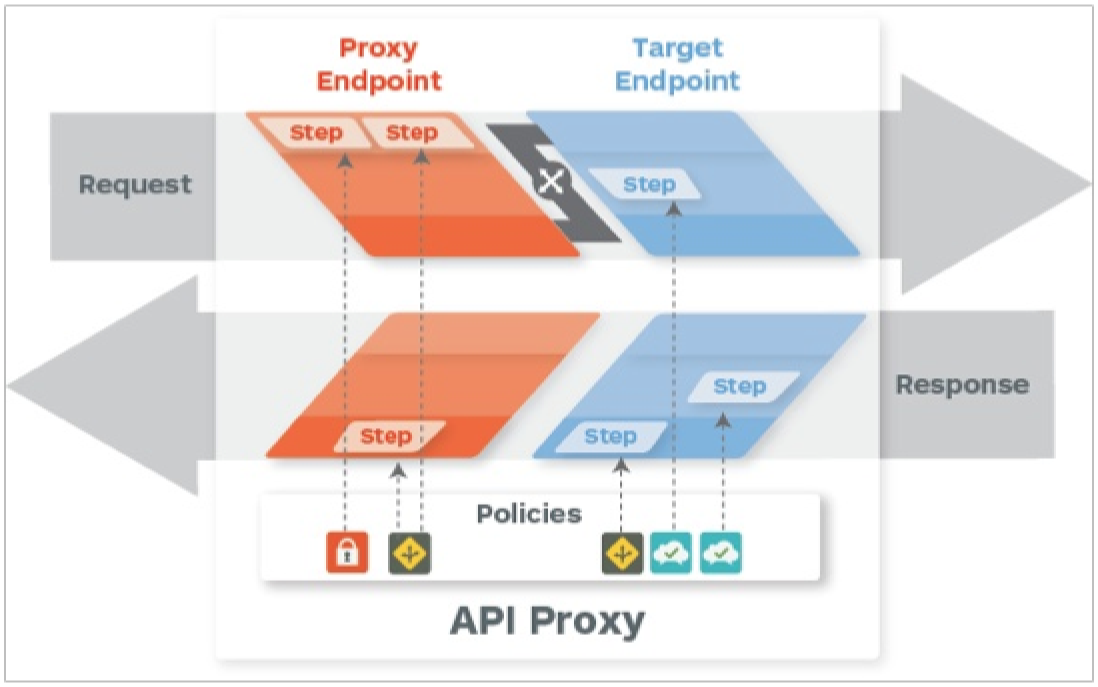
Lab 1: API Services - Creating an API Proxy

Overview
Apigee Edge enables you to quickly expose backend services as APIs. You do this by creating an API proxy that provides a facade for the backend service that you want to expose. The API proxy that you create on Edge decouples your backend service implementation from the API that developers consume. This shields developers from future changes to your backend services. As you update backend services, developers, insulated from those changes, can continue to call the API uninterrupted.



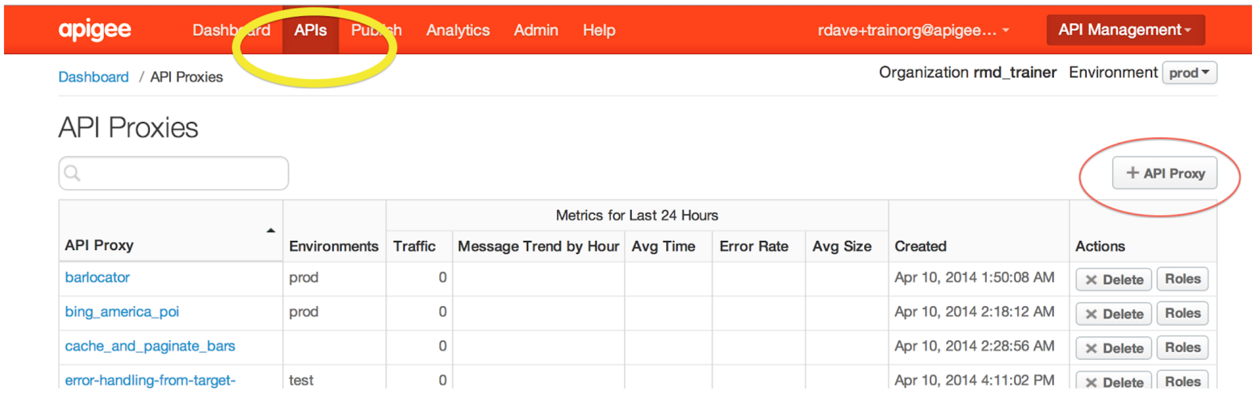
In this lab you will get a first-hand exposure of configuring proxies using a wizard from the user interface. You will build an API Proxy on Apigee Edge and add a few policies including caching. You will also learn to use the tracing tool to perform troubleshooting on the proxy as needed during development

Objective
After this In this lab you should be able to create API proxies on Apigee Edge for different targets and be able to configure simple policies.

Estimated Time: 45 mins

---

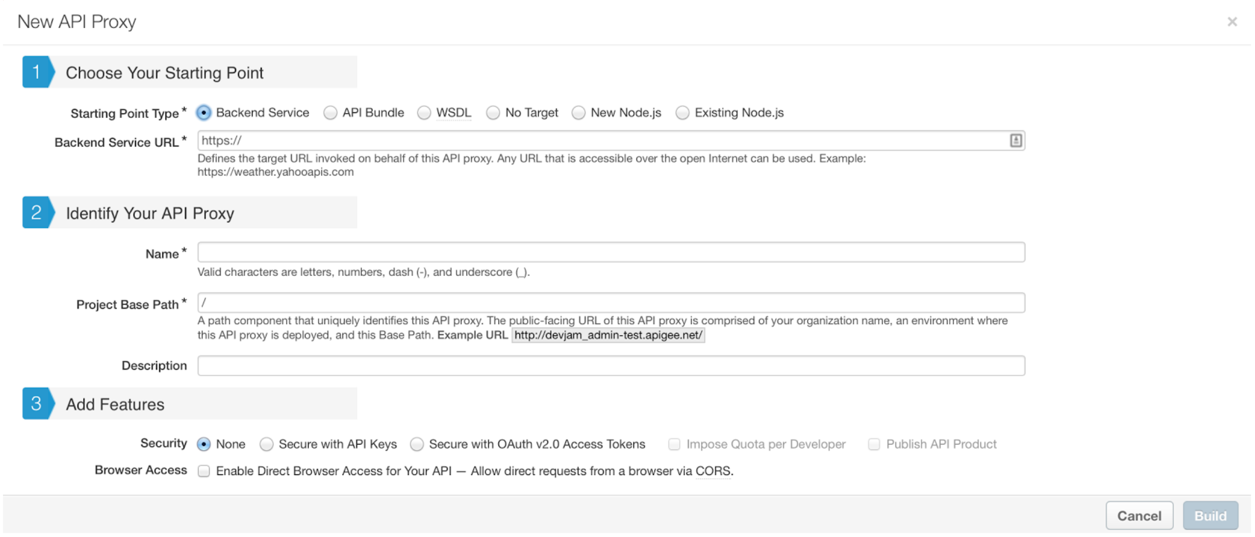
1. Login to the Apigee Edge Management User Interface (Management UI). On the top menu, click on the APIs item and you will see the API Proxies page (as below). When on that page click on the '+ API Proxy' button on the top right.



API Proxies

API Proxy	Environments	Metrics for Last 24 Hours						Created	Actions
		Traffic	Message Trend by Hour	Avg Time	Error Rate	Avg Size			
barlocator	prod	0						Apr 10, 2014 1:50:08 AM	✕ Delete Roles
bing_america_poi	prod	0						Apr 10, 2014 2:18:12 AM	✕ Delete Roles
cache_and_paginate_bars		0						Apr 10, 2014 2:28:56 AM	✕ Delete Roles
error-handling-from-target-	test	0						Apr 10, 2014 4:11:02 PM	✕ Delete Roles

2. Define the API Proxy / Facade



1 Choose Your Starting Point

Starting Point Type * [✓ Backend Service] API Bundle WSDL No Target New Node.js Existing Node.js

Backend Service URL * https://
Defines the target URL invoked on behalf of this API proxy. Any URL that is accessible over the open Internet can be used. Example: https://weather.yahooapis.com

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://devjam_admin-test.apigee.net/

Description

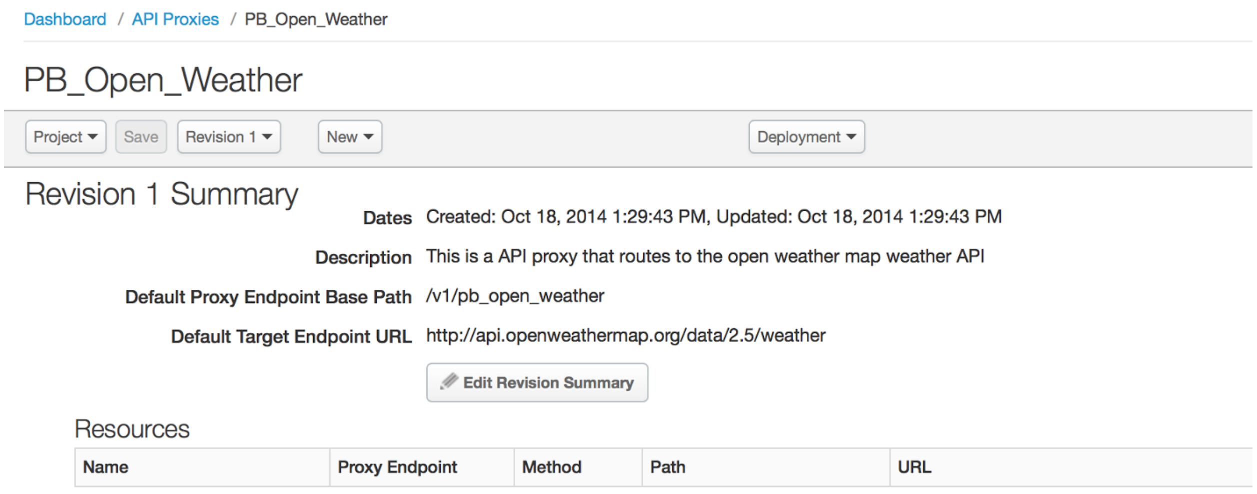
3 Add Features

Security [✓ None] Secure with API Keys Secure with OAuth v2.0 Access Tokens Impose Quota per Developer Publish API Product

Browser Access Enable Direct Browser Access for Your API — Allow direct requests from a browser via CORS.

a. Backend Service URL: http://api.openweathermap.org/data/2.5/weather
b. Name: **{Your Initials}**`_Open_Weather`
c. Base Path: Change the base path of the proxy to `/v1/`**{your initials}**`_open_weather` (It's a good idea to start including versioning as part of API development best practices).
d. Description: `This is a API proxy that routes to the open weather map weather API`
e. Do not select any other options

f. Click on 'Build' to create the API Proxy, once built click on 'Done'
g. You should see the following:



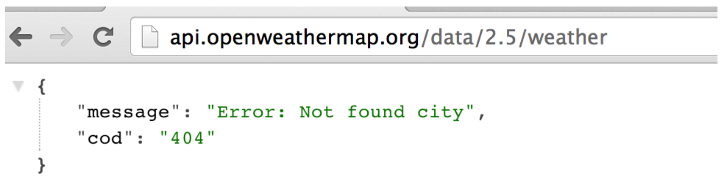
Revision 1 Summary

Dates	Created: Oct 18, 2014 1:29:43 PM, Updated: Oct 18, 2014 1:29:43 PM
Description	This is a API proxy that routes to the open weather map weather API
Default Proxy Endpoint Base Path	/v1/pb_open_weather
Default Target Endpoint URL	http://api.openweathermap.org/data/2.5/weather

Resources

Name	Proxy Endpoint	Method	Path	URL

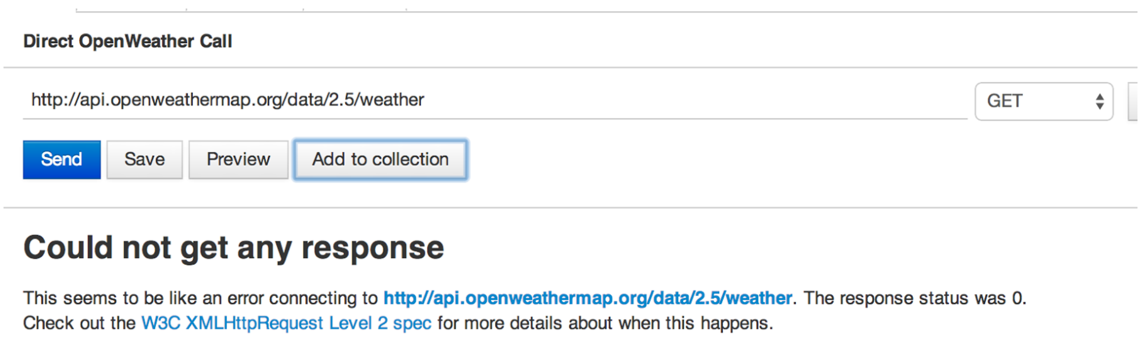
3. Lets do a quick test of the backend target endpoint. Invoke it a browser; you will see an error:
   a.



api.openweathermap.org/data/2.5/weather

```
{
    "message": "Error: Not found city",
    "cod": "404"
}
```

   b. Or, Invoke it from the POSTMAN (or curl, REST client, etc.) tool; you will see an error (POSTMAN is a popular REST client that is available as an Chrome extension. It make is easy to test APIs, set query params, headers and variety of authentication schemes. You are free to use any tool of your choice.)



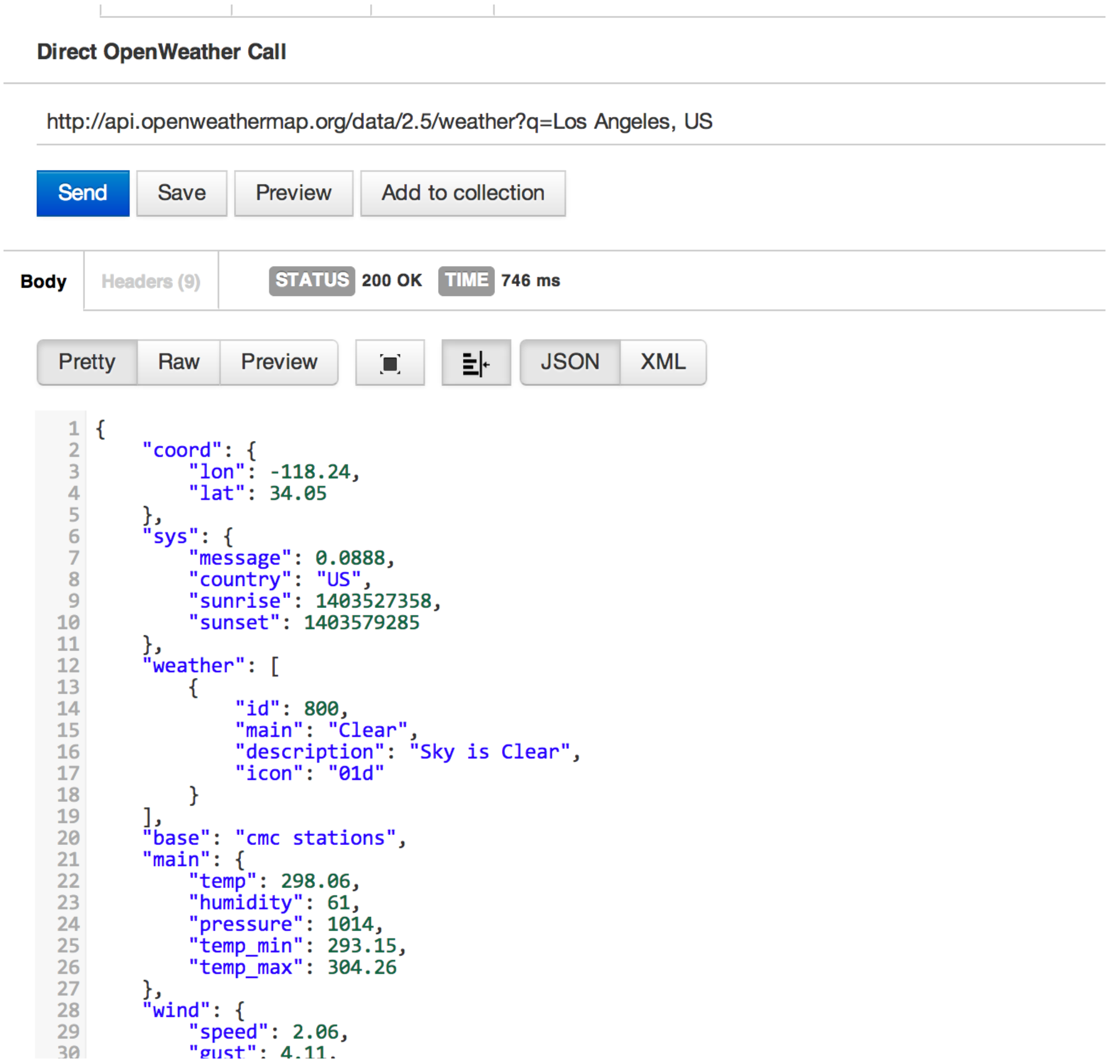
Direct OpenWeather Call

http://api.openweathermap.org/data/2.5/weather

Could not get any response

This seems to be like an error connecting to http://api.openweathermap.org/data/2.5/weather. The response status was 0.
Check out the W3C XMLHttpRequest Level 2 spec for more details about when this happens.

c. Now, specify the city and country at the end of the URL for e.g. q=Los Angeles, USA from POSTMAN and you will see:



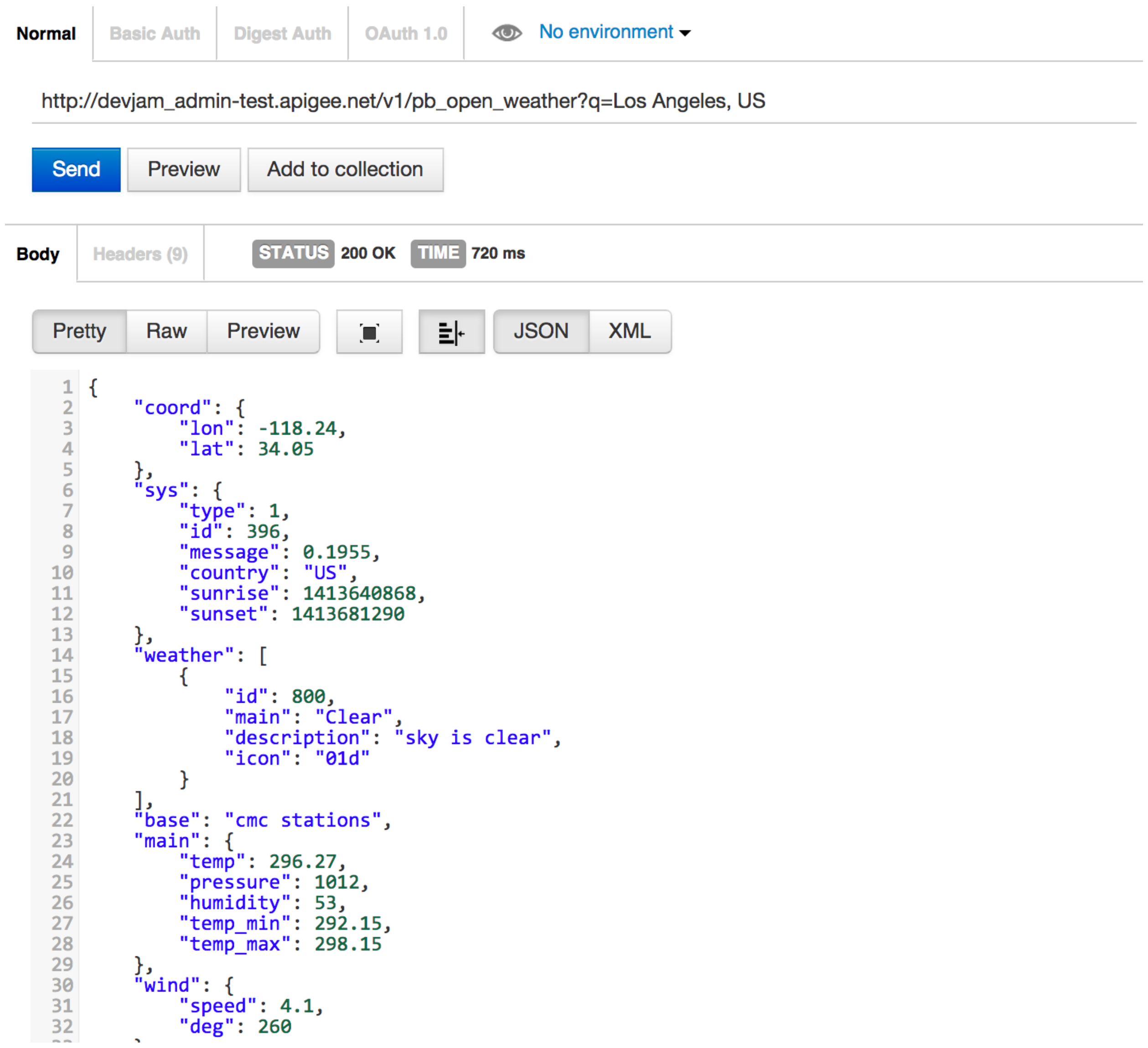
Direct OpenWeather Call

http://api.openweathermap.org/data/2.5/weather?q=Los Angeles, US

```
{
    "coord": {
        "lon": -118.24,
        "lat": 34.05
    },
    "sys": {
        "message": 0.0888,
        "country": "US",
        "sunrise": 1403527358,
        "sunset": 1403579285
    },
    "weather": [
        {
            "id": 800,
            "main": "Clear",
            "description": "Sky is Clear",
            "icon": "01d"
        }
    ],
    "base": "cmc stations",
    "main": {
        "temp": 298.06,
        "humidity": 61,
        "pressure": 1014,
        "temp_min": 293.15,
        "temp_max": 304.26
    },
    "wind": {
        "speed": 2.06,
        "gust": 4.11,
```

4. Now, let's call the API Proxy you just created
    a. You will issue '`http://{`**`org_name`**`}-{`**`env_name`**`}.apigee.net/v1/`**`{your initials}`**`_open_weather?q=Los Angeles, US`' from POSTMAN

    b. And, you will see:

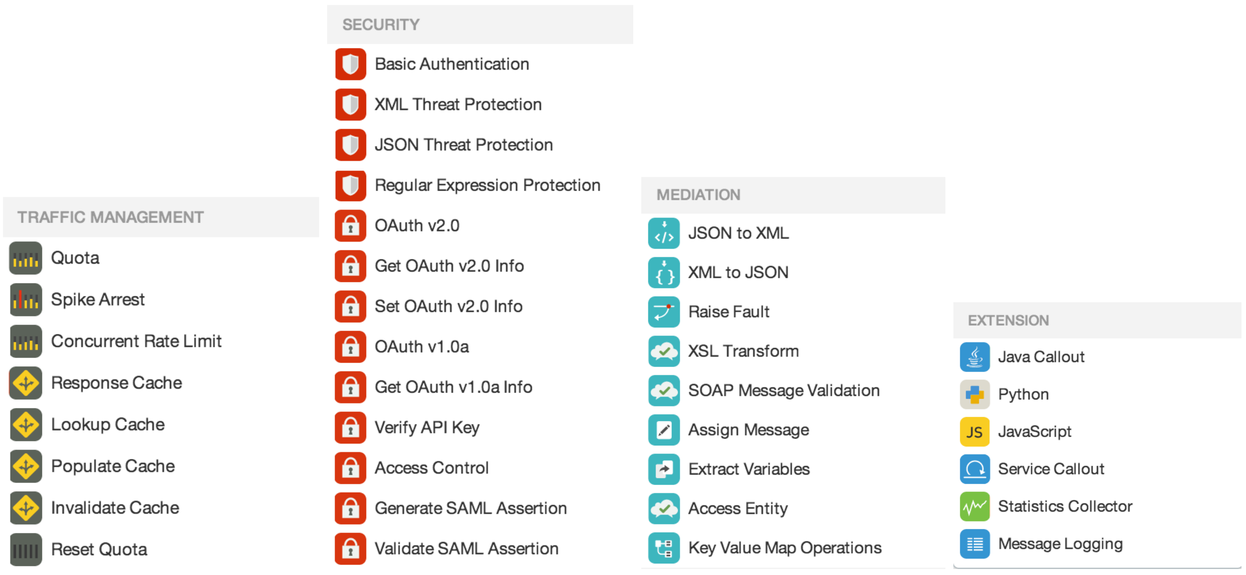


http://devjam_admin-test.apigee.net/v1/pb_open_weather?q=Los Angeles, US

STATUS 200 OK TIME 720 ms

```
{
    "coord": {
        "lon": -118.24,
        "lat": 34.05
    },
    "sys": {
        "type": 1,
        "id": 396,
        "message": 0.1955,
        "country": "US",
        "sunrise": 1413640868,
        "sunset": 1413681290
    },
    "weather": [
        {
            "id": 800,
            "main": "Clear",
            "description": "sky is clear",
            "icon": "01d"
        }
    ],
    "base": "cmc stations",
    "main": {
        "temp": 296.27,
        "pressure": 1012,
        "humidity": 53,
        "temp_min": 292.15,
        "temp_max": 298.15
    },
    "wind": {
        "speed": 4.1,
        "deg": 260
```

5. See the picture below for all available policies that can be attached to proxies.


SECURITY
Basic Authentication
XML Threat Protection
JSON Threat Protection
Regular Expression Protection
OAuth v2.0
Get OAuth v2.0 Info
Set OAuth v2.0 Info
OAuth v1.0a
Get OAuth v1.0a Info
Verify API Key
Access Control
Generate SAML Assertion
Validate SAML Assertion

TRAFFIC MANAGEMENT
Quota
Spike Arrest
Concurrent Rate Limit
Response Cache
Lookup Cache
Populate Cache
Invalidate Cache
Reset Quota

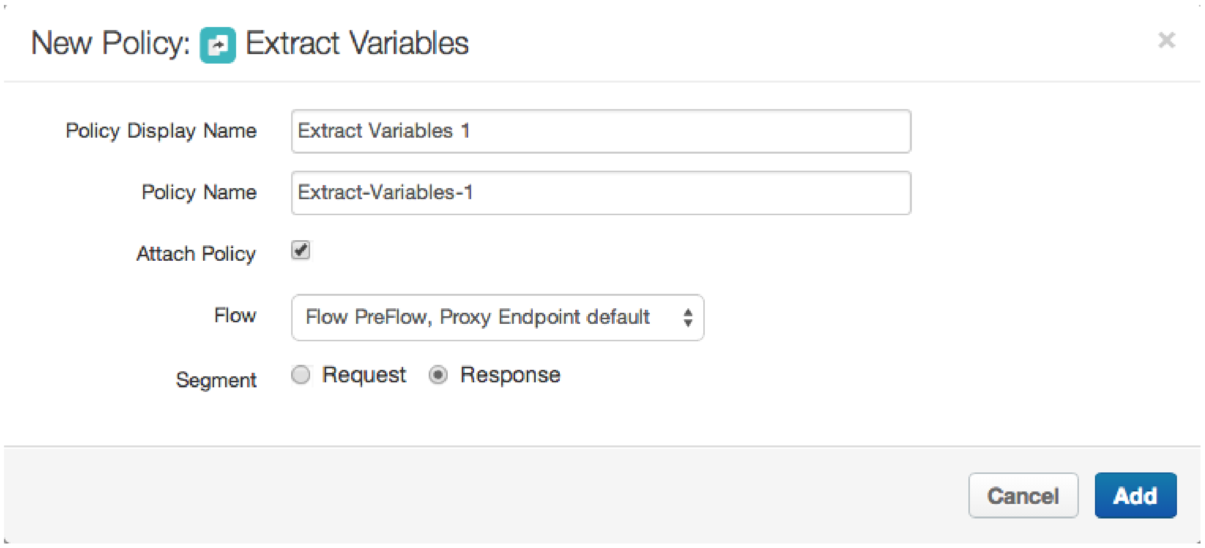
MEDIATION
JSON to XML
XML to JSON
Raise Fault
XSL Transform
SOAP Message Validation
Assign Message
Extract Variables
Access Entity
Key Value Map Operations

EXTENSION
Java Callout
Python
JavaScript
Service Callout
Statistics Collector
Message Logging

6. Now we will enhance the proxy to start collecting some statistics from the response payload. As part of this we will add a couple of policies including a Statistics Collector to the existing proxy that we have just created. Lets add a policy

   a. Go back to the API Management UI and make sure you are still on the same proxy. (If not click on APIs\API Proxies from the top menu and select the specific proxy). Click on the "Develop" button

   b. Click on "New Policy" and select "Extract Variables" from the "Mediation" category.


New Policy: Extract Variables

Policy Display Name	Extract Variables 1
Policy Name	Extract-Variables-1
Attach Policy	✓
Flow	Flow PreFlow, Proxy Endpoint default
Segment	Request [✓ Response]

Attach it to the "Response" segment of the Proxy Post Flow. Click on the "Add" button
Make sure the XML snippet in the bottom window looks like this:

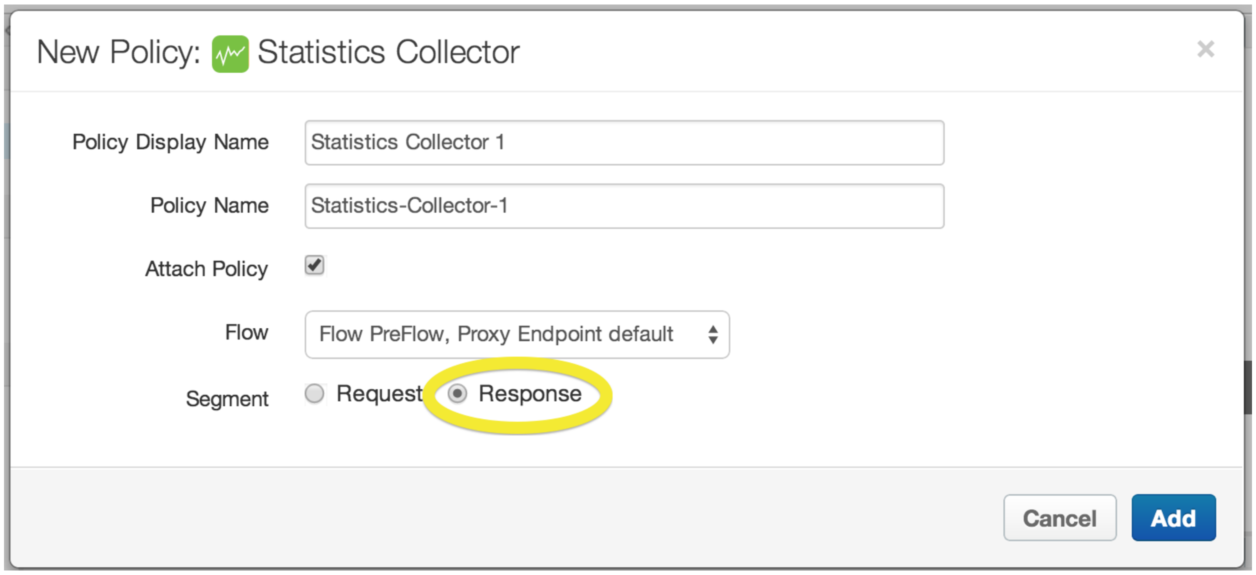
```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ExtractVariables async="false" continueOnError="false"
enabled="true" name="Extract-Variables-1">
     <DisplayName>Extract Variables 1</DisplayName>
     <Source clearPayload="false">response</Source>
     <FaultRules/>
     <Properties/>
     <JSONPayload>
          <Variable name="city_name">
          <JSONPath>$.name</JSONPath>
          </Variable>
     </JSONPayload>
</ExtractVariables>
```

You can also copy/paste the xml snippet from here into your policy editor.
(`https://gist.githubusercontent.com/prithpal/4d06b581423854 4f6eae/raw/7e08121cd190d20295f98dbf8206815f38620639/API%20 Services%20-%20Creating%20an%20API%20Proxy%20-%20Extract%2 0Variables`)

The JSONPath expression above extracts the name of the city from the response returned the OpenWeatherMap API and assigns it to a variable named "city_name". You can also extract similarly from an XML payload by providing XPATH expression.

c. Click on "New Policy" and select "Statistics Collector" from the "Extension" category. Attach it to the "Response" segment of the Proxy Post Flow.

Statistics Collector - Enables you to collect statistical data for messages processed in a flow, such as product ID, price, REST action, client and target URL, and message length (predefined flow variables as well as custom variables). The data is then provided to the analytics server, which analyzes the statistics and generates reports.


New Policy: Statistics Collector

Policy Display Name: Statistics Collector 1

Policy Name: Statistics-Collector-1

Attach Policy: ✓

Flow: Flow PreFlow, Proxy Endpoint default

Segment: Request [✓ Response]

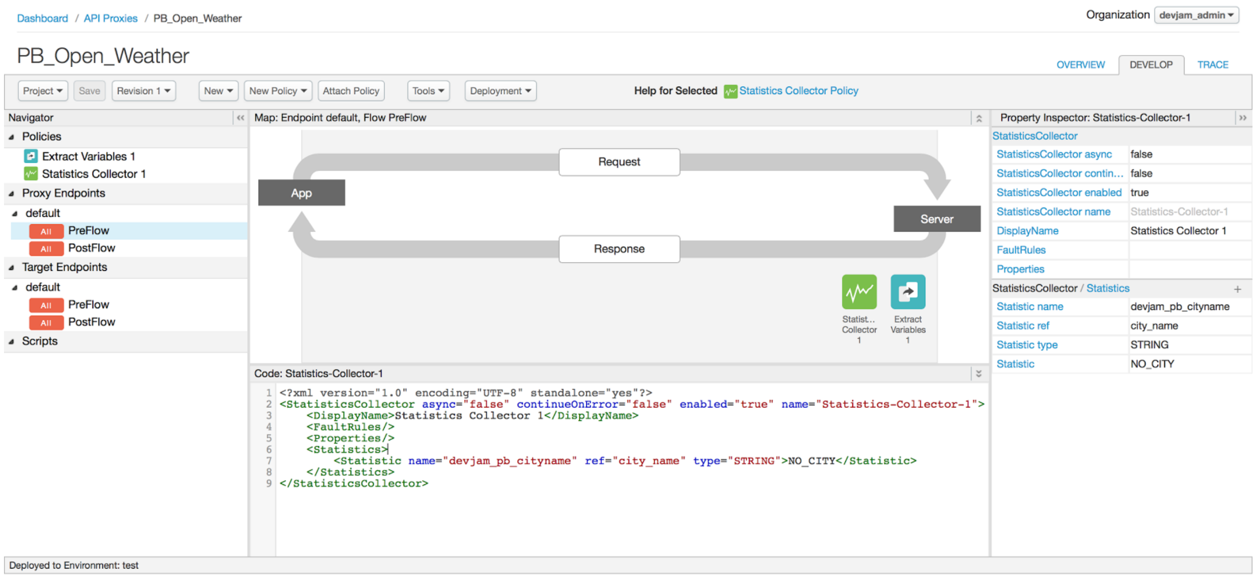
Click on the “Add” button Make sure the XML snippet in the bottom window looks like this:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<StatisticsCollector async="false" continueOnError="false"
enabled="true" name="Statistics-Collector-1">
     <DisplayName>Statistics Collector 1</DisplayName>
     <FaultRules/>
     <Properties/>
     <Statistics>
          <Statistic name="devjam_{your
          initials}_cityname" ref="city_name"
          type="STRING">NO_CITY</Statistic>
     </Statistics>
</StatisticsCollector>
```

* If you copy the XML snippet from the link, remember to change the statistic name to use your initials (as mentioned above) instead of the one in the snippet.

You can also copy/paste the xml snippet from here into your policy editor. (`https://gist.githubusercontent.com/prithpal/0f5ec55c2bf13f9e5985/raw/58a24bb7f6885521a6a87e742b0c8ae107380e16/API%20Services%20-%20Creating%20an%20API%20Proxy%20-%20Statistics%20Collector`)

d. Once the 2 policies have been added and you have clicked “Save”, you should see something similar to this:



Property Inspector: Statistics-Collector-1

StatisticsCollector	
StatisticsCollector async	false
StatisticsCollector contin...	false
StatisticsCollector enabled	true
StatisticsCollector name	Statistics-Collector-1
DisplayName	Statistics Collector 1
FaultRules	
Properties	

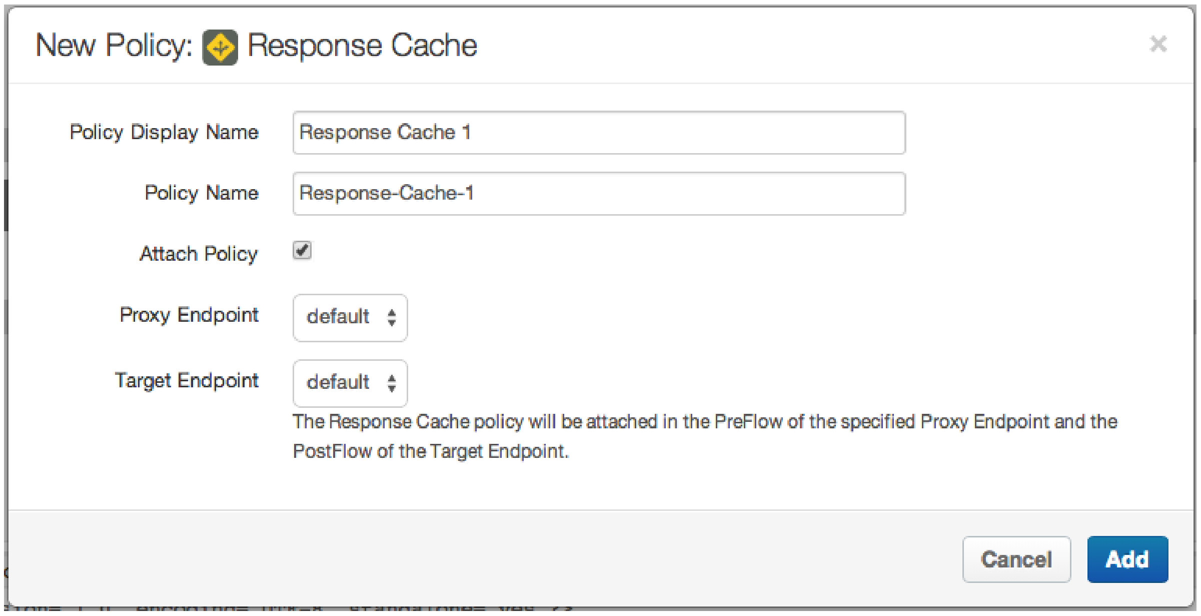
StatisticsCollector / Statistics	
Statistic name	devjam_pb_cityname
Statistic ref	city_name
Statistic type	STRING
Statistic	NO_CITY

Code: Statistics-Collector-1

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<StatisticsCollector async="false" continueOnError="false" enabled="true" name="Statistics-Collector-1">
    <DisplayName>Statistics Collector 1</DisplayName>
    <FaultRules/>
    <Properties/>
    <Statistics>
        <Statistic name="devjam_pb_cityname" ref="city_name" type="STRING">NO_CITY</Statistic>
    </Statistics>
</StatisticsCollector>
```

7. Lets go ahead and add a Response Cache policy. ResponseCache can be configured to save and periodically refresh copies of response messages. As apps make requests to the same URI, Apigee Edge can be configured to return cached responses, rather than forwarding those requests to the backend server.

    a. Click on “New Policy” and select “Response Cache” from the “Traffic Management” category. Make sure the “Attach Policy” checkbox is selected.


New Policy: Response Cache

Policy Display Name	Response Cache 1
Policy Name	Response-Cache-1
Attach Policy	✓
Proxy Endpoint	default
Target Endpoint	default

The Response Cache policy will be attached in the PreFlow of the specified Proxy Endpoint and the PostFlow of the Target Endpoint.

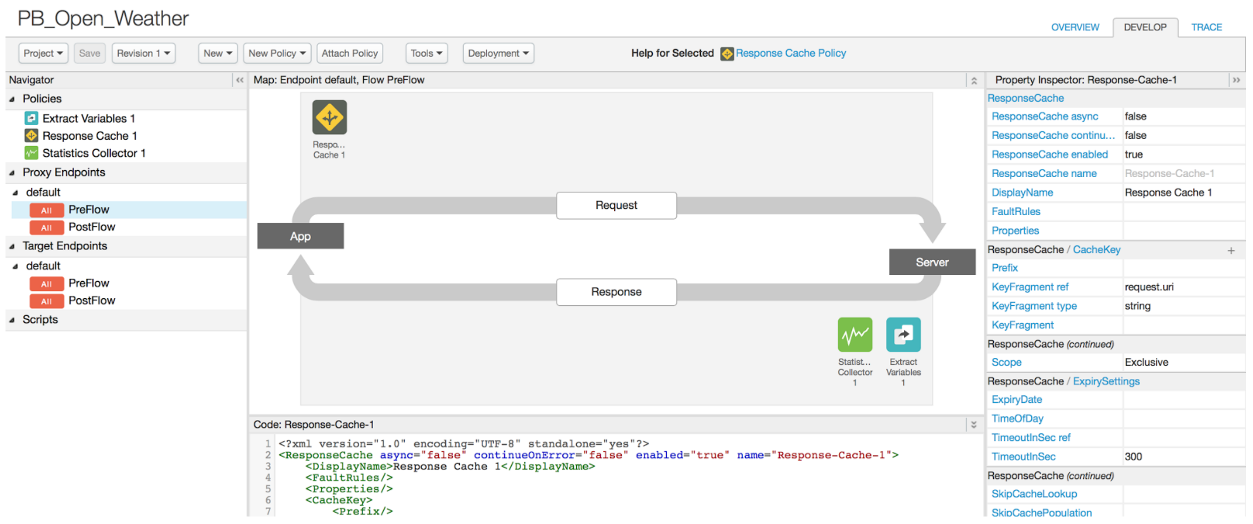
b. Click on the Add + button and make sure the XML snippet looks like this:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ResponseCache async="false" continueOnError="false"
enabled="true" name="Response-Cache-1">
    <DisplayName>Response Cache 1</DisplayName>
    <FaultRules/>
    <Properties/>
    <CacheKey>
        <Prefix/>
        <KeyFragment ref="request.uri" type="string"/>
    </CacheKey>
    <Scope>Exclusive</Scope>
    <ExpirySettings>
        <ExpiryDate/>
        <TimeOfDay/>
        <TimeoutInSec ref="">60</TimeoutInSec>
    </ExpirySettings>
    <SkipCacheLookup/>
    <SkipCachePopulation/>
</ResponseCache>
```

You can also copy/paste the xml snippet from here into your policy editor. (`https://gist.githubusercontent.com/prithpal/354e972a50450861ec0e/raw/a3c1c0ed9203f60936777f5717393c4ed60523f7/API%20Services%20-%20Creating%20an%20API%20Proxy%20-%20Response%20Cache`)

We have changed the timeout property on the Response Cache policy to 60 seconds for demonstration purposes so the cache expires sooner. Your cache expiration setting may be different in production scenarios.

c. Click on "Save" and you should see something similar to this:



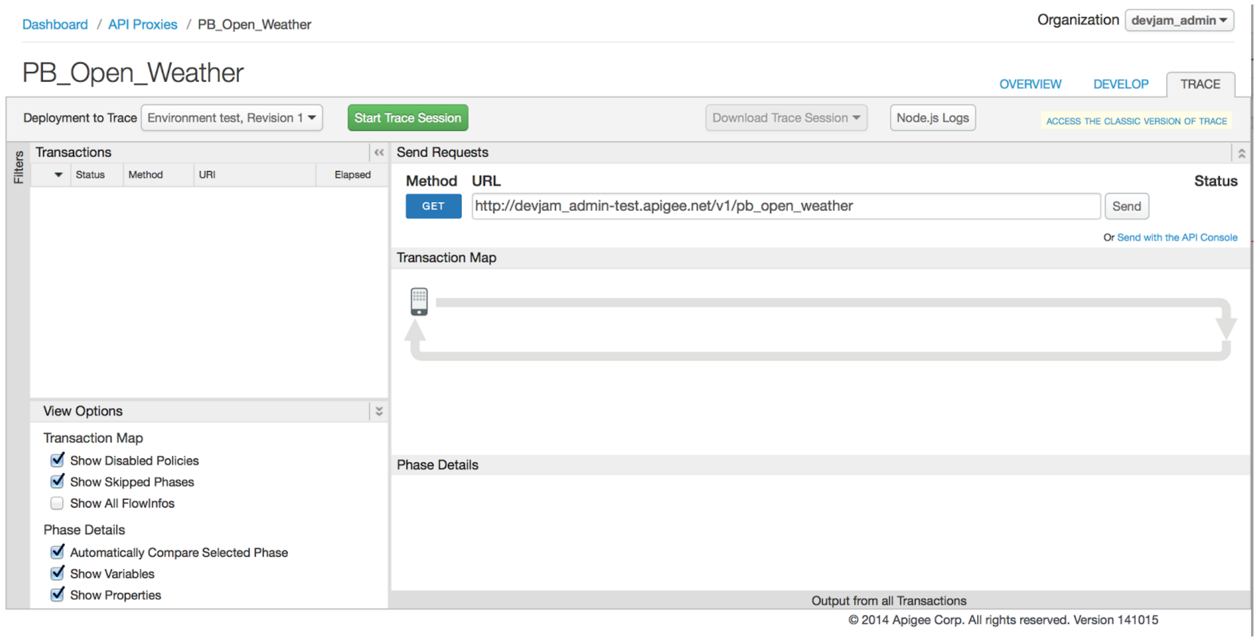
```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ResponseCache async="false" continueOnError="false" enabled="true" name="Response-Cache-1">
    <DisplayName>Response Cache 1</DisplayName>
    <FaultRules/>
    <Properties/>
    <CacheKey>
        <Prefix/>
```

Property Inspector: Response-Cache-1

ResponseCache	
ResponseCache async	false
ResponseCache continu...	false
ResponseCache enabled	true
ResponseCache name	Response-Cache-1
DisplayName	Response Cache 1
FaultRules	
Properties	
ResponseCache / CacheKey	
Prefix	
KeyFragment ref	request.uri
KeyFragment type	string
KeyFragment	
ResponseCache *(continued)*	
Scope	Exclusive
ResponseCache / ExpirySettings	
ExpiryDate	
TimeOfDay	
TimeoutInSec ref	
TimeoutInSec	300
ResponseCache *(continued)*	
SkipCacheLookup	
SkipCachePopulation	

8. Click on the “Trace” button (top right)

    a. You should see something like this:



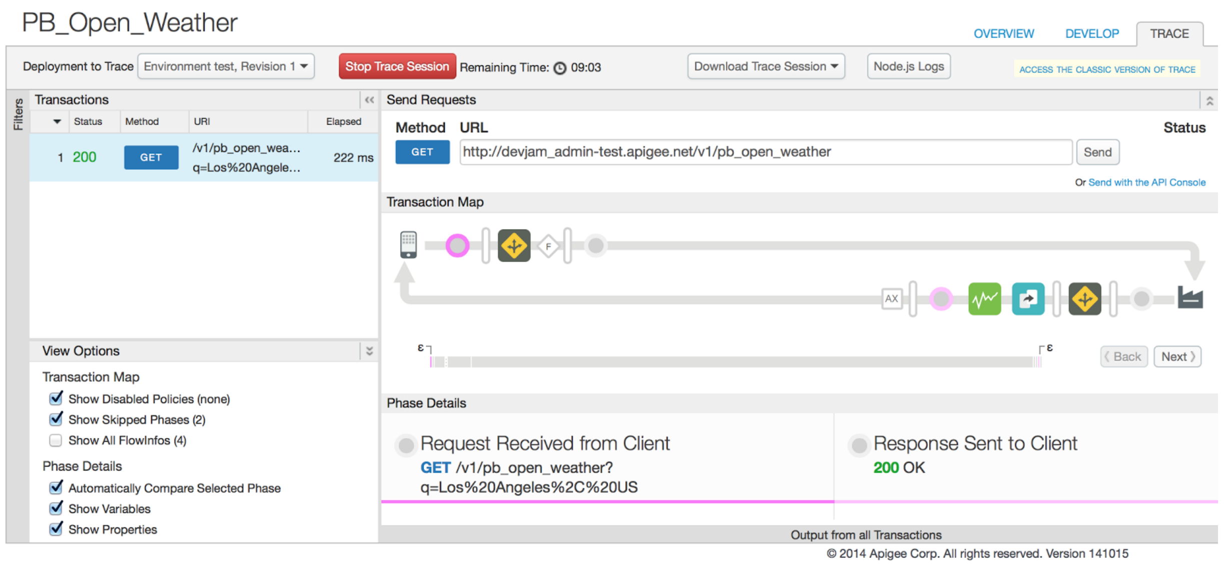
9. Tracing - [Trace](#) is a tool for troubleshooting and monitoring API proxies running on Apigee Edge.

    Trace lets you probe the details of each step through an API proxy flow. In this step you will use the trace tool in the background to see the request(s) and details in each step of the proxy to help troubleshoot errors if any and to see what is happening with the request every step of the way.

a. Click on the “Start Trace Session” green button. The button should turn red and have the text displayed as “Stop Trace Session”.

b. In another browser tab or window open the POSTMAN tool and invoke the API end-point (`http://{`**`org_name`**`}-{`**`env_name`**`}.apigee.net/v1/`**`{your initials}`**`_open_weather?q=Los Angeles`) as before.

c. Come back to the Management UI and see the trace window reflect details about the request we just sent to the proxy:


PB_Open_Weather

Deployment to Trace: Environment test, Revision 1
Stop Trace Session
Remaining Time: 09:03

Transactions

	Status	Method	URI	Elapsed
1	200	GET	/v1/pb_open_wea... q=Los%20Angele...	222 ms

Send Requests

Method: GET
URL: http://devjam_admin-test.apigee.net/v1/pb_open_weather

View Options

Transaction Map
- ☑ Show Disabled Policies (none)
- ☑ Show Skipped Phases (2)
- ☐ Show All FlowInfos (4)

Phase Details
- ☑ Automatically Compare Selected Phase
- ☑ Show Variables
- ☑ Show Properties

Phase Details

Request Received from Client
GET /v1/pb_open_weather?q=Los%20Angeles%2C%20US

Response Sent to Client
200 OK

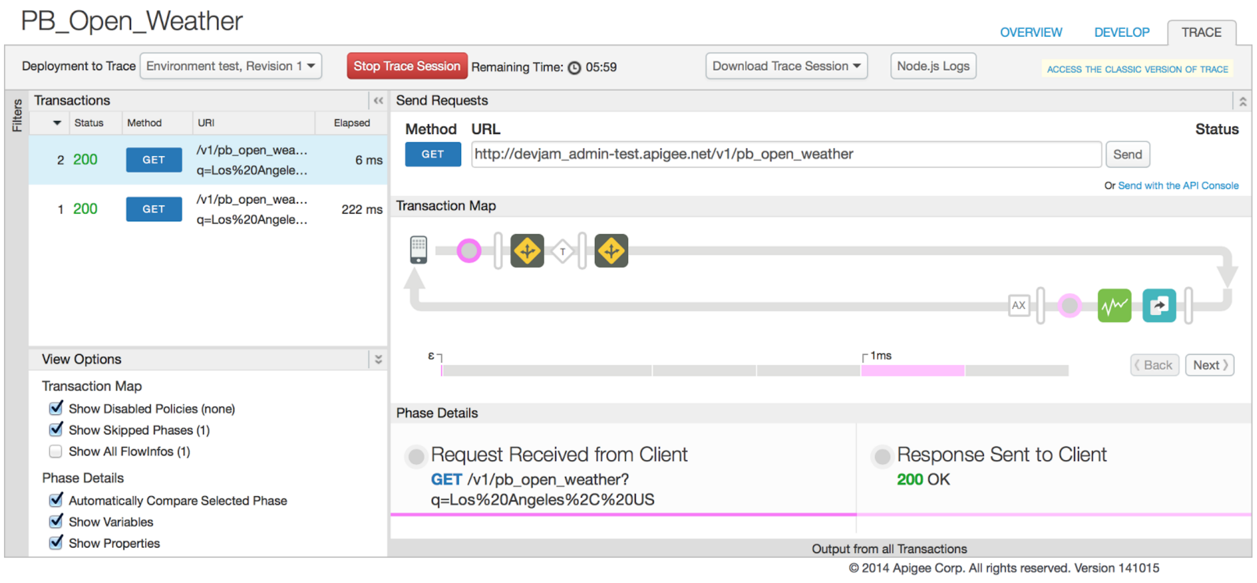
Output from all Transactions

© 2014 Apigee Corp. All rights reserved. Version 141015

d. Pay attention to the time taken for the request to process. Now lets see if our Response Cache policy kicks in.

e. Go back to the POSTMAN client and re-send the same request to the API proxy.

f. Go back to the Trace tool and you will see that this time the response was served from the cache (see the “T”which stands for true for a cache hit) as opposed to hitting the back-end target endpoint. Also see the response time reduce considerably as a result of caching.


PB_Open_Weather

Deployment to Trace: Environment test, Revision 1
Remaining Time: 05:59

Transactions

	Status	Method	URI	Elapsed
2	200	GET	/v1/pb_open_wea… q=Los%20Angele…	6 ms
1	200	GET	/v1/pb_open_wea… q=Los%20Angele…	222 ms

Send Requests

Method: GET
URL: http://devjam_admin-test.apigee.net/v1/pb_open_weather
Status

Transaction Map

View Options

Transaction Map
- [✓] Show Disabled Policies (none)
- [✓] Show Skipped Phases (1)
- [ ] Show All FlowInfos (1)

Phase Details
- [✓] Automatically Compare Selected Phase
- [✓] Show Variables
- [✓] Show Properties

Phase Details

Request Received from Client
GET /v1/pb_open_weather?q=Los%20Angeles%2C%20US

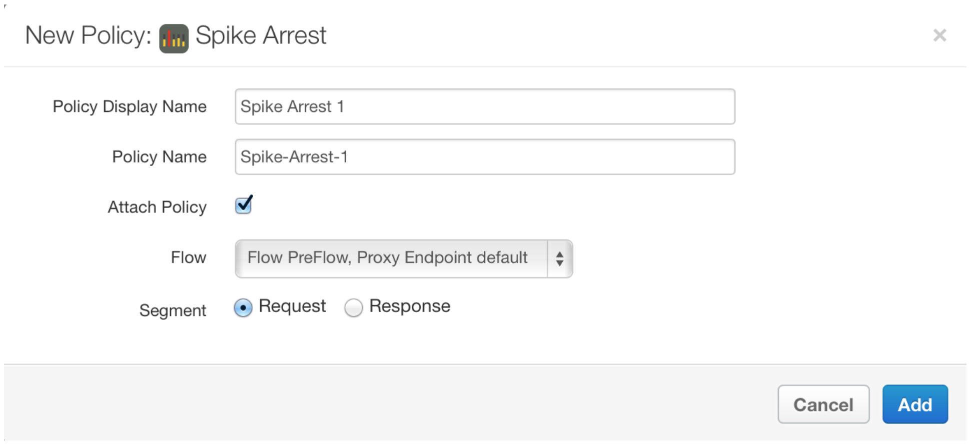
Response Sent to Client
200 OK

Output from all Transactions

© 2014 Apigee Corp. All rights reserved. Version 141015

10. Next we will add a 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 lags and downtime.

    a. Go back to the "Develop" tab on the proxy editor. Click on "New Policy" and add a new "Spike Arrest" policy to the request segment of the Pre-Flow.


New Policy: Spike Arrest

Policy Display Name	Spike Arrest 1
Policy Name	Spike-Arrest-1
Attach Policy	✓
Flow	Flow PreFlow, Proxy Endpoint default
Segment	[✓ Request] Response

Click on the Add button and make sure that the XML snippet looks like this:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<SpikeArrest async="false" continueOnError="false"
enabled="true" name="Spike-Arrest-1">
     <DisplayName>Spike Arrest 1</DisplayName>
```

```
    <Rate>10pm</Rate>
</SpikeArrest>
```

You can also copy/paste the xml snippet from here into your policy editor. (`https://gist.githubusercontent.com/prithpal/bf291171b37be7169287/raw/ce49ce2dfe73d58a74d4cd46d9a2478251d30d2e/API%20Services%20-%20Creating%20a%20Proxy%20-%20Spike%20Arrest`)

The runtime Spike Arrest behavior differs from what you might expect to see from the literal per-minute or per-second values you enter.

For example, say you enter a rate of 20pm (20 requests per minute). In testing, you might think you could send 20 requests in 1 second, as long as they came within a minute. But that's not how the policy enforces the setting. If you think about it, 20 requests inside a 1-second period could be considered a mini spike in some environments.

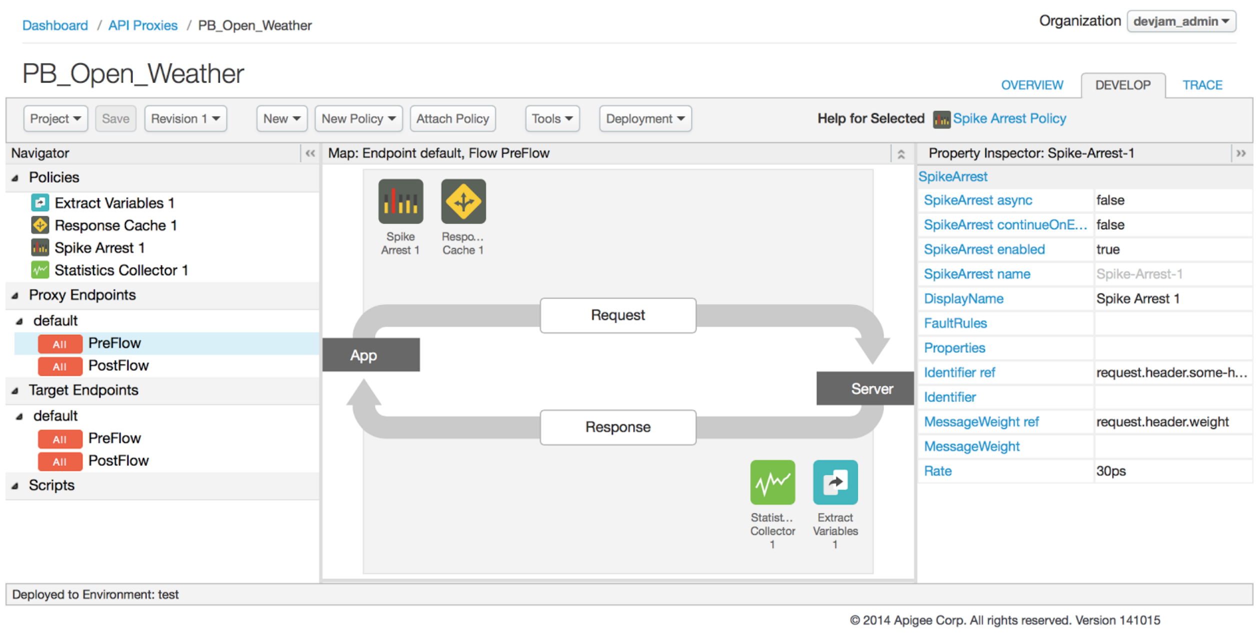
What actually happens, then? To prevent spike-like behavior, Spike Arrest smooths the allowed traffic by dividing your settings into smaller intervals:

Per-minute rates get smoothed into requests allowed intervals of seconds.
For example, 10pm gets smoothed like this:
60 seconds (1 minute) / 10pm = 6-second intervals, or about 1 request allowed every 6 seconds. A second request inside of 6 seconds will fail. Also, a 11th request within a minute will fail.

Per-second rates get smoothed into requests allowed in intervals of milliseconds.
For example, 10ps gets smoothed like this:
1000 milliseconds (1 second) / 10ps = 100-millisecond intervals, or about 1 request allowed every 100 milliseconds . A second request inside of 100ms will fail. Also, an 11th request within a second will fail.

Visit this link “How it works” section for additional details.

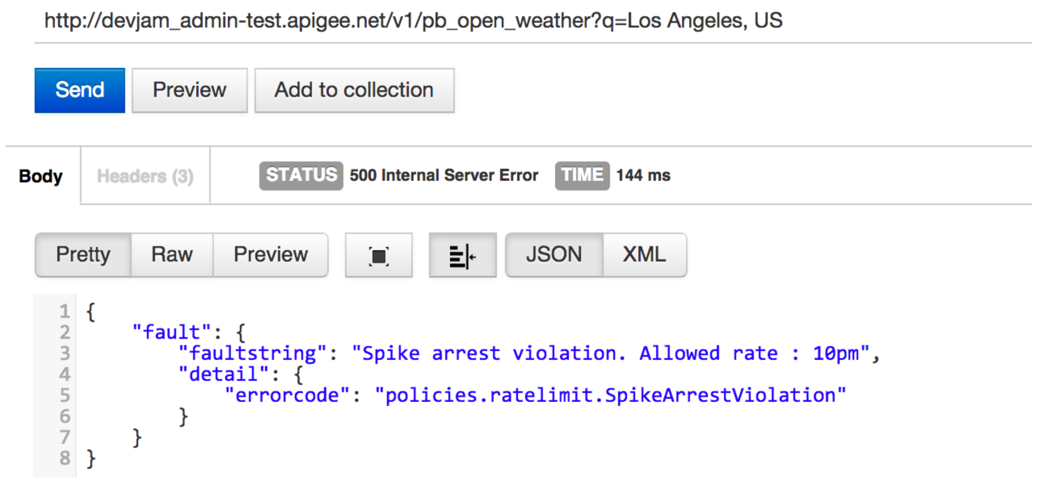
b. Now drag and move the “Spike Arrest” policy to before the “Response Cache” policy. (*Best Practice - The Spike Arrest policy should normally be the first policy in the request pipeline as a best practice*). You should see the screen like this:



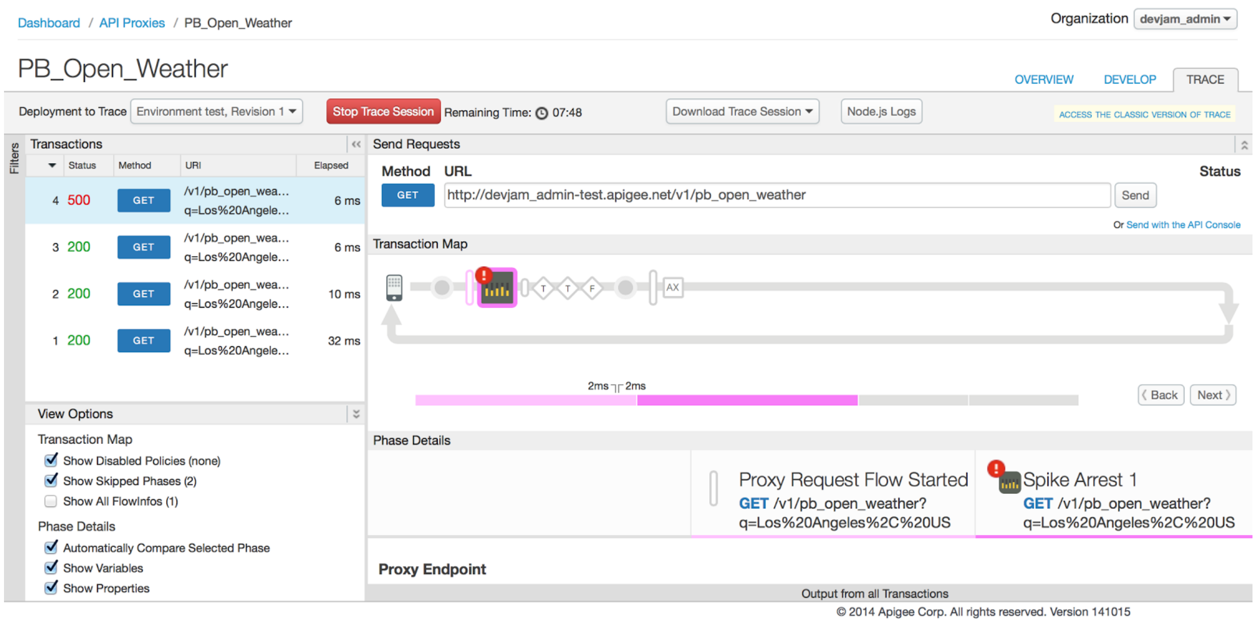
SpikeArrest	
SpikeArrest async	false
SpikeArrest continueOnE...	false
SpikeArrest enabled	true
SpikeArrest name	Spike-Arrest-1
DisplayName	Spike Arrest 1
FaultRules	
Properties	
Identifier ref	request.header.some-h...
Identifier	
MessageWeight ref	request.header.weight
MessageWeight	
Rate	30ps

c. Now lets go and put the Spike Arrest policy to test. Click on the “Trace Section” and click the “Start Trace Session” button so that it turns red in color.

d. Go to POSTMAN and hit the proxy endpoint a few times, you should see that in sometime the Spike Arrest policy kicks in and prevents additional requests from hitting the backend target endpoint. You can see the trace tool which captures the Spike Arrest policy violation event.


http://devjam_admin-test.apigee.net/v1/pb_open_weather?q=Los Angeles, US
Send
Preview
Add to collection
Body
Headers (3)
STATUS 500 Internal Server Error
TIME 144 ms
Pretty
Raw
Preview
JSON
XML

```
{
    "fault": {
        "faultstring": "Spike arrest violation. Allowed rate : 10pm",
        "detail": {
            "errorcode": "policies.ratelimit.SpikeArrestViolation"
        }
    }
}
```


Dashboard / API Proxies / PB_Open_Weather

Organization devjam_admin

PB_Open_Weather

OVERVIEW DEVELOP TRACE

Deployment to Trace: Environment test, Revision 1 — Stop Trace Session — Remaining Time: 07:48 — Download Trace Session — Node.js Logs — ACCESS THE CLASSIC VERSION OF TRACE

Transactions

	Status	Method	URI	Elapsed
4	500	GET	/v1/pb_open_wea... q=Los%20Angele...	6 ms
3	200	GET	/v1/pb_open_wea... q=Los%20Angele...	6 ms
2	200	GET	/v1/pb_open_wea... q=Los%20Angele...	10 ms
1	200	GET	/v1/pb_open_wea... q=Los%20Angele...	32 ms

Send Requests

Method: GET — URL: http://devjam_admin-test.apigee.net/v1/pb_open_weather — Send

Or Send with the API Console

Transaction Map

2ms 2ms

View Options

Transaction Map
- [✓] Show Disabled Policies (none)
- [✓] Show Skipped Phases (2)
- [ ] Show All FlowInfos (1)

Phase Details
- [✓] Automatically Compare Selected Phase
- [✓] Show Variables
- [✓] Show Properties

Phase Details

Proxy Request Flow Started
GET /v1/pb_open_weather?q=Los%20Angeles%2C%20US

Spike Arrest 1
GET /v1/pb_open_weather?q=Los%20Angeles%2C%20US

Proxy Endpoint

Output from all Transactions

© 2014 Apigee Corp. All rights reserved. Version 141015

Summary

In this lab you created a simple API proxy and added a few policies including Extract Variables, Response Caching, Statistics Collector and Spike Arrest. You used the trace tool to troubleshoot how the proxy responds to a request. Please visit the documentation to see the different kinds of policies you can decorate your proxies with.