンの履歴] ダイアログ・ボックスの使用方法」を参照してください。

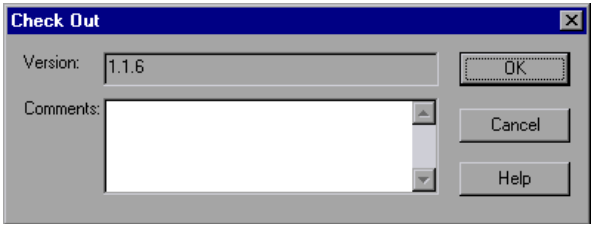
最新バージョンのスクリプトをチェック・アウトするには，次の手順を実行します。

1 チェック・アウトするスクリプトを開きます。詳細については，255 ページ「Quality Center プロジェクトからスクリプトを開く」を参照してください。

注：開いたスクリプトが現在チェック・インされていることを確認します。チェック・アウトしているスクリプトを開くと，[**チェック アウト**] オプションは無効になります。ほかのユーザがチェック・アウトしているスクリプトを開くと，[**バージョンの履歴**] オプションを除くすべての [**Quality Center バージョン管理**] オプションは無効になります。

---

2 [**ファイル**] > [**Quality Center バージョン管理**] > [**チェック アウト**] を選択します。[チェック アウト] ダイアログ・ボックスが開き，チェック・アウトされるスクリプトのバージョンが表示されます。



3 [**コメント**] ボックスに変更の詳細を入力します。

4 [**OK**] をクリックします。読み取り専用のスクリプトが閉じて，書き込み可能スクリプトとして再び自動的に開きます。

バージョン・コントロール・データベースへのスクリプトのチェック・イン

あるスクリプトがチェック・アウトされていても，Quality Center ユーザは以前にチェック・インされたバージョンを実行できます。たとえば，スクリプトのバージョン 1.2.3 をチェック・アウトし，いくつか変更を加えてスクリプトを保存するとします。バージョン 1.2.4（または別の番号を割り当てる）としてスクリプトをバージョン・コントロール・データベースに再びチェック・インするまで，Quality Center ユーザはバージョン 1.2.3 の実行を続行できます。

スクリプトの変更が終了し，Quality Center ユーザが新しいバージョンを使用できるようになったら，バージョン・コントロール・データベースにスクリプトをチェック・インします。

注：Quality Center データベースにチェック・インしない場合は，チェック・アウト操作を取り消すことができます。詳細については，267 ページ「チェック・アウト操作の取り消し」を参照してください。

スクリプトをバージョン・コントロール・データベースに再びチェック・インすると，Quality Center はチェック・アウト・ディレクトリからスクリプトのコピーを削除し，Quality Center プロジェクトのほかのユーザがそのスクリプトのバージョンを利用できるようにデータベースのロックを解除します。

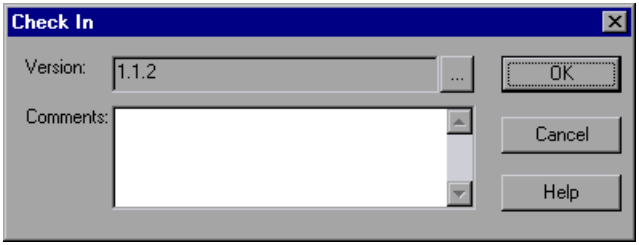
現在開いているスクリプトをチェック・インするには，次の手順を実行します。

1 現在開いているスクリプトをチェック・アウトしていることを確認します。詳細については，263 ページ「スクリプトのバージョン情報の表示」を参照してください。

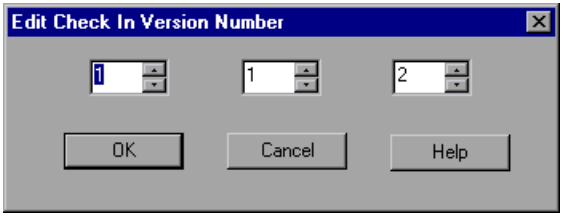
注：開いているスクリプトが現在チェック・インされていると，［**チェック イン**］オプションは無効になります。ほかのユーザがチェック・アウトしているスクリプトを開くと，［**バージョンの履歴**］オプションを除くすべての［**Quality Center バージョン管理**］オプションは無効になります。

---

2 ［**ファイル**］＞［**Quality Center バージョン管理**］＞［**チェック イン**］を選択します。［チェック イン］ダイアログ・ボックスが表示されます。



3 標準設定の新しいバージョン番号を受け入れて手順 7 に進むか，参照ボタンをクリックしてユーザ定義のバージョン番号を指定します。参照ボタンをクリックすると，［チェック イン］ダイアログ・ボックスが開きます。



4 バージョン番号を，手作業またはバージョン番号の各要素の横にある上向き矢印と下向き矢印を使って修正します。最初の要素に入力できる数字は 1 ～ 900 です。第 2，第 3 の要素に入力できる数字は 1 ～ 999 です。バージョン・コントロール・データベースにある対象のスクリプトの最新バージョン番号よりも低いバージョン番号は入力できません。

5 [**OK**] をクリックしてバージョン番号を保存し，[チェック イン] ダイアログ・ボックスを閉じます。

6 スクリプトのチェック・アウト時に変更の詳細を入力した場合，その詳細は [**コメント**] ボックスに表示されます。コメントの入力や修正はこのボックスで行えます。

7 [**OK**] をクリックし，スクリプトをチェック・インします。スクリプトが閉じて，読み取り専用のスクリプトとして再び自動的に開きます。

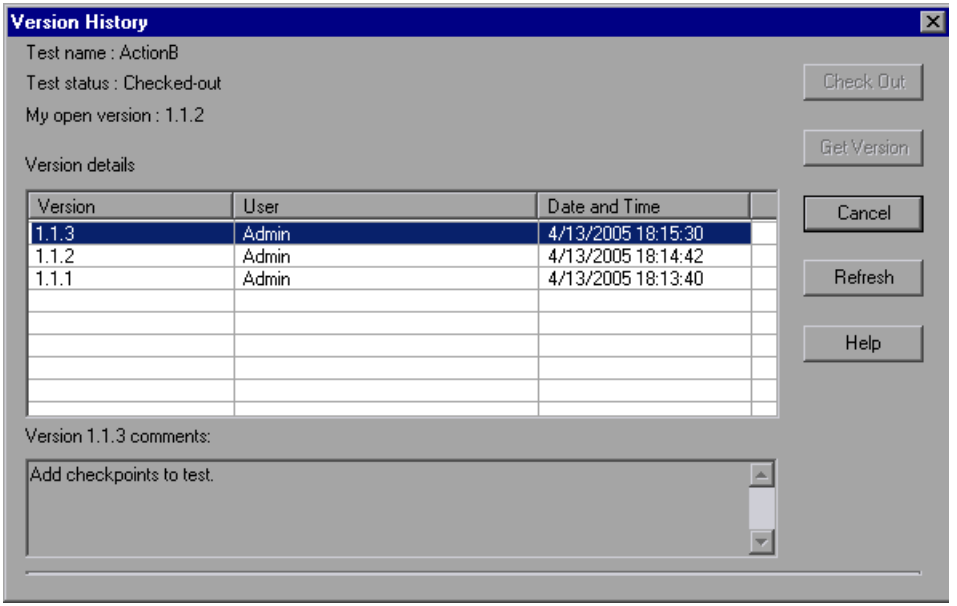
[バージョンの履歴] ダイアログ・ボックスの使用方法

[バージョンの履歴] ダイアログ・ボックスを使って，現在開いているスクリプトに関するバージョン情報の表示や，古いバージョンのスクリプトの表示および取得が行えます。

スクリプトのバージョン情報の表示

Quality Center バージョン・コントロール・データベースに格納されている任意の開いているスクリプトのバージョン情報を，現在のステータスに関係なく表示できます。

スクリプトの［バージョンの履歴］ダイアログ・ボックスを開くには，スクリプトを開き［**ファイル**］＞［**Quality Center バージョン管理**］＞［**バージョンの履歴**］を選択します。



［バージョンの履歴］ダイアログ・ボックスには次の情報が表示されます。

- ［**Test name**］：現在開いているスクリプトの名前。
- ［**Test status**］：スクリプトのステータス。次のいずれかになります。
  - ［**Checked-in**］：スクリプトは，現在バージョン・コントロール・データベースにチェック・インしています。現在読み取り専用形式で開いています。スクリプトをチェック・アウトして編集できます。
  - ［**Checked-out**］：スクリプトをチェック・アウトしています。現在読み取り / 書き込み可能な形式で開いています。
  - ［**Checked-out by ＜ユーザ名＞**］：スクリプトは現在別のユーザによってチェックアウトされています。現在読み取り専用形式で開いています。特定のユーザがスクリプトをチェック・インするまでこのスクリプトをチェック・アウトしたり編集したりすることはできません。

- ［**My open version**］：現在，お使いの QuickTest コンピュータで開いているスクリプトのバージョン。
- ［**Version details**］：スクリプトのバージョンの詳細。
  - ［**バージョン**］：スクリプトのバージョンの一覧。
  - ［**ユーザ**］：各バージョンのチェック・インをしたユーザです。
  - ［**日付**］：各バージョンがチェック・インされた日です。
  - ［**時刻**］：各バージョンがチェック・インされた時刻です。
- ［**Version comments**］：選択したスクリプトのバージョンがチェック・インされたときに入力されたコメント。

以前のスクリプトのバージョンを使った作業

古いバージョンのスクリプトは，読み取り専用モードで表示するか，チェック・アウトして最新バージョンとしてチェック・インできます。

古いバージョンのスクリプトを表示するには，次の手順を実行します。

1 Quality Center スクリプトを開きます。最新バージョンのスクリプトが開きます。詳細については，255 ページ「Quality Center プロジェクトからスクリプトを開く」を参照してください。

2 ［**ファイル**］＞［**Quality Center バージョン管理**］＞［**バージョンの履歴**］を選択します。［バージョンの履歴］ダイアログ・ボックスが表示されます。

3 **バージョン詳細リスト**から表示するバージョンを選択します。

4 ［**バージョンの取得**］ボタンをクリックします。QuickTest から，スクリプトがチェック・アウトされていないため読み取り専用モードで開くというメッセージが表示されます。

5 ［**OK**］をクリックし，QuickTest メッセージを閉じます。選択したバージョンが読み取り専用モードで開きます。

ヒント：QuickTest で現在開いているバージョン番号を確認するには，［バージョンの履歴］ダイアログ・ボックスの［**My open version**］の値を参照します。

［**バージョンの取得**］オプションを使用して読み取り専用モードで古いバージョンを開いたら，［**ファイル**］＞［**Quality Center バージョン管理**］＞［**チェック アウト**］を選択して，開いているスクリプトをチェック・アウトできます。これは，［バージョンの履歴］ダイアログ・ボックスで［**チェック アウト**］ボタンを使用するのと同等の機能です。

古いバージョンのスクリプトをチェック・アウトするには，次の手順を実行します。

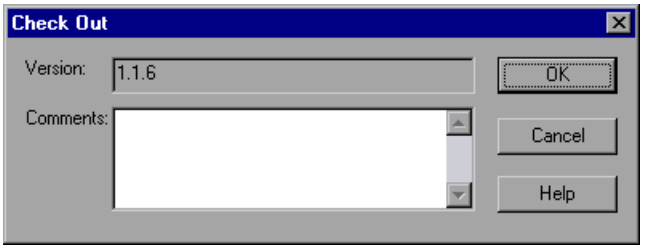
1 Quality Center スクリプトを開きます。最新バージョンのスクリプトが開きます。詳細については，255 ページ「Quality Center プロジェクトからスクリプトを開く」を参照してください。

2 ［**ファイル**］＞［**Quality Center バージョン管理**］＞［**バージョンの履歴**］を選択します。［バージョンの履歴］ダイアログ・ボックスが表示されます。

3 **バージョン詳細リスト**から表示するバージョンを選択します。

4 ［**チェック アウト**］ボタンをクリックします。確認メッセージが開きます。

5 古いバージョンのスクリプトをチェック・アウトすることを確認します。［チェック アウト］ダイアログ・ボックスが開き，チェック・アウトされるバージョンが表示されます。



6 ［**コメント**］ボックスに変更の詳細を入力します。

7 ［**OK**］をクリックします。開いていたスクリプトが閉じ，選択したバージョンが書き込み可能スクリプトとして開きます。

8 必要に応じてスクリプトを表示または編集します。

9 新規の最新バージョンとしてスクリプトを Quality Center データベースにチェック・インするには，［**ファイル**］＞［**Quality Center バージョン管理**］［**チェック イン**］を選択します。変更したスクリプトを Quality Center にアップロードしない場合は，［**ファイル**］＞［**Quality Center バージョン管理**］＞［**チェック アウトを元に戻す**］を選択します。

スクリプトのチェック・インの詳細については，261 ページ「バージョン・コントロール・データベースへのスクリプトのチェック・イン」を参照してください。チェック・アウトの取り消しの詳細については，267 ページ「チェック・アウト操作の取り消し」を参照してください。

チェック・アウト操作の取り消し

スクリプトをチェック・アウトした後に，変更したスクリプトを Quality Center にアップロードしないと決定した場合は，ほかの Quality Center ユーザがチェック・アウトできるように，チェック・アウト操作を取り消す必要があります。

チェック・アウト操作を取り消すには，次の手順を実行します。

1 チェック・アウトしたスクリプトをまだ開いていない場合は開きます。

2 ［**ファイル**］＞［**Quality Center バージョン管理**］＞［**チェック アウトを元に戻す**］を選択します。

3 ［**はい**］をクリックし，チェック・アウト操作の取り消しを確定します。
チェック・アウト操作が取り消されます。チェック・アウトしたスクリプトが閉じ，以前チェック・インしたバージョンが読み取り専用モードで再び開きます。

第 18 章

Performance Center によるスクリプトの管理

VuGen を Performance Center と統合することにより，Performance Center サーバへのスクリプトのアップロードおよび Performance Center サーバからのスクリプトのダウンロードが行えます。

本章では，次の項目について説明します。

- Performance Center によるスクリプトの管理について
- VuGen から Performance Center への接続
- 仮想ユーザ・スクリプトのアップロード
- 仮想ユーザ・スクリプトのダウンロード

Performance Center によるスクリプトの管理について

VuGen は，Mercury の Web 対応のグローバル負荷テスト・ツールである Performance Center と統合でき，異なる場所からシステムをテストできるようになっています。

VuGen のユーザ・インタフェースを使って，Performance Center サーバとの間でスクリプトのアップロードとダウンロードを実行できます。スクリプトを Performance Center にアップロードすることにより，スクリプトを仮想ユーザ・スクリプトのリストに追加して，テストで使用できるようになります。また，スクリプトをダウンロードして，編集したりローカルに保存したりすることもできます。

VuGen から Performance Center にアクセスしてアップロードやダウンロードを行うには，Performance Center がインストールされているサーバに接続する必要があります。

Performance Center の使用方法の詳細については，『**Performance Center ユーザーズ・ガイド**』を参照してください。

VuGen から Performance Center への接続

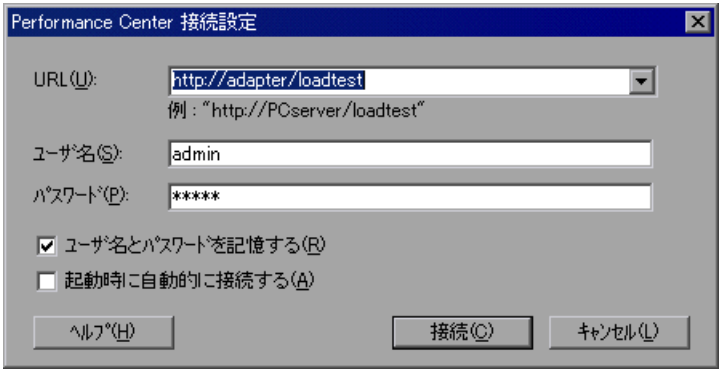
VuGen と Performance Center が連携して動作することにより，Performance Center との間で仮想ユーザ・スクリプトのアップロードとダウンロードを行う効率的な手段が提供されます。VuGen で Performance Center プロジェクトにアクセスするには，最初に VuGen を Performance Center がインストールされている Web サーバに接続する必要があります。接続後，仮想ユーザ・スクリプトをアップロードまたはダウンロードできます。詳細については，次を参照してください。

- 仮想ユーザ・スクリプトのアップロード
- 仮想ユーザ・スクリプトのダウンロード

Performance Center に接続するには，［Performance Center 接続設定］ダイアログ・ボックスを使用します。

VuGen を Performance Center に接続するには，次の手順を実行します。

1 VuGen で，［**ツール**］＞［**Performance Center への接続**］を選択します。［Performance Center への接続設定］ダイアログ・ボックスが開きます。



2 ［URL］ボックスに，Performance Center がインストールされている Web サーバの URL アドレスを入力します。URL アドレスは，次の形式で指定します。

http:// ＜サーバ名＞ /loadtest

3 ユーザ名とパスワードを入力します。詳しくは Performance Center の管理者に問い合わせてください。

4 ログイン処理を自動化するには，［**ユーザ名とパスワードを記憶する**］を選択します。指定したユーザ名とパスワードはレジストリに保存され，ダイアログ・ボックスを開くたびに表示されます。

5 VuGen の起動時に Performance Center への接続を自動的に開くには，［**起動時に自動的に接続する**］を選択します。VuGen は，表示された設定情報を使って Performance Center への接続を試行します。

6 Performance Center に接続するには，［**接続**］をクリックします。［Performance Center への接続］ダイアログ・ボックスに，接続ステータスが表示されます。

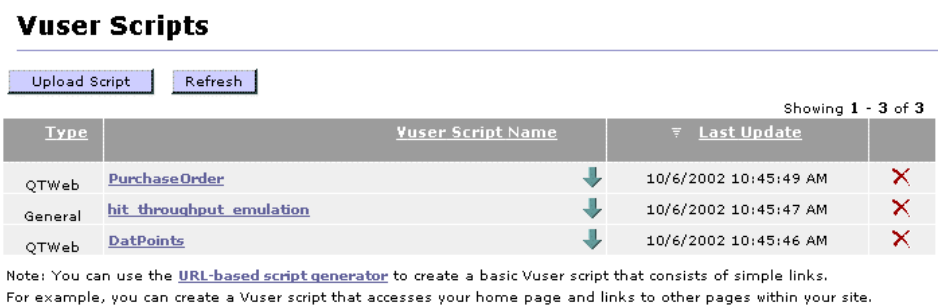
接続が確立されると，すべてのフィールドが読み取り専用で表示されます。

注：接続に失敗すると，接続に失敗した理由を示すダイアログ・ボックスが表示されます。Performance Center と Quality Center に同時に接続することはできません。

仮想ユーザ・スクリプトのアップロード

Performance Center プロジェクトで仮想ユーザ・スクリプトを使用するには，仮想ユーザ・スクリプトを Performance Center サーバにアップロードする必要があります。Performance Center サーバの仮想ユーザ・スクリプトのリストには，プロジェクトで使用できるすべての格納済みの仮想ユーザ・スクリプトが表示されます。

［仮想ユーザ スクリプト］ページを開くには，［**プロジェクト**］メニューの［**仮想ユーザ スクリプト**］を選択します。



仮想ユーザ・スクリプトが仮想ユーザ・スクリプトのリストにすでに追加されている場合は，Performance Center の［仮想ユーザ スクリプト］ページに追加されたスクリプトが表示されます。このリストには，負荷テスト中に仮想ユーザが使用できるすべての仮想ユーザ・スクリプトが示されています。VuGen を使用してアップロードされたスクリプトは，仮想ユーザ・スクリプトのリストに自動的に追加されます。

仮想ユーザ・スクリプトのリストに仮想ユーザ・スクリプトが存在せず，新しいスクリプトを追加する場合は，次の方法でスクリプトを追加できます。

- VuGen からの仮想ユーザ・スクリプトのアップロード
- Performance Center からの仮想ユーザ・スクリプトのアップロード

使用している VuGen のバージョンの確認

アップロードを処理するには，使用しているバージョンの VuGen を正しく設定し，VuGen を Performance Center に接続する必要があります。

VuGen のアップロード設定が正しいことを確認するには，次の手順を実行します。

1 VuGen を起動し，新規または既存の仮想ユーザ・スクリプトを開きます。

2 ［**ツール**］を選択します。

ドロップダウン・メニューに［**Performance Center への接続設定**］というメニュー項目が存在する場合は，使用しているバージョンの VuGen でスクリプトをアップロードできます。

使用しているバージョンの VuGen でスクリプトをアップロードできない場合や，マシンに VuGen をインストールしていない場合は，新しいバージョンの VuGen をインストールする必要があります。古いバージョンはアンインストールすることをお勧めします。VuGen をアンインストールするには，［**スタート**］メニューから［仮想ユーザ ジェネレータ］のアンインストール・オプションを選択します。

新しいバージョンの VuGen をインストールするには，次の手順を実行します。

1 ［**その他**］メニューから，［**ダウンロード**］を選択します。

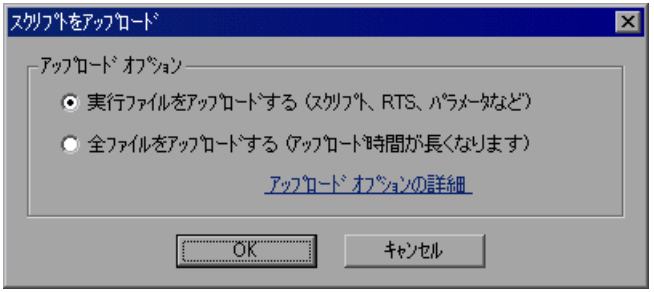
2 ［**Standalone LoadRunner VuGen**］を選択します。

3 ダウンロードの手順を進めます。

アップロードに対応する VuGen のバージョンをインストールすると，既存のスクリプトをアップロードしたり，アップロードするスクリプトを新たに記録したりできます。

アップロード・オプション

VuGen の内部から Performance Center Web サイトのスクリプト・リポジトリに仮想ユーザ・スクリプトをアップロードできます。要件に応じて，部分的なアップロードと完全なアップロードのいずれかを実行できます。次のオプションが使用できます。



- ［**実行ファイルをアップロードする**］：VuGen は最初にサーバからすべての主要なスクリプト・ファイル（**usr**，**c**，**cfg**，および **xml** ファイル）を削除します。データ・ファイルまたは古い記録データは削除されません。次に，VuGen はスクリプト・ファイル，実行環境設定，パラメータ・ファイルをアップロードします。
- ［**全ファイルをアップロードする**］：VuGen は最初にすべてのスクリプト・ファイルおよびデータ・ファイルをサーバから削除します。現在のスクリプト・ファイルとデータ・ファイル（記録データや再生の結果ディレクトリを含む）がアップロードされます。

実行時ファイルのみをアップロードすると，VuGen はすべての記録データと再生結果をアップロードするのではなくスクリプト・ファイルだけをアップロードするため，時間が短縮されます。

注：スクリプト・ファイルがすでにダウンロードされている場合は，標準ではダウンロードされたファイルだけがアップロードされます。新規に作成されたファイルをアップロードする場合（たとえば，ダウンロードしたスクリプトを再生してスナップショットを作成した場合など）は，すべてのファイルをアップロードするように指定する必要があります。

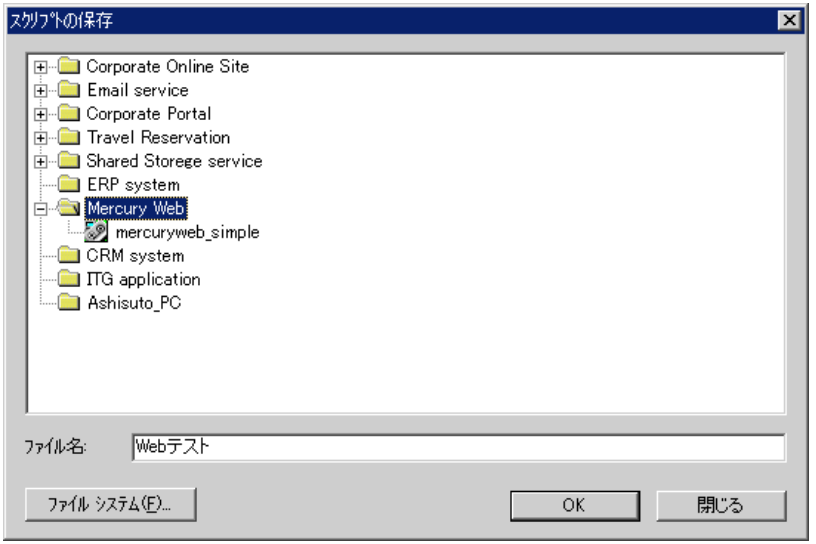
VuGen からの仮想ユーザ・スクリプトのアップロード

VuGen に接続すると，スクリプト・ファイルを Performance Center サーバにアップロードできます。詳細については，270 ページ「VuGen から Performance Center への接続」を参照してください。

VuGen が Performance Center に接続されていない場合は，仮想ユーザ・スクリプトをローカルのファイル・システムに保存できます。後で VuGen が Performance Center に接続されたら，VuGen でスクリプトを開き，それを以下に説明する方法で Performance Center にアップロードします。

VuGen から Performance Center に仮想ユーザ・スクリプトをアップロードするには，次の手順を実行します。

1 VuGen で，［**ファイル**］＞［**保存**］を選択します。［スクリプトの保存］ダイアログ・ボックスが開きます。



2 スクリプトの保存先となるプロジェクトを選択します。スクリプトの名前を［**ファイル名**］ボックスに入力します。

注：ファイル名は，英文字，数字，アンダーバーのみで構成し，250 文字以内で入力する必要があります。

---

3 [**OK**] をクリックします。[スクリプトをアップロード] ダイアログ・ボックスが表示されます。アップロード・オプション ([**実行ファイルをアップロードする**] または [**全ファイルをアップロードする**]) のいずれかを選択します。

4 [**OK**] をクリックすると，ファイルが Performance Center サーバにアップロードされます。

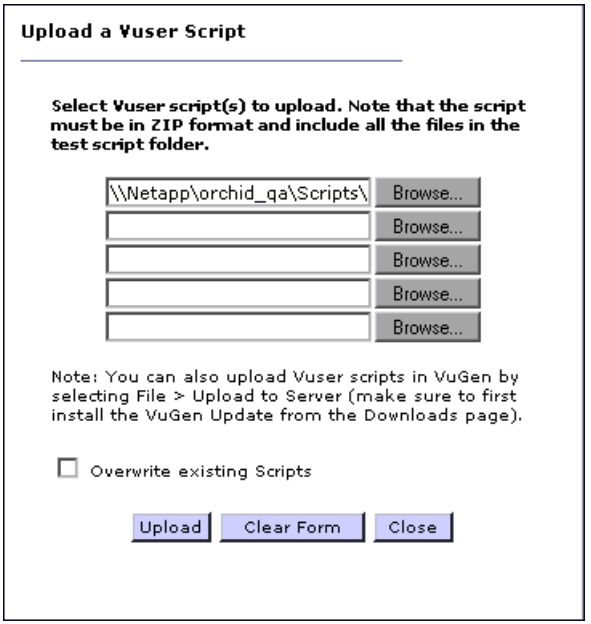
Performance Center からの仮想ユーザ・スクリプトのアップロード

VuGen がインストールされていない場合でも，Performance Center の [**仮想ユーザ スクリプト**] ページにある [スクリプトをアップロード] 機能を使用してスクリプトをアップロードできます。

Performance Center から仮想ユーザ・スクリプトをアップロードするには，次の手順を実行します。

1 [仮想ユーザ スクリプト] ページを開きます。

2 [**Upload Script**] をクリックします。[Upload a Vuser Script] ダイアログ・ボックスが表示されます。



3 [**Browse**] ボタンをクリックして，アップロードする各仮想ユーザ・スクリプトが格納されている zip ファイルを参照します。zip ファイルには，仮想ユーザ・スクリプト・フォルダの完全な内容（仮想ユーザ・スクリプト・ファイル（「.usr」ファイル）そのものと，関連するすべてのデータ・ファイルを含む）が格納されている必要があります。

4 zip ファイルを選択して，[**OK**] をクリックします。

5 仮想ユーザ・スクリプトのリストにすでに存在するスクリプトを置き換える場合は，[**Overwrite existing Scripts**] チェック・ボックスを選択します。

6 [**Upload**] をクリックすると，スクリプトがアップロードされ，仮想ユーザ・スクリプトのリストに追加されます。

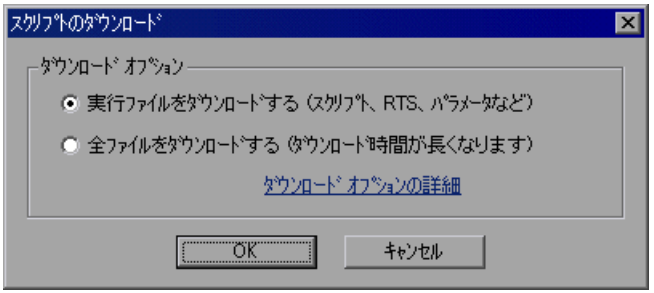
仮想ユーザ・スクリプトのダウンロード

VuGen と Performance Center が連携して動作することにより，Performance Center から仮想ユーザ・スクリプトをダウンロードし，VuGen で自動的に開いて編集するための効率的な手段が提供されます。スクリプト実行時のファイルのみをダウンロードするか，記録データや再生結果を含む完全なファイルをダウンロードするかを選択できます。

VuGen で Performance Center プロジェクトにアクセスするには，最初に VuGen を Performance Center がインストールされている Web サーバに接続する必要があります。接続後，ダウンロードするスクリプト・ファイルを選択できます。[Performance Center への接続設定] ダイアログ・ボックスから Performance Center に接続します。詳細については，270 ページ「VuGen から Performance Center への接続」を参照してください。

ダウンロード・オプション

Performance Center のスクリプト・リポジトリから VuGen に仮想ユーザ・スクリプトをダウンロードできます。要件に応じて，部分的なダウンロードと完全なダウンロードのいずれかを実行できます。次のオプションが使用できます。



- ［**実行ファイルをダウンロードする**］: VuGen は，ダウンロード時間を短縮するために，スクリプト・ファイルのみをダウンロードします。ダウンロードには，スクリプト・ファイル，実行環境の設定，およびパラメータ・ファイルが含まれます。
- ［**全ファイルをダウンロードする**］: スクリプト・ファイルとデータ・ファイル（記録データや再生結果ディレクトリを含む）がダウンロードされます。

実行時のファイルのみを対象とする部分的なダウンロードは，スクリプト・ファイルがダウンロードされるだけなので高速です。スクリプト・ファイルとデータ・ファイルをすべてダウンロードすると，ダウンロードに多くの時間がかかります。

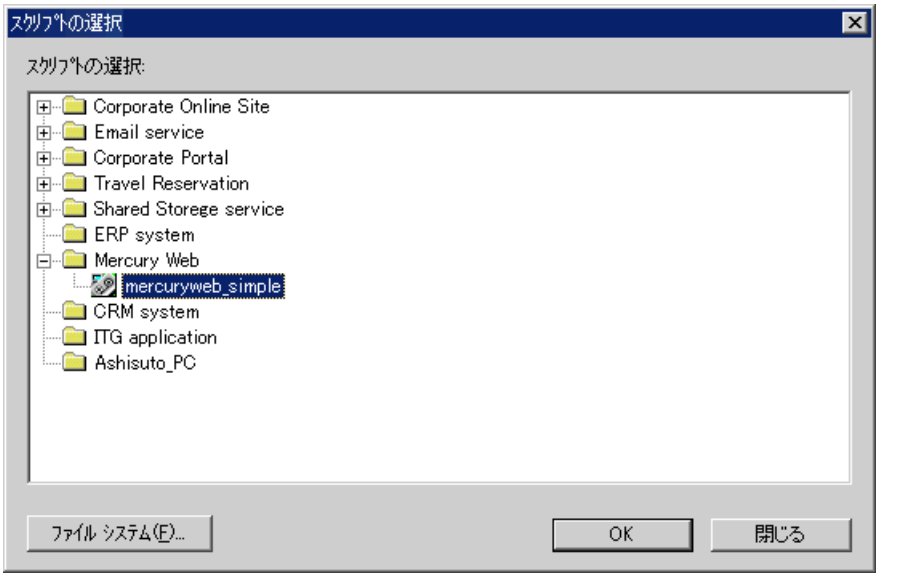
VuGen からの仮想ユーザ・スクリプトのダウンロード

Performance Center サーバに接続すると，スクリプト・ファイルを VuGen にダウンロードできます。

Performance Center から仮想ユーザ・スクリプトをダウンロードするには，次の手順を実行します。

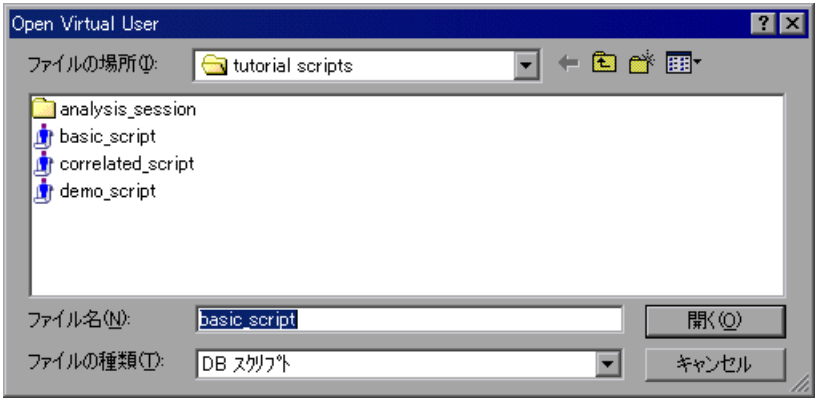
1 Performance Center サーバに接続します。詳細については，270 ページ「VuGen から Performance Center への接続」を参照してください。

2 VuGen で，［**ファイル**］＞［**開く**］を選択します。［スクリプトの選択］ダイアログ・ボックスが表示されます。



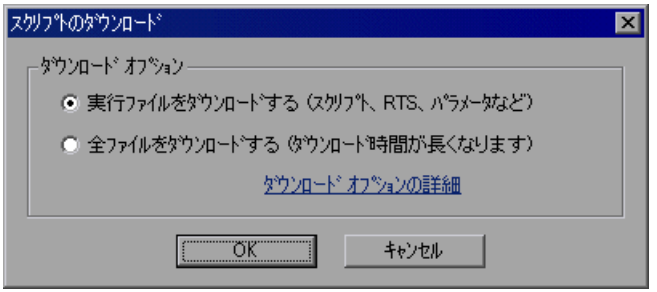
3 ダウンロードするスクリプトを選択します。

（Performance Center に接続していても）ローカル・ドライブからスクリプトを選択するには，［**ファイル システム**］をクリックします。［Open Virtual User］ダイアログ・ボックスが表示されます。



ダウンロードするファイルを選択して，［**開く**］をクリックします。Performance Center の［スクリプトの選択］ダイアログ・ボックスが再度表示されます。ダウンロードするスクリプトを選択します。

4 ［**OK**］をクリックします。［スクリプトのダウンロード］ダイアログ・ボックスが表示されます。



5 ダウンロード・オプションを［**実行ファイルをダウンロードする**］または［**全ファイルをダウンロードする**］のいずれかから選択します。

6 ［**OK**］をクリックすると，Performance Center からファイルがダウンロードされます。ダウンロードが完了すると，ダイアログ・ボックスが閉じてスクリプトが表示されます。

標準設定では，ダウンロードしたファイルは temp ディレクトリに保存されます。ファイルを別のディレクトリに保存するには，［**保存**］をクリックしてディレクトリを指定します。

第 3 部

SOA および Web サービスのテスト

第 19 章

サービス・エミュレーションの作成

サービス・エミュレーション・ツールでは，テストをする目的で Web サービスのエミュレーションを作成できます。

本章では，次の項目について説明します。

- サービス・エミュレーションの作成について
- エミュレーション・サービスの開始
- ホストの選択
- 新規サービスの追加
- エミュレートされたサービスの動作の設定
- エミュレートされたサービスの操作
- 仮想ユーザ・スクリプトでのエミュレートされたサービスの使用

サービス・エミュレーションの作成について

Mercury の SOA ソリューションでは，読者の環境で他の Web サービスをテストするために，サービスのエミュレーションを作成できるサービス・エミュレーション・ツールを提供しています。

エミュレートされたサービスを利用することで，実際のサービスにはアクセスできない開発の初期段階からでもテストを作成して実行できます。たとえば，サービスの開発がまだ完了していなかったり，サービスのホストが利用できなかったりした場合，エミュレートされたサービスを使用してアプリケーションの他のサービスをテストできます。

サービス・エミュレーションのインタフェースでは，サービスの操作とパラメータを指定する WSDL 文書をサービスに関連付けます。ルールと遅延を指定することでサービスの動作を定義します。

エミュレートされたサービスを作成するための手順は次のとおりです。

- エミュレーション・サービスの開始
- ホストの選択
- 新規サービスの追加
- エミュレートされたサービスの動作の設定
- エミュレートされたサービスの操作

エミュレーション・サービスの開始

サービス・エミュレーション・ツールでは，エミュレートされたサービスを実行できる Tomcat サーバが用意されています。インストール・プログラムにより，ローカル・マシンで実行されている Tomcat サーバを使用したサービスの実行を可能にする Axis サーブレットがインストールされます。

LoadRunner では，Tomcat サーバを手動でインストールする必要があります。サーバをインストールしたり，Mercury Service Emulation のインストールを更新したりするには，［**スタート**］＞［**プログラム**］＞［**Mercury LoadRunner**］＞［**Tools**］＞［**Service Emulation Setup**］を選択して，セットアップを実行します。

エミュレートされたサービスをローカル・マシンで作成または実行するには，エミュレーション・サービスを手作業で起動する必要があります。

エミュレーション・サービスを起動するには，次の手順を実行します。

- ［**スタート**］＞［**プログラム**］＞［**Mercury Service Emulation**］＞［**Start Emulation Service**］を選択します。

エミュレーション・サービスを停止するには，次の手順を実行します。

- ［**スタート**］＞［**プログラム**］＞［**Mercury Service Emulation**］＞［**Stop Emulation Service**］を選択します。

［**サービス**］ダイアログ・ボックス（［**スタート**］＞［**プログラム**］＞［**管理ツール**］＞［**サービス**］）からサービスを手作業で開始・停止することも可能です。サービス名は **Mercury ServiceEmulation** です。

ホストの選択

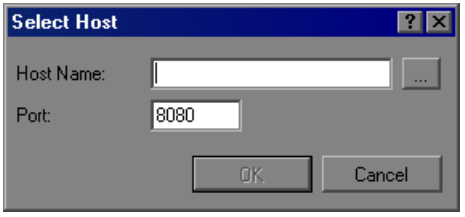
エミュレートされたサービスを作成するための最初の手順として，エミュレートされたサービス用のホストを選択します。セットアップを行ったとき，ローカル・マシンには Tomcat サーバがインストールされます。外部サーバを指定して，Tomcat サーバのインストール先とすることもできます。

標準設定では，サービス・エミュレーション・ツールは localhost をエミュレーション・サーバとして使用します。

外部ホストを追加するには，次の手順を実行します。

1 サービス・エミュレーション・ツールを起動します。［**スタート**］＞［**プログラム**］＞［**Mercury LoadRunner**］＞［**Tools**］＞［**Service Emulation Console**］を選択します。

2 ［**Main**］＞［**New Host**］を選択するか，［**Add Host**］ボタン をクリックして，［Select Host］ダイアログ・ボックスを開きます。



3 ネットワーク内でホストを参照するか，ホスト名を直接入力します。必要があれば，ポートの設定を標準設定値の 8080 から変更します。

4 ［**OK**］をクリックします。ホストが，ウィンドウの左側の表示枠のリストに追加されます。

新規サービスの追加

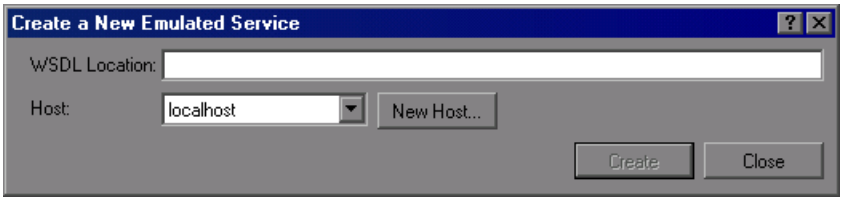
追加したホストごとに，エミュレートされたサービスを作成できます。

サービスを作成するには，サービスの操作とパラメータを定義する WSDL ファイルを指定します。WSDL ファイルを指定すると，サービス・エミュレーション・ツールはその WSDL ファイルの現在の構造を使用して，サービスの入力データおよび出力データの構造を定義します。

元の WSDL が変更されても，変更内容はエミュレートされたサービスには反映されません。更新された WSDL を使用するには，エミュレートされたサービスを作成し直します。

新規サービスを定義するには，次の手順を実行します。

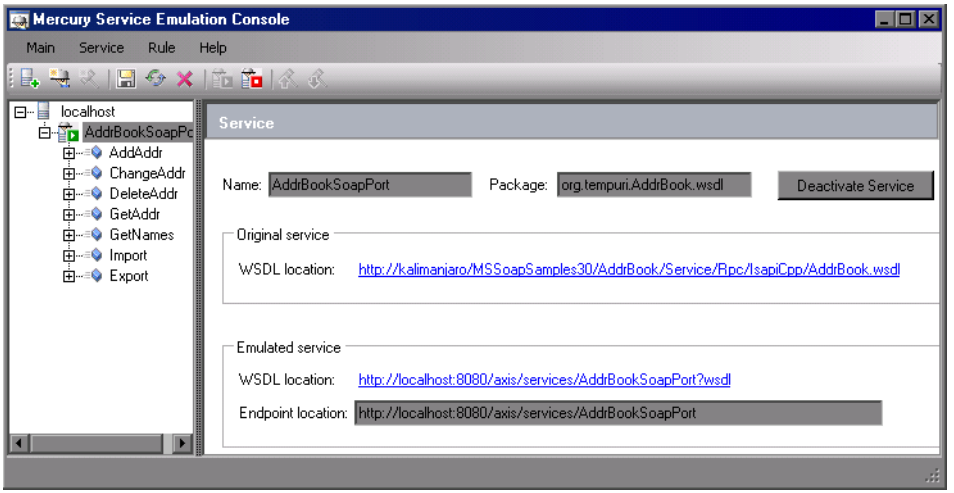
1 ［**Main**］＞［**New Emulated Service**］を選択するか，［**New Emulated Service**］ボタンをクリックして，［Emulate New Service］ダイアログ・ボックスを開きます。



2 ［**WSDL Location**］ボックスに，WSDL ファイルのフル・パスまたは URL を入力します。

3 リストからホストを選択するか，［**New Host**］をクリックして，新しいホストを追加します。

4 ［**Create**］をクリックすると新しいサービスが追加されます。ウィンドウの左側の表示枠にすべてのサービスが一覧表示されます。



5 特定のサービスを削除するには，左側の表示枠で削除するサービスを選択し，右クリックして表示されるメニューから［**Delete**］を選択します。

エミュレートされたサービスの動作の設定

サービス・エミュレーションでは，WSDL に定義されている各操作の動作を指定できます。サービスの操作の動作を指定するには，次を実行します。

- 標準の応答の設定
- ルールの設定

標準の応答の設定

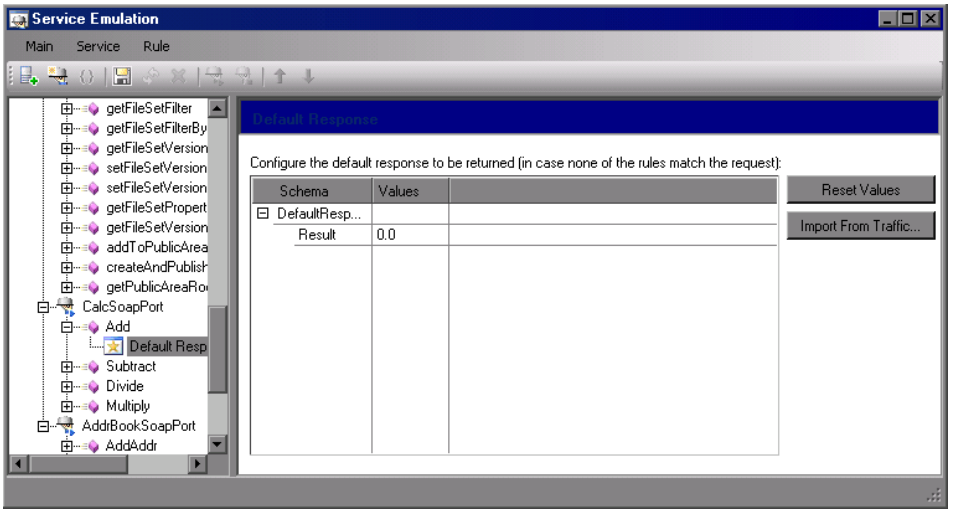
標準の応答とは，操作が供給すると期待される標準設定の値です。

標準の応答の値は，手作業で指定するか，またはサンプル結果を含む XML ファイルからインポートできます。

標準の応答を指定するには，次の手順を実行します。

1 左側の表示枠で，サービスと対象操作を展開します。

2 対象サービスの **Default Response** ノードを選択します。右側の表示枠に Default Response テーブルが表示されます。



3 [**Values**] カラムに標準の応答の値を指定します。

4 キャプチャされたトラフィックから値をインポートするには，[**Import from Traffic**] をクリックします。

5 必要なトラフィックを含んでいる XML ファイルを選択して，[**OK**] をクリックします。

6 標準設定の応答値を使用するようにサービスを設定するには，[**Reset Values**] をクリックします。

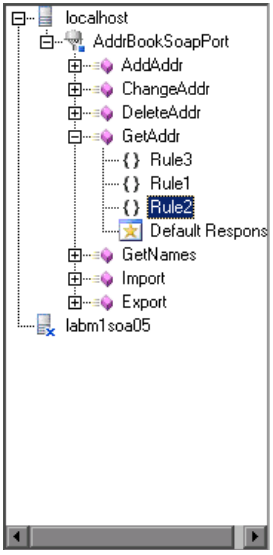
ルールの設定

標準の応答の設定に加えて，動作のルールも設定できます。ルールにより，サービス固有の動作，つまり要求および入力データに基づいた，期待される応答を定義します。

ルールを設定するには，引数および対応する応答に対して 1 つ以上の値を挿入します。たとえば，加算操作の場合，1 番目の引数に 4，2 番目の引数に 5，そして 9 という結果を指定できます。

必要に応じて，特定の引数を除外できます。これは，サービスがいずれかの引数の値にかかわらず，ある特定の応答を返すようにしたい場合に役立ちます。たとえば乗算操作の場合，1 番目の引数が 0 の場合，2 番目の引数の値に関係なく結果が 0 となるようなルールを設定できます。

操作に対して複数のルールを設定できます。ルールは優先度の順に配置します。競合が生じた場合，上位のルールが下位のルールに優先します。次の例では，優先順位が一番高いのはルール 3 です。



ルールを設定するには，次の手順を実行します。

1 左側の表示枠で，サービスと対象操作を展開します。

2 左側の表示枠で操作を選択し，右クリックして表示されるメニューから［**Add New Rule**］を選択するか，あるいは［**New Rule**］ボタンをクリックします。

3 要求と応答の値を指定します。

4 ルールに影響を与える各引数の横のチェック・ボックスを選択します。

5 ［**Save**］をクリックして，サービスに対して加えたすべての変更を保存します。

6 キャプチャされたトラフィックから値をインポートするには，［**Import from Traffic**］をクリックします。要求と応答の両方に対して値をインポートできます。

7 ルールをさらに追加するには，上記の手順を繰り返します。

8 ルールを上位に移動して優先度を上げるには，ツールバー・ボタンを使用するか，右クリックして［**Move up**］を選択します。

ルールを下位に移動するには，ツールバー・ボタンを使用するか，右クリックして［**Move down**］を選択します。

エミュレートされたサービスの操作

エミュレートされたサービスを作成したら，次の操作ができます。

- エミュレートされたサービスの再ロード
- 操作の遅延の挿入
- サービスの無効化と有効化

エミュレートされたサービスの再ロード

エミュレートされたサービスを再ロードすると，ホスト上に保存されている最後に保存したバージョンのサービスが復元されます。保存されていないルールはすべて失われます。別のユーザが別のマシンからサービスに加えた変更を保存した場合，サービスを再ロードをすることで，それらの変更を自分のマシンに反映することができます。

この機能は，変更を加えた後にそれらを破棄して，保存されているバージョンに戻りたいときに便利です。この機能のもう一つの使用法は，別のユーザがサービスの動作に変更を加えた場合に，サービスを再ロードしてその変更を取り込むことです。

サービスを再ロードするには，次の手順を実行します。

1 再ロードするサービスを選択します。

2 ［**Reload**］ボタン をクリックするか，右クリック・メニューから［**Reload**］を選択します。

操作の遅延の挿入

より正確にサービスをエミュレートするために，各操作の応答に対して遅延時間を挿入できます。この遅延の設定により，アプリケーションから要求が送信されてからサーバが応答を返すまでの遅延がエミュレートされます。

遅延を挿入するには，次の手順を実行します。

1 左側の表示枠で操作を選択します。

2 右側の表示枠で，遅延時間を秒単位で入力します。

サービスの無効化と有効化

サービスを作成したら，サービスを削除する代わりに一時的に無効にできます。サービスを無効にすると，そのサービスはテストの中で無視されるようになりますが，将来そのサービスのテストを実装する必要が生じたときにいつでも利用できます。

エミュレートされたサービスの一覧の中で，緑色の四角はサービスが有効であることを示し，赤色の四角は無効であることを示します。

サービスを無効にするには，サービスを左の表示枠の中で選択して，［**Deactivate**］ボタンをクリックするか，右側の表示枠で［**Deactivate Service**］をクリックします。

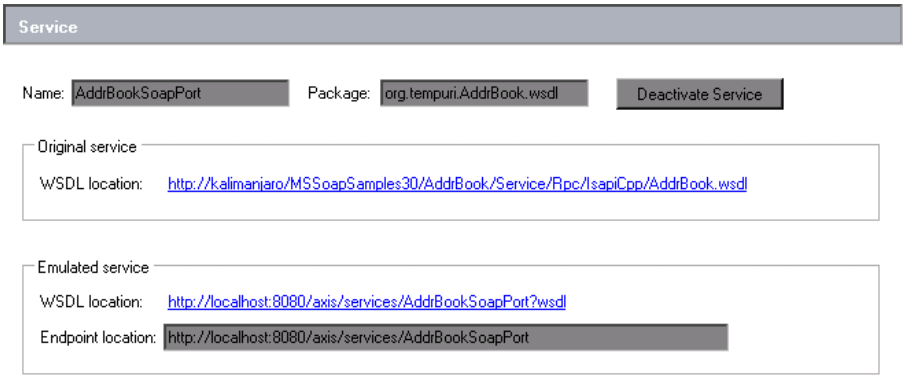
無効にしたサービスを有効にするには，左の表示枠の中で対象サービスを選択して，［**Activate**］ボタンをクリックするか，右側の表示枠で［**Activate Service**］をクリックします。

仮想ユーザ・スクリプトでのエミュレートされたサービスの使用

エミュレートされたサービスの作成後，それを仮想ユーザ・スクリプトにインポートしてテストで使用できます。

エミュレートされたサービスを使用するには，次の手順を実行します。

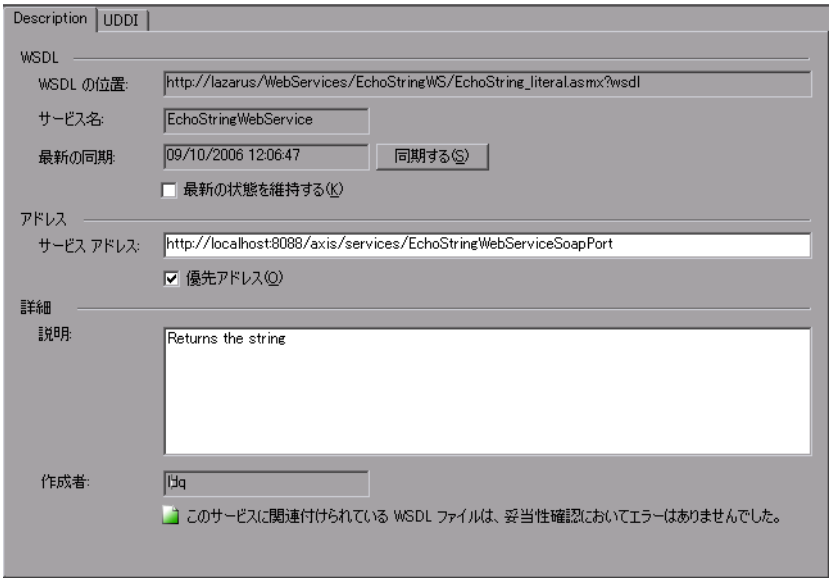
1 左の表示枠でサービスを選択し，右の表示枠の中で［**Endpoint location**］を選択してクリップボードにコピーします。



2 ［Service Management］ウィンドウを開き，［**SOA ツール**］＞［**Service Management**］を選択するか，［Service Management］ツールバー・ボタンをクリックします。

3 エミュレートするサービスを選択します。

4［**Override address**］オプションを有効にします。



5［**Endpoint location**］の値を［**Service Address**］ボックスに貼り付けます。テスト実行の間，サービスは指定された位置に対して応答します。

第 20 章

SOA 向けのテスト・スクリプトの作成

VuGen を使用して，セッションの記録，ネットワーク・トラフィックの分析，または手作業でのサービス動作の指定をすることにより，SOA テストを作成します。VuGen のテスト生成ウィザードで自動テストを作成できます。

本章では，次の項目について説明します。

- SOA テスト・スクリプトの作成について
- SOA テスト・スクリプトの概要
- テスト・アスペクトの選択
- 自動テストの作成
- スクリプトの表示と編集

SOA テスト・スクリプトの作成について

SOA システムは Web サービスに基づいて，インターネットをまたいでさまざまなプラットフォームの上で広く実行できる自己完結型のアプリケーションです。サービスは，XML（Extensible Markup Language）と SOAP（Simple Object Access Protocol）を使って作成されます。Web サービスは，新しいアプリケーションの開発，配備を短期間で実現する積み木の役割を果たします。

VuGen の生成ウィザードは，サービスをテストするスクリプトの作成工程を導いてくれます。ウィザードを使って，サービスのどのようなアスペクト（特徴・側面）をテストするのかを指定します。テストの対象にできるアスペクトとしては，さまざまなツールキットとの相互運用性，境界条件，標準規格への準拠などがあります。

VuGen では，選択したアスペクトごとにスクリプトが自動的に生成されます。

SOA テスト・スクリプトの概要

本項では，VuGen を使って SOA テスト・スクリプトを作成する工程の概略を説明します。

テスト・スクリプトを作成するには，次の手順を実行します。

1 **新しい SOA テストを作成します。**

組み込みのウィザードを使用して，特定のテスト・アスペクトを対象とした自動テストを作成します。詳細については，298 ページ「自動テストの作成」を参照してください。

2 **サービスを検証します。**

サービスを検証して，WS-I 標準規格に準拠していることを確認します。詳細については，373 ページ「WSDL ファイルの検証」を参照してください。

3 **チェックポイントを挿入します。**

サービスを検証してチェックポイントを追加します。詳細については，318 ページ「チェックポイントの設定」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。

実行環境の設定の詳細については，第 24 章「SOA/Web サービス・スクリプトの実行」および第 13 章「実行環境の設定」を参照してください。

5 **VuGen でスクリプトを実行します。**

VuGen で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。

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 15 章「スタンドアロン・モードでの仮想ユーザ・スクリプトの実行」を参照してください。

6 **スクリプトを保存します。**

ファイル・システムまたは Quality Center にスクリプトを保存します。Quality Center にスクリプトを保存すると，スクリプトをテスト・セットに保存して，Quality Center から直接機能テストおよび回帰テストを実行できます。

詳細については，Quality Center のマニュアルを参照してください。

7 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，400 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，または Business Process Monitor プロファイル）に組み込みます。詳細については，『**Mercury LoadRunner コントローラ・ユーザーズ・ガイド**』，**Performance Center** のマニュアル，または **Business Availability Center** のマニュアルを参照してください。

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

テスト・アスペクトの選択

VuGen の生成ウィザードは，サービスのさまざまなアスペクトをテストするスクリプトの作成に役立ちます。アスペクトごとに個別のスクリプトが作成されます。アスペクトに下位のアスペクトがある場合，VuGen により下位のアスペクトごとに個別のスクリプトが作成されます。各アスペクトには個別のスクリプトがあるため，Quality Center はテストのアスペクトごとに成功または失敗のステータスを割り当てることができます。

テスト・アスペクト

VuGen は，次のテスト・アスペクトをサポートします。

- **正常系テスト**：選択されたサービスを対象とした網羅的な正常系テストを生成します。サービスの操作の個別テストおよび相互テストの両方を行います。
- **標準規格への準拠**：サービスが WS-I や SOAP などの業界標準規格に正しく準拠しているかどうかを確認します。
- **サービスの相互運用性**：サービスの操作が，サポートされているすべての Web サービス・ツールキットと相互運用可能なことを確認します。このアスペクトに対しては複数のテストが作成されます。

- **セキュリティ・テスト**：
  - **SQL インジェクション**：SQL ステートメントとエラーを関係するパラメータに挿入し，SQL インジェクションに対してサービスが脆弱であるかどうかを確認します。
  - **クロス・サイト・スクリプティング**：複数のサイトにまたがってサービスを確認します。
- **境界条件のテスト**：異常系テストを使用してサービスをテストします。サービスの記述に基づき，サービスをその限界までテストすることを目的として VuGen によりデータ，タイプ，パラメータおよび実際の SOAP メッセージを操作するテストが作成されます。
  - **極端値**：サービスに無効なデータ・タイプを渡し，それらが受け付けられないことを確認します。
  - **Null 値**：サービスに NULL パラメータを渡し，それらが受け付けられないことを確認します。

自動テストの作成

テスト生成ウィザードは，自動テストの作成の手順を導いてくれます。

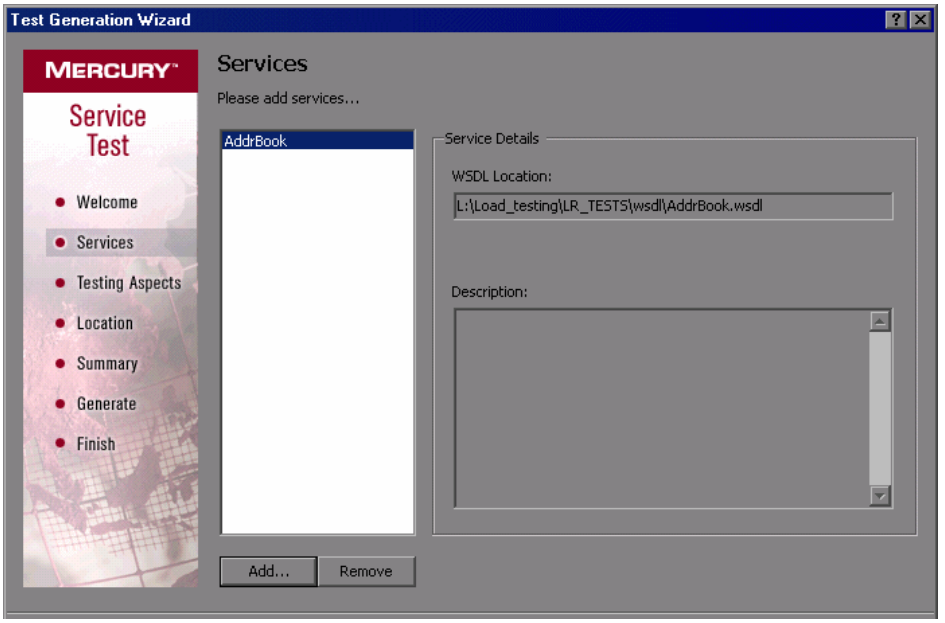
自動テストを作成するには，次の手順を実行します。

1 ［**ファイル**］＞［**新規作成**］を選択して，［New Virtual User］ダイアログ・ボックスを開きます。

2 左側の表示枠で，［**SOA Script Generator**］を選択します。

3 ［Selected Protocols］ボックスに［**Web Services**］が表示されていることを確認します。［**OK**］をクリックします。

4 テスト生成ウィザードの［Welcome］画面で，［**Next**］をクリックします。

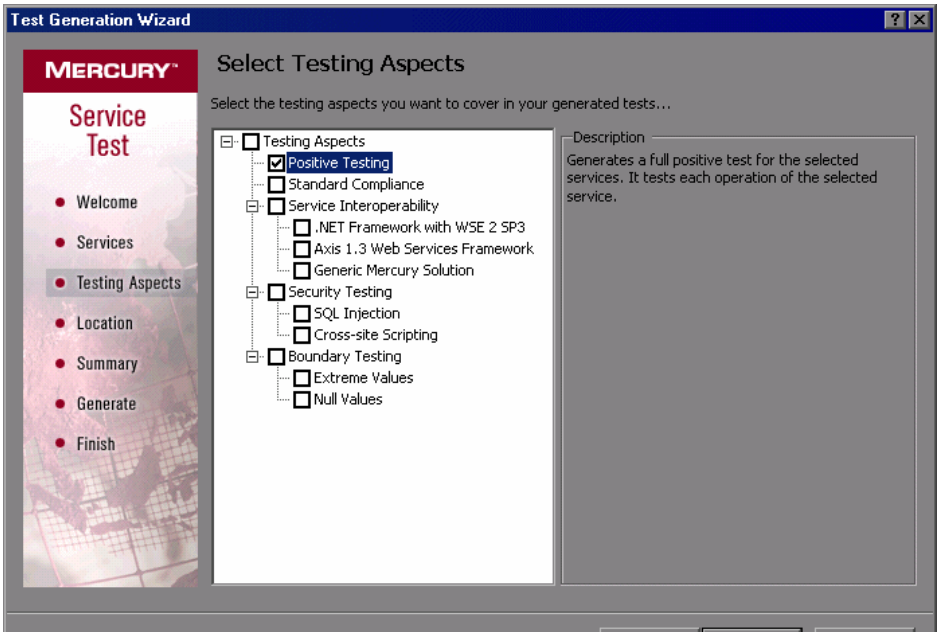


5 テストに少なくとも 1 つのサービスを追加します。［**Add**］をクリックし，［Import Service］ダイアログ・ボックスを開きます。

6 WSDL の情報を指定します。詳細については，369 ページ「サービスのインポート」を参照してください。

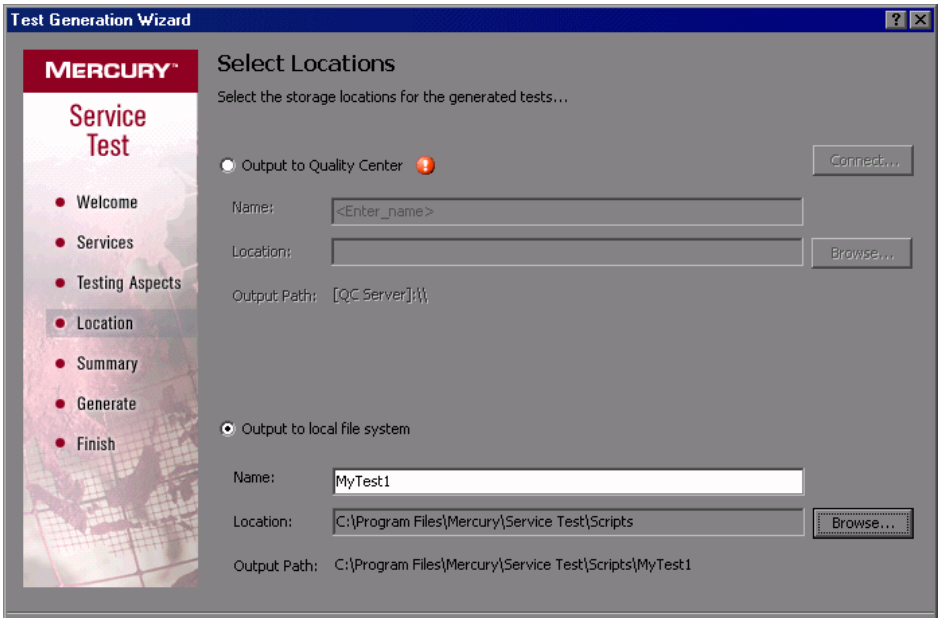
7 インポートするすべてのサービスについて，この手順を繰り返します。［**Next**］をクリックして，テスト・アスペクトを選択します。

8 必要なテスト・アスペクトおよび下位のアスペクトを選択します。詳細については，297 ページ「テスト・アスペクトの選択」を参照してください。



［**Next**］をクリックして，テストの出力場所を選択します。

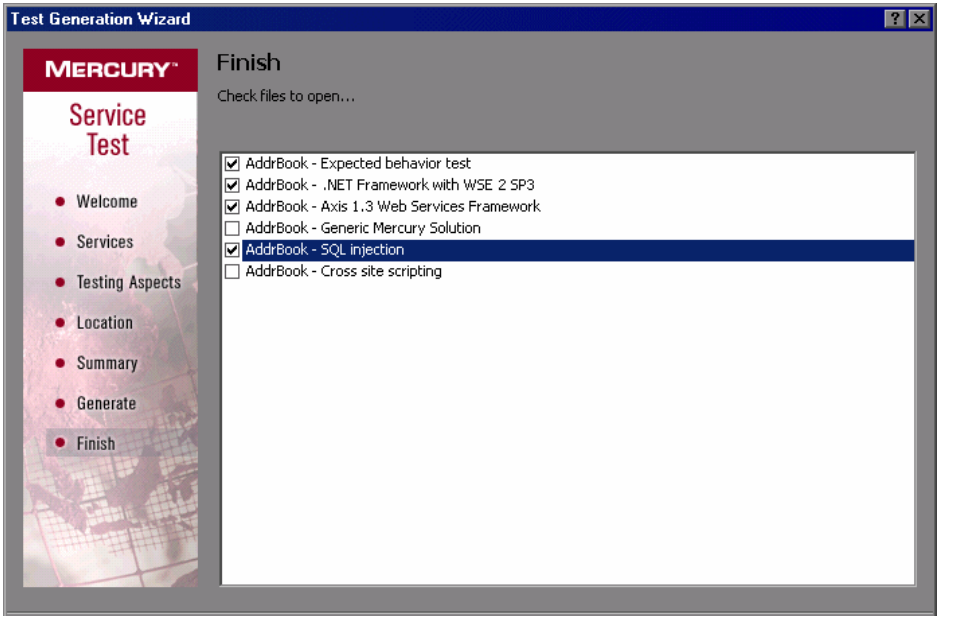
9 テスト名とテスト・スクリプトの場所（**Quality Center** または**ローカル・ファイル・システム**）を指定します。



Quality Center を指定した場合，［**Connect**］をクリックしてサーバにログオンし，［**Browse**］をクリックしてテスト・ノードを探します。

10 ［**Next**］をクリックして，選択したサービス，テスト・アスペクト，および出力場所のサマリを表示します。

11 ［**Generate**］をクリックすると，スクリプトの作成が始まります。VuGen によりテストの生成が完了すると，指定された場所に格納された実際のスクリプトが一覧表示されます。



12 VuGen を使用して開くスクリプトを選択し，［**Finish**］をクリックします。スクリプト・ジェネレータにより，選択したスクリプトが開かれます。

スクリプトの表示と編集

手作業で作成したテスト，またはテスト生成ウィザードによって自動的に作成されたすべてのテストの表示および編集ができます。

テストはツリー・ビューまたはスクリプト・ビューのどちらの形式でも表示できます。ツリー・ビューには，テストの手順がグラフィカル・インタフェースで表示されます。一方，スクリプト・ビューには，サービスをエミュレートする実際の **web_service_call** 関数が表示されます。

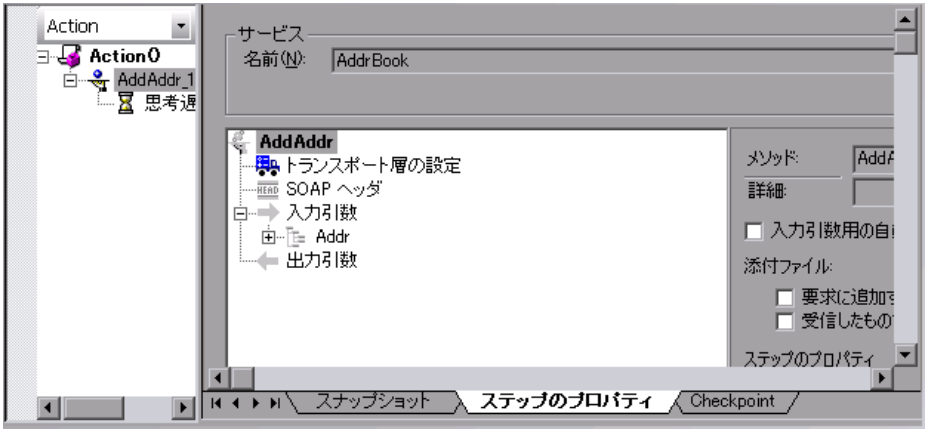
ツリー・ビュー

ツリー・ビューには，スクリプトが視覚的に表示されます。

自動的に作成されたスクリプトをツリー・ビューに表示するには，次の手順を実行します。

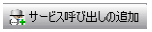
1 ［**ツリー**］ボタンをクリックするか，［**表示**］＞［**ツリー ビュー**］を選択します。

2 **vuser_init** が表示されている場合は，**Action** セクションを選択します。



3 ［**ステップのプロパティ**］タブを選択して，ステップのすべてのプロパティを表示します。

4 特定のノードを選択して，値を表示または変更します。値およびプロパティの変更の詳細については，第 22 章「Web サービス呼び出しプロパティの設定」を参照してください。



5 さらに Web サービス・ステップを追加するには，［**サービス呼び出しの追加**］ボタンをクリックします。詳細については，317 ページ「新しい Web サービス呼び出しの追加」を参照してください。

6 JMS 機能や SAML セキュリティなどの追加ステップを挿入するには，［**挿入**］＞［**新規ステップ**］を選択し，該当するステップを選択します。詳細については，「**オンライン関数リファレンス**」（［**ヘルプ**］＞［**関数リファレンス**］）を参照するか，ステップの挿入後に F1 キーをクリックします。

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

8 チェックポイントを設定するには，［**チェックポイント**］タブをクリックします。詳細については，318 ページ「チェックポイントの設定」を参照してください。

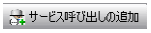
スクリプト・ビュー

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

自動的に作成されたスクリプトをスクリプト・ビューに表示するには，次の手順を実行します。

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

2 **vuser_init** が表示されている場合は，**Action** セクションを選択します。



3 さらに Web サービス・ステップを追加するには，［**サービス呼び出しの追加**］ボタンをクリックします。詳細については，第 22 章「Web サービス呼び出しプロパティの設定」を参照してください。

4 JMS 機能や SAML セキュリティなどの追加ステップを挿入するには，［**挿入**］＞［**新規ステップ**］を選択し，該当するステップを選択します。

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

詳細については，「**オンライン関数リファレンス**」（［**ヘルプ**］＞［**関数リファレンス**］）を参照するか，関数をクリックして F1 キーをクリックします。

VuGen でテストを再生して，テストが正しく動作することを確認します。サービスの検証や管理もできます。詳細については，第 23 章「Web サービスの管理」を参照してください。

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

第 21 章

Web サービス仮想ユーザ・スクリプトの作成

VuGen を使用して，記録またはサーバ・トラフィックの分析によって Web サービスをテストするスクリプトを作成します。

本章では，次の項目について説明します。

- Web サービス仮想ユーザ・スクリプトの作成について
- Web サービス仮想ユーザ・スクリプトの概要
- ワークフローの表示
- Web サービス・スクリプトの記録
- 記録対象アプリケーションの選択
- 新しい Web サービス呼び出しの追加
- チェックポイントの設定

Web サービス仮想ユーザ・スクリプトの作成について

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。サービスは，XML（Extensible Markup Language）と SOAP（Simple Object Access Protocol）を使って作成されます。Web サービスは，新しいアプリケーションの開発，配備を短期間で実現する積み木の役割を果たします。

テスト・ウィザードは，スクリプトを作成してサービスをテストする工程を導いてくれます。

Web サービスをテストするためのスクリプトは，記録，Web サービス呼び出しの手作業での挿入，サーバ・トラフィックの分析，SOA テスト・アスペクトの選択のいずれかの方法で作成できます。

スクリプトの作成方法	参照先
記録	317 ページ「新しい Web サービス呼び出しの追加」
Web サービス呼び出しの手作業での挿入	317 ページ「新しい Web サービス呼び出しの追加」
サーバ・トラフィックの分析	第 25 章「サーバ・トラフィック・スクリプトの作成」
SOA テスト・アスペクトの選択	第 20 章「SOA 向けのテスト・スクリプトの作成」

Web サービス仮想ユーザ・スクリプトの概要

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

テスト・スクリプトを作成するには，次の手順を実行します。

1 **新しい Web サービス・スクリプトを作成します。**

Web サービス・ウィザードおよびワークフロー画面を使用して新規スクリプトを作成し，「Web サービス」仮想ユーザ・タイプを選択します。詳細については，312 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。

2 **Web サービス呼び出しを追加し，引数の値を設定します。**

追加の Web サービス呼び出しを挿入し，それらの値を設定することで，スクリプトを拡張します。

詳細については，第 22 章「Web サービス呼び出しプロパティの設定」を参照してください。

3 **検証を実行してチェックポイントを挿入します。**

サービスを検証してチェックポイントを追加します。詳細については，350 ページ「Web サービス・セキュリティ・ポリシーの作成」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，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。

実行環境の設定の詳細については，386 ページ「Web サービス・ツールキットの実行環境の設定」および第 13 章「実行環境の設定」を参照してください。

5 **VuGen でスクリプトを実行します。**

VuGen で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。

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 15 章「スタンドアロン・モードでの仮想ユーザ・スクリプトの実行」を参照してください。

また，Mercury の Quality Center でテスト計画を作成して，テストを実行することもできます。詳細については，http://www.mercury.com/jp/ を参照するか，担当営業にお問い合わせください。

1 **スクリプトを保存します。**

ファイル・システムまたは Quality Center にスクリプトを保存します。Quality Center にスクリプトを保存すると，スクリプトをテスト・セットに保存して，Quality Center から直接機能テストおよび回帰テストを実行できます。

詳細については，Quality Center のマニュアルを参照してください。

2 **テスト結果を表示します。**

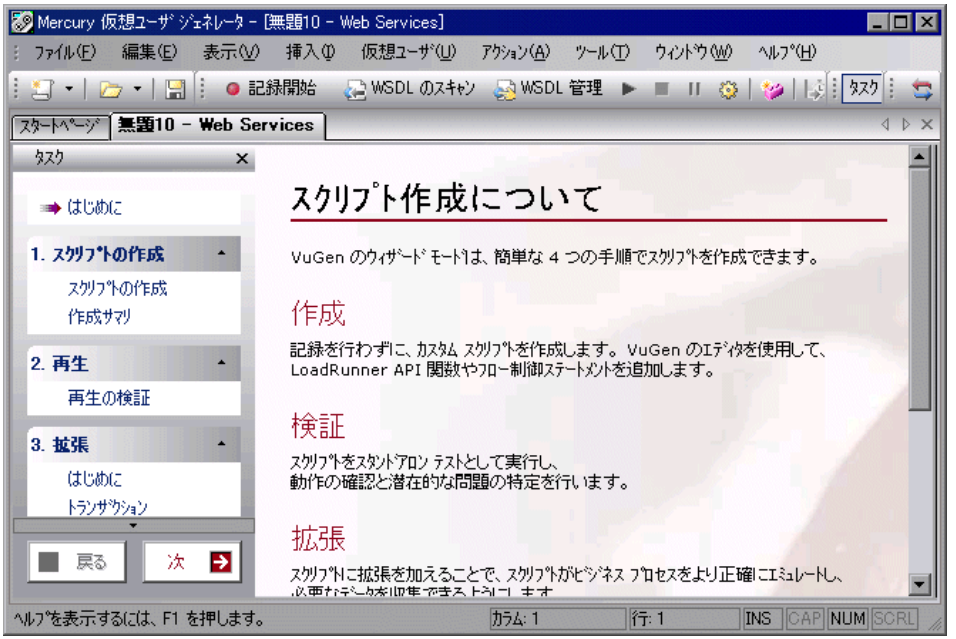
テスト結果ユーティリティを開き，再生のサマリを表示します。詳細については，400 ページ「Web サービス・レポートの表示」を参照してください。

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，または Business Process Monitor プロファイル）に組み込みます。詳細については，**LoadRunner コントローラ** のマニュアル，**Performance Center** のマニュアル，または **Business Availability Center** のマニュアルを参照してください。

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

ワークフローの表示

VuGen のワークフローは，スクリプトの準備の手順を導いてくれます。［タスク］表示枠をクリックすると，スクリプト作成の手順についての説明を読むことができ，記録に関する情報を表示できます。また，再生を検証することもできます。［**戻る**］および［**次**］ボタンを使用して，画面間を移動できます。



ワークフロー画面が見えない場合は，［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。タスクをクリックしてワークフローを表示します。

スクリプトの作成

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

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

2 つのアクションに関するリンクがあります。

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

作成サマリ

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

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

また，スクリプトを変更するための次のリンクも含まれます。

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

ワークフローのその他の詳細については，第 3 章「VuGen ワークフローの表示」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・セッションを記録することにより，一般的なビジネス・プロセスをエミュレートするテストを作成できます。記録の後，Web サービス呼び出し，およびその他の拡張機能を手作業で追加できます。

記録のためのサービスの指定

アプリケーションを記録するときは，Web サービス WSDL ファイルを使用して記録することも，WSDL ファイルを使用せずに記録することもできます。

WSDL ファイルを含める場合，VuGen では使用するメソッドを選択してそれらの引数の値を入力することでスクリプトを作成できます。これにより，記録された SOAP トラフィックの解析および Web サービスへの呼び出しの抽出を行うことで高レベルなスクリプトを作成できるようになります。

WSDL ファイルを指定しない場合，VuGen では Web サービス呼び出しステップの代わりに SOAP 要求を使用してテストが作成されます。

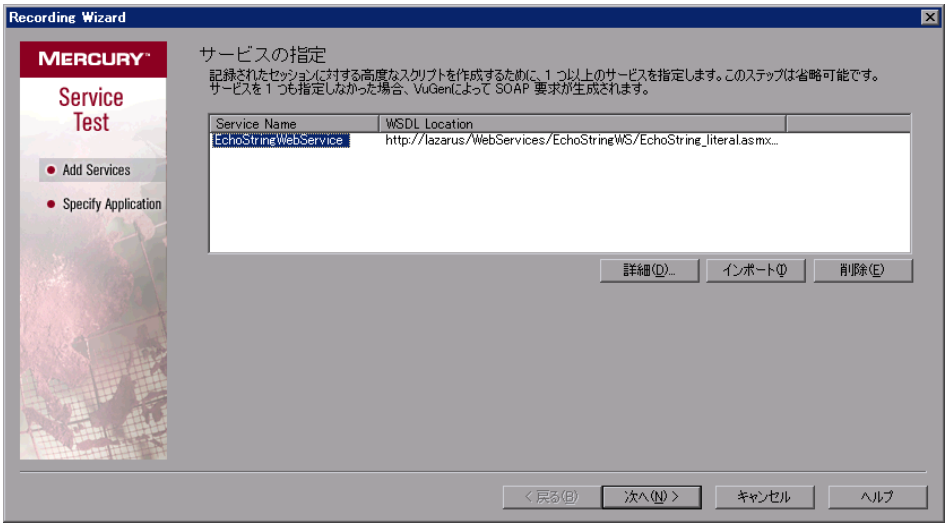
記録によって Web サービス・スクリプトを作成するには，次の手順を実行します。

1 **空のスクリプトを作成します。**

［**ファイル**］＞［**新規作成**］を選択して，［新規仮想ユーザ］ダイアログ・ボックスを開きます。［e ビジネス］カテゴリから［**Web Services**］プロトコルを選択します。［**OK**］をクリックします。

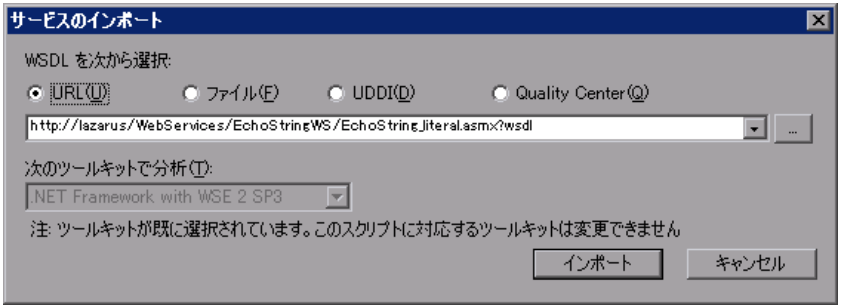
2 **記録プロセスを開始します。**

［**記録開始**］ボタンまたは Ctrl + R キーをクリックします。［サービスの指定］画面が開きます。



3 **必要に応じて，リストにサービスを追加します。**

高レベルの Web サービス・スクリプトを作成するには，［**インポート**］ボタンを使用して 1 つ以上のサービスを追加します。［サービスのインポート］ダイアログ・ボックスが開きます。



4 **WSDL のソースを選択します。**

WSDL とツールキットの場所を指定します。選択したツールキットはスクリプトに永続的に関連付けられます。詳細については，369 ページ「サービスのインポート」を参照してください。

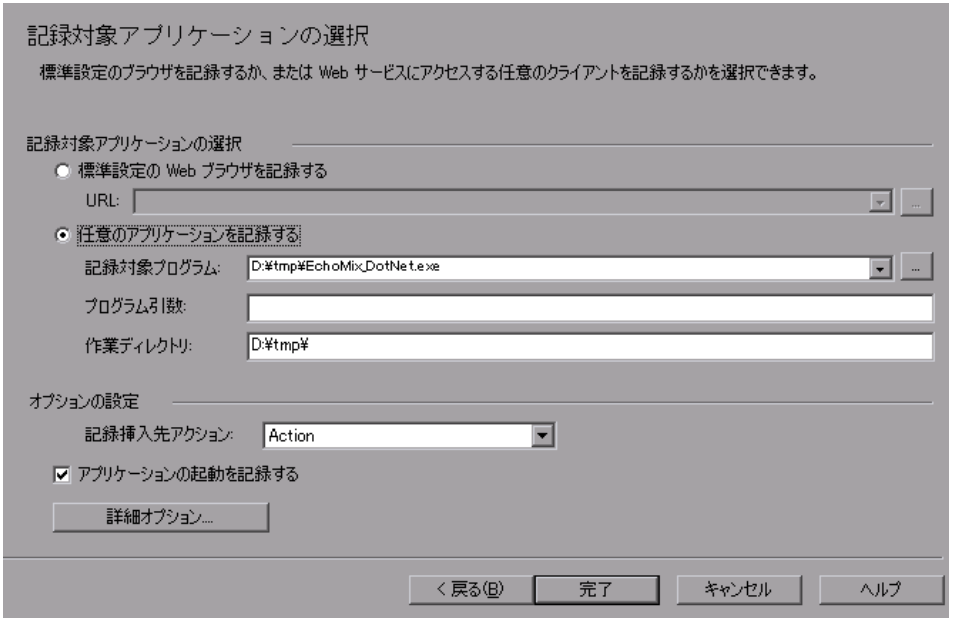
5 **Web サービスの詳細を入力します。**

［**詳細**］をクリックして，追加する Web サービスの詳細を入力します。詳細については，367 ページ「サービス・プロパティの表示と設定」を参照してください。

次の手順では，記録対象アプリケーションを選択します。

記録対象アプリケーションの選択

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



1 記録の詳細を指定します。

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

2 オプションを設定します。

- ［**記録挿入先アクション**］：生成されるコードの挿入先となるアクション。繰り返す必要のない起動処理がある場合は，それらを **vuser_init** セクションに置きます。記録中に **Action** などの別のセクションに切り替えることができます。

➤［**アプリケーションの起動を記録する**］：アプリケーションの起動処理をスクリプトの一部として記録します。特定の位置から，起動処理を含めずに記録を開始する場合は，このチェック・ボックスをクリアします。

➤［**詳細オプション**］：［記録オプション］ダイアログ・ボックスが開き，スクリプトの生成方法をカスタマイズできます。詳細については，第 5 章「スクリプト生成オプションの設定」を参照してください。

3 **［完了］をクリックします。**

VuGen によりアプリケーションが起動され，記録が開始します。対象アプリケーション内で必要なアクションを実行し，その後フローティング・ツールバーの［停止］ボタンを押します。WSDL をインポートしなかった場合，VuGen は **web_service_call** 関数または **soap_request** 関数を記録します。

4 **Web サービス呼び出しを追加します。**

さらに Web サービス・ステップを追加するには，［**サービス呼び出しの追加**］ボタンをクリックします。詳細については，327 ページ「Web サービス呼び出しプロパティの設定」を参照してください。

JMS 機能，SAML アサーション，またはトランザクションなどその他の拡張機能を作成するには，［**挿入**］＞［**新規ステップ**］を選択し，該当するステップを選択します。

5 **スクリプトをパラメータ化します。**

引数の値をパラメータに置き換えるには，スクリプト内で置き換える値を選択し，右クリックして表示されるメニューから［**パラメータで置換**］を選択します。より複雑なタイプの要素の場合には，159 ページ「XML パラメータのプロパティの設定」で説明しているように XML パラメータ・タイプを使用できます。

6 **スクリプトを保存します。**

ファイル・システムまたは Quality Center にスクリプトを保存します。

7 **スクリプトを実行します。**

ファイル・システムにスクリプトを保存した場合，スクリプトをスタンドアロン・テストとして VuGen から直接実行します。スクリプトが正しく機能したら，LoadRunner のシナリオに組み込むことができます。

Quality Center にスクリプトを保存した場合，不具合の追跡やステータス・レポートの生成を行いながらスクリプトを Quality Center から直接実行できます。

新しい Web サービス呼び出しの追加

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

Web サービス呼び出しを追加するには，次の手順を実行します。

1 テスト内の追加先の場所でカーソルをクリックします。

2 ［**サービス呼び出しの追加**］ボタンをクリックします。［新規 Web サービス呼び出し］ダイアログ・ボックスが開きます。使用するサービスと方法を選択します。この新規のスクリプトで，まだ WSDL をインポートしていなければ，この時点でインポートします。詳細については，369 ページ「サービスのインポート」を参照してください。

3 サービスにサンプルの入力値を設定するには，［**入力引数用の自動値を生成する**］を選択します。VuGen によりサンプル値が追加され，値が手作業で設定されたことを示すためにツリー・ノードが青色に変わります。

操作の入力引数をパラメータ化する方法の詳細については，159 ページ「XML パラメータのプロパティの設定」を参照してください。

4 ［トランスポート層の設定］ノードを選択して，SOAP メッセージ用の JMS トランスポート，非同期メッセージング，WS Addressing などの詳細オプションを指定します。詳細については，338 ページ「トランスポート層の設定」を参照してください。

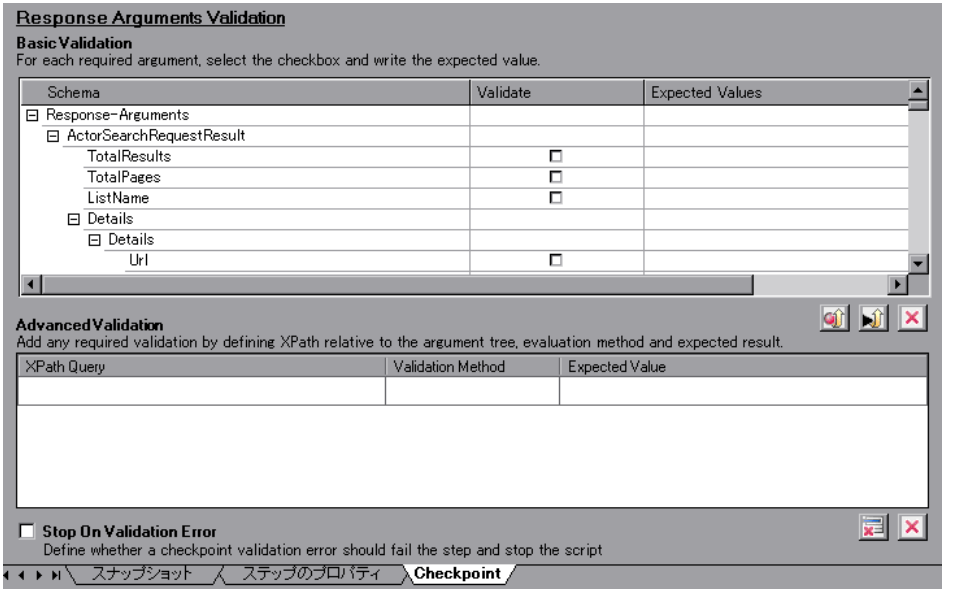
5 各ノードをクリックして引数値に対する設定を指定します。詳細については，第 22 章「Web サービス呼び出しプロパティの設定」を参照してください。

6 入力引数または出力引数に添付ファイルを追加するには，操作のメイン・ノードを選択し，必要な選択を行います。詳細については，334 ページ「添付ファイル」を参照してください。

チェックポイントの設定

機能テストで最も重要なタスクの 1 つは，サーバからの応答をチェックして，テストによって操作が正しく実行されたことを確認することです。Web サービスでは，応答には，それぞれが複数のデータ項目を含んだ引数が複数含まれていることがあります。

VuGen の［**Checkpoint**］タブは，テストに必要なチェックを一元的に定義するために使用します。



テストを実行する前に，引数に対して期待値を設定します。記録中または再生中にキャプチャされた期待値のセットを読み込むことができます。これは値を手動で入力せずに自動的に読み込むことができるので，引数の値が複数ある場合に便利です。

テスト実行の後，再生ログまたはテスト結果を確認して，結果が期待どおりであったかどうかを調べることができます。

VuGen では，メソッドのすべての引数がチェック・ボックス付きで表示されます。チェックポイントをテストに含めるには，該当するチェック・ボックスを選択します。記録された値または再生の値（存在する場合）を読み込んで，チェックする値のみを選択することができます。

オプションの［**Stop On Validation Error**］フラグは，チェックポイントが失敗した場合にステップを失敗したものとするかどうかを示します。

スクリプトでは，VuGen によって，［Checkpoint］タブで選択した行の数だけ CHECKPOINT 引数が追加されます。

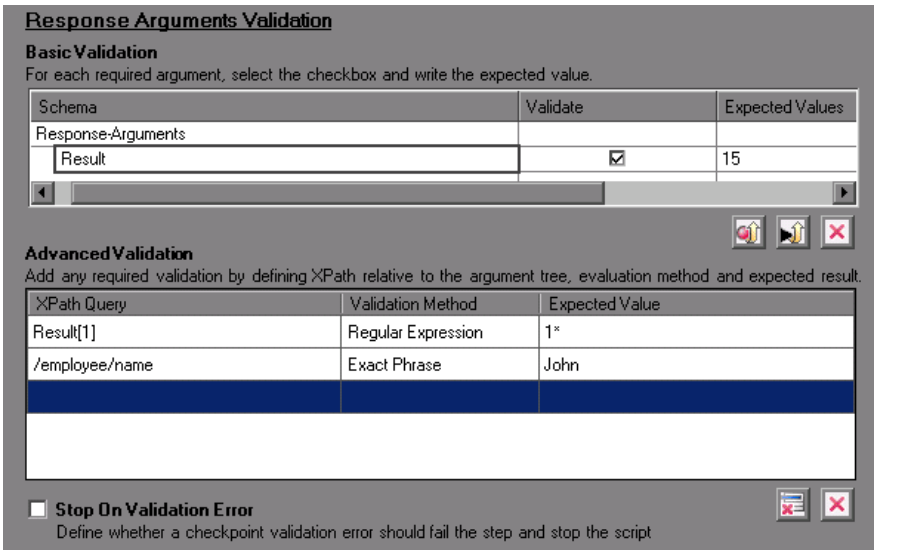
```
web_service_call( "StepName=Add_2",
        "SOAPMethod=Calc.CalcSoapPort.Add",
…
        BEGIN_CHECKPOINTS,
            StopOnValidationError=1,
            CHECKPOINT, "XPATH=Result[1]", "Value=13",
            CHECKPOINT, "XPATH=AddResult", "Expression=Hel*?",
        END_CHECKPOINTS,
        LAST);
```

VuGen では，**lr_xml_find** を使用して標準の XPATH 検証を行うことが可能です。メッセージ本体の検証については，チェックポイントを使用します。SOAP ヘッダーはチェックポイントを使用して検証できないため，その検証には **lr_xml_find** 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**」を参照してください（［**ヘルプ**］＞［**関数リファレンス**］）。

期待値のタイプ

再生中，VuGen では，検証のために期待値のセットが作成されます。これらは上部の［**Basic Validation**］セクションに一覧表示されます。

基本検証に加え，［**Advanced Validation**］を定義して，末端ノードではないノードを対象としたチェックポイントを検証したり，期待値を正規表現で定義したりできます。



基本検証では，VuGen は，［**Expected Values**］カラムの値に完全に一致する値を検索します。

詳細検証では，VuGen は，完全一致または正規表現に基づく一致を検索します。

詳細検証の値を定義するには，［**Advanced Validation**］セクションで XPATH クエリを入力します。出発点となる XPATH 表現を取得するには，基本検証からコピーして（右クリック・メニューから［**Copy Row XPath**］），［**Advanced Validation**］セクションに貼り付けることができます。

末端ノードではないノードを選択する場合，そのノードより下にある XML のすべてを指定する必要があります。

同じステップに基本検証と詳細検証の両方を定義することができます。

仮想ユーザ・スクリプト内では，VuGen により，完全一致は "Value=" で示し，正規表現は "Expression=" で示します。

```
BEGIN_CHECKPOINTS,
        CHECKPOINT, "XPATH=/AddResult[1]", "Value=50"
        CHECKPOINT, "XPATH=AddResult", "Expression=Hel*?",
END_CHECKPOINTS,
```

基本チェックポイントを設定するには，次の手順を実行します。

1 ［ツリー ビュー］（［**表示**］ > ［**ツリー ビュー**］）で，左側の表示枠からステップを選択します。

2 ［**Checkpoint**］タブを選択します。

3 上部のセクションにある［Clear All］ボタンをクリックし，期待値をすべてクリアします。

4 完全一致を検索するには，上部の［**Basic Validation**］セクションに期待値を指定します。

- 期待値を手作業で指定するには，［**Expected Values**］カラムに値を入力します。
- 記録セッションまたは再生セッションからデータを読み込むには，［**Load values from Recording Snapshot**］ボタンまたは［**Load values from Replay Snapshot**］ボタンをクリックします。VuGen は，記録または再生中にキャプチャされたデータを入力します。

5 ［**Validate**］カラムの中で，チェックの対象とするすべての結果のチェック・ボックスを選択します。

6 再生時に期待値が生成されなかった場合にステップを失敗させるよう仮想ユーザに指示するには，［**Stop On Validation Error**］を選択します。

7 スクリプトを実行して再生ログを確認し，サービスが期待値を返したかどうかを調べます。実行環境の設定で詳細ログを有効にすることをお勧めします。

一致がない場合，再生ログには該当する次のようなメッセージが表示されます。

```
Action.c(14): Failure: checkpoint "/AddResult[1]" expected value="3"
actual result="15"
Action.c(14):Error:Web service call "Add" 1/1 checks failed
```

8 ［テスト結果］を開いて（［**表示**］ > ［**テスト結果**］），検証の詳細レポートを確認します。テスト結果の表示の詳細については，次を参照してください。

詳細チェックポイントを設定するには，次の手順を実行します。

1 ［ツリー ビュー］（［**表示**］ > ［**ツリー ビュー**］）で，左側の表示枠からステップを選択します。

2 ［**Checkpoint**］タブを選択します。

3 一番下のセクション［**Advanced Validation**］に期待値を指定します。

- ［**XPath Query**］には，検索の条件を表す表現を指定します。上部ウィンドウの［**Basic Validation**］セクションから XPATH 表現をコピーできます。XPATH 式をコピーするには，［**Schema**］カラムの中でテキストを選択し，右クリック・メニューから［**Copy Row XPath**］を選択します。［**Advanced Validation**］セクションの中で，右クリック・メニューから［**貼り付け**］を選択し，貼り付けた内容を必要に応じて変更します。
- ［**Validation Method**］には，完全一致の表現または正規表現を指定します。
- ［**Expected Value**］には，正確な値または正規表現を指定します。

4 選択した行を削除するには，［**Advanced Validation**］セクションの［**Delete row**］ボタンをクリックします。

5 詳細チェックポイントのすべてクリアするには，［**Basic Validation**］セクションの［**Clear All**］ボタンをクリックします。

6 再生によって期待値が生成されなかった場合は停止するよう仮想ユーザに指示するには，［**Stop On Validation Error**］を選択します。

7 スクリプトを実行して再生ログを確認し，サービスが期待値を返したかどうかを調べます。一致がない場合，再生ログには該当するメッセージが表示されます。

8 ［テスト結果］を開いて（［**表示**］ > ［**テスト結果**］），検証の詳細レポートを確認します。テスト結果の表示の詳細については，次を参照してください。

チェックポイントのパラメータ化

チェックポイントの期待値をパラメータ化できます。これにより，仮想ユーザは複数の反復において異なる期待値を使用できるようになります。単純パラメータおよび XML タイプ・パラメータの両方を定義できます。

期待値をパラメータで置き換えるには，次の手順を実行します。

1 ［**仮想ユーザ**］＞［**パラメータ リスト**］を選択し，［パラメータ リスト］ダイアログ・ボックスを開きます。

2 「**File**」，「**XML**」など適切なタイプを選択して新しいパラメータを作成します。詳細については，130 ページ「パラメータ・タイプについて」を参照してください。

3 ［スクリプト］ビューで，CHECKPOINT セクションを見つけ出し，実際の値を，作成しておいたパラメータ名で置き換えます。

```
BEGIN_CHECKPOINTS,
        CHECKPOINT, "XPATH=EchoMixResult[1]", "Value=MyParam"
END_CHECKPOINTS,
```

チェックポイント結果の確認

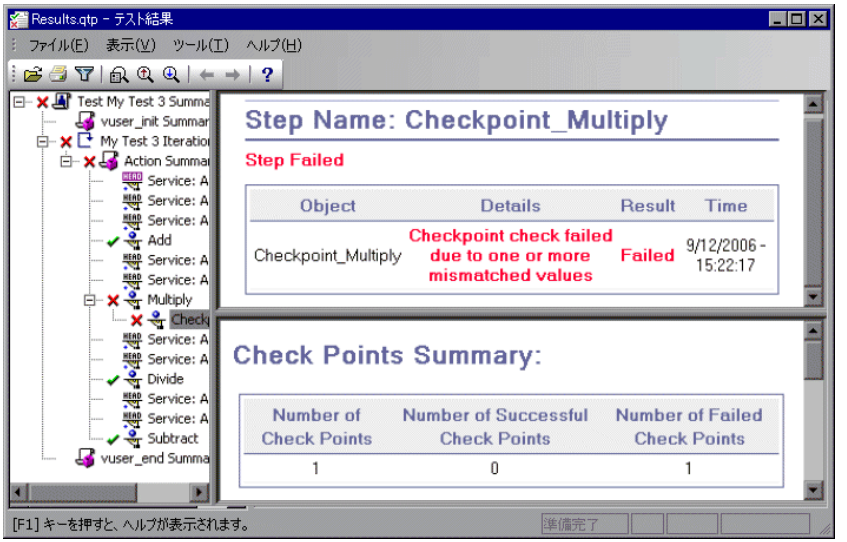
スクリプトの実行後，チェックポイント結果を表示して，ステータス（成功または失敗）および失敗の原因を確認します。

チェックポイントのテスト結果を表示するには，次の手順を実行します。

1 ［**表示**］＞［**テスト結果**］を選択して，［テスト結果］ウィンドウを開きます。

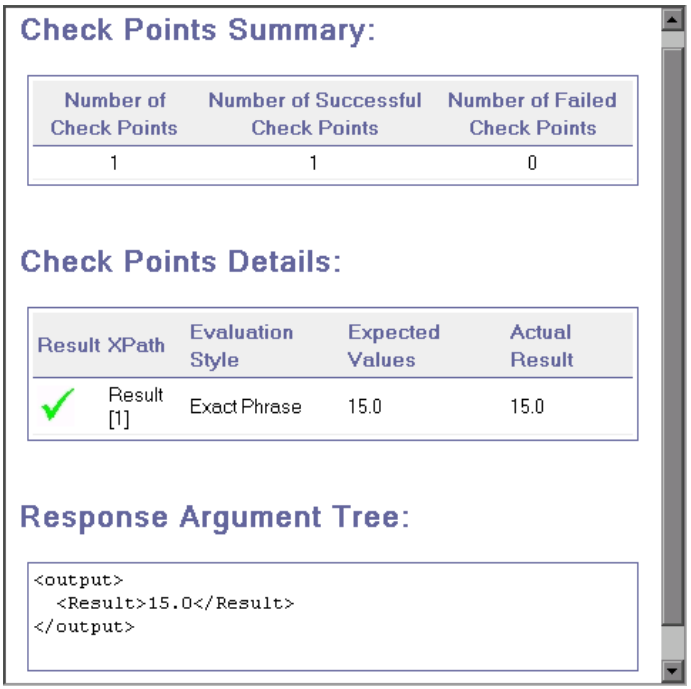
2 左側の表示枠で，表示するチェックポイントのステップを展開します。

3 左側の表示枠でチェックポイントの対象ステップをクリックします。右側の表示枠には，テスト実行の詳細が表示されます。



下側の表示枠には，チェックポイントに関する次の詳細情報が表示されます。

- 成功したチェックポイントおよび失敗したチェックポイントの数（複数の反復について）
- 期待値および実際の結果
- 評価のタイプ（完全一致または正規表現）
- 応答引数ツリー



テスト結果の表示の詳細については，第 16 章「テスト結果の表示」を参照してください。

第 22 章

Web サービス呼び出しプロパティの設定

VuGen を使用して，新規および既存の Web サービス呼び出しのプロパティを設定できます。

- Web サービス呼び出しプロパティの設定について
- Web サービス呼び出しプロパティについて
- トランスポート層の設定
- Web サービス・セキュリティ・ポリシーの作成
- SAML オプションの設定
- Web サービス・スクリプトの動作のカスタマイズ

Web サービス呼び出しプロパティの設定について

VuGen を使用して，スクリプト内の各 Web サービス呼び出しのプロパティを設定できます。新規の Web サービス呼び出しを追加する際にプロパティを設定できます。また，既存の Web サービス呼び出しを変更できます。

引数値，パラメータ化，トランスポート層，およびセキュリティに関してプロパティを設定できます。

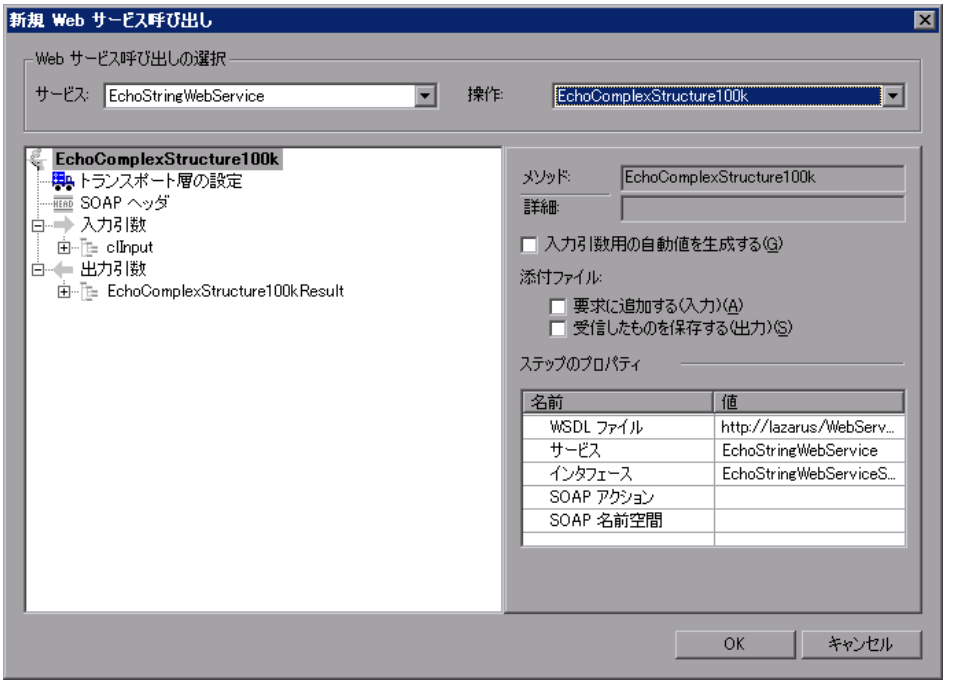
各サービスのメソッドのプロパティを設定して，メソッドをカスタマイズできます。

また，ステップを追加してスクリプトを拡張することで，セキュリティの設定や非同期メッセージのサポートなどを行うことができます。

Web サービス呼び出しプロパティについて

VuGen は，Web サービス呼び出しを手動で追加したり変更したりするためのインタフェースを備えています。

VuGen の［**ステップのプロパティ**］タブには，各サービスの操作がツリー階層に表示されます。ツリーのノードは，トランスポート層の設定，SOAP ヘッダー，入力引数，および出力引数を表します。



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

選択対象 ...	右側の表示枠の表示
メソッド名または操作名	• メソッドのプロパティ • 入力引数の値の自動生成を有効にするためのチェック・ボックス • 添付ファイルを含めるためのチェック・ボックス

［トランスポート層の設定］	詳細トランスポート・オプション： • 非同期ルーティングまたは WS-A ルーティングを使用した **HTTP/S トランスポート** • 応答 / 要求キュー情報を使用した **JMS トランスポート**・サポート
［SOAP ヘッダ］	エディット・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，338 ページ「SOAP ヘッダー」を参照してください。
入力引数ノード	• 各メソッドの［**名前**］とその［**値**］。 • 入力引数の値の自動生成を有効にするためのチェック・ボックス
引数単体	• ［**値**］：引数の値です。 • ［**この引数の自動値を生成**］：このノードの自動値を挿入します。
複雑な入力引数	• ［**XML**］：［**編集**］ボタン，［**インポート**］ボタン，および［**エクスポート**］ボタンを有効にします。XML を編集することで，引数値を手作業で挿入できます。XML 構造全体を 1 つの XML 型パラメータで置き換えるには，［**ABC**］アイコンをクリックします。詳細については，159 ページ「XML パラメータのプロパティの設定」を参照してください。 • ［**この引数の自動値を生成**］：この複合型ノードのすべての引数に対して，自動値を挿入します。 • ［**追加**］/［**削除**］：配列に要素を追加，または配列から要素を削除します。
［出力引数］	［**名前**］引数と，値を格納できる［**パラメータ**］。
［入力添付ファイル］	SOAP 要求をエンコードする［**添付ファイル形式**］：VuGen ツールキットの場合は DIME または MIME，.Net の場合は DIME のみ，Axis の場合は MIME のみです。

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

VuGen では，次の要素の引数値を設定できます。

➤ 入力引数値

➤ 出力引数

➤ 配列

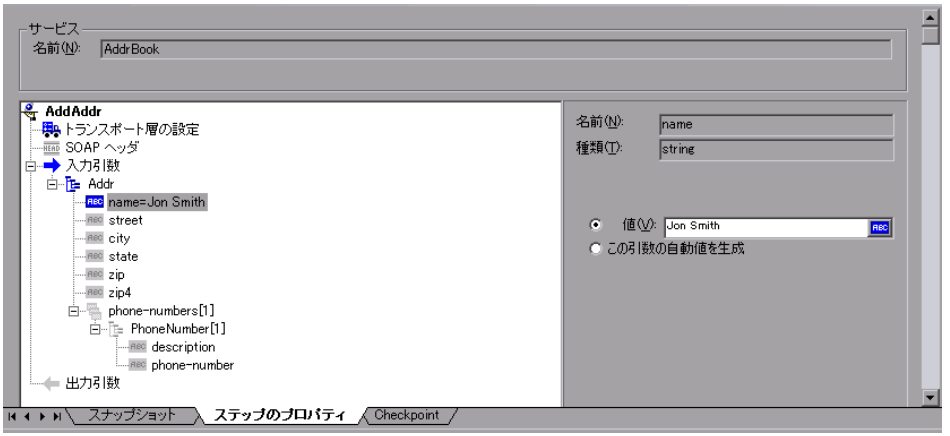
➤ 添付ファイル

➤ SOAP ヘッダー

VuGen では，手作業で編集された引数が青色で表示されます。

入力引数値

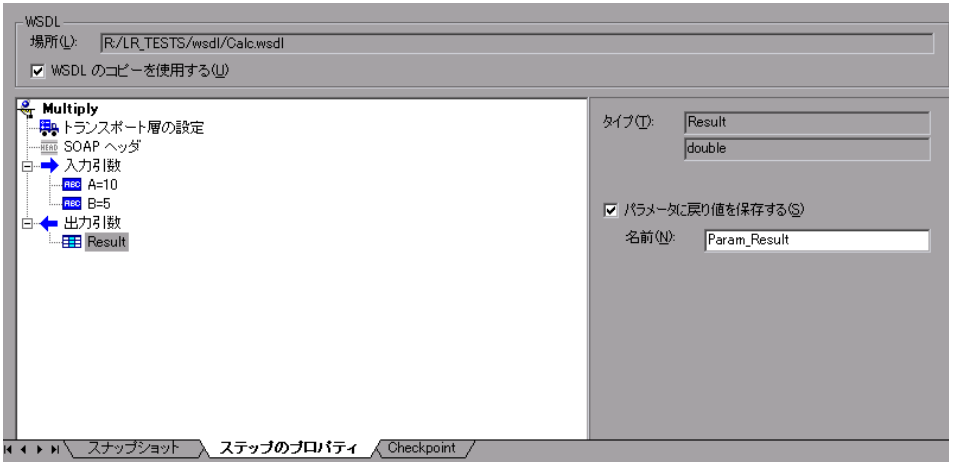
［入力引数］ノードでは，入力引数のすべての値を定義できます。



- ［**値**］：ノードの値を手作業で指定できます。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。
- ［**この引数の自動値を生成**］：この引数の値を VuGen に自動的に生成させる場合は，このオプションを選択します。または，ツリー階層で引数を選択し，右クリック・メニューから［**Generate Auto_Values**］を選択します。

出力引数

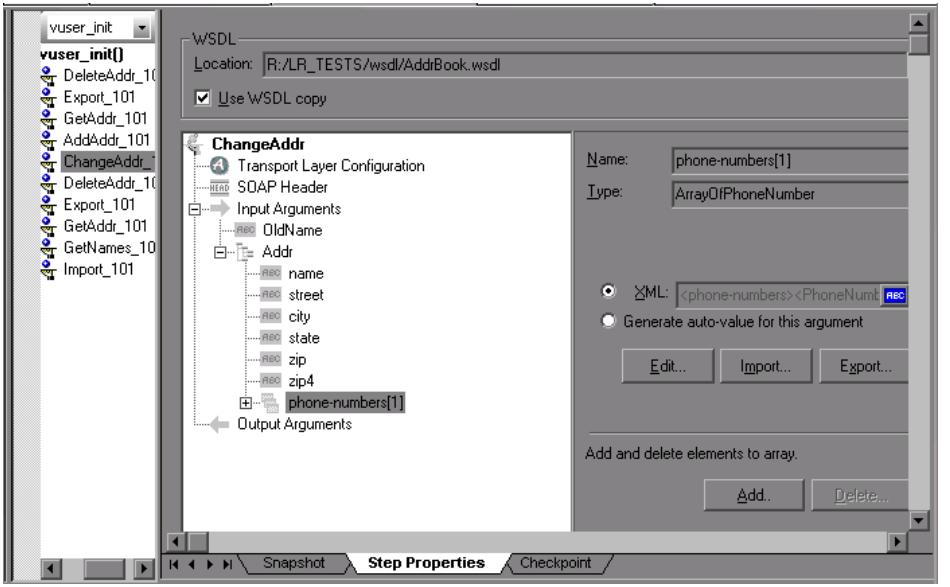
出力引数値を表示したり，出力引数値をパラメータまたは配列に保存したりできます。



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

配列

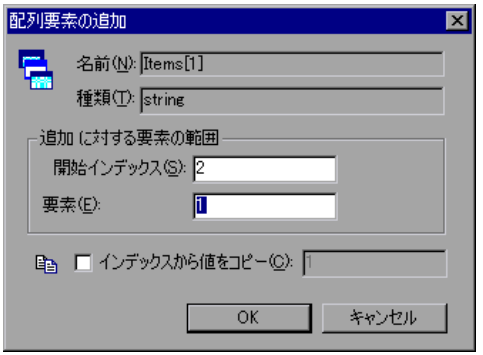
入力引数または出力引数の配列を対象に作業をするには，左側の表示枠でその配列を選択します。



- ［**XML**］：配列要素の値が含まれる XML ファイルのパス。XML を XML 型パラメータで置き換えるには，［**ABC**］アイコンをクリックします。詳細については，159 ページ「XML パラメータのプロパティの設定」を参照してください。
- ［**編集**］/［**インポート**］/［**エクスポート**］：複合型および配列を変更するには，要素および引数を選択して［**編集**］をクリックします。XML をインポートするには，［**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，［**エクスポート**］をクリックします。
- ［**追加**］：［配列要素の追加］ダイアログ・ボックスが開き，単純型または複合型のいずれかの配列要素を追加できます。
- ［**削除**］：配列要素を削除します。削除する要素の開始添字と個数を指定します。

配列要素の追加

［配列を対象に要素を追加、削除］セクションの［**追加**］をクリックすると，次のように［配列要素の追加］ダイアログ・ボックスが開きます。



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

添付ファイル

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

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

SOAP メッセージとともにバイナリ・データを送信するのに使用されるメカニズムは MIME（Multipurpose Internet Mail Extensions）仕様と，より効率的な新しいメカニズムである DIME（Direct Internet Message Encapsulation）仕様です。

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

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

入力添付ファイル

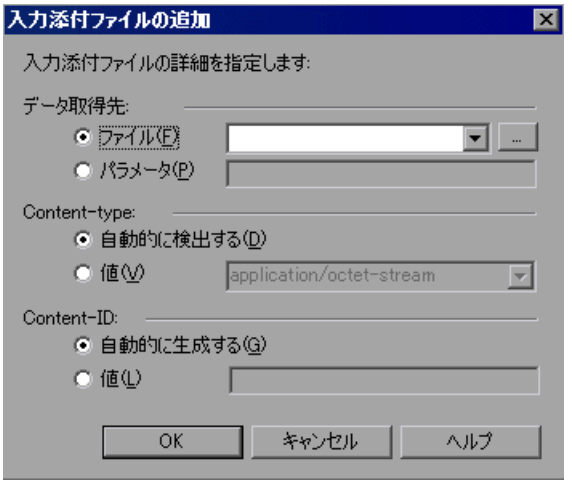
入力添付ファイルが要求メッセージに追加されます。

添付ファイルを要求に追加するには，次の手順を実行します。

1 添付ファイルを追加する対象となる操作を左側の表示枠で選択します。

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

［入力添付ファイルの追加］ダイアログ・ボックスが表示されます。



次の情報を指定します。

➤［**データ取得先**］：データの場所。バイナリ・データが含まれるファイルまたはパラメータを指定できます。

➤［**ファイル**］：次の 2 つの方法でファイルの場所を指定できます。

- 絶対パス：ファイルのフル・パス。このファイルは，スクリプトを実行しているすべてのマシンからアクセス可能でなければなりません。
- 相対パス（推奨）：ファイル名。この方法を使用した場合，VuGen は，再生時にスクリプトのフォルダ内で添付ファイルを検索します。添付ファイルをスクリプトのフォルダに追加するには，［**ファイル**］＞［**スクリプトにファイルを追加**］を選択し，ファイル名を指定します。

➤［**パラメータ**］：データが含まれるパラメータの名前を指定します。

➤［**Content-type**］：データが含まれるファイルのコンテント・タイプ。［**自動的に検出する**］オプションを指定すると，コンテント・タイプが自動的に判断されます。また，［**値**］ボックスの一般的なコンテント・タイプのリストから text/html や image/gif，あるいはほかのコンテント・タイプを選択できます。

［**Content-ID**］：コンテントの ID です。標準設定では，VuGen によって自動的に生成され，添付ファイルの一意の識別子として使用されます。必要に応じて，［**値**］ボックスに別の ID を指定できます。

出力添付ファイル

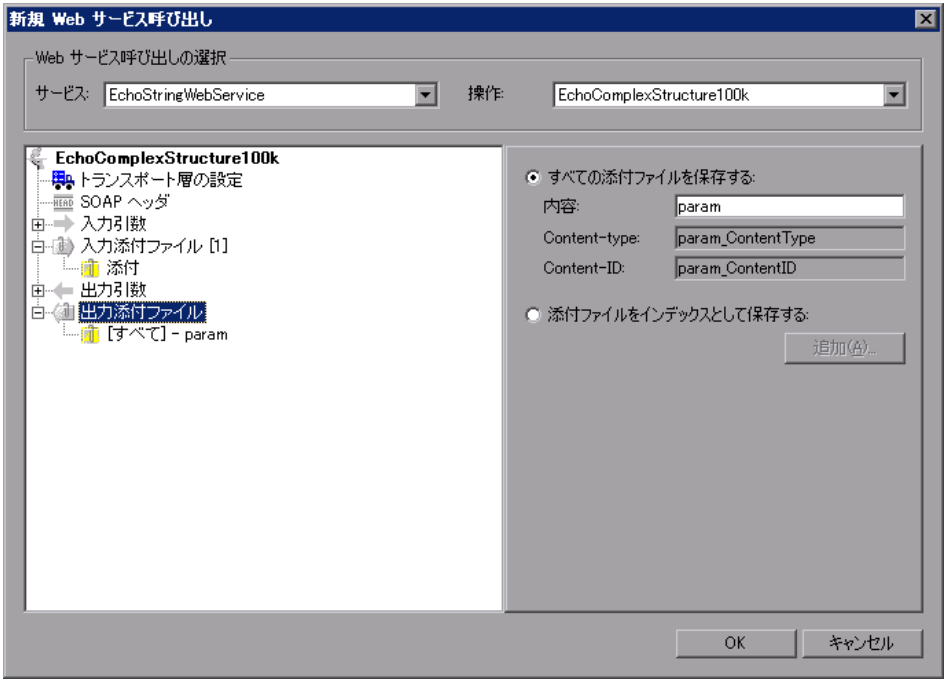
出力添付ファイルが応答メッセージに追加されます。

応答を添付ファイルとして保存するには，次の手順を実行します。

1 応答を保存する対象となる操作を左側の表示枠で選択します。

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

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



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

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

```
MyParam_1
MyParam_1_ContentType
MyParam_1_ContentID
```

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

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

SOAP ヘッダー

このビューは，メソッドのツリー・ビューで「**SOAP ヘッダ**」を選択したときに使用できます。右側の表示枠で，SOAP ヘッダ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，394 ページ「XML ツリーの編集」を参照してください。

トランスポート層の設定

VuGen を使用して，サービスのトランスポート層を設定できます。トランスポート層では，サーバへ / からのメッセージの伝送方法，つまり HTTP/HTTPS または JMS（Java Message Service）を指定できます。詳細については，346 ページ「JMS について」を参照してください。

HTTP/HTTPS トランスポートを使用する場合，非同期呼び出しおよび WS-Addressing を使用できます。

非同期メッセージ

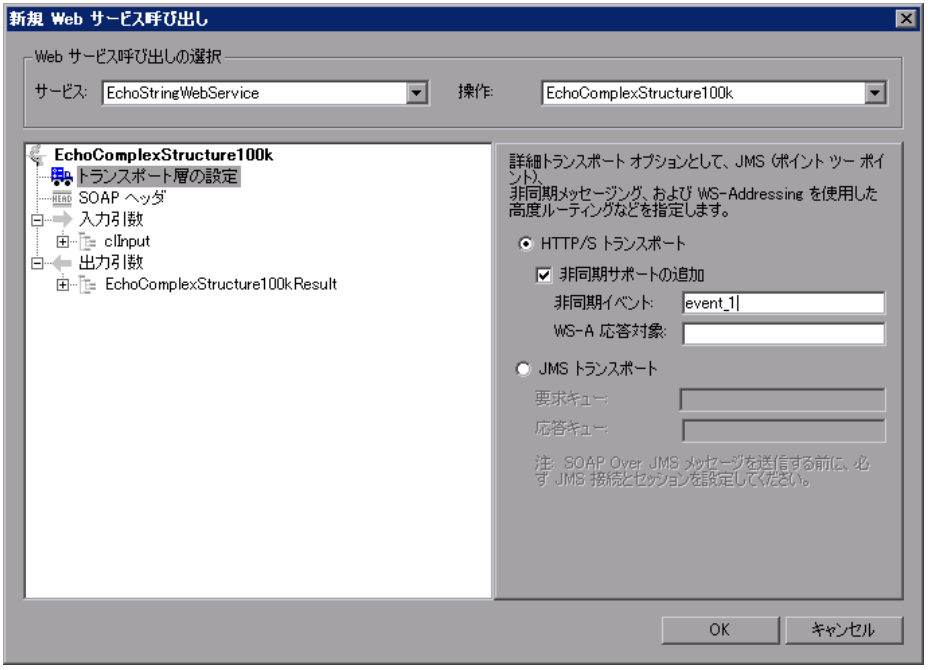
同期メッセージングでは，再生エンジンはサーバが応答を送信するまでスクリプト実行をブロックします。非同期モードでは，再生エンジンは前のメッセージに対するサーバの応答を待機せずにスクリプトを実行します。

VuGen は両方のメッセージングのタイプをサポートしています。

［スクリプト］ビューでは，追加された **AsyncEvent** パラメータを使用して非同期メッセージングが示されます。

```
web_service_call( "StepName=EchoString_101",
        "SOAPMethod=EchoRpcEncoded.EchoSoap.EchoString",
        "ResponseParam=response1",
        "Service=ExtendedECHO_rpc_encoded",
        "AsyncEvent=event_1",
        "Snapshot=t1157371707.inf",
        BEGIN_ARGUMENTS,
        "sec=7",
        "strString=mytext",
        END_ARGUMENTS,
        BEGIN_RESULT,
        "EchoStringResult=first_call",
        END_RESULT,
        LAST);
```

［ツリー］ビューで非同期メッセージングを有効にするには，［**非同期サポートの追加**］オプションを選択してイベントを指定します。



非同期メッセージングを使用している場合，**Web Service Wait for Event（Web サービス イベント待機）**ステップを使用するか，［スクリプト］ビューで **web_service_wait_for_event** 関数を使用して，スクリプト内で同期を実行します。この関数は，前の非同期サービス要求の応答を待機するよう仮想ユーザに指示する関数です。リスナは，サーバが応答するまでサービスの実行をブロックします。

Web Service Wait for Event ステップを追加する場合には，次を実行します。

- 待機する対象となる非同期イベントをすべてリストアップします。
- 次に，仮想ユーザに，すべてのイベントが応答を受信するのを待機させるか，イベントのいずれかが応答を受信するのを待機させるかを指定します。［**ANY**］を指定した場合，再生時にこの関数は応答を受信した最初のイベントの名前を返します。［**ALL**］を指定した場合，2 つ以上のイベントがキャプチャされたときは，キャプチャされたイベントのいずれか 1 つのイベント名が返されます。
- タイムアウト時間（ミリ秒）を指定します。指定したタイムアウト時間内に応答を受信したイベントがなかった場合，**web_service_wait_for_event** は NULL を返します。

次の例では，**web_service_wait_for_event** 呼び出しは，イベント event_1，event_2，または event_3 のいずれかが受け取られるまで 40 ミリ秒間待機します。

```
web_service_wait_for_event("StepName=First_Wait",
        "Quantifier=ANY",
        "Timeout=40",
        BEGIN_EVENTS,
        "event_1",
        "event_2",
        "event_3",
        END_EVENTS,
        LAST);
```

非同期メッセージングを指定してスクリプトを実行すると，再生ログにはイベントおよび入力引数 / 出力引数に関する情報が記録されます。

web_service_wait_for_event 関数の詳細については，「**オンライン関数リファレンス**」（［**ヘルプ**］＞［**関数リファレンス**］を選択するか，関数の上で **F1** キーを押します）を参照してください。

また，WS-Addressing を使用して，イベントを検出した場合のサービスの応答先を指定することもできます。詳細については，次項を参照してください。

WS-Addressing

WS-Addressing は，Web サービスがアドレス指定情報を伝えるのを可能にする標準を定義する規格です。この規格では，メッセージにおけるエンド・ツー・エンドのエンドポイントの識別を確実にするために，Web サービス・エンドポイントを識別します。これにより，エンドポイント・マネージャ，ファイアウォール，およびゲートウェイなど追加の処理ノードを持つネットワークを経由してメッセージを送信できます。WS-Addressing は，同期トランスポートおよび非同期トランスポートの両方を伝送される Web サービス・メッセージをサポートしています。

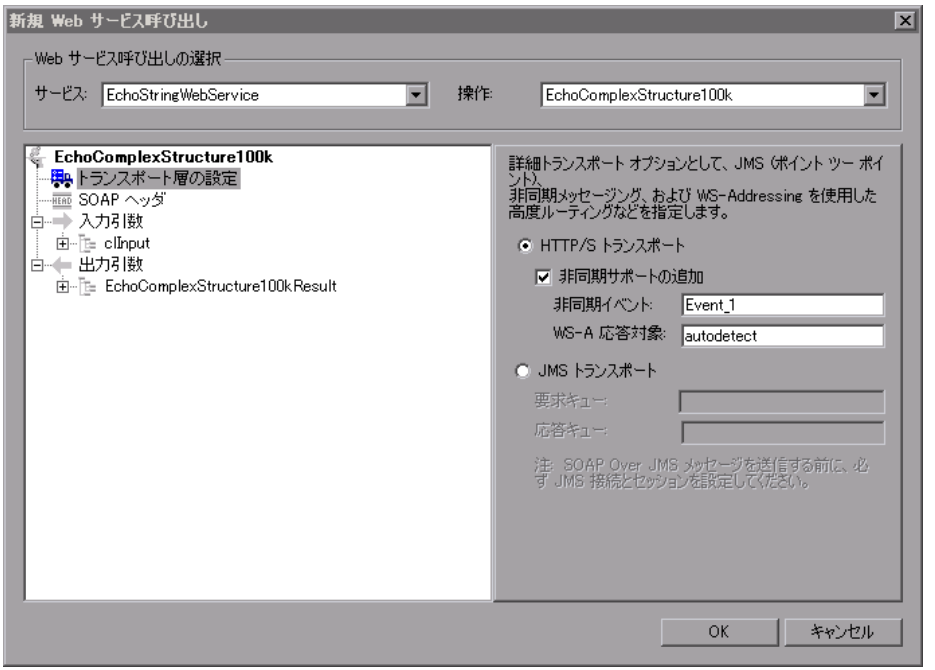
WS-Addressing を実装するには，サービスの応答先となる **WSAReplyTo** アドレスを指定します。

オプションの **WSAAction** 引数を使用すれば，自動検出が失敗した場合の SOAP アクションを定義できます。

次の例は，WS-Addressing を使用した一般的な SOAP メッセージを示します。

```
<S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
            xmlns:wsa="http://www.w3.org/2004/12/addressing">
   <S:Header>
      <wsa:MessageID>
         http://example.com/SomeUniqueMessageIdString
      </wsa:MessageID>
      <wsa:ReplyTo>
         <wsa:Address>http://myClient.example/someClientUser</wsa:Address>
      </wsa:ReplyTo>
<wsa:Address>http://myserver.example/DemoErrorHandler</wsa:Address>
      </wsa:FaultTo>
      <wsa:To>http://myserver.example/DemoServiceURI</wsa:To>
      <wsa:Action>http://myserver.example/DoAction</wsa:Action>
   </S:Header>
   <S:Body>
      <!-- Body of SOAP request message -->
   </S:Body>
</S:Envelope>
```

VuGen において，WS-Addressing を使用して SOAP 要求を作成するには，［**ステップのプロパティ**］タブの［トランスポート層の設定］ノードで **WSAReplyTo（WS-A 応答対象）**エントリを指定します。



WSAReplyTo（WS-A 応答対象）引数に対しては，IP アドレスを指定するか，マシンのホスト名を検出するようにサービスに指示するために **autotdetect** を指定できます。これは，複数の異なるマシンで同じスクリプトを実行する場合に便利です。

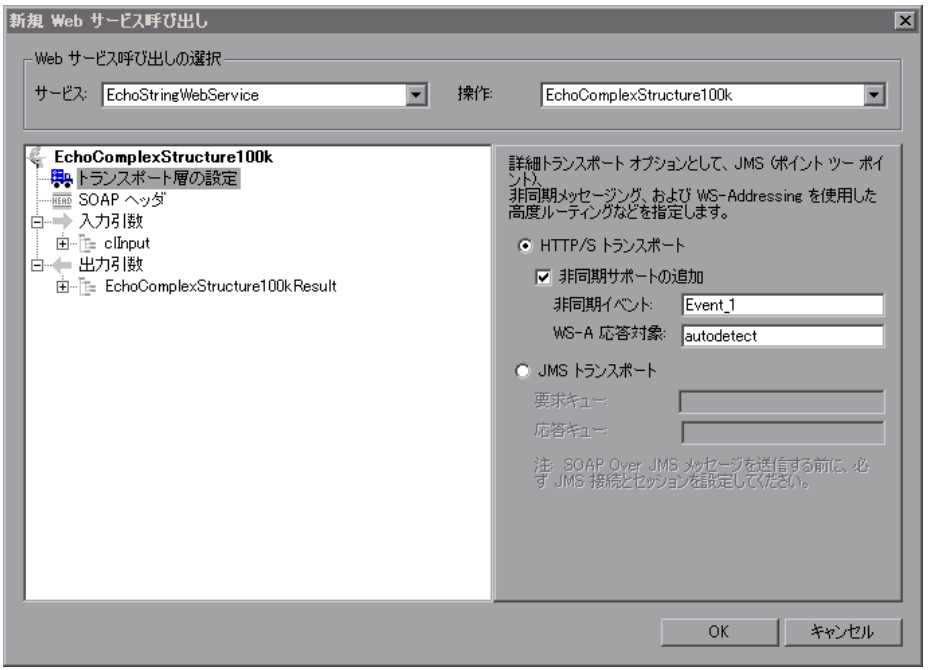
次の例では，サーバは **Event_1** を検出した場合，インタフェース **212.199.95.138** に応答します。

```
web_service_call( "StepName=Add_101",
        "SOAPMethod=Calc.CalcSoap.Add",
        "ResponseParam=response",
        "AsyncEvent=Event_1",
          "WSAReplyTo=212.199.95.138",
        "WSDL=http://lazarus/WebServices/CalcWS/Calc.asmx?wsdl",
        "UseWSDLCopy=1",
        "Snapshot=t1153825715.inf",
        BEGIN_ARGUMENTS,
           "first=1",
           "second=2",
        END_ARGUMENTS,
        BEGIN_RESULT,
           "AddResult=Param_AddResult1",
        END_RESULT,
        LAST);
```

WS-Addressing 呼び出しは，非同期および同期のどちらのモードでも発行できます。同期モードで WS-Addressing を使用するには，［トランスポート層］オプションの［**非同期イベント**］ボックスを空にしておきます。［スクリプト］ビューでは，**AsyncEvent** 引数を削除します。こうすることで，再生エンジンは完全な応答がサーバから受信されるまでスクリプトの実行をブロックします。

非同期メッセージおよび WS-Addressing のサポートを追加するには，次の手順を実行します。

1 新規の Web サービス呼び出しの場合，［**トランスポート層の設定**］ノードを選択します。既存の Web サービス呼び出しの場合，［ツリー］ビューでステップを選択し，［**ステップのプロパティ**］タブを選択します。［**トランスポート層の設定**］ノードを選択します。

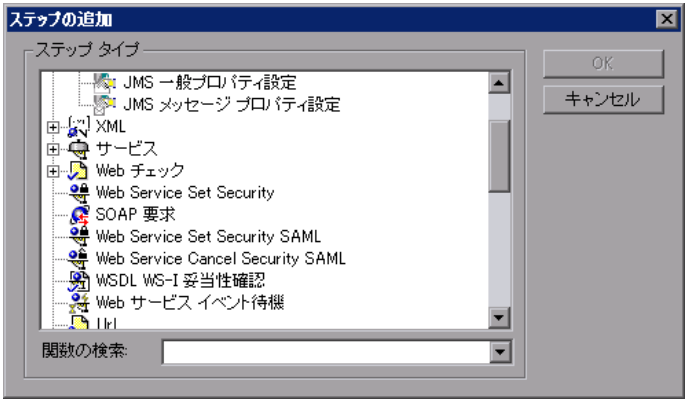


2 Web サービス呼び出しを非同期メッセージとしてマークするには，次の手順を実行します。

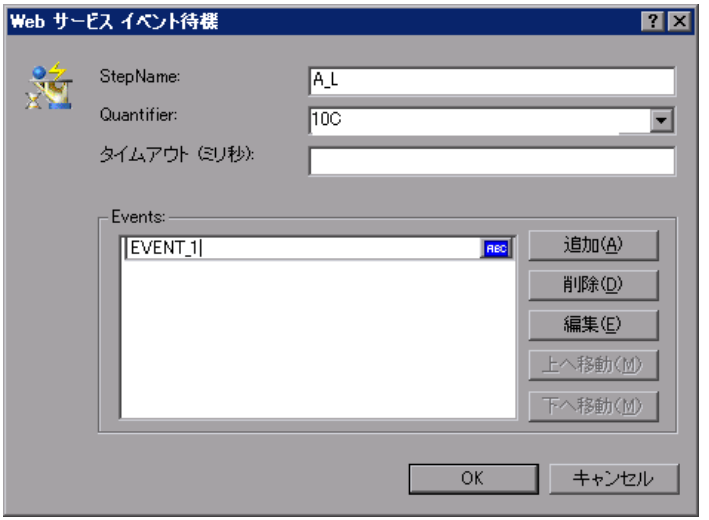
- ［**非同期サポートの追加**］オプションを選択します。これは，HTTP/S タイプのトランスポートの場合にのみ有効になります。
- ［**非同期イベント**］ボックスにイベント名を指定します。任意の名前を指定できます。

3 ［**WS-A 応答対象**］ボックスに IP アドレスを入力します。または，現在のホストを使用する場合は **autodetect** を入力します。これにより，イベントの発生時にサーバは指定された場所に応答します。

4 非同期イベントの場合，［**挿入**］＞［**新規ステップ**］を選択し，Web サービス呼び出しステップの後に **Web Service Wait For Event（Web サービス イベント待機）**ステップを追加します。



5 数量子，タイムアウト，およびイベント名を指定します。これにより，Web サービスは，指定されたイベントを待機してから応答します。



JMS について

JMS は，メッセージ・キューをメッセージのターゲットとして定義することで，ポイントツーポイントのメッセージングを実装します。複数の送信者が 1 つのメッセージ・キューにメッセージを送信し，受信者はそのキューからメッセージを取得します。VuGen は，JMS スクリプト関数を使用して，Web サービス呼び出しからキューへの JMS メッセージの送受信をサポートしています。

JMS トランスポートを使用してメッセージを送信するには，トランスポートを記述する次の項目を設定する必要があります。

- ［**JNDI 初期コンテキスト ファクトリ**］: JMS 接続ファクトリや JMS キューなどの JMS リソースの検索に使用する初期コンテキストを作成するファクトリ・クラスのクラス名。
- ［**JNDI プロバイダ**］: JMS 接続ファクトリや JMS キューなどの JMS リソースの検索に使用するサービス・プロバイダの URL。
- ［**JMS 接続ファクトリ**］: JMS 接続ファクトリの JNDI 名です。

また，VuGen では，受信メッセージに対するタイムアウトおよびプロセスごとの JMS 接続数を設定できます。

これらの設定はすべて JMS 実行環境の設定を使用して設定できます。詳細については，387 ページ「Web サービス JMS の実行環境の設定」を参照してください。

JMS スクリプティング関数

VuGen は，次の関数を使用して JMS トランスポートを実装します。

関数名	説明
jms_receive_message_queue	キューからメッセージを受信します。
jms_send_message_queue	メッセージをキューに送信します。
jms_send_receive_message_queue	指定されたキューにメッセージを送信し，指定されたキューからメッセージを受信します。

jms_set_general_property	次に送信されるメッセージの一般 JMS プロパティを設定します。
jms_set_message_property	次に送信されるメッセージの JMS ヘッダーまたはプロパティを設定します。テキスト・メッセージまたはバイト・メッセージのいずれかを送信できます。 JMSSelector，JMSCorelationID などのヘッダーを追加して，メッセージ・トランスポートをカスタマイズできます。

これらの関数の詳細については，「**オンライン関数リファレンス**」（[**ヘルプ**] ＞［**関数リファレンス**］を選択するか，関数の上で **F1** キーを押します）を参照してください。

トランスポート層としての JMS の使用

Web サービスに対して JMS トランスポートを使用すると，ユーザは一般の HTTP トランスポートではなく JMS トランスポートを使用して SOAP メッセージを送信できます。

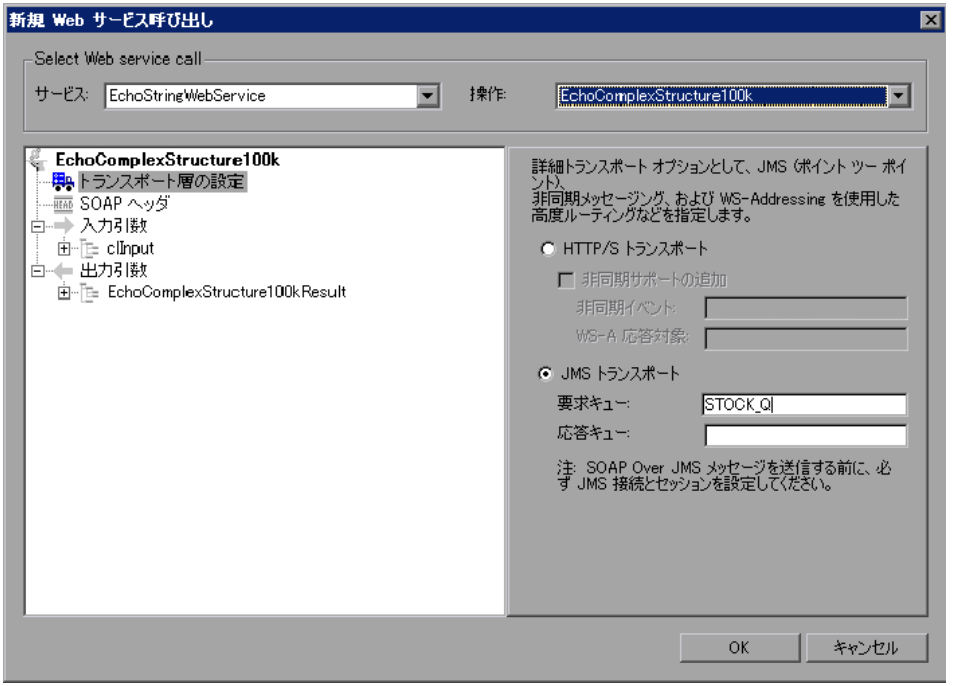
Web サービスは次のことが可能です。

- WSDL をインポートし，トランスポート層として HTTP ではなく JMS を指定する。
- HTTP を使用して SOAP メッセージを記録し，その記録済みスクリプトを使用して JMS トランスポートを使用経由でメッセージを送信する。
- Web サービス・スクリプトと JMS の送受信先（キュー）との間で一般 JMS メッセージを送受信する。

VuGen において JMS プロトコルを使用してメッセージを送受信するための基本手順は，次のとおりです。

1. WSDL ファイルをインポートします。詳細については，369 ページ「サービスのインポート」を参照してください。
2. 後述の手順に従ってキュー情報を入力します。
3. スクリプトを実行する前に JMS 実行環境の設定を設定します。詳細については，387 ページ「Web サービス JMS の実行環境の設定」を参照してください。

各 Web サービス呼び出しステップに対して，JMS トランスポート，要求キュー，および応答キューを指定できます。



JMS トランスポート情報を指定するには，次の手順を実行します。

1 ［ツリー］ビューで，［**ステップのプロパティ**］タブを選択し，［**トランスポート層の設定**］ノードを選択します。

2 ［**JMS トランスポート**］オプションを選択します。

3 ［**要求キュー**］および［**応答キュー**］の名前を指定します。要求および応答には同じキューを使用できます。

4 手動で追加のトランスポート情報を設定したり，プロパティの取得および設定をしたりするには，JMS 関数の 1 つを追加します。［**挿入**］＞［**新規ステップ**］を選択して，［**JMS 関数**］（**JMS Functions**）ノードを展開します。

現在のソリューションは再生専用のソリューションのため，クライアントとサーバの間で送受信される JMS メッセージは記録できません。また VuGen は同期呼び出しのみサポートします。

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**］を選択するか，関数の上で **F1** キーを押します）を参照してください。

Web サービス・セキュリティ・ポリシーの作成

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

メッセージの安全な送信を可能にするため，VuGen では，セキュリティ・トークン，SAML，および JMS など複数のセキュリティ・メカニズムをサポートしています。

トークンの詳細については，次を参照してください。

SAML の詳細については，356 ページ「SAML オプションの設定」を参照してください。

JMS の詳細については，338 ページ「トランスポート層の設定」を参照してください。

セキュリティ・トークンおよび暗号化

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

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

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

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

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

使用可能なトークンは，**Username and Password**，**X.509 Certificate**，**Kerberos Ticket**，**Kerberos2 Ticket**，**Security Context Token**，および **Derived Token** です。指定する必要がある情報は，トークンに応じて異なります。

- **User Name and Password**：**User Name and Password** トークンには，認証のためのユーザ識別情報（**User Name** および **Password**）が含まれます。

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

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

  仮想ユーザ・スクリプトに X.509 トークンを追加する場合は，**Logical Name**，**Store Name**，**Key identifier type**，**Key identifier value**，および **Store Location** 引数を指定します。

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

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

  仮想ユーザ・スクリプトに Kerberos トークンを追加する場合は，トークンの **Logical Name**，および Web サービス・マシンの **Host** 名および **Domain** 名を指定します。

- **Security Context Token**：このトークンは，有効期限が切れるまで繰り返し使用できるセキュリティ・トークンです。SOAP メッセージの送信者は，セキュリティ・コンテキスト・トークンを使用して，SOAP メッセージの送信者とターゲット Web サービスの間の一連の SOAP メッセージ（会話と呼ばれる）に署名をしたり，このメッセージを暗号化したりできます。このタイプのトークンの主な利点は次のとおりです。

  - セキュリティ・コンテキスト・トークンが期限切れになっていない限り，SOAP メッセージの送信者は，同じセキュリティ・コンテキスト・トークンを使用して，ターゲット Web サービスに送信する SOAP メッセージに署名をしたり，このメッセージを暗号化したりできます。
  - セキュリティ・コンテキスト・トークンは，対称鍵に基づいています。これは，SOAP メッセージのデジタル署名または暗号化において，非対称鍵以上にセキュリティ・コンテキスト・トークンを有効なものにします。

- セキュリティ・コンテキスト・トークンは，SOAP メッセージを送信することによって，あるセキュリティ・トークン・サービスから別のセキュリティ・トークン・サービスに要求できます。

仮想ユーザ・スクリプトに **Security Context** トークンを追加する場合は，**Logical Name**，**Base Token**，**Issuer Token**，**End Point URI**，および **Add applies to** 引数の値を指定します。

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

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

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

Token Type，**Logical Name**，**Base Token**，**Issuer Token**，**Derive From** 引数では，パラメータ化はサポートされません。

メッセージ署名および暗号化されたデータを使った作業

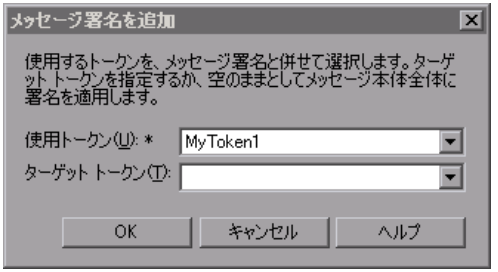
SOAP メッセージにセキュリティ・トークンを追加すると，セキュリティ・トークンは，XML 要素の形式で SOAP メッセージの WS-Security SOAP ヘッダーに追加されます。

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

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

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

VuGen では，暗号化およびメッセージ署名に関する情報を指定できます。



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

Web サービス・セキュリティの設定

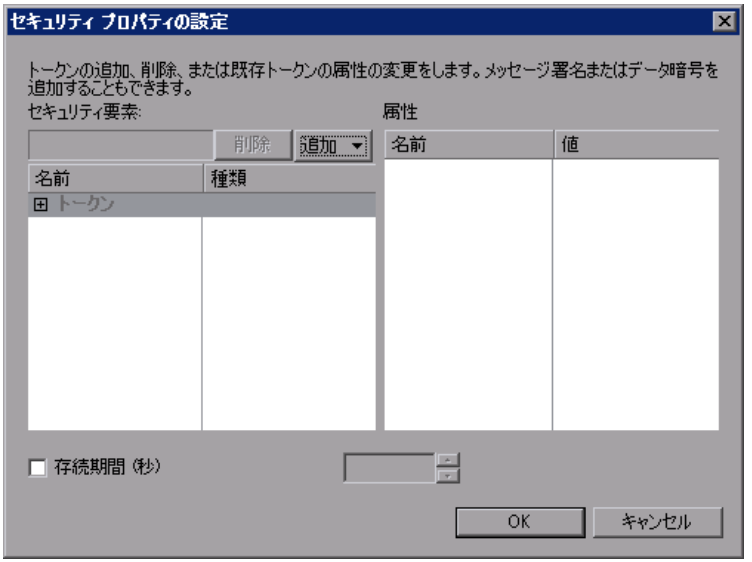
スクリプトを作成する際に，標準 WS-Security を使用して Web サービスにセキュリティを追加できます。

Web サービス・セキュリティを追加するには，次の手順を実行します。

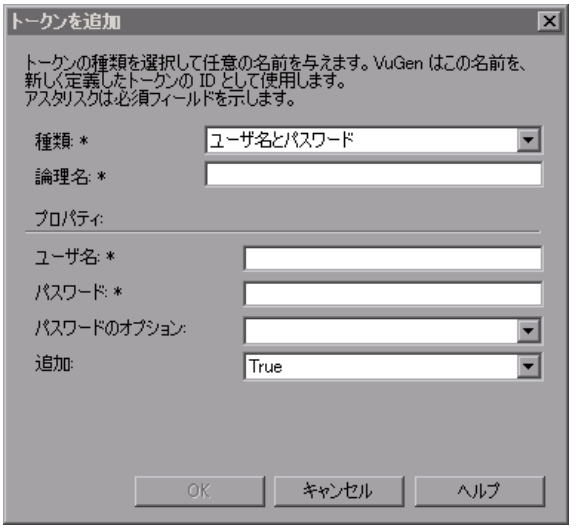
1 スクリプトの適切な場所をクリックします。スクリプト全体にセキュリティを適用するには，スクリプトの先頭にカーソルを置きます。

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

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



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



5 トークンのタイプを選択します。トークンのタイプの詳細については，350 ページ「セキュリティ・トークンおよび暗号化」を参照してください。

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

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

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

6 メッセージ・パケットの有効期間を指定するには，［**存続期間**］を選択して，時間（秒）を指定します。これは，メッセージのルーティングに障害があってネットワークを遮断している場合や，メッセージのルーティングが無限ループに陥っている場合に役立ちます。

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

SAML オプションの設定

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

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

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

注：SAML セキュリティおよび標準 Web サービス（**Web Service Set Security** ステップ）・セキュリティを同じステップに適用することはできません。Web サービス・セキュリティを取り消すには，**web_service_cancel_security** 関数を使用します。

ポリシー・ファイル

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

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

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

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

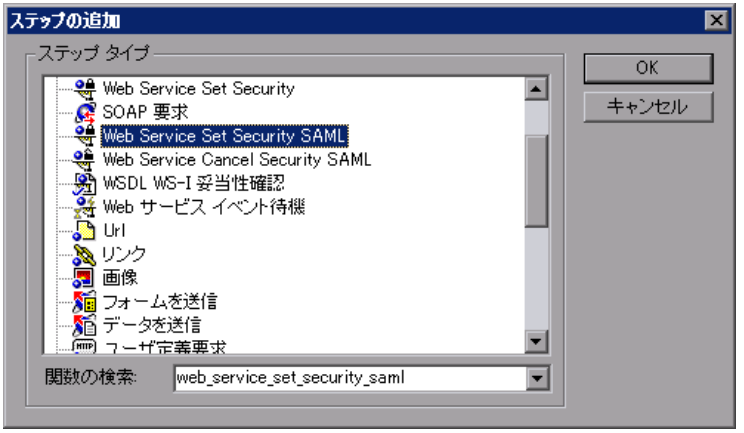
SAML ポリシー・ファイルの詳細については，MSDN Web サイトの SAML STS の例を参照してください。SAML Federation の動作をエミュレートする場合は，**samlFederationPolicy.config** ファイルをデータ・フォルダからスクリプト・フォルダにコピーして，これをポリシー・ファイルとして指定します。

SAML セキュリティを使用するには，マシンに WSE 3.0 が存在することを確認します。

SAML セキュリティを追加するには，次の手順を実行します。

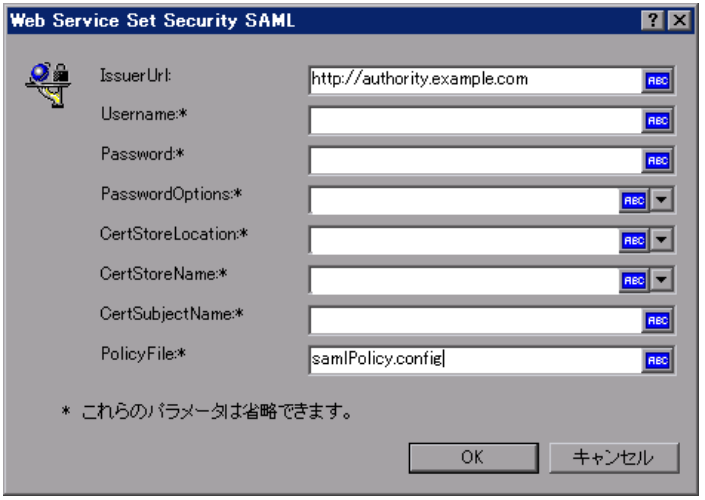
1 スクリプトの適切な場所をクリックします。スクリプト全体にセキュリティを適用するには，スクリプトの先頭にカーソルを置きます。

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



3 SAML セキュリティを追加するには，［**Web Services Set Security SAML**］を選択します。

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



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

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

これらの関数の詳細については，「**オンライン関数リファレンス**」（［**ヘルプ**］＞［**関数リファレンス**］を選択するか，関数の上で **F1** キーを押します）を参照してください。

Web サービス・スクリプトの動作のカスタマイズ

VuGen には，スクリプトの動作方法をカスタマイズできる高度な機能がいくつか用意されています。その機能は次のとおりです。

- ユーザ・ハンドラ
- .NET フィルタ
- 設定ファイル

ユーザ・ハンドラ

VuGen は，Web サービス・スクリプトにユーザ・ハンドラ API を提供します。ユーザ・ハンドラでは，DLL ファイルを介してユーザ定義ハンドラを定義することによって，スクリプトに機能を追加できます。ハンドラは，SOAP の要求および応答を処理するために関数をエクスポートします。これらの要求および応答では，SOAP エンベロープの取得，新しい SOAP エンベロープの設定，パラメータの取得または設定，ログ・メッセージの発行を行うことができます。

このメカニズムは，セキュリティ機能，メッセージ圧縮，フィルタの追加に使用できます。

＜LoadRunner＞¥include フォルダにある API ヘッダー・ファイル **LrWsHandlerAPI.h** には，多くのインライン・コメントおよび説明が含まれています。

VuGen には，ハンドラ作成のテンプレートとして使用できるサンプルの Visual Studio プロジェクトが用意されています。このサンプルは，要求および応答のエンベロープを取得し，それをパラメータに保存します。このサンプルは，**＜LoadRunner＞¥samples¥WebServices¥SampleWsHandler** フォルダにあります。このサンプルを使用するには，Visual Studio でサンプルを開き，必要な機能を追加します。要求 / 応答をパラメータに保存する必要がなければ，サンプルのそのセクションを削除できます。

サンプルを編集したら，サンプルを保存し，DLL をコンパイルします。プロジェクトをコンパイルすると，Visual Studio によって **＜LoadRunner＞¥bin** フォルダに **<user_handler_name>.DLL** ファイルが置かれます。

ユーザ・ハンドラの実装

ユーザ・ハンドラは，グローバルに，またはローカルに実装できます。

グローバルに，すなわちスクリプトのすべての要求でハンドラを使用するには，スクリプトのフォルダにある **default.cfg** ファイルに次のセクションを追加します。

```
[UserHandler]
Name=<name1>
Args=<args1>
Order=<BeforeSecurity/AfterSecurity/AfterAttachments>
```

- **Name**：ユーザ・ハンドラおよび DLL の名前。
- **Args**：ハンドラの設定引数のリスト。ハンドラの引数を取得するには，**GetArguments** メソッドを使用します。
- **Order**：仮想ユーザが要求のユーザ・ハンドラを処理する順序（**Before Security**（セキュリティの前），**After Security**（セキュリティの後），**After Attachments**（添付ファイルの後））。標準設定では，ユーザ・ハンドラはセキュリティの前に処理されます。

  要求メッセージの場合，仮想ユーザは，セキュリティ・ハンドラの**後で**添付ファイル・ハンドラを処理します。応答メッセージの場合，仮想ユーザは逆の順序でハンドラを処理します。

ローカルに，すなわち特定の要求でハンドラを使用するには，**web_service_call** 関数に次の引数を追加します。

```
UserHandlerName=<name1>
UserHandlerArgs=<args1>
UserHandlerOrder=<BeforeSecurity/AfterSecurity/AfterAttachments>
```

スクリプトを別のマシンにコピーしても，ハンドラ情報は保持されます。これは，ハンドラ情報がスクリプトのフォルダに定義されているためです。スクリプトの特定のステップでローカルに定義されているユーザ・ハンドラは，グローバルなハンドラ設定（スクリプトの **default.cfg** ファイルで定義）に優先します。

注：ユーザ・ハンドラ DLL は，それを呼び出すスクリプトを実行しているすべてのロード・ジェネレータ・マシンからアクセスできなければなりません。ユーザ・ハンドラ DLL は，たとえば ＜ LoadRunner ＞ ¥bin フォルダにコピーできます。

プロパティの取得および設定

ユーザ・ハンドラは，ハンドラおよびエンベロープのプロパティを取得，設定するための関数をいくつか提供します。プロパティを取得するには，Get 関数群を使用します。また，再生フレームワークからハンドラへ，あるいはハンドラ間で情報を渡すには，Set 関数群を使用します。

- **GetEnvelope**：エンベロープの内容を取得します。
- **GetEnvelopeLength**：エンベロープの長さを取得します。
- **SetEnvelope**：エンベロープの内容と長さを設定します。
- **SetContentType**：HTTP ヘッダーのコンテント・タイプに新しい値を設定します。
- **LogMessage**：再生ログへメッセージを発行します。
- **GetArguments**：DLL に渡すため，現在のハンドラに定義されている設定引数を取得します。
- **GetProperty**：ユーザ定義のプロパティ値を取得します。
- **SetProperty**：ユーザ定義のプロパティ値を設定します。

詳細については，＜ LoadRunner ＞ ¥include フォルダにある **LrWsHandlerAPI.h** ファイルのコメントを参照してください。

.NET フィルタ

メッセージに適用する .NET フィルタがある場合は，ハンドラ・メカニズムを使用して実装できます。

スクリプトの **default.cfg** ファイルに次の行を追加して，スクリプト全体にグローバルにフィルタを定義します。

```
[UserHandler]
Name=LrWsSoapFilterLoader
Args=<Filters InputFilterClass="class name" InputFilterLib="lib name" Out-
putFilterClass="class name" OutputFilterLib="lib name" />
Order=BeforeSecurity/AfterSecurity/AfterAttachments
```

web_service_call 関数に次の引数を追加して，特定のステップにフィルタを定義します。

```
UserHandlerName= LrWsSoapFilterLoader

UserHandlerArgs=<Filters InputFilterClass=¥"class name¥" InputFilter-
Lib=¥"lib name¥" OutputFilterClass=¥"class name¥" OutputFilterLib=¥"lib
name¥" />

UserHandlerOrder=BeforeSecurity/AfterSecurity/AfterAttachments
```

設定ファイル

＜ LoadRunner ＞ ¥bin フォルダにある **mmdrv.exe.config** ファイルには，スクリプトの動作に関する情報，およびその他の設定情報が含まれています。

アプリケーションに独自の設定ファイル **app.config** がある場合，このファイルをいくつかの方法で実装できます。

- app.config を **mmdrv.exe.config** という名前で保存して，既存の設定ファイルを上書きします。これにより，マシン上のすべてのスクリプトに設定情報が適用されます。
- **app.config** をスクリプトのフォルダに保存します。**app.config** の設定は **mmdrv.exe.config** の設定に優先します。また，app.config をスクリプトのフォルダに保存すれば，常にスクリプトに関連付けられることになり，ほかのマシンに個別にコピーする必要がなくなります。

  さらに，設定ファイルには，セキュリティ情報も含まれます。テスト用の無署名の証明書を許可するかどうか設定できます。

標準設定では，テストを円滑にするために無署名の証明書が使用できます。無署名の証明書を不許可にするには，**mmdrv.exe.config** ファイルの **allowTestRoot** フラグを **false** に変更します。

```
<security>
  <x509 storeLocation="currentuser" alllowTestRoot="false"
```

第 23 章

Web サービスの管理

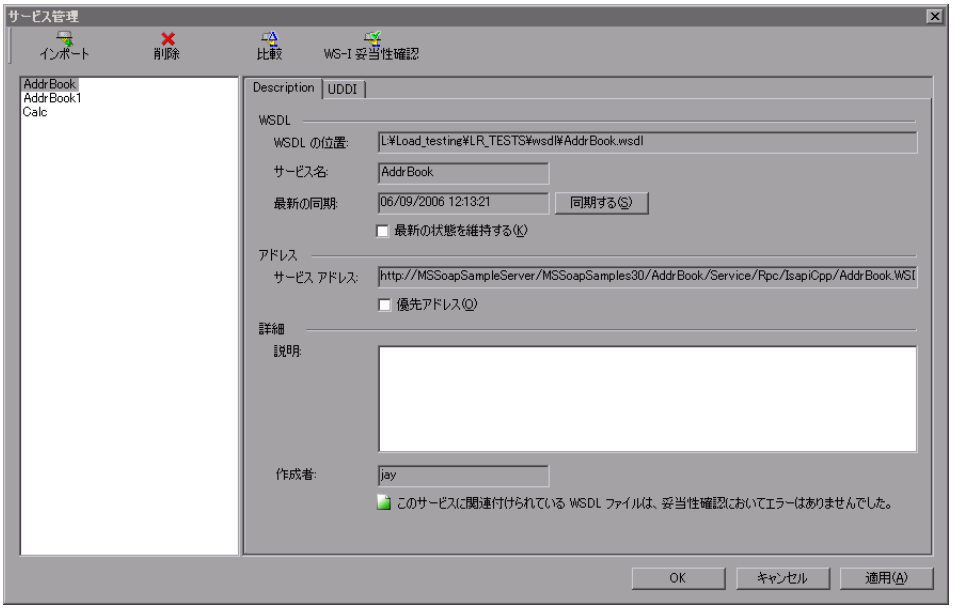
VuGen には，サービス・エントリに関連付けられている WSDL ファイルを検証，管理するユーティリティがあります。

本章では，次の項目について説明します。

- Web サービス仮想ユーザ・スクリプトの管理について
- サービス・プロパティの表示と設定
- サービスのインポート
- WSDL ファイルの検証
- WSDL ファイルおよび XML ファイルの比較

Web サービス仮想ユーザ・スクリプトの管理について

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



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

注：WSDL ファイルに対する検証と変更は，すべて作業用コピーを対象に実行されます。インポートされた WSDL ファイルを新しいバージョンに置き換える場合は，次の記述に従って，［**同期化**］オプションを使ってファイルを更新します。

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

［サービス管理］ウィンドウには次のインタフェースが用意されています。

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

サービス・プロパティの表示と設定

［サービス管理］ウィンドウでは，サービス・エントリの説明と UDDI 情報を表示できます。サービスのリストからサービスを選択すると，［サービス管理］ウィンドウの右側の表示枠にその詳細が表示されます。

サービスに関する注釈またはメモを追加するには，詳細説明を入力します。

説明

［サービス管理］ウィンドウの［**Description**］タブには，サービスに関する情報（［WSDL］，［アドレス］，［詳細］）が表示されます。

WSDL

- ［**WSDL の位置**］：WSDL ファイルの元のソース（読み取り専用）。
- ［**サービス名**］：Web サービスの名前（読み取り専用）。
- ［**最新の同期**］：ローカル・コピーを元のソースの WSDL ファイルで更新することにより同期化した最新の日付（読み取り専用）。
  - WSDL の作業コピーを手作業で更新するには，［**同期する**］をクリックします。
  - スクリプトを開くときはいつも WSDL を同期するよう VuGen に指示するには，［**最新の状態を維持する**］を選択します。

アドレス

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

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

これはサービスのエミュレーションを実装する場合に便利です。詳細については，292 ページ「仮想ユーザ・スクリプトでのエミュレートされたサービスの使用」を参照してください。

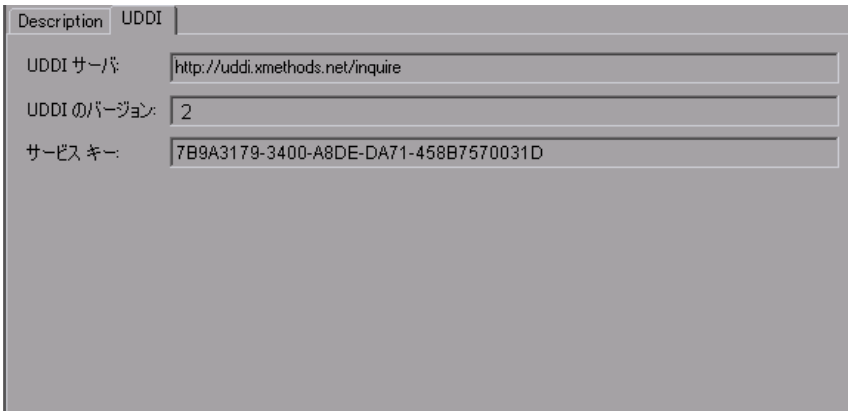
詳細

➤ [**説明**]：標準の設定では WSDL ファイルから取得される Web サービスの説明です。このテキスト領域は編集可能です。

➤ [**作成者**]：最初にサービスをインポートしたユーザの名前です（読み取り専用）。

UDDI 情報

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



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

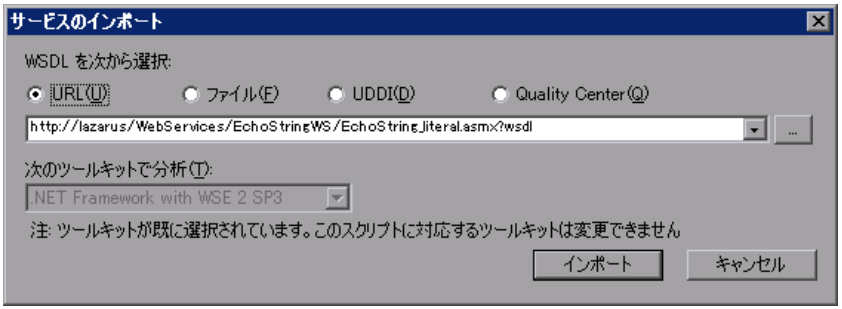
サービスのインポート

VuGen では，Web サービス呼び出しステップを使用する高レベルのテストを作成するためにサービスをインポートできます。通常は，WSDL ファイルをインポートすることでスクリプトの作成を開始します。スクリプトを記録する場合や，サーバ・トラフィックに基づいてスクリプトを作成する場合，WSDL のインポートは任意です。サービスを指定しない場合，VuGen では，SOAP トラフィックが解析され，Web サービス呼び出しの代わりに SOAP 要求を使用してより低いレベルのスクリプトが作成されます。また，［**挿入**］＞［**新規ステップ**］コマンドを使用して，SOAP 要求を手作業で追加することもできます。

サービスの指定のほかに，次の説明に従って，ツールキットを指定する必要もあります。

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

WSDL の指定時に，**URL**，**ファイル**，**UDDI**，または **Quality Center** 内の場所を指定できます。



- ［**URL**］：サービスの完全な URL です。URL を指定するときは，省略したものでなく，必ず完全な URL を入力します。標準設定のブラウザを開くには，テキスト・ボックスの右側にある［**参照**］ボタンをクリックします。
- ［**ファイル**］：WSDL ファイルの完全パスと名前を入力します。ファイル・システム上で WSDL を探して指定するには，テキスト・ボックスの右側にある［**参照**］ボタンをクリックします。
- ［**UDDI**］：Universal Description, Discovery and Integration の略。サービスの共通リポジトリです。詳細については，次を参照してください。

➤ [**Quality Center**]：Quality Center リポジトリに格納されたサービスです。詳細については，371 ページ「Quality Center からのサービスの選択」を参照してください。

ツールキットの選択

サービスを指定したら，ツールキットを選択する必要があります。

VuGen では，.NET Framework と WSE 2 バージョン SP3 および Axis 1.3 Web Services Framework ツールキットをサポートしています。.Net ツールキットおよび Axis ツールキットの場合，VuGen により，実際のツールキットを使用してスクリプトがインポート，記録，および再生されます。

VuGen は，.NET または Axis ツールキットを使用しないサービス用に，Glue バージョン 4.1.2，MS SOAP バージョン 3.0，または一般のツールキットなどをエミュレートする汎用ツールキットを備えています。再生の前に，汎用 Mercury ソリューションのエミュレーションを設定できます。詳細については，386 ページ「Web サービス・ツールキットの実行環境の設定」を参照してください。

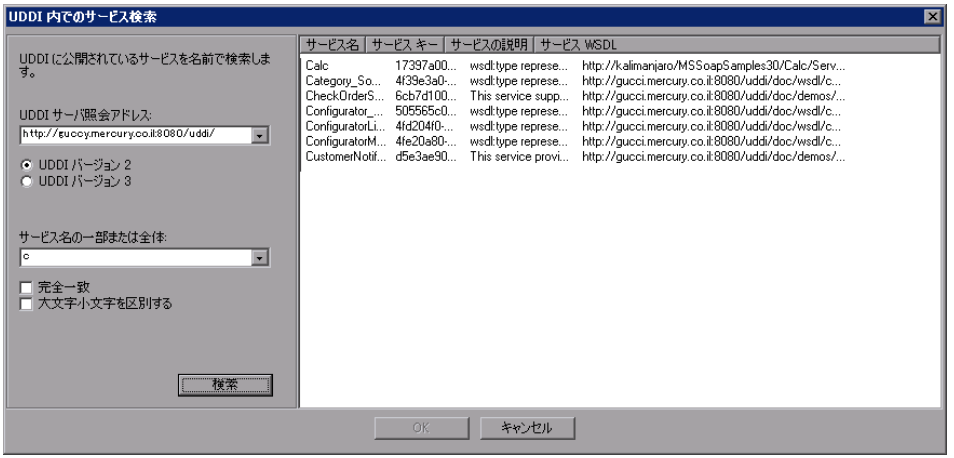
スクリプトの記録または WSDL ファイルのインポートを行う前にツールキットを選択します。選択したツールキットは，スクリプトに永続的に関連付けられ，以降のすべての記録，インポート，および再生で使用されます。

UDDI サーバ上のサービスの指定

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

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

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



UDDI サーバ上のサービスを検索するには，次の手順を実行します。

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

2 UDDI バージョンを指定します。

3 サービスの名前または名前の一部を指定します。該当する場合，［**完全一致**］または［**大文字小文字を区別する**］を選択して検索対象を絞り込みます。

4 ［**検索**］をクリックします。合致する結果がすべて表示されます。

5 リスト内でサービスをダブルクリックしてインポートします。

Quality Center からのサービスの選択

Mercury Quality Center は Mercury Service Test と連携して動作することで，サービス・エントリとテストを Quality Center に格納できるようにします。Quality Center 統合，Mercury サービス管理を使用して，テスト要件とテスト計画に従ってサービスの作成および編成ができます。

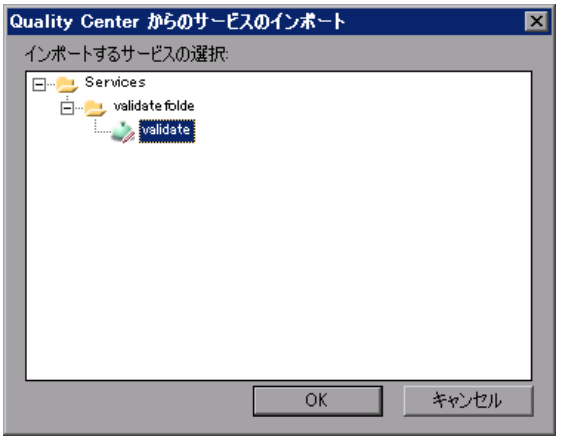
Quality Center からサービスを指定するには，次の手順を実行します。

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

2 Quality Center プロジェクトにまだ接続していなければ，［Quality Center 接続］をクリックして接続ダイアログ・ボックスを開きます。Quality Center に接続する方法の詳細については，252 ページ「Quality Center の接続と切断」を参照してください。

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

4 サービスをインポートするには，リスト内の該当サービスをダブルクリックします。



サービスの削除

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

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

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

WSDL ファイルの検証

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。WS-I の詳細については，http://ws-i.org を参照してください。

Web サービスが WS-I 準拠の場合，それは当該サービスが WS-I 規格に適合しており，複数のプラットフォームおよび複数の言語に対応することを意味します。

Web サービスの WS-I 準拠を主張するには，WS-I が配布する WS-I テスト・ツールを使用して Web サービスを検証する必要があります。テスト・ツールをインストールすると，VuGen にツールが統合化されます。

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。VuGen では Java 向けのツールの使用をお勧めします。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。

これらのツールは次の WS-I の Web サイトからダウンロードできます。

- **C# の場合**：http://www.ws-i.org/Testing/Tools/2004/12/SSBP_CS_Tools-BdAD.zip
- **Java の場合**：http://www.ws-i.org/Testing/Tools/2004/12/SSBP_Java_Tools-BdAD.zip

WSI_HOME という環境変数に **wsi-test-tools** のパスを値として定義すれば，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。

VuGen では，次のいくつかの領域で WS-I 検証をサポートしています。

- **サービス管理**：［サービス管理］ウィンドウから，1 つ以上の WSDL を手作業で検証します。詳細については，375 ページ「WSDL の検証」を参照してください。
- **WSDL WS-I 妥当再確認呼び出し**：スクリプトを実行するたびに WSDL を自動的に検証するには，検証ステップをスクリプトに挿入します。詳細については，375 ページ「検証の呼び出しの追加」を参照してください。
- **ステップごとの SOAP 妥当性確認**：特定のステップの SOAP メッセージを検証します。詳細については，379 ページ「SOAP メッセージの検証」を参照してください。

➤ **反復ごとの SOAP 妥当性確認**：反復全体の SOAP メッセージを検証します。詳細については，379 ページ「SOAP メッセージの検証」を参照してください。

WS-I 検証の設定

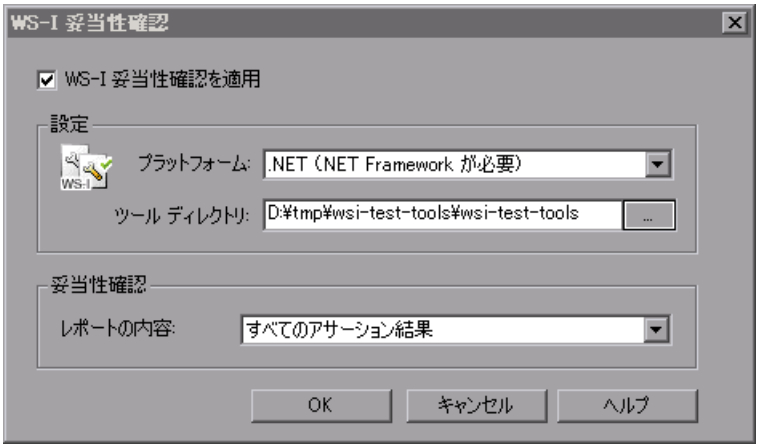
VuGen では，WSDL または，記録または再生中にサービスから送信された SOAP メッセージを検証できます。

検証を実行する前に，WS-I 検証が有効になっていて，要件に従って設定されていることを確認します。

WS-I 検証を有効にするには，次の手順を実行します。

1 ［**SOA Tools**］＞［**SOA 設定**］＞［**WS-I 妥当性確認**］を選択します。あるいは，［サービス管理］ウィンドウで［**WS-I 妥当性確認**］ボタンをクリックします。初めて検証を実行している場合は，［WS-I 妥当性確認］ボックスが開きます。

2 ［**WS-I 妥当性確認を適用**］を選択します。



3 Web サービスの**プラットフォーム**を選択します。Microsoft .NET（.NET Framework が必要），または Java（Sun Java）のいずれかです。

4 WS-I 検証ツール・ディレクトリ（**wsi-testing-tool**）の場所を指定します。

5 レポートに含めるメッセージを指定します。

➤ ［**すべてのアサーション結果**］：すべての結果を表示します。

➤ ［**結果が「失敗」のアサーション**］：「失敗」の結果ステータスを持つアサーションのみを表示します。

➤［**結果が「成功」以外のアサーション**］: 結果のステータスが「成功」とならなかったアサーションを表示します。

6［**OK**］をクリックします。

WSDL の検証

WSDL ファイルはスクリプトの作成前または作成後に検証できます。

WSDL ファイルを検証するには，次の手順を実行します。

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

2 検証するサービスを選択します。複数のサービスを選択するには **Ctrl** キーを使用します。

3［**WS-I 妥当性確認**］をクリックします。初めて検証を実行する場合は，374 ページ「WS-I 検証の設定」の説明に従って WS-I の設定を指定する必要があります。

4［**OK**］をクリックします。［妥当性チェック レポート］が開きます。

検証の呼び出しの追加

検証の呼び出しを手作業で追加して，スクリプトを再生するたびに特定のサービスを確認できます。検証が失敗した場合，スクリプトの実行は停止するので，独立のテスト・スクリプトを作成して検証を実行することをお勧めします。

スクリプトを実行すると，VuGen により検証が実行され，WSDL が WS-I に準拠しているかどうかを示すメッセージが出力ログに送信されます。また，検証レポートの場所も示されます。

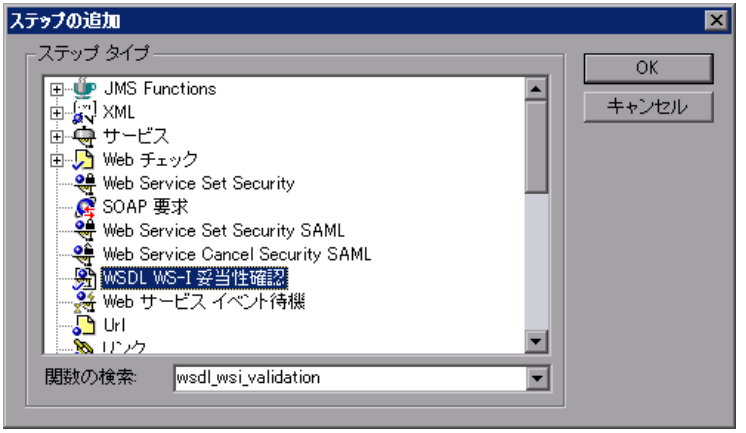
```
Action.c(33): WSDL WS-I validation for service "Calc" started
Action.c(33): Error: The service is not WS-I compliant
Action.c(33): For more information, see report in "C:¥Program
    Files¥Mercury¥Mercury SOA Tester¥scripts¥My Test
   3¥WSDL¥WSIValidationReport_Calc_CalcSoapBinding.xml"
```

スクリプトを複数回反復する場合は，検証ステップを **init** セクションに置くことをお勧めします。これにより，VuGen でサービスを反復のたびではなく 1 回だけ検証できます。

スクリプトに検証ステップを手作業で追加するには，次の手順を実行します。

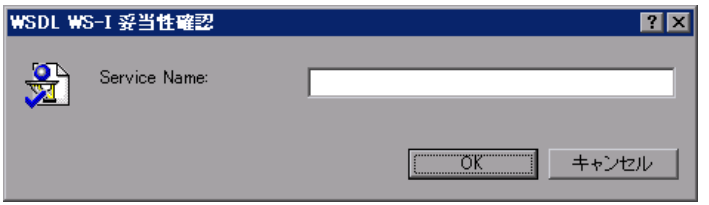
1 ツリー・ビューの中で，スクリプト内の該当する場所をクリックします。

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



3 ［**WSDL WS-I 妥当性確認**］を選択して，［**OK**］をクリックします。

4 検証するサービスの名前を入力します。



スクリプト・ビューでは，**wsdl_wsi_validation** 関数を追加できます。これらの関数の詳細については，「**オンライン関数リファレンス**」（［**ヘルプ**］＞［**関数リファレンス**］，または関数で **F1** キーをクリック）を参照してください。

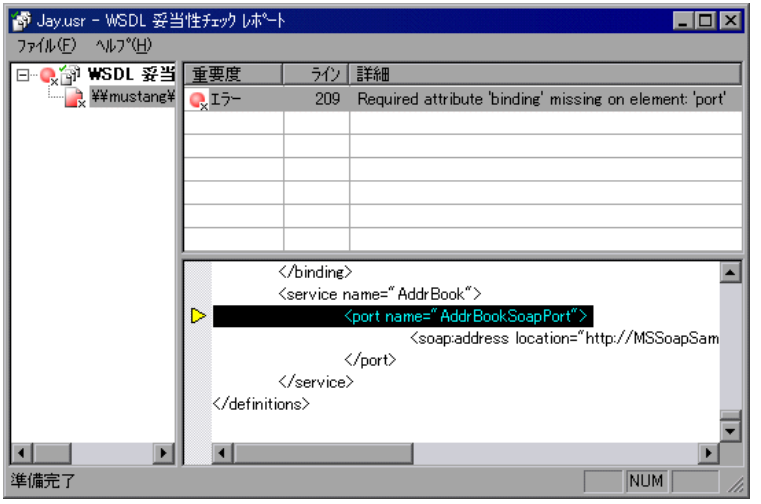
妥当性チェック・レポート

WSDL 妥当性チェック・レポートには，WSDL ファイルに関する情報が記載されます。［**重要度**］カラムの緑のアイコンは，問題がないことを意味します。また，黄色は警告を，赤はエラーを示します。

WSDL 妥当性チェック・レポートやネイティブ WS-I レポートなど複数のタイプのレポートを確認することで，準拠に関する問題の把握に役立てることができます。

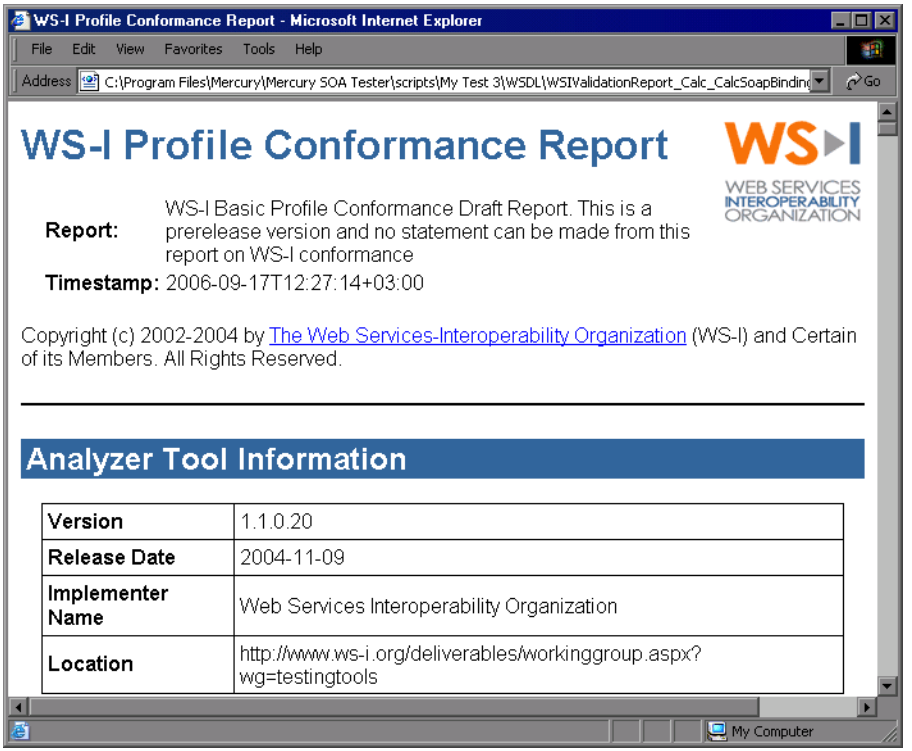
WSDL 妥当性チェック・レポート

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，WSDL ファイル内の該当する問題の箇所が表示されます。



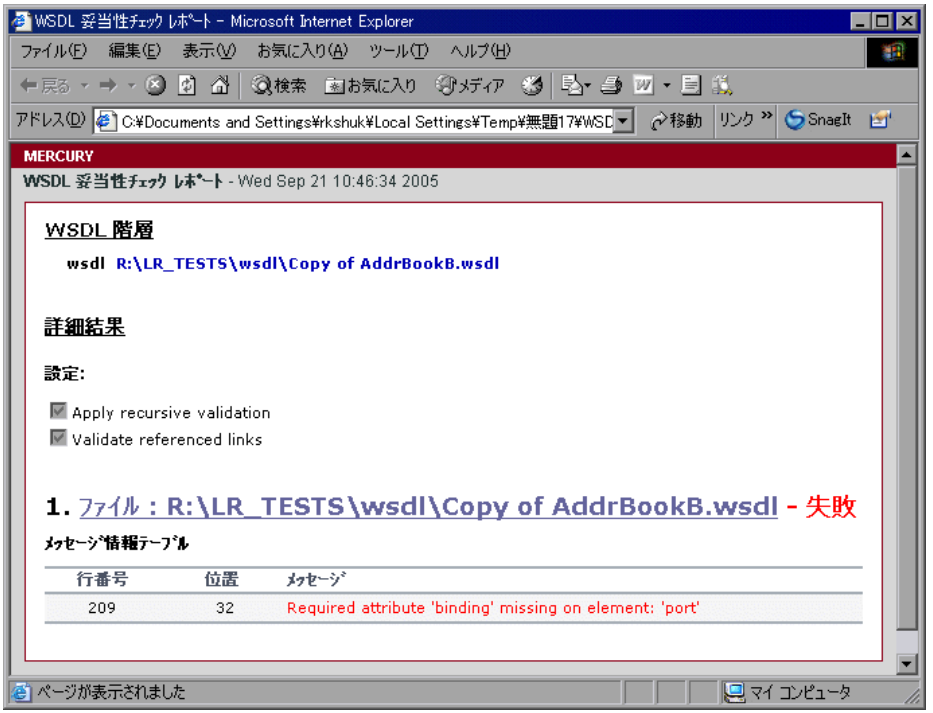
ネイティブ WS-I レポート

ネイティブ WS-I レポートは，WSDL が WS-I 規格に準拠または非準拠と判断された場合の理由を詳しく示す WS-I 基本プロファイル準拠レポートです。



ネイティブ WS-I レポートを表示するには，検証レポートの中でエラーを選択し，［**WS-I レポートを開く**］を選択します。ブラウザが開いて WS-I レポートが表示され，エラーの詳細が示されます。

また，後で見られるように，レポートを HTML ファイルに保存できます。



HTML 形式の検証サマリ・レポートを作成して保存するには，次の手順を実行します。

1 ［**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。

2 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。

3 WS-I ネイティブ・レポートを開くには，WS-I エラーを右クリックします。

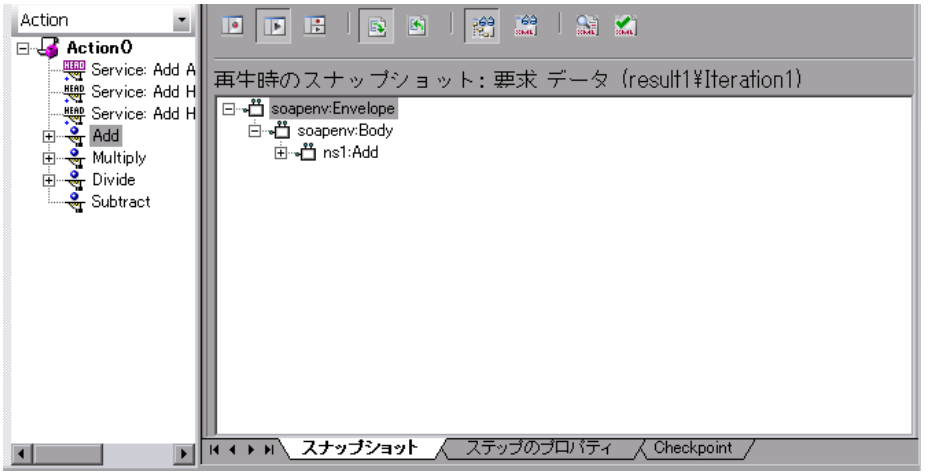
SOAP メッセージの検証

VuGen では，さまざまな応答の SOAP メッセージを検証できます。

特定のステップの SOAP メッセージを検証するには，次の手順を実行します。

1 ツリー・ビューの中で，検証するステップを選択します。

2［スナップショット］タブを選択します。



3［**SOAP の妥当性検証**］ボタンをクリックして，検証を開始します。

反復全体の SOAP メッセージを記録時または再生時に検証することもできます。

- 記録されたスナップショットを検証するには，［**SOA Tools**］＞［**SOAP 妥当性確認**］＞［**記録の妥当性確認**］を選択します。
- 現在の再生スナップショットを検証するには，［**SOA Tools**］＞［**SOAP 妥当性確認**］＞［**再生の妥当性確認**］を選択します。

VuGen により，検証データを含む検証レポートが表示されます。詳細については，375 ページ「WSDL の検証」を参照してください。

WSDL ファイルおよび XML ファイルの比較

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

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。ファイルを更新するには，［WSDL 比較レポート］の左側の表示枠でファイルを選択し，右クリック・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。

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，384 ページ「XML ファイルの比較」を参照してください。

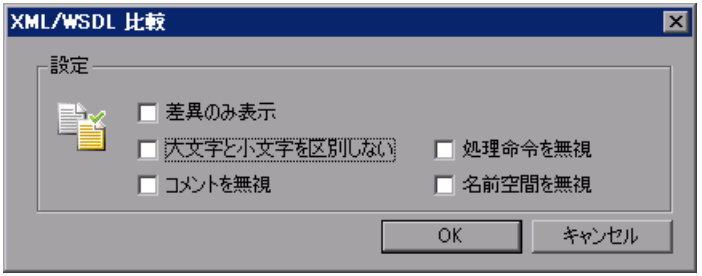
WSDL と XML の比較オプションの設定

VuGen では，WSDL 文書のローカル・コピーとグローバル・コピー，または XML ファイルの変更を比較する際に，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。

比較オプションを設定するには，次の手順を実行します。

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



2 ［**OK**］をクリックします。

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

比較レポート

VuGen 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。

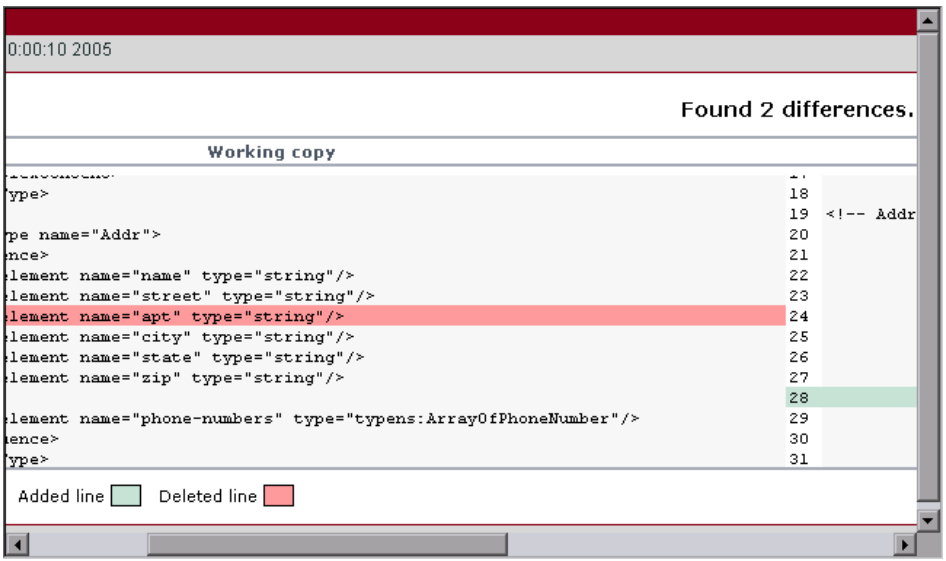
WSDL 比較レポートには，［**Working Copy**］と［**Original File**］の 2 つのカラムがあります。［Working Copy］はスクリプトと一緒に格納されている WSDL です。一方，［Original File］は元の場所（ネットワーク・ファイル・パスまたは URL）にある WSDL です。

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

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

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

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

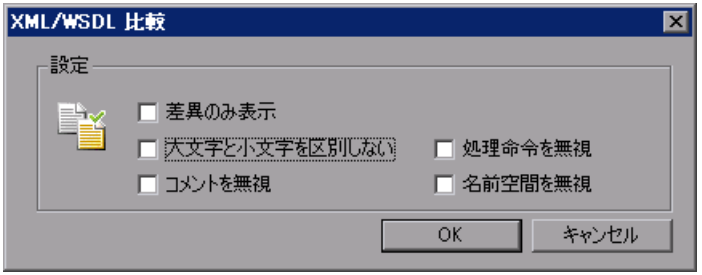


WSDL 比較の実行

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

WSDL ファイルを比較するには，次の手順を実行します。

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



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

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

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

5 ファイルを下にスクロールして相違点を探します。

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

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

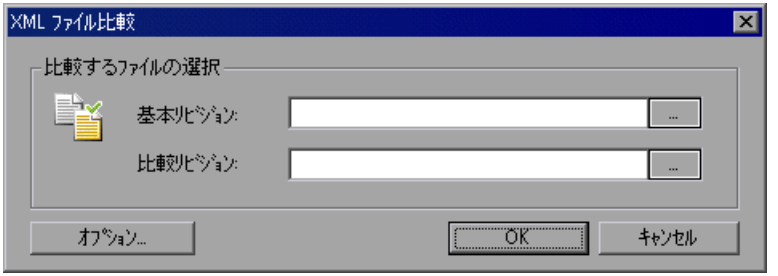
XML ファイルの比較

VuGen は，2 つの XML ファイルを比較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，381 ページ「WSDL と XML の比較オプションの設定」を参照してください。

2 つの XML ファイルを比較するには，次の手順を実行します。

1 ［**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，382 ページ「比較レポート」を参照してください。

第 24 章

SOA/Web サービス・スクリプトの実行

SOA/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，サービスが期待どおりに実行された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 サービス仮想ユーザの実行について
- Web サービス・ツールキットの実行環境の設定
- Web サービス JMS の実行環境の設定
- Web サービス・スクリプトのスナップショットの表示
- XML を使った作業
- 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

ツリー・ビューには，スクリプトを理解し調べることができる 3 つのタブがあります。

- ［**スナップショット**］: 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。詳細については，389 ページ「Web サービス・スクリプトのスナップショットの表示」を参照してください。
- ［**ステップのプロパティ**］: スクリプトの各ステップとその引数の値，添付ファイル，SOAP ヘッダー，およびトランスポート層の設定の詳細が示されます。詳細については，第 22 章「Web サービス呼び出しプロパティの設定」を参照してください。

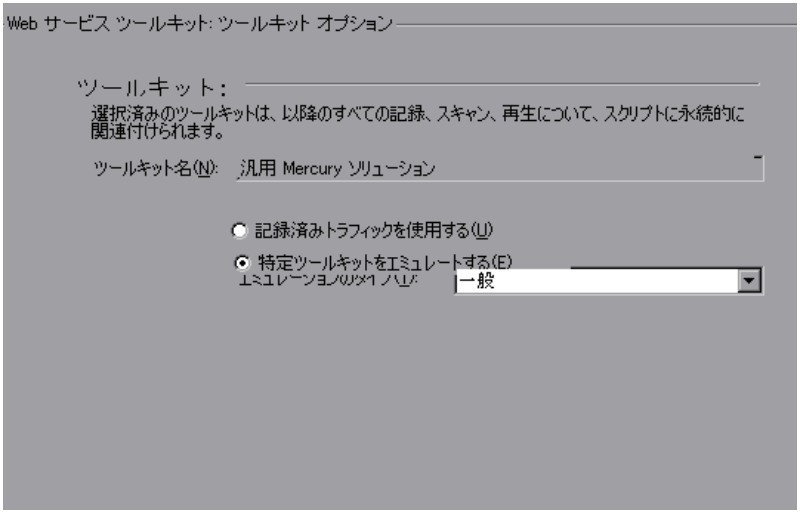
➤ [**Checkpoint**]：サービスが正しい結果となったかどうかの判断に役立つ検証ポイントです。詳細については，318 ページ「チェックポイントの設定」を参照してください。

スクリプトを実行する前に，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。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 サービス関連の設定（ツールキットおよび JMS）が含まれます。

詳細については，次の項または第 13 章「実行環境の設定」を参照してください。

Web サービス・ツールキットの実行環境の設定

実行環境を設定して，汎用 Mercury ソリューション・ツールキットで作成されたスクリプトに対してツールキットのエミュレーションを指定できます。この設定は .NET および Axis ツールキットには適用されません。



記録されたトラフィックを使用するか，エミュレーションの種類を指定できます。

[**記録済みトラフィックを使用する**]：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，このオプションを指定することで，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。

［**特定のツールキットをエミュレートする**］：記録した属性ではなく，特定のツールキットをエミュレートします。エミュレーションの種類（MS SOAP, Glue，または一般）を指定できます。

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 サービス ツールキット：ツールキット オプション**］ノードを選択します。

2 使用するオプション（［**記録済みトラフィックを使用する**］または［**特定のツールキットをエミュレートする**］）を指定します。

3 特定のツールキットをエミュレートするよう選択した場合，［**エミュレーションのタイプ**］を選択します。

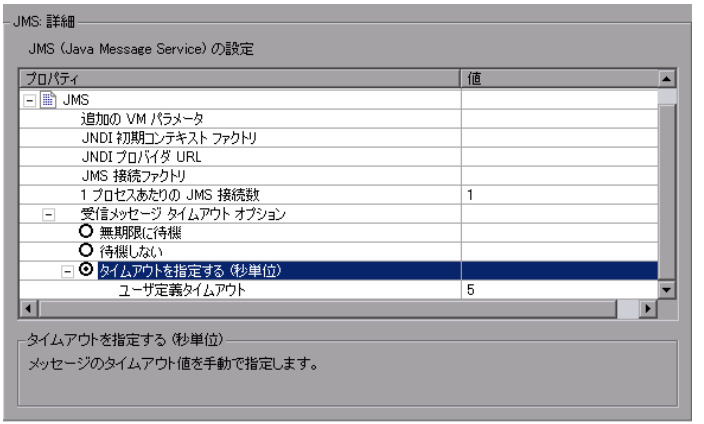
4 ［**OK**］をクリックします。

Web サービス JMS の実行環境の設定

Web サービス呼び出しのトランスポートとして JMS を使用するには，いくつかのリソースの割り当ておよび設定が必要です。これらのリソースには，JVM, JNDI 初期化パラメータ，JMS リソース，およびタイムアウト値が含まれます。

VuGen では，実行環境の設定でそれらのリソースのいくつかを設定できます。

VM（仮想マシン），JMS 接続，およびメッセージ・タイムアウトの領域のオプションを設定できます。



VM

［**外部 VM を使用する**］：標準以外の VM（仮想マシン）を選択できます。このオプションを無効にすると，仮想ユーザは VuGen で提供される JVM を使用します。

［**JVM ホーム**］：外部 JVM の場所です。JDK_HOME によって定義されている JDK のホーム・ディレクトリを指すようにします。VuGen では，JDK 1.4 以降をサポートしています。

［**クラスパス**］：ほかの必要なサポート・クラスとともにベンダによって実装された JMS クラスです。JMS 実装ベンダによって決定されます。

JMS

- ［**追加の VM パラメータ**］：Xbootclasspath や JVM マニュアルで指定されているパラメータ など，JVM に送る追加のパラメータです。
- ［**JNDI 初期コンテキスト ファクトリ**］：初期コンテキストを作成するファクトリ・クラスの完全指定クラス名です。次に例を示します。

  Weblogic — weblogic.jndi.WLInitialContextFactory

  Websphere — com.ibm.websphere.naming.WsnInitialContextFactory
- ［**JNDI プロバイダ URL**］：サービス・プロバイダの URL の文字列です。次に例を示します。

  Weblogic — t3://myserver:myport

  Websphere — iiop://myserver:myport

- ［**JMS 接続ファクトリ**］: JMS 接続ファクトリの JNDI 名です。1 つのスクリプトに指定できる接続ファクトリは 1 つだけです。
- ［**1 プロセスあたりの JMS 接続数**］: **mdrv** プロセス，または仮想ユーザごとの JMS 接続の数です。接続を共有するすべての仮想ユーザは，同じメッセージを受信します。標準設定は 1 仮想ユーザ，最大値は 50 仮想ユーザです。プロセスごとの接続数が少ないほどパフォーマンスは向上します。
- ［**受信メッセージ タイムアウト オプション**］: 受信メッセージのタイムアウトです。標準設定は［**待機しない**］です。
  - ［**無期限に待機**］: メッセージを必要なだけ待機してから次に進みます。
  - ［**待機しない**］: 受信メッセージを待機せず，すぐにスクリプトに制御を戻します。キューにメッセージがなければ操作は失敗となります（標準設定）。
  - ［**タイムアウトを指定する（秒単位）**］: メッセージのタイムアウト値を手作業で指定します。タイムアウトするまでにメッセージが到着しなければ操作は失敗となります。
    - ［**ユーザ定義タイムアウト**］: タイムアウトするまでにメッセージを待機する時間を指定します（秒単位）。標準設定は 5 秒です。

Web サービス・スクリプトの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。

スナップショットを表示する方法はいくつかあります。

- 記録または再生，あるいはその両方
- 要求データまたは応答データ
- ツリー・ビューまたは XML ビュー

スナップショット・ビューアには，XML コードを対象とした作業に使用できるいくつかのユーティリティ（［XML 要素の検索］および［SOAP の妥当性検証］）もあります。

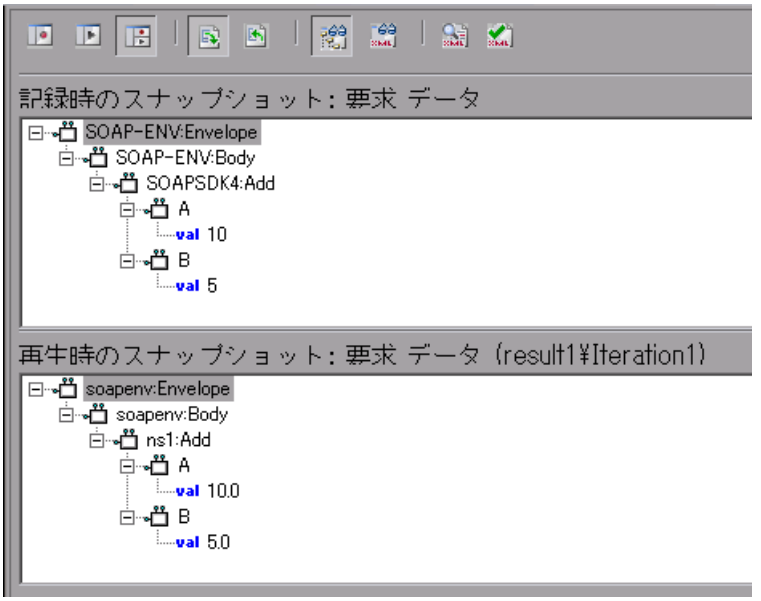
［スナップショット］ウィンドウのボタンを使用してビューを制御します。

	［記録時のスナップショット］
	［再生時のスナップショット］
	［記録時 vs 再生時のスナップショット］
	［要求］
	［応答］
	［ツリー ビュー］
	［XML ビュー］
	［XML 要素の検索］ (ツリー・ビューでのみ表示)
	［SOAP の妥当性検証］

選択されているオプションのボタンは拡大して表示されます。ツリー・ビューでは，ノードを展開して引数の値を表示できます（割り当てられている場合）。

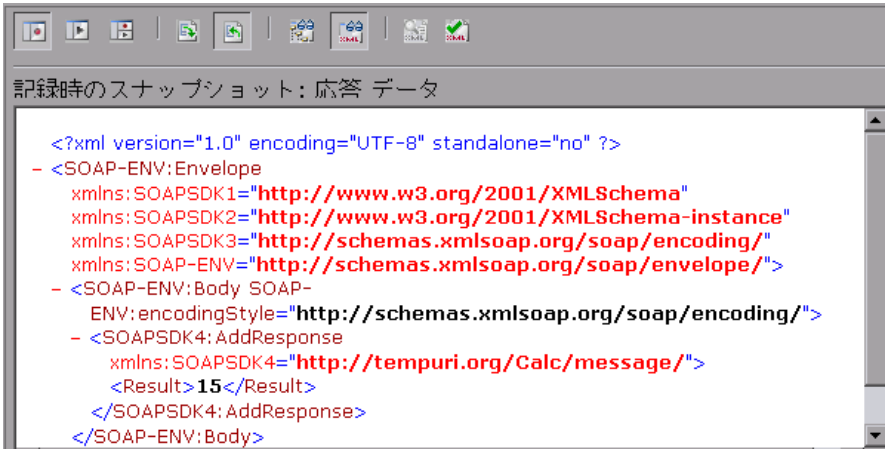
要求ビューには，Web サービス・セッション中にサーバに送られた値が表示されます。**応答**ビューには，サーバから返された結果の値が表示されます。

次の例では，［スナップショット］ウィンドウに**ツリー・ビュー**で**要求**データの**記録**および**再生**スナップショットが表示されています。



ノードの詳細については，ノードを選択し，右クリック・メニューから［**ノードのプロパティ**］を選択します。

XML ビューでは，SOAP エンベロープと XML 要素を表示できます。



XML を使った作業

Web サービスでは，XML の表示，クエリ，および編集ができます。また XML 構造全体をパラメータ化して 1 つのパラメータにできます。

以降の各項では，次の項目について説明します。

- XML ツリーに対するクエリ
- XML ツリーの編集
- XML 要素のパラメータ化
- SOAP 応答の保存とコピー

XML ツリーに対するクエリ

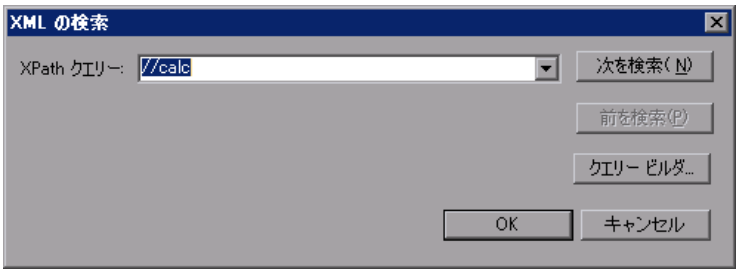
XML ツリーに対してクエリを行い，特定の要素や値の検索および検査ができます。

XML を検索するには，標準の XPath 構文に従ったクエリを使用します。有効な XPath クエリを作成するには，組み込みのクエリ・ビルダを使用します。XML 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。

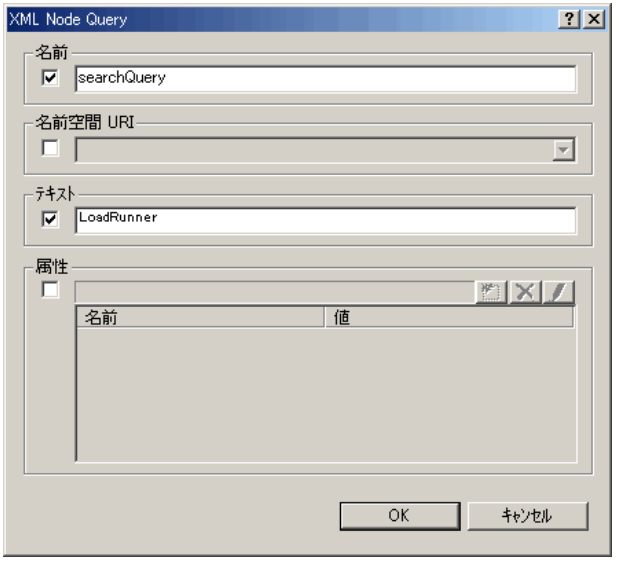
クエリを行うには，次の手順を実行します。

1 スナップショットのツリー・ビューで，要求または応答のスナップショットを選択し，［**XML 要素の検索**］ボタンをクリックします。

2 XPath クエリを入力します。



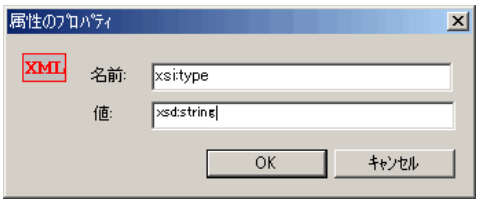
3 [**クエリー ビルダ**] をクリックします。これは XPath クエリの作成を支援します。[XML ノードのクエリ] ダイアログ・ボックスが開きます。1 つまたは複数の項目を検索対象として有効にします。



該当するボックスに検索テキストを入力します。

- [**名前**] セクションを有効にして，ノードまたは要素の名前を検索します。
- [**名前空間 URI**] セクションを有効にして，名前空間を検索します。
- [**テキスト**] セクションを有効にして，[名前] ボックスに表示されている要素の値を検索します。
- [**属性**] セクションを有効にして，属性を検索します。

  - 属性を追加するには，[**追加**] ボタンをクリックします。[属性のプロパティ] ボックスが開きます。属性の名前と値を入力します。[**OK**] をクリックします。

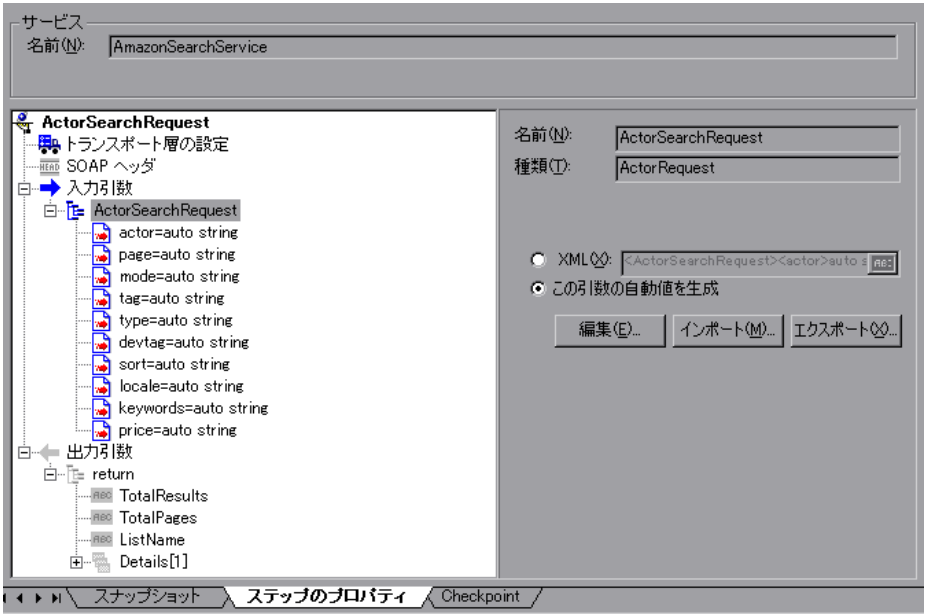


4［XML ノードのクエリ］ダイアログ・ボックスの中で［**OK**］をクリックします。VuGen は，クエリ・テキストを［XPath クエリー］ボックスに入れます。

5［**次を検索**］をクリックして検索を開始します。

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。

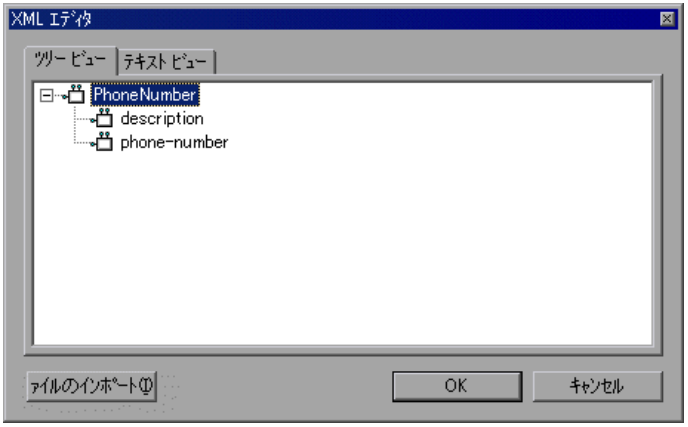
XML 文字列を処理するには，次の手順を実行します。

1 ツリー・ビューで，変更する入力引数が含まれるステップを選択し，［**ステップのプロパティ**］タブをクリックします。

2 メソッドのツリー階層で，複合型または配列引数をクリックします。最も右側の表示枠に，XML コードが単独の文字列として表示されます。［**XML**］オプションを選択します。

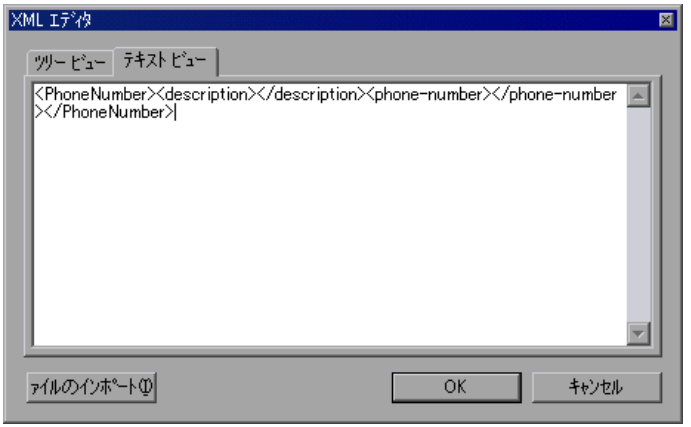
3 そのエントリに対応する XML コードを編集するには，［**編集**］をクリックします。XML 要素自体の変更ではなく，要素の値および配列要素の数に対する変更のみ保存されることを示す警告が発行されることがあります。

4 ［**OK**］をクリックします。［XML エディタ］ダイアログ・ボックスが開きます。



5 XML ツリー・ビューでノードをダブルクリックしてプロパティ・ダイアログ・ボックスを開きます。必要に応じて値を編集します。［**OK**］をクリックして，新しい値を保存します。

6 コードをテキスト・モードで編集するには，［**テキスト ビュー**］タブをクリックします。XML コードを手作業で編集します。



［**OK**］をクリックして変更内容を保存します。

7 以前に保存した XML ファイルをインポートするには，［**インポート**］をクリックし，ファイルの場所を指定します。ファイルを［XML エディタ］ダイアログ・ボックスで編集します。

8 XML データをファイルに保存してほかのテストで使用できるようにするには，［**エクスポート**］をクリックし，場所を指定します。

XML 要素のパラメータ化

VuGen では，すべての引数の値のパラメータ化をサポートしています。パラメータ化により，元の値を外部の値に置き換えることができます。パラメータ化の詳細については，第 9 章「VuGen パラメータを使った作業」を参照してください。

引数が単純で配列型でない場合，引数を単純パラメータに置き換えることができます。たとえば，足し算を行うサービスをテストする場合，各入力引数をパラメータに置き換え，値をファイルまたはテーブルに格納できます。

しかし，引数が多数の値を持つ複雑な構造である場合，XML タイプのパラメータを使用して構造全体を 1 つのパラメータに置き換えることができます。XML タイプのパラメータの値セットをいくつか作成して，反復ごとに異なる値セットを割り当てることもできます。詳細については，131 ページ「XML パラメータ・タイプ」を参照してください。

引数の値をパラメータで保存するには，次の手順を実行します。

1 ［**ステップのプロパティ**］タブに切り替えて，値をパラメータ化する親または子要素を選択します。

2 ［**入力引数**］ノードで，パラメータ化する引数を選択します。右側の表示枠で，［**値**］ボックスの［**ABC**］アイコンをクリックします。［パラメータの選択または作成］ダイアログ・ボックスが開きます。

3 パラメータの名前と種類を指定します。

4 ［**プロパティ**］をクリックしてパラメータのタイプ（ファイル，XML など）を設定し，値を割り当てます。

詳細については，第 9 章「VuGen パラメータを使った作業」を参照してください。

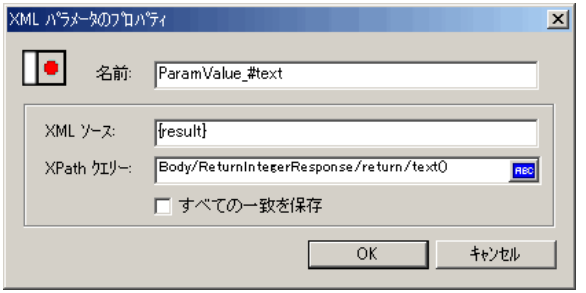
SOAP 応答の保存とコピー

入力引数の値を XML タイプのパラメータとして保存できほか，SOAP 応答をパラメータに保存したり，エディタで使用するためにコピーしたりできます。

SOAP 応答をパラメータに保存するには，次の手順を実行します。

1 ［**スナップショット**］タブに切り替えて，値をパラメータ化する対象の親または子要素を選択します。

2 右クリック・メニューから［**パラメータに XML を保存**］を選択します。［XML パラメータのプロパティ］ダイアログ・ボックスに，選択した XML 要素のプロパティが表示されます。



3 XML パラメータ名を指定して，［**OK**］をクリックします。

別のエディタ内で使用するために XML 構造をコピーするには，右クリックメニューから［**XML のコピー**］を選択します。

Web サービス関数の使用

Web サービス関数はサービスを呼び出します。また，セキュリティと同期化を提供します。関数は次のとおりです。

関数名	詳細
soap_request	SOAP 要求を実行します。
web_service_call	Web サービスを呼び出します。
web_service_set_security	セキュリティ・トークンを後続の SOAP および Web サービス呼び出しに追加します。詳細については，350 ページ「Web サービス・セキュリティ・ポリシーの作成」を参照してください。
web_service_cancel_security	先行する web_service_set_security 呼び出しによる設定を取り消します。
web_service_cancel_security_saml	後続の Web サービス呼び出しの SAML セキュリティを取り消します。
web_service_set_security_saml	後続の Web サービス呼び出しに SAML セキュリティを追加します。
web_service_wait_for_event	前の非同期サービス要求に対する応答を待機します。詳細については，338 ページ「トランスポート層の設定」を参照してください。

さらに，JMS 関数 **jms_<suffix>**，または XML 関数 **lr_xml_<suffix>** を使用して，スクリプトを強化することもできます。詳細については，「**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**]）。

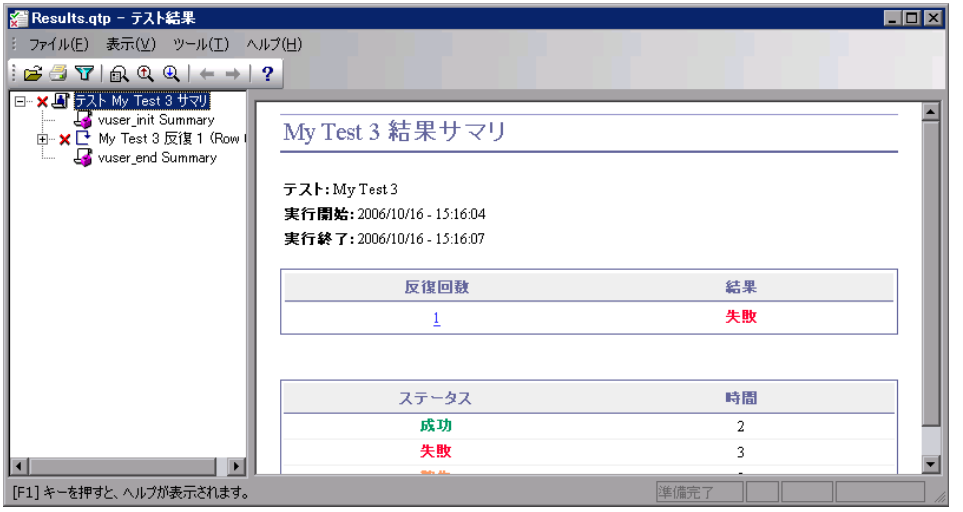
Web サービス・レポートの表示

Web サービス・スクリプトを実行した後，テスト結果ビューアを使用してテスト結果のサマリを表示できます。ビューアにはチェックポイントの結果も表示されます。

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 16 章「テスト結果の表示」を参照してください。

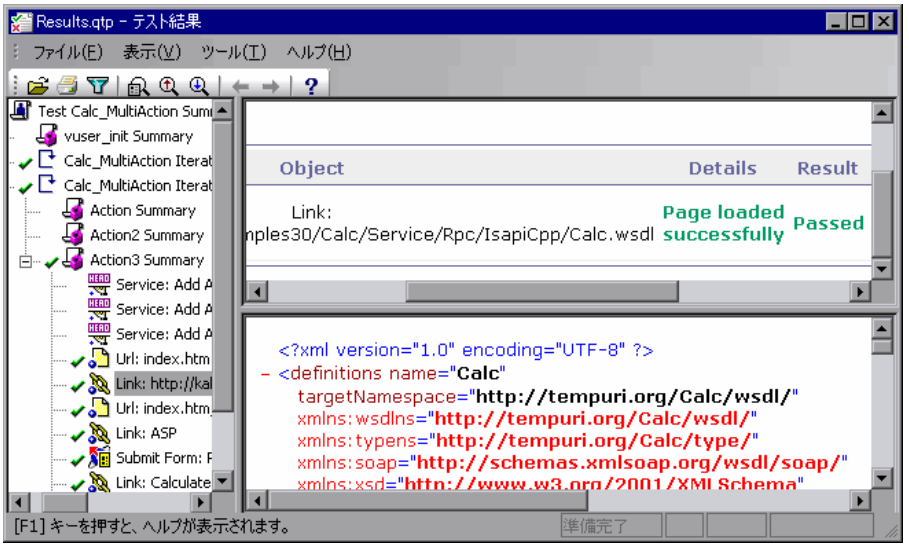
サマリ・レポートを開くには，［**表示**］＞［**テスト結果**］を選択します。

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。

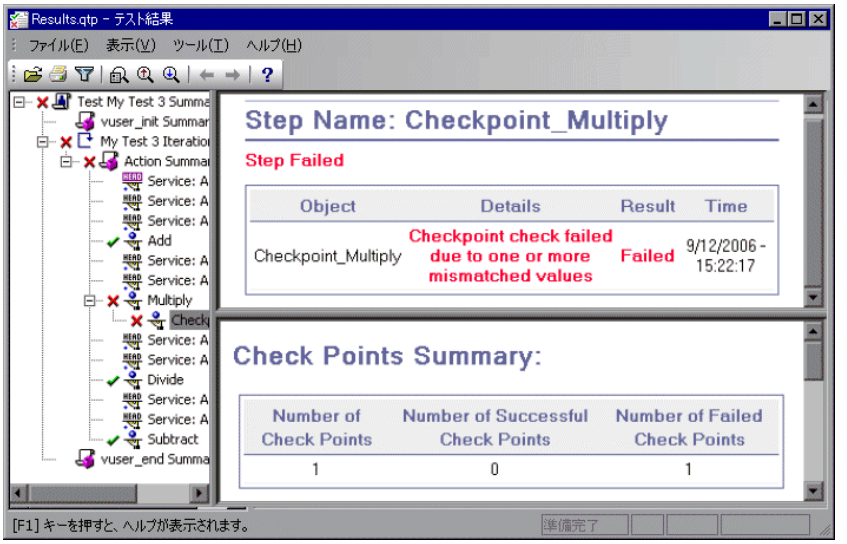
Web サービス呼び出しの場合は，［結果］ウィンドウの下の表示枠に SOAP 応答の内容が表示されます。



VuGen がスクリプトを解釈できない場合，またはほかのタイプのエラーが発生した場合は，問題を示すメッセージがレポートの右側の表示枠に表示されます。

［結果］ウィンドウにはチェックポイントの結果も表示されます。失敗の理由とともにサマリが表示されます。また［Expected Values］および［Actual Result］も表示されます。

チェックポイントの詳細を表示するには，左側の表示枠で該当するステップを展開し，［**チェックポイント**］ノードをクリックします。



テスト結果での作業の詳細については，第 16 章「テスト結果の表示」を参照してください。

第 25 章

サーバ・トラフィック・スクリプトの作成

VuGen を使用すると，サーバのトラフィックをキャプチャしたファイルの分析により，Web サービスをテストするスクリプトを作成できます。

本章では，次の項目について説明します。

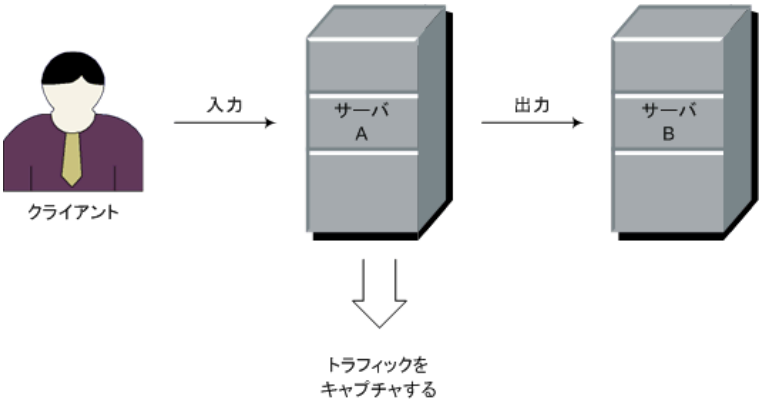
- サーバ・トラフィック・スクリプトの作成について
- サーバ・トラフィック・スクリプトの概要
- キャプチャ・ファイルの生成
- サーバ・トラフィックに基づく基本スクリプトの作成
- トラフィック情報の指定
- 受信フィルタまたは送信フィルタの選択
- SSL 証明書の指定

サーバ・トラフィック・スクリプトの作成について

エンタープライズ・システムや複合システムをテストする場合，クライアント側からパフォーマンスを測定することに重点が置かれます。通常は，VuGen によりアプリケーションまたはブラウザでユーザが実行するアクションが記録され，サーバに対するクライアントのアクションや要求をエミュレートするスクリプトが生成されます。

テスト環境によっては，サーバに対する要求を取得するためのクライアント・アプリケーションを記録することができない場合があります。これはたとえば，サーバがクライアントとして動作していたり，クライアント・アプリケーションを利用できなかったりする状況の場合にあり得ます。このような場合，VuGen の**トラフィック分析**機能を使用してスクリプトを作成できます。

トラフィック分析機能では，サーバ・ネットワーク・トラフィックが格納されているキャプチャ・ファイルを調査することで，サーバとの間で送受信された要求をエミュレートするスクリプトを作成します。サーバ・トラフィックを分析してスクリプトを作成する手順の詳細については，次項のサーバ・トラフィック・スクリプトの概要で説明します。



エミュレーションには，**受信トラフィック**と**送信トラフィック**の 2 つのタイプがあります。

受信トラフィック・スクリプトは，サーバに要求を送信したいけれども，たとえばセキュリティ上の制約があるために，クライアント・アプリケーションを利用できないような状況をエミュレートします。この場合，最も正確な解決方法は，サーバがクライアント側から**受信**するトラフィックに基づいてスクリプトを生成することです。

受信サーバ・ネットワーク・トラフィックを指定するには，サーバの IP アドレスとアプリケーションが対象としているポート番号を指定します。VuGen により，サーバが受信するすべてのトラフィックが調査され，関係するメッセージが抽出されて，スクリプトが作成されます。前出の図で，クライアントが利用できない状況では，受信スクリプトを作成して，**サーバ A** が受信する要求をエミュレートします。

送信トラフィック・スクリプトは，別のサーバに対してクライアントとして動作するサーバをエミュレートします。いくつかの内部サーバを持ったアプリケーション・サーバでは，サーバ・マシン間の通信のエミュレーションが必要なことがあります。たとえば前出の図の **サーバ A** と **サーバ B** の間です。この場合の解決方法は，特定のサーバから**送信**されたトラフィックに基づいてスクリプトを生成することです。

送信トラフィック・スクリプトを作成するには，送信トラフィックをエミュレートするサーバの IP アドレスを指定します。VuGen によりそのサーバから送信されるトラフィックが抽出されます。前出の図では，送信スクリプトを使用して **サーバ A** により **サーバ B** に送信される要求をエミュレートできます。

サーバ・トラフィック・スクリプトの概要

次の項では，サーバ・トラフィックを分析するスクリプトの作成プロセスを説明します。

1 **キャプチャ・ファイルを作成します。**

VuGen では，キャプチャ・ファイルを使用してサーバ・トラフィックを分析し，エミュレートします。詳細については，406 ページ「キャプチャ・ファイルの生成」を参照してください。

2 **新しい Web サービス・スクリプトを作成します。**

VuGen の主要インタフェースを使用して，新しい Web サービス・スクリプトを作成します。詳細については，408 ページ「サーバ・トラフィックに基づく基本スクリプトの作成」を参照してください。

3 **サービスを指定します（任意）。**

高レベルのスクリプトを作成するには，テストの対象となる Web サービスが記述されている WSDL をインポートします。

4 **トラフィック情報を指定します。**

トラフィック・ファイルの場所と，スクリプトが受信トラフィック向けか送信トラフィック向けかを指定します。詳細については，410 ページ「トラフィック情報の指定」を参照してください。

5 **トラフィック・フィルタの記録オプションを指定します。**

フィルタ・オプションを使用することで，スクリプトに含めるホストと除外するホストを指定できます。詳細については，411 ページ「受信フィルタまたは送信フィルタの選択」を参照してください。

6 **SSL 証明書情報を指定します。**

SSL を設定することで，スクリプトを生成するために HTTPS 経由のセキュアなトラフィックを分析できます。詳細については，412 ページ「SSL 証明書の指定」を参照してください。

キャプチャ・ファイルの生成

キャプチャ・ファイルは，ネットワークを通過したすべての TCP トラフィックのログを含んだトレース・ファイルです。スニッファ・アプリケーションを使用して，すべてのネットワーク・トラフィックのダンプを取得します。スニッファはネットワーク上のすべてのイベントをキャプチャし，キャプチャ・ファイルに保存します。

より小さく，処理しやすいスクリプトを生成するには，アプリケーションでアクションを実行する間にのみネットワーク・トラフィックをキャプチャするようにします。

注：キャプチャ・ファイルにはループバック・ネットワーク・トラフィックは含まれません。

キャプチャ・ファイルを取得するには，Mercury のコマンド・ライン・ユーティリティまたは既存のキャプチャ・ツールを使用します。

Mercury のコマンド・ライン・ユーティリティ

Mercury のコマンド・ライン・ユーティリティ **lrtcpdump** は，製品の **bin** ディレクトリにあります。プラットフォームごとに次の個別のユーティリティがあります。

lrtcpdump.exe（Windows），**lrtcpdump.hp9**，**lrtcpdump.ibm**，**lrtcpdump.linux**，および **lrtcpdump.solv4**

キャプチャ・ツールを起動する構文は次のとおりです。

```
lrtcpdump -i <インタフェース> -f <ファイル>
```

ここで**<インタフェース>**は，トラフィックをキャプチャするネットワーク・カードの名前です。**<ファイル>**は，情報を格納するキャプチャ・ファイルの名前です。

Windows プラットフォームでキャプチャ・ファイルを作成するには，次の手順を実行します。

1 ［**スタート**］＞［**ファイル名を指定して実行**］をクリックします。**cmd** と入力して［**OK**］をクリックし，コマンド・ウィンドウを開きます。

2 製品の **bin** ディレクトリにある **lrtcpdump.exe** プログラムをドラッグしてドロップするか，そのフル・パスを入力します。

3 次の構文を使用して，キャプチャ・ファイルのファイル名を渡します。
lrtcpdump -f ＜ファイル＞

たとえば，「**lrtcpdump -f mydump.cap**」のように入力します。

4 **lrtcpdump** により，ネットワーク・カードの選択を求められます。インタフェース・カードが複数ある場合は，それらが一覧表示されます。インタフェース・カードの番号（1，2，3 など）を入力し，**Enter** キーを押します。

5 アプリケーション内で，標準的な操作を実行します。

6 コマンドライン・ウィンドウに戻り，**Enter** キーを押してキャプチャ・セッションを終了します。

Unix プラットフォームでキャプチャ・ファイルを作成するには，次の手順を実行します。

1 製品の **bin** ディレクトリで，プラットフォームに適した **lrtcpdump** ユーティリティを探します。該当する lrtcpdump ユーティリティを UNIX マシンからアクセスできるフォルダにコピーします。たとえば HP プラットフォームの場合，**lrtcpdump.hp9** をコピーします。FTP を使用する場合，必ずバイナリ転送モードを使用してください。

2 ユーティリティを実行するために，root ユーザに切り替えます。

3 実行パーミッションを次のように与えます。
chmod 755 lrtcpdump. ＜プラットフォーム＞

4 UNIX プラットフォームでは，インタフェース・カードが複数ある場合，**lrtcpdump** はアルファベット順で最初のものを使用します。全インタフェースの一覧を取得するには，**ifconfig** コマンドを使用します。

5 インタフェースとファイル名を指定し，完全な構文でユーティリティを実行します。たとえば，「**lrtcpdump.hp9 -i etho -f mydump.cap**」のように入力します。ネットワーク・トラフィックのキャプチャが始まります。

6 アプリケーション内で，標準的な操作を実行します。

7 **lrtcpdump** を実行しているウィンドウに戻り，画面の指示に従ってキャプチャ・セッションを終了します。

8 VuGen を実行するマシンからアクセスできるネットワーク上の場所にキャプチャ・ファイルを格納します。

既存のキャプチャ・ツール

ほとんどの UNIX オペレーティング・システムは，キャプチャ・ツールを備えています。ほかにも，Ethereal/tcpdump などダウンロード可能なキャプチャ・ツールが数多くあります。

外部ツールを使用する場合，すべてのパケット・データがキャプチャされ，切り捨てられているものがないことを確認します。

注：一部のユーティリティには引数を追加する必要があります。たとえば，**tcpdump** には，データを切り捨てずにパケットをキャプチャするために引数 **-s 0** が必要です。

サーバ・トラフィックに基づく基本スクリプトの作成

記録して作成するスクリプトと同じように，サーバ・トラフィックからスクリプトを作成します。

スクリプトに Web サービスを任意で指定することもできます。サービスを指定した場合，VuGen は **web_service_call** 関数を含んだスクリプトを作成します。サービスを指定しなかった場合，VuGen は **soap_request** 関数を含んだスクリプトを作成します。

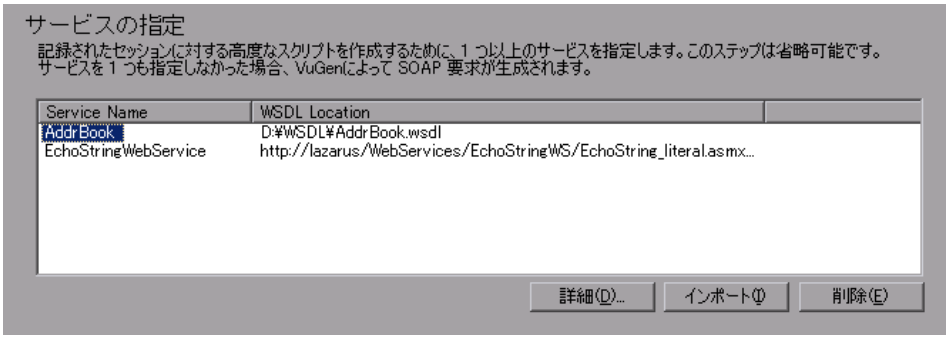
サーバ・トラフィック・スクリプトを作成するには，次の手順を実行します。

1 ［**ファイル**］＞［**新規作成**］を選択し，左側の表示枠で［**新規シングル プロトコル スクリプト**］をクリックします。

2 ［**Web Services**］プロトコルを選択して［**OK**］をクリックします。

3 ［**トラフィックの分析**］ボタンをクリックするか，［**仮想ユーザ**］＞［**トラフィックの分析**］を選択します。ウィザードが起動します。

4 リストに 1 つ以上のサービスを追加します。この手順は任意です。



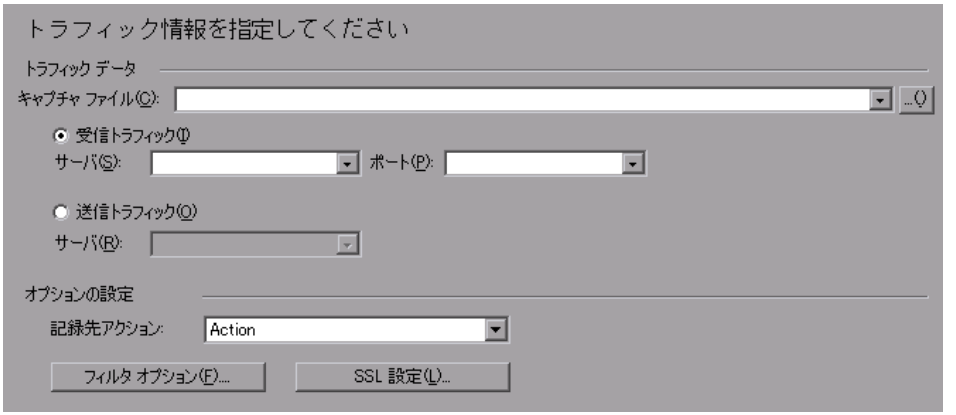
- 新規サービスを追加するには，［**インポート**］をクリックします。［サービスのインポート］ダイアログ・ボックスで，WSDL の場所を指定します。URL，ファイル，UDDI サーバ（Systinet など），または Quality Center 内の場所を指定できます。［サービスのインポート］ダイアログ・ボックスで，サービスを分析するためのツールキットも選択します。選択したツールキットはスクリプトに永続的に関連付けられます。変更はできません。詳細については，369 ページ「サービスのインポート」を参照してください。
- サービスの設定または詳細の表示を行うには，［**詳細**］をクリックして［サービス管理］ウィンドウを開きます。サービス管理の詳細については，第 23 章「Web サービスの管理」を参照してください。
- リストに表示されているサービスを削除するには，対象サービスを選択して［**削除**］をクリックします。

5 ウィザード画面の最下部にある［**次へ**］をクリックして，トラフィック・ファイルの情報を指定します。詳細については，次を参照してください。

6 トラフィックの情報を入力して［**完了**］をクリックすると，スクリプトが生成されます。

トラフィック情報の指定

トラフィック・ファイルにはすべてのネットワーク・トラフィックのダンプが含まれています。ウィザードを使用して，トラフィック・ファイルの場所と，スクリプトが受信トラフィックと送信トラフィックのどちらをエミュレートするのかを指定します。



［**キャプチャ ファイル**］：トラフィック・ファイルの名前とパスです。通常，拡張子は .cap です。

［**受信トラフィック**］の［**サーバ**］と［**ポート**］：受信トラフィックを調査する対象となるサーバの IP アドレスとポートです。

［**送信トラフィック**］の［**サーバ**］：送信トラフィックを調査する対象となるサーバの IP アドレスです。

［**記録先アクション**］：スクリプトを作成して挿入する対象となるセクションです。反復を使用する場合は，**Actions** セクションを指定します。

［**フィルタ オプション**］：スクリプトに含める IP アドレスまたは除外する IP アドレスを指定できるフィルタ・インタフェースが開きます。詳細については，次を参照してください。

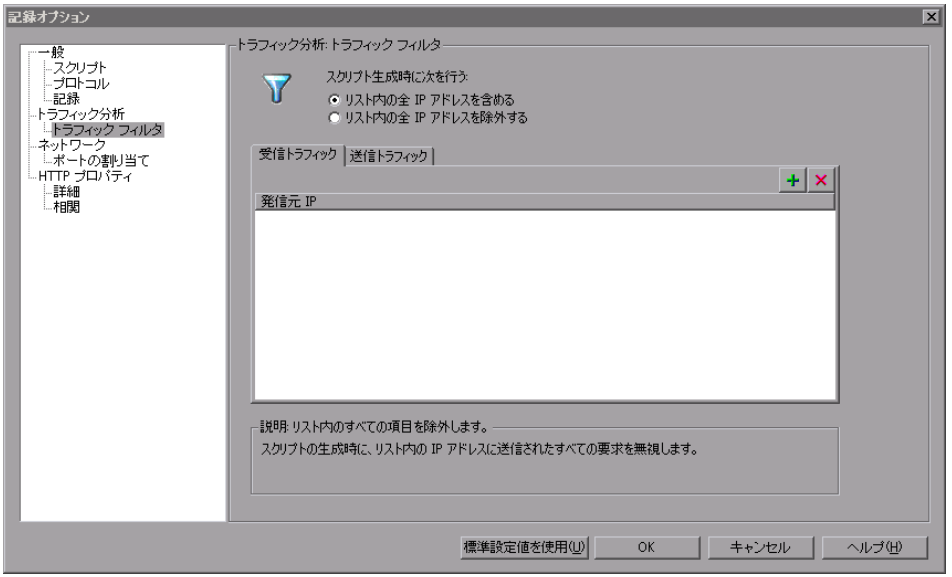
［**SSL 設定**］：SSL 証明書を追加して，必要な資格情報を使用してセキュア・サーバからのトラフィックを分析できるようにします。詳細については，412 ページ「SSL 証明書の指定」を参照してください。

受信フィルタまたは送信フィルタの選択

サーバの IP アドレスとポートを指定することで，サーバが受信する要求またはサーバが送信する要求を絞り込むフィルタを提供できます。

また，キャプチャ・ファイルを VuGen にロードする前に，外部ツールを使用してフィルタを適用できます。この場合，追加のフィルタリングは必要ありません。

要求にフィルタを適用するには，関連するホストの IP アドレスを選択します。包含フィルタまたは除外フィルタを適用できます。つまり，リスト内の IP アドレスのみを含めることも，リスト内の IP アドレスを除くすべての IP アドレスを含めることもできます。



トラフィック・ファイルにフィルタを適用するには，次の手順を実行します。

1 トラフィック・フィルタの記録オプションを開きます。［トラフィック情報を指定してください］ステップで［**フィルタ オプション**］をクリックするか，［**ツール**］＞［**記録オプション**］を選択します。［**トラフィック分析：トラフィック フィルタ**］ノードを選択します。

2 ［**リスト内の全 IP アドレスを含める**］または［**リスト内の全 IP アドレスを除外する**］のいずれかのフィルタ・オプションを選択します。

3 スクリプトのタイプ（**受信トラフィック**または**送信トラフィック**）に対応するタブを選択します。

4 ホストをリストに追加します。

このリストにホストを追加するには，［**追加**］ボタンをクリックします。リストに追加するサーバの IP アドレスを指定します。受信トラフィックの場合，含めるかまたは除外するサーバのポートを指定します。［**OK**］をクリックして設定を適用します。

エントリを削除するには，［**削除**］ボタンをクリックします。

スクリプトが作成されたら，フィルタを変更してスクリプトを再生成できます。キャプチャ・ファイルを分析しなおす必要はありません。

SSL 証明書の指定

セキュア・サーバからのトラフィックを分析するには，サーバの秘密鍵を含む証明書を指定する必要があります。

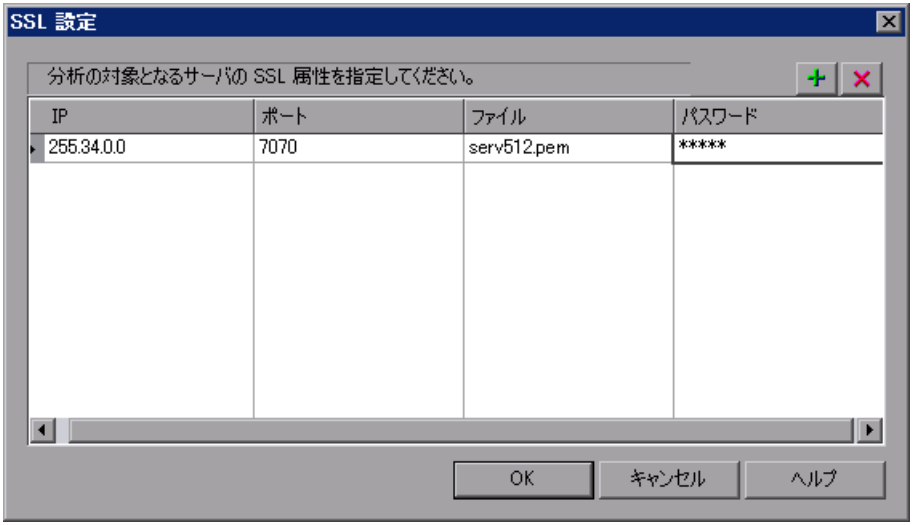
トラフィックが SSL により暗号化されている場合，復号するために証明書ファイルとパスワードを指定する必要があります。複数のサーバからのトラフィックをスクリプトに反映する場合は，SSL を使用する IP アドレスごとに個別の証明書とパスワードを指定する必要があります。

SSL 証明書を指定するには，次の手順を実行します。

1 ［トラフィック情報を指定してください］画面で，［**SSL 設定**］をクリックします。

2 証明書をリストに追加します。

このリストに証明書を追加するには，［**追加**］ボタンをクリックします。IP アドレス，ポート，証明書ファイル（拡張子は **.pem**）のパス，および証明書のパスワードを指定します。**pem** ファイルに秘密鍵が含まれていることを確認します。証明書の取得方法がわからない場合は，システム管理者に問い合わせてください。



リストからエントリを削除するには，［**削除**］ボタンをクリックします。

3 追加する証明書ごとに，上記の手順を繰り返します。

4 ［**OK**］をクリックし，ダイアログ・ボックスを閉じます。

第 4 部

Java 言語プロトコルでの作業

Java 言語に関連するプロトコルには，RMI-Java，CORBA-Java，EJB，および Jacada タイプがあります。プロトコルの詳細については，それぞれのプロトコルを説明する項を参照してください。「Java 言語プロトコルでの作業」では，すべての Java 仮想ユーザに共通して適用される項目を説明します。

第 26 章

Java 言語仮想ユーザ・スクリプトの記録

VuGen では，Java で書かれた，CORBA，RMI，EJB または Jacada などのプロトコルを使うアプリケーションまたはアプレットを記録できます。VuGen のナビゲーション・ツールを使用して，スクリプトに任意のメソッドを追加することもできます。

本章では，次の項目について説明します。

- 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。スクリプトが正常に機能することを確認したら，スクリプトを LoadRunner シナリオ，Tuning Module 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 38 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，760 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 38 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 27 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 38 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，421 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と置き換えます。文字列の全体または一部をパラメータ化できます。複数の引数を持つ関数には，複数のパラメータを定義できます。詳細については，第 9 章「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 29 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 15 章「スタンドアロン・モードでの仮想ユーザ・スクリプトの実行」を参照してください。

記録手順の詳細については，各仮想ユーザ・タイプの章を参照してください。

Java 言語仮想ユーザ・スクリプトについて

セッションを記録すると，VuGen によってサーバに対するすべての呼び出しが記録され，関数を含んだスクリプトが生成されます。これらの関数は，アプリケーションまたはアプレットにおけるユーザのすべてのアクションを表します。スクリプトにはプロパティの設定やネーミング・サービスの初期化（JNDI）など，適切な再生に必要な追加コードも含まれています。

記録されたスクリプトは，次の 3 つのセクションで構成されています。

- インポート
- コード
- 変数

インポート・セクションはスクリプトの先頭にあります。ここにはスクリプトのコンパイルに必要なすべてのパッケージへの参照が含まれます。**コード**・セクションには，Actions クラスと，**init**，**actions**，および **end** メソッド内に記録されたコードが含まれます。**end** メソッドの後の**変数**セクションには，コードで使用される変数のすべての型宣言が含まれます。

記録終了後，スクリプト内の関数に変更を加えたり，Java または Mercury の関数を追加してスクリプトを拡張したりできます。Java 仮想ユーザをスレッドとして実行する場合，スクリプトに追加する Java コードはスレッドセーフで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。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，619 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

この項は，Jacada タイプのスクリプトには適用されません。

Java 仮想ユーザ・スクリプトを作成または記録するときに，メソッドまたはクラスが保護されているクラスのメソッドを使用する必要が生じる場合があります。そのようなスクリプトをコンパイルすると，メソッドにアクセスできないことを示すコンパイル・エラーを受け取ることになります。

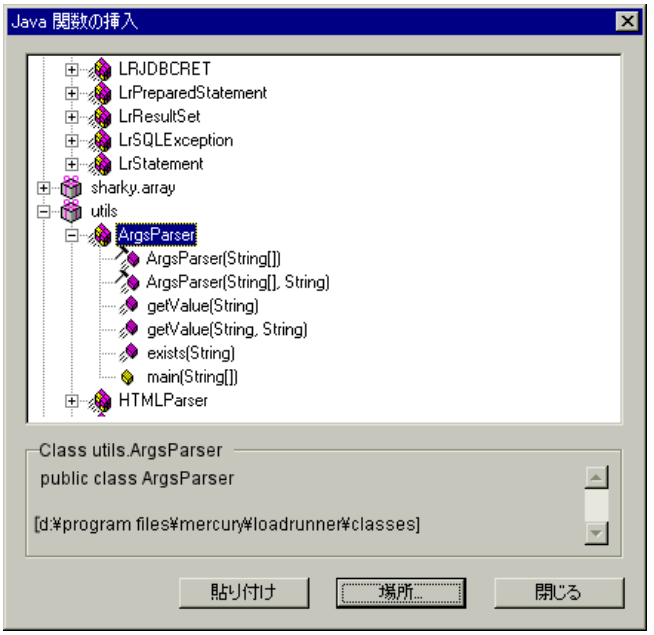
保護されているメソッドを仮想ユーザで使用するには，必要なメソッドのパッケージにその仮想ユーザを追加します。スクリプトの先頭に次の行を追加します。

```
package a.b.c;
```

ここで，**a.b.c** はディレクトリ階層を表します。VuGen はユーザ・ディレクトリにディレクトリ階層 a/b/c を作成し，そこで **Actions.java** ファイルをコンパイルして，パッケージの一部にします。**package** ステートメントは記録されません。手作業で挿入する必要があります。

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，423 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 ()

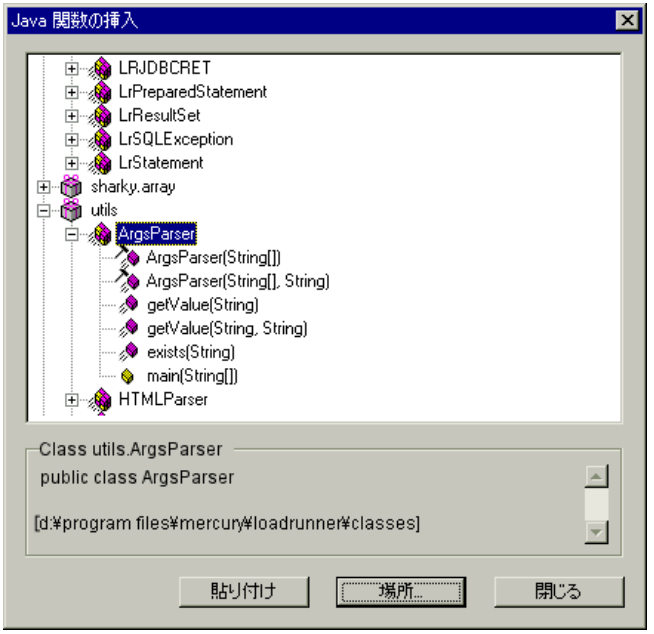
Java メソッドの手作業による挿入

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，425 ページ「スクリプト生成の設定」を参照してください。

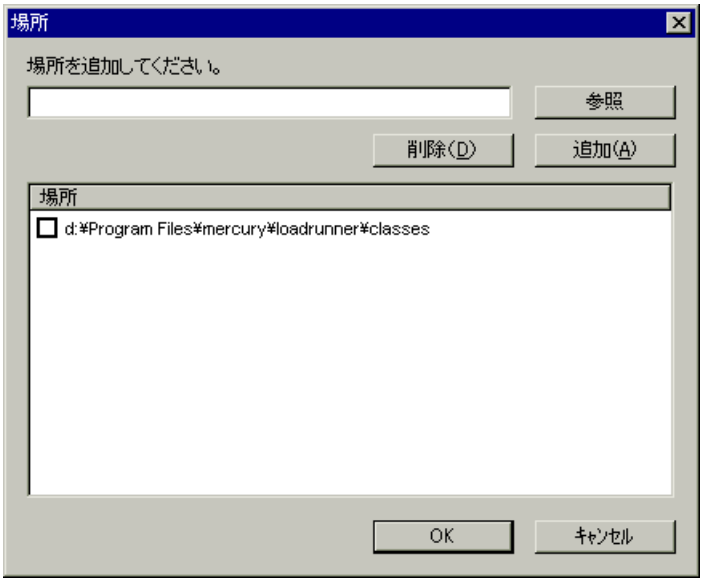
Java 関数を挿入するには，次の手順を実行します。

1 スクリプトの中で，挿入を行う場所をクリックします。関数の貼り付けを行うと，VuGen によってカーソルの位置に関数が貼り付けられます。

2 ［**挿入**］＞［**Java 関数の挿入**］を選択します。［Java 関数の挿入］ダイアログ・ボックスが表示されます。



3 [**場所**] をクリックします。[場所] ダイアログ・ボックスが表示されます。標準設定では，CLASSPATH 環境変数に定義されているパスの一覧が表示されます。



4 [**参照**] をクリックして別のパスまたはアーカイブをリストに追加します。パスを追加するには，[**参照**] > [**フォルダ**] を選択します。アーカイブ（**jar** または **zip**）を追加するには，[**参照**] > [**ファイル**] を選択します。フォルダまたはファイルを選択すると，VuGen によって [**場所を追加してください。**] ボックスにその名前が挿入されます。

5 [**追加**] をクリックして，リストに項目を追加します。

6 追加するパスまたはアーカイブごとに手順 4 と 5 を繰り返します。

7 リスト項目の左にあるチェック・ボックスを選択するかクリアします。選択した項目のメンバのリストが，Java クラス・ナビゲータに表示されます。

8 [**OK**] をクリックして [場所] ダイアログ・ボックスを閉じると，使用可能なパッケージが表示されます。

9 ナビゲータの各項目の左にあるプラスとマイナスの記号をクリックして，ツリーの分岐の表示と非表示を切り替えます。

10 オブジェクトを選択し，［**貼り付け**］をクリック 7 します。VuGen によってスクリプトのカーソルの位置にオブジェクトが挿入されます。1 つのクラスのすべてのメソッドをスクリプトに貼り付けるには，そのクラスを選択して［**貼り付け**］をクリックします。

11 使用するすべてのメソッドとクラスについて，手順 10 を繰り返します。

12 メソッドのパラメータを変更します。スクリプト生成の設定で **DefaultValues** を **true** に設定しておくと，VuGen によって挿入される標準設定値を使用できます。**DefaultValues** を **false** に設定すると，スクリプトに挿入するすべてのメソッドについてパラメータを追加する必要があります。

次に戻り値を変更します。たとえば，スクリプトで「(String)=LavaVersion.getVersionId();」というステートメントが生成された場合，**(String)** を文字列型変数に置き換えます。

13 import ステートメントや仮想ユーザ API Java 関数（第 38 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。

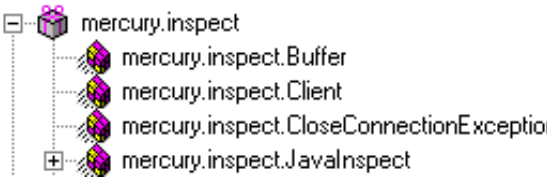
- クラス名のパス
- 自動トランザクション
- 標準のパラメータ値
- クラスの貼り付け

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
```

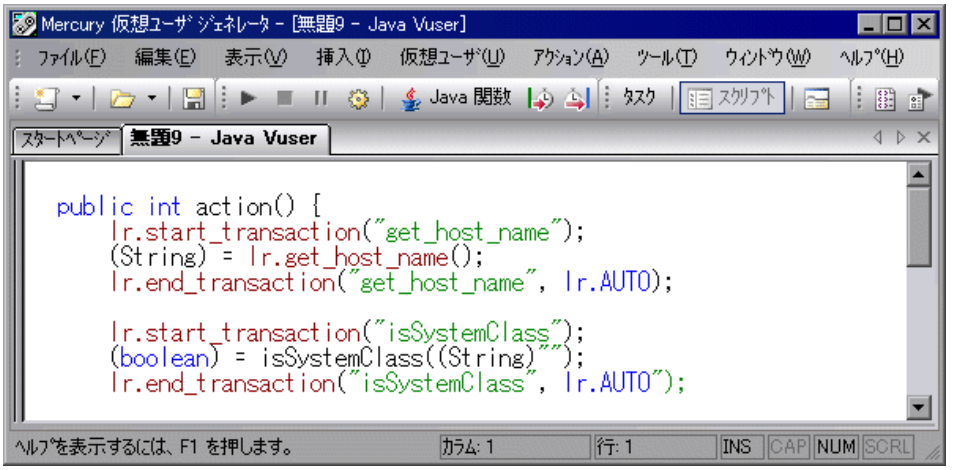
クラス名のパス

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 Vuser トランザクションが作成されます。このオプションを有効にすると，VuGen によってすべての Java メソッドが自動的に **lr.start_transaction** 関数と **lr.end_transaction** 関数で囲まれます。これにより，各メソッドのパフォーマンスを個別に追跡できます。標準設定では，このオプションは無効になっています。



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 27 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，次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 62 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。

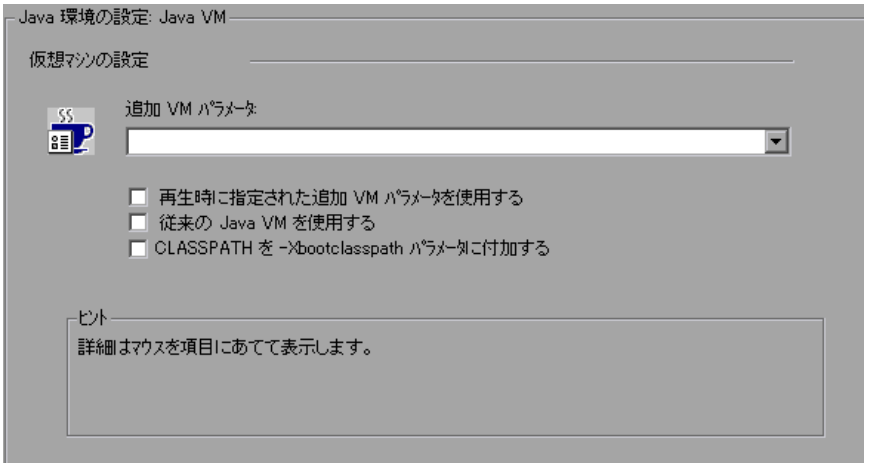
次の項目について記録オプションの設定が可能です。

- 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。

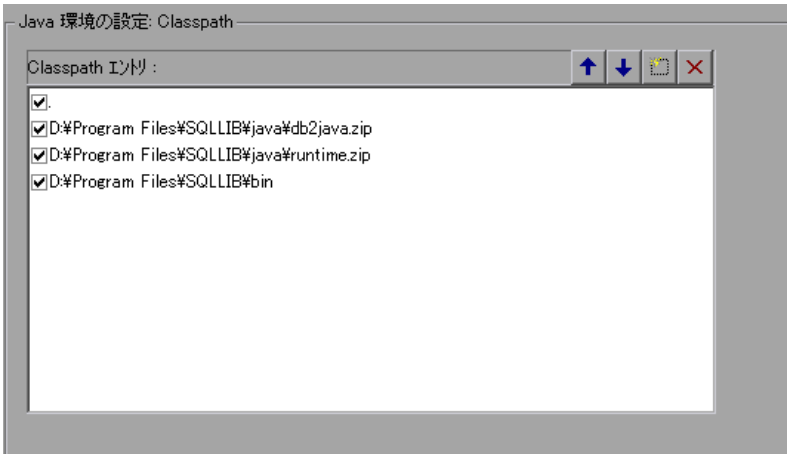
5 クラスパスを **Xbootclasspath** の前に追加する（つまり，文字列を前置する）には，［**CLASSPATH を -Xbootclasspath パラメータに付加する**］チェック・ボックスを選択します。

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 環境の設定**：**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 リストにクラスパスを追加するには，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。

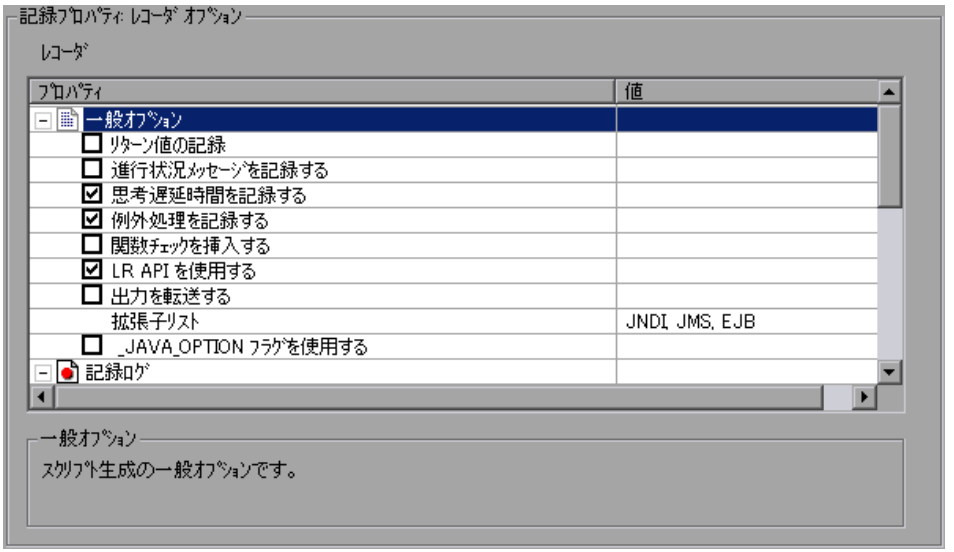
6 一覧の中でクラスパス・エントリを上方向に移動するには，エントリを選択し，上向き矢印ボタンをクリックします。

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_OPTION 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［記録ログ］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，［**Regular**］，［**Folded Serialized Object**］，または［**Unfolded Serialized Object**］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［**Regular**］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，［**記録のプロパティ：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，452 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。

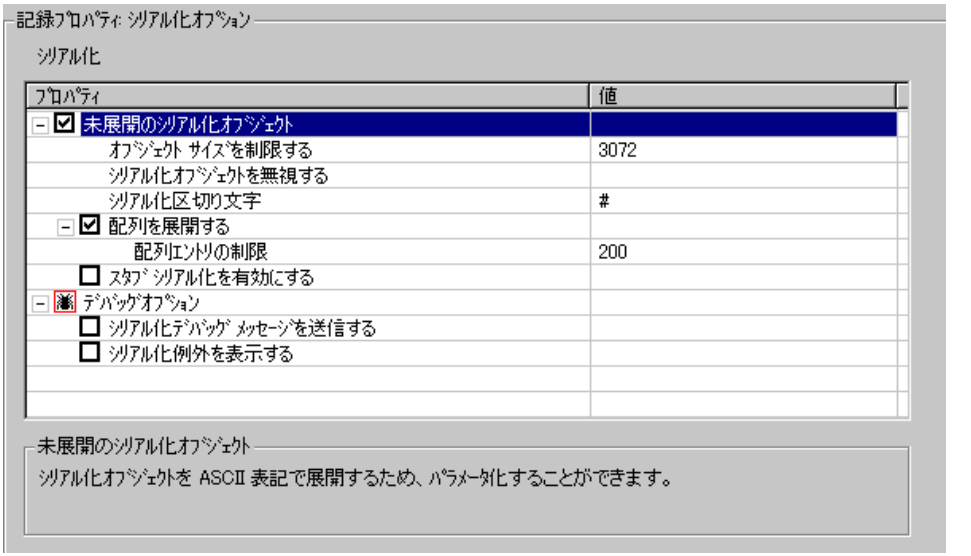
- ［**シリアル化オブジェクトを無視する**］：記録されたスクリプトの中で遭遇しても展開しないシリアル化オブジェクトを示します。オブジェクトはカンマで区切ります。
- ［**シリアル化区切り文字**］：ASCII 形式のオブジェクトで要素を区切る区切り文字を示します。VuGen では，これらの区切り文字に囲まれた文字列のみがパラメータ化されます。標準設定は「#」です。
- ［**配列を展開する**］：シリアル化されたオブジェクトの配列の要素を ASCII 形式に展開します。このオプションを無効にし，オブジェクトに配列が含まれている場合，オブジェクトは展開されません。標準設定では，このオプショ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，［**記録のプロパティ：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

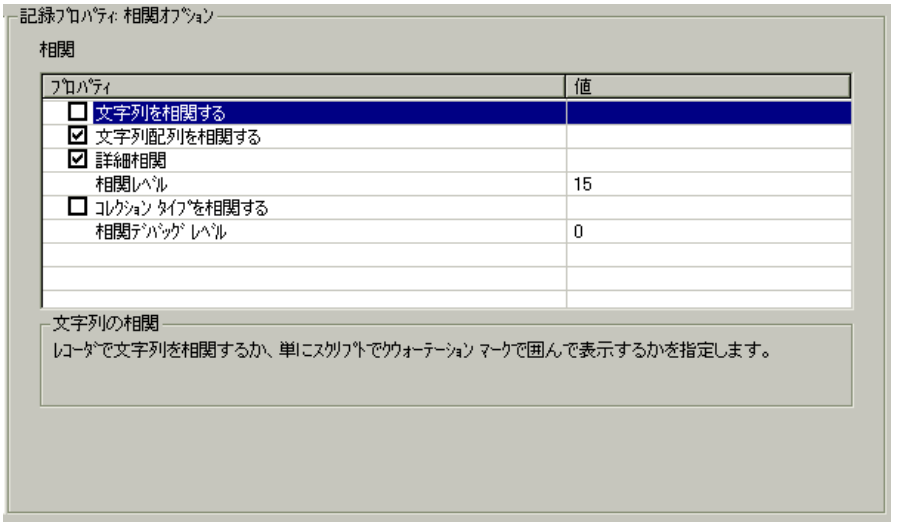
［**相関オプション**］では，VuGen で自動相関を行うかどうかを指定し，自動相関の範囲を制御します。相関については，第 28 章「Java スクリプトの相関」を参照してください。次のオプションを使用できます。

- ［**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（標準設定では有効）。

- ［**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，［**記録のプロパティ：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，［**クラスのダンプ**］，［**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 トレースを有効にする**］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，［**クラスのダンプ**］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［クラスのダンプ］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**]：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。これは，記録の重複を検出するのに有用です（標準では無効）。
- [**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。これは，マルチ・スレッド・アプリケーションを識別するのに便利です（標準設定では無効）。

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。[コンソール]，個別の［ファイル］，または［デバッグ］ファイルが選択できます。標準設定は［コンソール］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

- ［**クラスのダンプ**］：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，［**記録のプロパティ：デバッグ オプション**］ノードを選択します。

2 必要なオプションを有効にするか，値が必要なオプションに値を入力します。

3 すべてのオプションを標準の値に設定するには，［**標準設定値を使用**］をクリックします。

4 ［**OK**］をクリックしてダイアログ・ボックスを閉じ，記録を開始します。

CORBA オプション

次のオプションは Corba-Java プロトコルを対象としています。このオプションでは，CORBA 固有の記録プロパティといくつかのコールバック・オプションを設定できます。次のオプションが使用できます。

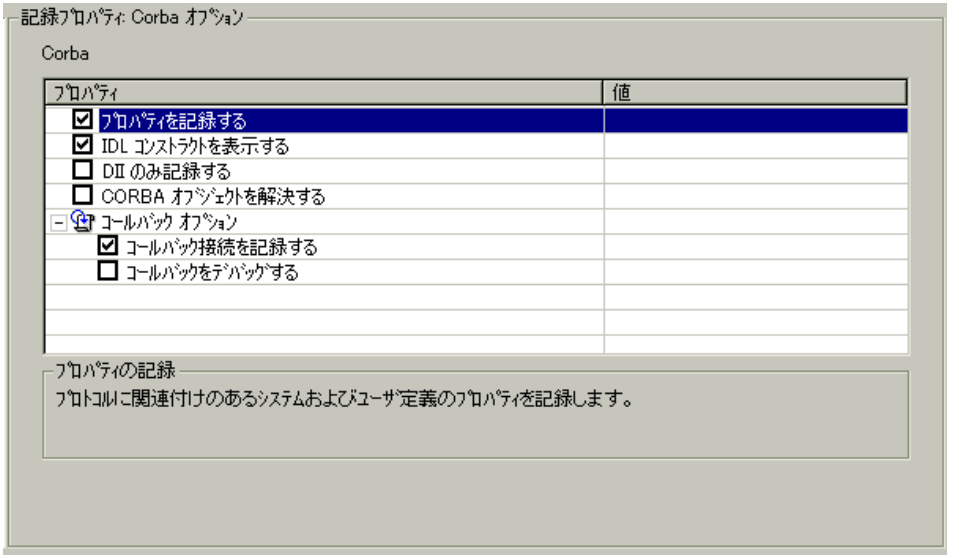
- ［**プロパティを記録する**］：プロトコルに関連するシステム・プロパティとユーザ定義のプロパティを記録します。標準設定では，このオプションは有効になっています。
- ［**IDL コンストラクトを表示する**］：パラメータを CORBA の呼び出しに対して渡すために使用される，インタフェース定義言語（IDL）の構成要素を表示します。標準設定では，このオプションは有効になっています。
- ［**Dll のみ記録する**］：DLL レベルのみで記録するよう VuGen に指示します。標準設定では，このオプションは無効になっています。
- ［**CORBA オブジェクトを解決する**］：相関によって，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，このオプションは無効になっています。

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，このオプションは無効になっています。
- ［**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，このオプションは無効になっています。

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，［**記録のプロパティ：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 28 章

Java スクリプトの相関

VuGen の相関機能を使うと，あるステートメントの結果を別のステートメントへの入力として使用して，Java 仮想ユーザ関数同士をリンクさせることができます。

本章では，次の項目について説明します。

- 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-Java または RMI-Java 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，452 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 27 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，［**未展開のシリアル化オブジェクト**］チェック・ボックスをクリアすると，**lr.deserialize** メソッドに引数を渡すことによって，シリアル化メカニズムを制御できます。最初の引数は整数で，ロードするバイナリ・ファイルの数を示しています。2 つ目の整数はブール値です。

true	VuGen のシリアル化メカニズムを使用します。
false	標準の Java シリアル化メカニズムを使用します。

次のコードは，シリアル化メカニズムが有効になっている状態で生成されたスクリプトを示します。

```
public class Actions{

 // Public 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。区切り文字の変更は，［記録オプション］の［**シリアル化オプション**］パネルで行います。区切り文字を使用するのは，再生時の文字列の解析処理を高速化するためです。

文字列を変更するときには，次のルールを守る必要があります。

- 行の順序は変更できません。パーサは，メンバ名ではなく，値を 1 つずつ読み取ります。
- 2 つの区切り文字の間にある値だけを変更できます。
- オブジェクト参照は変更できません。オブジェクト参照は，内部の一貫性を保つためだけに示されています。

- 「_NULL_」が値（Java の null 定数）として示されていることがあります。これは文字列型の値にのみ置換できます。
- オブジェクトはスクリプト内の任意の場所でシリアル化解除できます。たとえば，**init** メソッド内のすべてのオブジェクトをシリアル化解除し，**Action** メソッドの値を使用することができます。
- オブジェクトの内部的な一貫性を保ちます。たとえば，ベクトル・オブジェクトに要素数を示すメンバ（**element count**）があり，要素を追加した場合は，要素数を変更する必要があります。

  次のコードでは，ベクトルに，2 つの要素が含まれています。

```
public class Actions{

 // Public 関数：init
    public int init() throws Throwable {
        _string = "java.util.Vector CURRENTOBJECT = {" +
                  "int capacityIncrement = "#0#" +
                  "int elementCount = #2#" +
                  "java/lang/Object elementData[] = {" +
                       "elementData[0] = #First Element#" +
                       "elementData[1] = #Second Element#" +
                       "elementData[2] = _NULL_" +
                        ....
                       "elementData[9] = _NULL_" +

                  "}" +
        "}";
        _vector = (java.util.Vector)lr.deserialize(_string,0);
        map.set_vector(_vector);
    }
:
}
```

次の例では，ベクトルの要素の 1 つを「_NULL_」から「Third element」に変更しています。新しい要素の追加に伴って，「elementCount」メンバを「3」に変更しています。

```
public class Actions{

 // Public 関数：init
    public int init() throws Throwable {
        _string = "java.util.Vector CURRENTOBJECT = {" +
                "int capacityIncrement = "#0#" +
                "int elementCount = #3#" +
                "java/lang/Object elementData[] = {" +
                    "elementData[0] = #First Element#" +
                    "elementData[1] = #Second Element#" +
                    "elementData[2] = #Third Element#" +
                     ....
                    "elementData[9] = _NULL_" +
                "}" +
        "}";
        _vector = (java.util.Vector)lr.deserialize(_string,0);
        map.set_vector(_vector);
    }
:
}
```

オブジェクトを ASCII 表現に展開するシリアル化メカニズムは複雑なため，記録中に大きなオブジェクトを開くと，スクリプトの生成に時間がかかることがあります。この時間を短縮するために，シリアル化メカニズムのパフォーマンスを向上させるフラグを指定できます。

スクリプトに **lr.deserialize** を追加するときは，**action** メソッドではなく，**init** メソッドに追加することをお勧めします。VuGen によって文字列が一度だけシリアル化解除されればよいので，パフォーマンスが向上します。lr.deserialize が **action** メソッドにあると，VuGen では反復処理が行われるたびに文字列のシリアル化解除が行われることになります。

［記録オプション］の［**シリアル化オプション**］パネルでは，次のオプションを設定できます。

- ［シリアル化区切り文字］
- ［未展開のシリアル化オブジェクト］
- ［配列を展開する］
- ［配列エントリの制限］
- ［シリアル化オブジェクトを無視する］

記録オプションの詳細については，第 27 章「Java 記録オプションの設定」を参照してください。

第 29 章

Java 実行環境の設定

Java 仮想ユーザ・スクリプトを記録した後で，Java 仮想マシンの実行環境を設定できます。

本章では，次の項目について説明します。

- 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。

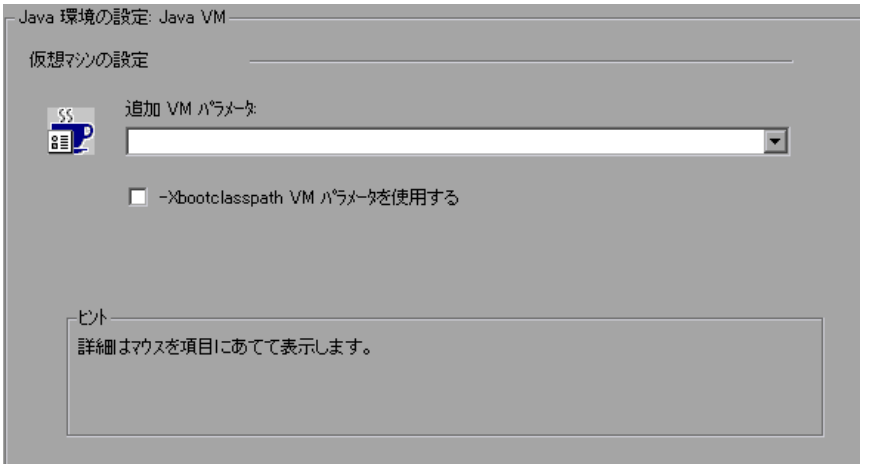
Java 関連の実行環境の設定は，［実行環境設定］ダイアログ・ボックスの［**Java VM**］オプションで行います。

［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［**実行環境設定の編集**］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 13 章「実行環境の設定」を参照してください。

JVM 実行環境の設定

［Java VM］セクションで，Java 仮想マシンの設定情報を入力します。



以下の項目の設定が可能です。

- ［**追加 VM パラメータ**］：仮想マシンで使用する任意のパラメータを入力します。
- ［**-Xbootclasspath VM パラメータを使用する**］：仮想ユーザを実行する際，自動的に **Xbootclasspath** 変数が設定されます。このダイアログ・ボックスを使って，**Xbootclasspath** で定義されているもの以外のパラメータを指定します。記録用に追加で VM パラメータを指定した場合，パラメータが保存され，再生時に使用されるように設定します。

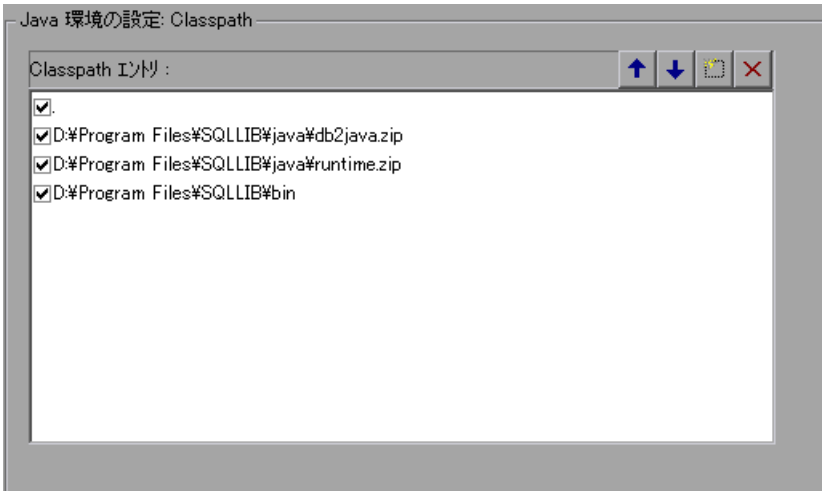
Java VM の実行環境を設定するには，次の手順を実行します。

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**］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。

［**クラスパスの追加**］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，対象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・エントリをリストの下方へ移動するには，対象エントリを選択して下向き矢印ボタンをクリックします。

6 クラスパス・エントリをリストの上方へ移動するには，対象エントリを選択して上向きの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・ボックスを閉じます。

第 5 部

アプリケーション配備ソリューション・プロトコル

第 30 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 31 章「LoadRunner Citrix エージェントの使用方法」を参照してください。スクリプトの作成に役立つヒントについては，502 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・スクリプトの開発の概要
- 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

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 4 章「VuGen を使った記録」を参照）。マルチ・プロトコル・モードでは，VuGen は記録中に Citrix と Web プロトコルの両方から関数を生成します。

次に示す例では，**ctrx_mouse_click** によってマウスの左クリックをシミュレートしています。

```
ctrx_mouse_click(44, 318, LEFT_BUTTON, 0, CTRX_LAST);
```

構文およびパラメータの詳細については，「**オンライン関数リファレンス**」（[**ヘルプ**] > [**関数リファレンス**]）を参照してください。

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・スクリプトの開発の概要

本項では，VuGen を使って Citrix ICA 仮想ユーザ・スクリプトを開発する工程の概要を説明します。このほか，502 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 スクリプトを開発するには，次の手順を実行します。

1 **クライアントとサーバの設定が正しいことを確認します。**

これらの設定の詳細については，469 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。記録中にビットマップとテキストの同期化を挿入します。詳細については，487 ページ「再生の同期化」を参照してください。

記録全般については，第 4 章「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 8 章「仮想ユーザ・スクリプトの拡張」を参照してください。

4 **パラメータを定義します（任意）。**

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 9 章「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，481 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，482 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の詳細については，第 13 章「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，502 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 15 章「スタンドアロン・モードでの仮想ユーザ・スクリプトの実行」を参照してください。

スクリプトを作成した後，スクリプトを自分の環境（LoadRunner シナリオ，Performance Center 負荷テスト，または Business Process Monitor プロファイル）に統合します。詳細については，**『Mercury LoadRunner コントローラ・ユーザーズ・ガイド』**，**Performance Center** のマニュアル，または **Mercury Business Availability Center** 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，XP，3，または 4）がインストールされていることを確認します。サーバのバージョンを調べるには，サーバのコンソール・ツールバーにある［**Citrix コネクション構成ツール**］を選択して，［**ヘルプ**］＞［**バージョン情報**］を選択します。
- ［**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，前回接続を解除したときと同じ作業領域が表示されます。新しいテスト実行のたびに初期状態の作業領域を使用できるようにすることが望まれます。

  テストのたびに初期状態の作業領域を表示するには，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が切断するたびにクライアントから接続を解除することによって，接続をリセットするようにします。
- MetaFrame 1.8 または XP サーバ

➤ MetaFrame 3 および 4 サーバ

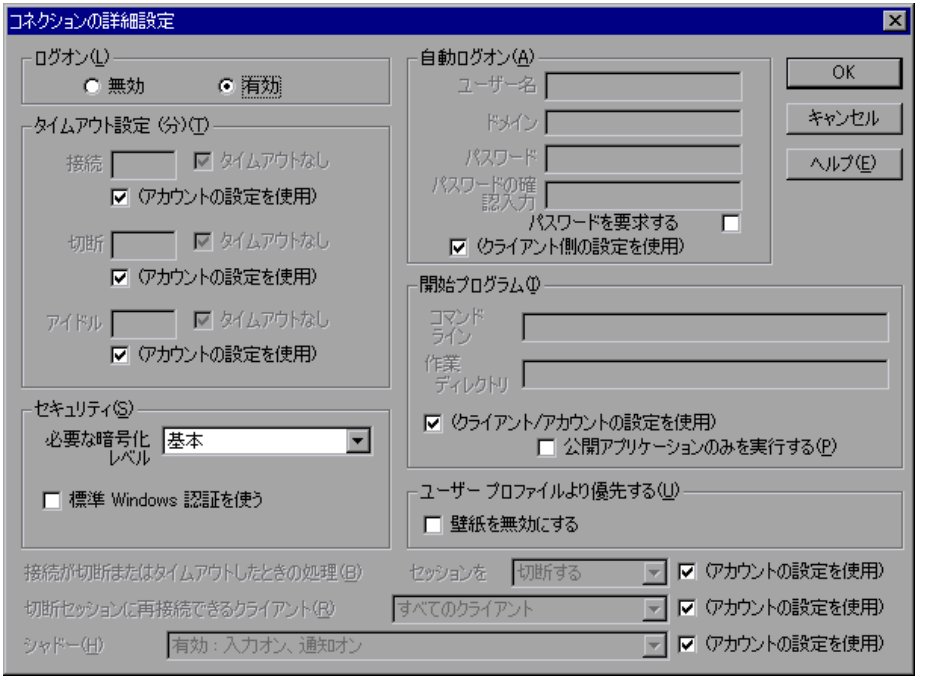
MetaFrame 1.8 または XP サーバ

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 コネクション構成ツール**］を選択します。

2 ica-tcp 接続名をダブルクリックします。［コネクションの編集］ダイアログ・ボックスが開きます。

3 ［**詳細**］ボタンをクリックします。［コネクションの詳細設定］ダイアログ・ボックスが開きます。



4 このダイアログの一番下のセクションにある［**接続が切断またはタイムアウトしたときの処理**］リスト・ボックス横の［**アカウントの設定を使用**］チェック・ボックスをクリアします。リスト・ボックスのエントリを「**リセットする**」に変更します。

5 ［**OK**］クリックします。

MetaFrame 3 および 4 サーバ

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 Select the ica-tcp 接続名を選択し，［**コネクション**］＞［**編集**］を選択します。あるいは，接続をダブルクリックします。［コネクションの編集］ダイアログ・ボックスが開きます。

3 ［**詳細**］ボタンをクリックします。［コネクションの詳細設定］ダイアログ・ボックスが開きます。

4 このダイアログの一番下のセクションにある［**接続が切断またはタイムアウトしたときの処理**］リスト・ボックス横の［**アカウントの設定を使用**］チェック・ボックスをクリアします。リスト・ボックスのエントリを「**リセットする**」に変更します。

5 ［**OK**］クリックします。

記録に関するヒント

スクリプトを記録するときは，効果的なスクリプトを作成できるように必ず次のガイドラインに従ってください。

シングル・プロトコル・スクリプトとマルチ・プロトコル・スクリプト

スクリプトを新規に作成するときは，シングル・プロトコルかマルチ・プロトコルのスクリプトを作成できます。簡単な Citrix ICA セッションを記録する場合は，シングル・プロトコル・スクリプトを使用します。ただし，Nfuse Web アクセス・セッションの記録時には，Citrix ICA と Web（HTML/HTTP）を対象とするマルチ・プロトコル・スクリプトを作成する必要があります。これによって，両方のプロトコルを記録できるようになります。詳細については，第 4 章「VuGen を使った記録」を参照してください。

適切なセクションに記録する

接続処理は「**vuser_init**」セクションに，終了処理は「**vuser_end**」セクションに記録します。これにより，接続もしくは接続解除時に反復が実行されないようになります。セクションへの記録の詳細については，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。たとえば，［**スタート**］＞［**プログラム**］＞［**Microsoft Word**］の場合は，「**プログラム**」という語をクリックします。

ウィンドウのサイズを変更しない

VuGen は，セッションの記録時におけるウィンドウのサイズ変更をサポートしていますが，記録中はウィンドウを動かしたり，ウィンドウの大きさを変えたりしないことをお勧めします。ウィンドウのサイズまたは位置を変更するには，スクリプトのツリー・ビューの中で該当する［**ウィンドウで同期**］ステップをダブルクリックし，ウィンドウの座標を変更します。

解像度の設定に矛盾がないことを確かめる

ビットマップの同期化を確実に成功させるために，解像度の設定が一致していることを確認します。記録用マシンの ICA クライアントの設定，記録オプション，および実行環境の設定を確認します。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。解像度間に不一致があると，必要な調整を行うためにサーバのトラフィックが増大します。

手動の同期ポイントを追加する

記録中に，イベント（たとえばアプリケーションの開始など）を待機する場合には，［**ビットマップ取得時に同期化**］や［**テキストで同期化**］などの同期化ポイントを手作業で追加することをお勧めします。詳細については，487 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

Sync on Bitmap ステップの場合は，ウィンドウを XP 形式ではなく「クラシック」のウィンドウ形式で記録します。

ウィンドウの形式を「クラシック」に変更するには，次の手順を実行します。

1 デスクトップをクリックします。

2 右クリックして表示されるメニューから［**プロパティ**］を選択します。

3 ［テーマ］タブを選択します。

4 ［テーマ］ドロップ・ダウン・リストから，［**Windows クラシック**］を選択します。

5 ［**OK**］クリックします。

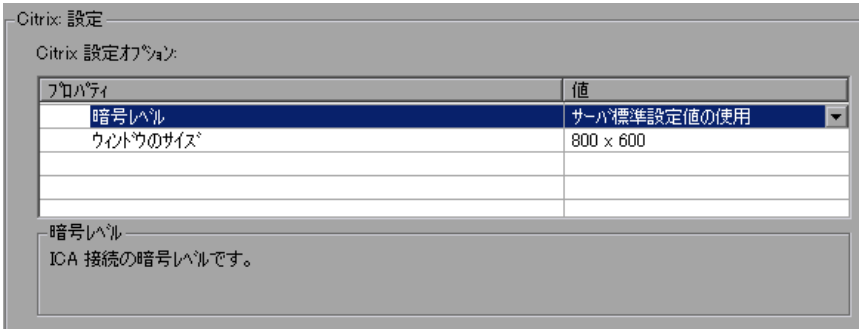
Citrix 記録オプションについて

次の項目について Citrix 記録オプションの設定が可能です。

- 設定記録オプション
- レコーダ記録オプション
- コード生成記録オプション
- ログイン記録オプション（シングル・プロトコルの Citrix ICA スクリプトの場合のみ）

設定記録オプション

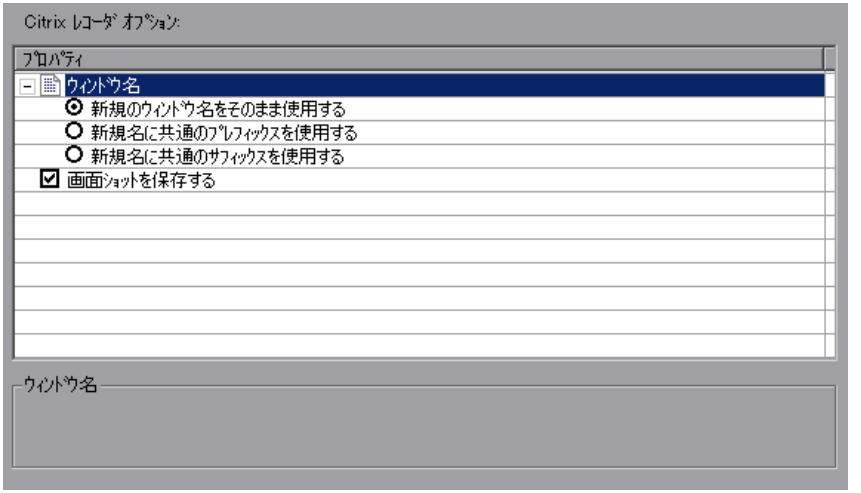
［**Citrix：設定**］記録オプションでは，記録セッション中の Citrix クライアントに対するウィンドウ・プロパティと暗号化設定を設定します。



- ［**暗号レベル**］：ICA 接続の暗号化レベル。［**Basic**］，［**128 Bit for Login Only**］，［**40 Bit**］，［**56 Bit**］，［**128 Bit**］，または［**Use Server Default**］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。

レコーダ記録オプション

［**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

アプリケーションには，記録中にアクティブなウィンドウの名前が変わるものがあります。スクリプトをそのまま再生しようとすると，仮想ユーザは最初のウィンドウ名を使用するため再生が失敗します。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。

たとえば，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。

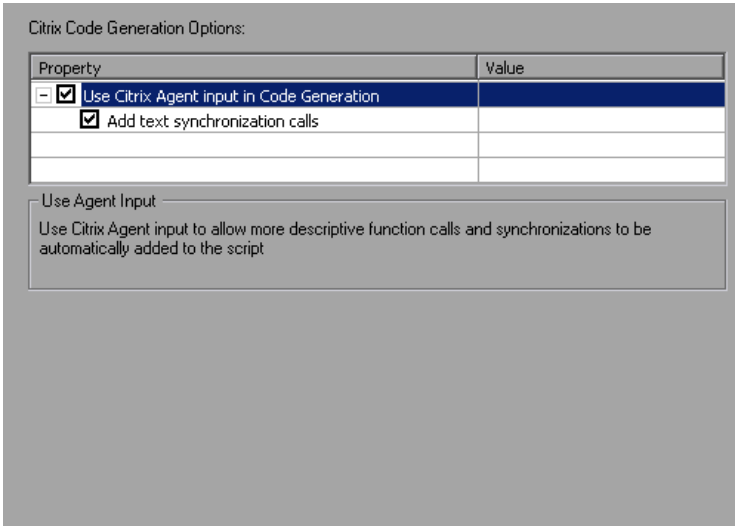
注：あるいは，記録後に実際のスクリプトでウィンドウ名を変更することもできます。スクリプト・ビューでウィンドウ名を見つけ，ウィンドウ名の先頭または末尾をワイルドカードの「*」で置き換えます。
ctrx_sync_on_window ("My Application*", ACTIVATE, ...CTRX_LAST);

スナップショット

［**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使用するディスク領域が増え，記録セッションの速度が低下します。

コード生成記録オプション

［**Citrix：コード生成**］：記録時に VuGen が情報をキャプチャする方法を設定できます。



➤ ［**Citrix エージェント入力コードを生成に使用する**］：Citrix エージェント入力を使用して，同期化関数が追加されたより説明的なスクリプトを生成します（標準設定では有効）。

➤ ［**テキスト同期化呼び出しを追加する**］：各マウス・クリックの前に，テキストの同期化を行う **Sync on Text** ステップを追加します（標準設定では無効）。

記録中に手動で追加したテキストの同期化ステップは，前述の設定の影響を受けません。前述のオプションを無効にしても，手動で追加したステップはスクリプトに現れます。記録時における **Sync on Text** ステップの追加の詳細については，490 ページ「手作業での同期化」を参照してください。

注：前述のオプションは，スクリプトの再生成にも使用可能です。たとえば，当初［**テキスト同期化呼び出しを追加する**］を無効にしてスクリプトを記録した場合でも，テキストの同期化が含まれるように記録後にスクリプトを再生成できます。スクリプトの再生成の詳細については，503 ページ「スクリプトの再生成」を参照してください。VuGen 8.1 以降を使用して，以前のバージョンの VuGen で作成したスクリプトを再生成した場合，そのスクリプトは以前のバージョンと互換性がなくなります。新しいスクリプトを作成した場合は互換性があります。

ログイン記録オプション

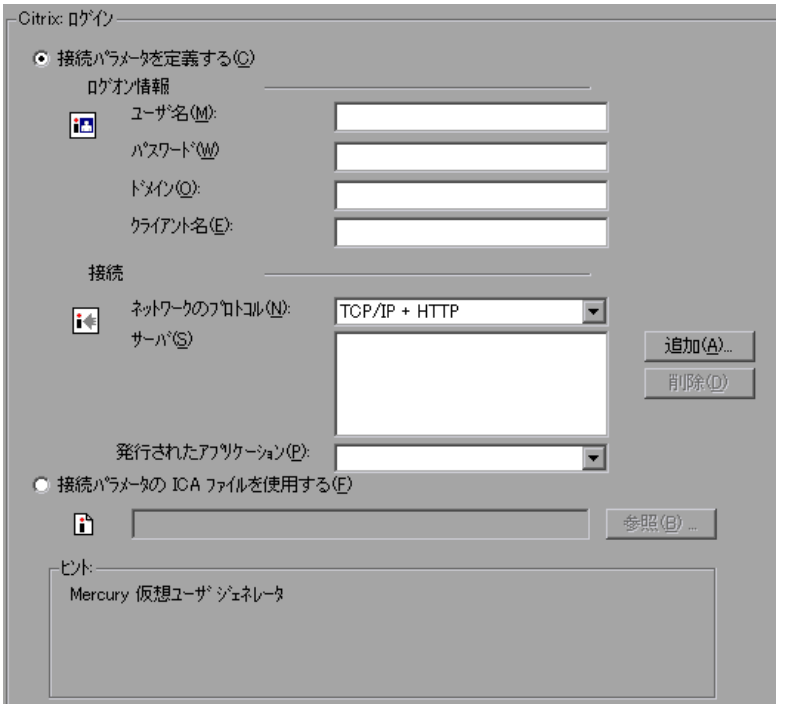
［**Citrix：ログイン**］記録オプションで，記録セッションに対する接続情報とログイン情報を設定します（NFUSE を使用する場合は，ログインが Web ページで行われるためログイン・オプションは使用できません）。

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", CTRX_LAST);
```

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。



Citrix セッションに関する次のユーザおよびサーバ情報を指定します。

［**ログオン情報**］：このセクションには次のログイン情報が含まれます。

- Citrix ユーザの**ユーザ名**。
- Citrix ユーザの**パスワード**。
- Citrix ユーザの**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）。

［**接続**］：このセクションには次のサーバ情報が含まれます。

- ［**ネットワークのプロトコル**］：TCP/IP か TCP/IP+HTTP のどちらかを選択します。ほとんどの Citrix サーバは TCP/IP をサポートしています。しかし，一部のサーバは，ある決まった HTTP ヘッダーを持つ TCP/IP のみ許可するように，管理者によって設定されています。通信上の問題が発生した場合は，TCP/IP + HTTP オプションを選択します。
- ［**サーバ**］：Citrix サーバの名前。新しいサーバをリストに追加するには，［**追加**］をクリックし，サーバ名（およびサーバの TCP/IP + HTTP 用ポート）を入力します。複数のサーバが適用されるのは［発行されたアプリケーション］を指定したときのみです。特定のアプリケーションのないデスクトップに接続しようとしている場合は，1 つのサーバのみが表示されます。
- ［**発行されたアプリケーション**］：Citrix サーバで認識される，**発行されたアプリケーション**の名前。ドロップダウン メニューに使用可能なアプリケーションのリストが含まれています。発行されたアプリケーションが指定されていない場合，VuGen はサーバのデスクトップを使用します。

  発行されたアプリケーションを指定しなければ，Citrix のロード・バランシングは機能しません。サーバのデスクトップにアクセスする場合にロード・バランシングを使用するには，デスクトップをサーバ・マシンで発行されたアプリケーションとして登録し，［**発行されたアプリケーション**］ドロップダウン・リストからこの名前を選択します。

接続パラメータの ICA ファイルの使用

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，［**接続パラメータに ICA ファイルを使用する**］を選択します。そしてその下のテキストボックスに，.ica ファイルのフル・パスを指定します。

ICA ファイルの形式の詳細については，495 ページ「ICA ファイルについて」，および Citrix の Web サイト www.citrix.com 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，［**接続パラメータに 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。これが標準設定です。［**接続**］情報と［**ログオン情報**］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：コード生成**］ノードを選択します。Citrix エージェントからの情報を使用して，より説明的なスクリプトを生成するには，ノードのオプションを有効にします。

5 ［**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。

6 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，［**画面ショットを保存する**］をクリアします。

7 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。［**一般：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。

8 ［**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。

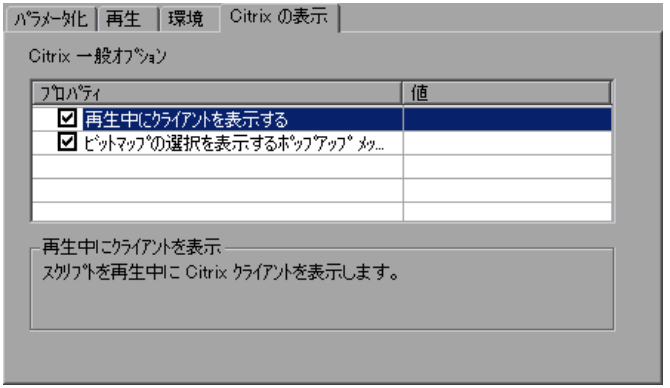
Citrix の表示設定

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［**ビットマップ同期を挿入**］または［**テキスト取得を挿入**］を選択すると，このメッセージが表示されます。

5 ［**OK**］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［**プロパティ**］を選択します。［**接続**］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の詳細については，第 13 章「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 14 章「ネットワーク実行環境の設定」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

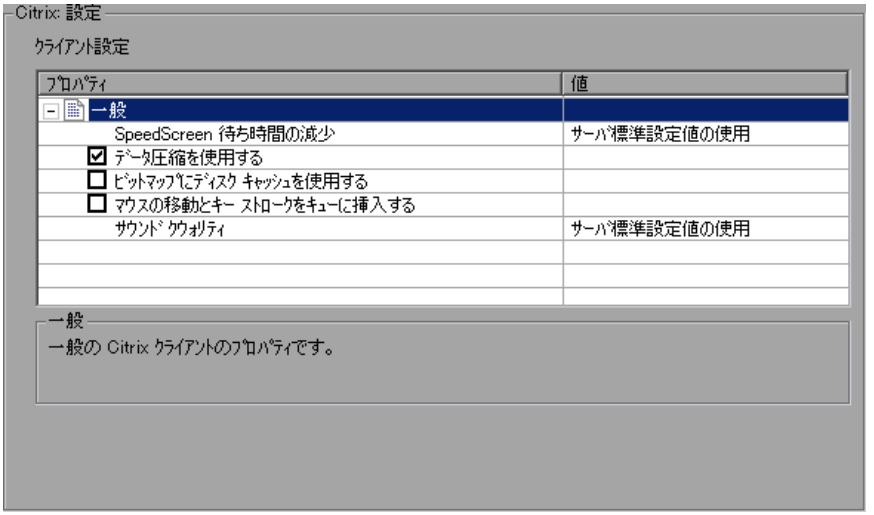
Citrix 設定実行環境の設定

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［**実行環境設定の編集**］ボタン をクリックするか，［**仮想ユーザ**］＞［**実行環境の設定**］を選択します。

2［**Citrix：設定**］ノードを選択します。［**一般**］プロパティを指定します。



必要なクライアント設定オプションを設定します。

- ［**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［**サーバの標準設定を使用する**］に設定し，マシンの標準設定を使用するようにします。
- ［**データ圧縮を使用する**］：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）。
- ［**ビットマップにディスク キャッシュを使用する**］：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）。

➤ [**マウスの移動とキー ストロークをキューに挿入する**]：マウスの動きとキーストロークのキューが作成され，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）。

➤ [**サウンド クウォリティ**]：サウンドのクウォリティを指定します。[**サーバの標準設定を使用する**]，[**サウンド オフ**]，[**高サウンド クウォリティ**]，[**中サウンド クウォリティ**]，および [**低サウンド 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・マシンの発行されたアプリケーションからサウンド・ファイルを再生できるようになります。

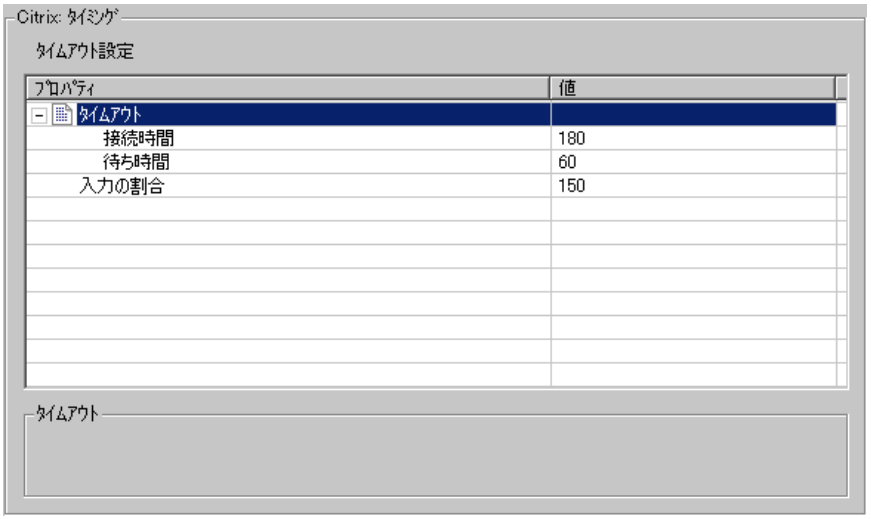
Citrix 時間設定実行環境の設定

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 [**実行環境設定の編集**] ボタン をクリックするか，[**仮想ユーザ**] > [**実行環境の設定**] を選択します。

2［**Citrix：タイミング**］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。

スクリプトの特定のセクションに対して待ち時間を設定するには，［**挿入**］＞［**ステップの追加**］ダイアログ・ボックスを使用して，**Set Waiting Time** ステップを挿入します。新しい待ち時間は，挿入したポイントから，スクリプトの最後，または次の **Set Waiting Time** ステップまでを対象として適用されます。

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。

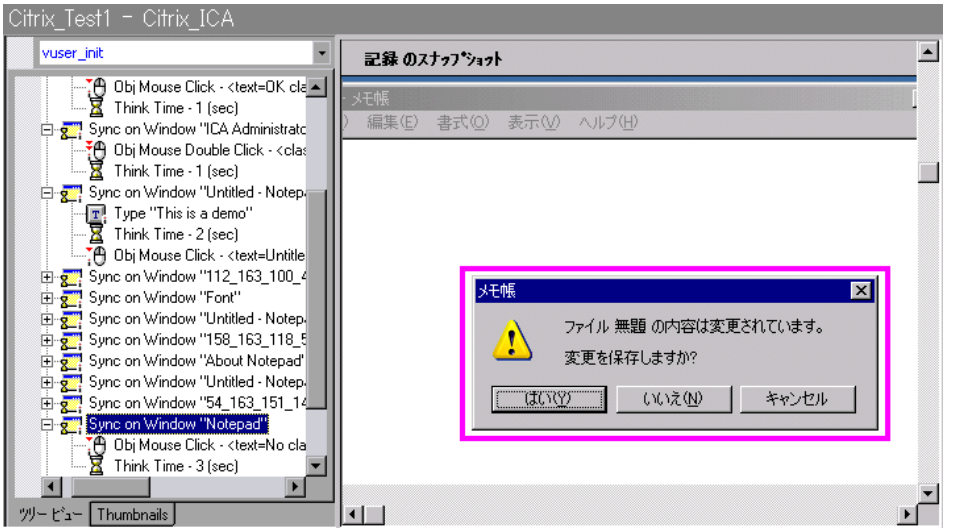
Citrix 仮想ユーザ・スクリプトの表示と変更

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 2 章「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。各ステップには，スナップショットが関連付けられています。スナップショットでは，クライアント・ウィンドウが表示されるほか，アクションの実行対象オブジェクトも強調表示されます。

- **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。さまざまなスクリプト・ビューの詳細については，18 ページ「仮想ユーザ・スクリプトの表示と変更」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，ビットマップの同期化を追加できます。Citrix エージェントが Citrix サーバ・マシンにインストールされている場合は，右クリック・メニューを使用してテキストとオブジェクトの同期化を追加することもできます。詳細については，第 31 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 右クリックしてコマンドのいずれか 1 つを選択します。ダイアログ・ボックスが開き，使用可能なプロパティが表示されます。

3 必要なプロパティを変更して［**OK**］をクリックします。VuGen によってステップがスクリプトに挿入されます。

再生の同期化

スクリプトの実行中，再生が正常に行われるようにするために，しばしばアクションを同期化する必要があります。同期化とは，スクリプトの中でイベントのタイミングを調整し，ウィンドウやオブジェクトが使用可能になるまで待ってからアクションを実行することを指します。たとえば，ウィンドウ内のボタンを押す前に特定のウィンドウがすでに開いているかどうかを知りたい場合があります。

VuGen は再生中のアクションの同期化を行う関数を自動的に生成します。同期化関数は手作業で追加することもできます。

自動同期化

記録中，仮想ユーザ・スクリプトの再生の同期化を支援するステップが VuGen によって自動的に生成されます。

- **Sync on Window**
- **Sync on Obj Info**
- **Sync on Text**

Sync on Window

Sync On Window ステップは，指定されたイベントが発生するまで，仮想ユーザに再生の再開を待機するよう指示するステップです。指定できるイベントは CREATE または ACTIVE です。Create イベントを指定した場合は，ウィンドウが作成されるまで待機します。Active イベントを指定した場合は，ウィンドウが作成されてアクティブになるまで（フォーカスを得るまで）待機します。VuGen は通常 CREATE イベントを待機する関数を生成します。しかし，次の命令がキーボード・イベントである場合，VuGen は ACTIVE イベントを待機する関数を生成します。

スクリプト・ビューでは，**Sync On Window** ステップに対応する関数呼び出しは **ctrx_sync_on_window** です。

Sync on Obj Info

Sync On Obj Info ステップは，指定されたオブジェクト・プロパティが発生するまで，仮想ユーザに再生の再開を待機するよう指示するステップです。使用可能な属性は，ENABLED，VISIBLE，FOCUSED，TEXT，CHECKED，LINES，および ITEM です。ENABLED，VISIBLE，FOCUSED，および CHECKED は，**true** または **false** を受け取ることができるブール属性です。その他の属性は，テキストまたは数値のオブジェクト値を必要とします。

このステップの主な目的は，オブジェクトを対象とするアクションを実行する前に，当該オブジェクトにフォーカスが当たるのを待機することです。

Citrix エージェントがインストールされていて，記録オプションの［**Citrix エージェント入力をコード生成に使用する**］オプションが有効になっている場合，VuGen は自動的に **Sync On Obj Info** ステップを生成します。標準設定では，この記録オプションは有効になっています。詳細については，476 ページ「コード生成記録オプション」を参照してください。

```
ctrx_sync_on_obj_info("Run=snapshot9", 120, 144, TEXT, "OK",
                      CTRX_LAST);
```

Sync on Text

テキストの同期化ステップ **Sync On Text** は，指定された位置にテキスト文字列が現れるまで，仮想ユーザに続行を待機するよう指示します。**Sync On Text** の再生時，仮想ユーザは，ステップのプロパティに指定されている変更可能な座標の矩形領域の中でテキストを検索します。

エージェントがインストールされている場合は（第 31 章「LoadRunner Citrix エージェントの使用方法」を参照），マウスのクリックまたはダブルクリックそれぞれの前にテキストの同期化ステップを自動的に生成するように VuGen を設定できます。標準設定では，自動的なテキストの同期化は無効になっています。この記録オプションを有効にする方法の詳細については，476 ページ「コード生成記録オプション」を参照してください。

このオプションを無効にしてスクリプトを記録した場合でも，このオプションを有効にしてスクリプトを再生成すると，VuGen によってスクリプト全体にテキストの同期化呼び出しが挿入されます。詳細については，476 ページ「コード生成記録オプション」を参照してください。

記録中および記録後，個々のステップに手作業でテキストの同期化を追加できます。詳細については，490 ページ「手作業での同期化」を参照してください。

スクリプト・ビューでは，**Sync On Text** ステップに対応する関数呼び出しは **ctrx_sync_on_text_ex** です。

次のコードの抜粋は，Mercury Citrix エージェントがインストールされていて，テキストの同期化が有効になっている Citrix の記録中に記録された **ctrx_sync_on_text_ex** 関数を示しています。

```
ctrx_sync_on_window ("ICA Seamless Host Agent", ACTIVATE, 0,
0,391,224, "snapshot1", CTRX_LAST);

ctrx_sync_on_text_ex (196, 198, 44, 14, "OK", "ICA Seamless Host
Agent=snapshot2", CTRX_LAST);

ctrx_obj_mouse_click ("<class=Button text=OK>", 196, 198,
LEFT_BUTTON, 0, "ICA Seamless Host Agent=snapshot2",
CTRX_LAST);
```

この関数の詳細については，「**オンライン関数リファレンス**」を参照してください（[**ヘルプ**] > [**関数リファレンス**]）。

手作業での同期化

自動同期化に加えて，記録中および記録後，手作業で同期化を追加できます。通常，この機能は，実際のウィンドウは変更されていないのに，ウィンドウ内部のオブジェクトが変更された場合に使用します。ウィンドウは変更されていないので，VuGen は **Sync on Window** ステップを検出も記録もしていません。

たとえば，特定のグラフィック・イメージがブラウザ・ウィンドウに出現するまで再生を待機する場合は，手作業で同期化を挿入します。また，複数のタブを持つ大きなウィンドウを記録している場合，新しいタブの内容が開くのを待機する同期化ステップを挿入できます。

次の項では，ビットマップを対象とする同期化について説明します。手作業による **Sync on Text** の追加の詳細については，512 ページ「テキストの取得」を参照してください。

記録中の手作業による同期化の追加

記録中に同期化を追加するには，フローティング・ツールバーを使用します。**Sync On Bitmap** 関数を使用すると，再生を再開する前にフォーカスを得る必要のあるクライアント・ウィンドウの領域を指定できます。

Sync On Bitmap

同期化対象ビットマップ領域を指定するには，次の手順を実行します。

1 [**SyncOnBitmap を挿入**] ボタンをクリックします。

2 対象とするビットマップを囲むように矩形領域を指定します。ツリー・ビューの現在のステップの後に **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", CTRX_LAST);
```

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。

記録後の手作業による同期化の追加

記録セッションの後に同期化を追加することもできます。同期化ステップを追加するには，［スナップショット］ウィンドウで右クリックし，次の同期化オプションを選択します。

- **Sync on Bitmap**：ビットマップが表示されるまで待機します。
- **Sync on Obj Info**：オブジェクトの属性が，指定の値になるまで待機します（エージェントがインストールされている場合のみ）。
- **Sync on Text**：指定したテキストが表示されるまで待機します（エージェントがインストールされている場合のみ）。

記録中，**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，同期化が失敗した原因を調べることができます。VuGen は，両方のビットマップを［Failed Bitmap Synchronization］ダイアログ・ボックスに表示します。詳細については，494 ページ「Failed Bitmap Synchronization」を参照してください。

ビットマップ名は sync_bitmap_ ＜ hash_value ＞ .bmp の形式となります。このビットマップは，スクリプトの出力ディレクトリに格納されます。シナリオまたはプロファイルの場合には，出力ファイルの書き込み先に格納されます。

追加の同期化

前記に加えて，同期化に間接的に影響するその他のいくつかのステップを追加することもできます。

- 待ち時間の設定
- ウィンドウの表示 / 非表示の確認
- ビットマップ変更の待機

待ち時間の設定

Set Waiting Time ステップは，ほかの Citrix 同期化関数の待機時間を設定します。この設定は，同一スクリプト内でこの関数以降のすべての関数に適用されます。たとえば，**Sync on Window** ステップがタイムアウトする場合，標準設定のタイムアウトである 60 秒を 180 秒に延ばすことができます。

このステップを挿入するには，［**挿入**］＞［**新規ステップ**］＞［**Set Waiting Time**］を選択します。

ウィンドウの表示 / 非表示の確認

Win Exist ステップは，Citrix クライアントでウィンドウが表示されているかどうかを調べます。フロー制御ステートメントを追加することにより，この関数を使用して，常に開いているとはかぎらない警告ダイアログ・ボックスなどのウィンドウを調べることができます。次の例では，**ctrx_win_exist** によってブラウザが起動されたかどうかを調べます。2 番目の引数は，ブラウザ・ウィンドウが開くまでの待機時間を示します。指定した時間開かなかった場合は，ブラウザによって自身のアイコンがダブルクリックされます。

```
if (!ctrx_win_exist("Welcome",6, CTRX_LAST))
  ctrx_mouse_double_click(34, 325, LEFT_BUTTON, 0, CTRX_LAST)
```

このステップを挿入するには，［**挿入**］＞［**新規ステップ**］＞［**Win Exist**］を選択します。

このステップのもう一つの有用な用途は，ウィンドウが閉じているかどうかを確認することです。ウィンドウが閉じるのを待機する必要がある場合は，**UnSet Window** や **ctrx_unset_window** などの同期化ステップを使用するべきです。

関数の詳細については，「**オンライン関数リファレンス**」（［**ヘルプ**］＞［**関数リファレンス**］）を参照してください。

ビットマップ変更の待機

領域内に表示されるデータまたは画像がどのようなものかわからなくても，変更が行われることはわかっている場合があります。これをエミュレートするには，**Sync on Bitmap Change** ステップまたはその対応する関数 **ctrx_sync_on_bitmap_change** を使用します。スナップショットを右クリックし，右クリック・メニューから［**Insert Sync on Bitmap**］を選択します。ステップまたは関数がカーソルの位置に挿入されます。

この関数の構文は次のとおりです。

ctrx_sync_on_bitmap (x 座標 , y 座標 , 幅 , 高さ , ハッシュ値 , CTRX_LAST);
ctrx_sync_on_bitmap_change (x 座標 , y 座標 , 幅 , 高さ ,
[初期待機時間 ,] [タイムアウト ,]
[初期ビットマップ値 ,] CTRX_LAST);

ctrx_sync_on_bitmap_change では，次のオプションの引数が使用できます。

- 初期待機時間の値――変更の有無の確認を開始する時間。
- タイムアウト――失敗する前に変更が発生するまで待機する最大の秒数。
- 初期ビットマップ値――ビットマップの初期ハッシュ値。仮想ユーザは，ハッシュ値が指定した初期ビットマップ値と異なる値になるまで待機します。

次の例では，記録された関数に変更を加えて，300 秒の初期待機時間と 400 秒のタイムアウトを割り当てています。

```
/* 記録された関数 */
ctrx_sync_on_bitmap(93, 227, 78, 52,
                    "66de3122a58baade89e63698d1c0d5dfa",
CTRX_LAST);
```

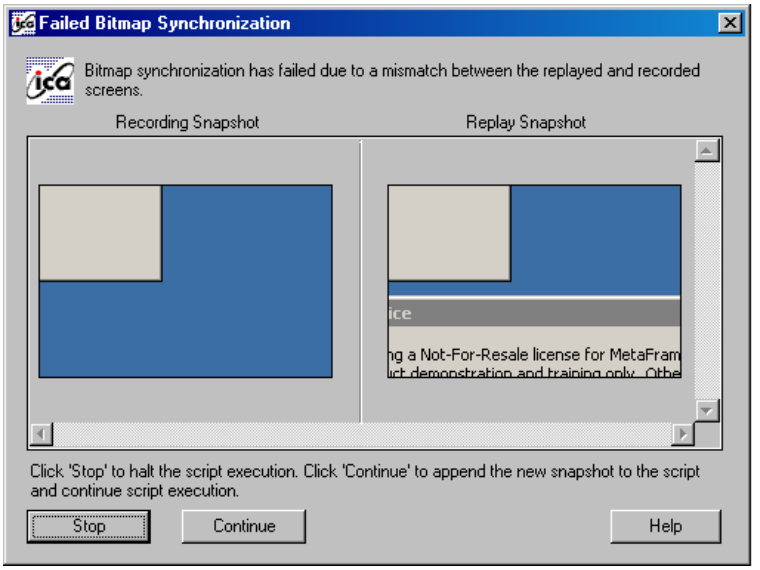
```
/* 関数に変更を加えて初期待機時間を 300，タイムアウトを 400 に設定
*/
ctrx_sync_on_bitmap_change(93, 227, 78, 52, 300, 400, CTRX_LAST);
```

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，474 ページ「設定記録オプション」を参照してください。

Failed Bitmap Synchronization

スクリプトの再生中に記録のスナップショットと再生のスナップショットの間に不一致があった場合，［Failed Bitmap Synchronization］ダイアログ・ボックスが開きます。

不一致をエラーとしてマークするか，それとも変更を採用するか，指定できます。



このダイアログ・ボックスが開いたら，次のどちらかのボタンをクリックして続行します。

- ［**Stop**］：スナップショット間の不一致をエラーとみなします。このエラーは，ほかのすべてのエラーと同じように処理され，実行を停止させます。特定の関数に［エラー時も処理を継続する］を指定できます。詳細については，504 ページ「エラー時も処理を継続する」を参照してください。
- ［**Continue**］：不一致を受け入れ，元のスナップショットと新しいスナップショットの両方を，将来の再生時に画面間を比較するための基準として使用します。再生時にどちらか一方のビットマップが返された場合，仮想ユーザは失敗しません。

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
```

詳細については，Citrix の Web サイト www.citrix.com を参照してください。

Citrix 関数の使用方法

Citrix 記録セッション中，VuGen によってクライアントとリモート・サーバ間のやり取りをエミュレートする関数が生成されます。生成された関数には，**ctrx** というプレフィックスが付きます。どの関数も記録セッションの後，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（使用廃止）	使用廃止。ただし，下位互換性確保のために利用可能。**ctrx_sync_on_text_ex** を使用してください。
ctrx_sync_on_text_ex（エージェント使用の場合のみ）	指定されたテキストが，指定された位置の付近に表示されるまで待機します。
ctrx_sync_on_window	ウィンドウが作成されるかアクティブになるのを待機します。
ctrx_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	エディット・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_execute_on_window	指定したウィンドウが表示されたら関数を実行します。
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	使用廃止。ただし，下位互換性確保のために利用可能。**ctrx_execute_on_window** を使用してください。

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，472 ページ「記録に関するヒント」を参照してください。

再生に関するヒント

ワイルドカード

ウィンドウ名の定義には，ワイルドカード（*）を使用できます。これは，接尾辞または接頭辞によって，再生中にウィンドウ名が変わる可能性がある場合に特に役立ちます。

次の例では，**Microsoft Internet Explorer** ウィンドウのタイトルをワイルドカードで置き換えています。

```
ctrx_mouse_click(573, 61, LEFT_BUTTON, 0, "Welcome to MSN.com - Microsoft Internet Explorer");

ctrx_mouse_click(573, 61, LEFT_BUTTON, 0, "* - Microsoft Internet Explorer");
```

詳細については，「**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。

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。

スクリプトの再生成

VuGen は，記録中，スクリプトとともにすべてのエージェント情報を保存します。標準設定では，VuGen はこの情報をスクリプトにも含めます（**Sync On Text** ステップを除く）。テキストの同期化の問題が発生した場合は，スクリプトを再生成して，テキストの同期化ステップを含めることができます。

また，記録オプションでエージェント情報の生成を無効にした場合でも，スクリプトを再生成して，この情報を含めることができます。

スクリプトの再生成は，手作業で変更を加えたスクリプトにも役立ちます。スクリプトを再生成すると，VuGen は手作業による変更をすべて廃棄し，最初に記録されたバージョンに戻します。

スクリプトを再生成するには，［**ツール**］＞［**スクリプトの再生成**］を選択し，必要なオプションを選択します。スクリプトの再生成の詳細については，80 ページ「仮想ユーザ・スクリプトの再生成」を参照してください。

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**』のターミナル・サービスに関する項を参照してください。

注：標準設定では，ターミナル・サーバでのセッションには，256 色のカラー・セットが使用されます。ターミナル・セッションを負荷テスト用に使用しようとしている場合は，必ず 256 色のカラー・セットを備えたマシンに記録してください。

デバッグに関するヒント

クライアントを 1 つだけインストールする

Citrix セッションに任意のアクションを記録するのに失敗した場合は，お使いのマシンに Citrix クライアントが 1 つしかインストールされていないことを確認してください。インストールされているクライアントが 1 つだけかどうかを調べるには，コントロールパネルから［プログラムの追加と削除］ダイアログ・ボックスを開き，Citrix ICA クライアントのエントリが 1 つだけあることを確認します。

ブレークポイントを追加する

問題が生じているコードの行を知るには，VuGen でスクリプトにブレークポイントを追加します。

スクリプトを同期化する

再生が失敗した場合，スクリプトに同期化関数を挿入して，対象ウィンドウがフォーカスを得るまで，待機時間を延ばす必要があるかもしれません。**lr_think_time** 関数を使って遅延時間を手作業で追加することもできますが，487 ページ「再生の同期化」で説明している同期化関数を使用することをお勧めします。

エラー時も処理を継続する

一致するウィンドウが見つからないなどのエラーが発生した後も実行を継続するように仮想ユーザを設定できます。［エラー時でも処理を継続する］は個々のステップに指定します。

これは，2 つのウィンドウのいずれかが開くことを知っているけれども，どちらが開くかわからない場合に特に役立ちます。つまり，どちらのウィンドウも正しいけれども，一方のみが開きます。

［エラー時も処理を継続する］を指定するには，次の手順を実行します。

ツリー・ビューの中でステップを右クリックし，［**プロパティ**］を選択します。［**ContinueOnError**］ボックスで［**CONTINUE_ON_ERROR**］を選択します。

スクリプト・ビューの中で関数を探し，**CTRX_LAST** の前に最後の引数として **CONTINUE_ON_ERROR** を追加します。

このオプションは次の関数には使用できません：**ctrx_key**，**ctrx_key_down**，**ctrx_key_up**，**ctrx_type**，**ctrx_set_waiting_time**，**ctrx_save_bitmap**，**ctrx_execute_on_window**，**ctrx_set_exception**。

拡張ログ

［拡張ログ］で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［再生ログ］タブ，またはスクリプト・ディレクトリの output.txt ファイルで参照できます。

スナップショット・ビットマップ

エラーの発生時，VuGen はスクリプトの**出力**ディレクトリに画面のスナップショットを保存します。このビットマップを見て，エラーが起きた原因を確認できます。

記録中，**ctrx_sync_on_bitmap** 関数に生成されたビットマップはスクリプトの **data** ディレクトリに保存されます。ビットマップ名は **＜ハッシュ値＞** .bmp の形式となります。再生中に同期化に失敗した場合には，生成されたビットマップはスクリプトの出力ディレクトリに書き込まれます。あるいは，シナリオ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

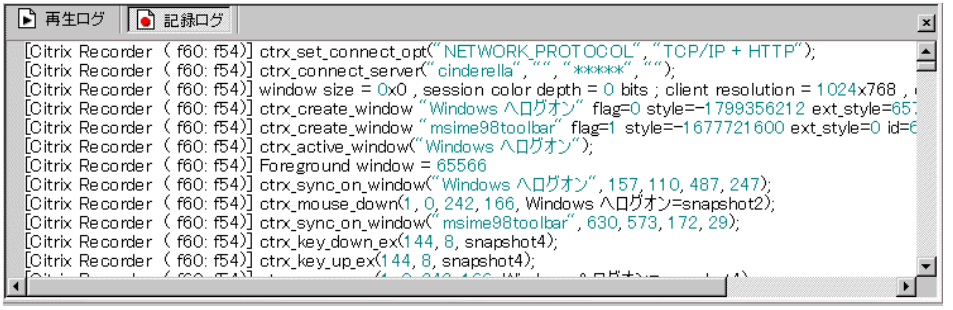
シナリオまたはチューニング・セッションの実行中に仮想ユーザを表示するには，仮想ユーザ・コマンド行ボックスに「-lr_citrix_vuser_view」と入力します。コントローラまたは Tuning Module Console で，［グループ情報］ダイアログ・ボックスを開き，［**詳細表示**］をクリックしてダイアログ・ボックスを拡張します。この操作は，テストのスケーラビリティに影響するので，問題のある仮想ユーザの振る舞いを調査する場合にだけ行うようにします。

スクリプトのスケーラビリティに対する影響を減らすには，コマンド・ラインの最後に仮想ユーザの ID を追加して，個々の仮想ユーザの詳細を表示します。
-lr_citrix_vuser_view **<VuserID>**

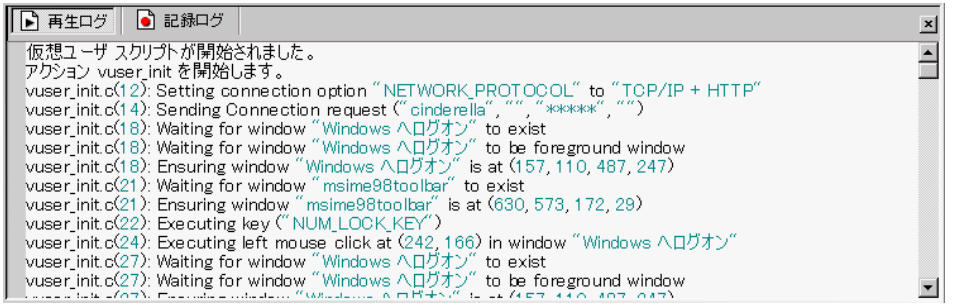
記録ログおよび再生ログの表示

記録に関する詳細情報を調べるには，［出力］ウィンドウに記録ログおよび再生ログを表示します。［出力］ウィンドウを開くには，［**表示**］＞［**出力ウィンドウ**］を選択します。

記録ログを表示するには，［**記録ログ**］タブを選択します。記録によって生成されたすべての関数，およびその時に発行された警告メッセージとエラーに関する詳細ログが表示されます。



再生ログを表示するには，［**再生ログ**］タブを選択します。VuGen によって実行されたすべてのアクション，および再生中に発行された警告とエラーに関する説明が表示されます。



ログ・メッセージに対応する，スクリプト内のステップに直接移動するには，再生ログの中で対象メッセージをダブルクリックします。

再生ログの情報の範囲は，ログの実行環境の設定によって変わります。詳細については，第 13 章「実行環境の設定」を参照してください。

第 31 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オプション・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オプション・ユーティリティです。このユーティリティは，製品 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。たとえば，VuGen は，エージェントなしで生成される［**マウス押下**］および［**マウス ダブルクリック**］の代わりに［**オブジェクト マウス クリック**］および［**オブジェクト マウス ダブルクリック**］ステップを生成します。

次の例は，エージェントをインストールした場合とインストールしない場合とで記録された同じマウスクリック操作を示しています。エージェントがインストールされている場合は，VuGen はクリック，ダブルクリック，解放などのすべてのマウス・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);
```

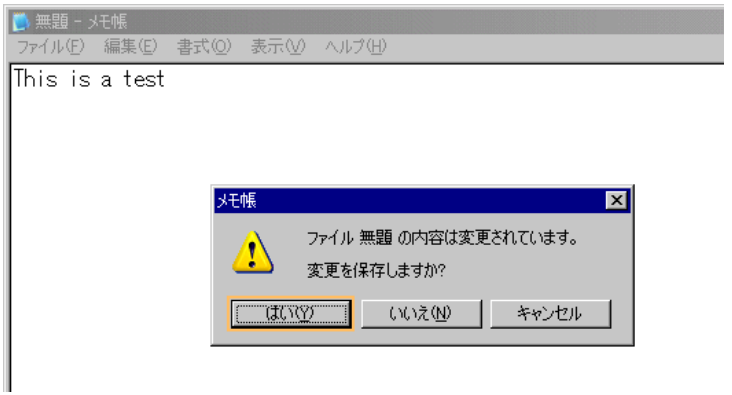
上の例で，**ctrx_obj_mouse_click** 関数の 1 番目の引数には，ウィンドウのタイトルとクラスのテキスト，つまりメモ帳が格納されています。エージェントは個々のオブジェクトに関する追加情報を提供しますが，仮想ユーザはウィンドウ名とオブジェクトの座標によってのみ，オブジェクトにアクセスします。

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，［**マウスのクリックを挿入**］，［**マウスのダブルクリックを挿入**］，［**ビットマップ同期を挿入**］，および［**ビットマップ値取得を挿入**］に限られます。256 色表示を使用している場合は，右クリック・メニューから［**ビットマップ取得時に同期化**］ステップおよび［**ビットマップ値の取得**］ステップを使用することはできません。

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右クリック・メニューから［**テキスト取得を挿入**］，［**テキスト同期を挿入**］，［**オブジェクト情報の取得を挿入**］，および［**オブジェクト情報同期を挿入**］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。

［**オブジェクト情報の取得**］ステップと［**オブジェクト情報取得時に同期化**］ステップは，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［**オブジェクト情報取得時に同期化**］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［**オブジェクト情報の取得**］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。［**テキストで同期化**］ステップと［**テキストの取得**］ステップについては，512 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・ボックスにフォーカスに入るまで待機することで同期化を行います。

```
ctrx_sync_on_obj_info("Font", 31, 59, FOCUSED, "TRUE", CTRX_LAST);
```

VuGen のオブジェクト検出機能を使用して，特定のオブジェクトに対してスナップショットの中から対話形式でアクションを実行できます。

エージェントの機能を使用して関数を対話形式で挿入するには，次の手順を実行します。

1 ツリー・ビューの中で，新しいステップを挿入する場所をクリックします。スナップショットが表示されていることを確認します。

2 スナップショットの中をクリックします。

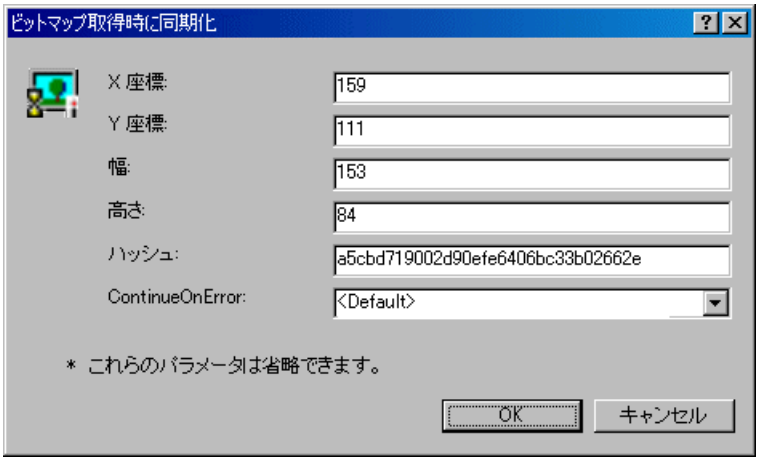
3 ビットマップをマークするには，ビットマップを右クリックして［**ビットマップ同期を挿入**］を選択します。

カーソルをドラッグして対象領域をマークする必要があることを示すメッセージが表示されます。［**OK**］をクリックし，選択するビットマップを対角方向にドラッグします。

マウスのボタンを放すと，スクリプトで現在選択されているステップの後に，ステップが挿入されます。

4 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。

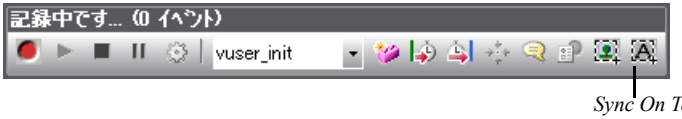
テキストの取得

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，［**テキストで同期化**］ステップを使用して保存します。

文字列を取得するには，次の手順を実行します。

1 記録中：ツールバーにある［**SyncOnText を挿入**］ボタンをクリックします。



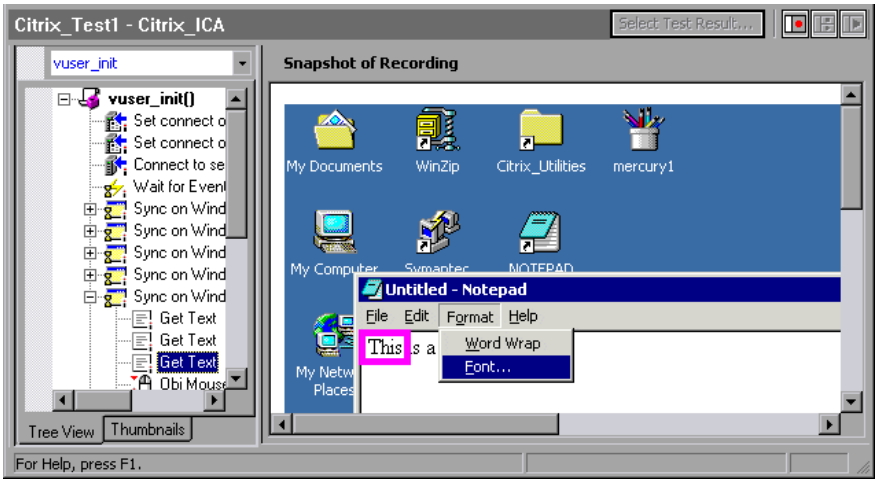
記録後：スナップショットの右クリック・メニューから［**テキスト同期を挿入**］を選択します。［ビットマップの選択］ダイアログが開き，同期化関数または情報提供関数を挿入するため領域をマークする必要があることを示します。

2 キャプチャするテキストの角をクリックし，マウスを対角方向にドラッグして保存するテキストをマークし，マウス・ボタンを離します。

3 記録中にステップを追加した場合は，現在の位置に［**テキストで同期化**］ステップが置かれ，テキストがバッファに保存されます。

記録後にステップを追加した場合は，［テキストの拡張同期］ダイアログ・ボックスが表示され，手作業でテキストを指定できます。

保存されたテキストはピンク色のボックスでマークされます。次のスナップショットでは，［**テキストで同期化**］ステップによってテキスト **This** を取得しています。



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**] > [**プログラム**] > [**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

次のスクリプトは，エージェントを含む実際の Citris ICA セッションを示します。

```
vuser_init()
{
   ctrx_set_connect_opt (NETWORK_PROTOCOL, "TCP/IP + HTTP");
   ctrx_connect_server ("Plato", "test", lr_decrypt("428c4445a14409b9"),
"QAlab");
   ctrx_wait_for_event ("LOGON");
   ctrx_sync_on_window ("ICA Seamless Host Agent", ACTIVATE, 0,
0,391,224, "snapshot1", CTRX_LAST);
   ctrx_sync_on_text (196, 198, "OK", TEXT, "ICA Seamless Host
Agent=snapshot2", CTRX_LAST);
   ctrx_obj_mouse_click ("<class=Button text=OK>", 196, 198,
LEFT_BUTTON, 0, "ICA Seamless Host Agent=snapshot2",
CTRX_LAST);
   lr_think_time(73);
return 0;
}
```

第 6 部

クライアント・サーバ・プロトコル

第 32 章

データベース仮想ユーザ・スクリプトの作成

VuGen を使用して，データベース・クライアント・アプリケーションとサーバの間の通信を記録することができます。その結果生成されるスクリプトをデータベース仮想ユーザ・スクリプトといいます。

本章では，次の項目について説明します。

- データベース仮想ユーザ・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
- グリッドを使った作業
- 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，次のデータベース・タイプがサポートされています：CtLib，DbLib，Informix，Oracle，ODBC，DB2-CLI。生成されたスクリプトには，データベース操作を表す LRD 関数が含まれています。LRD 関数の名前には **lrd** という接頭辞が付いており，1 つまたは複数のデータベース機能に対応します。たとえば，**lrd_fetch** 関数はフェッチ操作を表します。

記録されたセッションを実行すると，仮想ユーザ・スクリプトとデータベース・サーバとの間で直接通信が行われ，実際のユーザと同じ操作が実行されます。仮想ユーザの動作を設定して（実行環境の設定），操作の反復回数と反復間隔を指定できます。詳細については，第 13 章「実行環境の設定」を参照してください。

VuGen を使って，記録された定数をパラメータで置き換えることによって，スクリプトをパラメータ化できます。詳細については，第 9 章「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 12 章「ステートメントの相関」を参照してください。

トラブルシューティング情報とスクリプト作成のヒントについては，第 82 章「VuGen のデバッグのヒント」を参照してください。

データベース仮想ユーザの紹介

全国のカスタマー・サービス担当者がアクセスする顧客情報のデータベースがあるとします。この場合には，データベース仮想ユーザを使って，データベース・サーバが多数の情報の問い合わせに対応するという状況をエミュレートします。データベース仮想ユーザでは，次のことが可能です。

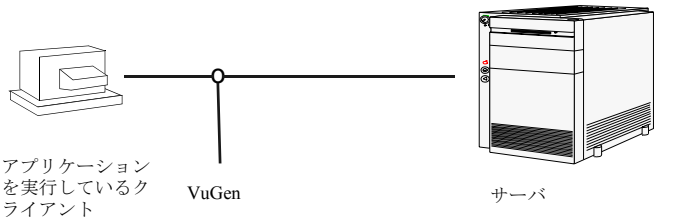
- サーバへの接続
- SQL クエリの発行
- 情報の検索と処理
- サーバからの切断

まず，利用可能なロード・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。

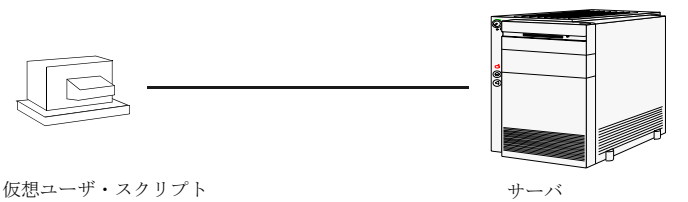
サーバ API への呼び出しが含まれるプログラムをデータベース（DB）仮想ユーザ・スクリプトといいます。データベース仮想ユーザ・スクリプトによって，クライアント・アプリケーションと，そのすべての操作がエミュレートされます。仮想ユーザによってスクリプトが実行されると，クライアント / サーバ・システムにユーザ負荷をかけた状態がエミュレートされます。仮想ユーザによって生成されるパフォーマンス・データは，レポートやグラフを使って分析できます。

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 4 章「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，付録 C「UNIX プラットフォームでのスクリプトのプログラミング」を参照してください。

データベース仮想ユーザ・スクリプトの概要

本項では，VuGen を使ったデータベース仮想ユーザ・スクリプトの作成プロセスの概要を説明します。

データベース仮想ユーザ・スクリプトを作成するには，次の手順を実行します。

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 4 章「VuGen を使った記録」を参照してください。

2 **スクリプトを拡張します。**

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 8 章「仮想ユーザ・スクリプトの拡張」を参照してください。

3 **パラメータを定義します（任意）。**

スクリプトに記録された固定値の代わりにパラメータを定義します。固定値をパラメータで置き換えることで，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 9 章「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 12 章「ステートメントの相関」を参照してください。

5 **実行環境を設定します。**

実行環境を設定することによって，スクリプト実行時の仮想ユーザ・スクリプトの振る舞いを制御します。設定には，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 13 章「実行環境の設定」を参照してください。

6 **VuGen からスクリプトを実行します。**

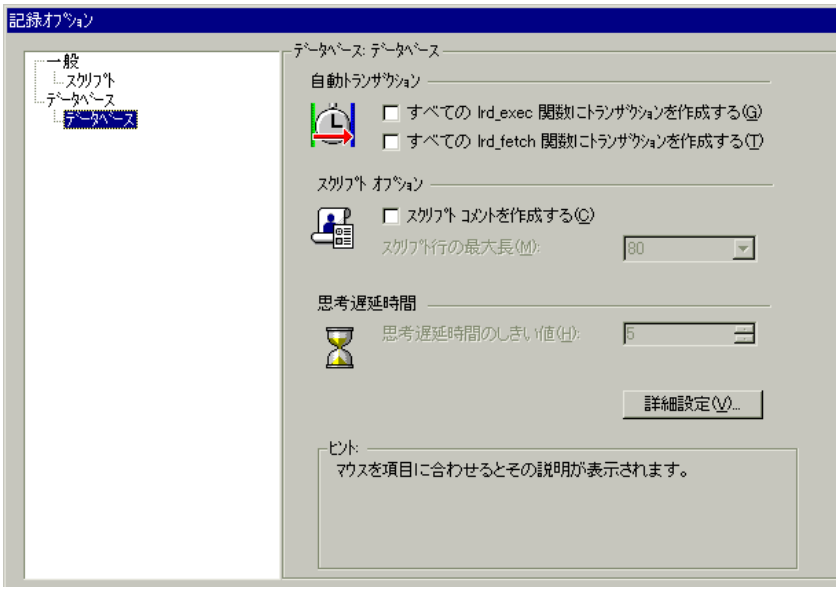
VuGen でスクリプトを保存して実行し，正しく動作することを確認します。

詳細については，第 15 章「スタンドアロン・モードでの仮想ユーザ・スクリプトの実行」を参照してください。

スクリプトを作成した後，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** のマニュアル，**Performance Center** のマニュアル，または **Business Availability Center** 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します。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します。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。

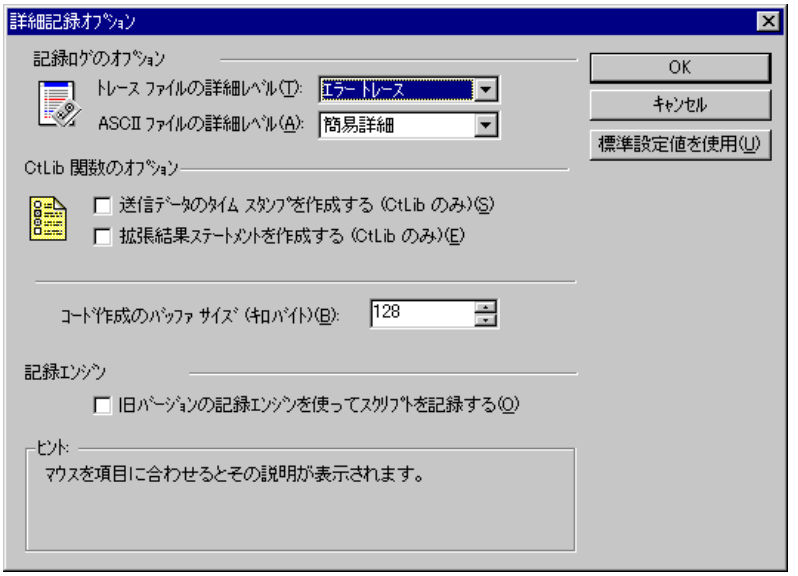
➤ ［**記録ログのオプション**］：トレース・ファイルと ASCII ログ・ファイルの詳細レベルを設定できます。トレース・ファイルに対して指定できるレベルとしては，［**オフ**］，［**エラー トレース**］，［**簡易トレース**］，および［**完全トレース**］があります。［エラー トレース］に設定すると，エラー・メッセージだけがログに書き込まれます。［簡易トレース］に設定すると，エラーのほかに，記録中に生成された関数のリストがログに書き込まれます。［完全トレース］に設定すると，メッセージ，告示，警告などがすべてログに書き込まれます。

記録セッションに関して ASCII タイプのログが生成されるようにも設定できます。指定できるレベルとしては，［**オフ**］，［**簡易詳細**］，［**完全詳細**］があります。［簡易詳細］設定では，すべての関数がログに書き込まれます。［完全詳細］設定では，生成されたすべての関数とメッセージが ASCII コードでログに書き込まれます。

- ［**CtLib 関数のオプション**］：送信データのタイム・スタンプと，拡張結果ステートメントが作成されるように設定できます。
  - ［**データ送信のタイム スタンプを作成する**］：標準設定では，VuGen は **mpszReqSpec** パラメータに **TotalLen** キーワードと **Log** キーワードを設定して **lrd_send_data** ステートメントを生成します。［詳細記録オプション］ダイアログ・ボックスでは，**TimeStamp** キーワードも生成するように指定することができます。既存のスクリプトでこの設定を変更した場合は，［**ツール**］＞［**スクリプトの再生成**］を選択して仮想ユーザ・スクリプトを生成します。標準設定で **Timestamp** キーワードを生成することは推奨されません。記録中に生成されたタイム・スタンプは再生中に生成されるものと異なるため，スクリプトの実行が失敗するからです。このオプションは，スクリプトの実行時に **lrd_send_data** に続く **lrd_result_set** が失敗した場合にだけ使用します。オプションを指定すれば，生成されたタイム・スタンプと，**lrd_send_data** を実行して失敗したときのタイム・スタンプを相関できるようになります。
  - ［**拡張結果ステートメントを作成する**］：標準設定では，結果セットを準備する段階で **lrd_result_set** 関数が生成されます。このオプションを選択すると，**lrd_result_set** 関数を拡張した関数である **lrd_result_set_ext** が生成されます。この関数は，結果セットを準備するほかに，**ct_results** からリターン・コードとタイプを発行します。
- ［**コード作成のバッファ サイズ**］：コード生成バッファの最大サイズをキロバイト単位で指定します。標準設定値は 128 キロバイトです。データベース・セッションが長い場合には，より大きなサイズを指定できます。
- ［**記録エンジン**］：VuGen 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	拡張フェッチを使用して結果セットの次の行を取得します（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	拡張フェッチを使用して結果セットから複数の行を取得します（ODBC のみ）。
lrd_reset_rows	取得された行の更新操作を準備します（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	返されたパラメータ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。たとえば，フェッチ操作に対しては **lrd_fetch** の代わりに **lrd_ora8_fetch** を記録します。

lrd_attr_set	LRDDBI ハンドルの属性を設定します。
lrd_attr_set_from_handle	LRDDBI ハンドル・ポインタを使用して属性を設定します。
lrd_attr_set_literal	リテラル文字列を使用して LRDDBI ハンドル属性を設定します。
lrd_env_init	LRDDBI ハンドルを割り当て，初期化します。

lrd_handle_alloc	LRDDBI ハンドルを明示的に割り当て，初期化します。
lrd_handle_free	LRDDBI ハンドルを明示的に解放します。
lrd_initialize_db	データベース処理環境を初期化します。
lrd_logoff	単純なデータベース・セッションを終了します。
lrd_logon	単純なデータベース・セッションを開始します。
lrd_logon_ext	(拡張された）単純なデータベース・セッションを開始します。
lrd_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_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	rowid の値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次の情報が表示されます。

- 記録された関数の並び
- データベース・クエリによって返されたデータを表示するグリッド
- クエリ中に取り出された行の数

関数の並び

VuGen ウィンドウに仮想ユーザ・スクリプトを表示すると，VuGen によって記録した操作のシーケンスを見ることができます。たとえば，標準的な Oracle データベース・セッションでは，次のような関数の並びが記録されます。

lrd_init	環境を初期化します。
lrd_open_connection	データベース・サーバに接続します。
lrd_open_cursor	データベース・カーソルを開きます。
lrd_stmt	SQL ステートメントとカーソルを関連付けます。
lrd_bind_col	ホスト変数をカラムにバインドします。
lrd_exec	SQL ステートメントを実行します。
lrd_fetch	結果セットから次の記録を取り出します。
lr_commit	データベース・トランザクションをコミットします。
lr_close_cursor	カーソルを閉じます。
lrd_close_connection	データベース・サーバから接続を解除します。
lrd_end	環境をクリーンアップします。

次に示すスクリプトは，Oracle サーバへの接続を開いて，ローカル設定を要求するクエリを実行したオペレータのアクションを VuGen で記録したものです。

```
lrd_init(&InitInfo, DBTypeVersion);
lrd_open_connection(&Con1, LRD_DBTYPE_ORACLE, "s1", "tiger",
"hp1", "", 0, 0, 0);
lrd_open_cursor(&Csr1, Con1, 0);
lrd_stmt(Csr1, "select parameter, value     from v$nls_parameters "
      "    where (upper(parameter) in ('NLS_SORT','NLS_CURRENCY',"
      "'NLS_ISO_CURRENCY', 'NLS_DATE_LANGUAGE',"
      "'NLS_TERRITORY'))", -1, 0 /*Non deferred*/, 1 /*Dflt Ora Ver*/, 0);
lrd_bind_col(Csr1, 1, &D1, 0, 0);
lrd_bind_col(Csr1, 2, &D2, 0, 0);
lrd_exec(Csr1, 0, 0, 0, 0, 0);
lrd_fetch(Csr1, 7, 7, 0, PrintRow2, 0);
…
lrd_close_cursor(&Csr1, 0);
lrd_commit(0, Con1, 0);
lrd_close_connection(&Con1, 0, 0);
lrd_end(0);
```

行情報

VuGen によって，SQL クエリごとに **lrd_fetch** 関数が生成されます。

```
lrd_fetch(Csr1, -4, 1, 0, PrintRow7, 0);
```

関数の 2 番目のパラメータは，取り出される行の数を示します。正数，負数のどちらも指定できます。

正数の値

正数の値は記録中に取り出された行の数を示し，すべての行が取り出されていないことを示します（オペレータがクエリ完了前にクエリを取り消した場合など）。

次の例では，データベース・クエリの実行時に 4 行を取り出していますが，すべてのデータは取り出していません。

```
lrd_fetch(Csr1, 4, 1, 0, PrintRow7, 0);
```

実行時には，正数によって示される数の行がスクリプトによって取り出されます（行が存在することが前提）。

負数の値

行の値が負数の場合，記録時にすべての行が取り出されたことを示します。負数の絶対値が，取り出された行の数です。

次の例では，結果セットの 4 行すべてを取り出しています。

```
lrd_fetch(Csr1, -4, 1, 0, PrintRow7, 0);
```

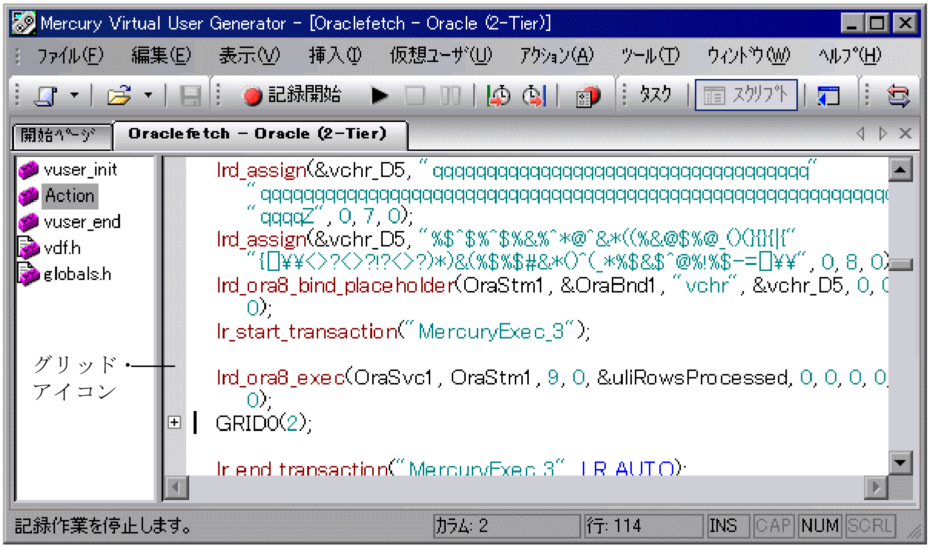
負数の値を含む **lrd_fetch** ステートメントを実行すると，実行時にテーブルから取り出せる行がすべて取り出されます。必ずしも記録時の行数と同じではありません。上の例では，テーブルの 4 行すべてが記録時に取り出されています。しかし，スクリプト実行時に，それ以上の行数がある場合は，それらがすべて取り出されます。

lrd_fetch の詳細については，「**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。

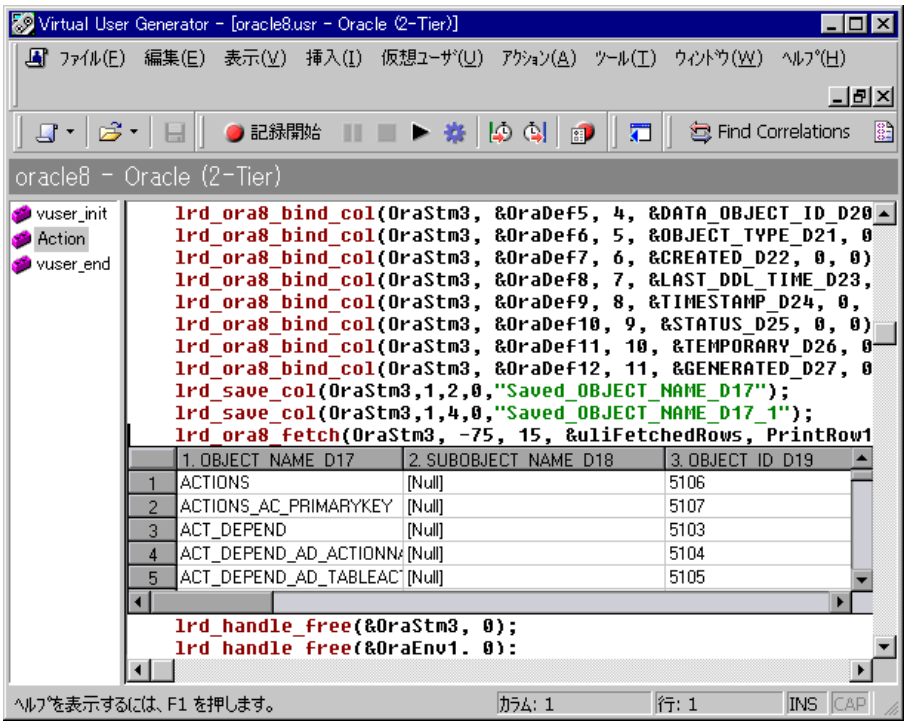
グリッドを使った作業

記録セッション中にクエリから返されたデータはグリッドに表示されます。グリッドを参照することで，アプリケーションによって SQL ステートメントがどのように生成されたかを確認したり，クライアント / サーバ・システムの効率を把握したりできます。

データ・グリッドは **GRID** ステートメントで表されます。データ・グリッドを開くには，GRID ステートメントの横の余白にあるアイコンをクリックします。



次の例では，ウィンドウにフライト予約データベースを対象に実行されたクエリが表示されています。クエリでは，フライト番号，空港名コード，出発地，曜日，およびその他のフライト関連情報が取得されています。



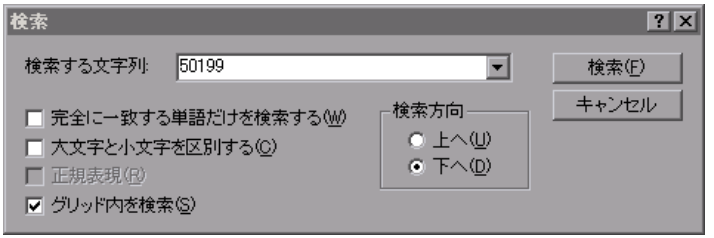
データ値が非常に長い場合は，その一部のみグリッドに表示されます。この切り捨ては，表示されているグリッドでのみ生じ，データには影響を与えません。

ウィンドウの表示枠は，幅を調整することが可能です。また，スクロール・バーを使って，最高 200 行までスクロールできます。この値を変更するには，マシンのオペレーティング・システム・フォルダ（C:\WINNT など）にある **vugen.ini** ファイルを開いて次のエントリを変更します。

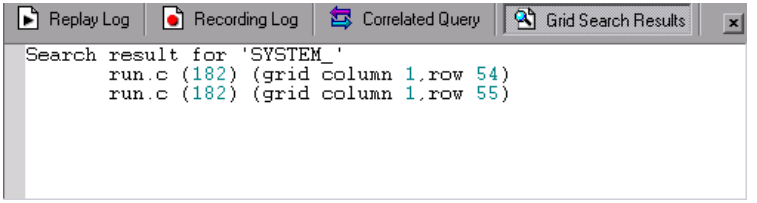
[general]max_line_at_grid=200

値を相関させる，あるいはデータをファイルに保存するには，セルをクリックし，右クリック・メニューの［**相関性を作成**］または［**ファイルへ保存**］を選択します。

グリッド全体でデータ（見えない部分も含む）を検索するには，［検索］ダイアログ・ボックスの［**グリッド内を検索**］を選択します。



出力ウィンドウの［**グリッド検索結果**］タブに結果が表示されます。



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。たとえば，リターン・コード「0」は，結果セットに利用可能な行が残っていることを示します。エラーが発生した場合，リターン・コードはエラーの種類を表します。たとえば，リターン・コード「2014」は，初期化中にエラーが発生したことを表します。

リターン・コードは 4 種類に分類され，それぞれ数値の範囲が決まっています。

リターン・コードの種類	範囲
情報	0 ～ 999
警告	1000 ～ 1999

リターン・コードの種類	範囲
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。次のスクリプトでは，**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。この振る舞いは次の方法で適用できます。

- **エラー処理のグローバル変更**：スクリプト全体またはスクリプトのセグメントにエラー処理を適用します。
- **エラー処理のローカル変更**：特定の関数にのみエラー処理を適用します。

エラー処理のグローバル変更

LRD_ON_ERROR_CONTINUE または LRD_ON_ERROR_EXIT ステートメントを発行することによって，仮想ユーザのエラー処理の方法を変更できます。標準設定では，データベース関連であれパラメータ関連であれ，エラーが生じると仮想ユーザによるスクリプトの実行が中止されます。この標準設定の振る舞いを変更するには，スクリプトに次の行を挿入します。

```
LRD_ON_ERROR_CONTINUE;
```

以降，仮想ユーザでエラーが発生してもスクリプトの実行が継続されます。

また，スクリプトの特定のセグメントだけでエラーが発生する場合に，仮想ユーザにスクリプトの実行を継続するように指定することもできます。たとえば，次のコードは，**lrd_stmt** や **lrd_exec** の関数内でエラーが発生した場合は，スクリプトの実行を継続するように仮想ユーザに指示します。

```
LRD_ON_ERROR_CONTINUE;
lrd_stmt(Csr1, "select …"…);
lrd_exec(…);
LRD_ON_ERROR_EXIT;
```

LRD_ON_ERROR_CONTINUE ステートメントを使うときは，重要かつ重大なエラーを見逃してしまう可能性があるので注意が必要です。

エラー処理のローカル変更

選択した関数の重要度を変更してエラー処理を設定できます。**lrd_stmt** や **lrd_exec** といったデータベース処理を実行する関数は，重要度を使用します。重要度は関数の最後のパラメータである **miDBErrorSeverity** で示します。このパラメータは，データベース・エラー（エラー・コード 2009）が発生した場合に，仮想ユーザにスクリプトの実行を継続させるかどうかを指示するものです。標準設定の「0」は，エラー発生時に仮想ユーザによるスクリプトの実行を中止することを示します。

たとえば，次のデータベース・ステートメントが失敗した場合は（たとえばテーブルが存在しない場合など），スクリプトの実行は中止されます。

```
lrd_stmt(Csr1, "insert into EMP values ('Smith',301)\n", -1, 1 /*Deferred*/,
        1 /*Dflt Ora Ver*/, 0);
```

データベース処理でエラーが発生した場合でも，仮想ユーザにスクリプトの実行を継続するように指示するには，ステートメントの重要度パラメータを「0」から「1」に変更します。

```
lrd_stmt(Csr1, "insert into EMP values ('Smith',301)\n", -1, 1 /*Deferred*/,
        1 /*Dflt Ora Ver*/, 1);
```

重要度を「1」に設定した場合，データベース・エラーが発生すると，警告が表示されます。特定の関数に重要度レベルを設定すると，その重要度レベルはその関数にだけ適用されることに注意してください。

CtLib 結果セット・エラー

CtLib の記録時には，アプリケーションによってステートメントが実行された後，利用可能な結果セットがすべて取り出されます。返された結果セットに取り出し可能なデータが含まれている場合は，アプリケーションでそのデータを対象にバインドおよび取り出しの操作が行われます。次に例を示します。

```
lrd_stmt(Csr15, "select * from all_types", -1, 148, -99999, 0);
lrd_exec(Csr15, 0, 0, 0, 0, 0);
lrd_result_set(Csr15, 1 /*Succeed*/, 4040 /*Row*/, 0);
lrd_bind_col(Csr15, 1, &tinyint_D41, 0, 0);
…
lrd_fetch(Csr15, -9, 0, 0, PrintRow3, 0);
```

結果セットに取り出し可能なデータが含まれていない場合，バインドと取り出しの操作は行えません。

スクリプトをパラメータ化すると，パラメータによっては結果データが取り出せなくなることがあります。新しいデータを取り出せない場合，特定のステートメントに対するバインドと取り出しの操作を記録した CtLib セッションは，実行できないことがあります。**lrd_bind_col** 関数または **lrd_fetch** 関数を実行しようとすると，エラーが発生し（LRDRET_E_NO_FETCHABLE_DATA：エラー・コード 2064），仮想ユーザによるスクリプトの実行が終了します。

このタイプのエラーが発生したときに，仮想ユーザにスクリプト実行の継続を指示することにより，実行を優先させることができます。次の行をスクリプトに挿入します。

```
LRD_ON_FETCHABLE_SET_ERR_CONT;
```

エラーの発生時にスクリプトの実行が終了する標準設定のモードに戻すには，次の行をスクリプトに入力します。

```
LRD_ON_FETCHABLE_SET_ERR_EXIT;
```

このステートメントを使うときは，重要かつ重大なエラーを見逃してしまう可能性があるので注意が必要です。

第 33 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
- データベース仮想ユーザ相関関数

以降の情報は，データベース（CtLib，DbLib，Informix，Oracle，ODBC，DB2-CLI）仮想ユーザ・スクリプト対象とします。

データベース仮想ユーザ・スクリプトの相関について

スクリプト実行時にエラーが発生した場合は，スクリプトの中でエラーが発生した場所を調べます。多くの場合，クエリを相関させることによって問題を解決できます。クエリの相関とは，実行時の値をパラメータに保存することです。保存した値は，同じスクリプト内の以降の場所で使用します。つまり，相関とは，あるステートメントの結果を別のステートメントへの入力として使用することです。

多くのクエリは，その入力が前のクエリの結果に依存します。この振る舞いをエミュレートするには，VuGen の相関機能を使用します。

スクリプトでの相関候補の検索

VuGen には，スクリプトを修正して確実に再生できるようにするための相関ユーティリティがあります。相関ユーティリティは，次の処理を行います。

- 相関候補を検索する。
- 適切な相関関数を挿入して，結果をパラメータに保存する。
- ステートメントの値をパラメータで置き換える。

自動相関は，スクリプト全体またはスクリプト内の特定の場所を対象に実行できます。

本項では，相関させる必要のあるステートメントを見つける方法について説明します。すでに相関させたい値がわかっている場合は，次の項に進んで，特定の値を相関させる方法を参照してください。

自動相関によるスクリプトの検索と相関は，次の手順で行います。

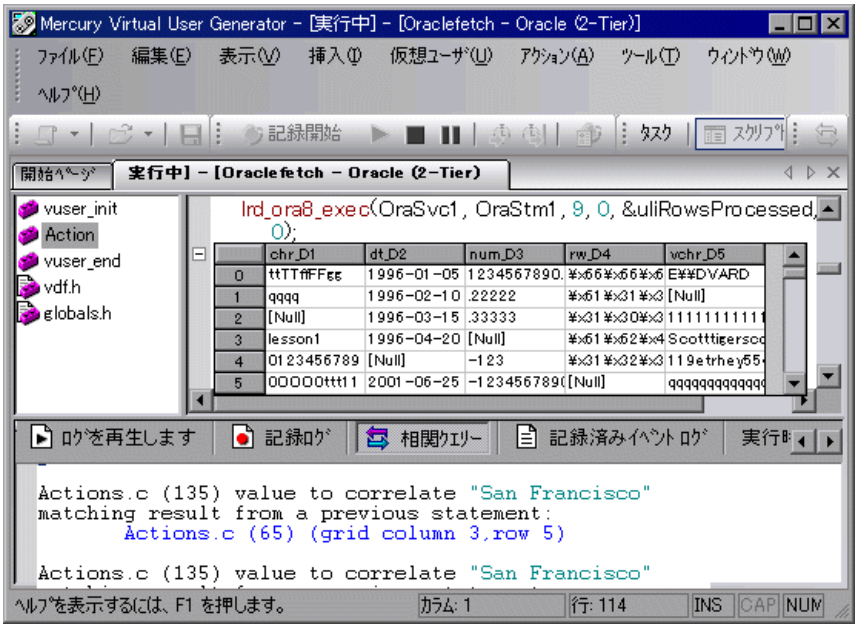
1 ［出力］ウィンドウを開きます。

［**表示**］＞［**出力ウィンドウ**］を選択して，ウィンドウの下部に出力タブを表示します。［再生ログ］タブでエラーがないか確認します。多くの場合，これらのエラーは相関によって修正できます。

2 ［**仮想ユーザ**］＞［**相関をスクリプトの中で検索**］を選択します。

VuGen によってスクリプト全体を対象とする検索が行われ，相関候補の値がすべて［**相関クエリー**］タブに表示されます。

次の例では，**lrd_ora8_stmt** 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。「**(grid column x, row y)**」という語をクリックします。グリッド内の値の位置にカーソルが移動します。

4 右クリック・メニューから［**相関性を生成**］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。

6 スクリプト内に現れる値をすべて検索するかどうかを尋ねるメッセージ・ボックスが開きます。選択したステートメント内の値だけを置き換える場合は，［**いいえ**］をクリックします。次の候補を検索して置き換える場合は［**はい**］をクリックします。

7 ［検索と置換］ダイアログ・ボックスが開きます。オリジナルのステートメントも含め，すべての置き換えを確認します。

8 ［検索と置換］ダイアログ・ボックスを閉じます。ステートメントの値がパラメータへの参照に置き換えられます。相関を取り消すと，直前のステップで作成されたステートメントは消去されます。

既知の値の相関

相関させる必要がある値がわかっている場合は，次の手続きを行います。

特定の値を相関させるには，次の手順を実行します。

1 相関させる値を含むクエリを使って，スクリプト内のステートメントを検索します。これは，通常 **lrd_assign**，**lrd_assign_bind**，**lrd_stmt** のいずれかの関数の引数です。引用符を含めずに値を選択します。

2 右クリック・メニューから［**相関を検索（カーソル位置）**］を選択します。選択した値を対象に相関が検索されます。

3 出力ウィンドウの［**相関クエリー**］タブで，相関させる結果をダブルクリックします。「**(grid column x, row y)**」という語をクリックします。グリッド内の値の位置にカーソルが移動します。

4 グリッド内で，相関させる値をクリックし，右クリック・メニューから［**相関性を生成**］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。

6 スクリプト内に現れる値をすべて検索するかどうかを尋ねるメッセージ・ボックスが開きます。選択したステートメント内の値だけを置き換える場合は，［**いいえ**］をクリックします。次の候補を検索して置き換える場合は［**はい**］をクリックします。

7 ［検索と置換］ダイアログ・ボックスが開きます。オリジナルのステートメントも含め，すべての置き換えを確認します。

8 ［検索と置換］ダイアログ・ボックスを閉じます。ステートメントの値がパラメータへの参照に置き換えられます。相関を取り消すと，直前のステップで作成されたステートメントは消去されます。

注：lrd_stmt 関数の値を相関させている場合は，データ型 date，time，binary（RAW，VARRAW）はサポートされません。

データベース仮想ユーザ相関関数

データベース仮想ユーザ・スクリプト（DbLib，CtLib，Oracle，Informix など）を使用するときに，VuGen の自動相関機能を使用して，スクリプトに適切な関数を挿入できます。相関関数は次のとおりです。

- **lrd_save_col** 関数では，表示枠内に表示されるクエリ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 34 章

DNS 仮想ユーザ・スクリプトの作成

VuGen では，DNS サーバに直接アクセスしてネットワークの動作状態をエミュレートできます。

本章では，次の項目について説明します。

- DNS 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・スクリプトをセッション・ステップまたはシナリオへ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**』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
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 35 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・スクリプトの開発の概要
- WinSock 記録オプションの設定
- LRS 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。たとえば，ネットワークの検査を行うために，Windows Sockets（WinSock）スクリプトを使って，バッファによって送受信される実際のデータを見ることができます。また，WinSock プロトコルは他の低レベルの通信セッションを記録するためにも使用できます。さらに，他のタイプの仮想ユーザでサポートされていないアプリケーションの記録と再生もできます。

Windows Sockets プロトコルを使用するアプリケーションを記録すると，記録されたアクションを表す関数が生成されます。各関数には，**lrs** という接頭辞が付きます。LRS 関数は，ソケット，データ・バッファ，および Windows Sockets 環境に対応します。VuGen を使って，アプリケーションの Winsock.dll または Wsock32.dll に対する API 呼び出しを記録します。

たとえば，**telnet** アプリケーションのアクションを記録して，スクリプトを作成できます。

次に示す例では，**lrs_send** を使って指定したソケットにデータを送信しています。

```
lrs_send("socket22", "buf44", LrsLastArg);
```

記録されたスクリプトは，VuGen のメイン・ウィンドウで表示および編集ができます。このウィンドウには，セッション中に記録された Windows Sockets API 呼び出しが表示されるので，記録時のネットワークの動作状況を追うことができます。

VuGen では，WinSock スクリプトを次の 2 つの方法で表示できます。

- スクリプトをアイコン形式で表示します。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，18 ページ「仮想ユーザ・スクリプトの表示と変更」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 36 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・スクリプトの開発の概要

本項では，VuGen を使って Windows Sockets 仮想ユーザ・スクリプトを開発する工程の概略を説明します。

Windows Sockets スクリプトを開発するには，次の手順を実行します。

1 **VuGen を使ってアクションを記録します。**

VuGen を起動し，Windows Sockets タイプを指定して，新しい仮想ユーザ・スクリプトを作成します。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 4 章「VuGen を使った記録」を参照してください。

2 **仮想ユーザ・スクリプトを拡張します。**

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 8 章「仮想ユーザ・スクリプトの拡張」を参照してください。

3 **パラメータを定義します（任意）。**

仮想ユーザ・スクリプトに記録された固定値に対してパラメータを定義します。固定値をパラメータ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 9 章「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 12 章「ステートメントの相関」を参照してください。

5 **実行環境を設定します。**

実行環境を設定することによって，スクリプト実行時の仮想ユーザの動作を制御します。設定には，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 13 章「実行環境の設定」および第 14 章「ネットワーク実行環境の設定」を参照してください。

6 **VuGen から仮想ユーザ・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 15 章「スタンドアロン・モードでの仮想ユーザ・スクリプトの実行」を参照してください。

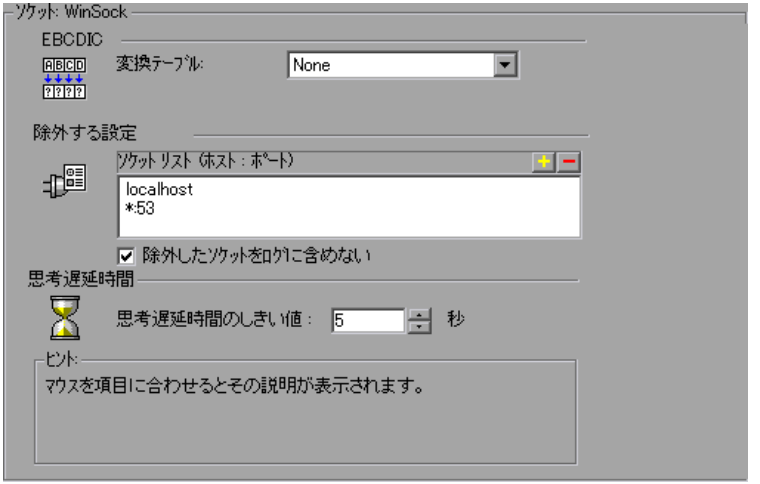
スクリプトを作成した後，スクリプトを自分の環境（LoadRunner シナリオ，Performance Center 負荷テスト，Tuning Module セッション，またはビジネス・プロセス・モニタ・プロファイル）に統合します。詳細については，『**Mercury LoadRunner コントローラ・ユーザーズ・ガイド**』，**Tuning Console** のマニュアル，**Performance Center** のマニュアル，または **Business Availability Center** のマニュアルを参照してください。

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

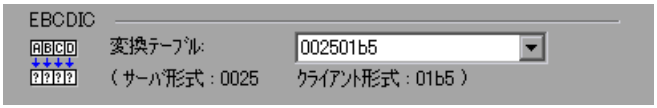
［記録オプション］ダイアログ・ボックスを開くには，［**ツール**］＞［**記録オプション**］を選択するか，［記録開始］ダイアログ・ボックスで［**オプション**］ボタンをクリックします。WinSock 用のオプションが表示されます。



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。［**なし**］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。

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ない場合，**lr_think_time** ステートメントは生成されません。

思考遅延時間のしきい値を設定するには，［**思考遅延時間のしきい値**］ボックスに必要な値（秒単位）を入力します。標準設定の値は 5 秒です。

LRS 関数の使用

Windows Sockets プロトコルを使ったクライアントとサーバの間の通信をエミュレートするために開発された関数を，LRS 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	ソケットを作成します。
lrs_disable_socket	ソケットに対する操作を無効にします。
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・ソケットへ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	バッファ用に割り当てられてい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，見つかった文字列からの相対位置に基づいて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・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・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	WinSock DLL を初期化します。
lrs_create_socket	ソケットを初期化します。
lrs_send	データグラムで，またはストリーム・ソケットへデータを送信します。
lrs_receive	データグラムまたはストリーム・ソケットからデータを受信します。
lrs_disable_socket	ソケットの処理を無効にします。
lrs_close_socket	開いているソケットを閉じます。
lrs_cleanup	WinSock DLL の使用を終了します。

VuGen では，Windows 上で Windows Sockets プロトコルを使用するアプリケーションの記録と再生がサポートされます。UNIX プラットフォームでは再生だけがサポートされます。

第 36 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，578 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。

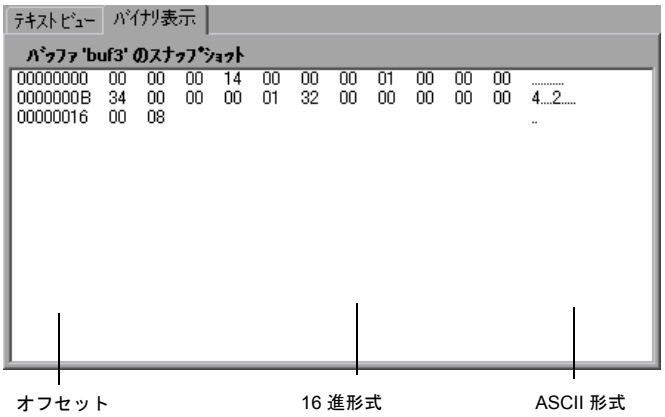
スナップショット・ウィンドウでのデータの表示

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。

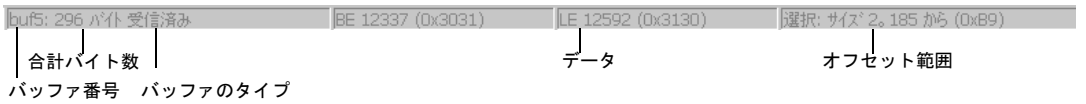
標準設定では，バッファ・データは読み取り専用として保存されます。バッファの内容を変更する場合は，バッファのテキスト・ビューで［**読み取り専用**］ボックスをクリアします。VuGen から，ブックマークとパラメータに影響がある可能性があるという警告が発行されます。

［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表示でも使用できます。テキスト・ビューで特定のデータの位置を見つけ，その位置をブックマークとして保存し，バイナリ表示の中でそのブックマークに直接ジャンプできます。

ブックマークは 1 バイトにも複数バイトにも付けられます。リスト中のブックマークをクリックすると，対応する場所が選択された状態で［バッファのスナップショット］ウィンドウに表示されます。初期設定では，そのデータがテキスト・ビューでは青で強調表示され，バイナリ表示ではブックマーク・ブロックが赤で表示されます。また，バイナリ表示でブックマークにカーソルを置くと，ポップアップ・テキスト・ボックスが開いてブックマークの名前が表示されます。

永久ブックマークと標準ブックマークを作成できます。永久ブックマークは，バッファの「バイナリ表示」内で常に印が付けられています（青いボックスで囲まれています)。バッファ内の異なる位置を指している場合でも，このブックマークは選択された状態で青いボックスの中に表示されています。カーソルの位置は赤でマークされます。一方，標準ブックマークには常に印が付けられているわけではありません。標準ブックマークは，そこにジャンプしたときには赤く印が付きますが，バッファ内でカーソルを移動すると選択されていない状態になります。標準設定のブックマークは永久ブックマークです。

ブックマークを処理するには，次の手順を実行します。

1 ブックマークを作成するには［バッファのスナップショット］（テキスト・ビューまたはバイナリ表示）で 1 つ以上のバイトを選択し，右クリックで表示されるメニューから［**新規ブックマーク**］を選択します。

2 ブックマーク・リストを表示するには，［**表示**］＞［**出力ウィンドウ**］を選択し，［**ブックマーク**］タブを選択します。

3 ブックマークに名前を割り当てるには，ブックマーク・リストのブックマークをクリックして，タイトルを編集します。

4 ブックマークの場所を変更するには，［**ブックマーク**］タブでブックマークを選択してから［バッファのスナップショット］で新しいデータを選択します。［**ブックマーク**］タブの［**変更**］をクリックします。

5 永久ブックマークを標準ブックマークに変更する場合（永久ブックマークは，カーソルを新しい場所に移動しても常にマークされた状態を維持します）は，ブックマークを選択してマウスを右クリックし，［**永久ブックマーク**］の横にあるチェックをクリアします。

6 リストに永久ブックマークだけを表示するには，［**ブックマーク**］タブで［**永久ブックマークのみ表示する**］チェック・ボックスを選択します。

7 特定のバッファのブックマークを表示するには，そのバッファからブックマークを選択し，［**フィルタ**］ボックスの［**選択バッファのみ**］を選択します。

8 ブックマークを削除するには，［**ブックマーク**］タブでブックマークを選択して［**削除**］をクリックします。

バッファ・データの修正

ツリー・ビューには，既存のデータを削除，変更，追加することによってデータを修正するためのいくつかのツールがあります。

- データの挿入
- データの編集
- データのパラメータ化

データの挿入

データ・バッファに数値を挿入できます。数値はシングルバイト，ダブルバイト，または 4 バイト値として挿入できます。

データ・バッファに数値を挿入するには，次の手順を実行します。

1 バッファ内をクリックします。

2 右クリックして表示されるメニューから［**詳細設定**］＞［**数字の挿入**］＞［**指定 ...**］を選択します。

3 挿入する ASCII 値を［**値**］ボックスに入力します。

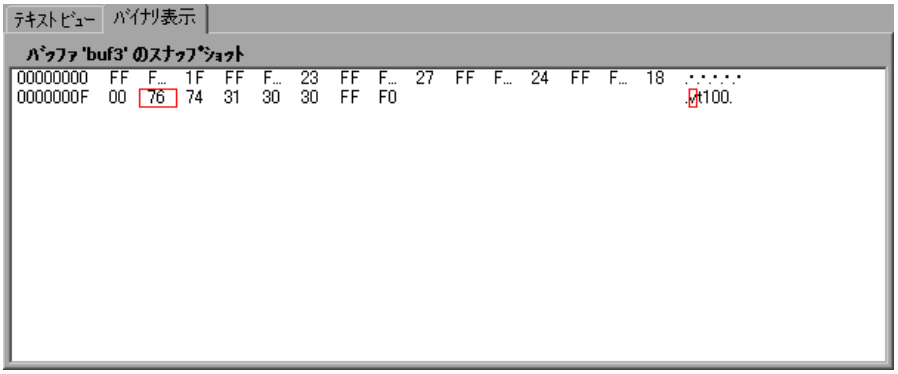
4 挿入するデータのサイズとして，1 バイト，2 バイト，または 4 バイトを［**サイズ**］ボックスから選択します。

5 ［**OK**］をクリックして完了します。VuGen によってデータの 16 進表現がバッファに挿入されます。

データの編集

バッファ・データに対して，標準的な編集操作（コピー，貼り付け，切り取り，削除，元に戻す）が行えます。［バイナリ表示］では，挿入する実際のデータを指定できます。VuGen では，データの形式（1，2，または 4 バイト，および 16 または 10 進値など）を指定できます。バイナリ・データをコピーし，数値としてバッファに挿入できます。［バイナリ表示］の右カラムには，10 進数または 16 進数を表示できます。

次の例では「**OK**」という語が選択されています。



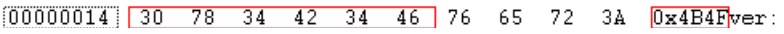
データの次行の先頭で単純なコピー（CTRL+C）と貼り付け（CTRL+V）操作を実行すると，実際のテキストが挿入されます。

```
00000014 4F 4B 76 65 72 3A 20 4D 69 63 OKver: Mic
```

［**詳細設定**］＞［**数値としてコピー**］＞［**10 進法**］を選択してからデータを貼り付けると，選択された文字の ASCII コードの 10 進値が挿入されます。

```
00000014 31 39 32 37 39 76 65 72 3A 20 19279ver:
```

［**詳細設定**］＞［**数値としてコピー**］＞［**16 進法**］を選択してからデータを貼り付けると，選択された文字の ASCII コードの 16 進値が挿入されます。



元戻しバッファに，選択したバッファに対して行われたすべての変更が保持されます。この情報はファイルに保存されるので，ファイルを閉じても利用できます。他の人に変更を取り消されないようにしたい場合は，元戻しバッファを空にします。元戻しバッファを空にするには，右クリックして表示されるメニューで［**詳細設定**］＞［**元戻しバッファを空にする**］を選択します。

バイナリ表示でバッファ・データを編集するには，次の操作を行います。

1 バッファ・データをコピーするには，次の手順を実行します。

- 文字としてコピーする場合は，1 つ以上のバイトを選択し，CTRL+C キーを押します。
- 10 進数としてコピーする場合は，右クリックして表示されるメニューから［**詳細設定**］＞［**数値としてコピー**］＞［**10 進法**］を選択します。
- 16 進数としてコピーする場合は，右クリックして表示されるメニューから［**詳細設定**］＞［**数値としてコピー**］＞［**16 進法**］を選択します。

2 データを貼り付けるには，次の手順を実行します。

- 単一バイト（クリップボードのデータのサイズを単一バイトと仮定した場合）として貼り付ける場合は，バッファ内の望みの場所をクリックして CTRL+V キーを押します。
- 短い形式（2 バイト）として貼り付ける場合は，右クリックして表示されるメニューから［**詳細設定**］＞［**数字の挿入**］＞［**短形式で貼り付け（2 バイト）**］を選択します。
- 長い形式（4 バイト）として貼り付ける場合は，右クリックして表示されるメニューから［**詳細設定**］＞［**数字の挿入**］＞［**長形式で貼り付け（4 バイト）**］を選択します。

3 データを削除するには，テキスト・ビューまたはバイナリ表示で削除するデータを選び，右クリックして表示されるメニューから［**削除**］を選択します。

データのパラメータ化

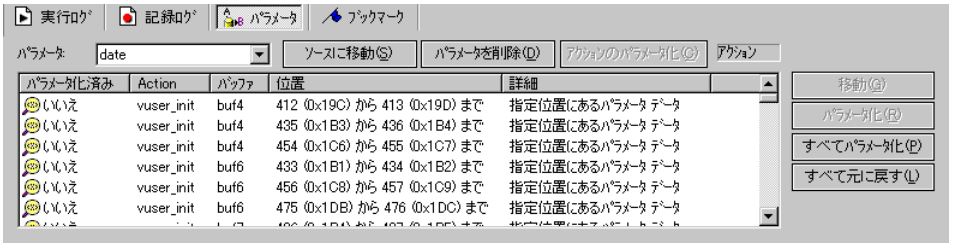
ツリー・ビューの［バッファのスナップショット］ビューでデータを直接パラメータ化できます。パラメータ化する範囲を指定して境界を指定できます。パラメータ化する文字列の境界を指定しないと，VuGen によってスクリプトに **lrs_save_param** 関数が挿入されます。境界を指定すると，境界引数を指定できる **lrs_save_searched_string** 関数がスクリプトに挿入されます。

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 9 章「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。

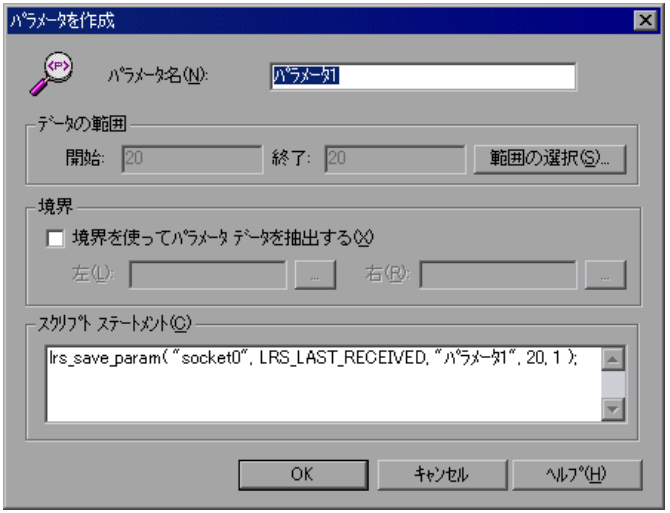


VuGen でパラメータを操作できます。

- **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。
- ［**境界**］セクションの［**左**］ボックスの隣にある参照ボタンをクリックします。バッファ内の選択を示す小さいダイアログ・ボックスが表示されます。バッファ内の境界を選択して［**完了**］をクリックします。右の境界についてもこの手順を繰り返します。

5 ［**スクリプト ステートメント**］セクションの引数に必要な修正を加えます。たとえば，**lrs_save_param** 関数に **_ex** を追加してエンコード・タイプを指定できます。これらの関数の詳細については，「**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，［**パラメータ**］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 特定のパラメータ置換を元に戻すと、［**パラメータ化済み**］カラムは［**いいえ**］に変更されます。取り消したパラメータ置換に対して再度パラメータ化ルールを適用するには、右クリック・メニューから［**パラメータで置換**］を選択します。

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。

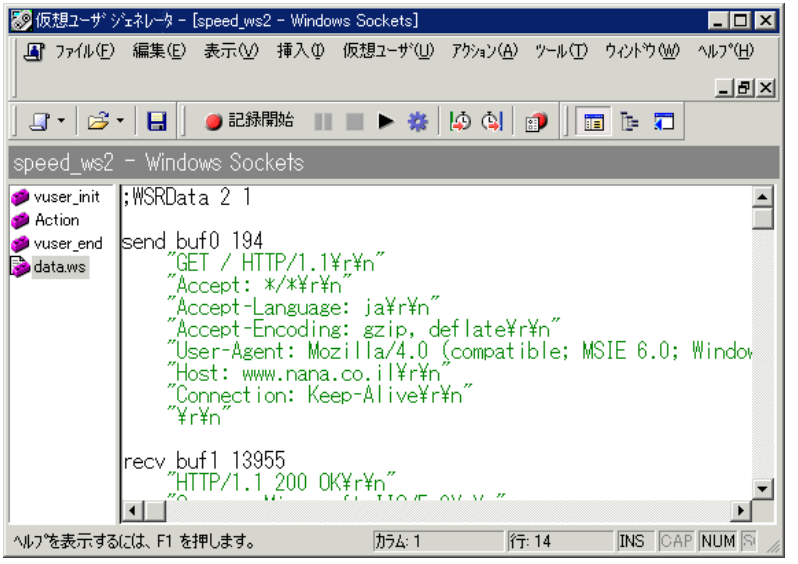
バッファで作成したパラメータが使えなくなっている場合，VuGen によって，パラメータが定義されたバッファ内のパラメータを削除するよう求められます。パラメータを削除すると，他のバッファも含め，すべての置換が元に戻ります。

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，558 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"
```

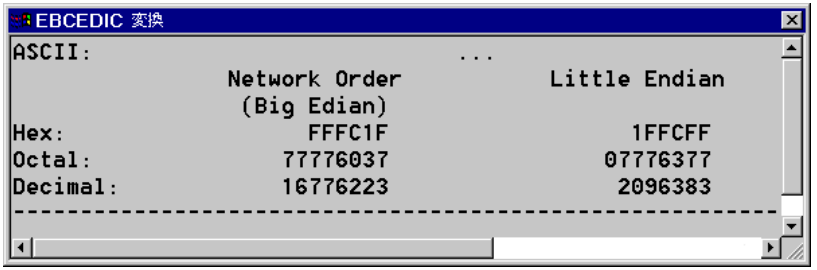
次の例は通常のデータ・ファイルのヘッダー，記述子およびデータを示します。

```
;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。

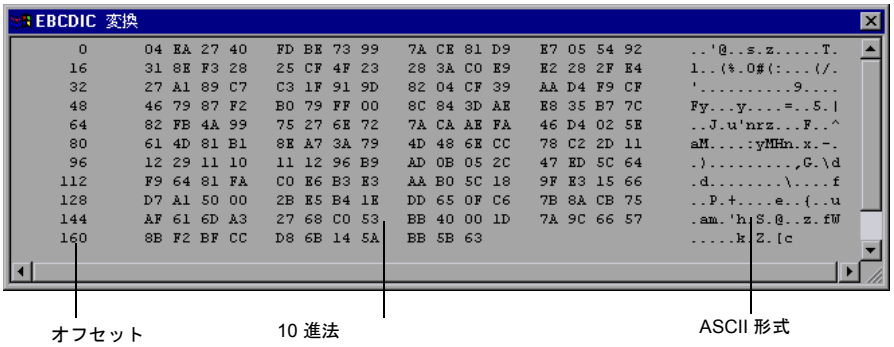
ビューア・ウィンドウでデータを表示するには，データを選択して F7 キーを押します。選択されたテキストが 4 文字以下の場合，VuGen によってそのデータが「**短い形式**」の 16 進，10 進，および 8 進で表示されます。



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，584 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，584 ページ「表示形式の設定」を参照してください。

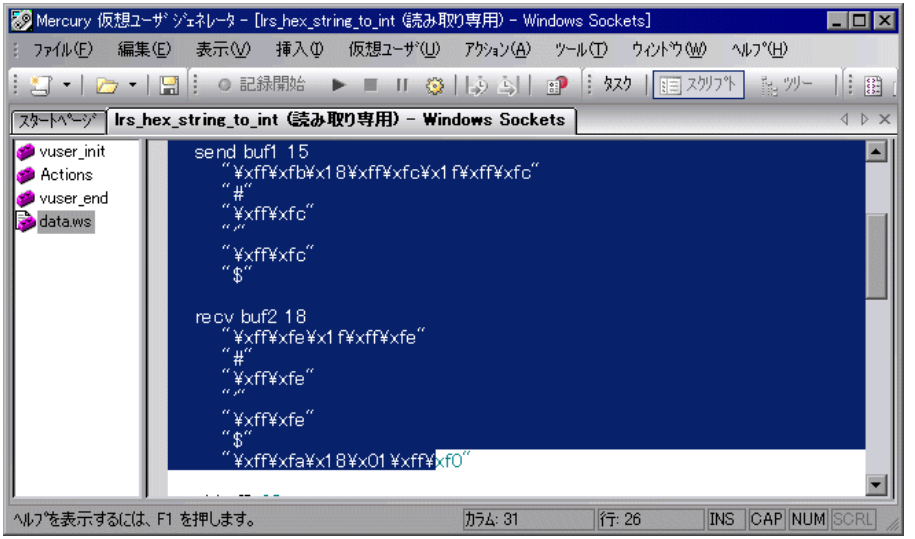
標準設定では，最初のカラムに，マークされた領域の先頭からの文字オフセットが表示されます。2 番目のカラムには，データが 16 進形式で表示されます。3 番目のカラムには，ASCII 形式でデータが表示されます。EBCDIC データを表示するときは，ASCII 形式の非印字文字（「/n」など）はすべてドットで表されます。



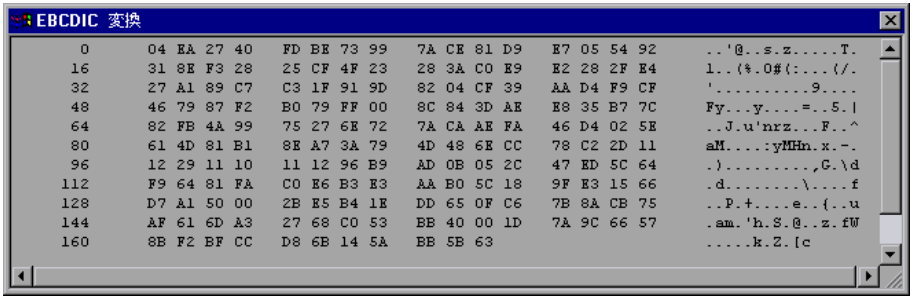
F7 キーを押すと表示されるビューア・ユーティリティは，特にパラメータ化を行う際に役立ちます。このユーティリティを使うことで，パラメータに保存するデータのオフセットを決定できます。

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 12 章「ステートメントの相関」を参照してください。

表示形式の設定

VuGen のビューア・ウィンドウ（F7 キー）でのバッファ・データの表示方法を指定できます。**＜LoarRunner ディレクトリ＞/dat** ディレクトリの **conv_frm.dat** ファイルには，次の表示パラメータが含まれています。

- **LongBufferFormat**：5 文字以上を表示するときに使われる形式です。オフセットには nn，16 進データには XX，ASCII データには aa を使います。
- **LongBufferHeader**：長いバッファ形式で表示する場合に，各バッファの先頭に表示するヘッダーです。
- **LongBufferFooter**：長いバッファ形式で表示する場合に，各バッファの末尾に表示するフッターです。
- **ShortBufferFormat**：4 文字以下を表示するときに使われる形式です。標準的なエスケープ・シーケンスと変換文字を使用できます。

  サポートされているエスケープ・シーケンス文字は次のとおりです。

¥a	ベル（アラート）
¥b	バックスペース
¥f	改ページ
¥n	改行
¥r	復帰
¥t	水平タブ
¥v	垂直タブ
¥'	引用符
¥"	二重引用符
¥¥	円記号
¥?	疑問符
¥ooo	ASCII 文字（8 進数）

サポートされている変換文字は次のとおりです。

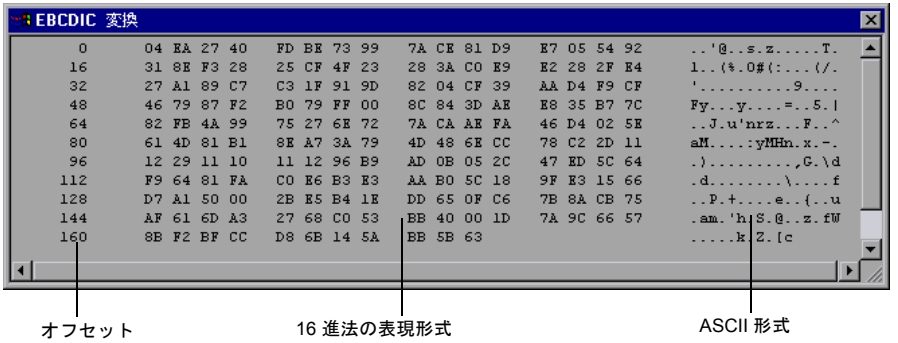
%a　　ASCII 表記
%BX　ビッグ・エンディアン（ネットワーク順序）16 進数
%BO　ビッグ・エンディアン（ネットワーク順序）8 進数
%BD　ビッグ・エンディアン（ネットワーク順序）10 進数
%LX　リトル・エンディアン 16 進数
%LO　リトル・エンディアン 8 進数
%LD　リトル・エンディアン 10 進数

- **AnyBufferHeader**：各バッファの先頭に表示するヘッダー。
- **AnyBufferFooter**：各バッファの末尾に表示するフッター。
- **NonPrintableChar**：非印字 ASCII 文字を表す文字。
- **PrintAllAscii**：非印字 ASCII 文字を強制的に出力するには，1 に設定します。

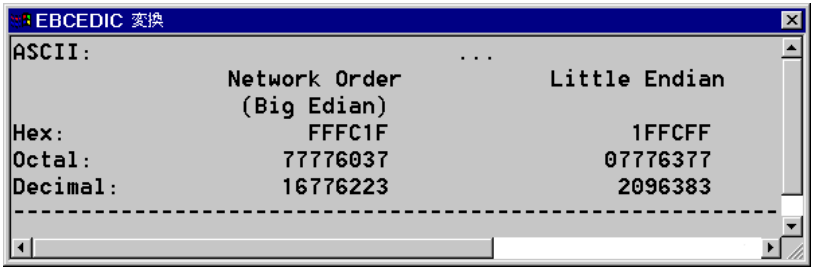
標準設定では，長い形式と短い形式が設定され，非印字文字はドットとして出力するように指定されています。

```
[BufferFormats]
LongBufferFormat=nnnnnnnn    XX XX XX XX  XX XX XX XX  XX XX XX
XX  XX XX XX XX    aaaaaaaaaaaaaaaa¥r¥n
LongBufferHeader=
LongBufferFooter=
ShortBufferFormat=ASCII:¥t¥t¥t%a¥r¥n¥t¥tNetwork Order¥t¥tLittle
Endian¥r¥n¥t¥t (Big
Edian)¥r¥nHex:¥t¥t%BX¥t¥t%LX¥r¥nOctal:¥t¥t%BO¥t¥t%LO¥r¥nDecimal:¥
t%BD¥t¥t%LD¥r¥n
AnyBufferHeader=
AnyBufferFooter=----------------------------------------------------------------
¥r¥n
NonPrintableChar=.
PrintAllAscii=0
```

標準設定の LongBufferFormat は次のように表示されます。



標準の ShortBufferFormat は次のように表示されます。



デバッグに関するヒント

VuGen では，いくつかの方法でスクリプトのデバッグを行えます。スクリプト実行に関する詳細メッセージを示すさまざまな出力ログとウィンドウを表示できます。

特に Windows Sockets 仮想ユーザ・スクリプトについては，「バッファの不一致」に関する追加情報が提供されます。バッファの不一致とは，受け取ったバッファ（再生中に生成されたもの）のサイズと期待バッファ（記録中に生成されたもの）の差を示します。ただし，受け取ったバッファと期待バッファが同じサイズの場合は，内容が異なっていても不一致のメッセージは発行されません。この情報は，システムや仮想ユーザ・スクリプトでの問題を見つけるのに役立ちます。

バッファの不一致情報は実行ログに記録できます。実行ログが表示されていないときは，［**表示**］＞［**出力ウィンドウ**］を選択して，実行ログを表示します。

バッファの不一致は必ずしも問題を示しているわけではありません。たとえば，バッファに以前のログイン時刻などの重要でないデータが含まれている場合，このような不一致は無視できます。

```
Mismatch (expected 54 bytes, 58 bytes actually received)
The expected buffer is:
=================
¥r¥n Last login: Wed Sep 2 10:30:18 from acme.mercury.c¥r¥n
=================
The received buffer is:
=================
¥r¥n Last login: Thu Sep 10 11:19:50 from dolphin.mercury.c¥r¥n
```

ただし，期待バッファと受信バッファの間でサイズが著しく異なる場合は，システムに問題があることを示します。対応するバッファでデータの不一致を確認してください。

重要な不一致かどうかを判断できるように，アプリケーションを完全に理解しておく必要があります。

WinSock スクリプトの手作業による相関

VuGen には，仮想ユーザ・スクリプトを相関させるためのユーザ・インタフェースがあります。相関は，動的なデータを利用する場合に必要です。WinSock 仮想ユーザ・スクリプトでよくある問題の 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。

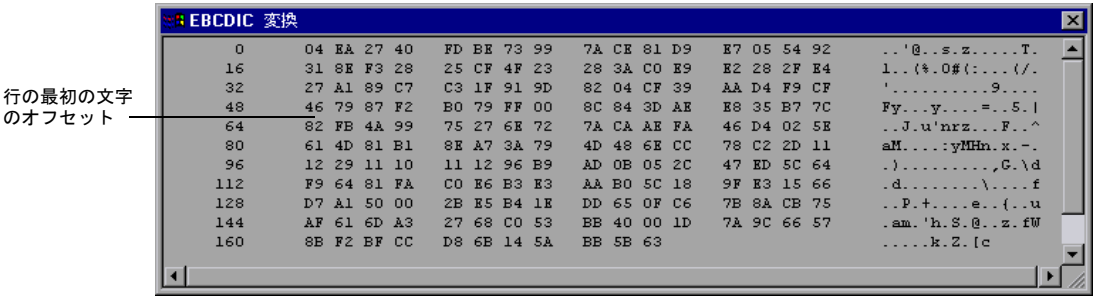
3 **data.ws** ファイル内で，データを受け取ったバッファを調べます（この例では buf47）。

```
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，580 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 7 部

ユーザ定義の仮想ユーザ・スクリプト

第 37 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。たとえば，仮想ユーザ関数を使って，サーバ・パフォーマンスの測定，サーバ負荷の制御，デバッグ・コードの追加，テストに含まれる仮想ユーザまたは監視対象の仮想ユーザについてのランタイム情報の取得などができます。

本章では，対象アプリケーションのライブラリやクラスと連携して動作する仮想ユーザ・スクリプトを VuGen エディタでプログラミングする方法について説明します。

また，Visual C および Visual Basic 環境からプログラミングを行って仮想ユーザ・スクリプトを開発することもできます。これらの環境では，開発アプリケーションに 仮想ユーザ API 関数のライブラリをインポートして仮想ユーザ・スクリプトを作成します。詳細については，第 80 章「Visual Studio による仮想ユーザ・スクリプトの作成」を参照してください。

ユーザ定義のスクリプトを作成するには，まず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したりできます。

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。

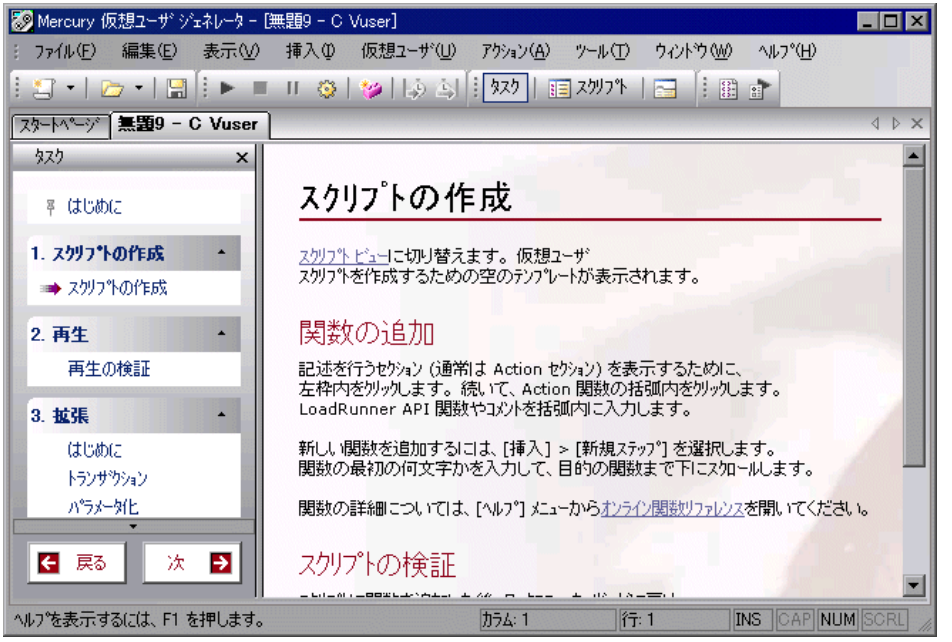
```
Action1()
{

    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「VuGen ワークフローの表示」を参照してください。

スクリプトの作成

［スクリプトの作成］ウィンドウには，Web Services スクリプトの作成のいくつかのガイドラインが含まれます。

- ［**関数の追加**］：関数の入力方法と入力場所を示します。
- ［**スクリプトの検証**］：関数の追加後のスクリプトの検証方法を示します。

C 言語の関数の使用についてのガイドライン

制御フローや構文など，標準 ANSI-C の規約はすべて，C 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。たとえば，次のコードは標準のオブジェクト **lr** を作成します。

```
Set lr = CreateObject("LoadRunner.LrApi")
```

JavaScript 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。たとえば，次のコードは標準のオブジェクト **lr** を作成します。

```
var lr = new ActiveXObject("LoadRunner.LrApi")
```

第 38 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。たとえば，テキストの前に 2 つのスラッシュ「//」を付けることによって，コメントを挿入できます。

CORBA，RMI，EJB，および Jacada 仮想ユーザに関する章では，記録によってスクリプトを作成する方法を説明しています。プログラミングによって Java コードのスクリプトを作成するには，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 **Business Availability Center** 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 13 章「実行環境の設定」を参照してください。次のサンプル **action** メソッドでは，仮想ユーザ ID を取り出して出力しています。

```
public int action() {
    lr.message("vuser: " + lr.get_vuser_id() + " xxx");
        return 0;
 }
```

end メソッド

end メソッドには，サーバからのログオフや環境の後始末など，スクリプトの終了時に仮想ユーザに実行させるコードを配置します。

end メソッドは，仮想ユーザがスクリプトの実行を終了するときに 1 回だけ実行されます。次の例に示す **end** メソッドでは「**End**」というメッセージを再生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値を記録します。
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ビジネス・プロセス・モニタ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。たとえば，コードの中で，リターン・コードが特定の値になっているかどうかを調べることができます。リターン・コードが正しい値であれば，lr.PASS ステータスを発行します。リターン・コードの値が正しくなければ，lr.FAIL ステータスを発行します。

トランザクションの開始と終了を示すには，次の手順を実行します。

1 **lr.start_transaction** 関数は，スクリプト内で処理の実行時間の計測を開始する場所に挿入します。

2 **lr.end_transaction** 関数は，スクリプト内で処理の実行時間の計測を終了する場所に挿入します。**lr.start_transaction** 関数で指定したトランザクション名を指定します。

3 トランザクションのステータス（lr.PASS または lr.FAIL）を指定します。

```
public int action() {

for(int i=0;i<10;i++)
  {
     lr.message("action()"+i);
     lr.start_transaction("trans1");
     lr.think_time(2);
     lr.end_transaction("trans1",lr.PASS);
  }
     return 0;
}
```

ランデブー・ポイントの挿入

次の項は，Business Availability Center には適用されません。

クライアント / サーバ・システムを対象に多数のユーザによる負荷をエミュレートするには，「**ランデブー・ポイント**」を作成して，多数の仮想ユーザが同時にタスクを実行するように同期させなければなりません。ある仮想ユーザがランデブー・ポイントに到達すると，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，ビジネス・プロセス・モニタ，またはチューニング・モジュール・コンソール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。たとえば，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，609 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 の再生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。再生ログでは，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 13 章「実行環境の設定」を参照してください。

lr.think_time 関数の詳細については，「**オンライン関数リファレンス**」（［**ヘルプ**］＞［**関数リファレンス**］）を参照してください。

コマンド・ライン引数の処理

スクリプトを実行するときにコマンド・ライン引数を指定することで，実行時に仮想ユーザ・スクリプトに値を渡すことができます。コントローラ，チューニング・モジュール，またはビジネス・プロセス・モニタで，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，**チューニング・モジュール・コンソール**のマニュアル，**Performance Center** のマニュアル，または **Business Availability Center** 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，再生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わかっているコードをスクリプトにインポートしたい場合は，それらの仮想ユーザをプロセスとして実行できます。仮想ユーザをスレッドまたはプロセスとして実行する方法の詳細については，第 13 章「実行環境の設定」を参照してください。

基本的な Java 仮想ユーザ・スクリプトを実行する場合，スクリプトは通常 1 個のスレッド（メイン・スレッド）で構成されています。Java 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。再生ログの最初のログ・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 8 部

分散コンポーネント・プロトコル

第 39 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 80 章「Visual Studio による仮想ユーザ・スクリプトの作成」を参照してください。

第 40 章「COM 仮想ユーザ・スクリプトの詳細」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 5 章「スクリプト生成オプションの設定」を参照してください。

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。たとえば，**Clock** オブジェクトには，**Clock**，**Alarm**，**Timer** のインタフェースがあるかもしれません。各インタフェースには，1 つまたは複数のメソッドがあります。たとえば，**Alarm** インタフェースには，**AlarmOn** メソッドおよび **AlarmOff** メソッドがあるかもしれません。

また，インタフェースは，1 つまたは複数のプロパティを持つことができます。メソッド呼び出しによる方法と，プロパティの値の設定または取得による方法の両方で同じ機能を実行できることがあります。たとえば，**Alarm Status** プロパティを **On** に設定した場合の結果は，**AlarmOn** メソッドを呼び出した場合の結果と同じです。

COM オブジェクトは，多数のインタフェースをサポートすることができます。コンポーネントには必ず **IUnknown** インタフェースが実装されており，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 5 章「スクリプト生成オプションの設定」を参照してください。COM 固有のオプションとフィルタの設定方法については，632 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 4 章「VuGen を使った記録」を参照してください。

2 **オブジェクト・フィルタを適切に設定し直します。**

生成されたログ・ファイルを使って，フィルタで記録対象のオブジェクトを選択し直します。詳細については，「記録対象の COM オブジェクトの選択」の項を参照してください。

3 **スクリプトを拡張します。**

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 8 章「仮想ユーザ・スクリプトの拡張」を参照してください。

4 **パラメータを定義します（任意）。**

スクリプトに記録された固定値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 9 章「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，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 13 章「実行環境の設定」を参照してください。

6 **VuGen からスクリプトを実行します。**

VuGen でスクリプトを保存して実行し，正しく動作することを確認します。

詳細については，第 15 章「スタンドアロン・モードでの仮想ユーザ・スクリプトの実行」を参照してください。

スクリプトを作成した後，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** のマニュアル，**Performance Center** のマニュアル，または **Business Availability Center** 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。

たとえば，以下は Visual Basic ライブラリのローカルの COM の例です。

```
Class JetES {039EA4C0-E696-11D0-878A-00A0C91EC756}
was loaded from type library "JET Expression Service Type Library"
({2358C810-62BA-11D1-B3DB-00600832C573} ver 4.0)
```

これは，サーバ上で負荷を生成しないため，追加してはなりません。

同様に，次に示す OLE DB および Microsoft Windows Common Controls はローカル・オブジェクトなので，そのクラスとライブラリは，サーバへの負荷を生成しません。したがって，これらも記録する必要はありません。

```
Class DataLinks {2206CDB2-19C1-11D1-89E0-00C04FD7A829} was
loaded from type library "Microsoft OLE DB Service Component 1.0 Type
Library"({2206CEB0-19C1-11D1-89E0-00C04FD7A829} ver 1.0)

Class DataObject {2334D2B2-713E-11CF-8AE5-00AA00C00905}
was loaded from type library "Microsoft Windows Common Controls 6.0
(SP3)"
({831FDD16-0C5C-11D2-A9FC-0000F8754DA1} ver 2.0)
```

ただし，たとえば，次の一覧は，サーバで負荷を生成するため記録する必要があるクラスを示します。

```
Class Order {B4CC7A90-1067-11D4-9939-00105ACECF9A}
was loaded from type library "FRS"
({B4CC7A8C-1067-11D4-9939-00105ACECF9A} ver 1.0)
```

たとえば VuGen とともにインストールされる flight_sample で使用される **FRS** ライブラリのクラスは，サーバの処理能力を必要するので記録する必要があります。

COM オブジェクトが別の COM オブジェクトを呼び出す場合，すべての呼び出しがタイプ情報ログ・ファイルにリストアップされます。たとえば，アプリケーションが **FRS** クラス関数を呼び出すたびに，**FRS** ライブラリは **ADO**（**ActiveX Data Object**）ライブラリを呼び出します。このような呼び出しの連鎖に含まれるいくつかの関数がフィルタに表示されている場合，VuGen によって連鎖を開始する最初の呼び出しだけが記録されます。**FRS** および **ADO** 呼び出しの両方を選択した場合は，**FRS** 呼び出しだけが記録されます。

また，フィルタで **ADO** ライブラリだけを選択した場合は，**ADO** ライブラリへの呼び出しが記録されます。多くの場合，連鎖における最初のリモート・オブジェクトへの呼び出しを記録するのが簡単です。ただし，場合によっては，アプリケーションが複数の異なる COM オブジェクトのメソッドを使用します。それらのメソッドがいずれもサーバに負荷をかける単独のオブジェクトを使用する場合は，最終的な共通オブジェクトだけを記録することもできます。

選択可能なオブジェクト

VuGen では，オブジェクトのタイプ・ライブラリを読み込むことができれば，そのオブジェクトを記録できます。タイプ・ライブラリがシステムにインストールされていない場合や VuGen でタイプ・ライブラリが見つからない場合には，対応する COM オブジェクトは［記録オプション］ダイアログ・ボックスに表示されません。これらの COM オブジェクトがアプリケーションによって使用されている場合，VuGen ではこれらのオブジェクトを識別できず，ファイル内に **INoTypeInfo** として示されます。

除外できるインタフェース

［記録オプション］ダイアログ・ボックスでは，タイプ・ライブラリのリストに含まれているすべてのインタフェースが，オブジェクトごとに表示されます。このダイアログ・ボックスで，各インタフェースを記録に含めるか除外するかを指定できます。ただし，**ADO**，**RDS**，**Remote** のオブジェクトは，フィルタに 1 つのグループとして含めることができます。その場合，フィルタには，これらの環境またはインタフェースの個々のオブジェクトまたはインタフェースは表示されません。また，タイプ・ライブラリから記録対象にインクルードしたオブジェクトのインタフェースが，タイプ・ライブラリにないために［記録オプション］ダイアログ・ボックスに表示されない場合があります。その場合は，VuGen のスクリプトを生成した後で，スクリプトに含まれるこれらのインタフェースを特定して，VuGen によって生成された interfaces.h ファイルでそれらのインタフェースの GUID 番号を確認します。この情報を使って，以下に説明する方法で，インタフェースを除外できます。

COM 記録オプションの設定

COM の［記録オプション］ダイアログ・ボックスを使って，フィルタ・オプションと COM のスクリプト編集オプションを設定します。レジストリ，ファイル・システム，Microsoft トランザクション・サーバ（MTS）のタイプ・ライブラリを探すには，オンライン・ブラウザを使用します。

詳細については，次を参照してください。

- オブジェクトのフィルタリング
- フィルタの設定
- COM スクリプト編集オプションの設定

オブジェクトのフィルタリング

フィルタ・オプションを使って，VuGen で記録する COM オブジェクトを指定できます。オブジェクトは環境およびライブラリの中から選択できます。

フィルタ・オプションで，標準フィルタの設定または代替フィルタの作成を行います。環境およびタイプ・ライブラリを指定することで，記録セッションを絞り込めます。

DCOM プロファイル

- ［**Default filter**］: COM 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尋ねるダイアログ・ボックスが表示されます。

インタフェースの横のチェック・ボックスをクリアすると，［除外インターフェース］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**］：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。

- ［**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 COM オブジェクトを選択するには，次の手順を実行します。

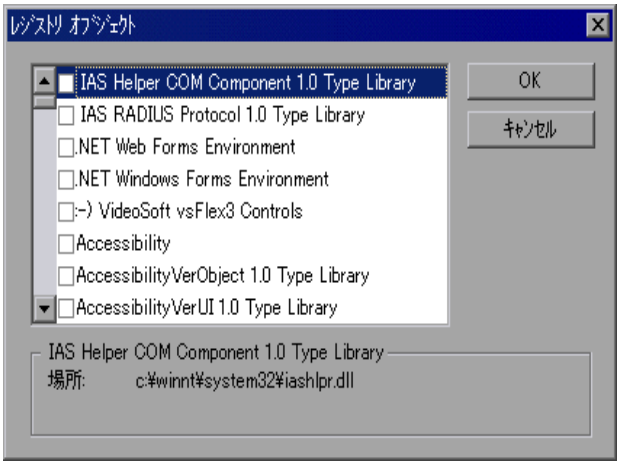
1 メイン・メニューの［**ツール**］＞［**記録オプション**］を選択するか，［記録開始］ダイアログ・ボックスの［**オプション**］をクリックします。記録オプションのツリーが表示されたダイアログ・ボックスが開きます。［COM/DCOM］の［**フィルタ**］ノードを選択します。

［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。

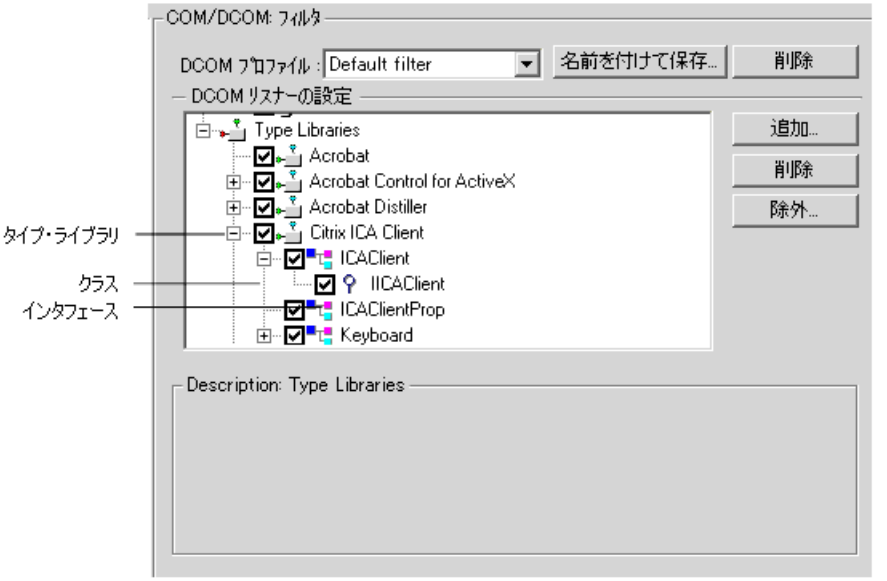
4 ファイル・システムからタイプ・ライブラリを追加するには，［**追加**］をクリックして［**ファイル システムの参照**］を選択します。

使用するファイルを選択して，［**OK**］をクリックします。

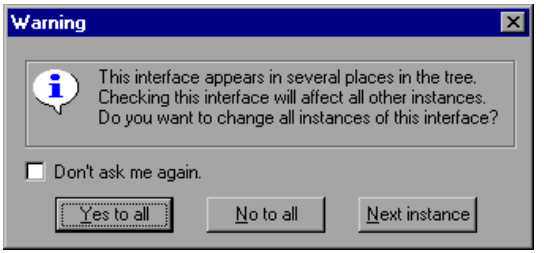
5 タイプ・ライブラリが［Type Libraries］リストに表示されたら，そのツリーの分岐を開いて，タイプ・ライブラリ内の使用可能なクラスをすべて表示できます。クラス・ツリーの分岐を開いて，そのクラスによってサポートされているインタフェースをすべて表示できます。

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。

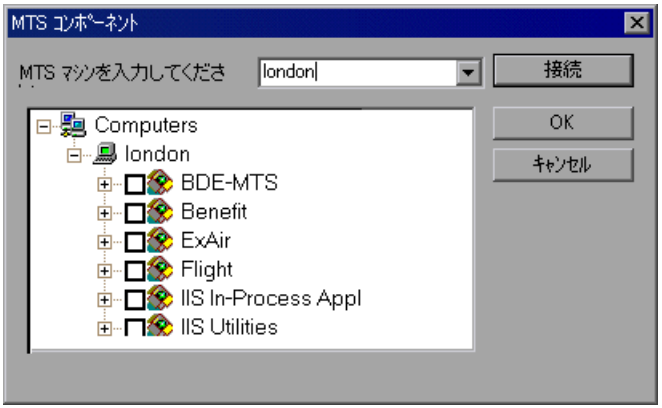


6 クラスの種類ごとに，異なるインタフェースを実装できます。除外されていない他のクラスにも実装されているインタフェースを除外しようとすると，VuGen によって次の警告が表示されます。



［**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。

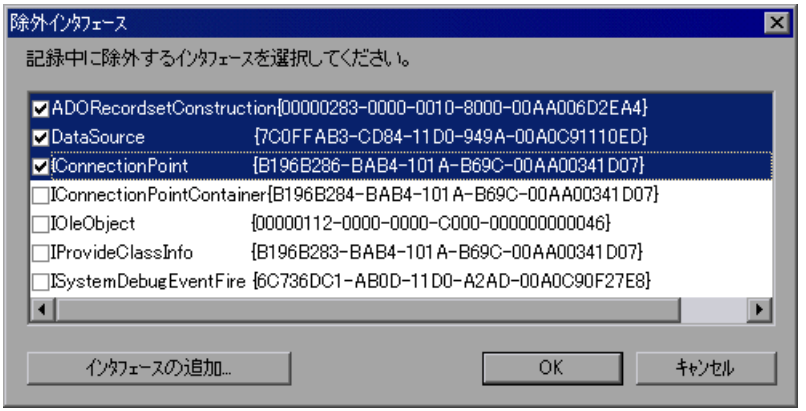


MTS サーバの名前を入力して［**接続**］をクリックします。MTS のコンポーネントを記録するには，マシンに MTS クライアントをインストールしておく必要があります。

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。



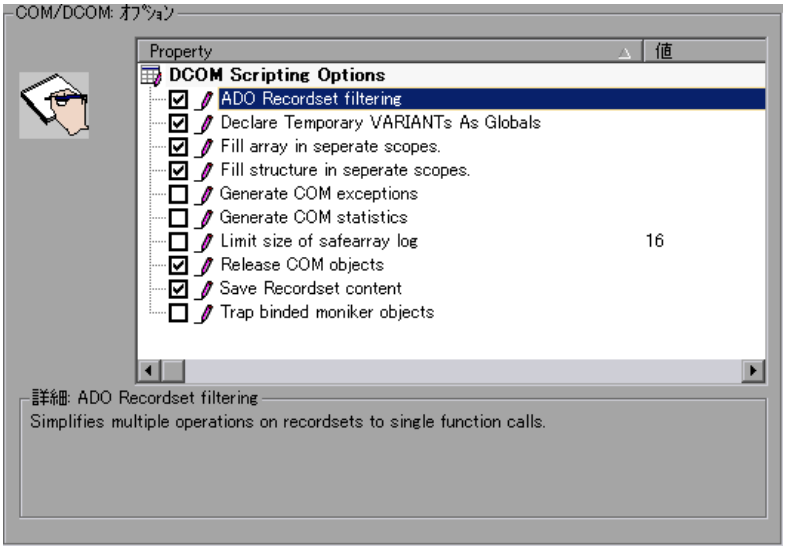
［除外インターフェース］ダイアログ・ボックスでチェックの印が付いているインタフェースが除外されます。表示されていないインタフェースを追加することもできます。［除外インタフェース］ダイアログ・ボックスで［**インタフェースの追加 ...**］をクリックし，GUID 番号（インタフェース ID）とインタフェース名を入力します。VuGen によって作成され，VuGen 画面の左側のカラムの選択ツリーに表示されている interfaces.h ファイルから，GUID をコピーすることもできます。［**インタフェースの追加 ...**］は，スクリプトによって不必要に呼び出されたものの，フィルタに表示され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 呼び出しごとのセーフ配列のログに出力される要素の数を 16 に制限します（標準設定では無効）。
- ［**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モニカ・オブジェクトをすべてトラップします（標準設定では無効）。

COM スクリプトのオプションを設定するには，次の手順を実行します。

1 メイン・メニューから［**ツール**］＞［**記録オプション**］を選択するか，［記録開始］ダイアログ・ボックスの［**オプション**］をクリックします。記録オプション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 40 章

COM 仮想ユーザ・スクリプトの詳細

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 39 章「COM 仮想ユーザ・スクリプトの記録」を参照してください。

本章では，次の項目について説明します。

- 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，次の通りです。

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。

クエリの結果はグリッドに表示されます。グリッドをスクロールすれば，最高 200 レコードまで見ることができます。詳細については，537 ページ「グリッドを使った作業」を参照してください。

COM スクリプトが行う基本的な処理

基本的な処理として，次のことを行います。

- オブジェクトのインスタンスの作成
- インタフェース・ポインタの取得
- インタフェース・メソッドの呼び出し

それぞれの処理は，独立のスコープで実行されます。

オブジェクトのインスタンスの作成

アプリケーションは，COM オブジェクトを使用するために，最初にインスタンスを作成し，そのオブジェクトのインタフェースへのポインタを取得します。

VuGen では，次の手順でオブジェクトのインスタンスが作成されます。

1 VuGen によって **lrc_GUID** が呼び出され，そのオブジェクト用の一意の ProgID が取得されて **pClsid** に格納されます。

```
GUID pClsid = lrc_GUID("student.student.1");
```

pClsid は，ProgID の「**student.student.1**」から変換された，オブジェクトの一意のグローバル CLSID です。

2 IUnknown インタフェース・ポインタが，集約オブジェクトへのポインタである場合，VuGen によってそのオブジェクトへのポインタが取得されます。取得できない場合は，VuGen によってポインタが NULL に設定されます。

```
IUnknown * pUnkOuter = (IUnknown*)NULL;
```

3 VuGen によって，作成するオブジェクトのコンテキストが設定されます。

```
unsigned long dwClsContext = lrc_ulong("7");
```

dwClsContext には，オブジェクトのコンテキストが含まれます（プロセス，ローカル，リモート，またはこれらのうちの複数の場所）。

4 VuGen によって，要求されたインタフェース ID を保持する変数が設定されます。この例では，**IUnknown** です。

```
GUID riid = IID_IUnknown;
```

riid には，**IUnknown** インタフェースのインタフェース ID が含まれます。

5 入力パラメータが用意された後に，**lrc_CoCreateInstance** への呼び出しにより，用意されたパラメータを使ってオブジェクトが作成されます。**IUnknown** インタフェースへのポインタがパラメータ IUnknown_0 に割り当てられます。このポインタは，次の呼び出しに必要です。

```
lrc_CoCreateInstance(&pClsid, pUnkOuter, dwClsContext, &riid,
(void**)&IUnknown_0, CHECK_HRES);
```

前述のとおり入力パラメータが用意されています。呼び出しが成功しているため，VuGen によって **CHECK_HRES** 値が挿入され，ユーザのシミュレーションの実行時にエラーの検査を行うように設定されます。呼び出しの結果，**IUnknown_0** に IUnknown インタフェースへのポインタが返されます。IUnknown_0 は，以降の呼び出しで使用されます。

インタフェースの取得

オブジェクトを作成した時点で VuGen がアクセスできるのは **IUnknown** インタフェースだけです。VuGen は，オブジェクトと通信するために **IUnknown** インタフェースを使用します。これは，**IUnknown** 標準インタフェースの **QueryInterface** メソッドを使って行われます。VuGen のメソッド呼び出しの最初のパラメータは，インタフェースのインスタンスです。この例では，最初のパラメータは事前に **CoCreateInstance** によって設定された **IUnknown_0** ポインタです。**QueryInterface** 呼び出しには，取得すべきインタフェースの ID が入力項目として必要です。QueryInterface 呼び出しによって，指定した ID に対応するインタフェースへのポインタが返されます。

インタフェースを取得するには，次の手順を実行します。

1 最初に VuGen によって Istudent インタフェースの ID のパラメータである **riid** が設定されます。

```
GUID riid = IID_Istudent;
```

2 **QueryInterface** を呼び出すと，Istudent オブジェクトに **Istudent** インタフェースがあれば，出力パラメータ Istudent_0 にそのインタフェースへのポインタが割り当てられます。

```
lrc_IUnknown_QueryInterface(IUnknown_0, &riid, (void**)&Istudent_0,
CHECK_HRES);
```

インタフェースを使ったデータの設定

インタフェースのメソッドを使って，データを設定する例を以下に示します。たとえば，アプリケーションでユーザが名前を入力するように求められるとします。この場合，名前を設定するためのメソッドが起動されます。VuGen によって 2 つのステートメントが記録されます。1 つは，名前の文字列を組み立てるために使用され，もう 1 つは名前のプロパティを設定します。

この関数呼び出し全体を組み立てるには，次の手順を実行します。

1 最初に，VuGen によって変数（**Prop Value**）に，入力された文字列と同じ値が設定されます。パラメータのタイプは，COM ファイルで使用される文字列型である BSTR です。

```
BSTR PropValue = lrc_BSTR("John Smith");
```

後で，「John Smith」をパラメータで置き換えることによって，この呼び出しをパラメータ化すると便利です。これによって，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。

```
lrc_Istudent_get_name(Istudent_0, &pVal, CHECK_HRES);
```

3 次に，VuGen によってこれらの値を保存するためのステートメントが生成されます。

```
//lrc_save_BSTR("param-name",pVal);
```

このステートメントは，コメントアウトされています。コメントを表す記号（//）を削除して，< param-name > を変数に変更できます。変数には，この値を格納することを表すような名前を付けることができます。VuGen によって，直前の呼び出しによって返された **pVal** の値を保存するために，この変数が使用されます。以降，パラメータ化した入力項目として，他のメソッドへの呼び出しでこの変数を使用できます。

IDispatch インタフェース

ほとんどの COM オブジェクトには，固有のインタフェースがあります。また多くは，汎用インタフェース **IDispatch** も実装しています。このインタフェースは VuGen によって特別な方法で変換されます。IDispatch は，IDispatch 以外の COM オブジェクト・インタフェースおよびメソッドをすべて公開する「特別なインタフェース」です。VuGen スクリプトからの **IDispatch:Invoke** メソッドへの呼び出しは，**lrc_Disp** 関数を使って実装されます。これらの呼び出しは，他のインタフェースへの呼び出しとは異なる形式で構成されます。

IDispatch インタフェースの **Invoke** メソッドは，メソッドの実行，プロパティ値の取得，プロパティの値またはプロパティの参照の値の設定を行うことができます。標準の **IDispatch:Invoke** メソッドでは，これらのさまざまな用途は **wflags** パラメータで示されます。VuGen では，これらはメソッドの呼び出し，またはプロパティの設定や取得を行う別々のプロシージャ呼び出しとして実装されます。

たとえば，**GetAgentsArray** メソッドを呼び出すための IDispatch への呼び出しは，次のようになります。

```
retValue = lrc_DispMethod1((IDispatch*)IDispatch_0, "GetAgentsArray",
/*locale*/1033, LAST_ARG, CHECK_HRES);
```

上記の呼び出しのパラメータは，次のとおりです。

IDispatch_0	以前に実行された **IUnknown:Queryinterface** メソッドへの呼び出しによって返された **IDispatch** インタフェースへのポインタです。
GetAgentsArray	呼び出すメソッドの名前です。VuGen の実際の動作では，この名前からメソッドの ID が取得されます。
1033	言語ロケールです。
LAST_ARG	引数がこれ以上ないことを **IDispatch** インタフェースに知らせるためのフラグです。
CHECK_HRES	記録時に呼び出しが成功したため，HRES の検査を行うことを示すフラグです。

さらに，**OPTIONAL_ARGS** という別のパラメータが存在することもあります。これは，VuGen によって標準パラメータ以外に追加引数が送られていることを示します。追加引数はそれぞれ，ID または名前と，その値の組み合わせで構成されています。たとえば，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 41 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 41 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 41 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。

本項では，相関させる必要のあるステートメントを見つける方法について説明します。すでに相関させたい値がわかっている場合は，次の項に進んで，特定の値を相関させる方法を参照してください。

自動相関で検出されたスクリプトの検索と相関を行うには，次の手順を実行します。

1 ［**表示**］＞［**出力ウィンドウ**］を選択して，ウィンドウの下部に出力タブを表示します。［再生ログ］タブでエラーがないか確認します。多くの場合，これらのエラーは相関によって修正できます。

2 [**仮想ユーザ**] > [**相関をスクリプトの中で検索**] を選択します。VuGen によってスクリプト全体を対象とする検索が行われ，相関候補の値がすべて [**相関クエリー**] タブに表示されます。

3 値を相関させます。[相関クエリー] タブで，相関させるクエリ結果をダブルクリックします。この例では，メッセージの「grid column x, row x」となっている行が該当します。VuGen によって，スクリプト内の値の位置にカーソルが移動します。

4 グリッドの中で，右クリック・メニューから [**相関性を生成**] 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 スクリプト内に現れる値をすべて検索するかどうかを尋ねるメッセージ・ボックスが開きます。選択したステートメント内の値だけを置き換える場合は，[**いいえ**] をクリックします。次の候補を検索して置き換える場合は [**はい**] をクリックします。

7 [検索と置換] ダイアログ・ボックスが開きます。オリジナルのステートメントも含め，すべての置き換えを確認します。

8 [検索と置換] ダイアログ・ボックスを閉じます。ステートメントの値がパラメータへの参照に置き換えられます。相関を取り消すと，直前のステップで作成されたステートメントは消去されます。

既知の値の相関

相関させるべき値がわかっている場合は次の手続きを行います。

特定の値を相関させるには，次の手順を実行します。

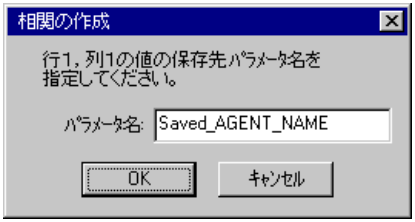
1 相関させる引数を見つけ（通常は **lrc_variant_** ステートメント内にあります），値を選択します（引用符は除きます）。

2 ［**仮想ユーザ**］＞［**相関を検索（カーソル位置）**］を選択します。

VuGen によって値が検索され，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。相関値が［相関クエリー］タブに一覧表示されます。

3 ［相関クエリー］タブで，相関させるクエリ結果をダブルクリックします。この例では，メッセージの「grid column x, row x」となっている行が該当します。VuGen によって，スクリプト内の値の位置にカーソルが移動します。

4 表示枠の中で，相関させる値を選び，［**仮想ユーザ**］＞［**相関を作成**］を選択します。結果の値を保存するパラメータの名前を入力するように求められます。



5 名前を指定するか，標準設定の名前をそのまま使用します。［**OK**］をクリックして作業を続けます。VuGen によって，パラメータに結果を保存する相関ステートメント（**lrc_save_** ＜タイプ＞）が挿入されます。

```
lrc_save_rs_param (Recordset20_0, 1, 1, 0, “Saved_AGENT_NAME”);
```

6 スクリプト内に現れる値をすべて検索するかどうかを尋ねるメッセージ・ボックスが開きます。選択したステートメント内の値だけを置き換える場合は，［**いいえ**］をクリックします。次の候補を検索して置き換える場合は［**はい**］をクリックします。

7 ［検索と置換］ダイアログ・ボックスが開きます。オリジナルのステートメントも含め，すべての置き換えを確認します。

第 41 章

COM 仮想ユーザ関数について

COM 仮想ユーザ関数は，COM アプリケーションを実行するユーザのアクションをエミュレートします。

本章では，次の項目について説明します。

- 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 または Visual Basic）を指定できます。詳細は，第 5 章「スクリプト生成オプションの設定」を参照してください。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。

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

次の呼び出しは，**Invoke** メソッドを使って **IDispatch** インタフェースを起動し，**Invoke** の **wflags** パラメータに，異なるフラグ値を設定します。

lrc_DispMethod	**IDispatch:Invoke** メソッドを使って，インタフェースのメソッドを起動します。
lrc_DispMethod1	メソッドを起動し，IDispatch インタフェースを使って同じ名前のプロパティを取得します。
lrc_DispPropertyGet	IDispatch インタフェースを使って，プロパティを取得します。
lrc_DispPropertyPut	IDispatch インタフェースを使って，プロパティを設定します。
lrc_DispPropertyPutRef	IDispatch インタフェースを使って，参照によってプロパティを設定します。

型指定関数

VuGen が自動的に記録する関数を補うために，スクリプトに手作業のプログラミングで型指定関数を挿入できます。この型変換関数は文字列データを指定した型に代入します。関数名の形式は次のとおりです。

lrc_ ＜型の名前＞

ここで，＜型の名前＞は，次のデータ型のいずれかです。

ascii_BSTR	ascii BSTR
bool	ブール型
BSTR	BSTR
BYTE	バイト型
char	文字変数
currency	通貨
date	日付

double	倍精度
dword	倍精度ワード
float	浮動小数点数
GUID	指定したオブジェクトの GUID
hyper	hyper 型整数
int	整数
long	long 型整数
short	short 型整数
uint	符号なし整数
ulong	符号なし long 型整数
uhyper	符号なし 64 ビット hyper 型整数
ushort	符号なし short 型整数

バリアント型

バリアントには任意の型の情報を含めることができます。たとえば，バリアントは文字列の配列にも倍精度ワードにもなります。また，バリアントの配列をバリアントとすることもできます。VuGen では文字列データをさまざまなバリアント型に変換できます。関数名の形式は，次のとおりです。

lrc_variant_ ＜型の名前＞

ここで，**＜型の名前＞**は，次のデータ型のいずれかです。

ascii BSTR	ascii BSTR のバリアント
bool	ブール型のバリアント
BSTR	BSTR のバリアント
BYTE	符号なし文字（バイト型）のバリアント
char	文字
CoObject	**IUnknown** インタフェース・ポインタ

currency	通貨のバリアント
date	日付のバリアント
DispObject	**IDispatch** インタフェース・ポインタ
float	浮動小数点数のバリアント
int	整数のバリアント
long	long 型整数のバリアント
scode	エラー・コードのバリアント
short	short 型整数のバリアント
uint	符号なし整数のバリアント
ulong	符号なし long 型整数のバリアント
ushort	符号なし short 型整数のバリアント

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
Error	scode 型のエラー
Float	浮動小数点数
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
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 42 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することができます。

本章では，次の項目について説明します。

- Corba-Java 仮想ユーザ・スクリプトの作成について
- 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。

Corba-Java 仮想ユーザ・スクリプトの作成について

VuGen を使って，CORBA（Common Object Request Broker Architecture）Java アプリケーションやアプレットを記録できます。記録を行うと，VuGen によって，ピュア Java を仮想ユーザ API 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** のマニュアル，**Performance Center** のマニュアル，または **Business Availability Center** のマニュアルを参照してください。

スクリプトを記録し，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（**org.omg.CORBA.ORB** など）は，スクリプトを実行するマシンに存在している必要があり，**classpath** 環境変数に指定されている必要があります。関数の構文とシステムの設定で留意すべき点については，第 38 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，672 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

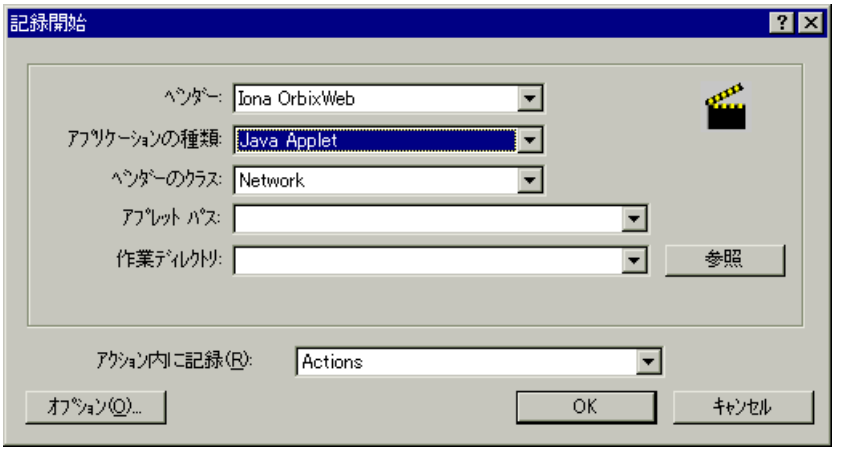
CORBA-Java 仮想ユーザの記録を開始する前に，アプリケーションまたはアプレットが記録用のマシンで正しく動作することを確認します。

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 38 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 ［ベンダー］リストから CORBA ベンダーを選択します。

3 ［**アプリケーションの種類**］ボックスで，以下の選択肢から適切な値を選択します。

［**Java アプレット**］: Sun のアプレットビューアを使って Java アプレットを記録します。

［**Java アプリケーション**］: Java アプリケーションを記録します。

［**Netscape**］または［**IExplore**］: ブラウザ内のアプレットを記録します。

［**Executable¥Batch**］: バッチ・ファイルから起動されるアプレットまたはアプリケーションを記録します。

［**リスナ**］: 記録の前に設定を初期化してアプリケーションを実行するバッチ・ファイルを待機するよう VuGen に指示します。このモードでは，JDK 1.2.x 以上を使って，システム変数 _JAVA_OPTIONS に値 **--Xrunjdkhook** を設定しなければなりません（JDK 1.1.x の場合，環境変数 **_classload_hook=JDKhook** を定義します）。

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［**ネットワーク**］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，［**ローカル**］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 アプレット	［アプレット パス］，［作業ディレクトリ］
Java アプリケーション	［アプリケーション メイン］，［作業ディレクトリ］，［アプリケーション パラメータ］
IExplore	［IExplore パス］，［URL］
Netscape	［Netscape パス］，［URL］
Executable¥Batch	［実行可能 ¥ バッチ］，［作業ディレクトリ］
リスナ	なし

注：作業ディレクトリを指定する必要があるのは，アプリケーションに作業ディレクトリの場所を指定する必要がある（たとえば，プロパティ・ファイルの読み込みや，ログ・ファイルの作成をする）場合だけです。

---

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 27 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。

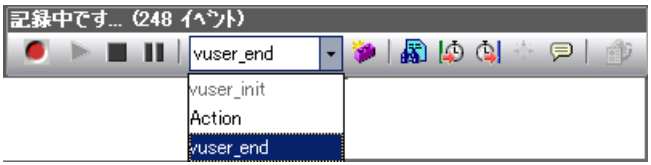
Actions クラス内のメソッド	記録対象アクション	エミュレーション内容	実行のタイミング
init	**vuser_init**	サーバへのログイン	初期化時
action	**Actions**	クライアントの動作	実行中
end	**vuser_end**	ログオフ処理	終了時または停止時

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［**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。［仮想ユーザを保存］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;
}
```

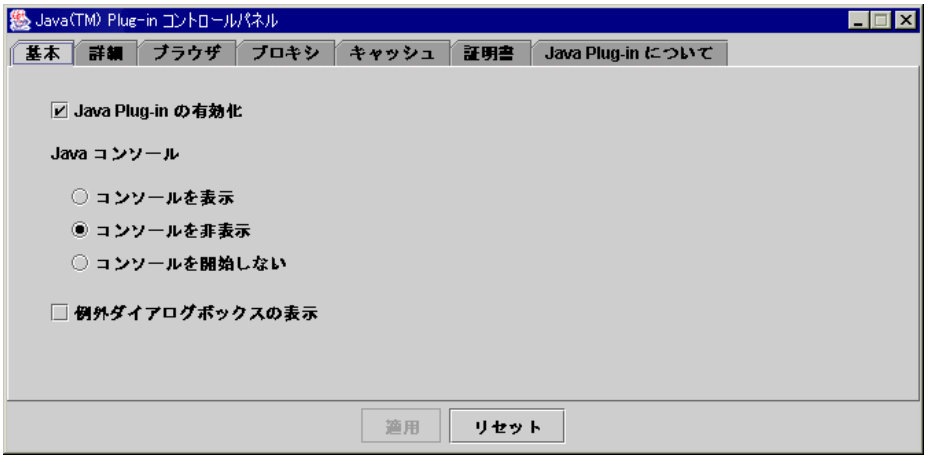
ORB シャットダウン・ステートメントは本製品用にカスタマイズされています。このカスタマイズは，1 つの仮想ユーザのシャットダウンによって他のすべての仮想ユーザがシャットダウンされないようにするものです。

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，［**Microsoft Internet Explorer**］チェック・ボックスを選択しなおして，［**適用**］をクリックします。

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 Corba クラスを必ず使用するようにするには，コマンド・ラインで次の java.exe パラメータを指定します。

Visigenic 3.4

-Dorg.omg.CORBA.ORBClass=com.visigenic.vbroker.orb.ORB
-Dorg.omg.CORBA.ORBSingletonClass=com.visigenic.vbroker.orb.
ORBSingleton

Visigenic 4.0

-Dorg.omg.CORBA.ORBClass=com.inprise.vbroker.orb.ORB
-Dorg.omg.CORBA.ORBSingletonClass=com.inprise.vbroker.orb.ORBSingleton

OrbixWeb 3.x

-Dorg.omg.CORBA.ORBClass=IE.Iona.OrbixWeb.CORBA.ORB
-Dorg.omg.CORBA.ORBSingletonClass=IE.Iona.OrbixWeb.CORBA.
singletonORB

OrbixWeb 2000

-Dorg.omg.CORBA.ORBClass=com.iona.corba.art.artimpl.ORBImpl
-Dorg.omg.CORBA.ORBSingletonClass=com.iona.corba.art.artimpl.
ORBSingleton

第 43 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，次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。記録を行うと，VuGen によって，ピュア Java を仮想ユーザ API 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** のマニュアル，**Performance Center** のマニュアル，または **Business Availability Center** 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している必要があり，**classpath** 環境変数に指定されている必要があります。関数の構文とシステムの設定で留意すべき点については，第 38 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，672 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

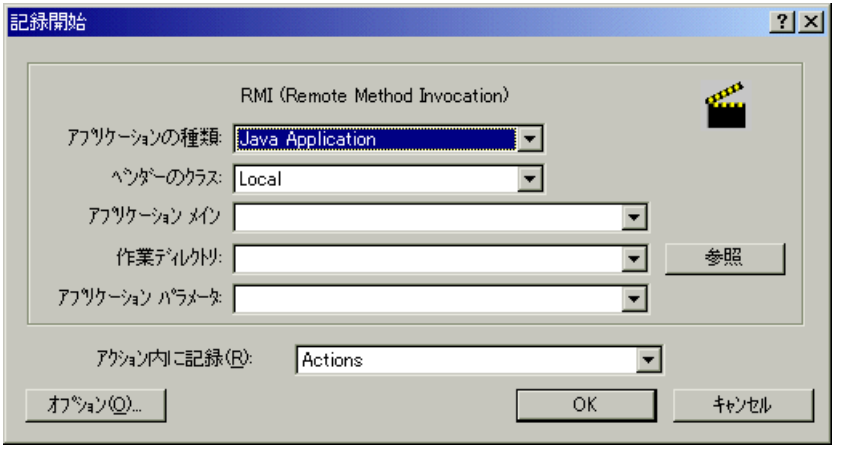
RMI 仮想ユーザを記録する前に，記録するマシンでアプリケーションまたはアプレットが正しく機能することを確認してください。

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 38 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 アプレット**］: Sun のアプレットビューアを使って Java アプレットを記録します。

［**Java アプリケーション**］：Java アプリケーションを記録します。

［**Netscape**］または［**IExplore**］：ブラウザ内のアプレットを記録します。

［**Executable¥Batch**］：バッチ・ファイルから起動されるアプレットまたはアプリケーションを記録します。

［**リスナ**］：記録の前に設定を初期化してアプリケーションを実行するバッチ・ファイルを待機するよう VuGen に指示します。このモードでは，JDK 1.2.x 以上を使って，システム変数 _JAVA_OPTIONS に値 **--Xrunjdkhook** を設定しなければなりません（JDK 1.1.x の場合，環境変数 **_classload_hook=JDKhook** を定義します）。

3 ［**ベンダーのクラス**］ボックスで［**ネットワーク**］または［**ローカル**］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 アプレット	［アプレット パス］，［作業ディレクトリ］
Java アプリケーション	［アプリケーション メイン］，［作業ディレクトリ］，［アプリケーション パラメータ］
IExplore	［IExplore パス］，［URL］
Netscape	［Netscape パス］，［URL］
Executable¥Batch	［実行可能 ¥ バッチ］，［作業ディレクトリ］
リスナ	なし

作業ディレクトリを指定する必要があるのは，アプリケーションに作業ディレクトリの場所を指定する必要がある（たとえば，プロパティ・ファイルの読み込みや，ログ・ファイルの作成をする）場合だけです。

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 27 章「Java 記録オプションの設定」を参照してください。

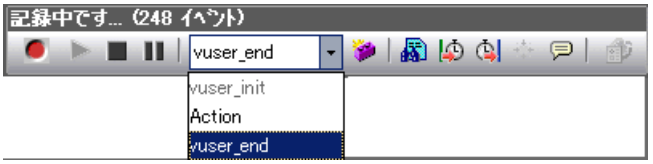
6 ［**アクション内に記録**］ボックスで，記録を挿入するセクションを選択します。Actions クラスには，**init**，**action**，**end** の 3 つのメソッドが含まれています。次の表に，各メソッドに何を含め，どのタイミングで呼び出されるかを示します。

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	実行中
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，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。

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］ツールバーで，［**停止**］ボタンをクリックします。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。［仮想ユーザを保存］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。

RMI 仮想ユーザ・スクリプトを使った作業

本項では，RMI 仮想ユーザ特有の Java 仮想ユーザ・スクリプトの要素について説明します。RMI は CORBA のような構造要素はなく，Serializable Java オブジェクトを使用します。最初のセクションでは，ネーミング・レジストリの初期化と設定を行います。次のセクションは，Java オブジェクト（Remote および Serializable）が見つかり，キャストされた場合に生成されます。その次のセクションには，Java オブジェクトを対象とするサーバ呼び出しが含まれます。RMI は CORBA とは異なり，専用のシャットダウン・セクションがありません。スクリプトの中でオブジェクトが何度も現れることがあります。

次に示すコードでは，ネーミング・レジストリを検索しています。この処理の後に，特定の Java オブジェクトを取得するための検索処理が行われます。その後，オブジェクトを使って **set_sum**，**increment**，**get_sum** といった呼び出しを実行できます。このコード例は，VuGen で必要とされるすべての RMI クラスをどのようにインポートするかを示します。

```
Import java.rmi.*;
Import java.rmi.registry.*;

:
:

// public 関数：action
public int action() throws Throwable {

    _registry = LocateRegistry.getRegistry("localhost",1099);

    counter = (Counter)_registry.lookup("Counter1");

    counter.set_sum(0);
    counter.increment();
    counter.increment();
    counter.get_sum();

    return lr.PASS;
    }
:
```

RMI 仮想ユーザを記録する際，スクリプトにはすべての関連オブジェクトのシリアル化を解除する **lr.deserialize** への複数の呼び出しが含まれていることがあります。**lr.deserialize** 呼び出しは，次の呼び出しに渡されるオブジェクトが，それ以前の呼び出しの戻り値と相関できない場合に生成されます。この場合，VuGen によってオブジェクトの状態が記録され，再生時に **lr.deserialize** 呼び出しを使って，オブジェクトの値が提示されます。シリアライズの解除は，VuGen によってオブジェクトがパラメータとして呼び出しに渡される前に行われます。詳細については，452 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 9 部

E - ビジネス・プロトコル

第 44 章

AMF 仮想ユーザ・スクリプトの作成

VuGen では，AMF 形式を使用する Flash Remoting をエミュレートする仮想ユーザを作成できます。

本章では，次の項目について説明します。

- AMF 仮想ユーザ・スクリプトの作成について
- AMF の用語について
- AMF 記録モードの設定
- AMF 関数を使った作業
- AMF スクリプトの相関
- AMF データの表示
- AMF スクリプトについて

以降の情報は，AMF 仮想ユーザ・スクリプトのみを対象とします。

AMF 仮想ユーザ・スクリプトの作成について

多くのクライアント・アプリケーションは，RPC（リモート・プロシージャ・コール）を使用してサーバと通信します。しかし，RPC では，インターネットを介して作業する場合，互換性およびセキュリティ上の問題が生じます。たいていの場合，ファイアウォールやプロキシ・サーバは，このタイプのトラフィックをブロックします。

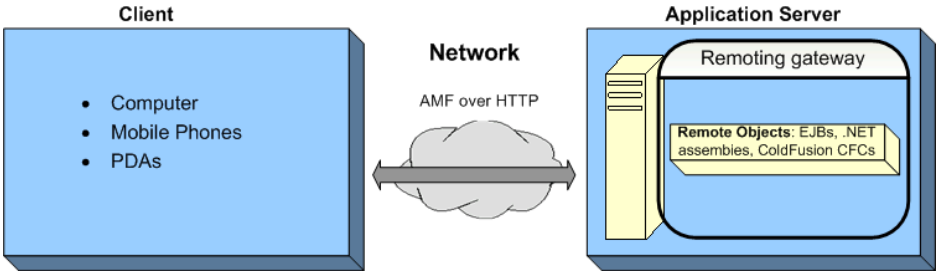
HTTP は，すべてのインターネット・ブラウザおよびサーバでサポートされます。したがって，インターネット経由で作業する場合，HTTP は，クライアント・アプリケーションとサーバ間の通信における好ましい方法であるといえます。

SOAP（XML ベースの形式）は，HTTP を使用してアプリケーション間の通信を行うための安全な方法を提供します。しかし，メッセージがテキスト・ベースであるため，Flash ファイルなどの大きなメッセージやほかの RIA（Rich Internet Application）を使用して作業する場合，SOAP では非効率的です。

この非効率性を解決するため，Macromedia は，HTTP を使用してバイナリ形式で通信する独自のプロトコル，すなわち AMF（Action Messaging Format）を開発しました。バイナリ形式の AMF データ・セットは，SOAP のテキスト・ベースの XML よりもサイズが大幅に小さくなります。

サーバに AMF メッセージを送信する代表的なクライアント・アプリケーションの例としては，パーソナル・コンピュータ上で Flash クリップを再生する Flash Player があります。Flash Player は，ゲートウェイを経由してアプリケーション・サーバにネイティブな Flash オブジェクトを送信します。ゲートウェイ（**Flash Remoting** ゲートウェイとも呼ばれます）は，Java（ColdFusion を含む）または .NET サーバにインストールされているサーバ側オブジェクトです。ゲートウェイは，Flash Player とサーバの間で要求を処理するブローカとして機能します。Flash オブジェクトをサーバのネイティブ・オブジェクトに変換し，それらをサーバ側の適切なサービスに渡します。

結果が返されると，ゲートウェイはその結果をシリアル化してネイティブ Flash オブジェクトに変換し，AMF を使用して Flash クライアントに送ります。



AMF トラフィックをエミュレートする仮想ユーザ・スクリプトを作成するには，［**e ビジネス**］カテゴリから［**AMF**］プロトコル・タイプを選択します。記録を開始するには，［**記録**］ボタンをクリックし，Web サーバに対する標準的なアクションを実行します。スクリプトの作成および記録の詳細については，第 4 章「VuGen を使った記録」を参照してください。

AMF スクリプトを作成した後，スクリプトを自環境（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**』，**Performance Center** のマニュアル，または **Business Availability Center** のマニュアルを参照してください。

AMF の用語について

次の表に，AMF に関係する，よく使用される用語の定義を示します。

用語 / 略語	説明
ActionScript	Macromedia Flash ムービーおよびアプリケーションの制御に使用されるスクリプト・プログラミング言語です。この構文は JavaScript に似ています。
AMF	Flash Remoting に使用される独自のバイナリ通信プロトコルです。
Flash Remoting	Flash Remoting では，AMF 形式を使用して，Flash Player とアプリケーション・サーバ間でデータを交換できます。
Flex	RIA（Rich Internet Application）を生成するための Macromedia のアプリケーション・サーバです。
SOAP	通常 HTTP を使用し，コンピュータ・ネットワーク経由で XML ベースのメッセージを交換するための標準です。

AMF 記録モードの設定

AMF プロトコルおよび Web プロトコルを使用して Flash Remoting セッションからスクリプトを生成する方法を VuGen に指定できます。オプションは次のとおりです。

- AMF および Web
- AMF のみ
- Web のみ

標準設定では，VuGen は AMF 呼び出しのみを生成します。記録対象プロトコルを設定するには，［記録オプション］を開き（［**ツール**］＞［**記録オプション**］），［**一般：プロトコル**］ノードを選択します。詳細については，63 ページ「プロトコルの追加と削除」を参照してください。

前述のいずれかのオプションで記録した場合でも，後でスクリプトを再生成して必要なプロトコルが含まれるように，または除外されるようにできます。スクリプトの再生成の詳細については，80 ページ「仮想ユーザ・スクリプトの再生成」を参照してください。

AMF および Web

AMF プロトコルと Web プロトコルの両方を有効にした場合，VuGen はビジネス・プロセス全体の関数を生成します。AMF データに遭遇すると，適切な AMF 関数を生成します。

次のコードの抜粋では，VuGen は Web 関数（**web_url**）と AMF 関数（**amf_call**，**amf_define_envelope_header_set**）の両方を生成しています。

```
web_url("flash",
        "URL=http://testlab:8200/flash/", "Resource=0",
        ...
        "Snapshot=t1.inf",
        EXTRARES,
        "Url=movies/XMLExample.swf", "Referer=", ENDITEM,
        "Url=movies/JavaBeanExample.swf", "Referer=", ENDITEM,
        LAST);

web_link("Sample JavaBean Movie Source",
        "Text=Sample JavaBean Movie Source",
        "Snapshot=t2.inf",
        EXTRARES,
        "Url=XMLExample.swf", "Referer=", ENDITEM,
        "Url=JavaBeanExample.swf", "Referer=", ENDITEM,
        LAST);

amf_set_version("0");

amf_define_header_set("Id=amf_header_set",
        HEADER,
        "Name=amf_server_debug",
        "MustUnderstand=true",
        "Data=<object><boolean
key=¥"coldfusion¥">true</boolean>                <boolean key=¥""
"amfheaders¥">false</boolean>...
        LAST);

amf_call("flashgateway.samples.FlashJavaBean.testDocument",
        "Gateway=http://testlab:8200/flashservices/gateway",
        "AMFHeaderSetId=amf_header_set",
        "Snapshot=t3.inf",
        MESSAGE,
        "Method=flashgateway.samples.FlashJavaBean.testDocument",
        "TargetObjectId=/1",
        BEGIN_ARGUMENTS,
        "<xmlString><![CDATA[<TEST
message=¥"test¥"><INSIDETEST/>        </TEST>]]></""xmlString>",
        END_ARGUMENTS,
        LAST);
```

AMF のみ

Flash Remoting をエミュレートするのに AMF 呼び出しのみを必要とする場合は，Web 呼び出しを無効にして，AMF 呼び出しのみを生成できます。

次の例は，AMF プロトコルを有効にし，Web プロトコルを無効にして，前述のセッションを記録したものです。

```
Actions()
{
amf_set_version("0");

amf_define_header_set("Id=amf_header_set",
      HEADER,
      "Name=amf_server_debug",
      "MustUnderstand=true",
      "Data=<object><boolean
key=¥"coldfusion¥">true</boolean>                <boolean key=¥""
"amfheaders¥">false</boolean>...
      LAST);

amf_call("flashgateway.samples.FlashJavaBean.testDocument",
      "Gateway=http://testlab:8200/flashservices/gateway",
      "AMFHeaderSetId=amf_header_set",
      "Snapshot=t3.inf",
      MESSAGE,
      "Method=flashgateway.samples.FlashJavaBean.testDocument",
      "TargetObjectId=/1",
      BEGIN_ARGUMENTS,
      "<xmlString><![CDATA[<TEST message=¥"test¥"><INSIDETEST
       /></TEST>]]></"
      "xmlString>",
      END_ARGUMENTS,
      LAST);
…
```

なお，この記録は，完全なビジネス・プロセスを示すものではありません。AMF を使用する Flash Remoting 呼び出しの例を示しているに過ぎません。

Web のみ

［Web のみ］のオプションは，代替として Web HTTP 技術を使用します。AMF 呼び出しは生成されません。代わりに，Flash Remoting 情報を持つ **web_custom_request** 関数が生成されます。

次の例は，前述のセグメントを AMF 関数なしで再生成したものです。関数の引数に AMF バイナリ文字列が含まれている点に注目してください。

```
web_url("flash",
        "URL=http://testlab:8200/flash/",
        "Resource=0",
        "RecContentType=text/html",
        "Referer=",
        "Snapshot=t1.inf",
        "Mode=HTML",
        EXTRARES,
        "Url=movies/XMLExample.swf", "Referer=", ENDITEM,
        "Url=movies/JavaBeanExample.swf", "Referer=", ENDITEM,
        LAST);

web_link("Sample JavaBean Movie Source",
        "Text=Sample JavaBean Movie Source",
        "Snapshot=t2.inf",
        EXTRARES,
        "Url=XMLExample.swf", "Referer=", ENDITEM,
        "Url=JavaBeanExample.swf", "Referer=", ENDITEM,
        LAST);

web_custom_request("gateway",
        "URL=http://testlab:8200/flashservices/gateway",
        "Method=POST",
        "Resource=0",
        "RecContentType=application/x-amf",
        "Referer=",
        "Snapshot=t3.inf",
        "Mode=HTML",
        "EncType=application/x-amf",

"BodyBinary=¥¥x00¥¥x00¥¥x00¥¥x01¥¥x00¥¥x10amf_server_debug¥¥x01
¥¥x00¥¥x00¥¥x00`¥¥x03¥¥x00¥ncoldfusion¥¥x01¥¥x01¥¥x00¥namfheader
s¥¥x01¥¥x00¥¥x00¥¥x03amf¥¥x01¥¥x00¥¥x00¥¥x0Bhttpheaders¥¥x01¥¥x
00¥¥x00¥trecordset¥¥x01¥¥x01¥¥x00¥¥x05error¥¥x01¥¥x01¥¥x00¥¥x05tra
ce¥¥x01¥¥x01¥¥x00¥¥x07m_debug¥¥x01¥¥x01¥¥x00¥¥x00¥t¥¥x00¥¥x01¥
¥x00/flashgateway.samples.FlashJavaBean.testDocument¥¥x00¥¥x02/1¥¥
x00¥¥x00¥¥x004¥n¥¥x00¥¥x00¥¥x00¥¥x01¥¥x0F¥¥x00¥¥x00¥¥x00*<TES
T message=¥"test¥"><INSIDETEST /></TEST>",
            LAST);
```

AMF 関数を使った作業

AMF 仮想ユーザ・スクリプト関数は Flash Remoting セッションをエミュレートします。各関数の先頭には，**amf** という接頭辞が付きます。

関数は次のとおりです。

関数名	説明
amf_call	AMF 要求を送信します。
amf_define_envelope_header_set	エンベロープのヘッダー・セットを定義します。
amf_define_header_set	AMF のヘッダー・セットを定義します。
amf_set_version	AMF のバージョン番号を設定します。

amf_define_header_set は，AMF 呼び出し全体のヘッダーを定義するのに対し，**amf_define_envelope_header_set** は，特定のメッセージのみを対象としてエンベロープ・ヘッダーを定義します。

次の例では，**amf_define_header_set** 関数がヘッダー・セットを定義しています。**amf_call** 関数はゲートウェイにアクセスし，サーバにメッセージを送信しています。

```
amf_define_header_set("Id=amf_header_set",
      HEADER,
      "Name=amf_server_debug",
      "MustUnderstand=true",
      "Data=<object><boolean
key=¥"coldfusion¥">true</boolean>                <boolean key=¥""
"amfheaders¥">false</boolean>...
      LAST);

amf_call("flashgateway.samples.FlashJavaBean.testDocument",
      "Gateway=http://testlab:8200/flashservices/gateway",
      "AMFHeaderSetId=amf_header_set",
      "Snapshot=t3.inf",
      MESSAGE,
      "Method=flashgateway.samples.FlashJavaBean.testDocument",
      "TargetObjectId=/1",
      BEGIN_ARGUMENTS,
      "<xmlString><![CDATA[<TEST message=¥"test¥"><INSIDETEST
       /></TEST>]]></"
      "xmlString>",
      END_ARGUMENTS,
      LAST);
```

これらの関数の詳細な構文については，「**オンライン関数リファレンス**」（[**ヘルプ**] > [**関数リファレンス**]）を参照してください。

AMF スクリプトの相関

たいていの場合，Flash アプリケーションには，スクリプトを実行するたびに変わる動的なデータが含まれています。たとえば，あるサーバでは，現在の日時で構成されるリンクを使用しています。あるいは，実行のたびにオブジェクト名が変わるかもしれません。

仮想ユーザ・スクリプトを記録するとき，VuGen はデータと引数値のセットを記録します。しかし，スクリプトの再生時に，サーバがこの引数を拒否し，エラーを発行することがあります。このエラーは，データが古くなってサーバで受け入れられなくなった動的なデータが原因であると考えられます。

これを回避するには，スクリプトの相関を行います。

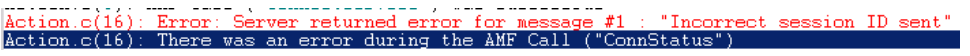
- 必要な値を取り出す準備として，サーバからの応答を保存します。
- サーバの応答から必要な値を取り出します。
- 値をパラメータに保存します。
- それらのパラメータを AMF 要求への入力として使用します。

このようなエラーは必ずしもはっきり分かるわけではなく，仮想ユーザ・ログ・ファイルを注意深く調べないと検出できないことがあります。仮想ユーザの実行時にエラーが発生した場合は，スクリプト内でのエラーの発生箇所を調べてください。多くの場合，相関によって，1 つのステートメントの結果を別のステートメントの入力項目として使用することで，問題を解決できます。

相関を実行するには，次の手順を実行します。

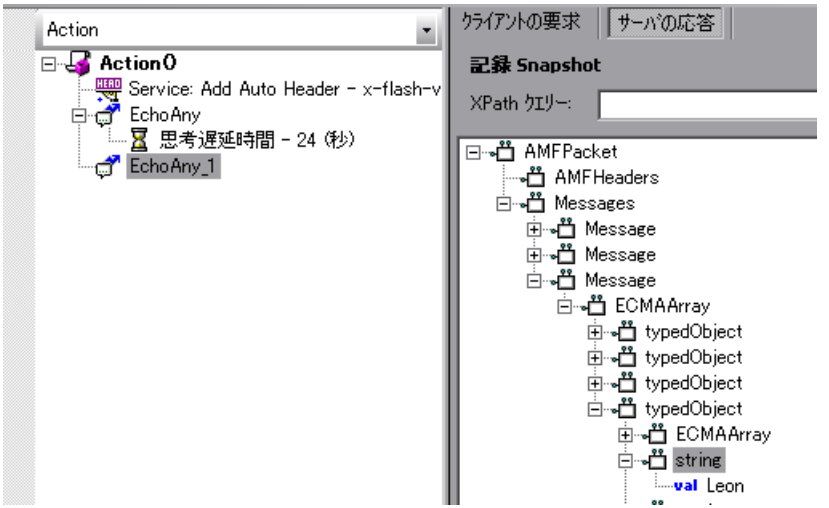
1 **動的な値が原因で失敗し，相関が必要となったスクリプト内のステップを探します。**

［再生ログ］を利用して問題のステップを探します。



2 先行するステップの中にある，正しい値を持っているサーバ応答を探します。

先行するステップをツリー・ビューの中で調べて，［**サーバの応答**］タブ内で値を探します。



3 サーバ応答全体をパラメータに保存します。

値を取り出す前に，サーバ応答全体をパラメータに次のように保存します。

- 対象の値を含んでいるサーバ応答に対応するステップ・ノード（[Action] 表示枠内）を右クリックし，［**プロパティ**］を選択します。
- ［AMF Call Properties］ダイアログの中で，**応答パラメータ**の名前を入力します。詳細については，702 ページ「[AMF Call Properties]」を参照してください。
- ［**OK**］をクリックして，新しいパラメータ名を保存します。

4 元のサーバ応答値をパラメータに保存します。

- ［Server Response］の XML ツリーの中で，値の上のノード（たとえば，string）を右クリックして［**Save value in parameter**］を選択します。

- ［XML Parameter Properties］ダイアログの中で，**Name** にパラメータ名を入力します。以降のステップでは，この名前を使用することになります。

➤ ［**OK**］をクリックします。これで，スクリプトに **lr_xml_get_values** という新しい関数が追加されます。

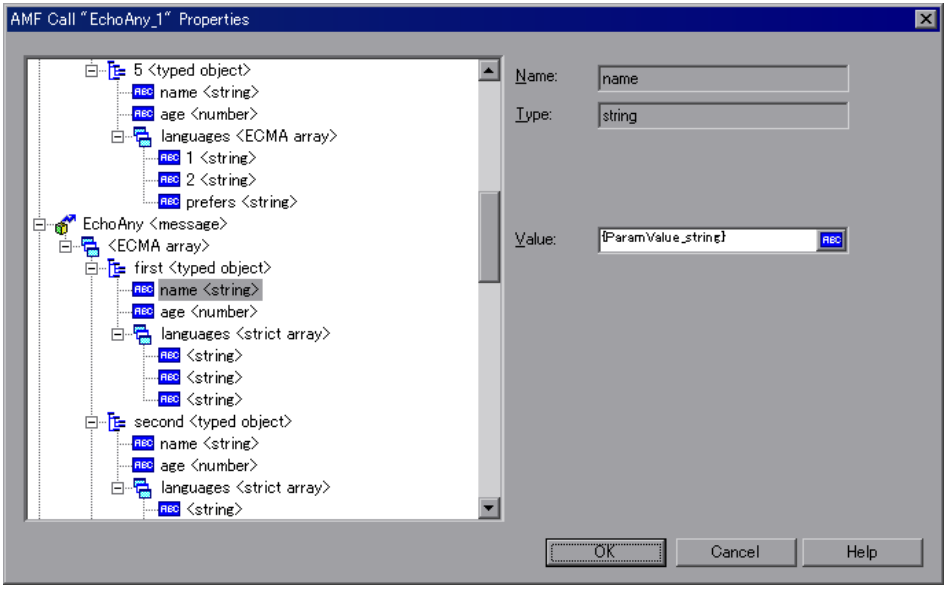
5 **パラメータを以降の呼び出しに挿入します。**

VuGen の編集ビューで，失敗した呼び出しを開始点に，オブジェクトに対する以降のすべての呼び出しに含まれている値を，定義したパラメータで置き換えます。

➤ 失敗している呼び出しに対応するステップ・ノード（［アクション］表示枠内）を右クリックし，［**プロパティ**］を選択します。

➤ 相関を必要としていた引数を探します。

➤ ［値］フィールドに，**{ParamValue_string}** のようにパラメータ名を中括弧で囲んで入力します。



➤ ［**OK**］をクリックします。

6 **スクリプトを実行します。**

引数値が，保存したパラメータ値で適切に置き換えられていることを確認します。

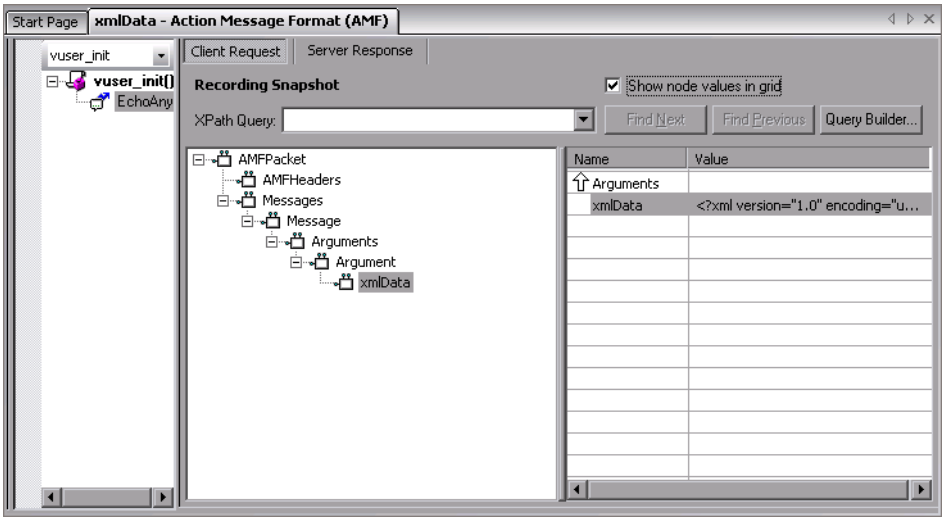
次の例では，**lr_xml_get_values** を使用して応答から値を取得し，**ParamValue_string** というパラメータを作成しています。この ParamValue_string パラメータは，次の **amf_call** 関数で使用しています。

```
amf_call(
	"ConnectService",
	"Gateway=http://labm1app08/AMF/EchoAMF/gateway.aspx",
	"Snapshot=t6.inf",
	"ResponseParameter=resp",
	MESSAGE,
	"Method=EchoAMF.SpecialCases.ConnectService",
	"TargetObjectId=/1",
	BEGIN_ARGUMENTS,
	END_ARGUMENTS,

	LAST);

lr_xml_get_values("XML={resp}",
	"FastQuery=/AMFPacket/Messages/Message/string",
	"ValueParam=ParamValue_string",
	LAST);

amf_call(
	"ConnStatus",
	"Gateway=http://labm1app08/AMF/EchoAMF/gateway.aspx",
	"Snapshot=t7.inf",
	MESSAGE,
	"Method=EchoAMF.SpecialCases.ConnStatus",
	"TargetObjectId=/2",
	BEGIN_ARGUMENTS,
	"<string>{ParamValue_string}</string>",
	END_ARGUMENTS,

	LAST);
```

Web 仮想ユーザの相関の詳細については，第 57 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。

AMF データの表示

AMF データを XML 形式で表示するには，該当するステップをツリー・ビューの中で選択します。VuGen の右表示枠に，クライアントの要求とサーバの応答が XML 形式で表示されます。



メッセージのすべての要素を表示するには，XML のすべてのノードを確実に展開します。XML 引数値をグリッドに表示するには，［**Show node values in grid**］を選択します。

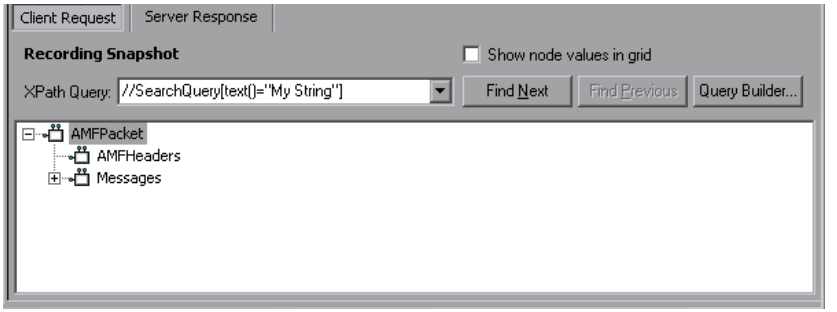
XML ツリーに対するクエリ

VuGen は，XML に対するクエリを作成して実行することが可能なクエリ・ビルダを備えています。

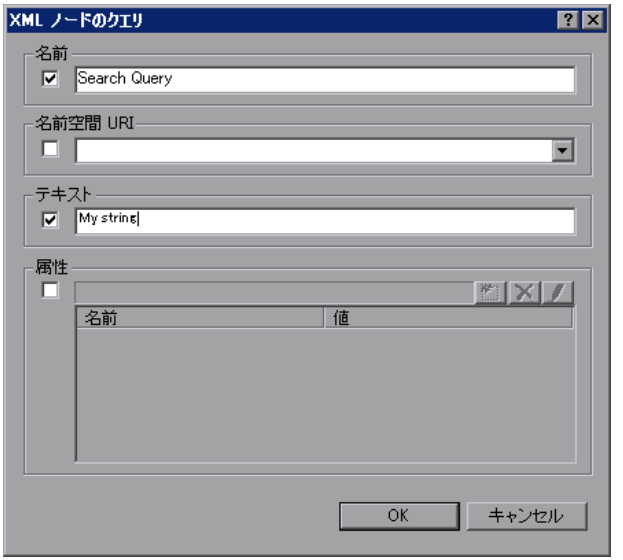
VuGen は展開可能なツリーに XML コードを表示します。XML 文書に対するクエリを実行し，特定の名前空間 URI，値，または属性を検索できます。すべてのクエリで大文字と小文字は区別されます。

クエリを行うには，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。



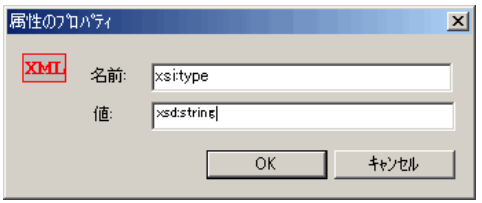
3 ［**名前**］セクションを有効にして，ノードまたは要素の名前を検索します。

4 ［**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。

5 ［**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。

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。

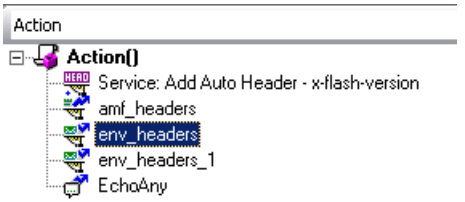


8［XML Node Query］ダイアログ・ボックスの中で［**OK**］をクリックします。VuGen は，クエリ・テキストを［XPath Query］ボックスに入れます。

9［**次を検索**］をクリックして検索を開始します。

AMF スクリプトについて

次の例では，AMF スクリプトに AMF 呼び出し，AMF ヘッダー，および AMF エンベロープが含まれています。各ノードの詳細を表示するには，ノードを選択し，右クリック・メニューから［Properties］を選択します。

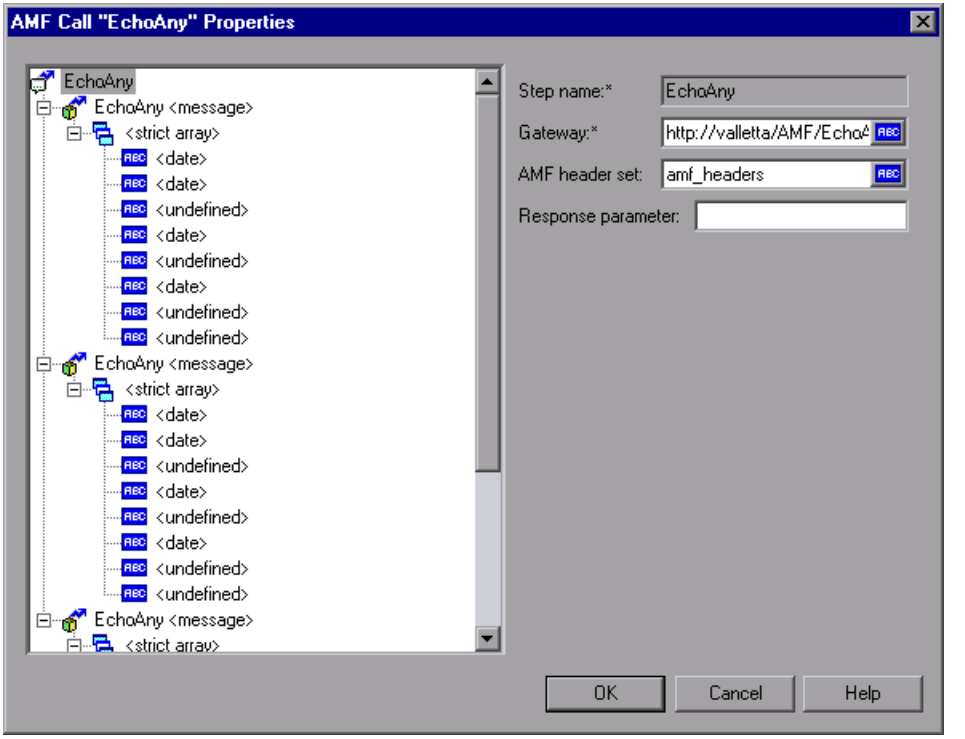


次の各項では，AMF ノードのプロパティについて説明します。

- ［AMF Call Properties］
- AMF ヘッダー・セット・プロパティ
- AMF エンベロープ・ヘッダー・セット・プロパティ

［AMF Call Properties］

［AMF Call Properties］ダイアログ・ボックスには，AMF 呼び出し，1 つ以上のメッセージ，各メッセージの引数を含む，AMF 要求の詳細が表示されます。次の例の中では，AMF 呼び出しは **EchoAny** で，**amf_header** ヘッダー・セットに関連付けられています。

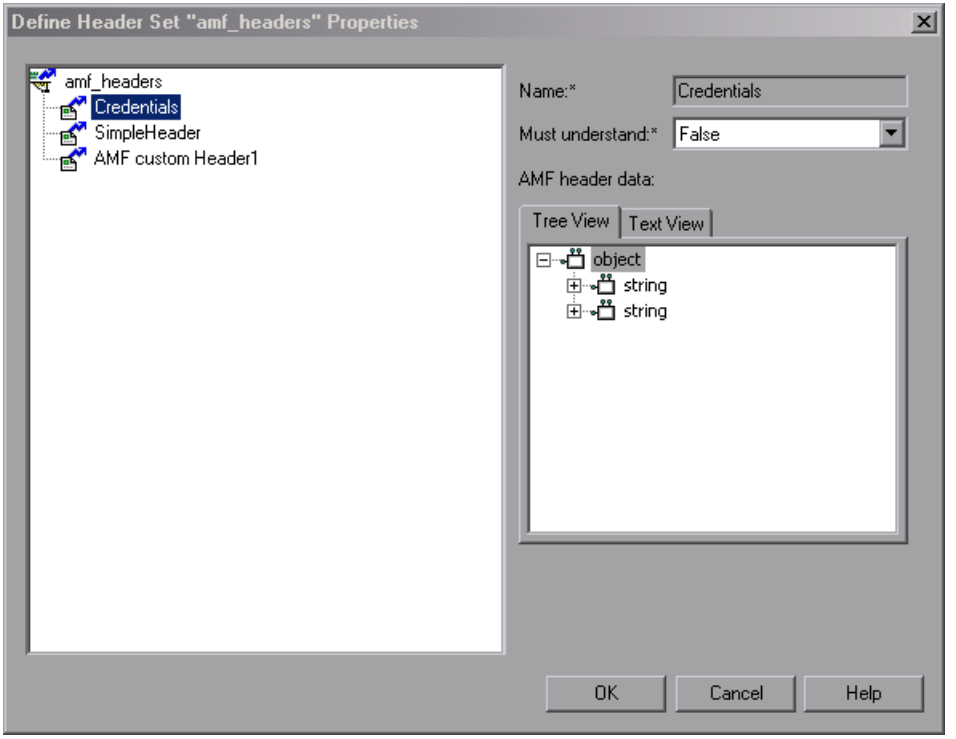


各 AMF 呼び出しには 3 つのレベルがあります。

- 「AMF Call」のレベルには，ステップ名，ゲートウェイ，AMF ヘッダー・セット，および応答パラメータが含まれます。
- 「AMF Message」のレベルには，メソッド，ターゲット・オブジェクト ID，およびエンベロープ・ヘッダー・セットが含まれます。
- 「AMF Argument」のレベルには，名前，タイプ，および値が含まれます。

AMF ヘッダー・セット・プロパティ

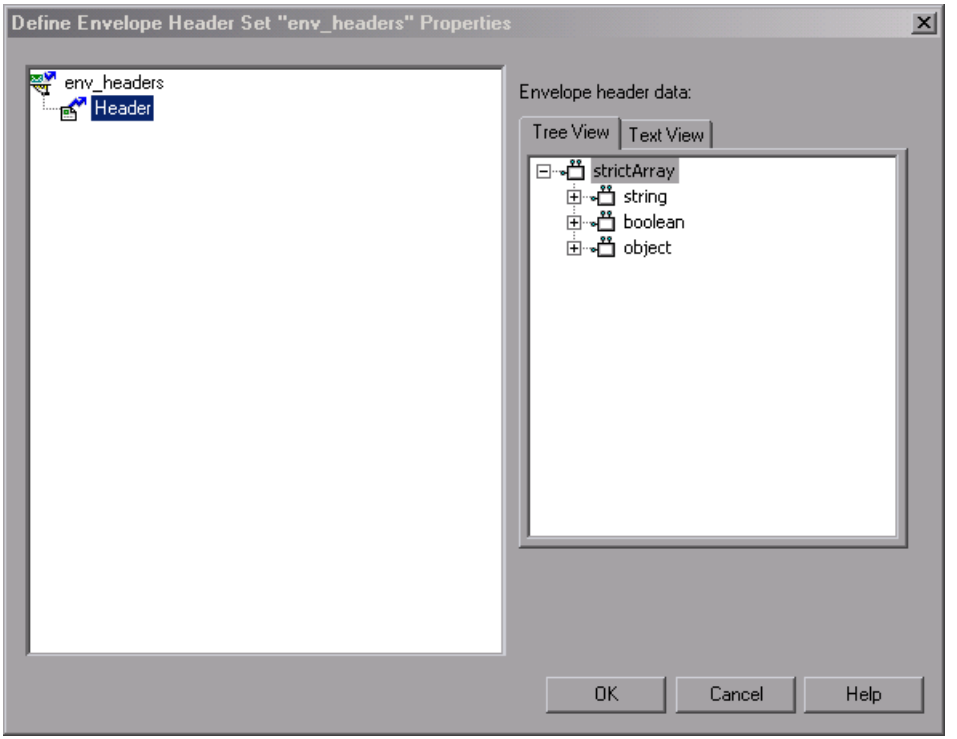
［AMF Header Set Properties］ダイアログ・ボックスには，1 つ以上の AMF ヘッダー名と，関連するヘッダー・データが XML 形式から成る AMF ヘッダー・セットの定義が表示されます。次の例では，ヘッダー・セットに 3 つのヘッダーがあります。



それぞれのヘッダーには，サーバがヘッダーを解釈できなかったときに呼び出しの処理を継続するかどうかを示す **Must understand** 引数があります。MustUnderstand が True の場合，対象ヘッダーは必須であり，解釈ができなかった場合には処理が中止されます。

AMF エンベロープ・ヘッダー・セット・プロパティ

［AMF Envelope Header Set Properties］ダイアログ・ボックスには，AMF エンベロープ・ヘッダー・セットと，各ヘッダーに対応するデータが表示されます。



第 45 章

FTP 仮想ユーザ・スクリプトの作成

VuGen では，FTP サーバに直接アクセスすることによってネットワークの動作状況をエミュレートできます。

本章では，次の項目について説明します。

- FTP 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。マルチ・プロトコルの指定の詳細については，第 4 章「VuGen を使った記録」を参照してください。

FTP プロトコルのスクリプトを作成するには，［e ビジネス］カテゴリで［File Transfer Protocol（FTP)］プロトコル・タイプを選択します。記録を開始するには，［**記録開始**］ボタンをクリックして，FTP サーバを対象に一般的なアクションを実行し記録します。スクリプトの作成と記録の詳細については，第 4 章「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 **Business Availability Center** 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が指定できます。詳細は，第 5 章「スクリプト生成オプションの設定」を参照してください。次項では C 言語を使用した仮想ユーザ・スクリプトについて説明します。

FTP 仮想ユーザ・スクリプト関数はファイル転送プロトコル（FTP）を記録します。各 FTP 関数は，**ftp** 接頭辞で始まります。

大部分の FTP 関数は，グローバル・セッションに使う関数と特定のメール・セッションの場所を示す関数が対になっています。すべてのセッションにアクションを適用するには，**ex** 接尾辞のないバージョンを使用します。特定のセッションにアクションを適用するには，**ex** 接尾辞のセッション識別子を持つバージョンを使用します。たとえば，**ftp_logon** はグローバルに FTP サーバにログオンしますが，**ftp_logon_ex** を使うと特定のセッションの FTP サーバにログオンします。

関数名	説明
ftp_delete[_ex]	FTP サーバからファイルを削除します。
ftp_dir[_ex]	FTP サーバで dir コマンドを実行します。
ftp_get[_ex]	FTP サーバからファイルを取得します。
ftp_get_last_error	FTP サーバから受信した最後のエラーを取り出します。
ftp_get_last_error_id	FTP サーバから受信した最後のエラーの ID を取り出します。
ftp_logon[_ex]	FTP サーバにログオンします。
ftp_logout[_ex]	FTP サーバからログアウトします。
ftp_mkdir[_ex]	FTP サーバ・マシン上にディレクトリを作成します。
ftp_put[_ex]	FTP サーバにファイルを置きます。

ftp_rendir[_ex] FTP サーバ・マシン上のディレクトリ名を変更します。

ftp_rmdir[_ex] FTP サーバ・マシン上のディレクトリを削除します。

ftp_get[_ex]，**ftp_put[_ex]**，および **ftp_dir[_ex]** 関数には，FTP セッションを正確にエミュレートできるようにする属性を設定できます。

PATH：FTP サーバにアップロードするファイルを指定します（**MSOURCE_PATH** が指定されていない場合にだけ使用できます）。

MPATH：FTP サーバにアップロードする複数のファイルを指定します（**ftp_dir** は対象外）。

TARGET_PATH：**ftp_put** の場合に，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（任意指定）。

LOCAL_PATH：**ftp_get** の場合に，ローカル・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（任意指定）。

MODE：取得モード ASCII または BINARY（任意指定）。標準設定は BINARY です。

PASSIVE：サーバとの通信を PASSIVE 転送モードに設定します（任意指定）。

これらの関数の詳細な構文については，「**オンライン関数リファレンス**」（[**ヘルプ**] ＞ [**関数リファレンス**]）を参照してください。

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 46 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，次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，713 ページ「識別名エントリの定義」を参照してください。

LDAP ディレクトリ・エントリは，政治的，地理的，組織的な境界を反映した階層構造で配置されています。国を表すエントリは，ツリーの一番上に現れます。その下には州や全国的な組織名を表すエントリが表示されます。さらにその下には，個人や組織，プリンタ，ドキュメントなどのエントリが表示されます。

VuGen では LDAP サーバとの通信を記録できます。VuGen によってユーザのアクションをエミュレートする関数を使ったスクリプトが生成されます。このスクリプトには，LDAP サーバへのログインとログアウト，エントリの追加と削除，およびエントリの照会が記述されます。

LDAP プロトコルのスクリプトを作成するには，［e ビジネス］カテゴリで［Lightweight Directory Access Protocol（LDAP）］プロトコル・タイプを選択します。記録を開始するには，［**仮想ユーザ**］＞［**記録開始**］を選択し，LDAP サーバを対象に一般的な操作を行います。記録の手順については，第 4 章「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 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 **Business Availability Center** 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が指定できます。詳細は，第 5 章「スクリプト生成オプションの設定」を参照してください。次項では C 言語を使用した仮想ユーザ・スクリプトについて説明します。

LDAP 仮想ユーザ・スクリプト関数は，LDAP プロトコルをエミュレートします。LDAP 関数は，**mldap** という接頭辞が付いています。

LDAP 関数はすべて，グローバル・セッション用の関数と局所的な特定のセッションを指定できる関数の対になっています。すべてのセッションにアクションを適用するには，接尾辞 **ex** のないバージョンを使用します。特定のセッションにアクションを適用するには，セッション識別子を指定できる接尾辞 **ex** を持つバージョンを使用します。たとえば，**mldap_logon** はグローバルに LDAP サーバにログオンしますが，**mldap_logon_ex** は特定セッションの 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。

これらの関数の詳細な構文については，「**オンライン関数リファレンス**」（[**ヘルプ**] ＞ [**関数リファレンス**]）を参照してください。

次の例では，ユーザが LDAP サーバ「ldap1」にログオンしています。このユーザはエントリを追加し，OU 属性を「**Sales**」から「**Marketing**」に変更しています。

```
Actions()
{
        // LDAP サーバにログオン
        mldap_logon("Login",
        "URL=ldap://johnsmith:tiger@ldap1:80",
                LAST);

        // Sally R. Jones にエントリを追加
        mldap_add("LDAP Add",
                "DN=cn=Sally R. Jones,OU=Sales, DC=com",
                "Name=givenName", "Value=Sally", ENDITEM,
                "Name=initials", "Value=R", ENDITEM,
                "Name=sn", "Value=Jones", ENDITEM,
                "Name=objectClass", "Value=contact", ENDITEM,
                LAST);

        // Sally の OU を「Marketing」に変更
        mldap_rename("LDAP Rename",
                "DN=CN=Sally R. Jones,OU=Sales,DC=com",
                "NewDN=OU=Marketing",
                LAST);

        // LDAP サーバからログアウト
        mldap_logoff();
    return 0;
}
```

識別名エントリの定義

LDAP API では，オブジェクトが**識別名**（DN）によって参照されます。DN は，カンマで区切られた**相対識別名**（RDN）の並びです。

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 47 章

Microsoft .NET 仮想ユーザ・スクリプトの記録

VuGen は，.NET Framework 環境で作成されたアプリケーションを記録します。

本章では，次の項目について説明します。

- .NET 仮想ユーザ・スクリプトの記録について
- Microsoft .NET 仮想ユーザを使った作業の概要
- 記録用フィルタの利点
- フィルタ設定についてのガイドライン
- 記録用フィルタの設定
- フィルタ・マネージャを使った作業
- Microsoft .NET 記録オプションの設定
- 記録設定の設定
- VuGen と Visual Studio でのスクリプトの表示
- .NET 環境の実行環境の設定
- データセットとグリッドの表示
- Microsoft .NET スクリプトの相関
- アプリケーションのセキュリティと権限の設定

以降の情報は，Microsoft .NET 仮想ユーザ・スクリプトを対象とします。

スクリプトを記録する前に，721 ページ「フィルタ設定についてのガイドライン」を読むことをお勧めします。このガイドラインは，アプリケーションを正確にエミュレートする最適なスクリプトの作成に役立ちます。

.NET 仮想ユーザ・スクリプトの記録について

Microsoft .NET Framework は，ASP.NET，Windows Forms，Web サービス，分散アプリケーション，またはこれらのモデルを組み合わせたアプリケーションなど，さまざまな種類のアプリケーションを構築する際の強固な基盤を提供します。

VuGen ではアプリケーション・レベルのプロトコルとして .NET をサポートしています。VuGen を使用して，.NET Framework で作成された Microsoft .NET クライアント・アプリケーションのユーザをエミュレートする仮想ユーザ・スクリプトを作成できます。VuGen はクライアントのすべてのアクションをメソッドおよびクラスとして記録し，C# または VB .NET 言語のスクリプトを作成します。

標準設定では，VuGen 環境は ADO .NET アプリケーションおよび .NET Remoting アプリケーションで構成されます。これ以外のクライアント／サーバ動作に基づいて作成されたアプリケーションを記録する場合に VuGen を設定する方法については，カスタマー・サポートにお問い合わせください。

制限事項

VuGen で Microsoft .NET アプリケーションを記録するには，次の制限事項があります。

- VuGen ではイベントやデリゲートをサポートしていますが，サポートされているのは .NET Remoting 非同期呼び出しだけです。
- Microsoft .NET スクリプトは，VuGen でシングル・プロトコルの記録のみサポートします。
- パブリック・フィールドに直接アクセスすることはできません。AUT はメソッドまたはプロパティを介してフィールドにアクセスする必要があります。
- VuGen はアプリケーションの静的フィールドは記録しません。クラス内のメソッドだけが記録されます。
- マルチ・スレッドをサポートするかどうかは，クライアント・アプリケーションに依存します。記録されたアプリケーションがマルチ・スレッドをサポートする場合は，仮想ユーザ・スクリプトもマルチ・スレッドをサポートします。
- 場合によっては，スクリプトを修正しないと反復を複数回実行できないことがあります。前の反復ですでに初期化されたオブジェクトの再初期化はできません。したがって，反復を複数回実行するには，各反復の終了時に開いている接続またはリモート・チャネルを必ずすべて閉じます。

- ユーザ定義のプロキシを使用した Remoting 呼び出しの記録はサポートされていません。
- 標準設定の ADO.NET フィルタを使用する場合，ADO.NET オブジェクトの **ExtendedProperties** プロパティの記録はサポートされていません。
- Framework 2.0 と互換性のない .NET Framework 1.1 で作成されたアプリケーションは記録されません。Framework 1.1 アプリケーションに互換性があるかどうかを確認するには，次の XML タグをアプリケーションの .config ファイルに追加します。

```
<configuration>
    <startup>
        <supportedRuntime version="v2.0.50727"/>
    </startup>
</configuration>
```

VuGen を介さずにアプリケーションを直接起動し，アプリケーションの動作を確認します。アプリケーションが正しく動作すれば，VuGen を使用してこのアプリケーションを記録できます。VuGen を使用してこの AUT を記録する前に，上記のタグは削除してください。この方法の詳細については，MSDN Knowledge Base を参照してください。

Microsoft .NET 仮想ユーザを使った作業の概要

本項では，Microsoft .NET 仮想ユーザ・スクリプトを作成するプロセスについて説明します。スクリプトを実行するマシンとすべてのロード・ジェネレータに AUT（テスト対象アプリケーション）がインストールされている必要があります。

基本となる .NET 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 を使ってアプリケーションを記録します。**

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 **Microsoft .NET** を指定します。記録対象アプリケーションを選び，記録オプションを設定します。詳細については，736 ページ「Microsoft .NET 記録オプションの設定」を参照してください。

一般的な記録方法の詳細については，第 4 章「VuGen を使った記録」を参照してください。

2 **スクリプトをデバッグし，フィルタを設定します。**

記録されたスクリプトを調べ，必要なメソッドが記録されているかどうかを確認します。必要があれば，フィルタを変更してスクリプトを記録しなおします。

フィルタ設定の詳細については，721 ページ「フィルタ設定についてのガイドライン」を参照してください。

3 **スクリプトを相関させます。**

スクリプト内の値を相関させ，スクリプトの以降の場所で使用できるように値をパラメータとして保存します。

詳細については，746 ページ「Microsoft .NET スクリプトの相関」を参照してください。

4 **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。再生が正常に行われるようにするために，スクリプトをロード・ジェネレータ・マシンまたはコントローラ経由でリモート実行する前に，VuGen でコンパイルして実行します。これは，スクリプトの設定またはスクリプトそのもののいずれを変更した場合でも行います。また，スクリプトを別の名前で保存した場合は，スクリプトをロード・ジェネレータで実行する前に VuGen で再生します。

詳細については，第 15 章「スタンドアロン・モードでの仮想ユーザ・スクリプトの実行」を参照してください。

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，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，742 ページ「.NET 環境の実行環境の設定」を参照してください。

スクリプトを作成した後，スクリプトを自分の環境（LoadRunner シナリオ，Performance Center 負荷テスト，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**』，**チューニング・コンソール** のマニュアル，**Performance Center** のマニュアル，または **Business Availability Center** のマニュアルを参照してください。

記録用フィルタの利点

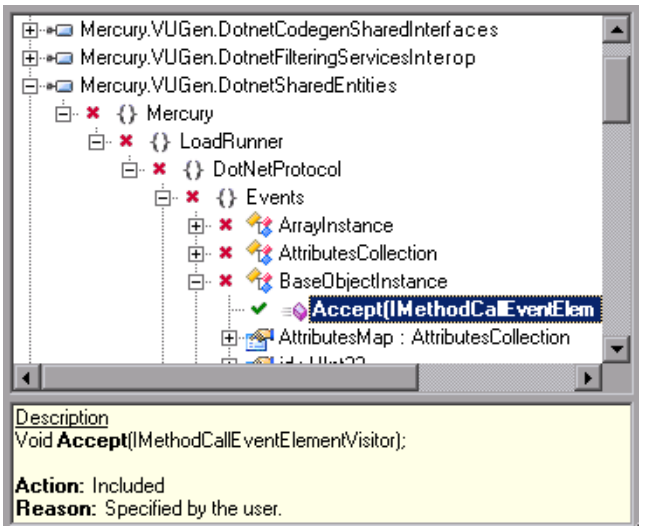
記録用のフィルタには，記録およびスクリプトの生成時に包含または除外するアセンブリ，インタフェース，名前空間，クラス，またはメソッドを指定します。

標準設定では，VuGen は ADO.NET および .NET Remoting 用に 2 つの組み込みシステム・フィルタを備えています。これらのフィルタは，標準の ADO.NET および Remoting に対応するインタフェースを包含するように作成されています。VuGen ではカスタム・フィルタも作成できます。

カスタム・フィルタには次のいくつかの利点があります。

- **Remoting**：.NET Remoting を対象とした作業を行う場合，リモート・メソッドに渡される引数を記録できるクラスを包含することが重要です。
- **欠落オブジェクト**：記録したスクリプトにアプリケーション内の特定のオブジェクトが記録されていない場合，フィルタを使用して欠落しているインタフェース，クラス，またはメソッドを包含することができます。
- **デバッグ**：エラーを返されてもその原因がわからない場合，問題の原因となった操作を絞り込むために，フィルタを使用してメソッド，クラス，またはインタフェースを除外できます。
- **保守性**：高水準のスクリプトを記録することで，スクリプトの保守と相関がしやすくなります。

フィルタ・マネージャを使用して既存のカスタム・フィルタを操作できます。フィルタ・マネージャには，アセンブリ，名前空間，クラス，メソッド，およびプロパティが色分けされたツリー階層として表示されます。



下の表示枠には，アセンブリ，名前空間，クラス，メソッド，プロパティ，またはイベントの説明が表示されます。また，それぞれの要素が包含されるのか除外されるのかが分かるほか，包含または除外の理由も示されます。次の項では，フィルタをカスタマイズするべき状況とカスタマイズする方法について説明します。

- フィルタ設定についてのガイドライン
- 記録用フィルタの設定
- フィルタ・マネージャを使った作業

フィルタ設定についてのガイドライン

.NET アプリケーションをテストする目的は，クライアントからの要求に対するサーバの反応を確認することです。負荷テストの場合には，多数のユーザによる負荷にサーバがどのように応答するかを確認します。

.NET アプリケーションを記録すると，スクリプトにはローカル・ユーティリティや GUI インタフェースの呼び出しなど，サーバに影響を与えないメソッドの呼び出しが含まれる場合があります。通常，こうした呼び出しはテストの目的には関係ないので，フィルタによって除外するのが適切です。

組み込みのフィルタである ADO.NET および .NET Remoting は，テストの目的にかなうサーバ関連のトラフィックのみを記録するように作成されています。ただし，状況によっては，.NET アプリケーションの呼び出しをキャプチャしたり不要な呼び出しを除外したりするために，ユーザ定義のカスタム・フィルタが必要になることがあります。フィルタ・マネージャを使用すれば，カスタム・フィルタを作成して，関係のない呼び出しの除外やサーバに関係する呼び出しのキャプチャができます。

記録に何を含めるかを判断できるように，テストを作成する前にアプリケーションについて理解を深め，アプリケーションの主要なクラスやメソッドを確認することをお勧めします。

アプリケーションのクラスに精通していない場合，フィルタにメソッドを包含するために，**Visual Studio** または**スタック・トレース**を使用して，アプリケーションに呼び出されるメソッドを調べることができます。VuGen では，アプリケーションによって呼び出されたすべてのメソッドをログに記録するスタック・トレース付きで記録することが可能です。

必要なメソッドとクラスを確認したら，フィルタ・マネージャを使用してそれらを包含するようにします。スクリプトを準備するときに，最適なフィルタにするためにフィルタを数回カスタマイズしなければならない場合があります。フィルタが最適であれば，スクリプトに無関係の多数の呼び出しを取り込まずに必要なメソッドが記録されます。

ヒント：VuGen 内でスクリプトがまずはコンパイル（Shift+F5）できるようになるまで，つまり，正しい構文のテストとなるように，フィルタを可能な限り調整します。次に，VuGen でのスクリプトの実行が可能になるまでフィルタにカスタマイズを加えます。

フロー制御やメッセージ・ステートメントなど，スクリプトにコードを手作業で追加する場合は，必ず VuGen 内での実行が可能なスクリプトが得られてからにしてください。これは，スクリプトを再記録または再生成すると，手作業による変更がすべて失われるためです。

包含または除外する要素の指定

カスタム・フィルタを作成するとき，基本のフィルタとして適切な標準設定のフィルタ（ADO.NET または Remoting）を選択するところから始めることをお勧めします。その後，次のどちらかの方法を使用してフィルタをカスタマイズできます。

トップ・ダウン方式。この方式では，関係する名前空間を包含し，クライアント／サーバの動作に関与しない個別のクラスを除外します。アプリケーションに精通しており，MyDataAccessLayer.dll のような GUI 要素が関与しないクライアント／サーバの動作をすべて実装する明確なアセンブリを特定できる場合には，この方法をお勧めします。

ボトム・アップ方式。標準設定のフィルタを使用し，個別のメソッドまたはクラスを包含するようにしてフィルタを調整していく方式です。明確なレイヤを特定できない場合，またはアプリケーションに精通していない場合は，この方式を使用します。すべての AUT アセンブリを追加して，その後余分なコンポーネントを 1 つずつ削除するようなことはしないでください。

次の項では，要素を包含または除外する際のガイドラインを示します。

- クラスを包含した結果，スクリプトに多くの無関係のメソッド呼び出しが含まれた場合は，フィルタに変更を加えて無関係のメソッドが除外されるか試してみます。
- スクリプトにクライアント／サーバの動作にかかわらない呼び出しが含まれた場合，フィルタからそのメソッドを除外します。
- 記録中，VuGen はこれまで遭遇したことのない構造の引数など，未知の入力引数を検出する場合があります。この引数がシリアライズをサポートしている場合，VuGen は引数を特別な形式で引数をファイルに保存することでその引数を**シリアライズ**します。再生中，VuGen は引数を**シリアル化解除**することでそれを復元します。

VuGen は，フィルタによって包含されなかった引数として渡されたオブジェクトをシリアライズします。オブジェクトの構造と動作を追跡するために，オブジェクトをシリアライズされた形式で使用するのではなく，フィルタに含めることをお勧めします。スクリプト内のシリアライズされたオブジェクトを特定するには，**LrReplayUtils.GetSerializedObject** メソッドへの呼び出しを検索します。

- 標準設定では，メソッドに対してルールが指定されていなければ，メソッドは除外されます。ただし Remoting 環境が有効な場合，標準設定では，明示的に包含していなくてもすべてのリモート呼び出しが追加されます。標準設定の動作を変更するには，ユーザ定義ルールを追加して，リモート・サーバを対象とする特定の呼び出しを除外できます。
- Remoting 呼び出しで渡された，フィルタによる包含対象になっていない種類の引数は，シリアル化メカニズムによって処理されます。引数がシリアライズされないようにするには，引数の構造と動作を記録するために，その種類の引数を明示的に包含するようにします。
- GUI 要素が関与する動作はすべて除外します。
- スクリプトのコンパイルに必要なユーティリティ用のアセンブリを包含するようにします。

要素を包含または除外する方法の詳細については，731 ページ「要素の包含および除外」を参照してください。

効果的なフィルタの定義

スクリプトを準備するときに，最適なフィルタにするためにフィルタを数回カスタマイズしなければならない場合があります。フィルタが最適であれば，スクリプトに無関係の多数の呼び出しを取り込まずに必要なメソッドが記録されます。

効果的なフィルタを定義するには，次の手順を実行します。

1 ADO.NET および .NET Remoting に基づいて新しいフィルタを作成します。不要なフィルタによって記録速度が低下する可能性があります。そのため AUT（テスト対象アプリケーション）が ADO.NET または Remoting を使用しないことがわかっている場合は，そのオプションをクリアします。

2 記録とコード生成の両方に対して，［**スタック トレース**］オプションを「True」に設定します。［記録オプション］を開き（Ctrl+F7），［**記録**］ノードを選択します。［**デバッグ オプション：スタック トレース**］と［**コードの生成：スタック トレースを表示する**］を有効にします。

3 アプリケーションを記録します。［**記録開始**］（Ctrl+R）をクリックすると記録が始まり，［停止］（Ctrl+F5）をクリックすると終了します。

4 スクリプトのステップを表示します。ステップを見てビジネス・ロジックを特定し相関を適用できる場合には，カスタム・フィルタを作成する必要はありません。しかし，スクリプトが非常に長かったり保守や相関が困難だったりする場合には，スクリプトのフィルタをカスタマイズする必要があります。

5 1 つ以上のクライアント・サーバ呼び出しをキャプチャまたはラップする呼び出しの中で，高水準のメソッドの識別を試みます。そのためには，Visual Studio で AUT ソース・ファイルを開くか（使用可能な場合），スクリプトのスタック・トレースを表示します。

6 関係するメソッドを包含するようにフィルタを設定します。あらかじめそれらのアセンブリを包含する必要がある場合があります。フィルタに要素を含めたり，フィルタから要素を除外したりする際のヒントについては，722 ページ「包含または除外する要素の指定」を参照してください。

7 アプリケーションを記録しなおします。フィルタを変更したら必ずアプリケーションを記録しなおしてください。

8 容易に保守と相関が行える簡単なスクリプトになるまで，手順 4 から 7 を繰り返します。

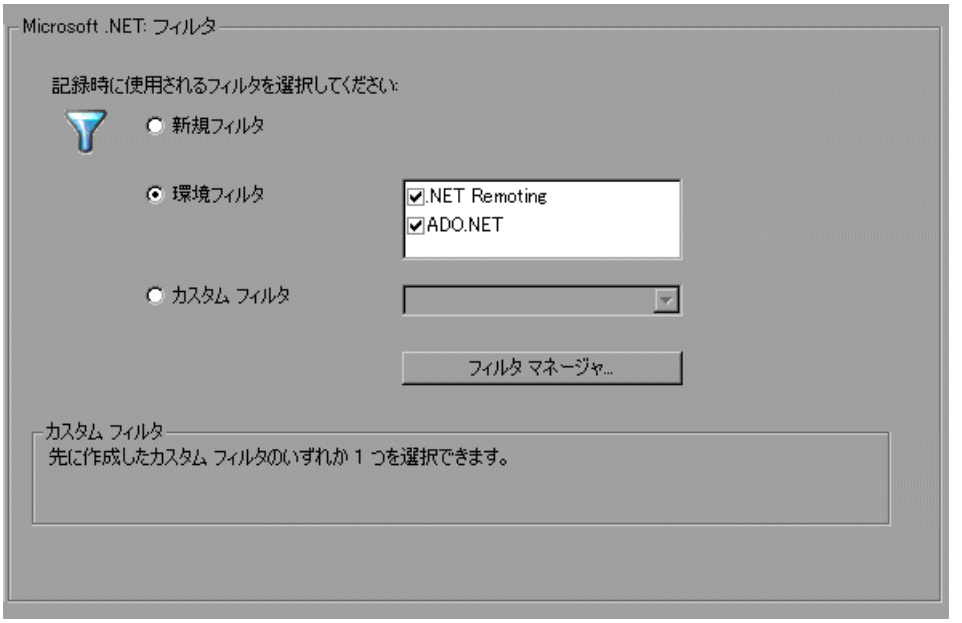
9 最適なスクリプトを作成したら，［**スタック トレース**］オプションを無効にしてスクリプトを再生成します。［記録オプション］を開き（Ctrl+F7），［**記録**］ノードを選択します。［**デバッグ オプション：スタック トレース**］と［**コードの生成：スタック トレースを表示する**］を無効にします。これにより，以降の記録のパフォーマンスが向上します。

10 スクリプトを相関させます。テストを正しく実行するために，相関を挿入して値をキャプチャし，スクリプトの以降の場所で使用する必要がある場合があります。組み込みの相関メカニズムの詳細については，746 ページ「Microsoft .NET スクリプトの相関」を参照してください。

記録用フィルタの設定

組み込みフィルタの ADO.NET および .NET Remoting は，ADO.NET および Remoting の標準インタフェースを含むように設計されています。フィルタの利点の詳細については，721 ページ「フィルタ設定についてのガイドライン」を参照してください。

記録にフィルタを適用する場合，最初のステップは適切なフィルタを選択することです。［**フィルタ**］記録オプションを使用して，1 つ以上の環境フィルタを使用するか新しいフィルタを作成できます。



- ［**新規フィルタ**］: 新しいフィルタを作成することを示します。
- ［**環境フィルタ**］: 使用可能な環境フィルタ（ADO.NET および .NET Remoting）の一覧が表示されます。
- ［**カスタム フィルタ**］: 現在のマシンで以前に作成したフィルタを表示します。

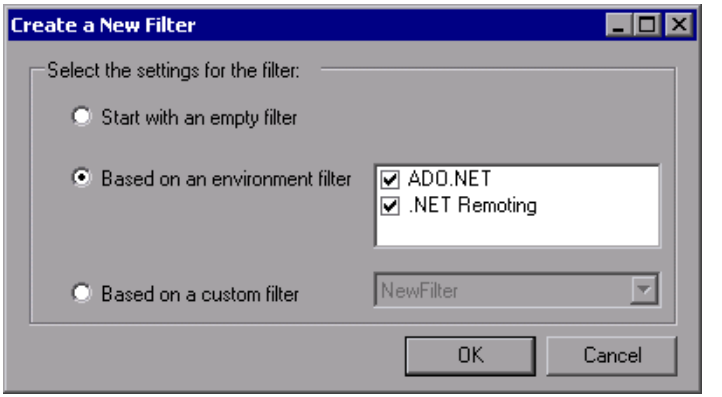
フィルタを作成したら，フィルタ・マネージャを使用してフィルタのプロパティを変更できます。詳細については，731 ページ「要素の包含および除外」を参照してください。

フィルタを指定するには，次の手順を実行します。

1 ［フィルタ］記録オプションを開きます。［**ツール**］＞［**記録オプション**］を選択し，［**Microsoft .NET：フィルタ**］ノードを選択します。

2 フィルタ・オプションとして，［**新規フィルタ**］，［**環境フィルタ**］または［**カスタム フィルタ**］を選択します。

3 新規フィルタの場合は［**作成**］をクリックします。［Create a New Filter］ダイアログ・ボックスが開きます。



4 いずれかのフィルタ・オプションを選択します。環境フィルタに基づく新しいフィルタを作成するには，環境フィルタの横にあるチェック・ボックスを 1 つ以上選択します。

5 ［**OK**］をクリックします。フィルタ・マネージャが開きます。

既存のフィルタの場合，フィルタ・マネージャを開くには，記録オプションのメイン・ダイアログ・ボックスで［**フィルタ マネージャ**］ボタンをクリックします。

フィルタに必要な変更を加えて保存します。詳細については，次を参照してください。

フィルタ・マネージャを使った作業

フィルタ・マネージャでは，環境フィルタ以外のフィルタの表示および変更ができます。環境フィルタは読み取り専用のため，表示はできますが変更はできません。

フィルタの管理および操作を行うには，フィルタ・マネージャ・ツールバーを使用します。

フィルタの管理

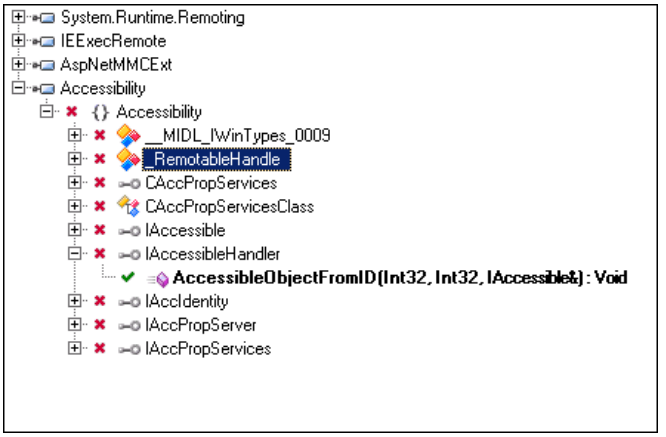
- ［**New**］：［Create a New Filter］ダイアログ・ボックスを開きます。ここで空のフィルタを作成するか，既存のフィルタに基づく新しいフィルタを作成します。
- ［**Save**］：フィルタに加えた変更を保存します。
- ［**Delete**］：選択したカスタム・フィルタを削除します。フィルタ・マネージャにより確認を求められます。

フィルタの操作

- ［**Add Reference**］：［Add Reference］ダイアログ・ボックスを開き，.NET Framework コンポーネントまたは Public Assemblies フォルダ内のアセンブリのリストを表示します。また，コンピュータを参照して，リストに表示されていないコンポーネントを検索できます。詳細については，729 ページ「参照の追加」を参照してください。
- ［**Remove Reference**］：フィルタ・マネージャの中で選択されているアセンブリと，アセンブリに関連付けられているすべての要素を削除します。フィルタ・マネージャにより確認を求められます。
- ［**Include**］，［**Exclude**］，および［**Reset**］：アセンブリ，名前空間，クラス，またはメソッドを，包含または除外します。包含または除外のルールを標準設定の状態にリセットすることもできます。詳細については，731 ページ「要素の包含および除外」を参照してください。
- ［**Back**］および［**Forward**］：ユーザが訪れた前のツリー・ノードまたは次のツリー・ノードに移動します。
- ［**View Impact Log**］：選択したフィルタ用の Impact ログを開きます。Impact ログには，直前のアクションの影響を受けたツリー・ノードが表示されます。詳細については，733 ページ「Impact ログの表示」を参照してください。

さらに，標準の Windows キーの組み合わせと右クリック・メニューを使用して，フィルタのコピー，貼り付け，および名前変更ができます。

フィルタ・マネージャ・ツリーでは，記号を使用して要素とそのステータスを示します。



- 要素アイコンは，要素の種類（アセンブリ，名前空間，クラス，メソッド，構造体，プロパティ，イベント，またはインタフェース）を表します。
- 要素アイコンの横のチェック・マークまたは X は，要素が包含または除外されているかどうかを示します。
- 太字の要素は，明示的に包含または除外されていることを示します。これは，ユーザの手作業によって，または環境フィルタの定義済みのルールによって，包含または除外された結果です。太字のノードをリセットすると，元の太字ではない状態に戻ります。

次の表に，各種の要素を表すアイコンを示します。

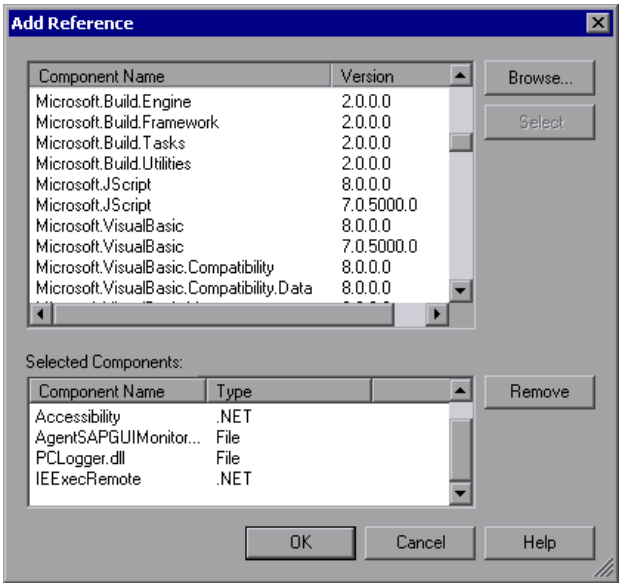
	アセンブリ			インタフェース
	ロードできなかったアセンブリ			メソッド
	一部がロードされたアセンブリ			静的メソッド
	クラス			名前空間
	コンストラクタ			プロパティ
	静的コンストラクタ			静的プロパティ
	イベント			構造体
	静的イベント			

参照の追加

新しいフィルタを作成する場合，フィルタの動作を指定するために，フィルタにアセンブリ参照を追加する必要があります。また，参照を既存のカスタム・フィルタに追加することもできます。

［Add Reference］ダイアログ・ボックスを開くと，GAC（グローバル・アセンブリ・キャッシュ）のすべての public アセンブリが一覧表示されます。

［**Browse**］ボタンを使用すると，一覧表示されていない参照を追加できます。



下の表示枠の中で，［**Type**］カラムを見ることで参照を区別できます。GAC 内に存在する参照は種類が「**.NET**」で，GAC 内に存在しない参照は種類が「**File**」になります。

参照のリストには，同じアセンブリの異なるバージョンが含まれている場合があります。その場合は，テストに最適なバージョンを選択します。

フィルタ・マネージャを使用して参照を追加するには，次の手順を実行します。

1 ツールバーの［**Add Reference**］ボタンをクリックします。［Add Reference］ダイアログ・ボックスが開きます。

2 一覧表示されている項目のいずれかを追加するには，［**Select**］をクリックします。複数のコンポーネントを選択するには Ctrl キーを押したまま，コンポーネントをクリックしていきます。下の表示枠に選択した参照が表示されます。

3 リストに表示されていないアセンブリを追加するには，［**Browse**］をクリックしてファイル・システムまたはネットワーク上で参照を検索します。

4 フィルタに追加するすべての参照に対して前記の手順を繰り返します。

5 下の表示枠に表示された参照のリストを確認します。リストから項目を削除するには，下の表示枠で削除する項目を選択して［**Delete**］をクリックします。

6 参照のリストの作成が完了したら，［**OK**］をクリックしてダイアログ・ボックスを閉じ，参照をフィルタに追加します。フィルタ・マネージャのツリーで，要素のリストの最後に参照が追加されます。選択した参照のいずれかが有効なアセンブリでない場合，エラー・メッセージが発行されます。

7 フィルタ・マネージャのツリーから参照を削除するには，親アセンブリ・ノードを選択して右クリック・メニューから［**Remove Reference**］を選択するか，ツールバーの［Remove Reference］ボタンをクリックします。

参照を追加したら，フィルタ・マネージャで表示して，正しいノードがすべて包含または除外されていることを確認してください。必要に応じて，特定の名前空間，クラス，またはメソッドを包含または除外することができます。詳細については，次の「要素の包含および除外」を参照してください。

要素の包含および除外

フィルタに参照を追加したら，フィルタ・マネージャのツリーでフィルタのすべてのノードを表示できます。特定の名前空間，クラス，またはメソッドを除外できます。あるいは，標準設定または別のルールによって除外されているものを包含するようにできます。フィルタ・マネージャの下部の表示枠の説明には，要素が包含または除外されている理由が示されます。

フィルタ・マネージャのツールバーには，要素を包含または除外するための次のボタンが表示されます。

［**Include**］：選択した要素を包含し，チェック・マークを付けます。親ノードを手作業で追加すると，ほかにルールが設定されていなければ，フィルタ・マネージャによりそのノードの下位の子要素が包含されます。たとえばクラスを追加すると，ユーザが明示的に除外したメソッドを除き，クラスのすべてのメソッドが包含されます。

［**Exclude**］：選択した要素を除外し，X を付けます。ほかのルールによって包含されていない限り，子要素も除外されます。標準設定では，**クラス**を除外するとフィルタ・マネージャはそのクラスに **Exclude** 属性を適用しますが，削除されたクラスのメソッド内の動作を記録エンジンが記録することは可能です。ただし，**メソッド**を除外するとフィルタ・マネージャは **Totally Exclude** を適用し，除外されたクラスのメソッド内のすべての動作を記録エンジンが記録できないようにします。上級ユーザは，フィルタ・ファイル内でこれらの設定を変更できます。詳細については，734 ページ「フィルタ・ファイルに関する詳細情報」を参照してください。

［**Reset**］：手作業で設定した包含または除外のルールを削除します。この場合，対象要素はほかの親要素の影響を受けます。

包含または除外のルールのプロパティは，次のとおりです。

- ルールは階層構造です。包含または除外するルールをクラスに追加すると，派生クラスはほかに指定がある場合を除き，同じルールに従います。
- クラスを対象とするルールは，クラスの public メソッド，派生クラス，および内部クラスにのみ影響します。
- 名前空間を対象とするルールは，すべてのクラスとその public メソッドに影響します。
- アセンブリを追加または削除しても，必ずしもアセンブリに含まれるクラスが影響を受けるとは限りません。アセンブリを削除しても，フィルタの階層構造に起因してアセンブリのメソッドが記録されることもあります。
- フィルタ作成時に考慮すべきこととして，**.cctor()** や **Dispose(bool)** などのいくつかのメソッドは，標準の階層ルールに従わないことが挙げられます。

注：親ノードをリセットしても，子ノードに適用した手作業の包含または除外のルールに優先しません。たとえば，メソッドを手作業で**除外**し，その後そのメソッドのクラス（標準設定ですべてのサブ・ノードを**包含**するクラス）をリセットしても，メソッドは除外されたままです。

プロパティおよびイベントは表示専用であり，フィルタ・マネージャを使用して包含または除外することはできません。また，システムに関連するいくつかの要素は保護されており，変更できません。

フィルタに要素を含めたり，フィルタから要素を除外したりする際のヒントについては，722 ページ「包含または除外する要素の指定」を参照してください。

アセンブリを追加または削除するには，729 ページ「参照の追加」の説明に従って［**Add Reference**］ボタンおよび［**Remove Reference**］ボタンを使用します。次の項では，名前空間，クラス，およびメソッドを包含する方法について説明します。

要素を包含または除外するには，次の手順を実行します。

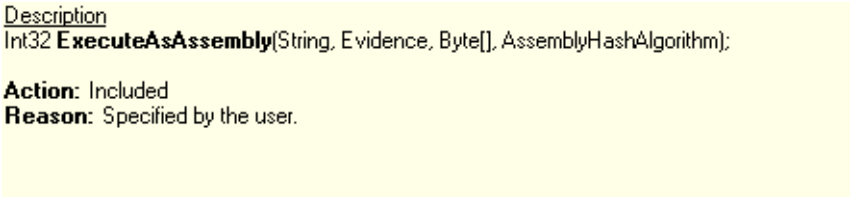
1 ツリー階層を展開し，名前空間，クラス，またはメソッドを選択します。

2 特定の要素を包含するには，その要素を選択して［**Include**］ボタンをクリックするか，右クリック・メニューで［Include］コマンドを使用します。

3 特定の要素を除外するには，［**Exclude**］ボタンをクリックするか，右クリック・メニューで［Exclude］コマンドを使用します。

4 要素を標準設定に戻すには，［**Reset**］ボタンをクリックするか，右クリック・メニューから［**Reset**］を選択します。

変更が適用されたことを確認するには，コンポーネントを選択して下の表示枠を確認します。

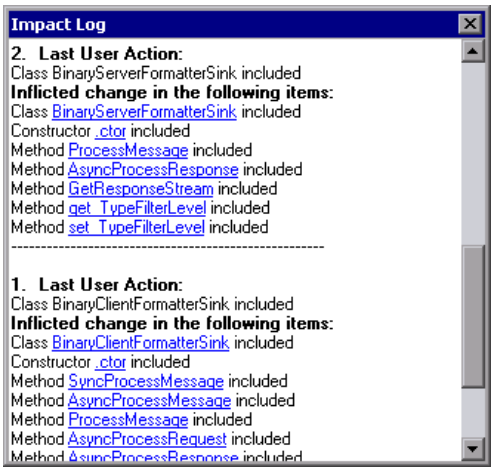


Impact ログの表示

Impact ログには，最後に加えた変更と，変更がフィルタに与えた影響が示されます。ユーザ・アクションが，最後に加えた変更を先頭に降順で一覧表示されます。

ログには，手作業で包含または除外されたことで影響を受けた要素ごとに，どのような影響を受けたかが示されます。フィルタ・マネージャ内の要素へのリンクも含まれます。

Impact ログを表示するには，フィルタ・マネージャのツールバーの［Impact Log］ボタンをクリックするか，［フィルタ マネージャ］ウィンドウで［**View**］＞［**View Impact Log**］を選択します。



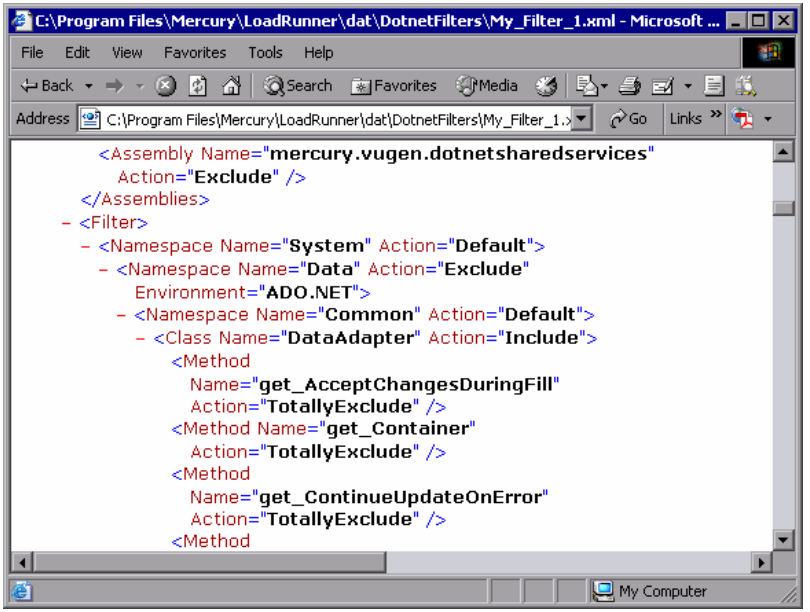
フィルタ・ファイルに関する詳細情報

フィルタ・マネージャのツリー階層には，public クラスと public メソッドのみが表示されます。public 以外のクラスやデリゲートは表示されません。

public 以外のクラスまたはメソッドを追加するには，フィルタの定義ファイルに手作業で入力します。

フィルタ定義ファイル「**＜フィルタ名＞ .xml**」は，インストール先の dat/DotnetFilters フォルダにあります。各要素に対して指定可能なアクション・プロパティは，**Include**，**Exclude**，または **Totally Exclude** です。詳細については，731 ページ「要素の包含および除外」を参照してください。

標準設定では，**クラス**を除外するとフィルタ・マネージャは **Exclude** を適用し，クラスを除外しますが，除外されたクラスによって生成された動作は包含されます。しかし，**メソッド**を除外すると **Totally Exclude** が適用され，すべての参照メソッドが除外されます。



たとえば，関数 A が関数 B を呼び出すとします。関数 A に **Excluded** が適用されている場合，サービスが関数 A を呼び出すと，スクリプトには関数 B の呼び出しが含まれます。しかし，関数 A に **Totally Excluded** が適用されている場合，スクリプトには関数 B の呼び出しは含まれません。関数 B は，関数 A からではなく直接呼び出された場合にのみ記録されます。

記録中 VuGen は，設定されたフィルタのバックアップ・コピーを，スクリプトの **data** フォルダに **RecordingFilterFile.xml** という名前で保存します。このファイルは，最後に記録を行ってからフィルタに変更を加えており，環境を復元する必要がある場合に役立ちます。

Microsoft .NET 記録オプションの設定

スクリプトの記録オプションと Microsoft .NET 固有の記録オプションの両方を設定できます。本項では，Microsoft .NET 固有の記録オプションについて説明します。スクリプト記録オプションの詳細については，第 5 章「スクリプト生成オプションの設定」を参照してください。

Microsoft .NET 固有の記録オプションは，記録設定と .NET フィルタを対象とします。

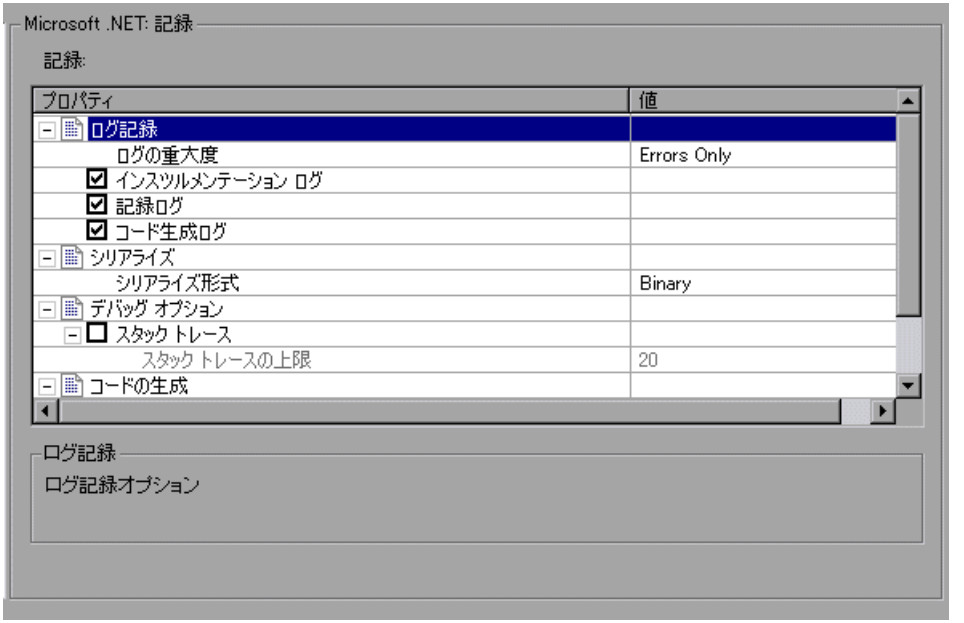
記録設定では，ログ，シリアライズ，デバッグ，およびログのトレース・レベルを制御できます。詳細については，737 ページ「記録設定の設定」を参照してください。

フィルタ・オプションでは，記録用のフィルタを標準の .NET Remoting または ADO.NET フィルタから選択できます。また，カスタム・フィルタを作成し，必要に応じて設定できます。詳細については，740 ページ「VuGen と Visual Studio でのスクリプトの表示」を参照してください。

記録オプションの中には，スクリプトの再生成に関係するものもあります。スクリプトを記録したら，コード生成オプションを変更して，別の設定または別の言語を使用してスクリプトを再生成できます。オプションを変更して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**］を選択します。［スクリプトの再生成］ダイアログ・ボックスで，［**オプション**］をクリックします。詳細については，80 ページ「仮想ユーザ・スクリプトの再生成」を参照してください。

記録設定の設定

［.NET 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**］を選択し，［**記録**］ノードを選択します。



次の領域に関して，スクリプトを設定できます。

- ログ
- シリアライズの設定
- デバッグ・オプション
- コード生成

ログ

- ［ログ］オプションを使用して，記録ログ・ファイルに記録される情報の詳細度を設定できます。［**ログの重大度**］：ログのレベルを［**Errors Only**］（標準設定），［**Debug**］，または［**Full Trace**］に設定します。［Full Trace］が最も詳細なログです。重大度の設定は以降の説明に従って有効にするすべてのログに適用されます。ログを詳細に記録すると記録時間が大幅に増えることがあるので，Mercury のサポートによる特別な指示がある場合を除き，必ず［**Errors Only**］ログを使用してください。
- ［**インスツルメンテーション ログ**］：インスツルメンテーション・プロセスに関するメッセージをログ記録します（標準設定では有効）。
- ［**記録ログ**］：記録中に発行されたメッセージをログ記録します（標準設定では有効）。
- ［**コード生成ログ**］：コード生成段階で発行されたメッセージをログ記録します（標準設定では有効）。

シリアライズの設定

［シリアライズ］設定ではシリアライズ形式を設定できます。

VuGen は記録中に未知のオブジェクトに遭遇した場合，オブジェクトがシリアライズをサポートしていればシリアライズを使用します。たとえば、フィルタに包含されていなかったために、その生成が記録されなかった入力引数などが、未知のオブジェクトとして考えられます。シリアライズによって，未知の引数をメソッドに渡すことによるコンパイルエラーがなくなります。オブジェクトがシリアライズされる場合，カスタム・フィルタを設定してこのオブジェクトを記録することをお勧めします。

- ［**シリアライズ形式**］：シリアライズをサポートするクラスの記録時に VuGen によって作成されるシリアライズ・ファイルの形式です。［**Binary**］，［**XML**］，または［**Both**］を設定できます。バイナリ形式の利点は，圧縮率が高いのでより高速であるということです。バイナリ形式の不利な点は，XML と同じようにはデータを操作できない点です。

デバッグ・オプション

デバッグ・オプションではスタックを追跡してサイズを指定できます。

- [**スタック トレース**]：スクリプト内の各呼び出しのスタックの内容を追跡します。これにより，アプリケーションで使用されたクラスおよびメソッドを確認できます。これは，フィルタに追加する参照，名前空間，クラス，またはメソッドを調べるのに便利です。追跡を有効にすると，記録時のアプリケーションのパフォーマンスに影響します。標準設定では，追跡は無効になっています。
- [**スタック トレースの上限**]：スタックに格納される呼び出しの最大数。標準設定値は 20 件までです。呼び出しの数が上限を超えた場合，VuGen はそれらを切り捨てます。

コード生成

[コードの生成] オプションでは，コード生成中に警告およびスタック・トレースを表示するかどうかを指定できます。

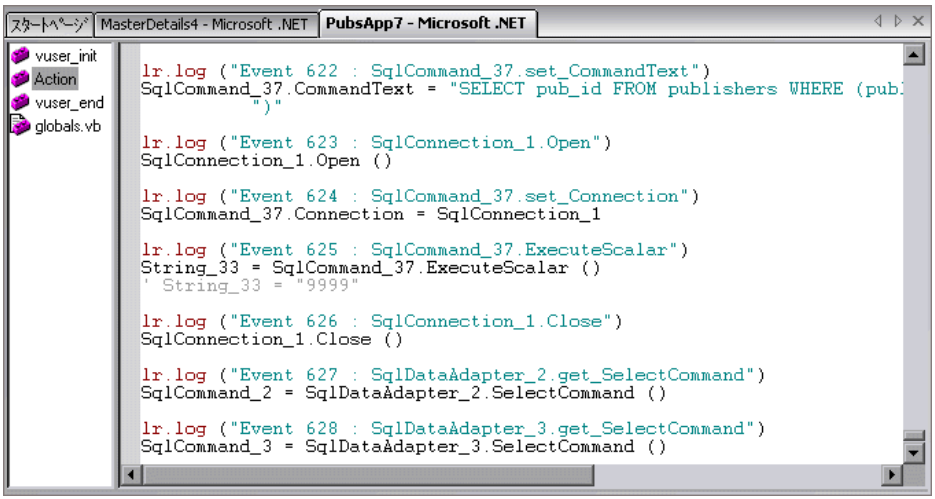
[**警告を表示する**]：コード生成中に発行された警告メッセージを表示します。

[**スタック トレースを表示する**]：記録されたスタック・トレースが存在する場合，表示します。

[**すべてのイベント サブスクリプションの表示**]：記録されたすべてのイベント・サブスクリプションのコードを生成します（標準設定では無効）。このオプションが無効な場合，VuGen は，発行者（イベントを発行するオブジェクト）と登録者（イベントの通知を受け取るオブジェクト）の両方がフィルタに追加されているイベントのコードのみを生成します。

VuGen と Visual Studio でのスクリプトの表示

記録後，VuGen のスクリプト・ビューでスクリプトを表示できます。



VuGen は発生したアクションを記録し，C# または VB コードを生成します。標準設定では，VuGen はすべてのステップを **lr.log** 呼び出しで囲みます。

スクリプトの再生時には，VuGen はまずスクリプトをコンパイルし，すべての呼び出しが有効で構文が正しいことを確かめます。VuGen はスクリプトを「**Script.dll**」という DLL ファイルとしてコンパイルし，スクリプトの **bin** フォルダに保存します。DLL ファイルには，Init，Actions，End という 3 つの関数が含まれます。

スクリプトを実行せずにコンパイルして構文を確認できます。VuGen からスクリプトを直接コンパイルするには，Shift + F5 キーを押すか，［**仮想ユーザ**］＞［**コンパイル**］を選択します。コンパイル・エラーが検出されると，［出力］ウィンドウにエラーが表示されます。エラーをダブルクリックすれば，スクリプトの問題が生じている行に移動できます。

VuGen からスクリプトを直接実行するには，F5 キーを押すか，［**仮想ユーザ**］＞［**実行**］を選択します。ブレークポイントおよびステップ単位の再生は，Microsoft .NET 仮想ユーザの場合には VuGen の エディタ・ウィンドウでサポートされていません。スクリプトをデバッグし，ブレークポイントを使用したりステップ単位で実行したりするには，次に説明に従って Visual Studio .NET の中で実行します。

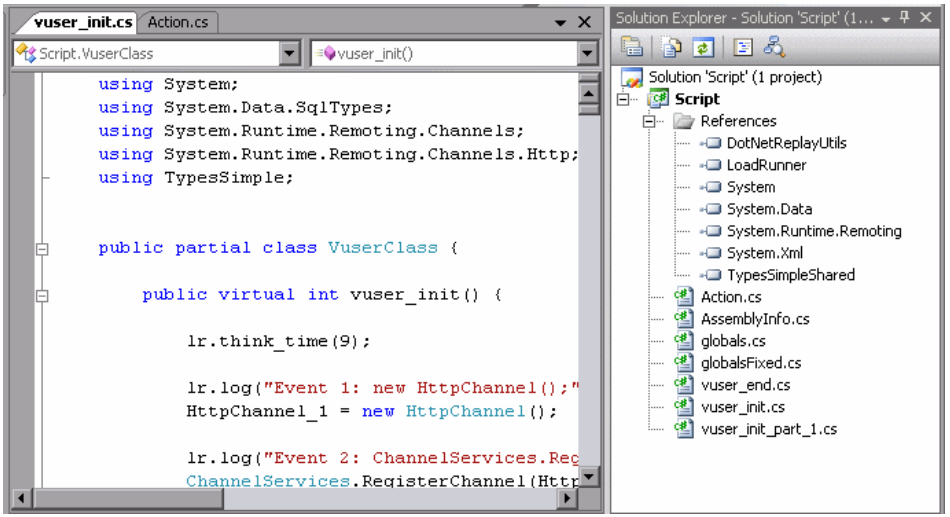
Visual Studio によるスクリプトの表示

Visual Studio は，スクリプトを表示，編集，およびデバッグするための付加的なツールを備えています。ブレークポイントの追加，変数値の表示，アセンブリ参照の追加，Visual Studio の IntelliSense を使用してのスクリプト編集ができます。また，デバッグのためにステップ単位でスクリプトを実行できます。

スクリプトの保存時に，スクリプトのフォルダに **Script.sln** という Visual Studio 2005 ソリューション・ファイルが作成されます。このソリューション・ファイルを Visual Studio .NET で開き，ソリューション・エクスプローラにすべてのコンポーネントを表示できます。

Visual Studio 2005 でソリューションを開くには，［**仮想ユーザ**］＞［**Visual Studio 2005 で開く**］を選択するか，VuGen のツールバーの［Visual Studio］ボタンをクリックします。

ソリューション・エクスプローラの中で **vuser_init.cs** など該当のセクションをダブルクリックし，スクリプトの内容を表示します。

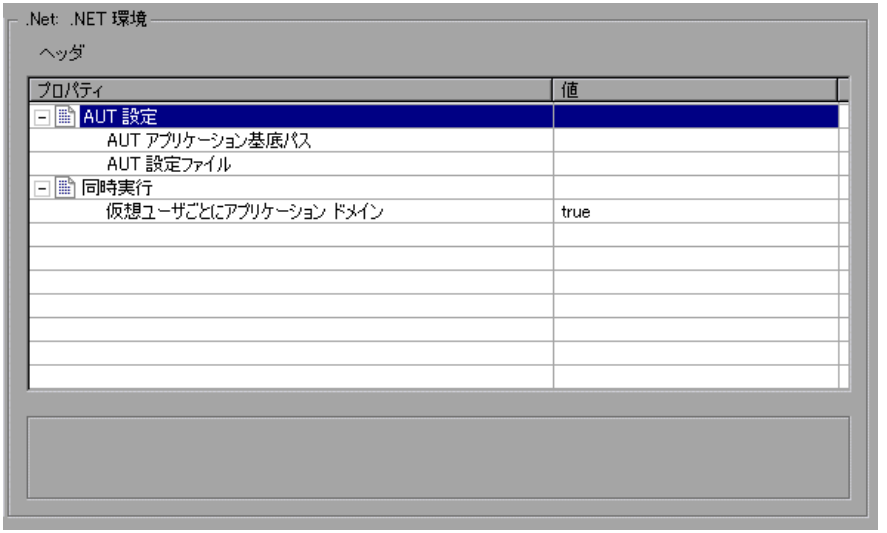


VuGen では，記録中に必要だったすべての参照は自動的にロードされます。ソリューション・エクスプローラを使用して，コンパイル時および再生時に使用する参照を追加できます。［**参照**］ノードを選択して，右クリック・メニューから［**参照の追加**］を選択します。

ソリューション・エクスプローラで **globals.cs** または **globals.vb** をクリックして，スクリプトで定義および使用された変数の一覧を表示します。

.NET 環境の実行環境の設定

Microsoft .NET 仮想ユーザ・スクリプトを実行する前に，.NET 環境設定を［実行環境設定］ダイアログ・ボックスから指定できます。



ペースや反復のオプションを設定するための一般的な Microsoft .NET スクリプトの実行時設定を設定することもできます。詳細については，第 13 章「実行環境の設定」を参照してください。

AUT 設定

［**AUT アプリケーション基底パス**］：再生時に DLL のロード元となる AUT（テスト対象アプリケーション）の基底ディレクトリです。標準設定では，記録時に必要なすべての DLL はスクリプトのディレクトリに格納されます。このオプションを使用して，見つからなかった AUT の DLL ファイルの場所を指定します。これは通常は，記録対象アプリケーションのインストール先パスです。AUT は，スクリプトを実行するマシンにインストールされている必要があります。このボックスを空白のままにしておくと，VuGen は再生時にアプリケーションの基底ディレクトリとしてローカル・スクリプト・ディレクトリを使用します。

［**AUT 設定ファイル**］：記録対象アプリケーションの設定ファイルのフル・パスとファイル名です。ロード・ジェネレータ・マシン上でスクリプトを実行する場合，アプリケーションのパスが正しいことを確認します。

同時実行

➤ [**仮想ユーザごとにアプリケーション ドメイン**]：各仮想ユーザを個別のアプリケーション・ドメインで実行できるようにします（標準設定では有効）。仮想ユーザを個別のアプリケーション・ドメインで実行すると，静的変数が仮想ユーザの間で共有されないので，互いをロックすることなしに仮想ユーザを個別に実行できます。

ADO.NET プロバイダは，負荷テストの精度に大きな影響を与える**接続プール**と呼ばれる機能を展開します。すべての仮想ユーザで使用されるアプリケーション・ドメインが 1 つだけのときは常に，接続プールは有効です。.NET Framework はデータベース接続を開いたまま維持し，新しい接続が要求された場合にそれらの再利用を試みます。多くの仮想ユーザが 1 つのアプリケーション・ドメインで実行されるため，仮想ユーザが互いに干渉する可能性があります。この動作は直線的ではなく，そのことが仮想ユーザの精度を下げる場合があります。標準設定では**有効**で，仮想ユーザごとに個別の接続プールが割り当てられます。つまり，仮想ユーザごとに接続プールはあるけれども，仮想ユーザは互いに干渉しません。この設定により精度は向上しますが，スケーラビリティは低下します。

このオプションを無効にした場合，データベースの接続プールを手作業で無効にする必要があります。次の表に，手作業で接続プールを無効にする方法を示します。

プロバイダ	オプション
.NET Framework Data Provider for SQL Server	"Pooling=false" または "Pooling=no"
.NET Framework Data Provider for Oracle	"Pooling=false" または "Pooling=no"
.NET Framework Data Provider for ODBC	接続プールは ODBC Driver Manager により管理されます。接続プールを有効または無効にするには，ODBC データ・ソース・アドミニストレータを使用します（[コントロール パネル] または [管理ツール] フォルダにあります）。[**接続プール**] タブでは，インストールされている ODBC ドライバごとに接続プール・パラメータを指定できます。
.NET Framework Data Provider for OLE DB	"OLE DB Services=-2"

Oracle Data Provider for .NET	"pooling=false"
Adaptive Server Enterprise ADO.NET Data Provider	"Pooling=False"

.NET リソースを指定するには，次の手順を実行します。

1 実行環境の設定を開きます。F4 キーを押すか，［**仮想ユーザ**］＞［**実行環境の設定**］を選択します。

2 左側の表示枠で［**.NET 環境**］ノードをクリックします。

3 ［**AUT アプリケーション基底パス**］ボックスに，DLL の基底フォルダを設定します。

4 ［**AUT 設定ファイル**］ボックスに，記録対象アプリケーションのパスを設定します。

5 ［**仮想ユーザごとにアプリケーション ドメイン**］を有効に設定することをお勧めします（標準設定）。

データセットとグリッドの表示

データセット，データ・テーブル，またはデータの読み込みアクションを返すメソッドを記録すると，データを表示するグリッドが生成されます。

データの読み込みを処理する場合，VuGen は各**読み取り**操作から取得したデータを収集し，再生ヘルパ関数 **DoDataRead** に変換します。

たとえば，次のようなアプリケーション・コードを記録したとします。

```
SqlDataReader reader = command.ExecuteReader();
    while( reader.Read() )
     {
        // カラム 1 にある文字列を取得するなど，値を読み取る
        string str = reader.GetString(1)
     }
```

VuGen はその後，スクリプトに次の行を生成します。

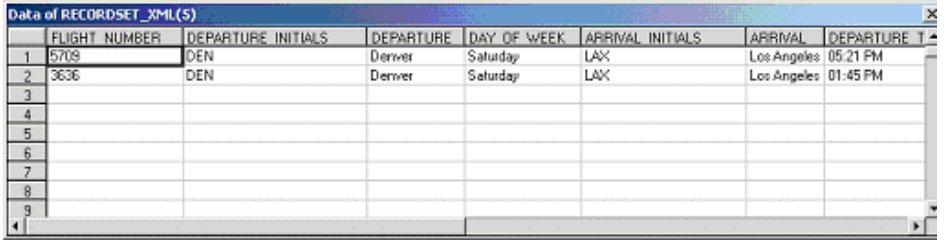
```
SqlDataReader_1 = SqlCommand_1.ExecuteReader();
LrReplayUtils.DoDataRead(SqlDataReader_1, out valueTable_1, true, 27);
```

ここで 2 つのパラメータは，記録時に，アプリケーションが取得可能な 27 のレコードをすべて読み込んだことを示します。したがって，再生時にスクリプトはすべての取得可能なレコードを読み込みます。

さらに，VuGen により，**読み取り**操作で取得したすべての情報を含むデータ・グリッドが生成されます。

再生時に，取得された実際の値を含んだ出力データ・テーブルを相関や検証に使用できます。**DoDataRead** 関数の詳細については，「**オンライン関数リファレンス**」（[**ヘルプ**] ＞ [**関数リファレンス**]）を参照してください。

標準設定では，スクリプトにはグリッドが表示されています。グリッドの表示を無効にし，グリッドを閉じた状態で表示させるには，[**表示**] ＞ [**データ グリッド**] を選択します。



データセットは XML ファイルに格納されます。XML ファイルはスクリプトの data/datasets フォルダにあります。データ・ファイルは，20.xml のように＜**インデックス名**＞ .xml ファイルの形式で表されます。1 つのファイルにいくつかのデータ・テーブルが含まれている可能性があるため，**datasets.grd** ファイルを参照してください。このファイルではデータを含む XML を特定するために，スクリプトのインデックスをファイルのインデックスに割り当てます。

グリッドの詳細については，537 ページ「グリッドを使った作業」を参照してください。

Microsoft .NET スクリプトの相関

セッションを記録し終えたら，スクリプト内の 1 つまたは複数の値を**相関**する必要がある場合があります。値の相関とは，スクリプトの再生中に値をキャプチャし，これをパラメータとして保存することです。このパラメータは，スクリプトのそれ以降の場所で使用できます。

VuGen では自動的に基本の相関が行われます。オブジェクトが関数呼び出しから返され，その後スクリプト内で呼び出されるか，またはそのオブジェクトが別のメソッドに渡される場合，VuGen は同じオブジェクト・インスタンスを使用します。

コーディングするか VuGen の組み込み相関ツールを使用してパラメータを手作業で保存することにより，スクリプト内の値をさらに相関できます。

VuGen 内で値を相関させるには，データセット内で値を探し出し，値を右クリック・メニューを使用してパラメータに保存します。ADO.NET 環境の値を相関させるには，次の手順を実行します。

1 **スクリプト内でデータセットを見つけます。**

スクリプトにグリッドを表示し，返されたデータセットを表示します。グリッドが表示されない場合は，［**表示**］＞［**データ グリッド**］を選択するか，適切な **DATASET_XML** ステートメントを展開します。次に例を示します。

	CustomerID	CompanyName	ContactName	ContactTitle	Address
1	ABC	ABC Company	John Smith	Owner	One My Way
2	ALFKI	Alfreds Futterkiste	Maria Anders	Sales Representative	Obere Str. 57
3	ANATR	Ana Trujillo Emparec	Ana Trujillo	Owner	Avda. de la C
4	ANTON	Antonio Moreno Tac	Antonio Moreno	Owner	Mataderos 2
5	AROUT	Around the Horn	Thomas Hardy	Sales Representative	120 Hanover

2 **値を探します。**

相関させる値を検索します。グリッド内で値を検索するには，［検索］ダイアログ・ボックスを開き（Ctrl+F），［**グリッド内を検索**］オプションを選択します。

3 **相関を作成します。**

相関させる値をグリッドの中でクリックし，右クリック・メニューから［**相関の作成**］を選択します。［相関の作成］ダイアログ・ボックスが開きます。

4 **パラメータ名を指定します。**

先に定義した変数と同一のパラメータ名を指定します。［**OK**］をクリックします。すべての候補を検索するかどうかを尋ねられます。［**OK**］をクリックします。

VuGen は各データセットの前に **lr.save_string** 関数を追加します。次に例を示します。

```
lr.save_string("MyCustomerID",
CustomerAndOrdersDataSet_3.Tables[“Customers”].Rows[0]["CompanyName"].ToString());
```

5 **スクリプトの以降の場所でパラメータを参照します。**

パラメータで置換する値を選択して，右クリック・メニューから［**パラメータで置換**］を選択します。保存した変数名を［**パラメータ名**］ボックスに入力します。［**OK**］をクリックします。文字列の値を評価する **lr.eval_string** 関数を使用してすべての値をパラメータで置換するかどうかを尋ねられます。

```
lr.message(“The customer ID is ”+ lr.eval_string("{MyCustomerID}") + ”);
```

ほかのプロトコルとは異なり，スクリプトにはアプリケーションまたはフレームワーク・メソッドに対する直接の呼出しが含まれます。したがって，文字列値を {paramName} で置換することはできません。代わりにパラメータの値を評価する **lr.eval_string** を使用する必要があります。

この方法は ADO.NET 環境に対して適用できます。プリミティブな値については，出力パラメータ値を含んだスクリプトを生成し，出力パラメータを調べて相関させる必要があります。

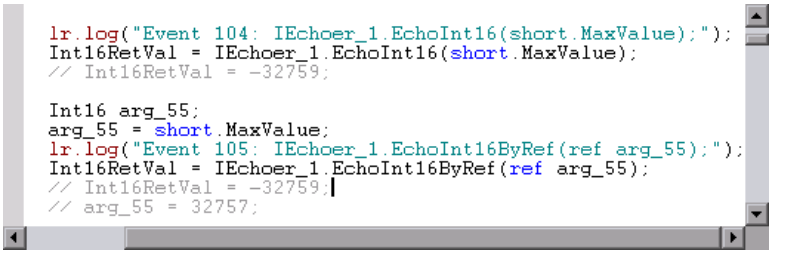
出力パラメータと相関させるには，次の手順を実行します。

1 ［**ツール**］＞［**記録オプション**］を選択し，［**一般：スクリプト**］ノードを選択します。

2 ［**Insert output parameter values**］オプションを有効にします。［**OK**］をクリックして［記録オプション］を閉じます。

3 ［**ツール**］＞［**スクリプトの再生成**］を選択して，スクリプトを再生成します。

4 コメントになっている出力プリミティブ値を検索して相関させます。



```
lr.log("Event 104: IEchoer_1.EchoInt16(short.MaxValue);");
Int16RetVal = IEchoer_1.EchoInt16(short.MaxValue);
// Int16RetVal = -32759;

Int16 arg_55;
arg_55 = short.MaxValue;
lr.log("Event 105: IEchoer_1.EchoInt16ByRef(ref arg_55);");
Int16RetVal = IEchoer_1.EchoInt16ByRef(ref arg_55);
// Int16RetVal = -32759;
// arg_55 = 32757;
```

相関関数の使用法の詳細については，「**オンライン関数リファレンス**」（[**ヘルプ**] > [**関数リファレンス**]）を参照してください。

アプリケーションのセキュリティと権限の設定

アプリケーションの記録中に発行されるセキュリティ例外は，記録を行うマシンにアプリケーションを記録するための十分な権限がないという，権限の不足によるものが一般的です。これは，アプリケーションがローカル・マシンになく，インターネットやネットワーク上にある場合によく起こります。

この問題を解決するには，記録マシンからアプリケーションおよびスクリプトに Full Trust 権限でアクセスできるようにする必要があります。

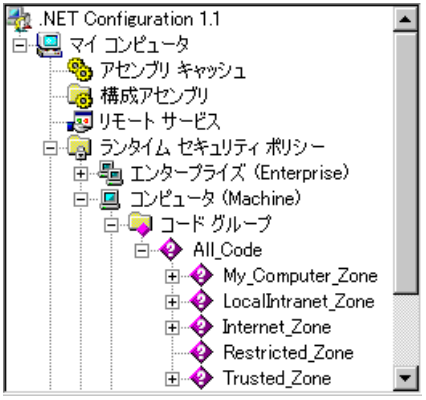
1 つの解決方法は，アプリケーションをローカル・マシンにコピーし，スクリプトをローカル・マシンに保存するというものです。ユーザは，ローカル・アプリケーションおよびフォルダには標準設定で Full Trust 権限を持つからです。

もう 1 つの解決方法は，各アプリケーション・フォルダおよびスクリプト・フォルダに Full Trust 権限を付与する新しいコード・グループを作成するというものです。

Full Trust 権限を特定のフォルダに付与するには，次の手順を実行します。

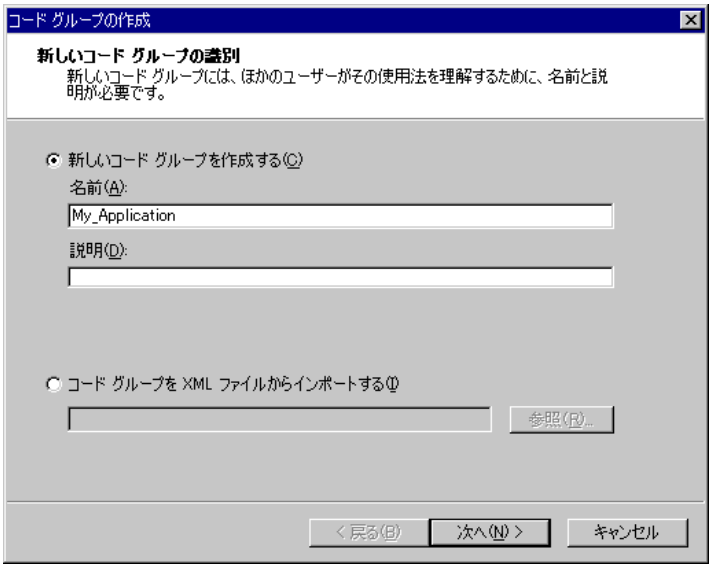
1 ［.NET Configuration］の設定を開きます。［**スタート**］＞［**プログラム**］＞［**管理ツール**］＞［**Microsoft .NET Framework 2.0 Configuration**］を選択します。［.NET Configuration］ウィンドウが開きます。

2 ［**ランタイム セキュリティ ポリシー**］ノードを展開して，マシンのコード・グループを表示します。



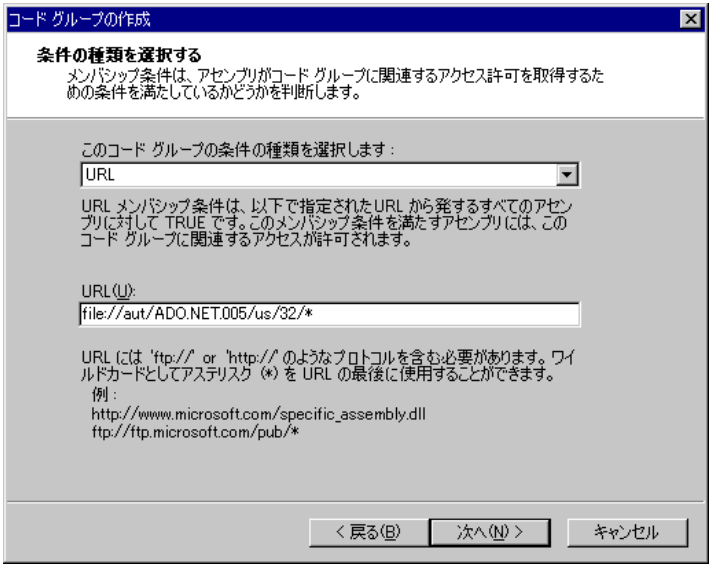
3 ［**All_Code**］ノードを選択します。

4［**操作**］＞［**新規作成**］を選択します。［コード グループの作成］ダイアログ・ボックスが表示されます。

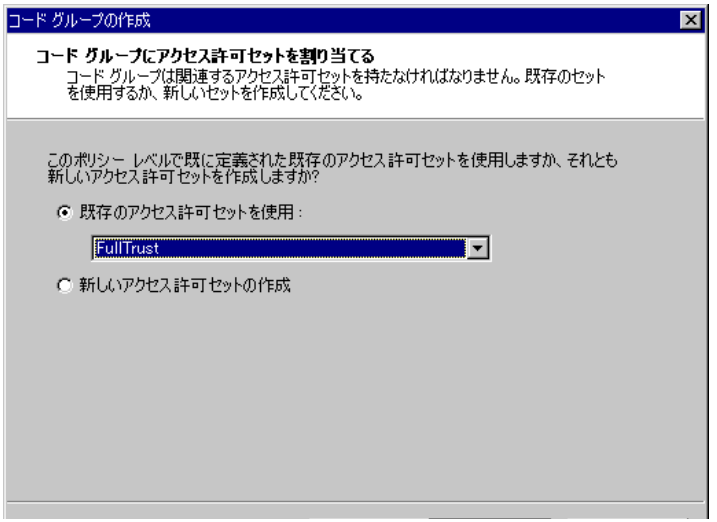


5 アプリケーションまたはスクリプトに新規コード・グループの名前を入力します。［**次へ**］をクリックします。

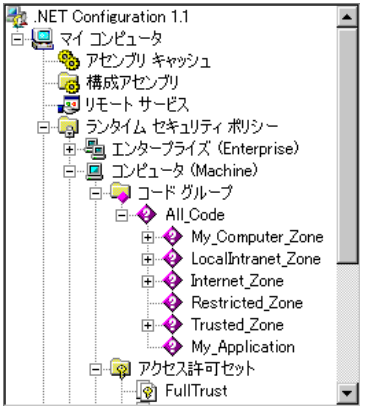
6 条件の種類に［**URL**］を選択します。［URL］ボックスで，アプリケーションまたはスクリプトのフル・パスを指定し（file://... 形式で），［**次へ**］をクリックします。



7 アクセス許可セットに［**FullTrust**］を選択します。［**次へ**］ボタンをクリックします。



8 ［ウィザードの完了］ダイアログ・ボックスで［**完了**］をクリックします。設定ツールにより，既存のグループの一覧に新しいコード・グループが追加されます。



9 記録するすべての .NET アプリケーションに上記の手順を繰り返します。

10 仮想ユーザ・スクリプト・フォルダに，上記の手順を繰り返します。

注：スクリプト・フォルダに，テストに参加しているすべてのロード・ジェネレータ・マシンに対する **FullTrust** 権限があることを確認してください。

第 48 章

Web 仮想ユーザ・スクリプトの作成

VuGen を使用して，クライアント・ブラウザ操作時のユーザ・アクションに基づく Web 仮想ユーザ・スクリプトを作成します。

本章では，次の項目について説明します。

- Web レベル仮想ユーザ・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 仮想ユーザのタイプの選択
- 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（Click and Script）仮想ユーザ・スクリプトおよび Web（HTML/HTTP）仮想ユーザ・スクリプトを対象とします。

Web レベル仮想ユーザ・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に組み込む方法の詳細については，『**LoadRunner コントローラ・ユーザーズ・ガイド**』を参照してください。

Web（Click and Script）仮想ユーザについては，記録されたスクリプトとイベントに関する情報が記載された，Microsoft Word 形式のビジネス・プロセス・レポートを作成できます。詳細については，第 6 章「ビジネス・プロセス・レポートの作成」を参照してください。

Web 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。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，Web サーバが同時に複数の要求に対してサービスを提供する状況をエミュレートします。このとき各仮想ユーザは次のような操作を行うものと考えられます。

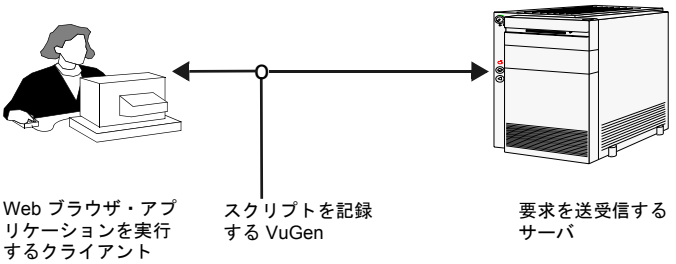
- ホーム・ページのロード
- 製品情報が掲載されているページへの移動
- クエリの送信
- サーバからの応答の待機

利用可能な複数のテスト用マシンに，数百の仮想ユーザを分散配置できます。各仮想ユーザでは API を通じてサーバにアクセスします。これにより，多数のユーザによる負荷の下でサーバのパフォーマンスを測定できます。

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・スクリプトを実行するとき，仮想ユーザはサーバと直接通信し，クライアント・ソフトウェアに依存しません。仮想ユーザ・スクリプトでは，クライアント・ソフトウェアを使わず，API 関数を使って Web サーバへの呼び出しを直接実行します。

Web 仮想ユーザのタイプの選択

新しい Web 仮想ユーザ・スクリプトを作成する場合，2 つのタイプの Web 仮想ユーザから選択できます。

- Web（Click and Script）
- Web（HTTP/HTTPS）

Web（Click and Script）

Web（Click and Script）仮想ユーザは，ユーザ・アクション GUI レベルで Web セッションを記録するためのソリューションです。VuGen は，Web インタフェースを対象としたアクションを直感的に表現する GUI レベルのスクリプトを作成します。たとえば，ボタンをクリックして情報を送信すると **web_button** 関数が生成され，エディット・ボックスにテキストを入力すると **web_edit_field** 関数が生成されます。

Web（Click and Script）仮想ユーザは，JavaScript などのクライアント側の非 HTML コードをサポートします。VuGen は，Web ページでのユーザ・アクションを正確にエミュレートする直感的なスクリプトを作成します。これに対して，Web（HTTP/HTML）スクリプトは JavaScript をサポートしません。その代わり，JavaScript のコードはページの **web_url** 関数のサブリソースとして含められます。

Web（Click and Script）仮想ユーザは大部分の相関を自動的に処理するため，スクリプト作成時間が短縮されます。ほとんどの場合，相関のルールを定義したり，記録後に手作業で相関を実行したりする必要はありません。

また，Web（Click and Script）仮想ユーザでは，スクリプトとその再生結果についてまとめた詳細なビジネス・プロセス・レポートを生成することもできます。

注：Web （Click and Script）仮想ユーザは，アプレットおよび VBScript をサポートしません。テスト対象 Web サイトにこれらの項目が含まれている場合は，Web（HTTP/HTML）ユーザを使用します。

Web（HTTP/HTTPS）

Web（HTTP/HTML）スクリプトを記録すると，VuGen は，インターネット経由の HTTP トラフィックおよびサーバ応答を記録します。スクリプトには，ブラウザにおけるユーザのアクションに関する詳細情報が含まれます。

Web（HTTP/HTML）仮想ユーザには，2 つの記録レベルがあります。**HTML ベースのスクリプト**と **URL ベースのスクリプト**の 2 つです。これらのレベルにより，仮想ユーザ・スクリプトの生成時に記録する情報および使用する関数を指定できます。記録レベルの選択の詳細については，829 ページ「記録レベルの選択」を参照してください。

この仮想ユーザ・タイプは，JavaScript をネイティブではサポートしません。その代わり，JavaScript を Web ページのリソースとして格納します。

ヒント：JavaScript を使用するアプリケーションを含むほとんどのアプリケーションに対しては，Web （Click and Script）仮想ユーザを使用します。アプレットまたは VBScript を使用するブラウザ・アプリケーション，または非ブラウザ・アプリケーションの場合は，Web（HTTP/HTML）仮想ユーザを使用します。

Web 仮想ユーザ・スクリプト入門

本項では，Web 仮想ユーザ・スクリプトを作成する工程の概要を説明します。

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。**

VuGen の［**スタート ページ**］タブをクリックし，［**新規仮想ユーザ スクリプト**］をクリックします。シングル・プロトコル・モードまたはマルチ・プロトコル・モードで，［**e ビジネス**］カテゴリから Web（Click and Script）または Web（HTTP/HTML）仮想ユーザ・スクリプトを選択します。

新規スクリプトの作成の詳細については，第 4 章「VuGen を使った記録」を参照してください。

2 **記録オプションを設定します。**

記録オプションを設定します。記録オプションの設定については，第 51 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。

記録レベルの選択の詳細については，第 52 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。

Web（Click and Script）固有のオプションの詳細については，第 49 章「Web（Click and Script）仮想ユーザ・スクリプトでの作業」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 4 章「VuGen を使った記録」を参照してください。

4 **記録した仮想ユーザ・スクリプトの機能を拡張します。**

トランザクション，ランデブー・ポイント，チェック，サービス・ステップを挿入して，仮想ユーザ・スクリプトを拡張します。

詳細については，第 55 章「負荷下の Web ページ検証」，第 56 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 57 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 9 章「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 13 章「実行環境の設定」を参照してください。

7 **相関を実行します。**

Web（HTTP/HTML）スクリプトの場合，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。

自動相関の設定方法の詳細については，第 57 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。記録後の相関の詳細については，第 58 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 15 章「スタンドアロン・モードでの仮想ユーザ・スクリプトの実行」および第 16 章「テスト結果の表示」を参照してください。

スクリプトを作成した後，スクリプトを自分の環境（LoadRunner シナリオ，Performance Center 負荷テスト，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，『**LoadRunner コントローラ**』，**Tuning Console** のマニュアル，**Performance Center** のマニュアル，または **Mercury Business Availability Center** 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，サーバのログオンとログオフ手続きの記録に使います。これらのセクションは仮想ユーザ・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。

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 49 章

Web（Click and Script）仮想ユーザ・スクリプトでの作業

Web（Click and Script）仮想ユーザは，ユーザ・アクション GUI レベルで Web セッションを記録するためのソリューションです。

本章では，次の項目について説明します。

- Web（Click and Script）仮想ユーザ・スクリプトでの作業について
- Web（Click and Script）仮想ユーザ・スクリプトの表示
- Web（Click and Script）記録オプションの設定
- Web（Click and Script）仮想ユーザで作業をする際のヒント

以降の情報は，Web（Click and Script），Oracle Web Applications 11i，および PeopleSoft Enterprise の各プロトコルを対象とします。

Web（Click and Script）仮想ユーザ・スクリプトでの作業について

Web（Click and Script）仮想ユーザは，ユーザ・アクション GUI レベルで Web セッションを記録するためのソリューションです。VuGen は，Web インタフェースを対象としたアクションを直感的に表現する GUI レベルのスクリプトを作成します。たとえば，ボタンをクリックして情報を送信すると **web_button** 関数が生成され，エディット・ボックスにテキストを入力すると **web_edit_field** 関数が生成されます。

これに対して，Web（HTTP/HTML）スクリプトは，低レベルのスクリプトであり，多くの場合は直観性が下がります。

Web セッションに対するスクリプト・タイプの選択の詳細については，756 ページ「Web 仮想ユーザのタイプの選択」を参照してください。

Web（Click and Script）仮想ユーザ・スクリプトの表示

通常，Web （Click and Script）仮想ユーザ・スクリプトには，ビジネス・プロセスを構成する複数のアクションが含まれています。GUI レベルで生成された記録済みの関数を見れば，記録セッションの間のユーザの正確なアクションを知ることができます。

たとえば，標準的な記録では，第 1 段階にサインインのプロセスが含まれています。ブラウザでサインイン・ページが開いたら，ユーザはユーザ名とパスワードを入力し，［**Sign In**］をクリックしてサインインします。

VuGen は，エディット・フィールドに入力されたデータを表す **web_edit_field** 関数を生成します。次の例では，ユーザは［**userid**］フィールドにテキストを入力し，暗号化される［**pwd**］フィールドにパスワードを入力しています。

```
vuser_init()
{
    web_browser("mercuryWebTours",
        DESCRIPTION,
        ACTION,
        "Navigate=http://localhost:1080/mercuryWebTours/",
        LAST);

    web_edit_field("username",
        "Snapshot=t2.inf",
        DESCRIPTION,
        "Type=text",
        "Name=username",
        "FrameName=navbar",
        ACTION,
        "SetValue=jojo",
        LAST);

    web_edit_field("password",
        "Snapshot=t3.inf",
        DESCRIPTION,
        "Type=password",
        "Name=password",
        "FrameName=navbar",
        ACTION,
        "SetEncryptedValue=440315c7c093c20e",
        LAST); ...
```

ボタンをクリックしてデータを送信すると，VuGen は **web_button** を生成します。ボタンが画像の場合は，**web_image_submit** が生成されます。次の例では，ユーザは［**login**］ボタンをクリックしています。

```
…
web_image_submit("Login",
        "Snapshot=t4.inf",
        DESCRIPTION,
        "Alt=Login",
        "Name=login",
        "FrameName=navbar",
        ACTION,
        "ClickCoordinates=31,6",
        LAST);}
```

次の項では，ユーザが「**Manage Assets**」分岐の下にある **Asset ExpressAdd** プロセスに移動する標準的なアクションを示します。ユーザは，対象となる分岐のテキスト・リンクをクリックすることで移動します。**web_text_link** 関数が生成されます。

```
web_text_link("Manage Assets_2",
        DESCRIPTION,
        "Text=Manage Assets",
        "Ordinal=2",
        "FrameName=main",
        LAST);

web_text_link("Use",
        DESCRIPTION,
        "Text=Use",
        "FrameName=main",
        LAST);

web_text_link("Asset ExpressAdd",
        DESCRIPTION,
        "Text=Asset ExpressAdd",
        "FrameName=main",
        LAST);
```

次の例では，仮想ユーザ関数によって，リスト項目の選択などの一般的なユーザ・アクションをエミュレートしています。

```
...
web_list("Year",
        DESCRIPTION,
        "Name=Year",
        "FrameName=CalFrame",
        ACTION,
        "Select=2000",
        LAST);
```

画像マップに関連付けられている画像をクリックすると，VuGen によって **web_map_area** 関数が生成されます。

```
web_map_area("map2_2",
        DESCRIPTION,
        "MapName=map2",
        "Ordinal=20",
        "FrameName=CalFrame",
        LAST);
```

Web（Click and Script）記録オプションの設定

スクリプトの記録を開始する前に，記録の対象と，記録時にスクリプトを生成する方法を VuGen に対して指定する記録オプションを設定できます。

一般，HTTP プロパティ，ネットワークの項目について共通の記録オプションの設定が可能です。

次項では，Web （Click and Script），Oracle Applications および PeopleSoft Enterprise の仮想ユーザに固有の GUI プロパティ記録オプションについて説明します。これらの設定は，記録するべきイベントと，それぞれのオブジェクトのどのプロパティを含めるべきかを指定します。

他の記録オプションの詳細については，該当する次の各項を参照してください。

➤ [**一般**：**スクリプト**]：第 5 章「スクリプト生成オプションの設定」を参照してください。

- ［**一般：記録**］：第 52 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。
- ［**ネットワーク：ポートの割り当て**］：第 7 章「ポートの割り当て設定」を参照してください。
- ［**HTTP プロパティ：詳細**］：819 ページ「記録オプションの詳細設定」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**］および［**ページタイトルで web_reg_find 関数を生成する**］オプションは GUI ベースのスクリプトには適用されません（次の GUI ベースのスクリプトの説明を参照）。
- ［**HTTP プロパティ：相関**］：第 57 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。Oracle および PeopleSoft のサーバに対しては組み込みのルールが存在します。それらのルールを有効にするには，サーバ名の横にあるチェック・ボックスを選択します。

  ほかにいくつかある HTTP プロパティは，Web（HTTP/HTML）スクリプトの場合にのみ設定できます。

GUI プロパティの詳細設定

次の範囲について，Web （Click and Script）記録の詳細オプションを設定できます。

- 記録設定
- コード生成設定

記録設定

記録設定により，何を記録するかを指定します。次の機能を有効または無効にできます。

［描画関連のプロパティ値の記録］

DOM オブジェクトのレンダリング関連プロパティ（offsetTop など）の値を記録して，再生時に利用できるようにします。これにより，再生速度が大幅に低下することがあります（標準設定では無効）。

[マウス イベントによる「クリック」を記録する]

Click () メソッドではなく，マウス・イベントをキャプチャすることでマウスのクリックを記録します。記録対象アプリケーションで DOM の click() メソッドが使用されている場合に有効にします。同じユーザ・アクションに対して複数の関数が生成されるのを防ぎます（標準設定では有効）。

[ソケット レベル データを記録する]

ソケット・レベル・データの記録を有効にします。このオプションを無効にした場合は，記録の前に，開始 URL を手動で追加する必要があります。また，スクリプトを HTML レベルで再生成できなくなります（標準設定では有効）。

コード生成設定

コード生成設定により，記録後のスクリプトの生成方法を指定できます。次の機能を有効または無効にできます。

[コンテキスト外ステップの生成を有効にする]

ActiveX コントロールや Java アプレットについては，URL ベースのスクリプトを作成するよう VuGen に指示することで，再生できます。これらの関数はネイティブの記録の一部ではないため，コンテキスト外の記録と呼ばれます（標準設定で無効）。

次の例は，コンテキスト外の記録オプションを有効にしてスクリプトを再生成したものです。

```
web_image_link("Search Flights Button",
        "Snapshot=t5.inf",
        DESCRIPTION,
        "Alt=Search Flights Button",
        "FrameName=navbar",
        ACTION,
        "ClickCoordinates=58,9",
        LAST);

web_add_cookie("MSO=SID&1141052844; DOMAIN=localhost");

web_add_cookie("MTUserInfo=hash&47&firstName&Joseph&expDate&%
0A&creditCard&&address1&234%20Willow%20Drive&lastName&Marshall
%0A&address2&San%20Jose%2FCA%2F94085&username&jojo;
DOMAIN=localhost");

web_url("FormDateUpdate.class",

"URL=http://localhost:1080/mercuryWebTours/FormDateUpdate.class",
        "Resource=0",
        "RecContentType=text/html",
        "Referer=",
        "UserAgent=Mozilla/4.0 (Windows 2000 5.0) Java/1.4.2_08",
        "Mode=HTTP",
        LAST);
…
```

このオプションを無効にすると，VuGen は，ActiveX コントロールおよび Java アプレットのコードを生成しません。次の例では，**web_image_link** 関数のみが生成され，クラス・ファイルが含まれる **web_url** 関数は生成されません。

```
web_image_link("Search Flights Button",
        "Snapshot=t5.inf",
        DESCRIPTION,
        "Alt=Search Flights Button",
        "FrameName=navbar",
        ACTION,
        "ClickCoordinates=58,9",
        LAST);
```

［ブラウザ・タイトルの自動検証を有効にする］

ブラウザ・タイトルの自動検証を有効にします（標準設定では無効）。

タイトル検証のタイプをカスタマイズすることもできます。

- ［**タイトル検証を行う対象**］
  - ［**ナビゲーションのたびに**］：ナビゲーションの後にのみ，タイトル検証を行います。フィールドが複数あるフォームの入力など，ユーザが同じページでいくつかの操作を実行した場合，タイトルは変わらないため，検証は不要です。
  - ［**ステップごとに**］：ステップごとにタイトル検証を行い，ステップによってブラウザのタイトルが変更されていないことを確認します。ブラウザのタイトルが変更されると，スクリプトが失敗する原因となることがあります。
- ［**タイトルがない場合に，URL を使用してタイトル検証を行います。**］：ブラウザ・ウィンドウにタイトルがない場合に，その URL を使用してステップごとにタイトル検証を行います。

Web イベント記録の設定

VuGen は，ドキュメントの表示中のマウスのクリックやキーの押下などのユーザ・アクションをきっかけとする HTML オブジェクト・イベントを記録することでスクリプトを作成します。

記録する必要のあるイベントが，VuGen の標準設定で自動的に記録されるものより多い場合や少ない場合があります。そのような場合，［記録オプション］の［Web イベント設定］ノードで，3 つの定義済みの設定から，いずれかを選択することで標準のイベント記録設定を変更できます。また，特定の条件に合わせて，イベント記録の設定を個別にカスタマイズすることもできます。

本項では，Web イベントの VuGen による処理の設定方法について説明します。

- 標準で使用するイベント記録設定の選択
- イベント記録設定のカスタマイズ
- オブジェクトのリッスン・イベントの追加と削除
- イベントのリッスン設定と記録設定の変更
- イベント記録設定のリセット

たとえば VuGen では，通常はリンク・オブジェクト上の mouseover イベントは記録されません。しかし，マウスを対象の上に移動するマウスオーバ・ハンドラがリンクに関連付けられている場合は，mouseover イベントを記録することが重要になるかもしれません。この場合，リンク・オブジェクトがハンドラに関連付けられている場合には，リンク・オブジェクト上の mouseover イベントが必ず記録されるように，設定をカスタマイズできます。

注：イベント設定はグローバルな設定のため，設定を変更した後で記録されるすべてのテストに影響します。

イベント設定の変更は，すでに記録されたテストには影響しません。必要なイベントが VuGen で記録されなかった場合や，不要なイベントが記録された場合は，イベント記録設定を変更し，テストの中でその変更の影響を受ける部分を再度記録します。

［ユーザ定義 Web イベント記録設定］の設定に対する変更は，ブラウザを開いているときには効果がありません。［記録オプション］の［Web イベント設定］ノードに必要な変更を行い，その変更を既存のテストに適用するには，開いているブラウザを更新し，新しい記録セッションを開始します。

標準で使用するイベント記録設定の選択

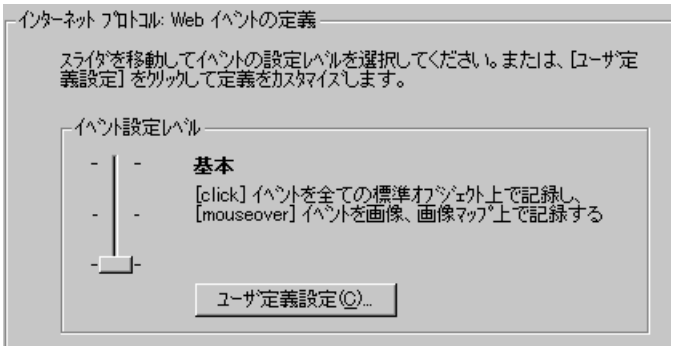
［記録オプション］の［Web イベント設定］ノードでは，［**基本**］，［**中**］，［**高**］の 3 つの定義済みのイベント設定レベルから選択できます。標準設定の場合，VuGen では**基本**設定レベルから使用されます。必要なすべてのイベントが VuGen で記録されない場合は，より高いレベルのイベント設定が必要になります。

レベル	説明
［基本］	**標準設定** • 画像，ボタン，ラジオ・ボタンなどの標準的な Web オブジェクトに対するクリック・イベントを必ず記録します。 • フォーム内での送信イベントを必ず記録します。 • ハンドラまたは動作が関連付けられているその他のオブジェクトでのクリック・イベントを記録します。ハンドラおよび動作の詳細については，776 ページ「リッスン条件」を参照してください。 • イメージおよびイメージ・マップ上の mouseover イベントは，そのイベントの後に同じオブジェクトに対するイベントが実行される場合にのみ記録します。
［中］	**基本**レベルで記録されるオブジェクトに加えて，HTML タグ・オブジェクトの <DIV>，<SPAN>，<TD> に対するクリック・イベントも記録します。
［高］	**中**レベルで記録されるオブジェクトに加えて，ハンドラまたは動作が関連付けられているオブジェクトに対する mouseover イベント，mousedown イベント，および double-click イベントを記録します。 ハンドラおよび動作の詳細については，776 ページ「リッスン条件」を参照してください。

標準で使用するイベント記録の設定を設定するには，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**］を選択します。

2［**GUI プロパティ：Web イベント設定**］ノードを選択します。



3 スライダを使って，標準で使用するイベント記録設定を選択します。［**基本**］，［**中**］，［**高**］を選択できます。

ヒント：［**ユーザ定義設定**］ボタンをクリックして，［ユーザ定義 Web イベント記録設定］ダイアログ・ボックスを開きます。ここでイベント記録の設定を定義できます。詳細については，次の「イベント記録設定のカスタマイズ」を参照してください。

4［**OK**］クリックします。

イベント記録設定のカスタマイズ

標準のイベント設定レベルで必要な記録が行われない場合は，［ユーザ定義 Web イベント記録設定］ダイアログ・ボックスを使って，イベント記録設定をカスタマイズできます。

画像，リンク，WebArea，WebButton など，標準的な Web 要素に対する Web イベントをカスタマイズできます。また，選択した HTML タグの記録動作を設定することもできます。必要な HTML タグ・オブジェクトを追加してから，その記録動作を設定します。たとえば，すべての **DIV** タグのすべての **mouseover** イベントを記録するように VuGen を設定できます。

［ユーザ定義 Web イベント記録設定］ダイアログ・ボックスでは，いくつかの方法でイベント記録をカスタマイズできます。次のことができます。

- VuGen で特別なリッスン設定または記録設定を適用する対象となるオブジェクトの有効化または無効化。
- VuGen がリッスンするべきイベントの追加または削除。
- イベントのリッスンと記録の設定変更

次のオプションを使って，イベント記録設定を変更できます。

オプション	説明
［イベント名］	オブジェクトに関連付けられているイベントのリストが表示されます。 • イベント表示枠にイベントを追加するには，［**イベント**］＞［**追加**］を選択します。使用するイベントを選択します。 • イベントを削除するには，［イベント名］カラムでイベントをクリックし，［**イベント**］＞［**削除**］を選択します。 詳細については，775 ページ「オブジェクトのリッスン・イベントの追加と削除」を参照してください。
［応答］	VuGen がいつイベントをリッスンするかの基準 • ［**常に**］：常にイベントをリッスンします。 • ［**ハンドラの場合**］：ハンドラが付加されているイベントをリッスンします。ハンドラは，Web ページに含まれているコードであり，通常はスクリプト言語で書かれている関数またはルーチンです。対応するイベントが発生したときに制御が渡されます。 • ［**動作の場合**］：DHTML 動作が付加されているイベントをリッスンします。DHTML 動作は，ページにおける特定の機能や振る舞いをカプセル化します。ページ上の標準的な HTML 要素に適用されている場合，その要素の標準設定の動作が拡張されます。 • ［**ハンドラまたは動作の場合**］：ハンドラまたは動作が付加されているイベントをリッスンします。 • ［**しない**］：イベントを一切リッスンしません。 詳細については，776 ページ「イベントのリッスン設定と記録設定の変更」を参照してください。

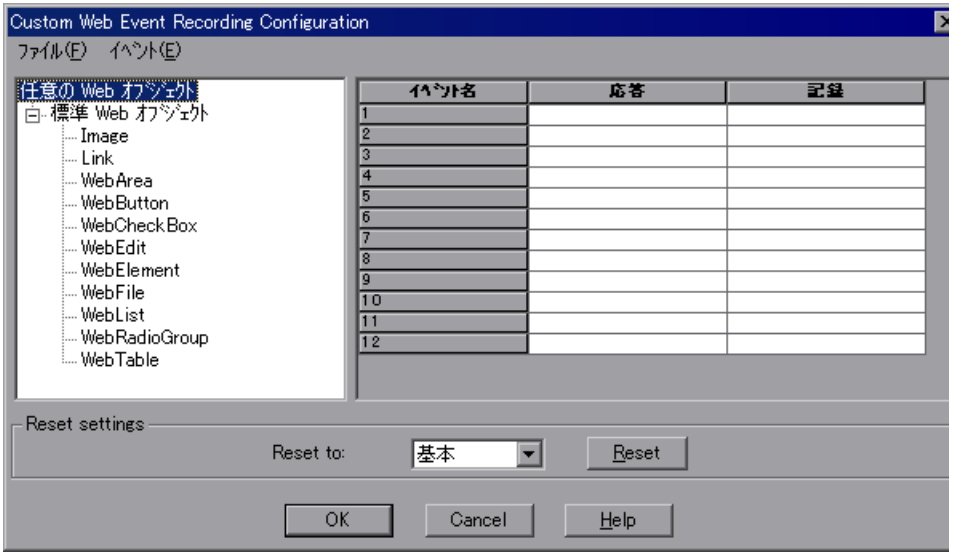
オプション	説明
[記録]	選択されたオブジェクトのイベントの記録を有効 / 無効にする，あるいは同じオブジェクトに対してその後イベントが発生した場合のみイベントの記録を有効 / 無効にします。
[リセット]	あらかじめ設定されているレベルにリセットします。[**基本**]，[**中**]，[**高**] を選択できます。

イベント記録の設定をカスタマイズするには，次の手順を実行します。

1 [記録オプション] ダイアログ・ボックスを開きます。[**ツール**] > [**記録オプション**] を選択します。

2 [**GUI プロパティ : Web イベント設定**] ノードを選択します。

3 [**ユーザ定義設定**] ボタンをクリックします。[ユーザ定義 Web イベント記録設定] ダイアログ・ボックスが開きます。



4 オブジェクトを指定します。

➤ 組み込み Web オブジェクトのいずれかを設定するには，左側の表示枠でそのオブジェクトを選択します。

➤ 別の HTML タグ・オブジェクトを指定するには，［**オブジェクト**］＞［**追加**］を選択し，イベントを記録する HTML タグの名前を入力します。

5 前述のように，右側の表示枠で，イベントごとに［**応答**］と［**記録**］の動作を指定します。記録したいイベントが右側の表示枠に表示されない場合は，後述のとおり，［**イベント**］＞［**追加**］を選択してイベントを追加します。

6 ［**OK**］をクリックしてカスタマイズを保存します。

7 あらかじめ設定されているレベルにリセットするには，［**リセット**］をクリックします。［**基本**］，［**中**］，［**高**］を選択できます。

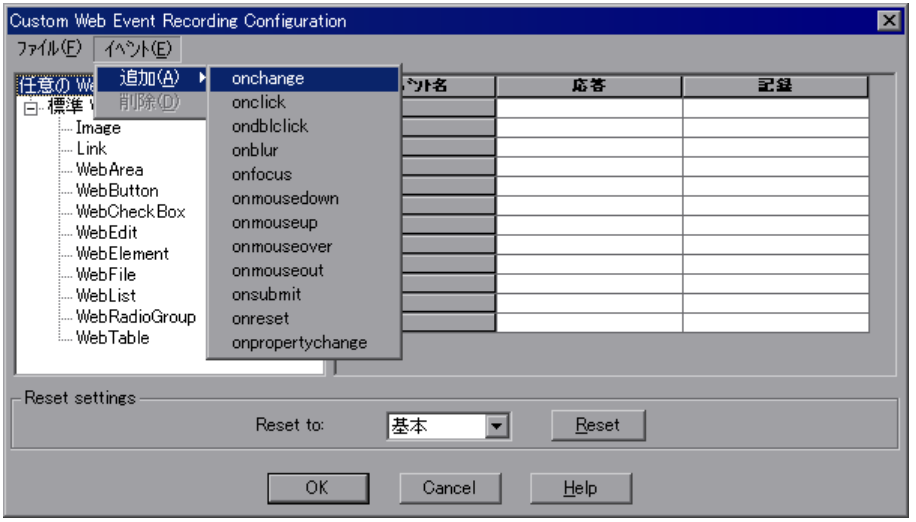
オブジェクトのリッスン・イベントの追加と削除

VuGen がオブジェクトへリッスンするトリガとなるイベントのリストを変更できます。

オブジェクトにリッスン・イベントを追加するには，次の手順を実行します。

1 ［ユーザ定義 Web イベント記録設定］ダイアログ・ボックスから，イベントを追加するオブジェクト（組み込み Web オブジェクトのいずれか，または HTML タグ・オブジェクト）を選択します。

2 ［**イベント**］＞［**追加**］を選択します。使用できるイベントのリストが開きます。



3 追加するイベントを選択します。イベントは［イベント名］列にアルファベット順で表示されます。標準設定では，VuGen はハンドラが付加されているイベントをリッスンし，そのイベントを（それが何らかのレベルでリッスンされている限り）必ず記録します。

リッスン設定および記録設定の詳細については，次の「イベントのリッスン設定と記録設定の変更」を参照してください。

オブジェクトからリッスン・イベントを削除するには，次の手順を実行します。

1 ［ユーザ定義 Web イベント記録設定］ダイアログ・ボックスの左側の表示枠から，イベントを削除するオブジェクトを選択します。

2 削除するイベントを［**イベント名**］カラムから選択します。

3 ［**イベント**］＞［**削除**］を選択します。［**イベント名**］カラムからイベントが削除されます。

イベントのリッスン設定と記録設定の変更

オブジェクトごとにリスト表示される各イベントについて，リッスン条件を選択し，記録するかどうかを設定できます。

注：リッスンと記録の設定は相互に独立です。つまり，オブジェクトに対するイベントをリッスンしても，それを記録しないことや，オブジェクトに対するイベントをリッスンせずに，そのイベントを記録することができます。詳細については，778 ページ「イベントのリッスンと記録を行うためのヒント」を参照してください。

リッスン条件

イベントごとに，VuGen がいつイベントをリッスンするか指定できます。

- ［**常に**］：オブジェクトでイベントが発生するたびにリッスンします。
- ［**ハンドラの場合**］：イベントにイベント・ハンドラが関連付けられている場合にのみリッスンします。
- ［**動作の場合**］：イベントに DHTML 動作が関連付けられている場合にのみリッスンします。

➤ ［**ハンドラまたは動作の場合**］：イベントにハンドラまたは DHTML 動作が関連付けられている場合にのみリッスンします。

➤ ［**しない**］：イベントを一切リッスンしません。

イベント・**ハンドラ**は，Web ページに含まれているコードであり，通常はスクリプト言語で書かれている関数またはルーチンです。対応するイベントが発生したときに制御が渡されます。

DHTML **動作**は，ページにおける特定の機能や振る舞いをカプセル化します。ページ上の標準的な HTML 要素に適用されている場合，その要素の標準設定の動作が拡張されます。

イベントに対するリッスン条件を指定するには，次の手順を実行します。

1 ［ユーザ定義 Web イベント記録設定］ダイアログ・ボックスから，リッスン条件を変更する対象となるオブジェクトを選択します。

2 変更するイベントの行で，［**応答**］列から必要なリッスン条件を選択します。

リストからリッスン条件を選択します。［**常に**］，［**ハンドラの場合**］，［**動作の場合**］，［**ハンドラまたは動作の場合**］，または［**しない**］のいずれかを選択できます。

記録ステータス

イベントごとに，対象イベントを記録するようにも，記録しないようにも，あるいは次のイベントが選択されたイベントに依存している場合だけ記録するようにも設定できます。

➤ ［**有効**］：VuGen が対象オブジェクトあるいはイベントの「バブリング先」である別のオブジェクトをリッスンしている場合，イベントが生じるたびにそれを記録します。

「バブリング」とは，子オブジェクトで発生したイベントが，イベントを処理するイベント・ハンドラに遭遇するまで，HTML コード内の階層をさかのぼる処理です。

- ［**無効**］：指定されたイベントを記録せず，イベント・バブリングがある場合はそれを無視します。
- ［**次のイベントで有効**］：（Image オブジェクトおよび WebArea オブジェクトの場合にのみ該当）［**有効**］との唯一の違いは，以降のイベントが同じオブジェクトで発生した場合にのみイベントを記録することです。たとえば，マウスオーバ操作によって画像リンクが変わるとします。この画像の上をマウスが通過するたびに，mouseover イベントを記録する必要はないかもしれません。ただし，リンクが mouseover イベント後に表示される画像によってのみ有効になるので，mouseover イベントを，同じオブジェクトに対するクリック・イベントの前に記録することが重要となります。

イベントの記録ステータスを設定するには，次の手順を実行します。

1 ［ユーザ定義 Web イベント記録設定］ダイアログ・ボックスから，記録ステータスを変更するオブジェクトを選択します。任意の Web オブジェクトを選択し，記録対象ページのすべての Web オブジェクトの記録ステータスを設定します。

2 変更対象のイベントの行で，記録ステータスを［記録］列から選択します。

イベントのリッスンと記録を行うためのヒント

理想的なリッスンと記録の設定を見つけるのが困難な場合があります。これらの設定を行うときには，次のガイドラインに留意します。

- オブジェクトのイベントを記録するには，VuGen がイベントをリッスンし，イベントが生じたときにそれを記録するように指定します。子プロジェクトのイベントは，親オブジェクトにそのイベントに対するハンドラまたは動作が含まれている場合にもリッスンできます。また，親オブジェクトのイベントは，子オブジェクトにそのイベントに対するハンドラまたは動作が含まれていても，リッスンできます。

  ただし，ソース・オブジェクト（どの親オブジェクトがハンドラまたは動作を含んでいるかによらずイベントが実際に発生したオブジェクト）に対するイベントは記録しなければなりません。

  たとえば，**onmouseover** イベント・ハンドラのあるテーブル・セルに 2 つのイメージが含まれているとします。ユーザがマウス・ポインタでどちらかのイメージに触れると，バブリングによってイベントがそのセルに送られます。このバブリングにはどちらのイメージが実際に触れられたかの情報が含まれています。この mouseover イベントは，次のようにして記録できます。

  - WebTable の mouseover イベントに対する**応答**を［**ハンドラの場合**］に設定する一方で（イベントが生じたときに VuGen にイベントが「聞こえる」ように），イベントは記録しないようにした上で，Image の mouseover イベントに対する**応答**は［**しない**］に設定する一方で，Image に対する記録は［**有効**］（WebTable レベルでリッスンした後の画像に対する mouseover イベントを記録します）に設定します。
  - Image の mouseover イベントに対する**応答**を［**常に**］（イメージ・タグが動作またはハンドラを含んでいなくても mouseover イベントをリッスン）に設定し，Image オブジェクトに対する記録ステータスを［**有効**］（イメージに対する mouseover イベントを記録）に設定します。

- 多数のオブジェクト上の多数のイベントをリッスンするように設定すると，VuGen のパフォーマンスが低下することがあるので，リッスンの設定は，必要なオブジェクトに限定するようにします。
- まれに，イベントが発生したオブジェクト（ソース・オブジェクト）をリッスンしていると，イベントが妨害されることがあります。

イベント記録設定のリセット

ユーザ定義の設定を行った後で標準の設定に戻すには，標準の Web イベント設定を使用するように VuGen を設定します。

注：標準設定をリセットすると，ユーザ定義の設定は完全になくなります。

基本の設定レベルに戻すには，次の手順を実行します。

1 ［記録オプション］で［**GUI プロパティ：Web イベント設定**］ノードを選択します。

2 ［**標準設定値を使用**］をクリックします。標準設定スライダが再度表示され，すべてのイベント設定が**基本**イベント記録設定レベルに戻されます。

また，特定の（基礎の）ユーザ定義設定，［基本］，［中］，または［高］を復元することもできます。

設定を特定のユーザ定義レベルにリセットするには，次の手順を実行します。

1 ［記録オプション］で［**GUI プロパティ：Web イベント設定**］ノードを選択します。

2 ［**ユーザ定義設定**］ボタンをクリックします。［ユーザ定義 Web イベント記録設定］ダイアログ・ボックスが開きます。

3 ［**リセット値**］ボックスで，使用する定義済みイベント記録レベルを選択します。

4 ［**リセット**］をクリックします。すべてのイベント設定が，選択したレベルの標準設定に戻されます。

Web（Click and Script）仮想ユーザで作業をする際のヒント

本項では，Web（Click and Script）ユーザに関する一般的な問題，解決策，およびヒントについて取り上げます。

- 記録に関する問題
- 記録に関するヒント
- 再生に関する問題
- 再生に関するヒント
- その他の問題

- その他のヒント
- Web（Click and Script）仮想ユーザ・スクリプトの拡張

記録に関する問題

次の項に，記録に関して最もよく起こる問題をリストアップします。

アプリケーションの記録時の動作が記録していないときと異なる

アプリケーションの動作が，記録をしているときとしていないときで異なる場合，その記録の問題が Web（Click and Script）に固有の問題かどうかを確認する必要があります。たとえば，Web ページが読み込まれない，内容の一部が欠落している，ポップアップ・ウィンドウが表示されないなどの症状があります。

新しい Web（HTTP/HTML）スクリプトを作成して，記録をしなおします。

Web（HTTP/HTML）にして記録を問題なく行えたら，ソケット・レベルの記録を無効にすることをお勧めします（783 ページ「ソケット・レベルの記録を無効にする」を参照）。この問題は，イベント・リスナーが原因の可能性があります。［**Web イベント設定**］記録オプションの設定を色々変えてみて該当するイベント・リスナーを無効にして，セッションを Web（Click and Script）ユーザとして記録しなおします。

イベント・リスナーを無効にするには，次の手順を実行します。

- ［記録オプション］ダイアログ・ボックスを開きます。［**ツール**］＞［**記録オプション**］を選び，［**GUI プロパティ：Web イベント設定**］ノードを選択します。
- ［**ユーザ定義設定**］をクリックして［**Web オブジェクト**］ノードを展開します。オブジェクトを選択します。
- ［**記録**］カラムのリストの中で，該当する Web オブジェクトについて［**無効**］を選択します。それでも記録が正常に行われない場合は，無効にしたリスナーを有効にして，別のリスナーを無効にしてみます。記録が正常に行われるまで，これらの手順を繰り返します。

動的メニューの操作が記録されない

動的メニューとは，選択する場所に応じて動的に変化するメニューのことです。動的メニューの操作が記録されなかった場合は，イベント設定モードを［高］にして再度記録してみてください。

設定レベルを［高］に設定するには，次の手順を実行します。

➤ ［記録オプション］ダイアログ・ボックスを開きます。［**ツール**］＞［**記録オプション**］を選び，［**GUI プロパティ：Web イベント設定**］ノードを選択します。

➤ スライダを［**高**］の位置まで移動します。

一部のユーザ・アクションが記録されない

ブラウザ内で Java アプレットが動作していないか確認してください。動作していない場合は，Web（HTTP/HTML）プロトコルを利用してスクリプトを記録してください。

記録に関するヒント

キーボードではなく，マウスを使用する

オブジェクトをマウスでクリックするほうが，キーボードを使用するよりも望ましいです。記録中は，ブラウザの表示枠内にある GUI オブジェクトだけを使用してください。ブラウザのアイコン，コントロール，［停止］ボタン，または［**表示**］＞［**最新の情報に更新**］などのメニュー項目は使用しないでください。ただし，［更新］，［ホーム］，［戻る］，［進む］の各ボタンおよびアドレス・バーは使用できます。

既存のスクリプトに上書きして記録しない

既存のスクリプトではなく，新規に作成したスクリプトに記録するのが最善です。

ショートカット・メニューを使用しない

記録中はショートカット・メニューの使用を避けます。ショートカット・メニューとは，右クリック・メニューなど，グラフィカル・ユーザ・インタフェースの項目をクリックしたときに現れるメニューのことです。

記録時にブラウザで作業をしない

記録時には，VuGen によって開かれたブラウザ・ウィンドウ以外のブラウザ・ウィンドウでは作業をしないようにします。

ダウンロードは待機する

すべてのダウンロードが完了するまで待ってから，ボタンをクリックしたり，テキスト・フィールドに入力を行ったりするなどのアクションを起こすようにします。

ページの読み込みを待機する

記録時には，ページの読み込みが完全に終わるのを待ってから次のステップを実行するのが最善です。ページ全体の読み込みが完了するまで待たなかった場合は，スクリプトを記録しなおしてください。

開始ページに移動する

アクションの最後のページに，反復の最初に利用可能であったリンクやボタンが含まれていない場合，次の反復は失敗します。たとえば，最初のページに「**Book A Flight**」というテキストリンクがある場合，ビジネス・プロセスの最後に同じリンクが表示されているように，記録の最後に，適切なページに移動するようにします。

イベント設定レベルは高くする

［**高**］のイベント設定レベルを使用してビジネス・プロセスを記録しなおします。イベント設定レベルの変更の詳細については，781 ページ「動的メニューの操作が記録されない」を参照してください。

ソケット・レベルの記録を無効にする

ソケット・レベルのメッセージをキャプチャすると，アプリケーションが中断することがあります。ほとんどの記録では，ソケット・レベル・データは必要ありません。ソケット・レベル・データが記録されないようにするには，記録オプションでこのオプションを無効にします。詳細については，766 ページ「記録設定」を参照してください。

［描画関連のプロパティ値の記録］を有効にする

アプリケーションのクライアント側のスクリプトがスタイル設定アクティビティを多用する場合，スクリプトの記録を開始する前に［**描画関連のプロパティ値の記録**］を有効にします。たとえば，**offsetTop** などの追加 DOM オブジェクトを記録するには，このオプションを有効にします。このオプションを有効にすると，再生速度が低下することがあります

［描画関連のプロパティ値の記録］を有効にするには，次の手順を実行します。

➤ ［記録オプション］ダイアログ・ボックスを開きます。［**ツール**］＞［**記録オプション**］を選び，［**GUI プロパティ：詳細**］ノードを選択します。

再生に関する問題

GUI オブジェクトが見つからない

問題のステップは，2 回目の反復の先頭で起きますか？

このエラーが 2 回目の反復の先頭で起きる場合，その原因はおそらく最初の反復のときに存在していた開始ページが，2 回目の反復のときに存在しなかったことです。アクションの最後のページに，反復の最初に利用可能であったリンクやボタンが含まれていない場合，次の反復は失敗します。たとえば，最初のページに「**Book A Flight**」というテキストリンクがある場合，ビジネス・プロセスの最後に同じリンクが表示されているように，適切なページに移動するようにします。

非 ASCII 文字を含んだテキスト・リンクですか？

非 ASCII 文字のときにこの問題が生じる場合，VuGen に対してデータを UTF-8 形式との間で変換を行うように設定します。

UTF-8 変換を有効にするには，次に手順を実行します。

➤ ［記録オプション］ダイアログ・ボックスを開きます。［**仮想ユーザ**］＞［**実行環境の設定**］を選び，［**インターネット プロトコル：プリファレンス**］ノードを選択します。

➤ ［**オプション**］をクリックして［詳細オプション］ダイアログ・ボックスを開きます。

➤ ［**UTF-8 から、または UTF-8 への変換**］オプションを探して，それを［**Yes**］に設定します。

あるいは，リンクが見つからなかった場合に表示される代替候補のリストを確認します。表示されている ¥xA0 のような 16 進表現のエスケープ・シーケンスや非標準の形式のテキストをそのまま入力します。

アプリケーションの中で当該アクション・シーケンスを 2 回実行することはできますか？

データベースから特定のユーザを削除するなど，プロセスによっては実行できるのは一度だけというものもあります。この場合，1 回目の反復以降，アクションはもはや有効ではなくなるので，再生は失敗します。対象ビジネス・プロセスを，記録しなおさずに，同じデータを使ってアプリケーションの中で 2 回以上繰り返して実行できることを確認します。

画像の「Id」，「Name」，「Alt」の各プロパティは空でしたか？

ツリー・ビューの中で，直前の画像ステップをダブル・クリックしてプロパティを表示します。「**Id**」，「**Name**」，「**Alt**」の各プロパティが空の場合，「**Src**」プロパティにファイル名を指定するなど，画像を識別するための他の情報を指定します。

あるいは，［**Ordinal**］引数を追加して，ページ内での画像の出現番号を指定します。［**Ordinal**］引数は，他の識別引数で一意に識別できない場合に，ページ上の各画像を一意に識別します。詳細については，オンライン関数リファレンス（［**ヘルプ**］ > ［**関数リファレンス**］）を参照してください。

ステップの記述が変わりましたか？

［出力］ウィンドウの［再生ログ］を確認して，問題のステップのオブジェクト一覧を探します。場合によっては，オブジェクト記述が実行ごとに変わっていることがあります。

解決の方法はいくつかあります。

- 新しい値が安定している場合には，［スクリプト ビュー］を開いて，ステップの DESCRIPTION 引数の値を手作業で変更します。
- 記述が実行のたびに変わる場合は，DESCRIPTION 引数の中で正規表現を使用します。詳細については，オンライン関数リファレンス（［**ヘルプ**］ > ［**関数リファレンス**］）を参照してください。
- あるいは，Name など，問題となっているオブジェクト記述プロパティを，Ordinal プロパティに置き換えます。詳細については，オンライン関数リファレンス（［**ヘルプ**］ > ［**関数リファレンス**］）を参照してください。

記録時にページの読み込みは完了しましたか？

記録時には，ページの読み込みが完全に終わるのを待ってから次のステップを実行するのが最善です。ページ全体の読み込みが完了するまで待たなかった場合は，スクリプトを記録しなおしてください。

再生に関するヒント

問題のトラブルシューティングに次のヒントを役立ててください。

並べ替えをしない

記録されたスクリプト内のステートメントは並べ替えないようにします。また，ある Action から別の Action へコードのセグメントをコピーすることも推奨されません。

非 ASCII 文字は変換する

リンクに非 ASCII 文字が含まれている場合，VuGen に対してデータを UTF-8 形式との間で変換を行うように設定します。

UTF-8 変換を有効にするには，次の手順を実行します。

➤ ［記録オプション］ダイアログ・ボックスを開きます。［**仮想ユーザ**］＞［**実行環境の設定**］を選び，［**インターネット プロトコル：プリファレンス**］ノードを選択します。

➤ ［**オプション**］をクリックして［詳細オプション］ダイアログ・ボックスを開きます。

➤ ［**UTF-8 から、または UTF-8 への変換**］オプションを探して，それを［**Yes**］に設定します。

あるいは，リンクが見つからなかった場合に表示される代替候補のリストを確認します。表示されている ¥xA0 のような 16 進表現のエスケープ・シーケンスや非標準の形式のテキストをそのまま入力します。

同じアクション・シーケンスを 2 回実行してみる

データベースから特定のユーザを削除するなど，プロセスによっては実行できるのは一度だけというものもあります。この場合，1 回目の反復以降，アクションはもはや有効ではなくなるので，再生は失敗します。対象ビジネス・プロセスを，記録しなおさずに，同じデータを使ってアプリケーションの中で 2 回以上繰り返して実行できることを確認します。

一意の画像プロパティを設定するようにする

ツリー・ビューの中で，直前の画像ステップをダブル・クリックしてプロパティを表示します。「**Id**」，「**Name**」，「**Alt**」の各プロパティが空の場合，「**Src**」

プロパティにファイル名を指定するなど，画像を識別するための他の情報を指定します。

あるいは，［**Ordinal**］引数を追加して，ページ内での画像の出現番号を指定します。［**Ordinal**］引数は，他の識別引数で一意に識別できない場合に，ページ上の各画像を一意に識別します。詳細については，オンライン関数リファレンス（［**ヘルプ**］＞［**関数リファレンス**］）を参照してください。

ステップの記述を確認する

「**GUI Object is not found**」というエラーが生じる場合，［出力］ウィンドウの［再生ログ］を確認して，問題のステップのオブジェクト一覧を探します。場合によっては，オブジェクト記述が実行ごとに変わっていることがあります。

解決の方法はいくつかあります。

- 新しい値が安定している場合には，［スクリプト ビュー］を開いて，ステップの DESCRIPTION 引数の値を手作業で変更します。
- 記述が実行のたびに変わる場合は，DESCRIPTION 引数の中で正規表現を使用します。詳細については，オンライン関数リファレンス（［**ヘルプ**］＞［**関数リファレンス**］）を参照してください。
- あるいは，Name など，問題となっているオブジェクト記述プロパティを，Ordinal プロパティに置き換えます。詳細については，オンライン関数リファレンス（［**ヘルプ**］＞［**関数リファレンス**］）を参照してください。

その他の問題

JavaScript でメモリ不足のエラー

実行環境の設定で JavaScript 用のメモリを増やします。

JavaScript 用のメモリ容量を増やすには，次の手順を実行します。

- ［記録オプション］ダイアログ・ボックスを開きます。［**仮想ユーザ**］＞［**実行環境の設定**］を選び，［**インターネット プロトコル：プリファレンス**］ノードを選択します。
- ［**オプション**］をクリックして［詳細オプション］ダイアログ・ボックスを開きます。

➤ ［**メモリ管理 : JavaScript ランタイム メモリのサイズ (KB)**］オプションと［**メモリ管理 : JavaScript スタック メモリのサイズ (KB)**］オプションを探します。

➤ メモリ・サイズを 512 以上の値に増やします。

VuGen に JavaScript エラーが表示される

VuGen の［再生ログ］に JavaScript エラーが表示される場合，Internet Explorer のスクリプト・エラーを有効にすることで，JavaScript のコードそのものにエラーがないことを確認します。

スクリプトのエラーを表示させるには，次の手順を実行します。

➤ Internet Explorer（IE）を開きます。［**ツール**］＞［**インターネット オプション**］を選択し，［**詳細設定**］タブを選択します。

➤ ［**ブラウズ**］セクション内の［**スクリプト エラーごとに通知を表示する**］を有効にします。

➤ アプリケーションを IE の中で再度実行します。IE にスクリプト・エラーが表示された場合，JavaScript アプリケーションに問題があることになります。アプリケーションを修正することが不可能な場合には，該当する再生エラーは無視することができます。

パラメータ化後に生じる問題

値をパラメータ化した後に問題に遭遇した場合は，値がアプリケーションに対して有効であることを確認してください。パラメータに使用されている値を使用してビジネス・プロセスを実行し，アプリケーションがその値を受け入れることを確認します。

スタイル設定アクションを利用するアプリケーションに関する問題

アプリケーションのクライアント側のスクリプトがスタイル設定アクティビティを多用する場合，［**描画関連のプロパティ値の記録**］を有効にしてから，スクリプトを記録しなおすべきです。これにより，追加の DOM オブジェクトの記録が可能となります。

［描画関連のプロパティ値の記録］を有効にするには，次の手順を実行します。

➤ ［記録オプション］ダイアログ・ボックスを開きます。［**ツール**］＞［**記録オプション**］を選び，［**GUI プロパティ：詳細**］ノードを選択します。

➤ ［**描画関連のプロパティ値の記録**］を有効にします。スクリプトを記録しなおします。

その他のヒント

問題のトラブルシューティングに次の追加のヒントを役立ててください。

警告を検索する

［再生ログ］の中で警告などを検索します。

応答を確認する

Web_reg_find を使用して，直前のステップの応答が正しいことを確認してください。詳細については，オンライン関数リファレンス（［**ヘルプ**］＞［**関数リファレンス**］）を参照してください。

代替ナビゲーションを使用する

問題が生じているステップや Java アプレットを使用しているステップでは，「**代替ナビゲーション**」を使用して Web（Click and Script）ステップを HTTP レベルのステップで置き換えます。HTTP レベルのステップでは，手作業による相関が必要になる場合があります。代替ナビゲーションを行うには，ツリー・ビューでステップを選択するか，スクリプト・ビューでテキストを選択し，右クリック・メニューから［**代替ナビゲーションに置き換え**］を選択します。

Web（Click and Script）仮想ユーザ・スクリプトの拡張

次の項では，スクリプトの作成に役立ついくつかの拡張機能について説明します。

以降に示す機能の大部分は API 関数に対する拡張機能です。関数とその引数の詳細については，**オンライン関数リファレンス**（［**ヘルプ**］＞［**関数リファレンス**］）を参照するか，任意の関数で F1 キーを押します。

条件ステップの追加

web_xxxx という形式の名前の Web（Click and Script）関数群では，再生時の条件アクションを指定できます。たとえば，要素を調べ，要素が見つかったときにのみアクションを実行する必要がある場合などに条件が役立ちます。

たとえば，インターネット検索を実行し，［**次へ**］をクリックしてすべての結果ページに移動したいとします。結果ページが何ページになるかわからないので，次のページがあることを示す［**次へ**］ボタンがあるかどうかを，ステップを失敗させずに調べる必要があります。次のコードでは，通知を伴う検証ステップを追加しています。［**次へ**］ボタンが見つかった場合は，そのボタンをクリックします。

```
While (web_text_link("Next",
DESCRIPTION,
    "Text=Next",
    VERIFICATION,
    "NotFound=Notify",
    ACTION,
    "UserAction=Click",
    LAST) == LR_PASS);
```

VERIFICATION セクションの構文と使用方法の詳細については，**オンライン関数リファレンス**（［**ヘルプ**］＞［**関数リファレンス**］）を参照してください。

ページ・タイトルの確認

web_browser ステップでは，タイトル検証記録オプションを使用して，ページ・タイトルの有無を調べることができます。仮想ユーザが，ステップごと，あるいは新しい最上位ウィンドウにナビゲーションするごとに，自動的にこのチェックを実行するようにできます。

さらに，完全一致検索および正規表現検索の両方を使用して，スクリプトの任意の位置にタイトル検証を手作業で追加することもできます。

```
web_browser("test_step",
DESCRIPTION,
...
VERIFICATION,
  "BrowserTitle=Title",
  ACTION,]
,
LAST);
```

詳細については，**オンライン関数リファレンス**（[**ヘルプ**] > [**関数リファレンス**]）を参照してください。

タイトル検証オプションは，記録オプション内で直接設定できます。詳細については，766 ページ「GUI プロパティの詳細設定」を参照してください。

テキスト・チェック検証

テキスト・チェックポイントを使用すれば，テキスト文字列が Web ページまたはアプリケーションの適切な場所に表示されているかどうかを検証し，その結果に基づいてアクションを実行できます。完全一致検索または正規表現検索を使用して，テキスト文字列が存在すること（**ContainsText**），あるいは，テキスト文字列が存在しないこと（**DoesNotContainText**）を確認できます。

たとえば，Web ページに「Flight departing from New York to San Francisco」という文が表示されるとします。「New York」という単語が「Flight departing from」と「to San Francisco」の間に表示されることを検査するテキスト・チェックポイントを作成できます（この例では，正規表現条件を使用する必要があります）。

これらのチェックポイントを実装するには，ステップの VERIFICATION セクションにテキスト・チェック関連の引数を追加します。仮想ユーザは，再生時に，ブラウザの HTML ドキュメントと子フレームの innerText を検索します。**NotFound** 引数は，オブジェクトが見つからなかったため，あるいはテキスト検証が失敗したために検証が失敗した場合に行うアクションを指定します（Error，Warning，または Notify）。

テキスト検証は，スクリプトの既存ステップに手作業で追加できます。テキスト検証は，要素を生成したステップの後に置くようにします。

テキスト検証の引数は次の Action 関数で有効です：**web_browser**，**web_element**，**web_list**，**web_text_link**，**web_table**，**web_text_area**。

注：同じタイプのテキスト検証は，ステップごとに 1 回だけ使用できます（たとえば，**ContainsText** を 2 回使用することはできません）。複数のテキストについて検証する必要がある場合は，検証を複数のステップに分けます。ただし，同じステップの中でも異なる検証であれば同時に使用できます（たとえば，**ContainsText** と **DoesNotContainText**）。この場合，ステップが成功するためには，すべての条件が満たされる必要があります。

次の例では，mercury.com からフランスの Web サイト mercury.com/fr に誘導されなかったかどうかを検証引数によって確認しています。

```
web_browser("www.mercury.com",
            ACTION,
            "Navigate=http://www.mercury.com/",
        LAST);

web_browser("Verify",
            VERIFICATION,
            "ContainsText=Go to Mercury France",
            "DoesNotContainText=Mercury.com in English",
        LAST);
```

パラメータへの JavaScript 値の保存

EvalJavaScript 引数は，Web ページの JavaScript を評価すること可能にします。

たとえば，ページ・タイトルと同じ名前のリンクをクリックしたいとします。次の例では，ドキュメントのタイトルを評価し，そのタイトルを次の **web_text_link** 関数で使用しています。

```
web_browser("GetTitle",
        ACTION,
        "EvalJavaScript=document.title;",
        "EvalJavaScriptResultParam=title",
        LAST);

web_text_link("Link",
        DESCRIPTION,
        "Text={title}",
        LAST);
```

ユーザ定義の記述を使った作業

何らかのグループに属するリンクを無作為にクリックしたいとします。たとえば，**mercury.com** で国を無作為に選択したいとしましょう。この種の操作は，通常の記述の照合ではできません。しかし，ユーザ定義記述引数を使用すれば，グループのすべてのリンクに共通の属性を使用してグループを特定できます。

DESCRIPTION 引数を使用して，対象要素に対して事前定義されていない属性も含め，要素の属性を指定します。仮想ユーザは，再生時に，DESCRIPTION セクションに指定されている属性を検索します。再生は，DESCRIPTION セクションの未知の引数について失敗することはありません。

たとえば，次のハイパーリンクを探したいとします。

Yahoo。この場合次のコードを使用します。

```
web_text_link("yahoo",
      DESCRIPTION,
      "Text=yahoo",
      "my_attribute=bar",
      LAST);
```

次の例では，関係するすべてのリンクが **newmerc-left-ct** という同じクラス名を持っているため，次のコードによって無作為にリンクをクリックできます。

```
web_text_link("Click",
                  DESCRIPTION,
                  "Class=newmerc-left-ct",
                  "Ordinal=random",
                  LAST);
```

次の関数では DESCRIPTION 引数はサポートされません。
web_browser，**web_map_area**，**web_radio_group**，**web_reg_dialog**。

第 50 章

Web 仮想ユーザ関数の使用

VuGen を使用して，Web サイトでのユーザ・アクションが記述された Web 仮想ユーザ・スクリプトを作成できます。各スクリプトには，実行された各アクションに直接対応する関数が含まれます。

本章では，次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（Click and Script）仮想ユーザ関数
- Web（HTTP/HTML）関数一覧
- キャッシュに格納された値の使用

以降の情報は，Web（Click and Script），Web（HTTP/HTML），Oracle Web Applications 11i，PeopleSoft Enterprise，およびワイヤレス仮想ユーザを対象とします。

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。

Web 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・スクリプトのアイコン・ベースの表現。これが標準設定のビューであり，「**ツリー・ビュー**」と呼びます（Wap 仮想ユーザでは利用できません）。
- 仮想ユーザ・スクリプトのテキスト・ベースの表現。これは，「**スクリプト・ビュー**」と呼びます。

詳細については，18 ページ「仮想ユーザ・スクリプトの表示と変更」を参照してください。

関数の追加と編集

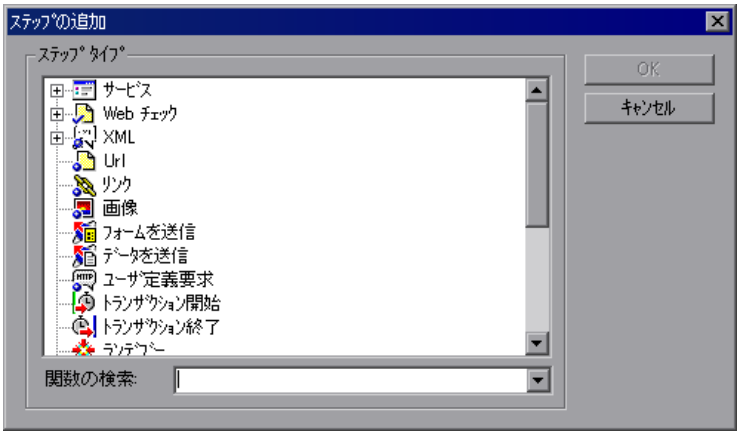
多くの Web 仮想ユーザ関数はブラウザまたはツールキットのセッション中に記録されます。

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 8 章「仮想ユーザ・スクリプトの拡張」を参照してください。

この項では，記録中と記録後に Web 仮想ユーザ関数を追加，編集する方法を，ツリー・ビューとスクリプト・ビューのそれぞれについて説明します。

仮想ユーザ・スクリプトに新しい関数を追加するには，次の手順を実行します。

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（Click and Script）仮想ユーザ関数

Web（Click and Script）仮想ユーザに固有の関数のリストを次に示します。

関数名	説明
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をエミュレートします。
web_check_box	チェック・ボックスを選択または選択解除します。
web_edit_field	テキストとパスワード入力タイプのためのデータを入力します。
web_element	HTML タグに定義されている要素をユーザがクリックする操作をエミュレートします。
web_file	ファイル入力タイプのパスを入力します。
web_image_link	ハイパーテキスト・リンクになっている画像をクリックするユーザの操作をエミュレートします。
web_image_submit	送信要求を起動する画像をクリックするユーザの操作をエミュレートします。
web_list	リスト・コントロールから項目を選択します。
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 呼び出しに対するユーザ応答を登録します。

関数名	説明
web_static_image	静的な画像をユーザがクリックする操作をエミュレートします。
web_table	テーブルに対するユーザ・アクション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	ユーザがハイパーリンク・テキストをクリックする操作をエミュレートします。

API 全般に関する注意事項

本項では Web（Click and Script）関数の全般的な注意事項について説明します。等号の前に「/RE」を付けてテキスト文字列の先頭に置くことで，ほとんどのオブジェクト記述に対して正規表現を指定できます。詳細については，オンライン関数リファレンス（[**ヘルプ**] > [**関数リファレンス**]）を参照してください。次に例を示します。

```
web_text_link("Manage Assets",
        DESCRIPTION,
        "Text/RE=(Manage Assets)|(Configure Assets)",
            LAST);
```

序数

Ordinal 属性は，同じ引数値が複数登場する場合に，それぞれを識別するために使用される，1 から始まるインデックス番号です。次の例では，記録されている 2 つの **web_text_link** 関数の引数がまったく同じです。序数だけが異なります。序数値の 2 は，2 番目の出現であることを示します。

```
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);
```

空文字列

引数を指定しないことと，引数を空の文字列として指定することとは，別の意味になります。引数を指定しないときは，標準設定値が使用されるか，または引数が無視されます。引数を列挙したものの，値として空の文字列を割り当てたときは，空の文字列に一致するものか，文字列がまったくないものとして一致するものが検索されます。たとえば，**id** 引数を省略すると，HTML 要素の **id** プロパティを無視するよう VuGen に指示することになります。"ID=" と指定した場合は，**id** プロパティを持たない HTML 要素か，空の ID を持つ HTML 要素が検索されます。

```
web_text_link("Manage Assets_2",
        DESCRIPTION,
        "Text=Manage Assets",
        "Id=",
        "FrameName=main",
        LAST);
```

Web（HTTP/HTML）関数一覧

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 関数は次のように分類されます。

- アクション関数
- 認証関数
- キャッシュ関数
- チェック関数
- 接続定義関数
- 同時実行グループ関数
- クッキー関数
- 相関関数
- フィルタ関数
- ヘッダー関数
- プロキシ・サーバ関数
- 再生関数
- その他の関数
- 制御型関数

Web（Click and Script）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，798 ページ「Web（Click and Script）仮想ユーザ関数」を参照してください。

アクション関数でない関数のほとんどは，Web（Click and Script）仮想ユーザ・スクリプトの中で使用できます。**web_concurrent_start** および **web_concurrent_end** は，Web（HTTP/HTML）仮想ユーザ・スクリプトに固有のものです。

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

証明書ファイルと鍵ファイルの場所と形式に関する情報を指定します。

web_set_user

Web サーバ（Web サーバのユーザ認証エリア）に，ログオ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。

同時実行グループ関数

web_concurrent_end 同時実行グループの終わりを示します。

web_concurrent_start 同時実行グループの始まりを示します。

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_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 次のアクション関数のダウンロード基準を設定します。

web_add_auto_filter 後続のすべてのアクション関数のダウンロード基準を設定します。

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	ユーザ定義ヘッダーを以降のすべての HTTP 要求に追加します。
web_add_header	ユーザ定義ヘッダーを次の HTTP 要求に追加します。
web_cleanup_auto_headers	以降の HTTP 要求にユーザ定義ヘッダーを追加しないようにします。
web_remove_auto_header	以降の HTTP 要求に特定のヘッダーを追加しないようにします。
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・サーバにリダイレクトするように指示します。
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	直前の HTTP 要求に関する特定の情報を返します。
web_report_data_point	データ・ポイントを指定して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，次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 8 章「仮想ユーザ・スクリプトの拡張」を参照してください。

次のステップの種類が 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**］ステップは，ブラウザに URL を入力したりブックマークを使用したりして，特定の Web ページにアクセスすると生成されます。各［URL］アイコンは，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**］ステップが追加されます。各［リンク］ステップは，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，このステップの標準のラベルとなります。
画像	記録中にハイパーグラフィック・リンクをクリックすると，［**画像**］ステップが仮想ユーザ・スクリプトに追加されます。各［画像］ステップは，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。

アイコンの種類	説明
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，［**ユーザ定義要求**］ステップが仮想ユーザ・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。

キャッシュに格納された値の使用

ブラウザのキャッシュにデータを保存しておいて，スクリプト内の後の時点で読み込むことができます。

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，スクリプトを実行する **VuserName** パラメータごとに **C:¥temp** にキャッシュ・ファイルを生成します。

```
web_dump_cache("paycheckcache","FileName=c:¥¥temp¥¥{Vuser
    Name}paycheck", "Replace=yes", LAST)
```

1 個の仮想ユーザを 10 回実行すると，VuGen は次の形式で 10 個のキャッシュ・ファイルを生成します。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，次の手順を実行します。

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。

```
Actions()
{
//     web_add_cookie("storedCookieCheck=true; domain=pbntas05;
path=/");

web_load_cache("ActionLoad","FileName=c:¥¥temp¥¥{VuserName}paych
eck",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;
    }
```

```
Actions()
{
//     web_add_cookie("storedCookieCheck=true; domain=pbntas05;
path=/");

web_load_cache("ActionLoad","FileName=c:¥¥temp¥¥{VuserName}paych
eck",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 51 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（Web/WinSocket 仮想ユーザ）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

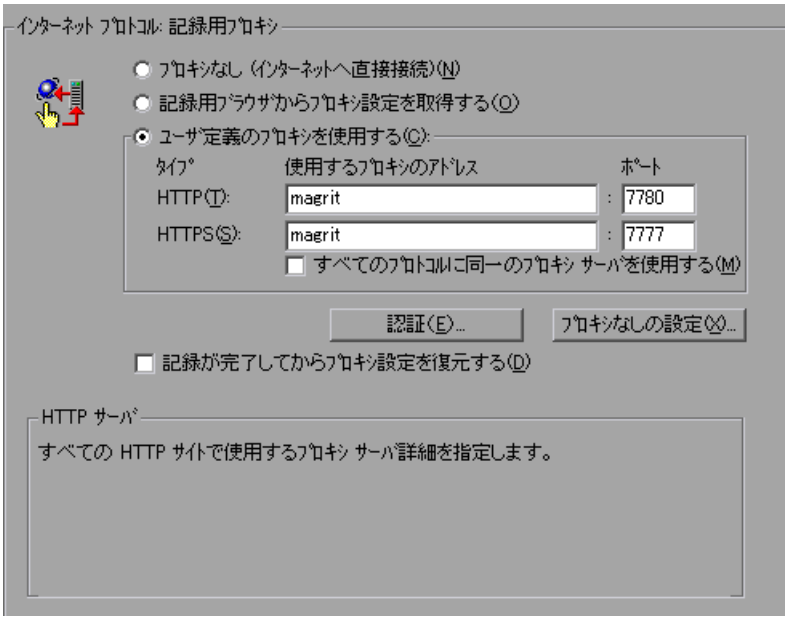
注：この項は Web/WinSocket デュアル・プロトコルにのみ適用されます。

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［**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。

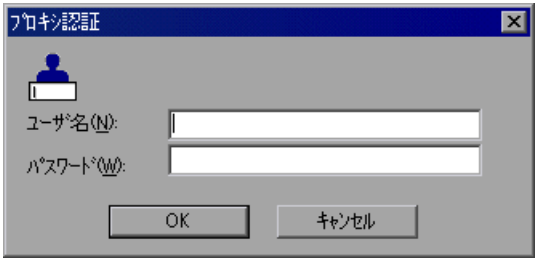


次のプロキシ・オプションのいずれかを選択できます。

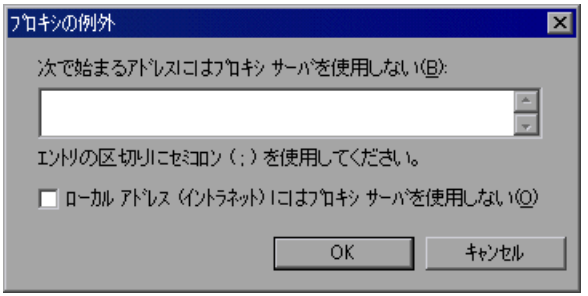
- ［**プロキシなし（インターネットへ直接接続）**］：常にインターネットへの直接接続を利用します。つまり，プロキシ・サーバを使用せずに直接接続します。通常，これは Internet Explorer の［設定を自動的に検出する］設定に対応します。
- ［**記録用ブラウザからプロキシ設定を取得する**］：記録用ブラウザのプロキシ設定を使用します。標準設定では，このオプションが選択されています。このオプションは，Web および WinSock 仮想ユーザに対しては使用できません。
- ［**ユーザ定義のプロキシを使用する**］：記録中，指定されたプロキシ・サーバを使用します。セキュアでない HTTP 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。

  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。たとえば，アプレット，ActiveX コントロール，およびマルチウィンドウ・アプリケーションの HTTP アクションを記録するときなどです。

記録オプションの詳細設定

［**HTTP プロパティ：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

［お気に入り］インターネット記録オプション

［お気に入り］オプションを使用して，思考遅延時間，コンテキストのリセット，スナップショットの保存，および **web_reg_find** 関数にかかわるコード生成の方法をカスタマイズできます。オプションのいくつかは，マルチ・プロトコル・モードでは利用できません。

- ［**思考遅延時間を記録する**］：（ワイヤレスの場合のみ）この設定は標準で有効になっており，VuGen に対して，思考遅延時間を記録し，**lr_think_time** 関数を生成するよう指示します。また，［**思考遅延時間しきい値**］を設定することもできます。実際の思考遅延時間がしきい値より短かった場合，**lr_think_time** 関数は生成されません。
- ［**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。
  - ［**サブ フレームで web_reg_find 関数を生成**］：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 Windows のユーザは，このオプションを選択することで，EUC-JP 文字エンコーディングを使用している Web サイトに対するサポートを有効にできます。この設定を有効にすると，EUC-JP 文字がマシンのロケールで使用されているロケールに変換され，VuGen に正しく表示されます。VuGen によって，すべての EUC-JP（日本語 UNIX）文字列をマシンのロケールに合わせて Shift-JIS（日本語版 Windows）エンコーディングに変換し，スクリプトに **web_sjis_to_euc_param** 関数を追加します（漢字のみ）。

記録エンジンの選択

VuGen の標準設定では，Web（HTTP/HTML）仮想ユーザに対して，シングル・プロトコルの記録を含むすべての記録でマルチプロトコルの記録エンジンが使用されます。

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，［記録オプション］の［詳細］ノードの［**以前の記録エンジンを使用してスクリプトを記録する**］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 Web（HTTP/HTML）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
- コンテント・タイプのフィルタリング
- リソース以外のコンテント・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。

たとえば，［**リスト内のヘッダを記録**］モードで「**Content-type**」のヘッダーを指定すると，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。

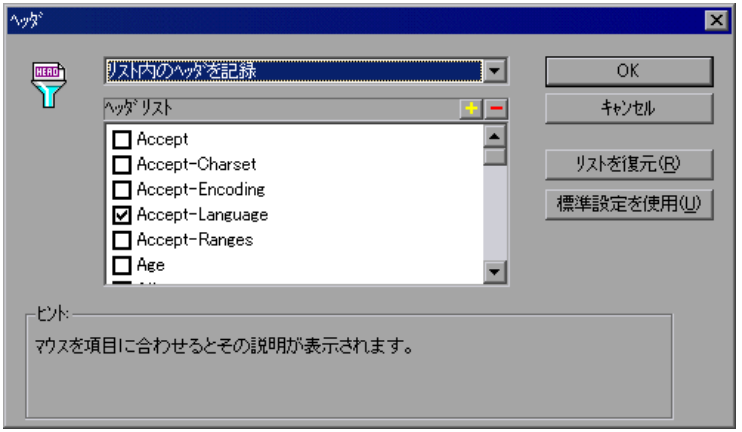
```
web_add_auto_header("Content-Type",
                    "application/x-www-form-urlencoded");
```

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，［**HTTP プロパティ：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 次のいずれかの方法を使用します。

- 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。

コンテント・タイプのフィルタリング

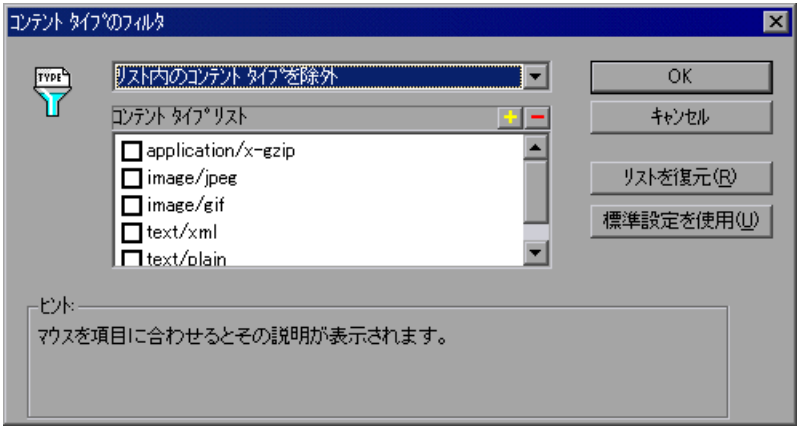
VuGen を使って，記録したスクリプトのコンテント・タイプをフィルタリングできます。記録するコンテント・タイプ，またはスクリプトから除外するコンテント・タイプを指定します。次の 3 つのモードがあります。［**コンテント タイプをフィルタにかけない**］，［**リスト内のコンテント タイプを除外**］，［**リストに定義されていないコンテント タイプを除外**］の 3 つです。最初のモードで作業すると，コンテント・タイプがフィルタリングされません。2 つ目のモードでは，選択されたコンテント・タイプだけが除外されます。［**リストに定義されていないコンテント タイプを除外**］を指定すると，チェック・マークを入れたコンテント・タイプ以外のすべてのコンテント・タイプが除外されます。標準では，フィルタは設定されていません。

たとえば，Web サイトのテキストと画像だけを対象とするには，［**リストに定義されていないコンテント 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，［**HTTP プロパティ：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。
- 特定のコンテント・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コンテント・タイプをリストから選択します。
- 特定のコンテント・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，記録するコンテント・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。

リソース以外のコンテント・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。

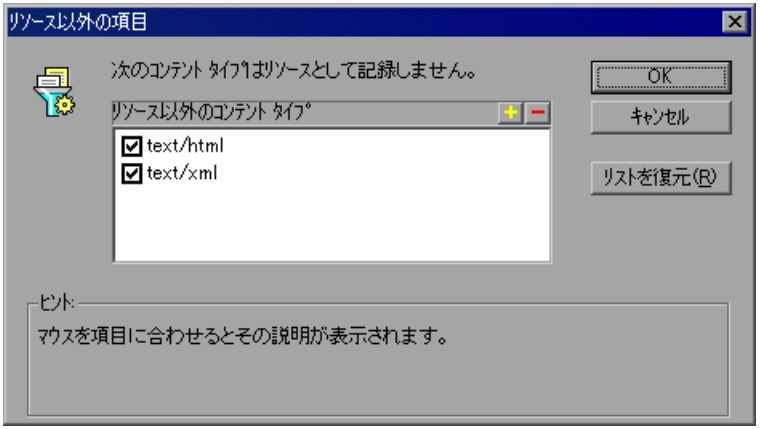
```
web_url("MercuryWebTours",
        "URL=http://localhost/MercuryWebTours/",
        "Resource=0",
        "RecContentType=text/html",
        "Referer=",
        "Snapshot=t1.inf",
        "Mode=HTML",
        LAST);
```

特定のコンテント・タイプをリソースとして処理しないように除外することができます。たとえば，**gif** タイプのリソースがリソースとして処理されないように設定し，無条件にダウンロードされるように設定できます。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，［**HTTP プロパティ：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コンテント・タイプのリストを表示します。



3 リストにコンテント・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，「ー」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 52 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
- 記録レベルの設定

以降の情報は，Web（HTTP/HTML），Web (Click and Script), Web/WinSocket，Oracle Web Applications 11i，PeopleSoft Enterprise の各仮想ユーザ・スクリプトを対象とします。

記録オプションの設定について

VuGen を使用して，ユーザが Web サイトで実行する標準的な操作を記録して，Web 仮想ユーザ・スクリプトを生成できます。

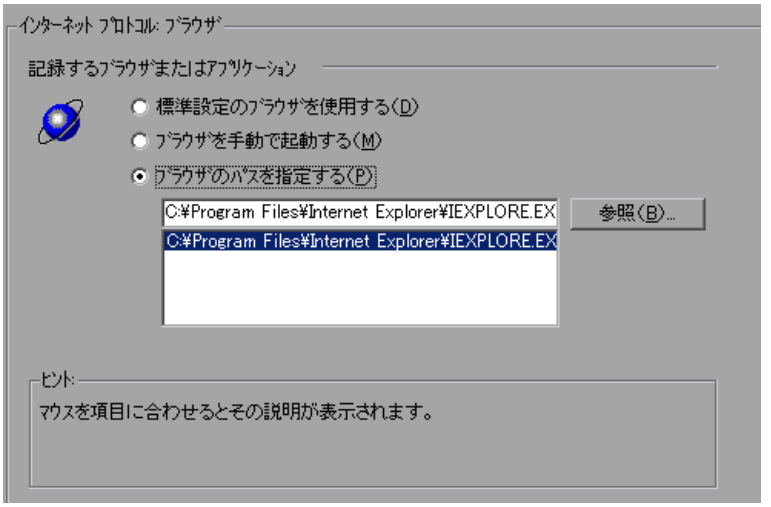
記録する前に，記録オプションを設定し，記録する情報，記録に使用するブラウザまたはクライアント，スクリプトの内容を指定します。

プロキシ設定やその他の詳細設定など，一般的な HTTP プロパティの記録オプションを設定できます。詳細については，第 51 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 57 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

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（Web/Winsocket デュアル・プロトコルのみ）。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［**HTTP プロパティ：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。

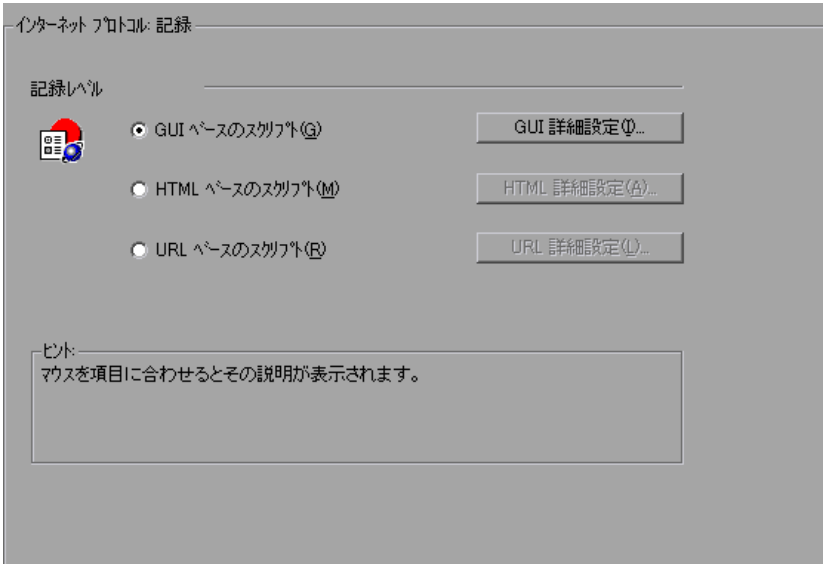
- ［**標準設定のブラウザを使用する**］：VuGen によって記録用コンピュータで通常使用する Web ブラウザとして指定されているブラウザが使用されます。
- ［**ブラウザを手動で起動する**］：VuGen によって記録の開始時にブラウザが起動されません。ユーザは，記録セッションを開始した後に，自分でブラウザまたはアプリケーションを起動しなければなりません。
- ［**ブラウザのパスを指定する**］：VuGen によって指定したブラウザが使用されます。パスのリストからパスを選択するか，［**参照**］ボタンをクリックして，使用するアプリケーションの場所を見つけます。

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・スクリプトの生成時に，記録する情報と使用する関数が指定できます。記録レベルは，目的または環境によって選択します。指定できるレベルは，**GUI ベースのスクリプト**，**HTML ベースのスクリプト**，および **URL ベースのスクリプト**です。（Web）HTTP/HTML および Web/Winsocket デュアル仮想ユーザの場合は，後者の 2 つのオプションのみ使用可能です。

以下のガイドラインに従って，どの記録レベルを選択するかを決定します。

- JavaScript を使用するアプリケーションを含む大部分のアプリケーションでは，［**GUI ベースのスクリプト**］を使用します。このレベルは，PeopleSoft Enterprise および Oracle Web Applications 11i 仮想ユーザの場合にも推奨されます。
- アプレットおよび VB スクリプトを使用するブラウザ・アプリケーションの場合は，［**HTML ベースのスクリプト**］を作成します。
- 非ブラウザ・アプリケーションの場合は，［**URL ベースのスクリプト**］を使用します。



GUI ベースのスクリプト・オプションは，HTML アクションを **web_text_link** などのコンテキスト・センシティブ GUI 関数として記録するよう VuGen を設定します。

```
/* GUI ベース・モード：JavaScript をサポートしている CS タイプ関数
*/
vuser_init()
{
web_browser("mercuryWebTours",
            DESCRIPTION,
            ACTION,
            "Navigate=http://localhost:1080/mercuryWebTours/",
            LAST);

web_edit_field("username",
            "Snapshot=t2.inf",
            DESCRIPTION,
            "Type=text",
            "Name=username",
            "FrameName=navbar",
            ACTION,
            "SetValue=jojo",
            LAST);
…
```

HTML ベースのスクリプト・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：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);
…
```

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。

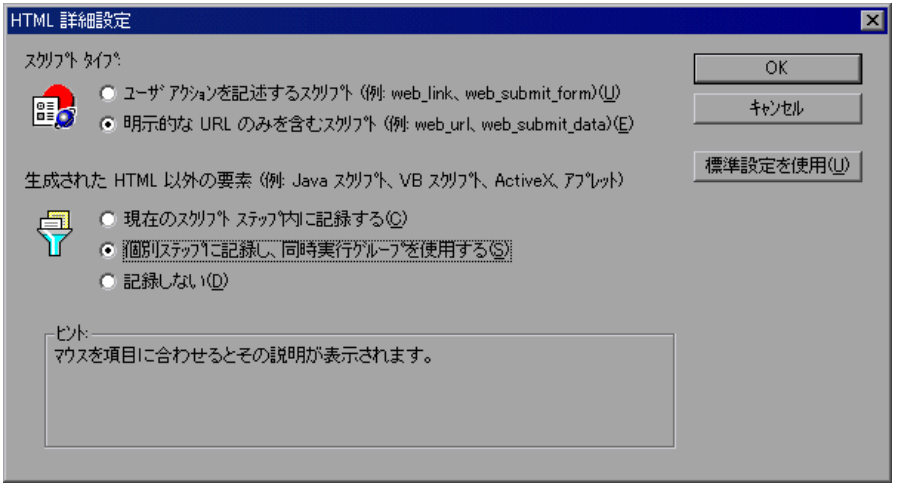
記録後，元の記録とは異なる方式でスクリプトを再生成することもできます。たとえば，スクリプトを HTML ベース・レベルで記録している場合，URL ベース・レベルで再生成できます。スクリプトを再生成するには，［**ツール**］＞［**スクリプトの再生成**］を選択し，［**オプション**］をクリックして，再生成のための記録オプションを設定します。

HTML ベースのオプションの詳細設定

Web（HTTP/HTML）および Web/Winsocket デュアル仮想ユーザの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 ベースのスクリプト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 ベースのスクリプトでは，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**）などの関数が作成されます。コンテキスト・センシティブ記録と似た非常にわかりやすいスクリプトが作成されます。

```
/* 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，JavaScript など，HTML 以外の項目が含まれています。通常こうした HTML 以外の要素には，それ自身のリソースが含まれているか，取得されるかします。たとえば，記録された Web ページから呼び出された JavaScript **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，**web_url**，**web_link**，**web_submit_data** などの関連する関数の引数としてリストされます。Web 関数の引数となったリソースには **EXTRARES** フラグが付けられます。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，同時実行グループ内に配置されます（**web_concurrent_start** と **web_concurrent_end** で囲まれます）。

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);
```

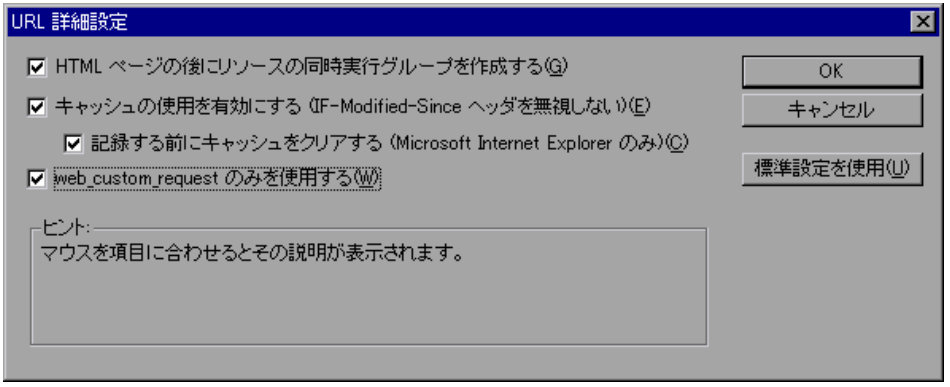
URL ベースのモードでの記録では，VuGen によってページ上のすべてのフレームの内容が記録され，**TargetFrame** 属性が省略されます。

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，リソースが同時実行グループとして記録されます（**web_concurrent_start** と **web_concurrent_end** で囲まれます）。リソースには，画像などのファイルと **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，その他のグラフィック・ファイル，インポートされたファイルも記録の対象となります。

ブラウザのキャッシュ

ブラウザのキャッシュによって，Web ページへのアクセスに要する時間を短縮するために，直近に表示されたページがマシンのメモリに格納されます。標準設定では，［**キャッシュの使用を有効にする**］オプションが無効になっています。つまり，VuGen によってサーバから直接すべてのページが取得され，記録時にブラウザのキャッシュが使用されません。

ただし，アプリケーションによっては，キャッシュなしでは実行できないものもあります。キャッシュを有効にして，新しく変更されたページのみを直接サーバから取得するには，［**キャッシュの使用を有効にする**］オプションを選択します。

HTTP ヘッダー，「**If-Modified-Since**」は，キャッシュされたリソースが最後のダウンロードからサーバ側で更新されたかどうかをクライアント側で検査するときに使用される要求です。リソースが変更されていると，クライアントによってそのリソースが再びキャッシュにダウンロードされます。そうでない場合は，サーバから HTTP ステータス・コード 304「Not Modified」が返されます。キャッシュが無効になっている場合，「**If-Modified-Since**」ヘッダーは抑止され，サーバからすべてのページが直接取得されます。このモードでは，応答ヘッダーの **Last-Modified**，**Expires**，および **Etag** だけでなく，要求ヘッダーの **If-None-Match** も削除されます。ブラウザ側でこれらの応答ヘッダーのいずれも受信されなかった場合，画像はブラウザのキャッシュに格納されません。

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（Click and Script）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，821 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

（次の説明は日本語版 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・マシンで正しく表示されません。このことは，EUC コードの日本語文字を仮想ユーザ・スクリプトで表示するときに影響します。

記録中，VuGen は HTTP ヘッダーを通じて Web ページで使われているコードを検出します。文字セットに関する情報が HTTP ヘッダーに存在しない場合は，HTML のメタ・タグを調べます。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］の［**詳細**］ノードの［**EUC-JP**］オプションを有効にします（日本語版 Windows でのみ使用できます）。

［**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。

2 記録オプション・ツリー内の［**一般：記録**］ノードを選択します。

3 記録レベルとして［**GUI ベースのスクリプト**］（使用可能な場合），［**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。

4 GUI ベースの記録の場合は，記録オプションを開いて（Ctrl + F7），［**GUI プロパティ：詳細**］ノードを選択し，記録時にイベントをキャプチャするための追加オプションを設定します。

詳細については，766 ページ「GUI プロパティの詳細設定」を参照してください。

5 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，833 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。

6 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。これらのオプションの詳細については，838 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。

7 日本語版 Windows のユーザの場合は，［**EUC-JP**］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-JP**］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 53 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・プリファレンスの設定
- 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 13 章「実行環境の設定」を参照してください。ネットワーク速度の実行環境の設定に関する詳細については，第 14 章「ネットワーク実行環境の設定」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。インターネットの実行環境設定では，プロキシ，ブラウザ，その他の詳細な環境設定が行えます。

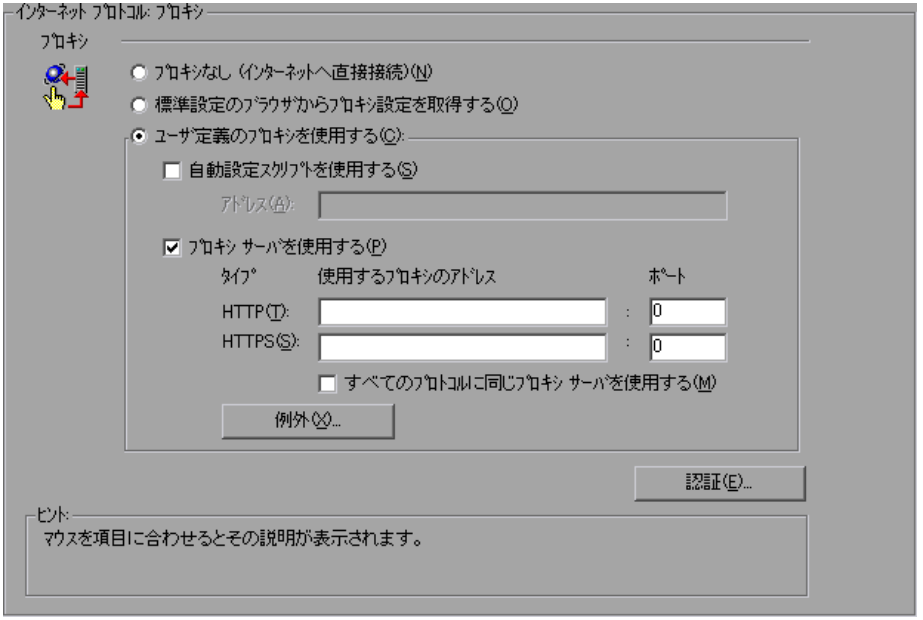
インターネット関連の実行環境の設定は，［実行環境設定］ダイアログ・ボックスで行います。必要な設定を行うためのノードをクリックします。

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い方については，第 50 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。



［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（849 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：再生中にプロキシの応答を待つように仮想ユーザに指示し，プロキシが基本認証をサポートしていると仮定しないようにするには，次のステートメントを追加します。
web_set_sockets_option("PROXY_INITIAL_BASIC_AUTH", "0");

認証

プロキシ・サーバ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，［**再生ログ**］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
…
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［**実行環境設定の編集**］ボタンをクリックするか，［**仮想ユーザ**］＞［**実行環境の設定**］を選択します。

2 ［**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 プロキシ・オプションを選択します。［**プロキシなし（インターネットへ直接接続）**］，［**標準設定のブラウザからプロキシ設定を取得する**］，［**ユーザ定義のプロキシを使用する**］があります。

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（たとえば，**http://hostname/proxy.pac**）またはファイル・サーバ上の場所（たとえば，**C:¥temp¥proxy.pac**）を指定できます。

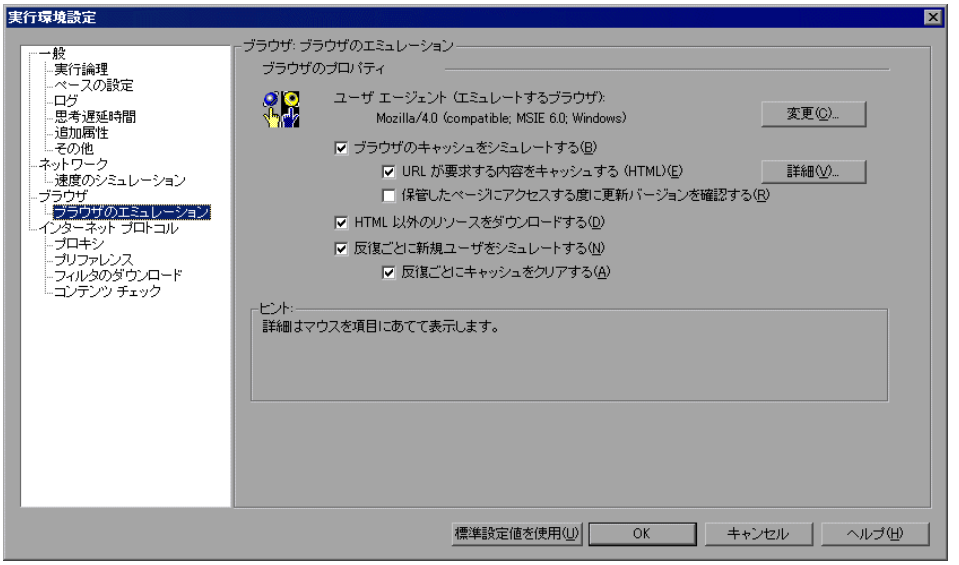
5 仮想ユーザがプロキシ・サーバを使わずに直接アクセスする URL を指定するには，［**例外**］をクリックし，それらの URL をリストに追加します。［例外］ダイアログ・ボックスでは，ローカル（イントラネット）アドレスへの直接アクセスも指定できます。

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（たとえば「pclt/default.htm」），プロトコルのバージョン（たとえば「HTTP/1.0」）が含まれています。2 行目以降には，「ヘッダー情報」が「属性名，コロン，値」の形式で含まれています。要求は，空白行で終わります。

仮想ユーザのヘッダーには，エミュレートされているブラウザの種類を示す「**User-Agent**」ヘッダーが含まれています。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，仮想ユーザはすべてのキャッシュ機能を無視し，要求ごとにすべてのリソースをダウンロードします。

キャッシュ・シュミレーションが無効になっているとき，リソースがページ上に複数回表示されていても，各リソースはページごとに一度だけダウンロードされます。リソースは画像，フレーム，または別の種類のスクリプト・ファイルです。

LoadRunner や Performance Center などで複数の仮想ユーザを実行しているときは，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で，キャッシュが利用されずにブラウザがエミュレートされます。

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。

➤ ［**URL が要求する内容をキャッシュする（HTML）**］：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。標準設定では，このオプションは有効になっています。

ヒント：仮想ユーザのメモリ使用量を減らすには，テストの明示的な要件でなければ，このオプションを無効にします。

キャッシュする内容の種類のリストに内容の種類をさらに追加するには，［**詳細設定**］をクリックします。詳細については，856 ページ「URL が要求する内容をキャッシュする－詳細」を参照してください。親オプションの［**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 によって画像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ファイルと，明示的にはページと共に記録されていない画像ファイル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。

ヒント：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。

URL が要求する内容をキャッシュする－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 ［**URL が要求する内容をキャッシュする**］オプションを有効にします。

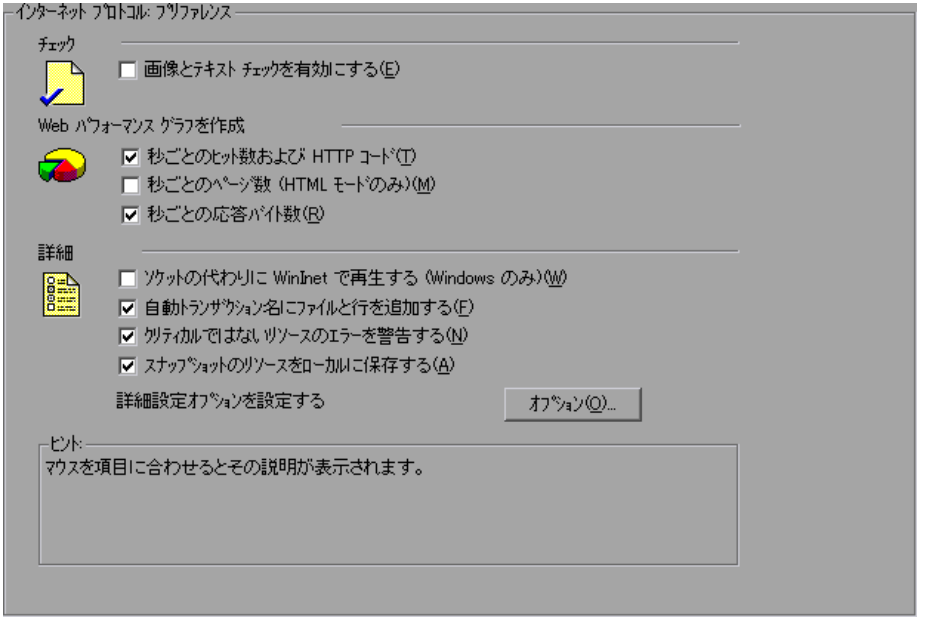
2 ［詳細］ボタンを押下して詳細設定を開き，プラス記号をクリックして，text/plain，text/xml，image/jpeg，image/gif などの追加内容タイプを選択します。内容名をテキスト・ボックスに入力します。

3 内容タイプをリストから削除するには，その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・プリファレンスの設定

［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，［**毎秒のヒット数**］，［**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 は標準のソケット再生の代わりに WinInet 再生エンジンを使用します。VuGen には，Sockets ベース（標準設定）および WinInet ベースの 2 つの HTTP 再生エンジンがあります。WinInet は Internet Explorer で使用されるエンジンであり，IE ブラウザに組み込まれている機能をすべてサポートしています。WinInet 再生エンジンの制約として，スケーラブルでないこと，UNIX をサポートしていないことが挙げられます。さらに，スレッドでの作業時に WinInet エンジンがモデム速度および接続数を正確にエミュレートしないことがあります。

VuGen 独自のソケット・ベースの再生は，負荷テストにおけるスケーラビリティに優れた小型軽量のエンジンです。また，スレッドでの使用時にも正確に機能します。ソケット・ベースのエンジンの制限事項としては，SOCKS プロキシがサポートされていない点があります。このタイプの環境を記録する場合は，WinInet 再生エンジンを使用してください。

➤ ［**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。特定の内容タイプを重要なものとして設定するには，非リソースのリストにその内容タイプを加えます。詳細については，825 ページ「リソース以外のコンテント・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・プリファレンス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，［HTTP のバージョン］、［Keep-Alive HTTP 接続］、［Accept-Language 要求ヘッダ］、［HTTP 要求接続タイムアウト］、［HTTP 要求受信タイムアウト］、［DNS のキャッシュ］、［ステップ ダウンロード タイムアウト］および［DNS のキャッシュ］などです。

HTTP

- ［**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「Keep-Alive HTTP 接続」を参照してください。
  - HTML 圧縮。868 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。
- ［**Keep-Alive HTTP 接続**］：Keep-Alive（キープ・アライブ）は，持続的な接続を可能にする HTTP 拡張機能を表す用語です。こうした長時間の HTTP セッションでは，複数のリクエストを同一の TCP 接続を介して送信できます。これにより，Web サーバとクライアントのパフォーマンスが向上します。

この Keep-Alive オプションは，Keep-Alive 接続をサポートする Web サーバがあって初めて機能します。この設定により，仮想ユーザ・スクリプトを実行するすべての仮想ユーザに対して Keep-Alive HTTP 接続を有効にすることができます（標準では有効）。

- ［**Accept-Language 要求ヘッダ**］：受け入れ言語のカンマ区切りリストを提供します。たとえば，**en-us**，**fr** などがあります。
- ［**HTTP エラーを警告とする**］：HTTP エラーによりリソースのダウンロードが失敗した場合に，エラーの代わりに警告を発行します。
- ［**HTTP 要求接続タイムアウト（秒）**］：仮想ユーザが，ステップ内の特定の HTTP 要求への接続を中断するまで待機する時間（秒単位）です。タイムアウトは，要求に対してサーバが安定しユーザに応答するための猶予を与えるものです（標準設定の値は 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 要求受信タイムアウト（秒）**］：仮想ユーザがステップ内の特定の HTTP 要求への応答を受信するまで待機する時間（秒単位）です。タイムアウトは，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）。
- ［**zlib ヘッダを要求**］：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「**No**」に設定します（標準設定は「Yes」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。

一般

- ［**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）。

- ［**UTF-8 から、または UTF-8 への変換**］：受信した HTML ページおよび送信されたデータの UTF-8 への変換と UTF-8 からの変換をします。記録オプションで UTF-8 のサポートを有効にします。詳細については，819 ページ「記録オプションの詳細設定」を参照してください（標準設定では無効）。

- ［**リソースによって発生したステップ タイムアウトを警告とする**］：タイムアウトの時間内にロードされなかったリソースが原因でタイムアウトが発生した場合に，エラーではなく警告を発行します。リソース以外の場合は，エラーが発行されます（標準では無効）。

- ［**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。

  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。
- ［**エラーとして発行された一致エラー数の上限**］: LB（左境界）または RB（右境界）を使用して，内容チェックにエラーとして発行されたエラーの一致数を制限します。これは，文字列が見つかったときに失敗となる（Fail=Found）一致に適用されます。以降の一致は，情報メッセージとして一覧表示されます。標準設定値は 10 です。
- ［**同じページに対する 'META refresh' の上限**］: META 更新がページごとに実行される最大回数。標準設定値は 2 です。

認証

- ［**認証の再試行時の固定思考遅延時間（ミリ秒）**］: ユーザによる認証情報（ユーザ名とパスワード）の入力をエミュレートするため，思考遅延時間を仮想ユーザ・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）。
- ［**NTLM2 セッション セキュリティを無効にする**］: より基本的な NTLM 2 セッション・セキュリティ応答ではなく，完全な NTLM 2 ハンドシェイク・セキュリティを使用します（標準設定は「No」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（標準設定は「No」です）。
- ［**大きい KDC 負荷をかける**］: 前の反復で取得した資格情報を再利用しません。この設定を有効にすると，KDC（キー配布サーバ）にかかる負荷が高くなります。サーバにかかる負荷を減らすには，このオプションを「**Yes**」に設定して，前の反復で取得した資格情報を再利用します。このオプションは Kerberos 認証が使用されている場合にのみ関係します（標準設定は「No」です）。

ログ

- ［**プリンタ バッファの行長**］: 行の折り返しを無効にして，要求，応答ヘッダー，ボディ，または JavaScript ソース，あるいはこれらすべてを印刷するための行の長さです。
- ［**プリンタ バッファ バイナリ ゼロのみエスケープ**］:
  - ［**Yes**］: 要求，応答ヘッダー，ボディ，または JavaScript ソース，あるいはこれらすべてを印刷する場合に，バイナリのゼロ値のみをエスケープします。
  - ［**No**］: 印刷不能な文字や制御文字をすべてエスケープします。

Web（Click and Script）固有

- ［**一般**］:
  - ［**ホームページの URL**］: ブラウザを開いたときに表示されるホーム・ページの URL（標準設定値は about:blank）。
  - ［**DOM ベースのスナップショット**］: サーバ応答の代わりに DOM からスナップショットを生成するよう VuGen に指示します（標準設定値は「**Yes**」）。
  - ［**HTTP による文字セットの変換**］: "Content-Type:....; charset=..." HTTP 応答ヘッダーに基づいて文字セットの変換を実行します。［UTF-8 から、または UTF-8 への変換］に優先します。
  - ［**META による文字セットの変更があったら解析しなおす**］: META タグによる文字セットの変更があったら解析しなおします。［**HTTP による文字セットの変換**］が有効な場合にのみ設定できます。［**Auto**］は，最初の反復で使用したときにのみ解析が有効になることを意味します。
  - ［**JavaScript エラー時に失敗する**］: JavaScript の評価エラーが発生した場合に，仮想ユーザが失敗します。標準設定値は「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。
- ［**タイマ**］:
  - ［**ステップ終了時にタイマを最適化する**］: 可能であれば，期限前に，ステップの終了時に期限切れになる setTimeout/setInterval/ ＜ META 更新＞を実行します（標準設定は「Yes」です）。

- ［**Single 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）。
- ［**累積 setTimeout/setInterval しきい値（秒）**］: window.setTimeout および window.setInterval メソッドのタイムアウトを指定します。遅延時間がこのタイムアウトを超えた場合，それ以上の window.setTimeout および window.setInterval 呼び出しは無視されます。タイムアウトはステップごとに累積されます（標準設定値は 15 秒です）。
- ［**ステップ終了時に setInterval を設定しなおす**］: **0** なら「No」，**1** なら「1 回」，**2** なら「Yes」です。

- ［**ActiveX サポート**］:
  - ［**ActiveX コントロールをサポートする**］: DOM で ActiveX オブジェクトを作成および実行します（Windows OS のみ）。標準設定は「No」です。
  - ［**ActiveX オブジェクトをサポートする**］: JavaScript で作成された ActiveX コントロールを実行します（Windows OS のみ）。標準設定は「Yes」です。
  - ［**非同期 ActiveX ナビゲーションのために WinInet を強制する**］: 非同期 ActiveX ナビゲーションのために，現在使用されている再生用のネットワーク機能の代わりに WinInet を強制します。
  - ［**同期 ActiveX ナビゲーションのために WinInet を強制する**］: 同期 ActiveX ナビゲーションのために，現在使用されている再生用のネットワーク機能の代わりに WinInet を強制します。
  - ［**ActiveX WinInet ナビゲーションにクッキーを追加する**］: HTTP ターボ再生により受け取ったクッキーを ActiveX WinInet ナビゲーションに追加します。
- ［**アプレット サポート**］:
  - ［**Java アプレットを実行する**］: Java アプレットを実行します（APPLET タグまたは classid のない OBJECT タグに遭遇したときなど）。
  - ［**Java アプレット ナビゲーションのために WinInet を強制する**］: Java アプレット・ナビゲーションのために，現在使用されている再生用のネットワーク機能の代わりに WinInet を強制します。

- **［履歴］:**
  - ［**履歴のサポート**］: テスト実行での window.history オブジェクトのサポートを有効にします。［Yes］，［No］，［Auto］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。
  - ［**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 100 ステップ）。
- **［ナビゲータのプロパティ］:**
  - ［**navigator.browserLanguage**］: ナビゲータ DOM オブジェクトの **browserLanguage** プロパティに設定されているブラウザ言語です。標準設定は記録された値です。標準設定では，古い記録エンジンで作成されたスクリプトは，**ja** を使用します。
  - ［**navigator.systemLanguage**］: ナビゲータ DOM オブジェクトの **systemLanguage** プロパティに設定されているシステム言語です。標準設定は記録された値です。標準設定では，古い記録エンジンで作成されたスクリプトは，**ja** を使用します。
  - ［**navigator.userLanguage**］: ナビゲータ DOM オブジェクトの **userLanguage** プロパティに設定されているユーザ言語です。標準設定は記録された値です。標準設定では，古い記録エンジンで作成されたスクリプトは，**ja** を使用します。
- **［メモリ管理］:**
  - ［**DOM メモリ割り当ての標準ブロック 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）。
  - ［**JavaScript ランタイム メモリのサイズ (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。
  - ［**JavaScript のスタック メモリ 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。

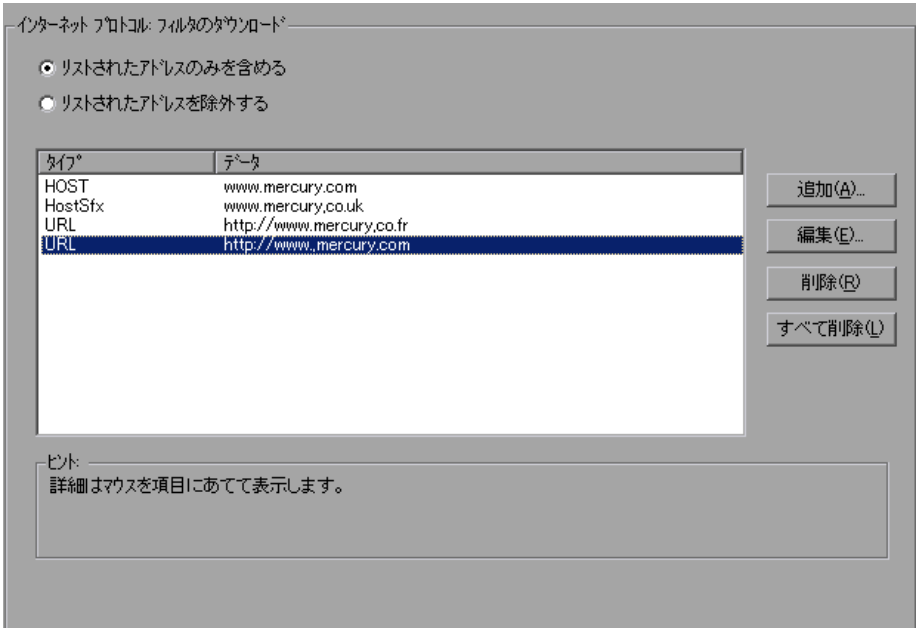
Web サイトのフィルタリング

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。

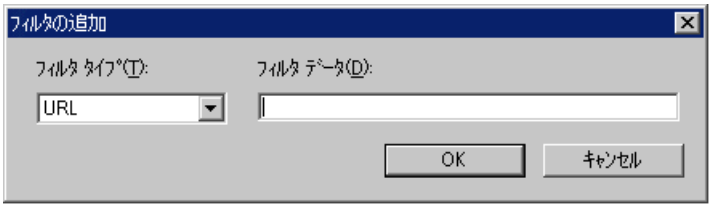


フィルタする Web サイトのリストを作成するには，次の手順を実行します。

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**］，［**ホスト**］，または［**ホストのサフィックス**］を選択し，URL などのフィルタ・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，202 ページ「実行環境設定のログの設定」を参照してください。

デバッグ情報には，次のものがあります。

- ログ情報
- トランザクションの失敗

➤ ゲートウェイとの接続ステータス（接続中，切断中，リダイレクト中のいずれか）（WAP のみ）

より詳細なデバッグ情報を得るには，**default.cfg** ファイルを編集します。［**Web**］セクションで，**LogFileWrite** フラグを **1** に設定します。生成されるトレース・ファイルには，スクリプト実行時のすべてのイベントが記録されます。

負荷テストを実施する場合は，必ず **LogFileWrite** フラグをクリアして，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：プリファレンス**］ノードを使用します。［**オプション**］をクリックして［詳細オプション］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，859 ページ「インターネット・プリファレンス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 54 章

Web ページの内容のチェック

仮想ユーザ・スクリプトを記録した後で，ページの内容をチェックするように実行環境を設定できます。

本章では，次の項目について説明します。

- Web ページのコンテンツ・チェックについて
- コンテンツ・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 ページのコンテンツ・チェックについて

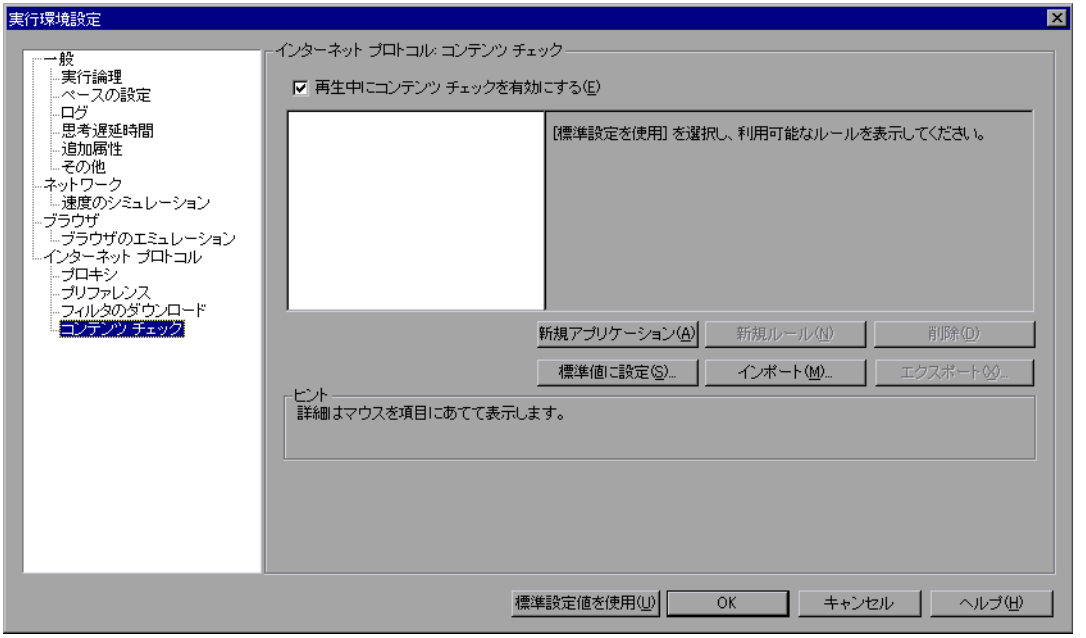
VuGen のコンテンツ・チェックのメカニズムを使って，ページ内容の特定の文字列を検査することができます。このオプションは，標準的ではないエラーを検出するのに役立ちます。通常の運用では，アプリケーション・サーバで障害が発生するとブラウザが汎用の HTTP エラー・ページを表示して，エラーの性質を通知します。この標準のエラー・ページは VuGen に認識され，エラーとして処理されるので，スクリプトは失敗として報告されます。しかし，アプリケーション・サーバの中には，VuGen がエラー・ページとして認識しない固有のエラー・ページを発行するものもあります。エラー・ページはサーバによって送信され，エラーが発生したことを示す特定の形式のテキスト文字列を含んでいます。

たとえば，エラーが発生したときに，「**ASP Error**」というテキストを含んでいるユーザ定義ページを発行するアプリケーションがあるとします。その場合には，VuGen に対して，サーバから返されたすべてのページでこのテキストを検索するように指定します。VuGen はこの文字列を検出すると，再生を失敗とします。なお，VuGen はページ本体は検索しますが，ヘッダーは検索しません。

コンテンツ・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 : コンテンツ チェック**］で，検索する文字列を指定します。複数のコンテンツ・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。

- コンテンツ・チェック・ルールについて
- アプリケーションとルールの追加と削除
- コンテンツ・チェック・ルールの定義



コンテンツ・チェック・ルールについて

VuGen のコンテンツ・チェックのメカニズムを使って，ページ・コンテンツの特定の文字列を検査することができます。このオプションは，標準的ではないエラーを検出するのに役立ちます。通常の運用では，アプリケーション・サーバで障害が発生するとブラウザが汎用の HTTP エラー・ページを表示して，エラーの性質を通知します。この標準のエラー・ページは VuGen に認識され，エラーとして処理されるので，スクリプトは失敗として報告されます。しかし，アプリケーション・サーバの中には，VuGen がエラー・ページとして認識しない固有のエラー・ページを発行するものもあります。エラー・ページはサーバによって送信され，エラーが発生したことを示す特定の形式のテキスト文字列を含んでいます。

たとえば，エラーが発生したときに，「**ASP Error**」というテキストを含んでいるユーザ定義ページを発行するアプリケーションがあるとします。その場合には，VuGen に対して，サーバから返されたすべてのページでこのテキストを検索するように指定します。VuGen はこの文字列を検出すると，再生を失敗とします。なお，VuGen はページ本体は検索しますが，ヘッダーは検索しません。

コンテンツ・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。

- ［**再生中にコンテンツ チェックを有効にする**］：再生中のコンテンツ・チェックを有効にします（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（Web (Click and Script)，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，ほかのユーザとの共有や，ほかのマシンでのインポートができます。
- ［**削除**］：選択したルールまたはアプリケーションを削除します。

ルールのインポートとエクスポート

- ［**インポート**］/［**エクスポート**］：ルール・ファイルをインポートまたはエクスポートします。アプリケーションとルールの指定は，**xml** という拡張子を持ったルール・ファイルに格納されます。ルールをファイルにエクスポートすることで他のマシンでも利用できるようになります。他のルール・ファイルをインポートすることも可能です。選択してインポートしたルールが既存のルールと矛盾する場合，矛盾するルールであることを示す警告が VuGen によって表示されます。その場合，既存のスクリプトに対して作成したルールを，インポートしようとしているルールと統合するか，現在のルールを上書きするように指定できます。［エクスポート］をクリックすると，［エクスポートするアプリケーションの選択］ダイアログ・ボックスが表示されます。

標準のルールの設定

- ［**標準値に設定**］：コンテンツ・チェックには，**インストール**，**標準設定**，**スクリプトごと**の 3 つのタイプがあります。「インストール」ルールは製品のインストール時に自動的に定められます。「標準設定」ルールはマシンで実行するすべてのスクリプトに適用されます。「スクリプトごと」のルールは，現在のスクリプトに対して定義されているものです。ルールを変更または追加したとき，その変更は現在のスクリプトにのみ適用されます。VuGen で，あるルールを標準設定ルールのリストに加えて，そのマシンのすべてのスクリプトに適用するには，［**標準値に設定**］をクリックします。

複数のスクリプトを使う場合や，製品のアップグレードを行う場合には，標準設定のルールとスクリプトごとのルールの間に矛盾が生じる場合があります。その場合，VuGen から，ルールを統合するかどうかの確認を求められます。ルールを統合すると（推奨），アプリケーションのルール・リストにルールが追加されます。

この操作は，左側の表示枠のアプリケーション・リストで有効になっているアプリケーションにのみ適用されます。現在のスクリプトで**有効**になっているアプリケーションがなければ，Defaults ファイルでも**有効**なアプリケーションはありません。既存の Defaults ファイルを上書きするには，［**はい**］をクリックします。操作を取り消して，既存の Defaults ファイルを維持する場合は，［**いいえ**］をクリックします。

ルールは標準の XML ファイルに保存されます。このルール・ファイルをエクスポートすれば，ほかのユーザとの共有や，ほかのマシンでのインポートができます。

［標準値に設定］をクリックして上書きを承認すると，VuGen によって次の処理が行われます。

1 Defaults ファイルの全アプリケーションに**無効**の印を付けます。

2 現在のスクリプトで**有効**になっているアプリケーションについて，Defaults ファイル内でのアプリケーションの有無に応じて，統合またはコピーを行います。アプリケーションが存在する場合，現在のスクリプトのルールが Defaults ファイルのルールに統合されます。アプリケーションが Defaults ファイルになければ，ルールは VuGen によって 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。

コンテンツ・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 : コンテンツ 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。

コンテンツ・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 : コンテンツ チェック**］ノードを選択します。

2 ［**再生中にコンテンツ チェックを有効にする**］オプションを選択します。

3 ［**新規アプリケーション**］ボタンをクリックして，内容を検査するアプリケーションのリストに新しいエントリを追加します。

4 既存のアプリケーションに対するルールを追加するには，［**新規ルール**］をクリックします。各アプリケーション・サーバに 1 つまたは複数のルールを適用できます。左側の表示枠でルールの横のチェック・ボックスのオン / オフを切り替えて，ルールの有効 / 無効を切り替えます。

5 実際のテキスト文字列を検索するには，［**検索するテキスト**］を選択して，検索したいテキストを入力します。テキストは，可能な限り具体的にすることをお勧めします。たとえば，「**エラー**」ではなく「**ASP エラー**」のように，アプリケーション固有のテキストを入力するようにします。

6 文字列の前後に付くテキストに基づいて検索するには，［**検索対象プレフィックス / サフィックス**］を選択し，前置記号と後置記号を指定します。

7 大文字と小文字を区別して指定するには，［**大文字と小文字を区別する**］を選択します。

8 ルールが，マシンに格納されているすべてのスクリプトに適用されるよう標準として設定するには，ルールまたはアプリケーションを選択して［**標準値に設定**］をクリックします。

9 ルールのファイルをエクスポートするには，［**エクスポート**］をクリックして，保存場所を指定します。

10 ルールのファイルをインポートするには，［**インポート**］をクリックし，ファイルの場所を探します。

11 アプリケーションやルールを削除するには，ルールを選択して［**削除**］ボタンをクリックします。

12 アプリケーション全体に標準の設定を使用するには，［**標準設定値を使用**］をクリックします。アプリケーションの一覧と標準の設定を含むダイアログ・ボックスが表示されます。矛盾がある場合は，ルールを上書きまたは統合できます。

第 55 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。

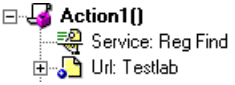
負荷下の検証について

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 チェックでは，Web ページに特定のオブジェクトがあるかどうかを検証します。オブジェクトは，テキスト文字列または画像です。

Web チェックによって，多数の仮想ユーザがアクセスしているときに Web サイトが正しく機能するか，つまりサーバが正しい Web ページを返すかどうかを確認できます。これが特に重要なのは，サイトに多数のユーザの負荷がかかるときです。大きな負荷がかかった状態では，サーバが間違ったページを返す可能性が高くなるからです。

たとえば，世界の主要都市の気温に関する情報を表示する Web サイトがあるとします。VuGen を使って，その Web サイトにアクセスする仮想ユーザ・スクリプトを作成します。

仮想ユーザはサイトにアクセスし，この Web ページのテキストをチェックします。たとえば，ページ内に「**Temperature**」という単語が表示されれば，チェックは合格です。サーバから正しいページが返されなかったために，「**Temperature**」という単語が表示されなければ，チェックは不合格になります。テキスト・チェックのステップは URL ステップの前に出現します。これは，VuGen によって次のステップに必要な検索情報が「登録」される，つまり，事前に準備されるためです。仮想ユーザ・スクリプトを実行すると，以降の Web ページを対象とするチェックが実行されます。



スクリプトを記録したときや，単独の仮想ユーザでスクリプトを実行したときには，サーバから正しいページを返されていたとしても，多数の仮想ユーザでサーバに負荷をかけた場合には，正しいページが返されないことがあります。サーバが過負荷になり，そのために無意味な，あるいは不正な HTML コードが返されることがあります。また，過負荷になったサーバから「**500 Server Error**」ページが返されることもあります。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 では，それぞれが異なるチェックの種類を示す数種類の Web チェック・アイコンを使用しています。

Web チェック・アイコン	詳細
テキスト 	テキスト・チェック。次のアクション関数ステップ内で（**web_reg_find**），またはビジネス・プロセスステップ全体の中で（**web_global_verification**），特定の文字列を検索します。
テキスト 	テキスト・チェック。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，882 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，882 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 では，Web ページのテキスト文字列を検索するチェックを追加できます。テキスト・チェックは記録中または記録後に追加できます。

テキスト・チェックを作成するときに，チェックの名前，チェックの範囲，チェックするテキスト，および検索条件を定義します。

記録中にテキスト・チェックを追加するには，次の手順を実行します。

1 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，対象となるテキストを選択します。

2 記録ツールバーの［**テキスト チェックを挿入**］ボタンをクリックします。VuGen によって **web_reg_find** 関数がスクリプトに追加されます。

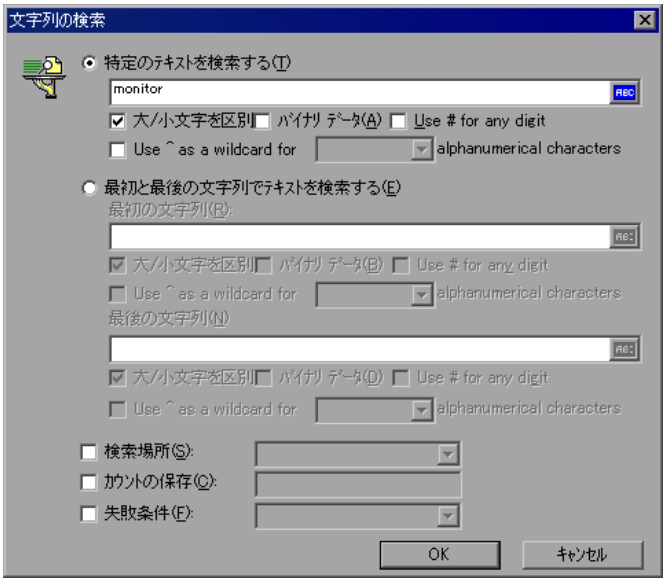
記録後のテキスト・チェックを追加するには，次の手順を実行します。

1 テキスト・チェックの対象となるステップのスナップショットに移動します。

2 スナップショットの中で，検証対象テキストを選択します。

3 右クリックして表示されるポップアップ・メニューから［**テキスト チェックを追加（web_reg_find)**］を選択します。［文字列の検索］ダイアログ・ボックスが開きます。

注：プロトコルによっては，スナップショットからではなく［サーバの応答］タブからテキスト・チェックを追加しなければならないというメッセージが発行されます。［**サーバの応答**］タブをクリックし，［**HTML ドキュメント**］タブを選択します。本文のノードを拡張し，次に示すとおり続行します。



web_reg_find では次の属性を使用できます。

- [**特定のテキストを検索する**]：検索するテキスト文字列です。この属性は空でない，NULL 終端文字列である必要があります。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

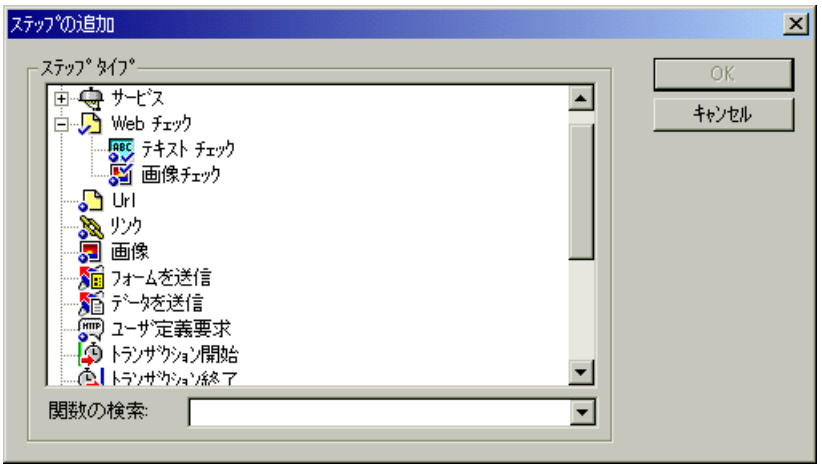
web_find 関数は下位互換性を保つためにだけ提供されているもので，HTML ベースのスクリプトに限定されている点が **web_reg_find** 関数とは異なります（［**記録オプション**］＞［**記録**］ノードを参照）。この関数にはインスタンスなどの追加の属性もあり，テキストの出現回数を決めることができます。標準のテキスト検索を実行する場合には，**web_reg_find** 関数のほうが便利です。

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。



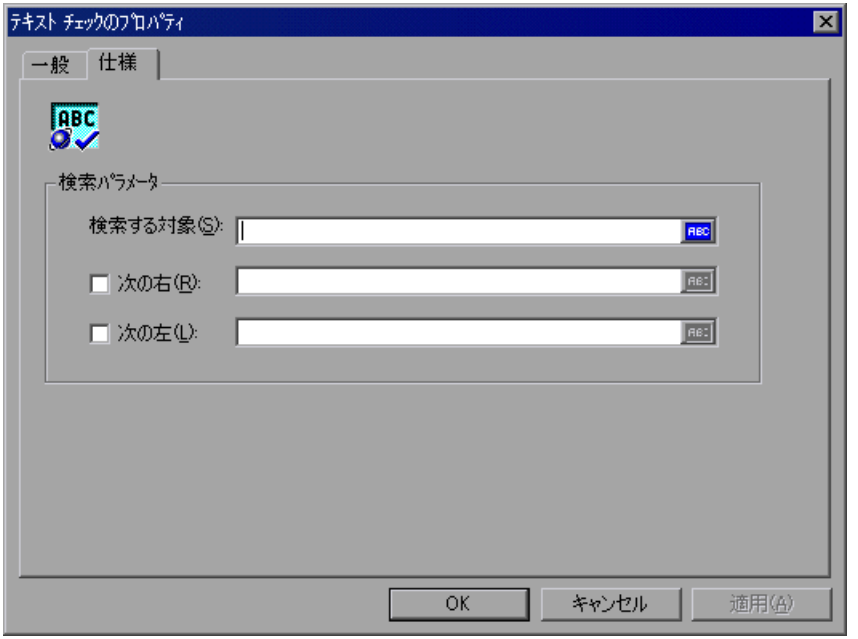
2 **web_find** 関数の場合は，［**Web チェック**］ノードを展開して［**テキスト チェック**］を選択します。**web_global_verification** 関数の場合は，［**サービス**］ノードを展開して関数名を選択します。［プロパティ］ダイアログ・ボックスが開きます。

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_global_verification 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。パラメータの指定については，第 9 章「VuGen パラメータを使った作業」を参照してください。

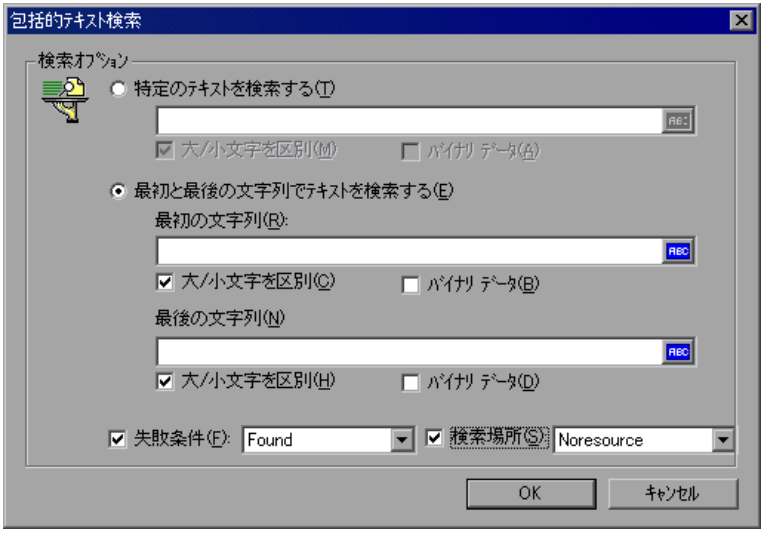
- ［**次の右**］または［**次の左**］：隣接するテキストを基準とする検索文字列の位置。該当するフィールドにテキストを入力します。たとえば，「support@mercuryinteractive.com」という文字列が「e-mail:」という単語の右に表示されることを検証する際，［**次の右**］を選択して，［**次の右**］ボックスに「e-mail:」と入力します。
- ［**ステップ名**］：テキスト・チェックの名前。［**一般**］タブをクリックし，テキスト・チェックの名前をボックスに入力します。後で識別しやすいようにわかりやすい名前を付けます。

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 13 章「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。パラメータの指定については，第 9 章「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**］：条件を満たす場合にスクリプトを失敗させます。失敗条件として，テキストが［**Found**］なのか（見つかったのか），または［**Not found**］なのか（見つからなかったのか）を指定することもできます。必要な動作を［**失敗条件**］ボックスで選択します。

テキスト・フラグ

登録された検索 **web_reg_find** を使用して検索テキストを指定するときは，フラグを追加して検索範囲を制御できます。

/IC は，大文字小文字を区別しないよう指定します。

/BIN は，バイナリ・データを指定します。

/DIG は，シャープ記号（#）を 1 桁の数値を表すワイルドカードとして解釈するよう指定します。DIG フラグではリテラルのシャープ記号は一致しません。

/ALNUM ＜大文字小文字＞は，キャレット記号（^）を 1 文字の US-ASCII 英数字を表すワイルドカードとして解釈するよう指定します。これには 3 つの構文があります。**ALNUMIC** は，大文字小文字を区別しません。**ALNUMLC** は，小文字にのみ一致します。**ALNUMUC** は，大文字にのみ一致します。ALNUM フラグではリテラルのキャレットは一致しません。

フラグを使用するには，属性 **TEXT** を入力し，その直後にスラッシュとフラグ名を入力します。たとえば，大文字小文字を区別せずに文字列を検索するには，"Text/IC= **検索対象テキスト** '" と入力します。

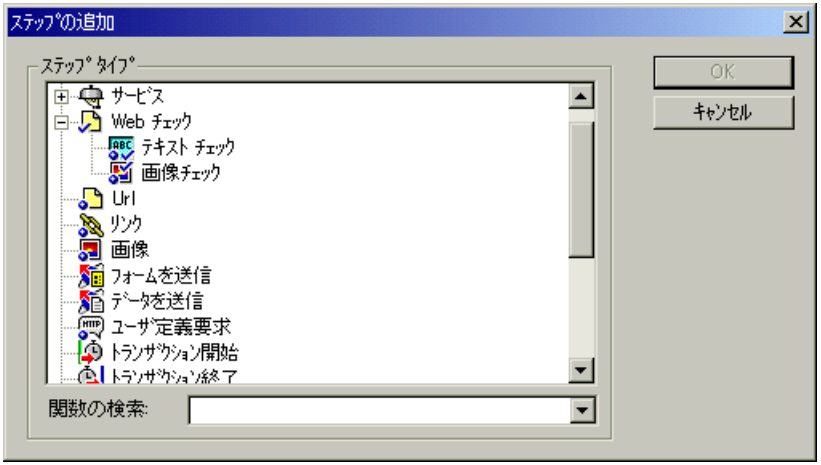
画像チェックの追加

VuGen を使って Web ページ内の画像を検索するユーザ定義チェックを追加できます。画像は ALT 属性と SRC 属性のどちらかまたは両方によって識別できます。

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。

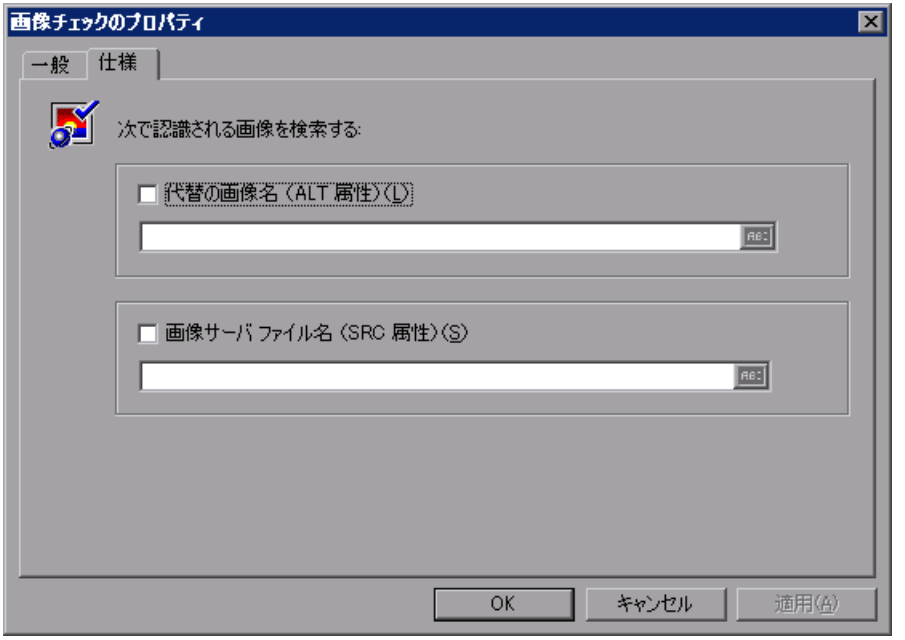
1 VuGen のメイン・ウィンドウで，チェックを実行する Web ページに対応するステップを右クリックします。ポップアップ・メニューから［**後に挿入**］を選択します。［ステップの追加］ダイアログ・ボックスが表示されます。



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。

- ［**代替の画像名（ALT 属性）**］: **ALT** 属性を使って画像を識別します。テキスト・ボックスに ALT 属性のテキスト値を入力します。スクリプトを実行すると，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。
- ［**画像サーバ ファイル名（SRC 属性）**］: SRC 属性を使って画像を識別します。テキスト・ボックスに SRC 属性のテキスト値を入力します。スクリプトを実行すると，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。パラメータの指定については，第 9 章「VuGen パラメータを使った作業」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。

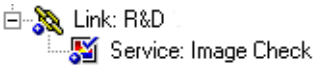
5 画像チェックに名前を付けるには，[**一般**] タブをクリックします。[**ステップ名**] ボックスに，画像チェックの名前を入力します。後で識別しやすいようにわかりやすい名前を付けます。

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，890 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 13 章「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 9 章「VuGen パラメータを使った作業」を参照してください。

第 56 章

Web とワイヤレス仮想ユーザ・スクリプトの変更

Web またはワイヤレス仮想ユーザ・スクリプトの記録が終わったら，VuGen を使って，記録したスクリプトを変更できます。新規ステップの追加のほか，既存のステップの編集，名前の変更，削除ができます。

本章では，次の項目について説明します。

- 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 48 章「Web 仮想ユーザ・スクリプトの作成」を参照してください。

本章では，VuGen を使用してツリー・ビューでスクリプトを変更する方法について説明します。テキスト・ベースのスクリプト・ビューでのスクリプトの変更については，「**オンライン関数リファレンス**」（［**ヘルプ**］＞［**関数リファレンス**］）を参照してください。

バイナリ・データの追加

バイナリ・コード・データを HTTP 要求の本体に含めるには，次の形式に従います。

¥x[char1][char2]

これは，[char1][char2] によって表される 16 進値です。

たとえば，¥x24 は 10 進数で表せば 16*2+4=36 となり，対応する文字は $ 記号です。同様に，¥x2B は + 記号です。

16 進法のシーケンスとして 1 桁のシーケンスは使用しないでください。たとえば，¥x2 は有効なシーケンスではありませんが，¥x02 は有効なシーケンスです。

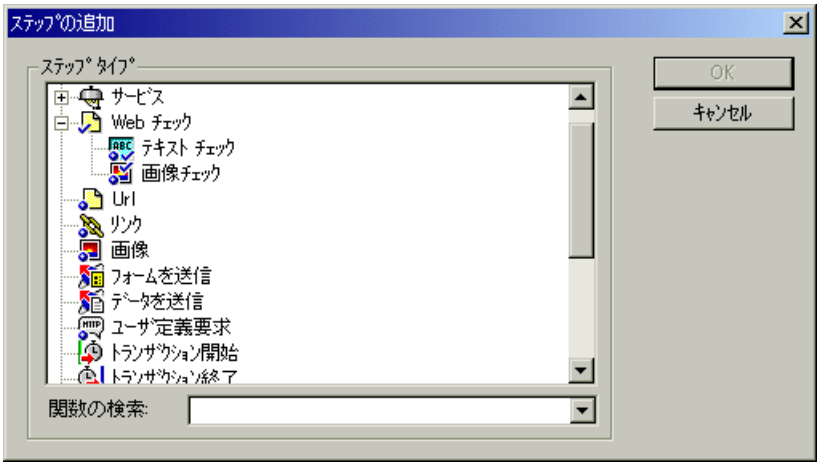
仮想ユーザ・スクリプトへのステップの追加

Web ブラウザまたはツールキットの記録セッション中に VuGen が記録するステップに加え，記録されたスクリプトにステップを追加することもできます。

仮想ユーザ・スクリプトにステップを追加するには，次の手順を実行します。

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 8 章「仮想ユーザ・スクリプトの拡張」
サービス・ステップ	917 ページ「サービス・ステップの変更」
Web チェック	918 ページ「Web チェックの変更（Web のみ）」
トランザクション	914 ページ「トランザクションの変更」
ランデブー・ポイント	915 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	916 ページ「思考遅延時間の変更」
「URL」ステップ	897 ページ「「URL」ステップの変更」
「リンク」ステップ	899 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	901 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	903 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	907 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	911 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 8 章「仮想ユーザ・スクリプトの拡張」

仮想ユーザ・スクリプトからのステップの削除

ブラウザ・セッションまたはツールキット・セッションの記録後に，VuGen を使って仮想ユーザ・スクリプトからステップを削除できます。

仮想ユーザ・スクリプトからステップを削除するには，次の手順を実行します。

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，829 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。

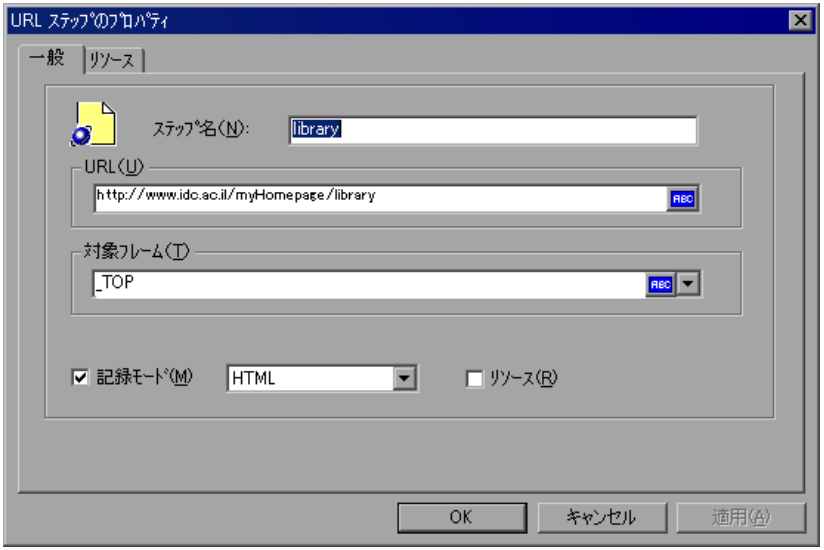
- **HTML**：自動的にすべてのリソースと画像をダウンロードし，以降のステップに必要な HTTP 情報を格納します。このモードは，Web のリンクが含まれるスクリプトに適しています。
- **HTTP**：再生時に，このステップのためにリソースをダウンロードしません。関数によって明示的に指定されたリソースだけをダウンロードします。

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。たとえば，省略する必要のある特定の画像を表すステップがある場合，その種類のリソースを含めないように設定できます。詳細については，839 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。パラメータの指定については，第 9 章「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（変更のあった）フレームの親の内容を置き換えます。
- ［**_SELF**］：最後の（変更のあった）フレームを置き換えます。

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）

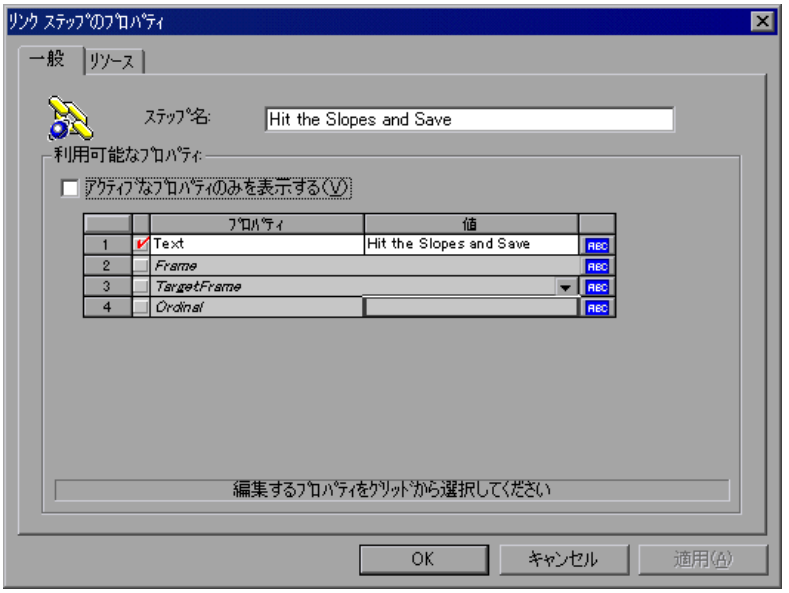
［リンク］ステップは，ハイパーテキスト・リンクをクリックすると，仮想ユーザ・スクリプトに追加されます。このステップは，**HTML** モードで記録するためのオプションを選択した場合にのみ記録されます。詳細については，第 52 章「Web 仮想ユーザの記録オプションの設定」を参照してください。

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，有効なプロパティと無効になっているプロパティの両方を表示します。プロパティを有効にするには，プロパティ名の左のセルをクリックします。［**値**］カラムでプロパティの値を割り当てます。

- ［**Text**］：ハイパーテキスト・リンクの正確な文字列。
- ［**Frame**］：リンクが属しているフレームの名前。

➤ ［**TargetFrame**］：次のターゲット・フレーム。

- ［**_TOP**］：ページ全体を置き換えます。
- ［**_BLANK**］：新規ウィンドウを開きます。
- ［**_PARENT**］：最後の（変更のあった）フレームの親の内容を置き換えます。
- ［**_SELF**］：最後の（変更のあった）フレームを置き換えます。

➤ ［**Ordinal**］：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。パラメータの指定については，第 9 章「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 52 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・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，有効なプロパティと無効になっているプロパティの両方を表示します。プロパティを有効にするには，プロパティ名の左のセルをクリックします。［**値**］カラムでプロパティの値を割り当てます。

- ［**ALT**］：画像の ALT 属性。
- ［**SRC**］：画像の SRC 属性。
- ［**MapName**］：画像に対応するマップ名。クライアント側イメージ・マップにのみ適用されます。

- ［**AreaAlt**］：クリックする領域の ALT 属性。クライアント側イメージ・マップにのみ適用されます。
- ［**AreaOrdinal**］：クリックする領域のシリアル番号。クライアント側イメージ・マップにのみ適用されます。
- ［**Frame**］：画像が配置されているフレームの名前。
- ［**TargetFrame**］：次のターゲット・フレーム。
  - ［**_TOP**］：ページ全体を置き換えます。
  - ［**_BLANK**］：新規ウィンドウを開きます。
  - ［**_PARENT**］：最後の（変更のあった）フレームの親の内容を置き換えます。
  - ［**_SELF**］：最後の（変更のあった）フレームを置き換えます。
- ［**Ordinal**］：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。パラメータの指定については，第 9 章「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）

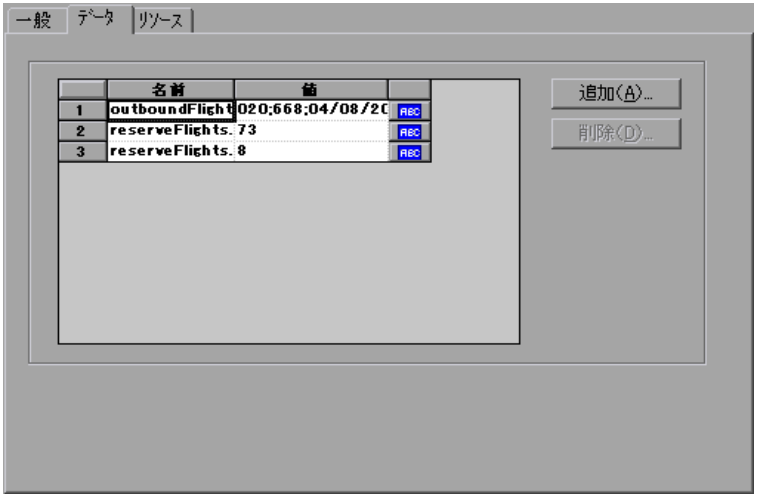
フォームを送信すると，「フォームを送信」ステップが仮想ユーザ・スクリプトに追加されます。このステップは，HTML（コンテキスト・センシティブ）モードで記録するためのオプションを選択した場合にのみ記録されます。詳細については，第 52 章「Web 仮想ユーザの記録オプションの設定」を参照してください。

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，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。

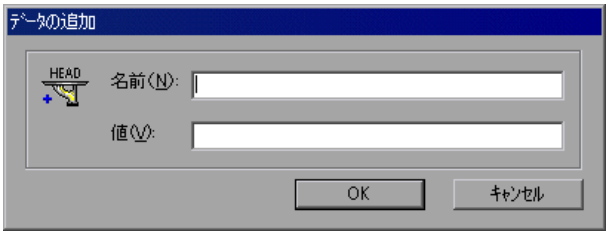
2 右クリックして表示されるメニューから［**プロパティ**］を選択します。［フォームを送信ステップのプロパティ］ダイアログ・ボックスが開きます。［**データ**］タブをクリックします。



- ［**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 9 章「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。

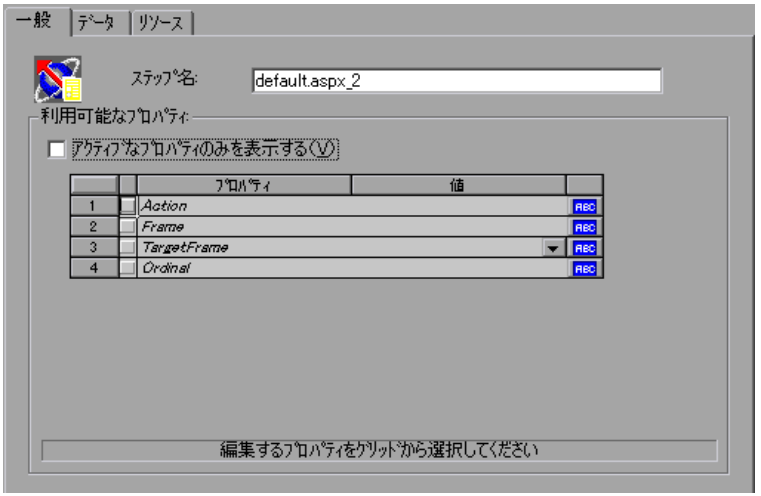
4 フォームの送信に新規データ引数を追加するには，［**追加**］をクリックします。［データの追加］ダイアログ・ボックスが開きます。



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。

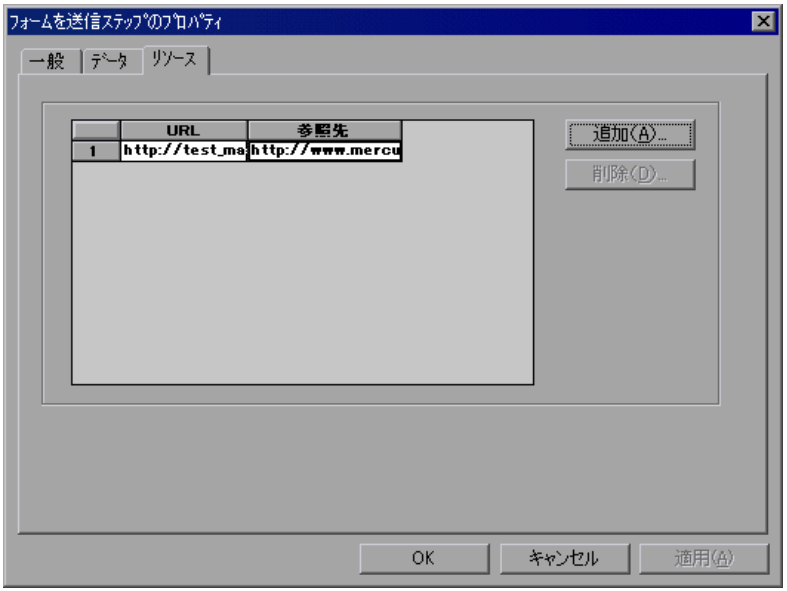
9 プロパティ・テーブルに，フォーム送信のプロパティが表示されます。

［**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**］：次のターゲット・フレーム。
  - ［**_TOP**］：ページ全体を置き換えます。
  - ［**_BLANK**］：新規ウィンドウを開きます。
  - ［**_PARENT**］：最後の（変更のあった）フレームの親の内容を置き換えます。
  - ［**_SELF**］：最後の（変更のあった）フレームを置き換えます。
- ［**Ordinal**］：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。パラメータの指定については，第 9 章「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 と参照先ページを追加します。



11 ［**OK**］をクリックして［フォームを送信ステップのプロパティ］ダイアログ・ボックスを閉じます。

「データを送信」ステップの変更

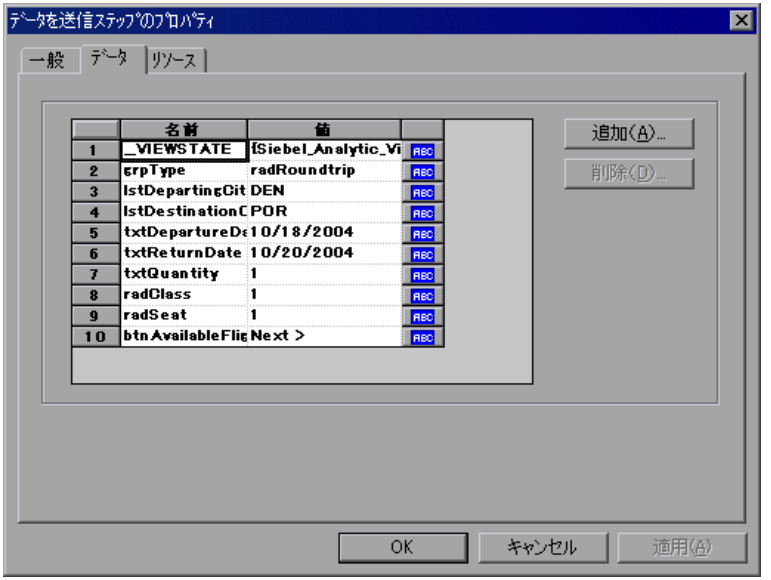
「データを送信」ステップは，Web サイトにデータ・フォームを送信して処理することを表します。「フォームを送信」ステップとは異なり，この要求を実行するときにフォームのコンテキストは必要ありません。

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。

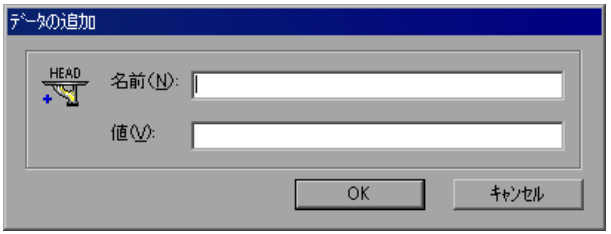
2 右クリックして表示されるメニューから［**プロパティ**］を選択します。［データを送信ステップのプロパティ］ダイアログ・ボックスが開きます。［**データ**］タブをクリックします。



- ［**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 9 章「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。

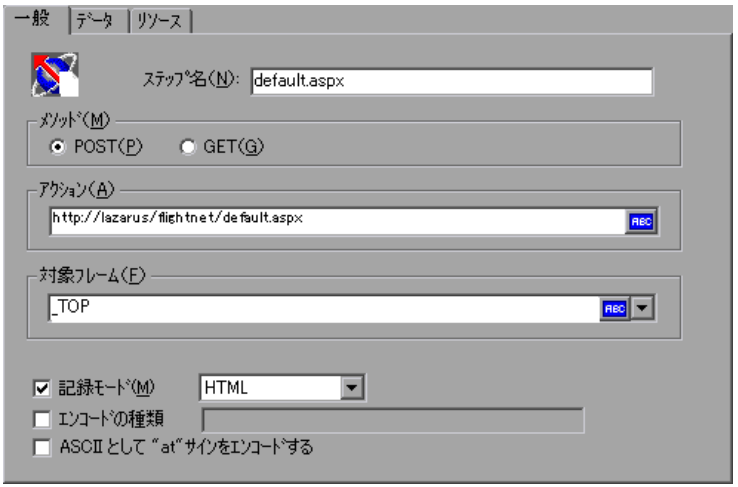
4 新規データを追加するには，［**追加**］をクリックします。［データの追加］ダイアログ・ボックスが開きます。



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。パラメータの指定については，第 9 章「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（変更のあった）フレームの親の内容を置き換えます。
- ［**_SELF**］: 最後の（変更のあった）フレームを置き換えます。

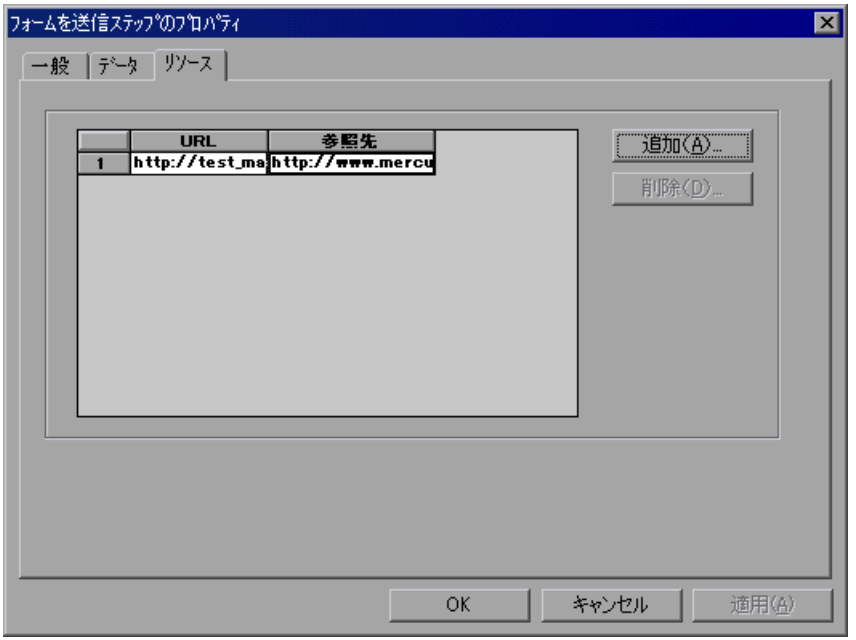
12 再生モードをカスタマイズするには，［**記録モード**］オプションを選択します。次の中から使用するモードを選択します : HTML または HTTP。詳細については，898 ページ「再生モードの設定」を参照してください。

13 **multipart/www-urlencoded** など，エンコーディング・タイプを指定するには，［**エンコードの種類**］チェック・ボックスを選択して，エンコーディング方式を指定します。

14 URL 中の「@」をエンコードするには，［**ASCII として "at" サインをエンコードする**］を選択します。

15 ［**OK**］をクリックして，［データを送信ステップのプロパティ］ダイアログ・ボックスを閉じます。

16 ステップのリソースを指定するには，［**リソース**］タブをクリックします。［**追加**］をクリックして，リソースの URL と参照先ページを追加します。



「ユーザ定義要求」ステップの変更

「ユーザ定義要求」とは，HTTP がサポートするいずれかの方法を使用した，URL に対するユーザ定義の HTTP 要求です。「ユーザ定義要求」ステップには，コンテキストはありません。

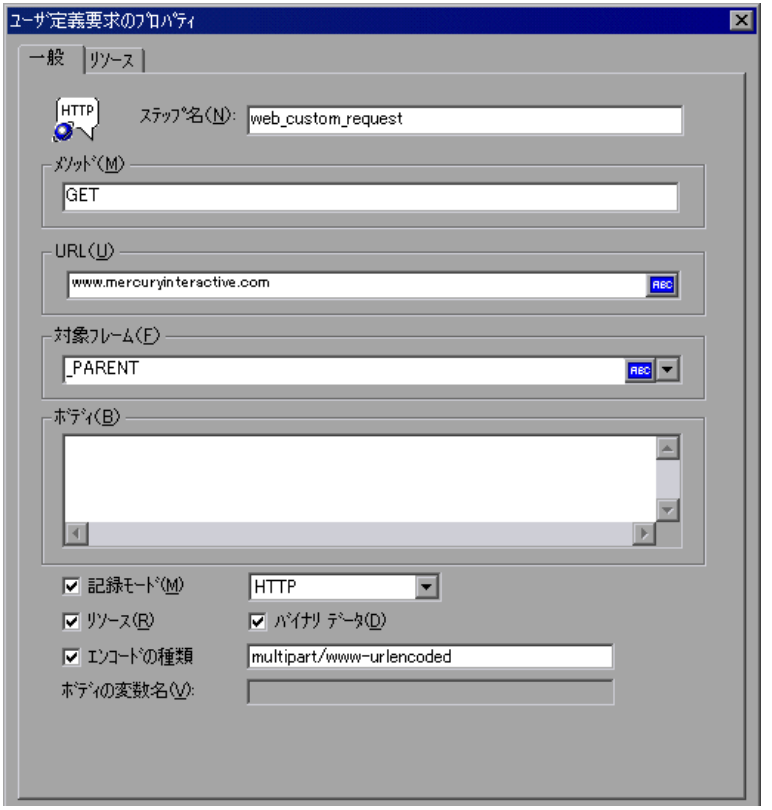
変更できるプロパティは，ステップ名，メソッド，URL，対象フレームおよび本体です。

VuGen には，ユーザ定義要求の本体を C 形式に変換する機能があります。たとえば，XML ツリーまたは大量のデータをユーザ定義要求の本体領域にコピーすることで，現在の関数で使用できるように，文字列を C 形式に変換できます。これにより，必要なエスケープ・シーケンス文字が挿入され，文字列の改行が削除されます。

［ユーザ定義要求］ステップのプロパティを変更するには，次の手順を実行します。

1 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・メニューから［**プロパティ**］を選択します。［ユーザ定義要求の［プロパティ］ダイアログ・ボックスが開きます。



3 ステップ名を変更するには，［**ステップ名**］ボックスに新しい名前を入力します。記録中は，URL の最後の部分が標準名となります。

4 ［**メソッド**］ボックスに，HTTP によってサポートされているメソッドを入力します。たとえば，GET，POST，HEAD です。

5 ［**URL**］ボックスに，要求する URL を入力します。

6 以下の［**対象フレーム**］から 1 つを選択します。

- ［**_TOP**］：ページ全体を置き換えます。
- ［**_BLANK**］：新規ウィンドウを開きます。
- ［**_PARENT**］：最後の（変更のあった）フレームの親の内容を置き換えます。
- ［**_SELF**］：最後の（変更のあった）フレームを置き換えます。

7 ［**ボディ**］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。

長さが 100 KB を超える Body 属性は、［**ボディの変数名**］ボックスの変数に置き換えられます。この変数は **include** フォルダにある **lrw_custom_body.h** ファイルで定義します。

8 ［**ボディ**］ボックスに貼り付けた文字列には，テキストを選択し，右クリックして表示されるメニューから［**C フォーマットに変換**］を選択します。

9 再生モードをカスタマイズするには，［**記録モード**］オプションを選択します。次の中から使用するモードを選択します：HTML または HTTP。詳細については，898 ページ「再生モードの設定」を参照してください。

10 項目をリソースとしてダウンロードしないようにするには，［**リソース**］オプションをクリアします。

11 **multipart/www-urlencoded** など，エンコーディング・タイプを指定するには，［**エンコードの種類**］を選択して，エンコーディング方式を指定します。

12 ［**OK**］をクリックして，［ユーザ定義要求のプロパティ］ダイアログ・ボックスを閉じます。

制御ステップの変更

制御ステップは，負荷テスト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

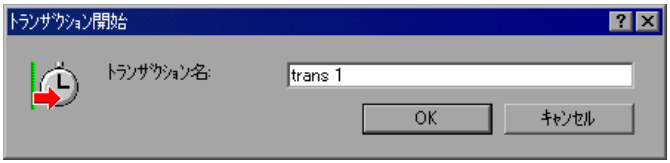
トランザクションとは，サーバの応答時間を測定するタスクや一連のアクションです。

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。

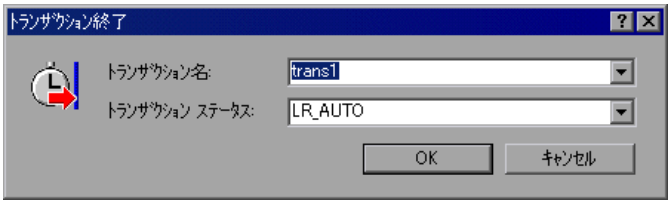


3 トランザクション名を変更するには，［**トランザクション名**］ボックスに新しい名前を入力して［**OK**］をクリックします。

「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・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

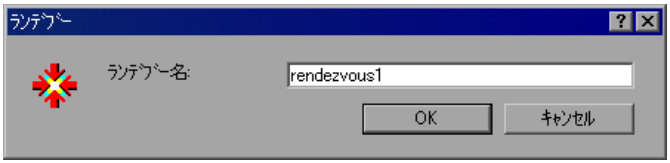
ランデブー・ポイントを使用して，複数の仮想ユーザが同時にタスクを実行するように同期させることができます。

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・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

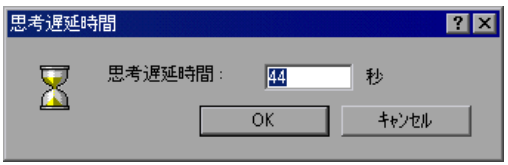
思考遅延時間は，アクション間で実際のユーザが待機する時間をエミュレートします。記録中，アクション間の時間があらかじめ定義したしきい値の 4 秒を超えると，VuGen によってそれぞれのユーザ・アクションの後に，思考遅延時間が自動的に仮想ユーザ・スクリプトに追加されます。

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 13 章「実行環境の設定」を参照してください。

サービス・ステップの変更

「サービス」ステップは，プロキシの設定，認証情報の送信，カスタム・ヘッダーの発行などの，カスタマイズのためのタスクを実行する関数です。「サービス」ステップは，Web サイトのコンテキストを一切変更しません。

記録中または記録後に「サービス」ステップをスクリプトに追加できます。

「サービス」ステップのプロパティを変更するには，次の手順を実行します。

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・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，テキスト文字列または画像です。

記録中または記録後に Web チェックをスクリプトに追加できます。

Web チェックのプロパティを変更するには，次の手順を実行します。

1 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。


—— 画像チェック・アイコン


—— テキスト・チェック・アイコン

2 右クリック・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 55 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 57 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。たとえば，Web サーバによっては，現在の日時を含むリンクを使用するものもあります。

Web 仮想ユーザ・スクリプトを記録すると，動的データがスクリプトに記録されることがあります。そのスクリプトによって，記録された変数が Web サーバに送信されても，変数は有効なものではなくなっています。これらの変数は Web サーバによって拒否され，エラーが発行されます。このようなエラーは必ずしもはっきり分かるわけではなく，仮想ユーザ・ログ・ファイルを注意深く調べないと検出できないことがあります。

仮想ユーザの実行時にエラーが発生した場合は，スクリプト内でのエラーの発生箇所を調べてください。多くの場合，相関によって，1 つのステートメントの結果を別のステートメントの入力項目として使用することで，問題を解決できます。

HTML ページ内の動的データは，次の形式になっている可能性があります。

- 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）
- JavaScript のクッキー

ケース 1

「Buy me now!」というテキストのハイパーテキスト・リンクを含む Web ページがあるとします。HTTP データを含むスクリプトを記録すると，その URL は VuGen によって次のように記録されます。

"http://host//cgi-bin/purchase.cgi?date=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，945 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，829 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 58 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，925 ページ「一致条件の追加」および 926 ページ「詳細相関ルール」を参照してください。

組み込み相関

組み込み相関では，サポートされているアプリケーション・サーバを対象に動的データを検出し，相関させることができます。ほとんどのサーバには，リンクおよび参照の作成時に必ず使用される，明確な構文ルール，つまり「コンテキスト」があります。

たとえば，BroadVision サーバで作成されるセッション ID は，次に示すように，必ず特定の区切り文字の間に挟まれています。左側は「BV_SessionID=」，右側は「&」です。

```
BV_SessionID=@@@@1303778278.0969956817@@@@&
```

サポートされているアプリケーション・サーバを対象にセッションを記録するときは，VuGen に組み込まれている既存ルールの 1 つを使用できます。各アプリケーション・サーバには 1 つ以上のルールを適用できます。また，ルールの横にあるチェック・ボックスを設定またはクリアして，特定のルールを有効または無効にできます。ルールの定義は VuGen の右側の表示枠に表示されます。

サポート対象外のアプリケーション・サーバのセッションを記録するとき，コンテキストが不明で相関のルールを決められない場合には，VuGen のスナップショット比較方式を使用することができます。この方式では，記録後に相関処理の手順を導いてくれます。詳細については，第 58 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**］オプションを選択すると，レコーダはリンク，フォーム・アクションまたはクッキーだけではなく，本体の全体を探します。テキストは，指定した境界を使って検索されます。

リンク / フォーム・アクション

［**リンクおよびフォームのアクションの中でパラメータを検索**］方式では，VuGen はリンク・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。

たとえば，文字列「sessionid=」の 2 回目の出現と「@」の間にあるテキストをパラメータに置き換えるとします。［**左の境界**］ボックスに，左の境界として「**sessionid=**」を指定し，［**右の境界**］ボックスに，右の境界として「**@**」を指定します。2 回目の出現を検索しているので，［**左の境界インスタンス**］ボックスで［**2**］を指定します。

右の境界の文字列が一定でない場合は，［**代替の右境界**］ボックスに，代わりに使用する右の境界を指定できます。指定した右の境界を一意に特定できないときに，この値が使用されます。

たとえば，Web ページに次の形式のリンクが含まれていたとします。

"SessionID=122@page.htm"
"Page.htm@SessionID=122&test.htm"

右の境界が一定でないので（「@」の場合と「&」の場合がある），右の境界として 1 つを指定するだけでは不十分です。この場合，代わりに使用する右の境界として「&」を指定します。

左と右の境界は，文字列を一意に特定するものでなければなりません。境界に動的データを含めることはできません。また，ドロップダウン・メニューの選択肢として用意されている［**文字列の最後**］または［**改行文字**］を，右の境界として指定することもできます。

このオプションでは，左右の境界はスクリプトに含まれている文字列内に必ず含まれている必要があります。サーバが境界を返すだけでは十分ではありません。この制限は他のアクション・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**］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**］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**］方式は，バッファの中で特定の文字列を検出すると **web_reg_add_cookie** 関数を挿入します。指定した接頭辞を持つクッキーを検出したときにのみ関数が追加されます。このオプションでは，検索するテキストとクッキーの接頭辞を指定する必要があります。

一致条件の追加

上記のルール以外に，文字列で次の項目を指定することによって，相関検索の種類を制御することもできます。

- ［**パラメータ前置記号**］：このルールに基づいて自動的に生成されたすべてのパラメータに，接頭辞を使用します。接頭辞によって，既存のユーザ・パラメータへの上書きを防ぐことができます。さらに，接頭辞によって，スクリプト内のパラメータが見分けやすくなります。たとえば，組み込みのルールの 1 つである Siebel-Web は Siebel_row_id という接頭辞を検索します。
- ［**大文字と小文字を区別する**］：境界を検索するときに大文字と小文字を区別します。
- ［**'#' で数字を置き換える**］：数値をすべてハッシュ記号（#）で置き換えます。ハッシュ記号はワイルドカードとして機能し，任意の数字を含むテキスト文字列を検索できます。たとえば，このオプションを有効にし，左の境界として「**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，次の詳細オプションも指定できます。

- ［**常に新規のパラメータを作成する**］：パラメータに置換された値が，前のインスタンスから変わっていない場合も，このルールに応じて新しいパラメータを作成します。Web サーバがページごとに異なる値を割り当てる場合には，このオプションを設定します。たとえば，NetDynamics サーバは，不正行為を最小限に抑えるために，ページごとにセッション ID を変更する場合などがあります。
- ［**完全一致のみパラメータで置換する**］：境界と境界の間のテキストが（最初のスナップショットから）見つかった値と完全に一致した場合のみ，記録された値をパラメータで置き換えます。見つかった文字列の前または後に別の文字がある場合，パラメータの置き換えは実行されません。

  たとえば，フォーム送信の際，VuGen によって境界 aaa と bbb の間の 1234 という文字群が記録され，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（たとえば，Name=12345）が含まれている場合でも，VuGen は，その値をパラメータで置き換えずに，12345 という値を使用します。
- ［**逆方向検索**］：文字列の最後から左境界を検索します。
- ［**左の境界インスタンス**］：一致として考えられる，文字列内（本体ではなく）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，**文字列の最後**，または**改行文字**を指定できます。

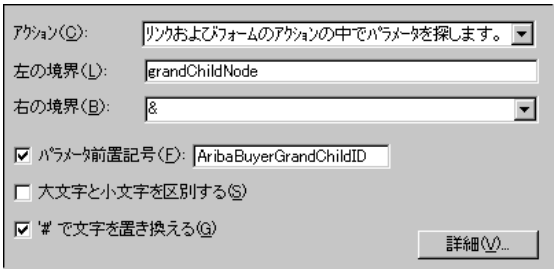
相関のルールの設定

［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，941 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。

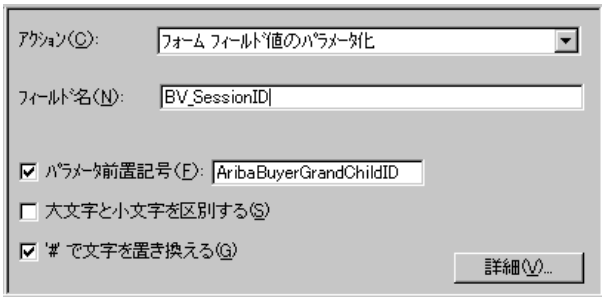
1 既存のルールをクリックするか，表示枠の下の［**新規ルール**］をクリックします。右側の表示枠に相関ルールが表示されます。



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。

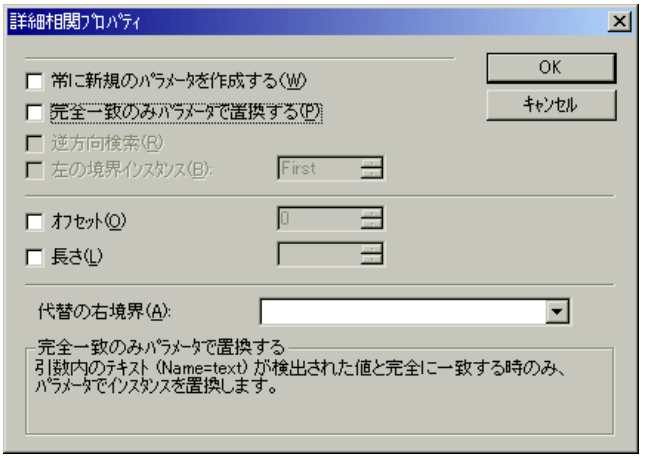
3 最初の 3 種類では，［**左の境界**］と［**右の境界**］ボックスでデータの境界を指定します。

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。

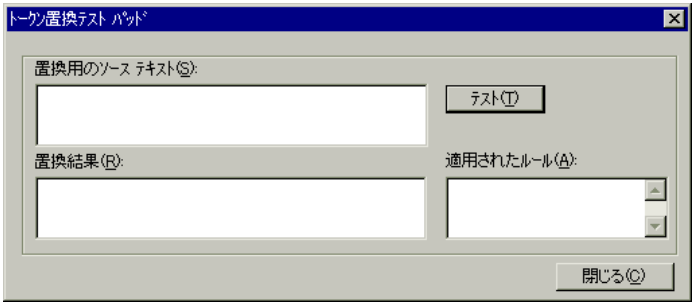
7 ［**テスト**］をクリックして，定義したルールをテストします。詳細については，929 ページ「ルールのテスト」を参照してください。

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。

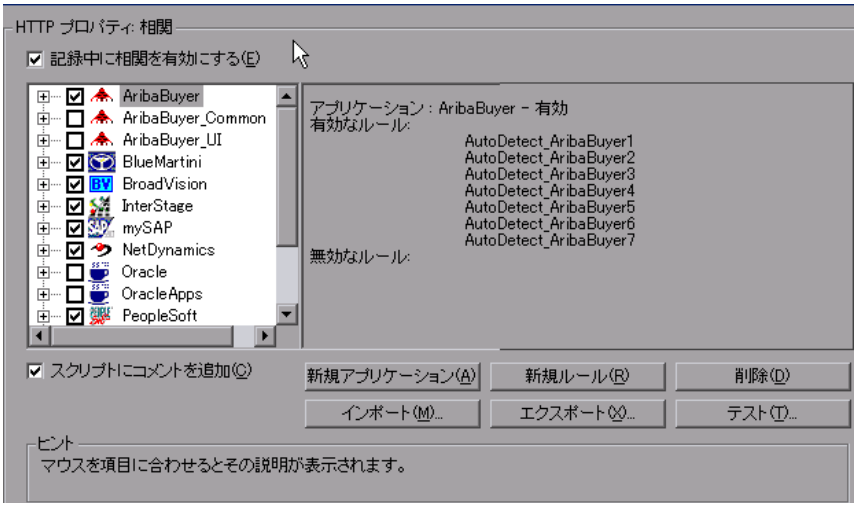
置換が行われた場合，パラメータ化されたソース・テキストは [**置換結果**] ボックスで確認でき，その置換に適用されたルールのリストは [**適用されたルール**] ボックスで確認できます。

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 [**HTTP プロパティ：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，927 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 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。

8 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。

9 ［**OK**］をクリックします。

第 58 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 57 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。

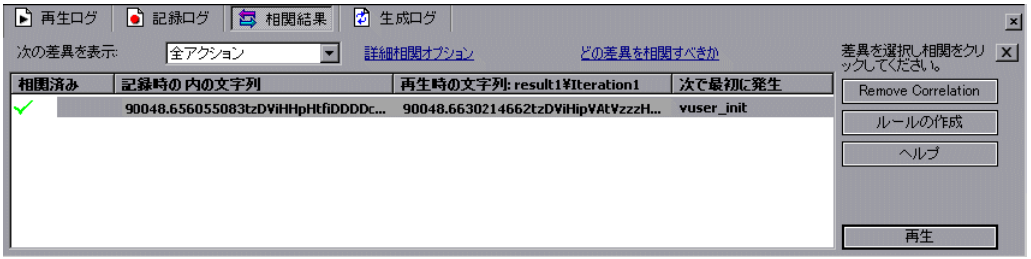
Web 相関メカニズムには，スナップショット間のテキストまたはバイナリの差異を表示できる，比較ユーティリティが組み込まれています。比較の後，差異を 1 つずつ相関させることも，一度にすべてを相関させることもできます。

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，945 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［**再生時の文字列**］の 2 つのカラムには，スナップショット間のテキストの差異が表示されます。その右の［**次で最初に発生**］カラムには，相関が最初に検出されたアクションが表示されます。

スナップショット間の差異を検出したら，相関を選択して［**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，937 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。

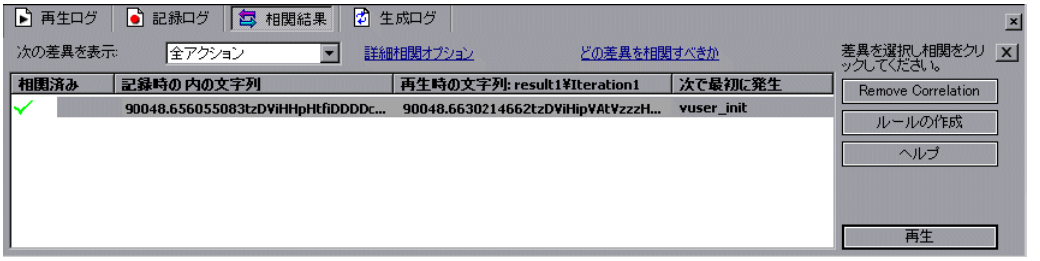
```
// [WCSPARAM WCSParam_Diff1 14 reserveFlights] Parameter
{WCSParam_Diff1} created by Correlation Studio
	web_reg_save_param( "WCSParam_Diff1", "LB= NAME=¥"",
"RB=¥"", "Ord=5", "Search=Body", "RelFrameId=1", LAST );
	web_submit_form("reservations.pl",
		"Snapshot=t4.inf",
		ITEMDATA,
		"Name=depart", "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。

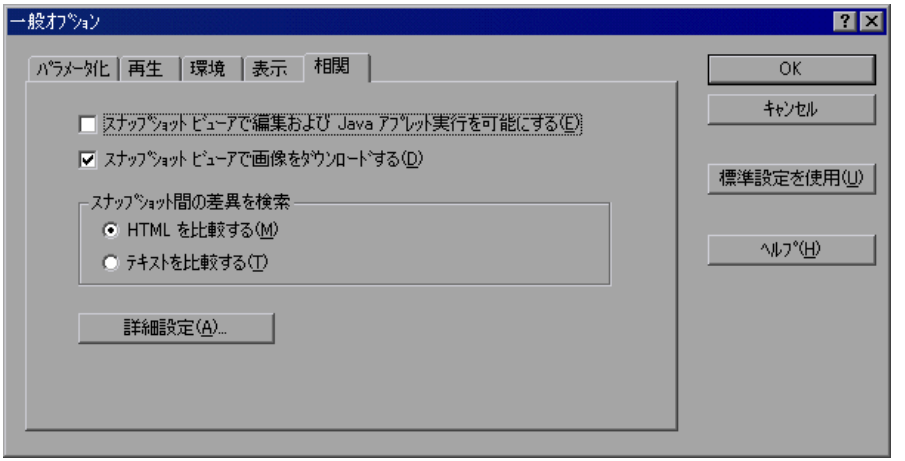
相関を選択し，[**ルールの作成**] をクリックします。相関を選択し，右クリックして表示されるメニューから [**相関ルールを作成**] を選択してルールを作成することもできます。



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。［相関の詳細］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**］：［相関の詳細］ダイアログ・ボックスを開きます。

［相関の詳細］ダイアログ・ボックス

このダイアログ・ボックスでは，区切り文字として扱う文字を指定できます。

➤ ［**区切り文字として扱う文字**］：1 つまたは複数の標準設定以外の区切り文字を指定します。

➤ ［**追加区切り文字**］：復帰，復帰改行，タブ文字など，標準設定の区切り文字を指定できます。設定を変更するには，区切り文字の横のチェック・ボックスをクリアします。

➤ ［**X 文字以下の差異を無視する**］：相関を行うしきい値を指定できます。VuGen は記録時の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。標準設定の長さは 4 文字です。

➤ ［**大きな相関に警告を発行**］：サイズが 10 KB 以上の文字列を相関させようとしたときに警告を表示します。

相関オプションの設定

セッションの記録を開始する前に，相関オプションを設定します。

相関のオプションを設定するには，次の手順を実行します。

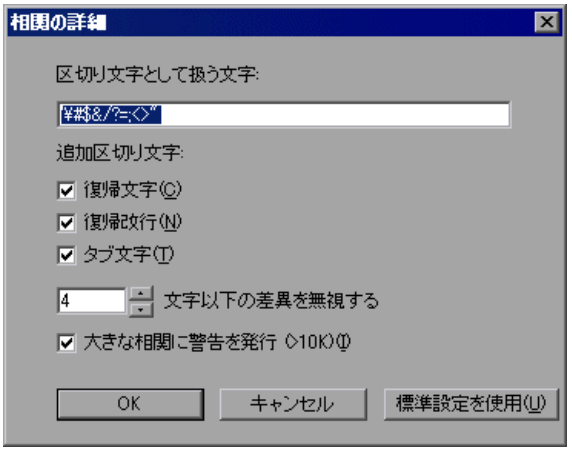
1 ［**ツール**］＞［**一般オプション**］を選び，［**相関**］タブを選択します。

2 ［**スナップショット ビューアで編集および Java アプレット実行を可能にする**］を選択して，スナップショット・ウィンドウでアプレットと Java スクリプトを実行できるようにします。

3 画像をスナップショット・ビューアに表示するよう VuGen に指示するには，［**スナップショット ビューアで画像をダウンロードする**］オプションを選択します。

4 比較の方法として，［**HTML を比較する**］または［**テキストを比較する**］（HTML の要素以外に対してのみ）を選択します。

5 区切り文字を設定するには，［**詳細設定**］をクリックして，［相関の詳細］ダイアログ・ボックスを開きます。



6 ［**区切り文字として扱う文字**］ボックスで，区切り文字として扱うすべての文字を指定します。

7 ［**追加区切り文字**］セクションで必要なオプションを選択して，標準で使用する 1 つまたは複数の区切り文字を指定します。

8 ［**X 文字以下の差異を無視する**］ボックスで，相関のしきい値を指定します。VuGen は記録時の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

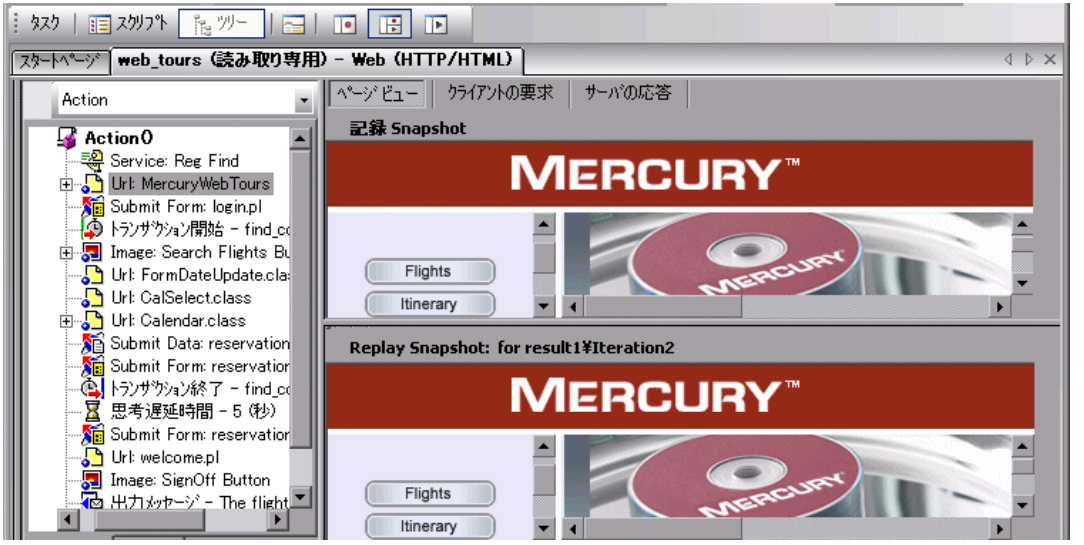
VuGen のスナップショット・ウィンドウを使って，スクリプト内のどの値が動的で相関が必要かを判断します。次の項では，スクリプト内を自動的に検索して差異を見つける方法と，VuGen を使って必要な相関を行う方法について説明します。

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。

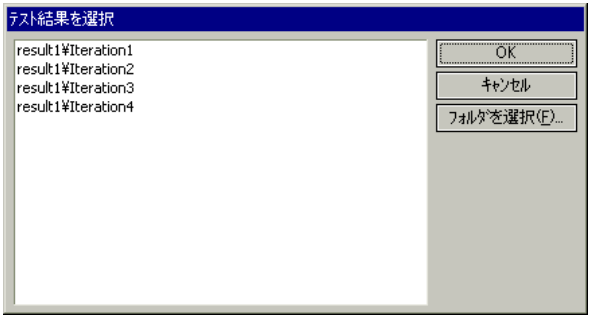
1 スクリプトを開き，ツリー・ビューで表示します（[**表示**] > [**ツリービュー**]）。スナップショットを表示します（[**表示**] > [**スナップショット**] > [**スナップショットを表示**]）。

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。

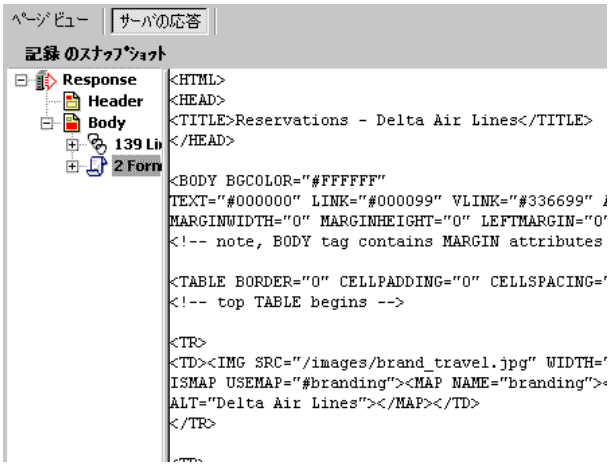


4 2 回目以降の再生時のスナップショットを使用するには，［**表示**］＞［**スナップショット**］＞［**反復の選択**］をクリックします。［テスト結果を選択］ダイアログ・ボックスが開き，スナップショット・ファイルが格納されているフォルダが表示されます。通常，スクリプトのフォルダの下には **result** フォルダと **Iteration** フォルダが表示されます。



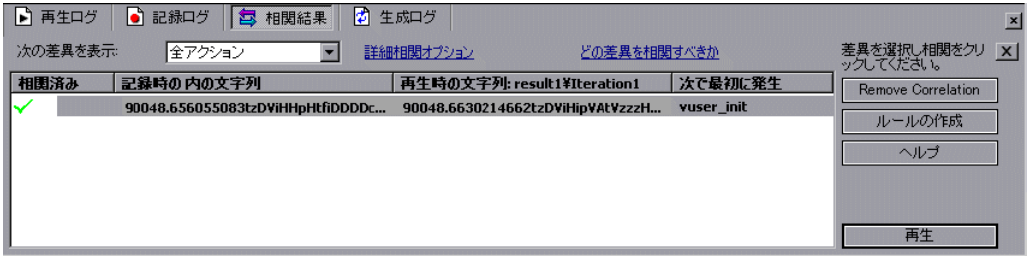
5 スクリプトのサブ・フォルダ以外のフォルダにあるスナップショット・ファイルを選択するには，［**フォルダを選択**］をクリックします。目的のフォルダの場所を見つけ，［**OK**］をクリックします。

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**] > [**相関を検索**] 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。

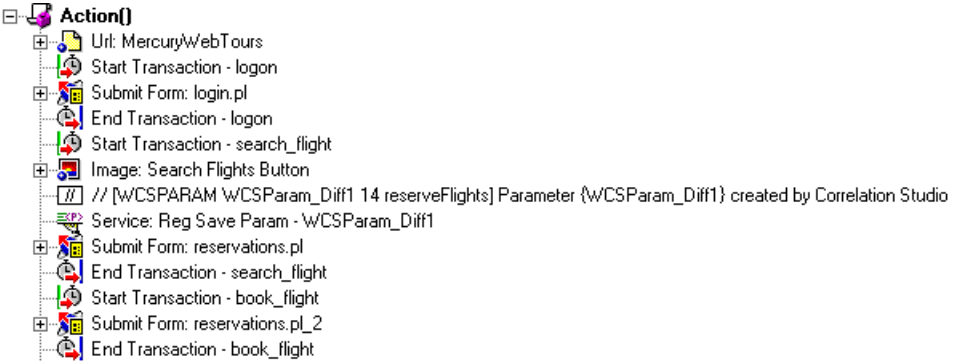
また，記録された文字列と再生された文字列の一部だけが異なる場合も，差異を相関させる必要があります。たとえば，SessionID 文字列の接頭辞と接尾辞が同じでも，中間の文字が異なる場合は相関させる必要があります。

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。

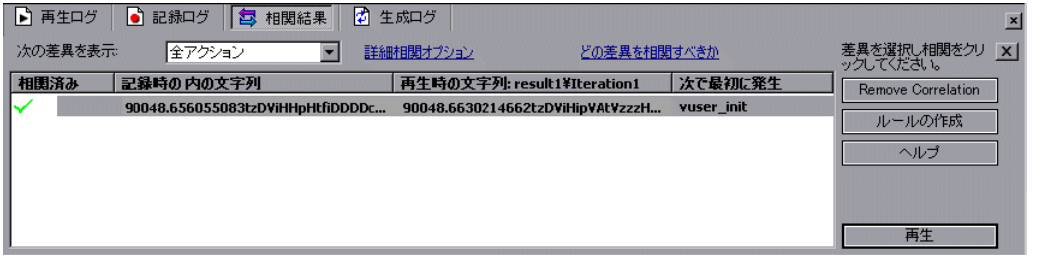
1 ［相関］タブに差異を表示し，相関させる差異を選択します。一度に相関させる差異は 1 つだけにするをことをお勧めします。

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［**ルールの作成**］をクリックします。これは，右クリックして表示されるメニューからも選択できます。VuGen は，ルールが作成されたことを確認するメッセージを発行します。



このルールを表示するには，［記録オプション］（CTRL +F7）を開き，［**相関**］ノードを選択します。［**Correlation Studio**］エントリを拡張し，ルールを選択します。

4 相関を取り消すには，差異を選択して［**Remove Correlation**］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。このパラメータは，空でない，NULL で終わる文字列でなければなりません。境界のパラメータでは，大文字と小文字が区別されます。大文字と小文字の区別をしないようにするには，境界の後に「/IC」を追加します。バイナリ・データを指定するには，境界の後に「/BIN」を指定します。
RB	パラメータまたは動的データの右の境界。このパラメータは，空でない，NULL で終わる文字列でなければなりません。境界のパラメータでは，大文字と小文字が区別されます。大文字と小文字の区別をしないようにするには，境界の後に「/IC」を追加します。バイナリ・データを指定するには，境界の後に「/BIN」を指定します。
RelFrameID	要求された URL を基準にした HTML ページの階層レベル。値は，ALL か数値です。
Search	検索の範囲（区切り文字で区切られたデータを検索する場所）。値は，Headers（ヘッダーだけを検索），Body（ヘッダーでなく本体のデータだけを検索），ALL（本体とヘッダーを検索）のいずれかです。標準の値は ALL です。

ORD	このパラメータは省略可能で，何回目に出現する検索内容かを序数で示します。標準の序数は 1 です。「All」を指定すると，パラメータの値が配列に保存されます。
SaveOffset	見つかった値において，パラメータに保存する部分の文字列の開始位置を示すオフセットです。標準設定は 0 です。オフセットの値は負でない数字でなくてはなりません。
Savelen	パラメータに保存する部分文字列の長さ。部分文字列は，見つかった値においてオフセット位置から始まります。標準設定は -1 で，文字列の末尾までを保存することを示します。
Convert	データに適用する変換方式。 HTML_TO_URL : HTML エンコードされたデータを URL エンコード形式に変換する。 HTML_TO_TEXT : HTML エンコードされたデータをプレーン・テキスト形式に変換する。

スクリプトを手作業で相関させるには，次の手順を実行します。

1 動的データを含むステートメントと，データの境界を示すパターンを探します。詳細については，950 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，948 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，次のガイドラインに従って行います。

- 動的データの場所は，記録されたスクリプト内ではなく，必ず HTML コード内で探すようにします。
- 動的データのすぐ左側にある文字列を特定します。この文字列は動的データの左の境界を示します。
- 動的データのすぐ右側にある文字列を特定します。この文字列は動的データの右の境界を示します。

- **web_reg_save_param** は，指定された境界の間（境界を含まない）にある文字を検索し，左の境界の 1 バイト後から，右の境界の 1 バイト前までの情報を保存します。**web_reg_save_param** は，境界文字の重複をサポートしません。たとえば，入力バッファが {a{b{c} で，「{」が左の境界，「}」が右の境界の場合，検索に一致するのは「c」で，それ以外に一致するものはありません。この場合，左右の境界は検出されましたが，境界の重複は認識されないので，「c」が唯一の一致項目となります。

  標準では，境界文字列は最大 256 文字です。最大文字数を増やすには，スクリプトに **web_set_max_html_param_len** 関数を挿入します。たとえば，次の関数は最大文字数を 1024 文字に増やします。

  ```
  web_set_max_html_param_len(“1024”);
  ```

第 59 章

XML ページのテスト

VuGen の Web 仮想ユーザは XML コードを含む Web ページをサポートします。

本章では，次の項目について説明します。

- XML 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。

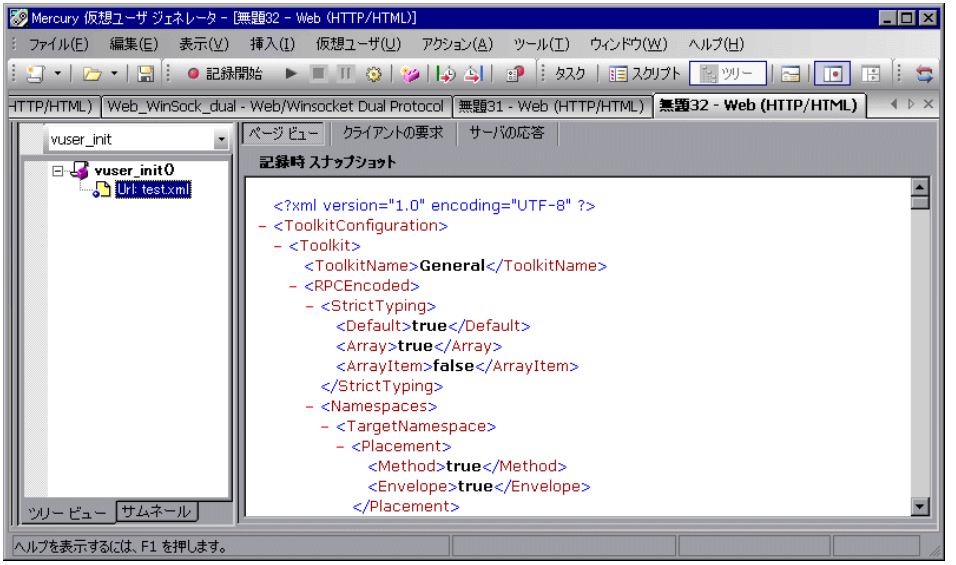
XML コードは，スクリプトに通常の URL ステップまたはユーザ定義の要求として現われます。VuGen は XTML を検出し，ユーザがそれぞれの文書型定義（DTD）およびそのエンティティと属性を参照できるようにします。MIME タイプが **RecContentType** 属性に表示された場合，あるいは再生中に返された MIME タイプが **xml**（**application/xml** または **text/xml**）で終了した場合に XML を解釈します。DTD は色分けされているので，各要素を識別できます。DTD のツリー・ビューの分岐の展開と折りたたみも可能です。

DTD を展開する場合には，属性の値をパラメータ化できます。また標準の相関関数を使って相関できるように値を保存できます。相関関数の詳細については，「**オンライン関数リファレンス**」（[**ヘルプ**] > [**関数リファレンス**]）を参照してください。

注 : HTML ページに埋め込まれた断片的な XML である **XML アイランド**を含む DTD は表示できません。VuGen は，全体が XML であるページのみを表示します。

XML の URL ステップとしての表示

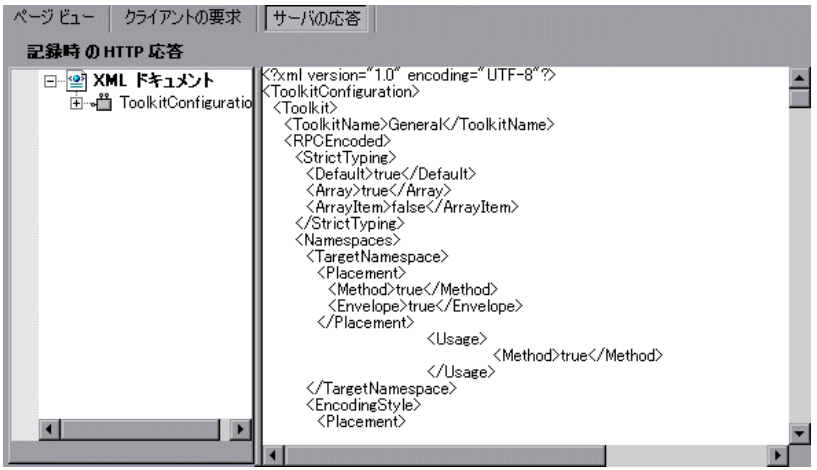
XML コードを含むページをテストする 1 つの方法は，VuGen で記録することです。まず，XML ページを通常の Web ページと同じ方法で記録します。VuGen は，DTD およびすべての XML 要素を記録します。XML ページのスナップショットは作成されませんが，XML ステップごとに，XML コードが［サーバの応答］タブのスナップショット・フレームに表示されます。



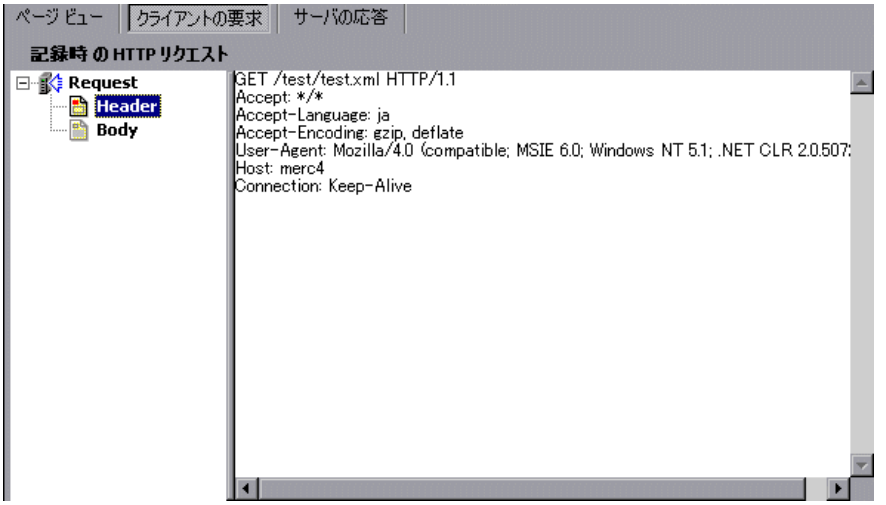
VuGen が作成した展開可能な DTD 階層は，色分けされてスナップショット表示枠に表示されます。項目の分岐を展開するにはプラス（+）記号をクリックし，折りたたむにはマイナス（−）記号をクリックします。XML タグは茶色，値は黒で表示されます。

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 9 章「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。

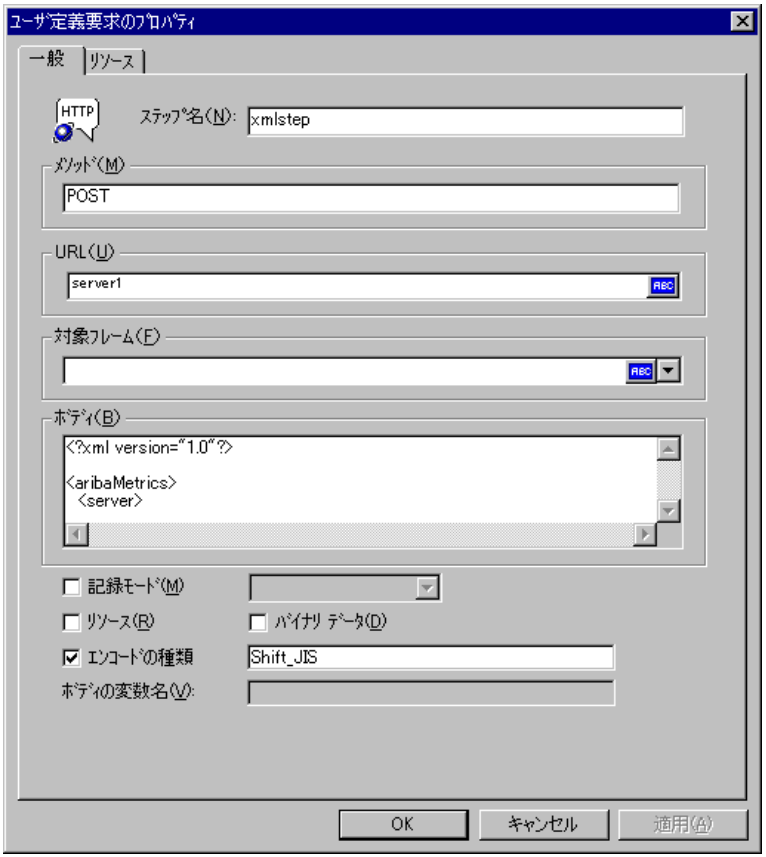
XML コードをユーザ定義の要求として追加するには，次の手順を実行します。

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 56 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。

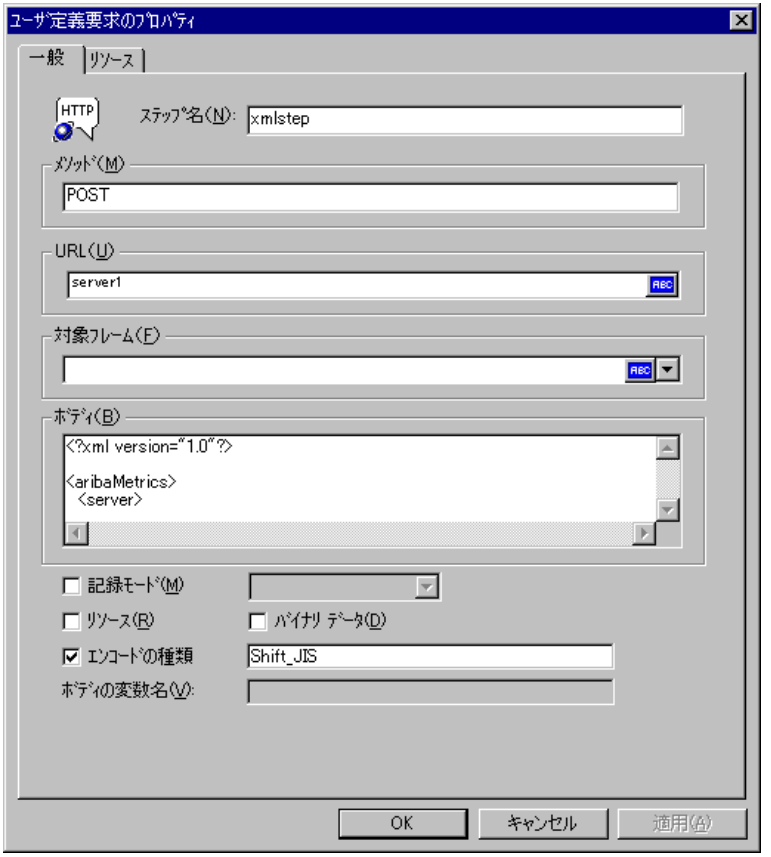
XML ユーザ定義要求のステップの表示

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。

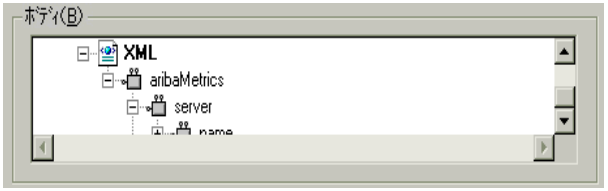
ユーザ定義の要求のステップの XML コードを表示するには，次の手順を実行します。

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。

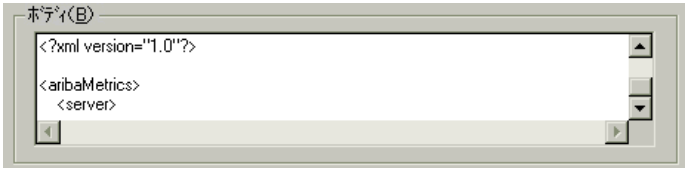
2 右クリック・メニューから［**プロパティ**］を選択します。［ユーザ定義要求のプロパティ］ダイアログ・ボックスが開きます。



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。

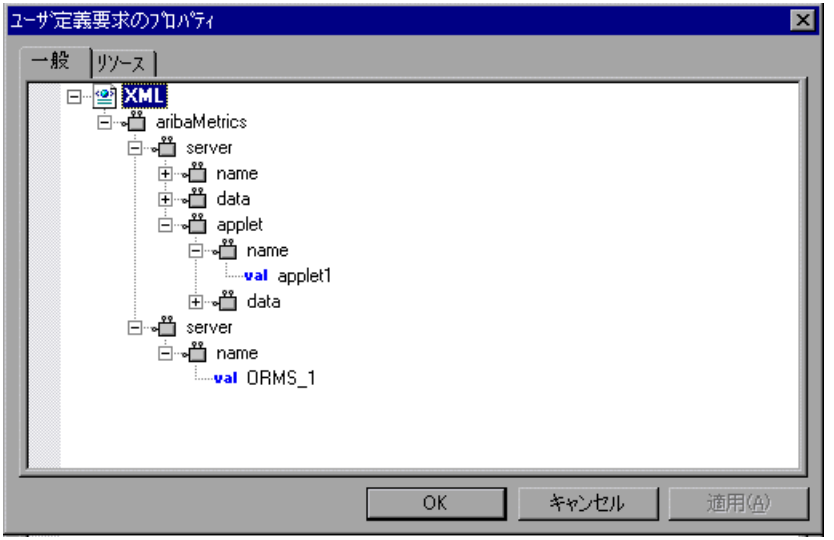


RecContentType 属性が「**text/xml**」以外に設定されているときには，コードはテキスト形式で表示されます。このモードのときには，XML コードは編集可能です。



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 60 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。質問と回答は次のよう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
- パフォーマンス問題

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。たとえば，秘密鍵がエクスポートできないなどの理由で，特定の証明書をダンプすることができない場合，**web_set_certificate_ex** がファイル名なしで生成されます。この場合には，WinInet 再生モードを使用しなければなりません。

- **再生**：**web_set_certificate_ex** が使用されていて，ファイル名の引数がある場合は，ソケット再生でのみ使用でき，ロード・ジェネレータ・マシンでの特別な設定は不要です。**web_set_certificate**，またはファイル名のない **web_set_certificate_ex** が使われている場合には，WinInet ベースの再生でのみ使用可能です。この場合，記録用のマシン上にすべての証明書を，**証明書リストに記載されている順番で**インストールする必要があります。これは，エクスポートとインポートによって行います。

質問 3：SSL サイトにアクセスする仮想ユーザ・スクリプトを記録すると，数多くの警告ポップアップ・メッセージが表示されます。これらのメッセージは表示が必要なのですか？もしそうなら，どのように対処すればよいのでしょうか？

回答：SSL サイトへのアクセスが記録できるように，VuGen ではオリジナルのサーバ証明書に代えて，VuGen 独自のサーバ証明書を提供します。これは次の 2 つのセキュリティ違反になります。

- 発行された証明書はユーザが接続しているサイト用のものではない。
- 発行された署名は未知の認証機関によるものである。

これらのセキュリティ違反があるため，記録用ブラウザによって警告ポップアップ・メッセージが表示されます。

使用しているブラウザが，Netscape 3.0 以上または Internet Explorer 4.0 以上であれば，これらの警告メッセージを無視するオプションがあります。これらのメッセージは無視しても構いません。

注：ポップアップ・メッセージは，スクリプトを記録するときにだけ表示され，スクリプトを実行するときには表示されません。ポップアップ・メッセージの一部（全部ではありません）を表示しないようにできます。

質問 4：IE や Netscape ではない Web アプリケーションを使っています。認知されていない証明書を使ってセキュア・サイトにアクセスすると，アプリケーションは自動的に処理を中断します。このようなアプリケーションを記録することはできますか？

回答：認知されていない証明書でセキュア・サイトにアクセスすると，IE と Netscape は警告を出します。一部のブラウザやアプリケーションは，認知されていない証明書に対して警告を出さずに，単にセキュア・サイトへの接続を遮断します。こ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。この問題を避けるには，Web(Click and Script) 仮想ユーザを使用してみます。

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。

この問題を避けるには，Web(Click and Script) 仮想ユーザを使用してみます。

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**］＞［**選択した仮想ユーザを表示**］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 Web/ Winsocket Dual Protocol を使用して，シングル・プロトコル・スクリプトを作成します。

2 ［**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。

3 ［**アプリケーションの記録開始**］ボタンをクリックします。VuGen は記録されるアプリケーションに必要なプロキシ設定を要求するプロンプトを出します。ホスト名とポート名をメモします。

4 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。

5 ［**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。

6 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。

パフォーマンス問題

質問 34：複数の仮想ユーザが同時に接続を試みることによって過負荷になるのを防ぐにはどうすればよいですか。

回答：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。詳細については，『**LoadRunner コントローラ・ユーザーズ・ガイド**』を参照してください。

第 61 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，次の項目について説明します。

- Web/WinSock 仮想ユーザ・スクリプトの記録について
- Web/WinSock 仮想ユーザ・スクリプトを使った作業の開始
- Web/WinSocket 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/Windows Sockets デュアル・プロトコルおよび 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 57 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 12 章「ステートメントの相関」を参照してください。

Web/WinSock 仮想ユーザ・スクリプトを使った作業の開始

この項では，VuGen を使用したデュアル・プロトコルの Web/WinSock 仮想ユーザ・スクリプトの開発工程の概略を説明します。

Web/WinSock 仮想ユーザ・スクリプトの作成は，次の手順を実行します。

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「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，978 ページ「Web/WinSocket 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 8 章「仮想ユーザ・スクリプトの拡張」を参照してください。

3 **パラメータを定義します（任意）。**

スクリプトに記録された固定値の代わりにパラメータを定義します。固定値をパラメータで置き換えると，同じビジネス・プロセスを異なる値で繰り返し実行できます。

詳細については，第 9 章「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 57 章「Web 仮想ユーザ・スクリプトの相関ルールの設定」と第 12 章「ステートメントの相関」を参照してください。

5 **実行環境を設定します。**

実行環境を設定することによって，スクリプト実行時の仮想ユーザの動作を制御します。設定には，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 13 章「実行環境の設定」を参照してください。

6 **VuGen からスクリプトを実行します。**

VuGen でスクリプトを保存して実行し，正しく動作することを確認します。

詳細については，第 15 章「スタンドアロン・モードでの仮想ユーザ・スクリプトの実行」を参照してください。

スクリプトを作成した後，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** のマニュアル，**Performance Center** のマニュアル，または **Business Availability Center** のマニュアルを参照してください。

Web/WinSocket 記録オプションの設定

スクリプトを記録する前に，Web，WinSock，および Web/WinSock に関連する記録オプションを設定します。

Web に関連する詳細記録オプションおよび相関記録オプションの詳細については，第 51 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。

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。これらの記録オプションの詳細については，第 35 章「WinSock 仮想ユーザ・スクリプトの作成」を参照してください。

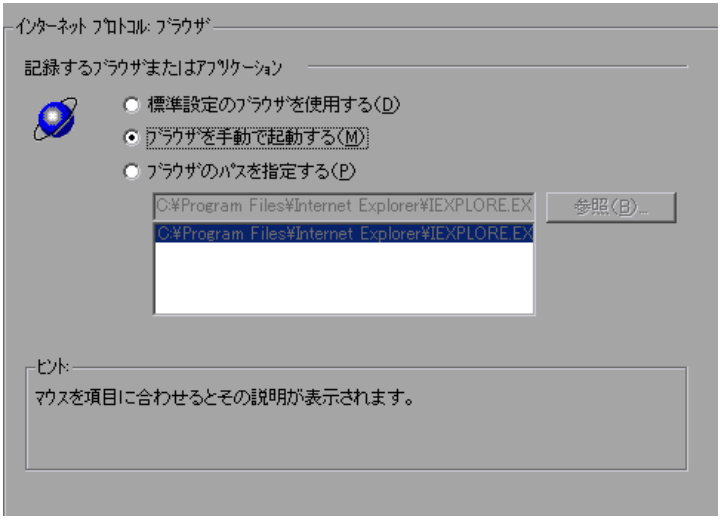
Web/WinSocket 固有の記録オプションを設定するには，次の項のいずれかを参照してください。

- ［ブラウザ］記録オプションの設定
- 記録用プロキシの設定
- ［Web トラップ］記録オプションの設定

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（982 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。

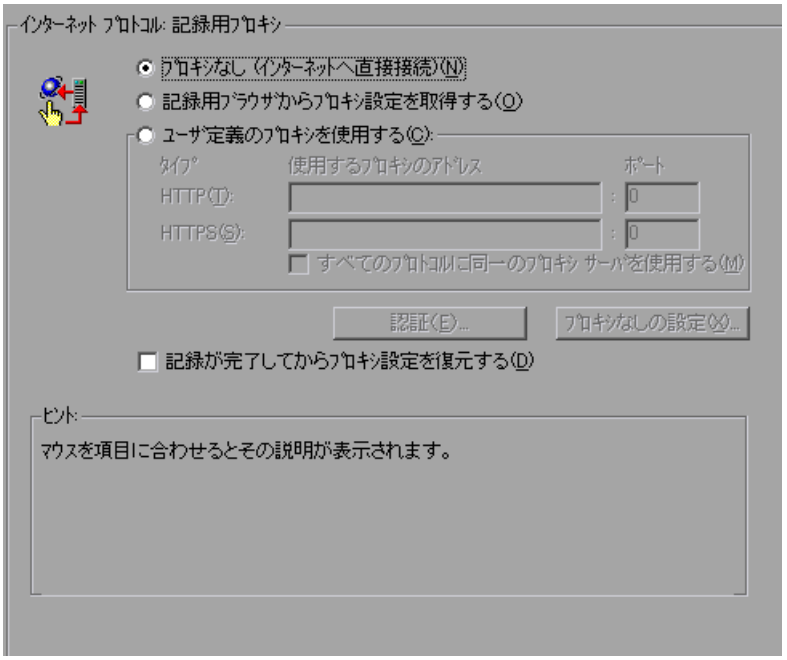
- ［**標準設定のブラウザを使用する**］：記録を行うコンピュータで標準設定の Web ブラウザを使用するよう VuGen に指示する場合に使います。［記録開始］ダイアログ・ボックスの［**記録するアプリケーション**］フィールドのアプリケーションは無視されます。ただし，このフィールドには使用しなくても値を入力する必要があります。ActiveX アプリケーションまたは Java テンプレートを記録するには，このオプションを使用します。

- ［**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。

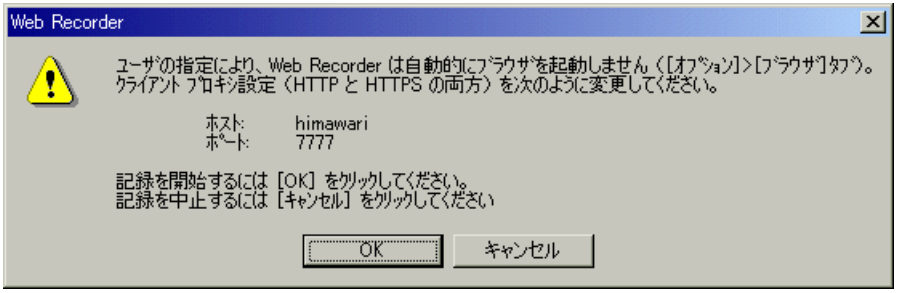
記録用プロキシの設定

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**］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 52 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 35 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，979 ページ「［ブラウザ］記録オプションの設定」を参照してください。

［Web トラップ］の設定では，通常は WinSock 関数として記録されるイベントを，Web 関数としてトラップまたは保存することができます。トラップのオプションを有効にすると，VuGen によって指定のポートでイベントが待機され，それらのイベントを Web イベントと見なし，適切な Web 関数が生成されます。この結果，読みやすく直観的に理解しやすいスクリプトを作成できます。

VuGen が Web イベントをリッスンするポートを指定する必要があります。そのポート上で行われるすべてのやり取りが，Web イベントとして処理され，Web 仮想ユーザ関数で表されます。標準ポート（HTTP の場合は 80，HTTPS の場合は 443）を使用するか，任意の IP とポートの組み合わせ（IP: ポート）を指定することができます。VuGen では，ワイルドカードを組み合わせて使用できるので，特定のホストのすべてのポートを指定することもできます。

たとえば，207.232.15.30:* は，ホスト・マシン 207.232.15.30 上のすべてのポートを示します。207.232.*.*:80 と指定した場合は，ドメイン 207.232 のすべてのマシン上の標準ポート 80 を示します。IP アドレスの 1 つの要素の中で数字とワイルドカードを混在させることはできません。たとえば，207.2*.32.9 という指定は無効です。

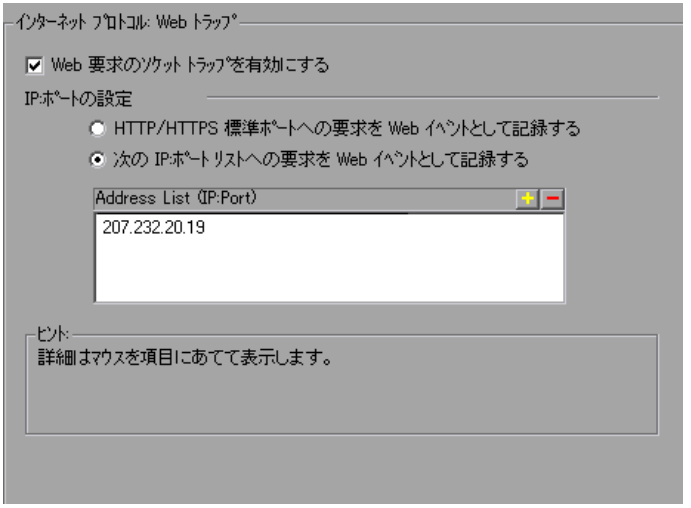
Web トラップを有効にするかどうかを決めるために，まず記録セッションを実行します。データ・ファイル **data.ws** を表示します。WinSock バッファ・データとして記録された HTTP または HTTPS データがある場合，これは別のポートを通じて要求を受け取ったことを示していることがあります。この場合，それらの要求について Web 関数が生成されるように，Web トラップを有効にする必要があります。

このオプションは，ブラウザを使って記録するのではなく，記録するアプリケーションを手作業で起動するときに特に役立ちます。アプリケーションを手作業で起動する方法については，979 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 によって記録される関数の挿入先となるスクリプトのセクションを選択できます。**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 によって空の仮想ユーザ・スクリプトが開き，［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**] をクリックして，記録オプションを変更します。詳細については，978 ページ「Web/WinSocket 記録オプションの設定」を参照してください。

6 [**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。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。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 55 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 [**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。［仮想ユーザを保存］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 8 章「仮想ユーザ・スクリプトの拡張」を参照してください。

スクリプトに変更を加えた後で，スクリプトを記録時の状態に戻す必要が生じた場合には，［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 4 章「VuGen を使った記録」を参照してください。

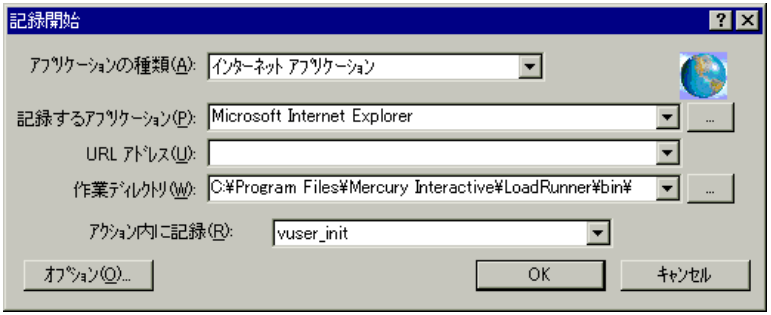
Palm アプリケーションの記録

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［**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。［仮想ユーザを保存］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 10 部

Enterprise Java Bean プロトコル

第 62 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，1008 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・フォルダの完全パスを指定します。
WebSphere Servers 3.x：配備された EJB フォルダの完全パスを指定します。
WebSphere Servers 4.0：アプリケーション・サーバのルート・ディレクトリを指定します。
Oracle OC4J：アプリケーション・サーバのルート・ディレクトリを指定します。
Sun J2EE Server：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完全パスを指定します。

指定しない場合は，クラスパスが検索されます。

リッスン・ポート

EJB Detector のリッスン・ポートです。標準設定のポートは 2001 です。ポート番号を変更したら，［**EJB スクリプトの生成**］ダイアログ・ボックスの［**ホスト**］の［**名前**］ボックスでも指定しなおす必要があります。

たとえば，ホストが「metal」で，標準設定のポートを使用している場合，「**metal**」と指定します。別のポート，たとえばポート 2002 を使用している場合は，「**metal:2002**」と指定します。

EJB Detector をバッチ・ファイルから実行するには，次の手順を実行します。

バッチ・ファイル **EJB_Detector.cmd** を使用して，EJB Detector を起動できます。このファイルは，圧縮ファイル **ejbdetector.jar** を解凍すると，インストールされている EJB Detector のルート・ディレクトリに置かれます。

1 EJB Detector の ルート・ディレクトリで env.cmd を開き，環境に応じて次の変数を変更します。

JAVA_HOME	インストールされている JDK のルート・ディレクトリ
DETECTOR_INS_DIR	インストールされた Detector のルート・ディレクトリ
APP_SERVER_DRIVE	アプリケーション・サーバのインストールをホストするドライブ
APP_SERVER_ROOT	次のガイドラインに従ってください。 **BEA WebLogic Servers 4.x および 5.x**：アプリケーション・サーバのルート・ディレクトリを指定します。 **BEA WebLogic Servers 6.x**：ドメイン・フォルダの完全パスを指定します。 **WebSphere Servers 3.x**：配備された EJB フォルダの完全パスを指定します。 **WebSphere Servers 4.0**：アプリケーション・サーバのルート・ディレクトリを指定します。 **Oracle OC4J**：アプリケーション・サーバのルート・ディレクトリを指定します。 **Sun J2EE Server**：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完全パスを指定します。
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。たとえば，次のように指定します。

```
C:¥>EJB_Detector [listen_port]
```

「listen_port」は，EJB Detector が受信する要求を読み取るポート番号を指定するための任意の引数です（標準設定では「2001」）。

EJB Detector の出力およびログ・ファイル

EJB Detector の出力を調べて，アクティブな EJB をすべて検出したかどうかを確認できます。出力ログには，EJB で検査されたパスが表示されます。検索が終わると，見つかった EJB の名前と場所のリストが表示されます。

たとえば，次のように表示されます。

```
Checking EJB Entry:f:/weblogic/myserver/ejb_basic_beanManaged.jar...
Checking EJB Entry:f:/weblogic/myserver/ejb_basic_statefulSession.jar...
Checking EJB
Entry:f:/weblogic/myserver/ejb_basic_statelessSession.jar...
------------------------ Found 3 EJBs --------------------------
** PATH:f:/weblogic/myserver/ejb_basic_beanManaged.jar
- BEAN:examples.ejb.basic.beanManaged.AccountBean
** PATH:f:/weblogic/myserver/ejb_basic_statefulSession.jar
- BEAN:examples.ejb.basic.statefulSession.TraderBean
** PATH:f:/weblogic/myserver/ejb_basic_statelessSession.jar
- BEAN:examples.ejb.basic.statelessSession.TraderBean
```

EJB が検出されなかった場合（「Found 0 EJBs」と表示されます），EJB jar ファイルが「Checking EJB Entry:...」行に表示されているかどうかを確認してください。リストに表示されていない場合は，「**検索ルート・ディレクトリ**」のパスが正しいかどうか確認します。検査が行われているのにもかかわらず EJB が検出されない場合は，EJB jar ファイルが配備可能か（アプリケーション・サーバに配備できるか）確認します。配備可能な jar ファイルには，Home Interface，Remote Interface，Bean の実装，Deployment Descriptor ファイル（xml ファイルまたは .ser ファイル）およびその他のベンダー固有のファイルが含まれています。

それでも問題が生じる場合は，DETECTOR_HOME¥classes ディレクトリにある **detector.properties** ファイルにデバッグ・プロパティを設定し，さらに詳しいデバッグ情報を取得します。

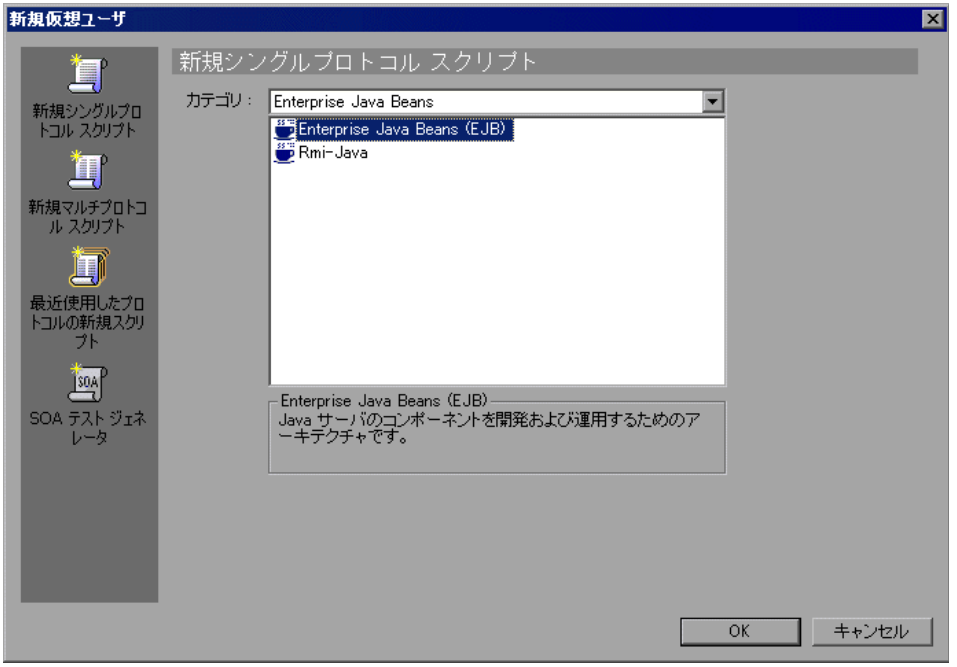
EJB が検出されると，HTTP サーバは初期化されて，VuGen の EJB テスト仮想ユーザからの要求を待機します。この通信過程に問題がある場合は，DETECTOR_HOME¥classes ディレクトリにある **webserver.properties** ファイルで webserver.enableLog プロパティを有効にします。

これにより，**webserver.log** ファイルに詳しいデバッグ情報や，その他の潜在的に重要なエラー・メッセージが出力されるようになります。

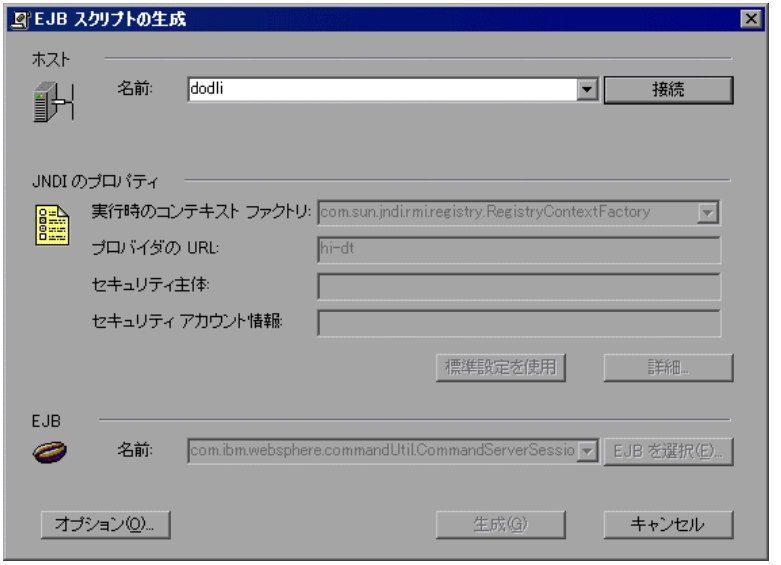
EJB テストの仮想ユーザの作成

EJB 仮想ユーザ・スクリプトを作成するには，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。

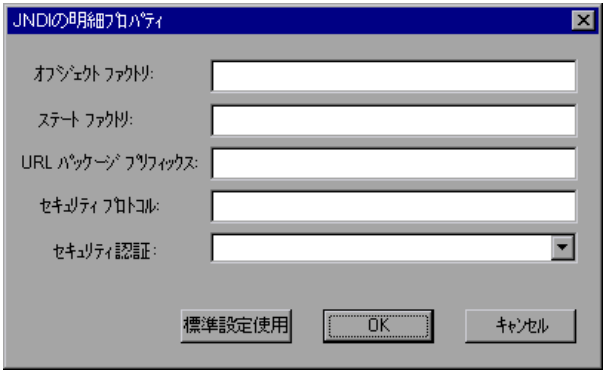
4 EJB Detector は，標準設定の JNDI プロパティを自動的に検出します。これらのプロパティは，編集ボックスで手作業で変更することもできます。変更可能なプロパティは，［**実行時のコンテキスト ファクトリ**］および［**プロバイダの URL**］の文字列です。

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。

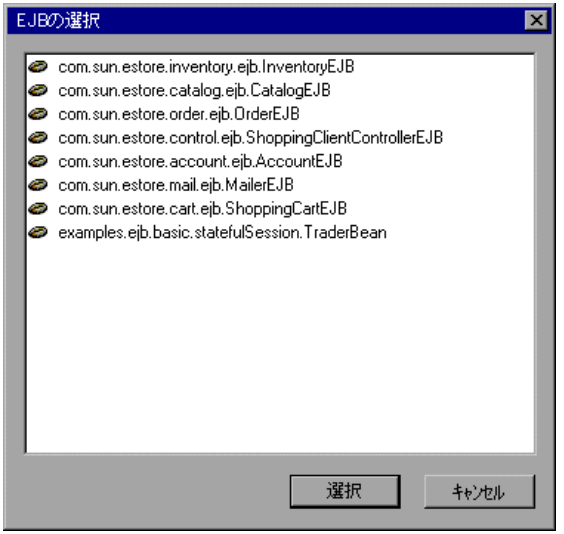
タイプ	実行時のコンテキスト・ファクトリ	プロバイダの URL
WebLogic	weblogic.jndi.WLInitialContextFactory	t3://<appserver_host>:7001
WebSphere 3.x	com.ibm.ejs.ns.jndi.CNInitialContextFactory	iiop://<appserver_host>:900
WebSphere 4.x	com.ibm.websphere.naming.WsnInitialContextFactory	iiop://<appserver_host>:900
Sun J2EE	com.sun.enterprise.naming.SerialInitContextFactory	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。

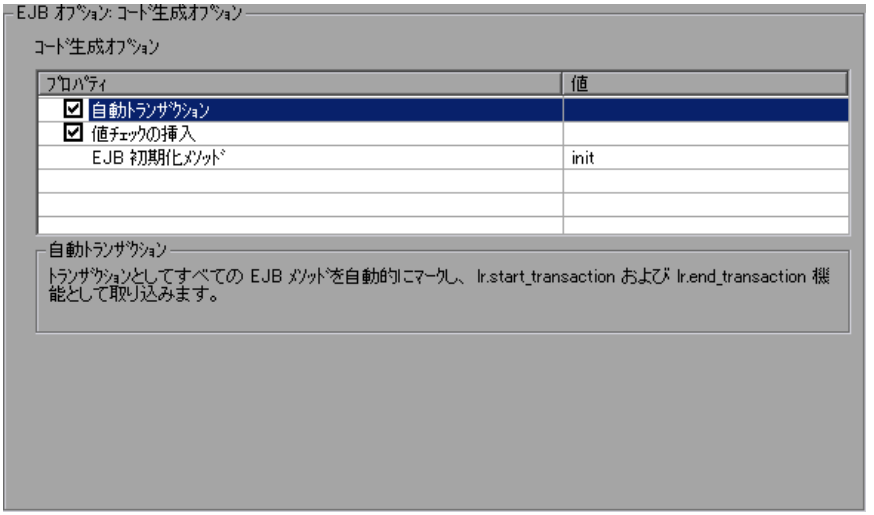
EJB 記録オプションの設定

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 27 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。たとえば，次の行では，プロバイダは EJB Detector のホスト名として dod を指定しています。

p.put(javax.naming.Context.PROVIDER_URL, "t3://dod:7001")

記録されたホスト名をアプリケーション・サーバ名で置き換えます。次に例を示します。

p.put(javax.naming.Context.PROVIDER_URL, "t3://bealogic:7001")

［EJB スクリプトの生成］ダイアログ・ボックスの［**JDNI のプロパティ**］セクションで，記録を開始する前にプロバイダの URL を指定しておくと，手作業で変更する必要がなくなります。

インスタンスの作成

EJB メソッドを実行する前に，スクリプトは，EJB の Bean インスタンスを作成します。インスタンスの作成は，トランザクションとしてマークされるので，スクリプトの実行後に分析できます。さらに，インスタンスの作成プロセスは，例外処理のために **try / catch** ブロックに入れられます。

セッション Beans では，EJB Home「作成」メソッドを使用して新しい EJB インスタンスを作成します。

次の例では，スクリプトは，Carmel EJB のインスタンスを作成します。

```
// Bean インスタンスの作成
Carmel _carmel = null;
try {
        lr.start_transaction("create");
        _carmel = _carmelhome.create();
        lr.end_transaction("create", lr.AUTO);
} catch (Throwable t) {
        lr.end_transaction("create", lr.FAIL);
        t.printStackTrace();
}
```

エンティティ Beans では，findByPrimaryKey メソッドを使用して既存のデータベースで EJB インスタンスを検索します。見つからなかった場合は，create メソッドを使用してその中に作成します。

次の例では，スクリプトは，アカウント EJB のインスタンスを検索し，見つからない場合には作成します。

```
// Bean インスタンスの検索
try {
      com.ibm.ejs.doc.account.AccountKey _accountkey = new
com.ibm.ejs.doc.account.AccountKey();
      _accountkey.accountId = (long)0;

      lr.start_transaction("findByPrimaryKey");
      _account = _accounthome.findByPrimaryKey(_accountkey);
      lr.end_transaction("findByPrimaryKey", lr.AUTO);
} catch (Throwable thr) {

      lr.end_transaction("findByPrimaryKey", lr.FAIL);
      lr.message("Couldn't locate the EJB object using a primary
key.Attempting to manually create the object...["+thr+"]");

      // Bean インスタンスの作成
      try {
            lr.start_transaction("create");
            _account =
_accounthome.create((com.ibm.ejs.doc.account.AccountKey)null);
            lr.end_transaction("create", lr.AUTO);
      } catch (Throwable t) {
            lr.end_transaction("create", lr.FAIL);
            t.printStackTrace();
      }

}
```

エンティティ Bean によって提供される他の 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。たとえば，整数の場合は 0，文字列の場合は空の文字列（""），複雑な Java オブジェクトの場合は NULL となります。必要に応じて，生成されたスクリプト内の標準値を変更します。

```
                _int1 = _carmel.buyTomatoes((int)0);
```

次の例では，パラメータ化を使用した，複雑なタイプの標準値の変更方法を示しています。

```
Detail details = new Details( < city > , < street > , < zip > , < phone >
);
JobProfile job = new  JobProfile( < department > , < position > , <
job_type > );
Employee employee=new Employee( < first > , < last > , details, job, <
salary > );
_int1 = _empbook.addEmployee((Employee)employee);
```

簡単な値や文字列を返すメソッドの場合には，VuGen はコメントアウトされたメソッド **lr.value_check** を挿入します。このメソッドを使用して，EJB メソッドの期待値を指定できます。この検証メソッドを使用するときには，コメントの記号（//）を削除し，期待値を指定します。たとえば，**carmel.buyTomatoes** メソッドは整数を返します。

```
_int1 = _carmel.buyTomatoes((int)0);
//lr.value_check(_int1, 0, lr.EQUALS);
```

メソッドが 500 という値を返すようにするには，コードを次のように変更します。

```
_int1 = _carmel.buyTomatoes((int)0);
lr.value_check(_int1, 500, lr.EQUALS);
```

メソッドが特定の値を返さなかったかどうかを検査する場合は，コードを次のように変更します。

```
_int1 = _carmel.buyTomatoes((int)0);
lr.value_check(_int1, 10, lr.NOT_EQUALS);
```

期待値が検出されない場合は，例外処理が行われ，その情報は出力ウィンドウのログに記録されます。

```
System.err: java.lang.Exception: lr.value_check failed.[Expected:500 Actual:5000]
```

EJB 仮想ユーザ・スクリプトは，標準 Java 規約をすべてサポートしています。たとえば，テキストの前に 2 つのスラッシュ「//」を付けることによって，コメントを挿入できます。

Java 仮想ユーザ・スクリプトは，スケーラブルなマルチ・スレッド・アプリケーションとして稼動します。スクリプトにユーザ定義のクラスを含める場合は，コードをスレッドセーフにする必要があります。コードをスレッドセーフにしないと，結果が不正確になることがあります。スレッドセーフでないコードの場合は，Java 仮想ユーザをプロセスとして実行します。つまり，プロセスごとに個別の Java 仮想マシンを作成します。その結果，スクリプトの拡張性は低くなります。

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，1006 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 26 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 13 章「実行環境の設定」を参照してください。

5 Java VM の実行環境を設定します（追加のクラスパス，VM パラメータをすべて指定します）。必ずアプリケーション・サーバ EJB クラスを入れます。テストされている実際の EJB クラスは仮想ユーザ・ディレクトリに保存され，再生中に自動的に取り出されます。追加のクラスパスの指定，JavaVM 実行環境の設定の詳細については，第 29 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，Crmiorb.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 **Business Availability Center** のマニュアルを参照してください。

第 11 部

ERP/CRM プロトコル

第 63 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・スクリプトの作成について
- Oracle NCA 仮想ユーザの開発の概要
- 記録作業のガイドライン
- 名前によるオブジェクトの記録の有効化
- Personal Home Page からの Oracle Applications の使用
- Oracle NCA 仮想ユーザ関数の使用
- Oracle NCA 仮想ユーザについて
- Oracle NCA 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

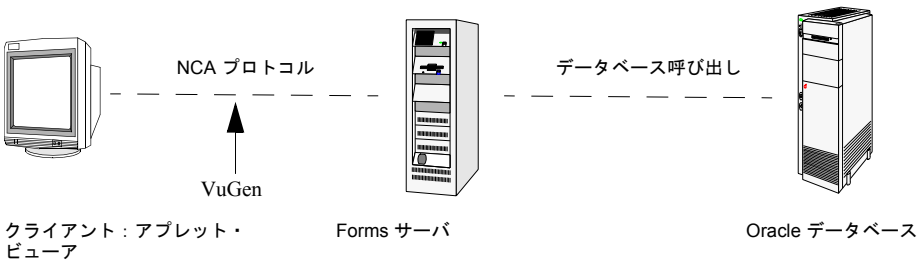
Oracle NCA は，Java ベースのデータベース・プロトコルです。データベース・クライアントであるアプレット・ビューアは，ブラウザを使用して起動します。NCA データベースに対するアクションは，アプレット・ビューアを使用して実行します。

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 4 章「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，1018 ページ「記録作業のガイドライン」と Readme ファイルを参照してください。

2 **スケルトン Oracle NCA 仮想ユーザ・スクリプトを作成します。**

VuGen を使用して，スケルトン Oracle NCA 仮想ユーザ・スクリプトを作成します。詳細については，第 4 章「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 4 章「VuGen を使った記録」を参照してください。

4 **仮想ユーザ・スクリプトを拡張します。**

［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 8 章「仮想ユーザ・スクリプトの拡張」を参照してください。

5 **スクリプトをパラメータ化します。**

記録された定数をパラメータと置き換えます。詳細については，第 9 章「VuGen パラメータを使った作業」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 13 章「実行環境の設定」を参照してください。

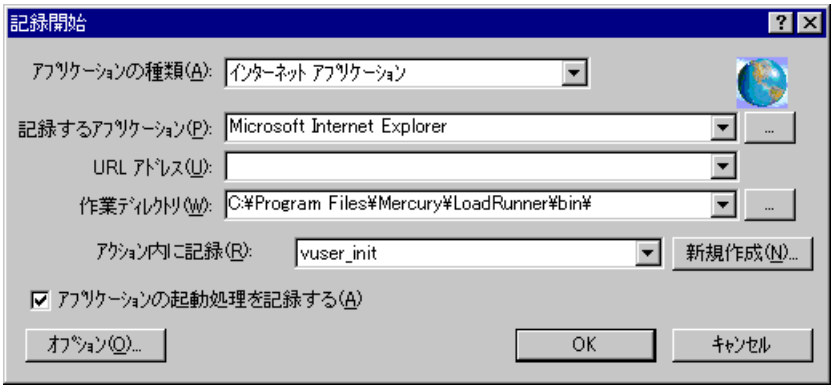
7 **仮想ユーザ・スクリプトを保存して実行します。**

VuGen からスクリプトを実行して，実行時の情報を実行ログで確認します。詳細については，第 15 章「スタンドアロン・モードでの仮想ユーザ・スクリプトの実行」を参照してください。

記録作業のガイドライン

Oracle NCA 仮想ユーザ・スクリプトを記録する際には，次のガイドラインに従います。

➤ Oracle NCA セッションを記録するときに VuGen が使用するブラウザを指定します。［記録開始］ダイアログ・ボックスの［**記録するアプリケーション**］リストで，使用するブラウザを選択します。このリストには，使用可能なブラウザがすべて含まれています。



➤ 記録を開始する前にすべてのブラウザを閉じます。

- **vuser_init** セクションにログイン・プロシージャを記録します。**Actions** セクションに典型的なビジネス・プロセスを記録します。スクリプトを実行するときに，特定のビジネス・プロセスを複数回反復するように指定できます。詳細については，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" >

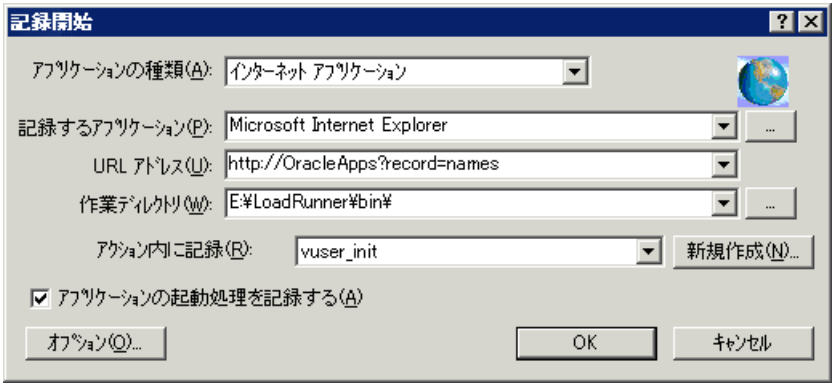
次に，Oracle キー「record=names」を次の形式で追加します。

<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他のファイルを参照しない場合にのみ有効です。

この方法では，［記録開始］ダイアログ・ボックスの URL の後，つまり記録する URL 名の後に「**?record=names**」を追加します。これにより，VuGen はセッションの中でオブジェクト名を記録できるようになります。



Forms 設定ファイル

Forms Web CGI 設定ファイル **formsweb.cfg** を参照するスタートアップ HTML ファイルがアプリケーションにある場合（よく行われる参照です）は，スタートアップ・ファイルに **record=names** を追加すると，問題が生じる場合があります。

この場合は，設定ファイルに変更を加える必要があります。

設定ファイルに変更を加えてオブジェクト名を記録できるようにするには，次の手順を実行します。

1 Forms Web CGI 設定ファイルに移動します。

2 このファイルに新しいパラメータを定義します（以下に示す Web CGI 設定ファイルの例を参照）。

```
serverApp=forecast
serverPort=9001
serverHost=easgdev1.dats.ml.com
connectMode=socket
archive=f60web.jar
archive_ie=f60all.cab
xrecord=names
```

3 スタートアップ HTML ファイルを開き，PARAM NAME="serverArgs" を探します。

4 **record=%xrecord%** のように，変数名を ServerArgs パラメータに引数として追加します。

```
<PARAM NAME="serverArgs" VALUE="module=%form% userid=%use-
rid% %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%

［**Find**］をクリックします。

4 「**ICX:Forms Launcher**」プロファイルに対する User の値を更新します。

- URL に対してパラメータが渡されていない場合，ユーザ指定値の後ろに次の文字列を追加します。

  ?play=&record=names

- URL に対してパラメータが渡されていれば，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 データベースとの接続を解除します。

編集オブジェクト関数

nca_edit_box_press	エディット・ボックスのメッセージをクリックします。
nca_edit_click	エディット・オブジェクト内でクリックします。
nca_edit_get_text	エディット・オブジェクト内のテキストを返します。

nca_edit_press	エディット・フィールド内の参照ボタンを押します。
nca_edit_set	エディット・オブジェクト内のすべての内容を置き換えます。

フレックス・オブジェクト関数

nca_flex_click_cell	［Flexfield］ウィンドウ内のテーブル・セルをクリックします。
nca_flex_get_cell_data	Flexfield セルからデータを取得します。
nca_flex_get_column_name	Flexfield ウィンドウ内のカラムの名前を取得します。
nca_flex_press_clear	［Flexfield］ウィンドウ内の［**Clear**］をクリックします。
nca_flex_press_find	［Flexfield］ウィンドウ内の［**Find**］をクリックします。
nca_flex_press_help	［Flexfield］ウィンドウ内の［**Help**］をクリックします。
nca_flex_press_lov	［Flexfield］ウィンドウ内の［List of Values］ボタンをクリックします。
nca_flex_press_ok	［Flexfield］ウィンドウ内の［**OK**］をクリックします。
nca_flex_set_cell_data	［Flexfield］ウィンドウにデータを挿入します。
nca_flex_set_cell_data_press_ok	データ入力後に［Flexfield］ウィンドウ内の［**OK**］をクリックします。

リスト項目関数

nca_list_activate_item	リスト内で項目を有効にします（ダブルクリック）。
nca_list_select_index_item	インデックスを使ってリスト項目を選択します。
nca_list_select_item	名前を使ってリスト項目を選択します。
nca_lov_auto_select	項目の最初の文字を指定します。

nca_lov_find_value	［List of Values］ウィンドウ内の［**Find**］をクリックします。
nca_lov_get_item_name	エントリのインデックス番号を使って候補値リスト内のエントリの名前を取得します。
nca_lov_retrieve_items	候補値リストを取得します。
nca_lov_select_index_item	インデックス番号を使って候補値リストから項目を選択します。
nca_lov_select_item	候補値リストから項目を選択します。
nca_lov_select_random_item	候補値リストから項目をランダムに選択します。

Java オブジェクト関数

nca_java_action	Java オブジェクトを対象にイベントを実行します。
nca_java_get_value	Java オブジェクトの値を取得します。
nca_java_set_area_class	Java オブジェクトのエリア・クラス・プロパティを設定します。
nca_java_set_reply_property	Java オブジェクトの応答プロパティを設定します。

メニュー・オブジェクト関数

nca_menu_select_item	メニューから項目を選択します。

メッセージ関数

nca_popup_message_press	ポップアップ・ウィンドウ内のボタンをクリックします。
nca_message_box_press	メッセージ・ウィンドウ内のボタンをクリックします。

オブジェクト関数

nca_obj_get_info	オブジェクト・プロパティの値を返します。
nca_obj_mouse_click	オブジェクト内でクリックします。

nca_obj_mouse_dbl_click	オブジェクト内でダブルクリックします。
nca_obj_status	指定したオブジェクトのステータスが返されます。
nca_obj_type	エディット・ボックスに特殊文字を入力します。

応答オブジェクト関数

nca_response_press_lov	［Response］ボックスのドロップダウン・リスト・ボタンをクリックします。
nca_response_press_ok	［Response］ボックス内の［**OK**］をクリックします。
nca_response_set_cell_data	［Response］ボックス内のセルにデータを挿入します。
nca_response_set_data	［Response］ボックスにデータを挿入します。

スクロール・オブジェクト関数

nca_scroll_drag_from_min	最小位置（0）から指定された距離だけスクロール・オブジェクトをドラッグします。
nca_scroll_line	指定された行数だけスクロールします。
nca_scroll_page	指定されたページ数だけスクロールします。

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 8 章「仮想ユーザ・スクリプトの拡張」を参照してください。

Oracle NCA 仮想ユーザについて

Oracle NCA 仮想ユーザ・スクリプトの作成時，VuGen は，クライアントとアプリケーション・サーバ間の NCA 通信をすべて記録します。記録中，VuGen はコンテキスト・センシティブ関数を生成します。コンテキスト・センシティブ関数は，データベースに対するアクションを GUI オブジェクト（ウィンドウ，リスト，ボタンなど）を基準にして表します。記録の実行中，VuGen によってコンテキスト・センシティブ関数が仮想ユーザ・スクリプトに挿入されます。

記録を終えた後で，スクリプト内の関数に変更を加えたり，関数を追加してスクリプトを拡張したりできます。仮想ユーザ・スクリプトの拡張については，第 8 章「仮想ユーザ・スクリプトの拡張」を参照してください。使用可能な Oracle NCA 仮想ユーザ関数の一覧については，1025 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 Opera-
tions");
nca_list_activate_item("FNDSCSGN.NAVIGATOR.LIST.0","+ Journals");
nca_list_activate_item("FNDSCSGN.NAVIGATOR.LIST.0","      Enter");
nca_button_press("GLXJEENT.TOOLBAR.LIST.0");
nca_lov_find_value("Batches","");
nca_lov_retrieve_items("Batches",1,9);
nca_lov_select_item("Batches","AR 1020 Receivables 2537:A 1020");
nca_edit_click("GLXJEENT.FOLDER_QF.BATCH_NAME.0");
…
```

Oracle Configurator アプリケーションを対象に実行するテストなど特定のテストでは，ある関数によって返される情報がセッション全体で必要になります。VuGen は，スクリプトに **web_reg_save_param** 関数を挿入することによって，動的な情報を自動的にパラメータに保存します。次の例では，接続情報が **NCAJServSessionID** という名前のパラメータに保存されています。

```
web_reg_save_param ("NCAJServSessionId", "LB=¥r¥n¥r¥n", "RB=¥r",
        LAST);
web_url("f60servlet",
       "URL=http://usscifforms05.sfb.na/servlet/f60servlet¥?config
       =mult", LAST);
```

上記の例では，右の境界は ¥r です。実際の右の境界は，システムによって異なる場合があります。

Oracle NCA 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，スクリプトが実ユーザを正確にエミュレートするようにできます。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 13 章「実行環境の設定」を参照してください。ネットワークに関係する設定については，第 14 章「ネットワーク実行環境の設定」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。

Oracle NCA 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**］: **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，1040 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。

- ［**Forms のバージョン**］：この設定は，記録を行った後にサーバがアップグレードされた場合にのみ変更します。

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場合にのみ必要です。

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［**実行環境設定の編集**］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ション**］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［**再試行限度**］を指定します。標準設定は 5 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，セキュア Oracle NCA アプリケーションおよびサーブレットをテストするためのヒントをいくつか取り上げます。

セキュア Oracle NCA アプリケーションのテスト

- 記録するプロトコルを選択するとき，プロトコル・リストから **Oracle NCA** だけ選択すればよく，**Web** プロトコルを選択する必要はありません。VuGen は，内部でセキュリティ情報を記録するため，明示的な Web 関数は不要です。
- ［ポートの割り当て］記録オプションで，ポート 443 の既存のエントリを削除して，Oracle サーバ名の新しいエントリを作成します。

  ［**サービス ID**］: HTTP
  ［**対象サーバ**］: Oracle Forms サーバの IP アドレスまたはロング・ホスト名
  ［**ポート**］: 443
  ［**接続の種類**］: SSL
  ［**SSL バージョン**］: 使用している SSL のバージョン。不明な場合は「SSL 2/3」を選択します。

  詳細については，第 7 章「ポートの割り当て設定」を参照してください。
- **nca_connect_server** コマンド実行中に NCA HTTPS スクリプトを再生するときに問題が生じた場合は，スクリプトの先頭に次の関数を挿入します。

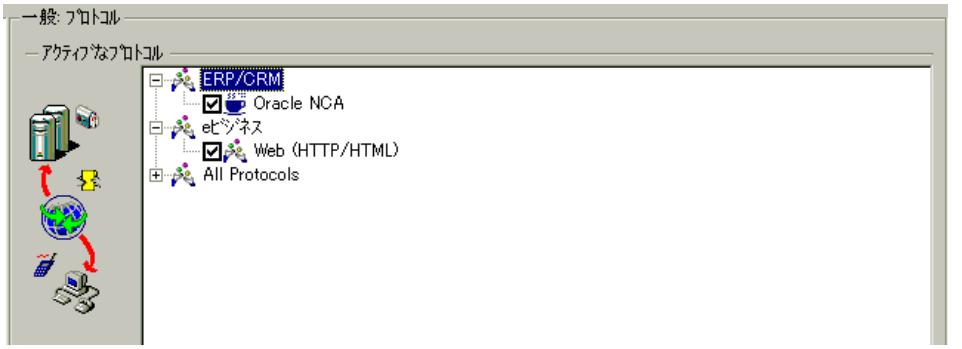
  ```
  web_set_sockets_option("SSL_VERSION","3");
  ```

サーブレットおよびその他の Oracle NCA アプリケーションのテスト

NCA セッションの中には，サーブレットを使用するものがあります。

- Forms Listener サーブレット
- NCA および HTTP 通信の両方を使用するアプリケーションまたはモジュール（Oracle Configurator など）
- NCA アプリケーションの初期化（アプレット，jar ファイル，gif ファイルのダウンロード）

サーブレットを記録するときは，Oracle NCA 関数と Web 関数の両方を記録する必要があります。そのためには，最初にマルチ・プロトコル・スクリプトを作成します。また，Oracle NCA を対象としたシングル・プロトコル・スクリプトを作成した場合は，［記録オプション］で［**一般：プロトコル**］ノードを開き，Web プロトコルを有効にします。これで，記録が開始できます。



アプリケーションでサーブレットが使用されているかどうか不確かな場合は，script ディレクトリの **default.cfg** ファイルを確認します。次のエントリを探します。

UseServletMode=

値が 1 または 2 ならば，サーブレットが使用されています。Oracle NCA に加えて HTTP の記録も有効にする必要があります。

すでにスクリプトが記録されている場合は，Web 関数を含めるようにコードを自動的に再生成できます。再度記録をする必要はありません。［**ツール**］＞［**スクリプトを再生成**］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他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 7 章「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*/,"<server-
Args>");
```

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.run-
forms¥?ICX_TICKET=5843A55058947ED3&RESP_APP=AR&RESP_KE
Y=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.run-
forms¥?ICX_TICKET=<icx_ticket>&RESP_APP=AR&RESP_KEY=REC
EIVABLES_MANAGER&SECGRP_KEY=STANDARD", LAST);
```

注：web_reg_save_param の左右の境界は，アプリケーションの設定によって異なる場合があります。

JServSessionIdroot

JServSessionIdroot 値は，セッション ID を格納するためにアプリケーションによって設定されるクッキーです。ほとんどの場合，この値は VuGen によって自動的に相関され，**web_reg_save_param** 関数が挿入されます。この関数が自動的に追加されなかった場合は，手作業で追加し，値をすべてパラメータ名で置き換えます。

相関させる必要がある値を特定するには，実行ログを開き（[**表示**] ＞ [**出力ウィンドウ**]），応答の本体を探します。

```
vuser_init.c(8):     Set-Cookie: JServSessionIdroot=my1sanw2n1.JS4; path=/¥r¥n
vuser_init.c(8):     Content-Length: 79¥r¥n
vuser_init.c(8):     Content-Type: text/plain¥r¥n
vuser_init.c(8):     ¥r¥n
vuser_init.c(8): 81-byte response body for "http://ABC-
123/servlet/oracle.forms.servlet.ListenerServlet?ifcmd=getinfo&ifhost=mercury&
ifip=123.45.789.12" (RelFrameId=1)
vuser_init.c(8):
/servlet/oracle.forms.servlet.ListenerServlet?JServSessionIdroot=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("NCAJServSes-
sionId","LB=¥r¥n¥r¥n","RB=¥r","ORD=1",LAST);

web_url("f60servlet",
      "URL= http://ABC-"123/servlet/oracle.forms.servlet.ListenerServ-
let?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，1031 ページ「Oracle NCA 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.ListenerServ-
let?JServSessionIdss2ser"
      "vlet=gk5q79uqy1 HTTP/1.1¥r¥n"
      "Pragma:1¥r¥n"
      ...
send buf110
      "POST /ss2servlet/oracle.forms.servlet.ListenerServlet?JServ-
SessionIdss2ser"
      "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.ListenerServ-
let?JServSessionIdss2ser"
      "vlet=gk5q79uqy1 HTTP/1.1¥r¥n"
      "Pragma:-12¥r¥n"
      ...
```

第 64 章

SAPGUI 仮想ユーザ・スクリプトの作成

成長を続けている ERP（Enterprise Resource Planning）の分野において，SAP は企業が自社のすべてのビジネス・プロセスを管理できるようにするソ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 66 章「SAP-Web 仮想ユーザ・スクリプトの作成」を参照してください。本章では，次の項目について説明します。

- SAPGUI 仮想ユーザ・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・スクリプトの作成手順
- 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

以降の情報は 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 6.20 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。

SAPGUI 仮想ユーザのための環境の確認

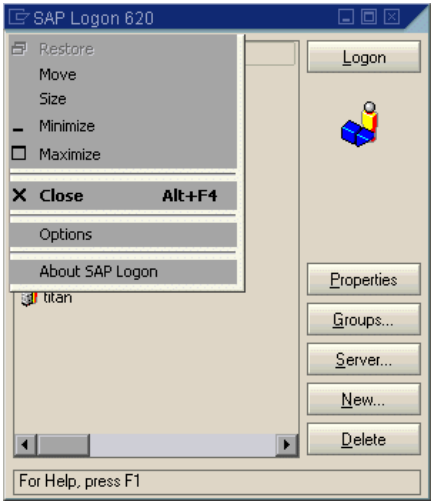
SAPGUI 仮想ユーザを記録するためのシステムの確認および設定の基本的な手順については，「パッチ・レベルの確認」および「スクリプティングの有効化」を参照してください。環境が適切に設定されれば，通常の SAP セッションを記録して VuGen で再生できます。

パッチ・レベルの確認

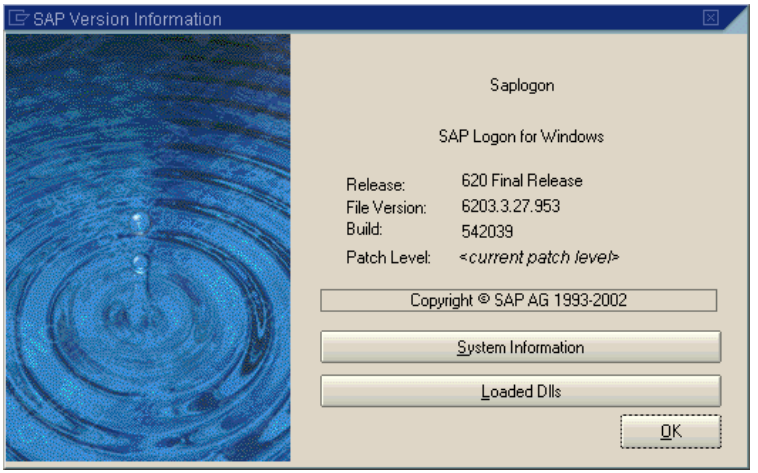
SAPGUI for Windows クライアントのパッチ・レベルは［About］ボックスから確認できます。サポートされている最も低いパッチ・レベルは 32 です。

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 CD2 の **Additional Components¥SAP_Tools¥VerifySAPGUI** フォルダにあります。詳細については，このユーティリティに付属の VerifyScripting.htm を参照してください。

以降の項では，スクリプティングを有効にするのに必要な手順について詳しく説明します。

- 設定の確認
- SAP Application Server でのスクリプティングの有効化
- SAPGUI 6.20 Client でのスクリプティングの有効化

設定の確認

スクリプティングを有効にするには，まず正しいバージョンのカーネルがインストールされていることを確認し，必要に応じてアップデートします。

SAP Application Server のバージョン別に必要な最低限のカーネル・パッチ・レベルを次の表で確認します。必要に応じて，最新のパッチをダウンロードしてインストールします。

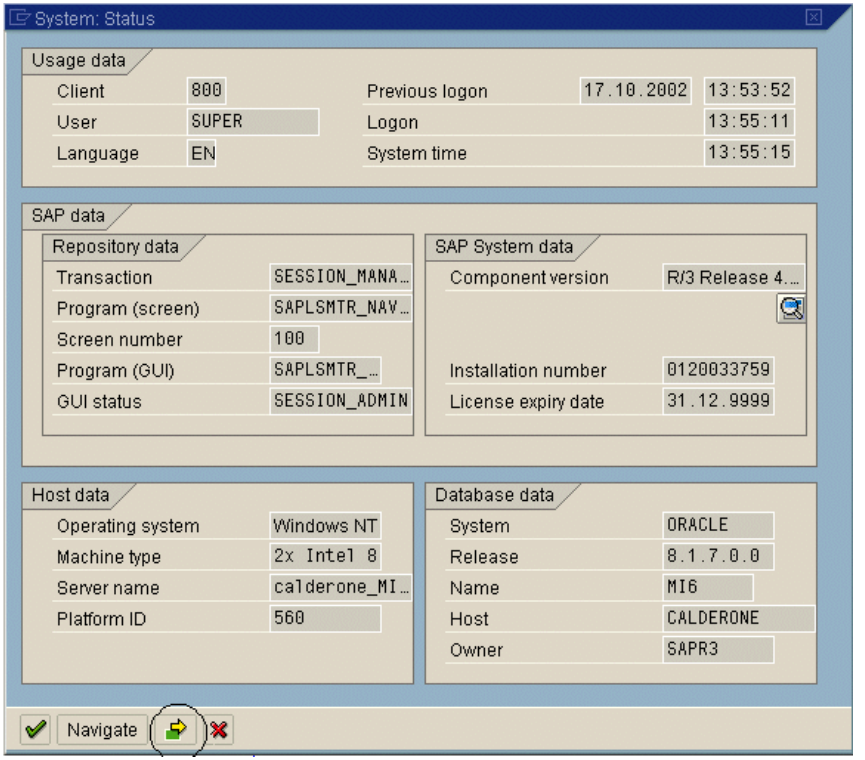
ソフトウェア・コンポーネント	リリース	パッケージ名	カーネル・パッチ・レベル
SAP_APPL	31I	SAPKH31I96	Kernel 3.1I レベル 650
SAP_APPL	40B	SAPKH40B71	Kernel 4.0B レベル 903
SAP_APPL	45B	SAPKH45B49	Kernel 4.5B レベル 753
SAP_BASIS	46B	SAPKB46B37	Kernel 4.6D レベル 948
SAP_BASIS	46C	SAPKB46C29	Kernel 4.6D レベル 948
SAP_BASIS	46D	SAPKB46D17	Kernel 4.6D レベル 948
SAP_BASIS	610	SAPKB61012	Kernel 6.10 レベル 360

カーネル・パッチ・レベルを確認するには，次の手順を実行します。

1 SAP システムにログインします。

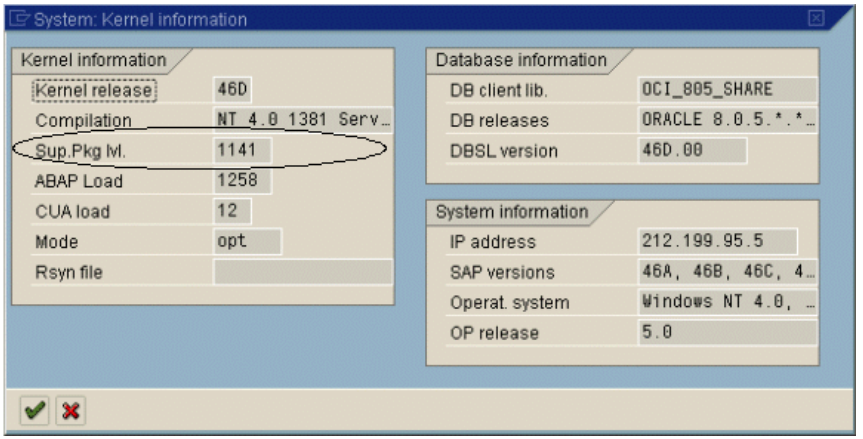
2 ［**System**］ > ［**Status**］を選択します。

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。

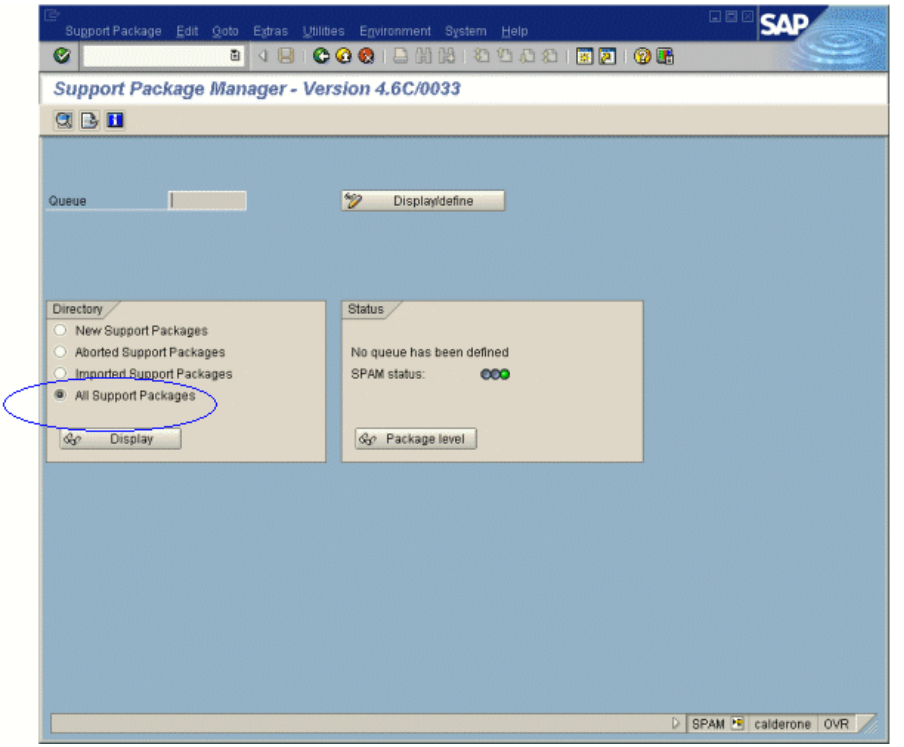
レベルが 948 より低い場合は，最新のバージョンのカーネルをダウンロードして，既存のカーネルをアップグレードする必要があります。このアップグレード方法の詳細については『SAP OSS note #480149』を参照してください。



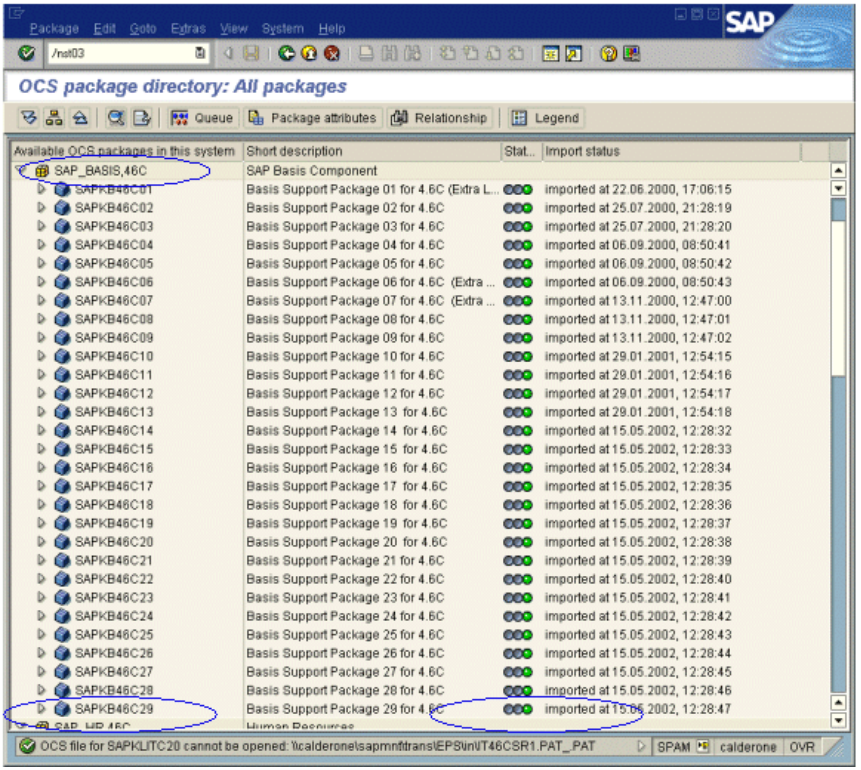
R/3 サポート・パッケージを確認するには，次の手順を実行します。

1 SAP システムにログインし，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。



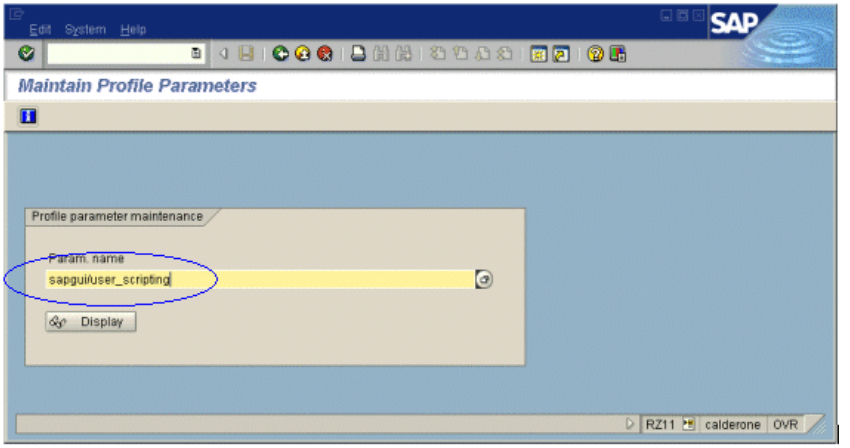
OCS パッケージがインストールされていない場合は，www.sap.com Web サイトからダウンロードしてインストールします。詳細については，『SAP OSS note ＃ 480149』を参照してください。

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。

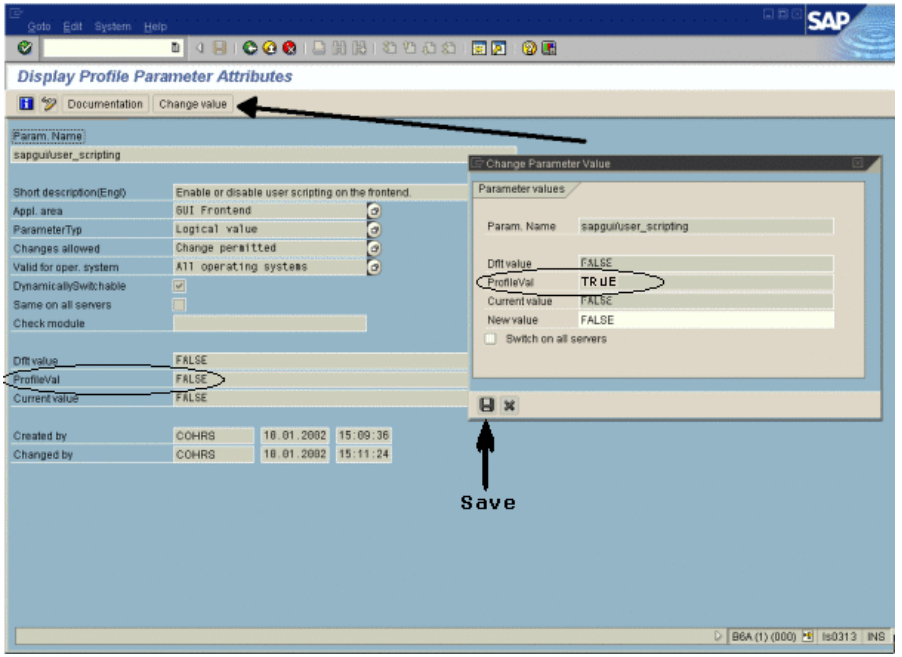
プロファイル・パラメータを変更するには，次の手順を実行します。

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，1047 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，1047 ページ「設定の確認」に記載されています。

Profile Value は，以下のバージョンのカーネルでは，トランザクション rz11 を使用して動的に有効化できます。アプリケーション・サーバを再起動する必要はありません。

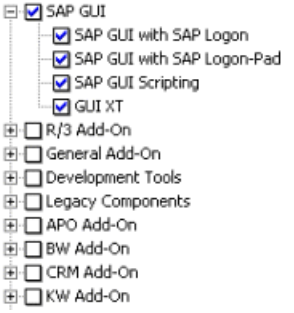
リリース	カーネル・バージョン	パッチ・レベル
4.6B，4.6C，4.6D	4.6D	972
6.10	6.10	391
6.20	すべてのバージョン	すべてのレベル

SAPGUI 6.20 Client でのスクリプティングの有効化

VuGen でスクリプトを実行できるようにするには，SAPGUI クライアントでもスクリプティングを有効にする必要があります。また，接続が確立されたときやスクリプトが GUI プロセスにアタッチされたときなどに表示される特定のメッセージが表示されないようにクライアントを設定する必要もあります。

VuGen で使用できるように SAPGUI クライアントを設定するには，次の手順を実行します。

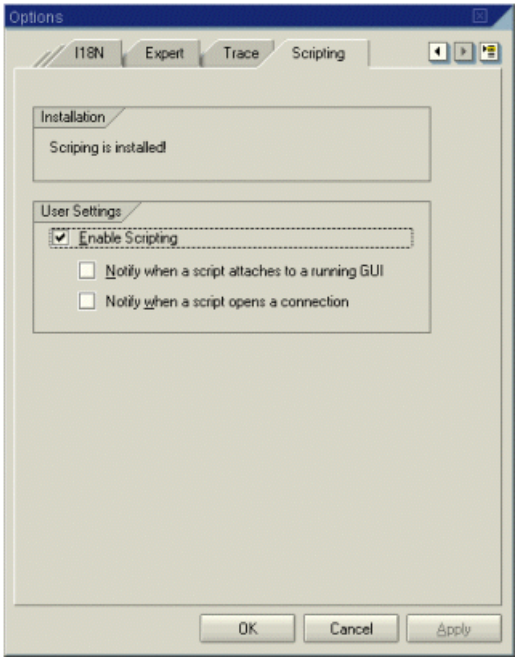
➤ **インストール中**：SAPGUI クライアントのインストール中に，［**SAP GUI Scripting**］オプションを有効にします。



➤ **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・スクリプトの作成手順

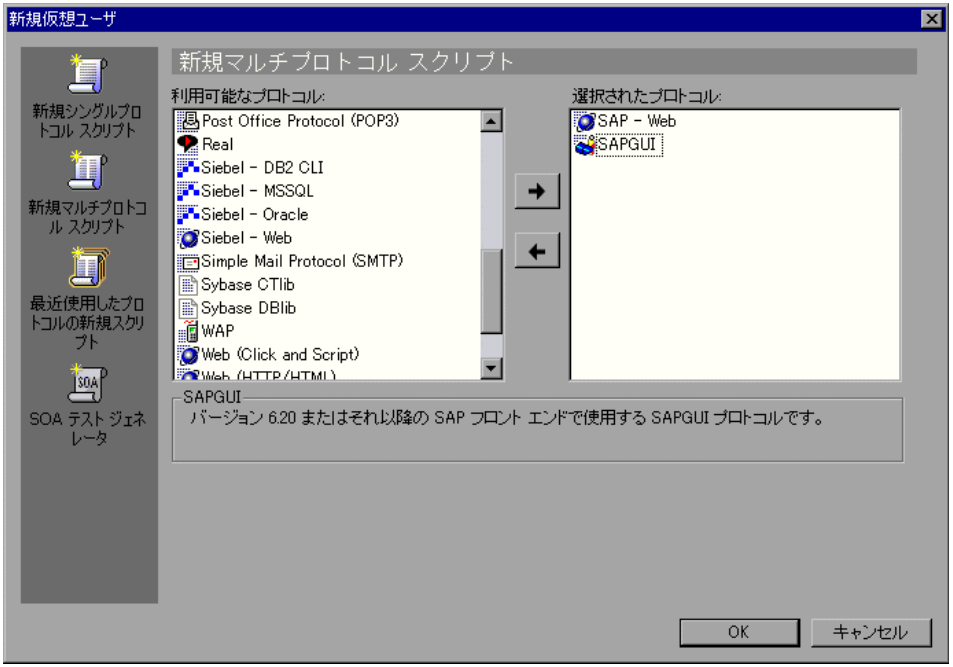
SAPGUI 仮想ユーザ・スクリプト作成の第一歩は，仮想ユーザとスクリプトのタイプを選択することです。SAP の仮想ユーザ・タイプ **SAPGUI** は，**ERP/CRM** カテゴリの下にあります。シングル・プロトコルとマルチ・プロトコルのどちらの仮想ユーザ・スクリプトでも作成できます。

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します。

SAPGUI スクリプトを開始するには，次の手順を実行します。

1 ［記録開始］ダイアログ・ボックスが開かない場合は，［**記録開始**］ボタンをクリックします。［記録開始］ダイアログ・ボックスが開きます。

2 VuGen 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**］: 記録の挿入先セクションを選択します。最初に選択できるセクションは vuser_init，Action1，および vuser_end です。

3 ［**OK**］をクリックして記録を開始します。

カーソル位置での記録

VuGen では，既存のスクリプトにアクションを記録することもできます。いくつかの理由から，既存のスクリプト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。たとえば，SAP サーバは記録セッション中に適用されなかったインベントリ警告を出すことがあります。

これは「カーソル位置での記録」と呼ばれる，新しいアクションの挿入または既存のアクションの置換を行えるようにする機能です。カーソルでの記録を開始すると，VuGen は次の 2 つのオプションの入力を求めるプロンプトを表示します。

- ［**アクションにステップを挿入する**］：既存のステップを一切上書きせずに，新しく記録されたステップを挿入します。新しいセグメントは追加されたセクションの始まりと終わりを表すコメントで囲まれます。このオプションは，記録中には存在しなかった，ときおり現れるポップアップ・ウィンドウの処理に適しています。

```
// Recording at the cursor - 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位置まで再生します。次に［記録］フローティング・ツールバーを表示して，記録を開始します。カーソルでの記録機能を使う場合には，**スクリプトの再生成**ユーティリティが無効になります。

注：カーソル位置で記録するには，既存の関数の直前で VuGen エディタの左余白をクリックする必要があります。

カーソルで記録するには，次の手順を実行します。

1 スクリプト・ビューを開き（[**表示**] > [**スクリプト ビュー**]），既存の関数の横の左余白をクリックします。

2 [**カーソル位置で記録**] ボタンをクリックします。 

選択を求めるプロンプトが表示されます。

3 [**アクションにステップを挿入する**] または [**スクリプトの残りを上書きする**] を選択します。[**OK**] クリックします。VuGen はスクリプトをカーソルの位置まで再生します。

4 [記録] フローティング・ツールバーが開くのを待ちます。SAPGUI クライアント内でアクションを開始し，必要に応じてセクションとアクションを切り替えます。

5 [**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。

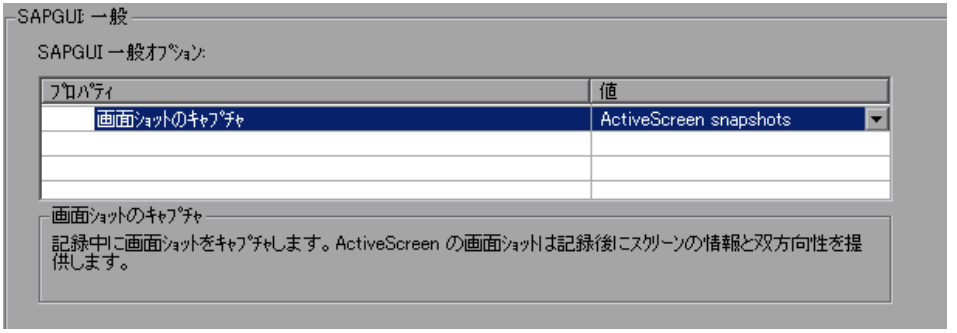
次の項目について記録オプションの設定が可能です。

- 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 51 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。

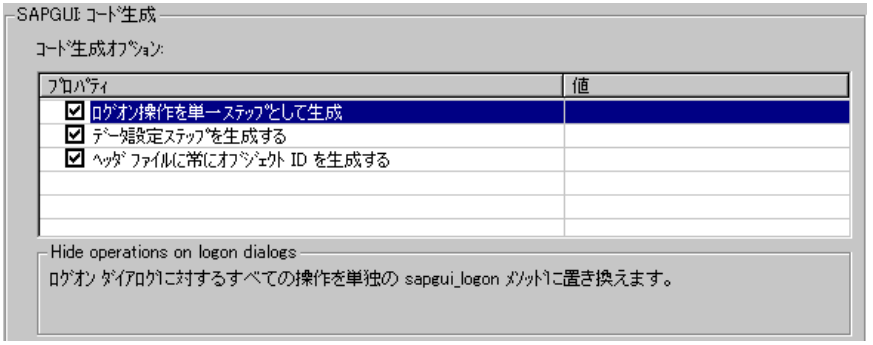
2 ［**画面ショットのキャプチャ**］のために，記録中に表示される SAPGUI 画面のスナップショットの保存方法を指定します。リストから項目を選択します。

［**ActiveScreen snapshots**］，［**Regular snapshots**］，または［**None**］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。

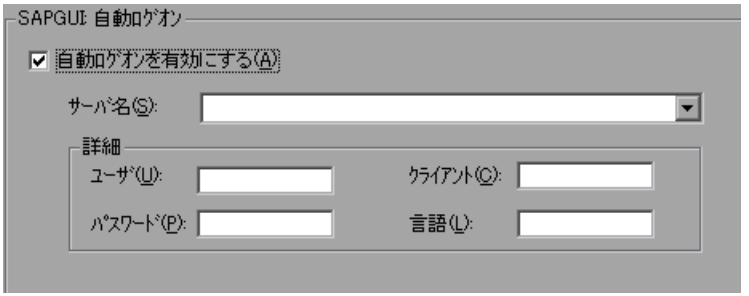
3 各セルに対して個別のデータ挿入ステップを生成するのではなく，テーブルおよびグリッド・コントロールに対してデータ挿入ステップを生成するには，［**データ設定ステップを生成する**］を選択します。

4 よりコンパクトで読みやすいスクリプトを作成するには，［**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。これにより，オブジェクト ID がスクリプトではなく個別のヘッダ・ファイルに置かれるようになります。このオプションを無効にすると，一般スクリプト設定で指定された文字列の長さに従って ID が生成されます。このオプションを無効にしても読みやすくなるだけで，オーバーヘッドに違いはありません。

5 ［**OK**］をクリックして設定を適用し，ダイアログ・ボックスを閉じます。

SAPGUI 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：自動ログオン**] 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。

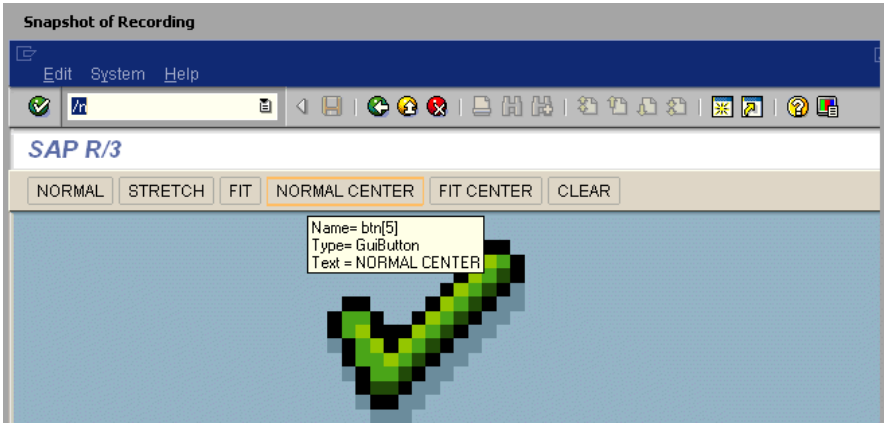
SAPGUI スクリプトのステップの対話的挿入

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，18 ページ「仮想ユーザ・スクリプトの表示と変更」を参照してください。

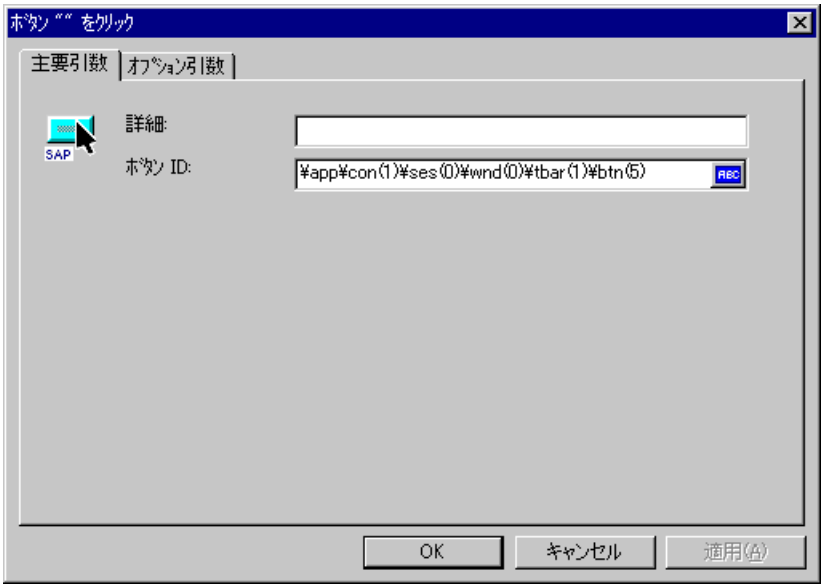
新しい関数を手作業で追加するだけでなく，SAPGUI 仮想ユーザのために，新しいステップをスナップショットから直接，対話形式で追加できます。右クリック・メニューを使用して，ビットマップ関連またはテキスト関連のステップを追加できます。

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［プロパティ］ダイアログ・ボックスの関連フィールドに挿入します。たとえば，上に示すように，NORMAL CENTER ボタンに対して **Press Button** ステップを挿入した場合，［プロパティ］ダイアログ・ボックスには次の ID が表示されます。



特定のオブジェクトにステップを対話的に挿入するには，次の手順を実行します。

1 ［バッファのスナップショット］ウィンドウ内をクリックします。

2 関数を追加するオブジェクト上にマウスを移動します。VuGen がそのオブジェクトを認識し，ボックスで囲んでいることを確認します。

3 右クリックで表示されるメニューから［**新規ステップの挿入**］を選択します。［ステップの追加］ボックスが開きます。

4 メニューからステップを選択します。ステップの「プロパティ」ダイアログ・ボックスが開き，関連するオブジェクトのコントロール 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。

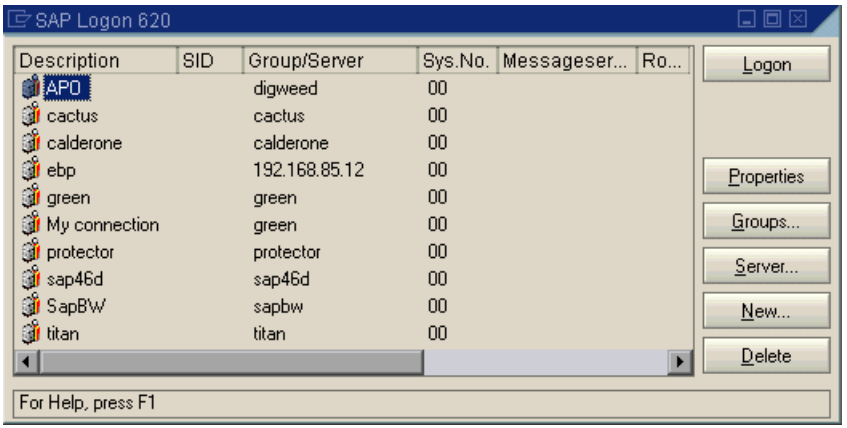
6 特定の場所に貼り付けるオブジェクトのコントロール ID を取得するには，右クリック・メニューから［**コントロール ID のコピー**］を選択します。コントロール ID がクリップボードに置かれます。［スクリプト］ビューから［プロパティ］ボックスまたは直接コードに貼り付けることができます。

SAPGUI 仮想ユーザ・スクリプトについて

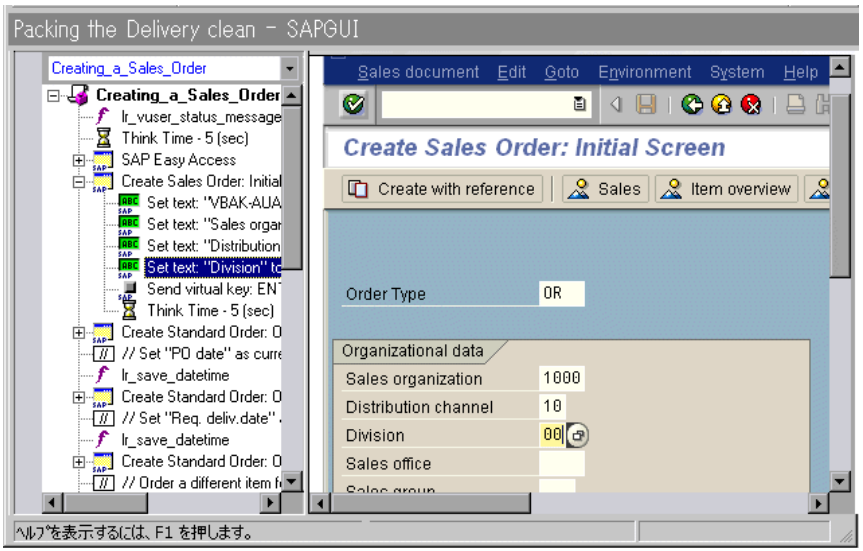
通常 SAPGUI 仮想ユーザ・スクリプトには，ビジネス・プロセスを構成する複数の SAP トランザクションが含まれています。ビジネス・プロセスは，ユーザ・アクションをエミュレートする関数で構成されます。ツリー・ビューを開くと，各ユーザ・アクションが仮想ユーザ・スクリプトのステップとして表示されます。

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。

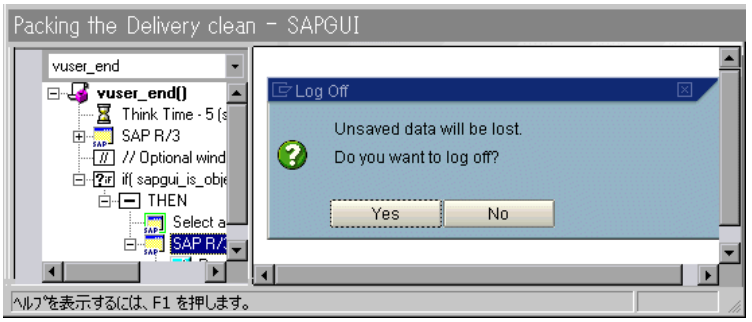
［Open Connection］ステップは，［SAP Logon］の［**Description**］リストにある接続名の 1 つを使用します。指定された名前がリストにない場合，仮想ユーザはその名前のサーバを検索します。



次のセクションでは，関数によって，メニューの選択やチェック・ボックスの設定など，一般的なユーザ操作がエミュレートされます。



最後のセクション **vuser_end** は，ログオフ手順を示しています。



SAPGUI と Web の両方に対してマルチ・プロトコルのスクリプトを記録しているときは，両方のプロトコルに対するステップが VuGen によって生成されます。スクリプト・ビューでは，**sapgui** と **web** の両方の関数を表示できます。

次の例は，SAPGUI クライアントによって Web コントロールが開かれるマルチ・プロトコル記録を示しています。**sapgui** 関数から **web** 関数に切り替わることに注意してください。

```
sapgui_tree_double_click_item("Use as general WWW browser,
REPTITLE",
      "shellcont/shell",
      "000732",
      "REPTITLE",
      BEGIN_OPTIONAL,
           "AdditionalInfo=sapgui1020",
      END_OPTIONAL);


...
sapgui_set_text("",
      "http:¥¥¥¥yahoo.com",
      "usr/txtEDURL",
      BEGIN_OPTIONAL,
           "AdditionalInfo=sapgui1021",
      END_OPTIONAL);


...
web_add_cookie("B=7pt5cisv1p3m2&b=2; DOMAIN=www.yahoo.com");

web_url("yahoo.com",
           "URL=http://yahoo.com/",
           "Resource=0",
           "RecContentType=text/html",
           "Referer=",
           "Snapshot=t1.inf",
           "Mode=HTML",
           EXTRARES,
"URL=http://srd.yahoo.com/hpt1/ni=17/ct=lan/sss=1043752588/t1=10437
52575385/d1=1251/d2=1312/d3=1642/d4=4757/0.4097009487287739/*1"
, "Referer=http://www.yahoo.com/", ENDITEM,
           LAST);
```

SAPGUI 仮想ユーザ・スクリプトの拡張

記録された仮想ユーザ・スクリプトを確認し終わったら，次の方法でそれを拡張します。

- **トランザクション**：トランザクション，ランデブー・ポイント，および制御フロー構造を，スクリプトに挿入します。詳細については，第 8 章「仮想ユーザ・スクリプトの拡張」を参照してください。
- **検証**：SAPGUI の検証関数を挿入し，SAPGUI オブジェクトの現在のステータスを検証します。詳細については，「検証関数の追加」を参照してください。
- **情報の取得**：SAPGUI の関数を挿入し，SAPGUI オブジェクトの現在の値を検証します。情報は **sapgui_get_xxx** 関数を使用して取得します。詳細については，1070 ページ「情報の取得」を参照してください。

パラメータの定義（任意）：仮想ユーザ・スクリプトに記録された固定値に対してパラメータを定義します。固定値をパラメータ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 9 章「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，再生ログには次のように値とパラメータ名が示されます。

```
Action.c(240):Pressed button " Save (Ctrl+S)"
Action.c(248):The type of the status bar is "Success"
Action.c(251):The value of parameter 2 in the status bar is "33232"
```

日付情報の保存

日付を使用するスクリプトを作成すると，正しく動作しないことがあります。たとえば，スクリプトを 6 月 2 日に記録し，それを 6 月 3 日に再生した場合，日付フィールドが正しくなりません。そのため，テキスト実行中に日付をパラメータに保存し，保存した値を他の日付フィールドへの入力として使用する必要があります。スクリプト実行中の現在の日付または時刻を保存するには，**lr_save_datetime** 関数を使用します。この関数を，日付情報を必要とする関数の前に挿入します。日付の形式はロケールに固有です。**lr_save_datetime** 関数の中ではロケールに応じた形式を使用します。たとえば，＜月＞.＜日＞.＜年＞の形式にする場合は，「%m.%d.%Y」と指定します。

次の例では，**lr_save_datetime** で現在の日付を保存します。この値を **sapgui_set_text** 関数で使い，2 日後の配送日を設定します。

```
lr_save_datetime("%d.%m.%Y", DATE_NOW + (2 * ONE_DAY),
    "paramDateTodayPlus2");
sapgui_set_text("Req. deliv.date",
        "{paramDateTodayPlus2}","usr/ctxtRV45A-KETDAT",
        BEGIN_OPTIONAL,
                "AdditionalInfo=sapgui1025",
        END_OPTIONAL);
```

第 65 章

SAPGUI 仮想ユーザ・スクリプトの実行

記録および手作業による機能強化を通して SAPGUI スクリプトを作成したら，VuGen の中で再生してその機能をテストします。本章では，次の項目について説明します。

- SAPGUI 仮想ユーザ・スクリプトの再生について
- 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
- SAPGUI 仮想ユーザ・スクリプトに関するヒント
- SAPGUI 仮想ユーザ・スクリプトのトラブルシューティング
- その他の参考資料

以降の情報は SAPGUI プロトコルと 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・スクリプトの再生について

本章では，SAPGUI 仮想ユーザ・スクリプトの実行について説明します。実行環境を設定して，テストまたは監視セッションの間の仮想ユーザの振る舞いを制御できます。

また本章には，SAPGUI 仮想ユーザを使用した作業のためのいくつかのガイドラインのほか，一般的な問題を解決するためのトラブルシューティングの項があります。

SAPGUI 仮想ユーザ・スクリプトは，**sapgui** という接頭辞で始まる SAPGUI 関数を使用して，一般的なビジネス・プロセスをエミュレートします。

再生中，この関数は SAPGUI オブジェクトでのユーザ・アクティビティをエミュレートします。

たとえば，**sapgui_select_radio_button** によって「**Blue**」ラジオ・ボタンが選択されます。

```
sapgui_select_radio_button("Blue",
        "usr/radRB7",
        BEGIN_OPTIONAL,
           "AdditionalInfo=sapgui1027",
        END_OPTIONAL);
```

SAPGUI のオプションのウィンドウの再生

SAPGUI 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，［**アクティブなウィンドウを選択**］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［**アクティブなウィンドウを選択**］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［アクティブなウィンドウ］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。

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，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，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。これらの設定は仮想ユーザ・スクリプトを実行する前に行います。一般の実行環境と SAPGUI 固有の実行環境の両方を設定できます。

この一般設定には，実行論理，ペース設定，ログ，思考遅延時間，パフォーマンスの設定が含まれます。一般の実行環境の設定については，第 13 章「実行環境の設定」を参照してください。SAPGUI 固有の設定については，以降の項を参照してください。

実行環境の設定が完了したら，仮想ユーザ・スクリプトを保存して VuGen から実行し，正しく動作することを確認します。仮想ユーザ・スクリプトをスタンドアロン・テストとして実行する方法の詳細については，第 15 章「スタンドアロン・モードでの仮想ユーザ・スクリプトの実行」を参照してください。

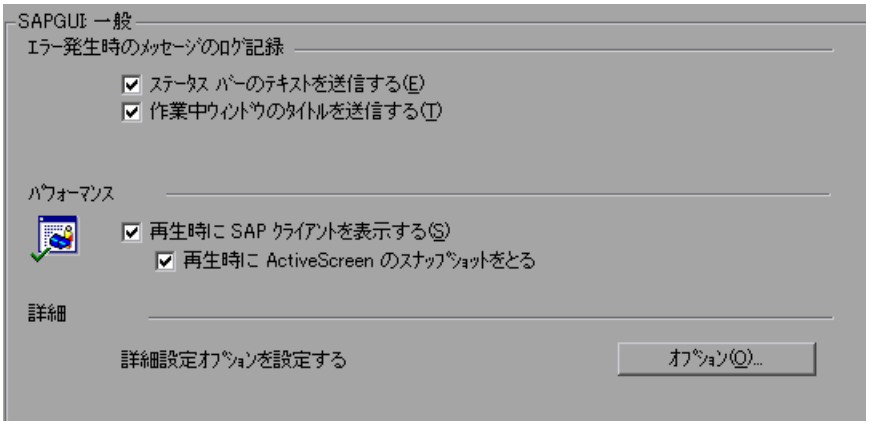
仮想ユーザ・スクリプトが正常に機能することを確認したら，スクリプトを自環境（LoadRunner シナリオ，Performance Center 負荷テスト，Tuning Module セッション，または Business Process Monitor プロファイル）に統合します。詳細については，『**LoadRunner コントローラ・ユーザーズ・ガイド**』，**Tuning Console** のマニュアル，**Performance Center** のマニュアル，または **Business Availability Center** のマニュアルを参照してください。

次の項目について SAPGUI 固有の実行環境を設定できます。

- SAPGUI：実行環境の一般設定
- SAPGUI：実行環境の詳細設定

SAPGUI：実行環境の一般設定

実行環境の一般設定では，SAPGUI 仮想ユーザ・スクリプトの一般設定が行えます。VuGen ではこれらの設定がスクリプトの実行時に使用されます。



［SAPGUI：一般］の［エラー発生時のメッセージのログ記録］では，エラーが発生するたびに仮想ユーザが実行ログに送信する情報を指定します。

- ［**ステータス バー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。

パフォーマンスの実行環境の設定では，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，1063 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，1077 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［**実行環境設定の編集**］ボタンをクリックするか，［**仮想ユーザ**］＞［**実行環境の設定**］を選択します。

2［**SAPGUI：一般**］ノードを選択します。

3［**エラー発生時のメッセージのログ記録**］セクションで，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。

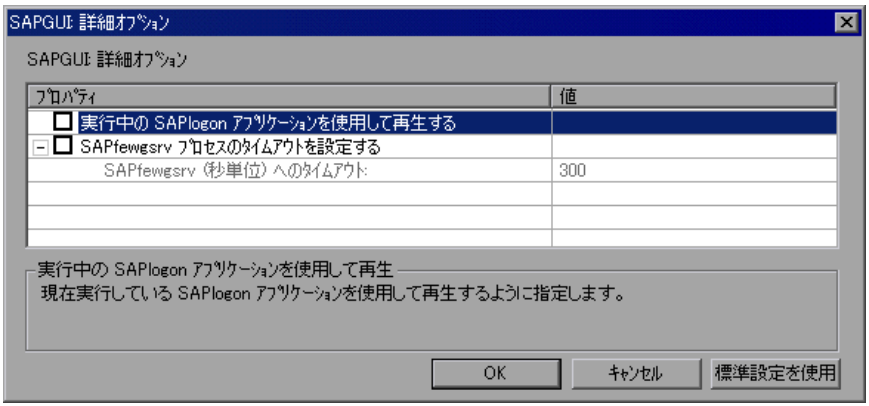
4 再生時に SAP ユーザ・インタフェースを表示するには，［**パフォーマンス**］セクションで［**再生時に SAP クライアントを表示する**］チェック・ボックスを選択します。

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 の詳細設定では，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修正できます。
- ［**SAPfewgsvr へのタイムアウト**］：**SAPfewgsvr.exe** プロセスのタイムアウトを秒単位で指定します。標準設定は 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［**関数の構文を表示**］機能を使用するか，または「**オンライン関数リファレンス**」を参照してください（［**ヘルプ**］＞［**関数リファレンス**］）。

ほとんどの関数は記録されますが，任意の関数をスクリプトに手作業で挿入することもできます。**sapgui_get** で始まるデータ取得関数と，**sapgui_is** で始まる検証用の関数は，記録されません。

sapgui 関数には，接続およびセッション関数，メソッドおよびプロパティ関数，検証およびデータ取得関数，およびオブジェクト関数があります。オブジェクト関数は SAPGUI オブジェクトの内部でアクションを実行する関数で，カレンダー関数，グリッド関数，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。

接続およびセッション関数

sapgui_create_session	新規 SAPGUI セッションを作成します。
sapgui_logon	SAP サーバにログインします。
sapgui_open_connection	SAP サーバへの接続を開きます。
sapgui_open_connection_ex	接続文字列で指定された SAP サーバへの接続を開きます。
sapgui_select_active_connection	指定され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	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	指定されたセルのショートカット・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	カレンダー内の指定された日付までスクロールします。
sapgui_calendar_select_interval	カレンダー内の一定範囲の日付を選択します。

グリッド関数

sapgui_grid_clear_selection	グリッド内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	テーブルのセルから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	グリッド内のチェック・ボックスの状態を確認します。
sapgui_grid_open_context_menu	グリッドで右クリックし，ショートカット・メニューを開きます。
sapgui_grid_press_button	グリッドのセル内のボタンを押します。

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	グリッド内で ENTER キーを押します。
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	グリッド内のツールバーのボタンをクリックします。
sapgui_grid_press_toolbar_context_button	グリッド内のツールバーのコンテキスト・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	グリッド内のセルを選択します。
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	グリッド内のセルを選択します。
sapgui_grid_select_columns	グリッド内のカラムを選択します。
sapgui_grid_select_context_menu	グリッド内でショートカット・メニューを選択します。
sapgui_grid_select_rows	グリッド内の行を選択します。

sapgui_grid_select_toolbar_menu	グリッド内でツールバー・メニューを選択します。
sapgui_grid_set_cell_data	グリッドの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	グリッド内のカラムの順序を設定します。

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	テーブル内のボタンを押します。
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	テーブル内のラジオ・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	テーブル行を選択または選択解除します。
sapgui_table_set_text	テーブルのセルにテキストを挿入します。

ツリー関数

sapgui_tree_click_link	ツリー内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・ノードのチェック・ボックスが選択されいているかどうかを確認します。
sapgui_tree_open_default_context_menu	ツリーの標準設定のショートカット・メニューを開きます。

sapgui_tree_open_header_context_menu	ツリー・ヘッダーのショートカット・メニューを開きます。
sapgui_tree_open_item_context_menu	ツリー項目のショートカット・メニューを開きます。
sapgui_tree_open_node_context_menu	ツリー・ノードのショートカット・メニューを開きます。
sapgui_tree_press_button	ツリー内の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	ツリー内でキーを押します。
sapgui_tree_scroll_to_item	ツリー項目へスクロールします。
sapgui_tree_scroll_to_node	ツリー・ノードへスクロールします。
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	ツリー内のノードを選択します。
sapgui_tree_unselect_all	ツリー内のすべての選択を取り消します。
sapgui_tree_unselect_column	ツリーのカラムの選択を解除します。
sapgui_tree_unselect_node	ツリーの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	ウィンドウを全画面サイズに設定します。
sapgui_window_resize	ウィンドウを指定されたサイズにサイズ変更します。
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・ボックスが選択されているかどうかを確認します。
sapgui_is_tab_selected	タブが選択されているかどうかを確認します。

一般オブジェクト関数

sapgui_htmlviewer_send_event	HTML ビューアにイベントを送信します。
sapgui_select_combobox_entry	リスト・エントリを選択します。
sapgui_press_button	ボタンを押します。
sapgui_select_active_window	指定したウィンドウをアクティブ・ウィンドウとして設定します。
sapgui_select_radio_button	ラジオ・ボタンを選択します。
sapgui_select_tab	タブを選択します。
sapgui_send_vkey	仮想キーを送信します。
sapgui_set_checkbox	チェック・ボックスを選択またはクリアします。
sapgui_set_focus	指定したオブジェクトにフォーカスを設定します。
sapgui_select_menu	指定したメニューを選択します。
sapgui_set_password	パスワード・フィールドのテキストを設定します。
sapgui_set_text	テキスト・ボックスにテキストを挿入します。
sapgui_set_ok_code	［Command］フィールドのテキストを設定します。

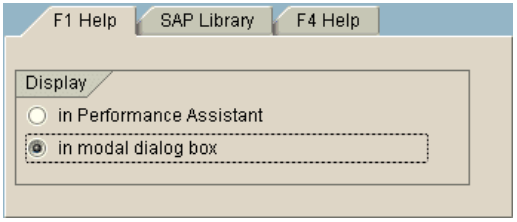
SAPGUI 仮想ユーザ・スクリプトに関するヒント

以降の各項では，SAPGUI 仮想ユーザの記録に関するヒント，再生に関するヒント，およびシナリオ内での再生に関するヒントについて説明します。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。ログオン手順は **vuser_init** セクションに，反復実行するアクションは **Actions** セクションに，ログオフ手順は **vuser_end** セクションに記録します。
- SAPGUI クライアントに Web コントロールが含まれているマルチ・プロトコル・スクリプトを記録する場合は，記録を開始する前に SAPLogon アプリケーションを終了します。
- F1 に対応したモーダル・ダイアログ・ボックスを使用します。F1 ヘルプをモード・ダイアログ・ボックスで開くように SAPGUI に指定します。［**Help**］＞［**Settings**］を選択します。［**F1 Help**］タブをクリックし，［Display］セクションの［**in modal dialog box**］オプションを選択します。



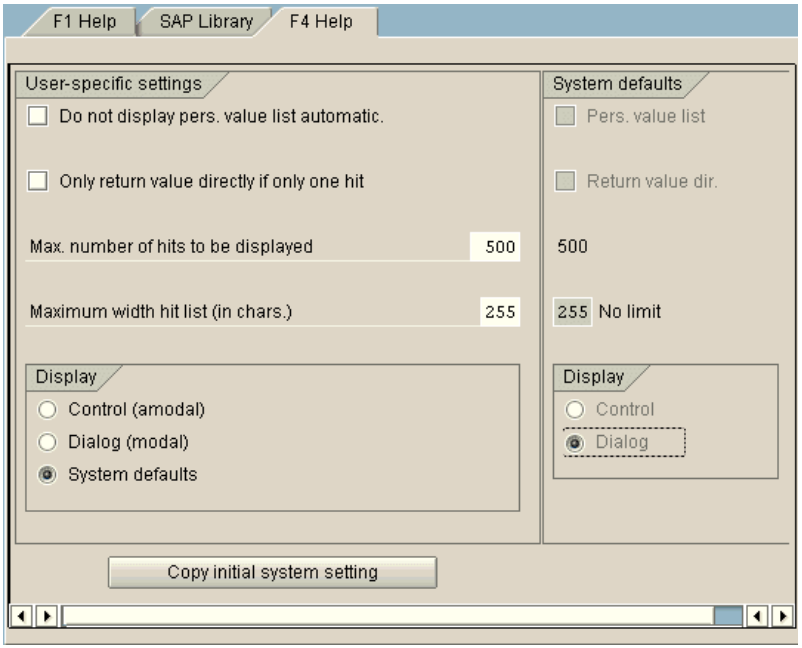
- F4 に対応したモーダル・ダイアログ・ボックスを使用します。F4 ヘルプをモード・ダイアログ・ボックスで開くように SAPGUI に指定します。

  SAP 管理者は次の手順を実行する必要があります。

  F4 ヘルプをモーダル・ダイアログ・ボックスで開くには，次の手順を実行します。

1 サーバからすべてのユーザがログオフしていることを確認してください。

2［**Help**］＞［**Settings**］を選択します。［**F4 Help**］タブをクリックします。



3［Display］セクション（左下）で，［System defaults］を選択します。

4［System defaults］セクションの［Display］部分（右下）で，［**Dialog**］を選択します。

5 変更を保存します。［**Copy initial system setting**］をクリックするか，CTRL キーを押しながら S キーを同時に押します。

6 ステータス・バーに「**Data was saved**」というメッセージが表示されているかどうか確認します。

7 セッションを終了します。

8 SAP Management Console を使って，サービスを再起動します。

再生に関するヒント

スクリプトをスタンドアロン・モードで再生する前に，このガイドラインを読んでください。

- 記録時に生成された **sapgui_logon** 関数の暗号化されたパスワードを，実際のパスワードに置き換えます。次に示すような関数の 2 つ目の引数（ユーザ名の次）がパスワードです：sapgui_logon("user", "pswd", "800", "EN"); セキュリティ向上のため，コード内のパスワードは暗号化できるようになっています。パスワード・テキスト（***** ではなく実際のテキスト）を選び，右クリック・メニューで［**文字列を暗号化**］を選択します。**lr_decrypt** 関数（sapgui_logon("user", lr_decrypt("3ea037b758"), "800", "EN");）がパスワードの位置に挿入されます。
- 初めてスクリプトを実行する場合は，再生時に SAPGUI ユーザ・インタフェースを表示するように VuGen を設定し，その UI を使って実行されている操作を確認できるようにします。再生中にユーザ・インタフェースを表示するには，実行環境の設定を開き，［**SAPGUI：一般**］ノードで［**再生時に SAP クライアントを表示する**］オプションを選択します。複数の仮想ユーザを実行すると UI の表示に多量のシステム・リソースが使用されるので，負荷シナリオの実行中はこのオプションを無効にします。

シナリオ内での再生に関するヒント

以降の各項では，コントローラ，コンソール，およびロード・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。

コントローラとコンソールの設定

LoadRunner シナリオまたはセッション・ステップを対象に作業をしているとき，負荷テストの構成でスクリプトを実行する場合は次の値を設定します。

- **ランプアップ**：（適切なログオンを保証するために）1 つずつスケジューラで設定します。
- **思考遅延時間**：実行環境に乱数の思考遅延時間を設定します。
- **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。

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。

➤ エージェント・モード：プロセス・モードで LoadRunner Remote Agent が実行されていることを確認します。サービス・モードはサポートされていません。

これを調べるには，Windows のタスクバーにあるエージェントのアイコン上にマウスを移動し，説明を読みます。「LoadRunner Agent Service」と説明に表示された場合は，サービスとしてエージェントが実行されています。

プロセスとしてエージェントを再起動するには，次の手順を実行します。

1 エージェントを停止します。LoadRunner Agent のアイコンを右クリックし，［**閉じる**］を選択します。

2 **magentproc.exe** を実行します。これは，LoadRunner または Tuning Module のインストール先の **launch_service¥bin** ディレクトリにあります。

3 次回マシンを起動したときに正しいエージェントが起動されるようにするには，Agent Service の開始方法を［自動］から［手動］に変更します。**magentproc.exe** へのショートカットを Windows の［スタートアップ］フォルダに追加します。

ターミナル・セッション：SAPGUI 仮想ユーザを実行するマシンは，使用できるグラフィック・リソースによっては，実行できる仮想ユーザの数が限られる可能性があります。各マシンの仮想ユーザ数を増やすには，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**］メニューから［**Mercury ＜製品名＞**］プログラム・グループを開き，［**Advanced Settings**］から［**Agent Configuration**］を選択して，［**ターミナル サービスを有効にする**］オプションを選択します。ロード・ジェネレータ・マシンのプロパティで，ターミナル・セッションの数を指定します。詳細については，『**LoadRunner コントローラ・ユーザーズ・ガイド**』の「ターミナル・サービスの設定」を参照してください。

注：LoadRunner エージェントがターミナル・セッションで実行されているときに，ターミナル・セッションのウィンドウが最小化されていると，エラー発生時にスナップショットがキャプチャされません。

SAPGUI 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：プロセス・モードで LoadRunner Remote Agent が実行されていることを確認します。サービス・モードはサポートされていません。詳細については，1091 ページ「再生に関するヒント」を参照してください。

質問 2：特定の SAPGUI コントロールが記録されないのはなぜですか？

回答：一部の SAPGUI コントロールはそのメニューまたはツールバーのコンテキストの中でのみサポートされます。問題のあるタスクについて，メニュー・オプションやショートカット・メニュー，ツールバーなどのさまざまな手段を使用して実行を試みます。

質問 3：VuGen でスクリプトの記録や再生ができないのはなぜですか？

回答：

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。1045 ページ「SAPGUI 仮想ユーザのための環境の確認」を参照してください。

3 SAPGUI for Windows クライアントで通知が無効にされていることを確認します。［Customizing of Local Layout］ボタンをクリックするか，ALT+F12 キーを押します。［**Options**］をクリックして［Scripting］タブを選択します。両方の［**Notify**］オプションをクリアします。

質問 4：スクリプトを実行しようとしたときに表示されるエラー・ポップアップ・メッセージはどのような意味ですか？

回答：SAP アプリケーションの中にはユーザごとに直前のレイアウトを格納するものがあります（どのフレームが表示か非表示か，など）。スクリプトの記録後に，格納されているレイアウトが変更された場合，再生に問題が生じることがあります。たとえば，ME52N トランザクションでは，「Document overview Off/On」ボタンによって，表示されるフレームの数が変わります。

この問題が発生した場合は，次のようにします。

1 再生を開始する前に，トランザクションの記録中と同じ位置に移動します。スナップショット・ビューアを使用すると，トランザクションが記録されたレイアウトを表示できます。

2 スクリプトにステートメントを追加して，再生中にトランザクションが望みのレイアウトになるようにします。たとえば，オプションのフレームによって再生が妨げられる場合は，フレームが開いているかどうかを確かめる検証関数を挿入します。フレームが開いている場合は，ボタンをクリックして閉じます。検証の例については，1069 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 と 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 66 章

SAP-Web 仮想ユーザ・スクリプトの作成

VuGen の SAP-Web 仮想ユーザを使用して，SAP Workplace および SAP Portal クライアントでの作業を記録することができます。

本章では，次の項目について説明します。

- SAP-Web 仮想ユーザ・スクリプトの作成について
- SAP-Web 仮想ユーザ・スクリプトの作成手順
- SAP-Web 記録オプションの設定
- SAP-Web 仮想ユーザ・スクリプトについて
- SAP-Web 仮想ユーザ・スクリプトの再生

以降の情報は，SAP-Web プロトコルにのみ適用されます。

SAP-Web 仮想ユーザ・スクリプトの作成について

まず，VuGen で典型的な SAP ビジネス・プロセスを記録します。VuGen では，ビジネス・プロセス中の SAP Workpalce または SAP Portal のアクティビティを記録し，仮想ユーザ・スクリプトを生成できます。ブラウザ内でアクションを実行すると，このアクティビティを説明する関数が生成されます。各関数の先頭には，**web** という接頭辞が付きます。

再生中，この関数は SAP Workplace または SAP Portal クライアントでのユーザ・アクティビティをエミュレートします。たとえば，次の **web_url** では PageBuilder を開いています。

```
web_url("PageBuilder[myPage]",
"URL=http://sonata.mercury.co.il/hrnp$30001/sonata.mercury.co.il:80/Acti
on/PageBuilder[myPage]?pageName=com.sappor-
tals.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・スクリプトの作成手順

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。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。

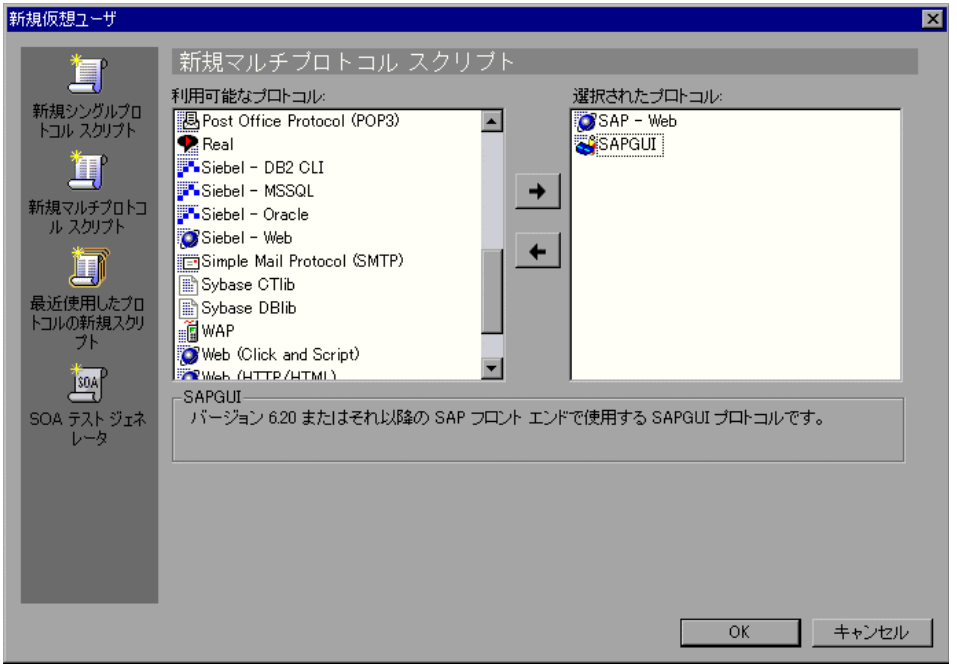
SAP-Web 仮想ユーザを作成するは，次の手順を実行します。

1 VuGen を起動し，［**ファイル**］＞［**新規作成**］を選択します。

2 SAPGUI コントロールが含まれない SAP Workplace または SAP Portal クライアントでのセッションを記録するには，**SAP-Web** 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 Dual Protocol**）
- 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）



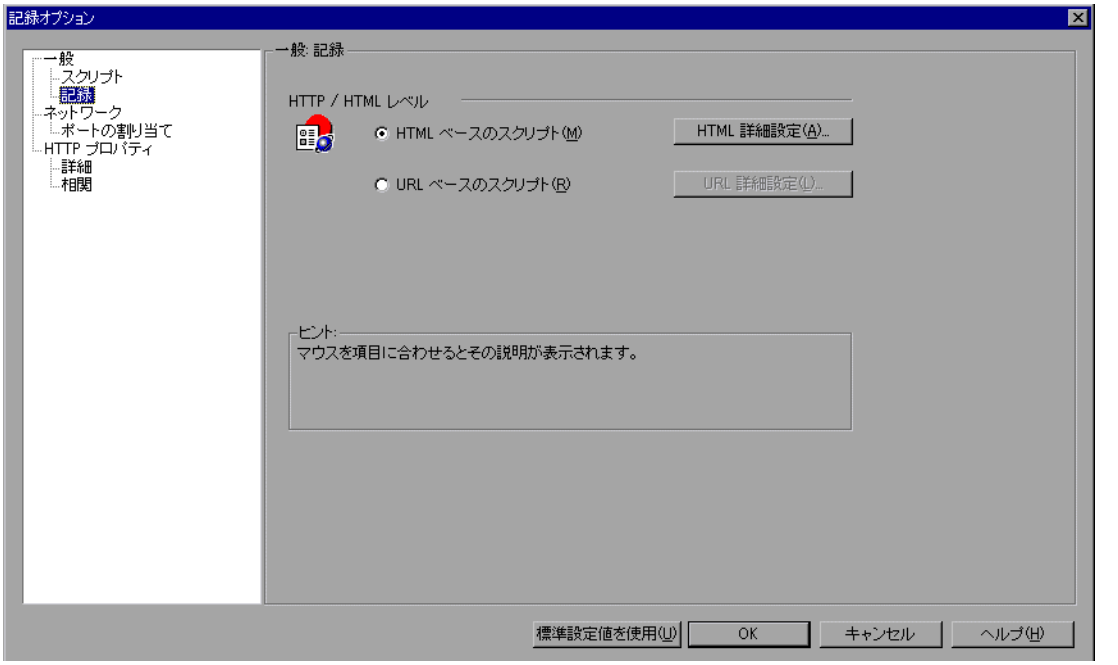
SAP-Web 記録オプションの設定

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。

［**一般：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 51 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。これらの設定は仮想ユーザ・スクリプトを実行する前に行います。実行環境の一般設定と Web 関連の設定が可能です。

一般設定には，実行論理，ペース設定，ログ，思考遅延時間，パフォーマンスの設定が含まれます。一般的な実行環境の設定の詳細については，第 13 章「実行環境の設定」を参照してください。SAP-Web 固有の実行環境の設定については，第 53 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 15 章「スタンドアロン・モードでの仮想ユーザ・スクリプトの実行」を参照してください。

仮想ユーザ・スクリプトが正常に機能することを確認したら，スクリプトを自環境（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 **Business Availability Center** のマニュアルを参照してください。

第 67 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 51 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 53 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，819 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，［**上書き**］をクリックします。詳細については，930 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，1112 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。

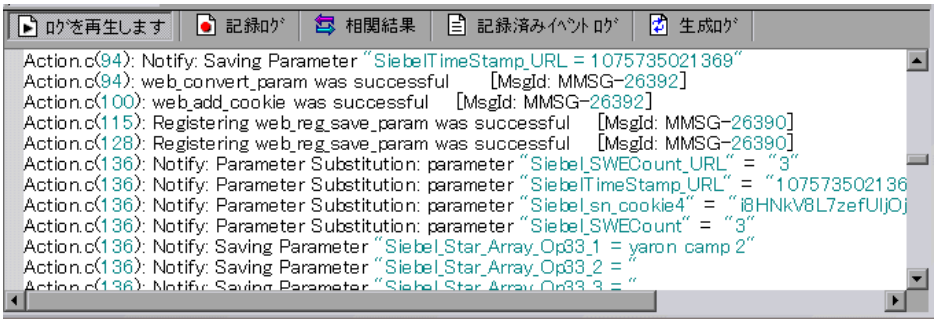
これらのルールは，相関ルールの一覧で参照できます（922 ページ「VuGen の相関ルールの使用」を参照）。これらのルールは，Siebel サーバの文字列に固有の境界条件を示します。

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 57 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（941 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。

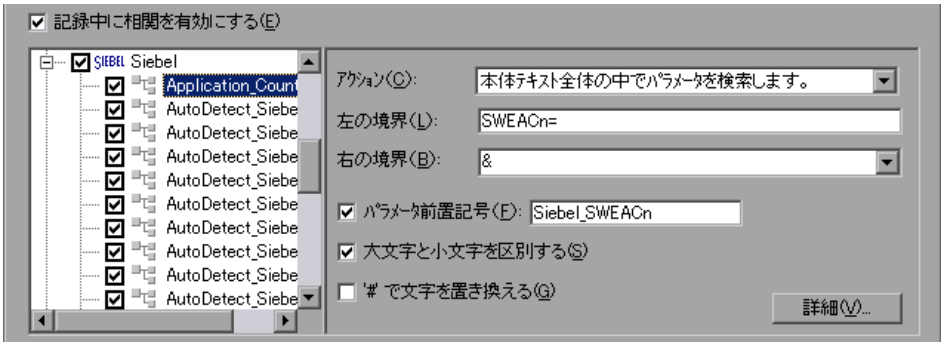
［再生ログ］タブには，VuGen がパラメータを登録，保存，および使用したタイミングが示されます。この情報を表示するには，拡張ログを有効にする必要

があります。詳細については，202 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}&SWE-Frame=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，1113 ページ「SWEC 相関」を参照してください。

ルールの有効化と無効化

通常の状況では，ルールを無効にする必要はありません。しかし，場合によっては，適用しないルールを無効にすることもできます。たとえば，英語専用のアプリケーションをテストするときは，日本語コンテンツのチェック・ルールを無効にします。

ルールを無効にするもう 1 つの理由は，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 桁の小さい数値です。記録されたどの値をパラメータで置換するべきか決めるのが困難なことがしばしばあります。

web_submit_data 関数では，VuGen は SWEC フィールドの数値だけを置換します。

URL では，文字列「SWEC=」または「SWEC`」の後に現れる場合にかぎり，この値を置換します。

すべての SWECount 相関のパラメータ名は同じです。

行 ID 長

場合によっては，**rowid** の前に，その長さが 16 進形式でエンコードされて置かれます。この長さは変更される可能性があるため，その値を相関させる必要があります。

たとえば，xxx6_1-4ABCyyy という文字列は長さの値と RowID で構成されています。ここで 6 は長さ，1-4ABC は RowID です。

文字列を相関させるパラメータを，詳細相関を使用して

xxx{rowid_Length}_{rowid}yyy

と定義した場合，VuGen は文字列の前に次の関数を生成します。

web_save_param_length("rowid", LAST);

この関数は **rowid** の値を取得し，その長さをパラメータ **rowid_length** に 16 進形式で保存します。

SWETS（タイム・スタンプ）

スクリプト内の SWETS の値は，1970 年 1 月 1 日午前 0 時を基点として経過したミリ秒数です。

VuGen は，スクリプト内にある空でないすべてのタイム・スタンプを，パラメータ {SiebelTimeStamp} に置き換えます。このパラメータに値を保存する前に，VuGen は次の関数を生成します。

web_save_timestamp_param("SiebelTimeStamp", LAST);

この関数によって，現在のタイム・スタンプが **SiebelTimeStamp** パラメータに保存されます。

Siebel-Web 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 8 章「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 12 部

レガシ・プロトコル

第 68 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。たとえば，あるメンテナンス会社の顧客情報を管理するサーバがあるとします。フィールド・サービス担当者は，修理を行うたびにターミナル・エミュレータを使ってモデム経由でサーバのデータベースにアクセスします。サービス担当者は顧客の情報にアクセスし，自分が行った修理の詳細を記録します。

RTE 仮想ユーザを使用して，この事例をエミュレートできます。RTE 仮想ユーザは次の操作を実行します。

1 コマンド・ラインで「**60**」と入力して，アプリケーション・プログラムを開きます。

2 フィールド・サービス担当者の番号として「**F296**」と入力します。

3 顧客番号として「**NY270**」と入力します。

4 画面上に 「詳細」という単語が表示されるまで待機します。「詳細」の表示は，画面上に顧客の詳細がすべて表示されたことを示します。

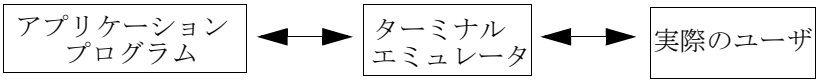
5 最新の修理の詳細として「**ガスケット P249 を交換，主要サービスを実施**」と入力します。

6 「**Q**」と入力してアプリケーション・プログラムを閉じます。

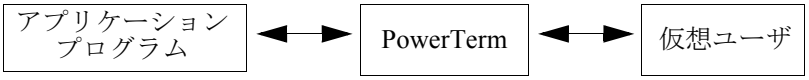
RTE 仮想ユーザ・スクリプトを作成するには，VuGen を使用します。スクリプト・ジェネレータは，実際のユーザがターミナル・エミュレータで行ったアクションを記録します。ターミナル・ウィンドウでのキーボード入力を記録し，対応するステートメントを生成し，それらを仮想ユーザ・スクリプトに挿入します。記録の実行中，スクリプト・ジェネレータはスクリプトに同期化関数を自動的に挿入します。詳細については，第 71 章「RTE 仮想ユーザ・スクリプトの同期化」を参照してください。

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要を説明します。

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 を使っ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 69 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 8 章「仮想ユーザ・スクリプトの拡張」を参照してください。

3 **パラメータを定義します（任意）。**

スクリプトに記録された固定値の代わりにパラメータを定義します。固定値をパラメータで置き換えることにより，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 9 章「VuGen パラメータを使った作業」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 13 章「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 15 章「スタンドアロン・モードでの仮想ユーザ・スクリプトの実行」を参照してください。

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	カーソルがターミナル・ウィンドウ内の指定した位置に出現するまで待ちます。
TE_wait_silent	クライアント・アプリケーションが指定した秒数の間無動作になるまで待ちます。
TE_wait_sync	システムが X SYSTEM（入力禁止）モードから戻るまで待機します。
TE_wait_sync_transaction	システムが最新の X SYSTEM モードを維持していた時間を記録します。
TE_wait_text	文字列が指定した位置に出現するまで待ちます。

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 9 章「VuGen パラメータを使った作業」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 69 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 設定オプションの設定
- 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の設定

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。

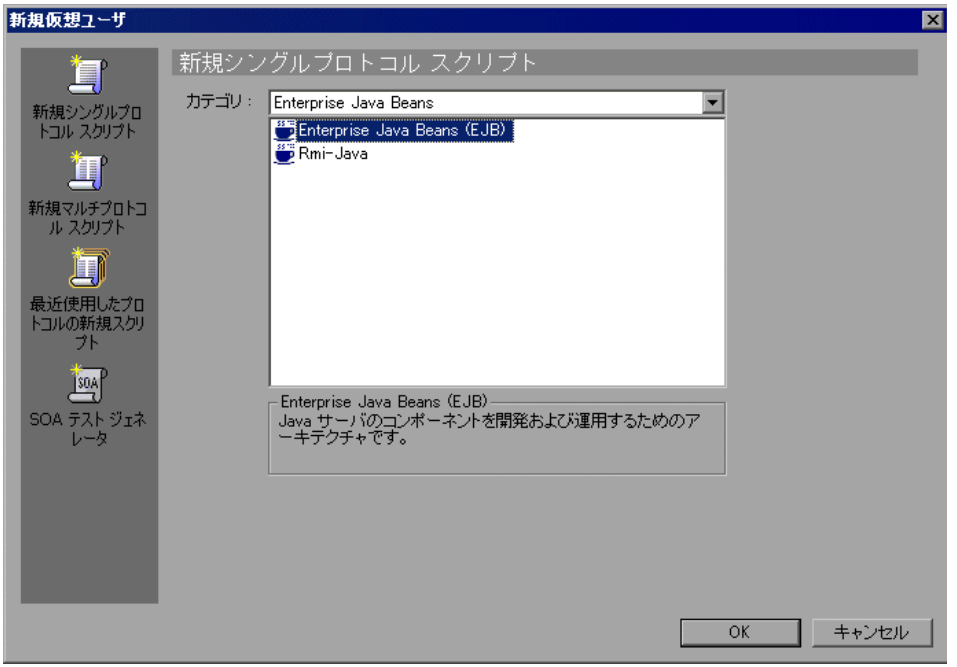
RTE 仮想ユーザ・スクリプトの新規作成

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 Emulation（RTE）**］を選択します。［**OK**］クリックします。VuGen によって空の RTE スクリプトが生成され，カーソルの位置が **vuser_init** セクション内で記録を開始する位置に表示されます。

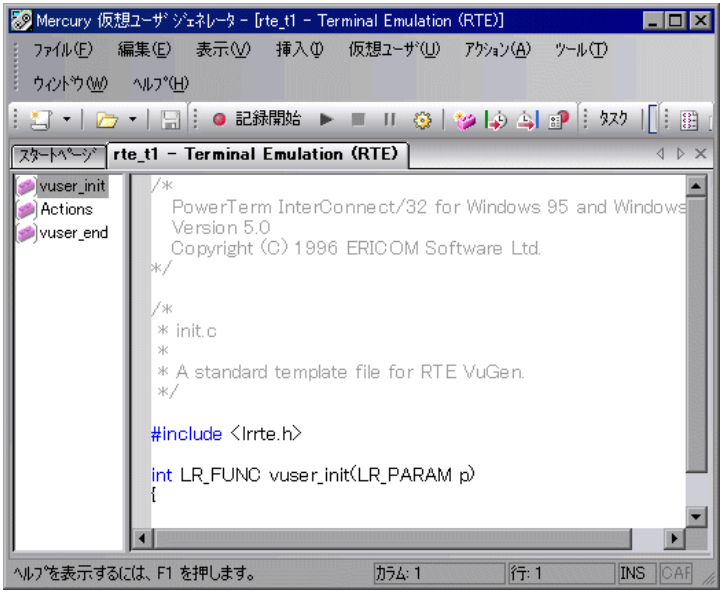
ターミナルの設定と接続の手順の記録

スケルトンの仮想ユーザ・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，必ず **vuser_init** セクションに記録するようにします。**vuser_init** セクションは，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 13 章「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。

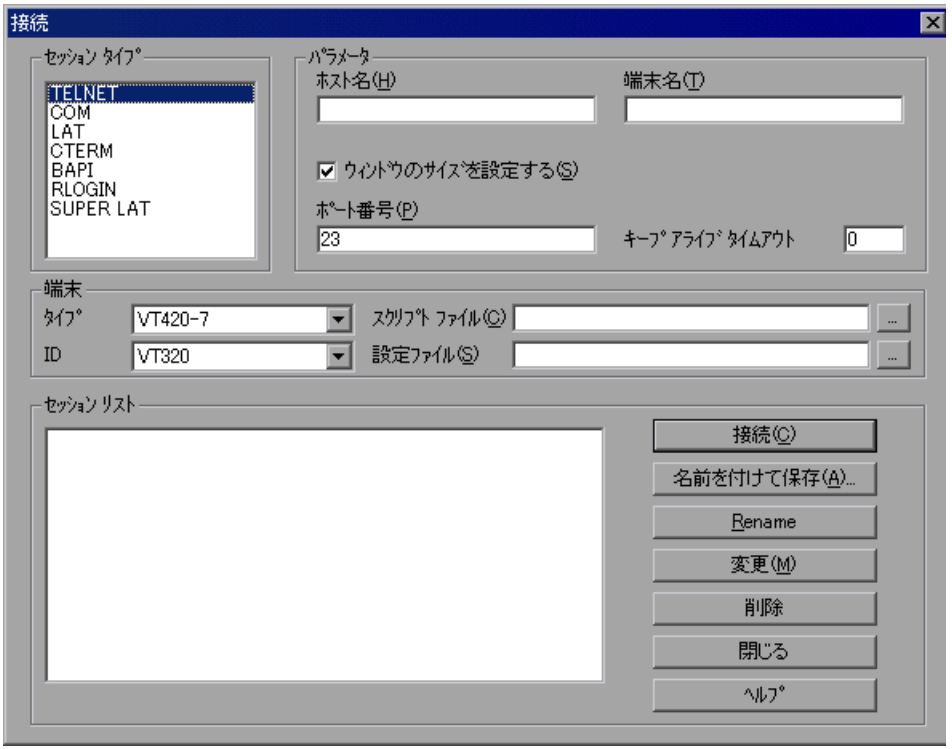
4 [**記録開始**] ボタンをクリックします。PowerTerm のメイン・ウィンドウが開きます。

5 PowerTerm のメニュー・バーから［**ターミナル**］＞［**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［**保持**］＞［**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 13 章「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，1141 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 13 章「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。

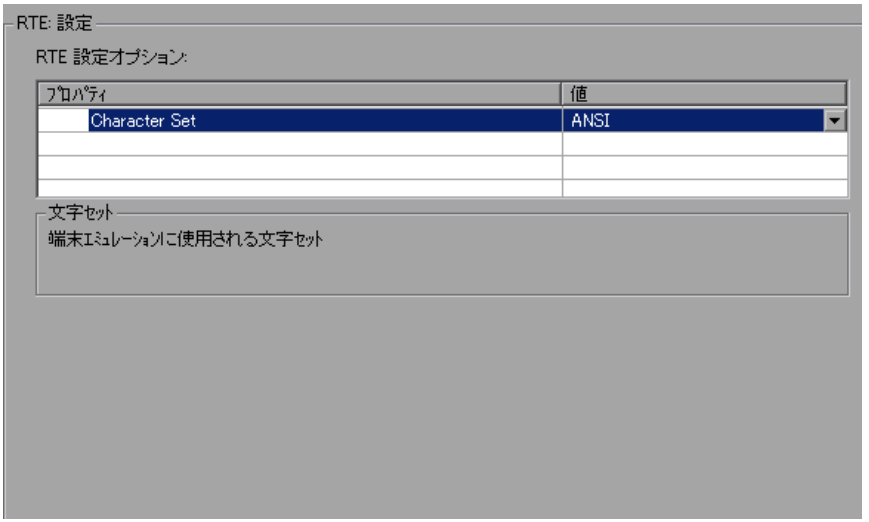
4［記録］ツールバーで，［**停止**］ボタンをクリックします。VuGen のメイン・ウィンドウに，記録されたすべてのステートメントが表示されます。

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。

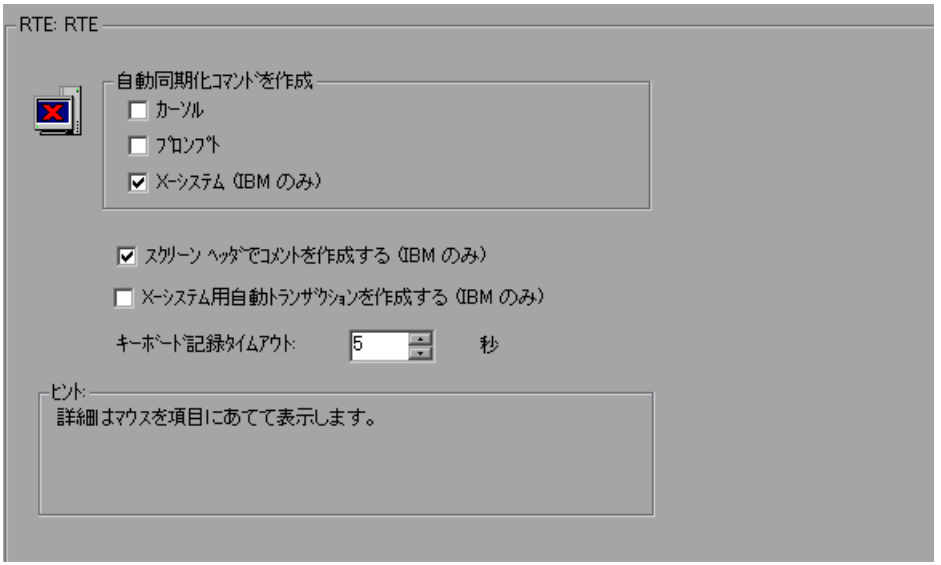
［設定］記録オプションを表示するには，ツールバーの［**記録オプションの編集**］ボタンをクリックするか，［**ツール**］＞［**記録オプション**］を選択します。［RTE］の［**設定**］ノードを選択します。



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［**記録オプションの編集**］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-System**］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［スクリーン ヘッダでコメントを作成する］チェック・ボックスを選択します。

標準設定では，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 が待つ時間の上限を設定するには，［**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，1145 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，1146 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 70 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。

たとえば，次の関数は，Return キーと，その後の Control キーと y キーの入力を表しています。

```
TE_type("<kReturn><kControl-y>");
```

その他の例としては，<kF1>，<kUp>，<kF10>，<kHelp>，<kTab> などがあります。

キー名を調べるには，キーの操作を記録した後，記録されたステートメントで名前を調べます。

注：TE_type ステートメントを（記録するのではなく）プログラムするときは，「**オンライン関数リファレンス**」（［**ヘルプ**］＞［**関数リファレンス**］）に記載されているキー定義を使用してください。

TE_type のタイムアウト値の設定

システムが X SYSTEM モード（または入力禁止モード）にある間，仮想ユーザが **TE_type** ステートメントを送信しようとした場合，その仮想ユーザは X SYSTEM モードが終了するまで入力を待たされます。システムが TE_XSYSTEM_TIMEOUT ミリ秒を超えて X SYSTEM モードを保持した場合は，**TE_type** 関数が TE_TIMEOUT エラーを返します。

TE_XSYSTEM_TIMEOUT の値は **TE_setvar** を使用して設定できます。TE_XSYSTEM_TIMEOUT の標準値は 30 秒です。

仮想ユーザに対する事前入力の許可

場合によっては，システムが X SYSTEM モード（または入力禁止モード）にある間も，仮想ユーザに対してキーストロークの送信を許可したいことがあります。たとえば，仮想ユーザに対して Break キーの押下を許可することができます。システムが X SYSTEM モードにある間も仮想ユーザに対してキーストロークの送信を許可するには，TE_ALLOW_TYPEAHEAD 変数を使用します。

事前入力を禁止するには，TE_ALLOW_TYPEAHEAD を 0 に設定し，事前入力を許可するには，0 以外の値に設定します。TE_ALLOW_TYPEAHEAD の値を設定するには **TE_setvar** を使用します。標準設定では，TE_ALLOW_TYPEAHEAD は 0 に設定されており，X SYSTEM モード中はキーストロークが送られません。

TE_type 関数とその規約の詳細については，「**オンライン関数リファレンス**」（[**ヘルプ**] ＞ [**関数リファレンス**]）を参照してください。

入力スタイルの設定

RTE 仮想ユーザに対しては，FAST と HUMAN の 2 つの入力スタイルを設定できます。FAST スタイルでは，仮想ユーザからターミナル・エミュレータへの入力ができるだけ高速に実行されます。HUMAN スタイルでは，文字を入力するたびに仮想ユーザが一時停止します。このため，人間のユーザによるキーボードでの入力が，仮想ユーザによって，より正確にエミュレートされます。

入力スタイルは **TE_typing_style** 関数を使用して設定します。**TE_typing_style** 関数の構文は次のとおりです。

int **TE_typing_style (**char ***style);**

ここで，**style** には FAST または HUMAN を指定できます。標準設定の入力スタイルは HUMAN です。HUMAN の入力スタイルを選択する場合は，次の形式になります。

HUMAN, **delay** [,**first_delay**]

delay には，キーストローク間の間隔（ミリ秒）を指定します。省略可能なパラメータ **first_delay** には，文字列の 1 文字目を入力する前の待ち時間（ミリ秒）を指定します。次に例を示します。

```
TE_typing_style ("HUMAN, 100, 500");
TE_type ("ABC");
```

上記の場合は，仮想ユーザは文字 A を入力する前に 0.5 秒待ちます。次に，文字「B」を入力する前に 0.1 秒待ち，その次に文字「C」を入力する前にさらに 0.1 秒待ちます。

TE_typing_style 関数とその規約の詳細については，「**オンライン関数リファレンス**」（［**ヘルプ**］＞［**関数リファレンス**］）を参照してください。

TE_typing_style 関数を使用する以外に，実行環境の設定を使用して入力スタイルを設定することもできます。詳細については，第 70 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。

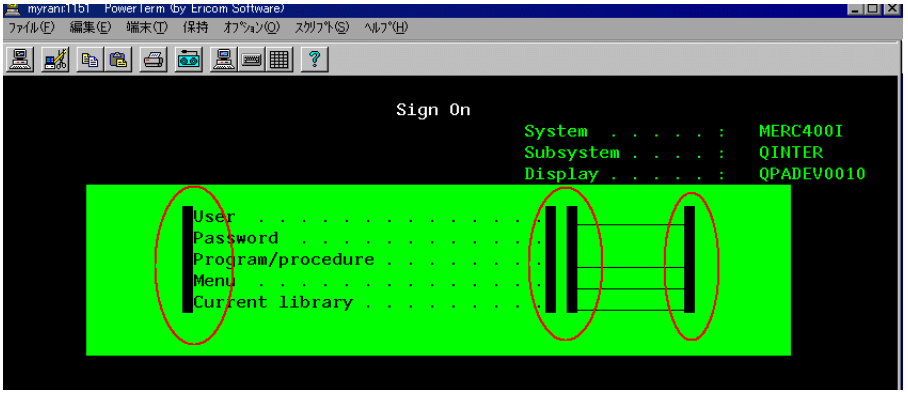
RteGenerateDeviceName 関数が仮想ユーザ・スクリプト内に存在する場合は，新しいデバイス名が必要になるたびに，仮想ユーザからこの関数が呼び出されます。**RteGenerateDeviceName** 関数がスクリプト内で定義されておらず，一意のデバイス名が必要になった場合は，**TE_connect** 関数によって必要な名前が生成されます。

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の設定

一部のターミナル・エミュレータでは，各フィールドの先頭と末尾を示すために区分文字が使用されます。これらの区分文字は画面上では空白として現れ，目に見えません。次に示すターミナル・エミュレータでは，フィールド区分文字を見えるようにするために画面の中央部分の色を反転させています。区分文字を楕円で囲んで示してあります。



TE_wait_text，**TE_get_text**，および **TE_find_text** 関数は，画面の指定の部分にある文字を識別することによって動作します。指定の部分の中にフィールド区分文字がある場合は，その文字を空白として識別するか，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 70 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。

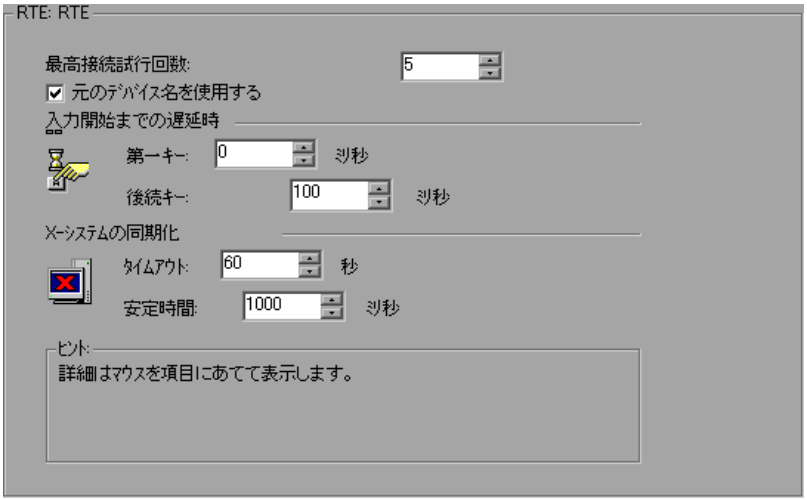
- ターミナル・エミュレータ実行環境の設定について
- 接続試行回数の変更
- 元のデバイス名の指定
- 入力遅延の設定
- X SYSTEM 同期化の設定

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 13 章「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［安定時間］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 71 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。たとえば，実際のユーザが次の一連のキーストロークを銀行アプリケーションに送信する場合を考えます。

1 「1」と入力して，銀行アプリケーションのメニューから「金融情報」を選択します。

2 「必要な情報を選択してください。」というメッセージが表示されたら，「3」と入力して，メニューから「ダウ・ジョーンズ工業平均株価」を選択します。

3 詳細なレポートが画面に表示されたら，「5」と入力して，銀行アプリケーションを終了します。

この例では，実際のユーザが各ステップで入力する前に画面上の合図を待っているので，銀行アプリケーションへの入力は同期化されています。この合図は，特定のメッセージが画面上に表示されること，または画面上のすべての情報が安定した状態になることのいずれかです。

仮想ユーザの入力は，TE 同期化関数 **TE_wait_sync**，**TE_wait_text**，**TE_wait_silent**，および **TE_wait_cursor** を使って同じ方法で同期化できます。これらの関数は，ターミナル・ウィンドウに入力する実際のユーザをエミュレートし，次のコマンドを入力する前にサーバの応答を待ちます。

TE_wait_sync 関数は，ブロック・モード（IBM）・ターミナルを同期化する場合にのみ使用されます。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，1163 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・エミュレータが調整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 ［**ツール**］＞［**記録オプション**］を選択します。［記録オプション］ダイアログ・ボックスが表示されます。

2 ［**X- システム用自動トランザクションを作成する**］オプションを選択して，［**OK**］をクリックします。

キャラクタ・モード（VT）・ターミナルの同期化

キャラクタ・モード（VT）・ターミナルでは，3 種類の同期化を使用できます。選択できる同期化の種類は，次の要因で決まります。

- ターミナル・エミュレータで実行するアプリケーションの設計
- 同期化する具体的なアクション

特定位置でのカーソル表示の待機

VT タイプのターミナルで推奨される同期化方法は，カーソル同期化です。カーソル同期化は，スクロール形式や TTY 形式のアプリケーションよりも，全画面形式やフォーム形式のアプリケーションで特に有効です。

カーソル同期化では，**TE_wait_cursor** 関数を使用します。RTE 仮想ユーザ・スクリプトを実行すると，**TE_wait_cursor** 関数は画面上の指定された位置にカーソルが表示されるまでスクリプトの実行を一時停止するように仮想ユーザに指示します。指定された位置にカーソルが表示されることは，アプリケーションがターミナル・エミュレータから次の入力を受け付ける準備ができたことを意味します。

TE_wait_cursor 関数の構文は次のとおりです。

int **TE_wait_cursor (**int **col**, int **row**, int **stable**, int **timeout);**

スクリプトの実行中，**TE_wait_cursor** 関数はカーソルが **col** と **row** で指定された位置に移動するまで待機します。

stable パラメータには，カーソルが指定された位置で安定している時間（ミリ秒）を指定します。VuGen を使ってスクリプトを記録する場合，**stable** は標準で 100 ミリ秒です。クライアント・アプリケーションが **timeout** パラメータで指定された時間内に安定しない場合，この関数は TIMEOUT を返します。VuGen を使ってスクリプトを記録する場合，**timeout** は標準で TIMEOUT の値（90 秒）に設定されます。**stable** パラメータの値と **timeout** パラメータの値は，どちらも記録されたスクリプトを直接編集することによって変更できます。

次のステートメントは，カーソルが安定した状態で 3 秒間待機します。カーソルが 10 秒以内に安定しない場合，この関数は TIMEOUT を返します。

```
TE_wait_cursor(10, 24, 3000, 10);
```

TE_wait_cursor 関数の詳細については，「**オンライン関数リファレンス**」（[**ヘルプ**］＞［**関数リファレンス**］）を参照してください。

スクリプトの記録中に **TE_wait_cursor** ステートメントを自動的に生成してスクリプトに挿入するように VuGen を設定できます。VuGen によって自動的に生成された **TE_wait_cursor** ステートメントの例を次に示します。

```
TE_wait_cursor(7, 20, 100, 90);
```

記録中に TE_wait_cursor ステートメントを自動的に生成してスクリプトに挿入するように VuGen を設定するには，次の手順を実行します。

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 ［**ツール**］＞［**記録オプション**］を選択します。［記録オプション］ダイアログ・ボックスが表示されます。

2 ［**自動同期化コマンドを作成**］の［**プロンプト**］チェック・ボックスを選択して，［**OK**］をクリックします。

VuGen によって自動的に生成された **TE_wait_text** ステートメントの例を次に示します。この関数は，「keys」という文字列が画面上の任意の場所に表示されるのを最大 20 秒間待ちます。VuGen が **TE_wait_text** 関数を生成するときは，省略可能なパラメータはすべて省略されます。

```
TE_wait_text("keys", 20);
```

ターミナルの安定の待機

カーソル同期化もテキスト同期化もうまく機能しない場合は，「安定同期化」を使用してスクリプトを同期化できます。「安定同期化」では，仮想ユーザはターミナル・エミュレータが安定するのを指定された時間だけ待ちます。エミュレータは，指定された期間内にサーバからの入力を受信しなかったときに安定したとみなされます。

注：安定同期化は，カーソル同期化とテキスト同期化がどちらもうまく機能しない場合にのみ使用します。

ターミナルが安定するまで待機するようにスクリプトを設定するには，**TE_wait_silent** 関数を使用します。ターミナルが安定している期間を指定します。ターミナルが指定された期間だけ安定していれば，**TE_wait_silent** 関数は，アプリケーションがターミナル画面へのテキストの表示を終了し，画面が安定したとみなします。

関数の構文は次のとおりです。

int **TE_wait_silent (**int **sec**, int **milli**, int **timeout);**

TE_wait_silent 関数は，ターミナル・エミュレータが安定するのを **sec**（秒）と **milli**（ミリ秒）で指定された期間だけ待ちます。エミュレータは，サーバからの入力を受信しなかったときに安定したとみなされます。**timeout** で指定された時間内にエミュレータが安定しない（文字の受信が終了しない）場合，この関数はエラーを返します。

たとえば，次のステートメントは画面が安定した状態で 3 秒間待機します。10 秒経過しても画面が安定しない場合，この関数はエラーを返します。

```
TE_wait_silent (3, 0, 10);
```

詳細については，「**オンライン関数リファレンス**」（[**ヘルプ**] > [**関数リファレンス**]）を参照してください。

第 72 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。関数の構文は次のとおりです。

int **TE_find_text (**char **pattern*, int *col1*, int *row1*, int *col2*, int *row2*,int **retcol*, int **retrow*, char **match*)**;**

この関数は，*col1*，*row1*，*col2*，*row2* で定義される四角形の内部で，*pattern* に一致するテキストを検索します。パターンに一致するテキストは *match* に返され，実際の行位置と列位置は *retcol* と *retrow* に返されます。検索は左上隅から行われます。複数の文字列が *pattern* に一致した場合は，左上隅に最も近い文字列が返されます。

pattern には正規表現を含めることができます。正規表現の使用方法の詳細については，「**オンライン関数リファレンス**」を参照してください。

TE_find_text ステートメントは，仮想ユーザ・スクリプトに手作業で入力する必要があります。**TE_find_text** 関数の構文の詳細については，「**オンライン関数リファレンス**」（[**ヘルプ**] > [**関数リファレンス**]）を参照してください。

画面からのテキストの読み取り

TE_get_text_line 関数は，画面上の指定された領域からテキスト行を読み取ります。関数の構文は次のとおりです。

char ***TE_get_text_line (**int *col*, int *row*, int *width*, char **text*)**;**

この関数は，テキスト行をターミナル画面から *text* バッファにコピーします。行の最初の文字は *col* と *row* で定義します。行の最後の文字の列座標は *width* で定義します。画面から読み取られたテキストは，*text* バッファに返されます。行内にタブや空白が含まれる場合は，それらに相当する数の空白が返されます。

また，**TE_get_cursor_position** 関数を使ってターミナル画面上のカーソルの現在位置を取得することもできます。**TE_get_line_attribute** 関数は，テキスト行内の文字の書式設定（太字，下線など）を返します。

TE_get_text_line ステートメントは，仮想ユーザ・スクリプトに手作業で入力する必要があります。**TE_get_text_line** 関数の構文の詳細については，「**オンライン関数リファレンス**」（[**ヘルプ**] > [**関数リファレンス**]）を参照してください。

第 13 部

メール・サービス・プロトコル

第 73 章

メール・サービス仮想ユーザ・スクリプトの作成

VuGen では，プロトコル・レベルでいくつかのメール・サービスをテストできます。VuGen は，メール送信，およびメール・サーバに対して実行されるほとんどの標準的な操作をエミュレートします。

本章では，次の項目について説明します。

- メール・サービス仮想ユーザ・スクリプトの作成について
- 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（C または Visual Basic）を指定できます。詳細については，第 5 章「スクリプト生成オプションの設定」を参照してください。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 4 章「VuGen を使った記録」を参照してください。

メール・サービス関数はすべて，対で用意されています。一方がグローバル・セッション用で，もう一方で特定のメール・セッションを指定します。たとえば，**imap_logon** はグローバルに IMAP サーバにログオンしますが，**imap_logon_ex** は特定のセッションのために IMAP サーバにログオンします。

メール・サービス仮想ユーザ・スクリプトの概要

本項では，VuGen を使用したメール・サービス仮想ユーザ・スクリプトの開発プロセスの概要を説明します。

メール・サービス仮想ユーザ・スクリプトを作成するには，次の手順を実行します。

1 **VuGen を使って基本スクリプトを作成します。**

VuGen を起動して，1 つまたは複数のメール・プロトコルを対象とする仮想ユーザ・スクリプトを新規作成します。

2 **VuGen を使って基本となるスクリプトを記録します（MAPI を除く）。**

記録するアプリケーションを選択します。アプリケーションを対象に標準的な操作を実行します。詳細については，第 4 章「VuGen を使った記録」を参照してください。

MAPI では，記録はサポートされていません。その代わりに空の MAPI スクリプトを作成し，**mapi** 関数を手作業で挿入します。使用例は，「**オンライン関数リファレンス**」（[**ヘルプ**] > [**関数リファレンス**]）を参照してください。

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 8 章「仮想ユーザ・スクリプトの拡張」を参照してください。

4 **パラメータを定義します（任意）。**

スクリプトに記録された固定値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 9 章「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 12 章「ステートメントの相関」を参照してください。

6 **実行環境を設定します。**

実行環境の設定では，スクリプト実行時の仮想ユーザの振る舞いを制御します。設定には，ループ，ログ，タイミング情報などがあります。

詳細については，第 13 章「実行環境の設定」を参照してください。

7 **VuGen でスクリプトを実行します。**

VuGen でスクリプトを保存して実行し，正しく動作することを確認します。

詳細については，第 15 章「スタンドアロン・モードでの仮想ユーザ・スクリプトの実行」を参照してください。

スクリプトを作成した後，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 **Business Availability Center** 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。これらの関数の詳細な構文については，「**オンライン関数リファレンス**」（[**ヘルプ**] ＞ [**関数リファレンス**]）を参照してください。

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_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。各 POP3 関数は **pop3** という接頭辞が付いています。これらの関数の詳細な構文については，「**オンライン関数リファレンス**」（［**ヘルプ**］＞［**関数リファレンス**］）を参照してください。

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。これらの関数の詳細な構文については，「**オンライン関数リファレンス**」（［**ヘルプ**］＞［**関数リファレンス**］）を参照してください。

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 14 部

ミドルウェア・プロトコル

第 74 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，次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・スクリプトの詳細
- 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 48 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** によってエラーのないことが確認された後，スクリプトがコンパイルされます。スクリプトが正しく機能することを確認したら，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**』，**Tuning Console** のマニュアル，**Performance Center** のマニュアル，または **Business Availability Center** 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 38 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。

次の項では，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 38 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 27 章「Java 記録オプションの設定」を参照してください。

4 **典型的なユーザ・アクションを記録します。**

スクリプトの記録を開始します。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 関数を仮想ユーザ・スクリプトに追加します。詳細については，第 38 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 62 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 9 章「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 29 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。**

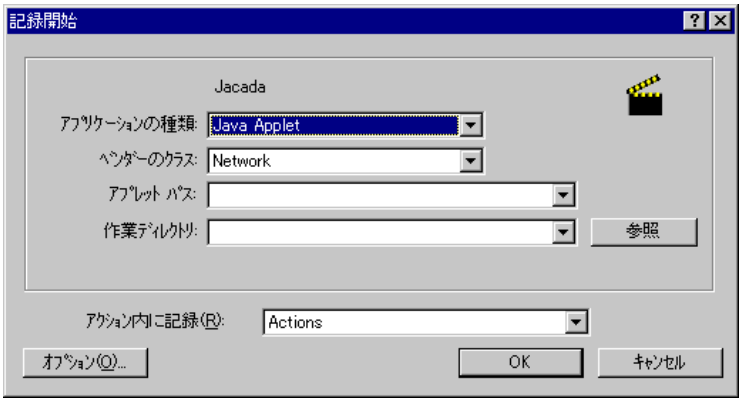
VuGen から スクリプトを実行して，実行時の情報を実行ログで確認します。詳細については，第 15 章「スタンドアロン・モードでの仮想ユーザ・スクリプトの実行」を参照してください。

Jacada 仮想ユーザの記録

Jacada スクリプトを記録して，完全互換の Java プログラムを作成します。

Jacada スクリプトを記録するには，次の手順を実行します。

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，［**ネットワーク**］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，［**ローカル**］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 27 章「Java 記録オプションの設定」を参照してください。

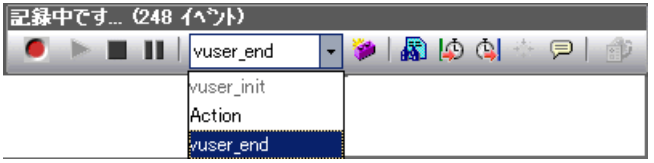
6 ［**アクション内に記録**］ボックスで，記録を挿入するセクションを選択します。vuser_init，Actions，および vuser_end の各セクションに対応する **init**，**action**，および **end** の 3 つのセクションがあります。次の表に，各メソッドに何を含め，どのタイミングで呼び出されるかを示します。

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	実行中
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，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。

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。

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すべてのクラスが，マシンの CLASSPATH 環境変数，または，実行環境設定の［**Classpath**］パネルで設定されている **Classpath Entries** に含まれている必要があります。

Jacada サーバは，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 38 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・スクリプトの詳細

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，スクリプト内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 38 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 75 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・スクリプトといいます。

本章では，次の項目について説明します。

- Tuxedo 仮想ユーザ・スクリプトについて
- 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・スクリプトの詳細
- Tuxedo バッファ・データの表示
- Tuxedo 仮想ユーザの環境設定の定義
- Tuxedo アプリケーションのデバッグ
- Tuxedo スクリプトの相関

以降の情報は，PeopleSoft-Tuxedo，Tuxedo 6 および Tuxedo 7 仮想ユーザ・スクリプトを対象とします。

Tuxedo 仮想ユーザ・スクリプトについて

Tuxedo アプリケーションを記録すると，VuGen は，記録されたアクションを記述する LRT 関数を生成します。これらの関数は，Tuxedo クライアントとサーバの間の通信をエミュレートします。各 LRT 関数は，接頭辞 **lrt** で始まります。

接頭辞 **lrt** に加え，**tp**，**tx**，**F** といった補助接頭辞を使用する関数もあります。これらの補助接頭辞は，実際の Tuxedo 関数と同様に，関数の型を示します。補助接頭辞 **tp** は，Tuxedo クライアントの tp セッションを示します。たとえば，**lrt_tpcall** は，サービス要求を送信して応答を待ちます。補助接頭辞 **tx** は，グローバルな tx セッションを表します。たとえば，**lrt_tx_begin** はグローバルなトランザクションを開始します。補助接頭辞 **F** は，FML バッファに関連する関数を表します。たとえば，**lrt_Finitialize** は既存のバッファを初期化します。

補助接頭辞のない関数は，標準 C 関数をエミュレートします。たとえば，**lrt_strcpy** は，C 関数 **strcpy** と同じように文字列をコピーします。

記録されたスクリプトは，VuGen のメイン・ウィンドウで表示および編集ができます。セッション中に記録された LRT 関数が VuGen ウィンドウに表示されるので，ネットワークの動作状況を視覚的に追跡できます。

記録前の作業

記録を行う前に，Tuxedo ディレクトリ %TUXDIR%¥bin がパスに含まれていることを確認します。

VuGen の再起動後に環境変数を変更している場合，VuGen は，環境変数の現在の値ではなく元の値を記録することがあります。

不整合を防ぐため，Tuxedo アプリケーションを記録する前には VuGen を再起動します。

Tuxedo 仮想ユーザ・スクリプトの概要

本項では，VuGen を使った Tuxedo 仮想ユーザ・スクリプトの作成プロセスの概要を説明します。

Tuxedo 仮想ユーザ・スクリプトの作成は，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使用した標準的な操作を記録します。

詳細については，第 4 章「VuGen を使った記録」を参照してください。

2 **スクリプトを拡張します。**

スクリプトに，トランザクション，ランデブー・ポイント，および制御フロー構造を挿入して，スクリプトを拡張します。

詳細については，第 8 章「仮想ユーザ・スクリプトの拡張」を参照してください。

3 **パラメータを定義します（任意）。**

スクリプトに記録された固定値の代わりにパラメータを定義します。固定値をパラメータで置き換えることにより，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 9 章「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 12 章「ステートメントの相関」を参照してください。

5 **実行環境を設定します。**

実行環境を設定することによって，スクリプト実行時の仮想ユーザの動作を制御します。設定には，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 13 章「実行環境の設定」を参照してください。

6 **VuGen からスクリプトを実行します。**

VuGen でスクリプトを保存して実行し，正しく動作することを確認します。

詳細については，第 15 章「スタンドアロン・モードでの仮想ユーザ・スクリプトの実行」を参照してください。

スクリプトを作成した後，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** のマニュアル，**Performance Center** のマニュアル，または **Business Availability Center** 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]	フィールド名に対するマップ・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	型付きバッファ情報を判別します。

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	クライアントを System/T アプリケーションに結合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に対する応答を返します。
lrt_tpgprio	最後に送信または受信された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	会話型接続でメッセージを受信します。
lrt_tpsend	会話型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す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]	失敗した最終 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・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，見つかった文字列を基準にして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。**lrt_save_searched_string** 関数は，文字列の出現場所を基準にした相対的な位置にある情報を保存する場合に使用します。PeopleSoft 仮想ユーザの場合には，**lrt_save_searched_string** を使用することをお勧めします。Peoplesoft サーバから返される応答バッファは，記録時と再生時でサイズが異なることが多いためです。

Tuxedo 仮想ユーザ・スクリプトの詳細

セッション記録後，記録されたコードを VuGen の組み込みエディタに表示できます。スクリプトをスクロールして，アプリケーションで生成された Tuxedo ステートメントの表示や，サーバによって返されたデータの検査ができます。VuGen ウィンドウには，記録された Tuxedo セッションに関する重要な情報が表示されます。メイン・ウィンドウにスクリプトを表示すると，VuGen が動作状況を記録したシーケンスを確認できます。

次の例では，VuGen がクライアントのアクションを Tuxedo の銀行アプリケーションに記録しています。クライアントは，銀行の口座を開き，必要な詳細のすべてを指定するアクションを実行しました。クライアントが開始残高としてゼロを指定したところで，セッションは中止されています。

```
lrt_abort_on_error();
lr_think_time(65);
tpresult_int = lrt_tpbegin(30, 0);
data_0 = lrt_tpalloc("FML", "", 512);
lrt_Finitialize((FBFR*)data_0);

/* データ・バッファ data_0 に新規口座情報を入力する */
lrt_Fadd_fld((FBFR*)data_0, "name=BRANCH_ID", "value=8",
LRT_END_OF_PARMS);
lrt_Fadd_fld((FBFR*)data_0, "name=ACCT_TYPE", "value=C",
LRT_END_OF_PARMS);
lrt_Fadd_fld((FBFR*)data_0, "name=MID_INIT", "value=Q",
LRT_END_OF_PARMS);
lrt_Fadd_fld((FBFR*)data_0,"name=PHONE","value=123-456-7890",
            LRT_END_OF_PARMS);

lrt_Fadd_fld((FBFR*)data_0, "name=ADDRESS", "value=1 Broadway
            New York, NY 10000", LRT_END_OF_PARMS);
lrt_Fadd_fld((FBFR*)data_0,"name=SSN","value=111111111",
LRT_END_OF_PARMS);

lrt_Fadd_fld((FBFR*)data_0,"name=LAST_NAME",
"value=Doe",LRT_END_OF_PARMS);

lrt_Fadd_fld((FBFR*)data_0,"name=FIRST_NAME",
"value=BJ",LRT_END_OF_PARMS);

lrt_Fadd_fld((FBFR*)data_0, "name=SAMOUNT",
"value=0.00",LRT_END_OF_PARMS);
```

```
/* 新規口座を開く */
tpresult_int = lrt_tpcall("OPEN_ACCT", data_0, 0, &data_0, &olen_2, 0);
lrt_tpabort(0);
lrt_tpcommit(0);
lrt_tpfree(data_0);
lrt_tpterm();
```

Tuxedo スクリプトでのパラメータの使用

第 9 章「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，文字列内の等号 (=) 以降の部分のみを対象にできます。次に例を示します。

```
lrt_Fadd_fld((FBFR*)data_0,"name=PHONE","value= < parameter_1 > ",
                LRT_END_OF_PARMS);
```

Tuxedo スクリプトの実行

Tuxedo アプリケーションの記録または実行の際に問題が発生した場合は，Tuxedo アプリケーションが VuGen なしで動作し，環境変数が正しく定義されていることを確認します。詳細については，次に示す「Tuxedo バッファ・データの表示」を参照してください。Tuxedo 変数の設定または変更をした後は，VuGen とアプリケーションを再起動して，変更を有効にする必要があります。アプリケーションが 16 ビットの場合は，NTVDM プロセスを強制終了する必要もあります。

実行時に問題が発生した場合は，サーバ側の Tuxedo ログ・ファイルでエラー・メッセージを確認します。標準設定では，このファイルは，環境変数 APPDIR で示されるディレクトリにあります。ファイル名は，ULOG.mmddyy の形式です（mmddyy は，現在の月，日，年を示してします)。1999 年 3 月 12 日のファイルは ULOG.031299 となります。このファイルの標準設定の場所は，サーバ上の環境変数 ULOGPFX を設定することで変更できます。ログ・ファイルはクライアント側にもあります。ULOGPFX 変数で場所が変更されていなければ，カレント・ディレクトリに置かれます。

Tuxedo バッファ・データの表示

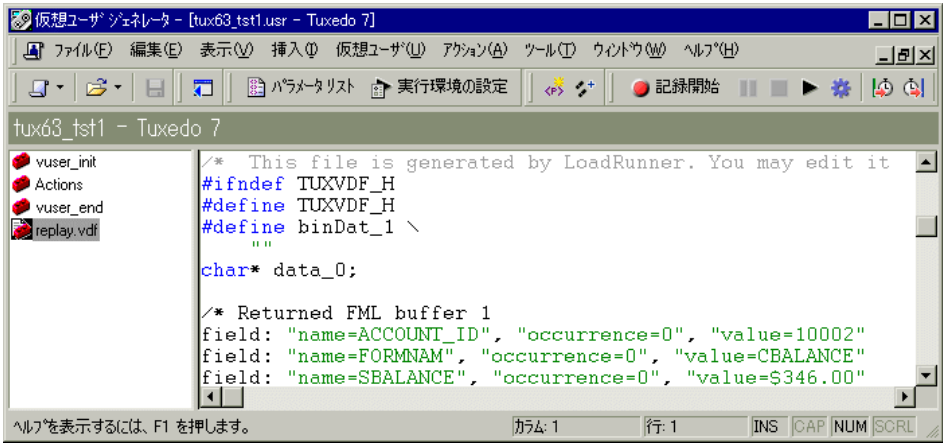
VuGen を使用して Tuxedo 仮想ユーザ・スクリプトを作成すると，アクションはスクリプトの 3 つのセクション **vuser_init**，**Actions**，**vuser_end** に記録されます。

受け取った，または転送されたデータはデータ・バッファに保管されますが，データ・バッファは非常に大きくなることがあります。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。

データ・ファイルを表示するオプションは，Tuxedo スクリプトでは標準で使用できます。



Tuxedo 仮想ユーザの環境設定の定義

次の項では，Windows および UNIX プラットフォームで動作する Tuxedo 仮想ユーザのシステム変数の設定について説明します。システム変数は，NT の場合は［コントロールパネル］の［システム］ダイアログ・ボックスで，UNIX の場合は .cshrc または .login ファイルで定義します。

TUXDIR	Tuxedo ソースのルート・ディレクトリ
FLDTBLDIR	FML バッファ情報を含むディレクトリのリスト。Windows では，ディレクトリの名前をセミコロンで区切ります。UNIX プラットフォームでは，ディレクトリの名前をコロンで区切ります。
FIELDTBLS	FML バッファ情報を含むファイルのリスト。Windows と UNIX のどちらのプラットフォームでも，ファイル名はカンマで区切ります。

次に例を示します。

```
SET FLDTBLDIR=%TUXDIR%¥udataobj;%TUXDIR%¥APPS¥WS (PC)
SET FIELDTBLS=bankflds,usysflds (PC)
setenv FLDTBLDIR $TUXDIR/udataobj:$TUXDIR/apps/bankapp (Unix)
setenv FIELDTBLS bank.flds,Usysflds (Unix)
```

実行中に，Tuxedo/WS ワークステーションの拡張を使用して，Tuxedo クライアントの次のシステム変数を定義する必要があります。

WSNADDR	ワークステーション・リスナ・プロセスのネットワーク・アドレスを指定します。これによって，クライアント・アプリケーションが Tuxedo にアクセスできるようになります。WSNADDR ステートメントで複数のアドレスを定義するには，各アドレスをカンマで区切る必要があります。
WSDEVICE	ネットワークにアクセスするデバイスを指定します。一部のネットワーク・プロトコルについては，この変数を定義する必要はありません。

次に例を示します。

```
SET WSNADDR=0x0002ffffc7cb4e4a (PC)
setenv WSNADDR 0x0002ffffc7cb4e4a (Unix)
setenv WSDEVICE /dev/tcp (Unix)
```

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**」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", ...);

/* 新規口座を開いて新規口座番号を保存する */
tpresult_int = lrt_tpcall("OPEN_ACCT", data_0, 0,&data_0, &olen_2, 0);
lrt_abort_on_error();
lrt_save_fld_val((FBFR*)data_0, "name=ACCOUNT_ID", 0,
"account_id");
```

```
/* 最初のクエリの結果を預金バッファに入力する */
lrt_Finitialize((FBFR*)data_0);
lrt_Fadd_fld((FBFR*)data_0, "name=ACCOUNT_ID",
"value= < account_id > ", LRT_END_OF_PARMS);
lrt_Fadd_fld((FBFR*)data_0, "name=SAMOUNT", "value=200.11", ...);
```

上の例では，アカウント ID が ACCOUNT_ID というフィールド名で表されています。記録時にフィールドがフィールド名ではなく ID 番号で表されるシステムもあります。

フィールド ID による相関は，次のように行われます。

```
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。
- **lrt_save_searched_string** 関数は，文字列の出現場所を基準にした相対的な位置にある情報を保存する場合に使用します。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

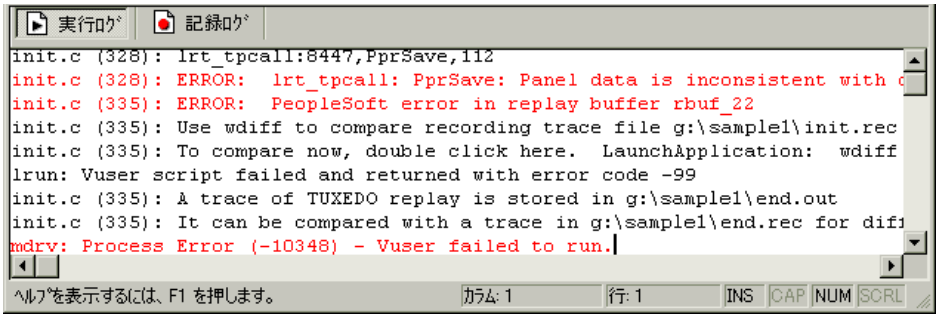
CARRAY バッファで作業を行っている場合，VuGen は Wdiff ユーティリティを使って比較できるログ・ファイルを生成します（記録中であれば拡張子は **.rec** に，再生中であれば拡張子は **.out** になります）。記録ログと再生ログの相違を調べて，CARRAY バッファのどの部分を相関させる必要があるか判断できます。

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］タブを調べます。

エラー・メッセージの後には，**Use wdiff to compare** という句で始まるステートメントが付いています。



3 実行ログのステートメントをダブルクリックして，**Wdiff** ユーティリティを起動します。

WDiff が開き，記録ファイルと再生ファイルの間の相違が黄色で強調表示されます。Wdiff ユーティリティの詳細については，第 12 章「ステートメントの相関」を参照してください。

lrt_save_parm を使用したステートメントを相関するには，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");
```

replay.vdf 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 15 部

ストリーミング・データ・プロトコル

第 76 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
- Media Player MMS 関数の使用

以降の情報は，RealPlayer および Media Player（MMS）プロトコルを対象とします。

ストリーミング・データ仮想ユーザ・スクリプトの記録について

ストリーミング・データ・プロトコルにより，メディアまたはストリーミング・データ・ファイルを再生するユーザをエミュレートできます。

ストリーミング・データ・プロトコルを使用してアプリケーションを記録すると，VuGen は記録時のアクションを記述する関数を生成します。RealPlayer セッションの場合，VuGen は接頭辞 **lreal** の付いた関数を生成します。Media Player セッションの場合，VuGen は接頭辞 **mms** の付いた関数を使用します。なお，Media Player 用の mms 関数では記録はサポートされておらず，再生のみ可能です。

ストリーミング・データ仮想ユーザ・スクリプトの概要

本項では，VuGen を使用して RealPlayer および Media Player のストリーミング・データ仮想ユーザ・スクリプトを作成する工程の概要を説明します。

RealPlayer または Media Player 仮想ユーザ・スクリプトを作成するには，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（読み取り専用）。**

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。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 4 章「VuGen を使った記録」を参照してください。

2 **スクリプトを拡張します。**

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 8 章「仮想ユーザ・スクリプトの拡張」を参照してください。

3 **パラメータを定義します（任意）。**

スクリプトに記録された固定値の代わりにパラメータを定義します。固定値をパラメータで置き換えることにより，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 9 章「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 12 章「ステートメントの相関」を参照してください。

5 **実行環境を設定します。**

実行環境を設定することによって，スクリプト実行時の仮想ユーザの動作を制御します。設定には，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 13 章「実行環境の設定」を参照してください。

6 **VuGen でスクリプトを実行します。**

VuGen で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 15 章「スタンドアロン・モードでの仮想ユーザ・スクリプトの実行」を参照してください。

スクリプトを作成した後，スクリプトを自環境（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 **Business Availability Center** 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。

Media Player MMS 関数の使用

Media Player の MMS プロトコルを使用してクライアントとサーバ間の通信をエミュレートするために作成された関数を，MMS 仮想ユーザ関数と呼びます。各 MMS 仮想ユーザ関数には，接頭辞 **mms** が付いています。

MMS 関数はすべて，グローバル・セッションと特定のセッション用にペアになっています。たとえば，**mms_close** は Media Player をグローバルに閉じ，**mms_close_ex** は特定のセッションでの Media Player を閉じます。

これらの関数の詳細な構文については，「**オンライン関数リファレンス**」（[**ヘルプ**］＞［**関数リファレンス**]）を参照してください。

関数名	説明
mms_close[_ex]	Media Player を閉じます。
mms_get_property[_ex]	Media Player クリップのプロパティを取得します。
mms_isactive[_ex]	Media Player がアクティブかどうか確認します。
mms_pause[_ex]	Media Player クリップの再生を一時停止します。
mms_play[_ex]	Media Player クリップを再生します。
mms_resume[_ex]	Media Player クリップの再生を再開します。
mms_sampling[_ex]	Media Player クリップをサンプリングします。
mms_set_property[_ex]	Media Player クリップのプロパティを設定します。
mms_set_timeout[_ex]	Media Player クリップのタイムアウト値を設定します。
mms_stop[_ex]	Media Player クリップの再生を停止します。

たとえば，**mms_play** 関数は，次の形式になります。

int **mms_play** (char message, ＜属性のリスト＞ , LAST);

次の例では，**mms_play** 関数は，**asf** ファイルを，継続時間を変えて再生します。

```
// 10 秒再生する
mms_play("Welcome","URL=mms://server/welcome.asf","duration=10",LAST);

// 5 秒待機後，クリップを最後まで再生する
mms_play ("Welcome","URL=mms://server/welcome.asf",
"duration=-1",
"starttime=5",LAST);
```

第 16 部

ワイヤレス・プロトコル

第 77 章

ワイヤレス仮想ユーザ・スクリプトの記録

VuGen を使用して標準的なワイヤレス・セッションを記録することによって，ワイヤレス・仮想ユーザ・スクリプトを生成できます。スクリプトを実行すると，生成された仮想ユーザによって，ツールキットまたは携帯電話と Web サーバ（または WAP 用ゲートウェイ）との間のユーザ・アクションがエミュレートされます。

本章では，次の項目について説明します。

- WAP プロトコルについて
- 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
- プッシュのサポート
- VuGen でのプッシュのサポート

以降の情報は，すべてのワイヤレス・プロトコルを対象とします。

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。

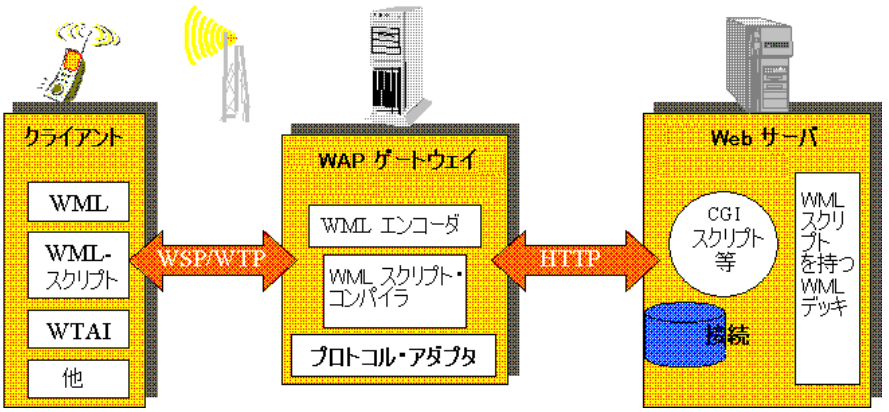
WAP ではさらに，次の条件を満たすプロキシ・サーバを規定しています。

- ワイヤレス・ネットワークと有線のインターネットの間のゲートウェイとして機能する。
- プロトコル変換機能がある。
- ワイヤレス受信のためにデータ転送を最適化する。

WAP アーキテクチャは WWW と非常によく似ています。すべてのコンテンツがインターネットの標準形式に似た形式で記述されます。コンテンツは WWW の領域では標準プロトコルで，ワイヤレスの領域（Wireless Session Protocol）では HTTP に似た最適化されたプロトコルを使って送信されます。WAP コンテンツはすべて WWW で標準的に使われている URL を使って指定します。

WAP では，オーサリングやパブリッシングの方法など，多くの部分が WWW 規格を使用しています。その一方で，ワイヤレス・デバイスおよびネットワークの特徴に合わせて，いくつかの WWW 規格が強化されています。「呼制御」および「メッセージング」などのモバイル・ネットワーク・サービスをサポートするための拡張機能が追加されています。WAP は，モバイル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 は，ツールキットで設定されている通信モード（WSP または HTTP）を自動的に検出します。ゲートウェイへのトラフィックを調べたい場合は，WSP モードで記録します。サーバおよびコンテンツ・プロバイダを検査したい場合は，HTTP モードでツールキット・セッションを記録して，ゲートウェイはバイパスすることができます。

VuGen は，ユーザ・セッションをエミュレートするためにユーザ定義の API 関数を使用します。ほとんどの関数は HTTP プロトコルを使用した標準の Web プロトコル関数です。いくつかの WAP 関数では，WAP 仮想ユーザ固有のアクションをエミュレートします。サポートされている関数の一覧については，1227 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。

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・スクリプトは，ワイヤレス・ブラウザを使用するユーザをエミュレートします。PC ベースの電話シミュレータ（ツールキット）を使用してユーザのブラウズ操作を記録します。その後，利用可能なテスト用マシンに，数百の仮想ユーザを分散配置し，各仮想ユーザから API を通じてサーバにアクセスします。これにより，多数のユーザによる負荷の下でサーバのパフォーマンスを測定できます。

本項では，VuGen を使ったワイヤレス仮想ユーザ・スクリプトの開発プロセスの概要を説明します。

ワイヤレス・スクリプトを作成するには，次の手順を実行します。

1 **VuGen を使って新しいスクリプトを作成します。**

［**ファイル**］＞［**新規作成**］を選択するか，［**新規作成**］ボタンをクリックして，シングル・プロトコル・モードまたはマルチ・プロトコル・モードで新しいスクリプトを作成します。

新規スクリプトの作成の詳細については，第 4 章「VuGen を使った記録」を参照してください。

2 **VuGen を使ってアクションを記録します。**

ツールキット・セッションでのアクションを記録します。VuGen は，自動的にツールキットの設定を検出し，記録時にその設定を使用します。

記録方法の詳細については，第 4 章「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 8 章「仮想ユーザ・スクリプトの拡張」を参照してください。

4 **パラメータを定義します（任意）。**

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 9 章「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。これらの設定には，実行論理，ペース設定，ログ，思考遅延時間，パフォーマンスの設定，ゲートウェイの設定が含まれます。

一般的な実行環境の設定の詳細については，第 13 章「実行環境の設定」を参照してください。

インターネット・プロトコルの一般的な実行環境設定の詳細については，第 53 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 78 章「WAP 実行環境の設定」を参照してください。

6 **相関を実行します。**

スクリプトを検査して，相関の必要な動的な値があるかどうか確認します。ワイヤレス・プロトコルでは，**web_reg_save_param** 関数を追加して手作業による相関を実行します。

詳細については，945 ページ「手作業による相関」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。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，第 15 章「スタンドアロン・モードでの仮想ユーザ・スクリプトの実行」を参照してください。

スクリプトを作成した後，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**』，**Performance Center** のマニュアル，または **Business Availability Center** 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 50 章「Web 仮想ユーザ関数の使用」を参照してください。

一般的な仮想ユーザ関数の名前には，**lr** という接頭辞が付いています。詳細については，35 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 50 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固定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_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	WAP ゲートウェイ接続用のクライアント機能を設定します。

wap_set_connection_mode	接続モードとセキュリティ・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。

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。クライアントに対してサービスを行うか情報を送信することで，サーバが応答します。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。この技術はまた，クライアント・サーバ型モデルに基づいていますが，サーバがデータを送信する前にクライアントから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）。このコンテンツ・タイプでは，モバイル・クライアント上のユーザ・エージェントがサービスをロードして実行できます。たとえば，WML デッキなどを実行できます。SL には，ユーザの介入なしに適宜ユーザ・エージェントによってロードされるサービスを示す URI が含まれています。
- **SI** ― サービス通知（SI）。このコンテンツ・タイプでは，エンド・ユーザに非同期で通知を送信できます。たとえば，新規メールの到着，株価の変動，ニュースのヘッドライン，広告などの通知が考えられます。

  最も基本的な形式の SI には，ショート・メッセージとサービスを示す URI が含まれています。メッセージは受信するとエンド・ユーザに表示され，ユーザは URI で示されたサービスを即座に開始するか後で処理するかを選択できます。SI の処理が後に延ばされると，クライアントはそれを格納して，エンド・ユーザは後のある特定の時点で処理できます。

VuGen でのプッシュのサポート

VuGen でのプッシュのサポートは，次の 3 つに分類できます。

- クライアント側でのプッシュのサポート ― プッシュ型メッセージを受信する機能です。
- WAP HTTP 仮想ユーザに対するプッシュのサポート ― プッシュ・イニシエータをエミュレートします。
- プッシュ型メッセージ（SI および SL）フォーマット・サービス ― プッシュ型メッセージをフォーマットします。

クライアント側におけるプッシュのサポート

VuGen は，クライアント側では，すべての再生モード（CO および CL）について，SL および SI の両方のプッシュ・サービスをサポートしています。**wap_wait_for_push** 関数は，仮想ユーザにプッシュ・メッセージの到着まで待機するように指示します。実行時の設定でこの関数のタイムアウトを設定します。

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 78 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，VuGen でサポートされている通常の HTTP POST メソッドに基づいています。現バージョンでは，最初の 2 つの操作だけが **wap_push_submit** および **wap_push_cancel** 関数によってサポートされています。

web_submit_data 関数を使用して，Web サーバにデータを送信できます。ただし，この関数では長く複雑な構造のデータを送信することは困難です。この種のデータの送信を可能にするため，またより直観的に理解できる API 関数を提供するために，XML メッセージ・データを適切にフォーマットする新しい API 関数 **wap_format_si_msg** と **wap_format_sl_msg** が追加されました。これらの関数の詳細については，「**オンライン関数リファレンス**」を参照してください。

第 78 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
- 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 53 章「インターネット実行環境の設定」を参照してください。

WAP 実行環境の設定について

WAP 仮想ユーザ・スクリプトを作成した後，WAP 固有の実行環境の設定を行います。これらの設定により，WAP 仮想ユーザの動作を制御して，WAP デバイスの実際のユーザを正確にエミュレートできるようになります。ゲートウェイおよび Radius に関する WAP 実行環境の設定が行えます。

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 53 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。

接続オプション

接続オプションでは，仮想ユーザが WAP ゲートウェイへの接続に使用する方式を指定します。

［**WAP ゲートウェイ**］：Web サーバに WAP ゲートウェイ経由でアクセスする仮想ユーザを実行します。

［**HTTP 直接**］：Web サーバに直接アクセスする仮想ユーザを HTTP モードで実行します。

注：［HTTP 直接］接続モードを選択した場合，残りの WAP ゲートウェイ・オプションは適用されません。

ゲートウェイの設定

仮想ユーザがゲートウェイ経由で接続する場合は，IP，ポート，WAP のバージョンの各オプションによってゲートウェイ接続が指定されます。

［**IP**］：ゲートウェイの IP アドレスを指定します。

［**PORT**］：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。

［**Wap のバージョン**］：適切な WAP のバージョン（**1.x（WSP）**または **2.0（HTTP プロキシ）**）を選択します。WAP 1.x（WSP）で記録した場合は，1.x（WSP）モードまたは 2.0（HTTP プロキシ）モードで仮想ユーザを実行できます。WAP 2.0（HTTP プロキシ）で記録した場合は，その同じモードでのみ仮想ユーザを実行できます。

ゲートウェイ接続モード

接続モードの設定は WAP バージョン 1.x（WSP）接続に適用されます。

［**コネクション指向モード**］：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 モード**］：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化**］：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。

ゲートウェイの詳細設定

WAP の機能およびゲートウェイの詳細オプションを設定するには，［ゲートウェイ］ノードの［**詳細設定**］オプションを展開します。

- ［**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）。詳細については，1229 ページ「プッシュのサポート」を参照してください。
- ［**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）。
- ［**CAPSessionResume**］：セッションの一時停止または再開の要求を可能にします。
- ［**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）。
  - ［**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）。
  - ［**MethodMOR**］：同時に発生可能な未処理の方式の数。
  - ［**PushMOR**］：同時に発生可能な未処理のプッシュ・トランザクションの数。

- ［**Bearer Type**］：伝送の基盤として使用されるベアラのタイプ。
  - ［**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（標準設定では有効）。
- ［**WTP の分割と再組み立て**］：WTP（Wireless Transport Protocol）による分割と再組み立て（SAR）を有効にします（標準設定では有効）。
  - ［**WTP 再送時間**］：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 簡略ハンドシェーク**］：リダイレクト・メッセージの受信時に，完全なハンドシェイクではなく略式のハンドシェイクを使用します（標準設定では無効）。
- ［**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 識別子**］：Deffie-Hellman 暗号化方式の識別子。この識別子は，Deffie-Hellman 暗号化方式を使用する Operwave ゲートウェイによる略式のハンドシェイクに必要です。
  - ［**ネットワーク MTU サイズ**］：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 ［**実行環境設定の編集**］ボタンをクリックするか，［**仮想ユーザ**］＞［**実行環境の設定**］を選択して，［実行環境設定］ダイアログ・ボックスを表示します。［**WAP：ゲートウェイ**］ノードを選択します。

2 スクリプトを（HTTP ではなく）WSP モードで再生する場合は，［**WAP ゲートウェイ**］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 WAP バージョンを **WAP 1.x（WSP）**または **WAP 2.0（HTTP）**から選択します。

5 WAP 1.x（WSP）の場合，接続モードとして，［**コネクション指向モード**］または［**コネクションレス モード**］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化**］オプションを選択します。

6 WAP 2.0（HTTP）の場合，［**詳細設定**］ノードを展開し，クライアントの機能とほかのゲートウェイ詳細オプションを設定します。これらのオプションの詳細については，前述の説明を参照してください。

Radius 接続データの設定

RADIUS（Remote Authentication Dial-In User Service）はクライアント・サーバ型プロトコルであり，リモート・アクセス・サーバが中央のサーバと通信して，ダイアルインのユーザを認証し，要求されているシステムやサービスへのアクセスを認めることができるソフトウェアです。

RADIUS では，すべてのリモート・サーバが共有できる中央のデータベースに企業がユーザ・プロファイルを保持できます。より厳格なセキュリティが用意されるので，単一の管理対象ネットワーク・ポイントで適用可能なポリシーを企業が設定できます。中央のサービスを利用することで，料金請求のための使用状況管理とネットワーク統計情報の蓄積が簡単になります。

RADIUS には次の 2 種類のサブプロトコルがあります。

- **Authentication** では，ユーザのアクセスを認証し管理します。
- **Accounting** では，料金請求のための使用状況管理とネットワーク統計情報の蓄積を行います。

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。標準設定は 120 秒です。
再送の再試行回数	伝送の失敗後に再送を試みる回数。標準設定値は 0 です。
サーバから返された属性をパラメータに保存する	仮想ユーザが，サーバから返された属性をパラメータ（後で使用可能）として保存できるようにします。標準設定は **False** です。
Radius クライアントの IP	Radius パケットの発信元 IP。通常は，1 台のロード・ジェネレータ・マシン上の異なる NIC カードから送信されるパケットを区別するために使用されます。

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［**ネットワークの種類**］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 ［**再送の再試行回数**］を指定します。

8 ［**サーバから返された属性をパラメータに保存する**］を有効にするかどうかを指定します。

9 ［**OK**］をクリックして設定を適用し，ダイアログ・ボックスを閉じます。

第 79 章

MMS 仮想ユーザ・スクリプトの作成

MMS（マルチメディア・メッセージング・サービス）仮想ユーザ・スクリプトを作成して，MMS の動作をエミュレートできます。

本章では，次の項目について説明します。

- MMS（マルチメディア・メッセージング・サービス）仮想ユーザ・スクリプトについて
- MMS 実行環境の設定
- コントローラでの MMS シナリオの実行
- MMS API

以降の情報は，マルチメディア・メッセージング・サービス仮想ユーザ・スクリプトを対象とします。

MMS（マルチメディア・メッセージング・サービス）仮想ユーザ・スクリプトについて

MMS（マルチメディア・メッセージング・サービス）は，SMS プロトコルの拡張機能です。SMS メッセージにはテキストのみ含められるのに対し，MMS では，MMS 対応の送受話器との間でさまざまなコンテンツを持つメッセージを送受信できます。テキスト，音声，電子メール・メッセージ，画像，ビデオ・クリップ，そしてストリーミング・データの形式のコンテンツが使用できます。また，携帯電話から電子メール・アドレスへマルチメディア・メッセージを送信することもできます。

一般に，MMS メッセージには添付ファイルの集合が含まれます。SMS メッセージ・サイズは 160 バイトに制限されていますが，MMS メッセージ・サイズは数 MB でも可能です。このような大きいサイズのメッセージが送信できるように，MMS では通常，第 3 世代（3G）通信網が必要です。

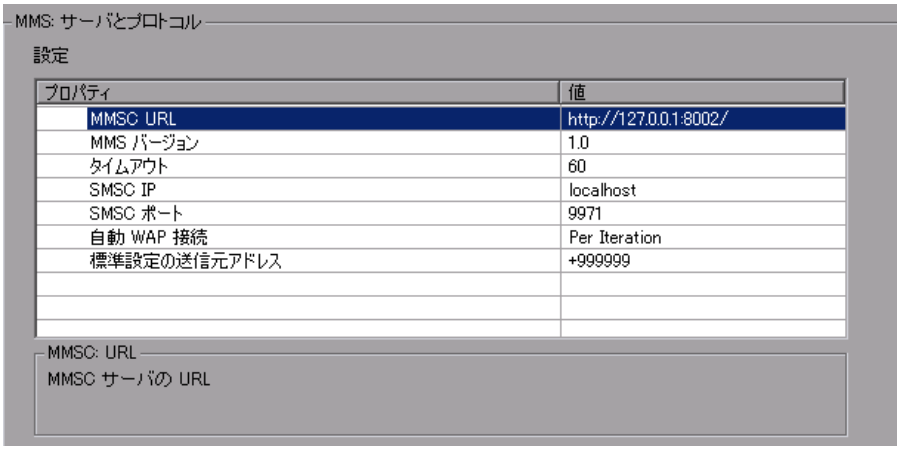
MMS メッセージを受信するために，携帯電話は SMS を介して MMS 通知を受信します。SMS メッセージは，SMPP，UCP，CIMD2 などのさまざまな SMS プロトコルを介して受信できます。SMS メッセージには，MMSC サーバのデータベースに格納されている MMS メッセージへの一意のパスが含まれます。携帯電話は，このパスを使用して，SMSC からメッセージをダウンロードします。VuGen の現在のバージョンでは，SMPP インタフェースを介した MMS 通知の受信をサポートしています。

MMS（マルチメディア・メッセージング・サービス）仮想ユーザ・スクリプトは，OMA（Open Mobile Alliance）で定義されている MMS プロトコルのバージョン 1.0 および 1.1 をサポートしています。MMS 仮想ユーザを使用することで，HTTP プロトコルを利用して，または WAP プロトコルを利用して WAP ゲートウェイを経由で，直接 MMSC サーバに MMS メッセージを送信できます。

MMS 実行環境の設定

スクリプトを実行する前に，実行環境の設定を行って，スクリプトが実ユーザを正確にエミュレートするようにできます。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 13 章「実行環境の設定」を参照してください。

次の項では，MMS（マルチメディア・メッセージング・サービス）仮想ユーザに固有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により，サーバとプロトコルの設定を行うことができます。



次のオプションを設定できます。

- ［**MMSC URL**］: MMSC（マルチメディア・メッセージング・センタ）サーバの URL です。
- ［**MMS バージョン**］: スクリプトで使用される MMS プロトコルのバージョンです。
- ［**タイムアウト**］: サーバが着信メッセージを待機する時間です。標準時間は 60 秒です。
- ［**SMSC IP**］: SMPP を介して MMS 通知を送信するのに使用される SMSC サーバの IP アドレスです。
- ［**SMSC ポート**］: SMPP を介して MMS 通知を送信するのに使用される SMSC サーバの IP ポートです。
- ［**自動 WAP 接続**］: WAP ゲートウェイに対する接続 / 接続解除を行うタイミングを定義します。この設定は，WAP ゲートウェイを使用する場合にのみ関係します。次の値を指定できます。
  - ［**Per Iteration**］: 各反復の始まりに接続し，各反復の終わりに接続を解除します（標準設定）。
  - ［**Per Send or Receive**］: 各メッセージの始まりに接続し，各メッセージの終わりに接続を解除します。
  - ［**None**］: 自動 WAP 接続を使用しません。
- ［**標準設定の送信元アドレス**］: Sender ヘッダーで送信される標準設定のアドレスです。標準設定値は +999999 です。

MMS サーバとプロトコルの設定を行うには，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［**実行環境設定の編集**］ボタンをクリックします。

2 実行環境の設定ツリーから［**MMS: サーバとプロトコル**］ノードを選択します。

3 前述の説明に従って，必要な値を選択します。

4 実行環境の設定ツリーから［**一般：その他**］を選択します。

5 ［マルチスレッド］の下の［**仮想ユーザをプロセスとして実行する**］を選択します。

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。

コントローラでの MMS シナリオの実行

MMS（マルチメディア・メッセージング・サービス）シナリオには，コマンド・ライン設定が必要です。

MMS コマンド・ラインの設定を行うには，次の手順を実行します。

1 ［シナリオのスケジュール］画面で［**詳細**］をクリックします。［グループ情報］ダイアログ・ボックスが表示されます。

2 ［コマンド ライン］ボックスが表示されていない場合は，［**詳細表示**］ボタンをクリックします。

3 ［コマンド ライン］のテキストの最後に -usingwininet yes を追加します。

4 ［**OK**］をクリックして，コマンド・ライン・スイッチを適用します。

MMS API

MMS API には次の関数が含まれます。

関数名	説明
mm_create_multipart (multipart_name, data, LAST)	後で送信関数に追加できるマルチパートの添付ファイルを作成します。添付ファイルには，**smil** ファイル，またはテキスト / 画像タイプのファイルが使用できます。 • **multipart_name**：マルチパートの添付ファイルの名前です。 • **data**：データ・ファイルの形式およびパスです。たとえば，"text=c:¥¥text.txt""image=c:¥¥image.gif""smil=c:¥¥file.smil" など。 • **その他の引数**：マルチパートの添付ファイルの MIME ヘッダー。
mm_create_submit_message (msg_name, recipient, LAST)	m-send-req メッセージを作成します。 • **msg_name**：メッセージ名です。 • **recipient**：メッセージの受信者です。指定可能な値は，"to= ＜番号＞ "，"cc= ＜番号＞ "，および "bcc= ＜番号＞ " です。 • **その他の引数**：subject や x-mms-priority など，MMS プロトコル仕様で定義されている m-send-req ヘッダー。添付ファイルの場合は，"multiparts= ＜ 1 番目のマルチパートの名前＞，＜ 2 番目のマルチパートの名前＞ " という形式を使用します。
mm_load_message (msg_name, file)	エンコードされたメッセージをファイルからロードします。メッセージは m-send-req タイプでなければなりません。 • **msg_name**：メッセージ名です。 • **file**：エンコードされたメッセージ・ファイルのパスです。
LPCSTR mm_message_get_header (msg_name, header)	指定されたメッセージのヘッダーの値を返します。 • **msg_name**：メッセージ名です。 • **header**：要求されたヘッダーです。

LPCSTR mm_message_get_multipart (msg_name, multipart_name, index)	インデックスに基づいて，指定されたメッセージのマルチパートを返します。 • **msg_name**：メッセージ名です。 • **mulitpart_name**：マルチパートの添付ファイルの名前です。 • **index**：マルチパートのインデックスです。
int mm_message_get_multipart_count (msg_name)	指定されたメッセージのマルチパートの数を返します。 • **msg_name**：メッセージ名です。
mm_multipart_get_header (multipart_name, header_name)	指定されたマルチパート名からヘッダーの値を返します。 • **mulitpart_name**：マルチパートの添付ファイルの名前です。 • **header_name**：要求されたヘッダーの名前です。
mm_notification_wait (out_msg, LAST)	MMS 通知メッセージを受信します。 • **out_msg**：出力メッセージの名前です。
mm_retrieve (out_msg,URI)	m-retrieve-conf タイプのメッセージを受信します。 • **out_msg**：出力メッセージの名前です。 • **URI**：メッセージへの一意のパスです。
mm_set_smpp_user (user_name, password, system_type)	SMPP ユーザの詳細を設定します。 • **user_name**：SMPP サーバに定義されているユーザの名前です。 • **password**：ユーザのパスワードです。 • **system_type**：SMPP サーバに定義されているシステム・タイプです（任意）。

mm_send_message (msg_name, conf_message, LAST)

m-send-req メッセージを送信します。標準設定では，メッセージは "application/vnd.wap.mms-message" content-type として送信されます。この値を変更するには，異なる content-type ヘッダーを指定します。

- **msg_name**：送信するメッセージです。
- **conf_message**：送信したメッセージに応えて受信した確認メッセージの名前です。確認メッセージを保存する必要がない場合は，空文字列を指定します。
- **その他の引数**：HTTP または WSP ヘッダー。

mm_send_acknowledge (transaction_id, report_allowed, LAST)

メッセージが送信されたという肯定応答を送信します。

- **transaction_id**：前に取得したメッセージのトランザクション ID です。
- **report allowed**：元のメッセージ送信者への送信レポートを許可するかどうかを示すフラグです。
- **system_type**：SMPP サーバに定義されているシステム・タイプです（任意）。

第 17 部

上級ユーザのために

第 80 章

Visual Studio による仮想ユーザ・スクリプトの作成

Visual C もしくは Visual Basic を使用して，Visual Studio で仮想ユーザ・スクリプトのテンプレートが作成できます。作成したテンプレートは，C または Visual Basic プログラムと同じようにコンパイルします。

本章では，次の項目について説明します。

- Visual Studio による仮想ユーザ・スクリプトの作成について
- Visual C による仮想ユーザ・スクリプトの作成
- Visual Basic による仮想ユーザ・スクリプトの作成
- 実行環境の設定とパラメータの設定

Visual Studio による仮想ユーザ・スクリプトの作成について

仮想ユーザ・スクリプトを作成するには，VuGen を使用したり Visual Studio などの開発環境を使用したりする方法があります。

VuGen

VuGen の記録機能を使用したり，VuGen エディタを使用して手作業でプログラミングしたりすることにより，Windows や UNIX プラットフォームで実行する仮想ユーザ・スクリプトを作成できます。Windows 環境で作成したスクリプトは Windows と UNIX 環境の両方で実行できます。記録は UNIX 環境で行うことはできません。

Visual Studio

Visual Studio を使えば，仮想ユーザ・スクリプトを Visual Basic，C，C++ でプログラミングすることができます。プログラムはダイナミック・リンク・ライブラリ（dll）としてコンパイルします。

本章では，Visual C と Visual Basic の開発環境でプログラミングによって仮想ユーザ・スクリプトを作成する方法について説明します。これらの環境では，開発アプリケーションに仮想ユーザ API のライブラリをインポートして仮想ユーザ・スクリプトを作成します。

また，VuGen エディタ上でも，アプリケーションのライブラリやクラスを組み込んで仮想ユーザ・スクリプトのプログラミングをすることもできます。VuGen では，C，Java，Visual Basic，VBScript または JavaScript のプログラミングが行えます。詳細については，第 37 章「ユーザ定義の仮想ユーザ・スクリプトの作成」を参照してください。

プログラミングで仮想ユーザ・スクリプトを作成する場合，VuGen テンプレートを大規模な仮想ユーザ・スクリプトの原形として使用できます。テンプレートで提供されるものは，次のとおりです。

- 正しいプログラム構造
- 仮想ユーザ API 呼び出し
- ダイナミック・リンク・ライブラリを作成するためのソース・コードとメイクファイル

テンプレートから基本的な仮想ユーザ・スクリプトを作成したら，スクリプトを，実行時の情報と統計値を指定して拡張します。詳細については，第 8 章「仮想ユーザ・スクリプトの拡張」を参照してください。

仮想ユーザ・スクリプトで使用可能なオンラインの C 言語の共通関数リファレンスについては，「**オンライン関数リファレンス**」（[**ヘルプ**] > [**関数リファレンス**]）を参照してください。

Visual C による仮想ユーザ・スクリプトの作成

Visual C で仮想ユーザ・スクリプトを作成する場合は，バージョン 6.0 以上の Visual C を使用してください。

Visual C を使用して仮想ユーザ・スクリプトを作成するには，次の手順を実行します。

1 Visual C で，ダイナミック・リンク・ライブラリ（dll）を作成する新規プロジェクトを開きます。[**ファイル**] > [**新規作成**] を選択し，[プロジェクト] タブをクリックします。

2 [ウィザード] で [**空の DLL プロジェクト**] を選択します。

3 プロジェクトに以下のファイルを追加します。

- 新規 **cpp** ファイルに，**init**，**run**，**end** の 3 つの関数をエクスポートしたもの（関数名は変更できます）。
- ライブラリ・ファイル lrun50.lib（< LoadRunner のインストール・ディレクトリ > ¥lib にあります）。

4 プロジェクトの設定で次の変更を行います。

- ［C/C++］タブを選択して，［**コード生成**（カテゴリ）］ > ［**使用するランタイムライブラリ**（リスト）］を選択し，これを［**マルチスレッド**（DLL）］に変更します。
- ［C/C++］タブを選択して，［**プリプロセッサ**（カテゴリ）］ > ［**プリプロセッサの定義**（編集フィールド）］の _DEBUG を削除します。

5 これで，クライアント・アプリケーションからコードを追加したり，通常通りプログラミングします。

6 仮想ユーザ API 関数を使ってスクリプトを拡張します。たとえば，メッセージを発行するには **lr_output_message**，トランザクションの開始を示すには **lr_start_transaction** を使用します。詳細については，「**オンライン関数リファレンス**」（［**ヘルプ**］ > ［**関数リファレンス**］）の「一般関数」を参照してください。

7 プロジェクトをビルドします。DLL として出力されます。

8 DLL と同じ名前のディレクトリを作成し，作成した DLL をこのディレクトリにコピーします。

9 **Template** ディレクトリの **lrvuser.usr** ファイルを開き，USR ファイル・キーを次のようにその DLL 名で上書きします。BinVuser= < **DLL_name** >

次の例は，**lr_output_messsage** 関数でどのセクションが実行されているかを示すメッセージを発行するものです。**lr_eval_string** 関数は，ユーザ名を取得します。次の例を使用する場合は，仮想ユーザ API のインクルード・ファイル **lrun.h** へのパスが正しいことを確認してください。

```
#include "c:¥mercury¥lrun_5¥include¥lrun.h"

extern "C" {
int __declspec(dllexport) Init (void *p)
{
      lr_output_message("in init");
return 0;
}

int __declspec(dllexport) Run (void *p)
{
      const char *str = lr_eval_string(" < name > ");
      lr_output_message("in run and parameter is %s", str);
return 0;
}

int __declspec(dllexport) End (void *p)
{
      lr_output_message("in end");
return 0;
}
} //extern C 終了
```

Visual Basic による仮想ユーザ・スクリプトの作成

Visual Basic を使用して仮想ユーザを作成するには，次の手順を実行します。

1 Micorsoft Visual Basic で新規プロジェクトを作成します。［**ファイル**］＞［**新しいプロジェクト**］を選択します。

2 ［**LoadRunner Virtual User**］を選択します。新しいプロジェクトが 1 つのクラスと 仮想ユーザ用のテンプレートで作成されます。

3 プログラミングを始める前に，プロジェクトを保存しておきます。［**ファイル**］＞［**プロジェクトの上書き保存**］を選択します。

4 オブジェクト・ブラウザ（[**表示**] メニュー）を開きます。「**LoadRunner Vuser**」ライブラリを選択し，仮想ユーザ・クラス・モジュールをダブルクリックしてテンプレートを開きます。テンプレートには，Vuser_Init, Vuser_Run，Vuser_End の 3 つのセクションが含まれています。

```
Option Explicit

Implements Vuser

Private Sub Vuser_Init()
' ここで仮想ユーザの初期化コードを実装する
End Sub

Private Sub Vuser_Run()
' ここで仮想ユーザの主なアクションを示すコードを実装する
End Sub

Private Sub Vuser_End()
' ここで仮想ユーザの終了コードを実装する
End Sub
```

5 これで，クライアント・アプリケーションからコードを追加したり，通常通りプログラミングします。

6 オブジェクト・ブラウザを使って，トランザクション，思考遅延時間，ランデブー，メッセージなど，使用する VuGen の要素をコードに追加します。

7 実行環境の設定とパラメータで，プログラムを拡張します。詳細については，1256 ページ「実行環境の設定とパラメータの設定」を参照してください。

8 仮想ユーザ・スクリプトをビルドします。[**ファイル**] > [**project_name.dll の作成**] を選択します。

プロジェクトは，仮想ユーザ・スクリプト形式（.usr）で保存されます。スクリプトは，プロジェクトと同じディレクトリに作成されます。

実行環境の設定とパラメータの設定

スクリプト用に DLL を作成したら，スクリプト・ファイル（.**usr**）を作成して，その設定を行います。VuGen で提供される **lrbin.bat** ユーティリティを使って，パラメータを定義し，Visual C や Visual Basic を使って作成したスクリプトの実行環境の設定を行います。このユーティリティは，製品のインストール・ディレクトリにある **bin** ディレクトリにあります。

実行環境の設定を行い，スクリプトをパラメータ化するには，次の手順を実行します。

1 製品の **bin** ディレクトリにある **lrbin.bat** をダブルクリックします。［Standalone Vuser Configuration］ダイアログ・ボックスが開きます。

2 ［**File**］＞［**New**］を選択します。**usr** ファイルとなるスクリプトの名前を指定します。このスクリプト名は，DLL を保存したディレクトリの名前と同じでなくてはなりません。

3 ［**Vuser**］＞［**Advanced**］を選択し，［Advanced］ダイアログ・ボックスに DLL の名前を入力します。

4 ［**Vuser**］＞［**Run-time Settings**］を選択して，実行環境の設定を定義します。［実行環境設定］ダイアログ・ボックスは，VuGen のインタフェースに表示されるものと同じです。詳細については，第 13 章「実行環境の設定」を参照してください。

5 ［**Vuser**］＞［**Parameter List**］を選択して，スクリプトにパラメータを定義します。［パラメータ］ダイアログ・ボックスは，VuGen のインタフェースに表示されるものと同じです。詳細については，第 9 章「VuGen パラメータを使った作業」を参照してください。

スクリプトをスタンドアロン・モードで実行して，テストします。［**Vuser**］＞［**Run Vuser**］を選択します。スクリプトの実行中は，仮想ユーザの実行ウィンドウが現れます。

6 ［**File**］＞［**Exit**］を選択して，設定ユーティリティを閉じます。

第 81 章

XML API プログラミング

完全な XML 構造をサポートする仮想ユーザ・スクリプトを作成できます。VuGen は，XML データの検索および操作を可能にする関数を提供します。

本章では，次の項目について説明します。

- XML API プログラミングについて
- XML 文書について
- XML 関数の使用方法
- XML 関数のパラメータの指定
- XML 属性での作業
- XML スクリプトの作成
- 記録されたセッションの拡張

以降の情報は，主に Web，Web サービス，およびワイヤレス仮想ユーザ・スクリプトを対象とします。

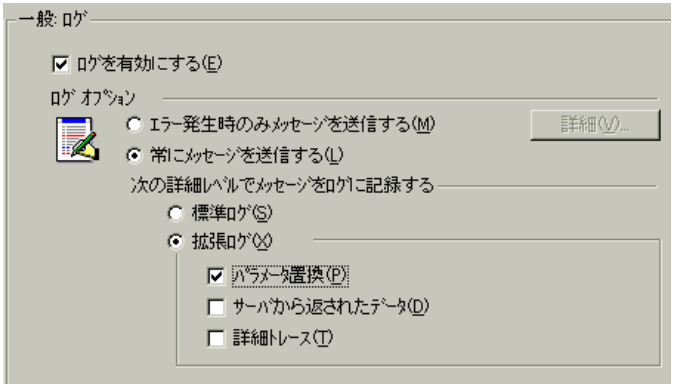
XML API プログラミングについて

VuGen の XML サポート機能により，テスト実行中に XML コードを動的に使用し，値を取得できます。効果的な XML スクリプトを作成するには，次の手順を実行します。

- 使用するプロトコル（通常 Web，Web サービス，またはワイヤレス）でスクリプトを記録します。
- XML の構造をスクリプトにコピーします。
- 動的データと XML 要素の値を取得するには，LR API の XML 関数を追加します。

LR API は，XML Path 言語である XPath を使用して，XML 文書のテキストを操作します。

［実行環境設定］を使用することによって，［再生ログ］ウィンドウに XML 要素の出力値が表示されるようにできます。VuGen には，行番号，一致した件数，値が表示されます。値を表示するには，パラメータ置換を有効にする必要があります。［実行環境設定］ダイアログ・ボックスで［**一般：ログ**］ノードを開き，［**拡張ログ**］を選択して，［**パラメータ置換**］を選択します。詳細については，第 13 章「実行環境の設定」を参照してください。



仮想ユーザ API の XML 関数はすべて，正常に検索された一致の件数か，失敗を表す 0 を返します。

XML 文書について

XML（eXtensible Markup Language）は，ユーザが独自にタグを定義できるマークアップ言語です。これらのタグを使用することによって，タグに挟まれたテキストに意味を与えます。これは，標準の HTML タグ（H1，P，DIV など）とは対照的です。標準の HTML タグはカスタマイズできず，テキストの内容を指定できません。

XML 文書は，多くのノードと分岐を持つツリーで構成されます。XML 文書を構成する部品を表すために，**タグ**，**要素**，**属性**という 3 つの用語がよく使用されます。次の例は，これらの用語を表しています。

```
<acme_org>
    <accounts_dept>
        <employee type='PT'>
            <name>John Smith</name>
            <cubicle>227</cubicle>
            <extension>2145</extension>
        </employee>
    </accounts_dept>
    <engineering_dept>
        <employee type='PT'>
            <name>Sue Jones</name>
            <extension>2375</extension>
        </employee>
    </engineering_dept>
</acme_org>
```

タグとは，左向きと右向きの山括弧の間にあるテキストです。<acme_org>，<employee>，<name> がタグの例です。タグには，開始タグ（<name> など）と終了タグ（</name> など）があります。上記の XML コードの抜粋は，John Smith と Sue Jones という 2 人の従業員がいる Acme という会社を表しています。

要素とは，開始タグおよび終了タグ，そしてそれらのタグに挟まれたすべての内容です。上記の例では，<employee> 要素には，<name>，<cubicle>，<extension> という 3 つの子要素が含まれています。

属性とは，要素の開始タグ内にある，名前と値の組み合わせです。この例では，**type='PT'** が <employee> 要素の属性です。

上記の例では，**name** というタグは **employee** の要素です。それぞれの要素には値があります。たとえば，**name** 要素の値は，「John Smith」という文字列です。

XML 関数の使用方法

以降では，XML ツリーのデータの使用方法について例を示します。いくつかの関数では情報を取得できます。またいくつかの関数では XML ツリーに情報を書き込めます。これらの例では，**Acme** という会社に所属する複数の従業員の名前と内線番号が入っている次の XML ツリーを使用します。

```
<acme_org>
    <accounting_dept>
        <employee type='PT'>
            <name>John Smith</name>
            <extension>2145</extension>
        </employee>
    </accounting_dept>
    <engineering_dept>
        <employee type='PT'>
            <name>Sue Jones</name>
            <extension>2375</extension>
        </employee>
    </engineering_dept>
</acme_org>
```

XML ツリーからの情報の読み取り

XML ツリーから情報を読み取る関数は次のとおりです。

lr_xml_extract	XML 文字列から XML 文字列フラグメントを抽出します。
lr_xml_find	XML 文字列に対するクエリを実行します。
lr_xml_get_values	クエリで検出された XML 要素の値を取得します。

クエリを使用して特定の値を取得するには，パス形式で親ノードと子ノードのタグを指定します。

たとえば，Accounting 部門の従業員の名前を取得するには，次の文字列を使用します。

```
lr_xml_get_values("XML={XML_Input_Param}",
"ValueParam=OutputParam",
"Query=/acme_org/accounting_dept/employee/name",
LAST);
```

拡張ログ機能が有効な場合，［再生ログ］ウィンドウには，この関数の出力が表示されます。

```
Output:
Action.c(20):"lr_xml_get_values" was successful, 1 match processed
Action.c(25):Query result = John Smith
```

XML 構造への書き込み

XML ツリーに値を書き込む関数は次のとおりです。

lr_xml_delete	XML 文字列からフラグメントを削除します。
lr_xml_insert	新しい XML フラグメントを XML 文字列に挿入します。
lr_xml_replace	XML 文字列のフラグメントを置き換えます。
lr_xml_set_values	クエリで検出された XML 要素の値を設定します。
lr_xml_transform	XML データに XSL（Extensible Stylesheet Language）変換を適用します。

最もよく使用される**書き込み**関数は **lr_xml_set_values** です。この関数は，XML 文字列の指定された要素の値を設定します。次の例では，**lr_xml_set_values** を使用して，XML 文字列の 2 つの **employee** 要素の内線番号を変更しています。

まず，**XML_Input_Param** というパラメータに XML 文字列を保存します。2 つの値を検索して置き換えます。そこで，**ExtensionParam_1** と **ExtensionParam_2** という新しい 2 つのパラメータを用意し，それらの値を新しい 2 つの内線番号 1111 および 2222 に設定します。

lr_xml_set_values には，**ExtensionParam_1** および **ExtensionParam_2** の値を受け取る「ValueName=ExtensionParam」という引数が含まれています。2 人の従業員の現在の内線番号を，これらのパラメータの値，1111 および 2222 で置き換えます。その後 **OutputParam** の値を評価し，新しい内線番号に確かに置き換えられたことを証明します。

```
Action() {

    int i, NumOfValues;
    char buf[64];

    lr_save_string(xml_input, "XML_Input_Param"); // 入力をパラメータとして保存する
    lr_save_string("1111", "ExtensionParam_1");
    lr_save_string("2222", "ExtensionParam_2");

    lr_xml_set_values("XML={XML_Input_Param}",
        "ResultParam=NewXmlParam", "ValueParam=ExtensionParam",
        "SelectAll=yes", "Query=//extension", LAST);

    NumOfValues= lr_xml_get_values("XML={NewXmlParam}",
        "ValueParam=OutputParam", "Query=//extension",
        "SelectAll=yes", LAST);

    for (i = 0; i < NumOfValues; i++) {/* MultiParam の複数の値を出力する */

        sprintf(buf, "Retrieved value %d : {OutputParam_%d}", i+1, i+1);
        lr_output_message(lr_eval_string(buf));
    }

    return 0;
}
```

Output:

```
Action.c(40): Retrieved value 1: 1111
Action.c(40): Retrieved value 2: 2222
```

XML 関数のパラメータの指定

ほとんどの XML API 関数では，**XML 要素**と**クエリ**を指定する必要があります。また，すべての結果を取得するか，それとも 1 つの結果を取得するか指定できます。

XML 要素の定義

検索する XML 要素を定義するには，XML 要素をそのまま文字列として指定するか，XML 要素が含まれるパラメータを指定します。次の例では，XML 入力文字列をそのまま文字列として定義する場合を示します。

```
"XML=<employee>JohnSmith</employee>"
```

また，**XML** 入力文字列には，XML データが格納されているパラメータを使用できます。次に例を示します。

```
"XML={EmployeeNameParam}"
```

XML ツリーの検索

XML タグに含まれている値（従業員の内線番号など）を検索するとします。必要とする値に対するクエリを作成します。クエリには，要素の場所と，取得または設定する要素を指定します。指定したパスにより，検索範囲が特定のタグに限定されます。また，ルートの下のすべてのノードで特定の種類の要素をすべて検索することもできます。

特定のパスを指定するには，「"Query=/ ＜ **XML フル・パス名**＞ / ＜**要素名**＞ "」を指定します。

同じ名前の要素をすべてのノードの下で検索するには，「"Query=// ＜**要素名**＞ "」を指定します。

VuGen における XML 関数の実装では，クエリの範囲は XML ツリー全体です。ツリー情報は，**xml** 引数の値として仮想ユーザ API 関数に送られます。

クエリの複数の検索

XML 要素に対してクエリを実行すると，標準では最初に一致したものだけが返されます。クエリで複数の値を取得するには，関数内で "SelectAll=yes" 属性を指定します。VuGen は，複数のパラメータを表すために _ ＜**連番**＞という形式の接尾辞を追加します。たとえば，**EmployeeName** という名前のパラメータを定義した場合は，**EmployeeName_1**，**EmployeeName_2**，**EmployeeName_3** などが作成されます。

lr_xml_set_values("XML={XML_Input_Param}",
"ResultParam=NewXmlParam", "ValueParam=ExtensionParam",
"SelectAll=yes", "Query=//extension", LAST);

パラメータに**書き込む**関数を使用すれば，書き込んだ後にパラメータの値を評価できます。たとえば次のコードでは，クエリによる複数の検索結果を取得して出力しています。

```
NumOfValues = lr_xml_get_values("Xml={XmlParam}", "Query=//name",
        "SelectAll=yes", "ValueParam=EmployeeName", LAST);
```

パラメータから値を**読み取る**関数の場合，パラメータの値は事前に定義しておく必要があります。また，パラメータは，**＜パラメータ名＞_＜連番＞**という形式（たとえば **Param**_1，**Param**_2，**Param**_3 など）を使用する必要もあります。このようなパラメータの集合をパラメータ・セットとも言います。

次の例では，**lr_xml_set_values** はパラメータ・セットから値を読み取り，その値を XPath クエリで使用しています。従業員の内線番号を表すパラメータ・セットには，**ExtensionParam** という名前が付いています。このパラメータ・セットには，**ExtensionParam_1** および **ExtensionParam_2** という 2 つのメンバがあります。**lr_xml_set_values** 関数は XML 入力文字列を検索し，最初に一致した値を 1111 に，2 番目に一致した値を 2222 に設定します。

```
lr_save_string("1111", "ExtensionParam_1");
lr_save_string("2222", "ExtensionParam_2");

lr_xml_set_values("XML={XML_Input_Param}",
      "ResultParam=NewXmlParam", "ValueParam=ExtensionParam",
      "SelectAll=yes", "Query=//extension", LAST);
```

XML 属性での作業

VuGen では属性がサポートされています。要素を操作する場合と同じように，簡単な表現を使用して XML の要素とノードの属性を操作できます。特定の属性，または特定の値を持つ属性を変更することもできます。

次の例では，**lr_xml_delete** を使って，**name** 属性を持っている最初の **cubicle** 要素を削除しています。

```
lr_xml_delete("Xml={ParamXml}",
                        "Query=//cubicle/@name",
                            "ResultParam=Result",
                        LAST
    );
```

次の例では，**lr_xml_delete** を使って，**Paul** という値の **name** 属性を持っている最初の cubicle 要素を削除しています。

```
lr_xml_delete("Xml={ParamXml}",
            "Query=//cubicle/@name="Paul",
            "ResultParam=Result",
            LAST
    );
```

XML スクリプトの作成

最初に，使用するプロトコルで新しいスクリプトを作成します。そのプロトコルでセッションを記録することも，また記録せずにスクリプト全体をプログラミングすることもできます。次のように，スクリプトの Actions セクションを作成します。

- XML 入力の宣言
- Actions セクション

XML 入力セクションには，入力変数として使用する XML ツリーを含めます。XML ツリーを **char** 型の変数として定義します。次に例を示します。

```
char *xml_input=
"<acme_org>"
    "<employee>"
        " <name>John Smith</name>"
        "<cubicle>227</cubicle>"
        "<extension>2145</extension>"
    "</employee>"
    "<employee>"
        "<name>Sue Jones</name>"
        "<cubicle>227</cubicle>"
        "<extension>2375</extension>"
    "</employee>"
"</acme_org>";
```

Action セクションには，変数の評価，および要素の値に対するクエリを含めます。次の例では，**lr_save_string** を使って XML 入力文字列を評価しています。入力変数を対象に，従業員名と内線番号を検索しています。

```
Action() {

/* 入力をパラメータとして保存する */
    lr_save_string(xml_input, "XML_Input_Param");

    /* クエリ 1 - 指定された要素から従業員名を取得する */
    lr_xml_get_values("XML={XML_Input_Param}",
        "ValueParam=OutputParam",
        "Query=/acme_org/employee/name", LAST);

    /* クエリ 2 - ルート以下のすべてのパスで内線番号を取得する */
    lr_xml_get_values("XML={XML_Input_Param}",
        "ValueParam=OutputParam",
        "Query=//extension", LAST);

    return 0;
}
```

記録されたセッションの拡張

セッションを記録し，必要な XML 関数と仮想ユーザ API 関数を手作業で追加することによって，XML スクリプトを作成できます。

次の例では，記録されたセッションを仮想ユーザ API 関数を使って拡張する方法を示します。記録された関数は，太字で示した **web_submit_data** だけです。

最初のセクションには，SOAP メッセージを表す変数 SOAPTemplate が XML 入力宣言として含まれています。

```
#include "as_web.h"

// SOAP メッセージ
const char*pSoapTemplate=
        "<soap:Envelope xmlns:soap=¥"http://schemas.xml-
soap.org/soap/envelope/¥">"
                "<soap:Body>"
                    "<SendMail xmlns=¥"urn:EmailIPortTypeInft-IEmailSer-
vice¥"/>"
                "</soap:Body>"
        "</soap:Envelope>";
```

次のセクションは，ユーザのアクションを表しています。

```
Action1()
{
    // 応答の本体を取得する
    web_reg_save_param("ParamXml", "LB=", "RB=", "Search=body",
LAST);

    // HTTP GET で天気をフェッチする
    web_submit_data("GetWeather",
                    "Action=http://glkev.net.innerhost.com/glkev_ws/
                        WeatherFetcher.asmx/GetWeather",
                        "Method=GET",
                        "EncType=",
                        "RecContentType=text/xml",
                        "Referer=http://glkev.net.innerhost.com
                      /glkev_ws/WeatherFetcher.asmx?op=GetWe
ather",
        "Snapshot=t2.inf",
        "Mode=HTTP",
        ITEMDATA,
        "Name=zipCode", "Value=10010", ENDITEM,
        LAST);

    // City の値を取得する
    lr_xml_get_values("Xml={ParamXml}",
                      "Query=City",
                      "ValueParam=ParamCity",
                      LAST
    );

    lr_output_message(lr_eval_string("***** City = {ParamCity} *****"));

    // State の値を取得する
    lr_xml_get_values("Xml={ParamXml}",
                      "Query=State",
                      "ValueParam=ParamState",
                      LAST
    );

    lr_output_message(lr_eval_string("***** State = {ParamState} *****"));
```

```
// テンプレートを使用して複数の値を一度に取得する
lr_xml_get_values_ex("Xml={ParamXml}",
                                "Template="
                                    "<Weather>"
                                        "<Time>{ParamTime}</Time>"
                                        "<Temperature>{ParamTemp}</Tempera-
ture>"
                                        "<Humidity>{ParamHumid}</Humid-
ity>"
                                        "<Conditions>{ParamCond}</Condi-
tions>"
                                    "</Weather>",
                                LAST
    );

    lr_output_message(lr_eval_string("***** Time = {ParamTime},
Temperature =
                                                {ParamTemp}, "
                                        "Humidity = {ParamHumid},
Conditions =                                    {ParamCond}
*****"));

    // 人に読める形式で予報を生成する
    lr_save_string(lr_eval_string("\r\n\r\n*** Weather Forecast for {ParamCity}, {ParamState}
***\r\n"
                                    "¥tTime:{ParamTime}¥r¥n"
                                    "¥tTemperature: {ParamTemp} deg. Fahr-
enheit¥r¥n"
                                    "¥tHumidity:{ParamHumid}¥r¥n"
                                    "¥t{ParamCond} conditions expected¥r¥n"
                                "¥r¥n"),
                                    "ParamForecast"
    );

    // SOAP テンプレートをパラメータに保存する
    lr_save_string(pSoapTemplate, "ParamSoap");
```

```
// 要求の本体を SOAP テンプレートに挿入する
      lr_xml_insert("Xml={ParamSoap}",
                              "ResultParam=ParamRequest",
                              "Query=Body/SendMail",
                              "position=child",
                              "XmlFragment="
                                        "<FromAddress>taurus@merc-int.com</Fro-
mAddress>"
                                        "<ToAddress>support@merc-int.com</ToAd-
dress>"
                                        "<ASubject>Weather Forecast</ASubject>"
                                        "<MsgBody/>",
                              LAST
      );

//
//          "<soap:Envelope xmlns:soap=¥"http://schemas.xml-
soap.org/soap/envelope/\">"
//                    "<soap:Body>"
//                        "<SendMail xmlns=¥"urn:EmailIPortTypeInft-IEmailSer-
vice¥"/>"
//                                  "<FromAddress>taurus@merc-int.com</Fro-
mAddress>"
//                                  "<ToAddress>support@merc-int.com</ToAd-
dress>"
//                                  "<ASubject>Weather Forecast</ASubject>"
//                                  "<MsgBody/>"
//                        "</SendMail>"
//                    "</soap:Body>"
//          "</soap:Envelope>";
//

      // 実際の予報のテキストを挿入する
      lr_xml_set_values("Xml={ParamRequest}",
                                "ResultParam=ParamRequest",
                                "Query=Body/SendMail/MsgBody",
                                "ValueParam=ParamForecast",
                                 LAST);
```

```
	// SOAP 用のヘッダーを追加する
	web_add_header("SOAPAction", "urn:EmailIPortTypeInft-IEmailSer-
vice");

	// 応答の本体を取得する
	web_reg_save_param("ParamXml", "LB=", "RB=", "Search=body",
LAST);

	// SOAP 要求を使用して予報を受信者に送信する
	web_custom_request("web_custom_request",
		"URL=http://webservices.matlus.com/scripts/emailwebservice.dll/soap/IEmailser-
vice",
		"Method=POST",
		"TargetFrame=",
		"Resource=0",
		"Referer=",
		"Body={ParamRequest}",
		LAST);

	// メールが送信されたことを確認する
	lr_xml_find("Xml={ParamXml}",
				"Query=Body/SendMailResponse/return",
				"Value=0",
				LAST
	);

	return 0;
}
```

第 82 章

VuGen のデバッグのヒント

本章では，エラーのない仮想ユーザ・スクリプトを作成できるように，詳細なデバッグ情報を取得する方法をいくつか紹介します。

- 一般的なデバッグのヒント
- C 関数を使用した追跡
- 付加的な C 言語のキーワードの追加
- 再生出力の検証
- データベース・アプリケーションのデバッグ
- Oracle Applications を使った作業
- Oracle 2-Tier 仮想ユーザでの一般的な問題の解決方法
- 2-tier データベースのスクリプト作成のヒント
- PeopleSoft-Tuxedo スクリプトの実行

一般的なデバッグのヒント

VuGen は，通常のテキスト・エディタとして使用できます。VuGen で，任意のテキスト・ファイルを開いて編集できます。再生中，下の出力ウィンドウにエラー・メッセージが表示されている場合は，その上でダブルクリックすると，VuGen は問題の原因となっているテキスト行にカーソルを移動します。また，エラー・コードにカーソルを置いて F1 キーを押すと，オンライン・ヘルプのエラー・コードの説明が表示されます。

C 関数を使用した追跡

C インタプリタの追跡オプション（バージョン 230 以上）を使用して，仮想ユーザ・スクリプトをデバッグできます。ci_set_debug ステートメントを使って，スクリプト内の特定の位置で，追跡とデバッグのオン / オフを切り替えることができます。

ci_set_debug(ci_this_context, int debug, int trace);

たとえば，スクリプトに次のステートメントを追加できます。

```
ci_set_debug(ci_this_context, 1, 1) /* 追跡とデバッグをオンにする */
ci_set_debug(ci_this_context, 0, 0) /* 追跡とデバッグをオフにする */
```

付加的な C 言語のキーワードの追加

VuGen で C スクリプトを実行すると，VuGen のパーサは組み込み C インタプリタを使ってスクリプト内の関数を解析します。標準パーサのライブラリに含まれていないキーワードを追加できます。標準設定では，インストール中に **size_t** や **DWORD** といった一般的な C++ キーワードが追加されます。リストを編集して，環境に合ったキーワードを追加します。

キーワードを追加するには，次の手順を実行します。

1 vugen_extra_keywords.ini ファイルを開きます。このファイルはお使いのコンピュータの< Windows >または< Windows > ¥System ディレクトリにあります。

2 EXTRA_KEYWORDS_C セクションに，C インタプリタ用のキーワードを追加します。

このファイルの形式は次のとおりです。

```
[EXTRA_KEYWORDS_C]
FILE=
size_t=
WORD=
DWORD=
LPCSTR=
```

再生出力の検証

再生出力を見ます（VuGen から，あるいは VuGen ドライバの出力を表示する **output.txt** ファイルから）。また，より詳細な再生出力を得るために，VuGen の実行環境の設定オプションで，ログの記録を拡充することもできます。

データベース・アプリケーションのデバッグ

以降のヒントは，データベース・アプリケーション（Oracle，ODBC，および Ctlib）に適用されます。

- デバッグ情報の生成
- コンパイラ情報の検証
- コード生成情報
- 前処理とコンパイルの情報

デバッグ情報の生成

注：本項で説明する情報の大部分は，VuGen のユーザ・インタフェースを使って表示するように設定できます。

VuGen には，インスペクタ「エンジン」が含まれています。**¥WINDOWS_DIR¥vugen.ini** を次のように編集して，VuGen レコーダが「インスペクタ」出力を作成するようにできます。

```
[LogMode]EnableAscii=ASCII_LOG_ON
```

このオプションが有効になっている場合，VuGen は記録終了時に Data ディレクトリに **vuser.asc** ファイルを作成します。このオプションは，デバッグの目的に限って使うべきものです。出力ファイルが非常に大きくなり（数 MB），マシンのパフォーマンスとディスク領域に深刻な影響を及ぼす可能性があるからです。

ODBC ベースのアプリケーションの場合は，［ODBC データ・ソース アドミニストレータ］（Windows の［コントロール パネル］にあります）で同様の追跡出力を得るように設定できます。［ODBC データ・ソース アドミニストレータ］を開いて，［トレース］タブで［トレースの開始］をクリックします。同様に，ODBC Developer Kit には呼び出しの追跡を行うスパイ・ユーティリティが用意されています。

詳細なデバッグ情報を有効にするには，¥WINDOWS_DIR¥vugen.ini ファイルに次のセクションを追加します。

```
[INSPECTOR]
TRACE_LEVEL=3
TRACE_FILENAME=c:¥tmp¥sqltrace.txt
```

sqltrace.txt ファイルには，記録中に行われたフック呼び出しについての有用な内部情報が含まれます。trace_level は 1 から 3 まであり，3 は最も詳細なデバッグ・レベルを表します。VuGen バージョン 5.02 以上では，ユーザ・インタフェースから追跡レベルを設定できます。

コンパイラ情報の検証

コード生成，前処理，コンパイルの各段階についての情報を表示して，エラーの原因を特定できます。

コード生成情報

Data ディレクトリ内の **vuser.log** ファイルを見ます。このファイルには，コード生成段階のログが含まれており，lrd 記録（すなわちすべてのデータベース・プロトコル）が終わるたびに，自動的に作成されます。

次にログ・ファイルの例を示します。

```
lrd_init: OK
lrd_option: OK
lrd_option: OK
lrd_option: OK
Code generation successful
```

```
lrd_option: OK
lrd_end: OK
```

いずれかのメッセージが OK（成功）ではない場合は，コード生成中に問題が生じています。

前処理とコンパイルの情報

実行中，VuGen は前処理とコンパイル処理の両方についての情報を表示します。

Oracle Applications を使った作業

Oracle Applications は，2-tier（「ファット」クライアント）パッケージのアプリケーションで，35 ものさまざまなモジュール（Oracle Human Resources，Oracle Financials など）で構成されています。

Oracle Applications 用の仮想ユーザの記録と再生のために，いくつか知っておくべきことがあります。

- 一般的なスクリプトには，何千ものイベント，バインド，およびアサインが含まれています。
- 一般的なスクリプトには，各ユーザ・セッションに多くの db 接続が含まれています。
- スクリプトには，ほとんどの場合相関したクエリが必要です。
- Oracle Applications のクライアントは 16 ビットのみです（Oracle Developer 2000 で開発されたもの）。つまり，デバッグに際して Oracle 32 ビット・クライアントがなければ，VuGen の Force 16-bit オプションを使う必要があります。

  新しいウィンドウが作成されると，アプリケーションは，表示用にファイル・システムから .xpf ファイルを取得します。VuGen はクライアント / サーバ・レベルで記録を行うため，現在はこれを考慮に入れていません。従って，パフォーマンスの測定はかなり不正確になります。多くの場合，パフォーマンスの問題はクライアントとファイル・サーバの間のボトルネックに関係するものだからです。現在，この問題の解決に向け，検討中です。

Oracle 2-Tier 仮想ユーザでの一般的な問題の解決方法

本項では，Oracle 仮想ユーザを扱っているときに生じるいくつかの一般的な問題と，その解決策を示します。

ORA-20001 と ORA-06512

lrd_stmt に次の PL/SQL ブロックが含まれている場合，再生中にエラー ORA-20001 と ORA-06512 が発生します。fnd_signon.audit_responsibility(...)

このステートメントが再生中に失敗するのは，新しい接続をするたびに一意のサインオン番号が割り当てられるためです。

解決策

この問題を解決するには，サインオン番号を扱う新しい相関ツールを使用する必要があります。サインオン番号は，ステートメント内で 2 番目に割り当てられる値です。

相関候補の値を検索した後，失敗したステートメントの 2 番目の lrd_assign_bind() の値を強調表示します。「相関クエリ」ウィンドウでは，値が実際に記録されたステートメントと同じ順番で表示されない場合があります。

置換する値が含まれるグリッドは，次の PL/SQL ブロックの含まれる lrd_stmt の後に現れます。fnd_signon.audit_user(...).

注：サインオン番号は接続ごとに一意なので，記録する新しい接続のそれぞれについて相関を行う必要があります。

解決策の例

次のステートメントは，2 番目の値「1498224」がそれぞれの新しい接続に一意のサインオン番号なので，再生で失敗します。

```
lrd_stmt(Csr6, "begin fnd_signon.audit_responsibility(:s,:l,:f,:a,:r,:t,:p)"
       "; end;", -1, 1, 1, 0);
   lrd_assign_bind(Csr6, "s", "D", &s_D216, 0, 0, 0);
   lrd_assign_bind(Csr6, "l", "1498224", &l_D217, 0, 0, 0);
   lrd_assign_bind(Csr6, "f", "1", &f_D218, 0, 0, 0);
   lrd_assign_bind(Csr6, "a", "810", &a_D219, 0, 0, 0);
   lrd_assign_bind(Csr6, "r", "20675", &r_D220, 0, 0, 0);
```

```
lrd_assign_bind(Csr6, "t", "Windows PC", &t_D221, 0, 0, 0);
lrd_assign_bind(Csr6, "p", "", &p_D222, 0, 0, 0);
lrd_exec(Csr6, 1, 0, 0, 0, 0);
```

このサインオン番号は，lrd_stmt で「fnd_signon.audit_user」を手がかりに見つけることができます。最初のプレースホルダの値「a」はそのままにしておきます。「a」への入力値はいつも「0」ですが，出力は要求された値です。

変更後のコード：

```
lrd_stmt(Csr4, "begin fnd_signon.audit_user(:a,:l,:u,:t,:n,:p,:s); end;", -1, 1, 1, 0);
lrd_assign_bind(Csr4, "a", "0", &a_D46, 0, 0, 0);
lrd_assign_bind(Csr4, "l", "D", &l_D47, 0, 0, 0);
lrd_assign_bind(Csr4, "u", "1001", &u_D48, 0, 0, 0);
lrd_assign_bind(Csr4, "t", "Windows PC", &t_D49, 0, 0, 0);
lrd_assign_bind(Csr4, "n", "OraUser", &n_D50, 0, 0, 0);
lrd_assign_bind(Csr4, "p", "", &p_D51, 0, 0, 0);
lrd_assign_bind(Csr4, "s", "14157", &s_D52, 0, 0, 0);
lrd_exec(Csr4, 1, 0, 0, 0, 0);

lrd_save_value(&a_D46, 0, 0, "saved_a_D46");
Grid0(17);

lrd_stmt(Csr6, "begin fnd_signon.audit_responsibility(:s,:l,:f,:a,:r,:t,:p)"
    "; end;", -1, 1, 1, 0);
lrd_assign_bind(Csr6, "s", "D", &s_D216, 0, 0, 0);
lrd_assign_bind(Csr6, "l", "<saved_a_D46>", &l_D217, 0, 0, 0);
lrd_assign_bind(Csr6, "f", "1", &f_D218, 0, 0, 0);
lrd_assign_bind(Csr6, "a", "810", &a_D219, 0, 0, 0);
lrd_assign_bind(Csr6, "r", "20675", &r_D220, 0, 0, 0);
lrd_assign_bind(Csr6, "t", "Windows PC", &t_D221, 0, 0, 0);
lrd_assign_bind(Csr6, "p", "", &p_D222, 0, 0, 0);
lrd_exec(Csr6, 1, 0, 0, 0, 0);
```

大きな数値の処理

大きな数（NUMBER データ型）は，GRID と ASCII ファイルでは異なる形式で表示されます。この違いにより，相関用に保存する値の検索時に数値を特定するのが困難になります。

たとえば，グリッド内に 1000003 と表示される値がある場合，これは記録ログ（ASCII ファイル）では 1e+0006 と表示されます。

対処方法

再生中にエラーが生じ，相関ツールが前回の結果の中で値を見つけられない場合は，この値が別の形式で表現されているものとしてグリッドの中を探します。

ORA-00960

このエラーは，記録されたスクリプトのカラム名が一意でない場合に生じます。たとえば，次のような場合です。

```
lrd_stmt(Csr9,"SELECT UOM_CODE, UOM_CODE, DESCRIPTION FROM "
        "MTL_UNITS_OF_MEASURE "
        "WHERE NVL(DISABLE_DATE, SYSDATE + 1) > "
        "SYSDATE ORDER BY UOM_CODE", -1, 1, 1, 0);
```

この場合，次のエラーが返されます。

```
"lrdo.c/fjParse:"oparse" ERROR return-code=960, oerhms=ORA-
00960:ambiguous column naming in select list".
```

対処方法

少なくとも 1 つの一意でないカラムに別名を追加し，この別名を新しい一意の名前にして使うように，ステートメントを変更します。次に例を示します。

```
lrd_stmt(Csr9,"SELECT UOM_CODE,UOM_CODE second, DESCRIPTION
FROM"
        "MTL_UNITS_OF_MEASURE "
        "WHERE NVL(DISABLE_DATE, SYSDATE + 1) > "
        "SYSDATE ORDER BY UOM_CODE", -1, 1, 1, 0);
```

別の対処方法：lrd ステートメントから ORDER BY を削除します。

ORA-2002

このエラーは，開いていないカーソルを使用しようとしたときに発生します。これは，複数の反復でユーザを再生し，スクリプトの複数のセクションに記録したときに生じます。

具体的には，カーソルが vuser_init セクションで開き，Actions セクションで閉じた場合に，カーソルを使用しようとすると 2 回目の反復でこのエラーが生じます。これは，カーソルが閉じられており，再び開かれていないからです。

たとえば，次のような場合です。vuser_init セクションに **lrd_open_cursor** があり，Actions セクションに **lrd_close_cursor** がある場合，このユーザの再生で複数回の反復を行うと，2 回目の反復でエラーが生じます。これは，開いていないカーソルを使用しようとしたためです（カーソルを最初の反復で閉じ，2 回目の反復では開いていないためです）。

対処方法

この問題を最も簡単に解決するには，問題のカーソルの **lrd_close_cursor** または **lrd_close_connection** を **vuser_end** セクションに移動します。

データベース・プロトコル (lrd)

記録された非同期操作の再生はサポートされていません。

クライアント・バージョンの誤り

実行している Oracle クライアントのバージョンが正しくない場合，次のエラーが返されます。

"Error:lrdo_open_connection:"olog" LDA/CDA return-code_019:unable to allocate memory in the user side"

対処方法

製品の bin ディレクトリにあるライブラリ情報を **lrd.ini** ファイルで修正します。このファイルには記録および再生中にどのバージョンの Mercury データベース・サポートがロードされるかを示す設定情報が含まれています。ファイルに各タイプのホストごとのセクションがあります。たとえば，**lrd.ini** ファイル内の HP/UX 上の Oracle セクションは次のようになります。

```
[ORACLE_HPUX]
;816=liblrdhpo816.sl
;81=liblrdhpo81.sl
;80=liblrdhpo80.sl
73=liblrdhpo73.sl
72=liblrdhpo72.sl
```

この設定により，クライアントが Oracle 8.1.6 を使用していれば Mercury ライブラリ liblrdhpo816.sl を，Oracle 8.1.5 を使用していれば，liblrdhpo81.sl を仮想ユーザが使用するということがわかります。

UNIX 上の再生では，lrd **ini** ファイルで使用するデータベースの正しいバージョンを表示するようにします。Oracle 8.1.5 を使用して HP/UX 用仮想ユーザを再生するとします。その場合，Oracle のそのほかのバージョンを示す行は行の先頭を「;（セミコロン）」でコメントアウトします。すると lrd.ini ファイルは次のようになります。

```
[ORACLE_HPUX]
;816=liblrdhpo816.sl
81=liblrdhpo81.sl
;80=liblrdhpo80.sl
73=liblrdhpo73.sl
72=liblrdhpo72.sl
```

アプリケーションが lrd.ini ファイルに記されている DLL を使用していない場合，Win32 も変更します。たとえば，PowerBuilder 6.5 は Oracle 8.0.5 を使用しますが，DLL は ora803.dll を使用し，ora805.dll は使用しません。その場合，ORACLE_WINNT の項目で 805 および 804 をコメントアウトするか，805 のセクションを，

```
805=lrdo32.dll+ora805.dll
```

から次のように変更します。

```
805=lrdo32.dll+ora803.dll
```

2-tier データベースのスクリプト作成のヒント

本項では 2-tier データベース・スクリプトのための解決策を示します。Siebel 固有の問題の解決策については，1288 ページ「Siebel 固有のスクリプト作成のヒント」を参照してください。

質問 1：アプリケーション自体は同じ値を扱えるのに，データ駆動のスクリプトで失敗する原因は？

回答：データ値の後に続く空白が原因である可能性があります。GUI に直接入力するデータ値ではおそらく切り詰められているとしても，データ・ファイルから手作業で除去する必要があります。タブ区切りファイルでは，後続の空白が見えにくく，問題が発見しづらくなります。一般に，カンマ区切りファイルをお勧めします。Excel を使って問題がないかどうか確かめることができます。

質問 2 : 2 回目の反復でカーソル状態が無効という SQL エラーが発生する原因は？

回答 : **lrd_close_cursor** 関数が生成されていないか，スクリプトの **action** セクションではなく，**end** セクションに生成されている可能性があります。スクリプトを反復できるようにするには，カーソル・クローズ関数を追加するか，**end** セクションから移動する必要があります。

反復のたびに新しいカーソルを開くのは資源効率の点から好ましくありません。従って，最初の反復の **actions** セクションで 1 回だけカーソルを開くことをお勧めします。それから［Iteration Number］タイプを使用して，反復番号を文字列として含む新しいパラメータを追加します。このパラメータに **IterationNum** パラメータという名前を付けます。次に **actions** セクション内で新しいカーソルを開く呼び出し

```
lrd_open_cursor(&Csr1, Con1, 0);
```

を次のように置換します。

```
if (!strcmp(lr_eval_string(" < IterationNum > "), "1"))
   lrd_open_cursor(&Csr1, Con1, 0);
```

質問 3 : vdf.h ファイルのデータ宣言が原因で VuGen で生成したコードのコンパイルができない場合の修正方法は？

回答 : 問題は，おそらく，VuGen でサポートされていない SQL のデータ型です。Microsoft SQL では多くの場合，**vdf.h** ファイル内の未定義エラー・メッセージを「DT_SZ」（ヌル終端文字列）に置き換えれば回避できます。これは実際のデータ型ではありませんが，VuGen でそのスクリプトを正常にコンパイルできます。カスタマー・サポートに問題を通知し，元のスクリプトをお送りください。

質問 4 : LRD Error 2048 の意味は？

回答 : VuGen が失敗します。記録時の割り当てよりも長い変数をバインドしようとしているためです。**vdf.h** ファイルで変数定義を拡大して，データベースから長い文字列を受け取れるようにします。このファイルで一意の数値識別子を検索します。その定義と長さがわかります。長さは構成要素の内 3 つ目の要素です。この長さを増やせばスクリプトを正常に再生できるようになります。

たとえばスクリプト内に次の行があるものとします。

```
lrd_assign(&_2_D354, " < ROW_ID > ", 0, 0, 0);
```

vdf.h ファイルで **_2_D354** を検索すると次のようになっています。

```
static LRD_VAR_DESC _2_D354 = {
        LRD_VAR_DESC_EYECAT, 1, 10, LRD_BYTYPE_ODBC,
        {0, 0, 0},  DT_SZ, 0, 0, 15, 12};
```

これを次のように変更します。

```
static LRD_VAR_DESC _2_D354 = {
        LRD_VAR_DESC_EYECAT, 1, 12, LRD_BYTYPE_ODBC,
        {0.0, 0, 0},  DT_SZ, 0, 0, 15, 12};
```

LRD_VAR_DESC の定義全体は **lrd.h** ファイルにあります。これを見つけるには「typedef struct LRD_VAR_DESC」で検索します。

質問 5 : ODBC および Oracle の使用で UPDATE，INSERT，DELETE によって影響を受けた行数を取得する方法は？

回答 : lrd 関数を使用して情報を取得します。ODBC には **lrd_row_count** 関数を使用します。構文は次のとおりです。

```
int rowcount;
   .
   .
   .
lrd_row_count(Csr33, &rowcount, 0);
```

lrd_row_count 関数は関連するステートメント実行の直後に使用します。

Oracle には **lrd_exec** の 4 番目の引数を使用します。

```
lrd_exec(Csr19, 1, 0, &rowcount, 0, 0);
```

Oracle の OCI 8 を使用している場合は，**lrd_ora8_exec** の 5 番目の引数を使用します。

```
lrd_ora8_exec(OraSvc1, OraStm3, 1, 0, &uliRowsProcessed, 0, 0, 0, 0, 0);
```

質問 6：キーの重複割り当て違反を防ぐ方法は？

回答：挿入を実行するときに重複キー違反が生じることがあります。問題を識別するために 2 つの記録を比較して，主キーを見つけることができます。このステートメントまたはそれ以前に出てきた UPDATE または INSERT ステートメントで相関クエリが使われていたかどうかチェックします。データ辞書を使って一意制約に違反しているカラムを見つけることができます。

Oracle では，一意制約違反が生じると，次のメッセージが表示されます。

ORA-00001:unique constraint (SCOTT.PK_EMP) violated

この例では，SCOTT は関連する一意インデックスの所有者で，PK_EMP がそのインデックス名です。SQL*Plus を使用してデータ辞書のクエリを行い，カラムを見つけます。このクエリのパターンは次のようになります。

select column_name from all_ind_columns where index_name = ' < IndexName > and index_owner = ' < IndexOwner > ';

select column_name from all_ind_columns where index_name = 'PK_EMP' and index_owner = 'SCOTT';

データベースに挿入された値が新しいため，以前のクエリでは見つかっていないかもしれませんが，以前のクエリよりも 1 つ多い戻り値として，以前のクエリの結果と関係していることがあります。

Microsoft SQL サーバでは次のいずれかのメッセージが表示されます。

Cannot insert duplicate key row in object 'newtab' with unique index 'IX_newtab'.

Violation of UNIQUE KEY constraint 'IX_Mark_Table'.Cannot insert duplicate key in object 'Mark_Table'.

Violation of PRIMARY KEY constraint 'PK_NewTab'.Cannot insert duplicate key in object 'NewTab'.

Query Analyzer を使用して，どのカラムがキーまたはインデックスで使用されているのかを検索します。このクエリのパターンは次のようになります。

```
select C.name
    from sysindexes A, sysindexkeys B, syscolumns C
    where C.colid = B.colid and C.id = B.id and
    A.id = B.id and A.indid = B.indid
    and A.name = ' < IndexName > ' and A.id = object_id(' < TableName > ')
```

```
select C.name
    from sysindexes A, sysindexkeys B, syscolumns C
    where C.colid = B.colid and C.id = B.id and
    A.id = B.id and A.indid = B.indid
    and A.name = 'IX_newtab' and A.id = object_id('newtab')
```

DB2 では次のメッセージが表示されます。

SQL0803N One or more values in the INSERT statement, UPDATE statement, or foreign key update caused by a DELETE statement are not valid because they would produce duplicate rows for a table with a primary key, unique constraint, or unique index.SQLSTATE=23505

まだ問題が続くようであれば，記録時，再生時のスクリプトの両方で更新および挿入で変更した行番号を確認します。UPDATE では，WHERE 句の条件式が間違っているために再生時に行を変更できないことがよくあります。これは直接エラーになりませんが，テーブルが正確に更新されず，後続の SELECT がクエリの相関時に間違った値を選択する原因になります。

また，マルチ・ユーザの再生中に問題がないことも確認します。特定のインスタンスでは，1 ユーザだけしか UPDATE を実行できません。この現象は Siebel で発生します。その場合には，手作業でループを書いて問題を回避する必要があります。

質問 7 : スクリプト再生後にデータベースが変更されているはずなのにされていないことがあります。

回答 : ユーザ・アプリケーションの UI からアプリケーションからアクセスできる現在のデータを調べ，それが更新された値かどうかを確認してください。値が更新されていない場合，それが変更されていないことを判定する必要があります。アプリケーションの記録中に UPDATE ステートメントが 1 つ以上の行を変更したために，再生時には変化がなかったということも考えられます。

次の項目を確認します。

- **ステートメントの検証**：UPDATE ステートメント内の WHERE 句条件式が正しいことを確認します。
- **相関の確認**：アプリケーションを 2 回記録して，それぞれの記録の UPDATE ステートメントを比較し，必要な相関が行われているか確認します。

- **行の総数の確認**：UPDATE の後で変更された行数を確認します。Oracle ではこの情報は **lrd_exec** の 4 番目のパラメータに格納されています。ODBC では，**lrd_row_count** を使用して行数を調べます。スクリプトに更新された行数を出力するコードを追加できます。出力値が 0 ならば，UPDATE はデータベースの変更に失敗したことがわかります。
- **SET 句のチェック**：UPDATE ステートメントの SET 句の条件式を確認します。必要な値がすべて相関されており，ハードコードされていないかどうかチェックします。UPDATE の 2 つの記録を比べることによって判断できます。

これが問題なのは，UPDATE が単独の仮想ユーザの再生時には動作するのに，複数の仮想ユーザでは動作しない場合です。ある仮想ユーザの UPDATE が他の仮想ユーザのものと干渉していることが考えられます。各仮想ユーザに同じ値で更新を行うように指定する場合を除いて，各仮想ユーザをパラメータ化してそれぞれの仮想ユーザが UPDATE 時に異なる値を使用するようにします。この場合，再試行論理を追加して UPDATE を再び試みます。

質問 8：Oracle Application を使用して記録されたステートメントの再生時に一意カラム名エラーを防ぐ方法は？たとえば，次のような場合です。

```
lrd_stmt(Csr9,"SELECT UOM_CODE, UOM_CODE, DESCRIPTION FROM "
        "MTL_UNITS_OF_MEASURE "
        "WHERE NVL(DISABLE_DATE, SYSDATE + 1) > "
        "SYSDATE ORDER BY UOM_CODE", -1, 1, 1, 0);
```

次のエラー・メッセージが発行されます。

```
"lrdo.c/fjParse:"oparse" ERROR return-code=960, oerhms=ORA-
00960:ambiguous column naming in select list".
```

回答：一意でないカラムの少なくとも 1 つに別名を追加して，この新しい一意の名前にマッピングするようにステートメントを変更します。次に例を示します。

```
lrd_stmt(Csr9,"SELECT UOM_CODE,UOM_CODE second, DESCRIPTION
FROM"
        "MTL_UNITS_OF_MEASURE "
        "WHERE NVL(DISABLE_DATE, SYSDATE + 1) > "
        "SYSDATE ORDER BY UOM_CODE", -1, 1, 1, 0);
```

Siebel 固有のスクリプト作成のヒント

本項では，Siebel データベース・ユーザのための解決策を示します。前項までの記述にも，一般的なデータベース・スクリプト作成のヒントがありますので参照してください。

質問 9：VuGen で仮想ユーザは正常に実行できるのにコントローラまたはコンソールでは重複キー割り当て違反になります。

回答：Siebel クライアントは S_SSA_ID テーブルの NEXT_SUFFIX カラムにキーを格納します。このクライアントには接尾辞値のブロックを取得できない場合に，その状況を検出してそこから回復するためのコードがあります。

VuGen は自動的に S_SSA_ID テーブルで，NEXT_SUFFIX および MODIFICATION_NUM フィールドを相関します。UPDATE 中，MODIFICATION_NUM フィールドは 1 つ増加し，NEXT_SUFFIX フィールドは 36 ベースで 100 増加します。ただし，クライアントが新しい接尾辞値のブロックを取得できない場合，VuGen ではインスタンスにコードが追加されません。その結果，新しい値をデータベースに挿入しようとすると，再生時に一意制約エラーが生じます。

1 回目の試行が失敗したときに再試行するために，スクリプトで接尾辞のブロックを取得する各箇所に，手作業でコードを追加します。スクリプト内で「SiebelPreSave」を検索して，該当箇所を見つけます。次の例に似たコードを含む **while** ループも追加する必要があります。この例は，Oracle のみに該当します。ODBC の場合は，**lrd_exec** の 4 番目の引数ではなく，**lrd_row_count** を使用します。

```
unsigned long lRowUpdated;
int nAttempt;

…

//「next_suffix」を取得するまでループが継続します。
lRowUpdated = 0;
nAttempt=0;

while (lRowUpdated != 1) {

   nAttempt++;
   if (nAttempt > 1)
      lr_output_message (".....Next suffix retry %d", nAttempt);
```

```
    else
    {
       lrd_open_cursor(&Csr13, Con1, 0);
       lrd_stmt(Csr13, "SELECT¥n T1.LAST_UPD,¥n T1.CREATED_BY,¥n "
          "T1.CONFLICT_ID,¥n T1.CREATED,¥n T1.NEXT_SUFFIX,¥n "
          "T1.ROW_ID,¥n T1.NEXT_PREFIX,¥n T1.CORPORATE_PREFIX,¥n "
          "T1.MODIFICATION_NUM,¥n T1.NEXT_FILE_SUFFIX,¥n      "
          "T1.LAST_UPD_BY¥n   FROM ¥n SIEBEL.S_SSA_ID T1", -1, 1, 1, 0);
    }
    lrd_bind_cols(Csr13, BCInfo_D375, 0);
    lrd_exec(Csr13, 0, 0, 0, 0, 0);

    SiebelPreSave_1();
    lrd_fetch(Csr13, -1, 4, 0, PrintRow26, 0);
    GRID(26);
    SiebelPostSave_1();

    if (nAttempt > 1)
    {
     lrd_open_cursor(&Csr14, Con1, 0);
     lrd_stmt(Csr14,"¥nUPDATE SIEBEL.S_SSA_ID SET¥n LAST_UPD_BY=:1,¥n "
     "NEXT_SUFFIX = :2,¥n      MODIFICATION_NUM = :3,¥n      LAST_UPD = "
     ":4¥n WHERE¥n ROW_ID = :5 AND MODIFICATION_NUM = :6¥n", -1, 1,
     1, 0);
    }
    lrd_assign_bind(Csr14, "6", " < modification_num > ", &_6_D376, 0,
                LRD_BIND_BY_NUMBER, 0);
    lrd_assign_bind(Csr14,"5", "0-11",&_5_D377,0,LRD_BIND_BY_NUMBER, 0);
    strcpy (szTimeAtNewButton, lr_eval_string("<Now>"));
    sprintf (szTimeStamp, "%s %s", lr_eval_string("<Today>"),
            szTimeAtNewButton);
    lr_save_string (szTimeStamp, "DateTimeStamp");
lrd_assign_bind(Csr14, "4", "<DateTimeStamp>", &_4_D378, 0,
       LRD_BIND_BY_NUMBER, 0);
lrd_assign_bind(Csr14, "3", "<next_modnum>", &_3_D379, 0,
       LRD_BIND_BY_NUMBER, 0);
lrd_assign_bind(Csr14, "2", "<next_suffix_x100>", &_2_D380, 0,
       LRD_BIND_BY_NUMBER, 0);
lrd_assign_bind(Csr14, "1", "1-1E1",&_1_D381,0,LRD_BIND_BY_NUMBER, 0);
```

```
// このアップデートでは接尾辞を正常に取得しない限り行を更新しません。
lrd_exec(Csr14, 1, 0, &lRowUpdated, 0, 0);
lrd_commit(0, Con1, 0);

}//while
    lr_output_message ("...Rows updated %ld", lRowUpdated);
```

質問 10：主キーの相関に使う正しい値を見つける方法は？

回答：Siebel は< **next_suffix** >をベース 36 で数学的に処理した結果に基づいてキーの値を生成する傾向があります。いくつかの記録同士を比較して，関連性を見つけられるかどうか試してみてください。Siebel で相関を行うときには，スクリプト再生に影響しないので日付フィールドを無視できます。

質問 11：重複キー違反による INSERT into S_SRV_REQ の失敗を解決する方法は？

回答：主キーは SR_NUM です。新しいバージョンの VuGen は，S_SSA_ID テーブルの NEXT_SUFFIX 値をベース 36 からベース 10 の同等の値に変換する lrd_siebel_str2num 関数を使って自動的にこのテーブルへの挿入を相関します。旧バージョンの VuGen ではこの相関が正しく行われないことがあります。

質問 12：VuGen でスクリプトを正しく再生するために必要な相関が自動的に行われません。足りない相関を追加する方法は？

回答：現在のところ VuGen は S_SSA_ID テーブルの NEXT_SUFFIX および MODIFICATION_NUM カラムの値を保存し，スクリプトで使用するときにその値をパラメータで置換しているだけです。従って手作業で相関を追加する必要があります。**print.inl** ファイル内の **SiebelPreSave** 関数と **SiebelPostSave** 関数の相関コードは，何を相関する必要があるかわかっている場合に，特定の値を相関する方法を示す例です。

- NEXT_FILE_SUFFIX および MODIFICATION_NUM カラムは S_SSA_ID テーブルから選択される場合があります。その場合，UPDATE ステートメントはベース 36 でこの文字列，および MODIFICATION_NUM に 1 を追加して NEXT_FILE_SUFFIX を更新します。NEXT_FILE_SUFFIX の値はテーブル内の FILE_REV_NUM フィールドに挿入されることがしばしばあります。このテーブルの名前はしばしば接尾辞 **_ATT** で終わり，それが添付ファイルであることを示します。
- Siebel が UPDATE ステートメントを実行するときには必ず，値が 1 ずつ増加する MODIFICATION_NUM カラムが存在します。VuGen は S_SSA_ID テーブルに対してだけ，自動的にこの相関を行います。それ以外は手作業で相関する必要があります。
- Siebel は記録を ID 番号で参照します。Siebel は通常，特定のタイプ（たとえば契約など）のすべての記録を検索し，そのタイプの特定の記録を更新または削除しようとするときに，ID 番号を使います。意味のある負荷テストを生成するには，再生中に ID 番号をパラメータで置換する必要があります。ID 番号は 1-QPF9 のように，1 桁以上の数字とハイフンに続く 1 桁以上の英数字からなります。VuGen ではこのパラメータ化は自動的に行われないので，手作業で行います。
- ほかにも相関またはパラメータ化の不足が見つかった場合は，VuGen の Siebel のサポート向上のために，Mercury のカスタマー・サポートにご連絡ください。

PeopleSoft-Tuxedo スクリプトの実行

Tuxedo 7.x で PeopleSoft-Tuxedo 仮想ユーザを実行するには，**mdrv.dat** ファイル内のライブラリ拡張子を変更する必要があります。

```
[PeopleSoft-Tuxedo]
WINNT_EXT_LIBS=lrt7.dll
```

第 83 章

その他の詳細情報

本章には，VuGen の上級ユーザのための情報が含まれます。

- 記録中に生成されるファイル
- 再生中に生成されるファイル
- UNIX コマンド・ラインからの仮想ユーザの実行
- 仮想ユーザの動作の指定
- コマンド・ライン・パラメータ
- OLE サーバの記録
- .dat ファイルの検証
- 新規仮想ユーザ・タイプの追加

記録中に生成されるファイル

記録されたテストに「vuser」という名前を付け，それを c:¥tmp の下に格納したとします。記録後に生成される，特に重要なファイルの一覧を以下に示します。

vuser.usr	仮想ユーザに関する情報（タイプ，テスト対象アプリケーション，アクション・ファイルなど）が含まれます。
vuser.bak	最後に保存した Vuser.usr の 1 つ前のコピー。
default.cfg	VuGen アプリケーションで定義されたすべての実行環境の設定（思考遅延時間，反復，ログ，Web）の一覧が含まれます。
vuser.asc	記録されている API 呼び出し。
vuser.grd	データベース・スクリプトのグリッドのカラム・ヘッダーが含まれています。
default.usp	スクリプトの実行ロジック（Actions セクションの実行方法など）が含まれます。
init.c	VuGen メイン・ウィンドウに表示される Vuser_init 関数とまったく同じもの。
run.c	VuGen メイン・ウィンドウに表示される Action 関数とまったく同じもの。
end.c	VuGen メイン・ウィンドウに表示される Vuser_end 関数とまったく同じもの。
vdf.h	スクリプトで使用される C 変数定義のヘッダー・ファイル。
¥Data	Data ディレクトリには，主にバックアップ用として使用されるすべての記録データが格納されます。データはこのディレクトリに格納されると，編集したり使用したりできなくなります。たとえば，**Vuser.c** は，**run.c** のコピーです。

Vuser.usr ファイルの例

```
[General]
Type=Oracle_NCA
DefaultCfg=default.cfg
AppName=C:¥PROGRA~1¥Netscape¥COMMUN~1¥Program¥netscape.exe
BuildTarget=
ParamRightBrace=>
ParamLeftBrace=<
NewFunctionHeader=0
MajorVersion=5
MinorVersion=0
ParameterFile=nca_test3.prm
GlobalParameterFile=
[Transactions]
Connect=
[Actions]
vuser_init=init.c
Actions=run.c
vuser_end=end.c
```

default.cfg ファイルの例

```
[General]
XlBridgeTimeout=120

[ThinkTime]
Options=NOTHINK
Factor=1
LimitFlag=0
Limit=1

[Iterations]
NumOfIterations=1
IterationPace=IterationASAP
StartEvery=60
RandomMin=60
RandomMax=90

[Log]
LogOptions=LogBrief
MsgClassData=0
MsgClassParameters=0
MsgClassFull=0
```

再生中に生成されるファイル

本項では，仮想ユーザを再生したときに何が起きるかを説明します。

1 プリプロセッサ用のコマンド・ライン・パラメータを含む **options.txt** ファイルが作成されます。

2 関連するすべての .c および .h ファイルに対する「includes」を含む **Vuser.c** ファイルが作成されます。

3 開発用ファイルからマクロ定義およびプリコンパイラ指示子などを「挿入する」ために C のプリプロセッサ **cpp.exe** が呼び出されます。

次のコマンド・ラインが使用されます。

```
cpp  -foptions.txt
```

4 **pre_cci.c** ファイルが作成されます。これも C ファイルです（**pre_cci.c** は，**options.txt** ファイルで定義されます）。このプロセスのあらゆる出力を含む **logfile.log** ファイル（このファイルも **options.txt** で定義されます）が作成されます。logfile.log ファイルは，プリプロセス処理の段階で何も問題がなければ，空のはずです。このファイルが空でなければ，コンパイルの次の段階で致命的なエラーが発生し，ほぼ確実に失敗します。

5 仮想ユーザ・ドライバ・プログラムによって実行時に解釈される，プラットフォームに依存する疑似バイナリ・ファイル（.ci）を作成するために，C コンパイラ **cci.exe** が起動されます。cci は **pre_cci.c** ファイルを入力として受け取ります。

6 **pre_cci.ci** ファイルが次のように作成されます。

```
cci  -errout  c:¥tmp¥Vuser¥logfile.log  -c  pre_cci.c
```

7 **logfile.log** ファイルは，コンパイル時の出力を格納するログ・ファイルです。

8 **pre_cci.ci** ファイルの名前はここで **Vuser.ci** に変わります。

コンパイル時には警告とエラーの両方が生成される可能性があり，ドライバはこのプロセスの結果を知らないので，ドライバはまず **logfile.log** ファイルにエントリがあるか調べます。logfile.log ファイルにエントリがある場合，ドライバは，**Vuser.ci** ファイルが生成されているかどうかを調べます。ファイル・サイズがゼロでなければ，cci がコンパイルに成功したということです。ファイル・サイズがゼロであれば，コンパイルは失敗しており，エラー・メッセージが表示されます。

9 関連するドライバが実行され，入力データとして **.usr** ファイルと **Vuser.ci** ファイルが使われます。次に例を示します。

```
mdrv.exe  -usr c:¥tmp¥Vuser¥Vuser.usr  -out c:¥tmp¥Vuser  -file
c:¥tmp¥Vuser¥Vuser.ci
```

.usr ファイルは，ドライバ・プログラムにどのデータベースが使用されているのかを知らせるために必要です。この段階で，実行のためにロードすべきライブラリがわかります。

10 実行中に出力されたすべてのメッセージを含む **output.txt** が（「out」変数によって定義されたパス内に）作成されます。これは，VuGen の実行時の出力ウィンドウおよび VuGen のメイン・ウィンドウの下部の表示枠に表示されるものとまったく同じです。

options.txt ファイルの例

```
-DCCI
-D_IDA_XL
-DWINNT
-Ic:¥tmp¥Vuser（Vuser インクルード・ファイルの名前と場所）
-IE:¥LRUN45B2¥include（インクルード・ファイルの名前と場所）
-ec:¥tmp¥Vuser¥logfile.log（出力ログ・ファイルの名前と場所）
 c:¥tmp¥Vuser¥VUSER.c（処理されるファイルの名前と場所）
```

Vuser.c ファイルの例

```
#include "E:¥LRUN45B2¥include¥lrun.h"
#include "c:¥tmp¥web¥init.c"
#include "c:¥tmp¥web¥run.c"
#include "c:¥tmp¥web¥end.c"
```

UNIX コマンド・ラインからの仮想ユーザの実行

VuGen には，仮想ユーザと同じ操作をコマンド・ラインから自動的に実行する UNIX シェル・スクリプト・ユーティリティ **run_db_Vuser.sh** が含まれています。このユーティリティは各再生ステップを個別に実行できます。このツールは，UNIX 上で再生するテストをデバッグするのに便利です。

run_db_Vuser.sh を $M_LROOT/bin ディレクトリに配置します。仮想ユーザ・タイプを再生するには，次のように入力します。

```
run_db_Vuser.sh   Vuser.usr
```

次のコマンド・ライン・オプションを使用することもできます。

-cpp_only このオプションは，プリプロセス処理のフェーズを開始します。この処理によって，「**Vuser.c**」が生成されます。

-cci_only このオプションは，コンパイルのフェーズを実行します。「**Vuser.c**」ファイルは入力データとして使用されます。この処理によって，「**Vuser.ci**」ファイルが生成されます。

-exec_only	このオプションは，「**Vuser.ci**」ファイルを入力データとして受け取り，再生ドライバを介して仮想ユーザを実行します。
-ci ci_file	このオプションを使って，実行する .ci ファイルの名前と場所を指定できます。2 つ目のパラメータに，.ci ファイルの場所を指定します。
-out output_directory	このオプションを使って，各種の処理によって作成される出力ファイルの場所を指定できます。2 つ目のパラメータに，ディレクトリの名前と場所を指定します。
-driver driver_path	このオプションを使って，仮想ユーザの実行に使用する実際のドライバ実行可能ファイルを指定できます。標準設定では，ドライバ実行可能ファイルは VuGen の .dat ファイル内の設定から取得されます。

最初の 3 つのオプションは，run_db_vuser を実行するとき，一度にどれか 1 つだけを使用できます。

仮想ユーザの動作の指定

VuGen は仮想ユーザ・スクリプトと仮想ユーザの動作を 2 つの独立した情報源として作成するので，たとえば待機時間，時間間隔，反復のループ，ログの記録などのユーザの動作を，仮想ユーザ・スクリプトを直接参照せずに設定できます。この機能により，仮想ユーザの設定が変更できると同時に，同じ仮想ユーザ・スクリプトについて「プロファイル」を複数格納できます。

標準設定では，仮想ユーザの動作は VuGen の［実行環境設定］ダイアログ・ボックスで指定されているとおりに，「**Vuser.cfg**」ファイルに定義されています。このファイルの，ユーザ動作ごとに異なる複数のバージョンを保存することができます。そして，関連する **.cfg** ファイルを参照する仮想ユーザ・スクリプトを実行できます。

サーバ・マシンから，使用する設定ファイルを指定して仮想ユーザ・スクリプトを実行できます。これを行うには，次のパラメータ指定を仮想ユーザのコマンド・ラインに追加します。

```
-cfg c:¥tmp¥profile2.cfg
```

コマンド・ライン・パラメータについては，1300 ページ「コマンド・ライン・パラメータ」を参照してください。

VuGen からは，振る舞いを定義したファイルを指定できません。VuGen は仮想ユーザと同じ名前の **.cfg** ファイルを自動的に使用します（もちろんファイル名を「**Vuser.cfg**」に変更することもできます)。ただし，上で説明した -cfg パラメータをドライバのコマンド・ラインの最後に追加すれば，コマンド・ラインから手作業でファイルを指定できます。

注 : UNIX 用のユーティリティ **run_db_vuser** では，このオプションはまだサポートされていません。

コマンド・ライン・パラメータ

仮想ユーザは，起動時にコマンド・ライン・パラメータを受け付けます。仮想ユーザ API には，コマンド・ライン・パラメータを参照するための関数がいくつかあります（**lr_get_attrib_double** など)。環境内で，スクリプト・ウィンドウのコマンド・ライン・エントリにパラメータを追加することで，仮想ユーザにコマンド・ライン・パラメータを送ることができます。

VuGen から仮想ユーザを実行するときは，コマンド・ライン・パラメータを指定できません。ただし，Windows のコマンド・ラインで他のすべてのドライバ・パラメータの後，つまり行の最後にパラメータを追加することでこれを手作業で指定できます。

```
mdrv.exe -usr c:¥tmp¥Vuser¥Vuser.usr    -out c:¥tmp¥vuser
vuser_command_line_params
```

注 : UNIX 用のユーティリティ **run_db_vuser** では，このオプションはまだサポートされていません。

OLE サーバの記録

VuGen では現在，OLE アプリケーションの記録はサポートされていません。OLE アプリケーションでは，実際のプロセスが標準のプロセス生成ルーチンによってではなく，OLE オートメーション・システムによって起動されます。ただし，以下に示すガイドラインに沿って，OLE アプリケーション用の仮想ユーザ・スクリプトが作成できます。

OLE サーバには，実行可能ファイルと DLL の 2 つの種類があります。

DLL サーバ

サーバが DLL である場合，このサーバは最終的にアプリケーションのプロセス空間にロードされ，VuGen は LoadLibrary への呼び出しを記録します。この場合，ユーザはこれが OLE アプリケーションであることに気付かないかもしれません。

実行可能サーバ

サーバが実行形式の場合は，以下に示す方法で VuGen から実行ファイルを起動する必要があります。

- まず，実際に記録する必要があるプロセスを特定します。多くの場合，アプリケーションの実行可能ファイルの名前がわかっています。名前がわからない場合は，対象アプリケーションを起動し，NT のタスク・マネージャでその名前を確認します。
- 必要なプロセスを特定したら，VuGen で［**記録開始**］をクリックします。アプリケーション名の入力を要求されるので，OLE アプリケーションの名前と，その後ろに「/Automation」というフラグを入力します。次に，VuGen からではなく通常の方法でユーザ・プロセスを実行します。VuGen は実行中の OLE サーバを記録し，同じサーバを別に起動することはありません。VuGen で OLE サーバのアクションを記録するには，ほとんどの場合，この手順でうまくいきます。
- それでもうまく記録できない場合は，**CmdLine** プログラムを使って，直接起動されないプロセスの完全なコマンド・ラインを調べます（このプログラムは，Mercury のカスタマー・サポート Web サイト http://support.mercuryinteractive.com の知識ベースの記事からダウンロードできます）。

CmdLine の使用法

次の例では，**CmdLine.exe** を使って，他のプロセスによって起動されるプロセス MyOleSrv.exe の完全なコマンド・ラインを調べています。

完全なコマンド・ラインを調べるには，次の手順を実行します。

1 **MyOleSrv.exe** の名前を **MyOleSrv.orig.exe** に変更します。

2 アプリケーションと同じディレクトリに **CmdLine.exe** を入れ，この名前を **MyOleSrv.exe** に変更します。

3 **MyOleSrv.exe** を起動します。この MyOleSrv.exe は，元のアプリケーションの完全なコマンド・ラインを含むポップアップ・メッセージ（追加情報を含む）を表示し，その情報を **c:¥temp¥CmdLine.txt** に書き込みます。

4 それぞれを元の名前に戻し，正しいコマンド・ライン・パラメータを使って OLE サーバ **MyOleSrv.exe** を VuGen から起動します。ユーザ・アプリケーションは VuGen からではなく，通常の方法で起動します。ほとんどの場合，VuGen は正しく記録を行います。

それでもうまく記録できない場合は，次の処理を行います。

1 OLE サーバ名を MyOleSrv.1.exe に，CmdLine を MyOleSrv.exe に変更します。

2 環境変数「CmdStartNotepad」と「CmdNoPopup」を「1」に設定します。CmdLine 環境変数については，1303 ページ「CmdLine 環境変数」の一覧を参照してください。

3 VuGen 以外からアプリケーションを起動します。「メモ帳」が開き，完全なコマンド・ラインが表示されます。コマンド・ライン引数を調べます。アプリケーションを数回起動し，コマンド・ライン引数を比較します。何度アプリケーションを起動しても引数が同じである場合は，CmdStartNotepad 環境変数をリセットします。そうでなければ，設定を「1」のままにしておきます。

4 VuGen で，コマンド・ライン・パラメータを使用して（「メモ帳」のウィンドウからコピー / 貼り付けで指定してください）プログラム MyOleSrv.1.exe を起動します。

5 VuGen 以外からアプリケーションを起動します。

CmdLine 環境変数

以下に示す環境変数を使うことで，CmdLine の実行を制御できます。

CmdNoPopup これが設定されていると，ポップアップ・ウィンドウが現れません。

CmdOutFileName これが設定されており，空でない場合は，CmdLine は c:¥temp¥CmdLine.txt のかわりにこのファイルを作成しようとします。

CmdStartNotepad これが設定されていると，出力ファイルがメモ帳に表示されます（CmdNoPopup との併用をお勧めします）。

.dat ファイルの検証

VuGen は，vugen.dat と mdrv.dat という 2 つの .dat ファイルを使用します。

vugen.dat

この vugen.dat ファイルは M_LROOT¥dat ディレクトリにあり，VuGen についての一般情報が含まれています。VuGen とコントローラまたはコンソールの両方で使用されます。

```
[Templates]
RelativeDirectory=template
```

Templates セクションは，VuGen プロトコル用のテンプレートの場所を示します。標準のエントリは，これらのテンプレートが相対 **template** ディレクトリにあることを示します。各プロトコルには，**template** の下にサブディレクトリがあり，この中には，そのプロトコル用のテンプレート・ファイルが含まれています。

次のセクションは **GlobalFiles** セクションです。

```
[GlobalFiles]
main.c=main.c
@@TestName@@.usr=test.usr
default.cfg=test.cfg
default.usp=test.usp
```

GlobalFiles セクションには，新規テストが作成されたときに VuGen がテスト・ディレクトリにコピーしたファイルの一覧が含まれます。たとえば，「user1」というテストがある場合，VuGen は **main.c**，**user1.usr**，および **user1.cfg** をテスト・ディレクトリにコピーします。

ActionFiles セクションには，仮想ユーザによって実行されるアクションを含むファイルの名前と，反復を実行する仮想ユーザが含まれます。

```
[ActionFiles]
@@actionFile@@=action.c
```

上に示した設定に加え，**vugen.dat** には，オペレーティング・システムおよびコンパイルに関連するその他の設定が含まれます。

mdrv.dat

mdrv.dat ファイルには，ライブラリ・ファイルとドライバの実行可能ファイルの場所を定義する，プロトコル別のセクションがあります。次の項では，新しいプロトコルを定義するためにファイルに追加すべき項目を説明します。

新規仮想ユーザ・タイプの追加

VuGen に新しい仮想ユーザのタイプまたはプロトコルを追加するのに必要な項目は次のとおりです。

- 新しいプロトコルの設定による **mdrv.dat** ファイルの編集
- **.cfg** ファイルの追加
- **.lrp** ファイルを挿入
- テンプレート・ディレクトリの作成

mdrv.dat ファイルの編集

まず，**mdrv.dat ファイル**を編集します。このファイルは，**M_LROOT¥dat** ディレクトリにあります。新しい仮想ユーザタイプのためのセクションを追加します。

[< **extension_name** >]
ExtPriorityType= < {internal, protocol} >
WINNT_EXT_LIBS= < NT 用 DLL 名>
WIN95_EXT_LIBS= < 95 用 DLL 名>
SOLARIS_EXT_LIBS= < Solaris 用 dll 名>
LINUX_EXT_LIBS= < Linux 用 dll 名>
HPUX_EXT_LIBS= < HP 用 dll 名>
AIX_EXT_LIBS= < IBM 用 dll 名>
LibCfgFunc= <設定関数名>
UtilityExt= <他の拡張子リスト>
WINNT_DLLS= <インタプリタ・コンテキストにロードする DLL（NT 用）>
WIN95_DLLS= <インタプリタ・コンテキストにロードする DLL（95 用）>
SOLARIS_DLLS= <インタプリタ・コンテキストにロードする dll（Solaris 用）>
LINUX_DLLS= <インタプリタ・コンテキストにロードする dll（Linux 用）>
HPUX_DLLS= <インタプリタ・コンテキストにロードする dll（HP 用）>
AIX_DLLS= <インタプリタ・コンテキストにロードする dll（IBM 用）>
ExtIncludeFiles= <追加インクルード・ファイル。複数のファイルをカンマで区切って指定できる>
ExtCmdLineConc= <追加コマンド・ライン（属性がある場合は値を連結する）>
ExtCmdLineOverwrite= <追加コマンド・ライン（属性がある場合は値を上書きする）>
CallActionByNameFunc= <インタプリタ exec_action 関数>
GetFuncAddress= <インタプリタ get_location 関数>
RunLogicInitFunc= < action_logic init 関数>
RunLogicRunFunc= < action_logic run 関数>
RunLogicEndFunc= < action_logic end 関数>

たとえば，Oracle NCA 仮想ユーザ・タイプは，以下で表されます。

```
[Oracle_NCA]
ExtPriorityType=protocol
WINNT_EXT_LIBS=ncarp11i.dll
WIN95_EXT_LIBS=ncarp11i.dll
LINUX_EXT_LIBS=liboranca11i.so
SOLARIS_EXT_LIBS=liboranca11i.so
HPUX_EXT_LIBS=liboranca11i.sl
AIX_EXT_LIBS=liboranca11i.so
LibCfgFunc=oracle_gui_configure
UtilityExt=lrun_api,HttpEngine
ExtCmdLineOverwrite=-WinInet No
ExtCmdLineConc=-UsingWinInet No
SecurityRequirementsFiles=oracle_nca.asl
SecurityMode=On
```

VuGen は，コードに変更を加えずに新しい仮想ユーザ・タイプを処理できるように設計されています。ただし，特別なビューを追加しなければならない場合もあります。

VuGen では汎用のドライバは提供されていませんが，既存のドライバをカスタマイズできます。カスタマイズしたドライバを使用するには，**mdrv.dat** を変更します。プラットフォームと既存のドライバの行を追加した後，カスタマイズしたドライバの名前の行を「**＜プラットフォーム＞ _DLLS= ＜再生用 DLL 名＞**」の形式で追加します。たとえば，SAP の再生用 DLL が SAPPLAY32.DLL という名前の場合は，次の 2 行を **mdrv.dat** の [sap] セクションに追加します。

```
WINNT=sapdrv32.exe
WINNT_DLLS=sapplay32.dll
```

CFG ファイルの追加

プロトコルに標準の［実行環境の設定］を設定するために，設定ファイルを任意で指定できます。これを行うには，**mdrv.dat** ファイル内の **LibCfgFunc** 変数で定義するか，テンプレートの下の新しいプロトコル・サブディレクトリに **default.cfg** というファイルをおきます。サンプルの .cfg は次のとおりです。

```
[ThinkTime]
Options=NOTHINK
Factor=1
LimitFlag=0
Limit=1

[Iterations]
NumOfIterations=1
IterationPace=IterationASAP
StartEvery=60
RandomMin=60
RandomMax=90

[Log]
LogOptions=LogExtended
MsgClassData=0
MsgClassParameters=0
MsgClassFull=1
```

LRP ファイルの挿入

dat¥protocols ディレクトリに，プロトコルを定義する **lrp** ファイルを挿入します。このファイルには，Protocol，Template，VuGen，API というセクションにプロトコルの設定情報が含まれています。プロトコルの中には，追加の実行環境設定オプションに応じて，これ以外のセクションがあるものもあります。

Protocol セクションには，プロトコルの名前，カテゴリ，説明，ビットマップの場所などが記述されています。

```
[Protocol]
Name=WAP
CommonName=WAP
Category=Wireless
Description=Wireless Application Protocol - used for Web-based, wireless
communication between mobile devices and content providers.
Icon=bitmaps¥wap.bmp
Hidden=0
Single=1
Multi=0
```

Template セクションには，スクリプトのさまざまなセクションの名前と標準のテスト名が記載されています。

```
[Template]
vuser_init.c=init.c
vuser_end.c=end.c
Action1.c=action.c
Default.usp=test.usp
@@TestName@@.usr=wap.usr
default.cfg=default.cfg
```

VuGen セクションには，記録と再生エンジン，必要な DLL と実行時ファイルに関する情報が記述されています。

API セクションには，プロトコルのスクリプト API 関数についての情報が記述されています。

プロトコル・ディレクトリ内の任意の lrp ファイルを新しいプロトコルのひな形として使用できます。

テンプレートの指定

lrp ファイルを追加したら，**M_LROOT**¥template の下にサブディレクトリを作成し，**lrp** ファイルで定義したプロトコル名に対応する名前を付けます。このサブディレクトリに，一般設定および実行環境設定のための標準の設定を収めた **default.cfg** ファイルをおきます。

新しいプロトコルのすべてのスクリプトでグローバルなヘッダー・ファイルを使用する場合には，**globals.h** という名前のファイルを追加します。このファイルには，新しいプロトコルのためのヘッダー・ファイルを指す include ステートメントを含めておきます。たとえば，template¥http サブディレクトリには，**globals.h** というファイルがあり，include ディレクトリにある **as_web.h** ファイルをインクルードしています。

```
#include #as_web.h"
```

第 18 部

付録

付録 A

外部関数の呼び出し

VuGen 使用時に，外部 DLL で定義されている関数を呼び出せます。スクリプトから外部関数を呼び出すことにより，スクリプトと実行環境全体で必要とするメモリを減らせます。

外部関数を呼び出すには，その関数が定義されている DLL をロードします。

DLL は次のようにしてロードできます。

- ローカル：1 つのスクリプトにロードする場合には，**lr_load_dll** 関数を使用します。
- グローバル：すべてのスクリプトにロードする場合には，vugen.dat ファイルにステートメントを追加します。

DLL のロード：ローカル

lr_load_dll 関数を使用して，DLL を仮想ユーザ・スクリプトにロードします。一度 DLL をロードすれば，その DLL で定義されている任意の関数をスクリプト内で宣言せずに呼び出せます。

DLL 内で定義されている関数を呼び出すには，次の手順を実行します。

1 **lr_load_dll** 関数を使用して，スクリプトの先頭で DLL を読み込みます。このステートメントを **vuser_init** セクションの先頭に置きます。**ci_load_dll** 関数が **lr_load_dll** 関数に置き換えられます。

次の構文を使用します。

lr_load_dll(**library_name**);

UNIX プラットフォームでは，DLL は共有ライブラリと呼ばれています。ライブラリの拡張子はプラットフォームによって異なります。

2 DLL 内で定義されている関数をスクリプト内の適切な場所で呼び出します。

次の例では，Test_1 テーブル作成後，**orac1.dll** で定義されている **insert_vals** 関数を呼び出します。

```
int LR_FUNC Actions(LR_PARAM p)
{
lr_load_dll("orac1.dll");
lrd_stmt(Csr1, "create table Test_1 (name char(15), id integer)¥n", -1,
        1 /*Deferred*/, 1 /*Dflt Ora Ver*/, 0);
lrd_exec(Csr1, 0, 0, 0, 0, 0);
/* insert_vals 関数を呼び出し，値をテーブルに挿入する。
*/insert_vals();
lrd_stmt(Csr1, "select * from Test_1¥n", -1, 1 /*Deferred*/, 1 /*Dflt Ora
Ver*/, 0);
lrd_bind_col(Csr1, 1, &NAME_D11, 0, 0);
lrd_bind_col(Csr1, 2, &ID_D12, 0, 0);
lrd_exec(Csr1, 0, 0, 0, 0, 0);
lrd_fetch(Csr1, -4, 15, 0, PrintRow14, 0);
…
```

注：DLL の完全パス名を指定できます。パスを指定しなかった場合は，lr_load_library が，C++ 関数（Windows プラットフォーム上の LoadLibrary）で使われる標準シーケンスを使って DLL を検索します。UNIX プラットフォーム（または同等のプラットフォーム）では，LD_LIBRARY_PATH 環境変数を設定できます。**lr_load_dll** 関数は，**dlopen** と同じ検索規則を使用します。詳細については，dlopen の main ページなどを参照してください。

DLL のロード：グローバル

DLL をグローバルにロードして，その関数をすべての仮想ユーザ・スクリプトで利用できます。一度 DLL をロードすれば，その DLL で定義されている任意の関数をスクリプト内で宣言せずに呼び出せます。

DLL 内で定義されている関数を呼び出すは，次の手順を実行します。

1 （アプリケーションの **dat** ディレクトリにある）**mdrv.dat** ファイルの適切なセクションに，ロードする DLL のリストを追加します。

次の構文を使用します。

PLATFORM_DLLS=**my_dll1**.dll, **my_dll2**.dll, ...

文字列 **PLATFORM** を使用するプラットフォームに置き換えます。プラットフォームのリストについては，**mdrv.dat** ファイルの最初のセクションを参照してください。

たとえば，NT プラットフォーム上の WinSock 仮想ユーザの DLL をロードするには，**mdrv.dat** ファイルに次の文を追加します。

```
[WinSock]
ExtPriorityType=protocol
WINNT_EXT_LIBS=wsrun32.dll
WIN95_EXT_LIBS=wsrun32.dll
LINUX_EXT_LIBS=liblrs.so
SOLARIS_EXT_LIBS=liblrs.so
HPUX_EXT_LIBS=liblrs.sl
AIX_EXT_LIBS=liblrs.so
LibCfgFunc=winsock_exten_conf
UtilityExt=lrun_api
ExtMessageQueue=0
ExtCmdLineOverwrite=-WinInet No
ExtCmdLineConc=-UsingWinInet No
WINNT_DLLS=user_dll1.dll, user_dll2.dll, ...
```

2 DLL 内で定義されている関数をスクリプト内の適切な場所で呼び出します。

付録 B

多国語を使った作業

VuGen では多国語環境がサポートされており，スクリプトを作成，実行する際に自国語のマシン上で英語その他の言語を使用できます。

本付録では，次の項目について説明します。

- 多国語を使った作業について
- 手作業による文字列エンコーディングの変換
- パラメータ・ファイル内の文字列エンコーディングの変換
- Web 記録および再生のための文字列エンコーディングの設定
- Accept-Language ヘッダの言語の指定
- プロトコルに関する制限
- Quality Center の統合

多国語を使った作業について

英語以外の言語で作業を行うときは，記録や再生時に VuGen がテキストのエンコーディングを認識していることを確認することが，主な問題になります。エンコーディングは，スクリプトで使用するすべてのテキストに適用されます。これには，Web 仮想ユーザの場合の HTTP ヘッダ内のテキストや HTML ページ，パラメータ・ファイル内のデータなどが含まれます。

Windows 2000 以降では，ANSI，Unicode，Unicode ビッグ・エンディアン，UTF-8 など，特定のエンコーディングがされたテキスト・ファイルを，メモ帳から直接保存できます。

標準設定では，VuGen はローカル・マシンのエンコーディング（ANSI）で動作します。多国語で作業を行っているサーバによっては，UTF-8 エンコーディングでの作業が必要になることがあります。このようなサーバに対して作業を行うためには，詳細記録オプションの中で，スクリプトが UTF-8 エンコーディングであるよう指定する必要があります。

手作業による文字列エンコーディングの変換

lr_convert_string_encoding 関数を使用すると，文字列のエンコーディング（UTF-8，Unicode，またはロケール・マシン・エンコーディング）を手作業で変換できます。関数の構文は次のとおりです。

```
lr_convert_string_encoding(char * sourceString, char * fromEncoding, char * toEncoding, char * paramName)
```

この関数は，結果の文字列（終端の NULL を含む）を第 3 引数 **paramName** に保存します。変換に成功した場合は 0 を返し，失敗した場合は -1 を返します。

fromEncoding 引数および **toEncoding** 引数の形式は次のとおりです。

LR_ENC_SYSTEM_LOCALE	NULL
LR_ENC_UTF8	"utf-8"
LR_ENC_UNICODE	"ucs-2"

次の例では，**lr_convert_string_encoding** によって "Hello world" をシステム・ロケールから Unicode に変換しています。

```
Action()
{
   int rc = 0;
   unsigned long  converted_buffer_size_unicode = 0;
   char              *converted_buffer_unicode = NULL;

   rc = lr_convert_string_encoding("Hello world", NULL,
LR_ENC_UNICODE, "stringInUnicode");
   if(rc < 0)
   {
      // エラー
   }
return 0;
}
```

実行ログでは，出力ウィンドウに次の情報が表示されます。

```
Output:
Starting action Action.
Action.c(7): Notify: Saving Parameter "stringInUnicode =
H¥x00e¥x00l¥x00l¥x00o¥x00 ¥x00w¥x00o¥x00r¥x00l¥x00d¥x00¥x00¥x00"
Ending action Action.
```

変換の結果は **paramName** 引数に保存されます。

パラメータ・ファイル内の文字列エンコーディングの変換

パラメータ・ファイルには，スクリプトの中で定義されたパラメータに対応するデータが収められています。このファイルはスクリプトのディレクトリに格納され，拡張子 ***.dat** が付けられます。スクリプトを実行するとき，仮想ユーザはこのデータを使用して，アクションをさまざまな値で実行します。

標準設定では，パラメータ・ファイルはマシンのエンコーディングを使用して保存されます。しかし，英語以外の言語で作業を行うときは，サーバにおいて文字列を UTF-8 で受信することが前提となっている場合に，パラメータ・ファイルを UTF-8 に変換しなければならないことがあります。Windows 2000 以降で作業を行っていれば，メモ帳から直接この変換を実行できます。

UTF-8 エンコーディングをパラメータ・ファイルに適用するには，次の手順を実行します。

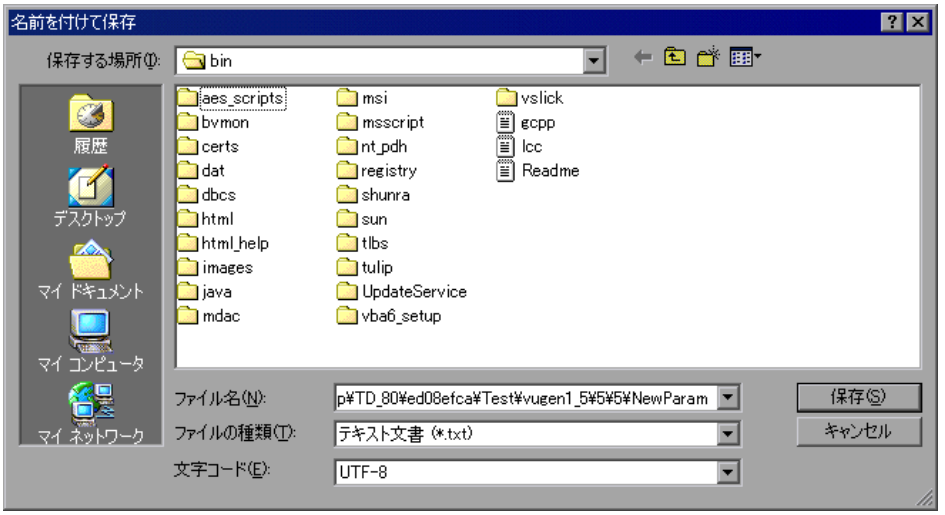
1 ［**仮想ユーザ**］＞［**パラメータ リスト**］を選択し，パラメータ・プロパティを表示します。

2 右側の表示枠にある［**ファイル パス**］ボックスで，パラメータ・ファイルを探します。

3 パラメータ・テーブルを表示した状態で，［**メモ帳で編集**］をクリックします。メモ帳が開き，パラメータ・ファイルが csv 形式で表示されます。

4 ［**ファイルの種類**］ボックスで，［**すべてのファイル**］を選択します。

［**文字コード**］ボックスで，エンコーディングのタイプとして［**UTF-8**］を選択します。



5 ［**保存**］をクリックします。既存のパラメータ・ファイルを上書きしてもよいかどうかの確認を求められます。［**はい**］をクリックします。

これで，パラメータ・ファイルが VuGen によって UTF-8 テキストとして認識されるようになります。ただし，表示上は通常の文字で表示されます。

Web 記録および再生のための文字列エンコーディングの設定

Web またはその他のインターネット・プロトコルを使って作業を行うときは，Web ページ・テキストの記録用のエンコーディングを指定できます。記録するサイトの言語がオペレーティング・システムの言語と一致している必要があります。1 つの記録の中で複数のエンコーディングを組み合わせることはできません。たとえば，UTF-8 を ISO-8859-1 や shift_jis と一緒に使用することはできません。

本項では，次について説明します。

- ［エンコーディング］記録オプション
- 手作業によるエンコーディングの有効化
- ブラウザの設定

［エンコーディング］記録オプション

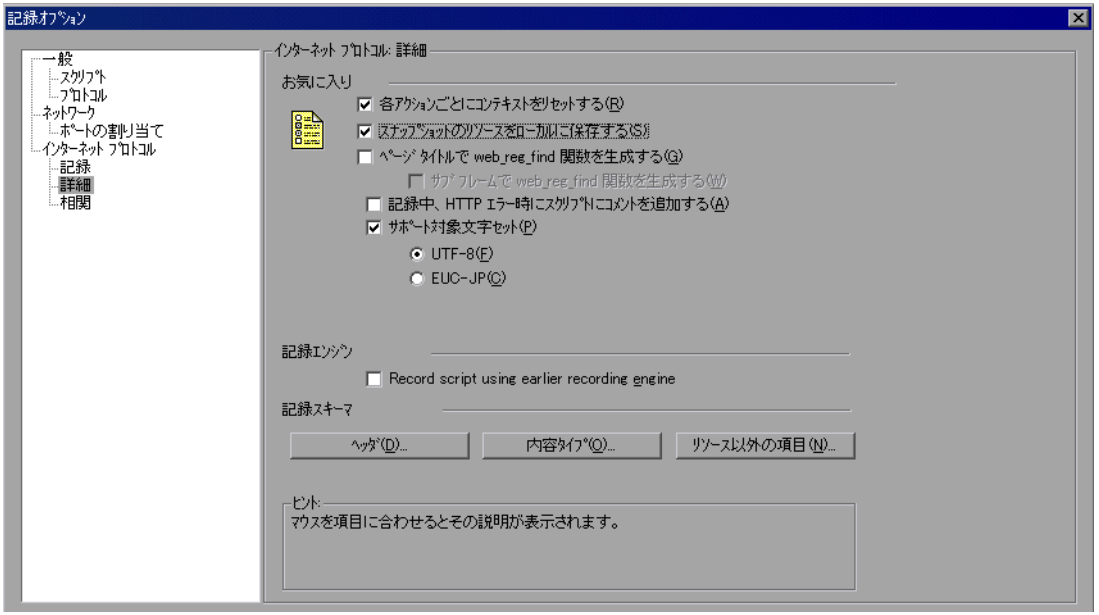
英語以外の言語の Web ページとして認識させるためには，ページの HTTP ヘッダまたは HTML メタ・タグの中で文字セットを指定する必要があります。そうしないと，EUC-JP エンコーディングが VuGen によって認識されず，Web サイトが正しく記録されません。英語以外の言語の要求を **EUC-JP** または **UTF-8** として記録するよう VuGen に指示するには，該当するオプションを［記録オプション］ダイアログ・ボックスの［**HTTP プロパティ：詳細**］ノードで選択します。

- ［**UTF-8**］：UTF-8 エンコーディングのサポートを有効にします。この設定を有効にすると，非 ASCII の UTF-8 文字がマシンのロケールで使用されているロケールに変換され，VuGen に正しく表示されます。
- ［**EUC-JP**］：日本語版 Windows のユーザは，このオプションを選択することで，EUC-JP 文字エンコーディングを使用している Web サイトに対するサポートを有効にできます。この設定を有効にすると，EUC-JP 文字がマシンのロケールで使用されているロケールに変換され，VuGen に正しく表示されます。VuGen によって，すべての EUC-JP（日本語 UNIX）文字列をマシンのロケールに合わせて Shift-JIS（日本語版 Windows）エンコーディングに変換し，スクリプトに **web_sjis_to_euc_param** 関数を追加します（漢字のみ）。

［記録オプション］で［**EUC-JP**］または［**UTF-8**］を選択すると，Web ページで異なるエンコーディングが使用されている場合でも，選択したエンコーディングによって強制的に Web ページが記録されます。たとえば，非 EUC でエンコードされた Web ページを EUC-JP として記録した場合，スクリプトでは正しく再生されません。

文字セット・エンコーディングを有効にするには，次の手順を実行します。

1 ［記録オプション］を開き（Ctrl+F7），［**詳細**］ノードを選択します。



2 ［**サポート対象文字セット**］を選択します。必要な文字エンコーディングとして，［UTF-8］または［EUC-JP］（漢字対応のオペレーティング・システムでのみ有効）を選択します。

3 ［**OK**］をクリックします。

これらの設定の詳細については，819 ページ「記録オプションの詳細設定」を参照してください。

手作業によるエンコーディングの有効化

web_sjis_to_euc_param 関数を使用すると，EUC-JP でエンコードされている HTML ページについて，その記録と再生を行うためのすべてのサポートを手作業で追加できます。これにより，EUC エンコードされた日本語文字を仮想ユーザ・スクリプトで正しく表示できるようになります。

web_sjis_to_euc_param を使用すると，パラメータの値が実行ログで EUC-JP エンコーディングを使用して表示されます。たとえば，**web_find** 関数を再生するとき，VuGen ではエンコードされた値が表示されます。これらの値には，**web_sjis_to_euc_param** 関数によって EUC に変換された文字列値や，［**実行環境設定**］＞［**ログ**］＞［**拡張ログ**］で有効になっているときのパラメータ置換が含まれます。

ブラウザの設定

記録中，スクリプト内の非英語の文字がエスケープされた 16 進数として表示されることがあります（たとえば文字列 "‹&" が "%DC%26" になるなど）。そのような場合は，URL を UTF-8 エンコーディングで送信しないようブラウザを設定することによって修正できます。Internet Explorer では，［**ツール**］＞［**インターネット オプション**］を選択し，［**詳細設定**］タブをクリックします。そして，［ブラウズ］セクションにある［**常に UTF-8 として URL を送信する**］オプションをクリアします。

web_sjis_to_euc_param の詳細については，「**オンライン関数リファレンス**」を参照してください。

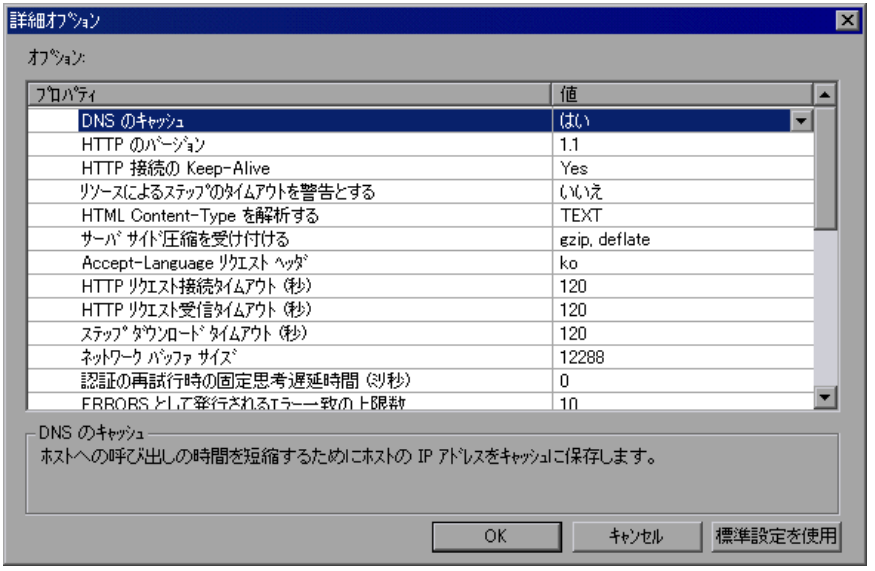
Accept-Language ヘッダの言語の指定

Web スクリプトを実行する前に，現在使用している言語に合わせてページのリクエスト・ヘッダを設定しておくことができます。［実行環境設定］の［インターネット プロトコル］で，**Accept-Language** リクエスト・ヘッダを設定します。このヘッダを通じて，許容されるすべての言語のリストがサーバに提供されます。

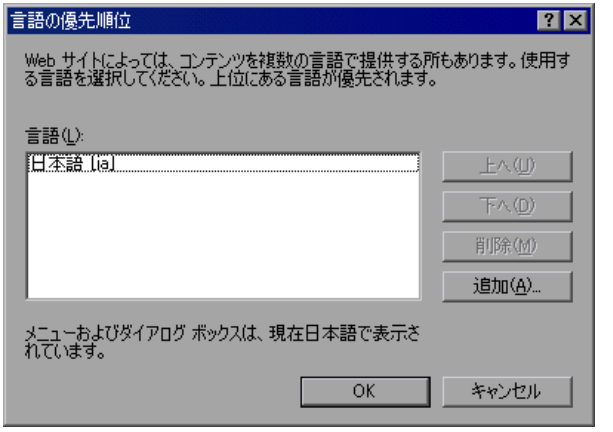
Accept-Language ヘッダを設定するには，次の手順を実行します。

1 ［実行環境設定］ダイアログ・ボックス（F4）を開き，［**インターネット プロトコル**：**プリファレンス**］ノードを選択します。

2 ［詳細］セクションで，［**オプション**］をクリックして［詳細オプション］ダイアログ・ボックスを開きます。



3［**Accept-Language リクエスト ヘッダ**］オプションを探します。［**値**］カラムで，一覧から必要な言語を選択します。この一覧は，使用しているブラウザの［インターネット オプション］の［言語］設定に基づくものです。



これらの設定の詳細については，859 ページ「インターネット・プリファレンスのその他のオプション」を参照してください。

プロトコルに関する制限

SMTP プロトコル

SMTP プロトコルを MS Outlook や MS Outlook Express を通じて使用している場合は，仮想ユーザ・スクリプトに記録される日本語テキストが正しく表示されません。ただし，スクリプトの記録と再生は正しく実行されます。

スクリプト名の長さ

COM，FTP，IMAP，SMTP，POP3，REAL，または VBA の VB モードで記録するときは，スクリプト名の長さがマルチバイト文字で 10 文字（21 バイト）に制限されます。

Quality Center の統合

Quality Center プロジェクトに保存されているスクリプトを VuGen から開く場合や，Quality Center プロジェクトに保存されているシナリオをコントローラから開く場合は，「Default」（英語）という名前の新しいテスト・セットを Quality Center プロジェクトに追加します。

付録 C

UNIX プラットフォームでのスクリプトのプログラミング

UNIX プラットフォームを利用する仮想ユーザは，プログラミングによって UNIX プラットフォーム向けのスクリプトを作成できます。プログラミングによってスクリプトを作成するには，テンプレートを使用します。

本付録では，次の項目について説明します。

- UNIX プラットフォームで実行可能な仮想ユーザ・スクリプトのプログラミングについて
- テンプレートの生成
- 仮想ユーザのアクションのプログラミング
- 仮想ユーザの実行環境設定
- トランザクションとランデブー・ポイントの定義
- スクリプトのコンパイル

UNIX プラットフォームで実行可能な仮想ユーザ・スクリプトのプログラミングについて

UNIX プラットフォームで実行可能な仮想ユーザ・スクリプトを作成する方法は 2 つあります。1 つは VuGen を使用する方法で，もう 1 つはプログラミングを行う方法です。

VuGen

VuGen を使って，UNIX プラットフォームで実行可能な仮想ユーザ・スクリプトを作成できます。Windows 環境でアプリケーションを記録して，それを UNIX で実行できます（記録は UNIX ではサポートされていません）。

プログラミング

UNIX だけの環境を使用している場合は，仮想ユーザ・スクリプトをプログラミングによって作成できます。スクリプトは C 言語または C++ 言語でプログラミングして，ダイナミック・ライブラリとしてコンパイルする必要があります。

本付録では，プログラミングによって仮想ユーザ・スクリプトを作成する方法について説明します。

プログラミングによってスクリプトを作成するには，仮想ユーザのテンプレートを土台に，より大きな仮想ユーザ・スクリプトを作成します。テンプレートで提供するものは，次のとおりです。

- 正しいプログラム構造
- 仮想ユーザ API 呼び出し
- ダイナミック・リンク・ライブラリを作成するためのソース・コードとメイクファイル

テンプレートに基づいて基本となるスクリプトを作成したら，実行時の仮想ユーザ情報と統計値を提供できるようにスクリプトを拡張します。詳細については，第 8 章「仮想ユーザ・スクリプトの拡張」を参照してください。

テンプレートの生成

VuGen には，テンプレートを作業ディレクトリにコピーするユーティリティが含まれています。このユーティリティは **mkdbtest** と呼ばれているもので，$M_LROOT¥bin にあります。このユーティリティを実行するには，次のように入力します。

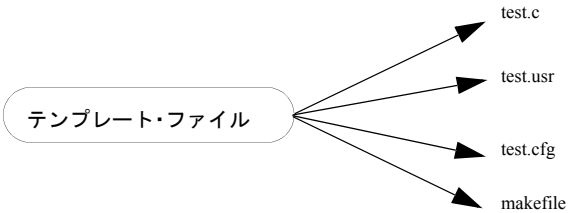
```
mkdbtest name
```

mkdbtest を実行すると，mkdbtest はテンプレート・ファイル **name.c** を格納するためのディレクトリ **name** を作成します。たとえば，次のように入力するとします。

```
mkdbtest test1
```

mkdbtest は，テンプレート・スクリプト **test1.c** を格納するためのディレクトリ **test1** を作成します。

mkdbtest ユーティリティを実行すると，4 つのファイル（**test.c**，**test.usr**，**test.cfg**，**Makefile**）を格納するためのディレクトリが作成されます。これらのファイルの「**test**」の部分には，mkdbtest で指定したテスト名が入ります。



仮想ユーザのアクションのプログラミング

仮想ユーザ・スクリプトの **test**.c，**test**.usr，**test**.cfg ファイルは，使用する仮想ユーザに応じてカスタマイズできます。

実際の仮想ユーザのアクションをプログラミングして，**test.c** ファイルに挿入します。このファイルには，プログラミングされる仮想ユーザ・スクリプトに必要な構造になっています。仮想ユーザ・スクリプトには，vuser_init，Actions，vuser_end の 3 つのセクションがあります。

C++ のユーザに対しては，テンプレートは **extern C** を定義します。エクスポートされる関数が不用意に変更されることがないように，すべての C++ ユーザに対して この定義を行う必要があります。

```
#include "lrun.h"
#if defined(__cplusplus) || defined(cplusplus) extern "C"
{
#endif
int LR_FUNC vuser_init(LR_PARAM p)
{
        lr_message("vuser_init done¥n");
        return 0;
}
 int Actions(LR_PARAM p)
{
        lr_message("Actions done¥n");
        return 0;
}
 int vuser_end(LR_PARAM p)
{
        lr_message("vuser_end done¥n");
        return 0;
}
#if defined(__cplusplus) || defined(cplusplus) }
#endif
```

仮想ユーザのアクションをプログラミングして空のスクリプトに直接挿入します。場所は，各セクションの **lr_message** 関数の前です。

vuser_init セクションは，初期化中に最初に実行されます。このセクションには，接続情報とログオンの処理を挿入します。**vuser_init** セクションは，スクリプトの実行時に一度だけ実行されます。

Actions セクションは初期化の後に実行されます。このセクションには，仮想ユーザによって実行される実際の操作を挿入します。Actions セクションを繰り返すように仮想ユーザを設定できます（**test**.cfg ファイルを使います）。

vuser_end セクションは，最後，つまり仮想ユーザのすべてのアクションの実行後に実行されます。このセクションには，クリーンアップとログオフの処理を挿入します。**vuser_end** セクションは，スクリプトの実行時に一度だけ実行されます。

注：Mercury のアプリケーションは，UNIX のシグナル，SIGHUP，SIGUSR1，SIGUSR2 を送信することによって仮想ユーザを制御します。仮想ユーザ・プログラムでは，これらのシグナルを使用しないでください。

仮想ユーザの実行環境設定

仮想ユーザの実行環境を設定するには，スクリプトともに作成する **default.cfg** と **default.usp** に変更を加えます。このような実行環境の設定は，VuGen の実行環境の設定に相当します。詳細については，第 13 章「実行環境の設定」を参照してください。**default.cfg** ファイルには，一般設定，思考遅延時間，およびログに関する設定が含まれています。**default.usp** ファイルには，実行論理とペース設定のための設定が含まれています。

一般オプション

UNIX 仮想ユーザ・スクリプト用の一般オプションが 1 つあります。

ContinueOnError は，エラーが発生しても実行を継続するように仮想ユーザに指示します。このオプションを有効にするには，値を 1 にします。このオプションを無効にするには，値を 0 にします。

次の例では，仮想ユーザはエラーが発生しても実行を継続します。

```
[General]
ContinueOnError=1
```

思考遅延時間のオプション

思考遅延時間のオプションを設定して，スクリプト実行時の仮想ユーザによる思考遅延時間の使用法を制御できます。次の表に従って，各パラメータ（Options，Factor，LimitFlag，Limit）を設定します。

オプション	Options	Factor	LimitFlag	Limit
思考遅延時間を無視	NOTHINK	なし	なし	なし
記録された思考遅延時間を使用	RECORDED	1.000	なし	なし
記録された思考遅延時間を乗じる値	MULTIPLY	数値	なし	なし
記録された思考遅延時間のランダムな割合	RANDOM	範囲	最小の割合	最大の割合
記録された思考遅延時間の上限	RECORDED/ MULTIPLY	数値（MULTIPLY 用）	1	秒単位の値

実行時に使用する思考遅延時間を制限するには，LimitFlag 変数を 1 に設定し，Limit により思考遅延時間の上限を秒単位で指定します。

次の例では，思考遅延時間を 50% ～ 150% のランダムな割合で乗じるように仮想ユーザに指定しています。

```
[ThinkTime]
Options=RANDOM
Factor=1
LimitFlag=0
Limit=0
ThinkTimeRandomLow=50
ThinkTimeRandomHigh=150
```

ログのオプション

ログ・オプションは，スクリプトの実行中に簡略または詳細ログ・ファイルを作成するために設定できます。

```
[Log]
LogOptions=LogBrief
MsgClassData=0
MsgClassParameters=0
MsgClassFull=0
```

次の表に従って，各パラメータ（LogOptions，MsGClassData，MsgClassParameters，MsgClassFull）を設定します。

ログの種類	LogOptions	MsgClassData	MsgClassParameters	MsgClassFull
Disable Logging（ログの無効化）	LogDisabled	なし	なし	なし
標準ログ	LogBrief	なし	なし	なし
パラメータの置換（のみ）	LogExtended	0	1	0
サーバから返されたデータ（のみ）	LogExtended	1	0	0
詳細トレース（のみ）	LogExtended	0	0	1
すべて	LogExtended	1	1	1

次の例では，仮想ユーザはサーバによって返されたすべてのデータと，置換に使用されたパラメータのログを記録します。

```
[Log]
LogOptions=LogExtended
MsgClassData=1
MsgClassParameters=1
MsgClassFull=0
```

反復と実行論理

反復のオプションを設定して，反復を複数回実行したり，反復のペースを制御したりできます。また，アクションの順番と重み付けを手作業で設定することもできます。スクリプトの実行論理と反復の設定を変更するには，**default.usp** ファイルを編集する必要があります。

Actions セクションを複数回反復するように仮想ユーザに指示するには，反復の回数を RunLogicNumOfIterations の値として設定します。

反復の間隔（ペース）を指定するには，次の表に従って RunLogicPaceType 変数と関連する値を設定します。

ペースの設定	RunLogicPaceType	関連変数
すぐに次の反復を開始する	Asap	なし
次の反復を開始する前に，指定した時間だけ待機する	Const	RunLogicPaceConstTime
反復の間隔をランダムにする	Random（ランダム）	RunLogicRandomPaceMin, RunLogicRandomPaceMax
反復の後，一定の時間待機する	ConstAfter	RunLogicPaceConstAfterTime
反復の後，ランダムな時間だけ待機する	After	RunLogicAfterPaceMin, RunLogicAfterPaceMax

次の例は，反復を 4 回実行し，反復の間隔はランダムな長さにするよう仮想ユーザに対して指示する設定です。ランダムな間隔の範囲は 60 ～ 90 秒です。

```
[RunLogicRunRoot]
MercIniTreeFather=""
MercIniTreeSectionName="RunLogicRunRoot"
RunLogicRunMode="Random"
RunLogicActionOrder="Action,Action2,Action3"
RunLogicPaceType="Random"
RunLogicRandomPaceMax="90.000"
RunLogicPaceConstTime="40.000"
RunLogicObjectKind="Group"
RunLogicAfterPaceMin="50.000"
Name="Run"
RunLogicNumOfIterations="4"
RunLogicActionType="VuserRun"
RunLogicAfterPaceMax="70.000"
RunLogicRandomPaceMin="60.000"
MercIniTreeSons="Action,Action2,Action3"
RunLogicPaceConstAfterTime="30.000"
```

トランザクションとランデブー・ポイントの定義

VuGen を使用せずに仮想ユーザ・スクリプトをプログラミングするときには，トランザクションとランデブーを有効にするために，仮想ユーザ・ファイルを手作業で設定する必要があります。これらの設定は，**test**.usr ファイルに含まれています。

```
[General]
Type=any
DefaultCfg=Test.cfg
BinVuser=libtest.libsuffix
RunType=Binary

[Actions]
vuser_init=
Actions=
vuser_end=

[Transactions]
transaction1=

[Rendezvous]
Meeting=
```

各トランザクションとランデブーは，**usr** ファイルに定義する必要があります。トランザクション名を [Transactions] セクションに追加します（トランザクション名に続けて "=" を指定します）。各ランデブー名を [Rendezvous] セクションに追加します（ランデブー名に続けて "=" を指定します）。セクションがない場合は，上記のように **usr** ファイルにセクションを追加します。

スクリプトのコンパイル

テンプレートを修正したら，スクリプト・ディレクトリの中で適切な **Makefile** を使用してスクリプトをコンパイルします。C++ でコンパイルをするときは，gnu コンパイラではなく，ネイティブ・コンパイラを使用する必要があります。コンパイラは，次のダイナミック・ライブラリを作成します。

- libtest.so（solaris）
- libtest.a（AIX）
- libtest.sl（HP）

Makefile の適切なセクションを変更することで，他のコンパイラ・フラグやライブラリを指定できます。

汎用のテンプレートを使った作業の場合には，アプリケーションのライブラリとヘッダー・ファイルをインクルードする必要があります。たとえば，アプリケーションで **testlib** というライブラリを使用している場合は，そのライブラリを LIBS セクションに指定します。

```
LIBS        = ¥
      -testlib ¥
      -lLrun50 ¥
      -lm
```

Makefile の変更後，作業ディレクトリのコマンド・ラインで「Make」と入力して，仮想ユーザ・スクリプト用のダイナミック・ライブラリ・ファイルを作成します。

スクリプトを作成したら，コマンド・ラインでその機能を確認します。

UNIX コマンド・ラインから仮想ユーザ・スクリプトを実行するには，次のように入力します。

```
mdrv -usr 'pwd' test.usr
```

ここで，**pwd** は仮想ユーザ・スクリプトが格納されているディレクトリへのフル・パスで，**test**.usr は仮想ユーザ・ファイル名です。スクリプトがサーバと通信し，要求されているすべてのタスクを実行することを確認します。

スクリプトが正常に機能することを確認した後，スクリプトを自分の環境（LoadRunner シナリオ，Performance Center 負荷テスト，Tuning Module セッション，または 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 **Business Availability Center** のマニュアルを参照してください。

付録 D

キーボード・ショートカットの使用

以下は，仮想ユーザ・ジェネレータで使用できるキーボード・ショートカットのリストです。

ALT + F8	現在のスナップショットを比較（Web 仮想ユーザのみ）
ALT + INS	新規ステップの作成
CTRL + A	すべて選択
CTRL + C	コピー
CTRL + F	検索
CTRL + G	指定行へ移動
CTRL + H	置換
CTRL + N	新規作成
CTRL + O	開く
CTRL + P	印刷
CTRL + S	保存
CTRL + V	貼り付け
CTRL + X	切り取り
CTRL + Y	やり直し
CTRL + Z	元に戻す
CTRL + F7	記録オプション
CTRL + F8	相関を検索

CTRL + SHIFT + SPACE	関数の構文を表示（Intellisense）
CTRL + SPACE	補完ウィザード（関数名を補完）
F1	ヘルプ
F3	次を検索（下方）
SHIFT + F3	次を検索（上方）
F4	実行環境の設定
F5	仮想ユーザを実行
F6	表示枠間を移動
F7	EBCDIC 変換ダイアログを表示（WinSock スクリプトの場合）
F9	ブレークポイントの切り替え
F10	仮想ユーザをステップごとに実行

付録 E

記録オプションと実行環境の設定

次の表に，仮想ユーザ・ジェネレータの記録オプションと実行環境の設定のアルファベット順およびカテゴリ別の索引を示します。

実行環境の設定

実行環境の設定では，スクリプトの再生時の仮想ユーザの振る舞いを設定できます。LoadRunner コントローラを使用して仮想ユーザを実行している場合，シナリオの仮想ユーザの振る舞いは実行環境の設定によって制御されます。

実行環境の設定の詳細を確認するには，いずれかの表でその設定を探し，右側のカラムの適切な項を参照してください。次に示すどちらのリストも使用できます。

- 実行環境の設定のアルファベット・50 音順リスト
- 実行環境の設定のカテゴリ別リスト

実行環境の設定のアルファベット・50 音順リスト

実行環境の設定	カテゴリ	参照先
.NET 環境	.NET	742 ページ「.NET 環境の実行環境の設定」
Java VM	Java 環境設定	460 ページ「JVM 実行環境の設定」
Radius	WAP	1238 ページ「Radius 接続データの設定」
RTE	RTE	1151 ページ「RTE 実行環境の設定」
一般	SAPGUI	1075 ページ「SAPGUI 実行環境の設定」

クライアントのエミュレート	Oracle NCA	1031 ページ「Oracle NCA 実行環境設定」
クラスパス	Java 環境設定	461 ページ「実行環境の設定のクラスパス・オプションの設定」
ゲートウェイ	WAP	1234 ページ「ゲートウェイ・オプションの設定」
コンテンツチェック	インターネット・プロトコル	869 ページ「Web ページのコンテンツ・チェックについて」
サーバおよびプロトコル	MMS (マルチメディア・メッセージング・サービス)	1242 ページ「MMS 実行環境の設定」
思考遅延時間	一般	207 ページ「思考遅延時間の設定」
実行論理	一般	200 ページ「ペースの設定と実行論理オプションの設定（シングル・アクション)」
詳細	JMS	387 ページ「Web サービス JMS の実行環境の設定」
設定	Citrix	482 ページ「Citrix 設定実行環境の設定」
速度のシミュレーション	ネットワーク	219 ページ「ネットワーク実行環境の設定について」
その他	一般	210 ページ「その他の設定」
タイミング	Citrix	484 ページ「Citrix 時間設定実行環境の設定」
ダウンロード・フィルタ	インターネット・プロトコル	866 ページ「Web サイトのフィルタリング」
ツールキット・オプション	Web サービス・ツールキット	386 ページ「Web サービス・ツールキットの実行環境の設定」
追加属性	一般	209 ページ「実行環境の追加属性の設定」
ブラウザのエミュレーション	ブラウザ	852 ページ「ブラウザのエミュレーション・プロパティの設定」

プリファレンス	インターネット・プロトコル	857 ページ「インターネット・プリファレンスの設定」
プロキシ	インターネット・プロトコル	847 ページ「プロキシ・オプションの設定」
ペースの設定	一般	199 ページ「実行環境のペースの設定(マルチ・アクション)」
ログ	一般	202 ページ「実行環境設定のログの設定」

実行環境の設定のカテゴリ別リスト

カテゴリ	実行環境の設定	参照先
.NET	.NET 環境	742 ページ「.NET 環境の実行環境の設定」
Citrix	設定	482 ページ「Citrix 設定実行環境の設定」
Citrix	タイミング	484 ページ「Citrix 時間設定実行環境の設定」
Java 環境設定	Java VM	460 ページ「JVM 実行環境の設定」
Java 環境設定	クラスパス	461 ページ「実行環境の設定のクラスパス・オプションの設定」
JMS	詳細	387 ページ「Web サービス JMS の実行環境の設定」
MMS (マルチメディア・メッセージング・サービス)	サーバおよびプロトコル	1242 ページ「MMS 実行環境の設定」
Oracle NCA	クライアントのエミュレート	1031 ページ「Oracle NCA 実行環境設定」
RTE	RTE	1151 ページ「RTE 実行環境の設定」
SAPGUI	一般	1075 ページ「SAPGUI 実行環境の設定」
WAP	Radius	1238 ページ「Radius 接続データの設定」

WAP	ゲートウェイ	1234 ページ「ゲートウェイ・オプションの設定」
Web サービス・ツールキット	ツールキット・オプション	386 ページ「Web サービス・ツールキットの実行環境の設定」
一般	思考遅延時間	207 ページ「思考遅延時間の設定」
一般	実行論理	200 ページ「ペースの設定と実行論理オプションの設定（シングル・アクション）」
一般	その他	210 ページ「その他の設定」
一般	追加属性	209 ページ「実行環境の追加属性の設定」
一般	ペースの設定	199 ページ「実行環境のペースの設定（マルチ・アクション）」
一般	ログ	202 ページ「実行環境設定のログの設定」
インターネット・プロトコル	コンテンツチェック	869 ページ「Web ページのコンテンツ・チェックについて」
インターネット・プロトコル	ダウンロード・フィルタ	866 ページ「Web サイトのフィルタリング」
インターネット・プロトコル	プリファレンス	857 ページ「インターネット・プリファレンスの設定」
インターネット・プロトコル	プロキシ	847 ページ「プロキシ・オプションの設定」
ネットワーク	速度のシミュレーション	219 ページ「ネットワーク実行環境の設定」
ブラウザ	ブラウザのエミュレーション	852 ページ「ブラウザのエミュレーション・プロパティの設定」

記録オプション

記録オプションでは，スクリプトの生成方法，および記録時に仮想ユーザに適用する設定を VuGen に指定できます。

記録オプションの詳細を確認するには，いずれかの表でそのオプションを探し，右側のカラムの適切な項を参照してください。次に示すどちらのリストも使用できます。

- 記録オプションのアルファベット・50 音順リスト
- 記録オプションのカテゴリ別リスト

記録オプションのアルファベット・50 音順リスト

記録オプション	カテゴリ	参照先
Corba オプション	記録プロパティ	444 ページ「CORBA オプション」
Java VM	Java 環境設定	431 ページ「Java 仮想マシン（JVM）記録オプション」
RTE	RTE	1141 ページ「RTE 記録オプションの設定」
Web イベント設定	GUI プロパティ	769 ページ「Web イベント記録の設定」
Web トラップ	HTTP プロパティ（Web/Winsock）	982 ページ「[Web トラップ] 記録オプションの設定」
WinSock	ソケット	558 ページ「WinSock 記録オプションの設定」
一般	プロトコル	63 ページ「プロトコルの追加と削除」
一般	SAPGUI	1060 ページ「SAPGUI 一般記録オプション」
オプション	COM/DCOM	639 ページ「COM スクリプト編集オプションの設定」
記録	Microsoft .NET	736 ページ「Microsoft .NET 記録オプションの設定」
記録	一般	829 ページ「記録レベルの選択」

記録プロキシ	HTTP プロパティ	816 ページ「プロキシ設定を使った作業」
クラスパス	Java 環境設定	433 ページ「クラスパス記録オプションの設定」
コード生成	Citrix	476 ページ「コード生成記録オプション」
コード生成	SAPGUI	1061 ページ「SAPGUI コード生成記録オプション」
コード生成オプション	EJB オプション	1001 ページ「EJB 記録オプションの設定」
自動ログオン	SAPGUI	1062 ページ「SAPGUI 自動ログオン記録オプション」
詳細	GUI プロパティ	766 ページ「GUI プロパティの詳細設定」
詳細	HTTP プロパティ	819 ページ「記録オプションの詳細設定」
シリアル化オプション	記録プロパティ	437 ページ「シリアル化オプション」
スクリプト	一般	83 ページ「スクリプト生成オプションの設定」
設定	Citrix	474 ページ「設定記録オプション」
設定	RTE	1140 ページ「RTE 設定オプションの設定」
相関	HTTP プロパティ	930 ページ「相関記録オプションの設定」
相関オプション	記録プロパティ	439 ページ「相関オプション」
データベース	データベース	525 ページ「データベース記録オプションの設定」
デバッグ・オプション	記録プロパティ	441 ページ「デバッグ・オプション」
トラフィック・フィルタ	トラフィック分析	410 ページ「トラフィック情報の指定」

フィルタ	COM/DCOM	634 ページ「フィルタの設定」
フィルタ	Microsoft .NET	725 ページ「記録用フィルタの設定」
ブラウザ	HTTP プロパティ（Web/WinSock）	828 ページ「記録用のブラウザの指定」
ポートの割り当て	ネットワーク	93 ページ「ポートの割り当て設定」
レコーダ	Citrix	475 ページ「レコーダ記録オプション」
レコーダ・オプション	記録プロパティ	434 ページ「レコーダ・オプション」
ログイン	Citrix	477 ページ「ログイン記録オプション」

記録オプションのカテゴリ別リスト

カテゴリ	記録オプション	参照先
Citrix	コード生成	476 ページ「コード生成記録オプション」
Citrix	設定	474 ページ「設定記録オプション」
Citrix	レコーダ	475 ページ「レコーダ記録オプション」
Citrix	ログイン	477 ページ「ログイン記録オプション」
COM/DCOM	オプション	639 ページ「COM スクリプト編集オプションの設定」
COM/DCOM	フィルタ	634 ページ「フィルタの設定」
EJB オプション	コード生成オプション	1001 ページ「EJB 記録オプションの設定」
GUI プロパティ	Web イベント設定	769 ページ「Web イベント記録の設定」
GUI プロパティ	詳細	766 ページ「GUI プロパティの詳細設定」
HTTP プロパティ	Web トラップ（Web/Winsock）	982 ページ「[Web トラップ] 記録オプションの設定」
HTTP プロパティ	詳細	819 ページ「記録オプションの詳細設定」

HTTP プロパティ	相関	930 ページ「相関記録オプションの設定」
HTTP プロパティ	ブラウザ（Web/WinSock）	828 ページ「記録用のブラウザの指定」
Java 環境設定	Java VM	431 ページ「Java 仮想マシン（JVM）記録オプション」
Java 環境設定	クラスパス	433 ページ「クラスパス記録オプションの設定」
Microsoft .NET	記録	736 ページ「Microsoft .NET 記録オプションの設定」
Microsoft .NET	フィルタ	725 ページ「記録用フィルタの設定」
RTE	RTE	1141 ページ「RTE 記録オプションの設定」
RTE	設定	1140 ページ「RTE 設定オプションの設定」
SAPGUI	一般	1060 ページ「SAPGUI 一般記録オプション」
SAPGUI	コード生成	1061 ページ「SAPGUI コード生成記録オプション」
SAPGUI	自動ログオン	1062 ページ「SAPGUI 自動ログオン記録オプション」
一般	記録	829 ページ「記録レベルの選択」
一般	スクリプト	83 ページ「スクリプト生成オプションの設定」
一般	プロトコル	63 ページ「プロトコルの追加と削除」
記録プロパティ	Corba オプション	444 ページ「CORBA オプション」
記録プロパティ	シリアル化オプション	437 ページ「シリアル化オプション」
記録プロパティ	相関オプション	439 ページ「相関オプション」
記録プロパティ	デバッグ・オプション	441 ページ「デバッグ・オプション」

記録プロパティ	レコーダ・オプション	434 ページ「レコーダ・オプション」
ソケット	WinSock	558 ページ「WinSock 記録オプションの設定」
データベース	データベース	525 ページ「データベース記録オプションの設定」
ネットワーク	ポートの割り当て	93 ページ「ポートの割り当て設定」

索引

記号

.NET 仮想ユーザ，「Microsoft .NET 仮想ユーザ・スクリプト」を参照
.NET 内のフィルタ
  Impact ログ 727
  管理 727
  設定 725
  設定のガイドライン 721
  選択 725
  操作 727
  定義 723
  包含する要素の指定 722
  利点 719
.NET フィルタ 362

数字

35090 791
35091 791
35092 790

A

ABC アイコン 127
Accept-Language リクエスト・ヘッダ 860, 1324
Action
  セクション 58
  メソッド 607
Actions クラス 605
ActiveX，サポートの有効化 864
ADO .NET 仮想ユーザ 715
ALNUM フラグ 886
AMF Call Properties ダイアログ・ボックス 702
AMF 仮想ユーザ・スクリプト
  エンベロープ・ヘッダー・セット・プロパティ 704
  概要 685, 701
  関数リスト 693
  記録モードの設定 687
  スクリプトの記録 685
  相関 695
  表示 699
  ヘッダー・セット・プロパティ 703
  呼び出しプロパティ 702
AMF の用語 687
ANSI 1140
ANSI C のサポート，ユーザ定義スクリプト 597
AssignToParam プロパティ（Web）890
AUT 設定 742

B

BIN フラグ 886

C

CARRAY バッファ 1207
Citrix エージェント 507
Citrix 仮想ユーザ・スクリプト 465–506
  概要 467
  関数リスト 497
  記録オプション 480
  記録と再生のヒント 502
  クライアントのバージョン 469
  サーバからの接続解除 469
  再生の同期化 487
  実行環境の設定 482
  表示設定 481
  編集 486
Citrix サーバの接続解除 469
CLR 仮想ユーザ 715
COM
  概要とインタフェース 626
  データ型 628
COM オブジェクトのインスタンスの作成 644
COM 仮想ユーザ・スクリプト
  IDispatch インタフェース 648, 655

lrc_ 型関数 655
Visual Basic コレクション 662
インスタンス化，オブジェクト 644
インスタンスを作成する関数 654
インタフェースの取得 645
インタフェース・ポインタ 642
エラー検査 643
オブジェクトのインスタンスの作成 644
開始 628
開発 625
概要 642
型指定関数 655
関数 653
記録オプション 632
記録対象の COM オブジェクトの選択 629
クラスのコンテキスト 627
スクリプトの構造 642
相関候補の検索 650
タイプ・ライブラリ 627
デバッグ関数 663
デバッグ用ログ・ファイル 629
パラメータ化関数 658
バリアント型変換関数 656
CORBA-Java 仮想ユーザ・スクリプト 665
記録 666
記録オプション（Corba オプション） 444
シリアル化オプション 437
相関オプション 439
デバッグ・オプション 441
レコーダ・オプション 434
CORBA 記録オプション・ダイアログ・ボックス 444
CtLib 519
オプション 527
結果セット・エラー 543
サーバ・メッセージのログの記録 206
C 仮想ユーザ 595
C 関数
仮想ユーザ・スクリプトでの使用 33
追加キーワード 1274
デバッグ用 1274
C 言語サポート
インタプリタ 34
規則 597

D

Date/Time（日時），パラメータ値 169
DB2-CLI 519
DBCS 1140
DbLib 519
DCOM 626
DCOM ノード 634
Detector，EJB 992
DIG フラグ 886
DLL，仮想ユーザ・スクリプトからの呼び出し 1313
DLL のロード
概要 1313
グローバル 1314
ローカル 1313
DN（LDAP）713
DNS 仮想ユーザ
概要 551
関数 552
DNS キャッシング
Web 861
DOM のメモリ割り当て 865
DOM メモリ割り当て 865
DSL 220

E

EBCDIC 変換 581
EJB
インスタンス 1004
仮想ユーザ・スクリプト 991
コード生成オプション 1001
メソッド 1006
EJB Detector 1004
コマンド・ライン 993
設定 992
ログ・ファイル 996
end メソッド 606
EUC-JP エンコーディング
記録オプション 820
設定 1322
EUC エンコードされた Web ページ 842
Expect プロパティ，Web チェック 891

F

failed bitmap synchronization 494
FIELDTBLS 環境設定 1204
Flash Remoting 685
FLDTBLDIR 環境設定 1204
Forms Listener 1034
Frame プロパティ，オブジェクト・チェック（Web）890
FTP プロトコル
　関数リスト 706
　記録 705

G

GUID 627
GUID（Global Unique Identifier）627
GUI 仮想ユーザ・スクリプト
　ツール 9
gzip 868

H

history オブジェクト，サポート 865
HotSync 987
HTML
　パラメータの最大文字数 951
HTML タグ・オブジェクト 772
HTML パラメータの最大文字数 951
HTML ビュー（Web スナップショット）24
HTML ベース・モード 829
HTML 用の圧縮（gzip）868
HTTP
　バッファ・サイズ（Web）862
HTTP 記録モード，WAP 1234

I

ica ファイル 495
IC フラグ 886
IDispatch インタフェース 648, 655
If-Modified-Since ヘッダー
　Web 855
IIOP 676
IMAP（Internet Messaging）1171
IMAP プロトコル 1171
Impact ログ，.NET フィルタ 727
include コマンド，.NET フィルタ 731
Informix 519
init メソッド 606
IntelliSense 40
ISDN 220
IUnknown インタフェース 627

J

Jacada 仮想ユーザ・スクリプト 1183
　概要 1189
　記録 1186
　再生 1188
JavaScript 仮想ユーザ 602
Java 仮想マシン
　記録オプション 431
　実行環境の設定 460
Java 仮想ユーザ（CORBA，RMI）
　記録オプション，Java VM 431
　記録オプション，シリアル化 437
　記録オプション，相関 439
　記録のヒント 672
　クラスパスの実行環境の設定 461
　実行環境の設定－ Java VM 460
Java 仮想ユーザ（すべて）
　Java メソッドの編集 605
　環境設定 617
　実行環境の設定 459, 462
　ステートメントの相関 447
　プログラミング 603
　ランデブー・ポイントの挿入 612
Java 仮想ユーザ（ユーザ定義）
　Java コードの使用 599
　テンプレートの作成 605
Java プラグイン 672
JDNI のプロパティ
　EJB Home の検索 1002
　指定 998
　詳細，コンテキスト・ファクトリ 999
JMS
　Web サービス 347
　概要 346
　関数 346
　実行環境の設定 388
　トランスポート・メソッド 346
Jscript 84

K

Keep-Alive 接続，Web 859

Kerebros
　認証 862
Knowledge Base xxvi

L

LDAP プロトコル
　WinSock 555
　関数リスト 710
　記録 709
libc 関数，呼び出し 597
LoadRunner Readme ファイル xxv
log
　詳細レベルの設定 - UNIX 1335
lrbin.bat ユーティリティ 1256
lrc 関数 641
lrd（データベース）関数 529
lreal 関数 1218
lrs 関数 561
lrt 関数 1196

M

MAPI
　関数を使った作業 1176
MatchCase プロパティ 890
Media Player 1218
Mercury Best Practices xxvii
Mercury Customer Support Web site xxvii
Mercury Home Page xxvii
META 更新 862
Microsoft .NET 仮想ユーザ・スクリプトの記録
　概要 716, 717
　記録 715
　記録オプション 736
　実行環境の設定 742
　制限事項 716
　相関 746
　データ・グリッドの表示 744
　トラブルシューティング 749
　フィルタの管理 727
　フィルタの操作 727
mkdbtmpl スクリプト（UNIX）1331
MMS 仮想ユーザ・スクリプト
　実行環境の設定 1242
MMS 関数（MS Media Player）1218
MMS マルチメディア・メッセージング 1241
MS
　Exchange プロトコル（MAPI）1176
　SQL Server，記録 519
MS Query 147
MTS のコンポーネント 637

N

NCA 仮想ユーザ，「Oracle NCA」を参照
NTLM
　セキュリティ 862
　認証 78

O

ODBC の記録 519
OnFailure プロパティ，Web チェック 891
Oracle
　2-tier データベースの記録 519
　バージョン 8.0 532
Oracle Configurator 1034
Oracle NCA 仮想ユーザ・スクリプト
　仮想ユーザ関数の使用 1025
　記録作業のガイドライン 1018
　サーブレット・テスト 1034
　作成 1015
　実行環境の設定 1031
　セキュアなアプリケーション 1034
　接続モードの確認 1035
　相関 1037
Oracle Web Applications 11i 仮想ユーザ・スクリプト
　GUI ベースの詳細オプション 766
　関数 798
　ヒント 780
Oracle アプリケーションのデバッグ 1277
OrbixWeb 674
OTA，Over-The-Air 1229

P

Palm
　アプリケーションの記録 987
　プロトコル 975
PAP，プッシュ・アクセス・プロトコル 1229
PeopleSoft Enterprise 仮想ユーザ・スクリプト
　GUI ベースの詳細オプション 766
　関数 798
　ヒント 780

PeopleSoft-Tuxedo 仮想ユーザ 1193
実行 1291
Performance Center
仮想ユーザ・スクリプトの管理 269
接続 270
POP3 （ポスト・オフィス）プロトコル 1178
PPG，プッシュ・プロキシ・ゲートウェイ 1229

Q

QC 251
Quality Center
インポート 371
仮想ユーザ・スクリプトの管理 251
仮想ユーザ・スクリプトを開く 255
スクリプトの管理 251
スクリプトの保存 257
接続 252
切断 254
バージョン管理 258
バージョン・コントロール 258
Quality Center からの切断 254
Quality Center からの VuGen の切断 254
Quality Center の接続と切断 252
Quality Center プロジェクトからスクリプトを開く 255
Quality Center プロジェクトへのスクリプトの保存 257
Quality Center への VuGen の接続 252
Quality Center への接続ダイアログ・ボックス 252
Quality Center を使ったスクリプト管理について 251

R

Radius
サポート 1238
実行環境の設定（WAP）1238
RealPlayer 1215
Repeat プロパティ，Web 仮想ユーザ 891
Report プロパティ，Web チェック 891
RMI-Java 仮想ユーザ・スクリプト
IIOP を介した記録 676
記録 677
シリアル化オプション 437
相関オプション 439
デバッグ・オプション 441
レコーダ・オプション 434
RTE 仮想ユーザ・スクリプト
PC キーボードの割り当て 1129
TE 関数の使用 1127
概要 1123, 1125
画面からのテキストの読み取り 1167
記録 1131
作成手順 1125
実行環境の設定 1152
同期化 1157
紹介 1124
run_db_vuser シェル・スクリプト 237

S

S_SSA_ID テーブル 1290
SAML オプション 356
SAPGUI/SAP-Web デュアル・プロトコル 1043, 1073
SAPGUI 仮想ユーザ・スクリプト
sapgui 関数の使用方法 1078
カーソル位置での記録 1058
関数リスト 1078
共通記録オプション 1060
記録 1043
記録オプションの設定 1060
コード生成記録オプション 1061
再生 1073
実行環境の設定 1074
自動ログオン記録オプション 1062
ステップの挿入 1063
スナップショット 1063
SAP-Web 仮想ユーザ・スクリプト
記録 1097
記録オプション 1100
実行環境の設定 1103
SED ユーティリティ 760
setInterval，setTimeout のしきい値 864
Siebel
2 層タイプのスクリプト作成ヒント 1288
ベース 36 キーの値 1290
Siebel-Web
記録 1106
相関 922, 1107
トラブルシューティング 1116

Siebel 相関ライブラリ 1107
Siebel の接尾辞値 1288
SJIS（Shift Japan Industry Standard）842
SMTP プロトコル 1179
SOAP 応答，保存 397
SOAP ヘッダー 338
SOA テスト
　開始 296
　開発 295
　実行 385
　自動テストの作成 297
　テスト・アスペクトの選択 297
SOA のテスト・アスペクト 297
Solaris
　ASCII 変換 559
SQL ステートメント 148
SSL
　サーバ・トラフィック・スクリプトの証明書 412
SWECount，相関 1115

T

TE（RTE）関数 1127
TestDirector，「Quality Center」を参照 251
TE システム変数 1163
TUXDIR 環境設定 1204
Tuxedo 仮想ユーザ・スクリプト 1193
　lrt（Tuxedo）関数の使用 1196
　Tuxedo 6，7 1193
　概要 1201
　環境設定 1204
　システム変数 1204
　実行 1202
　データ・バッファ 1203
　データ・ファイルの表示 1203
　バージョン 1205
　ログ・ファイル 1202

U

UDDI
　検索 370
　サーバ情報の指定 370
　情報 368
UNIX
　コマンド・ライン 236
URL ステップ
　定義（Web 仮想ユーザ）807
　変更 897
URL ステップのプロパティ・ダイアログ・ボックス
　Web 898
URL ベース・モード 829
UTF-8 変換
　記録オプション 820
　実行環境の設定 861
　設定 1322

V

VBScript 仮想ユーザ 601
VBA 実行環境の設定 217
VBA リファレンス 217
VB 仮想ユーザ 600
verify_generator 237
Visigenic 674
Visual Studio 1251
　スクリプトの表示 741
Visual Basic
　仮想ユーザ・スクリプト 1251
　スクリプト・オプション 84
Visual C，Visual Studio の使用 1251
VM （仮想マシン）431
VM 実行環境の設定，Web サービス 388
VuGen
　開始 15
　仮想ユーザ・スクリプト 31
　仮想ユーザ・スクリプトの記録 57
　環境オプション 16
　紹介 13
　ツールバー 72
VuGen でのスクリプトのバージョン管理 258
vuser_init，vuser_end セクション 58

W

WAP 仮想ユーザ・スクリプト
　実行環境の設定 1233
　デバッグ情報 867
Wdiff 188
web_set_user 関数の生成 78
Web（Click and Script）
　ActiveX サポート 864
　アプレット・サポート 864

一般オプション 863
タイマ 863
ナビゲータのプロパティ 865
メモリ管理 865
履歴オプション 865
Web（Click and Script）仮想ユーザ・スクリプト
GUI ベースの詳細オプション 766
Web ページの内容のチェック 869
インターネット記録オプション 815
関数 798
記録オプションの設定 765
記録済み関数の表示 762
記録レベルの選択 829
結果サマリ・レポート 243
実行環境の設定 846
実行時ビューア 234
ステップの削除 896
ステップの追加 895
説明 756
テキストと画像の検証 877
デバッグ機能の有効化 225
デバッグ・ツール 234
トラブルシューティングのヒント 789
内容のフィルタリング 823
プロキシ設定 816
変更 893
ユーザ定義ヘッダー 821
ログ表示オプション 225
Web/WinSock 仮想ユーザ・スクリプト 975
Web トラップ・オプション 982
開始 977
記録 985
プロキシ設定 980
Web イベント記録の設定 769–780
カスタマイズ 772–780
標準 771–772
Web イベント記録の定義ダイアログ・ボックス 771
Web 仮想ユーザ・スクリプト
インターネット記録オプション 815
概要 755
画像チェック 887
関数 795
記録オプション 827
記録用のブラウザの指定 828
結果サマリ・レポート 243
実行環境の設定 219, 846
実行時ビューア 234
紹介 753
ステップの削除 896
ステップの追加 895
セクション 759
相関 922
チェック 877
テキストと画像の検証 877
デバッグ機能の有効化 225
デバッグ・ツール 234
内容のフィルタリング 823
パワー・ユーザ向けのヒント 961
プロキシ設定 816
変更 893
ユーザ定義ヘッダー 821
ユーザ定義要求ステップ 911
ログ表示オプション 225
ログ表示オプションの設定 234
Web 仮想ユーザ・スクリプトの変更
URL ステップ 897
画像ステップ 901
思考遅延時間 916
データを送信ステップ 907
トランザクション 914
フォームを送信ステップ 903
ランデブー・ポイント 915
Web から Java への変換 760
Web 関数，使用 801
Web サービス仮想ユーザ・スクリプト
XML ツリー・クエリ 392, 699
概要 308
関数 399
記録 312
サービスの管理 365
作成 307, 327
実行 385
実行環境の設定 386
スナップショット 389
セキュリティ・ポリシー 350
パラメータ化 396
メッセージ署名 353
ユーザ・ハンドラ 360
レポート・ツール 400
Web サービスの管理 366

Web サービス呼び出し
追加 328
プロパティ 328
Web トラップ 982
Web トラップ・ノード 984
Web の相関 919
Web パフォーマンス・グラフ
Web 仮想ユーザの生成 858
Windows Sockets 仮想ユーザ・スクリプト
lrs 関数の使用 561
概要 556
記録 555
スクリプト・ビューとツリー・ビュー 556
ソケットの除外 560
データ・バッファ 578
データ・ファイル 580
データ・ファイルの表示 578
データを使った作業 565
WinInet エンジン（インターネット・プロトコル）858
WinSock データ内の移動 568
WS-Addressing 341
WSDL 文書
回帰テスト 380
添付ファイル 335
比較 381
WS-I 検証
設定 374
WS*xxx* Tuxedo 変数 1204

X

XML
Web サービスでのツリーの編集 394
属性での作業 1265
ツリーに対するクエリ 392
テスト 953
ファイルの比較 384
ユーザ定義の要求 956
要素のパラメータ化 396
XML API
プログラミング 1257
XP ウィンドウの形式，Citrix 473
X SYSTEM メッセージ（RTE）1158

Z

Zip ファイル
エクスポート 73
作業 66
使用 76
zlib ヘッダ 860

あ

アクション
インポート 77
オープン・モードの設定 27
順序の並べ替え 78
マルチ・アクションの記録 70
アクション・ステップ
関数リスト― Web 802
変更（Web）897
アクションの分割 85
アスペクト
SOA のテスト 297
アセンブリ，.NET での追加 729
圧縮ヘッダー，要求 860
アップロード
Performance Center へのスクリプトの 271
必要な VuGen のバージョン 272
アプリケーション配備ソリューション，Citrix 仮想ユーザ・タイプ 465–506
アプリケーション・サーバ，Oracle NCA 1018
暗号化データ，Web サービス・セキュリティのための 353
暗号化，パスワード 121

い

異常系テスト 297
一意カラム名 1287
一意の値のパラメータの割り当て 155
一意の数，パラメータ値 175
一時停止，仮想ユーザ 227
一致する次の文字列を置換コマンド 136
一般オプション
Citrix 表示 481
環境タブ 16
再生タブ 224
すべての仮想ユーザ 141
相関タブ 938
ダイアログ・ボックス 142

パラメータ化タブ 140
表示タブ（Web のみ）225
移動コマンド 234
イベント記録設定のカスタマイズ 772–780
手順 774
リッスン・イベントの追加 775
リッスン条件の指定 776
イベント記録の設定 769–780
設定レベル 771
リセット 779
レベルのカスタマイズ 772
印刷ダイアログ・ボックス（Web レポート）249
インポート
アクション 77
データベースからのデータ 146
インポート，サービス 369

う

ウィザード，ワークフロー 43
ウィンドウ名，Citrix 475
ウィンドウを並べて表示 235

え

エージェント，Citrix 507
エクスポート，スクリプト
Word ファイル 89
Zip ファイル 73
エスケープ・シーケンス 584
エディタのフォント 16
エディタ，フォントの設定 16
エミュレーション・サーバ
開始 284
エミュレートされたサービス
仮想ユーザ・スクリプト 292
結果 287
再ロード 290
作成 283
操作 290
操作のルール 288
動作 287
標準の応答 287
ホストの選択 285
ルールの設定 288
エラー処理 116, 211
COM 仮想ユーザ・スクリプト 643
グローバル変更 542
ローカル変更（重要度）542
エラー，スナップショットを生成する 211
エラーでスナップショットを生成する 211
エラーでも処理を継続する 116, 211
エラーの一致，制限 862
エラー・メッセージ・ダイアログ・ボックス 115
エンコード，EUC 842, 844
エンジン，記録 820

お

オートメーション対応 594
オプション
CtLib 527
lrd ログ 526
記録（RTE）1141
オプションのウィンドウ 1069
オプションのウィンドウ（SAPGUI）1074
オンライン・ブラウザ 234, 968
オンライン・リソース xxvi

か

カーソル位置での記録 1058
回帰テスト，WSDL 380
外部関数 1313
拡張結果設定 527
拡張ログ・オプション 205
画像ステップのプロパティ・ダイアログ・ボックス 902
画像チェック
Web 仮想ユーザ・スクリプト 887
変更（Web）901
仮想マシンの設定 388
仮想ユーザ
紹介 4
仮想ユーザ ID，パラメータ値 178
仮想ユーザ関数
AMF 693
ctrx（Citrix）497
C 関数の一覧 35
DNS 552
FTP 706
imap 1174
Java 608
lrc（COM）644

lr（C 関数）32
lrd（データベース）529
lreal 1218
lrs（WinSock）561
lrt（Tuxedo）1196
mapi 1176
mms（MS Media Player）1218
Oracle NCA 1025
pop3 1178
sapgui（SAP）1078
smtp 1179
TE（RTE）1127
Web HTTP 801
Web（Click and Script）798
Web サービス 399
一般（C）32
「オンライン関数リファレンス」を参照
外部，ユーザ定義 1313
構文 41
単語の自動補完 40
仮想ユーザ・ジェネレータ，「VuGen」を参照
仮想ユーザ情報の取得 112
仮想ユーザ情報の取得（Java）613
仮想ユーザ・スクリプト
C 言語サポート 597
Java 言語の記録 417
Performance Center のアップロード / ダウンロード 269
Performance Center へのアップロード 271
Quality Center の統合 251
SOA 自動スクリプト 295
UNIX，コンパイル 1339
UNIX での作成 1329–1340
UNIX での実行 236
VuGen の使用 221
Web サービス 307, 327
Zip からのインポート 66
Zip からの作業 66
拡張 105
関数の追加 105
コマンド・プロンプトからの実行 236
コメント，挿入 111
サーバ・トラフィック 403
再生成 80
実行 221
実行環境の設定 191
実行環境の設定ー Java 459–462
シナリオへの統合 239
ステップごと 229
ストリーミング・データ 1215
生成言語の選択 83
セクション 58
セッション・ステップの統合 239
タイプ，「仮想ユーザのタイプ」を参照
デバッグ機能 229
トランザクション 107
バージョンの履歴 263
パラメータ化 123
表示 18
表示実行モード 223
開く 66
プログラミング 593, 1329–1340
ユーザ定義 593
ランデブー・ポイント 110
仮想ユーザ・スクリプト内のブックマーク 233
仮想ユーザ・スクリプトの記録
AMF 685
CORBA-Java 666
DNS 551
FTP 705
LDAP 709
Oracle NCA 1017
RMI-Java 675
SAPGUI 1043
SAP-Web 1097
Tuxedo 1193
Web/WinSock 975
Window Sockets 555
概要 57
データベース 523
プロキシ設定 816
メール・サービス 1171
ワイヤレス 1223
仮想ユーザ・スクリプトのセクション 58
仮想ユーザ・スクリプトの同期化
カーソル表示の待機 1162
概要 118
概要（RTE）1157
キャラクタ・モード（VT）・ターミナル 1162
ターミナルの安定の待機 1165

テキスト表示の待機（RTE）1163
ブロック・モード（IBM）・ターミナル 1158
仮想ユーザ，定義 4
仮想ユーザの再生成
全プロトコル 80
仮想ユーザのタイプ
COM 644
CORBA-Java 665
EJB テスト 991
Jacada 1183
Java（プログラミング）603
Media Player 1215
Real Player 1215
一覧 7
一覧（ポップアップの説明）64
仮想ユーザの比較 188
画面上のテキストの検索（RTE）1168
環境設定
Java 617
Tuxedo 仮想ユーザ 1204
環境タブ 16
環境の実行環境の設定，.NET 742
関数
AMF 693
ctrx（Citrix）497
C 関数の一覧 35
DNS 552
FTP 706
GUI レベル（PeopleSoft，Oracle）798
imap 1174
Java 608
lrc（COM）641
lr （C 関数）32
lrd（データベース）529
lreal（Real Player）1218
lrs（WinSock）561
lrt（Tuxedo）1196
mapi 1176
mms 1218
pop3 1178
sapgui（SAP）1078
smtp 1179
TE（RTE）1127
WAP 1228
Web（Click and Script）798
Web 仮想ユーザ・スクリプト 801
Web サービス 399
構文 41
単語の自動補完 40
関数構文の表示 41
関数の表示 40
関数の無効化（SAPGUI）1074
管理
VuGen における Web サービス 365

き

キーボードのショートカット
記録オプション 88
実行環境の設定 192
ショートカット・リスト 1341
キーボードの割り当て 1129
キーボードの割り当て（RTE）1129
キーワード，追加 1274
規則，表記 xxviii
既存のパラメータで置換コマンド 135
基本のイベント記録設定レベル 771
キャッシュ
更新バージョンのチェック 855
反復ごとにクリア 855
ロードとダンプ 808
キャプチャ・ファイル
生成 406
境界条件のテスト 297
境界，相関のための定義 950
行カウント，取得 1284
行情報，データベース仮想ユーザ 536
記録
Web サービス 312
ステータス，オプション 777
記録エンジン 820
記録オプション 558
.NET 仮想ユーザ 736
.NET の記録 737
Corba オプション 444
Java 言語 429–444
RTE 1141
RTE の設定 1140
Web 827
WinSock 558
アルファベット順リスト 1347
インターネット・プロトコル 815

キーボードのショートカット 88
記録（Web）843
記録プロキシ（Web/WinSock）980
記録プロキシ（Web，ワイヤレス）816
コード生成（Citrix）476
詳細（Web，ワイヤレス）819
スクリプト（FTP，COM，Mail）84
データベース 525
デバッグ（Java）441
ブラウザ（Web）828
ポートの割り当て 94
レコーダ（Java）434
ログイン（Citrix）477
記録開始ダイアログ・ボックス 67
記録ボタン 67
記録ログ・タブ 75

く

クエリ
XML ツリー 392
区切り文字，カラム
データ・テーブル内 152
データ・ファイル内 150
クライアント，Citrix 対応 469
クライアント側のデジタル証明書 961
クライアントのエミュレーション
Oracle NCA 1031
クライアントのエミュレーション（Web サービス）386
クラスパス
記録オプション 433
実行環境の設定 461
グラフ
Web 仮想ユーザ・スクリプトでの有効化 858
グリッド
.NET での有効化 745
.NET で非表示 745
非表示 235
表示 537, 744
グループ名，パラメータ値 171
グローバル・ディレクトリ 141

け

形式
パラメータ化用 180
表示バッファのデータ 584
ゲートウェイの設定（WAP）1234
結果サマリ・レポート 243
Web サービス仮想ユーザ 400
Web スクリプトのデバッグ 243
印刷 249
概要 245
検索 248
情報のフィルタリング 247
ツリーの分岐 245
開く 248
結果サマリ・レポートの印刷 249
結果ディレクトリの指定ダイアログ・ボックス 224
言語エンコーディング 1322
言語ヘッダ 1324
検索と置換ダイアログ・ボックス 136
検証
SOAP メッセージ 379
WSDL ファイル 373
WS-I への準拠 374
サービス 375
手作業での呼び出しの追加 375
検証チェック
RTE 1167
SAPGUI 1069
Web 857
Web（クリック＆スクリプト）789

こ

高位のイベント記録設定レベル 771
更新，文書 xxvii
更新方法，パラメータ化 156, 181
構文，関数の表示 41
コード生成オプション（EJB）1001
コピーと貼り付け
RTE 仮想ユーザ 1138
WinSock 仮想ユーザ用の詳細設定 573
コマンド・プロンプト 236
コマンド・ライン引数
C 仮想ユーザ・スクリプトでの読み取り 119
Java 仮想ユーザ・スクリプトでの読み取り 616
UNIX 仮想ユーザ・スクリプト 236
解析関数 36

コメント
仮想ユーザ・スクリプトへの挿入 111
画面ヘッダー・コメント（RTE）1143
相関ステップの追加 925
コメントの挿入ダイアログ・ボックス 111
コメントを挿入ボタン 111
コントローラ
シナリオ 240
コンパイル
UNIX 仮想ユーザ・スクリプト 1339

さ
サーバ・サイド圧縮を受け付ける 860
サーバ・トラフィック
基本スクリプトの作成 408
スクリプトの概要 405
サービス
エミュレートされたサービスの有効化 291
管理 366
テスト 307, 327
リストからの削除 372
サービス・エミュレーション
概要 283
作成 283
新規エミュレーションの追加 286
サービス・ステップ
ツリー・ビューの変更（Web）917
サービス・ステップのプロパティ・ダイアログ・ボックス 917
最近使用したテストのリストからテストを開く 256
再生タブ，一般オプション・ダイアログ・ボックス 224
サムネイル
注釈の追加 29
名前の変更 29
表示 25
ワークフロー・ウィザード 28
参照，.NET のための追加 729

し
識別名 713
思考遅延時間
Siebel に推奨する思考遅延時間 1119
Web 仮想ユーザ・スクリプトの変更 916
関数（C）38
関数（Java）610
しきい値，WinSock 561
しきい値，データベース 525
実行環境の設定 207
挿入 118
ダイアログ・ボックス（Web ツリー・ビュー）916
定義 207
思考遅延時間ダイアログ・ボックス 118
システム変数
RTE 1163
Tuxedo 1204
持続的な接続，Web 859
実行環境の設定
.NET 環境 742
Java 459–462
JMS 387
MMS 1242
Oracle NCA 1031
Radius（WAP）1238
RTE 1152
VBA(Visual Basic Apps) 217
WAP 1233
アルファベット順リスト 1343
インターネット・プロトコル（Web など）845
キーボードのショートカット 192
クライアントのエミュレーション（Oracle NCA）1031
ゲートウェイ・ノード（WAP）1234
索引 1343
思考遅延時間 207
実行論理 193
手動で設定 1333
ショートカット 192
全プロトコル 191
速度のシミュレーション 220
その他 210
その他の属性 209
ダイアログ・ボックス 192
ツールキット・オプション（Web サービス）386
デバッグ情報（WAP）867
内容チェック・ノード（Web）869

ネットワーク 219
ブラウザ・エミュレーション・ノード 852
プリファレンス - 詳細 858
プリファレンス（インターネット・プロトコル）857
プロキシ（インターネット・プロトコル）847
ペースの設定ノード 199
ログ・ノード 202
実行環境の追加属性の設定 209
実行コマンド 226
実行時の完全トレース 205
実行時ビューア
VuGen での使用のヒント 968
VuGen で有効化 234
表示オプション 225
実行レポート（Web のみ）973
実行論理の設定 193
自動回復 16
自動検出，プロトコル 99
自動トランザクション
Web およびワイヤレス・プロトコル 858
一般 216
自動プロキシ設定スクリプト 849
シナリオ
VuGen で作成 240
仮想ユーザ・スクリプトの統合 239
受信トラフィック 410
出力ウィンドウ 613
再生タブ 227
実行時データ・タブ 228
非表示 227
表示 / 非表示 235
出力引数，Web サービス 332
出力メッセージ・ダイアログ・ボックス 115
順次方式でのパラメータの割り当て 154
詳細 GUI ダイアログ・ボックス 766
詳細記録オプション 819
詳細相関（Java）449
消失したスクリプトの回復 16
証明書
サーバ・トラフィックの SSL 412
署名，Web サービス・メッセージ 353
シリアル化（Java 相関）452
シリアル化オプション 439
新規仮想ユーザダイアログ・ボックス
RTE 1133
新規カラムの追加ダイアログ・ボックス 146
新規作成ボタン 1133
診断
VuGen で有効化 216

す

スクリプト・ジェネレータ，「VuGen」を参照
スクリプト生成言語 83
スクリプトのチェック・イン 261
スクリプト・モード 27
ステップの削除
Web 仮想ユーザ・スクリプトから削除 896
ステップ・ボタン 229
ストリーミング・データ・プロトコル
mms 関数 1218
RealPlayer 関数 1218
記録 1216
スナップショット
Citrix 仮想ユーザ 486
Citrix の記録での保存 476
SAPGUI 仮想ユーザ 1063
Web サービス・スクリプト 389
Web ページ 21
Winsock バッファ 566
XML 仮想ユーザ 954
エラーで生成する 211
再生時のスナップショットをローカルに保存する 859
すべて折りたたみ 246
すべて展開コマンド 246
すべて閉じるコマンド 235
スレッドセーフ・コード 620
スレッド，メイン（Java プログラミング）621

せ

制御ステップ
関数（Web）806
変更（Web）914
正常系テスト 297
生成ログ・タブ 75
セーフ配列ログ（COM）640

セキュリティ
Web サービスの設定 354
トークンと暗号化 350
セキュリティ属性
Web サービス仮想ユーザ 350
セキュリティ・ポリシー，Web サービス 350
セキュリティ例外，.NET 749
セッション・ステップ
仮想ユーザ・スクリプトの統合 239
接続試行回数，変更（RTE）1153
接続ダイアログ・ボックス（RTE）1136
接続，開いている接続を閉じる 716
接続プール 743
設定
アプリケーションのセキュリティと権限 749
設定，「実行環境の設定」を参照
設定ファイル
SAML セキュリティ 363
ユーザ・ハンドラ，mmdrv 363
設定レベル
カスタマイズ 772–780
標準 771–772

そ

相関
COM 仮想ユーザ 650
HTML ステートメント（Web）919
Java ステートメント 447
Microsoft .NET スクリプト 746
Siebel のデバッグに関するヒント 1288
Siebel-Web 922, 1107
SWECount 1115
Tuxedo 1210
Web 仮想ユーザのルール 923
概要 183
関数（C）185
関数（Java）187
既存パラメータの変更 190
既知のコンテキスト（Web）922
記録オプション -Java 439
記録後（Web，Wireless）933
最大パラメータ・サイズ 923
詳細プロパティ 928
スクリプト言語オプション 87
スナップショット（Web）933
データベース仮想ユーザ・スクリプトの検索 546
相関オプション，スクリプト記録オプション 87
相関クエリー・タブ
COM 651
データベース 546
相関結果タブ 941
相関タブ 930
相関を検索コマンド
データベース仮想ユーザ 546
送信トラフィック 410
速度のシミュレーション設定 220
その他の実行環境の設定 210

た

ターミナル・エミュレーション 1131
ターミナル・サービス
Citrix 仮想ユーザ 503
ターミナルの安定の待機（RTE）1165
タイムアウト
Citrix 接続 484
HTTP 要求 860
WAP 接続 860
タイム・スタンプ（データベース）527
ダウンロード
Performance Center から VuGen への 276
ダウンロード・フィルタ 866
多国語のサポート 1317–1327
パラメータ・ファイル 1320
タスク表示枠 44
妥当性チェック・レポート 377
単語の補完 40
端末設定ダイアログ・ボックス 1135

ち

チェック（Web）
概要 877
画像チェック 887
関数 803
スクリプトの変更 918
タイプ 879
テキスト 880
プロパティの定義 890
チェック・アウト操作の取り消し 267
チェック・イン・コマンド 261

チェックのプロパティ・ダイアログ・ボックス 918
チェックポイント
期待値 319
結果の表示 323
設定 318
パラメータ化 322
チェックポイント，Web サービス・スクリプト 318
チェックポイントでの期待値 319
遅延，エミュレートされたサービス 291
中位のイベント記録設定レベル 771
重複キー違反
Oracle，MSSQL 1285
Siebel 1288

つ

ツールキット
Web サービス・オプション 386
Web サービスの選択 370
実行環境の設定 386
ツリー・ビュー
ステップの挿入 20
すべての仮想ユーザ 20

て

定義，パラメータのプロパティ
一般 133
データ・ファイル 150
テーブル 152
停止，仮想ユーザ 227
データ・ウィザード 147
MS Query 147
SQL ステートメント 148
データ・グリッド
表示 744
データ・グリッドの有効化 235
データ・テーブルのパラメータ
行とカラムの追加 146
データ・ウィザードを使用したデータ・ソースのインポート 145
データ・ソースの作成 144, 160
データ・ソースの選択 144
データベースからのデータのインポート 146
編集 146
データの取り出し 536
データのバイナリ・ビュー（WinSock）567
データのブックマーク（WinSock）570
データの割り当て方法，パラメータ化 154, 156
データ・バッファ
Tuxedo 仮想ユーザ・スクリプト 1203
WinSock 仮想ユーザ・スクリプト 578
データ・ファイル
Windows Sockets 仮想ユーザ・スクリプト 580
パラメータ化用 130
データ・ファイルのパラメータ
行とカラムの追加 146
データ・ウィザードを使用したデータ・ソースのインポート 145
データ・ソースの作成 144, 160
データ・ソースの選択 144
データベースからのデータのインポート 146
編集 146
データベース仮想ユーザ・スクリプト
LRD 関数の使用法 529
エラー処理 541
概要 523
行情報 536
グリッドの表示 537
作成 519
相関 545
ヒント 1282
リターン・コード 540
データベース記録オプション 525
データベース・クエリ・ウィザード・ダイアログ・ボックス 147
データを送信ステップ
ダイアログ・ボックス（Web）908
定義 808
変更（Web）907
テーブル
パラメータ化用 131
テーブル・アイコン 129
テキスト
画面からのテキストの読み取り（RTE）1168
画面上のテキストの検索（RTE）1168
スナップショットでの比較（Web）974

テキスト・チェック
追加プロパティの定義 890
定義 879
フラグ 886
テキストの暗号化 120
テキストの取得ツール，Citrix 仮想ユーザ・スクリプト 512
テキストの同期化，Citrix 477
テキストの復号 120
テキスト・ビュー（WinSock）566
テスト結果
Web 仮想ユーザ 245
Web サービス仮想ユーザ 400
デバイス名（RTE）1153
デバッグ
Oracle アプリケーション 1277
Web 仮想ユーザ・スクリプト 243
Web 仮想ユーザ・スクリプトでの有効化 225
再生中 230
情報の取得（WAP）867
データベース・アプリケーション 1275
デバッグ機能の有効化 225
デバッグ・レベルの設定 205
デバッグ記録設定（Java）441
デバッグ・メッセージ・ダイアログ・ボックス 115
添付ファイル，WSDL 335
テンプレート
C 言語を使ったプログラミング 1320, 1331
C 言語を使ったプログラミング 1252
Java 仮想ユーザ 605
新規作成 66

と

同期化の失敗，Citrix 494
同期関数
Citrix テキストの生成 477
動作，DHTML 777
同時実行グループ関数 804
同時実行の設定，Web サービス 743
動的ポート 588
トークン置換テスト・パッド・ダイアログ・ボックス 929
トークン，パラメータ化 923
トラップ 982
トラフィック，サーバ 403
トラフィック情報，指定 410
トラフィックの転送 98
トラブルシューティング
2 層データベース 1282
Oracle アプリケーション 1277
Siebel 仮想ユーザ 1288
VuGen 1273
Web 仮想ユーザ・スクリプト 961
トラブルシューティング，.NET 749
トランザクション
Oracle DB のブレークダウンの制限 59
Web 仮想ユーザ 216
Web 仮想ユーザ・スクリプトの変更 914
入れ子 53, 109
ウィザード・ワークフロー 48
エディタ 48
関数 35
自動，Web 仮想ユーザ・スクリプト 216
出力ログ内 227
挿入 107
トランザクション・エディタ 28, 48
トランザクション開始ダイアログ・ボックス 108
トランザクション終了ダイアログ・ボックス 109
トランスポート層の設定 338

な

内部データ，パラメータ化 131, 168
内容タイプのフィルタ・ダイアログ・ボックス 824
内容タイプのフィルタリング（Web）823
内容チェック
エラーの制限 862
設定（Web）870

に

入力スタイル（RTE 仮想ユーザ）1146
入力引数，Web サービス 331
認証，記録時 78
認証の再試行時の思考遅延時間 862

ね

ネットワーク設定 220
ネットワークのバッファ・サイズ（インターネット）862

は

バージョン・コントロール 258
テストのチェック・イン 261
バージョン・コントロール・データベースからのスクリプトのチェック・アウト 259
バージョン・コントロール・データベースへのスクリプトのチェック・イン 261
バージョン・コントロール・データベースへのスクリプトの追加 259
バージョンの履歴 263
バージョンの履歴ダイアログ・ボックスの使用方法 263
バイナリ・コード・データ 894
ハイパーグラフィック・リンク・ステップ, Web 仮想ユーザ 807
ハイパーテキスト・リンク・ステップ
定義 807
変更 899
配列，Web サービス引数 333
パスワード・エンコーダ・ダイアログ・ボックス 121
パスワード，エンコーディング 121
バッファ・ナビゲータ（WinSock）568
バブリング 778
パラメータ
削除 139
スクリプト・ビューでの作成 126
ツリー・ビューでの作成 127
パラメータ・リストを使用した作成 139
変更 138
パラメータ化
COM，.NET，VB 126
Java 127
Tuxedo スクリプト 1202
UTF-8 エンコードされた 1320
XML 131
一意の値を使った更新 155
概要 124
括弧のスタイル 129, 140
既存パラメータの変更 190
既定値を復元 137
グローバル・ディレクトリ 141
シードによる乱数シーケンス 155
制限 125
データ・ファイル 130
データ・ファイルのプロパティ設定 150
テーブル 131
テーブルのプロパティ設定 152
内部データ・タイプの形式 180
内部データの使用 131, 168
名前，パラメータ 127
パラメータ値の更新 181
新規パラメータの作成 126
パラメータ・リスト 138
ファイルおよびテーブルからの値の割り当て 154
プロパティの定義 133
元に戻す（Web）137
ユーザ定義関数 132
ユーザ定義関数の使用 179
パラメータ化オプション 140
パラメータ化で使用する括弧 140
パラメータ化，長い文字列の置換 86
パラメータ形式
削除 181
追加 180
元に戻す 181
パラメータ・タイプ
Date/Time（日時）169
XML 131
一意の数 175
概要 130
仮想ユーザ ID 178
グループ名 171
データ・テーブル 167
データ・ファイル 130, 167
テーブル 131
内部データ 131, 167, 168
反復回数 172
ユーザ定義関数 132, 167, 179
乱数 174
ロード・ジェネレータ名 173
パラメータの既定値の復元 137

パラメータの選択または作成ダイアログ・ボックス 126
パラメータのプロパティ・ダイアログ・ボックス 133
パラメータを元に戻すコマンド 137
ハンドラ 777
ハンドラ・ルーチン，Web サービス 361
反復
　実行環境の設定 199
　反復ごとのパラメータ更新 182
反復回数，パラメータ値 172

ひ

非印字文字 585
比較
　WSDL ファイル 380
　XML ファイル 384
比較オプション，WSDL と XML 381
比較方法 939, 974
比較レポート 382
ビットマップの不一致 494
非同期メッセージ 338
非標準 HTTP アプリケーション 971
表記規則 xxviii
表示実行
　定義 223
　有効化 224
表示タブ，一般オプション 225
標準的なイベント記録設定 771–772
標準の応答
　エミュレートされたサービス 287
標準ログ・オプション 204
標準ログの実行環境の設定 205
開くコマンド 249
ヒント
　Siebel 固有 1288
　データベース関連 1282

ふ

ファイル，スクリプトへの追加 122
バッファのデータのオフセット（WinSock） 581
フィールド区分文字 1148
フィルタ・ダイアログ・ボックス（Web レポート）247
フィルタ・ファイル，.NET 向けの編集 734
フィルタ・マネージャを使った作業 727
フィルタリング
　.NET 仮想ユーザ 719
　サーバ・トラフィック・スクリプト 411
　ダウンロードされたリソース 866
　内容タイプ（Web，ワイヤレス）823
　レポート情報（Web）247
プール，接続 743
フォームを送信ステップ
　ダイアログ・ボックス 904
　定義 808
　変更 903
プッシュ・サポート 1229
ブラウザ
　起動（Web/WinSock）979
　キャッシュ（Web，ワイヤレス）855
　記録オプション（Web）828
　手作業での起動（Web）828
　場所の指定（Web）828
　標準設定のブラウザの使用（Web）828
ブラウザのエミュレーション設定，Web 852
フラグ，テキスト検索 886
プラグマ・モード 1031
プリファレンス，実行環境の設定 857
ブレークポイント 230
ブレークポイント・マネージャ 231
プロキシ・サーバ
　記録オプション（Web）816
　記録オプション（Web/WinSock）980
　実行環境の設定（インターネット）847
プロキシ認証ダイアログ・ボックス 818
プログラミング
　Visual Studio 1251
　仮想ユーザ・アクション 1331
　テンプレートの使用 1252, 1331
ブロック・サイズ，仮想ユーザの値の割り当て 151, 161, 164
プロトコル
　SAPGUI/SAP-Web 1043, 1073
　Web/WinSock 975
プロトコル，サポートされている一覧 64
プロパティ
　AssignToParam（Web）890
　Expect（Web）891
　Frame（Web）890

MatchCase（Web）890
OnFailure（Web）891
Repeat（Web）891
Report（Web）891
Web サービスの取得と設定 362
テキスト・チェック 890
プロパティの定義，テキスト・チェック 880
プロパティ，パラメータ
定義 133
データ・ファイルに対する定義 150
テーブルのための定義 152
文書の更新 xxvii

へ
ページ・ビュー（Web スナップショット）24
ペースの設定 199
ヘッダー
ユーザ定義 821
リスキー 821
ヘッダー・ファイル 41
変換
Web から Java への変換 760
ユーザ定義要求の C 形式への 911
変換，UNIX 上の ASCII 559
変換テーブルの設定 559

ほ
ポートの割り当て設定 94
ホスト接尾辞，フィルタリング 866
ポリシー・ファイル 357

ま
マルチ・アクション 60
マルチ・スレッド 215
マルチ・プロトコル 60

め
メール・サービス・プロトコル
IMAP 1174
MAPI 1176
POP3 1178
SMTP 1179
記録 1172
メソッド，Java 605
メッセージ
関数リスト 38
出力への送信 112
メッセージ署名 353
メモリ管理 865

も
文字エンコーディング 1322
文字セット，RTE 1140
文字列，長い文字列をパラメータで置換 86
モデム速度，実行環境の設定 220
元戻しバッファ，空にする（WinSock）573

ゆ
ユーザ・エージェント・ブラウザのエミュレーション 853
ユーザ定義 Web イベント記録の設定ダイアログ・ボックス 774
ユーザ定義関数のパラメータ 132
ユーザ定義関数，パラメータ化 179
ユーザ定義ステップ
XML 956
定義 808
変更（Web）911
ユーザ定義の仮想ユーザのタイプ
C 仮想ユーザ 595
JavaScript 仮想ユーザ 602
Java 仮想ユーザ 599
VBScript 仮想ユーザ 601
VB 仮想ユーザ 600
ユーザ定義の要求 842
ユーザ定義ヘッダー，Web およびワイヤレス用 821
ユーザ定義要求ダイアログ・ボックス（Web）912
ユーザ・ハンドラ，Web サービス 361

よ
読み取り専用 WinSock バッファ 566

ら
ライセンス情報 8
ライブラリ，スクリプトの作成 217
乱数，パラメータ値 174
ランダム方式でのパラメータの割り当て 155

ランデブー
ランデブー・ダイアログ・ボックス 110
ランデブー・ポイント
Java 仮想ユーザ 612
Web 仮想ユーザ・スクリプトの変更 915
ランデブー・ポイント，挿入 110

り
リソース
LoadRunner Readme ファイル xxv
リソース以外 825
リソース，除外 825
リターン・コード
データベース 540
リンク・ステップのプロパティ・ダイアログ・ボックス 899

る
ルール
相関結果からの作成 937
相関タブからの追加 941
相関内でのテスト 929
相関の詳細 926
相関のための定義 927
ルール，エミュレートされたサービスの設定 288
ルールの作成 937
ルールの追加 941

れ
レポート
Web サービス・スクリプト 400
XML の比較 382
妥当性チェック 377
レポート・ツリー，結果サマリ（Web）245

ろ
ロード・ジェネレータ名，パラメータ値 173
ロード・バランシング，Oracle NCA 1037
ログ
詳細レベルの設定 - PC 204
ログ記録オプションの無効化 203
ログ，実行環境の設定 202
ログのキャッシュ・サイズ 204
ログ表示オプション（Web）225
ログ・メッセージ・ダイアログ・ボックス 114

わ
ワークフロー・ウィザード 43
ワイヤレス仮想ユーザ・スクリプト
インターネット記録オプション 815
開始 1225
記録 1223
プロキシ設定 816
ユーザ定義ヘッダー 821
ワイルドカード，Citrix ウィンドウ名 502
割り当て，仮想ユーザの値
データ・テーブル 154
データ・ファイル 151, 161, 164