



PERFORMANCE
TEST SOLUTIONS
FOR ORACLE

Forms2test

Release 6

User's Guide

> Table of Contents

Preface	3
How Forms2test works	3
Preparing to use Forms2test	4
Requirements	4
Installation	4
Running Forms2test	5
Configuring Forms2test	5
The testing process	6
Configuration test	7
Load / stress test	7
Creating a script	8
Start recording	8
Defining transactions	10
Defining think time	12
Defining an iteration	13
Ending and saving a recording session	14
Parameterize a script	15
Script Editor	16
Parameter Name	18
Parameter Type	18
Parameter Properties	19
Running a test	21
Script schedule	21
Test Goal	24
Fail on Forms messages	26
Analyzing the results	30
Troubleshooting	33
What it means	36

> Preface

Welcome to the TestNext Forms2test Start Guide. This guide provides a step-by-step overview to using TestNext Forms2test. Forms2test performance tests an Oracle Forms environment by simulating users. During user simulation a lot of metrics, such as response times, data throughput, connections, network requests, etc., are captured and logged. At the end of a test the results are presented in various automatically generated graphical reports; the starting point for analysis.

> How Forms2test works

The Oracle Forms client, JInitiator (a Java applet), communicates with the Forms runtime, which runs on the Oracle Application Server, over HTTP. HTTP is a request/response protocol; a widely used and firewall friendly protocol for transferring data over a network. Each user action in a Forms application, for instance submitting a button or selecting a table row, results in network traffic over HTTP. This traffic notifies the Forms runtime on the application server of this event. This way the Forms runtime and the Forms client are always in sync.

Forms2test works on a record-and-playback principle. As you work with the Forms application under test the network traffic that the user generates are intercepted and recorded into a script (see figure 1). Including think and response times. With Forms2test you can create load on the application server by playing back the script a number of times concurrently to simulate real user sessions (see figure 2).

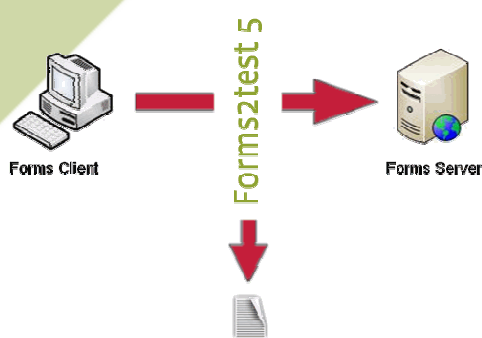


Figure 1: Record a script

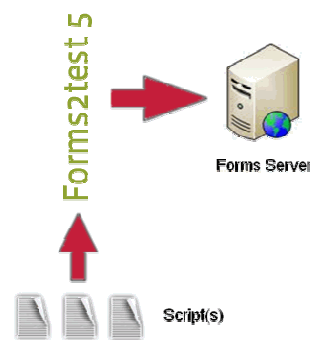


Figure 2: Playback script(s)

To intercept the network traffic Forms2test places itself between the browser and the Oracle Application Server. Forms2test acts like a local proxy. The client connects with Forms2test and Forms2test connects to the Oracle Application Server.

When you start recording a user session you will notice that the Forms application link points to a local endpoint (<http://localhost:3773/forms/...>) instead of the application link.

Even if the Forms application runs over a secure connection (https) the protocol of the local (Forms2test) endpoint remains http.

➤ Preparing to use Forms2test

Requirements

Forms2test requires your computing environment meets some minimum requirements. Forms2test is a 100% Java application and requires a fully compliant JVM 1.6 or higher.

The Java runtime (JRE) is free to download from the site of Sun:

<http://www.oracle.com/technetwork/java/javase/downloads/index.html>

Forms2test has been tested and should run correctly under Windows (NT, 2000, XP, Vista and Windows 7).

Tip: *The ephemeral port range for windows machines is limited to 5000. For performance testing this range is often too restrictive and can result in the error: "address already in use" during running a test scenario. You should read the following article from Microsoft carefully:Q196271*. You are strongly advised to increase the TCP/IP registry MaxUserPort setting as described in the article and to set the registry setting TCPTimedWaitDelay to 60 seconds. Do not forget to restart Windows for the changes to take effect!*

Installation

Installing Forms2test is a snap. Forms2test is available as a Windows installer. Launch the installer and install the full functional version into the directory where you want Forms2test to be installed.

Please note that installing Forms2test *will overwrite* your existing installation. However, settings and registrations won't be lost.

* <http://support.microsoft.com/default.aspx?scid=kb;en-us;196271>)

Running Forms2test

After the installation has completed the installer has created start menu items to launch (or uninstall) Forms2test.

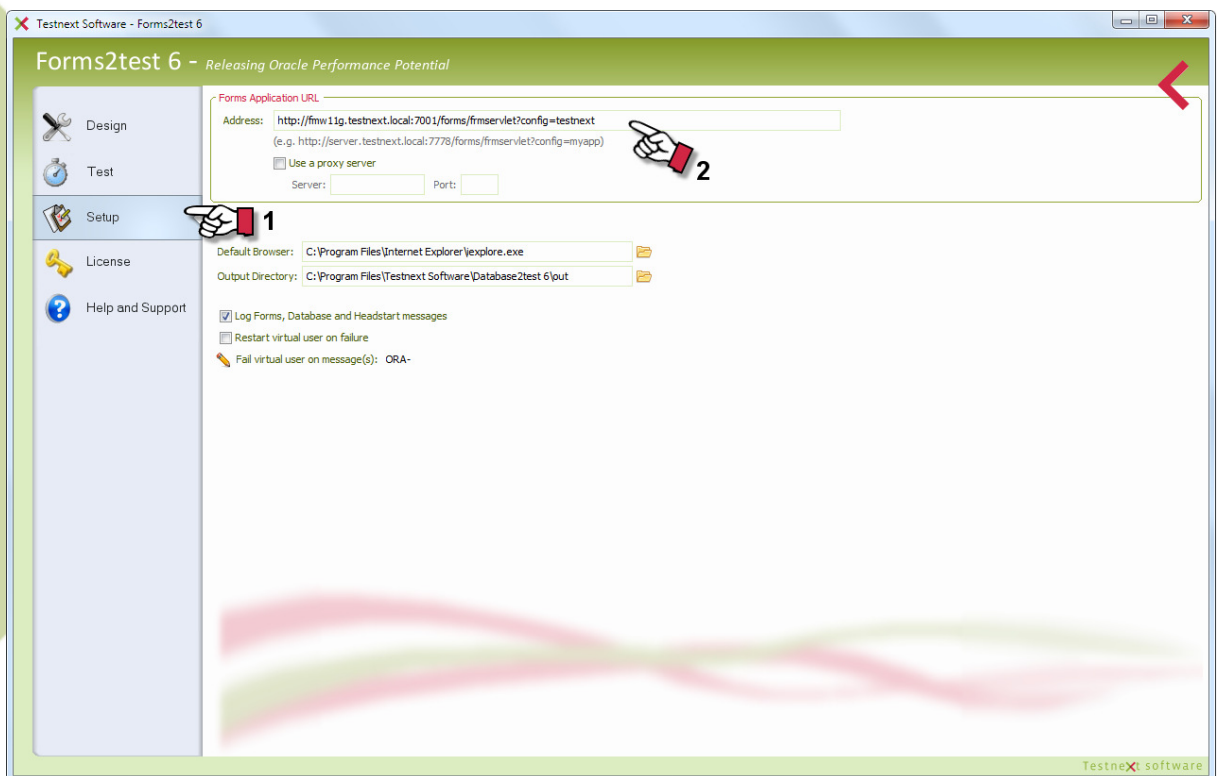
Note: If the start-up fails, ensure that the `JAVA_HOME` variable points to the location where you have installed Java 5. This variable is set in the `setenv.bat` shell script.

Configuring Forms2test

If Forms2test is started for the first time you are automatically directed to the Settings tab (1).

The only mandatory configuration setting for Forms2test is the URL of the Oracle Forms application (2).

If you are testing from behind a firewall/proxy server, you may need to provide Forms2test with the firewall/proxy server hostname and port number.



> The testing process

A performance test is a multi step process and consists of the following basic tests:

- Configuration test
- Load test
- Stress test (*optional*)

The objective of a configuration test is to verify that all elements that make up the Oracle Forms environment are properly and well scaled configured. The configurable elements are the hardware resources, the operating systems of the Oracle application server and the Oracle database server, the application server, including the Oracle HTTP Server (OHS), the Oracle container and Forms Services, and the Oracle database including the database listener.

For instance, insufficient physical memory or desktop heap (Windows) limits the maximum number of concurrent users, or the web server can handle a too limited number of concurrent connections or the database listener can spawn only a too limited number of dedicated database sessions.

These kind of scalability problems will without a performance test occur when the Forms application goes live.

The difference between a load and stress test is the objective of the test. For a stress test you are interested in the maximum load and/or concurrent users the Oracle Forms environment can handle. For a load test you are interested in the performance and resource usage of the Oracle Forms environment during typical production load. Form2test is suitable for both tests.

Although the stress test is an optional test it does provide some useful input for capacity planning.

Configuration test

For conducting a configuration test only a single script for simulating an 'inactive' user session is needed.

The user session starts the Forms application, then logs into the database stays idle for some time before closing the Oracle Forms application again.

The next step is to schedule a scenario setting the load behavior to start 1 'inactive' user session every 5 seconds gradually and start the scenario.

Example: A configuration test for 300 user sessions requires an idle time for at least 25 minutes ($= 300 * 5 / 60$). The duration of the configuration test will take 50 minutes. 25 minutes for the ramp up and 25 minutes for the ramp down.

Tip: During the configuration test you should run the Performance Monitor (Windows), nmon (AIX or Linux) or a comparable tool to collect the performance statistics for the application server. Especially, disk, memory and CPU (kernel and user) usage are very important counters.

If the test fails you should examine the log files to determine the cause of the failed user sessions. If the test passes you should examine the client side performance statistics and the server side statistics (see Analyzing the results).

Load / stress test

The first step for a load test is to create one or more test scenarios or test cases. A test scenario defines the typical working conditions for the Forms application under test. A test scenario defines the number of users to simulate, the typical business processes they perform and the series of steps to collect individual performance statistics for. For instance opening a form or steps that comprise a complex database operation.

These user actions are often based on predefined acceptance criteria, for instance the form/module should appear within 3 seconds.

The creation process for a stress test is similar to a load test. The test scenario is often based on the same set of scripts only the number of users to simulate and the load differs. A stress test consists out of more test scenarios. Each with an increasing load and number of users to simulate.

> Creating a script

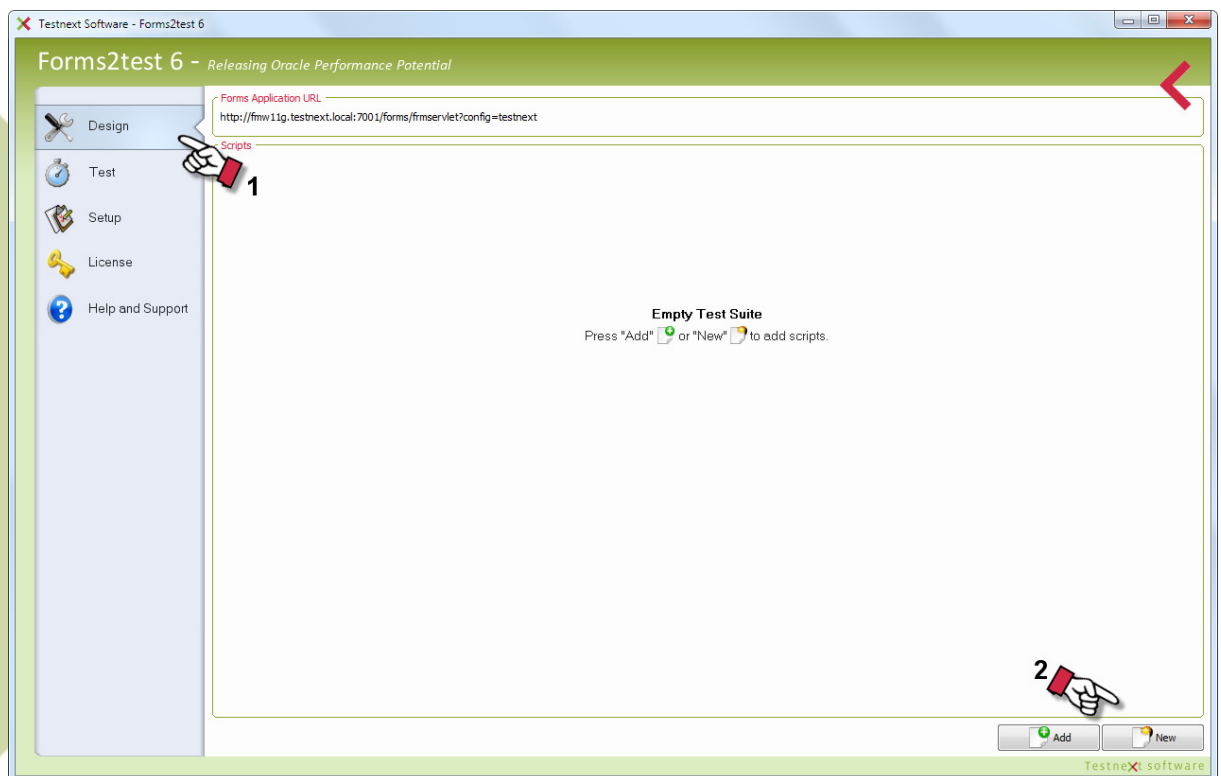
In this section you will learn how to create a script.

Make sure that the Forms application link points to the correct location (see *Preparing to use Forms2test*)

Start recording

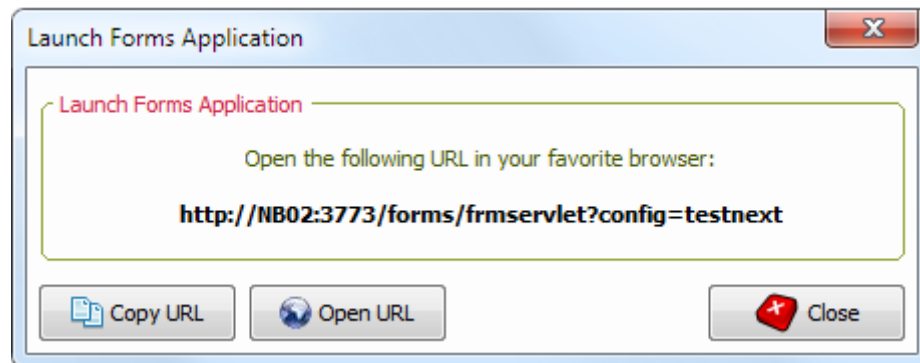
Choose Start > Programs > Forms2test 4.1 > Forms2test 4.1 to launch Forms2test.

Select the Design page (1) and click New (2) at the bottom right.



Two dialog boxes open, the “Open browser” dialog and the recording control panel.

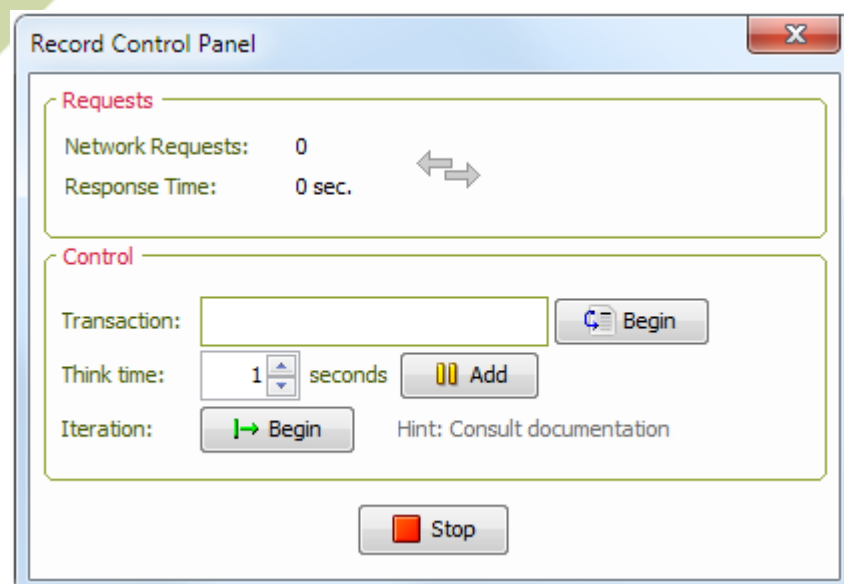
The “Open URL” dialog box shows the link for accessing the Oracle Forms application. Note that this link differs from the original link. The link refers to the local listening port of Forms2test instead of the application server (see *How Forms2test works*).



You can copy the link to the clipboard and paste the link into your favorite browser clicking Copy or you can open the link directly in the Internet Explorer clicking Open. Note that even when the Forms application runs over https, the Forms2test link uses the unsecure http protocol.

Either way, the dialog box will disappear and Forms2test starts recording when the web browser opens.

The recording control panel shows some network statistics (total number of network requests send to the server and the cumulative response time) and the controls to define a transaction and add think times.



Although the recording session will automatically close when you exit your Forms application you can always manually stop/cancel the recording session by pressing the Stop button.

Defining transactions

During recording a user session you can define specific series of application steps within a business process of interest. For instance logging on to the database, opening a form or processing an order.

With Forms2test you can designate a series of user steps you want to measure under load by defining them as a logical transaction.

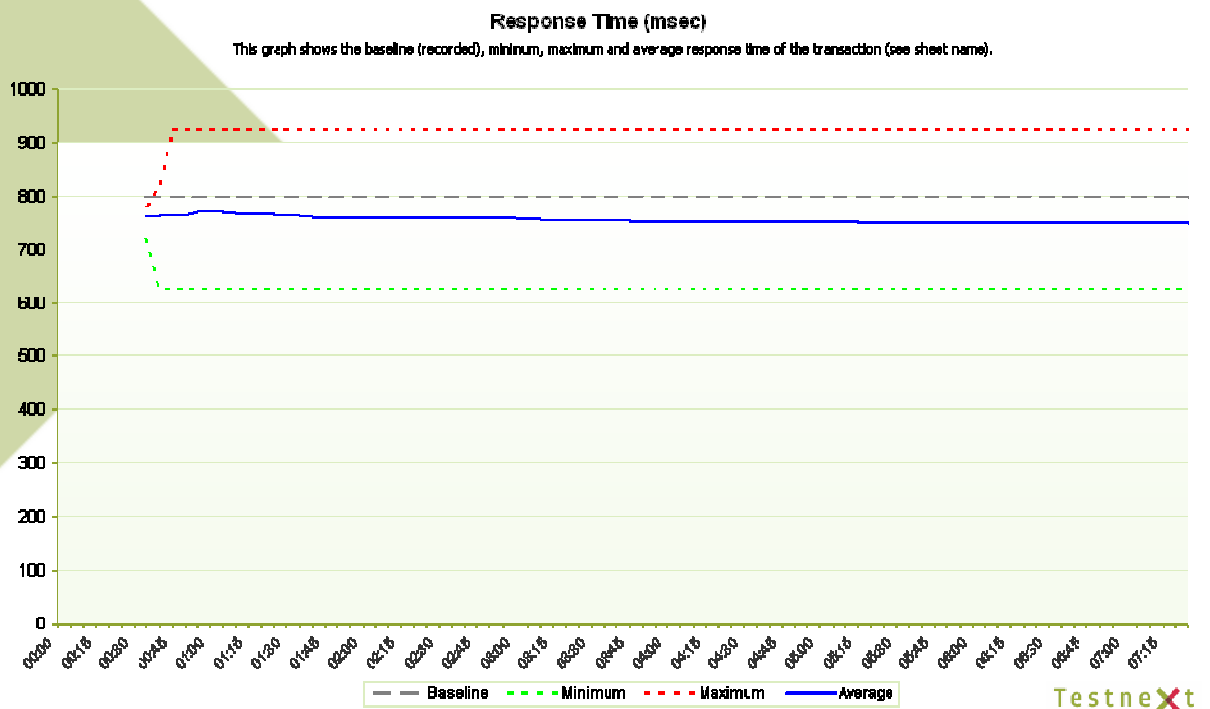
When playing back the user session Forms2test will capture the following performance statistics for each transaction:

- Average response time of the transaction in milliseconds,
- Minimum response time of the transaction in milliseconds, and the
- Maximum response time of the transaction in milliseconds.

For each transaction Forms2test will automatically generate a graphical report based on the statistics mentioned.

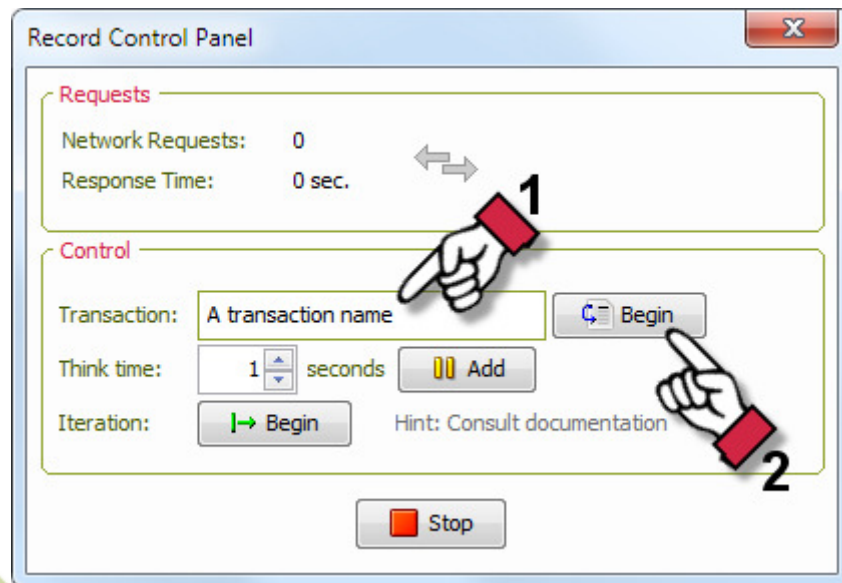
For example, the transaction report below shows that the minimum response time of the transaction during the test was 1400 milliseconds. Which is equal to the baseline response time. And the maximum response time was 2000 milliseconds.

The average response time of the transaction was about 1500 milliseconds.

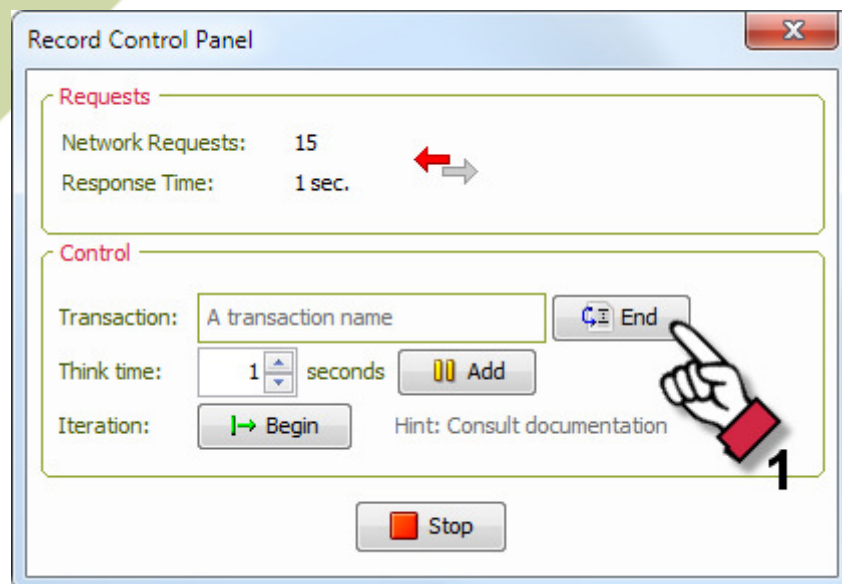


Transactions are often based on predefined acceptance criteria (SLA). You can define as many transaction as needed.

Defining the beginning of a transaction is simple. While recording a user session, (1) enter the name of the transaction in the record control panel and (2) click “Begin” to define the beginning of the transaction.



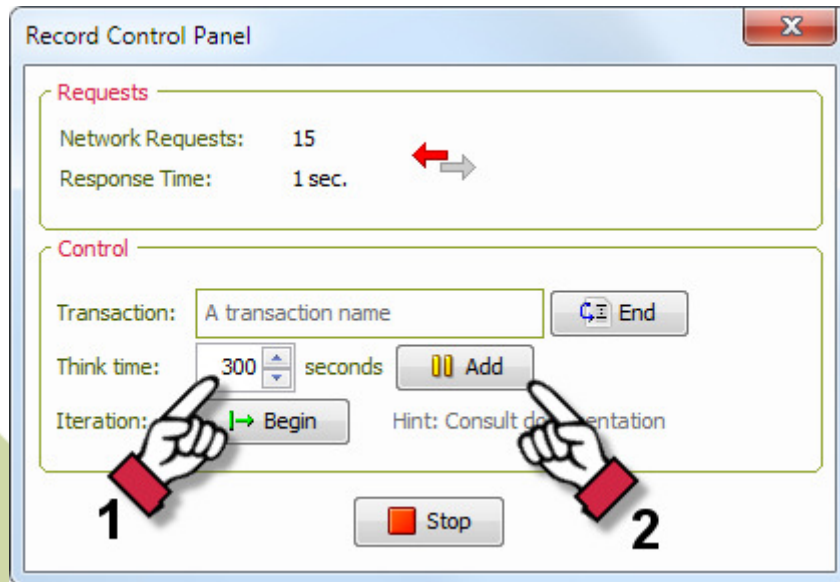
Then, continue with the Forms application till the end of the transaction. To define the end of the transaction, navigate to the record control panel and click “End” to define the end of the transaction.



Defining think time

During recording a user session you can define points to suspend execution. Also known as “think time”. When playing back the user session Forms2test will wait for the given number of seconds at the defined think time before continuing.

To define think times, enter the number of seconds to cease execution (1) and click Add (2) in the recording control panel.



After recording the script you can adjust the think time with the Script Editor. It's also possible to define random think time within a predefined range with the Script Editor.

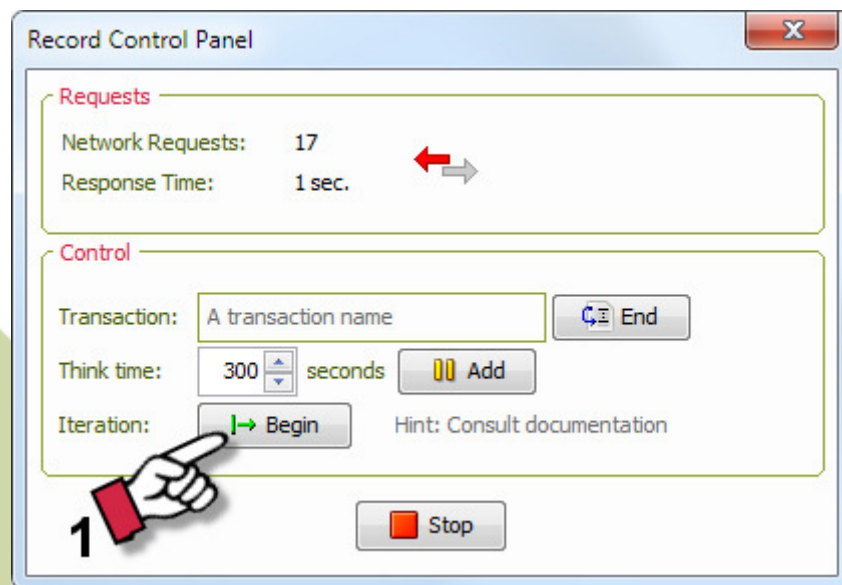
Defining an iteration

An iteration is a mechanism to iterate over a series of session steps by marking them as a single logical step. For instance entering a claim or altering a customer.

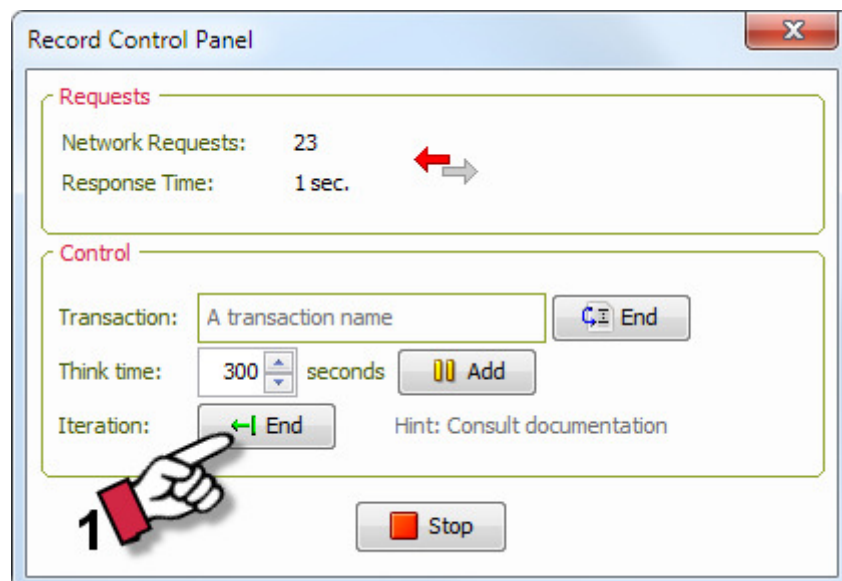
At most one iteration can be defined in a recording session.

While playing back the recorded session you can define how many times the iteration should be repeated and the idle time between two consecutive iterations (in seconds). For instance, six repetitions with an idle time of 10 minutes.

Defining the beginning of an iteration is simple. While recording a user session, click iteration Begin in the recording control panel.



Likewise, to define the end of the iteration, click iteration End. Make sure that the (GUI) state of the Forms application is exactly the same at the beginning of the iteration as at the end of the iteration!



Warning: Iteration for Oracle Forms are very error prone and sometimes not applicable. That differs per Forms application. The generated network traffic of the iteration has to be absolutely idempotent. That is, the generated network traffic has to be the same for each repetition. Therefore, the state of the Forms applications at the beginning of an iteration should be exactly the same at the end of the transaction.

The best way to find out whether an iteration is applicable is to playback the script with and without repetitions. If the script passes without repetitions and fails with repetitions then iterations can most probably not be used for that particular user session.

Ending and saving a recording session

After recording the business process, you end the recording session by closing the Oracle Forms application normally.

Forms2test will recognize this event and provide you with a dialog to save the recorded session to the file system.

Warning: When the Forms application has a trigger defined to load a HTML page on closing the application to close the browser window using Javascript, the Forms application will in fact terminate abnormally. As a consequence, Forms2test will not receive the application exit event and not provide the user with the save dialog.

Although this way of closing the Forms application is end user friendly, another undesired consequence is that the Forms runtime will stay resident until a timeout occurs (default 15 minutes).

➤ Parameterize a script

After recording and saving a user session you can replace recorded keyboard values which have been entered during the recording session with parameters. This is called “script parameterization”. Particularly for a user session where data has been inserted, deleted or updated using the recorded keyboard values will often fail due to violation of integrity constraints, i.e. primary keys. But also for 'read only' user sessions, replacing static keyboard values with parameters (parameterization) is very useful to touch a representative number of records in the database during a load test.

Tip: Before you parameterize the script be sure that the script runs fine and that you have made a backup of the script before editing the script.

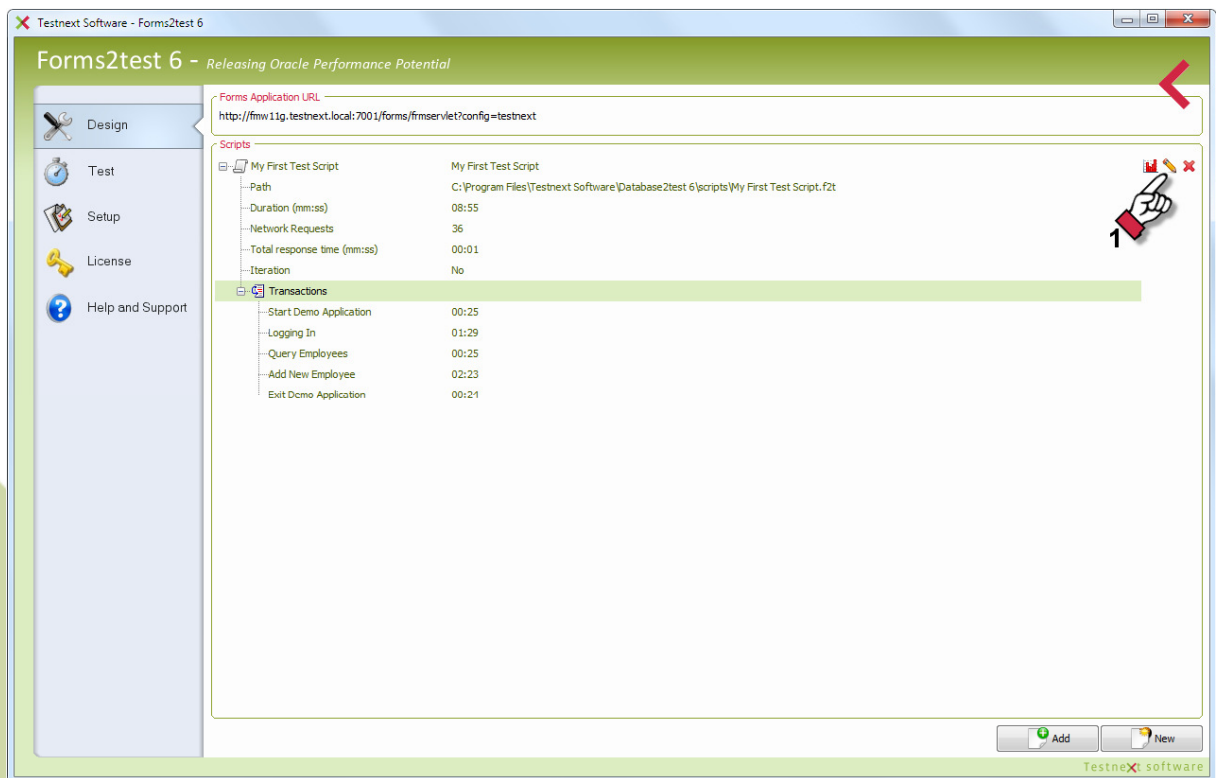
Forms2test supports parameterization of a script with the Script Editor. With the Script Editor the available recorded keyboard values can be replaced with the following parameter types:

- a constant literal value;
- a random numerical value;
- a random alphanumeric value;
- unique numbering (sequence);
- a value from a list (lov);
- a value from another parameter.

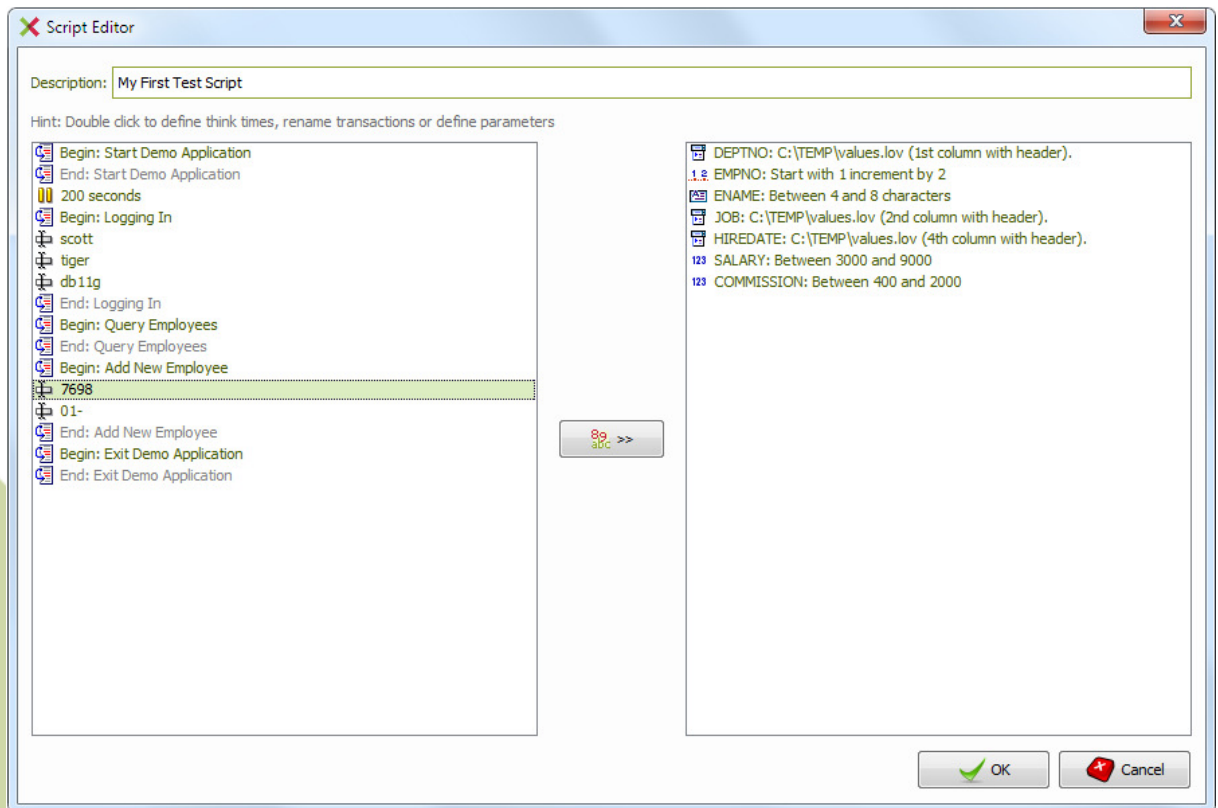
During playback all parameters for each virtual user will be replaced with dynamic runtime values. For example, when you have defined a sequence parameter the first virtual user uses the start value of the sequence, the second virtual user uses the next value, etc.

Script Editor

The Script Editor can be launched from the "Design" tab using the tool bar of a script (see image).



The recorded keyboard values which can be replaced with a parameter are shown in the left pane of the Script Editor together with the Transactions and Think times. The script parameters (if defined) are shown at the right pane (see image).

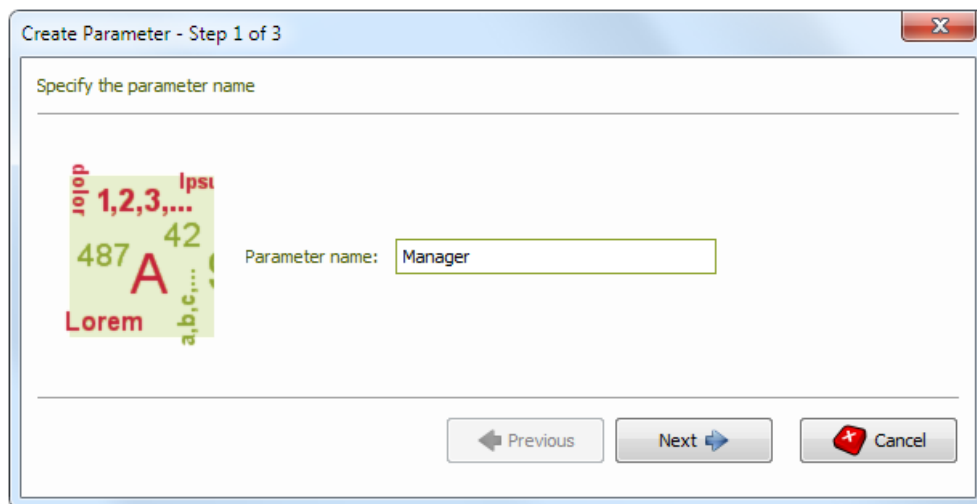


The recorded keyboard values can be replaced with a parameter using the shuttle button or alternatively you can double click on the value.

Then the parameter wizard opens to define in three steps the (1) parameter name, (2) parameter type and the parameter properties (3).

Parameter Name (Step 1)

Each parameter should have a name. With this name it is possible to replace all parameters with the same name with the same runtime value for a virtual user.

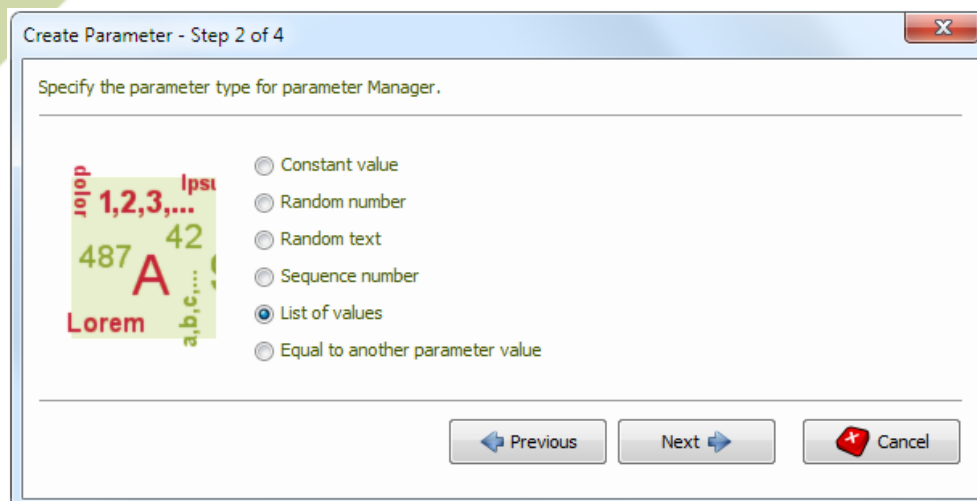


The screenshot shows a dialog box titled "Create Parameter - Step 1 of 3". The main instruction is "Specify the parameter name". On the left, there is a decorative graphic with text like "1,2,3,...", "42", "A", and "Lorem". To the right of this graphic is a text input field labeled "Parameter name:" containing the word "Manager". At the bottom, there are three buttons: "Previous" (disabled), "Next" (active), and "Cancel" (with a red star icon).

Parameter Type (Step 2)

Each parameter should have a parameter type. You can choose from six parameter types:

- Constant value
- Random number
- Random text
- Sequence number
- List of values (LOV)
- Equal to another parameter value



The screenshot shows a dialog box titled "Create Parameter - Step 2 of 4". The main instruction is "Specify the parameter type for parameter Manager.". On the left, there is the same decorative graphic as in the previous step. To the right, there is a list of six parameter types, each with a radio button: "Constant value", "Random number", "Random text", "Sequence number", "List of values" (which is selected), and "Equal to another parameter value". At the bottom, there are three buttons: "Previous" (disabled), "Next" (active), and "Cancel" (with a red star icon).

Parameter Properties (Step 3)

The next step is to define the parameter properties for the parameter type of step 2.

For a *random number* you should define the minimum (inclusive) and maximum (inclusive) for the random number parameter.

For a random alphanumeric value (*random text*) you have to provide the minimum length and the maximum length. At runtime the parameter value will consist of at least minimum length and no more than maximum length arbitrary ASCII characters.

For a *sequence number* you have to provide the start and increment value. Then for each virtual user the parameter value will be increased by the increment value.

Create Parameter - Step 3 of 3

Specify the random number range for parameter SALARY.

Random number between 3000 and 9000

Previous Finish Cancel

For a *list of values* you have to select the file containing the list of values. The list of values file is a flat ASCII file with the file extension .lov.

Create Parameter - Step 3 of 4

Specify a column containing the list of values for parameter Manager.

File: C:\TEMP\Order Entry.lov

Note: Each record is one line of the text file and each value of a record is separated from the next by a semicolon or a tab stop.

Previous Next Cancel

The file consists of an optional header followed by records. Each record is one line of the text file and each value of a record is separated from the next by a semicolon or a tab stop. The values will be cyclically reused as often as necessary.

Example list of values:

```
Customer ID;Sales Rep. ID;Product ID
380;173;1803
719;129;2430
269;100;3083
468;198;3004
756;151;1769
etc...
```

The next wizard step after selecting the lov file is to select the preferred column. For following screenshot the column containing the Sales Rep. ID's is selected.

Create Parameter - Step 4 of 4

Specify a column containing the list of values for parameter Manager.

Customer ID	Sales Rep. ID	Product ID	n/a	n/a
380	173	1803		
719	129	2430		
269	100	3083		
468	198	3004		
756	151	1769		

☒ First line contains column header(s)

Previous Finish Cancel

➤ Running a test

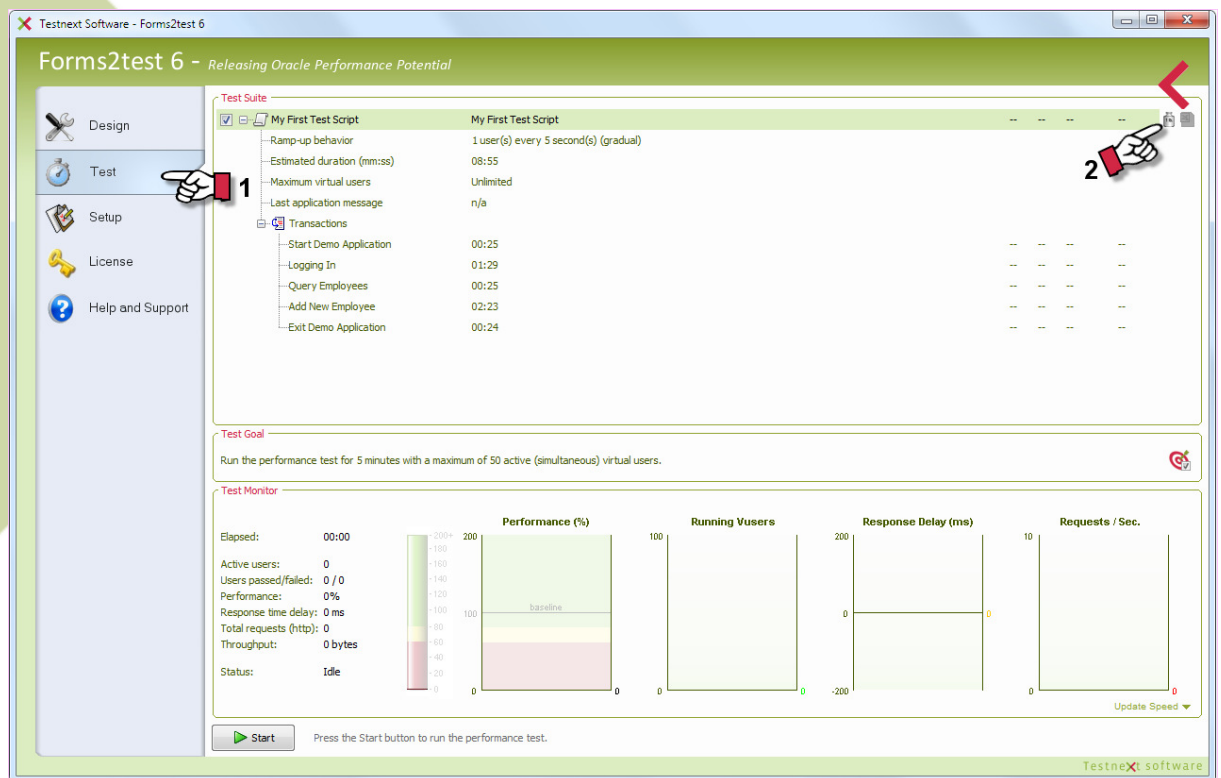
Once you have finished creating and parameterizing scripts, you are ready to define and run a test scenario.

Make sure that the Forms application link points to the correct location (see *Preparing to use Forms2test*)

The host and port number of the Oracle Forms application link will be used as the target environment for the performance test.

Warning: You can change to Forms application link to performance test other Forms environments. However, make sure that the module revisions are equal to the module revision used during script creation.

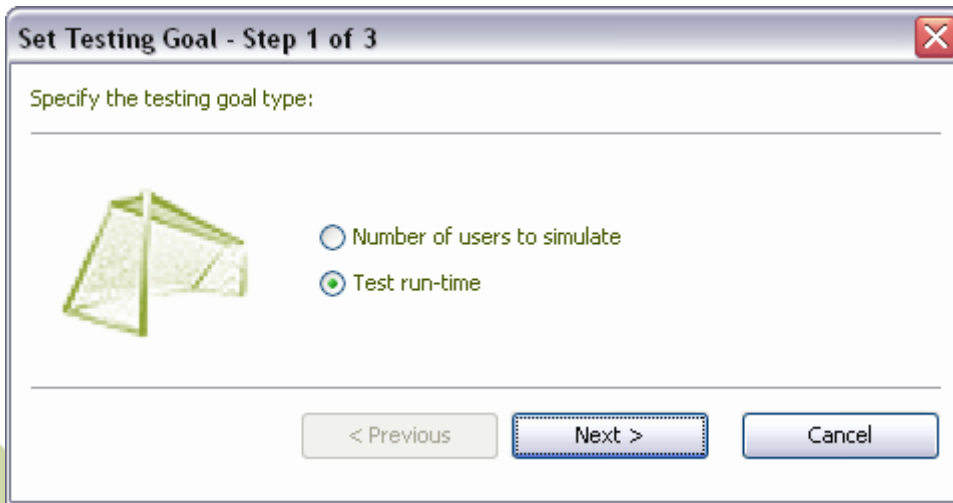
Script schedule



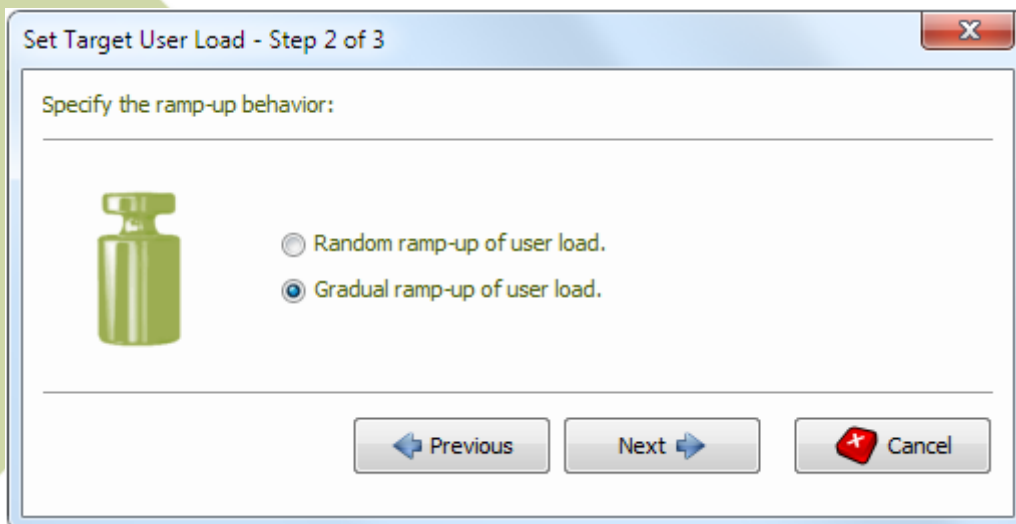
Select the Run page (1) and click the Schedule button (2) for each script that should be included into the test scenario.

The Schedule Script wizard opens to define in three steps (1) the user load, (2) the ramp-up behavior and (3) the maximum number of user sessions.

The first step lets you define the number of virtual users to start within a given time frame.

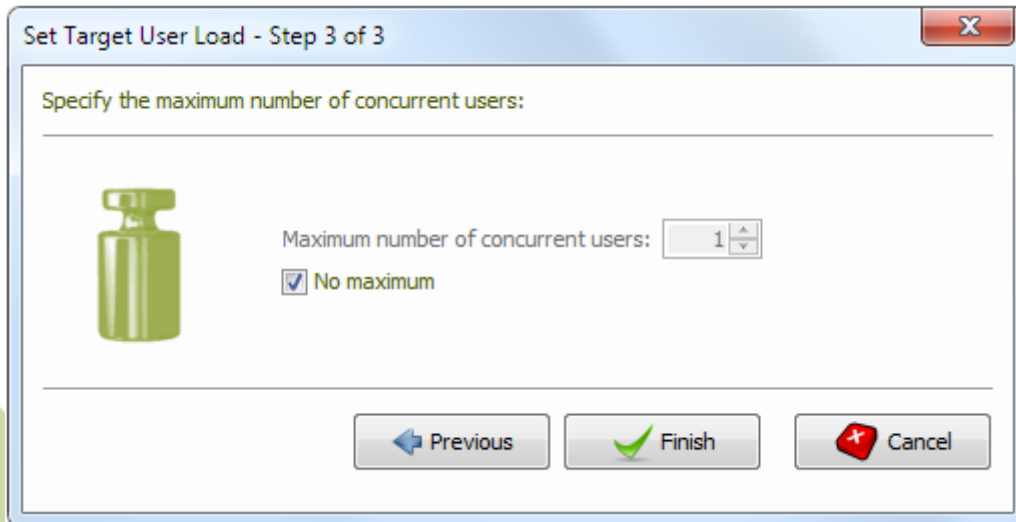


The next step is to define the ramp-up behavior within the time frame of step 1.



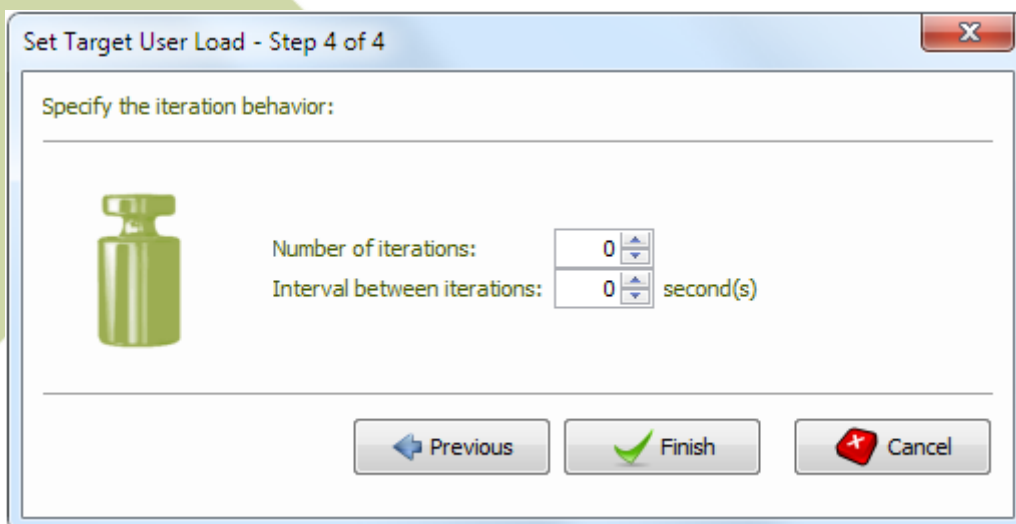
You can let the system determine when to start the users. For example, when you start 100 virtual users every 1800 seconds randomly. The the system will start 100 user sessions every 30 minutes randomly.

The next step, and for scripts without an iteration the last step, is to define the maximum number of users to simulate. When you do not define a maximum (default) the system will start the number of users each time frame until the target goal of the test is reached. However, when you define a maximum the system will not simulate more users than specified in this step for that script.



The screenshot shows a dialog box titled "Set Target User Load - Step 3 of 3". Inside, there is a green bottle icon on the left. To its right, the text "Specify the maximum number of concurrent users:" is followed by a text input field containing the number "1". Below this, there is a checked checkbox labeled "No maximum". At the bottom of the dialog, there are three buttons: "Previous" (with a left arrow), "Finish" (with a green checkmark), and "Cancel" (with a red X).

The last step only applies to script with an iteration. With this step you can define how many times the iteration should be repeated and the time to wait before starting the next iteration.



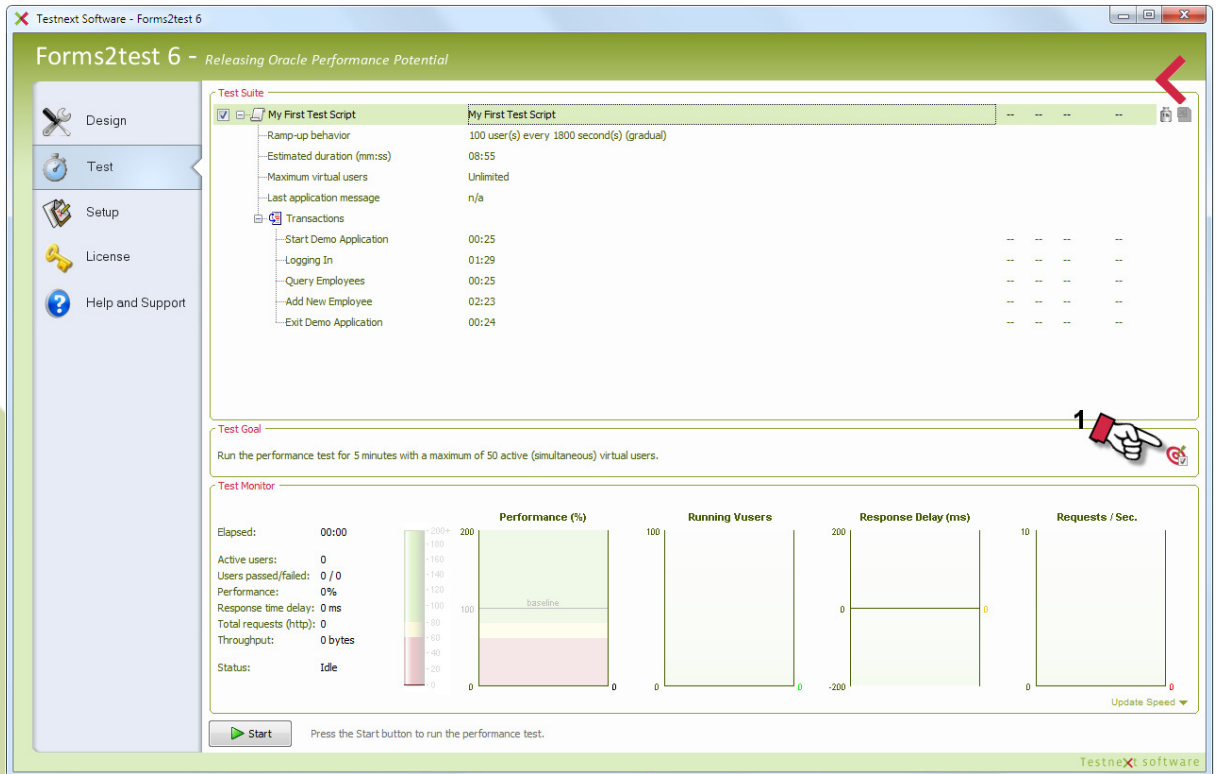
The screenshot shows a dialog box titled "Set Target User Load - Step 4 of 4". Inside, there is a green bottle icon on the left. To its right, the text "Specify the iteration behavior:" is followed by two input fields. The first is labeled "Number of iterations:" and contains the number "0". The second is labeled "Interval between iterations:" and contains the number "0" followed by the text "second(s)". At the bottom of the dialog, there are three buttons: "Previous" (with a left arrow), "Finish" (with a green checkmark), and "Cancel" (with a red X).

Warning: Starting many virtual users within a small time frame will definitely overload the application server. Schedule a reasonable load behavior.

Test Goal

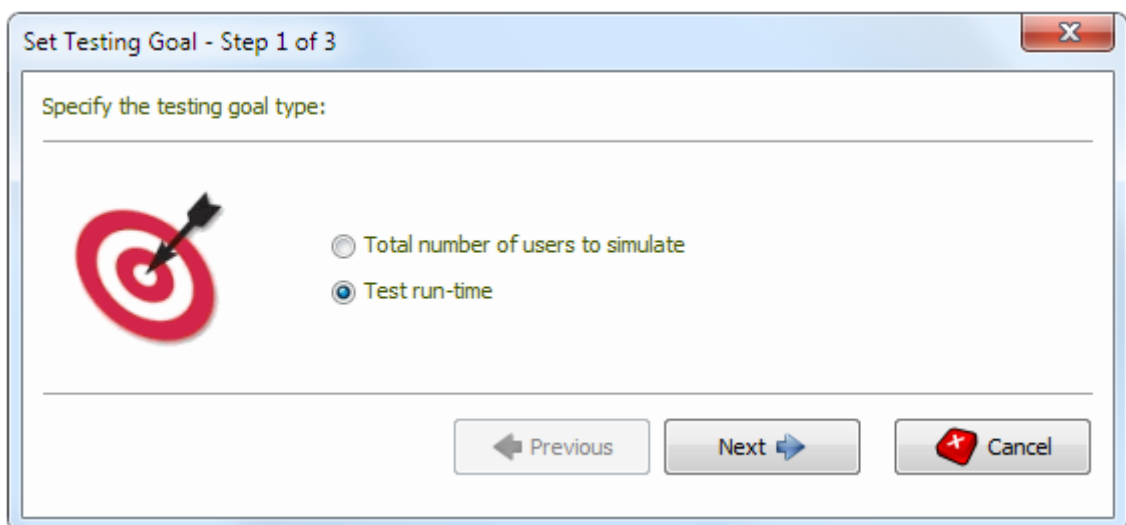
After defining the load behavior for each script individually (note the checkbox ticks for each script included into the test scenario), you should define the overall test goal. You can define the test goal with the Goal Wizard.

To open the Goal Wizard, press the edit button.



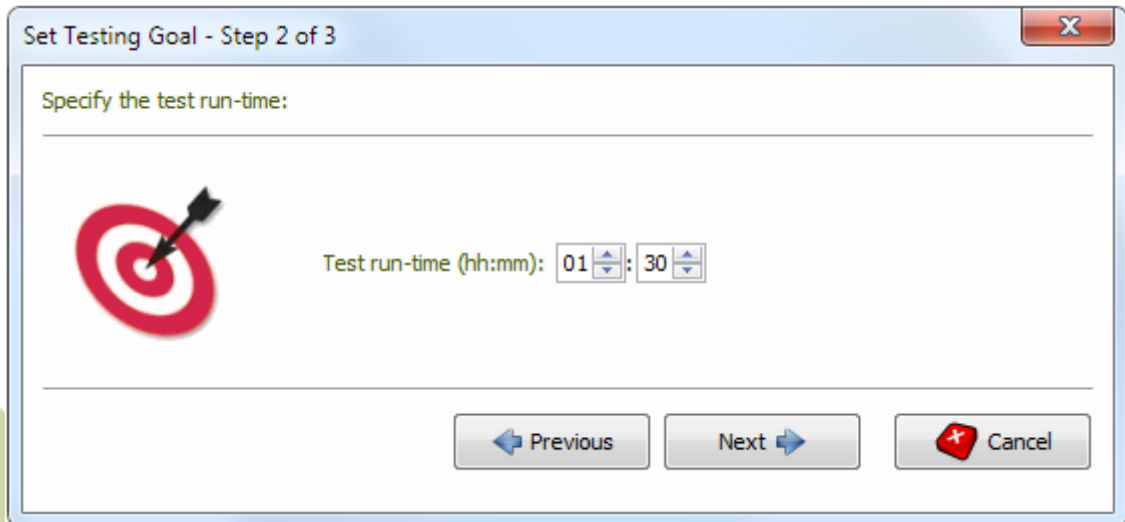
In three steps you can define the testing goal. First, you have to specify the goal type. Then the total number of users to simulate or the test duration and finally the maximum number of active (simultaneous) user sessions.

For the goal type you can choose between the number of users to simulate or a time based performance test.



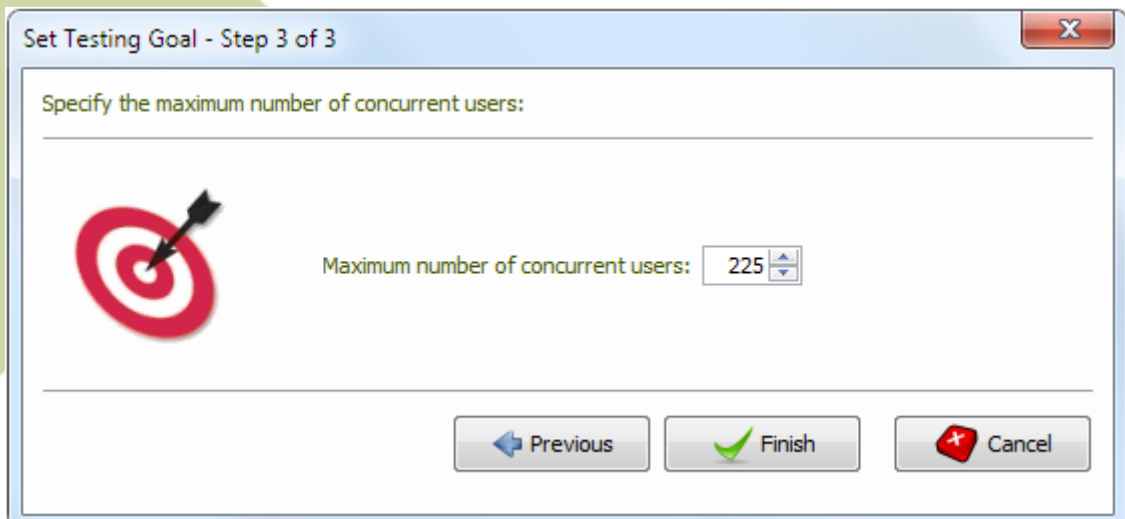
When you select a virtual user bound test you have to select the number of users to simulate for the next step.

When you select a run-time bound test you have to specify the test run-time. Note that when either test goal is reached, the system will not terminate immediate but wait for the active users to stop gracefully. This is the ramp-down phase of the test.



The dialog box is titled "Set Testing Goal - Step 2 of 3". It contains a section labeled "Specify the test run-time:". Below this label is a red target icon with an arrow in the center. To the right of the icon is a text field labeled "Test run-time (hh:mm):" followed by two spin boxes containing the values "01" and "30". At the bottom of the dialog are three buttons: "Previous" with a left arrow, "Next" with a right arrow, and "Cancel" with a red shield icon.

In the last step of the test goal wizard you specify the maximum number of active (concurrent) users.



The dialog box is titled "Set Testing Goal - Step 3 of 3". It contains a section labeled "Specify the maximum number of concurrent users:". Below this label is a red target icon with an arrow in the center. To the right of the icon is a text field labeled "Maximum number of concurrent users:" followed by a spin box containing the value "225". At the bottom of the dialog are three buttons: "Previous" with a left arrow, "Finish" with a green checkmark, and "Cancel" with a red shield icon.

Note: Forms2test allows you to purchase a license to simulate as many users as you need to effectively test your application.

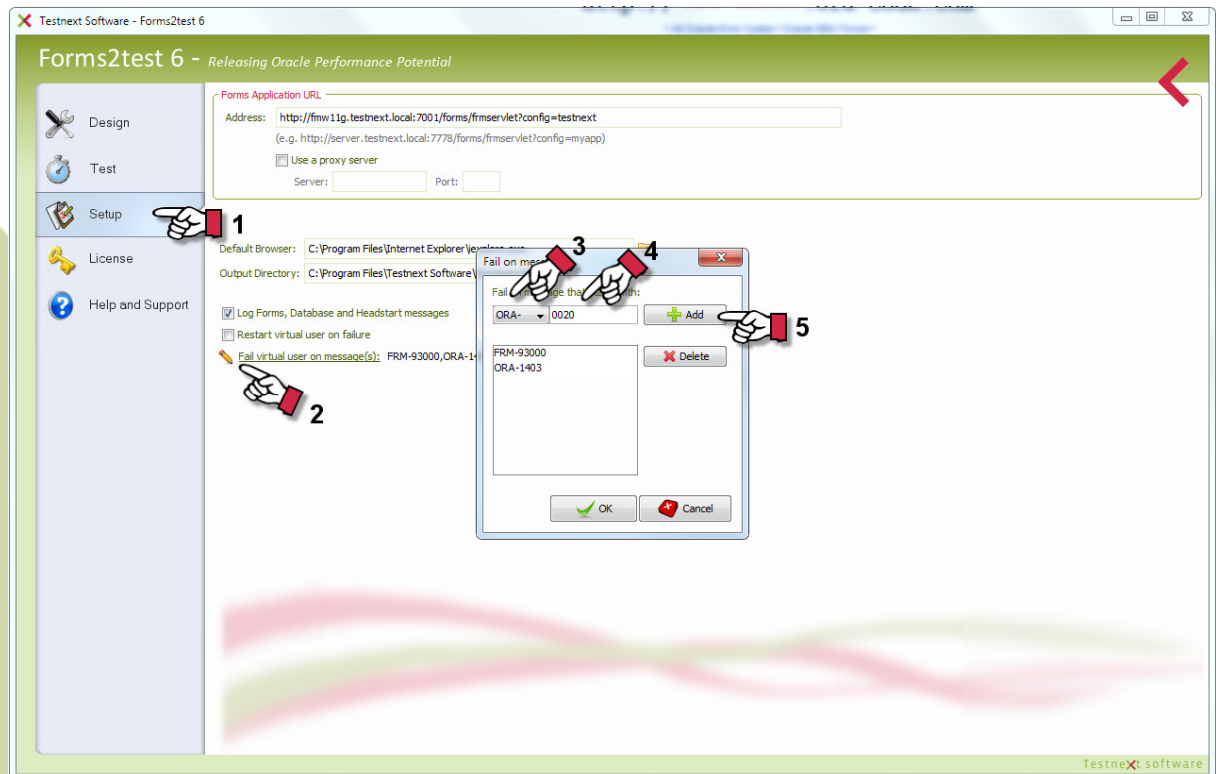
For the (full functional) evaluation version, however, you are licensed to simulate a maximum of 10 active users only.

Fail on Forms messages

During the performance test Oracle Forms might not always run correctly. Although Oracle Forms does not always throw an error, you can fail a virtual user on specific database (ORA-xxx), Forms (FRM-xxx) and Headstart (QMS-xxx) messages.

Select the fail on message button (2) on the Settings tab (1) as shown below, and define one or more message prefixes.

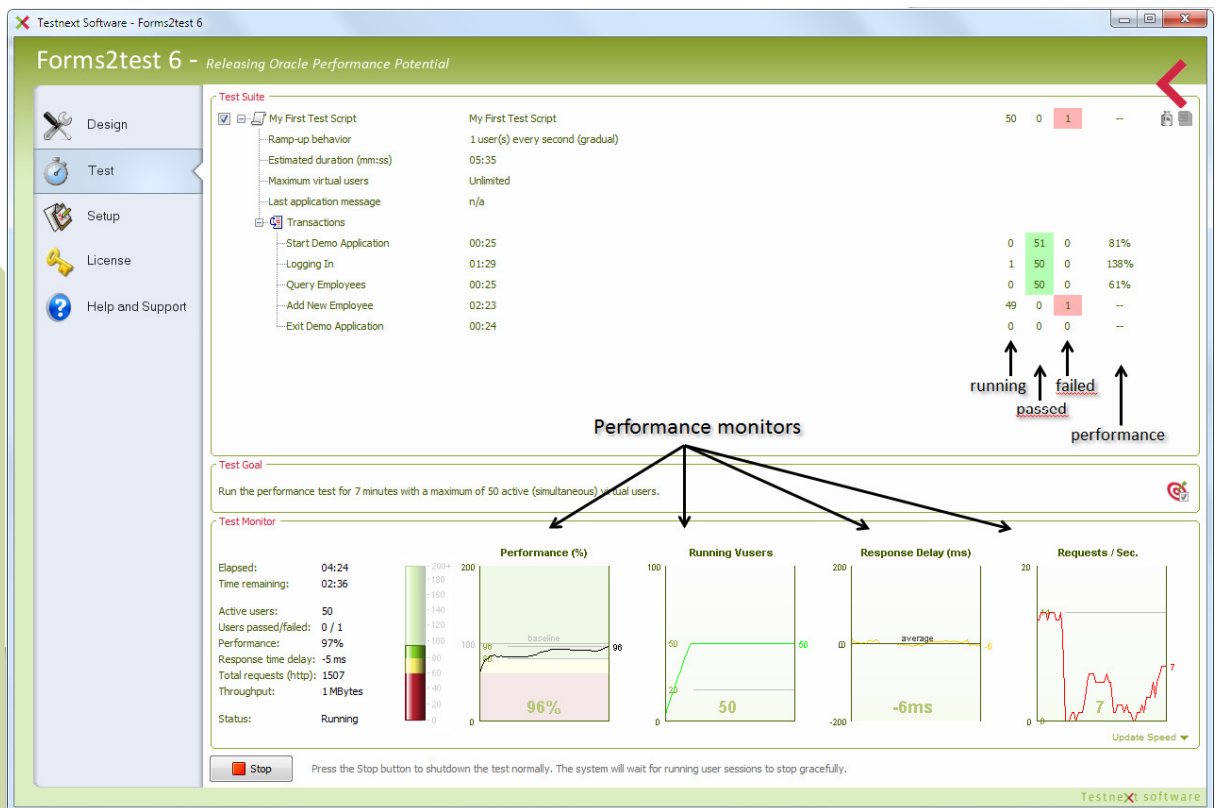
For the example below, a virtual user will fail on all database messages, all Forms messages that starts with FRM-92 and the Forms message FRM-93000 (Generic internal Exception message).



Before you start the test you should familiarize yourself with the Run tab. While the test is running you can see how the application performs in real time. You can view performance information on the online performance monitors and in tabular form.

The Run tab contains three sections. The Test Suite, The Test Goal and the Test Monitor.

The Test Suite section lets you define the test scenario and view the run status (running, passed and failed) for each script and their transactions together with the performance for the *passed* transactions and virtual users.



The Test Goal section shows the current testing goal.

The Test Monitor shows four online (graphical) performance monitors together with the overall test statistics for the test scenario.

The following performance monitors are displayed during the test:

Active Users - displays the number of simultaneous running users.

Performance (%) - displays the performance, i.e. the relation between the cumulative recorded response time and cumulative actual response time (see *What it means*).

Response Delay (ms) – displays the difference between the recorded response times and the actual response times for the last 5 seconds.

Requests / sec. – displays the number of network requests per second for the last 5 seconds.

The following online statistics are displayed during the test:

Elapsed (mm:ss) – The elapsed time for the running test.

Time/Users remaining – The remaining time or virtual users.

Active users – The number of running virtual users.

Users passed/failed – The total number of passed simulated users and the total number of failed users.

Performance – The actual performance.

Response time delay – the average difference between the recorded response time (baseline) and the actual response time in milliseconds for each request over the last 5 seconds.

Total requests (http) – the total number of network requests sent to the application server.

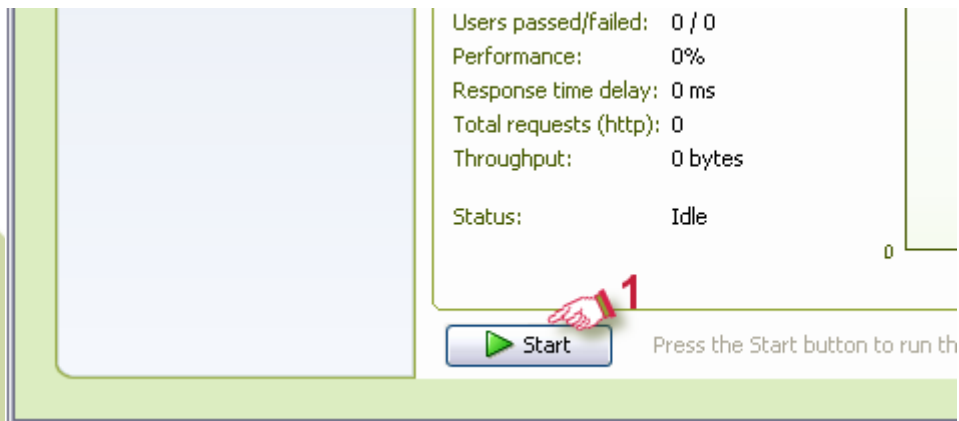
Throughput – the total number of bytes sent to and received from the application server.

Status – current status of the load test: running, stopping, aborting and stopped.

Now you are ready to actually run the performance test.

Forms2test has two run modes. Interactive (GUI) and batch (silent) mode. Batch or silent mode is required to schedule a performance test as a background job. You can run a performance test in batch mode using the operating system script console.bat. This script is located in the bin folder of the forms2test software. Before running the test in batch mode you should first define you test scenario using the graphical interface.

For running the performance test in interactive (GUI) mode you have to press the Start button on the Run tab.



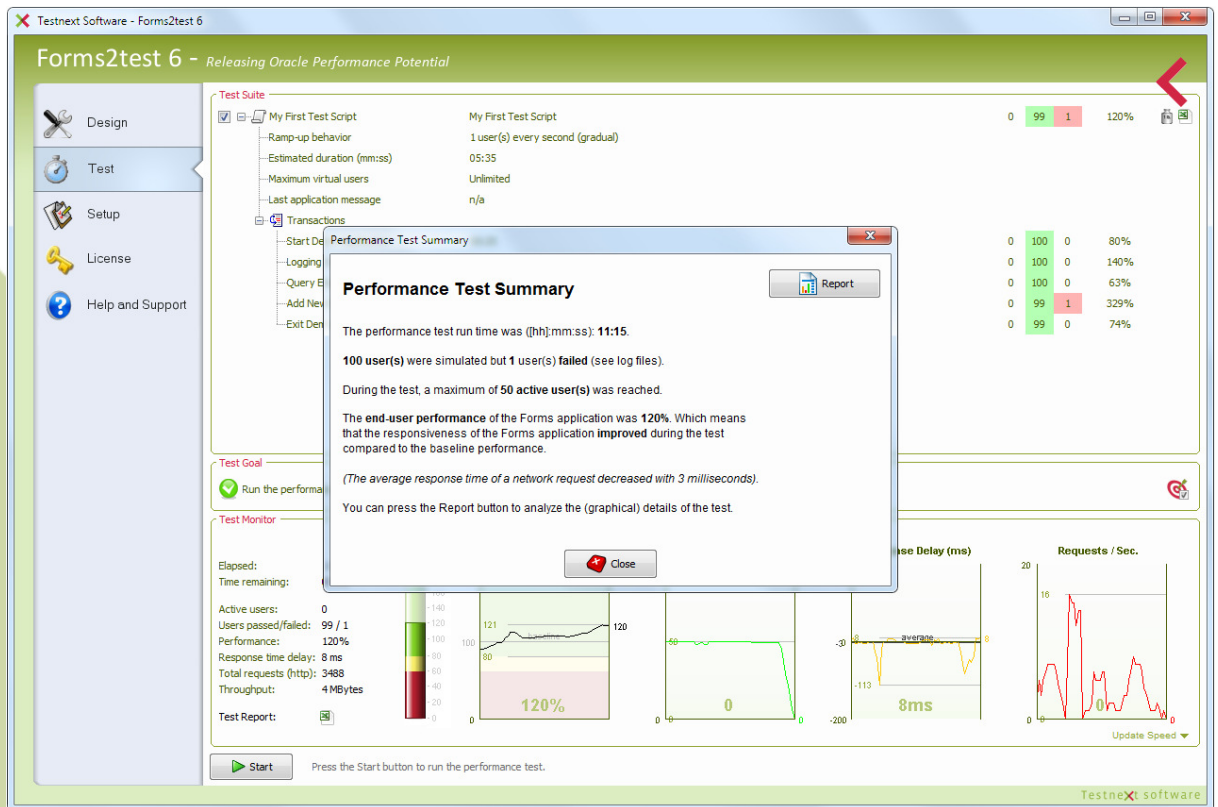
Note that you can stop and abort the test at any time. If you stop the test no more new user sessions are started but the system will wait for the running user sessions to complete gracefully.

However, when you abort the test no more new user sessions are started and the running sessions are terminated forcefully.

Tip: During the test open the Oracle Forms application from a browser and click through the application to experience the real performance under load.

➤ Analyzing the results

At completion of the performance test, Forms2test will automatically generate comprehensive graphical Excel reports using the captured performance statistics and show you a Summary Report dialog. From this dialog you can open the event and error log and the generated whole test report.



The performance reports are saved to the output directory (see Settings page). The performance reports have the following naming convention:

Whole scenario report: forms2test_<yyyymmdd>-<hhmiss>.xls

Per script reports: <script name>_<yyyymmdd>-<hhmiss>.xls

During the performance test, Forms2test captures performance statistics and error and log information to the log directory and the output directory. The output directory can be set in the settings page.

forms2test.err (log directory) – this file contains a log of errors that might be useful for problem determination. This log is stored in the form of a text file.
An error log entry has the following format:

[timestamp] [script name]([session id]) [user action]: [log message]

Example entry:

7-dec-2006 21:20:36 script(0_26) Database log on: ORA-12516: TNS: listener could not find available handler with matching protocol stack

This entry means that the user session identified by “0_26” (the 27th user session) for script “script” received database error ORA-12516 during the execution of user action “Database log on”.

forms2test.log (log directory) – this file contains runtime information about events during the test scenario. It shows among others the session startups, database and forms messages, runtime parameter values, etc. A log entry is similar to the error log

Example entry:

7-dec-2006 21:19:58 script(0_1) Querying data: FRM-40100: at first record

This entry means that user session identified by “0_1” (the 2nd user session) for script “script” received Forms message FRM-40100 during the execution of user action “Querying data”.

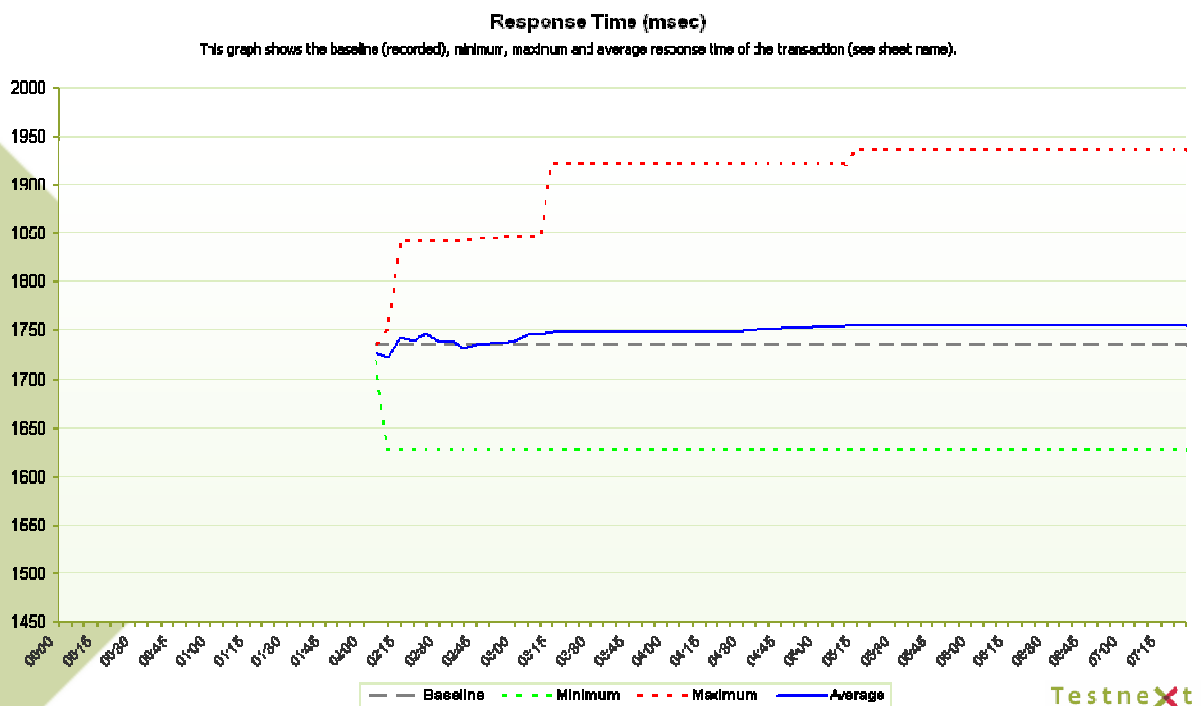
forms2test.csv (output directory) – this file contains whole scenario performance statistics captured by Forms2test every five seconds during the test.
This file is in a comma-separated values (CSV) format and can be imported into a spreadsheet directly.

<script name>.csv (output directory) – Besides the whole scenario performance statistics Forms2test captures also the performance statistics for each script. This file is also in a comma-separated values (CSV) format and contains the same metrics as the whole scenario performance metrics, but also the performance statistics for each user action.

The script reports also contain graphical performance results for each transaction. For a transaction the following statistics are available:

- Baseline response time (in milliseconds)
- Minimum response time during the test (in milliseconds)
- Maximum response time during the test (in milliseconds)
- Average response time during the test (in milliseconds)

The following example report shows the statistics for a transaction. The average response time of all network requests within a transaction is above the baseline response time, the maximum response time increased during the test and the minimum response time decreased.



➤ Troubleshooting

The first step to interpreting the results is to find the cause for failed user sessions, if any. You should examine the error log file to find corresponding error information. Most of the time failures are the result of differences in execution flow during creation of the script and play back. For instance, during playback a message dialog pops up that did not pop up during script creation. Parameterization can change the execution flow of the Forms application or module version and data differences. Often repeated transactions cause errors (see: *defining a transaction*) These type of errors will often result in the following generic error log entry:

Forms session <X> aborted: unable to communicate with runtime process

For error determination you can among others examine the inbound files for the failed user sessions to trace the application flow during the test. An inbound file contains all inbound network traffic for a specific user session. You can easily relate a user session to its corresponding inbound file by its file name. The file name contains the session identifier. From the network traffic, although partly binary, you can deduce the execution flow. Note that inbound files are only available in the output directory when the support logging is set in the configuration tab.

Another common error log entry is:

The Forms runtime failed to respond!

When the forms engine does not respond within the recorded response time *plus* 60 seconds (default threshold) Forms2test will fail and abort the running user session and log the error. This error may for instance occur when the user session waits for a lock or when the web server cannot handle all requests concurrently and puts requests to the backlog queue.

Tip: When the web server cannot handle all incoming requests concurrently it will put requests to the backlog. This situation will always have a negative affect on performance. You should consider setting the maximum number of requests the web server can handle to at least 1.1 times the expected number of end users. See *MaxClients* (Unix/Linux) or *ThreadsPerChild* (Windows).

The next error is directly related to the Operating System configuration of the computer running Forms2test.

Address already in use

The default number of sockets available to Forms2test is too restrictive. Forms2test requires a lot of ephemeral ports when simulating a great number of users. See *Preparing to use Forms2test* for the software requirements and how to extend the port range.

The following error will most likely happen when the middle tier Operating System is Windows.

Forms session <X> failed during startup: no response from runtime

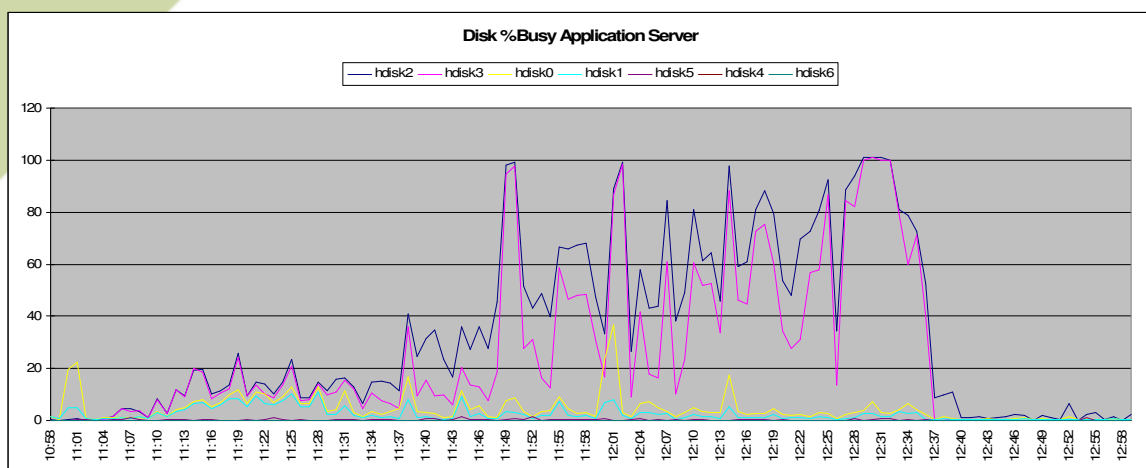
The error is occurring because Windows has exceeded the capacity of the operating systems Non-IO Desktop heap resource.

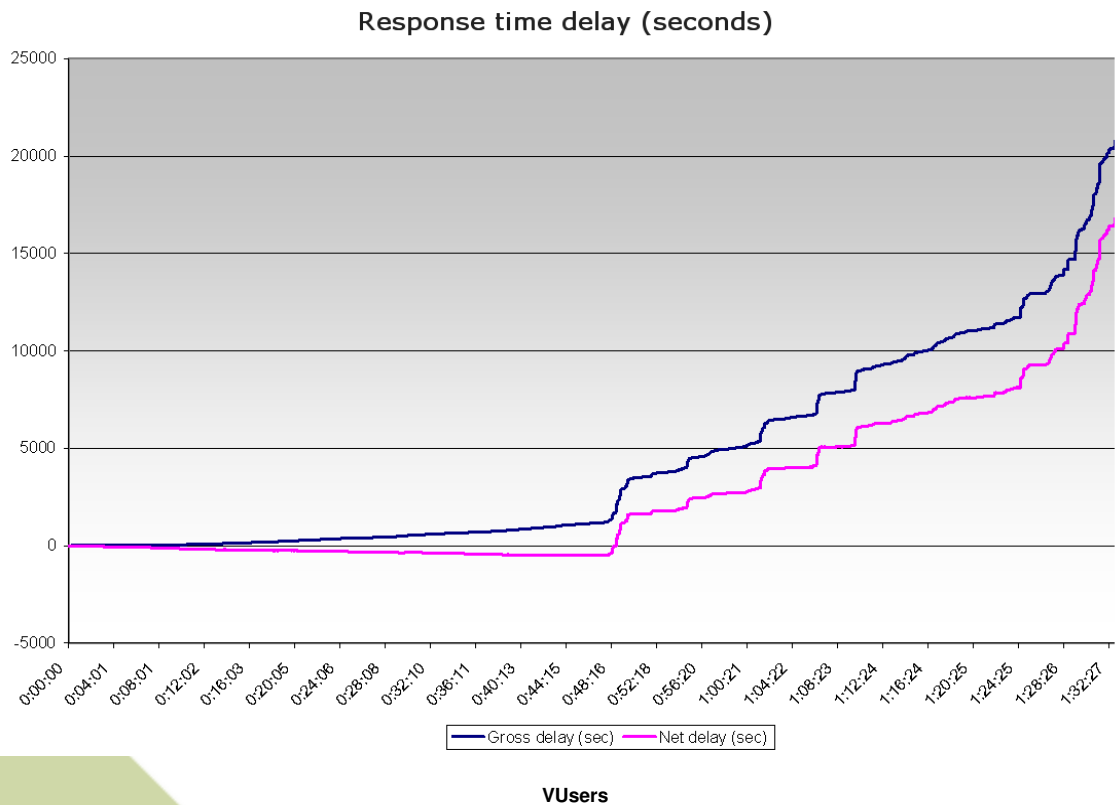
Increase the size of the Non-IO Desktop heap by increasing the value in the Registry: Go to key HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\Session Manager\SubSystems and increase the third SharedSection parameter from 512 (default) to 3072. Reboot for the new values to take effect (see Metalink Note 207706.1)

Often you can relate failed sessions to server statistics. In the following three graphs for instance you see a correlation between swapping (memory shortage) of the application server and the response time delay and number of failed user sessions.

The following errors were written to the error log:

- Forms session <X> aborted: unable to communicate with runtime process
- The Forms runtime failed to respond





The Oracle Forms environment appears to be very sensitive for physical memory shortage. As a rule of thumb you need about 12 MB of memory per runtime.

Note that a user session is not the same as an end user. Often an end user has more than one browser session opened.

> What it means

Elapsed – Elapsed time since the start of the performance test.

Active – Total number of running user sessions.

Passed – Total number of passed user sessions. A passed user session is a user session that did not receive an error during the test.

Note: *passed does not automatically mean that the test succeeded. You are strongly advised to analyze/verify the test results.*

Failed – Total number of failed user sessions.

Response time – The time between sending a request to the application server and receiving a response from the application server.

Performance – The relation between the cumulative response time during the test and the cumulative response time during script creation.

A performance *more* than 100% means that the cumulative response time during the test was less than the cumulative response time during script creation.

A performance *less* than 100% means that the cumulative latency during the test was more than the cumulative latency during script creation.

Requests (http) – number of requests sent to application server.

Requests per second – average number of requests per second for the last five seconds.

Connections (http) – total number of socket connection setups during the test.

Cumulative connection setup time (sec) – total amount of time elapsed for setting up the socket connections.

Throughput Inbound (kB) – Total amount of data received from application server.

Throughput Outbound (kB) – Total amount of data sent to application server.

Cumulative response time (sec) – Sum of the response time of the network requests during the test.

Cumulative recorded response time (sec) – Sum of the response time of the http requests during script creation. This statistic is called the performance reference or baseline.

Performance (active) – The performance of the active (running) sessions only.

Performance (passed) – The performance of the passed user sessions only.

Performance (overall) – The performance of all user sessions.

Response time delay per request (msec) – Average difference between the recorded response time and the response time during the test for all network requests over the last five seconds.

Average net response time delay per request (msec) – Average difference between the recorded response time and the response time during the test for all network requests (in milliseconds).

Average gross response time delay per request (msec) – Equal to the average net response time delay per request but without negative delays (in milliseconds). This metric is always positive and focused only on the 'real' delays.