PeopleSoft®

EnterpriseOne Xe Virtual User Tool PeopleBook

September 2000

J.D. Edwards World Source Company
7601 Technology Way
Denver, CO 80237

Table of Contents

Special Considerations for Simulated Playback

OneWorld Virtual User Tool Troubleshooting Tips and Techniques

Glossary

Index

OneWorld Virtual User Tool Overview

JD Edwards OneWorld Virtual User Tool is a collection of automated testing tools that you use to simulate on a single workstation the actions of more than one OneWorld user. You use the tool in conjunction with OneWorld Scripting Tool, an automated testing tool that allows you to create scripts that test OneWorld processes.

OneWorld Scripting Tool lets you capture and save results from each script playback session using hooks, in the form of code strategically placed both in OneWorld Scripting Tool and in OneWorld. OneWorld Scripting Tool's data capture and storage capability provides a precise record of a single playback session, including all API calls. OneWorld Virtual User Tool enables you to use the data you capture during OneWorld Scripting Tool playback session to generate scripts that you run to approximate actual stress on a server and on the network.

OneWorld Virtual User Tool:

- Provides a manageable solution for the creation and playback of virtual scripts that test a system's scalability
- Captures and logs data about each script playback session
- Bypasses OneWorld user interface to run virtual scripts
- Tests applications regardless of Object Configuration Manager (OCM) settings and database caching
- Recreates OneWorld's actual operations in a stressed server and network environment
- Supports n-tier testing
- Runs independently of most changes in OneWorld code
- Provides a mechanism for reporting and analyzing test results

In addition to being cost- and resource-efficient, OneWorld Virtual User Tool scripts are flexible because you can run them against any Object Configuration Manager (OCM) mapping, operating system, or database.

OneWorld Virtual User Tool batch mode script testing offers benefits in several other ways:

- Departments can reduce the amount of time required to create scripts
- Companies can reduce the amounts of time, personnel, and equipment invested in testing OneWorld applications

- Performance engineers can determine the scalability of a system running OneWorld
- Tools developers can test and debug jdekrnl and jdenet in an environment that realistically simulates heavy user load
- Applications developers can test OCM configurations and the performance of business functions in a stressed environment
- Business partners can size hardware to meet the needs of customers using OneWorld

The collection of tools that make up OneWorld Virtual User Tool is part of an automated testing tools architecture developed by J.D. Edwards. The following table presents the components of the architecture and summarizes the role of each in the process of creating virtual scripts:

Component of OneWorld Virtual User Tool architecture	Function in OneWorld Virtual User Tool Script Creation
OneWorld Scripting Tool	Automated testing tool used to write scripts that test OneWorld applications by simulating user input.
Hooks in OneWorld Scripting Tool and in OneWorld	Code that captures and records all data generated during the playback of a OneWorld Scripting Tool script.
Event Stream	Record of the continuous series of events captured by OneWorld Scripting Tool during script playback. Resides within OneWorld table F98214.
OneWorld table F98214	OneWorld table used to store the event stream.
Virtual Script Editor	Tool that allows user to import an event stream and to create a OneWorld Virtual User Tool script by applying rules that govern the way the script runs in batch mode.
OneWorld Virtual User Tool script	The combination of events from the event stream and rules automatically inserted by the Virtual Script Editor and manually inserted by the user.
Virtual Script Player	Tool that runs the OneWorld Virtual User Tool script, simulating the work of the OneWorld run-time engine.
vap.log file	Text file that contains any error messages that the Virtual Script Player sends during playback.

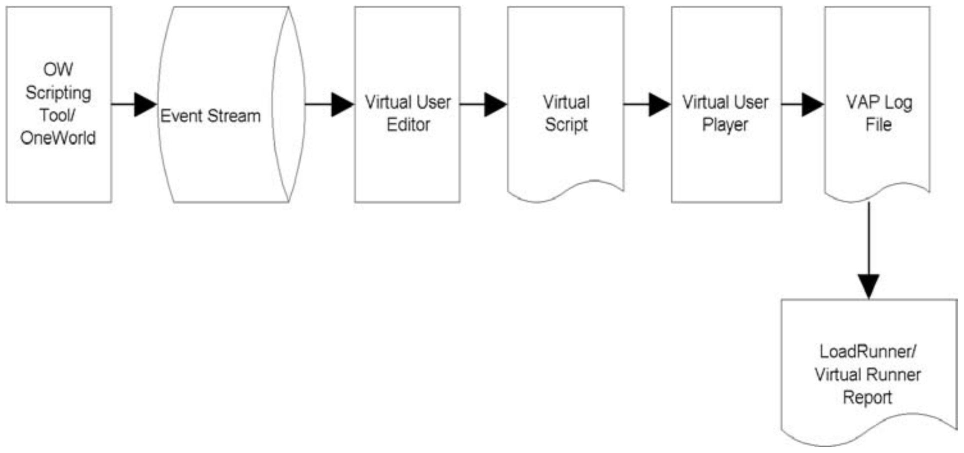
Virtual Runner	Program that allows you to launch multiple OneWorld Virtual User Tool scripts from a single workstation.

Each of these components is discrete. However, each plays a role that is integral to the entire process of creating a virtual script. We can summarize the stages of the script creation process the role of each component, and the way the roles relate as follows:

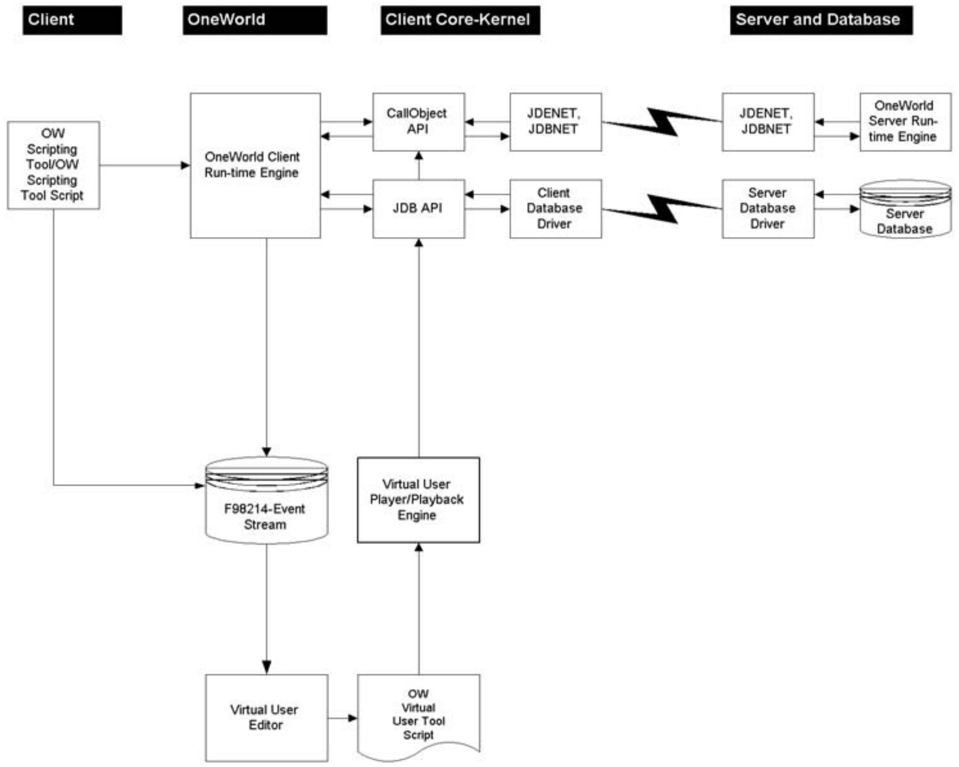
- Data capture: You write a OneWorld Scripting Tool script and play it back. Hooks in OneWorld Scripting Tool and in OneWorld capture the event stream and store this record of events in OneWorld table F98214
- Virtual script creation: You use the Virtual Script Editor to create the virtual script from the event stream.
- Virtual script playback: After you generate the script, you use the Virtual Script Player to play the script. The Virtual Script Player assumes the role of the OneWorld run-time engine. OneWorld Virtual User Tool bypasses OneWorld completely in playing back the virtual script.
- Error message generation: During playback of a virtual script, the Virtual Script Player sends any error messages to the vap.log file text file.
- Multiple script launch and playback analysis: Virtual Runner and Mercury Interactive's LoadRunner load testing tool to launch OneWorld Virtual User Tool scripts and to save and to analyze test results for each simulated user session. You use Virtual Runner to launch simulate multiple users on a single workstation; LoadRunner allows you to launch multiple script playback sessions from more than one workstation.

  Note: The number of simulated users will be determined, in part, by the power of the workstations and the operating system.

Another way to conceptualize the OneWorld Virtual User Tool architecture is to examine the flow of data from the time you write a OneWorld Scripting Tool script to the time you simulate multiple OneWorld sessions on a single workstation with the Virtual Script Player.



We can also view the OneWorld Virtual User Tool architecture as it is positioned within the larger universe of J.D. Edwards' configurable network computing (CNC) architecture. Such a view shows the flow of data that occurs during the playback of a OneWorld Scripting Tool script. Note again that data captured in OneWorld and In OneWorld Scripting Tool during script playback is routed to OneWorld table F98214. The Virtual Script Editor user imports the data and modifies it to produce a OneWorld Virtual User Tool script that the Virtual Script Player runs to simulate OneWorld activity.



In sum, OneWorld Virtual User Tool's design and architecture allow it to meet the challenge of accurately simulating server and network stress imposed by the dynamic interaction of multiple OneWorld users.

This guide discusses the following components of and tasks associated with JD Edwards OneWorld Virtual User Tool:

- ❑ Data Capture for OneWorld Virtual User Tool Scripts
- ❑ OneWorld Virtual User Tool Components
- ❑ Creating OneWorld Virtual User Tool Scripts
- ❑ Running OneWorld Virtual User Tool Scripts
- ❑ Special Considerations for Simulated Playback
- ❑ OneWorld Virtual User Tool Troubleshooting Tips and Techniques

Data Capture for OneWorld Virtual User Tool Scripts

Data Capture for OneWorld Virtual User Tool Scripts

Several APIs enable OneWorld to interact with any database or application server. The APIs communicate with OneWorld's middleware, which serves as the conduit for run-time data flowing from the client workstation to the server and back again. J.D. Edwards' automated testing tools capture this data, which provides you with the raw material to build a virtual script that accurately simulate OneWorld processes.

The following components of J.D. Edwards' automated testing tools architecture work together to capture, record, and store data about OneWorld processes, including the parameters of all API calls and all other OneWorld run-time events:

- OneWorld Scripting Tool, which allows you to write and play back a script to test OneWorld applications and to configure script playback so that OneWorld Scripting Tool captures and saves playback data.
- Hooks, or code, that reside in OneWorld and in OneWorld Scripting Tool that capture and record data generated by the playback of a OneWorld Scripting Tool script.
- Event stream, which is a time-stamped, chronological record of each OneWorld Scripting Tool and OneWorld event that occurs during script playback.
- OneWorld table F98214, which stores the event stream.

The placement of OneWorld code is important to the creation of OneWorld Virtual User Tool scripts. Because this code is positioned at the boundary between the OneWorld run-time engine and the OneWorld middleware, it captures data passing to the JDB and CallObject APIs before the APIs are routed to servers by the OCM. Therefore, you can reuse OneWorld Virtual User Tool scripts regardless of changes in OCM mappings.

Data Capture Components

Creating a virtual script requires that you first capture data from a OneWorld session in which you launch an application, press buttons, enter data to header controls, and so on. The automated testing tool architecture of which OneWorld Scripting Tool is a part enables you to capture the events of a OneWorld session by writing a script, configuring it for event capture, playing it back, and storing its results. You accomplish these tasks using three components:

- OneWorld Scripting Tool, which allows you to write a script, play it back in OneWorld, and save the results of the playback.
- Code that resides in OneWorld and in OneWorld Scripting Tool, which captures and records the data generated by OneWorld Scripting Tool and by OneWorld during script playback.
- OneWorld table F98214, which stores the data generated during script playback as an event stream, which is a continuous record of every OneWorld and OneWorld Scripting Tool event that occurred during script playback.

This chapter discusses each of the three data capture components:

- ❑ OneWorld Scripting Tool
- ❑ OneWorld and OneWorld Scripting Tool code
- ❑ Event stream
- ❑ OneWorld table F98214

OneWorld Scripting Tool

The process of creating a virtual script begins with OneWorld Scripting Tool, which you use to write a script that tests OneWorld processes. Playing back a OneWorld Scripting Tool script simulates OneWorld activities, but only as initiated by one user. However, you can capture the results of script playback, including the processes generated by OneWorld, save the data, and use it to create a virtual script that you run to simulate more than one user.

You perform the following tasks to set up and accomplish data capture using OneWorld Scripting Tool:

- ❑ OneWorld Scripting Tool script creation

- ❑ OneWorld Scripting Tool script playback configuration
- ❑ OneWorld Scripting Tool script playback

OneWorld Scripting Tool Script Creation

To begin the process of capturing data, you first write a OneWorld Scripting Tool script to test OneWorld processes such as launching applications, pressing buttons, entering data to header controls, and so on. For a complete discussion of the features of OneWorld Scripting Tool and using the tool to write scripts, see the *OneWorld Scripting Tool Guide*.

OneWorld Scripting Tool Playback Configuration

You capture data about OneWorld processes by playing back the OneWorld Scripting Tool script, but you must configure playback for data capture. Your configuration choices establish how much data you capture and ensure that OneWorld Scripting Tool saves the data.

You can capture data at one of two levels:

- Level 1 captures data only for initiating API calls that run alone or call other APIs. If you choose this option, you capture data about only these APIs.
- All API calls captures data not only about level 1 API calls, but also about any API calls spawned by a level 1 API.

You also configure script playback to save and display results data after playback.

OneWorld Scripting Tool Script Playback

After you have written a script and configured playback to capture the results, you play back the script. OneWorld Scripting Tool captures the playback data using internal code and code placed in OneWorld.

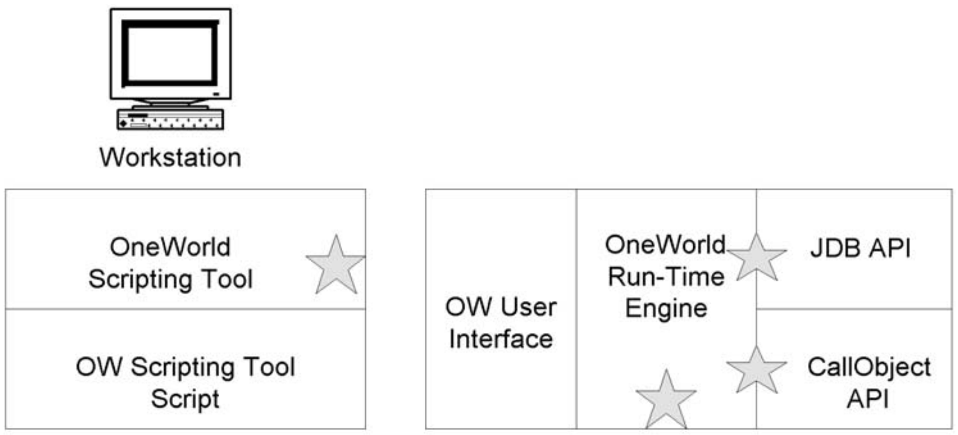
OneWorld and OneWorld Scripting Tool Code

Code is strategically placed in OneWorld Scripting Tool and in 32 JDB functions and one CallObject function in OneWorld to gather and store during OneWorld Scripting Tool script playback.

The placement of OneWorld code provides the following important advantages for creating OneWorld Virtual User Tool scripts:

- Comprehensive data capture, because code is positioned to capture both JDB and JDE CallObject API calls. This means that you capture both database and business function activity.
- Simplified script maintenance, because the code resides in slightly more than 30 JDB and CallObject functions combined, making changes in OneWorld code relatively easy to handle.
- Flexibility in running scripts, because data that you capture can be run independent of platform or OCM mapping considerations.

The stars in this diagram illustrate the placement of OneWorld and OneWorld Scripting Tool code for the capture of data during playback of a OneWorld Scripting Tool script:



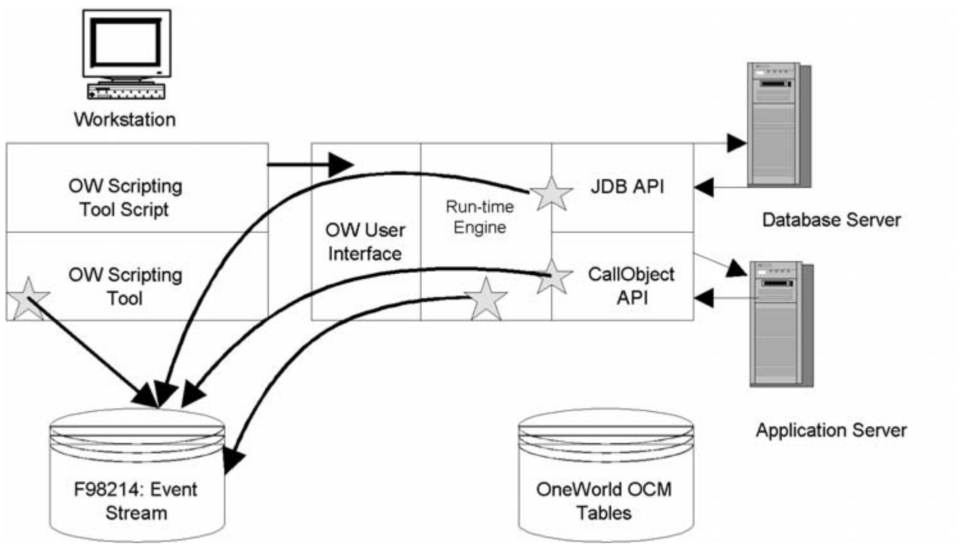
Code placed in OneWorld performs the following functions that lay the groundwork for the creation of a OneWorld Virtual User Tool script:

- Captures parameter data on JDB and CallObject API calls that occur during the playback of a OneWorld Scripting Tool script
- Writes the parameter data to a file-mapping object that OneWorld shares with OneWorld Scripting Tool
- Writes data on event rules, button presses, and event timing to the file-mapping object

Code placed in OneWorld Scripting Tool performs the following functions that lay the groundwork for the creation of a OneWorld Virtual User Tool script:

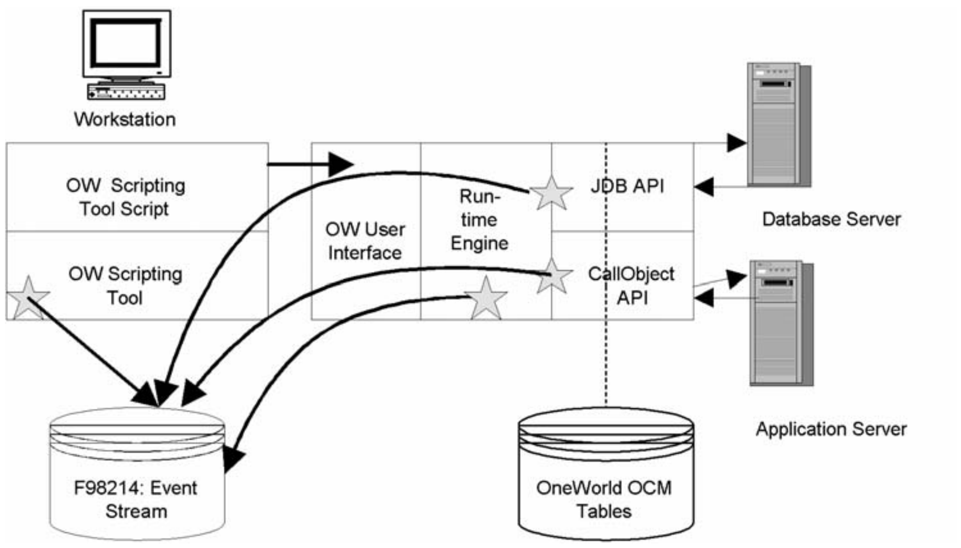
- Writes data on OneWorld Scripting Tool events to the file-mapping object
- Copies the OneWorld and OneWorld Scripting Tool data in the shared file space into a BLOB field in OneWorld database table F98214
- Enables the OneWorld Scripting Tool user to access the OneWorld table F98214

The following diagram illustrates the flow of playback data from the data-capture points to OneWorld table F98214, where the results are stored:



OneWorld Virtual User Tool scripts that you create using the data captured from a OneWorld Scripting Tool script are platform independent and can be run on any operating system with any OCM mappings because OneWorld code captures API call data before it reaches the OCM for mapping.

The dotted line in the following diagram illustrates the positioning of OneWorld data capture code in relation to the OCM:



Event Stream

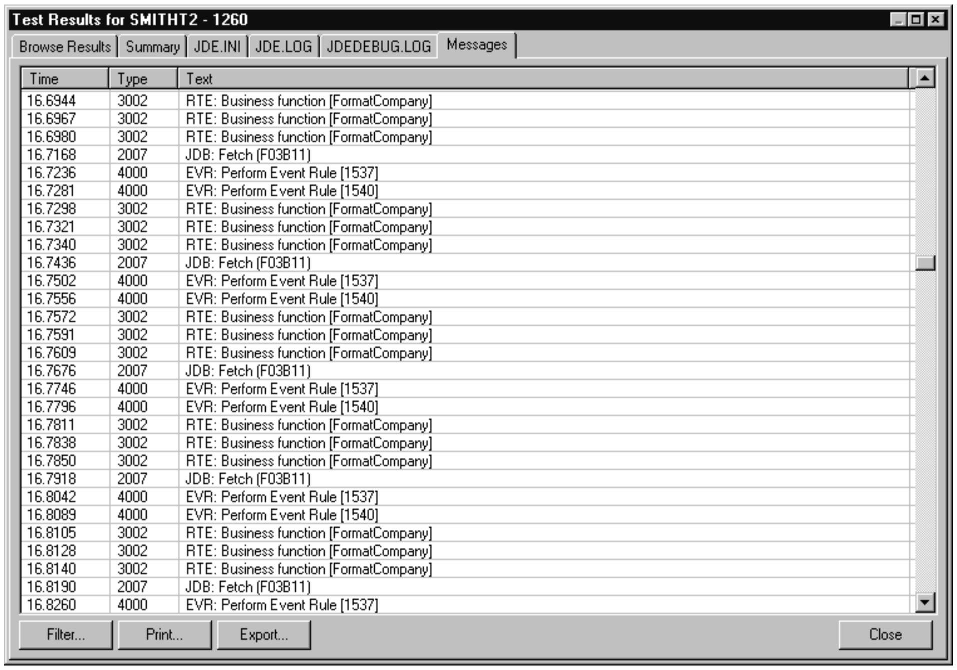
You generate an event stream when you run a OneWorld Scripting Tool script that you have configured to capture playback results. The event stream is a time-stamped, chronological record of every OneWorld Scripting Tool and OneWorld event that occurs during playback, including:

- JDB and CallObject API calls
- Thread identification
- Event rules
- OneWorld error and warning messages
- OneWorld Scripting Tool events confirming that the script and OneWorld are on the same form

OneWorld Table F98214

OneWorld table F98214 is a database repository that contains the results of all OneWorld Scripting Tool playback sessions that you have captured and saved. You can access the event stream for a script playback session through OneWorld Scripting Tool, the Virtual Script Editor, or OneWorld Analyzer Tool.

Using OneWorld Scripting Tool, you can view a summary of every event that occurred during script playback.



Time	Type	Text
16.6944	3002	RTE: Business function [FormatCompany]
16.6967	3002	RTE: Business function [FormatCompany]
16.6980	3002	RTE: Business function [FormatCompany]
16.7168	2007	JDB: Fetch (F03B11)
16.7236	4000	EVR: Perform Event Rule [1537]
16.7281	4000	EVR: Perform Event Rule [1540]
16.7298	3002	RTE: Business function [FormatCompany]
16.7321	3002	RTE: Business function [FormatCompany]
16.7340	3002	RTE: Business function [FormatCompany]
16.7436	2007	JDB: Fetch (F03B11)
16.7502	4000	EVR: Perform Event Rule [1537]
16.7556	4000	EVR: Perform Event Rule [1540]
16.7572	3002	RTE: Business function [FormatCompany]
16.7591	3002	RTE: Business function [FormatCompany]
16.7609	3002	RTE: Business function [FormatCompany]
16.7676	2007	JDB: Fetch (F03B11)
16.7746	4000	EVR: Perform Event Rule [1537]
16.7796	4000	EVR: Perform Event Rule [1540]
16.7811	3002	RTE: Business function [FormatCompany]
16.7838	3002	RTE: Business function [FormatCompany]
16.7850	3002	RTE: Business function [FormatCompany]
16.7918	2007	JDB: Fetch (F03B11)
16.8042	4000	EVR: Perform Event Rule [1537]
16.8089	4000	EVR: Perform Event Rule [1540]
16.8105	3002	RTE: Business function [FormatCompany]
16.8128	3002	RTE: Business function [FormatCompany]
16.8140	3002	RTE: Business function [FormatCompany]
16.8190	2007	JDB: Fetch (F03B11)
16.8260	4000	EVR: Perform Event Rule [1537]

See the *OneWorld Scripting Tool Guide* for a discussion of viewing test results.

From the Virtual Script Editor or from OneWorld Analyzer Tool, you can view each event in more detail. For example, you can select an API call and view the input and output parameter values for the call. If you import an event stream record from the database table to the Virtual Script Editor, you can modify the record so that you can play it as a OneWorld Virtual User Tool script.

For a discussion of using OneWorld Analyzer Tool to view script playback events, see the *OneWorld Analyzer Tool Guide*.

OneWorld Virtual User Tool Components

OneWorld Virtual User Tool Components

You use components that are external to OneWorld Virtual User Tool to capture, record, and store the data generated by a OneWorld Scripting Tool playback session:

- OneWorld Scripting Tool
- Data-capture code in OneWorld Scripting Tool and in OneWorld
- Event stream
- OneWorld table F98214, where the event stream from the playback session is stored

The internal components of OneWorld Virtual User Tool allow you to complete the OneWorld Virtual User Tool scripting process. These components are:

- Virtual Script Editor
- Virtual Script Player
- VSMEditor
- Virtual Runner

You use OneWorld Virtual User Tool's internal components to complete the following tasks:

- Import an event stream into the Virtual Script Editor
- Modify the event stream by adding rules that govern the passing of parameters and looping (repeated JDB Fetch calls to complete a database inquiry)
- Use the Virtual Script Editor to add rules automatically to handle thread identification and hRequest handles
- Generate a OneWorld Virtual User Tool script
- Run a OneWorld Virtual User Tool script on the Virtual Script Player
- Concatenate a series of individual Virtual User Tool scripts into one master script, using the VSMEditor
- Manage script playback, either from a single workstation or from multiple workstations, using Virtual Runner

You can also manage script playback using LoadRunner from Mercury Interactive.

This section discusses the following internal components of OneWorld Virtual User Tool and their respective functions:

- Virtual Script Editor
- Virtual Script Player
- VSMEditor
- Virtual Runner

Virtual Script Editor

The Virtual Script Editor allows you to create and to generate a OneWorld Virtual User Tool script that you can use to simulate the activity of many concurrent OneWorld users. Working with the Virtual Script Editor represents the second step in a three-step process of producing a OneWorld Virtual User Tool script playback session:

- Capturing data generated by OneWorld Scripting Tool script playback and storing the event stream in a results repository.
- Using the Virtual Script Editor to modify an event stream and to generate a OneWorld Virtual User Tool script that contains all the information required by the Virtual Script Player to simulate the activities of the OneWorld run-time engine.
- Playing back a modified event stream, which is the OneWorld Virtual User Tool script, using the Virtual Script Player.

The event stream is a chronological, time-stamped record of every event that occurs during playback of a OneWorld Scripting Tool script, including:

- User input
- Processing performed by the OneWorld run-time engine, such as thread creation
- Event rules; informative messages
- API calls to the OneWorld middleware

OneWorld Scripting Tool performs no editing during the process. The event stream represents a record of events that have already taken place. You cannot edit it by adding, deleting, or reordering data. To change it, you must generate a new one by modifying an existing OneWorld Scripting Tool script, or by creating a new script, and then replaying it.

Using the Virtual Script Editor, you can:

- View the titles of all the scripts whose results you stored in the F98214 table
- Import an event stream
- View an event stream as a single, continuous record
- View the timing of events by category, represented in a horizontal bar graph

- Choose an individual API call and view the input values sent to the server and the output values returned to the client workstation
- Create links between parameters of API calls so that parameter values can be passed between calls during virtual script playback
- Identify and designate loops so that the virtual script can handle repetitive processing tasks, such as database retrieval

The Virtual Script Editor helps you to address problems that you face in trying to create a script that you can run to simulate activities in a dynamic client/server system. These problems include:

- Identifying API parameters that require dynamic values
- Providing a way to pass values dynamically between API call parameters to avoid data conflict and record contention
- Making the values of hRequest handle parameter values dynamic to simulate concurrent OneWorld user activity
- Synchronizing timing between events during script playback to keep processing running regardless of network stress placed on the server
- Synchronizing timing between data-dependent APIs in threads running asynchronously to avoid one API starting before another has finished processing
- Identifying repetitive processing tasks, such as database inquiry, so that the Virtual Script Player can efficiently simulate the work of OneWorld

The Virtual Script Editor handles the following virtual script creation tasks automatically:

- Linking values of parameters in separate API calls so that values can be passed, provided that the calls meet certain criteria
- Storing as variables the values of hRequest handles
- Storing identification of thread IDs
- Storing information about time gaps between events in a single thread and between interthread-dependent events

You manually handle the following virtual script creation tasks:

- Linking values of parameters in separate API calls that do not meet all the criteria required for automatic value linking
- Identifying as loops repetitive processes such as database inquiry

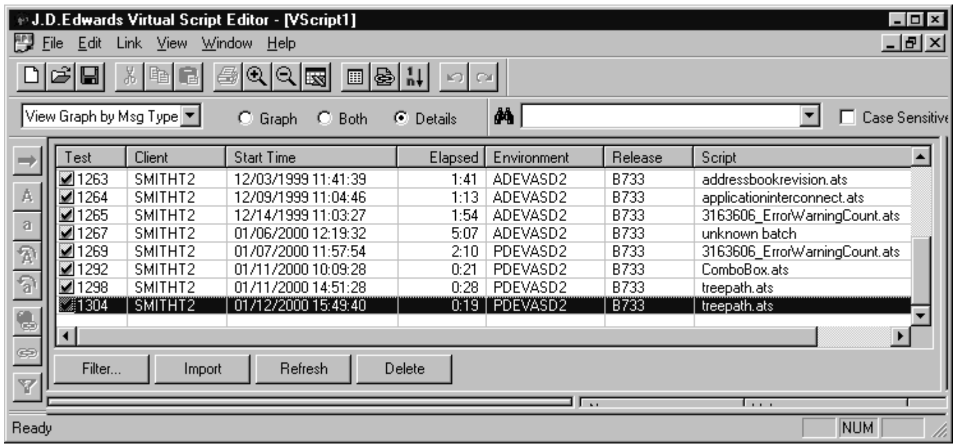
After you have completed these manual tasks, you use the Virtual Script Editor to generate the virtual script. The Virtual Script Player receives from the Virtual Script Editor all of the information necessary to run the virtual script.

This chapter discusses the components of the Virtual Script Editor form and the primary tasks the Virtual Script Editor handles to create a virtual script:

- ❑ Script list pane
- ❑ Event pane
- ❑ Event graph
- ❑ Parameter detail pane
- ❑ Source and target parameter identification
- ❑ Parameter value linking
- ❑ Looping identification
- ❑ hRequest handle value linking
- ❑ Thread identification
- ❑ Timing interval maintenance
- ❑ OneWorld Virtual User Tool script generation

Script List Pane

The script list pane in the Virtual Script Editor form displays in chronological order the OneWorld Scripting Tool script playback results that you saved.



Test	Client	Start Time	Elapsed	Environment	Release	Script
✓ 1263	SMITHT2	12/03/1999 11:41:39	1:41	ADEVASD2	B733	addressbookrevision.ats
✓ 1264	SMITHT2	12/09/1999 11:04:46	1:13	ADEVASD2	B733	applicationinterconnect.ats
✓ 1265	SMITHT2	12/14/1999 11:03:27	1:54	ADEVASD2	B733	3163606_ErrorWarningCount.ats
✓ 1267	SMITHT2	01/06/2000 12:19:32	5:07	ADEVASD2	B733	unknown batch
✓ 1269	SMITHT2	01/07/2000 11:57:54	2:10	PDEVASD2	B733	3163606_ErrorWarningCount.ats
✓ 1292	SMITHT2	01/11/2000 10:09:28	0:21	PDEVASD2	B733	ComboBox.ats
✓ 1298	SMITHT2	01/11/2000 14:51:28	0:28	PDEVASD2	B733	treepath.ats
✓ 1304	SMITHT2	01/12/2000 15:49:40	0:19	PDEVASD2	B733	treepath.ats

The script list pane displays script result information in the following columns:

- Test, which contains the database ID number assigned to each OneWorld Scripting Tool script playback session.

- Client, which contains the ID of the workstation on which you ran the test.
- Start Time, which contains the date and time at which you ran the test.
- Elapsed: the time it took the test to run, either to a successful conclusion, failure, or cancellation.
- Environment, which contains the OneWorld environment against which you ran the test.
- Release, which contains the OneWorld release, such as B73.3, against which you ran the test.
- Script, which contains the name that you assigned to the test.
- Status, which contains the result of the test, either success, failure or cancellation.

After you select a script, you choose from four buttons to manipulate the form view:

Virtual Script Editor Form Button	**Purpose**
Filter	Allows you to remove OneWorld Scripting Tool script playback results that you do not want, using criteria in the Filter form.
Import	Imports into the Virtual Script Editor the event stream from a test result that you choose.
Refresh	Refreshes the script list pane from the database.
Delete	Removes one or more tests from the database.

Event Pane

You click the Import button in the Virtual Script Editor form to import the results from a OneWorld Scripting Tool script playback session. The Virtual Script Editor populates the event pane with the event pane. The event stream contains a time-stamping of each event. Therefore, you can review OneWorld Scripting Tool events or API calls during playback that might have taken an unusual amount of time to run.

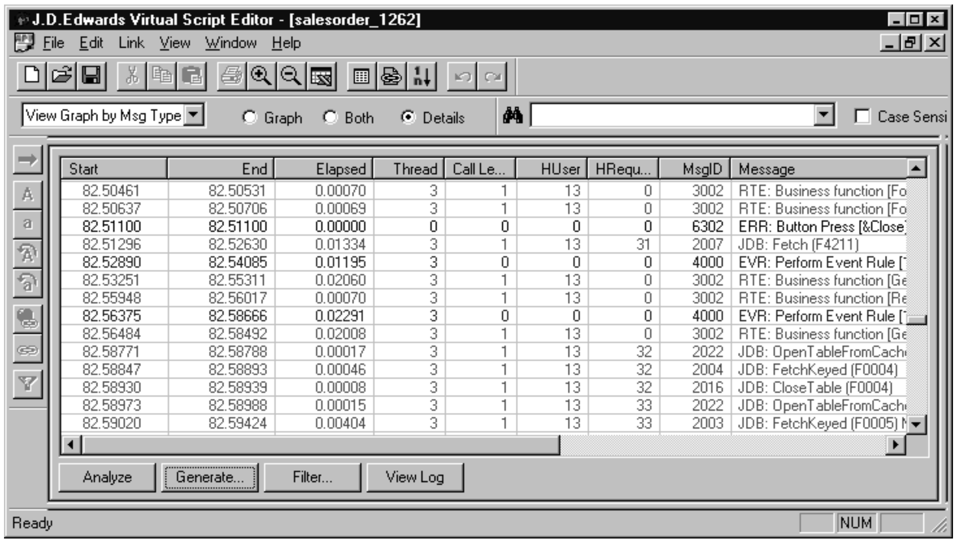
You use the event pane to view data about the following kind of playback events:

- CallObject APIs

- JDB APIs
- OneWorld Scripting Tool events
- Event rules
- Informative messages, including errors and warnings from OneWorld
- Thread creation

The event pane also contains the following columnar information about each event:

- Timing information, such as the start, end, and elapsed time of an event
- Thread identification
- Huser handle identification
- HRequest handle identification
- Call level
- Message entry identifying the event
- Message information about an event, such as JDB call to open a table from memory cache


J.D.Edwards Virtual Script Editor - [salesorder_1262]

View Graph by Msg Type   ( ) Graph   ( ) Both   (●) Details   ☐ Case Sensi

Start	End	Elapsed	Thread	Call Le...	HUser	HRequ...	MsgID	Message
82.50461	82.50531	0.00070	3	1	13	0	3002	RTE: Business function [Fo
82.50637	82.50706	0.00069	3	1	13	0	3002	RTE: Business function [Fo
82.51100	82.51100	0.00000	0	0	0	0	6302	ERR: Button Press [&Close]
82.51296	82.52630	0.01334	3	1	13	31	2007	JDB: Fetch (F4211)
82.52890	82.54085	0.01195	3	0	0	0	4000	EVR: Perform Event Rule [
82.53251	82.55311	0.02060	3	1	13	0	3002	RTE: Business function [Ge
82.55948	82.56017	0.00070	3	1	13	0	3002	RTE: Business function [Re
82.56375	82.58666	0.02291	3	0	0	0	4000	EVR: Perform Event Rule [
82.56484	82.58492	0.02008	3	1	13	0	3002	RTE: Business function [Ge
82.58771	82.58788	0.00017	3	1	13	32	2022	JDB: OpenTableFromCach
82.58847	82.58893	0.00046	3	1	13	32	2004	JDB: FetchKeyed (F0004)
82.58930	82.58939	0.00008	3	1	13	32	2016	JDB: CloseTable (F0004)
82.58973	82.58988	0.00015	3	1	13	33	2022	JDB: OpenTableFromCach
82.59020	82.59424	0.00404	3	1	13	33	2003	JDB: FetchKeyed (F0005) N

The message entry for each event includes an abbreviation that identifies the type of event that occurred. The following table summarizes the abbreviations and the type of event that each represents:

Abbreviation in Message Column of Event Stream	Type of Script Event
JDB	Database API call

RTE	CallObject API call
EVR	Event rule
LOG	OneWorld warning message
ERR	OneWorld error message
MSG	OneWorld Scripting Tool message
AUT	Action in OneWorld Scripting Tool (typing to control, for example)
THR	Thread action

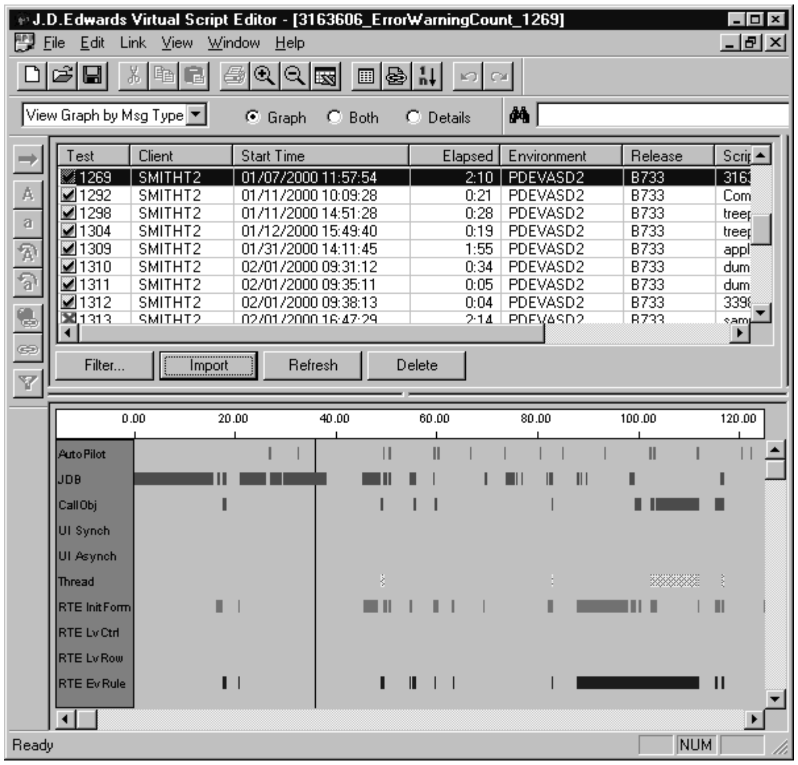
You use the following buttons to change the view in the event pane:

- Generate: allows you to generate a OneWorld Virtual User Tool script. Click this button only after you have completed editing the event stream in the Virtual Script Editor. For details on using the Virtual Script Editor, see *Editing the Virtual Script.*
- Filter: allows you to remove unwanted events from the list by applying criteria found in the Filter form.
- View Log: allows you to look at the log produced when you generate the OneWorld Virtual User Tool script. The log includes the number of lines in the script and the number of errors, if any. For more details, see *Generating the OneWorld Virtual User Tool Script.*

Event Graph

You can also view playback events by category in a horizontal bar graph by choosing the Graph option in the Virtual Script Editor. While the event stream pane presents the events of a OneWorld Scripting Tool script playback vertically, in a single chronological stream, the event graph presents the events horizontally across a timeline.

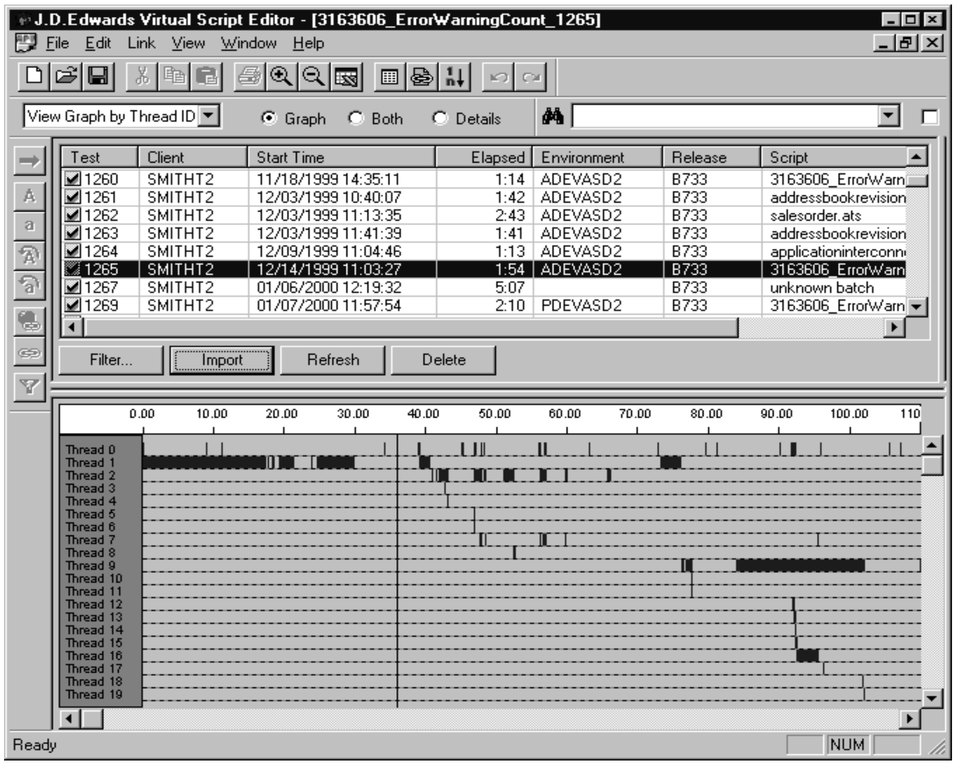
You can break the chronology down by message type, such as JDB API calls or event rules



View Graph by Msg Type [✓ Graph] Both Details

Test	Client	Start Time	Elapsed	Environment	Release	Scrip
1269	SMITHT2	01/07/2000 11:57:54	2:10	PDEVASD2	B733	316
1292	SMITHT2	01/11/2000 10:09:28	0:21	PDEVASD2	B733	Com
1298	SMITHT2	01/11/2000 14:51:28	0:28	PDEVASD2	B733	treep
1304	SMITHT2	01/12/2000 15:49:40	0:19	PDEVASD2	B733	treep
1309	SMITHT2	01/31/2000 14:11:45	1:55	PDEVASD2	B733	appl
1310	SMITHT2	02/01/2000 09:31:12	0:34	PDEVASD2	B733	dum
1311	SMITHT2	02/01/2000 09:35:11	0:05	PDEVASD2	B733	dum
1312	SMITHT2	02/01/2000 09:38:13	0:04	PDEVASD2	B733	339
1313	SMITHT2	02/01/2000 16:47:29	2:14	PDEVASD2	B733	sam

You can also break the chronology down by thread ID.



View Graph by Thread ID [✓ Graph] Both Details

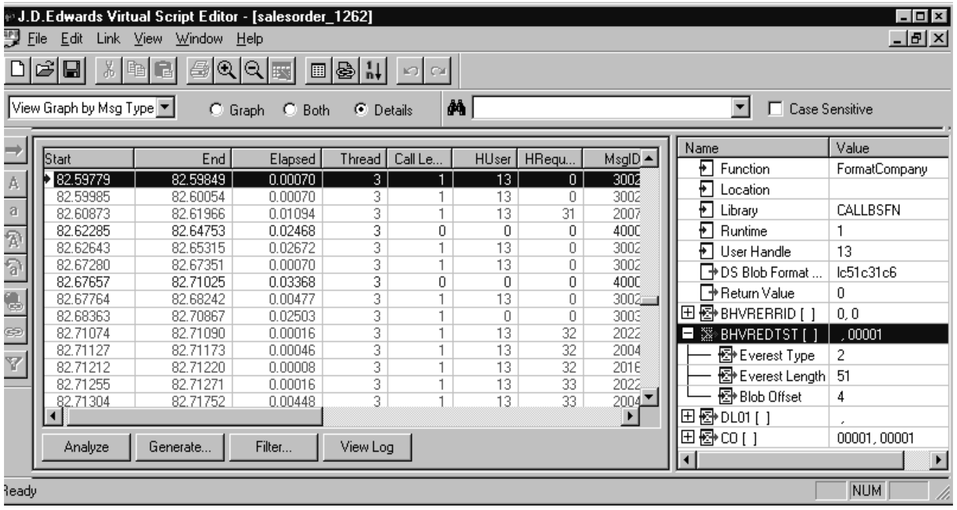
Test	Client	Start Time	Elapsed	Environment	Release	Script
1260	SMITHT2	11/18/1999 14:35:11	1:14	ADEVASD2	B733	3163606_ErrorWarn
1261	SMITHT2	12/03/1999 10:40:07	1:42	ADEVASD2	B733	addressbookrevision
1262	SMITHT2	12/03/1999 11:13:35	2:43	ADEVASD2	B733	salesorder.ats
1263	SMITHT2	12/03/1999 11:41:39	1:41	ADEVASD2	B733	addressbookrevision
1264	SMITHT2	12/09/1999 11:04:46	1:13	ADEVASD2	B733	applicationinterconn
1265	SMITHT2	12/14/1999 11:03:27	1:54	ADEVASD2	B733	3163606_ErrorWarn
1267	SMITHT2	01/06/2000 12:19:32	5:07		B733	unknown batch
1269	SMITHT2	01/07/2000 11:57:54	2:10	PDEVASD2	B733	3163606_ErrorWarn

The event graph provides you with another detailed snapshot of activity that occurred during OneWorld Scripting Tool script playback. You can pinpoint events of unusual duration, which can be helpful in debugging applications, analyzing network activity, or rewriting and rerunning the original OneWorld Scripting Tool script.

Parameter Detail Pane

You can view the parameters that make up an API call by clicking an API call event line in the event pane. The pane that appears shows the name of each parameter in the call and its value, if any. For example, the detail pane might display the value of the user handle parameter that a JDB call passes to the database.

This detail pane provides a complete snapshot of each API call at a given point in time. For example, the pane shows arrows that indicate the flow of data that occurred during the call. An arrow on the left side of the box next to the name of a parameter indicates that the call passed the value from the client workstation to the JDENET or database driver. An arrow on the right side of the box indicates that the call returned data from the server. In some cases, a box contains both arrows, indicating that data flowed in both directions.


J.D.Edwards Virtual Script Editor - [salesorder_1262]
File Edit Link View Window Help
Analyze
Generate...
Filter...
View Log
Ready
NUM

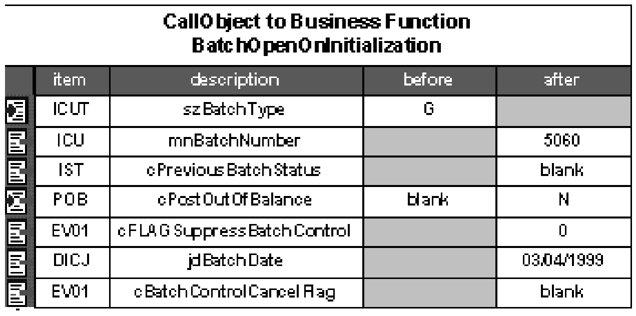
View Graph by Msg Type; ( ) Graph ( ) Both (●) Details; ☐ Case Sensitive

Start	End	Elapsed	Thread	Call Le...	HUser	HRequ...	MsgID
82.59779	82.59849	0.00070	3	1	13	0	3002
82.59985	82.60054	0.00070	3	1	13	0	3002
82.60873	82.61966	0.01094	3	1	13	31	2007
82.62285	82.64753	0.02468	3	0	0	0	4000
82.62643	82.65315	0.02672	3	1	13	0	3002
82.67280	82.67351	0.00070	3	1	13	0	3002
82.67657	82.71025	0.03368	3	0	0	0	4000
82.67764	82.68242	0.00477	3	1	13	0	3002
82.68363	82.70867	0.02503	3	1	0	0	3003
82.71074	82.71090	0.00016	3	1	13	32	2022
82.71127	82.71173	0.00046	3	1	13	32	2004
82.71212	82.71220	0.00008	3	1	13	32	2016
82.71255	82.71271	0.00016	3	1	13	33	2022
82.71304	82.71752	0.00448	3	1	13	33	2004

Name	Value
Function	FormatCompany
Location	
Library	CALLBSFN
Runtime	1
User Handle	13
DS Blob Format ...	lc51c31c6
Return Value	0
⊞ BHVRERRID [ ]	0, 0
⊟ BHVREDTST [ ]	, 00001
Everest Type	2
Everest Length	51
Blob Offset	4
⊞ DL01 [ ]	,
⊞ CO [ ]	00001, 00001

The parameter detail pane offers a before-and-after snapshot of script playback: before playback, parameters for a CallObject API such as BatchOpenOnInitialization contain no batch number of batch date parameter values.

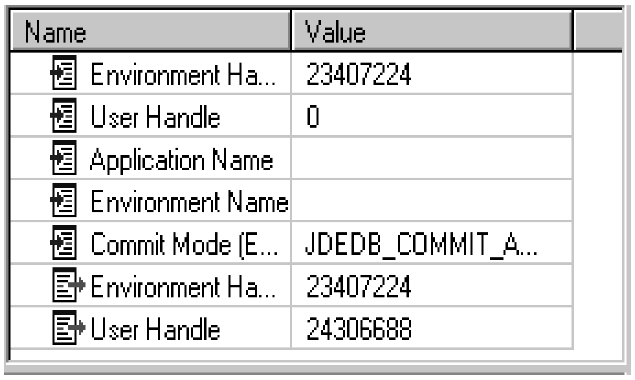
After playback, these parameters contain returned values.



CallObject to Business Function
BatchOpenOnInitialization

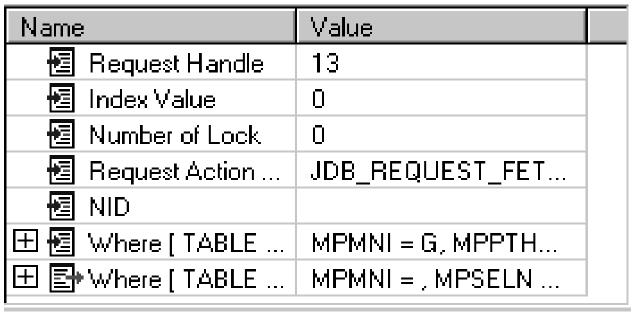
item	description	before	after
ICUT	szBatchType	G	
ICU	mnBatchNumber		5060
IST	cPreviousBatchStatus		blank
POB	cPostOutOfBalance	blank	N
EV01	cFLAGSuppressBatchControl		0
DICJ	jdBatchDate		03/04/1999
EV01	cBatchControlCancelFlag		blank

The parameter detail pane also displays the parameters of API calls that pass an environment handle to the database.



Name	Value
Environment Ha...	23407224
User Handle	0
Application Name	
Environment Name	
Commit Mode (E...	JDEDB_COMMIT_A...
Environment Ha...	23407224
User Handle	24306688

Finally, many API calls contain a request handle that points to a particular place in memory that the run-time engine has allocated for the call. The parameter for the request handle appears in the parameter detail pane if the API call used a request handle.



Name	Value
Request Handle	13
Index Value	0
Number of Lock	0
Request Action ...	JDB_REQUEST_FET...
NID	
⊞ Where [TABLE ...	MPMNI = G, MPPTH...
⊞ Where [TABLE ...	MPMNI = , MPSELN ...

The ability of the OneWorld Scripting Tool and OneWorld hooks to capture data at this level of detail is critically important to OneWorld Virtual User Tool because the goal of OneWorld Virtual User Tool is to simulate as closely as possible the actual activities of OneWorld. If the OneWorld Virtual User Tool script does not have the complete parameters of an API call, it cannot accurately model the activities of OneWorld and its interaction with the client workstation, the database server, and the application server.

See Also

- *Call Level*

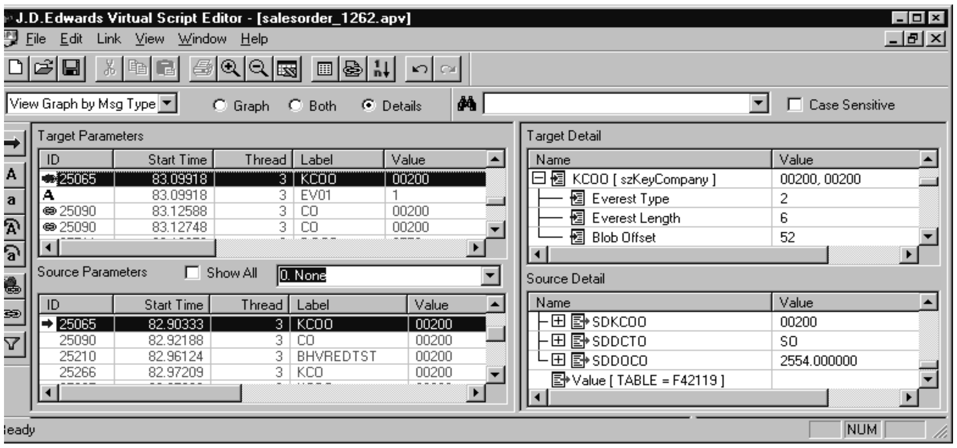
Source and Target Parameter Identification

The Virtual Script Editor provides detailed information about API calls in the event stream when you click the Link Parameters button in the toolbar. The Virtual Script Editor identifies the API calls made during script playback as source parameters or target parameters. A source parameter contains a value that OneWorld passes to a parameter in another API call. The parameter receiving the value is the target parameter.

Information about the source and target parameters appears in separate panes of the Virtual Script Editor form. Each of the panes contains the following information about API calls made during script playback:

Column Heading in Target Parameters Pane and in Source Parameters Pane	Information Displayed
ID	Displays in the source parameter pane a value that identifies that parameter. If you value-link the source parameter to a target parameter, the source parameter ID value appears next to the target parameter, along with a chain–link symbol indicating that you have linked the parameters.
Start Time	Specifies the time the event occurred during playback.
Thread	Identifies the thread generated by the OneWorld run–time engine in which the event occurred.
Label	Identifies the data dictionary alias of the parameter.
Value	Shows the value of the parameter contained within the JDB or CallObject API call.
Comment	Contains the variable name of a business function parameter and the type of data that it contains.
Event	Identifies the specific JDB or CallObject API called or OneWorld Scripting Tool event in which a value was entered.

To see the complete set of parameters for an API call that occurred during OneWorld Scripting Tool playback, you click an item in either pane. The Virtual Script Editor displays the parameter names and values for the selected call. Arrows again indicate the direction of the flow of data.


J.D.Edwards Virtual Script Editor - [salesorder_1262.apv]
File Edit Link View Window Help
View Graph by Msg Type
Graph
Both
Details
Case Sensitive

Target Parameters

ID	Start Time	Thread	Label	Value
25065	83.09918	3	KCOO	00200
A	83.09918	3	EV01	1
25090	83.12588	3	CO	00200
25090	83.12748	3	CO	00200

Source Parameters ☐ Show All 0. None

ID	Start Time	Thread	Label	Value
25065	82.90333	3	KCOO	00200
25090	82.92188	3	CO	00200
25210	82.96124	3	BHVREDTST	00200
25266	82.97209	3	KCO	00200

Target Detail

Name	Value
KCOO [szKeyCompany]	00200, 00200
Everest Type	2
Everest Length	6
Blob Offset	52

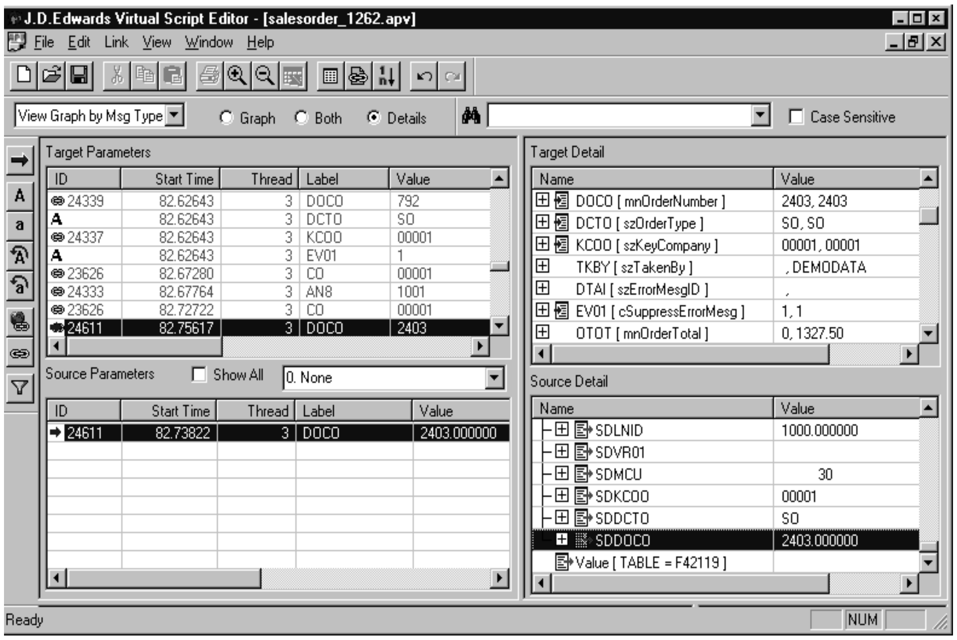
Source Detail

Name	Value
SDKCOO	00200
SDDCTO	SO
SDDOCO	2554.000000
Value [TABLE = F42119]	

Ready NUM

The detail panes provide a snapshot of the API calls that the applications generated during OneWorld Scripting Tool playback. You can examine the parameter values and the flow of the data to help to determine, for example, a parameter value used in one API call that OneWorld passed to another API call later in the script.

To find the parameter of an API call in an event line, you might have to click a node in the detail pane to expand a tree. For example, for a JDB call, find the Value node in the detail pane and expand the tree to expose all the column parameters in the database table. You can then search through the column parameters for the source parameter you are looking for.


J.D.Edwards Virtual Script Editor - [salesorder_1262.apv]
File Edit Link View Window Help
View Graph by Msg Type
Graph
Both
Details
Case Sensitive

Target Parameters

ID	Start Time	Thread	Label	Value
24339	82.62643	3	DOCO	792
A	82.62643	3	DCTO	SO
24337	82.62643	3	KCOO	00001
A	82.62643	3	EV01	1
23626	82.67280	3	CO	00001
24333	82.67764	3	AN8	1001
23626	82.72722	3	CO	00001
24611	82.75617	3	DOCO	2403

Source Parameters ☐ Show All 0. None

ID	Start Time	Thread	Label	Value
24611	82.73822	3	DOCO	2403.000000

Target Detail

Name	Value
DOCO [mnOrderNumber]	2403, 2403
DCTO [szOrderType]	SO, SO
KCOO [szKeyCompany]	00001, 00001
TKBY [szTakenBy]	, DEMODATA
DTAI [szErrorMesgID]	,
EV01 [cSuppressErrorMesg]	1, 1
OTOT [mnOrderTotal]	0, 1327.50

Source Detail

Name	Value
SDLNID	1000.000000
SDVR01	
SDMCU	30
SDKCOO	00001
SDDCTO	SO
SDDOCO	2403.000000
Value [TABLE = F42119]	

Ready NUM

Parameter Value Linking

After you import an event stream into the Virtual Script Editor, you are ready to create the virtual script, using the Target Parameters and Source Parameters panes. Using these panes, you complete the task of value linking. Value linking ensures that the virtual script can pass parameter values from one API to another. You identify a value-containing parameter in a source parameter API call and link the value to a target parameter in another API call. This process ensures the passing of a parameter value from one API to another API that requires the value.

In addition, the values contained in many API call parameters must be dynamic. For example, each time a user performs voucher entry, OneWorld creates a new batch number, a function that is essential to prevent the creation of duplicate keys. Value linking ensures that the Virtual Script Player can simulate this function. When you link the parameter value of two API calls, the Virtual Script Editor stores the value as a variable, and the value changes each time you run a virtual script.

For instance, a script might call the business function BatchOpenOnInitialization. For the parameter ICU, which is the batch number, suppose the API returns the value 5056. In turn, the script might call the business function BeginDoc, which uses the value 5056 as an input to the ICU parameter. To simulate multiple script playback, the value 5056 must change, to reflect the new batch numbers returned each time people using the system make these API calls. So long as you have linked the parameters, the batch number parameter value will change each time you run a virtual script.

In essence, value linking simulates the application logic that is used to run OneWorld operations. It codifies the relationship between one API call and another. When you run the virtual script, the Virtual Script Editor passes to the Virtual Script Player the ID number of the source parameter that you link to the target parameter. The Virtual Script Player uses this information to pass parameter values between API calls.

Several types of data necessary to run a virtual script are candidates for value linking:

- The client host name, which could change anytime a script is played back.
- Next numbers, which must change each time a script is run, in order to avoid producing duplicate data that would break the script.
- Valid values lists used in OneWorld Scripting Tool scripts, which must be designated as such in a OneWorld Virtual User Tool script so that during run time the Virtual Script Player draws new values from the list rather than using the same value over and over again.

This topic discusses:

❏ Automatic value linking of API call parameters

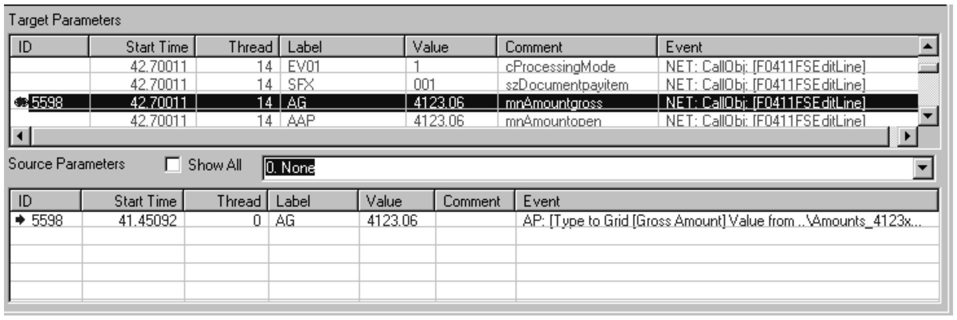
❑ Manual value linking of API call parameters

Automatic Value Linking of API Call Parameters

When you click the Virtual Script Editor's Link Parameters button, the tool automatically links some source parameters of API calls to certain target parameters of other API calls. The Virtual Script Editor accomplishes this automatic linking according to a set of rules. The following rules govern automatic linking of API calls:

- Data must have been entered in OneWorld Scripting Tool.
- The value of the target parameter must exactly match the value of the source parameter.
- The data dictionary ID of the target parameter must exactly match the data dictionary ID of the source parameter.

The Virtual Script Editor finds those parameters that meet each of these conditions and automatically links them.



Target Parameters

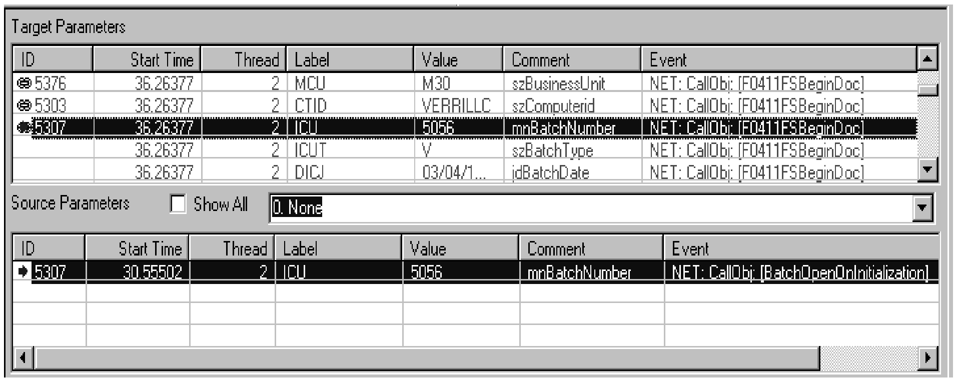
ID	Start Time	Thread	Label	Value	Comment	Event
	42.70011	14	EV01	1	cProcessingMode	NET: CallObj: [F0411FSEditLine]
	42.70011	14	SFX	001	szDocumentpayitem	NET: CallObj: [F0411FSEditLine]
5598	42.70011	14	AG	4123.06	mnAmountgross	NET: CallObj: [F0411FSEditLine]
	42.70011	14	AAP	4123.06	mnAmountopen	NET: CallObj: [F0411FSEditLine]

Source Parameters ☐ Show All 0. None

ID	Start Time	Thread	Label	Value	Comment	Event
5598	41.45092	0	AG	4123.06		AP: [Type to Grid [Gross Amount] Value from ..\Amounts_4123x...

Manual Value Linking of API Call Parameters

If the source and target parameter do not meet all three of these criteria, you must manually link them by clicking a parameter in each pane, and then clicking the Link Parameters button in the Virtual Script Editor tool bar.



Target Parameters

ID	Start Time	Thread	Label	Value	Comment	Event
5376	36.26377	2	MCU	M30	szBusinessUnit	NET: CallObj: [F0411FSBeginDoc]
5303	36.26377	2	CTID	VERRILLC	szComputerid	NET: CallObj: [F0411FSBeginDoc]
5307	36.26377	2	ICU	5056	mnBatchNumber	NET: CallObj: [F0411FSBeginDoc]
	36.26377	2	ICUT	V	szBatchType	NET: CallObj: [F0411FSBeginDoc]
	36.26377	2	DICJ	03/04/1...	jdBatchDate	NET: CallObj: [F0411FSBeginDoc]

Source Parameters ☐ Show All 0. None

ID	Start Time	Thread	Label	Value	Comment	Event
5307	30.55502	2	ICU	5056	mnBatchNumber	NET: CallObj: [BatchOpenOnInitialization]

In deciding the target parameter value to link to a source parameter value, you:

- Match data dictionary aliases
- Match values of the parameters
- Choose, in general, the event in the Source Parameter pane whose start time most closely matches the start time of the event in the Target Parameters pane

You do not have to link the values of source and target parameters when:

- APIs do not contain data dictionary items
- API call returns a zero or null value for the source parameter that might be value-linked to a target
- The data flow of the source parameter is indicated as bi-directional, but the input value and the return values are the same

You code as literal the values of any parameters that meet at least one of these criteria by clicking the Mark Literal button in the Virtual Script Editor form.

The content of the OneWorld Scripting Tool script also plays an important role in the your decisions on value linking. If the OneWorld Scripting Tool script you write contains a literal value that the script writes to a OneWorld grid column or header control, you cannot make that literal value dynamic by linking. The Virtual Script Player will be forced to use that literal value repeatedly during OneWorld Virtual User Tool script playback.

Because you can not make literal values dynamic, avoid using them often in a OneWorld Virtual User Tool script, The entry of the same value to a grid column or header control by multiple users does not accurately simulate the way people use OneWorld. To set up a more realistic scenario, when you write the OneWorld Scripting Tool script, create valid values lists containing more than one value. During OneWorld Virtual User Tool script playback, the Virtual Script Player goes to the .atd directory on your hard drive to retrieve the list's values, and then cycles through them, entering a different value in each simulated playback session until it reaches the end of the list, when it returns to the top of the list and repeats the cycle.

Looping Identification

OneWorld Virtual User Tool requires a method to handle repetitive processing, such as that which occurs when you click the Find button in a OneWorld Find/Browse form to perform a database inquiry. In this situation, a JDB Fetch call might return any number of values from the database. Looping rules give you a way to identify these repetitive retrievals. Having identified them all in a single step, you can more easily link the values of source and target parameters.

You can specify precisely the number of times you want the API to return to the database, fetching and retrieving values to be used as parameters in the script. This allows you to more accurately simulate the load placed on OneWorld. You can increase or decrease this number as you see fit, but the actual number of matches that the API returns based on the inquiry command you write in the OneWorld Scripting Tool script will likely determine the number of loops that you specify.

This topic discusses the following components of dynamic looping:

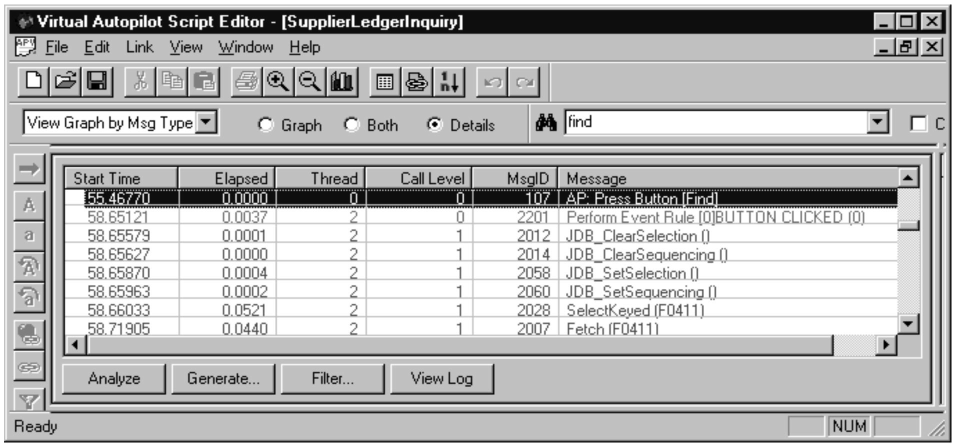
- ❑ Dynamic loop creation
- ❑ Dynamic loop designation
- ❑ Dynamic loop editing

Dynamic Loop Creation

You create a dynamic loop in the OneWorld Scripting Tool script by writing a command to press the Find button in a OneWorld Find/Browse form. This command triggers a string of JDB API calls in OneWorld, culminating in a Fetch call.

OneWorld Scripting Tool and OneWorld code records all of these events during playback. However, there might easily be thousands of events in a OneWorld Scripting Tool script. The Virtual Script Editor offers an easy way to locate the OneWorld Scripting Tool press Find button event and the repetitive processing that occurred as a result of the event. You type the word Find in the locator space in the Virtual Script Editor form. The Virtual Script Editor highlights the first line in the event stream pane that contains the word Find. You can then easily scroll down the pane to the Fetch call.

Following the Fetch call, you can scroll through the event stream to pinpoint the series of calls that resulted from the Fetch. This series of calls constitutes the loop.


Virtual Autopilot Script Editor - [SupplierLedgerInquiry]
File Edit Link View Window Help
View Graph by Msg Type
Graph
Both
Details
Analyze
Generate...
Filter...
View Log
Ready
NUM

View Graph by Msg Type: ( ) Graph ( ) Both (●) Details

find

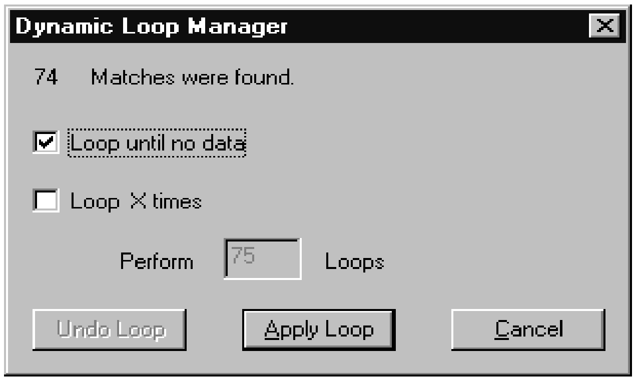
Start Time	Elapsed	Thread	Call Level	MsgID	Message
55.46770	0.0000	0	0	107	AP: Press Button [Find]
58.65121	0.0037	2	0	2201	Perform Event Rule [0]BUTTON CLICKED (0)
58.65579	0.0001	2	1	2012	JDB_ClearSelection ()
58.65627	0.0000	2	1	2014	JDB_ClearSequencing ()
58.65870	0.0004	2	1	2058	JDB_SetSelection ()
58.65963	0.0002	2	1	2060	JDB_SetSequencing ()
58.66033	0.0521	2	1	2028	SelectKeyed (F0411)
58.71905	0.0440	2	1	2007	Fetch (F0411)

Dynamic Loop Designation

The Virtual Script Editor allows you to designate dynamic loops in the OneWorld Virtual User Tool script. By doing so, you add looping rules to the script. These rules allow the Virtual Script Player to perform the repetitive processing that OneWorld performs.

When you right click the Fetch line in the event stream and choose Add Loop, the Virtual Script Editor produces a Dynamic Loop Manager form, which you use to apply a dynamic loop and to establish the rules by which the Virtual Script Player manages the loop at virtual run time.

You can instruct the Virtual Script Player to run the inquiry loop until the data runs out, or you can dictate that the Virtual Script Player loop a specific number of times.


Dynamic Loop Manager

74 Matches were found.

[✓] Loop until no data

[ ] Loop X times

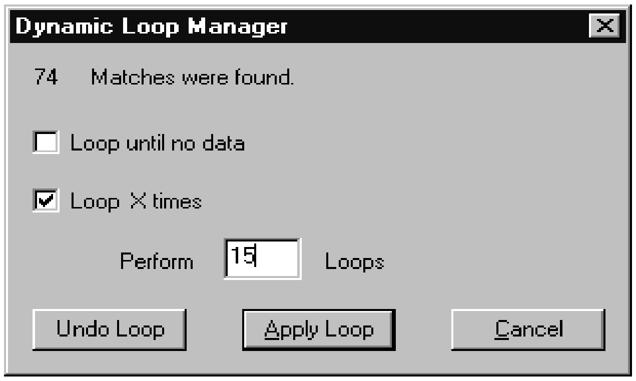
Perform 75 Loops

When you click the Apply button in the Dynamic Loop Manager, you establish the looping rule. The Virtual Script Editor indicates the loop in the event stream, in the Source Parameters pane, and in the Target Parameters pane by graying the sequence of API calls that are part of the loop.

Dynamic Loop Editing

You can edit established looping rules before generating a OneWorld Virtual User Tool script. You can also undo the loop if, for example, a series of calls does not constitute a loop. The ability to edit dynamic loops gives you an added measure of control over the creation of OneWorld Virtual User Tool scripts.

To edit the loop, you right click the Fetch command line in the event pane and choose Edit Loop Details. In the Dynamic Loop Manager form, you can change the number of times you want the script to loop, or you can choose the Undo Loop option to remove the loop.


Dynamic Loop Manager

74 Matches were found.

☐ Loop until no data

☑ Loop X times

Perform 15 Loops

hRequest Handle Value Linking

The Virtual Script Editor automatically stores hRequest handle parameter values for JDB API calls. This value represents the address of a memory block that OneWorld allocates for storage of information about an open table. The address provides you with entry to the database each time you need to open a table to perform a Fetch, FetchKeyed, SelectKeyed, FetchMatchingKey, or CloseTable function However, when you create a Virtual User Tool script and play it back, the hRequest handle parameter value probably changes. Playback could not continue if this value were constant.

The Virtual Script Editor handles the problem by storing the hRequest handle parameter value as a variable and passing the variable to the Virtual Script Player during playback. The hRequest handle variable's value changes to reflect the new address of a database table opened during script playback.

You can view the hRequest handle returned from the original API database call by clicking a call in the Source Parameters pane and viewing the details of the call. The Virtual Script Editor displays in the detail pane the request handle returned from the OpenTable API call.

If a database API call such as OpenTable leads to additional API calls, for example, FetchKeyed and CloseTable, the Virtual Script Player passes the new memory address of the opened table to these subsequent calls. During virtual playback, the subsequent APIs use the new handle to run SQL statements and to close the table.

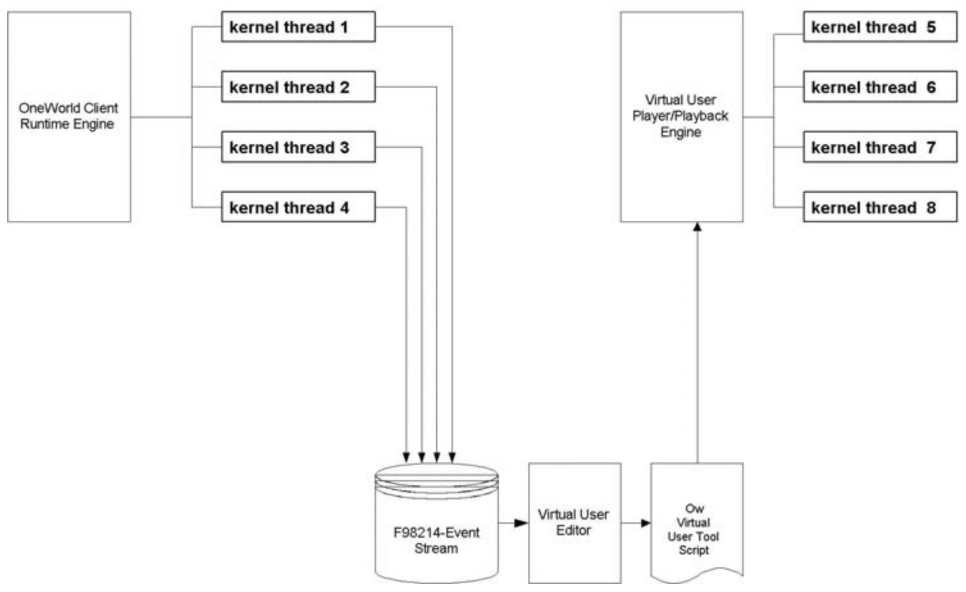
Thread Identification

The Virtual Script Editor also stores idThread numbers that OneWorld Scripting Tool gathers into the event stream during script playback. These identifier numbers represent the synchronous and asynchronous threads generated by OneWorld's run-time engine. The OneWorld run-time engine assigns each event to a thread and tags each thread with a number.

During virtual script playback, the Virtual Script Editor passes idThread parameters to the Virtual Script Player, which assigns different idThreads to each event and associates each script event, on the fly, with its new identifier.

The following diagram presents a simplified view of OneWorld Virtual User Tool's thread management strategy. Note that during OneWorld Virtual User Tool script playback, the Virtual Script Player renumbers the original threads generated during OneWorld Scripting Tool script playback. The Virtual Script Editor's role is to store the thread identification information and to pass it on through the virtual script.

OW Virtual User Tool Thread Generation and Management



Timing Interval Maintenance

The Virtual Script Editor also automatically handles problems of timing that might emerge in the creation of a OneWorld Virtual User Tool script. The time-stamped event stream log of events captures the length of time elapsed between each event. However, when you create a OneWorld Virtual User Tool script, you cannot account for the different environment in which the OneWorld Virtual User Tool script runs. For example, the workstation on which the script runs might be simulating 50 users. The power of the workstation might differ from the one on which the original script data was captured. Finally, the server against which the OneWorld Virtual User Tool script runs might be more or less powerful than the server against which the original script ran. All of these factors combine to make it likely that the time that a OneWorld Virtual User Tool script requires to run will differ from the time that the original script required to run.

The Virtual Script Editor handles this problem by preserving in the virtual script the time intervals that existed between events when you ran the original script.

The time intervals represent the length of time required by OneWorld to carry out processing between events. Thus, even if an API call during virtual script playback takes longer to carry out than the API call in the original script, the Virtual Script Player preserves the original time difference between one API call and the next.

The Virtual Script Player initialization file also contains timing parameters that govern the playback of the OneWorld Virtual User Tool script. You can adjust to a limited extent some of these parameters to change, for example, how fast the OneWorld Virtual User Tool script plays back.

See Also

- *Virtual Script Player Initialization File Parameters*
- *Playback Timing*

OneWorld Virtual User Tool Script Generation

The OneWorld Virtual User Tool script is the output from the Virtual Script Editor and the input to the Virtual Script Player. OneWorld Virtual User Tool scripts appear in text file form with a header and the edited list of OneWorld events you captured during script playback, imported into the Virtual Script Editor, and edited, both manually and automatically.

For ease and consistency of interpretation, each event in the script is structured in a particular way. For example, each event begins with the letter ‘e’ and is followed by a unique identifying number. In addition, each script identifies the environment, the network user, and contains an open table handle. However, it’s not necessary or even desirable that you look at a OneWorld Virtual User Tool script in order to run it.

OneWorld Virtual User Tool classifies three types of events and identifies them as such in the script:

- Functions, which include JDB and CallObject APIs.
- Assignment statements, which refer to values typed in OneWorld Scripting Tool.
- Conditional tests/branches, which are if/then statements.

OneWorld Virtual User Tool divides each event into parts and, in turn, identifies each of the parts on the basis of an assigned format and a unique value. In short, the OneWorld Virtual User Tool script contains the details necessary for the Virtual Script Player to simulate running the OneWorld kernel.

OneWorld Virtual User Tool identifies transaction boundaries, which you can set in the original script by designating a script command as the start of the transaction and another script command as the end of the transaction. Setting transaction boundaries can help you to analyze OneWorld's performance in running a particular series of tasks. For more information on setting transaction boundaries, see *Rectifying Irregular Transaction Times*.

Virtual Script Player

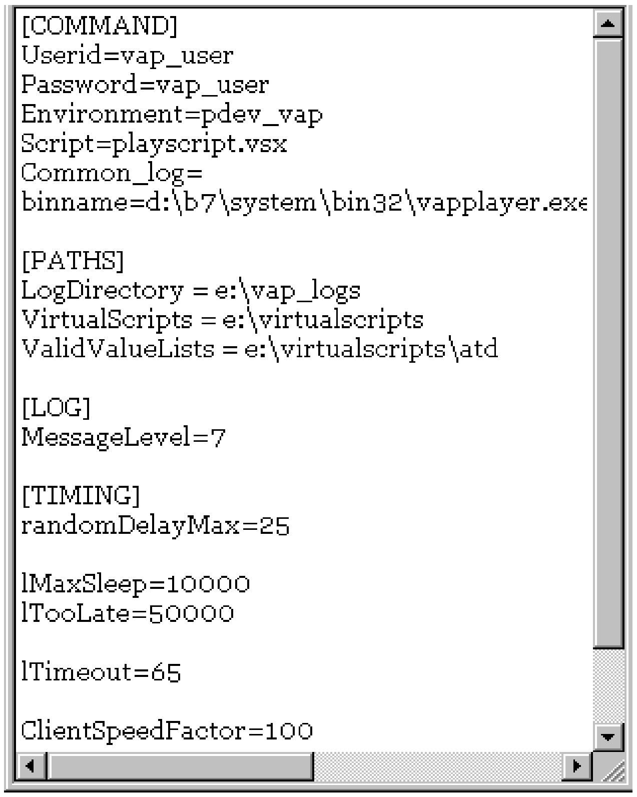
The Virtual Script Player uses the OneWorld Virtual User Tool script you generate in the Virtual Script Editor to simulate the concurrent activities of one or more OneWorld users. It bypasses the presentation layer of OneWorld and reproduces the OneWorld application calls to the JDB and CallObject middleware. This reproduction is based on the timing and the sequencing of data in the event stream you generate with OneWorld Scripting Tool, manipulate in the Virtual Script Editor, and generate in modified form in the OneWorld Virtual User Tool script. In essence, the Virtual Script Player assumes the role of the OneWorld run-time engine.

This chapter discusses the following essentials of the Virtual Script Player operation:

- ❑ Virtual Script Player initialization file parameters
- ❑ Virtual Script Player command line
- ❑ OneWorld environment initialization
- ❑ Modes of operation
- ❑ Preprocessing of valid values list data
- ❑ Valid values list processing
- ❑ Date formatting
- ❑ Script failure
- ❑ Virtual Script Player limitations

Virtual Script Player Initialization File Parameters

The vap.ini file is a text file that contains the parameters that define the way that the Virtual Script Player runs. These parameters govern the paths the Virtual Script Player follows to find files, synchronize playback timing, and set playback speed.



```
[COMMAND]
Userid=vap_user
Password=vap_user
Environment=pdev_vap
Script=playscript.vsx
Common_log=
binname=d:\b7\system\bin32\vapplayer.exe

[PATHS]
LogDirectory = e:\vap_logs
VirtualScripts = e:\virtualscripts
ValidValueLists = e:\virtualscripts\atd

[LOG]
MessageLevel=7

[TIMING]
randomDelayMax=25

lMaxSleep=10000
lTooLate=50000

lTimeout=65

ClientSpeedFactor=100
```

You can change the parameters, within established limits, to set the way the OneWorld Virtual User Tool scripts play. This topic discusses the following vap.ini file parameters:

- ❑ Command
- ❑ Paths
- ❑ Log
- ❑ Timing

Command

The Command section of the vap.ini file contains the parameters that are necessary for interaction between Virtual Runner, which manages script playback, and the Virtual Script Player, which runs playback. These parameters specify the user ID, password, OneWorld environment, script name, log file of summary playback statistics, and location of the Virtual Script Player executable.

The following table summarizes the [COMMAND] parameters and the meaning of each one:

Parameter	Meaning
UserID=	OneWorld Virtual User Tool user ID. Override on command line by entering -u and a user ID.
Password=	Password for OneWorld Virtual User Tool user. Override on command line by entering -p and a password.
Environment=	OneWorld Environment for OneWorld Virtual User Tool script playback. Override on command line by entering -e and an environment.
Script=playscript.vsx	Name of OneWorld Virtual User Tool script (user can specify full path name for script here.)
Common log=	Log file to which Virtual Script Player will write summary statistics for all playback sessions. Default folder is Vap_logs. Used only with Virtual Runner.
Binnmae=d:\b7\system \bin32\vapplayer.exe	Path by which Virtual Runner finds the Virtual Script Player executable.

Paths

The Path section of the vap.ini file identifies the directories for files needed by the Virtual Script Player. The contents of the needed files are:

- Log file, which gives detailed information on each OneWorld Virtual User Tool script playback session, the script name, and a line-by-line summary of each event in the script. The Virtual Script Player logs each event as it completes. The file also includes the start time and the date of the log.
- OneWorld Virtual User Tool script file, which stores all scripts that you might use for virtual playback.
- Valid values list file, which stores any valid values lists that the Virtual Script Player draws on for input values to run business functions. The Virtual Script Player uses valid values lists to get a new value each time it runs a business function.

The default file paths are as follows:

File/Contents	Parameter in vap.ini file	Default Path
Log of OneWorld Virtual User Tool playback events and messages	LogDirectory	c:\autopilot\VAP_LOGS
OneWorld Virtual User Tool scripts	VirtualScripts	c:\autopilot\VSX
Valid value lists	ValidValueLists	c:\autopilot\ATD

Log

You use the Log section of the vap.ini file to specify the type of messages that the Virtual Script Player will write to a log file during a OneWorld Virtual User Tool script playback session. These messages can be important for debugging purposes. The following table summarizes the available log parameters and the debug message level that each one represents:

Log Parameter	Debug Message Level
31	Maximum log output; flush log file after each message (LoadRunner excluded)
15	Parameter values and value substitutions
7	Error, warning, and status messages
3	Error and warning messages
1	Error messages only
0	Minimal messages

Note: You can cause the log file buffer to flush after every message by adding 16 to any parameter less than 31. However, you should not routinely do this, as flushing frequently increases file system overhead. For the same reason, you should not routinely set the log parameter at 31.

Timing

The Timing initialization parameters of the vap.ini file help you to specify the terms under which OneWorld Virtual User Tool scripts will play back:

- Rendezvous of multiple playback sessions, to control the amount of time the Virtual Script player delays a playback session following a rendezvous of multiple scripts running on a single workstation
- Synchronization of playback events, to set limits on the amount of time that threads can be inactive, events can occur behind the start time scheduled by the script, or that a thread has to wait for an API value or a handle parameter
- Playback speed, to adjust the amount of time between by events to compensate for a very speedy or slow client workstation

The following table lists the OneWorld Virtual User Tool timing initialization parameters, their default values, what they govern, and the kind of timing factor to which they relate:

Parameter Name	**Default Value**	**Meaning**	**Timing Factor**
RandomDelayMax	0 seconds; can be set as high as 3600	Allows user to set a maximum period that the Virtual Script Player will wait after the LoadRunner "OWLogin" rendezvous and environment initialization to begin each playback session. The default value means that following rendezvous, each player session proceeds without delay.	Rendezvous of multiple playback sessions
lMaxSleep	10,000 milliseconds	Establishes an upper limit on thread sleep time. Inactive threads must check on system status at least this often. If errors require the Virtual Script Player to shut down all threads, the parameter also determines the maximum amount of time required for the Player to shut down.	Playback synchronization
1TooLate	200 milliseconds -- set higher in debugging situations	Latest that any event can be run after the script schedules its start without causing virtual script playback to terminate.	Playback synchronization
1Timeout	60 seconds	Maximum number of seconds that an event has to run. If that number exceeds the parameter, Virtual Script Player terminates the playback session.	Playback synchronization

ClientSpeedFactor	100	Controls timing between script events by a constant factor. Decreasing the value of the parameter decreases the time between events.	Playback speed

See Also

- *Launching and Managing Multiple Script Playback*
- *Playback Timing*

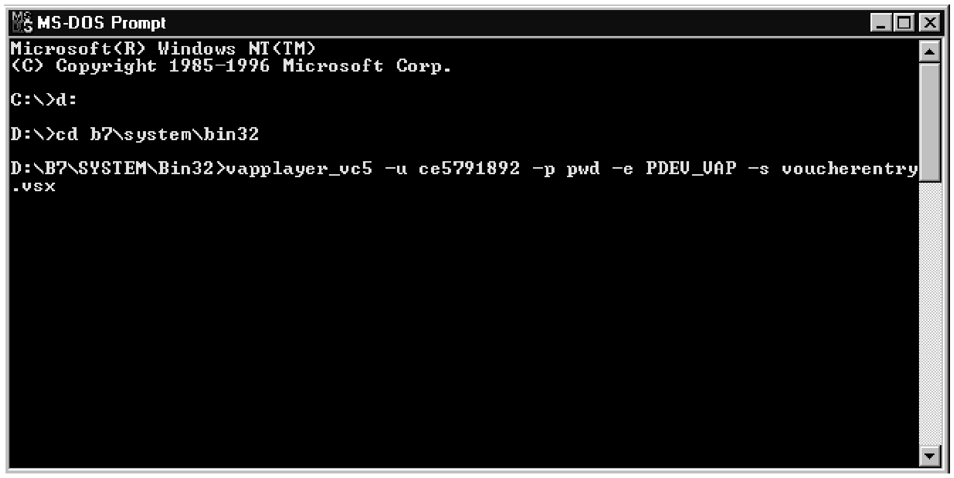
Virtual Script Player Command Line

You can launch the Virtual Script Player from LoadRunner, from Virtual Runner, or from the DOS command line. The command line must have entries that specify the user, the user's password, the environment, and the script name with a default extension of .vsx for any OneWorld Virtual User Tool script, although this extension is not required.

There are four required entries to the command line.

Command Line Abbreviation	Meaning	Sample Entry
–u	User	ce5791892
–p	OneWorld user ID	–p pwd
–e	Environment	–e PDEV_VAP
–s	Script Name	–s voucherentry100.vsx

The completed command line allows you to launch a virtual playback session.



```
Microsoft(R) Windows NT(TM)
(C) Copyright 1985-1996 Microsoft Corp.

C:\>d:

D:\>cd b7\system\bin32

D:\B7\SYSTEM\Bin32>vapplayer_vc5 -u ce5791892 -p pwd -e PDEV_VAP -s voucherentry
.vsx
```

OneWorld Environment Initialization

The Virtual Script Player does not immediately begin playing a OneWorld Virtual User Tool script upon launch from the DOS command line, from Virtual Runner, or from LoadRunner. In fact, the Virtual Script Player reads in the script and runs events that generate a OneWorld environment structure. The data driving OneWorld environment generation comes from entries to the command line. For example, one user might create a OneWorld Virtual User Tool script, but another user might play the script. During initialization, the Virtual Script Player passes in the OneWorld user ID of the user playing the script, thereby creating the proper OneWorld environment. Therefore, you can run the Virtual Script Player in a different environment than the one in which you or someone else created the OneWorld Virtual User Tool script.

OneWorld environment initialization takes approximately 15 to 30 seconds. LoadRunner regards this passage as initializing time, while the DOS command line reads it as busy activity.

Modes of Operation

The Virtual Script Player automatically detects if you have launched a OneWorld Virtual User Tool script from the DOS command line, from Virtual Runner, or from LoadRunner. If LoadRunner launches the script, the Virtual Script Player will respond to stop/pause commands and sends transaction times and log output to LoadRunner. In addition, the Virtual Script Player completes a LoadRunner rendezvous just after it has initialized the OneWorld environment.

Preprocessing of Valid Values List Data

OneWorld Virtual User Tool's preprocessing capability works in concert with the Virtual Script Editor and Virtual Script Player to use valid values lists during script playback. You flag valid value lists because virtual playback requires the values contained in these lists as parameters for API database calls.

When a OneWorld Virtual User Tool script specifies that a particular value originates in a OneWorld Scripting Tool valid values list, the Virtual Script Player reads the valid values list file. All valid value lists are identified by the extension .atd. Before the Virtual Script Player plays the script, it performs preprocessing that includes looking up the database values in the valid values list and storing them until they are required as parameters to API calls. When the Virtual Script Player runs the OneWorld Virtual User Tool script, the stored list supplies the parameters needed for JDB or CallObject calls.

Preprocessing plays an important role in the OneWorld Virtual User Tool scheme because it takes care of the look-up and load of the valid values that the Virtual Script Player needs for OneWorld Virtual User Tool script execution. This ensures that the required values exist prior to playback. If the Virtual Script Player had to run database lookups at the time of script playback, the result would be artificial load on the database, which would, in turn, distort the simulation of OneWorld activity that OneWorld Virtual User Tool seeks to achieve.

Valid Values List Processing

The Virtual Script Player defines the location of any valid value lists that are part of the OneWorld Virtual User Tool script in the vap.ini file. The Virtual Script Player reads valid value lists that are 64K or smaller into memory. If the file is larger than 64K, the Virtual Script Player must read it from the file. During virtual playback, if the Virtual Script Player reaches the end of a valid value list, it starts back at the beginning of the list, reuses the first value, and continues in sequence until virtual playback is complete.

Date Formatting

The Virtual Script Player must assume the format for date strings for valid value lists and for literal typed-in values from OneWorld Scripting Tool. Therefore, the Virtual Script Player supports different date formats that might appear in the OneWorld Virtual User Tool script, including mm/dd/yyyy and Julian date strings (i.e., 99064).

Caution: The Virtual Script Editor correctly formats date entries for literal values, but not for date entries in valid value lists.

Script Failure

Script failure might occur during the initialization process. For example, a branch event in the script might not refer to a valid event, or the events might not occur in the same thread. In that case, the script fails before it is launched because the Virtual Script Player can not validate the events. On initialization, the Virtual Script Player also validates function parameters. For example, a parameter such as Fetch might accept only 0 or 1 as values. If a different value is used, validation fails and, thus, the script fails before launching.

If the script fails during playback, the failure shuts down script processing. For most API calls, failure to return a success code causes the playback process to halt. The shutdown occurs without user intervention. LoadRunner, for example, returns a failure report, and the Virtual Script Player sends an error message to the log file, for example: LoadRunner/Test Name/Local1/Subdirectory Name. There is one subdirectory for every LoadRunner test session, meaning that 50 simulated user test sessions produce 50 subdirectories.

If you launch the Virtual Script Player from a command line or from Virtual Runner and script failure occurs, no error message appears on the screen. You must open the log file that stores the test session results and examine the messages, a task you complete by searching on the keyword “Error.”

Virtual Script Player Limitations

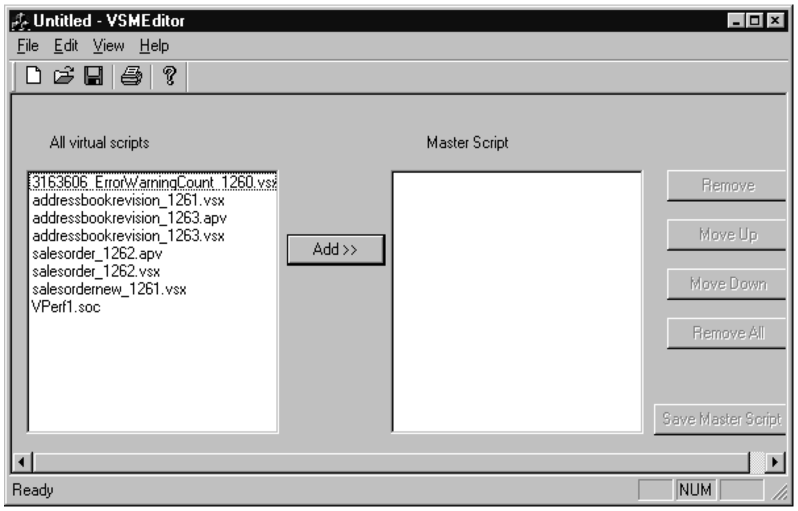
The overriding consideration for OneWorld Virtual User Tool script playback is that client workstations must not bottleneck the playback process. You must determine how many processes that the workstation can realistically support, based on an analysis of workstation memory and CPU capability. Running either Task Manager or Performance Monitor can assess these capabilities.

Other Virtual Script Player limitations are hard-coded. If the Virtual Script Player gets a script that exceeds these limitations, you receive error messages that require a service pack to address. First, the Virtual Script Player supports up to 30 active user handles and 60 active request handles per session. Second, the Virtual Script Player can process only a certain number of status messages per second under LoadRunner. If the playback exceeds that number, some of the messages are lost, but the Virtual Script Player does not shut down.

VSMEditor

After you have created a number of OneWorld Virtual User Tool scripts, the VSMEditor allows you to concatenate any number of those scripts into a single master script. Stringing together single scripts into a master is advantageous because you can run a series of unrelated tasks during testing.

You control the VSMEditor from the following form, which you access by clicking the VSMEditor executable.



3163606_ErrorWarningCount_1260.vsx
addressbookrevision_1261.vsx
addressbookrevision_1263.apv
addressbookrevision_1263.vsx
salesorder_1262.apv
salesorder_1262.vsx
salesordernew_1261.vsx
VPerf1.soc

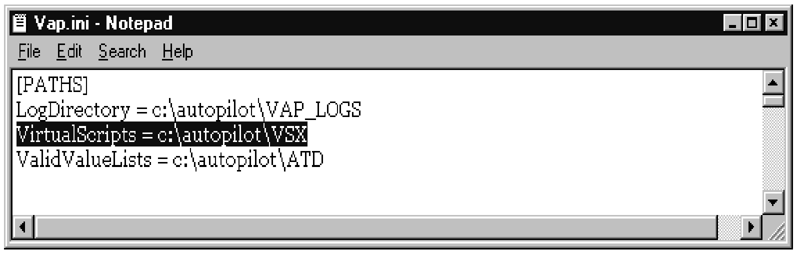
The essential components of the VSMEditor are:

- All Virtual Scripts list box
- Master Scripts list box
- VSM files

All Virtual Scripts List Box

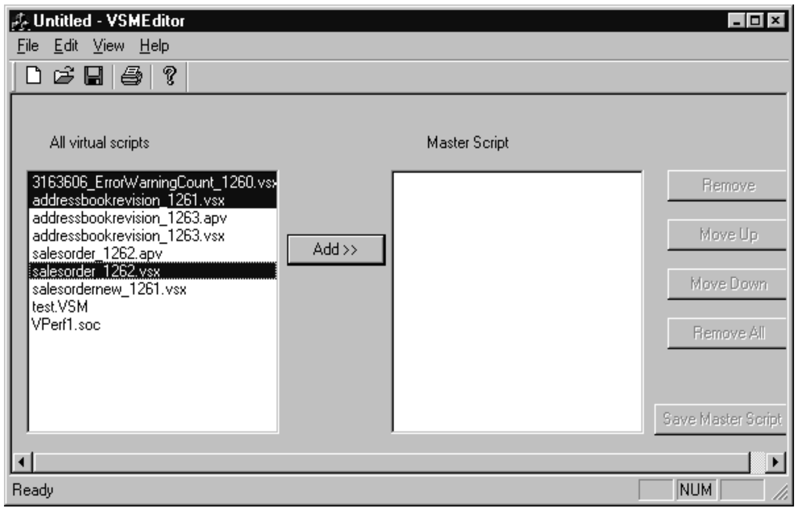
The All Virtual Scripts list box contains all OneWorld Virtual User Tool script files that you have created. These files contain a .vsx extension. In addition, any master scripts that you have created appear in this list box. Master scripts contain a .vsm extension. The location of any OneWorld Virtual User Tool script files that appear in the All Virtual Scripts list box is determined by the value of the VirtualScripts parameter of the vap.ini's PATHS section.

You enter the path to the location of your virtual scripts to set the VirtualScripts parameter:



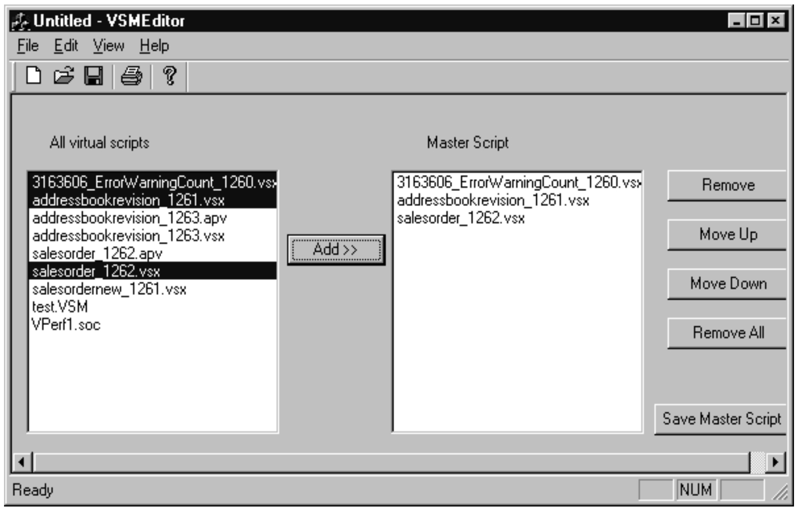
```
[PATHS]
LogDirectory = c:\autopilot\VAP_LOGS
VirtualScripts = c:\autopilot\VSX
ValidValueLists = c:\autopilot\ATD
```

You can use any script in the All Virtual Scripts list box to create a master script. You create the script concatenation by choosing one of the scripts in the box, and then holding down the Control key or the Shift key to choose additional scripts.



```
3163606_ErrorWarningCount_1260.vsx
addressbookrevision_1261.vsx
addressbookrevision_1263.apv
addressbookrevision_1263.vsx
salesorder_1262.apv
salesorder_1262.vsx
salesordernew_1261.vsx
test.VSM
VPerf1.soc
```

You click the Add button to add the files you chose to the Master Script list box.



All virtual scripts:

```
3163606_ErrorWarningCount_1260.vsx
addressbookrevision_1261.vsx
addressbookrevision_1263.apv
addressbookrevision_1263.vsx
salesorder_1262.apv
salesorder_1262.vsx
salesordernew_1261.vsx
test.VSM
VPerf1.soc
```

Master Script:

```
3163606_ErrorWarningCount_1260.vsx
addressbookrevision_1261.vsx
salesorder_1262.vsx
```

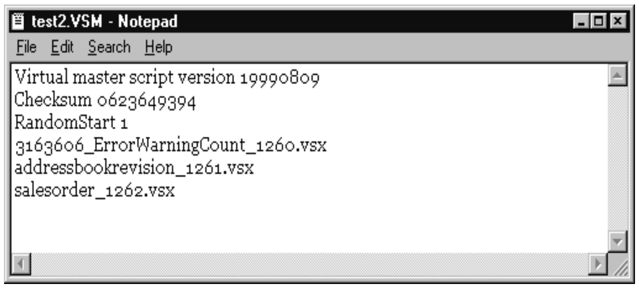
Master Scripts List Box

The Master Script list box shows all the scripts that you have currently chosen for addition to a new .vsm (virtual script master) file. You can manipulate the script list in the Master Script list box by using the buttons adjacent to the box:

- Remove button: deletes the chosen script from the Master Script list
- Move Up and Move Down buttons: shift the position of the selected script in the list
- Remove All button: deletes all script from the list
- Save Master Script button: saves the list of scripts as a .vsm file

VSM Files

The VSMEditor creates a .vsm text file when you save a master script. You can change these files only through the VSMEditor because the file contains a checksum value that verifies the file's integrity. The OneWorld Virtual User Tool scripts always run in the sequence listed in the .vsm file. However, the first script to run is chosen randomly when the RandomStart parameter in the text file is set to 1.



```
Virtual master script version 19990809
Checksum 0623649394
RandomStart 1
3163606_ErrorWarningCount_1260.vsx
addressbookrevision_1261.vsx
salesorder_1262.vsx
```

The actual OneWorld Virtual User Tool scripts are not included in the .vsm master file. Therefore, you should not delete scripts from the folder that contains the .vsx files.

Virtual Runner

Virtual Runner controls the Virtual Script Player sessions on a single workstation and provides command and control functions for Virtual Script Player testing, including the following:

- Allowing users to start one or more Virtual Script Player sessions on a single workstation
- Allowing users to play multiple iterations of a single script
- Reporting Player session status (pass/fail) to user
- Summarizing performance statistics over all Virtual Script Player sessions in a test

Virtual Runner Components

The key components of Virtual Runner are:

❑ Actions tools

❑ Player session columns

Actions Tools

You use the Actions tools to set up and to launch a Virtual Runner session. You can choose the scripts that you want to run as well as the number of script playback iterations. In addition, following playback you can access a log that contains pertinent information about the playback session.

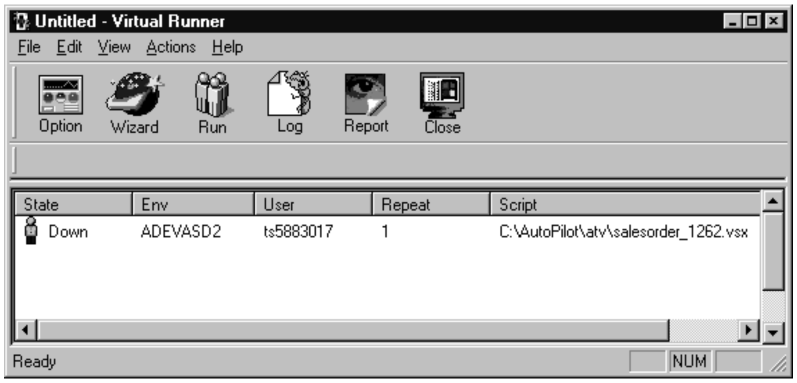
The Virtual Runner toolbar contains six buttons.

Virtual Runner Toolbar Button	Button Function
Option	Allows you to specify the User ID, Password, and Environment for the virtual playback session.

Wizard	Steps you through the process of specifying all the Virtual Script Player session parameters, including the number of scripts to run and the script playback iterations.
Run	Runs the virtual script playback session.
Log	Displays the Log Viewer screen, which provides information about the last completed Virtual Script Player session.
Report	Prints a copy of the Virtual Script Player session log information.
Close	Closes the Virtual Runner window after you have decided whether or not to save the results of the Virtual Script Player session.

Player Session Columns

After you complete setting up the parameters for the Virtual Script Player session, Virtual Runner displays the names of the scripts that you want to run. Initially, the status of the script is "Down," indicating that you have not yet run it.



State	Env	User	Repeat	Script
Down	ADEVASD2	ts5883017	1	C:\AutoPilot\atv\salesorder_1262.vsx

After you run a test, Virtual Runner changes the status to indicate success or failure.

Each column displays information about your Virtual Script Player sessions. The following descriptions summarize the purpose of each player session column:

- The State column indicates the current state of the player session. For example, after you complete a successful execution of a player session, this column displays the word "Success."

- The Env column indicates the specified environment for the current session. The environment is specified using the Options button, or when you use the Virtual Runner Wizard.
- The User column displays the User name you specified using the Options button, or when you use the Virtual Runner Wizard.
- The Repeat column specifies the number of times the script is repeated when executing the player session. You specify this parameter when you use the Virtual Runner Wizard.
- The Script column indicates the path and file name of the script for the current player session. You specify these parameters when you use the Virtual Runner Wizard.

Creating OneWorld Virtual User Tool Scripts

Creating Virtual User Scripts

Your goal in using OneWorld Virtual User Tool is to create a script that can simulate the activities of OneWorld as it handles the workload generated by many users. To achieve that goal, you utilize two key components of J.D. Edwards' automated testing tools architecture:

- OneWorld Scripting Tool
- Virtual Script Editor

Using these two tools, you accomplish a sequence of tasks to create a virtual script:

- Create a OneWorld Scripting Tool script
- Run the OneWorld Scripting Tool script with playback configured so that you can capture OneWorld and OneWorld Scripting Tool data (stream of consciousness)
- Import the event stream into the Virtual Script Editor
- Create value links between source parameters of API calls and the target parameters of other API calls to ensure that usable data flows between API calls when you run the virtual script
- Add loops to the OneWorld Virtual User Tool script to account for repetitive processing such as data retrieval
- Generate the OneWorld Virtual User Tool script, which the Virtual Script Player runs

After you have created a virtual script, the Virtual Script Player runs the script, and you can use Virtual Runner or LoadRunner to manage the number of sessions, either from a single or from multiple workstations.

This section discusses the required tasks that are part of the two main components of the process of creating a virtual script:

❑ Capturing and importing test results

❑ Editing the virtual script

Capturing and Importing Test Results

OneWorld Scripting Tool allows you to create scripts that test OneWorld applications. When you create a script, you can configure OneWorld Scripting Tool's playback function so that it captures and saves the results of your playback session, which it stores in OneWorld table 98214 as an event stream.

You can view the playback results in a variety of ways. You can view the event stream alone, you can view details of individual events, or you can timing information about groups of events and thread identifiers displayed in a horizontal bar graph.

The data OneWorld Scripting Tool captures provides the raw material for your Virtual OneWorld Scripting Tool script. After you capture OneWorld Scripting Tool script data, you import it to the Virtual Script Editor so that you can prepare a virtual script.

This chapter discusses the OneWorld Scripting Tool data capture process and its relation to preparing a Virtual OneWorld Scripting Tool script:

- ❑ Capturing Test Results
- ❑ Importing Test Results
- ❑ Viewing Test Results

Capturing Test Results

In order to gather the raw data for a virtual script, you must first write and run a OneWorld Scripting Tool script and capture the results of the playback as an event stream. You use the Tools option in the menu bar of the OneWorld Scripting Tool form to set up the capture mechanism.

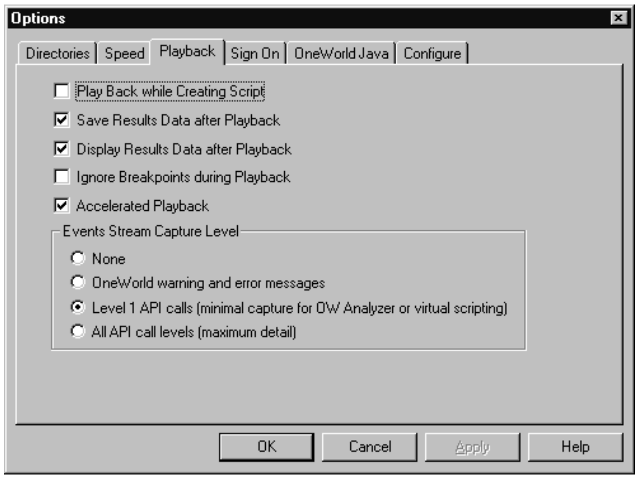
▶ To capture test results

1. Create the OneWorld Scripting Tool script or open an existing script.

   See the *OneWorld Scripting Tool Guide* for detailed instructions on creating OneWorld Scripting Tool scripts.

   Caution: Make sure that OneWorld Scripting Tool automatically logs into a OneWorld environment. A script that does not include the OneWorld log-on will not function correctly in Virtual OneWorld Scripting Tool because it will not contain the data required for the Virtual Script Player to initialize the OneWorld environment.

2. Open the OneWorld Scripting Tool script.
3. From the Tools menu, choose Options.
4. In the Options form, choose the Playback tab.
5. Choose the Save Results and Display Results options.
6. In the Event Stream Capture Level portion on the Playback tab, choose the Level 1 API calls option.



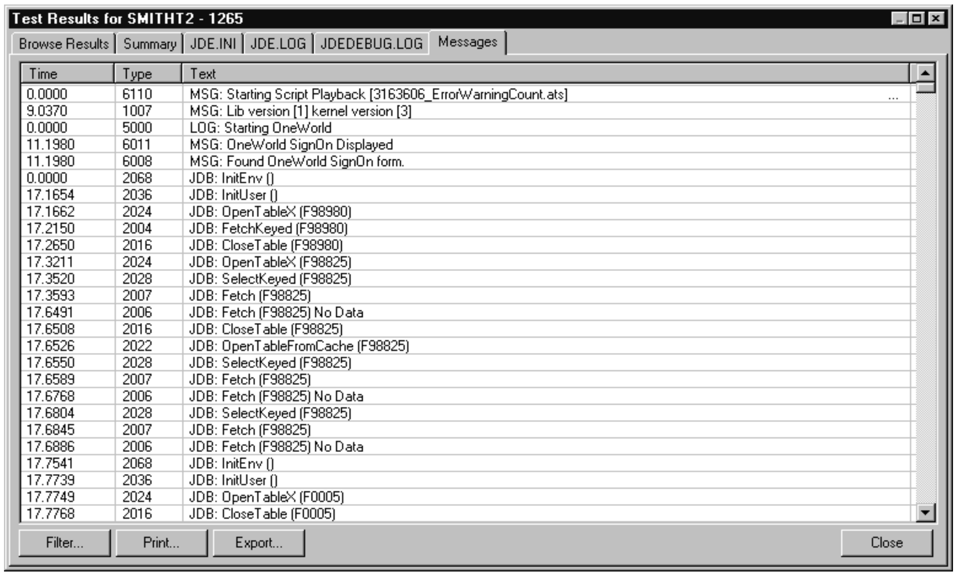
- [ ] Play Back while Creating Script
- [x] Save Results Data after Playback
- [x] Display Results Data after Playback
- [ ] Ignore Breakpoints during Playback
- [x] Accelerated Playback

Events Stream Capture Level
- ○ None
- ○ OneWorld warning and error messages
- ● Level 1 API calls (minimal capture for OW Analyzer or virtual scripting)
- ○ All API call levels (maximum detail)

Note: If you want to capture more script playback events, choose the All API call levels option. Remember, however, you generate a much larger event stream if you choose this option.

7. Click OK.
8. Save the OneWorld Scripting Tool script.
9. In the OneWorld Scripting Tool menu bar, click Play and choose Play from Top.

   OneWorld Scripting Tool runs the script. The Play From Top command generates test results for DENPCX (where DENPCX=the name of the machine on which OneWorld Scripting Tool resides). The OneWorld Scripting Tool Results form displays detailed information on the playback session.



Time	Type	Text
0.0000	6110	MSG: Starting Script Playback [3163606_ErrorWarningCount.ats] ...
9.0370	1007	MSG: Lib version [1] kernel version [3]
0.0000	5000	LOG: Starting OneWorld
11.1980	6011	MSG: OneWorld SignOn Displayed
11.1980	6008	MSG: Found OneWorld SignOn form.
0.0000	2068	JDB: InitEnv ()
17.1654	2036	JDB: InitUser ()
17.1662	2024	JDB: OpenTableX (F98980)
17.2150	2004	JDB: FetchKeyed (F98980)
17.2650	2016	JDB: CloseTable (F98980)
17.3211	2024	JDB: OpenTableX (F98825)
17.3520	2028	JDB: SelectKeyed (F98825)
17.3593	2007	JDB: Fetch (F98825)
17.6491	2006	JDB: Fetch (F98825) No Data
17.6508	2016	JDB: CloseTable (F98825)
17.6526	2022	JDB: OpenTableFromCache (F98825)
17.6550	2028	JDB: SelectKeyed (F98825)
17.6589	2007	JDB: Fetch (F98825)
17.6768	2006	JDB: Fetch (F98825) No Data
17.6804	2028	JDB: SelectKeyed (F98825)
17.6845	2007	JDB: Fetch (F98825)
17.6886	2006	JDB: Fetch (F98825) No Data
17.7541	2068	JDB: InitEnv ()
17.7739	2036	JDB: InitUser ()
17.7749	2024	JDB: OpenTableX (F0005)
17.7768	2016	JDB: CloseTable (F0005)

For complete details of the various tabs and information displayed in the Test Results form, see *OneWorld OneWorld Scripting Tool*.

10. Click the Close button to close the Test Results window.
11. Click File/Exit to close OneWorld Scripting Tool.

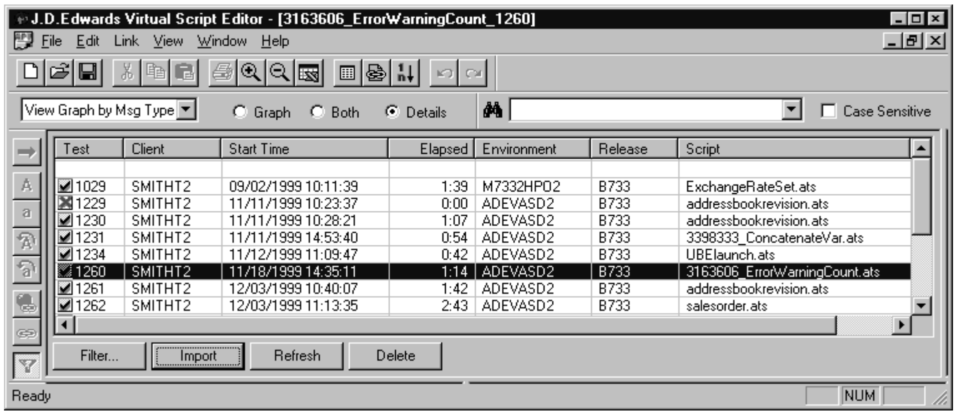
Importing Test Results

After you have run a OneWorld Scripting Tool script and saved the playback results, you can import the event stream into the Virtual Script Editor. Importing the script allows you to use the Virtual Script Editor to forge value links between source and target parameters of API calls; to identify, designate, and edit repetitive processing; and to generate a virtual script.

▶ To import playback results to the Virtual Script Editor

1. Open the Virtual Script Editor.

   The Virtual Script Editor form appears, populated by a chronological list of tests you have run, along with summary information about each one.



J.D.Edwards Virtual Script Editor - [3163606_ErrorWarningCount_1260]

View Graph by Msg Type ( ) Graph ( ) Both (●) Details ☐ Case Sensitive

Test	Client	Start Time	Elapsed	Environment	Release	Script
✓ 1029	SMITHT2	09/02/1999 10:11:39	1:39	M7332HPO2	B733	ExchangeRateSet.ats
☒ 1229	SMITHT2	11/11/1999 10:23:37	0:00	ADEVASD2	B733	addressbookrevision.ats
✓ 1230	SMITHT2	11/11/1999 10:28:21	1:07	ADEVASD2	B733	addressbookrevision.ats
✓ 1231	SMITHT2	11/11/1999 14:53:40	0:54	ADEVASD2	B733	3398333_ConcatenateVar.ats
✓ 1234	SMITHT2	11/12/1999 11:09:47	0:42	ADEVASD2	B733	UBElaunch.ats
✓ 1260	SMITHT2	11/18/1999 14:35:11	1:14	ADEVASD2	B733	3163606_ErrorWarningCount.ats
✓ 1261	SMITHT2	12/03/1999 10:40:07	1:42	ADEVASD2	B733	addressbookrevision.ats
✓ 1262	SMITHT2	12/03/1999 11:13:35	2:43	ADEVASD2	B733	salesorder.ats

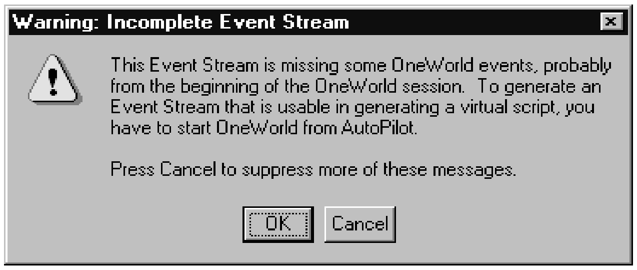
Ready

2. Select a script to import.
3. Click the Import button.

   After the Virtual Script Editor imports the script, an APEdit dialog box appears, confirming that the import was successful.

4. In the Virtual Script Editor dialog box, click OK.

   Caution: If you attempt to import a OneWorld Scripting Tool script that you captured without a OneWorld sign-on, OneWorld Virtual User Tool displays a warning dialog box.



Warning: Incomplete Event Stream

This Event Stream is missing some OneWorld events, probably from the beginning of the OneWorld session. To generate an Event Stream that is usable in generating a virtual script, you have to start OneWorld from AutoPilot.

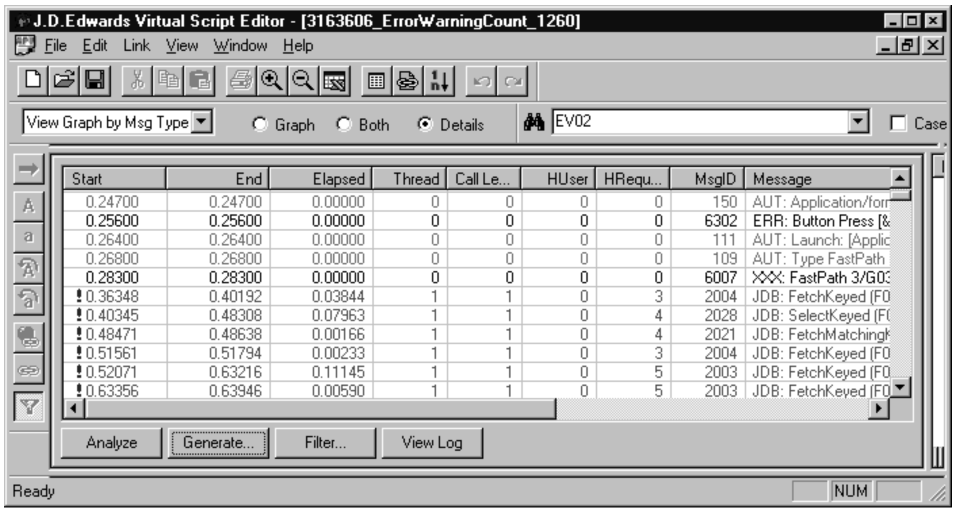
Press Cancel to suppress more of these messages.

If this message appears, you should recapture the data, making sure that you sign on to OneWorld through OneWorld Scripting Tool. To do so, close OneWorld, and then launch the OneWorld Scripting Tool script. OneWorld Scripting Tool handles your OneWorld sign-on.

Viewing Test Results

After you successfully import the results of a script playback, the event stream appears in event stream event pane of the Virtual Script Editor form.

Caution: An exclamation point next to a start time in a line of the event stream indicates that an error occurred during data capture.


J.D.Edwards Virtual Script Editor - [3163606_ErrorWarningCount_1260]
File Edit Link View Window Help
View Graph by Msg Type
Graph
Both
Details
EV02
Case
Analyze
Generate...
Filter...
View Log
Ready
NUM

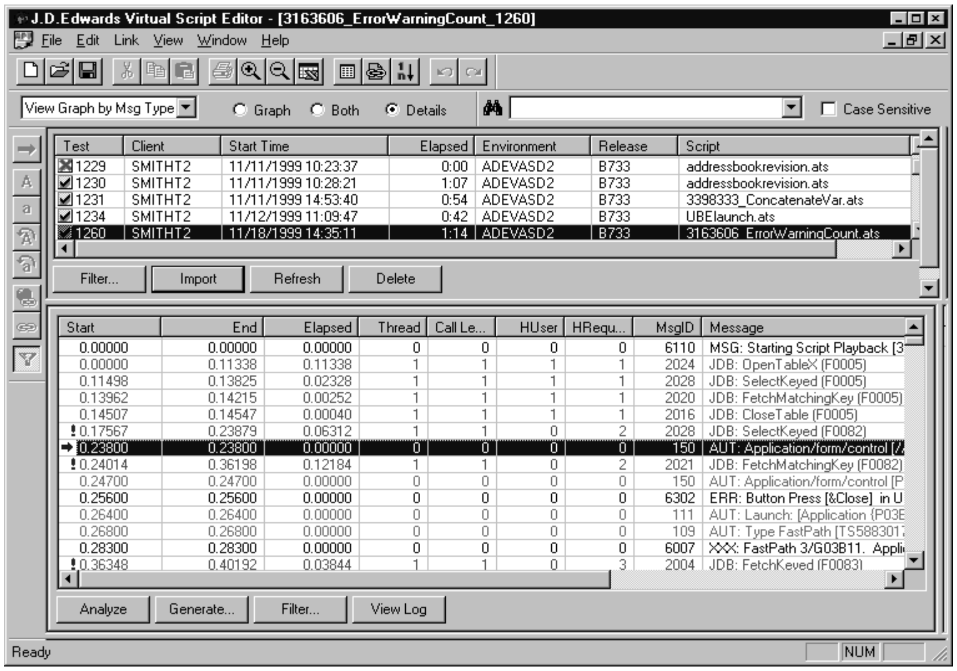
Start	End	Elapsed	Thread	Call Le...	HUser	HRequ...	MsgID	Message
0.24700	0.24700	0.00000	0	0	0	0	150	AUT: Application/for
0.25600	0.25600	0.00000	0	0	0	0	6302	ERR: Button Press [&
0.26400	0.26400	0.00000	0	0	0	0	111	AUT: Launch: [Applic
0.26800	0.26800	0.00000	0	0	0	0	109	AUT: Type FastPath
0.28300	0.28300	0.00000	0	0	0	0	6007	XXX: FastPath 3/G0
! 0.36348	0.40192	0.03844	1	1	0	3	2004	JDB: FetchKeyed (F0
! 0.40345	0.48308	0.07963	1	1	0	4	2028	JDB: SelectKeyed (F
! 0.48471	0.48638	0.00166	1	1	0	4	2021	JDB: FetchMatching
! 0.51561	0.51794	0.00233	1	1	0	3	2004	JDB: FetchKeyed (F0
! 0.52071	0.63216	0.11145	1	1	0	5	2003	JDB: FetchKeyed (F0
! 0.63356	0.63946	0.00590	1	1	0	5	2003	JDB: FetchKeyed (F0

If you find exclamation points in the event stream, you should investigate the possible causes for the error, and then edit and rerun the OneWorld Scripting Tool script. For more information on troubleshooting data capture problems, see *OneWorld Virtual User Tool Troubleshooting Tips and Techniques.*

To view the event stream alone, click the Details option in the toolbar. To view categories of playback events and/or thread activity represented in a horizontal bar graph by duration and time of occurrence, click the Graph option, then click the scroll bar button in the form to choose either View Graph by Message Type or View Graph by Thread ID. Click the Both option to view both the linear event stream and the horizontal bar graph representation.

▶ To view the event stream

1. In the toolbar of the Virtual Script Editor form, click one of the following options:
   - Details
   - Graph
   - Both
2. Choose Details to view the linear event stream.

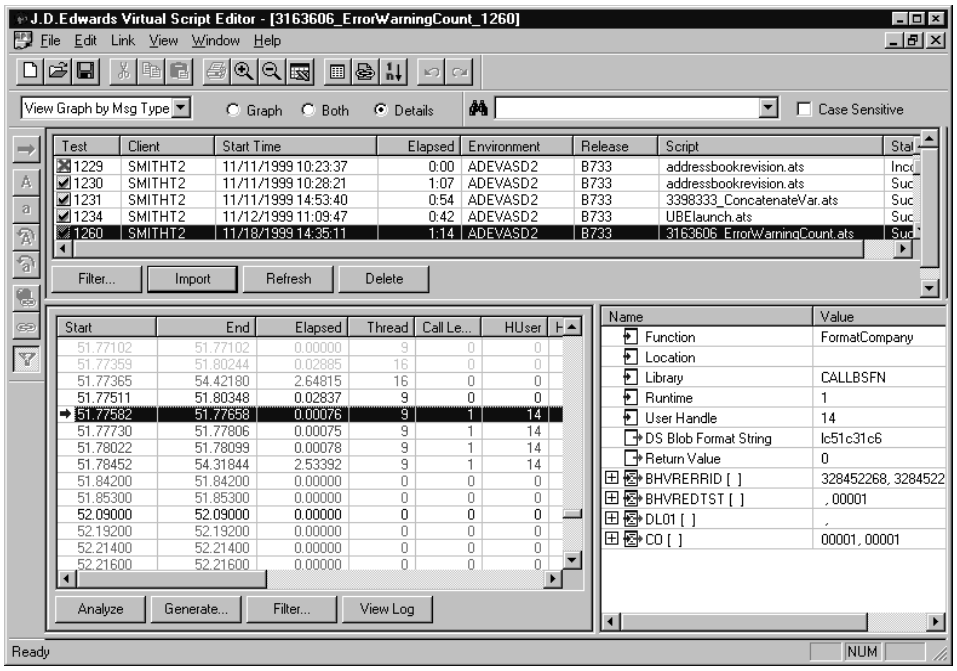


Test	Client	Start Time	Elapsed	Environment	Release	Script
1229	SMITHT2	11/11/1999 10:23:37	0:00	ADEVASD2	B733	addressbookrevision.ats
1230	SMITHT2	11/11/1999 10:28:21	1:07	ADEVASD2	B733	addressbookrevision.ats
1231	SMITHT2	11/11/1999 14:53:40	0:54	ADEVASD2	B733	3398333_ConcatenateVar.ats
1234	SMITHT2	11/12/1999 11:09:47	0:42	ADEVASD2	B733	UBElaunch.ats
1260	SMITHT2	11/18/1999 14:35:11	1:14	ADEVASD2	B733	3163606_ErrorWarningCount.ats

Start	End	Elapsed	Thread	Call Le...	HUser	HRequ...	MsgID	Message
0.00000	0.00000	0.00000	0	0	0	0	6110	MSG: Starting Script Playback [3
0.00000	0.11338	0.11338	1	1	1	1	2024	JDB: OpenTableX (F0005)
0.11498	0.13825	0.02328	1	1	1	1	2028	JDB: SelectKeyed (F0005)
0.13962	0.14215	0.00252	1	1	1	1	2020	JDB: FetchMatchingKey (F0005)
0.14507	0.14547	0.00040	1	1	1	1	2016	JDB: CloseTable (F0005)
! 0.17567	0.23879	0.06312	1	1	0	2	2028	JDB: SelectKeyed (F0082)
➡ 0.23800	0.23800	0.00000	0	0	0	0	150	AUT: Application/form/control [/
! 0.24014	0.36198	0.12184	1	1	0	2	2021	JDB: FetchMatchingKey (F0082)
0.24700	0.24700	0.00000	0	0	0	0	150	AUT: Application/form/control [P
0.25600	0.25600	0.00000	0	0	0	0	6302	ERR: Button Press [&Close] in U
0.26400	0.26400	0.00000	0	0	0	0	111	AUT: Launch: [Application {P03E
0.26800	0.26800	0.00000	0	0	0	0	109	AUT: Type FastPath [TS588301
0.28300	0.28300	0.00000	0	0	0	0	6007	XXX: FastPath 3/G03B11. Appli
! 0.36348	0.40192	0.03844	1	1	0	3	2004	JDB: FetchKeyed (F0083)

3. To view details about an individual event, select the event.

   The Virtual Script Editor displays the details in the event stream detail pane.



Test	Client	Start Time	Elapsed	Environment	Release	Script	Sta
1229	SMITHT2	11/11/1999 10:23:37	0:00	ADEVASD2	B733	addressbookrevision.ats	Inc
1230	SMITHT2	11/11/1999 10:28:21	1:07	ADEVASD2	B733	addressbookrevision.ats	Suc
1231	SMITHT2	11/11/1999 14:53:40	0:54	ADEVASD2	B733	3398333_ConcatenateVar.ats	Suc
1234	SMITHT2	11/12/1999 11:09:47	0:42	ADEVASD2	B733	UBElaunch.ats	Suc
1260	SMITHT2	11/18/1999 14:35:11	1:14	ADEVASD2	B733	3163606_ErrorWarningCount.ats	Suc

Start	End	Elapsed	Thread	Call Le...	HUser
51.77102	51.77102	0.00000	9	0	0
51.77359	51.80244	0.02885	16	0	0
51.77365	54.42180	2.64815	16	0	0
51.77511	51.80348	0.02837	9	0	0
➡ 51.77582	51.77658	0.00076	9	1	14
51.77730	51.77806	0.00075	9	1	14
51.78022	51.78099	0.00078	9	1	14
51.78452	54.31844	2.53392	9	1	14
51.84200	51.84200	0.00000	0	0	0
51.85300	51.85300	0.00000	0	0	0
52.09000	52.09000	0.00000	0	0	0
52.19200	52.19200	0.00000	0	0	0
52.21400	52.21400	0.00000	0	0	0
52.21600	52.21600	0.00000	0	0	0

Name	Value
Function	FormatCompany
Location	
Library	CALLBSFN
Runtime	1
User Handle	14
DS Blob Format String	lc51c31c6
Return Value	0
⊞ BHVRERRID [ ]	328452268, 3284522
⊞ BHVREDTST [ ]	, 00001
⊞ DL01 [ ]	,
⊞ CO [ ]	00001, 00001

4. To view in a horizontal bar graph categories of script playback events or the threads generated during playback, click the Graph option in the toolbar.

5. To choose the method of display for the graph, click the scroll bar button in the toolbar and choose on the following:
   - View Graph by Msg Type
   - View Graph by Thread ID
6. To view both the event stream and the horizontal bar graph, choose the Both option in the toolbar.

Editing the Virtual Script

After you import an event stream into the Virtual Script Editor, you can create a virtual script by completing two primary tasks: adding loops and creating value links. After you finish these tasks, you generate the virtual script. The Virtual Script Editor passes the loop and value-link information, as well as playback information that it stores automatically, to the Virtual Script Player, which runs the virtual script.

When you add loops, you dictate the number of data retrievals that OneWorld Virtual User Tool performs when you run the virtual script. You can limit the number of loops, or you can ensure that the Virtual Script Player loops until no more data is available.

When you create value links, you ensure that data necessary to run the virtual script will flow dynamically between parameters in API calls. For example, you must value-link APIs that use next numbers so that the Virtual Script Player retrieves the appropriate next number during virtual playback. If you fail to value-link the next number parameter in this scenario, the Virtual Script Player passes the same value used in the original script to the API parameter that requires it, which causes a duplicate key error. When you forge a value link, the Virtual Script Editor stores the parameter value in a variable. This ensures that the value will change each time you run the script, preventing duplicate keys and data contention.

The OneWorld Virtual User Tool set also allows you to concatenate virtual scripts into a master script list, using the VSMEditor. Using a master script enables you to test more than one script in a single virtual script playback session.

This chapter details the task steps you complete to edit the virtual script:

- ❑ Using the FIND utility
- ❑ Adding loops
- ❑ Value-linking parameters
- ❑ Linking values in inquiry scripts
- ❑ Linking values in entry scripts
- ❑ Generating the OneWorld Virtual User Tool Script
- ❑ Creating master scripts

Using the FIND Utility

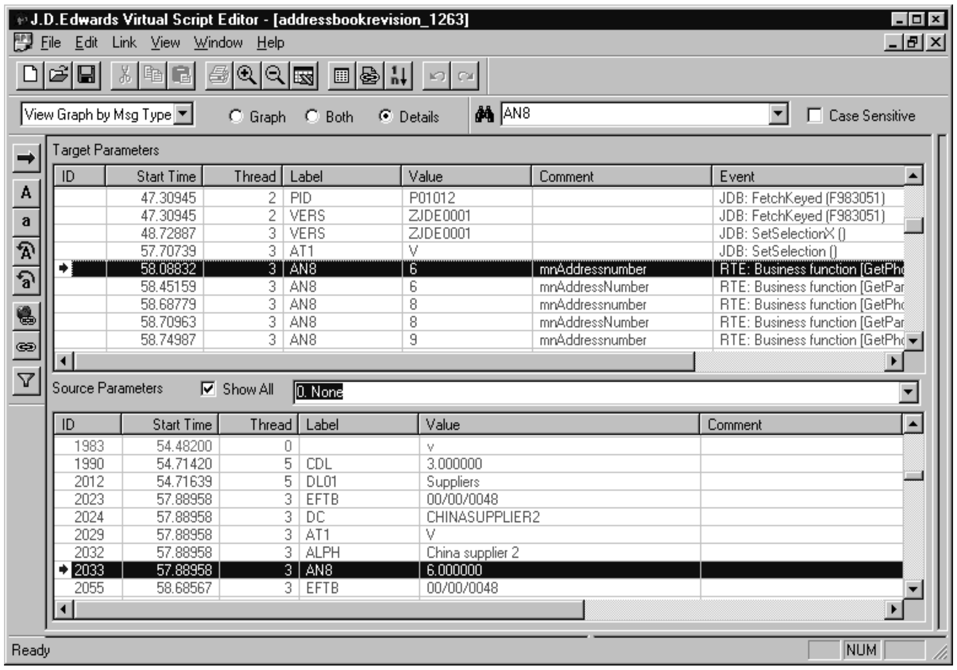
You use the FIND utility in the Virtual Script Editor to search for parameters that you will need to value link to create the OneWorld Virtual User Tool script. The FIND utility consists of a control, to which you enter a search string. When the Virtual Script Editor finds a match, it displays a blue arrow in the event line in the pane that contains the match.

> **Caution:** Make sure you click inside the pane where you want to find information before you use the FIND utility.

You can use the Virtual Script Editor FIND utility to:

- Search for valid value list values to link. Enter a list value to the FIND control.
- Find loops to process. Search for JDB Fetch calls.
- Find data dictionary aliases. Enter a data dictionary alias, such as AN8.

The Virtual Script Editor finds the first parameter with a data dictionary alias that matches your search criterion and indicates it with an arrow.


J.D.Edwards Virtual Script Editor - [addressbookrevision_1263]
File Edit Link View Window Help
View Graph by Msg Type
Graph
Both
Details
AN8
Case Sensitive

Target Parameters

ID	Start Time	Thread	Label	Value	Comment	Event
	47.30945	2	PID	P01012		JDB: FetchKeyed (F983051)
	47.30945	2	VERS	ZJDE0001		JDB: FetchKeyed (F983051)
	48.72887	3	VERS	ZJDE0001		JDB: SetSelectionX ()
	57.70739	3	AT1	V		JDB: SetSelection ()
➔	58.08832	3	AN8	6	mnAddressnumber	RTE: Business function [GetPh
	58.45159	3	AN8	6	mnAddressNumber	RTE: Business function [GetPar
	58.68779	3	AN8	8	mnAddressnumber	RTE: Business function [GetPh
	58.70963	3	AN8	8	mnAddressNumber	RTE: Business function [GetPar
	58.74987	3	AN8	9	mnAddressnumber	RTE: Business function [GetPh

Source Parameters ☑ Show All 0. None

ID	Start Time	Thread	Label	Value	Comment
1983	54.48200	0		v	
1990	54.71420	5	CDL	3.000000	
2012	54.71639	5	DL01	Suppliers	
2023	57.88958	3	EFTB	00/00/0048	
2024	57.88958	3	DC	CHINASUPPLIER2	
2029	57.88958	3	AT1	V	
2032	57.88958	3	ALPH	China supplier 2	
➔ 2033	57.88958	3	AN8	6.000000	
2055	58.68567	3	EFTB	00/00/0048	

Ready NUM

▶ To use the FIND utility

1. Click inside the pane you wish to search.
2. Type a value into the FIND control.
3. Check the Case Sensitive button, if necessary.

4. Press ENTER to run the search.

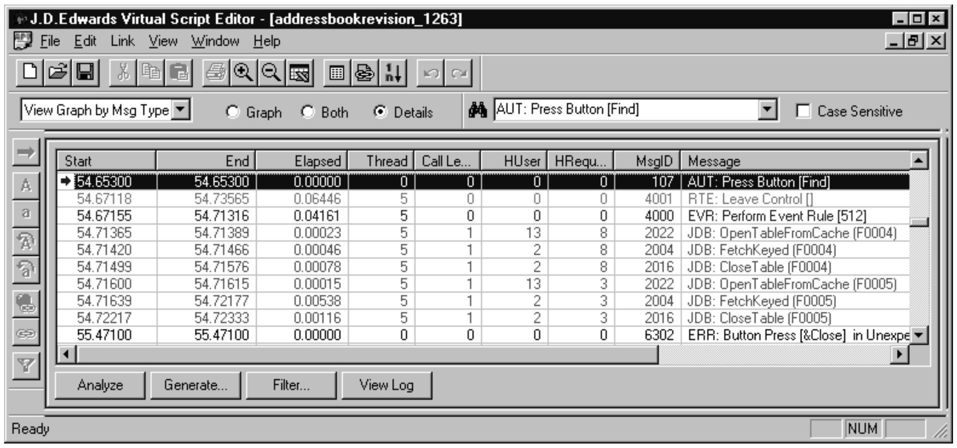
   Note: As you click a button to link or perform another task, you might lose the focus to the pane may. Be sure to reset the focus to the pane you are searching in, if necessary, by clicking inside the pane.

Adding Loops

Loops in OneWorld Virtual User Tool scripts simulate how OneWorld functions when it performs inquiries. Without loops, the Virtual Script Player tries to fetch the same number of records that were retrieved during the original playback, regardless of selection criteria or data available. Loops also allow you to identify and reduce the number of events that must be value linked. Because of this advantage, it is wise to generate loops before performing the value-linking function.

▶ To add a loop

1. Use the FIND utility to search the event stream event pane for AUT: Press Button [Find].



Details [✓]; AUT: Press Button [Find]

Start	End	Elapsed	Thread	Call Le...	HUser	HRequ...	MsgID	Message
54.65300	54.65300	0.00000	0	0	0	0	107	AUT: Press Button [Find]
54.67118	54.73565	0.06446	5	0	0	0	4001	RTE: Leave Control []
54.67155	54.71316	0.04161	5	0	0	0	4000	EVR: Perform Event Rule [512]
54.71365	54.71389	0.00023	5	1	13	8	2022	JDB: OpenTableFromCache (F0004)
54.71420	54.71466	0.00046	5	1	2	8	2004	JDB: FetchKeyed (F0004)
54.71499	54.71576	0.00078	5	1	2	8	2016	JDB: CloseTable (F0004)
54.71600	54.71615	0.00015	5	1	13	3	2022	JDB: OpenTableFromCache (F0005)
54.71639	54.72177	0.00538	5	1	2	3	2004	JDB: FetchKeyed (F0005)
54.72217	54.72333	0.00116	5	1	2	3	2016	JDB: CloseTable (F0005)
55.47100	55.47100	0.00000	0	0	0	0	6302	ERR: Button Press [&Close] in Unexpe

2. From the Find statement in the event stream event pane, move the cursor down the list of events until you find a Fetch statement in the Message column.
3. Right click and choose Add Loop.

   The Dynamic Loop Manager form appears.

4. In the Dynamic Loop Manager, choose one of the following options:
   - Loop until No Data: choose if you want OneWorld Virtual User Tool to exhaust data retrieval

- Loop X times: specify the number of loops you want the script to perform

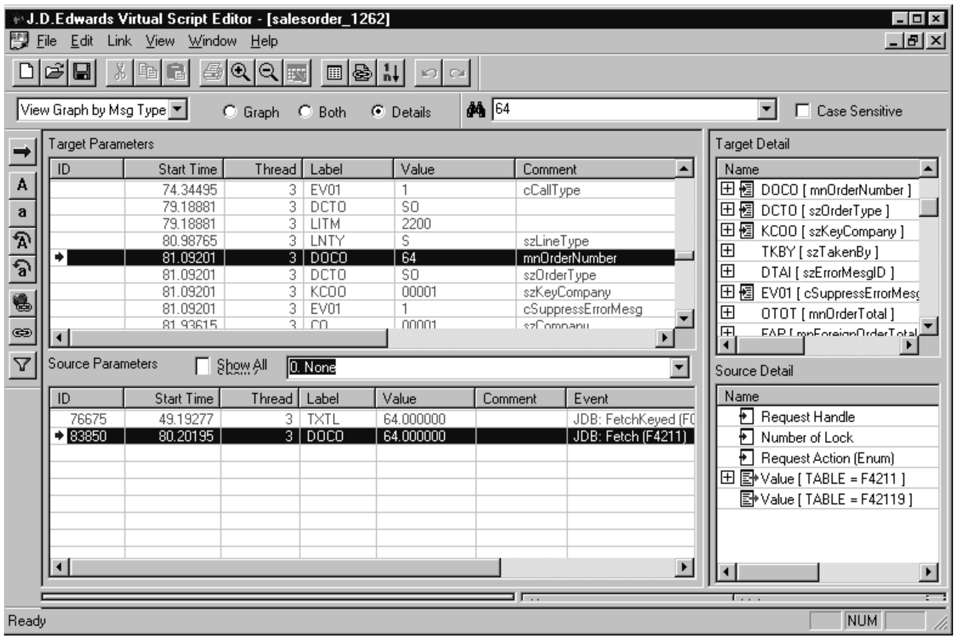
5. Click Apply Loop to add the loop.
6. To undo the loop, find the loop, launch the Dynamic Loop Manager form and click the Undo Loop button.

   Note: OneWorld Virtual User Tool colors events inside a loop light gray in the event stream, source and target panes. You do not need to consider these events when you perform value linking.

Value-Linking Parameters

Value linking provides a conduit for data to flow from function to function within OneWorld. For a OneWorld Virtual User Tool script to accurately simulate OneWorld activities, it must not produce any duplicate key values in the OneWorld database. Therefore, for scripts that enter new data to the database or update existing data, at a minimum, you must value link all next number, job number, and batch number parameter values in the Virtual Script Editor. You can run simple inquiry scripts without any value linking, but these scripts might not accurately simulate OneWorld operations.

The Virtual Script Editor links some values automatically, while you must link others manually. To manually link values, you click the Link Parameters button on the toolbar. The Virtual Script Editor displays the Source Parameters pane and the Target Parameters pane. When you click an API call in the Target Parameters pane, the Virtual Script Editor displays in the Source Parameters pane only those API calls with a parameter that contains a value that matches the target parameter, provided you leave the Show All option unchosen.


J.D.Edwards Virtual Script Editor - [salesorder_1262]
File Edit Link View Window Help
Ready
NUM

View Graph by Msg Type ( ) Graph ( ) Both (●) Details 64 [ ] Case Sensitive

Target Parameters

ID	Start Time	Thread	Label	Value	Comment
	74.34495	3	EV01	1	cCallType
	79.18881	3	DCTO	SO	
	79.18881	3	LITM	2200	
	80.98765	3	LNTY	S	szLineType
➔	81.09201	3	DOCO	64	mnOrderNumber
	81.09201	3	DCTO	SO	szOrderType
	81.09201	3	KCOO	00001	szKeyCompany
	81.09201	3	EV01	1	cSuppressErrorMesg
	81.93615	3	CO	00001	szCompany

Target Detail

Name
DOCO [mnOrderNumber]
DCTO [szOrderType]
KCOO [szKeyCompany]
TKBY [szTakenBy]
DTAI [szErrorMesgID]
EV01 [cSuppressErrorMesg
OTOT [mnOrderTotal]
FAP [mnForeignOrderTotal

Source Parameters [ ] Show All 0. None

ID	Start Time	Thread	Label	Value	Comment	Event
76675	49.19277	3	TXTL	64.000000		JDB: FetchKeyed (F(
➔ 83850	80.20195	3	DOCO	64.000000		JDB: Fetch (F4211)

Source Detail

Name
Request Handle
Number of Lock
Request Action (Enum)
Value [TABLE = F4211]
Value [TABLE = F42119]

To run scripts accurately, you should always value link the following types of parameters:

- Parameters passing the name of the machine where you ran the original OneWorld Scripting Tool script
- Parameters referencing the date the original OneWorld Scripting Tool script ran
- Parameters passing Next Numbers or serialized values (possibly labels of DOC, JOBS, MATH06, PYID, ICU)
- Parameters that use valid value list data. Linking these parameters ensures that the Virtual Script Player will use the .atd directory, where you store valid values list data, as the source from which to retrieve data during virtual script playback
- Labels containing the date the script ran

Note: You can use the FIND utility to quickly find functions containing data to be linked. Click the column header to reorder the table (usually by label, value, or ID) to group like information.

▶ To value-link parameters

1. Import a OneWorld Scripting Tool script into the Virtual Script Editor and click the Link Parameters button in the toolbar.

   The Source Parameters pane and Target Parameters pane appear.

2. In the Target Parameters pane, highlight a target parameter line item. The source parameters for that target will be displayed in the Source Parameters pane.

   Note: Do not choose the Show All option in the Source Parameters pane because doing so causes the Virtual Script Editor to display all the API calls in the pane.

3. To link a single parameter line item, select it and click the Link button.
4. To link all items in the script that match the source, target, label, and value parameters, select a representative parameter line item in the source pane and click the Link All button.

   Note: Some parameters in the Target Parameters pane do not have a value from a source parameter. You can mark these as “Literal” using the Mark Literal Button. If you do not wish to see the parameters you have marked as Literal, click Link in the menu bar and choose Filter Literals.

Linking Values in Inquiry Scripts

You are not required to forge value links between parameters in inquiry scripts, because an inquiry does not change or update any data in the system. However, you should value-link parameters that contain valid values list data to ensure that the data changes during playback of the virtual script.

If your script contains valid values data, you can run the virtual script, change the data and run it again to extend your stress testing. You can change the data in the list without creating new value links. During virtual script playback, the Virtual Script Editor passes the new valid values list data to the Virtual Script Player for use in the appropriate parameters.

▶ To link values in inquiry scripts

1. In the OneWorld Virtual User Tool Script Editor form, add any required loops to the OneWorld Virtual User Tool Script.
2. Find valid values list data in the event stream.
3. Link all source parameters containing valid values list data to the appropriate target parameters.
4. Document the data dictionary aliases that the Virtual Script Editor links.

   Note: You find data dictionary aliases in the Label column of the Source Parameters pane and the Target Parameters pane.

Linking Values in Entry Scripts

You are required to link values in entry scripts before you generate a virtual script, because such scripts change or update system data. Value linking ensures that Virtual Script Player can pass values between parameters and that key parameter values change during virtual script playback, preventing record duplication.

▶ To link values in entry scripts

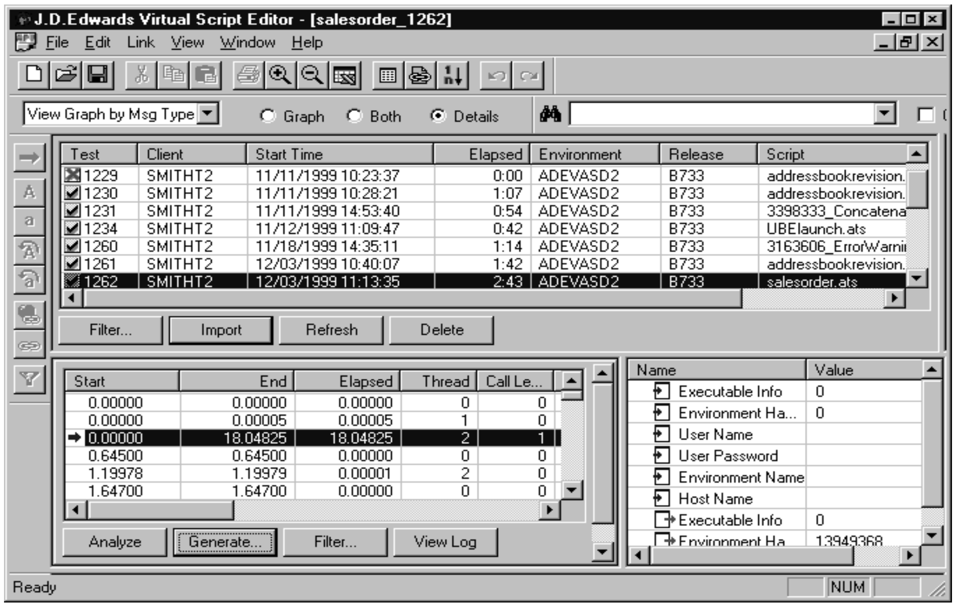
1. In the Virtual Script Editor form, add any required loops to the OneWorld Virtual User Tool Script.
2. Find and link any parameters that pass the machine name on which the OneWorld Scripting Tool script originally ran (these might be marked with CTID or MKEY data dictionary aliases or labels).
3. Find and link parameters that pass the date that the OneWorld Scripting Tool script originally ran.

4. Find and link parameters that pass Next Numbers or serialized values (possibly data dictionary aliases of DOC, JOBS, MATH06, PYID, ICU).
5. Find and link parameters that pass valid values list values.
6. Document the data dictionary aliases that the Virtual Script Editor links.

Generating the OneWorld Virtual User Tool Script

When you press the Generate button, the Virtual Script Editor produces a virtual script, which the Virtual Script Player uses to simulate playback. A Script Log form appears following generation, summarizing the number of lines in the script and the number of errors, if any. You must generate an error-free script before you attempt to run it. See *OneWorld Virtual User Tool Troubleshooting Tips and Techniques* for a discussion of identifying and eliminating errors.

To generate the OneWorld Virtual User Tool script for playback using the Virtual Runner program, the script you want to generate must be open in the script list pane.



Test	Client	Start Time	Elapsed	Environment	Release	Script
1229	SMITHT2	11/11/1999 10:23:37	0:00	ADEVASD2	B733	addressbookrevision.
1230	SMITHT2	11/11/1999 10:28:21	1:07	ADEVASD2	B733	addressbookrevision.
1231	SMITHT2	11/11/1999 14:53:40	0:54	ADEVASD2	B733	3398333_Concatena
1234	SMITHT2	11/12/1999 11:09:47	0:42	ADEVASD2	B733	UBElaunch.ats
1260	SMITHT2	11/18/1999 14:35:11	1:14	ADEVASD2	B733	3163606_ErrorWarni
1261	SMITHT2	12/03/1999 10:40:07	1:42	ADEVASD2	B733	addressbookrevision.
1262	SMITHT2	12/03/1999 11:13:35	2:43	ADEVASD2	B733	salesorder.ats

Start	End	Elapsed	Thread	Call Le...
0.00000	0.00000	0.00000	0	0
0.00000	0.00005	0.00005	1	0
0.00000	18.04825	18.04825	2	1
0.64500	0.64500	0.00000	0	0
1.19978	1.19979	0.00001	2	0
1.64700	1.64700	0.00000	0	0

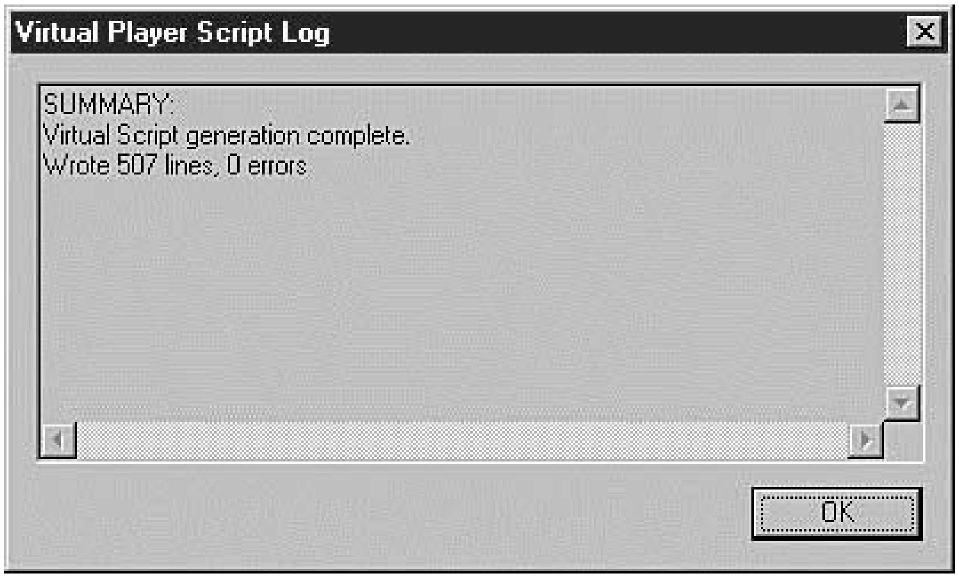
Name	Value
Executable Info	0
Environment Ha...	0
User Name	
User Password	
Environment Name	
Host Name	
Executable Info	0
Environment Ha	13949368

After OneWorld Virtual User Tool generates the script, the Virtual Player Script Log form appears and displays information about the script generation including the status (complete or incomplete), the number of lines and the number of errors. If the Script Log form indicates that errors occurred during generation, you must investigate the error summaries that appear in the form and attempt to correct them by editing and rerunning the OneWorld Scripting Tool script, and then repeating the steps for creating a virtual script.

▶ To generate the OneWorld Virtual User Tool script

1. In the event stream event pane, click the Generate button.
2. Assign a file name and location to which you want to save the generated script and click OK to begin script generation.

   After script generation completes, the Virtual Script Editor displays the Virtual Player Script Log form. If the summary reports any errors, you can not use the script for virtual playback.



```
SUMMARY:
Virtual Script generation complete.
Wrote 507 lines, 0 errors
```

3. Click OK to close the Virtual Player Script Log form.

   Note: Once you generate a virtual script, it is static. Any script changes you make in OneWorld Scripting Tool will require re-editing and re-generation in OneWorld Virtual User Tool. Thus, careful documentation of the editing process is critical to the production of repeatable results.

Creating Master Scripts

Using the VSMEditor tool, you can concatentate OneWorld Virtual User Tool scripts into a single master script. Concatenation gives you another testing option in that you can run test series of unrelated scripts.

▶ To create a master script

1. Start the VSMEditor.
2. In the All Virtual Scripts list box, choose the script files that you want to include in the master script.

3. When you have chosen all the virtual script text files that you want, click the Add button.

   VSMEditor adds the script files to the Master Script list box.

4. Manipulate the list in the Master Script list box by using the buttons adjacent to the box to remove script files or to change their order.
5. When you have settled on the content and order of the master script, click the Save Master Script button.

   The VSMEditor saves the master script as a .vsm file. The file includes:

   - Master script version
   - Checksum value to verify file integrity
   - RandomStart parameter: value of 1 means that the first script to run is chosen randomly
   - List of OneWorld Virtual User Tool Script files

Executing OneWorld Virtual User Tool Scripts

Running Virtual Scripts

After you generate one or more OneWorld Virtual User Tool scripts, you are ready to execute playback to simulate many users running OneWorld processes. If you want to simulate several users on a single workstation, you can launch the script either from a command line or from Virtual Runner.

Using Mercury Interactive's LoadRunner tool, you can also launch one or more OneWorld Virtual User Tool script playback sessions on more than one workstation. LoadRunner manages the playback sessions. Using LoadRunner as your script playback manager allows you to more accurately simulate the actual stress that users in a business environment might impose on the system.

This section discusses the two methods of running OneWorld Virtual User Tool script playback:

- ❏ Running virtual scripts from a single workstation
- ❏ Launching and managing multiple script playback

Running Virtual Scripts from a Single Workstation

You can launch the Virtual Script Player from a command line on a single machine or from Virtual Runner, which manages virtual script playback, in order to simulate more than one user on a single workstation. The Virtual Script Player accesses the .vsx file that you create when you generate a virtual script on the Virtual Script Editor. After you run the script, you check the log files for errors.

This chapter discusses the two ways to run virtual scripts from a single workstation:

- ❑ Running virtual scripts from a command line
- ❑ Running virtual scripts using Virtual Runner

Running Virtual Scripts from a Command Line

The Virtual Script Player accesses the .vsx file generated by the Virtual Script Editor. You can launch the Virtual Script Player from a command line on a single machine or from a LoadRunner controller when you want to run virtual scripts on more than one workstation.

The command line must have entries specifying the user, the environment, and the script name. The following table summarizes the required entries to the command line:

Command Line Abbreviation	Meaning	Sample Entry
–u	OneWorld user ID	–u JDE
–p	OneWorld user password	–p JDE
–e	Environment	–e PRD733
–s	Script Name + number of script iterations to run	–s Script1.vsx

After you complete the virtual script run, you can check the log file for error messages. You set the path to the log file in the PATHS section of the OneWorld Virtual User Tool initialization file.

When playback concludes, the Virtual Script Player.exe task disappears from the Task Manager window and a log in the \OneWorld Scripting Tool\VAP_Logs directory displays any errors encountered. You can change the directory location can be changed in the vap.ini file.

See Also

- *Virtual Script Player Initialization File Parameters*

▶ To run a virtual script from a command line

1. From the Start menu in Windows, choose Command Prompt from the Programs menu.
2. At the C: prompt, type the Virtual Script Player command with appropriate parameters: for example, Virtual Script Player -u JDE -p JDE -e PRD733 -s5 script1.vsx
3. Press Enter to run the command.
4. Check the progress of the execution by typing Ctrl-Alt-Del to access the Windows Task Manager.

   Note: The Processes tab displays the executable (Virtual Script Player.exe) and CPU activity associated with it. Otherwise, there is no indication on the screen of activity.

5. Search the OneWorld Virtual User Tool log for errors by clicking the Search menu, choosing Find, and searching on the keyword "error."

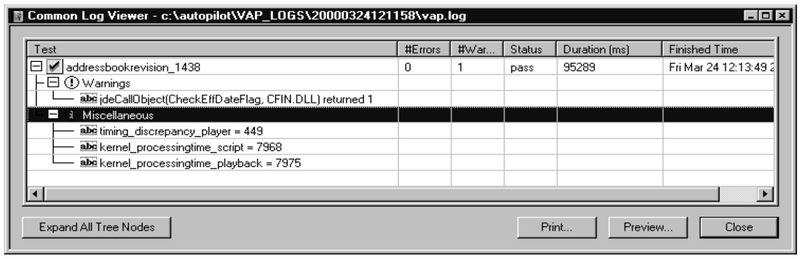
   Note: If errors exist, see *Debugging OneWorld Virtual User Tool Scripts.*

6. If the script contains valid values list data, change the data and play the script again.

Running Virtual Scripts Using Virtual Runner

The Virtual Runner program allows you to manage the playback of virtual scripts. You use it to specify the script, the number of player sessions, and the number of iterations you want to run in each session. You also specify the OneWorld environment against which you want to run the sessions.

After Virtual Runner completes running the sessions, it displays the status of each test. You can view log information about a test by clicking the Log button.


Common Log Viewer - c:\autopilot\VAP_LOGS\20000324121158\vap.log
Expand All Tree Nodes
Print...
Preview...
Close

Test	#Errors	#War...	Status	Duration (ms)	Finished Time
addressbookrevision_1438	0	1	pass	95289	Fri Mar 24 12:13:49 2
Warnings					
jdeCallObject(CheckEffDateFlag, CFIN.DLL) returned 1					
Miscellaneous					
timing_discrepancy_player = 449					
kernel_processingtime_script = 7968					
kernel_processingtime_playback = 7975					

You can expand the nodes in the Common Log Viewer form to see any error or warning messages that might have occurred during the Virtual Runner session. In addition to error and warning messages the form displays:

- Name of the test
- Number of errors
- Number of warnings
- Status of the test
- Duration of the test
- Time of completion

You use the Virtual Runner log in conjunction with the OneWorld and more detailed VAP logs to help debug failed sessions.

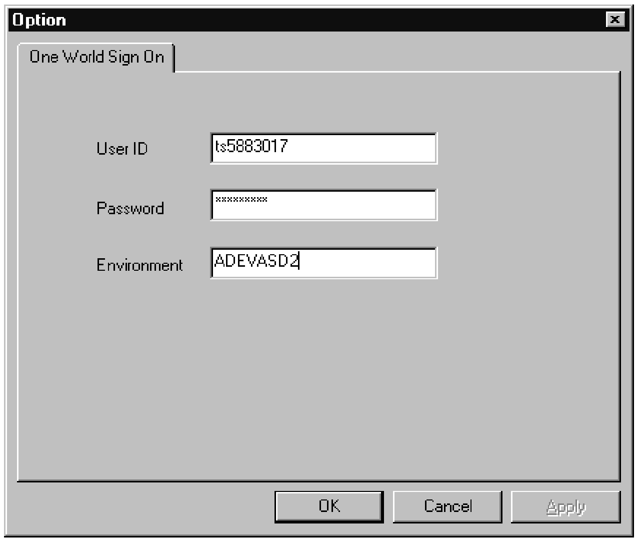
▶ To run virtual scripts using Virtual Runner

1. On your desktop or in the OneWorld Scripting Tool directory, double click the Virtual Runner icon.

   The Virtual Runner window appears.

2. Click the Option button located on the Virtual Runner toolbar.

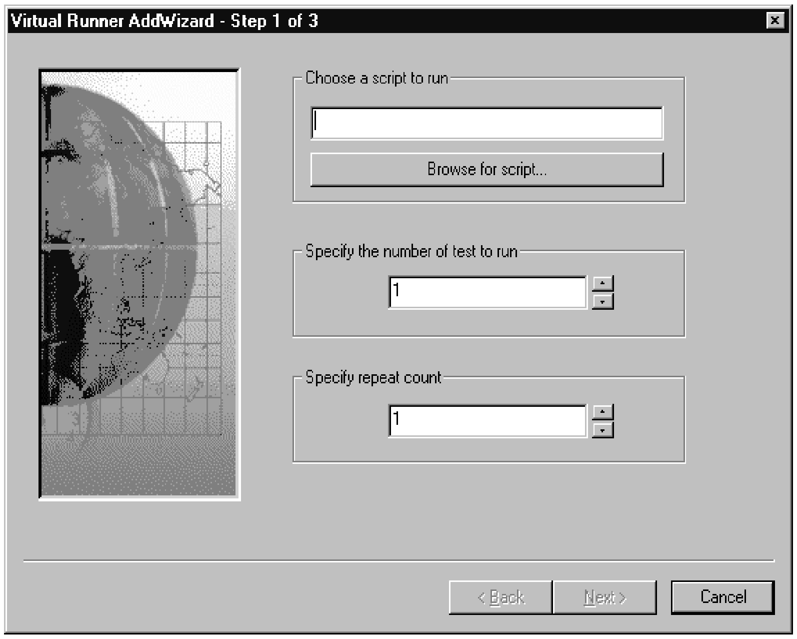
   The Option window appears.



User ID: ts5883017
Password: *********
Environment: ADEVASD2

3. Enter your User ID, password, and the OneWorld environment against which you want to run the test, and click OK.
4. Click the Wizard button.

The Virtual Runner Add Wizard - Step 1 of 3 window appears.

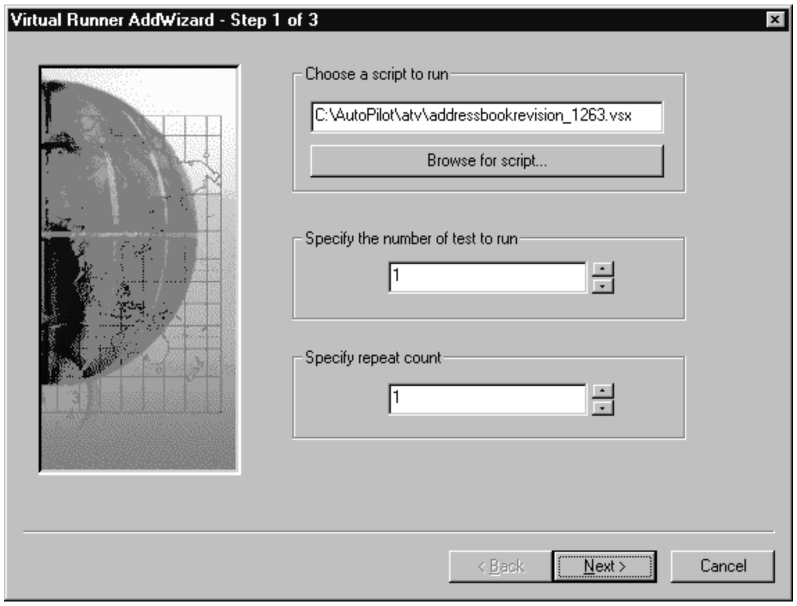


5. Click the Browse for script button to choose a script you want to run.

   The Choose a Virtual Player Script to run window appears.

6. Highlight the script you want and click the Open button.

   The name of the script appears in the Choose a script to run field.



Choose a script to run: C:\AutoPilot\atv\addressbookrevision_1263.vsx

Specify the number of test to run: 1

Specify repeat count: 1

7. Specify the number of number of player sessions to run on the workstation.

   Example: enter 4 to run four scripts simultaneously.

8. Specify the number of virtual script session iterations you wish to run.

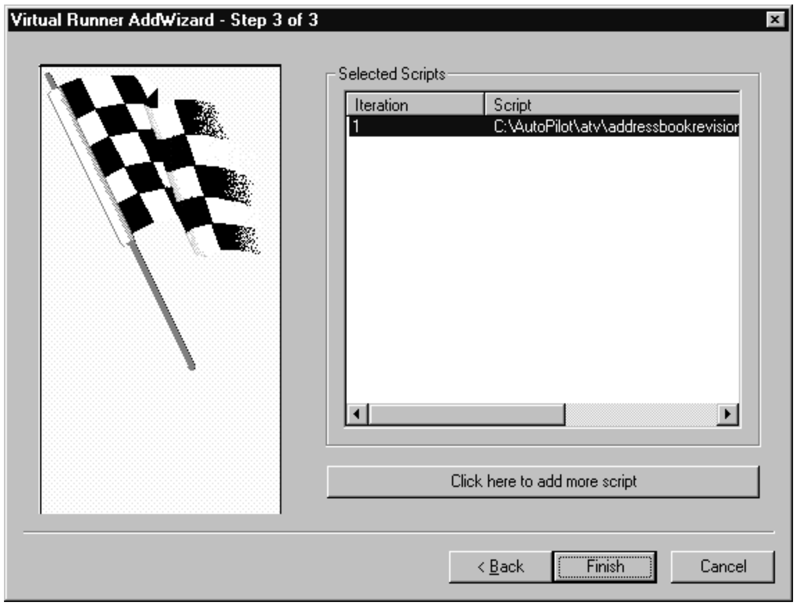
   Example: enter 5 to run the script five iterations sequentially.

9. Click the Next button.

   The Virtual Runner Add Wizard - Step 2 of 3 window appears. If you entered information into the Option window, then the Wizard pulls that information into this window.

10. If you did not enter information into the Option window, then enter your User ID, Password, and Environment
11. Click Next.

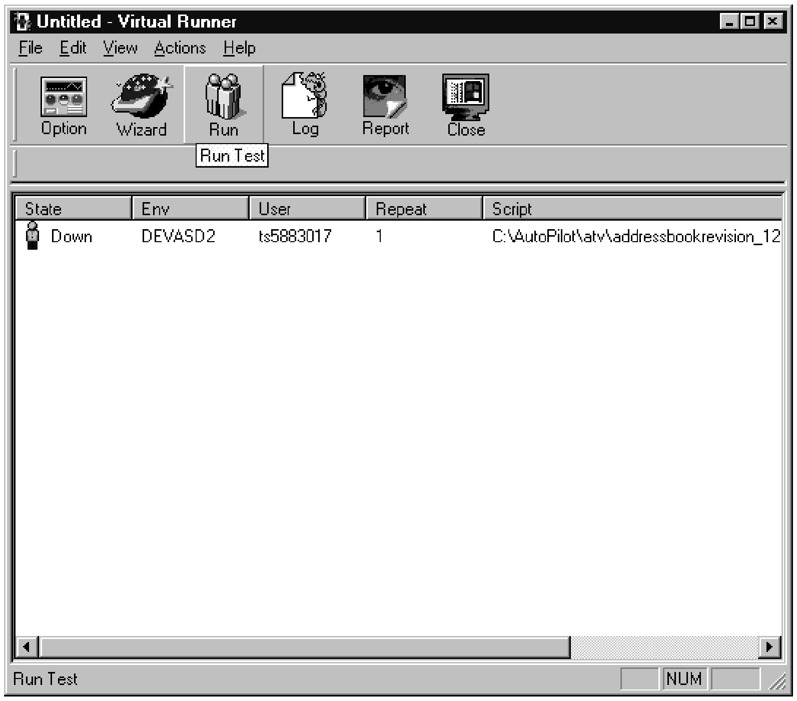
    The Virtual Runner Add Wizard - Step 3 of 3 window appears.


Virtual Runner AddWizard - Step 3 of 3
Selected Scripts
Click here to add more script
< Back
Finish
Cancel

Iteration	Script
1	C:\AutoPilot\atv\addressbookrevision

12. If you want to add another script, click the button to add more scripts and repeat Steps 5 to 11.
13. If you do not want to add additional scripts, click the Finish button to return to the Virtual Runner form.

    Virtual Runner displays the script or scripts that you selected and activates the Run button.



State	Env	User	Repeat	Script
Down	DEVASD2	ts5883017	1	C:\AutoPilot\atv\addressbookrevision_12

14. Click the Run button to begin script processing.

    The main Virtual Runner screen displays the message Starting Up, indicating the processing of the scripts has begun. The main Virtual Runner screen displays the message Running when Virtual Runner is processing the script or scripts. If the scripts successfully run, the screen displays the message Success. You are now ready to review the log file. If processing is not successful, a red failure message is displayed.

15. Click the Log button and click Yes to view the current test log.

    The Common Log Viewer screen appears.

16. Check the log for error and warning messages.
17. If the script contains valid values list data, change the data and play the script again.

Launching and Managing Multiple Script Playback

LoadRunner allows you to set up multiple workstations, each representing multiple users, from which you can launch playback sessions to simulate actual user load on the system. You provide LoadRunner with selected rendezvous points and transactions, which LoadRunner then reports to its controller. LoadRunner gathers and stores the results of each run. The LoadRunner controller workstation must have network connection to all of the workstations that are involved in the test and the controller must run Windows NT.

The following preparations set up a OneWorld Virtual User Tool test session using LoadRunner. Refer to the LoadRunner documentation for setup and installation information.

- ❑ Defining a script
- ❑ Defining the host machine
- ❑ Defining virtual users
- ❑ Setting rendezvous points
- ❑ Gathering LoadRunner results
- ❑ Running virtual playback from the LoadRunner controller
- ❑ Analyzing LoadRunner test results

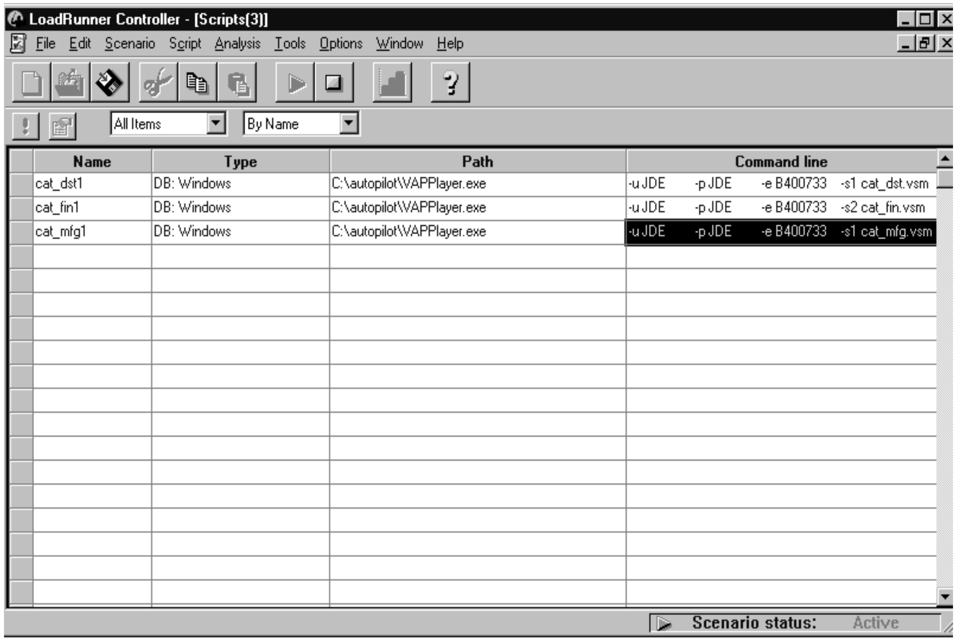
Defining a Script

You define the script that you want to play back so that LoadRunner can locate the Virtual Script Player and pass on to the Player the necessary script command line.

▶ To define a script

1. From the Start menu in Windows, choose Programs and open LoadRunner.
2. From the Window menu of the LoadRunner form, choose Scripts.
3. To define the scripts you will be using, make entries in the first empty line of the grid under the following column names:

- Name (of the script)
- Path (where the Virtual Script Player executable is found)
- Command Line (to be passed to the Virtual Script Player)


LoadRunner Controller - [Scripts(3)]
File Edit Scenario Script Analysis Tools Options Window Help
All Items
By Name
Scenario status: Active

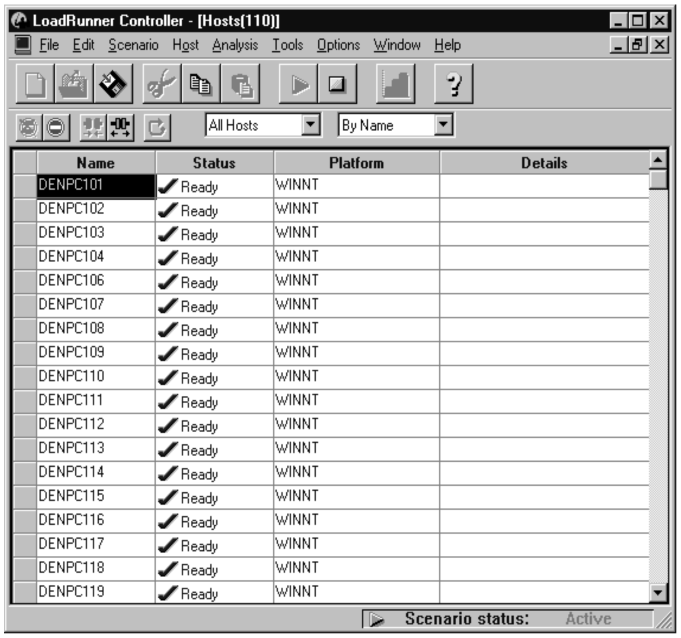
Name	Type	Path	Command line
cat_dst1	DB: Windows	C:\autopilot\VAPPlayer.exe	-u JDE -p JDE -e B400733 -s1 cat_dst.vsm
cat_fin1	DB: Windows	C:\autopilot\VAPPlayer.exe	-u JDE -p JDE -e B400733 -s2 cat_fin.vsm
cat_mfg1	DB: Windows	C:\autopilot\VAPPlayer.exe	-u JDE -p JDE -e B400733 -s1 cat_mfg.vsm

Defining the Host Machine

After you have defined the scripts, you define the host machine for the LoadRunner test, as well as the platform on which the test ran.

▶ To define the host machine

1. From the Window menu, choose Hosts.
2. To define the host machines, make entries in the first empty line of the grid under the following column names:
   - Name (of the machine)
   - Type (of platform)


LoadRunner Controller - [Hosts(110)]
File Edit Scenario Host Analysis Tools Options Window Help
All Hosts
By Name
Scenario status: Active

Name	Status	Platform	Details
DENPC101	✓ Ready	WINNT	
DENPC102	✓ Ready	WINNT	
DENPC103	✓ Ready	WINNT	
DENPC104	✓ Ready	WINNT	
DENPC106	✓ Ready	WINNT	
DENPC107	✓ Ready	WINNT	
DENPC108	✓ Ready	WINNT	
DENPC109	✓ Ready	WINNT	
DENPC110	✓ Ready	WINNT	
DENPC111	✓ Ready	WINNT	
DENPC112	✓ Ready	WINNT	
DENPC113	✓ Ready	WINNT	
DENPC114	✓ Ready	WINNT	
DENPC115	✓ Ready	WINNT	
DENPC116	✓ Ready	WINNT	
DENPC117	✓ Ready	WINNT	
DENPC118	✓ Ready	WINNT	
DENPC119	✓ Ready	WINNT	

Defining Virtual Users

After you define the script and the host machine, you define the virtual users who created the scripts that you want to run. You can define users individually or you can define a group as the virtual user.

▶ To define the virtual users

1. From the Window menu, choose Vusers.

   Note: Remember that each workstation can represent multiple users. You can define users individually or in groups.

2. To define users individually, in the Host field, enter the name of the host. In the Script field, enter the name of the script.

   The program automatically numbers the user for that host and automatically makes entries to the Status and Type fields.

3. To define groups, in the Group field, enter the name of the group.
4. To define users within the Group, choose a group, and add users by entering the host name and script name.

   Note: User number, Status, and Type are automatically entered.

Setting Rendezvous Points

You set the rendezvous point that defines for LoadRunner the time at which all virtual scripts pause before the tool releases them for virtual playback.

► **To set rendezvous points**

1. From the Window menu, choose Rendezvous.

   Note: The Members column indicates the number of users you have defined for your test scenario. The rendezvous points (Name field) are set in the OneWorld Virtual User Tool and should load automatically. A copy the Virtual Script Player executable must exist on the LoadRunner Controller workstation.

2. On the Rendezvous form, adjust the default timeout to a time period that will allow all virtual users to reach the rendezvous point before automatically releasing. Set the arrival policy to be higher than the number of users you are testing.

Gathering LoadRunner Results

The LoadRunner results directory typically has the following structure:

- VAPI (the test directory)
- User Name (from those defined in the Users window)
- Session Number Output.txt (the re-routed VAP_log from the client workstation)

► **To gather LoadRunner results**

1. From the Scenario menu of the LoadRunner form, choose Results File.
2. Enter the name you want to give the Results File.
3. Save the results.

Running Virtual Playback from the LoadRunner Controller

After you have prepared virtual script playback, you are ready to run the test from the LoadRunner controller.

▶ To run virtual playback from the LoadRunner controller

1. From the Hosts form in LoadRunner, click the Connect button to initialize all workstations in the test.

   Note: Use the Control or Shift key to choose all or some of the available workstations.

2. In the Vusers form, choose the group or users you want to include in the test and click the Initialize icon.

   The status changes from *Down* to *Ready* when the selected workstations are initialized. The Ready field in the upper panel identifies how many users in a group have been initialized.

3. When the selected workstations are at Ready status, click the Run icon.

   The Rendezvous form displays. Each user who reaches the rendezvous point is assigned an arrival time in the right panel.

4. When all selected users reach the rendezvous point, click the Release icon to begin the test.

   The Rendezvous form closes and the Vusers screen again displays the Status field, this time changed from *Ready* to *Running*. When the test is complete, the Rendezvous form appears.

5. Click the Release icon again to end the test run.

Analyzing LoadRunner Test Results

After you have run the test, the results are ready for analysis. To ensure that the scripts ran successfully, use the Windows Explorer FIND utility to search for errors.

▶ To analyze LoadRunner test results

1. From the Windows Explorer menu, choose the following items:
   - Tools
   - Find
   - Files or Folders
2. Enter “With error” in the Containing text: field.
3. Enter the Load Runner results directory in the Look In: field.
4. Click Find Now to find errors in the output.txt of all scripts that ran.

If no files return, all scripts ran successfully.

5. Review the files that were displayed to determine why they had errors.
6. From the LoadRunner form, choose Analysis to review results in graph form or in report form.

Special Considerations for Simulated Playback

Special Considerations for Simulated Playback

The OneWorld Virtual User Tool solves several simulated playback problems. All of the problems in one way or another revolve around the tool's ability to simulate accurately the workings of the OneWorld run-time engine. The Virtual Script Editor and the Virtual Script Player work together, with the Virtual Script Editor storing key playback information and passing it to the Virtual Script Player, which in turn uses the information to assume the role of the OneWorld run-time engine. This section explains important simulated playback problems and the ways that OneWorld Virtual User Tool resolves them:

- ❑ Playback timing
- ❑ Call level
- ❑ Synchronous and asynchronous API calls
- ❑ Think times

Playback Timing

To accurately simulate OneWorld activities, OneWorld Virtual User Tool must keep script events synchronized during playback. This presents a challenge because OneWorld Virtual User Tool attempts to simulate multiple users who are stressing the server and the network, while the data upon which OneWorld Virtual User Tool scripts is based is captured from a single user's script playback. This means that event start times and duration might change significantly during a virtual script playback session.

To meet this challenge, OneWorld Virtual User Tool must solve two separate problems.

- Manage changes in the duration of individual API calls and the lengths of time between these calls within a single thread. This involves accurately simulating the process time that OneWorld requires as the run-time engine handles user load.
- Handle timing differences that might affect interthread dependencies. These interdependencies occur, when, for example, an API call in one thread has a data dependency on an API call in another thread.

This chapter discusses OneWorld Virtual User Tool's approach to handling the following playback timing issues:

❑ API Playback timing

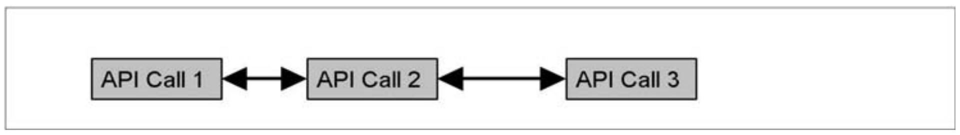
❑ Interthread timing

API Playback Timing

Simulating the timing of OneWorld processing during playback represents a key challenge to the efficacy of the OneWorld Virtual User Tool solution. The goal of OneWorld Virtual User Tool is to accurately simulate the stress that OneWorld users place on the server and on the network. However, the AutoPilot script, which contains the data upon which the OneWorld Virtual User Tool script is built, is not designed to create this stress. The time intervals between events in the AutoPilot script reflect the running of a single script against the OneWorld run-time engine. When you run a OneWorld Virtual User Tool script, the duration of events, and therefore the time intervals between those events will likely change, due to the server and network stress that the script is trying to simulate. The CPU power and memory capability of individual workstations can also affect the playback timing of OneWorld Virtual User Tool scripts.

AutoPilot provides the base for the Virtual Script Player to time the execution of events during virtual script playback. When AutoPilot executes a script in OneWorld, it captures each kernel function call, and it captures the start time and duration of each API call. Therefore, the script contains the gaps of time between each call, which occur as OneWorld carries out other processes. The API calls within a thread might be represented as blocks of time of various lengths with intervening spatial gaps that symbolize the time duration between each call:

OneWorld Scripting Tool Script Capture

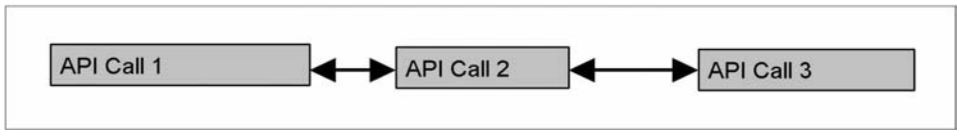


Preserving during data capture these chronological gaps provides the basis for simulating playback by many users, a situation that is likely to increase the length of time that is required to execute the same API calls.

For example, suppose that virtual playback on a single workstation simulates 10 users utilizing a server that is not as powerful as the server that was in use when the AutoPilot script playback session originally occurred. In this scenario, the duration of API call is likely to lengthen. This could cause one API call to overlap another, halting playback.

However, since the event stream has preserved the *intervals* between each call, virtual playback can proceed, regardless of the duration of any or all of the calls within a thread:

Stressed Playback in OW Virtual User Tool

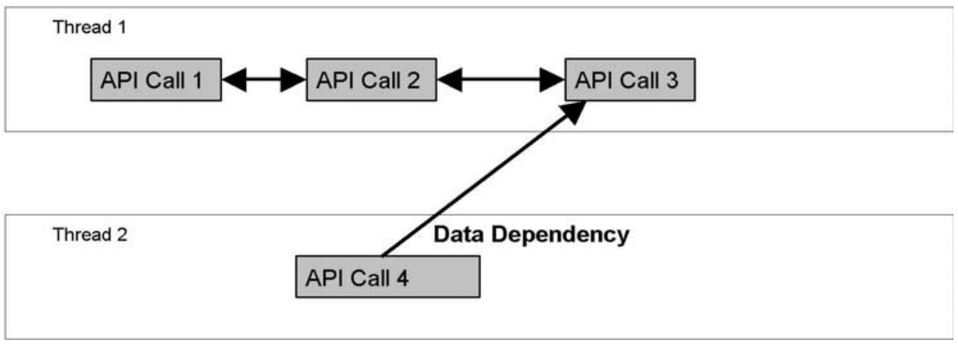


AutoPilot's ability to record the duration and length of time between API calls is also important because it gives you a tool to accurately determine the number of virtual users who can be simulated on a single workstation. For example, lengthy API calls might indicate an underpowered server, a workstation lacking the CPU and memory capability to handle the number of virtual user sessions that you desire, or an application bug. In each of these instances, you would likely scale back the number of users you wish to simulate in a OneWorld Virtual User Tool playback session.

Interthread Timing

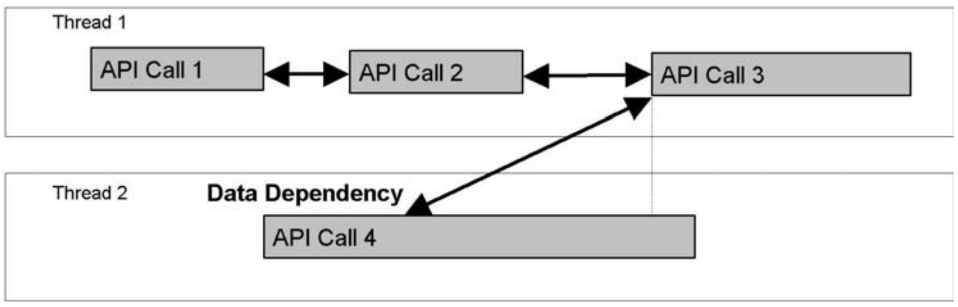
The Virtual Script Editor also plays a role in handling script playback timing so that OneWorld Virtual User Tool can simulate a stressed OneWorld environment. The OneWorld run-time engine, might, for example, spawn a thread that contains an API call with a data dependency on another call, which might, in turn, exist in a separate, asynchronously running thread. In this scenario, the OneWorld run-time engine handles the processing chores by noting that one API must finish before the data-dependent API can begin.

OW Scripting Tool Script Capture



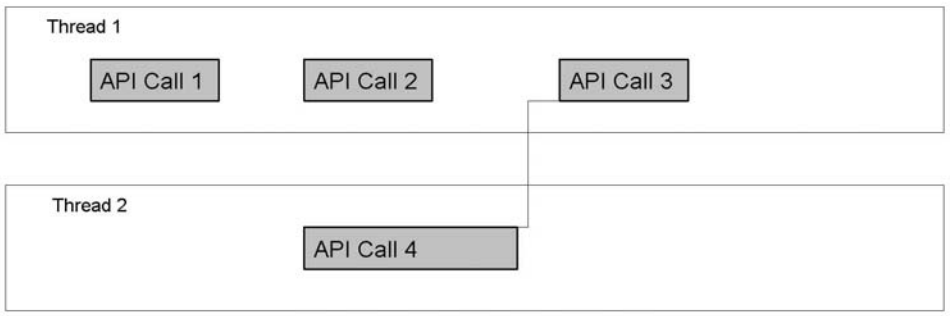
However, in a stressed environment, the duration of API calls might lengthen unpredictably. This might result in a data-dependent API call in one thread starting before the API upon which it depends has finished.

Stressed Playback in OW Virtual User Tool



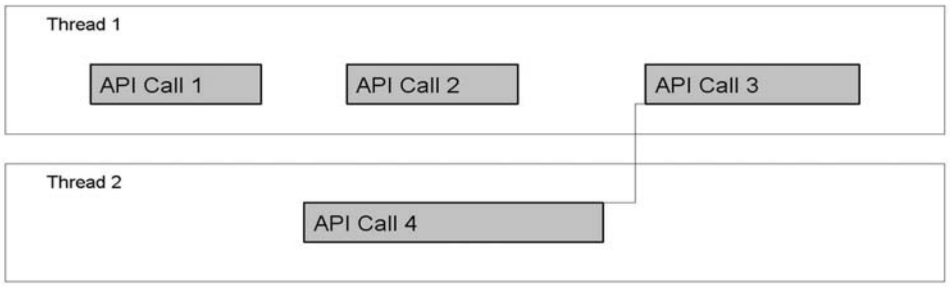
To deal with this potential problem, the Virtual Script Editor notes the data dependency when you forge value links between two API calls and preserves the timing interval between the calls.

OW Scripting Tool Script Capture



When you run the virtual script, the Virtual Script Player increases, on the fly, the interval between APIs in one thread so that an API in one thread has time to complete before a data-dependent API in a thread running asynchronously to it is called.

Stressed Playback in OW Virtual User Tool



In this way, OneWorld Virtual User Tool preserves the necessary time interval that existed between the data-dependent calls when you originally ran the script.

Clearly, the Virtual Script Player, in this scenario, manipulates the time interval between API calls in the first thread. However, the manipulation represents an attempt to fairly simulate what OneWorld does in reality. The OneWorld run-time engine functions as the brain that manages data-dependent APIs so that they can execute without breaking the system. It is therefore appropriate that the Virtual Script Player, in assuming the role of the run-time engine, simulate the run-time engine's responsibility, for example, the delay of one API's completion based on its logical relationship to another API.

Call Level

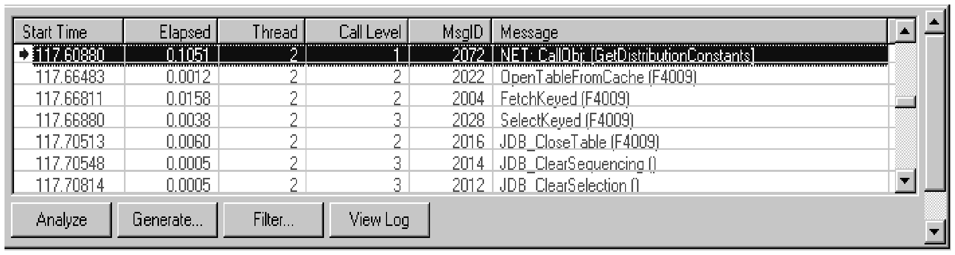
Some API calls invoke other API calls automatically within the same thread in OneWorld. Call level refers to an API call's position in the sequence of calls. For example, an EditLine business function might invoke a JDB Fetch call for a company number. In this example, call level of the EditLine business function is 1, call level of the JDB Fetch call is 2, and the AutoPilot event stream records the two separate API calls.

However, while the OneWorld run-time engine handles two separate API calls in this example, the processing occurs seamlessly: the second call follows immediately from the first without additional input from the user. For this reason, a OneWorld Virtual User Tool script contains only those API calls with a depth of 1. The Virtual Script Player automatically handles any API calls invoked by the original call, just as the OneWorld run-time engine would.

This OneWorld Virtual User Tool capability is important for playing scripts back in batch mode. If APIs with a call level greater than 1 were treated separately, repetitive processing would occur. Such repetition would not correctly simulate OneWorld processing.

The Virtual Script Editor provides a convenient way for you to view call level in the event stream. Each line that displays an initiating API call shows a call level of 1. Any calls that are invoked by the initiating call show a level of 2 or greater.

> **Note:** The Virtual Script Editor displays API calls with a call level greater than 1 only if you choose the Capture Performance Statistics option when you configure playback in AutoPilot. If you do not wish to view call levels greater than 1, choose the Capture Virtual Script event stream option.



Start Time	Elapsed	Thread	Call Level	MsgID	Message
117.60880	0.1051	2	1	2072	NET: CallObj: [GetDistributionConstants]
117.66483	0.0012	2	2	2022	OpenTableFromCache (F4009)
117.66811	0.0158	2	2	2004	FetchKeyed (F4009)
117.66880	0.0038	2	3	2028	SelectKeyed (F4009)
117.70513	0.0060	2	2	2016	JDB_CloseTable (F4009)
117.70548	0.0005	2	3	2014	JDB_ClearSequencing ()
117.70814	0.0005	2	3	2012	JDB_ClearSelection ()

Note that if you click on the detail line of an API call that has a call level of 2 or greater, the event stream detail pane displays no parameters, meaning that you cannot value-link any API call with a call level greater than 1. Therefore, no API calls with a call level greater than 1 appear in the Target Parameters pane.

Synchronous and Asynchronous Calls

As part of simulating OneWorld operations, OneWorld Virtual User Tool must be able to handle synchronous and asynchronous API calls, an important management responsibility of the OneWorld run-time engine. This ability ties into the Virtual Script Player's management of threads because an asynchronous call generates a separate thread.

A typical example of synchronous and asynchronous API call generation occurs when you enter data to a sales order line in OneWorld. You generate a synchronous call for each line edit. That is, the CallObject API for line 1 in a OneWorld grid precedes the CallObject API for line 2, and the CallObject API for line 2 does not occur until you have completed line 1. However, when you reach the end of a line, press the tab button, and proceed to line 2, you also generate an asynchronous API call that includes the data structure for the line you have just completed. The asynchronous CallObject API validates the data you entered to line 1 through a series of related API calls. Meanwhile, you move ahead and begin entering sales order data to line 2.

The OneWorld run-time engine manages this situation by generating a new thread for asynchronous calls and sending these calls to a queue for handling on a first-in, first-out (FIFO) basis. For example, you might enter 20 lines to the sales order entry grid. As you reach the end of each line and tab, it's likely that the system will generate a new asynchronous call. Therefore, a number of asynchronous calls might queue up for handling. When the run-time engine finishes processing the asynchronous calls, it kills the thread.

OneWorld Virtual User Tool handles the simulation of asynchronous call management through the operation of each part of its architecture. The OneWorld Scripting Tool and OneWorld hooks capture the timings of the synchronous and asynchronous calls that script playback generates. The Virtual Script Editor preserves the thread identifiers produced during playback, and the Virtual Script Player generates thread synchronization events in the OneWorld Virtual User Tool script based on the temporal relationships among events in the captured event stream.

The Virtual Script Player also manages the threads generated during virtual playback. When virtual playback yields an asynchronous call, the Virtual Script Player queues the calls in a new thread and manages them on the same FIFO terms that the OneWorld run-time engine uses, thereby managing interthread synchronization as well as event timing within threads.

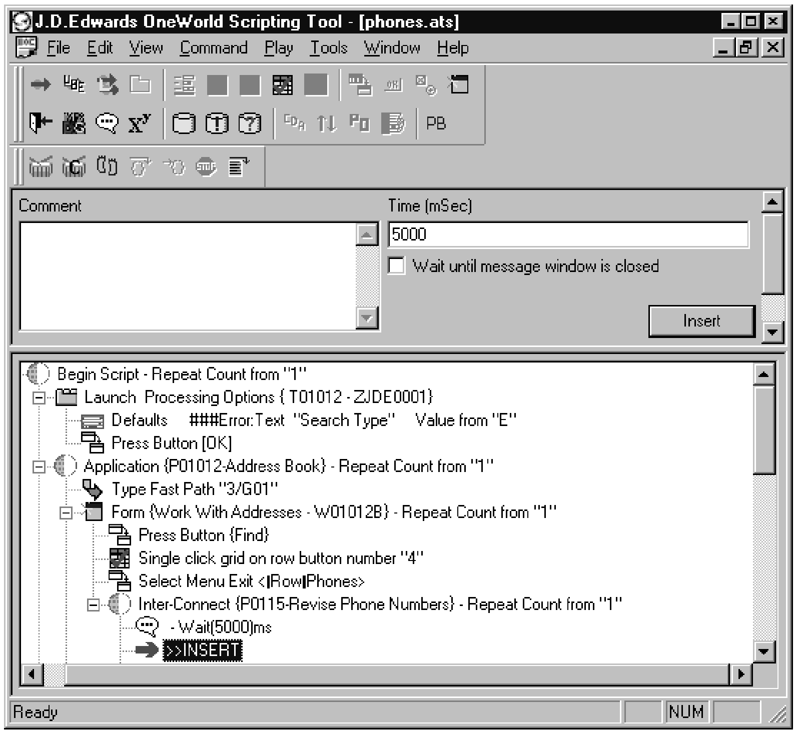
Synchronous and asynchronous call management provides another example of OneWorld Virtual User Tool's ability to accurately simulate the way OneWorld works, thereby providing you with a realistic picture of network and server stress.

See Also

- *Playback Timing*

Think Times

You insert think times in writing a OneWorld Scripting Tool script in an attempt to accurately simulate the way people use OneWorld. You click the Wait Before Proceeding button and insert pauses in the playback in millisecond increments.


J.D.Edwards OneWorld Scripting Tool - [phones.ats]
File Edit View Command Play Tools Window Help
PB
Insert
Ready
NUM

Comment

Time (mSec)

5000

☐ Wait until message window is closed

```
Begin Script - Repeat Count from "1"
  Launch  Processing Options { T01012 - ZJDE0001}
    Defaults    ###Error:Text  "Search Type"    Value from "E"
    Press Button [OK]
  Application {P01012-Address Book} - Repeat Count from "1"
    Type Fast Path "3/G01"
    Form {Work With Addresses - W01012B} - Repeat Count from "1"
      Press Button {Find}
      Single click grid on row button number "4"
      Select Menu Exit <|Row|Phones>
      Inter-Connect {P0115-Revise Phone Numbers} - Repeat Count from "1"
        - Wait(5000)ms
        >>INSERT
```

One possible reason for inserting wait periods to the script is that you may want to simulate the pauses that might occur as a person working in OneWorld makes voucher entries, for example. Such a person might pause occasionally to answer the phone or tend to other tasks, then return to making the entries.

The event stream generated during OneWorld Scripting Tool script playback records these wait times. They do not appear, however, with a label in the event stream pane if you import the event stream into the Virtual Script Editor. Rather, you recognize them by noting in the event stream pane the duration of time between the end of one event and the beginning of the next.

The OneWorld Virtual User Tool script that you create contains the waits inserted in the original script, and the Virtual Script Player handles the delays during script playback. The inclusion of think times provides another element that helps the OneWorld Virtual User Tool you simulate the OneWorld environment, which includes many users performing different tasks under a variety of circumstances.

You might be interested in analyzing the event stream to see the length of time the events in the OneWorld Scripting Tool script took to complete. Think times you insert in the script do not interfere with event duration analysis because the Virtual Script Editor's event graph does not reflect any wait times. If, for example, a five-second wait occurs between a CallObject API and an OpenTable API, the event graph displays only the amount of time that was required to execute the APIs. You thus get a true picture of the time OneWorld required to process the API calls.

OneWorld Virtual User Tool Troubleshooting Tips and Techniques

OneWorld Virtual User Tool Troubleshooting Tips and Techniques

You might encounter a script failure when you play back your OneWorld Virtual User Tool script. Troubleshooting OneWorld Virtual User Tool scripts falls into two categories. First, you must locate the source of the script failure, which might be in either OneWorld Virtual User Tool or in OneWorld. After you have accomplished that, you run through a short list of script debugging techniques. These techniques focus on correcting errors in business function and database API calls, transaction timing, and multiple playback sessions. You might also need to debug OneWorld. In some cases the problem lies in the original OneWorld Scripting Tool script or in application source code. Finally, you might have to check your OneWorld Scripting Tool script, if you created it without first validating it through replay.

You can not trace all failures of OneWorld Virtual User Tool scripts to a single source, nor can you debug all scripts following a single method. In learning tips and techniques for troubleshooting OneWorld Virtual User Tool scripts, you also learn the best solution to apply to a particular problem.

This section discusses the two main components of troubleshooting OneWorld Virtual User Tool script playback:

- ❑ Locating the causes of OneWorld Virtual User Tool script failures
- ❑ Debugging OneWorld Virtual User Tool scripts

For further discussion of both OneWorld Virtual User Tool and OneWorld Scripting Tool see also *OneWorld Scripting Tool Guide.*

Locating the Causes of OneWorld Virtual User Tool Script Failures

The vap.log file contains messages about each OneWorld Virtual User Tool script that you run. Therefore, it is the primary source of information concerning errors that might cause your script to fail. You set the message level in your OneWorld Virtual User Tool initialization file. You should generally set the message parameter at 0, 1, 3, or 7 to minimize the number of messages you collect. Setting the parameter higher will cause playback performance to suffer and at least potentially skew playback results, thereby making performance analysis difficult. However, when you are attempting to find the source of a script failure, increasing the message level parameter temporarily can help you to diagnose the problem.

If you fail to find the source of the script failure in OneWorld Virtual User Tool, you can use several procedures to troubleshoot OneWorld.

This chapter discusses the following troubleshooting tasks:

- ❑ Finding error entries in the OneWorld Virtual User tool log file
- ❑ Locating the log file in the event of early script failure
- ❑ Setting the MessageLevel parameter
- ❑ Identifying a OneWorld environment problem

See Also

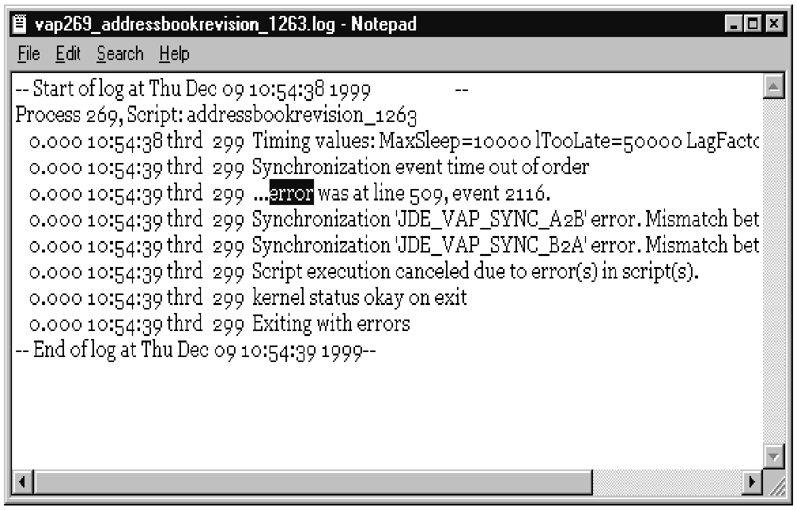
- *Virtual Script Player Initialization File Parameters*

Finding Error Entries in the OneWorld Virtual User Tool Log File

The Virtual Script Player sends an error message to the log file when a OneWorld Virtual User Tool script fails during execution. If you have launched the script from LoadRunner or from Virtual Runner, script failure halts the playback process, sending an instant signal that an error has occurred. However, if you have launched the script from the DOS command line, you will not receive an error message. In either case, to isolate an error, you go to the log file, which contains the test results, choose a test, open the text file, and search on the keyword "Error."

▶ To find entries in the OneWorld Virtual User Tool log file

1. In the vap.ini file, go to the [LOG] section to determine the location of the LogDirectory.
2. Follow the path to the LogDirectory.
3. Open the text file for the failed script.
4. In the Notepad form menu bar, click Search.
5. In the drop-down menu, click Find.
6. In the Find form, enter the word "Error."



```
-- Start of log at Thu Dec 09 10:54:38 1999          --
Process 269, Script: addressbookrevision_1263
  0.000 10:54:38 thrd  299  Timing values: MaxSleep=10000 lTooLate=50000 LagFactc
  0.000 10:54:39 thrd  299  Synchronization event time out of order
  0.000 10:54:39 thrd  299  ...error was at line 509, event 2116.
  0.000 10:54:39 thrd  299  Synchronization 'JDE_VAP_SYNC_A2B' error. Mismatch bet
  0.000 10:54:39 thrd  299  Synchronization 'JDE_VAP_SYNC_B2A' error. Mismatch bet
  0.000 10:54:39 thrd  299  Script execution canceled due to error(s) in script(s).
  0.000 10:54:39 thrd  299  kernel status okay on exit
  0.000 10:54:39 thrd  299  Exiting with errors
-- End of log at Thu Dec 09 10:54:39 1999--
```

7. Click Find Next.

See Also

- *Virtual Script Player Initialization File Parameters*

Locating the Log File in the Event of Early Script Failure

The Virtual Script Player reads the location of the log directory out of the VAP.ini text file. However, the script might fail before the Virtual Script Player has a chance to read the location. Therefore, when you go to the location of the log file that you specified as an initialization parameter, you will not find the test log. Despite the early failure, OneWorld Virtual User Tool did log the errors. To find the test log, you go to the root of the drive that contains your log file and look for it there.

▶ To locate the OneWorld Virtual User Tool log file in the event of early script failure

1. In the vap.ini file, determine the location of the LogDirectory.
2. Follow the path to the LogDirectory.
3. If the log that you are looking for is not in its usual location, go to the root of the drive and look for the log.

   Note: If you do not find a log file in either location, you must examine your OneWorld Virtual User Tool setup. Make sure that OneWorld Virtual User Tool has been completely and correctly installed.

Setting the MessageLevel Parameter

You can set a message level parameter in the vap.ini file. This setting controls the kind and number of error messages that you receive in the vap.log file when you play back a OneWorld Virtual User Tool script.

You might find very few messages in the log file as a result of setting the debug parameter too low. For example, if you set the parameter to zero, you will receive only a minimal number of messages. To address this, you go into the vap.ini file and change the MessageLevel parameter to 1, 3, 7, or 15. At each successive level, the Virtual Script Player writes more messages to the log. See the *Log* topic in the *Virtual Script Player* chapter of this guide for more discussion of the MessageLevel parameter.

▶ To set the message level parameter

1. In the vap.ini file, find the [Log] entry.
2. If the MessageLevel parameter is set lower than you want, change the setting.
3. Save your change and close the vap.ini file.

   Note: If the Virtual Script Player crashes while you are running a script, you might find very few messages in the log file. This occurs because the Virtual Script Player did not flush the log file buffer, in which messages are stored, before the crash. You can combat this by setting the message level parameter at 31. This parameter requires that the Virtual Script Player flush the log file buffer after each message. Remember, however, that the performance of the system degrades when you set the message level at 31, so you should not leave it at that level permanently.

Identifying a OneWorld Environment Problem

If your OneWorld Virtual User Tool script fails very early, even before the system completes its initial OneWorld logon, you might not be initializing a OneWorld environment. In this case, you can troubleshoot OneWorld operations rather than OneWorld Virtual User Tool operations. For example, you can try to log on to OneWorld Explorer and run it through several sample tasks, such as opening an application. Use the same user ID, password, and environment name when you log onto OneWorld that you used in logging onto the Virtual Script Player. You can also troubleshoot errors in OneWorld, as these might also prevent you from replaying your OneWorld Virtual User Tool script. If you have cleared any problems that might exist in running OneWorld Explorer, you can try running your OneWorld Virtual User Tool script again.

This topic details the steps necessary to troubleshoot problems you might encounter in running OneWorld:

- Diagnosing a OneWorld environment problem
- Investigating OneWorld errors

Diagnosing a OneWorld Environment Problem

Since OneWorld Virtual User Tool's primary task is to simulate OneWorld operations, it must be able to initialize a OneWorld environment at script playback time. For this to happen, OneWorld itself must be initializing correctly. To exclude the Virtual Script Player as the source of script failure, you might simply attempt to log on to OneWorld to make sure that it is opening and running correctly.

► **To diagnose a OneWorld environment problem**

1. Close the Virtual Script Player.
2. Log onto OneWorld Explorer.
3. Perform several OneWorld operations, such as opening an application, switching forms, adding data, and so on.
4. If you are certain that OneWorld is running correctly, try rerunning the script.

   Note: Be sure to use the same User ID, password, and environment that you use when you log onto the Virtual Script Player.

Investigating OneWorld Errors

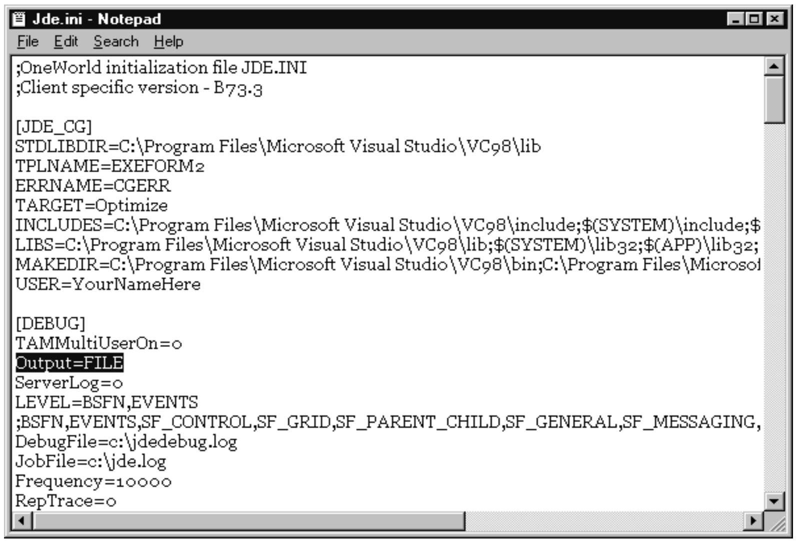
Even if OneWorld is initializing correctly, you might find errors that occur when you attempt to enter, add, or edit data in an application. To isolate errors that occur in OneWorld, you can turn on OneWorld debugging and attempt to correct the errors.

▶ To investigate OneWorld errors

1. Click the Windows Start menu and choose Run.
2. In the Open control of the Run form, type in jde.ini.

   The jde.ini file appears.

3. In the jde.ini file, go to the [DEBUG] section.
4. Enter the Output parameter as File.



```
;OneWorld initialization file JDE.INI
;Client specific version - B73.3

[JDE_CG]
STDLIBDIR=C:\Program Files\Microsoft Visual Studio\VC98\lib
TPLNAME=EXEFORM2
ERRNAME=CGERR
TARGET=Optimize
INCLUDES=C:\Program Files\Microsoft Visual Studio\VC98\include;$(SYSTEM)\include;$
LIBS=C:\Program Files\Microsoft Visual Studio\VC98\lib;$(SYSTEM)\lib32;$(APP)\lib32;
MAKEDIR=C:\Program Files\Microsoft Visual Studio\VC98\bin;C:\Program Files\Microsol
USER=YourNameHere

[DEBUG]
TAMMultiUserOn=0
Output=FILE
ServerLog=0
LEVEL=BSFN,EVENTS
;BSFN,EVENTS,SF_CONTROL,SF_GRID,SF_PARENT_CHILD,SF_GENERAL,SF_MESSAGING,
DebugFile=c:\jdedebug.log
JobFile=c:\jde.log
Frequency=10000
RepTrace=0
```

5. Run a OneWorld Scripting Tool script or open OneWorld and run applications that are failing in the OneWorld Virtual User Tool script.
6. Open the jdedebug.log file to evaluate any errors that occur.
7. If OneWorld displays any error messages in the status bar, click the stop sign.
8. Read the error messages that appear.
9. Right click error messages to display more troubleshooting information about each one.

   Caution: Be sure to return the Output parameter in the [DEBUG] section of the jde.ini file to NONE after you have corrected errors that prevent the OneWorld Virtual User Tool script from functioning correctly.

Debugging OneWorld Virtual User Tool Scripts

If you have spent time troubleshooting problems with OneWorld Virtual User Tool script playback, but still find that you are having trouble running the scripts, a business function call is likely causing the failure. You can check the log file to locate the source of the error, and you can identify the particular business function call that failed. It is important that you have the message level in the vap.ini file set at 15 so that the log file displays parameter values.

You might also encounter problems that complicate your performance characterization efforts. For example, transaction information that you incorrectly or incompletely enter in a OneWorld Scripting Tool script might cause irregular transaction times in the OneWorld Virtual User Tool script, thus making it difficult to draw accurate conclusions about OneWorld performance. In this case, you should troubleshoot the OneWorld Scripting Tool script, making sure that you have completely and accurately scripted input commands. If you modify the OneWorld Scripting Tool script, remember to run it again, capture the playback results, and reimport the event stream into the Virtual Script Editor.

OneWorld Virtual User Tool also allows you to play back a script multiple times in succession, another important feature for performance characterization. However, doing so might cause playback to lock, again defeating your efforts to draw clearly and confidently characterize system performance. In this event, check your disk space to make sure you have enough to handle the testing.

If you have exhausted all of the debugging possibilities discussed here, you must turn your attention to debugging OneWorld Explorer. Remember that if the same errors that appear in your OneWorld Virtual User Tool script appear when you run the application in OneWorld, you likely have a OneWorld problem that you must debug. Your OneWorld debugging efforts might include a call to OneWorld System Support.

You can gain additional insight into potential OneWorld problems by double clicking on the stop sign that appears in the status bar of a OneWorld form when an error occurs. When you perform this action, OneWorld displays explanatory text, including possible causes and solutions, which help you diagnose the source of the error. In addition, you can garner additional troubleshooting information by setting the Output parameter in the jde.ini file to FILE. However, remember that doing so will degrade OneWorld performance, so you should return the Output parameter in the jde.ini file to NONE after you have diagnosed and corrected any problems with the script.

This chapter discusses the following script-debugging tasks:

- ❑ Displaying business function parameters
- ❑ Diagnosing business function failures in OneWorld Explorer
- ❑ Troubleshooting value-linking errors
- ❑ Verifying the validity of OneWorld Virtual User Tool script data
- ❑ Identifying and correcting duplicate key errors
- ❑ Rectifying irregular transaction times
- ❑ Preventing multiple script playback problems
- ❑ Correcting uninitialized user handles

Displaying Business Function Parameters

Displaying the business function call parameters helps you to debug your OneWorld Virtual User Tool scripts. To do so, you set the MessageLevel parameter in the vap.ini file at 15 or at 31. At this level, the log file displays all the input and output parameter values of the business function API calls in the script, along with the text of any error messages, the file name of the business function, and the line number in the source code that contains the error.

▶ To display business function parameters

1. In the vap.ini file, find the [Log] entry.
2. Set the Message Level at 15 or 31.
3. Save your change and close the vap.ini file.

   Caution: Remember again that you should not set the MessageLevel parameter permanently at 31 as this will cause performance to degrade. Leaving the MessageLevel parameter at 15 does not significantly degrade performance, but it can cause many messages and a great deal of text to accumulate in the log file. Therefore, you should not leave the message level permanently set at 15, as doing so could consume a significant amount of disk space.

Diagnosing Business Function Failures in OneWorld Explorer

Your scripts must run properly in OneWorld Explorer before OneWorld Virtual User Tool can run them properly. Therefore, it makes sense to determine early on if business function API calls are failing in OneWorld when you run a OneWorld Scripting Tool script. To do so, you can turn on OneWorld debugging in the jde.ini file. When you run an application, you can right click the mouse and choose View System Log. From there you can view the jdedebug log.

You might set breakpoints in the OneWorld Scripting Tool script after commands that initialize a business function API call, which will allow you to check the jdedebug log in OneWorld at these key points.

By checking OneWorld's ability to execute the commands in the OneWorld Scripting Tool script, you either pinpoint or exclude OneWorld as a source of script failure. If OneWorld is causing the script failure, you work on debugging OneWorld. Conversely, if the business functions execute properly when you run the script in OneWorld, you concentrate on finding the source of the script failure in OneWorld Virtual User Tool.

▶ To diagnose business function failures in OneWorld

1. In the jde.ini file, go to the [DEBUG] section and set the output parameter to FILE.
2. In the OneWorld Scripting Tool script pane, right click a command line that follows a command that executes a business function (optional).
3. Click Toggle Breakpoint (optional).
4. Play back the OneWorld Scripting Tool script, either to the end or to a designated breakpoint.
5. In OneWorld, right click inside a OneWorld form.
6. Choose View System Log.
7. Click File.
8. In the drop-down menu, choose c:\jdedebug.log.
9. Troubleshoot the jdedebug.log file, searching for business functions.

Troubleshooting Value-Linking Errors

In order for your OneWorld Virtual User Tool script to run correctly, you must value-link all required target parameters to the appropriate source parameters, using the Virtual Script Editor. Failing to do so, or forging value links improperly could cause your script to fail.

This topic discusses identifying and troubleshooting possible value-linking errors:

- Researching value-linking errors in the Virtual Script Editor
- Verifying that value linking is functioning
- Identifying and correcting variable value-linking errors

Researching Value-Linking Errors in the Virtual Script Editor

A business function API call might fail when you run your OneWorld Virtual User Tool script because you incompletely value-linked the business function parameters in the event stream to the parameters in the OneWorld Virtual User Tool script while you were working in the Virtual Script Editor. Remember that you must value-link any parameters that do not use constant values during script playback. If you do not value-link these parameters, the script will fail because, typically, the script playback creates duplicate keys.

Parameters that require value linking include job number, document number, batch number, and any parameters to which you assign values from a valid values list. In addition, parameters such as computer identification and those containing a date value often require value linking.

After you examine the log file and perform necessary value linking that you might not have completed during script editing, you can rerun the script with the MessageLevel parameter in the [LOG] section of your vap.ini file set at 15. This setting allows you to capture parameter values and value substitutions in the log file.

▶ To research value-linking errors in the Virtual Script Editor

1. In the vap.ini file, set the MessageLevel parameter at 15.
2. Run the OneWorld Virtual User Tool script.
3. In the vap.log file, search for business function errors.
4. In the Virtual Script Editor, check your value linking.

   Note: Remember that you are required to provide value links for the following parameters:

   - Job number
   - Document number
   - Batch number
   - Any parameter that uses a value from a valid values list

The following parameters might frequently require value linking:

   - Computer identification
   - Those that require dates

5. Perform any necessary value linking in the Virtual Script Editor.
6. Rerun the OneWorld Virtual User Tool script.
7. Recheck the log file and look for business function API errors.

Verifying That Value Linking is Functioning

You can check to make sure that OneWorld Virtual User Tool is linking parameter values. To do so, you create and use valid values lists in your OneWorld Scripting Tool script. In the Virtual Script Editor, you value-link any parameters that use values from the valid values list. The Virtual Script Player, during OneWorld Virtual User Tool script playback, should link the values in the valid values lists to the appropriate parameters in the OneWorld Virtual User Tool script.

To verify that OneWorld Virtual User Tool performs the value linking, you can set your MessageLevel parameter at 15 and run the OneWorld Virtual User Tool script. After you run the script, you search the log file for valid values list data, identify that data, and change the data in the .atd file, which stores your valid values list data.

When you replay the script, OneWorld Virtual User Tool should use the new data from the valid values list. After you replay the OneWorld Virtual User Tool script, you can search the log file again for valid values list data to make sure that the Virtual Script Player used the new data rather than any of the old values. If the Virtual Script Player used any of the old values, you must go back to the Virtual Script Editor and make sure you have sufficiently linked all of the values from the valid values lists to the appropriate parameters in the OneWorld Virtual User Tool script.

▶ To verify that value linking is functioning

1. In the vap.ini file, set the MessageLevel parameter at 15.
2. Run the OneWorld Virtual User Tool script.
3. Check the log file for valid values list data.

   Note: You can search for valid values list data using the .atd extension. Check to make sure that the values you expect are present and look for any error messages associated with the data.

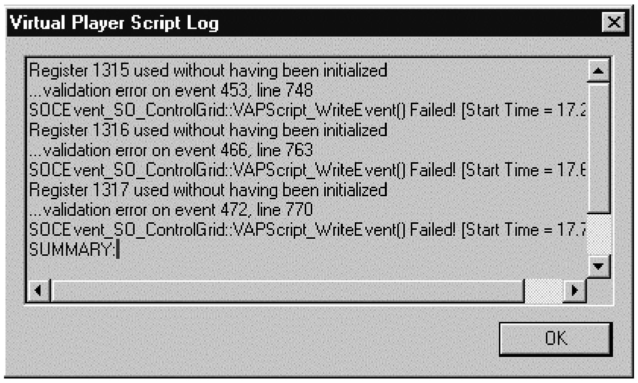
4. In the c:\.atd file, change the valid values list data.
5. In the Virtual Script Editor, make sure that you have value-linked all of the new data in the valid values list to the correct parameters in the OneWorld Virtual User Tool script.
6. Rerun the OneWorld Virtual User Tool script.
7. Recheck the log file for old valid values list data.
8. If you find any of the old valid values list data, recheck the value linking in the Virtual Script Editor.

Identifying and Correcting Variable Value-Linking Errors

A different value-linking-related error occurs if you declare a value in a OneWorld Scripting Tool script, but do not set its value. In this case, if you value-link the variable, the Virtual Script Editor registers errors in the script log during the virtual script generation process. Correcting the errors requires you to modify the OneWorld Scripting Tool script by setting the variable's value.

▶ **To identify and correct variable value-linking errors**

1. In the Virtual Script Editor, select a test and click the Generate button.
2. Study the Virtual Player Script Log form for validation error messages.



```
Register 1315 used without having been initialized
...validation error on event 453, line 748
SOCEvent_SO_ControlGrid::VAPScript_WriteEvent() Failed! [Start Time = 17.2
Register 1316 used without having been initialized
...validation error on event 466, line 763
SOCEvent_SO_ControlGrid::VAPScript_WriteEvent() Failed! [Start Time = 17.6
Register 1317 used without having been initialized
...validation error on event 472, line 770
SOCEvent_SO_ControlGrid::VAPScript_WriteEvent() Failed! [Start Time = 17.7
SUMMARY:
```

3. If validation error messages appear in the Virtual Script Log form, reopen the OneWorld Scripting Tool script.
4. Modify the OneWorld Scripting Tool script by setting a value for any declared variables that do not have a value that you value-linked in the Virtual Script Editor.
5. Save and rerun the OneWorld Scripting Tool script.

   Caution: Be sure that playback remains configured to capture the virtual script event stream.

6. Reimport the event stream into the Virtual Script Editor and regenerate the OneWorld Virtual User Tool script.

Verifying the Validity of OneWorld Virtual User Tool Script Data

Business function errors that occur in the OneWorld Virtual User Tool script might be caused by data errors. Data errors likely occur because the OneWorld environment against which you wrote the OneWorld Scripting Tool script differs from the OneWorld environment against which you attempt to play back the OneWorld Virtual User Tool script. To verify that the data you use in the OneWorld Virtual User Tool script is valid, you can update the values in your valid values list so they will work in the environment against which you run the OneWorld Virtual User Tool script. Alternatively, you can replay the OneWorld Scripting Tool script against the OneWorld environment in which you will run the OneWorld Virtual User Tool script. Remember that in this case you will have to reimport the event stream into the Virtual Script Editor and regenerate a OneWorld Virtual User Tool script by re-establishing value links.

▶ **To verify the validity of data in the OneWorld Virtual User Tool script**

1. In the vap.ini file, set the MessageLevel parameter to 15.
2. Run the OneWorld Virtual User Tool script.
3. Search the log file for business function errors.
4. Check the OneWorld environment against which you wrote the OneWorld Scripting Tool script and against which you ran the OneWorld Virtual User Tool script.
5. If the two environments are different, recreate your valid values lists so that they contain values that are valid for the OneWorld environment against which you are running the OneWorld Virtual User Tool script.

   Note: You can also replay the OneWorld Scripting Tool script in the same OneWorld environment against which you are running the OneWorld Virtual User Tool script. In that case, follow the next two steps.
6. Reimport the event stream into the Virtual Script Editor.
7. Re-generate a OneWorld Virtual User Tool script by forging value links between the source and target parameters.

Identifying and Correcting Duplicate Key Errors

JDB Insert and Update API calls might fail in the OneWorld Virtual User Tool script because of duplicate key errors. These errors occur when you attempt to enter to a key column two records with the same value. The duplicate key error prevents this from occurring. Failure to value-link all the necessary parameters in the OneWorld Virtual User Tool script could cause duplicate key errors. You can view updated and inserted JDB API parameter values in the Virtual Script Editor. However, duplicate keys could also result from an application error.

▶ **To identify and correct duplicate key errors**

1. In the OneWorld jde.ini file, go to the [DEBUG] section.
2. Change the Output parameter to FILE.
3. Play the OneWorld Scripting Tool script.
4. If you have duplicate key errors, they will appear in the jdedebug.log file.

5. Open the script in the Virtual Script Editor.
6. Check value linking for all JDB Insert and Update API calls.
7. When you are sure that you have value-linked all JDB Insert and Update API calls, rerun the script.
8. If you continue to get duplicate key errors, check the application for errors that might be causing the problem.

Rectifying Irregular Transaction Times

You measure transaction times by choosing events as start and end points in your OneWorld Scripting Tool script. For example, you might launch an application, move from one OneWorld form to another by pressing the Add button, then make entries to several header controls and grid columns in an active form before closing the form.

You might choose to label that entire sequence of commands, from launching the application to closing the form, as a transaction. To see how efficiently OneWorld handles this transaction, you label in the OneWorld Scripting Tool script launching the application as the start of the transaction, and label closing out of the form as the end of the transaction. You also apply a name to the transaction and attach that name to the start and to the end. You use the Wait/Comment command in OneWorld Scripting Tool to insert the start and end of the transaction into the script and to apply a name to the transaction.

If you do not include both a start and an end time for the transaction, you might find irregular or inexplicable transaction times in the log, or you might find that the transaction fails. Failing to ensure that the name that you applied to the start of the transaction matches precisely the name that you applied to the end of the transaction, including capital letters and any special characters, might also cause irregular transaction times or transaction failures.

Caution: At this stage, OneWorld Virtual User Tool transaction timing accuracy faces several limitations that make broad-based performance characterization assertions impossible. Accurate timings can only be achieved on a discrete workstation, while OneWorld Virtual User Tool simulates server load.

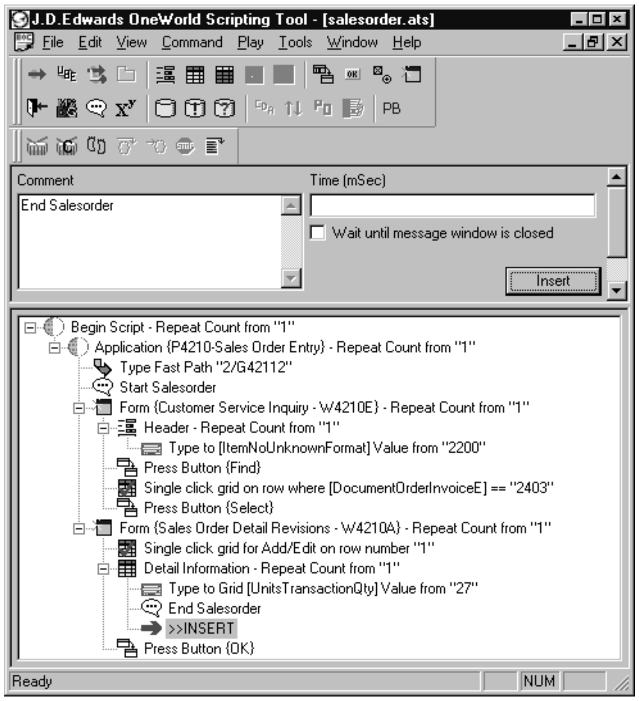
▶ To rectify irregular transaction times

1. In the script pane of the OneWorld Scripting Tool form, determine the command line that represents the start of the transaction, then place the insertion cursor directly above it.
2. In the OneWorld Scripting Tool menu bar, click Command.
3. Choose Wait/Comment.

4. In the unpopulated Comment list of the OneWorld Scripting Tool command pane, enter Start, a space, and a name for the transaction.
5. Click the Insert button.

   OneWorld Scripting Tool inserts a command line marking the start of the transaction.

6. Determine the command line that represents the end of the transaction, then place the insertion cursor after it.
7. In the OneWorld Scripting Tool menu bar, click Command.
8. Choose Wait/Comment.
9. In the unpopulated Comment list of the OneWorld Scripting Tool command pane, enter End, a space, and a name for the transaction.


J.D.Edwards OneWorld Scripting Tool - [salesorder.ats]
File Edit View Command Play Tools Window Help
PB
Comment
Time (mSec)
Wait until message window is closed
Insert
Ready
NUM

Comment: End Salesorder

```
Begin Script - Repeat Count from "1"
  Application {P4210-Sales Order Entry} - Repeat Count from "1"
    Type Fast Path "2/G42112"
    Start Salesorder
    Form {Customer Service Inquiry - W4210E} - Repeat Count from "1"
      Header - Repeat Count from "1"
        Type to [ItemNoUnknownFormat] Value from "2200"
      Press Button {Find}
      Single click grid on row where [DocumentOrderInvoiceE] == "2403"
      Press Button {Select}
    Form {Sales Order Detail Revisions - W4210A} - Repeat Count from "1"
      Single click grid for Add/Edit on row number "1"
      Detail Information - Repeat Count from "1"
        Type to Grid [UnitsTransactionQty] Value from "27"
        End Salesorder
        >>INSERT
      Press Button {OK}
```

OneWorld Scripting Tool inserts a command line marking the end of the transaction.

Caution: The name that you assign to the end of the transaction must exactly match the name that you assign to the start of the transaction.

10. Click the Insert button.

Preventing Multiple Script Playback Problems

OneWorld Virtual User Tool allows you to play back the same script in consecutive sessions or to simulate multiple users playing back scripts simultaneously. In either case, you must make sure that you have sufficient disk space to handle the load created by OneWorld Virtual User Tool script playback, particularly if you plan to run a long test involving many playback iterations or simulation of a large number of OneWorld users. Otherwise, you might find that OneWorld Virtual User Tool script playback locks up after only a few playbacks.

This topic discusses troubleshooting techniques to use when you run multiple virtual scripts:

- Debugging Virtual Runner
- Debugging LoadRunner

Debugging Virtual Runner

If Virtual Runner fails immediately when you click the Run button, first the Virtual Script Player.exe path specified in the vap.ini file. The vap.ini [COMMAND] section binname parameter specifies the full path of the Virtual Script Player.exe file. If that parameter is correct you must attempt to debug the script.

Virtual Script Player should behave the same whether you run under Virtual Runner control or from a command line. If it does not, you might be running two different copies of the Virtual Script Player.exe. This might occur if the vap.ini [COMMAND] binname parameter is pointing to an old version of the Virtual Script Player. Make sure that binname parameter points to the correct drive and directory, and that you discard any old versions of the Virtual Script Player that you might have on your workstation.

If you set Virtual Script Player to execute a virtual script multiple times in succession, and the script only runs a few times before locking up, you should check the available disk space. With high jde.ini error logging settings, jde.log and jdedebug.log can fill up a disk very quickly. Make sure that there is enough free space on all relevant disk drives before you start a long test.

Debugging LoadRunner

If you set the message level at a high number and you run many virtual user sessions, the network might become saturated and/or communications between LoadRunner controller and the host machines might become scrambled. You can address this problem by setting the MessageLevel parameter in the vap.ini files on all machines lower. This will decrease the volume of log file traffic.

The following table summarizes steps you can take to minimize OneWorld Virtual User Tool script playback problems:

Situation Potentially Affecting Playback	Possible Solution
OneWorld jde.log and jdedeb.log messages fill up disk quickly during OneWorld Virtual User Tool script playback.	In the [DEBUG] section of the jde.ini file, set Output parameter to NONE.
OneWorld Virtual User Tool log file fills with messages, consuming disk space.	In the [LOG] section of the vap.ini file, set MessageLevel parameter to 0, 1, 3, or 7.

Correcting Uninitialized User Handle Errors

An error labeled "Uninitialized User Handle" might cause your OneWorld Virtual User Tool script to fail. This error occurs when you attempt to create a OneWorld Virtual User Tool script using playback results that you obtained from the first run of a OnewWorld application when just in time installation occurs, or when you have OneWorld debugging turned on when you capture the results of OneWorld Scripting Tool script playback.

▶ To correct uninitialized user handle errors

1. In OneWorld Virtual User Tool, discard the results of the script generation attempt that failed.
2. In OneWorld Scripting Tool, rerun the script in the same environment that you created it.
3. Use the new results data to generate a new OneWorld Virtual User Tool script.

Glossary

Glossary

AAI. See automatic accounting instruction.

action message. With OneWorld, users can receive messages (system-generated or user-generated) that have shortcuts to OneWorld forms, applications, and appropriate data. For example, if the general ledger post sends an action error message to a user, that user can access the journal entry (or entries) in error directly from the message. This is a central feature of the OneWorld workflow strategy. Action messages can originate either from OneWorld or from a third-party e-mail system.

activator. In the Solution Explorer, a parent task with sequentially-arranged child tasks that are automated with a director.

ActiveX. A computing technology, based on object linking and embedding, that enables Java applet-style functionality for Web browsers as well as other applications. (Java is limited to Web browsers at this time.) The ActiveX equivalent of a Java applet is an ActiveX control. These controls bring computational, communications, and data manipulation power to programs that can "contain" them. For example, certain Web browsers, Microsoft Office programs, and anything developed with Visual Basic or Visual C++.

advance. A change in the status of a project in the Object Management Workbench. When you advance a project, the status change might trigger other actions and conditions such as moving objects from one server to another or preventing check-out of project objects.

alphanumeric character. A combination of letters, numbers, and symbols used to represent data. Contrast with numeric character and special character.

API. See application programming interface.

APPL. See application.

applet. A small application, such as a utility program or a limited-function spreadsheet. It is generally associated with the programming language Java, and in this context refers to Internet-enabled applications that can be passed from a Web browser residing on a workstation.

application. In the computer industry, the same as an executable file. In OneWorld, an interactive or batch application is a DLL that contains programming for a set of related forms that can be run from a menu to perform a business task such as Accounts Payable and Sales Order Processing. Also known as system.

application developer. A programmer who develops OneWorld applications using the OneWorld toolset.

application programming interface (API). A software function call that can be made from a program to access functionality provided by another program.

application workspace. The area on a workstation display in which all related forms within an application appear.

audit trail. The detailed, verifiable history of a processed transaction. The history consists of the original documents, transaction entries, and posting of records, and usually concludes with a report.

automatic accounting instruction (AAI). A code that refers to an account in the chart of accounts. AAIs define rules for programs that automatically generate journal entries, including interfaces between Accounts Payable, Accounts Receivable, Financial Reporting, General Accounting systems. Each system that interfaces with the General Accounting system has AAIs. For example, AAIs can direct the General Ledger Post program to post a debit to a specific expense account and a credit to a specific accounts payable account.

batch header. The information that identifies and controls a batch of transactions or records.

batch job. A task or group of tasks you submit for processing that the system treats as a single unit during processing, for example, printing reports and purging files. The computer system performs a batch job with little or no user interaction.

batch processing. A method by which the system selects jobs from the job queue, processes them, and sends output to the outqueue. Contrast with interactive processing.

batch server. A server on which OneWorld batch processing requests (also called UBEs) are run instead of on a client, an application server, or an enterprise server. A batch server typically does not contain a database nor does it run interactive applications.

batch type. A code assigned to a batch job that designates to which J.D. Edwards system the associated transactions pertain, thus controlling which records are selected for processing. For example, the Post General Journal program selects for posting only unposted transaction batches with a batch type of O.

batch-of-one immediate. A transaction method that allows a client application to perform work on a client workstation, then submit the work all at once to a server applicationfor further processing. As a batch process is running on the server, the client application can continue performing other tasks. See also direct connect, store and forward.

BDA. See Business View Design Aid.

binary string (BSTR). A length prefixed string used by OLE automation data manipulation functions. Binary Strings are wide, double-byte (Unicode) strings on 32-bit Windows platforms.

Boolean Logic Operand. In J.D. Edwards reporting programs, the parameter of the Relationship field. The Boolean logic operand instructs the system to compare certain records or parameters. Available options are:

EQ	Equal To.
LT	Less Than.
LE	Less Than or Equal To.
GT	Greater Than.
GE	Greater Than or Equal To.
NE	Not Equal To.
NL	Not Less Than.
NG	Not Greater Than.

browser. A client application that translates information sent by the World Wide Web. A client must use a browser to receive, manipulate, and display World Wide Web information on the desktop. Also known as a Web browser.

BSFN. See business function.

BSTR. See binary string.

BSVW. See business view.

business function. An encapsulated set of business rules and logic that can normally be reused by multiple applications. Business functions can execute a transaction or a subset of a transaction (check inventory, issue work orders, and so on). Business functions also contain the APIs that allow them to be called from a form, a database trigger, or a non-OneWorld application. Business functions can be combined with other business functions, forms, event rules, and other components to make up an application. Business functions can be created through event rules or third-generation languages, such as C. Examples of business functions include Credit Check and Item Availability.

business function event rule. See named event rule.

business view. Used by OneWorld applications to access data from database tables. A business view is a means for selecting specific columns from one or more tables whose data will be used in an application or report. It does not select specific rows and does not contain any physical data. It is strictly a view through which data can be handled.

Business View Design Aid (BDA). A OneWorld GUI ool for creating, modifying, copying, and printing business views. The tool uses a graphical user interface.

category code. In user defined codes, a temporary title for an undefined category. For example, if you are adding a code that designates different sales regions, you could change category code 4 to Sales Region, and define E (East), W (West), N (North), and S (South) as the valid codes. Sometimes referred to as reporting codes.

central objects. Objects that reside in a central location and consist of two parts: the central objects data source and central C components. The central objects data source contains OneWorld specifications, which are stored in a relational database. Central C components contain business function source, header, object, library, and DLL files and are usually stored in directories on the deployment server. Together they make up central objects.

check-in location. The directory structure location for the package and its set of replicated objects. This is usually \\deploymentserver\release\path_code\package\ packagename. The sub-directories under this path are where the central C components (source, include, object, library, and DLL file) for business functions are stored.

child. See parent/child form.

client/server. A relationship between processes running on separate machines. The server process is a provider of software services. The client is a consumer of those services. In essence, client/server provides a clean separation of function based on the idea of service. A server can service many clients at the same time and regulate their access to shared resources. There is a many-to-one relationship between clients and a server, respectively. Clients always initiate the dialog by requesting a service. Servers passively wait for requests from clients.

CNC. See configurable network computing.

component. In the ActivEra Portal, an encapsulated object that appears inside a workspace. Portal components

configurable client engine. Allows user flexibility at the interface level. Users can easily move columns, set tabs for different data views, and size grids according to their needs. The configurable client engine also enables the incorporation of Web browsers in addition to the Windows 95- and Windows NT-based interfaces.

configurable network computing. An application architecture that allows interactive and batch applications, composed of a single code base, to run across a TCP/IP network of multiple server platforms and SQL databases. The applications consist of reusable business functions and associated data that can be configured across the network dynamically. The overall objective for businesses is to provide a future-proof environment that enables them to change organizational structures, business processes, and technologies independently of each other.

constants. Parameters or codes that you set and the system uses to standardize information processing by associated programs. Some examples of constants are: validating bills of material online and including fixed labor overhead in costing.

control. Any data entry point allowing the user to interact with an application. For example, check boxes, pull-down lists, hyper-buttons, entry fields, and similar features are controls.

core. The central and foundation systems of J.D. Edwards software, including General Accounting, Accounts Payable, Accounts Receivable, Address Book, Financial Reporting, Financial Modeling and Allocations, and Back Office.

CRP. Conference Room Pilot.

custom gridlines. A grid row that does not come from the database, for example, totals. To display a total in a grid, sum the values and insert a custom gridline to display the total. Use the system function Insert Grid Row Buffer to accomplish this.

data dictionary. The OneWorld method for storing and managing data item definitions and specifications. J.D. Edwards has an active data dictionary, which means it is accessed at runtime.

data mart. Department-level decision support databases. They usually draw their data from an enterprise data warehouse that serves as a source of consolidated and reconciled data from around the organization. Data marts can be either relational or multidimensional databases.

data replication. In a replicated environment, multiple copies of data are maintained on multiple machines. There must be a single source that “owns” the data. This ensures that the latest copy of data can be applied to a primary place and then replicated as appropriate. This is in contrast to a simple copying of data, where the copy is not maintained from a central location, but exists independently of the source.

data source. A specific instance of a database management system running on a computer.

Data source management is accomplished through Object Configuration Manager (OCM) and Object Map (OM).

data structure. A group of data items that can be used for passing information between objects, for example, between two forms, between forms and business functions, or between reports and business functions.

data warehouse. A database used for reconciling and consolidating data from multiple databases before it is distributed to data marts for department-level decision support queries and reports. The data warehouse is generally a large relational database residing on a dedicated server between operational databases and the data marts.

data warehousing. Essentially, data warehousing involves off-loading operational data sources to target databases that will be used exclusively for decision support (reports and queries). There are a range of decision support environments, including duplicated database, enhanced analysis databases, and enterprise data warehouses.

database. A continuously updated collection of all information a system uses and stores. Databases make it possible to create, store, index, and cross-reference information online.

database driver. Software that connects an application to a specific database management system.

database server. A server that stores data. A database server does not have OneWorld logic.

DCE. See distributed computing environment.

DD. See data dictionary.

default. A code, number, or parameter value that is assumed when none is specified.

detail. The specific pieces of information and data that make up a record or transaction. Contrast with summary.

detail area. A control that is found in OneWorld applications and functions similarly to a spreadsheet grid for viewing, adding, or updating many rows of data at one time.

direct connect. A transaction method in which a client application communicates interactively and directly with a server application. See also batch-of-one immediate, store and forward.

director. An interactive utility that guides a user through the steps of a process to complete a task.

distributed computing environment (DCE). A set of integrated software services that allows software running on multiple computers to perform in a manner that is seamless and transparent to the end-users. DCE provides security, directory, time, remote procedure calls, and files across computers running on a network.

DLL. See dynamic link library.

DS. See data structure.

DSTR. See data structure.

duplicated database. A decision support database that contains a straightforward copy of operational data. The advantages involve improved performance for both operational and reporting environments. See also enhanced analysis database, enterprise data warehouse.

dynamic link library (DLL). A set of program modules that are designed to be invoked from executable files when the executable files are run, without having to be linked to the executable files. They typically contain commonly used functions.

dynamic partitioning. The ability to dynamically distribute logic or data to multiple tiers in a client/server architecture.

embedded event rule. An event rule that is specific to a particular table or application. Examples include form-to-form calls, hiding a field based on a processing option value, and calling a business function. Contrast with business function event rule. See also event rule.

employee work center. This is a central location for sending and receiving all OneWorld messages (system and user generated) regardless of the originating application or user. Each user has a mailbox that contains workflow and other messages, including Active Messages. With respect to workflow, the Message Center is MAPI compliant and supports drag and drop work reassignment, escalation, forward and reply, and workflow monitoring. All messages from the message center can be viewed through OneWorld messages or Microsoft Exchange.

encapsulation. The ability to confine access to and manipulation of data within an object to the procedures that contribute to the definition of that object.

enhanced analysis database. A database containing a subset of operational data. The data on the enhanced analysis database performs calculations and provides summary data to speed generation of reports and query response times. This solution is appropriate when external data must be added to source data, or when historical data is necessary for trend analysis or regulatory reporting. See also duplicated database, enterprise data warehouse.

enterprise data warehouse. A complex solution that involves data from many areas of the enterprise. This environment requires a large relational database (the data warehouse) that is a central repository of enterprise data, which is clean, reconciled, and consolidated. From this repository, data marts retrieve data to provide department-level decisions. See also duplicated database, enhanced analysis database.

enterprise server. A database server and logic server. See database server. Also referred to as host.

ER. See event rule.

ERP. See enterprise resource planning.

event. An action that occurs when an interactive or batch application is running. Example events are tabbing out of an edit control, clicking a push button, initializing a form, or performing a page break on a report. The GUI operating system uses miniprograms to manage user activities within a form. Additional logic can be attached to these miniprograms and used to give greater functionality to any event within a OneWorld application or report using event rules.

event rule. Used to create complex business logic without the difficult syntax that comes with many programming languages. These logic statements can be attached to applications or database events and are executed when the defined event occurs, such as entering a form, selecting a menu bar option, page breaking on a report, or selecting a record. An event rule can validate data, send a message to a user, call a business function, as well as many other actions. There are two types of event rules:

1 Embedded event rules.
2 Named event rules.

executable file. A computer program that can be run from the computer's operating system. Equivalent terms are "application" and "program.".

exit. 1) To interrupt or leave a computer program by pressing a specific key or a sequence of keys. 2) An option or function key displayed on a form that allows you to access another form.

facility. 1) A separate entity within a business for which you want to track costs. For example, a facility might be a warehouse location, job, project, work center, or branch/plant. Sometimes referred to as a business unit. 2) In Home Builder and ECS, a facility is a collection of computer language statements or programs that provide a specialized function throughout a system or throughout all integrated systems. For example, DREAM Writer and FASTR are facilities.

FDA. See Form Design Aid.

find/browse. A type of form used to:

1 Search, view, and select multiple records in a detail area.
2 Delete records.
3 Exit to another form.
4 Serve as an entry point for most applications.

firewall. A set of technologies that allows an enterprise to test, filter, and route all incoming messages. Firewalls are used to keep an enterprise secure.

fix/inspect. A type of form used to view, add, or modify existing records. A fix/inspect form has no detail area.

form. An element of OneWorld's graphical user interface that contains controls by which a user can interact with an application. Forms allow the user to input, select, and view information. A OneWorld application might contain multiple forms. In Microsoft Windows terminology, a form is known as a dialog box.

Form Design Aid (FDA). The OneWorld GUI development tool for building interactive applications and forms.

form interconnection. Allows one form to access and pass data to another form. Form interconnections can be attached to any event; however, they are normally used when a button is clicked.

form type. The following form types are available in OneWorld:

1 Find/browse.
2 Fix/inspect.
3 Header detail.
4 Headerless detail.
5 Message.
6 Parent/child.
7 Search/select.

fourth generation language (4GL). A programming language that focuses on what you need to do and then determines how to do it. Structured Query Language is an example of a 4GL.

graphical user interface (GUI). A computer interface that is graphically based as opposed to being character-based. An example of a character-based interface is that of the AS/400. An example of a GUI is Microsoft Windows. Graphically based interfaces allow pictures and other graphic images to be used in order to give people clues on how to operate the computer.

grid. See detail area.

GUI. See graphical user interface.

header. Information at the beginning of a table or form. This information is used to identify or provide control information for the group of records that follows.

header/detail. A type of form used to add, modify, or delete records from two different tables. The tables usually have a parent/child relationship.

headerless detail. A type of form used to work with multiple records in a detail area. The detail area is capable of of receiving input.

hidden selections. Menu selections you cannot see until you enter HS in a menu's Selection field. Although you cannot see these selections, they are available from any menu. They include such items as Display Submitted Jobs (33), Display User Job Queue (42), and Display User Print Queue (43). The Hidden Selections window displays three categories of selections: user tools, operator tools, and programmer tools.

host. In the centralized computer model, a large timesharing computer system that terminals communicate with and rely on for processing. In contrasts with client/server in that those users work at computers that perform much of their own processing and access servers that provide services such as file management, security, and printer management.

HTML. See hypertext markup language.

hypertext markup language. A markup language used to specify the logical structure of a document rather than the physical layout. Specifying logical structure makes any HTML document platform independent. You can view an HTML document on any desktop capable of supporting a browser. HTML can include active links to other HTML documents anywhere on the Internet or on intranet sites.

index. Represents both an ordering of values and a uniqueness of values that provide efficient access to data in rows of a table. An index is made up of one or more columns in the table.

inheritance. The ability of a class to recieve all or parts of the data and procedure definitions from a parent class. Inheritance enhances developement through the reuse of classes and their related code.

install system code. See system code.

integrated toolset. Unique to OneWorld is an industrial-strength toolset embedded in the already comprehensive business applications. This toolset is the same toolset used by J.D. Edwards to build OneWorld interactive and batch applications. Much more than a development environment, however, the OneWorld integrated toolset handles reporting and other batch processes, change management, and basic data warehousing facilities.

interactive processing. Processing actions that occur in response to commands you enter directly into the system. During interactive processing, you are in direct communication with the system, and it might prompt you for additional information while processing your request. See also online. Contrast with batch processing.

interface. A link between two or more computer systems that allows these systems to send information to and receive information from one another.

Internet. The worldwide constellation of servers, applications, and information available to a desktop client through a phone line or other type of remote access.

interoperability. The ability of different computer systems, networks, operating systems, and applications to work together and share information.

intranet. A small version of the Internet usually confined to one company or organization. An intranet uses the functionality of the Internet and places it at the disposal of a single enterprise.

IP. A connection-less communication protocol that by itself provides a datagram service. Datagrams are self-contained packets of information that are forwarded by routers based on their address and the routing table information contained in the routers. Every node on a TCP/IP network requires an address that identifies both a network and a local host or node on the network. In most cases the network administrator sets up these addresses when installing new workstations. In some cases, however, it is possible for a workstation, when booting up, to query a server for a dynamically assigned address.

IServer Service. Developed by J.D. Edwards, this internet server service resides on the web server, and is used to speed up delivery of the Java class files from the database to the client.

ISO 9000. A series of standards established by the International Organization for Standardization, designed as a measure of product and service quality.

J.D. Edwards Database. See JDEBASE Database Middleware.

Java. An Internet executable language that, like C, is designed to be highly portable across platforms. This programming language was developed by Sun Microsystems. Applets, or Java applications, can be accessed from a web browser and executed at the client, provided that the operating system or browser is Java-enabled. (Java is often described as a scaled-down C++). Java applications are platform independent.

Java Database Connectivity (JDBC). The standard way to access Java databases, set by Sun Microsystems. This standard allows you to use any JDBC driver database.

JavaScript. A scripting language related to Java. Unlike Java, however, JavaScript is not an object-oriented language and it is not compiled.

jde.ini. J.D. Edwards file (or member for AS/400) that provides the runtime settings required for OneWorld initialization. Specific versions of the file/member must reside on every machine running OneWorld. This includes workstations and servers.

JDEBASE Database Middleware. J.D. Edwards proprietary database middleware package that provides two primary benefits:

1. Platform-independent APIs for multidatabase access. These APIs are used in two ways:
   a. By the interactive and batch engines to dynamically generate platform-specific SQL, depending on the datasource request.
   b. As open APIs for advanced C business function writing. These APIs are then used by the engines to dynamically generate platform-specific SQL.
2. Client-to-server and server-to-server database access. To accomplish this OneWorld is integrated with a variety of third-party database drivers, such as Client Access 400 and open database connectivity (ODBC).

JDECallObject. An application programming interface used by business functions to invoke other business functions.

JDENET. J.D. Edwards proprietary middleware software. JDENET is a messaging software package.

JDENET communications middleware. J.D. Edwards proprietary communications middleware package for OneWorld. It is a peer-to-peer, message-based, socket based, multiprocess communications middleware solution. It handles client-to-server and server-to-server communications for all OneWorld supported platforms.

job queue. A group of jobs waiting to be batch processed. See also batch processing.

just in time installation (JITI). OneWorld's method of dynamically replicating objects from the central object location to a workstation.

just in time replication (JITR). OneWorld's method of replicating data to individual workstations. OneWorld replicates new records (inserts) only at the time the user needs the data. Changes, deletes, and updates must be replicated using Pull Replication.

KEY. A column or combination of columns that identify one or more records in a database table.

leading zeros. A series of zeros that certain facilities in J.D. Edwards systems place in front of a value you enter. This normally occurs when you enter a value that is smaller than the specified length of the field. For example, if you enter 4567 in a field that accommodates eight numbers, the facility places four zeros in front of the four numbers you enter. The result appears as: 00004567.

level of detail. 1) The degree of difficulty of a menu in J.D. Edwards software. The levels of detail for menus are as follows:

A	Major Product Directories.
B	Product Groups.
1	Basic Operations.
2	Intermediate Operations.
3	Advanced Operations.
4	Computer Operations.
5	Programmers.
6	Advanced Programmers Also known as menu levels.

2) The degree to which account information in the General Accounting system is summarized. The highest level of detail is 1 (least detailed) and the lowest level of detail is 9 (most detailed).

MAPI. See Messaging Application Programming Interface.

master table. A database table used to store data and information that is permanent and necessary to the system's operation. Master tables might contain data such as paid tax amounts, supplier names, addresses, employee information, and job information.

menu. A menu that displays numbered selections. Each of these selections represents a program or another menu. To access a selection from a menu, type the selection number and then press Enter.

menu levels. See level of detail.

menu masking. A security feature of J.D. Edwards systems that lets you prevent individual users from accessing specified menus or menu selections. The system does not display the menus or menu selections to unauthorized users.

Messaging Application Programming Interface (MAPI). An architecture that defines the components of a messaging system and how they behave. It also defines the interface between the messaging system and the components.

middleware. A general term that covers all the distributed software needed to support interactions between clients and servers. Think of it as the software that's in the middle of the client/server system or the "glue" that lets the client obtain a service from a server.

modal. A restrictive or limiting interaction created by a given condition of operation. Modal often describes a secondary window that restricts a user's interaction with other windows. A secondary window can be modal with respect to it's primary window or to the entire system. A modal dialog box must be closed by the user before the application continues.

mode. In reference to forms in OneWorld, mode has two meanings:

- An operational qualifier that governs how the form interacts with tables and business views. OneWorld form modes are: add, copy, and update.
- An arbitrary setting that aids in organizing form generation for different environments. For example, you might set forms generated for a Windows environment to mode 1 and forms generated for a Web environment to mode 2.

modeless. Not restricting or limiting interaction. Modeless often describes a secondary window that does not restrict a user's interaction with other windows. A modeless dialog box stays on the screen and is available for use at any time but also permits other user activities.

multitier architecture. A client/server architecture that allows multiple levels of processing. A tier defines the number of computers that can be used to complete some defined task.

named event rule. Encapsulated, reusable business logic created using through event rules rather than C programming. Contrast with embedded event rule. See also event rule.

NER. See named event rule.

network computer. As opposed to the personal computer, the network computer offers (in theory) lower cost of purchase and ownership and less complexity. Basically, it is a scaled-down PC (very little memory or disk space) that can be used to access network-based applications (Java applets, ActiveX controls) via a network browser.

network computing. Often referred to as the next phase of computing after client/server. While its exact definition remains obscure, it generally encompasses issues such as transparent access to computing resources, browser-style front-ends, platform independence, and other similar concepts.

next numbers. A feature you use to control the automatic numbering of such items as new G/L accounts, vouchers, and addresses. It lets you specify a numbering system and provides a method to increment numbers to reduce transposition and typing errors.

non–object librarian object. An object that is not managed by the object librarian.

numeric character. Digits 0 through 9 that are used to represent data. Contrast with alphanumeric characters.

object. A self-sufficient entity that contains data as well as the structures and functions used to manipulate the data. For OneWorld purposes, an object is a reusable entity that is based on software specifications created by the OneWorld toolset. See also object librarian.

object configuration manager (OCM). OneWorld's Object Request Broker and the control center for the runtime environment. It keeps track of the runtime locations for business functions, data, and batch applications. When one of these objects is called, the Object Configuration Manager directs access to it using defaults and overrides for a given environment and user.

object embedding. When an object is embedded in another document, an association is maintained between the object and the application that created it; however, any changes made to the object are also only kept in the compound document. See also object linking.

object librarian. A repository of all versions, applications, and business functions reusable in building applications. You access these objects with the Object Management Workbench.

object librarian object. An object managed by the object librarian.

object linking. When an object is linked to another document, a reference is created with the file the object is stored in, as well as with the application that created it. When the object is modified, either from the compound document or directly through the file it is saved in, the change is reflected in that application as well as anywhere it has been linked. See also object embedding.

object linking and embedding (OLE). A way to integrate objects from diverse applications, such as graphics, charts, spreadsheets, text, or an audio clip from a sound program. See also object embedding, object linking.

object management workbench (OMW). An application that provides check-out and check-in capabilities for developers, and aids in the creation, modification, and use of OneWorld Objects. The OMW supports multiple environments (such as production and development).

object-based technology (OBT). A technology that supports some of the main principles of object-oriented technology: classes, polymorphism, inheritance, or encapsulation.

object-oriented technology (OOT). Brings software development past procedural programming into a world of reusable programming that simplifies development of applications. Object orientation is based on the following principles: classes, polymorphism, inheritance, and encapsulation.

OCM. See object configuration manager.

ODBC. See open database connectivity.

OLE. See object linking and embedding.

OMW. Object Management Workbench.

OneWorld. A combined suite of comprehensive, mission-critical business applications and an embedded toolset for configuring those applications to unique business and technology requirements. OneWorld is built on the Configurable Network Computing technology- J.D. Edwards' own application architecture, which extends client/server functionality to new levels of configurability, adaptability, and stability.

OneWorld application. Interactive or batch processes that execute the business functionality of OneWorld. They consist of reusable business functions and associated data that are platform independent and can be dynamically configured across a TCP/IP network.

OneWorld object. A reusable piece of code that is used to build applications. Object types include tables, forms, business functions, data dictionary items, batch processes, business views, event rules, versions, data structures, and media objects. See also object.

OneWorld process. Allows OneWorld clients and servers to handle processing requests and execute transactions. A client runs one process, and servers can have multiple instances. OneWorld processes can also be dedicated to specific tasks (for example, workflow messages and data replication) to ensure that critical processes don't have to wait if the server is particularly busy.

OneWorld Web development computer. A standard OneWorld Windows developer computer with the additional components installed:

- JFC (0.5.1).
- Generator Package with Generator.Java and JDECOM.dll.
- R2 with interpretive and apllication controls/form.

online. Computer functions over which the system has continuous control. Users are online with the system when working with J.D. Edwards system provided forms.

open database connectivity (ODBC). Defines a standard interface for different technologies to process data between applications and different data sources. The ODBC interface is made up of a set of function calls, methods of connectivity, and representation of data types that define access to data sources.

open systems interconnection (OSI). The OSI model was developed by the International Standards Organization (ISO) in the early 1980s. It defines protocols and standards for the interconnection of computers and network equipment.

operand. See Boolean Logic Operand.

output. Information that the computer transfers from internal storage to an external device, such as a printer or a computer form.

output queue. See print queue.

package. OneWorld objects are installed to workstations in packages from the deployment server. A package can be compared to a bill of material or kit that indicates the necessary objects for that workstation and where on the deployment server the install program can find them. It is a point-in-time "snap shot" of the central objects on the deployment server.

package location. The directory structure location for the package and it's set of replicated objects. This is usually \\deployment server\release\path_code\package\ package name. The sub-directories under this path are where the replicated objects for the package will be placed. This is also referred to as where the package is built or stored.

parameter. A number, code, or character string you specify in association with a command or program. The computer uses parameters as additional input or to control the actions of the command or program.

parent/child form. A type of form that presents parent/child relationships in an application on one form. The left portion of the form presents a tree view that displays a visual representation of a parent/child relationship. The right portion of the form displays a detail area in browse mode. The detail area displays the records for the child item in the tree. The parent/child form supports drag and drop functionality.

partitioning. A technique for distributing data to local and remote sites to place data closer to the users who access. Portions of data can be copied to different database management systems.

path code. A pointer to a specific set of objects. A path code is used to locate:
1. Central Objects.
2. Replicated Objects.

platform independence. A benefit of open systems and Configurable Network Computing. Applications that are composed of a single code base can be run across a TCP/IP network consisting of various server platforms and SQL databases.

polymorphism. A principle of object-oriented technology in which a single mnemonic name can be used to perform similar operations on software objects of different types.

portability. Allows the same application to run on different operating systems and hardware platforms.

portal. A configurable Web object that provides information and links to the Web. Portals can be used as home pages and are typically used in conjunction with a Web browser.

primary key. A column or combination of columns that uniquely identifies each row in a table.

print queue. A list of tables, such as reports, that you have submitted to be written to an output device, such as a printer. The computer spools the tables until it writes them. After the computer writes the table, the system removes the table identifier from the list.

processing option. A feature of the J.D. Edwards reporting system that allows you to supply parameters to direct the functions of a program. For example, processing options allow you to specify defaults for certain form displays, control the format in which information prints on reports, change how a form displays information, and enter beginning dates.

program temporary fix (PTF). A representation of changes to J.D. Edwards software that your organization receives on magnetic tapes or diskettes.

project. An Object Management Workbench object used to organize objects in development.

published table. Also called a "Master" table, this is the central copy to be replicated to other machines. Resides on the "Publisher" machine. the Data Replication Publisher Table (F98DRPUB) identifies all of the Published Tables and their associated Publishers in the enterprise.

publisher. The server that is responsible for the Published Table. The Data Replication Publisher Table (F98DRPUB) identifies all of the Published Tables and their associated Publishers in the enterprise.

pull replication. One of the OneWorld methods for replicating data to individual workstations. Such machines are set up as Pull Subscribers using OneWorld's data replication tools. The only time Pull Subscribers are notified of changes, updates, and deletions is when they request such information. The request is in the form of a message that is sent, usually at startup, from the Pull Subscriber to the server machine that stores the Data Replication Pending Change Notification table (F98DRPCN).

purge. The process of removing records or data from a system table.

QBE. See query by example.

query by example (QBE). Located at the top of a detail area, it is used to search for data to be displayed in the detail area.

redundancy. Storing exact copies of data in multiple databases.

regenerable. Source code for OneWorld business functions can be regenerated from specifications (business function names). Regeneration occurs whenever an application is recompiled, eitherfor a new platform or when new functionality is added.

relationship. Links tables together and facilitates joining business views for use in an application or report. Relationships are created based on indexes.

release/release update. A "release" contains major new functionality, and a "release update" contains an accumulation of fixes and performance enhancements, but no new functionality.

replicated object. A copy or replicated set of the central objects must reside on each client and server that run OneWorld. The path code indicates the directory the directory where these objects are located.

run. To cause the computer system to perform a routine, process a batch of transactions, or carry out computer program instructions.

SAR. See software action request.

scalability. Allows software, architecture, network, or hardware growth that will support software as it grows in size or resource requirements. The ability to reach higher levels of performance by adding microprocessors.

search/select. A type of form used to search for a value and return it to the calling field.

selection. Found on J.D. Edwards menus, selections represent functions that you can access from a menu. To make a selection, type the associated number in the Selection field and press Enter.

server. Provides the essential functions for furnishings services to network users (or clients) and provides management functions for network administrators. Some of these functions are storage of user programs and data and management functions for the file systems. It may not be possible for one server to support all users with the required services. Some examples of dedicated servers that handle specific tasks are backup and archive servers, application and database servers.

servlet. Servlets provide a Java-based solution used to address the problems currently associated with doing server-side programming, including inextensible scripting solutions. Servlets are objects that conform to a specific interface that can be plugged into a Java-based server. Servlets are to the server-side what applets are to the client-side.

software. The operating system and application programs that tell the computer how and what tasks to perform.

software action request (SAR). An entry in the AS/400 database used for requesting modifications to J.D. Edwards software.

special character. A symbol used to represent data. Some examples are *, &, #, and /. Contrast with alphanumeric character and numeric character.

specifications. A complete description of a OneWorld object. Each object has its own specification, or name, which is used to build applications.

Specs. See specifications.

spool. The function by which the system stores generated output to await printing and processing.

spooled table. A holding file for output data waiting to be printed or input data waiting to be processed.

SQL. See structured query language.

static text. Short, descriptive text that appears next to a control variable or field. When the variable or field is enabled, the static text is black; when the variable or field is disabled, the static text is gray.

store and forward. A transaction method that allows a client application to perform work and, at a later time, complete that work by connecting to a server application. This often involves uploading data residing on a client to a server.

structured query language (SQL). A fourth generation language used as an industry standard for relational database access. It can be used to create databases and to retrieve, add, modify, or deleta data from databases. SQL is not a complete programming language because it does not contain control flow logic.

subfile. See detail.

submit. See run.

subscriber. The server that is responsible for the replicated copy of a Published Table. Such servers are identified in the Subscriber Table.

subscriber table. The Subscriber Table (F98DRSUB), which is stored on the Publisher Server with the Data Replication Publisher Table (F98DRPUB) identifies all of the Subscriber machines for each Published Table.

subsystem job. Within OneWorld, subsystem jobs are batch processes that continually run independently of, but asynchronously with, OneWorld applications.

summary. The presentation of data or information in a cumulative or totaled manner in which most of the details have been removed. Many of the J.D. Edwards systems offer forms and reports that are summaries of the information stored in certain tables. Contrast with detail.

system. See application.

System Code. System codes are a numerical representation of J.D. Edwards and customer systems. For example, 01 is the system code for Address Book. System codes 55 through 59 are reserved for customer development by customers. Use system codes to categorize within OneWorld. For example, when establishing user defined codes (UDCs), you must include the system code the best categorizes it. When naming objects such as applications, tables, and menus, the second and third characters in the object's name is the system code for that object. For example, G04 is the main menu for Acounts Payable, and 04 is its system code.

system function. A program module, provided by OneWorld, available to applications and reports for further processing.

table. A two-dimensional entity made up of rows and columns. All physical data in a database are stored in tables. A row in a table contains a record of related information. An example would be a record in an Employee table containing the Name, Address, Phone Number, Age, and Salary of an employee. Name is an example of a column in the employee table.

table design aid (TDA). A OneWorld GUI tool for creating, modifying, copying, and printing database tables.

table event rules. Use table event rules to attach database triggers (or programs) that automatically run whenever an action occurs against the table. An action against a table is referred to as an event. When you create a OneWorld database trigger, you must first determine which event will activate the trigger. Then, use Event Rules Design to create the trigger. Although OneWorld allows event rules to be attached to application events, this functionality is application specific. Table event rules provide embedded logic at the table level.

TAM. Table Access Management.

TBLE. See table.

TC. Table conversion.

TCP/IP. Transmission Control Protocol/Internet Protocol. The original TCP protocol was developed as a way to interconnect networks using many different types of transmission methods. TCP provides a way to establish a connection between end systems for the reliable delivery of messages and data.

TCP/IP services port. Used by a particular server application to provide whatever service the server is designed to provide. The port number must be readily known so that an application programmer can request it by name.

TDA. See table design aid.

TER. See table event rules.

Terminal Identification. The workstation ID number.Terminal number of a specific terminal or IBM user ID of a particular person for whom this is a valid profile.Header Field: Use the Skip to Terminal/User ID field in the upper portion of the form as an inquiry field in which you can enter the number of a terminal or the IBM user ID of a specific person whose profile you want the system to display at the top of the list. When you first access this form, the system automatically enters the user ID of the person signed on to the system.Detail Field: The Terminal/User ID field in the lower portion of the form contains the user ID of the person whose profile appears on the same line.A code identifying the user or terminal for which you accessed this window.

third generation language (3GL). A programming language that requires detailed information about how to complete a task. Examples of 3GLs are COBOL, C, Pascal and FORTRAN.

token. A referent to an object used to determine ownership of that object and to prevent non-owners from checking the object out in Object Management Workbench. An object holds its own token until the object is checked out, at which time the object passes its token to the project in which the object is placed.

trigger. Allow you to attach default processing to a data item in the data dictionary. When that data item is used on an application or report, the trigger is invoked by an event associated with the data item. OneWorld also has three visual assist triggers: calculator, calendar and search form.

UBE. Universal batch engine.

UDC Edit Control. Use a User-Defined Code (UDC) Edit Control for a field that accepts only specific values defined in a UDC table. Associate a UDC edit control with a database item or dictionary item. The visual assist Flashlight automatically appears adjacent to the UDC edit control field. When you click on the visual assist Flashlight, the attached search and select form displays valid values for the field. To create a UDC Edit Control, you must:

- Associate the data item with a specific UDC table in the Data Dictionary.
- Create a search and select form for displaying valid values from the UDC table.

uniform resource identifier (URI). A character string that references an internet object by name or location. A URL is a type of URI.

uniform resource locator (URL). Names the address (location) of a document on the Internet or an intranet. A URL includes the document's protocol and server name. The path to the document might be included as well. The following is an example of a URL: http://www.jdedwards.com. This is J.D. Edwards Internet address.

URI. See uniform resource identifier.

URL. See uniform resource locator.

user defined code (type). The identifier for a table of codes with a meaning you define for the system, such as ST for the Search Type codes table in Address Book. J.D. Edwards systems provide a number of these tables and allow you to create and define tables of your own. User defined codes were formerly known as descriptive titles.

user defined codes (UDC). Codes within software that users can define, relate to code descriptions, and assign valid values. Sometimes user defined codes are referred to as a generic code table. Examples of such codes are unit-of-measure codes, state names, and employee type codes.

UTB. Universal Table Browser.

valid codes. The allowed codes, amounts, or types of data that you can enter in a field. The system verifies the information you enter against the list of valid codes.

visual assist. Forms that can be invoked from a control to assist the user in determining what data belongs in the control.

vocabulary overrides. A feature you can use to override field, row, or column title text on forms and reports.

wchar_t. Internal type of a wide character. Used for writing portable programs for international markets.

web client. Any workstation that contains an internet browser. The web client communicates with the web server for OneWorld data.

web server. Any workstation that contains the IServer service, SQL server, Java menus and applications, and Internet middleware. The web server receives data from the web client, and passes the request to the enterprise server. When the enterprise server processes the information, it sends it back to the web server, and the web server sends it back to the web client.

WF. See workflow.

window. See form.

workflow. According to the Workflow Management Coalition, worlflow means "the automation of a business process, in whole or part, during which documents, information, or tasks are passed from one participant to another for action, according to a set of procedural rules.".

workgroup server. A remote database server usually containing subsets of data replicated from a master database server. This server does not performance an application or batch processing. It may or may not have OneWorld running (in order to replicate data).

workspace. In the ActivEra Portal, the main section of the Portal. A user might have access to several workspaces, each one configured differently and containing its own components.

worldwide web. A part of the Internet that can transmit text, graphics, audio, and video. The

World Wide Web allows clients to launch local or remote applications.

z file. For store and forward (network disconnected) user, OneWorld store and forward applications perform edits on static data and other critical information that must be valid to process an order. After the initial edits are complete, OneWorld stores the transactions in work tables on the workstation. These work table are called Z files. When a network connection is established, Z files are uploaded to the enterprise server and the transactions are edited again by a master business function. The master business function will then update the records in your transaction files.

Index

Index