**Spike Arrest with time limit in Apigee**

To **block a user temporarily** in Apigee after they hit a **Spike Arrest limit**, you need to combine **Spike Arrest**, **Quota**, and **Key-Value Maps (KVMs)** or **caches** to simulate a **temporary lockout** mechanism.

Here’s a step-by-step guide to implementing this behavior:

**Goal**

If a user exceeds the allowed spike arrest limit:

* Block further requests from that user **for a defined period** (e.g., 5 minutes)
* After that time, allow them to access again

**Step-by-Step Implementation**

**Step 1: Identify the User**

Use an identifier such as:

* API Key (client\_id)
* IP address
* OAuth token
* Custom header (x-user-id, etc.)

Example (extract API key):

<ExtractVariables name="ExtractClientID">

<Header name="x-api-key">

<Variable name="client\_id"/>

</Header>

</ExtractVariables>

**Step 2: Check if User is Blocked**

Use a **LookupCache** or **KVM** to check if the user is already blocked.

<LookupCache name="CheckBlockStatus">

<CacheKey>

<KeyFragment ref="client\_id"/>

</CacheKey>

<Scope>Exclusive</Scope>

<CacheResource>block-cache</CacheResource>

</LookupCache>

Then check:

<Condition>lookupcache.CheckBlockStatus.cachehit = true</Condition>

If true, return a 429 response:

<RaiseFault name="UserTemporarilyBlocked">

<FaultResponse>

<Set>

<StatusCode>429</StatusCode>

<ReasonPhrase>Too Many Requests</ReasonPhrase>

<Payload contentType="application/json">

{"error": "You are temporarily blocked. Try again later."}

</Payload>

</Set>

</FaultResponse>

</RaiseFault>

**Step 3: Apply Spike Arrest**

Apply a standard Spike Arrest policy (e.g., 3 requests per second):

<SpikeArrest name="SpikeArrestPerUser">

<Identifier ref="client\_id"/>

<Rate>3ps</Rate>

</SpikeArrest>

**Step 4: Block the User if SpikeArrest Throws an Error**

Use a FaultRule or Flow to handle the spike arrest violation. When the Spike Arrest fails, **store a flag** in a **cache** or **KVM** to block the user.

Example: AssignMessage to store a blocking flag

<AssignMessage name="SetBlockKey">

<AssignVariable>

<Name>block\_key</Name>

<Value>{client\_id}</Value>

</AssignVariable>

<AssignTo createNew="false" type="request"/>

</AssignMessage>

Then use:

<PopulateCache name="StoreBlockFlag">

<CacheKey>

<KeyFragment ref="block\_key"/>

</CacheKey>

<Scope>Exclusive</Scope>

<ExpirySettings>

<TimeoutInSec>300</TimeoutInSec> <!-- Block for 5 minutes -->

</ExpirySettings>

<CacheResource>block-cache</CacheResource>

</PopulateCache>

**Step 5: Add Cache Resource**

In your apiproxy/resources/cache folder, define a block-cache resource in apiproxy.xml:

<Cache name="block-cache">

<ExpirySettings>

<TimeoutInSec>600</TimeoutInSec>

</ExpirySettings>

</Cache>

**Summary of Flow**

| **Step** | **Action** |
| --- | --- |
| 1. Extract user ID | From API key or header |
| 2. Check block status | Lookup cache/KVM for existing block |
| 3. Apply spike arrest | Enforce rate limit using identifier |
| 4. On spike error | Populate cache/KVM to block user |
| 5. Respond to blocked | Return 429 if user is blocked |