Policy Mashup (Call Multiple APIs From Apigee Edge console)-

Step-1

Here we used two backend services-

- Google Geocoding web service
- Google Elevation web service

First you have to assign a backend service during creation of API Proxy -

http://maps.googleapis.com/maps/api/elevation/xml

It exposes an API that takes two query parameters:

- country: A two-letter country code
- postalcode: A postal code valid in that country

It returns a JSON result that includes the geocoded location for the center of that postal code, and the elevation at that location.

It uses a series of policies in order to accomplish this:

1. An AssignMessage policy to generate the request for the geocoding web service
2. A ServiceCallout policy to call it
3. An ExtractVariables policy to parse the response and extract the latitude and longitude
4. An AssignMesage policy to set the parameters for the elevation service
5. An ExtractVariables policy to parse the elevation response

6. A Javascript policy to generate the response JSON from the variables set by the previous policies.

7. A StatisticsCollector policy to send the values of "country" and "postalcode" to Analytics for use in a custom report.

Step-2

As mentioned above First An AssignMessage policy to generate the request for the geocoding web service-

Code-

```
<AssignMessage name="GenerateGeocodingRequest">
  <!-- Create a message to send to the geocoding service. It's a GET so all
       it needs in this case are query parameters -->
  <AssignTo createNew="true" type="request">GeocodingRequest</AssignTo>
  <Set>
            <QueryParams>
      <QueryParam name="address">{request.queryparam.postalcode}</QueryParam>
      <QueryParam name="region">{request.queryparam.country}</QueryParam>
      <QueryParam name="sensor">false</QueryParam>
    </QueryParams>
    <Verb>GET</Verb>
  </Set>
  <!-- Set variables for use in the final response -->
  <AssignVariable>
    <Name>PostalCode</Name>
    <Ref>request.queryparam.postalcode</Ref>
  </AssignVariable>
  <AssignVariable>
    <Name>Country</Name>
    <Ref>request.queryparam.country</Ref>
  </AssignVariable>
</AssignMessage>
```

- then, A ServiceCallout policy to call it-

Code-

```
<ServiceCallout name="ExecuteGeocodingRequest">
  <!-- Send the message we just made to the target, and save the result -->
  <Request variable="GeocodingRequest"/>
  <Response>GeocodingResponse</Response>
```

```
  <HTTPTargetConnection>
    <URL>http://maps.googleapis.com/maps/api/geocode/json</URL>
  </HTTPTargetConnection>
</ServiceCallout>
```

- then, An ExtractVariables policy to parse the response and extract the latitude and longitude-

Code-

```
<ServiceCallout name="ExecuteGeocodingRequest">
  <!-- Send the message we just made to the target, and save the result -->
  <Request variable="GeocodingRequest"/>
  <Response>GeocodingResponse</Response>
  <HTTPTargetConnection>
    <URL>http://maps.googleapis.com/maps/api/geocode/json</URL>
  </HTTPTargetConnection>
</ServiceCallout>
```

- then, An AssignMesage policy to set the parameters for the elevation service

Code-

```
<AssignMessage name="AssignElevationParameters">
  <!-- Remove query parameters from the original request and add the ones that the
elevation
       service will require -->
  <Remove>
    <QueryParams>
      <QueryParam name="country"/>
      <QueryParam name="postalcode"/>
    </QueryParams>
  </Remove>
  <Set>
    <QueryParams>
      <QueryParam name="locations">{geocoderesponse.latitude},
{geocoderesponse.longitude}</QueryParam>
      <QueryParam name="sensor">false</QueryParam>
    </QueryParams>
  </Set>
</AssignMessage>
```

- then, An ExtractVariables policy to parse the elevation response

Code-

```
<ExtractVariables name="ParseElevationResponse">
  <!-- Parse the XML that we got back from the elevation service using XPath.
       (That service could have returned JSON too but this is a sample, right?) -->
  <VariablePrefix>elevationresponse</VariablePrefix>
  <XMLPayload>
    <Variable name="elevation" type="float">
      <XPath>/ElevationResponse/result[1]/elevation/text()</XPath>
    </Variable>
  </XMLPayload>
</ExtractVariables>
```

- then, A Javascript policy to generate the response JSON from the variables set by the previous policies-

Code-

```
<!-- This policy references the JavaScript under /resources/jsc. Use this policy to attach
the JavaScript to Flow in the ProxyEndpoint configuration (/proxies/default.xml). -->
<Javascript name="GenerateResponse" timeout="10000">
  <ResourceURL>jsc://GenerateResponse.js</ResourceURL>
</Javascript>
```

- then, A StatisticsCollector policy to send the values of "country" and "postalcode" to Analytics for use in a custom report.

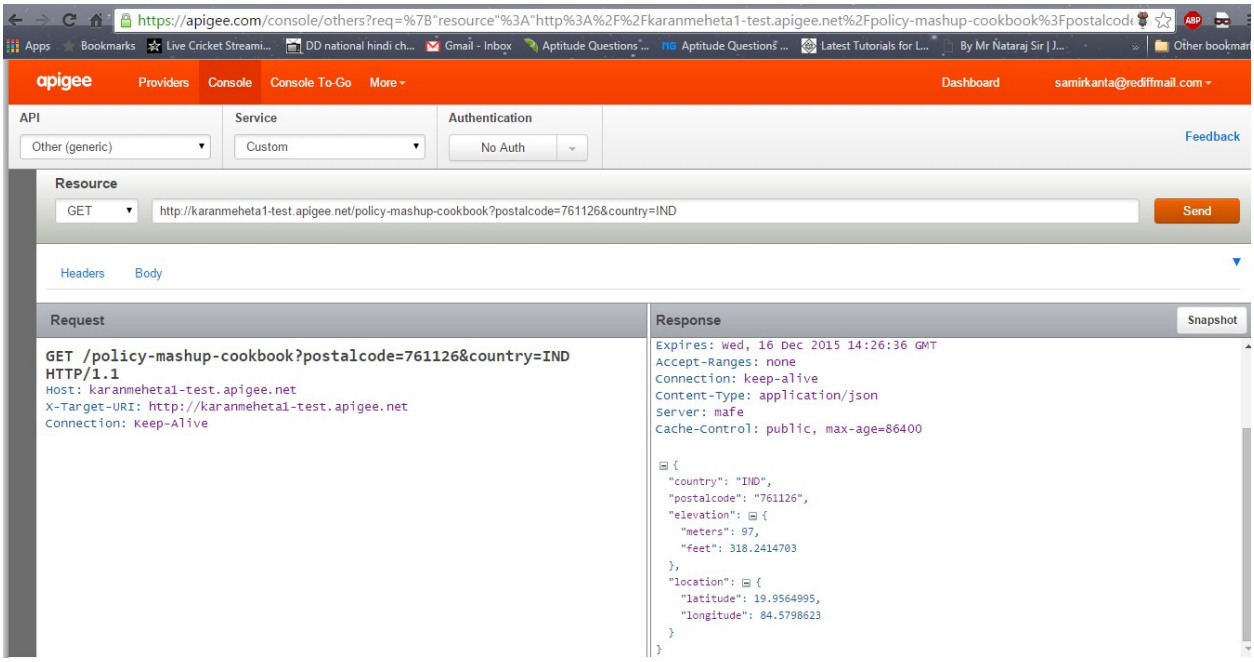
Step-3

- After Applying all the Policies into your API Proxy Application now you have to pass the values of "country" and "postalcode" to Analytics for use in a custom report

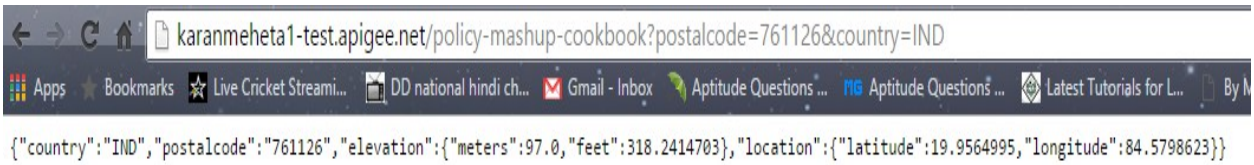
The URL we have to pass either from Browser (or) from Apigee Console-

http://karanmeheta1-test.apigee.net/policy-mashup-cookbook?postalcode=761126&country=IND

Response From Apigee Console-



Response From Browser-



{"country":"IND","postalcode":"761126","elevation":{"meters":97.0,"feet":318.2414703},"location":{"latitude":19.9564995,"longitude":84.5798623}}

Response-

HTTP/1.1 200 OK

X-Frame-Options:

SAMEORIGIN

Vary:

Accept-Encoding

Transfer-Encoding:

chunked

Date:

Tue, 15 Dec 2015 14:26:36 GMT

X-XSS-Protection:

1; mode=block
Expires:
Wed, 16 Dec 2015 14:26:36 GMT
Accept-Ranges:
none
Connection:
keep-alive
Content-Type:
application/json
Server:
mafe
Cache-Control:
public, max-age=86400


```
{
  "country": "IND",
  "postalcode": "761126",
  "elevation": {
    "meters": 97,
    "feet": 318.2414703
  },
  "location": {
    "latitude": 19.9564995,
    "longitude": 84.5798623
  }
}
```