

TECHNOLOGY



Automation Testing

Timers



A Day in the Life of an Automation Test Engineer

Alex learned about the JMeter tool used for various non-functional testing.

Now, while testing, he is coming across situations where he noticed that he needs to use timers to delay JMeter sending the next request because without timers, it sends requests in a fraction of seconds. This is creating an issue for him. He knows that there is a concept of a timer in JMeter which can resolve his problem.

In this lesson, he will learn how to use timers in JMeter to delay the subsequent request by a certain amount of time which can be configured by adding the value of delay time.



Learning Objectives

By the end of this lesson, you will be able to:

- 🕒 Analyze the use of timers in JMeter test elements
- 🕒 Understand when can a timer be executed in a scope
- 🕒 Remember the method of adding a timer in a test plan
- 🕒 Understand the method involved in pausing a thread in a test plan using random timers



Overview: JMeter Timer

Overview: JMeter Timer

JMeter sends requests without pausing between each request.



This eventually **overwhelms** the user's test server by making too many requests in a short amount of time.



Overview: JMeter Timer

The purpose of the **timer** in test elements is to pause a JMeter Thread representing a virtual user for a certain amount of time.



The main goal of using timers is to simulate a virtual user's **think time**.



Overview: JMeter Timer

A timer can solve the server overload problem.



In real life, visitors do not arrive at a website at the same time, but at different time intervals.
So, timer will help mimic the real-time behavior.

Overview: JMeter Timer

Think times vary mostly.



For example: If a user is watching images, the user usually takes seconds between performing the next actions.

Or, if the user replies to a forum post, it may take minutes to compose the response, and the number of requests, while the user is typing will be minimal or even zero.

Timer Scope and Processing

Just like JMeter Assertions, JMeter timers also have their scope.



Timers are executed before each sampler in their scope. If there is more than one timer in the scope, all the timers will be processed before the sampler occurs.

Timer Scope and Processing

Timer execution times are not recorded, so if the timer pauses the thread for 1 second and the sampler execution time is 3 seconds, 3 seconds will be recorded as the sampler execution time.

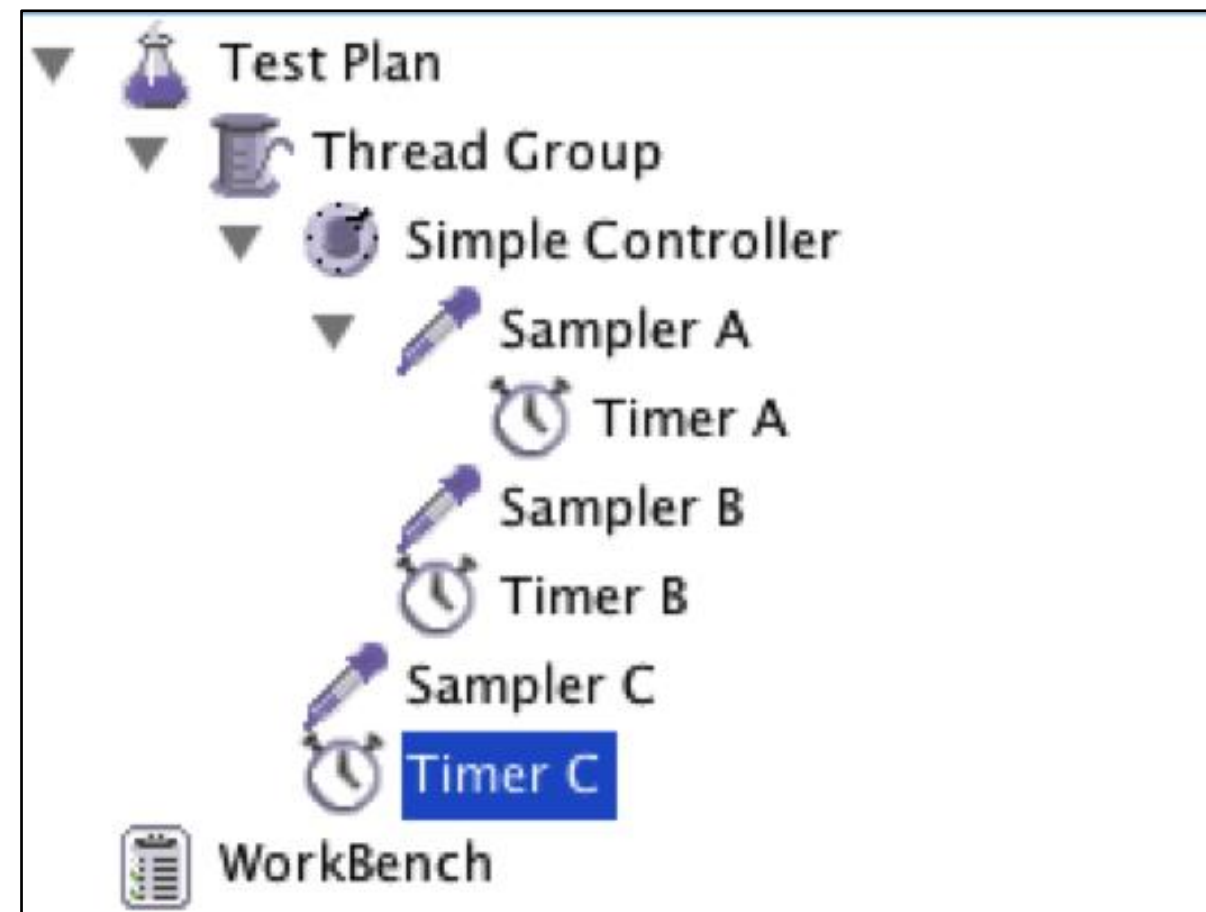


Image source: www.blazemeter.com



Timer Scope and Processing

In the image:



- Timer A - applicable to Sampler A only
- Timer B - applicable to Sampler A and Sampler B
- Timer C - applicable to Sampler A, Sampler B, and Sampler C
- The Timers are executed before the sampler's execution
- The Timer execution time is not added to the sampler execution time

Timer Scope and Processing

If a user needs to insert a delay after a sampler the user can use the Test Action sampler.



The Test Action sampler:

- Doesn't generate the sample result
- Can pause the thread for a static or random amount of time
- Can be used as a parent sampler for any JMeter timer if the delay logic is more complex than a static or random think time

Timer Scope and Processing

The **Test Action** sampler looks like this:

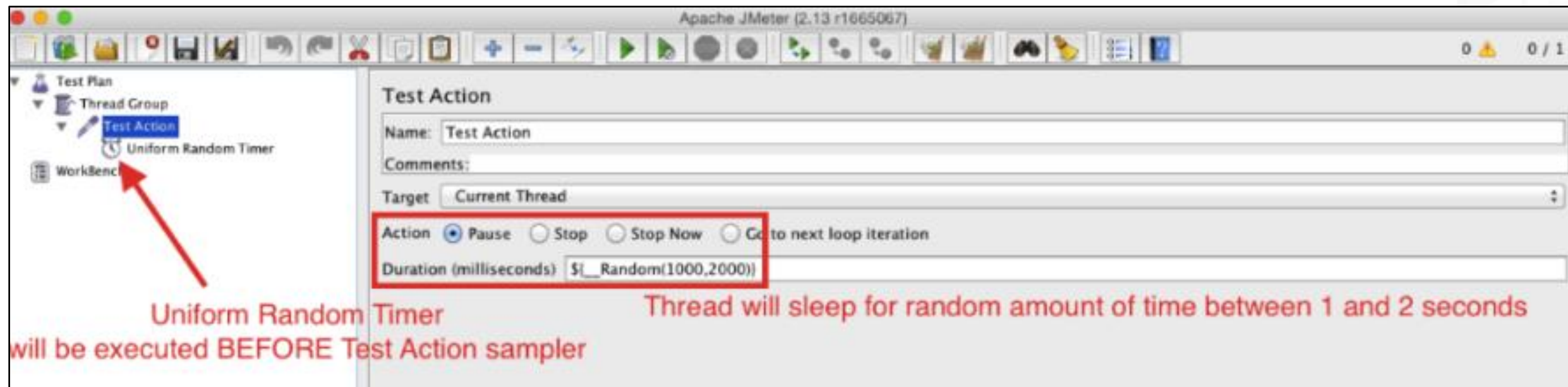


Image source: www.blazemeter.com

Timers

According to the current version of Apache JMeter (which is 5.4), the timer test elements available are:



TEST

- Constant timer
- Uniform random time
- Gaussian random timer
- Poisson random timer

Timers

As per the current version of Apache JMeter (which is 5.4), the timer test elements available are:



- Constant throughput timer
- Synchronizing timer
- BeanShell timer
- BSF timer
- JSR223 timer

Timers

The process of adding timer in test plans:

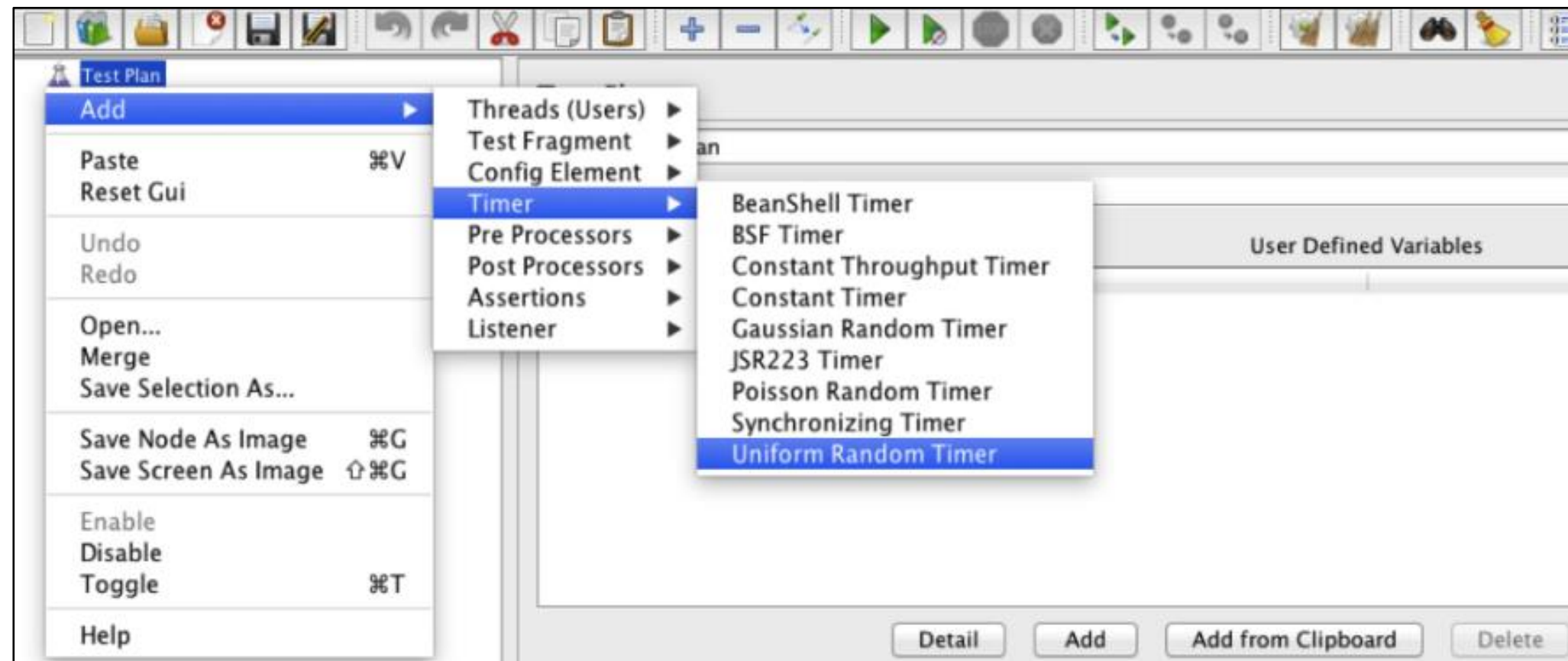
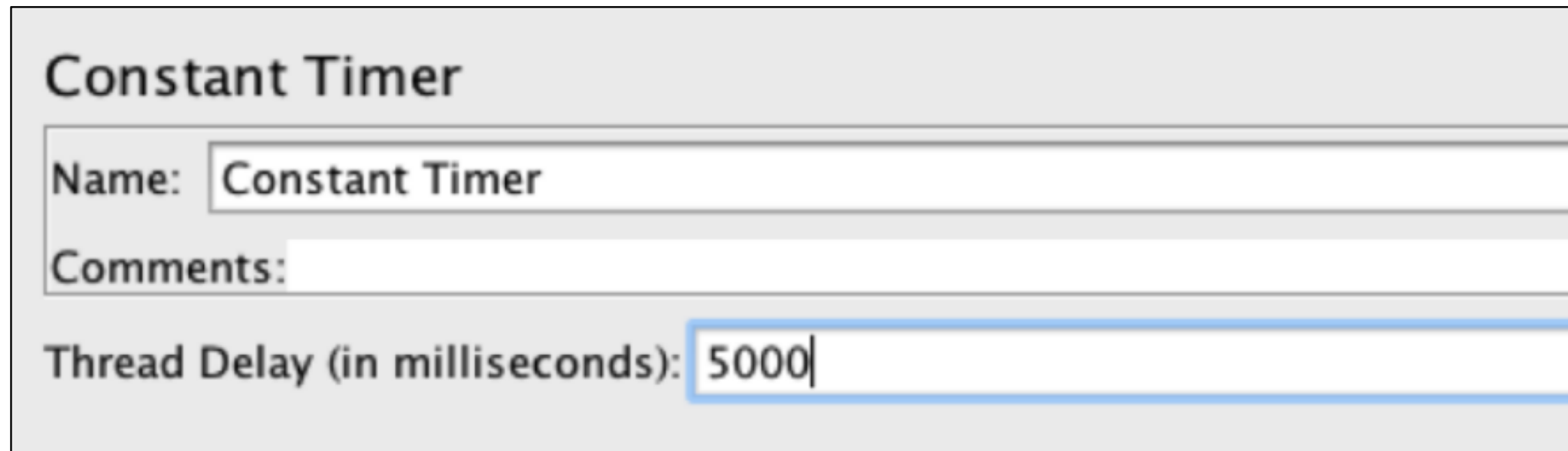


Image source: www.blazemeter.com

Constant Timer

The **constant timer** can be used to pause each thread for the same think time between requests.

A screenshot of the Blazemeter Constant Timer configuration interface. The interface has a light gray background and a title bar at the top that says "Constant Timer". Below the title bar, there are three input fields. The first field is labeled "Name:" and contains the text "Constant Timer". The second field is labeled "Comments:" and is empty. The third field is labeled "Thread Delay (in milliseconds):" and contains the value "5000". The "5000" is highlighted with a blue border.

Constant Timer

Name: Constant Timer

Comments:

Thread Delay (in milliseconds): 5000

Image source: www.blazemeter.com

The thread delay value is in milliseconds. Here the delay is 5000 ms in between each user request.

Constant Timer

In this image, the configuration will add a 5-second delay before the execution of each sampler, which is in the constant timer's scope.



Random Timers - Constant, Gaussian & Poisson Random Timers

Uniform Random Timer

The **uniform random timer** pauses the thread by a factor of:



- The next pseudorandom uniformly-distributed value in range between 0.0 (inclusive) and 1.0 (exclusive)
- Multiplied by **random delay maximum**
- Plus, **constant delay offset**

Uniform Random Timer

The timer pauses affected samplers by a random number of milliseconds in a range from 0 to 99 as the formula will look like $|0.X * 100 + 0|$ where X can be a digit between 0 and 9 inclusively.



The default configuration would look like the below:

- 100 random delay maximum
- 0 constant delay offset

Gaussian Random Timer

Gaussian random timer delays the request for a random amount of time. This timer works on a normal or gaussian distribution function.



Gaussian Random Timer

The gaussian random timer looks like this:

Gaussian Random Timer

Name: Gaussian Random Timer

Comments:

Thread Delay Properties

Deviation (in milliseconds): 100.0

Constant Delay Offset (in milliseconds): 300

- **Name:** It defines the Name of the timer.
- **Comments:** If any, then a user can provide them in this place.

Gaussian Random Timer

The thread delay properties are:



- **Deviation:** The number provided for deviation represents the deviation in delay provided in constant delay offset.
- **Constant delay offset:** Constant delay offset number is the number that shows a constant delay to be added in a random number generated.

Gaussian Random Timer

Here, the deviation (in milliseconds) provided is 100.0 and the constant delay offset (in milliseconds) is 300.



As per the gaussian random timer, the random number generated will be between 200 and 400 as the deviation is of 100 milliseconds.

Poisson Random Timer

The poisson random timer looks like this:

Poisson Random Timer

Name: Poisson Random Timer

Comments:

Thread Delay Properties

Lambda (in milliseconds): 100

Constant Delay Offset (in milliseconds): 300

Image source: www.blazemeter.com



Poisson Random Timer

The poisson random timer is similar to the gaussian random timer. It provides delay time between the requests for a random interval of time.



The total delay in time is the sum of Lambda (in milliseconds) and constant delay offset (in milliseconds). Random number generation is based on Poisson distribution.

Constant Throughput Timer

A constant throughput timer can pause the threads in a certain way that a specific scenario could be implemented.



A goal-oriented scenario is a load test type where the target is not to simulate a certain number of concurrent users but a particular number of requests per second.

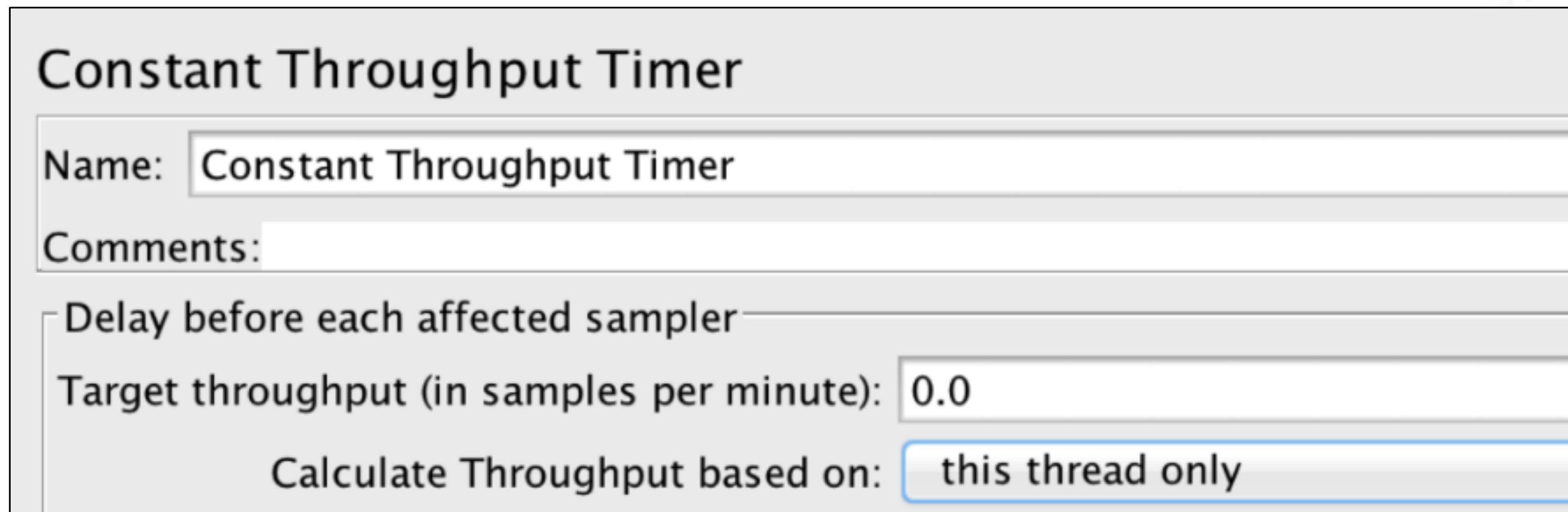
Constant Throughput Timer

A constant throughput timer works precisely on the minute level.



Constant Throughput Timer

It is advised that the user's test should last long enough, and the user can use a reasonable ramp-up time to avoid spikes.



The image shows a configuration window titled "Constant Throughput Timer". It contains several input fields and a dropdown menu. The "Name" field is filled with "Constant Throughput Timer". The "Comments" field is empty. The "Delay before each affected sampler" field is empty. The "Target throughput (in samples per minute)" field is filled with "0.0". The "Calculate Throughput based on:" dropdown menu is set to "this thread only".

Constant Throughput Timer

Name:

Comments:

Delay before each affected sampler:

Target throughput (in samples per minute):

Calculate Throughput based on:

Synchronizing Timer

Synchronizing Timer

The synchronizing timer is different from other timers.



It has only one parameter which is named as the **number of simulated users to Group by**.

Synchronizing Timer

The number provided in this parameter will be the number of threads it will wait for grouping and release.



For example: In case 30 is the number provided in the parameter named **number of simulated users to group by** and the number of threads is 60, in that case, 2 groups of 30 threads will be released.

Synchronizing Timer

In case the number of simulated users to group by is more than the number of threads, then it will hang the test as the timer will not work.



Key Takeaways

- The timer's purpose in test elements is to pause a JMeter thread representing a virtual user for a certain amount of time.
- A timer can solve the server overload problem.
- If there is more than one timer in scope, all the timers will be processed before the sampler occurs.
- Gaussian random timer delays the request for a random amount of time, and it works on a normal or gaussian distribution function.

