**JMeter Thread Group**

Absolutely! A **Thread Group** in JMeter is a crucial element that defines the number of threads (users) to simulate, the ramp-up period, and the loop count for your test. Here’s a breakdown of all the parameters:

**1. Number of Threads (Users)**

This parameter specifies the number of virtual users you want to simulate. Each thread represents one user performing the tasks defined in the test plan.

**2. Ramp-Up Period (in seconds)**

This is the time JMeter will take to start all the threads. For example, if you have 10 threads and a ramp-up period of 100 seconds, JMeter will start 1 thread every 10 seconds.

**3. Loop Count**

This determines how many times the test will be executed. If you set it to 1, each thread will execute the test plan once. If you set it to 5, each thread will execute the test plan five times.

**4. Scheduler**

Enabling this option allows you to set a specific duration for the test execution.

**5. Duration (seconds)**

When the scheduler is enabled, this parameter specifies the total duration for which the test should run. For example, if set to 300 seconds, the test will run for 5 minutes.

**6. Startup Delay (seconds)**

This parameter allows you to delay the start of the test by a specified number of seconds after clicking the start button.

**7. Action to be taken after a Sampler error**

This parameter defines what JMeter should do if an error occurs during the test. Options include:

* **Continue**: Continue with the next sampler.
* **Start Next Thread Loop**: Skip to the next iteration of the loop.
* **Stop Thread**: Stop the current thread.
* **Stop Test**: Stop the entire test.
* **Stop Test Now**: Immediately stop the entire test.

**8. Same user on each iteration**

When this option is enabled, JMeter will use the same user session for each iteration of the loop.

**9. Delay Thread creation until needed**

This option delays the creation of threads until they are actually needed, which can be useful for conserving resources.

**10. Specify Thread Lifetime**

This allows you to define the lifetime of each thread, specifying how long each thread should run before it is terminated.

**11. Thread Group Name**

This is simply the name you assign to the thread group for identification purposes.

By configuring these parameters, you can simulate various load conditions and understand how your application behaves under different scenarios. If you have any more questions or need further clarification, feel free to ask!

**Assertions- Frequently using assertions**

1. Response Assertion

2. Duration Assertion

3. Size Assertion

4. HTML Assertion

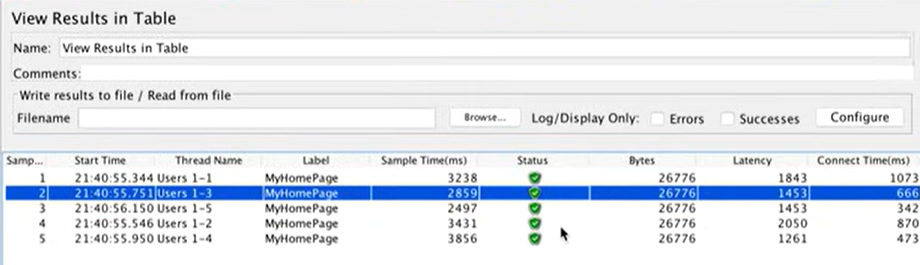
5. XML Assertion

6. XPATH Assertion

**Listners – Elements that gather information about the test executed.**

1. **View results in table**

{samples, start time, ThreadName1-1,Label,sampletime(ms),Status,Latency,connect time}



1. **View results in tree**

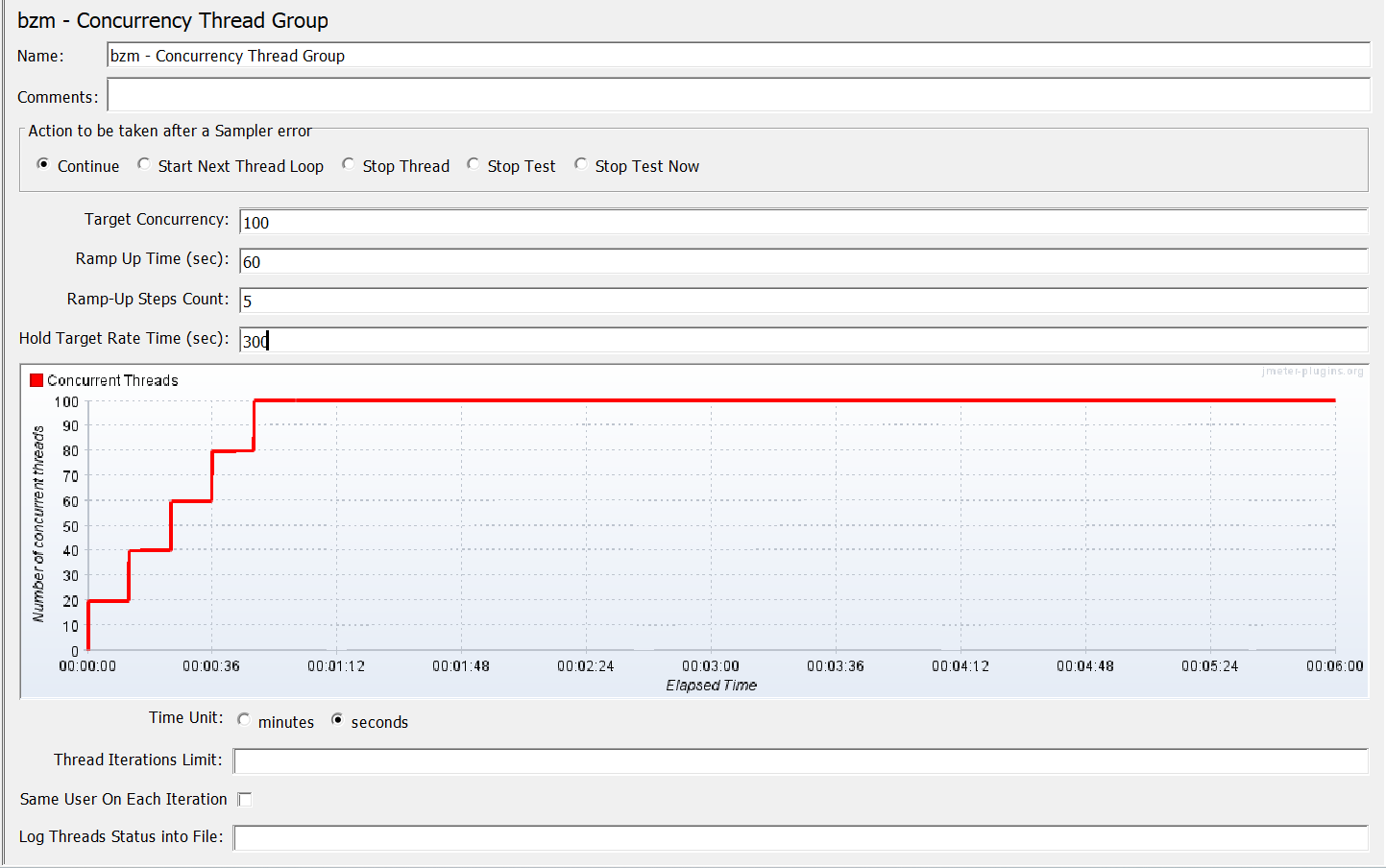
**Thread Group: Concurrency thread group**

The **Concurrency Thread Group** is part of the **JMeter Plugins** package (jpgc-casutg) and is used to simulate a **precise number of concurrent virtual users** for a defined duration and ramp-up strategy.

It gives you **more realistic and production-like load profiles** compared to the default Thread Group.

Diff. Between Normal thread group and Concurrency thread group,

| **Default Thread Group** | **Concurrency Thread Group** |
| --- | --- |
| Based on number of **iterations per thread** | Based on **concurrent active users** |
| Less control over steady-state load | Full control over **target concurrency** |
| Limited ramp-up control | Advanced **ramp-up and hold-time** handling |

****

| **Parameter** | **Meaning** |
| --- | --- |
| **Target Concurrency** | Total number of virtual users (threads) to maintain during the test |
| **Ramp-Up Time (sec)** | Time to ramp up from 0 to target concurrency |
| **Ramp-Up Steps Count** | How many steps to break ramp-up into (e.g., gradual steps) |
| **Hold Target Rate Time (sec)** | How long to maintain the full target concurrency |
| **Time to Hold Load (sec)** | Synonym for above (in newer versions) |
| **Iterations Limit** | (Optional) Max number of iterations per thread |
| **Thread Lifetime (sec)** | Time each thread should live (optional) |

**Example: Parameter Configures,**

| **Parameter** | **Value** |
| --- | --- |
| Target Concurrency | 100 users |
| Ramp-Up Time | 60 seconds |
| Steps | 5 |
| Hold Target Rate Time | 300 seconds |

**Calculation for the configuration**:

Within 60 Secs 100 users will be processed in below calculations,

**Users per step = Target Concurrency / Ramp-Up Steps**

= 100 / 5

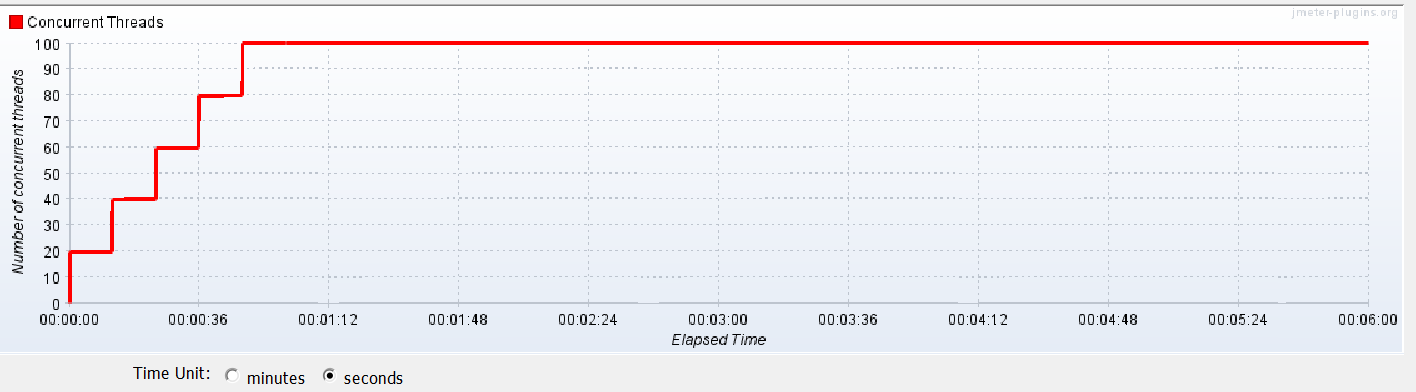
= **20** users per step

**Time per step = Ramp-Up Time / Ramp-Up Steps**

= 60 seconds / 5

= **12** seconds per step

**At 60th second all the 100 users have been activated, below is how it will simulate.**

****

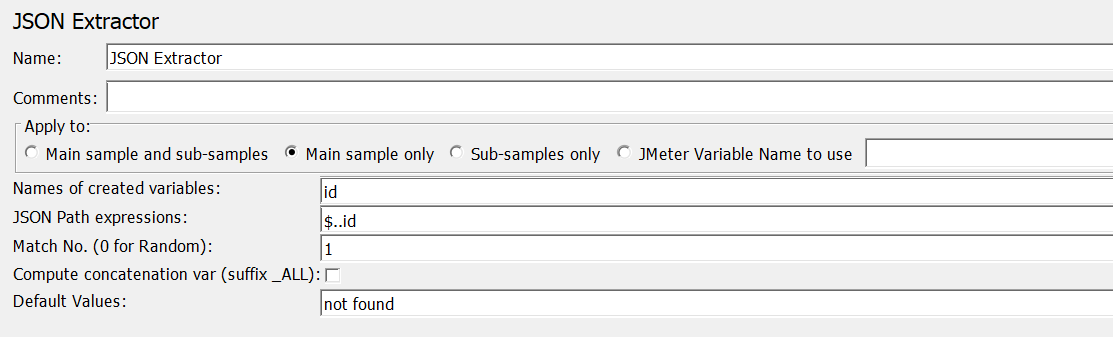
**Hold Target Rate Time**

It means JMeter will **maintain those 100 users** for **5 minutes** before it stops the test

**Extractors:**

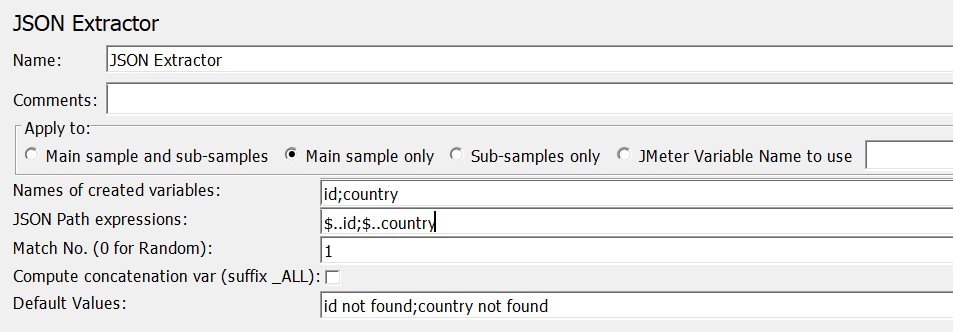
**Json Extractor-**

Below is the parameter



If you need to **create multiple variables in a single Extractor** below is the format:

**Json Extractor: -** To extract two variables at the same time



Trying to fetch **id and Country.**

| **Field** | **What it tells JMeter** |
| --- | --- |
| id;country | Create 2 variables: ${id} and ${country} |
| $..id;$..country | For ${id}, extract the **first** match from $..id. For ${country}, extract the **first** match from $..country |
| Match Numbers = 1 | Means: extract the **first** match for each expression separately (not from the same object!) |

JMeter fetching the first variable correctly i.e Id and second variable i.e. Country its fetching randomly

If the JSON structure is **nested or inconsistent**,

Now $..country will search **recursively** and still collect all country values, but **JMeter doesn’t know which one belongs to which id**.

So:

* ${id} → 1
* ${country} → might still be Mozambique, or **some other one**, depending on how the parser processes internally.

**Correct Approach:**

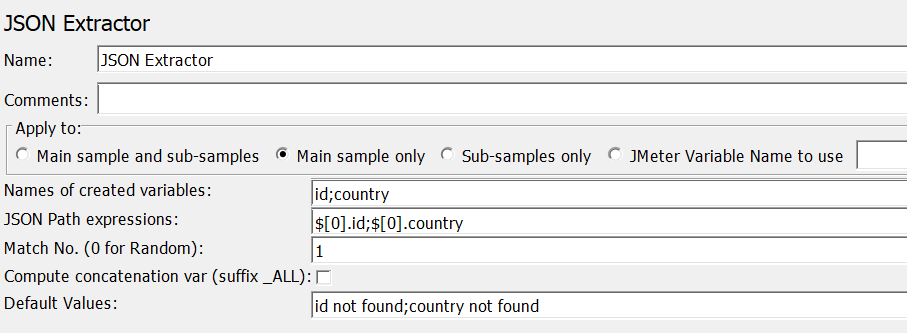
JSON Paths: $[0].id;$[0].country

$[0].id 🡪 Gets the id of the first user.

$[0].country 🡪 Gets the country of the first user

| **Expression** | **What it does** |
| --- | --- |
| $..id | Collects all id fields anywhere in the JSON |
| $..country | Collects all country fields anywhere |
| $[0].id | Gets the id of the first user |
| $[0].country | Gets the country of the first user |
| $[\*].id with Match -1 | Gets **all ids** |
| $[\*].country with Match -1 | Gets **all countries**, in order matching the ids |

**Updated Json Extractor-** To extract two variable’s first value at the same time.



**$.. is like** "search everywhere."  
**$[0] is like** "go to the first person, and get his info."

**Use $[0].id;$[0].country** to make sure both values are **from the same object**.

**What if ? I gave { Match Number = -1 }**

| **Match Number** | **Meaning** | **What JMeter Does** |
| --- | --- | --- |
| 1 | First match | Extracts only the **first occurrence** |
| 2, 3, etc. | Nth match | Extracts the **Nth occurrence** |
| 0 | Random match | JMeter picks a **random occurrence** from the matching values |
| -1 | Extract **all matches** | JMeter extracts **all occurrences** and creates indexed variables |

| **Match Number** | **When to Use** |
| --- | --- |
| 1 | When you only want the **first match** (default) |
| 0 | When **any one random value** is okay |
| -1 | When you want to **extract all matching values** (e.g., for loops or comparisons) |

**Output:**

| **JMeter Variable** | **Value** |
| --- | --- |
| ${id\_1} | 1 |
| ${country\_1} | Mozambique |
| ${id\_2} | 2 |
| ${country\_2} | Gabon |
| ${id\_3} | 3 |
| ${country\_3} | Micronesia |
| ... | ... |
| ${id\_10} | 10 |
| ${country\_10} | Hong Kong |
| ${id\_matchNr} | 10 |
| ${country\_matchNr} | 10 |

**Best practice while using Match Count as -1 , we can use for each and loop controllers effectively.**

### What happens if you set Match Number = -2 or or any negative number other than -1?

* JMeter will **treat it as invalid**.
* It will **not extract any value** for your variable.
* It may silently fail or log a warning in the **JMeter console/logs** like:
* [JSON Extractor] Invalid match number: -2. No variable extracted.

**Also, we can use the JMeter variable Json assertion if we using the Json extractor in the script.**