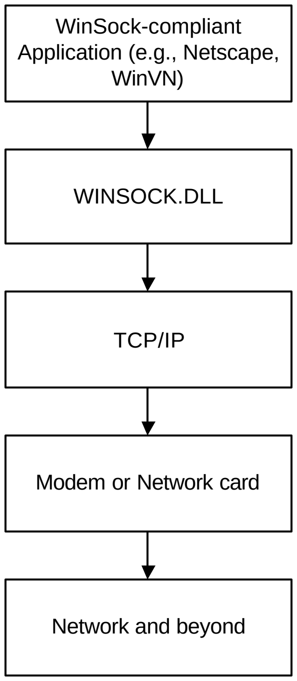
Winsock

What is Winsock?

WinSock is short for Windows Sockets, and is used as the interface between TCP/IP and Windows. TCP/IP has been called "the language of the Internet" and rightly so - most of the Internet is comprised of systems that use TCP/IP to talk to one another. Today's most popular Internet applications for Microsoft Windows and IBM OS/2 are developed according to the WinSock standard.

WinSock is a .DLL (Dynamic Link Library) and runs under Windows. WINSOCK.DLL is the interface to TCP/IP and, from there, on out to the Internet.

The easiest way to show how it works is with a diagram:



WINSOCK.DLL actually acts as a "layer" between your WinSock applications and your TCP/IP stack. Your WinSock applications tell WINSOCK.DLL what to do, WINSOCK.DLL translates these commands to your TCP/IP stack, and your TCP/IP stack passes them on to the Internet!

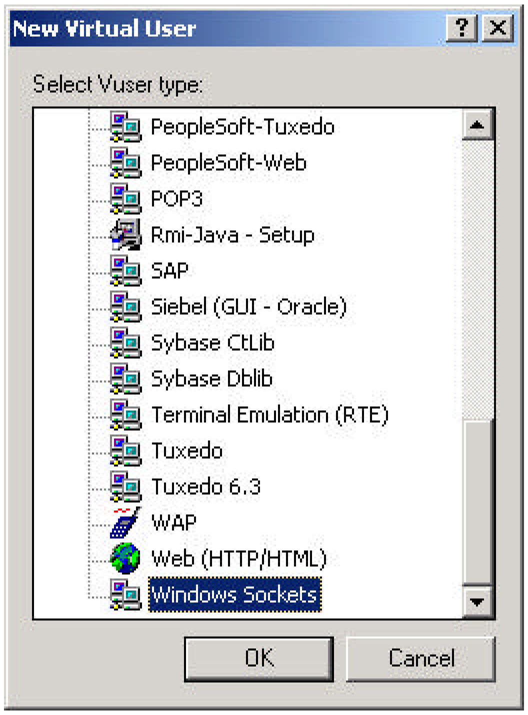
LoadRunner and Winsock

Before we launch into the need for LR Winsock solution, lets digress a little and discuss how various protocols work. From the above article introduction, you can gather that there are a number of higher-level protocols like FTP, HTTP etc. All Windows based applications like IE, WS-FTP use Winsock as an interface to communicate to the network. So any higher-level protocol will pass thru Winsock to get on to the network.

When do you use WinSock in LR? By now, you must be familiar with how LR works. LR essentially captures API calls during recording and plays them back. So when you create a LR web script, Vugen captures all the HTTP calls made from the browser. There are a number of other protocols like Oracle OCI, ODBC, Sybase, and SAP etc. that LR supports. In each of these cases, LR has hooks into the API to be able to capture the corresponding API calls. Now imagine a protocol that is not supported by LR. Since most network protocols use Winsock as an interface, we would be able to capture almost all application on Windows if we record at the Winsock layer. So we use Winsock when other protocols do not work.

Recording with Winsock

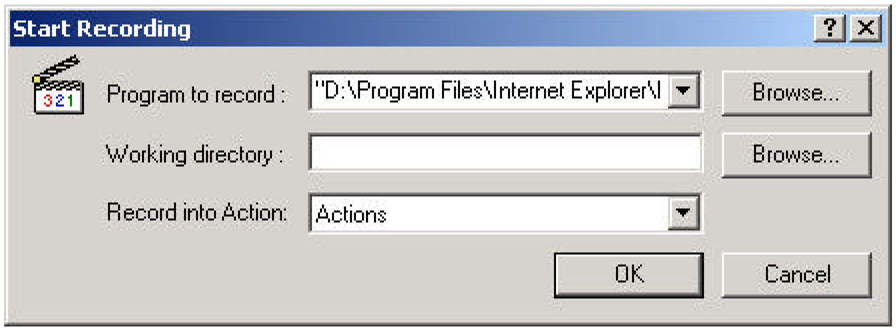
Create a new Vuser script in LR, and select Winsock as the Vuser type.


New Virtual User

Select Vuser type:

- PeopleSoft-Tuxedo
- PeopleSoft-Web
- POP3
- Rmi-Java - Setup
- SAP
- Siebel (GUI - Oracle)
- Sybase CtLib
- Sybase Dblib
- Terminal Emulation (RTE)
- Tuxedo
- Tuxedo 6.3
- WAP
- Web (HTTP/HTML)
- Windows Sockets

In web Vuser scripts, you specify the URL to record, and Vugen launches the browser and starts recording. But in Winsock, you could be recording against any application, not just a browser. So you need to specify the application to be recorded as shown below.


Start Recording

Program to record : "D:\Program Files\Internet Explorer\I

Working directory :

Record into Action: Actions

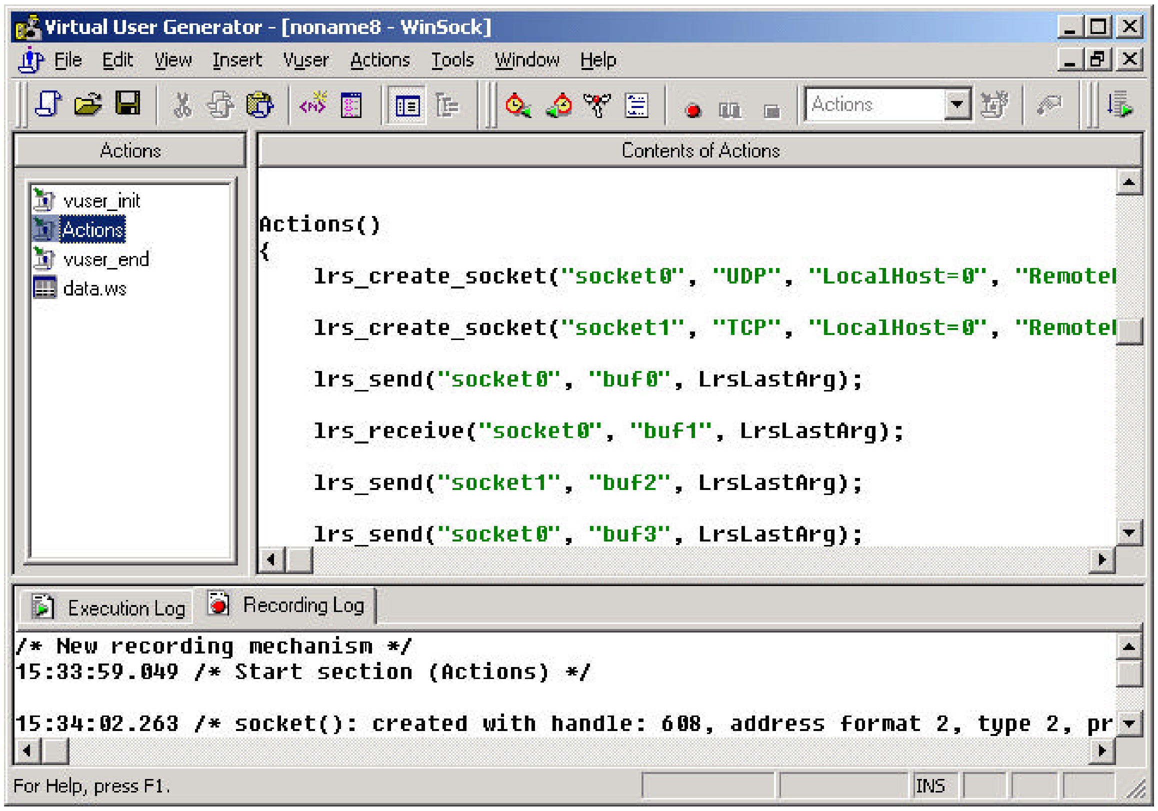
In this example, we are using creating a Winsock script for a web application, so we are invoking the browser by giving the path to IEXPLORE.EXE. This doesn't make much sense, because LR already supports HTTP recording, but I chose a web application just for ease of illustration.

Winsock scripts

A typical winsock script may look like this:

```
lrs_create_socket("socket0", "UDP", "LocalHost=0", "RemoteHost=doors:2084",  LrsLastArg);

lrs_create_socket("socket1", "TCP", "LocalHost=0", "RemoteHost=www2.yahoo.com:80",  LrsLastArg);

lrs_send("socket0", "buf0", LrsLastArg);

lrs_receive("socket0", "buf1", LrsLastArg);

lrs_send("socket1", "buf2", LrsLastArg);

lrs_send("socket0", "buf3", LrsLastArg);

lrs_receive("socket0", "buf4", LrsLastArg);

 …………
 …………
```

This is a recording against www.yahoo.com. As you can see, Winsock recordings primarily consist of opening a socket connection, sending buffers and receiving buffers. If you look at the script, you will find 4 sections instead of the usual 3 that you see in web scripts.



```
Actions()
{
	lrs_create_socket("socket0", "UDP", "LocalHost=0", "RemoteI
	
	lrs_create_socket("socket1", "TCP", "LocalHost=0", "RemoteI
	
	lrs_send("socket0", "buf0", LrsLastArg);
	
	lrs_receive("socket0", "buf1", LrsLastArg);
	
	lrs_send("socket1", "buf2", LrsLastArg);
	
	lrs_send("socket0", "buf3", LrsLastArg);
```

```
/* New recording mechanism */
15:33:59.049 /* Start section (Actions) */

15:34:02.263 /* socket(): created with handle: 608, address format 2, type 2, pr
```

The fourth section, called data.ws contains the buffers referenced in the lrs_send and lrs_receive statements in Actions.

Here is a sample of the buffers in data.ws –

```
send  buf0
            "!"

recv  buf1 1
            "!"

send  buf2
            "GET / HTTP/1.1\r\n"
            "Accept: */*\r\n"
            "Accept-Language: en-us\r\n"
            "Accept-Encoding: gzip, deflate\r\n"
            "User-Agent: Mozilla/4.0 (compatible; MSIE 5.01; Windows NT 5.0)\r\n"
            "Host: www.yahoo.com\r\n"
            "Connection: Keep-Alive\r\n"
            "Cookie: B=5oj83bst12l6m&b=2; Y=v=1&n=8lln5lmi3f8g1&l=6ef8c0d34b0/o&p=m2a2s"
            "qa4110004&r=3f&lg=us&intl=us; T=z=4TVE6A4ZqE6A9dIIpt30.NQNTYGNDE3NTYwM081&"
            "a=AAE&sk=DAAEwinHlDtEm/&d=c2wBTWpFQk16WXdNakUzTkRneQFhAUFBRQF6egE0VFZFNkFn"
            "V0E-; I=i1=010g1q1u24252e2i2k2p2r494a4g4m4q55565b5g6g6t6u7172737678797a7f7"
            "g7k7n7o888f8k8p8q989c9f9i9k9l9n9qacanapb0b3bdbfbhblbqbrbuc0c1c4cgcmcscteie"
            "jgr&ir=73\r\n"
            "\r\n"

send  buf3
            "!"

recv  buf4 1
            "!"

recv  buf5 2048
            "HTTP/1.0 200 OK\r\n"
            "Content-Length: 16030\r\n"
            "Content-Type: text/html\r\n"
            "\r\n"
            "<html><head><titl
```

Buf2 contains the request made for www.yahoo.com, and buf5 contains the response received from the server. As you can see, Winsock is not as pretty as Web! So we use this as a last resort. The number next to bufxx on a receive buffer denotes the total number of bytes received. For example, 2048 buffers were received on buf5.

NOTE: You will see a number of buffers that contain only an "!", that is just a network acknowledgement. That does not contribute anything to the application. You can comment these out. Commenting out the buffers in the data.ws will not help, you will have to comment out the corresponding lrs_send and lrs_receive buffers in the Actions sections. So in this example, buffers 0, 1, 3 and 4 can be commented out from the actions section like this:

```
  lrs_create_socket("socket0", "UDP", "LocalHost=0", "RemoteHost=doors:2084",  LrsLastArg);

   lrs_create_socket("socket1", "TCP", "LocalHost=0", "RemoteHost=www2.yahoo.com:80",  LrsLastArg);

  // lrs_send("socket0", "buf0", LrsLastArg);

 //  lrs_receive("socket0", "buf1", LrsLastArg);
```

```
  lrs_send("socket1", "buf2", LrsLastArg);

//  lrs_send("socket0", "buf3", LrsLastArg);

//  lrs_receive("socket0", "buf4", LrsLastArg);

  lrs_receive("socket1", "buf5", LrsLastArg);

  lrs_send("socket0", "buf6", LrsLastArg);

  ………..
  ………..
```

This will help run your scripts faster.

Creating Winsock Scripts

The following are the six steps involved in creating scripts (that work!).

1. Record the basic script using VuGen.
2. Enhance the script.
3. Define parameters (optional).
4. Correlate statements (optional).
5. Configure the run- time settings.
6. Run the script from VuGen.

1. Record the basic script using Vugen.

Invoke Vugen and create a new Vuser script as shown earlier in this chapter. This will be your basic script. You can comment out the buffers containing "!".

Exercise 1
As an exercise, create a simple Winsock script that logs into the MercuryWebTours site and logs off using jojo/bean. Save this script as Winsock_1. Create a similar script in web,you're your recording options to HTTP. Save this as WebWinsock_1. Do not capture and replace the dynamic session ID. Leave the script in the recorded state. Compare these two to get an idea of what Winsock scripts look like as compared to web scripts.

When you logged in successfully, you get a message that says "Welcome, jojo, ……". Since all the send and receive buffers are recorded in the data.ws file, you should be able to locate the string "Welcome, jojo" in your data.ws file.

1.1) *What is the buffer number in which you found "Welcome, jojo"? Can you explain why it was in a receive buffer and not a send buffer? And did the string appear exactly as "Welcome, jojo" or was it tagged with HTML?*

Take a closer look at the script. Search for MSO=SIDxxxxxxxxx in the data.ws file, where xxxxxxxxx is a 9 digit number. This is the total number of seconds elapsed since January 1 st, 1970. The MercuryWebTours site works based on the current time, and the cookie and session ID are based on the time. So timestamp (xxxxxxxxx) in your script corresponds to the time that you recorded. You will have to change this so as to capture the current time when a script is run, and convert the current time into a string and then a parameter. There is a function called time() in C that returns as an integer the number of seconds elapsed since Jan 1 st 1970.

Capture the time at the beginning of the script, convert this integer into a LR parameter, and replace all instances of this number in the data.ws file with the LR parameter. So this script will be set to use the current time.

1.2) Run the above script after making modifications for time. Search the execution log for the string ”incorrectly”. You will find “You've reached this page incorrectly”. Run the Script WebWinsock_1. You will see a similar message in the Runtime Viewer. Why did this happen?

NOTE: The receive buffers in data.ws file are not updated for each replay. The data.ws file is created during recording and remains intact during replay. So the content of the receive buffers is not of much interest once the script is recorded and working. LR uses the data in the send buffers to send requests to the network, and compares the responses received from the server to the responses as expected in the receive buffers. Also note that it only compares the size, not the actual content. If the receive buffer in data.ws has “rob” in it, and the server responds with a “bob”, LR will consider this as a success. But if the server responds with a “robert”, then LR will show that a mismatch occurred in the number of bytes expected.

Along the same lines, lets say LR is expecting to receive 500 bytes. Say it received the first 100 bytes in the first 10 seconds, it will timeout at that point and proceed further. So if you want LR to wait for a longer period of time, you will have to use lrs_set_recv_timeout function to modify the default timeout.

Consider the reverse situation, where you do not want to receive all the data on a given buffer, but only the first 100 bytes. Then you can use ltr_receive_ex function where you have the ability to specify the number of bytes that you want to receive.

2. Enhance the script.

Enhance the Vuser script by inserting transactions, rendezvous points, and control- flow structures into the script. Unlike web scripts, which are very readable, the steps in Winsock scripts are not too obvious. So it is very important that you insert comments, transactions and rendezvous points (if required) into the script while recording.

If there is any logic required in the script, insert the logic (This is a statement in general, does not apply to the above script, Winsock_1).

3. Define parameters (optional).

Define parameters for the fixed- values recorded into your script. By substituting fixed- values with parameters, you can repeat the same business process many times using different values. For example, in the above script, you could have replaced jojo/bean with different userIDs/passwords.

4. Correlate statements (optional).

Correlating statements enables you to use the result of one business process in a subsequent one. From the web scripts that you have done in the previous lessons, you know by now that the session ID generated by the server will have to be correlated so that the script works. That applies to Winsock scripts as well. So you will have to capture and correlate the session ID.

We will not do an exercise here on capturing and correlating the session ID, because a similar exercise is part of your certification. But we will discuss general aspects of correlation in this section.

For example, lets say you have to capture the PID from the following receive buffer **(This is an example from LR function reference):**

```
        "\r"
        "\x0 blah blah blah "
        "\r\n blah blah blah "
        "PID TT STAT TIME COMMAND\r\n PID 28469 q2"
        " S 0:01 -tcsh (tcsh)\r\n"
        ……
```

In a typical web script, you would use a web_create_html_param and use "PID " and " q2" as the boundaries to capture the data.

In a Winsock script, you would use lrs_save_param that saves data from a static or received buffer to a parameter. See the example below:

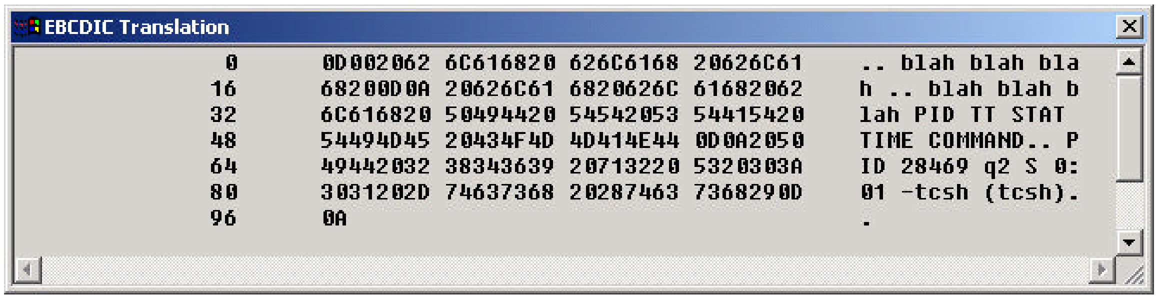
```
lrs_receive("socket2", "buf47", LrsLastArg);
lrs_save_param("socket2", NULL, "param1", 67, 5);
```

Unlike web_create_html_param, lrs_save_param goes after the request is made. In this example, the first statement is receiving buf47. The following are the arguments used in the lrs_save_param statement:

socket2: Capture from a buffer on socket2
NULL: NULL indicates that the parameter should be captured from the last received buffer. In this case it is buf47. If you were to capture from any other buffer, you will have to specify the buffer number.
param1: Name of the LR parameter in which to staore the captured value
67: Offset – explained below.
5: length of the to be captured

Offset: This is the number of bytes from the beginning of the buffer to start capturing. In the above example, the PID starts after 67 bytes from the beginning of buf47. How do you determine this?

Highlight the whole buffer in data.ws that contains the parameter to be captured and hit F7. This will bring up a window as shown below:



```
   0    0D002062 6C616820 626C6168 20626C61    .. blah blah bla
  16    68200D0A 20626C61 6820626C 61682062    h .. blah blah b
  32    6C616820 50494420 54542053 54415420    lah PID TT STAT
  48    54494D45 20434F4D 4D414E44 0D0A2050    TIME COMMAND.. P
  64    49442032 38343639 20713220 5320303A    ID 28469 q2 S 0:
  80    3031202D 74637368 20287463 7368290D    01 -tcsh (tcsh).
  96    0A                                     .
```

In the left column, you will see the offset that corresponds to that line. In the middle 4 columns, you see the EBCDIC translation of the buffer. And in the right column, you have the actual buffer itself. So if you look at the fifth line in the actual buffer, that is where you PID lies. But we want to start capturing after the third location in that line. The offset for that entire line is 64, and the offset for the PID within that line is 3, which makes the offset 67. (Now does that explain why I had to add all those blah blah blahs in the buffer to make it match my example? ☺)

NOTE: These is no statement similar to web_find in Winsock, you will have to capture the text you are trying to find and programmatically verify it that you got the correct text.

5. Configure the run- time settings.

The run- time settings control the Vuser behavior during script execution. These settings include loop, log, and timing information.

6. Run the script from Vugen.

Save and run the script from Vugen to verify that it runs correctly.

HAPPY WINSOCK !!

ANSWERS

1.1) *What is the buffer number in which you found “Welcome, jojo”? Can you explain why it was in a receive buffer and not a send buffer? And did the string appear exactly as “Welcome, jojo” or was it tagged with HTML?*

Buffer numbers do not always remain constant: they may change between recordings. When you login, based on the ID used, a welcome message is received from the server. Hence the string “Welcome, jojo” was found in a receive buffer. And the string would appear tagged in HTML. If you need a part of the string, you will have to programmatically extract it.

1.2) *Run the above script after making modifications for time. Search the execution log for the string ”incorrectly”. You will find “You've reached this page incorrectly”. Run the Script WebWinsock_1. You will see a similar message in the Runtime Viewer. Why did this happen?*

Because the session ID has not been correlated. We are trying to use a old session ID that is not valid anymore.