



Oracle[®] Trace Integration with Oracle[®] Forms 6i

An Oracle Technical White Paper
April 2000

FORMS 6i INTEGRATION WITH ORACLE TRACE

When you develop and deploy a Forms application, you may want to know what parts of the application can be optimized and what actions are taking time and/or resources. Forms 6i provides different ways to analyze the performance of an application :

- The Performance Collector gives you the ability to identify which tier in your architecture may be the source of a performance problem.

Refer to the “Performance Collector for Oracle Forms 6i” technical white paper for further information about this feature.

- Oracle Trace integration allows you to analyze in depth the performance and the resource consumption of your Forms application.

The integration of Oracle Trace functionality with Forms is a beta feature introduced in Forms Server release 6i and is available starting with Oracle Trace release 2.1.

This white paper discusses how to use Oracle Trace to gather performance information with Forms 6i and how to analyze this data.

WHAT IS ORACLE TRACE

Oracle Trace is part of Oracle Enterprise Manager (OEM). Oracle Trace is the Oracle tool to gather and analyze the performance of the Oracle Database, Forms Server 6i, and other products which implement the Oracle Trace API for tracing.

Oracle Forms 6i ships with some Oracle Trace executables to enable the trace collection. However, to take full advantage of Oracle Trace, you must install Diagnostics Pack, one of the optional packs provided in the Oracle Enterprise Manager product suite. The Diagnostics Pack contains a set of tools to administer the Oracle Trace collections remotely through a GUI interface and to efficiently view the Oracle Trace output.

In this white paper, we discuss :

- What Forms events are traced using Oracle Trace
- How to use Forms and Oracle Trace if you have not installed the Oracle Diagnostics Pack
- How to use Forms and Oracle Trace if you have installed the Oracle Diagnostics Pack

WHAT ARE THE FORMS EVENTS TRACED USING ORACLE TRACE

Oracle Trace only collects data for predefined events, of which there are two types:

- Duration Events
- Point Events

Duration events have a begin point and an end point. For example, a transaction is a duration event. Nested events can occur in a duration event, such as if an error occurs within a transaction.

Point events represent an instantaneous occurrence in the product. For example, the display of a canvas or a dialog are point events.

For Forms Server, those events are defined in a binary file (`ofoms.fdf`), shipped as part as Forms Server 6i and located in the following directory :

NT : <ORACLE_HOME>\net80\admin\fdf

Unix : <ORACLE_HOME>/network/admin/fdf

FORMS DURATION EVENTS AND ITEMS

The following table lists the Forms duration event number and name, describes the event and the specific and generic items associated with it. The items are the information displayed for this particular event.

Note: Each Point event also has a Timestamp. Duration events have a Start Timestamp and an End Timestamp.

For more information about events and items available through Oracle Trace, and for a definition of the generic items, please refer to the "Oracle Trace User's Guide".

Event Number and Name	Description and Items
1. Session	<p>Total time spent by the end user in the Forms Session, including login and logout.</p> <p>Specific items = Session Name, IPAddress.</p> <p>Generic items = UCPU, SCPU, INPUT_IO, OUPUT_IO, PAGEFAULTS, PAGEFAULTS_IO, MAXRS_SIZE.</p>
2. Form	<p>Total time spent by the end user in the specific form and the forms called from this form.</p> <p>Specific item = Form Name.</p> <p>Generic items = UCPU, SCPU, MAXRS_SIZE, CROSS_FAC (Session id).</p>
3. Query	<p>Total time spent by the end user in a specific query.</p> <p>Specific item = Block Name.</p> <p>Generic items = same as for # 2</p>
4. Trigger	<p>Total time spent by the end user in a specific trigger.</p> <p>Specific items = Block Name, Item Name, Trigger Id.</p> <p>Generic items = same as for # 2.</p>
5. LOV	<p>Total time spent by the end user in a LOV.</p> <p>Specific items = LOV Name, Blockname, Itemname</p> <p>Generic items = same as for # 2.</p>
11. Built-In	<p>Total time spent by the end user in a built- in.</p> <p>Specific item = Built-in Name.</p> <p>Generic items = same as for # 2.</p>
12. User Exit	<p>Total time spent by the end user in a user exit.</p> <p>Specific item = User Exit Name.</p> <p>Generic items = same as for # 2.</p>
13. SQL	<p>Total time spent by the end user in SQL code.</p>

	Specific item = SQL Statement. Generic items = same as for # 2.
14. Menu Create	Total time spent by the end user in creating a menu. Specific item = Menu Name. Generic items = same as for # 2.
41. ServerTime	Time spent in processing at the Forms Server. Specific item = none. Generic items = same as for # 2
42. DBTime	Total time spent at database. Specific item = SQL statement. Generic items = same as for # 2.
43. DBLogon	Time spent logging on to the database. Specific item = none. Generic items = same as for # 2.
44. DBLogoff	Time spent logging off from the database. Specific item = none. Generic items = same as for # 2.

FORMS POINT EVENTS AND ITEMS

The following table lists the Forms point event number and name, and describes the event and both specific and generic items associated with it.

Event Number and Name	Description and Items
31. Alert	Instant when an alert occurs. Specific item = Alert Name Generic items = UCPU, SCPU, CROSS_FAC (FormID).
32. Editor	Instant when an editor is invoked. Specific item = Editor Name Generic items = same as for # 31.
33. Window	Instant when a window is created. Specific item = Window Name Generic items = same as for # 31.
34. Canvas	Instant when the user visits a canvas from the user perspective. Specific item = Canvas Name Generic items = same as for # 31.
38. Timer	Instant when a timer activates. Specific item = Timer Name. Generic items = same as for # 31
39. Dialog	Instant when a dialog activates. Specific item = Dialog Name Generic items = same as for # 31

USING FORMS AND ORACLE TRACE WITHOUT DIAGNOSTICS PACK

Even if you have not installed Oracle Enterprise Manager and Oracle Diagnostics Pack, you can still use Oracle Trace to profile your Forms application. When you install Forms Server, the command line trace executables are automatically installed in your ORACLE_HOME and provide a manual way to analyze your Forms application performance.

To use Forms and Oracle Trace without the Diagnostics Pack, you must :

- Start the Trace Collection using a command line and run your Form application in trace mode
- Format the Trace output to produce text files containing the information about your Forms application

STARTING THE COLLECTION

Without the Oracle Diagnostics Pack installed, you need to use the Command Line Interface (CLI) to manage your data collections. The CLI is available from a MS/DOS window on Windows-based system or from the console on Unix-based systems.

1. Start the collection with the otrccol command

The otrccol command is used to manage the data collections:

To start the data collection, enter the following command :

```
otrccol start job_id input_parameter_file
```

Where:

- **job_id** can be any numeric value (e.g. : 1234)
- **input_parameter_file** is a text file containing specific parameter values that are required to start the collection
This file must use the following format :

```
col_name= my_collection
dat_file= <usually same as collection name>.dat
cdf_file= <usually same as collection name>.cdf
fdf_file= oforms.fdf
regid= 1 192216243 0 0 45 <database SID>
```

Where :

- col_name is the unique name you want to give to the collection
- dat_file is the name of the trace file that will be created
- cdf_file is the name of the trace definition file that will be created
- fdf_file is the name of the file containing the forms events that will be traced (this file is always of forms . fdf)
- regid is the registration identifier which contains the code for the company (192216243 is Oracle), the code for the product (45=Forms) and other internal information. The <SID> is mandatory but is not used when the Diagnostics Pack is not present (you need to enter a SID even if it doesn't exist).

Once you have started the collection, a process is running and waiting for a Forms application to run in trace mode.

2. Start your Forms in Trace mode

To start your Form in Trace mode, you need to add the parameter **pecs=trace** to the command line.

For example,. in Client/Server on NT :

```
ifrun60 module=myModule userid=scott/tiger@orcl pecs=trace
```

For example,. on the web using Forms Server, in the HTML page :

include 'pecs=trace' as part of the 'serverArgs' parameter defined in the HTML file used for running the form

For example,. on the web using Forms Server and the Forms CGI :

include 'otherParams=pecs=trace' as part of your specific section in the formsweb.cfg file

You application is now running in trace mode, and all the events occurring during the execution of the application are written to two output files specified in the parameter file used to start the collection.

3. Locate the output

Once you have run your Forms application in trace mode, two files have been generated :

- a file with the dat extension containing the trace output in binary format
- a file with the cdf extension containing the trace definition

Those files are located in the following directory :

NT :

```
<ORACLE_HOME>\otrace80\admin\cdf
```

Unix :

```
<ORACLE_HOME>/otrace/admin/cdf
```

4. Stop the collection with the otrccol command

To stop the data collection, enter the following command :

```
otrccol stop job_id input_parameter_file
```

where `job_id` and `input_parameter_file` are the same parameters as the one used to start the collection.

For more information about the otrccol command, please refer to the “Oracle Trace User’s Guide”.

FORMATTING THE OUTPUT

Once you have generated the trace files, you are ready to format the output to view the information you have gathered during the Forms session.

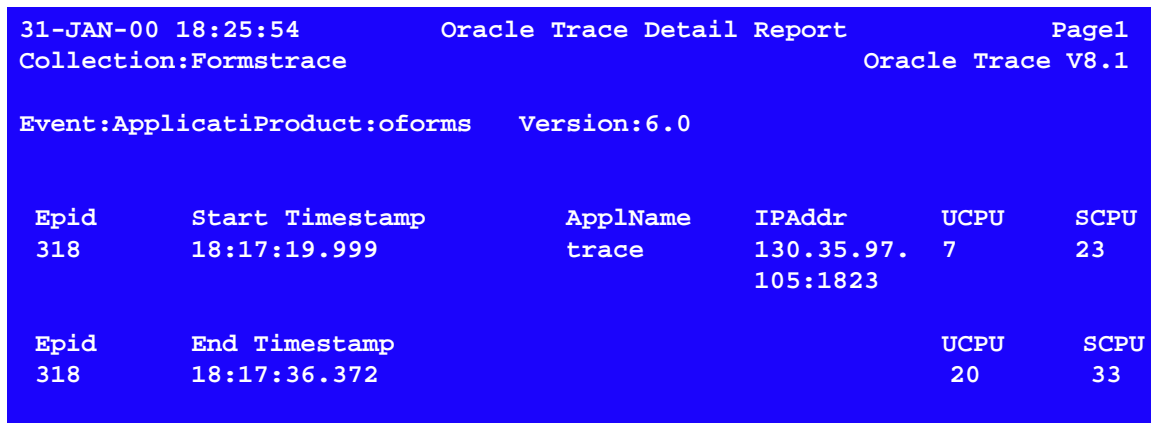
When you don't have the Oracle Diagnostics Pack installed, you can produce text files from the trace files using the Oracle Trace statistics reporting utility.

The Oracle Trace statistics reporting utility displays statistics for all items associated with each occurrence of a server event. These reports can be quite large. You can control the report output by using command parameters. Use the following command and optional parameters to produce a report:

```
otrcrrep [optional parameters] collection_name.cdf
```

Oracle Trace creates one text file per event and includes the entire set of items in that event. For example, one file for the triggers, one file for the built-ins, one file for the network, and so on.

The following screen illustrates an Oracle Trace Detail Report :



```
31-JAN-00 18:25:54      Oracle Trace Detail Report      Page1
Collection:Formstrace      Oracle Trace V8.1

Event:ApplicatiProduct:oforms  Version:6.0

Epid      Start Timestamp      ApplName      IPAddr      UCPU      SCPU
318      18:17:19.999      trace      130.35.97.  7      23
              105:1823

Epid      End Timestamp      UCPU      SCPU
318      18:17:36.372      20      33
```

For example, note the UCPU Start timestamp and UCPU End timestamp numbers. The start is 7 and the end is 20. To find the total amount of User CPU time for the event, subtract 20 - 7 to receive 13 hundredths of a second.

Using Optional Report Parameters

You can manipulate the output of the Oracle Trace reporting utility by using the following optional report parameters:

- `output_path` Specifies a full output path for the report files. If not specified, the files will be placed in the current directory.
- `-p [<pid>]` Organizes event data by process. If you specify a process ID (`pid`), you will create one file with all the events generated by that process in chronological order. If you omit the process ID, you will create one file for each process that participated in the collection. The output files are named as follows:
- `collection_Ppid.txt`
- `-P` Creates a report called `collection_PROCESS.txt` that lists all processes that participated in the collection. It does not include event data. You could create this report first to determine the specific processes for which you might want to create more detailed reports.
- `-w#` Sets report width, such as `-w132`; the default is 80 characters.
- `-l#` Sets the number of report lines per page; the default is 63 lines per page.
- `-h` Suppresses all event and item report headers, producing a shorter report.
- `-s` Used only with Net8 data (or SQL*Net for Oracle7).
- `-a` Creates a report containing all the events for all products, in the order they occur in the data collection (`.dat`) file.

Refer to the "Oracle® Trace User's Guide" for detailed information.

USING FORMS AND ORACLE TRACE WITH DIAGNOSTICS PACK

If you have the Oracle Diagnostics Pack installed, you can take advantage of the tools provided to manage the Oracle Trace collections remotely with **Oracle Trace Manager** and view the collected data with the **Trace Data Viewer**.

STARTING THE COLLECTION

To use Oracle Trace Manager, you first have to install Oracle Enterprise Manager on a middle tier machine and install the Intelligent agent on the nodes where you want to collect the data.

For more information about how to install OEM, please refer to the Oracle Enterprise Manager Documentation.

From Oracle Trace Manager, you are able to :

- Remotely discover the node where you will run the Data Collection.
- Start the collection on this node using a wizard (Figure 1) asking for the name of the collection and some optional parameters (maximum file size for the trace, time to run the trace, and the like)

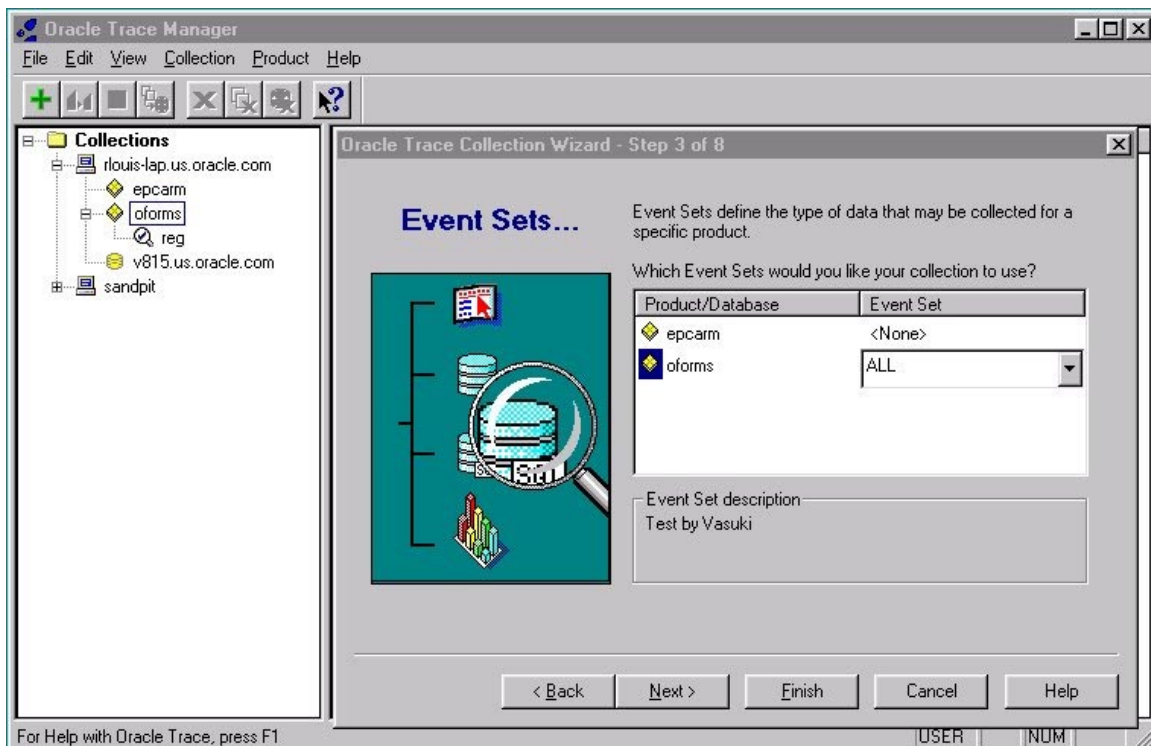


Figure 1 : Oracle Trace Manager Wizard

Once you have started the collection on a node, you can see that the collection is running from Oracle Trace Manager main screen, and you can stop it at any time remotely from the same tool.

Once the Data Collection is running, you can now run your Forms applications in trace mode, as explained in the previous section (adding the pecs=trace parameter on your Forms runtime command line).

FORMATTING THE OUTPUT

The Trace output generated when using Trace Manager is the same as when using Oracle Trace from the command line utilities. Only the way you start the collection differs. This means that after the application has been run in trace mode, the two trace files are generated (cdf and dat files) in the CDF directory as explained in the previous section.

To be able to view the content of the trace with the Trace Data Viewer, the trace output needs to be formatted to a database schema. To do so, using Oracle Trace Manager, right click on the collection you want to upload to the database and choose the format option (Figure 2). This will automatically format the trace to the database in the schema specified in the Trace Manager preferences.

Note : It is also possible to specify to Trace Manager to automatically upload the trace when the collection is stopped.

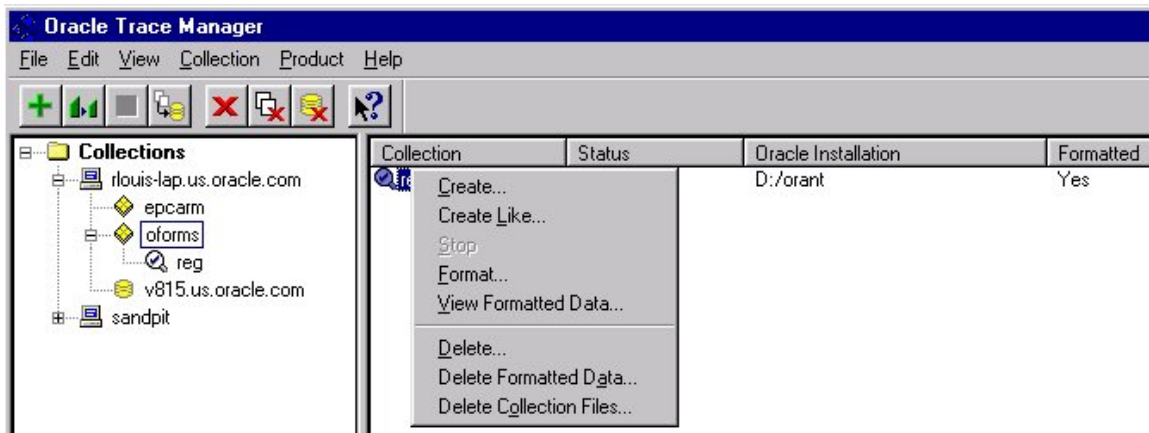


Figure 2 : Format the trace from Oracle Trace Manager

USING TRACE DATA VIEWER

The trace output is now stored in the database within a specific schema. To view and analyze the content of this trace, you launch Trace Data Viewer and connect to the database with the user used to upload the data.

From the Trace Data Viewer, you can analyze:

- The network traffic
- The time spent in the different events (Query, triggers, built-ins, ...)
- The CPU consumption for those events

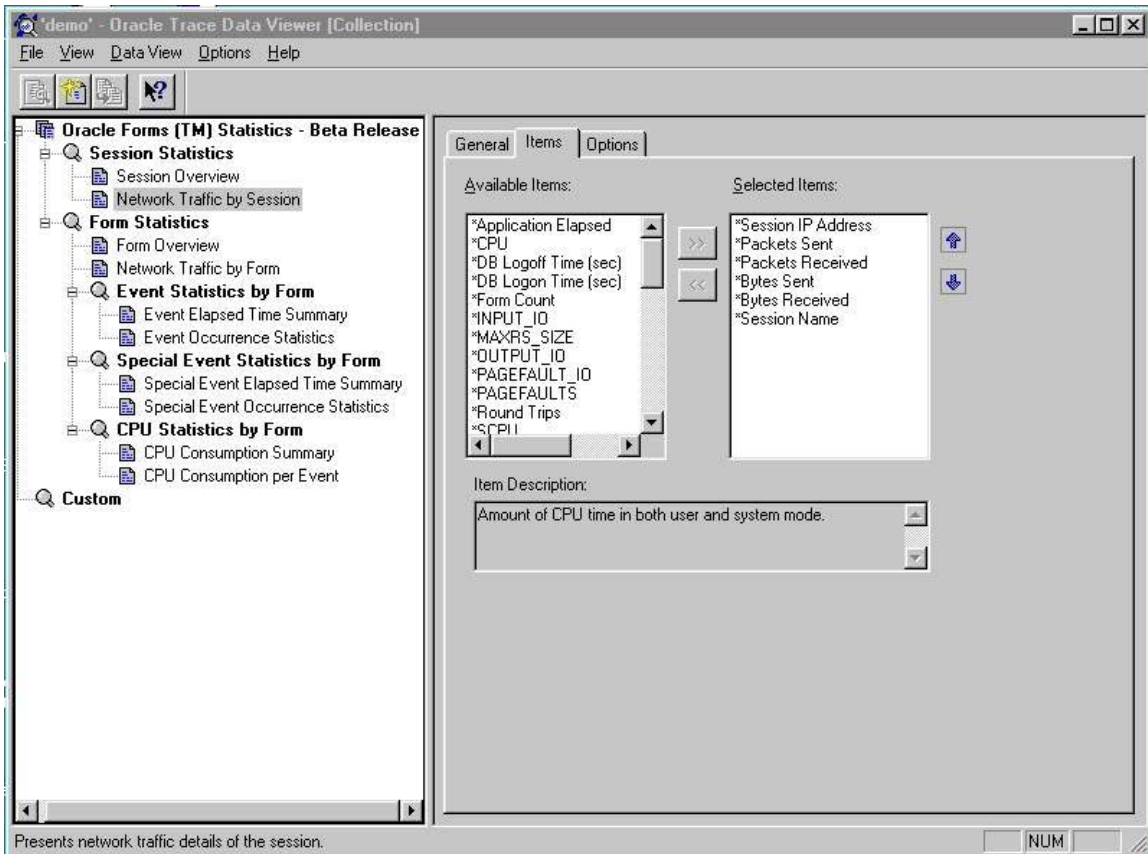
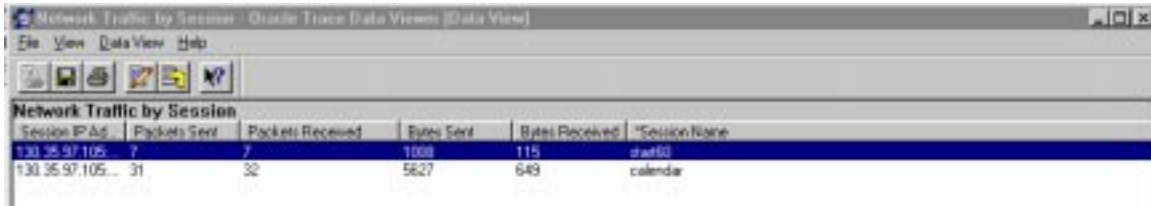


Figure 3 : Trace Data Viewer Main Screen

From the Trace Data Viewer main screen (Figure 3), you can drill down to specific information about Forms Sessions.

For instance :

The Network Traffic Statistics (Figure 4) contain the number of bytes and packets sent or received, the IP address connected, and the number of roundtrips during the session.



The screenshot shows a window titled "Network Traffic by Session - Oracle Trace Data Viewer (Data View)". It contains a table with the following data:

Session IP Ad	Packets Sent	Packets Received	Bytes Sent	Bytes Received	Session Name
130.35.97.105	7	7	1008	115	stat60
130.35.97.105	31	32	5627	649	calenda

Figure 4 : The Network Statistics

You can Drill-down to the level of detail you want (Figure 5), going from an overview to a drill-down analysis of specific events, such as : Server, Triggers, Built-ins, Query, and other types of events.

For each of these events, you have a detailed view of the time spent and the CPU consumption for the events.

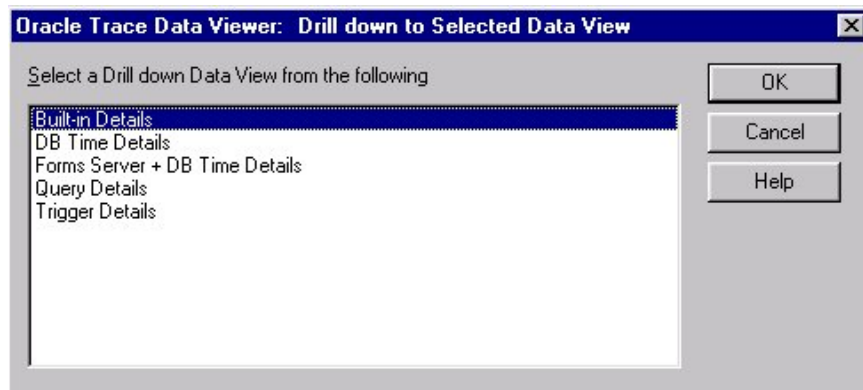


Figure 5 : Drill-Down Choice

When you choose a detailed view, you can sort the information displayed by CPU consumption or time elapsed by clicking on the column header of the column you want to sort by. For instance, this is very helpful to determine which trigger takes the most time to execute or which built-in consumes the most CPU resources (Figure 6).

TriggerName	BlockName	ItemName	*Trigger Elapsed	CPU
WHEN-BUTTON-PRESSED	CONTROL	EXIT_BUTTON	3.715000	3.000000
KEY-LISTVAL	EMP	HIREDATE	0.081000	8.000000
WHEN-BUTTON-PRESSED	EMP	LOV_BUTTON	0.081000	8.000000
WHEN-NEW-FORM-INSTANCE			0.010000	1.000000
WHEN-MOUSE-CLICK	DATE_BUTTON_BLOCK		0.010000	1.000000
WHEN-MOUSE-CLICK	DATE_BUTTON_BLOCK		0.010000	1.000000
WHEN-MOUSE-DOUBLECLICK	DATE_BUTTON_BLOCK		0.010000	1.000000

Figure 6 : Trigger details

Finally, Trace Data Viewer summarizes all the duration events and point events that occurred during the execution of each form with an average and standard deviation of CPU and elapsed time for all the duration events (Figure 7).

STATISTICS FOR ALL OCCURANCES OF THE CURRENT FORM WITHIN THE COLLECTION	
Form Name: START00	
Elapsed time statistics for the Form:	
Form Average:	58.644000
Form Standard Deviation:	0.000000
Form Total:	58.644000
Trigger Average:	2.113148
Trigger Standard Deviation:	3.511067
Trigger Total:	14.792000

Number of Occurance statistics for the Form:	
Number of times form was invoked:	1
Query Count:	0
Trigger Count:	7
LOV Count:	0
Builtin Count:	101
User Exit Count:	0
SQL Count:	0
Menu Count:	1
Block Count:	17
Item Count:	0

Figure 7: Extract from the Trace Data Viewer Summary

SUMMARY

The Oracle Trace integration with Forms *6i* gives you the in-depth ability to trace the performance and the resource consumption of your Forms applications. It provides a way to identify where in the code a potential performance problem may occur and shows you the areas where your application may be optimized.

There are two methods to manage the collection and analyze the results :

- Using the Oracle Trace command line utilities shipped with Oracle Forms
- Using the Oracle Trace tools part of Oracle Diagnostics Pack

For more information on diagnosing the performance and tuning your Forms applications, please visit the Oracle Technology Network web site :

<http://technet.oracle.com/products/developer/>

RELATED DOCUMENTS

Please refer to the following document for more information:

“Oracle® Enterprise Manager - Oracle® Trace User’s Guide,” Part Number A67837-01.



Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

Worldwide Inquiries:
+1.650.506.7000
Fax +1.650.506.7200
<http://www.oracle.com/>

Copyright © Oracle Corporation 2000
All Rights Reserved

This document is provided for informational purposes only, and the information herein is subject to change without notice. Please report any errors herein to Oracle Corporation. Oracle Corporation does not provide any warranties covering and specifically disclaims any liability in connection with this document.

Oracle is a registered trademark, and Oracle8i, Oracle8, PL/SQL, and Oracle Expert are trademarks of Oracle Corporation. All other company and product names mentioned are used for identification purposes only and may be trademarks of their respective owners.
