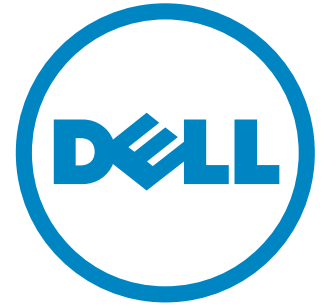


---

# Performance Testing

## - Advanced Load Runner Training



# Why Load Test An Application

- Does the application respond quickly enough for the intended users?
- Will the application handle the expected user load and beyond?
- Will the application handle the number of transactions required by the business?
- Is the application stable under expected and unexpected user loads?



# Functional vs. Load Web Testing

## *Functional test*

OBJECTIVE	EXAMPLE
Functionality	Do business processes function properly after implementation?

## *Load test*

OBJECTIVE	EXAMPLE
Stability	Will 2,000 concurrent hits crash the server?
Performance	Is response time acceptable according to specifications?
Functionality under load	Do business processes function properly under heavy load?

# Types of Performance Testing

## *Component Testing*

Find the behavior and performance of each tier.

## *Load Testing*

Find out whether the system can handle the expected load upon deployment under real-world conditions.

## *Stress Testing*

Find the application's breaking point. Apply testing that measures whether the application's environment is properly configured to handle expected or potentially unexpected high transaction volumes.

## *Volume Testing*

Find the stability of the system with respect to handling large amounts of data over extended time periods.



# Objectives of Performance Testing

## *Application Response Time*

How long does it take to complete a task?

## *Reliability*

How Stable is the system under a heavy work load?

## *Configuration Sizing*

Which configuration provides the best performance level?

## *Capacity Planning*

At what point does degradation in performance occur?

## *Acceptance*

Is the system stable enough to go into Production?

## *Bottleneck Identification*

What is the cause of degradation in performance?

## *Regression*

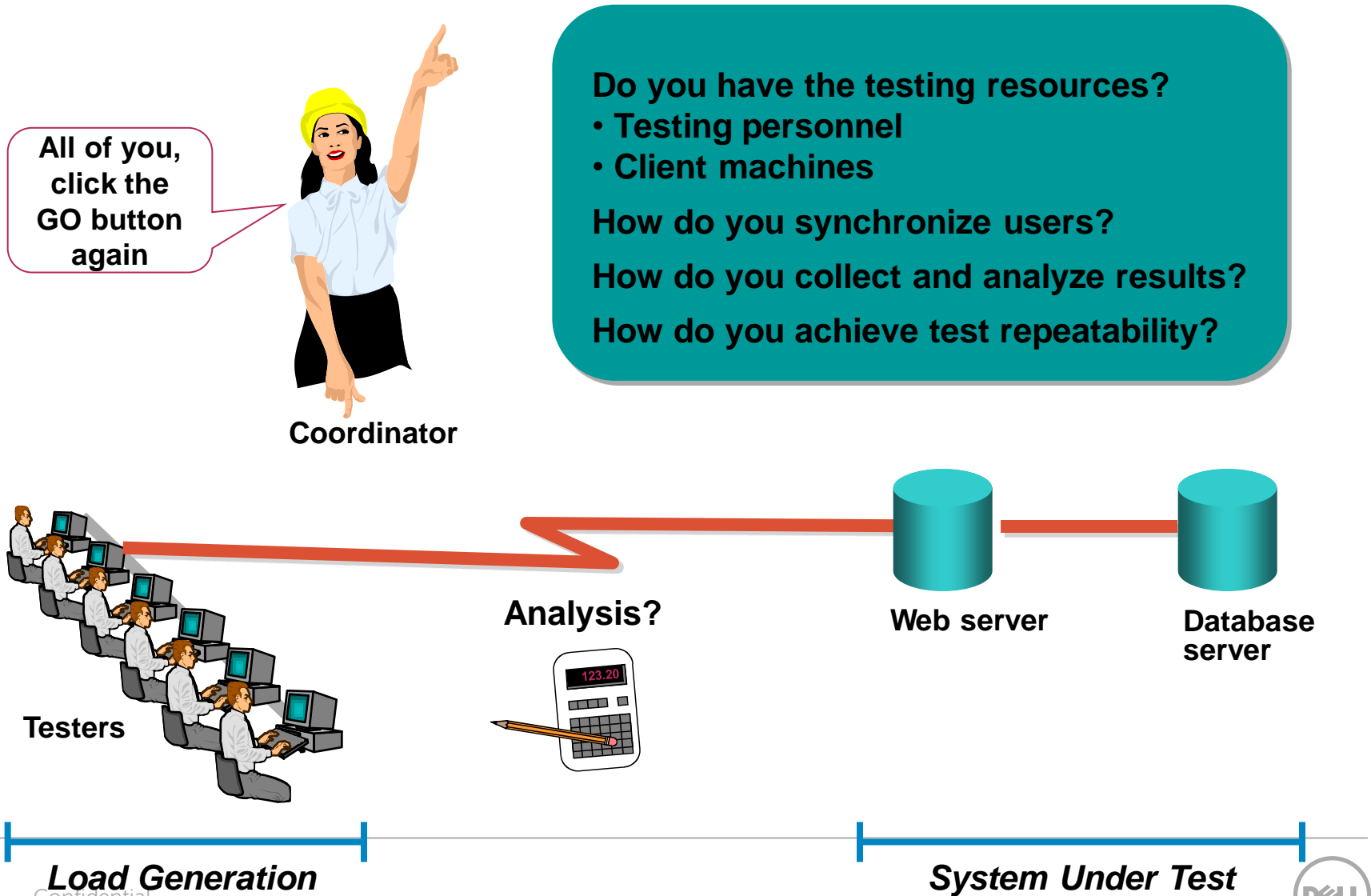
Does the new version of Software adversely affect response time?

## *Product Evaluation*

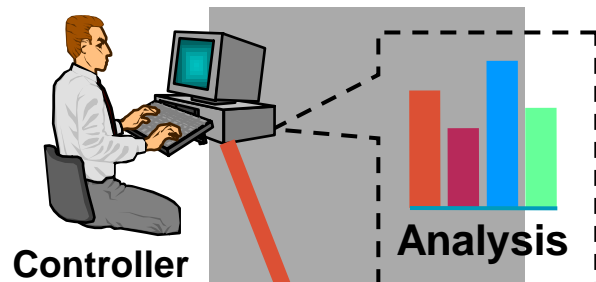
What is the best server for 100 users?



# Manual Testing Is Problematic

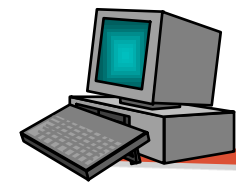


# The LoadRunner Solution

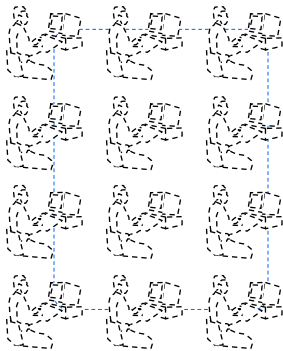


## Overcomes resource limitations

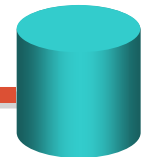
- Replaces testers with “Virtual Users”
- Runs many Vusers on few machines
- Controller manages the Vusers
- Meaningful results with analysis tools
- Repeats tests with scripted actions



**Vuser host**



**Web server**



**Database server**

**Load Generation**

**System Under Test**

# Load Testing Tools Available

- LoadRunner : HP (Formerly Mercury Interactive)
- e-Load :Emprix
- Silk Performer : Borland (Seague)
- QALoad : Compuware
- Rational Performance Tester : IBM Rational
- Web Load : Radview
- Neo Load : Neotys
- Open STA : Open Source.





# Introduction to Load Runner

- Load Runner is a HP (Mercury Interactive) Tool that predicts performance and behavior of the system
- By creating lots of load, you can see how the system reacts at peak levels or with simultaneous Users
- To test the application, LoadRunner emulates an environment where multiple users work concurrently. While the application is under load, LoadRunner accurately measures and analyzes the system performance, and its functionality



# Supporting Environments

- **Application Deployment Solution** - The Citrix protocol.
- **Client/Server** - MS SQL, ODBC, Oracle Web Applications 11i, DB2 CLI, Sybase Ctlb, Sybase Dblib, Windows Sockets, and DNS protocols.
- **Custom** - C templates, Visual Basic templates, Java templates, Javascript, and VBScript type scripts.
- **Distributed Components** - COM/DCOM, Corba-Java, and Rmi-Java protocols.
- **E-Business** - FTP, LDAP, Palm, Web (HTTP/HTML), Web Services, and the dual Web/Winsocket protocols.
- **Enterprise Java Beans** -EJB Testing and RMI-Java protocols.
- **ERP/CRM** - Baan, Oracle NCA, Peoplesoft 8, Peoplesoft-Tuxedo, SAP-Web, SAPGUI, SAPGUI/SAP-Web dual, and Siebel (Siebel-DB2 CLI, Siebel-MSSQL, Siebel-Web, and Siebel-Oracle) protocols.

# Supporting Environments

- Legacy

Terminal Emulation (RTE).

- Mailing Services

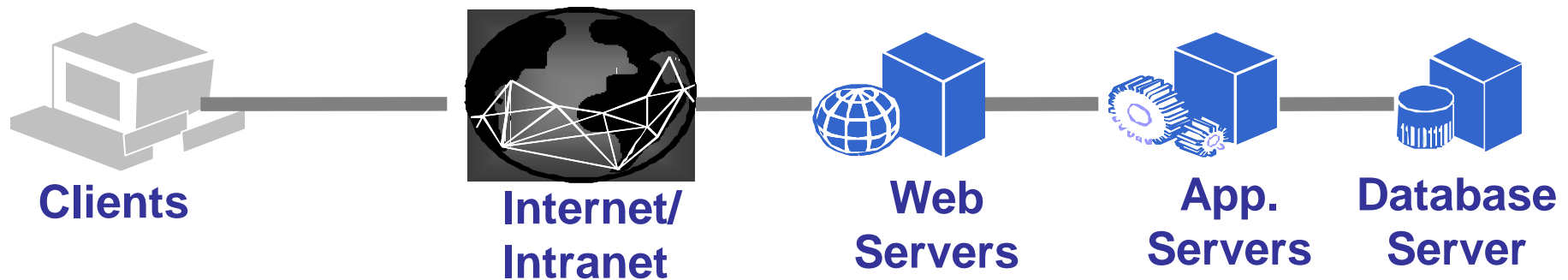
Internet Messaging (IMAP), MS Exchange (MAPI), POP3, and SMTP.

- Streaming

MediaPlayer and RealPlayer protocols.

- Wireless

i-Mode, VoiceXML, and WAP protocols.



# Supporting Environments

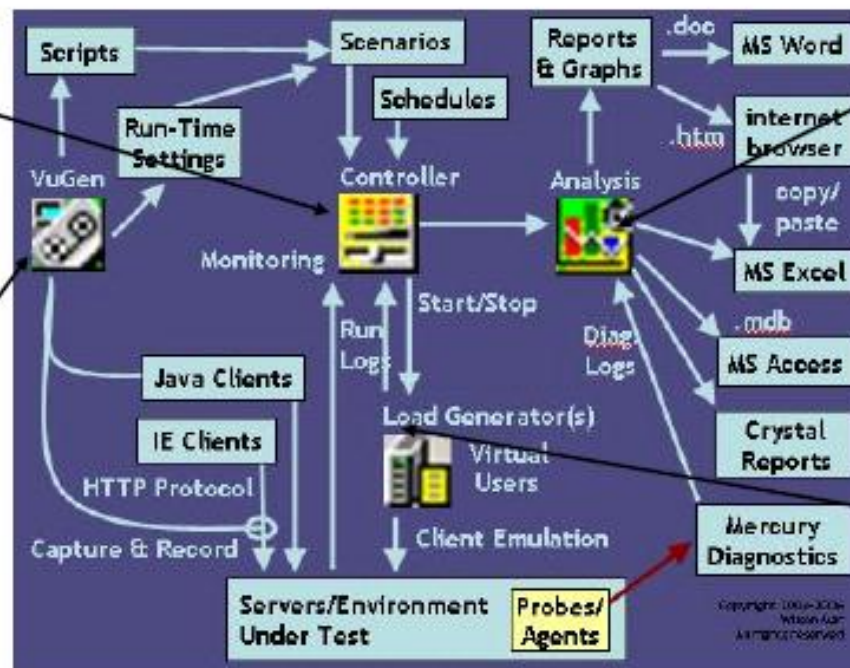
- Platforms
  - NT, 2000, XP
  - Sun
  - HP
  - IBM
  - Linux



# Load Runner Components

The Controller organizes, drives, manages, and monitors the load test.

The Virtual User Generator captures end-user business processes and creates an automated performance testing script, also known as a virtual user script.



The Analysis helps you view, dissect, and compare the performance results.

The Load Generators create the load by running virtual users.

# The LoadRunner Solution

*Virtual User Generator*

**Creates Scripts as one Single User.**

*LoadRunner Controller*

**Generates load and collects test results**

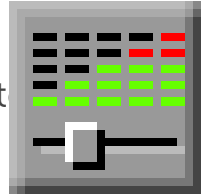
*LoadRunner Analysis*

**Compiles and displays test results with graphical and statistical tools**

# LoadRunner Terminology

- **Scenarios**

- Using LoadRunner, you divide your application performance testing requirements into scenarios.
- A scenario defines the events that occur during each testing sessions.
- For example, a scenario defines and controls the number of users to emulate, the actions that they perform, and the machines on which they run their emulations.



- **Vusers**

- In a scenario, LoadRunner replaces human users with virtual users or Vusers.
- When you run a scenario, Vusers emulate the actions of human users—submitting input to the server.
- A scenario can contain tens, hundreds, or even thousands of Vusers.



# LoadRunner Terminology

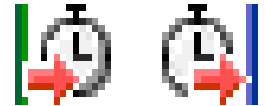
- **Vuser Scripts**

- The actions that a Vuser performs during the scenario are described in a Vuser script.
- When you run a scenario, each Vuser executes a Vuser script. Vuser scripts include functions that measure and record the performance of the server during the scenario.



- **Transactions**

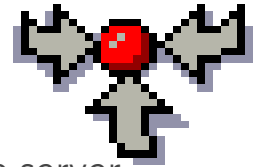
- To measure the performance of the server, you define transactions.
- Transactions measure the time that it takes for the server to respond to tasks submitted by Vusers.



Contd....



# LoadRunner Terminology

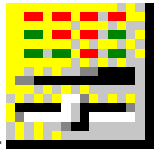


- **Rendezvous Points**

- You insert rendezvous points into Vuser scripts to emulate heavy user load on the server.
- Rendezvous points instruct multiple Vusers to perform tasks at exactly the same time.
- For example, to emulate peak load on the bank server, you insert a rendezvous point to instruct 100 Vusers to simultaneously deposit cash into their accounts.

- **Controller**

- You use the LoadRunner Controller to manage and maintain your scenarios.
- Using the Controller, you control all the Vusers in a scenario from a single workstation.



---

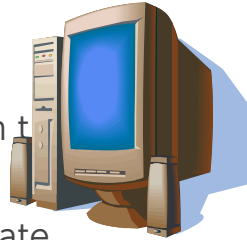
Contd....



# LoadRunner Terminology

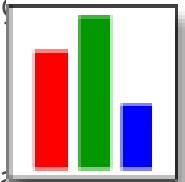
- **Hosts**

- When you execute a scenario, the LoadRunner Controller distributes each Vuser in the scenario to a host.
- The host is the machine that executes the Vuser script, enabling the Vuser to emulate the actions of a human user.

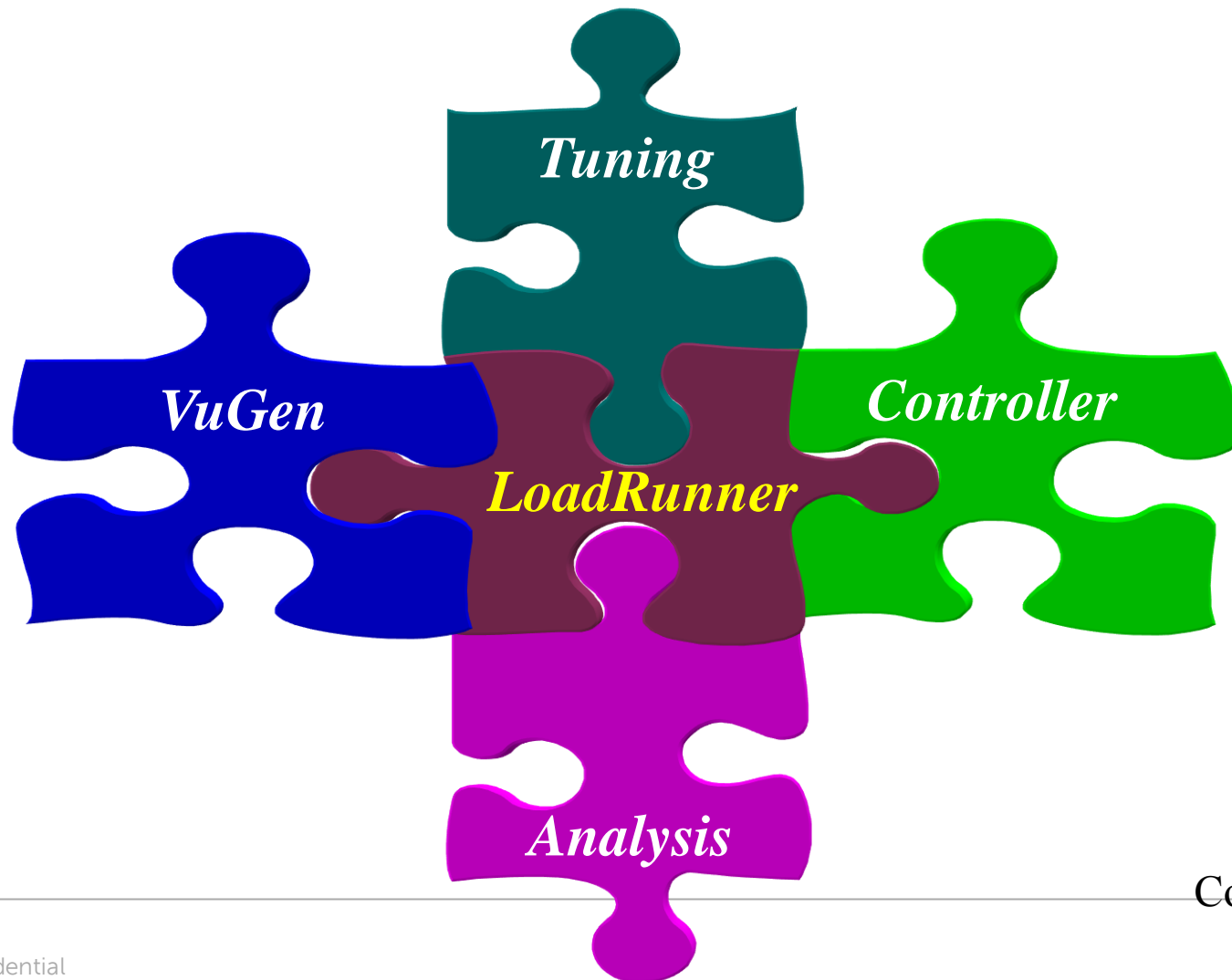


- **Performance Analysis**

- Vuser scripts include functions that measure and record system performance during load-testing sessions.
- During a scenario run, you can monitor the network and server resources.
- Following a scenario run, you can view performance analysis data in reports and graphs.



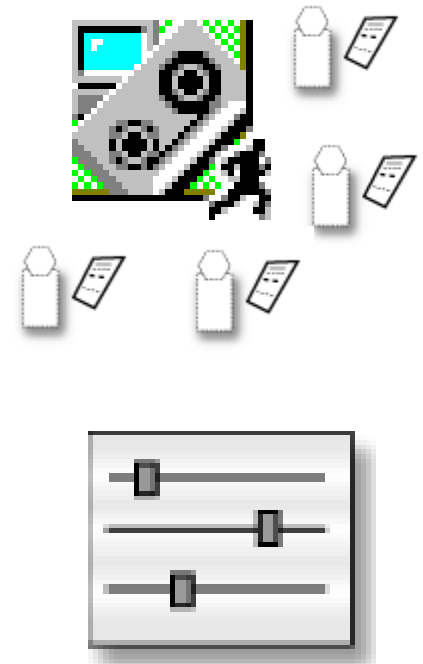
# LoadRunner Components



Contd....

# Components of LoadRunner 8.0

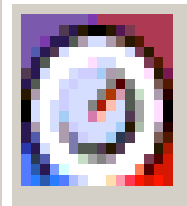
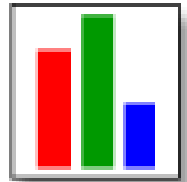
- **VuGen** (Virtual User Generator) – records Vuser Scripts that emulate the steps of real Users using the application
- The **Controller** is an administrative center for creating, maintaining, and executing scenarios. Starts and stops load tests, and perform other Administrative tasks



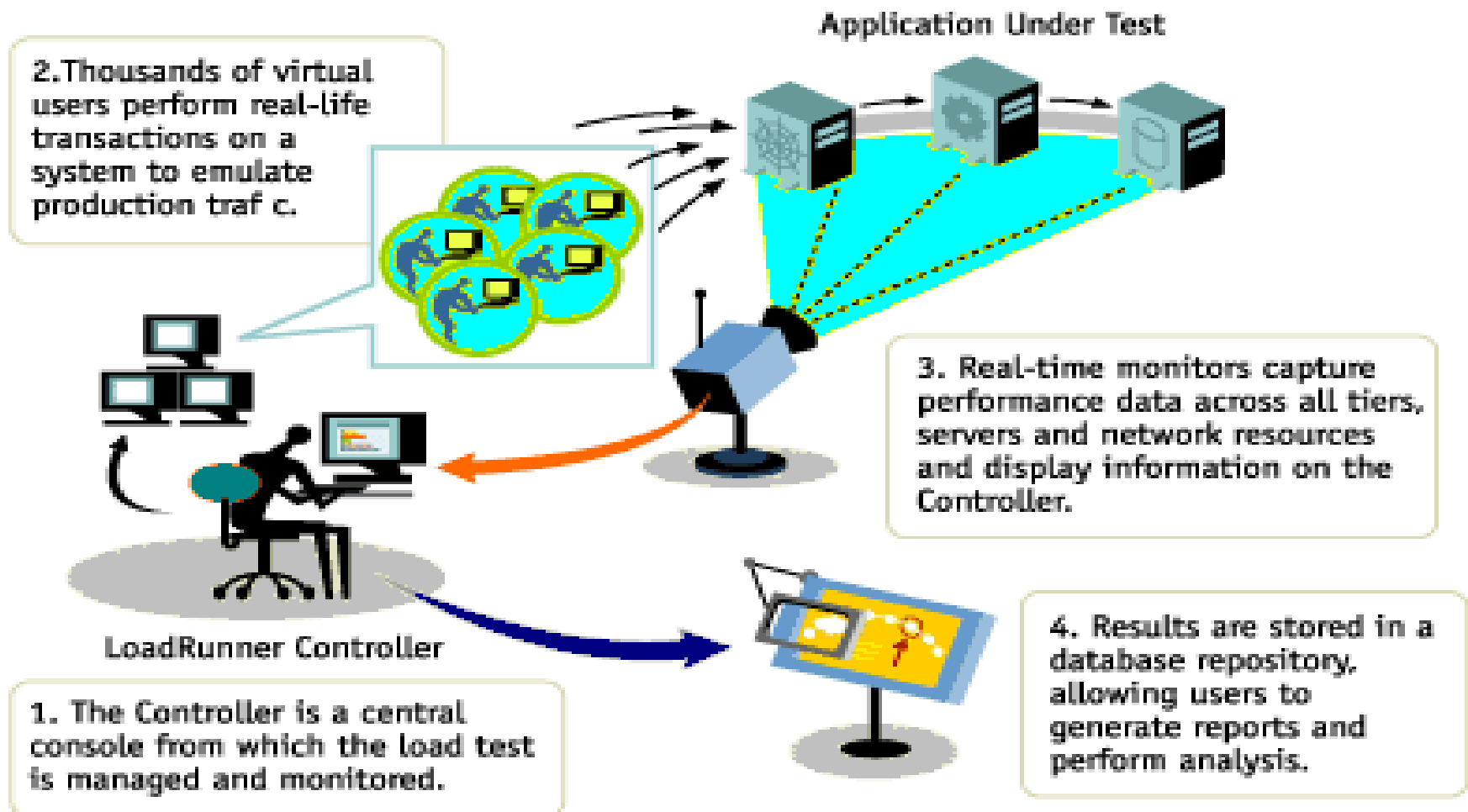
Contd....

# Components of LoadRunner 8.0

- **LR Analysis** uses the load test results to create graphs and reports that are used to correlate system information and identify both bottlenecks and performance issues.
- **Tuning** helps you quickly isolate and resolve performance bottlenecks. By adding a centralized tuning console to LoadRunner, the Mercury Tuning Module ensures that performance bottlenecks are resolved during testing, and helps you determine the optimized configuration settings for production.

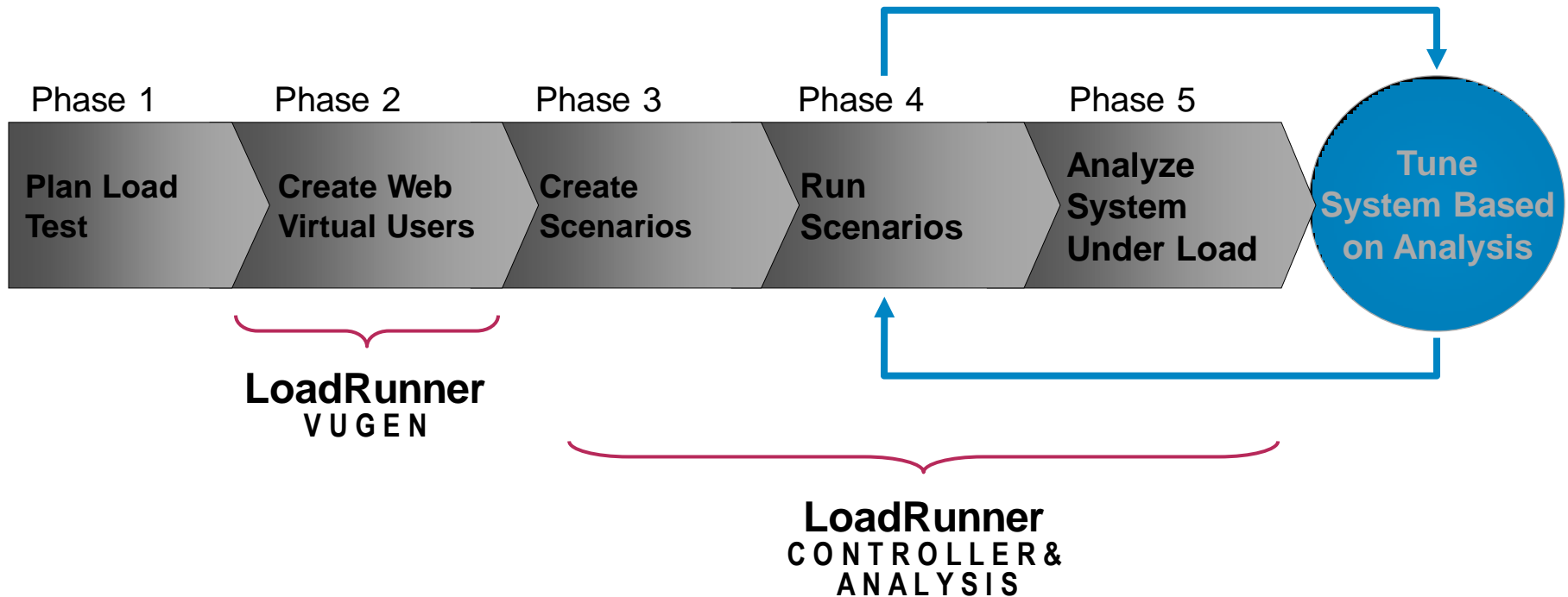


# How LoadRunner Works ?



# LoadRunner Expert Workflow

## "The Big Picture"



# Plan Load Test

**Identify Business Critical Scenarios.** Scenario means a manual work flow.  
Ex: Login → Open an Account → Logout.

## **Estimate User Load**

Performance Testing requirements will give an idea of users load or the number of users using the product. This will determine the load to be used against the product in testing.

## **Work Load:**

Ex: 100 user Running together. Of this 60 users book a Browse a website. 30 users search a product and 10 users buy the Product.



# What is Virtual User (Vuser) ?

- Virtual users or Vusers emulate the steps of real users. The steps that Vusers perform are recorded in a Vuser Script.

# What is VuGen (Virtual User Generator) ?

- VuGen records Vuser Scripts that emulate the steps of real users using the application
- VuGen not only records Vuser scripts, but also runs them. Running scripts from VuGen is useful for debugging
- VuGen records sessions on Windows platforms only. However, a recorded Vuser script can run on both Windows and UNIX platform.

# Process of Recording Script

- **Record** a basic script
- **Enhance** the basic script by adding the control-flow statements and other Mercury API functions into the Script
- **Configure** the Run-time settings
- **Verify** that the script runs correctly, run it in stand-alone mode
- **Integrate** into your test : a LoadRunner scenario, Performance Center load test, Tuning module session, Business process monitor profile



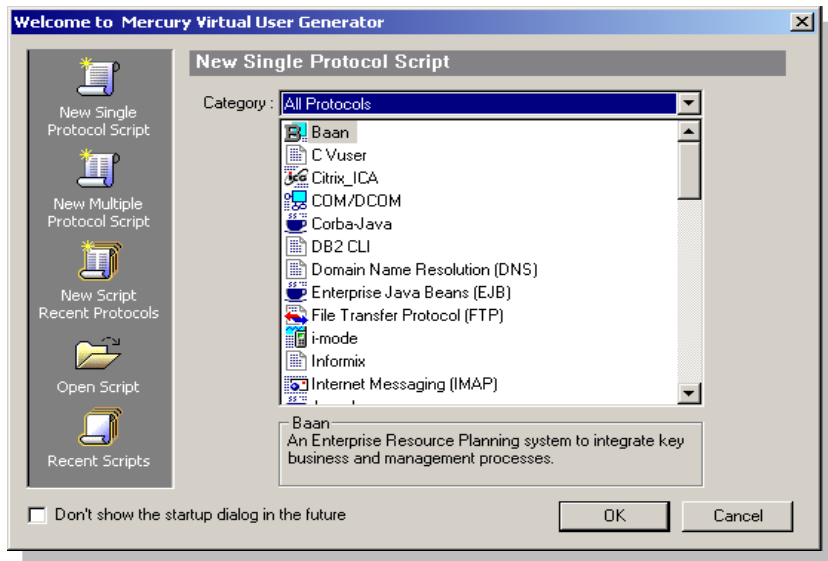
# VuGen

## What We can Do?

- Set up recording options
- Record the scripts
- Add Comments
- Insert Start and End Transactions
- Perform Correlation
- Add Checks
- Add C programming Statements wherever required.
- Insert Load Runner Functions if required.
- Do Parameterization.
- Add Rendezvous Point
- Create Multiple actions If required.
- Perform Run Time Settings

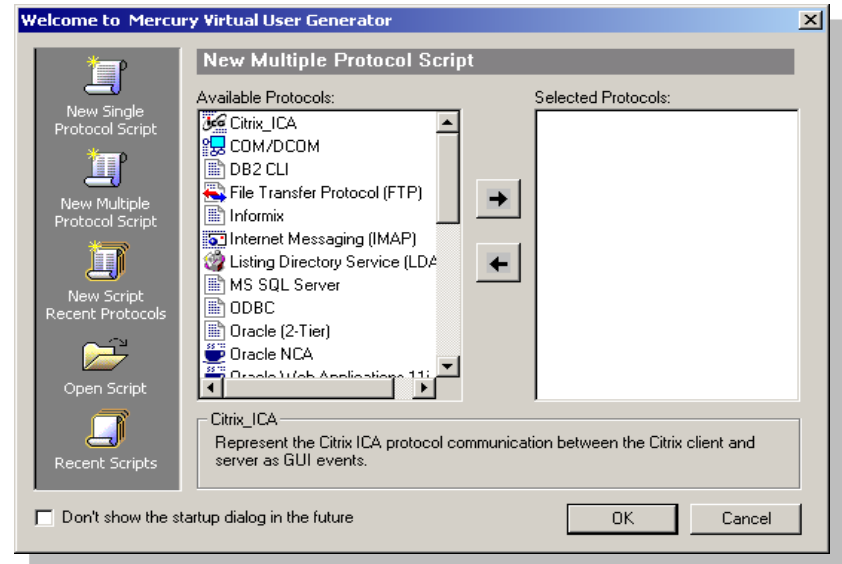
# Welcome Screen - VuGen

## *Single Protocol Script*



*Creates a single protocol Vuser script. This is the default option*

## *Multiple Protocol Script*



*Creates a multiple protocol Vuser script. VuGen displays all of the available protocols and allows you to specify which protocols to record*

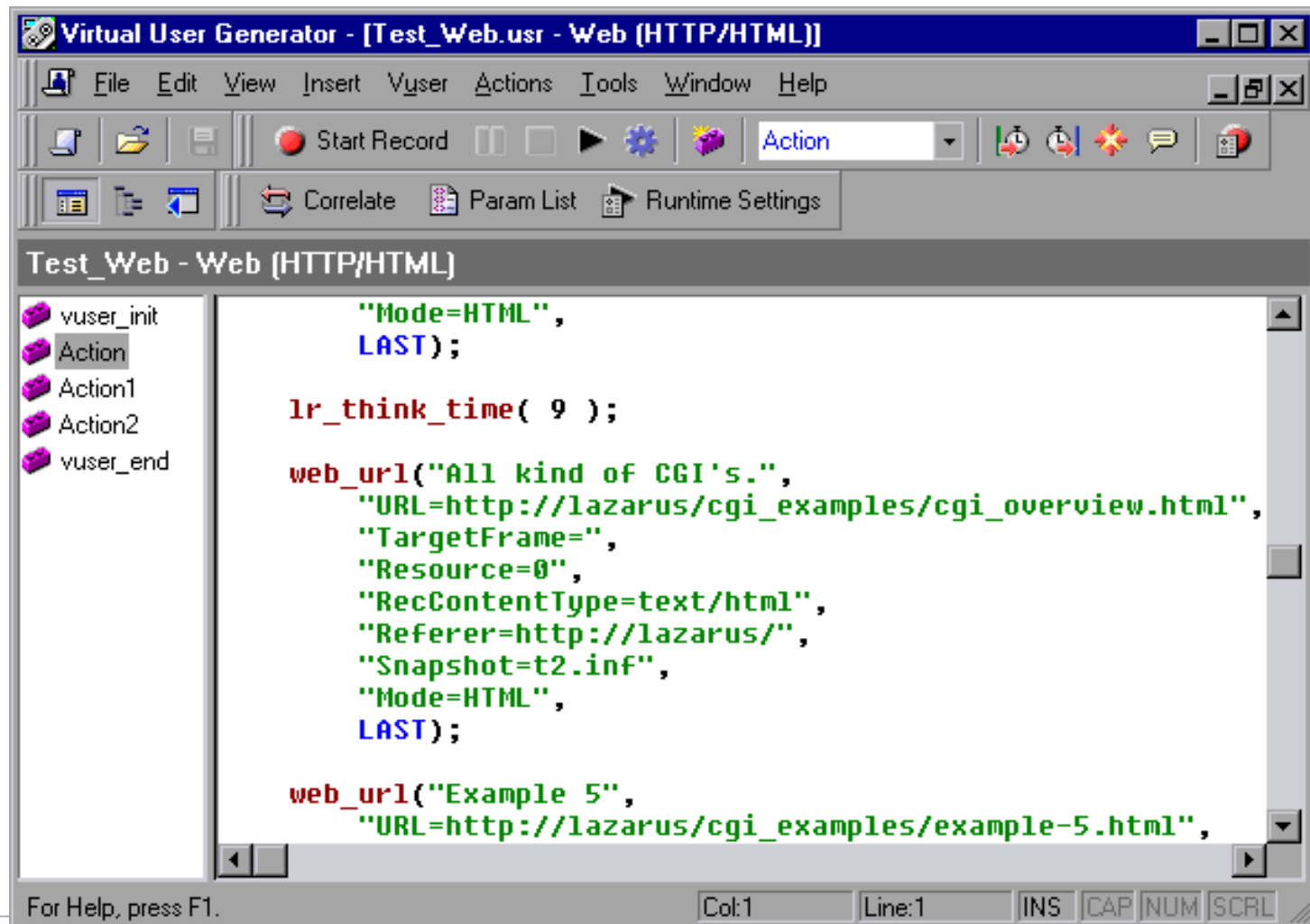
# Vuser Script Sections

- Each Vuser script contains at least three sections:
  - *vuser\_init*
  - one or more *Actions* and
  - *vuser\_end*.

Script Section	Used when recording...	Is executed when...
<i>vuser_init</i>	a login to a server	the Vuser is initialized (loaded)
<i>Actions</i>	client activity	the Vuser is in "Running" status
<i>vuser_end</i>	a logoff procedure	the Vuser finishes or is stopped

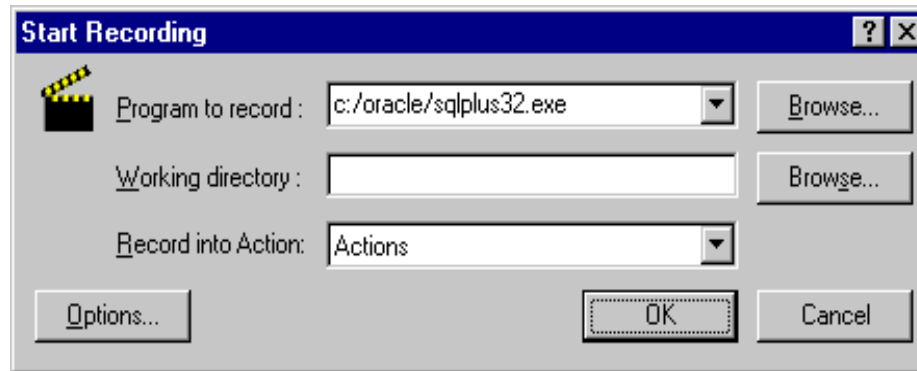
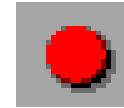


# VuGen Editor



# Recording Your Application

- Click the Start Recording Button
- For most Client / Server protocols, the following Screen opens



- Recording Tool Bar (Floating Tool Bar)





# Ending and Saving a Recording Session

To complete the recording:

- After you record a typical business process, you complete the recording session by performing the closing steps of your business process and saving the Vuser script.
- Switch to the *vuser\_end* section in the floating toolbar, and perform the log off or cleanup procedure.
- Click the stop      Recording button on the recording Tool Bar



# Enhancing Vuser Script

- After you record the Vuser Script you can enhance its capabilities by adding functions like
  - ***General Vuser Functions***
    - General Vuser functions greatly enhance the functionality of any Vuser Script. All general Vuser functions have an **LR** Prefix
  - ***Protocol - specific Vuser Functions***
    - Library functions used to enhance the script. (**LRS** - Windows, **LRT** - Tuxedo)
  - ***Standard ANSI C functions***
    - Enhancing the Vuser script by adding general C functions.
    - Like Adding Comments, Control flow statements, and so forth to your Vuser Script



# Enhancing Vuser Script

- **Inserting Transactions into Vuser Script**

- Inserting Rendezvous point
- Inserting Comments
- Obtaining Vuser Information

- **Sending Messages to output**

- *Log Messages*
  - Lr\_log\_message
- *Debug Messages*
  - Lr\_set\_debug\_message
  - Lr\_debug\_message
- *Error and Output Messages*
  - Lr\_error\_message
  - Lr\_output\_message



# Enhancing Vuser Script

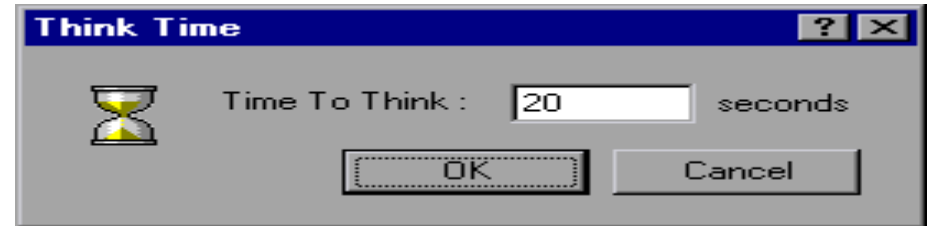
- **Handling errors on Vuser Script during execution** (*Runtime settings > Miscellaneous > Error handling*)
  - By default when a Vuser detects an error, the Vuser stops the execution
  - You can use the `lr_continue_on_error` function to override the `continue on error` runtime setting
  - To mark the segment, enclose it with `lr_continue_on_error(1);` and `lr_continue_on_error(0);` statements
- **Synchronizing Vuser Script**
  - Synchronize the execution of Vuser script with the output from your application
  - Synchronize applies only to RTE Vuser Scripts



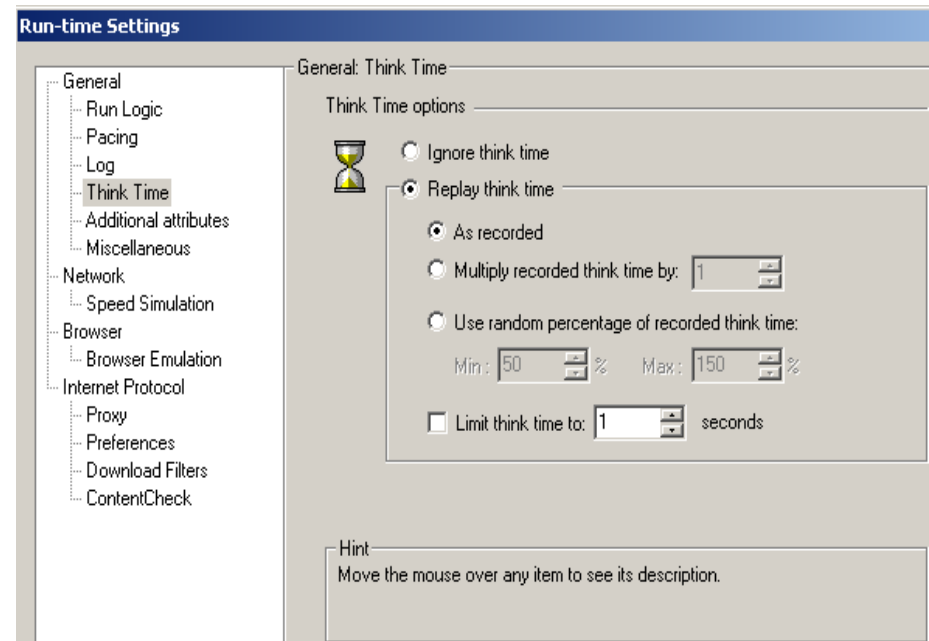
# Enhancing Vuser Script

- **Emulating User Think Time**

- The time that a user waits between performing successive action is known as the Think Time
- Vuser uses the `lr_think_time` function to emulate user think time



- **Vuser > Run-time settings > Think Time**



# Enhancing Vuser Script

## PARAMETERIZING

# Enhancing Vuser Script

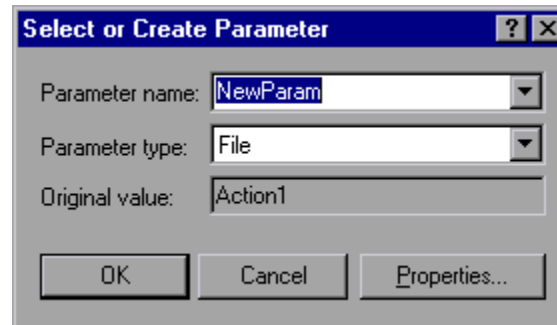
- **Parameterizing**
  - Parameterization involves the following two tasks:
    - Replacing the constant values in the Vuser script with parameters
    - Setting the properties and data source for the parameters
  - Parameterization Limitations
    - You can use parameterization only for the arguments within a function
    - You can't parameterize text strings that are not function arguments



# Enhancing Vuser Script

- **Creating Parameters**

- In a script View : **Select a string** and **select replace with parameter** from the Right click menu

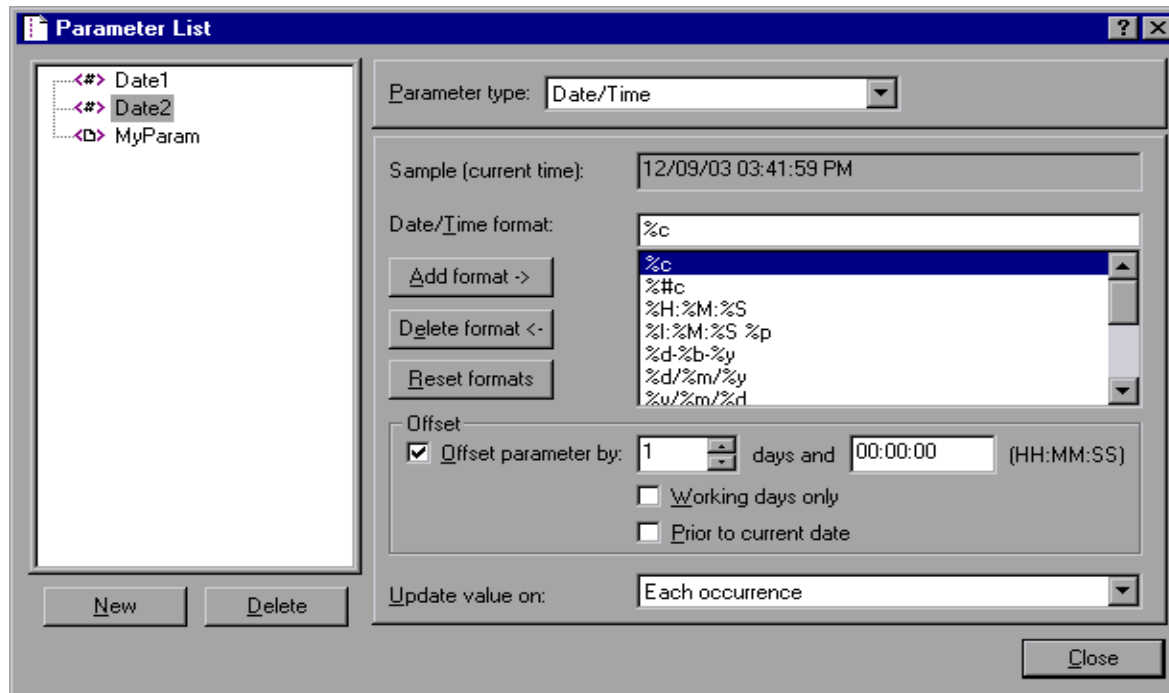


- Type the Name of the parameter in the appropriate box or select from the list
- Select parameter type from the parameter type list. The available types in the list are Date/Time, file, Group Name, Random number, Unique number, User defined function, or Vuser ID,



# Enhancing Vuser Script

- Vuser >Parameter List (or)  Param List



- VuGen creates new parameter, but does not automatically replace any selected string in the script

# Enhancing Vuser Script

**Submit Form Step Properties**

General | **Data** | Resource

	Name	Value	
1	username	{User_Name}	
2	password	{Password}	
3	login.x	52	REC
4	login.y	8	REC

Add...  
Delete...

OK Cancel Apply

File path: Names.dat Browse...

Add Column... Add Row... Delete Column... Delete Row...

	NewParam 1
1	Jim Taylor
2	Bob Smith
3	John Jones

Edit with Notepad... Data Wizard...

Select column:  
☒ By number: 1  
☐ By name:

File format:  
Column delimiter: Comma  
First data line: 1

Select next row: Unique  
Update value on: Each iteration  
When out of values: Continue with last value

Allocate Vuser values in the Controller:  
☐ Automatically allocate block size  
☒ Allocate 2 values for each Vuser



# Enhancing Vuser Script

- **Select Next Row**

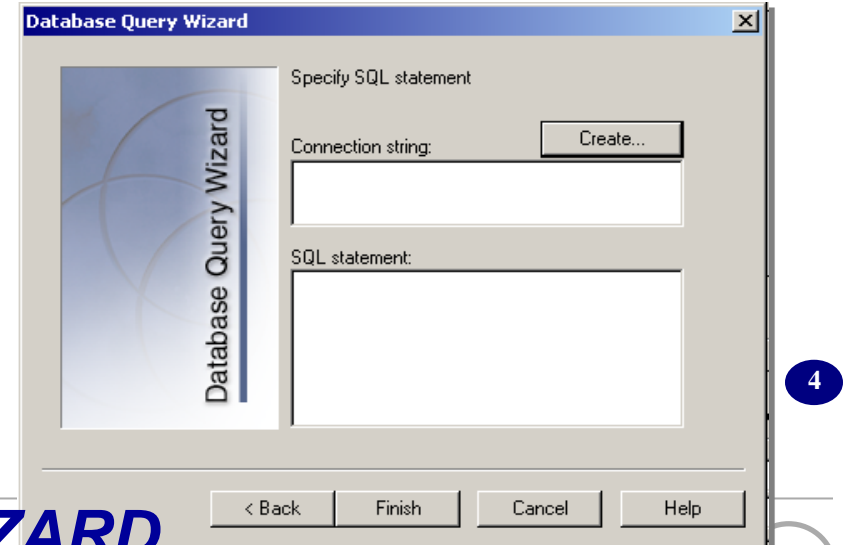
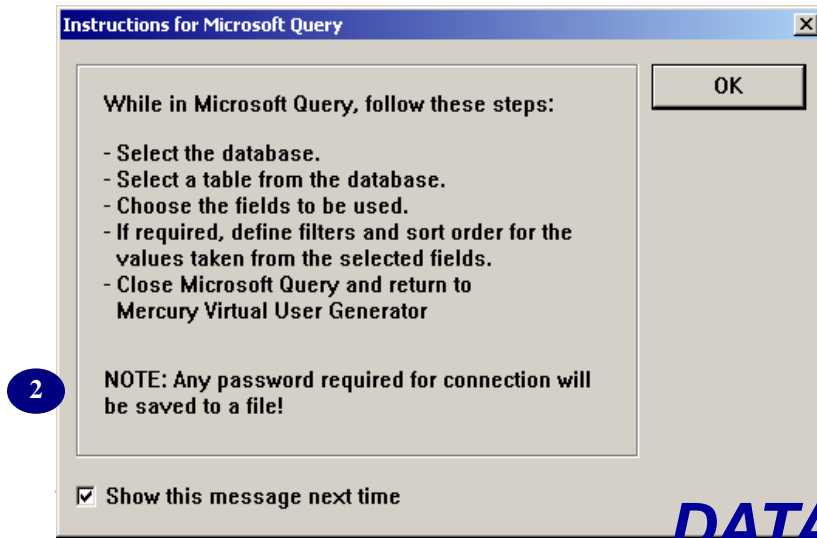
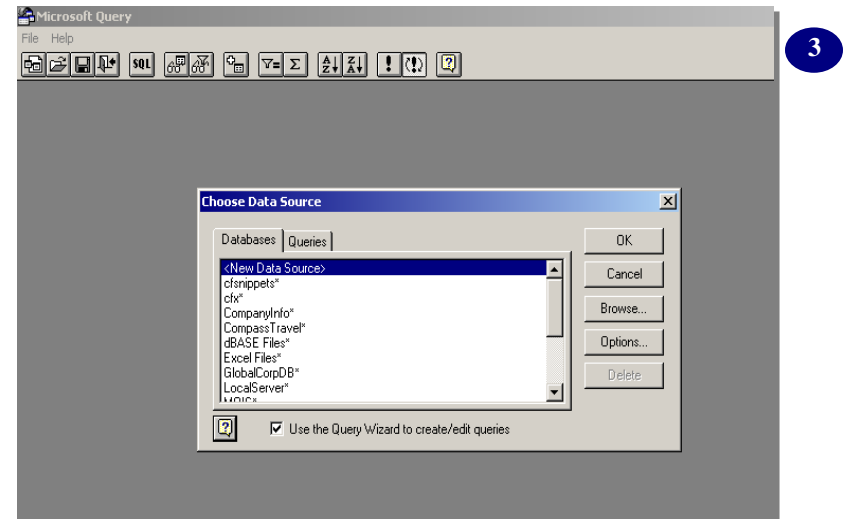
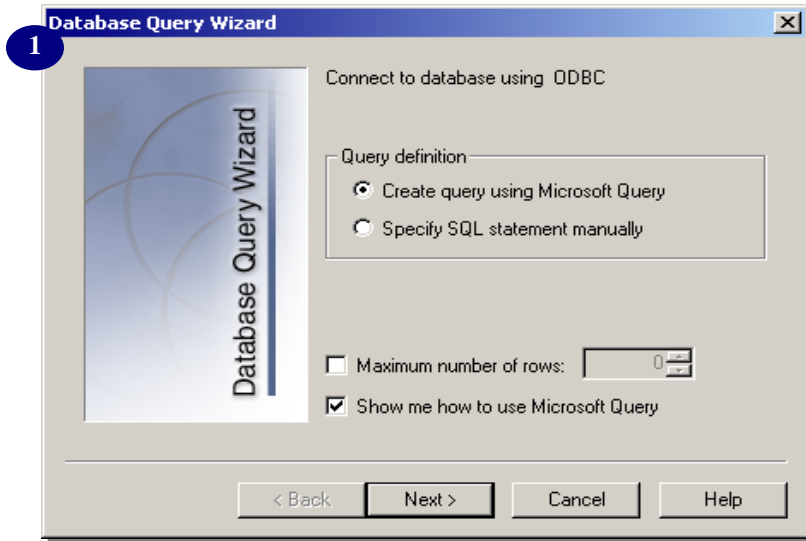
- Sequential
- Random
- Unique
- Same line as <Parameter\_Name>

- **Update Value on**

- Each iteration
  - Instructs the Vuser to use a new value for each script iteration
- Each occurrence
  - Instructs the Vuser to use a new value for each occurrence of the parameter
- Once
  - Instructs the Vuser to update the parameter value only once during the execution



# Enhancing Vuser Script



DATA WIZARD

# Enhancing Vuser Script

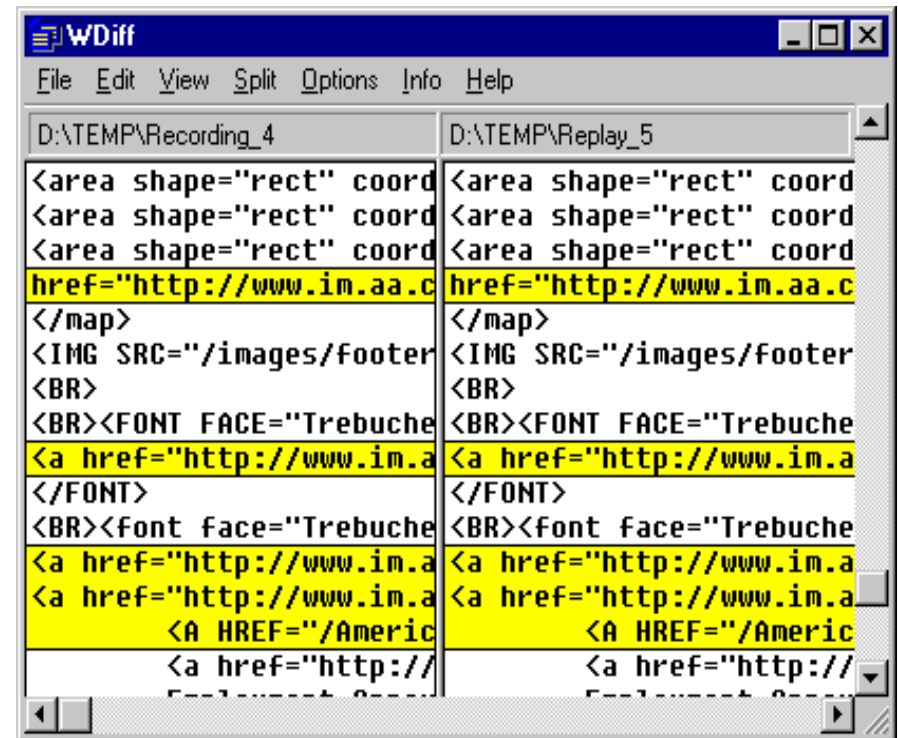
## CORRELATION



# Enhancing Vuser Script

## Primary reasons for correlating - To Generate dynamic code

- Determine which value to correlate
  - Using *WDiff* you can find which string to correlate
- Save the results using *Web\_reg\_save\_param* and *lrs\_save\_param*
- Replace the Saved variable in your query or in your statements



# Using Correlation in LoadRunner scripts - visual tutorial

- what is LoadRunner correlation and how to perform it. In my humble opinion, correlation is the key concept of LoadRunner. So, strong understanding of correlation is mandatory requirement for any test engineer, if he plans to be LoadRunner professional or even guru :)

## Example from a real practice:

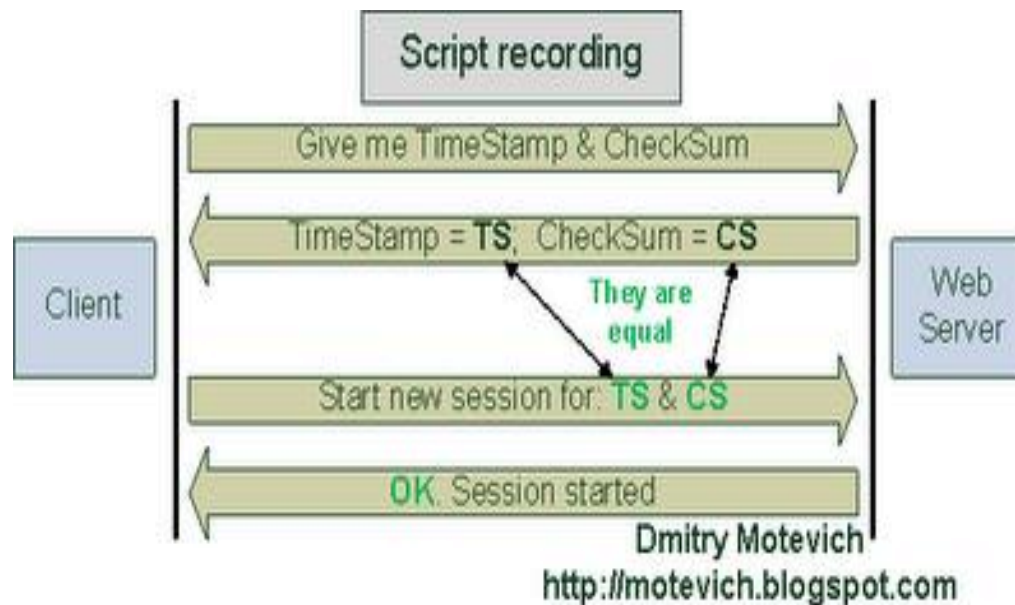
I recorded LoadRunner script for a web server, which contained two special fields - timestamp and checksum:

```
web_submit_data("rms.jsp",  
"Action=http://eprumossd0010:8400/RMS/jsp/rms.jsp",  
"Method=POST",  
"RecContentType=text/html",  
"Referer=http://eprumossd0010:8400/RMS/html/testFramework.html",  
"Snapshot=t4.inf",  
"Mode=HTML",  
ITEMDATA,  
"Name=TIMESTAMP", "Value=1192177661211", ENDITEM,  
"Name=CHECKSUM",  
"Value=715E19300D670ED77773BBF066DAAAE2866484B8", ENDITEM,  
// others parameters ...  
LAST);
```





Every time a client web browser connects to web server, server gets current time stamp, calculates checksum and sends them to client. These two fields are used to identify a current session. In other words, the pair of timestamp+checksum is analog of session ID. The scheme of this interaction is the following:



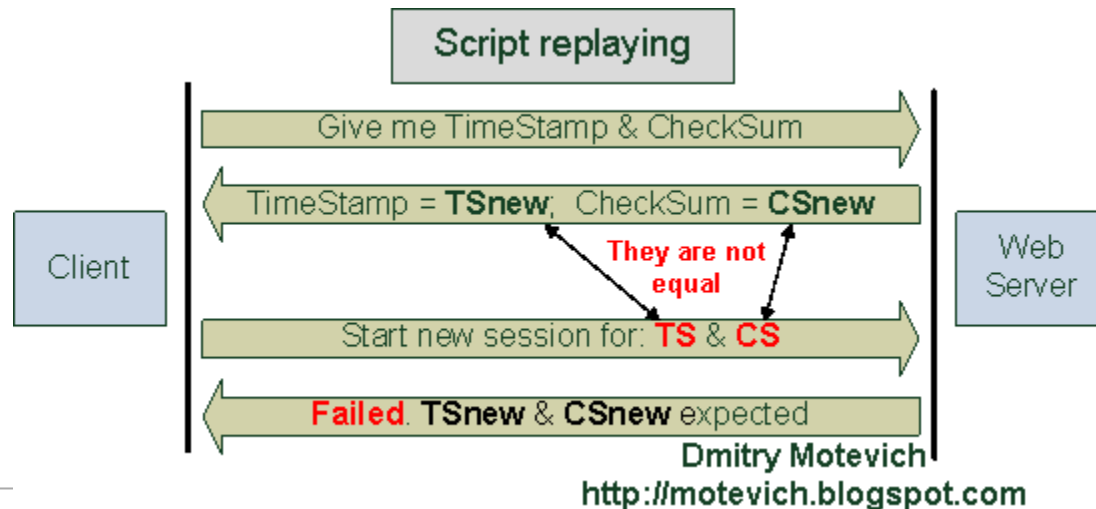
# Where is the problem? Let's replay the recorded LR script

The problem occurs when I try to execute my recorded script.

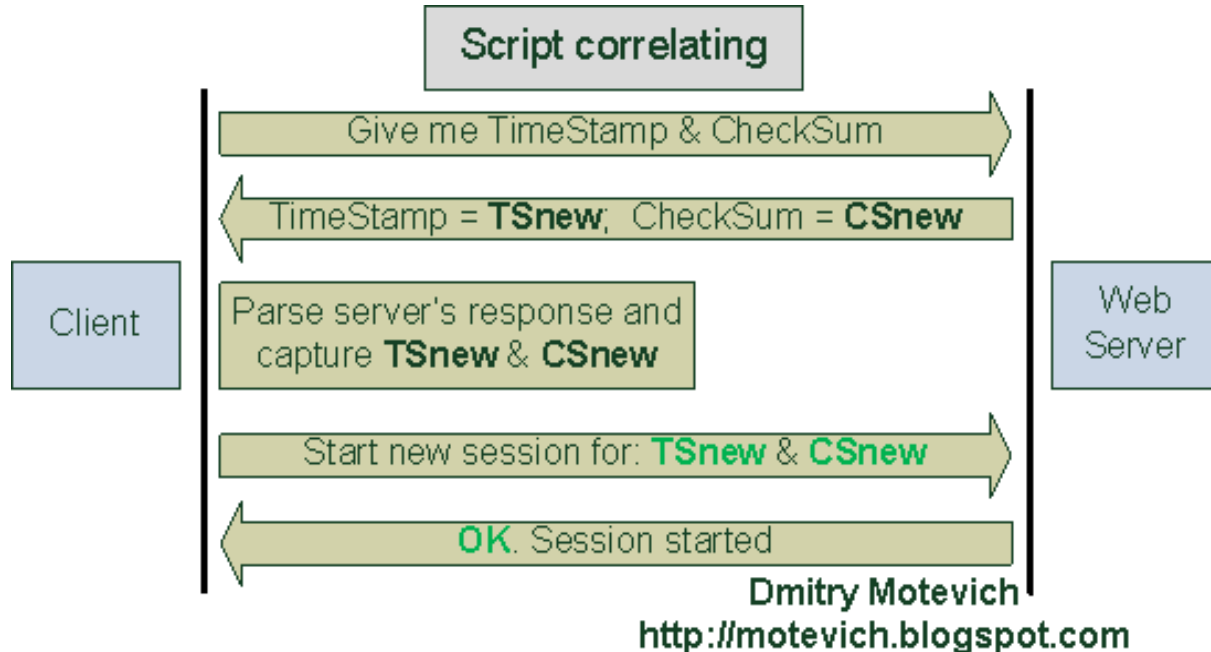
Web server checks its current time with a time stamp, sent by client. If client's data is out-of-date or incorrect, then server returns an error:

**The parameter "CHECKSUM" is not found or has invalid value.**

**There is the scheme for this interaction:**



Client cannot re-use old (i.e. hard-coded) values for times tamp and checksum. It must request new data. So, instead of hard-coded values, LR script should process dynamic data, returned from server. This can be done using a correlation:



# The definition of correlation is:

Correlation is the capturing of dynamic values passed from the server to the client. Correlation can be done with 2 ways.

## **1. Automatically**

## **2. Manually**

I will describe auto-correlation in the future posts. For now, I can say that this is not ideal solution. Sometimes, it does not work, or works incorrectly.

Manual correlation is a choice of real LoadRunner engineer. It's a kind of "must have" knowledge!

# Well, let's start investigating a manual correlation.

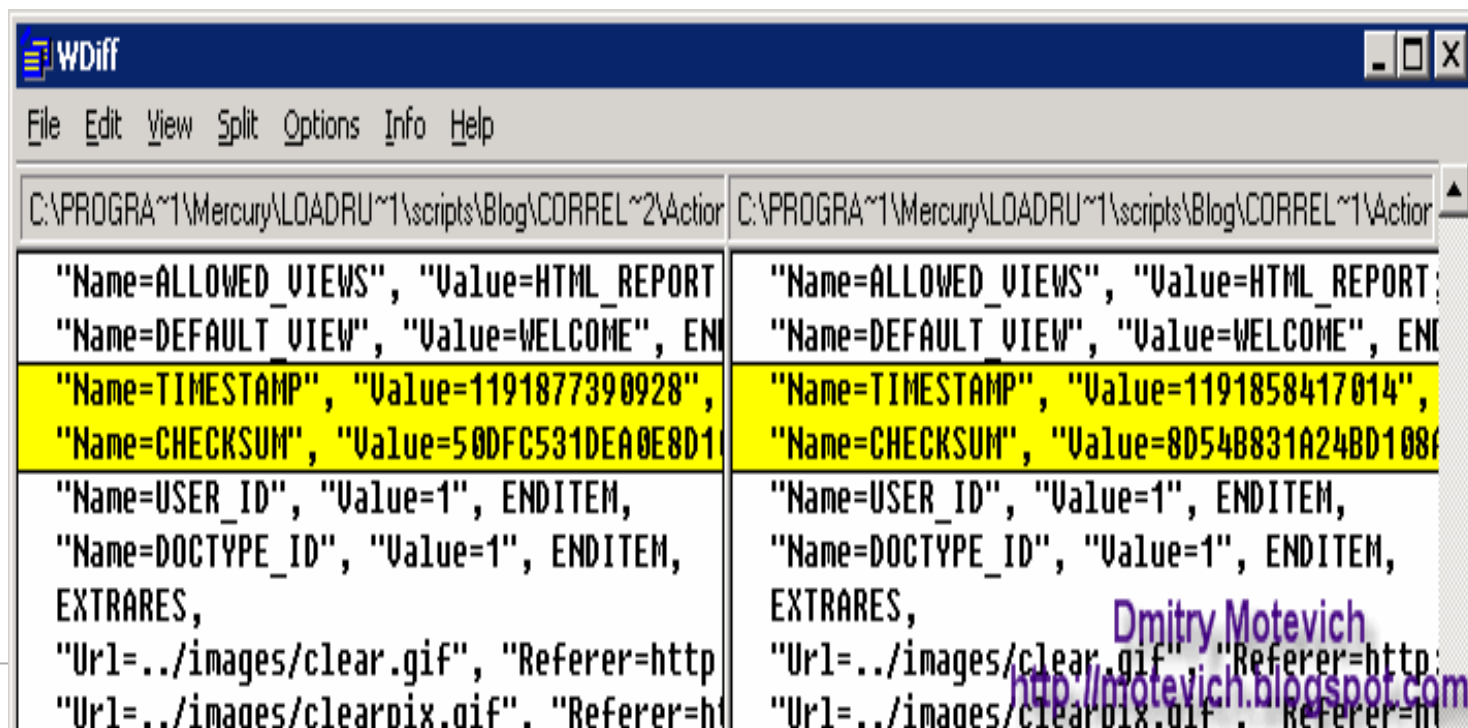
**The algorithm of manual correlation is the following:**

- ♦ Find a dynamic value to capture.
- ♦ Find server's response, containing the dynamic value.
- ♦ Capture the dynamic value.
- ♦ Special parameter will be used instead of dynamic value.
- ♦ Replace every occurrence of dynamic value in script with the parameter.
- ♦ Check changes.

# Now, I will describe each step in details:

## 1. Find a dynamic value to capture

I recommend to record and save two equal VuGen scripts. After that, open menu item "Tools / Compare with Scripts..." and you can compare both recorded scripts in WDiff:



- The differences are highlighted by yellow. This highlighting means that lines (parameters values) change from run to run. So, most probably, these values should be correlated.

**Tips:** Sometimes, comparing of two scripts cannot detect dynamic values. Imagine, that you recorded this script:

```
"Name=SessionID", "Value=A38E9002A41", ENDITEM,
```

```
"Name=CurrentMonthID", "Value=4", ENDITEM,
```

...

It's obvious, that SessionID should be correlated. What about CurrentMonthID parameter? Second recorded script can contain "Value=4" too. And it's possible, that your script will work correctly during the April (4th month is April), and will not work from 1st May!

- **Tips:** Look through the source code of recorded script. Timestamp, CheckSum, SessionID, and different IDs - all of them are potential candidates to be correlated.

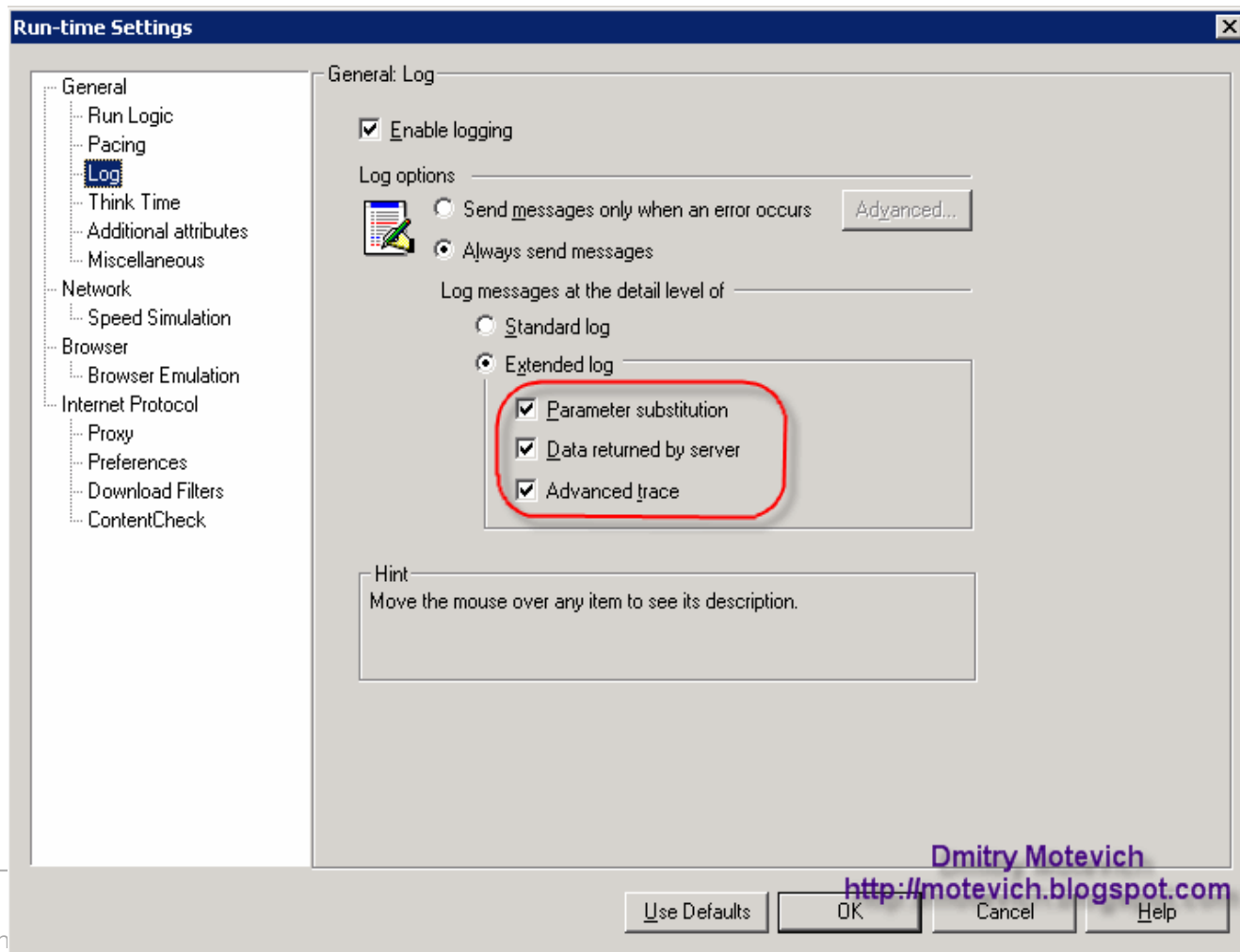
**Tips:** Check Replay (Execution) log carefully. Errors can be there. The widespread reason of script's errors is an absence of correlations.

**Tips:** Execute your script with enabled run-time viewer (menu "Tools / General Options.. / Display"). Any shown errors ("Page not found", "Session timeout", etc) indicate about potential correlations.





## 2. Find server's response, containing the dynamic value



Then execute script.

Open Replay (Execution) log and find server's response, which contains dynamic values of TIMESTAMP and CHECKSUM:

Replay Log   Recording Log   Correlation Results   Generation Log   Recorded Event Log

Action.c(13): t=1148ms: 216-byte response headers for "http://eprumossd0010:8400/RMS/isp/generateChecksum.jsp" (Rel  
Action.c(13): HTTP/1.1 200 OK\r\n  
Action.c(13): Server: Apache-Coyote/1.1\r\n  
Action.c(13): Set-Cookie: JSESSIONID=C14E5BE451B300\r\n  
Action.c(13): Content-Type: text/html; charset=ISO-8859-1\r\n  
Action.c(13): Content-Length: 174\r\n  
Action.c(13): Date: Fri, 12 Oct 2007 08:30:26 GMT\r\n  
Action.c(13): \r\n  
Action.c(13): t=1163ms: 174-byte response body for "http://eprumossd0010:8400/RMS/isp/generateChecksum.jsp" (Rel  
Action.c(13): \r\n  
Action.c(13): \r\n  
Action.c(13): \r\n  
Action.c(13): \r\n  
Action.c(13): \r\n  
Action.c(13): \r\n  
Action.c(13): <html>\r\n  
Action.c(13): <body>\r\n  
Action.c(13): <script language="javascript">\r\n  
Action.c(13): window.parent.setChecksum("1EF8DBF5310C99276CB012C2A7FE7A8C6D4DCE5C", 1192177826528);\r\n  
Action.c(13): </script>\r\n  
Action.c(13): </body>\r\n  
Action.c(13): </html>

Find

Find What: Checksum

Match Whole Word Only  
Match Case  
Regular Expression

Direction  
Up  
Down

Checksum's value   1192177826528  
TIMESTAMP's value

Dmitry Moteyich  
http://motevich.blogspot.com

- Great! Now we know, where server sends both dynamic values. And we found the step, that returns these values. This is 13th line - Action.c (13). Double click the line, containing timestamp's and checksum's values, 13th line of script will be opened:  
`web_submit_data("generateChecksum.jsp",  
"Action=http://eprumossd0010:8400/RMS/jsp/generateChecksum.jsp",  
"Method=POST",  
"RecContentType=text/html",  
...  
This means that server's response for generateChecksum.jsp page contains dynamic values which should be correlated.  
you to fine-tune your system during test execution.`

### 3. Capture the dynamic value

I will show two ways how to capture a dynamic value:

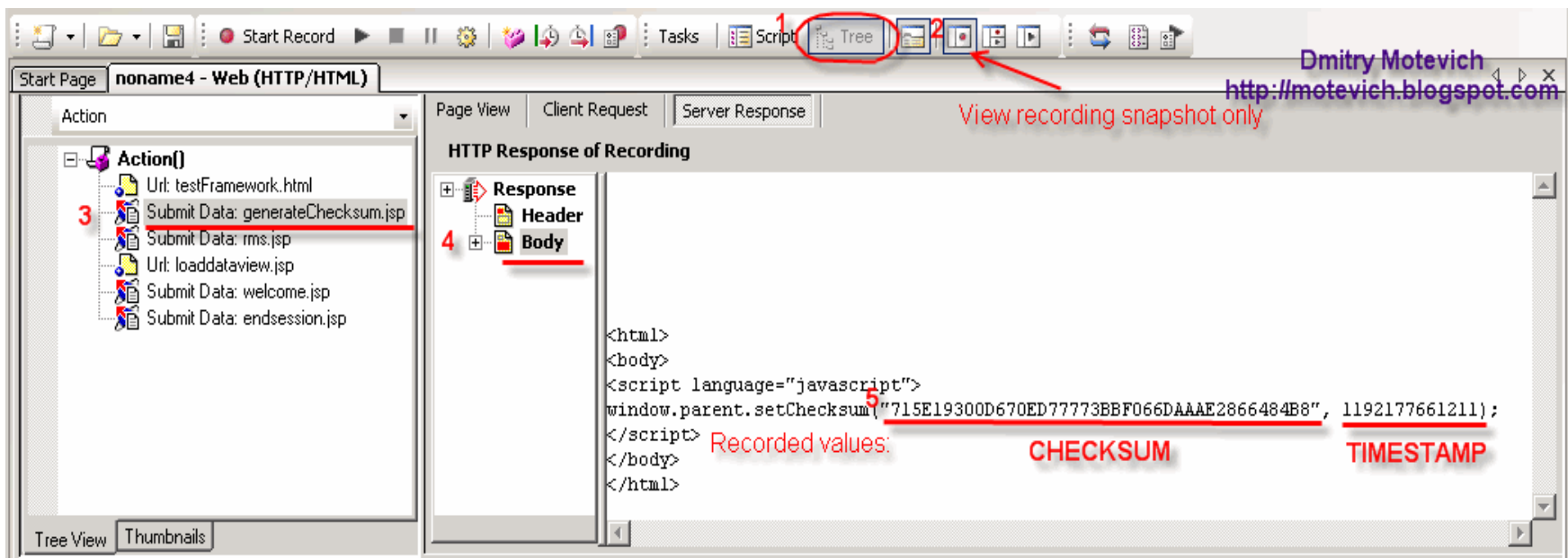
- Automatic capturing from Tree-view
- Manual from Script-view

These ways are similar enough.

Also, they use the same function – `web_reg_save_param`.

I start from:

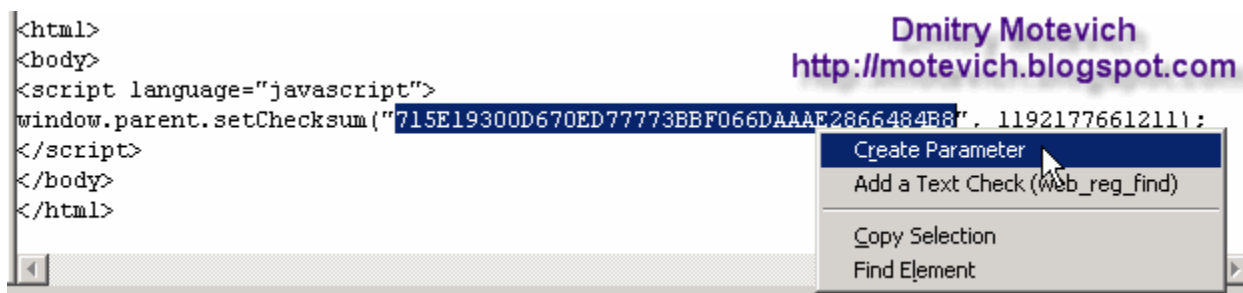
- Automatic capturing from Tree-view
- Open Tree view (menu "View / Tree view"):



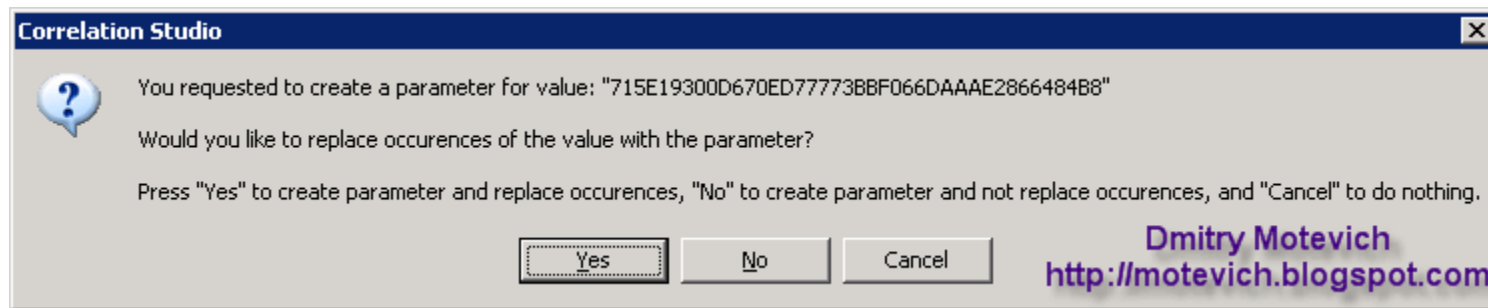
Then:

- click "View recording snapshot only" btn (2);
- select generateChecksum.jsp page from tree view (3);
- select "Body" to view body of server's response (4).

As result, you will see recorded values of CHECKSUM and TIMESTAMP (5).  
Now, select value of first dynamic value (checksum), right-click, and select "Create parameter":



After that you will see the message box:



You can create parameter for dynamic value.

If you want to replace all occurrences of dynamic value ("715E19...") in script, press "Yes" btn.

To not replace all occurrences, press "No" btn.

Tips: I recommend to not replace all occurrences of dynamic value. It can lead to incorrect results. It's more preferable to replace occurrences one by one with "Search and Replace" dlg.

OK, I click "No" btn

Return to Script-view ("View / Script View") and see changes. There are new lines, inserted before generateChecksum.jsp page:

```
// [WCSPARAM WCSPParam_Text1 40  
715E19300D670ED77773BBF066DAAAE2866484B8] Parameter  
{WCSPParam_Text1} created by Correlation Studio  
web_reg_save_param("WCSPParam_Text1",  
"LB=window.parent.setChecksum(\"",  
"RB=\",",  
"Ord=1",  
"RelFrameId=1",  
"Search=Body",  
"IgnoreRedirections=Yes",  
LAST);
```



web\_reg\_save\_param function finds and saves a text string from the next server's response. In other words, it captures a dynamic value.

In this example, web\_reg\_save\_param function will save the captured value into WCSParam\_Text1 parameter. The function finds the left boundary (window.parent.setChecksum("")) and after that it finds the right boundary (").

The string, found between left and right boundaries, will be saved to WCSParam\_Text1 parameter.

Ord attribute indicates the ordinal position of captured value.

In the example (Ord=1), we capture the value between first left boundary and left one.

Easy to guess, that "Search=Body" means search in a body of server's response. I recommend to study Help on web\_reg\_save\_param function.

**Note:** the capturing of TIMESTAMP parameter is similar. It generates the following code:

```
// [WCSPARAM WCSParam_Text2 13 1192177661211] Parameter  
{WCSParam_Text2} created by Correlation Studio  
web_reg_save_param("WCSParam_Text2",  
"LB=", "  
"RB=)",  
"Ord=1",  
"RelFrameId=1",  
"Search=Body",  
"IgnoreRedirections=Yes",  
LAST);
```

### ***Manual capturing from Script-view.***

Actually, this method consists in a manual writing of web\_reg\_save\_param function. It requires strong knowledge on this function and its parameters. There are many attributes of web\_reg\_save\_param function, and I do not want to duplicate HP's Help :)

***Tips:*** I recommend to rename default parameter (WCSParam\_Text1, 2, 3, etc) to something sensible. For example, in my example - I would prefer prmChecksum and prmTimeStamp to WCSParam\_Text1 and WCSParam\_Text2.



## 4. Replace every occurrence of dynamic value in script with the parameter

This is not difficult step.

Open "Search and Replace" dlg ("Edit / Replace..."). And replace one-by-one hard-coded values with a parameter.

Why it is important?

Imagine, that you have the following code:

```
web_submit_data("somepage",  
...  
"Name=OrderNumber", "Value=125", ENDITEM,  
"Name=UserID", "Value=125",
```

- If you create parameter for UserID, and perform replacing of all occurrences of its value ("125"), then it will produce the code:

```
web_submit_data("somepage",  
...  
"Name=OrderNumber", "Value={WCSPParam_Text1}", ENDITEM,  
"Name=UserID", "Value={WCSPParam_Text1}",
```

It may be wrong! OrderNumber can be static value and be equal to 125, while UserID may change.

Now, I assume that you replace all needed occurrences of hard-coded values. We have to perform last step:



## 5. Check changes

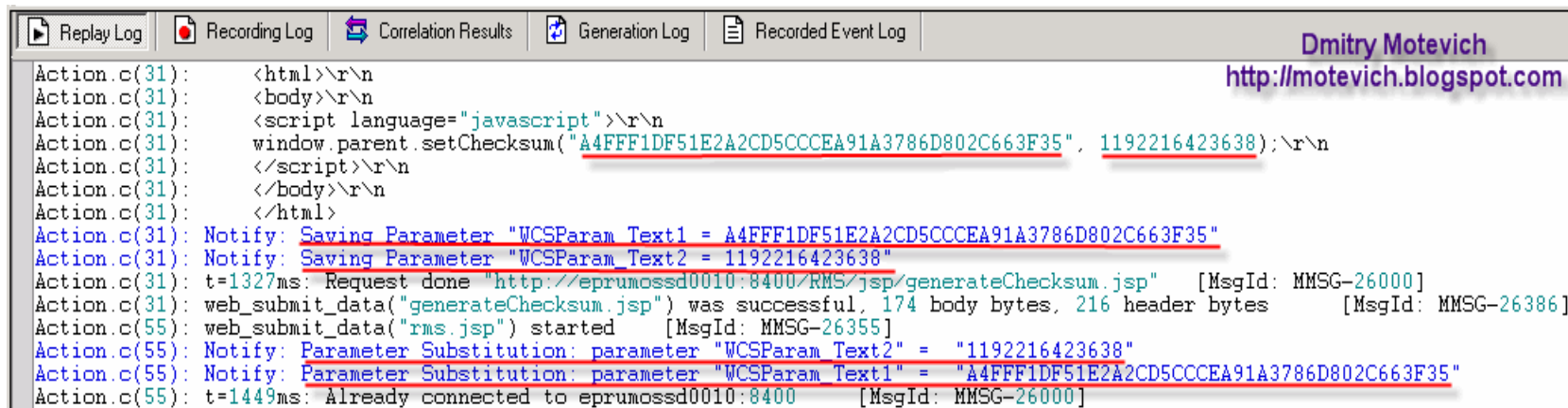
After above manipulations, our script will look like:

```
web_submit_data("rms.jsp",  
"Action=http://eprumossd0010:8400/RMS/jsp/rms.jsp",  
"Method=POST",  
"RecContentType=text/html",  
"Referer=http://eprumossd0010:8400/RMS/html/testFramework.html",  
"Snapshot=t4.inf",  
"Mode=HTML",  
ITEMDATA,  
"Name=TIMESTAMP", "Value={WCSPParam_Text2}", ENDITEM,  
"Name=CHECKSUM", "Value={WCSPParam_Text1}", ENDITEM,  
// others parameters ...  
LAST);
```

The statement "{WCSPParam\_Text1}" means "get value of WCSPParam\_Text1 parameter". So, current algorithm is:

- when server returns different values of CheckSum and TimeStamp
- then web\_submit\_data captures and places them into WCSPParam\_Text1 and WCSPParam\_Text2 parameters
- after that we use {WCSPParam\_Text1} and {WCSPParam\_Text2} get current values of parameters and use them in scripts

Let's run our modified script and see results of capturing from server's response:

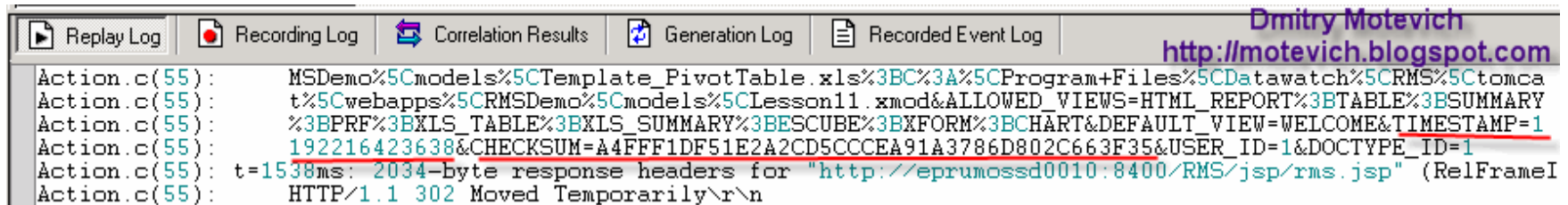


The screenshot shows a web debugging tool interface with a log of actions and notifications. The log is displayed in a window with tabs for "Replay Log", "Recording Log", "Correlation Results", "Generation Log", and "Recorded Event Log". The "Recording Log" tab is active, showing a list of actions and notifications. The log entries are as follows:

```
Action.c(31): <html>\r\n
Action.c(31): <body>\r\n
Action.c(31): <script language="javascript">\r\n
Action.c(31): window.parent.setChecksum("A4FFF1DF51E2A2CD5CCCEA91A3786D802C663F35", 1192216423638);\r\n
Action.c(31): </script>\r\n
Action.c(31): </body>\r\n
Action.c(31): </html>
Action.c(31): Notify: Saving Parameter "WCSPParam_Text1 = A4FFF1DF51E2A2CD5CCCEA91A3786D802C663F35"
Action.c(31): Notify: Saving Parameter "WCSPParam_Text2 = 1192216423638"
Action.c(31): t=1327ms: Request done "http://eprumossd0010:8400/RMS/jsp/generateChecksum.jsp" [MsgId: MMSG-26000]
Action.c(31): web_submit_data("generateChecksum.jsp") was successful, 174 body bytes, 216 header bytes [MsgId: MMSG-26386]
Action.c(55): web_submit_data("rms.jsp") started [MsgId: MMSG-26355]
Action.c(55): Notify: Parameter Substitution: parameter "WCSPParam_Text2" = "1192216423638"
Action.c(55): Notify: Parameter Substitution: parameter "WCSPParam_Text1" = "A4FFF1DF51E2A2CD5CCCEA91A3786D802C663F35"
Action.c(55): t=1449ms: Already connected to eprumossd0010:8400 [MsgId: MMSG-26000]
```

The log also includes a watermark for Dmitry Motevich at <http://motevich.blogspot.com>.

These are values, which are sent to a server by a client:



```
Replay Log | Recording Log | Correlation Results | Generation Log | Recorded Event Log | Dmitry Motevich  
http://motevich.blogspot.com  
Action.c(55): MSDemo%5Cmodels%5CTemplate_PivotTable.xls%3BC%3A%5CProgram+Files%5CDatawatch%5CRMS%5Ctomca  
Action.c(55): t%5Cwebapps%5CRMSDemo%5Cmodels%5CLesson11.xmod&ALLOWED_VIEWS=HTML_REPORT%3BTABLE%3BSUMMARY  
Action.c(55): %3BPRF%3BXLS_TABLE%3BXLS_SUMMARY%3BESCUBE%3BXFORM%3BCHART&DEFAULT_VIEW=WELCOME&TIMESTAMP=1  
Action.c(55): 192216423638&CHECKSUM=A4FFF1DF51E2A2CD5CCCEA91A3786D802C663F35&USER_ID=1&DOCTYPE_ID=1  
Action.c(55): t=1538ms: 2034-byte response headers for "http://eprumossd0010:8400/RMS/jsp/rms.jsp" (RelFrameI  
Action.c(55): HTTP/1.1 302 Moved Temporarily\r\n
```

You can see that dynamic values are saved to parameters and their values are substituted instead of parameters in scripts. Great! We have just correlated our script and it is working now!

Tips: To get/debug a captured value of parameter, use the following statements:

```
lr_output_message("Value of WCSParam_Text1: %s",  
lr_eval_string("{WCSParam_Text1}"));  
lr_output_message("Value of WCSParam_Text2: %s",  
lr_eval_string("{WCSParam_Text2}"));
```





Execute script again, the result is:

```
Replay Log | Recording Log | Correlation Results | Generation Log | Recorded Event Log
Action.c(31): window.parent.setChecksum("8C5320DCDA1E4E3DFBEFCC1A6A171C91172A32DF", 1192219152224);\r\n
Action.c(31): </script>\r\n
Action.c(31): </body>\r\n
Action.c(31): </html>
Action.c(31): Notify: Saving Parameter "WCSPParam_Text1 = 8C5320DCDA1E4E3DFBEFCC1A6A171C91172A32DF"
Action.c(31): Notify: Saving Parameter "WCSPParam_Text2 = 1192219152224"
Action.c(31): t=1144ms: Request done "http://eprumossd0010:8400/RMS/jsp/generateChecksum.jsp" [MsgId: MMSG-26000]
Action.c(31): web_submit_data("generateChecksum.jsp") was successful, 174 body bytes, 216 header bytes [MsgId: MMS
Action.c(55): Notify: Parameter Substitution: parameter "WCSPParam_Text1" = "8C5320DCDA1E4E3DFBEFCC1A6A171C91172A32DF"
Action.c(55): Value of WCSPParam_Text1: 8C5320DCDA1E4E3DFBEFCC1A6A171C91172A32DF
Action.c(56): Notify: Parameter Substitution: parameter "WCSPParam_Text2" = "1192219152224"
Action.c(56): Value of WCSPParam_Text2: 1192219152224
Action.c(58): web_submit_data("rms.jsp") started [MsgId: MMSG-26355]
Action.c(58): Notify: Parameter Substitution: parameter "WCSPParam_Text2" = "1192219152224"
Action.c(58): Notify: Parameter Substitution: parameter "WCSPParam_Text1" = "8C5320DCDA1E4E3DFBEFCC1A6A171C91172A32DF"
Action.c(58): t=1249ms: Already connected to eprumossd0010:8400 [MsgId: MMSG-26000]
```

Dmitry Motevich

<http://motevich.blogspot.com>



# Enhancing Vuser Script

```
web_reg_save_param("myval", "LB=userSession value=",  
"RB=>", "Ord=1", "RelFrameId=1.2.1", "Search=Body", LAST);
```

```
"Name=userSession", "Value={myval}"
```



**Right  
boundary  
Value**

**Storage  
Variable**

**Left  
boundary  
Value**

## Run-time Settings

# RunTime Settings

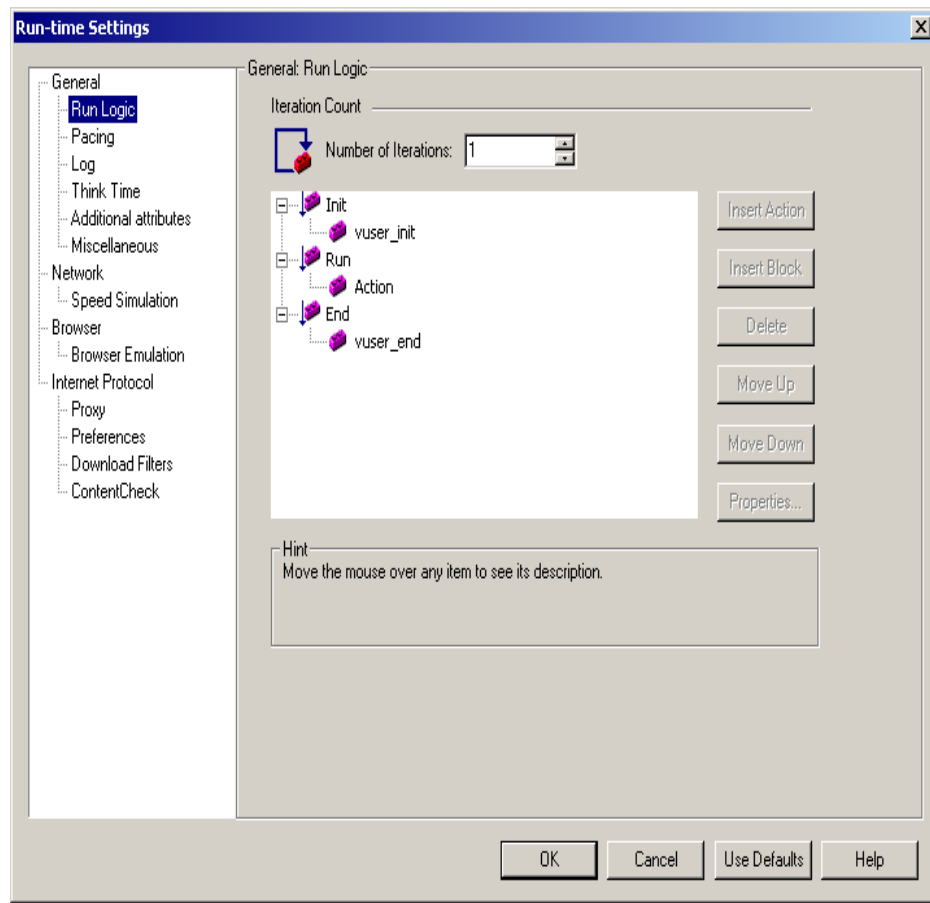
## Run Logic

You can instruct a Vuser to Repeat the *Run* section when you run the script. Each repetition is known as *iteration*

### ***Number of Iterations***

LoadRunner repeats all of the actions, the specified number of times.

If you specify a scenario duration in the controller, the duration setting overrides the Vusers iteration settings.



# RunTime Settings

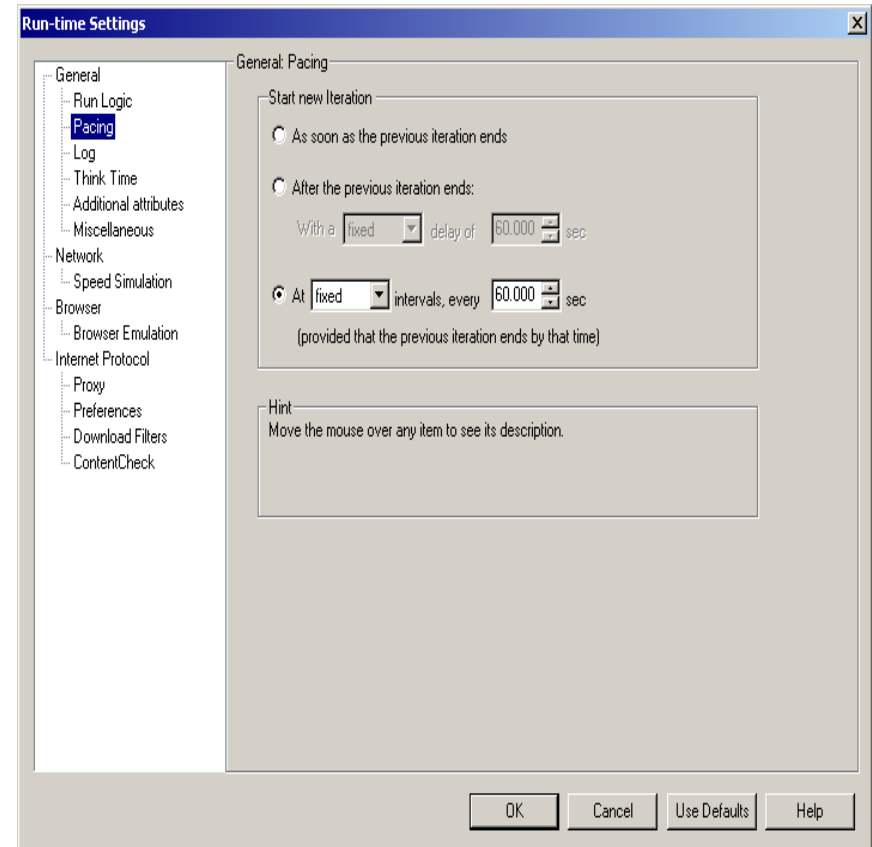
## Pacing

Control the time between iterations.

The pace tells the Vuser how long to wait between iterations of Vuser

You can instruct Vuser by following any of the method below

1. As soon as the previous iteration ends.
2. After the previous iteration ends with a fixed / random delay
3. At fixed / random intervals



# RunTime Settings

## Log

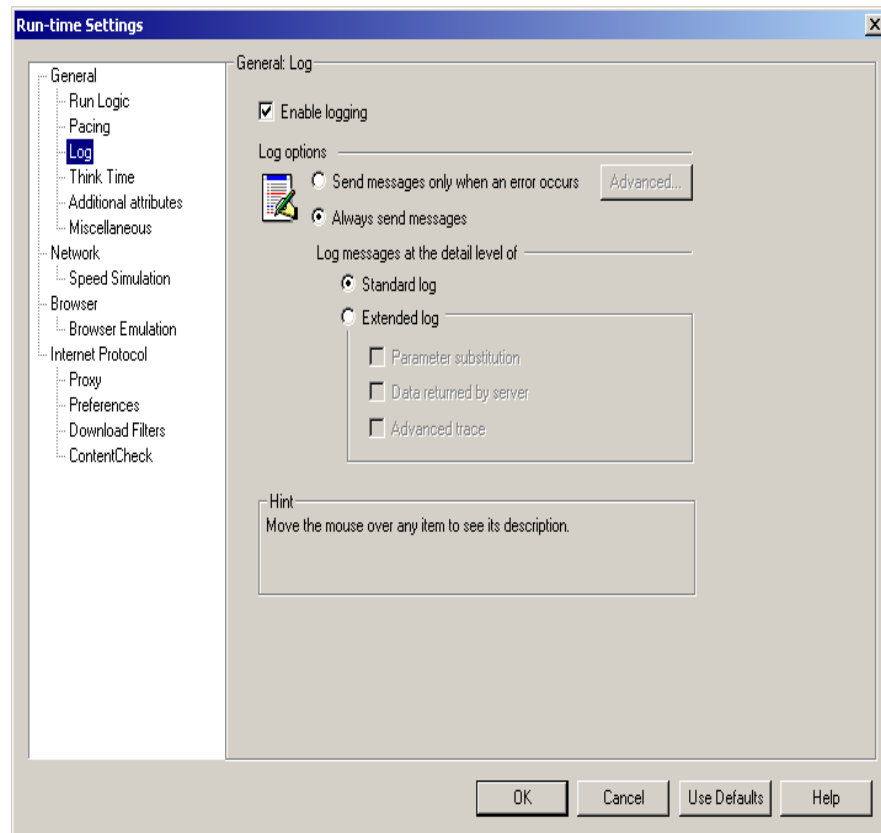
Vusers log information about themselves and their communication between server

Two types of Logs

- **Standard**
- **Extended**

VuGen writes log messages that you can view in execution log.

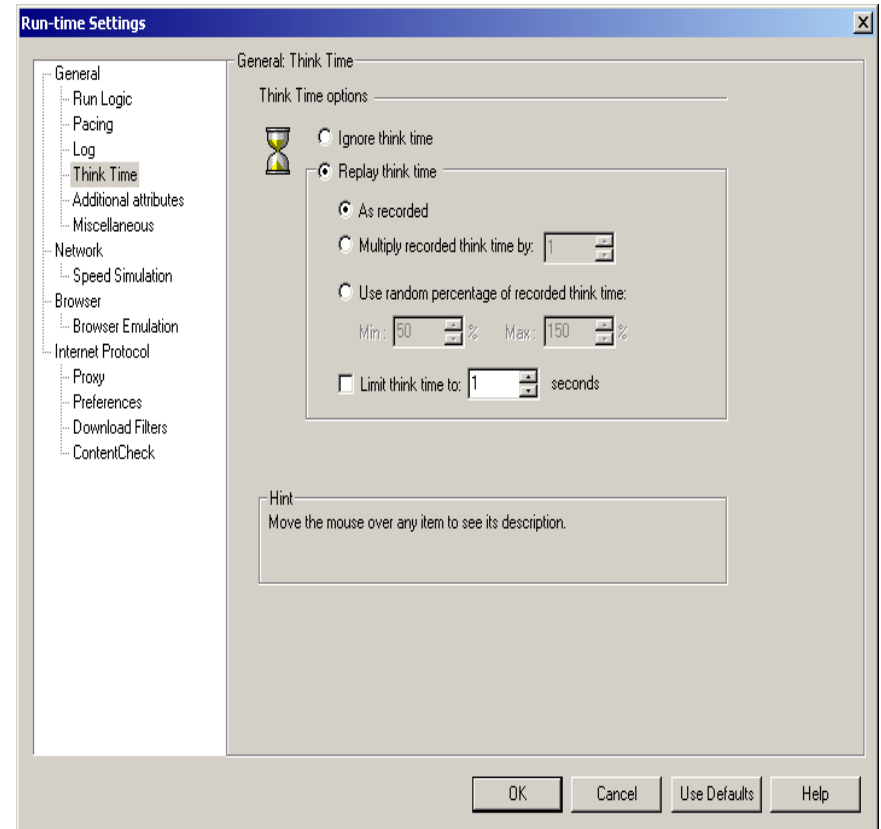
**lr\_log\_message**. Messages sent manually, using **lr\_message**, **lr\_output\_message**, and **lr\_error\_message**, are still issued



# RunTime Settings

## Think Time

When you run a Vuser script, the Vuser uses the think time values that were recorded into the script during the recording session. VuGen allows you to use the recorded think time, ignore it, or use a value related to the recorded time:

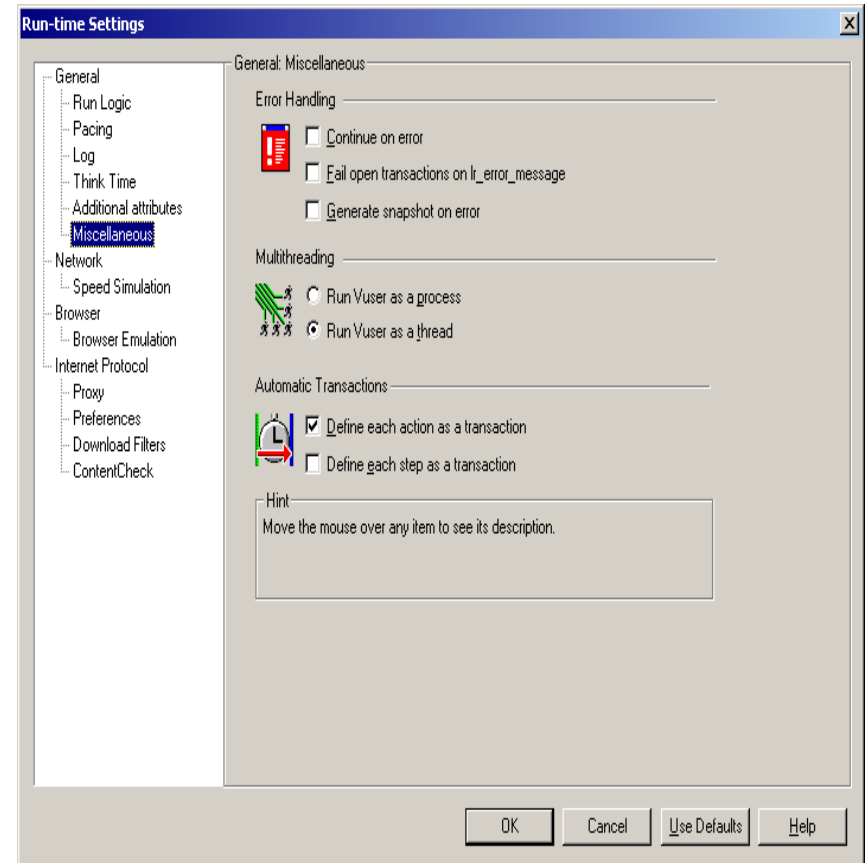


# RunTime Settings

## Miscellaneous

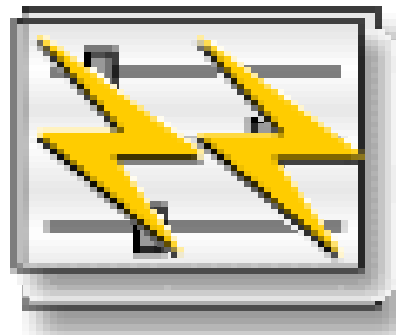
You can set the following Miscellaneous run-time options for a Vuser script:

- Error Handling
- Multithreading
  - Vusers support multithreaded environments. The primary advantage of a multithread environment is the ability to run more Vusers per load generator.
- Automatic Transactions





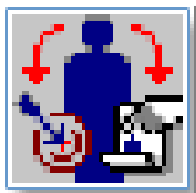
## CONTROLLER



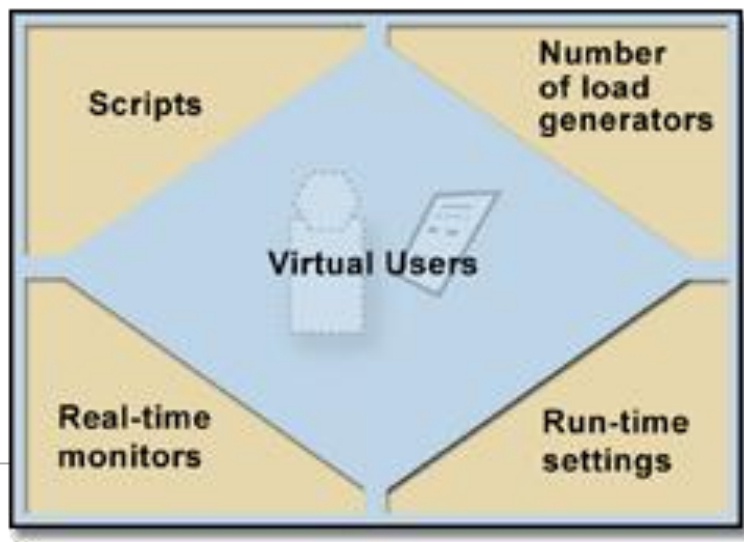
# LoadRunner 8.0 - Controller

- **What is Scenario?**

A scenario is a file that defines the Vusers execution, the number of Vusers to run, the goals of the test, the computer that hosts the Vusers, and the conditions under which to run the Load Test

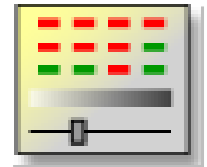


## Elements of a Scenario



# LoadRunner 8.0 - Controller

- **Controller organizes and manages scenario elements**
- **During scenario execution the controller :**
  - Runs Vuser Groups
  - Controls the initialize, run, pause, and stop conditions of each Vuser
  - Displays the status of each Vuser
  - Displays any messages from Vusers
  - Monitors system and network resources



# LoadRunner 8.0 - Controller

- Types of Scenarios

- **Manual Scenario**

- Manage your Load Test by specifying the number of Virtual users to run

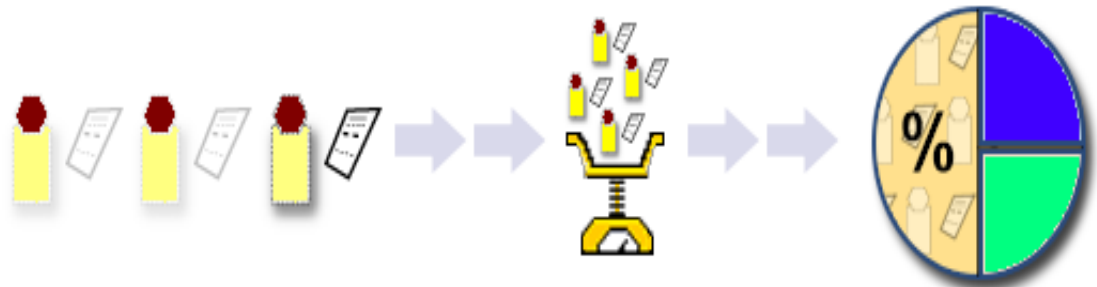
- **Goal-Oriented Scenario**

- Allow LoadRunner Controller to create a Scenario based on the goals you specify

# LoadRunner 8.0 - Controller

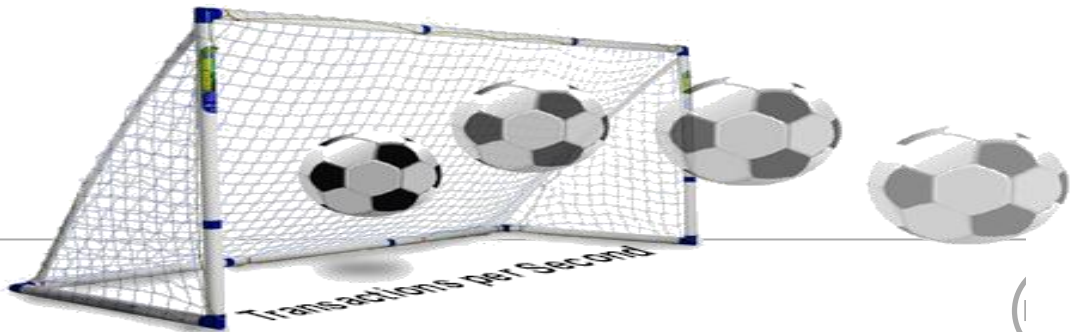
- Manual Scenario

- You control the number of Running Vusers at the time which they Run.
- You can specify how many Vusers run simultaneously
- Allows you to run the Vuser in Percentage mode



# LoadRunner 8.0 - Controller

- Goal-Oriented Scenario
  - Determine your system to achieve the particular goal
  - The goal may be **number of hits per second**, **Number of transaction per second**, etc.,
  - Manages Vusers Automatically to maintain and achieve the goal



# LoadRunner 8.0 - Controller

- Which scenario to use?

Examples

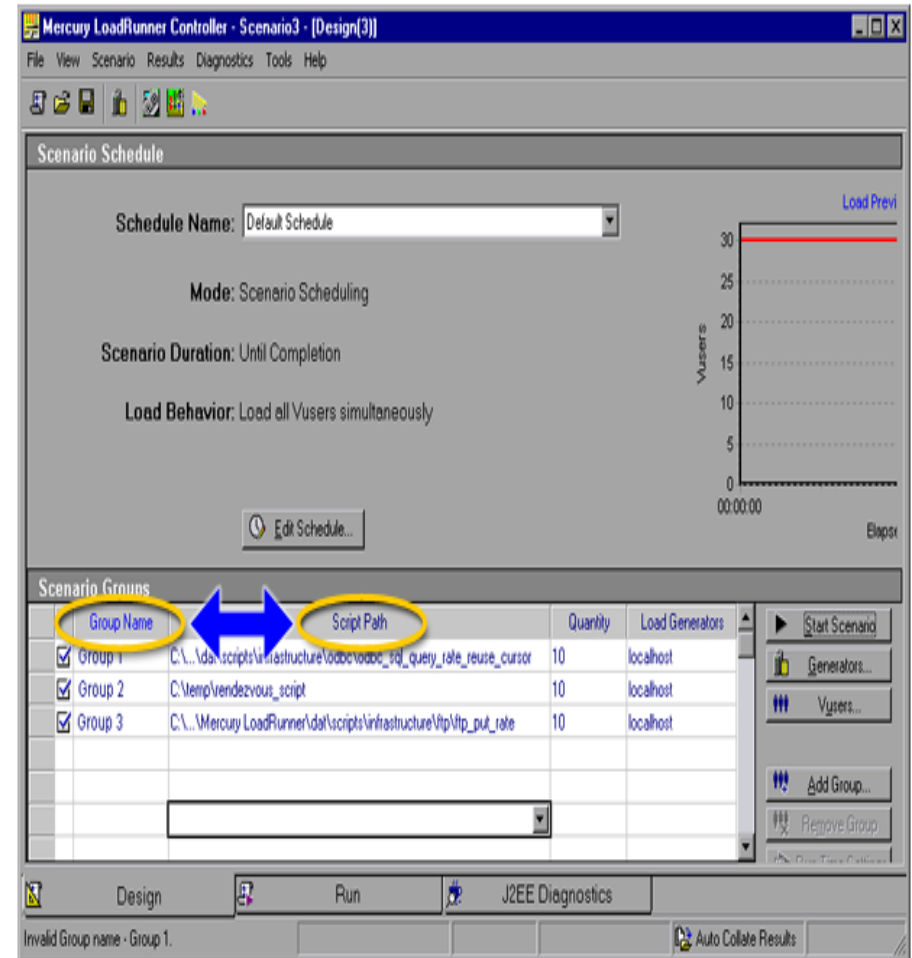
Scenario Outline	Scenario Type
<ul style="list-style-type: none"><li>•Script Should define Update</li><li>•When running the Load Test at peak load achieve 1000 concurrent users</li></ul>	Manual Scenario with 1000 users
<ul style="list-style-type: none"><li>•Define search transaction</li><li>•Response time of 8 seconds with 2000 concurrent users during non-peak hours</li><li>•Achieve response time of 12 Secs with 5000 concurrent users during peak hours</li></ul>	Goal-Oriented with transaction time as the 'Goal Type'



# LoadRunner 8.0 - Controller

## Vuser Groups

- Scenario consists of group of Vusers which emulate the Human users to interact with your application
- Each script you select is assigned a Vuser group
- Each Vuser group is assigned a number of Vusers
- You can Assign different script to each Vuser or You can assign the same script to all the Vusers

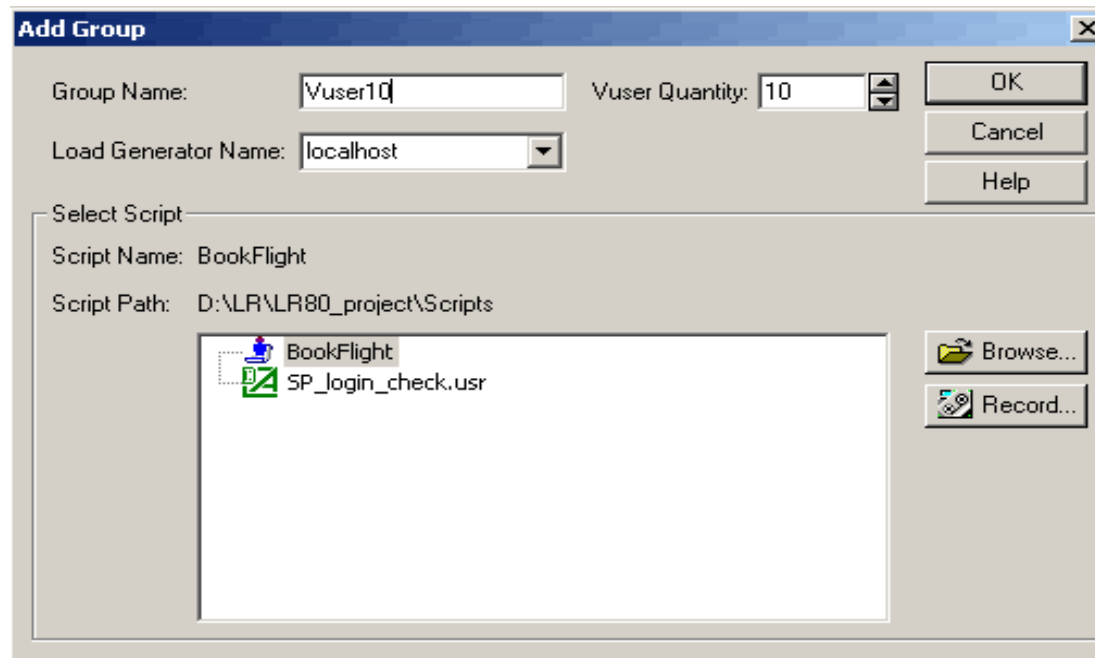




# LoadRunner 8.0 - Controller

- **Adding Vuser Group**

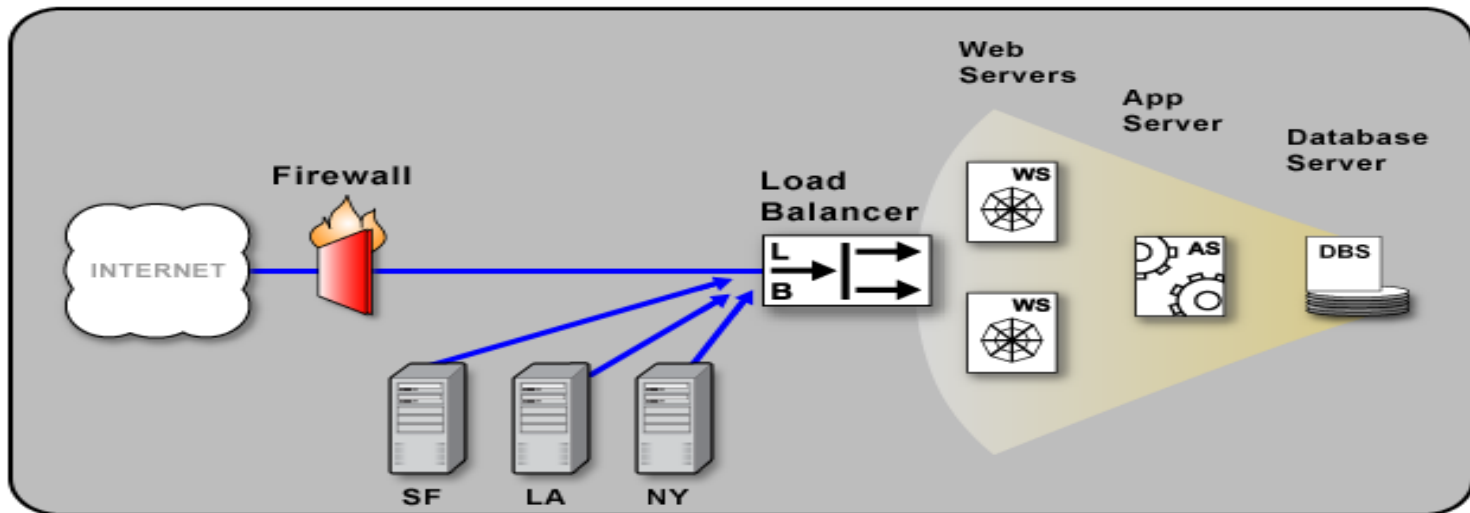
- Group Name
- Vuser Quantity
- Load Generator name



# LoadRunner 8.0 - Controller

## Load Generator for your Scenario

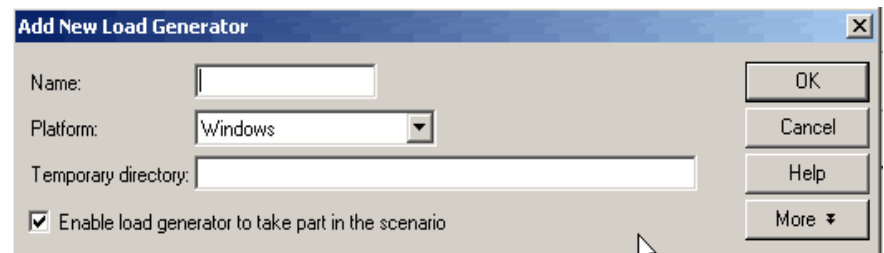
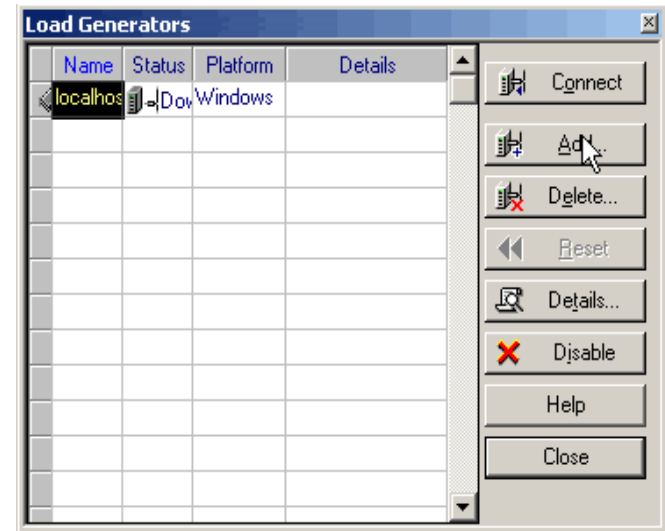
- Load Generator is a machine that serves as the host for running Vusers
- Its important to know that which script need to be run from which location
- For example customer activity, the function of location, workload of location...etc.,



# LoadRunner 8.0 - Controller

## Adding Load Generator

- Click the generators button to open the dialogue box
- Now click the add button to open the Add load generator dialogue box
- Enter the name and load generator platform which you want to add
- A machine must have installed LoadRunner agent to use as a Load Generator



# LoadRunner 8.0 - Controller

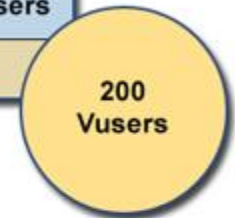
## Assigning Number of Vusers

**Simple scenarios use just one Vuser Script**

**To profile a more complex mix of users, assign several Vuser scripts based on “User profile” in one scenario**

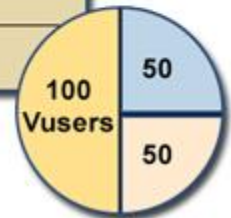
Single Script Example

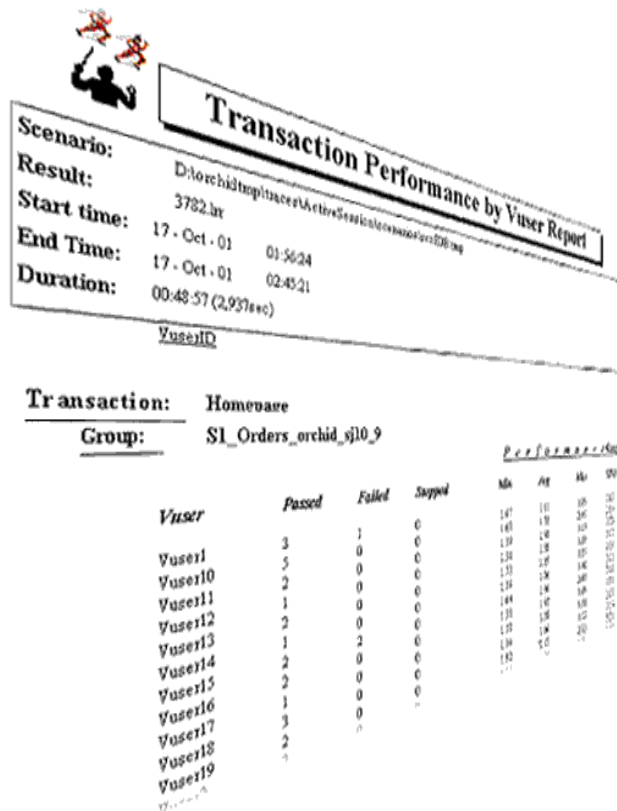
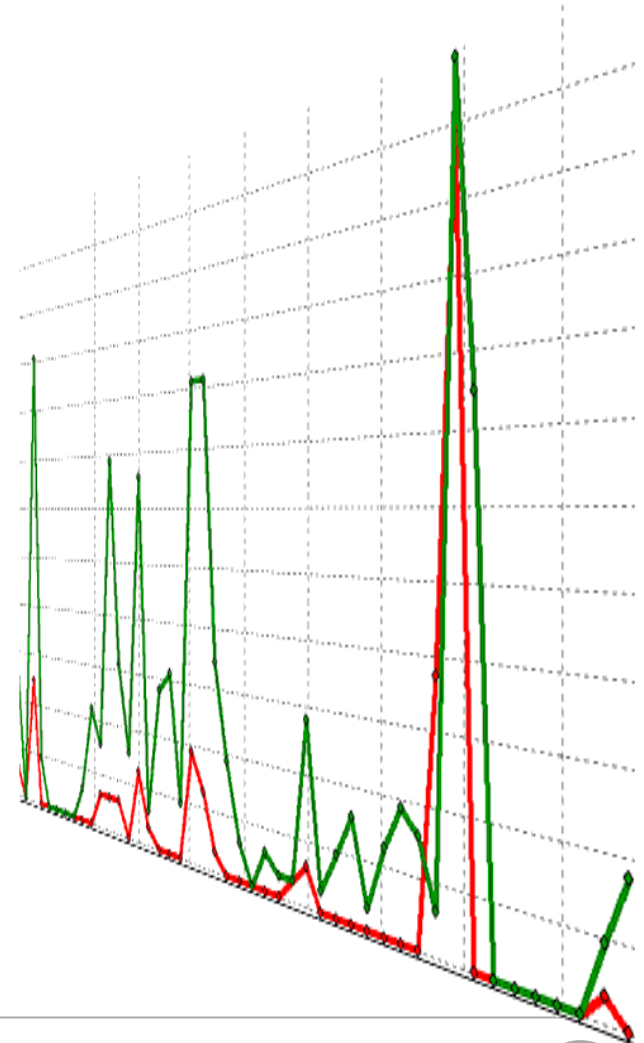
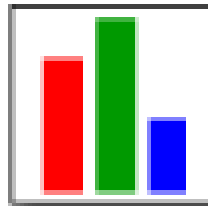
Business Processes	Number of Users
Purchase Ticket	200 (100%)



Multiple Script Example

Business Processes	Number of Users
Purchase Ticket	100 (50%)
View Flights	50 (25%)
Search Flights	50 (25%)





# LoadRunner Analysis

Analysis provides graphs and reports to help you analyze the performance of your system. These graphs and reports summarize the scenario execution.

*Using these graphs and reports, you can easily pinpoint and identify the bottlenecks in your Application*

# LoadRunner Analysis

To view a summary of the results *after* test execution, you can use one or more of the following tools:

- **Vuser log files** contain a full trace of the scenario run for each Vuser. These files are located in the scenario results directory.
- **Controller Output window** displays information about the scenario run.
- **Analysis graphs** help you determine system performance and provide information about transactions and Vusers.
- **Graph Data** and **Raw Data** views display the actual data used to generate the graph in a spreadsheet format.
- **Report** utilities enable you to view a Summary HTML report for each graph or a variety of Performance and Activity reports. You can create a report as a Microsoft Word document, which automatically summarizes and displays the test's significant data in graphical and tabular format.



# Analysis Basis



# LoadRunner - Analysis

## Creating Analysis Session

- When you run a scenario, data is stored in a result file with an **.lrr** extension. Analysis is the utility that processes the gathered result information and generates graphs and reports.
- When you work with the Analysis utility, you work within a *session*. An Analysis session contains at least one set of scenario results (**lrr file**). Analysis stores the display information and layout settings for the active graphs in a file with an **.lra** extension.



# LoadRunner - Analysis

## Methods of opening LoadRunner Analysis

- Open **Analysis** directly from the controller (**Results > Analyze Results**)
- **Start > Programs > Mercury LoadRunner > Applications > Analysis**
- **Start > Programs > Mercury LoadRunner > LoadRunner**, select the **Load Testing** or **Tuning tab**, and then click **Analyze Load Tests** or **Analyze Tuning Sessions**.
- You can also instruct controller to open analysis automatically after the Scenario execution by selecting **Results > Auto Analysis**



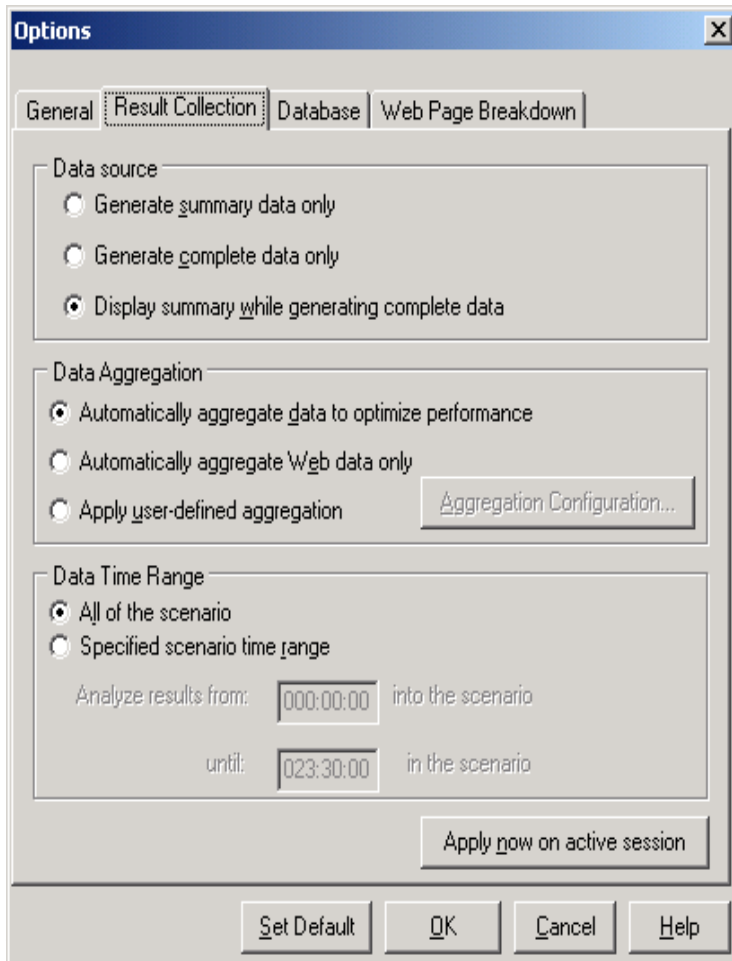
# Collating Execution Results

- When you run a scenario, by default all Vuser information is stored locally on each Vuser host
- After scenario execution the results are automatically collated or consolidated – results from all the hosts are transfer to results directory
- You disable automatic collation by choosing Results > Auto collate Results from the controller window
- You can collate manually by selecting Results > Collate Results
- If your results are not collated Analysis will automatically collate the results before generating the analysis data



# Viewing Summary Data

## Analysis : Tools → Options



### Generate Summary data only

View the summary data only. If this option is selected Analysis won't Process the data for advanced use with filtration

### Generate Complete data only

View only the complete data only after it has been Processed. Do not display the Summary

### Display Summary while generate Complete data only

View summary data while the complete data is being processed. After the processing, view the complete data. A bar below the graph indicates the complete data generation progress.



# Data Aggregation

- **Aggregate Data:**

Specify the data you want to aggregate in order to reduce the size of the database.

- **Select the type of data to aggregate:**

Specify the type(s) of graphs for which you want to aggregate data.

- **Select the graph properties to aggregate:**

Specify the graph properties— Vuser ID, Group Name, and Script Name—you want to aggregate. If you do not want to aggregate the failed Vuser data, select Do not aggregate failed Vusers.

The screenshot shows a 'Data Aggregation Configuration' dialog box. It has a title bar with a close button. The main area is titled 'Aggregation configuration'. It contains a radio button labeled 'Aggregate Data (available only for complete data)'. Below this, there's a section 'Select the type of data to aggregate:' with five checkboxes: 'Transactions (Response time, Per second)' (checked), 'Web (Hits per second, Throughput, Pages per second, HTTP return codes)' (checked), 'Monitors' (unchecked), 'Data Points' (checked), and 'Script Errors' (unchecked). Another section 'Select the graph properties to aggregate:' has four checkboxes: 'UserID' (checked), 'Group Name' (unchecked), 'Script Name' (unchecked), and 'Do not aggregate failed Vusers' (unchecked). Below these is a section 'Select the granularity you want to use:' with a numeric input field set to '1' and a unit dropdown set to 'seconds'. At the bottom, there's a radio button labeled 'Web data aggregation only' and a section 'Use granularity of' with a numeric input field set to '5' and a unit dropdown set to 'seconds' for 'Web data.'. The dialog box has 'OK', 'Cancel', and 'Help' buttons at the bottom right.

**Data Aggregation Configuration**

Aggregation configuration

☒ Aggregate Data (available only for complete data)

Select the type of data to aggregate:

☒ Transactions (Response time, Per second)

☒ Web (Hits per second, Throughput, Pages per second, HTTP return codes)

☐ Monitors

☒ Data Points

☐ Script Errors

Select the graph properties to aggregate:

☒ UserID ☐ Group Name ☐ Script Name

☐ Do not aggregate failed Vusers

Select the granularity you want to use: 1 seconds

☐ Web data aggregation only

Use granularity of 5 for Web data.

OK Cancel Help

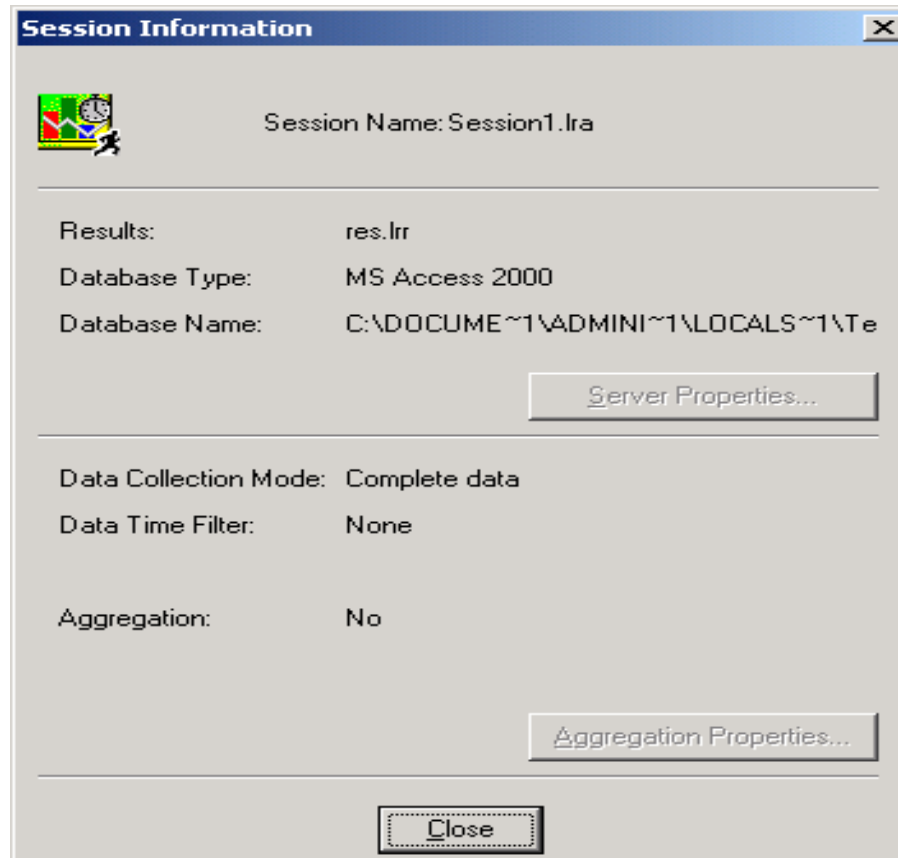
# Setting Database Options

- You can choose the database in which to store Analysis session result data and you can repair and compress your Analysis results and optimize the database that may have become fragmented.
- By default, LoadRunner stores Analysis result data in an Access 2000 database.
- If your Analysis result data exceeds two gigabytes, it is recommended that you store it on an SQL server

The screenshot shows the 'Options' dialog box with the 'Database' tab selected. The 'Access 2000' radio button is selected. The 'Server Name' is set to 'BYTES'. The 'User Name' is 'sa'. The 'Password' field is empty. The 'Logical storage location' is '\\BYTES\Data\'. The 'Physical storage location' is 'C:\MSSQL7\Data\'. There are buttons for 'Test parameters', 'Compact database', 'Set Default', 'OK', 'Cancel', and 'Help'.

# Session Information

You can view the properties of the current Analysis session in the Session Information dialog box.



# Analysis Graphs

## Analysis graphs are divided into the following categories:

- **Vuser Graphs** - Provide information about Vuser states and other Vuser statistics.
- **Error Graphs** - Provide information about the errors that occurred during the scenario.
- **Transaction Graphs** - Provide information about transaction performance and response time.
- **Web Resource Graphs** - Provide information about the throughput, hits per second, HTTP responses per second, number of retries per second, and downloaded pages per second for Web Vusers.
- **Web Page Breakdown Graphs** - Provide information about the size and download time of each Web page component.



# Analysis Graphs

- **User-Defined Data Point Graphs** - Provide information about the custom data points that were gathered by the online monitor.
- **System Resource Graphs** - Provide statistics relating to the system resources that were monitored during the scenario using the online monitor.
- **Network Monitor Graphs** - Provide information about the network delays.
- **Firewall Server Monitor Graphs** - Provide information about firewall server resource usage.
- **Web Server Resource Graphs** - Provide information about the resource usage for the Apache, iPlanet/Netscape, iPlanet(SNMP), and MS IIS Web servers.



# Analysis Graphs

- **Web Application Server Resource Graphs** - Provide information about the resource usage for various Web application servers.
- **Database Server Resource Graphs** - Provide information about database resources.
- **Streaming Media Graphs** - Provide information about resource usage of streaming media.
- **ERP/CRM Server Resource Graphs** - Provide information about ERP/CRM server resource usage.
- **Java Performance Graphs** - Provide information about resource usage of Java-based applications.
- **Application Component Graphs** - Provide information about resource usage of the Microsoft COM+ server and the Microsoft NET CLR server.
- **Application Deployment Solutions Graphs** - Provide information about resource usage of the Citrix MetaFrame and 1.8 servers.



# Analysis Graphs

- **Middleware Performance Graphs** - Provide information about resource usage of the Tuxedo and IBM WebSphere MQ servers.
- **Security Graphs** - Provide information about simulated attacks on the server using the Distributed Denial of Service graph.
- **Application Traffic Management Graphs** - Provide information about resource usage of the F5 BIG-IP server.
- **Infrastructure Resources Graphs** - Provide information about resource usage of FTP, POP3, SMTP, IMAP, and DNS Vusers on the network client.
- **Siebel Diagnostics Graphs** - Provide detailed breakdown diagnostics for transactions generated on Siebel Web, Siebel App, and Siebel Database servers.
- **Siebel DB Diagnostics Graphs** - Provide detailed breakdown diagnostics for SQLs generated by transactions on the Siebel system.

# Analysis Graphs

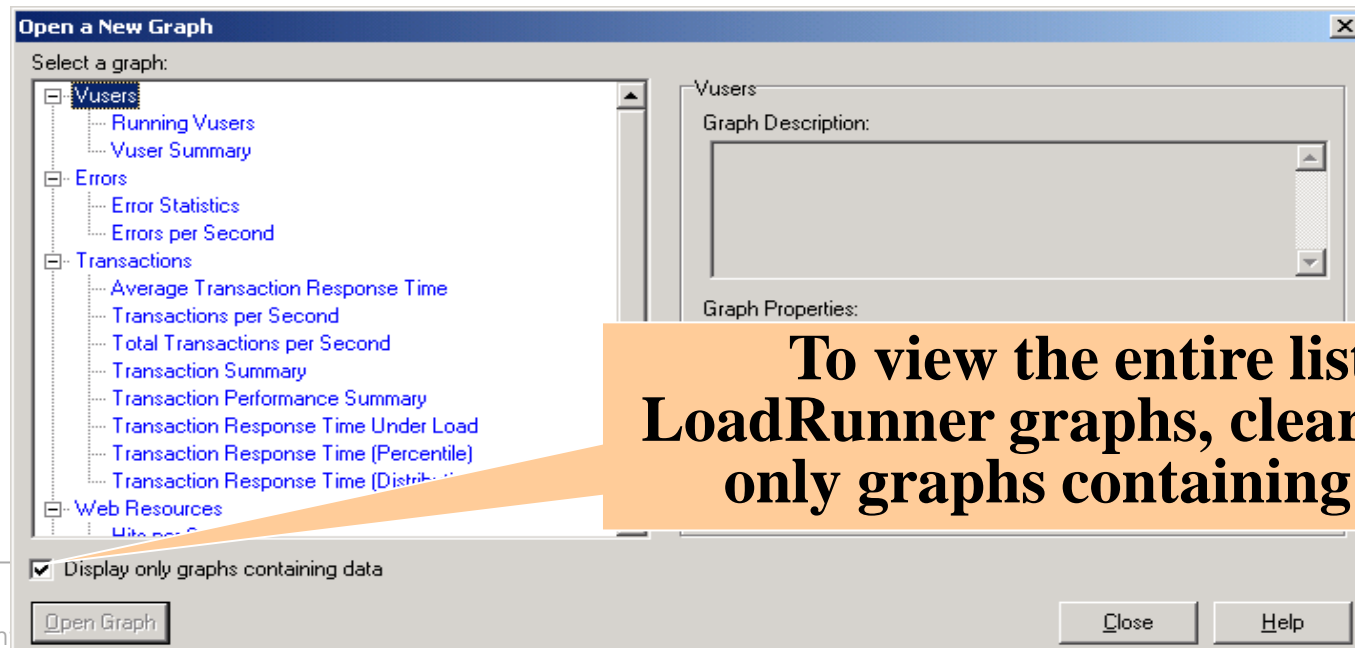
- **Oracle Diagnostics Graphs** - Provide detailed breakdown diagnostics for SQLs generated by transactions on the Oracle NCA system.
- **J2EE Diagnostics Graphs** - Provide information to trace, time, and troubleshoot individual transactions through J2EE Web, application, and database servers.



# Adding New Graph

## Graph > Add Graph, or click <New Graph>

- Graphs that contain data are listed in blue. By default, only graphs that contain data are listed. To view the entire list of LoadRunner graphs, clear Display only graphs containing data.
- Use the Scenario Elapsed Time field to limit the time range for which graph data is displayed.



# Filtering & Sorting Graph Data

You can filter and sort data that is displayed in a graph. You sort and filter graph data using the same dialog box.

## Filtering Graph Data

- You can filter graph data to show fewer transactions for a specific segment of the scenario.
- More specifically, you can display four transactions beginning from five minutes into the scenario and ending three minutes before the end of the scenario.
- You can filter for a single graph, in all graphs in a scenario, or in the summary graph.

# Filtering & Sorting Graph Data

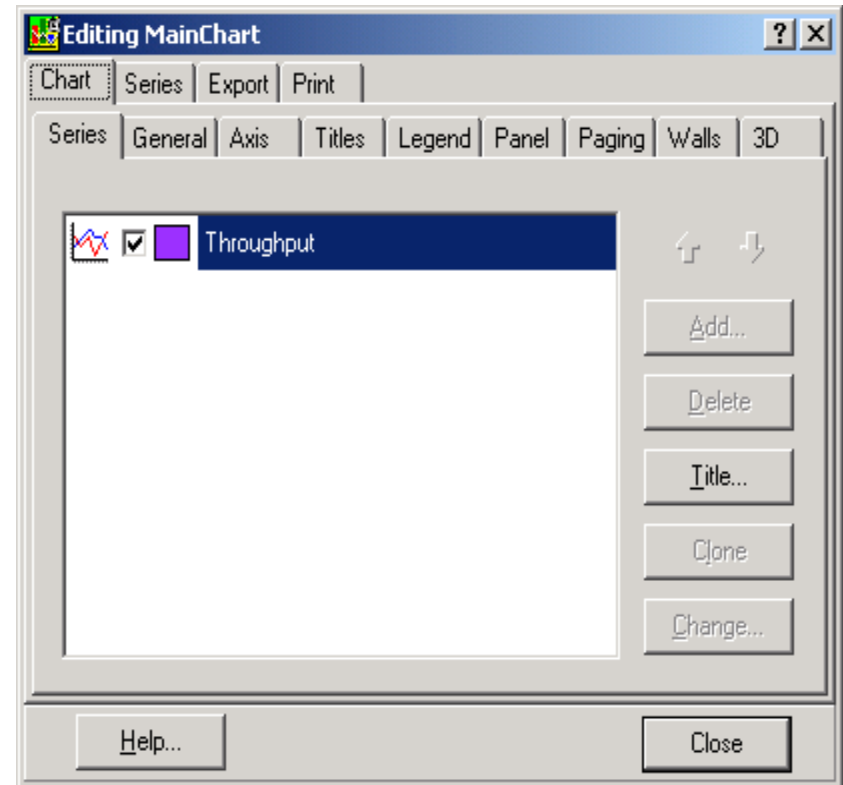
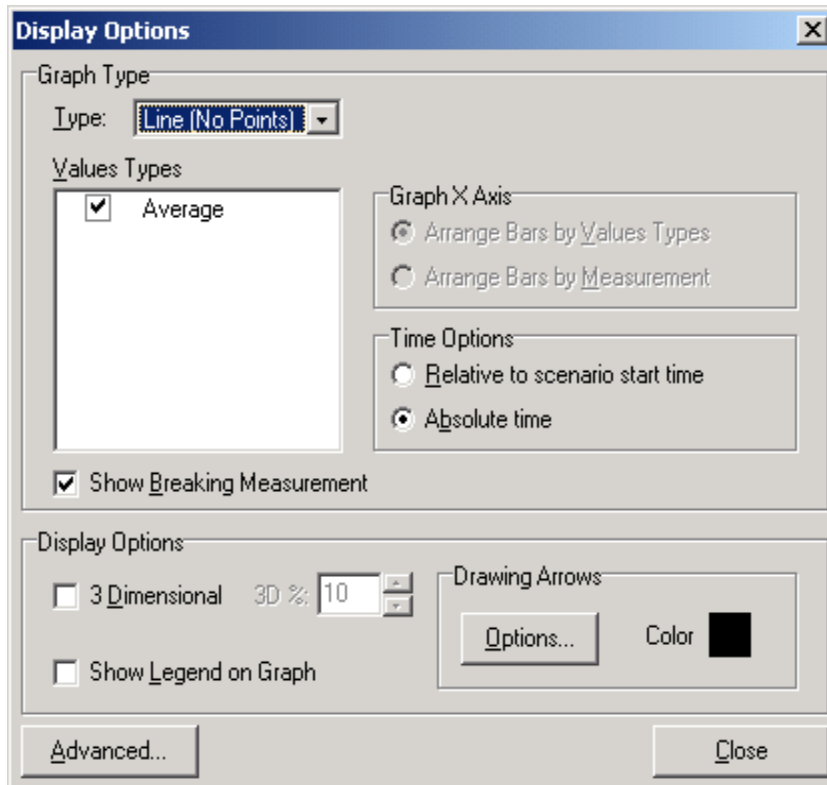
## Sorting Graph Data

- You can sort graph data to show the data in more relevant ways.
- *For example*, Transaction graphs can be grouped by the Transaction End Status, and Vuser graphs can be grouped by Scenario Elapsed Time, Vuser End Status, Vuser Status, and VuserID.

The image shows a 'Graph Settings' dialog box with a tab labeled 'Transaction Summary'. The 'Graph: Transaction Summary' section contains a 'Filter condition:' table. The table has three columns: 'Criteria', 'Values', and an empty column. The first row is 'Transaction Name' with a value of 'T\_01\_SignInToBeginSearch'. The second row is 'Transaction Response Time' with a value of '(((3.9071 <= Transaction Response Time) and (366.8382 >= Transaction Response Time))'. The third row is 'Scenario Elapsed Time' with a value of 'From VuserID: 1 to VuserID: 2'. The fourth row is 'Transaction Hierarchial Path' with a value of 'From VuserID: 1 to VuserID: 2'. The fifth row is 'Transaction End Status' with a value of 'From VuserID: 1 to VuserID: 2'. The sixth row is 'VuserID' with a value of 'From VuserID: 1 to VuserID: 2'. The seventh row is 'Think Time' with a value of 'From VuserID: 1 to VuserID: 2'. Below the table is a 'Group By:' section. It has two boxes: 'Available groups:' containing 'Transaction Hierarchial Path' and 'Selected groups:' containing 'VuserID'. There are arrows between the boxes to move items. On the right side of the dialog are buttons for 'OK', 'Cancel', 'Help', and 'Set Default'.

Criteria	Values
Transaction Name	T_01_SignInToBeginSearch
Transaction Response Time	(((3.9071 <= Transaction Response Time) and (366.8382 >= Transaction Response Time))
Scenario Elapsed Time	From VuserID: 1 to VuserID: 2
Transaction Hierarchial Path	From VuserID: 1 to VuserID: 2
Transaction End Status	From VuserID: 1 to VuserID: 2
VuserID	From VuserID: 1 to VuserID: 2
Think Time	From VuserID: 1 to VuserID: 2

# Configuring Basic Graph Display Options

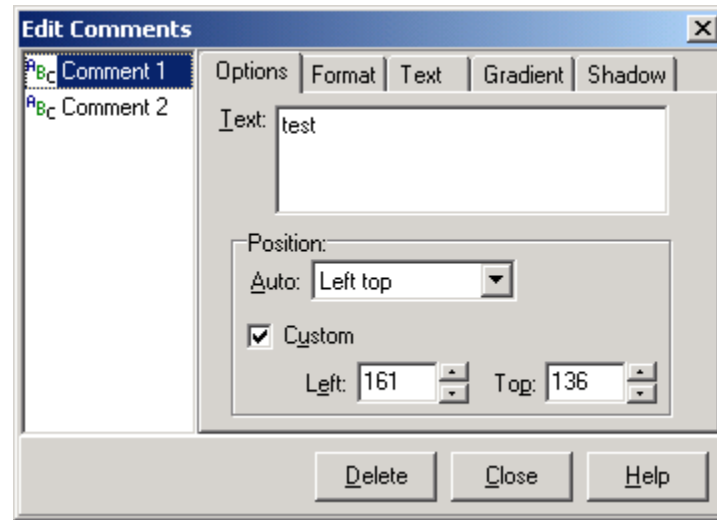
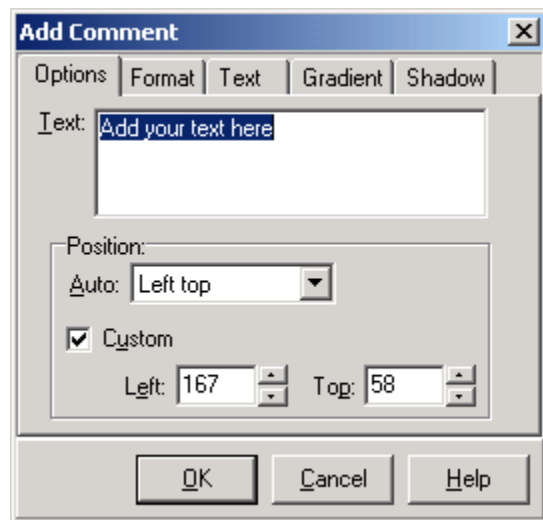


View → Display Options

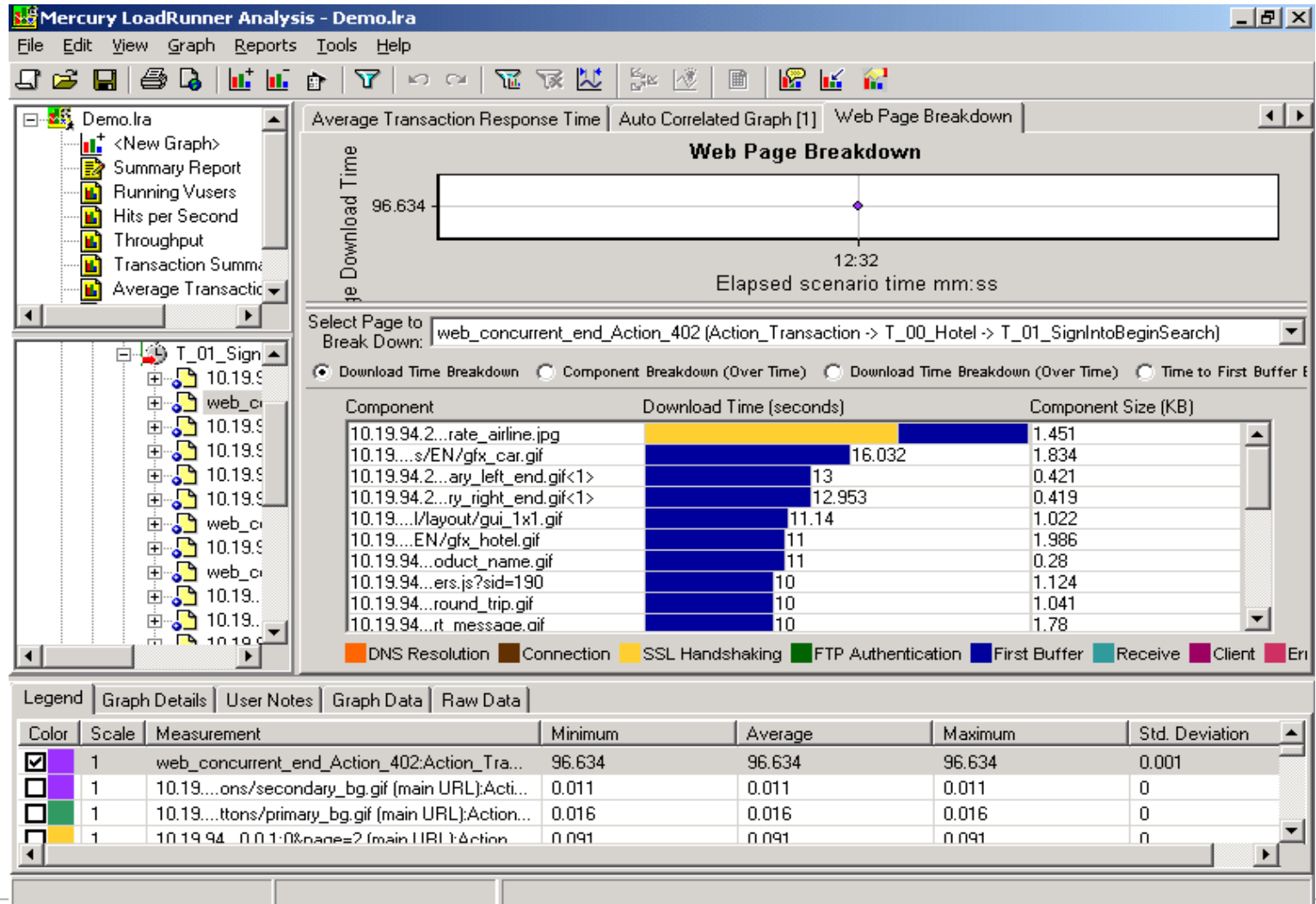


# Configuring Basic Graph Display Options

- Adding Comments and Arrows



# Web Page Break Down



# Transaction Report

- Transaction reports provide performance information about the transactions defined within the Vuser scripts. These reports give you a statistical breakdown of your results and allow you to print and export the data.
- Transaction Reports are divided into the following categories
  - **Activity**
  - **Performance**
    - *Data Point, Detailed Transaction, Transaction Performance by Vuser*

**Activity reports** provide information about the number of Vusers and the number of transactions executed during the scenario run. The available Activity reports are *Scenario Execution*, *Failed Transaction*, and *Failed Vusers*.

**Performance reports** analyze Vuser performance and transaction times. The available Performance reports are *Data Point*, *Detailed Transaction*, and *Transaction Performance by Vuser*.

# Performance Monitoring

- Metrics to Be Measured on All Servers

Object	Counter	Instance
<b>Network</b>		
Network Interface	Bytes Received/sec	Each NIC card
Network Interface	Bytes Sent/sec	Each NIC card
Network Interface	Packets Received Discarded	Each NIC card
Network Interface	Packets Outbound Discarded	Each NIC card
<b>Processors</b>		
Processor	% Processor Time	_Total
Processor	% Interrupt Time	_Total
Processor	% Privileged Time	_Total
System	Processor Queue Length	N/A
System	Context Switches/sec	N/A



# Performance Monitoring

Object	Counter	Instance
<b>Memory</b>		
Memory	Available MBytes	N/A
Memory	Pages/sec	N/A
Memory	Cache Faults/sec	N/A
Server	Pool Nonpaged Failures	N/A
<b>Process</b>		
Process	Page Faults / sec	Total
Process	Working Set	(Monitored process)
Process	Private Bytes	(Monitored process)
Process	Handle Count	(Monitored process)



# Technology Specific Monitors

- **Throughput:** ASP.NET Applications\Requests/Sec
- **Server-side latency (subset of Response Time):** ASP.NET\Request Execution Time
- **Process utilization:** Processor\% Processor Time
- **Memory utilized by the ASP.NET worker process:** Process\Private Bytes (aspnet\_wp)
- **?Free memory available for the server:** Memory\Available MBytes

**Table 16.10: Metrics for Application Performance Goals**

Object	Counter	Instance
ASP.NET Applications	Requests/Sec	Application virtual directory
ASP.NET	Request Execution Time	N/A
ASP.NET Applications	Requests In Application Queue	Application virtual directory
Processor	% Processor Time	_Total
Memory	Available MBytes	N/A
Process	Private Bytes	aspnet_wp



# SQL Server Specific

Object	Counter	Instance
SQL Server: General Statistics	User Connections	N/A
SQL Server: Access Methods	Index Searches/sec	N/A
SQL Server: Access Methods	Full Scans/sec	N/A
SQL Server: Databases	Transactions/sec	(Your Database)
SQL Server: Databases	Active Transactions	(Your Database)
SQL Server: Locks	Lock Requests/sec	_Total
SQL Server: Locks	Lock Timeouts/sec	_Total
SQL Server: Locks	Lock Waits/sec	_Total
SQL Server: Locks	Number of Deadlocks/sec	_Total
SQL Server: Locks	Average Wait Time (ms)	_Total
SQL Server: Latches	Average Latch Wait Time (ms)	N/A
SQL Server: Cache Manager	Cache Hit Ratio	_Total
SQL Server: Cache Manager	Cache Use Counts/sec	_Total

# Reporting

- Reporting Should have the Following

Section	Item	Details
Hardware details	Web server(s)	Processor: 2 gigahertz (GHz) (dual processor) Memory: 1 GB RAM Number of servers: 2 Load balancing: Yes
	Server(s) running SQL Server	Processor: 2 GHZ (dual processor) Memory: 1 GB RAM Number of servers: 1 Load balancing: No
	Client(s)	Memory: 1 GB RAM Number of servers: 2
	Network	Total Bandwidth for the setup: 100 megabits per second (Mbps) Network Bandwidth between client and server: 100 Mbps
Software details	Web server(s)	Operating system: Microsoft® Windows 2000® Advanced Server Service Pack SP4 Web server: IIS 5.0 Platform: .NET Framework 1.1
	Servers running SQL Server	Operating system: Windows 2000 Advanced Server Service Pack SP4 Database Server: SQL Server 2000 Service Pack SP3





# Reporting Elements

Configuration details	IIS configuration	<p>Session time out:</p> <p>Http-Keep alive:</p>
	Machine.Config configuration	<p>MaxConnections:</p> <p>MaxWorkerThreads:</p> <p>MaxIOThreads</p> <p>MinFreeThreads</p> <p>MinLocalRequestFreeThreads:</p> <p>executionTimeOut:</p>
	Web.Config configuration	<pre>&lt;compilation debug="false"/&gt; &lt;authentication mode="Windows" /&gt;  &lt;trace enabled="false" requestLimit="10" pageOutput="false" traceMode="SortByTime" localOnly="true" /&gt;  &lt;sessionState mode="InProc" timeout="20"/&gt;</pre>
	Application-specific configuration	<p>You can include application specific configuration here such as custom attributes added to the configuration file.</p>

# Work Load Profile

User profile	Percentage
Browse	50 percent
Search	30 percent
Order	20 percent
Total	100 percent
Metric	Value
Number of simultaneous users	100 – 1,600 users
Total number tests	16
Test duration	1 hour
Think time	random time of 10 seconds

# Performance Objective

Performance Objective	Metric
Throughput and response time	Requests/second: Response time/TTLB:
System performance (for each server)	Processor utilization: Memory: Disk I/O (for database server):
Application-specific metrics (custom performance counters)	Orders/second: Searches/second:



# Performance Details

- **Web Server Metrics**
- **SQL Server Metrics**
- **Throughput versus user load**
- **Response time versus user load**
- **Resource utilization versus user load**
- **Potential Bottlenecks**



# Best Practices for Performance Testing - Do

- Clear the application and database logs after each performance test run. Excessively large log files may artificially skew the performance results.
- Identify the correct server software and hardware to mirror your production environment.
- Use a single graphical user interface (GUI) client to capture end-user response time while a load is generated on the system. You may need to generate load by using different client computers, but to make sense of client-side data, such as response time or requests per second, you should consolidate data at a single client and generate results based on the average values.
- Include a buffer time between the incremental increases of users during a load test.
- Use different data parameters for each simulated user to create a more realistic load simulation.
- Monitor all computers involved in the test, including the client that generates the load. This is important because you should not overly stress the client.
- Prioritize your scenarios according to critical functionality and high-volume transactions.
- Use a zero think time if you need to fire concurrent requests,. This can help you identify bottleneck issues.
- Stress test critical components of the system to assess their independent thresholds.



# Best Practices for Performance Testing – Don't

- Do not allow the test system resources to cross resource threshold limits by a significant margin during load testing, because this distorts the data in your results.
- Do not run tests in live production environments that have other network traffic. Use an isolated test environment that is representative of the actual production environment.
- Do not try to break the system during a load test. The intent of the load test is not to break the system. The intent is to observe performance under expected usage conditions. You can stress test to determine the most likely modes of failure so they can be addressed or mitigated.
- Do not place too much stress on the client test computers.



# Thank You

- Queries are welcome
- Feed Back

