



PERFORMANCE
TEST SOLUTIONS
FOR ORACLE

Mapviewer2test

Release 5

User's Guide

> Table of Contents

Preface	3
Preparing to use Mapviewer2test	4
Requirements	4
Installation	4
Running Mapviewer2test	5
Configuring Mapviewer2test	5
The testing process	6
Creating a script	7
Create a MapViewer XML map request	8
Create a MapViewer Maps map request	9
Create a MapViewer WMS map request	10
Establishing the baseline performance	12
Defining the target load	13
Parameterize a script	14
Adding map request parameters	14
Defining map request parameters	16
Parameter Type	16
Parameter Properties	17
Running a performance test	18
Defining the test scenario	18
Defining the test goal	19
Analyzing the results	25
What it means	28

> Preface

Welcome to the TestNext Mapviewer2test User's Guide. This guide provides a step-by-step overview to using TestNext Mapviewer2test.

Mapviewer2test is a performance test tool dedicated to Oracle MapViewer.

It can be used to simulate a heavy load on a server to test its strength or to analyze overall performance under different load types. You can use it to make a graphical analysis of performance or to test your server behavior under heavy concurrent load.

Mapviewer2test supports all releases of Oracle MapViewer (10.1.2, 10.1.3 and 11g) and supports all map interfaces, including Oracle MapViewer XML, Oracle MapViewer Maps and Oracle MapViewer WMS.

With Mapviewer2test you can define and run a performance test in a few steps using the powerful wizards.

During the performance test a lot of metrics, such as response times, data throughput, network 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.

➤ Preparing to use Mapviewer2test

Requirements

Mapviewer2test requires your computing environment meets some minimum requirements. Mapviewer2test should be installed on a reasonably modern Windows desktop with at least 1.2GHz processor and 512Mb of RAM (memory).

Mapviewer2test is a 100% Java application and requires a fully compliant JVM 1.6 or higher. We recommend you to download the distribution of Mapviewer2test including the Java 6 runtime.

Mapviewer2test has been tested and should run correctly under Windows (NT, 2000, 2003, 2008, XP, Vista and Windows 7).

Although it is not required you are strongly encouraged to install the demo dataset that is part of the Oracle MapViewer distribution before using Mapviewer2test, see

<http://www.oracle.com/technology/software/products/mapviewer/index.html>

MapViewer2test contains some sample scripts for the datasets and these sample scripts are a good starting point for getting familiar with Mapviewer2test.

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 Mapviewer2test is a snap. Mapviewer2test is available as a Windows installer.

Launch the installer and install the full functional version into the directory where you want Mapviewer2test to be installed.

Please note that installing Mapviewer2test *will overwrite* your existing installation. However, scripts, settings and registrations won't be lost.

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

Running Mapviewer2test

After the installation has completed the installer has created start menu items to launch (or uninstall) Mapviewer2test.

Note: If the start-up fails, ensure that the `JAVA_HOME` variable points to the location where you have installed Java 6 or higher. This variable is set in the `setenv.bat` OS script.

Configuring Mapviewer2test

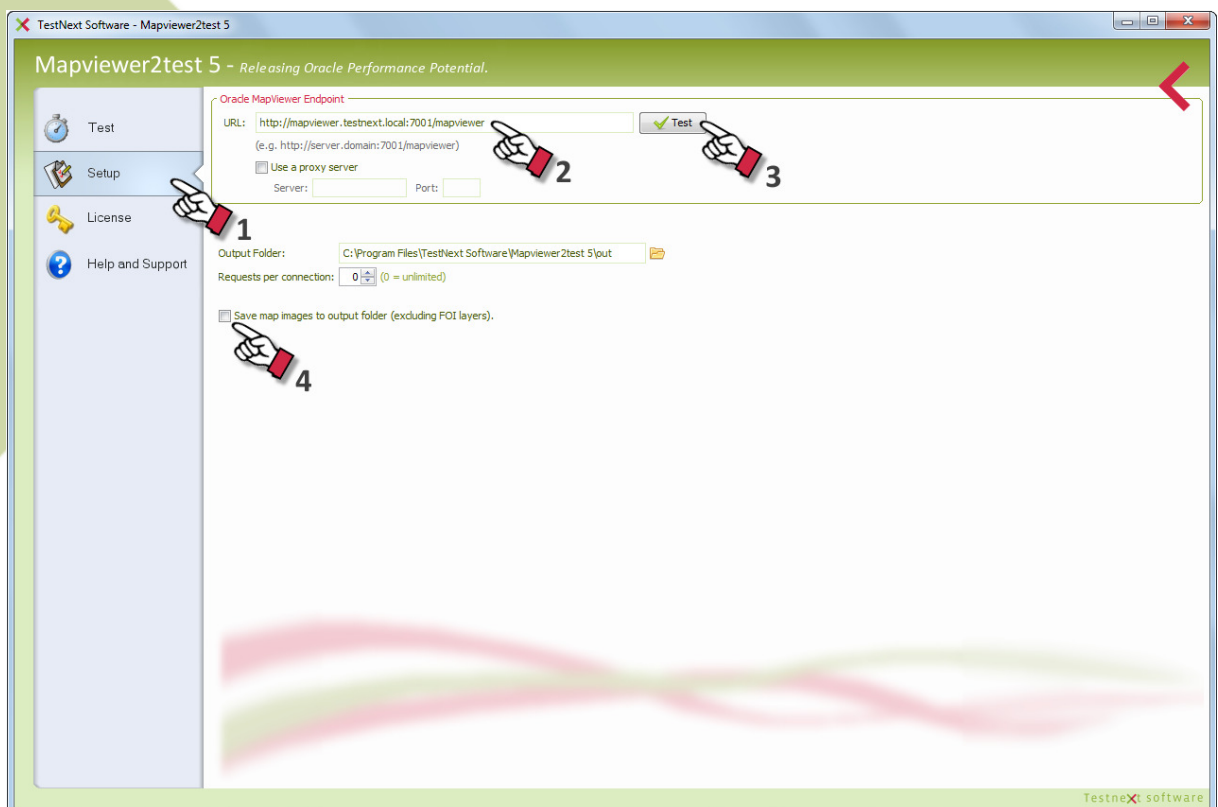
If Mapviewer2test is started for the first time you are automatically directed to the Settings section (1).

The only required configuration setting for Mapviewer2test is the service URL of Oracle MapViewer (2).

If you are testing from behind a firewall/proxy server, you may need to provide Mapviewer2test with the firewall/proxy server hostname and port number.

After setting the service URL you should test the availability of the service by pressing the Test URL button (3).

Optionally you can save the generated map images to a local output folder. This option is especially interesting for testing scripts (4).



> The testing process

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

- Load test, and
- Stress test (*optional*)

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 the Oracle MapViewer environment can handle.

For a load test you are interested in the performance and resource usage of the Oracle MapViewer environment during typical production load.

Mapviewer2test is suitable for both type of tests.

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

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. A test scenario defines the type of requests (MapViewer XML, Maps or WMS) and number of map requests.

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 map requests of the target load differs.

➤ Creating a script

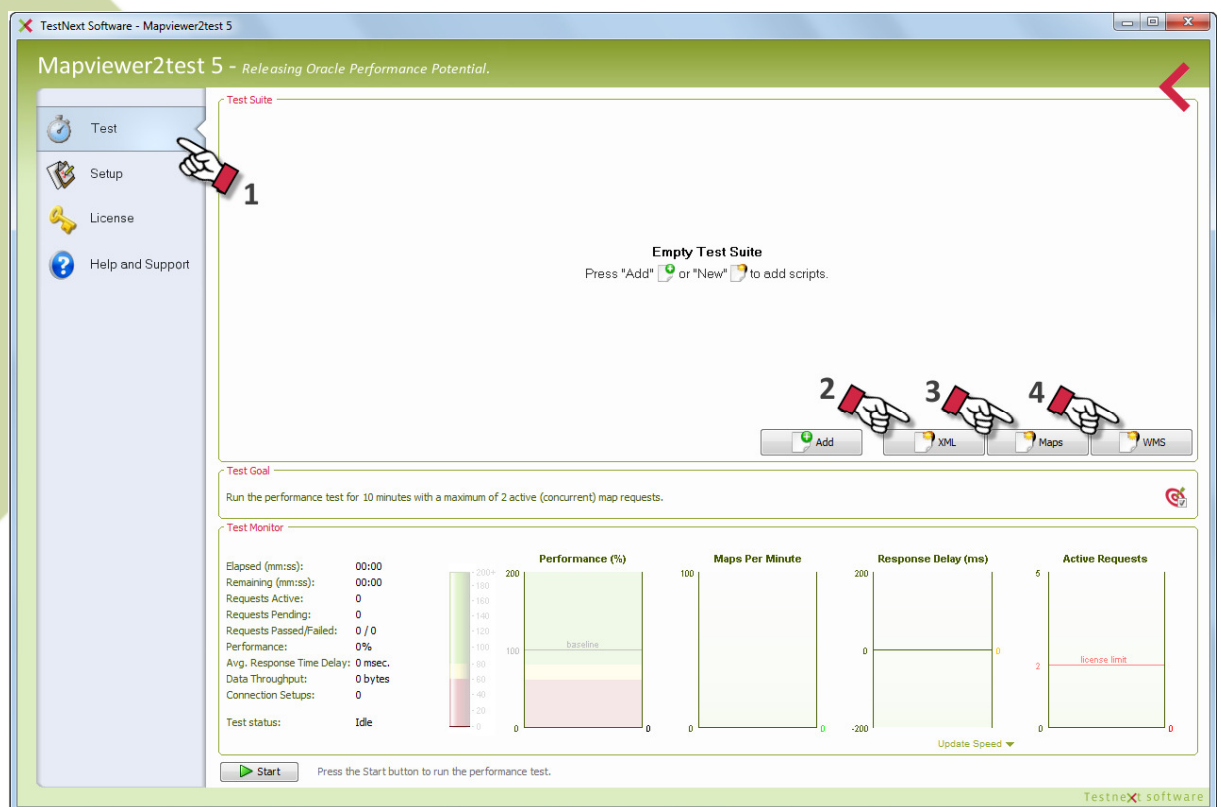
In this section you will learn how to create a Mapviewer2test script.

Make sure that the Oracle MapViewer service URL points to the correct location (see *Preparing to use Mapviewer2test*)

Choose Start > Programs > Mapviewer2test 4 > Mapviewer2test 4 to launch Mapviewer2test.

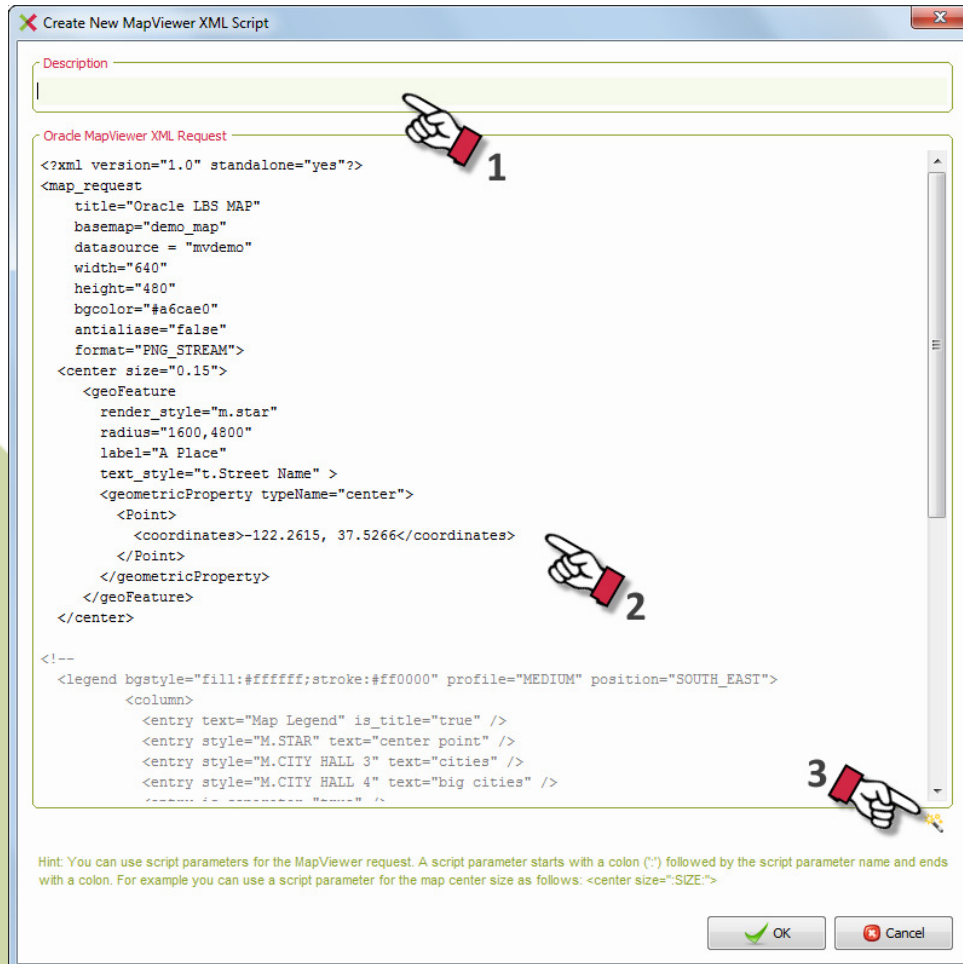
Select the Test menu option (1) and select the preferred request type, MapViewer XML (2), Oracle Maps (3) or MapViewer WMS (4).

Note: Oracle Maps is only available from Oracle MapViewer 10g release 3. Choosing the Oracle Maps request type for older releases of Oracle Mapviewer will result in an error message.



Create a MapViewer XML map request

When you select to create a new MapViewer XML map request the following request editor pops up:



First, you may enter an appropriate description for the request (1). The description will be visible in the Test Suite.

The second step is to define the XML map request (2). The request editor will provide you with a default XML request. Similar like the Oracle MapViewer web console to submit an XML request.

To avoid entering valid XML request you can also use the Map Content wizard (see 3) to define your XML map request. The wizard will paste the appropriate and valid XML request for you.

You should adjust the attribute values according to your situation like the datasource, basemap and ordinate information.

Note that you can also parameterize an attribute (see Parameterizing a map request).

Hint: Setting the MapViewer logging level to 'finest' (see `mapViewerConfig.xml`) MapViewer will record all XML requests to the log file. You can use these XML requests to define XML-scripts.

Create a MapViewer Maps map request

When you select to create a new MapViewer Maps map request the following request editor pops up:

The screenshot shows a dialog box titled "Create New Oracle Maps Script". It contains four main sections, each with a numbered callout (1-4) pointing to it:

- 1 Description:** A text input field for entering a description.
- 2 Map Tile Layers:** A dropdown menu for selecting a map tile layer. The current selection is "MVDEMO.CUSTOMER_MAP".
- 3 Map Tile Layer Parameters:** A section containing several input fields for parameters:
 - Coordinate system: 8307
 - Coordinate system boundary: -180.0,-90.0,180.0,90.0
 - Zoom levels: 10
 - Map width (pixels): 600
 - Map height (pixels): 400
 - Minimum zoom level: 0
 - Maximum zoom level: 9
 - Minimum center X: -180.0
 - Maximum center X: 180.0
 - Minimum center Y: -90.0
 - Maximum center Y: 90.0
- 4 Theme Based FOI Layers (Optional):** Four dropdown menus for selecting theme-based FOI layers, labeled "Theme based FOI layer 1" through "4".

At the bottom of the dialog, there is a hint: "Hint: Select first the map tile layer. Then fill out the required test parameters which will be used to generate the load." and two buttons: "OK" and "Cancel".

The first step (1) is to enter an appropriate description which will be visible in the Test Suite. The next step (2) is to select the preferred map image cache. The editor will only show the map cache instances that are created before with the Oracle MapViewer web console. Step (3) is to modify the several test parameters. For instance when your Oracle Maps application has a different map width or map height you should modify the default values. And finally, you can add up to four predefined FOI themes (4) During the test Mapviewer2test will generate Oracle Maps request using the provided test parameter values.

Create a MapViewer WMS map request

When you create a new MapViewer WMS map request the following request editor pops up:

The screenshot shows a dialog box titled "Create New WMS Script". It contains the following elements:

- Description:** A text field with a red hand icon pointing to it labeled "1".
- WMS Required Parameters:** A section containing several input fields:
 - REQUEST:** Set to "GetMap". A red hand icon points to it labeled "2".
 - SERVICE:** Set to "WMS".
 - VERSION:** Set to "1.1.1". A red hand icon points to it labeled "3".
 - FORMAT:** Set to "image/png".
 - SRS:** An empty field.
 - LAYERS:** An empty field with a "+" button and a help icon (?) to its right.
 - BBOX:** An empty field with a help icon (?) to its right.
 - WIDTH:** Set to "600".
 - HEIGHT:** Set to "400".
- Advanced:** A button with a dropdown arrow, pointed to by a red hand icon labeled "4".
- Hint:** A text block explaining script parameters, pointed to by a red hand icon labeled "5".
- Buttons:** "Edit WMS URL...", "Clear", "OK", and "Cancel".
- Map Content Wizard:** A red hand icon labeled "6" points to a small icon in the bottom right corner.

The first step (1) is to enter an appropriate description which will be visible in the Test Suite. The next step is to fill out at least the required WMS parameters (2). For each parameter you can open a help dialog showing the description for the parameter (3).

You can enter the optional WMS parameters and Oracle MapViewer specific WMS parameters by pressing the Advanced button (4).

Optionally, you can directly modify/enter the WMS URL by clicking the Edit WMS Query button (5).

To accelerate entering all WMS parameters you can also use the Map Content wizard (6) to define your WMS request. The wizard will fill out the appropriate WMS parameters for you.

Note that you can also parameterize a WMS parameter value (see Parameterizing a map request).

Create New WMS Script

Description

WMS Required Parameters

REQUEST

GetMap

SERVICE

WMS

VERSION

1.1.1

FORMAT

image/png

?

SRS

?

LAYERS

+

?

BBOX

?

WIDTH

600

?

HEIGHT

400

?

Basic

WMS Optional Parameters

TRANSPARENT

?

BGCOLOR

?

EXCEPTIONS

application/vnd.ogc.se_xml

WMS Optional MapViewer Parameters

DATASOURCE

?

BASEMAP

?

DYNAMIC_STYLES

?

LEGEND_REQUEST

?

MVTHEMES

?

Hint: You can use script parameters for any WMS parameter value. A script parameter starts with a colon (':') followed by the script parameter name and ends with a colon. For example you can define 4 script parameters for WMS parameter BBOX using the value :MIN_X;:MIN_Y;:MAX_X;:MAX_Y:

Edit WMS URL...

Clear

OK

Cancel

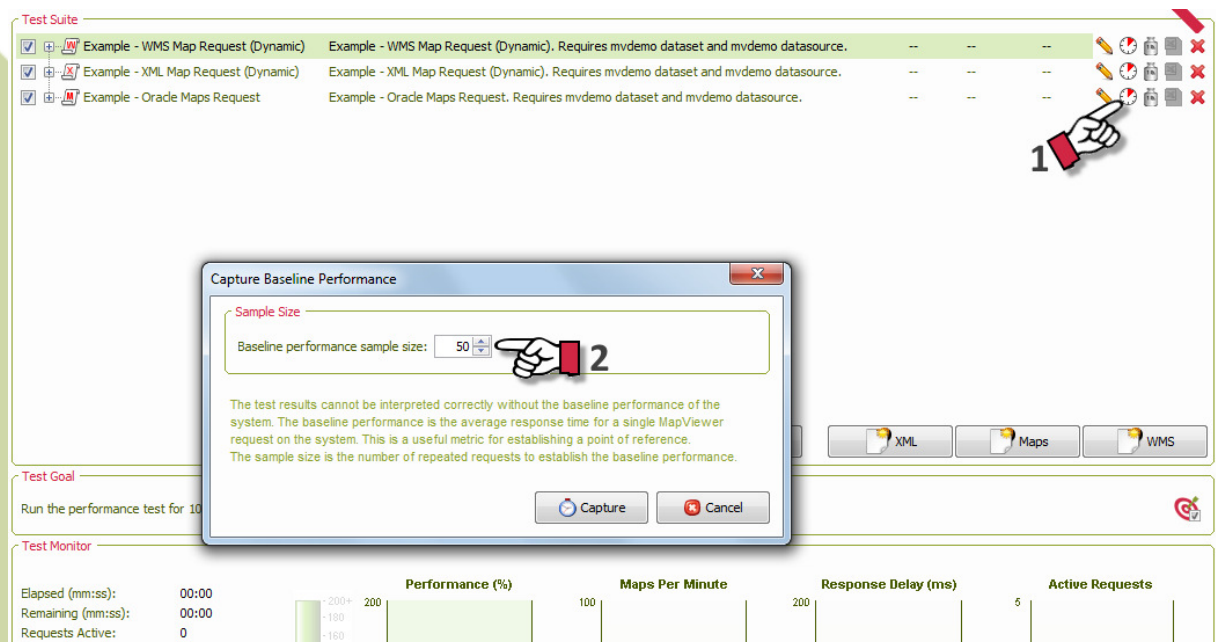
Establishing the baseline performance

The next step after creating the map request is to capture the baseline performance for the map request. The baseline performance represents the (average) response time for a single map request on the system.

This is a useful metric for establishing a point of reference. During the test the performance is determined relative to this metric.

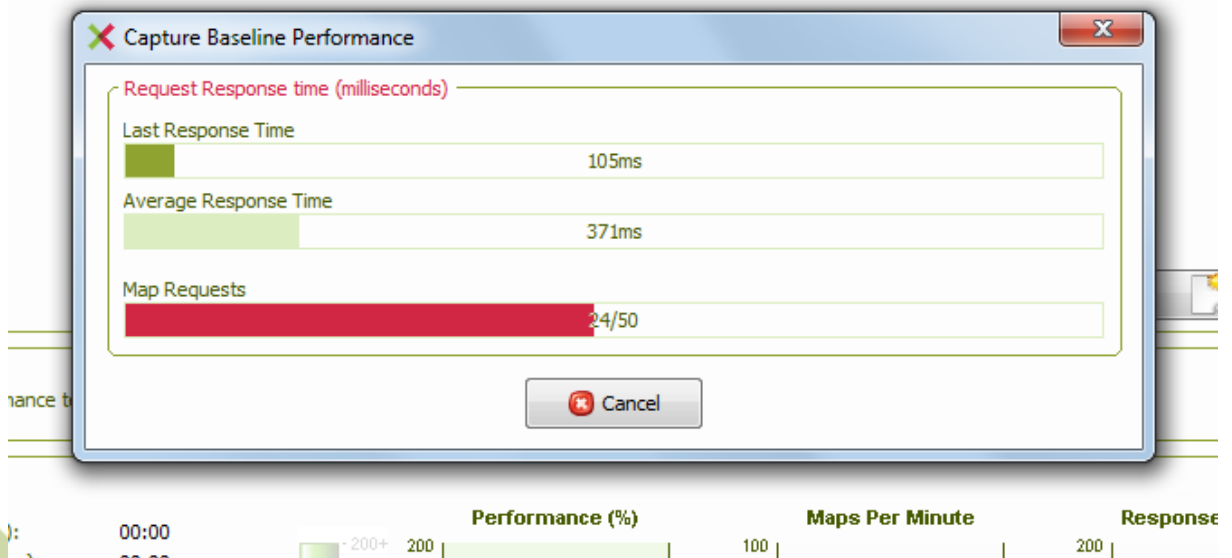
The baseline performance is a mandatory metric. You cannot run a performance test without establishing the baseline performance first.

To establish the baseline performance you have to select to the corresponding button (1). A dialog pops up with an option to set the sample size. The sample size tells Mapviewer2test how many times the map requests will be sent to the Oracle MapViewer to determine a reliable average response time.



The sample size should depend on whether the map request has parameters or not. If the map request is static, i.e. has no parameters, the sample size could be low (default 10). However, when the map request is dynamic, i.e. has parameters, the sample size should be sufficiently high to get a representative performance baseline.

When the baseline performance is being captured the following dialog is visible showing the intermediate results.



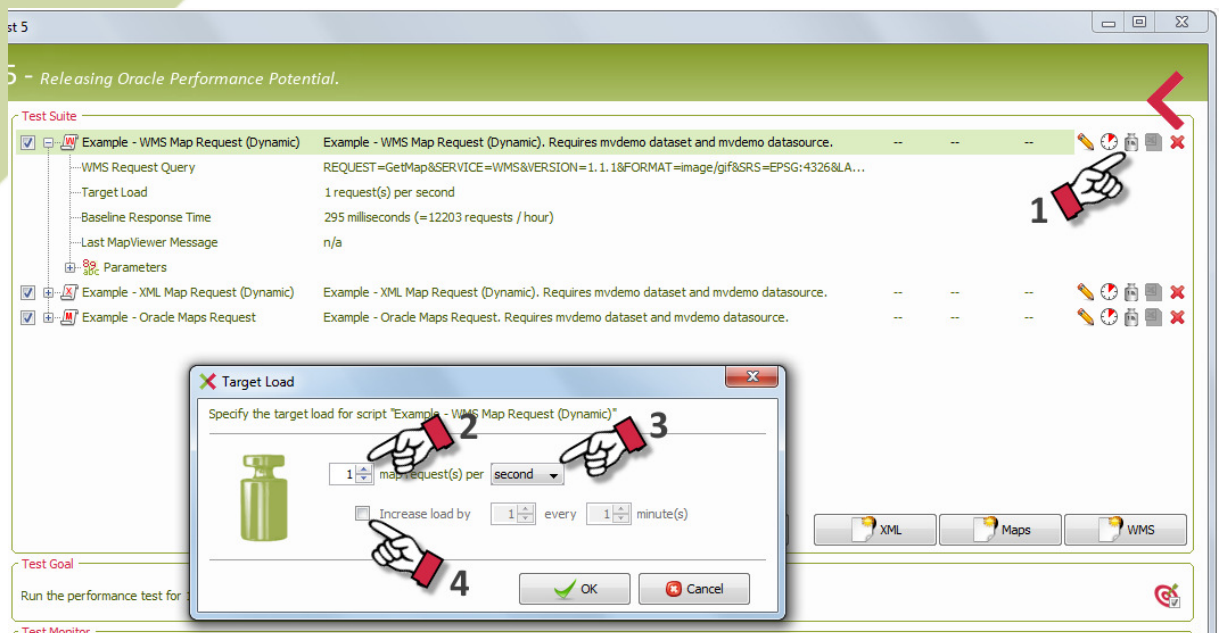
Defining the target load

After establishing the baseline performance you should define the target load for the map request during the performance test.

To define the target load you should press the corresponding button (1). A dialog pops up to specify the target load for the map request.

You should specify the number of requests (2) and the corresponding interval (3).

Optionally, you can increment the load during the test (4).



> Parameterize a map request

During script creation you can also parameterize a map request by replacing static values by parameters. At runtime the dynamic parameter values will be generated based on the corresponding parameter definition. Parameterization is very useful to conduct a realistic performance test.

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

Adding map request parameters

The first step to parameterization is to add one or more parameters to a map request.

Adding parameters to a map request is only available for MapViewer XML and MapViewer WMS request types. For Oracle Maps the parameters are always created automatically based on the values supplied during request creation.

To add a parameter to a map request you should replace a static value by the parameter name enclosed by colons, i.e. **:<parameter name>:**.

The request editors will automatically recognize this parameter construct.

For example, to replace the ordinates of a point element by two parameters, you should replace the line:

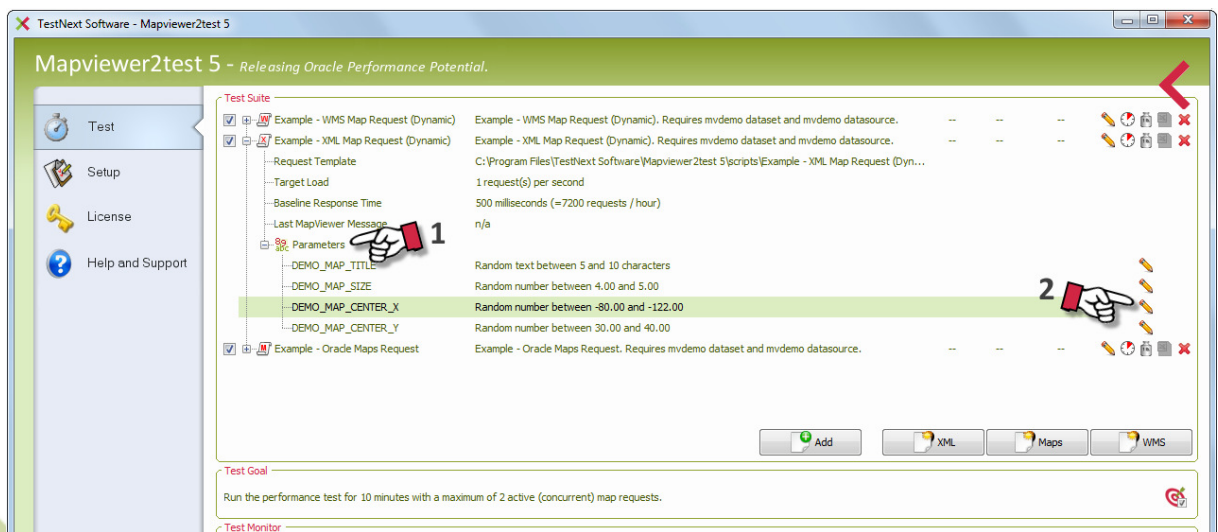
```
<Point>
  <coordinates>-122.2615,37.5266</coordinates>
</Point>
```

by

```
<Point>
  <coordinates>:DEMO_MAP_CENTER_X:, : DEMO_MAP_CENTER_Y:</coordinates>
</Point>
```

Note: The parameter name is limited to 32 characters including underscores.

This will result in the following a script with two parameters (1). You can adjust the parameter definition by double clicking on the parameter name or by clicking the corresponding edit button (2):



For parameterizing a bounding box the presented construct is not suitable. Therefore you need a more comprehensive parameterization construct.

For instance to parameterize the BBOX parameter (WMS) you can use the following solution:

```
:BBOX_MIN_X: , :BBOX_MIN_Y: , :BBOX_MIN_X+4: , :BBOX_MIN_Y+4:
```

For the lower left ordinates of the bounding box you just add two parameters, BBOX_MIN_X and BBOX_MIN_Y. To make the upper left of the bounding box relative to the lower left you can define a constant offset with the following construct: :BBOX_MIN_X+4:

This way you tell Mapviewer2test to add 4 units of the coordinate system to the parameter value of parameter BBOX_MIN_X. Likewise you do this for parameter BBOX_MIN_Y.

Tip: The included sample scripts are a good starting point for getting familiar with parameterization.

Defining map request parameters

After adding parameters to a map request you have to define the parameter.

Mapviewer2test supports 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.

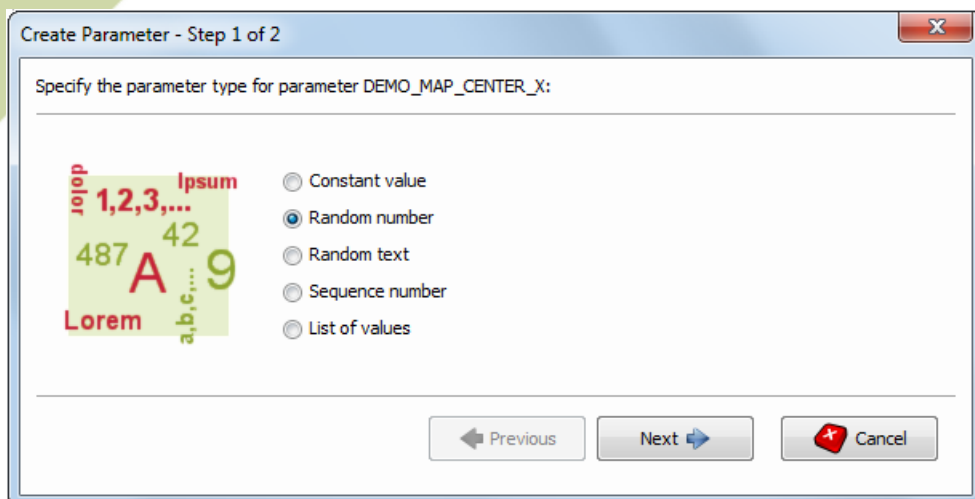
The random numerical value parameter type will be the most popular parameter type.

For each parameter added to the map request the parameter wizard will open to help you to define in two steps the parameter type and corresponding parameter properties.

Parameter Type

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

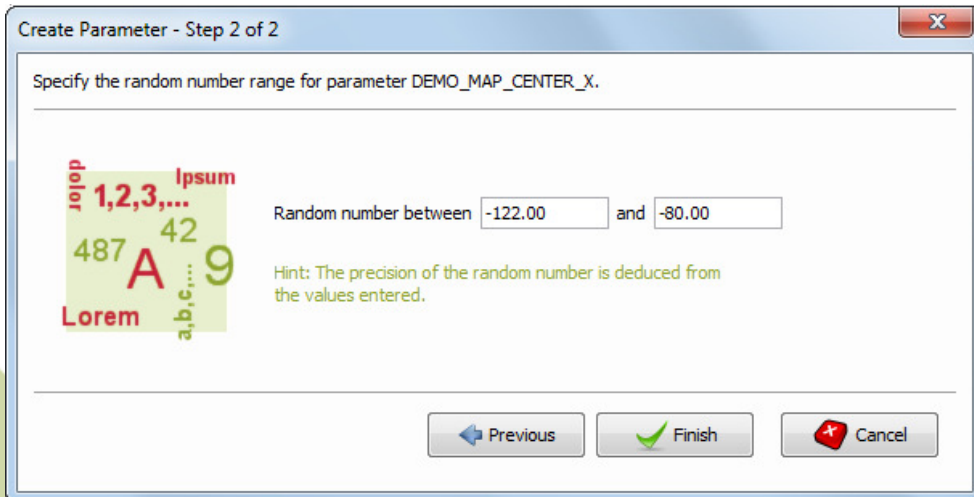
- Constant value
- Random number
- Random text
- Sequence number
- List of values (LOV)
- Equal to another parameter value



Parameter Properties

The next step is to define the parameter properties for the parameter type of step 1.

For a *random number* you should define the minimum (inclusive) and maximum (inclusive) for the random number parameter. Note that the precision (that is, digits to the right of the decimal point) of the random values during the test are deduced from the values entered.



Create Parameter - Step 2 of 2

Specify the random number range for parameter DEMO_MAP_CENTER_X.

Random number between -122.00 and -80.00

Hint: The precision of the random number is deduced from the values entered.

Previous Finish Cancel

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.

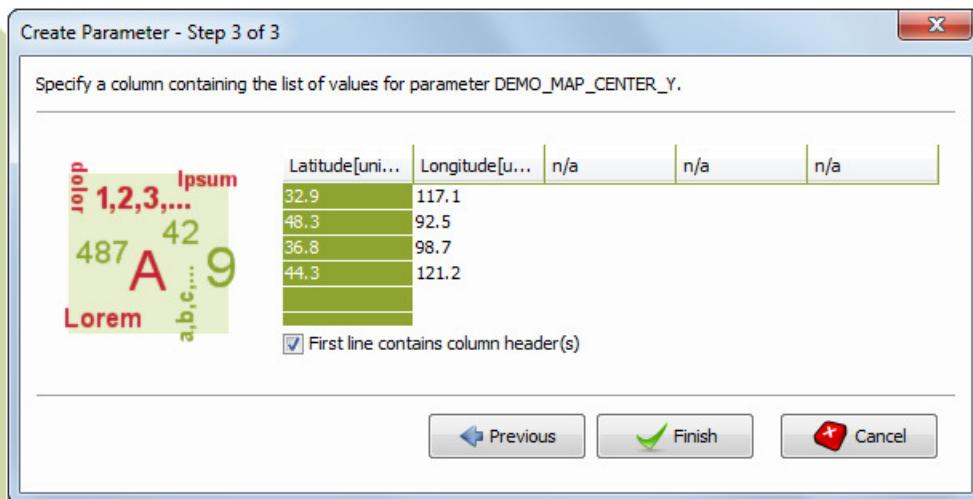
For a *sequence number* you have to provide the start and increment value. Then for each virtual user the parameter value will be increased by the increment value.

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.
 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:

```
Latitude[unit="deg"];Longitude[unit="degrees west"]
32.9;117.1
48.3;92.5
36.8;98.7
44.3;121.2
etc...
```

The next wizard step after selecting the lov file is to select the preferred column. For following screenshot the column containing the latitude is selected.



➤ Running a performance test

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

Make sure that the Oracle MapViewer service URL points to the correct location (see *Preparing to use Mapviewer2test*)

The service URL will be used as the target environment for the performance test.

Defining the test scenario

A typical test scenario consists out of one or more scripts, i.e. map requests.

You can add a map request to the test scenario by enabling the map request in the Test Suite. Note that all parameters of the map request should be defined and that the performance baseline is established before you can add the map request to the test scenario.

Warning: *Starting many map requests within a small time frame will definitely overload the application server. Schedule a reasonable load behavior!!*

Defining the test goal

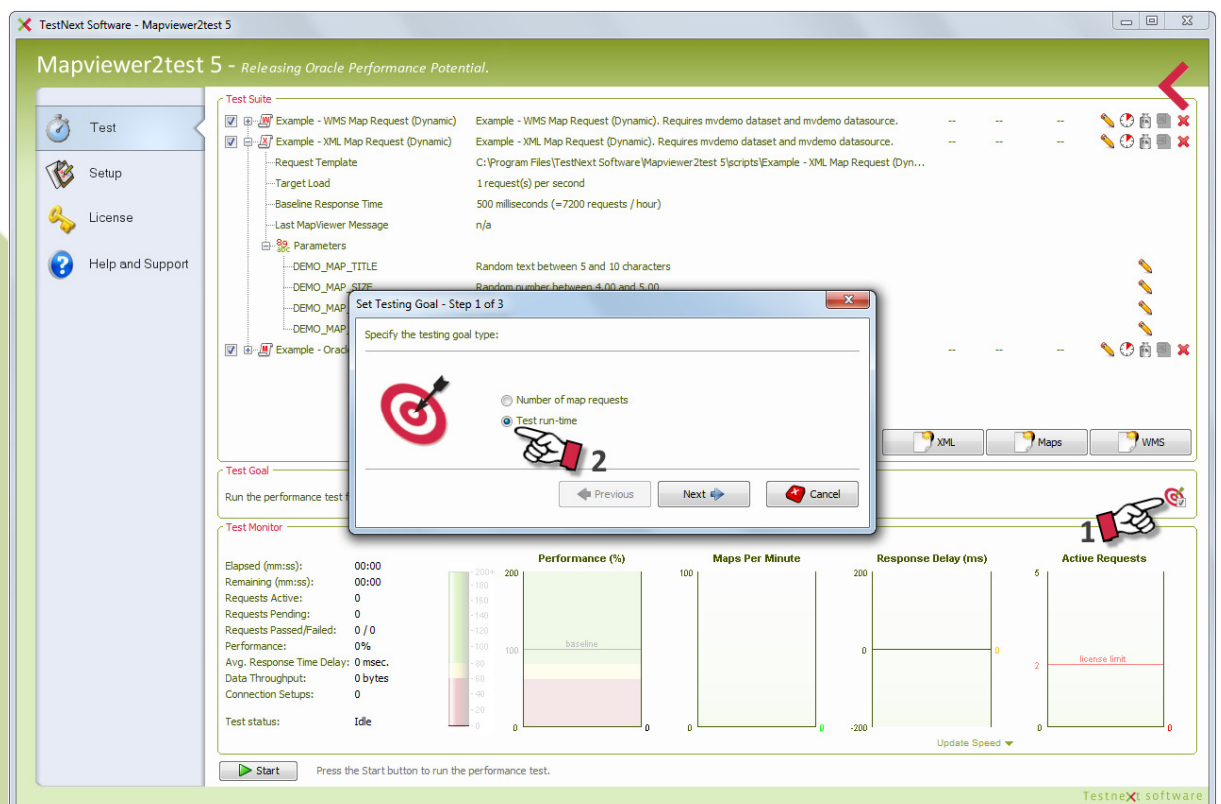
The next step is to define the test goal. Press the test goal button (1) in the Test Goal pane to invoke the wizard.

In three steps you can define the testing goal.

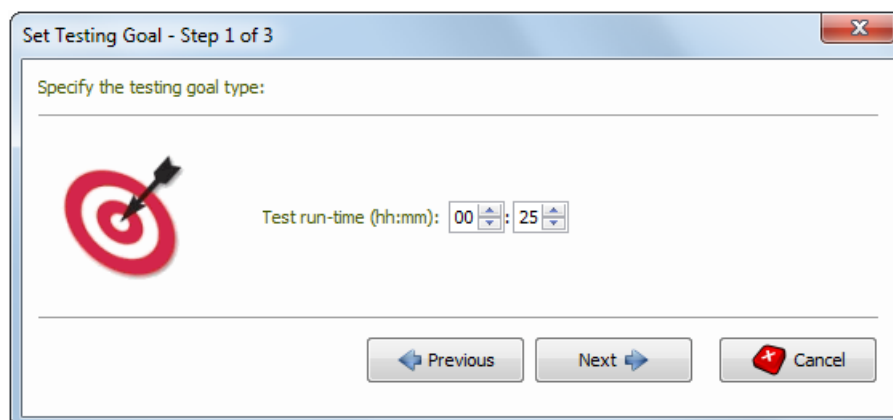
First, you have to specify the goal type. Then the total number of map request to send or the test duration and finally the maximum number of active (concurrent) map requests.

Mapviewer2test has support for two test goal types. A test based on the *test runtime* and a test on the *number of map requests* (2).

Select the preferred test goal and press Next.

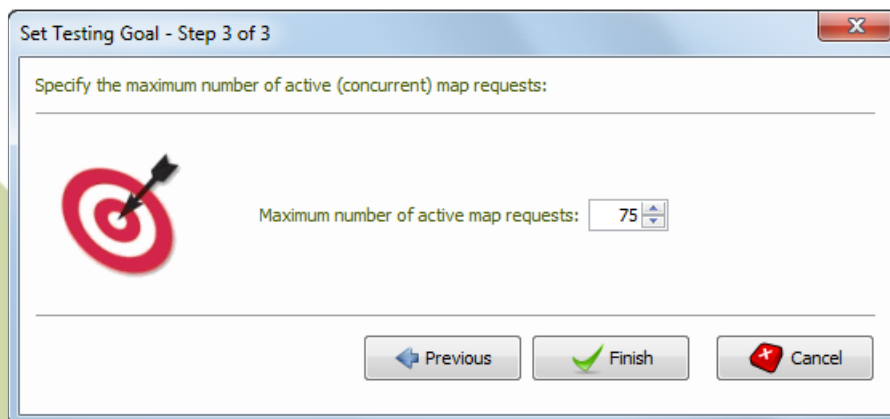


Note that when either test goal is reached, the system will not terminate immediate but wait for the active map requests to stop gracefully. This is the ramp-down phase of the test.



The last step is to define the maximum number of active requests for the test. Setting this value to 75 means that at any moment no more than 75 concurrent map requests will be send to the server irrespective of the target load at script level.

Note: Mapviewer2test allows you to purchase a license to send as many map requests as you need to effectively test your Oracle MapViewer service.
For the (full functional) evaluation version, however, you are licensed to set a maximum of 2 active map requests only.



Before you start the test you should familiarize yourself with the Test page. While the test is running you can see how Oracle MapViewer 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*, *Test Goal* and the *Test Monitor*.

The Test Suite section lets you define the test scenario and view the run status, passed map requests (1) and failed map requests (2) together with the performance for all *passed* map requests (3).



The Test Goal section shows the current testing goal.

The Test Monitor (4) shows five online (graphical) performance monitors together with the overall performance statistics for the test scenario.

The following performance monitors are displayed during the test:

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

Maps Per Minute – shows the actual and target number of generated maps per minute.

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

Active Requests - shows the number of simultaneous running map requests.

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 map requests.

Requests active – The number of currently running map requests.

Requests pending – The number of pending map requests.

Requests passed/failed – The total number of passed map requests and the total number of failed map requests.

Performance – The actual performance.

Avg. response time delay – the average difference between the baseline response time and the actual response time in milliseconds for each map request over the last 5 seconds.

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

Connection Setups – total number of socket connections.

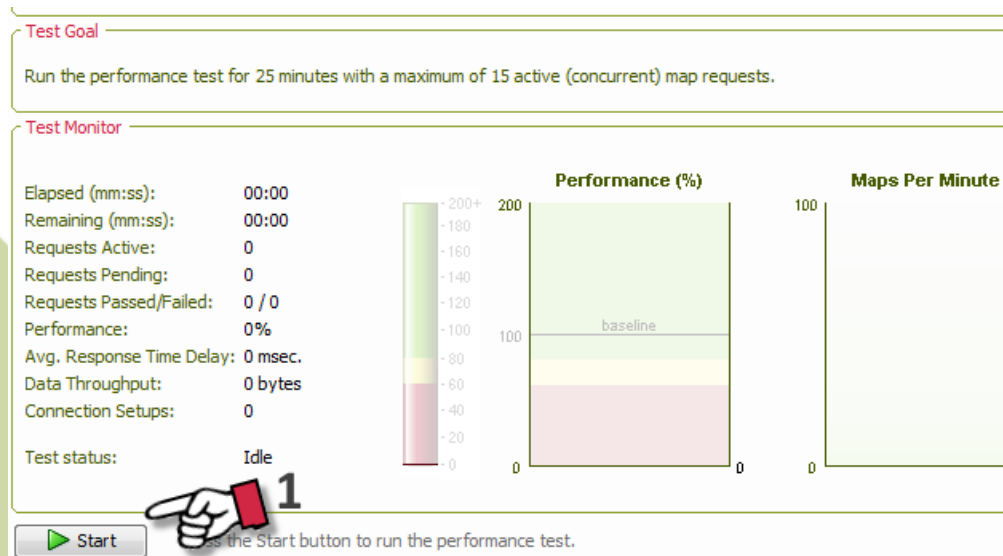
Status – current status of the load test: running, stopping, aborting and stopped.

Now you are ready to actually run the performance test.

Mapviewer2test 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 Mapviewer2test software.

Before running the test in batch mode you should first define your test scenario using the graphical interface.

For running the performance test in interactive (GUI) mode you have to press the Start button on the Test page (1).

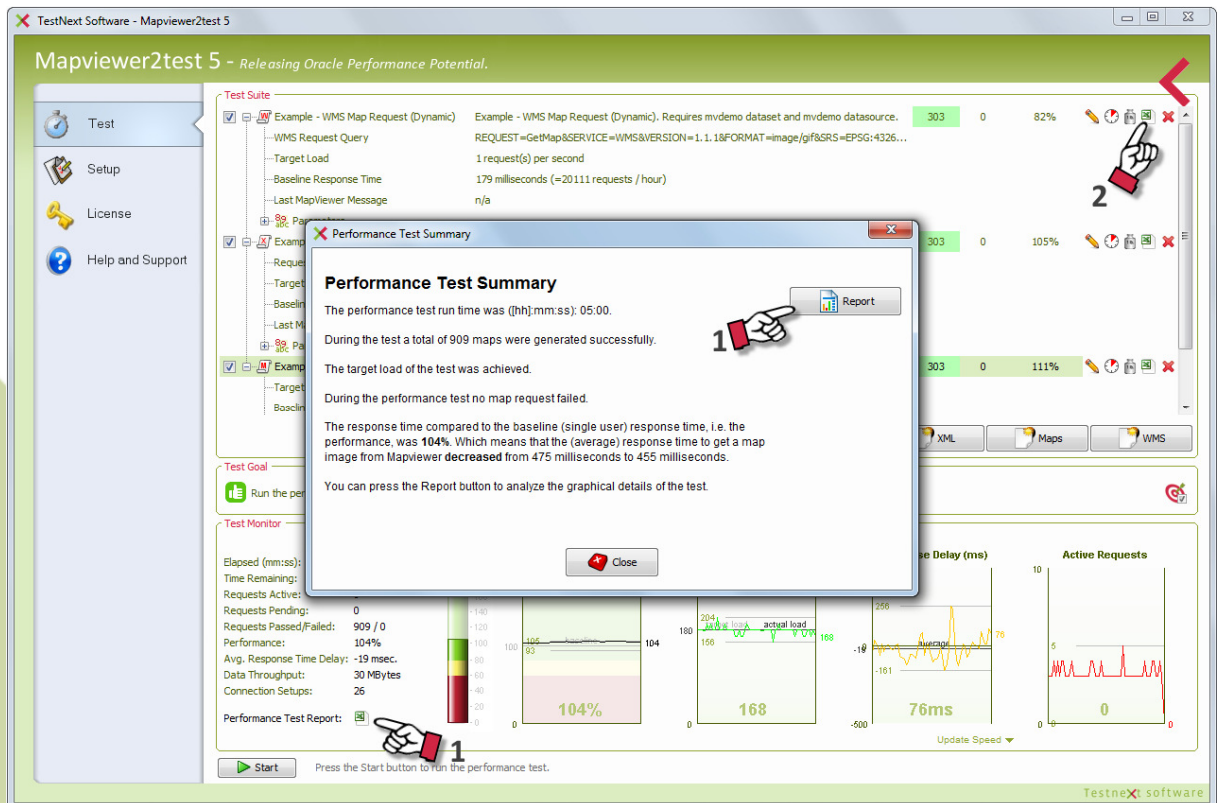


Note that you can stop and abort the test at any time. If you stop the test no more new map requests are sent but the system will wait for the active map requests to complete gracefully.

However, when you abort the test no more new map requests are started and the active map requests are terminated forcefully.

➤ Analyzing the results

At completion of the performance test, Mapviewer2test will automatically generate comprehensive graphical Excel workbooks using the captured performance statistics and show you a test summary.



To view the overall test report select the Report button (see 1). Mapviewer2test will automatically launch Excel and open the generated report for you.

To view the test report for an individual script select the Excel button (see 2).

A generated Excel workbook (spreadsheet) will contain the following worksheets:

- Performance (*graph*)
- Response Time (*graph*)
- Cumulative Response Time (*graph*)
- Response Time Delay (*graph*)
- Data Throughput (*graph*)
- Network Connection Setup Time (*graph*)
- Test Load (*graph*)
- Map Requests (*graph*)
- Response Time Distribution, script only (*graph*)
- Individual Response Times, script only (*graph*)
- Statistics (*data*)
- Response Times, script only (*data*)



All performance reports are physically saved to the output directory (see Settings Page). The performance reports have the following naming convention, where <hhmiss> is the start time of the performance test:

Overall test report: mapviewer2test_<yyyymmdd>-<hhmiss>.xls

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

During the performance test, Mapviewer2test 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.

mapviewer2test.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]

mapviewer2test.log (log directory) – this file contains runtime information about events during the test scenario. It shows among others the runtime parameter values, etc.

mapviewer2test.csv (output directory) – this file contains whole scenario performance statistics captured by Mapviewer2test 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 Mapviewer2test 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.

The graphical Excel workbooks are generated from the csv files mentioned.

> What it means

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

Active Requests – Total number of running map requests.

Pending Requests - Delayed requests due to active request limit.

Passed Requests – Total number of passed map requests.

Timed Out Requests – Total number of map requests that timed out.

Failed Requests – Total number of failed map requests.

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

Actual Performance – The relation between the cumulative response time during the test and the cumulative baseline response time for the last 5 seconds.

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

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

Overall performance – The overall performance

Target load – Scheduled target load in maps per minute.

Actual load – Established load during the test in maps per minute.

Generated Map Images – Generated map images per minute.

Connection Setups – 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 times of the map requests during the test.

Cumulative baseline response time (sec) - Sum of the baseline response times for the map requests.

Cumulative Gross Response Time (sec) – Sum of only the positive response times of the map requests during the test.

Response time delay per map request (msec) – Average difference between the baseline response times and the actual response times during the test over the last five seconds.

Average net response time delay per request (msec) – Average difference between the baseline response times and the actual response time during the test for all map requests (in milliseconds).