

## Web Stress Testing

Matt Odhner, Application Center Microsoft

### Introduction

Performance and capacity planning is of top management concern for Internet web applications. The lack of proactive and continuous stress testing and capacity planning leads to unexpected problems with availability and performance. Downtime could be financially devastating to today's web based companies.

Stress test tools can alleviate this problem by providing in-depth performance information on all the pages that comprise an application. Further, capacity planning can help to reduce administration. The process involves finding resource problems in a web application, determining performance and stability characteristics at development time, and continuing capacity monitoring after deployment.

Web Application Stress is a resource kit quality tool created at Microsoft to help find performance and capacity problems on Microsoft's online properties. It became clear that providing a tool like this to customers would help developers and administrators locate bottlenecks, avoid financial losses, ensure customer satisfaction, and preserve a company's external image.



#### **Matt Odhner**

Matt is a Program Manager on the Application Server team. He joined Microsoft in 1993 and managed the development of the original "Homer" web stress tool while working as a test lead in the Microsoft IT department. Since that time the tool has gained in popularity, mainly for its ease of use, both inside and outside of Microsoft. The tool is available for free download from:

<http://webtool.rte.microsoft.com>.

He is currently managing the development of Application Center Test, Microsoft's premier product in the web performance and capacity planning tool market.

---

## Web Application Stress tool

Microsoft Web Application Stress (WAS) is a web stress tool that is available for free download from the Internet web site: <http://webtool.rte.microsoft.com> and is also available in the Windows 2000 resource kit. The tool is designed to performance test web applications.

Before you get started stress testing your application, you'll want to confirm that the web application works as expected on at least a functional level. Make sure all of your pages display correctly with a browser, confirm that all images load correctly, and ensure that your database or any other backend services are running correctly before using the tool. The tool will not explicitly inform you if a backend service, such as SQL server, is not functioning as expected however, the report does capture HTTP result codes, so you can determine whether pages are missing or not responding fast enough.

### ***The Test Platform***

A lot of the accuracy of your web test depends on the test platform you use to test your application. Ideally, you'll have a test bed that contains the exact same web server hardware used in production. If you plan to deploy your site to two or more web servers in production, make sure you use at least two web servers in the lab. This will help you work out any state management issues that may arise from utilizing an array of web servers (also known as a web farm).

WAS must be installed on each client machine, although you can control all client machines centrally from one of the clients, which is known as the controlling client. The controlling client communicates with the other clients via DCOM, so this must be enabled on all client machines, which it is by default on NT 4.0 and Windows 2000 machines.

The stress clients should ideally be machines with Windows NT 4.0 workstation or server (with at least 32MB of RAM and Service Pack 4.0 or greater installed on them) or any flavor of Windows 2000 (with at least 32MB of RAM). WAS requires about 10 MB of disk space to install. The amount of RAM you need is dependent on how complex your test scripts are, how much data you are collecting in the report, and the complexity of your web server application. It is recommended that you also install Internet Explorer 5.0 or above on the client machines. This will be helpful if you plan to create scripts using the browser record feature, discussed later.

There is no hard and fast rule on the number of clients you'll need. As a rule of thumb, the more complex your web application, the less clients you'll need to bring the web servers to full utilization. The less complex your web application, the more clients you need to handle the speed at which the report data is returned to the client machines. A good way to test the number needed is to start testing, and compare load on clients and server - if your clients are fully tasked but your server isn't, then you need more clients (see below for more on this).

### ***Installation***

The tool installs an NT service, so you will need Administrative rights on all the client machines to ensure that the install goes smoothly. The tool takes about 40 minutes to download from the web site if you are using a 56K modem, or about 10 seconds to download and install if you have a T1 connection to the Internet. Everything needed is downloaded in a `SETUP.EXE`. Go to each client machine and run the executable using the Run option in the start menu of Windows NT or Windows 2000. Follow the setup instructions and go with the defaults. There aren't a lot - it's a pretty straightforward installation.

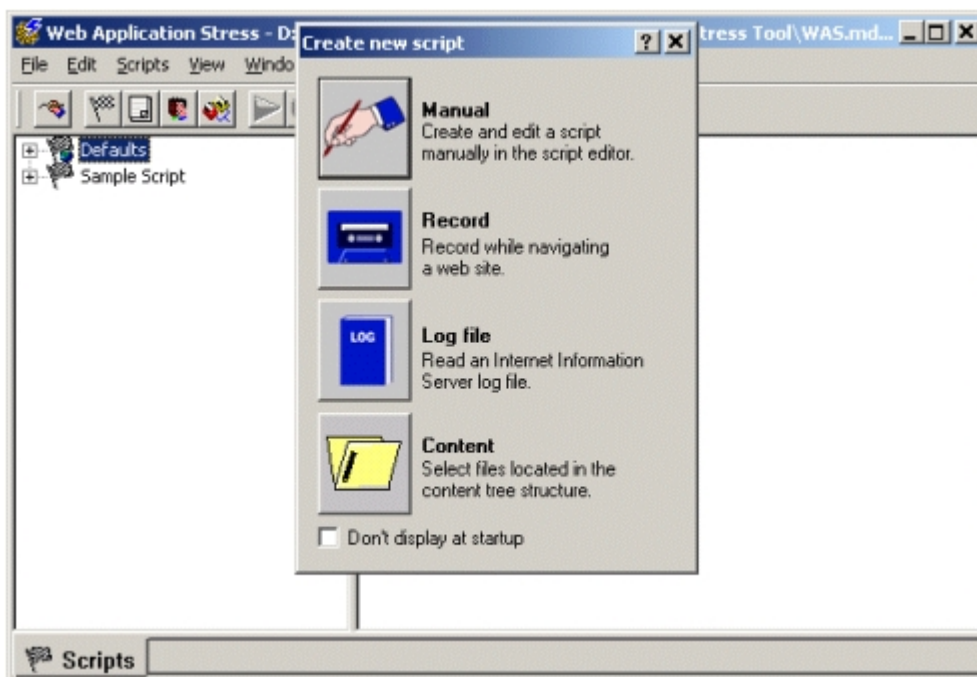
During the installation, whether successful or not, an `INSTALL.LOG` is created and placed in the folder where you installed the tool. If you have difficulties with the setup you should look at this file using notepad and troubleshoot any problems reported.

## Opening the Tool for the First Time

Since you can centrally control all the client machines from one particular client, whichever client you decide to work from becomes the controlling client. The same bits are installed on all client machines, so the only thing that differentiates the controller machine from the other clients is the script database.

The script database is a Microsoft Access 98 .MDB file. If you want to look at the database itself, you can open it in Access. If you open it in versions of Access later than Microsoft Access 98 you will not be able to edit it without corrupting the database file, So don't do it! The file is named WAS.MDB by default, but you can change the name and/or copy the file to other client machines if you would like to use another client machine as the controller. If by chance the database becomes corrupt, you can stop the NT service from the CMD line by typing "net stop webtool" and then you can delete the corrupted database. The tool will automatically create a new MDB file when it is started again.

To open the tool, select Start, Programs, Microsoft Web Application Stress, Microsoft Web Application Stress tool. An interface similar to the following figure should be displayed.



Close the 'Create New Script' dialog for now by selecting the X in the top right corner. The left hand window of Web Application Stress is known as the script view. This view displays all of the scripts stored in the active database. Each script has several nodes of its own which we'll discuss in more detail later.

The scripts view has a "Defaults" node and a "Sample Script" node. The Sample Script contains six script items that are used to test the Web Application Stress tool's basic functionality. The files associated with this script are located in a folder named "samples" inside the folder location where you installed WAS. Move these to your Internet Information Server web content root folder to use them.

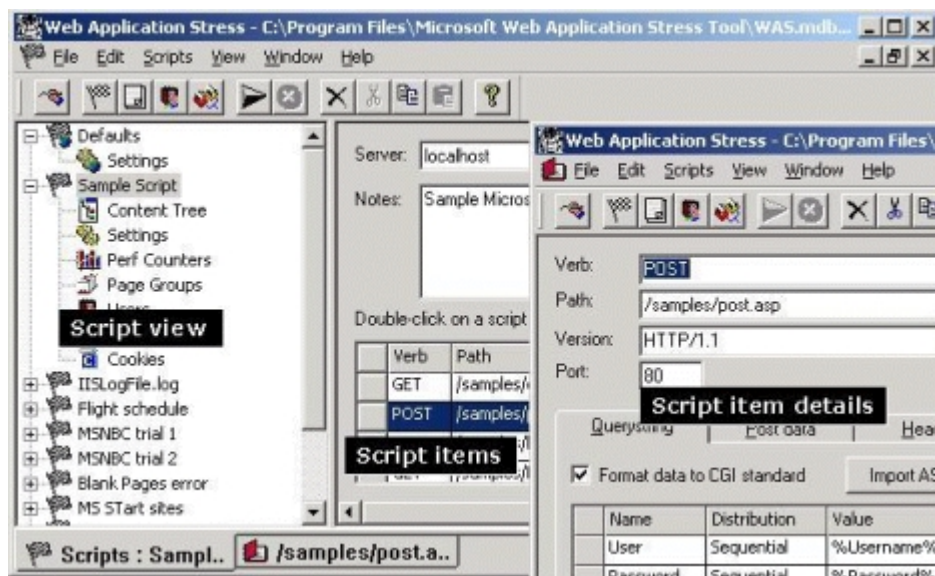
## Script Item Details View

Using your mouse, highlight the row header of one of the script items in the Sample Script, and then double-click.

Double-click here

Verb	Path	Group	Delay
GET	/samples/cookie.asp	Default	0
POST	/samples/post.asp	Default	0

This opens the script item details view. From this view you can edit the querystring name-value pairs, add or change POST data, modify the header, enable a secure socket layer, and add Remote Data Service (RDS) requests.



## Script Nodes

Each script contains seven nodes that pertain to the script they are under. Add a new script by right-clicking the Sample Script and selecting Add from the short-cut menu. A new script is created named "New Script". All seven nodes are viewable when the script is expanded.

### Content Tree Node

The first node under New Script is called Content Tree. This node allows you to add script items to a script by selecting files in a folder. The ellipsis next to the Content Location edit box allows you to select a specific folder. The contents of the folder are displayed in the lower pane. Files may be added to the script by clicking the check box next to the file names.

The Virtual Root edit box is used to precede the script items with a specific virtual root, or vroot. This is available because IIS vroots can map to any number of directories and subdirectories. You can add a vroot in the following format "/vroot1/vroot2/vroot3". Make sure you press "apply" after adding the vroot.

## Settings Node

The next node under a script is called Settings. This is where the properties of a script are located. Select the Settings node and change the Test Run Time to 1 minute, down from the default of 15. Leave the other settings as they are for the time being but look over the other options in this view to get an idea of what can be configured for a script.

You may also set the default settings for all new scripts by selecting the Defaults node and changing those options. Keep in mind that the settings in the Defaults node will not affect existing scripts, such as the Sample Script – they only apply to all new scripts you create.

The screenshot shows the Settings Node configuration window with several sections and their corresponding annotations:

- Concurrent Connections:**
  - Stress level (threads):  → Number of concurrent connections
  - Stress multiplier (sockets per thread):  → Number of sockets that will be used within each thread
- Test Run Time:**
  - Days:  Hrs:  Mins:  Sec:  → Test run time
- Request Delay (in milliseconds):**
  - Use random delay
  - Min:  Max:  → Apply a delay between min and max for all script items
- Suspend:**
  - Warmup: Hrs:  Mins:  Sec:  → Prevents report data from being collected for the period defined in warm-up or cool-down
  - Cooldown: Hrs:  Mins:  Sec:
- Bandwidth:**
  - Throttle bandwidth  → Increases the number of concurrent connections
- Redirects:**
  - Follow HTTP redirects Max:  → Turns on redirect support and applies a limit to the redirect depth
- Throughput:**
  - Use users, passwords, and save cookies
  - Save page statistics → Toggles report data collection and run-time functionality
- Name resolution:**
  - Resolve network lookups on remote clients → Used when clients are in multiple subnets, different from the master client

Let's discuss each one of these settings because their meaning is not clear at first glance and it is important for you to understand these options in order to create an accurate test script.

### Concurrent Connections

At the top, inside the concurrent connections bounding box there are two edit boxes, Stress level (threads) and Stress multiplier (sockets per thread). WAS can direct various levels of stress against a web server, and these fields determine that level of stress.

The total users connected, also called Concurrent Users, are equal to the stress level (threads) multiplied by the stress multiplier (sockets per thread). For most applications, using 1 to 100 threads is sufficient with 1 socket per thread.

Stress level is the total number of Windows NT threads that are created across all of the clients. Each thread can create multiple sockets and each socket is a concurrent request. The following formula explains this relationship:

$$\text{Total Concurrent Requests} = \text{stress level (threads)} \times \text{stress multiplier (sockets per thread)} = \text{Total Number Sockets}$$

From the web server's perspective, it just sees lots of incoming socket connections. The server doesn't care what the thread architecture is behind the incoming requests, so your stress is equal to the multiple of threads and sockets, not simply threads alone.

In general there isn't much reason to increase the stress multiplier (sockets per thread) value above 1. However, there is one scenario where increasing the sockets per thread is useful. This is the case where you are using one client machine and a high number of threads.

When a test begins, the WAS threads are divided among all client machines used in the test. Using extra client machines increases the stress on the web server, because each WAS thread has more CPU, memory, and network bandwidth. However, if the client machines are not utilizing more than 50% of their available CPU, memory, or bandwidth, then adding more client machines will have negligible effect on the test.

Determine the maximum number of concurrent users you expect to see on the web site at any given time during the day, and then use this value as the stress level (threads). To get more unique users, add more users to the Users node. You probably don't need to add tons of users since it is unlikely that every request will be from a unique user. Once you determine that number, see if one client can handle the load without becoming fully saturated on some resource, like the processor utilization. If it can't, add more clients until all of the client machine processors stay below 80 - 85%.

### ***Test Run Time***

The tool can be configured to run for a period of time, but cannot be set to run for a specific number of iterations. For example, adding 200 to the seconds edit box will be reformatted at 3 minutes and 20 seconds as soon as the seconds edit box loses focus.

### ***Request Delay***

The request delay is useful for shaping the request curve into a more random distribution. Examining web server request logs, you'll notice that requests arrive at a fairly random distribution, becoming heavier at certain times during the day and then falling off.

If you simply want to see the maximum number of requests that the web server is capable of serving, then you may want to turn the request delay off. If you are trying to create a realistic request curve against the web server, turn the request delay on and examine the logs of your production servers to ascertain the values that are appropriate for minimum and maximum. Remember that these values are in milliseconds (ms), so be sure to divide by 1000.

### ***Suspend***

The warm-up and cool-down values are used to prevent the tool from collecting report data for a preset amount of time. However, the script is still requesting pages at full throttle. This is

useful if you would like to allow the web server to "warm-up" before collecting data. For example, to allow caches to fill, COM components to instantiate, etc.

The cool-down prevents the report from collecting data at the end of a test. This is useful if you want to eliminate any random results that may occur as the threads are shutting down at the end of a test run.

### ***Bandwidth***

Bandwidth throttling is a hot topic in the web stress tool industry. To do bandwidth throttling correctly requires a separate software or hardware router that is modifying the TCP/IP packets at a low level. WAS does not provide this level of bandwidth throttling. This feature is mainly useful for increasing the number of concurrent connections during a test run.

**Keep in mind that bandwidth throttling in WAS is only effective on files greater than 1.2 KB. Also note that throttled bandwidth reads 256 bytes at a time, while un-throttled bandwidth can read up to 8K.**

### ***Redirects***

Following redirects is enabled by default. We also added a redirect depth to prevent your ASP code from getting the tool into an infinite loop in case there is an error in the web page redirect code. Although the tool does not allow you to create scripts that request pages from multiple domains, it will follow a redirect to one other domain.

Also note that the tool will modify the verb of a redirected `POST` request into a `GET` request. The HTTP specification is not clear what to do in this scenario, but the most popular browsers behave this way.

### ***Throughput***

If your client machines are at a premium, and you feel that performance monitor will provide all the report information you'll need, you can unselect the `Save page statistics` option to save some additional processing power on the clients. This will then translate into being able to produce a greater load with fewer client machines.

If your site does not use cookies or authentication, and you are not using the `%USERNAME%` and `%PASSWORD%` string replacements, you can turn off the `Use users, passwords, and save cookies` option to seek even more throughput out of a given set of client machines.

### ***Name Resolution***

The chances of you needing to modify this setting are slim, but if each of your client machines are in a different subnet, you want to select this check box so that the name resolution of the web server occurs on each individual client machine instead of only the master client.

### ***Perf Counters Node***

The Perf Counters node allows you to specify which performance counters, and which computers you will be collecting counters from, during a test run. The counters are collected at the interval specified in the `Collection interval` box.

Counter results are displayed in the reports window, divided into percentiles. These counters are also collected in a file named `HCOUNTERS.CSV`, located in the folder where the tool is installed. This file contains the entire set of counter values collected over all intervals and may be opened for analysis in Excel. `HCOUNTERS.CSV` is overwritten with each successive test, so be sure to copy it to a separate folder or rename it before starting a new test if you intend to keep all the

performance monitor counters from a previous test run. The report only stores a summary of the counters.

## **Page Group Node**

Returning to the Sample Script, notice that the last two script items contain the text "adGrp" under the Group column. These are page groups. A page group is shown as "Default" unless you change it. Page groups allow you to control script item execution order as well as the frequency with which script items will be requested.

The Page Group node displays all the page groups associated with the current script. You may also change the distribution percentages from this view. Notice that keep-alives are enabled for entire page groups at a time. You add, change, or delete page groups by selecting an item in the Group column from within the script items window.

Following is a list of ways that you can use page groups:

- Page groups provide a logical grouping of script items. For example, a web page may contain several elements that the browser requests individually such as images, frames, and style sheets. If you record a script you will see that all of the elements of a particular page are requested sequentially. Since the browser requests these elements, you must include them in your script, as opposed to simply requesting the page where the images reside. You can use a page group to collect all of these components into one transaction. This allows the tool to request all the items in a page group sequentially, and do so with only one user. In other words, they help produce a more accurate stress script because elements of a group are requested sequentially, regardless of their location in the script items window. This is more realistic because this is how a browser would request these elements.
- Page groups also prevent a random delay from occurring between the script items within a single page group. Only those from a different page group get a random delay imposed prior to the request. The idea here is to simulate the way popular browsers use four threads to request all the items in a page simultaneously. The browser would not randomize the requests to all the items in a particular page.
- Page groups can also provide more accurate reporting statistics because all of the pages in a group are summarized in the report. In a WAS report, there is a section called Page Groups where information regarding each group is displayed. You can view the distribution, number of hits, and result codes associated with a particular page group.
- You can assign a configurable percentage to a specific page group. For example, if you know the home page will be requested 80 times more than the other pages on your site, you can assign all of the elements of the home page into a page group and assign a value of 80 to the page group in the Distribution column of the Page Groups section in the script properties. You can then evenly distribute the remaining pages among the remaining 20%.
- Finally, page groups make your script more readable because groups of files are associated with each other by a page group name. Once you add a page group to one script item, that name becomes available in the page group drop-down and can quickly be selected and assigned to multiple pages in a script.

For more information on page groups, view the following article:

<http://webtool.rte.microsoft.com/Threads/WASThreads.htm>.

## **Users Node**

Users provide a couple of features, namely:

- They are a logical storage location for cookies
- They are passed in the header to sites that require authentication
- They can be used in the %USERNAME% and %PASSWORD% string replacements in the querystring

**WAS users are not the same as stress level (threads), a setting located in the Concurrent connections section of the Settings node. Increasing the number of users has no bearing on the amount of load that is generated, so the two concepts should be kept separate**

When creating new users, the prefix is handy because it allows you to automatically create lots of unique usernames and passwords. This increases the realism of a test that requires authentication or lots of unique users. You can also generate these lists in an Excel spreadsheet, save the Excel file to the comma separated value format (.csv), then use the Import feature to bring the users' names and passwords into the WAS database.

When a test begins, all WAS users are divided among the WAS threads given by the Stress level setting. As requests are made, each WAS thread uses the username, password, and cookie from a new user from the pool allocated to that thread. This prevents two users with the same name from ever requesting the web server at the same time.

If WAS has been configured with fewer users than threads, some threads will not have users, and any authenticated pages will fail, and any interaction with cookies will be disabled for those threads. For this reason, whenever testing personalized websites it is important to have more WAS users than threads. The tool will warn you if this situation is present.

Adding too many users taxes memory on the clients, so watch the client memory counter in the performance monitor during the test. Too many users will also cause the stress test to take an inordinately long time to start. The number of users you should create depends on many factors, such as the amount of memory on the client machines and the number of unique users you would like to simulate. We have successfully created over 60,000 users for various test runs, but this was on several beefy client machines.

## ***Clients Node***

If you are still in the Users view, select Scripts from the View menu to return to the Script view, then select the Clients node under the Sample Script and double-click on the Default client group. This opens the Clients view, where you can add and delete client machines from the current group, or add new groups of client machines. You can configure WAS to use multiple client machines when running a stress test using this view.

If this is the first time you have seen the Clients view, you will see that localhost is the only client and that it has a check box next to it. This means that the current machine is acting as a WAS client.

To use additional client machines you must install WAS on all clients and the Webtool service needs to be running. Additionally, you may find that you are required to add your name and password to the WAS service on every client machine in order for the machines to properly communicate with each other.

Since WAS exploits Distributed Component Object Model (DCOM) to communicate with client machines, your clients can be located anywhere on the network, or even across the country if your bandwidth permits. All of these clients can be controlled from one client known as the

master client. The master is responsible for starting a test and sending the script data to all client machines, as well as collecting and summarizing the reporting data when a test is complete. If a machine fails for some reason during a test run, all of the data from that client is ignored.

Run a quick test and then check the WAS report for specific information on the status of your client machines. The report summarizes the clients used, the number of threads used by each client, the number of requests that each client produced, and socket error information. One or two socket errors, over the length of a quick stress run, is probably acceptable. More than a few socket errors is an indication of network, net card, client machine, or authentication failure. If all the requests are socket errors, check to make sure your web server name was changed from the default of "localhost" to the name of the web server you are interested in stressing. This is a common issue.

Here are some things to consider when determining how many client machines to use in a stress test:

- If your test requires a great deal of stress you will need to use more threads in your script properties. The more threads you use, the more powerful the client machine is required to be in order to accommodate for the thread activity. If one client machine is incapable of handling the thread count you will need to use multiple client machines. Use the performance monitor to watch the processors and memory on the client(s) while running the stress. Very high and sustained processor utilization of 80% or above is a good indication that more clients are required to produce the level of stress you desire.
- If your stress test consists of a good deal of small HTML pages, you will find that this demands more processor time on the client machines because WAS spends more time collecting and organizing the resultant data than when stress testing light pages. Use the performance monitor to watch the client machine processor utilization. If you see sustained utilization of greater than 80%, it is time to consider decreasing the stress level or increasing the number of client machines.
- Make sure you use at least as many threads as there are client machines in your test. The reason for this is that WAS distributes the threads across client machines. If there are not enough threads, certain clients will not be used.

## **Cookies Node**

The Cookies node displays all the cookies associated with the active users in a script. If no cookies have been assigned, or if not all users have received a cookie during the test run, their cookie will be blank.

It is a good idea to delete the cookies before starting a new test run, unless your stress test requires a pre-determined cookie value on first access. You can delete all the cookies by selecting the top left corner cell of the cookies view, and pushing the *delete* key.

WAS stores all user cookies on a per-server basis and keeps them separate. For example, User1 might have a unique cookie used for stress tests against `www.microsoft.com` and a separate cookie used for tests against `home.microsoft.com`. If you wanted to do a quick test pass against a specific machine by IP address, all new cookies would be created because the IP address wouldn't string match to either DNS name, even if this IP maps to the same physical computers.

WAS ignores the `Domain`, `Expires`, and `Path` fields of cookies and uses the above logic for managing them. If necessary the `WAS.MDB` file can be opened directly to edit the DNS name associated with a particular set of cookies.

WAS uses two types of cookies, dynamic and static. The dynamic cookies can be created and modified as the test runs. The static cookies will not change during the test. Static cookies are created in two ways, by going to the header tab of the script item details and creating a new cookie in the header, or by recording a script where the Record browser cookie option in Step 1 of the browser record wizard is selected.

## ***Reports***

Select Reports from the View menu to open the Reports view. If there are no reports, you'll need to run a stress test against the Sample Script first. Expand the Sample Script report to display all of the Report nodes. Assuming you have run the sample script at least once, there should be a node whose title is the date and time in which your latest test was started. Expand and select the top level of this Report node to view a summary of this test.

Expand the Page Data node and select the first script item. The right hand pane displays detailed information regarding this script item. Reports provide the response time for specific pages by determining when the page has finished downloading on the client. This is a good source of performance data.

The Time To First Byte (TTFB) calculates the time from the request for the page until WAS receives the first byte of data, in milliseconds. The Time To Last Byte (TTLB) calculates the total time from the request until the last byte of data has been received on the client, in milliseconds. This number includes the TTFB time and any additional time needed to receive the last byte of data. All of the requests are sorted and then the data is divided into percentiles.

## ***Percentiles Defined***

Several report fields – namely content-length, response time, and performance monitor counters – are expressed in percentiles. Percentiles provide a more descriptive view of the data than a simple standard deviation would.

Percentiles summarize information about a large collection of data points. A percentile X is the value of the data point where X percent of the remaining data points are of a lesser value.

For example, assume that WAS has four million response time measurements. The minimum measurement, also known as the 0th percentile, might be 400 ms for a typical test. The maximum response time measurement, or 100th percentile, might be 60,000 ms. This is a very wide range of values, but it starts to paint a picture of how fast this particular web page is.

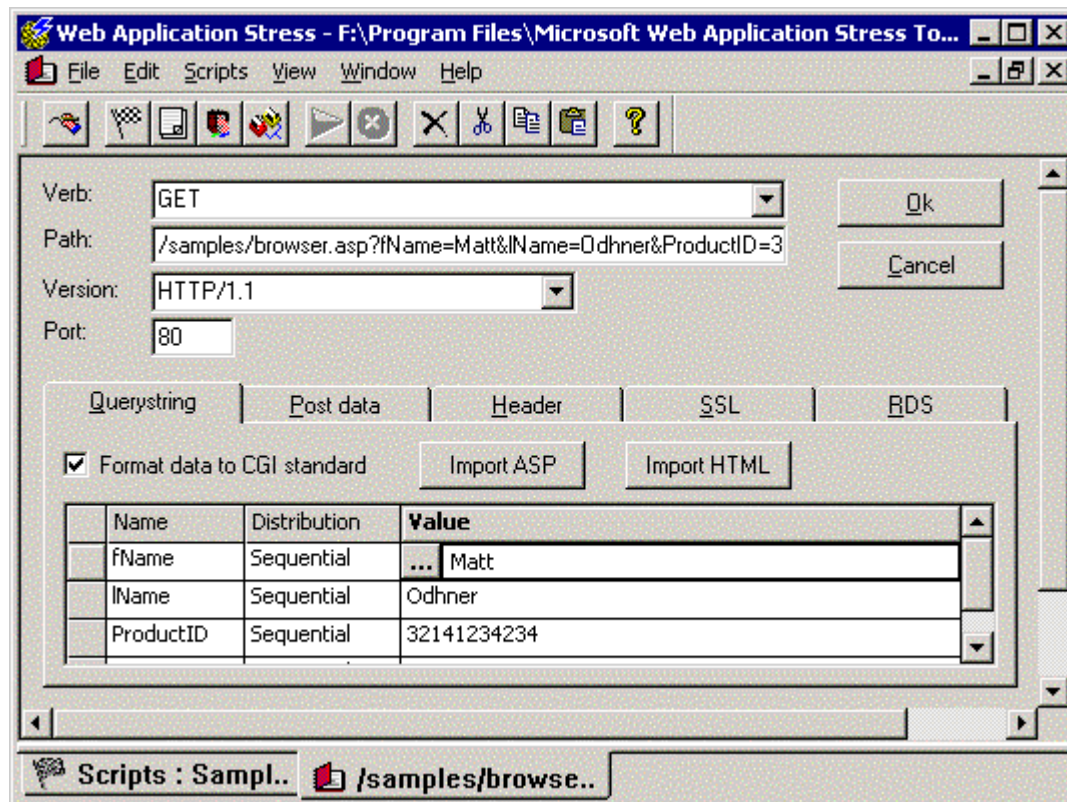
The trouble is, you need to know whether most measurements were near the 400 ms mark, the 60,000 ms mark, or somewhere in between. The 50th percentile represents the midpoint response time (or median) at which two million measurements were slower and the other two million measurements were faster. For a typical test, this will be 1000ms, much closer to the minimum. To further visualize performance, the 25th percentile is taken. This is the response time at which exactly one million measurements were less, and the other three million measurements were equal or greater. The 75th percentile is also taken. Typically, the 25th percentile will be 800 ms, and the 75th 1200 ms.

With these percentiles, it can be shown that a full 50% of users will see response times between 800ms and 1200ms. A lucky 25% of users will see fast response times between 400 and 800ms, while an unlucky remainder will experience waits from 1200 to 60,000 ms.

When reading percentiles, the 50th percentile is the most common value, while the 25th and 75th percentiles illustrate where most values have fallen. The minimum and maximum give clues about the full possible range of values that will sometimes occur.

## Querystrings and the Querystring Editor

WAS makes the process of creating a list of name-value pairs very easy by displaying a grid that contains the names in the left column, and the values in the last column, as shown in the figure below:



Notice the ellipsis next to the field value in the first row of the querystring. This symbol appears whenever the value field is selected in the querystring editor. Clicking it displays the field values editor. You may add new values manually to the field editor, separating each value with a *RETURN*, or you can paste them into the field value editor from another application, such as a list from Excel.

You can use %Username% and %Password% in the querystring fields. These are not literal values; they are special variables that tell WAS to pass the next available user and next available password from the Users view. WAS automatically cycles through the user names and passwords, passing the next set with each post.

## Creating a Script with the Script Wizards

### Manual Script Creation

You can create a script manually, by typing the script items into the script items grid. This is not the recommended method to use if you are new to the tool and web stress testing in general. It is useful for quick and dirty test scripts that contain only a couple of script items.

### Browser Record Script Creation

WAS contains a feature that allows you to record all of your browser activity and create a test script. The resulting script can then be used to stress the pages you requested with the browser. WAS does this by using a built-in proxy server that records all activity and pushes it through port 8000 back into the tool.

If you are using Internet Explorer 5.0 or above, there is no setup required for browser record script creation. Selecting the browser record wizard should launch your browser. If you are using a browser other than IE, or a version earlier than IE 5.0, you'll need to manually configure the browser proxy settings. If this is the case, here are the things you should do prior to recording a script:

Clean your browser cache. Since WAS Tool will record everything your browser does it will also record the fact that a browser does not perform a GET on pages that are in its cache. However, this may be the behavior you are interested in recording. If you are using IE, right click the browser icon on your desktop, select **properties**, select the **General** tab and click the **Delete Files** button. While in the browser properties window, select the **Connection** tab and, in the **Proxy server** section, change your browser settings so that the proxy server is pointing to "localhost" and the port used is "8000". Also, uncheck the check box for **Bypass proxy server for local (Intranet) addresses**.

This configuration will work for most scenarios. However, if you are having trouble recording, see the "*Things to Keep in Mind when Recording a Script*" section below. Also, remember that the configuration needs to be reversed after you have finished recording, if you are not using IE5.0

### **Recording Wizard Options**

You can select the desired recording options from Step 1 of 2 in the record wizard. I suggest that you leave these options unchecked unless you have some special purpose script to create. Here is what these features accomplish:

- **Record delay between requests** – records the time between browser clicks, in milliseconds. These delays appear in the delay column of the script item grid.
- **Record browser cookies** – records the cookies that the browser passes to the server as static cookies. Unlike dynamic cookies, static cookies are not assigned to any particular user. They are hard-coded into the header of the script item they pertain to. They will not change during playback. If you want automatic (dynamic) cookies, leave this option unchecked.
- **Record the host header** – records host header information, assuming your browser supports them. If you want automatic host header information passed back and forth with the server during the replay of the script, don't select this option – leave it unchecked.

### **Things to Keep in Mind when Recording a Script**

- You can select the **Change Group** button on the record window at any time to create a new page group as you are recording.
- WAS only supports one server name when you replay the script. Therefore, you should only record browser activity on one server or domain name. If you switch between different domains during a recording session, the playback of the recorded script will not produce the expected results.
- In IE properties, in the **Automatic Configuration** section on the **Connection** tab, click on the **Configuration** button to see if you have an automatic configuration setup URL. This could prevent the WAS record feature from working correctly. Jot down the URL in the **Automatic Configuration** dialog and delete it while you record your script.
- If you are using WAS with a proxy server on your corporate network and you are requesting pages from a site outside of your corporate network, and your company is using Microsoft Proxy Server, you'll need to change the **Logon As** option for the Webtool service to log on as yourself. You may also need to install proxy client software for your proxy server. If using Microsoft Proxy Server, you can install the

Microsoft Windows Proxy Client 2.0. This is also known as the Winsock proxy client and is available on the CD that the proxy server software came on.

- When IE is setup to use a proxy, it will refuse to negotiate NTLM authentication. However, it will work when recording basic authentication. Unless basic authentication is used, turn off authentication on the web server, and then record the script. When you replay the recorded script, turn authentication on the web server back on. WAS will automatically authenticate using the users and passwords you supply. If the script was recorded using basic authentication, WAS uses the same users and passwords used while creating the script.
- You cannot record secure socket layer (SSL) encrypted web sites. The reason is that the data seen by the WAS proxy will be encrypted and so WAS is not able to determine which URL you are requesting. To build an SSL script, you can record the script without SSL. Once your script is built, convert all the requested paths to use SSL by opening the script item details view and selecting the SSL tab. You can enable it for all script items by choosing Apply to all. Another way around this is to turn the recorder off and just hit the site normally, and when you're done you can import the IIS log and build a script that way.

### ***Log File Script Creation***

Another script creation method is the log import feature, which allows you to create a test script from a web server log that is in one of the supported formats. The following log file formats are supported:

- Microsoft IIS Log File Format
- NCSA Common Log File Format
- W3C Extended Log File Format

WAS automatically understands the formats listed above and therefore it is not necessary to choose the format prior to importing a log. Importing from a log file can be useful for tracking down a hard-to-reproduce application error caused by the sequence of page requests. It is also useful for creating a script that requires authentication or SSL encryption, since these types of applications cannot be browser recorded.

If the log file contains more than 3,000 lines, the import process will take a long time and starting a test will also require an inordinate amount of time. For this reason, the log import wizard contains import options that help decrease the number of script items to a workable number, without losing accuracy. Although the size of a script is only limited by the size of RAM on the client machine, it is not recommended that you exceed 3,000 script items for any one script on most hardware.

If there are errors in the log file, these script items display in red. This allows you to quickly scan the imported script for errors.

### ***Creating a Script Using the Object Model***

WAS contains an object model that can be used to create, configure, and start web stress tests. When the test completes, the object model can be used to create custom reports, perhaps an HTML report that is updated daily through automated test cases.

The help file contains a sample of every property and method in the "Microsoft Web Application Stress Object Model" topic. Be sure to review the "Getting started with Web Application Stress scripting" topic before you attempt to create scripts using this object model.

Scripts created using the object model must be hosted in another application such as Windows Scripting Host (WSH), or Visual Basic, or even the Office VBA editor. The tool itself does not

provide an editor. Another thing to keep in mind before you create these scripts is that they cannot be used to take conditional action while a test is running.

To follow is an example of a WAS object model script that creates a new test script with one script item and sets some of the script properties:

```
Sub AddingScriptData()  
    Set objWAS = CreateObject("WAS.EngControl.1")  
    Set objScript = objWAS.Scripts.Add  
  
    With objScript  
        .sName = "My New Script"  
        .ScriptItems.sServer = "localhost"  
        .NumberOfThreads = 4  
        .SocketsPerThread = 2  
        .TestTime = DateAdd("s", 150, 0)  
  
        .Warmup = 0  
        .Cooldown = 0  
        .Notes = "Sample script using Web Application Stress" &  
            " objects."  
        ' add a script item as well  
        .ScriptItems.Add.sURL = "/default.htm"  
    End With  
End Sub
```

## ***Running a Test***

Once you have created a script and configured all the settings, users, and clients, it is time to start the test. Select the Sample Script and choose Run from the Scripts menu, and allow the test to complete. If you recorded a new script with your browser or imported the script from a log file, be sure to change the server name from its default of "localhost" to that of the server or domain you are testing.

## ***Using the Report Data***

Upon completion of a successful test, the master client collects and collates the data from all client machines. This data is then summarized in a report that provides an indication of the best and worst performing pages in a web application.

## ***Analyzing the Results of a Test***

The first place to look is the top-level report view. This displays the global results of a test. Look to see if there were result code errors. If so, the test is probably not valid. Fix these errors and run the script again.

The next thing to examine is the socket errors. It is OK to have a few socket errors over the length of a test but if there are a large number of socket errors, you may have a problem with the script, the stress level with which the test was run, or a slow page. If a page does not respond for two minutes, WAS considers the page unavailable and closes the socket. Since no result code is returned, this page will not display as a result code error. Socket timeouts indicate a page that is not returning under high stress. You can determine which page causes the greatest number of socket errors from the page data statistics.

**You can increase or decrease the socket timeout threshold by changing the following registry key:**  
**HKEY\_LOCAL\_MACHINE\Software\Microsoft\WAS\Timeout.**

The next thing to examine is the test client's node. See if all of your clients were started and used during the test. If not, you may want to fix the problem and re-run the test script.

After this I generally view the Page Summary node to get an indication of which pages are the slowest. If there are lots of them, you can copy the data into Excel and sort it and chart it. You'll need to use the Data, Text to columns feature after pasting the data into Excel.

When a slow page is located in the Page Summary it is a good idea to view the page details because the summary only displays the average time to first and last byte. The Page Data will display the percentiles, which provide a better indication of what happened during the test.

The performance monitor counters you collected over the length of the test are displayed in the Perf Counters node. The counters are also displayed with percentiles. If you would like to view all the counters that were collected over the length of the test, open `hcounters.csv` in Excel.

After determining the slowest page, you can stress test this page alone and apply the mHz/request/second formula, which is explained in help topic "Stress testing overview".

## ***Determining Your Requirements***

The following three methods are the most common techniques for evaluating the performance of web sites in relation to Web Application Stress.

### ***Total Number of Users***

At the time of writing, `www.yahoo.com` was serving over 43 million unique customers per day. This figure is often expressed as unique users per month. WAS should be configured with the appropriate number of users from the Users node. The site must then be stressed for long enough for all users to make at least one request. Each WAS user makes requests in order, so if the site is performing at 10 requests per second then the test must last at least 1000 seconds for a 10,000 user base.

For user base testing, the Stress level and Stress multiplier do not need to be manipulated except to control how long the test runs. WAS can handle 20,000 users on a 64 MB client machine, but may become very slow on start-up with more users. For testing large user bases, it is helpful to scale a million users per month requirement down to thousands of users per hour, and perform the evaluation at that scale.

This evaluation method is useful for sites that store user data using a combination of unique client-side cookies and a backend data store. Due to the fact that WAS saves persistent client cookies, this will properly stress the storage, memory, and CPU requirements of the system. For web sites that do not use authentication or client-side cookies, the other types of performance evaluation are more useful.

### ***Total Concurrent Users***

At the time of writing, the `www.msnbc.com` web site experienced 2,000 or more simultaneous connections at peak times during the day. Concurrent user requirements can be tested exactly, using the WAS stress level multiplied by the stress multiplier. For requirements above a few hundred concurrent users, it may be necessary to enable bandwidth throttling at the 28.8 or 56K level, and to mix static content in with ASP files. Even the best, most efficient code will suffer from queuing if stressed with many concurrent users over a fast network, especially if most requests are for ASP pages.

### ***Peak Request Rate***

At the time of writing, `www.microsoft.com` machines handled 50 ASP requests per second at peak times during the day. There is no way to force WAS to request files at a specific rate; it

will always request them as fast as possible, gated by the web server response time and the bandwidth throttling setting. When evaluating request rate requirements, the best approach is to configure the stress level and stress multiplier such that the concurrent users mirror the expected real world conditions. Then measure the web server request rate against this requirement using performance monitor counters.

## ***Troubleshooting***

If you are having difficulty running a valid test, it may be time to capture a trace. WAS has a built in trace feature that captures all the data sent back and forth between the client and the web server. This is a sort of poor man's network monitor, but I think it is easier to setup and understand. To turn on the tracing feature, add the following registry key, `HKEY_LOCAL_MACHINE\Software\Microsoft\WAS\SessionTrace`, as a DWORD and change its value to 1. Set the threads to 1 before running a stress test with the trace feature on. The reason for this is that the trace will create a new trace file for each thread, and you generally just need one trace file to troubleshoot.

After running the test, a trace file called `trace_1.txt` is created in the folder where the tool is installed. You can double-click on it to open it into notepad.

## **Additional Reading Material**

Here are some additional resources that discuss performance issues related to Internet Information Server and other Microsoft web products.

- <http://www.microsoft.com/TechNet/iis/sol.asp>
- <http://msdn.microsoft.com/library/techart/windnamistakes.htm>
- <http://msdn.microsoft.com/workshop/server/asp/server102599.asp>
- <http://msdn.microsoft.com/workshop/server/asp/server092799.asp>
- <http://msdn.microsoft.com/workshop/server/asp/server122799.asp>
- <http://www.microsoft.com/siteserver/commerce/support/highcapacity.htm>

## **Other Web Stress Tools Available**

There are lots of web stress tools available. Here are several of the main players in this space, in no particular order:

- **Mercury Interactive**, Loadrunner/Winrunner - <http://www.merc-int.com/products/>
- **Radview**, Webload - <http://www.radview.com/webload/>
- **RSW**, eLoad - <http://www.rswsoftware.com/>
- **Segue**, Silkperformer - [http://www.segue.com/html/s\\_solutions/silk/s\\_performer.htm](http://www.segue.com/html/s_solutions/silk/s_performer.htm)
- **Bluecurve**, Dynameasure - <http://www.bluecurve.com/products/products.htm>
- **Rational**, Rational Suite / PerformanceStudio - <http://www.rational.com/products/rs/pstudio/index.jttml>

## **Summary**

Web stress tools are used to help people find the answers to current and projected demands on a web application. Web Application Stress contains the functionality necessary to do modeling and prediction on a wide variety of web server components and help identify the bottlenecked resources. This tool also provides performance metrics that help developers and administrators make informed decisions on what content to deploy.