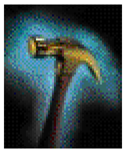


Performance Collector for Oracle® Forms 6*i*

An Oracle® Technical White Paper

April 2001

INTRODUCTION

An Oracle® Forms application may require frequent user interaction and thus *response time* plays a major role in determining the productivity of the Forms user. The 3-tiers of the Forms Server architecture - the Forms Client, the Forms Server and the database server - are well designed to provide minimum response time to the end user, however, due to the flexibility supported by Forms in terms of user exits, libraries and program units the performance of a Forms application depends in part on the Forms designer. The designer needs a tool or a technique to deliver an application without performance flaws. Performance Collector, which appears as part of Oracle Forms release 6*i*, is one of the features designed to support the application developer or performance analyst in determining the time spent by a Forms application in each tier of the 3-tier architecture.

BACKGROUND

A typical network infrastructure for using the Forms application is shown in Figure 1. In this, each user action from the Forms Client is communicated to the Forms Server. The Forms Server processes the user request and communicates the response back to the Forms Client. While processing the user requests, the Forms Server might also communicate with the database server.

In further discussions in this paper, the duration for which a Forms user is using the Forms application is termed as a session. A typical user session comprises a sequence of round trips between the Forms Client and the Forms Server and between the Forms Server and the database server as Figure 1 illustrates.

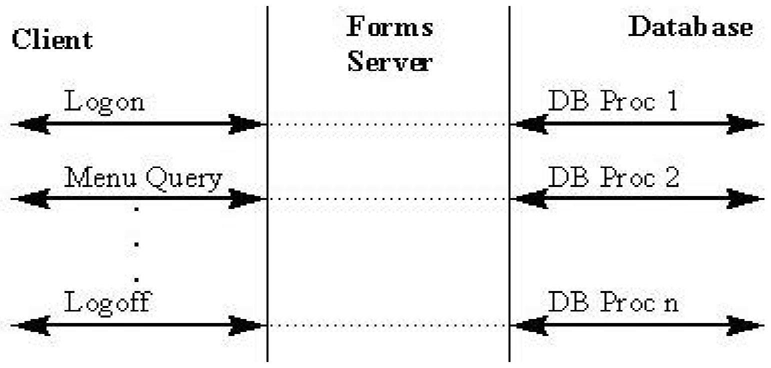


Figure 1 : A Typical User Session with Forms Server

Figure 2 illustrates a typical scenario. (The continuous lines indicate the communication between the Forms Client and the Forms Server at the start and end of the user action. The dotted lines indicate the communication between the Forms Server and the database server. The smaller dotted lines between the Forms Client and Forms Server indicate the roundtrips between the Forms Client and Forms Server, during the user action processing which is not explicitly known to the end user).

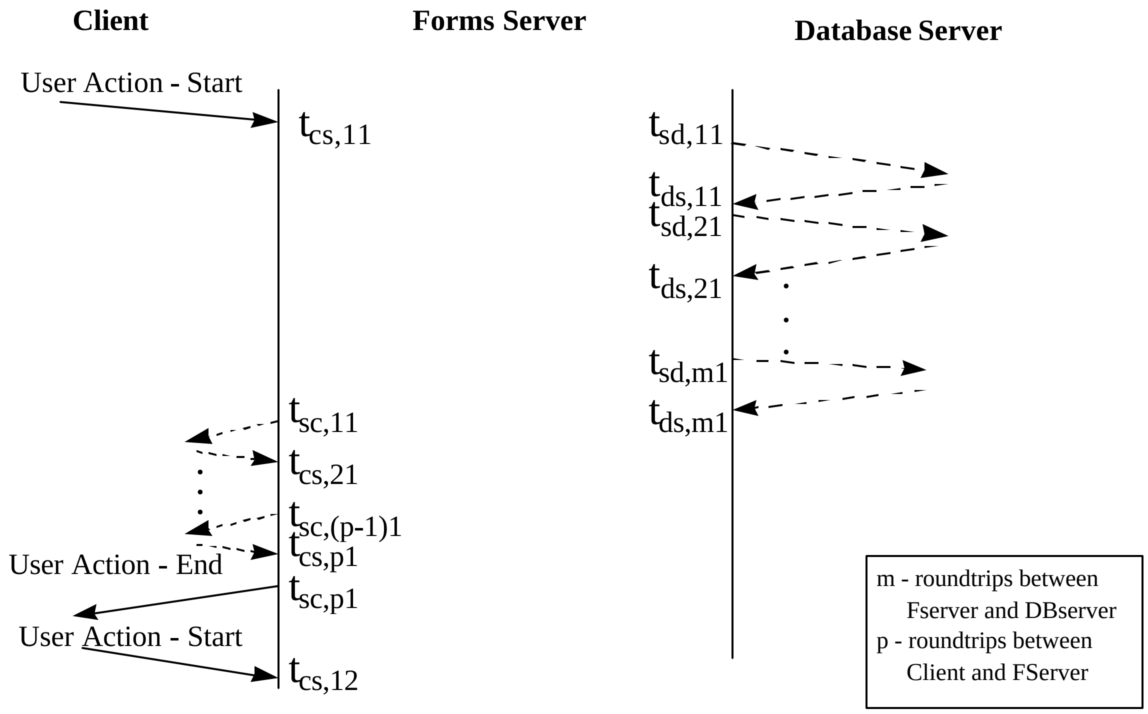


Figure 2 : Communication due to a User Action

$t_{cs,11}$	Instant at which the Forms Server receives the user action request from the Forms Client for UserAction 1
$t_{cs,p1}$	Instant at which the Forms Server receives the last packet from Forms Client during roundtrips for UserAction 1
$t_{sd,i1}$	Instant at which the Forms Server sends request to the database server i = 1,…m for UserAction 1
$t_{ds,i1}$	Instant at which the Forms Server receives the response from the database server i = 1,…m for UserAction 1

$t_{sc,k1}$	Instant at which the Form Server sends roundtrip packets to the Forms Client k = 1,…p. for UserAction 1
$t_{sc,p1}$	Instant at which the Form Server sends last response packet to the Forms Client for UserAction 1
$T_{cs,12}$	Instant at which the Form Server receives the NEXT User Action Request from the Forms Client for UserAction 2

The illustration also helps in deriving the time spent during various parts of the transaction as described below:

$t_{cs,12} - t_{sc,p1}$ = Forms Client Processing Time + User Think Time (for User action 2) + Network latency

$t_{cs,i1} - t_{sc,(i-1)1}$ = Forms Client Processing Time + Network latency , i = 2,…p

$t_{ds,k1} - t_{sd,k1}$ = Processing Time at the database server + Network Latency k = 1,…m

Following the same approach, in User Action 1:

$\sum_{i=2,\ldots p}(t_{cs,i\,1} - t_{sc,(i-1)1})$ = Forms Client Processing Time + p * Network latency

$\sum_{i\,=\,1\ldots p}(t_{sc,i1} - t_{cs,i1}) - \sum_{k=0,1\ldots k}(t_{ds,k1} - t_{sd,k1})$ = Processing Time at the Forms Server

$\sum_{k=1\ldots m}(t_{ds,k\,1} - t_{sd,k1})$ = Processing Time at the database server + m * Network Latency

where:

m = no. of roundtrips between the Forms Server and database server
p = no. of roundtrips between the Forms Server and Forms Client.

Forms Performance Collector uses the above approach to determine the timing components at the 3-tiers as illustrated further in this paper.

FORMS PERFORMANCE COLLECTOR

Once enabled, the Performance Collector records time stamps in the log file whenever the Forms application's thread of execution enters or exits the Forms Server. The following are all of the possible time stamp labels that can be recorded in the log file:

Recorded as	Description
FSERVER_START	Instant at which the Forms Server receives the Forms Client request
FSERVER_END	Instant at which the Forms Server sends a response to the Client
DBLOGON_START	Instant at which the Forms Server sends a logon request to the database server.
DBLOGON_END	Instant at which the Forms Server receives the logon response from the database server.
DB_START	Instant at which the Forms Server sends a query or other request to the database server
DB_END	Instant at which the Forms Server receives a query response from the database server
DBLOGOFF_START	Instant at which the Forms Server sends a logoff request to the database server
DBLOGOFF_END	Instant at which the Forms Server receives the logoff response from the database server

Table 1. Forms Server Time Stamps

Since time stamps are only recorded when the thread of execution for a Forms application either enters or exits the Forms Server, the difference between two successive time stamp points represents time spent in one of the 3-tiers. The interval of time spent in one tier is called a Within-Tier Event. Within-Tier Events are grouped into 4 categories - Forms Server, Database, Network, and Client. **Note** that Network and Client categories are actually the two subdivisions that make up the Forms Client tier. The Network category does **not** include time spent on the network communicating between the Forms Server and the database server.

Each Within-Tier Event is uniquely defined by an ordered pair of time stamps - the initial time stamp and the final time stamp. For example, a **FSPrelogon** Within-Tier Event starts with a **FSERVER_START** time stamp and ends with a **DBLOGON_START** time stamp. This indicates that the thread of execution entered the Forms Server (in response to a user action on the Forms Client), the Forms Server processed the request, then the thread of execution exited the Forms Server and entered the database server in order to perform a database logon operation. To be considered a **FSPrelogon** Within-Tier Event, **DBLOGON_START** must be the first time stamp recorded after the **FSERVER_START** time stamp.

1. Forms Server Within-Tier Events

FSPreLogon FSERVER_START ® DBLOGON_START

Time spent in the Forms Server before a logon request is sent to the database server.

FSPostLogon DBLOGON_END ® FSERVER_END

Time spent in the Forms Server after the response to a logon request is received from the database server.

FSPreDB FSERVER_START ® DB_START

Time spent in the Forms Server after processing a Forms Client request and before a query is sent to the database server.

FSMidDB DB_END ® DB_START

Time spent in the Forms Server between successive queries to the database server. Arises when query results are not deliverable in a single round trip between the Forms Server and the database server.

FSPostDB DB_END ® FSERVER_END

Time spent in the Forms Server after a query is sent to the database server and before responding to the Forms Client.

PreLogoff FSERVER_START ® DBLOGOFF_START

Time spent in the Forms Server before a logoff request is sent to the database server.

PostLogoff DBLOGOFF_END ® FSERVER_END

Time spent in the Forms Server after the response to a logoff request is received from the database server.

2. Database Within-Tier Events

DBLogon DBLOGON_START ® DBLOGON_END

Time spent in the database server and on the network processing a logon request.

DBLogoff DBLOGOFF_START ® DBLOGOFF_END

Time spent in the database server and on the network processing a logoff request.

DBProc DB_START ® DB_END

Time spent in the database server and on the network processing a database query.

3. Network Within-Tier Events

Network FSERVER_END ® FSERVER_START *

The duration between when the Forms Server sent a response to the Forms Client and the next request was received from the Forms Client **minus** time spent either executing the Forms Client or waiting for user input to the Forms Client.

4. Client Within-Tier Events

Network FSERVER_END ® FSERVER_START *

Time spent executing the Forms Client or waiting for the user input to the Forms Client. Does **not** include time spent on the network communicating between Forms Client and Forms Server.

The start and end timestamp point names, the Forms Server Event names and the Group names are further used in this paper in discussing the reporting and output analysis of the performance data.

USING THE FORMS PERFORMANCE COLLECTOR

The Performance Collector is built with the *Forms Runtime Diagnostic* (FRD) logging facility. The Performance Collector collects time stamps at predefined points as previously explained and records them in an ASCII output file. A PERL script is supplied to parse and report the data meaningfully into HTML and optionally into a Comma Separated Value (CSV) file.

With understanding of the 3-tier architecture, Performance Collector's breakdown of data into 4 categories (Forms Server, Database, Network, and Client), and the definitions of the Within-Tier Events, discussed in the previous section, the data can be easily processed using any spreadsheet application.

Using Performance Collector requires three steps :

1. Enabling performance collection
2. Reporting the data for analysis
3. Analyzing the output

ENABLING PERFORMANCE COLLECTION

Performance Collector is enabled by the command line option `record=performance` on the Forms Client. Alternately, Performance Collector can be started by specifying `record=all` to have both Performance Collector and Forms Runtime Diagnostics (FRD) data written to the log file. The Performance Collector PERL script will extract only the Performance Collector data from the resulting log file in either case.

Forms 3-tier Modes (Sockets, HTTP, Listener Servlet)

Include `record=performance` as part of the `serverArgs` parameter in the HTML file:

```
<param name="serverArgs" value = "module=put_module_name_here
userid=put_userid_here record=performance
log=put_file_name_here">
```

or include `record=all` as part of the `serverArgs` parameter:

```
<param name="serverArgs" value = "module=put_module_name_here
userid=put_userid_here record=all log=put_file_name_here">
```

The user actions and the timestamp details are written into the log file denoted by `put_file_name_here.` If the log file name is not specified then a log file is created with a unique name of the format `perf_xxxx` where `xxxx` is the process id of the Forms Server process.

Forms Client/Server Mode

When running Performance Collector in Client/Server mode only the raw time stamp data will be available. The PERL script is only supported for 3-tier modes. Since there is only the Forms Client in Client/Server mode (Server referring to the database server) **FSERVER_START** and **FSERVER_END** time stamps will not be recorded.

REPORTING THE DATA FOR ANALYSIS

The data collected by the Performance Collector is processed using a PERL script to produce meaningful reports for analysis. Perl 5 (or a newer version), available on the web for free download, can be used to run the PERL script. The PERL script accepts the recorded log file, the optional output CSV file name, and the OS (operating system) on which the Forms Server was running when the performance data was collected. The output includes HTML files and optionally a CSV file, viewable with a spreadsheet application.

To analyze the data :

- Locate the PERL script `f60parse.pl` in the following directory :
  ```
  <Oracle_Home>\forms60\Perl
  ```
- Run the Perl script: using the following command :
  ```
  perl f60parse.pl -input=infile [-output=ofile] [-os=n|u]
  ```
  where:

  `infile` = Data log file recorded while running the application

  `ofile` = Optional CSV file

`os` = operating system, can be either **n** or **u**

By default, the results are written in HTML format to six different files - index.html, summary.html, runtime.html, detailed.html, detailed2.html, event.html, glossary.html Index.html is the first HTML page you should look at and will contain links to the five other HTML files.

ANALYZING THE OUTPUT

Log File Output

The raw data output is contained

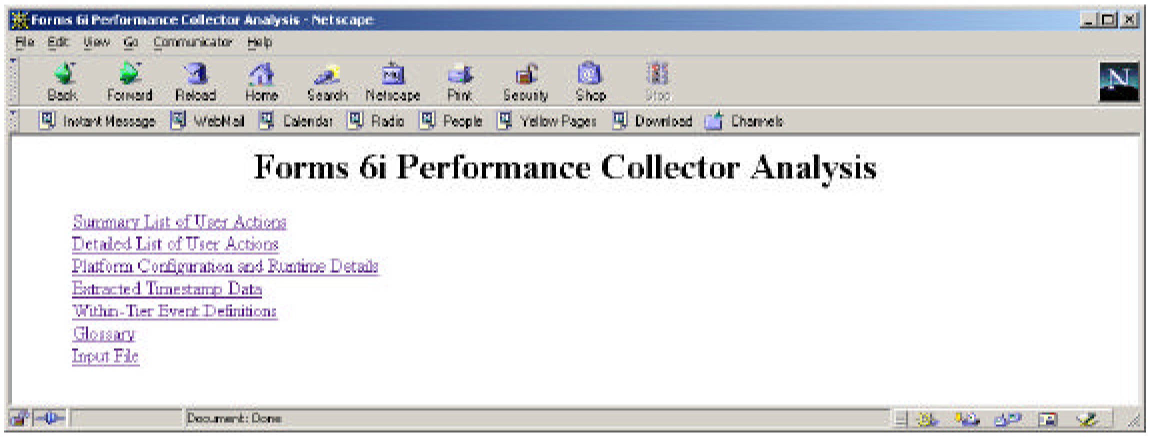
```
##### CTIME STARTS HERE
# 2 - MODULE1:EMP.EMPNO.166935470
MENU DEFAULT Query eXecute

TSE FSERVER_START -1 4577 166935470
TSE DB_START 0 0 166935480
TSE DB_END 0 0  166935480
TSE DB_START 0 0 166935480
TSE DB_END 0 0  166935480
TSE FSERVER_END -1  0 166935530
##### CTIME STARTS HERE
# 3 - MODULE1:EMP.EMPNO.166936902
MENU DEFAULT Record Next

TSE FSERVER_START -1 1342 166936902
TSE DB_START 0 0 166936902
TSE DB_END 0 0  166936902
TSE DB_START 0 0 166936902
TSE DB_END 0 0  166936902
TSE FSERVER_END -1  0 166936902
```

HTML Output

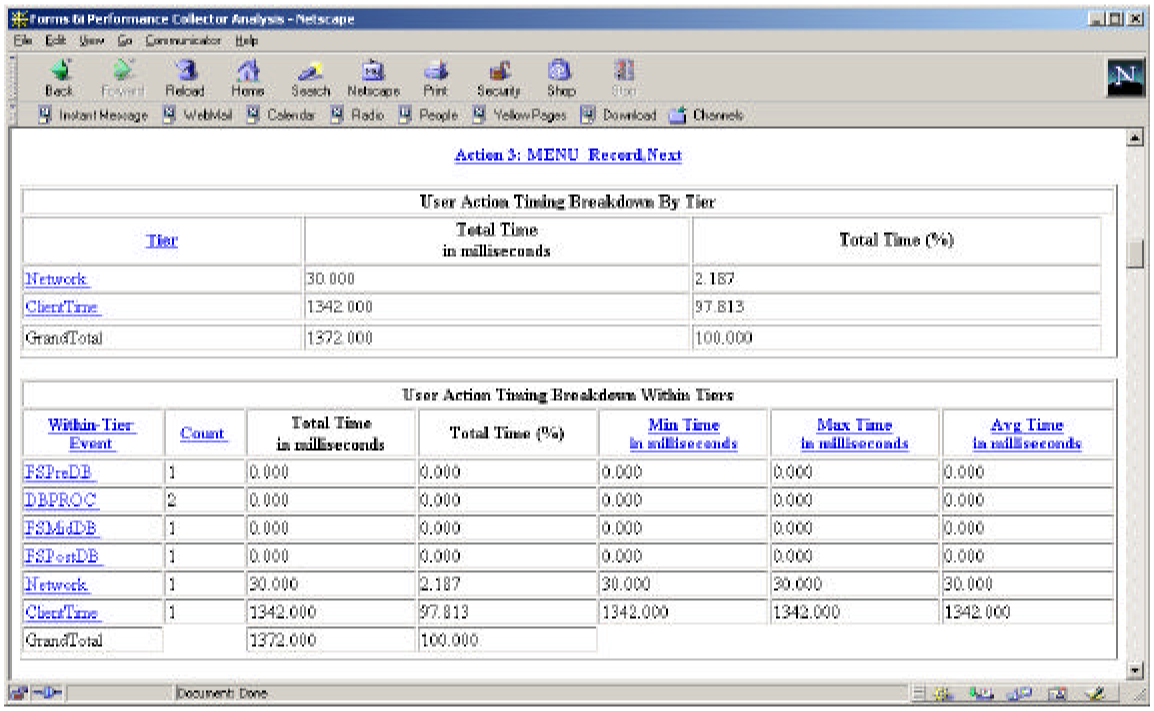
The HTML report allows the performance data to be viewed with any standard web browser. Index.html is the starting point and the table of contents for the HTML presentation. Most terms used in the HTML pages are hyperlinked to glossary.html which contains a list of useful definitions.


Forms 6i Performance Collector Analysis - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Shop Stop
Instant Message WebMail Calendar Radio People Yellow Pages Download Channels
Document: Done

Forms 6i Performance Collector Analysis

Summary List of User Actions
Detailed List of User Actions
Platform Configuration and Runtime Details
Extracted Timestamp Data
Within-Tier Event Definitions
Glossary
Input File

Figure 3 : Index.html


Forms 6i Performance Collector Analysis - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Shop Stop
Instant Message WebMail Calendar Radio People Yellow Pages Download Channels
Document: Done

Action 3: MENU Record.Next

User Action Timing Breakdown By Tier

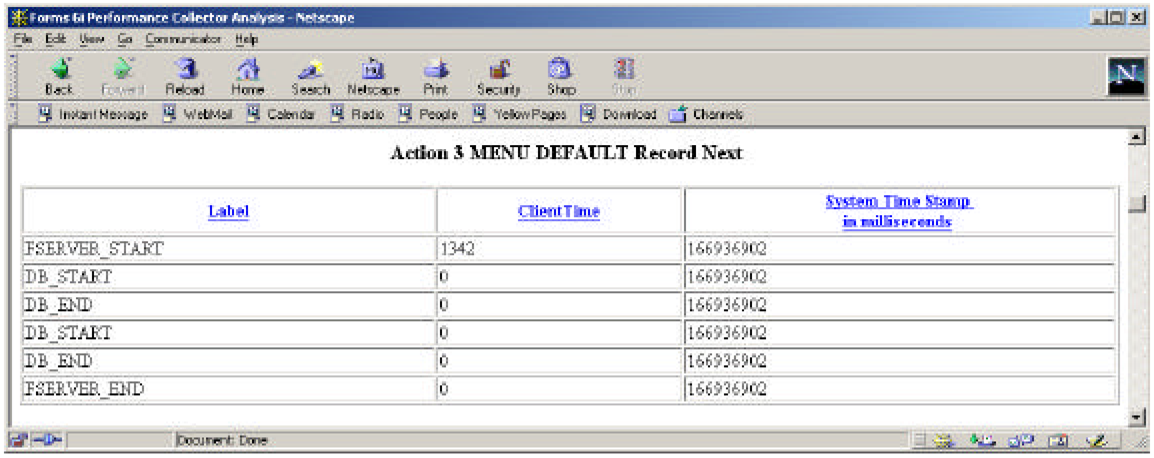
Tier	Total Time in milliseconds	Total Time (%)
Network	30.000	2.187
ClientTime	1342.000	97.813
GrandTotal	1372.000	100.000

User Action Timing Breakdown Within Tiers

Within-Tier Event	Count	Total Time in milliseconds	Total Time (%)	Min Time in milliseconds	Max Time in milliseconds	Avg Time in milliseconds
FSPreDB	1	0.000	0.000	0.000	0.000	0.000
DBPROC	2	0.000	0.000	0.000	0.000	0.000
FSMidDB	1	0.000	0.000	0.000	0.000	0.000
FSPostDB	1	0.000	0.000	0.000	0.000	0.000
Network	1	30.000	2.187	30.000	30.000	30.000
ClientTime	1	1342.000	97.813	1342.000	1342.000	1342.000
GrandTotal		1372.000	100.000			

Figure 4 : Detailed.html

Detailed.html presents two views of every user action - time spent in each tier and time spent in each Within-Tier Event. Statistics like the minimum, maximum, and average time per Within-Tier Event per user action are also displayed.


Forms 6i Performance Collector Analysis - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Shop Stop
Instant Message WebMail Calendar Radio People Yellow Pages Download Channels
Document: Done

Action 3 MENU DEFAULT Record Next

Label	Client Time	System Time Stamp in milliseconds
FSERVER_START	1342	166936902
DB_START	0	166936902
DB_END	0	166936902
DB_START	0	166936902
DB_END	0	166936902
FSERVER_END	0	166936902

Figure 5 : Detailed2.html

Detailed2.html presents the raw time stamp data that was used to create the Tier-By-Tier and Within-Tier views of detailed.html.

CSV Output

The optional output file is a Comma Separated Value (CSV) file that can be loaded in any standard spreadsheet application. This comprises of three parts:

- List of User Actions
- Tier-by-Tier Summary per User Action
- Within-Tier Event per User Action

SUMMARY

With Oracle Forms release 6i, you now have an option when troubleshooting and improving performance. Evaluating performance of Oracle Forms is essential in order to ensure that the software provides the minimum response time in the given operating environment and maximizes resource utilization. The Forms Performance Collector helps you to determine where to focus your performance tuning.

OTHER USEFUL REFERENCES

1. “An overview of Oracle Forms Server Architecture,” Technical White Paper.

2. “How to Tune your Oracle Forms Server Applications,” Technical White Paper.

3. “Integration of Oracle Forms with Oracle Trace,” Technical White Paper.