

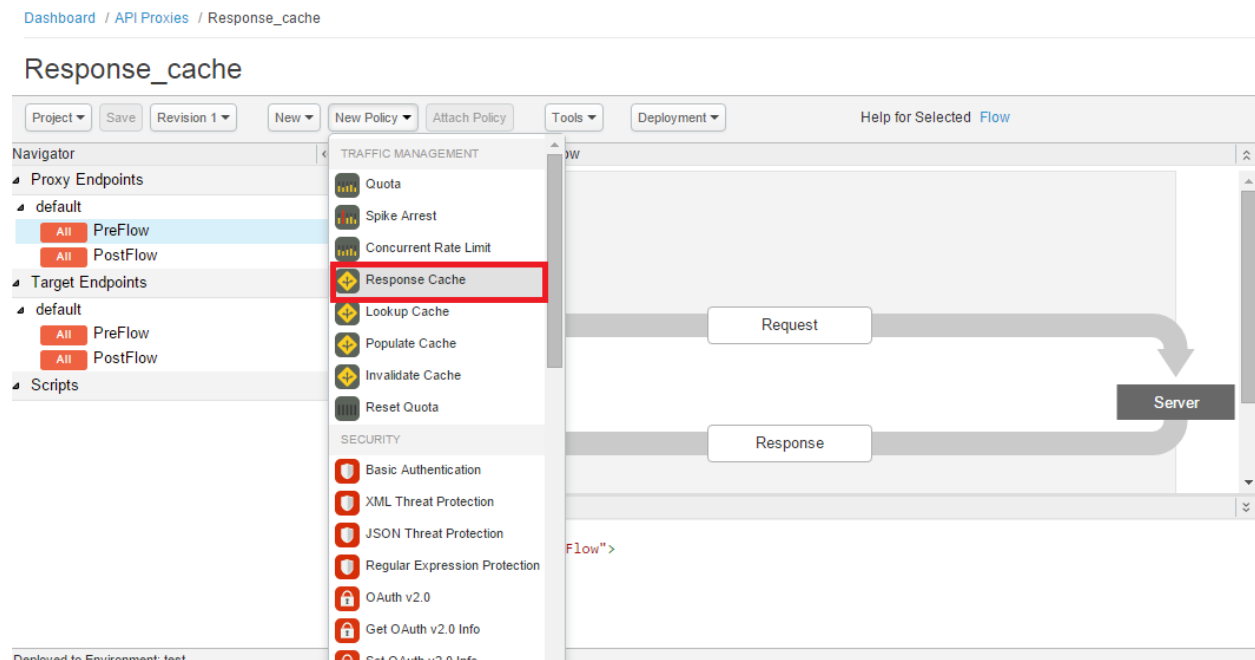
Response Cache policy:

Response Cache, Caches data from a backend resource, reducing the number of requests to the resource. As apps make requests to the same URI, you can use this policy to return cached responses instead of forwarding those requests to the backend server. The ResponseCache policy can improve your API's performance through reduced latency and network traffic.


You'll likely find ResponseCache most useful when backend data used by your API is updated only periodically. For example, imagine you have an API that exposes weather report data refreshed only every ten minutes. By using ResponseCache to return cached responses between refreshes, you can decrease the number of requests reaching the backend. This also reduces the number of network hops.

Step 1:

First Select Response Cache from the list of Policies from Apigee Edge Console



Then fill the required fields as per your requirement

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.

Cancel

Add


- Then Click on “Add” button.
- Click on “Save” to save the Current Revision.

Step 2:

After adding Response Cache Policy into Apigee Edge Console the Console will be-

Dashboard / API Proxies / Response_cache

Response_cache

Project ▾ Save Revision 1 ▾ New ▾ New Policy ▾ Attach Policy Tools ▾ Deployment ▾
Help for Selected  Response Cache Policy


Policies
Response Cache 1

Proxy Endpoints
default
All PreFlow
All PostFlow

Target Endpoints
default
All PreFlow
All PostFlow

Scripts

Map: Endpoint default, Flow PreFlow


Resp...
Cache 1

Request

Code: Response-Cache-1

```

1 <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
2 <ResponseCache async="false" continueOnError="false" enabled="true" name="Response-Cache-1">
3   <DisplayName>Response Cache 1</DisplayName>
4   <Properties/>
5   <CacheKey>
6     <Prefix/>
7     <KeyFragment ref="request.uri" type="string"/>
8   </CacheKey>
9   <Scope>Exclusive</Scope>
10  <ExpirySettings>
11    <ExpiryDate/>
12    <TimeOfDay/>
13    <TimeoutInSec ref="">3600</TimeoutInSec>
14  </ExpirySettings>
15  <SkipCacheLookup/>
16  <SkipCachePopulation/>
17 </ResponseCache>

```

Code:

```
<?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>
  <Properties/>
  <CacheKey>
    <Prefix/>
    <KeyFragment ref="request.uri" type="string"/>
  </CacheKey>
  <Scope>Exclusive</Scope>
  <ExpirySettings>
    <ExpiryDate/>
    <TimeOfDay/>
    <TimeoutInSec ref="">3600</TimeoutInSec>
  </ExpirySettings>
  <SkipCacheLookup/>
  <SkipCachePopulation/>
</ResponseCache>
```

Step 3:

- Before applying Response Cache the Response is like this-

Response_cache

Deployment to Trace Environment test, Revision 2					Stop
Filters	Transactions				
	Status	Method	URI	Elapsed	
7	200	GET	/v1/response_...	10 ms	
6	200	GET	/v1/response_...	11 ms	
5	200	GET	/v1/response_...	43 ms	
4	200	GET	/v1/response_...	51 ms	
3	200	GET	/v1/response_...	8 ms	
2	200	GET	/v1/response_...	43 ms	
1	200	GET	/v1/response_...	480 ms	
View Options					

- After Applying Response Cache the response is-

Response_cache

Deployment to Trace: Environment test, Revision 2

Stop Trace Session Remaining Time: 08:33

Download Trace Session Node.js Logs

Transactions

Filters	Status	Method	URI	Elapsed
7	200	GET	/v1/response_...	4 ms
6	200	GET	/v1/response_...	4 ms
5	200	GET	/v1/response_...	4 ms
4	200	GET	/v1/response_...	5 ms
3	200	GET	/v1/response_...	4 ms
2	200	GET	/v1/response_...	5 ms
1	200	GET	/v1/response_...	11 ms

Send Requests

Method: GET URL: http://karanmeheta1-test.apigee.net/v1/response_cache Status: 200

Transaction Map

Back Next

- For the first time the response came from the backend that's why it took time and also the response is stored on the response cache. But when the user requested for the second time it didn't went to the backend again it got the response from the Response Cache itself , that's why it took very less time to get the response.

Response_cache

Deployment to Trace: Environment test, Revision 2

Stop Trace Session

Transactions

Filters	Status	Method	URI	Elapsed
7	200	GET	/v1/response_...	4 ms
6	200	GET	/v1/response_...	4 ms
5	200	GET	/v1/response_...	4 ms
4	200	GET	/v1/response_...	5 ms
3	200	GET	/v1/response_...	4 ms
2	200	GET	/v1/response_...	5 ms
1	200	GET	/v1/response_...	11 ms

View Options