

Spike Arrest Policy:

The Spike Arrest policy protects against traffic spikes. It throttles the number of requests processed by an API proxy and sent to a backend, protecting against performance logs and downtimes.

where:

you can attach this policy anywhere in the flow, we recommend that you attach it in the following location so that it can provide spike protection at the immediate entry point of your API proxy i.e PreFlow.

Following are elements and attributes you can configure on this policy.

```
<SpikeArrest async="false" continueOnError="false" enabled="true" name="Spike-Arrest-1">
  <DisplayName>Custom label used in UI</DisplayName>
  <Rate>30ps</Rate>
  <Identifier ref="request.header.some-header-name"/>
  <MessageWeight ref="request.header.weight"/>
</SpikeArrest>
```

<Rate> element

Specifies the rate at which to limit traffic spikes (or bursts). Specify a number of requests that are allowed in per minute or per second intervals.

Ex:

```
<Rate>10ps</Rate>
```

```
<Rate>30pm</Rate>
```

{int}ps (number of requests per second, smoothed into intervals of milliseconds)

{int}pm (number of requests per minute, smoothed into intervals of seconds)

How spike arrest works

- Generally protect against traffic spikes rather than as a way to limit traffic to a specific number of requests.
- The runtime Spike Arrest behavior differs from what you might expect to see from the literal per-minute or per-second values you enter.

Per-minute rates get smoothed into full requests allowed in intervals of **seconds**.

For example, 30pm gets smoothed like this:

60 seconds (1 minute) / 30pm = 2-second intervals, or 1 request allowed every 2 seconds. A second request inside of 2 seconds will fail. Also, a 31st request within a minute will fail.

Per-second rates get smoothed into full requests allowed in intervals of **milliseconds**.

For example, 10ps gets smoothed like this:

1000 milliseconds (1 second) / 10ps = 100-millisecond intervals, or 1 request allowed every 100 milliseconds. A second request inside of 100ms will fail. Also, an 11th request within a second will fail.

Step 1:

Initially, create an API Proxy by providing WSDL and Project Base path.

New API Proxy

1 Choose Your Starting Point

Starting Point Type * ☐ Backend Service ☐ API Bundle ☒ WSDL ☐ No Target ☐ New Node.js ☐ Existing Node.js

WSDL Source * [Change WSDL Source](#)

API Proxy Type * ☒ REST to SOAP to REST ☐ Pass-Through SOAP

API Proxy Discovered from WSDL
6 Operations

Include	WSDL Operation	Description	HTTP Method	REST API Path	REST API Parameters
<input checked="" type="checkbox"/>	GetWeather	Get weather report for all major cities around the world.	GET	weather	CityName, CountryName
<input checked="" type="checkbox"/>	GetCitiesByCountry	Get all major cities by country name(full / part).	GET	citiesbycountry	CountryName
Port Type: GlobalWeatherHttpGet					
<input type="checkbox"/>	GetWeather	Get weather report for all major cities around the world.	GET	weather	CityName, CountryName

2 Identify Your API Proxy

Name *
Valid characters are letters, numbers, dash (-), and underscore (_).

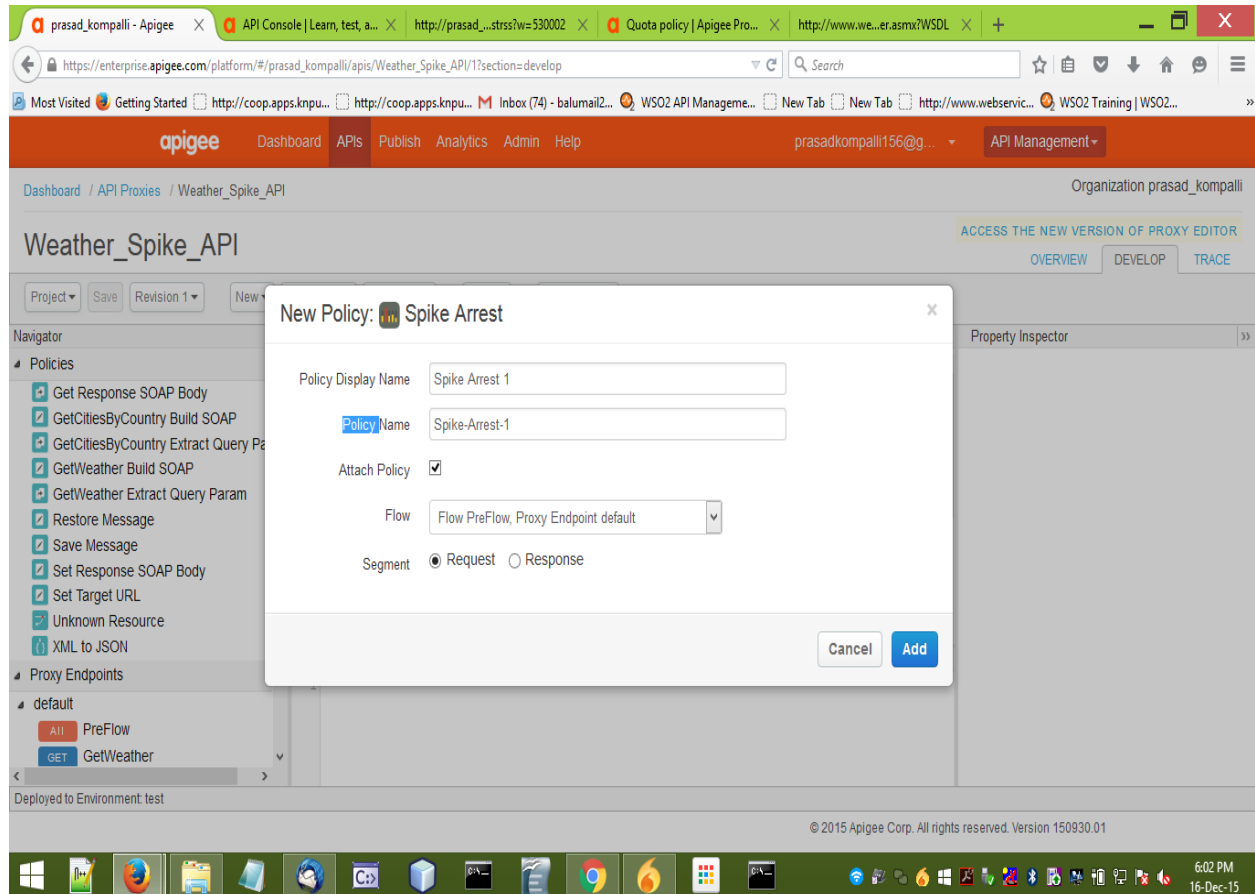
Project Base Path *
A path component that uniquely identifies this API proxy. The public-facing URL of this API proxy is comprised of your organization name, an environment where this API proxy is deployed, and this Base Path. Example URL http://prasad_kompalli-test.apigee.net/v1/weather_spike

Description

3 Add Features

Step 2 :

In order to protect against traffic spikes and user based on the hits, select Spike Arrest Policy in New Policy tab.



Step 3:

Here select the *Spike Arrest Policy* and edit the code and specify the *Rate* based on our requirement.

The screenshot shows the Apigee API Management console interface. The top navigation bar includes the Apigee logo and links to Dashboard, APIs, Publish, Analytics, Admin, and Help. The user is logged in as 'prasadkompalli156@g...'. The current view is for the 'Weather_Spike_API' under the 'API Proxies' section.

The main content area displays the 'Weather_Spike_API' configuration. On the left, a 'Policies' list includes various actions like 'Get Response SOAP Body', 'GetCitiesByCountry Build SOAP', etc. The central pane shows the 'Map: Endpoint default, Flow PreFlow' configuration. A 'Spike Arrest 1' policy is applied, and its XML configuration is shown:

```

1 <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
2 <SpikeArrest async="false" continueOnError="false" enabled="true">
3   <DisplayName>Spike Arrest 1</DisplayName>
4   <Properties/>
5   <Identifier ref="request.header.some-header-name"/>
6   <MessageWeight ref="request.header.weight"/>
7   <Rate>10pm</Rate>
8 </SpikeArrest>

```

On the right, the 'Property Inspector: Spike-Arrest-1' shows the following properties:

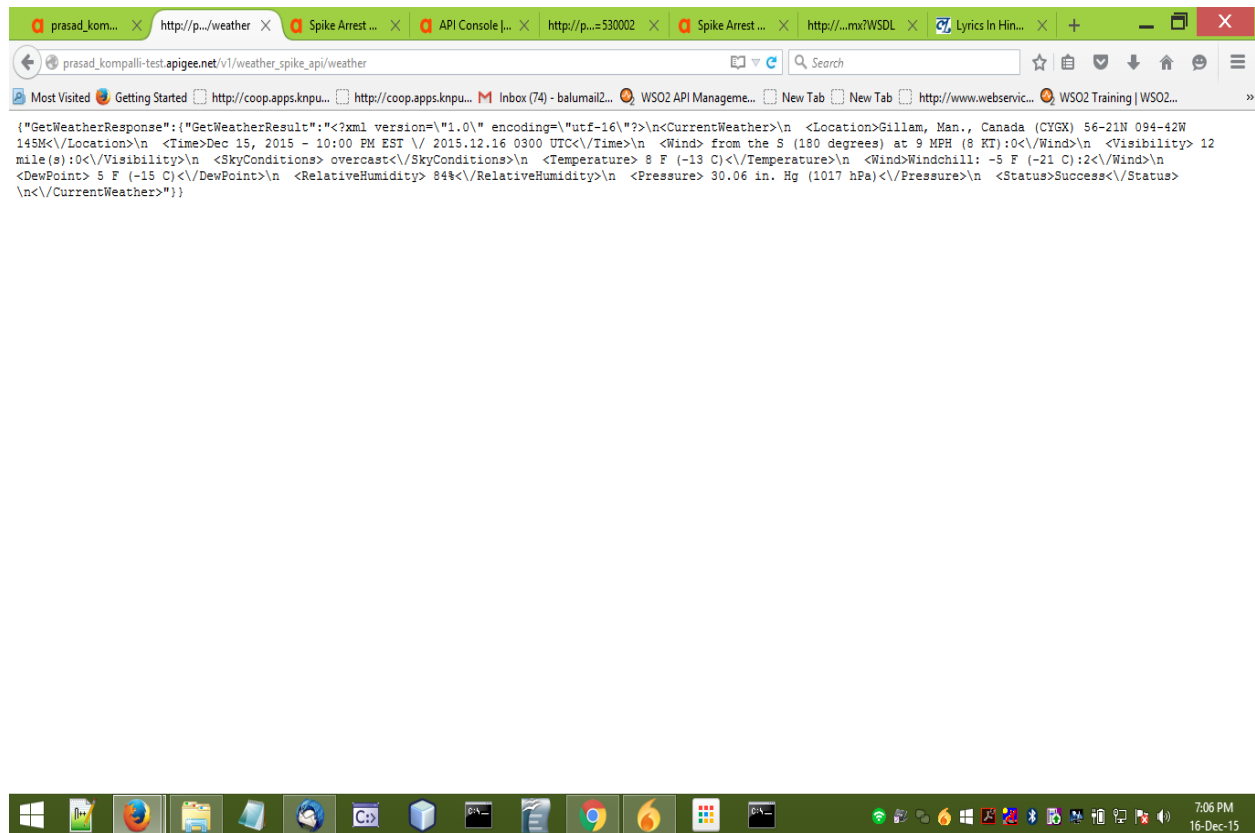
Property	Value
SpikeArrest async	false
SpikeArrest continueOnError	false
SpikeArrest enabled	true
SpikeArrest name	Spike-Arrest
DisplayName	Spike Arrest
Properties	
Identifier ref	request.header.some-header-name
Identifier	
MessageWeight ref	request.header.weight
MessageWeight	

The bottom status bar indicates the policy is 'Deployed to Environment: test' and shows the copyright notice: '© 2015 Apigee Corp. All rights reserved. Version 150930.01'.

Step 4 :

Change the *Rate* based on our requirement.

In this example, it is changed to 10 requests per minute. i.e It spikes 1 request per 6 seconds.



Step 6:

After hitting the URL for 10 times, it will show an error as follows. And within 6 seconds it avoids second request.



After completion of 1 minute the data again show and follow the spike arrest policy.