



Transforming LoadRunner Data into Information and Action

2010-02-01



&



=



Start Collaborating Today!

Transforming LoadRunner Data into Information and Action

Introduction

Today's online web applications need to deliver high efficiency and stability while supporting thousands of users simultaneously. Rather than going live and hoping for the best, it's better to test applications beforehand in a test environment. HP's LoadRunner can help do this by simulating hundreds or thousands of concurrent users applying workloads to a system using relatively few hardware resources. With LoadRunner, a tester can divide performance testing requirements into end user scenarios that replace human users with virtual users (Vusers) and emulate the actions of real users working with an application through a recorded script file. LoadRunner's Vusers generate result data as they perform their scripted transactions. After execution of the script, LoadRunner utilizes the data to generate graphs and reports which can be used to analyze and assess the performance of the application under various load scenarios.

LoadRunner, however, has limitations. When testers need information that goes beyond the standard reports or want to do detailed analysis, accessing LoadRunner data is cumbersome. For example, executing the same script and scenario multiple times to do trending analysis or results verification requires the time consuming extraction of raw data from LoadRunner's results files.

Addressing this problem requires an in-depth understanding of the features and functions of LoadRunner, plus a keen knowledge of tools and techniques for organizing and managing this data. But it's worth it as this allows for greater in-depth analysis which can lead to easier identification of performance bottlenecks and benchmarking for improvements over time.

Using LoadRunner's Database

Leveraging LoadRunner's results data beyond the standard reports presents challenges in three key areas: efficiently extracting and saving the data in a "user friendly" format, managing the data, and performing further analysis.

LoadRunner treats the results of each run as a separate data set regardless of which scripts or scenarios the runs use. To extract details such as the run time, script, and scenario, session files need to be opened and reviewed. This is an investigative process since it is not documented where the specific data fields are stored.

By default, LoadRunner stores analysis results data in an Access 2000 database. This is good for small single user applications but often testers work as a team running different scenarios simultaneously on different machines. To organize and make better use of such data collected by LoadRunner, the results data should be saved in a multi-user database such as Microsoft SQL Server 2000/2005. Using a full function multi-user database also provides the benefit of many analysis features as part of the product.

Connecting LoadRunner to MS SQL 2000 is done through Database Options. However, the configuration for SQL Server 2005 can be tricky, in part due to new security features in SQL Server 2005. SQL Server

2005 enables only a limited number of core features and services by default and requires an administrator to enable other features. To connect with LoadRunner, **xp_cmdshell** service in SQL Server 2005 has to be enabled, even though it is not in the default list. This service lets LoadRunner synchronize shared and physical server directories. This important tidbit of information is not mentioned in any of LoadRunner's help resources. To get it to work, you need to have a thorough understanding of the security configuration options of SQL Server 2005 (see Figure 1 below).

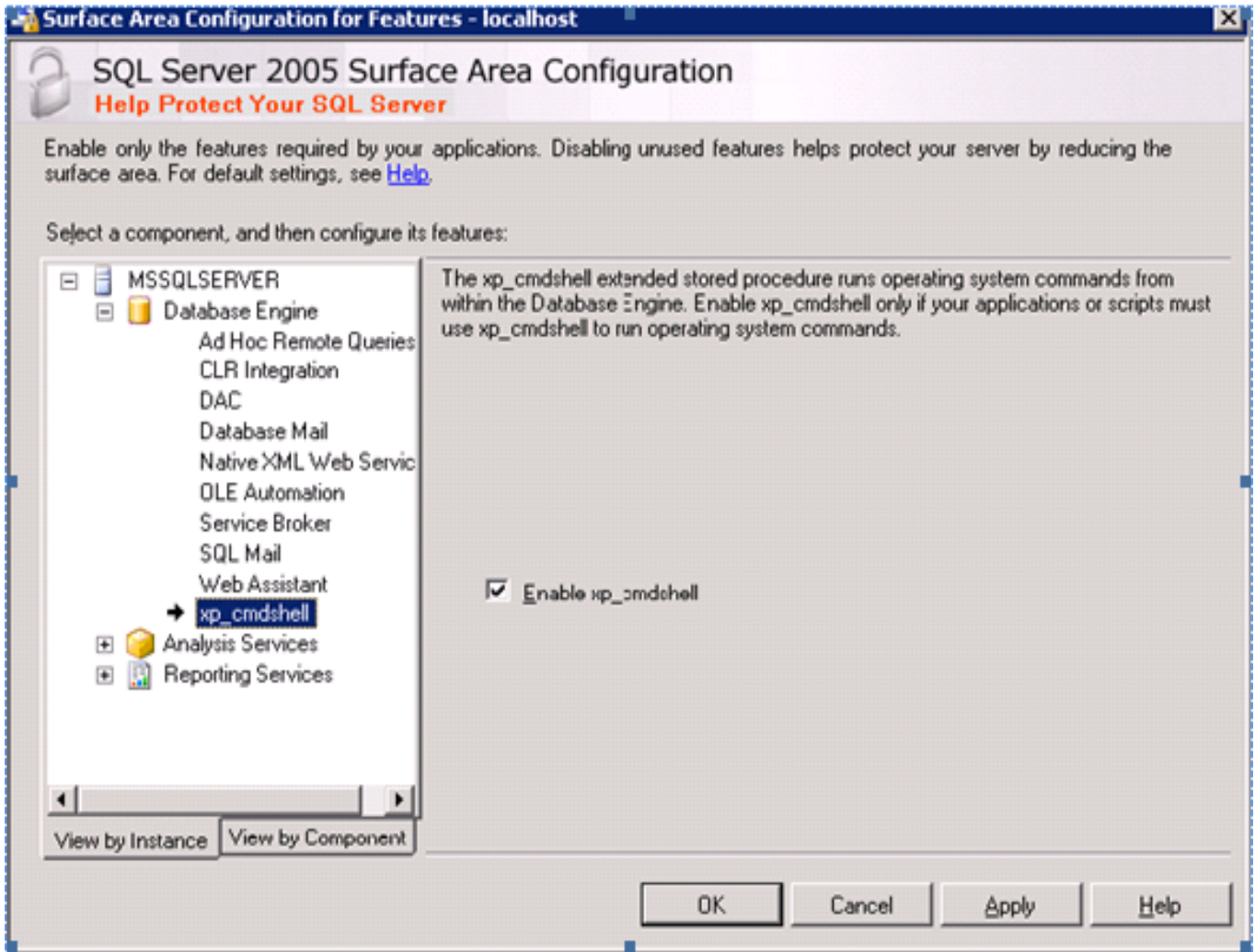


Figure 1 - MS SQL Server 2005 configuration for LoadRunner database connection

Find the right Database

LoadRunner produces analysis results automatically after a run is completed and a related database for test results, but the database has a meaningless file name. It is hard to determine which database contains the results from the previous test run using only the database instance name. However, you can find the name of the correct results database and other related information in the Session file by selecting "File" and then "Session information" in LoadRunner.

```
[Workspace]
SessionStartTime=0
SessionEndTime=1785
VerNo=9
IsSummaryData=0
TimeFormat=0
SesLang=ENU
ABuild=947
DataBaseName=res1222730DB
DBType=2
CurrServer=DBServer
CurrUser=sa
CurrPswd=
StorageLoc=\\DBServer\Data\
PhStorageLoc=C:\Program Files\Microsoft SQL
Server\MSSQL.1\MSSQL\Data\
WinIntSec=1
NumResults=1
```

Figure 2 - Configuration information in a “Session” file

Understanding the Raw Data

To make better use of the data stored in the large number of tables available in the LoadRunner database, we need to first determine which tables store the data of interest and which are irrelevant. Some tables have “regular” names but many others do not. We have found that the “regular” named tables store meaningful raw data, while tables with irregular names are often “repeats” of similar data. An example of this is given below. For raw data related to resource usage, there are two relevant tables: **WebEvent_meter** and **Event_map**.

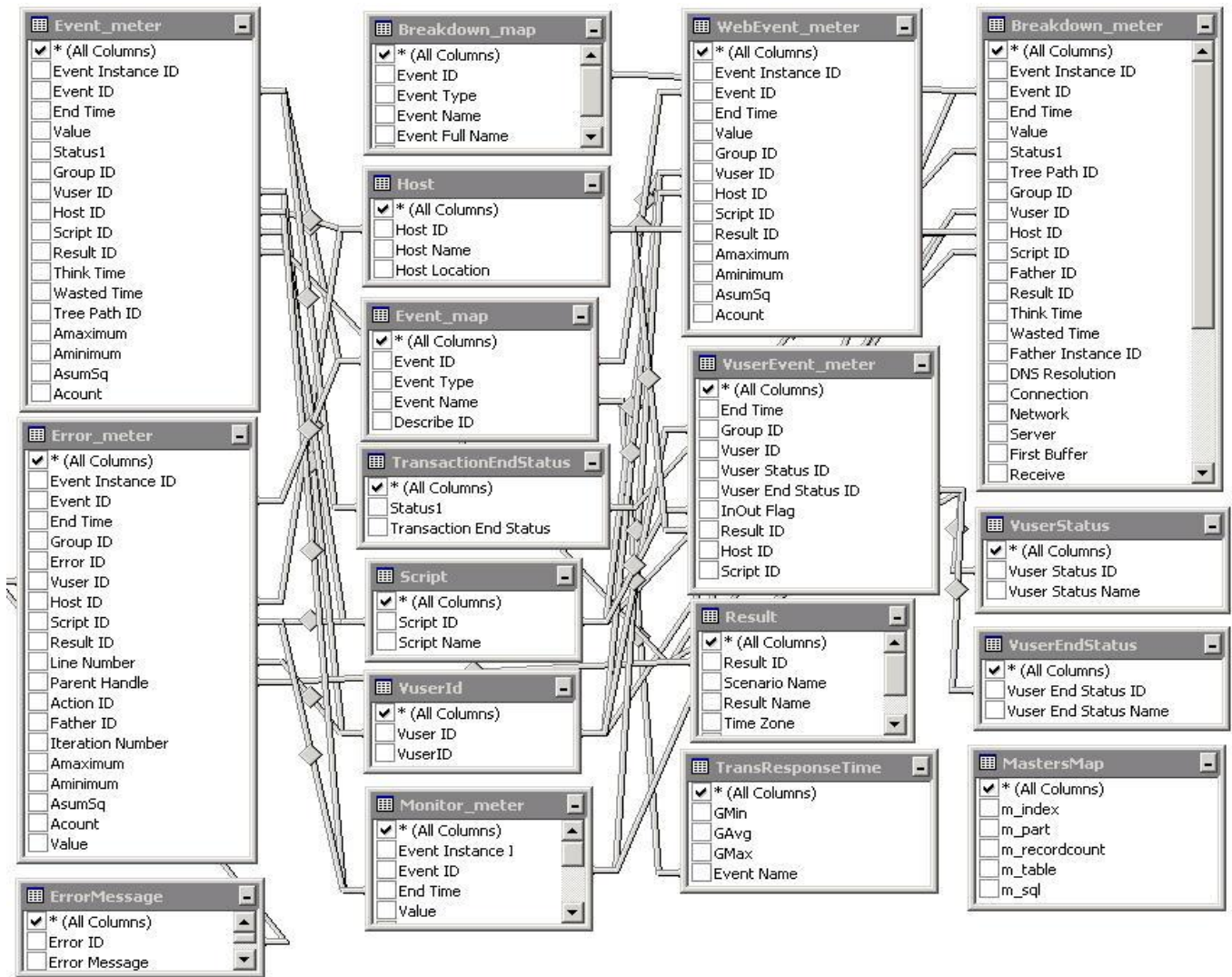


Figure 3 - LoadRunner tables with raw data

To reuse the result data in those “operated” tables, we need to examine the table named **MasterMap** in LoadRunner’s result database. This table has 2 important columns named **m_table** and **m_sql** as shown below in Figure 4.

Table - dbo.MastersMap					
	m_index	m_part	m_recordcount	m_table	m_sql
	0	0	1	T1250452T	SELECT [Scenario Name] INTO [#&%TABLE%@!] From Result ;
	1	0	18	T1249007T	SELECT Sum([Value]*[Account])/5 as [SumGAll], 5 as [Interval - Value], [
	2	0	129	T1247581T	SELECT sum((([Value])*([Account]))/sum([Account]) as [Response_Time], I
	3	0	7	T1245400T	SELECT Min([Aminimum]) as [Minimum], sum((([Value])*([Account]))/sum([
	4	0	7	T1243437T	SELECT Sum([Account]) as [CountAll], [Event Name], [Transaction End SI
	5	0	18	T1241728T	SELECT Sum([Value]*[Account])/5 as [SumGAll], 5 as [Interval - Value], [
	6	0	18	T1239781T	SELECT Sum([Value]*[Account])/5 as [SumGAll], 5 as [Interval - Value], [
	7	0	13	T1238694T	SELECT Sum([InOut Flag]) as [SumAll], (((FLOOR(([End Time]-0)/1))*1
	8	0	11	T1237608T	SELECT Sum([InOut Flag]) as [SumAll], (((FLOOR(([End Time]-0)/2))*2

Figure 4 - MasterMap table

LoadRunner generates **m_table** according to the SQL statements in **m_sql** from which the calculations or logical operations performed by LoadRunner can be viewed. A careful examination of the information is then needed to verify that LoadRunner used the data in the tables of **m_table** to generate related graphs in Analysis. Sometimes the LoadRunner Analysis process generates data in the table which is not correct. After verification, we can store the data in **m_table** for data mining and further analysis.

An Example: Getting Throughput Data

As an example, the throughput graph is a standard report in LoadRunner. This shows the amount of throughput during each second of the load test scenario run. It is measured in bytes and represents the amount of data that the Vusers received from the application at any given second. It helps us evaluate the amount of load that Vusers generate. However, if you want to store the results and trend them over time, or do a graph other than the standard graph, you need to extract the data. Now, let's create a Throughput table which is similar to the one generated by LoadRunner.

1. Open **MasterMap**, and find the **m_sql** for **Throughput** table:

```
SELECT
    Sum([Value]*[Acount])/ G as [SumGAll],
    G as [Interval - Value],
    [Describe ID] as [Value - Describe ID],
    ((( FLOOR( ([End Time]-0)/ G))* G) + 0) as [Granu],
    'Throughput' As EventName
From
    (WebEvent_meter inner join Event_map on (WebEvent_meter.[Event ID] =
    Event_map.[Event ID]))
where
    ([WebEvent_meter].[Event ID] = 15)
group by
    [Describe ID], ((( FLOOR( ([End Time]-0)/ G))* G) + 0)
order by
    ((( FLOOR( ([End Time]-0)/ G))* G) + 0);
```

To gather more details about LoadRunner Analysis graphs, refer to the .def files in the location:

```
disk:\\ ...\Mercury\LoadRunner\bin\dat
```

Putting It All Together

Throughput is just one out of several types of data in LoadRunner. If you want to go beyond the standard graph in LoadRunner and integrate results of different types or do analysis, then you need better results management and data mining capabilities. It is also useful to be able to integrate test results from other commonly used testing tools such as Borland SilkTest, Borland SilkProformer and Mercury WinRunner. For trending analysis, you need to extend dimensions to track important information such as product versions/build numbers, test platform configurations or even test case versions over time. Putting all this

disparate data together into a usable manner will transform it into valuable information that can be used to improve software development processes, verify testing results, and in the end, increase software quality.