



Load testing with **WAPT**: Quick Start Guide

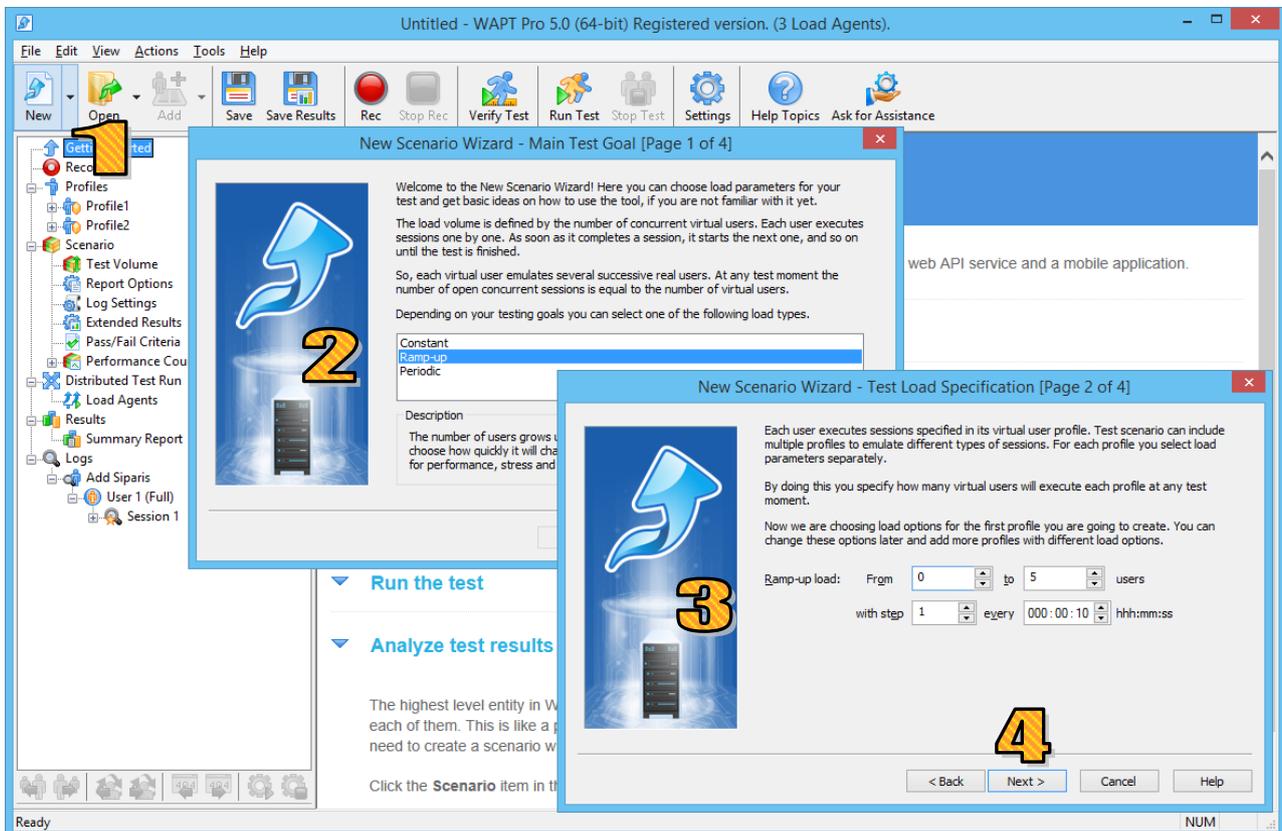
This document describes step by step how to create a simple typical test for a web application, execute it and interpret the results. A brief insight is provided on the advanced features of WAPT, extension modules and the x64 Load Engine.

Part I: How to create and run a test

Test scenario

We will start with creating a test scenario. It defines the general parameters of a test, such as the number of virtual users, type of load and test duration.

1 Click the **“New”** button on the toolbar. This will launch the **New Scenario Wizard**.



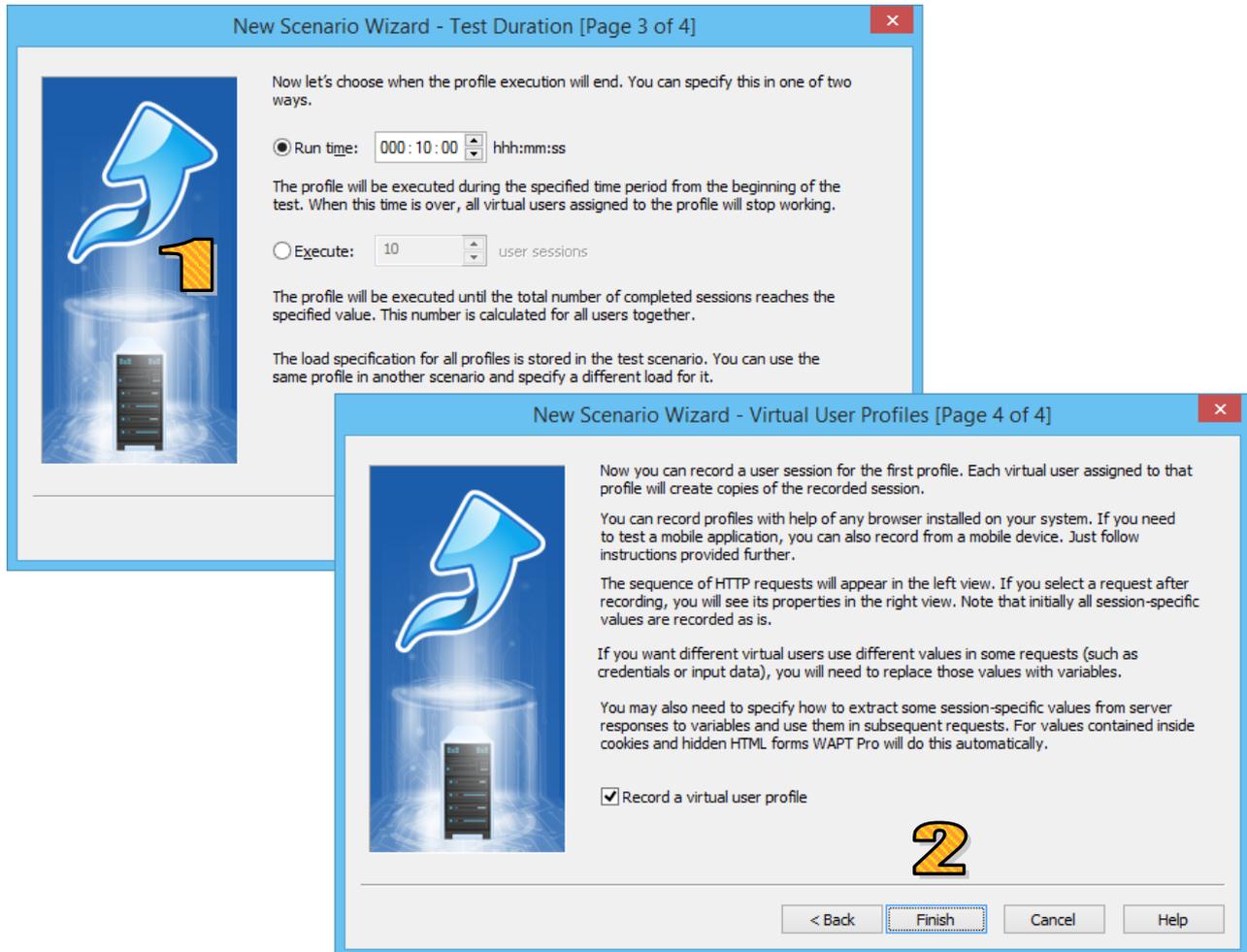
2 In this example we will create a simple performance test. Choose the "Ramp-up" option on the first page of the Wizard and click the **“Next”** button to continue.

3 On the second page of the Wizard you can specify the basic parameters for the ramp-up load. Here you choose how fast the number of virtual users will grow throughout the test. This will let you compare the performance of your web application in different test periods depending on the increasing load. You can keep default values for now, because everything you select in the Wizard can be adjusted later on the **“Test Volume”** page.

4 Click the **“Next”** button to proceed to Page 3.

Test duration options

1 On Page 3 you can either specify the exact test duration or set the total number of sessions to execute by all virtual users before the test ends. Note that the number of sessions executed in the test is not the same as the number of concurrent virtual users. Each user executes its sessions one by one sequentially. As soon as a session is finished, it starts a new one. This guarantees that the load is not reduced after a session is over. The load volume in each test period depends on the number of virtual users working concurrently.



2 The last page of the Wizard contains some important hints on how to create a test and interpret its results.

Click the “**Finish**” button to proceed to the design of the user sessions that will be emulated in the test.

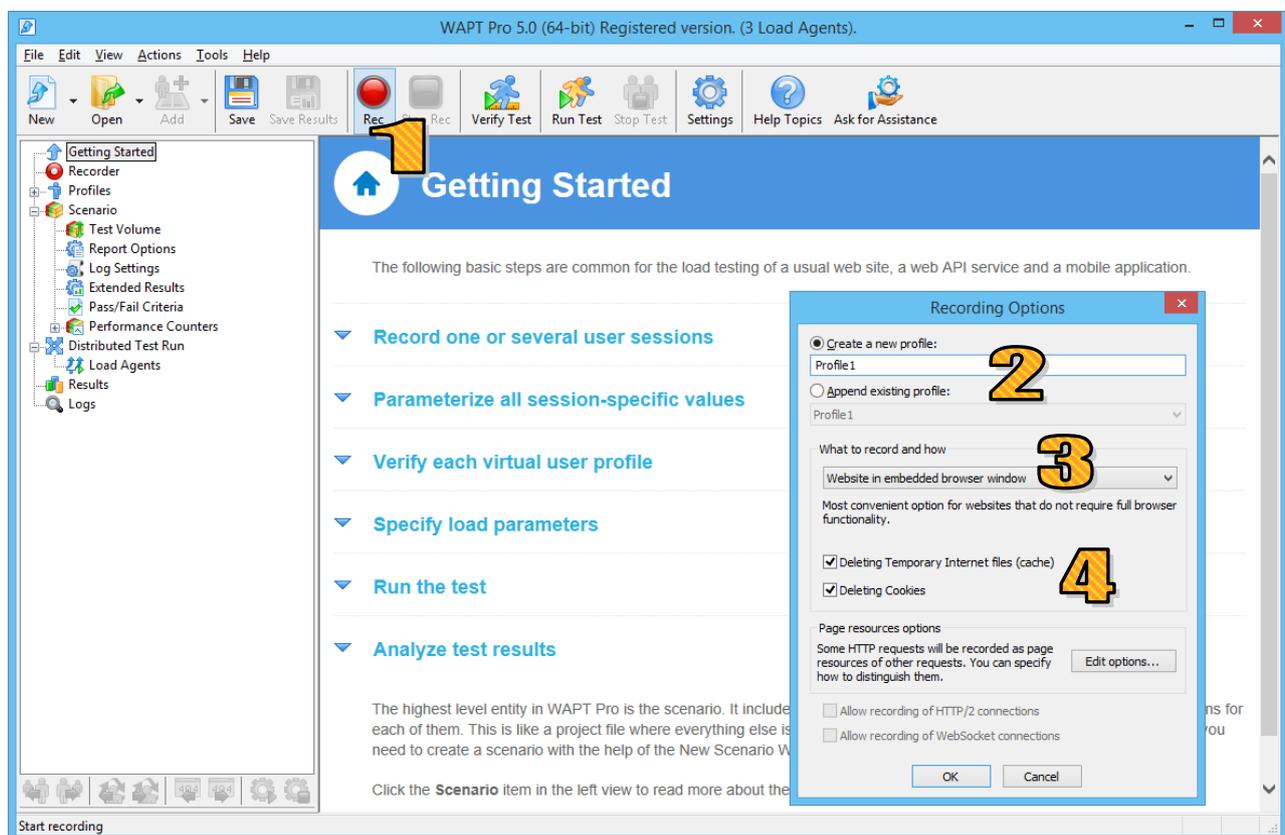
Virtual user profiles

WAPT uses virtual user **profiles** to specify how sessions are emulated. Other testing tools use scripts for the same purpose, but profiles are easier to understand and to work with. Each profile represents a sequence of session steps. A basic step is a single HTTP request. To facilitate your work with profiles WAPT arranges requests on two levels. **Page requests** perform transactions and load main pages of your application. They can also have associated **resource requests** that load additional media files required to display each page.

You can use multiple profiles in a test. The number of virtual users and other load parameters mentioned earlier are specified for each profile separately. Profiles are usually created by recording typical user sessions with your application.

1 After you complete the New Scenario Wizard, WAPT will automatically proceed to recording a profile. In the future you can click the “**Rec**” button on the toolbar to record another one. This will open the “**Recording Options**” dialog.

2 Choose a name for your profile.



3 You can either use the embedded browser window, or choose an external browser. If you want to record a mobile application, you also have an option for that.

4 It is strictly recommended to delete browser cache files and cookies before starting the recording. This is required because any stored data related to your application may affect the way the current session is performed. A session essentially based on the previous user activity cannot be correctly reproduced as a separate one in the test. If you leave the corresponding options checked, WAPT will perform this cleanup automatically. Note that it may take a while when done for the first time.

Recording a user session

By default WAPT will use the embedded browser window for recording.

- 1 Type or paste the URL of your web application to the address bar and click the “Go” button or press Enter. Note that it is very important to start with the landing page URL, not with some internal application page. If you already have some page opened in that browser window, click the “Clear Page” button to remove it and start over.
- 2 As you navigate through the web site inside the browser window, WAPT will record all HTTP requests generated by your application. You will see them appearing in the left view inside the “Recorder” folder.
- 3 During recording all requests are placed on the same level and you cannot edit them. However you can add bookmarks to separate transactions. They will be preserved after the recording is transformed to the profile structure.



- 4 After you complete the session, click the “Stop Rec” button on the toolbar. WAPT will process the recorded content. It will separate resource requests, such as images, .js and .css files from the page requests to put them on the inner level in the profile. It will also analyze values contained inside requests to convert some of them into variables. This will finally result in a new profile appearing in the left view under the “Profiles” folder.

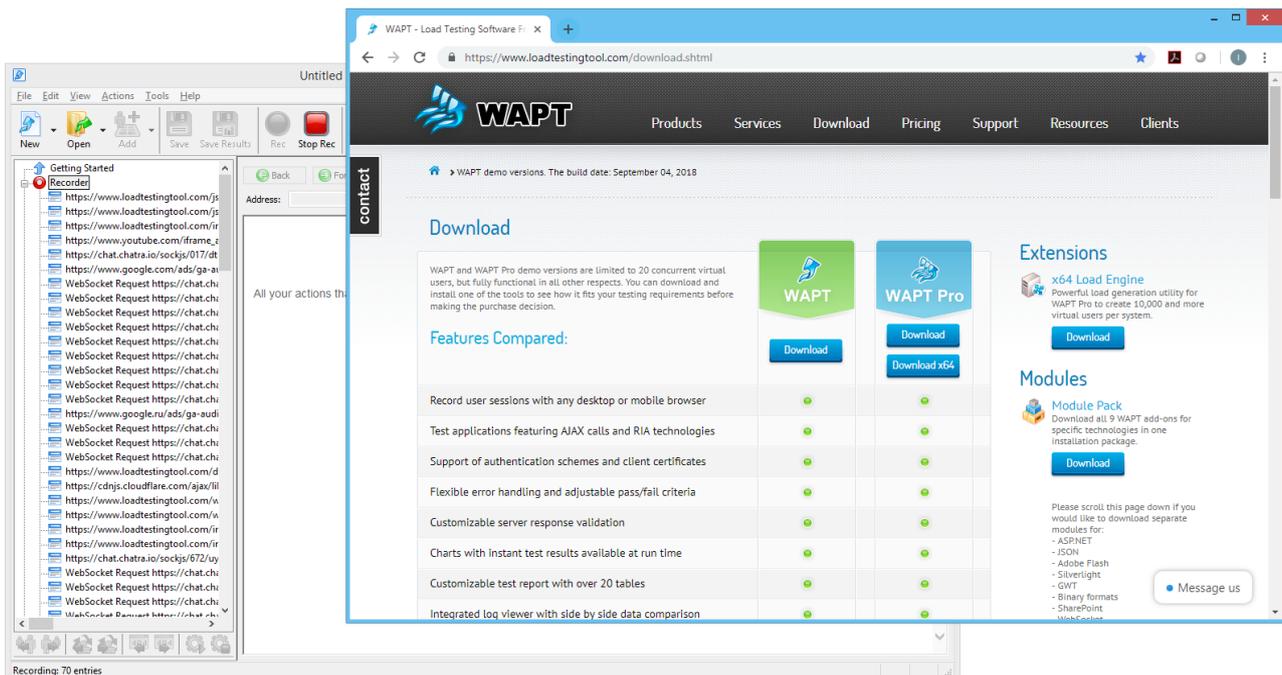
You can record several more profiles in a similar way, or proceed with just one.

Recording with an external browser

While the embedded browser is a convenient tool for recording, its functionality is limited. In the following cases you will need to use an external browser instead:

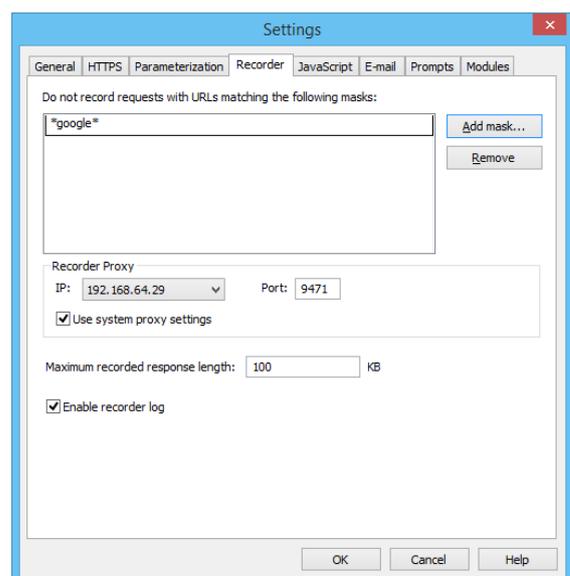
- Your web application does not work inside the embedded window as expected;
- You would like to use HTTP/2 or WebSocket protocol in the test;
- Your application is compatible only with a particular browser.

You can select an external browser for recording in the **“Recording Options”** dialog. WAPT will run the selected browser automatically. You can keep it on top of WAPT window or maximize.



It is recommended to close all other browser tabs and windows at the time of recording. Leave only one where you open your application. Otherwise you will also record all background requests generated by other applications.

When you use an external browser for recording, WAPT needs to temporarily change your system proxy settings, which will affect all applications using them. Some browsers also generate additional HTTP requests on startup to check for updates and send information to their vendors. Those requests will also be recorded. To avoid including them in the profile you can add all such websites to the skip list on the **“Recorder”** tab in WAPT settings. This list is also useful for filtering requests to third party services utilized by your application, but not required for the session emulation.



Recording a mobile application

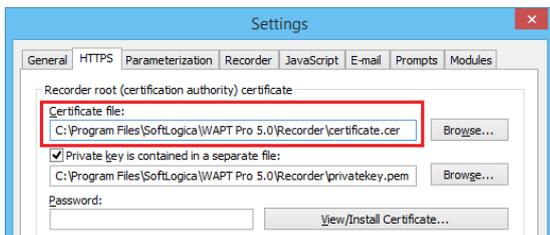
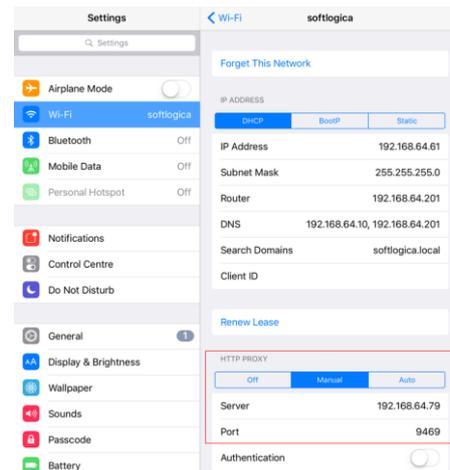
If you want to test a mobile application backend with WAPT, you will need to have that application running in a device that you can connect to the same LAN with the WAPT system. This is usually done over the wireless connection. The device is only required for recording. You will not need it when working with the recorded profile and running tests.

To record a mobile application session, choose the **“Mobile Application (manual proxy configuration)”** option in the **“Recording Options”** dialog. Since WAPT cannot access your device automatically, you will need to configure couple things by yourself.

First, you will need to specify WAPT system as the HTTP proxy for the active Wi-Fi connection on your device. The IP address and port number should be taken from the message shown by WAPT when you start recording. Those options are configurable on the **“Recorder”** tab in WAPT settings, so you can specify any port and IP address available on the system before starting recording.

WAPT Pro works as a proxy between your browser and the target web site. You can use any browser running on this system or any other system on your LAN, including mobile devices connected via Wi-Fi. You only need to set the browser proxy settings, so that it would use the following HTTP proxy when connecting to your web site:

Server name (IP): 192.168.64.79
Port: 9469

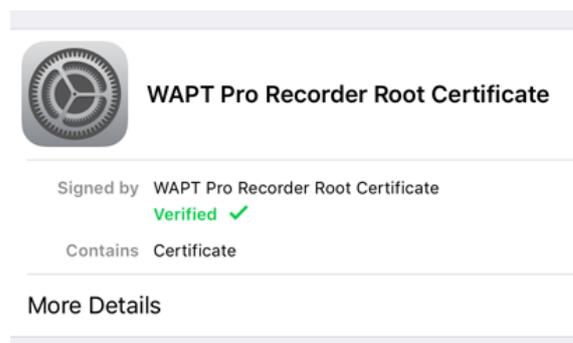


Next, you will need to install the recorder certificate to your device. This is required to enable decoding of HTTPS requests. You can find that certificate in the WAPT installation folder. The full path is provided on the **“HTTPS”** tab in settings.

The easiest way to transfer the certificate file to the device is to send it as an attachment to an email account accessible therefrom.

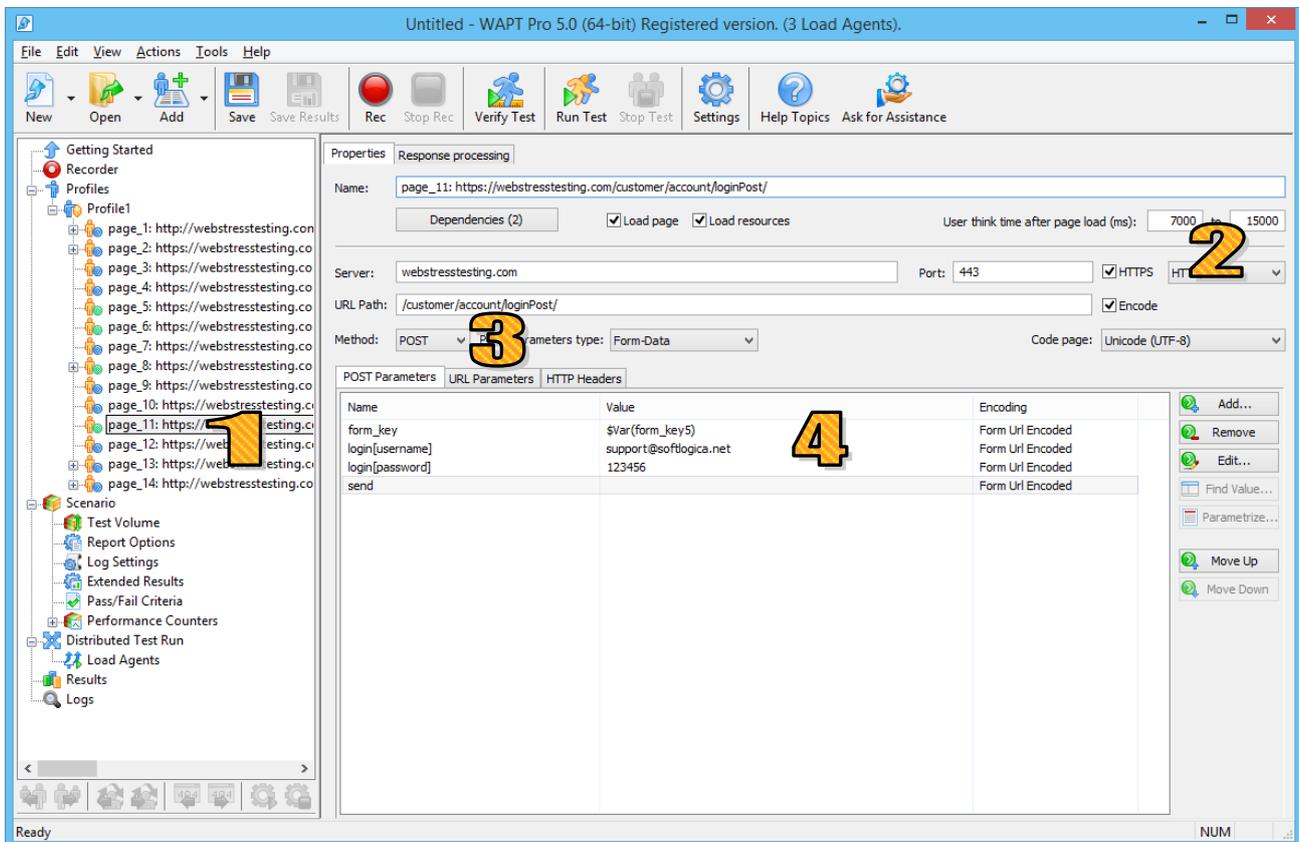
The installation procedure depends on the OS version, but you can do this with any iOS or Android device. A very detailed step by step instruction is available on WAPT website here: <https://www.loadtestingtool.com/help/performance-test-for-mobile-application.shtml>

If you have problems with a custom OS version or cannot follow any step in the instruction, you are always welcome to contact our technical support team.



Properties of a request

1 Once you have recorded a profile, you can expand it to the sequence of requests in the left view. Select a request there to see its properties in the right view on the “**Properties**” tab.



2 Here you have a number of modifiable options. For example, you can adjust think time, which is a pause made after executing the request. The duration of that pause is randomized within the specified range. By changing it you can adjust the pace of the profile execution.

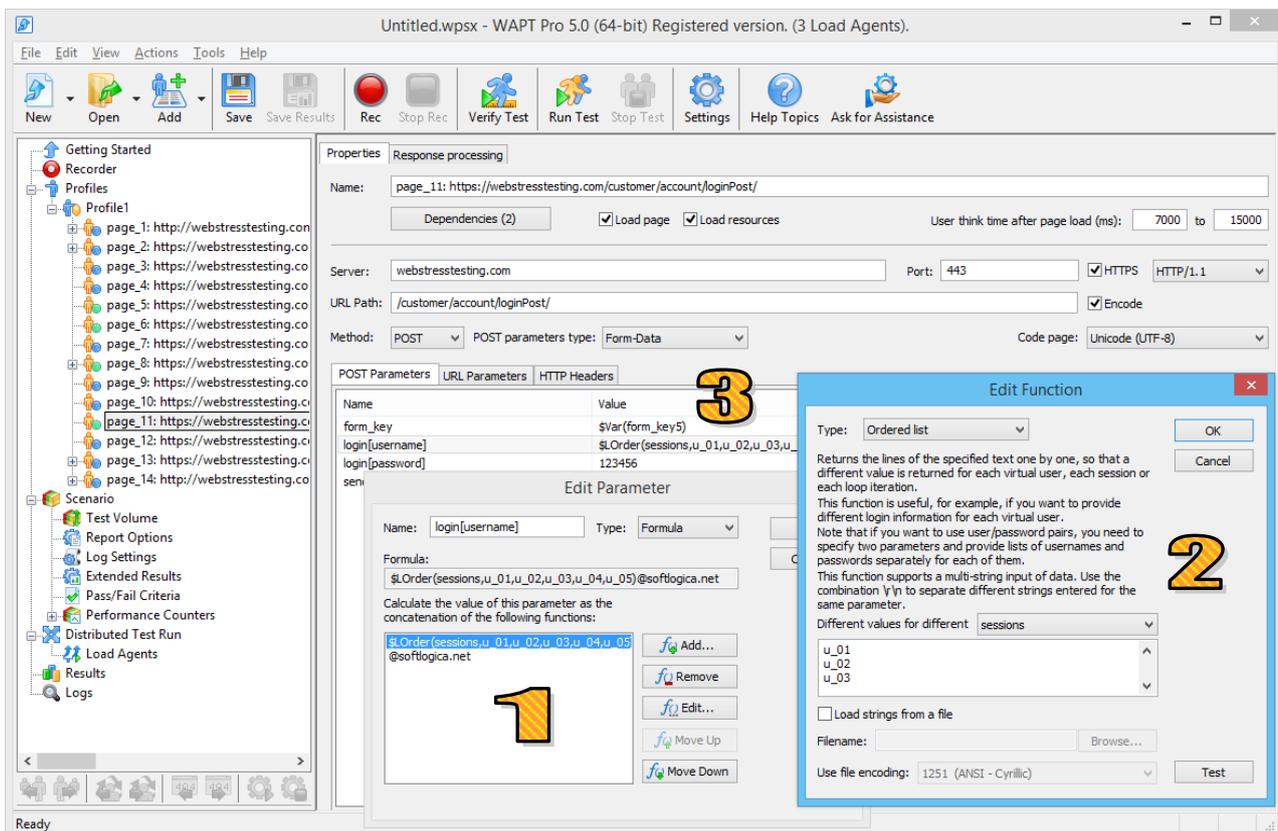
3 A very important property is the HTTP method, which is usually GET or POST. You never need to change it from one to another, but you can see that its selection affects other options. POST requests usually contain values that are sent to the server in order to perform transactions. This may be a login request containing user name and password, a web form submission with filled in data or other application-specific action requiring input values.

4 An HTTP request may include parameters. In GET requests they are specified inside the request URL after the “?” character. POST requests have another list of parameters that are passed inside the request body. In WAPT you can edit both lists for each request on the “**URL Parameters**” and “**POST Parameters**” tabs. You can double-click a parameter to edit its properties in a separate dialog. If you click the parameter line inside the “**Encode**” column, you will be able to change the encoding scheme.

Session-specific values

There are two reasons why you may need to modify the initially recorded values of parameters for a request. First, you may want to provide different values for use in sessions emulated in the test. For example, you can specify multiple user accounts instead of the one used to record the original session. This is called **“Parameterization”**.

1 The value of each parameter is specified as a concatenation of a list of functions inside the **“Edit Parameter”** dialog.



2 Some of these functions return session-specific values. For example, the **“Ordered List”** function returns values one by one picking them from a predefined list, which you can specify right there or load from a file. Note that if you need to use a value multiple times within the same session, it is preferable to assign it to a variable. This will let you use that variable in all places where you need it instead of repeating the same function specification.

3 Second reason why parameters need to be modified is that in most applications certain values are initially generated by the server and transferred to the client inside responses. At further steps of the same session those values must be sent back to the server. Otherwise it will not work correctly. Those values are usually passed as parameters of subsequent requests or even as parts of their URLs. WAPT creates variables for the majority of such values automatically, so very often you can see the **“\$Var(..)”** notation in a parameter. This means that the actual value is contained in the variable specified inside the parentheses. That variable can have a different value in each emulated session. Variables can be used in almost any request property, including server name, URL, parameters and headers.

Extracting values from responses

Variables are created and assigned values on the “**Response processing**” tab. It is available for any request. That tab serves three important purposes.

1 It lets you review the originally recorded request and response to it. This information is not used anyhow to construct requests in the test sessions. You cannot edit the content of the tabs in the middle of that view, but you can search for any values there, including ones you may need to extract to variables. Note that in case the server redirected request one or more times, you can see the content of each redirect by selecting it in the list above the tabs.

The screenshot shows the WAPT Pro 5.0 interface. The main workspace is titled 'Response processing'. It contains a 'Variables' table with the following data:

Name	Value
form_key2	\$Hidden(form_key,1)
key	\$Search(hidden value=',' />,1)

The 'Recorded HTTP request and response' section shows the following HTML snippet:

```

<div class="col2-set">
  <input type="hidden" name="form_key" value="ltY7lwWjMFMgie4" />
  <div class="col-1 new-users">
    <div class="content">
      <h2>New Here?</h2?
    </div>
  </div>
</div>

```

The 'Validation Rules' section has the following settings:

- Valid only if response body Contains
- Valid only if response time degrades less than 2 times compared to the baseline
- Apply profile validation rules
- Ignore HTTP errors
- Do not follow redirect

2 The upper view contains the list of variables assigned in each emulated session after completing the request. The most commonly used function for extracting values from responses is called “**\$Search**”. It finds values by the bounding text strings. This works because in most cases it is possible to specify bounds that remain unchanged from session to session while the value between them is unique in each session.

Note that you cannot use a variable before assigning it. A very common mistake is to extract a value from a response and attempt to use it in the same request.

Variables are maintained independently inside each emulated session for the whole its duration. You cannot pass values from one session to another inside variables, but you can reassign the same variable multiple times in different requests.

3 Here you also have a number of additional useful features that tell WAPT what to do with the server response. For example, you can make WAPT identify application-specific errors even if they are not reported through the HTTP response codes.

Test verification

After you have parameterized all session-specific values with help of functions and variables you can verify the profile to check that it is capable of emulating sessions correctly.

1 Click the “**Verify Test**” button on the toolbar. WAPT will let you select profiles for verification, and then it will execute each of the selected profiles one time.

Test execution parameters:
 Test status: finished
 Test started at: 26.09.2018 17:55:19
 Scenario name: Untitled.wpsx
 Test run comment:
 Test executed by: Iis (ANTARES-WIN8)
 Test executed on: localhost

Response codes

Code	Request	Pages	Hits
Profile2	-	9	87
200 OK	Profile2.All	8	80
200 OK	Profile2.page_1: http://webstresstesting.com/	1	37
	Profile2.page_2: https://webstresstesting.com/customer/account/login/	1	37
	Profile2.page_3: https://webstresstesting.com/customer/account/create/	1	1
	Profile2.page_4: https://webstresstesting.com/customer/account/login/	1	1
	Profile2.page_5: https://webstresstesting.com/customer/account/loginPost/	1	1
	Profile2.page_6: https://webstresstesting.com/customer/account/loginPost/	1	1
	Profile2.page_7: https://webstresstesting.com/customer/account/create/	1	1
	Profile2.page_8: https://webstresstesting.com/customer/account/createpost/	1	1
302 Found	Profile2.page_5: https://webstresstesting.com/customer/account/loginPost/	1	1
	Profile2.page_6: https://webstresstesting.com/customer/account/loginPost/	1	1
	Profile2.page_7: https://webstresstesting.com/customer/account/create/	1	1
	Profile2.page_8: https://webstresstesting.com/customer/account/createpost/	1	1
404 Not Found	Profile2.page_8: https://webstresstesting.com/customer/account/createpost/	0	1
Network error	Profile2.page_8: https://webstresstesting.com/customer/account/createpost/	0	1
Other errors	Profile2.page_9: https://webstresstesting.com/customer/account/logout/	1	1

Report generated at 26.09.2018 17:55:30 by WAPT Pro 5.0 (64-bit) © SoftLogica 2018

2 When the verification is finished, WAPT will show a report containing status codes for each request. If you see any errors here, this may indicate a problem, but it is recommended that you also check if the same error occurred in the originally recorded session. For example, the 404 error is often reported because of a broken link, which may be a minor functional problem.

Response codes starting with “3” (like 302) are often mistaken for errors, but they are regular HTTP redirects processed by WAPT automatically. If you see the 401 code here, this means that the server requires authentication. So, you should provide credentials in the profile properties. After that the same request will still produce the 401 code, which is normal, but it will be followed by the “200 OK” code.

If you see a “Network error”, this means that WAPT could not connect to the target website. You should check that your network configuration permits direct connection to it.

3 The last column contains the number of hits, which include requests to page resources.

4 If you see any errors in the verification report, you should proceed to a more detailed analysis of the session steps listed in the log. In fact, even if you do not see any errors, you are still recommended to review the logs to make sure that the server responded correctly to each request.

Log viewer

Log Viewer provides detailed information on all requests, responses, and errors occurred during a test run or verification.

1 Expand the “**Logs**” folder in the left view and select a session. It will be the only one in the tree if you browse verification log for a profile.

2 In the upper right view you will see log lines in chronological order. Items highlighted in green show requests completed successfully (basing on the status code). Similar lines highlighted in red show requests completed with errors. Note that several lines of log may correspond to a single request, which is shown in the last column. In case of redirects you will also see several requests corresponding to the same page. Only the final one will be painted green or red to indicate its status. Requests using variables have an additional “**Values of variables**” line preceding them, which may be red in case some variables could not be assigned.

The screenshot shows the WAPT Pro 5.0 (64-bit) Registered version. (3 Load Agents) interface. The left sidebar shows a tree view with 'Logs' expanded and 'Session 1' selected (marked with a yellow '1'). The main window displays a log table with columns: Time, Operation, Status, and Page. The log entries are as follows:

Time	Operation	Status	Page
17:55:30:099	Resolved ip = 192.241.135.4		page_8: https://webstresstesting.com/customer/account/cre...
17:55:30:099	Send request. connection 2		page_8: https://webstresstesting.com/customer/account/cre...
17:55:30:099	GET /customer/account/	200 OK	page_8: https://webstresstesting.com/customer/account/cre...
17:55:30:678	Completed.		page_8: https://webstresstesting.com/customer/account/cre...
17:55:30:678	Started		res_8.1: https://webstresstesting.com/js/varien/weee.js
17:55:30:678	Resolved ip = 192.241.135.4		res_8.1: https://webstresstesting.com/js/varien/weee.js
17:55:30:678	Send request. connection 2		res_8.1: https://webstresstesting.com/js/varien/weee.js
17:55:30:678	GET /js/varien/weee11.js	404 Not Found	res_8.1: https://webstresstesting.com/js/varien/weee.js
17:55:30:678	Started		res_8.2: https://webstresstesting.com/js/varien/a2.js
17:55:30:678	Resolved ip = 127.0.0.1		res_8.2: https://webstresstesting.com/js/varien/a2.js
17:55:30:678	Connection 3: disconnection		Connection 3: disconnection

The detailed view below the table shows the response body for the selected request (marked with a yellow '3'). The response is HTML code for a 'Customer Service' link. A search bar at the bottom is set to 'Customer Service' (marked with a yellow '5'). The response content is as follows:

```

</div>
</script>
</div>
<div class="links">
  <div class="block-title">
    <strong><span>Company</span></strong>
  </div>
  <ul>
    <li><a href="http://webstresstesting.com/about-magento-demo-store/">About Us</a></li>
    <li><a href="http://webstresstesting.com/contacts/">Contact Us</a></li>
    <li><a href="http://webstresstesting.com/customer-service/">Customer Service</a></li>
    <li><a href="http://webstresstesting.com/privacy-policy-cookie-restriction-mode/">Privacy Policy</a></li>
  </ul>
</div>
<div class="links">
  <div class="block-title"><strong><span>Quick Links</span></strong></div>
  <ul>
    <li class="first"><a href="
  
```

3 You can select any line and see the details in the lower part of the view. For each request you can switch between different tabs containing request and response headers and bodies.

4 You can compare any part of the request or response with the initially recorded content. This way you can find the differences and identify session-specific values. You can also check if the server produced a significantly different response, which may indicate a problem.

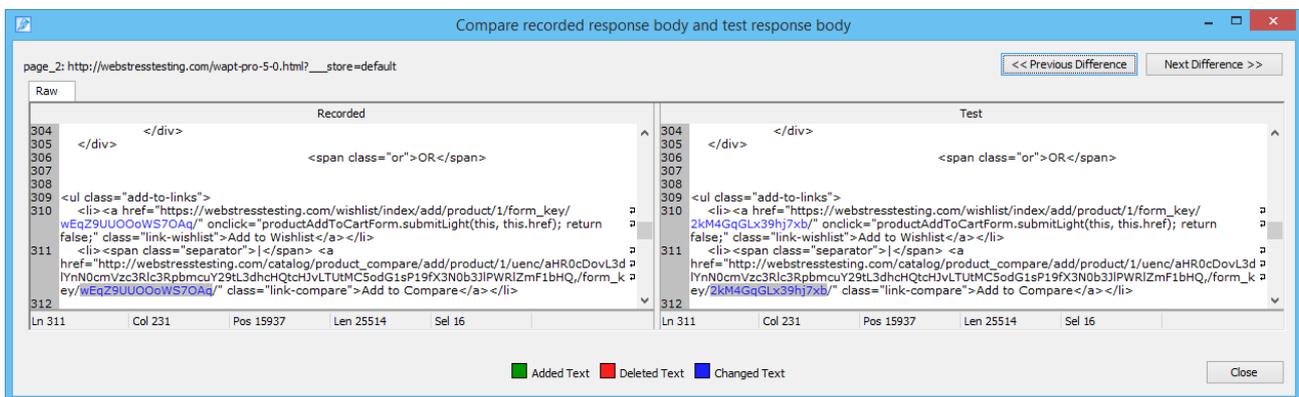
5 The useful search option is also available here.

Note that by default logging is disabled for efficiency reasons. So, if you want to get logs after a test run, you should enable this feature on the “**Log and Report Settings**” page before starting the test. You can save all log files by choosing “**File | Save Logs...**” from the menu.

Test debugging

Test verification and log viewer functionality lets you check the work of each profile step by step. This is similar to debugging a computer program. The difference is that instead of moving along with the code execution you browse the verification log. Ideally you need to check each server response in the verification session even if you do not see explicitly reported errors. Correctly emulated session should produce responses very similar to the original ones. They may differ in values that are session-specific by design; however they should not have logically different structure and meaning.

So, if you find a response with an error code or unexpected content, check the previous response and move further up the log step by step. The goal is to find the earliest response with a problem. Checking responses this way may be a tedious task. Fortunately you have the **“Compare with recorded”** button in the log viewer, which lets you automate the comparison process to a certain extent. You can see the original and verification data side by side.

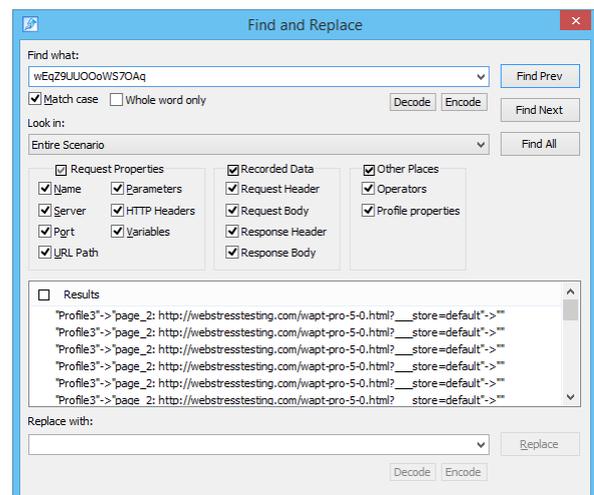


Minor differences found here may suggest which values are session-specific. Such values may require replacement with variables in subsequent requests.

After you find the first potentially incorrect response in the log, check the parameters of the corresponding request in the profile. Some of them can contain IDs requiring parameterization. Sometimes they are found in the request URL and headers.

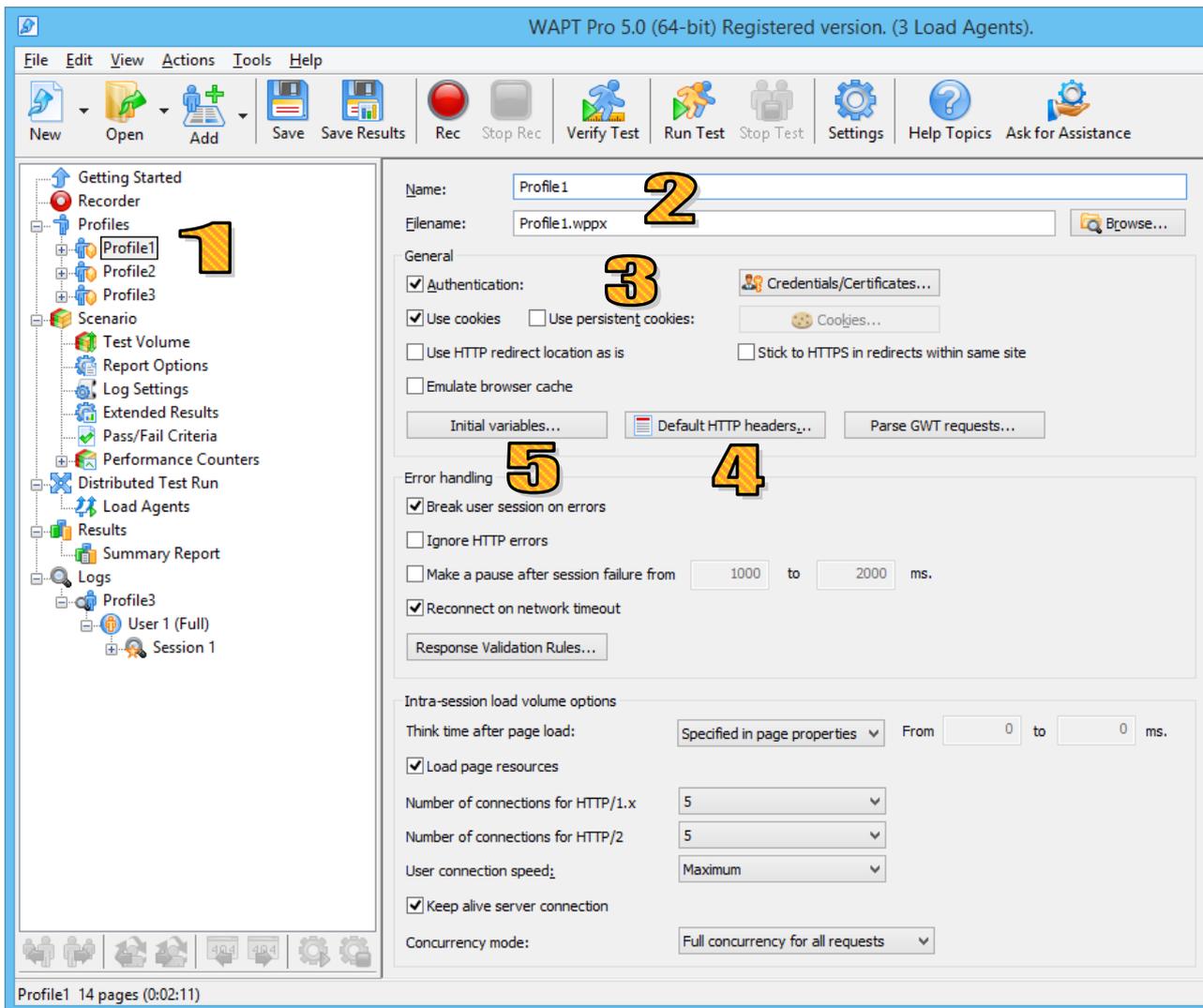
There is a very reliable way to check if a value is session-specific. You can records two sessions by following exactly the same steps and compare corresponding requests in two profiles. If some value is different, it must be parameterized. Note also that values may depend on the user account and input data, so you can try altering this as well.

In the majority of cases a session-specific value used in a request can be extracted from one of the responses to previous requests. You can use the **“Find and Replace”** dialog to find the exact place where you can create a variable. Usually such values are extracted with help of the **“\$Search”** function, which uses left and right bounding text strings to identify the value.



Properties of a virtual user profile

1 There are a number of options associated with each profile. To edit them, select the profile in the left view.



2 Each profile is stored in a file with the “.wppx” extension. You can change its name here.

3 If your web site requires authentication or a client certificate, check the “**Authentication**” option and click the “**Credentials/Certificates...**” button to specify a set of credentials that virtual users will use when running this profile.

4 Note that “**Default HTTP headers**” and “**Response validation rules**” options can be overwritten in each request.

5 If your profile uses session-specific values in the very first request, you can assign variables before the beginning of a user session. Click the “**Initial variables**” button for that.

Test load specification

After making sure that all your profiles are working correctly you can specify the load parameters for the actual test.

- 1 Select the “**Test Volume**” item in the left view inside the “**Scenario**” folder.

- 2 In the right view you can see the list of all your profiles. Check the ones you want to use in the test.

- 3 You can specify certain load options separately for each profile. Note that these options are shown for the currently selected profile (highlighted with blue selection). If you want to edit options of a different one, select it in the list.

In the above example, we have 3 profiles with different types of load (constant, ramp-up and periodic). Second profile (with the ramp-up load) is selected and its options are shown below the list.

- 4 The graph at the bottom of the page shows how the load will be distributed between profiles during the test. Each profile is shown with a different color.

- 5 In the Pro version you can also specify different load agents for different profiles. By default virtual users are allocated to agents automatically, which means that all agents execute all profiles.

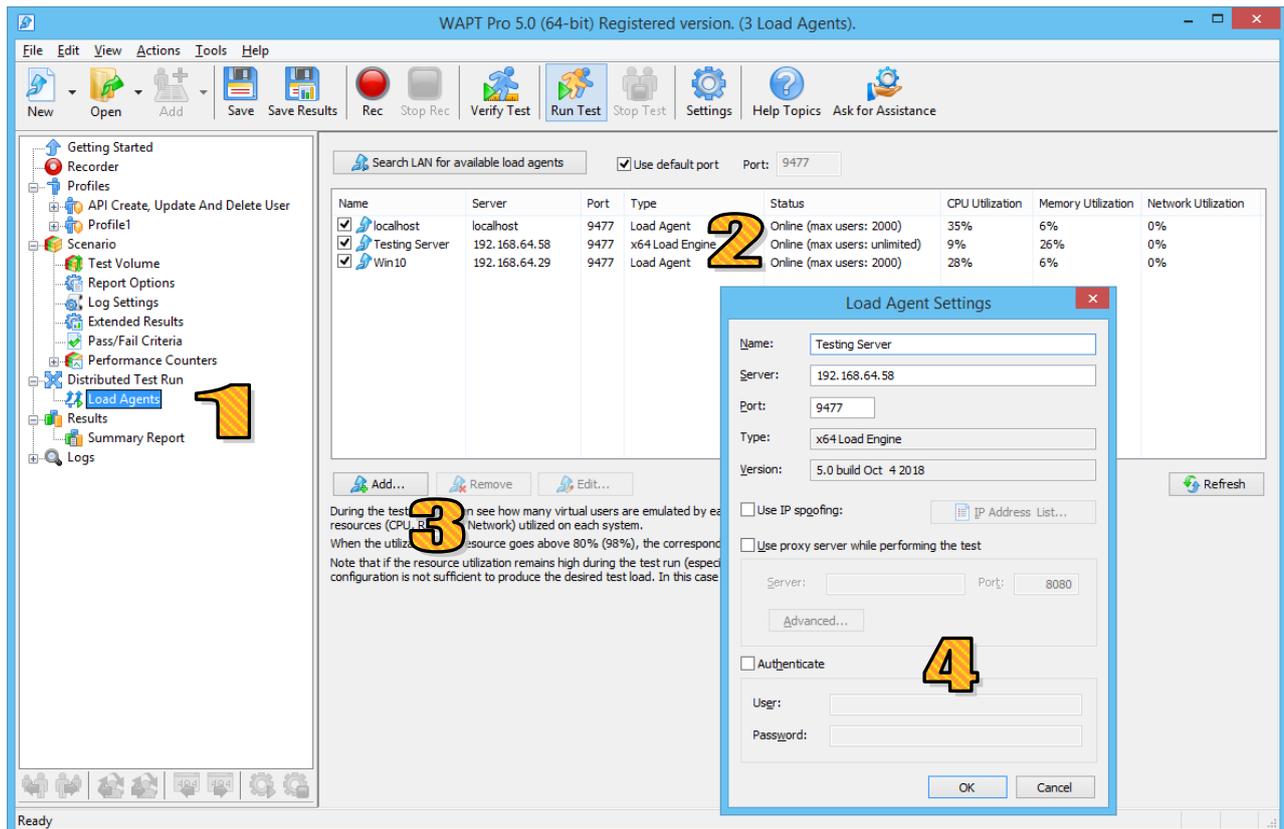
Now we have finished designing our test, so we can save it. Click the “**Save**” button on the toolbar to save your test scenario to a file. All profiles will be also saved to separate files in the same folder. Keep all these files, if you want to open the test in the future.

Load agents*

With the Pro version you can use several systems for load generation. Each of those systems must run a **load agent** or **x64 Load Engine**, which is a more powerful version of an agent.

1 Select the “**Load Agents**” item in the left view inside the “**Distributed Test Run**” folder to manage load agents.

2 One agent is installed by default with the workplace component, so you can see *localhost* in the list of agents. If you also installed agents on other systems within your LAN, click the “**Search LAN for available load agents**” button to find them.



3 If you have agents located outside your LAN, you can add them manually with the help of the “**Add**” button. The connection between the workplace component and load agents is done over TCP/IP. Make sure that your network configuration and firewall settings permit it.

4 If you select an agent in the list and click the “**Edit...**” button (or double-click an agent), you will be able to edit the agent properties in a separate dialog.

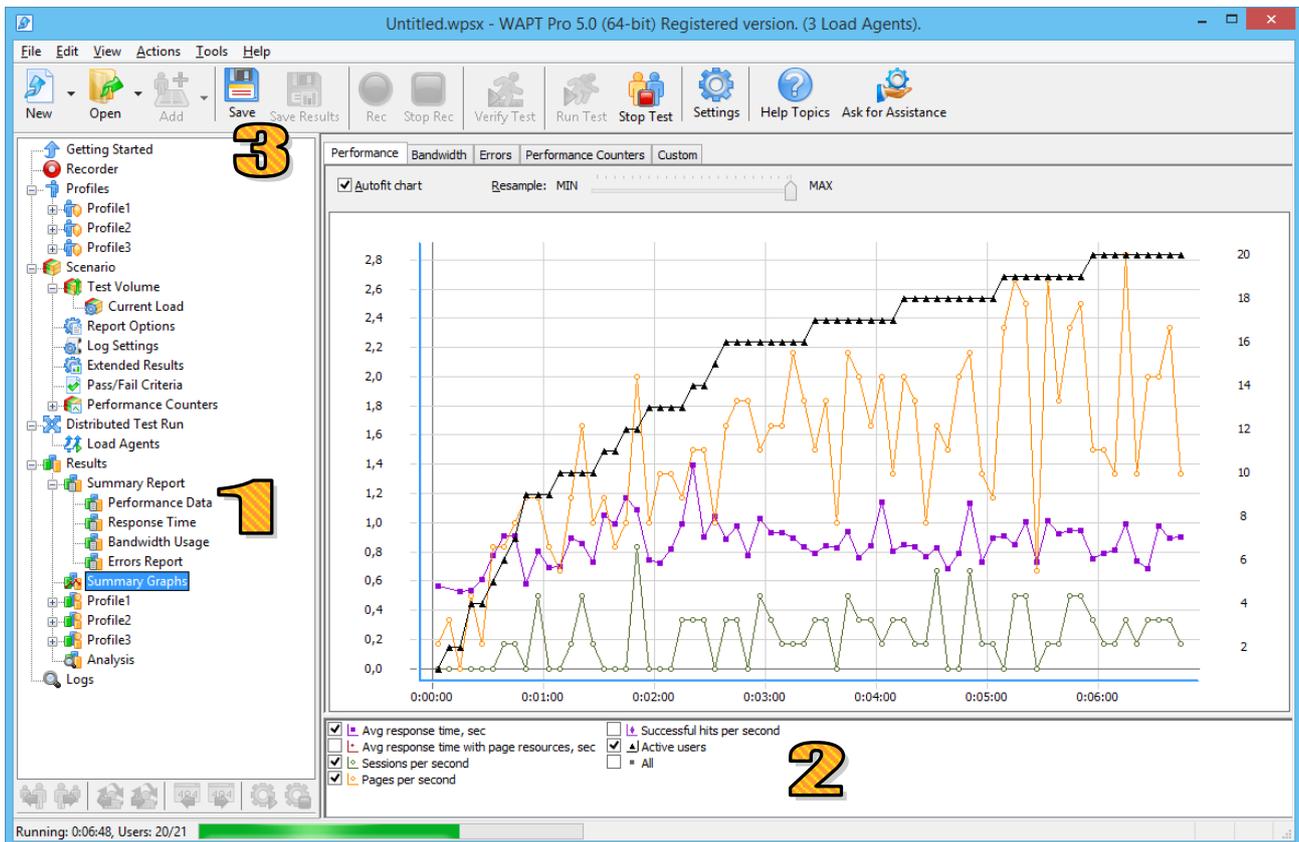
Now put checkmarks next to agents that you would like to use in the test and click the “**Run Test**” button on the toolbar to start it. If you use the regular version of WAPT, the load will be generated by the only one built-in agent, which you do not need to select.

* Available only in Pro version

Test results

After the test is started you can browse performance charts in real time (just wait few seconds for the first data to appear).

1 You can choose from a number of views inside the “**Results**” folder. Here you can find graphs for each profile and even for a particular request. Summary graphs and report tables for the completed period of the test are also available.



2 When browsing graphs you can choose between several tabs at the top and also select parameters to plot. Each parameter is depicted with a specific shape and color. All graphs have two vertical scales. Bottom left corner image () near the parameter description indicates that its graph is plotted according to the left scale. Bottom right corner image () refers to the right scale. For example, the number of virtual users on the above screenshot changes from 0 to 20, which corresponds to the levels marked on the right scale. All other values on that chart are measured according to the left scale.

3 You can save the results of a test run either as an HTML report, or as a special results file with the “.wprx” extension. In the latter case you will be able to open that file with WAPT at any time to continue browsing results.

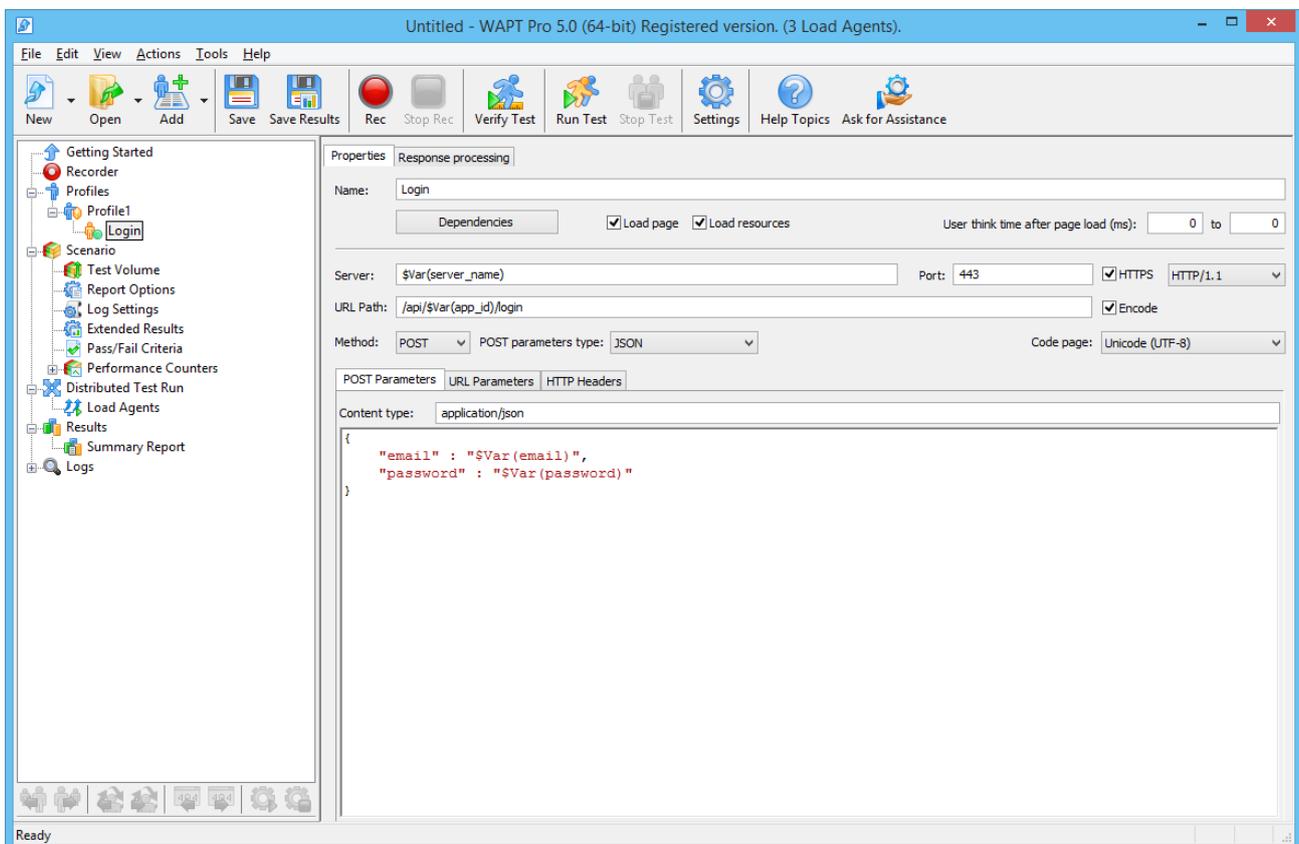
Part II: Advanced features

Creating an API test

Instead of recording profiles with a browser, you can create them manually by adding HTTP requests one by one. This is often required if you need to test a web API or a web service that does not have a client application suitable for recording.

You can create a new profile by selecting **“New | Profile...”** from the menu. The **“Add | Request”** menu will let you add requests to it.

Note that when you create a test that way, you need to make sure that all request properties are set correctly according to the specification of your API. This includes choosing the right HTTP method, server, URL path and HTTP headers. If your API call requires POST method, you need to specify the request body, which usually contains values wrapped in a JSON or an XML structure. You can use variables to pass any values, so a typical request will look as follows.



You can add several requests to a single profile, if they depend on each other and constitute a session. If you emulate fully independent REST calls, you can have one request per profile.

If you have troubles implementing such test, try finding an application that uses at least some of the API functions. If you record calls produced by that application with WAPT, you may get a clue.

Bunch parameterization

If you have a long profile with multiple requests passing the same session-specific value as a parameter, you don't have to repeat the same parameterization procedure several times. The **find and replace** functionality facilitates the process, but there is even more efficient **bunch parameterization** feature. It lets you create a variable for a value and replace all its occurrences with that variable from a single dialog.

1 Select the parameter bearing the value and click the **"Parametrize"** button to the right. The **"Parameterization"** dialog will appear. Alternatively you can open it through the **"Edit"** menu and enter the value manually. This lets you use this feature for values contained inside URLs and headers.

The screenshot shows the WAPT interface. In the background, the 'URL Parameters' dialog is open, displaying a table of parameters:

Name	Value	Encode
auth_key	\$Var(auth_key1)	<input checked="" type="checkbox"/>
referer	\$Var(referer1)	<input checked="" type="checkbox"/>
ips_username	\$LOrder(2, john, jane, steve)	<input checked="" type="checkbox"/>
ips_password	33b6b8f4aa254ead	<input checked="" type="checkbox"/>
secure_key	880ea6a14ea49e853634fbd5015a024	<input checked="" type="checkbox"/>

The 'Parameterization' dialog is overlaid on top. It contains the following elements:

- A search field containing the value '880ea6a14ea49e853634fbd5015a024'.
- Checkboxes for 'Match case' and 'Whole word only'.
- Buttons for 'Find', 'Decode', and 'Encode'.
- A left pane titled 'Choose the source from which you want to create a new variable' containing a list of URLs.
- A right pane titled 'Select all uses where you want to insert the variable' containing a list of URLs with checkboxes.
- A 'Variable name' field containing 'autoparam1'.
- A 'Create variable and parameterize selected uses' button.

2 Click the **"Find"** button to find all occurrences of the value in the profile. In the left pane of the dialog you will see all available "sources". This means all places in the server responses from which the value can be extracted. The right pane contains the list of all "uses" of the value.

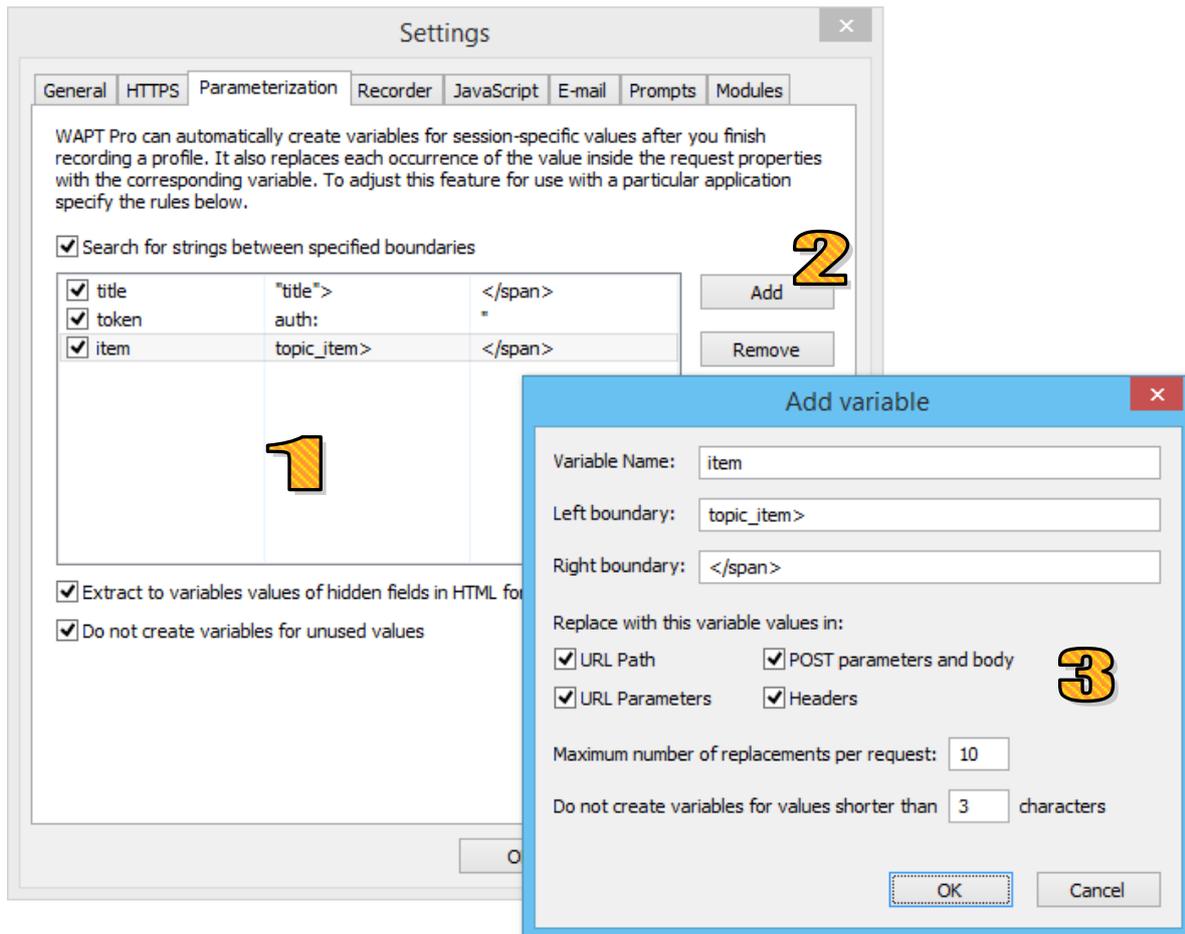
You should choose one source and tick all "uses" that you want to parameterize. You can click on items inside both panes to see the details in the main WAPT window. It is usually preferable to select the earliest source, i.e. the place where the value appeared for the first time.

3 Specify the variable name and click the **"Create variable and parameterize selected uses"** button. WAPT will automatically create a variable from the selected source and assign it with the help of the **"\$Search()"** function. You will be able to see that variable in the processing of the corresponding response. It will also replace the original value in all selected uses. Note that it is generally recommended to check the bounding strings selected for the extraction automatically. You may want to adjust them in some cases.

Custom parameterization rules

You may need to rebuild profiles for an updated version of your web application for regression testing. Copying parameterization from the original implementation may be a tedious task. However in the majority of cases the values are extracted to variables with help of the “**\$Search()**” function. Such parameterization can be fully automated with custom rules applied after recording a profile. This feature is similar to the automatic handling of the hidden values of HTML forms.

1 Click the “**Settings**” button on the toolbar and switch to the “**Parameterization**” tab to see the list of rules.

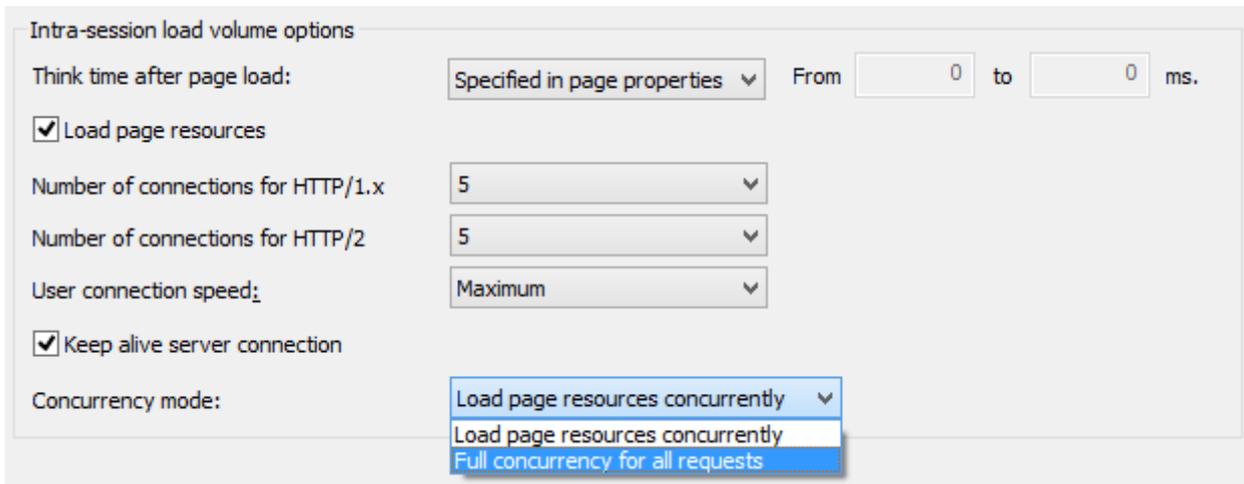


2 Click the “**Add**” button to create a new rule.

3 The “**Add variable**” dialog lets you specify how variables will be created in a newly recorded profile and where they should replace the originally recorded values. The limitations on the value length and the number of replacements are useful to avoid a situation with excessive number of occurrences processed by mistake. This can make profiles poorly readable and consuming too much CPU time on execution. That is why you should add rules only for values that are really session-specific.

Concurrency modes

The latest version of WAPT can execute concurrent requests within user sessions with help of the **“Full concurrency for all requests”** option available in the profile properties. It lets you emulate browser behavior more realistically.



However this feature should be used with caution, because in some cases a request may require certain data obtained from responses to some of the preceding requests. You can view and edit such dependencies in the request properties.



The list of original dependencies is based on variables and cookie values analyzed by WAPT after recording. If you add a variable in the processing of a response, corresponding dependences will be added automatically to all subsequent requests where that variable is used.

In some cases a dependence may be required by the session logic rather than by the use of values. In such case you can add it to the list manually.

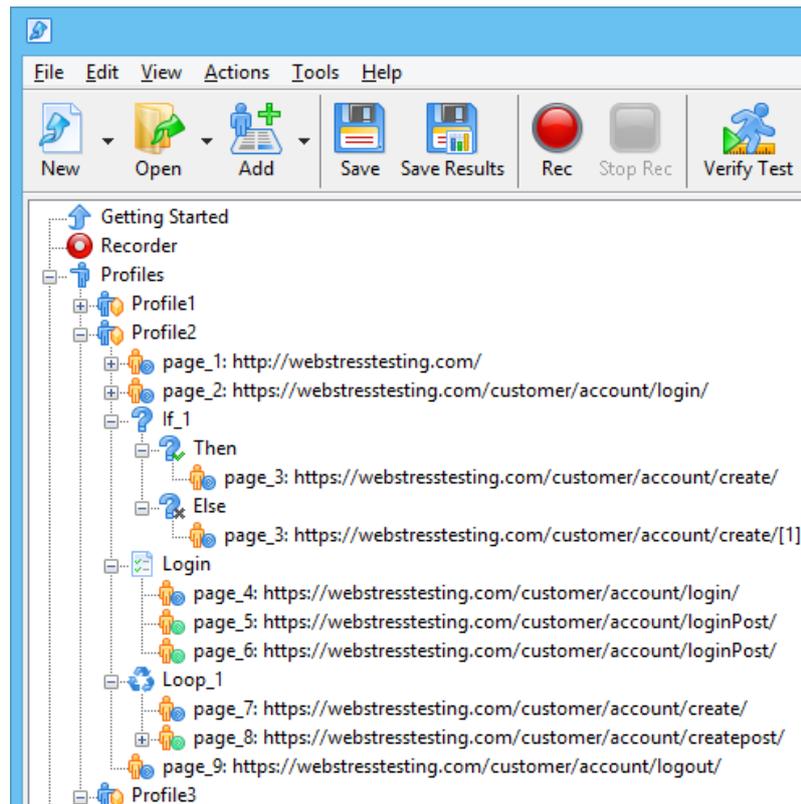
In all cases if request B depends on request A, the execution of B cannot start until A is completely finished.

If you set concurrency mode to **“Load page resources concurrently”**, all page requests will be executed sequentially, which is the same as making each request depend on the previous one in the profile.

Operators

You can use operators to control the execution of the test sessions. Note that some operators can change the order of requests or execute multiple copies of them. This makes the emulated session significantly different from the originally recorded one. That is why you should use these features with caution and make sure that the resulting sequence can be correctly interpreted and responded by the server.

Operators can be added through the “**Add**” menu. After you add an operator you can drag and drop requests to it in the left view.



The **if-then-else** operator lets you execute one of two alternative branches depending on a condition applied to a variable value.

The **loop** operator executes several requests in a cycle with a fixed number of iterations.

Conditional cycles can be implemented with help of the **while** operator.

The **task** operator does not affect the execution of the included requests, but can be used to group requests corresponding to a particular transaction. This lets you better organize the profile structure and measure the transaction time.

There are operators that suspend session execution until a condition is met. The **rendezvous point** operator synchronizes a number of virtual users at a specific step of their sessions. The **completion point** operator makes virtual user wait for all currently executing requests within its session to complete before sending the next request.

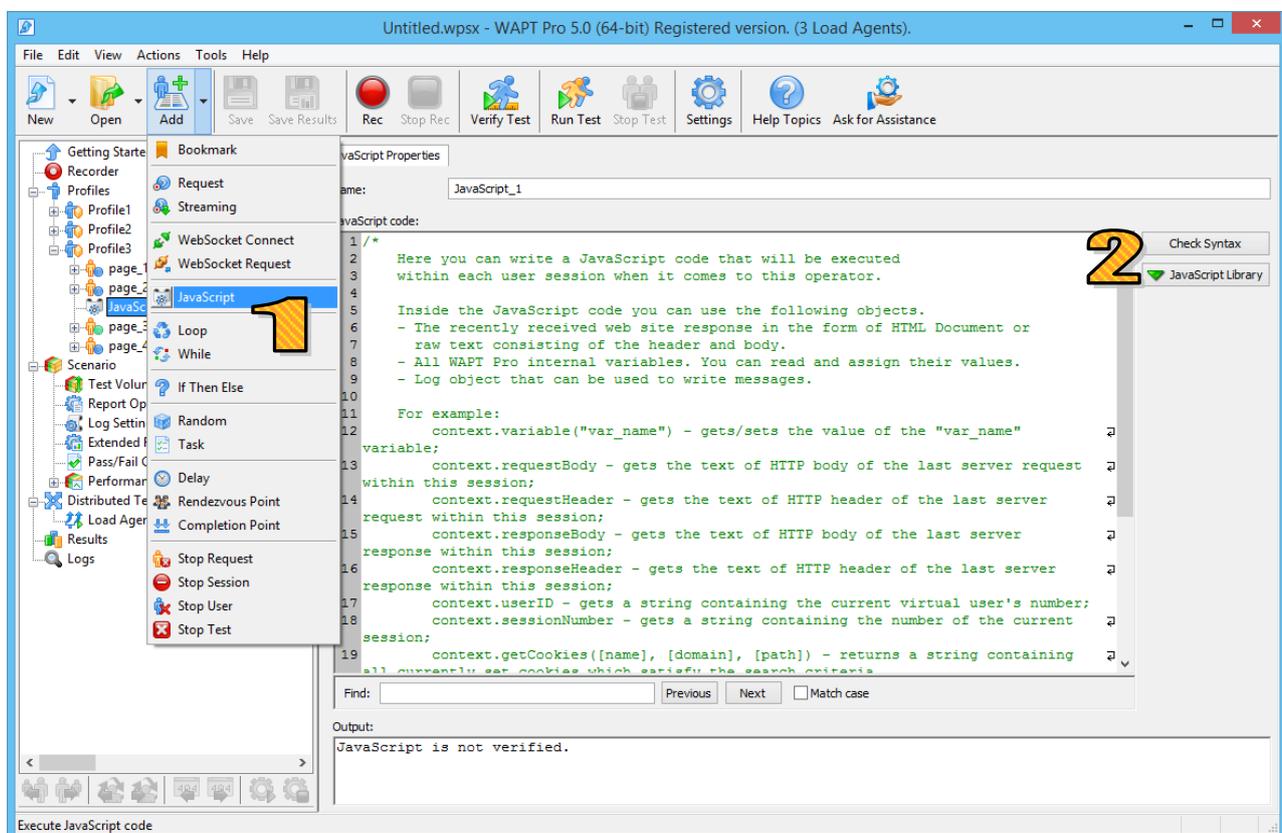
Finally, there are operators that stop current session, user or the whole test. They can be used together with the conditional operator to operate the test execution effectively.

The use of JavaScript*

You may find that the standard set of built-in WAPT functions like **\$Search()** is not sufficient to fully emulate the work of your web application. Some session-specific values may be generated by JavaScript code normally executed by the client part of the application in a browser. A very common example is the password encryption.

WAPT emulates browsers on HTTP level. It does not execute JavaScript code on the downloaded pages automatically. This would require too much system resources for multiple virtual users. To run a JavaScript code you need to specify it directly in the profile.

1 Select the request after which you want to insert the code. Choose **“Add | JavaScript”** on the toolbar. The JavaScript operator will be added to the profile. Select it to edit the code in the right view. Initially that pane contains a short instruction on how to use this feature. Here you can call functions defined in the WAPT Pro JavaScript library.



2 Click the **“JavaScript Library”** button to add files with additional functions. You can also access this list from settings. Click the **“Check Syntax”** button to check your code. The result will be displayed in the output window below.

Note that JavaScript code can be used only for calculations. The results of such calculations should be assigned to WAPT variables. You can use those variables in subsequent requests. However you cannot initiate new requests or use GUI functions in the code.

Another way to call a JavaScript function is to assign a variable with a special built-in function called JavaScript. It takes actual function name and arguments as its parameter.

* Available only in Pro version

Adjusting load at runtime*

You can change some load parameters while running the test.

- 1 After the test is started, the **“Current Load”** item becomes available inside the **“Test Volume”** item. It lets you monitor the current load and adjust it without stopping the test.
- 2 Click the **“Change...”** link next to the properties of a profile. This will open the **“Edit profile load parameters”** dialog where you can switch the profile to the manual load mode and specify the desired options.

Test started at: 26.09.2018 17:47:11
Time elapsed: 0:08:12
Remaining test time: 0:01:48 [Change...](#)

Profile	Sessions completed	Sessions to go	Remaining time	Current users	
Profile1 ramp-up: from 0 to 8 users	13	N/A	0:01:48	8	Change...
Profile2 ramp-up: from 1 to 8 users	19	N/A	0:01:48	8	Change...
Profile3 ramp-up: from 0 to 5 users	82	N/A	0:01:48	5	Change...

Running: 0:08:12, Users: 21/21

- 3 On the **“Load Agents”** page you can also change the set of load agents executing the test. If you uncheck an agent currently used in the test, WAPT Pro will prompt you, if you would like to drop all sessions being executed by the agent. A similar prompt is shown, if you put checkmark next to an inactive agent.

Change agent status (Agent name: localhost, Status: Operating)

Choose what you would like to do:

- Stop using this agent and disconnect immediately
Current user sessions will be dropped
- Keep running

Change agent status (Agent name: DENE88, Status: Online)

Choose what you would like to do:

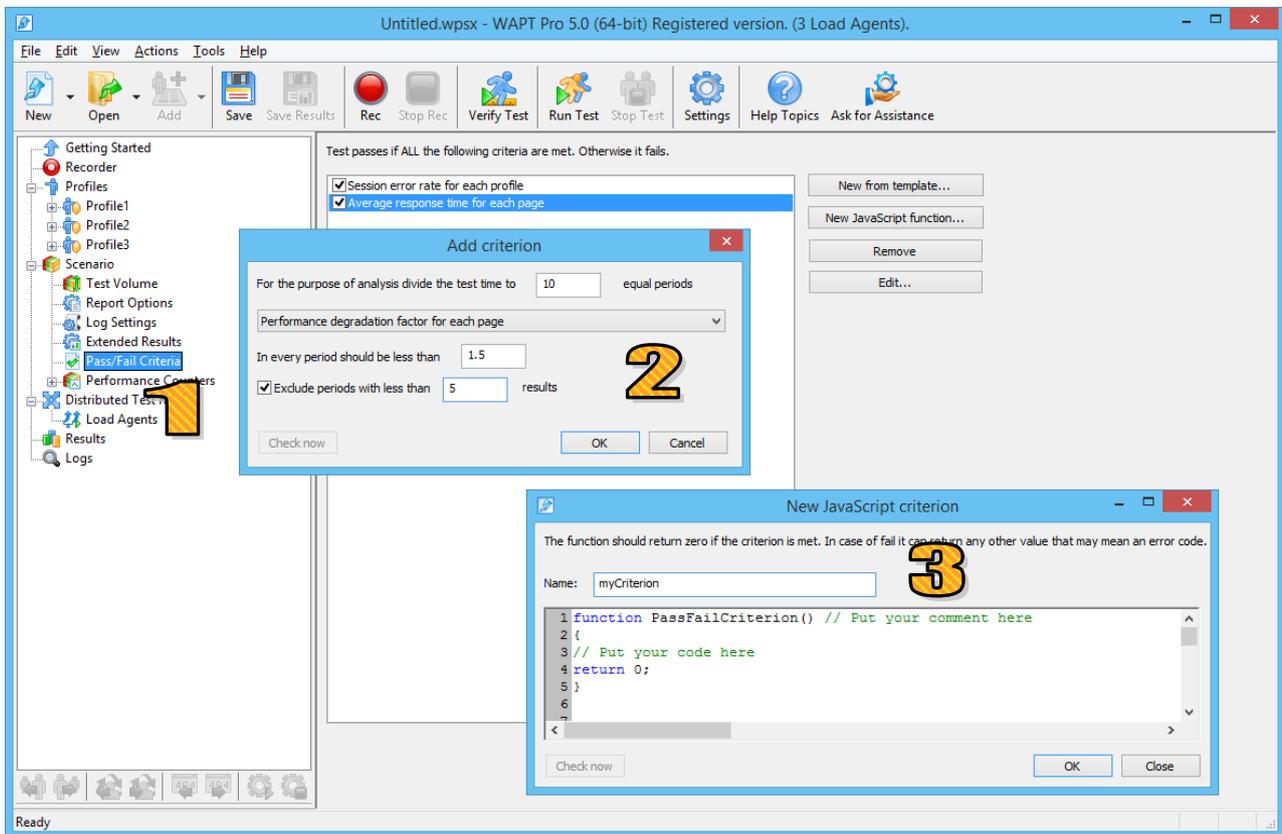
- Start using this agent in the current test
- Skip

* Available only in Pro version

Pass/fail criteria

If you have established requirements for the performance of your web application, you can make WAPT automatically check corresponding conditions and mark test as passed or failed.

1 Select the **“Pass/fail Criteria”** item in the left view. The right view will contain the list of criteria that will be applied to the test results.



2 Click the **“New from template...”** button to add a criterion basing on the standard templates. Each of them checks one of the parameters, such as the error rate or the response time.

3 In the Pro version you can also use criteria implemented with help of a JavaScript code. This will let you check very specific conditions.

4 Criteria are applied to the results automatically on the completion of the test. Only if all criteria pass, the test is marked as successful. You can see that status in the Summary Report.

Test result: FAILURE

Pass/Fail Criteria

Name	Result
Session error rate for each profile	SUCCESS
Average response time for each page	FAILURE

Test result: SUCCESS

Pass/Fail Criteria

Name	Result
Session error rate for each profile	SUCCESS
Average response time for each page	SUCCESS

Extension Modules

WAPT functionality can be extended with a number of modules. Each module adds features related to a specific technology or data format. The following modules will let you design tests with less manual work on the parameterization.

- **Module for ASP.net testing**
- **Module for JSON format**
- **Module for SharePoint testing**
- **Module for PeopleSoft testing**

We recommend installing these modules in case your web application uses the corresponding technologies.

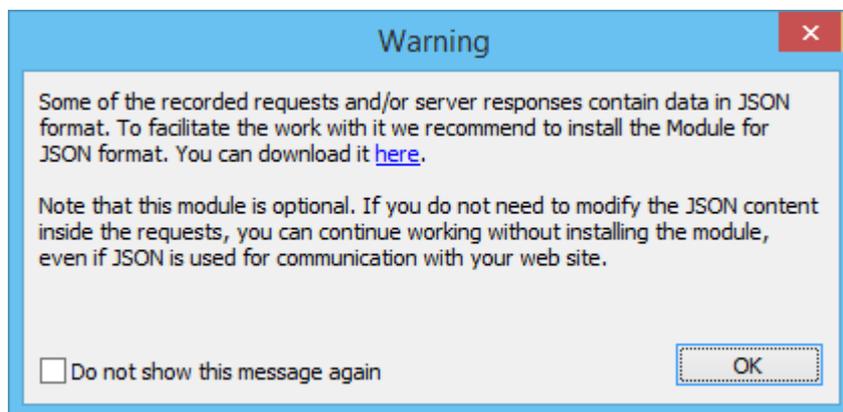
There are several modules that enable conversion of data structures from the native framework format to XML and vice versa.

- **Module for Adobe Flash testing**
- **Module for Silverlight testing**
- **Module for GWT testing**

Applications using WebSocket protocol for instant data updates will require the **Module for WebSocket testing** for recording and testing of such connections.

Finally, there is a module that allows working with any binary data that may appear inside requests and responses: **Module for binary formats**.

It is very easy to find out if you need a specific module to test your web application. When you finish recording a profile, WAPT will show a prompt with recommendations.



Each module can be installed and licensed separately. You also have an option to get all 9 modules in one package by installing the **Module Pack**.

Performance counters*

In addition to the client-side metrics, such as response time, WAPT Pro can collect performance information directly from the target servers. Those counter values are added to the test report and graphs along with other parameters.

- Expand the **“Performance Counters”** item in the left view. You will see counter groups:
 - **SNMP** is common for all types of UNIX systems; it can be enabled on Windows as well;
 - **WMI** is native for Windows servers;
 - **ODBC** is used to monitor database performance;
 - **Apache** is specific for Apache web server.

The screenshot displays the WAPT Pro 5.0 (64-bit) interface. On the left, the 'Performance Counters' tree is expanded to show the 'SNMP' group. The main window shows the 'SNMP Counters' configuration page. A yellow '1' highlights the 'SNMP' group in the tree. A yellow '2' highlights the 'Add new server...' link. A yellow '3' highlights the 'Add counters from repository' button. A yellow '4' highlights the 'Test' button. A dialog box titled 'WAPT Pro 5.0 (64-bit)' shows a list of server connection and counter status messages, all marked as 'OK'.

- Select an interface group in the tree and click the **“Add new server...”** link in the right view. This will create a new server. You can add counters to it.

- WAPT Pro has a set of predefined counters for a number of server tools. You can add and use them without modification. Click the **“Add counters from repository...”** button on the server properties page for that.

Each counter is implemented as a JavaScript function that returns counter value. You can use the implementation of the predefined counters as examples and create custom ones in a similar way.

- Click the **“Test”** button in the server properties page to check that the counter collection works properly.

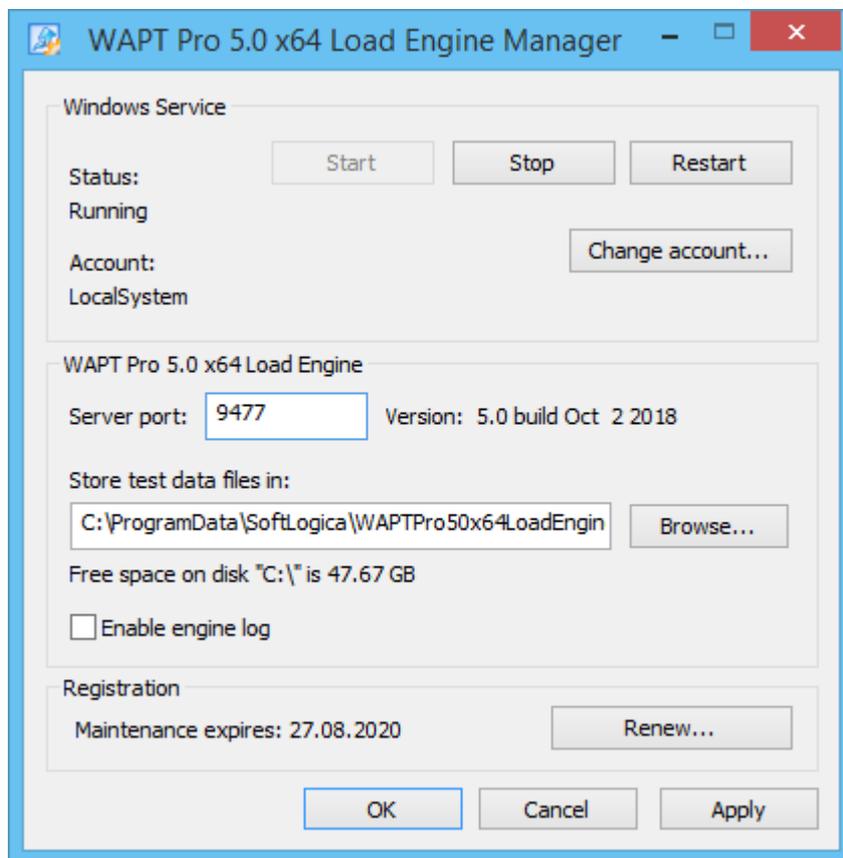
* Available only in Pro version

x64 Load Engine*

Regular load agents that come with WAPT Pro by default can emulate up to 2,000 users per system. This is a relatively small number in case you need to run a high capacity test. You have an option to replace regular agents with one or several **x64 Load Engines**, which are identical in functionality, but let you fully utilize the resources of high performance servers. A single engine can run from 10,000 to 50,000 virtual users per system. The exact number depends on your test specification, but it is limited only by the performance of your hardware.

Note that WAPT Pro installation package does not include the **x64 Load Engine**. You can download this product separately. The engine can be installed on any system with 64 bit Windows OS starting from Windows 7.

The engine runs as a system service. It can be launched and managed from the **Load Engine Manager** available under the Start menu on the system where the engine is installed.



By default this service is launched automatically, so you can start using it immediately after the installation. There is no need to start it manually after the system reboot.

* *Compatible only with Pro version*